„Program = Logic + Control“

Prozedurale/imperative Sprachen:
- Abläufe formulieren
- Computer führt aus

von-Neumann-Maschine

Idee von deklarativen/logischen/funktionalen Programmiersprachen:
- Zusammenhänge formulieren
- Computer erschließt weitere Zusammenhänge

Built-in Prädikate

Programm:
Nutzerdefinierte Prädikate

Interpreter:
Laufzeitsystem, Beweis-Maschine

„Program = Logic + Control“

Idee von deklarativen/logischen/funktionalen Programmiersprachen:
- Zusammenhänge formulieren
- Computer erschließt weitere Zusammenhänge

Deklarative Semantik

Built-in Prädikate

Programm:
Nutzerdefinierte Prädikate

Interpreter:
Laufzeitsystem, Beweis-Maschine

Ablauf der Beweismaschine:
Prozedurale Semantik

Ziele der Software-Technologie:

... Der zweite Aspekt erfordert eine Sprache,
die im Idealfall ,,nahe am Problem'' ist,
so daß die Konzepte der Problemlösung direkt und
schlüssig formuliert werden können. ...
Die Verbindung zwischen der Sprache,
in der wir programmieren/denken,
und den Problemen und Lösungen,
die wir uns vorstellen können, ist sehr eng. ...

Bjarne Stroustrup:
Die C++ Programmiersprache, S. 10
(,,Philosophische Bemerkung'')

Probleme mit Software:

Softwaresysteme gehören zu den komplexesten Gebilden,
die je von Menschenhand geschaffen wurden.
Strukturen und Abläufe in großen Systemen sind
im einzelnen oft nicht mehr überschaubar.
Man kann sie weder im vorhinein, beim Entwurf,
noch im nachhinein, beim Testen, beim Betrieb
und in der Wartung vollständig verstehen.

Denert: Software-Engeneering, S. 4

Probleme mit Software:

Das entscheidende Charakteristikum der industriell einsetzbaren Software ist, daß es für den einzelnen Entwickler sehr schwierig, wenn nicht gar unmöglich ist, alle Feinheiten des Designs zu verstehen.
Einfach ausgedrückt, überschreitet die Komplexität solcher Systeme die Kapazität der menschlichen Intelligenz.

Booch: Objektorientierte Analyse und Design

Prolog basiert auf Prädikatenlogik

Vorbild für „Formalismus“:

– exakt, präzise, (theoretisch) beherrscht

Aufbau:

–Zeichen
–Ausdrücke (rekursive Definition)
–Sätze („Theorie“) **Th**
- syntaktisch bestimmt: **Th** = Abl(**Ax**)
- semantisch bestimmt: **Th** = allgemeingültige Sätze einer Struktur

Nach speziellen formalen Regeln erzeugbare Formeln

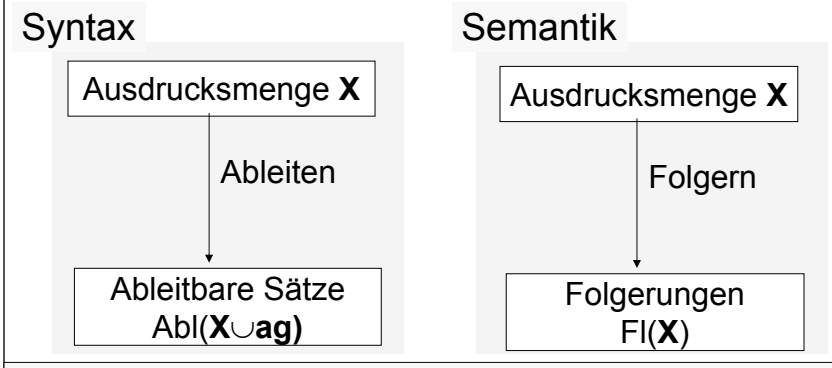
Beweise

„Irreflexive, transitive Relationen sind asymmetrisch“

- Inhaltlicher Beweis:
  - Argumentation
- Formaler (syntaktischer) Beweis:
  - Umformung von Ausdrücken („Kalkül“)
- Theorembeweiser:
  - (Syntaktisches) Verfahren zur Entscheidung, ob ein Ausdruck zu einer Satzmenge (Theorie) gehört:

$H \in \mathbf{Th}$?

Axiomatische Behandlung der Logik



Korrektheit von Abl :	$Abl(\mathbf{X} \cup \mathbf{ag}) \subseteq Fl(\mathbf{X})$
Vollständigkeit von Abl :	$Abl(\mathbf{X} \cup \mathbf{ag}) \supseteq Fl(\mathbf{X})$
Äquivalenz von Abl :	$Abl(\mathbf{X} \cup \mathbf{ag}) = Fl(\mathbf{X})$

Formales Ableiten (Resolutionsregel)

Für Klauseln („Disjunktion von Literalen“)

Voraussetzung:

K1 und K2 unifizierbar mittels Unifikator σ

K =Res(K1,K2,σ) entsteht durch

- „Vereinigung“ von σ(K1) und σ(K2)
- Streichen komplementärer Literale

Formales Ableiten (Resolution)

KA1: $\neg R(x,x)$

KA2: $\neg R(u,y) \vee \neg R(y,z) \vee R(u,z))$

K1: $R(c,f(c))$

K2: $R(f(c),c)$

K3 =Res(K1,K2,σ): $\neg R(w,y) \vee \neg R(y,w)$

mit $\sigma(u)= \sigma(z)= \sigma(x)= w$

K4 = Res(K1,K3,σ): $\neg R(f(c),c)$

mit $\sigma(w)=c \quad \sigma(y)=f(c)$

K5 = Res(K2,K4,σ): □

Enscheidbarkeit in der Logik

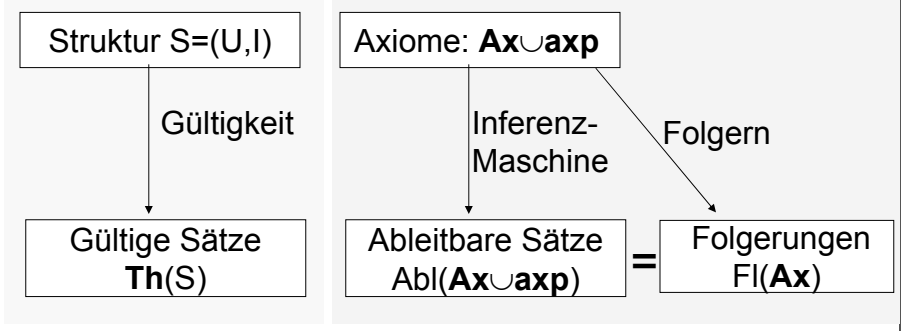
(rein logisch) allgemeingültige Sätze:
ag = Abl(**axp**) = Fl(∅)

H ∈ **ag** ?

- Entscheidbar im AK
- Unentscheidbar im PK1

  (aber aufzählbar, da axiomatisierbar)

Formalisierung einer Domäne

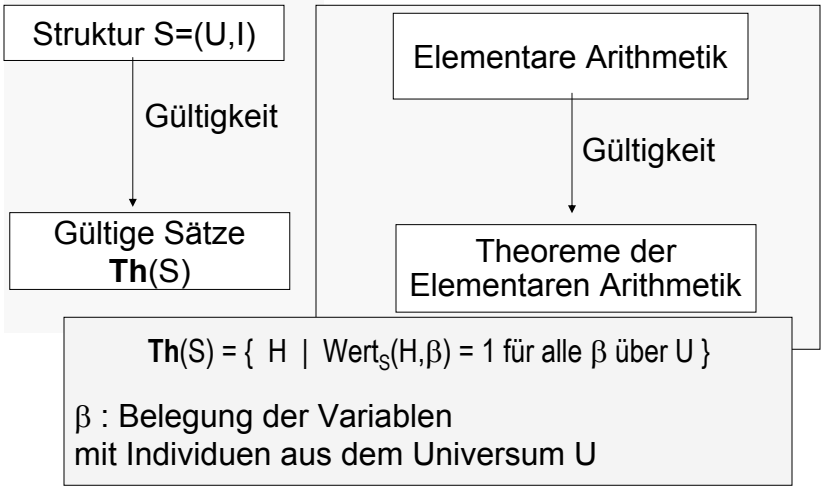


Korrektheit der Formalisierung : Abl(**Ax**∪**axp**) ⊆ **Th**(S)

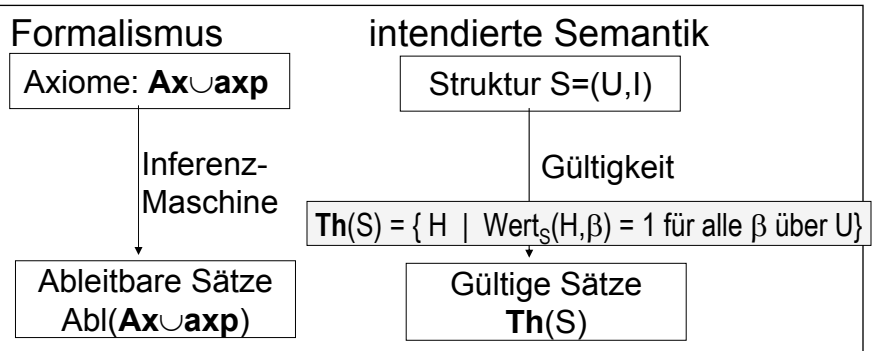
Vollständigkeit der Formalisierung: Abl(**Ax**∪**axp**) ⊇ **Th**(S)

Äquivalenz der Formalisierung : Abl(**Ax**∪**axp**) = **Th**(S)

Beispiel: Formalisierung der Arithmetik



Formalisierung einer Domäne

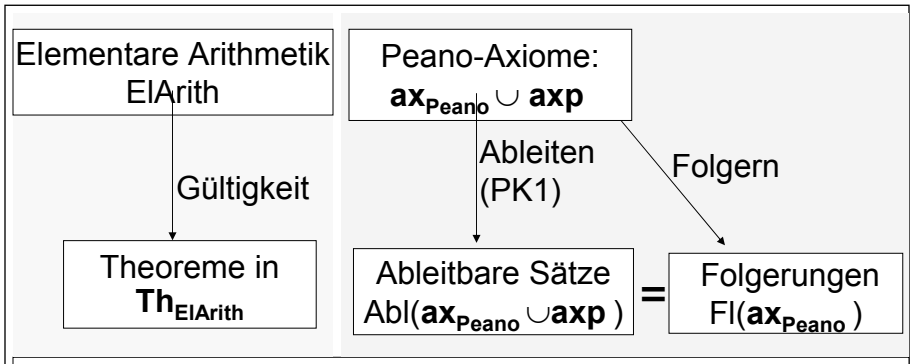


Korrektheit der Formalisierung : Abl(**Ax**∪**axp**) ⊆ **Th**(S)

Vollständigkeit der Formalisierung: Abl(**Ax**∪**axp**) ⊇ **Th**(S)

Äquivalenz der Formalisierung : Abl(**Ax**∪**axp**) = **Th**(S)

Beispiel: Formalisierung der Arithmetik



Korrektheit der Formalisierung :

$$Abl(\mathbf{Ax_{Peano}} \cup \mathbf{axp}) \subseteq \mathbf{Th_{ElArith}}$$

Unvollständigkeit der Formalisierung:

$$Abl(\mathbf{Ax_{Peano}} \cup \mathbf{axp}) \neq \mathbf{Th_{ElArith}}$$