Mengenlehre

∪ vereinen ∩ Schnitt

- Sammlung von Objekten, betrachtet als Einheit

Teilmenge: $M_1 \subseteq M_2$; $x \in M_1 \Rightarrow x \in M_2$

Echte Teilmenge: $M_1 \subset M_2$, $M_1 \subseteq M_2$ $M_1 \neq M_2$

$\mathbb{N} = \{0,1,2,3\ldots\}$ Natürliche Zahlen

$\mathbb{Z}$ = Ganze Zahlen (auch negative)

$\mathbb{Q} = \{\frac{p}{q} \mid p \in \mathbb{Z}, q \in \mathbb{N}^{\geq 1}\}$ Rationale Zahlen

$\mathbb{R}$ = Reele Zahlen (alle Zahlen)

①

Primzahlen: 2,3,5,7,11,13,17,19,23,29,31,37,41,43,47,53 ...

Rechenregeln: Mengen

- Kommutativgesetz:
  - $M \cup P = P \cup M$
  - $M \cap P = P \cap M$
- Assoziativgesetz:
  - $M \cup (P \cup S) = (M \cup P) \cup S$
  - $M \cap (P \cap S) = (M \cap P) \cap S$
- Distributivgesetz:
  - $M \cup (P \cap S) = (M \cup P) \cap (M \cup S)$
  - $M \cap (P \cup S) = (M \cap P) \cup (M \cap S)$
- De Morgan:
  - $\overline{M \cup P} = \overline{M} \cap \overline{P}$
  - $\overline{M \cap P} = \overline{M} \cup \overline{P}$
- Neutrale Elemente:
  - $M \cup \emptyset = M$
  - $M \cap T = M$
- Inverse Elemente:
  - $M \cup \overline{M} = T$
  - $M \cap \overline{M} = \emptyset$
- Absorbierende Elemente:
  - $M \cup T = T$
  - $M \cap \emptyset = \emptyset$
- Idempotenz
  - $M \cup M = M$
  - $M \cap M = M$
- doppeltes Komplement:
  - $\overline{\overline{M}} = M$

$M = \{x,y\}$

$2^M = \{\emptyset, \{x\}, \{y\}, \{x,y\}\}$

W und F

$F^I = 0$ $W^I = 1$

a	b	a ∧ b
0	0	0
0	1	0
1	0	0
1	1	1

Konjunktion ∧

Aussagenlogik, Interpretation, Tabellen

a	b	a → b
0	0	1
0	1	1
1	0	0
1	1	1

Implikation

a	b	a ∨ b
0	0	0
0	1	1
1	0	1
1	1	1

Disjunktion

a	b	a → b
0	0	1
0	1	0
1	0	0
1	1	1

Äquivalenz

Literal: Aussagenvariable

z.B. $\{A,B\} \rightarrow \{A, \neg A, B, \neg B\}$

Aussagenlogik

Aussagenvariablen: Atomare Aussagen; A = Es regnet

↳ wahr oder falsch

Syntax: Lehre von wohlgeformten Ausdrücken, Regeln zum Erzeugen von Formeln aus vorgegebenen Symbolen

Semantik: Lehre von Bedeutung von Ausdrücken; wahr oder falsch; 0,1

Vorrang der Junktoren: $\neg$; $\wedge$; $\vee$; $\rightarrow$; $\leftrightarrow$

$A \mapsto 1$ „es regnet"

$I(A) = 1$

$A^I = 1$

Interpretationen

Tautologie: Formel, die in jeder Interpretation wahr ist.

$a \leftrightarrow a$ (Identität)

$\neg(a \wedge \neg a)$ (Ausgeschlossener Widerspruch)

$a \vee \neg a$ (ausgeschlossenes Drittes

$a \rightarrow b \Leftrightarrow \neg b \rightarrow \neg a$ (Kontraposition)

$(a \rightarrow b) \wedge (a \rightarrow \neg b) \rightarrow \neg a$ (Widerspruchsbeweis)

$(a \rightarrow b) \wedge a \rightarrow b$ (Modus Ponens)

$(a \rightarrow b) \wedge (\neg a \rightarrow b) \rightarrow b$ (Fallunterscheidung)

Logische Implikation: Jedes Modell von a ist auch eins für b

z.B. $F \not\vdash a$; $a \vdash W$

Logische Äquivalenz: Jede Interpretation ordnet beiden Wahrheitswert zu

z.B. $a \leftrightarrow b \equiv (a \rightarrow b) \wedge (b \rightarrow a) \equiv a \wedge b \vee \neg a \wedge \neg b$

$a \rightarrow b \equiv \neg a \vee b$

$a \wedge b \equiv \neg(\neg a \vee \neg b)$

$a \vee b \equiv \neg(\neg a \wedge \neg b)$

Rechenregeln Aussagenlogik

Kommutativ: $a \wedge b \equiv b \wedge a$

Assoziativ: $(a \wedge b) \wedge c \equiv a \wedge (b \wedge c)$

Distributiv: $(a \wedge b) \vee c \equiv (a \vee c) \wedge (b \vee c)$

De Morgan: $\neg(a \wedge b) \equiv \neg a \vee \neg b$

Neutral: $a \wedge W \equiv a$ $a \vee F \equiv a$

Invers: $a \wedge \neg a \equiv F$ $a \vee \neg a \equiv W$

Absorption: $a \wedge F \equiv F$ $a \vee W \equiv W$

Idempotenz: $a \wedge a \equiv a$ $a \vee a \equiv a$

Doppelte Negation: $\neg\neg a \equiv a$

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

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

∨ zwischen Elementen

∧ zwischen Mengen

Klausel

$a = (X \vee Y) \wedge (\neg Y \vee Z) \wedge \neg Z \wedge \neg X$

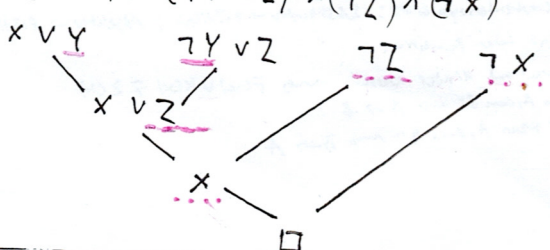
$K(a) = \{\{X,Y\}, \{\neg Y, Z\}, \{\neg Z\}, \{\neg X\}\}$

AG Menge geschriebene Disjunktion von Literalen

Resolutionsprinzip

- Konjunktive Normalform KNF; Konjunktion von Disjunktionen (Klauseln) von Literalen
- Paare mit gegensätzlichen Literalen finden
- Resolvente erzeugen, nicht äquivalent! ; jedoch selbe Erfüllbarkeit
- leere Klausel: $\square$ (Widerspruch)

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



$S \vdash^* C \Rightarrow$ Herleitung durch endlich viele Schritte

Transformation in die KNF

1. a) eliminiere materiale Äquivalenz
   $a \leftrightarrow b \rightsquigarrow (a \rightarrow b) \wedge (b \rightarrow a)$
   b) eliminiere materiale Implikation
   $a \rightarrow b \rightsquigarrow \neg a \vee b$
2. De Morgan
   $\neg(a \wedge b) \rightsquigarrow \neg a \vee \neg b$
3. a) doppelte Negation
   $\neg\neg a \rightsquigarrow a$
   b) negierte Konstanten
   $\neg W \rightsquigarrow F$
4. Distributivgesetz
   $a \vee (b \wedge c) \rightsquigarrow (a \vee b) \wedge (a \vee c)$
5. Assoziativgesetz
   $a \wedge (b \wedge c) \rightsquigarrow a \wedge b \wedge c$
   $a \vee (b \vee c) \rightsquigarrow a \vee b \vee c$

Aussagenlogik: Resolutionsverfahren

Testen auf logische Implikation und Erfüllbarkeit (anhand Bsp.)

- Implizieren Prämissen 1–5 Konklusion 6?
  - Ist jedes Modell von 1–5 auch eins für 6?
    $1 \wedge 2 \wedge 3 \wedge 4 \wedge 5 \not\models 6$
- Umgekehrt: Existiert ein Modell von 1–5, in dem 6 nicht gilt?
  - Ist $1 \wedge 2 \wedge 3 \wedge 4 \wedge 5 \wedge \neg 6$ erfüllbar?
    - ↳ ja, dann gilt Implikation nicht
    - ↳ nicht erfüllbar → Implikation gilt
  - mittels Resolution testen

Optimierung Resolutionsverfahren

- Ignoriere Tautologien $\{X, \neg X, \dots\}$
- Ignoriere C_1, wenn ein $C_2 \subset C_1$ existiert
- kleine Klauseln priorisieren
- Klauseln mit hoher Ableitungstiefe priorisieren
- Horn-Klausel: Höchstens ein positives Literal
- Horn-Regel: $X \wedge Y \wedge Z \dots \rightarrow W$

Resolution: sucht Widerspruch; effizient für unerfüllbar ②
Tableau: sucht Modell; Effizient für erfüllbar

Teilformeln:

a ist Teilformel von b wenn eine von den Bed. erfüllt

- $a \equiv b$
- $a \equiv \neg c$; b ist Teilformel von c
- $a \equiv c_1 \circ c_2$; b ist Teilformel von c_1 oder c_2 $(\circ \in \{\wedge, \vee, \rightarrow, \leftrightarrow\})$; Wildcard

Beispiel: $(a = A \wedge \neg(B \vee \neg A) \vee (A \rightarrow \neg C))$

Teilformeln: $\{a, A \wedge \neg(B \vee \neg A), A \rightarrow \neg C, A, \neg(B \vee \neg A), B \vee \neg A, B, \neg A, \neg C, C\}$

Terminierung Tableau Algorithmus

- endlich viele Teilformeln
- Regelanwendung fügt nur Teilformeln
- jede Teilformel höchstens eine Regel
- Anzahl Spalten beschränkt (Disjunktionen)
- Formeln werden nie entfernt.

Tableau-Algorithmus

- $a \wedge b \rightarrow$ und-Regel $\rightarrow$ a, b
- $a \vee b \rightarrow$ oder-Regel $\rightarrow$ a | b
- Clashes gibt es zwischen Formeln in der aktuellen Spalte und der darüberliegenden
1. Jede Spalte ein Clash → es existiert kein Modell
2. Offene keine Regel mehr anwendbar, mindestens eine Spalte clash-frei → es gibt ein Modell

Negations-Normalform NNF

1. Elimination $\leftrightarrow$ und $\rightarrow$
2. De Morgan
3. Doppelte Negation entfernen

- $\wedge$-Regel zuerst anwenden
- $\vee$-Regel so anwenden, dass möglichst wenige offene Spalten verbleiben
- NNF nicht in KNF transformieren
- immer eine Spalte komplett abarbeiten, dann zur nächsten

Prädikatenlogik

Kartesisches Produkt

$$M_1 \times M_2 = \{(x, y) \mid x \in M_1, y \in M_2\}$$

Beispiel: $M_1 = \{1, 2, 3\}$ $M_2 = \{a, b\}$

$M_1 \times M_2 = \{(1, a), (2, a), (3, a), (1, b), (2, b), (3, b)\}$

Relation

Eine Relation ist eine Teilmenge des kartesischen Produkts,
z.B. $R(x, y)$

- homogen, falls alle Elemente selber Typ
- heterogen, falls unterschiedliche Typen
- linkstotal: für jeden Wert der Domäne gibt es einen Wert in der Zielmenge.
- rechtseindeutig: Höchstens ein Element in der Zielmenge

Funktionen $M \rightarrow N$ $f \subseteq (M \times N)$

- partielle Funktion: rechtseindeutig
- (totale) Funktion: linkstotal und rechtseindeutig

M: Definitionsmenge
N: Zielmenge

- Funktion ordnet (jedem) Element der Definitionsmenge höchstens eins der Zielmenge zu.

Domäne ≙ Definitionsmenge

- nullstellige Funktionen beschreiben Konstanten

Prädikat → liefert true/false
Funktion → liefert Element aus de Domäne

$\Delta^{\mathcal{I}} = \{ \dots$
$f(x) = \dots$
$P(x) = \dots$
$c = \dots$

Beispiel Signatur:
$(\{+^{(2)}, \cdot^{(2)}, 0^{(0)}, 1^{(0)}\}, \{=\})$ (Arithmetik)

Beispiel Struktur: $(\mathbb{N}, +_{\mathbb{N}}, \cdot_{\mathbb{N}}, 0_{\mathbb{N}}, 1_{\mathbb{N}})$

Syntax

Variablen: x, y; Konstantensymbole: a, b, c; Funktionssymbole: f(x); h(x,y)
Prädikate: R(x), S(x,y) Quantoren: $\forall$, $\exists$

Terme ohne Variablen → Grundterme
Jede Konstante ist ein Term
Jedes nullstellige Prädikat ist ein Atom. (+ Aussagenvariable)
Atom: P(x,y) kein Atom: P(x,y) → P(x)
Quantoren binden Variablen / Sonst frei
Vorkommen von

Terme: x; f(x,y)
Grundterme: c; f(c,a)

f; P; $\forall, \exists$; $\neg$; $\wedge$; $\vee$; $\rightarrow$; $\leftrightarrow$ Vorrang

Übersetzung

Eigenname → Konstantensymbol; Substantiv → Prädikat; Adjektiv → Prädikat
Verb → Prädikat oder Funktion
und, oder „Frauen und Kinder zuerst" ⇝ $F(x) \vee K(x) \rightarrow Z(x)$
wenn, dann „Wenn A, dann B" ⇝ $A \rightarrow B$
$\forall$ mit $\rightarrow$ „Nur wenn A, dann B" ⇝ $B \rightarrow A$
$\exists$ mit $\wedge$

Prädikatenlogik Fortsetzung

Resolution

1. NNF

1. Elimination $\leftrightarrow$ und $\rightarrow$
   - $a \leftrightarrow b \equiv (a \rightarrow b) \wedge (b \rightarrow a)$
   - $a \rightarrow b \equiv \neg a \vee b$
2. De Morgan und Quantoren
   $\neg \forall x\, a \equiv \exists x \neg a$
   $\neg \exists x\, a \equiv \forall x \neg a$
3. Doppelte Negation entfernen

2. Skolemisierung

- Def.: NNF und keine $\exists$
- Variablen die im Scope von $\exists$, aber nicht $\forall$ liegen
  ↳ Konstanten
- Variablen im Scope von $\exists$ und $\forall$
  ↳ Funktion

→ zuerst: Miniscoping

- Quantoren von innen nach außen bearbeiten
- Bsp. $\forall x \exists y (P(x) \wedge Q(y))$
  ⇝ $\forall x (P(x) \wedge \exists y Q(y))$
  ⇝ $\forall x P(x) \wedge \exists y Q(y)$

3. Allquantoren entfernen

- mehrfach quantifizierte Variablen umbenennen
- Bsp: $\forall x (P(x) \vee Q(x)) \vee \forall x\, P(x,x)$
  ⇝ $\forall x (P(x) \vee Q(x)) \vee \forall y\, P(y,y)$
  ⇝ $P(x) \vee Q(x) \vee P(y,y)$

4. KNF und Resolution

Bsp.: $H(c) \wedge (\neg H(x) \vee M(x, f(x)))$

$\{H(c)\}$ $\{\neg H(x), M(x, f(x))\}$

$/[x/c]$

$M(c, f(c))$

⋮ usw.

Unifikation → Abbildung Variablen auf Terme

Variablen-Umbenennung (gleiche Variable in mehreren Klauseln)
Unifikator für P(x,c) und P(d,x) wird nicht gefunden.
→ Disjunkt umbenennen:
P(x, c) P(d, y)

Occurs-Check

- X kann nicht auf Term abgebildet werden, der x enthält
- Bsp: $[X/f(x)]$

Tableau

- Gleiches Prinzip → auch NNF
- neue Regeln: $\exists$-Regel und $\forall$-Regel
  - $\exists$-Regel:
    - $\exists$ eliminieren durch Skolemisierung
    - $\forall$ vorher durch $\forall$-Regel bearbeiten
    - Ein mal anwendbar
  - $\forall$-Regel:
    - $\forall$ entfernen und Konstante einsetzen
    - unendlich oft anwendbar
  - S-Regel:
    - es kann auch eine Unifikation angewendet werden, die sich auf jede Formel des Tableaus auswirkt
      → Clashe erzeugen

Prolog

Der Cut

- Symbol: !
- Beim Erreichen des Cut:
  - Verzweigungspunkt gelöscht
  - Variablenbindung fixiert

Fail

- löst Backtracking aus
- zusammen mit Cut → kann Ausnahmen beschreiben

likes(vincent, hahuna) :- !, fail.

Negation

not(Goal) :- Goal, !, fail.
not(Goal).

Listen-Operationen

- member(T, L)
  → true wenn Term T in Liste L enthalten ist

member(X, [X|_]).
member(X, [_|T]) :- member(X,T).

- Length([], 0)
  → gibt Anzahl der Elemente
- append([], L, L)
- Aneinanderfügen von Listen

append([a,b], [1,2], A)
→ A = [a,b,1,2]

- prefix, suffix, sublist

Listen

[Head | Tail]

Head: • erstes Element der Liste
• beliebiger Term

Tail: • Rest der Liste
• ist immer eine Liste

[]: Leere Liste

Bsp.: [vincent | [mia | [jules | []]]]
oder [vincent, mia, jules]

Rekursion

Def. p ist rekursiv, wenn es im Kopf als auch im Rumpf vorkommt

Bsp numeral(0).
numeral(s(X)) :- numeral(X).

→ im Suchbaum: Variablen umbenennen

Prolog

Syntax

- **Atome**
  - Strings aus Buchstaben, Zahlen und Unterstrich beginnend mit Kleinbuchstaben
    z.B. butch, honey_bunny
  - beliebige strings in einfachen Anführungszeichen
    z.B. 'Vincent'
- **Variablen**
  - beginnen mit Großbuchstabe oder Unterstrich
    z.B. X, Y, Variable, _var
- **Funktor**

z.B.: loves(marsellus, mia)

- Verschachtelung erlaubt
- Faustregel: äußerster Funktor → Prädikat
  innere Funktoren → Funktionen

z.B. • knows(hoons, father(butch)).
Prädikat (zweistellig) — Funktion (einstellig)

- father(butch).
  einstelliges Prädikat

- **Goal**
  - Jedes Atom und jeder komplexe Term ist ein Goal
- **anonyme Variable** _
  - mit jedem Term unifizierbar
  - kann bei jedem Vorkommen unterschiedlich gebunden werden
  - Platzhalter für Terme, die uninteressant sind

Regeln

1. <Kopf> :- <Rumpf>.
   Kopf wenn Rumpf
   Der Rumpf impliziert den Kopf
2. <Kopf> :- <Rumpf>.
   Rumpf: <Goal1>, <Goal2>, ...
   Konjunktion (nk) der Goals impliziert Kopf
3. <Kopf> :- <Rumpf>
   Rumpf: <Goal1>; <Goal2>; ...
   Disjunktion (v) der Goals im Rumpf impliziert Kopf

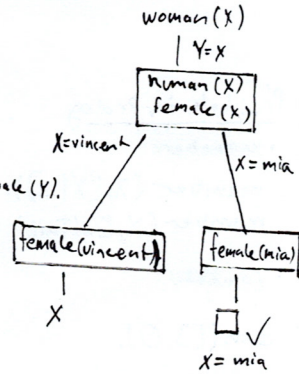
Prädikate

- =/2 → erfolgreich, wenn beide Argumente unifizierbar
  z.B. woman(X) = woman(mia).
- \= → Negation von =
  → gibt nie Variablenbindungen zurück
- == → testen ob Argumente bereits gleich sind
  → berücksichtigt bestehende Variablenbindungen
  → Negation: \==
  → ein bisschen so wie String-comparison
- is → arithmetische Operationen
  → rechte Seite wird ausgewertet
  → Ergebnis mit linker Seite unifiziert
  z.B. X is 2+3 → X=5

Suchbaum

Bsp.

human(vincent).
male(vincent).
human(mia).
female(mia).
woman(Y) :- human(Y), female(Y).



- X < Y
- X =< Y
- X =:= Y
- X =\= Y
- X >= Y
- X > Y

→ beide Seiten müssen korrekte arithmetische Terme sein
→ linke und rechte Seite wird ausgewertet
→ true/false

Prolog Fortsetzung

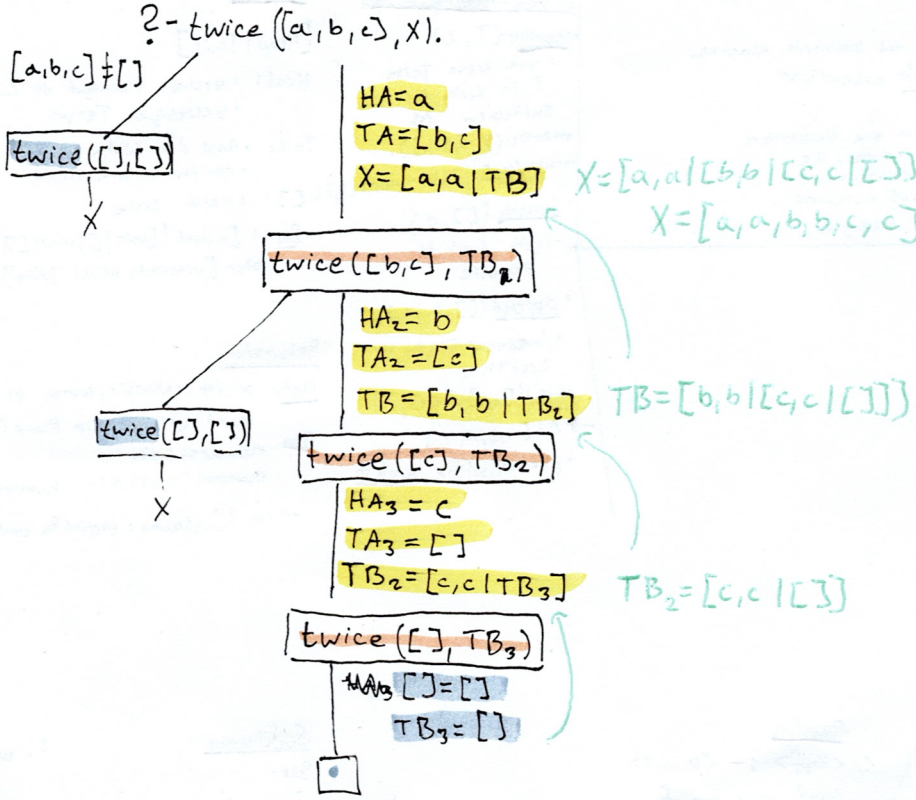
Atom: beginnt mit Kleinbuchstabe
Variable: beginnt mit Großbuchstabe oder Unterstrich
} einfache Terme

Komplexer Term: besteht aus:
Funktor: Atom
Argumente: beliebige Terme

Prolog Suchbaum + Rekursion

mit welchem „Input" wird twice als Nächstes aufgerufen

```
twice([],[]).
twice([HA|TA],[HA,HA|TB]) :- twice(TA,TB).
```



Prädikate in Prolog

- **member:**

```
member(X,[X|_]).
member(X,[_|Tail]) :- member(X,Tail).
```

- **length**

```
len([],0).
len([_|Tail],N) :- len(Tail,N1), N is N1+1.
```

- **append**

```
append(A,T,T) :- member(A,T), !.
append(A,T,[A|T]).
```

- **concatenate**

```
concat([],L,L).
concat([X1|L1],L2,[X1|L3]) :- concat(L1,L2,L3).
```

- **reverse list**

```
rev([],[]).
rev([H|T],Rev) :-
    rev(T,RevT), concat(RevT,[H],Rev).
```

- **order**

```
order([X,Y|Tail]) :- X=<Y, order([Y|Tail]).
order([X]).
```