

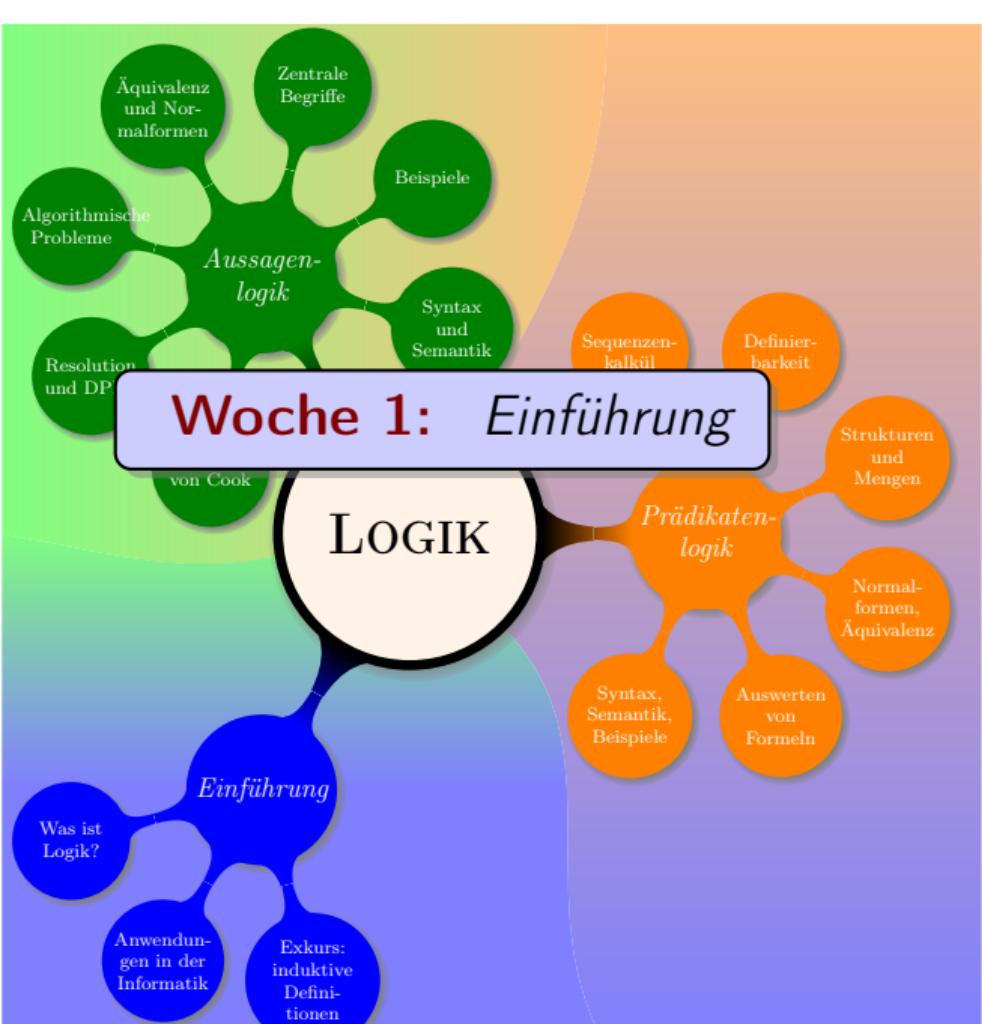
Logik

Stephan Kreutzer

Lehrstuhl für Logik und Semantik
Technische Universität Berlin

WS 2022/2023





Woche 1: Einführung

Nov.	17	18	19	20	21	22	23	<i>Einführung</i>
	24	25	26	27	28	29	30	<i>Aussagenlogik</i>
	31	1	2	3	4	5	6	<i>Normalformen</i>
	7	8	9	10	11	12	13	<i>Resolution</i>
Dez.	14	15	16	17	18	19	20	<i>Kompaktheit</i>
	21	22	23	24	25	26	27	<i>DPLL, Satz von Cook</i>
	28	29	30	1	2	3	4	<i>Strukturen und FO</i>
	5	6	7	8	9	10	11	<i>Prädikatenlogik</i>
	12	13	14	15	16	17	18	<i>Komplexität von FO</i>
Jan.	19	20	21	22	23	24	25	<i>Weihnachten</i>
	26	27	28	29	30	31	1	<i>Neujahr</i>
	2	3	4	5	6	7	8	<i>Normalformen</i>
	9	10	11	12	13	14	15	<i>Definierbarkeit</i>
	16	17	18	19	20	21	22	<i>EF-Spiele</i>
Feb.	23	24	25	26	27	28	29	<i>EF-Spiele</i>
	30	31	1	2	3	4	5	<i>Sequenzenskalkül AL</i>
	6	7	8	9	10	11	12	<i>Sequenzenskalkül FO</i>
	13	14	15	16	17	18	19	<i>Ausblick</i>

1.1 Was ist Logik?

Logik?

Ursprünge der Logik. Das Wort *Logik* stammt aus dem Altgriechischen „Kunst des Denkens“ oder „Kunst des Argumentierens“.

Beispiel.

- Alle Vögel können fliegen.
- Tweety ist ein Vogel.
- Also kann Tweety fliegen.

Stimmt das Argument?

Wir können das Argument auf zwei Ebenen diskutieren: **Inhaltlich**.

Können alle Vögel fliegen? Wer oder was ist Tweety? Are birds real?

Logik?

Ursprünge der Logik. Das Wort *Logik* stammt aus dem Altgriechischen „Kunst des Denkens“ oder „Kunst des Argumentierens“.

Beispiel.

- Alle Vögel können fliegen.
- Tweety ist ein Vogel.
- Also kann Tweety fliegen.

Stimmt das Argument?

Wir können das Argument auf zwei Ebenen diskutieren: **Abstrakt**

In dem Beispiel wird eine Schlussfolgerung aus gemachten Annahmen gezogen.

Ist das überhaupt korrekt so?

Kann man aus den Annahmen die gemachte Behauptung wirklich folgern?

Logik?

Beispiel

Annahmen.

- Alle Vögel können fliegen.
- Tweety ist ein Vogel.

Schlussfolgerung.

- Also kann Tweety fliegen.

Beispiel

Annahmen.

- Alle Ersetzungschiffren sind anfällig für brute-force Angriffe.
- Der Caesar Chiffre ist ein Ersetzungschiffre.

Schlussfolgerung.

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

Stimmt das Argument?

Wir können das Argument auf zwei Ebenen diskutieren: **Abstrakt**

In dem Beispiel wird eine Schlussfolgerung aus gemachten Annahmen gezogen.

Ist das überhaupt korrekt so?

Kann man aus den Annahmen die gemachte Behauptung wirklich folgern?

Logik?

Beispiel

Annahmen.

- Alle Vögel können fliegen.
- Tweety ist ein Vogel.

Schlussfolgerung.

- Also kann Tweety fliegen.

Beispiel

Annahmen.

- Alle Ersetzungschiffren sind anfällig für brute-force Angriffe.
- Der Caesar Chiffre ist ein Ersetzungschiffre.

Schlussfolgerung.

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

Stimmt das Argument?

Wir können d

In dem Beisp

Ist das überhaupt

Kann man au

Abstrakte Argumentation

Annahmen.

- Wenn $x \vee y$ dann $x \wedge F$.
- $x \vee$

Schlussfolgerung.

- $x \wedge F$

gezogen.

gern?

Logik?

Beispiel

Annahmen.

- Alle Vögel können fliegen.
- Tweety ist ein Vogel.

Schlussfolgerung.

- Also kann Tweety fliegen.

Beispiel

Annahmen.

- Alle Ersetzungschiffren sind anfällig für brute-force Angriffe.
- Der Caesar Chiffre ist ein Ersetzungschiffre.

Schlussfolgerung.

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

Stimmt das Argument?

Wir können d

In dem Beisp

Ist das überhaupt

Kann man au

Abstrakte Argumentation

Annahmen.

- Wenn A dann B.

$$(A \rightarrow B)$$

$$A$$

Schlussfolgerung.

- B

$$B$$

gezogen.

gern?

Logik?

Logik. Die Logik stellt Methoden bereit, um solche Argumentationsketten untersuchen zu können.

Sprache. Es werden (formale) Sprachen bereitgestellt, mit denen solche Aussagen präzise formuliert werden können.

Beispiele. Aussagenlogik, Beschreibungslogiken, ...

Methoden. Es werden Methoden entwickelt, um die Korrektheit der Schlussfolgerungen überprüfen zu können.

Abstrakte Argumentation.
Annahmen.

- Wenn A dann B.
- A
- B

Schlussfolgerung.

Anwendung der Logik: Wissensrepräsentation

Wissensrepräsentation. Eine praktische Anwendung der Logik sind **Wissensrepräsentationssysteme**.

Solchen Systemen enthalten fachspezifisches Wissen aus einem Anwendungsbereich und erlauben es, daraus neue Folgerungen abzuleiten, Hypothesen zu testen usw.

Einerseits können solche Systeme Fachpersonen unterstützen indem sie gewisse Aufgaben automatisieren. Andererseits machen sie Fachwissen auch außerhalb der jeweiligen Disziplin zugänglich und nutzbar.

Systeme gibt es z.B. zur Diagnoseunterstützung in der Medizin, in der Biologie und vielen anderen Bereichen.

Beispiel.

- Alle Vögel können fliegen.
- Tweety ist ein Vogel.
- Also kann Tweety fliegen.

Wissensrepräsentationssysteme

Wissensrepräsentationssysteme bestehen aus zwei Teilen.

T-Box (terminology box). Hier werden Konzepte und deren Zusammenhänge beschrieben.

Beispiel. Wenn x ein Vogel ist, dann kann x fliegen.

A-Box (assertion box). Hier werden grundlegende Fakten (Axiome) festgelegt.

Beispiel. „Tweety ist ein Vogel“ „Tweety ist gelb.“

Beispiel.

- Alle Vögel können fliegen.
- Tweety ist ein Vogel.
- Also kann Tweety fliegen.

Beschreibungslogiken. Zur Formalisierung der Konzepte und deren Zusammenhänge werden sogenannte **Beschreibungslogiken** verwendet.

Aus der Wissensbasis können automatisch neue Fakten abgeleitet werden, Hypothesen getestet oder Inkonsistenzen gefunden werden.

Wie das Beispiel zeigt, steht und fällt der Ansatz damit, dass das in der Wissensbasis gespeicherte Wissen korrekt ist.

1.2 Weitere Beispiele für Logiken

Bereits bekannte Logiken: SQL

Datenbankanfragesprachen. Aus *Informationssysteme und Datenanalyse* kennen Sie die Sprache SQL, mit der Datenbanken abgefragt werden können.

$$T(x) = \text{SELECT } x \text{ WHERE TUTORIUM}(x, "Logik") \text{ AND} \\ (\text{TUTOR}(x, "Meike") \text{ OR TUTOR}(x, "Lydia"))$$

Eine SQL-Anweisung liefert die Menge aller Antworttupel zurück, also eine Relation.

Elemente von SQL.

Formel. $T(x) = \text{SELECT } x \text{ WHERE TUTOR}(x, "Logik") \dots$

Datenbank. legt die verwendeten Relationen Tutor, ... und damit die Werte fest, die für die Variablen x verwendet werden können.

Auswertung. liefert alle Belegungen der Variablen in Select zurück, die die Bedingungen in WHERE erfüllen.

Bereits bekannte Logiken: Reguläre Ausdrücke

Reguläre Ausdrücke. Sei $\Sigma := \{a, b\}$.

$$\sigma := (a + b)^* abab^* a^* a$$

Der Ausdruck definiert eine Sprache $L(\sigma) \subseteq \Sigma^*$.

Eine andere Sichtweise. Der Ausdruck σ „gilt für“ die Wörter $w \in \Sigma^*$, die in der Sprache $L(\sigma)$ liegen.

Elemente der Logik RE.

Formel. $\sigma := (a + b)^* abab^* a^* a$

Wertebereich. Das Alphabet Σ bestimmt die Menge Σ^* der möglichen Wörter und die Symbole, die in σ verwendet werden können.

Auswertung. Liegt ein Wort in der durch σ definierten Sprache?

Bereits bekannte Logiken: Aussagenlogik

Aus *Formale Sprachen und Automaten* kennen Sie bereits:

1.2.1 Definition (Syntax der Aussagenlogik)

Sei V eine Menge von aussagenlogischen Variablen.

Sei $V \cap \{\top, \perp, \neg, \wedge, \vee, \rightarrow, \leftrightarrow, (,)\} = \emptyset$.

Dann ist die Menge $A(V)$ der aussagenlogischen Formeln zu V definiert als kleinste Menge mit:

- $V \cup \{\top, \perp\} \subseteq A(V)$;
- wenn $\varphi \in A(V)$, dann auch $\neg\varphi \in A(V)$;
- wenn $\{\varphi_1, \varphi_2\} \subseteq A(V)$, dann auch $\{(\varphi_1 \wedge \varphi_2), (\varphi_1 \vee \varphi_2), (\varphi_1 \rightarrow \varphi_2), (\varphi_1 \leftrightarrow \varphi_2)\} \subseteq A(V)$.

1.2.3 Definition (Semantik der Aussagenlogik)

Sei V eine Menge von aussagenlogischen Variablen.

Eine Variablenbelegung β weist jedem Element $p \in V$ genau ein Element $\beta(p) \in B \triangleq \{W, F\}$ der Wahrheitswerte zu.

Die zu β gehörige Auswertung $[\cdot]^\beta$ weist jedem Element $\varphi \in A(V)$ genau ein Element $[\varphi]^\beta \in B$ zu. ($[\psi]^\beta$ darf kurz als $[\psi]$ geschrieben werden, wenn keine Verwechslungsgefahr besteht.)

Wir verwenden p, q, \dots als Metavariablen für aussagenlogische Variablen.

Wir verwenden φ, ψ, \dots als Metavariablen für aussagenlogische Formeln. Formeln in $A(V)$, die

Gemeinsamkeiten

	Formel	Umgebung in der ausgewertet wird	Bedeutung
AL	$\varphi := (X \vee Y)$	$\{0, 1\}$	$[\![\varphi]\!]^\beta$
SQL	SELECT x ...	Datenbank, Schema, Wertebereich	Semantik von SQL wann ist ein Tupel in der Antwort
RE	$a^* abab^*$	Σ^*	Wann liegt $w \in \Sigma^*$ in $L(\sigma)$

Die Umgebung (engl. domain)

Um die Bedeutung einer Formel bestimmen zu können, muss noch die „Umgebung“ festgelegt werden, in der die Formeln interpretiert werden sollen.

Beispiel: SQL Formeln über einem Datenbankschema sind für alle Datenbanken über diesem Schema definiert.

Elemente einer „Logik“

Die Semantikfunktion

Die Semantik der Logik legt fest, ob eine Formel φ für ein bestimmtes Element D der Umgebung gilt.

Formal: Relation \models , die genau die Paare (D, φ) enthält, für die φ in D gilt.

(D : Element der Umgebung, φ : Formel)

Logik

Informatik. Wissenschaft der Informationen, deren Verarbeitung, Analyse, Spezifikation, etc.

Daher sind Sprachen, mit denen Informationen präzise beschrieben werden können, ein zentrales Element in der Informatik.

Sprachen. In den meisten Modulen sind Sprachen „Mittel zum Zweck“.

Sie lernen zum Beispiel SQL Anfragen zu schreiben, behandeln Auswertungsalgorithmen, etc.

Thema dieser Vorlesung.

Im Modul „Logik“ geht es um die „Sprachen an sich“.

Die Sprachen, deren Aufbau und Eigenschaften stehen im Vordergrund, nicht ihre Anwendungen.

Thema der Vorlesung

Logik. In der Logik geht es um die „Sprachen an sich“.

Die Sprachen, deren Aufbau und Eigenschaften stehen im Vordergrund, nicht ihre Anwendungen.

Abstraktion. Die Logiken, die wir betrachten, sind sehr abstrakt und sehr reduziert.

Konkret beschäftigen wir uns mit zwei fundamentalen Logiken, der **Aussagenlogik** und der **Prädikatenlogik**. Die Elemente dieser Logiken finden sich in vielen praktischen Sprachen wieder.

Abstraktion hilft, Bezüge zwischen Anwendungsgebieten zu verstehen.

Ziele.

- Verständnis für den Umgang mit abstrakten Sprachen.
- Den Aufbau solcher Sprachen verstehen und ein Gefühl für die Möglichkeiten aber auch Grenzen dieser Sprachen entwickeln.
- Sie sollen verstehen, was eigentlich den Kern einer solchen Sprache ausmacht, und was eher syntaktischer Zucker ist.

Search Operators in GMail

Weitere Beispiele

Search Operators in GMail

The screenshot shows the Gmail inbox interface with a search bar at the top containing the query: "Logik" AND NOT ("Prüfung" OR "krank"). The search results are displayed below, showing 1-100 of about 12700 messages. The left sidebar includes links for Compose, Inbox (432), Starred, Snoozed, Important, Sent, and Drafts (19). The main area shows a list of messages with checkboxes, stars, and reply/forward icons.

Compose

Inbox 432

Starred

Snoozed

Important

Sent

Drafts 19

Logik AND NOT ("Prüfung" OR "krank")

1-100 of about 12700

Compose

Inbox 432

Starred

Snoozed

Important

Sent

Drafts 19

Logik AND NOT ("Prüfung" OR "krank")

1-100 of about 12700

Suchen in GMail

Suchanfrage in GMail. $Q = \text{"Logik"} \text{ AND NOT } (\text{"Prüfung"} \text{ OR } \text{"krank"})$

Wir suchen also alle EMails x in denen "Logik" vorkommt, aber nicht "Prüfung" oder "krank".

$$Q(x) = \text{,,Logik"}(x) \text{ AND NOT } (\text{,,Prüfung"}(x) \text{ OR } \text{,,krank"}(x))$$

Logik oder Nicht-Logik. Die Anfrage enthält zwei Arten von "Operatoren".

- *Datenabhängige Operatoren.* "Logik", "Prüfung", "krank" , ...

Der Term "Logik" bestimmt eine *Aussage*:

"in der betrachteten EMail kommt das Wort Logik vor".

Diese Aussage kann *wahr* oder *falsch* sein.

- *Logische Operatoren.* "AND", "NOT", "OR", ...

Logische Operatoren verknüpfen solche Aussagen zu komplexeren Aussagen.

Aussagenlogik

Programmiersprachen.

```
if f(i) >= n and g(i+1) < 2n:  
    ....
```

Aussagenlogik.

Die *Aussagenlogik* untersucht (*atomare*) *Aussagen*, die mit Hilfe von logischen Verknüpfungen zu komplexen Aussagen kombiniert werden können.

Wir werden Begriffe wie *logische Folgerung*, *Äquivalenz* ... einführen und Methoden kennen lernen, logische Folgerung oder Äquivalenz zu beweisen bzw. algorithmisch zu entscheiden.

Anfragesprachen.

$Q(x) = \text{„Logik"}(x) \text{ AND NOT}(\text{„Prüfung"}(x) \text{ OR } \text{„krank"}(x))$

Aussagenlogik

Programmiersprachen.

```
if f(i) >= n and g(i+1) < 2n:
```

....

Algorithmische Probleme.

Auswertungsproblem. Berechne $\{x : Q(x) \text{ ist wahr}\}$

Für eine EMail x , bestimme, ob $Q(x)$ wahr ist.

Äquivalenz. Sind zwei Anfragen Q und Q' äquivalent?

Das ist zum Beispiel bei der Anfrageoptimierung wichtig.

Anfragesprachen.

$Q(x) = \text{„Logik“}(x) \text{ AND NOT } (\text{„Prüfung“}(x) \text{ OR } \text{„krank“}(x))$

Anfrageoptimierung.

$Q(x) := R(x) \wedge (Q(x) \vee P(x))$

Besser auszuwerten.

$Q'(x) := ((R(x) \wedge Q(x)) \vee (R(x) \wedge P(x)))$

Frage.

Beschreiben Q und Q' die gleiche Datenbankanfrage? Sind sie also äquivalent?

Aussagenlogik

Programmiersprachen.

```
if f(i) >= n and g(i+1) < 2n:  
    ....
```

Aussagenlogische Prinzipien.

Wie die Beispiele zeigen, ist die Aussagenlogik fester Bestandteil vieler Abfrage-, Spezifikations- oder Programmiersprachen.

Die Methoden und Eigenschaften der Aussagenlogik, die wir hier abstrakt behandeln werden, lassen sich daher auf all diese Bereiche übertragen und dort einsetzen.

Anfragesprachen.

$$Q(x) = \text{„Logik“}(x) \text{ AND NOT } (\text{„Prüfung“}(x) \text{ OR } \text{„krank“}(x))$$

Suche in Webseiten

$$Q(x) = \text{„Logikaufgaben“}(x) \text{ AND } (\text{„Erfüllbarkeit“}(x) \text{ OR } \text{„Resolution“}(x))$$

Suche in Webseiten.

Nun suchen wir Webseiten zur Logik, auf die mindestens eine andere Logikseite verlinkt.

$$\begin{aligned} Q(x) &= (\text{„Logikaufgaben“}(x) \text{ AND } (\text{„Erfüllbarkeit“}(x) \text{ OR } \text{„Resolution“}(x))) \\ &\quad \text{AND } (\text{„Es gibt Webseite } y \text{“ } \text{„Logik“}(y) \text{ AND } \text{„Link“}(y, x)) \end{aligned}$$

Hier verlassen wir den Bereich der Aussagenlogik und kommen zur Prädikatenlogik bzw. zu SQL

Schaltkreisdesign \mathcal{T}

Spezifikation φ

Logik in der computer-unterstützten Verifikation

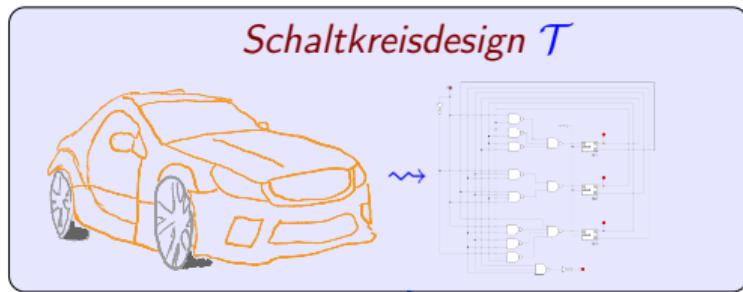


Aufgabe \rightarrow Aufgabe

Verifiziere $\mathcal{T} \models \varphi$.

Beispiel: Hard- und Software Verifikation

Beispiel. Model-Checking in der computerunterstützten Verifikation.



Verifiziere $\mathcal{T} \models \varphi$.

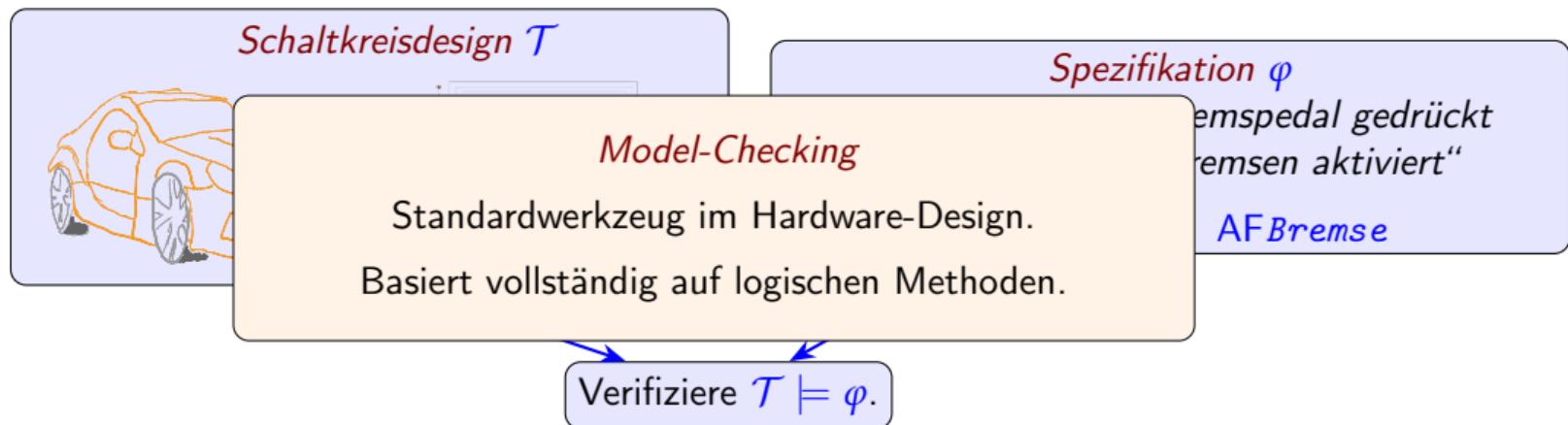
Ziel. Vollautomatische Überprüfung, ob der Schaltkreis die Spezifikation erfüllt. Testen reicht nicht, wir wollen einen Beweis, dass es keine Fehler gibt.

Dazu. Die Spezifikation muss absolut eindeutig und präzise formuliert werden.

Wiederum liefert die Logik das nötige Werkzeug.

Beispiel: Hard- und Software Verifikation

Beispiel. Model-Checking in der computerunterstützten Verifikation.



Ziel. Vollautomatische Überprüfung, ob der Schaltkreis die Spezifikation erfüllt. Testen reicht nicht, wir wollen einen Beweis, dass es keine Fehler gibt.

Dazu. Die Spezifikation muss absolut eindeutig und präzise formuliert werden.

Wiederum liefert die Logik das nötige Werkzeug.

Model-Checking

Model-Checking. Beim Model-Checking besteht die Eingabe aus

- einem Transitionssystem \mathcal{T} , das einen Schaltkreis, Controller, Kommunikationsprotokoll, etc. formalisiert und
- einer Spezifikation φ , meistens eine Formel einer *temporalen Logik*.

Die Aufgabe besteht darin,

1. möglichst vollautomatisch zu überprüfen, ob \mathcal{T} die Spezifikation φ erfüllt und
2. falls nicht, einen fehlerhaften Ablauf des Prozesses zu reproduzieren.

Um „ $\mathcal{T} \models \varphi$ “ zu überprüfen, wird das Problem oft auf das Erfüllbarkeitsproblem der Aussagenlogik (SAT) reduziert und durch SAT-Löser gelöst.

Model-Checking

Model-Checking. Beim Model-Checking besteht die Eingabe aus

- einem Transitionssystem \mathcal{T} , das einen Schaltkreis, Controller,
Kommunikationsprotokoll etc. formalisiert und

- einer Formel φ , die überprüft werden soll.

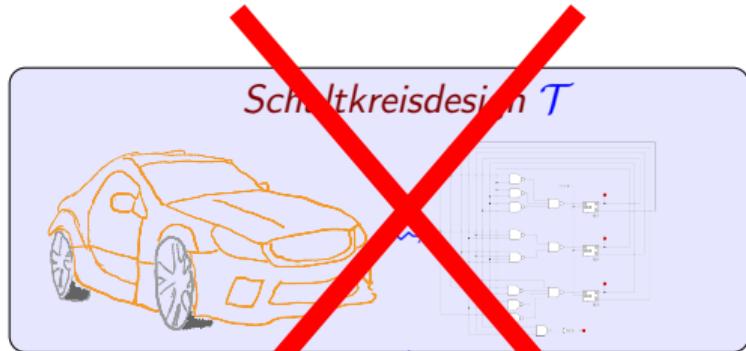
Die Ausgabe ist ein Ja/Nein-Antwort auf die Frage:

1. kann der Controller den Schaltkreis so steuern, dass *Kann man sich die Ingenieure nicht sparen?*

2. falls nicht, einen fehlerhaften Ablauf des Prozesses zu reproduzieren.

Um „ $\mathcal{T} \models \varphi$ “ zu überprüfen, wird das Problem oft auf das Erfüllbarkeitsproblem der Aussagenlogik (SAT) reduziert und durch SAT-Löser gelöst.

Synthese



Idee der Synthese.

Schreibe vollständige Spezifikation der *Anforderungen*, die der Controller erfüllen soll ↪ Formel φ .

Daraus soll dann (vollautomatisch) ein Controller \mathcal{T} erzeugt werden, der die Spezifikation φ erfüllt.

D.h. wird lösen das *Erfüllbarkeitsproblem* der Spezifikationslogik.

Zusammenfassung

Anwendungen der Logik finden sich in sehr vielen Bereichen der Informatik.

Neben den Datenbanken und der Verifikation besonders der Bereich der Programmiersprachen und deren Semantik.

Im Bereich der Algorithmen für NP-schwere Probleme hat sich in den letzten Jahren eine Methode als sehr erfolgreich erwiesen, die auf dem Erfüllbarkeitsproblem der Aussagenlogik basiert.

1.3 Organisatorisches

Wer wir sind

Fachgebietsleitung.

Stephan Kreutzer, stephan.kreutzer@tu-berlin.de

Wissenschaftliche Mitarbeiter.

Maximilian Gorsky, m.gorsky@tu-berlin.de

(Dario Cavallaro)

Tutor:innen.

Till,

Michelle,

Johannes,

Sebastian,

Elias

Vorlesung und Übungen

Termine.

Vorlesung	Do, 10-12	HE 101	Stephan Kreutzer
Großübung	Fr, 10-12	Zoom	Max Gorsky (ab 11.11.)

Tutorien

Wöchentliche Hausaufgaben.

Wir veröffentlichen jede Woche ein Hausaufgabenblatt.

Die Bearbeitung ist freiwillig.

Die Abgaben werden korrigiert und zurückgegeben.

Die Lösungen werden in der Großübung besprochen.

Portfolioprüfung

Portfolioprüfung. Anmeldung bis 23.11.2022

1. Hausarbeit	Abgabe: 7.12.2022	ISIS	10 PP
1. LK (MC-Test)	16.12.2022, 10:00 - 12:00	ISIS	30 PP
2. Hausarbeit	Abgabe: 9.02.2023	ISIS	20 PP
2. LK	04.03.2023, 15:30 - 18:30	Präsenz	40 PP

Es gilt der Notenschlüssel 1 der Fakultät IV.

Schriftliche Hausarbeiten.

Es gibt zwei schriftliche Hausarbeiten als Portfolioelemente.

Abgabe in Gruppen von 2 - 4 Personen.

Materialien

Folien.

Der Foliensatz für die gesamte Vorlesung wird auf ISIS veröffentlicht.

Nach jeder Vorlesung veröffentlichen wir die annotierten Folien auf der ISIS-Seite.

Skript.

Es ausführliches Skript finden Sie auf der ISIS-Seite.

Videos.

Die Videos der letzten Jahre finden Sie auf der ISIS-Seite.

Hinweis. Wir erwarten nicht, dass Sie die Videos anschauen.

Inhaltlich sind die Videos mit der Vorlesung weitestgehend gleich.

Sonstiges.

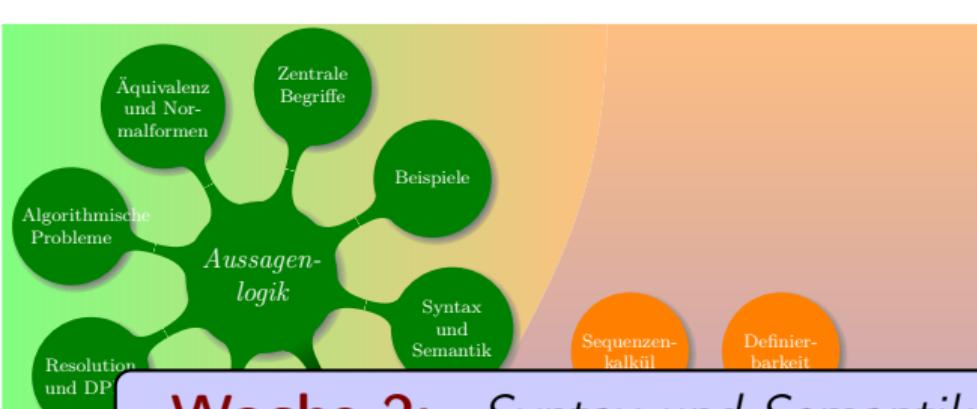
Wir veröffentlichen verschiedenes weiteres Übungsmaterial während des Semesters.

Sonstiges

Lassen Sie sich nicht durch Kommentare vergangener Jahrgänge
irritieren.

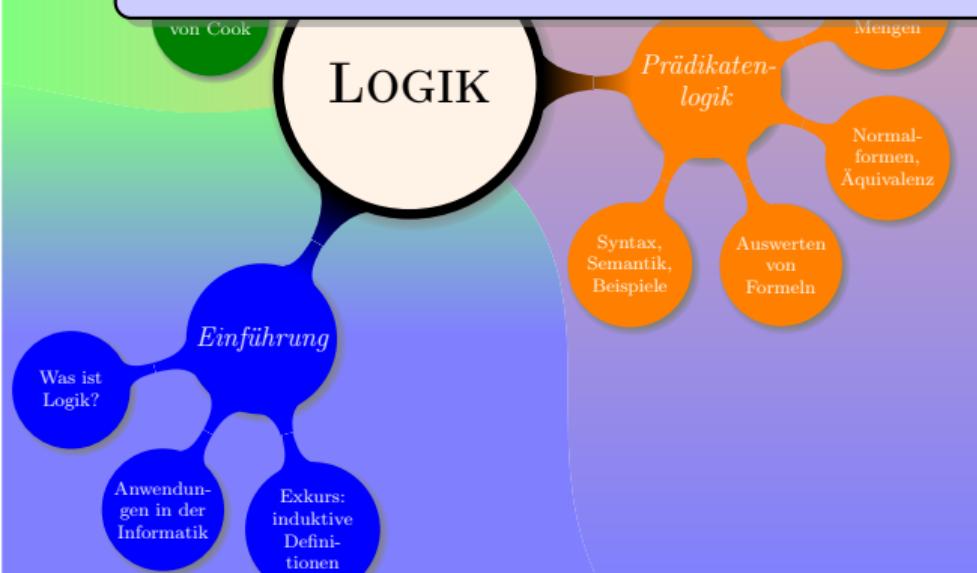
Bei Problemen, melden Sie sich bei uns.

Viel Erfolg



Nov.	17	18	19	20	21	22	23	<i>Einführung</i>
	24	25	26	27	28	29	30	<i>Aussagenlogik</i>
	31	1	2	3	4	5	6	<i>Normalformen</i>
	7	8	9	10	11	12	13	<i>Resolution</i>
Dez.	14	15	16	17	18	19	20	<i>Kompaktheit</i>
	21	22	23	24	25	26	27	<i>DPLL, Satz von Cook</i>

Woche 2: Syntax und Semantik der Aussagenlogik



5	6	7	8	9	10	11	<i>Prädikatenlogik</i>	
12	13	14	15	16	17	18	<i>Komplexität von FO</i>	
Jan.	19	20	21	22	23	24	25	<i>Weihnachten</i>
	26	27	28	29	30	31	1	<i>Neujahr</i>
	2	3	4	5	6	7	8	<i>Normalformen</i>
	9	10	11	12	13	14	15	<i>Definierbarkeit</i>
	16	17	18	19	20	21	22	<i>EF-Spiele</i>
Feb.	23	24	25	26	27	28	29	<i>EF-Spiele</i>
	30	31	1	2	3	4	5	<i>Sequenzenskalkül AL</i>
	6	7	8	9	10	11	12	<i>Sequenzenskalkül FO</i>
	13	14	15	16	17	18	19	<i>Ausblick</i>

2.1 Syntax der Aussagenlogik

Syntax der Aussagenlogik

Definition (Aussagenvariablen).

Wir fixieren eine abzählbar unendliche Menge **AVar** von Aussagenvariablen, die V_i für alle $i \geq 0$ enthält.

Definition. (Alphabet)

Das **Alphabet** der Aussagenlogik ist

$$\Sigma_{AL} := \text{AVar} \cup \{\top, \perp, \neg, \vee, \wedge, \rightarrow, \leftrightarrow, (,)\}$$

Definition.

Die Klasse **AL** der *aussagenlogischen Formeln* wird durch folgende Grammatik definiert:

$$\varphi ::= \top \mid \perp \mid X \in \text{AVar} \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid (\varphi \rightarrow \varphi) \mid (\varphi \leftrightarrow \varphi)$$

Syntax der Aussagenlogik

Die Aussagenlogik ist also induktiv über folgende Regeln definiert:

Basis (Grundmenge).

- \top, \perp sind aussagenlogische Formeln.
- Jede Variable $X \in \text{AVar}$ ist eine aussagenlogische Formel.

\top, \perp und die Variablen werden *atomare Formeln* oder *Atome* genannt.

Induktionsschritt (Regeln).

- 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
$$(\varphi \vee \psi), \quad (\varphi \wedge \psi), \quad (\varphi \rightarrow \psi), \quad (\varphi \leftrightarrow \psi)$$

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

Grammatik. $\varphi ::= \top \mid \perp \mid X \in \text{AVar} \mid \neg\varphi \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid (\varphi \rightarrow \varphi) \mid (\varphi \leftrightarrow \varphi)$

Beispiele

Syntaktisch korrekte Formeln.

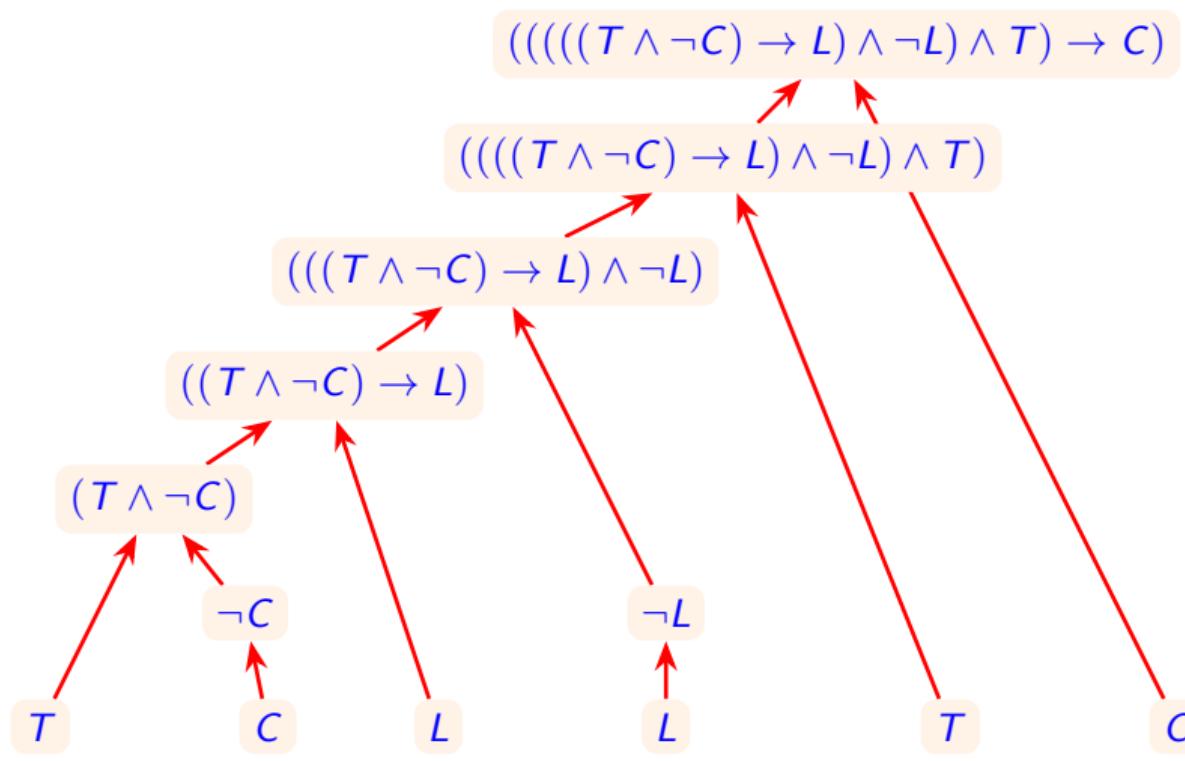
- V_0
- $(T \wedge \neg C)$
- $((A \wedge B) \wedge C) \vee D$

Syntaktisch inkorrekte Formeln.

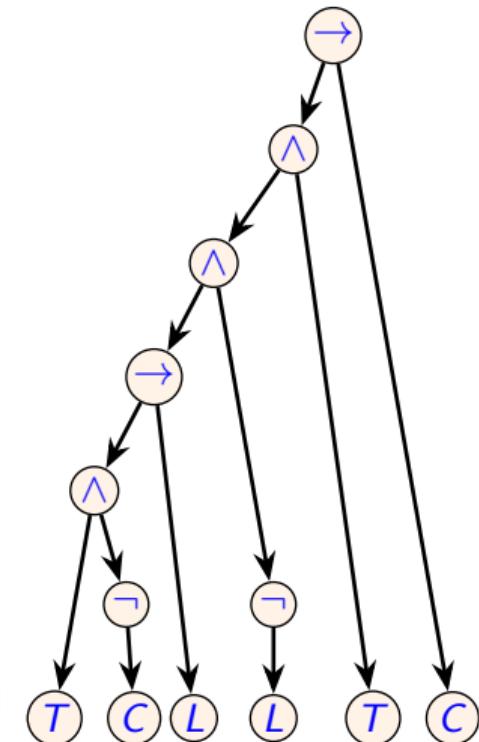
- $(T \& \neg C)$ (falsches Operatorsymbol)
- $A \wedge B$ (fehlende Klammern)
- $(A \wedge B \vee C)$ (fehlende Klammern)
- $\neg(A)$ (zuviele Klammern Klammern)

Grammatik. $\varphi ::= \top \mid \perp \mid X \in \text{AVar} \mid \neg \varphi \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid (\varphi \rightarrow \varphi) \mid (\varphi \leftrightarrow \varphi)$

Beispiel: Induktiver Aufbau einer Formel



vgl. Syntaxbaum



Grammatik. $\varphi ::= T \mid \perp \mid X \in \text{AVar} \mid \neg \varphi \mid (\varphi \wedge \varphi) \mid (\varphi \vee \varphi) \mid (\varphi \rightarrow \varphi) \mid (\varphi \leftrightarrow \varphi)$

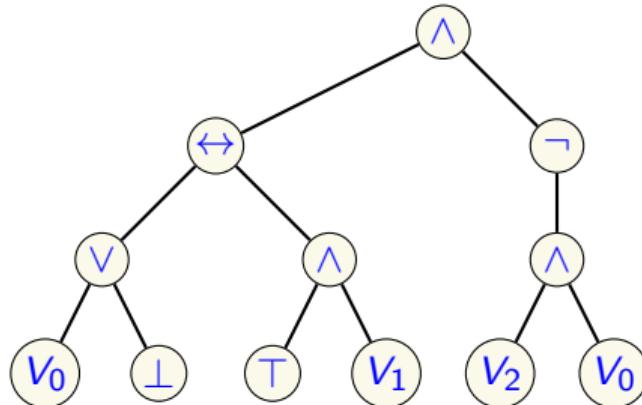
Syntax- oder Ableitungsbäume

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

Der Syntaxbaum der Formel

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

ist definiert wie folgt:



Präferenzregeln

Präferenzregeln. Um unnötige Klammern zu vermeiden,

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

Beispiel.

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

Oder $\neg(X \wedge Y \rightarrow T)$ für $\neg((X \wedge Y) \rightarrow T)$.

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

Präferenzregeln

Notation. Wenn $\Phi = \{\varphi_1, \dots, \varphi_n\} \subseteq \text{AL}$ eine *endliche* Menge von Formeln ist, schreiben wir

- $\vee \Phi$ als Abkürzung für $(\varphi_1 \vee \dots \vee \varphi_n)$ die Disjunktion über alle Formeln aus Φ und
- $\wedge \Phi$ als Abkürzung für $(\varphi_1 \wedge \dots \wedge \varphi_n)$ die Konjunktion über alle Formeln aus Φ .

Alternative Schreibweise. $\wedge_{i=1}^n \varphi_i$, $\vee_{1 \leq i \leq n} \varphi_i$, usw.

Beispiel.

Wir schreiben also $\wedge\{X_i : 1 \leq i \leq n\}$.

Aber wir können nicht $\wedge\{X_i : i \geq 1\}$ schreiben, da die Menge nicht endlich ist.

Präferenzregeln

Notation. Wenn $\Phi = \{\varphi_1, \dots, \varphi_n\} \subseteq \text{AL}$ eine *endliche* Menge von Formeln ist, schreiben wir

- $\vee \Phi$ als Abkürzung für $(\varphi_1 \vee \dots \vee \varphi_n)$ die Disjunktion über alle Formeln aus Φ und
- $\wedge \Phi$ als Abkürzung für $(\varphi_1 \wedge \dots \wedge \varphi_n)$ die Konjunktion über alle Formeln aus Φ .

Alternative Schreibweise. $\wedge_{i=1}^n \varphi_i, \quad \vee_{1 \leq i \leq n} \varphi_i,$ usw.

Beispiel.

$$\wedge \underbrace{\{\neg(X_{i,j} \vee Y_j^c) : 1 \leq i, j, c \leq 1909, i = 2^j, c = 3 \cdot i + 1\}}_{\begin{array}{l} \text{strikte} \\ \text{AL-Syntax} \end{array}}$$

$\underbrace{\quad}_{\text{beliebige mathematische Definition}}$
 $\underbrace{\quad}_{\text{keine syntaktische Einschränkung}}$

Unterformeln (intuitiv)

Definition. Die Menge $\text{sub}(\varphi)$ der *Unterformeln* einer Formel φ ist die Menge aller *Unterwörter* von φ , die selbst wieder korrekte Formeln sind.

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

Unterformeln

Definition.

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

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

Wir fassen sub als Funktion $\text{sub} : \text{AL} \rightarrow \mathcal{P}(\text{AL})$ auf, die jeder Formel φ die Menge $\text{sub}(\varphi)$ ihrer Unterformeln zuweist.

Beispiel

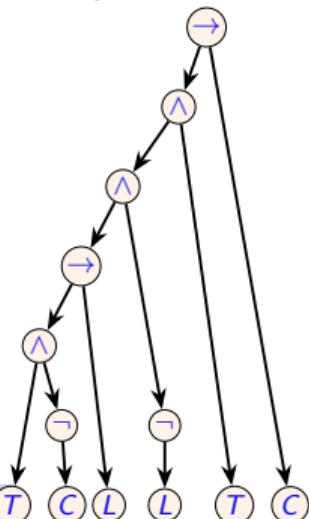
Beispiel. Sei $\varphi := (((((T \wedge \neg C) \rightarrow L) \wedge \neg L) \wedge T) \rightarrow C)$.

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

Definition. Menge $\text{sub}(\varphi)$:

- φ atomar $\rightsquigarrow \text{sub}(\varphi) := \{\varphi\}$.
- $\varphi := \neg \psi \rightsquigarrow \text{sub}(\varphi) := \{\varphi\} \cup \text{sub}(\psi)$.
- Für alle $* \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$:
 $\varphi := (\varphi_1 * \varphi_2) \rightsquigarrow \text{sub}(\varphi) := \{\varphi\} \cup \text{sub}(\varphi_1) \cup \text{sub}(\varphi_2)$.

Syntaxbaum



Zusammenfassung

Inhalt.

- Syntax der Aussagenlogik
- Präferenzregeln um die Notation zu vereinfachen
- Definition der Unterformeln einer Formel

2.2 Semantik der Aussagenlogik

Wahrheitsbelegungen

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

$$\text{var}(\varphi) := \text{AVar} \cap \text{sub}(\varphi).$$

Definition.

1. Eine *Wahrheitsbelegung*, oder kurz *Belegung*, ist eine partielle Funktion

$$\beta : \text{AVar} \rightarrow \{0, 1\}.$$

2. Eine Belegung β ist eine *Belegung für* eine Formel φ , oder ist *passend für* φ , wenn $\text{var}(\varphi) \subseteq \text{def}(\beta)$.

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

Semantik der Aussagenlogik

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

Induktionsbasis.

- $\llbracket \perp \rrbracket^\beta := 0 \quad \llbracket \top \rrbracket^\beta := 1$
- Für alle $X \in \text{AVar}$ gilt $\llbracket X \rrbracket^\beta := \beta(X)$

Belegung:
 $\beta : \text{AVar} \rightarrow \{0, 1\}$

passend für φ :
 $\text{var}(\varphi) \subseteq \text{def}(\beta)$.

Induktionsschritt. Für zusammengesetzte Formeln φ ist $\llbracket \varphi \rrbracket^\beta$ wie folgt definiert:

- $\llbracket \neg \varphi \rrbracket^\beta := 1 - \llbracket \varphi \rrbracket^\beta$
- $\llbracket (\varphi \wedge \psi) \rrbracket^\beta := \min\{\llbracket \varphi \rrbracket^\beta, \llbracket \psi \rrbracket^\beta\}$ $\llbracket (\varphi \vee \psi) \rrbracket^\beta := \max\{\llbracket \varphi \rrbracket^\beta, \llbracket \psi \rrbracket^\beta\}$
- $\llbracket (\varphi \rightarrow \psi) \rrbracket^\beta := \begin{cases} 1 & \text{wenn } \llbracket \varphi \rrbracket^\beta = 0 \text{ oder } \llbracket \psi \rrbracket^\beta = 1 \\ 0 & \text{sonst} \end{cases}$
- $\llbracket (\varphi \leftrightarrow \psi) \rrbracket^\beta = 1$ genau dann, wenn $\llbracket \varphi \rrbracket^\beta = \llbracket \psi \rrbracket^\beta$

Semantik der Aussagenlogik

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

Induktionsprinzip:

- $\llbracket \perp \rrbracket^\beta = 0$ „tertium non datur“
- Für alle Aussagen φ : Der Semantik liegt das Grundprinzip *tertium non datur* zugrunde:
Eine Aussage ist entweder wahr oder falsch.

Induktionsprinzip für zusammengesetzte Formeln: γ ist $\llbracket \gamma \rrbracket$ wie folgt definiert

Belegung:
 $\beta : \text{AVar} \rightarrow \{0, 1\}$

passend für φ :
 $\text{var}(\varphi) \subseteq \text{def}(\beta)$.

- $\llbracket \neg \varphi \rrbracket^\beta := 1 - \llbracket \varphi \rrbracket^\beta$
- $\llbracket (\varphi \wedge \psi) \rrbracket^\beta := \min\{\llbracket \varphi \rrbracket^\beta, \llbracket \psi \rrbracket^\beta\}$ $\llbracket (\varphi \vee \psi) \rrbracket^\beta := \max\{\llbracket \varphi \rrbracket^\beta, \llbracket \psi \rrbracket^\beta\}$
- $\llbracket (\varphi \rightarrow \psi) \rrbracket^\beta := \begin{cases} 1 & \text{wenn } \llbracket \varphi \rrbracket^\beta = 0 \text{ oder } \llbracket \psi \rrbracket^\beta = 1 \\ 0 & \text{sonst} \end{cases}$
- $\llbracket (\varphi \leftrightarrow \psi) \rrbracket^\beta = 1$ genau dann, wenn $\llbracket \varphi \rrbracket^\beta = \llbracket \psi \rrbracket^\beta$

Beispiel

Beispiel. Sei $\varphi := (((((T \wedge \neg C) \rightarrow L) \wedge \neg L) \wedge T)$.

- Für $\beta : T \mapsto 1 \quad C \mapsto 1 \quad L \mapsto 0$ gilt $\llbracket \varphi \rrbracket^\beta = 1$.

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

- Für $\beta : T \mapsto 1 \quad C \mapsto 0 \quad L \mapsto 0$ gilt $\llbracket \varphi \rrbracket^\beta = 0$.

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

Semantik.

- $\llbracket \perp \rrbracket^\beta := 0$
- $\llbracket \top \rrbracket^\beta := 1$
- Für alle $X \in \text{AVar}$ gilt
 $\llbracket X \rrbracket^\beta := \beta(X)$
- $\llbracket \neg \varphi \rrbracket^\beta := 1 - \llbracket \varphi \rrbracket^\beta$
- $\llbracket (\varphi \wedge \psi) \rrbracket^\beta := \min\{\llbracket \varphi \rrbracket^\beta, \llbracket \psi \rrbracket^\beta\}$
- $\llbracket (\varphi \vee \psi) \rrbracket^\beta := \max\{\llbracket \varphi \rrbracket^\beta, \llbracket \psi \rrbracket^\beta\}$
- $\llbracket (\varphi \rightarrow \psi) \rrbracket^\beta :=$
 1 wenn $\llbracket \varphi \rrbracket^\beta = 0$ oder $\llbracket \psi \rrbracket^\beta = 1$
 0 sonst
- $\llbracket (\varphi \leftrightarrow \psi) \rrbracket^\beta = 1$ gdw $\llbracket \varphi \rrbracket^\beta = \llbracket \psi \rrbracket^\beta$

Eine Bemerkung zur Implikation

Bemerkung. Logische Implikation → 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, auch ψ wahr sein muss.

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* $\varphi \oplus \psi$

		<i>Exklusives Oder</i>
$\llbracket \varphi \rrbracket^\beta$	$\llbracket \psi \rrbracket^\beta$	$\llbracket \varphi \oplus \psi \rrbracket^\beta$
0	0	0
0	1	1
1	0	1
1	1	0

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

Notation

Notation. Um die Lesbarkeit zu erhöhen, schreiben wir:

- Belegungen: β, γ, \dots
- Formeln: $\varphi, \psi, \varphi' \dots$
- Mengen von Formeln: Φ, Ψ, \dots
- Wir werden auch X, Y, \dots für Variablen verwenden

Das Koinzidenz Lemma

Lemma (Koinzidenzlemma).

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

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

Dann gilt $\llbracket \varphi \rrbracket^{\beta} = \llbracket \varphi \rrbracket^{\beta'}$.

Zusammenfassung

Die Semantik einer Logik bestimmt die Bedeutung der Formeln.

Variablen in der Aussagenlogik sind Platzhalter für Aussagen, die wahr oder falsch sein können.

Eine (passende) *Belegung* weist den Variablen einer Formel Wahrheitswerte zu und bestimmt damit die *Umgebung* in der die Formel ausgewertet wird.

Eine Formel φ hat unter jeder passenden Belegung β einen eindeutigen Wahrheitswert $\llbracket \varphi \rrbracket^\beta$.

2.3 Abstrakte Beispiele

Ein abstraktes Beispiel

Beispiel. Sei $\varphi := M \wedge (B \wedge (\neg F \vee G))$.

Es gilt $\text{var}(\varphi) := \{M, B, F, G\}$.

Eine Belegung β *passt* also zu φ , wenn $\{M, B, F, G\} \subseteq \text{def}(\beta)$.

Seien β_1, β_2 definiert durch

	M	B	F	G
β_1	1	1	1	0
β_2	1	1	0	0

Dann gilt:

$$[M \wedge (B \wedge (\neg F \vee G))]^{\beta_1} = 0 \quad \text{aber} \quad [M \wedge (B \wedge (\neg F \vee G))]^{\beta_2} = 1.$$

Semantik.

- $[\perp]^{\beta} := 0$
- $[\top]^{\beta} := 1$
- Für alle $X \in \text{AVar}$ gilt $[X]^{\beta} := \beta(X)$
- $[\neg \varphi]^{\beta} := 1 - [\varphi]^{\beta}$
- $[(\varphi \wedge \psi)]^{\beta} := \min\{[\varphi]^{\beta}, [\psi]^{\beta}\}$
- $[(\varphi \vee \psi)]^{\beta} := \max\{[\varphi]^{\beta}, [\psi]^{\beta}\}$
- $[(\varphi \rightarrow \psi)]^{\beta} :=$
 - 1 wenn $[\varphi]^{\beta} = 0$ oder $[\psi]^{\beta} = 1$
 - 0 sonst
- $[(\varphi \leftrightarrow \psi)]^{\beta} = 1$ gdw $[\varphi]^{\beta} = [\psi]^{\beta}$

Belegung:

$$\beta : \text{AVar} \rightarrow \{0, 1\}$$

passend für φ :

$$\text{var}(\varphi) \subseteq \text{def}(\beta).$$

Ein abstraktes Beispiel

Beispiel 2. Sei $\varphi_2 := (X \rightarrow Y) \leftrightarrow (\neg X \vee Y)$. Es gilt $\text{var}(\varphi_2) := \{X, Y\}$.

Um die Belegungen zu finden, die φ erfüllen, kann eine Wahrheitstafel benutzt werden.

X	Y	$(X \rightarrow Y)$	$(\neg X \vee Y)$	φ_2
0	0	1	1	1
0	1	1	1	1
1	0	0	0	1
1	1	1	1	1

Jede zu φ_2 passende Belegung erfüllt die Formel.

Ein abstraktes Beispiel

Beispiel 3. Sei

$$\varphi_3 := (X \leftrightarrow \neg Y) \wedge ((X \wedge P) \vee (\neg X \wedge W)) \wedge (P \rightarrow Y) \wedge (W \rightarrow \neg Y)$$

Es gilt $\text{var}(\varphi_3) := \{X, Y, P, W\}$.

Um die Belegungen zu finden, die φ_3 erfüllen, kann eine Wahrheitstafel benutzt werden.

X	Y	P	W	$X \leftrightarrow \neg Y$	$(X \wedge P) \vee (\neg X \wedge W)$	$P \rightarrow Y$	$W \rightarrow \neg Y$	φ_3
1	1	1	1	0	1	1	1	0
1	0	1	0	1	1	0	1	0
1	0	0	0	1	0	1	1	0
1	0	0	1	1	0	1	1	0

Die Formel φ_3 hat keine erfüllende Belegung.

Ein abstraktes Beispiel

Drei Beispiele.

1. $\varphi_1 := M \wedge (B \wedge (\neg F \vee G))$.

Die Formel wurde durch β_2 erfüllt, durch β_1 aber nicht.

2. $\varphi_2 := (X \rightarrow Y) \leftrightarrow (\neg X \vee Y)$.

Die Formel wurde durch jede passende Belegung erfüllt.

3. $\varphi_3 := (X \leftrightarrow \neg Y) \wedge ((X \wedge P) \vee (\neg X \wedge W)) \wedge (P \rightarrow Y) \wedge (W \rightarrow \neg Y)$.

Die Formel wurde durch keine Belegung erfüllt.

Erfüllbarkeit und Allgemeingültigkeit

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

1. Eine zu φ passende Belegung β erfüllt φ , oder ist ein *Modell* von φ , wenn $\llbracket \varphi \rrbracket^\beta = 1$.
Wir schreiben $\beta \models \varphi$.
2. φ ist *erfüllbar*, wenn es eine Belegung β gibt, die φ erfüllt.
Andernfalls ist φ *unerfüllbar*.
3. φ ist *allgemeingültig*, oder eine *Tautologie*, wenn jede zu φ passende Belegung φ erfüllt.

Erfüllbarkeit und Allgemeingültigkeit

Definition. Sei $\varphi \in AL$ eine Formel.

1. Eine zu φ passende Belegung β **erfüllt** φ , oder ist ein **Modell** von φ , wenn $\llbracket \varphi \rrbracket^\beta = 1$.

Wir schreiben $\beta \models \varphi$.

$$\varphi_1 := M \wedge (B \wedge (\neg F \vee G))$$

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

Andernfalls ist φ **unerfüllbar**.

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

$$\varphi_3 := (X \leftrightarrow \neg Y) \wedge ((X \wedge P) \vee (\neg X \wedge W)) \wedge (P \rightarrow Y) \wedge (W \rightarrow \neg Y)$$

$$\varphi_2 := (X \rightarrow Y) \leftrightarrow (\neg X \vee Y)$$

2.4 Beispiele: Grundlagen des formalen Beweisens

Beispiel

Beispiel. Betrachten wir folgende Argumentation.

Bekannt:

Wenn ein Graph zusammenhängend ist und keinen Kreis enthält, dann enthält er $n - 1$ Kanten.

Angenommen, ein gegebener Graph G

- enthält $> n - 1$ Kanten und
- ist zusammenhängend.

Dann enthält G einen Kreis.

Ist das Argument *gültig*?

Formalisierung des Beispiels

Beispiel. Ist das Argument *gültig*?

Bekannt:

Wenn ein Graph zusammenhängend ist und keinen
Kreis enthält, dann enthält er $n - 1$ Kanten.

Angenommen, ein gegebener Graph G

- enthält $> n - 1$ Kanten und
- ist zusammenhängend.

Dann enthält G einen Kreis.

Formalisierung.

1. $A \wedge \neg B \rightarrow C$
2. $\neg C$
3. A

Daraus soll B folgen.

Wir müssen also entscheiden, ob die Formel $((A \wedge \neg B \rightarrow C) \wedge \neg C \wedge A) \rightarrow B$ allgemeingültig ist.

Beispiel

Beispiel. Sie kann nicht zu hause sein, da sie an Bord oder zuhause 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.

2.5 Beispiele: Logik und Algorithmen I

Sudoku leicht gemacht

Aussagenlogik als algorithmisches Hilfsmittel

Sudoku

Sudoku. Betrachten wir 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					

Regeln.

Fülle Felder mit Zahlen aus $1, \dots, 9$, so dass jede Zahl genau einmal in

- jeder Zeile
- jeder Spalte
- jedem Block vorkommt.

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

Sudoku

Sudoku. Betrachten wir 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						

Regeln.

Fülle Felder mit Zahlen aus $1, \dots, 9$, so dass jede Zahl genau einmal in

- jeder Zeile
- jeder Spalte
- jedem Block vorkommt.

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

Sudoku

Ziel. Sudokus mit Hilfe der Aussagenlogik lösen.

Wir wollen also zu einem gegebenen Sudoku S eine Formel φ_S konstruieren, so dass die erfüllenden Belegungen von φ_S genau den Lösungen des Sudokus entsprechen.

Es muss also einen inhaltlichen Zusammenhang zwischen einer erfüllenden Belegung und einer Beschriftung des Sudokus geben.

Herangehensweise.

Wir müssen uns als erstes überlegen, welche Variablen wir verwenden wollen und was die Variablen bedeuten sollen.

Idee 1. Für jede Position (i, j) eine Variable $X_{i,j}$.

Die Variablen werden mit der Zahl belegt, die an der Position stehen soll.

Problem. Variablen können nur **wahr** oder **falsch** werden.

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

Regeln.

- Fülle Felder mit $1, \dots, 9$ s.d. jede Zahl genau einmal in
 - jeder Zeile
 - jeder Spalte
 - jedem Block vorkommt.

Sudoku

Ziel. Konstruiere zu gegebenen Sudoku S eine Formel φ_S , so dass:
erfüllenden Belegungen von φ_S entsprechen Lösungen des Sudokus

Herangehensweise.

Wir müssen uns als erstes überlegen, welche Variablen wir verwenden wollen und was die Variablen bedeuten sollen.

Variablen. Können nur **wahr** oder **falsch** werden.

Was soll es bedeuten, wenn eine Belegung β eine Variable X mit 1 belegt?

Idee 2. Für jede Position (i, j) und jede mögliche Beschriftung $c \in \{1, \dots, 9\}$ führen wir eine Variable $X_{i,j}^c$ ein.

$X_{i,j}^c$ wird **wahr**, wenn an der Stelle (i, j) die Zahl c steht.

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

Regeln.

- Fülle Felder mit $1, \dots, 9$ s.d. jede Zahl genau einmal in
 - jeder Zeile
 - jeder Spalte
 - jedem Block vorkommt.

Sudoku

Ziel. Konstruiere zu gegebenen Sudoku S eine Formel φ_S , so dass:
erfüllenden Belegungen von φ_S entsprechen Lösungen des Sudokus

Variablen. Für jede Position (i, j) und jede mögliche Beschriftung $c \in \{1, \dots, 9\}$ führen wir eine Variable $X_{i,j}^c$ ein.

$X_{i,j}^c$ wird **wahr**, wenn an der Stelle (i, j) die Zahl c steht.

Belegungen und Lösungen.

Aus Lösung L des Sudokus können wir Belegung β_L konstruieren:

$$\beta_L(X_{i,j}^c) = 1 \text{ gdw. } L \text{ das Feld } (i, j) \text{ mit } c \text{ beschriftet.}$$

Umgekehrt, wollen wir aus einer Belegung β eine Beschriftung L_β konstruieren:

$$L_\beta \text{ beschriftet Position } (i, j) \text{ mit } c \text{ gdw. } \beta(X_{i,j}^c) = 1.$$

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

Regeln.

- Fülle Felder mit $1, \dots, 9$ s.d.
jede Zahl genau einmal in
 - jeder Zeile
 - jeder Spalte
 - jedem Block
 vorkommt.

Sudoku

Variablen. Für jede Position (i, j) und jede mögliche Beschriftung $c \in \{1, \dots, 9\}$ führen wir eine Variable $X_{i,j}^c$ ein.

$X_{i,j}^c$ wird **wahr**, wenn an der Stelle (i, j) die Zahl c steht.

Von Belegungen β zu Lösungen L_β .

L_β beschrifft die Position (i, j) mit der Zahl c gdw. $\beta(X_{i,j}^c) = 1$.

Dazu muss φ_S sicherstellen, dass es für alle (i, j) genau ein $c \in \{1, \dots, 9\}$ gibt, so dass $\beta_S(X_{i,j}^c) = 1$.

Sei

$$\varphi_{\text{beschr}} := \bigwedge_{1 \leq i, j \leq 9} \bigvee_{c=1}^9 \left(X_{i,j}^c \wedge \bigwedge_{\substack{d \neq c, \\ d \in \{1, \dots, 9\}}} \neg X_{i,j}^d \right)$$

Erfüllende Belegungen β von φ_{beschr} entsprechen genau Beschriftungen der Sudoku-Felder mit Zahlen aus $\{1, \dots, 9\}$.

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

Regeln.

- Fülle Felder mit $1, \dots, 9$ s.d. jede Zahl genau einmal in
 - jeder Zeile
 - jeder Spalte
 - jedem Block
 vorkommt.

Sudoku

Variablen. Für jede Position (i, j) und jede mögliche Beschriftung $c \in \{1, \dots, 9\}$ führen wir eine Variable $X_{i,j}^c$ ein.

$X_{i,j}^c$ wird **wahr**, wenn an der Stelle (i, j) die Zahl c steht.

Belegungen und Beschriftungen. Die Formel φ_{beschr} garantiert uns, dass erfüllende Belegungen β genau möglichen Beschriftungen L_β der Sudoku-Felder entsprechen.

Aber: L_β muss keine gültige Lösung des gegebenen Sudokus sein!

Wir müssen noch folgendes sicherstellen:

- Die Beschriftung L_β entspricht den vorgegebenen Feldern.
- Die Beschriftung L_β erfüllt die Sudoku-Regeln.

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

Regeln.

Fülle Felder mit $1, \dots, 9$ s.d.
jede Zahl genau einmal in
 - jeder Zeile
 - jeder Spalte
 - jedem Block
vorkommt.

Formalisieren der vorausgefüllten Felder

Variablen. Für jede Position (i, j) und jede mögliche Beschriftung $c \in \{1, \dots, 9\}$ führen wir eine Variable $X_{i,j}^c$ ein.

$\beta(X_{i,j}^c) = 1$ gdw. L_β an die Stelle (i, j) die Zahl c schreibt.

Vorgegebene Anfangsbeschriftung.

“Die Beschriftung L_β entspricht den vorgegebenen Feldern.”

$$\varphi_{\text{anfang}}^S := \bigwedge \{X_{i,j}^c : \text{die Pos. } (i, j) \text{ ist mit } c \text{ vorausgefüllt}\}$$

Im Beispiel oben: $\varphi_{\text{anfang}}^S := X_{1,5}^6 \wedge X_{1,6}^8 \wedge X_{1,7}^1 \wedge \dots$

Eine Belegung β die $\varphi_{\text{beschr}} \wedge \varphi_{\text{anfang}}^S$ erfüllt entspricht also einer Beschriftung L_β , die dem gegebenen Sudoku entspricht.

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

Regeln.

- Fülle Felder mit $1, \dots, 9$ s.d. jede Zahl genau einmal in
 - jeder Zeile
 - jeder Spalte
 - jedem Block vorkommt.

Formalisieren der Sudoku-Regeln

Variablen. Für jede Position (i, j) und jede mögliche Beschriftung $c \in \{1, \dots, 9\}$ führen wir eine Variable $X_{i,j}^c$ ein.

$\beta(X_{i,j}^c) = 1$ gdw. L_β an die Stelle (i, j) die Zahl c schreibt.

Die Sudoku-Regeln. “Die Beschriftung L_β erfüllt die Sudoku-Regeln”

- In jeder Zeile kommt jede Zahl aus $\{1, \dots, 9\}$ genau einmal vor.

Für alle Zeilen i :

für verschiedene Spalten $j \neq j'$ und Zahlen c gilt nicht:

Position (i, j) und Position (i, j') ist mit c beschriftet.

$$\varphi_{\text{zeile}} := \bigwedge_{i=1}^9 \bigwedge_{1 \leq j < j' \leq 9} \bigwedge_{c=1}^9 \neg(X_{i,j}^c \wedge X_{i,j'}^c).$$

- In jeder Spalte kommt jede Zahl aus $\{1, \dots, 9\}$ genau einmal vor.
- In jedem Block kommt jede Zahl aus $\{1, \dots, 9\}$ genau einmal vor.

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

Regeln.

Fülle Felder mit $1, \dots, 9$ s.d.
jede Zahl genau einmal in

- jeder Zeile
- jeder Spalte
- jedem Block

 vorkommt.

Formalisieren der Sudoku-Regeln

Regel für Zeilen. “In jeder Zeile kommt jedes $c \in \{1, \dots, 9\}$ genau einmal vor.”

Für alle Zeilen i und Spalten $j \neq j'$ und Zahlen c gilt **nicht**:
Position (i, j) und Position (i, j') ist mit c beschriftet.

$$\varphi_{\text{zeile}} := \bigwedge_{i=1}^9 \bigwedge_{1 \leq j < j' \leq 9} \bigwedge_{c=1}^9 \neg(X_{i,j}^c \wedge X_{i,j'}^c).$$

Regel für Spalten. “In jeder Spalte kommt jede Zahl aus $\{1, \dots, 9\}$ genau einmal vor.”

$$\varphi_{\text{spalte}} := \bigwedge_{j=1}^9 \bigwedge_{1 \leq i < i' \leq 9} \bigwedge_{c=1}^9 \neg(X_{i,j}^c \wedge X_{i',j}^c).$$

Regel für Blöcke. “In jedem Block kommt jede Zahl aus $\{1, \dots, 9\}$ genau einmal vor.”

$$\varphi_{\text{block}} :=$$

$$\Lambda \{ \neg(X_{i,j}^c \wedge X_{i',j'}^c) : (i,j) \neq (i',j'), \lfloor \frac{i-1}{3} \rfloor = \lfloor \frac{i'-1}{3} \rfloor \text{ und } \lfloor \frac{j-1}{3} \rfloor = \lfloor \frac{j'-1}{3} \rfloor \}$$

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

Regeln.

- Fülle Felder mit $1, \dots, 9$ s.d.
- jede Zahl genau einmal in
 - jeder Zeile
 - jeder Spalte
 - jedem Block
- vorkommt.

Formalisierung von Sudoku in der Aussagenlogik

Variablen. Für jede Position (i, j) und jede mögliche Beschriftung $c \in \{1, \dots, 9\}$ führen wir eine Variable $X_{i,j}^c$ ein.

Die Formel φ_S . Sei S ein gegebenes Sudoku.

$$\varphi_S := \varphi_{\text{anfang}}^S \wedge \varphi_{\text{beschr}} \wedge \varphi_{\text{zeile}} \wedge \varphi_{\text{spalte}} \wedge \varphi_{\text{block}}$$

Dann gilt: Jede Belegung β , die φ_S erfüllt, induziert eine gültige Lösung L_β des Sudokus S .

L_β beschriftet die Position (i, j) mit c gdw. $\beta(X_{i,j}^c) = 1$.

Jede gültige Lösung L des Sudokus S induziert eine Belegung β_L , die φ_S erfüllt.

$\beta(X_{i,j}^c) = 1$ gdw. L die Position (i, j) mit c beschriftet.

Lemma. φ_S ist genau dann erfüllbar, wenn das Sudoku S lösbar ist.

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

Regeln.

- Fülle Felder mit $1, \dots, 9$ s.d.
- jede Zahl genau einmal in
 - jeder Zeile
 - jeder Spalte
 - jedem Block
- vorkommt.

Zusammenfassung

Wie wir am Beispiel des Sudokus gesehen haben, können Probleme dieser Art, sogenannte *constraint satisfaction problems*, recht leicht in der Aussagenlogik formalisiert werden.

Die Belegungen der Aussagenvariablen entsprechen dabei potentiellen Lösungen des Sudokus.

Erfüllende Belegungen entsprechen dann genau den korrekten Lösungen.

Wir brauchen also nur noch schnelle Verfahren, um Formeln der Aussagenlogik auf Erfüllbarkeit testen zu können.

↗ SAT-Löser und der DPLL Algorithmus

2.6 Beispiele: Effizientes Lösen NP-vollständiger Probleme

3-Färbbarkeit

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

Wir nennen $c : V(G) \rightarrow \{1, 2, 3\}$ eine *3-Färbung* von G .

Das Problem 3-COL. Das zugehörige Entscheidungsproblem

3-COL

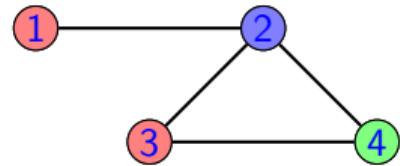
Eingabe. Graph G .

Problem. Entscheide, ob G 3-färbbar ist.

ist NP-vollständig.

Lösung durch die Aussagenlogik. Wir werden im Verlauf dieses Moduls sehen, wie das Problem mit Hilfe der Aussagenlogik gelöst werden kann.

Schritt 1: Formalisierung. Konstruiere aus einem Graph G eine Formel $\varphi_G \in \text{AL}$ die genau dann erfüllbar ist, wenn G 3-färbbar ist.



Eine Anwendung: 3-Färbbarkeit

Lemma.

Zu jedem endlichen Graph G können wir in polynomieller Zeit eine Formel φ_G konstruieren, so dass G genau dann 3-färbbar ist, wenn φ_G erfüllbar ist.

Grundidee.

3-Färbbarkeit. Wir suchen eine Funktion $c : V(G) \rightarrow \{1, 2, 3\}$, die die Knoten von G korrekt färbt.

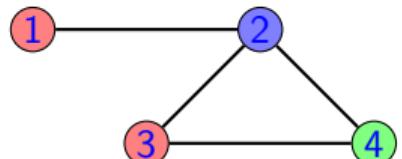
Erfüllbarkeit. Wir suchen eine Belegung $\beta : \text{var}(\varphi) \rightarrow \{0, 1\}$, die die Formel erfüllt.

Ziel. Konstruiere aus dem Graph G eine Formel φ_G , so dass

- **Belegungen** von $\text{var}(\varphi)$ den möglichen **Beschriftungen** der Knoten mit Farben aus $\{1, 2, 3\}$ entsprechen und
- **erfüllende Belegungen** genau den **korrekten 3-Färbungen** von G entsprechen.

Definition.

$G = (V, E)$ 3-färbbar, wenn
3-Färbung
 $c : V(G) \rightarrow \{C_1, C_2, C_3\}$
 existiert, so dass
 $c(u) \neq c(v)$
 für alle $\{u, v\} \in E(G)$.



Eine Anwendung: 3-Färbbarkeit

Lemma.

Zu jedem endlichen Graph G können wir in polynomieller Zeit eine Formel φ_G konstruieren, so dass G genau dann 3-färbbar ist, wenn φ_G erfüllbar ist.

Variablen. $X_{u,i}$ für alle $u \in V(G)$ und $1 \leq i \leq 3$.

Intention. Variable $X_{u,i}$ wird wahr, wenn Knoten u mit C_i gefärbt wird.

Färbungen vs. Belegungen.

Eine Funktion $c : V(G) \rightarrow \{C_1, C_2, C_3\}$ entspricht der Belegung β_c mit $\beta_c(X_{u,i}) = 1$ gdw. $c(u) = C_i$.

Plan. Belegung β entspricht Funktion $c : V(G) \rightarrow \{C_1, C_2, C_3\}$ mit $c(u) = C_i$ gdw. $\beta(X_{u,i}) = 1$. Ist aber falsch!

Definition.

$G = (V, E)$ 3-färbbar, wenn

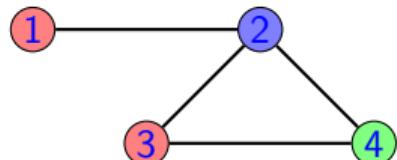
3-Färbung

$c : V(G) \rightarrow \{C_1, C_2, C_3\}$

existiert, so dass

$c(u) \neq c(v)$

für alle $\{u, v\} \in E(G)$.



Beweis des Lemmas

Färbungen vs. Belegungen.

$c : V(G) \rightarrow \{C_1, C_2, C_3\}$ entspricht β_c mit $\beta_c(X_{u,i}) = 1$ gdw. $c(u) = C_i$.

β entspricht $c : V(G) \rightarrow \{C_1, C_2, C_3\}$ mit $c(u) = C_i$ gdw. $\beta(X_{u,i}) = 1$.

Bedingungen an Färbungsfunktion.

$$\varphi_1 := \bigwedge \{ (X_{u,1} \vee X_{u,2} \vee X_{u,3}) : u \in V(G) \}$$

„Jeder Knoten erhält eine Farbe“

$$\varphi_2 := \bigwedge \{ \neg(X_{u,i} \wedge X_{u,j}) : u \in V(G), 1 \leq i \neq j \leq 3 \}$$

„Kein Knoten erhält zwei Farben“

$$\varphi_3 := \bigwedge \{ \bigwedge_{i=1}^3 \neg(X_{u,i} \wedge X_{v,i}) : \{u, v\} \in E(G) \}$$

„Keine monochromatische Kante“

Beobachtung. Belegungen β mit $\beta \models \varphi_1 \wedge \varphi_2 \wedge \varphi_3$ entsprechen genau den gültigen Färbungen.

Definition.

$G = (V, E)$ 3-färbbar, wenn

3-Färbung

$c : V(G) \rightarrow \{C_1, C_2, C_3\}$

existiert, so dass

$c(u) \neq c(v)$

für alle $\{u, v\} \in E(G)$.

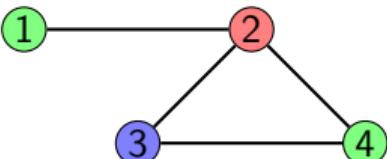
Variablen.

$X_{u,i}$ für $u \in V(G), 1 \leq i \leq 3$.

Bedeutung.

$\beta_c(X_{u,i}) = 1$ gdw $c(u) = C_i$.

Beispiel für 3-Färbbarkeit



Formel $\Phi_G := \varphi_1 \wedge \varphi_2 \wedge \varphi_3$

$$\varphi_1 := \wedge \{ (X_{1,1} \vee X_{1,2} \vee X_{1,3}), (X_{2,1} \vee X_{2,2} \vee X_{2,3}), (X_{2,1} \vee X_{2,2} \vee X_{2,3}), (X_{2,1} \vee X_{2,2} \vee X_{2,3}) \}$$

$$\varphi_2 := \wedge \{ (\neg(X_{1,1} \wedge X_{1,2}) \wedge \neg(X_{1,1} \wedge X_{1,3}) \wedge \neg(X_{1,2} \wedge X_{1,3})), \dots \}$$

$$\varphi_3 := \wedge \{ (\neg(X_{1,1} \wedge X_{2,1}) \wedge \neg(X_{1,2} \wedge X_{2,2}) \wedge \neg(X_{1,3} \wedge X_{2,3})), \dots \}$$

Formeln.

$$\varphi_1 := \wedge \{ (X_{u,1} \vee X_{u,2} \vee X_{u,3}) : u \in V(G) \}$$

„Jeder Knoten erhält eine Farbe“

$$\varphi_2 := \wedge \{ \neg(X_{u,i} \wedge X_{u,j}) : u \in V(G), 1 \leq i \neq j \leq 3 \}$$

„Kein Knoten erhält zwei Farben“

$$\varphi_3 := \wedge \{ \wedge_{i=1}^3 \neg(X_{u,i} \wedge X_{v,i}) : \{u, v\} \in E(G) \}$$

„Keine monochromatische Kante“

Erfüllende Belegungen vs. Färbungen.

	$X_{1,1}$	$X_{1,2}$	$X_{1,3}$	$X_{2,1}$	$X_{2,2}$	$X_{2,3}$	$X_{3,1}$	$X_{3,2}$	$X_{3,3}$	$X_{4,1}$	$X_{4,2}$	$X_{4,3}$
β_1	1	0	0	0	0	1	1	0	0	0	1	0
β_2	0	1	0	1	0	0	0	0	1	0	1	0

Beweis des Lemmas

Behauptung. φ_G ist genau dann erfüllbar, wenn G 3-färbbar ist.

Beweis der Rückrichtung. Sei $c : V(G) \rightarrow \{C_1, C_2, C_3\}$ eine 3-Färbung von G .

Definiere β_c durch: $\beta_c(X_{u,i}) = 1$ gdw. $c(u) = C_i$.

Dann gilt:

- $\beta_c \models \varphi_1$, denn für alle $u \in V(G)$:
wenn $c(u) = C_i$, dann $\beta_c(X_{u,i}) = 1$ und somit $\beta_c \models (X_{u,1} \vee X_{u,2} \vee X_{u,3})$.
- $\beta_c \models \varphi_2$, denn für alle $u \in V(G)$ gibt es nur ein i mit $c(u) = C_i$.
Daher gilt $\beta_c(X_{u,i}) = 1$ auch nur für einen Index i .
Für $i \neq j$ kann daher $(X_{u,i} \wedge X_{u,j})$ niemals in β_c wahr werden.
- $\beta_c \models \varphi_3$, folgt genauso.

Also gilt $\beta_c \models \varphi_G$.

Formel. $\varphi_G := \varphi_1 \wedge \varphi_2 \wedge \varphi_3$

$\varphi_1 := \wedge \{(X_{u,1} \vee X_{u,2} \vee X_{u,3}) : u \in V(G)\}$

„Jeder Knoten erhält eine Farbe“

$\varphi_2 := \wedge \{\neg(X_{u,i} \wedge X_{u,j}) : u \in V(G), 1 \leq i \neq j \leq 3\}$

„Kein Knoten erhält zwei Farben“

$\varphi_3 := \wedge \{ \wedge_{i=1}^3 \neg(X_{u,i} \wedge X_{v,i}) : \{u, v\} \in E(G)\}$

„Keine monochromatische Kante“

Beweis des Lemmas

Behauptung. φ_G ist genau dann erfüllbar, wenn G 3-färbbar ist.

Beweis der Rückrichtung. Sei $c : V(G) \rightarrow \{C_1, C_2, C_3\}$ eine 3-Färbung von G .

Definiere β_c durch: $\beta_c(X_{u,i}) = 1$ gdw. $c(u) = C_i$.

Also gilt $\beta_c \models \varphi_G$.

Hinrichtung. Umgekehrt, sei $\beta \models \varphi_G$ ein erfüllende Belegung.

Da β die Formeln φ_1 und φ_2 erfüllt, gibt es für jeden Knoten $u \in V(G)$ genau ein $i_u \in \{1, 2, 3\}$ mit $\beta(X_{u,i_u}) = 1$.

Definiere Färbung $c : V(G) \rightarrow \{C_1, C_2, C_3\}$ durch $c(u) := C_{i_u}$.

Da β die Formeln in 3) erfüllt, ist c eine gültige 3-Färbung. \dashv

Formel. $\varphi_G := \varphi_1 \wedge \varphi_2 \wedge \varphi_3$

$\varphi_1 := \wedge \{(X_{u,1} \vee X_{u,2} \vee X_{u,3}) : u \in V(G)\}$

„Jeder Knoten erhält eine Farbe“

$\varphi_2 := \wedge \{\neg(X_{u,i} \wedge X_{u,j}) : u \in V(G), 1 \leq i \neq j \leq 3\}$

„Kein Knoten erhält zwei Farben“

$\varphi_3 := \wedge \{ \wedge_{i=1}^3 \neg(X_{u,i} \wedge X_{v,i}) : \{u, v\} \in E(G)\}$

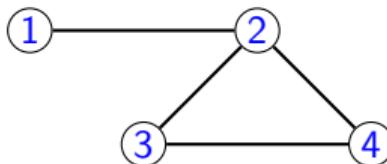
„Keine monochromatische Kante“

Anwendung: 3-Färbarkeit

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

Lemma.

Zu jedem endlichen Graph G können wir in polynomieller Zeit eine Formel φ_G konstruieren, so dass G genau dann 3-färbbar ist, wenn φ_G erfüllbar ist.



Anwendung: 3-Färbbarkeit

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

Lemma.

Zu jedem endlichen Graph G können wir in polynomieller Zeit eine Formel φ_G konstruieren, so dass G genau dann 3-färbbar ist, wenn φ_G erfüllbar ist.

Moderne SAT-Löser können Erfüllbarkeit sehr effizient entscheiden (meistens).

Somit erhalten wir einen einfachen und effizienten Algorithmus für 3-COL.

Wie das geht, werden wir in Woche 6 genauer besprechen.

Formalisieren algorithmischer Probleme in Aussagenlogik

Teilmengen einer Menge. Sei M eine Menge mit n Elementen.

Teilmengen $N \subseteq M$ können wir wie folgt durch Belegungen kodieren:

- Wir führen für jedes Element $m \in M$ eine Variable X_m ein.
- Eine Belegung $\beta : \{X_m : m \in M\} \rightarrow \{0, 1\}$ entspricht dann genau der Teilmenge $M_\beta = \{m \in M : \beta(X_m) = 1\}$.

Kodieren von Funktionen. Seien M und N endliche Mengen.

Funktionen $f : M \rightarrow N$ können wir wie folgt durch Belegungen β_f kodieren.

Variablen. $V := \{X_{m,n} : m \in M, n \in N\}$

Belegung. $\beta : V \rightarrow \{0, 1\}$ entspricht $f_\beta : M \rightarrow \mathcal{P}(N)$ mit $f_\beta(m) = \{n \in N : \beta(X_{m,n}) = 1\}$.

Bedingungen. Betrachte Formel $\varphi := \bigwedge \{\bigvee_{n \in N} (X_{m,n} \wedge \bigwedge_{n' \neq n} \neg X_{m,n'}) : m \in M\}$.

Wenn $\beta \models \varphi$, dann entspricht β einer Funktion $f_\beta : M \rightarrow N$.

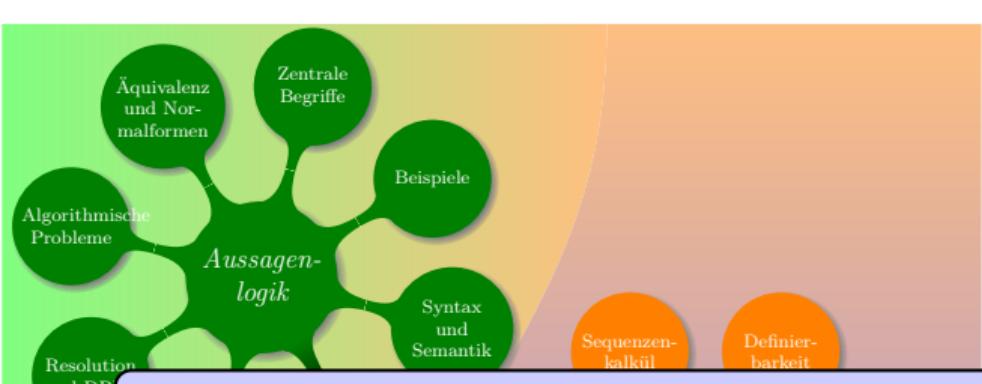
Zusammenfassung

Reduktion auf SAT.

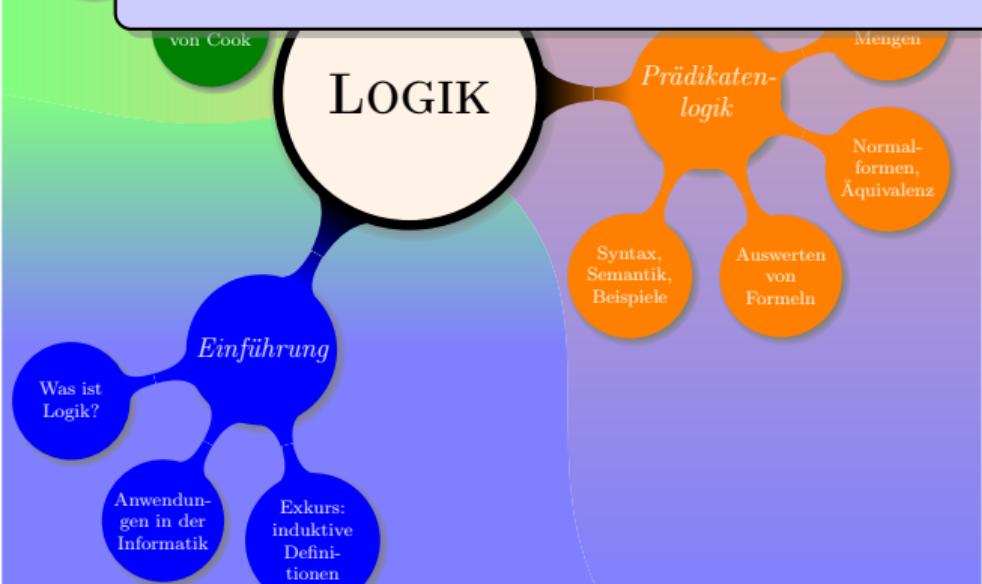
- Bestimmte algorithmische Probleme lassen sich recht einfach in der Aussagenlogik formalisieren.
- Dies gilt z.B. für viele *Constraint Satisfaction Probleme* oder NP-vollständige Probleme aus der Graphentheorie.
- Wir erhalten damit ein sehr allgemeines und mächtiges Werkzeug um schwere algorithmische Probleme in der Praxis effizient zu lösen.

Formalisierungstricks.

- Welcher Teil des Problems soll den erfüllenden Belegungen entsprechen?
- Kodierungstricks helfen bei der konkreten Formalisierung.



Woche 3: Normalformen und Substitution



Nov.	17	18	19	20	21	22	23	<i>Einführung</i>
	24	25	26	27	28	29	30	<i>Aussagenlogik</i>
	31	1	2	3	4	5	6	<i>Normalformen</i>
	7	8	9	10	11	12	13	<i>Resolution</i>
Dez.	14	15	16	17	18	19	20	<i>Kompaktheit</i>
	21	22	23	24	25	26	27	<i>DPLL, Satz von Cook</i>
					2	3	4	<i>Strukturen und FO</i>
					5	6	7	<i>Prädikatenlogik</i>
Jan.	9	10	11	12	13	14	15	<i>Komplexität von FO</i>
	16	17	18	19	20	21	22	<i>Weihnachten</i>
	23	24	25	26	27	28	29	<i>Neujahr</i>
	30	31	1	2	3	4	5	<i>Normalformen</i>
Feb.	9	10	11	12	13	14	15	<i>Definierbarkeit</i>
	16	17	18	19	20	21	22	<i>EF-Spiele</i>
	23	24	25	26	27	28	29	<i>EF-Spiele</i>
	30	31	1	2	3	4	5	<i>Sequenzenkalkül AL</i>
	6	7	8	9	10	11	12	<i>Sequenzenkalkül FO</i>
	13	14	15	16	17	18	19	<i>Ausblick</i>

3.1 Die zentralen Begriffe der Aussagenlogik

Definition der wichtigsten Begriffe der Logik

Seien $\varphi, \psi \in \text{AL}$ Formeln und $\Phi, \Psi \subseteq \text{AL}$ Formelmengen.

Folgerung. ψ folgt aus φ , geschrieben $\varphi \models \psi$, wenn für alle zu φ und ψ passenden Belegungen β gilt: Wenn $\beta \models \varphi$, dann auch $\beta \models \psi$.

Äquivalenz. φ und ψ sind äquivalent, geschrieben $\varphi \equiv \psi$, wenn für alle zu φ, ψ passenden Belegungen β gilt: $\beta \models \varphi$ genau dann, wenn $\beta \models \psi$.

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

Erfüllbarkeit. φ ist erfüllbar, wenn es eine Belegung β gibt, die φ erfüllt.
Andernfalls ist φ unerfüllbar.

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

Logische oder semantische Folgerung

Definition (semantische Folgerung).

1. Eine Formel ψ folgt aus einer Formel φ , geschrieben $\varphi \models \psi$, wenn für jede zu φ und ψ passende Belegung β gilt:

Wenn $\beta \models \varphi$, dann auch $\beta \models \psi$.

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

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

Beispiele.

1. $\{(X \wedge Y)\} \models (X \vee Y) ?$
2. $\{X, Y\} \models (X \vee Y) ?$
3. $\{X\} \models (X \wedge Y) ?$

Ein Beispiel

Beispiel. Sei G ein Graph. Betrachten wir folgende Annahmen.

Annahmen.

1. Wenn ein Graph zusammenhängend ist und keinen Kreis enthält, dann enthält er genau $n - 1$ Kanten.
2. G enthält $> n - 1$ Kanten.
3. G ist zusammenhängend.

Folgerung.

Daraus wollen wir folgern, dass G einen Kreis enthält.

Semantische Folgerung

Wie können wir solche logische Schlüsse formalisieren?

Gegeben eine Menge von Voraussetzungen:

- | | |
|---|---|
| <ol style="list-style-type: none"> 1. „Wenn G zusammenhängend und
G enthält keinen Kreis
dann hat G genau $n - 1$ Kanten“, 2. „G hat mehr als $n - 1$ Kanten“, 3. „G ist zusammenhängend“ | Wenn A und $\neg B$
dann C,
$\neg C,$
A |
|---|---|

können wir daraus „ G enthält einen Kreis“ schließen?

Es folgt B

Formalisierung.

$$\{(A \wedge \neg B \rightarrow C), \quad \neg C, \quad A\} \quad \models \quad B?$$

Anwendungen in der Wissensrepräsentation

Auf ähnliche Art kann man **Wissensrepräsentationssysteme** oder **Expertensysteme** aufbauen.

Wissenbasis. $\text{temp} > 39 \rightarrow \text{Fieber}$

$(\text{Fieber} \wedge (\text{Nacken steif} \vee \text{Kopfschmerz})) \rightarrow \text{Meng.-mögl.}$

...

Konkrete Situation.

Arzt fragt Symptome *Temperatur, Kopfschmerz ...* ab.

D.h. es wird eine partielle Variablenbelegung β erstellt.

Testen von Hypothesen.

Arzt kann Hypothesen testen.

Folgt $\neg \text{ „Meng.-mögl.“}$ aus der Wissenbasis und β ?

Hier werden sog. **Wissensrepräsentationslogiken** verwendet.

Definition der wichtigsten Begriffe der Logik

Seien $\varphi, \psi \in \text{AL}$ Formeln und $\Phi, \Psi \subseteq \text{AL}$ Formelmengen.

Folgerung. ψ folgt aus φ , geschrieben $\varphi \models \psi$, wenn für alle zu φ und ψ passenden Belegungen β gilt: Wenn $\beta \models \varphi$, dann auch $\beta \models \psi$.

Äquivalenz. φ und ψ sind äquivalent, geschrieben $\varphi \equiv \psi$, wenn für alle zu φ, ψ passenden Belegungen β gilt: $\beta \models \varphi$ genau dann, wenn $\beta \models \psi$.

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

Erfüllbarkeit. φ ist erfüllbar, wenn es eine Belegung β gibt, die φ erfüllt.
Andernfalls ist φ unerfüllbar.

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

Beziehungen zwischen den einzelnen Begriffen

Beziehungen zwischen den Begriffen

Lemma. Sei $\Phi \subseteq AL$ und $\psi, \psi' \in AL$.

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. Φ ist unerfüllbar genau dann, wenn $\Phi \models \varphi$ für alle $\varphi \in AL$ gilt.
4. Sei $\Phi_0 \subseteq \Phi$. Wenn $\Phi_0 \models \psi$, dann auch auch $\Phi \models \psi$.
5. $\varphi \equiv \psi$ genau dann, wenn $(\varphi \leftrightarrow \psi)$ allgemeingültig ist.
6. φ ist allgemeingültig $\iff \varphi \equiv \top$.

Zentrale Begriffe

Folgerung $\varphi \models \psi$: für alle β gilt:
Wenn $\beta \models \varphi$, dann $\beta \models \psi$.

Äquivalenz $\varphi \equiv \psi$: für alle β gilt:
 $\beta \models \varphi$ gdw. $\beta \models \psi$.

β erfüllt φ , ist **Modell** von φ ,
kurz $\beta \models \varphi$, wenn $[\![\varphi]\!]^\beta = 1$.

φ ist **erfüllbar**, wenn es Belegung β gibt, die φ erfüllt.

Andernfalls ist φ **unerfüllbar**.

φ **allgemeingültig** wenn jede passende Belegung φ erfüllt.

Definition der wichtigsten Begriffe der Logik

Seien $\varphi, \psi \in \text{AL}$ Formeln und $\Phi, \Psi \subseteq \text{AL}$ Formelmengen.

Folgerung. ψ folgt aus φ , geschrieben $\varphi \models \psi$, wenn für alle zu φ und ψ passenden Belegungen β gilt: Wenn $\beta \models \varphi$, dann auch $\beta \models \psi$.

Äquivalenz. φ und ψ sind äquivalent, geschrieben $\varphi \equiv \psi$, wenn für alle zu φ, ψ passenden Belegungen β gilt: $\beta \models \varphi$ genau dann, wenn $\beta \models \psi$.

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

Erfüllbarkeit. φ ist erfüllbar, wenn es eine Belegung β gibt, die φ erfüllt.
Andernfalls ist φ unerfüllbar.

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

3.2 Nützliche Äquivalenzen

Wiederholung: Äquivalenz

Definition. Zwei Formeln $\varphi, \psi \in \text{AL}$ der Aussagenlogik sind *äquivalent*, wenn für alle zu φ und ψ passenden Belegungen β gilt:

$$\beta \models \varphi \quad \Leftrightarrow \quad \beta \models \psi.$$

Nützliche Äquivalenzen

Theorem. Für alle $\psi, \varphi, \vartheta \in AL$:

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

Beweis der de Morganschen Regeln

Lemma (de Morgansche Regel). Für alle $\varphi, \psi \in \text{AL}$ gilt:

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

Beweis. Sei β eine passende Belegung.

Dann gilt

$$\llbracket \neg(\psi \wedge \varphi) \rrbracket^\beta = 1 \quad \text{gdw.} \quad \llbracket (\psi \wedge \varphi) \rrbracket^\beta = 0$$

gdw. mindestens eins von $\llbracket \psi \rrbracket^\beta, \llbracket \varphi \rrbracket^\beta$ gleich 0

gdw. mindestens eins von $\llbracket \neg\psi \rrbracket^\beta, \llbracket \neg\varphi \rrbracket^\beta$ gleich 1.

$$\text{gdw. } \llbracket (\neg\psi \vee \neg\varphi) \rrbracket^\beta = 1.$$

Beweis der Distributivität

Lemma (Distributivität). Für alle $\varphi, \psi, \vartheta \in \text{AL}$ gilt

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

Beweis. Sei β eine passende Belegung.

Fall 1 $\llbracket \psi \vee (\varphi \wedge \vartheta) \rrbracket^\beta = 0$.

Es gilt also $\llbracket \psi \rrbracket^\beta = 0$ und $\llbracket (\varphi \wedge \vartheta) \rrbracket^\beta = 0$.

Daraus folgt, dass $\llbracket \varphi \rrbracket^\beta = 0$ oder $\llbracket \vartheta \rrbracket^\beta = 0$.

O.B.d.A. sei $\llbracket \varphi \rrbracket^\beta = 0$.

Dann gilt aber $\llbracket \psi \vee \varphi \rrbracket^\beta = 0$ und somit $\llbracket (\psi \vee \varphi) \wedge (\psi \vee \vartheta) \rrbracket^\beta = 0$.

Beweis der Distributivität

Lemma (Distributivität). Für alle $\varphi, \psi, \vartheta \in \text{AL}$ gilt

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

Beweis. Sei β eine passende Belegung.

Fall 1 $\llbracket \psi \vee (\varphi \wedge \vartheta) \rrbracket^\beta = 0$.

Fall 2 $\llbracket \psi \vee (\varphi \wedge \vartheta) \rrbracket^\beta = 1$.

Es gilt also $\llbracket \psi \rrbracket^\beta = 1$ oder $\llbracket (\varphi \wedge \vartheta) \rrbracket^\beta = 1$.

Falls $\llbracket \psi \rrbracket^\beta = 1$ so folgt $\llbracket \psi \vee \varphi \rrbracket^\beta = 1$ und $\llbracket \psi \vee \vartheta \rrbracket^\beta = 1$

und somit $\llbracket (\psi \vee \varphi) \wedge (\psi \vee \vartheta) \rrbracket^\beta = 1$.

Andernfalls gilt $\llbracket \varphi \rrbracket^\beta = \llbracket \vartheta \rrbracket^\beta = 1$.

Dann aber gilt $\llbracket \psi \vee \varphi \rrbracket^\beta = \llbracket \psi \vee \vartheta \rrbracket^\beta = 1$ und somit

$\llbracket (\psi \vee \varphi) \wedge (\psi \vee \vartheta) \rrbracket^\beta = 1$. □

Große Disjunktionen und Konjunktionen

Erinnerung. Die folgende Notation ist oft nützlich:

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

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

Beispiel. $\bigwedge_{i=1}^{999} X_i \rightarrow Y$

„Wenn alle 999 X_i wahr sind, so muss auch Y wahr sein.“

Rechtfertigung. 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.

3.3 Substitution

Substitution

Erinnerung. Zwei Formeln $\varphi, \psi \in \text{AL}$ sind äquivalent, geschrieben $\varphi \equiv \psi$, wenn für alle Belegungen β gilt: $\beta \models \varphi \Leftrightarrow \beta \models \psi$.

Beispiel. Für alle $X, Y \in \text{AVar}$:

$$X \rightarrow Y \equiv \neg X \vee Y$$

$$X \leftrightarrow Y \equiv (X \rightarrow Y) \wedge (Y \rightarrow X)$$

Beweis mittels Wahrheitstafeln.

$\llbracket X \rrbracket^\beta$	$\llbracket Y \rrbracket^\beta$	$\llbracket X \rightarrow Y \rrbracket^\beta$	$\llbracket \neg X \vee Y \rrbracket^\beta$	$\llbracket X \leftrightarrow Y \rrbracket^\beta$	$\llbracket (X \rightarrow Y) \wedge (Y \rightarrow X) \rrbracket^\beta$
0	0	1	1	1	1
0	1	1	1	0	0
1	0	0	0	0	0
1	1	1	1	1	1

Substitution

Erinnerung. Zwei Formeln $\varphi, \psi \in \text{AL}$ sind äquivalent, geschrieben $\varphi \equiv \psi$, wenn für alle Belegungen β gilt: $\beta \models \varphi \Leftrightarrow \beta \models \psi$.

Beispiel. Für alle $X, Y \in \text{AVar}$:

$$X \rightarrow Y \equiv \neg X \vee Y$$

$$X \leftrightarrow Y \equiv (X \rightarrow Y) \wedge (Y \rightarrow X)$$

Lemma (Substitutionslemma, intuitiv).

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

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

$$\varphi \rightarrow \psi \equiv \neg \varphi \vee \psi$$

$$\varphi \leftrightarrow \psi \equiv (\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$$

Substitution

Definition. Eine **Substitution** ist eine partielle Abbildung

$$\mathcal{S} : \text{AVar} \rightarrow \text{AL}$$

von Aussagenvariablen auf aussagenlogische Formeln mit endlichem
Definitionsbereich,

d.h. \mathcal{S} ist nur für endlich viele Variablen definiert.

Beispiel. Sei

$$\mathcal{S} : \{V_1, V_2\} \rightarrow \text{AL}$$

definiert durch

$$\mathcal{S}(V_1) := V_2 \text{ und}$$

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

Substitution

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

Induktionsbasis.

- $\perp\mathcal{S} := \perp$
- $\top\mathcal{S} := \top$

- Wenn $X \in \text{AVar}$, dann $X\mathcal{S} := \begin{cases} \mathcal{S}(X) & \text{wenn } X \in \text{def}(\mathcal{S}) \\ X & \text{sonst} \end{cases}$

Induktionsschritt.

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

Informell. $\varphi\mathcal{S}$ entsteht aus φ indem alle Variablen $X \in \text{def}(\mathcal{S})$ durch $\mathcal{S}(X)$ ersetzt werden.

Beispiel.

$$\mathcal{S} : \{V_1, V_2\} \rightarrow \text{AL}$$

definiert durch

$$\mathcal{S}(V_1) := V_2 \text{ und}$$

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

Beispiel. Sei $\varphi := V_1 \wedge V_2 \rightarrow V_1 \vee V_3$.

Dann gilt

$$\varphi\mathcal{S} = (V_1 \wedge V_2)\mathcal{S} \rightarrow (V_1 \vee V_3)\mathcal{S}$$

$$= V_1\mathcal{S} \wedge V_2\mathcal{S} \rightarrow V_1\mathcal{S} \vee V_3\mathcal{S}$$

$$= V_2 \wedge (V_0 \vee V_1) \rightarrow V_2 \vee V_3.$$

Das Substitutionslemma (formal)

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

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

Beweis des Substitutionslemmas

Lemma. Sei \mathcal{S} eine Substitution und seien $\varphi, \varphi' \in \text{AL}$ Formeln.

Dann gilt $\varphi \equiv \varphi' \Rightarrow \varphi\mathcal{S} \equiv \varphi'\mathcal{S}$.

Beweisansatz.

Voraussetzungen. Wir wissen bereits, dass $\varphi \equiv \varphi'$.

D.h. für alle (passenden) Belegungen β gilt: $\beta \models \varphi$ gdw. $\beta \models \varphi'$

Dieses Wissen müssen wir im Beweis irgendwie benutzen.

Zu zeigen. Wir müssen zeigen, dass $\varphi\mathcal{S} \equiv \varphi'\mathcal{S}$. D.h.

für alle (passenden) Belegungen muss β gelten: $\beta \models \varphi\mathcal{S}$ gdw. $\beta \models \varphi'\mathcal{S}$

Da wir über φ, φ' nichts wissen, müssen wir also irgendwie Fragen wie

$\beta \models \varphi\mathcal{S}$ oder $\beta \models \varphi'\mathcal{S}$ auf $\beta' \models \varphi$ bzw. $\beta' \models \varphi'$ reduzieren,

möglicherweise für eine andere Belegung β' .

Beweis des Substitutionslemmas

Lemma. Sei \mathcal{S} eine Substitution und seien $\varphi, \varphi' \in \text{AL}$ Formeln.

Dann gilt $\varphi \equiv \varphi' \Rightarrow \varphi\mathcal{S} \equiv \varphi'\mathcal{S}$.

Notation. Sei \mathcal{S} eine Substitution.

Eine Belegung β ist passend für \mathcal{S} , wenn sie passend für alle Formeln $\mathcal{S}(X)$ mit $X \in \text{def}(\mathcal{S})$ ist.

Ist β eine zu \mathcal{S} passende Belegung, so definieren wir $\beta\mathcal{S}$ durch

$$\beta\mathcal{S}(X) := \begin{cases} \llbracket \mathcal{S}(X) \rrbracket^\beta & \text{wenn } X \in \text{def}(\mathcal{S}) \\ \beta(X) & \text{wenn } X \in \text{def}(\beta) \setminus \text{def}(\mathcal{S}) \end{cases}$$

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

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

Beispiel.

Sei $\mathcal{S} : X \mapsto (Y \vee \neg Z)$ und
 $\beta : Y \mapsto 1 \quad Z \mapsto 1$.

Dann ist

$$\beta\mathcal{S}(Y) := 1 \text{ und}$$

$$\beta\mathcal{S}(X) := \llbracket Y \vee \neg Z \rrbracket^\beta = 1.$$

Sei $\varphi := X \vee Y$.

Dann ist $\varphi\mathcal{S} := (Y \vee \neg Z) \vee Y$.

Beweis des Substitutionslemmas

Lemma A. Für alle Formeln φ und alle zu φ und \mathcal{S} passenden Belegungen β gilt: $\beta \models \varphi\mathcal{S} \iff \beta\mathcal{S} \models \varphi$.

Beweis. Strukturelle Induktion über den Formelaufbau.

Induktionsbasis.

- Für $\varphi \in \{\top, \perp\}$ gilt $\varphi\mathcal{S} = \varphi$
- Für $\varphi := X$, wobei $X \in \text{AVar} \setminus \text{def}(\mathcal{S})$, gilt:
 $\beta\mathcal{S}(X) = \beta(X)$ und $\varphi = \varphi\mathcal{S}$ und daher $\llbracket \varphi\mathcal{S} \rrbracket^\beta = \llbracket \varphi \rrbracket^{\beta\mathcal{S}}$.
- Für $\varphi := X \in \text{def}(\mathcal{S})$ gilt: $\varphi\mathcal{S} = \mathcal{S}(X)$ und $\beta\mathcal{S}(X) = \llbracket \mathcal{S}(X) \rrbracket^\beta$.
Somit, $\beta \models \varphi\mathcal{S} \iff \beta\mathcal{S} \models \varphi$.

Notation. Sei \mathcal{S} eine Substitution.

β passend für \mathcal{S} , wenn β zu allen $\mathcal{S}(X)$ mit $X \in \text{def}(\mathcal{S})$ passt.

Ist β eine zu \mathcal{S} passende Belegung, so definieren wir $\beta\mathcal{S}$ durch

$$\beta\mathcal{S}(X) := \begin{cases} \llbracket \mathcal{S}(X) \rrbracket^\beta & : X \in \text{def}(\mathcal{S}) \\ \beta(X) & : X \in \text{def}(\beta) \setminus \text{def}(\mathcal{S}) \end{cases}$$

Beispiel.

Sei $\mathcal{S} : X \mapsto (Y \vee \neg Z)$ und
 $\beta : Y \mapsto 1 \quad Z \mapsto 1$.

Dann ist

$$\beta\mathcal{S}(Y) := 1 \text{ und}$$

$$\beta\mathcal{S}(X) := \llbracket Y \vee \neg Z \rrbracket^\beta = 1.$$

Sei $\varphi := X \vee Y$.

Dann ist $\varphi\mathcal{S} := (Y \vee \neg Z) \vee Y$.

Beweis des Substitutionslemmas

Induktionsschritt.

- 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 $\ast \in \{\vee, \rightarrow, \leftrightarrow\}$ ist analog.

Notation. Sei \mathcal{S} eine Substitution.

Ist β eine zu \mathcal{S} passende Belegung, so definieren wir $\beta\mathcal{S}$ als

$$\beta\mathcal{S}(X) := \llbracket \mathcal{S}(X) \rrbracket^\beta \text{ wenn } X \in \text{def}(\mathcal{S})$$

$$\beta\mathcal{S}(X) := \beta(X) \text{ wenn } X \in \text{def}(\beta) \setminus \text{def}(\mathcal{S})$$

Beispiel.

Sei $\mathcal{S} : X \mapsto (Y \vee \neg Z)$ und
 $\beta : Y \mapsto 1 \quad Z \mapsto 1$.

Dann ist

$$\beta\mathcal{S}(Y) := 1 \text{ und}$$

$$\beta\mathcal{S}(X) := \llbracket Y \vee \neg Z \rrbracket^\beta = 1.$$

Sei $\varphi := X \vee Y$.

Dann ist $\varphi\mathcal{S} := (Y \vee \neg Z) \vee Y$.

Beweis des Substitutionslemmas

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

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

Beweis. Seien φ, φ' äquivalente Formeln.

Wir zeigen, dass $\varphi\mathcal{S} \equiv \varphi'\mathcal{S}$, d.h. dass 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 .

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

Das schließt den Beweis des Substitutionslemmas ab. □

Lemma A.

Für alle Formeln φ und alle zu φ und \mathcal{S} passenden Belegungen β gilt:
 $\beta \models \varphi\mathcal{S} \iff \beta\mathcal{S} \models \varphi$.

Notation

Notation.

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

Wenn X_1, \dots, X_n Variablen in φ sind und $\psi_1, \dots, \psi_n \in \text{AL}$, dann schreiben wir

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

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

$$\text{def}(\mathcal{S}) := \{X_1, \dots, X_n\} \text{ und } \mathcal{S}(X_i) := \psi_i.$$

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

Wir legen damit die Reihenfolge X_1, \dots, X_n fest und können dann $\varphi(\psi_1, \dots, \psi_n)$ für $\varphi[X_1/\psi_1, \dots, X_n/\psi_n]$ schreiben.

Beispiel

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

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

Äquivalenzen. Wir wissen bereits, dass

$$X \rightarrow Y \equiv \neg X \vee Y \quad \text{und} \quad X \leftrightarrow Y \equiv (X \rightarrow Y) \wedge (Y \rightarrow X)$$

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

$$\varphi \rightarrow \psi \equiv \neg \varphi \vee \psi$$

$$\varphi \leftrightarrow \psi \equiv (\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$$

Können wir daraus folgern, dass

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

Das Ersetzungslemma

Lemma. Sei $\varphi \in \text{AL}$ 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 $\varphi \equiv \varphi'$.

Korollar. Für alle Formeln φ, ψ :

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

3.4 Einschub: Boolesche Funktionen

Boolesche Funktionen

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

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

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

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

$$\begin{aligned} f_\varphi &: \{0, 1\}^n \rightarrow \{0, 1\} \\ f_\varphi(v_1, \dots, v_n) &:= \llbracket \varphi \rrbracket^\beta \end{aligned}$$

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

Beispiel.

$$\varphi := (X_1 \wedge X_2) \vee (X_1 \wedge X_3) \vee (X_2 \wedge X_3)$$

$$f_\varphi(1, 0, 1) = 1 \quad f_\varphi(0, 1, 1) = 1$$

$$f_\varphi(1, 1, 0) = 1 \quad f_\varphi(1, 1, 1) = 1$$

$$f_\varphi(v_1, v_2, v_3) = 0 \quad \text{sonst}$$

$f_\varphi(\bar{v})$: Mehrheitsfunktion

Boolesche Funktionen

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

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

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

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

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

Wir definieren nun die Funktion f durch die Formel

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

Beispiel. $\varphi := (X_1 \wedge X_2) \vee (X_1 \wedge X_3) \vee (X_2 \wedge X_3)$

$f_{\varphi}(1, 0, 1) = 1$	$f_{\varphi}(0, 1, 1) = 1$
$f_{\varphi}(1, 1, 0) = 1$	$f_{\varphi}(1, 1, 1) = 1$
$f_{\varphi}(v_1, v_2, v_3) = 0$	sonst

$f_{\varphi}(\bar{v})$: Mehrheitsfunktion

Für $(v_1, v_2, v_3) = (0, 1, 1)$ ist
 $\varphi_{0,1,1} = (\neg X_1 \wedge X_2 \wedge X_3)$.

$\varphi_f(X_1, X_2, X_3) :=$	
$(\neg X_1 \wedge X_2 \wedge X_3)$	$(=: \varphi_{0,1,1})$
\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	
$(X_1 \wedge X_2 \wedge X_3)$	$(=: \varphi_{1,1,1})$

Boolesche Funktionen

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

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

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

1. Wenn $f(\bar{v}) = 1$, dann $\beta \models \varphi_{\bar{v}}$ und $\varphi_{\bar{v}}$ ist Teil der Disjunktion in φ_f . Also $\beta \models \varphi_f$.
2. Umgekehrt, wenn $\beta \models \varphi_f$, dann muss es ein Disjunktionsglied $\varphi_{\bar{w}}$ geben, so dass $\beta \models \varphi_{\bar{w}}$.

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

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

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

Das schließt den Beweis ab.



Beispiel. $\varphi := (X_1 \wedge X_2) \vee (X_1 \wedge X_3) \vee (X_2 \wedge X_3)$

$$\begin{array}{ll} f_\varphi(1, 0, 1) = 1 & f_\varphi(0, 1, 1) = 1 \\ f_\varphi(1, 1, 0) = 1 & f_\varphi(1, 1, 1) = 1 \\ f_\varphi(v_1, v_2, v_3) = 0 & \text{sonst} \end{array}$$

$f_\varphi(\bar{v})$: Mehrheitsfunktion

Für $(v_1, v_2, v_3) = (0, 1, 1)$ ist $\varphi_{0,1,1} = (\neg X_1 \wedge X_2 \wedge X_3)$.

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

Formeln vs. Booleschen Funktionen

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

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

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

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

Beweis. Es gibt 2^n verschiedene Belegungen der Variablen X_1, \dots, X_n .

Also existieren 2^{2^n} verschiedene n -stellige Boolesche Funktionen und somit 2^{2^n} aussagenlogische Formeln in den Variablen X_1, \dots, X_n .

3.5 Normalformen

Normalformen

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

Reduzierte Formeln

Die schon bekannten Äquivalenzen liefern

$$\varphi \rightarrow \psi \equiv \neg\varphi \vee \psi$$

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

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

Korollar.

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

Wir nennen solche Formeln **reduziert**.

Äquivalenzen.

1. $\neg\neg\varphi \equiv \varphi$
2. $\varphi \rightarrow \psi \equiv \neg\varphi \vee \psi$
3. $\varphi \leftrightarrow \psi \equiv (\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$
4. $\neg(\varphi \wedge \psi) \equiv \neg\varphi \vee \neg\psi$
 $\neg(\varphi \vee \psi) \equiv \neg\varphi \wedge \neg\psi$
5. $\psi \wedge (\varphi \vee \theta) \equiv (\psi \wedge \varphi) \vee (\psi \wedge \theta)$
 $\psi \vee (\varphi \wedge \theta) \equiv (\psi \vee \varphi) \wedge (\psi \vee \theta)$
6. $\psi \wedge (\psi \vee \varphi) \equiv \psi \vee (\psi \wedge \varphi) \equiv \psi$
7. $\psi \wedge \varphi \equiv \varphi \wedge \psi$
 $\psi \vee \varphi \equiv \varphi \vee \psi$
8. $\psi \wedge (\varphi \wedge \theta) \equiv (\psi \wedge \varphi) \wedge \theta$
 $\psi \vee (\varphi \vee \theta) \equiv (\psi \vee \varphi) \vee \theta$

Negationsnormalform

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

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

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

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

Ein Algorithmus für die NNF

Eingabe. Eine Formel $\varphi \in \text{AL}$ ohne $\rightarrow, \leftrightarrow$.

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 $\varphi := \varphi'$.

Beispiel

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

Der Algorithmus erzeugt folgende Zwischenformeln:

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

NNF(φ). Eingabe. Eine Formel $\varphi \in \text{AL}$ ohne $\rightarrow, \leftrightarrow$.

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 $\varphi := \varphi'$.

Korrektheit und Vollständigkeit

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

Äquivalenz. Folgt aus dem Ersetzungslemma.

Terminierung. Wir definieren eine Funktion

$$h : \text{AL} \rightarrow \mathbb{N},$$

die die Höhe einer Formel angibt, wie folgt:

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

Die Höhe 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 Höhe 1 hat.

Beweis des Lemmas

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

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

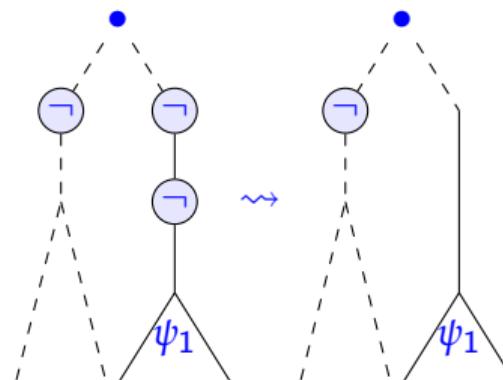
Beweis.

Angenommen, $\psi := \neg\neg\psi_1$.

Wir ersetzen also ψ durch ψ_1 .

Dadurch erhöht sich die Tiefe der restlichen Negationsformeln nicht.

Die Zahl dieser Formeln reduziert sich um 2 und daher $f(\varphi') < f(\varphi)$.



Beweis des Lemmas

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

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, $\psi := \neg(\psi_1 \vee \psi_2)$ oder $\psi := \neg(\psi_1 \wedge \psi_2)$.

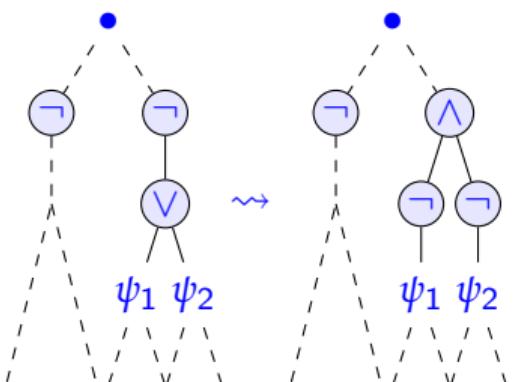
Dann bleibt die Tiefe aller Negationsformeln außer ψ gleich.

In der Summe wird $3^{h(\psi)}$ durch

$$3^{h(\neg\psi_1)} + 3^{h(\neg\psi_2)} \leq 2 \cdot 3^{h(\psi)-1} < 3^{h(\psi)}$$

ersetzt.

Also gilt $f(\varphi') < f(\varphi)$. □



Normalformen

Definition. Ein Literal L ist eine Aussagenvariable $X \in \text{AVar}$ oder deren Negation $\neg X$.

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

Normalformen

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

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

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

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

Theorem.

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.

Äquivalenzen.

1. $\neg\neg\varphi \equiv \varphi$
2. $\varphi \rightarrow \psi \equiv \neg\varphi \vee \psi$
3. $\varphi \leftrightarrow \psi \equiv (\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$
4. $\neg(\varphi \wedge \psi) \equiv \neg\varphi \vee \neg\psi$
 $\neg(\varphi \vee \psi) \equiv \neg\varphi \wedge \neg\psi$
5. $\psi \wedge (\varphi \vee \vartheta) \equiv (\psi \wedge \varphi) \vee (\psi \wedge \vartheta)$
 $\psi \vee (\varphi \wedge \vartheta) \equiv (\psi \vee \varphi) \wedge (\psi \vee \vartheta)$
6. $\psi \wedge (\psi \vee \varphi) \equiv \psi \vee (\psi \wedge \varphi) \equiv \psi$
7. $\psi \wedge \varphi \equiv \varphi \wedge \psi$
 $\psi \vee \varphi \equiv \varphi \vee \psi$
8. $\psi \wedge (\varphi \wedge \vartheta) \equiv (\psi \wedge \varphi) \wedge \vartheta$
 $\psi \vee (\varphi \vee \vartheta) \equiv (\psi \vee \varphi) \vee \vartheta$

Boolesche Funktionen

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

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

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

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

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

Wir definieren nun die Funktion f durch die Formel

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

Beispiel. $\varphi := (X_1 \wedge X_2) \vee (X_1 \wedge X_3) \vee (X_2 \wedge X_3)$

$f_{\varphi}(1, 0, 1) = 1$	$f_{\varphi}(0, 1, 1) = 1$
$f_{\varphi}(1, 1, 0) = 1$	$f_{\varphi}(1, 1, 1) = 1$
$f_{\varphi}(v_1, v_2, v_3) = 0$	sonst

$f_{\varphi}(\bar{v})$: Mehrheitsfunktion

Für $(v_1, v_2, v_3) = (0, 1, 1)$ ist
 $\varphi_{0,1,1} = (\neg X_1 \wedge X_2 \wedge X_3)$.

$\varphi_f(X_1, X_2, X_3) :=$	
$(\neg X_1 \wedge X_2 \wedge X_3)$	$(=: \varphi_{0,1,1})$
\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	
$(X_1 \wedge X_2 \wedge X_3)$	$(=: \varphi_{1,1,1})$

Normalformen

Theorem.

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

Beweis.

1. Teil 1 folgt aus dem Beweis der Äquivalenz zu Booleschen Funktionen, da die dort konstruierten Formeln in disjunktiver Normalform sind.
2. Sei $\varphi \in AL$.

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

Mit Hilfe der de Morganschen Gesetze erhält man

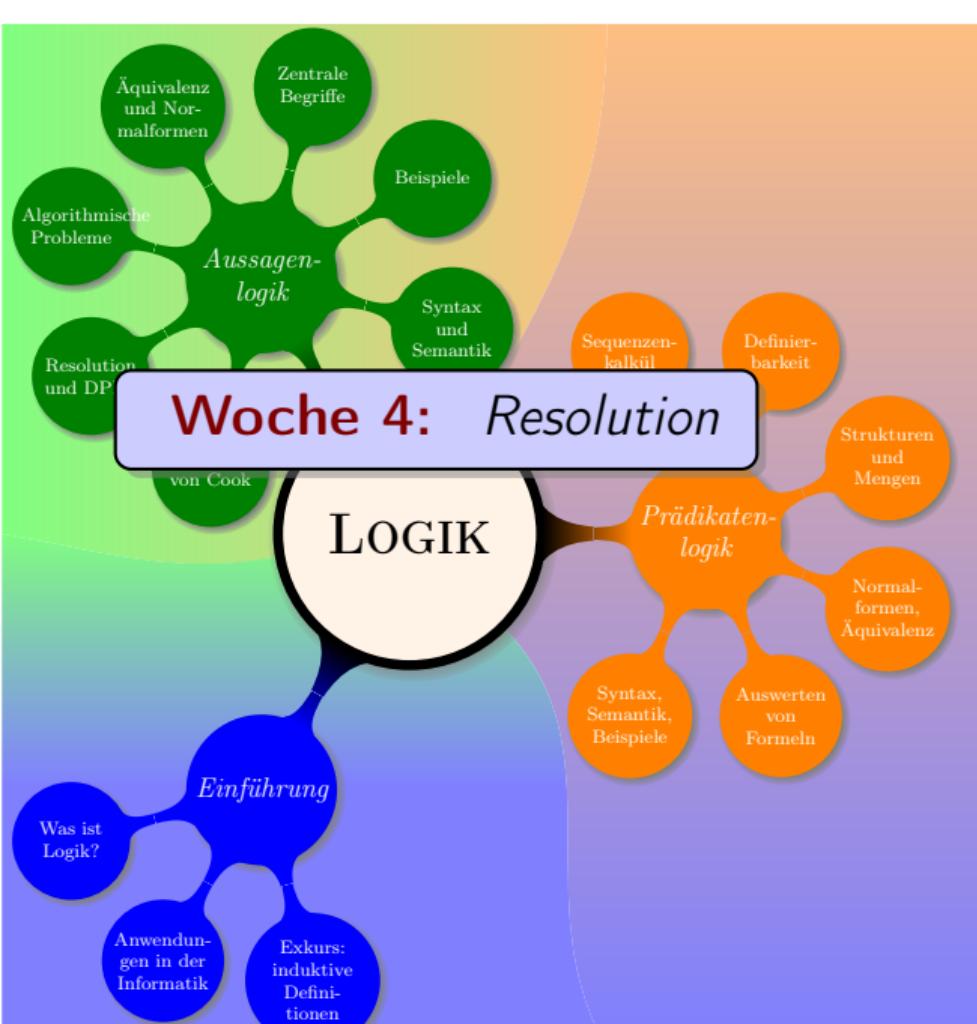
$$\varphi \equiv \neg \bigvee_{i=1}^n \bigwedge_{j=1}^{n_i} L_{i,j} \quad \equiv \quad \bigwedge_{i=1}^n \bigvee_{j=1}^{n_i} \overline{L_{i,j}}. \quad \square$$

Normalformen

Bemerkung.

- Formeln in **disjunktiver Normalform** können sehr effizient auf Erfüllbarkeit getestet werden.
- Allerdings gibt es Formeln $\varphi_n \in \text{AL}$, 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 $\varphi \in \text{AL}$ in Polynomialzeit eine Formel $\psi \in \text{AL}$ in **konjunktiver Normalform** konstruiert werden, so dass
$$\varphi \text{ genau dann erfüllbar ist, wenn } \psi \text{ erfüllbar ist.}$$

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



Nov.	17	18	19	20	21	22	23	<i>Einführung</i>
	24	25	26	27	28	29	30	<i>Aussagenlogik</i>
	31	1	2	3	4	5	6	<i>Normalformen</i>
	7	8	9	10	11	12	13	<i>Resolution</i>
Dez.	14	15	16	17	18	19	20	<i>Kompaktheit</i>
	21	22	23	24	25	26	27	<i>DPLL, Satz von Cook</i>
	28	29	30	1	2	3	4	<i>Strukturen und FO</i>
	5	6	7	8	9	10	11	<i>Prädikatenlogik</i>
	12	13	14	15	16	17	18	<i>Komplexität von FO</i>
Jan.	19	20	21	22	23	24	25	<i>Weihnachten</i>
	26	27	28	29	30	31	1	<i>Neujahr</i>
	2	3	4	5	6	7	8	<i>Normalformen</i>
	9	10	11	12	13	14	15	<i>Definierbarkeit</i>
	16	17	18	19	20	21	22	<i>EF-Spiele</i>
Feb.	23	24	25	26	27	28	29	<i>EF-Spiele</i>
	30	31	1	2	3	4	5	<i>Sequenzkalkül AL</i>
	6	7	8	9	10	11	12	<i>Sequenzkalkül FO</i>
	13	14	15	16	17	18	19	<i>Ausblick</i>

Wiederholung

Definition der wichtigsten Begriffe der Logik

Seien $\varphi, \psi \in \text{AL}$ Formeln und $\Phi, \Psi \subseteq \text{AL}$ Formelmengen.

Folgerung. ψ folgt aus φ , geschrieben $\varphi \models \psi$, wenn für alle zu φ und ψ passenden Belegungen β gilt: Wenn $\beta \models \varphi$, dann auch $\beta \models \psi$.

Äquivalenz. φ und ψ sind äquivalent, geschrieben $\varphi \equiv \psi$, wenn für alle zu φ, ψ passenden Belegungen β gilt: $\beta \models \varphi$ genau dann, wenn $\beta \models \psi$.

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

Erfüllbarkeit. φ ist erfüllbar, wenn es eine Belegung β gibt, die φ erfüllt.
Andernfalls ist φ unerfüllbar.

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

Nützliche Äquivalenzen

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

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

Substitution

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

Induktionsbasis.

- $\perp\mathcal{S} := \perp \quad \top\mathcal{S} := \top$
- Wenn $X \in \text{AVar}$, dann $X\mathcal{S} := \begin{cases} \mathcal{S}(X) & \text{wenn } X \in \text{def}(\mathcal{S}) \\ X & \text{sonst} \end{cases}$

Induktionsschritt.

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

Informell. $\varphi\mathcal{S}$ entsteht aus φ indem alle Variablen $X \in \text{def}(\mathcal{S})$ durch $\mathcal{S}(X)$ ersetzt werden.

Substitution. partielle Abbildung $\mathcal{S} : \text{AVar} \rightarrow \text{AL}$ mit endlichem Definitionsbereich.

Beispiel. Sei $\varphi := V_1 \wedge V_2 \rightarrow V_1 \vee V_3$.

Dann gilt

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

Das Substitutionslemma (formal)

Substitutionslemma.

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

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

Ersetzungslemma.

Sei $\varphi \in \text{AL}$ 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 $\varphi \equiv \varphi'$.

Boolesche Funktionen

Definition. Eine (n -stellige) Boolesche Funktion, ist eine Funktion

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

\mathbb{B}^n : Menge aller n -stelligen Booleschen Funktionen.

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

$$\begin{aligned} f_\varphi &: \{0, 1\}^n \rightarrow \{0, 1\} \\ f_\varphi(v_1, \dots, v_n) &:= \llbracket \varphi \rrbracket^\beta \end{aligned}$$

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

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

Beispiel.

$$\varphi := (X_1 \wedge X_2) \vee (X_1 \wedge X_3) \vee (X_2 \wedge X_3)$$

$$f_\varphi(1, 0, 1) = 1 \quad f_\varphi(0, 1, 1) = 1$$

$$f_\varphi(1, 1, 0) = 1 \quad f_\varphi(1, 1, 1) = 1$$

$$f_\varphi(v_1, v_2, v_3) = 0 \quad \text{sonst}$$

$f_\varphi(\bar{v})$: Mehrheitsfunktion

3.5 Normalformen

Normalformen

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

Reduzierte Formeln

Die schon bekannten Äquivalenzen liefern

$$\varphi \rightarrow \psi \equiv \neg\varphi \vee \psi$$

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

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

Korollar.

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

Wir nennen solche Formeln **reduziert**.

Äquivalenzen.

1. $\neg\neg\varphi \equiv \varphi$
2. $\varphi \rightarrow \psi \equiv \neg\varphi \vee \psi$
3. $\varphi \leftrightarrow \psi \equiv (\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$
4. $\neg(\varphi \wedge \psi) \equiv \neg\varphi \vee \neg\psi$
 $\neg(\varphi \vee \psi) \equiv \neg\varphi \wedge \neg\psi$
5. $\psi \wedge (\varphi \vee \theta) \equiv (\psi \wedge \varphi) \vee (\psi \wedge \theta)$
 $\psi \vee (\varphi \wedge \theta) \equiv (\psi \vee \varphi) \wedge (\psi \vee \theta)$
6. $\psi \wedge (\psi \vee \varphi) \equiv \psi \vee (\psi \wedge \varphi) \equiv \psi$
7. $\psi \wedge \varphi \equiv \varphi \wedge \psi$
 $\psi \vee \varphi \equiv \varphi \vee \psi$
8. $\psi \wedge (\varphi \wedge \theta) \equiv (\psi \wedge \varphi) \wedge \theta$
 $\psi \vee (\varphi \vee \theta) \equiv (\psi \vee \varphi) \vee \theta$

Negationsnormalform

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

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

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

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

Ein Algorithmus für die NNF

Eingabe. Eine Formel $\varphi \in \text{AL}$ ohne $\rightarrow, \leftrightarrow$.

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 $\varphi := \varphi'$.

Beispiel

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

Der Algorithmus erzeugt folgende Zwischenformeln:

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

NNF(φ). Eingabe. Eine Formel $\varphi \in \text{AL}$ ohne $\rightarrow, \leftrightarrow$.

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 $\varphi := \varphi'$.

Korrektheit und Vollständigkeit

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

Äquivalenz. Folgt aus dem Ersetzungslemma.

Terminierung. Wir definieren eine Funktion

$$h : \text{AL} \rightarrow \mathbb{N},$$

die die Höhe einer Formel angibt, wie folgt:

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

Die Höhe 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 Höhe 1 hat.

Beweis des Lemmas

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

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

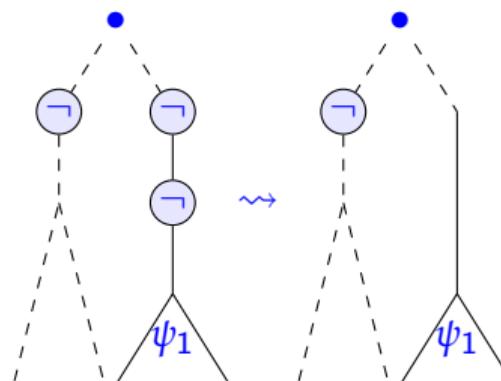
Beweis.

Angenommen, $\psi := \neg\neg\psi_1$.

Wir ersetzen also ψ durch ψ_1 .

Dadurch erhöht sich die Tiefe der restlichen Negationsformeln nicht.

Die Zahl dieser Formeln reduziert sich um 2 und daher $f(\varphi') < f(\varphi)$.



Beweis des Lemmas

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

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, $\psi := \neg(\psi_1 \vee \psi_2)$ oder $\psi := \neg(\psi_1 \wedge \psi_2)$.

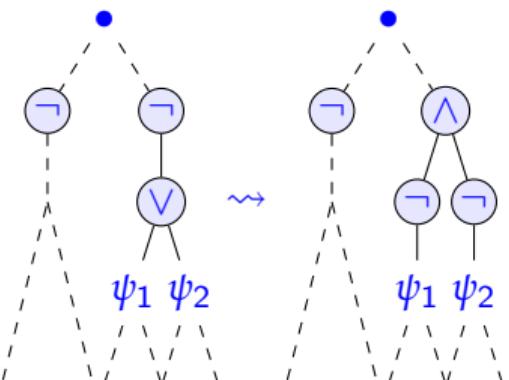
Dann bleibt die Tiefe aller Negationsformeln außer ψ gleich.

In der Summe wird $3^{h(\psi)}$ durch

$$3^{h(\neg\psi_1)} + 3^{h(\neg\psi_2)} \leq 2 \cdot 3^{h(\psi)-1} < 3^{h(\psi)}$$

ersetzt.

Also gilt $f(\varphi') < f(\varphi)$. □



Negationsnormalform

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

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

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

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

Normalformen

Definition. Ein Literal L ist eine Aussagenvariable $X \in \text{AVar}$ oder deren Negation $\neg X$.

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

Normalformen

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

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

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

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

Theorem.

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.

Äquivalenzen.

1. $\neg\neg\varphi \equiv \varphi$
2. $\varphi \rightarrow \psi \equiv \neg\varphi \vee \psi$
3. $\varphi \leftrightarrow \psi \equiv (\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$
4. $\neg(\varphi \wedge \psi) \equiv \neg\varphi \vee \neg\psi$
 $\neg(\varphi \vee \psi) \equiv \neg\varphi \wedge \neg\psi$
5. $\psi \wedge (\varphi \vee \vartheta) \equiv (\psi \wedge \varphi) \vee (\psi \wedge \vartheta)$
 $\psi \vee (\varphi \wedge \vartheta) \equiv (\psi \vee \varphi) \wedge (\psi \vee \vartheta)$
6. $\psi \wedge (\psi \vee \varphi) \equiv \psi \vee (\psi \wedge \varphi) \equiv \psi$
7. $\psi \wedge \varphi \equiv \varphi \wedge \psi$
 $\psi \vee \varphi \equiv \varphi \vee \psi$
8. $\psi \wedge (\varphi \wedge \vartheta) \equiv (\psi \wedge \varphi) \wedge \vartheta$
 $\psi \vee (\varphi \vee \vartheta) \equiv (\psi \vee \varphi) \vee \vartheta$

Erinnerung: Boolesche Funktionen

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

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

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

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

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

Wir definieren nun die Funktion f durch die Formel

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

Beispiel. $\varphi := (X_1 \wedge X_2) \vee (X_1 \wedge X_3) \vee (X_2 \wedge X_3)$

$f_{\varphi}(1, 0, 1) = 1$	$f_{\varphi}(0, 1, 1) = 1$
$f_{\varphi}(1, 1, 0) = 1$	$f_{\varphi}(1, 1, 1) = 1$
$f_{\varphi}(v_1, v_2, v_3) = 0$	sonst

$f_{\varphi}(\bar{v})$: Mehrheitsfunktion

Für $(v_1, v_2, v_3) = (0, 1, 1)$ ist
 $\varphi_{0,1,1} = (\neg X_1 \wedge X_2 \wedge X_3)$.

$\varphi_f(X_1, X_2, X_3) :=$	
$(\neg X_1 \wedge X_2 \wedge X_3)$	$(=: \varphi_{0,1,1})$
\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	
$(X_1 \wedge X_2 \wedge X_3)$	$(=: \varphi_{1,1,1})$

Normalformen

Theorem.

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

Beweis.

1. Teil 1 folgt aus dem Beweis der Äquivalenz zu Booleschen Funktionen, da die dort konstruierten Formeln in disjunktiver Normalform sind.
2. Sei $\varphi \in AL$.

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

Mit Hilfe der de Morganschen Gesetze erhält man

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

Normalformen

Bemerkung.

- Formeln in **disjunktiver Normalform** können sehr effizient auf Erfüllbarkeit getestet werden.
- Allerdings gibt es Formeln $\varphi_n \in \text{AL}$, 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 $\varphi \in \text{AL}$ in Polynomialzeit eine Formel $\psi \in \text{AL}$ in **konjunktiver Normalform** konstruiert werden, so dass
$$\varphi \text{ genau dann erfüllbar ist, wenn } \psi \text{ erfüllbar ist.}$$

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

4.1 Algorithmische Logikprobleme

Algorithmische Logikprobleme

Algorithmische Probleme in der Logik.

Auswertungsproblem. Gegeben eine Belegung β und eine Formel $\varphi \in AL$, entscheide ob $\beta \models \varphi$.

Äquivalenzproblem. Gegeben $\varphi, \psi \in AL$, entscheide ob $\varphi \equiv \psi$.

Semantische Folgerung. Gegeben $\Phi \subseteq AL$ und $\psi \in AL$, entscheide ob $\Phi \models \psi$.

Erfüllbarkeitsproblem. Gegeben $\varphi \in AL$, entscheide, ob φ erfüllbar ist.
(Berechne ggf. eine erfüllende Belegung.)

Allgemeingültigkeitsproblem. Gegeben $\varphi \in AL$, entscheide, ob φ allgemeingültig ist.

Zeige. Diese Probleme können alle auf nur zwei Probleme reduziert werden: *Auswertungsproblem* und *Erfüllbarkeitsproblem*

Zentrale Begriffe

Folgerung $\varphi \models \psi$: für alle β gilt:
Wenn $\beta \models \varphi$, dann $\beta \models \psi$.

Äquivalenz $\varphi \equiv \psi$: für alle β gilt:
 $\beta \models \varphi$ gdw. $\beta \models \psi$.

β erfüllt φ , ist Modell von φ ,
kurz $\beta \models \varphi$, wenn $\llbracket \varphi \rrbracket^\beta = 1$.

φ ist erfüllbar, wenn es Belegung β gibt, die φ erfüllt.

Andernfalls ist φ unerfüllbar.

φ allgemeingültig wenn jede passende Belegung φ erfüllt.

Reduktion auf Erfüllbarkeit

Äquivalenzproblem. Gegeben: Formeln $\varphi, \psi \in \text{AL}$. Frage: Gilt $\varphi \equiv \psi$?

Das Problem kann leicht auf semantische Folgerung reduziert werden.

Denn: $\varphi \equiv \psi$ gdw. $\varphi \models \psi$ und $\psi \models \varphi$.

Wenn wir Folgerungen entscheiden können, dann also auch Äquivalenz.

Folgerung. Gegeben: $\Phi \subseteq \text{AL}$ und Formel $\psi \in \text{AL}$. Frage: Gilt $\Phi \models \psi$?

Das Problem kann leicht auf Erfüllbarkeit reduziert werden.

Denn: $\Phi \models \psi$ gdw. $\Phi \cup \{\neg\psi\}$ unerfüllbar.

Wenn wir Erfüllbarkeit entscheiden können, dann also auch Folgerung.

Allgemeingültigkeit. Gegeben: $\varphi \in \text{AL}$. Frage: Ist φ allgemeingültig?

Das Problem kann leicht auf Erfüllbarkeit reduziert werden.

Denn φ allgemeingültig gdw. $\neg\varphi$ unerfüllbar.

Zentrale Begriffe

Folgerung $\varphi \models \psi$: für alle β gilt:
Wenn $\beta \models \varphi$, dann $\beta \models \psi$.

Äquivalenz $\varphi \equiv \psi$: für alle β gilt:
 $\beta \models \varphi$ gdw. $\beta \models \psi$.

β erfüllt φ , ist Modell von φ ,
kurz $\beta \models \varphi$, wenn $\llbracket \varphi \rrbracket^\beta = 1$.

φ ist erfüllbar, wenn es Belegung β gibt,
die φ erfüllt.
Andernfalls ist φ unerfüllbar.

φ allgemeingültig wenn jede passende Belegung φ erfüllt.

Algorithmische Logikprobleme

Wir brauchen also nur noch zwei algorithmische Probleme zu lösen.

Algorithmische Probleme in der Logik.

Auswertungsproblem.

Gegeben eine Belegung β und eine Formel $\varphi \in AL$, entscheide ob $\beta \models \varphi$.

Lösung. Das Auswertungsproblem der Aussagenlogik kann sehr leicht gelöst werden, z.B. durch einen einfachen rekursiven Algorithmus entsprechend der Definition der Semantikfunktion $[\![\varphi]\!]^\beta$.

Erfüllbarkeitsproblem.

Gegeben $\varphi \in AL$, entscheide, ob φ erfüllbar ist.

(Berechne ggf. eine erfüllende Belegung.)

Lösung. In der Theorie? Schwierig. In der Praxis? Machbar.

Erfüllbarkeitstest

Wir brauchen also schnelle Verfahren um Formeln auf (Un-)Erfüllbarkeit zu testen.

Verfahren zum Test auf (Un-)Erfüllbarkeit von AL-Formeln.

- Wahrheitstafeln
- **Resolution**
- **Der DPLL Algorithmus**
- **Der aussagenlogische Sequenzenkalkül**
-

Das Wahrheitstafelverfahren

Das Wahrheitstafelverfahren.

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

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.

4.2 Einführung in die Resolution

Der Resolutionskalkül

Der **Resolutionskalkül** ist ein Verfahren um die Unerfüllbarkeit von Formeln in **konjunktiver Normalform** zu beweisen.

Der Kalkül enthält nur eine einzige Regel und lässt sich daher sehr einfach implementieren.

Wir werden uns zunächst auf endliche Formelmengen beschränken.

Danach werden wir einen Satz beweisen, der uns in bestimmten Situationen auch die Behandlung unendlicher Formelmengen erlaubt.

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.

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

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

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., $Y \vee Z \vee V \rightsquigarrow \{Y, Z, V\}.$

Klauseln

Definition.

- Eine **Klausel** ist eine endliche Menge von Literalen.
- Zu $\varphi := \bigwedge_{i=1}^n \bigvee_{j=1}^{m_i} L_{i,j}$ definieren wir Menge $\mathcal{C}(\varphi)$ von Klauseln:
 - zu $\bigvee_{j=1}^{m_i} L_{i,j}$ definieren wir $C_i := \{L_{i,j} : 1 \leq j \leq m_i\}$
 - $\mathcal{C}(\varphi) := \{C_1, \dots, C_n\}$.
- Umgekehrt, entspricht jeder Menge $\mathcal{C} := \{C_1, \dots, C_n\}$ von Klauseln

$$C_i := \{L_{i,j} : 1 \leq j \leq m_i\}$$
 die Formel $\varphi(\mathcal{C}) := \bigwedge_{i=1}^n \bigvee_{j=1}^{m_i} L_{i,j}$.
- Falls $\mathcal{C} := \emptyset$, definieren wir $\varphi(\mathcal{C}) := \top$.
- Die **leere Klausel** wird mit \square bezeichnet und wir definieren $\varphi(\square) := \perp$.

Beispiel.

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

$$C_1 := \{X, Y\}$$

$$C_2 := \{\neg X, Z\}$$

$$\mathcal{C}(\varphi) := \{C_1, C_2\}.$$

Formeln vs. Klauselmengen

Notation. Wir erweitern Notation für Formeln auf Klauselmengen.

- Für eine Belegung β und Klauselmenge \mathcal{C} schreiben wir $\beta \models \mathcal{C}$ für $\beta \models \varphi(\mathcal{C})$
- Wir schreiben $\mathcal{C} \models \mathcal{C}$ falls jede \mathcal{C} erfüllende Belegung auch \mathcal{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.

Beobachtung. Eine Belegung β erfüllt eine Klauselmenge \mathcal{C} , wenn jede Klausel $C \in \mathcal{C}$ ein Literal L enthält, so dass $\llbracket L \rrbracket^\beta = 1$.

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

4.3 Resolution

Resolution

Definition. Seien C, C_1, C_2 Klauseln.

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

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

Wir sagen, dass C_1 und C_2 **resolviert** werden. Die Menge der Resolventen von C_1 und C_2 bezeichnen wir mit $\text{Res}(C_1, C_2)$.

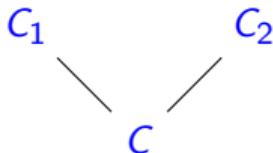
Erinnerung.

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

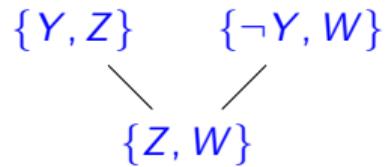
Klausel $C := \{L_1, \dots, L_n\}$ entspricht $\varphi(C) := \bigvee_{i=1}^n L_i$

Klauselmenge $C := \{C_1, \dots, C_n\}$ entspricht $\varphi(C) := \bigwedge_{i=1}^n \varphi(C_i)$.

Graphische Darstellung



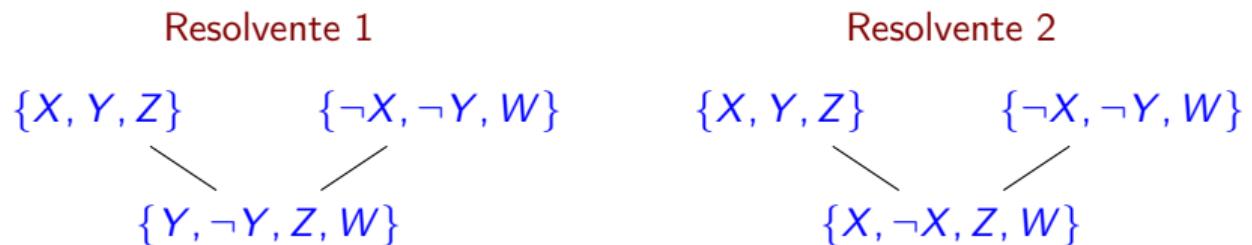
Beispiel



Bemerkung

Zwei Klauseln können mehr als eine Resolvente haben.

Beispiel. Für $C_1 := \{X, Y, Z\}$ und $C_2 := \{\neg X, \neg Y, W\}$ gilt:



Resolvente.
 C_1, C_2 Klauseln,

L Literal
mit $L \in C_1, \bar{L} \in C_2$.

Resolvente $C =$
 $C_1 \setminus \{L\} \cup C_2 \setminus \{\bar{L}\}$.

Dies ist aber ein degenerierter Fall, der im weiteren keine Rolle spielen wird.

Insbesondere ist in diesem Fall die Resolvente immer allgemeingültig, da sie ein Literal und sein duales Literal enthält.

Aussagenlogische Resolution

Lemma.

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

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

Beweis. Wir zeigen zunächst, dass $\{C_1, C_2\} \models C$.

Zu zeigen: Jede Belegung β mit $\beta \models \{C_1, C_2\}$ erfüllt auch C .

Sei β eine Belegung mit $\beta \models \{C_1, C_2\}$ und L das resolierte Literal.

Resolvente.
 C_1, C_2 Klauseln,
 L Literal
mit $L \in C_1, \bar{L} \in C_2$.
Resolvente $C =$
 $C_1 \setminus \{L\} \cup C_2 \setminus \{\bar{L}\}$.

Fall 1: $[L]^\beta = 1$

Fall 2: $[L]^\beta = 0$

Es gilt $\beta \models C_2$.

Also existiert $L' \in C_2 \setminus \{L\}$ mit $[L']^\beta = 1$.

Da nach Definition $L' \in C$, folgt $\beta \models C$.

Es gilt $\beta \models C_1$.

Also existiert $L' \in C_1 \setminus \{L\}$ mit $[L']^\beta = 1$.

Da nach Definition $L' \in C$, folgt $\beta \models C$.

In beiden Fällen gilt also $\beta \models C$.

Aussagenlogische Resolution

Lemma.

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 (Forts.). Wir zeigen nun, dass \mathcal{C} und $\mathcal{C} \cup \{C\}$ äquivalent sind.

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

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

\Leftarrow : Wenn $\beta \models \mathcal{C} \cup \{C\}$, dann $\beta \models \mathcal{C}$.

\Rightarrow : Bereits gesehen: $\{C_1, C_2\} \models C$. Da $C_1, C_2 \in \mathcal{C}$, folgt $\mathcal{C} \models \mathcal{C} \cup \{C\}$.

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

Resolutionsableitungen

Definition.

1. Eine **Resolutionsableitung** einer Klausel C aus einer Klauselmenge \mathcal{C} ist eine Sequenz (C_1, \dots, C_n) , so dass

- $C_n = C$ und

- für alle $1 \leq i \leq n$ gilt

$$C_i \in \mathcal{C} \quad \text{oder} \quad \text{es gibt } j, k < i \text{ mit } C_i \in \text{Res}(C_j, C_k).$$

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 .

Resolvente.

C_1, C_2 Klauseln,

L Literal

mit $L \in C_1, \bar{L} \in C_2$.

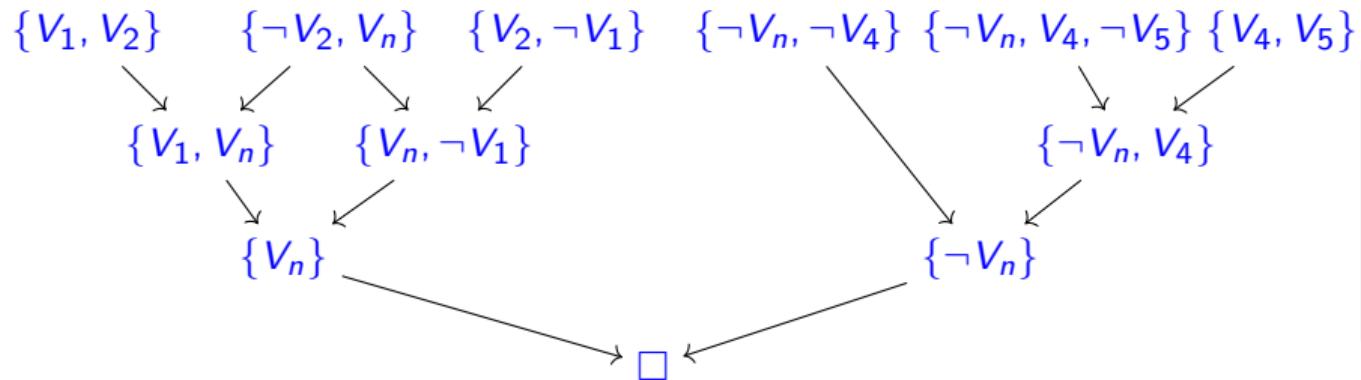
Resolvente $C =$

$C_1 \setminus \{L\} \cup C_2 \setminus \{\bar{L}\}$.

Beispiel einer Resolutionswiderlegung

$$\varphi := (V_1 \vee V_2) \wedge (\neg V_2 \vee V_n) \wedge (V_2 \vee \neg V_1) \wedge (\neg V_n \vee \neg V_4) \wedge (\neg V_n \vee V_4 \vee \neg V_5) \wedge (V_4 \vee V_5)$$

Klauselform.



Resolutionsableitung von \square .

$$\begin{aligned}
 & (\{V_1, V_2\}, \quad \{\neg V_2, V_n\}, \quad \{V_1, V_n\}, \quad \{V_2, \neg V_1\}, \quad \{V_n, \neg V_1\}, \quad \{V_n\}, \\
 & \quad \{V_4, V_5\}, \quad \{\neg V_n, V_4, \neg V_5\}, \quad \{\neg V_n, V_4\}, \quad \{\neg V_n, \neg V_4\}, \quad \{\neg V_n\}, \quad \square)
 \end{aligned}$$

Res.wid. von \mathcal{C} .
Sequenz (C_1, \dots, C_n) :
 $C_n = C$ und
für alle $1 \leq i \leq n$ gilt
 $C_i \in \mathcal{C}$ oder
es gibt $j, k < i$ mit
 $C_i \in \text{Res}(C_j, C_k)$.

Resolvente.
 C_1, C_2 Klauseln,
 L Literal
mit $L \in C_1, \bar{L} \in C_2$.
Resolvente $C =$
 $C_1 \setminus \{L\} \cup C_2 \setminus \{\bar{L}\}$.

Der aussagenlogische Resolutionskalkül

Theorem. Eine Menge \mathcal{C} von Klauseln hat genau dann eine Resolutionswiderlegung, wenn \mathcal{C} unerfüllbar ist. D.h. es gilt

$$\mathcal{C} \vdash_R \square \quad \text{gdw.} \quad \mathcal{C} \text{ ist unerfüllbar} \quad \text{gdw.} \quad \mathcal{C} \models \perp$$

Lemma. (Korrektheit des Resolutionskalküls)

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

Lemma. (Vollständigkeit des Resolutionskalküls)

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

4.4 Korrektheit und Vollständigkeit der Resolution

Der aussagenlogische Resolutionskalkül

Theorem. Eine Menge \mathcal{C} von Klauseln hat genau dann eine Resolutionswiderlegung, wenn \mathcal{C} unerfüllbar ist. D.h. es gilt

$$\mathcal{C} \vdash_R \square \quad \text{gdw.} \quad \mathcal{C} \text{ ist unerfüllbar} \quad \text{gdw.} \quad \mathcal{C} \models \perp$$

Lemma. (Korrektheit des Resolutionskalküls)

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

Lemma. (Vollständigkeit des Resolutionskalküls)

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

Korrektheit des Resolutionskalküls

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

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

Korollar. (Korrektheit des Resolutionskalküls)

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

Res.wid. von \mathcal{C} .
 Sequenz (C_1, \dots, C_n) :
 $C_n = C$ und
 für alle $1 \leq i \leq n$ gilt
 $C_i \in \mathcal{C}$ oder
 es gibt $j, k < i$ mit
 $C_i \in \text{Res}(C_j, C_k)$.

Theorem.

Eine Menge \mathcal{C} von Klauseln hat genau dann eine Resolutionswiderlegung, wenn \mathcal{C} unerfüllbar ist. D.h. es gilt

$$\mathcal{C} \vdash_R \square \quad \text{gdw.} \quad \mathcal{C} \text{ ist unerfüllbar} \quad \text{gdw.} \quad \mathcal{C} \models \perp$$

Korrekteit der Resolution

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

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

Beweis. Sei (C_1, \dots, C_n) eine Resolutionsableitung von C aus \mathcal{C} .

Per Induktion ber i zeigen wir, dass $\mathcal{C} \models C_i$ fr alle $1 \leq i \leq n$.

Fr $i = n$ 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 fr $1, \dots, i$.

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

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

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

Res.wid. von \mathcal{C} .
Sequenz (C_1, \dots, C_n) :
 $C_n = C$ und
fr alle $1 \leq i \leq n$ gilt
 $C_i \in \mathcal{C}$ oder
es gibt $j, k < i$ mit
 $C_i \in \text{Res}(C_j, C_k)$.

Resolvente.
 C_1, C_2 Klauseln,
 L Literal
mit $L \in C_1, \bar{L} \in C_2$.
Resolvente $C =$
 $C_1 \setminus \{L\} \cup C_2 \setminus \{\bar{L}\}$.

Lemma.
Sei \mathcal{C} Klauselmenge,
 $C_1, C_2 \in \mathcal{C}$ und
 $C \in \text{Res}(C_1, C_2)$.
Dann gilt:
 $\{C_1, C_2\} \models C$ und
 $C \equiv \mathcal{C} \cup \{C\}$

Korrektheit des Resolutionskalküls

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

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

Korollar. (Korrektheit des Resolutionskalküls)

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

Res.wid. von \mathcal{C} .
Sequenz (C_1, \dots, C_n) :
 $C_n = C$ und
für alle $1 \leq i \leq n$ gilt
 $C_i \in \mathcal{C}$ oder
es gibt $j, k < i$ mit
 $C_i \in \text{Res}(C_j, C_k)$.

Theorem.

Eine Menge \mathcal{C} von Klauseln hat genau dann eine Resolutionswiderlegung, wenn \mathcal{C} unerfüllbar ist. D.h. es gilt

$$\mathcal{C} \vdash_R \square \quad \text{gdw.} \quad \mathcal{C} \text{ ist unerfüllbar} \quad \text{gdw.} \quad \mathcal{C} \models \perp$$

Der aussagenlogische Resolutionskalkül

Theorem. Eine Menge \mathcal{C} von Klauseln hat genau dann eine Resolutionswiderlegung, wenn \mathcal{C} unerfüllbar ist. D.h. es gilt

$$\mathcal{C} \vdash_R \square \quad \text{gdw.} \quad \mathcal{C} \text{ ist unerfüllbar} \quad \text{gdw.} \quad \mathcal{C} \models \perp$$

Lemma. (Korrektheit des Resolutionskalküls)

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

Lemma. (Vollständigkeit des Resolutionskalküls)

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

Vollständigkeit und Korrektheit der Resolution

Lemma. (Vollständigkeit des Resolutionskalküls)

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

Dazu beweisen wir zunächst die folgende Behauptung.

Aus der Behauptung folgt sofort die Vollständigkeit der Resolution für endliche Formelmengen.

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

Beweis der Behauptung

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

Beweis. Der Beweis der Behauptung wird per Induktion über n geführt.

Induktionsbasis $n = 1$.

Wir wissen: \mathcal{C} ist unerfüllbar und enthält keine Variablen.

Es gibt nur zwei Klauselmengen ohne Variablen: $\{\}$ und $\{\square\}$.

Nach Voraussetzung ist \mathcal{C} unerfüllbar.

Also ist $\mathcal{C} := \{\square\}$ und somit existiert eine Resolutionswiderlegung.



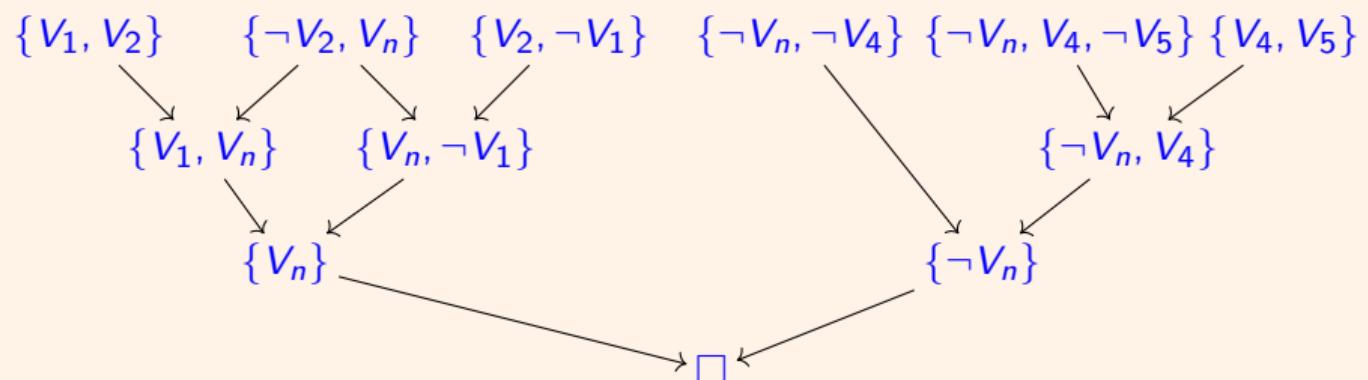
Beispiel einer Resolutionswiderlegung

Induktionsschritt $n \rightarrow n+1$.

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

$$\mathcal{C} := \{V_1, V_2\}, \quad \{\neg V_2, V_n\}, \quad \{V_2, \neg V_1\}, \quad \{\neg V_n, \neg V_4\}, \quad \{\neg V_n, V_4, \neg V_5\}, \quad \{V_4, V_5\}$$

Resolutionswiderlegung.



Beweis der Behauptung

Induktionsschritt $n \rightarrow n+1$.

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

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

D.h., man erhält z.B. \mathcal{C}^+ indem

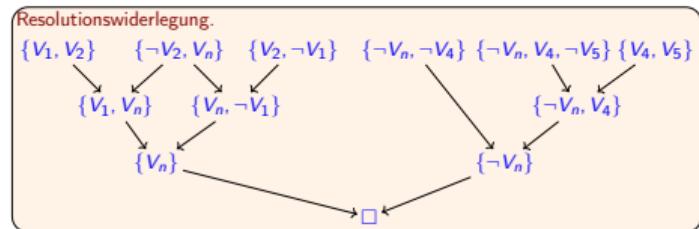
- alle Klauseln, die V_n enthalten entfernt werden und
- $\neg V_n$ aus den anderen entfernt wird.

Behauptung. \mathcal{C}^+ und \mathcal{C}^- sind beide unerfüllbar.

Beweis. Angenommen \mathcal{C}^+ wäre erfüllbar, z.B. durch $\beta \models \mathcal{C}^+$.

Dann würde $\beta' := \beta \cup \{V_n \mapsto 1\}$ die Menge \mathcal{C} erfüllen.

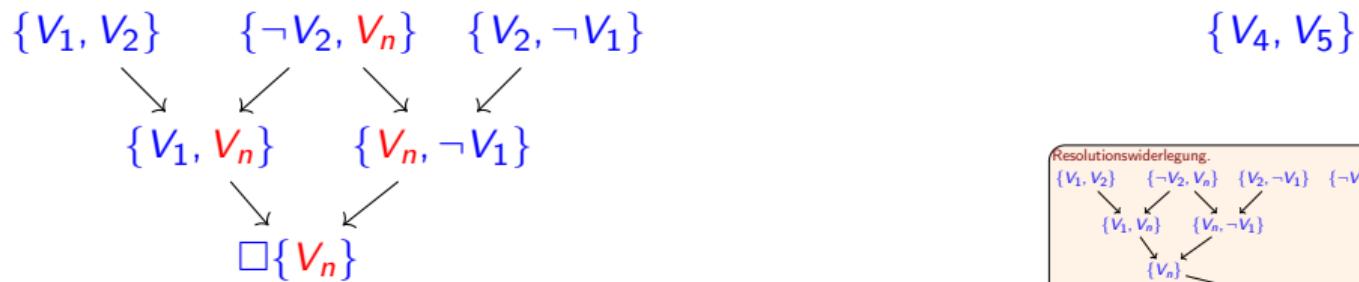
Analog für \mathcal{C}^- . □



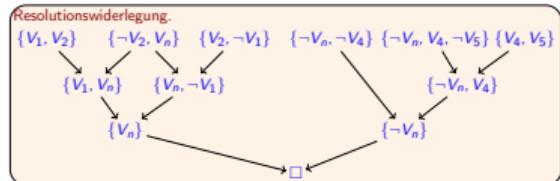
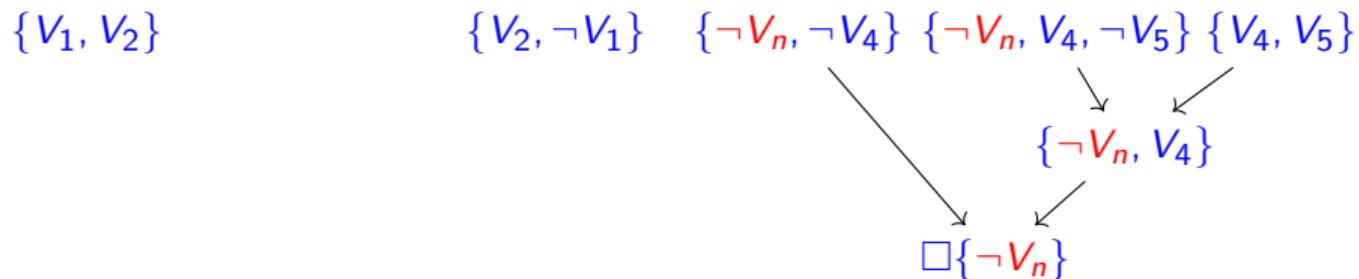
Beispiel einer Resolutionswiderlegung

$$\mathcal{C} := \{V_1, V_2\}, \quad \{\neg V_2, V_n\}, \quad \{V_2, \neg V_1\}, \quad \{\neg V_n, \neg V_4\}, \quad \{\neg V_n, V_4, \neg V_5\}, \quad \{V_4, V_5\}$$

Die Menge \mathcal{C}^- .



Die Menge \mathcal{C}^+ .



Beweis der Behauptung (Forts.)

$n \rightarrow n+1$. Sei \mathcal{C} eine unerfüllbare Klauselmenge in den Variablen $\{V_1, \dots, V_n\}$.

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

Schon gezeigt. \mathcal{C}^+ und \mathcal{C}^- sind beide unerfüllbar. \square

Nach IV ex. Res.-Abl. (C_1, \dots, C_s) und (D_1, \dots, D_t) von $C_s = D_t = \square$ aus \mathcal{C}^+ bzw. \mathcal{C}^- .

Falls (C_1, \dots, C_s) schon eine Ableitung von \square aus \mathcal{C} ist, sind wir fertig.

Wenn nicht, 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, \dots, C'_s) von $\{\neg V_n\}$ aus \mathcal{C} .

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

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

Also ist $(C'_1, \dots, C'_s, D'_1, \dots, D'_t, \square)$ eine Resolutionswiderlegung von \mathcal{C} . \square

Vollständigkeit des Resolutionskalküls

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

Lemma. (Vollständigkeit des Resolutionskalküls im Endlichen)

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

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

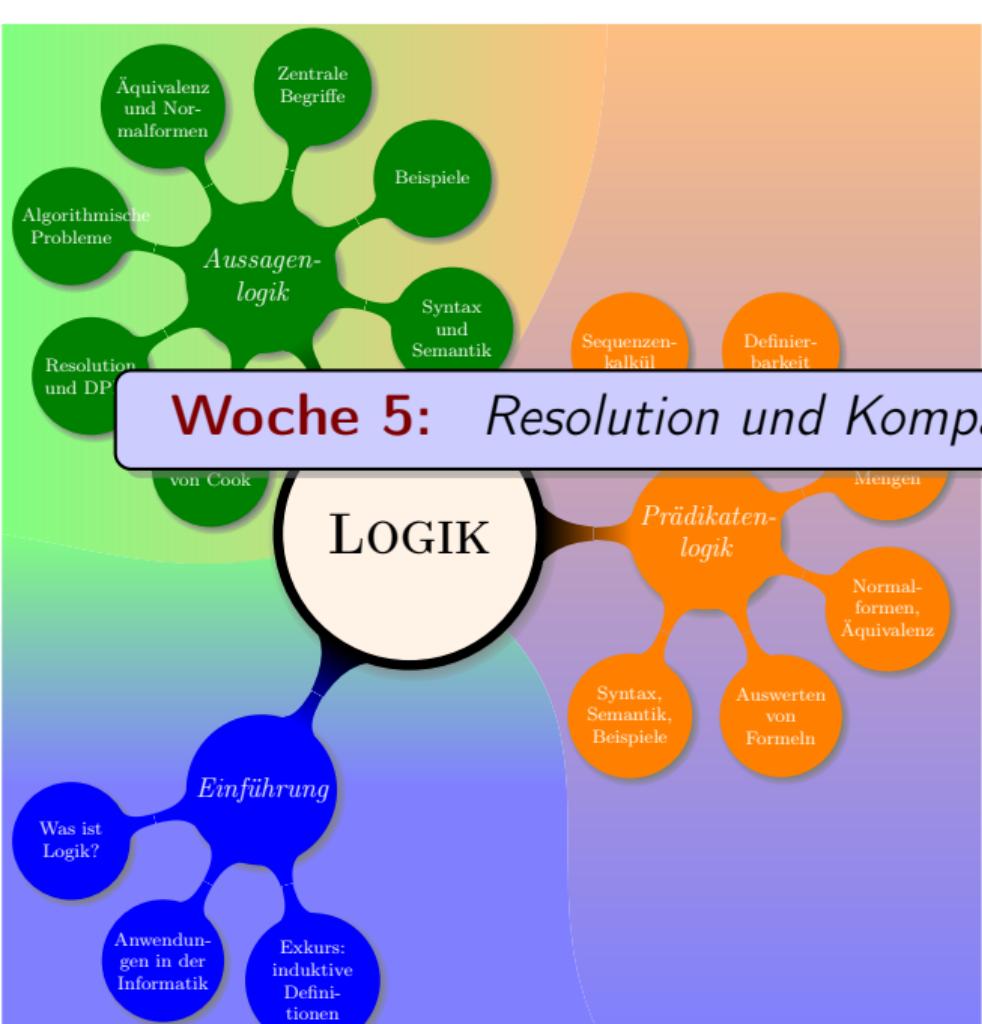
Da \mathcal{C} endlich, dann enthält sie nur endlich viele Variablen.

O.B.d.A. benutzt \mathcal{C} nur Variablen aus der Menge $\{V_1, \dots, V_n\}$ für ein $n \in \mathbb{N}$.

Der Beweis folgt damit sofort aus der Behauptung. □

Vollständigkeit im Unendlichen? Für unendliche Klauselmengen ist die bewiesene Behauptung nicht stark genug.

Hierzu brauchen wir noch den **Kompaktheitssatz**.



Nov.	17	18	19	20	21	22	23	<i>Einführung</i>
	24	25	26	27	28	29	30	<i>Aussagenlogik</i>
	31	1	2	3	4	5	6	<i>Normalformen</i>
	7	8	9	10	11	12	13	<i>Resolution</i>
Dez.	14	15	16	17	18	19	20	<i>Kompaktheit</i>
	21	22	23	24	25	26	27	<i>DPLL, Satz von Cook</i>
	28	29	30	1	2	3	4	<i>Strukturen und FO</i>
	5	6	7	8	9	10	11	<i>Prädikatenlogik</i>
Jan.	12	13	14	15	16	17	18	<i>Komplexität von FO</i>
	19	20	21	22	23	24	25	<i>Weihnachten</i>
	26	27	28	29	30	31	1	<i>Neujahr</i>
	2	3	4	5	6	7	8	<i>Normalformen</i>
	9	10	11	12	13	14	15	<i>Definierbarkeit</i>
	16	17	18	19	20	21	22	<i>EF-Spiele</i>
Feb.	23	24	25	26	27	28	29	<i>EF-Spiele</i>
	30	31	1	2	3	4	5	<i>Sequenzenkalkül AL</i>
	6	7	8	9	10	11	12	<i>Sequenzenkalkül FO</i>
	13	14	15	16	17	18	19	<i>Ausblick</i>

Wiederholung

Aussagenlogische Resolution

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

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

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

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., $Y \vee Z \vee V \rightsquigarrow \{Y, Z, V\}.$

Resolution

Definition. Seien C, C_1, C_2 Klauseln.

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

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

Wir sagen, dass C_1 und C_2 **resolviert** werden. Die Menge der Resolventen von C_1 und C_2 bezeichnen wir mit $\text{Res}(C_1, C_2)$.

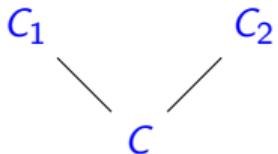
Erinnerung.

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

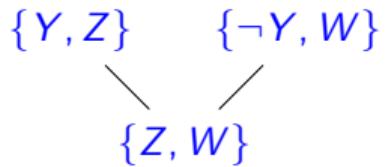
Klausel $C := \{L_1, \dots, L_n\}$ entspricht $\varphi(C) := \bigvee_{i=1}^n L_i$

Klauselmenge $C := \{C_1, \dots, C_n\}$ entspricht $\varphi(C) := \bigwedge_{i=1}^n \varphi(C_i)$.

Graphische Darstellung



Beispiel



Aussagenlogische Resolution

Lemma.

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

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

Beweis. Wir zeigen zunächst, dass $\{C_1, C_2\} \models C$.

Zu zeigen: Jede Belegung β mit $\beta \models \{C_1, C_2\}$ erfüllt auch C .

Sei β eine Belegung mit $\beta \models \{C_1, C_2\}$ und L das resovierte Literal.

Resolvente.
 C_1, C_2 Klauseln,
 L Literal
mit $L \in C_1, \bar{L} \in C_2$.
Resolvente $C =$
 $C_1 \setminus \{L\} \cup C_2 \setminus \{\bar{L}\}$.

Fall 1: $[L]^\beta = 1$

Fall 2: $[L]^\beta = 0$

Es gilt $\beta \models C_2$.

Also existiert $L' \in C_2 \setminus \{L\}$ mit $[L']^\beta = 1$.

Da nach Definition $L' \in C$, folgt $\beta \models C$.

Es gilt $\beta \models C_1$.

Also existiert $L' \in C_1 \setminus \{L\}$ mit $[L']^\beta = 1$.

Da nach Definition $L' \in C$, folgt $\beta \models C$.

In beiden Fällen gilt also $\beta \models C$.

Resolutionsableitungen

Definition.

1. Eine **Resolutionsableitung** einer Klausel C aus einer Klauselmenge \mathcal{C} ist eine Sequenz (C_1, \dots, C_n) , so dass

- $C_n = C$ und

- für alle $1 \leq i \leq n$ gilt

$$C_i \in \mathcal{C} \quad \text{oder} \quad \text{es gibt } j, k < i \text{ mit } C_i \in \text{Res}(C_j, C_k).$$

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 .

Resolvente.

C_1, C_2 Klauseln,

L Literal

mit $L \in C_1, \bar{L} \in C_2$.

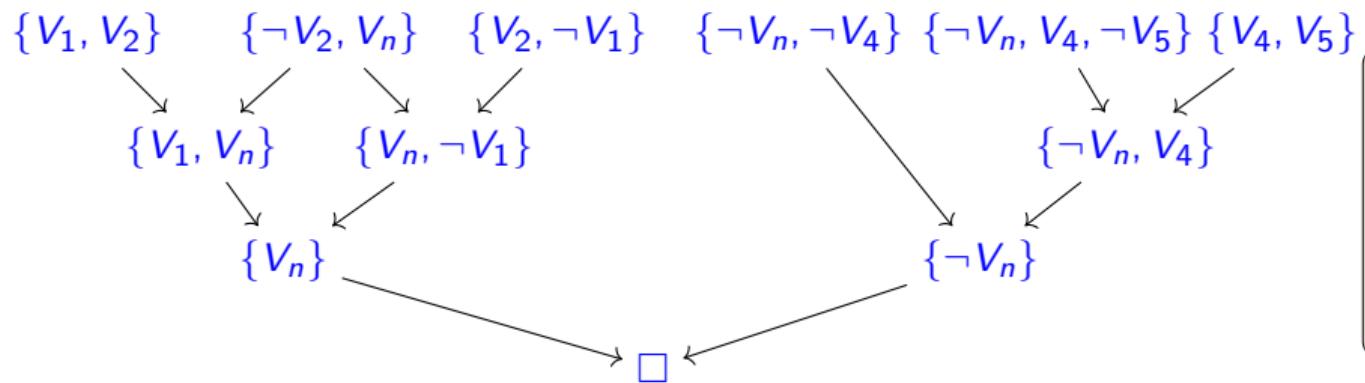
Resolvente $C =$

$C_1 \setminus \{L\} \cup C_2 \setminus \{\bar{L}\}$.

Beispiel einer Resolutionswiderlegung

$$\varphi := (V_1 \vee V_2) \wedge (\neg V_2 \vee V_n) \wedge (V_2 \vee \neg V_1) \wedge (\neg V_n \vee \neg V_4) \wedge (\neg V_n \vee V_4 \vee \neg V_5) \wedge (V_4 \vee V_5)$$

Klauselform.



Resolutionsableitung von \square .

$$\begin{aligned}
 & (\{V_1, V_2\}, \quad \{\neg V_2, V_n\}, \quad \{V_1, V_n\}, \quad \{V_2, \neg V_1\}, \quad \{V_n, \neg V_1\}, \quad \{V_n\}, \\
 & \quad \{V_4, V_5\}, \quad \{\neg V_n, V_4, \neg V_5\}, \quad \{\neg V_n, V_4\}, \quad \{\neg V_n, \neg V_4\}, \quad \{\neg V_n\}, \quad \square)
 \end{aligned}$$

Res.wid. von \mathcal{C} .
Sequenz (C_1, \dots, C_n) :
 $C_n = C$ und
für alle $1 \leq i \leq n$ gilt
 $C_i \in \mathcal{C}$ oder
es gibt $j, k < i$ mit
 $C_i \in \text{Res}(C_j, C_k)$.

Resolvente.
 C_1, C_2 Klauseln,
 L Literal
mit $L \in C_1, \bar{L} \in C_2$.
Resolvente $C = C_1 \setminus \{L\} \cup C_2 \setminus \{\bar{L}\}$.

5.1 Vollständigkeit und Korrektheit der Resolution

Der aussagenlogische Resolutionskalkül

Theorem. Eine Menge \mathcal{C} von Klauseln hat genau dann eine Resolutionswiderlegung, wenn \mathcal{C} unerfüllbar ist. D.h. es gilt

$$\mathcal{C} \vdash_R \square \quad \text{gdw.} \quad \mathcal{C} \text{ ist unerfüllbar} \quad \text{gdw.} \quad \mathcal{C} \models \perp$$

Lemma. (Korrektheit des Resolutionskalküls)

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

Lemma. (Vollständigkeit des Resolutionskalküls)

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

Korrektheit des Resolutionskalküls

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

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

Korollar. (Korrektheit des Resolutionskalküls)

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

Res.wid. von \mathcal{C} .
Sequenz (C_1, \dots, C_n) :
 $C_n = C$ und
für alle $1 \leq i \leq n$ gilt
 $C_i \in \mathcal{C}$ oder
es gibt $j, k < i$ mit
 $C_i \in \text{Res}(C_j, C_k)$.

Theorem.

Eine Menge \mathcal{C} von Klauseln hat genau dann eine Resolutionswiderlegung, wenn \mathcal{C} unerfüllbar ist. D.h. es gilt

$$\mathcal{C} \vdash_R \square \quad \text{gdw.} \quad \mathcal{C} \text{ ist unerfüllbar} \quad \text{gdw.} \quad \mathcal{C} \models \perp$$

Korrektheit der Resolution

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

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

Beweis. Sei (C_1, \dots, C_n) eine Resolutionsableitung von C aus \mathcal{C} .

Per Induktion über i zeigen wir, dass $\mathcal{C} \models C_i$ für alle $1 \leq i \leq n$.

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 $1, \dots, i$.

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

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

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

Res.wid. von \mathcal{C} .
Sequenz (C_1, \dots, C_n) :
 $C_n = C$ und
für alle $1 \leq i \leq n$ gilt
 $C_i \in \mathcal{C}$ oder
es gibt $j, k < i$ mit
 $C_i \in \text{Res}(C_j, C_k)$.

Resolvente.
 C_1, C_2 Klauseln,
 L Literal
mit $L \in C_1, \bar{L} \in C_2$.
Resolvente $C =$
 $C_1 \setminus \{L\} \cup C_2 \setminus \{\bar{L}\}$.

Lemma.
Sei \mathcal{C} Klauselmenge,
 $C_1, C_2 \in \mathcal{C}$ und
 $C \in \text{Res}(C_1, C_2)$.
Dann gilt:
 $\{C_1, C_2\} \models C$ und
 $C \equiv \mathcal{C} \cup \{C\}$

Korrektheit des Resolutionskalküls

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

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

Korollar. (Korrektheit des Resolutionskalküls)

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

Res.wid. von \mathcal{C} .
Sequenz (C_1, \dots, C_n) :
 $C_n = C$ und
für alle $1 \leq i \leq n$ gilt
 $C_i \in \mathcal{C}$ oder
es gibt $j, k < i$ mit
 $C_i \in \text{Res}(C_j, C_k)$.

Theorem.

Eine Menge \mathcal{C} von Klauseln hat genau dann eine Resolutionswiderlegung, wenn \mathcal{C} unerfüllbar ist. D.h. es gilt

$$\mathcal{C} \vdash_R \square \quad \text{gdw.} \quad \mathcal{C} \text{ ist unerfüllbar} \quad \text{gdw.} \quad \mathcal{C} \models \perp$$

Der aussagenlogische Resolutionskalkül

Theorem. Eine Menge \mathcal{C} von Klauseln hat genau dann eine Resolutionswiderlegung, wenn \mathcal{C} unerfüllbar ist. D.h. es gilt

$$\mathcal{C} \vdash_R \square \quad \text{gdw.} \quad \mathcal{C} \text{ ist unerfüllbar} \quad \text{gdw.} \quad \mathcal{C} \models \perp$$

Lemma. (Korrektheit des Resolutionskalküls)

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

Lemma. (Vollständigkeit des Resolutionskalküls)

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

Vollständigkeit und Korrektheit der Resolution

Lemma. (Vollständigkeit des Resolutionskalküls)

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

Dazu beweisen wir zunächst die folgende Behauptung.

Aus der Behauptung folgt sofort die Vollständigkeit der Resolution für endliche Formelmengen.

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

Beweis der Behauptung

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

Beweis. Der Beweis der Behauptung wird per Induktion über n geführt.

Induktionsbasis $n = 1$.

Wir wissen: \mathcal{C} ist unerfüllbar und enthält keine Variablen.

Es gibt nur zwei Klauselmengen ohne Variablen: $\{\}$ und $\{\square\}$.

Nach Voraussetzung ist \mathcal{C} unerfüllbar.

Also ist $\mathcal{C} := \{\square\}$ und somit existiert eine Resolutionswiderlegung.



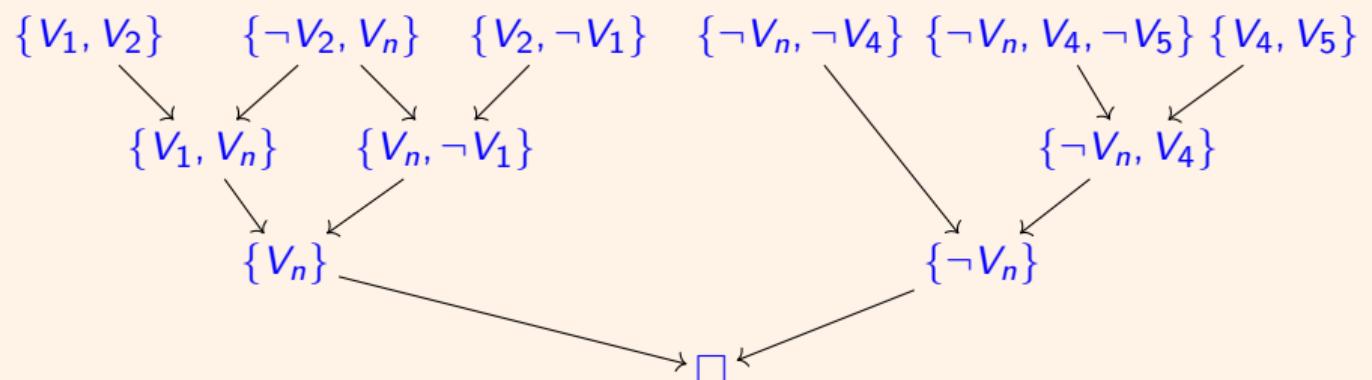
Beispiel einer Resolutionswiderlegung

Induktionsschritt $n \rightarrow n+1$.

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

$$\mathcal{C} := \{V_1, V_2\}, \quad \{\neg V_2, V_n\}, \quad \{V_2, \neg V_1\}, \quad \{\neg V_n, \neg V_4\}, \quad \{\neg V_n, V_4, \neg V_5\}, \quad \{V_4, V_5\}$$

Resolutionswiderlegung.



Beweis der Behauptung

Induktionsschritt $n \rightarrow n+1$.

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

Definiere $\mathcal{C}^+ := \{C \setminus \{\neg V_n\} : C \in \mathcal{C} \text{ und } V_n \notin C\}$

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

D.h., man erhält z.B. \mathcal{C}^+ indem

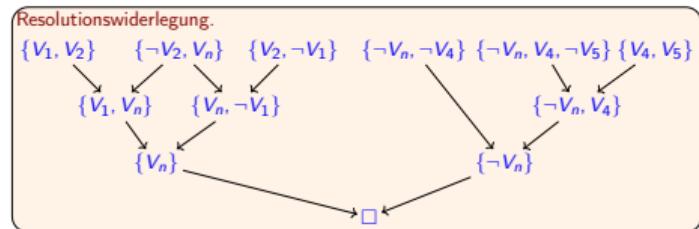
- alle Klauseln, die V_n enthalten entfernt werden und
- $\neg V_n$ aus den anderen entfernt wird.

Behauptung. \mathcal{C}^+ und \mathcal{C}^- sind beide unerfüllbar.

Beweis. Angenommen \mathcal{C}^+ wäre erfüllbar, z.B. durch $\beta \models \mathcal{C}^+$.

Dann würde $\beta' := \beta \cup \{V_n \mapsto 1\}$ die Menge \mathcal{C} erfüllen.

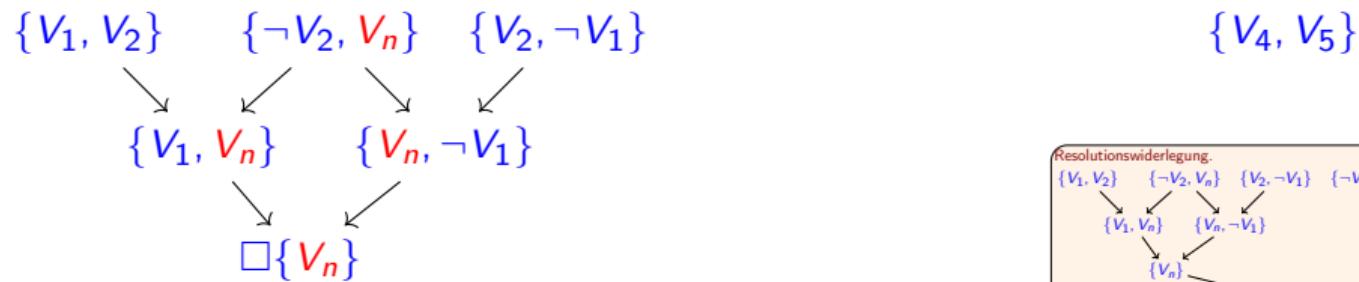
Analog für \mathcal{C}^- . □



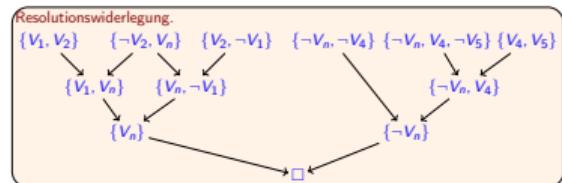
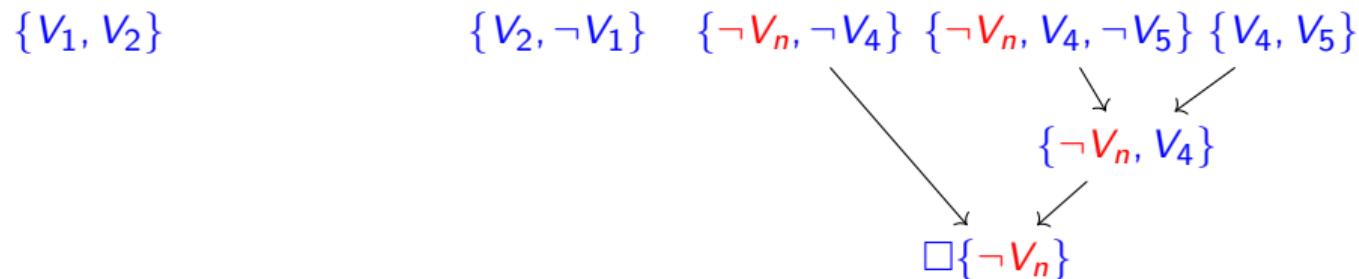
Beispiel einer Resolutionswiderlegung

$$\mathcal{C} := \{V_1, V_2\}, \quad \{\neg V_2, V_n\}, \quad \{V_2, \neg V_1\}, \quad \{\neg V_n, \neg V_4\}, \quad \{\neg V_n, V_4, \neg V_5\}, \quad \{V_4, V_5\}$$

Die Menge \mathcal{C}^- .



Die Menge \mathcal{C}^+ .



Beweis der Behauptung (Forts.)

$n \rightarrow n+1$. Sei \mathcal{C} eine unerfüllbare Klauselmenge in den Variablen $\{V_1, \dots, V_n\}$.

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

Schon gezeigt. \mathcal{C}^+ und \mathcal{C}^- sind beide unerfüllbar. \square

Nach IV ex. Res.-Abl. (C_1, \dots, C_s) und (D_1, \dots, D_t) von $C_s = D_t = \square$ aus \mathcal{C}^+ bzw. \mathcal{C}^- .

Falls (C_1, \dots, C_s) schon eine Ableitung von \square aus \mathcal{C} ist, sind wir fertig.

Wenn nicht, 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, \dots, C'_s) von $\{\neg V_n\}$ aus \mathcal{C} .

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

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

Also ist $(C'_1, \dots, C'_s, D'_1, \dots, D'_t, \square)$ eine Resolutionswiderlegung von \mathcal{C} . \square

Vollständigkeit des Resolutionskalküls

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

Lemma. (Vollständigkeit des Resolutionskalküls im Endlichen)

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

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

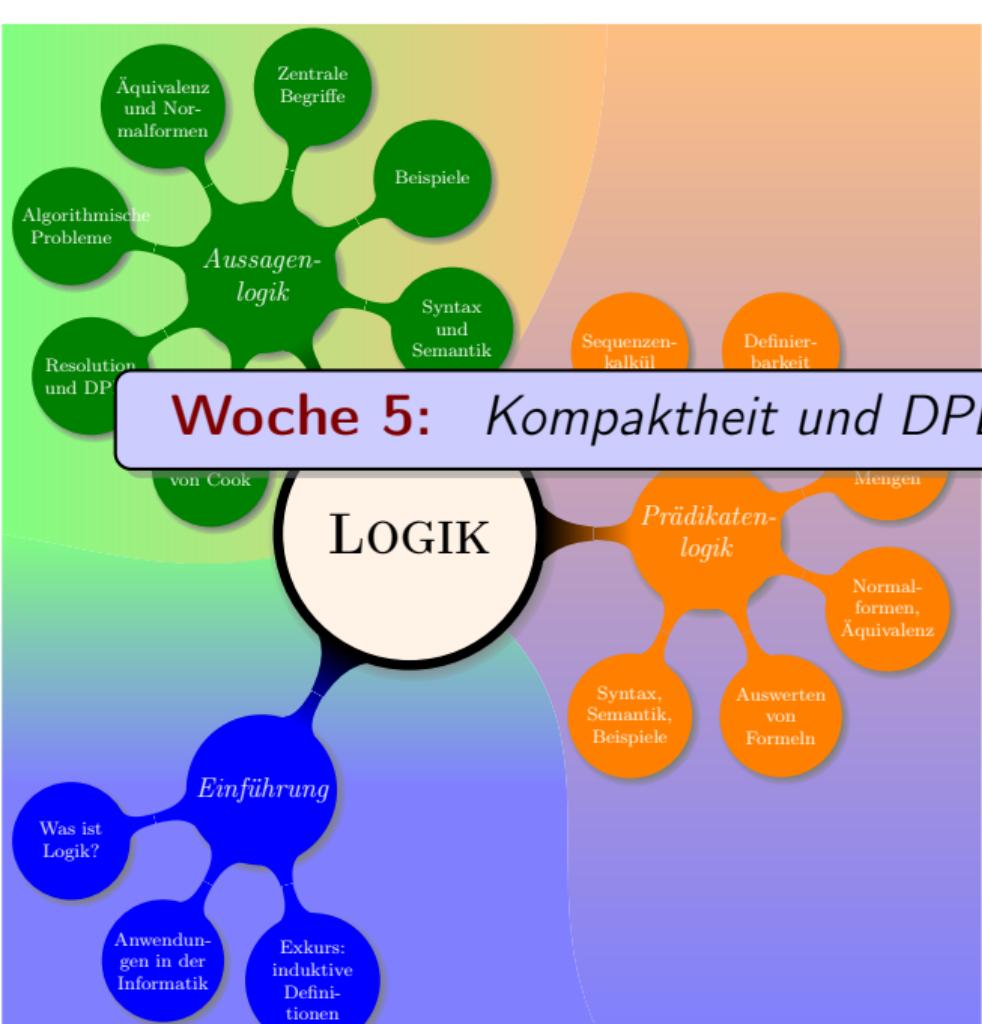
Da \mathcal{C} endlich, enthält \mathcal{C} nur endlich viele Variablen.

O.B.d.A. benutzt \mathcal{C} nur Variablen aus der Menge $\{V_1, \dots, V_n\}$ für ein $n \in \mathbb{N}$.

Der Beweis folgt damit sofort aus der Behauptung. □

Vollständigkeit im Unendlichen? Für unendliche Klauselmengen ist die bewiesene Behauptung nicht stark genug.

Hierzu brauchen wir noch den **Kompaktheitssatz**.



Woche 5: Kompaktheit und DPLL

Nov.	17	18	19	20	21	22	23	<i>Einführung</i>
	24	25	26	27	28	29	30	<i>Aussagenlogik</i>
	31	1	2	3	4	5	6	<i>Normalformen</i>
	7	8	9	10	11	12	13	<i>Resolution</i>
Dez.	14	15	16	17	18	19	20	<i>Kompaktheit</i>
	21	22	23	24	25	26	27	<i>DPLL, Satz von Cook</i>
	28	29	30	1	2	3	4	<i>Strukturen und FO</i>
	5	6	7	8	9	10	11	<i>Prädikatenlogik</i>
	12	13	14	15	16	17	18	<i>Komplexität von FO</i>
Jan.	19	20	21	22	23	24	25	<i>Weihnachten</i>
	26	27	28	29	30	31	1	<i>Neujahr</i>
	2	3	4	5	6	7	8	<i>Normalformen</i>
	9	10	11	12	13	14	15	<i>Definierbarkeit</i>
	16	17	18	19	20	21	22	<i>EF-Spiele</i>
Feb.	23	24	25	26	27	28	29	<i>EF-Spiele</i>
	30	31	1	2	3	4	5	<i>Sequenzenkalkül AL</i>
	6	7	8	9	10	11	12	<i>Sequenzenkalkül FO</i>
	13	14	15	16	17	18	19	<i>Ausblick</i>

5.1 Der Kompaktheitssatz

Der Kompaktheitssatz der Aussagenlogik

Theorem. (Kompaktheits- oder Endlichkeitssatz)

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

1. Φ ist erfüllbar gdw. jede endliche Teilmenge $\Phi' \subseteq \Phi$ erfüllbar ist.
2. $\Phi \models \psi$ gdw. eine endliche Teilmenge $\Phi_0 \subseteq \Phi$ existiert mit $\Phi_0 \models \psi$.

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

Erinnerung.

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 $\beta \models \Phi$.

Beweis des Satzes

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

Behauptung. Eine Menge $\Phi \subseteq AL$ ist genau dann erfüllbar, wenn jede endliche Teilmenge $\Phi' \subseteq \Phi$ erfüllbar ist.

Vorüberlegungen.

1. Offenbar ist die Behauptung trivial, wenn Φ bereits endlich ist.

Sei Φ also eine unendliche Menge aussagenlogischer Formeln.

2. O.B.d.A. 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.

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

Komplektheitssatz

Sei $\Phi \subseteq AL, \psi \in AL$.

1. Φ erfüllbar gdw. alle endlichen $\Phi' \subseteq \Phi$ erfüllbar.
2. $\Phi \models \psi$ gdw. es ex. endl. $\Phi_0 \subseteq \Phi$ mit $\Phi_0 \models \psi$.

Beweis des Satzes: Hinrichtung

Behauptung. Eine Menge $\Phi \subseteq AL$ ist genau dann erfüllbar, wenn jede endliche Teilmenge $\Phi' \subseteq \Phi$ erfüllbar ist.

Hinrichtung.

Wir nehmen an, dass Φ erfüllbar ist.

Dann existiert eine Belegung β , die jede Formel in Φ erfüllt.

Also erfüllt β auch jede endliche Teilmenge von Φ .

Es folgt, dass jede endliche Teilmenge von Φ erfüllbar ist. \dashv

Kompaktheitssatz

Sei $\Phi \subseteq AL$, $\psi \in AL$.

1. Φ erfüllbar gdw. alle endlichen $\Phi' \subseteq \Phi$ erfüllbar.
2. $\Phi \models \psi$ gdw. es ex. endl. $\Phi_0 \subseteq \Phi$ mit $\Phi_0 \models \psi$.

Beweis des Satzes: Rückrichtung

Behauptung. Eine Menge $\Phi \subseteq AL$ ist genau dann erfüllbar, wenn jede endliche Teilmenge $\Phi' \subseteq \Phi$ erfüllbar ist.

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

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

Für all $n \geq 0$ definieren wir

$$\Phi_n := \{\varphi \in \Phi : \text{var}(\varphi) \subseteq \{X_1, \dots, X_n\}\}.$$

Φ_n enthält also alle Formeln aus Φ in denen nur die Variablen X_1, \dots, X_n vorkommen (es müssen aber nicht alle vorkommen).

Beobachtung. Für alle i gilt $\Phi_i \subseteq \Phi_{i+1} \subseteq \Phi$

$$\Phi_0 \subseteq \Phi_1 \subseteq \dots \subseteq \Phi.$$

Beweis des Satzes: Rückrichtung

Bereits gesehen.

$$\Phi_n := \{\varphi \in \Phi : \text{var}(\varphi) \subseteq \{X_1, \dots, X_n\}\}$$

- Es gibt $\leq 2^{2^n}$ paarweise nicht-äquivalente Formeln in n Variablen.
- Da Φ keine paarweise äquivalenten Formeln enthält, gilt $|\Phi_n| \leq 2^{2^n}$.

Nach Voraussetzung gibt es also für jedes $n \geq 0$ eine Belegung β_n mit $\beta_n \models \Phi_n$ und somit auch $\beta_n \models \Phi_i$ für alle $i \leq n$.

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

Wir konstruieren induktiv eine Belegung $\alpha : \{X_1, \dots\} \rightarrow \{0, 1\}$ und Mengen $I_n \subseteq I_0$, so dass für alle n folgende Eigenschaft $(*)$ gilt:

Eigenschaft $(*)$.

- I_n ist unendlich, $\beta_0, \dots, \beta_{n-1} \notin I_n$ und
- für alle $\beta, \beta' \in I_n$ und $j \leq n$ gilt $\beta(X_j) = \beta'(X_j) = \alpha(X_j)$.

Rückrichtung.
Wenn alle endl. $\Phi' \subseteq \Phi$ erfüllbar, dann Φ erfüllbar.

Beweis des Kompaktheitssatzes

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	...
β_1	0	1						
β_2	0	1	1	1	0	0	1	...
β_3	1	1	1	1	0	0	1	...
β_4	0	1	1	1	0	0	1	...
β_5	0	1	1	0	0	0	1	...
β_6	0	0	1	1	0	0	1	...
β_7	0	1	1	1	0	0	1	...
β_8	0	1	1	0	0	0	1	...
β_9	1	0	1	1	0	0	1	...
β_{10}	0	1	0	0	0	0	1	...
β_{11}	0	1	1	0	0	0	1	...
:
α	0	1	1	0	0			...

Konstr. $\alpha : \{X_1, \dots\} \rightarrow \{0, 1\}$ und $I_n \subseteq I_0$, $n \geq 1$:

Für alle n gilt Eigenschaft (*):

- I_n ist unendlich, $\beta_1, \dots, \beta_{n-1} \notin I_n$ und
- für alle $\beta, \beta' \in I_n$ und $j \leq n$ gilt
 $\beta(X_j) = \beta'(X_j) = \alpha(X_j)$.

Induktionsbasis $n = 1$.

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

Setze

$$\alpha(X_1) := t \text{ und}$$

$$I_1 := \{\beta \in I_0 : \beta(X_1) = t \text{ und } \beta \neq \beta_1\}.$$

Offenbar ist (*) erfüllt.

Beweis des Kompaktheitssatzes

	X_1	X_2	X_3	X_4	X_5	X_6	X_7	...
β_1	0	1						
β_2	0	1	1	1	0	0	1	...
β_3	1	1	1	1	0	0	1	...
β_4	0	1	1	1	0	0	1	...
β_5	0	1	1	0	0	0	1	...
β_6	0	0	1	1	0	0	1	...
β_7	0	1	1	1	0	0	1	...
β_8	0	1	1	0	0	0	1	...
β_9	1	0	1	1	0	0	1	...
β_{10}	0	1	0	0	0	0	1	...
β_{11}	0	1	1	0	0	0	1	...
:
α	0	1	1	0	0			...

Konstr. $\alpha : \{X_1, \dots\} \rightarrow \{0, 1\}$ und $I_n \subseteq I_0$, $n \geq 1$:

Für alle n gilt Eigenschaft (*):

- I_n ist unendlich, $\beta_1, \dots, \beta_{n-1} \notin I_n$ und
- für alle $\beta, \beta' \in I_n$ und $j \leq n$ gilt
 $\beta(X_j) = \beta'(X_j) = \alpha(X_j)$.

Induktionsvoraussetzung. Für $i < n$ seien $I_i, \alpha(X_i)$ schon konstruiert so dass (*) gilt.

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

Setze

$$\alpha(X_n) := t \text{ und}$$

$$I_n := \{\beta \in I_{n-1} : \beta(X_n) = t \text{ und } \beta \neq \beta_{n-1}\}.$$

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 $\beta_i \models \varphi$ für alle $i \geq n$, insb. 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 $i \leq n$ gilt $\alpha(X_i) = \beta(X_i)$ und $\beta \models \varphi$, folgt
also $\alpha \models \varphi$. □

Konstr. $\alpha : \{X_1, \dots\} \rightarrow \{0, 1\}$ und $I_n \subseteq I_0$, $n \geq 1$:

Für alle n gilt Eigenschaft $(*)$:

- I_n ist unendlich, $\beta_1, \dots, \beta_{n-1} \notin I_n$ und
- für alle $\beta, \beta' \in I_n$ und $j \leq n$ gilt
 $\beta(X_j) = \beta'(X_j) = \alpha(X_j)$.

Der Kompaktheitssatz der Aussagenlogik

Theorem. (Kompaktheits- oder Endlichkeitssatz)

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

1. Φ ist erfüllbar gdw. jede endliche Teilmenge $\Phi' \subseteq \Phi$ erfüllbar ist.
2. $\Phi \models \psi$ gdw. eine endliche Teilmenge $\Phi_0 \subseteq \Phi$ existiert mit $\Phi_0 \models \psi$.

Beweis von Teil 2. aus Teil 1.

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 $\Phi_0 \setminus \{\neg\psi\} \models \psi$.
- Andernfalls ist $\Phi_0 \subseteq \Phi$, und da Φ_0 unerfüllbar ist, folgt $\Phi_0 \models \psi$.

Die Umkehrung ist trivial. □

Anwendung in der Resolution

Vollständigkeit des Resolutionskalküls

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

Lemma. (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

Theorem. 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.

5.2 Der DPLL Algorithmus

Der DPLL Algorithmus

Wir werden als nächstes einen 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**.

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

DPLL Algorithmen

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

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

Basis Algorithmus DPLL(Φ, β)

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

Partielle Belegung β mit $\text{def}(\beta) \subseteq \text{var}(\Phi)$.

Ausgabe. $\beta' \supseteq \beta$ mit $\beta' \models \Phi$ oder unerfüllbar, wenn kein β' existiert.

Algorithmus. Wenn $\llbracket \Phi \rrbracket^\beta = 1$, gib β zurück.

Wenn $\llbracket \Phi \rrbracket^\beta = 0$, gib unerfüllbar zurück.

Sonst (d.h. wenn der Wert von Φ unter β unbestimmt ist)

reduce(β, Φ);

branch(β); Wähle unbelegte Variable X aus $\text{var}(\Phi)$ und Wahrheitswert t .

$\beta' := \text{DPLL}(\Phi, \beta[X \mapsto t])$.

wenn $\beta' \neq \text{unerfüllbar}$, gibt β' zurück

sonst gib $\text{DPLL}(\Phi, \beta[X \mapsto 1 - t])$ zurück.

Einheitsresolution

Die Funktion $\text{reduce}(\beta, \Phi)$ dient der Reduktion der Formel nach jedem Schritt.

Meistens wird nur Einheitsresolution verwendet.

Unit Clause Propagation.

Enthält Φ Klausel $C = \{L_1, \dots, L_k\}$ in der alle bis auf 1 Literal L_i durch β belegt werden, so muss jede erfüllende Belegung $\beta' \supseteq \beta$ das Literal L_i mit 1 belegen.

Ist $L_i = X$, können wir also direkt $[X \mapsto 1]$ zu β hinzufügen.

Ist $L_i = \neg X$, fügen wir $[X \mapsto 0]$ zu β hinzu.

Dies wird solange wiederholt, bis es keine Einer-Klauseln mehr gibt.

Basis DPLL(Φ, β)

Ausgabe.

$\beta' \supseteq \beta$ mit $\beta' \models \Phi$ oder unerf.

Algorithmus.

$\llbracket \Phi \rrbracket^\beta = 1 \rightsquigarrow \text{return } \beta$.

$\llbracket \Phi \rrbracket^\beta = 0 \rightsquigarrow \text{return unerf.}$

Sonst

$\text{reduce}(\beta, \Phi);$

$\text{branch}(\beta):$

Wähle $X \in \text{var}(\Phi)$, $t \in \{0, 1\}$.

$\beta' := \text{DPLL}(\Phi, \beta[X \mapsto t]).$

wenn $\beta' \models \Phi$, return β'

sonst

return $\text{DPLL}(\Phi, \beta[X \mapsto 1 - t])$

Branching

Reduce. Die Funktion $\text{reduce}(\beta, \Phi)$; dient der Reduktion der Formel nach jedem Schritt.

Branch. Die Funktion $\text{branch}(\beta, \Phi)$; wählt die nächste Variable aus.

Hier gibt es verschiedenste Heuristiken.

dynamic largest individual sum (DLIS). Wähle Literal L das am häufigsten vorkommt und setze es auf 1.

dynamic largest combined sum (DLCS). Wähle Variable, die am häufigsten vorkommt.

Maximal occurrence in minimal clauses (MOM). Wähle Variable die am häufigsten in kürzesten Klauseln vorkommt.

Basis DPLL(Φ, β)

Ausgabe.

$\beta' \supseteq \beta$ mit $\beta' \models \Phi$ oder unerf.

Algorithmus.

$\llbracket \Phi \rrbracket^\beta = 1 \rightsquigarrow \text{return } \beta$.

$\llbracket \Phi \rrbracket^\beta = 0 \rightsquigarrow \text{return unerf}$.

Sonst

$\text{reduce}(\beta, \Phi)$;

$\text{branch}(\beta)$:

Wähle $X \in \text{var}(\Phi)$, $t \in \{0, 1\}$.

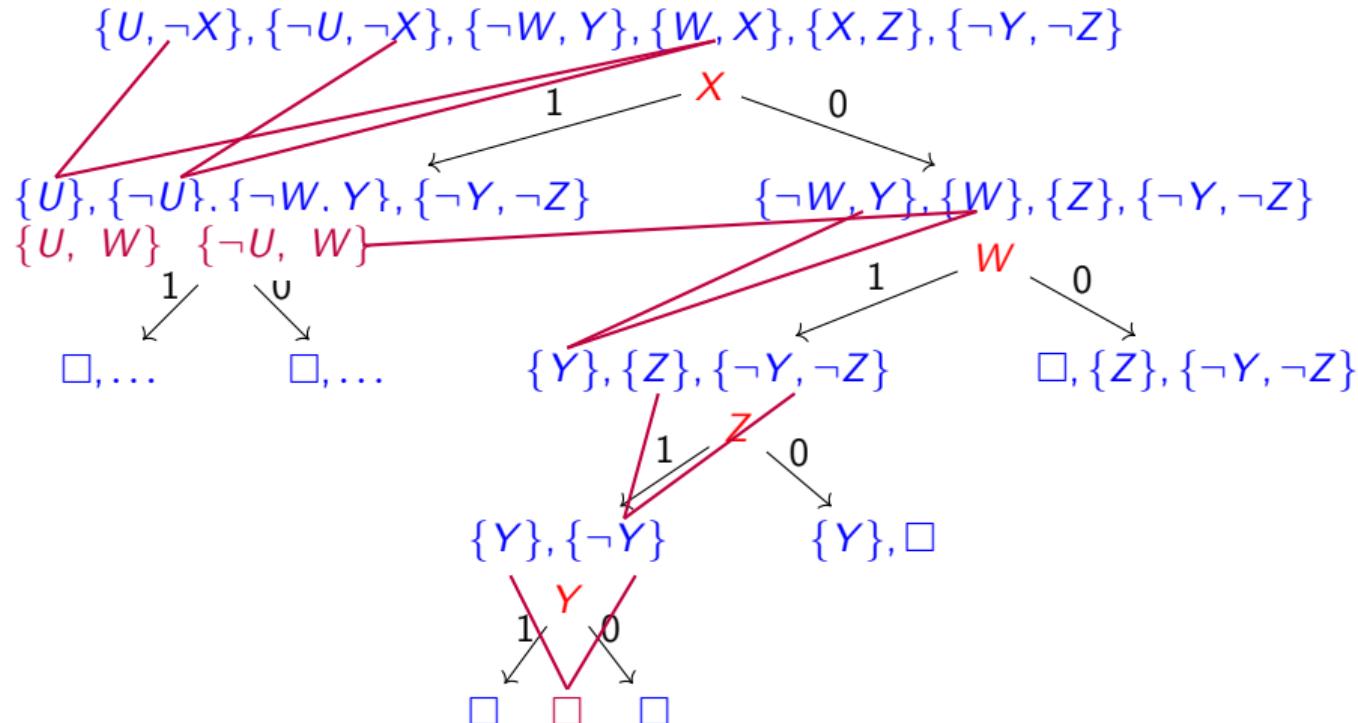
$\beta' := \text{DPLL}(\Phi, \beta[X \mapsto t])$.

wenn $\beta' \models \Phi$, return β'

sonst

return $\text{DPLL}(\Phi, \beta[X \mapsto 1 - t])$

Beispiel für den DPLL-Algorithmus



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.

DPLL vs. Resolution

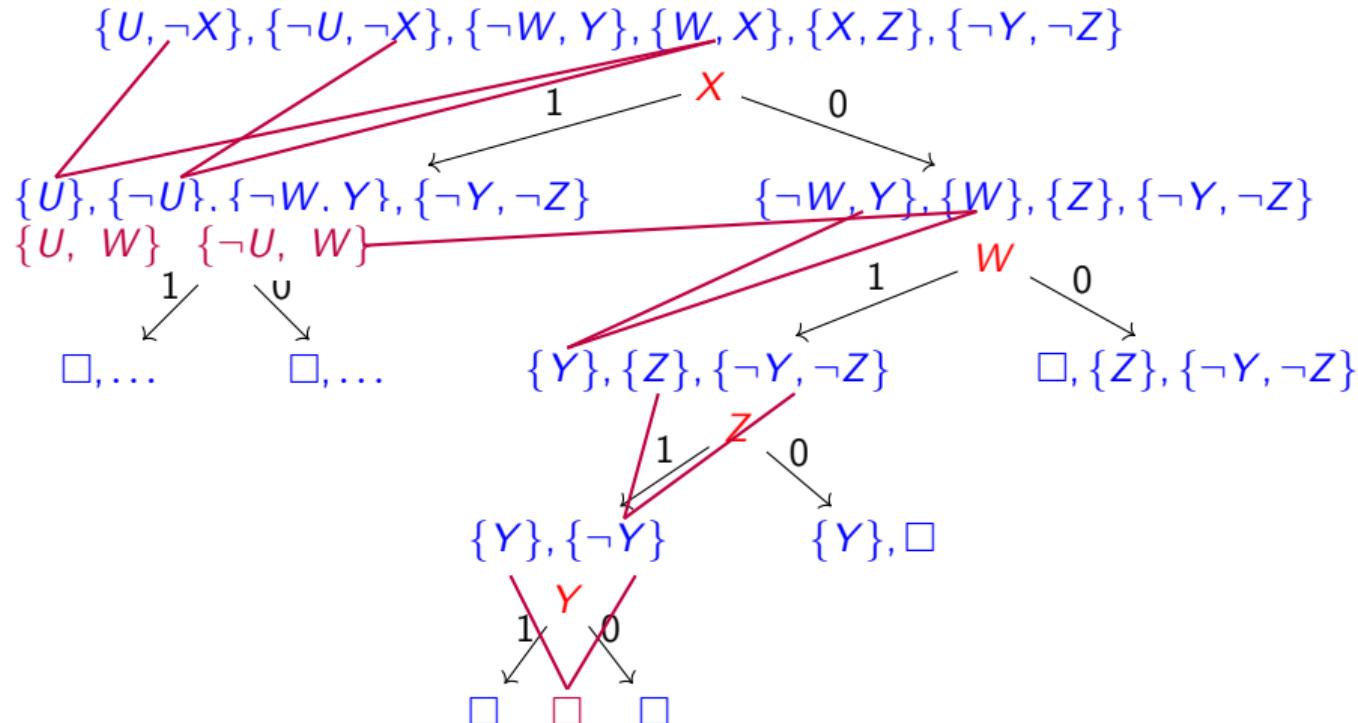
Es besteht ein enger Zusammenhang zwischen Resolutionswiderlegungen und Widerlegungen einer Formel durch den DPLL-Algorithmus.

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.

Beispiel für den DPLL-Algorithmus





Nov.	17 18 19 20 21 22 23	Einführung
	24 25 26 27 28 29 30	Aussagenlogik
	31 1 2 3 4 5 6	Normalformen
	7 8 9 10 11 12 13	Resolution
Dez.	14 15 16 17 18 19 20	Kompaktheit
	21 22 23 24 25 26 27	DPLL, Satz von Cook
	28 29 30 1 2 3 4	Strukturen und FO
	5 6 7 8 9 10 11	Prädikatenlogik
	12 13 14 15 16 17 18	Komplexität von FO
Jan.	19 20 21 22 23 24 25	Weihnachten
	26 27 28 29 30 31 1	Neujahr
	2 3 4 5 6 7 8	Normalformen
	9 10 11 12 13 14 15	Definierbarkeit
	16 17 18 19 20 21 22	EF-Spiele
Feb.	23 24 25 26 27 28 29	EF-Spiele
	30 31 1 2 3 4 5	Sequenzkalkül AL
	6 7 8 9 10 11 12	Sequenzkalkül FO
	13 14 15 16 17 18 19	Ausblick

Wiederholung

Der aussagenlogische Resolutionskalkül

Theorem. Eine Menge \mathcal{C} von Klauseln hat genau dann eine Resolutionswiderlegung, wenn \mathcal{C} unerfüllbar ist. D.h. es gilt

$$\mathcal{C} \vdash_R \square \quad \text{gdw.} \quad \mathcal{C} \text{ ist unerfüllbar} \quad \text{gdw.} \quad \mathcal{C} \models \perp$$

Lemma. (Korrektheit des Resolutionskalküls)

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

Lemma. (Vollständigkeit des Resolutionskalküls)

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

Der Kompaktheitssatz der Aussagenlogik

Theorem. (Kompaktheits- oder Endlichkeitssatz)

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

1. Φ ist erfüllbar gdw. jede endliche Teilmenge $\Phi' \subseteq \Phi$ erfüllbar ist.
2. $\Phi \models \psi$ gdw. eine endliche Teilmenge $\Phi_0 \subseteq \Phi$ existiert mit $\Phi_0 \models \psi$.

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

Erinnerung.

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 $\beta \models \Phi$.

DPLL Algorithmen

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

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

Basis Algorithmus DPLL(Φ, β)

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

Partielle Belegung β mit $\text{def}(\beta) \subseteq \text{var}(\Phi)$.

Ausgabe. $\beta' \supseteq \beta$ mit $\beta' \models \Phi$ oder unerfüllbar, wenn kein β' existiert.

Algorithmus. Wenn $\llbracket \Phi \rrbracket^\beta = 1$, gib β zurück.

Wenn $\llbracket \Phi \rrbracket^\beta = 0$, gib unerfüllbar zurück.

Sonst (d.h. wenn der Wert von Φ unter β unbestimmt ist)

reduce(β, Φ);

branch(β); Wähle unbelegte Variable X aus $\text{var}(\Phi)$ und Wahrheitswert t .

$\beta' := \text{DPLL}(\Phi, \beta[X \mapsto t])$.

wenn $\beta' \neq \text{unerfüllbar}$, gibt β' zurück

sonst gib $\text{DPLL}(\Phi, \beta[X \mapsto 1 - t])$ zurück.

Einheitsresolution

Die Funktion $\text{reduce}(\beta, \Phi)$ dient der Reduktion der Formel nach jedem Schritt.

Meistens wird nur Einheitsresolution verwendet.

Unit Clause Propagation.

Enthält Φ Klausel $C = \{L_1, \dots, L_k\}$ in der alle bis auf 1 Literal L_i durch β belegt werden, so muss jede erfüllende Belegung $\beta' \supseteq \beta$ das Literal L_i mit 1 belegen.

Ist $L_i = X$, können wir also direkt $[X \mapsto 1]$ zu β hinzufügen.

Ist $L_i = \neg X$, fügen wir $[X \mapsto 0]$ zu β hinzu.

Dies wird solange wiederholt, bis es keine Einer-Klauseln mehr gibt.

Basis DPLL(Φ, β)

Ausgabe.

$\beta' \supseteq \beta$ mit $\beta' \models \Phi$ oder unerf.

Algorithmus.

$[\Phi]^\beta = 1 \rightsquigarrow \text{return } \beta$.

$[\Phi]^\beta = 0 \rightsquigarrow \text{return unerf.}$

Sonst

$\text{reduce}(\beta, \Phi);$

$\text{branch}(\beta):$

Wähle $X \in \text{var}(\Phi)$, $t \in \{0, 1\}$.

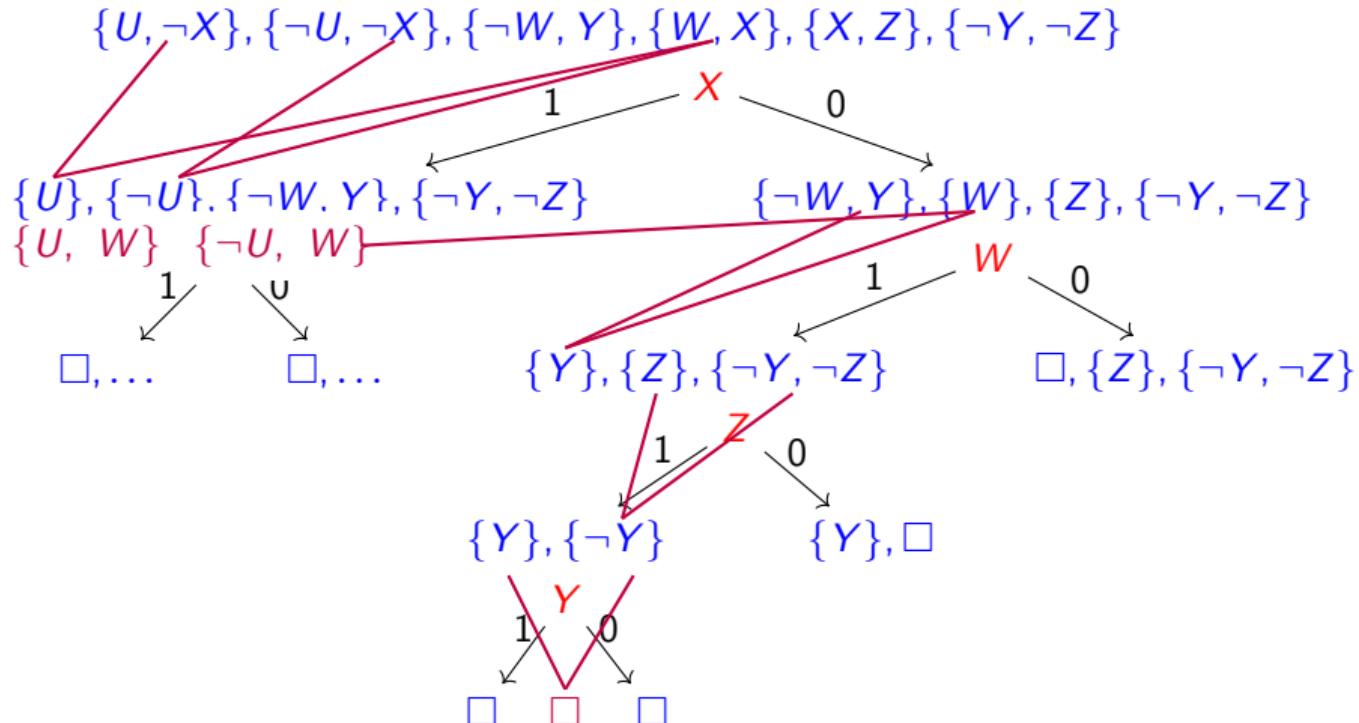
$\beta' := \text{DPLL}(\Phi, \beta[X \mapsto t]).$

wenn $\beta' \models \Phi$, return β'

sonst

return $\text{DPLL}(\Phi, \beta[X \mapsto 1 - t])$

Beispiel für den DPLL-Algorithmus



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.

Der Satz von Cook: NP-Vollständigkeit von SAT

NP-Vollständigkeit von SAT

Definition.

Das aussagenlogische Erfüllbarkeitsproblem (SAT) ist das Problem

SAT

Eingabe. Eine aussagenlogische Formel $\varphi \in AL$

Problem. Entscheide, ob φ erfüllbar ist.

Satz.

(Cook 1970, Levin 1973)

SAT ist NP-vollständig.

6.1 NP-Vollständigkeit

Schwere und leichte Probleme

3-Färbbarkeit

Definition (3-Färbbarkeit).

Ein Graph G ist **3-färbbar**, wenn es eine Funktion

$$c : V(G) \rightarrow \{C_1, C_2, C_3\}$$

gibt, so dass

$$c(u) \neq c(v) \text{ für alle Kanten } \{u, v\} \in E(G).$$

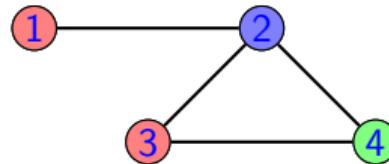
Wir nennen $c : V(G) \rightarrow \{1, 2, 3\}$ eine **3-Färbung** von G .

Das Problem **3-COL**. Das zugehörige Entscheidungsproblem:

3-COL

Eingabe. Graph G .

Problem. Entscheide, ob G 3-färbbar ist.



Leicht oder schwer?

Entscheidungsproblem 3-COL.

3-COL.

Eingabe. Ein Graph G .

Ausgabe. 1, wenn G 3-färbbar ist, sonst 0.

3-Färbung.

$$c : V(G) \rightarrow \{C_1, C_2, C_3\}$$

mit

$$c(u) \neq c(v)$$

für alle $\{u, v\} \in E(G)$.

Frage. Wie schwer (algorithmisch gesehen) ist es, das Problem zu lösen?

Kann man bei einer Eingabe G mit n Knoten z.B. in

- Linearzeit $O(n)$ lösen?
- polynomieller Zeit $O(n^d)$ lösen, für eine Konstante d ?
- in exponentieller Zeit $2^{O(n)}$ lösen?
- oder in Zeit $O(2^{2^{2^{2^{2^n}}}})$? Oder gar $O\left(2^{\cdot^{\cdot^{\cdot^2}}} n\right)$?

Schummeln ist nicht.

Überprüfen einer vorgeschlagenen Lösung für $3\text{-COL}(G)$.

Sei $c : V(G) \rightarrow \{C_1, C_2, C_3\}$ eine Funktion.

Wir können sehr leicht überprüfen, ob c eine 3-Färbung von G ist:

Überprüfe für jede Kante $\{u, v\} \in E(G)$, ob $c(u) \neq c(v)$.

Zahl der möglichen Lösungen.

Es gibt $3^{|V(G)|}$ mögliche Funktionen $c : V(G) \rightarrow \{C_1, C_2, C_3\}$.

Wir können also alle Funktionen nacheinander überprüfen.

Dies liefert einen Algorithmus, der 3-COL in Zeit $3^{O(n)} \cdot n^c$ löst.

Offenes Problem. Aber geht es auch in polynomieller Zeit?

Komplexitätsklassen

Komplexitätsklassen.

Eine Komplexitätsklasse \mathcal{C} ist eine Klasse algorithmischer Probleme.

Formal: Klasse von Sprachen über einem Alphabet.

Intuitiv: Klasse von Problemen, die mit Hilfe eines bestimmten Ressourceneinsatzes gelöst werden können.

Beispiel.

Die Klasse EXP aller Probleme P die bei Eingaben der Größe n in Zeit $2^{O(n^d)}$, für ein $d \geq 1$, gelöst werden können.

Wir haben gesehen, dass $3\text{-COL} \in \text{EXP}$.

Komplexitätsklassen

Komplexitätsklassen.

Eine Komplexitätsklasse \mathcal{C} ist eine Klasse algorithmischer Probleme.

Formal: Klasse von Sprachen über einem Alphabet.

Die Klasse "PTIME". Klasse aller Probleme, die durch einen Algorithmus gelöst werden können, dessen Laufzeit auf Eingaben der Länge n durch $p(n)$ beschränkt ist.

Hierbei ist $p(x)$ ein festes Polynom (unabhängig von n).

Formal ist PTIME die Klasse aller Sprachen, die von einer **deterministischen** Turing-Maschine mit polynomieller Zeitschranke $p(n)$ gelöst werden können.

Beispiele.

- Berechnen maximaler Flüsse in Netzwerken.
- 2-Färbbarkeit von Graphen.
- Das Auswertungsproblem der Aussagenlogik.

Bemerkung. Probleme in PTIME werden meistens durch explizite Konstruktion oder Berechnung der Lösung gelöst.

Komplexitätsklassen

Die Klasse "NP". Klasse aller Probleme,

- die von einer *nicht-deterministischen* Turingmaschine mit polynomieller Zeitschranke $p(n)$ gelöst werden können.
- bei denen
 1. die *Größe einer Lösung* für eine Eingabe der Größe n *polynomiell* beschränkt ist und
 2. für die eine vorgeschlagene Lösung in *Polynomialzeit* überprüft werden kann.

D.h. wir können für eine Eingabe x eine Lösung raten und diese dann in polynomieller Zeit (in $|x|$) verifizieren.

Bemerkung.

Probleme in *NP* werden meistens durch Suchverfahren gelöst.

D.h. man berechnet die Lösung nicht, sondern sucht geschickt nach einer Lösung und überprüft, dass diese auch stimmt.

Beispiele.

- Das SAT-Problem ist in NP.

Eingabe: $\varphi(X_1, \dots, X_n) \in \text{AL}$.

Gesucht: Belegung β von X_1, \dots, X_n mit $[\![\varphi]\!]^\beta = 1$

Größe von β polynomiell in Größe von φ .

Wir können in polynomieller Zeit überprüfen, ob $[\![\varphi]\!]^\beta = 1$

Komplexitätsklassen

Die Klasse "NP". Klasse aller Probleme,

- die von einer *nicht-deterministischen* Turingmaschine mit polynomieller Zeitschranke $p(n)$ gelöst werden können.
- bei denen
 1. die *Größe einer Lösung* für eine Eingabe der Größe n *polynomiel* beschränkt ist und
 2. für die eine vorgeschlagene Lösung in *Polynomialzeit* überprüft werden kann.

D.h. wir können für eine Eingabe x eine Lösung raten und diese dann in polynomieller Zeit (in $|x|$) verifizieren.

Bemerkung.

Probleme in *NP* werden meistens durch Suchverfahren gelöst.

D.h. man berechnet die Lösung nicht, sondern sucht geschickt nach einer Lösung und überprüft, dass diese auch stimmt.

Beispiele.

- Das SAT-Problem ist in NP.
- 3-COL.
- Hat G ein dominating set der Größe k ?
- Hat G einen Hamiltonkreis?
- Set Cover
- Rucksackprobleme
- Constraint Satisfaction Problems
- Integer Programming

NP-Vollständigkeit

Die Komplexitätsklassen P und NP. Die Probleme in P können (im Prinzip) effizient gelöst werden, d.h. in Polynomialzeit. Bei Problemen in NP wissen (und glauben) wir das nicht.

Wir können aber Probleme in NP lösen, indem wir eine kurze, d.h. polynomiell große, Lösung raten und dann effizient überprüfen.

P \subseteq NP. Man sieht sofort, dass jedes Problem in P auch in NP liegt.

Es sind also nicht alle Probleme in NP „schwer“.

Woran erkennt man die „schweren“ Probleme in NP? **Gar nicht**

NP-Vollständigkeit

Definition.

Ein Problem L in NP ist *NP-vollständig*, wenn man mit einem Polynomialzeitalgorithmus für L als Unterprogramm jedes andere Problem in NP auch in Polynomialzeit lösen könnte.

Wenn also ein einziges *NP-vollständiges* Problem in Polynomialzeit gelöst werden kann, dann auch alle anderen NP -Probleme, d.h. es gilt $P = NP$.

Beispiele NP-vollständiger Probleme

Wie zeigt man NP-Vollständigkeit?

Um zu beweisen, dass ein Problem L NP-vollständig ist, müssen wir folgendes zeigen:

1. $L \in \text{NP}$ und
2. falls L in Polynomialzeit gelöst werden kann, dann können alle NP-Probleme in Polynomialzeit gelöst werden.

Reduktionen. Wenn man schon ein NP-vollständiges Problem M kennt, dann reicht es zu zeigen, dass ein Polynomialzeitalgorithmus für L benutzt werden kann, um M in Polynomialzeit zu lösen.

Nur wo bekommt man das „erste“ NP-vollständige Problem her?

Die Logik zur Rettung. Dies gelang Ende der 60er Jahre *Stephen Cook* für das SAT Problem, der für seine bahnbrechenden Arbeiten zur NP-Vollständigkeit den Turing-Award erhielt.

Stephen Cook



(c) Jiří Janíček, CC-BY-SA-3.0

Richard Karp



(c) Rama,
Cc-by-sa-2.0-fr

Alan Turing



6.2 Der Satz von Cook

NP-Vollständigkeit von SAT

Definition.

Das aussagenlogische Erfüllbarkeitsproblem (SAT) ist das Problem

SAT

Eingabe. Eine aussagenlogische Formel $\varphi \in AL$

Problem. Entscheide, ob φ erfüllbar ist.

Satz.

(Cook 1970, Levin 1973)

SAT ist NP-vollständig.

NP-Vollständigkeit. Um zu beweisen, dass SAT NP-vollständig ist, müssen wir folgendes zeigen:

1. $SAT \in NP$ und
2. falls SAT in Polynomialzeit gelöst werden kann, dann können alle NP-Probleme in Polynomialzeit gelöst werden.

Beweis des Satzes von Cook

Satz. SAT ist NP-vollständig. (Cook 1970, Levin 1973)

NP-Vollständigkeit. Um zu beweisen, dass SAT NP-vollständig ist, müssen wir folgendes zeigen:

1. $\text{SAT} \in \text{NP}$
2. falls SAT in Polynomialzeit gelöst werden kann, dann können alle NP-Probleme in Polynomialzeit gelöst werden.

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

Eingabe: $\varphi(X_1, \dots, X_n) \in \text{AL}$.

Gesucht: Belegung β von X_1, \dots, X_n mit $\llbracket \varphi \rrbracket^\beta = 1$

Größe von β polynomiell in Größe von φ .

Wir können in polynomieller Zeit überprüfen, ob $\llbracket \varphi \rrbracket^\beta = 1$

Beweis des Satzes von Cook

NP-Härte von SAT. Wir müssen zeigen, dass wenn SAT in Polynomialzeit lösbar wäre, dann wären alle Probleme $L \in \text{NP}$ in PTIME.

1. Sei $L \in \text{NP}$. Dann gibt es ein Polynom $p(n)$ und eine *nicht-deterministische* TM M , die L in Zeit $O(p(n))$ löst.
2. Für jede feste NTM M und jedes Polynom $p(n)$, zeigt man nun:
Zu jedem Eingabewort w der Länge n existiert eine Formel φ_w mit:
 M akzeptiert w in Zeit $p(n)$ gdw. φ_w erfüllbar ist.
 φ_w kann in Zeit $O(n^d)$ konstruiert werden, für ein festes $d = d(M, p)$.
3. Wenn man also SAT in pol. Zeit lösen könnte, dann könnte man auch in pol. Zeit entscheiden, ob M eine Eingabe w der Länge n in Zeit $p(n)$ akzeptiert. Also könnte man L in pol. Zeit entscheiden.

6.3 Das Wortproblem endlicher Automaten als SAT-Problem

Das Wortproblem regulärer Sprachen

Beispiel. Wir betrachten das Wortproblem regulärer Sprachen.

Sei Σ ein Alphabet.

Sei $\mathcal{A} := (Q, \Sigma, q_0, \Delta, F)$ ein endlicher, nicht-deterministischer Automat.

Wortproblem für \mathcal{A} .

Eingabe. $w \in \Sigma^*$

Ausgabe. 1 wenn \mathcal{A} das Wort w akzeptiert, sonst 0.

Reduktion auf SAT. Wir wollen das Problem auf SAT reduzieren.

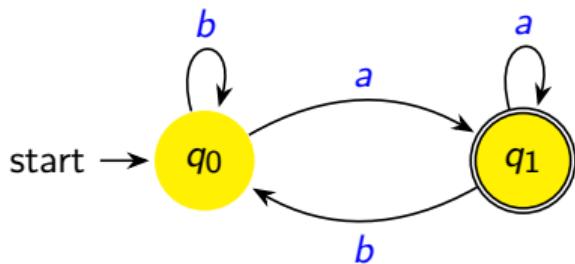
Zu jedem $w \in \Sigma^*$ müssen wir eine Formel $\varphi_w \in \text{AL}$ konstruieren, so dass

\mathcal{A} akzeptiert w gdw. φ_w ist erfüllbar.

Beispiel für das Wortproblem

Beispiel

Sei $Q := \{q_0, q_1\}$ und $F := \{q_1\}$. Sei $\Sigma := \{a, b\}$.



Wort w : $a \quad b \quad a \quad b \quad a$

Lauf von \mathcal{A} : $q_0 \quad q_1 \quad q_0 \quad q_1 \quad q_0 \quad q_1$

Das Wortproblem regulärer Sprachen

Beispiel. Wir betrachten das Wortproblem regulärer Sprachen.

Sei Σ ein Alphabet.

Sei $\mathcal{A} := (Q, \Sigma, q_0, \Delta, F)$ ein endlicher, nicht-deterministischer Automat.

Wortproblem für \mathcal{A} .

Eingabe. $w \in \Sigma^*$

Ausgabe. 1 wenn \mathcal{A} das Wort w akzeptiert, sonst 0.

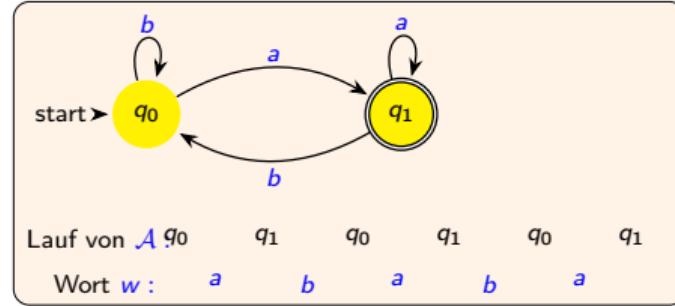
Reduktion auf SAT. Wir wollen das Problem auf SAT reduzieren.

Zu jedem $w \in \Sigma^*$ müssen wir eine Formel $\varphi_w \in \text{AL}$ konstruieren, so dass

\mathcal{A} akzeptiert w gdw. φ_w ist erfüllbar.

Formalisierung in der Aussagenlogik

Reduktion auf SAT. Zu jedem $w \in \Sigma^*$ müssen wir eine Formel $\varphi_w \in \text{AL}$ konstruieren, so dass \mathcal{A} akzeptiert w gdw. φ_w ist erfüllbar.

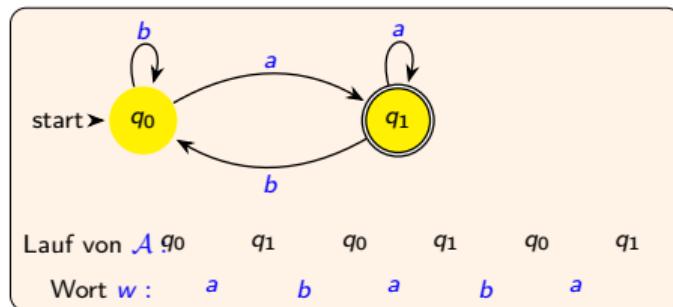


Formalisierung in der Aussagenlogik

Reduktion auf SAT. Zu jedem $w \in \Sigma^*$ müssen wir eine Formel $\varphi_w \in \text{AL}$ konstruieren, so dass \mathcal{A} akzeptiert w gdw. φ_w ist erfüllbar.

Idee. Die Formel φ_w soll so aus w und \mathcal{A} konstruiert werden, dass

- eine Belegung β von $\text{var}(\varphi_w)$ einem möglichen Lauf $\rho(\beta)$ von \mathcal{A} auf w entspricht und
- wenn $\beta \models \varphi_w$, dann ist $\rho(\beta)$ ein akzeptierender Lauf von \mathcal{A} auf w .

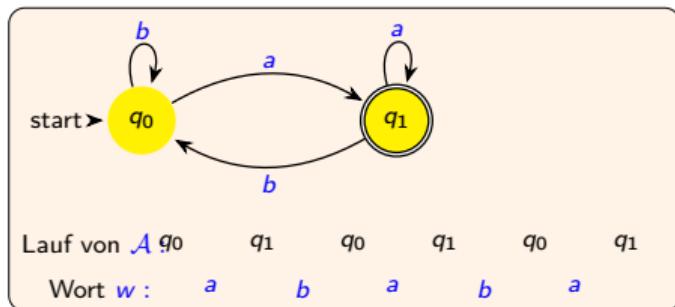


Umsetzung in der Aussagenlogik

- Wir müssen das Wort $w = a_1, \dots, a_n$ durch Variablen kodieren.
- Wir brauchen Variablen, die bestimmen, in welchem Zustand der Automat nach dem i -ten Buchstaben ist.
- Die Formel muss sicherstellen, dass die so kodierte Zustandsfolge auch mit dem Automat \mathcal{A} übereinstimmt.

D.h. wenn eine Belegung der Variablen behauptet, dass \mathcal{A} nach dem i -ten Buchstaben im Zustand q ist und nach dem $i+1$ -ten im Zustand q' , dann muss $(q, a_{i+1}, q') \in \Delta$ sein.

- Und dann muss der letzte Zustand noch in F liegen.



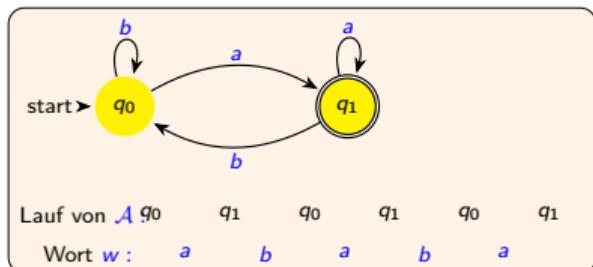
Lauf eines Automaten

Variablen. Sei $w = a_1, \dots, a_n$ ein Wort der Länge n und $\Sigma := \{a, b\}$.

- $\{S_i^c : c \in \Sigma, 1 \leq i \leq n\}$ „ $S_i^c = 1$ gdw. $a_i = c$ “
- $\{Q_i^q : q \in Q, 0 \leq i \leq n\}$ „ $Q_i^q = 1$, wenn \mathcal{A} nach i Schritten in Zustand q ist“

Variablen.

Pos	0	1	2	3	4	5
S^a		1	0	1	0	1
S^b		0	1	0	1	0
Q^{q_0}	1	0	1	0	1	0
Q^{q_1}	0	1	0	1	0	1



Lauf eines Automaten

Variablen. Sei $w = a_1, \dots, a_n$ ein Wort der Länge n und $\Sigma := \{a, b\}$.

- $\{S_i^c : c \in \Sigma, 1 \leq i \leq n\}$ „ $S_i^c = 1$ gdw. $a_i = c$ “
- $\{Q_i^q : q \in Q, 0 \leq i \leq n\}$ „ $Q_i^q = 1$, wenn \mathcal{A} nach i Schritten in Zustand q ist“

Formel φ_w .

- „Kodieren des Eingabeworts durch Variablen“
 $\varphi_{w,Eingabe} := \bigwedge_{i=1}^n (S_i^{a_i} \wedge \bigwedge_{c \in \Sigma \setminus \{a_i\}} \neg S_i^c)$
- „Nach jedem Buchstaben ist der Automat in genau einem Zustand“

$$\varphi_{w,Zustand} := \bigwedge_{i=0}^n \left(\bigvee_{q \in Q} (Q_i^q \wedge \bigwedge_{q' \in Q \setminus \{q\}} \neg Q_i^{q'}) \right)$$

- „Zustandsübergänge kodieren gültigen Lauf“
 $\varphi_{w,Lauf} := Q_0^{q_0} \wedge \bigwedge_{i=1}^n \left(\bigvee_{(q,a,q') \in \Delta} (Q_{i-1}^q \wedge S_i^a \wedge Q_i^{q'}) \right)$

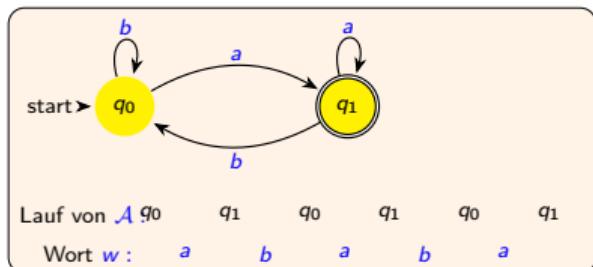
- „Der letzte Zustand ist in F “

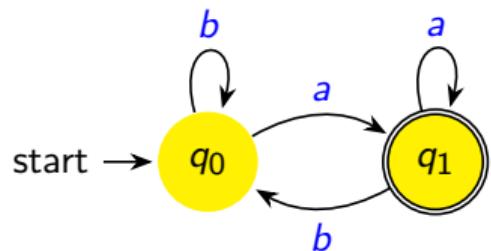
$$\varphi_{w,end} := \bigvee_{q \in F} Q_n^q$$

$$\varphi_w := \varphi_{w,Eingabe} \wedge \varphi_{w,Zustand} \wedge \varphi_{w,Lauf} \wedge \varphi_{w,end}$$

Variablen.

Pos	0	1	2	3	4	5
S^a		1	0	1	0	1
S^b		0	1	0	1	0
Q^{q_0}	1	0	1	0	1	0
Q^{q_1}	0	1	0	1	0	1





	Position:	1	2	3	4	5
	Wort w :	a	b	a	b	a
	Lauf von \mathcal{A} :	q_0	q_1	q_0	q_1	q_0

Variablen.

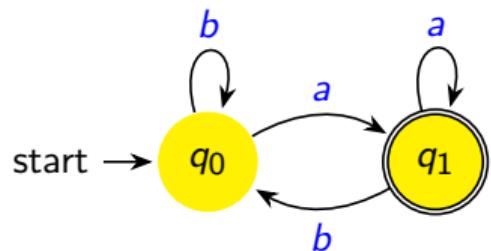
Pos	0	1	2	3	4	5
S^a	1	0	1	0	1	
S^b	0	1	0	1	0	
Q^{q_0}	1	0	1	0	1	0
Q^{q_1}	0	1	0	1	0	1

Formel φ_w .

$$\begin{aligned}
 \varphi_{w,Eingabe} &:= \bigwedge_{i=0}^n (S_i^{a_i} \wedge \bigwedge_{c \in \Sigma \setminus \{a_i\}} \neg S_i^c) \\
 \varphi_{w,Zustand} &:= \bigwedge_{i=0}^n \left(\bigvee_{q \in Q} (Q_i^q \wedge \bigwedge_{q' \in Q \setminus \{q\}} \neg Q_i^{q'}) \right) \\
 \varphi_{w,Lauf} &:= Q_0^{q_0} \wedge \bigwedge_{i=1}^n \left(\bigvee_{(q,a,q') \in \Delta} (Q_{i-1}^q \wedge S_i^a \wedge Q_i^{q'}) \right) \\
 \varphi_{w,end} &:= \bigvee_{q \in F} Q_n^q \\
 \varphi_w &:= \varphi_{w,Eingabe} \wedge \varphi_{w,Zustand} \wedge \varphi_{w,Lauf} \wedge \varphi_{w,end}
 \end{aligned}$$

Variablen.

$$S_1^a, S_1^b, \dots, S_5^a, S_5^b, \quad Q_0^{q_0}, Q_0^{q_1}, \dots, Q_5^{q_0}, Q_5^{q_1}$$



	Position:	1	2	3	4	5
	Wort w :	a	b	a	b	a
	Lauf von \mathcal{A} :	q_0	q_1	q_0	q_1	q_0

Variablen.

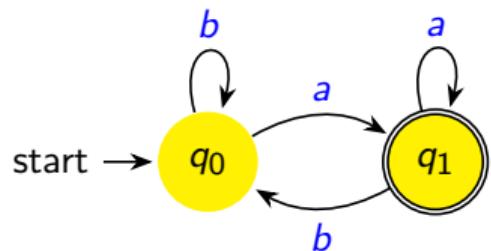
Pos	0	1	2	3	4	5
S^a	1	0	1	0	1	
S^b	0	1	0	1	0	
Q^{q_0}	1	0	1	0	1	0
Q^{q_1}	0	1	0	1	0	1

Formel φ_w .

$$\begin{aligned}
 \varphi_{w,Eingabe} &:= \bigwedge_{i=0}^n (S_i^{a_i} \wedge \bigwedge_{c \in \Sigma \setminus \{a_i\}} \neg S_i^c) \\
 \varphi_{w,Zustand} &:= \bigwedge_{i=0}^n \left(\bigvee_{q \in Q} (Q_i^q \wedge \bigwedge_{q' \in Q \setminus \{q\}} \neg Q_i^{q'}) \right) \\
 \varphi_{w,Lauf} &:= Q_0^{q_0} \wedge \bigwedge_{i=1}^n \left(\bigvee_{(q,a,q') \in \Delta} (Q_{i-1}^q \wedge S_i^a \wedge Q_i^{q'}) \right) \\
 \varphi_{w,end} &:= \bigvee_{q \in F} Q_n^q \\
 \varphi_w &:= \varphi_{w,Eingabe} \wedge \varphi_{w,Zustand} \wedge \varphi_{w,Lauf} \wedge \varphi_{w,end}
 \end{aligned}$$

„Kodieren des Eingabeworts durch Variablen“

$$\varphi_{ababa,Eing.} := (S_1^a \wedge \neg S_1^b) \quad \wedge \quad (S_2^b \wedge \neg S_2^a) \quad \wedge \quad (S_3^a \wedge \neg S_3^b) \quad \wedge \quad \dots$$



	Position:	1	2	3	4	5
	Wort w :	a	b	a	b	a
	Lauf von \mathcal{A} :	q_0	q_1	q_0	q_1	q_0

Variablen.

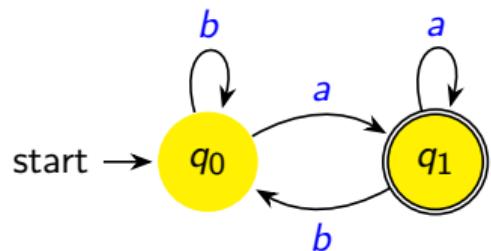
Pos	0	1	2	3	4	5
S^a	1	0	1	0	1	
S^b	0	1	0	1	0	
Q^{q_0}	1	0	1	0	1	0
Q^{q_1}	0	1	0	1	0	1

Formel φ_w .

$$\begin{aligned}
 \varphi_{w,Eingabe} &:= \bigwedge_{i=0}^n (S_i^{a_i} \wedge \bigwedge_{c \in \Sigma \setminus \{a_i\}} \neg S_i^c) \\
 \varphi_{w,Zustand} &:= \bigwedge_{i=0}^n \left(\bigvee_{q \in Q} (Q_i^q \wedge \bigwedge_{q' \in Q \setminus \{q\}} \neg Q_i^{q'}) \right) \\
 \varphi_{w,Lauf} &:= Q_0^{q_0} \wedge \bigwedge_{i=1}^n \left(\bigvee_{(q,a,q') \in \Delta} (Q_{i-1}^q \wedge S_i^a \wedge Q_i^{q'}) \right) \\
 \varphi_{w,end} &:= \bigvee_{q \in F} Q_n^q \\
 \varphi_w &:= \varphi_{w,Eingabe} \wedge \varphi_{w,Zustand} \wedge \varphi_{w,Lauf} \wedge \varphi_{w,end}
 \end{aligned}$$

„Nach jedem Buchstaben ist der Automat in genau einem Zustand“

$$((Q_0^{q_0} \wedge \neg Q_0^{q_1}) \vee (\neg Q_0^{q_0} \wedge Q_0^{q_1})) \wedge ((Q_1^{q_0} \wedge \neg Q_1^{q_1}) \vee (\neg Q_1^{q_0} \wedge Q_1^{q_1})) \dots$$



	Position:	1	2	3	4	5
	Wort w :	a	b	a	b	a
	Lauf von \mathcal{A} :	q_0	q_1	q_0	q_1	q_0

Variablen.

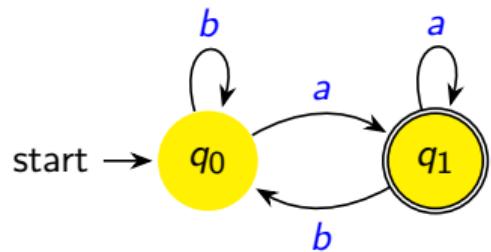
Pos	0	1	2	3	4	5
S^a	1	0	1	0	1	
S^b	0	1	0	1	0	
Q^{q_0}	1	0	1	0	1	0
Q^{q_1}	0	1	0	1	0	1

Formel φ_w .

$$\begin{aligned}
 \varphi_{w,Eingabe} &:= \bigwedge_{i=0}^n (S_i^{a_i} \wedge \bigwedge_{c \in \Sigma \setminus \{a_i\}} \neg S_i^c) \\
 \varphi_{w,Zustand} &:= \bigwedge_{i=0}^n \left(\bigvee_{q \in Q} (Q_i^q \wedge \bigwedge_{q' \in Q \setminus \{q\}} \neg Q_i^{q'}) \right) \\
 \varphi_{w,Lauf} &:= Q_0^{q_0} \wedge \bigwedge_{i=1}^n \left(\bigvee_{(q,a,q') \in \Delta} (Q_{i-1}^q \wedge S_i^a \wedge Q_i^{q'}) \right) \\
 \varphi_{w,end} &:= \bigvee_{q \in F} Q_n^q \\
 \varphi_w &:= \varphi_{w,Eingabe} \wedge \varphi_{w,Zustand} \wedge \varphi_{w,Lauf} \wedge \varphi_{w,end}
 \end{aligned}$$

„Zustandsübergänge kodieren gültigen Lauf“

$$\begin{aligned}
 Q_0^{q_0} \wedge \left((Q_0^{q_0} \wedge S_1^a \wedge Q_1^{q_1}) \vee (Q_0^{q_0} \wedge S_1^b \wedge Q_1^{q_0}) \vee (Q_0^{q_1} \wedge S_1^a \wedge Q_1^{q_1}) \vee \right. \\
 \left. (Q_0^{q_1} \wedge S_1^b \wedge Q_1^{q_0}) \right) \wedge \dots
 \end{aligned}$$



Position: 1 2 3 4 5

Wort w : a b a b a

Lauf von \mathcal{A} : q0 q1 q0 q1 q0 q1

Variablen.

Pos	0	1	2	3	4	5
S^a	1	0	1	0	1	
S^b	0	1	0	1	0	
Q^{q_0}	1	0	1	0	1	0
Q^{q_1}	0	1	0	1	0	1

Formel φ_w .

$$\varphi_{w,Eingabe} := \bigwedge_{i=0}^n (S_i^{a_i} \wedge \bigwedge_{c \in \Sigma \setminus \{a_i\}} \neg S_i^c)$$

$$\varphi_{w,Zustand} := \bigwedge_{i=0}^n \left(\bigvee_{q \in Q} (Q_i^q \wedge \bigwedge_{q' \in Q \setminus \{q\}} \neg Q_i^{q'}) \right)$$

$$\varphi_{w,Lauf} := Q_0^{q_0} \wedge \bigwedge_{i=1}^n \left(\bigvee_{(q,a,q') \in \Delta} (Q_{i-1}^q \wedge S_i^a \wedge Q_i^{q'}) \right)$$

$$\varphi_{w,end} := \bigvee_{q \in F} Q_i^q$$

$$\varphi_w := \varphi_{w,Eingabe} \wedge \varphi_{w,Zustand} \wedge \varphi_{w,Lauf} \wedge \varphi_{w,end}$$

„Der letzte Zustand ist in F “

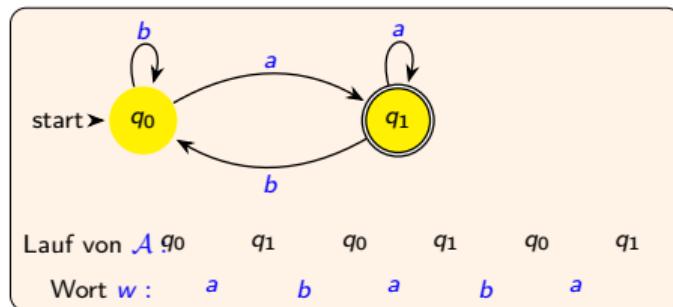
$$\varphi_{ababa,end} := Q_5^{q_1}$$

Formalisierung in der Aussagenlogik

Reduktion auf SAT. Zu jedem $w \in \Sigma^*$ müssen wir eine Formel $\varphi_w \in \text{AL}$ konstruieren, so dass \mathcal{A} akzeptiert w gdw. φ_w ist erfüllbar.

Idee. Die Formel φ_w soll so aus w und \mathcal{A} konstruiert werden, dass

- eine Belegung β von $\text{var}(\varphi_w)$ einem möglichen Lauf $\rho(\beta)$ von \mathcal{A} auf w entspricht und
- wenn $\beta \models \varphi_w$, dann ist $\rho(\beta)$ ein akzeptierender Lauf von \mathcal{A} auf w .



Reduktion des regulären Wortproblems auf SAT

Theorem. Für jedes endliche Alphabet Σ und jede reguläre Sprache $\mathcal{L} \subseteq \Sigma^*$ kann das Wortproblem für \mathcal{L} in Polynomialzeit auf SAT reduziert werden.

D.h., es existiert ein Algorithmus $\mathcal{A}_{\mathcal{L}}$ der auf Eingabe $w \in \Sigma^*$ der Länge n in Zeit $O(n^c)$, für eine Konstante c , eine Formel φ_w berechnet, so dass

$$w \in \mathcal{L} \text{ gdw. } \varphi_w \text{ erfüllbar ist.}$$

Folgerung. Wir können also das Wortproblem für \mathcal{L} mit Hilfe eines SAT-Lösers lösen, indem wir zunächst die Formel φ_w ausrechnen und diese dann an den SAT-Löser als Eingabe geben.

Der Satz von Cook

Beweis des Satzes von Cook

Satz. SAT ist NP-vollständig. (Cook 1970, Levin 1973)

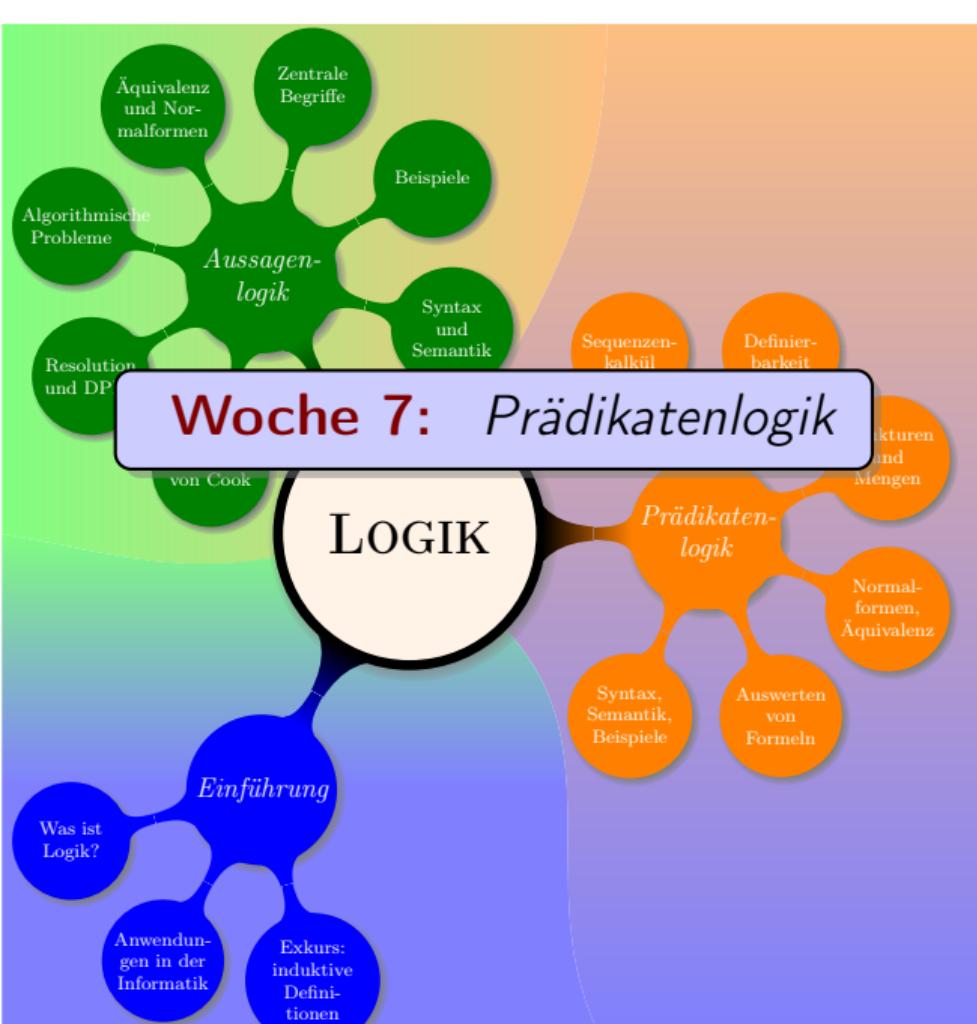
NP-Vollständigkeit. Um zu beweisen, dass SAT NP-vollständig ist, müssen wir folgendes zeigen:

1. $\text{SAT} \in \text{NP}$
2. falls SAT in Polynomialzeit gelöst werden kann, dann können alle NP-Probleme in Polynomialzeit gelöst werden.

Beweis des Satzes von Cook

NP-Härte von SAT. Wir müssen zeigen, dass wenn SAT in Polynomialzeit lösbar wäre, dann wären alle Probleme $L \in \text{NP}$ in PTIME.

1. Sei $L \in \text{NP}$. Dann gibt es ein Polynom $p(n)$ und eine *nicht-deterministische* TM M , die L in Zeit $O(p(n))$ löst.
2. Für jede feste NTM M und jedes Polynom $p(n)$, zeigt man nun:
Zu jedem Eingabewort w der Länge n existiert eine Formel φ_w mit:
 M akzeptiert w in Zeit $p(n)$ gdw. φ_w erfüllbar ist.
 φ_w kann in Zeit $O(n^d)$ konstruiert werden, für ein festes $d = d(M, p)$.
3. Wenn man also SAT in pol. Zeit lösen könnte, dann könnte man auch in pol. Zeit entscheiden, ob M eine Eingabe w der Länge n in Zeit $p(n)$ akzeptiert. Also könnte man L in pol. Zeit entscheiden.



Nov.	17	18	19	20	21	22	23	<i>Einführung</i>
	24	25	26	27	28	29	30	<i>Aussagenlogik</i>
	31	1	2	3	4	5	6	<i>Normalformen</i>
	7	8	9	10	11	12	13	<i>Resolution</i>
Dez.	14	15	16	17	18	19	20	<i>Kompaktheit</i>
	21	22	23	24	25	26	27	<i>DPLL, Satz von Cook</i>
	28	29	30	1	2	3	4	<i>Strukturen und FO</i>
	5	6	7	8	9	10	11	<i>Prädikatenlogik</i>
	12	13	14	15	16	17	18	<i>Komplexität von FO</i>
Jan.	19	20	21	22	23	24	25	<i>Weihnachten</i>
	26	27	28	29	30	31	1	<i>Neujahr</i>
	2	3	4	5	6	7	8	<i>Normalformen</i>
	9	10	11	12	13	14	15	<i>Definierbarkeit</i>
	16	17	18	19	20	21	22	<i>EF-Spiele</i>
Feb.	23	24	25	26	27	28	29	<i>EF-Spiele</i>
	30	31	1	2	3	4	5	<i>Sequenzenkalkül AL</i>
	6	7	8	9	10	11	12	<i>Sequenzenkalkül FO</i>
	13	14	15	16	17	18	19	<i>Ausblick</i>

7.1 Einleitung

Grenzen der Aussagenlogik

Die Aussagenlogik formalisiert das Schließen über Aussagen die entweder wahr oder falsch sein können.

Die eigentliche Bedeutung der Aussagen ist dabei irrelevant.

Um über Aussagen der folgenden Form zu sprechen, ist die Aussagenlogik also nicht geeignet:

Für jede reelle Zahl x gibt es eine natürliche Zahl $n > x$ die größer als x ist.

Wir müssen über verschiedene Arten von Objekten sprechen.

Einige Aussagen müssen für alle Objekte gelten, andere nur für einige.

Prädikatenlogik

Wir führen eine Logik ein, in der solche Aussagen gemacht werden können.

Intuitiv haben wir

- **Variablen** für Elemente einer Menge von Objekten, z. B. den reellen Zahlen, anstatt nur wahr oder falsch.
- Möglichkeiten, Variablen zu vergleichen, z.B. $x < y$, $x = y$ abhängig vom Kontext.
- **Verknüpfungen** wie \neg , \vee , \wedge , \rightarrow , \leftrightarrow um komplexere Formeln bilden zu können.
- Möglichkeiten um zu sagen, dass es **ein Element gibt** mit bestimmten Eigenschaften oder das **alle Elemente** bestimmte Eigenschaften haben.

$$\forall x (\mathbb{R}(x) \rightarrow \exists y (\mathbb{N}(y) \wedge x < y))$$

Um dies zu erreichen, müssen wir folgendes festlegen:

- Den **Kontext** in dem wir arbeiten, d.h. Relationen $<$, $+$, ... die wir verwenden dürfen \rightsquigarrow Strukturen
- Die **logische Sprache** in der wir die Eigenschaften ausdrücken wollen \rightsquigarrow Prädikatenlogik

Relationen

Relationen

Definition. Sei $k \geq 1$ und A eine Menge.

1. A^k ist die Menge aller k -Tupel von Elementen aus A .
2. Eine k -stellige Relation auf A ist eine Teilmenge von A^k .

Bemerkung. Wir erlauben auch $k = 0$.

Eine nullstellige Relation $R \subseteq A^0$ ist entweder \emptyset oder $\{()\}$.

Notation.

Für einige spezielle Relationssymbole wie $<$, $=$ benutzen wir **Infix** Notation.

Z.B. schreiben wir $a = b$ und $a < b$ statt $(a, b) \in =$ bzw. $(a, b) \in <$.

Eigenschaften binärer Relationen

Definition. Eine binäre Relation $R \subseteq A^2$ einer Menge A ist

- **reflexiv**, wenn $(a, a) \in R$, für alle $a \in A$.
- **symmetrisch**, wenn aus $(a, b) \in R$ immer $(b, a) \in R$ folgt, für alle $a, b \in A$.
- **antisymmetrisch**, wenn $(a, b) \in R$ und $(b, a) \in R$ zusammen $a = b$ impliziert, für alle $a, b \in A$.
- **transitiv**, wenn aus $(a, b) \in R$ und $(b, c) \in R$ immer $(a, c) \in R$ folgt, für alle $a, b, c \in A$.

Definition. Eine Äquivalenzrelation ist eine binäre Relation, die **reflexiv**, **transitiv** und **symmetrisch** ist.

Beispiele für Äquivalenzrelationen

Beispiel. Einige Beispiele für Äquivalenzrelationen

- **Gleichheit.** Für jede Menge A

$$\{(a, a) \in A^2 : a \in A\}$$

- **Gleichmächtigkeit.** Für jede Menge A

$$\{(B, C) \in \mathcal{P}(A)^2 : B, C \text{ haben die gleiche Kardinalität}\}$$

- **Logische Äquivalenz.**

$$\{(\varphi, \psi) \in \mathbf{AL}^2 : \varphi \equiv \psi\}$$

Relationen.

reflexiv:

$$(a, a) \in R, \text{ für alle } a \in A.$$

symmetrisch:

$$(a, b) \in R \Rightarrow (b, a) \in R.$$

antisymmetrisch:

$$(a, b) \in R \wedge (b, a) \in R \Rightarrow a = b.$$

transitiv:

$$(a, b) \in R \wedge (b, c) \in R \Rightarrow (a, c) \in R.$$

Definition. Eine Äquivalenzrelation ist eine binäre Relation, die reflexiv, transitiv und symmetrisch ist.

Ordnungen

Definition. Sei A eine Menge.

1. Eine (strikte) partielle Ordnung $<$ über A ist eine irreflexive und transitive binäre Relation über A .
2. Eine (strikte) lineare Ordnung $<$ über A ist eine partielle Ordnung über A , so dass für alle $a, b \in A$:

$$a < b, \quad a = b \quad \text{oder} \quad b < a \quad (*)$$

Relationen.

reflexiv:

$$(a, a) \in R, \text{ für alle } a \in A.$$

symmetrisch:

$$(a, b) \in R \Rightarrow (b, a) \in R.$$

antisymmetrisch:

$$(a, b) \in R \wedge (b, a) \in R \Rightarrow a = b.$$

transitiv:

$$(a, b) \in R \wedge (b, c) \in R \Rightarrow (a, c) \in R.$$

Definition. Eine Äquivalenzrelation ist eine binäre Relation, die reflexiv, transitiv und symmetrisch ist.

7.2 Strukturen

Signaturen

Definition. Eine **Signatur** ist eine Menge σ von **Relationssymbolen**, **Funktionssymbolen** und **Konstantensymbolen**.

Jedes Relationssymbol $R \in \sigma$ und jedes Funktionssymbol $f \in \sigma$ hat eine **Stelligkeit**

$$ar(R) \in \mathbb{N} \text{ bzw. } ar(f) \in \mathbb{N}.$$

Beispiel.

$$\sigma := \{\langle, +, \cdot, 0, 1\} \text{ mit Stelligkeiten } ar(\langle) = ar(+) = ar(\cdot) = 2.$$

Notation.

- Wir verwenden griechische Symbole σ, τ für Signaturen.
- Für Relationssymbole verwenden wir $R, P, Q, R', \langle, \leq, \dots$
- Für Funktionssymbole verwenden wir $f, g, h, +, *, \cdot, \dots$
- Für Konstantensymbole verwenden wir $c, d, 0, 1, \dots$

Strukturen

Definition. Sei σ eine Signatur.

Eine σ -Struktur \mathcal{A} besteht aus

- einer nicht-leeren Menge A , dem Universum von \mathcal{A}
- einer k -stelligen Relation $R^A \subseteq A^k$ für jedes k -stellige Relationssymbol $R \in \sigma$
- einer k -stelligen Funktion $f^A : A^k \rightarrow A$ für jedes k -stellige Funktionssymbol $f \in \sigma$
- einem Element $c^A \in A$ für jedes Konstantensymbol $c \in \sigma$.

Bemerkung. Man beachte den Unterschied zwischen einem Symbol $R \in \sigma$ oder $f \in \sigma$ und seiner Interpretation R^A bzw. f^A .

Strukturen

Notation.

Wir verwenden kalligraphische Buchstaben $\mathcal{A}, \mathcal{B}, \dots$ für Strukturen und entsprechende lateinische Buchstaben A, B, \dots für deren Universen.

Wir schreiben σ -Strukturen oft als Tupel

$$\mathcal{A} := (A, (R^A)_{R \in \sigma})$$

oder, falls $\sigma := \{R_1, \dots, R_n\}$ endlich ist, auch

$$\mathcal{A} := (A, R_1^A, \dots, R_n^A).$$

Bemerkung. In der Literatur werden Strukturen oft mit Buchstaben in Fraktur bezeichnet: \mathfrak{A} \mathfrak{B} \mathfrak{C}

Beispiel: Arithmetische Strukturen

Sei $\sigma_{ar} := \{+, *, 0, 1\}$ die Signatur der Arithmetik, wobei

- $+, *$ binäre Funktionssymbole und
- $0, 1$ Konstantensymbole sind.

Wir können σ -Strukturen zur Modellierung von Körpern, etc. benutzen.

Beispiel. σ_{ar} -Struktur $\mathcal{N} := (\mathbb{N}, +^{\mathcal{N}}, *^{\mathcal{N}}, 0^{\mathcal{N}}, 1^{\mathcal{N}})$ mit Universum \mathbb{N} und

- $+^{\mathcal{N}}, *^{\mathcal{N}}$ als Addition bzw. Multiplikation der natürlichen Zahlen und
- $0^{\mathcal{N}} := 0$ und $1^{\mathcal{N}} := 1$.

Beispiel. Eine andere σ_{ar} -Struktur ist $\mathcal{Z} := (\mathbb{Z}, +^{\mathcal{Z}}, *^{\mathcal{Z}}, 0^{\mathcal{Z}}, 1^{\mathcal{Z}})$ mit Universum \mathbb{Z} und

- $+^{\mathcal{Z}}, *^{\mathcal{Z}}$ als Addition und Multiplikation der ganzen Zahlen und
- $0^{\mathcal{Z}} := 0$ und $1^{\mathcal{Z}} := 1$.

Beispiel: Arithmetische Strukturen

Bemerkung. σ_{ar} -Strukturen müssen nicht „natürliche“ arithmetische Strukturen wie die reellen oder ganzen Zahlen sein.

Wir können genauso eine σ_{ar} -Struktur

$$\mathcal{A} := (\mathbb{N}, +^{\mathcal{A}}, *^{\mathcal{A}}, 0^{\mathcal{A}}, 1^{\mathcal{A}})$$

mit Universum \mathbb{N} definieren, wobei

- $+^{\mathcal{A}}(a, b) := a^2 + b^2$,
- $*^{\mathcal{A}}$ die übliche **Addition** der natürlichen Zahlen ist und
- $0^{\mathcal{A}} := 17$ sowie $1^{\mathcal{A}} := 0$.

Tropische Geometrie. $\{+, \cdot\}$ -Struktur $\mathfrak{T} := (\mathbb{R}, +^{\mathfrak{T}}, \cdot^{\mathfrak{T}})$ wobei

- $a +^{\mathfrak{T}} b := \max\{a, b\}$
- $a \cdot^{\mathfrak{T}} b := a + b$.

Graphen als Strukturen

Definition. Sei $\sigma_{\text{Graph}} := \{E\}$ die Signatur der Graphen.

Mit jedem gerichteten Graph (V, E) assoziieren wir eine σ_{Graph} -Struktur $\mathcal{G} := (G, E^{\mathcal{G}})$ mit

- $G := V$
- $E^{\mathcal{G}} := E$.

Notation. Für Graphen und deren Strukturen \mathcal{G} weichen wir bisweilen von der Konvention ab und bezeichnen das Universum von \mathcal{G} als V .

Gefärbte Graphen

Gefärbte Graphen. Jeder Knoten kann mit einer Farbe aus einer festen Menge \mathcal{C} von Farben gefärbt sein.

Sei \mathcal{C} eine endliche Menge und sei $\sigma := \{E\} \cup \mathcal{C}$ mit $E \notin \mathcal{C}$.

Wir modellieren \mathcal{C} -gefärbte Graphen (V, E) als Strukturen

$$\mathcal{G} := (V, E^{\mathcal{G}}, (C^{\mathcal{G}})_{c \in \mathcal{C}})$$

wobei $C^{\mathcal{G}}$ alle Knoten mit Farbe \mathcal{C} enthält.

Beispiel.

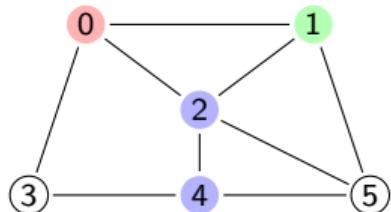
Sei $\mathcal{C} := \{\text{Rot, Grün, Blau}\}$.

In \mathcal{G} gilt:

$$\text{Rot}^{\mathcal{G}} = \{0\},$$

$$\text{Grün}^{\mathcal{G}} = \{1\} \text{ und}$$

$$\text{Blau}^{\mathcal{G}} = \{2, 4\}.$$



Prädikatenlogik

Wir führen eine Logik ein, in der solche Aussagen gemacht werden können.

Intuitiv haben wir

- **Variablen** für Elemente einer Menge von Objekten, z. B. den reellen Zahlen, anstatt nur wahr oder falsch.
- Möglichkeiten, Variablen zu vergleichen, z.B. $x < y$, $x = y$ abhängig vom Kontext.
- **Verknüpfungen** wie \neg , \vee , \wedge , \rightarrow , \leftrightarrow um komplexere Formeln bilden zu können.
- Möglichkeiten um zu sagen, dass es **ein Element gibt** mit bestimmten Eigenschaften oder das **alle Elemente** bestimmte Eigenschaften haben.

$$\forall x (\mathbb{R}(x) \rightarrow \exists y (\mathbb{N}(y) \wedge x < y))$$

Um dies zu erreichen, müssen wir folgendes festlegen:

- Den **Kontext** in dem wir arbeiten, d.h. Relationen $<$, $+$, ... die wir verwenden dürfen \rightsquigarrow Strukturen
- Die **logische Sprache** in der wir die Eigenschaften ausdrücken wollen \rightsquigarrow Prädikatenlogik

7.3 Syntax der Prädikatenlogik

Syntax der Prädikatenlogik: Terme

Definition (Variablen erster Stufe).

Wir fixieren eine abzählbar unendliche Menge Var von *Variablen erster Stufe*, oder kurz *Variablen*, die v_i für alle $i \in \mathbb{N}$ enthält.

Definition. Sei σ eine Signatur.

Die Menge \mathcal{T}_σ der σ -Terme ist induktiv definiert wie folgt:

Basisfall.

- $v_i \in \mathcal{T}_\sigma$ für alle $v_i \in \text{Var}$
- $c \in \mathcal{T}_\sigma$ für alle Konstantensymbole $c \in \sigma$

Induktionsschritt.

Ist $f \in \sigma$ ein Funktionssymbol, $\text{ar}(f) = k$ und $t_1, \dots, t_k \in \mathcal{T}_\sigma$, dann ist

$$f(t_1, \dots, t_k) \in \mathcal{T}_\sigma.$$

Ein Term, in dem keine Variablen vorkommen, heißt *Grundterm*.

Syntax der Prädikatenlogik: Terme

Definition (Variablen erster Stufe).

Wir fixieren eine abzählbar unendliche Menge Var von *Variablen erster Stufe*, oder kurz *Variablen*, die v_i für alle $i \in \mathbb{N}$ enthält.

Definition. Sei σ eine Signatur.

Die Menge \mathcal{T}_σ der σ -Terme ist induktiv definiert wie folgt:

Basisfall.

- $v_i \in \mathcal{T}_\sigma$ für alle $v_i \in \text{Var}$
- $c \in \mathcal{T}_\sigma$ für

Bemerkung. Wir werden $+, \cdot, \dots$ in Infixnotation verwenden, auch wenn das streng genommen keine prädikatenlogischen Terme sind.

Induktionsschritt

Ist $f \in \sigma$ ein Funktionssymbol, $\text{ar}(f) = k$ und $t_1, \dots, t_k \in \mathcal{T}_\sigma$, dann ist

$$f(t_1, \dots, t_k) \in \mathcal{T}_\sigma.$$

Ein Term, in dem keine Variablen vorkommen, heißt **Grundterm**.

Beispiele.

Sei $\sigma := \{<, +, *, 0, 1\}$.

Folgende Ausdrücke sind σ -Terme.

- $((x + x) * y)$
- $((1 + 1) * 0) + 1)$
- $x * x + y$

Syntax der Prädikatenlogik: Formeln

Definition. Sei σ eine Signatur. Die Menge $\text{FO}[\sigma]$ der *prädikatenlogischen Formeln über σ* ist induktiv wie folgt definiert.

Basisfall.

- $t = t' \in \text{FO}[\sigma]$ für alle Terme $t, t' \in \mathcal{T}_\sigma$.
- $R(t_1, \dots, t_k) \in \text{FO}[\sigma]$, für alle k -stelligen Relationssymbole $R \in \sigma$ und alle $t_1, \dots, t_k \in \mathcal{T}_\sigma$.

Formeln der Form $t = t'$ und $R(t_1, \dots, t_k)$ heißen *atomar*.

Induktionsschritt.

- Wenn $\varphi \in \text{FO}[\sigma]$, dann $\neg\varphi \in \text{FO}[\sigma]$.
- Wenn $\varphi, \psi \in \text{FO}[\sigma]$, dann $(\varphi \vee \psi) \in \text{FO}[\sigma]$, $(\varphi \wedge \psi) \in \text{FO}[\sigma]$, $(\varphi \rightarrow \psi) \in \text{FO}[\sigma]$ und $(\varphi \leftrightarrow \psi) \in \text{FO}[\sigma]$.
- Wenn $\varphi \in \text{FO}[\sigma]$ und $x \in \text{Var}$, dann $\exists x\varphi \in \text{FO}[\sigma]$ und $\forall x\varphi \in \text{FO}[\sigma]$.

$\text{FO}[\sigma]$ heißt die *Prädikatenlogik über σ* oder die *Sprache/Logik erster Stufe über σ* .

Beispiele

Die Sprache der Graphen. Sei $\sigma_{Graph} := \{E\}$.

Die folgenden Ausdrücke sind Formeln in $FO[\sigma_{Graph}]$.

- $E(x, y)$
- $\exists x \forall y (E(x, y) \vee x = y)$
- $\exists x_1 \exists x_2 ((E(x, x_1) \wedge E(x_1, x_2)) \wedge E(x_2, y))$

Die Sprache der Arithmetik und Ordnung. Sei $\sigma := \{<, +, *\}$.

Die folgenden Ausdrücke sind Formeln in $FO[\sigma]$.

- $x < x + x$
- $\forall x \exists y x < y$
- $\exists x \neg \exists y y < x$

Definition $FO[\sigma]$.

Atomare Formeln.

- $t = t' \in FO[\sigma]$ für alle $t, t' \in T_\sigma$.
- $R(t_1, \dots, t_k) \in FO[\sigma]$, für alle $R \in \sigma$ und $t_1, \dots, t_k \in T_\sigma$.

Zusammengesetzte Formeln.

- Wenn $\varphi \in FO[\sigma]$, dann $\neg \varphi \in FO[\sigma]$.
- Wenn $\varphi, \psi \in FO[\sigma]$, dann $(\varphi \vee \psi), (\varphi \wedge \psi) \in FO[\sigma]$, $(\varphi \rightarrow \psi), (\varphi \leftrightarrow \psi) \in FO[\sigma]$.
- Wenn $\varphi \in FO[\sigma]$ und $x \in Var$, dann $\exists x \varphi \in FO[\sigma]$ und $\forall x \varphi \in FO[\sigma]$.

Bemerkung. Wir werden $<$, $+$ in Infixnotation verwenden, auch wenn das streng genommen keine prädikatenlogischen Formeln sind.

Notation

Notation.

1. Wir vereinbaren die gleichen Klammerregeln wie in der Aussagenlogik.

Wir schreiben also $(\varphi_1 \wedge \varphi_2 \wedge \varphi_3)$ statt $((\varphi_1 \wedge \varphi_2) \wedge \varphi_3)$.

Vorsicht: Die Formeln $\exists x E(x, x) \vee \exists z E(x, z)$ und $\exists x (E(x, x) \vee \exists z E(x, z))$ haben eine komplett andere Bedeutung.

2. Relationssymbole $<, \leq, >, \geq, \dots$ und Funktionssymbole $+, *, \dots$ werden wir oft in Infixnotation schreiben.

D.h. wir verwenden Formeln der Form $x < y + z$ statt korrekt zu schreiben: $< (x, +(y, z))$.

Freie und gebundene Variablen

Definition. Sei σ eine Signatur.

Wir definieren $\text{var}(t)$ als die Menge der in einem σ -Term t vorkommenden Variablen.

Formal wird $\text{var}(t)$ wie folgt induktiv definiert:

- Wenn $t := v_i \in \text{Var}$, dann $\text{var}(t) := \{v_i\}$.
- Wenn $t := c$ für ein Konstantensymbol $c \in \sigma$, dann
$$\text{var}(t) := \emptyset.$$
- Wenn $t := f(t_1, \dots, t_k)$ für ein k -stelliges Funktionssymbol $f \in \sigma$ und Terme $t_1, \dots, t_k \in \mathcal{T}_\sigma$, dann
$$\text{var}(t) := \text{var}(t_1) \cup \dots \cup \text{var}(t_k).$$

Freie und gebundene Variablen

Definition. Sei σ eine Signatur und sei $\varphi \in \text{FO}[\sigma]$.

Die Menge $\text{frei}(\varphi)$ der *freien Variablen* von φ ist induktiv definiert als:

- Wenn $\varphi := t_1 = t_2$, für $t_1, t_2 \in \mathcal{T}_\sigma$, dann $\text{frei}(\varphi) := \text{var}(t_1) \cup \text{var}(t_2)$.
- Wenn $\varphi := R(t_1, \dots, t_k)$ für $R \in \sigma$ und $t_1, \dots, t_k \in \mathcal{T}_\sigma$, dann $\text{frei}(\varphi) := \bigcup_{i=1}^k \text{var}(t_i)$.
- $\text{frei}(\neg\varphi) := \text{frei}(\varphi)$ für alle $\varphi \in \text{FO}[\sigma]$.
- $\text{frei}((\varphi * \psi)) := \text{frei}(\varphi) \cup \text{frei}(\psi)$ für alle $* \in \{\vee, \wedge, \rightarrow, \leftrightarrow\}$ und $\varphi, \psi \in \text{FO}[\sigma]$.
- Wenn $\varphi := \exists x\psi$ oder $\varphi := \forall x\psi$, für $x \in \text{Var}$ und $\psi \in \text{FO}[\sigma]$, dann $\text{frei}(\varphi) := \text{frei}(\psi) \setminus \{x\}$.

Beispiele.

- $\text{frei}(E(x, y)) = \{x, y\}$
- $\varphi := \exists x \forall y (x = y \vee E(x, y))$
 $\text{frei}(\varphi) = \emptyset$
- $\varphi := \exists x_1 (E(x, x_1) \wedge E(x_1, y))$
 $\text{frei}(\varphi) = \{x, y\}$
- $\varphi := \exists y (x < y \wedge \exists xy < x)$
 $\text{frei}(\varphi) = \{x\}$

Eine Formel φ mit $\text{frei}(\varphi) := \emptyset$ heißt ein *Satz*.

Eine Variable, die in φ vorkommt, aber nicht frei ist, heißt *gebunden*.

Wir schreiben $\varphi(v_1, \dots, v_k)$ um zu sagen, dass $\text{frei}(\varphi) \subseteq \{v_1, \dots, v_k\}$.

7.4 Semantik der Prädikatenlogik

Belegungen

Definition. Sei σ eine Signatur und \mathcal{A} eine σ -Struktur.

1. Eine *Belegung* in \mathcal{A} ist eine Funktion $\beta : \text{def}(\beta) \rightarrow A$ mit $\text{def}(\beta) \subseteq \text{Var}$.
 β ist *passend* für $f \in \text{FO}[\sigma] \cup \mathcal{T}_\sigma$, wenn $\text{frei}(f) \subseteq \text{def}(\beta)$.
2. Eine *σ -Interpretation* ist ein Paar (\mathcal{A}, β) , bestehend aus einer σ -Struktur \mathcal{A} und einer Belegung β in \mathcal{A} .
 $\mathcal{I} := (\mathcal{A}, \beta)$ ist passend für $f \in \text{FO}[\sigma] \cup \mathcal{T}_\sigma$, wenn β zu f passt.

Notation. Sei β eine Belegung, $x \in \text{Var}$ und $a \in A$.

1. $\beta[x/a]$ bezeichnet die Belegung β' mit $\beta'(y) := \beta(y)$ für alle $y \in \text{Var} \setminus \{x\}$ und $\beta'(x) := a$.
2. $\mathcal{I}[x/a]$ bezeichnet die Interpretation $(\mathcal{A}, \beta[x/a])$.

Semantik der Prädikatenlogik: Terme

Definition. Sei σ eine Signatur.

Induktiv über den Termaufbau definieren wir eine Funktion $\llbracket \cdot \rrbracket^{\mathcal{I}}$, die jedem Term $t \in \mathcal{T}_\sigma$ und jeder σ -Interpretation $\mathcal{I} := (\mathcal{A}, \beta)$ für t einen Wert $\llbracket t \rrbracket^{\mathcal{I}} \in \mathcal{A}$ zuweist.

Basisfall.

- $\llbracket x \rrbracket^{\mathcal{I}} := \beta(x)$ für alle $x \in \text{Var}$
- $\llbracket c \rrbracket^{\mathcal{I}} := c^{\mathcal{A}}$ für alle Konstantensymbole $c \in \sigma$.

Induktionsschritt.

Ist $f \in \sigma$ eine k -stelliges Funktionssymbol und $t_1, \dots, t_k \in \mathcal{T}_\sigma$ dann

$$\llbracket f(t_1, \dots, t_k) \rrbracket^{\mathcal{I}} := f^{\mathcal{A}}(\llbracket t_1 \rrbracket^{\mathcal{I}}, \dots, \llbracket t_k \rrbracket^{\mathcal{I}}).$$

Definition.

σ : Signatur, \mathcal{A} : σ -Struktur.

- **Belegung:** $\beta : \text{def}(\beta) \rightarrow \mathcal{A}$ mit $\text{def}(\beta) \subseteq \text{Var}$.
- β **passend** für f : $\text{frei}(f) \subseteq \text{def}(\beta)$.
- **σ -Interpretation:** (\mathcal{A}, β)

Semantik der Prädikatenlogik: Formeln

Definition. Sei σ eine Signatur.

Induktiv über den Formelaufbau definieren wir eine Funktion $\llbracket \cdot \rrbracket^{\mathcal{I}}$, die jeder Formel $\varphi \in \text{FO}[\sigma]$ und jeder σ -Interpretation $\mathcal{I} := (\mathcal{A}, \beta)$ für φ einen Wahrheitswert $\llbracket \varphi \rrbracket^{\mathcal{I}} \in \{0, 1\}$ zuordnet.

Basisfall.

- Für alle Terme $t, t' \in \mathcal{T}_{\sigma}$ definieren wir

$$\llbracket t = t' \rrbracket^{\mathcal{I}} := \begin{cases} 1 & \text{wenn } \llbracket t \rrbracket^{\mathcal{I}} = \llbracket t' \rrbracket^{\mathcal{I}} \\ 0 & \text{sonst.} \end{cases}$$

- Für alle k -stelligen Relationssymbole $R \in \sigma$ und alle Terme $t_1, \dots, t_k \in \mathcal{T}_{\sigma}$ definieren wir

$$\llbracket R(t_1, \dots, t_k) \rrbracket^{\mathcal{I}} := \begin{cases} 1 & \text{wenn } (\llbracket t_1 \rrbracket^{\mathcal{I}}, \dots, \llbracket t_k \rrbracket^{\mathcal{I}}) \in R^{\mathcal{A}} \\ 0 & \text{sonst.} \end{cases}$$

Definition.

σ : Signatur, \mathcal{A} : σ -Struktur.

- **Belegung:** $\beta : \text{def}(\beta) \rightarrow \mathcal{A}$ mit $\text{def}(\beta) \subseteq \text{Var}$.
- β **passend** für f : $\text{frei}(f) \subseteq \text{def}(\beta)$.
- **σ -Interpretation:** (\mathcal{A}, β)

Definition $\llbracket t \rrbracket^{\mathcal{I}}$.

Basisfall.

- $\llbracket x \rrbracket^{\mathcal{I}} := \beta(x)$ für $x \in \text{Var}$
- $\llbracket c \rrbracket^{\mathcal{I}} := c^{\mathcal{A}}$ für $c \in \sigma$.

Induktionsschritt.

- $\llbracket f(t_1, \dots, t_k) \rrbracket^{\mathcal{I}} := f^{\mathcal{A}}(\llbracket t_1 \rrbracket^{\mathcal{I}}, \dots, \llbracket t_k \rrbracket^{\mathcal{I}})$.

Semantik der Prädikatenlogik: Formeln

Induktionsschritt.

- Die Semantik der Verknüpfungen $\neg, \vee, \wedge, \rightarrow, \leftrightarrow$ ist wie in der Aussagenlogik definiert. Z.B. wenn $\varphi := \neg\psi \in \text{FO}[\sigma]$ dann definieren wir

$$\llbracket \varphi \rrbracket^{\mathcal{I}} := 1 - \llbracket \psi \rrbracket^{\mathcal{I}}.$$

- Wenn $\varphi := \exists x\psi \in \text{FO}[\sigma]$ dann definieren wir

$$\llbracket \varphi \rrbracket^{\mathcal{I}} := \begin{cases} 1 & \text{es gibt } a \in A, \text{ so dass } \llbracket \psi \rrbracket^{\mathcal{I}[x/a]} = 1 \\ 0 & \text{sonst.} \end{cases}$$

- Wenn $\varphi := \forall x\psi \in \text{FO}[\sigma]$ definieren wir

$$\llbracket \varphi \rrbracket^{\mathcal{I}} := \begin{cases} 1 & \llbracket \psi \rrbracket^{\mathcal{I}[x/a]} = 1 \text{ für alle } a \in A \\ 0 & \text{sonst.} \end{cases}$$

Definition.

σ : Signatur, \mathcal{A} : σ -Struktur.

- Belegung:** $\beta : \text{def}(\beta) \rightarrow A$ mit $\text{def}(\beta) \subseteq \text{Var}$.
- β passend für f :** $\text{frei}(f) \subseteq \text{def}(\beta)$.
- σ -Interpretation:** (\mathcal{A}, β)

Definition $\llbracket t \rrbracket^{\mathcal{I}}$.

Basisfall.

- $\llbracket x \rrbracket^{\mathcal{I}} := \beta(x)$ für $x \in \text{Var}$
- $\llbracket c \rrbracket^{\mathcal{I}} := c^{\mathcal{A}}$ für $c \in \sigma$.

Induktionsschritt.

- $\llbracket f(t_1, \dots, t_k) \rrbracket^{\mathcal{I}} := f^{\mathcal{A}}(\llbracket t_1 \rrbracket^{\mathcal{I}}, \dots, \llbracket t_k \rrbracket^{\mathcal{I}})$.

Die Modellbeziehung

Definition. Sei $\varphi \in \text{FO}[\sigma]$ eine Formel und sei $\Phi \subseteq \text{FO}[\sigma]$ eine Formelmenge.

1. Eine σ -Interpretation \mathcal{I} erfüllt φ , wenn \mathcal{I} zu φ passt und $\llbracket \varphi \rrbracket^{\mathcal{I}} = 1$.

Wir sagen auch: \mathcal{I} ist ein **Modell** von φ und schreiben $\mathcal{I} \models \varphi$.

2. Eine σ -Interpretation \mathcal{I} erfüllt Φ , wenn \mathcal{I} zu allen $\psi \in \Phi$ passt und $\llbracket \psi \rrbracket^{\mathcal{I}} = 1$ für alle $\psi \in \Phi$.

Wir sagen auch: \mathcal{I} ist ein **Modell** von Φ und schreiben $\mathcal{I} \models \Phi$.

Das Koinzidenzlemma

Lemma (Koinzidenzlemma).

Seien σ, τ, τ' Signaturen, so dass $\sigma \subseteq \tau \cap \tau'$ und

$\mathcal{I} := (\mathcal{A}, \beta)$: τ -Interpretation und $\mathcal{J} := (\mathcal{B}, \gamma)$: τ' -Interpretation,

so dass

- $A = B$ und
- $S^A = S^B$ für alle Symbole, die in σ vorkommen.

Dann gilt:

1. Ist $t \in \mathcal{T}_\sigma$ ein σ -Term und $\beta(x) = \gamma(x)$ für alle $x \in \text{var}(t)$, dann

$$\llbracket t \rrbracket^{\mathcal{I}} = \llbracket t \rrbracket^{\mathcal{J}}.$$

2. Ist $\varphi \in \text{FO}[\sigma]$ eine Formel und $\beta(x) = \gamma(x)$ für alle $x \in \text{frei}(\varphi)$, dann

$$\llbracket \varphi \rrbracket^{\mathcal{I}} = \llbracket \varphi \rrbracket^{\mathcal{J}}.$$

7.5 Beispiele für prädikatenlogische Formeln

Beispiele

Sei $\sigma := \{+, *, 0, 1\}$: Signatur der Arithmetik

$\mathcal{A} := (\mathbb{N}, +^{\mathcal{A}}, *^{\mathcal{A}}, 0^{\mathcal{A}}, 1^{\mathcal{A}})$: Struktur über \mathbb{N} mit üblicher Interpretation von $+^{\mathcal{A}}, *^{\mathcal{A}}, 0^{\mathcal{A}}, 1^{\mathcal{A}}$.

Sei $\beta : x \mapsto 2, y \mapsto 3$ und $\mathcal{I} := (\mathcal{A}, \beta)$.

Beispiele.

- $\llbracket x * x \rrbracket^{\mathcal{I}} := \beta(x) *^{\mathcal{A}} \beta(x) = 4$.
- $\llbracket x * x = y + 1 \rrbracket^{\mathcal{I}} = 1$, da $\beta(x) *^{\mathcal{A}} \beta(x) := 4 = \beta(y) +^{\mathcal{A}} 1^{\mathcal{A}} := 3 + 1$.
- Sei $\varphi(x, y) := \exists z (x * x = y + z)$.

Um zu zeigen, dass $\llbracket \varphi \rrbracket^{\mathcal{I}} = 1$ müssen wir ein Element $a \in \mathbb{N}$ mit $\llbracket x * x = y + z \rrbracket^{\mathcal{I} \cup \{z \mapsto a\}} = 1$ finden.

Sei $\beta' := \beta \cup \{z \mapsto 1\}$ und $\mathcal{I}' := (\mathcal{A}, \beta')$.

Dann gilt $\llbracket x * x = y + z \rrbracket^{\mathcal{I}'} = 1$ und daher $\llbracket \exists z (x * x = y + z) \rrbracket^{\mathcal{I}} = 1$.

Semantik.

- $\llbracket t = t' \rrbracket^{\mathcal{I}} := \begin{cases} 1 & \text{wenn } \llbracket t \rrbracket^{\mathcal{I}} = \llbracket t' \rrbracket^{\mathcal{I}} \\ 0 & \text{sonst.} \end{cases}$
- $\llbracket R(t_1, \dots, t_k) \rrbracket^{\mathcal{I}} := \begin{cases} 1 & (\llbracket t_1 \rrbracket^{\mathcal{I}}, \dots, \llbracket t_k \rrbracket^{\mathcal{I}}) \in R^{\mathcal{A}} \\ 0 & \text{sonst.} \end{cases}$
- $\neg, \vee, \wedge, \rightarrow, \leftrightarrow$ wie in AL
- $\llbracket \exists x \psi \rrbracket^{\mathcal{I}} := \begin{cases} 1 & \llbracket \psi \rrbracket^{\mathcal{I}[x/a]} = 1 \\ & \text{für ein } a \in A \\ 0 & \text{sonst.} \end{cases}$
- $\llbracket \forall x \psi \rrbracket^{\mathcal{I}} := \begin{cases} 1 & \llbracket \psi \rrbracket^{\mathcal{I}[x/a]} = 1 \\ & \text{für alle } a \in A \\ 0 & \text{sonst.} \end{cases}$

Beispiele: Arithmetik

$\sigma := \{+, *, <, 0, 1\}$: Signatur der Arithmetik

$\mathcal{A} := (\mathbb{N}, +^{\mathcal{A}}, *^{\mathcal{A}}, <^{\mathcal{A}}, 0^{\mathcal{A}}, 1^{\mathcal{A}})$: Struktur über den natürlichen Zahlen mit der üblichen Interpretation von $+, *, <, 0, 1$

Beispiele. Was drücken folgende Formeln aus?

- $\varphi_1(x) := \forall y \forall z (y * z = x \rightarrow (y = 1 \vee z = 1))$

- $\varphi_2 := \forall y \exists x (y < x \wedge \varphi_1(x))$

Anmerkung. Hier wird φ_1 in die Formel φ_2 „eingesetzt“. Im allgemeinen nicht unproblematisch, siehe nächste Woche.

- $\varphi_3 := \forall x \forall y (x * y = y * x)$

- $\varphi_4 := \forall x \forall y \forall z ((x * y) * z = x * (y * z))$

Semantik.

- $\llbracket t = t' \rrbracket^{\mathcal{I}} := \begin{cases} 1 & \text{wenn } \llbracket t \rrbracket^{\mathcal{I}} = \llbracket t' \rrbracket^{\mathcal{I}} \\ 0 & \text{sonst.} \end{cases}$
- $\llbracket R(t_1, \dots, t_k) \rrbracket^{\mathcal{I}} := \begin{cases} 1 & (\llbracket t_1 \rrbracket^{\mathcal{I}}, \dots, \llbracket t_k \rrbracket^{\mathcal{I}}) \in R^{\mathcal{A}} \\ 0 & \text{sonst.} \end{cases}$
- $\neg, \vee, \wedge, \rightarrow, \leftrightarrow$ wie in AL
- $\llbracket \exists x \psi \rrbracket^{\mathcal{I}} := \begin{cases} 1 & \llbracket \psi \rrbracket^{\mathcal{I}[x/a]} = 1 \\ & \text{für ein } a \in A \\ 0 & \text{sonst.} \end{cases}$
- $\llbracket \forall x \psi \rrbracket^{\mathcal{I}} := \begin{cases} 1 & \llbracket \psi \rrbracket^{\mathcal{I}[x/a]} = 1 \\ & \text{für alle } a \in A \\ 0 & \text{sonst.} \end{cases}$

Beispiele: Ordnungen

Sei $\sigma := \{<\}$ die Signatur strikter Ordnungen.

1. „ $<$ ist irreflexiv“

für alle x gilt $x \not< x$

$$\forall x \neg x < x$$

2. „ $<$ ist transitiv“

für alle x, y, z , wenn $x < y$ und $y < z$ dann $x < z$

$$\forall x \forall y \forall z ((x < y \wedge y < z) \rightarrow x < z)$$

3. Das heißt, wir können wie folgt sagen, dass $<$ eine partielle Ordnung ist:

$$\varphi_{\text{par-ord}} := (\forall x \neg x < x) \wedge (\forall x \forall y \forall z ((x < y \wedge y < z) \rightarrow x < z))$$

4. “ $<$ ist total”: $\varphi_t := \forall x \forall y (x < y \vee x = y \vee y < x)$

5. Eine lineare Ordnung $<$ wird formalisiert durch $\varphi_{\text{ord}} := \varphi_{\text{par-ord}} \wedge \varphi_t$.

Semantik.

- $\llbracket t = t' \rrbracket^{\mathcal{I}} := \begin{cases} 1 & \text{wenn } \llbracket t \rrbracket^{\mathcal{I}} = \llbracket t' \rrbracket^{\mathcal{I}} \\ 0 & \text{sonst.} \end{cases}$
- $\llbracket R(t_1, \dots, t_k) \rrbracket^{\mathcal{I}} := \begin{cases} 1 & (\llbracket t_1 \rrbracket^{\mathcal{I}}, \dots, \llbracket t_k \rrbracket^{\mathcal{I}}) \in R^A \\ 0 & \text{sonst.} \end{cases}$
- $\neg, \vee, \wedge, \rightarrow, \leftrightarrow$ wie in AL
- $\llbracket \exists x \psi \rrbracket^{\mathcal{I}} := \begin{cases} 1 & \llbracket \psi \rrbracket^{\mathcal{I}[x/a]} = 1 \\ & \text{für ein } a \in A \\ 0 & \text{sonst.} \end{cases}$
- $\llbracket \forall x \psi \rrbracket^{\mathcal{I}} := \begin{cases} 1 & \llbracket \psi \rrbracket^{\mathcal{I}[x/a]} = 1 \\ & \text{für alle } a \in A \\ 0 & \text{sonst.} \end{cases}$

Beispiele: Graphen

Die Sprache der Graphen. Sei $\sigma_{Graph} := \{E\}$.

- $\varphi := \exists x \forall y (x = y \vee E(x, y))$

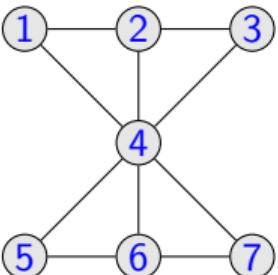
$$frei(\varphi) := \emptyset$$

φ gilt in einem Graph, wenn es einen Knoten mit Kanten zu allen anderen gibt.

- $\varphi := \exists x_1 \exists x_2 ((E(x, x_1) \wedge E(x_1, x_2)) \wedge E(x_2, y))$

$$frei(\varphi) := \{x, y\}.$$

es gibt einen Pfad der Länge 1 oder 3 von $\beta(x)$ zu $\beta(y)$



Semantik.

- $\llbracket t = t' \rrbracket^{\mathcal{I}} := \begin{cases} 1 & \text{wenn } \llbracket t \rrbracket^{\mathcal{I}} = \llbracket t' \rrbracket^{\mathcal{I}} \\ 0 & \text{sonst.} \end{cases}$
- $\llbracket R(t_1, \dots, t_k) \rrbracket^{\mathcal{I}} := \begin{cases} 1 & (\llbracket t_1 \rrbracket^{\mathcal{I}}, \dots, \llbracket t_k \rrbracket^{\mathcal{I}}) \in R^A \\ 0 & \text{sonst.} \end{cases}$
- $\neg, \vee, \wedge, \rightarrow, \leftrightarrow$ wie in AL
- $\llbracket \exists x \psi \rrbracket^{\mathcal{I}} := \begin{cases} 1 & \llbracket \psi \rrbracket^{\mathcal{I}[x/a]} = 1 \\ & \text{für ein } a \in A \\ 0 & \text{sonst.} \end{cases}$
- $\llbracket \forall x \psi \rrbracket^{\mathcal{I}} := \begin{cases} 1 & \llbracket \psi \rrbracket^{\mathcal{I}[x/a]} = 1 \\ & \text{für alle } a \in A \\ 0 & \text{sonst.} \end{cases}$

Ein ausführliches Beispiel: Vertex Cover

Definition. Ein *vertex cover* eines ung. Graphs $G := (V, E)$ ist eine Menge $X \subseteq V$, s.d. für alle Kanten $e := (u, v) \in E$, $u \in X$ oder $v \in X$.

Problem. Gegeben G , $k \in \mathbb{N}$, enthält G ein vertex cover der Größe $\leq k$.

Kann dies in der Prädikatenlogik formalisiert werden?

Schritt 1. Schreiben Sie das Problem in natürlicher Sprache auf.

G enthält ein vertex cover der Größe $\leq k$ wenn

- es gibt eine Menge X von $\leq k$ Knoten, so dass
- jede Kante (u, v) einen Endpunkt u oder v in X hat.

Diese Formalisierung benutzt

- eine Menge X über die wir in der Prädikatenlogik nicht quantifizieren können
- eine Aussage der Form **für alle Kanten**, was wir ebenfalls nicht benutzen können

Semantik.

- $\llbracket t = t' \rrbracket^{\mathcal{I}} := \begin{cases} 1 & \text{wenn } \llbracket t \rrbracket^{\mathcal{I}} = \llbracket t' \rrbracket^{\mathcal{I}} \\ 0 & \text{sonst.} \end{cases}$
- $\llbracket R(t_1, \dots, t_k) \rrbracket^{\mathcal{I}} := \begin{cases} 1 & (\llbracket t_1 \rrbracket^{\mathcal{I}}, \dots, \llbracket t_k \rrbracket^{\mathcal{I}}) \in R^A \\ 0 & \text{sonst.} \end{cases}$
- $\neg, \vee, \wedge, \rightarrow, \leftrightarrow$ wie in AL
- $\llbracket \exists x \psi \rrbracket^{\mathcal{I}} := \begin{cases} 1 & \llbracket \psi \rrbracket^{\mathcal{I}[x/a]} = 1 \\ & \text{für ein } a \in A \\ 0 & \text{sonst.} \end{cases}$
- $\llbracket \forall x \psi \rrbracket^{\mathcal{I}} := \begin{cases} 1 & \llbracket \psi \rrbracket^{\mathcal{I}[x/a]} = 1 \\ & \text{für alle } a \in A \\ 0 & \text{sonst.} \end{cases}$

Ein ausführliches Beispiel: Vertex Cover

Definition. Ein *vertex cover* eines ung. Graphs $G := (V, E)$ ist eine Menge $X \subseteq V$, s.d. für alle Kanten $e := (u, v) \in E$, $u \in X$ oder $v \in X$.

Problem. Gegeben G , $k \in \mathbb{N}$, enthält G ein vertex cover der Größe $\leq k$.

Kann dies in der Prädikatenlogik formalisiert werden?

Schritt 1. Schreiben Sie das Problem in natürlicher Sprache auf.

G enthält ein vertex cover der Größe $\leq k$ wenn

- es gibt eine Menge X von $\leq k$ Knoten, so dass
- jede Kante (u, v) einen Endpunkt u oder v in X hat.

Diese Formalisierung benutzt

- eine Menge X über die wir in der Prädikatenlogik nicht quantifizieren können
- eine Aussage der Form **für alle Kanten**, was wir ebenfalls nicht benutzen können

Semantik.

- $\llbracket t = t' \rrbracket^{\mathcal{I}} := \begin{cases} 1 & \text{wenn } \llbracket t \rrbracket^{\mathcal{I}} = \llbracket t' \rrbracket^{\mathcal{I}} \\ 0 & \text{sonst.} \end{cases}$
- $\llbracket R(t_1, \dots, t_k) \rrbracket^{\mathcal{I}} := \begin{cases} 1 & (\llbracket t_1 \rrbracket^{\mathcal{I}}, \dots, \llbracket t_k \rrbracket^{\mathcal{I}}) \in R^A \\ 0 & \text{sonst.} \end{cases}$
- $\neg, \vee, \wedge, \rightarrow, \leftrightarrow$ wie in AL
- $\llbracket \exists x \psi \rrbracket^{\mathcal{I}} := \begin{cases} 1 & \llbracket \psi \rrbracket^{\mathcal{I}[x/a]} = 1 \\ & \text{für ein } a \in A \\ 0 & \text{sonst.} \end{cases}$
- $\llbracket \forall x \psi \rrbracket^{\mathcal{I}} := \begin{cases} 1 & \llbracket \psi \rrbracket^{\mathcal{I}[x/a]} = 1 \\ & \text{für alle } a \in A \\ 0 & \text{sonst.} \end{cases}$

Anwendung: Relationale Datenbanken

Beispiel: Relationale Datenbanken

Eine **relationale Datenbank** ist eine endliche Menge von “**Tabellen**”.

Z. B. könnte eine Filmdatenbank wie imdb.org wie folgt aussehen

Schauspieler		
Schausp.	ID	Geburtsdatum
George Clooney	1	6. Mai 1961
Scarlett Johansson	2	22. November 1984
Jeff Daniels	3	19. Februar 1955
...

Filme		
Titel	Regie	Schau.
Good night ... and good luck	George Clooney	1
Good night ... and good luck	George Clooney	3
Lost in translation	Sofia Coppola	2
...

Die Menge τ von Tabellennamen heißt **Datenbankschema**.

Beispiel: Relationale Datenbanken

Relationale Datenbanken.

- Jede **Spalte** einer Tabelle in der Datenbank enthält Einträge vom selben Typ, z.B. Wörter oder Zahlen.

In Datenbankterminologie werden Spaltenname **Attribute** genannt.

Jedes Attribut i hat einen Typ D_i , genannt **domain**.

- Jede **Zeile** der Tabelle enthält ein Tupel $(x_1, \dots, x_n) \in D_1 \times D_2 \times \dots \times D_n$.

Eine Datenbanktabelle kann daher als n -stellige Relation über der Menge $\mathcal{D} := D_1 \cup \dots \cup D_n$ aufgefasst werden.

Eine relationale Datenbank mit Schema τ kann also als τ -Struktur \mathcal{D} wie folgt geschrieben werden:

- Das Universum $A := \mathcal{D}$ ist die Vereinigung aller Domains.
- für jede Tabelle $R \in \tau$ enthält die Struktur eine Relation $R^{\mathcal{D}}$, die alle Tupel der Tabelle enthält.

Beispiel: Relationale Datenbanken

Der domain aller Einträge sind Zeichenketten. Sei Σ^* die Menge aller Zeichenketten über dem Alphabet $\{a, \dots, z, A, \dots, Z, 0, \dots, 9\}$.

Die Filmdatenbank entspricht folgender Struktur \mathcal{D} über der Signatur

$$\sigma := \{ \text{Actors}, \text{Movies} \}:$$

- Das Universum ist $\mathcal{D} := \Sigma^*$
- Die Relation

$(\text{George Clooney}, 1, \text{6 May 1961}),$

$$\text{Actors}^{\mathcal{D}} := \{ (\text{Scarlett Johansson}, 2, \text{22 November 1984}), \\ (\text{Jeff Daniels}, 3, \text{19 February 1955}) \}$$

- Die Relation

$(\text{Good night ... and good luck}, \text{George Clooney}, 1),$

$$\text{Movies}^{\mathcal{D}} := \{ (\text{Good night ... and good luck}, \text{George Clooney}, 3), \\ (\text{Lost in translation}, \text{Sofia Coppola}, 2) \}$$

Datenbankanfragen

Datenbank als Struktur.

Die Filmdatenbank entspricht der Struktur $\mathcal{D} = (D, \text{Actors}^{\mathcal{D}}, \text{Movies}^{\mathcal{D}})$ mit

$\text{Actors}^{\mathcal{D}} := \{ (\text{George Clooney}, 1, 1.5.1961),$
 $(\text{Scarlett Johansson}, 2, 22.11.1984),$
 $(\text{Jeff Daniels}, 3, 19.2.1955) \}$

$\text{Movies}^{\mathcal{D}} := \{ (\text{Good night ...}, \text{George Clooney}, 1),$
 $(\text{Good night ...}, \text{George Clooney}, 3),$
 $(\text{Lost in translation}, \text{Sofia Coppola}, 2) \}$

Datenbankanfragen als Formeln.

Die Menge der Paare von Filmtiteln und SchauspielerInnen, die in dem Film mitspielen, wird durch folgende Formel definiert:

$$\varphi(F, S) := \exists x_{id} (\exists x_{dat} \text{Actors}(S, x_{id}, x_{dat}) \wedge \exists x_{reg} \text{Movies}(F, x_{reg}, x_{id}))$$

„Gib alle Paare (*Filmtitel, Schausp.*) aus, wobei Filmtitel der Titel eines Films ist, in dem Schausp. mitspielt“

Datenbanken vs. Logik

Datenbanken

Datenbankschema τ

Datenbank \mathcal{D}

SQL-Abfrage $Q(\text{Title})$

```
SELECT Title  
FROM Movies  
WHERE Director='G. Clooney'
```

durch Q definierte View

(materialisiert oder nicht)

Logik

(Relationale) Signatur τ

τ -Struktur \mathcal{A}

Formel $\varphi(x_{\text{title}}) \in \text{FO}[\tau]$

$$\varphi(\mathcal{A}) := \{a \in A : (\mathcal{A}, [x_{\text{title}} / a]) \models \varphi\}$$

(die durch φ in \mathcal{A} definierte Relation)

7.6 Substrukturen und Homomorphismen

Substrukturen und Äquivalenz zwischen Strukturen

Substrukturen

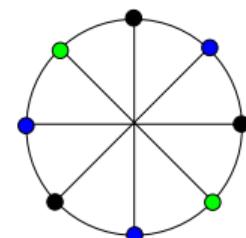
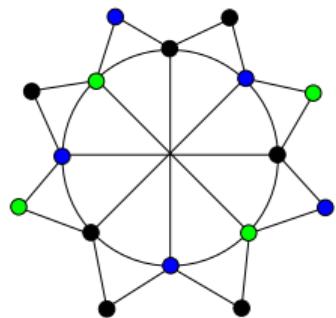
Definition.

Sei τ eine Signatur und seien \mathcal{A}, \mathcal{B} τ -Strukturen.

- \mathcal{A} ist eine **Substruktur** von \mathcal{B} , geschrieben als $\mathcal{A} \subseteq \mathcal{B}$, wenn $A \subseteq B$ und
 - für alle k -stelligen Relationssymbole $R \in \tau$ und alle $\bar{a} \in A^k$ gilt

$$\bar{a} \in R^{\mathcal{A}} \quad \text{gdw.} \quad \bar{a} \in R^{\mathcal{B}}$$
 - für alle k -stelligen Funktionssymbole $f \in \tau$ und alle $\bar{a} \in A^k$ gilt

$$f^{\mathcal{A}}(\bar{a}) = f^{\mathcal{B}}(\bar{a})$$
 - für alle Konstantensymbole $c \in \tau$ gilt $c^{\mathcal{A}} = c^{\mathcal{B}}$.
- Wenn $\mathcal{A} \subseteq \mathcal{B}$, dann ist \mathcal{B} eine **Erweiterung** von \mathcal{A} .



τ -abgeschlossene Mengen

Definition.

Sei τ eine Signatur und \mathcal{B} eine τ -Struktur mit Universum B .

Eine Menge $A \subseteq B$ heißt τ -abgeschlossen in \mathcal{B} , wenn

1. $c^{\mathcal{B}} \in A$ für alle Konstantensymbole $c \in \tau$ und
2. wenn $f \in \tau$ ein k -stelliges Funktionssymbol ist und $\bar{a} \in A^k$ ein k -Tupel von Elementen, so ist $f^{\mathcal{B}}(\bar{a}) \in A$.

Beispiele. Sei $\mathcal{Z} := (\mathbb{Z}, <^{\mathcal{Z}}, +^{\mathcal{Z}})$, wobei $<^{\mathcal{Z}}$ und $+^{\mathcal{Z}}$ die natürliche Ordnung und Addition auf \mathbb{Z} ist.

Frage. Ist die Menge $N := \{0, 1, \dots\}$ τ -abgeschlossen? Ja.

Antwort. Ja, denn wenn $a, b \in N$, dann ist auch $a +^{\mathcal{Z}} b \in N$.

Definition.

$A \subseteq B$, wenn $A \subseteq B$ und

- für alle $R \in \tau$ und $\bar{a} \in A^k$,
 $\bar{a} \in R^A$ gdw. $\bar{a} \in R^B$
- für alle $f \in \tau$ und $\bar{a} \in A^k$,
 $f^A(\bar{a}) = f^B(\bar{a})$
- für alle $c \in \tau$, $c^A = c^B$.

τ -abgeschlossene Mengen

Definition.

Sei τ eine Signatur und \mathcal{B} eine τ -Struktur mit Universum B .

Eine Menge $A \subseteq B$ heißt τ -abgeschlossen in \mathcal{B} , wenn

1. $c^{\mathcal{B}} \in A$ für alle Konstantensymbole $c \in \tau$ und
2. wenn $f \in \tau$ ein k -stelliges Funktionssymbol ist und $\bar{a} \in A^k$ ein k -Tupel von Elementen, so ist $f^{\mathcal{B}}(\bar{a}) \in A$.

Beispiele. Sei $\mathcal{Z} := (\mathbb{Z}, <^{\mathcal{Z}}, +^{\mathcal{Z}})$, wobei $<^{\mathcal{Z}}$ und $+^{\mathcal{Z}}$ die natürliche Ordnung und Addition auf \mathbb{Z} ist.

Frage. Ist die Menge $N := \{0, 1, \dots\}$ τ -abgeschlossen? Ja.

Frage. Ist die Menge $M := \{-1, 0, 1, 2, \dots\}$ τ -abgeschlossen? Nein.

Antwort. Nein, denn $-1 \in M$ aber $-1 +^{\mathcal{Z}} -1 = -2 \notin M$.

Definition.

$A \subseteq B$, wenn $A \subseteq B$ und

- für alle $R \in \tau$ und $\bar{a} \in A^k$,
 $\bar{a} \in R^A$ gdw. $\bar{a} \in R^B$
- für alle $f \in \tau$ und $\bar{a} \in A^k$,
 $f^A(\bar{a}) = f^B(\bar{a})$
- für alle $c \in \tau$, $c^A = c^B$.

τ -abgeschlossene Mengen

Definition.

Sei τ eine Signatur und \mathcal{B} eine τ -Struktur mit Universum B .

Eine Menge $A \subseteq B$ heißt τ -abgeschlossen in \mathcal{B} , wenn

1. $c^{\mathcal{B}} \in A$ für alle Konstantensymbole $c \in \tau$ und
2. wenn $f \in \tau$ ein k -stelliges Funktionssymbol ist und $\bar{a} \in A^k$ ein k -Tupel von Elementen, so ist $f^{\mathcal{B}}(\bar{a}) \in A$.

Beispiele. Sei $\mathcal{Z} := (\mathbb{Z}, <^{\mathcal{Z}}, +^{\mathcal{Z}})$, wobei $<^{\mathcal{Z}}$ und $+^{\mathcal{Z}}$ die natürliche Ordnung und Addition auf \mathbb{Z} ist.

Frage. Ist die Menge $N := \{0, 1, \dots\}$ τ -abgeschlossen? Ja.

Frage. Ist die Menge $M := \{-1, 0, 1, 2, \dots\}$ τ -abgeschlossen? Nein.

Frage. Ist die Menge $G := \{\dots, -4, -2, 0, 2, 4, \dots\}$ τ -abgeschlossen?

Antwort. Ja, denn wenn $a, b \in G$, dann sind a, b gerade Zahlen und die Summe zweier gerader Zahlen ist gerade.

Definition.

$A \subseteq B$, wenn $A \subseteq B$ und

- für alle $R \in \tau$ und $\bar{a} \in A^k$,
 $\bar{a} \in R^A$ gdw. $\bar{a} \in R^B$
- für alle $f \in \tau$ und $\bar{a} \in A^k$,
 $f^A(\bar{a}) = f^B(\bar{a})$
- für alle $c \in \tau$, $c^A = c^B$.

τ -abgeschlossene Mengen

Definition.

Sei τ eine Signatur und \mathcal{B} eine τ -Struktur mit Universum B .

Eine Menge $A \subseteq B$ heißt τ -abgeschlossen in \mathcal{B} , wenn

1. $c^{\mathcal{B}} \in A$ für alle Konstantensymbole $c \in \tau$ und
2. wenn $f \in \tau$ ein k -stelliges Funktionssymbol ist und $\bar{a} \in A^k$ ein k -Tupel von Elementen, so ist $f^{\mathcal{B}}(\bar{a}) \in A$.

Lemma. Wenn $A \subseteq B$, dann ist A τ -abgeschlossen.

Beweis. Per Definition gilt:

- $c^A = c^{\mathcal{B}}$, also $c^{\mathcal{B}} \in A$.
- $f^A(\bar{a}) = f^{\mathcal{B}}(\bar{a})$, also ist $f^{\mathcal{B}}(\bar{a}) \in A$, für alle $\bar{a} \in A^k$. □

Definition.

$A \subseteq B$, wenn $A \subseteq B$ und

für alle $R \in \tau$ und $\bar{a} \in A^k$,
 $\bar{a} \in R^A$ gdw. $\bar{a} \in R^B$,

für alle $f \in \tau$ und $\bar{a} \in A^k$,
 $f^A(\bar{a}) = f^B(\bar{a})$

für alle $c \in \tau$, $c^A = c^{\mathcal{B}}$.

τ -abgeschlossene Mengen

Definition.

Sei τ eine Signatur und \mathcal{B} eine τ -Struktur mit Universum B .

Eine Menge $A \subseteq B$ heißt τ -abgeschlossen in \mathcal{B} , wenn

1. $c^{\mathcal{B}} \in A$ für alle Konstantensymbole $c \in \tau$ und
2. wenn $f \in \tau$ ein k -stelliges Funktionssymbol ist und $\bar{a} \in A^k$ ein k -Tupel von Elementen, so ist $f^{\mathcal{B}}(\bar{a}) \in A$.

Lemma. Wenn $A \subseteq B$, dann ist A τ -abgeschlossen.

Umgekehrt. Für jede τ -abgeschlossene Menge $A \subseteq B$ existiert genau eine Substruktur $\mathcal{A} \subseteq \mathcal{B}$ mit Universum A .

Definition.

$A \subseteq B$, wenn $A \subseteq B$ und

- für alle $R \in \tau$ und $\bar{a} \in A^k$,
 $\bar{a} \in R^A$ gdw. $\bar{a} \in R^B$,
- für alle $f \in \tau$ und $\bar{a} \in A^k$,
 $f^A(\bar{a}) = f^B(\bar{a})$
- für alle $c \in \tau$, $c^A = c^B$.

τ -abgeschlossene Mengen

Definition.

Sei τ eine Signatur und \mathcal{B} eine τ -Struktur mit Universum B .

Eine Menge $A \subseteq B$ heißt τ -abgeschlossen in \mathcal{B} , wenn

1. $c^{\mathcal{B}} \in A$ für alle Konstantensymbole $c \in \tau$ und
2. wenn $f \in \tau$ ein k -stelliges Funktionssymbol ist und $\bar{a} \in A^k$ ein k -Tupel von Elementen, so ist $f^{\mathcal{B}}(\bar{a}) \in A$.

Lemma. Wenn $A \subseteq B$, dann ist A τ -abgeschlossen.

Umgekehrt. Für jede τ -abgeschlossene Menge $A \subseteq B$ existiert genau eine Substruktur $\mathcal{A} \subseteq \mathcal{B}$ mit Universum A .

Definition. Der τ -Abschluss einer Menge $A \subseteq B$ ist die kleinste τ -abgeschlossene Menge $\text{cl}_{\tau}(A)$ mit $A \subseteq \text{cl}_{\tau}(A)$.

Für $A \subseteq B$ definieren wir die durch A induzierte Substruktur von \mathcal{B} als die Substruktur von \mathcal{B} mit Universum $\text{cl}_{\tau}(A)$.

Definition.

$A \subseteq B$, wenn $A \subseteq B$ und

- für alle $R \in \tau$ und $\bar{a} \in A^k$,
 $\bar{a} \in R^A$ gdw. $\bar{a} \in R^B$
- für alle $f \in \tau$ und $\bar{a} \in A^k$,
 $f^A(\bar{a}) = f^B(\bar{a})$
- für alle $c \in \tau$, $c^A = c^B$.

τ -abgeschlossene Mengen

Definition.

Sei τ eine Signatur und \mathcal{B} eine τ -Struktur mit Universum B .

Eine Menge $A \subseteq B$ heißt τ -abgeschlossen in \mathcal{B} , wenn

1. $c^{\mathcal{B}} \in A$ für alle Konstantensymbole $c \in \tau$ und
2. wenn $f \in \tau$ ein k -stelliges Funktionssymbol ist und $\bar{a} \in A^k$ ein k -Tupel von Elementen, so ist $f^{\mathcal{B}}(\bar{a}) \in A$.

Lemma. Wenn $A \subseteq B$, dann ist A τ -abgeschlossen.

Umgekehrt. Für jede τ -abgeschlossene Menge $A \subseteq B$ existiert genau eine Substruktur $\mathcal{A} \subseteq \mathcal{B}$ mit Universum A .

Definition. Der τ -Abschluss einer Menge $A \subseteq B$ ist die kleinste τ -abgeschlossene Menge $\text{cl}_{\tau}(A)$ mit $A \subseteq \text{cl}_{\tau}(A)$.

Für $A \subseteq B$ definieren wir die durch A induzierte Substruktur von \mathcal{B} als die Substruktur von \mathcal{B} mit Universum $\text{cl}_{\tau}(A)$.

Definition.

$A \subseteq B$, wenn $A \subseteq B$ und

- für alle $R \in \tau$ und $\bar{a} \in A^k$,
 $\bar{a} \in R^A$ gdw. $\bar{a} \in R^B$
- für alle $f \in \tau$ und $\bar{a} \in A^k$,
 $f^A(\bar{a}) = f^B(\bar{a})$
- für alle $c \in \tau$, $c^A = c^B$.

Beispiel. Sei $\mathcal{Z} := (\mathbb{Z}, <^{\mathcal{Z}}, +^{\mathcal{Z}})$, wobei $<^{\mathcal{Z}}$, $+^{\mathcal{Z}}$ die natürliche Ordnung und Addition auf \mathbb{Z} sind.

Die von $\{0, 1\}$ induzierte Substruktur von \mathcal{Z} ist $\mathcal{N} := (\mathbb{N}, <^{\mathcal{N}}, +^{\mathcal{N}})$.

Denn: \mathcal{N} muss 0 und 1 enthalten, und wegen des Abschlusses unter $+^{\mathcal{Z}}$ auch $1+1=2$ und daher auch $1+2=3$ etc.

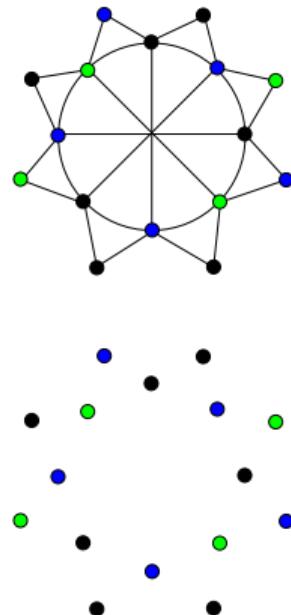
Expansionen und Redukte

Definition. Sei $\sigma \subseteq \tau$ eine Signatur und sei \mathcal{B} eine τ -Struktur.

Das σ -Redukt $\mathcal{B}|_{\sigma}$ von \mathcal{B} ist definiert als die σ -Struktur $\mathcal{B}|_{\sigma}$ mit

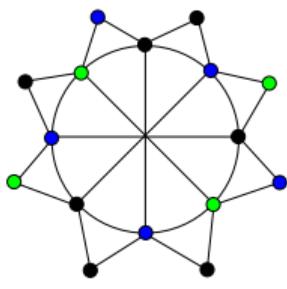
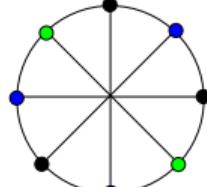
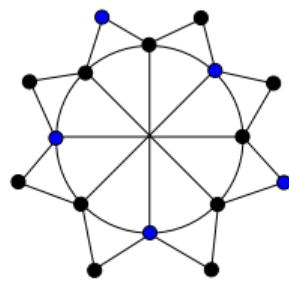
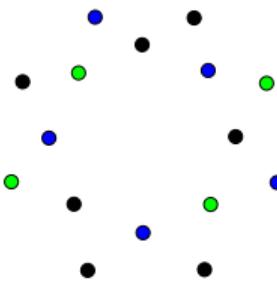
- Universum \mathcal{B} und
- $S^{\mathcal{B}|_{\sigma}} = S^{\mathcal{B}}$ für jedes (Relations-, Funktions-, Konstanten-) Symbol $S \in \sigma$.

\mathcal{B} heißt Expansion von $\mathcal{B}|_{\sigma}$.



Beispiel

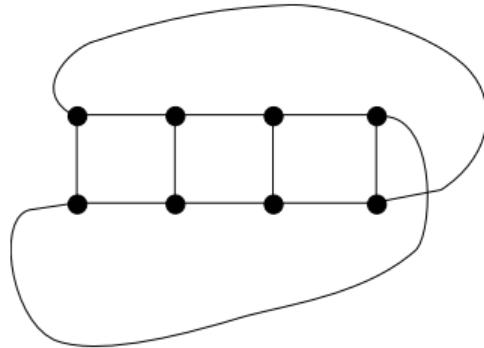
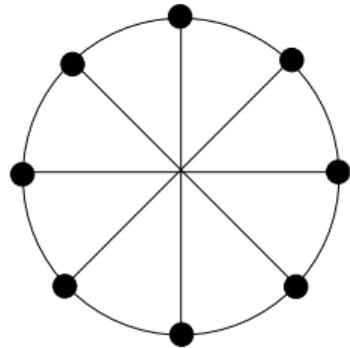
Beispiel. Eine $\sigma := \{E, \text{Blue}, \text{Green}\}$ -Struktur, Substruktur und Redukte.

 \mathcal{G}  $\mathcal{H} \subseteq \mathcal{G}$
Substruktur $\mathcal{G}|_{\{E, \text{Blue}\}}$
Redukt $\mathcal{G}|_{\{\text{Blue}, \text{Green}\}}$
Redukt

Homomorphismen

Wann sind zwei Strukturen gleich?

Frage. Sind die folgenden zwei Graphen verschieden?



Mögliche Antworten.

Ja wenn wir daran interessiert sind, wie sie gezeichnet sind.

Nein wenn wir uns nur für ihre Knoten und Verbindungen dazwischen interessieren.

Homomorphismen

Definition. Seien \mathcal{A}, \mathcal{B} zwei σ -Strukturen.

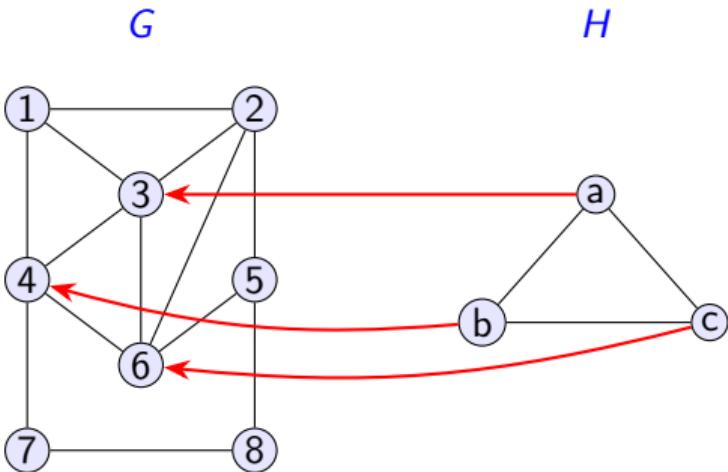
Ein **Homomorphismus** von \mathcal{A} in \mathcal{B} ist eine Funktion $h : A \rightarrow B$, so dass

- für alle k -stelligen Relationssymbole $R \in \sigma$ und $\bar{a} := a_1, \dots, a_k \in A^k$ gilt
wenn $\bar{a} \in R^{\mathcal{A}}$ dann auch $(h(a_1), \dots, h(a_k)) \in R^{\mathcal{B}}$.
- für alle k -stelligen Funktionssymbole $f \in \sigma$ und $\bar{a} := a_1, \dots, a_k \in A^k$ gilt
$$h(f^{\mathcal{A}}(\bar{a})) = f^{\mathcal{B}}(h(a_1), \dots, h(a_k)).$$
- für alle Konstantensymbole $c \in \sigma$ gilt $h(c^{\mathcal{A}}) = c^{\mathcal{B}}$.

Notation. $h : \mathcal{A} \rightarrow_{hom} \mathcal{B} : h$ ist ein Homomorphismus von \mathcal{A} nach \mathcal{B} .

Beispiel

Wir betrachten die folgenden Graphen G und H .



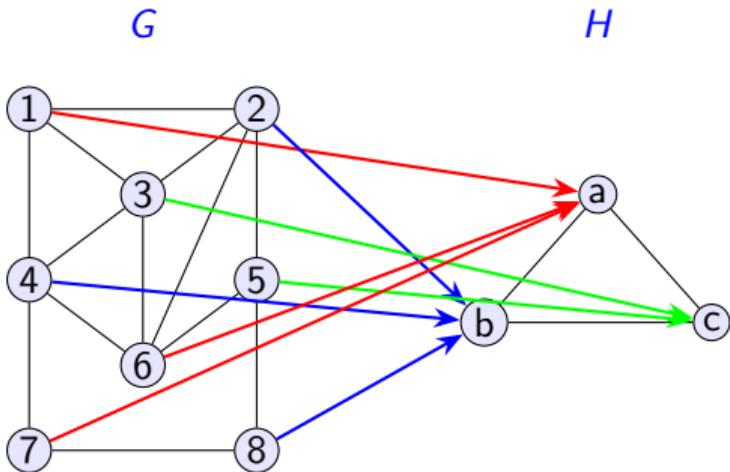
Homomorphismus $h : \mathcal{A} \rightarrow_{hom} \mathcal{B}$.

Funktion $h : A \rightarrow B$, so dass

1. für alle $R \in \sigma$ und $a_1, \dots, a_k \in A^k$:
wenn $\bar{a} \in R^A$ dann
 $(h(a_1), \dots, h(a_k)) \in R^B$.
2. für alle $f \in \sigma$ und $a_1, \dots, a_k \in A^k$:
 $h(f^A(\bar{a})) = f^B(h(a_1), \dots, h(a_k))$.
3. für alle $c \in \sigma$ gilt $h(c^A) = c^B$.

Beispiel

Wir betrachten die folgenden Graphen G und H .



Es gilt $G \rightarrow_{hom} H$ und $H \rightarrow_{hom} G$.

Homomorphismus $h : \mathcal{A} \rightarrow_{hom} \mathcal{B}$.

Funktion $h : A \rightarrow B$, so dass

1. für alle $R \in \sigma$ und $a_1, \dots, a_k \in A^k$:
wenn $\bar{a} \in R^A$ dann
 $(h(a_1), \dots, h(a_k)) \in R^B$.
2. für alle $f \in \sigma$ und $a_1, \dots, a_k \in A^k$:
 $h(f^A(\bar{a})) = f^B(h(a_1), \dots, h(a_k))$.
3. für alle $c \in \sigma$ gilt $h(c^A) = c^B$.

Isomorphismen

Definition. Seien \mathcal{A}, \mathcal{B} zwei σ -Strukturen.

Ein **Isomorphismus** von \mathcal{A} in \mathcal{B} ist eine Funktion $I : \mathcal{A} \rightarrow \mathcal{B}$, so dass

- I eine Bijektion zwischen \mathcal{A} und \mathcal{B} ist
- für alle k -stelligen Relationssymbole $R \in \sigma$ und alle $\bar{a} := a_1, \dots, a_k \in \mathcal{A}^k$ gilt

$$\bar{a} \in R^{\mathcal{A}} \quad \text{gdw.} \quad (I(a_1), \dots, I(a_k)) \in R^{\mathcal{B}}.$$

- für alle k -stelligen Funktionssymbole $f \in \sigma$ und alle $\bar{a} := a_1, \dots, a_k \in \mathcal{A}^k$ gilt

$$I(f^{\mathcal{A}}(\bar{a})) = f^{\mathcal{B}}(I(a_1), \dots, I(a_k)).$$

- für alle Konstantensymbole $c \in \sigma$ gilt $I(c^{\mathcal{A}}) = c^{\mathcal{B}}$.

Notation. $I : \mathcal{A} \cong \mathcal{B}$: I ist ein Isomorphismus von \mathcal{A} nach \mathcal{B} .

Homomorphismus $h : \mathcal{A} \rightarrow_{hom} \mathcal{B}$.

Funktion $h : \mathcal{A} \rightarrow \mathcal{B}$, so dass

1. für alle $R \in \sigma$ und $a_1, \dots, a_k \in \mathcal{A}^k$:
wenn $\bar{a} \in R^{\mathcal{A}}$ dann
 $(h(a_1), \dots, h(a_k)) \in R^{\mathcal{B}}$.
2. für alle $f \in \sigma$ und $a_1, \dots, a_k \in \mathcal{A}^k$:
 $h(f^{\mathcal{A}}(\bar{a})) = f^{\mathcal{B}}(h(a_1), \dots, h(a_k))$.
3. für alle $c \in \sigma$ gilt $h(c^{\mathcal{A}}) = c^{\mathcal{B}}$.

Iso- und Homomorphismen

Definition. Sei σ eine Signatur.

1. Zwei σ -Strukturen \mathcal{A}, \mathcal{B} sind **isomorph**, geschrieben $\mathcal{A} \cong \mathcal{B}$, wenn es einen Isomorphismus zwischen \mathcal{A} und \mathcal{B} gibt.
2. Zwei σ -Strukturen \mathcal{A}, \mathcal{B} sind **homomorph**, geschrieben $\mathcal{A} \rightarrow_{\text{hom}} \mathcal{B}$, wenn es einen Homomorphismus von \mathcal{A} nach \mathcal{B} gibt.

Beispiele.

- Wenn A, B endliche Mengen der gleichen Kardinalität sind, dann sind die \emptyset -Strukturen $(A, \emptyset) \cong (B, \emptyset)$.
- Wenn A, B endliche Mengen gleicher Kardinalität und $<^A, <^B$ lineare Ordnungen auf A, B sind, dann $(A, <^A) \cong (B, <^B)$.
Aber: $(\mathbb{Z}, <) \not\cong (\mathbb{N}, <)$

Homomorphismus $h : \mathcal{A} \rightarrow_{\text{hom}} \mathcal{B}$.

Funktion $h : A \rightarrow B$, so dass

1. für alle $R \in \sigma$ und $a_1, \dots, a_k \in A^k$:
wenn $\bar{a} \in R^A$ dann $(h(a_1), \dots, h(a_k)) \in R^B$.
2. für alle $f \in \sigma$ und $a_1, \dots, a_k \in A^k$:
 $h(f^A(\bar{a})) = f^B(h(a_1), \dots, h(a_k))$.
3. für alle $c \in \sigma$ gilt $h(c^A) = c^B$.

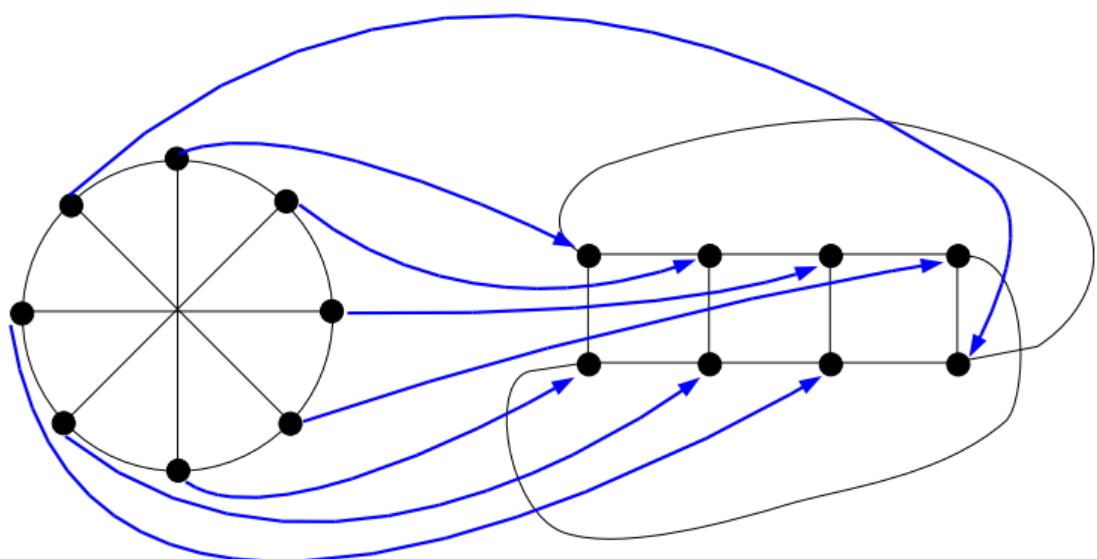
Isomorphismus $h : \mathcal{A} \cong \mathcal{B}$. Bijektion

$I : A \rightarrow B$, so dass

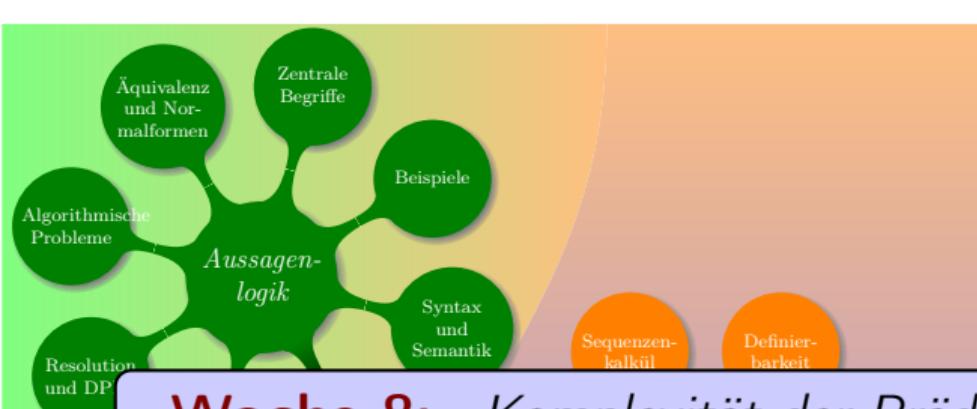
1. für alle $R \in \sigma$ und $a_1, \dots, a_k \in A^k$:
 $\bar{a} \in R^A$ gdw $(I(a_1), \dots, I(a_k)) \in R^B$.
2. für alle $f \in \sigma$ und $a_1, \dots, a_k \in A^k$:
 $I(f^A(\bar{a})) = f^B(I(a_1), \dots, I(a_k))$.
3. für alle $c \in \sigma$ gilt $I(c^A) = c^B$.

Beispiel

Frage. Sind die beiden folgenden Graphen gleich?

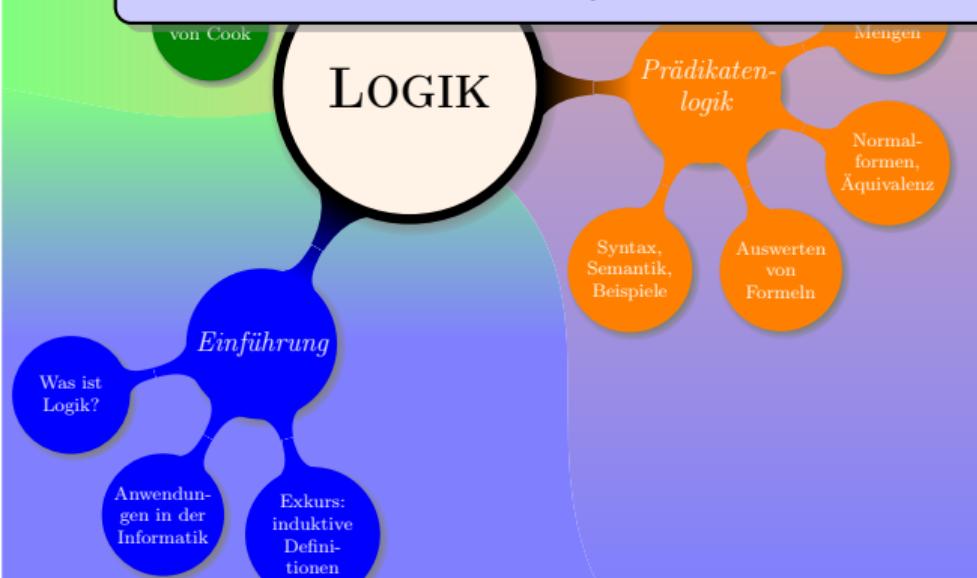


- Isomorphismus $h : \mathcal{A} \cong \mathcal{B}$.** Bijektion
 $I : \mathcal{A} \rightarrow \mathcal{B}$, so dass
1. für alle $R \in \sigma$ und $a_1, \dots, a_k \in A^k$:
 $\bar{a} \in R^{\mathcal{A}}$ gdw $(I(a_1), \dots, I(a_k)) \in R^{\mathcal{B}}$.
 2. für alle $f \in \sigma$ und $a_1, \dots, a_k \in A^k$:
 $I(f^{\mathcal{A}}(\bar{a})) = f^{\mathcal{B}}(I(a_1), \dots, I(a_k))$.
 3. für alle $c \in \sigma$ gilt $I(c^{\mathcal{A}}) = c^{\mathcal{B}}$.



Nov.	17	18	19	20	21	22	23	<i>Einführung</i>
	24	25	26	27	28	29	30	<i>Aussagenlogik</i>
	31	1	2	3	4	5	6	<i>Normalformen</i>
	7	8	9	10	11	12	13	<i>Resolution</i>
Dez.	14	15	16	17	18	19	20	<i>Kompaktheit</i>
	21	22	23	24	25	26	27	<i>DPLL, Satz von Cook</i>

Woche 8: Komplexität der Prädikatenlogik



2	3	4	5	6	7	8	9	10	11	<i>Strukturen und FO</i>
12	13	14	15	16	17	18	19	20	21	<i>Prädikatenlogik</i>
Jan.	20	21	22	23	24	25	26	27	28	<i>Komplexität von FO</i>
	29	30	31	1	2	3	4	5	6	<i>Weihnachten</i>
	7	8	9	10	11	12	13	14	15	<i>Neujahr</i>
	16	17	18	19	20	21	22	23	24	<i>Normalformen</i>
Feb.	23	24	25	26	27	28	29	30	31	<i>Definierbarkeit</i>
	1	2	3	4	5	6	7	8	9	<i>EF-Spiele</i>
	10	11	12	13	14	15	16	17	18	<i>EF-Spiele</i>
	19	20	21	22	23	24	25	26	27	<i>Sequenzenkalkül AL</i>
	28	29	30	31	1	2	3	4	5	<i>Sequenzenkalkül FO</i>
	6	7	8	9	10	11	12	13	14	<i>Ausblick</i>
	15	16	17	18	19	20	21	22	23	

8.1 Wichtige Begriffe der Prädikatenlogik

Formeln mit freien Variablen vs. Sätze

Formeln mit freien Variablen vs. Sätze

Formeln mit freien Variablen

$$\varphi_1(x) := \forall y \forall z (y * z = x \rightarrow (y = 1 \vee z = 1))$$

$$\varphi_4(x, y) := \exists z (x * x = y + z)$$

Sätze

$$\varphi_2 := \forall y \exists x (y < x \wedge \varphi_1(x))$$

$$\varphi_3 := \forall x \forall y \forall z ((x < y \wedge y < z) \rightarrow x < z)$$

Formeln $\varphi(x)$.

Eine Formel $\varphi(x)$ sagt etwas über ein Element innerhalb einer Struktur aus. D.h. $\varphi(x)$ beschreibt eine Eigenschaft eines Elements.

Wenn $\beta(x) = a$ eine Belegung von x ist, dann gilt $(\mathcal{A}, \beta) \models \varphi(x)$, wenn a die Eigenschaft φ hat.

Sätze ψ .

Ein Satz ψ sagt etwas über die Struktur insgesamt aus.

Ohne freie Variablen brauchen wir keine Belegung β .

D.h. $\mathcal{A} \models \psi$, wenn die Struktur die Eigenschaft ψ hat.

Modellklassen und die Relation $\varphi(\mathcal{A})$

Formeln $\varphi(x)$. Eine Formel $\varphi(x)$ sagt etwas über ein Element innerhalb einer Struktur aus.

Sätze ψ . Ein Satz ψ sagt etwas über die Struktur insgesamt aus.

Oft interessieren wir uns für die „Menge“ aller Objekte, die eine Formel bzw. einen Satz erfüllen.

Formeln. Bei Formeln $\varphi(x)$ ist diese „Menge“ die Menge $\varphi(\mathcal{A})$ der Elemente einer Struktur \mathcal{A} , die die Formel erfüllen.

Sätze. Bei einem Satz ψ ist diese „Menge“ die Klasse aller Strukturen, in denen der Satz gilt.

Die Relation $\varphi(\mathcal{A})$

Definition. Sei \mathcal{A} eine σ -Struktur und $\varphi(x_1, \dots, x_k) \in \text{FO}[\sigma]$.

Wir definieren

$$\varphi(\mathcal{A}) := \{(a_1, \dots, a_k) \in A^k : (\mathcal{A}, [x_1/a_1, \dots, x_k/a_k]) \models \varphi\}.$$

Hinweis. Die Relation $\varphi(\mathcal{A})$ hängt nicht nur von \mathcal{A} sondern auch von der Sequenz $(x_1, \dots, x_k) \in \text{Var}^k$ ab.

Wir müssen daher diese Sequenz jeweils angeben, bevor wir die Notation benutzen können.

Vergleiche mit Methoden in Java.

```
Boolean phi(int x1, ..., int xk)
```

Mit x_1, \dots, x_k wird eine Ordnung der Parameter festgelegt.

Wir können dann `Boolean b = phi(3, 5, ..., 17);` benutzen.

Modellklassen und definierbare Relationen

Definition (definierbare Relationen).

Sei \mathcal{A} eine σ -Struktur und $\varphi(x_1, \dots, x_k) \in \text{FO}[\sigma]$. Wir definieren

$$\varphi(\mathcal{A}) := \{(a_1, \dots, a_k) \in A^k : (\mathcal{A}, [x_1/a_1, \dots, x_k/a_k]) \models \varphi\},$$

und sagen, dass φ die Relation $\varphi(\mathcal{A})$ in \mathcal{A} definiert.

Umgekehrt nennen wir eine Relation $R \subseteq A^k$ FO -definierbar in \mathcal{A} , wenn es eine Formel $\varphi(x_1, \dots, x_k) \in \text{FO}$ gibt, so dass $\varphi(\mathcal{A}) = R$.

Definition (Modellklassen).

Sei σ eine Signatur und $\Phi \subseteq \text{FO}[\sigma]$ eine Menge von σ -Sätzen.

Die **Modellklasse** von Φ , geschrieben $\text{Mod}(\Phi)$, ist die Klasse aller σ -Strukturen \mathcal{A} mit $\mathcal{A} \models \Phi$.

Falls $\Phi := \{\varphi\}$ nur einen Satz enthält, schreiben wir kurz $\text{Mod}(\varphi)$.

Erfüllbarkeit und Allgemeingültigkeit

Definition. Sei $\varphi \in \text{FO}[\sigma]$ eine Formel, $\Phi \subseteq \text{FO}[\sigma]$ eine Formelmenge und \mathcal{I} eine σ -Interpretation.

1. \mathcal{I} erfüllt φ , wenn \mathcal{I} zu φ passt und $\llbracket \varphi \rrbracket^{\mathcal{I}} = 1$.

Wir sagen auch: \mathcal{I} ist ein **Modell** von φ und schreiben $\mathcal{I} \models \varphi$.

2. \mathcal{I} passt zu Φ , wenn sie zu allen $\psi \in \Phi$ passt. \mathcal{I} erfüllt Φ , wenn \mathcal{I} zu Φ passt und alle $\psi \in \Phi$ erfüllt.

Wir sagen auch: \mathcal{I} ist ein **Modell** von Φ und schreiben $\mathcal{I} \models \Phi$.

3. Φ ist **erfüllbar**, wenn es ein Modell hat. Ansonsten ist Φ **unerfüllbar**.

4. Φ ist **allgemeingültig**, oder eine **Tautologie**, wenn alle zu Φ passenden Interpretationen Φ erfüllen.

5. φ ist **erfüllbar/unerfüllbar/allgemeingültig**, wenn $\{\varphi\}$ erfüllbar/unerfüllbar/allgemeingültig ist.

Beispiel zu Erfüllbarkeit

Erinnerung. Satz $\varphi_{ord} \in \text{FO}[\{<\}]$ mit $\mathcal{A} \models \varphi_{ord}$ gdw. $<^{\mathcal{A}}$ ist lineare Ordnung.

Beispiele. Sei $\sigma := \{<\}$.

1. $\varphi := \varphi_{ord} \wedge \forall x \exists y \ y < x$ ist erfüllbar, z.B. durch $(\mathbb{Z}, <)$,

aber nicht allgemeingültig, da $(\mathbb{N}, <) \not\models \varphi$

2. $\psi := \varphi_{ord} \rightarrow \forall x \forall y \neg(y < x \wedge x < y)$ ist allgemeingültig.

φ_{ord} gilt nur in $\{<\}$ -Strukturen \mathcal{A} , in denen $<^{\mathcal{A}}$ eine strikte lineare Ordnung ist.

Wenn $<^{\mathcal{A}}$ aber eine strikte Ordnung ist, dann ist $<^{\mathcal{A}}$ auch immer anti-symmetrisch, d.h. es kann keine Elemente a, b geben, so dass $a < b$ und $b < a$.

Logische Folgerung

Definition. Sei σ eine Signatur, $\Phi \subseteq \text{FO}[\sigma]$ und $\psi \in \text{FO}[\sigma]$.

ψ ist eine **Folgerung** von Φ , geschrieben $\Phi \models \psi$, wenn für jede zu Φ und ψ passende σ -Interpretation \mathcal{I} gilt:

$$\mathcal{I} \models \Phi \implies \mathcal{I} \models \psi.$$

Notation. Statt $\emptyset \models \psi$ schreiben wir $\models \psi$.

Beispiel zu logischer Folgerung

Erinnerung. Satz $\varphi_{ord} \in \text{FO}[\{<\}]$ mit $\mathcal{A} \models \varphi_{ord}$ gdw. $<^{\mathcal{A}}$ ist lineare Ordnung.

Für einen Satz φ gilt also:

$$\varphi_{ord} \models \varphi \iff \varphi \text{ gilt in allen linearen Ordnungen.}$$

Es gilt also z.B.

$$\varphi_{ord} \models \forall x \forall y \exists z (z \leq x \wedge z \leq y)$$

wobei $t \leq t'$ für die Formel $(t < t' \vee t = t')$ steht.

Eigenschaften der Folgerungsbeziehung

Lemma.

1. Für alle $\Phi \subseteq \text{FO}[\sigma]$ und $\psi \in \text{FO}[\sigma]$:

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

2. Für alle $\psi \in \text{FO}[\sigma]$:

$$\models \psi \iff (\psi \text{ ist eine Tautologie})$$

Äquivalenz zwischen Formeln

Definition. Sei σ eine Signatur.

Zwei σ -Formeln $\varphi, \psi \in \text{FO}[\sigma]$ sind äquivalent, geschrieben $\varphi \equiv \psi$, wenn für alle σ -Interpretationen \mathcal{I} passend zu φ und ψ :

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

Bemerkung. Nach Definition gilt für alle Formeln $\varphi, \psi \in \text{FO}[\sigma]$

$$\varphi \equiv \psi \iff (\varphi \leftrightarrow \psi \text{ ist allgemeingültig})$$

Zusammenfassung

1. Formeln mit freien Variablen vs. Sätze
2. Definierbare Relationen $\varphi(\mathcal{A})$.
3. Modellklassen $\text{Mod}(\varphi)$.
4. Erfüllbarkeit und Allgemeingültigkeit
5. Logische Folgerung
6. Äquivalenz

8.2 Komplexität der Prädikatenlogik

Auswerten prädikatenlogischer Formeln

Das Auswerten prädikatenlogischer Formeln ist viel schwerer als das Auswerten aussagenlogischer Formeln.

Top-Down Auswertung. Auswerten der Formel von „außen“ nach „innen“.

D.h. beginnend bei den äußersten Quantoren $\exists x \dots / \forall x \dots$ testen wir jede mögliche Belegung der Variablen durch.

Bottom-up Auswertung. Auswerten der Formel von „innen“ nach „außen“.

Beginnend bei den atomaren Formeln $\varphi(\bar{x}) := R(x_1, \dots, x_r)$ berechnen wir alle erfüllenden Variablenbelegungen, d.h. $\varphi(\mathcal{A})$.

Top-Down Auswertung prädikatenlogischer Formeln

$\text{MC}(\mathcal{A}, \beta, \varphi)$.

Eingabe: endliche σ -Struktur \mathcal{A} , $\varphi(x_1, \dots, x_k) \in \text{FO}$
Belegung β der freien Variablen von φ .

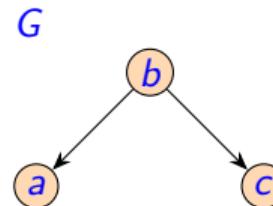
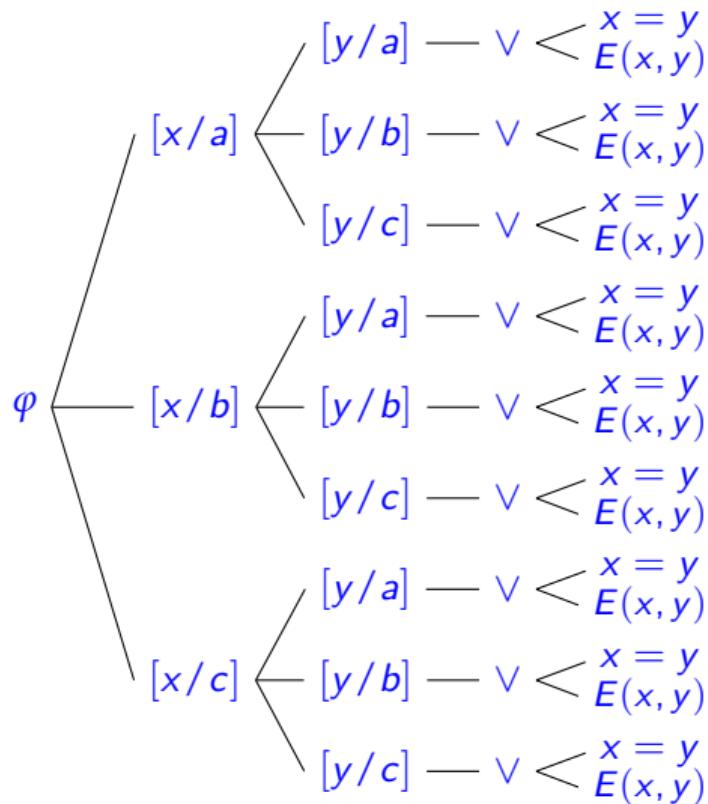
Ausgabe: 1 wenn $(\mathcal{A}, \beta) \models \varphi$, 0 sonst.

Algorithmus. Fallunterscheidung anhand des Formelaufbaus.

- $\varphi = \exists x_i \psi(x_i)$. **for all** $a \in A$:
 if $\text{MC}(\mathcal{A}, \beta[x_i/a], \psi) = 1$ **then return** 1.
 return 0.
- $\varphi = \forall x_i \psi(x_i)$. **for all** $a \in A$:
 if $\text{MC}(\mathcal{A}, \beta[x_i/a], \psi) = 0$ **then return** 0.
 return 1.
- $\varphi = (\varphi_1 \vee \varphi_2)$. **return** $\max\{\text{MC}(\mathcal{A}, \beta, \varphi_1), \text{MC}(\mathcal{A}, \beta, \varphi_2)\}$
- $\varphi = R(t_1, \dots, t_k)$. für ein k -stelliges Relationssymbol $R \in \sigma$.
 Berechne $a_1 := t_1, \dots, a_k := t_k$ in \mathcal{A}
 if $(a_1, \dots, a_k) \in R^{\mathcal{A}}$ **then return** 1 **else return** 0.

(Weitere Fälle analog)

Beispiel: Top-Down Auswertung



$$\varphi := \exists x \forall y (x = y \vee E(x, y)).$$

Bottom-Up Auswertung prädikatenlogischer Formeln

$\text{MC2}(\mathcal{A}, \varphi)$.

Eingabe: endliche σ -Struktur \mathcal{A} , $\varphi(x_1, \dots, x_k) \in \text{FO}$

Ausgabe: $\varphi(\mathcal{A}) = \{(a_1, \dots, a_k) \in A^k : (\mathcal{A}, [x_1/a_1, \dots, x_k/a_k]) \models \varphi\}$.

Hinweis. Hier auch Terme betrachten.

Algorithmus. Fallunterscheidung anhand des Formelaufbaus.

- $\varphi = R(x_1, \dots, x_k)$. für ein k -stelliges Relationssymbol $R \in \sigma$.

Hinweis.

$\bar{x} := \text{frei}(\varphi_1) \vee \text{frei}(\varphi_2)$

Return $R^{\mathcal{A}} = \{(a_1, \dots, a_k) \in A^k : \bar{a} \in R^{\mathcal{A}}\}$.

- $\varphi = (\varphi_1(\bar{x}) \vee \varphi_2(\bar{x}))$. Berechne $R_1 = \text{MC2}(\mathcal{A}, \varphi_1)$ und $R_2 = \text{MC2}(\mathcal{A}, \varphi_2)$

Return $R_1 \cup R_2$

Weitere Fälle analog

- $\varphi = \exists x_i \psi(x_1, \dots, x_i, \dots, x_r)$. Berechne $R = \text{MC2}(\mathcal{A}, \psi)$

return $\{(a_1, \dots, a_{i-1}, a_{i+1}, \dots, a_r) : (a_1, \dots, a_r) \in R\}$.

- $\varphi = \forall x_i \psi(x_1, \dots, x_i, \dots, x_r)$. Berechne $R = \text{MC2}(\mathcal{A}, \psi)$

return $\{(a_1, \dots, a_{i-1}, a_{i+1}, \dots, a_r) : (a_1, \dots, a_{i-1}, a, a_{i+1}, \dots, a_r) \in R \text{ für alle } a \in A\}$.

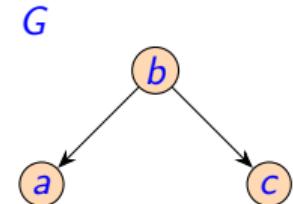
Anmerkung zu $\varphi_1(\bar{x}) \vee \varphi_2(\bar{y})$. Es ist effizienter, nur $\text{MC2}(\mathcal{A}, \varphi_1(\bar{x}))$ und $\text{MC2}(\mathcal{A}, \varphi_2(\bar{y}))$ auszurechnen und die Ergebnisse „sinnvoll“ zusammenzusetzen.

Beispiel: Bottom-Up Auswertung

Bottom-Up Auswertung.

Unterformel	Auswertung
$x = y$	$\{(a, a), (b, b), (c, c)\}$
$E(x, y)$	$\{(b, a), (b, c)\}$
$(x = y \vee E(x, y))$	$\{(a, a), (b, b), (c, c), (b, a), (b, c)\}$
$\forall y(x = y \vee E(x, y))$	$\{(b)\}$
$\exists x \forall y(x = y \vee E(x, y))$	$\{()\}$

$$\varphi := \exists x \forall y (x = y \vee E(x, y)).$$



Auswerten prädikatenlogischer Formeln

Das Auswerten prädikatenlogischer Formeln ist viel schwerer als das Auswerten aussagenlogischer Formeln.

Top-Down Auswertung. Auswerten der Formel von „außen“ nach „innen“.

D.h. beginnend bei den äußersten Quantoren $\exists x \dots / \forall x \dots$ testen wir jede mögliche Belegung der Variablen durch.

Vorteil. Es wird relativ wenig Platz benötigt.

Bottom-up Auswertung. Auswerten der Formel von „innen“ nach „außen“.

Beginnend bei den atomaren Formeln $\varphi(\bar{x}) := R(x_1, \dots, x_r)$ berechnen wir alle erfüllenden Variablenbelegungen, d.h. $\varphi(\mathcal{A})$.

Vorteil. Wir sparen uns die vielen rekursiven Aufrufe, verbrauchen aber eventuell sehr viel Platz.

Komplexität des Auswertungsproblems

Laufzeitabschätzung des top-down Algorithmus'.

Sei \mathcal{A} eine Struktur mit Universum A und φ eine Formel der Länge $|\varphi|$.

Der Algorithmus durchläuft für jeden Quantor in φ alle Elemente in A .

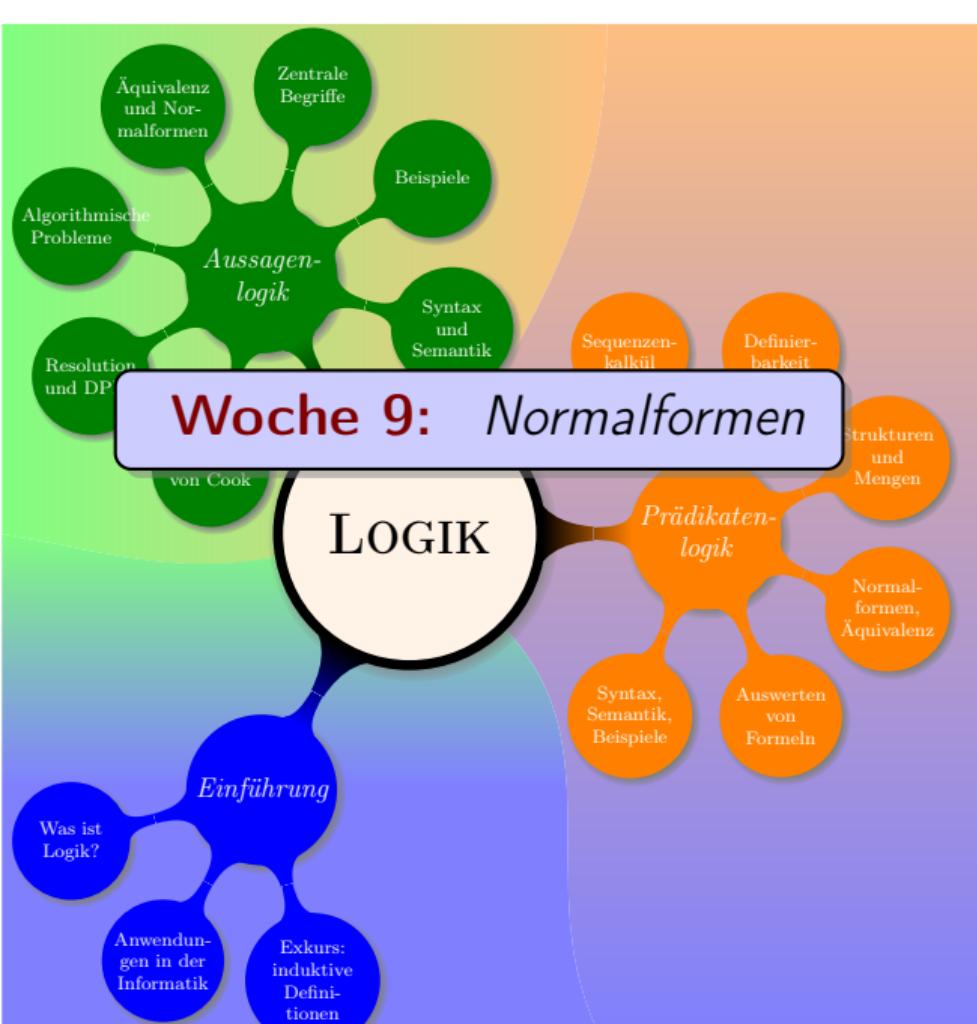
Es ergibt sich eine Laufzeit von $|A|^{O(|\varphi|)}$.

Allerdings wird nur $O(|\varphi| \cdot |A|)$ Platz benötigt.

Komplexität des Auswertungsproblems.

Das Auswertungsproblem für die Prädikatenlogik ist

- lösbar in Zeit exponentiell in der Formellänge aber nur polynomiell in der Strukturgröße.
- PSPACE vollständig.



Nov.	17	18	19	20	21	22	23	<i>Einführung</i>
	24	25	26	27	28	29	30	<i>Aussagenlogik</i>
	31	1	2	3	4	5	6	<i>Normalformen</i>
	7	8	9	10	11	12	13	<i>Resolution</i>
Dez.	14	15	16	17	18	19	20	<i>Kompaktheit</i>
	21	22	23	24	25	26	27	<i>DPLL, Satz von Cook</i>
	28	29	30	1	2	3	4	<i>Strukturen und FO</i>
	5	6	7	8	9	10	11	<i>Prädikatenlogik</i>
	12	13	14	15	16	17	18	<i>Komplexität von FO</i>
Jan.	19	20	21	22	23	24	25	<i>Weihnachten</i>
	26	27	28	29	30	31	1	<i>Neujahr</i>
	2	3	4	5	6	7	8	<i>Normalformen</i>
	9	10	11	12	13	14	15	<i>Definierbarkeit</i>
	16	17	18	19	20	21	22	<i>EF-Spiele</i>
Feb.	23	24	25	26	27	28	29	<i>EF-Spiele</i>
	30	31	1	2	3	4	5	<i>Sequenzkalkül AL</i>
	6	7	8	9	10	11	12	<i>Sequenzkalkül FO</i>
	13	14	15	16	17	18	19	<i>Ausblick</i>

9.1 Substitution

Erinnerung: Beispiele

$\sigma := \{+, *, <, 0, 1\}$: Signatur der Arithmetik

$\mathcal{A} := (\mathbb{N}, +^{\mathcal{A}}, *^{\mathcal{A}}, <^{\mathcal{A}}, 0^{\mathcal{A}}, 1^{\mathcal{A}})$: Struktur über den natürlichen Zahlen mit der üblichen Interpretation von $+, *, <, 0, 1$.

Beispiele.

- $\varphi_1(x) := \forall y (\exists z (y * z = x \rightarrow (y = 1 \vee z = 1)))$

- $\varphi_2 := \forall y \exists x (y < x \wedge \varphi_1(x))$

Anmerkung. Hier wird φ_1 in die Formel φ_2 „eingesetzt“. Im allgemeinen nicht unproblematisch, siehe nächste Woche.

Formeln als Unterformeln. Sei $\varphi(x_1, x_2)$ eine Formel.

Wir wollen eine andere Formel $\psi := \dots \exists y \exists z \varphi(y, z) \dots$ definieren.

Dazu wollen wir in φ die freien Variablen x_1 und x_2 durch y und z ersetzen.
 \rightsquigarrow Substitution

Substitution: informell

Analog zur Aussagenlogik wollen wir einen Begriff der **Substitution** einführen.

Ziel ist es, Variablen **sinnvoll** durch **Terme** zu ersetzen.

Wenn wir z.B. in der σ_{ar} -Formel

$$\exists y \ y * y = x + x$$

Variablen durch Formeln zu ersetzen wäre sinnlos. Warum?

die Variable **x** durch **(1 + 1)** ersetzen, erhalten wir

$$\exists y \ y * y = (1 + 1) + (1 + 1).$$

Substitution: informell

Das folgende Beispiel zeigt potentielle Probleme.

Beispiel. Sei $\varphi := \exists y y * y = x + x$.

1. Wenn wir in φ die freie Variable x durch y ersetzen, erhalten wir

$$\exists y y * y = y + y$$

was eine andere Bedeutung hat.

Wir müssen also auf Konflikte mit gebundenen Variablen achten.

2. Wenn wir in φ die gebundene Variable y durch x ersetzen, erhalten wir die Formel

$$\exists x x * x = x + x$$

ebenfalls mit anderer Bedeutung.

Wir sollten daher nur freie Variablen substituieren.

Substitution

Definition. Sei σ eine Signatur.

Beispiel.

$$\begin{aligned}\mathcal{S} : & \quad x \mapsto y + z \\ & \quad y \mapsto z + v.\end{aligned}$$

$$\text{var}(\mathcal{S}) := \{y, z, v\}.$$

1. Eine σ -Substitution ist eine Abbildung $\mathcal{S} : \text{def}(\mathcal{S}) \rightarrow \mathcal{T}_\sigma$ mit endlichem Wertebereich $\text{def}(\mathcal{S}) \subseteq \text{Var}$.
2. Für eine Substitution \mathcal{S} definieren wir $\text{var}(\mathcal{S})$ als die Menge der Variablen, die in einem Term im Bild der Substitution vorkommen, d.h.

$$\text{var}(\mathcal{S}) := \bigcup_{x \in \text{def}(\mathcal{S})} \text{var}(\mathcal{S}(x)).$$

Substitution in Termen

Definition. Sei \mathcal{S} eine σ -Substitution.

Induktiv über die Struktur von Termen definieren wir für jeden Term $t \in \mathcal{T}_\sigma$ den Term $t\mathcal{S}$, der durch **Anwendung** von \mathcal{S} auf t entsteht, als:

- Wenn $t := x$, wobei $x \in \text{Var}$, dann

$$t\mathcal{S} := \begin{cases} \mathcal{S}(x) & \text{wenn } x \in \text{def}(\mathcal{S}) \\ x & \text{sonst.} \end{cases}$$

- Wenn $t := c$, für ein Konstantensymbol $c \in \sigma$, dann $t\mathcal{S} := c$.
- Wenn $t := f(t_1, \dots, t_k)$, für ein k -stelliges Funktionssymbol $f \in \sigma$ und σ -Terme $t_1, \dots, t_k \in \mathcal{T}_\sigma$, dann
 $t\mathcal{S} := f(t_1\mathcal{S}, \dots, t_k\mathcal{S})$.

Beispiel.

$$\mathcal{S} : \begin{array}{l} x \mapsto y + z \\ y \mapsto z + v. \end{array}$$

$$\text{var}(\mathcal{S}) := \{y, z, v\}.$$

Für $t := x + y$ gilt

$$t\mathcal{S} := ((y + z) + (z + v)).$$

Substitution in Formeln

Definition. Sei \mathcal{S} eine σ -Substitution.

Induktiv definieren wir für $\varphi \in \text{FO}[\sigma]$ die Formel $\varphi\mathcal{S}$ als:

- Für $\varphi := R(t_1, \dots, t_k)$ gilt $\varphi\mathcal{S} := R(t_1\mathcal{S}, \dots, t_k\mathcal{S})$.
(wobei $t_1, \dots, t_k \in T_\sigma$, R k -stell. Relationssymbol)
- Für $\varphi := t_1 = t_2$ gilt $\varphi\mathcal{S} := t_1\mathcal{S} = t_2\mathcal{S}$ (wobei $t_1, t_2 \in T_\sigma$).
- Für $\varphi := \neg\psi$ gilt $\varphi\mathcal{S} := \neg\psi\mathcal{S}$.
- Für $\varphi := (\psi_1 * \psi_2)$ gilt $\varphi\mathcal{S} := (\psi_1\mathcal{S} * \psi_2\mathcal{S})$
(wobei $\psi_1, \psi_2 \in \text{FO}[\sigma]$ und $* \in \{\vee, \wedge, \rightarrow, \leftrightarrow\}$).
- Wenn $\varphi := \exists x\psi$, wobei $x \in \text{Var}$ und $\psi \in \text{FO}[\sigma]$, dann gilt:
 1. $\varphi\mathcal{S} := \exists x\psi\mathcal{S}'$, falls $x \notin \text{var}(\mathcal{S})$, wobei $\mathcal{S}' := \mathcal{S}|_{\text{def}(\mathcal{S}) \setminus \{x\}}$.
 2. Wenn $x \in \text{var}(\mathcal{S})$, wähle $y \in \text{Var} \setminus (\text{frei}(\varphi) \cup \text{var}(\mathcal{S}))$ und setze
 $\varphi\mathcal{S} := \exists y\psi\mathcal{S}'$, wobei $\mathcal{S}' := \mathcal{S}|_{\text{def}(\mathcal{S}) \setminus \{x\}} \cup \{x \mapsto y\}$.
- Der Fall $\varphi := \forall x\psi$ ist analog.

Beispiel.

$$\mathcal{S} : \begin{aligned} x &\mapsto y + z \\ y &\mapsto z + v. \end{aligned}$$

$$\text{var}(\mathcal{S}) := \{y, z, v\}.$$

Für $t := x + y$ gilt

$$t\mathcal{S} := ((y + z) + (z + v)).$$

Sei $\varphi := \exists a \forall z (x + y + z = a + x)$:

$$\begin{aligned} \varphi\mathcal{S} &:= \exists a (\forall z (x + y + z = a + x))\mathcal{S} \\ &= \exists a \forall v_0 (x + y + z = a + x)\mathcal{S}' \\ \mathcal{S}' &:= \mathcal{S} \cup \{z \mapsto v_0\} \\ &= \exists a \forall v_0 (x + y + z)\mathcal{S}' = (a + x)\mathcal{S}' \\ &= \exists a \forall v_0 ((y + z) + (z + v) + v_0 \\ &= (a + (y + z))). \end{aligned}$$

Notation

Notation.

- Analog zur Aussagenlogik schreiben wir für eine Substitution \mathcal{S} mit $\text{def}(\mathcal{S}) := \{x_1, \dots, x_n\}$ und $\mathcal{S}(x_i) := t_i, 1 \leq i \leq n$,

$$[x_1/t_1, \dots, x_n/t_n].$$

Das erlaubt uns, $\varphi[x_1/t_1, \dots, x_n/t_n]$ statt $\varphi\mathcal{S}$ zu schreiben.

- Für $\varphi(x_1, \dots, x_k) \in \text{FO}$ und $t_1, \dots, t_k \in \mathcal{T}_\sigma$ schreiben wir

$$\varphi[t_1, \dots, t_k] \quad \text{statt} \quad \varphi[x_1/t_1, \dots, x_k/t_k].$$

Vergleiche mit Methoden in Java.

Boolean phi(int $x_1, \dots, int x_k$)

Indem wir x_1, \dots, x_k spezifizieren, fixieren wir eine Ordnung der Parameter.

Boolean b = phi(3, 5, ..., 17);

Das Substitutionslemma

Lemma. Sei \mathcal{S} eine σ -Substitution. Für alle σ -Formeln φ, ψ :

$$\varphi \equiv \psi \implies \varphi\mathcal{S} \equiv \psi\mathcal{S}$$

Lemma (Ersetzungslemma).

Sei τ eine Signatur und seien $\varphi, \psi, \vartheta \in \text{FO}[\tau]$.

Sei ϑ eine Teilformel von ψ und $\vartheta \equiv \varphi$. Ferner, sei ψ' die Formel, die aus ψ entsteht, indem ϑ durch φ ersetzt wird.

Dann gilt $\psi \equiv \psi'$.

Beispiele

$\sigma := \{+, *, <, 0, 1\}$: Signatur der Arithmetik

$\mathcal{A} := (\mathbb{N}, +^{\mathcal{A}}, *^{\mathcal{A}}, <^{\mathcal{A}}, 0^{\mathcal{A}}, 1^{\mathcal{A}})$: Struktur über den natürlichen Zahlen mit der üblichen Interpretation von $+, *, <, 0, 1$

Beispiele.

- $\varphi_1(x) := \forall y (\exists z (y * z = x \rightarrow (y = 1 \vee z = 1)))$
- $\varphi_2 := \forall y \exists x (y < x \wedge \varphi_1(x)) \quad \exists z (\varphi_1(z) \wedge z * z = x)$

Anmerkung. Hier wird φ_1 in die Formel φ_2 „eingesetzt“. Im allgemeinen nicht unproblematisch, siehe nächste Woche.

Unterformeln

Mit Hilfe der Substitution können wir nun Unterformeln benutzen.

9.2 Logische Äquivalenz

Erinnerung: Äquivalenz zwischen Formeln

Definition. Sei σ eine Signatur.

Zwei σ -Formeln $\varphi, \psi \in \text{FO}[\sigma]$ sind äquivalent, geschrieben $\varphi \equiv \psi$, wenn für alle σ -Interpretationen \mathcal{I} passend zu φ und ψ :

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

Bemerkung. Nach Definition gilt für alle Formeln $\varphi, \psi \in \text{FO}[\sigma]$

$$\varphi \equiv \psi \iff (\varphi \leftrightarrow \psi \text{ ist allgemeingültig})$$

Erinnerung: Nützliche Äquivalenzen der Aussagenlogik

Theorem. Für alle $\psi, \varphi, \vartheta \in AL$:

1. $\neg\neg\varphi \equiv \varphi$ (Elimination doppelter Negation)
2. $\varphi \rightarrow \psi \equiv \neg\varphi \vee \psi$ (Elim. der Implikation)
3. $\varphi \leftrightarrow \psi \equiv (\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$ (Elim. der Biimplikation)
4. $\begin{aligned} \neg(\varphi \wedge \psi) &\equiv \neg\varphi \vee \neg\psi \\ \neg(\varphi \vee \psi) &\equiv \neg\varphi \wedge \neg\psi \end{aligned}$ (de Morgansche Regeln)
5. $\begin{aligned} \varphi \wedge (\psi \vee \vartheta) &\equiv (\varphi \wedge \psi) \vee (\varphi \wedge \vartheta) \\ \varphi \vee (\psi \wedge \vartheta) &\equiv (\varphi \vee \psi) \wedge (\varphi \vee \vartheta) \end{aligned}$ (Distributivität)
6. $\varphi \wedge (\varphi \vee \psi) \equiv \varphi \vee (\varphi \wedge \psi) \equiv \varphi$ (Absorptionsgesetz)
7. $\begin{aligned} \varphi \wedge \psi &\equiv \psi \wedge \varphi \\ \varphi \vee \psi &\equiv \psi \vee \varphi \end{aligned}$ (Kommutativität von \wedge und \vee)
8. $\begin{aligned} \varphi \wedge (\psi \wedge \vartheta) &\equiv (\varphi \wedge \psi) \wedge \vartheta \\ \varphi \vee (\psi \vee \vartheta) &\equiv (\varphi \vee \psi) \vee \vartheta \end{aligned}$ (Assoziativität von \wedge und \vee)

Einige Äquivalenzen der Prädikatenlogik

Lemma. Sei $\varphi, \psi \in \text{FO}[\sigma]$ und $x, y \in \text{Var}$.

$$1. \neg \exists x \varphi \equiv \forall x \neg \varphi, \quad \neg \forall x \varphi \equiv \exists x \neg \varphi$$

2. Wenn $x \notin \text{frei}(\varphi)$ dann

$$\begin{array}{lll} \varphi \vee \exists x \psi & \equiv & \exists x (\varphi \vee \psi), \\ \varphi \wedge \exists x \psi & \equiv & \exists x (\varphi \wedge \psi), \end{array} \quad \begin{array}{lll} \varphi \wedge \forall x \psi & \equiv & \forall x (\varphi \wedge \psi) \\ \varphi \vee \forall x \psi & \equiv & \forall x (\varphi \vee \psi) \end{array}$$

3. Wenn y nicht in φ vorkommt, dann gilt

$$\exists x \varphi \equiv \exists y \varphi[x/y], \quad \forall x \varphi \equiv \forall y \varphi[x/y]$$

1. $\neg \neg \varphi \equiv \varphi$
2. $\varphi \rightarrow \psi \equiv \neg \varphi \vee \psi$
3. $\varphi \leftrightarrow \psi \equiv (\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$
4. $\neg(\varphi \wedge \psi) \equiv \neg \varphi \vee \neg \psi$
 $\neg(\varphi \vee \psi) \equiv \neg \varphi \wedge \neg \psi$
5. $\psi \wedge (\varphi \vee \theta) \equiv (\psi \wedge \varphi) \vee (\psi \wedge \theta)$
 $\psi \vee (\varphi \wedge \theta) \equiv (\psi \vee \varphi) \wedge (\psi \vee \theta)$
6. $\psi \wedge (\psi \vee \varphi) \equiv \psi \vee (\psi \wedge \varphi) \equiv \psi$
7. $\psi \wedge \varphi \equiv \varphi \wedge \psi$
 $\psi \vee \varphi \equiv \varphi \vee \psi$
8. $\psi \wedge (\varphi \wedge \theta) \equiv (\psi \wedge \varphi) \wedge \theta$
 $\psi \vee (\varphi \vee \theta) \equiv (\psi \vee \varphi) \vee \theta$

Semantischer Beweis von Teil 1

Lemma. Sei $\varphi, \psi \in \text{FO}[\sigma]$ und $x \in \text{Var}$.

$$1. \neg \exists x \varphi \equiv \forall x \neg \varphi, \quad \neg \forall x \varphi \equiv \exists x \neg \varphi$$

Beweis. Wir zeigen hier $\neg \exists x \varphi \equiv \forall x \neg \varphi$. Die andere Aussage folgt analog.

Sei $\mathcal{I} := (\mathcal{A}, \beta)$ eine passende Interpretation.

Zu zeigen: $\mathcal{I} \models \neg \exists x \varphi$ gdw. $\mathcal{I} \models \forall x \neg \varphi$

1. Wenn $\mathcal{I} \models \neg \exists x \varphi$, dann gibt es kein $a \in A$ so dass $\mathcal{I}[x/a] \models \varphi$.

Für jedes $a \in A$ gilt also $\mathcal{I}[x/a] \not\models \varphi$, d.h. $\mathcal{I}[x/a] \models \neg \varphi$.

Also folgt $\mathcal{I} \models \forall x \neg \varphi$.

2. Wenn $\mathcal{I} \models \forall x \neg \varphi$, dann gilt $\mathcal{I}[x/a] \models \neg \varphi$, und somit $\mathcal{I}[x/a] \not\models \varphi$, für alle $a \in A$.

Also kann es kein $a \in A$ geben mit $\mathcal{I}[x/a] \models \varphi$.

Es folgt $\mathcal{I} \models \neg \exists x \varphi$.

Einige Äquivalenzen der Prädikatenlogik

Lemma. Sei $\varphi, \psi \in \text{FO}[\sigma]$ und $x, y \in \text{Var}$.

$$1. \neg \exists x \varphi \equiv \forall x \neg \varphi, \quad \neg \forall x \varphi \equiv \exists x \neg \varphi$$

2. Wenn $x \notin \text{frei}(\varphi)$ dann

$$\varphi \vee \exists x \psi \equiv \exists x (\varphi \vee \psi), \quad \varphi \wedge \forall x \psi \equiv \forall x (\varphi \wedge \psi)$$

$$\varphi \wedge \exists x \psi \equiv \exists x (\varphi \wedge \psi), \quad \varphi \vee \forall x \psi \equiv \forall x (\varphi \vee \psi)$$

3. Wenn y nicht in φ vorkommt, dann gilt

$$\exists x \varphi \equiv \exists y \varphi[x/y], \quad \forall x \varphi \equiv \forall y \varphi[x/y]$$

Sequenzkalkül. Die anderen Äquivalenzen werden wir später mit Hilfe des Sequenzkalküls beweisen.

9.3 Normalformen

Normalformen

Ähnlich wie bei der Aussagenlogik werden wir als nächstes einige syntaktische Normalformen für die Prädikatenlogik einführen, die uns das Arbeiten mit Formeln in bestimmten Situationen erleichtern.

Konkret werden wir folgende Normalformen vorstellen:

- Reduzierte Formeln
- Negationsnormalform
- Pränexnormalform

Reduzierte Formeln

Wir haben bereits gesehen, dass folgende Äquivalenzen gelten:

1. $(\psi \wedge \varphi) \equiv \neg(\neg\varphi \vee \neg\psi)$
2. $(\psi \leftrightarrow \varphi) \equiv (\varphi \rightarrow \psi) \wedge (\psi \rightarrow \varphi)$
3. $(\psi \rightarrow \varphi) \equiv (\neg\psi \vee \varphi)$
4. $\forall x\psi \equiv \neg\exists x\neg\psi$

Mit Hilfe dieser Äquivalenzen können wir also jede Formel der Prädikatenlogik in eine äquivalente Formel umwandeln, in denen die Symbole $\leftrightarrow, \rightarrow, \wedge$ sowie der \forall -Quantor nicht vorkommen.

Wir nennen solche Formeln **reduzierte Formeln**.

Die Negationsnormalform

Definition. Eine prädikatenlogische Formel ist in *Negationsnormalform* (NNF), wenn sie die Verknüpfungen \rightarrow , \leftrightarrow nicht enthält und Negation nur direkt vor atomaren Formeln vorkommt.

Beispiele.

- $\exists x(\neg P(x, y) \vee Q(y))$ ist in NNF.
- $\exists x(P(x, y) \rightarrow Q(y))$ ist nicht in NNF.
- $\exists x(\neg R(x, y) \wedge (\forall z P(x) \wedge \neg \forall z Q(x)))$ ist nicht in NNF.

Theorem. Jede Formel der Prädikatenlogik ist logisch äquivalent zu einer Formel in Negationsnormalform.

Beweis

Definition. Eine prädikatenlogische Formel ist in *Negationsnormalform*, wenn sie die Verknüpfungen $\rightarrow, \leftrightarrow$ nicht enthält und Negation nur direkt vor atomaren Formeln vorkommt.

Theorem. Jede Formel der Prädikatenlogik ist logisch äquivalent zu einer Formel in Negationsnormalform.

Beweis.

- Wissen bereits: Verknüpfungen $\rightarrow, \leftrightarrow$ können eliminiert werden.
- Durch wiederholte Anwendung der De Morganschen Regeln

Siehe Aussagenlogik.

$$\neg(\psi \wedge \varphi) \equiv (\neg\psi \vee \neg\varphi) \text{ und } \neg(\psi \vee \varphi) \equiv (\neg\psi \wedge \neg\varphi)$$

sowie der Äquivalenzen

$$\neg\nexists x\psi \equiv \forall x\neg\psi \text{ und } \neg\forall x\psi \equiv \exists x\neg\psi \text{ und } \neg\neg\psi \equiv \psi$$

kann jede Formel in eine äquivalente Formel in NNF umgewandelt werden.

Beweis

Definition. Eine prädikatenlogische Formel ist in *Negationsnormalform* Verknüpfungen $\rightarrow, \leftrightarrow$ nicht enthält und Negation nur direkt vor Formeln vorkommt.

Theorem. Jede Formel der Prädikatenlogik ist logisch äquivalent einer Formel in Negationsnormalform.

Beweis.

- Wissen bereits: Verknüpfungen $\rightarrow, \leftrightarrow$ können eliminiert werden
- Durch wiederholte Anwendung der De Morganschen Regeln

Beispiel.

$$\begin{aligned} & \neg \forall x (\exists y P(x, y) \wedge \exists z (Q(z) \wedge Q(x))) \\ & \equiv \exists x \neg (\exists y P(x, y) \wedge \exists z (Q(z) \wedge Q(x))) \\ & \equiv \exists x (\neg \exists y P(x, y) \vee \neg \exists z (Q(z) \wedge Q(x))) \\ & \equiv \exists x (\forall y \neg P(x, y) \vee \forall z \neg (Q(z) \wedge Q(x))) \\ & \equiv \exists x (\forall y \neg P(x, y) \vee \forall z (\neg Q(z) \vee \neg Q(x))) \end{aligned}$$

Siehe Aussagenlogik.

$$\neg(\psi \wedge \varphi) \equiv (\neg\psi \vee \neg\varphi) \quad \text{und} \quad \neg(\psi \vee \varphi) \equiv (\neg\psi \wedge \neg\varphi)$$

sowie der Äquivalenzen

$$\neg \exists x \psi \equiv \forall x \neg \psi \quad \text{und} \quad \neg \forall x \psi \equiv \exists x \neg \psi \quad \text{und} \quad \neg \neg \psi \equiv \psi$$

kann jede Formel in eine äquivalente Formel in NNF umgewandelt werden.

Die Pränexnormalform

Definition. Eine Formel φ ist **bereinigt**, wenn

- keine Variable in φ sowohl frei als auch gebunden vorkommt und
- keine Variable zweimal quantifiziert wird.

Definition. Eine Formel φ ist in **Pränexnormalform (PNF)**, wenn sie

1. bereinigt ist und
2. die Form

$$Q_1 x_1 \dots Q_I x_I \psi(x_1, \dots, x_I)$$

hat, wobei ψ quantorenfrei und $Q_i \in \{\exists, \forall\}$ ist.

Q_1, \dots, Q_I heißt der (Quantoren-) Präfix von ψ .

Beispiel. $\exists u \forall x \exists y \exists z (R(u, u) \wedge R(x, y) \wedge \neg R(y, y) \wedge R(y, z))$

Theorem. Jede Formel der Prädikatenlogik kann effektiv in eine äquivalente Formel in Pränexnormalform übersetzt werden.

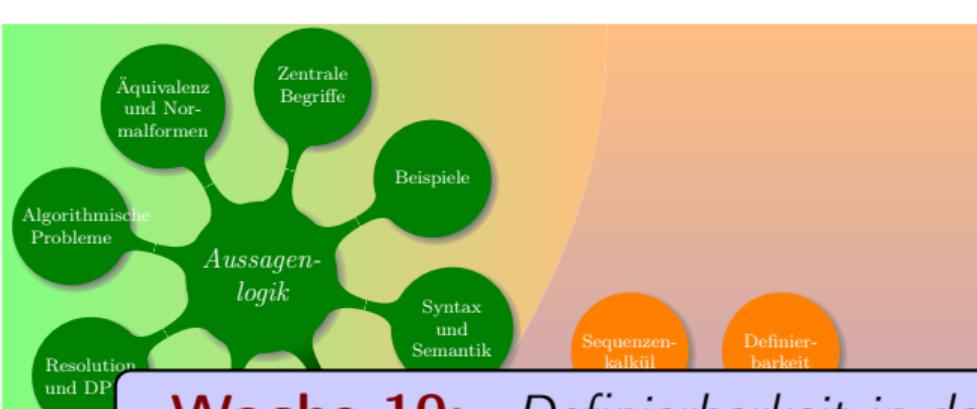
Lemma. Wenn y nicht in φ vorkommt, dann gilt:

$$\begin{aligned}\exists x \varphi &\equiv \exists y \varphi[x/y] \\ \forall x \varphi &\equiv \forall y \varphi[x/y]\end{aligned}$$

Lemma.

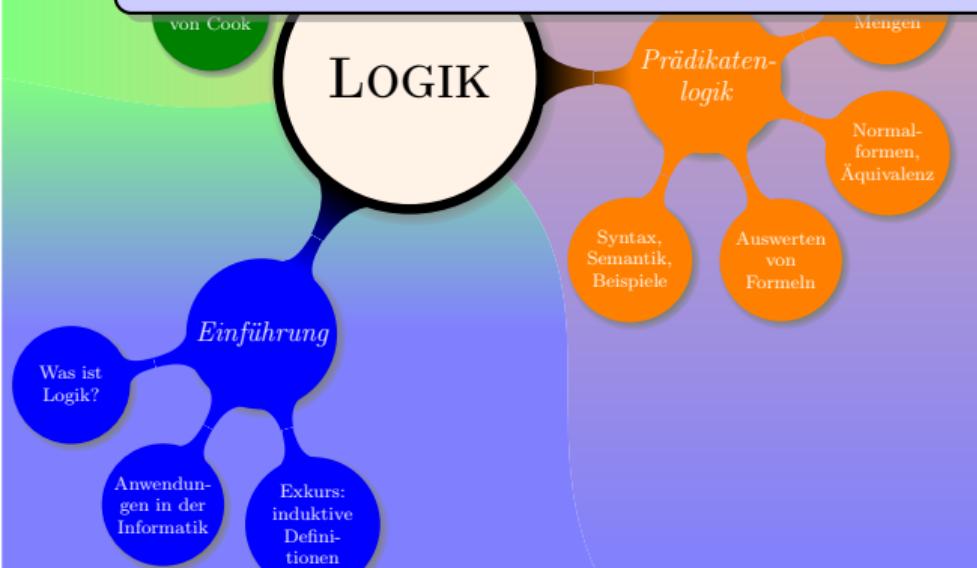
Wenn $x \notin \text{frei}(\varphi)$, dann

$$\begin{aligned}(\varphi \vee \exists x \psi) &\equiv \exists x (\varphi \vee \psi) \\ (\varphi \vee \forall x \psi) &\equiv \forall x (\varphi \vee \psi) \\ (\varphi \wedge \exists x \psi) &\equiv \exists x (\varphi \wedge \psi) \\ (\varphi \wedge \forall x \psi) &\equiv \forall x (\varphi \wedge \psi)\end{aligned}$$



Nov.	17	18	19	20	21	22	23	<i>Einführung</i>
	24	25	26	27	28	29	30	<i>Aussagenlogik</i>
	31	1	2	3	4	5	6	<i>Normalformen</i>
	7	8	9	10	11	12	13	<i>Resolution</i>
Dez.	14	15	16	17	18	19	20	<i>Kompaktheit</i>
	21	22	23	24	25	26	27	<i>DPLL, Satz von Cook</i>

Woche 10: Definierbarkeit in der Prädikatenlogik: I



5	6	7	8	9	10	11	<i>Prädikatenlogik</i>	
12	13	14	15	16	17	18	<i>Komplexität von FO</i>	
Jan.	19	20	21	22	23	24	25	<i>Weihnachten</i>
	26	27	28	29	30	31	1	<i>Neujahr</i>
	2	3	4	5	6	7	8	<i>Normalformen</i>
	9	10	11	12	13	14	15	<i>Definierbarkeit</i>
	16	17	18	19	20	21	22	<i>EF-Spiele</i>
Feb.	23	24	25	26	27	28	29	<i>EF-Spiele</i>
	30	31	1	2	3	4	5	<i>Sequenzkalkül AL</i>
	6	7	8	9	10	11	12	<i>Sequenzkalkül FO</i>
	13	14	15	16	17	18	19	<i>Ausblick</i>

10.1 Definierbarkeit in der Prädikatenlogik

Definierbarkeit in der Prädikatenlogik

Beispiel. Betrachten wir eine Datenbank \mathcal{D} mit Fluginformationen.

Tabelle:

Flug(Fluggesellschaft, Flugnummer, Zeit, Start, Ziel)

Frage. Ist es möglich (evtl. mit Zwischenstopps) von s nach t zu fliegen?

Definierbarkeit. Ist diese Anfrage in FO definierbar?

D.h. gibt es $\varphi(x, y) \in \text{FO}$, so dass für $s, t \in D$ gilt:

Es ist möglich, von s nach t zu fliegen

gdw.

$\mathcal{D} \models \varphi[s, t]?$

Wiederholung: Modellklassen und die Relation $\varphi(\mathcal{A})$

Formeln $\varphi(x)$. Eine Formel $\varphi(x)$ sagt etwas über ein Element innerhalb einer Struktur aus.

Sätze ψ . Ein Satz ψ sagt etwas über die Struktur insgesamt aus.

Oft interessieren wir uns für die „Menge“ aller Objekte, die eine Formel bzw. einen Satz erfüllen.

Formeln. Bei Formeln $\varphi(x_1, \dots, x_k)$ ist diese „Menge“ die *durch φ in einer Struktur \mathcal{A} definierte Relation*

$$\varphi(\mathcal{A}) := \{(a_1, \dots, a_k) \in A^k : (\mathcal{A}, [x_1/a_1, \dots, x_k/a_k]) \models \varphi\}.$$

Umgekehrt: $R \subseteq A^k$ ist *FO-definierbar in \mathcal{A}* , wenn es eine Formel $\varphi(x_1, \dots, x_k) \in \text{FO}$ gibt, so dass $\varphi(\mathcal{A}) = R$.

Sätze. Bei einem Satz $\varphi \in \text{FO}[\sigma]$ ist diese „Menge“ die *Modellklasse*

$$\text{Mod}(\varphi) := \{\mathcal{A} : \mathcal{A} \text{ } \sigma\text{-Struktur mit } \mathcal{A} \models \varphi\}.$$

Modellklassen und Axiomensysteme

Definition.

Sei σ eine Signatur und \mathcal{C} eine Klasse \mathcal{C} von σ -Strukturen.

1. \mathcal{C} wird **axiomatisiert** durch eine Menge $\Phi \subseteq \text{FO}[\sigma]$, oder Φ ist ein **Axiomensystem** für \mathcal{C} , wenn $\mathcal{C} = \text{Mod}(\Phi)$.
2. \mathcal{C} ist **FO-axiomatisierbar**, wenn $\mathcal{C} = \text{Mod}(\Phi)$ für eine Menge $\Phi \subseteq \text{FO}[\sigma]$.
3. \mathcal{C} ist **FO-definierbar**, oder **endlich axiomatisierbar**, wenn $\mathcal{C} = \text{Mod}(\varphi)$ für einen einzigen Satz $\varphi \in \text{FO}[\sigma]$.
Äquivalent. $\mathcal{C} = \text{Mod}(\Phi)$ für eine endlich Menge $\Phi \subseteq \text{FO}[\sigma]$.

Bemerkung.

Eine Klasse von Strukturen ist **endlich FO-axiomatisierbar** gdw. sie durch **einen einzigen FO-Satz** definiert wird.

Notation. Wir sagen meistens kurz **definierbar** statt **FO-definierbar**.

Beispiel: Strikte Lineare Ordnungen

Lineare Ordnungen. Sei $\sigma := \{<\}$ die Signatur der Ordnungen und sei

$$\varphi_{ord} := \begin{aligned} & \forall x \neg x < x \wedge \\ & \forall x \forall y \forall z ((x < y \wedge y < z) \rightarrow x < z) \wedge \\ & \forall x \forall y (x < y \vee x = y \vee y < x) \end{aligned}$$

Dann ist $\text{Mod}(\varphi_{ord})$ die Klasse aller strikten linearen Ordnungen.

Beobachtung.

1. Es gilt $\varphi_{ord} \models \forall x \forall y ((x < y \wedge y < x) \rightarrow x = y)$

(Jede strikte lineare Ordnung ist antisymmetrisch.)

2. Aber $\varphi_{ord} \not\models \exists x \forall y (x = y \vee x < y)$

(Nicht jede lineare Ordnung hat ein kleinstes Element.)

Bemerkung. Gäbe es ein (automatisches) Beweisverfahren für \models , könnten Aussagen der Ordnungstheorie automatisch bewiesen werden

↔ *Sequenzkalkül* und *Theorembeweiser*

Definition.

Sei $\Phi \subseteq \text{FO}[\sigma]$ und \mathcal{C} Klasse von σ -Strukturen.

1. \mathcal{C} axiomatisiert durch Φ , oder Φ ist *Axiomensystem* für \mathcal{C} , wenn $\mathcal{C} = \text{Mod}(\Phi)$.
2. \mathcal{C} ist *FO-axiomatisierbar*, wenn $\mathcal{C} = \text{Mod}(\Phi)$ für eine $\Phi \subseteq \text{FO}[\sigma]$.
3. \mathcal{C} ist *FO-definierbar*, wenn $\mathcal{C} = \text{Mod}(\Phi)$ für ein $\varphi \in \text{FO}[\sigma]$.

Beispiel: lineare Ordnungen

Beobachtung.

1. Die Klasse der linearen Ordnungen ist durch φ_{ord} definierbar.

$$\text{Mod}(\varphi_{ord}) = \{ \text{Klasse der linearen Ordnungen} \}.$$

2. Die Klasse der *unendlichen linearen Ordnungen* ist nicht FO-definierbar (wird später bewiesen).

Sie ist aber FO-axiomatisierbar, z.B. durch

$$\Phi_{\infty-ord} := \{\varphi_{ord}\} \cup \{\varphi_i : i \geq 1\} \quad \text{mit} \quad \varphi_i := \exists x_1 \dots \exists x_i \bigwedge_{1 \leq j < j' \leq i} x_j < x_{j'}.$$

Frage. Kann man die Klasse der endlichen linearen Ordnungen definieren oder axiomatisieren?

Antwort. Nein, die Klasse der *endlichen linearen Ordnungen* ist nicht einmal axiomatisierbar. (Beweis später)

Beispiel: Erreichbarkeit

Beispiel. Betrachte eine Datenbank \mathcal{D} mit Fluginformationen.

Flug(Fluggesellschaft, Flugnummer, Zeit, Start, Ziel)

Frage. Ist es möglich (evtl. mit Umsteigen) von s nach t zu fliegen?

Vereinfachung. Wir vereinfachen das Beispiel ein wenig.

Flug(Start, Ziel)

Formalisierung als Modellklasse.

Signatur $\sigma := \{E, s, t\}$

s, t Konstantensymbole

E 2-stelliges Relationssymbol

Frage. Ist folgende Klasse FO -definierbar?

$\text{REACH} := \{\mathcal{A} : \mathcal{A} \text{ } \sigma\text{-Struktur, es gibt einen Pfad von } s^{\mathcal{A}} \text{ nach } t^{\mathcal{A}}\}.$

Beispiel: Erreichbarkeit

Beispiel. Betrachte eine Datenbank \mathcal{D} mit Fluginformationen.

Flug(Fluggesellschaft, Flugnummer, Zeit, Start, Ziel)

Frage. Ist es möglich (evtl. mit Umsteigen) von s nach t zu fliegen?

Vereinfachung. Wir vereinfachen das Beispiel ein wenig.

Flug(Start, Ziel)

Formalisierung als Modellklasse.

Signatur $\sigma := \{E, s, t\}$

s, t Konstantensymbole

E 2-stelliges Relationssymbol

Freie Variablen werden
durch Konstantensymbole
modelliert.

Frage. Ist folgende Klasse FO-definierbar?

$\text{REACH} := \{\mathcal{A} : \mathcal{A} \text{ } \sigma\text{-Struktur, es gibt einen Pfad von } s^{\mathcal{A}} \text{ nach } t^{\mathcal{A}}\}.$

Erreichbarkeit in der Prädikatenlogik

Frage. Ist folgende Klasse FO-definierbar?

$\text{REACH} := \{\mathcal{A} : \mathcal{A} \text{ } \sigma\text{-Struktur, es gibt einen Pfad von } s^{\mathcal{A}} \text{ nach } t^{\mathcal{A}}\}.$

Versuch einer Antwort.

1. Direktflug: $\varphi_1 := E(s, t)$
2. Ein Stopp : $\varphi_2 := \exists x_1 (E(s, x_1) \wedge E(x_1, t))$
3. Zwei Stopps : $\varphi_3 := \exists x_1 \exists x_2 (E(s, x_1) \wedge E(x_1, x_2) \wedge E(x_2, t))$

Beobachtung. Für jede feste Zahl von Stopps existiert eine Formel.

Definierbarkeit in der Prädikatenlogik

Frage. Ist folgende Klasse FO-definierbar?

REACH := $\{\mathcal{A} : \mathcal{A} \text{ } \sigma\text{-Struktur, es gibt einen Pfad von } s^{\mathcal{A}} \text{ nach } t^{\mathcal{A}}\}$.

Unbeschränkt viele Zwischenstopps.

Aber gibt es eine Formel φ die genau dann in einer σ -Struktur gilt, wenn es eine Möglichkeit gibt, von s nach t zu fliegen, egal wieviele Stopps eingelegt werden müssen?

Beweis? Angenommen, wir wollen zeigen, dass es keine solche gibt?

Wie kann man eine solche Aussage zeigen?

Methoden.

- Für einige Definierbarkeitsfragen liefert der Kompaktheitssatz eine einfache Antwort.
- *Ehrenfeucht-Fraïssé Spiele* und m -Äquivalenz.

Definierbarkeit in der Prädikatenlogik

Inhalt dieses Abschnitts. Methoden, um zu Überprüfen, ob bestimmte Eigenschaften von Strukturen in der Prädikatenlogik definierbar sind.

Beispiel. Datenbank mit Fluginformationen.

Flug(Fluggesellschaft, Flugnummer, Zeit, Start, Ziel)

Ist es möglich, von s nach t (evtl. mit Zwischenstopps) zu fliegen?

Ist $\text{REACH} := \{\mathcal{A} : \mathcal{A} \text{ } \sigma\text{-Struktur, es gibt einen Pfad von } s^{\mathcal{A}} \text{ nach } t^{\mathcal{A}} \}$
definierbar oder axiomatisierbar?

Lernziele.

- Ein Gefühl dafür zu bekommen, welche Arten von Eigenschaften in der Prädikatenlogik beschrieben werden können.
- Methoden kennenzulernen, mit denen man beweisen kann, dass bestimmte Eigenschaften nicht in der Prädikatenlogik ausgedrückt werden können.

Ein weiteres Beispiel: Wortsprachen

Wortsprachen

Definierbare Sprachen. Sei $\Sigma := \{a, b\}$.

Wir wollen Sprachen $L \subseteq \Sigma^*$ durch logische Formeln definieren.

Dazu modellieren wir $w \in \Sigma^+$ als *Wortstruktur* \mathcal{W}_w .

Signatur. $\sigma_\Sigma := \{\leq, P_a, P_b\}$

\leq 2-stelliges Relationssymbol

P_a, P_b 1-stellige Relationssymbole

Wörter. $w = a_1 \dots a_n$

$\mathcal{W}_w := (W, \leq^{\mathcal{W}_w}, P_a^{\mathcal{W}_w}, P_b^{\mathcal{W}_w})$

$W := \{1, \dots, n\}$

$\leq^{\mathcal{W}_w}$:= natürliche Ordnung auf W

$P_a^{\mathcal{W}_w} := \{i : a_i = a\}$

$P_b^{\mathcal{W}_w} := \{i : a_i = b\}$

Beispiel. $w := a \ b \ a \ a \ b$

1 2 3 4 5

$\mathcal{W}_{abaab} := (W, \leq^{\mathcal{W}_w}, P_a^{\mathcal{W}_w}, P_b^{\mathcal{W}_w})$

$W := \{1, \dots, 5\}$

$P_a^{\mathcal{W}_w} := \{1, 3, 4\}$

$P_b^{\mathcal{W}_w} := \{2, 5\}$

Wortsprachen

Signatur. Sei $\Sigma := \{a, b\}$.

$$\sigma_\Sigma := \{\leq, P_a, P_b\}$$

\leq 2-stelliges Relationssymbol

P_a, P_b 1-stellige Relationssymbole

Wörter. $w = a_1 \dots a_n$

$$\mathcal{W}_w := (W, \leq^{\mathcal{W}_w}, P_a^{\mathcal{W}_w}, P_b^{\mathcal{W}_w})$$

$W := \{1, \dots, n\}$

$\leq^{\mathcal{W}_w}$:= natürliche Ordnung auf W

$P_a^{\mathcal{W}_w} := \{i : a_i = a\}$

$P_b^{\mathcal{W}_w} := \{i : a_i = b\}$

Definierbare Sprachen. Satz $\varphi \in \text{FO}[\sigma_\Sigma]$ definiert

$$\mathcal{L}(\varphi) := \{w \in \Sigma^+ : \mathcal{W}_w \models \varphi\}.$$

Beispiel. $w := a \ b \ a \ a \ b$

1 2 3 4 5

$$\mathcal{W}_{abaab} := (W, \leq^{\mathcal{W}_w}, P_a^{\mathcal{W}_w}, P_b^{\mathcal{W}_w})$$

$W := \{1, \dots, 5\}$

$P_a^{\mathcal{W}_w} := \{1, 3, 4\}$

$P_b^{\mathcal{W}_w} := \{2, 5\}$

Beispiel.

$$\varphi := \forall x(P_a(x) \rightarrow \exists y(x \leq y \wedge P_b(y))).$$

$\mathcal{L}(\varphi) := \{w : \text{nach jedem } a \text{ kommt irgendwann ein } b\}.$

Definierbare Sprachen

Definierbare Sprachen. Ist die

Sprache aller Wörter, die eine gerade Anzahl von as enthalten,
in FO definierbar?

Welche Arten formaler Sprachen sind FO -definierbar?

Alle regulären Sprachen?

Sind alle FO -definierbaren Sprachen regulär?

Sprachen und Modellklassen. $L \subseteq \Sigma^+$ ist FO -definierbar, wenn es $\varphi \in \text{FO}[\sigma_\Sigma]$ gibt, so dass $L = \mathcal{L}(\varphi) = \{w : \mathcal{W}_w \models \varphi\}$.

Im Prinzip suchen wir also $\varphi \in \text{FO}[\sigma_\Sigma]$ mit $\{\mathcal{W}_w : w \in L\} = \text{Mod}(\varphi)$.

Problem. Es kann auch andere σ_Σ -Strukturen \mathcal{B} geben, die φ erfüllen,
aber keine Wortstrukturen sind.

Dann gilt $\{\mathcal{W}_w : w \in L\} \neq \text{Mod}(\varphi)$, obschon das nur daran liegt,
dass $\text{Mod}(\varphi)$ Strukturen enthält, die uns gar nicht interessieren.

Definition.

Sei $\Phi \subseteq \text{FO}[\sigma]$ und
 \mathcal{C} Klasse von σ -Strukturen.

1. \mathcal{C} axiomatisiert durch Φ , oder
 Φ ist **Axiomensystem** für \mathcal{C} ,
wenn $\mathcal{C} = \text{Mod}(\Phi)$.
2. \mathcal{C} ist **FO-axiomatisierbar**, wenn
 $\mathcal{C} = \text{Mod}(\Phi)$ für eine $\Phi \subseteq \text{FO}[\sigma]$.
3. \mathcal{C} ist **FO-definierbar**, wenn
 $\mathcal{C} = \text{Mod}(\Phi)$ für ein $\varphi \in \text{FO}[\sigma]$.

Relative Definierbarkeit

Definition. Sei σ eine Signatur und \mathcal{K} eine Klasse von σ -Strukturen.

Eine Klasse $\mathcal{C} \subseteq \mathcal{K}$ ist in \mathcal{K} FO-definierbar, wenn es einen Satz $\psi \in \text{FO}[\sigma]$ gibt, so dass

$$\mathcal{C} = \{\mathcal{A} \in \mathcal{K} : \mathcal{A} \models \psi\}$$

Wir schreiben $\text{Mod}_{\mathcal{K}}(\varphi) := \{\mathcal{A} \in \mathcal{K} : \mathcal{A} \models \varphi\}$.

Sprachen als Modellklassen. Sei \mathfrak{W} die Klasse aller σ_{Σ} -Wortstrukturen.

Dann ist $\mathcal{L} \subseteq \Sigma^+$ FO-definierbar gdw. es $\varphi \in \text{FO}[\sigma_{\Sigma}]$ gibt mit

$$\{\mathcal{W}_w : w \in \mathcal{L}\} = \text{Mod}_{\mathfrak{W}}(\varphi).$$

Beispiel. Ist die Sprache

$$\mathcal{L}_{\text{EVEN-}a} := \{w \in \{a, b\}^+ : w \text{ enthält eine gerade Anzahl } as\}$$

FO-definierbar in \mathfrak{W} ?

Die Klasse EVEN_{\leq}

Beispiel. Ist die Sprache

$$\mathcal{L}_{\text{EVEN-}a} := \{w \in \{a, b\}^+ : w \text{ enthält eine gerade Anzahl } as\}$$

FO-definierbar in \mathfrak{W} ?

Vereinfachung. Ob $w \in \mathcal{L}_{\text{EVEN-}a}$ hängt nur von den vorkommenden as ab.

Es reicht daher, Wörter über dem Alphabet $U := \{a\}$ zu betrachten, d.h.

$$\sigma_U\text{-Wortstrukturen } \mathcal{W} := (W, \leq^{\mathcal{W}}, P_a^{\mathcal{W}}) \text{ mit } P_a^{\mathcal{W}} = W.$$

Da $P_a^{\mathcal{W}} = W$, können wir die Relation $P_a^{\mathcal{W}}$ ignorieren und nur die lineare Ordnung $(W, \leq^{\mathcal{W}})$ betrachten.

Die Klasse EVEN_{\leq} . Die Frage, ob $\mathcal{L}_{\text{EVEN-}a}$ FO-definierbar ist, reduziert sich auf die FO-Definierbarkeit der Klasse

$$\text{EVEN}_{\leq} := \{(A, \leq) : A \text{ ist endlich und gerader Länge}\}$$

in der Klasse \mathcal{O} aller endlichen linearen Ordnungen.

Beispiele dieses Abschnitts

Frage 1: Kann man in FO zählen?

Ist die Klasse

$$\text{EVEN}_{\leq} := \{(A, \leq) : A \text{ ist endlich und gerader Länge}\}$$

in der Klasse \mathcal{O} aller endlichen linearen Ordnungen FO -definierbar?

Das ist letztlich die Frage, ob die Prädikatenlogik zählen kann.

Frage 2: Erreichbarkeit.

$$\text{Signatur } \sigma := \{E, s, t\}$$

s, t Konstantensymbole

E 2-stelliges Relationssymbol

Ist folgende Klasse FO -definierbar?

$$\text{REACH} := \{\mathcal{A} : \mathcal{A} \text{ } \sigma\text{-Struktur, es gibt einen Pfad von } s^{\mathcal{A}} \text{ nach } t^{\mathcal{A}}\}.$$

Dahinter steht die Frage, ob die Prädikatenlogik **Schleifenkonstrukte** oder **Rekursion** ausdrücken kann.

Zusammenfassung

Definition.

Sei σ eine Signatur und \mathcal{C} eine Klasse \mathcal{C} von σ -Strukturen.

1. \mathcal{C} wird **axiomatisiert** durch eine Menge $\Phi \subseteq \text{FO}[\sigma]$, oder Φ ist ein **Axiomensystem** für \mathcal{C} , wenn $\mathcal{C} = \text{Mod}(\Phi)$.
2. \mathcal{C} ist **FO-axiomatisierbar**, wenn $\mathcal{C} = \text{Mod}(\Phi)$ für eine Menge $\Phi \subseteq \text{FO}[\sigma]$.
3. \mathcal{C} ist **FO-definierbar**, oder **endlich axiomatisierbar**, wenn $\mathcal{C} = \text{Mod}(\varphi)$ für einen einen Satz $\varphi \in \text{FO}[\sigma]$.
Äquivalent. $\mathcal{C} = \text{Mod}(\Phi)$ für eine endlich Menge $\Phi \subseteq \text{FO}[\sigma]$.
4. Eine Klasse $\mathcal{C} \subseteq \mathcal{K}$ ist **in \mathcal{K} FO-definierbar**, wenn es einen Satz $\psi \in \text{FO}[\sigma]$ gibt, so dass

$$\mathcal{C} = \{\mathcal{A} \in \mathcal{K} : \mathcal{A} \models \psi\}$$

Wir schreiben $\text{Mod}_{\mathcal{K}}(\varphi) := \{\mathcal{A} \in \mathcal{K} : \mathcal{A} \models \varphi\}$.

10.2 Der Quantorenrang von Formeln

Der Quantorenrang einer Formel

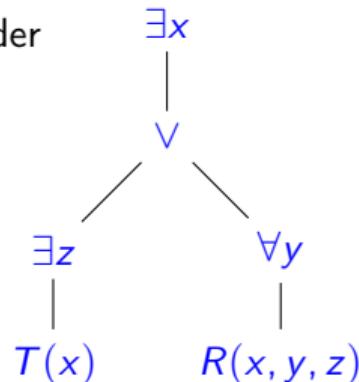
Definition. Der *Quantorenrang* $\text{qr}(\psi)$ einer Formel $\psi \in \text{FO}$ ist induktiv definiert durch:

- $\text{qr}(\psi) := 0$ für quantorenfreie Formeln ψ
- $\text{qr}(\neg\psi') := \text{qr}(\psi')$
- $\text{qr}((\varphi * \psi)) := \max\{\text{qr}(\varphi), \text{qr}(\psi)\}$ für $* \in \{\vee, \wedge, \rightarrow, \leftrightarrow\}$
- $\text{qr}(\exists x\varphi) = \text{qr}(\forall x\varphi) = 1 + \text{qr}(\varphi)$

Der Quantorenrang ist also die maximale *Schachtelungstiefe* der Quantoren in einer Formel.

Beispiel.

- $\text{qr}(\exists x\forall y(x = y \vee R(x, y, z))) = 2$
- $\text{qr}(\exists x(\exists z T(x) \vee \forall y R(x, y, z))) = 2$
- $\text{qr}((\forall z \neg E(x, z) \vee \forall z E(z, x))) = 1$



Zahl paarweise nicht-äquivalenter Formeln

Definition. Wir nennen eine Signatur σ *relational*, wenn σ nur Relationssymbole enthält.

Lemma. Sei σ eine endliche relationale Signatur.

Für alle $m, k \geq 0$ gibt es nur endlich viele paarweise nicht-äquivalente Formeln $\psi(x_1, \dots, x_k) \in \text{FO}[\sigma]$ vom Quantorenrang $\leq m$.

Folgerung. Für $m \geq 0$ (und $k = 0$) sei

$$\text{Qr}_m := \{\varphi \in \text{FO}[\sigma] : qr(\varphi) \leq m\}$$

und

$$\equiv_m := \{(\varphi, \psi) : \varphi, \psi \in \text{Qr}_m \text{ und } \psi \equiv \varphi\}.$$

Dann ist \equiv_m eine Äquivalenzrelation auf Qr_m mit endlichem Index.

Ein nützliches, wenn auch etwas technisches Lemma

Lemma. Sei σ eine endliche relationale Signatur.

Für alle $m, k \geq 0$ gibt es nur endlich viele paarweise nicht-äquivalente Formeln $\psi(x_1, \dots, x_k) \in \text{FO}[\sigma]$ vom Quantorenrang $\leq m$.

Definition. Sei $\Phi \subseteq \text{FO}[\sigma]$. Die Klasse $BK(\Phi)$ der Booleschen Kombinationen von Φ ist die kleinste Klasse für die gilt:

- $\Phi \subseteq BK(\Phi)$
- $(\varphi \wedge \psi), (\varphi \vee \psi), \neg \varphi \in BK(\Phi)$ für alle $\varphi, \psi \in BK(\Phi)$

Lemma. Sei $\Phi \subseteq \text{FO}[\sigma]$ und \mathcal{I}, \mathcal{B} σ -Interpretationen.

Wenn $\mathcal{I} \models \varphi$ gdw. $\mathcal{J} \models \varphi$ für alle $\varphi \in \Phi$,

dann $\mathcal{I} \models \psi$ gdw. $\mathcal{J} \models \psi$ für alle $\psi \in BK(\Phi)$.

Beweis des Hilfslemmas

Lemma. Sei $\Phi \subseteq \text{FO}[\sigma]$ und \mathcal{I}, \mathcal{J} σ -Interpretationen.

Wenn $\mathcal{I} \models \varphi$ gdw. $\mathcal{J} \models \varphi$ für alle $\varphi \in \Phi$,

dann $\mathcal{I} \models \psi$ gdw. $\mathcal{J} \models \psi$ für alle $\psi \in BK(\Phi)$.

Beweis. Für jedes $\theta \in \Phi$ führen wir eine Aussagenvariable X_θ ein.

Sei nun $\psi \in BK(\Phi)$. Wir übersetzen ψ in eine aussagenlogische Formel ψ_{AL} , indem wir jede Unterformel $\theta \in \Phi$ von ψ durch X_θ ersetzen.

Beispiel. Sei $\Phi := \{P(x), E(x, y), \exists z(E(x, z) \wedge E(z, y))\}$ und

$$\psi(x, y) := (P(x) \wedge E(x, y)) \vee (\neg P(x) \wedge \exists z(E(x, z) \wedge E(z, y))).$$

$$\text{Dann } \psi_{\text{AL}} := (X_{P(x)} \wedge X_{E(x, y)}) \vee (\neg X_{P(x)} \wedge X_{\exists z(E(x, z) \wedge E(z, y))})$$

Beweis des Hilfslemmas

Lemma. Sei $\Phi \subseteq \text{FO}[\sigma]$ und \mathcal{I}, \mathcal{J} σ -Interpretationen.

Wenn $\mathcal{I} \models \varphi$ gdw. $\mathcal{J} \models \varphi$ für alle $\varphi \in \Phi$,
dann $\mathcal{I} \models \psi$ gdw. $\mathcal{J} \models \psi$ für alle $\psi \in BK(\Phi)$.

Beweis. Für jedes $\theta \in \Phi$ führen wir eine Aussagenvariable X_θ ein.

Sei nun $\psi \in BK(\Phi)$. Wir übersetzen ψ in eine aussagenlogische Formel ψ_{AL} , indem wir jede Unterformel $\theta \in \Phi$ von ψ durch X_θ ersetzen.

\mathcal{I} und \mathcal{J} induzieren Belegungen $\beta_{\mathcal{I}}, \beta_{\mathcal{J}}$ für ψ_{AL} wie folgt:

$$\begin{aligned}\beta_{\mathcal{I}}(X_\theta) &:= 1 \text{ gdw. } \mathcal{I} \models \theta \\ \beta_{\mathcal{J}}(X_\theta) &:= 1 \text{ gdw. } \mathcal{J} \models \theta \quad \text{für alle } \theta \in \Phi.\end{aligned}$$

Nach Konstruktion gilt (*):

$$\begin{aligned}\mathcal{I} \models \psi \text{ gdw. } \beta_{\mathcal{I}} \models \psi_{\text{AL}} \text{ und} \\ \mathcal{J} \models \psi \text{ gdw. } \beta_{\mathcal{J}} \models \psi_{\text{AL}}.\end{aligned}$$

Da $\mathcal{I} \models \varphi$ gdw. $\mathcal{J} \models \varphi$ für alle $\varphi \in \Phi$, folgt $\beta_{\mathcal{I}} = \beta_{\mathcal{J}}$ und (wegen (*)):

$$\mathcal{I} \models \psi \text{ gdw. } \beta_{\mathcal{I}} \models \psi_{\text{AL}} \text{ gdw. } \beta_{\mathcal{J}} \models \psi_{\text{AL}} \text{ gdw. } \mathcal{J} \models \psi. \quad \square$$

Erweiterung des Lemmas

Lemma. Sei $\Phi \subseteq \text{FO}[\sigma]$ endlich. Dann gibt es nur endlich viele paarweise nicht-äquivalente Formeln in $BK(\Phi)$.

Beweis. Wie zuvor definieren wir

$$\begin{aligned} V &:= \{X_\varphi : \varphi \in \Phi\} \\ \psi \in BK(\Phi) &\rightsquigarrow \psi_{AL} \in AL \quad \text{ersetze Unterformeln } \theta \in \Phi \text{ durch } X_\theta \\ \sigma\text{-Interpretation } \mathcal{I} &\rightsquigarrow \text{mit Belegung } \beta_{\mathcal{I}} \quad \beta_{\mathcal{I}}(X_\theta) := 1 \text{ gdw. } \mathcal{I} \models \theta, \text{ für alle } \theta \in \Phi. \end{aligned}$$

Wie eben gilt für alle $\psi \in BK(\Phi)$ und σ -Interpretationen \mathcal{I} :

$$\mathcal{I} \models \psi \text{ gdw. } \beta_{\mathcal{I}} \models \psi_{AL}.$$

Falls also $\psi_{AL} \equiv \psi'_{AL}$, dann auch $\psi \equiv \psi'$.

Wir wissen bereits, dass es nur endlich viele paarweise nicht-äquivalente aussagenlogische Formeln über der Variablenmenge V gibt.

Also gibt es nur endlich viele paarweise nicht-äquivalente Formeln in $BK(\Phi)$. □

Die Zahl paarweise nicht-äquivalenter Formeln

Notation. Sei X eine Menge und \bar{y} ein Tupel.

Wir schreiben $\bar{y} \subseteq X$ als Abkürzung für „ $y_i \in X$ für alle i “.

Lemma. Sei σ eine endliche relationale Signatur.

Für alle $m, k \geq 0$ gibt es nur endlich viele paarweise nicht-äquivalente Formeln $\psi(x_1, \dots, x_k) \in \text{FO}[\sigma]$ vom Quantorenrang $\leq m$.

Beweis. Für alle $m, k \geq 0$ sei $\mathcal{L}_{m,k}$ eine maximale Menge paarweise nicht-äquivalenter Formeln $\psi(\bar{y})$ mit Quantorenrang $\leq m$ und $\bar{y} \subseteq \{x_1, \dots, x_k\}$.

Zu zeigen ist also, dass $\mathcal{L}_{m,k}$ für alle $m, k \geq 0$ endlich ist.

Wir führen den Beweis per Induktion über m .

Die Zahl paarweise nicht-äquivalenter Formeln

Lemma. Sei σ eine endliche relationale Signatur.

Für alle $m, k \geq 0$ gibt es nur endlich viele paarweise nicht-äquivalente Formeln $\psi(x_1, \dots, x_k) \in \text{FO}[\sigma]$ vom Quantorenrang $\leq m$.

Beweis. Für alle $m, k \geq 0$ sei $\mathcal{L}_{m,k}$ eine maximale Menge paarweise nicht-äquivalenter Formeln $\psi(\bar{y})$ mit Quantorenrang $\leq m$ und $\bar{y} \subseteq \{x_1, \dots, x_k\}$.

Zu zeigen ist also, dass $\mathcal{L}_{m,k}$ für alle $m, k \geq 0$ endlich ist.

Wir führen den Beweis per Induktion über m .

Induktionsverankerung. Sei $m = 0$ (und k beliebig).

Da σ endlich und relational ist, existieren nur endlich viele verschiedene atomare Formeln $\psi(y_1, \dots, y_r)$ mit $\bar{y} \subseteq \{x_1, \dots, x_k\}$.

Aus dem vorherigen Lemma folgt, dass es nur endlich viele paarweise nicht-äquivalente quantorenfreie Formeln $\psi(\bar{y})$ mit $\bar{y} \subseteq \{x_1, \dots, x_k\}$ gibt.

Beispiel. $\sigma := \{E, P\}$

$X = \{x_1, x_2\}$ ($k = 2$)

Atomare Formeln:

$E(x_1, x_2), E(x_1, x_1),$
 $E(x_2, x_1), E(x_2, x_2),$
 $P(x_1), P(x_2)$

Lemma. Sei $\Phi \subseteq \text{FO}[\sigma]$ endlich.
 Es gibt nur endlich viele paarweise nicht-äquivalente Formeln in $BK(\Phi)$.

Beweis des Lemmas

Induktionsvoraussetzung. Für alle $k \geq 0$ ist $\mathcal{L}_{m-1,k}$ endlich.

Induktionsschritt. Sei $k \geq 0$ beliebig. Wir müssen die Aussage nun für Formeln $\psi(x_1, \dots, x_k) \in \mathcal{L}_{m,k}$ mit Quantorenrang $\leq m$ beweisen.

Beobachtung. Formeln $\varphi(x_1, \dots, x_k)$ mit Quantorenrang m sind Boolesche Kombinationen von

- Formeln $\psi \in \mathcal{L}_{m-1,k}$ mit Quantorenrank $< m$ und
- Formeln der Form $\exists y\psi$ oder $\forall y\psi$, wobei $\text{qr}(\psi) < m$.

Beispiel. Sei $m = 2$.

$$\varphi := E(x_1, x_2) \vee \neg \exists x_1 \forall y P(x_1, y) \vee \exists z (\forall x_2 E(z, x_2) \vee \exists x_3 (P(x_3) \wedge E(z, x_2)))$$

$\mathcal{L}_{m,k}$: max. Menge paarweise nicht-äquiv. Formeln $\psi(x_1, \dots, x_k)$ mit $\text{qr}(\psi) \leq m$.

Beweis des Lemmas

Induktionsvoraussetzung. Für alle $k \geq 0$ ist $\mathcal{L}_{m-1,k}$ endlich.

Induktionsschritt. Sei $k \geq 0$ beliebig. Wir müssen die Aussage nun für Formeln $\psi(x_1, \dots, x_k) \in \mathcal{L}_{m,k}$ mit Quantorenrang $\leq m$ beweisen.

Konsequenz.

Da $\exists y\psi \equiv \exists x_{k+1}\psi[y/x_{k+1}]$ und $\forall y\psi \equiv \forall x_{k+1}\psi[y/x_{k+1}]$ für alle $y \notin X$, können wir O.B.d.A. annehmen, dass φ eine Bool. Komb. von Formeln

1. $\psi \in \mathcal{L}_{m-1,k}$ und Formeln
2. $\exists z\psi$ oder $\forall z\psi$ mit $z \in \{x_1, \dots, x_k, x_{k+1}\}$ und $\psi \in \mathcal{L}_{m-1,k+1}$.

Sei $\mathcal{L}' := \mathcal{L}_{m-1,k} \cup \{\exists z\psi, \forall z\psi : z \in \{x_1, \dots, x_{k+1}\}, \psi \in \mathcal{L}_{m-1,k+1}\}$.

Nach IV sind $\mathcal{L}_{m-1,k}$, $\mathcal{L}_{m-1,k+1}$ und daher auch \mathcal{L}' endlich.

Aus vorherigem Lemma folgt, dass $BK(\mathcal{L}')$ und somit $\mathcal{L}_{m,k}$ endlich ist. □

$\mathcal{L}_{m,k}$: max. Menge paarweise nicht-äquiv. Formeln $\psi(x_1, \dots, x_k)$ mit $qr(\psi) \leq m$.

Lemma. Sei $\Phi \subseteq FO[\sigma]$ endlich. Es gibt nur endlich viele paarweise nicht-äquivalente Formeln in $BK(\Phi)$.

Zusammenfassung

Quantorenrang. Max. Schachtelungstiefe der Quantoren.

Lemma. Sei σ eine endliche relationale Signatur.

Für alle $m, k \geq 0$ gibt es nur endlich viele paarweise nicht-äquivalente Formeln $\psi(x_1, \dots, x_k) \in \text{FO}[\sigma]$ vom Quantorenrang $\leq m$.

Definition. Sei $\Phi \subseteq \text{FO}[\sigma]$. Die Klasse $BK(\Phi)$ der Booleschen Kombinationen von Φ ist die kleinste Klasse für die gilt:

- $\Phi \subseteq BK(\Phi)$
- $(\varphi \wedge \psi), (\varphi \vee \psi), \neg \varphi \in BK(\Phi)$ für alle $\varphi, \psi \in BK(\Phi)$

Lemma. Sei $\Phi \subseteq \text{FO}[\sigma]$ und \mathcal{I}, \mathcal{J} σ -Interpretationen.

Wenn $\mathcal{I} \models \varphi$ gdw. $\mathcal{J} \models \varphi$ für alle $\varphi \in \Phi$,

dann $\mathcal{I} \models \psi$ gdw. $\mathcal{J} \models \psi$ für alle $\psi \in BK(\Phi)$.

Lemma. Sei $\Phi \subseteq \text{FO}[\sigma]$ endlich. Dann gibt es nur endlich viele paarweise nicht-äquivalente Formeln in $BK(\Phi)$.

10.3 Elementare Äquivalenz

Wiederholung: Erreichbarkeit in der Prädikatenlogik

Signatur. $\sigma := \{E, s, t\}$ E 2-stelliges Rel.symb, s, t Konstantensymb.

Frage. Ist folgende Klasse FO-definierbar?

$\text{REACH} := \{\mathcal{A} : \mathcal{A} \text{ } \sigma\text{-Struktur, es gibt einen Pfad von } s^{\mathcal{A}} \text{ nach } t^{\mathcal{A}}\}.$

Versuch einer Antwort.

1. Direktflug: $\varphi_1 := E(s, t)$
2. Ein Stopp : $\varphi_2 := \exists x_1 (E(s, x_1) \wedge E(x_1, t))$
3. Zwei Stopps : $\varphi_3 := \exists x_1 \exists x_2 (E(s, x_1) \wedge E(x_1, x_2) \wedge E(x_2, t))$

Beobachtung. Für jede feste Zahl von Stopps existiert eine Formel.

Erreichbarkeit in der Prädikatenlogik

Frage. Ist folgende Klasse FO-definierbar?

REACH := $\{\mathcal{A} : \mathcal{A} \text{ } \sigma\text{-Struktur, es gibt einen Pfad von } s^{\mathcal{A}} \text{ nach } t^{\mathcal{A}}\}$.

Behauptung. Es gibt keinen Satz der Prädikatenlogik, der **REACH** definiert.

In anderen Worten.

Es existiert kein $\varphi \in \text{FO}[\sigma]$, so dass für alle $\mathcal{A} \in \text{REACH}$ gilt: $\mathcal{A} \models \varphi$ und für alle $\mathcal{B} \notin \text{REACH}$ gilt: $\mathcal{B} \not\models \varphi$.

für alle $\varphi \in \text{FO}[\sigma]$ gibt es ein $\mathcal{A} \in \text{REACH}$ mit $\mathcal{A} \not\models \varphi$ oder es existiert ein $\mathcal{B} \notin \text{REACH}$ mit $\mathcal{B} \models \varphi$.

Beweisversuch. Wir wollen zeigen, dass es für jeden Satz $\varphi \in \text{FO}[\sigma]$ zwei σ -Strukturen \mathcal{A} und \mathcal{B} gibt, so dass

1. $\mathcal{A} \in \text{REACH}, \mathcal{B} \notin \text{REACH}$ (d.h. in \mathcal{A} ex. ein Weg von s nach t , in \mathcal{B} aber nicht.)
2. $\mathcal{A} \models \varphi$ und $\mathcal{B} \models \varphi$ oder aber $\mathcal{A} \not\models \varphi$ und $\mathcal{B} \not\models \varphi$

Erreichbarkeit in der Prädikatenlogik

Behauptung. Kein FO-Satz definiert

$\text{REACH} := \{\mathcal{A} : \mathcal{A} \text{ } \sigma\text{-Struktur, es gibt einen Pfad von } s^{\mathcal{A}} \text{ nach } t^{\mathcal{A}}\}.$

Beweisversuch. Zeige, dass es für alle $\varphi \in \text{FO}[\sigma]$ σ -Strukturen \mathcal{A}, \mathcal{B} gibt, mit

1. $\mathcal{A} \in \text{REACH}, \mathcal{B} \notin \text{REACH}$ (in \mathcal{A} ex. Weg von s nach t , in \mathcal{B} nicht.)
2. $\mathcal{A} \models \varphi$ und $\mathcal{B} \models \varphi$ oder aber $\mathcal{A} \not\models \varphi$ und $\mathcal{B} \not\models \varphi$

Problem. Wir müssen für jedes $\varphi \in \text{FO}[\sigma]$ neue Strukturen finden.

Besser. Zeige: es gibt σ -Strukturen \mathcal{A}, \mathcal{B} , so dass für alle $\varphi \in \text{FO}[\sigma]$

1. $\mathcal{A} \in \text{REACH}, \mathcal{B} \notin \text{REACH}$ (in \mathcal{A} ex. Weg von s nach t , in \mathcal{B} nicht.)
2. $\mathcal{A} \models \varphi$ und $\mathcal{B} \models \varphi$ oder aber $\mathcal{A} \not\models \varphi$ und $\mathcal{B} \not\models \varphi$

Elementare Äquivalenz

Definition. Zwei σ -Strukturen \mathcal{A}, \mathcal{B} sind *elementar äquivalent*, geschrieben $\mathcal{A} \equiv \mathcal{B}$, wenn für alle σ -Sätze ψ gilt:

$$\mathcal{A} \models \psi \iff \mathcal{B} \models \psi$$

Zeige. Es ex. $\mathcal{A} \in \text{Reach}$, $\mathcal{B} \notin \text{Reach}$
s.d. für alle φ :
 $\mathcal{A} \models \varphi$, $\mathcal{B} \models \varphi$ oder $\mathcal{A} \not\models \varphi$, $\mathcal{B} \not\models \varphi$.

Besser. Zeige: es gibt σ -Strukturen \mathcal{A}, \mathcal{B} , so dass für alle $\varphi \in \text{FO}[\sigma]$

1. $\mathcal{A} \in \text{REACH}$, $\mathcal{B} \notin \text{REACH}$ (in \mathcal{A} ex. Weg von s nach t , in \mathcal{B} nicht.)
2. $\mathcal{A} \models \varphi$ und $\mathcal{B} \models \varphi$ oder aber $\mathcal{A} \not\models \varphi$ und $\mathcal{B} \not\models \varphi$
3. $\mathcal{A} \equiv \mathcal{B}$.

Problem. Solche Strukturen \mathcal{A}, \mathcal{B} kann es nicht geben.

Nach Voraussetzung enthält \mathcal{A} einen $s^{\mathcal{A}} - t^{\mathcal{A}}$ -Pfad

$$P = (s = v_0, v_1, \dots, v_n = t).$$

Also $\mathcal{A} \models \psi_k := \exists x_0 \exists x_1 \dots \exists x_n (x_0 = s \wedge x_n = t \wedge \bigwedge_{1 \leq i < n} E(x_i, x_{i+1}))$.

Aus $\mathcal{A} \equiv \mathcal{B}$ folgt $\mathcal{B} \models \psi_k$ und daher existiert auch in \mathcal{B} ein $s^{\mathcal{B}} - t^{\mathcal{B}}$ -Pfad.

m -Äquivalenz

Erinnerung. Der Quantorenrang $\text{qr}(\psi)$ einer Formel $\psi \in \text{FO}$ ist die maximale Schachtelungstiefe der Quantoren.

Definition. Sei $m \in \mathbb{N}$.

Zwei σ -Strukturen \mathcal{A}, \mathcal{B} sind m -äquivalent, geschrieben $\mathcal{A} \equiv_m \mathcal{B}$, wenn für alle σ -Sätze ψ mit Quantorenrang $\text{qr}(\psi) \leq m$ gilt:

$$\mathcal{A} \models \psi \iff \mathcal{B} \models \psi$$

Beobachtung. $\mathcal{A} \equiv \mathcal{B}$ genau dann, wenn $\mathcal{A} \equiv_m \mathcal{B}$ für alle $m \in \mathbb{N}$.

m-Äquivalenz

Definierbarkeit in der Prädikatenlogik. *m*-Äquivalenz eignet sich oft besser als elementare Äquivalenz zum Beweis der Nicht-Definierbarkeit bestimmter Aussagen.

Behauptung. Kein FO-Satz definiert

$$\text{REACH} := \{\mathcal{A} : \mathcal{A} \text{ } \sigma\text{-Struktur, es gibt einen Pfad von } s^{\mathcal{A}} \text{ nach } t^{\mathcal{A}}\}.$$

Beweisansatz. Es reicht, für alle $m \geq 0$ zwei σ -Strukturen

$\mathcal{A}_m, \mathcal{B}_m$ zu finden, so dass

- $\mathcal{A}_m \in \text{REACH}$ aber $\mathcal{B}_m \notin \text{REACH}$ und
- $\mathcal{A}_m \equiv_m \mathcal{B}_m$

Definition. Sei $m \in \mathbb{N}$.

\mathcal{A}, \mathcal{B} *m*-äquivalent, $\mathcal{A} \equiv_m \mathcal{B}$, wenn $\mathcal{A} \models \psi \iff \mathcal{B} \models \psi$ für alle ψ mit $\text{qr}(\psi) \leq m$.

m-Äquivalenz und Definierbarkeit

Lemma. Sei σ eine Signatur und \mathcal{C}, \mathcal{K} Klassen von σ -Strukturen.

Wenn es für alle $m \geq 1$ σ -Strukturen $\mathcal{A}_m, \mathcal{B}_m \in \mathcal{K}$ gilt, so dass

- $\mathcal{A}_m \in \mathcal{C}$ aber $\mathcal{B}_m \notin \mathcal{C}$
- $\mathcal{A}_m \equiv_m \mathcal{B}_m$,

dann gibt es keinen Satz $\varphi \in \text{FO}[\sigma]$ der \mathcal{C} in \mathcal{K} definiert.

Beweis (durch Widerspruch).

Ang., es gäbe $\varphi \in \text{FO}[\sigma]$ mit $\text{Mod}_{\mathcal{K}}(\varphi) = \mathcal{C}$.

Sei $m := \text{qr}(\varphi)$. Betrachte $\mathcal{A}_m, \mathcal{B}_m$.

Nach Voraussetzung gilt $\mathcal{A}_m \in \mathcal{C}$ und somit $\mathcal{A}_m \models \varphi$.

Da aber $\mathcal{A}_m \equiv_m \mathcal{B}_m$, gilt auch $\mathcal{B}_m \models \varphi$.

Widerspruch zu $\mathcal{B}_m \notin \mathcal{C}$. □

Definition. \mathcal{K} Klasse von σ -Strukturen.

Klasse $\mathcal{C} \subseteq \mathcal{K}$ ist in \mathcal{K} FO-definierbar, wenn es $\psi \in \text{FO}[\sigma]$ gibt, so dass

$$\mathcal{C} = \{\mathcal{A} \in \mathcal{K} : \mathfrak{A} \models \psi\}.$$

$$\text{Mod}_{\mathcal{K}}(\varphi) := \{\mathcal{A} \in \mathcal{K} : \mathcal{A} \models \varphi\}.$$

m-Äquivalenz

Wir erweitern die elementare und *m*-Äquivalenz noch auf Strukturen mit ausgezeichneten Elementen und Formeln mit freien Variablen.

Definition. Seien \mathcal{A}, \mathcal{B} σ -Strukturen und $\bar{a} \in A^k, \bar{b} \in B^k$.

1. (\mathcal{A}, \bar{a}) und (\mathcal{B}, \bar{b}) sind ***m*-äquivalent**, geschrieben $(\mathcal{A}, \bar{a}) \equiv_m (\mathcal{B}, \bar{b})$, wenn für alle σ -Formeln $\psi(\bar{x})$ mit Quantorenrang $\text{qr}(\psi) \leq m$ und freien Variablen $\bar{x} := x_1, \dots, x_k$ gilt:

$$\mathcal{A} \models \psi[\bar{a}] \iff \mathcal{B} \models \psi[\bar{b}].$$

2. (\mathcal{A}, \bar{a}) und (\mathcal{B}, \bar{b}) sind **elementar äquivalent**, geschrieben $(\mathcal{A}, \bar{a}) \equiv (\mathcal{B}, \bar{b})$, wenn für alle σ -Formeln $\psi(\bar{x})$ und freien Variablen $\bar{x} := x_1, \dots, x_k$ gilt:

$$\mathcal{A} \models \psi[\bar{a}] \iff \mathcal{B} \models \psi[\bar{b}].$$

m-Äquivalenz und Definierbarkeit

Lemma. Sei σ eine Signatur und \mathcal{C}, \mathcal{K} Klassen von σ -Strukturen.

Wenn es für alle $m \geq 1$ σ -Strukturen $\mathcal{A}_m, \mathcal{B}_m \in \mathcal{K}$ gibt, so dass

- $\mathcal{A}_m \in \mathcal{C}$ aber $\mathcal{B}_m \notin \mathcal{C}$
- $\mathcal{A}_m \equiv_m \mathcal{B}_m$,

dann gibt es keinen Satz $\varphi \in \text{FO}[\sigma]$ der \mathcal{C} in \mathcal{K} definiert.

Frage. Wie kann man denn zeigen, dass $\mathcal{A} \equiv_m \mathcal{B}$?

Definition. Sei $m \in \mathbb{N}$.

\mathcal{A}, \mathcal{B} *m*-äquivalent, $\mathcal{A} \equiv_m \mathcal{B}$,
wenn $\mathcal{A} \models \psi \iff \mathcal{B} \models \psi$
für alle ψ mit $\text{qr}(\psi) \leq m$.

Definition. \mathcal{K} Klasse von σ -Strukturen.

Klasse $\mathcal{C} \subseteq \mathcal{K}$ ist in \mathcal{K} FO-definierbar,
wenn es $\psi \in \text{FO}[\sigma]$ gibt, so dass

$$\mathcal{C} = \{\mathcal{A} \in \mathcal{K} : \mathfrak{A} \models \psi\}.$$

$$\text{Mod}_{\mathcal{K}}(\varphi) := \{\mathcal{A} \in \mathcal{K} : \mathcal{A} \models \varphi\}.$$

10.4 Partielle Isomorphismen

Wiederholung: m -Äquivalenz

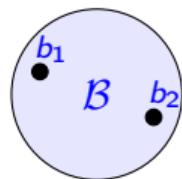
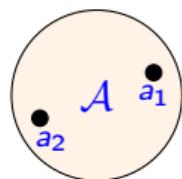
Definition. Seien \mathcal{A}, \mathcal{B} σ -Strukturen und $\bar{a} \in A^k, \bar{b} \in B^k$.

1. (\mathcal{A}, \bar{a}) und (\mathcal{B}, \bar{b}) sind **m -äquivalent**, geschrieben $(\mathcal{A}, \bar{a}) \equiv_m (\mathcal{B}, \bar{b})$, wenn für alle σ -Formeln $\psi(\bar{x})$ mit Quantorenrang $\text{qr}(\psi) \leq m$ und freien Variablen $\bar{x} := x_1, \dots, x_k$ gilt:

$$\mathcal{A} \models \psi[\bar{a}] \iff \mathcal{B} \models \psi[\bar{b}].$$

2. (\mathcal{A}, \bar{a}) und (\mathcal{B}, \bar{b}) sind **elementar äquivalent**, geschrieben $(\mathcal{A}, \bar{a}) \equiv (\mathcal{B}, \bar{b})$, wenn für alle σ -Formeln $\psi(\bar{x})$ und freien Variablen $\bar{x} := x_1, \dots, x_k$ gilt: $\mathcal{A} \models \psi[\bar{a}] \iff \mathcal{B} \models \psi[\bar{b}]$.

Notation. Für $k = 0$ schreiben wir kurz $\mathcal{A} \equiv \mathcal{B}$ und $\mathcal{A} \equiv_m \mathcal{B}$.



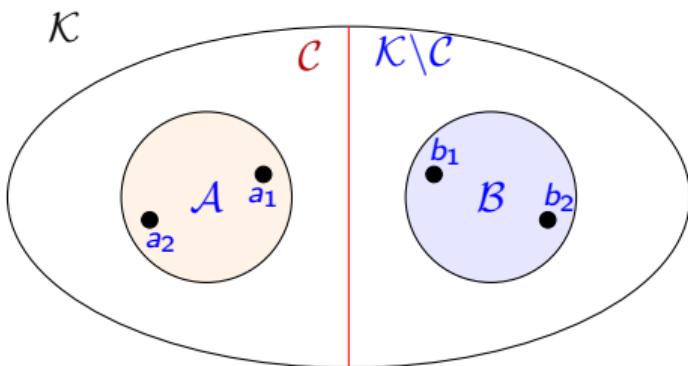
Wiederholung: m -Äquivalenz und Definierbarkeit

Lemma. Sei σ eine Signatur und \mathcal{C}, \mathcal{K} Klassen von σ -Strukturen.

Wenn es für alle $m \geq 0$ σ -Strukturen $\mathcal{A}_m, \mathcal{B}_m \in \mathcal{K}$ gibt, so dass

- $\mathcal{A}_m \in \mathcal{C}$ aber $\mathcal{B}_m \notin \mathcal{C}$
- $\mathcal{A}_m \equiv_m \mathcal{B}_m$,

dann gibt es keinen Satz $\varphi \in \text{FO}[\sigma]$ der \mathcal{C} in \mathcal{K} definiert.



Frage. Wie kann man denn zeigen, dass $\mathcal{A} \equiv_m \mathcal{B}$?

Definition. Sei $m \in \mathbb{N}$ und

$\bar{a} \in A^k, \bar{b} \in B^k$.

$(\mathcal{A}, \bar{a}) \equiv_m (\mathcal{B}, \bar{b})$, wenn

$\mathcal{A} \models \psi[\bar{a}] \iff \mathcal{B} \models \psi[\bar{b}]$
für alle $\psi(\bar{x})$ mit $\text{qr}(\psi) \leq m$.

Definition. \mathcal{K} Klasse von σ -Strukturen.

Klasse $\mathcal{C} \subseteq \mathcal{K}$ ist in \mathcal{K} FO-definierbar, wenn es $\psi \in \text{FO}[\sigma]$ gibt, so dass

$$\mathcal{C} = \{\mathcal{A} \in \mathcal{K} : \mathcal{A} \models \psi\}.$$

$$\text{Mod}_{\mathcal{K}}(\varphi) := \{\mathcal{A} \in \mathcal{K} : \mathcal{A} \models \varphi\}.$$

Partielle Isomorphismen

Frage. Wie kann man denn zeigen, dass $\mathcal{A} \equiv_m \mathcal{B}$?

Antwort für $m = 0$. Partielle Isomorphismen.

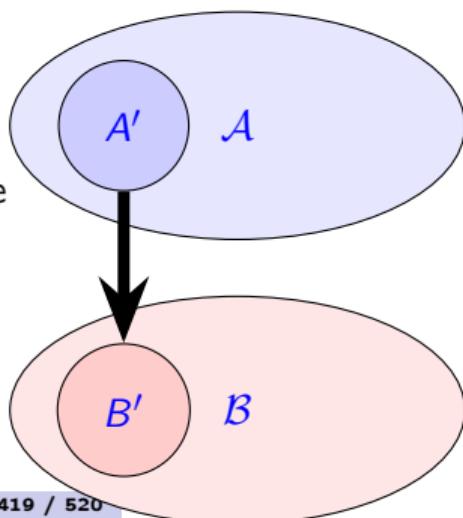
Vereinbarung. Zur Vereinfachung der Notation betrachten wir in diesem Abschnitt nur **relationale Signaturen**, d.h. Signaturen, in denen nur Relationssymbole vorkommen.

Definition. Sei σ eine (relationale) Signatur.

Ein **partieller Isomorphismus** zwischen zwei σ -Strukturen \mathcal{A}, \mathcal{B} ist eine injektive Abbildung $h : A' \rightarrow B$, für ein $A' \subseteq A$, so dass für alle $R \in \sigma \cup \{=\}$ und alle $a_1, \dots, a_k \in A'$, wobei $k = ar(R)$,

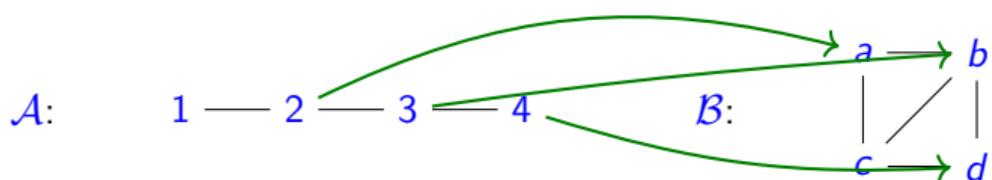
$$(a_1, \dots, a_k) \in R^A \quad \text{gdw.} \quad (h(a_1), \dots, h(a_k)) \in R^B.$$

Isomorphismus $h : \mathcal{A} \cong \mathcal{B}$.
 $h : A \rightarrow B$ bijektiv, so dass
für alle $R \in \sigma$ mit $k = ar(R)$
und $a_1, \dots, a_k \in A^k$ gilt:
 $(a_1, \dots, a_k) \in R^A$
gdw.
 $(h(a_1), \dots, h(a_k)) \in R^B$.



Beispiele zu partiellen Isomorphismen

Sei $\sigma := \{E\}$ und seien \mathcal{A}, \mathcal{B} wie folgt gegeben:

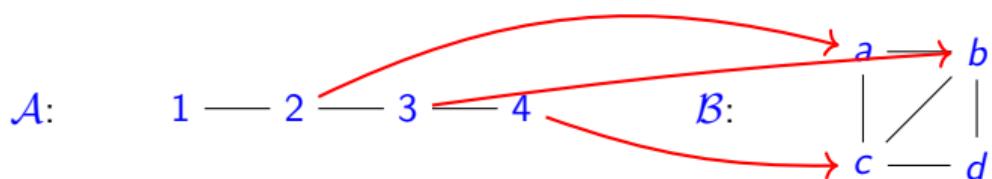


Die Abbildung $\pi : 2 \mapsto a, 3 \mapsto b, 4 \mapsto d$ ein partieller Isomorphismus zwischen \mathcal{A} und \mathcal{B} .

Partieller Isomorphismus h .
 $h : A' \subseteq A \rightarrow B' \subseteq B$ injektiv,
so dass
für alle $R \in \sigma \cup \{=\}$ mit
 $k = ar(R)$
und $a_1, \dots, a_k \in A'^k$ gilt:
 $(a_1, \dots, a_k) \in R^A$
gdw.
 $(h(a_1), \dots, h(a_k)) \in R^B$.

Beispiele zu partiellen Isomorphismen

Sei $\sigma := \{E\}$ und seien \mathcal{A}, \mathcal{B} wie folgt gegeben:



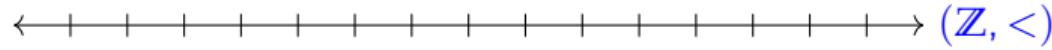
Die Abbildung $\pi : 2 \mapsto a, 3 \mapsto b, 4 \mapsto d$ ein partieller Isomorphismus zwischen \mathcal{A} und \mathcal{B} .

Die Abbildung definiert durch $\pi : 2 \mapsto a, 3 \mapsto b, 4 \mapsto c$ ist jedoch kein partieller Isomorphismus zwischen \mathcal{A} und \mathcal{B} .

Partieller Isomorphismus h .
 $h : A' \subseteq A \rightarrow B' \subseteq B$ injektiv,
so dass
für alle $R \in \sigma \cup \{=\}$ mit
 $k = ar(R)$
und $a_1, \dots, a_k \in A'^k$ gilt:
 $(a_1, \dots, a_k) \in R^A$
gdw.
 $(h(a_1), \dots, h(a_k)) \in R^B$.

Beispiele zu partiellen Isomorphismen

Sei $\sigma := \{<\}$, $\mathcal{A} := (\mathbb{N}, <^{\mathcal{A}})$ und $\mathcal{B} := (\mathbb{Z}, <^{\mathcal{B}})$.



Frage. Was sind die partiellen Isomorphismen zwischen \mathcal{A} und \mathcal{B} ?

Antwort. Alle Abbildungen $\pi : \mathbb{N} \rightarrow \mathbb{Z}$ die **ordnungserhaltend** sind.

D.h. wenn π die Menge $A' \subseteq \mathbb{N}$ auf $B' \subseteq \mathbb{Z}$ abbildet und

$$a = a_1 < a_2 < \cdots < a_n$$

dann gilt

$$\pi(a_1) < \pi(a_2) < \cdots < \pi(a_n).$$

0-Äquivalenz

Lemma. Sei σ eine relationale Signatur, \mathcal{A}, \mathcal{B} σ -Strukturen,
 $\bar{a} := a_1, \dots, a_k \in A^k$ und $\bar{b} := b_1, \dots, b_k \in B^k$.

Dann sind folgende Aussagen äquivalent:

1. Die Abbildung

$$h : \{a_1, \dots, a_k\} \rightarrow \{b_1, \dots, b_k\}$$

$$a_i \mapsto b_i \quad \text{für alle } 1 \leq i \leq k$$

ist ein partieller Isomorphismus.

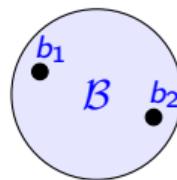
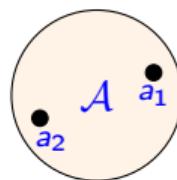
2. Für alle atomaren Formeln $\psi(x_1, \dots, x_k)$ gilt:

$$\mathcal{A} \models \psi[\bar{a}] \quad \text{gdw.} \quad \mathcal{B} \models \psi[\bar{b}]$$

3. Für alle quantorenfreien Formeln $\psi(x_1, \dots, x_k)$ gilt:

Def. $\equiv_0 \quad \mathcal{A} \models \psi[\bar{a}] \quad \text{gdw.} \quad \mathcal{B} \models \psi[\bar{b}]$

4. $(\mathcal{A}, \bar{a}) \equiv_0 (\mathcal{B}, \bar{b})$



Beweis (1) \Rightarrow (2)

Voraussetzung. $\bar{a} \in A^k, \bar{b} \in B^k$ und $h : \{a_1, \dots, a_k\} \rightarrow \{b_1, \dots, b_k\}$
 mit $h(a_i) := b_i$ für alle $1 \leq i \leq k$ ist ein partieller Isomorphismus.

Sei $\psi(x_1, \dots, x_k)$ atomar. Zu zeigen: $\mathcal{A} \models \psi[\bar{a}]$ gdw. $\mathcal{B} \models \psi[\bar{b}]$.

Da ψ atomar gilt $\psi := R(x_{i_1}, \dots, x_{i_j})$ oder $\psi := x_i = x_j$.

Wir betrachten hier den Fall $\psi := R(x_{i_1}, \dots, x_{i_j})$.

Es gilt

$\mathcal{A} \models \psi[\bar{a}]$ gdw. $(a_{i_1}, \dots, a_{i_j}) \in R^{\mathcal{A}}$ Semantik von FO

gdw. $(b_{i_1}, \dots, b_{i_j}) \in R^{\mathcal{B}}$ Definition partieller Isomorphismen

gdw. $\mathcal{B} \models \psi[\bar{b}]$.

Lemma.

Folgende Aussagen äquivalent:

1. $h : A' \rightarrow B'$ mit $h(a_i) = b_i$ ist partieller Isom.
2. $\psi(x_1, \dots, x_k)$ atomar:
 $\mathcal{A} \models \psi[\bar{a}]$ gdw. $\mathcal{B} \models \psi[\bar{b}]$
3. $\psi(x_1, \dots, x_k)$ quantorenfrei:
 $\mathcal{A} \models \psi[\bar{a}]$ gdw. $\mathcal{B} \models \psi[\bar{b}]$
4. $(\mathcal{A}, \bar{a}) \equiv_0 (\mathcal{B}, \bar{b})$

Beweis (2) \Rightarrow (3)

Voraussetzung. Für alle atomaren Formeln $\psi(x_1, \dots, x_k)$ gilt
 $\mathcal{A} \models \psi[\bar{a}]$ gdw. $\mathcal{B} \models \psi[\bar{b}]$.

Zu Zeigen. Die gleiche Aussage gilt für alle quantorenfreien Formeln.

Beweis. Die Aussage folgt sofort aus Lemma BK.

Lemma.

Folgende Aussagen äquivalent:

1. $h : A' \rightarrow B'$ mit $h(a_i) = b_i$ ist partieller Isom.
2. $\psi(x_1, \dots, x_k)$ atomar:
 $\mathcal{A} \models \psi[\bar{a}]$ gdw. $\mathcal{B} \models \psi[\bar{b}]$
3. $\psi(x_1, \dots, x_k)$ quantorenfrei:
 $\mathcal{A} \models \psi[\bar{a}]$ gdw. $\mathcal{B} \models \psi[\bar{b}]$
4. $(\mathcal{A}, \bar{a}) \equiv_0 (\mathcal{B}, \bar{b})$

Lemma BK. Sei $\Phi \subseteq \text{FO}[\sigma]$ und seien \mathcal{I}, \mathcal{J} σ -Interpretationen.

Wenn

$\mathcal{I} \models \varphi$ gdw. $\mathcal{J} \models \varphi$ f.a. $\varphi \in \Phi$,

dann

$\mathcal{I} \models \psi$ gdw. $\mathcal{J} \models \psi$ f.a. $\psi \in BK(\Phi)$.

Beweis (3) \Rightarrow (1)

Voraussetzung. Für alle quantorenfreien Formeln $\psi(\bar{x})$ gilt

$$\mathcal{A} \models \psi[\bar{a}] \text{ gdw. } \mathcal{B} \models \psi[\bar{b}].$$

Zu Zeigen. Abb. h mit $h(a_i) = b_i$, $1 \leq i \leq k$, ist part. Isomorphismus.

Injektivität von h : wenn $a_i \neq a_j$, dann $h(a_i) \neq h(a_j)$.

Sei also $i < j$ mit $a_i \neq a_j$.

Dann gilt $\mathcal{A} \models (\neg x_i = x_j)[\bar{a}]$.

Aus der Voraussetzung folgt daher $\mathcal{B} \models (\neg x_i = x_j)[\bar{b}]$ und somit

$$b_i \neq b_j \quad \text{d.h.} \quad h(a_i) \neq h(a_j).$$

Lemma.

Folgende Aussagen äquivalent:

1. $h : A' \rightarrow B'$ mit $h(a_i) = b_i$ ist partieller Isom.
2. $\psi(x_1, \dots, x_k)$ atomar:
 $\mathcal{A} \models \psi[\bar{a}] \text{ gdw. } \mathcal{B} \models \psi[\bar{b}]$
3. $\psi(x_1, \dots, x_k)$ quantorenfrei:
 $\mathcal{A} \models \psi[\bar{a}] \text{ gdw. } \mathcal{B} \models \psi[\bar{b}]$
4. $(\mathcal{A}, \bar{a}) \equiv_0 (\mathcal{B}, \bar{b})$

Partieller Isomorphismus h .

$h : A' \subseteq A \rightarrow B' \subseteq B$ injektiv,
so dass

für alle $R \in \sigma \cup \{=\}$ mit
 $k = ar(R)$

und $a_1, \dots, a_k \in A'^k$ gilt:

$$(a_1, \dots, a_k) \in R^A \quad \text{gdw.} \\ (h(a_1), \dots, h(a_k)) \in R^B.$$

Beweis (3) \Rightarrow (1)

Voraussetzung. Für alle quantorenfreien Formeln $\psi(\bar{x})$ gilt

$$\mathcal{A} \models \psi[\bar{a}] \text{ gdw. } \mathcal{B} \models \psi[\bar{b}].$$

Zu Zeigen. Abb. h mit $h(a_i) = b_i$, $1 \leq i \leq k$, ist part. Isomorphismus.

Injectivität von h : wenn $a_i \neq a_j$, dann $h(a_i) \neq h(a_j)$.

Relationserhaltung: für alle r -stelligen $R \in \sigma \cup \{=\}$ und $1 \leq i_1, \dots, i_r$ gilt:

$$(a_{i_1}, \dots, a_{i_r}) \in R^{\mathcal{A}} \text{ gdw. } (b_{i_1}, \dots, b_{i_r}) \in R^{\mathcal{B}}.$$

Sei also $R \in \sigma$ und $1 \leq i_1, \dots, i_r \leq k$ wie zuvor. Es gilt:

$$(a_{i_1}, \dots, a_{i_r}) \in R^{\mathcal{A}} \quad \text{gdw. } \mathcal{A} \models R(x_{i_1}, \dots, x_{i_r})[\bar{a}]$$

$$\text{gdw. } \mathcal{B} \models R(x_{i_1}, \dots, x_{i_r})[\bar{b}]$$

$$\text{gdw. } (b_{i_1}, \dots, b_{i_r}) \in R^{\mathcal{B}}.$$

□

Lemma.

Folgende Aussagen äquivalent:

1. $h : A' \rightarrow B'$ mit $h(a_i) = b_i$ ist partieller Isom.
 2. $\psi(x_1, \dots, x_k)$ atomar:
- $$\mathcal{A} \models \psi[\bar{a}] \text{ gdw. } \mathcal{B} \models \psi[\bar{b}]$$
3. $\psi(x_1, \dots, x_k)$ quantorenfrei:
- $$\mathcal{A} \models \psi[\bar{a}] \text{ gdw. } \mathcal{B} \models \psi[\bar{b}]$$
4. $(\mathcal{A}, \bar{a}) \equiv_0 (\mathcal{B}, \bar{b})$

Partieller Isomorphismus h .

$h : A' \subseteq A \rightarrow B' \subseteq B$ injektiv, so dass

für alle $R \in \sigma \cup \{=\}$ mit $k = ar(R)$

und $a_1, \dots, a_k \in A'^k$ gilt:

$$(a_1, \dots, a_k) \in R^{\mathcal{A}} \text{ gdw. }$$

$$(h(a_1), \dots, h(a_k)) \in R^{\mathcal{B}}.$$

0-Äquivalenz

Lemma. Sei σ eine relationale Signatur, \mathcal{A}, \mathcal{B} σ -Strukturen,
 $\bar{a} := a_1, \dots, a_k \in A^k$ und $\bar{b} := b_1, \dots, b_k \in B^k$.

Dann sind folgende Aussagen äquivalent:

1. Die Abbildung

$$h : \{a_1, \dots, a_k\} \rightarrow \{b_1, \dots, b_k\}$$

$$a_i \mapsto b_i \quad \text{für alle } 1 \leq i \leq k$$

ist ein partieller Isomorphismus.

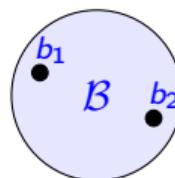
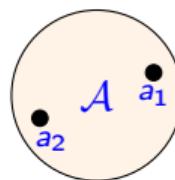
2. Für alle atomaren Formeln $\psi(x_1, \dots, x_k)$ gilt:

$$\mathcal{A} \models \psi[\bar{a}] \quad \text{gdw.} \quad \mathcal{B} \models \psi[\bar{b}]$$

3. Für alle quantorenfreien Formeln $\psi(x_1, \dots, x_k)$ gilt:

Def. $\equiv_0 \quad \mathcal{A} \models \psi[\bar{a}] \quad \text{gdw.} \quad \mathcal{B} \models \psi[\bar{b}]$

4. $(\mathcal{A}, \bar{a}) \equiv_0 (\mathcal{B}, \bar{b})$



Partielle Isomorphismen

Frage. Wie kann man denn zeigen, dass $\mathcal{A} \equiv_m \mathcal{B}$?

Antwort für $m = 0$. Partielle Isomorphismen.

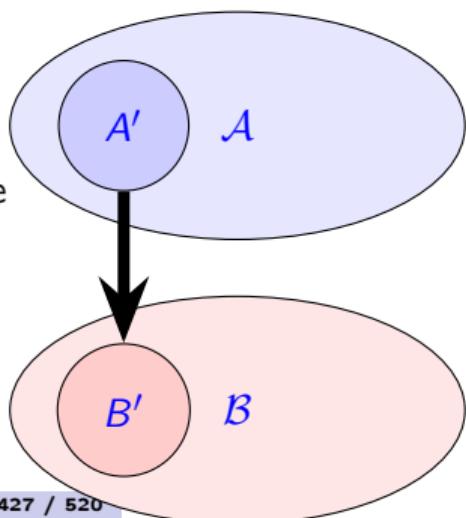
Vereinbarung. Zur Vereinfachung der Notation betrachten wir in diesem Abschnitt nur **relationale Signaturen**, d.h. Signaturen, in denen nur Relationssymbole vorkommen.

Definition. Sei σ eine (relationale) Signatur.

Ein **partieller Isomorphismus** zwischen zwei σ -Strukturen \mathcal{A}, \mathcal{B} ist eine injektive Abbildung $h : A' \rightarrow B$, für ein $A' \subseteq A$, so dass für alle $R \in \sigma \cup \{=\}$ und alle $a_1, \dots, a_k \in A'$, wobei $k = ar(R)$,

$$(a_1, \dots, a_k) \in R^A \quad \text{gdw.} \quad (h(a_1), \dots, h(a_k)) \in R^B.$$

Isomorphismus $h : \mathcal{A} \cong \mathcal{B}$.
 $h : A \rightarrow B$ bijektiv, so dass
für alle $R \in \sigma$ mit $k = ar(R)$
und $a_1, \dots, a_k \in A^k$ gilt:
 $(a_1, \dots, a_k) \in R^A$
gdw.
 $(h(a_1), \dots, h(a_k)) \in R^B$.



10.5 Ehrenfeucht-Fraïssé Spiele

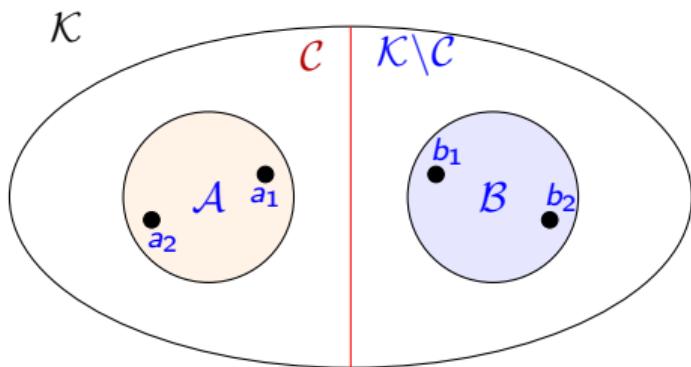
Wiederholung: m -Äquivalenz und Definierbarkeit

Lemma. Sei σ eine Signatur und \mathcal{C}, \mathcal{K} Klassen von σ -Strukturen.

Wenn es für alle $m \geq 1$ σ -Strukturen $\mathcal{A}_m, \mathcal{B}_m \in \mathcal{K}$ gibt, so dass

- $\mathcal{A}_m \in \mathcal{C}$ aber $\mathcal{B}_m \notin \mathcal{C}$
- $\mathcal{A}_m \equiv_m \mathcal{B}_m$,

dann gibt es keinen Satz $\varphi \in \text{FO}[\sigma]$ der \mathcal{C} in \mathcal{K} definiert.



Frage. Wie kann man denn zeigen, dass $\mathcal{A} \equiv_m \mathcal{B}$?

Definition. Sei $m \in \mathbb{N}$ und

$\bar{a} \in A^k, \bar{b} \in B^k$.

$(\mathcal{A}, \bar{a}) \equiv_m (\mathcal{B}, \bar{b})$, wenn

$\mathcal{A} \models \psi[\bar{a}] \iff \mathcal{B} \models \psi[\bar{b}]$
für alle $\psi(\bar{x})$ mit $\text{qr}(\psi) \leq m$.

Definition. \mathcal{K} Klasse von σ -Strukturen.

Klasse $\mathcal{C} \subseteq \mathcal{K}$ ist in \mathcal{K} FO-definierbar, wenn es $\psi \in \text{FO}[\sigma]$ gibt, so dass

$$\mathcal{C} = \{\mathcal{A} \in \mathcal{K} : \mathcal{A} \models \psi\}.$$

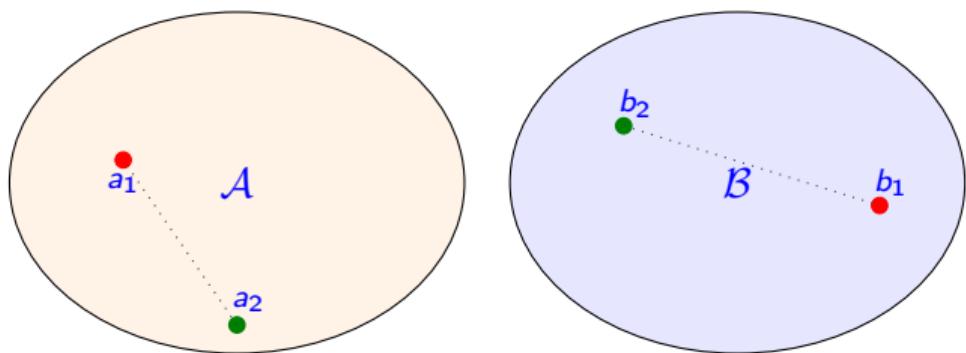
$$\text{Mod}_{\mathcal{K}}(\varphi) := \{\mathcal{A} \in \mathcal{K} : \mathcal{A} \models \varphi\}.$$

Partielle Isomorphismen

Frage. Wie kann man denn zeigen, dass $\mathcal{A} \equiv_m \mathcal{B}$?

Antwort für $m = 0$. Partielle Isomorphismen.

Antwort für $m > 0$?



Behauptung. $\mathcal{A} \models \varphi$ aber $\mathcal{B} \not\models \varphi$.

Partieller Isomorphismus h .
 $h : A' \subseteq A \rightarrow B' \subseteq B$ injektiv,
so dass
für alle $R \in \sigma \cup \{=\}$ mit
 $k = ar(R)$
und $a_1, \dots, a_k \in A'^k$ gilt:
 $(a_1, \dots, a_k) \in R^A$
gdw.
 $(h(a_1), \dots, h(a_k)) \in R^B$.

Formel.

$$\varphi := \exists x (R(x) \wedge \forall y E(x, y))$$

$R(x)$: „ x ist rot“

Ehrenfeucht-Fraïssé Spiele

Seien σ eine relationale Signatur, $m, k \in \mathbb{N}$, \mathcal{A}, \mathcal{B} σ -Strukturen und $\vec{a}' := a'_1, \dots, a'_k \in A^k$, $\vec{b}' := b'_1, \dots, b'_k \in B^k$.

Spieler und deren Ziele.

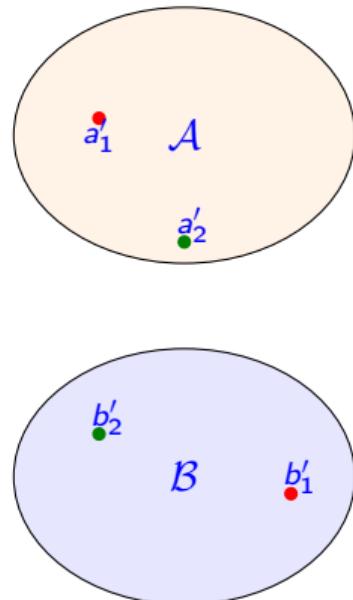
Das m -Runden Ehrenfeucht-Fraïssé Spiel $\mathfrak{G}_m(\mathcal{A}, \vec{a}', \mathcal{B}, \vec{b}')$ wird von zwei Spielern, dem Herausforderer (**H**) und der Duplikatorin (**D**), gespielt.

Herausforderers Ziel: Zeige, dass $(\mathcal{A}, \vec{a}') \not\equiv_m (\mathcal{B}, \vec{b}')$

Duplikatorins Ziel: Zeige, dass $(\mathcal{A}, \vec{a}') \equiv_m (\mathcal{B}, \vec{b}')$

Notation.

Ist $k = 0$ so schreiben wir kurz $\mathfrak{G}_m(\mathcal{A}, \mathcal{B})$.



Ehrenfeucht-Fraïssé Spiele

Die Regeln des Spiels. Eine **Partie** des Spiels besteht aus m Runden.

In Runde $i = 1, \dots, m$:

1. Herausforderer wählt ein Element $a'_i \in A$ oder $b'_i \in B$.
2. Danach antwortet die Duplikatorin. Hat der Herausforderer $a'_i \in A$ gewählt, wählt die Duplikatorin $b'_i \in B$.

Andernfalls wählt sie $a'_i \in A$.

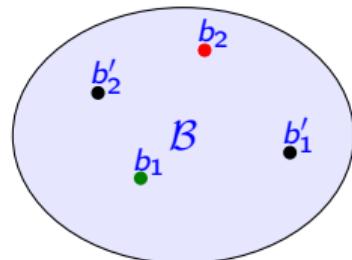
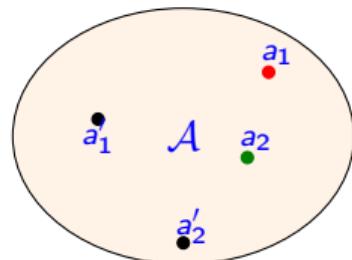
Gewinnbedingung. Nach Runde m wird der Gewinner ermittelt:

Die Duplikatorin hat gewonnen, wenn die Abbildung

$$h : a'_1 \mapsto b'_1, \dots, a'_k \mapsto b'_k, a_1 \mapsto b_1, \dots, a_m \mapsto b_m$$

ein partieller Isomorphismus von \mathcal{A} nach \mathcal{B} ist.

Herausforderer:
 $(\mathcal{A}, \vec{a}') \not\equiv_m (\mathcal{B}, \vec{b}')$.
 Duplikatorin:
 $(\mathcal{A}, \vec{a}') \equiv_m (\mathcal{B}, \vec{b}')$.



Beispiel

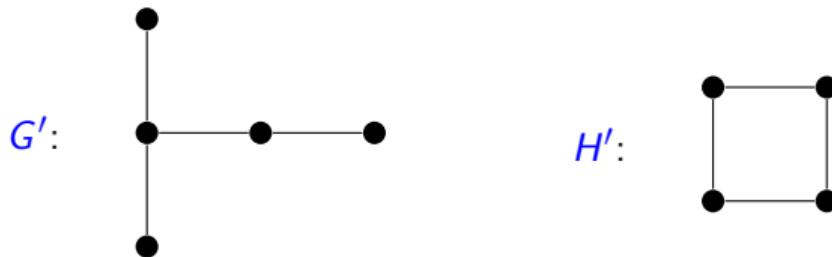
Herausforderer gewinnt $\mathfrak{G}_2(G, H)$ für die Graphen



indem er in Runde 1 den mittleren Knoten a_1 in G wählt.

In Runde 2 wählt der dann einen Knoten b_2 in H , der nicht zu Knoten b_1 benachbart ist.

Beispiel



Frage. Wer gewinnt $\mathfrak{G}_2(G', H')$?

Duplikatorin gewinnt $\mathfrak{G}_2(G', H')$, denn in beiden Graphen gibt es zu jedem Knoten sowohl einen Nachbarn als auch einen Nicht-Nachbarn.

Herausforderer gewinnt $\mathfrak{G}_3(G', H')$, da es in G' drei Knoten gibt, die paarweise nicht benachbart sind.

Was bedeutet eigentlich „gewinnnt“?

Gewinnen? Was bedeutet es eigentlich, dass Duplikatorin *das Spiel* $\mathfrak{G}_2(\mathcal{A}, \mathcal{B})$ gewinnt? Eigentlich gewinnt Sie ja nur eine Partie!

Vielleicht hat Herausforderer ja nicht besonders schlau gespielt?

Strategie: Abbildung, die für jede Runde und jeden möglichen Spielstand den nächsten Zug der Spieler:in angibt.

Gewinnstrategie: Strategie, mit der die Spieler:in jede Partie gewinnt, egal wie die Gegenspieler:in zieht.

Lemma (determinierte Spiele). In jedem Spiel $\mathfrak{G}_m(\mathcal{A}, \mathcal{B})$ hat genau eine der beiden Spieler:innen eine Gewinnstrategie.

Das Spiel gewinnen. Hat eine der beiden Spieler:innen eine Gewinnstrategie, dann sagen wir, dass sie das Spiel gewinnt.

Eigenschaften von EF-Spielen

Sei σ eine Signatur und \mathcal{A}, \mathcal{B} σ -Strukturen.

Eigenschaften von EF-Spielen.

1. Herausforderer gewinnt $\mathfrak{G}_1(\mathcal{A}, \mathcal{B})$, wenn es ein $R \in \sigma$ und ein $a \in A$ gibt mit $(a, \dots, a) \in R^{\mathcal{A}}$ aber für alle $b \in B$ gilt $(b, \dots, b) \notin R^{\mathcal{B}}$ oder umgekehrt.
2. Wenn Herausforderer das Spiel $\mathfrak{G}_m(\mathcal{A}, \mathcal{B})$ gewinnt, dann gewinnt er auch $\mathfrak{G}_{m'}(\mathcal{A}, \mathcal{B})$ für alle $m' > m$.
3. Umgekehrt gilt: Wenn die Duplikatorin das Spiel $\mathfrak{G}_m(\mathcal{A}, \mathcal{B})$ gewinnt, dann gewinnt sie auch $\mathfrak{G}_{m'}(\mathcal{A}, \mathcal{B})$ für alle $m' < m$.
4. Wenn es ausgezeichnete Elemente $\bar{a}' \in A^k$ und $\bar{b}' \in B^k$ gibt, dann gewinnt Duplikatorin $\mathfrak{G}_0(\mathcal{A}, \bar{a}', \mathcal{B}, \bar{b}')$ genau dann, wenn $h : a'_i \mapsto b'_i$, für alle i , ein partieller Isomorphismus ist.
5. Wenn $\mathcal{A} \cong \mathcal{B}$, dann gewinnt die Duplikatorin $\mathfrak{G}_m(\mathcal{A}, \mathcal{B})$ für alle $m \geq 0$.

Partieller Isomorphismus h .
 $h : A' \subseteq A \rightarrow B' \subseteq B$ injektiv,
so dass
für alle $R \in \sigma \cup \{=\}$ mit
 $k = ar(R)$
und $a_1, \dots, a_k \in A'^k$ gilt:
 $(a_1, \dots, a_k) \in R^{\mathcal{A}}$
gdw.
 $(h(a_1), \dots, h(a_k)) \in R^{\mathcal{B}}$.

EF-Spiele auf linearen Ordnungen

Theorem.

Seien $\mathcal{A} := (A, \leq^{\mathcal{A}})$ und $\mathcal{B} := (B, \leq^{\mathcal{B}})$ endliche lineare Ordnungen.

Dann gilt für alle $m \in \mathbb{N}$:

Die Duplikatorin gewinnt das Spiel $\mathfrak{G}_m(\mathcal{A}, \mathcal{B})$

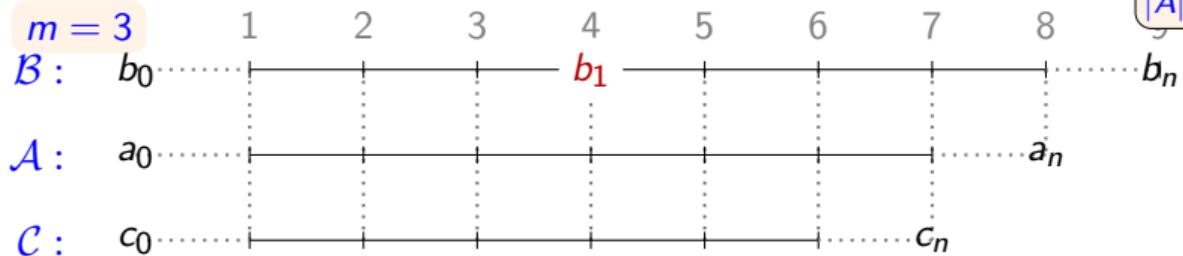
$$\iff$$

$|A| = |B|$ oder $|A|, |B| \geq 2^m - 1$.

Beweisskizze

Theorem.

D gewinnt $\mathfrak{G}_m(\mathcal{A}, \mathcal{B}) \iff |\mathcal{A}| = |\mathcal{B}| \text{ oder } |\mathcal{A}|, |\mathcal{B}| \geq 2^m - 1$.



Strukturen. Seien $\mathcal{A} = (A, \leq^{\mathcal{A}})$ und $\mathcal{B} := (B, \leq^{\mathcal{B}})$.

O.B.d.A. nehmen wir an, dass $A = \{1, \dots, n_a\}$ und $B := \{1, \dots, n_b\}$.

Zur Vereinfachung sei $a_0 = b_0 = 0$, $a_n = n_a + 1$ und $b_n = n_b + 1$.

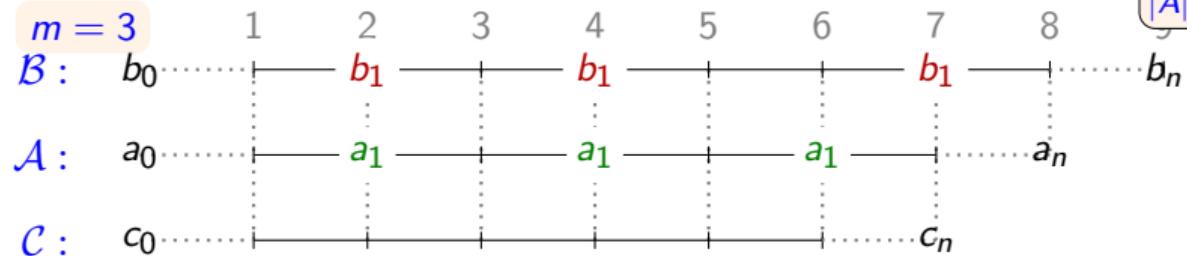
Invariante. Ang., nach Runde i wurden a_1, \dots, a_i und b_1, \dots, b_i gezogen.

Dann gilt für alle $j, j' \in \{0, \dots, i, n\}$:

1. $a_j < a_{j'}$ gdw. $b_j < b_{j'}$ und
2. $|A(j, j')| = |B(j, j')|$ oder $|A(j, j')|, |B(j, j')| \geq 2^{m-i} - 1$.

$(B(j, j')) := \{b \in B : b_j < b < b_{j'} \text{ oder } b_{j'} < b < b_j\}$ $A(j, j')$ analog)

Beweisskizze



Runde 1.

Fall 1. H zieht $b_1 \in B$ „in der Mitte“: $|B(b_0, b_1)|, |B(b_1, b_n)| \geq 2^{m-1} - 1$.

D wählt $a_1 \in A$ mit $|A(a_0, a_1)|, |A(a_1, a_n)| \geq 2^{m-1} - 1$.

a_1 existiert, da $|A| \geq 2^m - 1$ und $2 \cdot (2^{m-1} - 1) - 1 = 2^m - 1$.

Fall 2. H zieht $b_1 \in B$ „weit links“: $|B(b_0, b_1)| < 2^{m-1} - 1$.

D wählt $a_1 \in A$ mit $|A(a_0, a_1)| = |B(b_0, b_1)|$.

Fall 3. H zieht $b_1 \in B$ „weit rechts“: $|B(b_1, b_n)| > 2^{m-1} - 1$.

D wählt $a_1 \in A$ mit $|A(a_1, a_n)| = |B(b_1, b_n)|$.

Theorem.

D gewinnt $\mathfrak{G}_m(\mathcal{A}, \mathcal{B}) \iff |\mathcal{A}| = |\mathcal{B}| \text{ oder } |\mathcal{A}|, |\mathcal{B}| \geq 2^m - 1$.

Strukturen. O.B.d.A.

$$\mathcal{A} = (A, \leq^{\mathcal{A}})$$

$$A = \{1, \dots, n_a\}$$

$$a_0 = 0 \text{ und } a_n = n_a$$

$$\mathcal{B} := (B, \leq^{\mathcal{B}}).$$

$$B = \{1, \dots, n_b\}.$$

$$b_0 = 0 \text{ und } b_n = n_b.$$

Invariante. Nach Runde i :

Gezogen: $a_1, \dots, a_i, b_1, \dots, b_i$.

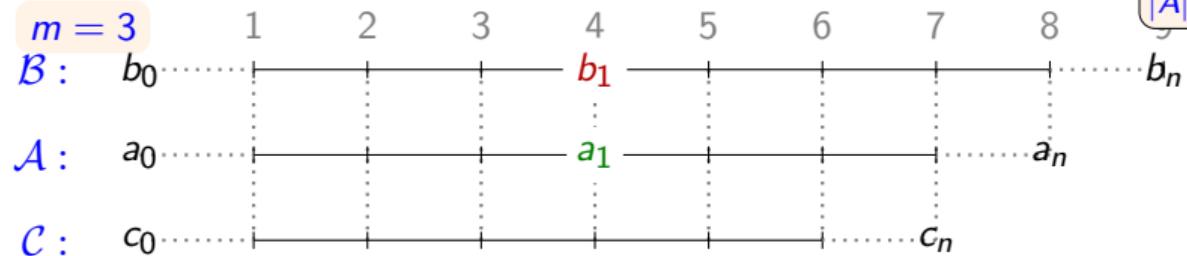
Für alle $j, j' \in \{0, \dots, i, n\}$:

1. $a_j < a_{j'}$ gdw. $b_j < b_{j'}$ und
2. $|A(j, j')| = |B(j, j')|$ oder

$$\left. \begin{aligned} |A(j, j')| \\ |B(j, j')| \end{aligned} \right\} \geq 2^{m-i} - 1.$$

$$B(j, j') = \{b : b_j < b < b_{j'} \vee b_{j'} < b < b_j\}$$

Beweisskizze



Runde 2.

Beobachtung. Wenn H in $B(b_0, b_1)$ oder $A(a_0, a_1)$ zieht, muss H in $A(a_0, a_1)$ bzw. $B(b_0, b_1)$ antworten.

Analog wenn H in $B(b_1, b_n)$ oder $A(a_1, a_n)$ zieht.

Nach Runde 1 spielen wir also eigentlich zwei getrennte Spiele $\mathfrak{G}_2(\mathcal{A}[A(a_0, a_1)], \mathcal{B}[B(b_0, b_1)])$ und $\mathfrak{G}_2(\mathcal{A}[A(a_1, a_n)], \mathcal{B}[B(b_1, b_n)])$.

Wegen der Invariante gelten für beide Spiele wieder die Voraussetzungen des Theorems.

Wir können also einfach rekursiv weiter spielen.

Theorem.

D gewinnt $\mathfrak{G}_m(\mathcal{A}, \mathcal{B}) \iff |\mathcal{A}| = |\mathcal{B}| \text{ oder } |\mathcal{A}|, |\mathcal{B}| \geq 2^m - 1$.

Strukturen. O.B.d.A.

$$\mathcal{A} = (A, \leq^{\mathcal{A}})$$

$$A = \{1, \dots, n_a\}$$

$$a_0 = 0 \text{ und } a_n = n_a$$

$$\mathcal{B} = (B, \leq^{\mathcal{B}})$$

$$B = \{1, \dots, n_b\}$$

$$b_0 = 0 \text{ und } b_n = n_b$$

Invariante. Nach Runde i :

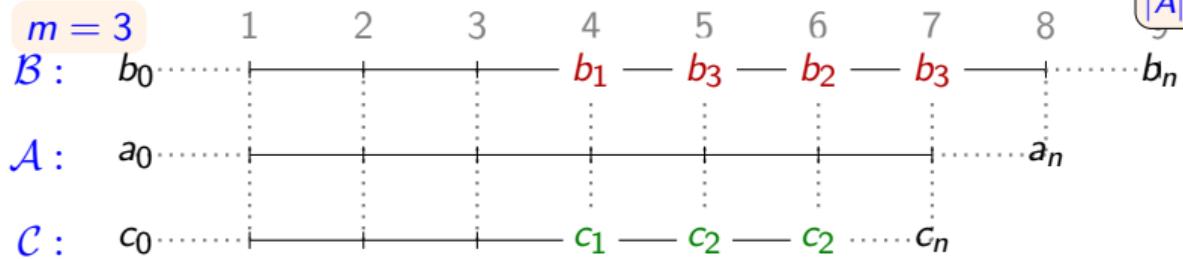
Gezogen: $a_1, \dots, a_i, b_1, \dots, b_i$.

Für alle $j, j' \in \{0, \dots, i, n\}$:

1. $a_j < a_{j'}$ gdw. $b_j < b_{j'}$ und
2. $|A(j, j')| = |B(j, j')|$ oder
 $|A(j, j')| \quad |B(j, j')| \geq 2^{m-i} - 1$.

$$B(j, j') = \{b : b_j < b < b_{j'} \vee b_{j'} < b < b_j\}$$

Beweisskizze



Das Spiel $\mathfrak{G}_3(\mathcal{B}, \mathcal{C})$. H zieht b_1 „in der Mitte“ von \mathcal{B} .

D antwortet mit c_1 .

Nun gilt aber $|C(c_0, c_1)| < 2^{m-1} - 1$ oder $|C(c_1, c_n)| < 2^{m-1} - 1$
aber $|B(b_0, b_1)|, |B(b_1, b_n)| \geq 2^{m-1} - 1$.

Falls $|C(c_1, c_n)| < 2^{m-1} - 1$, spielt H mit der gleichen Strategie auf $B(b_1, b_n)$ weiter, ansonsten auf $B(b_0, b_1)$.

Theorem.

D gewinnt $\mathfrak{G}_m(\mathcal{A}, \mathcal{B}) \iff |\mathcal{A}| = |\mathcal{B}| \text{ oder } |\mathcal{A}|, |\mathcal{B}| \geq 2^m - 1$.

Strukturen. O.B.d.A.

$$\mathcal{A} = (A, \leq^{\mathcal{A}})$$

$$A = \{1, \dots, n_a\}$$

$$a_0 = 0 \text{ und } a_n = n_a$$

$$\mathcal{B} := (B, \leq^{\mathcal{B}})$$

$$B := \{1, \dots, n_b\}$$

$$b_0 = 0 \text{ und } b_n = n_b$$

Invariante. Nach Runde i :

Gezogen: $a_1, \dots, a_i, b_1, \dots, b_i$.

Für alle $j, j' \in \{0, \dots, i, n\}$:

1. $a_j < a_{j'} \text{ gdw. } b_j < b_{j'} \text{ und }$
2. $|A(j, j')| = |B(j, j')| \text{ oder } \begin{cases} |A(j, j')| \\ |B(j, j')| \end{cases} \geq 2^{m-i} - 1$.

$$B(j, j') = \{b : b_j < b < b_{j'} \vee b_{j'} < b < b_j\}$$

EF-Spiele auf linearen Ordnungen

Theorem.

Seien $\mathcal{A} := (A, \leq^{\mathcal{A}})$ und $\mathcal{B} := (B, \leq^{\mathcal{B}})$ endliche lineare Ordnungen.

Dann gilt für alle $m \in \mathbb{N}$:

Die Duplikatorin gewinnt das Spiel $\mathfrak{G}_m(\mathcal{A}, \mathcal{B})$

$$\iff$$

$|A| = |B|$ oder $|A|, |B| \geq 2^m - 1$.

Ehrenfeucht-Fraïssé Spiele

Wichtige Variante des m -Runden Ehrenfeucht-Fraïssé Spiels:

Spiel $\mathfrak{G}(\mathcal{A}, \bar{a}, \mathcal{B}, \bar{b})$ mit unbeschränkter Zugzahl.

Hier wählt der Herausforderer zunächst eine Zahl $m \geq 0$ und dann wird das m -Runden Spiel $\mathfrak{G}_m(\mathcal{A}, \bar{a}, \mathcal{B}, \bar{b})$ gespielt.

Theorem (Satz von Ehrenfeucht). Sei σ eine Signatur, $k \in \mathbb{N}$ und \mathcal{A}, \mathcal{B} zwei σ -Strukturen mit $\bar{a} \in A^k, \bar{b} \in B^k$.

1. Folgende Aussagen sind äquivalent:
 - 1.1 $(\mathcal{A}, \bar{a}) \equiv (\mathcal{B}, \bar{b})$
 - 1.2 Die Duplikatorin gewinnt $\mathfrak{G}((\mathcal{A}, \bar{a}), (\mathcal{B}, \bar{b}))$
2. Für alle $m \geq 0$ sind folgende Aussagen sind äquivalent:
 - 2.1 $(\mathcal{A}, \bar{a}) \equiv_m (\mathcal{B}, \bar{b})$
 - 2.2 Die Duplikatorin gewinnt $\mathfrak{G}_m((\mathcal{A}, \bar{a}), (\mathcal{B}, \bar{b}))$

10.6 Der Satz von Ehrenfeucht

Der Satz von Ehrenfeucht

Theorem (Satz von Ehrenfeucht).

Sei σ eine Signatur, \mathcal{A}, \mathcal{B} σ -Strukturen,

$\bar{a}' := a'_1, \dots, a'_k \in A^k$ und $\bar{b}' := b'_1, \dots, b'_k \in B^k$.

1. Folgende Aussagen sind äquivalent:

1.1 $(\mathcal{A}, \bar{a}') \equiv (\mathcal{B}, \bar{b}')$

1.2 Die Duplikatorin gewinnt $\mathfrak{G}(\mathcal{A}, \bar{a}', \mathcal{B}, \bar{b}')$

2. Für alle $m \geq 0$ sind folgende Aussagen sind äquivalent:

2.1 $(\mathcal{A}, \bar{a}') \equiv_m (\mathcal{B}, \bar{b}')$

2.2 Die Duplikatorin gewinnt $\mathfrak{G}_m(\mathcal{A}, \bar{a}', \mathcal{B}, \bar{b}')$

(Erinnerung: „gewinnt“ \triangleq Gewinnstrategie)

Beweis des Satzes

Intuition. Seien \mathcal{A}, \mathcal{B} zwei σ -Strukturen, $\bar{a}' \in A^k, \bar{b}' \in B^k$ und $m \geq 0$.

Wir wollen zeigen, dass $(\mathcal{A}, \bar{a}') \equiv_m (\mathcal{B}, \bar{b}')$ genau dann gilt, wenn die Duplikatorin das Spiel $\mathfrak{G}_m(\mathcal{A}, \bar{a}', \mathcal{B}, \bar{b}')$ gewinnt, d.h. eine Gewinnstrategie hat.

Beobachtung.

D hat eine Gewinnstrategie in $\mathfrak{G}_m(\mathcal{A}, \mathcal{B})$ gdw. sie

- für jeden Zug a_1 von H in \mathcal{A} ein Element $b_1 \in B$ wählen kann, so dass sie eine Gewinnstrategie im Spiel $\mathfrak{G}_{m-1}(\mathcal{A}, \bar{a}', a_1, \mathcal{B}, \bar{b}', b_1)$ hat und
- für jeden Zug b_1 von H in \mathcal{B} ein Element $a_1 \in A$ wählen kann, so dass sie eine Gewinnstrategie im Spiel $\mathfrak{G}_{m-1}(\mathcal{A}, \bar{a}', a_1, \mathcal{B}, \bar{b}', b_1)$ hat.

Es bietet sich daher ein Beweis des Satzes per Induktion über m an.

Theorem. Für $m \geq 0$:

$(\mathcal{A}, \bar{a}) \equiv_m (\mathcal{B}, \bar{b})$ gdw.

D gewinnt $\mathfrak{G}_m(\mathcal{A}, \bar{a}, \mathcal{B}, \bar{b})$

Der Basisfall $m = 0$.

Sei $m = 0$.

Zu zeigen.

$(\mathcal{A}, \bar{a}') \equiv_0 (\mathcal{B}, \bar{b}')$ gdw. D gewinnt $\mathfrak{G}_0(\mathcal{A}, \bar{a}', \mathcal{B}, \bar{b}')$.

Beweis.

D gewinnt $\mathfrak{G}_0(\mathcal{A}, \bar{a}', \mathcal{B}, \bar{b}')$

gdw.

$h : a'_1 \mapsto b'_1, \dots, a'_l \mapsto b'_l$ ein partieller Isomorphismus ist.

Daher ist zu zeigen, dass dies genau dann gilt, wenn $(\mathcal{A}, \bar{a}) \equiv_0 (\mathcal{B}, \bar{b})$.

Genau das besagt das schon bewiesene Lemma.

Theorem. Für $m \geq 0$:

$(\mathcal{A}, \bar{a}) \equiv_m (\mathcal{B}, \bar{b})$ gdw.

D gewinnt $\mathfrak{G}_m(\mathcal{A}, \bar{a}, \mathcal{B}, \bar{b})$

Lemma.

Folgende Aussagen äquivalent:

1. $h : A' \rightarrow B'$ mit $h(a_i) = b_i$ ist partieller Isom.
2. $\psi(x_1, \dots, x_k)$ atomar:
 $\mathcal{A} \models \psi[\bar{a}]$ gdw. $\mathcal{B} \models \psi[\bar{b}]$
3. $\psi(x_1, \dots, x_k)$ quantorenfrei:
 $\mathcal{A} \models \psi[\bar{a}]$ gdw. $\mathcal{B} \models \psi[\bar{b}]$
4. $(\mathcal{A}, \bar{a}) \equiv_0 (\mathcal{B}, \bar{b})$

Der Satz von Ehrenfeucht

Theorem (Satz von Ehrenfeucht).

Sei σ eine Signatur, \mathcal{A}, \mathcal{B} σ -Strukturen,

$\bar{a}' := a'_1, \dots, a'_k \in A^k$ und $\bar{b}' := b'_1, \dots, b'_k \in B^k$.

1. Folgende Aussagen sind äquivalent:

1.1 $(\mathcal{A}, \bar{a}') \equiv (\mathcal{B}, \bar{b}')$

1.2 Die Duplikatorin gewinnt $\mathfrak{G}(\mathcal{A}, \bar{a}', \mathcal{B}, \bar{b}')$

2. Für alle $m \geq 0$ sind folgende Aussagen sind äquivalent:

2.1 $(\mathcal{A}, \bar{a}') \equiv_m (\mathcal{B}, \bar{b}')$

2.2 Die Duplikatorin gewinnt $\mathfrak{G}_m(\mathcal{A}, \bar{a}', \mathcal{B}, \bar{b}')$

(Erinnerung: „gewinnt“ \triangleq Gewinnstrategie)

m-Isomorphietypen

Als Hilfsmittel zum Beweis des Satzes von Ehrenfeucht verwenden wir folgende induktiv definierte Formeln $\varphi_{\mathcal{A}, \bar{a}}^m(\bar{x})$:

***m*-Isomorphietypen oder Hintikka-Formeln.** Sei σ eine Signatur.

Sei \mathcal{A} eine σ -Struktur, $\bar{a} := a_1, \dots, a_k \in A^k$ und $\bar{x} := x_1, \dots, x_k$.

$$\varphi_{\mathcal{A}, \bar{a}}^0(\bar{x}) := \bigwedge \left\{ \psi(\bar{x}) : \psi \text{ ist atomar oder negiert atomar und } \mathcal{A} \models \psi[\bar{a}] \right\}$$

und für $m \geq 0$

$$\varphi_{\mathcal{A}, \bar{a}}^{m+1}(\bar{x}) := \bigwedge_{a \in A} \left\{ \exists x_{k+1} \varphi_{\mathcal{A}, \bar{a}, a}^m(\bar{x}, x_{k+1}) \right\} \wedge \forall x_{k+1} \bigvee_{a \in A} \left\{ \varphi_{\mathcal{A}, \bar{a}, a}^m(\bar{x}, x_{k+1}) \right\}.$$

Beispiel

$$\varphi_{\mathcal{A}, \bar{a}}^0(\bar{x}) := \bigwedge \left\{ \psi(\bar{x}) : \psi \text{ ist atomar oder negiert atomar und } \mathcal{A} \models \psi[\bar{a}] \right\}$$

$$\varphi_{\mathcal{A}, \bar{a}}^{m+1}(\bar{x}) := \bigwedge_{a \in A} \left\{ \exists x_{k+1} \varphi_{\mathcal{A}, \bar{a}, a}^m(\bar{x}, x_{k+1}) \right\} \wedge \forall x_{k+1} \bigvee_{a \in A} \left\{ \varphi_{\mathcal{A}, \bar{a}, a}^m(\bar{x}, x_{k+1}) \right\}$$

$m = 0$.

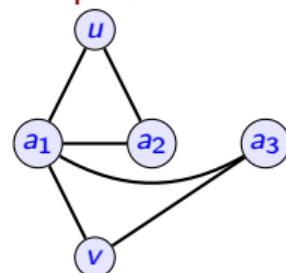
$$\varphi_{\mathcal{A}, uv}^0(x_1, x_2) := \left(\begin{array}{lcl} \neg E(x_1, x_2) & \wedge & \neg E(x_2, x_1) \\ \neg E(x_1, x_1) & \wedge & \neg E(x_2, x_2) \\ \neg x_1 = x_2 & \wedge & \neg x_2 = x_1 \\ x_1 = x_1 & \wedge & x_2 = x_2 \end{array} \right).$$

$m > 0$.

$$\varphi_{\mathcal{A}, uv}^1(x_1, x_2) := \exists x_3 \left(\begin{array}{lcl} E(x_1, x_3) & \wedge & E(x_2, x_3) \\ \neg x_3 = x_1 & \wedge & \neg x_2 = x_3 \\ \varphi_{\mathcal{A}, uv}^0 \dots \end{array} \right)$$

$\wedge \dots$ (entspr. für $a = a_2, a_3, a = u, a = v$)

Graph \mathcal{A} .



$$\left(\begin{array}{c} (\varphi_{\mathcal{A}, uva_1}^0 \text{ für } a = a_1) \\ x_1 \hat{=} u \quad x_2 \hat{=} v \\ x_3 \hat{=} a_1 \end{array} \right)$$

m-Isomorphietypen

Hintikka-Formeln.

$$\varphi_{\mathcal{A}, \bar{a}}^0(\bar{x}) := \bigwedge \left\{ \psi(\bar{x}) : \psi \text{ ist atomar oder negiert atomar und } \mathcal{A} \models \psi[\bar{a}] \right\}$$

$$\varphi_{\mathcal{A}, \bar{a}}^{m+1}(\bar{x}) := \bigwedge_{a \in A} \left\{ \exists x_{k+1} \varphi_{\mathcal{A}, \bar{a}, a}^m(\bar{x}, x_{k+1}) \right\} \wedge \forall x_{k+1} \bigvee_{a \in A} \left\{ \varphi_{\mathcal{A}, \bar{a}, a}^m(\bar{x}, x_{k+1}) \right\}$$

Beobachtung.

- Der Quantorenrang von $\varphi_{\mathcal{A}, \bar{a}}^m$ ist genau m .
- Wir haben bereits bewiesen, dass es für jedes $m \geq 0$ und jede feste Menge X von Variablen nur eine endliche Anzahl paarweise nicht-äquivalenter Formeln mit Quantorenrang $\leq m$ und freien Variablen aus X gibt.

Weil hier der Quantorenrang m und die freien Variablen x_1, \dots, x_{k+1} beschränkt sind, sind die großen Konjunktionen und Disjunktionen endlich, auch wenn das Universum A unendlich groß ist.

Ein technisches Hilfslemma

Lemma.

Sei σ eine Signatur, \mathcal{A}, \mathcal{B} σ -Strukturen, $\bar{a} := (a_1, \dots, a_k) \in A^k$ und $\bar{b} := (b_1, \dots, b_k) \in B^k$.

Für alle $m \geq 0$ sind folgende Aussagen äquivalent:

1. Die Duplikatorin gewinnt $\mathfrak{G}_m(\mathcal{A}, \bar{a}, \mathcal{B}, \bar{b})$.
2. $\mathcal{B} \models \varphi_{\mathcal{A}, \bar{a}}^m[\bar{b}]$.
3. $(\mathcal{A}, \bar{a}) \equiv_m (\mathcal{B}, \bar{b})$.

Beweis 3 \Rightarrow 2

Beweis (3) \Rightarrow (2).

Voraussetzung: $(\mathcal{A}, \bar{a}) \equiv_m (\mathcal{B}, \bar{b})$

Zu zeigen: $\mathcal{B} \models \varphi_{\mathcal{A}, \bar{a}}^m[\bar{b}]$.

Angenommen $(\mathcal{A}, \bar{a}) \equiv_m (\mathcal{B}, \bar{b})$.

Da $\mathcal{A} \models \varphi_{\mathcal{A}, \bar{a}}^m[\bar{a}]$ und $\text{qr}(\varphi_{\mathcal{A}, \bar{a}}^m) = m$

folgt sofort $\mathcal{B} \models \varphi_{\mathcal{A}, \bar{a}}^m[\bar{b}]$.

Lemma.

1. \mathcal{D} gewinnt $\mathfrak{G}_m(\mathcal{A}, \bar{a}, \mathcal{B}, \bar{b})$
- \Leftrightarrow 2. $\mathcal{B} \models \varphi_{\mathcal{A}, \bar{a}}^m[\bar{b}]$
- \Leftrightarrow 3. $(\mathcal{A}, \bar{a}) \equiv_m (\mathcal{B}, \bar{b})$.

Beweis 1 \iff 2

Beweis (1) \iff (2).

1. Die Duplicatorin gewinnt $\mathfrak{G}_m(\mathcal{A}, \bar{a}, \mathcal{B}, \bar{b})$.
2. $\mathcal{B} \models \varphi_{\mathcal{A}, \bar{a}}^m[\bar{b}]$.

Der Beweis folgt per Induktion über m .

Für $m = 0$ gilt:

D gewinnt $\mathfrak{G}_0(\mathcal{A}, \bar{a}, \mathcal{B}, \bar{b})$

gdw. $\bar{a} \mapsto \bar{b}$ ist ein partieller Isomorphismus von \mathcal{A} nach \mathcal{B}

gdw. $\mathcal{B} \models \varphi_{\mathcal{A}, \bar{a}}^0[\bar{b}]$ (haben wir eben bewiesen).

Lemma.

1. D gewinnt $\mathfrak{G}_m(\mathcal{A}, \bar{a}, \mathcal{B}, \bar{b})$
- \iff 2. $\mathcal{B} \models \varphi_{\mathcal{A}, \bar{a}}^m[\bar{b}]$
- \iff 3. $(\mathcal{A}, \bar{a}) \equiv_m (\mathcal{B}, \bar{b})$.

Hintikka-Formeln.

$$\begin{aligned}\varphi_{\mathcal{A}, \bar{a}}^0(\bar{x}) &:= \\ \wedge \left\{ \psi(\bar{x}) : \psi \text{ ist atomar oder negiert} \right. \\ &\quad \left. \text{atomar und } \mathcal{A} \models \psi[\bar{a}] \right\}\end{aligned}$$

$$\begin{aligned}\varphi_{\mathcal{A}, \bar{a}}^{m+1}(\bar{x}) &:= \\ \wedge_{a \in A} \left\{ \exists x_{k+1} \varphi_{\mathcal{A}, \bar{a}, a}^m(\bar{x}, x_{k+1}) \right\} \wedge \\ \forall x_{k+1} \vee_{a \in A} \left\{ \varphi_{\mathcal{A}, \bar{a}, a}^m(\bar{x}, x_{k+1}) \right\}\end{aligned}$$

Beweis 2 \iff 1

Für $m > 0$ gilt:

- D gewinnt $\mathfrak{G}_m(\mathcal{A}, \bar{a}, \mathcal{B}, \bar{b})$
- \iff für alle $a \in A$ gibt es ein $b \in B$,
so dass D $\mathfrak{G}_{m-1}(\mathcal{A}, \bar{a}a, \mathcal{B}, \bar{b}b)$ gewinnt und
- für alle $b \in B$ gibt es ein $a \in A$,
 so dass D $\mathfrak{G}_{m-1}(\mathcal{A}, \bar{a}a, \mathcal{B}, \bar{b}b)$ gewinnt
- $\stackrel{!}{\iff}$ für alle $a \in A$ gibt es ein $b \in B$,
 so dass $\mathcal{B} \models \varphi_{\mathcal{A}, \bar{a}a}^{m-1}[\bar{b}b]$ und
- für alle $b \in B$ gibt es ein $a \in A$,
 so dass $\mathcal{B} \models \varphi_{\mathcal{A}, \bar{a}a}^{m-1}[\bar{b}b]$
- $\iff \mathcal{B} \models \left(\bigwedge_{a \in A} \exists x_{k+1} \varphi_{\mathcal{A}, \bar{a}a}^{m-1}(\bar{x}, x_{k+1}) \wedge \forall x_{k+1} \bigvee_{a \in A} \varphi_{\mathcal{A}, \bar{a}a}^{m-1}(\bar{x}, x_{k+1}) \right) [\bar{b}]$
- $\iff \mathcal{B} \models \varphi_{\mathcal{A}, \bar{a}}^m[\bar{b}]$.

Lemma.

1. D gewinnt $\mathfrak{G}_m(\mathcal{A}, \bar{a}, \mathcal{B}, \bar{b})$
- \iff 2. $\mathcal{B} \models \varphi_{\mathcal{A}, \bar{a}}^m[\bar{b}]$
- \iff 3. $(\mathcal{A}, \bar{a}) \equiv_m (\mathcal{B}, \bar{b})$.

Hintikka-Formeln.

$$\begin{aligned}\varphi_{\mathcal{A}, \bar{a}}^0(\bar{x}) &:= \\ &\wedge \left\{ \psi(\bar{x}) : \psi \text{ ist atomar oder negiert} \right. \\ &\quad \left. \text{atomar und } \mathcal{A} \models \psi[\bar{a}] \right\} \\ \varphi_{\mathcal{A}, \bar{a}}^{m+1}(\bar{x}) &:= \\ &\bigwedge_{a \in A} \left\{ \exists x_{k+1} \varphi_{\mathcal{A}, \bar{a}, a}^m(\bar{x}, x_{k+1}) \right\} \wedge \\ &\quad \forall x_{k+1} \bigvee_{a \in A} \left\{ \varphi_{\mathcal{A}, \bar{a}, a}^m(\bar{x}, x_{k+1}) \right\}\end{aligned}$$

Beweis (1) \Rightarrow (3)

Beweis (1) \Rightarrow (3).

1. Die Duplikatorin gewinnt $\mathfrak{G}_m(\mathcal{A}, \bar{a}, \mathcal{B}, \bar{b})$.
3. $(\mathcal{A}, \bar{a}) \equiv_m (\mathcal{B}, \bar{b})$.

Der Fall $m = 0$ wurde schon bewiesen. Sei nun $m > 0$.

Angenommen, D gewinnt $\mathfrak{G}_m(\mathcal{A}, \bar{a}, \mathcal{B}, \bar{b})$.

Sei $\psi(x_{i_1}, \dots, x_{i_n}) \in \text{FO}$ mit $\text{qr}(\psi) \leq m$ und $i_1, \dots, i_n \in \{1, \dots, k\}$.

Zu zeigen: $\mathcal{A} \models \psi[\bar{a}'] \iff \mathcal{B} \models \psi[\bar{b}']$

$$\bar{a}' := (a_{i_1}, \dots, a_{i_n}) \quad \bar{b}' := (b_{i_1}, \dots, b_{i_n}).$$

ψ ist eine Boolesche Kombination aus Formeln

1. mit Quantorenrang $< m$ und
2. Formeln der Form $\exists x_{k+1} \chi(\bar{x}', x_{k+1})$, wobei $\text{qr}(\chi) = m - 1$.

Es reicht daher, Formeln des Typs 2 zu betrachten.

Lemma.

1. D gewinnt $\mathfrak{G}_m(\mathcal{A}, \bar{a}, \mathcal{B}, \bar{b})$
- \Leftrightarrow 2. $\mathcal{B} \models \varphi_{\mathcal{A}, \bar{a}}^m[\bar{b}]$
- \Leftrightarrow 3. $(\mathcal{A}, \bar{a}) \equiv_m (\mathcal{B}, \bar{b})$.

Hintikka-Formeln.

$$\begin{aligned}\varphi_{\mathcal{A}, \bar{a}}^0(\bar{x}) &:= \\ &\wedge \left\{ \psi(\bar{x}) : \psi \text{ ist atomar oder negiert} \right. \\ &\quad \left. \text{atomar und } \mathcal{A} \models \psi[\bar{a}] \right\}\end{aligned}$$

$$\begin{aligned}\varphi_{\mathcal{A}, \bar{a}}^{m+1}(\bar{x}) &:= \\ &\wedge_{a \in A} \left\{ \exists x_{k+1} \varphi_{\mathcal{A}, \bar{a}, a}^m(\bar{x}, x_{k+1}) \right\} \wedge \\ &\forall x_{k+1} \vee_{a \in A} \left\{ \varphi_{\mathcal{A}, \bar{a}, a}^m(\bar{x}, x_{k+1}) \right\}\end{aligned}$$

Beweis 1 \iff 3

Sei also $\psi = \exists x_{k+1} \chi(\bar{x}', x_{k+1})$, wobei $\text{qr}(\chi) = m - 1$.

Angenommen, $\mathcal{A} \models \psi[\bar{a}']$.

Dann gibt es ein $a \in A$ mit $\mathcal{A} \models \chi[\bar{a}', a]$.

Wähle ein solches a .

Da $\text{D } \mathfrak{G}_m(\mathcal{A}, \bar{a}, \mathcal{B}, \bar{b})$ gewinnt, gibt es ein $b \in B$, so dass $\text{D } \mathfrak{G}_{m-1}(\mathcal{A}, \bar{a}a, \mathcal{B}, \bar{b}b)$ gewinnt.

Nach Induktionsvoraussetzung gilt also

$$\mathcal{A} \models \chi[\bar{a}', a] \iff \mathcal{B} \models \chi[\bar{b}', b].$$

Mit $\mathcal{A} \models \chi[\bar{a}', a]$ folgt daraus $\mathcal{B} \models \chi[\bar{b}', b]$ und daher $\mathcal{B} \models \psi[\bar{b}']$.

Der Fall $\mathcal{B} \models \psi[\bar{b}']$ folgt analog.

Ein technisches Hilfslemma

Lemma.

Sei σ eine Signatur, \mathcal{A}, \mathcal{B} σ -Strukturen, $\bar{a} := (a_1, \dots, a_k) \in A^k$ und $\bar{b} := (b_1, \dots, b_k) \in B^k$.

Für alle $m \geq 0$ sind folgende Aussagen äquivalent:

1. Die Duplikatorin gewinnt $\mathfrak{G}_m(\mathcal{A}, \bar{a}, \mathcal{B}, \bar{b})$.
2. $\mathcal{B} \models \varphi_{\mathcal{A}, \bar{a}}^m[\bar{b}]$.
3. $(\mathcal{A}, \bar{a}) \equiv_m (\mathcal{B}, \bar{b})$.

Der Satz von Ehrenfeucht

Theorem (Satz von Ehrenfeucht).

Sei σ eine Signatur, \mathcal{A}, \mathcal{B} σ -Strukturen,

$\bar{a}' := a'_1, \dots, a'_k \in A^k$ und $\bar{b}' := b'_1, \dots, b'_k \in B^k$.

1. Folgende Aussagen sind äquivalent:

1.1 $(\mathcal{A}, \bar{a}') \equiv (\mathcal{B}, \bar{b}')$

1.2 Die Duplikatorin gewinnt $\mathfrak{G}(\mathcal{A}, \bar{a}', \mathcal{B}, \bar{b}')$

2. Für alle $m \geq 0$ sind folgende Aussagen sind äquivalent:

2.1 $(\mathcal{A}, \bar{a}') \equiv_m (\mathcal{B}, \bar{b}')$

2.2 Die Duplikatorin gewinnt $\mathfrak{G}_m(\mathcal{A}, \bar{a}', \mathcal{B}, \bar{b}')$

(Erinnerung: „gewinnt“ \triangleq Gewinnstrategie)

Bemerkung

Bemerkung. Sei σ eine Signatur und $m \in \mathbb{N}$.

Aus dem technischen Hilfslemma folgt, dass \equiv_m nur *endlich* viele Äquivalenzklassen hat.

Die Äquivalenzklasse, zu der eine gegebene Struktur \mathcal{A} gehört, wird dabei durch den Satz $\varphi_{\mathcal{A}}^m$ definiert.

10.7 Anwendungen von EF-Spielen

Wiederholung: Beispiele dieses Abschnitts

Frage 1: Kann man in FO zählen?

Ist die Klasse

$$\text{EVEN}_{\leq} := \{(A, \leq) : A \text{ ist endlich und gerader Länge}\}$$

in der Klasse \mathcal{O} aller endlichen linearen Ordnungen FO -definierbar?

Das ist letztlich die Frage, ob die Prädikatenlogik zählen kann.

Frage 2: Erreichbarkeit.

$$\text{Signatur } \sigma := \{E, s, t\}$$

s, t Konstantensymbole

E 2-stelliges Relationssymbol

Ist folgende Klasse FO -definierbar?

$$\text{REACH} := \{\mathcal{A} : \mathcal{A} \text{ } \sigma\text{-Struktur, es gibt einen Pfad von } s^{\mathcal{A}} \text{ nach } t^{\mathcal{A}}\}.$$

*Dahinter steht die Frage, ob die Prädikatenlogik **Schleifenkonstrukte** oder **Rekursion** ausdrücken kann.*

Zählen in der Prädikatenlogik

Frage 1: Kann man in FO zählen?

Ist die Klasse

$$\text{EVEN}_{\leq} := \{(A, \leq) : A \text{ ist endlich und gerader Länge}\}$$

in der Klasse \mathcal{O} aller endlichen linearen Ordnungen FO-definierbar?

Das ist letztlich die Frage, ob die Prädikatenlogik zählen kann.

Antwort. Die Klasse EVEN_{\leq} ist **nicht** FO-definierbar in \mathcal{O} .

Für $m \geq 0$ wähle $\mathcal{A}_m := (A, \leq^{\mathcal{A}})$ mit $A = \{1, \dots, 2^m\}$
 $\mathcal{B}_m := (B, \leq^{\mathcal{B}})$ mit $B = \{1, \dots, 2^m + 1\}$.

Dann ist $\mathcal{A}_m \in \text{EVEN}_{\leq}$ und $\mathcal{B}_m \in \mathcal{O} \setminus \text{EVEN}_{\leq}$.

Wie in Video 12.2 bewiesen, gewinnt \mathbf{D} das Spiel $\mathfrak{G}_m(\mathcal{A}_m, \mathcal{B}_m)$, und, nach dem Satz von Ehrenfeucht, gilt $\mathcal{A}_m \equiv_m \mathcal{B}_m$

Somit ist EVEN_{\leq} nicht in \mathcal{O} definierbar (siehe Video 11.3).

Definition. \mathcal{K} Klasse von σ -Strukturen.

Klasse $\mathcal{C} \subseteq \mathcal{K}$ ist in \mathcal{K} FO-definierbar, wenn es $\psi \in \text{FO}[\sigma]$ gibt, so dass

$$\mathcal{C} = \{\mathcal{A} \in \mathcal{K} : \mathfrak{A} \models \psi\}.$$

$$\text{Mod}_{\mathcal{K}}(\varphi) := \{\mathcal{A} \in \mathcal{K} : \mathcal{A} \models \varphi\}.$$

Lemma. Seien \mathcal{C}, \mathcal{K} Klassen von σ -Strukturen.

Wenn für alle $m \geq 1 \mathcal{A}_m, \mathcal{B}_m \in \mathcal{K}$ existieren, so dass

- $\mathcal{A}_m \in \mathcal{C}$ aber $\mathcal{B}_m \notin \mathcal{C}$
- $\mathcal{A}_m \equiv_m \mathcal{B}_m$,

dann ist \mathcal{C} nicht in \mathcal{K} FO-definierbar.

Theorem. Seien $\mathcal{A} := (A, \leq^{\mathcal{A}})$ und $\mathcal{B} := (B, \leq^{\mathcal{B}})$ endliche lineare Ordnungen. Dann gilt für alle $m \in \mathbb{N}$:
 \mathbf{D} gewinnt $\mathfrak{G}_m(\mathcal{A}, \mathcal{B})$

$$\iff |A| = |B| \text{ oder } |A|, |B| \geq 2^m - 1.$$

Erreichbarkeit

Frage 2: Erreichbarkeit.

Signatur $\sigma := \{E, s, t\}$

s, t Konstantensymbole

E 2-stelliges Relationssymbol

Ist folgende Klasse **FO**-definierbar?

$\text{REACH} := \{(\mathcal{A}, s, t) : (\mathcal{A}, s, t) \text{ } \sigma\text{-Struktur, } s, t \in A,$
 $\text{ es gibt einen Pfad von } s \text{ nach } t \}.$

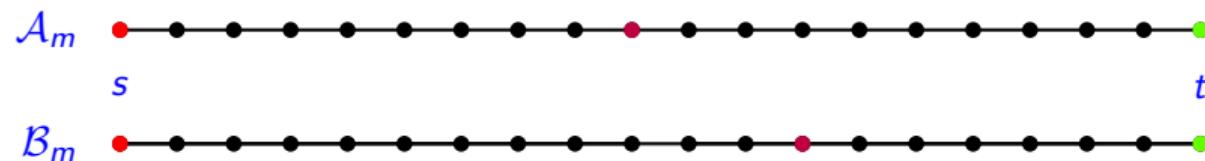
Dahinter steht die Frage, ob die Prädikatenlogik **Schleifenkonstrukte** oder **Rekursion** ausdrücken kann.

Antwort. REACH ist nicht **FO**-definierbar.

Beweis: Idee 1

Behauptung. REACH ist nicht **FO**-definierbar.

Versuch 1. Wir versuchen eine ähnliche Idee wie bei den Ordnungen.

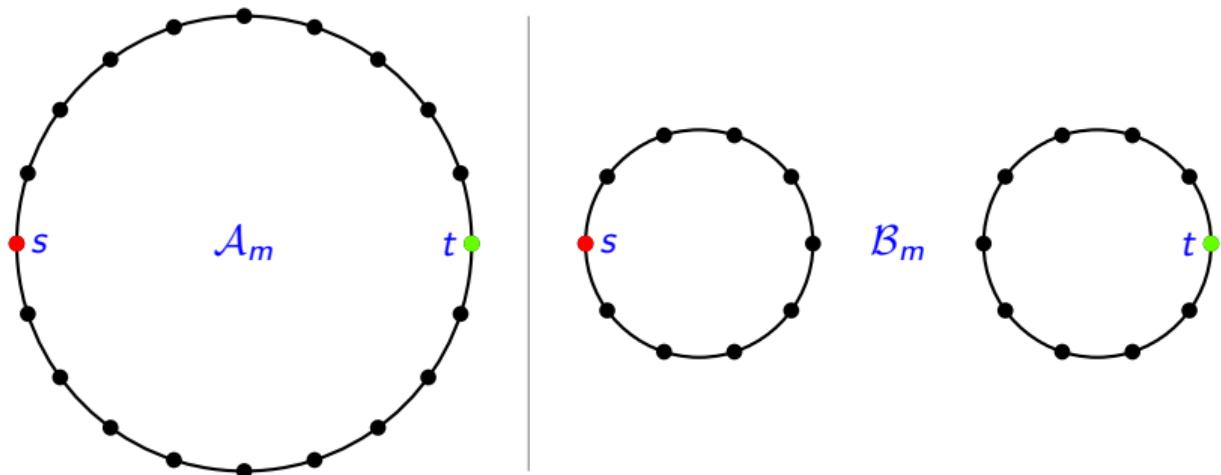


Problem. \mathcal{B}_m hat jetzt **4** Knoten vom Grad **1**, \mathcal{A}_m aber nur **2**.

Beweis: Idee 2

Behauptung. REACH ist nicht **FO**-definierbar.

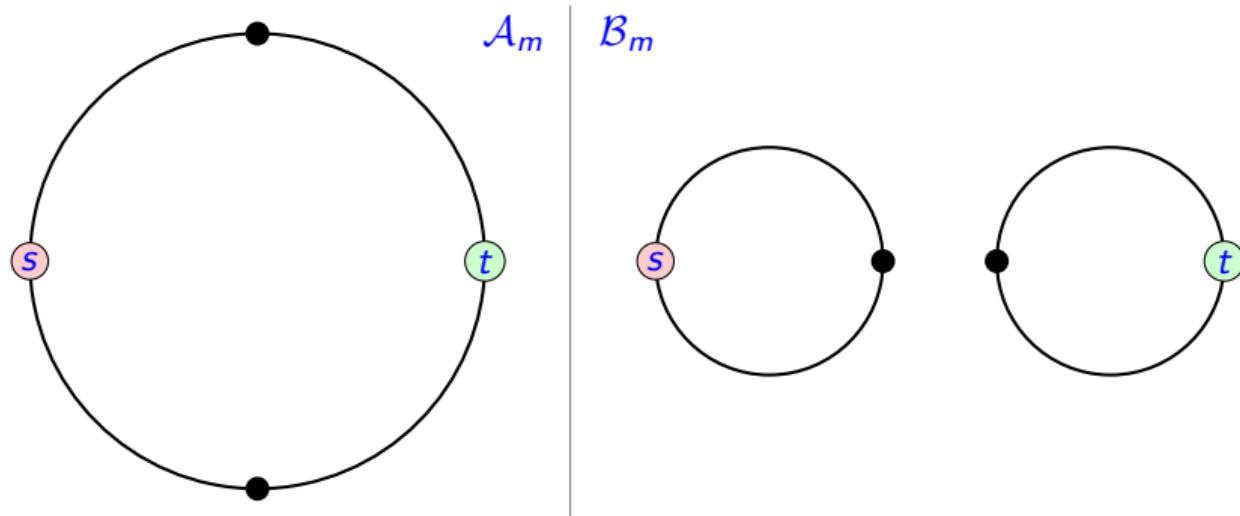
Versuch 2. Um das Problem zu vermeiden wählen wir statt langen Pfaden einfach lange Kreise.



Frage. Wie lang müssen die Kreise sein?

Kreislänge für $m = 0$.

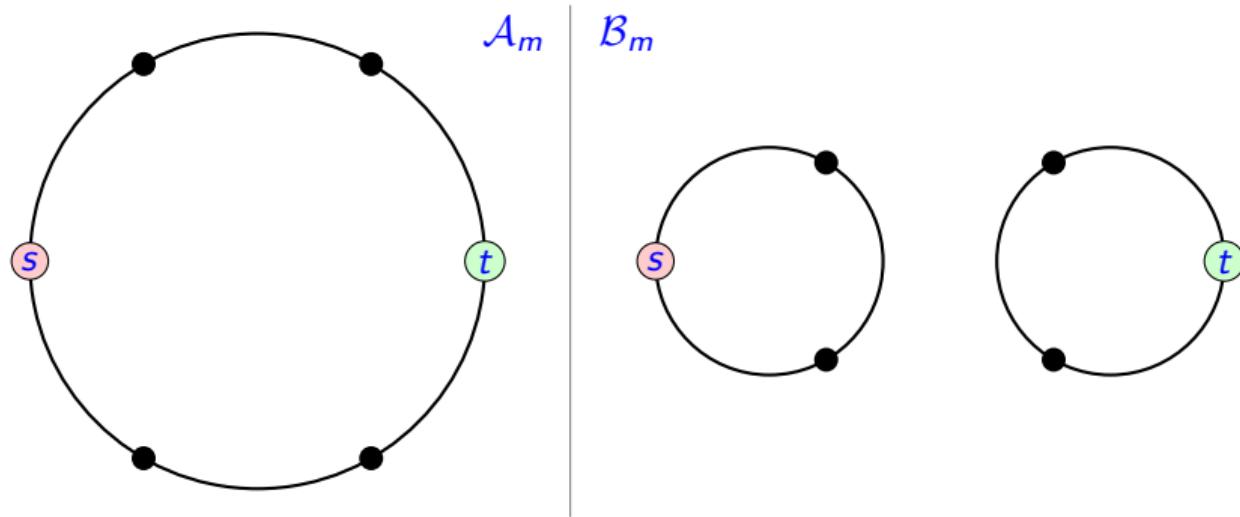
Das 0-Runden Spiel.



Für $m = 0$ muss in jede Richtung 1 Knoten zwischen s und t liegen.

Kreislänge für $m = 1$.

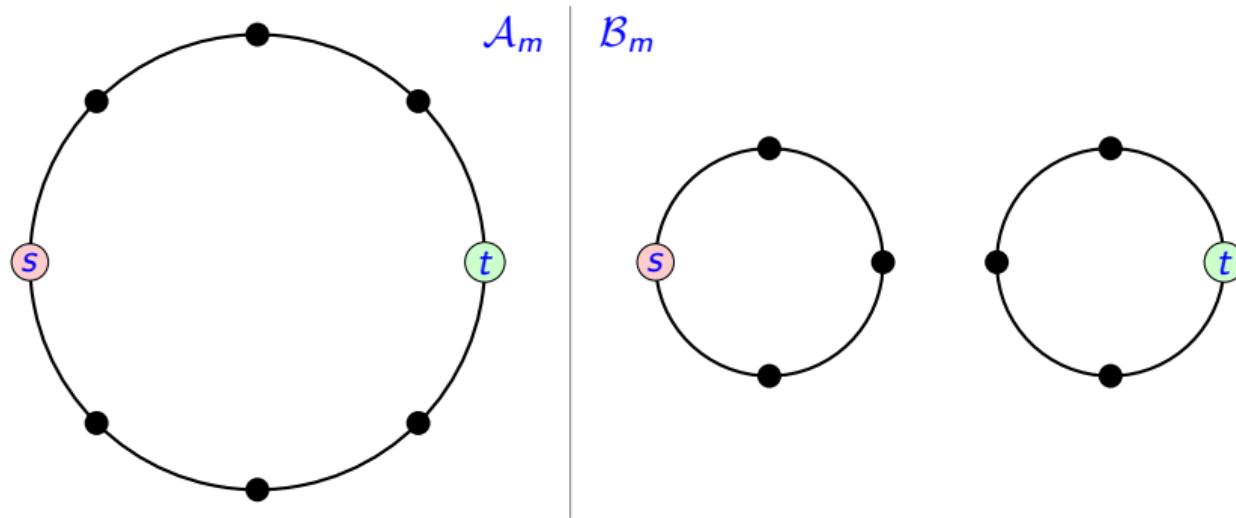
Das 1-Runden Spiel.



Für $m = 1$ müssen in jede Richtung 2 Knoten zwischen s und t liegen.

Kreislänge für $m = 2; 3$ oder 4 ?

Das 2-Runden Spiel. 3 oder 4 Knoten zwischen s und t ?

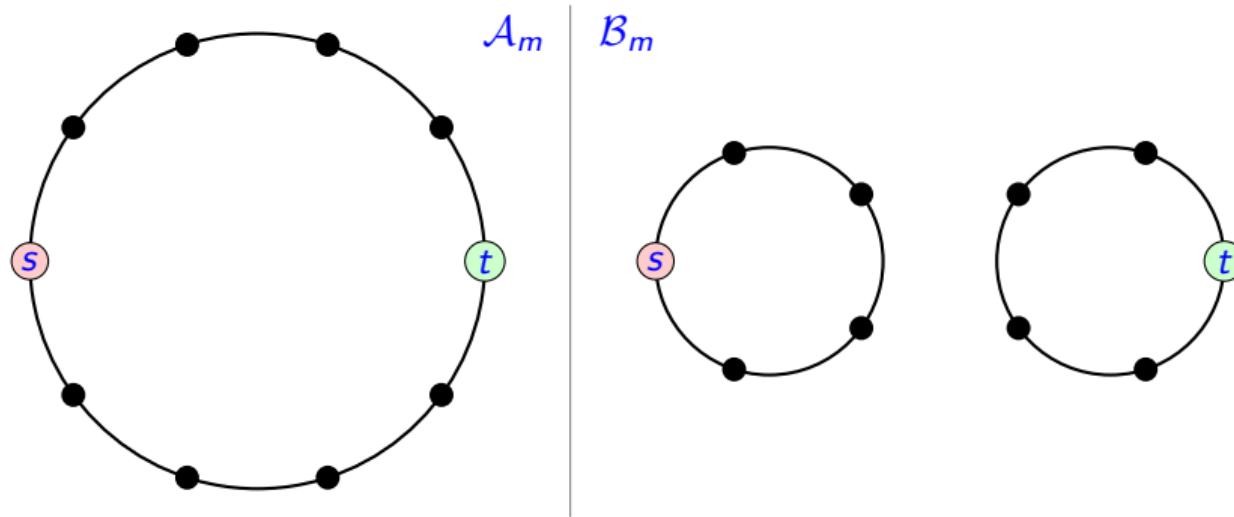


Für $m = 2$ müssen in jede Richtung 4 Knoten zwischen s und t liegen.

Beobachtung: Die Zahl der Knoten zwischen s und t verdoppelt sich in jedem Schritt.

Kreislänge für $m = 2; 3$ oder 4 ?

Das 2-Runden Spiel. 3 oder 4 Knoten zwischen s und t ?



Für $m = 2$ müssen in jede Richtung 4 Knoten zwischen s und t liegen.

Beobachtung: Die Zahl der Knoten zwischen s und t verdoppelt sich in jedem Schritt.

Beweis, dass REACH nicht definierbar ist

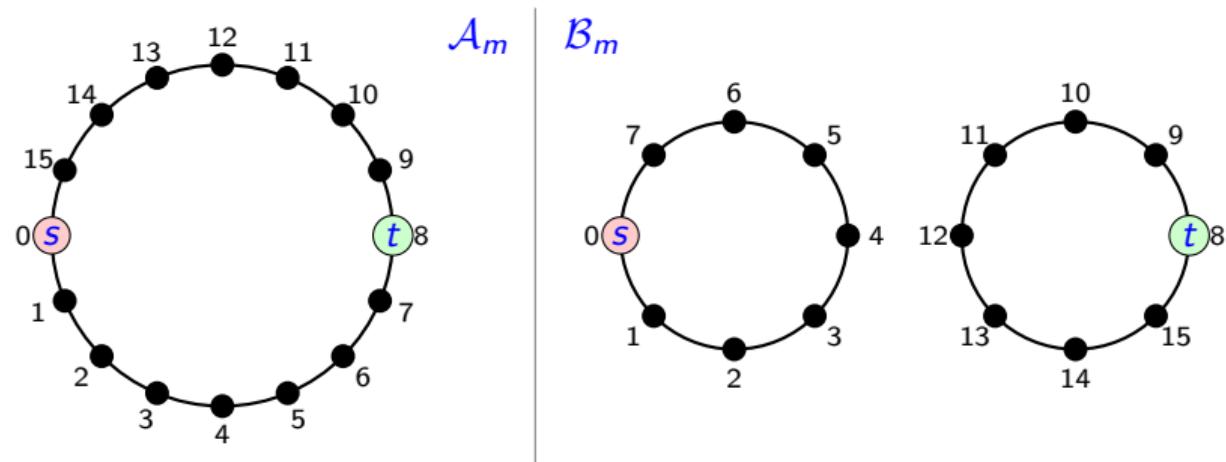
Behauptung. REACH ist nicht **FO**-definierbar.

Beweis. Für $m \geq 0$ sei $M = 2^{m+2}$ und

$\mathcal{A}_m := (A_m, E^{\mathcal{A}_m})$ ein Kreis der Länge M und

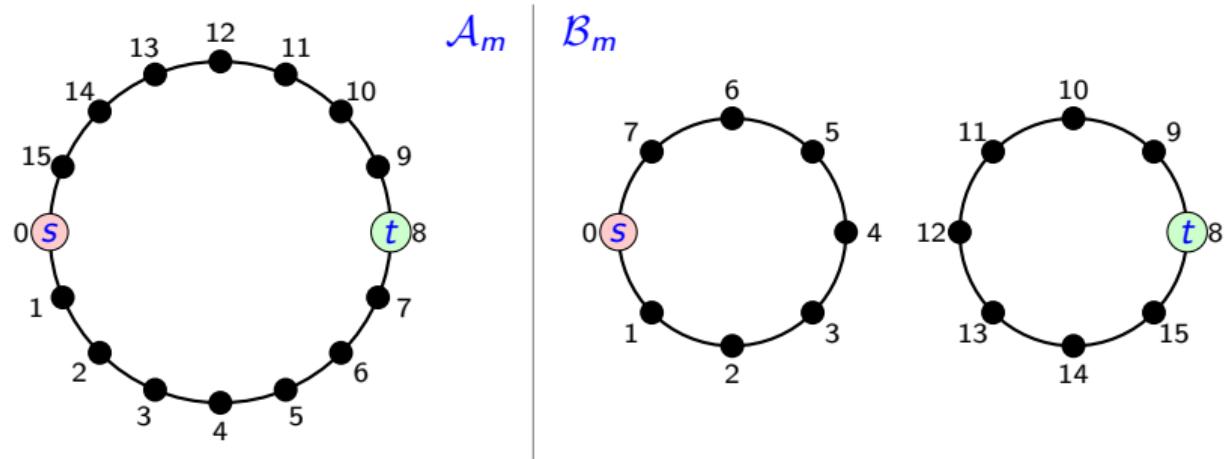
$\mathcal{B}_m := (B_m, E^{\mathcal{B}_m})$ zwei disjunkte Kreise der Länge $\frac{1}{2}M$.

Sei $A_m := \{a_0, \dots, a_M\}$ und $B_m := \{b_0, \dots, b_M\}$.



Beweis, dass REACH nicht definierbar ist

Behauptung. \mathbf{D} gewinnt $\mathfrak{G}_m(\mathcal{A}_m, a_0, a_{\frac{M}{2}}, \mathcal{B}_m, b_0, b_{\frac{M}{2}})$



Invariante $(*)$. Ang., nach i Runden sind $a_{j_1}, \dots, a_{j_i}, b_{j_1}, \dots, b_{j_i}$ gezogen.

Für alle $I, I' \in \{0, j_1, \dots, j_i, \frac{M}{2}\}$ und $r \leq 2^{m-i}$:

$$a_I \oplus r = a_{I'} \text{ gdw. } b_I \oplus r = b_{I'}$$

Notation.

$$a_j \oplus r = a_{j+r \bmod M}$$

$$b_j \oplus r = b_{j+r \bmod \frac{M}{2}}$$

$$b'_j \oplus r = b'_{j+r \bmod \frac{M}{2}}$$

Spielverlauf.

Nach Runde $i = m$:

$(*) \Rightarrow h : a_{j_i} \mapsto b_{j_i}$ part Isom.

Nach Runde $i = 0$:

$(*)$ nach Konst. erfüllt.

Wenn $(*)$ nach Runde i gilt,
kann \mathbf{D} $(*)$ nach Runde $i+1$
erfüllen.

Erreichbarkeit

Frage 2: Erreichbarkeit.

Signatur $\sigma := \{E\}$

E 2-stelliges Relationssymbol

Ist folgende Klasse FO-definierbar?

$\text{REACH} := \{(\mathcal{A}, s, t) : (\mathcal{A}, s, t) \text{ } \sigma\text{-Struktur, } s, t \in A,$
 $\text{es gibt einen Pfad von } s \text{ nach } t \}.$

Dahinter steht die Frage, ob die Prädikatenlogik **Schleifenkonstrukte** oder **Rekursion** ausdrücken kann.

Antwort. REACH ist nicht FO-definierbar.

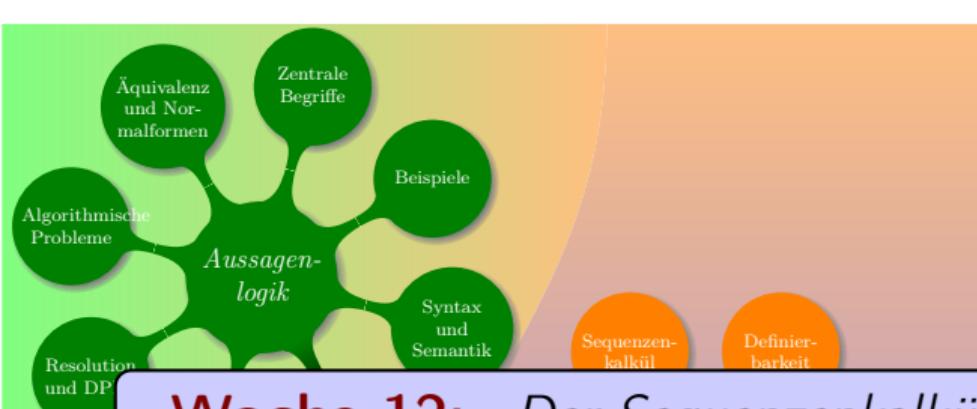
Beispiele Nicht-Definierbarer Klassen

Nicht-Definierbare Klassen.

- REACH ist nicht FO-definierbar.
- EVEN \leq ist nicht FO-definierbar.
- Die Sprache aller Wörter $w \in \{a, b\}^+$ mit genauso vielen a 's wie b 's ist nicht definierbar.
- Die Klasse aller unendlichen Strukturen.
- ...

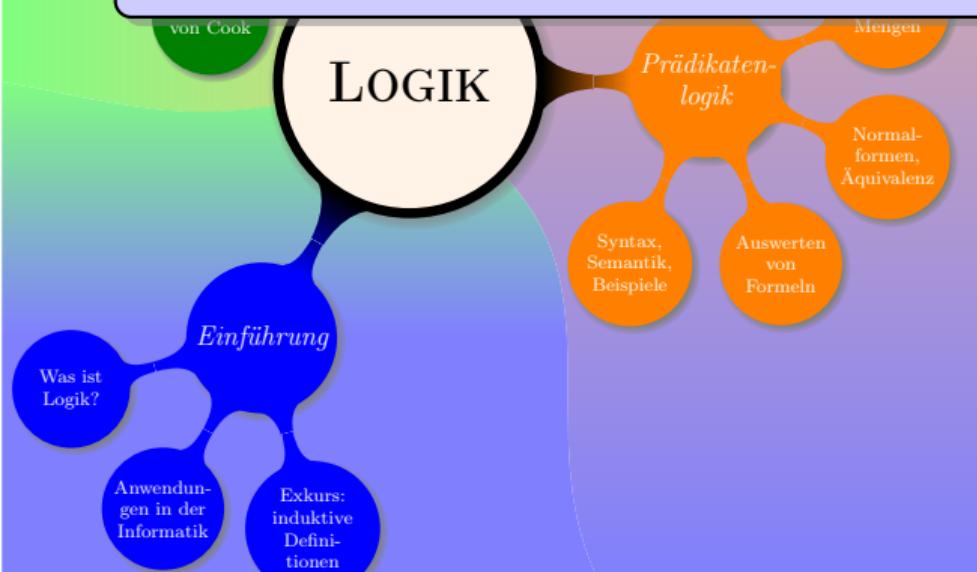
Nicht-Axiomatisierbare Klassen.

- Die Klasse aller endlichen Strukturen.
- ...



Nov.	17	18	19	20	21	22	23	<i>Einführung</i>
	24	25	26	27	28	29	30	<i>Aussagenlogik</i>
	31	1	2	3	4	5	6	<i>Normalformen</i>
	7	8	9	10	11	12	13	<i>Resolution</i>
Dez.	14	15	16	17	18	19	20	<i>Kompaktheit</i>
	21	22	23	24	25	26	27	<i>DPLL, Satz von Cook</i>

Woche 12: Der Sequenzenkalkül der Aussagenlogik



Jan.	5	6	7	8	9	10	11	<i>Prädikatenlogik</i>
	12	13	14	15	16	17	18	<i>Komplexität von FO</i>
	19	20	21	22	23	24	25	<i>Weihnachten</i>
	26	27	28	29	30	31	1	<i>Neujahr</i>
	2	3	4	5	6	7	8	<i>Normalformen</i>
	9	10	11	12	13	14	15	<i>Definierbarkeit</i>
	16	17	18	19	20	21	22	<i>EF-Spiele</i>
Feb.	23	24	25	26	27	28	29	<i>EF-Spiele</i>
	30	31	1	2	3	4	5	<i>Sequenzenkalkül AL</i>
	6	7	8	9	10	11	12	<i>Sequenzenkalkül FO</i>
	13	14	15	16	17	18	19	<i>Ausblick</i>

12.1 Einleitung

Erinnerung: Aussagenlogische Schlüsse

Logische Folgerung

Voraussetzungen:

- „Wenn G zshg. und kreisfrei, dann $|E(G)| = n - 1$.“
- „ G enthält $> n - 1$ Kanten.“
- „ G ist zusammenhängend.“

Folgerung: „ G enthält einen Kreis“?

$$\{Z \wedge \neg K \rightarrow N, \neg N, Z\} \Rightarrow K$$

Logische Äquivalenz.

$$(X \rightarrow Y) \iff (\neg Y \rightarrow \neg X)$$

Beweis. Sei β eine Belegung.

- $\beta \models X \rightarrow Y$
gdw. $\beta(Y) = 1$ oder $\beta(X) = 0$
- $\beta \not\models \neg Y$ oder $\beta \models \neg X$
gdw. $\beta \models \neg Y \rightarrow \neg X$

Beweisverfahren. Gibt es (syntaktische) Verfahren, um Folgerungen aus einer Menge von Voraussetzungen abzuleiten? Eine Methode,

- mit der nur korrekte Schlüsse gezogen werden können?
- die allgemein genug ist, alle gültigen Schlüsse ziehen zu können?

Eine solche Methode ist die Resolution.

Wir werden jetzt eine weitere Methode kennen lernen, den **Sequenzenkalkül**.

Sequenzenkalkülbeweise

Logische Folgerung.

$$(X \rightarrow Y) \models (\neg Y \rightarrow \neg X)$$

Regel

Sequenzenkalkülbeweis.

$$\frac{\begin{array}{c} (\Rightarrow \neg) \frac{\overline{X, \neg Y \Rightarrow X}}{\neg Y \Rightarrow X, \neg X} \\ (\Rightarrow \rightarrow) \frac{\overline{\Rightarrow X, (\neg Y \rightarrow \neg X)}}{(\neg Y \rightarrow \neg X)} \end{array}}{(X \rightarrow Y) \Rightarrow (\neg Y \rightarrow \neg X)}$$

Voraussetzungen

Logische Folgerung.

$$\left\{ \begin{array}{l} (Z \wedge \neg K) \rightarrow N, \\ \neg N, \\ Z \end{array} \right\} \Rightarrow K$$

Sequenzenkalkülbeweis.

$$\frac{\begin{array}{c} (\Rightarrow \wedge) \frac{\overline{\neg N, Z \Rightarrow K, Z, \neg K}}{\neg N, Z \Rightarrow K, Z \wedge \neg K} \\ (\neg \Rightarrow) \frac{\overline{N, Z \Rightarrow N, K}}{N, \neg N, Z \Rightarrow K} \end{array}}{(Z \wedge \neg K) \rightarrow N, \neg N, Z \Rightarrow K}$$

Folgerung

Logische Ableitungen

Der Sequenzenkalkül formalisiert die Art, in der wir aus Aussagen neue Folgerungen ableiten.

Wir führen dazu zunächst den Begriff der **Sequenz** ein.

Sequenz. Aussage der Form:

$$\underbrace{\left\{ \begin{array}{l} \text{„}G \text{ zshg. und kreisfrei } \Rightarrow |E(G)| = n - 1. \text{“} \\ \text{„}G \text{ enthält } > n - 1 \text{ Kanten.“} \\ \text{„}G \text{ ist zusammenhängend.“} \end{array} \right\}}_{\text{Voraussetzung}} \Rightarrow \underbrace{\text{„}G \text{ hat Kreis“}}_{\text{Konklusion}}$$

Sequenzen.

$$\{(X \rightarrow Y)\} \Rightarrow \{(\neg Y \rightarrow \neg X)\}$$

$$\{(Z \wedge \neg K) \rightarrow N, \neg N, Z\} \Rightarrow \{K\}$$

$$\Phi \Rightarrow \Delta$$

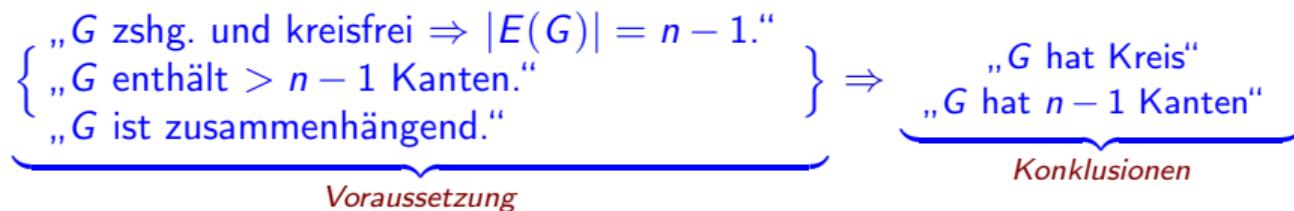
Die Bedeutung einer Sequenz ist, dass aus den Voraussetzungen die Konklusion geschlossen werden kann.

Logische Ableitungen

Wir werden auch unendliche Mengen von Voraussetzungen zulassen.

In der Mitte eines Beweises ist es oft nützlich, verschiedene Möglichkeiten zu betrachten, also Hypothesen aufzustellen. Wir werden daher zwischenzeitlich mehr als eine Konklusion betrachten.

Beispiel. Zum Beispiel könnten wir als Zwischenschritt annehmen



Bedeutung: Wenn die Voraussetzungen gelten, gilt **mindestens eine** Konklusion.

Wir können dann aus der Voraussetzung „G enthält $> n - 1$ Kanten“ die eigentliche Folgerung ableiten.

Ein Sequenzenkalkül

Wir stellen zunächst einen Sequenzenkalkül für die Aussagenlogik vor und erweitern ihn später auf die Prädikatenlogik.

- Es gibt viele verschiedene Sequenzenkalküle, abhängig von den in der Logik verwendeten Verknüpfungen.
- Wir werden hier einen Kalkül für Formeln ohne \leftrightarrow vorstellen.
- Wie bereits gezeigt, ist dies keine Einschränkung der Allgemeinheit.

12.2 Der Sequenzenkalkül der Aussagenlogik

Sequenzen

Definition.

1. Eine **Sequenz** ist eine Aussage der Form

$$\Phi \Rightarrow \Delta$$

für Multimengen $\Phi, \Delta \subseteq AL$.

Wir nennen Φ die **Voraussetzungen** und Δ die **Konklusionen**.

2. Eine Sequenz $\Phi \Rightarrow \Delta$ ist **gültig**, wenn jede Belegung β , die alle Formeln in Φ erfüllt, mindestens eine Formel in Δ erfüllt.
3. Ist $\Phi \Rightarrow \Delta$ nicht gültig, so gibt es eine Belegung β , die alle Formeln in Φ erfüllt, aber keine in Δ .

Wir sagen: β falsifiziert die Sequenz.

Sequenzen.

$$\{(X \rightarrow Y)\} \Rightarrow \{(\neg Y \rightarrow \neg X)\}$$

$$\{(Z \wedge \neg K) \rightarrow N, \neg N, Z\} \Rightarrow \{K\}$$

Gültige Sequenzen.

$$\{(X \wedge Y)\} \Rightarrow \{X, Y\}$$

$$\{(X \vee Y)\} \Rightarrow \{X, Y\}$$

$$\{(X \vee Y), Z, F\} \Rightarrow \{Z, X, Y\}$$

Nicht gültige Sequenzen.

$$\{(X \vee Y)\} \Rightarrow \{X \wedge Y\}$$

Ein Sequenzenkalkül

Definition.

1. Ein **Axiom** des Sequenzenkalküls ist eine Sequenz $\Phi \Rightarrow \Delta$, so dass $\Phi \cap \Delta \neq \emptyset$.

Beispiele.

$$\{A, \neg(X \vee Y), F\} \Rightarrow \{A, B\}$$

$$\{(X \wedge Y), F\} \Rightarrow \{(X \wedge Y)\}$$

$$\left\{ \begin{array}{l} \text{„G zshg. und kreisfrei $\Rightarrow |E(G)| = n - 1$.“} \\ \text{„$G$ enthält $> n - 1$ Kanten.“} \\ \text{„G ist zusammenhängend.“} \end{array} \right\} \Rightarrow \begin{array}{l} \text{„G hat Kreis“} \\ \text{„G ist zusammenhängend“} \end{array}$$

2. Ein **Theorem** ist eine gültige Sequenz $\emptyset \Rightarrow \{\psi\}$, d.h. eine Sequenz ohne Voraussetzungen mit genau einer Konklusion.

Beispiel. $\Rightarrow \{(X \rightarrow Y) \rightarrow (\neg X \vee Y)\}$

Definition.

Sequenz: $\Phi \Rightarrow \Delta$,

$\Phi, \Delta \subseteq AL$: Multimengen.

Φ : Voraussetzungen

Δ : Konklusionen.

$\Phi \Rightarrow \Delta$ gültig:

jedes β mit $\beta \models \Phi$
erfüllt ein $\delta \in \Delta$.

β falsifiziert $\Phi \Rightarrow \Delta$:

$\beta \models \Phi$ aber $\beta \not\models \delta$
für alle $\delta \in \Delta$.

Einfache Eigenschaften von Sequenzen

Beobachtung.

1. Jede Sequenz $\Phi \Rightarrow \Delta$ mit $\Phi \cap \Delta \neq \emptyset$ ist gültig.
2. Eine Sequenz $\Phi \Rightarrow \emptyset$ ist gültig gdw. Φ unerfüllbar ist.
3. Eine Sequenz $\emptyset \Rightarrow \Delta$ (mit Δ endl.) ist gültig gdw. $\bigvee \Delta$ gültig ist.

Definition.

- Eine Sequenz ist eine Aussage der Form $\Phi \Rightarrow \Delta$, für Multimengen $\Phi, \Delta \subseteq AL$. Wir nennen Φ die Voraussetzungen und Δ die Konklusionen.
- Eine Sequenz $\Phi \Rightarrow \Delta$ ist gültig, wenn jede Belegung β , die alle Formeln in Φ erfüllt, mindestens eine Formel in Δ erfüllt.
- Ist $\Phi \Rightarrow \Delta$ nicht gültig, so gibt es eine Belegung β , die alle Formeln in Φ erfüllt, aber keine in Δ .

Wir sagen: β falsifiziert die Sequenz.

Sequenzen.

$$\{(X \rightarrow Y)\} \Rightarrow \{(\neg Y \rightarrow \neg X)\}$$

$$\{(Z \wedge \neg K) \rightarrow N, \neg N, Z\} \Rightarrow \{K\}$$

Gültige Sequenzen.

$$\{(X \wedge Y)\} \Rightarrow \{X, Y\}$$

$$\{(X \vee Y)\} \Rightarrow \{X, Y\}$$

$$\{(X \vee Y), Z, F\} \Rightarrow \{Z, X, Y\}$$

Nicht gültige Sequenzen.

$$\{(X \vee Y)\} \Rightarrow \{X \wedge Y\}$$

Notation

Notation.

1. Φ, Δ, \dots bezeichnen Multimengen von Formeln.

(Die Reihenfolge der Formeln ist unwichtig, aber Formeln müssen mehrfach vorkommen dürfen.)

Dabei wird Δ immer endlich sein, Φ kann endlich oder unendlich sein.

2. Wir schreiben Multimengen $\{\varphi_1, \dots, \varphi_n\}$ als $\varphi_1, \dots, \varphi_n$
3. Wenn $\Phi \subseteq \text{AL}$ eine Menge ist und $\psi \in \text{AL}$, dann schreiben wir Φ, ψ statt $\Phi \cup \{\psi\}$.

Beispiel.

$$\underbrace{(X \rightarrow (Y \vee Z)), \quad X, \quad \neg Z}_{\Phi} \quad \Rightarrow \quad \Delta \overbrace{\quad \quad Y \quad \quad}^{\Delta}$$

Regeln

Das Ziel des Sequenzenkalküls ist das Ableiten gültiger Sequenzen aus anderen gültigen Sequenzen durch Regeln der Form

$$a) \quad \frac{\Phi \Rightarrow \Delta}{\Phi' \Rightarrow \Delta'} \quad \text{oder} \quad b) \quad \frac{\Phi_1 \Rightarrow \Delta_1 \quad \Phi_2 \Rightarrow \Delta_2}{\Phi' \Rightarrow \Delta'}$$

Wir nennen $\Phi \Rightarrow \Delta$ die **Prämissen** der Regel und $\Phi' \Rightarrow \Delta'$ ihre **Konsequenz**.

Beispiel.

$$(\wedge \Rightarrow) \quad \frac{p, q \Rightarrow q}{p \wedge q \Rightarrow q}$$

Idee. Wenn man für konkret gegebene Mengen $\Phi, \Delta, \Phi', \Delta'$ einmal gezeigt hat, dass $\Phi \Rightarrow \Delta$ gültig ist, dann stellt die Regel a) sicher, dass auch $\Phi' \Rightarrow \Delta'$ gültig ist.

Die Regeln des aussagenlogischen Sequenzenkalküls

$$(\neg \Rightarrow) \frac{\Phi \Rightarrow \Delta, \psi}{\Phi, \neg \psi \Rightarrow \Delta}$$

$$(\Rightarrow \neg) \frac{\Phi, \psi \Rightarrow \Delta}{\Phi \Rightarrow \Delta, \neg \psi}$$

$$(\wedge \Rightarrow) \frac{\Phi, \psi, \varphi \Rightarrow \Delta}{\Phi, \psi \wedge \varphi \Rightarrow \Delta}$$

$$(\Rightarrow \wedge) \frac{\Phi \Rightarrow \Delta, \psi \quad \Phi \Rightarrow \Delta, \varphi}{\Phi \Rightarrow \Delta, \psi \wedge \varphi}$$

$$(\vee \Rightarrow) \frac{\Phi, \varphi \Rightarrow \Delta \quad \Phi, \psi \Rightarrow \Delta}{\Phi, \varphi \vee \psi \Rightarrow \Delta}$$

$$(\Rightarrow \vee) \frac{\Phi \Rightarrow \Delta, \varphi, \psi}{\Phi \Rightarrow \Delta, \varphi \vee \psi}$$

$$(\rightarrow \Rightarrow) \frac{\Phi \Rightarrow \Delta, \varphi \quad \Phi, \psi \Rightarrow \Delta}{\Phi, \varphi \rightarrow \psi \Rightarrow \Delta}$$

$$(\Rightarrow \rightarrow) \frac{\Phi, \varphi \Rightarrow \Delta, \psi}{\Phi \Rightarrow \Delta, \varphi \rightarrow \psi}$$

Sequenzenkalkülbeweise

Wir werden Regeln zu komplizierteren Beweisen kombinieren.

Beispiel.

$$\begin{array}{c} (\Rightarrow \wedge) \frac{\overline{p, q, r \Rightarrow q} \quad \overline{p, q, r \Rightarrow r}}{p, q, r \Rightarrow q \wedge r} \\ (\wedge \Rightarrow) \frac{p, q, r \Rightarrow q \wedge r}{p \wedge q, r \Rightarrow q \wedge r} \end{array}$$

Aus diesem Beweis können wir folgende Aussage ablesen:

wenn die Sequenzen $p, q, r \Rightarrow q$ und $p, q, r \Rightarrow r$ gültig sind,
dann ist auch $p \wedge q, r \Rightarrow q \wedge r$ gültig. (wird später bewiesen)

Wir wissen bereits, dass $p, q, r \Rightarrow q$ und $p, q, r \Rightarrow r$ gültig sind, da sie Axiome sind.

Also können wir folgern, dass $p \wedge q, r \Rightarrow q \wedge r$ gültig ist.

SK-Regeln AL.

$$(\neg \Rightarrow) \frac{\Phi \Rightarrow \Delta, \psi}{\Phi, \neg \psi \Rightarrow \Delta}$$

$$(\Rightarrow \neg) \frac{\Phi, \psi \Rightarrow \Delta}{\Phi \Rightarrow \Delta, \neg \psi}$$

$$(\wedge \Rightarrow) \frac{\Phi, \psi, \varphi \Rightarrow \Delta}{\Phi, \psi \wedge \varphi \Rightarrow \Delta}$$

$$(\Rightarrow \wedge) \frac{\Phi \Rightarrow \Delta, \psi \quad \Phi \Rightarrow \Delta, \varphi}{\Phi \Rightarrow \Delta, \psi \wedge \varphi}$$

$$(\vee \Rightarrow) \frac{\Phi, \varphi \Rightarrow \Delta \quad \Phi, \psi \Rightarrow \Delta}{\Phi, \varphi \vee \psi \Rightarrow \Delta}$$

$$(\Rightarrow \vee) \frac{\Phi \Rightarrow \Delta, \varphi, \psi}{\Phi \Rightarrow \Delta, \varphi \vee \psi}$$

$$(\rightarrow \Rightarrow) \frac{\Phi \Rightarrow \Delta, \varphi \quad \Phi, \psi \Rightarrow \Delta}{\Phi, \varphi \rightarrow \psi \Rightarrow \Delta}$$

$$(\Rightarrow \rightarrow) \frac{\Phi, \varphi \Rightarrow \Delta, \psi}{\Phi \Rightarrow \Delta, \varphi \rightarrow \psi}$$

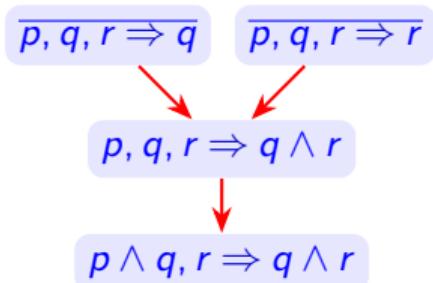
Beweisebäume

Definition. Ein **Beweis** im Sequenzenkalkül ist ein Baum, dessen Knoten wie folgt mit Sequenzen beschriftet sind.

- Die Blätter sind mit Axiomen beschriftet.
- Jeder innere Knoten ist mit der Konsequenz einer Regel beschriftet. Die Kinder des Knotens sind mit den Prämissen der Regel beschriftet. Jeder innere Knoten hat also entweder ein oder zwei Kinder.

Beispiel.

$$\begin{array}{c}
 (\Rightarrow \wedge) \frac{\overline{p, q, r \Rightarrow q} \quad \overline{p, q, r \Rightarrow r}}{p, q, r \Rightarrow q \wedge r} \\
 (\wedge \Rightarrow) \frac{p, q, r \Rightarrow q \wedge r}{\overline{p \wedge q, r \Rightarrow q \wedge r}}
 \end{array}$$



Notation. Wir „überstreichen“ Axiome um das Ende des Astes zu markieren. Dies entspricht der Beobachtung, dass Axiome aus der leeren Premissenmenge hergeleitet werden können.

Beweisbare Sequenzen

Definition. Eine Sequenz $\Phi \Rightarrow \Delta$ ist im Sequenzenkalkül **beweisbar**, oder kann **abgeleitet** werden, wenn sie als Beschriftung eines Knotens in einem Beweis vorkommt.

$$\begin{array}{c} (\Rightarrow \wedge) \frac{p, q, r \Rightarrow q \quad p, q, r \Rightarrow r}{(\wedge \Rightarrow) \frac{p, q, r \Rightarrow q \wedge r}{p \wedge q, r \Rightarrow q \wedge r}} \end{array}$$

Beispiel.

Die Sequenz $p \wedge q, r \Rightarrow q \wedge r$ ist im Sequenzenkalkül beweisbar.

Sequenzenkalkülbeweise. Wir werden später beweisen, dass alle ableitbaren Sequenzen gültig sind.

Dies wird dann zeigen, dass $p \wedge q, r \Rightarrow q \wedge r$ gültig ist und somit ist

$$(p \wedge q) \wedge r \rightarrow (q \wedge r)$$

eine Tautologie.

In dieser Art wird der Sequenzenkalkül zum Beweis logisch korrekter Schlüsse benutzt.

12.3 Beispiele für Sequenzenkalkülbeweise

Beispiel: Doppelte Negation

Wir beweisen die Äquivalenz $\varphi \equiv \neg\neg\varphi$.

$$(\Rightarrow \neg) \frac{\overline{\varphi \Rightarrow \varphi}}{\Rightarrow \neg\varphi, \varphi}$$

$$(\neg \Rightarrow) \frac{\overline{\varphi \Rightarrow \varphi}}{\neg\varphi, \varphi \Rightarrow}$$

$$(\Rightarrow \neg) \frac{\overline{\neg\varphi, \varphi \Rightarrow}}{\varphi \Rightarrow \neg\neg\varphi}$$

SK-Regeln AL.

$$(\neg \Rightarrow) \frac{\Phi \Rightarrow \Delta, \psi}{\Phi, \neg\psi \Rightarrow \Delta}$$

$$(\Rightarrow \neg) \frac{\Phi, \psi \Rightarrow \Delta}{\Phi \Rightarrow \Delta, \neg\psi}$$

$$(\wedge \Rightarrow) \frac{\Phi, \psi, \varphi \Rightarrow \Delta}{\Phi, \psi \wedge \varphi \Rightarrow \Delta}$$

$$(\Rightarrow \wedge) \frac{\Phi \Rightarrow \Delta, \psi \quad \Phi \Rightarrow \Delta, \varphi}{\Phi \Rightarrow \Delta, \psi \wedge \varphi}$$

$$(\vee \Rightarrow) \frac{\Phi, \varphi \Rightarrow \Delta \quad \Phi, \psi \Rightarrow \Delta}{\Phi, \varphi \vee \psi \Rightarrow \Delta}$$

$$(\Rightarrow \vee) \frac{\Phi \Rightarrow \Delta, \varphi, \psi}{\Phi \Rightarrow \Delta, \varphi \vee \psi}$$

$$(\rightarrow \Rightarrow) \frac{\Phi \Rightarrow \Delta, \varphi \quad \Phi, \psi \Rightarrow \Delta}{\Phi, \varphi \rightarrow \psi \Rightarrow \Delta}$$

$$(\Rightarrow \rightarrow) \frac{\Phi, \varphi \Rightarrow \Delta, \psi}{\Phi \Rightarrow \Delta, \varphi \rightarrow \psi}$$

Beispiel: Tertium Non Datur

Beispiel. Wir wollen beweisen, dass $\varphi \vee \neg\varphi$ allgemeingültig ist.

$$\begin{array}{c} (\Rightarrow \neg) \frac{\overline{\varphi \Rightarrow \varphi}}{\Rightarrow \neg\varphi, \varphi} \\ (\Rightarrow \vee) \frac{\overline{\neg\varphi \vee \varphi}}{\Rightarrow \neg\varphi \vee \varphi} \end{array}$$

Das Prinzip $\neg\varphi \vee \varphi$ ist eine der wichtigsten Grundprinzipien der Logik und wird **tertium non datur** genannt. („law of excluded middle“)

SK-Regeln AL.

$$(\neg \Rightarrow) \frac{\Phi \Rightarrow \Delta, \psi}{\Phi, \neg\psi \Rightarrow \Delta}$$

$$(\Rightarrow \neg) \frac{\Phi, \psi \Rightarrow \Delta}{\Phi \Rightarrow \Delta, \neg\psi}$$

$$(\wedge \Rightarrow) \frac{\Phi, \psi, \varphi \Rightarrow \Delta}{\Phi, \psi \wedge \varphi \Rightarrow \Delta}$$

$$(\Rightarrow \wedge) \frac{\Phi \Rightarrow \Delta, \psi \quad \Phi \Rightarrow \Delta, \varphi}{\Phi \Rightarrow \Delta, \psi \wedge \varphi}$$

$$(\vee \Rightarrow) \frac{\Phi, \varphi \Rightarrow \Delta \quad \Phi, \psi \Rightarrow \Delta}{\Phi, \varphi \vee \psi \Rightarrow \Delta}$$

$$(\Rightarrow \vee) \frac{\Phi \Rightarrow \Delta, \varphi, \psi}{\Phi \Rightarrow \Delta, \varphi \vee \psi}$$

$$(\rightarrow \Rightarrow) \frac{\Phi \Rightarrow \Delta, \varphi \quad \Phi, \psi \Rightarrow \Delta}{\Phi, \varphi \rightarrow \psi \Rightarrow \Delta}$$

$$(\Rightarrow \rightarrow) \frac{\Phi, \varphi \Rightarrow \Delta, \psi}{\Phi \Rightarrow \Delta, \varphi \rightarrow \psi}$$

Beispiel: Kommutativität der Disjunktion

Beispiel. Kommutativität der Disjunktion

$$\begin{array}{c}
 (\Rightarrow \vee) \frac{\overline{\varphi \Rightarrow \psi, \varphi}}{\varphi \Rightarrow \psi \vee \varphi} \quad (\Rightarrow \vee) \frac{\overline{\psi \Rightarrow \psi, \varphi}}{\psi \Rightarrow \psi \vee \varphi} \\
 (\vee \Rightarrow) \frac{\varphi \vee \psi \Rightarrow \psi \vee \varphi}{\varphi \vee \psi \Rightarrow \psi \vee \varphi}
 \end{array}$$

SK-Regeln AL.	
($\neg \Rightarrow$)	$\frac{\Phi \Rightarrow \Delta, \psi}{\Phi, \neg \psi \Rightarrow \Delta}$
($\Rightarrow \neg$)	$\frac{\Phi, \psi \Rightarrow \Delta}{\Phi \Rightarrow \Delta, \neg \psi}$
($\wedge \Rightarrow$)	$\frac{\Phi, \psi, \varphi \Rightarrow \Delta}{\Phi, \psi \wedge \varphi \Rightarrow \Delta}$
($\Rightarrow \wedge$)	$\frac{\Phi \Rightarrow \Delta, \psi \quad \Phi \Rightarrow \Delta, \varphi}{\Phi \Rightarrow \Delta, \psi \wedge \varphi}$
($\vee \Rightarrow$)	$\frac{\Phi, \varphi \Rightarrow \Delta \quad \Phi, \psi \Rightarrow \Delta}{\Phi, \varphi \vee \psi \Rightarrow \Delta}$
($\Rightarrow \vee$)	$\frac{\Phi \Rightarrow \Delta, \varphi, \psi}{\Phi \Rightarrow \Delta, \varphi \vee \psi}$
($\rightarrow \Rightarrow$)	$\frac{\Phi \Rightarrow \Delta, \varphi \quad \Phi, \psi \Rightarrow \Delta}{\Phi, \varphi \rightarrow \psi \Rightarrow \Delta}$
($\Rightarrow \rightarrow$)	$\frac{\Phi, \varphi \Rightarrow \Delta, \psi}{\Phi \Rightarrow \Delta, \varphi \rightarrow \psi}$

Beispiel

Das Ziel ist der Beweis folgender Aussage:

$(X \wedge Y) \rightarrow (X \vee Y)$ ist eine Tautologie.

Wir beweisen, dass die Sequenz $\Rightarrow (X \wedge Y) \rightarrow (X \vee Y)$ gültig ist.

Beweis.

$$\begin{array}{c}
 (\Rightarrow \vee) \frac{}{X, Y \Rightarrow X, Y} \\
 (\wedge \Rightarrow) \frac{}{X, Y \Rightarrow X \vee Y} \\
 (\Rightarrow \rightarrow) \frac{X \wedge Y \Rightarrow X \vee Y}{\Rightarrow (X \wedge Y) \rightarrow (X \vee Y)}
 \end{array}$$

SK-Regeln AL.

$$(\neg \Rightarrow) \frac{\Phi \Rightarrow \Delta, \psi}{\Phi, \neg \psi \Rightarrow \Delta}$$

$$(\Rightarrow \neg) \frac{\Phi, \psi \Rightarrow \Delta}{\Phi \Rightarrow \Delta, \neg \psi}$$

$$(\wedge \Rightarrow) \frac{\Phi, \psi, \varphi \Rightarrow \Delta}{\Phi, \psi \wedge \varphi \Rightarrow \Delta}$$

$$(\Rightarrow \wedge) \frac{\Phi \Rightarrow \Delta, \psi \quad \Phi \Rightarrow \Delta, \varphi}{\Phi \Rightarrow \Delta, \psi \wedge \varphi}$$

$$(\vee \Rightarrow) \frac{\Phi, \varphi \Rightarrow \Delta \quad \Phi, \psi \Rightarrow \Delta}{\Phi, \varphi \vee \psi \Rightarrow \Delta}$$

$$(\Rightarrow \vee) \frac{\Phi \Rightarrow \Delta, \varphi, \psi}{\Phi \Rightarrow \Delta, \varphi \vee \psi}$$

$$(\rightarrow \Rightarrow) \frac{\Phi \Rightarrow \Delta, \varphi \quad \Phi, \psi \Rightarrow \Delta}{\Phi, \varphi \rightarrow \psi \Rightarrow \Delta}$$

$$(\Rightarrow \rightarrow) \frac{\Phi, \varphi \Rightarrow \Delta, \psi}{\Phi \Rightarrow \Delta, \varphi \rightarrow \psi}$$

12.4 Korrektheit und Vollständigkeit des Sequenzenkalküls

Vollständigkeit und Korrektheit

Der Sequenzenkalkül ist vollständig und korrekt.

Korrektheit.

Nur gültige Sequenzen können im Sequenzenkalkül hergeleitet werden.

Vollständigkeit.

Alle gültigen Sequenzen können im Sequenzenkalkül hergeleitet werden.

Korrektheit des Sequenzenkalküls

Definition (Korrektheit).

1. Eine Regel

$$\frac{\Phi \Rightarrow \Delta}{\Phi' \Rightarrow \Delta'}$$

ist **korrekt**, wenn für alle Multimengen $\Phi, \Phi' \subseteq \text{AL}$ und endlichen Multimengen $\Delta, \Delta' \subseteq \text{AL}$ gilt:

Wenn $\Phi \Rightarrow \Delta$ gültig ist, dann ist auch $\Phi' \Rightarrow \Delta'$ gültig.

2. Eine Regel

$$\frac{\Phi_1 \Rightarrow \Delta_1 \quad \Phi_2 \Rightarrow \Delta_2}{\Phi \Rightarrow \Delta}$$

ist **korrekt**, wenn für alle Multimengen $\Phi, \Phi_1, \Phi_2 \subseteq \text{AL}$ und endliche Multimengen $\Delta, \Delta_1, \Delta_2 \subseteq \text{AL}$ aus der Gültigkeit von $\Phi_1 \Rightarrow \Delta_1$ und $\Phi_2 \Rightarrow \Delta_2$ die Gültigkeit von $\Phi \Rightarrow \Delta$ folgt.

Korrektheit des Sequenzenkalküls

Lemma. Die Regeln des Sequenzenkalküls sind korrekt.

Allgemeine Beweisstrategie. $(\cdot) \frac{\Phi \Rightarrow \Delta}{\Phi' \Rightarrow \Delta'}$

Voraussetzung. Nehme an, dass $\Phi \Rightarrow \Delta$ gültig ist.

Zu zeigen. Zeige, dass jede Belegung, die alle Formeln in Φ' erfüllt, auch mindestens eine Formel in Δ' erfüllt.

Beweisansatz.

- Sei β eine Belegung mit $\beta \models \Phi'$.
- Zeige, dass $\beta \models \Phi$.
- Daraus folgt nach Voraussetzung, dass es ein $\delta \in \Delta$ gibt mit $\beta \models \delta$.
Hierbei wird die Gültigkeit von $\Phi \Rightarrow \Delta$ verwendet.
- Zeige mit Hilfe dieser Aussage, dass es ein $\delta' \in \Delta'$ gibt mit $\beta \models \delta'$.

Korrekttheit des Sequenzenkalküls

Wir werden eine etwas stärkere Behauptung beweisen.

Definition.

Sei $\Phi \Rightarrow \Delta$ eine Sequenz und β eine zu $\Phi \cup \Delta$ passende Belegung.

1. β falsifiziert die Sequenz $\Phi \Rightarrow \Delta$, falls sie alle $\varphi \in \Phi$ erfüllt aber kein $\delta \in \Delta$.
2. β erfüllt die Sequenz $\Phi \Rightarrow \Delta$, falls sie ein $\varphi \in \Phi$ nicht erfüllt oder aber mindestens ein $\delta \in \Delta$ erfüllt.

Lemma. Für jede Regel des Sequenzenkalküls und jede Belegung β die zu allen Formeln der Regel passt gilt:

β erfüllt die Konsequenz einer Regel genau dann, wenn β alle Prämissen erfüllt.

Beweis des Lemmas

$$\text{Fall 1: (Disjunktion) . } (\Rightarrow \vee) \frac{\Phi \Rightarrow \Delta, \varphi, \psi}{\Phi \Rightarrow \Delta, \varphi \vee \psi}$$

Beweis. Sei β eine zu allen Formeln der Regel passende Belegung.

Zeige. β falsifiziert $\Phi \Rightarrow \Delta, \varphi, \psi$ gdw. β falsifiziert $\Phi \Rightarrow \Delta, \varphi \vee \psi$.

\Rightarrow . Angenommen, β falsifiziert $\Phi \Rightarrow \Delta, \varphi, \psi$.

Also gilt $\beta \models \Phi$ aber $\beta \not\models \delta$ für alle $\delta \in \Delta, \varphi, \psi$.

Dann gilt aber $\beta \not\models \varphi \vee \psi$ und somit falsifiziert β die Sequenz $\Phi \Rightarrow \Delta, \varphi \vee \psi$.

\Leftarrow . Angenommen, β falsifiziert $\Phi \Rightarrow \Delta, \varphi \vee \psi$.

Also gilt $\beta \models \Phi$ aber $\beta \not\models \delta$ für alle $\delta \in \Delta, \varphi \vee \psi$.

Dann gilt aber $\beta \not\models \varphi$ und $\beta \not\models \psi$ und somit falsifiziert β die Sequenz $\Phi \Rightarrow \Delta, \varphi, \psi$.

Definition.

β falsifiziert $\Phi \Rightarrow \Delta$:

$\beta \models \Phi$ aber

$\beta \not\models \delta$ für alle $\delta \in \Delta$.

β erfüllt $\Phi \Rightarrow \Delta$:

$\beta \not\models \varphi$ für ein $\varphi \in \Phi$ oder

$\beta \models \delta$ für ein $\delta \in \Delta$

Beweis des Lemmas

Fall 2. Disjunktion links. $(\vee \Rightarrow) \frac{\Phi, \varphi \Rightarrow \Delta \quad \Phi, \psi \Rightarrow \Delta}{\Phi, \varphi \vee \psi \Rightarrow \Delta}$

Beweis. Sei β eine zu allen Formeln der Regel passende Belegung.

Zeige: β falsifiziert $\Phi, \varphi \vee \psi \Rightarrow \Delta$ gdw. β falsifiziert $\Phi, \varphi \Rightarrow \Delta$ oder $\Phi, \psi \Rightarrow \Delta$.

\Rightarrow . Wenn $\beta \models \Phi, \varphi \vee \psi \Rightarrow \Delta$ falsifiziert, gilt $\beta \models \varphi'$ für alle $\varphi' \in \Phi \cup \{\varphi \vee \psi\}$ und $\beta \not\models \delta$ für alle $\delta \in \Delta$.

Es gilt also $\beta \models \varphi$ oder $\beta \models \psi$. Im ersten Fall wird aber die erste Prämisse falsifiziert, im zweiten Fall die zweite Prämisse.

\Leftarrow . Wenn β eine der Sequenzen $\Phi, \varphi \Rightarrow \Delta$ oder $\Phi, \psi \Rightarrow \Delta$ falsifiziert, dann gilt $\beta \models \varphi'$ für alle $\varphi' \in \Phi$ sowie $\beta \models \varphi$ oder $\beta \models \psi$. Außerdem gilt $\beta \not\models \delta$ für alle $\delta \in \Delta$.

Es gilt also auch $\beta \models \varphi \vee \psi$ und somit wird die Konklusion falsifiziert.

Korrekttheit des Sequenzenkalküls

Lemma. Für jede Regel des Sequenzenkalküls und jede Belegung β die zu allen Formeln der Regel passt gilt:

β erfüllt die Konsequenz einer Regel genau dann, wenn β alle Prämissen erfüllt.

Folgerung. Die Regeln des Sequenzenkalküls sind korrekt.

Theorem. Jede im Sequenzenkalkül beweisbare Sequenz ist gültig.

Sequenzenkalkül

Ein Vorteil des Sequenzenkalküls ist, dass er eine systematische Beweissuche erlaubt.

Wir werden einen Algorithmus angeben, der zu jeder Sequenz $\Phi \Rightarrow \Delta$ entweder,

- einen Beweis im Sequenzenkalkül konstruiert oder
- eine Belegung, die alle Konsequenzen falsifiziert aber alle Voraussetzungen erfüllt.

Im zweiten Fall ist die Belegung ein Gegenbeispiel zur Gültigkeit der Sequenz.

Beispiel

Erinnern wir uns an den Sequenzenkalkülbeweis der folgenden Aussage.

$$\Rightarrow (X \rightarrow Y) \rightarrow (\neg Y \rightarrow \neg X).$$

Beweis.

$$\begin{array}{c} (\Rightarrow \neg) \frac{\overline{X, \neg Y \Rightarrow X}}{\neg Y \Rightarrow X, \neg X} \quad (\neg \Rightarrow) \frac{\overline{Y \Rightarrow Y, \neg X}}{Y, \neg Y \Rightarrow \neg X} \\ (\Rightarrow \Rightarrow) \frac{\overline{\Rightarrow X, (\neg Y \rightarrow \neg X)}}{(\Rightarrow \rightarrow) \frac{\overline{(X \rightarrow Y) \Rightarrow (\neg Y \rightarrow \neg X)}}{(\Rightarrow \rightarrow) \frac{\overline{\Rightarrow (X \rightarrow Y) \rightarrow (\neg Y \rightarrow \neg X)}}{}} \end{array}$$

Beobachtung.

- Es kann mehrere Beweise der gleichen Sequenz geben.
- Aber für jede nicht-atomare Formel in einer Sequenz gibt es genau eine anwendbare Regel.

Beweis.

$$\begin{array}{c} (\Rightarrow \Rightarrow) \frac{\overline{X \Rightarrow X, Y} \quad \overline{X, Y \Rightarrow Y}}{(X \rightarrow Y), X \Rightarrow Y} \\ (\Rightarrow \neg) \frac{\overline{(X \rightarrow Y) \Rightarrow \neg X, Y}}{(\neg \Rightarrow) \frac{\overline{(X \rightarrow Y), \neg Y \Rightarrow \neg X}}{(\Rightarrow \rightarrow) \frac{\overline{(X \rightarrow Y) \Rightarrow (\neg Y \rightarrow \neg X)}}{(\Rightarrow \rightarrow) \frac{\overline{\Rightarrow (X \rightarrow Y) \rightarrow (\neg Y \rightarrow \neg X)}}{}}} \end{array}$$

Beweissuche im Sequenzenkalkül

Wir wollen die folgende Sequenz beweisen $X \vee Y \Rightarrow X \wedge Y$.

Beweisversuch.

$$\frac{\begin{array}{c} (\vee \Rightarrow) \frac{\begin{array}{c} \overline{X \Rightarrow X} & Y \Rightarrow X \\ \overline{X \vee Y \Rightarrow X} & \end{array}}{(\Rightarrow \wedge) \frac{\overline{X \Rightarrow Y} & \overline{Y \Rightarrow Y}}{X \vee Y \Rightarrow X \wedge Y}} \end{array}}{} \quad$$

- Der Beweisversuch liefert einen Baum dessen Blätter alle mit Sequenzen beschriftet sind, die nur noch atomare Formeln enthalten. Es sind also keine weiteren Regeln mehr anwendbar.
- Aber nur zwei Blätter sind Axiome. Die anderen können falsifiziert werden.
- Wir wählen z.B. β durch $\beta(X) := 1$ und $\beta(Y) := 0$.
- Dies falsifiziert ein Blatt aber auch die Originalsequenz.
- D.h., der Versuch die Sequenz $X \vee Y \Rightarrow X \wedge Y$ zu beweisen führt zu einer falsifizierenden Belegung eines Blattes.
- Nach eben bewiesenem Lemma, falsifiziert dies auch die Originalsequenz.

Lemma. Für jede Regel und jede passende Belegung β gilt:

β erfüllt die Konsequenz der Regel gdw.

β alle Prämissen erfüllt.

Vollständigkeit des Sequenzenkalküls

Definition. Ein Ableitungsbaum \mathcal{T} einer Sequenz \mathcal{S} ist ein Baum, in dem

- die Wurzel mit \mathcal{S} beschriftet ist und
- jeder innere Knoten mit der Konsequenz einer Regel und dessen Kinder mit den Prämissen dieser Regel beschriftet sind.

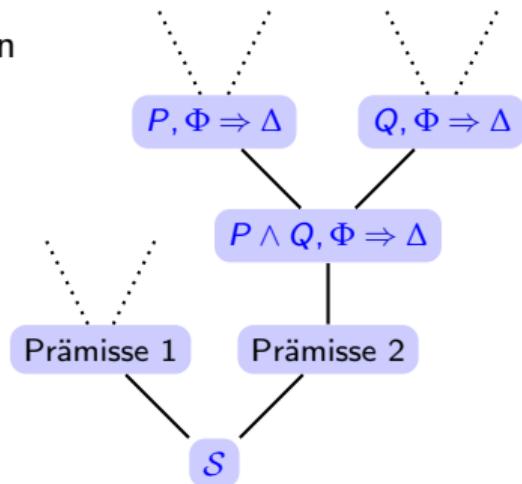
Ein Blatt von \mathcal{T} ist

- *positiv*, wenn es mit einem Axiom beschriftet ist und
- *negativ*, wenn es mit einer Sequenz $\Phi \Rightarrow \Delta$ beschriftet ist, so dass Φ und Δ disjunkte Mengen von Variablen sind.

\mathcal{T} ist *vollständig*, wenn alle Blätter positiv oder negativ sind.

Ein vollständiger Ableitungsbaum für \mathcal{S} ist eine *Widerlegung*, wenn er ein negatives Blatt enthält.

Bemerkung. Ein Beweis für \mathcal{S} ist ein vollständiger Ableitungsbaum dessen Blätter alle positiv sind.



Algorithmus zur Konstruktion von Beweisen

Eingabe. Eine Sequenz $\Phi \Rightarrow \Delta$

Ausgabe. Beweis oder falsifizierende Belegung für $\Phi \Rightarrow \Delta$.

Algorithmus. Konstruiere Ableitungsbaum \mathcal{T} für \mathcal{S} induktiv.

Initialisiere \mathcal{T} als Baum mit einem mit $\Phi \Rightarrow \Delta$ beschrifteten Knoten.

while es gibt unmarkiertes Blatt t **do** ($\Phi' \Rightarrow \Delta'$ Beschriftung von t)

if t negativ **then return** β mit $\beta(X) = 1$ für $X \in \Phi'$ und $\beta(X) = 0$ sonst
 else if t positiv **then** markiere t mit (+).

else

 wähle nicht-atomare Formel $\varphi \in \Phi' \cup \Delta'$ und wende Regel auf φ an.

 füge für jede Prämisse P ein mit P beschriftetes Kind von t hinzu.

od

Sind alle Blätter markiert, gib \mathcal{T} als Beweis für $\Phi \Rightarrow \Delta$ zurück.

Vollständigkeit des Sequenzenkalküls

Theorem. Der Algorithmus terminiert auf jeder endlichen Sequenz $\mathcal{S} := \Phi \Rightarrow \Delta$ (d.h. $\Phi \cup \Delta$ endlich).

Wenn $\Phi \Rightarrow \Delta$ gültig ist, liefert der Algorithmus einen Beweis, anderenfalls gibt er eine Widerlegung zurück.

Beweis. Wir zeigen zunächst **Terminierung**.

Für $\varphi \in AL$ sei $h(\varphi)$ die Zahl der Verknüpfungen $\wedge, \vee, \neg, \rightarrow$ in φ .

Die **Komplexität** einer Sequenz $\Phi \Rightarrow \Delta$ ist definiert als

$$\sum\{h(\varphi) : \varphi \in \Phi \cup \Delta\}.$$

Beobachtung. Ist K die Kosequenz und sind P bzw. P_1, P_2 die Prämissen einer Regel, dann gilt:

$$h(K) > h(P) \text{ bzw. } h(K) > \max\{h(P_1), h(P_2)\}.$$

Die Komplexität der Kinder jedes Knotens nimmt also ab. Daher muss der Algorithmus terminieren.

SK-Regeln AL.

$$(\neg \Rightarrow) \frac{\Phi \Rightarrow \Delta, \psi}{\Phi, \neg \psi \Rightarrow \Delta}$$

$$(\Rightarrow \neg) \frac{\Phi, \psi \Rightarrow \Delta}{\Phi \Rightarrow \Delta, \neg \psi}$$

$$(\wedge \Rightarrow) \frac{\Phi, \psi, \varphi \Rightarrow \Delta}{\Phi, \psi \wedge \varphi \Rightarrow \Delta}$$

$$(\Rightarrow \wedge) \frac{\Phi \Rightarrow \Delta, \psi \quad \Phi \Rightarrow \Delta, \varphi}{\Phi \Rightarrow \Delta, \psi \wedge \varphi}$$

$$(\vee \Rightarrow) \frac{\Phi, \varphi \Rightarrow \Delta \quad \Phi, \psi \Rightarrow \Delta}{\Phi, \varphi \vee \psi \Rightarrow \Delta}$$

$$(\Rightarrow \vee) \frac{\Phi \Rightarrow \Delta, \varphi, \psi}{\Phi \Rightarrow \Delta, \varphi \vee \psi}$$

$$(\rightarrow \Rightarrow) \frac{\Phi \Rightarrow \Delta, \varphi \quad \Phi, \psi \Rightarrow \Delta}{\Phi, \varphi \rightarrow \psi \Rightarrow \Delta}$$

$$(\Rightarrow \rightarrow) \frac{\Phi, \varphi \Rightarrow \Delta, \psi}{\Phi \Rightarrow \Delta, \varphi \rightarrow \psi}$$

Vollständigkeit des Sequenzenkalküls

Theorem. Der Algorithmus terminiert auf jeder endlichen Sequenz $\mathcal{S} := \Phi \Rightarrow \Delta$ (d.h. $\Phi \cup \Delta$ endlich).

Wenn $\Phi \Rightarrow \Delta$ gültig ist, liefert der Algorithmus einen Beweis, anderenfalls gibt er eine Widerlegung zurück.

Beweis (Teil 2). Der Algorithmus terminiert,

- bei einem negativen Blatt und konstruiert daraus eine falsifizierende Belegung für \mathcal{S} .
- wenn alle Blätter positiv sind. Dann ist der Ableitungsbaum ein Beweis von \mathcal{S} und die Sequenz damit gültig. \square

SK-Regeln AL.

$$(\neg \Rightarrow) \frac{\Phi \Rightarrow \Delta, \psi}{\Phi, \neg \psi \Rightarrow \Delta}$$

$$(\Rightarrow \neg) \frac{\Phi, \psi \Rightarrow \Delta}{\Phi \Rightarrow \Delta, \neg \psi}$$

$$(\wedge \Rightarrow) \frac{\Phi, \psi, \varphi \Rightarrow \Delta}{\Phi, \psi \wedge \varphi \Rightarrow \Delta}$$

$$(\Rightarrow \wedge) \frac{\Phi \Rightarrow \Delta, \psi \quad \Phi \Rightarrow \Delta, \varphi}{\Phi \Rightarrow \Delta, \psi \wedge \varphi}$$

$$(\vee \Rightarrow) \frac{\Phi, \varphi \Rightarrow \Delta \quad \Phi, \psi \Rightarrow \Delta}{\Phi, \varphi \vee \psi \Rightarrow \Delta}$$

$$(\Rightarrow \vee) \frac{\Phi \Rightarrow \Delta, \varphi, \psi}{\Phi \Rightarrow \Delta, \varphi \vee \psi}$$

$$(\rightarrow \Rightarrow) \frac{\Phi \Rightarrow \Delta, \varphi \quad \Phi, \psi \Rightarrow \Delta}{\Phi, \varphi \rightarrow \psi \Rightarrow \Delta}$$

$$(\Rightarrow \rightarrow) \frac{\Phi, \varphi \Rightarrow \Delta, \psi}{\Phi \Rightarrow \Delta, \varphi \rightarrow \psi}$$

Vollständigkeit des Sequenzenkalküls

Definition. Sei $\Phi \subseteq AL$ eine Menge von Formeln.

Eine Formel $\psi \in AL$ kann aus Φ hergeleitet werden, geschrieben
 $\Phi \vdash_S \psi$, wenn es eine endliche Teilmenge $\Phi' \subseteq \Phi$ gibt, so dass
 $\Phi' \Rightarrow \psi$ im Sequenzenkalkül beweisbar ist.

Insbesondere kann ψ aus der leeren Menge hergeleitet werden, d.h.
 $\vdash_S \psi$, wenn $\emptyset \Rightarrow \psi$ im Sequenzenkalkül beweisbar ist.

Definition. Sei $\Phi \subseteq AL$ eine endliche oder unendliche Formelmenge.

Φ ist **inkonsistent**, wenn es eine Formel $\psi \in AL$ gibt, so dass
 $\Phi \vdash_S \psi$ und $\Phi \vdash_S \neg\psi$.

Andernfalls ist Φ **konsistent**.

Vollständigkeit des Sequenzenkalküls

Das vorherige Theorem und der Algorithmus zeigen die Vollständigkeit des Sequenzenkalküls für endliche Formelmengen.

Mit Hilfe des Kompaktheitssatzes folgt die Vollständigkeit für beliebige Formelmengen.

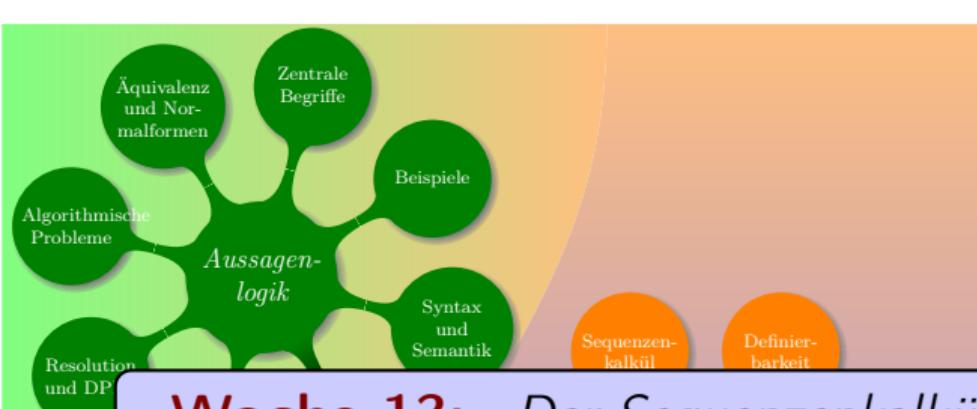
Theorem. (Vollständigkeit und Korrektheit)

Sei $\Phi \subseteq AL$ eine Menge von Formeln und sei $\psi \in AL$.

1. Φ ist konsistent genau dann, wenn Φ erfüllbar ist.
2. $\Phi \vdash_S \psi$ genau dann, wenn $\Phi \models \psi$.

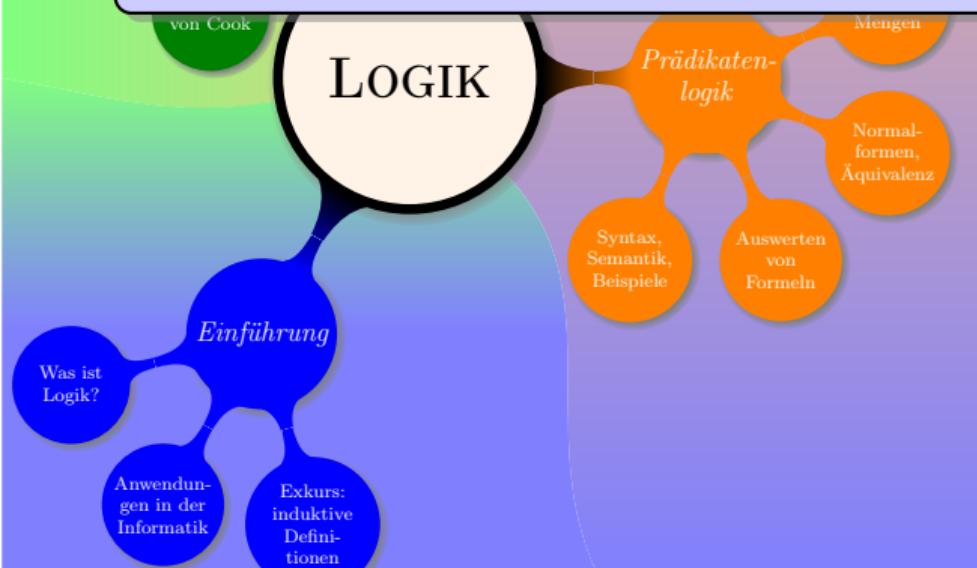
Ableitbarkeit vs. Semantische Folgerung

Sequenzenkalkül	Semantische Konzepte
Sequenz $\Phi \Rightarrow \psi$ ist gültig \vdash_S	Aus Φ folgt logisch ψ \models
Φ ist konsistent	Φ ist erfüllbar
$\emptyset \Rightarrow \psi$ gültig	ψ allgemeingültig
$\psi \Rightarrow \emptyset$ gültig	ψ unerfüllbar



Nov.	17	18	19	20	21	22	23	<i>Einführung</i>
	24	25	26	27	28	29	30	<i>Aussagenlogik</i>
	31	1	2	3	4	5	6	<i>Normalformen</i>
	7	8	9	10	11	12	13	<i>Resolution</i>
Dez.	14	15	16	17	18	19	20	<i>Kompaktheit</i>
	21	22	23	24	25	26	27	<i>DPLL, Satz von Cook</i>

Woche 13: Der Sequenzenkalkül der Prädikatenlogik



Jan.	5	6	7	8	9	10	11	<i>Prädikatenlogik</i>
	12	13	14	15	16	17	18	<i>Komplexität von FO</i>
	19	20	21	22	23	24	25	<i>Weihnachten</i>
	26	27	28	29	30	31	1	<i>Neujahr</i>
	2	3	4	5	6	7	8	<i>Normalformen</i>
	9	10	11	12	13	14	15	<i>Definierbarkeit</i>
	16	17	18	19	20	21	22	<i>EF-Spiele</i>
Feb.	23	24	25	26	27	28	29	<i>EF-Spiele</i>
	30	31	1	2	3	4	5	<i>Sequenzenkalkül AL</i>
	6	7	8	9	10	11	12	<i>Sequenzenkalkül FO</i>
	13	14	15	16	17	18	19	<i>Ausblick</i>

13.1 Der Sequenzenkalkül der Prädikatenlogik

Erweiterung des Sequenzenkalküls auf FO

Wir erweitern den Sequenzenkalkül für die Prädikatenlogik.

Dazu können wir alle bisherigen Regeln und Definitionen direkt übernehmen.

Wir brauchen nur noch Regeln für die Quantoren.

Dies erfordert allerdings etwas Vorarbeit.

Wir haben gesehen, dass bei Substitutionen die Kombination von freien und gebundenen Variablen Probleme bereiten kann.

Um ähnliche Probleme im Sequenzenkalkül zu vermeiden, werden wir freie Variablen durch Konstantensymbole substituieren und somit nur mit Sätzen arbeiten.

Elimination von Variablen

Lemma. Sei σ eine Signatur.

Sei $\varphi(x_1, \dots, x_k)$ eine Formel mit $frei(\varphi) \subseteq \{x_1, \dots, x_k\}$.

Seien c_1, \dots, c_k Konstantensymbole mit $c_i \notin \sigma$ für alle $1 \leq i \leq k$.

φ ist erfüllbar gdw. $\varphi[x_1/c_1, \dots, x_k/c_k]$ erfüllbar ist.

Beweis. Wenn φ erfüllbar ist, dann gibt es

- eine σ -Struktur \mathcal{A} und
- eine Belegung β mit $\{x_1, \dots, x_k\} \subseteq def(\beta)$,

so dass $(\mathcal{A}, \beta) \models \varphi$.

Aber dann erfüllt die $\sigma \cup \{c_1, \dots, c_k\}$ -Struktur \mathcal{B} mit $\mathcal{B}|_\sigma = \mathcal{A}$ und $c_i^\mathcal{B} := \beta(x_i)$ die Formel $\varphi[x_1/c_1, \dots, x_k/c_k]$.

Die Umkehrung ist analog.

Elimination von Variablen

Das Lemma zeigt, dass Erfüllbarkeit, Gültigkeit, ... von Formeln auf entsprechende Aussagen über Sätzen reduziert werden können.

Notation. In der Formulierung des Sequenzenkalküls werden wir ausschließlich mit Sätzen arbeiten.

Das heißt, wann immer wir $\Phi, \Delta, \varphi, \psi, \dots$ schreiben, meinen wir Sätze bzw. Mengen von Sätzen.

Ausname. Wenn wir $\varphi(x)$ oder $\psi(x)$ schreiben, dann soll das bedeuten, dass die Formeln eine freie Variable x haben.

Hinweis. Für den Rest des Abschnitts fixieren wir eine Signatur σ und eine abzählbar unendliche Menge c_0, c_1, \dots von Konstantensymbolen $c_i \notin \sigma$, für alle $i \geq 0$.

Das heißt, formal arbeiten wir in der Signatur $\tau := \sigma \cup \{c_0, c_1, \dots\}$.

Sequenzen

Definition.

1. Eine **Sequenz** ist eine Aussage der Form

$$\Phi \Rightarrow \Delta$$

für Multimengen $\Phi, \Delta \subseteq \text{FO}$.

Wir nennen Φ die **Voraussetzungen** und Δ die **Konklusionen**.

2. Eine Sequenz $\Phi \Rightarrow \Delta$ ist **gültig**, wenn jede σ -Interpretation \mathcal{I} , die alle Formeln in Φ erfüllt, mindestens eine Formel in Δ erfüllt.
3. Ist $\Phi \Rightarrow \Delta$ nicht gültig, dann gibt es eine σ -Interpretation \mathcal{I} , die alle $\varphi \in \Phi$ erfüllt, aber kein $\delta \in \Delta$.

Wir sagen: \mathcal{I} falsifiziert die Sequenz.

Sequenzen.

$$\{\forall x R(x)\} \Rightarrow \{\exists y Q(y)\}$$

$$\{\forall x \forall y (x = y \vee E(x, y)), \neg E(s, t)\} \Rightarrow \{s = t\}$$

Gültige Sequenzen.

$$\{\forall x \forall y (x = y \vee E(x, y)), \neg E(s, t)\} \\ \Rightarrow \{s = t\}$$

$$\{\exists x R(x), \forall y Q(y)\} \Rightarrow \{\exists x R(x)\}$$

$$\{\exists x R(x), \forall y Q(y)\} \Rightarrow \{\exists y R(y)\}$$

Nicht gültige Sequenzen.

$$\{\forall x R(x)\} \Rightarrow \{\exists y Q(y)\}$$

Axiome und Regeln

Definition. Ein **Axiom** des Sequenzenkalküls ist eine Sequenz $\Phi \Rightarrow \Delta$, so dass $\Phi \cap \Delta \neq \emptyset$.

Beispiel für ein Axiom.
 $\{\exists xR(x), \forall yQ(y)\} \Rightarrow \{\exists xR(x)\}$

Die Regeln des Sequenzenkalküls. Die Regeln des Sequenzenkalküls der Prädikatenlogik bestehen aus

- den Regeln des aussagenlogischen Kalküls erweitert um
- die **Gleichheitsregel**,
- die **Substitutionsregel** und
- die **Quantorenregeln**.

Die Regeln für Gleichheit und Substitution

Die Regel für Gleichheit.

$$(==\Rightarrow) \frac{\Phi, t = t \Rightarrow \Delta}{\Phi \Rightarrow \Delta}$$

Die Regeln der Substitution.

$$(S \Rightarrow) \frac{\Phi, \psi(t) \Rightarrow \Delta}{\Phi, t \doteq t', \psi(t') \Rightarrow \Delta}$$

$$(\Rightarrow S) \frac{\Phi, \Rightarrow \Delta, \psi(t)}{\Phi, t \doteq t' \Rightarrow \Delta, \psi(t')}$$

$t \doteq t'$: steht für $t = t'$ oder $t' = t$.

t, t' : Terme ohne freie Variablen

$\psi(x)$: Formel mit $frei(\psi) = \{x\}$.

Beispiel. Sei $\sigma := \{R, f, c\}$, wobei R ein Relationssymbol, f ein Funktionssymbol und c ein Konstantensymbol ist.

$$(\Rightarrow S) \frac{\overline{Rfc \Rightarrow Rfc}}{Rfc, fc = c \Rightarrow Rffc}$$

(setze $\psi(x) := Rfx$ und wende $(\Rightarrow S)$ an)

Die Quantorenregeln

Die Regeln für den Existenzquantor.

$$(\exists \Rightarrow) \frac{\Phi, \psi(c) \Rightarrow \Delta}{\Phi, \exists x \psi(x) \Rightarrow \Delta} \quad \text{wenn } c \text{ nicht in } \Phi, \Delta \text{ und } \psi(x) \text{ vorkommt}$$

$$(\Rightarrow \exists) \frac{\Phi \Rightarrow \Delta, \psi(t)}{\Phi \Rightarrow \Delta, \exists x \psi(x)} \quad t \text{ Term}$$

Beispiel. Das folgende ist ein Beweis für $\exists x \exists y Rxy \Rightarrow \exists y \exists x Rxy$

$$\begin{array}{c} (\Rightarrow \exists) \frac{\overline{R(c, d) \Rightarrow R(c, d)}}{R(c, d) \Rightarrow \exists x R(x, d)} \\ (\Rightarrow \exists) \frac{\overline{R(c, d) \Rightarrow \exists y \exists x R(x, y)}}{R(c, d) \Rightarrow \exists y \exists x R(x, y)} \\ (\exists \Rightarrow) \frac{\overline{\exists y R(c, y) \Rightarrow \exists y \exists x R(x, y)}}{\exists y R(c, y) \Rightarrow \exists y \exists x R(x, y)} \\ (\exists \Rightarrow) \frac{\overline{\exists x \exists y R(x, y) \Rightarrow \exists y \exists x R(x, y)}}{\exists x \exists y R(x, y) \Rightarrow \exists y \exists x R(x, y)} \end{array}$$

Die Quantorenregeln

Die Regeln für den Allquantor.

$$(\forall \Rightarrow) \frac{\Phi, \psi(t) \Rightarrow \Delta}{\Phi, \forall x \psi(x) \Rightarrow \Delta}$$

$$(\Rightarrow \forall) \frac{\Phi \Rightarrow \Delta, \psi(c)}{\Phi \Rightarrow \Delta, \forall x \psi(x)}$$

wenn c nicht in Φ, Δ und $\psi(x)$ vorkommt

Beispiel. Das folgende ist ein Beweis für $\exists y \forall x Rxy \Rightarrow \forall y \exists x Rxy$

$$\begin{array}{c}
 (\Rightarrow \exists) \frac{}{R(c, d) \Rightarrow R(c, d)} \\
 (\forall \Rightarrow) \frac{R(c, d) \Rightarrow \exists x R(x, d)}{\forall y R(c, y) \Rightarrow \exists x R(x, d)} \\
 (\Rightarrow \forall) \frac{\forall y R(c, y) \Rightarrow \exists x R(x, d)}{\forall y R(c, y) \Rightarrow \forall y \exists x R(x, y)} \\
 (\exists \Rightarrow) \frac{\forall y R(c, y) \Rightarrow \forall y \exists x R(x, y)}{\exists x \forall y R(x, y) \Rightarrow \forall y \exists x R(x, y)}
 \end{array}$$

Die Regeln des prädikatenlogischen Sequenzenkalküls

$$(\neg \Rightarrow) \frac{\Phi \Rightarrow \Delta, \psi}{\Phi, \neg \psi \Rightarrow \Delta}$$

$$(\wedge \Rightarrow) \frac{\Phi, \psi, \varphi \Rightarrow \Delta}{\Phi, \psi \wedge \varphi \Rightarrow \Delta}$$

$$(\vee \Rightarrow) \frac{\Phi, \varphi \Rightarrow \Delta \quad \Phi, \psi \Rightarrow \Delta}{\Phi, \varphi \vee \psi \Rightarrow \Delta}$$

$$(\rightarrow \Rightarrow) \frac{\Phi \Rightarrow \Delta, \varphi \quad \Phi, \psi \Rightarrow \Delta}{\Phi, \varphi \rightarrow \psi \Rightarrow \Delta}$$

$$(\forall \Rightarrow) \frac{\Phi, \psi(t) \Rightarrow \Delta}{\Phi, \forall x \psi(x) \Rightarrow \Delta}$$

$$(\exists \Rightarrow) \frac{\Phi, \psi(c) \Rightarrow \Delta}{\Phi, \exists x \psi(x) \Rightarrow \Delta} \text{ c n.i. } \Phi, \Delta, \psi$$

$$(S \Rightarrow) \frac{\Phi, \psi(t) \Rightarrow \Delta}{\Phi, t \doteq t', \psi(t') \Rightarrow \Delta}$$

$$(\Rightarrow \neg) \frac{\Phi, \psi \Rightarrow \Delta}{\Phi \Rightarrow \Delta, \neg \psi}$$

$$(\Rightarrow \wedge) \frac{\Phi \Rightarrow \Delta, \psi \quad \Phi \Rightarrow \Delta, \varphi}{\Phi \Rightarrow \Delta, \psi \wedge \varphi}$$

$$(\Rightarrow \vee) \frac{\Phi \Rightarrow \Delta, \varphi, \psi}{\Phi \Rightarrow \Delta, \varphi \vee \psi}$$

$$(\Rightarrow \rightarrow) \frac{\Phi, \varphi \Rightarrow \Delta, \psi}{\Phi \Rightarrow \Delta, \varphi \rightarrow \psi}$$

$$(\Rightarrow \forall) \frac{\Phi \Rightarrow \Delta, \psi(c)}{\Phi \Rightarrow \Delta, \forall x \psi(x)} \text{ c n. i. } \Phi, \Delta, \psi$$

$$(\Rightarrow \exists) \frac{\Phi \Rightarrow \Delta, \psi(t)}{\Phi \Rightarrow \Delta, \exists x \psi(x)}$$

$$(\Rightarrow S) \frac{\Phi, \Rightarrow \Delta, \psi(t)}{\Phi, t \doteq t' \Rightarrow \Delta, \psi(t')}$$

$$(\Rightarrow =) \frac{\Phi, t = t \Rightarrow \Delta}{\Phi \Rightarrow \Delta}$$