Logik für Informatiker

Vorlesung 1 und 2: Einführung in die Logik

Babeş-Bolyai Universität, Department für Informatik, Cluj-Napoca
csacarea@cs.ubbcluj.ro

Organisatorisches

Vorbemerkungen

- Die Vorlesung wird (teilweise) folienbasiert gehalten;
- Die Folien enthalten nur die wichtigsten Aspekte (Definitionen, Sätze, knappe Beispiele, wichtige Bemerkungen);
- Alles was sonst eine Vorlesung ausmacht (Erläuterungen, ausführliche Beispiele, Beweise von Sätzen , Anwendungen, Querverweise auf andere Gebiete der Informatik, etc.) gibt es nur in der Vorlesung selbst.

Sprechstunden

Nach Vereinbarung

Übungen

- *Autonomes Lösen* der Aufgaben
- ohne vorherige Wiederholung seitens des Dozenten
- ohne Musterlösung an der Tafel (kommt dann später als extra Datei).

Notwendig dafür ist, dass

- man während der Vorlesung mitschreibt
- man sich für die nächste Übungsstunde zu Hause vorbereitet
- Mitschrift und Folien mitgebracht werden
- Initiative ergriffen wird.

Vorteile

- aktives Erlernen der Inhalte ist effektiver
- Zusammenhänge aus dem Vorlesungsstoff werden selbst entdeckt und ausgearbeitet
- Man erlernt strukturiert und unabhängig zu denken
- *team work* wird erlernt
- Erklären wird geübt und erlernt
- Trainieren für die Prüfung ;-)
- You have finished your study of . . . Your personal strengths include pro-activity and team work, you are communicative and willing to cooperate. (typical job advertisement)

Organisatorisches

- Sprechstunden! (Consultaţii): Ihr stellt (logische) Fragen, ich beantworte sie. Wann? Nach Vereinbarung!
- Prüfungen: Ich stelle Fragen, ihr beantwortet sie :)
- Einmal im Monat: TEST IN LOGIK: Jeweils 30 Minuten, während der Vorlesung.
- Die Endnote besteht aus: Note in der schriftlichen Klausur + maximal 1,5 Punkte aus den drei Testen + verschiedene Bonuspunkte die während des Semesters vergeben werden

Organisatorisches

- Alle Lehrinhalte werden hier hochgeladen.
- Deine Deutschkenntnisse sind sind zwischen ähm und hä?. Dann kannst du kostenlos die Deutschkurse, die vom IDLF - Institut für Deutsche Lehre und Forschung angeboten werden besuchen.
- oder du belegst den Kurs Kommunikation in deutscher Sprache in IT.
- Sprachkurse für Informatiker werden extra angeboten. Bitte anmelden!
- Gilt auch für alle anderen!

Organisatorisches: Q & A

- Alle Prof's bieten Sprechstunden an
- Wie kontaktiere ich die Professoren?

Organisatorisches: Q & A

- Alle Prof's bieten Sprechstunden an
- Wie kontaktiere ich die Professoren? E-mail ist am einfachsten

Organisatorisches: Q & A

- Alle Prof's bieten Sprechstunden an
- Wie kontaktiere ich die Professoren? E-mail ist am einfachsten
- Wo finde ich die E-mail Addressen?

Organisatorisches: Q & A

- Alle Prof's bieten Sprechstunden an
- Wie kontaktiere ich die Professoren? E-mail ist am einfachsten
- Wo finde ich die E-mail Addressen?
- Wühle bitte auf der Homepage der Fakultät. Dort findest du viele interessante Informationen und sollte dein virtuelles zu Hause sein für die nächsten 3 Jahren...

Logik für Informatiker

- Mathematisch?

Logik für Informatiker

- Mathematisch?
- Ja
- Unverständlich?

Logik für Informatiker

- Mathematisch?
- Ja
- Unverständlich?
- Mal schauen...

Logik für Informatiker

- Mathematisch?
- Ja
- Unverständlich?
- Mal schauen...
- Reine Theorie ohne praktischen Nutzen?

Logik für Informatiker

- Mathematisch?
- Ja
- Unverständlich?
- Mal schauen...
- Reine Theorie ohne praktischen Nutzen?
- nein: Verifikation von Hardware, Software, Protokollen Sprachverarbeitung und Wissensrepräsentation, Abfragensprachen für Datenbanken;

Hilfe! Ich bin ein Informatiker...

- Rechnerarchitektur (logische Schaltungen);
- Software Engineering (Testen und Verifikation);
- Programmiersprachen (Semantik, logische Programmierung) ;
- Datenbanken (relationale Algebra);
- Künstliche Intelligenz (automatisches Beweisen);
- Komplexität der Algorithmen.

J. Y. Halpern, R. Harper, N. Immerman, P.G.Kolaitis, M.Y. Vardi, V.Vianu, On the Unusual Effectiveness of Logic in Computer Science Bulletin of Symbolic Logic 7(2), 2001, 213-233

Logik = Train your brain!

Logik = Train your brain!

Logik = Train your brain!

Formale Logik

Ziel

- Formalisierung und Automatisierung rationalen Denkens,
- Rational richtige Ableitung von neuem Wissen aus gegebenem Wissen.

Rolle der Logik in der Informatik

- Anwendung innerhalb der Informatik: Spezifikation, Programmentwicklung, Programmverifikation
- Werkzeug für Anwendungen außerhalb der Informatik: Künstliche Intelligenz, Wissensrepräsentation

Was ist Logik?

Die Logik ist zu verstehen als die *Wissenschaft des formal richtigen Schließens* und fungiert als vorgängige Wissenschaft für alle anderen Wissenschaften (insbesondere der Mathematik).

Wortabstammung

logike techne = die Wissenschaft des Denkens; logos = Wort, Schlußfolgerung

Aristotel, Prior Analytics:

A deduction is speech (logos) in which, certain things having been supposed, something different from those supposed results of necessity because of their being so. Smith, R., Aristotles Logic, SEP (Spring 2014), Ed. N. Zalta (ed.)

Was sagt man so über Logik?

Contrariwise [...] if it was so, it might be; and if it were so, it would be; but as it isnt, it aint. Thats logic.
Lewis Carroll, Through the Looking-Glass, and What Alice Found There

Logic is the beginning of wisdom, [...], not the end.
Captain Spock, Star Trek VI: The Undiscovered Country (1991)

Was ist nun Logik?

Logik

Ist eine formale Sprache, die aus Syntax und Semantik besteht. Jede Logik hat auch ein Deduktionsmechanismus, ein sogenannter Kalkül des Schließens, welches logische Schlußfolgerungen ermöglicht.

Die klassische Logik

Syntax: $p \to q$. Deduktion: $\frac{p \to q, p}{q}$ (modus ponens)

Semantik:

p	q	$\to$
0	0	1
0	1	1
1	0	0
1	1	1

Gibt es mehrere Logiken?

Ja!

- Klassische Logik;
- modale Logik;
- Temporale, dynamische und epistemische Logik;
- Intuitionistische Logik;
- Lineare Logik;
- Mehrwertige Logik;
- Fuzzy Logik;
- Logiken für das Überprüfen der Korrektheit von Programmen;
- Hoare Logik, dynamische Logik;
- Separation Logic, Matching Logic
- Beschreibungslogiken (Semantic Web)
- etc.

Jetzt geht es los!

Formale Logik

Syntax

welche Formeln?

Semantik

Modelle (Strukturen) Wann ist eine Formel wahr (in einer Struktur)?

Deduktionsmechanismus/Kalkül

Ableitung neuer wahrer Formeln.

Aussagenlogik

Die Welt besteht aus Fakten die wahr oder falsch sein können.

Syntax der Aussagenlogik: Logische Zeichen

$$\wedge, \vee, \neg, \rightarrow, \leftrightarrow, \bot, \top$$

Vokabular der Aussagenlogik

- Abzählbare Menge von Symbolen:

$$\Pi = \{P_0, \ldots, P_n\}$$

oder

$$\Pi = \{P_0, P_1, \ldots\}.$$

Bezeichnungen für Symbole in Π:

  - atomare Aussagen
  - Atome
  - Aussagenvariablen

Formeln der Aussagenlogik

Induktive Definition!

Aussagenformeln

$\bot$	Falsum
$\top$	Verum
$P, P \in \Pi$	atomare Formel
$\neg F$	Negation
$(F \wedge G)$	Konjunktion
$(F \vee G)$	Disjunktion
$(F \rightarrow G)$	Implikation
$(F \leftrightarrow G)$	Äquivalenz

Konvention zur Notation

Beispiele:
Sei $\Pi = \{P, Q, R\}$
$\bot, P, \neg Q, P \wedge \neg Q, (P \vee (\neg R \wedge T))$ sind Formeln
Wir schreiben $P \wedge Q \wedge R$ statt $(P \wedge Q) \wedge R$.

Das 8-Damen Problem

Man platziere acht Damen so auf einem Schachbrett, dass sie sich gegenseitig nicht bedrohen.

Das 8-Damen Problem

Beschreibung des Problems

Für jedes Feld des Schachbretts eine aussagenlogische Variable D_{ij} die den Wert wahr hat, wann immer auf dem Feld (i, j) eine Dame steht.

Das 8-Damen Problem

Beispiel

Auf dem Feld (5, 7) steht eine Dame $\mapsto D_{57}$ wahr.

Einschränkungen pro Feld F_{ij}

Falls auf dem Feld (5, 7) eine Dame steht:

- keine andere Dame auf Feld (5,1), (5,2), (5,3), (5,4),(5,5), (5,6), (5,8);
- keine andere Dame auf Feld (1,7), (2,7), (3,7), (4,7),(6,7), (7,7), (8,7);
- keine andere Dame auf Feld (6,8), (4,6), (3,5), (2,4), (1,3);
- keine andere Dame auf Feld (4,8), (6,6), (7,5), (8,4);
- (ähnliche Bedingungen für alle Felder (i, j)).

Das 8-Damen Problem

Einschränkungen pro Feld F_{ij}

- keine andere Dame auf Feld (5,1), (5,2), (5,3), (5,4),(5,5), (5,6), (5,8);

Das 8-Damen Problem

Einschränkungen pro Feld F_{ij}

- keine andere Dame auf Feld (5,1), (5,2), (5,3), (5,4),(5,5), (5,6), (5,8);
- $D_{5,7} \rightarrow \neg D_{5,8} \wedge \neg D_{5,6} \wedge \neg D_{5,5} \wedge \neg D_{5,4} \wedge \neg D_{5,3} \wedge \neg D_{5,2} \wedge \neg D_{5,1}$;

Das 8-Damen Problem

Einschränkungen pro Feld F_{ij}

- keine andere Dame auf Feld (5,1), (5,2), (5,3), (5,4),(5,5), (5,6), (5,8);
- $D_{5,7} \rightarrow \neg D_{5,8} \wedge \neg D_{5,6} \wedge \neg D_{5,5} \wedge \neg D_{5,4} \wedge \neg D_{5,3} \wedge \neg D_{5,2} \wedge \neg D_{5,1}$;
- keine andere Dame auf Feld (1,7), (2,7), (3,7), (4,7),(6,7), (7,7), (8,7);

Das 8-Damen Problem

Einschränkungen pro Feld F_{ij}

- keine andere Dame auf Feld (5,1), (5,2), (5,3), (5,4),(5,5), (5,6), (5,8);
- $D_{5,7} \rightarrow \neg D_{5,8} \wedge \neg D_{5,6} \wedge \neg D_{5,5} \wedge \neg D_{5,4} \wedge \neg D_{5,3} \wedge \neg D_{5,2} \wedge \neg D_{5,1}$;
- keine andere Dame auf Feld (1,7), (2,7), (3,7), (4,7),(6,7), (7,7), (8,7);
- $D_{5,7} \rightarrow \neg D_{1,7} \wedge \neg D_{2,7} \wedge \neg D_{3,7} \wedge \neg D_{4,7} \wedge \neg D_{6,7} \wedge \neg D_{7,7} \wedge \neg D_{8,7}$;

Das 8-Damen Problem

Einschränkungen pro Feld F_{ij}

- keine andere Dame auf Feld (5,1), (5,2), (5,3), (5,4),(5,5), (5,6), (5,8);
- $D_{5,7} \rightarrow \neg D_{5,8} \wedge \neg D_{5,6} \wedge \neg D_{5,5} \wedge \neg D_{5,4} \wedge \neg D_{5,3} \wedge \neg D_{5,2} \wedge \neg D_{5,1}$;
- keine andere Dame auf Feld (1,7), (2,7), (3,7), (4,7),(6,7), (7,7), (8,7);
- $D_{5,7} \rightarrow \neg D_{1,7} \wedge \neg D_{2,7} \wedge \neg D_{3,7} \wedge \neg D_{4,7} \wedge \neg D_{6,7} \wedge \neg D_{7,7} \wedge \neg D_{8,7}$;
- keine andere Dame auf Feld (6,8), (4,6), (3,5), (2,4), (1,3);
- $D_{5,7} \rightarrow \neg D_{6,8} \wedge \neg D_{4,6} \wedge \neg D_{3,5} \wedge \neg D_{2,4} \wedge \neg D_{1,3}$
- keine andere Dame auf Feld (4,8), (6,6), (7,5), (8,4);
- $D_{5,7} \rightarrow \neg D_{4,8} \wedge \neg D_{6,6} \wedge \neg D_{7,5} \wedge \neg D_{8,4}$;
- (ähnliche Bedingungen für alle Felder (i, j)).

Das 8-Damen Problem

$F_{5,7}$:

$$D_{5,7} \rightarrow \neg D_{5,8} \wedge \neg D_{5,6} \wedge \neg D_{5,5} \wedge \neg D_{5,4} \wedge \neg D_{5,3} \wedge \neg D_{5,2} \wedge \neg D_{5,1}$$
$$D_{5,7} \rightarrow \neg D_{1,7} \wedge \neg D_{2,7} \wedge \neg D_{3,7} \wedge \neg D_{4,7} \wedge \neg D_{6,7} \wedge \neg D_{7,7} \wedge \neg D_{8,7}$$
$$D_{5,7} \rightarrow \neg D_{6,8} \wedge \neg D_{4,6} \wedge \neg D_{3,5} \wedge \neg D_{2,4} \wedge \neg D_{1,3}$$
$$D_{5,7} \rightarrow \neg D_{4,8} \wedge \neg D_{6,6} \wedge \neg D_{7,5} \wedge \neg D_{8,4}$$

Das 8-Damen Problem

Globale Einschränkungen

Das 8-Damen Problem

Globale Einschränkungen

Für jedes k mit $1 \leq k \leq 8$:

$$R_k := D_{1,k} \vee D_{2,k} \vee D_{3,k} \vee D_{4,k} \vee D_{5,k} \vee D_{6,k} \vee D_{7,k} \vee D_{8,k}.$$

Das 8-Damen Problem

Struktur: Wahrheitswerte für die atomaren Aussagen $D_{i,j}$
Modell für $F_{i,j}(R_k)$: Wahrheitswerte fur die atomaren Aussagen $D_{i,j}$ so dass $F_{i,j}$ wahr (bzw. R_k wahr).
Lösung des 8-Damen Problems: Eine aussagenlogische Struktur beschreibt eine Lösung des 8-DamenProblems genau dann, wenn sie ein Modell der Formeln

- $F_{i,j}$ für alle $1 \leq i, j \leq 8$;
- R_k für alle $1 \leq k \leq 8$ ist.

Syntax: Beispiel

Q: *Worin besteht das Geheimnis Ihres langen Lebens?*, wurde ein 100 Jähriger gefragt.

A: *Ich halte mich streng an die Diätregeln*:

- Wenn ich kein Bier zu einer Mahlzeit trinke, dann habe ich immer Fisch.
- Immer wenn ich Fisch und Bier zur selben Mahlzeit habe, verzichte ich auf Eis.
- Wenn ich Eis habe oder Bier meide, dann rühre ich Fisch nicht an.

Syntax: Beispiel

Wenn ich kein Bier zu einer Mahlzeit trinke, dann habe ich immer Fisch.

Syntax: Beispiel

Wenn ich kein Bier zu einer Mahlzeit trinke, dann habe ich immer Fisch.

$\neg B \rightarrow F$

Syntax: Beispiel

Wenn ich kein Bier zu einer Mahlzeit trinke, dann habe ich immer Fisch.

$\neg B \rightarrow F$

Immer wenn ich Fisch und Bier zur selben Mahlzeit habe, verzichte ich auf Eis.

Syntax: Beispiel

Wenn ich kein Bier zu einer Mahlzeit trinke, dann habe ich immer Fisch.

$\neg B \rightarrow F$

Immer wenn ich Fisch und Bier zur selben Mahlzeit habe, verzichte ich auf Eis.

$F \wedge B \rightarrow \neg E$

Syntax: Beispiel

Wenn ich kein Bier zu einer Mahlzeit trinke, dann habe ich immer Fisch.

$\neg B \rightarrow F$

Immer wenn ich Fisch und Bier zur selben Mahlzeit habe, verzichte ich auf Eis.

$F \wedge B \rightarrow \neg E$

Wenn ich Eis habe oder Bier meide, dann ruhre ich Fisch nicht an.

Syntax: Beispiel

Wenn ich kein Bier zu einer Mahlzeit trinke, dann habe ich immer Fisch.

$\neg B \rightarrow F$

Immer wenn ich Fisch und Bier zur selben Mahlzeit habe, verzichte ich auf Eis.

$F \wedge B \rightarrow \neg E$

Wenn ich Eis habe oder Bier meide, dann ruhre ich Fisch nicht an.

$E \vee \neg B \rightarrow \neg F$

Von Syntax zu Semantik

$\neg B \rightarrow F$
$F \wedge B \rightarrow \neg E$
$E \vee \neg B \rightarrow \neg F$
Wir möchten wissen, welche Menus solche Diätregeln erfullen.

Beispiel:

- kein Bier, Fisch und Eis: erfüllt 3. Diätregel nicht!
- Bier, Fisch, kein Eis: erfüllt alle Diätregeln.

Von Syntax zu Semantik

$\neg B \to F$
$F \wedge B \to \neg E$
$E \vee \neg B \to \neg F$

Wir möchten wissen, welche Menus solche Diätregeln erfüllen.

Regeln	Formalisierung
kein Bier, Fisch und Eis:	
erfüllt 3. Diätregel nicht!	$B \mapsto$ falsch, $F \mapsto$ wahr, $E \mapsto$ wahr
Bier, Fisch, kein Eis:	
erfüllt alle Diätregeln	$B \mapsto$ wahr, $F \mapsto$ wahr, $E \mapsto$ falsch

Beispiel

- $\neg B \to F$
- $F \wedge B \to E$
- $E \vee \neg B \to \neg F$

Wir möchten wissen, welche Menüs solche Diätregeln erfüllen.

Beispiel	Formalisierung
0: falsch, 1: wahr	$\mathcal{A}\colon \{B, F, E\} \to \{0, 1\}$
Kein Bier, aber Fisch und Eis	$\mathcal{A}(B) = 0, \mathcal{A}(F) = 1, \mathcal{A}(E) = 1$
erfüllt 3. Diätregel nicht!	
Bier, Fisch, kein Eis	$\mathcal{A}(B) = 1, \mathcal{A}(F) = 1, \mathcal{A}(E) = 0$
erfüllt alle Diätregeln	

Semantik der Aussagenlogik

Aussagenvariablen haben für sich keine Bedeutung.

Semantik der Aussagenlogik

Aussagenvariablen haben für sich keine Bedeutung.

Um eine Bedeutung der Variablen zuzuordnen, brauchen wir WertebelegungenValuationen

- 1 Symbol für den Wahrheitswert wahr
- 0 Symbol für den Wahrheitswert falsch

Eine Wertebelegung (Valuation, Interpretation, Struktur, Modell) ist eine Abbildung

$$\mathcal{A}\colon \Pi \to \{0, 1\}.$$

Beispiel:

A	B	C
0	1	0

3 Variablen

Wieviele Modelle existieren für drei Variablen?

Semantik der Aussagenlogik

Auswertung von Formeln

Sei $\mathcal{A}\colon \Pi \to \{0,1\}$ eine Wertebelegung.

$\mathcal{A}^*\colon \mathit{For}_\Pi \to \{0,1\}$ wird wie folgt definiert:

- $\mathcal{A}^*(\bot) = 0$
- $\mathcal{A}^*(\top) = 1$
- $\mathcal{A}^*(P) = \mathcal{A}(P)$, falls $P \in \Pi$

Semantik der Aussagenlogik

Auswertung von Formeln

Sei $\mathcal{A}\colon \Pi \to \{0,1\}$ eine Wertebelegung.

$\mathcal{A}^*\colon For_\Pi \to \{0,1\}$ wird wie folgt definiert:

$$\mathcal{A}^*(\neg F) = \begin{cases} 0 & \text{falls } \mathcal{A}^*(F) = 1 \\ 1 & \text{falls } \mathcal{A}^*(F) = 0. \end{cases}$$

Semantik der Aussagenlogik

Auswertung von Formeln

Sei $\mathcal{A}\colon \Pi \to \{0,1\}$ eine Wertebelegung.

$\mathcal{A}^*\colon For_\Pi \to \{0,1\}$ wird wie folgt definiert:

$$\mathcal{A}^*(F_1 \wedge F_2) = \begin{cases} 0 & \text{falls } \mathcal{A}^*(F_1) = 0 \text{ oder } \mathcal{A}^*(F_2) = 0 \\ 1 & \text{falls } \mathcal{A}^*(F_1) = \mathcal{A}^*(F_2) = 1. \end{cases}$$

Semantik der Aussagenlogik

Auswertung von Formeln

Sei $\mathcal{A}\colon \Pi \to \{0,1\}$ eine Wertebelegung.

$\mathcal{A}^*\colon For_\Pi \to \{0,1\}$ wird wie folgt definiert:

$$\mathcal{A}^*(F_1 \vee F_2) = \begin{cases} 0 & \text{falls } \mathcal{A}^*(F_1) = \mathcal{A}^*(F_2) = 0 \\ 1 & \text{falls } \mathcal{A}^*(F_1) = 1 \text{ oder } \mathcal{A}^*(F_2) = 1. \end{cases}$$

Semantik der Aussagenlogik

Auswertung von Formeln

Sei $\mathcal{A}\colon \Pi \to \{0,1\}$ eine Wertebelegung.

$\mathcal{A}^*\colon For_\Pi \to \{0,1\}$ wird wie folgt definiert:

$$\mathcal{A}^*(F_1 \to F_2) = \begin{cases} 1 & \text{falls } \mathcal{A}^*(F_1) = 0 \text{ oder } \mathcal{A}^*(F_2) = 1 \\ 0 & \text{falls } \mathcal{A}^*(F_1) = 1 \text{ und } \mathcal{A}^*(F_2) = 0. \end{cases}$$

$$\mathcal{A}^*(F_1 \leftrightarrow F_2) = \begin{cases} 1 & \text{falls } \mathcal{A}^*(F_1) = \mathcal{A}^*(F_2) = 1 \\ 0 & \text{sonst} \end{cases}$$

Semantik der Aussagenlogik

Auswertung von Formeln

Sei $\mathcal{A}\colon \Pi \to \{0,1\}$ eine Wertebelegung.

$\mathcal{A}^*\colon For_\Pi \to \{0,1\}$ wird induktiv über den Aufbau von F wie folgt definiert:

$\mathcal{A}^*(\bot) = 0$
$\mathcal{A}^*(\top) = 1$
$\mathcal{A}^*(P) = \mathcal{A}(P)$
falls $P \in \Pi$ eine aussagenlogische Variable ist
$\mathcal{A}^*(\neg F) = 1 - \mathcal{A}^*(F)$
$\mathcal{A}^*(F\rho G) = B_\rho(\mathcal{A}^*(F), \mathcal{A}^*(G))$
$\rho \in \{\vee, \wedge, \to, \leftrightarrow\}$
$B_\rho(x,y)$ wird berechnet entsprechend der Wahrheitstafel für ρ
z.B. $B_\vee(0,1) = (0 \vee 1) = 1$; $B_\to(1,0) = (1 \to 0) = 0$

Wir schreiben normalerweise $\mathcal{A}$ statt $\mathcal{A}^*$.

Wahrheitstafeln für die logischen Operatoren

P	$\neg P$
0	1
1	0

$\wedge$	0	1
0	0	0
1	0	1

$\vee$	0	1
0	0	1
1	1	1

$\rightarrow$	0	1
0	1	1
1	0	1

$\leftrightarrow$	0	1
0	1	0
1	0	1

Beispiel

Sei $\mathcal{A} : \{P, Q, R\} \to \{0, 1\}$ mit $\mathcal{A}(P) = 1, \mathcal{A}(Q) = 0, \mathcal{A}(R) = 1$.

$$\mathcal{A}^*((P \wedge (Q \vee \neg P)) \to R)$$

Beispiel

Sei $\mathcal{A} : \{P, Q, R\} \to \{0, 1\}$ mit $\mathcal{A}(P) = 1, \mathcal{A}(Q) = 0, \mathcal{A}(R) = 1$.

$$\mathcal{A}^*((P \wedge (Q \vee \neg P)) \to R) \quad = \quad \mathcal{A}^*((P \wedge (Q \vee \neg P)) \to \mathcal{A}^*(R)$$

Beispiel

Sei $\mathcal{A} : \{P, Q, R\} \to \{0, 1\}$ mit $\mathcal{A}(P) = 1, \mathcal{A}(Q) = 0, \mathcal{A}(R) = 1$.

$$\begin{aligned} \mathcal{A}^*((P \wedge (Q \vee \neg P)) \to R) &= \mathcal{A}^*(P \wedge (Q \vee \neg P) \to \mathcal{A}^*(R) \\ &= (\mathcal{A}^*(P) \wedge \mathcal{A}^*(Q \vee \neg P)) \to \mathcal{A}^*(R) \end{aligned}$$

Beispiel

Sei $\mathcal{A} : \{P, Q, R\} \to \{0, 1\}$ mit $\mathcal{A}(P) = 1, \mathcal{A}(Q) = 0, \mathcal{A}(R) = 1$.

$$
\begin{aligned}
\mathcal{A}^*((P \wedge (Q \vee \neg P)) \to R) &= \mathcal{A}^*(P \wedge (Q \vee \neg P) \to \mathcal{A}^*(R) \\
&= (\mathcal{A}^*(P) \wedge \mathcal{A}^*(Q \vee \neg P)) \to \mathcal{A}^*(R) \\
&= (\mathcal{A}(P) \wedge (\mathcal{A}(Q) \vee \neg \mathcal{A}(P))) \to \mathcal{A}(R) \\
&= (1 \wedge (0 \vee \neg 1)) \to 1 \\
&= 1
\end{aligned}
$$

Beispiel

- $\neg B \rightarrow F$
- $F \wedge B \rightarrow \neg E$
- $E \vee \neg B \rightarrow \neg F$

Menü

Formalisierung:

$$\mathcal{A} : \{B, F, E\} \rightarrow \{0, 1\}$$

kein Bier, Fisch und Eiscreme

$$\mathcal{A}(B) = 0, \mathcal{A}(F) = 1, \mathcal{A}(E) = 1$$

erfüllt 3. Diätregel nicht!

1. $\mathcal{A}(\neg B \rightarrow F) = \neg\mathcal{A}(B) \rightarrow \mathcal{A}(F) = \neg 0 \rightarrow 1 = 1 \rightarrow 1 = 1$
2. $\mathcal{A}(F \wedge B \rightarrow \neg E) = \mathcal{A}(F) \wedge \mathcal{A}(B) \rightarrow \neg\mathcal{A}(E)$
   $= (1 \wedge 0) \rightarrow \neg 1 = 0 \rightarrow 0 = 1$
3. $\mathcal{A}(E \wedge \neg B \rightarrow \neg F) = (\mathcal{A}(E) \wedge \neg\mathcal{A}(B)) \rightarrow \neg\mathcal{A}(F)$
   $= (1 \wedge \neg 0) \rightarrow \neg 1 = 1 \rightarrow 0 = 0$

Wahrheitstabellen: Beispiel

$F = (A \vee C) \wedge (B \vee \neg C)$

A	B	C	$(A \vee C)$	$\neg C$	$(B \vee \neg C)$	$(A \vee C) \wedge (B \vee \neg C)$
0	0	0				
0	0	1				
0	1	0				
0	1	1				
1	0	0				
1	0	1				
1	1	0				
1	1	1				

Wahrheitstabellen: Beispiel

$F = (A \vee C) \wedge (B \vee \neg C)$

A	B	C	$(A \vee C)$	$\neg C$	$(B \vee \neg C)$	$(A \vee C) \wedge (B \vee \neg C)$
0	0	0	0	1	1	
0	0	1	1	0	0	
0	1	0	0	1	1	
0	1	1	1	0	1	
1	0	0	1	1	1	
1	0	1	1	0	0	
1	1	0	1	1	1	
1	1	1	1	0	1	

Wahrheitstabellen: Beispiel

$F = (A \vee C) \wedge (B \vee \neg C)$

A	B	C	$(A \vee C)$	$\neg C$	$(B \vee \neg C)$	$(A \vee C) \wedge (B \vee \neg C)$
0	0	0	0	1	1	0
0	0	1	1	0	0	0
0	1	0	0	1	1	0
0	1	1	1	0	1	1
1	0	0	1	1	1	1
1	0	1	1	0	0	0
1	1	0	1	1	1	1
1	1	1	1	0	1	1

Modell einer Formelmenge

Definition: Interpretation $\mathcal{A}$ ist Modell einer Formel $F \in \mathrm{For}_\Pi$, falls

$$\mathcal{A}^*(F) = 1.$$

Definition: Interpretation $\mathcal{A}$ ist Modell einer Formelmenge $M \subseteq \mathrm{For}_\Pi$, falls

$$\mathcal{A}^*(F) = 1 \text{ für alle } F \in M$$

Notation:

$\mathcal{A} \models F$

$\mathcal{A} \models M$

Beispiel

$F = (A \vee C) \wedge (B \vee \neg C)$

A	B	C	$(A \vee C)$	$\neg C$	$(B \vee \neg C)$	$(A \vee C) \wedge (B \vee \neg C)$
0	0	0	0	1	1	0
0	0	1	1	0	0	0
0	1	0	0	1	1	0
0	1	1	1	0	1	1
1	0	0	1	1	1	1
1	0	1	1	0	0	0
1	1	0	1	1	1	1
1	1	1	1	0	1	1

Sei $\mathcal{A} : \{A, B, C\} \to \{0, 1\}$ mit $\mathcal{A}(A) = 0, \mathcal{A}(B) = 1, \mathcal{A}(C) = 1$.

$\mathcal{A} \models (A \vee C) \wedge (B \vee \neg C)$

$\mathcal{A} \models \{(A \vee C), (B \vee \neg C)\}$

Gültigkeit und Erfüllbarkeit

Definition: F gilt in $\mathcal{A}$ (oder $\mathcal{A}$ ist Modell von F) gdw. $\mathcal{A}(F) = 1$.

Notation: $\mathcal{A} \models F$

Definition: F ist (allgemein-) gültig (oder eine Tautologie)
gdw.: $\mathcal{A} \models F$, für alle $A : \Pi \to \{0, 1\}$

Notation: $\models F$

Definition: F heißt erfüllbar gdw. es $\mathcal{A} : \Pi \to \{0, 1\}$ gibt, so dass $\mathcal{A} \models F$.

Sonst heißt F unerfüllbar (oder eine Kontradiktion).

Beispiel

$F = (A \vee C) \wedge (B \vee \neg C)$

A	B	C	$(A \vee C)$	$\neg C$	$(B \vee \neg C)$	$(A \vee C) \wedge (B \vee \neg C)$
0	0	0	0	1	1	0
0	0	1	1	0	0	0
0	1	0	0	1	1	0
0	1	1	1	0	1	1
1	0	0	1	1	1	1
1	0	1	1	0	0	0
1	1	0	1	1	1	1
1	1	1	1	0	1	1

F ist nicht allgemeingültig:

$\mathcal{A}_1(F) = 0$ für $\mathcal{A}_1 : \{A, B, C\} \to \{0, 1\}$ mit $\mathcal{A}(A) = \mathcal{A}(B) = \mathcal{A}(C) = 0$.

F ist erfüllbar (also ist F nicht unerfüllbar):

$\mathcal{A}_2(F) = 1$ für $\mathcal{A} : \{A, B, C\} \to \{0, 1\}$ mit $\mathcal{A}(A) = 0, \mathcal{A}(B) = 1, \mathcal{A}(C) = 1$.

Tautologien und Kontradiktionen

Tautologien: Formel, die stets wahr sind.

Beispiele: $p \vee (\neg p)$ (Prinzip vom ausgeschlossenen Dritten)
(Tertium non datur)

$p \leftrightarrow \neg\neg p$ (Prinzip der doppelten Negation)

Kontradiktionen: Formel, die stets falsch sind.

Beispiel: $p \wedge \neg p$

- Die Negation einer Tautologie ist eine Kontradiktion
- Die Negation einer Kontradiktion ist eine Tautologie

Beispiel

Die folgenden Formeln sind Tautologien:

(1) $(p \to \neg p) \to (\neg p)$

(2) $(p \wedge (p \to q)) \to q$

(3) $(p \wedge q) \to p$

$(p \wedge q) \to q$

(4) $p \to (p \vee q)$

$q \to (p \vee q)$

(5) $(p \to q) \to [(q \to r) \to (p \to r)]$

(6) $(((p \to q) \wedge (q \to r)) \wedge p) \to r$

Zusammenfassung

- Syntax (Formeln)
- Semantik

  Wertebelegungen/Valuationen/Modelle

  Auswertung von Formeln / Wahrheitstabellen

  $\mathcal{A} \models F$ g.d.w. $\mathcal{A}(F) = 1$.

  Gültigkeit und Erfüllbarkeit

  F allgemeingültig/Tautologie:

  – $\mathcal{A}(F) = 1$ für alle Wertebelegungen $\mathcal{A}$

  F erfüllbar

  – es gibt Wertebelegung $\mathcal{A}$ mit $\mathcal{A}(F) = 1$

  F unerfüllbar/Kontradiktion

  – $\mathcal{A}(F) = 0$ für alle Wertebelegungen $\mathcal{A}$

Zusammenfassung

Eine Formelmenge N ist:

- erfüllbar, wenn es eine Wertebelegung $\mathcal{A} : \Pi \to \{0, 1\}$ gibt, für die alle Formeln in N wahr sind (d.h. mit $\mathcal{A}(F) = 1$ für alle $F \in N$).
- unerfüllbar, wenn es keine Wertebelegung $\mathcal{A} : \Pi \to$ gibt, für die alle Formeln in N wahr sind (d.h. es gibt keine Wertebelegung $\mathcal{A}$ mit $\mathcal{A}(F) = 1$ für alle $F \in N$).

Beispiel

$N = \{P \wedge Q, \neg Q \vee R\}$ ist erfüllbar:

Für $\mathcal{A} : \Pi \to \{0, 1\}$ mit $\mathcal{A}(P) = \mathcal{A}(Q) = \mathcal{A}(R) = 1$ gilt:
$\mathcal{A}(P \wedge Q) = 1$ und $\mathcal{A}(\neg Q \vee R) = 1$ (alle Formeln in N sind wahr in $\mathcal{A}$).

$N = \{P \wedge Q, \neg Q \wedge R\}$ ist nicht erfüllbar (unerfüllbar):

Für jede Wertebelegung $\mathcal{A}$ mit $\mathcal{A}(P \wedge Q) = 1$, ist $\mathcal{A}(Q) = 1$
Für jede Wertebelegung $\mathcal{A}$ mit $\mathcal{A}(\neg Q \wedge R) = 1$, ist $\mathcal{A}(Q) = 0$.

Es kann keine Wertebelegung $\mathcal{A} : \Pi \to \{0, 1\}$ geben mit
$\mathcal{A}(P \wedge Q) = 1$ und $\mathcal{A}(\neg Q \wedge R) = 1$.

Logik für Informatiker

Vorlesung 3: Zweiwertige Modelle

Babeş-Bolyai Universität, Department für Informatik, Cluj-Napoca
csacarea@cs.ubbcluj.ro

Wiederholung

- Syntax: Wie schreibt man Formeln (Propositionen, Aussagenlogische Formeln)
- Semantik: Wie berechnen wir den Wahrheitswert der Formeln?
  - Wertebelegungen (Valuationen, Modelle)
  - Wahrheitstafel für die logischen Operatoren
  - Auswertung von Formeln / Wahrheitstabellen
  - Modell einer Formel(menge)
  - Gültigkeit und Erfüllbarkeit
  - Tautologien und Kontradiktionen
  - Folgerung und Äquivalenz
- Kalküle: kommt noch...

Folgerung und Äquivalenz

Definition: F impliziert G (oder G folgt aus F),

gdw.: für alle $\mathcal{A} : \Pi \to \{0,1\}$ gilt: Wenn $\mathcal{A} \models F$, dann $\mathcal{A} \models G$.

Notation: $F \models G$

Definition: F und G sind äquivalent

gdw.: für alle $\mathcal{A} : \Pi \to \{0,1\}$ gilt: $\mathcal{A} \models F$ gdw. $A \models G$.

Notation: $F \equiv G$.

Erweiterung auf Formelmengen N in natürlicher Weise, z.B.:

$N \models G$ gdw.: für alle $\mathcal{A} : \Pi \to \{0,1\}$ gilt:
falls $\mathcal{A} \models F$, für alle $F \in N$,
so $\mathcal{A} \models G$.

Folgerung und Äquivalenz

Intuition:

- F impliziert G (oder G folgt aus F),
  gdw.: für jede Wertebelegung, für die F wahr ist, auch G wahr ist.

- Erweiterung auf Formelmengen N in natürlicher Weise, z.B.:
  $N \models G$ gdw.: für alle Wertebelegungen, für denen alle Formeln in N wahr sind, ist G auch wahr.

- Zwei Formeln F und G sind logisch äquivalent (Notation: $F \equiv G$) wenn sie in den gleichen Modellen wahr sind

Beispiel: $(P \to Q) \equiv (\neg Q \to \neg P)$ (Kontraposition)

Beispiel

$F = (A \vee C) \wedge (B \vee \neg C) \qquad G = (A \vee B)$

Überprüfe, ob $F \models G$

A	B	C	$(A \vee C)$	$(B \vee \neg C)$	$(A \vee C) \wedge (B \vee \neg C)$	$(A \vee B)$
0	0	0				
0	0	1				
0	1	0				
0	1	1				
1	0	0				
1	0	1				
1	1	0				
1	1	1				

Beispiel

$F = (A \vee C) \wedge (B \vee \neg C) \qquad G = (A \vee B)$

Überprüfe, ob $F \models G$

A	B	C	$(A \vee C)$	$(B \vee \neg C)$	$(A \vee C) \wedge (B \vee \neg C)$	$(A \vee B)$
0	0	0	0	1	0	
0	0	1	1	0	0	
0	1	0	0	1	0	
0	1	1	1	1	1	
1	0	0	1	1	1	
1	0	1	1	0	0	
1	1	0	1	1	1	
1	1	1	1	1	1	

Beispiel

$F = (A \vee C) \wedge (B \vee \neg C) \qquad G = (A \vee B)$

Überprüfe, ob $F \models G$

A	B	C	$(A \vee C)$	$(B \vee \neg C)$	$(A \vee C) \wedge (B \vee \neg C)$	$(A \vee B)$
0	0	0	0	1	0	0
0	0	1	1	0	0	0
0	1	0	0	1	0	1
0	1	1	1	1	1	1
1	0	0	1	1	1	1
1	0	1	1	0	0	1
1	1	0	1	1	1	1
1	1	1	1	1	1	1

Beispiel

$F = (A \vee C) \wedge (B \vee \neg C) \qquad G = (A \vee B)$

Überprüfe, ob $F \models G$: Ja, $F \models G$

A	B	C	$(A \vee C)$	$(B \vee \neg C)$	$(A \vee C) \wedge (B \vee \neg C)$	$(A \vee B)$
0	0	1	1	0	0	0
0	0	0	0	1	0	0
0	1	1	1	1	1	1
0	1	0	0	1	0	1
1	0	1	1	0	0	1
1	0	0	1	1	1	1
1	1	1	1	1	1	1
1	1	0	1	1	1	1

Beispiel

$F = (A \vee C) \wedge (B \vee \neg C) \qquad G = (A \vee B)$

Überprüfe, ob $F \models G$: Ja, $F \models G$

.... aber es ist nicht wahr dass $G \models F$ (Notation: $G \not\models F$)

A	B	C	$(A \vee C)$	$(B \vee \neg C)$	$(A \vee C) \wedge (B \vee \neg C)$	$(A \vee B)$
0	0	1	1	0	0	0
0	0	0	0	1	0	0
0	1	1	1	1	1	1
0	1	0	0	1	0	1
1	0	1	1	0	0	1
1	0	0	1	1	1	1
1	1	1	1	1	1	1
1	1	0	1	1	1	1

Tautologien und Kontradiktionen

Tautologien, allgemeingültige Formeln:

Formeln, die stets wahr sind.

Kontradiktionen, unerfüllbare Formeln:

Formel, die stets falsch sind.

- Die Negation einer Tautologie ist eine Kontradiktion
- Die Negation einer Kontradiktion ist eine Tautologie

Theorem. F ist allgemeingültig gdw. $\neg F$ ist unerfüllbar.

Beweis 1: Aus der Wahrheitstafel.

Beweis 2: F allgemeingültig gdw. $\mathcal{A}(F)=1$ für alle $\mathcal{A}:\Pi\to\{0,1\}$

gdw. $\mathcal{A}(\neg F)=0$ für alle $\mathcal{A}:\Pi\to\{0,1\}$ gdw. $\neg F$ unerfüllbar

Allgemeingültigkeit und Folgerung

F, G Formeln

Theorem. $F \models G$ gdw. $\models F \to G$.

Beweis:

$$
\begin{aligned}
F \models G \quad & \text{g.d.w.} \quad \text{für alle } \mathcal{A} : \Pi \to \{0,1\}, \text{ falls } \mathcal{A}(F) = 1 \text{ so } \mathcal{A}(G) = 1 \\
& \text{g.d.w.} \quad \text{für alle } \mathcal{A} : \Pi \to \{0,1\}, (\mathcal{A}(F) \to \mathcal{A}(G)) = 1 \\
& \text{g.d.w.} \quad \text{für alle } \mathcal{A} : \Pi \to \{0,1\}, \mathcal{A}(F \to G) = 1 \\
& \text{g.d.w.} \quad \models F \to G
\end{aligned}
$$

Allgemeingültigkeit und Folgerung

F, G Formeln; N Formelmenge.

Theorem. $N \cup \{F\} \models G$ gdw. $N \models F \to G$.

Beweis: "$\Rightarrow$"

Annahme: $N \cup \{F\} \models G$ d.h. für alle $\mathcal{A}{:}\Pi{\to}\{0,1\}$,
falls $[\mathcal{A}(H){=}1$ für alle Formeln $H{\in}N{\cup}\{F\}]$ so $\mathcal{A}(G){=}1$.

Wir beweisen, dass $N \models F \to G$, d.h. für alle $\mathcal{A} : \Pi \to \{0,1\}$
falls $[\mathcal{A}(H) = 1$ für alle Formeln $H \in N]$ so $\mathcal{A}(F \to G) = 1$.

Sei $\mathcal{A} : \Pi \to \{0,1\}$ mit $\mathcal{A}(H) = 1$ für alle Formeln $H \in N$.

Fall 1: $\mathcal{A}(F) = 0$. Dann $\mathcal{A}(F \to G) = 1$.

Fall 2: $\mathcal{A}(F) = 1$, d.h. $[\mathcal{A}(H) = 1$ für alle Formeln $H \in N \cup \{F\}]$. Dann $\mathcal{A}(G) = 1$ und somit $\mathcal{A}(F \to G) = 1$.

Allgemeingültigkeit und Folgerung

F, G Formeln; N Formelmenge.

Theorem. $N \cup \{F\} \models G$ gdw. $N \models F \to G$.

Beweis: "$\Leftarrow$"

Annahme: $N \models F \to G$ d.h. für alle $\mathcal{A}{:}\Pi{\to}\{0,1\}$,
falls $[\mathcal{A}(H){=}1$ für alle Formeln $H{\in}N]$ so $\mathcal{A}(F \to G){=}1$.

Wir beweisen, dass $N \cup \{F\} \models G$, d.h. für alle $\mathcal{A} : \Pi \to \{0,1\}$
falls $[\mathcal{A}(H) = 1$ für alle Formeln $H \in N \cup \{F\}]$ so $\mathcal{A}(G) = 1$.

Sei $\mathcal{A} : \Pi \to \{0,1\}$ mit $\mathcal{A}(H) = 1$ für alle Formeln $H \in N \cup \{F\}$.

Dann (i) $\mathcal{A}(F) = 1$ und
(ii) $[\mathcal{A}(H) = 1$ für alle Formeln $H \in N]$, also $\mathcal{A}(F \to G) = 1$.

Es folgt, dass $1 = \mathcal{A}(F \to G) = (\mathcal{A}(F) \to \mathcal{A}(G)) = (1 \to \mathcal{A}(G)) = \mathcal{A}(G)$,
so $\mathcal{A}(G) = 1$.

Allgemeingültigkeit und Folgerung

F, G Formeln.

Theorem. $F \equiv G$ gdw. $\models F \leftrightarrow G$.

Beweis:

$$
\begin{aligned}
F \equiv G \quad & \text{g.d.w. für alle } \mathcal{A} : \Pi \to \{0,1\},\ \mathcal{A}(F) = \mathcal{A}(G) \\
& \text{g.d.w. für alle } \mathcal{A} : \Pi \to \{0,1\},\ (\mathcal{A}(F) \leftrightarrow \mathcal{A}(G)) = 1 \\
& \text{g.d.w. für alle } \mathcal{A} : \Pi \to \{0,1\},\ \mathcal{A}(F \leftrightarrow G) = 1 \\
& \text{g.d.w. } \models F \leftrightarrow G
\end{aligned}
$$

Allgemeingültigkeit und Folgerung: Zusammenfassung

F, G Formeln; N Formelmenge.

Theorem. $F \models G$ gdw. $\models F \to G$.

Theorem. $N \cup \{F\} \models G$ gdw. $N \models F \to G$.

Theorem. $F \equiv G$ gdw. $\models F \leftrightarrow G$.

Unerfüllbarkeit, Allgemeingültigkeit und Folgerung

F, G Formeln.

Theorem. F ist allgemeingültig gdw. $\neg F$ ist unerfüllbar.

Beweis F wahr für jede Wertebelegung gdw. $\neg F$ falsch für jede Wertebelegung.

Theorem. $F \models G$ gdw. $F \wedge \neg G$ ist unerfüllbar.

Beweis:

$$\begin{array}{lll} F \models G & \text{gdw.} & \models F \to G \quad \text{d.h. } F \to G \text{ allgemeingültig} \\ & \text{gdw.} & \neg(F \to G) \text{ unerfüllbar.} \\ & \text{gdw.} & F \wedge \neg G \text{ unerfüllbar.} \end{array}$$

... da $\neg(F \to G) \equiv \neg(\neg F \vee G) \equiv \neg\neg F \wedge \neg G \equiv F \wedge \neg G$.

Unerfüllbarkeit, Allgemeingültigkeit und Folgerung

F, G Formeln; N Formelmenge.

Theorem. $N \models G$ gdw. $N \cup \{\neg G\}$ ist unerfüllbar.

Beweis: "$\Rightarrow$"

Annahme: $N \models G$ d.h. für alle $\mathcal{A}{:}\Pi \to \{0,1\}$,
falls $[\mathcal{A}(H){=}1$ für alle Formeln $H \in N]$ so $\mathcal{A}(G){=}1$.

Zu zeigen: $N \cup \{\neg G\}$ unerfüllbar.

Beweis durch Widerspruch: Wir nehmen an, $N \cup \{\neg G\}$ erfüllbar, d.h. es gibt $\mathcal{A}{:}\Pi \to \{0,1\}$, mit

$$[\mathcal{A}(H){=}1 \text{ für alle Formeln } H \in N \cup \{\neg G\}].$$

Dann $[\mathcal{A}(H){=}1$ für alle Formeln $H \in N]$ und $\mathcal{A}(\neg G) = 1$ (d.h. $\mathcal{A}(G) = 0$). Widerspruch.

Unerfüllbarkeit, Allgemeingültigkeit und Folgerung

F, G Formeln; N Formelmenge.

Theorem. $N \models G$ gdw. $N \cup \{\neg G\}$ ist unerfüllbar.

Beweis: "$\Leftarrow$"

Annahme: $N \cup \{\neg G\}$ unerfüllbar.

Zu zeigen: $N \models G$ d.h. für alle $\mathcal{A}{:}\Pi \to \{0,1\}$,
falls $[\mathcal{A}(H){=}1$ für alle Formeln $H{\in}N]$ so $\mathcal{A}(G){=}1$.

Beweis: Sei $\mathcal{A}{:}\Pi \to \{0,1\}$, mit $[\mathcal{A}(H){=}1$ für alle Formeln $H{\in}N]$.

Falls $\mathcal{A}(G) = 0$, wäre $\mathcal{A}$ ein Modell für $N \cup \{\neg G\}$. Das ist aber unmöglich, da wir angenommen haben, dass $N \cup \{\neg G\}$ unerfüllbar ist.

Es folgt, dass $\mathcal{A}(G) = 1$.

Unerfüllbarkeit, Allgemeingültigkeit und Folgerung: Zusammenfassung

F, G Formeln; N Formelmenge.

Theorem. F ist allgemeingültig gdw. $\neg F$ ist unerfüllbar.

Theorem. $F \models G$ gdw. $F \wedge \neg G$ ist unerfüllbar.

Theorem. $N \models G$ gdw. $N \cup \{\neg G\}$ ist unerfüllbar.

Nota bene: falls N unerfüllbar, so $N \models G$ für jede Formel G
... auch für $\perp$.

Notation: $N \models \perp$ für N unerfüllbar.

Strukturelle Induktion für Aussagenlogik

Zu zeigen:

Alle Formeln haben Eigenschaft p
Für alle $F \in PROP$, gilt $p(F)$.

Induktionsanfang:

Wir beweisen, dass $p(A)$ für alle atomaren Formeln gilt.
Beweise $p(\bot), p(\top)$ und $p(q_i)$ für die aussagenlogischen Konstanten q_i, $i \in \mathbb{N}$.

Strukturelle Induktion für Aussagenlogik

Induktionsvoraussetzung:

Sei F eine Formel (die nicht atomar ist).
Annahme: alle Teilformeln von F, die nicht gleich F sind, haben Eigenschaft p.

Induktionsschritt:

Zeige, dass auch F Eigenschaft p hat.

Strukturelle Induktion für Aussagenlogik

Beweis durch Fallunterscheidung:
Fall 1: $F = \neg G$. Induktionvoraussetzung: $p(G)$ gilt. Folgere, dass $p(F)$ gilt.
Fall 2: $F = G \wedge H$. Induktionvoraussetzung: $p(G), p(H)$ gelten. Folgere, dass $p(F)$ gilt.
Fall 3: $F = G \vee H$. Induktionvoraussetzung: $p(G), p(H)$ gelten. Folgere, dass $p(F)$ gilt.
Fall 4: $F = G \rightarrow H$. Induktionvoraussetzung: $p(G), p(H)$ gelten. Folgere, dass $p(F)$ gilt.
Fall 5: $F = G \leftrightarrow H$. Induktionvoraussetzung: $p(G), p(H)$ gelten. Folgere, dass $p(F)$ gilt.

Zweiwertige Interpretation aussagenlogischer Formeln

Sei $\mathcal{A}\colon PC \to \{0,1\}$ eine Belegung. Die zweiwertige Interpretation $\mathcal{A}^*$ wird induktiv über den Aufbau von PROP wie folgt definiert:
$\mathcal{A}^*(\bot) = 0$,
$\mathcal{A}^*(\top) = 1$,
$\mathcal{A}^*(p) = \mathcal{A}(p)$, für alle $p \in PC$,
$\mathcal{A}^*(\neg F) = 1 - \mathcal{A}^*(F)$,
$\mathcal{A}^*(F\ op\ G) = B_{op}(\mathcal{A}^*(F), \mathcal{A}^*(G))$, $B_{op}(x,y)$ wird entsprechend der Wahrheitstafel der Operation $\mathtt{op}$ berechnet.
Wir scheiben normalerweise $\mathcal{A}$ statt $\mathcal{A}^*$ und $\mathtt{op}$ statt B_{op}.

Tautologien. Wiederholung:

1. $(p \to \neg p) \to (\neg p)$
2. $(p \wedge (p \to q)) \to q$
3. $(p \wedge q) \to p$
4. $(p \wedge q) \to q$
5. $p \to (p \vee q)$
6. $q \to (p \vee q)$
7. $(p \to q) \to [(q \to r) \to (p \to r)]$
8. $(((p \to q) \wedge (q \to r)) \wedge p) \to r$

Deduktionsmechanismen: Wahrheitstafelmethode

Jede Formel F enthält endlich viele Aussagenvariablen (propositionale Konstanten). $\mathcal{A}(F)$ ist nur von den Werten dieser Aussagenvariablen abhängig.
F enthält n Aussagenvariablen: Es folgt 2^n Wertbelegungen notwendig um zu uberprüfen, ob F erfüllbar/unerfüllbar/allgemeingültig ist oder nicht.
Dafür verwenden wir die Wahrheitstafel.

Deduktionsmechanismen: Wahrheitstafelmethode

Ein erster Kalkül $\rightarrow$ Wahrheitstabelle.

- F allgemeingültig (Tautologie): $\mathcal{A}(F) = 1$ für alle Wertbelegungen,
- F erfüllbar: $\mathcal{A}(F) = 1$ für zumindest eine Wertbelegung,
- F unerfüllbar: $\mathcal{A}(F) = 0$ für alle Wertbelegungen.

Äquivalenzen

Zwei Formeln F und G sind logisch äquivalent (Notation: $F \equiv G$) wenn sie in den gleichen Modellen wahr sind

Beispiel: $(P \to Q) \equiv (\neg Q \to \neg P)$ (Kontraposition)

Anwendung: Wichtige Äquivalenzen

Die folgenden Äquivalenzen sind für alle Formeln F, G, H gültig:

$$(F \wedge F) \equiv F$$
$$(F \vee F) \equiv F \qquad \text{(Idempotenz)}$$

$$(F \wedge G) \equiv (G \wedge F)$$
$$(F \vee G) \equiv (G \vee F) \qquad \text{(Kommutativität)}$$

$$(F \wedge (G \wedge H)) \equiv ((F \wedge G) \wedge H)$$
$$(F \vee (G \vee H)) \equiv ((F \vee G) \vee H) \qquad \text{(Assoziativität)}$$

$$(F \wedge (F \vee G)) \equiv F$$
$$(F \vee (F \wedge G)) \equiv F \qquad \text{(Absorption)}$$

$$(F \wedge (G \vee H)) \equiv ((F \wedge G) \vee (F \wedge H))$$
$$(F \vee (G \wedge H)) \equiv ((F \vee G) \wedge (F \vee H)) \qquad \text{(Distributivität)}$$

Wichtige Äquivalenzen

Die folgenden Äquivalenzen sind für alle Formeln F, G, H gültig:

$(\neg\neg F) \equiv F$	(Doppelte Negation)
$\neg(F \wedge G) \equiv (\neg F \vee \neg G)$	
$\neg(F \vee G) \equiv (\neg F \wedge \neg G)$	(De Morgan's Regeln)
$(F \rightarrow G) \equiv (\neg G \rightarrow \neg F)$	(Kontraposition)
$(F \rightarrow G) \equiv (\neg F \vee G)$	(Elimination Implikation)
$F \leftrightarrow G \equiv (F \rightarrow G) \wedge (G \rightarrow F)$	(Elimination Äquivalenz)

Wichtige Äquivalenzen

Die folgenden Äquivalenzen sind für alle Formeln F, G, H gültig:

$(F \wedge G) \equiv F$, falls G Tautologie

$(F \vee G) \equiv \top$, falls G Tautologie (Tautologieregeln)

$(F \wedge G) \equiv \bot$, falls G unerfüllbar

$(F \vee G) \equiv F$, falls G unerfüllbar (Tautologieregeln)

Wichtige Äquivalenzen für $\bot/\top$

$$(A \wedge \neg A) \equiv \bot$$

$$(A \vee \neg A) \equiv \top \qquad \text{(Tertium non datur)}$$

$$(A \wedge \top) \equiv A$$

$$(A \wedge \bot) \equiv \bot$$

Wichtige Äquivalenzen: Zusammengefasst

$(F \wedge F) \equiv F$	$(F \vee F) \equiv F$	(Idempotenz)
$(F \wedge G) \equiv (G \wedge F)$	$(F \vee G) \equiv (G \vee F)$	(Kommutativität)
$(F \wedge (G \wedge H)) \equiv ((F \wedge G) \wedge H)$		
$(F \vee (G \vee H)) \equiv ((F \vee G) \vee H)$		(Assoziativität)
$(F \wedge (F \vee G)) \equiv F$		
$(F \vee (F \wedge G)) \equiv F$		(Absorption)
$(F \wedge (G \vee H)) \equiv ((F \wedge G) \vee (F \wedge H))$		
$(F \vee (G \wedge H)) \equiv ((F \vee G) \wedge (F \vee H))$		(Distributivität)
$(\neg\neg F) \equiv F$		(Doppelte Negation)
$\neg(F \wedge G) \equiv (\neg F \vee \neg G)$		
$\neg(F \vee G) \equiv (\neg F \wedge \neg G)$		(De Morgan's Regeln)
$(F \rightarrow G) \equiv (\neg G \rightarrow \neg F)$		(Kontraposition)
$(F \rightarrow G) \equiv (\neg F \vee G)$		(Elimination Implikation)
$F \leftrightarrow G \equiv (F \rightarrow G) \wedge (G \rightarrow F)$		(Elimination Äquivalenz)

Ein erster Kalkül: Wahrheitstabellen

Erfüllbarkeitstest:

Jede Formel F enthält endlich viele Aussagenvariablen.
$\mathcal{A}(F)$ ist nur von den Werten dieser Aussagenvariablen abhängig.

F enthält n Aussagenvariablen:

$\Rightarrow$ 2^n Wertbelegungen notwendig um zu überprüfen, ob F erfüllbar ist oder nicht.

$\Rightarrow$ Wahrheitstafel

$\Rightarrow$ Das Erfüllbarkeitsproblem ist entscheidbar

Es existieren viel bessere Methoden als Wahrheitstafeltests um die Erfüllbarkeit einer Formel zu überprüfen.

Ein zweiter Kalkül: logische Umformung

Definition:

Äquivalenzumformung:

- (Wiederholte) Ersetzung einer (Unter-)Formel durch äquivalente Formel

Teilformeln

Eine Formel F, die als Teil einer Formel G auftritt, heißt Teilformel von G.

- F ist eine Teilformel von F
- $\left.\begin{array}{l} F = \neg G \text{ und} \\ H \text{ Teilformel von } G \end{array}\right\} \rightarrow \quad H$ Teilformel von F
- $\left.\begin{array}{l} F = F_1 \ \rho \ F_2 \\ (\text{wo } \rho \in \{\vee, \wedge, \Rightarrow, \Leftrightarrow\}) \\ H \text{ Teilformel von } F_1 \text{ oder } F_2 \end{array}\right\} \rightarrow \quad H$ Teilformel von F

Substitutionstheorem

Theorem.
Seien F und G äquivalente Formeln. Sei H eine Formel mit (mindestens) einem Vorkommen der Teilformel F.

Dann ist H äquivalent zu H', wobei H' aus H hervorgeht, indem (irgend) ein Vorkommen von F in H durch G ersetzt wird.

Beispiel:

$A \vee B \equiv B \vee A$

impliziert

$(C \wedge (A \vee B)) \equiv (C \wedge (B \vee A))$

Substitutionstheorem

Theorem.
Seien F und G äquivalente Formeln. Sei H eine Formel mit (mindestens) einem Vorkommen der Teilformel F.

Dann ist H äquivalent zu H', wobei H' aus H hervorgeht, indem (irgend) ein Vorkommen von F in H durch G ersetzt wird. $p(H)$

Beweis: Strukturelle Induktion.

Induktionsbasis: Beweisen, dass $p(H)$ für alle Formeln H in $\{\bot, \top\} \cup \Pi$ gilt.

Beweis: Falls $H \in \{\bot, \top\} \cup \Pi$ und F Teilformel von H, so muss $F = H$ sein. Dann ist die Formel H', die aus H hervorgeht, indem F (= die ganze Formel H) durch G ersetzt wird, gleich G.

Aber dann: $H = F \equiv G = H'$.

Substitutionstheorem

Beweis: (Fortsetzung)
Sei H eine Formel, $H \notin \{\bot, \top\} \cup \Pi$. Sei F eine Teilformel von H.

Fall 1: $F = H$. Dann $H' = G$ (wie vorher), so $H = F \equiv G = H'$.

Fall 2: $F \neq H$.

Induktionsvoraussetzung: Annahme: $p(H')$ gilt für alle dir. Teilformeln H' von H.

Induktionsschritt: Beweis, dass $p(H)$ gilt (durch Fallunterscheidung):

Fall 2.1: $H = \neg H_1$. Da $F \neq H$, ist F eine Teilformel von H_1.

Induktionvoraussetzung: $p(H_1)$ gilt, d.h. $H_1 \equiv H_1'$, wobei H_1' aus H_1 hervorgeht, indem (irgend) ein Vorkommen von F in H_1 durch G ersetzt wird.

Da $H = \neg H_1$, ist $H' = \neg H_1'$.

Dann für alle $\mathcal{A} : \Pi \to \{0, 1\}$: $\mathcal{A}(H) = \mathcal{A}(\neg H_1) = \neg \mathcal{A}(H_1) \overset{I.V.}{=} \neg \mathcal{A}(H_1') = \mathcal{A}(\neg H_1') = \mathcal{A}(H')$

Somit ist bewiesen, dass $H \equiv H'$.

Substitutionstheorem

Beweis: (Fortsetzung)
Induktionsschritt: Beweis, dass $p(H)$ gilt (durch Fallunterscheidung):

Fall 2.2: $H = H_1$ op H_2. Da $F \neq H$, ist F Teilformel von H_1 oder von H_2.

Fall 2.2.1 F ist eine Teilformel von H_1.
Induktionvoraussetzung: $p(H_1)$ gilt, d.h. $H_1 \equiv H_1'$, wobei H_1' aus H_1 hervorgeht, indem (irgend) ein Vorkommen von F in H_1 durch G ersetzt wird.

Da $H = H_1$ op H_2, und F in H_1 vorkommt, so $H' = H_1'$ op H_2.
Dann für alle $\mathcal{A} : \Pi \to \{0,1\}$: $\mathcal{A}(H) = \mathcal{A}(H_1 \text{ op } H_2) = \mathcal{A}(H_1) \text{ op } \mathcal{A}(H_2) \overset{I.V.}{=} \mathcal{A}(H_1') \text{ op } \mathcal{A}(H_2) = \mathcal{A}(H_1' \text{ op } H_2) = \mathcal{A}(H')$.

Somit ist bewiesen, dass $H \equiv H'$.

Fall 2.2.2 F ist eine Teilformel von H_2. Analog.

Ein zweiter Kalkül: logische Umformung

Definition:

Äquivalenzumformung:

- (Wiederholte) Ersetzung einer (Unter-)Formel durch äquivalente Formel
- Anwendung des Substitutionstheorems

Logik für Informatiker

Vorlesung 4: Zweiwertige Modelle

Babeş-Bolyai Universität, Department für Informatik, Cluj-Napoca
csacarea@cs.ubbcluj.ro

Ein zweiter Kalkül: logische Umformung

Definition:

Äquivalenzumformung:

- (Wiederholte) Ersetzung einer (Unter-)Formel durch äquivalente Formel

Teilformeln

Eine Formel F, die als Teil einer Formel G auftritt, heißt Teilformel von G.

- F ist eine Teilformel von F
- $\left.\begin{array}{l} F = \neg G \text{ und} \\ H \text{ Teilformel von } G \end{array}\right\} \rightarrow \quad H$ Teilformel von F
- $\left.\begin{array}{l} F = F_1 \ \rho \ F_2 \\ (\text{wo } \rho \in \{\vee, \wedge, \Rightarrow, \Leftrightarrow\}) \\ H \text{ Teilformel von } F_1 \text{ oder } F_2 \end{array}\right\} \rightarrow \quad H$ Teilformel von F

Substitutionstheorem

Theorem.
Seien F und G äquivalente Formeln. Sei H eine Formel mit (mindestens) einem Vorkommen der Teilformel F.

Dann ist H äquivalent zu H', wobei H' aus H hervorgeht, indem (irgend) ein Vorkommen von F in H durch G ersetzt wird.

Beispiel:

$A \vee B \equiv B \vee A$

impliziert

$(C \wedge (A \vee B)) \equiv (C \wedge (B \vee A))$

Substitutionstheorem

Theorem.
Seien F und G äquivalente Formeln. Sei H eine Formel mit (mindestens) einem Vorkommen der Teilformel F.

Dann ist H äquivalent zu H', wobei H' aus H hervorgeht, indem (irgend) ein Vorkommen von F in H durch G ersetzt wird. $p(H)$

Beweis: Strukturelle Induktion.

Induktionsbasis: Beweisen, dass $p(H)$ für alle Formeln H in $\{\bot, \top\} \cup \Pi$ gilt.

Beweis: Falls $H \in \{\bot, \top\} \cup \Pi$ und F Teilformel von H, so muss $F = H$ sein. Dann ist die Formel H', die aus H hervorgeht, indem F (= die ganze Formel H) durch G ersetzt wird, gleich G.

Aber dann: $H = F \equiv G = H'$.

Substitutionstheorem

Beweis: (Fortsetzung)
Sei H eine Formel, $H \notin \{\bot, \top\} \cup \Pi$. Sei F eine Teilformel von H.

Fall 1: $F = H$. Dann $H' = G$ (wie vorher), so $H = F \equiv G = H'$.

Fall 2: $F \neq H$.

Induktionsvoraussetzung: Annahme: $p(H')$ gilt für alle dir. Teilformeln H' von H.

Induktionsschritt: Beweis, dass $p(H)$ gilt (durch Fallunterscheidung):

Fall 2.1: $H = \neg H_1$. Da $F \neq H$, ist F eine Teilformel von H_1.

Induktionvoraussetzung: $p(H_1)$ gilt, d.h. $H_1 \equiv H_1'$, wobei H_1' aus H_1 hervorgeht, indem (irgend) ein Vorkommen von F in H_1 durch G ersetzt wird.

Da $H = \neg H_1$, ist $H' = \neg H_1'$.

Dann für alle $\mathcal{A} : \Pi \to \{0, 1\}$: $\mathcal{A}(H) = \mathcal{A}(\neg H_1) = \neg \mathcal{A}(H_1) \overset{I.V.}{=} \neg \mathcal{A}(H_1') = \mathcal{A}(\neg H_1') = \mathcal{A}(H')$

Somit ist bewiesen, dass $H \equiv H'$.

Substitutionstheorem

Beweis: (Fortsetzung)
Induktionsschritt: Beweis, dass $p(H)$ gilt (durch Fallunterscheidung):

Fall 2.2: $H = H_1$ op H_2. Da $F \neq H$, ist F Teilformel von H_1 oder von H_2.

Fall 2.2.1 F ist eine Teilformel von H_1.
Induktionvoraussetzung: $p(H_1)$ gilt, d.h. $H_1 \equiv H_1'$, wobei H_1' aus H_1 hervorgeht, indem (irgend) ein Vorkommen von F in H_1 durch G ersetzt wird.

Da $H = H_1$ op H_2, und F in H_1 vorkommt, so $H' = H_1'$ op H_2.
Dann für alle $\mathcal{A} : \Pi \to \{0,1\}$: $\mathcal{A}(H) = \mathcal{A}(H_1 \text{ op } H_2) = \mathcal{A}(H_1) \text{ op } \mathcal{A}(H_2) \overset{I.V.}{=} \mathcal{A}(H_1') \text{ op } \mathcal{A}(H_2) = \mathcal{A}(H_1' \text{ op } H_2) = \mathcal{A}(H')$.

Somit ist bewiesen, dass $H \equiv H'$.

Fall 2.2.2 F ist eine Teilformel von H_2. Analog.

Ein zweiter Kalkül: logische Umformung

Definition:

Äquivalenzumformung:

- (Wiederholte) Ersetzung einer (Unter-)Formel durch äquivalente Formel
- Anwendung des Substitutionstheorems

Ein zweiter Kalkül: logische Umformung

Theorem

Äquivalenzumformung bildet mit den aufgelisteten wichtigen Äquivalenzen einen vollständigen Kalkul:
Wenn F und G logisch äquivalent sind, kann F in G umgeformt werden.

Anwendung

Test für Erfüllbarkeit/Unerfüllbarkeit/Allgemeingültigkeit.

Beispiel

$$(P \to Q) \wedge \neg((Q \to R) \to (P \to R)).$$

Beispiel

$$(P \to Q) \land \neg((Q \to R) \to (P \to R))$$

$\equiv (\neg P \lor Q) \land \neg((\neg Q \lor R) \to (\neg P \lor R))$	(Elimination Implikation)
$\equiv (\neg P \lor Q) \land \neg(\neg(\neg Q \lor R) \lor (\neg P \lor R))$	(Elimination Implikation)
$\equiv (\neg P \lor Q) \land (\neg\neg(\neg Q \lor R) \land \neg(\neg P \lor R))$	(De Morgan's Regel, $\lor$)
$\equiv (\neg P \lor Q) \land \quad ((\neg Q \lor R) \land (\neg\neg P \land \neg R))$	(Doppelte Negation, De Morgan, $\lor$)
$\equiv (\neg P \lor Q) \land ((\neg Q \lor R) \land (P \land \neg R))$	(Doppelte Negation)
$\equiv (\neg P \lor Q) \land ((\neg Q \land P \land \neg R) \lor (R \land P \land \neg R))$	(Distributivität)
$\equiv (\neg P \lor Q) \land ((\neg Q \land P \land \neg R) \lor (R \land \neg R \land P))$	(Kommutativität)
$\equiv (\neg P \lor Q) \land ((\neg Q \land P \land \neg R) \lor \bot)$	(Äquivalenzen mit $\bot$)
$\equiv (\neg P \lor Q) \land (\neg Q \land P \land \neg R)$	(Äquivalenzen mit $\bot$)
$\equiv (\neg P \land \neg Q \land P \land \neg R) \lor (Q \land \neg Q \land P \land \neg R)$	(Distributivität)
$\equiv (\neg P \land P \land \neg Q \land \neg R) \lor (Q \land \neg Q \land P \land \neg R)$	(Kommutativität)
$\equiv ((\neg P \land P) \land \neg Q \land \neg R) \lor ((Q \land \neg Q) \land P \land \neg R)$	(Assoziativität)
$\equiv \bot \lor \bot \equiv \bot$	(Äquivalenzen mit $\bot$)

Kalküle

- Wahrheitstabellen
- Äquivalenzumformungen
- nicht besonders effizient...
- Ziel: Kalkül(e) zur systematischen Überprüfung von Erfüllbarkeit (für Formeln und/oder Formelmengen)

Normalformen

Definition:

- Atom: aussagenlogische Variable
- Literal: Atom, oder

  Negation eines Atoms

Beispiel. Sei $\Pi = \{P, Q, R\}$.

Atome: $P, \quad Q, \quad R$

Literale: $P, \quad \neg P, \quad Q, \quad \neg Q, \quad R, \quad \neg R$

Normalformen

Definition:

- Atom: aussagenlogische Variable
- Literal: Atom, oder

  Negation eines Atoms

Definition:

Klausel: Eine Disjunktion von Literalen

- mehrstellige Disjunktionen $(P \vee \neg Q \vee R)$, $(P \vee P \vee \neg Q)$
- einstellige Disjunktionen P
- die nullstellige Disjunktion (leere Klausel) $\perp$

Normalformen

Definition:

Konjunktive Normalform (KNF): Eine Konjunktion von Disjunktionen von Literalen, d.h., eine Konjunktion von Klauseln

mehrstellig, einstellig oder nullstellig

Beispiele:

$(P \vee \neg Q) \wedge (Q \vee \neg R \vee \neg S)$

$P \vee Q$

$P \wedge (Q \vee R)$

$P \wedge Q$

$P \wedge P$

$\top$

Normalformen

Definition:

Disjunktive Normalform (DNF): Eine Disjunktion von Konjunktionen von Literalen.

mehrstellig, einstellig oder nullstellig

Beispiele:

$(P \wedge \neg Q) \vee (Q \wedge \neg R \wedge \neg S)$

$P \wedge Q$

$P \vee (Q \wedge R)$

$P \vee Q$

$P \vee P$

$\perp$

Normalformen

Eigenschaften:

- Zu jeder aussagenlogischen Formel gibt es:
  - eine äquivalente Formel in KNF
  - eine äquivalente Formel in DNF
- Diese äquivalenten Formeln in DNF bzw. KNF sind nicht eindeutig
- Solche Formeln können aus einer Wahrheitstafel abgelesen werden
- Solche Formeln können durch Umformungen hergestellt werden

Normalformen

Eigenschaften:

- Zu jeder aussagenlogischen Formel gibt es:
  - eine äquivalente Formel in KNF
  - eine äquivalente Formel in DNF
- Diese äquivalenten Formeln in DNF bzw. KNF sind nicht eindeutig
- Solche Formeln können aus einer Wahrheitstafel abgelesen werden
- Solche Formeln können durch Umformungen hergestellt werden

Beispiel

$F: \quad (P \vee Q) \wedge ((\neg P \wedge Q) \vee R)$

P	Q	R	$(P \vee Q)$	$\neg P$	$(\neg P \wedge Q)$	$((\neg P \wedge Q) \vee R)$	F
0	0	0	0	1	0	0	0
0	0	1	0	1	0	1	0
0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	1
1	0	0	1	0	0	0	0
1	0	1	1	0	0	1	1
1	1	0	1	0	0	0	0
1	1	1	1	0	0	1	1

Beispiel: DNF

$F: \quad (P \vee Q) \wedge ((\neg P \wedge Q) \vee R)$

P	Q	R	$(P \vee Q)$	$\neg P$	$(\neg P \wedge Q)$	$((\neg P \wedge Q) \vee R)$	F
0	0	0	0	1	0	0	0
0	0	1	0	1	0	1	0
0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	1
1	0	0	1	0	0	0	0
1	0	1	1	0	0	1	1
1	1	0	1	0	0	0	0
1	1	1	1	0	0	1	1

Beispiel: DNF

$F: \quad (P \vee Q) \wedge ((\neg P \wedge Q) \vee R)$

P	Q	R	$(P \vee Q)$	$\neg P$	$(\neg P \wedge Q)$	$((\neg P \wedge Q) \vee R)$	F
0	0	0	0	1	0	0	0
0	0	1	0	1	0	1	0
0	1	0	1	1	1	1	1
0	1	1	1	1	1	1	1
1	0	0	1	0	0	0	0
1	0	1	1	0	0	1	1
1	1	0	1	0	0	0	0
1	1	1	1	0	0	1	1

DNF: $(\neg P \wedge Q \wedge \neg R) \vee (\neg P \wedge Q \wedge R) \vee (P \wedge \neg Q \wedge R) \vee (P \wedge Q \wedge R)$

Beispiel: KNF

$F: \quad (P \vee Q) \wedge ((\neg P \wedge Q) \vee R)$

P	Q	R	$(P \vee Q)$	$\neg P$	$(\neg P \wedge Q)$	$((\neg P \wedge Q) \vee R)$	F	$\neg F$
0	0	0	0	1	0	0	0	1
0	0	1	0	1	0	1	0	1
0	1	0	1	1	1	1	1	0
0	1	1	1	1	1	1	1	0
1	0	0	1	0	0	0	0	1
1	0	1	1	0	0	1	1	0
1	1	0	1	0	0	0	0	1
1	1	1	1	0	0	1	1	0

DNF für F: $(\neg P \wedge Q \wedge \neg R) \vee (\neg P \wedge Q \wedge R) \vee (P \wedge \neg Q \wedge R) \vee (P \wedge Q \wedge R)$

DNF für $\neg F$: $(\neg P \wedge \neg Q \wedge \neg R) \vee (\neg P \wedge \neg Q \wedge R) \vee (P \wedge \neg Q \wedge \neg R) \vee (P \wedge Q \wedge \neg R)$

KNF für F: $(P \vee Q \vee R) \wedge (P \vee Q \vee \neg R) \wedge (\neg P \vee Q \vee R) \wedge (\neg P \vee \neg Q \vee R)$

Normalformen

DNF für F:

$$\boxed{\bigvee_{\substack{\mathcal{A}:\{P_1,\ldots,P_n\}\to\{0,1\}\\ \mathcal{A}(F)=1}} (P_1^{\mathcal{A}(P_1)} \wedge \cdots \wedge P_n^{\mathcal{A}(P_n)})}$$

wobei:

$P^0 = \neg P$

$P^1 = P$

Theorem

Für alle Interpretationen $\mathcal{A}' : \{P_1, \ldots, P_n\} \to \{0, 1\}$:

$$\mathcal{A}'(F) = 1 \quad \text{gdw.} \quad \mathcal{A}'\Big(\bigvee_{\substack{\mathcal{A}:\{P_1,\ldots,P_n\}\to\{0,1\}\\ \mathcal{A}(F)=1}} (P_1^{\mathcal{A}(P_1)} \wedge \cdots \wedge P_n^{\mathcal{A}(P_n)})\Big) = 1.$$

Normalformen

DNF für F:

$$\boxed{\bigvee_{\substack{\mathcal{A}:\{P_1,\ldots,P_n\}\to\{0,1\}\\ \mathcal{A}(F)=1}} (P_1^{\mathcal{A}(P_1)} \wedge \cdots \wedge P_n^{\mathcal{A}(P_n)})}$$

wobei:

$P^0 = \neg P$

$P^1 = P$

KNF für F: $\neg F'$,

wobei F' die DNF von $\neg F$ ist.

KNF für F:

$$\boxed{\bigwedge_{\substack{\mathcal{A}:\{P_1,\ldots,P_n\}\to\{0,1\}\\ \mathcal{A}(F)=0}} (P_1^{1-\mathcal{A}(P_1)} \vee \cdots \vee P_n^{1-\mathcal{A}(P_n)})}$$

Umformung in KNF

Vier Schritte:

1. Elimination von $\leftrightarrow$

   Verwende $A \leftrightarrow B \equiv (A \rightarrow B) \wedge (B \rightarrow A)$

2. Elimination von $\rightarrow$

   Verwende $A \rightarrow B \equiv (\neg A \vee B)$

3. "Nach innen schieben" von $\neg$

   Verwende de Morgans Regeln und $\neg\neg A \equiv A$

$\mapsto$ Negationsnormalform (NNF)

Eine logische Formel ist in Negationsnormalform (NNF), falls die Negationsoperatoren in ihr nur direkt über atomaren Aussagen vorkommen.

Umformung in KNF

Vier Schritte:

1. Elimination von $\leftrightarrow$

   Verwende $A \leftrightarrow B \equiv (A \rightarrow B) \wedge (B \rightarrow A)$

2. Elimination von $\rightarrow$

   Verwende $A \rightarrow B \equiv (\neg A \vee B)$

3. "Nach innen schieben" von $\neg$

   Verwende de Morgans Regeln und $\neg\neg A \equiv A$ (NNF)

4. "Nach innen schieben" von $\vee$

   Verwende Distributivität von $\vee$ über $\wedge$

Umformung in KNF: Beispiel

Gegeben:

$$P \leftrightarrow (Q \vee R)$$

1. Elimination von $\leftrightarrow$

$$(P \rightarrow (Q \vee R)) \wedge ((Q \vee R) \rightarrow P)$$

2. Elimination von $\rightarrow$

$$(\neg P \vee Q \vee R) \wedge (\neg(Q \vee R) \vee P)$$

3. "Nach innen schieben" von $\neg$ (NNF)

$$(\neg P \vee Q \vee R) \wedge ((\neg Q \wedge \neg R) \vee P)$$

4. "Nach innen schieben" von $\vee$

$$(\neg P \vee Q \vee R) \wedge (\neg Q \vee P) \wedge (\neg R \vee P)$$

Umformung in DNF

Vier Schritte:

1. Elimination von $\leftrightarrow$

   Verwende $A \leftrightarrow B \equiv (A \rightarrow B) \wedge (B \rightarrow A)$

2. Elimination von $\rightarrow$

   Verwende $A \rightarrow B \equiv (\neg A \vee B)$

3. "Nach innen schieben" von $\neg$

   Verwende de Morgans Regeln und $\neg\neg A \equiv A$ (NNF)

4. "Nach innen schieben" von $\wedge$

   Verwende Distributivität von $\wedge$ über $\vee$

KNF: Mengenschreibweise

Notation:

Klausel als Menge von Literalen

Formel in KNF als Menge von Klauseln

Beispiel:

$$(P \vee Q \vee R) \wedge (P \vee Q \vee \neg R) \wedge (\neg P \vee Q \vee R) \wedge (\neg P \vee \neg Q \vee R)$$

$$\{ \ \{P, Q, R\}, \ \{P, Q, \neg R\}, \ \{\neg P, Q, R\}, \ \{\neg P, \neg Q, R\} \ \}$$

KNF: Mengenschreibweise

Bedeutung der leeren Menge

- Leere Klausel

  $=$ leere Menge von Literalen

  $=$ leere Disjunktion

  $= \bot$

- Leere Menge von Klausels

  $=$ leere Konjunktion

  $= \top$

Vereinfachung der KNF: Subsumption

Theorem (Subsumption Regel)
Enthält eine KNF-Formel (= Klauselmenge) Klauseln K, K' mit

$$K \subset K'$$

dann entsteht eine äquivalente Formel, wenn K' weggelassen wird.

Beweis:

$K = \{L_1, \ldots, L_p\} \subseteq \{L_1, \ldots, L_p, L_{p+1}, \ldots, L_m\} = K'$

F enthält $K \wedge K'$

$$\begin{aligned} K \wedge K' = & \quad (L_1 \vee \cdots \vee L_p) \wedge ((L_1 \vee \cdots \vee L_p) \vee L_{p+1} \vee \ldots L_m) \\ \equiv & \quad (L_1 \vee \cdots \vee L_p) = K \qquad \text{(Absorption)} \end{aligned}$$

Erfüllbarkeit und Allgemeingültigkeit

- Syntax
- Semantik
- Formeln
- Schön, aber wem hilft's?

$\Longrightarrow$ Automatische Bestimmung der Erfüllbarkeit und Allgemeingültigkeit!

Erfüllbarkeit und Allgemeingültigkeit

- Eine Formel ist erfüllbar: Es existiert eine Variablenbelegung, so dass die Formel wahr ist.
- Algorithmus?

Erfüllbarkeit und Allgemeingültigkeit

- Eine Formel ist erfüllbar: Es existiert eine Variablenbelegung, so dass die Formel wahr ist.
- Algorithmus?
- Brute Force: Probiere alle Möglichkeiten und wähle diejenigen, die die Formel wahr machen.

Erfüllbarkeit und Allgemeingültigkeit

- Die Formel ist allgemeingültig (Tautologie) falls alle Belegungen die Formel wahr machen!
- Algorithmus?

Erfüllbarkeit und Allgemeingültigkeit

- Die Formel ist allgemeingültig (Tautologie) falls alle Belegungen die Formel wahr machen!
- Algorithmus?
- Brute Force: Probiere alle Möglichkeiten und überprüfe, ob sie die Formel wahr machen.
- Gibt es bessere Algorithmen? JA!

Satisfiability Problems - Erfüllbarkeit

- Viele Probleme können als eine Liste von Bedingungen/Einschränkungen (engl. constraints) beschrieben werden $\rightarrow$ wir suchen Belegungen, die diese Einschränkungen erfüllen.
- Erfolgreich - JA! $\Rightarrow$ Problem gelöst, Antwort gefunden.
- Beispiele: Stundenplan an der Uni
- Schichtenarbeit im Krankenhaus
- Ein jeder hat seine eigene Bedingungen: Vorlesungen dürfen nicht in Laborräume verlegt werden, Termine dürfen sich nicht überlappen, es sollten nicht mehr als 10 Stunden/Tag eingeplant werden, etc.
- Viele solcher Situationen können als eine Liste von Einschränkungen beschrieben werden.

Existiert eine Zurodnung von Stunden/Räume, bzw. Personal, die alle Einschränkungen erfüllt?

Satisfiability Problems - Erfüllbarkeit

- Programme automatisch (!) testen
- Spezifiziere die Logik Einschränkungen logical constraints). Z.B. Kontrollsystem für Bahnfahrt
- Beschreibe die Situation *Zwei Züge fahren aufeinander auf derselben Gleisstrecke* als logische Aussage
- Teste ob die logischen Einschränkungen + obige Situation sind gleichzeitig erfüllbar
- JA? $\Rightarrow$ Systemfehler!

KNF

- Erfüllbarkeitsprobleme werden als KNF Formeln geschrieben:

$$(A \vee B \vee \neg C) \wedge (B \vee D) \wedge (\neg A) \wedge (B \vee C)$$

- $(A \vee B \vee \neg C)$ ist eine Klausel:

KNF

- Erfüllbarkeitsprobleme werden als KNF Formeln geschrieben:

$$(A \vee B \vee \neg C) \wedge (B \vee D) \wedge (\neg A) \wedge (B \vee C)$$

- $(A \vee B \vee \neg C)$ ist eine Klausel: Disjunktion von Literale
- $A, B, \neg C$ sind Literale:

KNF

- Erfüllbarkeitsprobleme werden als KNF Formeln geschrieben:

$$(A \vee B \vee \neg C) \wedge (B \vee D) \wedge (\neg A) \wedge (B \vee C)$$

- $(A \vee B \vee \neg C)$ ist eine Klausel: Disjunktion von Literale
- $A, B, \neg C$ sind Literale: eine Konstante oder die Negation einer Konstanten
- Jede Klausel ist eine Anforderung die erfüllbar sein muss und es gibt mehrere Möglichkeiten diese Anforderung zu erfüllen.
- Jede Aussage der propositionalen Logik läßt sich in KNF bringen!

KNF Umformung

Vier Schritte:

1. Elimination von $\leftrightarrow$

   Verwende $A \leftrightarrow B \equiv (A \rightarrow B) \wedge (B \rightarrow A)$

2. Elimination von $\rightarrow$

   Verwende $A \rightarrow B \equiv (\neg A \vee B)$

3. "Nach innen schieben" von $\neg$

   Verwende de Morgans Regeln und $\neg\neg A \equiv A$ (NNF)

4. "Nach innen schieben" von $\vee$

   Verwende Distributivität von $\vee$ über $\wedge$

KNF Umformung

- Jede Aussage läßt sich in KNF umformen aber die Länge der Formel kann exponentiell wachsen!

KNF Vereinfachung

- Ziel ist Formeln in KNF zu vereinfachen.
- Manchmal landet man auf die sogenannte leere Klausel und leere Formel
- Was ist die Bedeutung dieser leeren Klauseln und leeren Formeln?

KNF Vereinfachung

- Eine leere Klausel ist falsch: Es gibt keine Bedingungen zu erfüllen. Im Allgemeinen, eine Disjunktion ohne Disjunktionsglieder ist falsch.
- Weshalb?

KNF Vereinfachung

- Eine leere Klausel ist falsch: Es gibt keine Bedingungen zu erfüllen. Im Allgemeinen, eine Disjunktion ohne Disjunktionsglieder ist falsch.
- Weshalb? Um eine Disjunktion wahr zu machen, muss man mindestens ein Glied wahr machen. Es gibt aber keine Glieder. Daher kann die leere Disjunktion nie wahr sein!
- Eine leere Aussage (ohne Klauseln) ist wahr: Keine Anforderungen...
- Weshalb?

KNF Vereinfachung

- Eine leere Klausel ist falsch: Es gibt keine Bedingungen zu erfüllen. Im Allgemeinen, eine Disjunktion ohne Disjunktionsglieder ist falsch.
- Weshalb? Um eine Disjunktion wahr zu machen, muss man mindestens ein Glied wahr machen. Es gibt aber keine Glieder. Daher kann die leere Disjunktion nie wahr sein!
- Eine leere Aussage (ohne Klauseln) ist wahr: Keine Anforderungen...
- Weshalb? Eine Konjunktion ohne Konjunktionsglieder ist wahr. Um diesen Ausdruck wahr zu machen, müssen wir alle Glieder wahr machen. Es gibt aber keine Glieder, d.h. der Ausdruck ist trivialerweise wahr.

KNF Vereinfachung

- Eine Aussage die die leere Klausel enthält ist falsch!
- Es gibt eine unmögliche Anforderung...
- Weshalb?

KNF Vereinfachung

- Eine Aussage die die leere Klausel enthält ist falsch!
- Es gibt eine unmögliche Anforderung...
- Weshalb? Die leere Klausel ist falsch, die Aussage ist eine Konjunktion von Klauseln, d.h. falsch

Erfüllbarkeitsalgorithmen

- Gegeben eine Aussage in KNF, wie können wir deren Erfüllbarkeit testen?

Erfüllbarkeitsalgorithmen

- Gegeben eine Aussage in KNF, wie können wir deren Erfüllbarkeit testen?
- Wir testen, ob es eine Belegung existiert, so dass die Aussage wahr ist.
- D.h. wir suchen eine solche Belegung/Modell.
- Erste Methode: Brute Force - Teste alle Möglichkeiten durch. Funktioniert, aber die Anzahl möglicher Belegungen wächst exponentiell zur Variablenanzahl.
- SCHNECKENLANGSAM!!!

Erfüllbarkeitsalgorithmen

- Wir basteln einen sogenannten Suchbaum
- Startpunkt ist eine Variable P und alle möglichen Belegungen von P: 0 oder 1, falsch oder wahr.
- Auf dem einen Ast nehmen wir an P sei falsch, auf dem anderen wahr.
- Die Belegung falsch vereinfacht unser Problem. Weshalb?

Erfüllbarkeitsalgorithmen

- Wir basteln einen sogenannten Suchbaum
- Startpunkt ist eine Variable P und alle möglichen Belegungen von P: 0 oder 1, falsch oder wahr.
- Auf dem einen Ast nehmen wir an P sei falsch, auf dem anderen wahr.
- Die Belegung falsch vereinfacht unser Problem. Weshalb?
- Wir suchen eine Belegung, so dass alle Variablen erfüllen eine Menge von Anforderungen. Falls P falsch ist, dann haben wir eine kürzere Menge von Anforderungen für unser Problem.

Erfüllbarkeitsalgorithmen

- Gegeben eine Belegung für eine Variable, können wir die Aussage vereinfachen und dann können wir den Vorgang mit einer anderen Variable wiederholen.

Beispiel

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

Beispiel

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

- Lasst uns jetzt P falsch machen. Was geht ab?

Beispiel

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

- Lasst uns jetzt P falsch machen. Was geht ab? Die Menge der Einschränkungen vereinfacht sich. Wie?

Beispiel

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

- Lasst uns jetzt P falsch machen. Was geht ab? Die Menge der Einschränkungen vereinfacht sich. Wie?
- $P \vee Q$ ist dann erfüllungsäquivalent zu Q. Weshalb?

Beispiel

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

- Lasst uns jetzt P falsch machen. Was geht ab? Die Menge der Einschränkungen vereinfacht sich. Wie?
- $P \vee Q$ ist dann erfüllungsäquivalent zu Q. Weshalb?
- $P \vee \neg Q \vee R$ wird zu $\neg Q \vee R$
- $\neg P \vee \neg T$ ist erfüllbar und wird weggelassen!
- $P \vee S$ wird S
- Wir erhalten

$$(Q) \wedge (\neg Q \vee R) \wedge (T \vee \neg R) \wedge (S) \wedge (T \vee R \vee S) \wedge (\neg S \vee T)$$

- Wir erhalten eine Aussage die P frei ist und ist einfacher als die ursprüngliche Aussage zu der sie erfüllungsäquivalent ist!

Belege und vereinfache Prozedur

- Sei ϕ eine Aussage und U ein Literal (d.h. eine Variable oder eine negierte Variable)
- Entferne alle Klauseln die U enthalten (diese Klauseln sind erfüllt)
- Entferne $\neg U$ aus allen anderen Klauseln
- Bezeichne die verinfachte Klausel mit $\phi(U)$
- Die Prozedur funktioniert sowohl mit positive als auch mit negative Literale

Beispiel: Suchbaum

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

Beispiel: Suchbaum

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

$\phi(\neg P)$ $\phi(P)$

$$(Q) \wedge (\neg Q \vee R) \wedge (T \vee \neg R)$$
$$\wedge (S) \wedge (T \vee R \vee S) \wedge (\neg S \vee T)$$

Beispiel: Suchbaum

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

$\phi(\neg P)$ $\phi(P)$

$$(Q) \wedge (\neg Q \vee R) \wedge (T \vee \neg R)$$
$$\wedge (S) \wedge (T \vee R \vee S) \wedge (\neg S \vee T)$$

$\phi(\neg Q)$ $\phi(Q)$

Beispiel: Suchbaum

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

$\phi(\neg P)$ $\qquad$ $\phi(P)$

$$(Q) \wedge (\neg Q \vee R) \wedge (T \vee \neg R)$$
$$\wedge (S) \wedge (T \vee R \vee S) \wedge (\neg S \vee T)$$

$\phi(\neg Q)$ $\qquad$ $\phi(Q)$

$$() \wedge (T \vee \neg R) \wedge (S)$$
$$\wedge (T \vee R \vee S) \wedge (\neg S \vee T)$$

Beispiel: Suchbaum

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

$\phi(\neg P)$ $\qquad$ $\phi(P)$

$$(Q) \wedge (\neg Q \vee R) \wedge (T \vee \neg R)$$
$$\wedge (S) \wedge (T \vee R \vee S) \wedge (\neg S \vee T)$$

$\phi(\neg Q)$ $\qquad$ $\phi(Q)$

$$() \wedge \cancel{(T \vee \neg R)} \wedge (S)$$
$$\wedge \cancel{(T \vee R \vee S)} \wedge (\neg S \vee T)$$

$$(R) \wedge (T \vee \neg R) \wedge (S)$$
$$\wedge (T \vee R \vee S) \wedge (\neg S \vee T)$$

Beispiel: Suchbaum

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

$\phi(\neg P)$ $\phi(P)$

$$(Q) \wedge (\neg Q \vee R) \wedge (T \vee \neg R)$$
$$\wedge (S) \wedge (T \vee R \vee S) \wedge (\neg S \vee T)$$

$\phi(\neg Q)$ $\phi(Q)$

$$() \wedge \cancel{(T \vee \neg R)} \wedge (S)$$
$$\wedge \cancel{(T \vee R \vee S)} \wedge (\neg S \vee T)$$

$$(R) \wedge (T \vee \neg R) \wedge (S)$$
$$\wedge (T \vee R \vee S) \wedge (\neg S \vee T)$$

$\phi(\neg R)$ $\phi(R)$

Beispiel: Suchbaum

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

$\phi(\neg P)$ $\phi(P)$

$$(Q) \wedge (\neg Q \vee R) \wedge (T \vee \neg R)$$
$$\wedge (S) \wedge (T \vee R \vee S) \wedge (\neg S \vee T)$$

$\phi(\neg Q)$ $\phi(Q)$

$$() \wedge \cancel{(T \vee \neg R)} \wedge (S)$$
$$\wedge \cancel{(T \vee R \vee S)} \wedge (\neg S \vee T)$$

$$(R) \wedge (T \vee \neg R) \wedge (S)$$
$$\wedge (T \vee R \vee S) \wedge (\neg S \vee T)$$

$\phi(\neg R)$ $\phi(R)$

$$() \wedge (S) \wedge \cancel{(T \vee S)} \wedge (\neg S \vee T)$$

Beispiel: Suchbaum

Beispiel: Suchbaum

Gibt es eine schlauere Möglichkeit?

Beispiel: Suchbaum

- JA!
- Falls ein Literal alleine in einer Klausel erscheint (s. Q, R, T, S) dann wählen wir folgende Belegung: wahr für positive Literale und falsch für negative Literale
- Die letzte Formel der Baums ergibt mit S wahr die Aussage $(T) \wedge (T)$.
- Die einzige Möglichkeit ist T wahr, d.h. die Formel ist wahr.
- Wir erhalten somit die Belegung P falsch, Q wahr, R wahr, S wahr, T wahr.
- Mit dieser Belegung ist die ursprüngliche Formel wahr.

Ein weiteres Beispiel

$$(T \vee X) \wedge (\neg S \vee T) \wedge (S \vee X)$$

- T kommt nur als positives Literal vor.
- Wir müssen nicht unbedingt T als wahr betrachten, schaden tut es auf jeden Fall nicht!
- Daraus lernen wir folgendes: Falls eine Variable als positives Literal vorkommt, wird sie als wahr betrachtet, andernfalls als falsch.
- Im Beispiel: T wahr $\Rightarrow$ alle Klauseln die T enthalten werden entfernt. Es bleibt nur $(S \vee X)$ übrig
- Also wahr.

DPLL Algorithmus

- Aus all dem bisher gelerntem basteln wir nun einen Algorithmus.
- DPLL Algorithmus: Davis, Putnam, Logeman und Loveland.
- Es gibt auch andere Algorithmen die die Erfüllbarkeit von Aussagen testen (satisfiability algorithms): GSAT, WalkSAT
- Der Algorithmus nimmt als Eingabe eine aussagenlogische Formeln und ergibt die Antwort wahr falls die Aussage erfüllbar ist und falsch andernfalls.
- Es basiert auf Rekursion.

DPLL Algorithmus

- If ϕ is empty, return true (embrace truth)

DPLL Algorithmus

- If ϕ is empty, return true (embrace truth)
- If there is an empty clause in ϕ, return false (reject falsity)
- `Bemerkung: Eine leere Klausel ist immer falsch, d.h. falls wir eine leere Klausel in der Aussage haben, muss die ganze Aussage falsch sein.`

DPLL Algorithmus

- If ϕ is empty, return true (embrace truth)
- If there is an empty clause in ϕ, return false (reject falsity)
- `Bemerkung: Eine leere Klausel ist immer falsch, d.h. falls wir eine leere Klausel in der Aussage haben, muss die ganze Aussage falsch sein.`
- If there is a unit clause U in ϕ, return DPLL($\phi(U)$) (accept the inevitable)
- `Bemerkung: Einheitsklauseln besitzen nur ein Literal!` Dieses Literal wird automatisch belegt mit wahr oder falsch, die Aussage wird vereinfacht und DPLL wird wieder aufgerufen für die vereinfachte Aussage.

DPLL Algorithmus

- If there is a pure literal U in ϕ, return DPLL($\phi(U)$) (go with the flow)
- `Bemerkung: Reine Literale kommen ausschließlich positiv oder negativ in` ϕ `vor`. Ein reines Literal wird mit wahr oder falsch belegt, die Formel wird vereinfacht und DPLL wird erneut aufgerufen (Rekursiver Aufruf!)

DPLL Algorithmus

- If there is a pure literal U in ϕ, return DPLL($\phi(U)$)
  (go with the flow)
- `Bemerkung: Reine Literale kommen ausschließlich positiv oder negativ in` ϕ `vor`. Ein reines Literal wird mit wahr oder falsch belegt, die Formel wird vereinfacht und DPLL wird erneut aufgerufen (Rekursiver Aufruf!)
- For some variable V
  (take a guess)
- `Bemerkung: Falls keine der vorherigen Bedingungen erfüllt wird, wählen wir zufällig irgendeine Variable.`

DPLL Algorithmus

- If DPLL($\phi(V)$) then return true
- `Bemerkung: Belege` V `mit wahr`. Vereinfache und wende erneut DPLL an. Falls das Ergebnis wahr ist, dann ist auch die ursprüngliche Aussage wahr.

DPLL Algorithmus

- If DPLL($\phi(V)$) then return true
- `Bemerkung: Belege` V `mit wahr`. Vereinfache und wende erneut DPLL an. Falls das Ergebnis wahr ist, dann ist auch die ursprüngliche Aussage wahr.
- Else return DPLL($\phi(\neg V)$).
- `Bemerkung: Falls nicht, dann belege` V `mit falsch`. Vereinfache und wende erneut DPLL rekursiv an.

Wie findet man schnell die GUTEN Variablen?

- MOMS Heuristik
- Maximum number of Occurrences, Minimum Sized clauses
- Wähle die Variable die am häufigsten erscheint in Klauseln minimaler Länge.
- Wähle stark eingeschränkte Variablen. Falls es schief gehen sollte, dann bitte schön so früh wie nur möglich! $\Rightarrow$ Keine Riesenbäume, wenn möglich
- Intuitiv: Belegen der stark eingeschränkten Variablen kann uns schnell die Lösung liefern.

Korrektheit und Vollständigkeit

- Falls ein Algorithmus korrekt ist, dann ist jede Antwort eine korrekte Antwort
- Vollständigkeit: Wir finden immer eine Antwort
- DPLL ist korrekt und vollständig
- DPLL ist ein systematischer Algorithmus. Er überspringt nur diejenigen Belegungen die garantiert zur Unerfüllbarkeit führen.
- Deshalb kann es manchmal langsam sein...
- Deshalb ist es zu empfehlen auf korrekte aber nicht vollständige Algorithmen zu greifen: Falls diese Algorithmen eine Antwort liefern ist sie korrekt. ABER eine richtige Antwort kann nicht immer gefunden werden.
- Diese Algorithmen sind schneller als vollständige Algorithmen

Logik für Informatiker

Vorlesung 5: SAT-Algorithmen und Hornformeln

Babeş-Bolyai Universität, Department für Informatik, Cluj-Napoca
csacarea@cs.ubbcluj.ro

Wiederholung: DPLL-Algorithmus

- If ϕ is empty, return true (embrace truth)

Wiederholung: DPLL-Algorithmus

- If ϕ is empty, return true (embrace truth)
- If there is an empty clause in ϕ, return false (reject falsity)
- `Bemerkung: Eine leere Klausel ist immer falsch, d.h. falls wir eine leere Klausel in der Aussage haben, muss die ganze Aussage falsch sein.`

Wiederholung: DPLL-Algorithmus

- If ϕ is empty, return true (embrace truth)
- If there is an empty clause in ϕ, return false (reject falsity)
- `Bemerkung: Eine leere Klausel ist immer falsch, d.h. falls wir eine leere Klausel in der Aussage haben, muss die ganze Aussage falsch sein.`
- If there is a unit clause U in ϕ, return DPLL($\phi(U)$) (accept the inevitable)
- `Bemerkung: Einheitsklauseln besitzen nur ein Literal!` Dieses Literal wird automatisch belegt mit wahr oder falsch, die Aussage wird vereinfacht und DPLL wird wieder aufgerufen für die vereinfachte Aussage.

DPLL Algorithmus

- If there is a pure literal U in ϕ, return DPLL($\phi(U)$) (go with the flow)
- `Bemerkung: Reine Literale kommen ausschließlich positiv oder negativ in` ϕ `vor`. Ein reines Literal wird mit wahr oder falsch belegt, die Formel wird vereinfacht und DPLL wird erneut aufgerufen (Rekursiver Aufruf!)

DPLL Algorithmus

- If there is a pure literal U in ϕ, return DPLL($\phi(U)$)
  (go with the flow)
- `Bemerkung: Reine Literale kommen ausschließlich positiv oder negativ in` ϕ `vor`. Ein reines Literal wird mit wahr oder falsch belegt, die Formel wird vereinfacht und DPLL wird erneut aufgerufen (Rekursiver Aufruf!)
- For some variable V
  (take a guess)
- `Bemerkung: Falls keine der vorherigen Bedingungen erfüllt wird, wählen wir zufällig irgendeine Variable.`

DPLL Algorithmus

- If DPLL($\phi(V)$) then return true
- `Bemerkung: Belege` V `mit wahr`. Vereinfache und wende erneut DPLL an. Falls das Ergebnis wahr ist, dann ist auch die ursprüngliche Aussage wahr.

DPLL Algorithmus

- If DPLL($\phi(V)$) then return true
- `Bemerkung: Belege` V `mit wahr`. Vereinfache und wende erneut DPLL an. Falls das Ergebnis wahr ist, dann ist auch die ursprüngliche Aussage wahr.
- Else return DPLL($\phi(\neg V)$).
- `Bemerkung: Falls nicht, dann belege` V `mit falsch`. Vereinfache und wende erneut DPLL rekursiv an.

Systematische Suche: DPLL

1. (embrace truth)
2. (reject falsity)
3. (accept the inevitable)
4. (go with the flow)
5. (take a guess)

MOMS Heuristik

DPLL Algorithmus revisited

- Der DPLL-Algorithmus (Davis/Putnam/Logemann/Loveland) entspricht Backtracking mit Inferenz.
- rekursiver Aufruf `DPLL(`Δ, I`)` für Klauselmenge Δ und partielle Belegung I
- Ergebnis ist erfüllende Belegung, die I erweitert; unsatisfiable, wenn keine solche Belegung existiert
- oberster Aufruf als `DPLL(`$\Delta, \emptyset$`)`

Inferenz in DPLL:

- simplify: nachdem der Variablen v der Wert d zugewiesen wird, werden alle Klauseln vereinfacht, die über v sprechen
- Unit Propagation: Variablen, die in Klauseln ohne weitere Variablen (Einheitsklauseln) auftreten, werden sofort belegt

DPLL Algorithmus revisited

Verifikation

Algorithmus **DPLL-Prozedur**

Funktion DPLL($\mathcal{C}$):

if $\emptyset \in \mathcal{C}$ **then** **return true**; *// unerfüllbar*

if $\mathcal{C} = \emptyset$ **then** **return false**; *// erfüllbar*

if $\mathcal{C}$ enthält 1-Klausel $\{L\}$ **then**

Sei $\mathcal{C}'$ die Klauselmenge, die aus $\mathcal{C}$ entsteht, indem

(1) alle Klauseln, die L enthalten, entfernt werden und

(2) in den verbleibenden Klausel alle Literale $\overline{L}$ entfernt werden

(wobei $\overline{L} = \neg X$, wenn $L = X$ und $\overline{L} = X$, wenn $L = \neg X$)

DPLL($\mathcal{C}'$); *// rekursiver Aufruf*

if $\mathcal{C}$ enthält isoliertes Literal L **then**

Sei $\mathcal{C}'$ die Klauselmenge, die aus $\mathcal{C}$ entsteht, indem alle Klauseln, die L enthalten, entfernt werden.

DPLL($\mathcal{C}'$); *// rekursiver Aufruf*

// Nur wenn keiner der obigen Fälle zutraf:

Wähle Atom A, dass in C vorkommt;

return DPLL($\mathcal{C} \cup \{\{A\}\}$) $\wedge$ DPLL($\mathcal{C} \cup \{\{\neg A\}\}$) *// Fallunterscheidung*

DPLL Algorithmus revisited

Verifikation

if $\mathcal{C}$ enthält 1-Klausel $\{L\}$ **then**
Sei $\mathcal{C}'$ die Klauselmenge, die aus $\mathcal{C}$ entsteht, indem
(1) alle Klauseln, die L enthalten, entfernt werden und
(2) in den verbleibenden Klausel alle Literale $\overline{L}$ entfernt werden
(wobei $\overline{L} = \neg X$, wenn $L = X$ und $\overline{L} = X$, wenn $L = \neg X$)

- (1) entspricht der Subsumtion:

$$\{A\}, \ldots, \{A, B, C\}, \ldots$$

entferne $\{A, B, C\}$, da $\{A\}$ die Klausel $\{A, B, C\}$ subsumiert.

- (2) entspricht der Resolution mit anschließender Subsumtion

$$\begin{array}{ccc} \{L\} & & \{\overline{L}, L'_1, \ldots, L'_n\} \\ & \diagdown \quad \diagup & \\ & \{L'_1, \ldots, L'_n\} & \end{array}$$

- Entfernen von $\{A\}$: „Setze $I(A) = 1$“, sonst sowieso nicht erfüllbar

DPLL Algorithmus revisited

Verifikation

if $\mathcal{C}$ enthält isoliertes Literal L **then**
Sei $\mathcal{C}'$ die Klauselmenge, die aus $\mathcal{C}$ entsteht, indem alle Klauseln, die L enthalten, entfernt werden.
DPLL($\mathcal{C}'$); *// rekursiver Aufruf*

Entspricht der Löschregel für isolierte Literale

// Nur wenn keiner der obigen Fälle zutraf:
Wähle Atom A, dass in C vorkommt;
return DPLL($\mathcal{C} \cup \{\{A\}\}$) $\wedge$ DPLL($\mathcal{C} \cup \{\{\neg A\}\}$) *// Fallunterscheidung*

Fallunterscheidung, ob A wahr oder falsch ist.

Die Suche nach dem Kuchendieb

Ein Logik-Krimi

Die Suche nach dem Kuchendieb

- Omi Leckerschmatz backt am Sonntag einen Kuchen.
  Nachdem der Kuchen fertig ist, wird er zum Ausruhen auf die Fensterbank gestellt.
  Die drei Bengel Hatzmatz, Schnappmäulchen und Mampf-Schlingel spielen im Hof.
  Als Omi vom Gottesdienst zurückkommt bemerkt sie, dass der Kuchen geklaut worden ist.

Die Suche nach dem Kuchendieb

- Omi Leckerschmatz backt am Sonntag einen Kuchen.
  Nachdem der Kuchen fertig ist, wird er zum Ausruhen auf die Fensterbank gestellt.
  Die drei Bengel Hatzmatz, Schnappmäulchen und Mampf-Schlingel spielen im Hof.
  Als Omi vom Gottesdienst zurückkommt bemerkt sie, dass der Kuchen geklaut worden ist.
- Die drei Verdächtige sind natürlich unsere drei Bengel.

Die Suche nach dem Kuchendieb

- Omi Leckerschmatz backt am Sonntag einen Kuchen.
  Nachdem der Kuchen fertig ist, wird er zum Ausruhen auf die Fensterbank gestellt.
  Die drei Bengel Hatzmatz, Schnappmäulchen und Mampf-Schlingel spielen im Hof.
  Als Omi vom Gottesdienst zurückkommt bemerkt sie, dass der Kuchen geklaut worden ist.
- Die drei Verdächtige sind natürlich unsere drei Bengel.
- Genau einer von ihnen ist der Dieb.

Die Suche nach dem Kuchendieb

- Omi Leckerschmatz backt am Sonntag einen Kuchen.
  Nachdem der Kuchen fertig ist, wird er zum Ausruhen auf die Fensterbank gestellt.
  Die drei Bengel Hatzmatz, Schnappmäulchen und Mampf-Schlingel spielen im Hof.
  Als Omi vom Gottesdienst zurückkommt bemerkt sie, dass der Kuchen geklaut worden ist.
- Die drei Verdächtige sind natürlich unsere drei Bengel.
- Genau einer von ihnen ist der Dieb.
- Wer ist der Dieb falls wir folgendes wissen:
  - Unschuldige sagen immer die Wahrheit.
  - Schnappmäulchen: *Hatzmatz ist unschuldig*.
  - Hatzmatz: *Mampf-Schlingel ist unschuldig*.

Die Suche nach dem Kuchendieb

Modellierung

- Bezeichne Hatzmatz, Schnappmäulchen und Mampf-Schlingel mit H, S, M. Die Variablen H, S, M stehen dann für *ist schuldig*.
- $(H \vee S \vee M) \wedge (H \rightarrow \neg(S \vee M)) \wedge (S \rightarrow \neg(H \vee M)) \wedge (M \rightarrow \neg(H \vee S))$ (Genau einer von ihnen ist der Dieb.)
- $\neg S \rightarrow \neg H$ (Schnappmäulchen: *Hatzmatz ist unschuldig*.)
- $\neg H \rightarrow \neg M$ (Hatzmatz: *Mampf-Schlingel ist unschuldig*.)

Die Suche nach dem Kuchendieb

DPLL anwenden

Wissen:	Klauselmenge dazu:
$H \vee S \vee M$	$\{H, S, M\}$
$H \rightarrow \neg(S \vee M)$	$\{\neg H, \neg S\}$
$S \rightarrow \neg(H \vee M)$	$\{\neg H, \neg M\}$
$M \rightarrow \neg(H \vee S)$	$\{\neg S, \neg M\}$
$\neg S \rightarrow \neg H$	$\{S, \neg H\}$
$\neg H \rightarrow \neg M$	$\{H, \neg M\}$

Die Suche nach dem Kuchendieb

DPLL anwenden

DPLL$(\{\{H, S, M\}, \{\neg H, \neg S\}, \{\neg H, \neg M\}, \{\neg S, \neg M\}, \{S, \neg H\}, \{H, \neg M\}\})$

- Keine 1-Klauseln
- Keine isolierten Literale
- Daher Fallunterscheidung

(1) DPLL$(\{\{S\}, \{H, S, M\}, \{\neg H, \neg S\}, \{\neg H, \neg M\}, \{\neg S, \neg M\}, \{S, \neg H\}, \{H, \neg M\}\})$

$\wedge$

(2) DPLL$(\{\{\neg S\}, \{H, S, M\}, \{\neg H, \neg S\}, \{\neg H, \neg M\}, \{\neg S, \neg M\}, \{S, \neg H\}, \{H, \neg M\}\})$

Die Suche nach dem Kuchendieb

DPLL anwenden

Fall (1):
DPLL($\{\{S\},\{H,S,M\},\{\neg H,\neg S\},\{\neg H,\neg M\},\{\neg S,\neg M\},\{S,\neg H\},\{H,\neg M\}\}$)

- 1-Klausel $\{S\}$
- entferne alle Klauseln die $\{S\}$ enthalten:
  $\{\underline{\{S\}},\underline{\{H,S,M\}},\{\neg H,\neg S\},\{\neg H,\neg M\},\{\neg S,\neg M\},\underline{\{S,\neg H\}},\{H,\neg M\}\}$

  ergibt $\{\{\neg H,\neg S\},\{\neg H,\neg M\},\{\neg S,\neg M\},\{H,\neg M\}\}$
  entferne $\neg S$ aus allen Klauseln: $\{\{\neg H,\underline{\neg S}\},\{\neg H,\neg M\},\{\underline{\neg S},\neg M\},\{H,\neg M\}\}$

  ergibt $(\{\{\neg H\},\{\neg H,\neg M\},\{\neg M\},\{H,\neg M\}\})$
  Rekursiver Aufruf: DPLL($\{\{\neg H\},\{\neg H,\neg M\},\{\neg M\},\{H,\neg M\}\}$)

DPLL($\{\{\neg H\},\{\neg H,\neg M\},\{\neg M\},\{H,\neg M\}\}$)

- 1-Klausel $\{\neg H\}$
- Entfernen der Klauseln mit $\neg H$ und Löschen von H ergibt $\{\{\neg M\}\}$
  Rekursiver Aufruf: DPLL($\{\{\neg M\}\}$)

DPLL($\{\{\neg M\}\}$) ergibt rekursiver Aufruf: DPLL($\{\}$)
DPLL($\{\}$) ergibt `false` $\rightarrow$ erfüllbar, Modell?

DPLL Modell generieren

1. Isolierte Literale werden als wahr angenommen.
2. Literale in 1-Klauseln werden ebenfalls als wahr angenommen.
3. Dadurch nicht belegte Variablen können für das vollständige Modell beliebig belegt werden

Komplexität

Im Worst Case:

- Exponentielle Laufzeit
- Exponentiell in: Anzahl der verschiedenen Variablen

Besser geht es vermutlich nicht, da SAT NP-vollständig

DPLL zum Problemlösen

- Idee: Kodiere Problem als Aussagenlogische Formel
- Sodass: Modell löst Problem
- Beachte: Erst Umformung in KNF (geht schneller mit KNF), dann DPLL
- Oft: Kodierung direkt als KNF
- Im Folgenden: Wir betrachten einige Beispielanwendungen.

Logelei

DPLL anwenden

Abianer sagen die Wahrheit, Bebianer lügen

1. Knasi: Knisi ist Abianer. $knasi \leftrightarrow knisi$
2. Knesi: Wenn Knösi Bebianer, dann ist auch Knusi ein Abianer. $knesi \leftrightarrow (\neg knoesi \rightarrow knusi)$
3. Knisi: Wenn Knusi Abianer, dann ist Knesi Bebianer. $knisi \leftrightarrow (knusi \rightarrow \neg knesi)$
4. Knosi: Knesi und Knüsi sind beide Abianer. $knosi \leftrightarrow (knesi \wedge knuesi)$
5. Knusi: Wenn Knüsi Abianer ist, dann ist auch Knisi Abianer. $knusi \leftrightarrow (knuesi \rightarrow knisi)$
6. Knösi: Entweder ist Knasi oder Knisi Abianer. $knusi \leftrightarrow (knasi XOR knisi)$
7. Knüsi: Knosi ist Abianer. $knuesi \leftrightarrow knosi$

Logelei

DPLL anwenden

- Wie wird das Problem gelöst?

Logelei

DPLL anwenden

- Wie wird das Problem gelöst?
- Für die Klauselmenge existiert ein Modell:
  Knesi und Knusi sind Abianer, Knüsi, Knosi, Knösi, Knisi sind Bebianer.

GSAT

- Ein Bergsteiger Algorithmus im Raum aller Belegungen

GSAT

- Ein Bergsteiger Algorithmus im Raum aller Belegungen
- Starte mit einer zufälligen Belegung aller Variablen
- Wir definieren die Nachbarbelegungen als diejenigen Belegungen die sich nur auf ein Bit von der initialen Belegung unterscheiden.
- Benutze die Nachbarbelegung mit minimalen Kosten
- Kosten =

GSAT

- Ein Bergsteiger Algorithmus im Raum aller Belegungen
- Starte mit einer zufälligen Belegung aller Variablen
- Wir definieren die Nachbarbelegungen als diejenigen Belegungen die sich nur auf ein Bit von der initialen Belegung unterscheiden.
- Benutze die Nachbarbelegung mit minimalen Kosten
- Kosten = Anzahl unerfüllter Klauseln in der jeweiligen Belegung

GSAT

- `Loop n times`
- % Wir durchführen n verschiedene Bergsteige
- `Wähle zufällig die Belegung` $\mathcal{A}$.
- % Wir starten mit verschiedene initiale Belegungen
- `Loop m times`
- % Wir betrachten die Kosten aller Nachbarbelegungen

GSAT

- `Flip the variable that results in lowest cost`
- `Exit if cost is zero`
- % Kosten = 0 $\Rightarrow$ wir haben eine erfüllende Belegung gefunden. EXIT

GSAT vs DPLL

- GSAT is korrekt
- GSAT ist nicht vollständig
- Kann nicht benutzt werden um alle erfüllende Belegungen zu finden
- In SAT Wettbewerbe war es für eine Weile schneller aber zur Zeit werden die Heuristiken von DPLL verbessert und DPLL wird wieder schneller
- Schwach beschränkte Aufgaben sind einfach für DPLL und GSAT (solche Aufgaben haben viele Lösungen)
- Stark beschränkte Aufgaben sind einfach für DPLL aber schwer für GSAT. Solche Aufgaben haben eine oder nur sehr wenige Lösungen. Sie sind einfacher für DPLL weil die Vereinfachung der Formel schnell zu allen Lösungen führt. Für GSAT aber ist das Finden einer Lösung sehr schwierig, da es nicht weiß wo sich diese Lösung versteckt hat.

GSAT vs DPLL

- Aufgaben die nicht schwach und auch nicht stark beschränkt sind, sind schwer für beide Algorithmen, eigentlich für alle bekannte Algorithmen...

WALKSAT

GSAT mit noise (noise = Zufall)

- `Loop n times`
- `Wähle zufällig Belegung` $\mathcal{A}$
- % Hat dieselbe Aussentruktur wie GSAT. Es gibt eine äußere Schleife von n Iterationen auf verschiedene zufällig gewählte Belegungen.
- `Loop` m `times`
- % Diesmal werden die Schritte verschieden sein.

WALKSAT

- `Randomly select unsatisfied clause` C
- % Wähle zufällig eine unerfüllte Klausel C. Dann werfen wir eine Münze. Die Wahrscheinlichkeit ist $p = 0.5$.
- Mit der Wahrscheinlichkeit $p = 0.5$ entweder verändern wir den Wahrheitswert der Variablen in der Klausel C die den kleinsten Kosten ergibt oder
- Wähle zufällig eine Variable in C und verändere deren Wahrheitswert.
- % Der Grund dafür ist, dass es manchmal ratsam ist seine Situation zu verschlechtern, so dass es später viel besser sein kann. Mehr darüber in der KI Vorlesung!
- EXIT falls die Kosten Null sind.

Horn-Formeln

Horn-Formel: Formel in KNF, in der jede Klausel höchstens ein positives Literal enthält.

Notation: als Implikation

$\neg P_1 \vee \cdots \vee \neg P_n \vee P$	$P_1 \wedge \cdots \wedge P_n \rightarrow P$	$P_1, \ldots, P_n \rightarrow P$
$\neg P_1 \vee \cdots \vee \neg P_n$	$P_1 \wedge \cdots \wedge P_n \rightarrow \bot$	$P_1, \ldots, P_n \rightarrow$
P	$\top \rightarrow P$	$\rightarrow P$

$P_1 \wedge \cdots \wedge P_n$: Rumpf

P: Kopf $\qquad\qquad$ $\rightarrow P$: Fakt

Horn-Formeln: Beispiele

Klausel	Literalmengen	Implikationen
$\neg P$	$\{\neg P\}$	$P \rightarrow$
$Q \vee \neg R \vee \neg S$	$\{Q, \neg R, \neg S\}$	$R, S \rightarrow Q$
$\neg Q \vee \neg S$	$\{\neg Q, \neg S\}$	$Q, S \rightarrow$
R	$\{R\}$	$\rightarrow R$
$\neg Q \vee P$	$\{\neg Q, P\}$	$Q \rightarrow P$

Erfüllbarkeitsproblem für Horn-Formeln

Theorem

Die Erfüllbarkeit von Horn-Formeln ist in quadratischer Zeit entscheidbar.

Lemma.

Sei F eine Horn-Formel die keine Fakten enthält. Dann ist F erfüllbar.

Beweis: Sei $\mathcal{A}\colon \Pi \to \{0,1\}$ mit $\mathcal{A}(P) = 0$ für alle $P \in \Pi$. Dann ist $\mathcal{A}(F) = 1$.

Erfüllbarkeitsproblem für Horn-Formeln

Theorem

Die Erfüllbarkeit von Horn-Formeln ist in quadratischer Zeit entscheidbar.

Beweis: (Idee)

Ziel: $\mathcal{A} : \Pi \to \{0, 1\}$ mit $\mathcal{A}(F) = 1$.

Falls keine Fakten in F: F erfüllbar.

Sonst: Für alle Fakten $\to P$ in F: $\mathcal{A}(P) := 1$;

Wiederhole das Verfahren für F', entstanden aus F durch Ersetzung von P mit $\top$.

Erfüllbarkeitstest für Horn-Formeln

Eingabe: $F = D_1 \wedge \cdots \wedge D_n$ eine Hornformel

(die Klausel D_i enthält höchstens ein positives Literal)

Ein Atom in F zu markieren, bedeutet, es an allen Stellen seines Auftretens in F zu markieren

Erfüllbarkeitstest für Horn-Formeln

0: IF keine Fakten (Klausel "$\rightarrow A$") vorhanden
  THEN Ausgabe: erfüllbar
  ELSE markiere alle Fakten in F (Atome A mit $\rightarrow A$ in F)

1: IF keine Klausel $A_1 \wedge \cdots \wedge A_n \rightarrow B$ in F, so dass
    alle Atome in $A_1, \ldots, A_n$ markiert aber B nicht
  THEN Ausgabe: erfüllbar
  ELSE wähle die erste solche Klausel
  IF B leer
    THEN Ausgabe: unerfüllbar
    ELSE markiere überall B in F
  GOTO 1

Beispiel 1

$$(\neg P) \wedge (Q) \wedge (\neg P \vee R) \wedge (\neg Q \vee S) \wedge (T \vee \neg W) \wedge (\neg S \vee U) \wedge$$
$$(\neg U \vee \neg T \vee P \vee \neg Z) \wedge (\neg Q \vee \neg S \vee \neg U \vee W)$$

Konjunktion von Implikationen:

$$\begin{array}{rcr}
(P \to \bot) & \wedge & P \to \\
(\top \to Q) & \wedge & \to Q \\
(P \to R) & \wedge & P \to R \\
(Q \to S) & \wedge & Q \to S \\
(W \to T) & \wedge & W \to T \\
(S \to U) & \wedge & S \to U \\
((U \wedge T \wedge Z) \to P) & \wedge & U, T, Z \to P \\
((Q \wedge S \wedge U) \to W) & & Q, S, U \to W
\end{array}$$

Beispiel 1

$$(\neg P) \wedge (Q) \wedge (\neg P \vee R) \wedge (\neg Q \vee S) \wedge (T \vee \neg W) \wedge (\neg S \vee U) \wedge$$
$$(\neg U \vee \neg T \vee P \vee \neg Z) \wedge (\neg Q \vee \neg S \vee \neg U \vee W)$$

$$\begin{aligned} P &\rightarrow \\ &\rightarrow \underline{Q} \\ P &\rightarrow R \\ \underline{Q} &\rightarrow S \\ W &\rightarrow T \\ S &\rightarrow U \\ U, T, Z &\rightarrow P \\ \underline{Q}, S, U &\rightarrow W \end{aligned}$$

Markierte Atome und Erklärung:

$\{Q\}$ initialer Fakt wegen $\rightarrow Q$

Beispiel 1

$$(\neg P) \wedge (Q) \wedge (\neg P \vee R) \wedge (\neg Q \vee S) \wedge (T \vee \neg W) \wedge (\neg S \vee U) \wedge$$
$$(\neg U \vee \neg T \vee P \vee \neg Z) \wedge (\neg Q \vee \neg S \vee \neg U \vee W)$$

$$\begin{aligned} P &\rightarrow \\ \rightarrow \underline{Q} \\ P &\rightarrow R \\ \underline{Q} &\rightarrow \underline{S} \\ W &\rightarrow T \\ \underline{S} &\rightarrow U \\ U, T, Z &\rightarrow P \\ \underline{Q}, \underline{S}, U &\rightarrow W \end{aligned}$$

Markierte Atome und Erklärung:

$\{Q\}$ initialer Fakt wegen $\rightarrow Q$

$\{Q, S\}$ wegen $Q \rightarrow S$

Beispiel 1

$$(\neg P) \wedge (Q) \wedge (\neg P \vee R) \wedge (\neg Q \vee S) \wedge (T \vee \neg W) \wedge (\neg S \vee U) \wedge$$
$$(\neg U \vee \neg T \vee P \vee \neg Z) \wedge (\neg Q \vee \neg S \vee \neg U \vee W)$$

$$P \rightarrow$$
$$\rightarrow \underline{Q}$$
$$P \rightarrow R$$
$$\underline{Q} \rightarrow \underline{S}$$
$$W \rightarrow T$$
$$\underline{S} \rightarrow \underline{U}$$
$$\underline{U}, T, Z \rightarrow P$$
$$\underline{Q}, \underline{S}, \underline{U} \rightarrow W$$

Markierte Atome und Erklärung:

$\{Q\}$ initialer Fakt wegen $\top \rightarrow Q$

$\{Q, S\}$ wegen $Q \rightarrow S$

$\{Q, S, U\}$ wegen $S \rightarrow U$

Beispiel 1

$$(\neg P) \wedge (Q) \wedge (\neg P \vee R) \wedge (\neg Q \vee S) \wedge (T \vee \neg W) \wedge (\neg S \vee U) \wedge$$
$$(\neg U \vee \neg T \vee P \vee \neg Z) \wedge (\neg Q \vee \neg S \vee \neg U \vee W)$$

$$\begin{aligned}
P &\rightarrow \\
&\rightarrow \underline{Q} \\
P &\rightarrow R \\
\underline{Q} &\rightarrow \underline{S} \\
\underline{W} &\rightarrow T \\
\underline{S} &\rightarrow \underline{U} \\
\underline{U}, T, Z &\rightarrow P \\
\underline{Q}, \underline{S}, \underline{U} &\rightarrow \underline{W}
\end{aligned}$$

Markierte Atome und Erklärung:

$\{Q\}$ initialer Fakt wegen $\top \rightarrow Q$

$\{Q, S\}$ wegen $Q \rightarrow S$

$\{Q, S, U\}$ wegen $S \rightarrow U$

$\{Q, S, U, W\}$ wegen $Q, S, U \rightarrow W$

Beispiel 1

$$(\neg P) \wedge (Q) \wedge (\neg P \vee R) \wedge (\neg Q \vee S) \wedge (T \vee \neg W) \wedge (\neg S \vee U) \wedge$$
$$(\neg U \vee \neg T \vee P \vee \neg Z) \wedge (\neg Q \vee \neg S \vee \neg U \vee W)$$

$$P \rightarrow$$
$$\rightarrow \underline{Q}$$
$$P \rightarrow R$$
$$\underline{Q} \rightarrow \underline{S}$$
$$\underline{W} \rightarrow \underline{T}$$
$$\underline{S} \rightarrow \underline{U}$$
$$\underline{U}, \underline{T}, Z \rightarrow P$$
$$\underline{Q}, \underline{S}, \underline{U} \rightarrow \underline{W}$$

Markierte Atome und Erklärung:

$\{Q\}$ initialer Fakt wegen $\top \rightarrow Q$

$\{Q, S\}$ wegen $Q \rightarrow S$

$\{Q, S, U\}$ wegen $S \rightarrow U$

$\{Q, S, U, W\}$ wegen $Q, S, U \rightarrow W$

$\{Q, S, U, W, T\}$ wegen $W \rightarrow T$

Beispiel 1

$(\neg P) \wedge (Q) \wedge (\neg P \vee R) \wedge (\neg Q \vee S) \wedge (T \vee \neg W) \wedge (\neg S \vee U) \wedge$
$(\neg U \vee \neg T \vee P \vee \neg Z) \wedge (\neg Q \vee \neg S \vee \neg U \vee W)$ Erfüllbar

$$
\begin{aligned}
P &\rightarrow \\
\rightarrow &\ \underline{Q} \\
P &\rightarrow R \\
\underline{Q} &\rightarrow \underline{S} \\
\underline{W} &\rightarrow \underline{T} \\
\underline{S} &\rightarrow \underline{U} \\
\underline{U}, \underline{T}, Z &\rightarrow P \\
\underline{Q}, \underline{S}, \underline{U} &\rightarrow \underline{W}
\end{aligned}
$$

Markierte Atome und Erklärung:

$\{Q\}$ initialer Fakt wegen $\top \rightarrow Q$

$\{Q, S\}$ wegen $Q \rightarrow S$

$\{Q, S, U\}$ wegen $S \rightarrow U$

$\{Q, S, U, W\}$ wegen $Q, S, U \rightarrow W$

$\{Q, S, U, W, T\}$ wegen $W \rightarrow T$

Keine weiteren Schritte möglich, da es keine Implikation gibt,deren linke Seite vollständig markiert ist und die rechte Seite nicht

Modell: $\mathcal{A}(Q) = \mathcal{A}(S) = \mathcal{A}(U) = \mathcal{A}(W) = \mathcal{A}(T) = 1, \ \mathcal{A}(P) = \mathcal{A}(R) = \mathcal{A}(Z) = 0$

Markierte Atome: wahr; nicht markierte Atome: falsch.

Beispiel 2

$$(\neg P) \wedge (\neg P \vee R) \wedge (\neg Q \vee S) \wedge (T \vee \neg W) \wedge (\neg S \vee U) \wedge$$
$$(\neg U \vee \neg T \vee P \vee \neg Z) \wedge (\neg Q \vee \neg S \vee \neg U \vee W)$$

Erfüllbar

$$P \rightarrow$$
$$P \rightarrow R$$
$$Q \rightarrow S$$
$$W \rightarrow T$$
$$S \rightarrow U$$
$$U, T, Z \rightarrow P$$
$$Q, S, U \rightarrow W$$

Markierte Atome und Erklärung:

Keine Schritte möglich, da es keine Implikation gibt,deren linke Seite vollständig markiert ist und die rechte Seite nicht

Modell: $\mathcal{A}(P) = \mathcal{A}(Q) = \mathcal{A}(R) = \mathcal{A}(S) = \mathcal{A}(T) = \mathcal{A}(U) = \mathcal{A}(W) = \mathcal{A}(Z) = 0$

Beispiel 3

$$(\neg P) \wedge (Q) \wedge (\neg P \vee R) \wedge (\neg Q \vee S) \wedge (T \vee \neg W) \wedge (\neg S \vee U) \wedge$$
$$(\neg U \vee \neg T \vee P \vee \neg Z) \wedge (\neg Q \vee \neg S \vee \neg U)$$

$$\begin{aligned}
P &\rightarrow \\
\rightarrow &\ \underline{Q} \\
P &\rightarrow R \\
\underline{Q} &\rightarrow \underline{S} \\
W &\rightarrow T \\
\underline{S} &\rightarrow \underline{U} \\
\underline{U}, T, Z &\rightarrow P \\
\underline{Q}, \underline{S}, \underline{U} &\rightarrow
\end{aligned}$$

Markierte Atome und Erklärung:

$\{Q\}$ initialer Fakt wegen $\top \rightarrow Q$

$\{Q, S\}$ wegen $Q \rightarrow S$

$\{Q, S, U\}$ wegen $S \rightarrow U$

Unerfüllbar

Q, S, U markiert, aber Kopf von

$Q, S, U \rightarrow$ leer

Logik für Informatiker

Vorlesung 6: Modellierung

Babeş-Bolyai Universität, Department für Informatik, Cluj-Napoca
csacarea@cs.ubbcluj.ro

Motivation für Aussagenlogik

- Aussagenlogik erlaubt Repräsentation von Wissen und Schlussfolgerungen auf Grundlage dieses Wissens
- viele Anwendungsprobleme direkt kodierbar:
  - Constraint-Satisfaction-Probleme aller Art
  - Schaltkreisentwurf und -verifikation
- viele Probleme verwenden Logik als einen Bestandteil:
  - Handlungsplanung
  - General Game Playing
  - Beschreibungslogik-Anfragen (Semantic Web)

Algorithmische Fragestellungen

Wesentliche Fragestellungen:

- Schlussfolgern ($\Theta \models \phi$?): Folgt aus Formeln Θ die Formel ϕ logisch?
- Äquivalenz ($\phi \equiv \psi$): Sind Formeln ϕ und ψ logisch äquivalent?
- Erfüllbarkeit (SAT): Ist Formel ϕ erfüllbar? Falls ja, finde eine erfüllende Belegung.

Das Erfüllbarkeitsproblem

Das Erfüllbarkeitsproblem (SAT)

Gegeben:

aussagenlogische Formel in konjunktiver Normalform (KNF)
Üblicherweise repräsentiert als Paar (V, Δ):

- V Menge von Aussagevariablen (Propositionen)
- Δ Menge von Klauseln über V
  (Klausel = Menge von Literalen v bzw. $\neg v$ mit $v \in V$)

Gesucht:

- erfüllende Belegung der Formel (Modell)
- oder Beweis, dass keine erfüllende Belegung existiert

SAT ist ein berühmtes NP-vollständiges Problem (Cook 1971; Levin 1973).

Relevanz von SAT für Informatik

SAT-Algorithmen sehr wichtig und sehr intensiv erforscht!

Constraint-Statisfaction-Problem

SAT kann als Constraint-Satisfaction-Problem (CSP) aufgefasst werden:

- CSP-Variablen = Aussagevariablen Wertebereiche = $\{0, 1\}$ Constraints = Klauseln

Allerdings haben wir hier oft Constraints, die mehr als zwei Variablen betreffen $\Rightarrow$ mehrwertige Logiken!

Constraint-Statisfaction-Problem

Was ist ein Constraint?

constraint = Einschränkung, Nebenbedingung (math.)
Bedingung, die jede Lösung eines Problems erfüllen muss

Verwendung:

- Mathematik: Anforderung an die Lösung eines Optimierungsproblems (z. B. Gleichung, Ungleichung)
- Software-Testing: Spezifikation von Invarianten für Überprüfung von Datenkonsistenz (z.B. Assertions)
- Datenbanken: für Integritätsbedingungen (z.B. foreign key)

Constraint-Satisfaction-Probleme informell

Gegeben:

- eine Menge von Variablen mit einem bestimmten Wertebereich
- eine Menge von Constraints (Bedingungen), die diese Variablen erfüllen müssen meist binär, d. h. jede Bedingung spricht über zwei Variablen

Gesucht:

- eine Belegung der Variablen, die alle Constraints erfüllt

Beispiele

Didaktische Beispiele

... die aber zu viel komplexere Problemstellungen weiterführen

- lateinische Quadrate
- Sudoku
- 8-Damen-Problem Graphfärbung
- Erfüllbarkeit in Aussagenlogik

Komplexere Beispiele:

- Datenbankanfragen
- Gleichungs- und Ungleichungssysteme

Beispiel: lateinische Quadrate

lateinische Quadrate

Wie kann man

- eine $n \times n$-Matrix mit n Symbolen bilden,
- so dass jedes Symbol genau einmal in jeder Zeile und Spalte auftritt?

$$[1] \quad \begin{bmatrix} 1 & 2 \\ 2 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & 2 & 3 \\ 2 & 3 & 1 \\ 3 & 1 & 2 \end{bmatrix} \quad \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 3 & 4 & 1 \\ 3 & 4 & 1 & 2 \\ 4 & 1 & 2 & 3 \end{bmatrix}$$

Es gibt 12 verschiedene lateinische Quadrate der Grösse 3, 576 der Grösse 4, 161'280 der Grösse 5, ..., 5'524'751'496'156'892'842'531'225'600 der Grösse 9.

Beispiel: Sudoku

Sudoku

Wie kann man

- eine partiell gefüllte 9×9-Matrix mit den Zahlen 1–9 füllen,
- so dass jede Zeile, jede Spalte und jeder der neun 3×3-Blöcke jede Zahl genau einmal enthält?

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

Beispiel: Sudoku

Sudoku

Wie kann man

- eine partiell gefüllte 9×9-Matrix mit den Zahlen 1–9 füllen,
- so dass jede Zeile, jede Spalte und jeder der neun 3×3-Blöcke jede Zahl genau einmal enthält?

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

Beispiel: Sudoku

Sudoku

Wie kann man

- eine partiell gefüllte 9×9-Matrix mit den Zahlen 1–9 füllen,
- so dass jede Zeile, jede Spalte und jeder der neun 3×3-Blöcke jede Zahl genau einmal enthält?

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

Zusammenhang lateinische Quadrate?

Sudoku: Trivia

- wohlgeformte Sudokus haben genau eine Lösung
- dafür müssen im Minimalfall 17 Felder vorausgefüllt werden (McGuire et al., 2012)
- 6.670.903.752.021.072.936.960 gelöste Konfigurationen
- nur 5.472.730.538 unter Ausnutzung von Symmetrie

Beispiel: 8-Damen-Problem

8-Damen-Problem

Wie kann man

- 8 Damen auf einem Schachbrett platzieren,
- so dass sich keine zwei Damen bedrohen?

- ursprünglich 1848 vorgestellt
- Varianten: Brettgrösse; andere Figuren; mehr Dimensionen

Das Problem hat 12 Lösungen, wenn man symmetrische Lösungen (Rotationen, Spiegelungen) nur einmal zählt

8-Damen-Problem: Eine Lösung

eine Lösung des 8-Damen-Problems

Beispiel: Graphfärbung

Graphfärbung

Wie kann man

- die Knoten eines gegebenen Graphen mit k Farben
- so einfärben, dass zwei benachbarte Knoten nie dieselbe Farbe haben?

(Der Graph und k sind Parameter des Problems.)

NP-vollständiges Problem

- selbst für den Spezialfall planarer Graphen und $k = 3$
- einfach für $k = 2$ (auch für allgemeine Graphen)

Vier-Farben-Problem

- berühmtes mathematisches Problem: Vier-Farben-Problem
- kann man Landkarten immer mit 4 Farben einfärben?
- Vermutung aufgestellt von Francis Guthrie (1852)
- 1890 erster Beweis, dass 5 Farben ausreichen
- mehrmals falsche Beweise, die über 10 Jahre Bestand hatten
- gelöst 1976 durch Appel und Haken: 4 Farben reichen
- Appel und Haken reduzierten Problem auf 1936 Fälle, die durch Computer überprüft wurden
- erstes grosses offenes Problem in der Mathematik, das per Computer gelöst wurde
- führte zu Kontroverse: Ist das ein Beweis?

Beispiel: Erfüllbarkeit Aussagenlogik

Erfüllbarkeit in Aussagenlogik

Wie kann man

- eine gegebene Menge von Aussagevariablen so mit wahr/falsch belegen,
- dass eine gegebene Menge von Klauseln (Formeln der Art $X \vee \neg Y \vee Z$) erfüllt (wahr) wird?

Anmerkungen:

- NP-vollständig (Cook 1971; Levin 1973)
- Klauselform (statt beliebiger Logikformeln) ist keine echte Einschränkung
- Beschränkung auf Klauseln der Länge 3 wäre ebenfalls keine echte Einschränkung

Zusammenhang zu vorigen Problemen (z.B. Sudoku)?

Praktische Anwendungen

- Es gibt tausende praktische Anwendungen von Constraint-Satisfaction-Problemen.
- Dies gilt allein schon für das Erfüllbarkeitsproblem der Aussagenlogik.

Einige Beispiele:

- Verifikation digitaler Schaltkreise
- Timetabling (Erstellen von Zeit- und Belegungsplänen)
- Planung von Wasser- und Elektrizitätsleitungen, Strassen und ähnlicher Infrastruktur
- Zuweisung von Frequenzspektren (Rundfunk, Mobilfunk)
- Konfiguration von elektronischen Geräten (z. B. Drucker)

Mehr darüber in der KI Vorlesung (4. Semester)!

Aussagenlogische Modellierung

Beispiel 1: Sudoku

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

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

Sudoku

Aussagenlogisches Modell

Koordinaten der Felder:

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

Aussagensymbole:

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

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

Interpretationen beschreiben also Beschriftungen des 9×9-Gitters.

Ziel:

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

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

Sudoku

Aussagenlogisches Modell

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

Beschriftungen:

"Auf jedem Feld steht mindestens eine Zahl":

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

"Auf jedem Feld steht höchstens eine Zahl":

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

Sudoku

Aussagenlogisches Modell

Zeilen:
„Jede Zahl kommt in jeder Zeile vor“:

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

Spalten:
„Jede Zahl kommt in jeder Spalte vor“:

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

Blöcke:
„Jede Zahl kommt in jedem Block vor“:

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

Sudoku

Aussagenlogisches Modell

Anfangsbeschriftung:

Sei A die Anfangsbeschriftung. Wir setzen

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

Automatische Lösung von Sudokus:

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

Automatische Hardwareverifikation

Digitale Schaltkreise

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

  Beispiel:

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

- Schaltkreise sind Kombinationen solcher Schaltelemente. Beispiel:

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

Digitale Schaltkreise

Formalisierung in der Aussagenlogik

Schaltelement:

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

Beispiel:

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

Aussagensymbole:

$P_1,\ P_2,\quad Q_1,\ Q_2$

Formeln:

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

$Q_2 \leftrightarrow (P_1 \wedge P_2)$

Digitale Schaltkreise

Formalisierung in der Aussagenlogik

Schaltkreis:

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

Digitale Schaltkreise

Verifikation

Ziel:

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

Methode:

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

Bemerkung

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

Beispiele für Korrektheitsbedingungen

Schaltkreis:

Einige Korrektheitsbedingungen:

- Bei jeder Eingabe ist mindestens eine Ausgabe 1:

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

- Bei keiner Eingabe sind mehr als zwei Ausgaben 1:

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

Vollständige Spezifikation des Ein-/Ausgabeverhaltens:

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

Logik für Informatiker

Vorlesung 7: Resolution

Babeş-Bolyai Universität, Department für Informatik, Cluj-Napoca
csacarea@cs.ubbcluj.ro

Resolution

Zusammenfassung

- Algorithmische Entscheidung ob eine Formel erfüllbar oder unerfüllbar ist.
- DPLL, GSAT, WALKSAT, Markierungsalgorithmus für Hornformeln.
- Weshalb?
- Mit welchem Ziel?

Resolution

Zusammenfassung

- Algorithmische Entscheidung ob eine Formel erfüllbar oder unerfüllbar ist.
- DPLL, GSAT, WALKSAT, Markierungsalgorithmus für Hornformeln.
- Weshalb?
- Mit welchem Ziel?
- Industrie Anwendung: KI, Industrie 4.0
  - Lösen von NP-vollständige Probleme in technische Anwendungen: Stichwort sehr großer Suchraum.
  - Die Logikschaltungen, aus denen die Computern bestehen, sind elektronische Implementierungen der Logik.

Anwendungen der Logik

```
if (real world applications are those which make money){
  then if (programming makes money){
    print("Programming is a real world application of propositional logic")
    }
  }
```

Anwendungen

- Ein Prozessor (wie in Ihrem Computer oder Mobiltelefon oder in jedem nicht-trivialen elektronischen Gerät) ist nur ein großer Satz, der in der Aussagenlogik geschrieben ist.
- Weitere Anwendungen: Programmieren
  - Die merge Phase des merge sort Algorithmus: Mergesort betrachtet die zu sortierenden Daten als Liste und zerlegt sie in kleinere Listen, die jede für sich sortiert werden. Die sortierten kleinen Listen werden dann im Reißverschlussverfahren zu größeren Listen zusammengefügt (engl. (to) merge), bis wieder eine sortierte Gesamtliste erreicht ist.
  - Das Verfahren arbeitet bei Arrays in der Regel nicht in-place, es sind dafür aber (trickreiche) Implementierungen bekannt, in welchen die Teil-Arrays üblicherweise rekursiv zusammengeführt werden.
  - Verkettete Listen sind besonders geeignet zur Implementierung von Mergesort, dabei ergibt sich die in-place-Sortierung fast von selbst.

Merge-sort Algorithmus

- Das Befehl `if q = k or (p not= j and B[p] < B[q])`
  `then (A[r] := B[p]; p:=p+1)`
  `else (A[r] := B[q]; q:=q+1)`
- reduziert sich auf: Gegeben `not(P and Q)` zeige `not( Q or (not P and R)) = (P or (not Q and not R))`

Resolution

Klauselmengen

Für den Rest des Kapitels:

- Voraussetzung: Formeln sind in konjunktiver Normalform
- Darstellung von Klauseln als Menge C von Literalen
- Darstellung von Formeln als Menge Δ von Klauseln

Beispiel

Sei $\varphi = (P \vee Q) \wedge \neg P$.

- φ ist in konjunktiver Normalform,
- φ besteht aus den Klauseln $(P \vee Q)$ und $\neg P$.
- Darstellung von φ als Menge von Mengen von Literalen: $\{\{P, Q\}, \{\neg P\}\}$

Unterscheide $\square$ (leere Klausel) vs. $\emptyset$ (leere Menge von Klauseln).

Resolution: Idee

Beobachtung

- Prüfen auf Allgemeingültigkeit kann auf Prüfen von Unerfüllbarkeit zurückgeführt werden.
- Formel ϕ allgemeingültig gdw. $\neg\phi$ unerfüllbar.

Resolution: Idee

- Methode, um Unerfüllbarkeit einer Formel ϕ zu prüfen.
- Idee: Leite aus ϕ neue Formeln ab, die aus ϕ logisch folgen.
- Wenn leere Klausel $\square$ abgeleitet werden kann, dann ist ϕ unerfüllbar.

Resolutionskalkül

Definition: Resolutionsregel (einzige Regel des Kalküls)

$$\frac{C_1 \cup \{P\} \qquad \{\neg P\} \cup C_2}{C_1 \cup C_2}$$

wobei

- P eine aussagenlogische Variable
- C_1, C_2 Klauseln (können leer sein)

Definition:

$C_1 \cup C_2$ heißt Resolvente von $C_1 \cup \{P\}, C_2 \cup \{\neg P\}$

Resolution

Beispiel

Gegeben die Klauselmenge:

$$M = \{\{P_1, P_2\}, \{P_1, \neg P_2\}, \{\neg P_1, P_2\}, \{\neg P_1, \neg P_2\}\}$$

Resolution

Beispiel

Gegeben die Klauselmenge:

$$M = \{\{P_1, P_2\}, \{P_1, \neg P_2\}, \{\neg P_1, P_2\}, \{\neg P_1, \neg P_2\}\}$$

Resolution:

$$\frac{\{P_1, P_2\} \quad \{P_1, \neg P_2\}}{}$$

Resolution

Beispiel

Gegeben die Klauselmenge:

$$M = \{\{P_1, P_2\}, \{P_1, \neg P_2\}, \{\neg P_1, P_2\}, \{\neg P_1, \neg P_2\}\}$$

Resolution:

$$\frac{\{P_1, P_2\} \quad \{P_1, \neg P_2\}}{\{P_1\}}$$

$$\underline{\{\neg P_1, P_2\} \quad \{\neg P_1, \neg P_2\}}$$

Resolution

Beispiel

Gegeben die Klauselmenge:

$$M = \{\{P_1, P_2\}, \{P_1, \neg P_2\}, \{\neg P_1, P_2\}, \{\neg P_1, \neg P_2\}\}$$

Resolution:

$$\frac{\{P_1, P_2\} \quad \{P_1, \neg P_2\}}{\{P_1\}}$$

$$\frac{\{\neg P_1, P_2\} \quad \{\neg P_1, \neg P_2\}}{\{\neg P_1\}}$$

Resolution

Beispiel

Gegeben die Klauselmenge:

$$M = \{\{P_1, P_2\}, \{P_1, \neg P_2\}, \{\neg P_1, P_2\}, \{\neg P_1, \neg P_2\}\}$$

Resolution:

$$\frac{\{P_1, P_2\} \quad \{P_1, \neg P_2\}}{\{P_1\}}$$

$$\frac{\{\neg P_1, P_2\} \quad \{\neg P_1, \neg P_2\}}{\{\neg P_1\}}$$

$$\frac{\{P_1\} \quad \{\neg P_1\}}{\bot}$$

Resolution

Beispiel

Gegeben die Klauselmenge:

$$M = \{\{P_1, P_2\}, \{P_1, \neg P_2\}, \{\neg P_1, P_2\}, \{\neg P_1, \neg P_2\}\}$$

Resolution:

$$\frac{\{P_1, P_2\} \quad \{P_1, \neg P_2\}}{\{P_1\}}$$

$$\frac{\{\neg P_1, P_2\} \quad \{\neg P_1, \neg P_2\}}{\{\neg P_1\}}$$

$$\frac{\{P_1\} \quad \{\neg P_1\}}{\bot}$$

Insgesamt: $M \vdash_{\mathrm{Res}} \bot$

also: M unerfüllbar

Resolution

Beispiel

Zu zeigen:

$$(P \to Q) \to ((Q \to R) \to (P \to R))$$

ist allgemeingültig

Resolution

Beispiel

Zu zeigen:

$$(P \to Q) \to ((Q \to R) \to (P \to R))$$

ist allgemeingültig

Dazu zeigen wir, dass

$$\neg[(P \to Q) \to ((Q \to R) \to (P \to R))]$$

unerfüllbar ist.

Resolution

Beispiel

Zu zeigen:

$$(P \to Q) \to ((Q \to R) \to (P \to R))$$

ist allgemeingültig

Dazu zeigen wir, dass

$$\neg[(P \to Q) \to ((Q \to R) \to (P \to R))]$$

unerfüllbar ist.

Klauselnormalform:

$$\{\{\neg P, Q\}, \{\neg Q, R\}, \{P\}, \{\neg R\}\}$$

Resolution

Beispiel

Klauselnormalform:

$$M = \{\{\neg P, Q\}, \{\neg Q, R\}, \{P\}, \{\neg R\}\}$$

Ableitung der leeren Klausel aus M:

(1)	$\{\neg P, Q\}$	gegeben
(2)	$\{\neg Q, R\}$	gegeben
(3)	$\{P\}$	gegeben
(4)	$\{\neg R\}$	gegeben

Resolution

Beispiel

Klauselnormalform:

$$M = \{\{\neg P, Q\}, \{\neg Q, R\}, \{P\}, \{\neg R\}\}$$

Ableitung der leeren Klausel aus M:

(1)	$\{\neg P, Q\}$	gegeben
(2)	$\{\neg Q, R\}$	gegeben
(3)	$\{P\}$	gegeben
(4)	$\{\neg R\}$	gegeben
(5)	$\{Q\}$	aus (1) und (3)

Resolution

Beispiel

Klauselnormalform:

$$M = \{\{\neg P, Q\}, \{\neg Q, R\}, \{P\}, \{\neg R\}\}$$

Ableitung der leeren Klausel aus M:

(1)	$\{\neg P, Q\}$	gegeben
(2)	$\{\neg Q, R\}$	gegeben
(3)	$\{P\}$	gegeben
(4)	$\{\neg R\}$	gegeben
(5)	$\{Q\}$	aus (1) und (3)
(6)	$\{R\}$	aus (2) und (5)

Resolution

Beispiel

Klauselnormalform:

$$M = \{\{\neg P, Q\}, \{\neg Q, R\}, \{P\}, \{\neg R\}\}$$

Ableitung der leeren Klausel aus M:

(1)	$\{\neg P, Q\}$	gegeben
(2)	$\{\neg Q, R\}$	gegeben
(3)	$\{P\}$	gegeben
(4)	$\{\neg R\}$	gegeben
(5)	$\{Q\}$	aus (1) und (3)
(6)	$\{R\}$	aus (2) und (5)
(7)	$\perp$	aus (4) und (6)

Resolution

Beispiel

Die Menge

$$E = \{P_1 \vee \neg P_2, \neg P_1 \vee P_2, \neg P_1 \vee \neg P_2, P_1 \vee P_2\}$$

ist unerfüllbar.

Klauselmenge:

$$M = \{\{P_1, P_2\}, \{P_1, \neg P_2\}, \{\neg P_1, P_2\}, \{\neg P_1, \neg P_2\}\}$$

Resolution:

$$\frac{\{P_1, P_2\} \quad \{P_1, \neg P_2\}}{\{P_1\}}$$

$$\frac{\{\neg P_1, P_2\} \quad \{\neg P_1, \neg P_2\}}{\{\neg P_1\}}$$

$$\frac{\{P_1\} \quad \{\neg P_1\}}{\bot}$$

Resolution: Bemerkungen

- Vorsicht bei Klauseln mit mehreren Resolutionsmöglichkeiten
- Zwei Klauseln können mehr als eine Resolvente haben
- z.B.: $\{A, B\}$ und $\{\neg A, \neg B\}$
- $\{A, B, C\}$ und $\{\neg A, \neg B, D\}$ haben NICHT $\{C, D\}$ als Resolvente

Heuristik: Immer möglichst kleine Klauseln ableiten!

Resolutionsregel

Graphische Darstellung

$$A \vee B_1 \vee \ldots \vee B_n \qquad\qquad \neg A \vee C_1 \vee \ldots \vee C_m$$

$$B_1 \vee \ldots \vee B_n \vee C_1 \vee \ldots \vee C_m$$

Resolutionsverfahren

1. Starte mit Klauselmenge C
2. Wähle Elternklauseln E_1, E_2 aus C
3. Füge Resolvente R zur Klauselmenge C hinzu:

$$C := C \cup \{R\}$$

4. Iteriere die obigen Schritte solange bis: Keine neue Resolvente mehr herleitbar, oder C enthält die leere Klausel $\square$.

Resolution

Beispiel

Zeige

$$(A \wedge (A \Rightarrow B) \wedge (B \Rightarrow C)) \implies C \text{ allgemeingültig.}$$

Starte mit negierter Formel (Test auf Widersprüchlichkeit)

$$\neg((A \wedge (A \Rightarrow B) \wedge (B \Rightarrow C)) \implies C)$$

CNF-Berechnung ergibt die Klauselmenge:

$$\{\{A\}, \{-A, B\}, \{-B, C\}, \{-C\}\}$$

Resolution

Beispiel

Resolution

Resolvente

Sei F eine Klauselmenge und

$$\mathrm{Res}(F) = F \cup \{R \mid R \text{ ist eine Resolvente zweier Klauseln aus } F\}$$

$\mathrm{Res}^0(F) = F$

$\mathrm{Res}^{n+1}(F) = \mathrm{Res}(\mathrm{Res}^n(F))$

$\mathrm{Res}^\star(F) = \bigcup_{n\in\mathbb{N}} \mathrm{Res}^n(F)$

(bezeichnet die Vereinigung der Resolventen aus aller möglichen Resolutionsschritte auf F)

Resolution

Resolvente

Sei F eine Klauselmenge und

$$\mathrm{Res}(F) = F \cup \{R \mid R \text{ ist eine Resolvente zweier Klauseln aus } F\}$$

$\mathrm{Res}^0(F) = F$

$\mathrm{Res}^{n+1}(F) = \mathrm{Res}(\mathrm{Res}^n(F))$

$\mathrm{Res}^\star(F) = \bigcup_{n\in\mathbb{N}} \mathrm{Res}^n(F)$

(bezeichnet die Vereinigung der Resolventen aus aller möglichen Resolutionsschritte auf F)

Notation: Falls $C \in \mathrm{Res}^\star(F)$, so schreiben wir $F \vdash_{\mathrm{Res}} C$.

Resolution

Resolvente

Sei F eine Klauselmenge und

$$\mathrm{Res}(F) = F \cup \{R \mid R \text{ ist eine Resolvente zweier Klauseln aus } F\}$$

$\mathrm{Res}^0(F) = F$

$\mathrm{Res}^{n+1}(F) = \mathrm{Res}(\mathrm{Res}^n(F))$

$\mathrm{Res}^\star(F) = \bigcup_{n\in\mathbb{N}} \mathrm{Res}^n(F)$

(bezeichnet die Vereinigung der Resolventen aus aller möglichen Resolutionsschritte auf F)

Notation: Falls $C \in \mathrm{Res}^\star(F)$, so schreiben wir $F \vdash_{\mathrm{Res}} C$.

Definition: Beweis für C (aus F): $C_1, \ldots C_n$, wobei:
$C_n = C$ und für alle $1 \leq i \leq n$: ($C_i \in F$ oder C_i Resolvente für $\frac{C_{j_1} \quad C_{j_2}}{C_i}$ mit $j_1, j_2 < i$).

Resolution

Beispiel

Klauselnormalform:

$$M = \{\{\neg P, Q\}, \{\neg Q, R\}, \{P\}, \{\neg R\}\}$$

(1)	$\{\neg P, Q\}$	gegeben
(2)	$\{\neg Q, R\}$	gegeben
(3)	$\{P\}$	gegeben
(4)	$\{\neg R\}$	gegeben
(5)	$\{Q\}$	aus (1) und (3)
(6)	$\{R\}$	aus (2) und (5)

Beweis für $\{R\}$ aus M

Resolution

Beispiel

Klauselnormalform:

$$M = \{\{\neg P, Q\}, \{\neg Q, R\}, \{P\}, \{\neg R\}\}$$

(1)	$\{\neg P, Q\}$	gegeben
(2)	$\{\neg Q, R\}$	gegeben
(3)	$\{P\}$	gegeben
(4)	$\{\neg R\}$	gegeben
(5)	$\{Q\}$	aus (1) und (3)
(6)	$\{R\}$	aus (2) und (5)
(7)	$\perp$	aus (4) und (6)

Beweis für $\perp$ aus M (Widerspruch)

Resolution

Korrektheit und Vollständigkeit

Theorem (Korrektheit)
Für eine Menge M von Klauseln gilt: Falls $M \vdash_{\text{Res}} \bot$, so M unerfüllbar.

Falls aus M die leere Klausel durch Resolution herleitbar ist, ist M unerfüllbar (es gibt keine Wertebelegung $\mathcal{A}$ in der alle Klauseln in M wahr sind).

Äquivalent:

Falls M erfüllbar ist, ist die leere Klausel durch Resolution nicht herleitbar.

Theorem (Vollständigkeit)
Für eine Menge M von Klauseln gilt: Falls M unerfüllbar, so $M \vdash_{\text{Res}} \bot$.

Falls M unerfüllbar ist (d.h. es gibt keine Wertebelegung $\mathcal{A}$ in der alle Klauseln in M wahr sind), ist die leere Klausel aus M durch Resolution herleitbar.

Äquivalent:

Falls aus M die leere Klausel durch Resolution nicht herleitbar ist, ist M erfüllbar.

Resolution

Korrektheit

Theorem (Korrektheit)
Für eine Menge M von Klauseln gilt: Falls $M \vdash_{\mathsf{Res}} \bot$, so M unerfüllbar.

Resolution

Korrektheit

Theorem (Korrektheit)

Für eine Menge M von Klauseln gilt: Falls wir aus M mit Hilfe des Resolutionsverfahrens das Falsum $\perp$ herleiten können, so ist M unerfüllbar.

Beweis:

1. Wir zeigen, dass falls $C \in \mathrm{Res}^*(M)$, so $M \equiv M \cup \{C\}$. ($M \cup \{C\}$: Notation für $M \wedge C$.)
2. Es folgt, dass falls aus M Falsum herleitbar ist, so ist $\perp$ in der Resolvente von M, d.h. $M \equiv M \cup \{\perp\}$.
3. Aber $M \cup \{\perp\}$ ist unerfüllbar, deshalb ist auch M unerfüllbar.

Resolution

Korrektheit

(1) Wir zeigen, dass falls $C \in \mathrm{Res}^*(M)$, so $M \equiv M \cup \{C\}$.

Lemma: $C_1 \vee P, C_2 \vee \neg P \models C_1 \vee C_2$

Beweis:

Sei $\mathcal{A}$ Interpretation mit $\mathcal{A}(C_1 \vee P) = 1$ und $\mathcal{A}(C_2 \vee \neg P) = 1$.

Zu zeigen: $\mathcal{A}(C_1 \vee C_2) = 1$.

Fall 1: $\mathcal{A}(C_1) = 1$. Dann $\mathcal{A}(C_1 \vee C_2) = 1$.

Fall 2: $\mathcal{A}(C_1) = 0$. Dann $\mathcal{A}(P) = 1$.
Da $\mathcal{A}(C_2 \vee \neg P) = 1$, so $\mathcal{A}(C_2) = 1$, d.h. $\mathcal{A}(C_1 \vee C_2) = 1$.

Resolution

Korrektheit

Theorem: Falls $C \in \mathrm{Res}^\star(F)$, so $F \equiv F \cup \{C\}$.

Beweis: Annahme: $C \in \mathrm{Res}^n(F)$. Zu zeigen: $F \equiv F \wedge C$, i.e.:
Für alle $\mathcal{A} : \Pi \to \{0,1\}$ gilt: $\mathcal{A}(F) = 1$ gdw. $(\mathcal{A}(F) = 1$ und $\mathcal{A}(C) = 1)$.

- Sei $\mathcal{A}{:}\Pi{\to}\{0,1\}$ mit $\mathcal{A}(F){=}1$ (d.h. $\mathcal{A}(C'){=}1$ für jede Klausel $C'{\in}F$).
  Zu zeigen: Für alle $m \in \mathbb{N}$ gilt $p(m)$: Falls $D \in \mathrm{Res}^m(F)$, so $\mathcal{A}(D) = 1$.
  Beweis durch Induktion:
  IB: $m = 0$: Sei $D \in \mathrm{Res}^0(F) = F$. Dann $\mathcal{A}(D) = 1$.
  IV: Annahme: $p(m)$ is wahr. **IS:** Zu zeigen: $p(m+1)$ wahr.
  Sei $D \in \mathrm{Res}^{m+1}(F)$. Fall 1: $D \in \mathrm{Res}^m(F)$. Nach IV, $\mathcal{A}(D) = 1$.
  Fall 2: $D{=}C_1{\vee}C_2$ Resolvente von $C_1{\vee}P$, $C_2{\vee}\neg P \in \mathrm{Res}^m(F)$. Nach IV:
  $\mathcal{A}(C_1{\vee}P){=}\mathcal{A}(C_2{\vee}\neg P){=}1$. Da $C_1{\vee}P, C_2{\vee}\neg P \models C_1{\vee}C_2$ folgt $\mathcal{A}(D){=}1$.
  Dann $\mathcal{A}(C) = 1$, so $(\mathcal{A}(F) = 1$ und $\mathcal{A}(C) = 1)$, d.h. $\mathcal{A}(F \wedge C) = 1$.
- Sei $\mathcal{A} : \Pi \to \{0,1\}$ mit $\mathcal{A}(F \wedge C) = 1$.
  Dann $\mathcal{A}(F) = 1$ und $\mathcal{A}(C) = 1$, so $\mathcal{A}(F) = 1$.

Resolution

Vollständigkeit

Theorem.
Für jede endliche Menge M von Klauseln gilt:
falls M unerfüllbar, so $M \vdash_{\mathsf{Res}} \bot$.

Resolution

Vollständigkeit

Theorem.
Für jede endliche Menge M von Klauseln gilt:
falls M unerfüllbar, so $M \vdash_{\mathsf{Res}} \bot$.

Beweis: Induktion:

$p(n)$: Sei M Menge von Klauseln mit n Aussagenvariablen.
Falls M unerfüllbar, so $M \vdash_{\mathsf{Res}} \bot$

Resolution

Vollständigkeit

Theorem.
Für jede endliche Menge M von Klauseln gilt:
falls M unerfüllbar, so $M \vdash_{\text{Res}} \bot$.

Beweis: Induktion:

$p(n)$: Sei M Menge von Klauseln mit n Aussagenvariablen.
Falls M unerfüllbar, so $M \vdash_{\text{Res}} \bot$

Induktionsbasis: $n = 0$.
Dann $M = \{\bot\}$, d.h. $M \vdash_{\text{Res}} \bot$

Resolution

Vollständigkeit

Theorem.
Für jede endliche Menge M von Klauseln gilt:
falls M unerfüllbar, so $M \vdash_{\text{Res}} \bot$.

Beweis: Induktion:

$p(n)$: Sei M Menge von Klauseln mit n Aussagenvariablen.
Falls M unerfüllbar, so $M \vdash_{\text{Res}} \bot$

Induktionsbasis: $n = 0$. Dann $M = \{\bot\}$, d.h. $M \vdash_{\text{Res}} \bot$

Induktionsvoraussetzung: $p(n)$ gilt.
Induktionsschritt: Beweise, dass $p(n+1)$ gilt.

Sei M Menge von Klauseln mit Aussagenvariablen $\{P_1, \ldots, P_n, P_{n+1}\}$.

M_0 sei aus M entstanden durch Ersetzung von P_{n+1} durch $\bot$
M_1 sei aus M entstanden durch Ersetzung von P_{n+1} durch $\top$

Resolution

Vollständigkeit

M_0 sei aus M entstanden durch Ersetzung von P_{n+1} durch $\bot$:
- P_{n+1} wird aus allen Klauseln gelöscht,
- Klauseln, die $\neg P_{n+1}$ enthalten werden ebenfalls gelöscht

M_1 sei aus M entstanden durch Ersetzung von P_{n+1} durch $\top$:
- $\neg P_{n+1}$ wird aus allen Klauseln gelöscht,
- Klauseln, die P_{n+1} enthalten werden ebenfalls gelöscht

Fakten:

- M_0, M_1 enthalten nur Aussagenvariablen $\{P_1, \ldots, P_n\}$
- M_0, M_1 unerfüllbar

Resolution

Vollständigkeit

M_0 sei aus M entstanden durch Ersetzung von P_{n+1} durch $\bot$:
- P_{n+1} wird aus allen Klauseln gelöscht,
- Klauseln, die $\neg P_{n+1}$ enthalten werden ebenfalls gelöscht

M_1 sei aus M entstanden durch Ersetzung von P_{n+1} durch $\top$:
- $\neg P_{n+1}$ wird aus allen Klauseln gelöscht,
- Klauseln, die P_{n+1} enthalten werden ebenfalls gelöscht

Fakten:

- M_0, M_1 enthalten nur Aussagenvariablen $\{P_1, \ldots, P_n\}$
- M_0, M_1 unerfüllbar

Induktionsvoraussetzung:

$M_0 \vdash_{\text{Res}} \bot$, i.e. es gibt $C_1, C_2, \ldots, C_m$ Beweis (aus M_0) für $\bot$

P_{n+1} zurück: $C'_1, C'_2, \ldots, C'_m$ Beweis (aus M) für $\bot$ oder P_{n+1}

$M_1 \vdash_{\text{Res}} \bot$, i.e. es gibt $D_1, D_2, \ldots, D_k$ Beweis (aus M_1) für $\bot$

$\neg P_{n+1}$ zurück: $D'_1, D'_2, \ldots, D'_k$ Beweis (aus M) für $\bot$ oder $\neg P_{n+1}$

Resolution

Vollständigkeit

M_0 sei aus M entstanden durch Ersetzung von P_{n+1} durch $\bot$:
- P_{n+1} wird aus allen Klauseln gelöscht,
- Klauseln, die $\neg P_{n+1}$ enthalten werden ebenfalls gelöscht

M_1 sei aus M entstanden durch Ersetzung von P_{n+1} durch $\top$:
- $\neg P_{n+1}$ wird aus allen Klauseln gelöscht,
- Klauseln, die P_{n+1} enthalten werden ebenfalls gelöscht

Fakten:

- M_0, M_1 enthalten nur Aussagenvariablen $\{P_1, \ldots, P_n\}$
- M_0, M_1 unerfüllbar

Induktionsvoraussetzung:

$$\left.\begin{array}{l} M_0 \vdash_{\mathrm{Res}} \bot, \text{ i.e. es gibt } C_1, C_2, \ldots, C_m \text{ Beweis (aus } M_0\text{) für } \bot \\ \quad P_{n+1} \text{ zurück: } C'_1, C'_2, \ldots, C'_m \text{ Beweis (aus } M\text{) für } \bot \text{ oder } P_{n+1} \\ M_1 \vdash_{\mathrm{Res}} \bot, \text{ i.e. es gibt } D_1, D_2, \ldots, D_k \text{ Beweis (aus } M_1\text{) für } \bot \\ \quad \neg P_{n+1} \text{ zurück: } D'_1, D'_2, \ldots, D'_k \text{ Beweis (aus } M\text{) für } \bot \text{ oder } \neg P_{n+1} \end{array}\right\} \Rightarrow M \vdash_{\mathrm{Res}} \bot$$

Resolution

Vollständigkeit

Theorem.
Für jede endliche Menge M von Klauseln gilt:
falls M unerfüllbar, so $M \vdash_{\mathsf{Res}} \bot$.

Es gilt auch:

Theorem.
Für jede Menge M von Klauseln gilt:
falls M unerfüllbar, so $M \vdash_{\mathsf{Res}} \bot$.

Resolution

Terminierung

Theorem.
Aussagenlogische Resolution (für Klauselmengen in Mengennotation) terminiert, für jede endliche Menge von Klauseln.

Resolution

Terminierung

Theorem.
Aussagenlogische Resolution (für Klauselmengen in Mengennotation) terminiert, für jede endliche Menge von Klauseln.

Beweis: Es gibt nicht mehr als 2^{2n} Klauseln (in Mengennotation) mit n Aussagenvariablen.

$\mapsto$ nicht mehr als $(2^{2n})^2$ mögliche Anwendungen der Resolutionsregel.

Logik für Informatiker

Vorlesung 8: Prädikatenlogik

Babeş-Bolyai Universität, Department für Informatik, Cluj-Napoca
csacarea@cs.ubbcluj.ro

Aussagenlogik

- Aussagenlogik ist deklarativ: Syntaxelemente entsprechen Fakten
- Aussagenlogik erlaubt konjunktive / disjunktive / negative Aussagen (im Gegensatz zu vielen Datenstrukturen und Datenbanken)
- Aussagenlogik ist kompositional: Bedeutung von $F_1 \wedge F_2$ leitet sich ab aus der von F_1 und F_2
- Bedeutung in Aussagenlogik ist kontextunabhängig (im Gegensatz zu natürlicher Sprache)
- **ABER** Aussagenlogik hat nur beschränkte Ausdruckskraft (im Vergleich zu natürlicher Sprache)

Beispiele:

Die Aussage Jede natürliche Zahl ist entweder gerade oder ungerade erfordert eine Formel für jede Zahl.

Aussagenlogik nicht ausreichend

Aussagenlogik nicht ausreichend

Logic: another thing that penguins aren't very good at.

Weitere Logiken

Logik		
Aussagenlogik	Fakten	wahr/falsch
Prädikatenlogik	Objekte, Funktionen, Relationen	wahr/falsch
Temporallogik	Fakten, Zeitpunkte	wahr/falsch
Mehrwertige Logik	Fakten	wahr/falsch/unbekannt
Fuzzy-Logik	Fakten	[0, 1]
...	...	...

Prädikatenlogik

- Reichere Struktur

Prädikatenlogik

- Reichere Struktur
- Objekte (Elemente): Leute, Häuser, Zahlen, Theorien, Farben, Jahre, ...
- Relationen (Eigenschaften): rot, rund, prim, mehrstöckig, ..., ist Bruder von, ist größer als, ist Teil von, hat Farbe, besitzt, ... $=, \geq$, ...
- Funktionen: +, Mitte von, Vater von, Anfang von, ...

Syntax der Prädikatenlogik: Logische Zeichen

Wie in der Aussagenlogik

$\top$ Symbol für die Formel "wahr" (Formel, die immer wahr ist)
$\bot$ Symbol für die Formel "falsch" (Formel, die immer falsch ist)

$\neg$ Negationssymbol ("nicht")

$\wedge$ Konjunktionssymbol ("und")
$\vee$ Disjunktionssymbol ("oder")
$\rightarrow$ Implikationssymbol ("wenn . . . dann")
$\leftrightarrow$ Symbol für Äquivalenz ("genau dann, wenn")

( ) die beiden Klammern

Quantoren

$\forall$ Allquantor ("für alle")
$\exists$ Existenzquantor ("es gibt")

Syntax der Prädikatenlogik: Vokabular

Definition

Prädikatenlogische Signatur: Paar $\Sigma = (\Omega, \Pi)$ wobei:

- Ω eine Menge von Funktionssymbolen f mit Stelligkeit $n \geq 0$, geschrieben f/n,

  z.B. 2/0, koblenz/0, c/0, sqrt/1, leftLegOf/1, +/2, -/2

- Π Menge von Prädikatensymbolen p mit Stelligkeit $m \geq 0$, geschrieben p/m

  z.B. bruderVon/1, even/1, $> /2, \approx /2, \ldots$

Bemerkung: Das Gleichheitsprädikat $\approx$ kann (muss aber nicht) enthalten sein. Es wird infix notiert.

Syntax der Prädikatenlogik: Vokabular

- Funktionssymbole mit Stelligkeit $n = 0$ heißen Konstante z.B. 1, 2, a, b, c
- Prädikatensymbole mit Stelligkeit $n = 0$ heißen Aussagenvariablen

Syntax der Prädikatenlogik: Vokabular

Variablen

- Die Prädikatenlogik erlaubt die Formulierung abstrakter (schematischer) Aussagen.
- Technisches Hilfmittel hierfür sind die Variablen.
- Wir nehmen an, dass X eine vorgegebene Menge von abzählbar unendlich vielen Symbolen ist, die wir für (die Bezeichnung von) Variablen verwenden.

Terme

Terme über Σ (bzw. Σ-Terme) werden nach folgenden syntaktischen Regeln gebildet:

$$
\begin{array}{llll}
s, t, u, v & ::= & x & , x \in X \quad \text{(Variable)} \\
 & \mid & f(s_1, \ldots, s_n) & , f/n \in \Omega \quad \text{(F-Terme)}
\end{array}
$$

Mit $T_\Sigma(X)$ bezeichnen wir die Menge der Σ-Terme

Menge $T_\Sigma(X)$ der Σ-Terme:

Die kleinste Menge mit: $X \subseteq T_\Sigma(X)$

Wenn
- $f \in \Omega$,
- n ist die Stelligkeit von f
- $t_1, \ldots, t_n \in T_\Sigma(X)$

dann $f(t_1, \ldots, t_n) \in T_\Sigma(X)$

Terme

- Terme sind also vollständig geklammerte Ausdrücke, die wir auch als markierte, geordnete Bäume auffassen können.
- Die Knoten sind mit Funktionssymbolen oder Variablen markiert.
- Jeder mit einem Funktionssymbol f der Stelligkeit n markierte Knoten hat genau n Unterbäume, einen für jedes Argument von f.

Beispiele

$\Sigma = (\Omega, \Pi)$

$\Omega = \{Jan/0, Vater/1, Mutter/1\}$

$\{x, y, z\} \subseteq X$

Terme:

x	Jan
$Vater(x)$	$Vater(Jan)$
$Mutter(x)$	$Mutter(Jan)$
$Vater(Mutter(x))$	$Vater(Mutter(Jan))$

Beispiele

$\Sigma = (\Omega, \Pi)$

$\Omega = \{0/0, 1/0, succ/1, +/2\}$

$\{x, y, z\} \subseteq X$

Terme:

$x, y, z, 0, 1$

$succ(x), succ(0), succ(1)$

$x + y, x + z, x + 0, x + 1, (x + (succ(y) + 1)), \ldots$

Beispiele

$\Sigma = (\Omega, \Pi)$

$\Omega = \{0/0, 1/0, succ/1, +/2\}$

$\{x, y, z\} \subseteq X$

Terme:

$x, y, z, 0, 1$

$succ(x), succ(0), succ(1)$

$x + y, x + z, x + 0, x + 1, (x + (succ(y) + 1)), \ldots$ Infix

$+(x, y), +(x, z), +(x, 0), +(x, 1), +(x, +(succ(y), 1))$ Präfix

Atome

Atome (Atomare Formeln) über Σ genügen dieser Syntax:

$$\begin{array}{llll} A, B & ::= & p(s_1, \ldots, s_m) & , p/m \in \Pi \\ \left[\quad & | & (s \approx t) & \text{(Gleichung)} \quad \right] \end{array}$$

wobei $s_1, \ldots, s_m, s, t \in T_\Sigma(X)$.

Ist $m = 0$, so handelt es sich bei p um eine Aussagenvariable.

Atome:
Formeln der Form $p(s_1, \ldots, s_m)$, wobei $p/m \in \Pi$ und $s_1, \ldots, s_m \in T_\Sigma(X)$.

Beispiele

$\Sigma = (\Omega, \Pi)$

$\Omega = \{Jan/0, Anna/0, Vater/1, Mutter/1\}$

$\Pi = \{Mann/1, Frau/1, Bruder/2, \approx /2\}$

$\{x, y, z\} \subseteq X$

Atome:

$Mann(x)$	$Mann(Jan)$	$Mann(Anna)$
$Mann(Vater(x))$	$Frau(Vater(Jan))$	$Mann(Vater(Jan))$
$Bruder(x, y)$	$Bruder(x, Jan)$	$Bruder(x, Anna)$

$Vater(Jan) \approx Vater(Anna)$

$Vater(x) \approx Mutter(y)$

Beispiele

$\Sigma = (\Omega, \Pi)$

$\Omega = \{0/0, 1/0, succ/1, +/2\}$

$\Pi = \{\leq, \approx, \text{even}, \text{odd}\}$

$\{x, y, z\} \subseteq X$

Atome:

$x \leq y, \quad z \approx 0, \quad 0 \leq 1$

$succ(x) \leq succ(0), \quad succ(1) \approx succ(0)$

$x + y \leq x + z, \quad x + 0 \approx x + 1 \quad (x + (succ(y) + 1)) \leq z, \ldots$ Infix

$\text{even}(succ(0)), \quad \text{even}(x), \quad \text{even}(x + 1), \quad \text{odd}(x + (succ(y) + y))$

Beispiele

$\Sigma = (\Omega, \Pi)$

$\Omega = \{0/0, 1/0, succ/1, +/2\}$

$\Pi = \{\leq, \approx, \text{even}, \text{odd}\}$

$\{x, y, z\} \subseteq X$

Atome:

$x \leq y, \quad z \approx 0, \quad 0 \leq 1$

$succ(x) \leq succ(0), \quad succ(1) \approx succ(0)$

$x + y \leq x + z, \quad x + 0 \approx x + 1 \quad (x + (succ(y) + 1)) \leq z, \ldots$ Infix

Präfix:

$\leq (+(x, y), +(x, z)), \quad +(x, 0) \approx +(x, 1), \quad \leq (+(x, +(succ(y), 1)), z)$

$\text{even}(succ(0)), \quad \text{even}(x), \quad \text{even}(x + 1), \quad \text{odd}(x + (succ(y) + y))$

Literale

$$
\begin{array}{lcll}
L & ::= & A & \text{(positives Literal)} \\
 & | & \neg A & \text{(negatives Literal)}
\end{array}
$$

Beispiele:

$$
\begin{array}{rr}
Mann(Vater(x)) & Vater(Jan) \approx Vater(Anna) \\
\neg Mann(Vater(x)) & \neg(Vater(Jan) \approx Vater(Anna))
\end{array}
$$

$$
\begin{array}{rr}
x \leq y & (x + (succ(y) + 1)) \approx z \\
\neg x \leq y & \neg((x + (succ(y) + 1)) \approx z)
\end{array}
$$

Klauseln

$$
\begin{array}{lcll}
C, D & ::= & \bot & \text{(leere Klausel)} \\
 & | & L_1 \vee \ldots \vee L_k,\ k \geq 1 & \text{(nichtleere Klausel)}
\end{array}
$$

Beispiele:

$\bot$

$Mann(Vater(x)) \quad \vee \quad \neg(Vater(Jan) \approx Vater(Anna))$

$\neg Frau(Vater(x))$

$\neg(x \leq y) \quad \vee \quad (x + (succ(y) + 1)) \approx z$

$x \leq y$

$\neg(x \leq y) \quad \vee \quad \neg(x \leq y) \quad \vee \quad (x \leq y)$

Zusammenfassung

Syntax

1. **Logische Symbole:**
   1.1: Wie in der Aussagenlogik: $\top, \bot$; $\neg$; $\vee, \wedge, \rightarrow, \leftrightarrow$
   1.2: Quantoren: $\forall$, $\exists$.

2. **Nichtlogische Symbole:** Signatur $\Sigma = (\Omega, \Pi)$,
   2.1: Ω Menge von Funktionssymbolen. Notation: f/n: f hat Stelligkeit $n \geq 0$,
   2.2: Π Menge von Prädikatensymbolen. Notation: p/m: p hat Stelligkeit $m \geq 0$.
   (Das Gleichheitsprädikat $\approx$ kann (muss aber nicht) enthalten sein.)

   Funktionsssymbole mit Stelligkeit $n = 0$ heißen Konstanten.

   Prädikatensymbole mit Stelligkeit $n = 0$ heißen Aussagenvariablen.

3. **Variablen:** X vorgegebene Menge von abzählbar unendlich vielen Symbolen, die wir für (die Bezeichnung von) Variablen verwenden.

Formeln

Menge For_Σ der Formeln über Σ:

Die kleinste Menge, die

- Alle atomaren Formeln enthält,
- $\top \in \mathsf{For}_\Sigma$, $\bot \in \mathsf{For}_\Sigma$,
- Wenn $F, G \in \mathsf{For}_\Sigma$, dann auch
  $\neg F, F \wedge G, F \vee G, F \rightarrow G, F \leftrightarrow G \in \mathsf{For}_\Sigma$,
- Wenn $F \in \mathsf{For}_\Sigma$ und $x \in X$, dann
  $\forall x F \in \mathsf{For}_\Sigma, \exists x F \in \mathsf{For}_\Sigma$

Beispiele

X Variablenmenge, $x, y, z \in X$; $\Sigma = \{\Omega, \Pi\}$, mit

$\Omega = \{f/2, g/3, a/0, b/0\}$ und $\Pi = \{p/2, q/1, r/0\}$

	Term	Atom	Literal	Klausel	Formel	Nichts
x						
a						
$f(g(a, y, b), x)$						
$g(a, x)$						
$p(g(a, y, b), x)$						
$p(q(a), b)$						
$\neg q(g(a, b, x))$						
$\neg q(g(a, b, x)) \vee a$						
$\neg q(g(a, b, x)) \vee p(a, a)$						
$\forall x(p(x, f(x, x)))$						
$\forall b(p(b, f(b, b)))$						

Beispiele

X Variablenmenge, $x, y, z \in X$; $\Sigma = \{\Omega, \Pi\}$, mit

$\Omega = \{f/2, g/3, a/0, b/0\}$ und $\Pi = \{p/2, q/1, r/0\}$

	Term	Atom	Literal	Klausel	Formel	Nichts
x	ja	nein	nein	nein	nein	nein
a	ja	nein	nein	nein	nein	nein
$f(g(a, y, b), x)$	ja	nein	nein	nein	nein	nein
$g(a, x)$	nein	nein	nein	nein	nein	ja
$p(g(a, y, b), x)$	nein	ja	ja	ja	ja	nein
$p(q(a), b)$	nein	nein	nein	nein	nein	ja
$\neg q(g(a, b, x))$	nein	nein	ja	ja	ja	nein
$\neg q(g(a, b, x)) \vee a$	nein	nein	nein	nein	nein	ja
$\neg q(g(a, b, x)) \vee p(a, a)$	nein	nein	nein	ja	ja	nein
$\forall x(p(x, f(x, x)))$	nein	nein	nein	nein	ja	nein
$\forall b(p(b, f(b, b)))$	nein	nein	nein	nein	nein	ja

Beispiel: Tante Agatha

Jemand, der in Schloss Dreadbury wohnt, hat Tante Agatha ermordet. Agatha, ihr Butler und ihr Neffe Charles waren die einzigen Bewohner von Schloss Dreadbury. Ein Mörder hasst immer sein Opfer und ist niemals reicher als sein Opfer. Charles hasst niemanden, den Tante Agatha gehasst hat. Agatha hat jeden gehasst außer ihrem Butler. Der Butler hasst jeden, der nicht reicher ist als Tante Agatha. Der Butler hasst jeden, den Tante Agatha gehasst hat. Niemand hasst jeden. Agatha war nicht der Butler.

Wer hat Tante Agatha ermordet?

Beispiel: Tante Agatha

Jemand, der in Schloss Dreadbury wohnt, hat Tante Agatha ermordet.

Agatha, ihr Butler und ihr Neffe Charles waren die einzigen Bewohner von Schloss Dreadbury.

Beispiel: Tante Agatha

Jemand, der in Schloss Dreadbury wohnt, hat Tante Agatha ermordet.

- $\exists x\ (\text{schlossbewohner}(x) \wedge \text{ermordet}(x, a))$

Agatha, ihr Butler und ihr Neffe Charles waren die einzigen Bewohner von Schloss Dreadbury.

Beispiel: Tante Agatha

Jemand, der in Schloss Dreadbury wohnt, hat Tante Agatha ermordet.

- $\exists x\ (\text{schlossbewohner}(x) \wedge \text{ermordet}(x, \text{a}))$

Agatha, ihr Butler und ihr Neffe Charles waren die einzigen Bewohner von Schloss Dreadbury.

- $\forall x\ (\text{schlossbewohner}(x) \leftrightarrow (x \approx \text{a} \vee x \approx \text{b} \vee x \approx \text{c}))$

Beispiel: Tante Agatha

Ein Mörder hasst immer sein Opfer und ist niemals reicher als sein Opfer.

Charles hasst niemanden, den Tante Agatha gehasst hat.

Agatha hat jeden gehasst außer ihrem Butler.

Beispiel: Tante Agatha

Ein Mörder hasst immer sein Opfer und ist niemals reicher als sein Opfer.

- $\forall x, y\ (\text{ermordet}(x, y) \rightarrow \text{hasst}(x, y))$
  $\forall x, y\ (\text{ermordet}(x, y) \rightarrow \neg\, \text{reicher}(x, y))$

Charles hasst niemanden, den Tante Agatha gehasst hat.

Agatha hat jeden gehasst außer ihrem Butler.

Beispiel: Tante Agatha

Ein Mörder hasst immer sein Opfer und ist niemals reicher als sein Opfer.

- $\forall x, y\ (\text{ermordet}(x, y) \rightarrow \text{hasst}(x, y))$
  $\forall x, y\ (\text{ermordet}(x, y) \rightarrow \neg\, \text{reicher}(x, y))$

Charles hasst niemanden, den Tante Agatha gehasst hat.

- $\forall x\ (\text{hasst}(c, x) \rightarrow \neg\, \text{hasst}(a, x))$

Agatha hat jeden gehasst außer ihrem Butler.

Beispiel: Tante Agatha

Ein Mörder hasst immer sein Opfer und ist niemals reicher als sein Opfer.

- $\forall x, y\ (\text{ermordet}(x, y) \rightarrow \text{hasst}(x, y))$
  $\forall x, y\ (\text{ermordet}(x, y) \rightarrow \neg\, \text{reicher}(x, y))$

Charles hasst niemanden, den Tante Agatha gehasst hat.

- $\forall x\ (\text{hasst}(c, x) \rightarrow \neg\, \text{hasst}(a, x))$

Agatha hat jeden gehasst außer ihrem Butler.

- $\forall x\ (\neg\, \text{hasst}(a, x) \leftrightarrow x \approx b)$

Beispiel: Tante Agatha

Der Butler hasst jeden, der nicht reicher ist als Tante Agatha.

- $\forall x\ (\neg\, \mathrm{reicher}(x, \mathrm{a}) \rightarrow \mathrm{hasst}(\mathrm{b}, x))$

Der Butler hasst jeden, den Tante Agatha gehasst hat.

- $\forall x\ (\mathrm{hasst}(\mathrm{a}, x) \rightarrow \mathrm{hasst}(\mathrm{b}, x))$

Niemand hasst jeden.

- $\forall x\ \exists y\ (\neg\, \mathrm{hasst}(x, y))$

Agatha war nicht der Butler.

- $\neg\, \mathrm{a} \approx \mathrm{b}$

Beispiel: Arithmetik

$$\Sigma_{PA} = (\Omega_{PA}, \Pi_{PA})$$
$$\Omega_{PA} = \{0/0, +/2, */2, s/1\}$$
$$\Pi_{PA} = \{\leq /2, < /2, \approx /2\}$$
$+, *$ infix; $* >_p +$

Formelbeispiele über dieser Signatur sind

$$\forall x, y(x \leq y \leftrightarrow \exists z(x + z \approx y))$$
$$\exists x \forall y(x + y \approx y)$$
$$\forall x, y(x * s(y) \approx x * y + x)$$
$$\forall x, y(s(x) \approx s(y) \rightarrow x \approx y)$$
$$\forall x \exists y\ x < y$$

Gebundene und freie Variablen

Definitionen:

- In QxF, $Q \in \{\exists, \forall\}$, heißt F der Bindungsbereich des Quantors Qx.
- Ein Auftreten einer Variablen x heißt gebunden, wenn es zum Bindungsbereich eines Quantors Qx gehört.
- Alle anderen Auftreten von Variablen heißen frei.

Formeln ohne freie Variablen heißen Satzformen.

Variablenfreie Formeln heißen Grundformeln.

Beispiel

$$\overbrace{\forall y \quad (\overbrace{\forall x \quad p(x)}^{\textit{Bind.}} \quad \rightarrow \quad q(x,y))}^{\textit{Bindungsbereich}}$$

Beispiel

$$p(z) \to \forall x\,(q(x,z) \land \exists z\, r(y,z))$$

- x gebunden
- y frei
- z frei und gebunden

Substitution

Mit $F[s/x]$ bezeichnen wir das Resultat der Substitution aller freien Auftreten von x in F durch den Term s. $F[s/x]$ sei durch strukturelle Induktion über den Aufbau von F wie folgt definiert:

$$
\begin{aligned}
x[s/x] &= s \\
x'[s/x] &= x' \;;\quad \text{falls } x' \neq x \\
f(s_1, \ldots, s_n)[s/x] &= f(s_1[s/x], \ldots, s_n[s/x]) \\
\bot[s/x] &= \bot \\
\top[s/x] &= \top \\
p(s_1, \ldots, s_n)[s/x] &= p(s_1[s/x], \ldots, s_n[s/x]) \\
(u \approx v)[s/x] &= (u[s/x] \approx v[s/x]) \\
\neg F[s/x] &= \neg(F[s/x]) \\
(F \rho G)[s/x] &= (F[s/x] \rho G[s/x]) \;;\quad \text{für alle binären Junktoren } \rho \in \{\wedge, \vee, \rightarrow, \leftrightarrow\} \\
(QyF)[s/x] &= Qz((F[z/y])[s/x]) \;;\quad z \text{ neue Variable}
\end{aligned}
$$

Beispiel

Terme:

$\Sigma = (\Omega, \Pi)$, mit $\Omega = \{f/1, g/2, a/0\}$, $\Pi = \{p/2, \approx /2\}$

$x, y, z, u \in X$

$$g(f(x), g(f(x), z)) \quad [g(y, u)/x]$$
$$= g(f(g(y, u)), g(f(g(y, u)), z))$$

Beispiel

Atome:

$\Sigma = (\Omega, \Pi)$, mit $\Omega = \{f/1, g/2, a/0\}$, $\Pi = \{p/2, \approx /2\}$

$x, y, z, u \in X$

$$p(g(f(x), g(f(x), z)),\ y)\ \ [g(y,u)/x]$$
$$= p(g(f(g(y,u)), g(f(g(y,u)), z)),\ y)$$

Beispiel

Formeln ohne Quantoren:

$\Sigma = (\Omega, \Pi)$, mit $\Omega = \{f/1, g/2, a/0\}$, $\Pi = \{p/2, \approx /2\}$

$x, y, z, u \in X$

$$p(g(f(x), g(f(x), z)),\ y) \quad \wedge \quad (g(x, y) \approx f(g(z, z))) \quad [g(y, u)/x]$$

$$= p(g(f(g(y, u)), g(f(g(y, u)), z)), y) \quad \wedge \quad (g(g(y, u), y) \approx f(g(z, z)))$$

Formeln mit Quantoren: Problematik

$$(QyF)[s/x] = Qz((F[z/y])[s/x]) \; ; \quad z \text{ neue Variable}$$

Der Grund für die Umbenennung der gebundenen Variablen y in eine neue „unbenutzte“ Variable z ist die Vermeidung des Einfangens freier Variablen in s.

Sollte y in s auftreten, wären sonst diese Auftreten nach erfolgter Substitution gebunden.

Beispiel

Formeln mit Quantoren:

$\Sigma = (\Omega, \Pi)$, mit $\Omega = \{f/1, g/2, a/0\}$, $\Pi = \{p/2, \approx /2\}$

$$(\forall x(p(x, g(f(x), y))) \ \wedge \ \exists z(x \approx g(y, z))) \quad [g(y, z)/x]$$

Beispiel

Formeln mit Quantoren:

$\Sigma = (\Omega, \Pi)$, mit $\Omega = \{f/1, g/2, a/0\}$, $\Pi = \{p/2, \approx /2\}$

$$(\forall x(p(x, g(f(x), y))) \ \wedge \ \exists z(x \approx g(y, z)) \ ) \quad [g(y, z)/x]$$

$$= \forall x(p(x, g(f(x), y)))[g(y, z)/x] \ \wedge \ \exists z(x \approx g(y, z))[g(y, z)/x]$$

Beispiel

Formeln mit Quantoren:

$\Sigma = (\Omega, \Pi)$, mit $\Omega = \{f/1, g/2, a/0\}$, $\Pi = \{p/2, \approx /2\}$

$$
\begin{aligned}
&(\forall x(p(x, g(f(x), y))) \ \wedge \ \exists z(x \approx g(y, z)) \ ) \quad [g(y, z)/x] \\
&\quad = \forall x(p(x, g(f(x), y)))[g(y, z)/x] \ \wedge \ \exists z(x \approx g(y, z))[g(y, z)/x] \\
&\quad = \forall v(p(x, g(f(x), y))\,[v/x])[g(y, z)/x] \wedge \exists u(((x \approx g(y, z))[u/z])[g(y, z)/x])
\end{aligned}
$$

Beispiel

Formeln mit Quantoren:

$\Sigma = (\Omega, \Pi)$, mit $\Omega = \{f/1, g/2, a/0\}$, $\Pi = \{p/2, \approx /2\}$

$$
\begin{aligned}
&(\forall x(p(x, g(f(x), y))) \ \wedge \ \exists z(x \approx g(y, z)) \ ) \quad [g(y, z)/x] \\
&\quad = \forall x(p(x, g(f(x), y)))[g(y, z)/x] \ \wedge \ \exists z(x \approx g(y, z))[g(y, z)/x] \\
&\quad = \forall v(p(x, g(f(x), y))\,[v/x])[g(y, z)/x] \wedge \exists u(((x \approx g(y, z))[u/z])[g(y, z)/x]) \\
&\quad = \forall v(p(v, g(f(v), y)))[g(y, z)/x] \ \wedge \ \exists u((x \approx g(y, u))[g(y, z)/x])
\end{aligned}
$$

Beispiel

Formeln mit Quantoren:

$\Sigma = (\Omega, \Pi)$, mit $\Omega = \{f/1, g/2, a/0\}$, $\Pi = \{p/2, \approx /2\}$

$$
\begin{aligned}
&(\forall x(p(x, g(f(x), y))) \ \wedge \ \exists z(x \approx g(y, z)) \ ) \quad [g(y, z)/x] \\
&\quad = \forall x(p(x, g(f(x), y)))[g(y, z)/x] \ \wedge \ \exists z(x \approx g(y, z))[g(y, z)/x] \\
&\quad = \forall v(p(x, g(f(x), y))\,[v/x])[g(y, z)/x] \wedge \exists u(((x \approx g(y, z))[u/z])[g(y, z)/x]) \\
&\quad = \forall v(p(v, g(f(v), y)))[g(y, z)/x] \ \wedge \ \exists u((x \approx g(y, u))[g(y, z)/x]) \\
&\quad = \forall v(p(v, g(f(v), y))) \ \wedge \ \exists u(g(y, z) \approx g(y, u))
\end{aligned}
$$

Zusammenfassung

"And thus, dear students, we have arrived at the formula for understanding women."

Logik für Informatiker

Vorlesung 9: Prädikatenlogik

Babeş-Bolyai Universität, Department für Informatik, Cluj-Napoca
csacarea@cs.ubbcluj.ro

Prädikatenlogik

- Ausdrucksreichere Logik
- Signatur $\Sigma = (\Omega, \Pi)$
- Konstanten, Funktionen, Prädikate, Variablen
- Terme, Atome
- Formeln

Gebundene und freie Variablen

Definitionen:

- In QxF, $Q \in \{\exists, \forall\}$, heißt F der Bindungsbereich des Quantors Qx.
- Ein Auftreten einer Variablen x heißt gebunden, wenn es zum Bindungsbereich eines Quantors Qx gehört.
- Alle anderen Auftreten von Variablen heißen frei.

Formeln ohne freie Variablen heißen Satzformen.

Variablenfreie Formeln heißen Grundformeln.

Beispiel

$$\overbrace{\forall y \quad (\overbrace{\forall x \quad p(x)}^{\textit{Bind.}} \quad \rightarrow \quad q(x,y))}^{\textit{Bindungsbereich}}$$

Beispiel

$$p(z) \to \forall x\,(q(x,z) \land \exists z\, r(y,z))$$

- x gebunden
- y frei
- z frei und gebunden

Substitution

Mit $F[s/x]$ bezeichnen wir das Resultat der Substitution aller freien Auftreten von x in F durch den Term s. $F[s/x]$ sei durch strukturelle Induktion über den Aufbau von F wie folgt definiert:

$$
\begin{aligned}
x[s/x] &= s \\
x'[s/x] &= x' \;;\quad \text{falls } x' \neq x \\
f(s_1, \ldots, s_n)[s/x] &= f(s_1[s/x], \ldots, s_n[s/x]) \\
\bot[s/x] &= \bot \\
\top[s/x] &= \top \\
p(s_1, \ldots, s_n)[s/x] &= p(s_1[s/x], \ldots, s_n[s/x]) \\
(u \approx v)[s/x] &= (u[s/x] \approx v[s/x]) \\
\neg F[s/x] &= \neg(F[s/x]) \\
(F \rho G)[s/x] &= (F[s/x] \rho G[s/x]) \;;\quad \text{für alle binären Junktoren } \rho \in \{\wedge, \vee, \rightarrow, \leftrightarrow\} \\
(QyF)[s/x] &= Qz((F[z/y])[s/x]) \;;\quad z \text{ neue Variable}
\end{aligned}
$$

Formeln mit Quantoren: Problematik

$$(QyF)[s/x] = Qz((F[z/y])[s/x]) \; ; \quad z \text{ neue Variable}$$

Der Grund für die Umbenennung der gebundenen Variablen y in eine neue „unbenutzte“ Variable z ist die Vermeidung des Einfangens freier Variablen in s.

Sollte y in s auftreten, wären sonst diese Auftreten nach erfolgter Substitution gebunden.

Substitution allgemein

Definition. Substitutionen sind Abbildungen

$$\sigma : X \to T_\Sigma(X),$$

so dass der Bereich von σ, d.h. die Menge

$$dom(\sigma) = \{x \in X \mid \sigma(x) \neq x\},$$

endlich ist. Die Menge der eingeführten Variablen, d.h. der Variablen die in einem der Terme $\sigma(x)$, für $x \in dom(\sigma)$, auftreten, wird mit $codom(\sigma)$ bezeichnet.

Substitutionen schreiben wir auch als $[s_1/x_1, \ldots, s_n/x_n]$, x_i pw. verschieden, und meinen dann die Abbildung

$$[s_1/x_1, \ldots, s_n/x_n](y) = \begin{cases} s_i, & \text{falls } y = x_i \\ y, & \text{sonst} \end{cases}$$

Ab jetzt schreiben wir für die Applikation $\sigma(x)$ von Substitutionen $x\sigma$.

Anwendung einer Substitution

„Homomorphe“ Fortsetzung von σ auf Terme und Formeln:

$$
\begin{aligned}
f(s_1, \ldots, s_n)\sigma &= f(s_1\sigma, \ldots, s_n\sigma) \\
\bot\sigma &= \bot \\
\top\sigma &= \top \\
p(s_1, \ldots, s_n)\sigma &= p(s_1\sigma, \ldots, s_n\sigma) \\
(u \approx v)\sigma &= (u\sigma \approx v\sigma) \\
\neg F\sigma &= \neg(F\sigma) \\
(F\rho G)\sigma &= (F\sigma\rho G\sigma)\ ; \quad \text{für alle binären Junktoren } \rho \\
(QxF)\sigma &= Qz((F[z/x])\sigma); \ \text{wobei } z \text{ „neue“ Variable}
\end{aligned}
$$

Abänderung einer Substitution σ an x zu t:

$$
\sigma[x \to t](y) = \begin{cases} t, & \text{falls } y = x \\ \sigma(y), & \text{sonst} \end{cases}
$$

Beispiel

Anwendung einer Substitution auf Terme

$\Sigma = (\Omega, \Pi)$, mit $\Omega = \{f/1, g/2, a/0\}$, $\Pi = \{p/2, \approx /2\}$

$$g(f(x), g(f(x), z)) \quad [g(y,u)/x, f(x)/y, a/z]$$
$$= g(f(g(y,u)), g(f(g(y,u)), a))$$

Beispiel

Anwendung einer Substitution auf Atome

$\Sigma = (\Omega, \Pi)$, mit $\Omega = \{f/1, g/2, a/0\}$, $\Pi = \{p/2, \approx /2\}$

$$p(g(f(x), g(f(x), z)),\ y)\ \ [g(y,u)/x, f(x)/y, a/z]$$
$$= p(g(f(g(y,u)), g(f(g(y,u)), a)), f(x))$$

Beispiel

Anwendung einer Substitution auf Formeln ohne Quantoren

$\Sigma = (\Omega, \Pi)$, mit $\Omega = \{f/1, g/2, a/0\}$, $\Pi = \{p/2, \approx /2\}$

$$p(g(f(x), g(f(x), z)), y) \ \wedge \ (g(x, y) \approx f(g(z, z))) \ [g(y, u)/x, f(x)/y, a/z]$$
$$= p(g(f(g(y, u)), g(f(g(y, u)), a)), f(x)) \ \wedge \ (g(g(y, u), f(x)) \approx f(g(a, a)))$$

Beispiel

Anwendung einer Substitution auf Formeln mit Quantoren

$\Sigma = (\Omega, \Pi)$, mit $\Omega = \{f/1, g/2, a/0\}$, $\Pi = \{p/2, \approx /2\}$

$$(\forall x(p(x, g(f(x), y))) \wedge \exists z(x \approx g(y, z)))[g(y, z)/x, f(x)/y, a/z]$$

Beispiel

Anwendung einer Substitution auf Formeln mit Quantoren

$\Sigma = (\Omega, \Pi)$, mit $\Omega = \{f/1, g/2, a/0\}$, $\Pi = \{p/2, \approx /2\}$

$$(\forall x(p(x, g(f(x), y))) \wedge \exists z(x \approx g(y, z)))[g(y, z)/x, f(x)/y, a/z]$$

$$= \forall x(p(x, g(f(x), y)))[g(y, z)/x, f(x)/y, a/z] \ \wedge$$
$$\exists z(x \approx g(y, z))[g(y, z)/x, f(x)/y, a/z]$$

Beispiel

Anwendung einer Substitution auf Formeln mit Quantoren

$\Sigma = (\Omega, \Pi)$, mit $\Omega = \{f/1, g/2, a/0\}$, $\Pi = \{p/2, \approx /2\}$

$$(\forall x(p(x, g(f(x), y))) \wedge \exists z(x \approx g(y, z)))[g(y, z)/x, f(x)/y, a/z]$$

$$\begin{aligned} &= \forall x(p(x, g(f(x), y)))[g(y, z)/x, f(x)/y, a/z] \ \wedge \\ &\quad \exists z(x \approx g(y, z))[g(y, z)/x, f(x)/y, a/z] \\ &= \forall v((p(x, g(f(x), y))[v/x])[g(y, z)/x, f(x)/y, a/z]) \ \wedge \\ &\quad \exists u(((x \approx g(y, z))[u/z])[g(y, z)/x, f(x)/y, a/z]) \end{aligned}$$

Beispiel

Anwendung einer Substitution auf Formeln mit Quantoren

$\Sigma = (\Omega, \Pi)$, mit $\Omega = \{f/1, g/2, a/0\}$, $\Pi = \{p/2, \approx /2\}$

$$(\forall x(p(x, g(f(x), y))) \wedge \exists z(x \approx g(y, z)))[g(y, z)/x, f(x)/y, a/z]$$

$$= \forall x(p(x, g(f(x), y)))[g(y, z)/x, f(x)/y, a/z] \ \wedge \ \exists z(x \approx g(y, z))[g(y, z)/x, f(x)/y, a/z]$$

$$= \forall v((p(x, g(f(x), y))[v/x])[g(y, z)/x, f(x)/y, a/z]) \ \wedge \ \exists u(((x \approx g(y, z))[u/z])[g(y, z)/x, f(x)/y, a/z])$$

$$= \forall v(p(v, g(f(v), f(x)))) \ \wedge \ \exists u((x \approx g(y, u))[g(y, z)/x, f(x)/y, a/z])$$

Beispiel

Anwendung einer Substitution auf Formeln mit Quantoren

$\Sigma = (\Omega, \Pi)$, mit $\Omega = \{f/1, g/2, a/0\}$, $\Pi = \{p/2, \approx /2\}$

$$(\forall x(p(x, g(f(x), y))) \wedge \exists z(x \approx g(y, z)))[g(y, z)/x, f(x)/y, a/z]$$

$$= \forall x(p(x, g(f(x), y)))[g(y, z)/x, f(x)/y, a/z] \ \wedge \ \exists z(x \approx g(y, z))[g(y, z)/x, f(x)/y, a/z]$$

$$= \forall v((p(x, g(f(x), y))[v/x])[g(y, z)/x, f(x)/y, a/z]) \ \wedge \ \exists u(((x \approx g(y, z))[u/z])[g(y, z)/x, f(x)/y, a/z])$$

$$= \forall v(p(v, g(f(v), f(x)))) \ \wedge \ \exists u((x \approx g(y, u))[g(y, z)/x, f(x)/y, a/z])$$

$$= \forall v(p(v, g(f(v), f(x)))) \ \wedge \ \exists u(g(y, z) \approx g(f(x), u))$$

Semantik der Prädikatenlogik

- Semantik geben bedeutet für logische Systeme, einen Begriff von Wahrheit für Formeln zu definieren. Das hier für die Prädikatenlogik zu definierende Konzept geht auf Tarski zurück.
- In der klassischen Logik (zurückgehend auf Aristoteles) gibt es *nur* die zwei Wahrheitswerte wahr und falsch, die wir mit 1 und 0 bezeichnen.

Strukturen

Definition.

Eine Σ-Struktur (bzw. Σ-Interpretation bzw. Σ-Modell) ist ein Tripel

$$\mathcal{A} = (U,\ (f_{\mathcal{A}} : U^n \to U)_{f/n \in \Omega},\ (p_{\mathcal{A}} \subseteq U^m)_{p/m \in \Pi})$$

wobei $U \neq \emptyset$ eine Menge, genannt Universum von $\mathcal{A}$.

Oft identifizieren wir U mit $\mathcal{A}$, wenn die Interpretation der Funktions- und Prädikatensymbole eindeutig aus dem Kontext hervorgeht.

Mit Σ-Str bezeichnen wir die Menge aller Σ-Strukturen.

Strukturen

Beispiel:

$\Sigma = (\{+/2, 0/0\}, \{\leq, \approx\})$

$\mathcal{N} = (\mathbb{N}, \{0_{\mathcal{N}}, +_{\mathcal{N}} : \mathbb{N} \times \mathbb{N} \to \mathbb{N}\}, \{\leq_{\mathcal{N}} \subseteq \mathbb{N} \times \mathbb{N}, \approx_{\mathcal{N}}\})$

$0_{\mathcal{N}} = 0 \in \mathbb{N},$
$+_{\mathcal{N}}(n_1, n_1) = n_1 + n_2 \in \mathbb{N}$

$\leq_{\mathcal{N}} = \{(n_1, n_2) \mid n_1 \leq n_2\}$

Strukturen

Beispiel:

$\Sigma = (\{+/2, 0/0\}, \{\leq, \approx\})$

Eine andere Σ-Struktur:

$\mathcal{A} = (\mathbb{N}, \{0_{\mathcal{A}}, +_{\mathcal{A}} : \mathbb{N} \times \mathbb{N} \to \mathbb{N}\}, \{\leq_{\mathcal{A}} \subseteq \mathbb{N} \times \mathbb{N}, \approx_{\mathcal{N}}\})$

$$0_{\mathcal{A}} = 1 \in \mathbb{N},$$

$$+_{\mathcal{A}}(n_1, n_2) = \begin{cases} 1 & \text{wenn } n_2 = 0 \\ n_1^{n_2} & \text{wenn } n_2 \neq 0 \end{cases}$$

$$\leq_{\mathcal{A}} = \{(n_1, n_2) \mid n_1 = n_2^2\}$$

Strukturen

Beispiel:

$\Sigma = (\{+/2, 0/0\}, \{\leq, \approx\})$

Eine dritte Σ-Struktur:

$\mathcal{B} = (\{a, b\}, \{0_{\mathcal{B}}, +_{\mathcal{B}}{:}\{a, b\}\times\{a, b\}\to\{a, b\}\}, \{\leq_{\mathcal{B}}\subseteq \{a, b\}\times\{a, b\}, \approx_{\mathcal{B}}\})$

$$0_{\mathcal{B}} = a \in \{a, b\},$$
$$+_{\mathcal{B}}(a, a) = a; \quad +_{\mathcal{B}}(a, b) = b;$$
$$+_{\mathcal{B}}(b, a) = b; \quad +_{\mathcal{B}}(b, b) = b;$$

$$\leq_{\mathcal{B}} = \{(a, a), (a, b), (b, b)\}$$

Valuationen

Variablen für sich haben keine Bedeutung. Hierfür müssen Wertebelegungen (Valuationen) explizit oder implizit aus dem Kontext zur Verfügung stehen.

Definition.

Unter einer (Variablen-) Belegung oder einer Valuation (über einer Σ-Struktur A) versteht man eine Abbildung

$$\beta \colon X \to U.$$

Wert eines Terms in A bzgl. β

Induktive Definition:

$$\mathcal{A}(\beta)(x) = \beta(x), \qquad x \in X$$
$$\mathcal{A}(\beta)(f(s_1, \ldots, s_n)) = f_{\mathcal{A}}(\mathcal{A}(\beta)(s_1), \ldots, \mathcal{A}(\beta)(s_n)), \qquad f/n \in \Omega$$

Wert eines Terms in $\mathcal{A}$ bzgl. β, $\mathcal{A}(\beta)(t)$:

- Falls $t = x \in X$: $\mathcal{A}(\beta)(t) = \beta(x)$
- Falls $t = c$ eine Konstante: $\mathcal{A}(\beta)(t) = c_{\mathcal{A}}$
- Falls $t = f(t_1, \ldots, t_n)$:
  $\mathcal{A}(\beta)(t) = f_{\mathcal{A}}(\mathcal{A}(\beta)(t_1), \ldots, \mathcal{A}(\beta)(t_n))$

Wert eines Terms in A bzgl. β

Beispiel:

$\Sigma = (\{+/2, 0/0\}, \{\leq, \approx\})$

$\mathcal{N} = (\mathbb{N}, \{0_{\mathcal{N}}, +_{\mathcal{N}} : \mathbb{N} \times \mathbb{N} \to \mathbb{N}\}, \{\leq_{\mathcal{N}} \subseteq \mathbb{N} \times \mathbb{N}, \approx_{\mathcal{N}}\})$

$0_{\mathcal{N}} = 0 \in \mathbb{N},$
$+_{\mathcal{N}}(n_1, n_1) = n_1 + n_2 \in \mathbb{N}$

$\beta : \{x, y, z\} \to \mathbb{N}$ mit $\beta(x) = 5, \beta(y) = 10, \beta(z) = 3$

$$\begin{aligned}
&\mathcal{N}(\beta)((x + (y + z)) + (z + 0)) = \\
&\quad = (\beta(x) +_{\mathcal{N}} (\beta(y) +_{\mathcal{N}} \beta(z))) +_{\mathcal{N}} (\beta(z) +_{\mathcal{N}} 0_{\mathcal{N}}) = \\
&\quad = (5 + (10 + 3)) + (3 + 0) = 21
\end{aligned}$$

Wahrheitswert einer Formel in A bzgl. β

Die Menge der Wahrheitswerte sei $\{0, 1\}$.

$\mathcal{A}(\beta) : \mathrm{For}_\Sigma \to \{0, 1\}$ wird induktiv über Aufbau von F wie folgt definiert:

$$\mathcal{A}(\beta)(\bot) = 0$$
$$\mathcal{A}(\beta)(\top) = 1$$

Wahrheitswert einer Formel in A bzgl. β

$$\mathcal{A}(\beta)(p(s_1, \ldots, s_n)) = 1 \quad \text{g.d.w.} \quad (\mathcal{A}(\beta)(s_1), \ldots, \mathcal{A}(\beta)(s_n)) \in p_{\mathcal{A}}$$

$$\mathcal{A}(\beta)(s \approx t) = 1 \quad \text{g.d.w.} \quad \mathcal{A}(\beta)(s) = \mathcal{A}(\beta)(t)$$

Wahrheitswert einer Formel in A bzgl. β

$$\mathcal{A}(\beta)(\neg F) = 1 \quad \text{g.d.w.} \quad \mathcal{A}(\beta)(F) = 0$$

$$\mathcal{A}(\beta)(F \rho G) = \rho_b(\mathcal{A}(\beta)(F), \mathcal{A}(\beta)(G))$$

mit ρ_b die ρ zugeordnete Boolesche Funktion

Wahrheitswert einer Formel in A bzgl. β

$$\mathcal{A}(\beta)(\neg F) = 1 \quad \text{g.d.w.} \quad \mathcal{A}(\beta)(F) = 0$$

$$\mathcal{A}(\beta)(F \rho G) = \rho_b(\mathcal{A}(\beta)(F), \mathcal{A}(\beta)(G))$$

mit ρ_b die ρ zugeordnete Boolesche Funktion

Erklärung:

- $\mathcal{A}(\beta)(\neg F) = \neg_b \mathcal{A}(\beta)(F)$, wobei $\neg_b$ die Negation auf den Wahrheitswerten $\{0, 1\}$ ist, mit Wahrheitstabelle:

w	$\neg_b w$
0	1
1	0

Wahrheitswert einer Formel in A bzgl. β

$$\mathcal{A}(\beta)(\neg F) = 1 \quad \text{g.d.w.} \quad \mathcal{A}(\beta)(F) = 0$$

$$\mathcal{A}(\beta)(F \rho G) = \rho_b(\mathcal{A}(\beta)(F), \mathcal{A}(\beta)(G))$$

mit ρ_b die ρ zugeordnete Boolesche Funktion

Erklärung:

- $\mathcal{A}(\beta)(F \wedge G) = \mathcal{A}(\beta)(F) \wedge_b \mathcal{A}(\beta)(G)$, wobei $\wedge_b$ die Konjunktion auf den Wahrheitswerten $\{0, 1\}$ ist, mit Wahrheitstabelle:

$\wedge_b$	0	1
0	0	0
1	0	1

Wahrheitswert einer Formel in A bzgl. β

$$\mathcal{A}(\beta)(\neg F) = 1 \quad \text{g.d.w.} \quad \mathcal{A}(\beta)(F) = 0$$

$$\mathcal{A}(\beta)(F \rho G) = \rho_b(\mathcal{A}(\beta)(F), \mathcal{A}(\beta)(G))$$

mit ρ_b die ρ zugeordnete Boolesche Funktion

Erklärung:

- $\mathcal{A}(\beta)(F \vee G) = \mathcal{A}(\beta)(F) \vee_b \mathcal{A}(\beta)(G)$, wobei $\vee_b$ die Disjunktion auf den Wahrheitswerten $\{0, 1\}$ ist, mit Wahrheitstabelle:

$\vee_b$	0	1
0	0	1
1	1	1

Wahrheitswert einer Formel in A bzgl. β

$$\mathcal{A}(\beta)(\neg F) = 1 \quad \text{g.d.w.} \quad \mathcal{A}(\beta)(F) = 0$$

$$\mathcal{A}(\beta)(F \rho G) = \rho_b(\mathcal{A}(\beta)(F), \mathcal{A}(\beta)(G))$$

mit ρ_b die ρ zugeordnete Boolesche Funktion

Erklärung:

- $\mathcal{A}(\beta)(F \to G) = \mathcal{A}(\beta)(F) \to_b \mathcal{A}(\beta)(G)$, wobei $\to_b$ die Implikation auf den Wahrheitswerten $\{0, 1\}$ ist, mit Wahrheitstabelle:

$\to_b$	0	1
0	1	1
1	0	1

Wahrheitswert einer Formel in A bzgl. β

$$\mathcal{A}(\beta)(\neg F) = 1 \quad \text{g.d.w.} \quad \mathcal{A}(\beta)(F) = 0$$

$$\mathcal{A}(\beta)(F \rho G) = \rho_b(\mathcal{A}(\beta)(F), \mathcal{A}(\beta)(G))$$

mit ρ_b die ρ zugeordnete Boolesche Funktion

Erklärung:

- $\mathcal{A}(\beta)(F \leftrightarrow G) = \mathcal{A}(\beta)(F) \leftrightarrow_b \mathcal{A}(\beta)(G)$, wobei $\leftrightarrow_b$ die Äquivalenz auf den Wahrheitswerten $\{0, 1\}$ ist, mit Wahrheitstabelle:

$\leftrightarrow_b$	0	1
0	1	0
1	0	1

Beispiel

$\Sigma = (\Omega, \Pi), X \quad \Omega = \{0/0, s/1, +/2, */2\}$

$\Pi = \{\leq /2, < /2, \text{gerade}/1, \text{ungerade}/1\}$ mit Gleichheit $\approx$.

$$\begin{aligned}
\mathcal{A} = (\mathbb{N}, \{ & +_{\mathcal{A}}, *_{\mathcal{A}} : \mathbb{N} \times \mathbb{N} \to \mathbb{N}, \\
& n_1 +_{\mathcal{A}} n_2 = n_1 + n_2 \\
& n_1 *_{\mathcal{A}} n_2 = n_1 \cdot n_2 \\
& s_{\mathcal{A}} : \mathbb{N} \to \mathbb{N}, \\
& s(n) = n + 1 \\
& 0_{\mathcal{A}} = 0 \in \mathbb{N}\}
\end{aligned}$$

$$\begin{aligned}
& \{\leq, <_{\mathcal{A}} \subseteq \mathbb{N} \times \mathbb{N} \ ) \\
& (n_1, n_2) \in \leq_{\mathcal{A}} \text{ iff } n_1 \leq n_2 \text{ in } \mathbb{N} \\
& (n_1, n_2) \in <_{\mathcal{A}} \text{ iff } n_1 < n_2 \text{ in } \mathbb{N} \\
& \text{gerade}_{\mathcal{A}}, \text{ungerade}_{\mathcal{A}} \subseteq \mathbb{N}\} \\
& \text{gerade}_{\mathcal{A}} = \{0, 2, 4, \ldots, 2k \ldots\} \\
& \text{ungerade}_{\mathcal{A}} = \{1, 3, 5, \ldots, 2k + 1 \ldots\},
\end{aligned}$$

Konvention: Auf den nächsten Folien wird der Unterschied zwischen den natürlichen Zahlen 0, 1 und den Wahrheitswerten 0 (falsch) und 1 (wahr) deutlich gemacht, indem wir die Wahrheitswerte in orangener Farbe schreiben.

Nota Bene:

- Der Wert eines Termes t in $\mathcal{A}$ bzgl. β ist ein Element in das Universum von $\mathcal{A}$.
- Der Wahrheitswert einer Formel F in $\mathcal{A}$ bzgl. β ist ein Wahrheitswert (0 oder 1).

Beispiel

$\Sigma = (\Omega, \Pi), X \quad \Omega = \{0/0, s/1, +/2, */2\}$

$\Pi = \{\leq /2, < /2, \text{gerade}/1, \text{ungerade}/1\}$ mit Gleichheit $\approx$.

$\mathcal{A} = (\mathbb{N}, \{+_{\mathcal{A}}, *_{\mathcal{A}} : \mathbb{N} \times \mathbb{N} \to \mathbb{N},$
$n_1 +_{\mathcal{A}} n_2 = n_1 + n_2$
$n_1 *_{\mathcal{A}} n_2 = n_1 \cdot n_2$

$\{\leq, <_{\mathcal{A}} \subseteq \mathbb{N} \times \mathbb{N}\ )$
$(n_1, n_2) \in \leq_{\mathcal{A}}$ iff $n_1 \leq n_2$ in $\mathbb{N}$
$(n_1, n_2) \in <_{\mathcal{A}}$ iff $n_1 < n_2$ in $\mathbb{N}$

$s_{\mathcal{A}} : \mathbb{N} \to \mathbb{N},$
$s(n) = n + 1$
$0_{\mathcal{A}} = 0 \in \mathbb{N}\}$

$\text{gerade}_{\mathcal{A}}, \text{ungerade}_{\mathcal{A}} \subseteq \mathbb{N}\}$
$\text{gerade}_{\mathcal{A}} = \{0, 2, 4, \ldots, 2k \ldots\}$
$\text{ungerade}_{\mathcal{A}} = \{1, 3, 5, \ldots, 2k + 1 \ldots\},$

$\beta : X \to \mathbb{N}$ mit $\beta(x) = 1, \beta(y) = 4$

(1) $\mathcal{A}(\beta)(\text{gerade}(x)) = 0$	Erklärung:	$\mathcal{A}(\beta)(x) = \beta(x) = 1 \notin \text{gerade}_{\mathcal{A}}$
(2) $\mathcal{A}(\beta))(s(s(x) + s(0)) \approx y) = 1$	Erklärung:	$\mathcal{A}(\beta)(s(s(x) + s(0))) = s_{\mathcal{A}}(s_{\mathcal{A}}(\beta(x)) +_{\mathcal{A}} s_{\mathcal{A}}(0_{\mathcal{A}})$
		$= ((1+1)+(0+1))+1 = 4$
		$\mathcal{A}(\beta)(y) = 4$
		und $4 = 4$
(3) $\mathcal{A}(\beta))(x \approx y) = 0$	Erklärung:	$\mathcal{A}(\beta)(x) = 1; \quad \mathcal{A}(\beta)(y) = 4, \quad 1 \neq 4$

Beispiel

$\Sigma = (\Omega, \Pi), X \quad \Omega = \{0/0, s/1, +/2, */2\}$

$\Pi = \{\leq /2, < /2, \text{gerade}/1, \text{ungerade}/1\}$ with equality $\approx$.

$\mathcal{A} = (\mathbb{N}, \{+_{\mathcal{A}}, *_{\mathcal{A}} : \mathbb{N} \times \mathbb{N} \to \mathbb{N},$
$n_1 +_{\mathcal{A}} n_2 = n_1 + n_2$
$n_1 *_{\mathcal{A}} n_2 = n_1 \cdot n_2$

$\{\leq, <_{\mathcal{A}} \subseteq \mathbb{N} \times \mathbb{N}\ )$
$(n_1, n_2) \in \leq_{\mathcal{A}}$ iff $n_1 \leq n_2$ in $\mathbb{N}$
$(n_1, n_2) \in <_{\mathcal{A}}$ iff $n_1 < n_2$ in $\mathbb{N}$

$s_{\mathcal{A}} : \mathbb{N} \to \mathbb{N},$
$s(n) = n + 1$
$0_{\mathcal{A}} = 0 \in \mathbb{N}\}$

$\text{gerade}_{\mathcal{A}}, \text{ungerade}_{\mathcal{A}} \subseteq \mathbb{N}\}$
$\text{gerade}_{\mathcal{A}} = \{0, 2, 4, \ldots, 2k \ldots\}$
$\text{ungerade}_{\mathcal{A}} = \{1, 3, 5, \ldots, 2k + 1 \ldots\},$

$\beta : X \to \mathbb{N}$ mit $\beta(x) = 1, \beta(y) = 4$

(4) $\mathcal{A}(\beta)(\neg\text{gerade}(x)) = \neg_b \mathcal{A}(\beta)(\text{gerade}(x)) = \neg_b 0 = 1$

(5) $\mathcal{A}(\beta)(x \leq s(0) \vee \text{gerade}(x)) = \mathcal{A}(\beta)(x \leq s(0)) \vee_b \mathcal{A}(\beta)(\text{gerade}(x)) = 1 \vee_b 0 = 1$

Erklärung: $\mathcal{A}(\beta)(x \leq s(0)) = 1$: $\mathcal{A}(\beta)(x)=1$, $\mathcal{A}(s(0))=s_{\mathcal{A}}(0_{\mathcal{A}})=0+1=1$, und $(1,1) \in \leq_{\mathcal{A}}$.

(6) $\mathcal{A}(\beta)(y \leq s(0) \to \text{ungerade}(y)) = \mathcal{A}(\beta)(y \leq s(0)) \to_b \mathcal{A}(\beta)(\text{ungerade}(y)) = 0 \to_b 0 = 1$

Erklärung: $\mathcal{A}(\beta)(y \leq s(0))=0$, da $\beta(y)=4$, $\beta(s(0))=1$ und $(4,1) \notin \leq_{\mathcal{A}}$;
$\mathcal{A}(\beta)(\text{ungerade}(y))=0$ da $\beta(y)=4 \notin \text{ungerade}_{\mathcal{A}}$.

Wahrheitswert einer Formel in A bzgl. β

$$\mathcal{A}(\beta)(\forall x F) = \min_{a \in U}\{\mathcal{A}(\beta[x \mapsto a])(F)\} = \begin{cases} 1 & \text{falls } \mathcal{A}(\beta[x \mapsto a])(F) = 1 \text{ für alle } a \in U \\ 0 & \text{sonst} \end{cases}$$

$$\mathcal{A}(\beta)(\exists x F) = \max_{a \in U}\{\mathcal{A}(\beta[x \mapsto a])(F)\} = \begin{cases} 1 & \text{falls } \mathcal{A}(\beta[x \mapsto a])(F) = 1 \text{ für mindestens ein } a \in U \\ 0 & \text{sonst} \end{cases}$$

Für $x \in X$ und $a \in U$ bezeichne $\beta[x \mapsto a] : X \to U$ die Belegung, mit

$$\beta[x \mapsto a](y) := \begin{cases} a & \text{falls } x = y \\ \beta(y) & \text{sonst} \end{cases}$$

Wahrheitswert einer Formel in A bzgl. β

$$\mathcal{A}(\beta)(\forall x F) = \min_{a \in U}\{\mathcal{A}(\beta[x \mapsto a])(F)\} = \begin{cases} 1 & \text{falls } \mathcal{A}(\beta[x \mapsto a])(F) = 1 \text{ für alle } a \in U \\ 0 & \text{sonst} \end{cases}$$

$$\mathcal{A}(\beta)(\exists x F) = \max_{a \in U}\{\mathcal{A}(\beta[x \mapsto a])(F)\} = \begin{cases} 1 & \text{falls } \mathcal{A}(\beta[x \mapsto a])(F) = 1 \text{ für mindestens ein } a \in U \\ 0 & \text{sonst} \end{cases}$$

Für $x \in X$ und $a \in U$ bezeichne $\beta[x \mapsto a] : X \to U$ die Belegung, mit

$$\beta[x \mapsto a](y) := \begin{cases} a & \text{falls } x = y \\ \beta(y) & \text{sonst} \end{cases}$$

Intuition: $\forall$: verallgemeinerte Konjunktion ($\wedge_b$ ist minimum auf $\{0, 1\}$)

$\exists$: verallgemeinerte Disjunktion ($\vee_b$ ist maximum auf $\{0, 1\}$)

- Drei Logiker kommen in eine Bar.
- Wollt ihr alle ein Bier?, fragt die Kellnerin.
- Weiß ich nicht, sagt der erste Logiker.
- Weiß ich nicht, sagt der zweite Logiker.
- Ja!, sagt der dritte Logiker.

- Logiker 1 will ein Bier, aber er weiß nicht, was seine Begleiter wollen, deshalb kann er die Frage weder mit *ja* noch mit *nein* beantworten.
- Logiker 2 kann aus der Antwort von Logiker 1 schließen, dass der Durst auf ein Bier hat. Denn wäre das nicht der Fall, dann könnte er die Frage mit *nein* beantworten. (ein Satz, der mit *Alle* beginnt, wird schon durch eine einzige Ausnahme falsch.)
  Logiker 2 möchte auch gern ein Bier, aber weil er nichts über den Durst von Logiker 3 weiß, muss er auch mit *Weiß ich nicht* antworten.
- Erst Logiker 3 kann eine definitive Antwort geben: Er weiß, dass seine beiden Begleiter ein Bier trinken möchten, er selber möchte auch eins - also antwortet er mit *Ja!*

Beispiel

$\Sigma = (\Omega, \Pi), X \quad \Omega = \{0/0, s/1, +/2, */2\}$

$\Pi = \{\leq /2, < /2, \text{gerade}/1, \text{ungerade}/1\}$ with equality $\approx$.

$\mathcal{A} = (\mathbb{N}, \{+_{\mathcal{A}}, *_{\mathcal{A}} : \mathbb{N} \times \mathbb{N} \to \mathbb{N},$
$n_1 +_{\mathcal{A}} n_2 = n_1 + n_2$
$n_1 *_{\mathcal{A}} n_2 = n_1 \cdot n_2$

$\{\leq, <_{\mathcal{A}} \subseteq \mathbb{N} \times \mathbb{N}$)
$(n_1, n_2) \in \leq_{\mathcal{A}}$ iff $n_1 \leq n_2$ in $\mathbb{N}$
$(n_1, n_2) \in <_{\mathcal{A}}$ iff $n_1 < n_2$ in $\mathbb{N}$

$s_{\mathcal{A}} : \mathbb{N} \to \mathbb{N},$
$s(n) = n + 1$
$0_{\mathcal{A}} = 0 \in \mathbb{N}\}$

$\text{gerade}_{\mathcal{A}}, \text{ungerade}_{\mathcal{A}} \subseteq \mathbb{N}\}$
$\text{gerade}_{\mathcal{A}} = \{0, 2, 4, \ldots, 2k \ldots\}$
$\text{ungerade}_{\mathcal{A}} = \{1, 3, 5, \ldots, 2k + 1 \ldots\},$

$\beta : X \to \mathbb{N}$ mit $\beta(x) = 1, \beta(y) = 4$

(7) $\mathcal{A}(\beta)(\forall x \text{gerade}(x)) = \min_{a \in \mathbb{N}} \mathcal{A}(\beta[x \mapsto a](\text{gerade}(x)) = 0$

Erklärung:
Falls $a = 2k$ so $\beta[x \mapsto a](x) = a \in \text{gerade}_{\mathcal{A}}$. Dann $\mathcal{A}(\beta[x \mapsto a](\text{gerade}(x)) = 1$.
Falls $a = 2k + 1$ so $\beta[x \mapsto a](x) = a \notin \text{gerade}_{\mathcal{A}}$. Dann $\mathcal{A}(\beta[x \mapsto a](\text{gerade}(x)) = 0$.

Es gibt $a \in \mathbb{N}$ mit $\mathcal{A}(\beta[x \mapsto a](\text{gerade}(x)) = 0$.
$\min_{a \in \mathbb{N}} \mathcal{A}(\beta[x \mapsto a](\text{gerade}(x)) = \min\{1, 0\} = 0$

Beispiel

$\Sigma = (\Omega, \Pi), X \quad \Omega = \{0/0, s/1, +/2, */2\}$

$\Pi = \{\leq /2, < /2, \text{gerade}/1, \text{ungerade}/1\}$ with equality $\approx$.

$\mathcal{A} = (\mathbb{N}, \{+_{\mathcal{A}}, *_{\mathcal{A}} : \mathbb{N} \times \mathbb{N} \to \mathbb{N},$
$n_1 +_{\mathcal{A}} n_2 = n_1 + n_2$
$n_1 *_{\mathcal{A}} n_2 = n_1 \cdot n_2$

$\{\leq, <_{\mathcal{A}} \subseteq \mathbb{N} \times \mathbb{N}\ )$
$(n_1, n_2) \in \leq_{\mathcal{A}}$ iff $n_1 \leq n_2$ in $\mathbb{N}$
$(n_1, n_2) \in <_{\mathcal{A}}$ iff $n_1 < n_2$ in $\mathbb{N}$

$s_{\mathcal{A}} : \mathbb{N} \to \mathbb{N},$
$s(n) = n + 1$
$0_{\mathcal{A}} = 0 \in \mathbb{N}\}$

$\text{gerade}_{\mathcal{A}}, \text{ungerade}_{\mathcal{A}} \subseteq \mathbb{N}\}$
$\text{gerade}_{\mathcal{A}} = \{0, 2, 4, \ldots, 2k \ldots\}$
$\text{ungerade}_{\mathcal{A}} = \{1, 3, 5, \ldots, 2k + 1 \ldots\},$

$\beta : X \to \mathbb{N}$ mit $\beta(x) = 1, \beta(y) = 4$

(8) $\mathcal{A}(\beta)(\exists x \text{gerade}(x)) = \max_{a \in \mathbb{N}} \mathcal{A}(\beta[x \mapsto a](\text{gerade}(x)) = 1$

Erklärung:
Falls $a = 2k$ so $\beta[x \mapsto a](x) = a \in \text{gerade}_{\mathcal{A}}$. Dann $\mathcal{A}(\beta[x \mapsto a](\text{gerade}(x)) = 1$.
Falls $a = 2k + 1$ so $\beta[x \mapsto a](x) = a \notin \text{gerade}_{\mathcal{A}}$. Dann $\mathcal{A}(\beta[x \mapsto a](\text{gerade}(x)) = 0$.

Es gibt $a \in \mathbb{N}$ mit $\mathcal{A}(\beta[x \mapsto a](\text{gerade}(x)) = 1$.
$\max_{a \in \mathbb{N}} \mathcal{A}(\beta[x \mapsto a](\text{gerade}(x)) = \max\{1, 0\} = 1$

Beispiel

$$U_{\mathbb{N}} = \{0, 1, 2, \ldots\}$$
$$0_{\mathbb{N}} = 0 \in U_{\mathbb{N}}$$
$$s_{\mathbb{N}} : U_{\mathbb{N}} \to U_{\mathbb{N}}, \; s_{\mathbb{N}}(n) = n + 1$$
$$+_{\mathbb{N}} : U_{\mathbb{N}}^2 \to U_{\mathbb{N}}, \; +_{\mathbb{N}}(n, m) = n + m$$
$$*_{\mathbb{N}} : U_{\mathbb{N}}^2 \to U_{\mathbb{N}}, \; *_{\mathbb{N}}(n, m) = n * m$$
$$\leq_{\mathbb{N}} = \text{“kleiner-gleich”} \subseteq U_{\mathbb{N}}^2$$
$$<_{\mathbb{N}} = \text{“kleiner”} \subseteq U_{\mathbb{N}}^2$$

Mit $\beta(x) = 1, \beta(y) = 3$ ergibt sich beispielsweise:

$$\begin{aligned}
\mathbb{N}(\beta)(s(x) + s(0)) &= 3 \\
\mathbb{N}(\beta)(x + y \approx s(y)) &= 1 \\
\mathbb{N}(\beta)(\forall x, y(x + y \approx y + x)) &= 1 \\
\mathbb{N}(\beta)(\forall z \; z \leq y) &= 0 \\
\mathbb{N}(\beta)(\forall x \exists y \; x < y) &= 1
\end{aligned}$$

Beispiel

(1) $\mathbb{N}(\beta)(s(x)+s(0)) = s_{\mathbb{N}}(\beta(x)) +_{\mathbb{N}} s_{\mathbb{N}}(0_{\mathbb{N}}) = (1+1)+(0+1) = 3$

(2) $\mathbb{N}(\beta)(x+y \approx s(y)) = 1$

Erklärung:

$$\mathbb{N}(\beta)(x+y) = \beta(x) +_{\mathbb{N}} \beta(y) = 1+3 = 4$$
$$\mathbb{N}(\beta)(s(y)) = s_{\mathbb{N}}(\beta(y)) = 3+1 = 4.$$

(3)
$$\begin{aligned}
\mathbb{N}(\beta)(\forall x, y(x+y \approx y+x)) &= \min_{a\in\mathbb{N}} \mathbb{N}(\beta[x\mapsto a])(\forall y(x+y\approx y+x)) \\
&= \min_{a\in\mathbb{N}} \min_{b\in\mathbb{N}} \mathbb{N}(\beta[x\mapsto a, y\mapsto b])(x+y\approx y+x) \\
&= \min_{a,b\in\mathbb{N}} \mathbb{N}(\beta[x\mapsto a, y\mapsto b])(x+y\approx y+x) \\
&= 1
\end{aligned}$$

da für alle $a, b \in \mathbb{N}$:

$$\mathbb{N}(\beta[x\mapsto a, y\mapsto b])(x+y) = a+b = b+a = \mathbb{N}(\beta[x\mapsto a, y\mapsto b])(y+x)$$

Beispiel

(4) $\mathbb{N}(\beta)(\forall z\ z \leq y) = \min_{a \in \mathbb{N}} \mathbb{N}(\beta[z \mapsto a])(z \leq y) = 0$

Erklärung:

Falls $a = 4$, so $\mathbb{N}(\beta[z \mapsto a])(z \leq y) = 0$, da:

$\mathbb{N}(\beta[z \mapsto a])(z) = a = 4$

$\mathbb{N}(\beta[z \mapsto a])(y) = \beta(y) = 3$

und $(4, 3) \notin \leq_{\mathbb{N}}$.

(5) $\mathbb{N}(\beta)(\forall x \exists y\ x < y) = \min_{a \in \mathbb{N}} \mathbb{N}(\beta[x \mapsto a])(\exists y\ x < y)$

$= \min_{a \in \mathbb{N}} \max_{b \in \mathbb{N}} \mathbb{N}(\beta[x \mapsto a, y \mapsto b])(x < y) = 1$

Erklärung:

Für jede Zahl $a \in \mathbb{N}$: $\max_{b \in \mathbb{N}} \mathbb{N}(\beta[x \mapsto a, y \mapsto b])(x < y) = 1$,

da es gibt $b = a + 1 \in \mathbb{N}$ mit $\mathbb{N}(\beta[x \mapsto a, y \mapsto b])(x < y) = 1$

weil $\mathbb{N}(\beta[x \mapsto a, y \mapsto b])(x) = a$

$\mathbb{N}(\beta[x \mapsto a, y \mapsto b])(y) = b = a + 1$ und $(a, a + 1) \in <_{\mathbb{N}}$.

Logik für Informatiker

Vorlesung 10: Prädikatenlogik

Babeş-Bolyai Universität, Department für Informatik, Cluj-Napoca
csacarea@cs.ubbcluj.ro

Syntax

1. **Logische Symbole:**
   1.1: Wie in der Aussagenlogik: $\top, \bot$; $\neg$; $\vee, \wedge, \rightarrow, \leftrightarrow$
   1.2: Quantoren: $\forall$, $\exists$.
2. **Nichtlogische Symbole:** Signatur $\Sigma = (\Omega, \Pi)$,
   2.1: Ω Menge von Funktionssymbolen. Notation: f/n: f hat Stelligkeit $n \geq 0$,
   2.2: Π Menge von Prädikatensymbolen. Notation: p/m: p hat Stelligkeit $m \geq 0$.
   (Das Gleichheitsprädikat $\approx$ kann (muss aber nicht) enthalten sein.)

   Funktionsssymbole mit Stelligkeit $n = 0$ heißen Konstante

   Prädikatensymbole mit Stelligkeit $n = 0$ heißen Aussagenvariablen
3. **Variablen:** X vorgegebene Menge von abzählbar unendlich vielen Symbolen ist, die wir für (die Bezeichnung von) Variablen verwenden.

- Terme
- Formeln
- Substitutionen

Semantik

- Σ-Struktur

$$\mathcal{A} = (U,\ (f_{\mathcal{A}} : U^n \to U)_{f/n \in \Omega},\ (p_{\mathcal{A}} \subseteq U^m)_{p/m \in \Pi})$$

  wobei $U \neq \emptyset$ eine Menge, genannt Universum von $\mathcal{A}$.
- Valuation: Abbildung $\beta : X \to U$
- Wert eines Terms in $\mathcal{A}$ bzgl. β
- Wahrheitswert einer Formel in $\mathcal{A}$ bzgl. β

Beispiel

$$U_{\mathbb{N}} = \{0, 1, 2, \ldots\}$$
$$0_{\mathbb{N}} = 0 \in U_{\mathbb{N}}$$
$$s_{\mathbb{N}} : U_{\mathbb{N}} \to U_{\mathbb{N}},\ s_{\mathbb{N}}(n) = n + 1$$
$$+_{\mathbb{N}} : U_{\mathbb{N}}^2 \to U_{\mathbb{N}},\ \ +_{\mathbb{N}}(n, m) = n + m$$
$$*_{\mathbb{N}} : U_{\mathbb{N}}^2 \to U_{\mathbb{N}},\ \ *_{\mathbb{N}}(n, m) = n * m$$
$$\leq_{\mathbb{N}} = \text{“kleiner-gleich”} \subseteq U_{\mathbb{N}}^2$$
$$<_{\mathbb{N}} = \text{“kleiner”} \subseteq U_{\mathbb{N}}^2$$

Mit $\beta(x) = 1, \beta(y) = 3$ ergibt sich beispielsweise:

$$\begin{aligned}
\mathbb{N}(\beta)(s(x) + s(0)) &= 3 \\
\mathbb{N}(\beta)(x + y \approx s(y)) &= 1 \\
\mathbb{N}(\beta)(\forall x, y(x + y \approx y + x)) &= 1 \\
\mathbb{N}(\beta)(\forall z\ z \leq y) &= 0 \\
\mathbb{N}(\beta)(\forall x \exists y\ x < y) &= 1
\end{aligned}$$

Beispiel

(1) $\mathbb{N}(\beta)(s(x)+s(0)) = s_{\mathbb{N}}(\beta(x)) +_{\mathbb{N}} s_{\mathbb{N}}(0_{\mathbb{N}}) = (1+1)+(0+1) = 3$

(2) $\mathbb{N}(\beta)(x+y \approx s(y)) = 1$

Erklärung:

$$\mathbb{N}(\beta)(x+y) = \beta(x) +_{\mathbb{N}} \beta(y) = 1+3 = 4$$
$$\mathbb{N}(\beta)(s(y)) = s_{\mathbb{N}}(\beta(y)) = 3+1 = 4.$$

(3)
$$\begin{aligned}
\mathbb{N}(\beta)(\forall x, y(x+y \approx y+x)) &= \min_{a\in\mathbb{N}} \mathbb{N}(\beta[x\mapsto a])(\forall y(x+y\approx y+x)) \\
&= \min_{a\in\mathbb{N}} \min_{b\in\mathbb{N}} \mathbb{N}(\beta[x\mapsto a, y\mapsto b])(x+y\approx y+x) \\
&= \min_{a,b\in\mathbb{N}} \mathbb{N}(\beta[x\mapsto a, y\mapsto b])(x+y\approx y+x) \\
&= 1
\end{aligned}$$

da für alle $a, b \in \mathbb{N}$:

$$\mathbb{N}(\beta[x\mapsto a, y\mapsto b])(x+y) = a+b = b+a = \mathbb{N}(\beta[x\mapsto a, y\mapsto b])(y+x)$$

Beispiel

(4) $\mathbb{N}(\beta)(\forall z\ z \leq y) = \min_{a \in \mathbb{N}} \mathbb{N}(\beta[z \mapsto a])(z \leq y) = 0$

Erklärung:

Falls $a = 4$, so $\mathbb{N}(\beta[z \mapsto a])(z \leq y) = 0$, da:

$\mathbb{N}(\beta[z \mapsto a])(z) = a = 4$

$\mathbb{N}(\beta[z \mapsto a])(y) = \beta(y) = 3$

und $(4, 3) \notin \leq_{\mathbb{N}}$.

(5) $\mathbb{N}(\beta)(\forall x \exists y\ x < y) = \min_{a \in \mathbb{N}} \mathbb{N}(\beta[x \mapsto a])(\exists y\ x < y)$

$= \min_{a \in \mathbb{N}} \max_{b \in \mathbb{N}} \mathbb{N}(\beta[x \mapsto a, y \mapsto b])(x < y) = 1$

Erklärung:

Für jede Zahl $a \in \mathbb{N}$: $\max_{b \in \mathbb{N}} \mathbb{N}(\beta[x \mapsto a, y \mapsto b])(x < y) = 1$,

da es gibt $b = a + 1 \in \mathbb{N}$ mit $\mathbb{N}(\beta[x \mapsto a, y \mapsto b])(x < y) = 1$

weil $\mathbb{N}(\beta[x \mapsto a, y \mapsto b])(x) = a$

$\mathbb{N}(\beta[x \mapsto a, y \mapsto b])(y) = b = a + 1$ und $(a, a + 1) \in <_{\mathbb{N}}$.

Gültigkeit und Erfüllbarkeit

Definition. F gilt in $\mathcal{A}$ unter β:

$$\mathcal{A}, \beta \models F \text{ g.d.w. } \mathcal{A}(\beta)(F) = 1$$

Definition. F gilt in $\mathcal{A}$ ($\mathcal{A}$ ist Modell von F):

$$\mathcal{A} \models F \text{ g.d.w. } \mathcal{A}, \beta \models F, \text{ für alle } \beta : X \to U_{\mathcal{A}}$$

Definition. F ist (allgemein-) gültig:

$$\models F \text{ g.d.w. } \mathcal{A} \models F, \text{ für alle } \mathcal{A} \in \Sigma\text{-Str}$$

Definition. F heißt erfüllbar gdw. es $\mathcal{A}$ und β gibt, so dass $\mathcal{A}, \beta \models F$.
Sonst heißt F unerfüllbar.

Folgerung und Äquivalenz

Definition. F impliziert G (oder G folgt aus F), i.Z. $F \models G$
gdw. für alle $\mathcal{A} \in \Sigma$-Str und $\beta : X \to U_{\mathcal{A}}$ gilt:
Falls $\mathcal{A}, \beta \models F$, so $\mathcal{A}, \beta \models G$.

Definition. F und G sind äquivalent
gdw. für alle $\mathcal{A} \in \Sigma$-Str und $\beta : X \to U_{\mathcal{A}}$ gilt:
$\mathcal{A}, \beta \models F$ genau dann, wenn $\mathcal{A}, \beta \models G$.

Erweiterung auf Formelmengen N in natürlicher Weise:

Definition. $N \models G$ gdw.
für alle $\mathcal{A} \in \Sigma$-Str und $\beta : X \to U_{\mathcal{A}}$:
falls $(\mathcal{A}, \beta \models F$, für alle $F \in N)$, so $(\mathcal{A}, \beta \models G)$.

Folgerung / Äquivalenz und Gültigkeit

Satz. $F \models G$ gdw. $(F \to G)$ ist gültig

Beweis

$$
\begin{aligned}
F \models G \quad & \text{gdw.} \quad \text{für alle } \mathcal{A} \in \Sigma\text{-Str und } \beta : X \to U_{\mathcal{A}}\text{: Falls } \mathcal{A}, \beta \models F\text{, so } \mathcal{A}, \beta \models G. \\
& \text{gdw.} \quad \text{für alle } \mathcal{A} \in \Sigma\text{-Str und } \beta : X \to U_{\mathcal{A}}\text{: } \mathcal{A}(\beta)(F) \to_b \mathcal{A}(\beta)(G) = 1 \\
& \text{gdw.} \quad \text{für alle } \mathcal{A} \in \Sigma\text{-Str und } \beta : X \to U_{\mathcal{A}}\text{: } \mathcal{A}(\beta)(F \to G) = 1
\end{aligned}
$$

Satz. F und G sind äquivalent gdw. $(F \leftrightarrow G)$ ist gültig.

Beweis

$$
\begin{aligned}
F \text{ und } G \text{ sind äquivalent} \quad & \text{gdw.} \quad F \models G \text{ und } F \models G \\
& \text{gdw.} \quad \models F \to G \text{ und } \models G \to F \\
& \text{gdw.} \quad \models F \leftrightarrow G
\end{aligned}
$$

Gültigkeit und Unerfüllbarkeit

Nachweis von Gültigkeit (und damit Folgerung oder Äquivalenz) durch Unerfüllbarkeitstest:

F gültig genau dann, wenn $\neg F$ unerfüllbar

$N \models F$ genau dann, wenn $N \cup \neg F$ unerfüllbar

Eigenschaften von Quantoren

Quantoren gleicher Art kommutieren

$\forall x \forall y$ ist das gleiche wie $\forall y \forall x$

$\exists x \exists y$ ist das gleiche wie $\exists y \exists x$

Informell: Für jede Formel F gilt: $\forall x \forall y F \equiv \forall y \forall x F$; $\exists x \exists y F \equiv \exists y \exists x F$.

Theorem

Sei $\Sigma = (\Omega, \Pi)$ eine Signatur. Für alle Σ-Formeln F gilt:

(1) $\forall x \forall y\, F \equiv \forall y \forall x\, F$

(2) $\exists x \exists y\, F \equiv \exists y \exists x\, F$

Eigenschaften von Quantoren

Verschiedene Quantoren kommutieren NICHT

Beispiel:

$\forall x \exists y \;\; Mutter(y, x)$ Jeder hat eine Mutter (richtig)

$\exists y \forall x \;\; Mutter(y, x)$ Es gibt eine Person, die die Mutter von jedem ist (falsch)

Bemerkung: $\forall x \exists y F \equiv \exists y \forall x F$ gilt nicht immer.
Es gibt eine Formel F so dass $\forall x \exists y F$ und $\exists y \forall x F$ nicht logisch äquivalent.

Theorem. Sei $\Sigma = (\Omega, \Pi)$ eine Signatur. Für alle Σ-Formeln F gilt:

$$\exists x \forall y \, F \models \forall y \exists x \, F$$

Es gibt eine Signatur $\Sigma = (\Omega, \Pi)$ und eine Formel F mit:

$$\forall x \exists y \, F \not\models \exists y \forall x \, F$$

Eigenschaften von Quantoren

Dualität der Quantoren

$\forall x \ldots$ ist das gleiche wie $\neg\exists x \neg \ldots$

$\exists x \ldots$ ist das gleiche wie $\neg\forall x \neg \ldots$

Beispiel:

$\forall x\, \mathrm{mag}(x, \mathrm{eiscreme})$ ist das gleiche wie $\neg\exists x\, \neg\mathrm{mag}(x, \mathrm{eiscreme})$

$\exists x\, \mathrm{mag}(x, \mathrm{broccoli})$ ist das gleiche wie $\neg\forall x\, \neg\mathrm{mag}(x, \mathrm{broccoli})$

Informell: Für jede Formel F gilt: $\forall x F \equiv \neg\exists x \neg F$; $\exists x F \equiv \neg\forall x \neg F$.

Theorem. Sei $\Sigma = (\Omega, \Pi)$ eine Signatur. Für alle Σ-Formeln F gilt:

(1) $\forall x\, F \equiv \neg\exists x\, \neg F$

(2) $\exists x\, F \equiv \neg\forall x\, \neg F$

Eigenschaften von Quantoren

$\forall$ distributiert über $\wedge$

$\forall x(\ldots \wedge \ldots)$ ist das gleiche wie $(\forall x \ldots) \wedge (\forall x \ldots)$

Beispiel

$\forall x\,(studiert(x) \wedge arbeitet(x))$ ist das gleiche wie

$(\forall x\, studiert(x)) \wedge (\forall x\, arbeitet(x))$

Informell: Für alle Formeln F, G gilt: $\forall x(F \wedge G) \equiv \forall x F \wedge \forall x G$.

Theorem. Sei $\Sigma = (\Omega, \Pi)$ eine Signatur. Für alle Σ-Formeln F, G gilt:

$$\forall x(F \wedge G) \equiv \forall x F \wedge \forall x G$$

Eigenschaften von Quantoren

$\exists$ distributiert über $\vee$

$\exists x(\ldots \vee \ldots)$ ist das gleiche wie $(\exists x \ldots) \vee (\exists x \ldots)$

Beispiel

$\exists x\,(gerade(x) \vee ungerade(x))$ ist das gleiche wie

$(\exists x\, gerade(x)) \vee (\exists x\, ungerade(x))$

Informell: Für alle Formeln F, G gilt: $\exists x(F \vee G) \equiv \exists x F \vee \exists x G$.

Theorem. Sei $\Sigma = (\Omega, \Pi)$ eine Signatur. Für alle Σ-Formeln F, G gilt:

$$\exists x\,(F \vee G) \equiv (\exists x\, F) \vee (\exists x\, G)$$

Eigenschaften von Quantoren

$\forall$ distributiert NICHT über $\vee$

$\forall x(\ldots \vee \ldots)$ ist NICHT das gleiche wie $(\forall x \ldots) \vee (\forall x \ldots)$

Beispiel

$\forall x\,(gerade(x) \vee ungerade(x))$ ist NICHT das gleiche wie

$(\forall x\, gerade(x)) \vee (\forall x\, ungerade(x))$

Theorem. Es gibt eine Signatur $\Sigma = (\Omega, \Pi)$ und Σ-Formeln F, G mit:

$$\forall x(F \vee G) \not\equiv \forall x F \vee \forall x G$$

(1) $\forall x F \vee \forall x G \models \forall x(F \vee G)$
(2) $\forall x(F \vee G) \not\models \forall x F \vee \forall x G$

Eigenschaften von Quantoren

$\exists$ distributiert NICHT über $\wedge$

$\exists x(\ldots \wedge \ldots)$ ist NICHT das gleiche wie $(\exists x \ldots) \wedge (\exists x \ldots)$

Beispiel

$\exists x\,(gerade(x) \wedge ungerade(x))$ ist NICHT das gleiche wie
$(\exists x\, gerade(x)) \wedge (\exists x\, ungerade(x))$

Theorem. Es gibt eine Signatur $\Sigma = (\Omega, \Pi)$ und Σ-Formeln F, G mit:

$$\exists x(F \wedge G) \not\equiv \exists x F \wedge \exists x G$$

(1) $\exists x(F \wedge G) \models \exists x F \wedge \exists x G$
(2) $\exists x F \wedge \exists x G \not\models \exists x(F \wedge G)$

Zusammenfassung/Wichtige Äquivalenzen

Wichtige Äquivalenzen

- $\forall x \forall y F \equiv \forall y \forall x F$
- $\exists x \exists y F \equiv \exists y \exists x F$
- $\forall x (F \wedge G) \equiv (\forall x F) \wedge (\forall x G)$
- $\exists x (F \vee G) \equiv (\exists x F) \vee (\exists x G)$
- $\forall x F \equiv \neg \exists x \neg F$ $\qquad$ $\neg \forall x F \equiv \exists x \neg F$
- $\exists x F \equiv \neg \forall x \neg F$ $\qquad$ $\neg \exists x F \equiv \forall x \neg F$

Zusammenfassung/Wichtige Äquivalenzen

- $\forall x \forall y F \equiv \forall y \forall x F$
- $\exists x \exists y F \equiv \exists y \exists x F$
- $\forall x (F \wedge G) \equiv (\forall x F) \wedge (\forall x G)$
- $\exists x (F \vee G) \equiv (\exists x F) \vee (\exists x G)$
- $\forall x F \equiv \neg \exists x \neg F \qquad \neg \forall x F \equiv \exists x \neg F$
- $\exists x F \equiv \neg \forall x \neg F \qquad \neg \exists x F \equiv \forall x \neg F$

Falls x in der Formel H nicht vorkommt, dann:

- $(\forall x F(x)) \vee H \equiv \forall x (F(x) \vee H)$
- $(\forall x F(x)) \wedge H \equiv \forall x (F(x) \wedge H)$
- $(\exists x F(x)) \vee H \equiv \exists x (F(x) \vee H)$
- $(\exists x F(x)) \wedge H \equiv \exists x (F(x) \wedge H)$

Zusammenfassung

Aber Vorsicht

$\forall x \exists y F \not\equiv \exists y \forall x F$

- $\exists y \forall x F \models \forall x \exists y F$
- $\forall x \exists y F \not\models \exists y \forall x F$

$\forall x (F \vee G) \not\equiv (\forall x F) \vee (\forall x G)$

- $(\forall x F) \vee (\forall x G) \models \forall x (F \vee G)$
- $\forall x (F \vee G) \not\models \forall x F \vee \forall x G$

$\exists x (F \wedge G) \not\equiv (\exists x F) \wedge (\exists x G)$

- $\exists x (F \wedge G) \models (\exists x F) \wedge (\exists x G)$
- $(\exists x F) \wedge (\exists x G) \not\models \exists x (F \wedge G)$

Andere wichtige Äquivalenzen

$(F \wedge F) \equiv F$	$(F \vee F) \equiv F$	(Idempotenz)
$(F \wedge G) \equiv (G \wedge F)$	$(F \vee G) \equiv (G \vee F)$	(Kommutativität)
$(F \wedge (G \wedge H)) \equiv ((F \wedge G) \wedge H)$		
$(F \vee (G \vee H)) \equiv ((F \vee G) \vee H)$		(Assoziativität)
$(F \wedge (F \vee G)) \equiv F$		
$(F \vee (F \wedge G)) \equiv F$		(Absorption)
$(F \wedge (G \vee H)) \equiv ((F \wedge G) \vee (F \wedge H))$		
$(F \vee (G \wedge H)) \equiv ((F \vee G) \wedge (F \vee H))$		(Distributivität)
$(\neg\neg F) \equiv F$		(Doppelte Negation)
$\neg(F \wedge G) \equiv (\neg F \vee \neg G)$		
$\neg(F \vee G) \equiv (\neg F \wedge \neg G)$		(De Morgan's Regeln)
$(F \rightarrow G) \equiv (\neg G \rightarrow \neg F)$		(Kontraposition)
$(F \rightarrow G) \equiv (\neg F \vee G)$		(Elimination Implikation)
$F \leftrightarrow G \equiv (F \rightarrow G) \wedge (G \rightarrow F)$		(Elimination Äquivalenz)

Strukturelle Induktion

- Für Terme
- Für Formeln in Prädikatenlogik

Strukturelle Induktion: Terme

Menge $T_\Sigma(X)$ der Σ-Terme:

Die kleinste Menge mit:
- $X \subseteq T_\Sigma(X)$

Wenn
- $f \in \Omega$,
- n ist die Stelligkeit von f
- $t_1, \ldots, t_n \in T_\Sigma(X)$

dann $f(t_1, \ldots, t_n) \in T_\Sigma(X)$

Strukturelle Induktion: Terme

$\Sigma = (\Omega, \Pi)$

Sei $p(t)$ eine Eigenschaft der Σ-Terme in Prädikatenlogik

Behauptung: Für alle Terme t, $p(t)$ gilt

Beweis durch strukturelle Induktion:

Induktionsbasis: Zu zeigen:
$p(t)$ gilt für $t \in X$ und für alle Konstanten.

Sei t ein Term (nicht Variable oder Konstante).
Induktionsvoraussetzung:
$p(s)$ gilt für alle Teilterme s von t (mit $s \neq t$)

Induktionsschritt: Zu zeigen: $p(t)$ gilt:

Fallunterschied über alle $f \in \Omega$ mit $t = f(t_1, \ldots, t_n)$.

Strukturelle Induktion: Formeln

Menge For_Σ der Formeln über Σ:

Die kleinste Menge, die

- Alle atomaren Formeln enthält,
- $\top \in \mathsf{For}_\Sigma$, $\bot \in \mathsf{For}_\Sigma$,
- Wenn $F, G \in \mathsf{For}_\Sigma$, dann auch
  $\neg F, F \wedge G, F \vee G, F \rightarrow G, F \leftrightarrow G \in \mathsf{For}_\Sigma$,
- Wenn $F \in \mathsf{For}_\Sigma$ und $x \in X$, dann
  $\forall x F \in \mathsf{For}_\Sigma, \exists x F \in \mathsf{For}_\Sigma$

Strukturelle Induktion: Formeln

Sei $p(F)$ eine Eigenschaft der Σ-Formeln in Prädikatenlogik

Behauptung: Für alle Formeln F, $p(F)$ gilt

Beweis durch strukturelle Induktion:

Induktionsbasis: Zu zeigen:

$p(F)$ gilt für $F \in \{\top, \bot\}$ und für alle atomaren Formeln.

Sei F eine Formel (nicht atomar oder $\top$ oder $\bot$).

Induktionsvoraussetzung:

$p(G)$ gilt für alle Teilformeln G von F (mit $G \neq F$)

Induktionsschritt: Zu zeigen: $p(F)$ gilt:

Fall 1	$F = \neg G$	Fall 2	$F = G_1 \vee G_2$
Fall 3	$F = G_1 \wedge G_2$	Fall 4	$F = G_1 \rightarrow G_2$
Fall 5	$F = G_1 \leftrightarrow G_2$	Fall 6	$F = \forall x G$
		Fall 7	$F = \exists x G$

Substitutionen und Valuationen

Theorem (Substitutionslemma)
Für alle Σ-Strukturen $\mathcal{A}$, Wertebelegungen β, Σ-Formeln F, Variablen x und Terme t gilt:

$$\mathcal{A}, \beta \models F[t/x] \text{ g.d.w. } \mathcal{A}, \beta[x \mapsto \mathcal{A}(\beta)(t)] \models F$$

Beweis: Strukturelle Induktion

Plan: Wir benutzen folgendes Lemma:

Lemma: Für alle Σ-Strukturen $\mathcal{A}$, Wertebelegungen β, Variable x und Terme t_i, t:
$\mathcal{A}(\beta)(t_i[t/x]) = \mathcal{A}(\beta[x \mapsto \mathcal{A}(\beta)(t)])(t_i)$

Substitutionen und Valuationen

Lemma: Für alle Σ-Strukturen $\mathcal{A}$, Wertebelegungen β, Variable x und Terme t_i, t:
$\mathcal{A}(\beta)(t_i[t/x]) = \mathcal{A}(\beta[x \mapsto \mathcal{A}(\beta)(t)])(t_i)$

Beweis Strukturelle Induktion

$$p(t_i) \quad \mathcal{A}(\beta)(t_i[t/x]) = \mathcal{A}(\beta[x \mapsto \mathcal{A}(\beta)(t)])(t_i)$$

Zu zeigen: Für alle Terme t_i, $p(t_i)$ gilt.

1. **Induktionsbasis:** $p(t_i)$ gilt für $t_i \in X$ und für $t_i = c$ Konstante.

Fall 1: $t_i \in X$.

- Fall 1a: $t_i = x$. Dann: $t_i[t/x] = t$.
  $\mathcal{A}(\beta)(t_i[t/x]) = \mathcal{A}(\beta)(t) = \mathcal{A}(\beta[x \mapsto \mathcal{A}(\beta)(t)])(x) = \mathcal{A}(\beta[x \mapsto \mathcal{A}(\beta)(t)])(t_i)$.
- Fall 1a: $t_i = y$, mit $y \neq x$. Dann: $t_i[t/x] = y$
  $\mathcal{A}(\beta)(t_i[t/x]) = \mathcal{A}(\beta)(y) = \mathcal{A}(\beta[x \mapsto \mathcal{A}(\beta)(t)])(y) = \mathcal{A}(\beta[x \mapsto \mathcal{A}(\beta)(t)])(t_i)$.

Fall 2: $t_i = c$, $c/0 \in \Omega$ (Konstante). Dann: $t_i[t/x] = c$

$$\mathcal{A}(\beta)(t_i[t/x]) = \mathcal{A}(\beta)(c) = c_{\mathcal{A}} = \mathcal{A}(\beta[x \mapsto \mathcal{A}(\beta)(t)])(c) = \mathcal{A}(\beta[x \mapsto \mathcal{A}(\beta)(t)])(t_i).$$

Substitutionen und Valuationen

Lemma: Für alle Σ-Strukturen $\mathcal{A}$, Wertebelegungen β, Variable x und Terme t_i, t:
$\mathcal{A}(\beta)(t_i[t/x]) = \mathcal{A}(\beta[x \mapsto \mathcal{A}(\beta)(t)])(t_i)$

Beweis (ctd.)

2. **Induktionsvoraussetzung:** Sei t_i Term (nicht Variable oder Konstante).

   Annahme: $p(s)$ gilt für alle Teilterme s von t_i (mit $s \neq t_i$)

3. **Induktionsschritt:** Zu zeigen: $p(t_i)$ gilt, wobei $t_i = f(s_1, \ldots, s_n)$.

$$
\begin{aligned}
\mathcal{A}(\beta)(t_i[t/x]) &= \mathcal{A}(\beta)(f(s_1, \ldots, s_n)[t/x]) = \\
&= \mathcal{A}(\beta)(f(s_1[t/x], \ldots, s_n[t/x]) = && \text{(Anw. Subst.)} \\
&= f_{\mathcal{A}}(\mathcal{A}(\beta)(s_1[t/x]), \ldots, \mathcal{A}(\beta)(s_n[t/x])) \\
&\overset{IV}{=} f_{\mathcal{A}}(\mathcal{A}(\beta[x \mapsto \mathcal{A}(\beta)(t)])(s_1), \ldots \mathcal{A}(\beta[x \mapsto \mathcal{A}(\beta)(t)])(s_n)) \\
&= \mathcal{A}(\beta[x \mapsto \mathcal{A}(\beta)(t)])(f(s_1, \ldots, s_n)) \\
&= \mathcal{A}(\beta[x \mapsto \mathcal{A}(\beta)(t)])(t_i)
\end{aligned}
$$

Substitutionen und Valuationen

Theorem (Substitutionslemma)
Für alle Σ-Strukturen $\mathcal{A}$, Wertebelegungen β, Σ-Formeln F, Variablen x und Terme t gilt:

$$\mathcal{A}, \beta \models F[t/x] \text{ g.d.w. } \mathcal{A}, \beta[x \to \mathcal{A}(\beta)(t)] \models F$$

Beweis: Strukturelle Induktion

1. **Induktionsbasis:** Zu zeigen: Die Eigenschaft gilt für $\top$, $\bot$ und alle atomaren Formeln.

Fall 1: $F = \top$ Dann $F[t/x] = \top$;
$\mathcal{A}(\beta)(F[t/x]) = \mathcal{A}(\beta)(\top) = 1 = \mathcal{A}(\beta[x \mapsto \mathcal{A}(\beta)(t)])(\top) = \mathcal{A}(\beta[x \mapsto \mathcal{A}(\beta)(t)])(F)$

Fall 2: $F = \bot$ Dann $F[t/x] = \bot$;
$\mathcal{A}(\beta)(F[t/x]) = \mathcal{A}(\beta)(\bot) = 0 = \mathcal{A}(\beta[x \mapsto \mathcal{A}(\beta)(t)])(\bot) = \mathcal{A}(\beta[x \mapsto \mathcal{A}(\beta)(t)])(F)$.

Fall 3: $F = p(t_1, \ldots, t_n)$ mit $p/n \in \Pi$. Dann $F[t/x] = p(t_1[t/x], \ldots, t_n[t/x])$.
$\mathcal{A}(\beta)(F[t/x]) = \mathcal{A}(\beta)(p(t_1[t/x], .., t_n[t/x])) = 1$ iff $(\mathcal{A}(\beta)(t_1[t/x]), .., \mathcal{A}(\beta)(t_n[t/x])) \in p_{\mathcal{A}}$
$\mathcal{A}(\beta[x \mapsto \mathcal{A}(\beta)(t)])(p(t_1, .., t_n)) = 1$ iff $(\mathcal{A}(\beta[x \mapsto \mathcal{A}(\beta)(t)])(t_1), .., \mathcal{A}(\beta[x \mapsto \mathcal{A}(\beta)(t)])(t_n)) \in p$
Lemma: $\mathcal{A}(\beta)(t_i[t/x]) = \mathcal{A}(\beta[x \mapsto \mathcal{A}(\beta)(t)])(t_i)$. Äquivalenz folgt daraus.

Substitutionen und Valuationen

Theorem (Substitutionslemma)
Für alle Σ-Strukturen $\mathcal{A}$, Wertebelegungen β, Σ-Formeln F, Variablen x und Terme t gilt:

$$\mathcal{A}, \beta \models F[t/x] \text{ g.d.w. } \mathcal{A}, \beta[x \to \mathcal{A}(\beta)(t)] \models F$$

Beweis: ctd.

2. **Induktionsvoraussetzung:** Sei F eine Formel (nicht atomar oder $\top$ oder $\bot$).
   Annahme: $p(G)$ gilt für alle Teilformeln G von F (mit $G \neq F$)
3. **Induktionsschritt:** Zu zeigen: $p(F)$ gilt:

Fall 1: $F = \neg G$. Dann $F[t/x] = \neg(G[t/x])$.

$\mathcal{A}(\beta)(F[t/x]) = \mathcal{A}(\beta)(\neg(G[t/x]) = 1$ iff $\mathcal{A}(\beta)(G[t/x]) = 0$ iff (Ind.Voraus.)
$\mathcal{A}(\beta[x \mapsto \mathcal{A}(\beta)(t)])(G)=0$ iff $\mathcal{A}(\beta[x \mapsto \mathcal{A}(\beta)(t)])(\neg G)=1$ iff $\mathcal{A}(\beta[x \mapsto \mathcal{A}(\beta)(t)])(F)=1$

Fall 2-5: $F = G_1 \mathrm{op} G_2$, $\mathrm{op} \in \{\vee, \wedge, \to, \leftrightarrow\}$. Dann $F[t/x] = G_1[t/x] \mathrm{op} G_2[t/x]$.

$\mathcal{A}(\beta)(F[t/x]) = \mathcal{A}(\beta)(G_1[t/x] \mathrm{op} G_2[t/x]) = \mathcal{A}(\beta)(G_1[t/x]) \mathrm{op}_B \mathcal{A}(\beta)(G_2[t/x]) =$ (I.V.)
$\mathcal{A}(\beta[x \mapsto \mathcal{A}(\beta)(t)])(G_1) \mathrm{op}_B \mathcal{A}(\beta[x \mapsto \mathcal{A}(\beta)(t)])(G_2) = \mathcal{A}(\beta[x \mapsto \mathcal{A}(\beta)(t)])(\underbrace{G_1 \mathrm{op} G_2}_{F})$.

Substitutionen und Valuationen

Theorem (Substitutionslemma)
Für alle Σ-Strukturen $\mathcal{A}$, Wertebelegungen β, Σ-Formeln F, Variablen x und Terme t gilt:

$$\mathcal{A}, \beta \models F[t/x] \text{ g.d.w. } \mathcal{A}, \beta[x \to \mathcal{A}(\beta)(t)] \models F$$

Fall 6: $F = \forall y G$.

Fall 6.1: $y = x$. Dann $F[t/x] = \forall x G[t/x] = \forall x G$.

$\mathcal{A}(\beta)(F[t/x]) = \mathcal{A}(\beta)(\forall x G) = \min\{\mathcal{A}(\beta[x \mapsto a])(G) \mid a \in U_{\mathcal{A}}\}$

$\mathcal{A}(\beta[x \mapsto \mathcal{A}(\beta)(t)])(\forall x G) = \min\{\mathcal{A}(\beta[x \mapsto \mathcal{A}(\beta)(t)][x \mapsto a])(G) \mid a \in U_{\mathcal{A}}\} =$
$\min\{\mathcal{A}(\beta[x \mapsto a])(G) \mid a \in U_{\mathcal{A}}\}$.

Fall 6.2: $y \neq x$, t enthält nicht y. Dann $F[t/x] = \forall y G[t/x] = \forall y (G[t/x])$.

$\mathcal{A}(\beta)(F[t/x]) = \mathcal{A}(\beta)(\forall y (G[t/x])) = \min\{\mathcal{A}(\beta[y \mapsto a])(G[t/x]) \mid a \in U_{\mathcal{A}}\} =$ (Ind.Vor.)
$\min\{\mathcal{A}(\beta[y \mapsto a, x \mapsto \mathcal{A}(\beta[y \mapsto a])(t)])(G) \mid a \in U_{\mathcal{A}}\} =$
$\min\{\mathcal{A}(\beta[y \mapsto a, x \mapsto \mathcal{A}(\beta)(t)])(G) \mid a \in U_{\mathcal{A}}\}$ (y kommt in t nicht vor)

$\mathcal{A}(\beta[x \mapsto \mathcal{A}(\beta)(t)])(\forall y G) = = \min\{\mathcal{A}(\beta[x \mapsto \mathcal{A}(\beta)(t)][y \mapsto a])(G) \mid a \in U_{\mathcal{A}}\} =$
$\min\{\mathcal{A}(\beta[y \mapsto a, x \mapsto \mathcal{A}(\beta)(t)])(G) = \mid a \in U_{\mathcal{A}}\}$.

Substitutionen und Valuationen

Theorem (Substitutionslemma)
Für alle Σ-Strukturen $\mathcal{A}$, Wertebelegungen β, Σ-Formeln F, Variablen x und Terme t gilt:

$$\mathcal{A}, \beta \models F[t/x] \text{ g.d.w. } \mathcal{A}, \beta[x \to \mathcal{A}(\beta)(t)] \models F$$

Fall 6: $F = \forall y G$.

Fall 6.3: $y \neq x$, t enthält y. Dann $F[t/x] = \forall y' G[y'/y][t/x]$.

$$\begin{aligned}
\mathcal{A}(\beta)(F[t/x]) &= \mathcal{A}(\beta)(\forall y'(G[y'/y][t/x]) = \\
&= \min\{\mathcal{A}(\beta[y' \mapsto a])(G[y'/y][t/x]) \mid a \in U_{\mathcal{A}}\} = && \text{(Ind.Vor.)} \\
&= \min\{\mathcal{A}(\beta[y' \mapsto a, x \mapsto \mathcal{A}(\beta[y' \mapsto a])(t))(G[y'/y]) \mid a \in U_{\mathcal{A}}\} = && \text{(Ind.Vor.)} \\
&= \min\{\mathcal{A}(\beta[y' \mapsto a, x \mapsto \mathcal{A}(\beta[y' \mapsto a])(t), y \mapsto a])(G) \mid a \in U_{\mathcal{A}}\} = \\
&\qquad [\text{Rem:} \mathcal{A}(\beta[y' \mapsto a, x \mapsto \mathcal{A}(\beta[y' \mapsto a])(t)])(y') = a] \\
&= \min\{\mathcal{A}(\beta[x \mapsto \mathcal{A}(\beta)(t), y \mapsto a])(G) \mid a \in U_{\mathcal{A}}\} && (y' \text{ kommt in } t \text{ u. } G \text{ nicht vor}
\end{aligned}$$

$$\begin{aligned}
\mathcal{A}(\beta[x \mapsto \mathcal{A}(\beta)(t)])(\forall y G) &= \min\{\mathcal{A}(\beta[x \mapsto \mathcal{A}(\beta)(t)][y \mapsto a])(G) \mid a \in U_{\mathcal{A}}\} = \\
&= \min\{\mathcal{A}(\beta[y \mapsto a, x \mapsto \mathcal{A}(\beta)(t)])(G) \mid a \in U_{\mathcal{A}}\}.
\end{aligned}$$

Substitutionen und Valuationen

Theorem (Substitutionslemma)
Für alle Σ-Strukturen $\mathcal{A}$, Wertebelegungen β, Σ-Formeln F, Variablen x und Terme t gilt:

$$\mathcal{A}, \beta \models F[t/x] \text{ g.d.w. } \mathcal{A}, \beta[x \to \mathcal{A}(\beta)(t)] \models F$$

Fall 7: $F = \exists y G$.
Ähnlich zu Fall 6.

Substitutionen und Valuationen

Theorem (Substitutionslemma)
Für alle Σ-Strukturen $\mathcal{A}$, Wertebelegungen β, Σ-Formeln F, Variablen x und Terme t gilt:

$$\mathcal{A}, \beta \models F[t/x] \quad \text{g.d.w. } \mathcal{A}, \beta[x \mapsto \mathcal{A}(\beta)(t)] \models F$$

Allgemeiner gilt für beliebige Substitutionen σ:

Theorem.

$$\mathcal{A}, \beta \models F\sigma \quad \text{g.d.w. } \mathcal{A}, \beta \circ \sigma \models F,$$

wobei $\beta \circ \sigma : X \to \mathcal{A}$ die Wertebelegung mit $\beta \circ \sigma(x) = \mathcal{A}(\beta)(x\sigma)$, für alle Variablen x.

Beweis: $\sigma = [t_1/x_1, \ldots t_n/x_n]$
Induktion nach Anzahl n von Variablen in $dom(\sigma) = \{x_1, \ldots, x_n\}$

Umbenennung von Variablen

Lemma

Für alle Σ-Formeln F, Variablen x gilt:

$$\forall x\, F \equiv \forall z\, F[z/x]$$

wobei z eine neue Variable ist.

Zusammenfassung: Syntax und Semantik

- Prädikatenlogische Signatur
- Term, Atom, Formel
- Prädikatenlogisches Modell
- Auswertung von Formeln in Modellen
- Erfüllbarkeit, Gültigkeit; Folgerung, Äquivalenz
- Eigenschaften von Quantoren (Vertauschbarkeit untereinander und mit $\wedge, \vee$)
- Substitutionslemma

Schönen
Nikolaustag !

Logik für Informatiker

Vorlesung 11: Normalformen

Babeş-Bolyai Universität, Department für Informatik, Cluj-Napoca
csacarea@cs.ubbcluj.ro

Zusammenfassung: Syntax und Semantik

- Prädikatenlogische Signatur
- Term, Atom, Formel
- Prädikatenlogisches Modell
- Auswertung von Formeln in Modellen
- Erfüllbarkeit, Gültigkeit; Folgerung, Äquivalenz
- Eigenschaften von Quantoren (Vertauschbarkeit untereinander und mit $\wedge, \vee$)
- Substitutionslemma

Algorithmische Probleme

Gültigkeit(F):

$\models F$? (Σ fest)

Erfüllbarkeit(F):

F erfüllbar? (Σ fest)

Modelltest(F,A):

$A \models F$? (Σ fest)

Ausblick: Kalküle, Entscheidbarkeit

Kalküle

Es gibt korrekte und vollständige Kalküle für Prädikatenlogik
(z.B. Resolution, Tableaux)

Aber diese Kalküle können die Erfüllbarkeit von Formeln NICHT entscheiden

Ausblick: Kalküle, Entscheidbarkeit

Kalküle

Es gibt korrekte und vollständige Kalküle für Prädikatenlogik
(z.B. Resolution, Tableaux)

Aber diese Kalküle können die Erfüllbarkeit von Formeln NICHT entscheiden

Aussagenlogik

Allgemeingültigkeit, Erfüllbarkeit, Unerfüllbarkeit ENTSCHEIDBAR

Prädikatenlogik

- Es gibt Σ, so dass Gültigkeit(F) unentscheidbar.
- Menge der allgemeingültigen Formeln REKURSIV AUFZÄHLBAR
- Menge der unerfüllbaren Formeln REKURSIV AUFZÄHLBAR
- Menge der erfüllbaren Formeln NICHT REKURSIV AUFZÄHLBAR

Rekursiv Aufzählbar

Informelle Definition. Als rekursiv aufzählbare Menge

(auch semi-entscheidbare Menge, positiv semi-entscheidbare Menge, halb-entscheidbare Menge, berechenbar aufzählbare Menge, kurz r.e., c.e)

wird in der Berechenbarkeitstheorie eine aufzählbare Menge A (z.B. eine Teilmenge von $\mathbb{N}$) bezeichnet, wenn es einen Algorithmus gibt, der die Elemente von A aufzählt.

Äquivalent ist folgende Definition: es gibt einen Algorithmus, der 1 ausgibt wenn die Eingabe in A ist, und auf anderen Eingaben 0 ausgibt oder nicht hält.

Ziel

- Kalküle zur systematischen Überprüfung von Erfüllbarkeit (für Formeln und/oder Formelmengen)

Normalformen, Skolemisierung, Herbrandmodelle

Vorteile von Normalformen

- Reduktion der logischen Konzepte
- einfache Datenstrukturen für Beweisverfahren

Negationsnormalform

Definition. Eine Formel $F \in \mathrm{For}_\Sigma$ ist in Negationsnormalform (NNF), falls:

- $\rightarrow, \leftrightarrow$ kommen in F nicht vor
- jedes Negationszeichen in F steht direkt vor einem Atom (insbes. auch kein $\neg\neg$)

Beispiele:

NNF:

$$P, \quad \neg P, \quad (\neg P \vee Q) \wedge (R \vee (Q \wedge \neg P))$$

$$p(x, y) \vee \neg q(y)$$

nicht NNF:

$$\neg\neg P, \quad \neg(P \vee Q)$$

$$p(x, y) \rightarrow q(x)$$

Bereinigte Formeln

Definition. Eine Formel $F \in For_\Sigma$ ist bereinigt, falls:

- Keine Variable in F sowohl gebunden als auch frei vorkommt
- Keine Variable mehr als einmal in F quantifiziert ist

Beispiele:

Bereinigt

$$P \vee Q$$

$$\forall x \exists y (p(x) \vee q(x, y) \vee \exists z\, r(x, z))$$

Nicht bereinigt

$$p(x) \vee \forall x\, q(x)$$

$$(\forall x\, p(x)) \vee (\forall x\, q(x))$$

Pränexe Normalform

Definition: Pränexe Formeln sind von der Form

$$Q_1 x_1 \dots Q_n x_n \; F,$$

wobei F quantorenfrei, $Q_i \in \{\forall, \exists\}$. Hierbei heißt $Q_1 x_1 \dots Q_n x_n$ der Quantorenpräfix und F die Matrix der Formel.

Beispiele

In Pränexnormalform

$$p(x) \vee q(x)$$

$$\forall x \exists y \, (p(x) \vee q(y))$$

Nicht in Pränexnormalform

$$(\forall x \, p(x)) \vee (\exists y \, q(y))$$

Pränexe Normalform

Theorem. Zu jeder Formel $F \in \mathrm{For}_\Sigma$ gibt es eine äquivalente Formel in Pränexnormalform

Konstruktiver Beweis

(1) Formel in NNF transformieren

1. Elimination von $\leftrightarrow$
2. Elimination von $\rightarrow$
3. "Nach innen schieben" von $\neg$
   - aussagenlogische Umformungen (de Morgans Regeln; $\neg\neg A \equiv A$)
   - $\neg\forall x F(x) \equiv \exists x \neg F(x)$
   - $\neg\exists x F(x) \equiv \forall x \neg F(x)$

(2) Formel bereinigen

(3) Alle Quantoren nach vorne (Reihenfolge unverändert lassen)

Beispiel

$$F = \forall x \Big((\exists y R(x,y) \wedge \forall y \neg S(x,y)) \rightarrow \neg(\exists y R(x,y) \wedge P) \Big)$$

(1) NNF

1.1. Elimination von $\rightarrow$

$$\forall x \Big(\neg(\exists y R(x,y) \wedge \forall y \neg S(x,y)) \vee \neg(\exists y R(x,y) \wedge P) \Big)$$

1.2. "Nach innen schieben" von $\neg$

$$\forall x \Big((\forall y \neg R(x,y) \vee \exists y S(x,y)) \vee (\forall y \neg R(x,y) \vee \neg P) \Big)$$

(2) Formel bereinigen

$$\forall x \Big((\forall y_1 \neg R(x,y_1) \vee \exists y_2 S(x,y_2)) \vee (\forall y_3 \neg R(x,y_3) \vee \neg P) \Big)$$

(3) Alle Quantoren nach vorne (Reihenfolge unverändert lassen)

$$\forall x \forall y_1 \exists y_2 \forall y_3 \Big((\neg R(x,y_1) \vee S(x,y_2)) \vee (\neg R(x,y_3) \vee \neg P) \Big)$$

Beispiel

$$F := (\forall x((p(x) \vee q(x,y)) \wedge \exists z\, r(x,y,z))) \rightarrow ((p(z) \wedge q(x,z)) \wedge \forall z\, r(z,x,y))$$

1. NNF

$$\begin{aligned}
F \equiv\ & \neg(\forall x((p(x) \vee q(x,y)) \wedge \exists z\, r(x,y,z))) \vee ((p(z) \wedge q(x,z)) \wedge \forall z\, r(z,x,y)) && \text{Elim } \rightarrow \\
\equiv\ & \exists x \neg((p(x) \vee q(x,y)) \wedge \exists z\, r(x,y,z))) \vee ((p(z) \wedge q(x,z)) \wedge \forall z\, r(z,x,y)) && (\text{De Morgan}, \neg\forall \equiv \exists\neg) \\
\equiv\ & \exists x(\neg(p(x) \vee q(x,y)) \vee \neg\exists z\, r(x,y,z))) \vee ((p(z) \wedge q(x,z)) \wedge \forall z\, r(z,x,y)) && (\text{De Morgan}) \\
\equiv\ & \exists x((\neg p(x) \wedge \neg q(x,y)) \vee \forall z\, \neg r(x,y,z)) \vee ((p(z) \wedge q(x,z)) \wedge \forall z\, r(z,x,y)) && (\text{De Morgan}, \neg\exists \equiv \forall\neg) \\
=:\ & F_1
\end{aligned}$$

2. Bereinigung

$$F_1 \equiv \exists x_1((\neg p(x_1) \wedge \neg q(x_1,y)) \vee \forall z_1\, \neg r(x_1,y,z_1)) \vee ((p(z) \wedge q(x,z)) \wedge \forall z_2\, r(z_2,x,y)) \quad =: \quad F_2$$

3. Alle Quantoren nach vorne $\mapsto$ Pränexnormalform

$$F_2 \equiv \underbrace{\exists x_1 \forall z_1 \forall z_2\, ((\neg p(x_1) \wedge \neg q(x_1,y)) \vee \neg r(x_1,y,z_1)) \vee ((p(z) \wedge q(x,z)) \wedge r(z_2,x,y))}_{\text{Pränexnormalform von } F}$$

Skolem Normalform

Definition. Eine Formel $F \in \mathrm{For}_\Sigma$ ist in Skolemnormalform (SNF), falls:

- F ist in Pränexnormalform
- F enthält nur universelle Quantoren

Beispiele:

In Skolemnormalform

$$\forall x \forall y (p(x) \vee q(y))$$

Nicht in Skolemnormalform

$$\forall x\, p(x) \vee \forall y\, q(y)$$

$$\forall x \exists y\, (p(x) \vee q(y))$$

Skolemisierung

Skolemisierung: Transformation $\Rightarrow_S$:

$$\forall x_1, \ldots, x_n \exists y F \quad \Rightarrow_S \quad \forall x_1, \ldots, x_n F[f(x_1, \ldots, x_n)/y]$$

wobei f/n ein neues Funktionssymbol (Skolemfunktion).

Skolemisierung: Beispiel

Gegeben:

$\forall w(\exists x(p(w,x) \vee \forall y\,(q(w,x,y) \wedge \exists z\, r(y,z))))$

Pränexnormalform:

$\forall w \exists x \forall y \exists z((p(w,x) \vee (q(w,x,y) \wedge r(y,z))))$

Skolemisierung: $x \mapsto sk_x(w)$, $z \mapsto sk_z(w,y)$

$\forall w \forall y((p(w, sk_x(w)) \vee (q(w, sk_x(w), y) \wedge r(y, sk_z(w,y)))))$

Skolemisierung

Zusammen:

$$F \overset{*}{\Rightarrow}_P \underbrace{G}_{\text{pränexe Form}} \overset{*}{\Rightarrow}_S \underbrace{H}_{\text{pränex, kein } \exists}$$

Theorem:

Seien F, G und H wie oben angenommen. Dann:

(1) F und G sind äquivalent.

(2) G erfüllbar (bzgl. Σ-Str) gdw. H erfüllbar (bzgl. Σ'-Str)

wobei $\Sigma' = (\Omega \cup SKF, \Pi)$, wenn $\Sigma = (\Omega, \Pi)$.

Skolemisierung

Lemma. Sei $G = \forall x_1, \ldots, x_n \exists y G_1$ und $H = \forall x_1, \ldots, x_n G_1[f(x_1, \ldots, x_n)/y]$, wobei f ein neues Funktionssymbol (Skolemfunktion) ist.

G erfüllbar (bezgl. Σ-Str.) genau dann, wenn H erfüllbar (bezgl. Σ'-Str.).

wobei: $\Sigma = (\Omega, \Pi)$ und $\Sigma' = (\Omega \cup \{f\}, \Pi)$.

Beweis: (1) G erfüllbar $\Rightarrow$ H erfüllbar

Sei $\mathcal{A}$ eine Σ-Struktur und $\beta : X \to U_{\mathcal{A}}$ mit $\mathcal{A}(\beta)(G) = 1$.

$$\begin{aligned} 1 = \mathcal{A}(\beta)(G) &= \min_{a_1,\ldots,a_n \in U_{\mathcal{A}}} \mathcal{A}(\beta[x_1 \mapsto a_1, \ldots, x_n \mapsto a_n])(\exists y G_1) \\ &= \min_{a_1,\ldots,a_n \in U_{\mathcal{A}}} \max_{b \in U_{\mathcal{A}}} \mathcal{A}(\beta[x_1 \mapsto a_1, \ldots, x_n \mapsto a_n, y \mapsto b])(G_1) \end{aligned}$$

d.h. für alle $a_1, \ldots, a_n \in U_{\mathcal{A}}$ es gibt ein $b \in U_{\mathcal{A}}$ mit:
$\mathcal{A}(\beta[x_1 \mapsto a_1, \ldots, x_n \mapsto a_n, y \mapsto b])(G_1) = 1$.

Wir definieren eine Σ'-struktur $\mathcal{A}'$, in der alle Funktionen und Prädikate in Σ wie in $\mathcal{A}$ definiert sind und in der wir für alle $a_1, \ldots, a_n \in U_{\mathcal{A}}$, $f(a_1, \ldots, a_n) := b$ definieren.

Dann $\mathcal{A}'(\beta[x_1 \mapsto a_1, \ldots, x_n \mapsto a_n])(G_1[f(x_1, \ldots, x_n)/y]) = 1$.

Skolemisierung

Lemma. Sei $G = \forall x_1, \ldots, x_n \exists y G_1$ und $H = \forall x_1, \ldots, x_n G_1[f(x_1, \ldots, x_n)/y]$, wobei f ein neues Funktionssymbol (Skolemfunktion) ist.

G erfüllbar (bezgl. Σ-Str.) genau dann, wenn H erfüllbar (bezgl. Σ'-Str.).

wobei: $\Sigma = (\Omega, \Pi)$ und $\Sigma' = (\Omega \cup \{f\}, \Pi)$.

Beweis: (2) H erfüllbar $\Rightarrow$ G erfüllbar

Sei $\mathcal{A}'$ eine Σ'-Struktur und $\beta : X \to U_{\mathcal{A}'}$ mit $\mathcal{A}'(\beta)(H) = 1$.

$$\begin{aligned} 1 = \mathcal{A}'(\beta)(H) &= \min_{a_1,\ldots,a_n \in U_{\mathcal{A}'}} \mathcal{A}'(\beta[x_1 \mapsto a_1, \ldots, x_n \mapsto a_n])(G_1[f(x_1, \ldots, x_n)/y]) \\ &= \min_{a_1,\ldots,a_n \in U_{\mathcal{A}'}} (\beta[x_1 \mapsto a_1, \ldots, x_n \mapsto a_n, y \mapsto f_{\mathcal{A}'}(a_1, \ldots, a_n)])(G_1) \end{aligned}$$

d.h. für alle $a_1, \ldots, a_n \in U_{\mathcal{A}'}$ es gibt ein $b = f_{\mathcal{A}'}(a_1, \ldots, a_n) \in U_{\mathcal{A}'}$ mit:
$\mathcal{A}(\beta[x_1 \mapsto a_1, \ldots, x_n \mapsto a_n, y \mapsto b])(G_1) = 1$.

Dann gilt: $\mathcal{A}(\beta)(G) = \min_{a_1,\ldots,a_n \in U_{\mathcal{A}}} \mathcal{A}(\beta[x_1 \mapsto a_1, \ldots, x_n \mapsto a_n])(\exists y G_1)$

$$= \min_{a_1,\ldots,a_n \in U_{\mathcal{A}}} \max_{b \in U_{\mathcal{A}}} \mathcal{A}(\beta[x_1 \mapsto a_1, \ldots, x_n \mapsto a_n, y \mapsto b])(G_1) = 1$$

wobei $\mathcal{A}$ ist $\mathcal{A}'$ ohne die Funktion f.

Skolemisierung

Theorem:

Seien F, G und H wie oben angenommen. Dann:

(1) F und G sind äquivalent.

(2) G erfüllbar (bzgl. Σ-Str) gdw. H erfüllbar (bzgl. Σ'-Str)

wobei $\Sigma' = (\Omega \cup SKF, \Pi)$, wenn $\Sigma = (\Omega, \Pi)$.

Logik für Informatiker

Vorlesung 12: Resolution

Babeş-Bolyai Universität, Department für Informatik, Cluj-Napoca
csacarea@cs.ubbcluj.ro

Klauselnormalform (konjunktive Normalform

Transformationsregeln $\Rightarrow_K$

(1)	$(F \leftrightarrow G)$	$\Rightarrow_K$	$(F \rightarrow G) \wedge (G \rightarrow F)$	
(2)	$(F \rightarrow G)$	$\Rightarrow_K$	$(\neg F \vee G)$	
(3)	$\neg(F \vee G)$	$\Rightarrow_K$	$(\neg F \wedge \neg G)$	
(4)	$\neg(F \wedge G)$	$\Rightarrow_K$	$(\neg F \vee \neg G)$	
(5)	$\neg\neg F$	$\Rightarrow_K$	F	
$(\neg\forall)$	$\neg\forall x F$	$\Rightarrow_K$	$\exists x \neg F$	
$(\neg\exists)$	$\neg\exists x F$	$\Rightarrow_K$	$\forall x \neg F$	(NNF)
(P)	Pränex Normalform			
(S)	Skolemisierung			
(6)	$(F \wedge G) \vee H$	$\Rightarrow_K$	$(F \vee H) \wedge (G \vee H)$	
(7)	$(F \wedge \top)$	$\Rightarrow_K$	F	
(8)	$(F \wedge \bot)$	$\Rightarrow_K$	$\bot$	
(9)	$(F \vee \top)$	$\Rightarrow_K$	$\top$	
(10)	$(F \vee \bot)$	$\Rightarrow_K$	F	(KNF)

Übersicht

$$
\begin{array}{llll}
F & \Rightarrow^*_P & Q_1 y_1 \ldots Q_n y_n \; G & (G \text{ quantorenfrei}) \quad \text{Pränexnormalform} \\
 & \Rightarrow^*_S & \forall x_1, \ldots, x_m \; H & (H \text{ quantorenfrei}) \quad \text{Skolemnormalform} \\
 & \Rightarrow^*_K & \underbrace{\underbrace{\forall x_1, \ldots, x_n}_{\text{weglassen}} \bigwedge_{i=1}^{k} \underbrace{\bigvee_{j=1}^{n_i} L_{ij}}_{\text{Klauseln } C_i}}_{F'} & \text{Skolemnormalform mit Matrix in KNF}
\end{array}
$$

$N = \{C_1, \ldots, C_k\}$ heißt Klausel(normal)form (KNF) von F.
Merke: Die Variablen in Klauseln sind implizit allquantifiziert.
Falls F freie Variablen enthält, werden diese Variablen mit Konstanten ersetzt ($F(x)$ erfüllbar gdw. $\exists x F(x)$ erfüllbar)

Theorem: F ist erfüllbar, gdw. F' erfüllbar, gdw. N erfüllbar.

Viel Optimierungspotential vorhanden, wenn nur Erfüllbarkeit bewahrt werden muß und kann: Größenexplosion, kleine Stelligkeit von Skolemfunktionen.

Beispiel

$$F := \exists z \Big((\forall x(p(u,z,x))) \to (\forall y(q(z,y) \land (\exists x r(y,x)))) \Big)$$

Beispiel

$$F := \exists z\Big(\big(\forall x(p(u,z,x))\big) \to \big(\forall y(q(z,y) \wedge (\exists x r(y,x)))\big)\Big)$$

Pränexnormalform:

$$\begin{aligned}
F &\equiv \exists z\Big(\big(\neg\forall x p(u,z,x)\big) \vee \big(\forall y(q(z,y) \wedge (\exists x r(y,x)))\big)\Big) \\
&\equiv \exists z\Big(\big(\exists x \neg p(u,z,x)\big) \vee \big(\forall y(q(z,y) \wedge (\exists x r(y,x)))\big)\Big) && \text{(NNF)} \\
&\equiv \exists z\Big(\big(\exists x \neg p(u,z,x)\big) \vee \big(\forall y(q(z,y) \wedge (\exists x_1 r(y,x_1)))\big)\Big) && \text{(Bereinigung)} \\
&\equiv \exists z \exists x \forall y \exists x_1 (\neg p(u,z,x) \vee (q(z,y) \wedge r(y,x_1))) && \text{(Pränexnormalform)}
\end{aligned}$$

Beispiel

$$F := \exists z\Big(\big(\forall x(p(u,z,x))\big) \to \big(\forall y(q(z,y) \land (\exists x r(y,x)))\big)\Big)$$

Pränexnormalform:

$$\begin{aligned}
F &\equiv \exists z\Big(\big(\neg\forall x p(u,z,x)\big) \lor \big(\forall y(q(z,y) \land (\exists x r(y,x)))\big)\Big) \\
&\equiv \exists z\Big(\big(\exists x \neg p(u,z,x)\big) \lor \big(\forall y(q(z,y) \land (\exists x r(y,x)))\big)\Big) && \text{(NNF)} \\
&\equiv \exists z\Big(\big(\exists x \neg p(u,z,x)\big) \lor \big(\forall y(q(z,y) \land (\exists x_1 r(y,x_1)))\big)\Big) && \text{(Bereinigung)} \\
&\equiv \exists z \exists x \forall y \exists x_1 (\neg p(u,z,x) \lor (q(z,y) \land r(y,x_1))) && \text{(Pränexnormalform)}
\end{aligned}$$

Skolemisierung $u \mapsto sk_u, z \mapsto sk_z; x \mapsto sk_x; x_1 \mapsto sk_{x_1}(y)$

$$\Rightarrow_S^* \forall y(\neg p(sk_u, sk_z, sk_x) \lor (q(sk_z, y) \land r(y, sk_{x_1}(y))))$$

(Erfüllbarkeitsäquivalente Formel in Skolemnormalform)

Beispiel

$$F := \exists z\Big(\big(\forall x(p(u,z,x))\big) \to \big(\forall y(q(z,y) \wedge (\exists x r(y,x)))\big)\Big)$$

Pränexnormalform:

$$\begin{aligned}
F &\equiv \exists z\Big(\big(\neg\forall x p(u,z,x)\big) \vee \big(\forall y(q(z,y) \wedge (\exists x r(y,x)))\big)\Big) \\
&\equiv \exists z\Big(\big(\exists x \neg p(u,z,x)\big) \vee \big(\forall y(q(z,y) \wedge (\exists x r(y,x)))\big)\Big) && \text{(NNF)} \\
&\equiv \exists z\Big(\big(\exists x \neg p(u,z,x)\big) \vee \big(\forall y(q(z,y) \wedge (\exists x_1 r(y,x_1)))\big)\Big) && \text{(Bereinigung)} \\
&\equiv \exists z \exists x \forall y \exists x_1 (\neg p(u,z,x) \vee (q(z,y) \wedge r(y,x_1))) && \text{(Pränexnormalform)}
\end{aligned}$$

Skolemisierung $u \mapsto sk_u, z \mapsto sk_z; x \mapsto sk_x; x_1 \mapsto sk_{x_1}(y)$

$$\Rightarrow_S^* \ \forall y(\neg p(sk_u, sk_z, sk_x) \vee (q(sk_z, y) \wedge r(y, sk_{x_1}(y))))$$

(Erfüllbarkeitsäquivalente Formel in Skolemnormalform)

Skolemnormalform mit Matrix in KNF:

$$\Rightarrow_K^* \ \forall y((\neg p(sk_u, sk_z, sk_x) \vee q(sk_z, y)) \wedge (\neg p(sk_u, sk_z, sk_x) \vee r(y, sk_{x_1}(y))))$$

Klauselmenge: $N = \{\{\neg p(sk_u, sk_z, sk_x), q(sk_z, y)\}, \{\neg p(sk_u, sk_z, sk_x), r(y, sk_{x_1}(y))\}\}$

Resolution für Grundklauseln

Aussagenlogische Klauseln entsprechen Grundklauseln und umgekehrt.

Resolutionsregel:

$$\frac{C \cup \{A\} \qquad \{\neg A\} \cup D}{C \cup D}$$

$C \cup D$: Resolvente

A: resolviertes Atom

Beispiel

1.	$\{\neg P(f(a)),\quad Q(b)\}$	(gegeben)
2.	$\{P(f(a)),\quad Q(b)\}$	(gegeben)
3.	$\{\neg P(g(b,a)),\quad \neg Q(b)\}$	(gegeben)
4.	$\{P(g(b,a))\}$	(gegeben)
5.	$\{Q(b)\}$	(Res. 2. in 1.)
6.	$\{\neg P(g(b,a))\}$	(Res. 5. in 3.)
8.	$\perp$	(Res. 4. in 6.)

Resolution für Grundklauseln

Resolutionsregel:

$$\frac{C \vee A \qquad \neg A \vee D}{C \vee D}$$

$C \vee D$: Resolvente

A: resolviertes Atom

Faktorisieren:

$$\frac{C \vee L \vee L}{C \vee L}$$

„$\vee$“ wird in Klauseln als assoziativ und kommutativ aufgefaßt.

Beispiel

1.	$\neg P(f(a)) \vee \neg P(f(a)) \vee Q(b)$	(gegeben)
2.	$P(f(a)) \vee Q(b)$	(gegeben)
3.	$\neg P(g(b,a)) \vee \neg Q(b)$	(gegeben)
4.	$P(g(b,a))$	(gegeben)
5.	$\neg P(f(a)) \vee Q(b) \vee Q(b)$	(Res. 2. in 1.)
6.	$\neg P(f(a)) \vee Q(b)$	(Fakt. 5.)
7.	$Q(b) \vee Q(b)$	(Res. 2. in 6.)
8.	$Q(b)$	(Fakt. 7.)
9.	$\neg P(g(b,a))$	(Res. 8. in 3.)
10.	$\perp$	(Res. 4. in 9.)

Korrektheit und Vollständigkeit

- Aussagenlogische Resolution ist korrekt und vollständig.
- Mengennotation: Resolutionsregel
- Klauselnotation: Resolutionsregel + Faktorisieren

Prädikatenlogische Resolution

Grundidee

- Vor Resolutionsschritt durch geeignete Substitution komplementäres Paar von Literalen erzeugen

Prädikatenlogische Resolution

Möglichkeit für Resolutionsregel

$$\frac{C_1 \cup \{L\} \qquad C_2 \cup \{\neg L'\}}{C_1\sigma \cup C_2\sigma}$$

wobei

- die Elternklauseln keine Variablen gemeinsam haben (bereinigt)
  $\mapsto$ ggf. umbenennen
- $\sigma(L) = \sigma(L')$

Prädikatenlogische Resolution

Grundidee

Vor Resolutionsschritt durch geeignete Substitution komplementäres Paar von Literalen erzeugen

Möglichkeit für Resolutionsregel

$$\frac{C_1 \cup \{L\} \qquad C_2 \cup \{\neg L'\}}{C_1\sigma \cup C_2\sigma}$$

wobei

- die Elternklauseln keine Variablen gemeinsam haben (bereinigt)
  $\mapsto$ ggf. umbenennen
- $\sigma(L) = \sigma(L')$

Nachteil: Viel zu viele Substitutionen σ mit $\sigma(L) = \sigma(L')$
Idee: Wähle die "allgemeinste" Substitution, mit $\sigma(L) = \sigma(L')$

Beispiel

$$\frac{\{p(x,f(x)),q(f(x))\} \qquad \{\neg r(f(y)),\neg q(y)\}}{\{p(x,f(x))\sigma,\neg r(f(y))\sigma\}}$$

$L = q(f(x))$, $L' = q(y)$

Nachteil: Viel zu viele Substitutionen σ mit $\sigma(L) = \sigma(L')$:

$\sigma_1 \qquad \sigma_1(x) = a,\ \sigma_1(y) = f(a)$

$\sigma_1 \qquad \sigma_1(x) = f(a),\ \sigma_1(y) = f(f(a))$

$\ldots$

Beispiel

$$\frac{\{p(x,f(x)),q(f(x))\} \qquad \{\neg r(f(y)),\neg q(y)\}}{\{p(x,f(x))\sigma,\neg r(f(y))\sigma\}}$$

$L = q(f(x))$, $L' = q(y)$

Nachteil: Viel zu viele Substitutionen σ mit $\sigma(L) = \sigma(L')$:

$\sigma_1 \qquad \sigma_1(x) = a,\ \sigma_1(y) = f(a)$

$\sigma_1 \qquad \sigma_1(x) = f(a),\ \sigma_1(y) = f(f(a))$

$\ldots$

Idee: Wähle die "allgemeinste" Substitution, mit $\sigma(L) = \sigma(L')$

$q(f(x)) \stackrel{?}{=} q(y) \quad \Rightarrow \quad f(x) \stackrel{?}{=} y \quad \Rightarrow \quad y \stackrel{?}{=} f(x)$

$\sigma = [f(x)/y] \qquad \sigma(y) = f(x),\ \sigma(z) = z$ for $z \neq y$.

Unifikation

Sei $E = \{s_1 \stackrel{?}{=} t_1, \ldots, s_n \stackrel{?}{=} t_n\}$ (s_i, t_i Terme oder Atome) eine Menge von Gleichheitsproblemen.

Definition: Eine Substitution σ heißt ein Unifikator von E g.d.w.

$$\forall 1 \leq i \leq n : s_i\sigma = t_i\sigma.$$

Existiert ein Unifikator, so heißt E unifizierbar.

Definition: σ heißt allgemeiner als τ

$$\sigma \leq \tau \;\; :\Leftrightarrow \;\; \text{es gibt Subst.} \varrho : \sigma \circ \varrho = \tau$$

wobei $(\sigma \circ \varrho)(x) := \varrho(\sigma(x))$ die Komposition von σ und ϱ als Abbildungen.[a]

[a] Ist wohldefiniert, weil $\sigma \circ \varrho$ einen endlichen Bereich hat.

Beispiel

$E = \{q(f(x)) \stackrel{?}{=} q(y)\}$

$\sigma_1 = [a/x, f(a)/y]$ ist ein Unifikator von E

$\sigma_2 = [f(a)/x, f(f(a))/y]$ ist ein Unifikator von E

$\sigma = [f(x)/y]$ ist ein Unifikator von E

σ ist allgemeiner als σ_1 und als σ_2:

$\sigma_1 = \sigma \circ [a/x]$

$$\sigma \circ [a/x](y) = \sigma(y)[a/x] = f(x)[a/x] = f(a)$$
$$\sigma \circ [a/x](x) = \sigma(x)[a/x] = x[a/x] = a$$

$\sigma_2 = \sigma \circ [f(a)/x]$

$$\sigma \circ [f(a)/x](y) = \sigma(y)[f(a)/x] = f(x)[f(a)/x] = f(f(a))$$
$$\sigma \circ [f(a)/x](x) = \sigma(x)[f(a)/x] = x[f(a)/x] = f(a)$$

Fakten

- Jeder Term ist mit sich selbst unifizierbar (mittels id)
- Terme der Gestalt $f(s_1, \ldots, s_n)$, $f(t_1, \ldots, t_n)$ sind unifizierbar g.d.w. si und t_i unifizierbar für $1 \leq i \leq n$
- Atome der Gestalt $p(s_1, \ldots, s_n)$, $p(t_1, \ldots, t_n)$ sind unifizierbar g.d.w. si und t_i unifizierbar für $1 \leq i \leq n$
- Terme der Gestalt $f(s_1, \ldots, s_n)$, $g(t_1, \ldots, t_m)$ sind niemals unifb.
- Atome der Gestalt $p(s_1, \ldots, s_n)$, $q(t_1, \ldots, t_n)$ sind niemals unifb.
- Eine Variable x und ein Term t, der x nicht enthält, sind immer unifb. (mittels $[t/x]$)
- Eine Variable x und ein Term $t \neq x$, der x enthält, sind niemals unifizierbar

Unifikationsalgorithmus von Martelli/Montanari

(1)	$t \stackrel{?}{=} t, E$	$\Rightarrow_{MM}$	E
(2)	$f(s_1, \ldots, s_n) \stackrel{?}{=} f(t_1, \ldots, t_n), E$	$\Rightarrow_{MM}$	$s_1 \stackrel{?}{=} t_1, \ldots, s_n \stackrel{?}{=} t_n, E$
(3)	$f(\ldots) \stackrel{?}{=} g(\ldots), E$	$\Rightarrow_{MM}$	$\perp$
(4)	$x \stackrel{?}{=} t, E$	$\Rightarrow_{MM}$	$x \stackrel{?}{=} t, E[t/x]$ falls $x \in var(E), x \notin var(t)$
(5)	$x \stackrel{?}{=} t, E$	$\Rightarrow_{MM}$	$\perp$ falls $x \neq t, x \in var(t)$
(6)	$t \stackrel{?}{=} x, E$	$\Rightarrow_{MM}$	$x \stackrel{?}{=} t, E$ falls $t \notin X$

Beispiel 1

$$\{f(g(a,x),g(y,b)) \stackrel{?}{=} f(x,g(v,w)), f(x,g(v,w)) \stackrel{?}{=} f(g(x,a),g(v,b)\}$$

$$\stackrel{(2)}{\Rightarrow}_{MM} \quad \{g(a,x) \stackrel{?}{=} x, g(y,b) \stackrel{?}{=} g(v,w), x \stackrel{?}{=} g(x,a), g(v,w) \stackrel{?}{=} g(v,b)\}$$

$$\stackrel{(5)}{\Rightarrow}_{MM} \quad \bot$$

Beispiel 2

$$\{f(g(a,x),g(y,b)) \stackrel{?}{=} g(x,g(v,w)), f(x,g(v,w)) \stackrel{?}{=} f(g(x,a),g(v,b)\}$$

$$\stackrel{(3)}{\Rightarrow}_{MM} \quad \bot$$

Beispiel 3

$$\{f(g(a,x),g(y,b)) \stackrel{?}{=} f(z,g(v,w)), f(z,g(v,w)) \stackrel{?}{=} f(g(x,a),g(v,b)\}$$

$$\stackrel{(2)}{\Rightarrow}_{MM} \quad \{g(a,x) \stackrel{?}{=} z, g(y,b) \stackrel{?}{=} g(v,w), z \stackrel{?}{=} g(x,a), g(v,w) \stackrel{?}{=} g(v,b)\}$$

$$\stackrel{(4)}{\Rightarrow}_{MM} \quad \{z \stackrel{?}{=} g(a,x), g(y,b) \stackrel{?}{=} g(v,w), g(a,x) \stackrel{?}{=} g(x,a), g(v,w) \stackrel{?}{=} g(v,b)\}$$

Beispiel 3

$$\{f(g(a,x),g(y,b)) \stackrel{?}{=} f(z,g(v,w)), f(z,g(v,w)) \stackrel{?}{=} f(g(x,a),g(v,b)\}$$

$$\begin{aligned}
&\stackrel{(2)}{\Rightarrow}_{MM} && \{g(a,x) \stackrel{?}{=} z, g(y,b) \stackrel{?}{=} g(v,w), z \stackrel{?}{=} g(x,a), g(v,w) \stackrel{?}{=} g(v,b)\} \\
&\stackrel{(4)}{\Rightarrow}_{MM} && \{z \stackrel{?}{=} g(a,x), g(y,b) \stackrel{?}{=} g(v,w), g(a,x) \stackrel{?}{=} g(x,a), g(v,w) \stackrel{?}{=} g(v,b)\} \\
&\Rightarrow^{*}_{MM} && \{z \stackrel{?}{=} g(a,x), y \stackrel{?}{=} v, b \stackrel{?}{=} w, a \stackrel{?}{=} x, x \stackrel{?}{=} a, v \stackrel{?}{=} v, w \stackrel{?}{=} b\}
\end{aligned}$$

Beispiel 3

$$\{f(g(a,x),g(y,b)) \stackrel{?}{=} f(z,g(v,w)), f(z,g(v,w)) \stackrel{?}{=} f(g(x,a),g(v,b)\}$$

$$\begin{aligned}
&\stackrel{(2)}{\Rightarrow}_{MM} && \{g(a,x) \stackrel{?}{=} z, g(y,b) \stackrel{?}{=} g(v,w), z \stackrel{?}{=} g(x,a), g(v,w) \stackrel{?}{=} g(v,b)\} \\
&\stackrel{(4)}{\Rightarrow}_{MM} && \{z \stackrel{?}{=} g(a,x), g(y,b) \stackrel{?}{=} g(v,w), g(a,x) \stackrel{?}{=} g(x,a), g(v,w) \stackrel{?}{=} g(v,b)\} \\
&\Rightarrow^*_{MM} && \{z \stackrel{?}{=} g(a,x), y \stackrel{?}{=} v, b \stackrel{?}{=} w, a \stackrel{?}{=} x, x \stackrel{?}{=} a, v \stackrel{?}{=} v, w \stackrel{?}{=} b\} \\
&\Rightarrow^*_{MM} && \{z \stackrel{?}{=} g(a,x), y \stackrel{?}{=} v, b \stackrel{?}{=} w, a \stackrel{?}{=} x, x \stackrel{?}{=} a, w \stackrel{?}{=} b\}
\end{aligned}$$

Beispiel 3

$$\{f(g(a,x),g(y,b)) \stackrel{?}{=} f(z,g(v,w)), f(z,g(v,w)) \stackrel{?}{=} f(g(x,a),g(v,b)\}$$

$$\begin{aligned}
&\stackrel{(2)}{\Rightarrow}_{MM} && \{g(a,x) \stackrel{?}{=} z, g(y,b) \stackrel{?}{=} g(v,w), z \stackrel{?}{=} g(x,a), g(v,w) \stackrel{?}{=} g(v,b)\} \\
&\stackrel{(4)}{\Rightarrow}_{MM} && \{z \stackrel{?}{=} g(a,x), g(y,b) \stackrel{?}{=} g(v,w), g(a,x) \stackrel{?}{=} g(x,a), g(v,w) \stackrel{?}{=} g(v,b) \\
&\Rightarrow^*_{MM} && \{z \stackrel{?}{=} g(a,x), y \stackrel{?}{=} v, b \stackrel{?}{=} w, a \stackrel{?}{=} x, x \stackrel{?}{=} a, v \stackrel{?}{=} v, w \stackrel{?}{=} b\} \\
&\Rightarrow^*_{MM} && \{z \stackrel{?}{=} g(a,x), y \stackrel{?}{=} v, b \stackrel{?}{=} w, a \stackrel{?}{=} x, x \stackrel{?}{=} a, w \stackrel{?}{=} b\} \\
&\Rightarrow^*_{MM} && \{z \stackrel{?}{=} g(a,a), y \stackrel{?}{=} v, b \stackrel{?}{=} b, a \stackrel{?}{=} a, x \stackrel{?}{=} a, w \stackrel{?}{=} b\} \\
&\Rightarrow^*_{MM} && \{z \stackrel{?}{=} g(a,a), y \stackrel{?}{=} v, x \stackrel{?}{=} a, w \stackrel{?}{=} b\}
\end{aligned}$$

Beispiel 3

$$\{f(g(a,x),g(y,b)) \stackrel{?}{=} f(z,g(v,w)), f(z,g(v,w)) \stackrel{?}{=} f(g(x,a),g(v,b)\}$$

$$\begin{aligned}
&\stackrel{(2)}{\Rightarrow}_{MM} && \{g(a,x) \stackrel{?}{=} z, g(y,b) \stackrel{?}{=} g(v,w), z \stackrel{?}{=} g(x,a), g(v,w) \stackrel{?}{=} g(v,b)\}\\
&\stackrel{(4)}{\Rightarrow}_{MM} && \{z \stackrel{?}{=} g(a,x), g(y,b) \stackrel{?}{=} g(v,w), g(a,x) \stackrel{?}{=} g(x,a), g(v,w) \stackrel{?}{=} g(v,b)\\
&\Rightarrow^*_{MM} && \{z \stackrel{?}{=} g(a,x), y \stackrel{?}{=} v, b \stackrel{?}{=} w, a \stackrel{?}{=} x, x \stackrel{?}{=} a, v \stackrel{?}{=} v, w \stackrel{?}{=} b\}\\
&\Rightarrow^*_{MM} && \{z \stackrel{?}{=} g(a,x), y \stackrel{?}{=} v, b \stackrel{?}{=} w, a \stackrel{?}{=} x, x \stackrel{?}{=} a, w \stackrel{?}{=} b\}\\
&\Rightarrow^*_{MM} && \{z \stackrel{?}{=} g(a,a), y \stackrel{?}{=} v, b \stackrel{?}{=} b, a \stackrel{?}{=} a, x \stackrel{?}{=} a, w \stackrel{?}{=} b\}\\
&\Rightarrow^*_{MM} && \{z \stackrel{?}{=} g(a,a), y \stackrel{?}{=} v, x \stackrel{?}{=} a, w \stackrel{?}{=} b\}
\end{aligned}$$

Allgemeinster Unifikator:

$$[g(a,a)/z, v/y, a/x, b/w]$$

Vorsicht: a, b sind Konstanten. $[g(a,a)/z, v/y, x/a, w/b]$ ist keine Substitution!

Beispiel 4

$\Sigma = \{\Omega, \Pi\} \quad \Omega = \{f/2, g/2, a/0, b/0\}, \Pi = \{p/2, q/1\}$
$x, y, z \in X$

$$\{p(g(x,a), g(f(x,b),y)) \stackrel{?}{=} q(g(f(x,b),y))\} \quad \stackrel{(3)}{\Rightarrow}_{MM} \bot \qquad \text{(weil } p \neq q\text{)}$$

Beispiel 4

$$\Sigma = \{\Omega, \Pi\} \quad \Omega = \{f/2, g/2, a/0, b/0\}, \Pi = \{p/2, q/1\}, \quad x, y, z, u \in X$$

$$\{p(g(y,a), g(f(x,b),z)) \stackrel{?}{=} p(z, g(f(g(u,y),b),u), \quad q(a) \stackrel{?}{=} q(y))\}$$

$$\begin{array}{ll}
\stackrel{(2)}{\Rightarrow}_{MM} & \{g(y,a) \stackrel{?}{=} z, \quad g(f(x,b),z) \stackrel{?}{=} g(f(g(u,y),b),u), \quad q(a) \stackrel{?}{=} q(y)\} \\
\stackrel{(6,2)}{\Rightarrow}_{MM} & \{z \stackrel{?}{=} g(y,a), \quad f(x,b) \stackrel{?}{=} f(g(u,y),b), \quad z \stackrel{?}{=} u, \quad q(a) \stackrel{?}{=} q(y)\} \\
\stackrel{(4)}{\Rightarrow}_{MM} & \{z \stackrel{?}{=} g(y,a), \quad f(x,b) \stackrel{?}{=} f(g(u,y),b), g(y,a) \stackrel{?}{=} u, \quad q(a) \stackrel{?}{=} q(y)\} \\
\stackrel{(2,6)}{\Rightarrow}_{MM} & \{z \stackrel{?}{=} g(y,a), \quad x \stackrel{?}{=} g(u,y), b \stackrel{?}{=} b, \quad u \stackrel{?}{=} g(y,a), \quad q(a) \stackrel{?}{=} q(y)\} \\
\stackrel{(1)}{\Rightarrow}_{MM} & \{z \stackrel{?}{=} g(y,a), \quad x \stackrel{?}{=} g(u,y), \quad u \stackrel{?}{=} g(y,a), \quad q(a) \stackrel{?}{=} q(y)\} \\
\stackrel{(4)}{\Rightarrow}_{MM} & \{z \stackrel{?}{=} g(y,a), \quad x \stackrel{?}{=} g(g(y,a),y), \quad u \stackrel{?}{=} g(y,a), \quad g(a) \stackrel{?}{=} g(y)\} \\
\stackrel{(2)}{\Rightarrow}_{MM} & \{z \stackrel{?}{=} g(y,a), \quad x \stackrel{?}{=} g(g(y,a),y), \quad u \stackrel{?}{=} g(y,a), \quad a \stackrel{?}{=} y\} \\
\stackrel{(6)}{\Rightarrow}_{MM} & \{z \stackrel{?}{=} g(y,a), \quad x \stackrel{?}{=} g(g(y,a),y), \quad u \stackrel{?}{=} g(y,a), \quad y \stackrel{?}{=} a\} \\
\stackrel{(4)}{\Rightarrow}_{MM} & \{z \stackrel{?}{=} g(a,a), \quad x \stackrel{?}{=} g(g(a,a),a), \quad u \stackrel{?}{=} g(a,a), \quad y \stackrel{?}{=} a\}
\end{array}$$

Allgemeinster Unifikator (mgu): $[g(a,a)/z, \quad g(g(a,a),a)/x, \quad g(a,a)/u, \quad a/y]$

Unifikation: Haupteigenschaften

Definition. Eine Substitition σ heißt idempotent, wenn $\sigma \circ \sigma = \sigma$.

Lemma.
σ ist idempotent gdw. $dom(\sigma) \cap codom(\sigma) = \emptyset$.

Theorem.

1. $E \Rightarrow^*_{MM} \bot$ gdw. E nicht unfizierbar.
2. E unifizierbar gdw. $E \Rightarrow^*_{MM} x_1 \stackrel{?}{=} u_1, \ldots, x_k \stackrel{?}{=} u_k$,
   mit x_i pw. verschieden, $x_i \notin var(u_j), 1 \leq i, j \leq k$.
3. Falls $E \Rightarrow^*_{MM} x_1 \stackrel{?}{=} u_1, \ldots, x_k \stackrel{?}{=} u_k$,
   mit x_i pw. verschieden, $x_i \notin var(u_j)$ so
   $\sigma = [u_1/x_1, \ldots, u_k/x_k]$ ist allgemeinster Unifikator von E.

Unifikation: Haupteigenschaften

Theorem.
E unifizierbar g.d.w. es gibt allgemeinsten Unifikator σ von E, so dass:
(1) σ idempotent und
(2) $dom(\sigma) \cup codom(\sigma) \subseteq var(E)$.

Notation: $\sigma = \text{mgu}(E)$ („most general unifier“)

Prädikatenlogische Resolution

Grundidee

Vor Resolutionsschritt durch geeignete Substitution komplementäres Paar von Literalen erzeugen

Möglichkeit für Resolutionsregel

$$\frac{C_1 \cup \{L\} \qquad C_2 \cup \{\neg L'\}}{C_1\sigma \cup C_2\sigma}$$

wobei

- die Elternklauseln keine Variablen gemeinsam haben (bereinigt)
  $\mapsto$ ggf. umbenennen
- $\sigma = \mathrm{mgu}(L, L')$

Beispiel

$$\{L\} \cup C_1 = \{\underbrace{p(a,x)}_{L}, p(x,x)\} \qquad \{\neg L'\} \cup C_2 = \{\neg \underbrace{p(y,y)}_{L'}\}$$

Allgemeinster Unifikator von L, L':

$$\begin{aligned} \{p(a,x) \stackrel{?}{=} p(y,y)\} &\Rightarrow_{MM} \{a \stackrel{?}{=} y, x \stackrel{?}{=} y\} \\ &\Rightarrow_{MM} \{y \stackrel{?}{=} a, x \stackrel{?}{=} a\} \end{aligned}$$

mgu(L, L'): $\quad \sigma = [a/y, a/x]$

Beispiel

$\{L\} \cup C_1 = \{\underbrace{p(a,x)}_{L}, p(x,x)\}$ $\qquad$ $\{\neg L'\} \cup C_2 = \{\neg \underbrace{p(y,y)}_{L'}\}$

Allgemeinster Unifikator von L, L':

$$\begin{aligned} \{p(a,x) \stackrel{?}{=} p(y,y)\} &\Rightarrow_{MM} \{a \stackrel{?}{=} y, x \stackrel{?}{=} y\} \\ &\Rightarrow_{MM} \{y \stackrel{?}{=} a, x \stackrel{?}{=} a\} \end{aligned}$$

mgu(L, L'): $\quad \sigma = [a/y, a/x]$

$$\frac{C_1 \cup \{L\} \qquad C_2 \cup \{\neg L'\}}{C_1\sigma \cup C_2\sigma}$$

$R := \{p(x,x)\}\sigma \cup \{\}\sigma = \{p(a,a)\}$

Beispiel

$$\{L\} \cup C_1 = \{\underbrace{p(a,x)}_{L}, p(x,x)\} \qquad \{\neg L'\} \cup C_2 = \{\neg \underbrace{p(y,y)}_{L'}\}$$

Allgemeinster Unifikator von L, L':

$$\begin{aligned} \{p(a,x) \stackrel{?}{=} p(y,y)\} &\Rightarrow_{MM} \{a \stackrel{?}{=} y, x \stackrel{?}{=} y\} \\ &\Rightarrow_{MM} \{y \stackrel{?}{=} a, x \stackrel{?}{=} a\} \end{aligned}$$

mgu(L, L'): $\quad \sigma = [a/y, a/x]$

$$\frac{C_1 \cup \{L\} \qquad C_2 \cup \{\neg L'\}}{C_1\sigma \cup C_2\sigma}$$

$R := \{p(x,x)\}\sigma \cup \{\}\sigma = \{p(a,a)\}$

$$\frac{\{p(a,x), p(x,x)\} \qquad \{\neg p(y,y)\}}{\{p(a,a)\}}$$

Prädikatenlogische Resolution

Resolutionsregel in dieser Form alleine unvollständig für Prädikatenlogik

Beispiel:

$\{\{p(x), p(y)\}, \{\neg p(u), \neg p(v)\}\}$

- unerfüllbar
- aber nur Resolventen der Länge 2

Logik für Informatiker

Vorlesung 13, Teil 2: Resolution

Babeş-Bolyai Universität, Department für Informatik, Cluj-Napoca
csacarea@cs.ubbcluj.ro

Terminierung und Korrektheit des Unifikationsalgorithmus

Satz

Der Unifikationsalgorithmus terminiert für jede Klausel K und er ist korrekt, d.h., er liefert einen mgu für die Klausel K gdw. K unifizierbar ist.

Beweis

- Die Terminierung des Algorithmus folgt, da in jedem Durchlauf der Schleife von Schritt 2 - 6 die Zahl der Variablen in $\sigma(K)$ um 1 abnimmt.
- Falls der Algorithmus mit Erfolg terminiert und eine Substitution σ ausgibt, so ist σ offensichtlich ein Unifikator von K, da $\mid \sigma(K) \mid = 1$ gilt.
- Sofern die Klausel K also nicht unifizierbar ist, muss der Algorithmus daher mit einem Clash oder Occur Failure abbrechen.

Beweis

- Es bleibt zu zeigen, dass bei jeder unifizierbaren Klausel auch tatsächlich ein Unifikator gefunden wird und dass dieser Unifikator dann auch ein allgemeinster Unifikator ist.
- Sei $m \geq 0$ die Anzahl der Schleifendurchläufe, die bei Eingabe der Klausel K stattfinden.
- Für alle $0 \leq i \leq m$ sei σ_i der Wert von σ nach dem i-ten Schleifendurchlauf.
- Wir zeigen die folgende Behauptung für alle $0 \leq i \leq m$:
  (*) Für jeden Unifikator σ' von K gilt $\sigma' = \sigma' \circ \sigma_i$.

Beweis

- Aus der Behauptung (*) folgt, dass zum Schluss nicht mit Clash oder Occur Failure abgebrochen werden kann.
- Denn wurde im (m + 1)-ten Schleifendurchlauf abgebrochen, so wäre $\sigma_m(K)$ nicht unifizierbar.
- Da aber K unifizierbar ist, hat K einen Unifikator $\sigma' = \sigma' \circ \sigma_m$. Somit ist σ' auch Unifikator von $\sigma_m(K)$.
- Da die Schleife nur m mal durchlaufen wird, muss dann also $\mid \sigma_m(K) \mid = 1$ gelten, d.h., σ_m ist Unifikator von K.
- Da außerdem für jeden Unifikator σ_0 von K eine Substitution $\delta = \sigma_0$ mit $\sigma_0 = \delta \circ \sigma_m$ existiert, ist σ_m dann auch mgu von K.

Beweis

- Wir beweisen nun die Behauptung (*) durch Induktion über i.
- Im Induktionsanfang $i = 0$ ist $\sigma_0 = \emptyset$ die Identität. Daher gilt dann auch $\sigma' = \sigma' \circ \sigma_0$ für alle σ'.
- Im Induktionsschritt $i > 0$ wird im i-ten Schleifendurchlauf eine Variable X im einen Literal und ein Term t im anderen Literal gefunden und es ergibt sich $\sigma_i = \{X/t\} \circ \sigma_{i-1}$.
- Für jeden Unifikator σ' von K gilt nach der Induktionshypothese $\sigma' = \sigma' \circ \sigma_{i-1}$.
- Damit folgt:

Beweis

- $\sigma' \circ \sigma_i$
  $= \sigma' \circ \{X/t\} \circ \sigma_{i-1}$ (Def. von σ_i)
  $= \sigma' \circ \sigma_{i-1}$ (da $\sigma' \circ \{X/t\} = \sigma'$)

Beweis

- Um $\sigma' \circ \{X/t\} = \sigma'$ zu zeigen, erkennt man zunächst, dass die beiden Substitutionen auf allen Variablen $Y \neq X$ offensichtlich identisch sind.
- Bei der Variablen X ergibt sich

$$(\sigma' \circ \{X/t\})(X) = \sigma'(t) = \sigma'(X),$$

da σ' Unifikator von $\sigma_{i-1}(K)$ ist (denn $\mid \sigma'(K) \mid = \mid \sigma'(\sigma_{i-1}(K)) \mid = 1$) und jeder Unifikator von $\sigma_{i-1}(K)$ auch die Terme X und t unifizieren muss.

Prädikatenlogische Resolution

Definition

Seien K_1 und K_2 Klauseln. Dann ist die Klausel R Resolvent von K_1 und K_2 gdw. die folgenden drei Bedingungen gelten:

- *Es gibt Variablenumbenennungen v_1 und v_2, so dass $v_1(K_1)$ und $v_2(K_2)$ keine gemeinsamen Variablen enthalten.*
- *Es gibt Literale $L_1, \ldots, L_m \in v_1(K_1)$ und Literale $L'_1, \ldots, L'_n \in v_2(K_2)$ mit $n, m \geq 1$, so dass $\{L_1, \ldots, L_m, L'_1, \ldots, L'_n\}$ mit einem mgu σ unifizierbar ist.*
- $R = \sigma((v_1(K_1) \setminus \{L_1, \ldots, L_m\} \cup (v_2(K_2) \setminus \{L'_1, \ldots, L'_n\}))$

Prädikatenlogische Resolution

- Für eine Klauselmenge $\mathcal{K}$ definieren wir wie bei der aussagenlogischen Resolution:
- $Res(\mathcal{K}) = \mathcal{K} \cup \{R \mid R \text{ ist Resolvent zweier Klauseln aus } K\}$
- $Res^0(\mathcal{K}) = \mathcal{K}$
- $Res^{n+1}(\mathcal{K}) = Res(Res^n(\mathcal{K}))$ für alle $n \geq 0$
- $Res^*(\mathcal{K}) = \bigcup_{n \geq 0} Res^n(\mathcal{K})$

Prädikatenlogische Resolution

- Offensichtlich ist die aussagenlogische Resolution ein Spezialfall der prädikatenlogischen Resolution, denn bei variablenfreien Klauseln fällt die Definition der prädikatenlogischen Resolution mit der Definition der aussagenlogischen Resolution zusammen.
- Analog zur aussagenlogischen Resolution gilt $\square \in Res(\mathcal{K})$ gdw. es eine Folge von Klauseln $K_1, \ldots, K_m$ gibt, so dass $K_m = \square$ ist und so dass für alle $1 \leq i \leq m$ gilt:
  - $K_i \in \mathcal{K}$ oder
  - K_i ist eine Resolvente von K_j und K_t für $j, t < i$.

Prädikatenlogische Resolution

Zur Darstellung von Resolutionsbeweisen schreiben wir wieder das folgende Diagramm, um deutlich zu machen, dass R durch Resolution aus K_1 und K_2 entsteht.

$$\begin{tikzcd} K_1 \arrow[dr, dash] & & K_2 \arrow[dl, dash] \\ & R & \end{tikzcd}$$

Eine genauere Darstellung ist möglich, indem man die eliminierten Literale unterstreicht und die Umbenennungen und den Unifikator explizit angibt.

Beispiel

$p, q, r \in \Delta_1, f, g \in \Sigma_1$

$\{\underline{\mathsf{p}(\mathsf{f}(X))}, \neg\mathsf{q}(Z), \underline{\mathsf{p}(Z)}\}$

$\{\underline{\neg\mathsf{p}(X)}, \mathsf{r}(\mathsf{g}(X))\}$

$\{\neg\mathsf{q}(\mathsf{f}(X)), \mathsf{r}(\mathsf{g}(\mathsf{f}(X)))\}$

$\nu_1 = \varnothing$
$\nu_2 = \{X/U, U/X\}$
$\sigma = \{Z/\mathsf{f}(X), U/\mathsf{f}(X)\}$

Prädikatenlogisches Resolutionslemma

Lemma

Sei $\mathcal{K}$ eine Menge von Klauseln. Falls $K_1, K_2 \in \mathcal{K}$ und R Resolvente von K_1 und K_2 ist, dann sind $\mathcal{K}$ und $\mathcal{K} \cup \{R\}$ äquivalent.

Beweis

- Wie in der Aussagenlogik folgt aus $S \models \mathcal{K} \cup \{R\}$ trivialerweise $S \models \mathcal{K}$ für alle Strukturen S.
- Es genügt, hierbei Strukturen statt Interpretationen zu betrachten, da jede Klausel die allquantifizierte Disjunktion ihrer Literale repräsentiert und jede Klauselmenge die Konjunktion ihrer Klauseln.
- Somit gilt $\mathcal{K} \cup \{R\} \models \mathcal{K}$.

Beweis

- Umgekehrt sei nun S eine Struktur, die $\mathcal{K}$ erfüllt.
- Es gilt
  $R = \sigma((v_1(K_1) \setminus \{L_1, \ldots, L_m\} \cup (v_2(K_2) \setminus \{L'_1, \ldots, L'_n\}))$.
  Hierbei sind v_1 und v_2 Variablenumbenennungen, so dass $v_1(K_1)$ und $v_2(K_2)$ keine gemeinsamen Variablen enthalten.
- Außerdem sind $L_1, \ldots, L_m \in v_1(K_1)$ und $'_1, \ldots, L'_n \in v_2(K_2)$ mit $n, m \geq 1$, so dass $\{L_1, \ldots, L_m, L'_1, \ldots, L'_n\}$ mit dem mgu σ unifizierbar sind.
- Es gilt also $\sigma(L_1) = \cdots = \sigma(L_m) = L$ und $\sigma(L'_1) = \cdots = \sigma(L'_n) = L$ für ein Literal L.

Beweis

Wir nehmen an, dass $S \not\models \mathcal{K} \cup \{R\}$ gilt. Aus $S \models \mathcal{K}$ folgt dann $S \not\models R$. Sei

$$\nu_1(K_1) = \{L_1, \ldots, L_m, L_{m+1}, \ldots, L_p\} \qquad \text{und} \qquad \nu_2(K_2) = \{L'_1, \ldots, L'_n, L'_{n+1}, \ldots, L'_q\}$$

mit $p \geq m$ und $q \geq n$. Dann entsteht R durch Allquantifizierung der Formel

$$\sigma(L_{m+1} \vee \ldots \vee L_p \vee L'_{n+1} \vee \ldots \vee L'_q).$$

Sei $S = (\mathcal{A}, \alpha)$. Es existiert also eine Interpretation $I = (\mathcal{A}, \alpha, \beta)$ mit

$$I \not\models \sigma(L_{m+1} \vee \ldots \vee L_p \vee L'_{n+1} \vee \ldots \vee L'_q).$$

Beweis

- Sei $\sigma = \{X_1/t_1, \ldots, X_k/t_k\}$ und sei I' die Interpretation $I[\![X_1/I(t_1), \ldots, X_k/I(t_k)]\!]$.
- Nach dem Substitutionslemma gilt dann

$$(**) \qquad I' \not\models L_{m+1} \vee \cdots \vee L_p \vee L'_{n+1} \vee \cdots \vee L'_q.$$

- Da aber $S \models K_1$ und $S \models K_2$ gilt, folgt auch $S \models v_1(K_1)$ und $S \models v_2(K_2)$ und somit $I' \models L_1 \vee \cdots \vee L_m \vee L_{m+1} \vee \cdots \vee L_p$ und $I' \models L'_1 \vee \cdots \vee L'_n \vee L'_{n+1} \vee \cdots \vee L'_q$.

Beweis

- Mit (**) folgt $I' \vDash L_1 \vee \cdots \vee L_m$ und $I' \vDash L'_1 \vee \cdots \vee L'_n$
- Mit dem Substitutionslemma ergibt sich $I \vDash \sigma(L_1 \vee \cdots \vee L_m)$ und $I \vDash \sigma(L'_1 \vee \cdots \vee L'_n)$.
- Wir erhalten also den folgenden Widerspruch: $I \vDash L$ und $I \vDash \overline{L}$.

Resolutionsalgorithmus

- Überführe $\mathcal{K}$ in KNF, bzw. in die entsprechende Klauselmenge $\mathcal{K}(\psi)$.
- Berechne $Res^*(\mathcal{K}(\psi))$. Sofern die leere Klausel gefunden wurde, brich ab und gib true zurück. Sofern $Res(\mathcal{K}(\psi))$ komplett berechnet wurde, brich ab und gib false zurück.

- Dieser Semi-Entscheidungsalgorithmus kann auch mit false abbrechen, denn falls $Res^*(\{K(\psi)\})$ endlich ist und die leere Klausel nicht enthält, folgt die Erfüllbarkeit der Klauselmenge $K(\psi)$ aus der Vollständigkeit des Resolutionskalküls.
- Im Allgemeinen ist $Res^*(\{K(\psi)\})$ allerdings unendlich und dann terminiert der obige Algorithmus bei erfüllbaren Klauselmengen $K(\psi)$ nicht.

Die Effizienz dieses Algorithmus läßt sich allerdings noch weiter verbessern, denn bislang müssen noch alle möglichen Resolutionsschritte zwischen allen Klauseln betrachtet werden. Hierbei müssen natürlich auch die Klauseln betrachtet werden, die im Laufe der Resolution entstanden sind.

Einschränkungen der Resolution

- Das Problem beim Unerfüllbarkeitsnachweis durch Resolution ist, dass es sehr viele Resolutionsmöglichkeiten gibt, was zu einer kombinatorischen Explosion führt.
- Unser Ziel ist daher, den Suchraum durch Verwendung spezieller Resolutionsstrategien zu verkleinern, ohne jedoch dabei die Vollständigkeit zu verlieren.
- Lineare Resolution
- Input Resolution: vollständig nur für Horn Klauselmengen.
- SLD Resolution: wird bei der Logikprogrammierung verwendet.

Lineare Resolution

- Die Idee der linearen Resolution besteht darin, dass bei jedem Resolutionsschritt eine der beiden Elternklauseln der zuletzt erzeugte Resolvent sein muss.

Lineare Resolution

Definition

Sei $\mathcal{K}$ eine Klauselmenge. Die leere Klausel $\square$ ist aus der Klausel K in $\mathcal{K}$ linear resolvierbar gdw. es eine Folge von Klauseln $K_1, \ldots, K_m$ gibt, so dass $K_1 = K \in \mathcal{K}$ und $K_m = \square$ ist und so dass für alle $2 \leq i \leq m$ gilt: K_i ist ein Resolvent von K_{i-1} und einer Klausel aus $\{K_1, \ldots, K_{i-1}\} \cup \mathcal{K}$.

Beispiel

Diese Herleitung entspricht aber keinem linearen Resolutionsbeweis, denn wenn man im ersten Schritt den Resolvent $\{q\}$ *bildet, so müsste dieser Resolvent dann eine der Elternklauseln im nächsten Schritt sein. Eine Herleitung der leeren Klausel mit linearer Resolution wäre in diesem Beispiel wie folgt möglich:*

Korrektheit und Vollständigkeit der linearen Resolution

Sei $\mathcal{K}$ eine Menge von Klauseln. Dann ist $\mathcal{K}$ unerfüllbar gdw. $\square$ aus einer Klausel K in $\mathcal{K}$ linear resolvierbar ist. Falls $\mathcal{K}$ eine minimale unerfüllbare Menge ist (d.h., falls für jedes $K \in \mathcal{K}$ die Menge $\mathcal{K} \setminus \{\mathcal{K}\}$ erfüllbar ist), dann ist $\square$ sogar aus jeder Klausel K in $\mathcal{K}$ linear resolvierbar.

Input Resolution

- Um die Möglichkeiten der Resolution noch weiter zu reduzieren, schränken wir die lineare Resolution nun noch weiter ein.
- Bei der linearen Resolution muss eine der Elternklauseln in jedem Resolutionsschritt der letzte Resolvent sein. Die andere Elternklausel konnte jedoch noch frei gewählt werden (d.h., es konnte eine Klausel aus der ursprünglichen Klauselmenge oder aber ein bereits früher gebildeter Resolvent sein).
- Wir verbieten nun die letzte Möglichkeit: nun muss in jedem Schritt zwischen dem zuletzt gebildeten Resolvent und einer der ursprünglichen Eingabe - Klauseln resolviert werden.
- Aus diesem Grund bezeichnet man diese Einschränkung als Input - Resolution.

Definition

Sei $\mathcal{K}$ eine Klauselmenge. Die leere Klausel $\square$ ist aus der Klausel K in $\mathcal{K}$ durch Input - Resolution herleitbar gdw. es eine Folge von Klauseln $K_1, \ldots, K_m$ gibt, so dass $K_1 = K \in \mathcal{K}$ und $K_m = \square$ ist und so dass für alle $2 \leq i \leq m$ gilt: K_i ist ein Resolvent von K_{i-1} und einer Klausel aus $\mathcal{K}$.

Aus der Definition folgt sofort, dass die Input - Resolution ein Spezialfall der linearen Resolution ist, d.h., jeder Input - Resolutionsbeweis ist auch ein linearer Resolutionsbeweis. Der Umkehrschluss gilt natürlich nicht. In der Tat ist die Input - Resolution im Unterschied zur linearen Resolution nicht mehr vollständig, wie das folgende Beispiel zeigt.

Beispiel

- Wir betrachten das Beispiel auf S. 54:
- Während wir dort die Unerfüllbarkeit mit linearer Resolution nachweisen konnten, gelingt dies mit Input - Resolution nicht. Durch Resolution zweier Klauseln aus der ursprünglichen Klauselmenge lassen sich nur die folgenden Klauseln herleiten:

$$\{q\}, \{\neg q\}, \{p\}, \{\neg p\}, \{q, \neg q\}, \{p, \neg p\}.$$

Beispiel

- Falls man eine der ersten vier entstandenen Klauseln wieder mit einer Klausel der ursprünglichen Menge resolviert, so entsteht wieder eine der ersten vier Klauseln. Wenn man eine der letzten beiden (allgemeingültigen) Klauseln mit einer Klausel der ursprünglichen Menge resolviert, so erhält man die Klausel der ursprünglichen Menge.
- Eine Herleitung der leeren Klausel ist also nur dann möglich, wenn man zwei der durch Resolution entstandenen Klauseln (wie z.B. $\{q\}$ und $\{\neg q\}$) miteinander resolviert. Dies ist jedoch bei Input - Resolution nicht zulässig.

- Die Input-Resolution ist auf Horn-Klauseln vollständig!

Horn-Klauseln

Definition

Eine Klausel K ist eine Hornklausel gdw. sie höchstens ein positives Literal enthält (d.h., höchstens eines ihrer Literale ist eine atomare Formel und die anderen Literale sind negierte atomare Formeln). Eine Hornklausel heißt negativ, falls sie nur negative Literale enthält (d.h., falls sie die Gestalt $\{\neg A_1, \ldots, \neg A_k\}$ für atomare Formeln $A_1, \ldots, A_k$ hat). Eine Hornklausel heißt definit, falls sie ein positives Literal enthält (d.h., falls sie die Gestalt $\{B, \neg C_1, \ldots, \neg C_n\}$ für atomare Formeln $B, C_1, \ldots, C_n$ hat).

- Eine Menge definiter Hornklauseln entspricht einer Konjunktion von Implikationen
- $\{\{p, \neg q\}, \{\neg r, \neg p, s\}, \{s\}\}$ ist äquivalent zur Formel

$$((p \vee \neg q) \wedge (\neg r \vee \neg p \vee s) \wedge s)$$

und damit zur folgenden Formel

$$(q \rightarrow p) \wedge (r \wedge p \rightarrow s) \wedge s).$$

Zusammenhang zur Logikprogrammierung

- Fakten sind definite Hornklauseln ohne negative Literale (d.h., sie enthalten genau ein positives Literal). Ein Beispiel ist die Klausel $\{s\}$. In der Logikprogrammierung schreibt man: `s.`
- Regeln sind definite Hornklauseln mit negativen Literalen. Ein Beispiel ist die Klausel $\{\neg r, \neg p, s\}$. In der Logikprogrammierung schreibt man: `s :- r, p.`
- Anfragen sind negative Hornklauseln. Ein Beispiel ist die Klausel $\{\neg p, \neg q\}$. In der Logikprogrammierung schreibt man: ?- p,q.

SLD-Resolution

Definition

Sei $\mathcal{K}$ eine Hornklauselmenge mit $\mathcal{K} = \mathcal{K}^d \uplus \mathcal{K}^n$, wobei $\mathcal{K}^d$ die definiten Klauseln und $\mathcal{K}^n$ die negativen Klauseln von $\mathcal{K}$ enthält. Die leere Klausel $\square$ ist aus der Klausel K in $\mathcal{K}^n$ durch SLD-Resolution herleitbar gdw. es eine Folge von Klauseln $K_1, \ldots, K_m$ gibt, so dass $K_1 = K \in \mathcal{K}^n$ und $K_m = \square$ ist und so dass für alle $2 \leq i \leq m$ gilt: K_i ist ein Resolvent von K_{i-1} und einer Klausel aus K_d.

Man erkennt sofort, dass in einer SLD–Resolution alle Klauseln $K_1, \ldots, K_m$ negativ sind. SLD–Resolutionen haben also die folgende Gestalt:

Hierbei sind $K_1, \ldots, K_{n+1} \in \mathcal{K}^d$ definite Hornklauseln aus der Eingabemenge, $N \in \mathcal{K}^n$ ist eine negative Hornklausel aus der Eingabemenge und die Resolventen $R_1, \ldots, R_n$ sind ebenfalls negative Hornklauseln.

Korrektheit und Vollständigkeit der SLD-Resolution

Sei $\mathcal{K}$ eine Menge von Hornklauseln. Dann ist $\mathcal{K}$ unerfüllbar gdw. $\square$ aus einer negativen Klausel N in $\mathcal{K}$ durch SLD - Resolution herleitbar ist.

Logik für Informatiker

Vorlesung 13, Teil 1: Resolution

Babeş-Bolyai Universität, Department für Informatik, Cluj-Napoca
csacarea@cs.ubbcluj.ro

Prädikatenlogische Resolution

Wiederholung

Grundidee

- Vor Resolutionsschritt durch geeignete Substitution komplementäres Paar von Literalen erzeugen

Prädikatenlogische Resolution

Wiederholung

Möglichkeit für Resolutionsregel

$$\frac{C_1 \cup \{L\} \qquad C_2 \cup \{\neg L'\}}{C_1\sigma \cup C_2\sigma}$$

wobei

- die Elternklauseln keine Variablen gemeinsam haben (bereinigt)
  $\mapsto$ ggf. umbenennen
- $\sigma(L) = \sigma(L')$

Prädikatenlogische Resolution

Wiederholung

Grundidee

Vor Resolutionsschritt durch geeignete Substitution komplementäres Paar von Literalen erzeugen

Möglichkeit für Resolutionsregel

$$\frac{C_1 \cup \{L\} \qquad C_2 \cup \{\neg L'\}}{C_1\sigma \cup C_2\sigma}$$

wobei

- die Elternklauseln keine Variablen gemeinsam haben (bereinigt)
  $\mapsto$ ggf. umbenennen
- $\sigma(L) = \sigma(L')$

Nachteil: Viel zu viele Substitutionen σ mit $\sigma(L) = \sigma(L')$
Idee: Wähle die "allgemeinste" Substitution, mit $\sigma(L) = \sigma(L')$

Unifikationsalgorithmus von Martelli/Montanari

(1)	$t \stackrel{?}{=} t, E$	$\Rightarrow_{MM}$	E
(2)	$f(s_1, \ldots, s_n) \stackrel{?}{=} f(t_1, \ldots, t_n), E$	$\Rightarrow_{MM}$	$s_1 \stackrel{?}{=} t_1, \ldots, s_n \stackrel{?}{=} t_n, E$
(3)	$f(\ldots) \stackrel{?}{=} g(\ldots), E$	$\Rightarrow_{MM}$	$\perp$
(4)	$x \stackrel{?}{=} t, E$	$\Rightarrow_{MM}$	$x \stackrel{?}{=} t, E[t/x]$ falls $x \in var(E), x \notin var(t)$
(5)	$x \stackrel{?}{=} t, E$	$\Rightarrow_{MM}$	$\perp$ falls $x \neq t, x \in var(t)$
(6)	$t \stackrel{?}{=} x, E$	$\Rightarrow_{MM}$	$x \stackrel{?}{=} t, E$ falls $t \notin X$

Prädikatenlogische Resolution

Grundidee

Vor Resolutionsschritt durch geeignete Substitution komplementäres Paar von Literalen erzeugen

Möglichkeit für Resolutionsregel

$$\frac{C_1 \cup \{L\} \qquad C_2 \cup \{\neg L'\}}{C_1\sigma \cup C_2\sigma}$$

wobei

- die Elternklauseln keine Variablen gemeinsam haben (bereinigt)
  $\mapsto$ ggf. umbenennen
- $\sigma = \mathrm{mgu}(L, L')$

Prädikatenlogische Resolution

Resolutionsregel in dieser Form alleine unvollständig für Prädikatenlogik

Beispiel:

$\{\{p(x), p(y)\}, \{\neg p(u), \neg p(v)\}\}$

- unerfüllbar
- aber nur Resolventen der Länge 2

Prädikatenlogische Resolution

Faktorisierung

$$\frac{\{L_1, \ldots, L_n\} \cup C}{(\{L_1, \ldots, L_n\} \cup C)\sigma}$$

wobei

- σ allgemeinster Unifikator (MGU) von $\{L_1, \ldots, L_n\}$ ist

$(\{L_1, \ldots, L_n\} \cup C)\sigma$ heißt Faktor von $\{L_1, \ldots, L_n\} \cup C$

Beispiel für Faktorisierung

$$\frac{\{p(x),p(y),r(y,z)\}}{\{p(x),r(x,z)\}} \qquad \mathrm{mgu}(p(x), p(y)) = [x/y]$$

$$\frac{\{p(x),p(y),p(a),r(y,z)\}}{\{p(a),r(a,z)\}} \qquad \mathrm{mgu}(p(x), p(y), p(a)) = [a/x, a/y]$$

$$\frac{\{p(b),p(y),p(a),r(y,z)\}}{\{p(b),p(a),r(b,z)\}} \qquad \mathrm{mgu}(p(b), p(y)) = [b/y]$$

$$\frac{\{p(b),p(y),p(a),r(y,z)\}}{\{p(b),p(a),r(a,z)\}} \qquad \mathrm{mgu}(p(y), p(a)) = [a/y]$$

Resolutionskalkül RES für allgemeine Klauseln

$$\frac{C \cup \{A_1\} \qquad D \cup \{\neg A_2\}}{(C \cup D)\sigma} \quad \text{falls } \sigma = \text{mgu}(A_1, A_2) \qquad \text{[Resolution]}$$

$$\frac{C \cup \{L_1, L_2\}}{(C \cup \{L_1\})\sigma} \quad \text{falls } \sigma = \text{mgu}(L_1, L_2) \qquad \text{[Faktorisierung]}$$

Es wird immer implizit angenommen, dass die Variablen in einer der beiden Prämissen der Resolutionsregel ggfs. (bijektiv) umbenannt werden, so dass sie disjunkt mit denen der anderen Prämisse sind.

Dieses implizite Umbenennen werden wir nicht formalisieren.

Welche Variablennamen man verwendet ist egal.

Beispielsweise könnte man sich vorstellen, dass am Anfang alle Klauseln paarweise variablendisjunkt sind und das Unifikatoren so gewählt werden, dass in ihrem Wertebereich nur neue Variablen vorkommen.

Beispiel

1.	$\{P(x), P(f(x)), \neg Q(x)\}$	[Gegeben]
2.	$\{\neg P(y)\}$	[Gegeben]
3.	$\{P(g(x', x)), Q(x)\}$	[Gegeben]

Beispiel

1.	$\{P(x), P(f(x)), \neg Q(x)\}$	[Gegeben]
2.	$\{\neg P(y)\}$	[Gegeben]
3.	$\{P(g(x', x'')), Q(x'')\}$	[Gegeben; Bereinigt]
4.	$\{P(f(x)), \neg Q(x)\}$	[Res. 1, 2], $\mathrm{mgu}(P(x), P(y)) = [x/y]$
5.	$\{\neg Q(x)\}$	[Res. 4, 2], $\mathrm{mgu}(P(y), P(f(x))) = [f(x)/y]$
6.	$\{Q(x'')\}$	[Res. 3, 2], $\mathrm{mgu}(P(y), P(g(x', x''))) = [g(x', x'')/y$
7.	$\perp$	[Res. 5, 6], $\mathrm{mgu}(Q(x), Q(x'')) = [x/x'']$

Notation

Sei N eine Klauselmenge und

$$\mathrm{Res}(N) = N \cup \{R \mid R \text{ ist eine Resolvente zweier Klauseln aus } N \text{ oder Resultat der Faktorisierung einer Klausel aus } N\}$$

$$\mathrm{Res}^0(N) = N$$

$$\mathrm{Res}^{n+1}(N) = \mathrm{Res}(\mathrm{Res}^n(N))$$

$$\mathrm{Res}^\star(N) = \bigcup_{n \in \mathbb{N}} \mathrm{Res}^n(N)$$

(bezeichnet die Vereinigung der Ergebnisse aus aller möglichen Resolutions- und Faktorisierungsschritte auf N)

Anwendungen

- Logische Programmierung (Prolog, Answer Set Programming)
- Industrielle Anwendungen
- SAT Solver
- Theorem Prover
- Software Verifikation

Last but not least

Nicht vergessen Spass zu haben!

Logik für Informatiker

Vorlesung 14: Herbrand Strukturen

Babeş-Bolyai Universität, Department für Informatik, Cluj-Napoca
csacarea@cs.ubbcluj.ro

Herbrand-Struktur

Sei (Σ, Δ) eine Signatur. Eine Herbrand-Struktur (oder freie Struktur) hat die Gestalt $(T(\Sigma), \alpha)$, wobei für alle $f \in \Sigma_n$ mit $n \in \mathbb{N}$ gilt: $\alpha_f(t_1, \ldots, t_n) = f(t_1, \ldots, t_n)$. Falls eine Herbrand-Struktur Modell einer Formel ϕ ist, so bezeichnen wir sie auch als Herbrand-Modell von ϕ.

Erfüllbarkeitstest durch Herbrand-Strukturen

Der folgende Satz zeigt, dass man sich zur Untersuchung der Erfüllbarkeit in der Tat auf Herbrand-Strukturen beschränken kann.

Satz

Sei $\Phi \subseteq F(\Sigma, \Delta, V)$ eine Menge von Formeln in Skolem-Normalform. Dann ist ϕ erfüllbar gdw. sie ein Herbrand-Modell besitzt.

Nachdem wir nun wissen, dass man zur Überprüfung der Unerfüllbarkeit einer Formel ϕ in Skolem-Normalform nur Herbrand-Strukturen betrachten muss, reduzieren wir dies nun auf die Überprüfung der Unerfüllbarkeit einer (unendlichen) Menge von Formeln ohne Variablen.
Dies entspricht der Überprüfung der Unerfüllbarkeit für eine (unendliche) Menge aussagenlogischer Formeln.
Die Idee hierzu besteht darin, alle in ϕ allquantifizierten Variablen mit allen möglichen Grundtermen zu instantiieren.
Auf diese Weise entsteht die Herbrand-Expansion von ϕ.

Herbrand Expansion

Sei $\phi \in F(\Sigma, \Delta, \mathcal{V})$ eine Formel in Skolem-Normalform, wobei $\phi = \forall X_1, \ldots, X_n \psi$ für eine quantorfreie Formel ψ. Die folgende Formelmenge $E(\phi)$ wird als Herbrand-Expansion von ϕ bezeichnet:

$$E(\phi) = \{\psi[X_1/t_1, \ldots, X_n/t_n] \mid t_1, \ldots, t_n \in T(\Sigma)\}.$$

Die Herbrand-Expansion einer Formel $\forall X_1, \ldots, X_n \psi$ ist also die Menge aller Grundinstanzen von ψ.

Beispiel

Wir betrachten die Formel ϕ:

$$\forall \texttt{X} \texttt{mutterVon}(\texttt{renate}, \texttt{susanne}) \wedge \neg \texttt{mutterVon}(\texttt{X}, \texttt{susanne})$$

Die Herbrand-Expansion von ϕ ist:
$E(\phi) = \{mutterVon(renate, susanne) \wedge \neg mutterVon(karin, susanne),$
$mutterVon(renate, susanne) \wedge \neg mutterVon(renate, susanne),$
$mutterVon(renate, susanne) \wedge$
$\neg mutterVon(datum(karin, werner, 1982), susanne), \ \ldots\}.$

Erfüllbarkeit der Herbrand-Expansion

Sei ϕ eine Formel in Skolem-Normalform. Dann ist ϕ erfüllbar gdw. $E(\phi)$ erfüllbar ist.

- Pradikatenlogische Formeln ohne Variablen entsprechen aussagenlogischen Formeln. Der Grund ist, dass man jede atomare Teilformel $p(t_1, \ldots, t_n)$ als aussagenlogische Variable ansehen kann, die entweder wahr oder falsch sein kann. Es genügt zu untersuchen, ob die Menge $E(\phi)$ ein Herbrand-Modell hat.
- Herbrand-Strukturen unterscheiden sich nur in der Deutung der Prädikatssymbole und bei variablenfreien Formeln bedeutet die Suche nach einem Herbrand-Modell nichts anderes als die Suche nach einer aussagenlogischen Belegung der aussagenlogischen Variablen $p(t_1, \ldots, t_n)$ mit wahr oder falsch.
- Die Frage nach der Unerfüllbarkeit einer Formel ist jetzt also reduziert worden auf die Suche nach einer erfüllenden Belegung für eine (im Allgemeinen unendliche) Menge aussagenlogischer Formeln.

Beispiel

- Wir betrachten noch einmal die Herbrand-Expansion der Formel ϕ aus dem obigen Beispiel.
- Ersetzt man hier jede atomare Formel $p(t_1, \ldots, t_n)$ ohne Variablen durch eine aussagenlogische Variable $V_{p(t_1,\ldots,t_n)}$, so ergibt sich aus $E(\phi)$ die folgende unendliche Menge aussagenlogischer Formeln:
  $\{V_{mutterVon(renate,susanne)} \wedge \neg V_{mutterVon(karin,susanne)},$
  $V_{mutterVon(renate,susanne)} \wedge \neg V_{mutterVon(renate,susanne)},$
  $V_{mutterVon(renate,susanne)} \wedge \neg V_{mutterVon(datum(karin,werner,1982),susanne)},$
  $\ldots\}.$
- Die Formel ϕ und damit $E(\phi)$ ist nun genau dann erfüllbar, wenn es eine Belegung der aussagenlogischen Variablen $V_{p(t_1,\ldots,t_n)}$ mit wahr oder falsch gibt, die die obige aussagenlogische Formelmenge erfüllt.

Beispiel

- Natürlich gibt es solch eine Belegung nicht, denn die Formel
  $V_{mutterVon(renate,susanne)} \wedge \neg V_{mutterVon(renate,susanne)}$
  ist weder für die Belegung von $V_{mutterVon(renate,susanne)}$ mit wahr noch mit falsch erfüllt.

Algorithmus von Gilmore

- Eingabe: Formeln $\phi_1, \ldots, \phi_k, \phi \in F(\Sigma, \Delta, \mathcal{V})$
- Ziel: herauszufinden, ob $\{\phi_1, \ldots, \phi_k\} \vDash \phi$ gilt.

Algorithmus von Gilmore

- Sei ψ die Formel $\phi_1 \wedge \cdots \wedge \phi_k \wedge \neg\phi$. Es gilt $\{\phi_1, \ldots, \phi_k\} \vDash \phi$ gdw. ψ unerfüllbar ist.
- Überführe ψ in Skolem-Normalform. Die entstehende Formel ist erfüllbarkeitsäquivalent zur ursprünglichen Formel.
- Wähle eine Aufzählung $\{\psi_1, \psi_2, \ldots\} = E(\forall\psi)$ der Herbrand-Expansion von $\forall\psi$ und ersetze dabei alle atomaren Formeln durch aussagenlogische Variablen.
- $E(\forall\psi)$ ist genau dann (aussagenlogisch) erfüllbar, wenn $\forall\psi$ erfüllbar ist. Nach dem Kompaktheitssatz der Aussagenlogik ist dies genau dann der Fall, wenn alle endlichen Teilmengen von $E(\forall\psi)$ aussagenlogisch erfüllbar sind.

Algorithmus von Gilmore

- Prüfe, ob $\psi_1, \psi_1 \wedge \psi_2, \psi_1 \wedge \psi_2 \wedge \psi_3, \ldots$ aussagenlogisch erfüllbar sind (indem man alle möglichen Wahrheitsbelegungen der vorkommenden aussagenlogischen Variablen durchtestet). Sofern eine dieser Formeln nicht erfüllbar ist, brich ab und gib true zurück.

Bemerkung

- Dieser Algorithmus terminiert und liefert genau dann das Ergebnis true zurück, wenn $\{\phi_1, \ldots, \phi_k\} \vDash \phi$ gilt.
- Sofern dies nicht gilt, terminiert der Algorithmus nicht.
- Der Nachteil des Algorithmus ist allerdings, dass er noch nicht sehr zielgerichtet vorgeht und daher noch sehr ineffizient ist.