

Logik Teil 2: Prädikatenlogik Grundlagen

Prädikatenlogik

Für viele Zwecke in der Informatik und Mathematik **abstrahiert** die Aussagenlogik zu stark.

Betrachte z. B. die Beispiele aus der Einleitung:

Alle Menschen sind sterblich
Sokrates ist ein Mensch

Sokrates ist sterblich

Jedes P ist auch ein Q
 x ist ein P

 x ist ein Q

- $\forall n \in \mathbb{N} : \exists n' \in \mathbb{N} : n' = nf(n)$
- $\forall n \in \mathbb{N} : nf(n) \neq 0$
- ...

Bei diesen Aussagen geht es nicht nur um Wahrheitswerte:
Objekte (Menschen, natürliche Zahlen) und Quantifizierung sind zentral!

Prädikatenlogik

Die Prädikatenlogik wurde von Frege gegen Ende des 19. Jh. eingeführt.

Zentrale Elemente:

1. Formeln zusammengesetzt aus Objektvariablen, Booleschen Operatoren und Quantoren
2. eine Semantik, die Objekte sowie deren Eigenschaften und Beziehungen erfasst

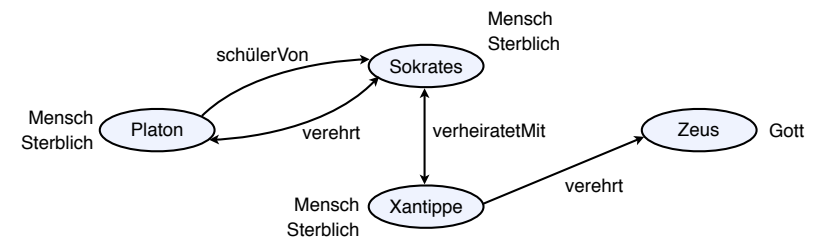
Prädikatenlogik spielt zentrale Rolle in Informatik, Mathematik, Philosophie

Andere Namen: Logik erster Stufe, First-order Logic, Predicate calculus

Abkürzung: FO

Vorschau 1

Eine semantische Struktur der Logik erster Stufe:



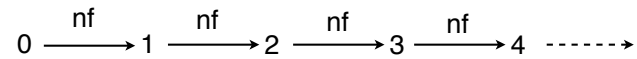
Zu dieser Struktur passende Beispielformeln:

$$\forall x (\text{Mensch}(x) \rightarrow \text{Sterblich}(x))$$

$$\exists x (\exists y (\text{vereehrt}(x, y) \wedge \text{Gott}(y)) \wedge$$

$$\exists y (\text{verheiratetMit}(x, y) \wedge \forall z (\text{vereehrt}(y, z) \rightarrow \neg \text{Gott}(z))))$$

Eine semantische Struktur der Logik erster Stufe:



Zu dieser Struktur passende Beispielformeln:

$$\forall x \exists y (y = \text{nf}(x))$$

$$\exists x \forall y \neg (x = \text{nf}(y))$$

$$y = \text{nf}(\text{nf}(x))$$

NEXT →

2.1 Strukturen

2.2 Syntax und Semantik der Prädikatenlogik

2.3 Auswertung und Datenbanken

2.4 Äquivalenz, Erfüllbarkeit, Gültigkeit

2.5 Pränex-Normalform

2.6 Unentscheidbarkeit

2.7 Theorien

Strukturen

Die Semantik der Prädikatenlogik basiert auf so genannten **Strukturen**.

Man kann sehr viele Dinge als Struktur repräsentieren:

- Graphen und Hypergraphen
- Wörter (im Sinne der formalen Sprachen)
- Relationale Datenbanken
- Transitionssysteme aus der Hard/Software-Verifikation
- Mathematische Strukturen wie Gruppen, Ringe, Körper
- etc

Dies macht die Prädikatenlogik zu einem **sehr generellen Werkzeug**.

Strukturen

Die Namen, die in einer Struktur verwendet werden, bilden deren **Signatur**.

Definition Signatur

Eine *Signatur* τ ist eine Menge von *Relations-* und *Funktionssymbolen*. Jedes dieser Symbole hat eine feste endliche *Stelligkeit*. Nullstellige Funktionssymbole nennen wir *Konstantensymbole*.

Beispiel:

Die Signatur der Arithmetik ist $\{+, \cdot, 0, 1\}$, wobei

+ und · zweistellige Funktionssymbole

0 und 1 Konstantensymbole

Mehr Beispiele:

- Die Signatur eines gerichteten Graphen ist $\{E\}$, mit E zwei-stelligem Relationssymbol (das die Kanten repräsentiert)
- Die Signatur einer Datenbank besteht aus je einem n -stelligem Relationssymbol für jede n -spaltige Tabelle

Eine Signatur heißt

- *relational*, wenn sie keine Funktionssymbole enthält
- *funktional*, wenn sie keine Relationssymbole enthält

Die folgenden Definitionen (Struktur, Formel) beziehen sich alle auf eine Signatur τ

Notation: Normalerweise verwenden wir

- P, Q, R für Relationssymbole
Relationssymbole nennen wir auch *Prädikate*
- f, g, h für Funktionssymbole
- c, d, e für Konstantensymbole
- σ, τ für Signaturen

Statt Stelligkeit sagen wir auch *Arität*

Definition Struktur

Eine *Struktur* \mathfrak{A} besteht aus

- einer nichtleeren Menge A , dem *Universum* von \mathfrak{A}
- für jedes n -stellige Relationssymbol P eine Relation $P^{\mathfrak{A}} \subseteq A^n$
- für jedes n -stellige Funktionssymbol f eine Funktion $f^{\mathfrak{A}} : A^n \rightarrow A$

Eine Struktur, die genau die Symbole in τ interpretiert, heißt τ -*Struktur*.

Folgendes ist implizit:

- jedes unäre Relationssymbol wird als Teilmenge von A interpretiert
- jedes Funktionssymbol wird als *totale* Funktion interpretiert
- jedes Konstantensymbol wird als Element von A interpretiert

Notation:

- Strukturen bezeichnen wir mit Buchstaben in Frakturschrift $\mathfrak{A}, \mathfrak{B}, \mathfrak{C}$,
- der entsprechende lateinische Buchstabe A, B, C steht für das Universum der Struktur
- die Elemente des Universums bezeichnen wir mit a, b
- $\mathfrak{A} = (A, P_1^{\mathfrak{A}}, P_2^{\mathfrak{A}}, \dots, f_1^{\mathfrak{A}}, f_2^{\mathfrak{A}}, \dots)$ bezeichnet also eine Struktur über der Signatur $\{P_1, P_2, \dots, f_1, f_2, \dots\}$ mit Universum A

Strukturen, Graphen, Algebren

Strukturen generalisieren Graphen und Hypergraphen:

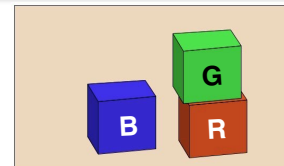
- Struktur $(U, R^{\mathfrak{A}})$ mit R binärem Relationssymbol ist nichts weiter als ein gerichteter Graph (und umgekehrt)
- Strukturen mit mehreren binären Relationssymbolen entsprechen dann kantenbeschrifteten (gerichteten) Graphen
- Unäre Relationssymbole liefern Knotenbeschriftungen im Graph
- n -stellige Relationssymbole mit $n > 2$ entsprechen (gerichteten) Hypergraphen

T2.1

Strukturen generalisieren zudem Algebren:

Eine Struktur für eine funktionale Signatur ist nichts weiter als eine Algebra (im Sinne der universellen Algebra)

Strukturen – Beispiel 1



repräsentiert als Struktur:

Signatur:

- unäre Relationssymbole Block, R, G, B
- binäre Relationssymbole auf, unter, neben
- Konstantensymbol Lieblingsblock

Struktur \mathfrak{A} :

- $A = \{rb, gb, bb\}$
- $\text{Block}^{\mathfrak{A}} = \{rb, gb, bb\}$, $\text{R}^{\mathfrak{A}} = \{rb\}$, $\text{G}^{\mathfrak{A}} = \{gb\}$, $\text{B}^{\mathfrak{A}} = \{bb\}$
- $\text{auf}^{\mathfrak{A}} = \{(gb, rb)\}$, $\text{unter}^{\mathfrak{A}} = \{(rb, gb)\}$, $\text{neben}^{\mathfrak{A}} = \{(bb, rb), (rb, bb)\}$
- $\text{Lieblingsblock}^{\mathfrak{A}} = rb$

T2.2

Strukturen – Beispiel 2

Relationale Datenbank ist eine endliche Sammlung von Tabellen

Jeder Tabelle T zugeordnet ist Spaltenzahl n und Attribute D_1, \dots, D_n
(Attribute z. B. Integers, Strings etc.)

Konkrete Datenbankinstanz I ordnet dann jedem T endliche Tupelmengen $T^I \subseteq D_1 \times \dots \times D_n$ zu

I kann als (endliche) Struktur

$$\mathfrak{A} = (A, T_1^{\mathfrak{A}}, T_2^{\mathfrak{A}}, \dots, T_k^{\mathfrak{A}})$$

repräsentiert werden, wobei

- T_1, \dots, T_k Relationssymbole für die Tabellen der Datenbank sind
- A die Vereinigung über alle Attribute ist, eingeschränkt auf die (endlich vielen) in I verwendeten Objekte

Strukturen – Beispiel 2

Betrachte eine Datenbank mit 2 Tabellen:

- Tabelle *Film*, 3 Spalten:
 - Titel (Typ String)
 - Jahr (Typ pos. Integer)
 - Regisseur (Typ String)
- Tabelle *Schauspieler_in*, 2 Spalten:
 - Name (Typ String)
 - Filmtitel (Typ String)

Beispielinstanz I :

Film:

Titel	Jahr	Regisseur
Die Vögel	1963	Hitchcock
Marnie	1964	Hitchcock
Goldfinger	1964	Hamilton

Schauspieler_in:

Name	Filmtitel
Connery	Marnie
Connery	Goldfinger
Hedren	Die Vögel

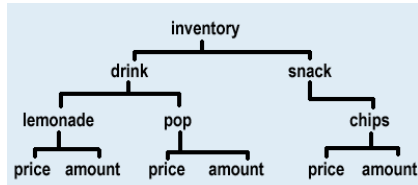
Als Struktur:

T2.3

Strukturen – Beispiel 3

XML-Dokument kann als endliche, baumförmige Struktur gesehen werden

```
<inventory>
  <drink>
    <lemonade>
      <price>$2.50</price>
      <amount>20</amount>
    </lemonade>
    <pop>
      <price>$1.50</price>
      <amount>10</amount>
    </pop>
  </drink>
  <snack>
    <chips>
      <price>$4.50</price>
      <amount>60</amount>
    </chips>
  </snack>
</inventory>
```



Signatur:

binäre Relationssymbole **succ** für „successor“ und **sord** für „successor order“

sowie ein unäres Relationssymbol für jedes Tag (**drink**, **snack** usw.)

T2.4

Strukturen – Beispiel 4

Strukturen aus der Mathematik, z. B. Arithmetik der natürlichen Zahlen:

$$\mathfrak{N} = (\mathbb{N}, +^{\mathfrak{N}}, \cdot^{\mathfrak{N}}, 0^{\mathfrak{N}}, 1^{\mathfrak{N}}) \quad (\text{unendlich!})$$

wobei

- $+^{\mathfrak{N}}, \cdot^{\mathfrak{N}}$ die natürliche Interpretation von $+$ und \cdot sind:

$$+^{\mathfrak{N}}(x, y) = x + y \quad \cdot^{\mathfrak{N}}(x, y) = x \cdot y$$

- $0^{\mathfrak{N}} = 0$ und $1^{\mathfrak{N}} = 1$

(0, 1 sowohl Konstantensymbole als auch Elemente des Universums)

Bei offensichtlicher Interpretation lassen wir das $\cdot^{\mathfrak{N}}$ oft weg, also z.B.

$$\mathfrak{N} = (\mathbb{N}, +, \cdot, 0, 1)$$

Analog definiert man z.B. $\mathfrak{R} = (\mathbb{R}, +, \cdot, 0, 1)$

(überabzählbar!)

Strukturen – Beispiel 5

Auch Ordnungen lassen sich als Struktur auffassen, z.B.:

- $\mathfrak{N}_{<} = (\mathbb{N}, <)$
 - $\mathfrak{R}_{<} = (\mathbb{R}, <)$
- („<“ binäres Relationssymbol)

In der Informatik werden solche Strukturen oft als Repräsentation von Zeit aufgefasst, die Elemente von \mathbb{N} bzw. \mathbb{R} sind dann die Zeitpunkte

Man kann auch zusätzliche unäre Relationssymbole zulassen, also

$$\mathfrak{A} = (\mathbb{N}, <, P_1^{\mathfrak{A}}, P_2^{\mathfrak{A}}, \dots)$$

wobei eine beliebige Interpretation der P_1, P_2, \dots möglich ist

Mögliche Interpretation:

Jedes P_i repräsentiert eine Aussage (im Sinn der Aussagenlogik),
 $x \in P_i^{\mathfrak{A}}$ bedeutet: „Aussage P_i ist wahr zum Zeitpunkt x “

T2.5

Prädikatenlogik



2.1 Strukturen

2.2 Syntax und Semantik der Prädikatenlogik

2.3 Auswertung und Datenbanken

2.4 Äquivalenz, Erfüllbarkeit, Gültigkeit

2.5 Pränex-Normalform

2.6 Unentscheidbarkeit

2.7 Theorien

Syntax

Analog zu den zwei verschiedenen Zutaten von Signaturen und Strukturen (Relationssymbole und Funktionssymbole):

Formeln der Prädikatenlogik bestehen aus zwei Bestandteilen:

- **Terme**, die aus (Objekt)variablen, Konstanten- und Funktionssymbolen gebildet werden
- **Formeln** bestehen dann aus Termen, den Booleschen Operatoren, Quantoren und Relationssymbolen

Wir definieren die Syntax daher in zwei Schritten.

Syntax (Schritt 1)

Wir fixieren eine abzählbar unendliche Menge $\text{VAR} = \{x_1, x_2, x_3, \dots\}$ von *Objektvariablen*.

Definition Term

Die Menge der *Terme* ist induktiv wie folgt definiert:

- jede Variable ist ein Term.
- wenn t_1, \dots, t_n Terme sind und f ein n -stelliges Funktionssymbol, dann ist auch $f(t_1, \dots, t_n)$ ein Term

Beachte: jedes Konstantensymbol ist damit ebenfalls ein Term!

Beispiele:

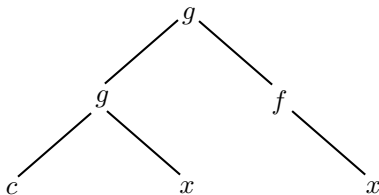
$$x, c, f(x), g(x, x), g(f(x), c), g(g(c, c), f(x))$$

$$1 + ((1 + 1) \cdot 1)$$

$$1 + ((x + 1) \cdot y)$$

Sprechweisen und Konventionen

- Wir bezeichnen Terme mit s und t
- Es ist oft nützlich, Terme als Bäume aufzufassen, z.B. $g(g(c, x), f(x))$ als



- Für Funktionssymbole wie $+$ und \cdot verwenden wir Infix-Notation, also $x + c$ statt $+(x, c)$

Syntax (Schritt 2)

Definition FO-Formeln

Die Menge der *Formeln* der Prädikatenlogik ist induktiv wie folgt definiert:

- sind t_1, t_2 Terme, dann ist $t_1 = t_2$ eine Formel
- sind t_1, \dots, t_n Terme und P ein n -stelliges Relationssymbol, dann ist $P(t_1, \dots, t_n)$ eine Formel
- wenn φ und ψ Formeln sind, dann auch $\neg\varphi$, $(\varphi \wedge \psi)$, $(\varphi \vee \psi)$
- wenn φ eine Formel ist und $x \in \text{VAR}$, dann sind $\exists x \varphi$ und $\forall x \varphi$ Formeln

Die Menge aller Formeln über einer Signatur τ bezeichnen wir mit $\text{FO}(\tau)$.

Beispiele: $x = c$ $(P(x) \wedge Q(x)) \vee P(y)$ $\forall x \exists y P(x, f(y))$

$\forall x (\exists y \text{neben}(y, x) \vee \exists y \text{auf}(y, x))$

$\exists y (\text{Film}(x, y, \text{Hitchcock}) \wedge \text{Schauspieler}(\text{Connery}, x))$

Sprechweisen und Konventionen

- Atome und deren Negation heißen *Literale*
- Statt $\neg(t = t')$ schreiben wir auch $t \neq t'$
- \rightarrow und \leftrightarrow sind analog zur AL definiert
- Klammern werden weggelassen, wenn das Resultat eindeutig ist, wobei \neg, \exists, \forall stärker binden als \wedge, \vee und \wedge, \vee stärker binden als $\rightarrow, \leftrightarrow$

Also z. B. $\exists x P(x) \vee Q(x)$ für $(\exists x P(x)) \vee Q(x)$,
nicht für $\exists x (P(x) \vee Q(x))$

Freie und gebundene Variablen

Ein *Vorkommen* einer Variable in einer Formel kann durch einen Quantor *gebunden* sein oder nicht (dann ist die Variable *frei*)

Beispiel:

$$\begin{array}{ccc} \varphi = P(x) \wedge \exists x Q(x) & & \\ | & & | \\ \text{frei} & & \text{gebunden} \end{array}$$

Einige Konventionen:

- Wenn wir eine Formel mit $\varphi(x_1, \dots, x_n)$ bezeichnen, so sind x_1, \dots, x_n die freien Variablen in φ ; für obige Formel: $\varphi(x)$
- Formeln ohne freie Variablen heißen *Satz*

Freie und gebundene Variablen

Präzise definiert man die Menge der freien Variablen wie folgt.

Mit $\text{Var}(\varphi)$ bezeichnen wir die Menge der in der Formel φ vorkommenden Variablen.

Definition Freie Variable

Sei φ eine Formel. Die Menge $\text{Frei}(\varphi)$ der *freien Variablen* von φ ist induktiv wie folgt definiert:

- Für atomare Formeln φ ist $\text{Frei}(\varphi) = \text{Var}(\varphi)$
- $\text{Frei}(\neg\varphi) = \text{Frei}(\varphi)$
- $\text{Frei}(\varphi \wedge \psi) = \text{Frei}(\varphi \vee \psi) = \text{Frei}(\varphi) \cup \text{Frei}(\psi)$
- $\text{Frei}(\exists x \varphi) = \text{Frei}(\forall x \varphi) = \text{Frei}(\varphi) \setminus \{x\}$

Semantik (Schritt 1)

Struktur interpretiert nur Symbole, aber keine Variablen – dafür Zuweisung:

Definition Zuweisung

Sei \mathfrak{A} eine Struktur. Eine *Zuweisung* in \mathfrak{A} ist eine Abbildung $\beta : \text{VAR} \rightarrow A$. Man erweitert β wie folgt induktiv auf Terme:

- wenn $t = f(t_1, \dots, t_k)$, dann $\beta(t) = f^{\mathfrak{A}}(\beta(t_1), \dots, \beta(t_k))$

Ein Paar (\mathfrak{A}, β) mit β Zuweisung in \mathfrak{A} heißt *Interpretation*.

Beachte: der implizite Induktionsanfang ist:

- wenn $t = x \in \text{VAR}$, dann $\beta(t) = \beta(x)$
- wenn $t = c$ Konstante, dann $\beta(t) = c^{\mathfrak{A}}$

T2.6

Semantik (Schritt 2)

Definition Semantik von FO

Wir definieren Erfülltheitsrelation \models zwischen Interpretationen (\mathfrak{A}, β) und FO-Formeln induktiv wie folgt:

- $\mathfrak{A}, \beta \models t = t'$ gdw. $\beta(t) = \beta(t')$
- $\mathfrak{A}, \beta \models P(t_1, \dots, t_n)$ gdw. $(\beta(t_1), \dots, \beta(t_n)) \in P^{\mathfrak{A}}$
- $\mathfrak{A}, \beta \models \neg \varphi$ gdw. $\mathfrak{A}, \beta \not\models \varphi$
- $\mathfrak{A}, \beta \models \varphi \wedge \psi$ gdw. $\mathfrak{A}, \beta \models \varphi$ und $\mathfrak{A}, \beta \models \psi$
- $\mathfrak{A}, \beta \models \varphi \vee \psi$ gdw. $\mathfrak{A}, \beta \models \varphi$ oder $\mathfrak{A}, \beta \models \psi$
- $\mathfrak{A}, \beta \models \exists x \varphi$ gdw. ein $a \in A$ existiert mit $\mathfrak{A}, \beta[x/a] \models \varphi$
- $\mathfrak{A}, \beta \models \forall x \varphi$ gdw. für alle $a \in A$ gilt, dass $\mathfrak{A}, \beta[x/a] \models \varphi$

Wie β , außer
 $x \mapsto a$

Wenn $\mathfrak{A}, \beta \models \varphi$, dann ist (\mathfrak{A}, β) ein *Modell* für φ .

T2.7

Koinzidenzlemma

Analog zur Aussagenlogik: $(\mathfrak{A}, \beta) \models \varphi$ ist unabhängig von Interpretation der Symbole und Variablen, die in φ gar nicht (bzw. nicht frei) vorkommen.

$\text{sig}(\varphi)$ bezeichne die Signatur der Formel φ , also die Menge der in φ vorkommenden Relations- und Funktionssymbole

Koinzidenzlemma

Sei φ eine FO-Formel und $(\mathfrak{A}, \beta), (\mathfrak{A}', \beta')$ Interpretationen, so dass

- $A = A'$;
- $S^{\mathfrak{A}} = S^{\mathfrak{A}'}$ für alle $S \in \text{sig}(\varphi)$
- für alle $x \in \text{Frei}(\varphi)$ gilt: $\beta(x) = \beta'(x)$

Dann $\mathfrak{A}, \beta \models \varphi$ gdw. $\mathfrak{A}', \beta' \models \varphi$

Beweis per Induktion über die Struktur von φ .

Koinzidenzlemma

Wenn wir mit einer Formel φ arbeiten, so erlaubt uns das Koinzidenzlemma, in Zuweisungen nur die Variablen $\text{Frei}(\varphi)$ zu betrachten.

Das erlaubt insbesondere folgende Notation:

Für eine Formel $\varphi(x_1, \dots, x_k)$ schreiben wir

$$\mathfrak{A} \models \varphi[a_1, \dots, a_k]$$

wenn $\mathfrak{A}, \beta \models \varphi$, wobei $\beta(x_i) = a_i$ für $1 \leq i \leq k$

Wenn φ Satz ist, dann wird daraus einfach $\mathfrak{A} \models \varphi$

Isomorphielemma

Es existiert ein Isomorphismus zwischen zwei Strukturen gdw. diese sich nur durch Umbenennen der Elemente des Universums unterscheiden.

Definition Isomorphismus

Seien \mathfrak{A} und \mathfrak{B} Strukturen. Bijektion $\pi : A \rightarrow B$ ist *Isomorphismus*, wenn folgende Bedingungen erfüllt sind:

- Für jedes n -stellige Relationssymbol P und alle $a_1, \dots, a_n \in A^n$:
 $(a_1, \dots, a_n) \in P^{\mathfrak{A}}$ gdw. $(\pi(a_1), \dots, \pi(a_n)) \in P^{\mathfrak{B}}$
- Für jedes n -stellige Funktionssymbol f und alle $a_1, \dots, a_n \in A^n$:
 $\pi(f^{\mathfrak{A}}(a_1, \dots, a_n)) = f^{\mathfrak{B}}(\pi(a_1), \dots, \pi(a_n))$

Beispiel

T2.8

Isomorphielemma

Isomorphielemma

Seien $\mathfrak{A}, \mathfrak{B}$ Strukturen und $\pi : A \rightarrow B$ ein Isomorphismus.
 Dann gilt für alle Formeln $\varphi(x_1, \dots, x_n)$ und alle $a_1, \dots, a_n \in A$:

$$\mathfrak{A} \models \varphi[a_1, \dots, a_n] \text{ gdw. } \mathfrak{B} \models \varphi[\pi(a_1), \dots, \pi(a_n)]$$

T2.9

Insbesondere gilt also für alle Sätze φ : $\mathfrak{A} \models \varphi$ gdw. $\mathfrak{B} \models \varphi$.

Intuitiv:

- FO kann nicht zwischen isomorphen Strukturen unterscheiden
- Die Namen der Elemente des Universums sind Schall und Rauch

Prädikatenlogik

- 2.1 Strukturen
- 2.2 Syntax und Semantik der Prädikatenlogik
- 2.3 Auswertung und Datenbanken**
- 2.4 Äquivalenz, Erfüllbarkeit, Gültigkeit
- 2.5 Pränex-Normalform
- 2.6 Unentscheidbarkeit
- 2.7 Theorien



Auswertung

Definition Auswertungsproblem

Das *Auswertungsproblem der Prädikatenlogik* ist:

Gegeben: FO-Formel $\varphi(x_1, \dots, x_n)$, endliche Interpretation (\mathfrak{A}, β)
 mit β Zuweisung für x_1, \dots, x_n

Frage: Gilt $\mathfrak{A}, \beta \models \varphi$?

Theorem

Das Auswertungsproblem der Prädikatenlogik erster Stufe ist PSpace-vollständig.

Wir wollen hier nur Entscheidbarkeit in PSpace beweisen.

PSpace-Härte zeigt man über eine Reduktion von QBF,
 (siehe VL Komplexitätstheorie)

Auswertung

Function $\text{ausw}(\mathfrak{A}, \beta, \varphi)$:

input : endl. Interpretation (\mathfrak{A}, β) , FO-Formel φ

output: Wahrheitswert: 1 für $\mathfrak{A}, \beta \models \varphi$, 0 für $\mathfrak{A}, \beta \not\models \varphi$

switch φ **do**

case $\varphi = (t = t')$: **if** $\beta(t) = \beta(t')$ **then return 1 else return 0**

case $\varphi = P(t_1, \dots, t_k)$:

if $(\beta(t_1), \dots, \beta(t_k)) \in P^{\mathfrak{A}}$ **then return 1 else return 0**

case $\varphi = \neg\psi$: **return** $1 - \text{ausw}(\mathfrak{A}, \beta, \psi)$

case $\varphi = \psi \wedge \vartheta$: **return** $\min\{\text{ausw}(\mathfrak{A}, \beta, \psi), \text{ausw}(\mathfrak{A}, \beta, \vartheta)\}$

case $\varphi = \psi \vee \vartheta$: **return** $\max\{\text{ausw}(\mathfrak{A}, \beta, \psi), \text{ausw}(\mathfrak{A}, \beta, \vartheta)\}$

case $\varphi = \exists x \psi$:

 rufe $\text{ausw}(\mathfrak{A}, \beta[x/a], \psi)$ für alle $a \in A$

if ein Ruf erfolgreich then return 1 else return 0

case $\varphi = \forall x \psi$:

 rufe $\text{ausw}(\mathfrak{A}, \beta[x/a], \psi)$ für alle $a \in A$

if alle Rufe erfolgreich then return 1 else return 0

T2.10

Lemma

Der Algorithmus

1. ist korrekt: $\text{ausw}(\mathfrak{A}, \beta, \varphi) = 1$ gdw. $\mathfrak{A}, \beta \models \varphi$
2. benötigt nur polynomiell viel Platz

Für den Beweis:

T2.11

Die *Schachteltiefe* $\text{st}(\varphi)$ einer Formel φ ist induktiv definiert wie folgt:

- $\text{st}(t = t') = \text{st}(P(t_1, \dots, t_k)) = 0$
- $\text{st}(\neg\varphi) = \text{st}(\exists x \varphi) = \text{st}(\forall x \varphi) = \text{st}(\varphi) + 1$
- $\text{st}(\varphi \wedge \psi) = \text{st}(\varphi \vee \psi) = \max\{\text{st}(\varphi), \text{st}(\psi)\} + 1$

Leicht per Induktion zu zeigen: $\text{st}(\varphi) \leq |\varphi|$ ($|\varphi|$ ist Länge von φ)

PS: Der Algorithmus benötigt natürlich exponentiell viel Zeit

Man kann FO auf natürliche Weise als *Anfragesprache* für DBen sehen:

- schon gesehen: Datenbankinstanz \approx relationale Struktur
- Antwort auf FO-Anfrage $\varphi(x_1, \dots, x_n)$ bzgl. Datenbankinstanz \mathfrak{A} :

$$\text{ans}(\mathfrak{A}, \varphi) = \{(a_1, \dots, a_n) \in A^n \mid \mathfrak{A} \models \varphi[a_1, \dots, a_n]\}$$

Film:

Titel	Jahr	Regisseur
Die Vögel	1963	Hitchcock
Marnie	1964	Hitchcock
Goldfinger	1964	Hamilton

Schauspieler_in:

Name	Filmtitel
Connery	Marnie
Connery	Goldfinger
Hedren	Die Vögel

$$\varphi = \exists y (\text{Film}(\underline{x}, y, \text{Hitchcock}) \wedge \text{Schauspieler}(\text{Connery}, \underline{x}))$$

$$\text{ans}(\mathfrak{A}, \varphi) = \{\text{Marnie}\}$$

T2.12

FO und Datenbanken

In diesem Zusammenhang wird FO auch das *relationale Kalkül* genannt

FO/das relationale Kalkül ist im Wesentlichen nichts anderes als SQL!

Beispiele:

```
SELECT Titel FROM Film WHERE Regisseur = Hitchcock
```

$\exists y \text{ Film}(\underline{x}, y, \text{Hitchcock})$

```
SELECT Name, Jahr FROM Schauspieler, Film
WHERE Schauspieler.Titel = Film.Titel
```

$\exists z \exists z' (\text{Schauspieler}(\underline{x}, z) \wedge \text{Film}(z, \underline{y}, z'))$

FO und Datenbanken

Sei *Kern-SQL* die Einschränkung von SQL auf

```
SELECT FROM WHERE (in Bedingungen sind = und AND erlaubt),
UNION,
MINUS
```

Die meisten anderen Elemente von SQL dienen nur der Benutzbarkeit, erhöhen aber nicht die Ausdrucksstärke

Nicht sehr schwer:

jede Kern-SQL-Anfrage kann in äquivalente FO-Anfrage übersetzt werden (äquivalent = dieselben Antworten auf jeder Datenbank)

Für die Übersetzung FO \Rightarrow SQL muss man eine Einschränkung machen:

Domänenunabhängigkeit der FO-Anfrage

Intuitiv: Anfrage ist *domänenunabhängig*, wenn Antworten *nicht* von Elementen abhängen, die in *keiner* Relation/Tabelle vorkommen.

Definition Domänenunabhängigkeit

Eine FO-Formel φ ist *domänenunabhängig*, wenn für alle Strukturen $\mathfrak{A} = (A, P_1^{\mathfrak{A}}, P_2^{\mathfrak{A}}, \dots)$ und $B \supseteq A$ gilt:

$$\text{ans}(\mathfrak{A}, \varphi) = \text{ans}((B, P_1^{\mathfrak{A}}, P_2^{\mathfrak{A}}, \dots), \varphi).$$

T2.13

Domänen*abhängige* FO-Anfragen sind meist *sinnlos*:

$$\neg \exists y \text{Schauspieler}(x, y)$$

liefert alle in der Datenbank verwendeten Strings und Zahlen *außer* Connery und Hedren

Alle Übersetzungen von Kern-SQL-Anfragen in FO sind domänenunabhängig, z. B.:

$$\exists y \text{Film}(x, y, \text{Hitchcock})$$

denn Elemente aus $A \setminus \bigcup_{i \geq 1} P_i^{\mathfrak{A}}$ werden nie als Antwort zurückgegeben

Die Umkehrung gilt auch, aber modulo Äquivalenz:

Folgendes Resultat von 1970 ist die Grundlage für die Entwicklung der relationalen Datenbanksysteme.

(Codd arbeitete bei IBM, implementierte die erste relationale Datenbank „System R“)

Theorem (Codd)

Jede domänen*unabhängige* FO-Anfrage ist *äquivalent* zu einer Anfrage in Kern-SQL und umgekehrt. Die Übersetzung benötigt nur lineare Zeit.

Unser Algorithmus für FO-Auswertung kann also auch zur SQL-Anfragebeantwortung verwendet werden!

- 2.1 Strukturen
- 2.2 Syntax und Semantik der Prädikatenlogik
- 2.3 Auswertung und Datenbanken
- 2.4 Äquivalenz, Erfüllbarkeit, Gültigkeit**
- 2.5 Pränex-Normalform
- 2.6 Unentscheidbarkeit
- 2.7 Theorien

NEXT



Äquivalenz

Definition Äquivalenz

Zwei FO-Formeln φ and ψ mit $\text{Frei}(\varphi) = \text{Frei}(\psi)$ sind *äquivalent*, wenn für alle Interpretationen (\mathfrak{A}, β) gilt: $\mathfrak{A}, \beta \models \varphi$ gdw. $\mathfrak{A}, \beta \models \psi$.
Wir schreiben dann $\varphi \equiv \psi$.

Der Begriff einer *Teilformel* einer FO-Formel kann auf die offensichtliche Weise induktiv definiert werden, analog zur Aussagenlogik.

Auch in FO sind äquivalente Formeln austauschbar:

Ersetzungslemma

Seien φ und ψ äquivalente FO-Formeln, ϑ eine Formel mit $\varphi \in \text{TF}(\vartheta)$ und ϑ' eine Formel, die sich aus ϑ ergibt, indem ein beliebiges Vorkommen von φ durch ψ ersetzt wird. Dann gilt $\vartheta \equiv \vartheta'$.

Beweis per Induktion über die Struktur von ϑ .

Äquivalenz

Leicht zu sehen: alle Äquivalenzen aus der Aussagenlogik gelten auch in FO, z. B.:

$$\varphi \wedge \psi \equiv \neg(\neg\varphi \vee \neg\psi) \quad \text{für beliebige FO-Formeln } \varphi, \psi$$

Natürlich gibt es auch interessante FO-spezifische Äquivalenzen, z. B.:

- $\forall x \varphi \equiv \neg \exists x \neg \varphi$ (Dualität von \exists und \forall)
- $\exists x (\varphi \vee \psi) \equiv \exists x \varphi \vee \exists x \psi$ (\exists distribuiert über \vee)
- $\forall x (\varphi \wedge \psi) \equiv \forall x \varphi \wedge \forall x \psi$ (\forall distribuiert über \wedge)
- $\exists x \exists y \varphi \equiv \exists y \exists x \varphi$
- $\forall x \forall y \varphi \equiv \forall y \forall x \varphi$

Äquivalenz

FO-Formel heißt *reduziert*, wenn sie nur die Junktoren \neg, \wedge und nur den Quantor \exists enthält

Lemma

Jede FO-Formel kann in Linearzeit in eine äquivalente *reduzierte* FO-Formel gewandelt werden.

Beweis klar wegen

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

$$\forall x \varphi \equiv \neg \exists x \neg \varphi$$

In Induktionsbeweisen müssen wir also nur \neg, \wedge, \exists betrachten

Erfüllbarkeit, Gültigkeit, Konsequenz

Folgende Begriffe sind exakt analog zur Aussagenlogik:

Definition Erfüllbarkeit, Gültigkeit, Konsequenz

Ein Satz φ heißt

- *erfüllbar*, wenn er ein Modell hat (Struktur, die ihn wahr macht)
- *gültig* oder *Tautologie*, wenn jede Struktur ein Modell von φ ist
- *Konsequenz* von Satz ψ , wenn für alle Strukturen \mathfrak{A} mit $\mathfrak{A} \models \psi$ auch $\mathfrak{A} \models \varphi$ gilt (wir schreiben dann $\psi \models \varphi$)

T2.14

Man kann diese Begriffe natürlich auch für Formeln mit freien Variablen definieren; verwendet dann Interpretationen statt Strukturen.

- 2.1 Strukturen
- 2.2 Syntax und Semantik der Prädikatenlogik
- 2.3 Auswertung und Datenbanken
- 2.4 Äquivalenz, Erfüllbarkeit, Gültigkeit
- NEXT** → **2.5 Pränex-Normalform**
- 2.6 Unentscheidbarkeit
- 2.7 Theorien

FO-Formel φ ist *bereinigt*, wenn

- keine Variable in φ sowohl frei als auch gebunden auftritt
- keine Variable mehr als einmal quantifiziert wird

Jede Formel kann leicht durch *Umbenennung quantifizierter Variablen* bereinigt werden, z.B.:

$$\exists y (P(x, y) \wedge \forall x Q(x, y)) \text{ äquivalent zu } \exists y (P(x, y) \wedge \forall z Q(z, y))$$

Definition Pränex-Normalform

Eine FO-Formel φ ist in *Pränex-Normalform (PNF)*, wenn sie bereinigt ist und die Form

$$Q_1 x_1 \cdots Q_n x_n \varphi$$

hat, wobei $Q_i \in \{\exists, \forall\}$ und φ quantorenfrei.

Theorem

Jede FO-Formel kann in Linearzeit in eine äquivalente Formel in PNF gewandelt werden.

Für den Beweis benötigen wir folgende Äquivalenzen:

Falls x nicht frei in φ vorkommt, gilt:

- $\varphi \vee \exists x \psi \equiv \exists x (\varphi \vee \psi)$
- $\varphi \wedge \exists x \psi \equiv \exists x (\varphi \wedge \psi)$
- $\varphi \vee \forall x \psi \equiv \forall x (\varphi \vee \psi)$
- $\varphi \wedge \forall x \psi \equiv \forall x (\varphi \wedge \psi)$

T2.15

T2.16

T2.17

Beweis des Theorems

Beispiel

- 2.1 Strukturen
- 2.2 Syntax und Semantik der Prädikatenlogik
- 2.3 Auswertung und Datenbanken
- 2.4 Äquivalenz, Erfüllbarkeit, Gültigkeit
- 2.5 Pränex-Normalform
- NEXT** → **2.6 Unentscheidbarkeit**
- 2.7 Theorien

Unentscheidbarkeit

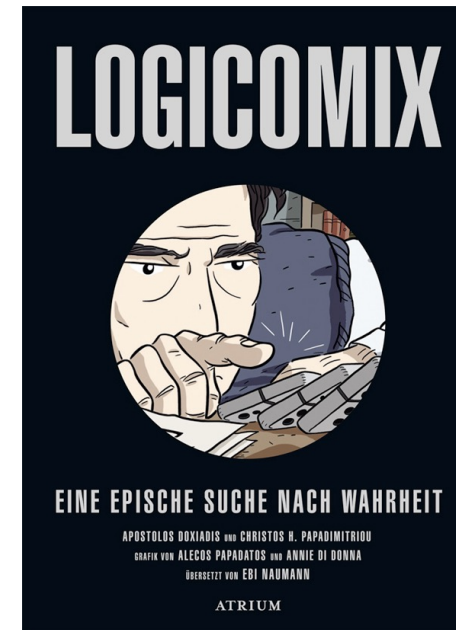
Bis in die 1930er **hofften** viele Mathematiker, dass die Prädikatenlogik oder ähnlich ausdrucksstarke Logiken **entscheidbar** sein würden.

Besonders prominent ist **Hilbert**, der 1928 die Lösung des „Entscheidungsproblems“ der Logik als **eines der wichtigsten offenen Probleme der Mathematik** bezeichnet hat.

Da wichtige Teile der Mathematik in FO formalisierbar (z. B. Gruppentheorie): viele **manuelle mathematische Beweise** könnten durch **automatische** ersetzt werden.

Aber das wäre dann doch zu schön, um wahr zu sein ...

Lesetipp



Unentscheidbarkeit

Wir zeigen, dass die **Gültigkeit** von FO-Formeln **unentscheidbar** ist.
(gegeben ein FO-Satz, entscheide ob dieser gültig ist)

Unentscheidbarkeit von Erfüllbarkeit und Konsequenz folgt dann per Reduktion von Gültigkeit, denn analog zur Aussagenlogik gilt:

- Satz φ ist gültig gdw. $\neg\varphi$ unerfüllbar ist
- Satz φ ist gültig gdw. $\varphi_{\text{taut}} \models \varphi$, wobei φ_{taut} beliebige Tautologie

Der Beweis ist per **Reduktion des Postschen Korrespondenzproblems**

Unentscheidbarkeit

Wir verwenden eine Reduktion des Postschen Korrespondenzproblems

Definition Postsches Korrespondenzproblem (PCP)

Gegeben: Eine Folge $F = (u_1, v_1), \dots, (u_k, v_k)$ von Wortpaaren,
mit $u_i, v_i \in \{0, 1\}^*$

Frage: Gibt es eine Indexfolge i_1, \dots, i_ℓ , so dass $u_{i_1} \dots u_{i_\ell} = v_{i_1} \dots v_{i_\ell}$?

Eine solche Folge heißt **Lösung** für F .

T2.18

Bekannt aus VL „Theoretische Informatik 2“:

Theorem (Post)

Das PCP ist unentscheidbar.

Unentscheidbarkeit

Ziel: Für gegebenes PCP F einen FO-Satz φ_F konstruieren, so dass:
 F hat eine Lösung gdw. φ_F gültig ist.

Verwendete Signatur:

- ein Konstantensymbol c_ε
- zwei einstellige Funktionssymbole f_0 und f_1
- ein zweistelliges Relationssymbol P

Intuition:

- c_ε, f_0, f_1 erzeugen alle Wörter
- P kennzeichnet Wortpaare, die F erzeugen kann

T2.19

Unentscheidbarkeit

Schreibweise:

für Wort $w = w_1 \cdots w_n \in \{0, 1\}^*$ steht $t_w(x)$ für $f_{w_n}(f_{w_{n-1}}(\cdots f_{w_1}(x)))$

Für PCP $F = (u_1, v_1), \dots, (u_k, v_k)$ setze

$$\varphi_F = (\varphi \wedge \psi) \rightarrow \exists x P(x, x)$$

wobei

$$\varphi = \bigwedge_{i=1, \dots, k} P(t_{u_i}(c_\varepsilon), t_{v_i}(c_\varepsilon))$$

$$\psi = \forall x \forall y \left(P(x, y) \rightarrow \bigwedge_{i=1, \dots, k} P(t_{u_i}(x), t_{v_i}(y)) \right)$$

Lemma

F hat eine Lösung gdw. φ_F gültig ist.

T2.20

Unentscheidbarkeit

Theorem (Church, Turing)

In FO sind Gültigkeit, Erfüllbarkeit, Konsequenz unentscheidbar.

Unentscheidbarkeit gilt auch für *relationale* Signaturen:

- ersetze c_ε durch unäres Relationssymbol A_ε
- ersetze f_0, f_1 durch binäre Relationssymbole P_0, P_1
- erzwingen das „richtige Verhalten“:

$$\begin{aligned} & \exists x (A_\varepsilon(x) \wedge \forall y (A_\varepsilon(y) \rightarrow x = y)) \\ & \forall x \exists y P_i(x, y) \\ & \forall x \forall y \forall z ((P_i(x, y) \wedge P_i(x, z)) \rightarrow y = z) \end{aligned} \quad \text{für } i \in \{0, 1\}$$

Beachte: Datenbanken haben rein relationale Signaturen.

Auch die Gleichheit ist durch beliebiges Relationssymbol simulierbar.

Unentscheidbarkeit bzgl. endlicher Modelle

Beachte:

Die vorgestellte Reduktion erfordert **unendliche Modelle**.

In der Informatik benötigt man aber meist nur **endliche Modelle** (z. B. Datenbanken).

Das liefert unterschiedliche Begriffe von Erfüllbarkeit, Tautologie etc.

Z. B. ist folgende Formel erfüllbar, aber nicht endlich erfüllbar

$$\begin{aligned} & \forall x \neg R(x, c) \wedge \\ & \forall x \exists y R(x, y) \wedge \\ & \forall x \forall x' \forall y (R(x, y) \wedge R(x', y) \rightarrow x = x') \end{aligned}$$

T2.21

Ihre Negation ist also eine Tautologie in endlichen Modellen, aber nicht im Allgemeinen.

Theorem (Trakhtenbrot)

Folgende Probleme sind unentscheidbar:

- Endliche Gültigkeit:
Ist eine FO-Formel in allen endlichen Interpretationen erfüllt?
- Endliche Erfüllbarkeit:
Hat eine FO-Formel ein endliches Modell?
- Endliche Konsequenz:
Gilt für zwei FO-Formeln φ, ψ und alle endlichen Interpretationen (\mathfrak{A}, β) mit $\mathfrak{A}, \beta \models \varphi$ auch $\mathfrak{A}, \beta \models \psi$?

Beweis z. B. durch Reduktion des Halteproblems

Da alle Formeln im Beweis von Trakhtenbrot's Theorem **domänen-unabhängig** sind, sind auch folgende SQL-Probleme **unentscheidbar**:

- Gegeben eine SQL-Anfrage, entscheide ob es eine Datenbank-Instanz gibt, für die die Anfrage eine nicht-leere Antwort liefert
- Gegeben zwei SQL-Anfragen, entscheide ob für jede Datenbank-Instanz gilt: die Antwort für die erste Anfrage ist eine Teilmenge der Antwort für die zweite Anfrage (*Query containment*)
- Gegeben zwei SQL-Anfragen, entscheide ob sie für alle Datenbank-Instanzen dieselben Antworten liefern.

Diese Probleme sind von praktischer Bedeutung, z. B. für die Anfrageoptimierung in relationalen Datenbanksystemen.

Trotz Unentscheidbarkeit ...

Es gibt aber auch Positives zu berichten:

- Die gültigen FO-Formeln sind *rekursiv aufzählbar* (Teil 3); dies ist die Grundlage für automatisches Theorembeweisen
- Intuitiv:
- Wenn man den Beweiser nach einem *gültigen* mathematischen Theorem fragt, findet er schließlich einen Beweis.
 - Wenn man den Beweiser nach einem *nicht gültigen* Theorem fragt, terminiert er nicht.
- Über verschiedenen wichtigen Strukturklassen wie *Wörtern und Bäumen* erhält man *Entscheidbarkeit* (Teil 4) — sogar für Logik 2. Stufe.

Wichtige Anwendungen in der Verifikation

Trotz Unentscheidbarkeit ...

Es gibt aber auch Positives zu berichten:

- Verschiedene *syntaktische Einschränkungen* liefern *Entscheidbarkeit*:
 1. Nur unäre Relationssymbole, keine Funktionssymbole
 2. Nur 2 Variablen statt unendlich viele
 3. Formeln in PNF mit eingeschränktem Quantorenpräfix, z.B. $\exists^* \forall^*$
 4. Guarded Fragment: bei $\exists x \varphi$ und $\forall x \varphi$ Form von φ eingeschränkt

Wichtige Anwendungen in der KI und der Verifikation

- Im Folgenden: verschiedene wichtige *FO-Theorien* sind *entscheidbar*
- Wichtig z. B. für das Theorembeweisen in der Mathematik

- 2.1 Strukturen
- 2.2 Syntax und Semantik der Prädikatenlogik
- 2.3 Auswertung und Datenbanken
- 2.4 Äquivalenz, Erfüllbarkeit, Gültigkeit
- 2.5 Pränex-Normalform
- 2.6 Unentscheidbarkeit

NEXT



2.7 Theorien

Manche Strukturen haben eine besonders große Bedeutung, z. B.:

Arithmetik der natürlichen Zahlen $(\mathbb{N}, +, \cdot, 0, 1)$

Die in dieser Struktur erfüllten FO-Sätze sind mathematisch von großer Bedeutung.

Bereits gesehen:

die Existenz unendlich vieler Primzahl-Zwillinge ist in FO beschreibbar.

Weiteres Beispiel: Goldbachsche Vermutung

$$\forall x (x > 2 \wedge \text{Even}(x) \rightarrow \exists y \exists y' (\text{Prim}(y) \wedge \text{Prim}(y') \wedge x = y + y'))$$

wobei $\text{Even}(x) = \exists y (x = y + y)$, etc.

Theorien sind „kohärente“ Mengen von FO-Sätzen,

z. B. diejenigen Sätze, die in einer ausgewählten Struktur wahr sind.

Mengen von Formeln (endlich oder unendlich):

Begriffe wie Modell, Erfüllbarkeit und Konsequenz übertragen sich leicht

Z.B. ist \mathfrak{A} *Modell* für Menge Γ von Sätzen gdw. $\mathfrak{A} \models \varphi$ für alle $\varphi \in \Gamma$

Für die folgende Definition nehmen wir (implizit) eine feste Signatur τ an, über der alle Formeln gebildet werden.

Definition FO-Theorie

Eine *FO-Theorie* ist eine erfüllbare Menge T von FO-Sätzen, die unter Konsequenz abgeschlossen ist:

$$T \models \varphi \text{ impliziert } \varphi \in T \quad \text{für alle Sätze } \varphi$$

T heißt *vollständig*, wenn für alle Sätze φ gilt: $\varphi \in T$ oder $\neg\varphi \in T$

Beispiele:

T2.22

1. Menge aller Tautologien (in einer fixen Signatur τ) ist FO-Theorie
enthalten in allen anderen Theorien, nicht vollständig

2. Sei \mathfrak{A} eine τ -Struktur. Dann ist

$$\text{Th}(\mathfrak{A}) = \{\varphi \text{ ist } \tau\text{-Satz} \mid \mathfrak{A} \models \varphi\}$$

eine vollständige FO-Theorie.

3. Wenn Ω erfüllbare Menge von FO-Sätzen, dann ist

$$\text{Abschluss}(\Omega) = \{\varphi \text{ ist } \tau\text{-Satz} \mid \Omega \models \varphi\}$$

FO-Theorie (im Allgemeinen nicht vollständig)

Beispiele für FO-Theorien

4. Sei \mathcal{K} eine Klasse von τ -Strukturen. Dann ist

$$\text{Th}(\mathcal{K}) = \bigcap_{\mathfrak{A} \in \mathcal{K}} \text{Th}(\mathfrak{A})$$

eine FO-Theorie (im Allgemeinen nicht vollständig).

Entscheidbarkeit von Theorien

Definition Entscheidbarkeit von Theorien

Theorie Γ ist *entscheidbar*, wenn folgendes Problem entscheidbar ist:

Gegeben: ein FO-Satz φ

Frage: ist $\varphi \in \Gamma$?

Wenn $\Gamma = \text{Th}(\mathfrak{A})$, dann ist das also folgendes Problem:

Gegeben φ , entscheide ob $\mathfrak{A} \models \varphi$.

Das ist **nicht** Erfüllbarkeit oder Gültigkeit, denn wir reden über eine feste Struktur anstatt über alle Strukturen.

Das ist **nicht** das bereits besprochene Auswertungsproblem, denn interessante \mathfrak{A} sind unendlich!

Einige wichtige FO-Theorien

Arithmetik der natürlichen Zahlen: $\text{Th}(\mathbb{N}, +, \cdot, 0, 1)$

Unentscheidbar und nicht rekursiv aufzählbar
(letzteres ist Gödels berühmter 1. Unvollständigkeitssatz)

Presburger-Arithmetik: $\text{Th}(\mathbb{N}, +, 0, 1)$

z.B. $\forall x ((x + x = x) \rightarrow x = 0)$

Entscheidbar, ungefähr 2ExpSpace-vollständig

Zu schwach, um wirklich interessante mathematische Probleme auszudrücken, aber wichtige Anwendungen in der Informatik!

Skolem-Arithmetik: $\text{Th}(\mathbb{N}, \cdot, 0, 1)$

Entscheidbar, ungefähr 3ExpSpace-vollständig

Einige wichtige FO-Theorien

Arithmetik der reellen Zahlen: $\text{Th}(\mathbb{R}, +, \cdot, 0, 1)$

$$\text{z. B. } \forall x \forall y \forall z \left(\underbrace{(x = y \cdot y)}_{x = y^2} \wedge \underbrace{(x = z \cdot z)}_{x = z^2} \rightarrow (y = z \vee \underbrace{y = z - (2 \cdot z)}_{y = -z}) \right)$$

Entscheidbar, ungefähr ExpSpace-vollständig

Man **vergrößert** hier also den Zahlenbereich und bekommt dadurch ein **einfacheres** Entscheidungsproblem.

Beachte:

Aus der Unentscheidbarkeit von $\text{Th}(\mathbb{N}, +, \cdot, 0, 1)$ folgt, dass man **keine** FO-Formel $\text{Nat}(x)$ finden kann, die \mathbb{N} in $\text{Th}(\mathbb{R}, +, \cdot, 0, 1)$ **definiert**.

Charakterisierung der Vollständigkeit

Interessanterweise hängt der Begriff der Vollständigkeit sehr eng mit der Definition von Theorien durch Strukturen zusammen.

Zwei τ -Strukturen $\mathfrak{A}, \mathfrak{A}'$ heißen *elementar äquivalent*, wenn für alle Sätze $\varphi \in \text{FO}(\tau)$ gilt: $\mathfrak{A} \models \varphi$ gdw. $\mathfrak{A}' \models \varphi$

Lemma

Sei Γ eine FO-Theorie. Dann sind die folgenden Aussagen äquivalent:

1. Γ ist vollständig
2. $\Gamma = \text{Th}(\mathfrak{A})$ für eine Struktur \mathfrak{A}
3. alle Modelle $\mathfrak{A}, \mathfrak{A}'$ von Γ sind elementar äquivalent

T2.23