

Formale Modellierung  
Vorlesung 1 vom 16.04.15: Einführung

Christoph Lüth

Universität Bremen

Sommersemester 2015

Organisatorisches

► Veranstalter:

Christoph Lüth  
christoph.lueth@dfki.de  
MZH 3110, Tel. 59830

► Termine:

- Vorlesung: Montag, 16 – 18, MZH 1470
- Übung: Donnerstag, 14 – 16, MZH 5210

► Webseite:

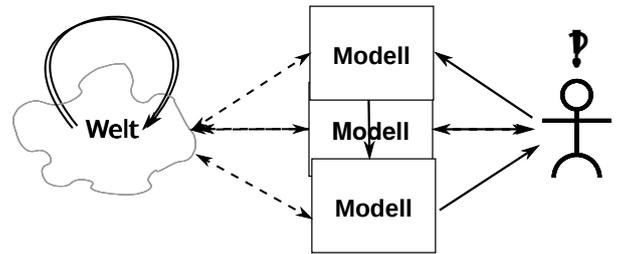
<http://www.informatik.uni-bremen.de/~cx1/lehre/fm.ss15>

Warum formale Modellierung?



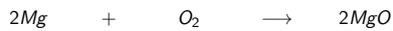
Die Vasa, 10. August 1628

Modellierung — Das Prinzip

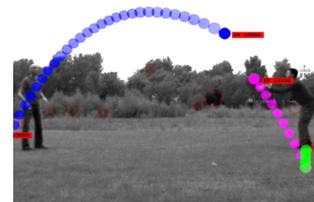


► Grundlegendes Prinzip der Naturwissenschaften

Modellierung — Beispiele

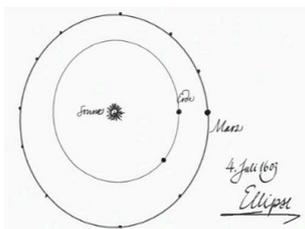


Modellierung — Beispiele



$$\begin{aligned} x(t) &= v_0 t \cos(\beta) \\ y(t) &= v_0 t \sin(\beta) - \frac{g}{2} t^2 \end{aligned}$$

Modellierung — Beispiele



$$\left(\frac{T_1}{T_2}\right)^2 = \left(\frac{a_1}{a_2}\right)^3$$

Arten der Modellierung

► Physikalischen Systeme:

► Modellierung durch kontinuierliche Mathematik (Analysis, DGL)

► Frage: wie modellieren wir Programme und ihr Verhalten?

► Modellierung von Programmen: Berechenbarkeitsbegriff

► Turing-Maschinen, rekursive Funktionen, ...

► Modellierung der Eigenschaften: formale Logik

► Formale Logik ist die Grundlage der Modellierung in der Informatik

## Was ist mit der UML?

- ▶ **Allgemeine** Modellierungssprache für **problemorientierte** Spezifikationen
- ▶ Ziel ist nicht der **Beweis** von Eigenschaften
- ▶ Nur **bestimmte** Aspekte sind formal
- ▶ Als **Grundlage** nicht geeignet

9 [17]

## Lernziele

1. **Modellierung** — Formulierung von Eigenschaften
2. **Beweis** — Formaler Beweis der Eigenschaften
3. **Spezifikation** und **Verifikation** — Eigenschaften von Programmen

10 [17]

## Themen

- ▶ **Formale Logik:**
  - ▶ Aussagenlogik ( $A \wedge B, A \rightarrow B$ ), Prädikatenlogik ( $\forall x.P$ )
  - ▶ Formales Beweisen: natürliches Schließen
  - ▶ Induktion, induktive Datentypen, Rekursion
  - ▶ Berechenbarkeitsmodelle
  - ▶ Die Gödel-Theoreme
- ▶ **Spezifikation und Verifikation:**
  - ▶ Formale Modellierung von Programmen
  - ▶ Temporale Logik
  - ▶ Modellprüfung

11 [17]

## Der Theorembeweiser Isabelle

- ▶ **Interaktiver** Theorembeweiser
- ▶ Entwickelt in **Cambridge** und **München**
- ▶ Est. 1993 (?), ca. 500 Benutzer
- ▶ Andere: PVS, Coq, ACL-2
- ▶ Vielfältig benutzt:
  - ▶ VeriSoft (D) — <http://www.verisoft.de>
  - ▶ L4.verified (AUS) — <http://ertos.nicta.com.au/research/l4.verified/>
  - ▶ SAMS (Bremen) — <http://www.projekt-sams.de>

12 [17]

## Formale Logik

- ▶ Formale (symbolische) Logik: Rechnen mit **Symbolen**
- ▶ **Programme:** Symbolmanipulation
- ▶ **Auswertung:** Beweis
- ▶ **Curry-Howard-Isomorphie:**  
funktionale Programme  $\cong$  konstruktiver Beweis

13 [17]

## Geschichte

- ▶ Gottlob **Frege** (1848– 1942)
  - ▶ 'Begriffsschrift, eine der arithmetischen nachgebildete Formelsprache des reinen Denkens' (1879)
- ▶ Georg **Cantor** (1845– 1918), Bertrand **Russel** (1872– 1970), Ernst **Zermelo** (1871– 1953)
  - ▶ Einfache Mengenlehre: inkonsistent (Russel's Paradox)
  - ▶ Axiomatische Mengenlehre: Zermelo-Fränkel
- ▶ David **Hilbert** (1862– 1943)
  - ▶ Hilbert's Programm: 'mechanisierte' Beweistheorie
- ▶ Kurt **Gödel** (1906– 1978)
  - ▶ Vollständigkeitssatz, Unvollständigkeitssätze

14 [17]

## Grundbegriffe der formalen Logik

- ▶ **Ableitbarkeit**  $\mathcal{Th} \vdash P$ 
  - ▶ Syntaktische Folgerung
- ▶ **Gültigkeit**  $\mathcal{Th} \models P$ 
  - ▶ Semantische Folgerung
- ▶ **Klassische Logik:**  $P \vee \neg P$
- ▶ **Entscheidbarkeit**
  - ▶ Aussagenlogik
- ▶ **Konsistenz:**  $\mathcal{Th} \not\vdash \perp$ 
  - ▶ Nicht alles ableitbar
- ▶ **Vollständigkeit:** jede gültige Aussage ableitbar
  - ▶ Prädikatenlogik erster Stufe

15 [17]

## Unvollständigkeit

- ▶ Gödels 1. **Unvollständigkeitssatz:**
  - ▶ Jede Logik, die Peano-Arithmetik formalisiert, ist entweder **inkonsistent** oder **unvollständig**.
- ▶ Gödels 2. **Unvollständigkeitssatz:**
  - ▶ Jede Logik, die ihre eigene Konsistenz beweist, ist **inkonsistent**.
- ▶ Auswirkungen:
  - ▶ Hilbert's Programm terminiert nicht.
  - ▶ **Programme** nicht vollständig spezifizierbar.
  - ▶ **Spezifikationssprachen** immer **unvollständig** (oder uninteressant).
  - ▶ **Mit anderen Worten:** Es bleibt **spannend**.

16 [17]

## Nächste Woche

- ▶ Aussagenlogik
- ▶ Erstes Übungsblatt

## Formale Modellierung

### Vorlesung 2 vom 20.04.15: Aussagenlogik und natürliches Schließen

Christoph Lüth

Universität Bremen

Sommersemester 2015

16.21.32 2015-07-13

1 [17]

## Heute

- ▶ Einführung in die **formale Logik**
- ▶ **Aussagenlogik**
  - ▶ Beispiel für eine einfache Logik
  - ▶ Guter Ausgangspunkt
- ▶ **Natürliches Schließen**
  - ▶ Wird auch von Isabelle verwendet.
- ▶ Buchempfehlung:  
Dirk van Dalen: **Logic and Structure**. Springer Verlag, 2004.

2 [17]

## Fahrplan

- ▶ **Teil I: Formale Logik**
  - ▶ Einführung
  - ▶ **Aussagenlogik (PL): Syntax und Semantik, Natürliches Schließen**
  - ▶ Konsistenz & Vollständigkeit der Aussagenlogik
  - ▶ Prädikatenlogik (FOL): Syntax und Semantik
  - ▶ Konsistenz & Vollständigkeit von FOL
  - ▶ FOL mit induktiven Datentypen
  - ▶ FOL mit rekursiven Definitionen
  - ▶ Logik höherer Stufe (HOL): Syntax und Eigenschaften
  - ▶ Berechnungsmodelle (Models of Computation)
  - ▶ Die Unvollständigkeitssätze von Gödel
- ▶ **Teil II: Spezifikation und Verifikation**

3 [17]

## Formalisierung von Aussagen

- ▶ Beispielaussagen:
  1. John fuhr weiter und stieß mit einem Fußgänger zusammen.
  2. John stieß mit einem Fußgänger zusammen und fuhr weiter.
  3. Wenn ich das Fenster öffne, haben wir Frischluft.
  4. Wenn wir Frischluft haben, dann ist  $1 + 3 = 4$
  5. Wenn  $1 + 2 = 4$ , dann haben wir Frischluft.
  6. John arbeitet oder ist zu Hause.
  7. Euklid war ein Grieche oder ein Mathematiker.
- ▶ Probleme natürlicher Sprache:
  - ▶ Mehrdeutigkeit
  - ▶ Synonyme
  - ▶ Versteckte (implizite) Annahmen

4 [17]

## Formale Logik

- ▶ Ziel: **Formalisierung** von **Folgerungen** wie
  - ▶ Wenn es regnet, wird die Straße nass.    ▶ Nachts ist es dunkel.
  - ▶ Es regnet.    ▶ Es ist hell.
  - ▶ Also ist die Straße nass.    ▶ Also ist es nicht nachts.
- ▶ Eine **Logik** besteht aus
  - ▶ Einer Sprache  $\mathcal{L}$  von **Formeln** (Aussagen)
  - ▶ Einer **Semantik**, die Formeln eine **Bedeutung** zuordnet
  - ▶ **Schlussregeln** (Folgerungsregeln) auf den Formeln.
- ▶ Damit: **Gültige** ("wahre") Aussagen berechnen.

5 [17]

## Beispiel für eine Logik

- ▶ Sprache  $\mathcal{L} = \{\clubsuit, \spadesuit, \heartsuit, \diamondsuit\}$
- ▶ Schlussregeln:
 

Aus $\diamondsuit$ folgt $\clubsuit$	Aus $\diamondsuit$ folgt $\spadesuit$	Aus $\clubsuit$ und $\spadesuit$ folgt $\heartsuit$	$\diamondsuit$ gilt immer
$\frac{\diamondsuit}{\clubsuit} \alpha$	$\frac{\diamondsuit}{\spadesuit} \beta$	$\frac{\clubsuit \spadesuit}{\heartsuit} \gamma$	$\frac{}{\diamondsuit} \delta$
- ▶ Beispielleitung:  $\heartsuit$

6 [17]

## Aussagenlogik

- ▶ Sprache  $Prop$  gegeben durch:
  1. Variablen (Atome)  $V \subseteq Prop$  (Menge  $V$  gegeben)
  2.  $\perp \in Prop$
  3. Wenn  $\phi, \psi \in Prop$ , dann
    - ▶  $\phi \wedge \psi \in Prop$
    - ▶  $\phi \vee \psi \in Prop$
    - ▶  $\phi \rightarrow \psi \in Prop$
    - ▶  $\phi \leftrightarrow \psi \in Prop$
  4. Wenn  $\phi \in Prop$ , dann  $\neg\phi \in Prop$ .
- ▶ NB. Präzedenzen:  $\neg$  vor  $\wedge$  vor  $\vee$  vor  $\rightarrow$ ,  $\leftrightarrow$

7 [17]

## Wann ist eine Formel gültig?

- ▶ **Semantische Gültigkeit**  $\models P$ 
  - ▶ **Übersetzung** in semantische Domäne
  - ▶ Variablen sind **wahr** oder **falsch**
  - ▶ Operationen verknüpfen diese Werte
- ▶ **Syntaktische Gültigkeit**  $\vdash P$ 
  - ▶ Formale Ableitung
  - ▶ Natürliches Schließen
  - ▶ Sequenzkalkül
  - ▶ Andere (Hilbert-Kalkül, gleichungsbasierte Kalküle, etc.)

8 [17]

## Semantik

- ▶ Domäne:  $\{0, 1\}$  (0 für falsch, 1 für wahr)

Definition (Semantik aussagenlogischer Formeln)

Für Valuation  $v : V \rightarrow \{0, 1\}$  ist  $\llbracket \cdot \rrbracket_v : Prop \rightarrow \{0, 1\}$  definiert als

$$\begin{aligned} \llbracket w \rrbracket_v &= v(w) \quad (\text{mit } w \in V) \\ \llbracket \perp \rrbracket_v &= 0 \\ \llbracket \phi \wedge \psi \rrbracket_v &= \min(\llbracket \phi \rrbracket_v, \llbracket \psi \rrbracket_v) \\ \llbracket \phi \vee \psi \rrbracket_v &= \max(\llbracket \phi \rrbracket_v, \llbracket \psi \rrbracket_v) \\ \llbracket \phi \rightarrow \psi \rrbracket_v &= 0 \iff \llbracket \phi \rrbracket_v = 1 \text{ und } \llbracket \psi \rrbracket_v = 0 \\ \llbracket \phi \leftrightarrow \psi \rrbracket_v &= 1 \iff \llbracket \phi \rrbracket_v = \llbracket \psi \rrbracket_v \\ \llbracket \neg \phi \rrbracket_v &= 1 - \llbracket \phi \rrbracket_v \end{aligned}$$

9 [17]

## Semantische Gültigkeit und Folgerung

- ▶ Semantische Gültigkeit:  $\models \phi$

$$\models \phi \text{ gdw. } \llbracket \phi \rrbracket_v = 1 \text{ für alle } v$$

- ▶ Semantische Folgerung: sei  $\Gamma \subseteq Prop$ , dann

$$\Gamma \models \psi \text{ gdw. } \llbracket \psi \rrbracket_v = 1 \text{ wenn } \llbracket \phi \rrbracket_v = 1 \text{ für alle } \phi \in \Gamma$$

10 [17]

## Beweisen mit semantischer Folgerung

- ▶ Die Wahrheitstabellenmethode:

- ▶ Berechne  $\llbracket \phi \rrbracket_v$  für alle Möglichkeiten für  $v$

- ▶ Beispiel:  $\models (\phi \rightarrow \psi) \leftrightarrow (\neg\psi \rightarrow \neg\phi)$

$\phi$	$\psi$	$\phi \rightarrow \psi$	$\neg\psi$	$\neg\phi$	$\neg\psi \rightarrow \neg\phi$	$(\phi \rightarrow \psi) \leftrightarrow (\neg\psi \rightarrow \neg\phi)$
0	0	1	1	1	1	1
0	1	1	0	1	1	1
1	0	0	1	0	0	1
1	1	1	0	0	1	1

- ▶ Problem: Aufwand exponentiell  $2^a$  zur Anzahl  $a$  der Atome

- ▶ Vorteil: Konstruktion von Gegenbeispielen

11 [17]

## Syntaktische Gültigkeit: Natürliches Schließen

- ▶ Sprache  $\mathcal{L} = \{\clubsuit, \spadesuit, \heartsuit, \diamondsuit\}$

- ▶ Schlussregeln:

$$\begin{array}{c} \diamondsuit \\ \vdots \\ \heartsuit \\ \hline \heartsuit \end{array} \delta'$$

- ▶ Beispielableitung:  $\heartsuit$

12 [17]

## Natürliches Schließen (ND) für Aussagenlogik

- ▶ Vorgehensweise:

1. Erst Kalkül nur für  $\wedge, \rightarrow, \perp$
2. Dann Erweiterung auf alle Konnektive.

- ▶ Für jedes Konnektiv: Einführungs- und Eliminationsregel

- ▶ NB: konstruktiver Inhalt der meisten Regeln

13 [17]

## Natürliches Schließen — Die Regeln

$$\begin{array}{c} \frac{\phi \quad \psi}{\phi \wedge \psi} \wedge I \\ \frac{[\phi] \quad \vdots}{\psi} \rightarrow I \\ \frac{\perp}{\phi} \perp \end{array} \quad \begin{array}{c} \frac{\phi \wedge \psi}{\phi} \wedge E_L \quad \frac{\phi \wedge \psi}{\psi} \wedge E_R \\ \frac{\phi \quad \phi \rightarrow \psi}{\psi} \rightarrow E \\ \frac{[\phi \rightarrow \perp] \quad \vdots}{\perp} \text{raa} \end{array}$$

14 [17]

## Die fehlenden Konnektive

- ▶ Einführung als Abkürzung:

$$\begin{aligned} \neg\phi &\stackrel{\text{def}}{=} \phi \rightarrow \perp \\ \phi \vee \psi &\stackrel{\text{def}}{=} \neg(\neg\phi \wedge \neg\psi) \\ \phi \leftrightarrow \psi &\stackrel{\text{def}}{=} (\phi \rightarrow \psi) \wedge (\psi \rightarrow \phi) \end{aligned}$$

- ▶ Ableitungsregeln als Theoreme.

15 [17]

## Die fehlenden Schlussregeln

$$\begin{array}{c} \frac{[\phi] \quad \vdots}{\perp} \neg I \\ \frac{\phi \quad \neg\phi}{\perp} \neg E \\ \frac{\phi \quad \psi}{\phi \vee \psi} \vee I_L \quad \frac{\psi}{\phi \vee \psi} \vee I_R \\ \frac{\phi \vee \psi \quad \sigma}{\sigma} \vee E \\ \frac{\phi \rightarrow \psi \quad \psi \rightarrow \phi}{\phi \leftrightarrow \psi} \leftrightarrow I \\ \frac{\phi \quad \phi \leftrightarrow \psi}{\psi} \leftrightarrow E_L \quad \frac{\psi \quad \phi \leftrightarrow \psi}{\phi} \leftrightarrow E_R \end{array}$$

16 [17]

## Zusammenfassung

- ▶ Formale Logik **formalisiert** das (natürlichsprachliche) Schlußfolgern
- ▶ **Logik**: Formeln, Semantik, Schlußregeln (Kalkül)
- ▶ **Aussagenlogik**: Aussagen mit  $\wedge, \longrightarrow, \perp$ 
  - ▶  $\neg, \vee, \longleftrightarrow$  als abgeleitete Operatoren
- ▶ **Semantik** von Aussagenlogik  $\llbracket \cdot \rrbracket_v : Prop \rightarrow \{0, 1\}$
- ▶ Natürliches **Schließen**: intuitiver Kalkül
- ▶ Nächste Woche:
  - ▶ Konsistenz und Vollständigkeit von Aussagenlogik

# Formale Modellierung

## Vorlesung 3 vom 27.04.15: Konsistenz und Vollständigkeit der Aussagenlogik

Christoph Lüth

Universität Bremen

Sommersemester 2015

## Fahrplan

- ▶ Teil I: Formale Logik
  - ▶ Einführung
  - ▶ Aussagenlogik (PL): Syntax und Semantik, Natürliches Schließen
  - ▶ Konsistenz & Vollständigkeit der Aussagenlogik
  - ▶ Prädikatenlogik (FOL): Syntax und Semantik
  - ▶ Konsistenz & Vollständigkeit von FOL
  - ▶ FOL mit induktiven Datentypen
  - ▶ FOL mit rekursiven Definitionen
  - ▶ Logik höherer Stufe (HOL): Syntax und Eigenschaften
  - ▶ Berechnungsmodelle (Models of Computation)
  - ▶ Die Unvollständigkeitssätze von Gödel
- ▶ Teil II: Spezifikation und Verifikation

## Das Tagesmenü

- ▶ Eigenschaften der Aussagenlogik (PL)
- ▶  $\Gamma \vdash \phi$  vs.  $\Gamma \models \phi$ :
  - ▶ Korrektheit
  - ▶ Konsistenz
  - ▶ Vollständigkeit

## Eigenschaften der Aussagenlogik

- ▶ Prop bildet eine Boolesche Algebra:

$$\begin{aligned} \models (\phi \vee \psi) \vee \sigma &\leftrightarrow \phi \vee (\psi \vee \sigma) \\ \models (\phi \wedge \psi) \wedge \sigma &\leftrightarrow \phi \wedge (\psi \wedge \sigma) \\ \models \phi \vee \psi &\leftrightarrow \psi \vee \phi \\ \models \phi \wedge \psi &\leftrightarrow \psi \wedge \phi \\ \models \phi \vee (\psi \wedge \sigma) &\leftrightarrow (\phi \vee \psi) \wedge (\phi \vee \sigma) \\ \models \phi \wedge (\psi \vee \sigma) &\leftrightarrow (\phi \wedge \psi) \vee (\phi \wedge \sigma) \\ \models \neg(\phi \vee \psi) &\leftrightarrow \neg\phi \wedge \neg\psi \\ \models \neg(\phi \wedge \psi) &\leftrightarrow \neg\phi \vee \neg\psi \\ \models \phi \vee \phi &\leftrightarrow \phi \\ \models \phi \wedge \phi &\leftrightarrow \phi \\ \models \neg\neg\phi &\leftrightarrow \phi \end{aligned}$$

## Eigenschaften der Aussagenlogik

- ▶ Rechnen in Prop:
  - ▶ Substitutivität: wenn  $\models \phi_1 \leftrightarrow \phi_2$ , dann  $\models \psi[\phi_1] \leftrightarrow \psi[\phi_2]$  für Atom  $p$ .
  - ▶ Sei  $\phi \approx \psi$  gdw.  $\models \phi \leftrightarrow \psi$ , dann ist  $\approx$  eine Äquivalenzrelation
- ▶ Damit: algebraisches Umformen als Beweisprinzip
  - ▶ Beispiele:  $\models (\phi \rightarrow (\psi \rightarrow \sigma)) \leftrightarrow (\phi \wedge \psi \rightarrow \sigma)$   
 $\models \phi \rightarrow \psi \rightarrow \phi$
- ▶ Anwendung: konjunktive und disjunktive Normalformen (CNF/DNF)

## Eigenschaften der Aussagenlogik

- ▶ Operatoren durch andere definierbar:

$$\begin{aligned} \models (\phi \leftrightarrow \psi) &\leftrightarrow (\phi \rightarrow \psi) \wedge (\psi \rightarrow \phi) \\ \models (\phi \rightarrow \psi) &\leftrightarrow (\neg\phi \vee \psi) \\ \models \phi \vee \psi &\leftrightarrow \neg(\neg\phi \wedge \neg\psi) \\ \models \phi \wedge \psi &\leftrightarrow \neg(\neg\phi \vee \neg\psi) \\ \models \neg\phi &\leftrightarrow (\phi \rightarrow \perp) \\ \models \perp &\leftrightarrow (\phi \wedge \neg\phi) \\ \models \top &\leftrightarrow (\phi \vee \neg\phi) \end{aligned}$$

- ▶  $(\wedge, \neg)$  und  $(\vee, \perp)$  sind ausreichend (functional complete)
- ▶ Ein Operator reicht:  $A \mid B$  (Sheffer-Strich),  $A \downarrow B$  (weder-noch)

## Korrektheit (Soundness)

- ▶  $\Gamma \vdash \phi$ : Ableitbarkeit
- ▶  $\Gamma \models \phi$ : semantische 'Wahrheit'
- ▶ Ist alles wahr, was wir ableiten können? (Korrektheit)
- ▶ Ist alles ableitbar, was wahr ist? (Vollständigkeit)

### Theorem 1 (Korrektheit von ND in der Aussagenlogik)

Wenn  $\Gamma \vdash \phi$ , dann  $\Gamma \models \phi$

Beweis: Induktion über der Ableitung  $\Gamma \vdash \phi$

- ▶ Nützliches Korollar:  $\Gamma \not\models \phi$  dann  $\Gamma \not\vdash \phi$

## Konsistenz

- ▶ Nur konsistente Logiken (Mengen von Aussagen) sind sinnvoll.

### Definition 2 (Konsistenz)

Menge  $\Gamma$  von Aussagen konsistent gdw.  $\Gamma \not\vdash \perp$

### Lemma 3 (Charakterisierung von Konsistenz)

Folgende Aussagen sind äquivalent:

- $\Gamma$  konsistent
- Es gibt kein  $\phi$  so dass  $\Gamma \vdash \phi$  und  $\Gamma \vdash \neg\phi$
- Es gibt ein  $\phi$  so dass  $\Gamma \not\vdash \phi$
- $\Gamma$  inkonsistent ( $\Gamma \vdash \perp$ )
- Es gibt ein  $\phi$  so dass  $\Gamma \vdash \phi$  und  $\Gamma \vdash \neg\phi$
- Für alle  $\phi$ ,  $\Gamma \vdash \phi$

## Maximale Konsistenz

- ▶ Wenn es  $v$  gibt so dass  $\llbracket \psi \rrbracket_v = 1$  für  $\psi \in \Gamma$ , dann  $\Gamma$  konsistent.

### Definition 4 (Maximale Konsistenz)

$\Gamma$  **maximal konsistent** gdw.

- (i)  $\Gamma$  konsistent, und
- (ii) wenn  $\Gamma \subseteq \Gamma'$  und  $\Gamma'$  konsistent, dann  $\Gamma = \Gamma'$

### Lemma 5 (Konstruktion maximal konsistenter Mengen)

Für jedes konsistente  $\Gamma$  gibt es **maximal konsistentes**  $\Gamma^*$  mit  $\Gamma \subseteq \Gamma^*$

9 [12]

## Eigenschaften maximal konsistenter Mengen

- ▶ Wenn  $\Gamma \cup \{\phi\}$  inkonsistent, dann  $\Gamma \vdash \neg\phi$  (Beweis:  $\neg I$ )
- ▶ Wenn  $\Gamma \cup \{\neg\phi\}$  inkonsistent, dann  $\Gamma \vdash \phi$  (Beweis:  $raa$ )

### Lemma 6

Wenn  $\Gamma$  maximal konsistent, dann **geschlossen** unter Ableitbarkeit:  
 $\Gamma \vdash \phi$  dann  $\phi \in \Gamma$ .

- ▶ Wenn  $\Gamma$  maximal konsistent ist, dann:
  - (i) entweder  $\phi \in \Gamma$  oder  $\neg\phi \in \Gamma$
  - (ii)  $\phi \wedge \psi \in \Gamma$  gdw.  $\phi, \psi \in \Gamma$
  - (iii)  $\phi \rightarrow \psi \in \Gamma$  gdw. (wenn  $\phi \in \Gamma$  dann  $\psi \in \Gamma$ )

10 [12]

## Vollständigkeit

### Lemma 7

Wenn  $\Gamma$  konsistent, dann gibt es  $v$  so dass  $\llbracket \phi \rrbracket_v = 1$  für  $\phi \in \Gamma$ .

Damit:

- ▶ Wenn  $\Gamma \not\vdash \phi$  dann gibt es  $v$  so dass  $\llbracket \psi \rrbracket_v = 1$  für  $\psi \in \Gamma$ ,  $\llbracket \phi \rrbracket_v = 0$ .
- ▶ Wenn  $\Gamma \not\vdash \phi$  dann  $\Gamma \not\models \phi$ .

### Theorem 8 (Vollständigkeit von ND in der Aussagenlogik)

Wenn  $\Gamma \models \phi$ , dann  $\Gamma \vdash \phi$

- ▶ Aus Entscheidbarkeit von  $\models$  folgt Entscheidbarkeit von  $\vdash$

11 [12]

## Zusammenfassung

- ▶ Aussagenlogik ist eine **Boolesche Algebra**.
  - ▶ Äquivalenzumformung als Beweisprinzip
- ▶ Aussagenlogik und natürliches Schließen sind **korrekt** und **vollständig**.
  - ▶ Beweis der Vollständigkeit: maximale Konsistenz
  - ▶ Konstruktion des Herbrand-Modells, Aufzählung aller (wahren, ableitbaren) Aussagen
- ▶ Aussagenlogik ist **entscheidbar**: für  $\Gamma$  und  $\phi$ ,  $\Gamma \vdash \phi$  oder  $\Gamma \not\vdash \phi$ .
- ▶ Nächste VL: Prädikatenlogik

12 [12]

# Formale Modellierung

## Vorlesung 4 vom 04.05.15: Prädikatenlogik erster Stufe

Christoph Lüth

Universität Bremen

Sommersemester 2015

16.21.34 2015-07-13

1 [16]

## Fahrplan

- ▶ **Teil I: Formale Logik**
  - ▶ Einführung
  - ▶ Aussagenlogik (PL): Syntax und Semantik, Natürliches Schließen
  - ▶ Konsistenz & Vollständigkeit der Aussagenlogik
  - ▶ **Prädikatenlogik (FOL): Syntax und Semantik**
  - ▶ Konsistenz & Vollständigkeit von FOL
  - ▶ FOL mit induktiven Datentypen
  - ▶ FOL mit rekursiven Definitionen
  - ▶ Logik höherer Stufe (HOL): Syntax und Eigenschaften
  - ▶ Berechnungsmodelle (Models of Computation)
  - ▶ Die Unvollständigkeitssätze von Gödel
- ▶ **Teil II: Spezifikation und Verifikation**

2 [16]

## Das Tagesmenü

- ▶ Von Aussagenlogik zur Prädikatenlogik
- ▶ Logik mit **Quantoren**
- ▶ **Semantik** der Prädikatenlogik
- ▶ **Natürliches Schließen** mit Quantoren

3 [16]

## Eine Beispielspezifikation

### Das Flugbuchungssystem

Das **Flugbuchungssystem** soll eine Menge von Flügen verwalten, Anfragen beantworten und Buchungen vornehmen.

Ein **Flug** hat einen Startflughafen und ein Zielflughafen (durch ihr IATA-Kürzel repräsentiert), eine eindeutige Kennung, einen Starttermin, eine Ankunftszeit, sowie eine Anzahl von verfügbaren Plätzen.

Eine **Flugbuchung** für einen durch die Flugnummer und Starttermin identifizierten Flug soll eine Anzahl von Plätzen auf diesem Flug reservieren. Sind die verfügbaren Plätze für einen Flug erschöpft, können keine weiteren Buchungen vorgenommen werden.

Eine **Anfrage** besteht aus den Daten (Start, Ziel, Datum) für einen Flug, und liefert die Anzahl freier Plätze auf diesem Flug zurück.

4 [16]

## Beschränkungen der Aussagenlogik

- ▶ **Beschränkung** der Aussagenlogik:
  - ▶ Die Menge unserer Atome ist **unstrukturiert** und **flach**.
  - ▶ Wir können nicht zwischen **Logik** (Meta-Ebene) und **Objekt** unterscheiden.
  - ▶ Wir können keine **strukturellen** Eigenschaften beschreiben.
  - ▶ Wir können keine Aussagen über **Existenz** von Objekten machen.
- ▶ **Ziel**: Formalisierung von Aussagen wie
  - ▶ "Ein Flug hat eine **eindeutige** Kennung."
  - ▶ "Alle Menschen sind sterblich. Sokrates ist ein Mensch. Also ist Sokrates sterblich."
  - ▶ "Alle Zahlen sind ein Produkt von Primfaktoren."

5 [16]

## Prädikatenlogik: Erweiterung der Sprache

- ▶ **Terme** beschreiben die zu formalisierenden Objekte.
- ▶ **Formeln** sind logische Aussagen.
- ▶ Eine **Signatur**  $\Sigma$  beschreibt Prädikate und Funktionen:
  - ▶ **Prädikatsymbole**:  $P_1, \dots, P_n$ ,  $\doteq$  mit **Arität**  $ar(P_i) \in \mathbb{N}$ ,  $ar(\doteq) = 2$
  - ▶ **Funktionsymbole**:  $f_1, \dots, f_m$  mit **Arität**  $ar(f_i) \in \mathbb{N}$
- ▶ Menge  $X$  von **Variablen** (abzählbar viele)
- ▶ **Konnektive**:  $\wedge, \rightarrow, \perp, \forall, \exists$ , abgeleitet:  $\vee, \leftrightarrow, \neg, \leftarrow, \rightarrow, \exists$
- ▶ Die **Trennung** zwischen **Termen** und **Formeln** ist der wesentliche Abstraktionsschritt in der Prädikatenlogik.

6 [16]

## Terme

- ▶ Menge  $Term_\Sigma$  der **Terme** (zur Signatur  $\Sigma$ ) gegeben durch:
  - ▶ Variablen:  $X \subseteq Term_\Sigma$
  - ▶ Funktionssymbol  $f \in \Sigma$  mit  $ar(f) = n$  und  $t_1, \dots, t_n \in Term_\Sigma$ , dann  $f(t_1, \dots, t_n) \in Term_\Sigma$
  - ▶ Sonderfall:  $n = 0$ , dann ist  $f$  eine **Konstante**,  $f \in Term_\Sigma$

7 [16]

## Formeln

- ▶ Menge  $Form_\Sigma$  der **Formeln** (zur Signatur  $\Sigma$ ) gegeben durch:
  - ▶  $\perp \in Form_\Sigma$
  - ▶ Wenn  $\phi \in Form_\Sigma$ , dann  $\neg\phi \in Form_\Sigma$
  - ▶ Wenn  $\phi, \psi \in Form_\Sigma$ , dann  $\phi \wedge \psi \in Form_\Sigma$ ,  $\phi \vee \psi \in Form_\Sigma$ ,  
 $\phi \rightarrow \psi \in Form_\Sigma$ ,  $\phi \leftrightarrow \psi \in Form_\Sigma$
  - ▶ Wenn  $\phi \in Form_\Sigma$ ,  $x \in X$ , dann  $\forall x.\phi \in Form_\Sigma$ ,  $\exists x.\phi \in Form_\Sigma$
  - ▶ Prädikatsymbol  $p \in \Sigma$  mit  $ar(p) = m$  und  $t_1, \dots, t_m \in Term_\Sigma$ , dann  $p(t_1, \dots, t_m) \in Form_\Sigma$ 
    - ▶ Sonderfall:  $t_1, t_2 \in Term_\Sigma$ , dann  $t_1 \doteq t_2 \in Form_\Sigma$

8 [16]

## Freie und gebundene Variable

### Definition (Freie und gebundene Variablen)

Variablen in  $t \in \text{Term}, p \in \text{Form}$  sind **frei**, **gebunden**, oder **bindend**:

- (i)  $x$  **bindend** in  $\forall x.\phi, \exists x.\psi$
- (ii) Für  $\forall x.\phi$  und  $\exists x.\phi$  ist  $x$  in Teilformel  $\phi$  **gebunden**
- (iii) Ansonsten ist  $x$  **frei**

- ▶  $FV(\phi)$ : Menge der freien Variablen in  $\phi$
- ▶ Beispiel:  

$$(q(x) \vee \exists x.\forall y.p(f(x), z) \wedge q(a)) \vee \forall r(x, z, g(x))$$
- ▶ Formel (Term)  $s$  **geschlossen**, wenn  $FV(s) = \emptyset$
- ▶ **Abschluss** einer Formel:  $Cl(\phi) = \forall z_1 \dots z_k.\phi$  für  $FV(\phi) = \{z_1, \dots, z_k\}$

9 [16]

## Semantik: Strukturen

### Definition (Struktur $\mathfrak{A}$ zur Signatur $\Sigma$ )

$\mathfrak{A} = (A, f, P)$  mit

- (i)  $A$  nicht-leere Menge (**Universum**)
- (ii) für  $f \in \Sigma$  mit  $ar(f) = n$ ,  $n$ -stellige Funktion  $f_{\mathfrak{A}} : A^n \rightarrow A$
- (iii) für  $P \in \Sigma$  mit  $ar(P) = n$ ,  $n$ -stellige Relation  $P_{\mathfrak{A}} \subseteq A^n$

- ▶ Für  $a \in A$ , Konstante  $\bar{a} \in \text{Term}_{\Sigma}$
- ▶ Damit Auswertung von **geschlossenen** Termen:  $\llbracket \cdot \rrbracket_{\mathfrak{A}} : \text{Term}_{\Sigma} \rightarrow A$

$$\begin{aligned} \llbracket \bar{a} \rrbracket_{\mathfrak{A}} &= a \\ \llbracket f(t_1, \dots, t_n) \rrbracket_{\mathfrak{A}} &= f_{\mathfrak{A}}(\llbracket t_1 \rrbracket_{\mathfrak{A}}, \dots, \llbracket t_n \rrbracket_{\mathfrak{A}}) \end{aligned}$$

10 [16]

## Semantische Gültigkeit

- ▶ Auswertung von **Formeln**:  $\llbracket \cdot \rrbracket_{\mathfrak{A}} : \text{Form}_{\Sigma} \rightarrow \{0, 1\}$

$$\begin{aligned} \llbracket \perp \rrbracket_{\mathfrak{A}} &= 0 \\ \llbracket \phi \wedge \psi \rrbracket_{\mathfrak{A}} &= \min(\llbracket \phi \rrbracket_{\mathfrak{A}}, \llbracket \psi \rrbracket_{\mathfrak{A}}) & \llbracket \neg \phi \rrbracket_{\mathfrak{A}} &= 1 - \llbracket \phi \rrbracket_{\mathfrak{A}} \\ \llbracket \phi \vee \psi \rrbracket_{\mathfrak{A}} &= \max(\llbracket \phi \rrbracket_{\mathfrak{A}}, \llbracket \psi \rrbracket_{\mathfrak{A}}) \\ \llbracket \phi \rightarrow \psi \rrbracket_{\mathfrak{A}} &= \max(1 - \llbracket \phi \rrbracket_{\mathfrak{A}}, \llbracket \psi \rrbracket_{\mathfrak{A}}) \\ \llbracket \phi \leftrightarrow \psi \rrbracket_{\mathfrak{A}} &= 1 - |\llbracket \phi \rrbracket_{\mathfrak{A}} - \llbracket \psi \rrbracket_{\mathfrak{A}}| \end{aligned}$$

$$\begin{aligned} \llbracket P(t_1, \dots, t_n) \rrbracket_{\mathfrak{A}} &= \begin{cases} 1 & \langle \llbracket t_1 \rrbracket_{\mathfrak{A}}, \dots, \llbracket t_n \rrbracket_{\mathfrak{A}} \rangle \in P_{\mathfrak{A}} \\ 0 & \text{sonst} \end{cases} \\ \llbracket t_1 \doteq t_2 \rrbracket_{\mathfrak{A}} &= \begin{cases} 1 & \llbracket t_1 \rrbracket_{\mathfrak{A}} = \llbracket t_2 \rrbracket_{\mathfrak{A}} \\ 0 & \text{sonst} \end{cases} \\ \llbracket \forall x.\phi \rrbracket_{\mathfrak{A}} &= \min(\{\llbracket \phi \rrbracket_{\mathfrak{A}} \mid a \in A\}) \\ \llbracket \exists x.\phi \rrbracket_{\mathfrak{A}} &= \max(\{\llbracket \phi \rrbracket_{\mathfrak{A}} \mid a \in A\}) \end{aligned}$$

- ▶ Damit **semantische Gültigkeit (Wahrheit)**:  
 $\mathfrak{A} \models \phi$  gdw.  $\llbracket Cl(\phi) \rrbracket_{\mathfrak{A}} = 1$ ,  $\models \phi$  gdw.  $\mathfrak{A} \models \phi$  für alle  $\mathfrak{A}$

11 [16]

## Syntaktische Gültigkeit: Natürliches Schließen

- ▶ Die alten Regeln blieben ( $\rightarrow I, \rightarrow E, \wedge I, \wedge E, \vee I, \vee E, \neg I, \neg E, \text{raa}, \perp, \dots$ )

- ▶ Mutatis mutandis:  $\text{Form}_{\Sigma}$  statt  $\text{Prop}$

- ▶ Dazu benötigen wir Regeln für die Quantoren.

- ▶ Zu behandelnde **Probleme**:

- ▶ Substitution
- ▶ Bindung

12 [16]

## Substitution

- ▶  $t[x]^{[s]}$  ist **Ersetzung** von  $x$  durch  $s$  in  $t$
- ▶ Definiert durch strukturelle **Induktion**:

$$\begin{aligned} y[x]^{[s]} &\stackrel{\text{def}}{=} \begin{cases} s & x = y \\ y & x \neq y \end{cases} \\ f(t_1, \dots, t_n)[x]^{[s]} &\stackrel{\text{def}}{=} f(t_1[x]^{[s]}, \dots, t_n[x]^{[s]}) \\ \perp[x]^{[s]} &\stackrel{\text{def}}{=} \perp \\ (\phi \wedge \psi)[x]^{[s]} &\stackrel{\text{def}}{=} \phi[x]^{[s]} \wedge \psi[x]^{[s]} \\ (\phi \rightarrow \psi)[x]^{[s]} &\stackrel{\text{def}}{=} \phi[x]^{[s]} \rightarrow \psi[x]^{[s]} \\ P(t_1, \dots, t_n)[x]^{[s]} &\stackrel{\text{def}}{=} P(t_1[x]^{[s]}, \dots, t_n[x]^{[s]}) \\ (\forall y.\phi)[x]^{[s]} &\stackrel{\text{def}}{=} \begin{cases} \forall y.\phi & x = y \\ \forall y.(\phi[x]^{[s]}) & x \neq y, y \notin FV(s) \\ \forall z.((\phi[y]^{[z]})[x]^{[s]}) & x \neq y, y \in FV(s) \\ & \text{mit } z \notin FV(s) \cup FV(\phi) \\ & \text{(z frisch)} \end{cases} \end{aligned}$$

13 [16]

## Natürliches Schließen mit Quantoren

$$\frac{\phi}{\forall x.\phi} \forall I \quad (*) \qquad \frac{\forall x.\phi}{\phi[x]^{[t]}} \forall E \quad (\dagger)$$

- ▶ (\*) **Eigenvariablenbedingung**:  
 $x$  nicht **frei** in offenen Vorbedingungen von  $\phi$  ( $x$  beliebig)

- ▶ (\dagger) Ggf. **Umbenennung** durch Substitution

- ▶ **Gegenbeispiele** für verletzte Seitenbedingungen

14 [16]

## Der Existenzquantor

$$\exists x.\phi \stackrel{\text{def}}{=} \neg \forall x.\neg \phi$$

$$\frac{\phi[x]^{[t]}}{\exists x.\phi} \exists I \quad (\dagger) \qquad \frac{\exists x.\phi \quad \psi}{\psi} \exists E \quad (*)$$

- ▶ (\*) **Eigenvariablenbedingung**:  
 $x$  nicht frei in  $\psi$ , oder einer offenen Vorbedingung außer  $\phi$
- ▶ (\dagger) Ggf. **Umbenennung** durch Substitution

15 [16]

## Zusammenfassung

- ▶ **Prädikatenlogik**: Erweiterung der Aussagenlogik um

- ▶ Konstanten- und Prädikatensymbole
- ▶ Gleichheit
- ▶ Quantoren

- ▶ Semantik der Prädikatenlogik: **Strukturen**

- ▶ Bilden Operationen und Prädikate der Logik ab

- ▶ Das **natürliche Schließen** mit Quantoren

- ▶ **Variablenbindungen** — Umbenennungen bei Substitution
- ▶ **Eigenvariablenbedingung**

- ▶ Das nächste Mal: FOL at work, FOL in Isabelle

- ▶ Nächste VL: **Vollständigkeit**

16 [16]

Formale Modellierung  
Vorlesung 5 vom 18.05.15: Eigenschaften der Prädikatenlogik erster Stufe

Christoph Lüth

Universität Bremen

Sommersemester 2015

## Organisatorisches

- Die Übung am Donnerstag, 21.05.2015 **fällt aus!**

## Fahrplan

- Teil I: Formale Logik
  - Einführung
  - Aussagenlogik (PL): Syntax und Semantik, Natürliches Schließen
  - Konsistenz & Vollständigkeit der Aussagenlogik
  - Prädikatenlogik (FOL): Syntax und Semantik
  - Konsistenz & Vollständigkeit von FOL
  - FOL mit induktiven Datentypen
  - FOL mit rekursiven Definitionen
  - Logik höherer Stufe (HOL): Syntax und Eigenschaften
  - Berechnungsmodelle (Models of Computation)
  - Die Unvollständigkeitssätze von Gödel
- Teil II: Spezifikation und Verifikation

## Das Tagesmenü

- Wiederholung: natürliches Schließen mit FOL
- Regeln für die Gleichheit
- Beispiele: Graphen, natürliche Zahlen
- Vollständigkeit von FOL
- Unentscheidbarkeit von FOL

## Natürliches Schließen mit Quantoren

$$\frac{\phi}{\forall x.\phi} \forall I \quad (*) \qquad \frac{\forall x.\phi}{\phi[x/t]} \forall E \quad (\dagger)$$

- (\*) **Eigenvariablenbedingung:**  
x nicht frei in offenen Vorbedingungen von  $\phi$  (x beliebig)
- (†) Ggf. Umbenennung durch Substitution
- Gegenbeispiele für verletzte Seitenbedingungen

## Der Existenzquantor

$$\exists x.\phi \stackrel{def}{=} \neg\forall x.\neg\phi$$

$$\frac{\phi[x/t]}{\exists x.\phi} \exists I \quad (\dagger) \qquad \frac{\begin{array}{c} [\phi] \\ \vdots \\ \psi \end{array}}{\exists x.\phi} \exists E \quad (*)$$

- (\*) **Eigenvariablenbedingung:**  
x nicht frei in  $\psi$ , oder einer offeneren Vorbedingung außer  $\phi$
- (†) Ggf. Umbenennung durch Substitution

## Regeln für die Gleichheit

- Reflexivität, Symmetrie, Transitivität:

$$\frac{}{x \doteq x} \text{ refl} \qquad \frac{x \doteq y}{y \doteq x} \text{ sym} \qquad \frac{x \doteq y \quad y \doteq z}{x \doteq z} \text{ trans}$$

- Kongruenz:

$$\frac{x_1 \doteq y_1, \dots, x_n \doteq y_n}{f(x_1, \dots, x_n) \doteq f(y_1, \dots, y_n)} \text{ cong}$$

- Substitutivität:

$$\frac{x_1 \doteq y_1, \dots, x_m \doteq y_m \quad P(x_1, \dots, x_m)}{P(y_1, \dots, y_m)} \text{ subst}$$

## Wiederholung: Konsistenz und Vollständigkeit

- Korrektheit: wenn  $\Gamma \vdash \phi$  dann  $\Gamma \models \phi$ 
  - Beweis: Induktion über Struktur der Ableitung
- Konsistenz: wenn  $\Gamma \models \phi$  dann  $\Gamma \vdash \phi$ 
  - Beweis: Konstruktion der maximal konsistenten Theorie
  - Wenn  $\Gamma$  konsistent, gibt es Valuation die  $\Gamma$  wahr macht.
- Frage: Korrektheit und Konsistenz für Prädikatenlogik?

## Korrektheit des natürlichen Schließens

### Lemma 1 (Korrektheit von ND)

Wenn  $\Gamma \vdash \phi$ , dann  $\Gamma \models \phi$

Beweis: **Induktion** über der Ableitung  $\Gamma \vdash \phi$

► Neu hier: Fall  $\forall x.\phi(x)$

► Beweis folgt durch Definition von  $\mathfrak{A} \models \forall x.\phi(x)$

9 [15]

## Vorbereitende Definitionen

### Definition 2 (Theorien, Henkin-Theorien)

- (i) Eine **Theorie** ist eine unter Ableitbarkeit geschlossene Menge  $T \subseteq \mathcal{F}orm_{\Sigma}$
- (ii) **Henkin-Theorie**: Für jedes  $\exists x.\phi(x) \in T$  gibt es **Witness**  $c$  mit  $\exists x.\phi(x) \rightarrow \phi(c) \in T$

### Definition 3

$T'$  ist **konservative** Erweiterung von  $T$  wenn  $T' \cap \Sigma(T) = T$

- Alle Theoreme in  $T'$  in der Sprache von  $T$  sind schon Theoreme in  $T$
- Beispiel:  $\wedge, \rightarrow, \perp$  und volle Aussagenlogik

### Lemma 4 (Konservative Erweiterung bewahrt Konsistenz)

$T$  **konsistent**,  $T'$  **konservative Erweiterung**, dann  $T'$  **konsistent**.

10 [15]

## Maximal konsistente Theorien

### Definition 5

Sei  $T$  Theorie zur Signatur  $\Sigma$ :

$$\Sigma^* = \Sigma \cup \{c_{\phi} \mid \exists x.\phi(x) \in T\}$$

$$T^* = T \cup \{\exists x.\phi(x) \rightarrow \phi(c_{\phi}) \mid \exists x.\phi(x) \text{ geschlossen}\}$$

### Lemma 6

$T^*$  **konservative** Erweiterung von  $T$

11 [15]

## Konstruktion maximal konsistenter Theorien

### Lemma 7

Sei  $T$  Theorie, und seien

$$T_0 = T, T_{n+1} = T_n^*, T_{\omega} = \bigcup_{n \geq 0} T_n$$

Dann ist  $T_{\omega}$  eine **Henkin-Theorie** und **konservativ** über  $T$

### Lemma 8 (Lindenbaum)

Jede **konsistente** Theorie ist in einer **maximal konsistenten** Theorie enthalten (**Henkin-Erweiterung**)

12 [15]

## Vollständigkeit von ND

### Lemma 9 (Existenz von Modellen)

Wenn  $\Gamma$  **konsistent**, dann hat  $\Gamma$  ein **Modell**.

► Beweis: Maximal konsistente Henkin-Erweiterung als Modell

► **Herbrand-Modell**, universelles **Term-Modell**

► Korollar: Wenn  $\Gamma \not\vdash \phi$ , dann  $\Gamma \not\models \phi$

### Theorem 10 (Vollständigkeit von ND)

$\Gamma \vdash \phi$  **gdw.**  $\Gamma \models \phi$

13 [15]

## Entscheidbarkeit

### Theorem 11 (Kompaktheit)

$\Gamma$  hat ein **Modell** **gdw.** jede **endliche** Teilmenge  $\Delta \subseteq \Gamma$  hat ein **Modell**

► Aus Vollständigkeit folgt **nicht** Entscheidbarkeit:

### Theorem 12 (Church)

**Prädikatenlogik** ist **unentscheidbar**.

Beweis:

► Kodierung eines unentscheidbaren Theorie in FOL

► Hier: Kodierung von **Turing-Maschinen** — konstruiere Formel  $U$  so dass  $\vdash U$  **gdw.** Turing-Maschine  $M$  akzeptiert Eingabe  $w$

14 [15]

## Zusammenfassung

► Prädikatenlogik erster Stufe (FOL)

► Natürliches Schließen in FOL: **Substitution** und **Eigenvariablenbedingung**.

► FOL ist

► **konsistent**,

► **vollständig**,

► aber **nicht entscheidbar**.

15 [15]

Formale Modellierung  
Vorlesung 6 vom 28.05.15: FOL mit induktiven Datentypen und Rekursion

Christoph Lüth

Universität Bremen

Sommersemester 2015

## Fahrplan

- ▶ Teil I: Formale Logik
  - ▶ Einführung
  - ▶ Aussagenlogik (PL): Syntax und Semantik, Natürliches Schließen
  - ▶ Konsistenz & Vollständigkeit der Aussagenlogik
  - ▶ Prädikatenlogik (FOL): Syntax und Semantik
  - ▶ Konsistenz & Vollständigkeit von FOL
  - ▶ FOL mit induktiven Datentypen
  - ▶ FOL mit rekursiven Definitionen
  - ▶ Logik höherer Stufe (HOL): Syntax und Eigenschaften
  - ▶ Berechnungsmodelle (Models of Computation)
  - ▶ Die Unvollständigkeitssätze von Gödel
- ▶ Teil II: Spezifikation und Verifikation

## Das Tagesmenü

- ▶ Modellierung natürlicher Zahlen als Beispiel für einen induktiven Datentyp
- ▶ Beweis durch Induktion und Gleichungen
- ▶ Modelle der natürlichen Zahlen

## Die natürlichen Zahlen

- ▶ Der einfachste Datentyp
  - ▶ Aber hinreichend (Turing-mächtig)
- ▶ Wie in FOL formulieren?
  - ▶ Axiomatisch
- ▶ Was sind die Modelle?

## Regeln für die Gleichheit

- ▶ Reflexivität, Symmetrie, Transitivität:

$$\frac{}{x \doteq x} \text{ refl} \quad \frac{x \doteq y}{y \doteq x} \text{ sym} \quad \frac{x \doteq y \quad y \doteq z}{x \doteq z} \text{ trans}$$

- ▶ Kongruenz:

$$\frac{x_1 \doteq y_1, \dots, x_n \doteq y_n}{f(x_1, \dots, x_n) \doteq f(y_1, \dots, y_n)} \text{ cong}$$

- ▶ Substitutivität:

$$\frac{x_1 \doteq y_1, \dots, x_m \doteq y_m \quad P(x_1, \dots, x_m)}{P(y_1, \dots, y_m)} \text{ subst}$$

## Axiomatisierung der natürlichen Zahlen

- ▶ Axiome (erster Versuch):

$$\begin{aligned} \forall x. s(x) \neq 0 & \quad (N1) \\ \forall x. \forall y. s(x) \doteq s(y) \rightarrow x \doteq y & \quad (N2) \\ \forall x. x \neq 0 \rightarrow \exists y. x \doteq s(y) & \quad (N3) \\ \forall x. 0 + x \doteq x & \quad (A1) \\ \forall x. \forall y. s(x) + y \doteq s(x + y) & \quad (A2) \end{aligned}$$

- ▶ Beweise in ND

$$(N1)(N2)(A1)(A2) \vdash \forall x. s(0) + x \doteq s(x)$$

## Modelle für Presburger-Arithmetik

- ▶ Angefangen mit "0" und "s"
- ▶ Axiome (N1), (N2)
- ▶ Füge hinzu: (N3) und

$$\forall x. x \neq \underbrace{s \dots s}_n(x) \quad (K_n)$$

- ▶ "Mehrere" Kopien von  $\mathbb{N}$  weg, Zyklen weg —  $\mathbb{Z}$  bleibt.
- ▶  $\mathbb{N}$  ist das Standardmodell.
- ▶ Alle anderen Strukturen  $\mathbb{N} + \mathbb{Z}$ ,  $\mathbb{N} + \mathbb{Z} + \mathbb{Z}$ , ... sind Nichtstandardmodelle.

## Induktionsschema

- ▶ Axiome für die Multiplikation:

$$\begin{aligned} x \cdot 0 \doteq 0 & \quad (M1) \\ x \cdot s(y) \doteq x \cdot y + x & \quad (M2) \end{aligned}$$

- ▶ Induktionsschema:

$$(P(0) \wedge \forall x. P(x) \rightarrow P(s(x))) \rightarrow \forall x. P(x) \quad (\text{ISNat})$$

- ▶  $P(\$)$  Formelschema

- ▶  $\$$  ausgezeichnetes, neues Symbol ("Loch") und  $P(t) \stackrel{\text{def}}{=} P(\$) \left[ \frac{t}{\$} \right]$

## Hilft das Induktionsschema zum Beweisen?

- ▶ Es gelten:

$$(N1), (N2), (ISNat) \vdash (N3)$$
$$(N1), (N2), (ISNat) \vdash (K_n)$$

- ▶ Beweise in ND

$$(N1)(N2)(A1)(A2)(ISNat) \vdash \forall x. x + 0 \doteq x$$

... und auch

$$(N1)(N2)(A1)(A2)(ISNat) \vdash \forall x. \forall y. x + s(y) \doteq s(x + y)$$

... und auch

$$(N1)(N2)(A1)(A2)(ISNat) \vdash \forall x. \forall y. x + y \doteq y + x$$

9 [12]

## Und was ist mit den Modellen?

- ▶ Ist  $\mathbb{Z}$  jetzt weg?

- ▶ Sei  $PA^\infty \stackrel{\text{def}}{=} (N1), (N2), (ISNat) +$  neues Symbol  $\infty$  und Axiome

$$\infty \neq 0, \infty \neq s(0), \infty \neq s(s(0)), \dots$$

### Theorem (Kompaktheit)

$\Gamma$  hat ein Modell gdw. jede endliche Teilmenge  $\Delta \subseteq \Gamma$  hat ein Modell

### Theorem (Löwenheim-Skolem Theorem)

Wenn FOL-Theorie  $T$  ein unendliches Modell  $M$  hat, dann hat es Modelle beliebiger Größe (Kardinalität).

- ▶ Also hat  $PA^\infty$  Modell, das aber größer ist als  $\mathbb{N}$
- ▶ Es kann in FOL keine Axiomatisierung für  $\mathbb{N}$  geben, die keine Nichtstandardmodelle hat

10 [12]

## Axiomatisierungen der natürlichen Zahlen

- ▶ **Presburger-Arithmetik**

- ▶ 5 Axiome:  $(N1)(N2)(A1)(A2)(ISNat)$
- ▶ Konsistent und vollständig
- ▶ Entscheidbar (Aufwand  $2^{2^n}$ ,  $n$  Länge der Aussage)
- ▶ Enthält Nichtstandardmodelle

- ▶ **Peano-Arithmetik**

- ▶ 7 Axiome:  $(N1)(N2)(A1)(A2)(M1)(M2)(ISNat)$
- ▶ Konsistent, aber unvollständig (bzgl. Standard-Modellen)
- ▶ Enthält Nichtstandardmodelle
- ▶ Nicht entscheidbar

11 [12]

## Zusammenfassung

- ▶ Jede Axiomenmenge zur Formalisierung der Natürlichen Zahlen hat Nichtstandardmodelle
- ▶ Induktionsschema für erzeugte Datentypen
- ▶ Strukturelle Induktionsschema
  - ▶ Einfach, aber zum Beweisen zu rigide

12 [12]

Formale Modellierung  
Vorlesung 7 vom 01.06.15: FOL mit Induktion und Rekursion

Christoph Lüth

Universität Bremen

Sommersemester 2015

Fahrplan

- ▶ Teil I: Formale Logik
  - ▶ Einführung
  - ▶ Aussagenlogik (PL): Syntax und Semantik, Natürliches Schließen
  - ▶ Konsistenz & Vollständigkeit der Aussagenlogik
  - ▶ Prädikatenlogik (FOL): Syntax und Semantik
  - ▶ Konsistenz & Vollständigkeit von FOL
  - ▶ FOL mit induktiven Datentypen
  - ▶ FOL mit rekursiven Definitionen
  - ▶ Logik höherer Stufe (HOL): Syntax und Eigenschaften
  - ▶ Berechnungsmodelle (Models of Computation)
  - ▶ Die Unvollständigkeitssätze von Gödel
- ▶ Teil II: Spezifikation und Verifikation

Das Tagesmenü

- ▶ Beweis von Eigenschaften von Funktionen mit FOL-ND
  - ▶ Rekursive Funktionen und wohlfundierte Induktion
- ▶ Terminierende Funktionen und abgeleitete Induktionsschemata
- ▶ Axiomatische Definition von Theorien ist gefährlich

Beweis der Eigenschaften von Funktion

- ▶ Definiere  $\leq$  und half:

$$\forall x. 0 \leq x \quad (L1)$$

$$\forall x. \forall y. x \leq y \rightarrow s(x) \leq s(y) \quad (L2)$$

$$\text{half}(0) = 0 \quad (H1)$$

$$\text{half}(s(0)) = 0 \quad (H2)$$

$$\forall x. \text{half}(s(s(x))) = s(\text{half}(x)) \quad (H3)$$

- ▶ Beweise

$$(\text{Presburger})(L1)(L2)(H1)(H2)(H3) \vdash \forall x. \text{half}(x) \leq x$$

Mehr Information

- ▶ Besser zum beweisen wäre wenn man gleich hätte

$$\frac{\begin{array}{c} [ \text{half}(c) \leq c ] \\ \vdots \\ \text{half}(0) \leq 0 \quad \text{half}(s(0)) \leq s(0) \quad \text{half}(s(s(c))) \leq s(s(c)) \end{array}}{\forall x. \text{half}(x) \leq x}$$

- ▶ Vergleiche:
  - $\text{half}(0) = 0 \quad (H1)$
  - $\text{half}(s(0)) = 0 \quad (H2)$
  - $\forall x. \text{half}(s(s(x))) = s(\text{half}(x)) \quad (H3)$

- ▶ Generiere Induktionsschema aus rekursiven Funktionsdefinitionen

$$\frac{\begin{array}{c} [ P(c) ] \\ \vdots \\ P(0) \quad P(s(0)) \quad P(s(s(c))) \end{array}}{\forall x. P(x)}$$

Wohlfundierte Induktion

- ▶ Wohlfundiertes Induktionsschema

$$(\forall y. (\forall x. x < y \wedge P(x)) \Rightarrow P(y)) \rightarrow \forall x. P(x)$$

- ▶  $<$  wohlfundierte Relation:

$$\forall X \subseteq \mathbb{N}. X \neq \emptyset \rightarrow \exists x \in X. \forall y \in X. \neg(y < x)$$

Beweis mit wohlfundierter Induktion

- ▶  $<$ -Relation

$$\forall x. 0 < s(x) \quad \forall x, y. x < y \rightarrow s(x) < s(y)$$

- ▶ Beweise  $<$  ist wohlfundiert

$$\frac{\begin{array}{c} [ \forall x. x < c \wedge P(x) ] \\ \vdots \\ P(c) \end{array}}{\forall x. P(x)}$$

$$\frac{\begin{array}{c} \left[ \begin{array}{l} \forall x. x < c \\ \text{half}(x) \leq x \\ c = 0 \end{array} \right] \quad \left[ \begin{array}{l} \forall x. x < c \\ \text{half}(x) \leq x \\ c = s(0) \end{array} \right] \quad \left[ \begin{array}{l} \forall x. x < c \\ \text{half}(x) \leq x \\ \exists u. c = s(s(u)) \end{array} \right] \\ \vdots \\ c = 0 \vee \\ c = s(0) \vee \\ \exists u. c = s(s(u)) \end{array}}{\forall x. \text{half}(x) \leq x}$$

Zulässige Induktionsschema

- ▶ Wann darf man die Rekursionsstruktur verwenden?
- ▶ Definierte Funktion muss ...
  - ▶ eindeutig definiert sein und ...

$$P_0 \rightarrow f(x_1, \dots, x_n) = t_0$$

$$\vdots$$

$$P_n \rightarrow f(x_1, \dots, x_n) = t_n$$

$$P_i \wedge P_j \leftrightarrow \perp, \forall i \neq j$$

- ▶ terminierend
- ▶ Rekursive Definition nach wohlfundierter Relation garantiert Terminierung
- Für jeden atomaren, rekursiven Aufruf  $f(t_1, \dots, t_n)$  erzeuge Terminierungshypothese

$$P_i \rightarrow (x_1, \dots, x_n) > (t_1, \dots, t_n)$$

## Grenzen

$$\forall x. x < 101 \rightarrow f(x) = f(f(x+11))$$
$$\forall x. \neg(x < 101) \rightarrow f(x) = x - 10$$

- ▶  $f$  terminiert immer
- ▶  $f$  ist

$$f(x) := \begin{cases} x - 10 & \text{if } x > 100 \\ 91 & \text{if } x \leq 100 \end{cases}$$

- ▶ Definition der geeigneten wohlfundierten Relation extrem schwierig.

9 [11]

$$\begin{aligned} f(99) &= f(f(110)) & f(87) &= f(f(98)) \\ &= f(100) & &= f(f(f(109))) \\ &= f(f(111)) & &= f(f(99)) \\ &= f(101) & &= f(f(f(110))) \\ &= 91 & &= f(f(100)) \\ & & &= f(f(f(111))) \\ & & &= f(f(101)) \\ & & &= f(91) \\ & & &= f(f(102)) \\ & & &= f(92) \\ & & &= f(f(103)) \\ & & &= f(93) \\ & & &\dots \text{ etc } \dots \\ & & &= f(99) \\ & & &\text{(siehe links)} \\ & & &= 91 \end{aligned}$$

10 [11]

## Zusammenfassung

- ▶ Strukturelle Induktionsschema
  - ▶ Einfach, aber zum Beweisen zu rigide
- ▶ Wohlfundiertes Induktionsschema
  - ▶ Mächtig und flexibel, wenig Hilfestellung beim Beweisen
- ▶ Wohlfundierte Relation aus Rekursionsstruktur terminierender Funktionen
  - ▶ Angepasst an Beweisproblem und vorhandene Definitionsgleichungen
  - ▶ Terminierungsbeweis notwendig (einfache Fälle automatisierbar, i.A. unentscheidbar)
- ▶ Axiomatisierung "von Hand" fehleranfällig
  - ▶ Kernkonzept in Isabelle: **konservative Erweiterung**
  - ▶ Dazu nötig: Rekursion und Induktion als **abgeleitetes** Prinzip

11 [11]

# Formale Modellierung

## Vorlesung 8 vom 08.06.15: Logik Höherer Stufe

Christoph Lüth

Universität Bremen

Sommersemester 2015

16.21.38 2015-07-13

1 [16]

## Fahrplan

- ▶ Alles über **Logik höherer Stufe (Higher-order Logic, HOL)**:
  - ▶ Typen und Terme
  - ▶ Die Basis-Axiome
  - ▶ Definierte Operatoren

2 [16]

## Fahrplan

- ▶ **Teil I: Formale Logik**
  - ▶ Einführung
  - ▶ Aussagenlogik (PL): Syntax und Semantik, Natürliches Schließen
  - ▶ Konsistenz & Vollständigkeit der Aussagenlogik
  - ▶ Prädikatenlogik (FOL): Syntax und Semantik
  - ▶ Konsistenz & Vollständigkeit von FOL
  - ▶ FOL mit induktiven Datentypen
  - ▶ FOL mit rekursiven Definitionen
  - ▶ **Logik höherer Stufe (HOL): Syntax und Eigenschaften**
  - ▶ Berechnungsmodelle (Models of Computation)
  - ▶ Die Unvollständigkeitssätze von Gödel
- ▶ **Teil II: Spezifikation und Verifikation**

3 [16]

## Logik höherer Stufe

- ▶ **Ziel:** Formalisierung von Mathematik
  - ▶ "Logik für Erwachsene"
- ▶ **Problem:** Mögliche Inkonsistenz (Russel's Paradox)
- ▶ **Lösung:** Restriktion vs. Ausdrucksstärke
- ▶ **Alternative Grundlagen:**
  - ▶ Andere Typtheorien (Martin-Löf, Calculus of Constructions)
  - ▶ Ungetypte Mengenlehre (ZFC)
- ▶ **HOL:** guter **Kompromiss**, weit verbreitet.
  - ▶ Klassische Logik höherer Stufe nach Church
  - ▶ Schwächer als ZFC, stärker als Typtheorien

4 [16]

## Warum Logik höherer Stufe?

- ▶ **Aussagenlogik:** keine Quantoren
- ▶ **Logik 1. Stufe:** Quantoren über Terme
$$\forall x y. x = y \longrightarrow y = x$$
- ▶ **Logik 2. Stufe:** Quantoren über Prädikaten und Funktionen
$$\forall P.(P 0 \wedge \forall x.P x \longrightarrow P(S x)) \longrightarrow \forall x.P x$$
- ▶ **Logik 3. Stufe:** Quantoren über Argumenten von Prädikaten
- ▶ **Logik höherer Stufe (HOL):** alle endlichen Quantoren
  - ▶ Keine wesentlichen Vorteile von Logik 2. Ordnung

5 [16]

## Vermeidung von Inkonsistenzen

- ▶ **Russel's Paradox**
  - ▶  $R = \{X \mid X \notin X\}$
  - ▶ Abhilfe: Typen
- ▶ **Gödel's 2. Unvollständigkeitssatz:**
  - ▶ Jede **Logik**, die ihre eigene **Konsistenz** beweist, ist **inkonsistent**.
- ▶ Unterscheidung zwischen **Termen** und **Aussagen**
  - ▶ Dadurch **in** HOL keine Aussage **über** HOL

6 [16]

## Typen

- ▶ Typen *Type* gegeben durch
  - ▶ **Typkonstanten:**  $c \in C_{Type}$  (Menge  $C_{Type}$  durch Signatur gegeben)
    - ▶ *Prop*, *Bool*  $\in C_{Type}$ : *Prop* alle Terme, *Bool* alle Aussagen
  - ▶ **Typvariablen:**  $\alpha \in \mathcal{V}_{Type}$  (Menge  $\mathcal{V}_{Type}$  fest)
  - ▶ **Funktionen:**  $s, t \in Type$  dann  $s \Rightarrow t$  in *Type*
- ▶ **Konvention:** Funktionsraum nach rechts geklammert
$$\alpha \Rightarrow \beta \Rightarrow \gamma \text{ für } \alpha \Rightarrow (\beta \Rightarrow \gamma)$$

7 [16]

## Terme

- ▶ Terme *Term* gegeben durch
  - ▶ **Konstanten:**  $c \in C$  (Menge  $C$  durch Signatur gegeben)
  - ▶ **Variablen:**  $v \in \mathcal{V}$
  - ▶ **Applikation:**  $s, t \in Term$  dann  $s t \in Term$
  - ▶ **Abstraktion:**  $x \in \mathcal{V}, t \in Term$  dann  $\lambda x. t \in Term$
- ▶ **Konventionen:** Applikation links geklammert, mehrfache Abstraktion
$$\lambda x y z. f x y z \text{ für } \lambda x. \lambda y. \lambda z. ((f x) y) z$$

8 [16]

## Basis-Syntax

$= :: \alpha \Rightarrow \alpha \Rightarrow \text{Bool}$   
 $\longrightarrow :: \text{Bool} \Rightarrow \text{Bool} \Rightarrow \text{Bool}$   
 $\iota :: (\alpha \Rightarrow \text{Bool}) \Rightarrow \alpha$

▶ Einbettung (wird weggelassen)  
 $\text{trueprop} :: \text{Bool} \Rightarrow \text{Prop}$

▶ Basis-Operatoren:  $=, \longrightarrow, \iota$

▶ Syntaktische Konventionen:
 

- ▶ Bindende Operatoren:  $\forall, \exists, \iota$   
 $\forall x.P \equiv \forall(\lambda x.P)$
- ▶ Infix-Operatoren:  $\wedge, \vee, \longrightarrow, =$
- ▶ Mixfix-Operator:  
 $\text{if } b \text{ then } p \text{ else } q \equiv \text{if } b \text{ } p \text{ } q$

$\neg :: \text{Bool} \Rightarrow \text{Bool}$   
 $\top :: \text{Bool}$   
 $\perp :: \text{Bool}$   
 $\text{if} :: \text{Bool} \Rightarrow \alpha \Rightarrow \alpha \Rightarrow \alpha$   
 $\forall :: (\alpha \Rightarrow \text{Bool}) \Rightarrow \text{Bool}$   
 $\exists :: (\alpha \Rightarrow \text{Bool}) \Rightarrow \text{Bool}$   
 $\wedge :: \text{Bool} \Rightarrow \text{Bool} \Rightarrow \text{Bool}$   
 $\vee :: \text{Bool} \Rightarrow \text{Bool} \Rightarrow \text{Bool}$

9 [16]

## Basis-Axiome I: Gleichheit

▶ Reflexivität:

$$\frac{}{t = t} \text{ refl}$$

▶ Substitutivität:

$$\frac{s = t \quad P \ s}{P \ t} \text{ subst}$$

▶ Extensionalität:

$$\frac{f \ x = g \ x}{(\lambda x. f \ x) = (\lambda x. g \ x)} \text{ ext } (*)$$

(\*) **Eigenvariablenbedingung**:  $x$  nicht frei in offenen Vorbedingungen

▶ Einführungsregel:

$$\frac{}{(P \longrightarrow Q) \longrightarrow (Q \longrightarrow P) \longrightarrow (P = Q)} \text{ iff}$$

10 [16]

## Abgeleitete Operatoren

$\top \equiv (\lambda x. x) = (\lambda x. x)$   
 $\forall P \equiv (P = \lambda x. \top)$   
 $\exists P \equiv \forall Q. (\forall x. P \ x \longrightarrow Q) \longrightarrow Q$   
 $\perp \equiv \forall P. P$   
 $\neg P \equiv P \longrightarrow \perp$   
 $P \wedge Q \equiv \forall R. (P \longrightarrow Q \longrightarrow R) \longrightarrow R$   
 $P \vee Q \equiv \forall R. (P \longrightarrow R) \longrightarrow (Q \longrightarrow R) \longrightarrow R$   
 $\text{if } P \text{ then } x \text{ else } y \equiv \iota z. (P = \top \longrightarrow z = x) \wedge (P = \perp \longrightarrow z = y)$

11 [16]

## Basis-Axiome II: Implikation und Auswahl

▶ Einführungsregel Implikation:

$$\frac{\begin{array}{c} [P] \\ \vdots \\ Q \end{array}}{P \longrightarrow Q} \text{ impl}$$

▶ Eliminationsregel  
Auswahloperator:

$$\frac{(\lambda x. x = a) = a}{(\lambda x. x = a) = a} \text{ the\_eq}$$

▶ HOL ist klassisch:

$$\frac{}{(P = \top) \vee (P = \perp)} \text{ true\_or\_false}$$

▶ Eliminationsregel Implikation:

$$\frac{P \longrightarrow Q \quad P}{Q} \text{ mp}$$

12 [16]

## Die Basis-Axiome (Isabelle-Syntax)

$\text{refl} : t = t$   
 $\text{subst} : \llbracket s = t; P(s) \rrbracket \Longrightarrow P(t)$   
 $\text{ext} : \llbracket \lambda x. fx = gx \rrbracket \Longrightarrow (\lambda x. fx) = (\lambda x. gx)$   
 $\text{impl} : \llbracket P \Longrightarrow Q \rrbracket \Longrightarrow P \longrightarrow Q$   
 $\text{mp} : \llbracket P \longrightarrow Q; P \rrbracket \Longrightarrow Q$   
 $\text{iff} : (P \longrightarrow Q) \longrightarrow (Q \longrightarrow P) \longrightarrow (P = Q)$   
 $\text{the\_eq} : (\lambda x. x = a) = a$   
 $\text{true\_or\_false} : (P = \top) \vee (P = \perp)$

13 [16]

## Eigenschaften der Logik höherer Stufe

▶ **Konsistent** (soweit wir wissen)

▶ **Unvollständig**

▶ ... und damit unentscheidbar

▶ Beweis: folgt aus den Gödelschen Unvollständigkeitssätzen

14 [16]

## Erweiterungen

- ▶ Weitere **Operatoren**
- ▶ Weitere **Typen**: natürliche Zahlen, Datentypen
- ▶ Alle Erweiterungen sind **konservativ** und damit **konsistenzbewahrend**

15 [16]

## Zusammenfassung

Logik **höherer Stufe** (HOL):

- ▶ Syntax basiert auf dem **einfach getypten  $\lambda$ -Kalkül**
- ▶ **Drei** Basis-Operatoren, **acht** Basis-Axiome
- ▶ Rest folgt durch **konservative Erweiterung** — Donnerstag

16 [16]

# Formale Modellierung

## Vorlesung 9 vom 15.06.15: Berechnungsmodelle

Christoph Lüth

Universität Bremen

Sommersemester 2015

## Fahrplan

- ▶ Teil I: Formale Logik
  - ▶ Einführung
  - ▶ Aussagenlogik (PL): Syntax und Semantik, Natürliches Schließen
  - ▶ Konsistenz & Vollständigkeit der Aussagenlogik
  - ▶ Prädikatenlogik (FOL): Syntax und Semantik
  - ▶ Konsistenz & Vollständigkeit von FOL
  - ▶ FOL mit induktiven Datentypen
  - ▶ FOL mit rekursiven Definitionen
  - ▶ Logik höherer Stufe (HOL): Syntax und Eigenschaften
  - ▶ Berechnungsmodelle (Models of Computation)
  - ▶ Die Unvollständigkeitssätze von Gödel
- ▶ Teil II: Spezifikation und Verifikation

## Fahrplan für heute

- ▶ Eine Behauptung von Church
- ▶ Registermaschinen
- ▶ Rekursiv definierbare Funktionen
- ▶ Universelle Berechenbarkeit

## Die Churchsche Behauptung

### Church's Thesis (1936)

Intuitiv **berechenbare** Funktionen sind genau die durch Rekursion definierbaren.

- ▶ Nicht beweisbar: das Konzept "intuitiv berechenbar" ist nicht formalisierbar.
- ▶ Aber: es gibt sehr **viele, unterschiedliche** (und meist voneinander unabhängig gefundene) **formale** Definition von **Berechenbarkeit**, die alle **gleich mächtig** sind.
- ▶ Es spricht alles für die Churchsche Behauptung.

## Registermaschinen

- ▶ Variation von **Turing-Maschinen**, konzeptionell etwas einfacher.

### Registermaschinen (Minsky)

- ▶ Sequenz von **Registern**  $R_1, R_2, \dots$ , die natürliche Zahlen enthalten
- ▶ **Programm** besteht aus:
  - ▶ Endliche Anzahl von **Zuständen**  $S_0, S_1, S_2, \dots$ , und
  - ▶ für  $i > 0$ , Anweisung ( $S_0$ : **terminaler** Zustand)

- ▶ Anweisungen:



Erhöhe  $R_j$  um 1 und gehe zu  $S_k$  Wenn  $R_j = 0$ , gehe zu  $S_j$ ; ansonsten verringere  $R_j$  um 1 und gehe zu  $S_k$ .

## Beispielprogramme

Programm 1:

- 1: (1, -, 1, 2)
- 2: (2, -, 3, 0)
- 3: (1, +, 2)

$R'_1 := R_2;$   
 $R'_2 := 0$

Programm 2:

- 1: (3, -, 1, 2)
- 2: (2, -, 3, 6)
- 3: (3, +, 4)
- 4: (1, +, 5)
- 5: (1, +, 2)
- 6: (3, -, 7, 0)
- 7: (2, +, 6)

$R'_3 := 0;$   
 $R'_1 := R_1 + 2 * R_2;$   
 $R'_2 := R_2$

## Berechenbarkeit durch Registermaschine

- ▶ NB: Jede Registermaschine hat **endlich** viele Register, aber Gesamtanzahl **unbeschränkt**.
- ▶ Eine Registermaschine  $R$  **berechnet** eine Funktion  $f : \mathbb{N}^k \rightarrow \mathbb{N}$  (mit einem Programm  $P$ ), wenn beim Start (von  $S_1$ ) mit Registerinhalt  $(n_1, n_2, \dots, n_k, 0, \dots)$  die Maschine schließlich mit Registerinhalt  $f(n_1, \dots, n_k)$  in  $R_1$  anhält.
- ▶ Aber welche Funktionen sind **berechenbar**?

## Theorem 1: Berechenbare Funktionen

- (i) Die **Projektionen**  $\pi_i(n_1, \dots, n_k) = n_i$  ist berechenbar.
- (ii)  $const_0 : \mathbb{N} \rightarrow \mathbb{N}$  und  $succ : \mathbb{N} \rightarrow \mathbb{N}$  sind berechenbar.
- (iii) **Komposition**: Sei  $f$  mit Arität  $k$  berechenbar, und  $g_1, \dots, g_k$  mit Arität  $l$  berechenbar, dann ist  $h$  berechenbar:

$$h \stackrel{def}{=} f(g_1(n_1, \dots, n_l), \dots, g_k(n_1, \dots, n_l))$$

- (iv) **Rekursion**: Seien  $f$  und  $g$  berechenbar mit Arität  $k$  bzw.  $k + 2$ , dann ist  $h$  berechenbar:

$$h(n_1, \dots, n_k, 0) \stackrel{def}{=} f(n_1, \dots, n_k)$$

$$h(n_1, \dots, n_k, n_{k+1} + 1) \stackrel{def}{=} g(n_1, \dots, n_k, n_{k+1}, h(n_1, \dots, n_k, n_{k+1}))$$

- (v) **Minimalisierung ( $\mu$ -Operator)**: Sei  $f$  berechenbar mit Arität  $k + 1$ , dann ist  $g$  berechenbar:

$$g(n_1, \dots, n_k) = n \text{ mit } f(n_1, \dots, n_k, n) = 0 \text{ und } \forall m. m < n \rightarrow f(n_1, \dots, n_k, m) > 0$$

## Berechenbarkeit, Rekursion, Primitive Rekursion

- ▶ **Primitiv rekursive Funktionen:** kleinste unter (i) – (iv) abgeschlossene Klasse von Funktionen
- ▶ **Rekursive Funktionen:** kleinste unter (i) – (v) abgeschlossene Klasse von Funktionen
- ▶ Primitiv rekursive Funktionen sind **total**, aber nicht jede totale rekursive Funktion ist primitiv rekursiv. (Gegenbeispiel: Ackermann)
- ▶ Es gilt der Umkehrschluss:

### Theorem 2

Jede rekursive Funktion ist berechenbar.

9 [13]

## Kodierung von Programmen

- ▶ Programme können durch Zahlen **kodiert** werden (Gödel-Kodierung):

$$(j, +, k) \mapsto 2^j \cdot 5^k$$
$$(j, -, k, l) \mapsto 2^j \cdot 3 \cdot 5^k \cdot 7^l$$
$$i_1, \dots, i_n \mapsto 2^{i_1} \cdot 3^{i_2} \cdot \dots \cdot p_{n-1}^{i_n}$$

- ▶ Notation:  $f_{n,k}$  ist die durch  $n$  kodierte Funktion der Arität  $k$ .

### Theorem 3 (Universelle Funktionen)

Es gibt **universelle** rekursive Funktion  $u$  so dass

- ▶  $u(n, k, m) = r$  wenn  $n$  ein Programm kodiert,  $k$  ein  $k$ -Tupel  $(m_1, \dots, m_k)$  und  $f_{n,k}(m_1, \dots, m_k) = r$ ,
- ▶ und ansonsten  $u(n, k, m)$  undefiniert.

10 [13]

## Rekursive Funktionen und Peano-Arithmetik

### Theorem 4 (Rekursion ist PA)

Jede rekursive Funktion ist in PA definierbar.

- ▶ Ohne Multiplikation funktioniert der Beweis nicht.
- ▶ Deshalb: nicht jede rekursive Funktion in Presburger-Arithmetik definierbar.

11 [13]

## Weitere Modelle von Berechenbarkeit

- ▶ Turing-Maschinen (remember those?)
- ▶ Chomsky-Grammatiken (Typ 0)
- ▶  $\lambda$ -Kalkül
- ▶ Kombinatorlogik
- ▶  $\pi$ -Kalkül
- ▶ Neuronale Netze
- ▶ Gegenbeispiel: **Aktoren** (Carl Hewitt)
  - ▶ Literaturlage nicht eindeutig ...

12 [13]

## Zusammenfassung

- ▶ Behauptung von Church: intuitiv berechenbare Funktionen sind genau durch Turing-Maschinen berechenbare.
- ▶ Vielzahl von **formalen** Berechnungsmodellen, welche die gleiche **Mächtigkeit** haben.
- ▶ Wir haben gesehen:
  - ▶ Registermaschinen
  - ▶ Rekursive Funktionen
  - ▶ In Peano-Arithmetik definierbare Funktionen
- ▶ Beweis der Mächtigkeit durch **Kodierung**:
  - ▶ Berechnungen können in Zahlen kodiert werden
  - ▶ Damit **universelle** Berechnungen (Compiler!) möglich
- ▶ Nächste Woche: Grenzen der Beweisbarkeit — die Gödelschen Unvollständigkeitssätze

13 [13]

Formale Modellierung  
Vorlesung 10 vom 19.06.14: Die Unvollständigkeitssätze von Gödel

Christoph Lüth

Universität Bremen

Sommersemester 2015

Fahrplan

- ▶ Teil I: Formale Logik
  - ▶ Einführung
  - ▶ Aussagenlogik (PL): Syntax und Semantik, Natürliches Schließen
  - ▶ Konsistenz & Vollständigkeit der Aussagenlogik
  - ▶ Prädikatenlogik (FOL): Syntax und Semantik
  - ▶ Konsistenz & Vollständigkeit von FOL
  - ▶ FOL mit induktiven Datentypen
  - ▶ FOL mit rekursiven Definitionen
  - ▶ Logik höherer Stufe (HOL): Syntax und Eigenschaften
  - ▶ Berechnungsmodelle (Models of Computation)
  - ▶ Die Unvollständigkeitssätze von Gödel
- ▶ Teil II: Spezifikation und Verifikation

Das Tagesmenü

- ▶ Die Gödelschen Unvollständigkeitssätze

Gödels erster Unvollständigkeitssatz

Jede konsistente Theorie, die hinreichend expressiv ist, um die natürlichen Zahlen zu formalisieren, erlaubt die Formulierung von wahren Aussagen, die weder beweisbar noch widerlegbar sind.

- ▶ Zu jeder Formel  $\varphi$  gibt es eine natürliche Zahl, die diese Formel eindeutig kodiert  $[\varphi]$
- ▶ Zu jedem ND-Beweis  $D$  für  $\varphi$  gibt es eine natürliche Zahl, die diesen Beweis eindeutig kodiert  $[D]$
- ▶ Beweisbarkeit von  $\varphi$  in  $\mathbb{N}$  ist als Prädikat  $\text{Provable}([\varphi])$  formalisierbar in PA
- ▶ Konstruktion einer Formel mit Aussage "Ich bin nicht beweisbar"  
 $\varphi \leftrightarrow \neg \text{Prov}([\varphi])$

Gödel-Kodierung

- ▶ Folgende Funktion ist definierbar in PA:

$$(n, m) \stackrel{\text{def}}{=} 2^n \cdot 3^m$$

- ▶ Es gibt eindeutige Projektionen (ebenfalls definierbar in PA):

$$\text{Left}((n, m)) = n \quad \text{Right}((n, m)) = m$$

- ▶ Kodierung von Sequenzen ( $p_k$  ist die  $k$ -te Primzahl):

$$\langle n_0, \dots, n_{k-1} \rangle \stackrel{\text{def}}{=} 2^{n_0+1} \cdot 3^{n_1+1} \cdot 5^{n_2+1} \dots p_k^{n_{k-1}+1}$$

- ▶ Alternative Definition nach Cantor:

$$(n, m) = \frac{n + m^2 + 3x + 2y}{2}$$

Gödel-Kodierung für Terme

Signatur  $\Sigma = (\mathcal{F}, \mathcal{P})$ , Variablen  $X$

- ▶ Variablen  $x_1, x_2, \dots \in X$ :

$$[x_i] \stackrel{\text{def}}{=} (0, i)$$

- ▶ Funktionen  $f_1, \dots \in \mathcal{F}$ :

$$[f_i] \stackrel{\text{def}}{=} (1, i)$$

- ▶ Terme

$$[f_i(t_1, \dots, t_n)] \stackrel{\text{def}}{=} \langle [f_i], [t_1], \dots, [t_n] \rangle$$

Gödel-Kodierung für Formeln

Signatur  $\Sigma = (\mathcal{F}, \mathcal{P})$ , Variablen  $X$

- ▶ Prädikate  $p_1, \dots \in \mathcal{P}$ ,  $\perp \stackrel{\text{def}}{=} p_1, \neg \stackrel{\text{def}}{=} p_2$

$$[p_i] \stackrel{\text{def}}{=} (2, i)$$

- ▶ Atome

$$[p_i(t_1, \dots, t_n)] \stackrel{\text{def}}{=} \langle [p_i], [t_1], \dots, [t_n] \rangle$$

- ▶ Konnektive und Quantoren

$$\begin{aligned} [\neg] &= (3, 1), [\wedge] = (3, 2), [\vee] = (3, 3) \\ [\rightarrow] &= (3, 4), [\leftrightarrow] = (3, 5), [\forall] = (3, 6), [\exists] = (3, 7) \end{aligned}$$

Gödel-Kodierung für Formeln II

Signatur  $\Sigma = (\mathcal{F}, \mathcal{P})$ , Variablen  $X$

- ▶  $[\neg\varphi] = (\neg, [\varphi])$
- ▶  $[\psi \diamond \varphi] = (\diamond, [\psi], [\varphi])$  mit  $\diamond \in \{\wedge, \vee, \rightarrow, \leftrightarrow\}$
- ▶  $[Q x_i. \varphi] = (Q, [x_i], [\varphi])$  mit  $Q \in \{\forall, \exists\}$

Lemma 1 (Eigenschaften der Gödel-Kodierung)

- ▶ Sei  $G \stackrel{\text{def}}{=} \{[\varphi] \mid \varphi \text{ Variable, Term, oder Formel}\}$
- ▶  $G$  ist entscheidbar
- ▶  $[n] = \varphi \Leftrightarrow [\varphi] = n$  ist eindeutig definiert auf  $G$
- ▶ Substitutionsfunktion  $\text{subst}(n, x, t) = m$  definierbar auf  $G$

$$[\varphi[t/x]] = \text{subst}([\varphi], [x], [t])$$

## Gödel-Kodierung für Ableitungen

- ▶ Gödel Kodierung für Hypothesen Liste:

$$[[\varphi_1, \dots, \varphi_n]] = \begin{cases} 1 & \text{if } n = 0 \\ \langle (4, [\varphi_1]), \dots, (4, [\varphi_n]) \rangle & \text{if } n > 0 \end{cases}$$

$$n \in h \Leftrightarrow \begin{cases} \perp & \text{if } h = 1 \\ \top & \text{if } h = (4, n) \vee h = \langle (4, n), m \rangle \\ n \in m & \text{if } h = \langle (4, q), m \rangle, \neg(q = n) \end{cases}$$

- ▶ Definition von **Konkatenation** \* und **Streichen** von Hypothesen

9 [16]

## Gödel-Kodierung für Ableitungen

$$\left[ \frac{D_1 \quad D_2}{\phi \wedge \psi} \wedge I \right] = \langle (5, [\wedge]), \left[ \frac{D_1}{\phi} \right], \left[ \frac{D_2}{\psi} \right], [\phi \wedge \psi] \rangle$$

$$\left[ \frac{D}{\phi \wedge \psi} \wedge E_L \right] = \langle (6, [\wedge]), \left[ \frac{D}{\phi \wedge \psi} \right], [\phi] \rangle$$

10 [16]

## Gödel-Kodierung für Ableitungen

$$\left[ \frac{D}{\phi \rightarrow \psi} \rightarrow I \right] = \langle (5, [\rightarrow]), \left[ \frac{D}{\psi} \right], [\phi \rightarrow \psi] \rangle$$

$$\left[ \frac{D_1 \quad D_2}{\phi \rightarrow \psi} \rightarrow E \right] = \langle (6, [\rightarrow]), \left[ \frac{D_1}{\phi} \right], \left[ \frac{D_2}{\psi} \right], [\psi] \rangle$$

11 [16]

## Gödel-Kodierung für Ableitungen

- ▶ Entsprechend für RAA,  $\forall I$ ,  $\forall E$
- ▶ Definiere  $\text{Der}(p, h, z)$ :  $[p]$  ist Beweis für  $[z]$  aus Hypothesen  $[h]$

$$\text{Der}(p, h, z) \stackrel{\text{def}}{=} \begin{cases} (4, z) \in h & \text{Hypothese} \\ \vee \exists p_1, h_1, z_1, p_2, h_2, z_2. & \wedge I \\ \quad \text{Der}(p_1, h_1, z_1) \wedge \text{Der}(p_2, h_2, z_2) \wedge & \\ \quad h = h_1 * h_2 \wedge & \\ \quad p = \langle (5, [\wedge]), p_1, p_2, [\perp z_1] \wedge [\perp z_2] \rangle & \\ \vee \exists p_1, h_1, z_1, u. & \rightarrow I \\ \quad \text{Der}(p_1, h_1, z_1) \wedge & \\ \quad h = \text{Streiche}(u, h_1) \wedge & \\ \quad p = \langle (5, [\rightarrow]), p_1, [\perp u] \rightarrow [\perp z_1] \rangle & \\ \dots & \end{cases}$$

12 [16]

## Beweisbarkeit

- ▶ Peano-Axiome + Erweiterung: PA Sei  $Ax : \mathbb{N}$  Prädikat

$$Ax(n) \leftrightarrow \bigvee_{\varphi \in PA} n = [\varphi]$$

- ▶  $\text{Prov}(p, f)$ :  $p$  is Gödelnummer eines ND-Beweis für  $[f]$

$$\text{Prov}(p, f) \Leftrightarrow \exists h. (\text{Der}(p, h, f) \wedge \forall g. g \in h \Rightarrow Ax(g))$$

- ▶  $\text{Thm}(f)$ :  $[f]$  ist ein Theorem

$$\text{Thm}(f) \leftrightarrow \exists p. \text{Prov}(p, f)$$

13 [16]

## Fixpunkt-Theorem

### Theorem 2 (Fixpoint Theorem)

For each formula  $\varphi(x)$  with only one free variable  $x$  there exists a formula  $\psi$  such that  $\vdash \varphi([\psi]) \leftrightarrow \psi$

14 [16]

## Gödels erster Unvollständigkeitssatz

Jede konsistente Theorie, die hinreichend expressiv ist, um PA zu formalisieren erlaubt die Formulierung von wahren Aussagen, die weder beweisbar noch widerlegbar sind.

- ▶ Fixpunktsatz anwenden auf  $\varphi(x) \stackrel{\text{def}}{=} \neg \text{Thm}([\varphi])$

- ▶ Es gibt  $\psi$  so dass  $\vdash \psi \leftrightarrow \neg \text{Thm}([\psi])$

- ▶ Gödel sentence: "Ich bin nicht beweisbar"

- ▶ Es gilt  $\text{PA} \models \psi \leftrightarrow \neg \text{Thm}([\psi])$

- ▶ Annahme:  $\text{PA} \vdash \psi$ , dann  $\text{PA} \models \text{Thm}([\psi])$

$$\Leftrightarrow \text{PA} \models \exists x. \text{Prov}(x, [\psi]) \quad \Leftrightarrow \text{PA} \models \text{Prov}(n, [\psi]) \text{ for some } n$$

$$\Leftrightarrow \vdash \text{Prov}(n, [\psi]) \text{ for some } n \quad \Leftrightarrow \vdash \psi$$

$$\Rightarrow \vdash \neg \text{Thm}([\psi]) \quad \Rightarrow \text{PA} \models \neg \text{Thm}([\psi])$$

- ▶ **Widerspruch** — also ist  $\psi$  wahr in **PA**, aber  $\text{PA} \not\vdash \psi$

15 [16]

## Zusammenfassung

### Gödels erster Unvollständigkeitssatz

Jede konsistente Theorie, die hinreichend expressiv ist, um die Peano-Arithmetik (**PA**) zu formalisieren, ist entweder konsistent oder unvollständig (erlaubt die Formulierung von wahren Aussagen, die weder beweisbar noch widerlegbar sind).

### Gödels zweiter Unvollständigkeitssatz

When **PA** consistent is, können wir die Konsistenz von **PA** nicht herleiten ( $\text{PA} \not\vdash \text{ConsisPA}$ ).

- ▶ Beweis durch Kodierung von Formeln und Ableitbarkeit in **PA**
- ▶ Reflektion der Beweisbarkeit in einer Formel
- ▶ Konstruktion einer Formel mit der Aussage "Ich bin nicht beweisbar"

16 [16]

Formale Modellierung  
Vorlesung 11 vom 28.06.2014: Formale Modellierung von Software

Christoph Lüth

Universität Bremen

Sommersemester 2015

16:21:41 2015-07-13

1 [24]

## Fahrplan

- ▶ Teil I: Formale Logik
- ▶ Teil II: Spezifikation und Verifikation
  - ▶ Formale Modellierung von Software
  - ▶ Temporale Logik und Modellprüfung
  - ▶ Zusammenfassung, Rückblick, Ausblick

2 [24]

## Das Tagesmenü

- ▶ Modellierung von Software: Spezifikation
  - ▶ Modellierung des Verhaltens (nicht des Programmes)
- ▶ Ein weites Feld:
  - ▶ Entwicklungsmodelle, Vorgehensmodelle, ...
  - ▶ Informelle Sprachen (UML)
- ▶ Hier: formale Spezifikation

3 [24]

## Algebraische Spezifikation

- ▶ Idee: Spezifikation ist **Signatur**, Programme sind **Algebren**
- ▶ Mathematische Grundlage: universelle Algebra
- ▶ Geschichtliches:
  - ▶ Entstanden um 1976 (ADJ-Gruppe)
  - ▶ In den 80ern Vielzahl von algebraischen Sprachen
  - ▶ Ende 90er Entwicklung der Einheitssprache CASL
- ▶ Beispielsprachen: CASL, OBJ, Maude

4 [24]

## Die Grundidee

- ▶ Deklaration von Typen und Operationen in **Signatur**
- ▶ Gewünschte **Eigenschaften** als **Axiome**
- ▶ Semantik: *lose* (alle Algebren) vs. *initial* (Termalgebra)

5 [24]

## Ein klassisches Beispiel

- ▶ Ein **Stack** hat zwei **Sorten** und vier **Operationen**:

```
typedecl 'a stack
axiomatization
  empty :: "'a stack"
  push  :: "'a stack => 'a => 'a stack"
  pop   :: "'a stack => 'a stack"
  top   :: "'a stack => 'a"
```

- ▶ **Axiome**: pop und top invers zu push

```
where a1: "top (push s x) = x"
      a2: "pop (push s x) = s"
```

- ▶ pop, top partiell?
- ▶ Keine Seiteneffekte

6 [24]

## Modellbasierte Spezifikation

- ▶ Grundidee: Konstruktion eines (**nicht-ausführbaren**) **Modells**
- ▶ Basiert auf konsistenter, ausdrucksstarker Logik:
  - ▶ Mengenlehre (getypt, ungetypt),
  - ▶ Typtheorie
  - ▶ HOL
- ▶ Geschichtliches:
  - ▶ VDM, entwickelt früher 70er (IBM-Labor Wien)
  - ▶ Früher industrieller Einsatz
  - ▶ Standardisierung in den 90ern (VDM, Z)
- ▶ Beispielsprachen: VDM, Z, B

7 [24]

## Das klassische Beispiel

- ▶ Der Stack modellbasiert:
  - ▶ **Sehr** einfach — der Stack ist eine Liste

```
type_synonym 'a stack = "'a list"
definition empty :: "'a stack"
where "empty == []"
definition push  :: "'a stack => 'a => 'a stack"
where "push s a == a# s"
definition pop   :: "'a stack => 'a stack"
where "pop s == tl s"
definition top   :: "'a stack => 'a"
where "top s == hd s"
```

8 [24]

## Vor- und Nachteile

- ▶ Algebraische Spezifikationen:
  - ▶ Abstrakter, leichter zu schreiben
  - ▶ aber werden leicht inkonsistent
- ▶ Modellbasierte Spezifikationen:
  - ▶ Konsistenz garantiert, ausdrucksmächtiger
  - ▶ aber manchmal zu mächtig

9 [24]

## Weitere Modellierungssprachen

- ▶ JML — **light-weight** oder **code-based** specification
- ▶ UML — **semi-formal**

10 [24]

## Java Modeling Language (JML)

- ▶ Zentral: **funktionale Korrektheit**
- ▶ "Design by contract"
- ▶ Spezifikation nahe am Code (Annotationen)
- ▶ Vor/Nachbedingungen, Invarianten
- ▶ Werkzeuge: ESC/Java2, Mobius

11 [24]

## JML: Erstes Beispiel

```
public abstract class LinearSearch
{
    //@ requires j >= 0;
    public abstract /*@ pure @*/ boolean f(int j);

    //@ ensures 0 <= \result;
    //@ ensures (\exists int j; 0 <= j && j <= \result; f(j));
    public abstract /*@ pure @*/ int limit();

    /*@ public normal_behavior
       @ requires (\exists int i; 0 <= i && i <= limit(); f(i));
       @ assignable \nothing;
       @ ensures f(\result) &&
       @ (\forall int i; 0 <= i && i < \result; ! f(i));
       @*/
    public int find()
    }
}
```

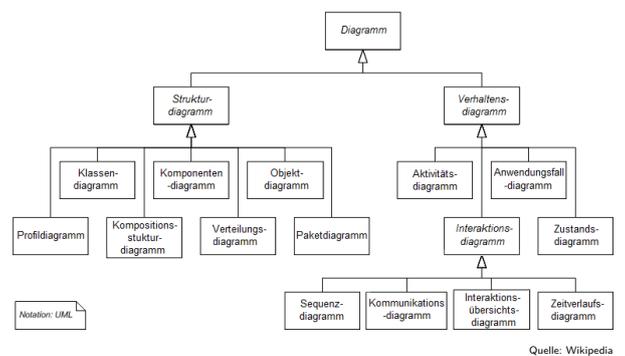
12 [24]

## UML als formale Spezifikationsprache

Diagrammtyp	Modellierte Aspekte	Formal
Klassendiagramm	Statische Systemstruktur	Ja
Paketdiagramm	Pakete, Namensräume	Nein
Objektdiagramm	Zustand von Objekten	(Ja)
Kompositionsstrukturdiagramm	Kollaborationen	Nein
Komponentendiagramm	Dynamische Systemstruktur	(Nein)
Verteilungsdiagramm	Implementierungsaspekte	Nein
Use-Case-Diagramm	Ablauf en gros	Nein
Aktivitätsdiagramm	Ablauf en detail	Nein
Zustandsdiagramm	Zustandsübergänge	Ja
Sequenzdiagramm	Kommunikation	Ja
Kommunikationsdiagramm	Struktur der Kommunikation	(Ja)
Zeitverlaufdiagramm	Echtzeitaspekte	(Ja)

13 [24]

## Diagramme in UML 2.3



14 [24]

## OCL

- ▶ Object Constraint Language
- ▶ Mathematisch **präzise** Sprache für UML
- ▶ OO meets Z
- ▶ Entwickelt in den 90ern
- ▶ Formale **Constraints** an UML-Diagrammen

15 [24]

## OCL Basics

- ▶ **Getypte** Sprache
- ▶ Dreiwertige Logik
- ▶ Ausdrücke immer im **Kontext**:
  - ▶ **Invarianten** an Klassen, Interfaces, Typen
  - ▶ **Vor/Nachbedingungen** an Operationen oder Methoden

16 [24]

## OCL Syntax

- ▶ Invarianten:

```
context class
  inv: expr
```

- ▶ Vor/Nachbedingungen:

```
context Type :: op(arg1 : Type) : ReturnType
  pre: expr
  post: expr
```

- ▶ expr ist ein OCL-Ausdruck vom Typ Boolean

17 [24]

## Undefiniertheit in OCL

- ▶ Undefiniertheit **propagiert** (alle Operationen **strikt**)

- ▶ Ausnahmen:

- ▶ Boolesche Operatoren (and, or **beidseitig** nicht-strikt)
- ▶ Fallunterscheidung
- ▶ Test auf Definiertheit: oclIsUndefined mit

$$\text{oclIsUndefined}(e) = \begin{cases} \text{true} & e = \perp \\ \text{false} & \text{otherwise} \end{cases}$$

- ▶ Resultierende Logik: **dreiwertig**

18 [24]

## Dreiwertige Logik

- ▶ Wahrheitstabelle (**starke Kleene-Logik,  $K_3$** ):

	$\neg$									
$\perp$	$\perp$		$\wedge$	$\perp$	0	1	$\vee$	$\perp$	0	1
0	1		$\perp$	$\perp$	0	$\perp$	$\perp$	$\perp$	0	1
1	0		0	0	0	0	0	$\perp$	0	1
			1	$\perp$	0	1	1	1	1	1
	$\rightarrow$	$\perp$	0	1			$\leftrightarrow$	$\perp$	0	1
$\perp$	$\perp$	$\perp$	$\perp$	1			$\perp$	$\perp$	$\perp$	$\perp$
0	1	1	1				0	$\perp$	1	0
1	$\perp$	0	1				1	$\perp$	0	1

- ▶ Fun Fact:  $K_3$  hat **keine Tautologien**.

- ▶ Alternative: **schwache Kleene-Logik** (alle Operatoren strikt)

19 [24]

## OCL Typen

- ▶ Basistypen:

- ▶ Boolean, Integer, Real, String
- ▶ OclAny, OclType, OclVoid

- ▶ Collection types: Set, OrderedSet, Bag, Sequences

- ▶ Modelltypen

20 [24]

## Basistypen und Operationen

- ▶ Integer ( $\mathbb{Z}$ )

- ▶ Real ( $\mathbb{R}$ )

- ▶ Integer Subklasse von Real
- ▶ round, floor von Real nach Integer

- ▶ String (Zeichenketten)

- ▶ substring, toReal, toInteger, characters etc.

- ▶ Boolean (Wahrheitswerte)

- ▶ or, xor, and, implies
- ▶ Sowie Relationen auf Real, Integer, String

21 [24]

## Collection Types

- ▶ Set, OrderedSet, Bag, Sequence

- ▶ Operationen auf allen Kollektionen:

- ▶ size, includes, count, isEmpty, flatten
- ▶ Kollektionen werden immer flachgeklopft

- ▶ Set

- ▶ union, intersection,

- ▶ Bag

- ▶ union, intersection, count

- ▶ Sequence

- ▶ first, last, reverse, prepend, append

22 [24]

## Collection Types: Iteratoren

- ▶ Iteratoren: Funktionen höherer Ordnung

- ▶ Alle definiert über iterate :

```
coll-> iterate(elem: Type, acc: Type= expr | expr[elem, acc])
```

```
iterate(e: T, acc: T= v)
{
  acc= v;
  for (Enumeration e= c.elements(); e.hasMoreElements();){
    e= e.nextElement();
    acc.add(expr[e, acc]); // acc= expr[e, acc]
  }
  return acc;
}
```

- ▶ Iteratoren sind alle **strikt**

23 [24]

## Zusammenfassung

- ▶ Formale **Spezifikation** definieren das **Verhalten** eines Softwaresystems

- ▶ Verschiedene Ansätze:

- ▶ **Algebraische** Spezifikation (axiomatisch, abstrakt)
- ▶ **Modellbasierte** Spezifikation (konkret, konsistent)
- ▶ **Leichtgewichtige** Spezifikation (annotierter Code)

- ▶ In der Praxis oft hybride Ansätze (z.B. UML)

24 [24]

Formale Modellierung  
Vorlesung 12 vom 05.07.2015: Temporale Logik und Modellprüfung

Christoph Lüth

Universität Bremen

Sommersemester 2015

## Fahrplan

- ▶ Teil I: Formale Logik
- ▶ Teil II: Spezifikation und Verifikation
  - ▶ Formale Modellierung von Software
  - ▶ Temporale Logik und Modellprüfung
  - ▶ Zusammenfassung, Rückblick, Ausblick

## Organisatorisches

- ▶ Übung am Donnerstag fällt aus — Ersatztermin?

## Tagesmenu: Temporale Logik und Modellprüfung

- ▶ Modellierung des Programmes als **endliche Zustandsmaschine**
- ▶ **Abstraktion** über Zuständen, Zustandsübergang als **primäres** Konzept
- ▶ Temporale Logik: Logik über **Pfade** von Zustandsübergängen
  - ▶ Temporal im Sinne von *tempus fugit*

## Endliche Zustandsmaschine

### Definition (Finite State Machine, FSM)

Eine FSM ist  $\mathcal{M} = \langle \Sigma, \rightarrow \rangle$  mit

- ▶  $\Sigma$  eine **endliche** Menge von **Zuständen**, und
- ▶  $\rightarrow \subseteq \Sigma \times \Sigma$  eine **Zustandsübergangsrelation**, mit  $\rightarrow$  linkstotal:

$$\forall s \in \Sigma. \exists s' \in \Sigma. s \rightarrow s'$$

- ▶ Varianten dieser Definition: Anfangszustände; Zustandsvariablen oder benannte Zustandsübergänge
- ▶ NB: Kein Endzustand, und keine Ein/Ausgabe (Unterschied zu **Automaten**)
- ▶ Wenn  $\rightarrow$  eine Funktion ist (rechtseindeutig), dann ist die FSM **deterministisch**, ansonsten **nicht-deterministisch**.
- ▶ Jede nicht-deterministische FSM kann durch die Power-State-Konstruktion deterministisch gemacht werden.

## Einfaches Beispiel

- ▶ Getränkemaschine für Kaffee
- ▶ Nimmt 10c oder 20c Münzen
- ▶ Kleiner Kaffee 10c, großer Kaffee 20c
- ▶ Nimmt nicht mehr als zwei Münzen
- ▶ Geldrückgabe

## Linear Temporal Logic (LTL) and Pfade

- ▶ LTL ist die Logik über **Ausführungspfade** in einer FSM.
- ▶ Wir definieren erst Pfade, dann LTL-Formeln, dann eine Erfülltheitsrelation.

### Definition (Pfade)

Für eine FSM  $\mathcal{M} = \langle \Sigma, \rightarrow \rangle$  ist ein **Pfad** in  $\mathcal{M}$  eine (unendliche) Sequenz  $\langle s_1, s_2, s_3, \dots \rangle$  mit  $s_i \in \Sigma$  und  $s_i \rightarrow s_{i+1}$  für alle  $i$ .

- ▶ Notation: Sei  $p = \langle s_1, s_2, s_3, \dots \rangle$  ein Pfad, dann ist  $p_i \stackrel{\text{def}}{=} s_i$  (Selektion) und  $p^i \stackrel{\text{def}}{=} \langle s_i, s_{i+1}, \dots \rangle$  (Suffix ab Position  $i$ ).

## Lineare Temporale Logik (LTL)

$\phi ::=$	$\top \mid \perp \mid q$	— True, false, atomar
	$\mid \neg\phi \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \phi_1 \longrightarrow \phi_2$	— Aussagenlog. Formeln
	$\mid X\phi$	— Nächster Zustand
	$\mid F\phi$	— Irgendwann
	$\mid G\phi$	— Immer
	$\mid \phi_1 U \phi_2$	— Bis

- ▶ Präzedenzen: unäre Operatoren; dann  $U$ ; dann  $\wedge, \vee$ ; dann  $\longrightarrow$ .
- ▶ Eine atomare Formel  $p$  ist ein **Zustandsprädikat**. Andere (äquivalente) Möglichkeit: Zustände mit atomaren Prädikaten zu benennen (Kripke-Struktur).
- ▶ Andere Operatoren wie  $\phi R \psi$  (release) oder  $\phi W \psi$  (schwaches until).

## Erfüllung und Modelle für LTL

Die **Erfüllbarkeitsrelation** für einen Pfad  $p$  und eine LTL-Formel  $\phi$  ist induktiv wie folgt definiert:

$$\begin{array}{lll}
 p \models \top & & p \models \phi \wedge \psi \quad \text{gdw} \quad p \models \phi \text{ und } p \models \psi \\
 p \not\models \perp & & p \models \phi \vee \psi \quad \text{gdw} \quad p \models \phi \text{ oder } p \models \psi \\
 p \models q \quad \text{gdw} \quad q(p_1) & & p \models \phi \rightarrow \psi \quad \text{gdw} \quad \text{wenn } p \models \phi \\
 p \models \neg\phi \quad \text{gdw} \quad p \not\models \phi & & \text{dann } p \models \psi \\
 \\
 p \models X\phi \quad \text{gdw} \quad p^2 \models \phi & & \\
 p \not\models G\phi \quad \text{gdw} \quad \text{für alle } i \text{ gilt } p^i \not\models \phi & & \\
 p \models F\phi \quad \text{gdw} \quad \text{es gibt } i \text{ mit } p^i \models \phi & & \\
 p \models \phi U \psi \quad \text{gdw} \quad \text{es gibt } i \text{ mit } p^i \models \psi \text{ und für } j = 1, \dots, i-1, p^j \models \phi & & 
 \end{array}$$

### Definition (Modell einer LTL-Formel)

Eine FSM  $\mathcal{M}$  erfüllt eine LTL-Formel  $\phi$ ,  $\mathcal{M} \models \phi$ , gdw. jeder Pfad  $p$  in  $\mathcal{M}$  erfüllt.

9 [19]

## Äquivalenzen

### Definition (Äquivalenz)

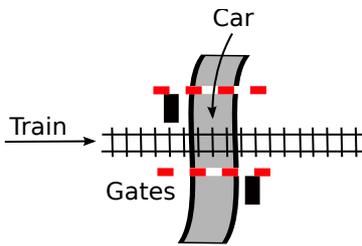
Zwei Formeln sind äquivalent,  $\phi \equiv \psi$  gdw. für alle FSM  $\mathcal{M}$  und Pfade  $p$  in  $\mathcal{M}$ ,  $p \models \phi \leftrightarrow p \models \psi$

► Es gelten aussagenlogische Tautologien z.B.  $\neg(\phi \vee \psi) \equiv \neg\phi \wedge \neg\psi$

$$\begin{array}{lll}
 F(\phi \vee \psi) \equiv F\phi \vee F\psi & \neg F\phi \equiv G(\neg\phi) & FGF\phi \equiv GF\phi \\
 G(\phi \wedge \psi) \equiv G\phi \wedge G\psi & \neg G\phi \equiv F(\neg\phi) & GFG\phi \equiv FG\phi \\
 \neg X\phi \equiv X(\neg\phi) & & \\
 \\
 XF\phi \equiv FX\phi & F\phi \equiv \phi \vee XF\phi & \\
 XG\phi \equiv GX\phi & G\phi \equiv \phi \wedge XG\phi & \\
 X(\phi U \psi) \equiv X\phi U X\psi & \phi U \psi \equiv \psi \vee (\phi \wedge X(\phi U \psi)) & 
 \end{array}$$

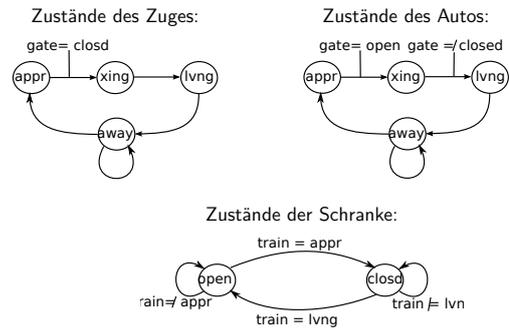
10 [19]

## Längeres Beispiel: der Bahnübergang



11 [19]

## Modellierung des Bahnübergangs



12 [19]

## Die FSM

► Zustände sind eine endliche Abbildung der Variablen  $Car$ ,  $Train$ ,  $Gate$  auf Wertebereiche:

$$\begin{array}{l}
 \Sigma_{Car} = \{appr, xing, lvng, away\} \\
 \Sigma_{Train} = \{appr, xing, lvng, away\} \\
 \Sigma_{Gate} = \{open, clsd\}
 \end{array}$$

oder ein Tripel  $S \in \Sigma = \Sigma_{Car} \times \Sigma_{Train} \times \Sigma_{Gate}$ .

► Zustandsübergang **komponentenweise**, bspw.:

$$\begin{array}{l}
 \langle away, open, away \rangle \rightarrow \langle appr, open, away \rangle \\
 \langle appr, open, away \rangle \rightarrow \langle xing, open, away \rangle \\
 \dots
 \end{array}$$

13 [19]

## Bahnübergang — Formalisierung von Eigenschaften

► Bahn und Auto überqueren den Übergang nie zur selben Zeit:

$$G\neg(car = xing \wedge train = xing)$$

► Ein Auto kann den Übergang immer wieder verlassen:

$$G(car = xing \rightarrow F(car = lvng))$$

► Ein annähernder Zug darf irgendwann den Bahnübergang passieren:

$$G(train = appr \rightarrow F(train = xing))$$

► Es gibt Autos, die den Bahnübergang passieren:

$$F(car = xing) \text{ ist etwas anderes!}$$

► Nicht in LTL auszudrücken!

14 [19]

## Computational Tree Logic (CTL)

► Grenzen der LTL: Quantifikation über **Pfaden**

► z.B. Existenz eines Pfades mit einer bestimmten Eigenschaft

► Computational Tree Logic (CTL): Erweiterung der LTL um existentielle/universelle Quantoren über modalen Pfadoperatoren.

► Modale Operatoren: die Zustandsübergänge betreffend

► Beispiel:  $AF p$ ,  $EG q$ ,  $A[p U q]$

► Name: Pfade im **Berechnungsbaum** durch Auffalten der FSM.

► Beispiel Berechnungsbäume für die Getränkemaschine

15 [19]

## LTL und CTL

► CTL ist ausdrucksstärker als LTL, aber das gilt auch **anders herum!**

► D.h. es gibt Eigenschaften, die in LTL ausgedrückt werden können, aber nicht in CTL.

► Beispiel: in allen Pfaden, in denen  $p$  auftritt, tritt auch  $q$  auf.

► LTL:  $Fp \rightarrow Fq$

► CTL: **Weder**  $AF p \rightarrow AF q$  **noch**  $AG(p \rightarrow AF q)$

► Die Logik **CTL\*** kombiniert die Mächtigkeit von LTL und CTL.

16 [19]

## Modellprüfung (Model-Checking)

- ▶ Das **Model-Checking Problem**:  
Gegeben Modell  $\mathcal{M}$  und Eigenschaft  $\phi$ , gilt  $\mathcal{M} \models \phi$ ?
- ▶ Das Grundproblem beim Model-Checking ist **Zustandsexplosion**.
  - ▶ Eine typische 32-Bit Ganzzahlvariable hat über 4 Mrd. Zustände!
- ▶ Die Theorie bietet wenig Anlass zu Hoffnung:

### Theorem (Komplexität von Modellprüfung)

- (i) Model-Checking für LTL ohne  $U$  ist NP-vollständig.
- (ii) Model-Checking für LTL ist PSPACE-vollständig.
- (iii) Model-Checking für CTL ist EXPTIME-vollständig.

- ▶ Gute Nachricht: wenigstens **entscheidbar**
  - ▶ Schlüsseltechnik: **Zustandsabstraktion** und **Zustandskompression**

17 [19]

## Model-Checking Werkzeuge

- ▶ **NuSMV2** (Edmund Clarke, Ken McMillan)
  - ▶ Web Seite: <http://nusmv.fbk.eu/>
- ▶ **Spin** (Gerard Holzmann)
  - ▶ Web Seite: <http://spinroot.com/>
- ▶ NuSMV vs. Spin:
  - ▶ Spin (Promela) ist näher an einer Programmiersprache
  - ▶ NuSMV unterstützt auch CTL

18 [19]

## Zusammenfassung

- ▶ Temporale Logik: **Pfade** in **Zustandsautomaten**
- ▶ Aussagenlogik plus modale Operatoren:
  - ▶ LTL — linear über einen Pfad: X, G, F, U
  - ▶ CTL — verzweigend: AX, EX, AG, EG, AF, EF, A[ U ], E[ U ]
  - ▶ LTL für **Sicherheitseigenschaften**, CTL für **Verfügbarkeit**.
- ▶ In der Praxis: LTL/CTL für Modellprüfung (**model checking**)
  - ▶ Modellierung des Systems als FSM  $\mathcal{M}$ , Eigenschaften als LTL/CTL-Formel  $\phi$ , Überprüfung ob  $\mathcal{M} \models \phi$ .
  - ▶ **Entscheidbar**, aber mit hoher Komplexität (**Zustandsexplosion**)
- ▶ Model-Checker wie NuSMV entscheiden das Model-Checking-Problem
  - ▶ Bei negativer Antwort **Gegenbeispiel**.
  - ▶ Vertrauenswürdigkeit: bei positiver Antwort? Wie gut ist das Modell?

19 [19]

Formale Modellierung  
 Vorlesung 13 vom 13.07.2015: Zusammenfassung, Rückblick,  
 Ausblick

Christoph Lüth  
 Universität Bremen  
 Sommersemester 2015

## Fahrplan

- ▶ Teil I: Formale Logik
- ▶ Teil II: Spezifikation und Verifikation
  - ▶ Formale Modellierung von Software
  - ▶ Temporale Logik und Modellprüfung
  - ▶ Zusammenfassung, Rückblick, Ausblick

## Heute in diesem Theater

- ▶ Zusammenfassung und Rückblick
- ▶ Formale Modellierung und Formale Methoden in der Praxis
- ▶ ... und jetzt?

## Unsere Reise durch die Logik

	Entscheidbar?	Vollständig?	Konsistent?	Werkzeuge (Beweiser)
Aussagenlogik	J	J	J	SAT-Solver
Presburger	J	J	J	SMT-Beweiser: Z3, CVC
Peano-Ar.	N	J	J	
FOL	N	J	J	ATPs: SPASS, Vampire
FOL + Induktion	N	N	J	KIV, KeY, Inka
HOL	N	N	J	ITPs: Isabelle, Coq, PVS

## Aussagenlogik

- ▶ Formeln und Bedeutung
- ▶ Beweisprinzipien:
  - ▶ Wahrheitstabelle, natürliches Schließen, Äquivalenzumformung, Resolution
- ▶  $\models P$  vs.  $\vdash P$
- ▶ Warum ist Aussagenlogik entscheidbar?

## Prädikatenlogik

- ▶ Formeln und Bedeutung
- ▶ Welche Beweisprinzipien?
- ▶ Besonderheit beim natürlichen Schließen?
- ▶ Warum ist Prädikatenlogik vollständig?
- ▶ ... und warum nicht mehr entscheidbar?

## Induktion und Logik höherer Stufe

- ▶ Wie axiomatisieren wir die natürlichen Zahlen?
- ▶ Wie sehen Modelle der natürlichen Zahlen aus (und was ist ein Nichtstandardmodell)?
- ▶ Was ist der Unterschied zwischen natürlicher Induktion und wohlfundierter Induktion?
- ▶ Wie funktioniert der Beweis für die Unvollständigkeitssätze?
- ▶ Warum ist Logik höherer Stufe nicht mehr vollständig?
- ▶ Was ist eine konservative Erweiterung?

## Die Gödelschen Unvollständigkeitssätze

- ▶ Kerntechnik: Gödelkodierung
- ▶ Kodierung von Termen, Formeln, Ableitung in Peano-Arithmetik (**PA**)
  - ▶ Warum Peano?
- ▶ Beweisidee: Logik, in der **PA** formalisierbar ist, kann potenziell über sich selbst reden.
- ▶ Technisch:
  - ▶  $\text{Thm}(f)$  gdw.  $[f]$  ist ein Theorem
  - ▶ Fixpunktsatz: zu Formel  $\varphi(x)$  gibt es  $\psi$  so dass  $\vdash \varphi([ \psi ])$   $\leftrightarrow \psi$
  - ▶ Gödel-sentence:  $\varphi(x) \stackrel{\text{def}}{=} \neg \text{Thm}(x)$

## UML

- ▶ Was ist formal an der UML?
  - ▶ Klassendiagramme, Zustands- und Sequenzdiagramme
- ▶ Was ist OCL?
  - ▶ Eine Sprache zur Einschränkung der Modellklasse
  - ▶ Woraus besteht die OCL?
  - ▶ Welche Logik benutzt die OCL?
  - ▶ Welche Typen kennt die OCL?

9 [20]

## Temporallogik

- ▶ Was sind temporale Logiken?
- ▶ Welche Operatoren haben LTL und CTL? Was ist der Unterschied?
- ▶ Wie ist Gültigkeit für LTL/CTL definiert?
- ▶ Ist LTL/CTL entscheidbar? ... vollständig?
- ▶ Was ist das Modelchecking-Problem?
- ▶ Was ist das Problem beim Modelchecking?

10 [20]

## Modellierung, formale Modellierung, Programme und formale Methoden

- ▶ Formale Logik — Mathematik
- ▶ Programme und Berechenbarkeit
- ▶ Formale Methoden: Anwendung der Methoden der Logik auf Programme
- ▶ Automatisierte Beweisverfahren: Anwendung von Programmen auf die Logik

11 [20]

## Formale Modellierung: Geschichtlicher Rückblick

- ▶ Gottlob Frege (1848– 1942)
  - ▶ 'Begriffsschrift, eine der arithmetischen nachgebildete Formelsprache des reinen Denkens' (1879)
- ▶ Georg Cantor (1845– 1918), Bertrand Russel (1872– 1970), Ernst Zermelo (1871– 1953)
  - ▶ Einfache Mengenlehre: inkonsistent (Russel's Paradox)
  - ▶ Axiomatische Mengenlehre: Zermelo-Fränkel
- ▶ David Hilbert (1862– 1943)
  - ▶ Hilbert's Programm: 'mechanisierte' Beweistheorie
- ▶ Kurt Gödel (1906– 1978)
  - ▶ Vollständigkeitssatz, Unvollständigkeitssätze

12 [20]

## Formale Methoden: Geschichtlicher Rückblick

- ▶ Ziel: Methoden, um die Korrektheit von Programmen sicherzustellen
- ▶ Erste Ansätze: Alan Turing (1949)
- ▶ Robert Floyd und CAR Hoare: Floyd-Hoare-Kalkül (1969/1971)
- ▶ Korrektheit durch Konstruktion: Dijkstra, Gries und andere (1972 ff)
- ▶ Problem: sehr viele, größtenteils triviale Beweise

13 [20]

## Automatisches Theorembeweisen

- ▶ Automatisches Beweisen: Wurzeln in der Mathematik, ursprünglich Teil der KI:
  - ▶ Termersetzung (Thue, Semi-Thue-Systeme: 1910; Schönfinkel, Kombinatorlogik: 1930)
  - ▶ SAT (Davis-Putnam, 1960; Davis, Logemann and Loveland, 1962)
  - ▶ Resolution (Robinson, 1965: Unifikation)
- ▶ Früher Enthusiasmus, dann Ernüchterung; durch leistungsfähigere Rechner und Algorithmen späte Blüte.

14 [20]

## Formale Modellierung und Formale Methoden

- ▶ Das LCF System (Robin Milner: Stanford LCF, Cambridge LCF, Edinburgh LCF, ab 1972)
  - ▶ Entwickelt als "Programmbeweissystem"
  - ▶ "Stammvater" vieler moderner Beweise: Isabelle, Coq, HOL4, HOL light
- ▶ NQTHM (Boyer-Moore, ab 1971)
  - ▶ Heute: ACL-2
- ▶ Zwei Schulen: getypt vs. ungetypt, expansiv vs. Beweisobjekte

15 [20]

## Formale Methoden

- ▶ Stetiger Fortschritt auf vielen Ebenen
- ▶ Statische Programmanalyse (eg. WCET, AbsInt)
- ▶ Modellbasierte Entwicklung (insbes. SCADE und andere)
- ▶ Beweisbasierte Verfahren (Microsoft's SLAM, B-Methode)
- ▶ Hardwareverifikation: Intel, AMD, Infineon, ...
- ▶ L4.verified

16 [20]

## Formale Modellierung in der Mathematik

- ▶ Offene mathematische Probleme und ihre Lösung:
  - ▶ Perelman und Poincaré; Andrew Wiles und Fermat's letztes Theorem; die Riemannsche Vermutung
  - ▶ Beweise werden zunehmend komplexer
- ▶ Rechnergestützte Beweisverfahren:
  - ▶ Vierfarbenproblem (Appel-Haken, 1970)
- ▶ Vollständig formalisierte Beweisverfahren:
  - ▶ Vierfarbenproblem in Coq (Gonthier, 2005)
  - ▶ Die Keplersche Vermutung und Flyspeck (Hales, 2002– 2014)

17 [20]

## Stand der Kunst

- ▶ Formale Modellierung Stand der Kunst
  - ▶ Luft- und Raumfahrt
  - ▶ Automotive
  - ▶ ... **nicht** im Finanzbereich!
- ▶ In den kommenden Jahren: weitere Anwendungsgebiete
  - ▶ Spezialisierte Techniken für bestimmte Anwendungsfälle (DSLs)

18 [20]

## ... und jetzt?

- ▶ Besuchen Sie auch: Formale Methoden der Softwaretechnik (Master-Wahlveranstaltung)
- ▶ Bachelor/Diplomarbeiten am DFKI/AGRA
- ▶ Andere Gruppen an der Uni Bremen

19 [20]

Tschüß!



20 [20]