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

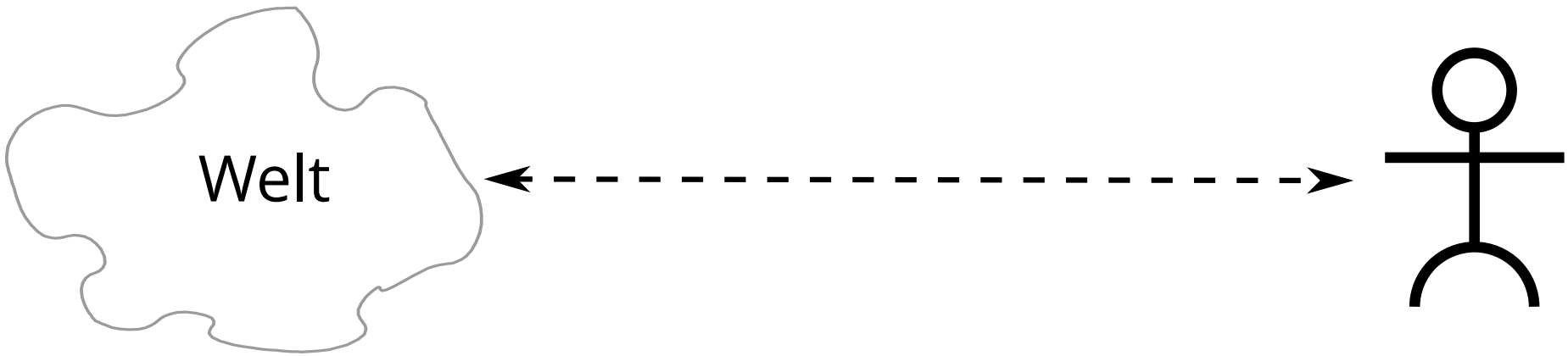
- Webseite:

http://www.informatik.uni-bremen.de/~cxl/lehre/fm.ss15

Warum formale Modellierung?

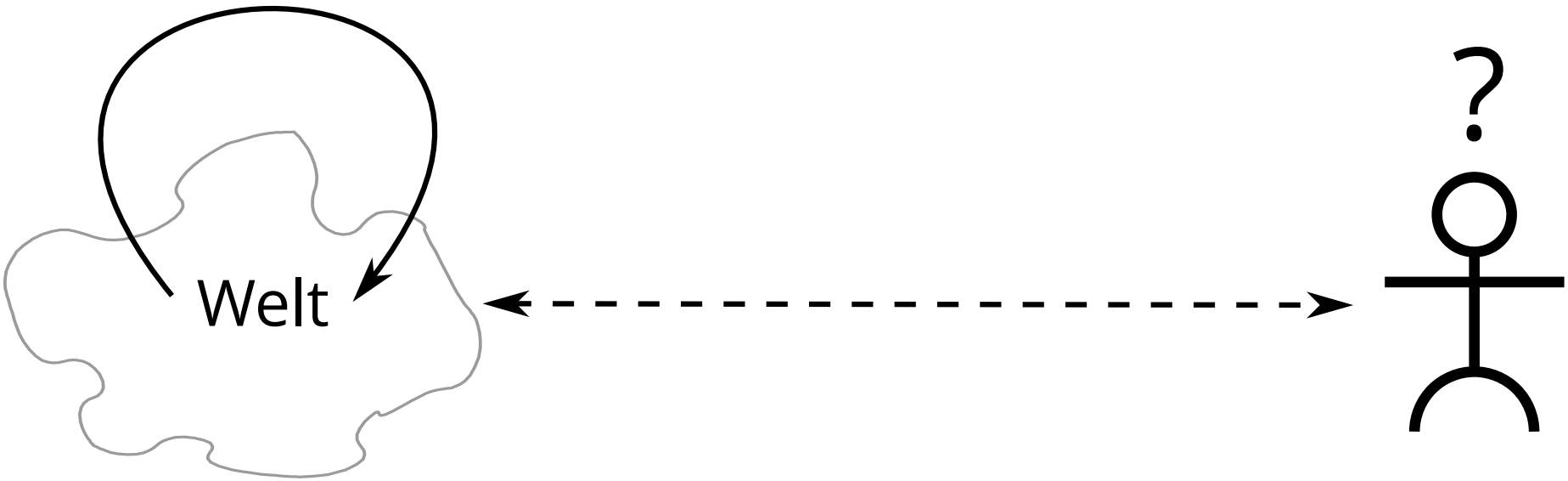
Die Vasa, 10. August 1628

Modellierung — Das Prinzip



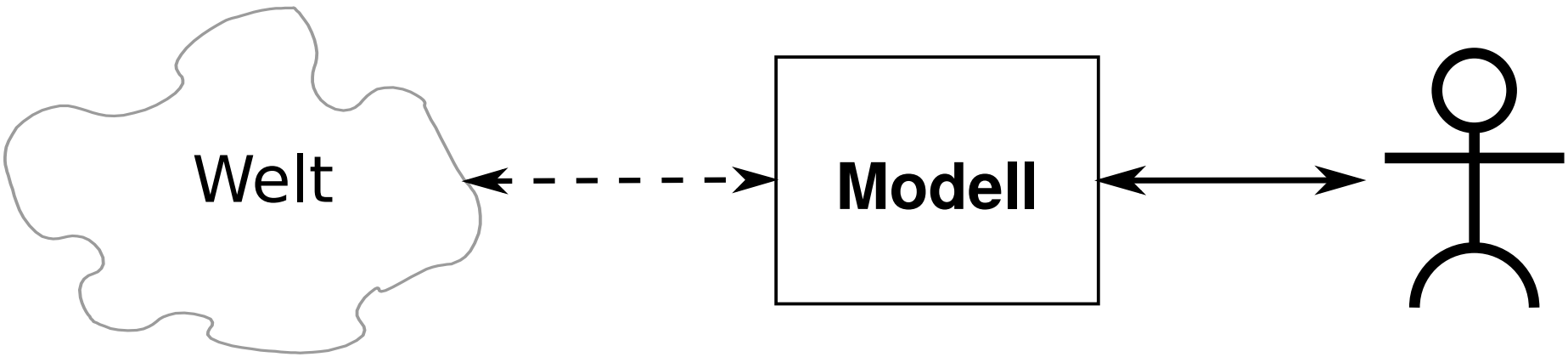
- Grundlegendes Prinzip der Naturwissenschaften

Modellierung — Das Prinzip



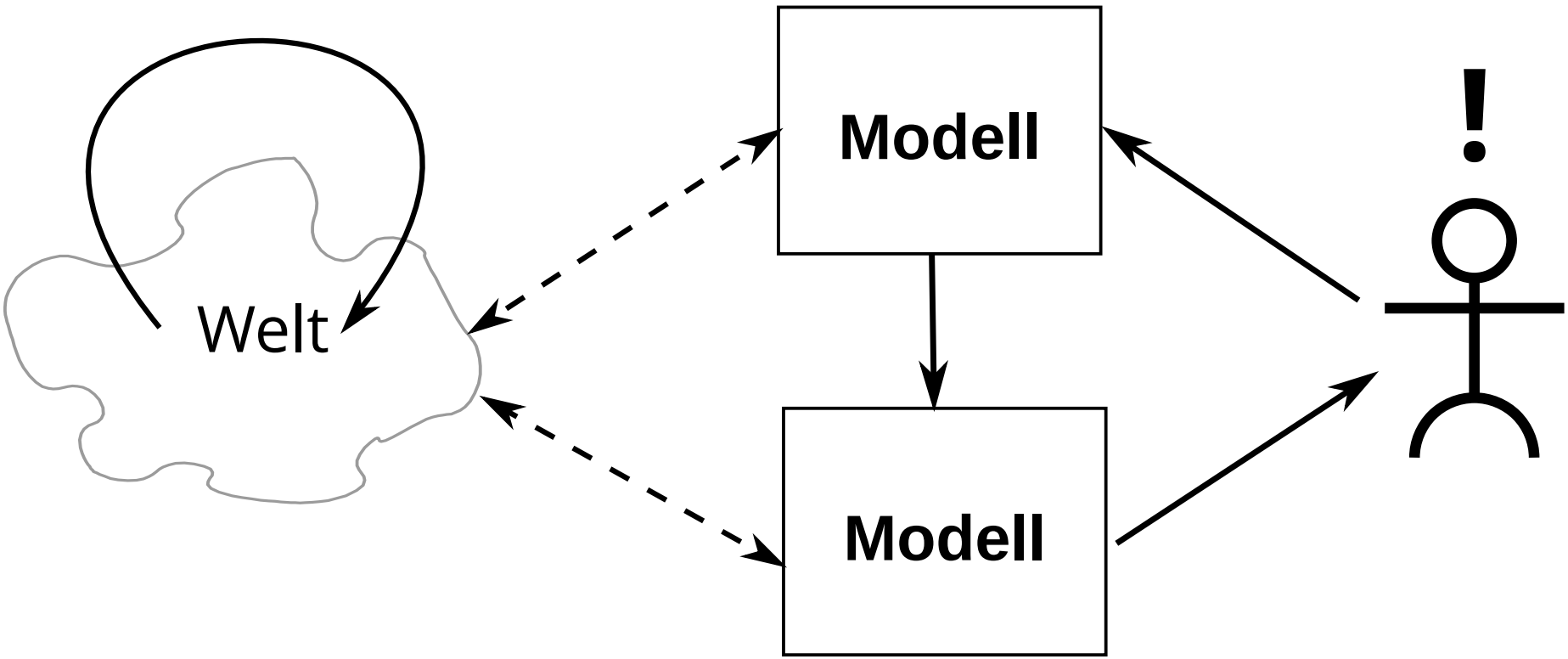
- Grundlegendes Prinzip der Naturwissenschaften

Modellierung — Das Prinzip



- Grundlegendes Prinzip der Naturwissenschaften

Modellierung — Das Prinzip



- Grundlegendes Prinzip der Naturwissenschaften

Modellierung — Beispiele

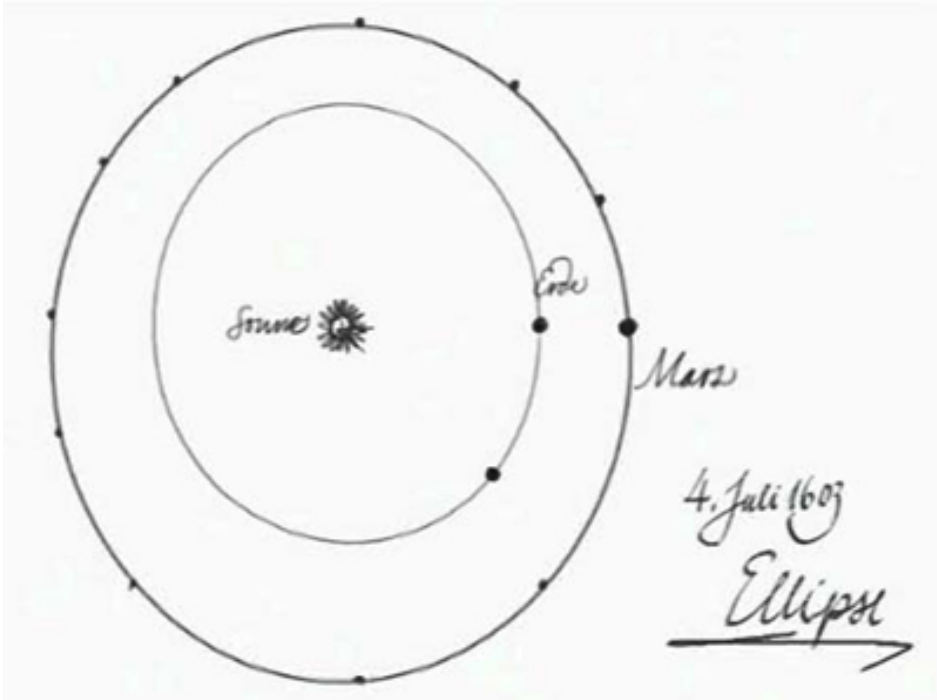


$$2Mg + O_2 \longrightarrow 2MgO$$

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?

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

Lernziele

1. Modellierung — Formulierung von Eigenschaften

Lernziele

1. Modellierung — Formulierung von Eigenschaften

2. Beweis — Formaler Beweis der Eigenschaften

Lernziele

1. Modellierung — Formulierung von Eigenschaften

2. Beweis — Formaler Beweis der Eigenschaften

3. Spezifikation und Verifikation — Eigenschaften von Programmen

Themen

- Formale Logik:
  - Aussagenlogik ($A \wedge B$, $A \longrightarrow B$), 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

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`

Formale Logik

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

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

Grundbegriffe der formalen Logik

- Ableitbarkeit $\mathcal{T}h \vdash P$
  - Syntaktische Folgerung
- Gültigkeit $\mathcal{T}h \models P$
  - Semantische Folgerung
- Klassische Logik: $P \vee \neg P$
- Entscheidbarkeit
  - Aussagenlogik
- Konsistenz: $\mathcal{T}h \nvdash \bot$
  - Nicht alles ableitbar
- Vollständigkeit: jede gültige Aussage ableitbar
  - Prädikatenlogik erster Stufe

Unvollständigkeit

- Gödels 1. Unvollständigkeitssatz:
  - Jede Logik, die Peano-Arithmetik formalisiert, ist entweder inkonsistent oder unvollständig.

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.

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 spezifierbar.
  - Spezifikationssprachen immer unvollständig (oder uninteressant).

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 spezifierbar.
  - Spezifikationssprachen immer unvollständig (oder uninteressant).
  - Mit anderen Worten: Es bleibt spannend.

Nächste Woche

- Aussagenlogik
- Erstes Übungsblatt

Formale Modellierung
Vorlesung 2 vom 20.04.15: Aussagenlogik und natürliches Schließen

Christoph Lüth

Universität Bremen

Sommersemester 2015

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.

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
  - Berechungsmodelle (Models of Computation)
  - Die Unvollständigkeitssätze von Gödel
- Teil II: Spezifikation und Verifikation

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

Formale Logik

- Ziel: Formalisierung von Folgerungen wie
  - Wenn es regnet, wird die Straße nass.

Formale Logik

- Ziel: Formalisierung von Folgerungen wie
  - Wenn es regnet, wird die Straße nass.
  - Es regnet.

Formale Logik

- Ziel: Formalisierung von Folgerungen wie
  - Wenn es regnet, wird die Straße nass.
  - Es regnet.
  - Also ist die Straße nass.

Formale Logik

- Ziel: Formalisierung von Folgerungen wie
  - Wenn es regnet, wird die Straße nass.
  - Es regnet.
  - Also ist die Straße nass.
  - Nachts ist es dunkel.

Formale Logik

- Ziel: Formalisierung von Folgerungen wie
  - Wenn es regnet, wird die Straße nass.
  - Es regnet.
  - Also ist die Straße nass.
  - Nachts ist es dunkel.
  - Es ist hell.

Formale Logik

- Ziel: Formalisierung von Folgerungen wie
  - Wenn es regnet, wird die Straße nass.
  - Es regnet.
  - Also ist die Straße nass.
  - Nachts ist es dunkel.
  - Es ist hell.
  - 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.

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

- Beispielableitung: $\heartsuit$

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
$\dfrac{\diamondsuit}{\clubsuit}\ \alpha$	$\dfrac{\diamondsuit}{\spadesuit}\ \beta$	$\dfrac{\clubsuit \quad \spadesuit}{\heartsuit}\ \gamma$	$\dfrac{}{\diamondsuit}\ \delta$

- Beispielableitung: $\heartsuit$

Aussagenlogik

- Sprache $\mathcal{P}rop$ gegeben durch:
  1. Variablen (Atome) $V \subseteq \mathcal{P}rop$ (Menge V gegeben)
  2. $\bot \in \mathcal{P}rop$
  3. Wenn $\phi, \psi \in \mathcal{P}rop$, dann
     - $\phi \wedge \psi \in \mathcal{P}rop$
     - $\phi \vee \psi \in \mathcal{P}rop$
     - $\phi \longrightarrow \psi \in \mathcal{P}rop$
     - $\phi \longleftrightarrow \psi \in \mathcal{P}rop$
  4. Wenn $\phi \in \mathcal{P}rop$, dann $\neg\phi \in \mathcal{P}rop$.
- NB. Präzedenzen: $\neg$ vor $\wedge$ vor $\vee$ vor $\longrightarrow, \longleftrightarrow$

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
  - Sequenzenkalkül
  - Andere (Hilbert-Kalkül, gleichungsbasierte Kalküle, etc.)

Semantik

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

Definition (Semantik aussagenlogischer Formeln)

Für Valuation $v : V \to \{0, 1\}$ ist $[\![\cdot]\!]_v : \mathit{Prop} \to \{0, 1\}$ definiert als

$$
\begin{aligned}
[\![w]\!]_v &= v(w) \quad (\text{mit } w \in V) \\
[\![\bot]\!]_v &= 0 \\
[\![\phi \wedge \psi]\!]_v &= \min([\![\phi]\!]_v, [\![\psi]\!]_v) \\
[\![\phi \vee \psi]\!]_v &= \max([\![\phi]\!]_v, [\![\psi]\!]_v) \\
[\![\phi \longrightarrow \psi]\!]_v &= 0 \iff [\![\phi]\!]_v = 1 \text{ und } [\![\psi]\!]_v = 0 \\
[\![\phi \longleftrightarrow \psi]\!]_v &= 1 \iff [\![\phi]\!]_v = [\![\psi]\!]_v \\
[\![\neg\phi]\!]_v &= 1 - [\![\phi]\!]_v
\end{aligned}
$$

Semantische Gültigkeit und Folgerung

- Semantische Gültigkeit: $\models \phi$

$$\models \phi \text{ gdw. } [\![\phi]\!]_v = 1 \text{ für alle } v$$

- Semantische Folgerung: sei $\Gamma \subseteq \mathit{Prop}$, dann

$$\Gamma \models \psi \text{ gdw. } [\![\psi]\!]_v = 1 \text{ wenn } [\![\phi]\!]_v = 1 \text{ für alle } \phi \in \Gamma$$

Beweisen mit semantischer Folgerung

- Die Wahrheitstabellenmethode:
  - Berechne $[\![\phi]\!]_v$ für alle Möglichkeiten für v
- Beispiel: $\models (\phi \longrightarrow \psi) \longleftrightarrow (\neg\psi \longrightarrow \neg\phi)$

ϕ	ψ	$\phi \longrightarrow \psi$	$\neg\psi$	$\neg\phi$	$\neg\psi \longrightarrow \neg\phi$	$(\phi \longrightarrow \psi) \longleftrightarrow (\neg\psi \longrightarrow \neg\phi)$
0	0	1	1	1	1	1

Beweisen mit semantischer Folgerung

- Die Wahrheitstabellenmethode:
  - Berechne $[\![\phi]\!]_v$ für alle Möglichkeiten für v
- Beispiel: $\models (\phi \longrightarrow \psi) \longleftrightarrow (\neg\psi \longrightarrow \neg\phi)$

ϕ	ψ	$\phi \longrightarrow \psi$	$\neg\psi$	$\neg\phi$	$\neg\psi \longrightarrow \neg\phi$	$(\phi \longrightarrow \psi) \longleftrightarrow (\neg\psi \longrightarrow \neg\phi)$
0	0	1	1	1	1	1
0	1	1	0	1	1	1

Beweisen mit semantischer Folgerung

- Die Wahrheitstabellenmethode:
  - Berechne $[\![\phi]\!]_v$ für alle Möglichkeiten für v
- Beispiel: $\models (\phi \longrightarrow \psi) \longleftrightarrow (\neg\psi \longrightarrow \neg\phi)$

ϕ	ψ	$\phi \longrightarrow \psi$	$\neg\psi$	$\neg\phi$	$\neg\psi \longrightarrow \neg\phi$	$(\phi \longrightarrow \psi) \longleftrightarrow (\neg\psi \longrightarrow \neg\phi)$
0	0	1	1	1	1	1
0	1	1	0	1	1	1
1	0	0	1	0	0	1

Beweisen mit semantischer Folgerung

- Die Wahrheitstabellenmethode:
  - Berechne $[\![\phi]\!]_v$ für alle Möglichkeiten für v
- Beispiel: $\models (\phi \longrightarrow \psi) \longleftrightarrow (\neg\psi \longrightarrow \neg\phi)$

ϕ	ψ	$\phi \longrightarrow \psi$	$\neg\psi$	$\neg\phi$	$\neg\psi \longrightarrow \neg\phi$	$(\phi \longrightarrow \psi) \longleftrightarrow (\neg\psi \longrightarrow \neg\phi)$
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

Beweisen mit semantischer Folgerung

- Die Wahrheitstabellenmethode:
  - Berechne $[\![\phi]\!]_v$ für alle Möglichkeiten für v
- Beispiel: $\models (\phi \longrightarrow \psi) \longleftrightarrow (\neg\psi \longrightarrow \neg\phi)$

ϕ	ψ	$\phi \longrightarrow \psi$	$\neg\psi$	$\neg\phi$	$\neg\psi \longrightarrow \neg\phi$	$(\phi \longrightarrow \psi) \longleftrightarrow (\neg\psi \longrightarrow \neg\phi)$
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

Syntakische Gültigkeit: Natürliches Schließen

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

$$\frac{\diamondsuit}{\clubsuit}\,\alpha \qquad \frac{\diamondsuit}{\spadesuit}\,\beta \qquad \frac{\clubsuit \quad \spadesuit}{\heartsuit}\,\gamma \qquad \frac{\begin{array}{c}[\diamondsuit]\\ \vdots \\ \heartsuit\end{array}}{\heartsuit}\,\delta'$$

- Beispielableitung: $\heartsuit$

Natürliches Schließen (ND) für Aussagenlogik

- Vorgehensweise:
  1. Erst Kalkül nur für $\wedge, \longrightarrow, \bot$
  2. Dann Erweiterung auf alle Konnektive.
- Für jedes Konnektiv: Einführungs- und Eliminationsregel
- NB: konstruktiver Inhalt der meisten Regeln

Natürliches Schließen — Die Regeln

$$\frac{\phi \quad \psi}{\phi \wedge \psi}\ \wedge I \qquad \frac{\phi \wedge \psi}{\phi}\ \wedge E_L \qquad \frac{\phi \wedge \psi}{\psi}\ \wedge E_R$$

$$\frac{\begin{array}{c}[\phi]\\ \vdots \\ \psi\end{array}}{\phi \longrightarrow \psi}\ \longrightarrow I \qquad \frac{\phi \quad \phi \longrightarrow \psi}{\psi}\ \longrightarrow E$$

$$\frac{\bot}{\phi}\ \bot \qquad \frac{\begin{array}{c}[\phi \longrightarrow \bot]\\ \vdots \\ \bot\end{array}}{\phi}\ \text{raa}$$

Die fehlenden Konnektive

- Einführung als Abkürzung:

$$\neg\phi \stackrel{def}{=} \phi \longrightarrow \bot$$

$$\phi \vee \psi \stackrel{def}{=} \neg(\neg\phi \wedge \neg\psi)$$

$$\phi \longleftrightarrow \psi \stackrel{def}{=} (\phi \longrightarrow \psi) \wedge (\psi \longrightarrow \phi)$$

- Ableitungsregeln als Theoreme.

Die fehlenden Schlußregeln

$$\dfrac{\begin{array}{c}[\phi]\\ \vdots\\ \bot\end{array}}{\neg\phi}\ \neg I \qquad \dfrac{\phi \quad \neg\phi}{\bot}\ \neg E$$

$$\dfrac{\phi}{\phi \vee \psi}\ \vee I_L \qquad \dfrac{\psi}{\phi \vee \psi}\ \vee I_R \qquad \dfrac{\phi \vee \psi \qquad \begin{array}{c}[\phi]\\ \vdots\\ \sigma\end{array} \qquad \begin{array}{c}[\psi]\\ \vdots\\ \sigma\end{array}}{\sigma}\ \vee E$$

$$\dfrac{\phi \longrightarrow \psi \quad \psi \longrightarrow \phi}{\phi \longleftrightarrow \psi}\ \longleftrightarrow I \qquad \dfrac{\phi \quad \phi \longleftrightarrow \psi}{\psi}\ \longleftrightarrow E_L \qquad \dfrac{\psi \quad \phi \longleftrightarrow \psi}{\phi}\ \longleftrightarrow E_R$$

Zusammenfassung

- Formale Logik formalisiert das (natürlichsprachliche) Schlußfolgern
- Logik: Formeln, Semantik, Schlußregeln (Kalkül)
- Aussagenlogik: Aussagen mit $\wedge$, $\longrightarrow$, $\bot$
  - $\neg$, $\vee$, $\longleftrightarrow$ als abgeleitete Operatoren
- Semantik von Aussagenlogik $[\![\cdot]\!]_v : \mathit{Prop} \to \{0, 1\}$
- 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
  - Berechungsmodelle (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

- $\mathcal{P}rop$ bildet eine Boolesche Algebra:

$$\models (\phi \vee \psi) \vee \sigma \longleftrightarrow \phi \vee (\psi \vee \sigma)$$
$$\models (\phi \wedge \psi) \wedge \sigma \longleftrightarrow \phi \wedge (\psi \wedge \sigma)$$
$$\models \phi \vee \psi \longleftrightarrow \psi \vee \phi$$
$$\models \phi \wedge \psi \longleftrightarrow \psi \wedge \phi$$
$$\models \phi \vee (\psi \wedge \sigma) \longleftrightarrow (\phi \vee \psi) \wedge (\phi \vee \sigma)$$
$$\models \phi \wedge (\psi \vee \sigma) \longleftrightarrow (\phi \wedge \psi) \vee (\phi \wedge \sigma)$$
$$\models \neg(\phi \vee \psi) \longleftrightarrow \neg\phi \wedge \neg\psi$$
$$\models \neg(\phi \wedge \psi) \longleftrightarrow \neg\phi \vee \neg\psi$$
$$\models \phi \vee \phi \longleftrightarrow \phi$$
$$\models \phi \wedge \phi \longleftrightarrow \phi$$
$$\models \neg\neg\phi \longleftrightarrow \phi$$

Eigenschaften der Aussagenlogik

- Rechnen in $\mathcal{P}rop$:
  - Substitutivität:
    wenn $\models \phi_1 \longleftrightarrow \phi_2$, dann $\models \psi\left[\begin{smallmatrix}\phi_1\\ p\end{smallmatrix}\right] \longleftrightarrow \psi\left[\begin{smallmatrix}\phi_2\\ p\end{smallmatrix}\right]$ für Atom p.
  - Sei $\phi \approx \psi$ gdw. $\models \phi \longleftrightarrow \psi$, dann ist $\approx$ eine Äquivalenzrelation
- Damit: algebraisches Umformen als Beweisprinzip
  - Beispiele: $\models (\phi \longrightarrow (\psi \longrightarrow \sigma)) \longleftrightarrow (\phi \wedge \psi \longrightarrow \sigma)$
    $\models \phi \longrightarrow \psi \longrightarrow \phi$
- Anwendung: konjunktive und disjunktive Normalformen (CNF/DNF)

Eigenschaften der Aussagenlogik

- Operatoren durch andere definierbar:

$$\models (\phi \longleftrightarrow \psi) \longleftrightarrow (\phi \longrightarrow \psi) \wedge (\psi \longrightarrow \phi)$$
$$\models (\phi \longrightarrow \psi) \longleftrightarrow (\neg\phi \vee \psi)$$
$$\models \phi \vee \psi \longleftrightarrow (\neg\phi \longrightarrow \psi)$$
$$\models \phi \vee \psi \longleftrightarrow \neg(\neg\phi \wedge \neg\psi)$$
$$\models \phi \wedge \psi \longleftrightarrow \neg(\neg\phi \vee \neg\psi)$$
$$\models \neg\phi \longleftrightarrow (\phi \longrightarrow \bot)$$
$$\models \bot \longleftrightarrow (\phi \wedge \neg\phi)$$
$$\models \top \longleftrightarrow (\phi \vee \neg\phi)$$

Eigenschaften der Aussagenlogik

- Operatoren durch andere definierbar:

$$\models (\phi \longleftrightarrow \psi) \longleftrightarrow (\phi \longrightarrow \psi) \wedge (\psi \longrightarrow \phi)$$
$$\models (\phi \longrightarrow \psi) \longleftrightarrow (\neg\phi \vee \psi)$$
$$\models \phi \vee \psi \longleftrightarrow (\neg\phi \longrightarrow \psi)$$
$$\models \phi \vee \psi \longleftrightarrow \neg(\neg\phi \wedge \neg\psi)$$
$$\models \phi \wedge \psi \longleftrightarrow \neg(\neg\phi \vee \neg\psi)$$
$$\models \neg\phi \longleftrightarrow (\phi \longrightarrow \bot)$$
$$\models \bot \longleftrightarrow (\phi \wedge \neg\phi)$$
$$\models \top \longleftrightarrow (\phi \vee \neg\phi)$$

- $(\wedge, \neg)$ und $(\vee, \bot)$ 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)

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 Γ von Aussagen konsistent gdw. $\Gamma \nvdash \bot$

Lemma 3 (Charakterisierung von Konsistenz)

Folgende Aussagen sind äquivalent:

(i) Γ *konsistent*

(ii) *Es gibt kein* ϕ *so dass* $\Gamma \vdash \phi$ *und* $\Gamma \vdash \neg\phi$

(iii) *Es gibt ein* ϕ *so dass* $\Gamma \nvdash \phi$

Konsistenz

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

Definition 2 (Konsistenz)

Menge Γ von Aussagen konsistent gdw. $\Gamma \nvdash \bot$

Lemma 3 (Charakterisierung von Konsistenz)

Folgende Aussagen sind äquivalent:

(iv) Γ *inkonsistent* ($\Gamma \vdash \bot$)

(v) *Es gibt ein* ϕ *so dass* $\Gamma \vdash \phi$ *und* $\Gamma \vdash \neg\phi$

(vi) *Für alle* ϕ, $\Gamma \vdash \phi$

Maximale Konsistenz

- Wenn es v gibt so dass $[\![\psi]\!]_v = 1$ für $\psi \in \Gamma$, dann Γ konsistent.

Definition 4 (Maximale Konsistenz)

Γ maximal konsistent gdw.

(i) Γ konsistent, und

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

Lemma 5 (Konstruktion maximal konsistenter Mengen)

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

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

- Wenn Γ 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) $\phi \longrightarrow \psi \in \Gamma$ gdw. (wenn $\phi \in \Gamma$ dann $\psi \in \Gamma$)

Vollständigkeit

Lemma 7

Wenn Γ *konsistent, dann gibt es* v *so dass* $[\![\phi]\!]_v = 1$ *für* $\phi \in \Gamma$.

Damit:

- Wenn $\Gamma \nvdash \phi$ dann gibt es v so dass $[\![\psi]\!]_v = 1$ für $\psi \in \Gamma$, $[\![\phi]\!]_v = 0$.
- Wenn $\Gamma \nvdash \phi$ 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 $\vdash$

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
- Ausagenlogik ist entscheidbar: für Γ und ϕ, $\Gamma \vdash \phi$ oder $\Gamma \nvdash \phi$.
- Nächste VL: Prädikatenlogik

Formale Modellierung
Vorlesung 4 vom 04.05.15: Prädikatenlogik erster Stufe

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
  - Berechungsmodelle (Models of Computation)
  - Die Unvollständigkeitssätze von Gödel
- Teil II: Spezifikation und Verifikation

Das Tagesmenü

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

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 Ankunftsermin, 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.

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."

Prädikatenlogik: Erweiterung der Sprache

- Terme beschreiben die zu formalisierenden Objekte.
- Formeln sind logische Aussagen.
- Eine Signatur Σ beschreibt Prädikate und Funktionen:
  - Prädikatensymbole: $P_1, \ldots, P_n, \doteq$ mit Arität $ar(P_i) \in \mathbb{N}$, $ar(\doteq) = 2$
  - Funktionssymbole: $f_1, \ldots, f_m$ mit Arität $ar(t_i) \in \mathbb{N}$
- Menge X von Variablen (abzählbar viele)
- Konnektive: $\wedge, \longrightarrow, \bot, \forall$, abgeleitet: $\vee, \longleftrightarrow, \neg, \longleftrightarrow, \exists$
- Die Trennung zwischen Termen und Formeln ist der wesentliche Abstraktionsschritt in der Prädikatenlogik.

Terme

- Menge $\mathcal{T}erm_\Sigma$ der Terme (zur Signatur Σ) gegeben durch:
  - Variablen: $X \subseteq \mathcal{T}erm_\Sigma$
  - Funktionssymbol $f \in \Sigma$ mit $ar(f) = n$ und $t_1, \ldots, t_n \in \mathcal{T}erm_\Sigma$, dann $f(t_1, \ldots, t_n) \in \mathcal{T}erm_\Sigma$
  - Sonderfall: $n = 0$, dann ist f eine Konstante, $f \in \mathcal{T}erm_\Sigma$

Formeln

- Menge $\mathcal{F}orm_\Sigma$ der Formeln (zur Signatur Σ) gegeben durch:
  - $\bot \in \mathcal{F}orm_\Sigma$
  - Wenn $\phi \in \mathcal{F}orm_\Sigma$, dann $\neg\phi \in \mathcal{F}orm_\Sigma$
  - Wenn $\phi, \psi \in \mathcal{F}orm_\Sigma$, dann $\phi \wedge \psi \in \mathcal{F}orm_\Sigma$, $\phi \vee \psi \in \mathcal{F}orm_\Sigma$, $\phi \longrightarrow \psi \in \mathcal{F}orm_\Sigma$, $\phi \longleftrightarrow \psi \in \mathcal{F}orm_\Sigma$

Formeln

- Menge $\mathcal{F}orm_\Sigma$ der Formeln (zur Signatur Σ) gegeben durch:
  - $\bot \in \mathcal{F}orm_\Sigma$
  - Wenn $\phi \in \mathcal{F}orm_\Sigma$, dann $\neg\phi \in \mathcal{F}orm_\Sigma$
  - Wenn $\phi, \psi \in \mathcal{F}orm_\Sigma$, dann $\phi \wedge \psi \in \mathcal{F}orm_\Sigma$, $\phi \vee \psi \in \mathcal{F}orm_\Sigma$, $\phi \longrightarrow \psi \in \mathcal{F}orm_\Sigma$, $\phi \longleftrightarrow \psi \in \mathcal{F}orm_\Sigma$
  - Wenn $\phi \in \mathcal{F}orm_\Sigma, x \in X$, dann $\forall x.\phi \in \mathcal{F}orm_\Sigma, \exists x.\phi \in \mathcal{F}orm_\Sigma$
  - Prädikatensymbol $p \in \Sigma$ mit $ar(p) = m$ und $t_1, \ldots, t_m \in \mathcal{T}erm_\Sigma$, dann $p(t_1, \ldots, t_m) \in \mathcal{F}orm_\Sigma$
    - Sonderfall: $t_1, t_2 \in \mathcal{T}erm_\Sigma$, dann $t_1 \doteq t_2 \in \mathcal{F}orm_\Sigma$

Freie und gebundene Variable

Definition (Freie und gebundene Variablen)

Variablen in $t \in \mathcal{T}erm, p \in \mathcal{F}orm$ 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 ϕ gebunden

(iii) Ansonsten ist x frei

- $FV(\phi)$: Menge der freien Variablen in ϕ
- 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: $Cl(\phi) = \forall z_1 \ldots z_k.\phi$ für $FV(\phi) = \{z_1, \ldots, z_k\}$

Semantik: Strukturen

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

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

(i) A nicht-leere Menge (Universum)

(ii) für $f \in \Sigma$ mit $ar(f) = n$, n-stellige Funktion $f_{\mathfrak{A}} : A^n \to A$

(iii) für $P \in \Sigma$ mit $ar(P) = n$, n-stellige Relation $P_{\mathfrak{A}} \subseteq A^n$

- Für $a \in A$, Konstante $\overline{a} \in \mathcal{T}erm_{\Sigma}$
- Damit Auswertung von geschlossenen Termen: $[\![\cdot]\!]_{\mathfrak{A}} : \mathcal{T}erm_{\Sigma} \to A$

$$[\![\overline{a}]\!]_{\mathfrak{A}} = a$$
$$[\![f(t_1, \ldots, t_n]\!]_{\mathfrak{A}} = f_{\mathfrak{A}}([\![t_1]\!]_{\mathfrak{A}}, \ldots, [\![t_n]\!]_{\mathfrak{A}})$$

Semantische Gültigkeit

- Auswertung von Formeln: $[\![\cdot]\!]_{\mathfrak{A}} : \mathcal{F}orm_{\Sigma} \to \{0, 1\}$

$$
\begin{aligned}
[\![\bot]\!]_{\mathfrak{A}} &= 0 & [\![\neg\phi]\!]_{\mathfrak{A}} &= 1 - [\![\phi]\!]_{\mathfrak{A}} \\
[\![\phi \wedge \psi]\!]_{\mathfrak{A}} &= \min([\![\phi]\!]_{\mathfrak{A}}, [\![\psi]\!]_{\mathfrak{A}}) & [\![\phi \vee \psi]\!]_{\mathfrak{A}} &= \max([\![\phi]\!]_{\mathfrak{A}}, [\![\psi]\!]_{\mathfrak{A}}) \\
[\![\phi \longrightarrow \psi]\!]_{\mathfrak{A}} &= \max(1 - [\![\phi]\!]_{\mathfrak{A}}, [\![\psi]\!]_{\mathfrak{A}}) \\
[\![\phi \longleftrightarrow \psi]\!]_{\mathfrak{A}} &= 1 - |[\![\phi]\!]_{\mathfrak{A}} - [\![\psi]\!]_{\mathfrak{A}}|
\end{aligned}
$$

$$
\begin{aligned}
[\![P(t_1, \ldots, t_n)]\!]_{\mathfrak{A}} &= \begin{cases} 1 & \langle [\![t_1]\!]_{\mathfrak{A}}, \ldots, [\![t_n]\!]_{\mathfrak{A}} \rangle \in P_{\mathfrak{A}} \\ 0 & \text{sonst} \end{cases} \\
[\![t_1 \doteq t_2]\!]_{\mathfrak{A}} &= \begin{cases} 1 & [\![t_1]\!]_{\mathfrak{A}} = [\![t_2]\!]_{\mathfrak{A}} \\ 0 & \text{sonst} \end{cases} \\
[\![\forall x.\phi]\!]_{\mathfrak{A}} &= \min(\{[\![\phi\left[\tfrac{\bar{a}}{x}\right]]\!]_{\mathfrak{A}} \mid a \in A\}) \\
[\![\exists x.\phi]\!]_{\mathfrak{A}} &= \max(\{[\![\phi\left[\tfrac{\bar{a}}{x}\right]]\!]_{\mathfrak{A}} \mid a \in A\})
\end{aligned}
$$

- Damit semantische Gültigkeit (Wahrheit):

$$\mathfrak{A} \models \phi \text{ gdw. } [\![Cl(\phi)]\!]_{\mathfrak{A}} = 1, \models \phi \text{ gdw. } \mathfrak{A} \models \phi \text{ für alle } \mathfrak{A}$$

Syntaktische Gültigkeit: Natürliches Schließen

- Die alten Regeln blieben ($\longrightarrow I, \longrightarrow E, \wedge I, \wedge E_L, \wedge E_R, \mathsf{raa}, \bot, \ldots$)
  - Mutatis mutandis: $\mathcal{F}orm_\Sigma$ statt $\mathcal{P}rop$
- Dazu benötigen wir Regeln für die Quantoren.
- Zu behandelnde Probleme:
  - Substitution
  - Bindung

Substitution

- $t\left[\begin{smallmatrix} s \\ x \end{smallmatrix}\right]$ ist Ersetzung von x durch s in t
- Definiert durch strukturelle Induktion:

$$
\begin{aligned}
y\left[\begin{smallmatrix} s \\ x \end{smallmatrix}\right] &\stackrel{def}{=} \begin{cases} s & x = y \\ y & x \neq y \end{cases} \\
f(t_1, \ldots, t_n)\left[\begin{smallmatrix} s \\ x \end{smallmatrix}\right] &\stackrel{def}{=} f(t_1\left[\begin{smallmatrix} s \\ x \end{smallmatrix}\right], \ldots, t_n\left[\begin{smallmatrix} s \\ x \end{smallmatrix}\right]) \\
\bot\left[\begin{smallmatrix} s \\ x \end{smallmatrix}\right] &\stackrel{def}{=} \bot \\
(\phi \wedge \psi)\left[\begin{smallmatrix} s \\ x \end{smallmatrix}\right] &\stackrel{def}{=} \phi\left[\begin{smallmatrix} s \\ x \end{smallmatrix}\right] \wedge \psi\left[\begin{smallmatrix} s \\ x \end{smallmatrix}\right] \\
(\phi \longrightarrow \psi)\left[\begin{smallmatrix} s \\ x \end{smallmatrix}\right] &\stackrel{def}{=} \phi\left[\begin{smallmatrix} s \\ x \end{smallmatrix}\right] \longrightarrow \psi\left[\begin{smallmatrix} s \\ x \end{smallmatrix}\right] \\
P(t_1, \ldots, t_n)\left[\begin{smallmatrix} s \\ x \end{smallmatrix}\right] &\stackrel{def}{=} P(t_1\left[\begin{smallmatrix} s \\ x \end{smallmatrix}\right], \ldots, t_n\left[\begin{smallmatrix} s \\ x \end{smallmatrix}\right])
\end{aligned}
$$

Substitution

- $t\left[\begin{smallmatrix} s \\ x \end{smallmatrix}\right]$ ist Ersetzung von x durch s in t
- Definiert durch strukturelle Induktion:

$$
\begin{aligned}
y\left[\begin{smallmatrix} s \\ x \end{smallmatrix}\right] &\stackrel{def}{=} \begin{cases} s & x = y \\ y & x \neq y \end{cases} \\
f(t_1, \ldots, t_n)\left[\begin{smallmatrix} s \\ x \end{smallmatrix}\right] &\stackrel{def}{=} f(t_1\left[\begin{smallmatrix} s \\ x \end{smallmatrix}\right], \ldots, t_n\left[\begin{smallmatrix} s \\ x \end{smallmatrix}\right]) \\
\bot\left[\begin{smallmatrix} s \\ x \end{smallmatrix}\right] &\stackrel{def}{=} \bot \\
(\phi \wedge \psi)\left[\begin{smallmatrix} s \\ x \end{smallmatrix}\right] &\stackrel{def}{=} \phi\left[\begin{smallmatrix} s \\ x \end{smallmatrix}\right] \wedge \psi\left[\begin{smallmatrix} s \\ x \end{smallmatrix}\right] \\
(\phi \longrightarrow \psi)\left[\begin{smallmatrix} s \\ x \end{smallmatrix}\right] &\stackrel{def}{=} \phi\left[\begin{smallmatrix} s \\ x \end{smallmatrix}\right] \longrightarrow \psi\left[\begin{smallmatrix} s \\ x \end{smallmatrix}\right] \\
P(t_1, \ldots, t_n)\left[\begin{smallmatrix} s \\ x \end{smallmatrix}\right] &\stackrel{def}{=} P(t_1\left[\begin{smallmatrix} s \\ x \end{smallmatrix}\right], \ldots, t_n\left[\begin{smallmatrix} s \\ x \end{smallmatrix}\right]) \\
(\forall y.\phi)\left[\begin{smallmatrix} s \\ x \end{smallmatrix}\right] &\stackrel{def}{=} \begin{cases} \forall y.\phi & x = y \\ \forall y.(\phi\left[\begin{smallmatrix} s \\ x \end{smallmatrix}\right]) & x \neq y, y \notin FV(s) \\ \forall z.((\phi\left[\begin{smallmatrix} z \\ y \end{smallmatrix}\right])\left[\begin{smallmatrix} s \\ x \end{smallmatrix}\right]) & x \neq y, y \in FV(s) \\ & \text{mit } z \notin FV(s) \cup FV(\phi) \\ & (z \text{ frisch}) \end{cases}
\end{aligned}
$$

Natürliches Schließen mit Quantoren

$$\frac{\phi}{\forall x.\phi}\ \forall I\ \ (*) \qquad\qquad \frac{\forall x.\phi}{\phi\left[\begin{smallmatrix} t \\ x \end{smallmatrix}\right]}\ \forall E\ \ (\dagger)$$

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

Der Existenzquantor

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

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

- (*) Eigenvariablenbedingung:
  x nicht frei in ψ, oder einer offenenen Vorbedingung außer ϕ
- (†) Ggf. Umbenennung durch Substitution

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

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
  - Berechungsmodelle (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

$$\frac{\phi}{\forall x.\phi}\ \forall I\ \ (*) \qquad\qquad \frac{\forall x.\phi}{\phi\left[\begin{smallmatrix} t \\ x \end{smallmatrix}\right]}\ \forall E\ \ (\dagger)$$

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

Der Existenzquantor

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

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

- (*) Eigenvariablenbedingung:
  x nicht frei in ψ, oder einer offenenen Vorbedingung außer ϕ
- (†) 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:

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

- Substitutivität:

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

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 Γ konsistent, gibt es Valuation die Γ wahr macht.
- Frage:Korrektheit und Konsistenz für Prädkatenlogik?

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 $\mathfrak{A} \models \forall x.\phi(x)$

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 $\exists x.\phi(x) \longrightarrow \phi(c) \in T$

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: $\wedge, \longrightarrow, \bot$ und volle Aussagenlogik

Lemma 4 (Konservative Erweiterung bewahrt Konsistenz)

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

Maximal konsistente Theorien

Definition 5

Sei T Theorie zur Signatur Σ:

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

Lemma 6

T^* konservative Erweiterung von T

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_ω eine Henkin-Theorie und konservativ über T

Lemma 8 (Lindenbaum)

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

Vollständigkeit von ND

Lemma 9 (Existenz von Modellen)

Wenn Γ *konsistent, dann hat* Γ *ein Modell.*

- Beweis: Maximal konsistente Henkin-Erweiterung als Modell
- Herbrand-Modell, universelles Term-Modell
- Korrollar: Wenn $\Gamma \nvdash \phi$, dann $\Gamma \not\models \phi$

Theorem 10 (Vollständigkeit von ND)

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

Entscheidbarkeit

Theorem 11 (Kompaktheit)

Γ 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

Zusammenfassung

- Prädikatenlogik erster Stufe (FOL)
- Natürliches Schließen in FOL: Substitution und Eigenvariablenbedingung.
- FOL ist
  - konsistent,
  - vollständig,
  - aber nicht entscheidbar.

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
  - Berechungsmodelle (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:

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

- Kongruenz:

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

- Substitutivität:

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

Axiomatisierung der natürlichen Zahlen

- Axiome (erster Versuch):

$$\forall x.\ \mathsf{s}(x) \neq 0 \tag{N1}$$
$$\forall x.\, \forall y.\ \mathsf{s}(x) \doteq \mathsf{s}(y) \longrightarrow x \doteq y \tag{N2}$$
$$\forall x.\, x \neq 0 \longrightarrow \exists y.\, x \doteq \mathsf{s}(y) \tag{N3}$$
$$\forall x.\, 0 + x \doteq x \tag{A1}$$
$$\forall x.\, \forall y.\ \mathsf{s}(x) + y \doteq \mathsf{s}(x + y) \tag{A2}$$

- Beweise in ND

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

Modelle für Presburger-Arithmetik

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

$$\forall x.x \neq \underbrace{\mathsf{s}\ldots\mathsf{s}}_{n}(x) \qquad (\mathsf{K}_n)$$

- “Mehrere” Kopien von $\mathbb{N}$ weg, Zyklen weg — $\mathbb{Z}$ bleibt.
- $\mathbb{N}$ is das Standardmodell.
- Alle anderen Strukturen $\mathbb{N} + \mathbb{Z}$, $\mathbb{N} + \mathbb{Z} + \mathbb{Z}$, $\ldots$ sind Nichtstandardmodelle.

Induktionsschema

- Axiome für die Multiplikation:

$$x \cdot 0 \doteq 0 \tag{M1}$$
$$x \cdot \mathsf{s}(y) \doteq x \cdot y + x \tag{M2}$$

- Induktionsschema:

$$(P(0) \wedge \forall x.P(x) \longrightarrow P(\mathsf{s}(x))) \longrightarrow \forall x.P(x) \tag{ISNat}$$

- $P(\$)$ Formelschema
  - \$ ausgezeichnetes, neues Symbol ("Loch") und $P(t) \stackrel{def}{=} P(\$)\left[\begin{smallmatrix} t \\ \$ \end{smallmatrix}\right]$

Hilft das Induktionsschema zum Beweisen?

- Es gelten:

$$(\mathsf{N1}), (\mathsf{N2}), (\mathsf{ISNat}) \vdash (\mathsf{N3})$$
$$(\mathsf{N1}), (\mathsf{N2}), (\mathsf{ISNat}) \vdash (\mathsf{K}_n)$$

- Beweise in ND

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

. . . und auch

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

. . . und auch

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

Und was ist mit den Modellen?

- Ist $\mathbb{Z}$ jetzt weg?

Und was ist mit den Modellen?

- Ist $\mathbb{Z}$ jetzt weg?
- Sei $PA^{\infty} \stackrel{def}{=} (\mathsf{N1}), (\mathsf{N2}), (\mathsf{ISNat})+$ neues Symbol ∞ und Axiome

$$\infty \neq 0, \infty \neq \mathsf{s}(0), \infty \neq \mathsf{s}(\mathsf{s}(0)), \ldots$$

Und was ist mit den Modellen?

- Ist $\mathbb{Z}$ jetzt weg?
- Sei $PA^{\infty} \stackrel{def}{=} (\mathsf{N1}), (\mathsf{N2}), (\mathsf{ISNat})+$ neues Symbol ∞ und Axiome

$$\infty \neq 0, \infty \neq \mathsf{s}(0), \infty \neq \mathsf{s}(\mathsf{s}(0)), \ldots$$

Theorem (Kompaktheit)

Γ *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^{∞} Modell, das aber größer ist als $\mathbb{N}$
- Es kann in FOL keine Axiomatiserung für $\mathbb{N}$ geben, die keine Nichtstandardmodelle hat

Axiomatisierungen der natürlichen Zahlen

- Presburger-Arithmetik
  - 5 Axiome: (N1)(N2)(A1)(A2)(ISNat)
  - Konsistent und vollständig
  - Entscheidbar (Aufwand $2^{2^{cn}}$, 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

Zusammenfassung

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

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
  - Berechungsmodelle (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:

$$\forall x.0 \leq x \tag{L1}$$
$$\forall x.\forall y.x \leq y \longrightarrow \mathsf{s}(x) \leq \mathsf{s}(y) \tag{L2}$$

$$\mathsf{half}(0) = 0 \tag{H1}$$
$$\mathsf{half}(\mathsf{s}(0)) = 0 \tag{H2}$$
$$\forall x.\, \mathsf{half}(\mathsf{s}(\mathsf{s}(x))) = \mathsf{s}(\mathsf{half}(x)) \tag{H3}$$

- Beweise

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

Mehr Information

- Besser zum beweisen wäre wenn man gleich hätte

$$\dfrac{\mathsf{half}(0) \leq 0 \quad \mathsf{half}(\mathsf{s}(0)) \leq \mathsf{s}(0) \quad \mathsf{half}(\mathsf{s}(\mathsf{s}(c))) \leq \mathsf{s}(\mathsf{s}(c)) \qquad \left[\begin{array}{c} \mathsf{half}(c) \leq c \\ \vdots \end{array}\right]}{\forall x.\, \mathsf{half}(x) \leq x}$$

Mehr Information

- Besser zum beweisen wäre wenn man gleich hätte

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

- Vergleiche:

$$\mathsf{half}(0) = 0 \tag{H1}$$

$$\mathsf{half}(\mathsf{s}(0)) = 0 \tag{H2}$$

$$\forall x.\, \mathsf{half}(\mathsf{s}(\mathsf{s}(x))) = \mathsf{s}(\mathsf{half}(x)) \tag{H3}$$

Mehr Information

- Besser zum beweisen wäre wenn man gleich hätte

$$\dfrac{\begin{array}{c}\left[\ \mathsf{half}(c) \leq c\ \right] \\ \vdots \\ \mathsf{half}(0) \leq 0 \quad \mathsf{half}(\mathsf{s}(0)) \leq \mathsf{s}(0) \quad \mathsf{half}(\mathsf{s}(\mathsf{s}(c))) \leq \mathsf{s}(\mathsf{s}(c))\end{array}}{\forall x.\, \mathsf{half}(x) \leq x}$$

- Vergleiche:

$$\mathsf{half}(0) = 0 \tag{H1}$$
$$\mathsf{half}(\mathsf{s}(0)) = 0 \tag{H2}$$
$$\forall x.\, \mathsf{half}(\mathsf{s}(\mathsf{s}(x))) = \mathsf{s}(\mathsf{half}(x)) \tag{H3}$$

- Generiere Induktionschema aus rekursiven Funktionsdefinitionen

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

Wohlfundierte Induktion

- Wohlfundiertes Induktionsschema

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

- $<$ wohlfundierte Relation:

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

Beweis mit wohlfundierter Induktion

- $<$-Relation

$$\forall x.0 < \mathsf{s}(x) \qquad \forall x, y.x < y \longrightarrow \mathsf{s}(x) < \mathsf{s}(y)$$

- Beweise $<$ ist wohlfundiert
-

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

Beweis mit wohlfundierter Induktion

- $<$-Relation

$$\forall x.0 < \mathsf{s}(x) \qquad\qquad \forall x, y.x < y \longrightarrow \mathsf{s}(x) < \mathsf{s}(y)$$

- Beweise $<$ ist wohlfundiert
-

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

$$\forall x.\ \mathsf{half}(x) \leq x$$

Zulässige Induktionsschema

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

$$P_0 \longrightarrow f(x_1, \ldots, x_n) = t_0$$
$$\vdots$$
$$P_n \longrightarrow f(x_1, \ldots, x_n) = t_n$$

  $P_i \wedge P_j \longleftrightarrow \bot, \forall i \neq j$
  - terminierend
- Rekursive Definition nach wohlfundierter Relation garantiert Terminierung
  Für jeden atomaren, rekursiven Aufruf $f(t_1, \ldots, t_n)$ erzeuge Terminierungshypothese

$$P_i \longrightarrow (x_1, \ldots, x_n) > (t_1, \ldots, t_n)$$

Grenzen

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

Grenzen

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

- 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.

$$
\begin{aligned}
f(99) &= f(f(110)) \\
&= f(100) \\
&= f(f(111)) \\
&= f(101) \\
&= 91
\end{aligned}
$$

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

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

Formale Modellierung
Vorlesung 8 vom 08.06.15: Logik Höherer Stufe

Christoph Lüth

Universität Bremen

Sommersemester 2015

Fahrplan

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

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
  - Berechungsmodelle (Models of Computation)
  - Die Unvollständigkeitssätze von Gödel
- Teil II: Spezifikation und Verifikation

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

Warum Logik höherer Stufe?

- Aussagenlogik: keine Quantoren
- Logik 1. Stufe: Quantoren über Terme

$$\forall x\, y.\, x = y \longrightarrow y = x$$

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

$$\forall P.(P\,0 \wedge \forall x.P\,x \longrightarrow P\,(S\,x)) \longrightarrow \forall x.P\,x$$

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

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

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

Vermeidung von Inkonsistenzen

- Russell'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

Typen

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

Terme

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

Basis-Syntax

$$
\begin{array}{ll}
= & :: \alpha \Rightarrow \alpha \Rightarrow \mathit{Bool} \\
\longrightarrow & :: \mathit{Bool} \Rightarrow \mathit{Bool} \Rightarrow \mathit{Bool} \\
\iota & :: (\alpha \Rightarrow \mathit{Bool}) \Rightarrow \alpha \\
 & \\
\neg & :: \mathit{Bool} \Rightarrow \mathit{Bool} \\
\top & :: \mathit{Bool} \\
\bot & :: \mathit{Bool} \\
\mathit{if} & :: \mathit{Bool} \Rightarrow \alpha \Rightarrow \alpha \Rightarrow \alpha \\
\forall & :: (\alpha \Rightarrow \mathit{Bool}) \Rightarrow \mathit{Bool} \\
\exists & :: (\alpha \Rightarrow \mathit{Bool}) \Rightarrow \mathit{Bool} \\
\wedge & :: \mathit{Bool} \Rightarrow \mathit{Bool} \Rightarrow \mathit{Bool} \\
\vee & :: \mathit{Bool} \Rightarrow \mathit{Bool} \Rightarrow \mathit{Bool}
\end{array}
$$

- Einbettung (wird weggelassen)
  $\mathit{trueprop} :: \mathit{Bool} \Rightarrow \mathit{Prop}$
- Basis-Operatoren: $=, \longrightarrow, \iota$
- Syntaktische Konventionen:
  - Bindende Operatoren: $\forall$, $\exists$, ι
    $\forall x.P \equiv \forall(\lambda x.P)$
  - Infix-Operatoren: $\wedge, \vee, \longrightarrow, =$
  - Mixfix-Operator:
    $\mathit{if}\ b\ \mathit{then}\ p\ \mathit{else}\ q \equiv \mathit{if}\ b\ p\ q$

Basis-Axiome I: Gleichheit

- Reflexivität:

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

- Substitutivität:

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

- Extensionalität:

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

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

- Einführungsregel:

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

Abgeleitete Operatoren

$$
\begin{aligned}
\top &\equiv (\lambda x.\, x) = (\lambda x.\, x) \\
\forall P &\equiv (P = \lambda x.\, \top) \\
\exists P &\equiv \forall Q.(\forall x.P\ x \longrightarrow Q) \longrightarrow Q \\
\bot &\equiv \forall P.\, P \\
\neg P &\equiv P \longrightarrow \bot \\
P \wedge Q &\equiv \forall R.\, (P \longrightarrow Q \longrightarrow R) \longrightarrow R \\
P \vee Q &\equiv \forall R.\, (P \longrightarrow R) \longrightarrow (Q \longrightarrow R) \longrightarrow R \\
\textit{if } P \textit{ then } x \textit{ else } y &\equiv \iota z.(P = \top \longrightarrow z = x) \wedge (P = \bot \longrightarrow z = y)
\end{aligned}
$$

Basis-Axiome II: Implikation und Auswahl

- Einführungsregel Implikation:

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

- Eliminationsregel Implikation:

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

- Eliminationsregel Auswahloperator:

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

- HOL ist klassisch:

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

Die Basis-Axiome (Isabelle-Syntax)

$$\begin{aligned}
\text{refl}: &\quad t = t \\
\text{subst}: &\quad [\![s = t; P(s)]\!] \Longrightarrow P(t) \\
\text{ext}: &\quad [\![\bigwedge x.fx = gx]\!] \Longrightarrow (\lambda x.fx) = (\lambda x.gx) \\
\text{impl}: &\quad [\![P \Longrightarrow Q]\!] \Longrightarrow P \longrightarrow Q \\
\text{mp}: &\quad [\![P \longrightarrow Q; P]\!] \Longrightarrow Q \\
\text{iff}: &\quad (P \longrightarrow Q) \longrightarrow (Q \longrightarrow P) \longrightarrow (P = Q) \\
\text{the_eq}: &\quad (\iota x.x = a) = a \\
\text{true_or_false}: &\quad (P = \top) \vee (P = \bot)
\end{aligned}$$

Eigenschaften der Logik höherer Stufe

- Konsistent (soweit wir wissen)
- Unvollständig
  - . . . und damit unentscheidbar
  - Beweis: folgt aus den Gödelschen Unvollständigkeitssätzen

Erweiterungen

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

Zusammenfassung

Logik höherer Stufe (HOL):

- Syntax basiert auf dem einfach getypten λ-Kalkül
- Drei Basis-Operatoren, acht Basis-Axiome
- Rest folgt durch konservative Erweiterung — Donnerstag

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
  - Berechungsmodelle (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 voneinandar 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 $R_1, R_2, \ldots$, die natürliche Zahlen enthalten
- Programm besteht aus:
  - Endliche Anzahl von Zuständen $S_0, S_1, S_2, \ldots$, und
  - für $i > 0$, Anweisung (S_0: terminaler Zustand)

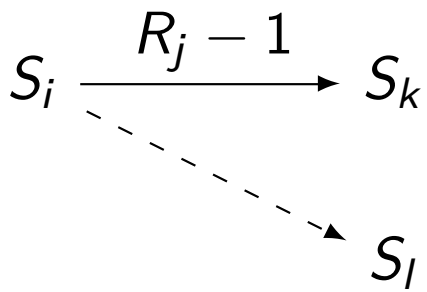
- Anweisungen:

$(j, +, k)$

$$S_i \xrightarrow{R_j + 1} S_k$$

Erhöhe R_j um 1 und gehe zu S_k

$(j, -, k, l)$



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

Beispielprogramme

Programm 1:

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

Programm 2:

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

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: $(2, +, 6)$

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 $f : \mathbb{N}^k \to \mathbb{N}$ (mit einem Programm P), wenn beim Start (von S_1) mit Registerinhalt $(n_1, n_2, \ldots, n_k, 0, \ldots)$ die Maschine schließlich mit Registerinhalt $f(n_1, \ldots, n_k)$ in R_1 anhält.

- Aber welche Funktionen sind berechenbar?

Theorem 1: Berechenbare Funktionen

(i) Die Projektionen $\pi_i(n_1, \ldots, n_k) = n_i$ ist berechenbar.

(ii) $const_0 : \mathbb{N} \to \mathbb{N}$ und $succ : \mathbb{N} \to \mathbb{N}$ sind berechenbar.

(iii) Komposition: Sei f mit Arität k berechenbar, und $g_1, \ldots, g_k$ mit Arität l berechenbar, dann ist h berechenbar:

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

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

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

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

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

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

Berechenbarkeit, Rekursion, Primitive Rekursion

- Primitiv rekursive Funktionen: kleinste unter (i) – (iv) abgeschlosene 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.

Kodierung von Programmen

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

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

- 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 $(m_1, \ldots, m_k)$ und $f_{n,k}(m_1, \ldots, m_k) = r$,*
- *und ansonsten $u(n, k, m)$ undefiniert.*

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.

Weitere Modelle von Berechenbarkeit

- Turing-Maschinen (remember those?)
- Chomsky-Grammatiken (Typ 0)
- λ-Kalkül
- Kombinatorlogik
- π-Kalkül
- Neuronale Netze
- Gegenbeispiel: Aktoren (Carl Hewitt)
  - Literaturlage nicht eindeutig ...

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

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

Gödel-Kodierung

- Folgende Funktion ist definierbar in PA:

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

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

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

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

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

- 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 $x_1, x_2, \cdots \in X$:

$$\lceil x_i \rceil \stackrel{def}{=} (0, i)$$

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

$$\lceil f_i \rceil \stackrel{def}{=} (1, i)$$

- Terme

$$\lceil f_i(t_1, \ldots t_n) \rceil \stackrel{def}{=} \langle \lceil f_i \rceil, \lceil t_1 \rceil, \ldots \lceil t_n \rceil \rangle$$

Gödel-Kodierung für Formeln

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

- Prädikate $p_1, \cdots \in \mathcal{P}$, $\bot \stackrel{def}{=} p_1$, $\doteq \stackrel{def}{=} p_2$

$$\lceil p_i \rceil \stackrel{def}{=} (2, i)$$

- Atome

$$\lceil p_i(t_1, \ldots t_n) \rceil \stackrel{def}{=} \langle \lceil p_i \rceil, \lceil t_1 \rceil, \ldots \lceil t_n \rceil \rangle$$

- Konnektive und Quantoren

$$\lceil \neg \rceil = (3,1), \lceil \wedge \rceil = (3,2), \lceil \vee \rceil = (3,3)$$
$$\lceil \longrightarrow \rceil = (3,4), \lceil \longleftrightarrow \rceil = (3,5), \lceil \forall \rceil = (3,6), \lceil \exists \rceil = (3,7)$$

Gödel-Kodierung für Formeln II

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

- $\lceil \neg\varphi \rceil = (\lceil \neg \rceil, \lceil \varphi \rceil)$
- $\lceil \psi \diamond \varphi \rceil = \langle \lceil \diamond \rceil, \lceil \psi \rceil, \lceil \varphi \rceil \rangle$ mit $\diamond \in \{\wedge, \vee, \longrightarrow, \longleftrightarrow\}$
- $\lceil Q\ x_i.\, \varphi \rceil = \langle \lceil Q \rceil, \lceil x_i \rceil, \lceil \varphi \rceil \rangle$ mit $Q \in \{\forall, \exists\}$

Lemma 1 (Eigenschaften der Gödel-Kodierung)

- *Sei* $G \stackrel{def}{=} \{\lceil \varphi \rceil | \varphi$ *Variable, Term, oder Formel* $\}$
- G *ist entscheidbar*
- $\lfloor n \rfloor = \varphi :\Leftrightarrow \lceil \varphi \rceil = n$ *ist eindeutig definiert auf* G
- *Substitutionsfunktion* $\mathrm{subst}(n, x, t) = m$ *definierbar auf* G

$$\lceil \varphi[t/x] \rceil = \mathrm{subst}(\lceil \varphi \rceil, \lceil x \rceil, \lceil t \rceil)$$

Gödel-Kodierung für Ableitungen

- Gödel Kodierung für Hypothesen Liste:

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

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

- Definition von Konkatenation $*$ und Streichen von Hypothesen

Gödel-Kodierung für Ableitungen

$$\left\lceil \dfrac{\overset{D_1}{\phi} \quad \overset{D_2}{\psi}}{\phi \wedge \psi}\ \wedge I \right\rceil = \langle (5, \lceil \wedge \rceil), \left\lceil \overset{D_1}{\phi} \right\rceil, \left\lceil \overset{D_2}{\psi} \right\rceil, \lceil \phi \wedge \psi \rceil \rangle$$

$$\left\lceil \dfrac{\overset{D}{\phi \wedge \psi}}{\phi}\ \wedge E_L \right\rceil = \langle (6, \lceil \wedge \rceil), \left\lceil \overset{D}{\phi \wedge \psi} \right\rceil, \lceil \phi \rceil \rangle$$

Gödel-Kodierung für Ableitungen

$$\left\lceil \dfrac{\begin{array}{c} D \\ \psi \end{array}}{\phi \longrightarrow \psi}\ \longrightarrow I \right\rceil \quad = \quad \langle (5, \lceil \longrightarrow \rceil), \left\lceil \begin{array}{c} D \\ \psi \end{array} \right\rceil, \lceil \phi \longrightarrow \psi \rceil \rangle$$

$$\left\lceil \dfrac{\begin{array}{cc} D_1 & \overset{D_2}{} \\ \phi & \phi \longrightarrow \psi \end{array}}{\psi}\ \longrightarrow E \right\rceil \quad = \quad \langle (6, \lceil \longrightarrow \rceil), \left\lceil \begin{array}{c} D_1 \\ \phi \end{array} \right\rceil, \left\lceil \phi \overset{D_2}{\longrightarrow} \psi \right\rceil, \lceil \psi \rceil \rangle$$

Gödel-Kodierung für Ableitungen

- Entsprechend für RAA, $\forall I$, $\forall E$
- Definiere $\mathrm{Der}(p, h, z)$: $\lfloor p \rfloor$ ist Beweis für $\lfloor z \rfloor$ aus Hypothesen $\lfloor h \rfloor$

$$
\begin{array}{lll}
\mathrm{Der}(p,h,z) \stackrel{def}{=} (4,z) \in h & & \text{Hypothese} \\
\quad \vee & \exists p_1, h_1, z_1, p_2, h_2, z_2. & \wedge I \\
& \mathrm{Der}(p_1,h_1,z_1) \wedge \mathrm{Der}(p_2,h_2,z_2) \wedge & \\
& h = h_1 * h_2 \wedge & \\
& p = \langle (5, \lceil \wedge \rceil), p_1, p_2, \lceil \lfloor z_1 \rfloor \wedge \lfloor z_2 \rfloor \rceil \rangle & \\
\quad \vee & \exists p_1, h_1, z_1, u. & \longrightarrow I \\
& \mathrm{Der}(p_1,h_1,z_1) \wedge & \\
& h = \mathrm{Streiche}(u, h_1) \wedge & \\
& p = \langle (5, \lceil \longrightarrow \rceil), p_1, \lceil \lfloor u \rfloor \longrightarrow \lfloor z_2 \rfloor \rceil \rangle & \\
\quad \ldots & &
\end{array}
$$

Beweisbarkeit

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

$$\mathcal{A}x(n) \longleftrightarrow \bigvee_{\varphi \in PA} n = \lceil \varphi \rceil$$

- $\mathrm{Prov}(p, f)$: p is Gödelnummer eines ND-Beweis für $\lfloor f \rfloor$

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

- $\mathrm{Thm}(f)$: $\lfloor f \rfloor$ ist ein Theorem

$$\mathrm{Thm}(f) \longleftrightarrow \exists p.\, \mathrm{Prov}(p, f)$$

Fixpunkt-Theorem

Theorem 2 (Fixpoint Theorem)

For each formula $\varphi(x)$ with only one free variable x there exists a formula ψ such that $\vdash \varphi(\lceil\psi\rceil) \longleftrightarrow \psi$

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 $\varphi(x) \stackrel{def}{=} \neg \mathsf{Thm}(x)$
- Es gibt ψ so dass $\vdash \psi \longleftrightarrow \neg \mathsf{Thm}(\lceil \psi \rceil)$
  - Gödel sentence: "Ich bin nicht beweisbar"
- Es gilt $\mathbf{PA} \models \psi \longleftrightarrow \neg \mathsf{Thm}(\lceil \psi \rceil)$
- Annahme: $\mathbf{PA} \vdash \psi$, dann $\mathbf{PA} \models \mathsf{Thm}(\lceil \psi \rceil)$

$\Leftrightarrow \mathbf{PA} \models \exists x.\, \mathsf{Prov}(x, \lceil \psi \rceil)$ $\quad$ $\Leftrightarrow \mathbf{PA} \models \mathsf{Prov}(n, \lceil \psi \rceil)$ for some n

$\Leftrightarrow \vdash \mathsf{Prov}(n, \lceil \psi \rceil)$ for some n $\quad$ $\Leftrightarrow \vdash \psi$

$\Rightarrow \vdash \neg \mathsf{Thm}(\lceil \psi \rceil)$ $\quad$ $\Rightarrow \mathbf{PA} \models \neg \mathsf{Thm}(\lceil \psi \rceil)$

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

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** konsistent ist, können wir die Konsistenz von **PA** nicht herleiten ($\mathbf{PA} \nvdash \mathit{Consis}_{\mathbf{PA}}$).

- 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”

Formale Modellierung
Vorlesung 11 vom 28.06.2014: Formale Modellierung von Software

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

Das Tagesmenü

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

Algebraische Spezikation

- 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

Die Grundidee

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

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

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

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"
```

Vor- und Nachteile

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

Weitere Modellierungssprachen

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

Java Modeling Language (JML)

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

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()
}
```

UML als formale Spezifikationssprache

Diagrammtyp	Modellierte Aspekte	Formal
Klassendiagramm	Statische Systemstruktur	

UML als formale Spezifikationssprache

Diagrammtyp	Modellierte Aspekte	Formal
Klassendiagramm	Statische Systemstruktur	Ja
Paketdiagramm	Pakete, Namensräume	

UML als formale Spezifikationssprache

Diagrammtyp	Modellierte Aspekte	Formal
Klassendiagramm	Statische Systemstruktur	Ja
Paketdiagramm	Pakete, Namensräume	Nein
Objektdiagramm	Zustand von Objekten	

UML als formale Spezifikationssprache

Diagrammtyp	Modellierte Aspekte	Formal
Klassendiagramm	Statische Systemstruktur	Ja
Paketdiagramm	Pakete, Namensräume	Nein
Objektdiagramm	Zustand von Objekten	(Ja)
Kompositionsstrukturdiagramm	Kollaborationen	

UML als formale Spezifikationssprache

Diagrammtyp	Modellierte Aspekte	Formal
Klassendiagramm	Statische Systemstruktur	Ja
Paketdiagramm	Pakete, Namensräume	Nein
Objektdiagramm	Zustand von Objekten	(Ja)
Kompositionsstrukturdiagramm	Kollaborationen	Nein
Komponentendiagramm	Dynamische Systemstruktur	

UML als formale Spezifikationssprache

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	

UML als formale Spezifikationssprache

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	

UML als formale Spezifikationssprache

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	

UML als formale Spezifikationssprache

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	

UML als formale Spezifikationssprache

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	

UML als formale Spezifikationssprache

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	

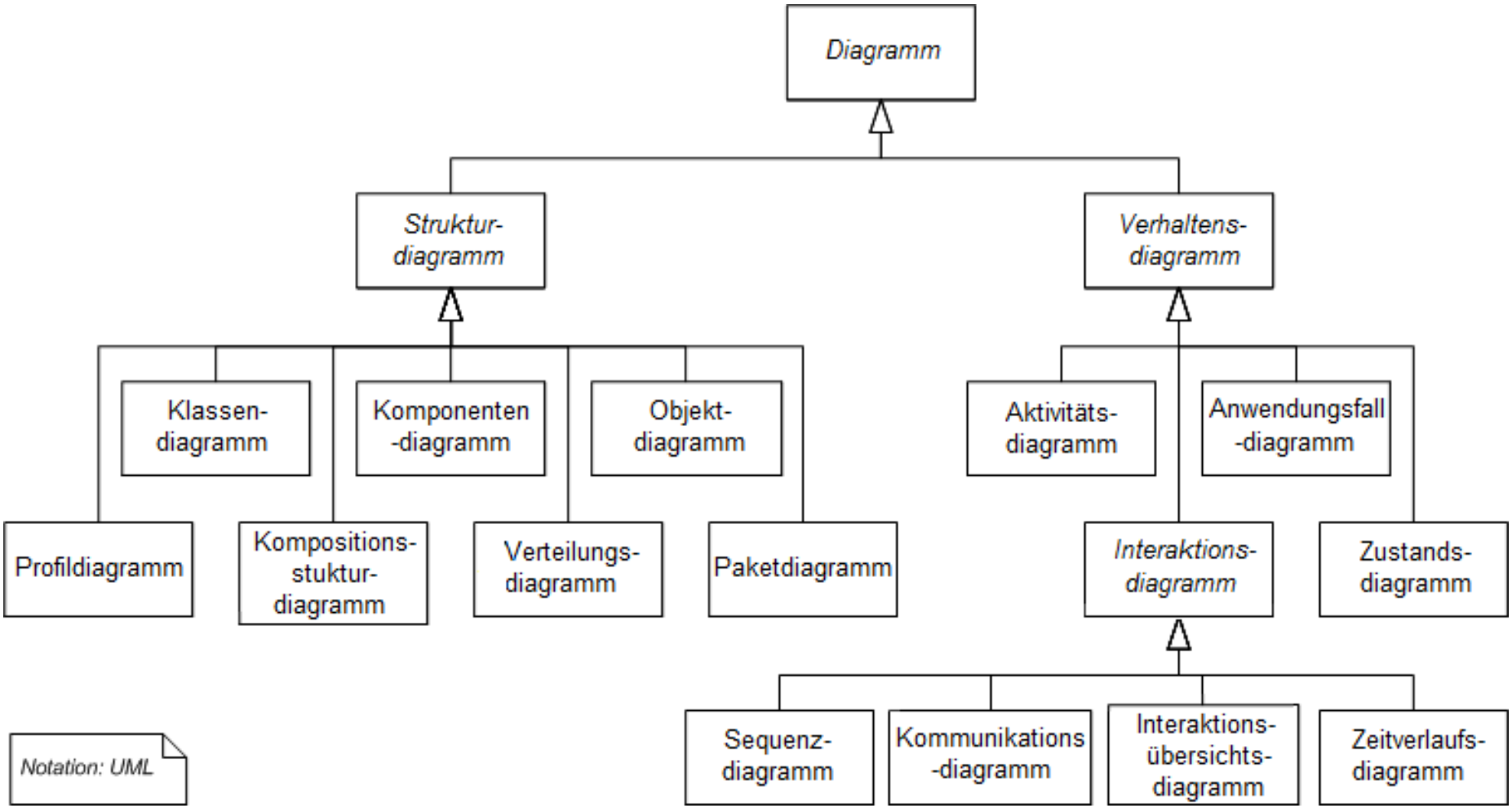
UML als formale Spezifikationssprache

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)
Zeitverlaufsdiagramm	Echtzeitaspekte	

UML als formale Spezifikationssprache

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)
Zeitverlaufsdiagramm	Echtzeitaspekte	(Ja)

Diagramme in UML 2.3



Quelle: Wikipedia

OCL

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

OCL Basics

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

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`

Undefiniertheit in OCL

- Undefiniertheit propagiert (alle Operationen strikt)
- Ausnahmen:
  - Boolsche Operatoren (`and`, `or` beidseitig nicht-strikt)
  - Fallunterscheidung
  - Test auf Definiertheit: `oclIsUndefined` mit

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

- Resultierende Logik: dreiwertig

Dreiwertige Logik

- Wahrheitstabelle (starke Kleene-Logik, K_3):

	$\neg$
$\bot$	$\bot$
0	1
1	0

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

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

$\longrightarrow$	$\bot$	0	1
$\bot$	$\bot$	$\bot$	1
0	1	1	1
1	$\bot$	0	1

$\longleftrightarrow$	$\bot$	0	1
$\bot$	$\bot$	$\bot$	$\bot$
0	$\bot$	1	0
1	$\bot$	0	1

- Fun Fact: K_3 hat keine Tautologien.
- Alternative: schwache Kleene-Logik (alle Operatoren strikt)

OCL Typen

- Basistypen:
  - `Boolean`, `Integer`, `Real`, `String`
  - `OclAny`, `OclType`, `OclVoid`
- Collection types: `Set`, `OrderedSet`, `Bag`, `Sequences`
- Modelltypen

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`

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`

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

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)

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

- Σ 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 Kaffe $10c$, 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 $\langle s_1, s_2, s_3, \ldots \rangle$ mit $s_i \in \Sigma$ und $s_i \rightarrow s_{i+1}$ für alle i.

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

Lineare Temporale Logik (LTL)

$$
\begin{array}{rll}
\phi ::= & \top \mid \bot \mid q & \text{— True, false, atomar} \\
\mid & \neg\phi \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \phi_1 \longrightarrow \phi_2 & \text{— Aussagenlog. Formeln} \\
\mid & \mathsf{X}\,\phi & \text{— Nächster Zustand} \\
\mid & \mathsf{F}\,\phi & \text{— Irgendwann} \\
\mid & \mathsf{G}\,\phi & \text{— Immer} \\
\mid & \phi_1\, U\, \phi_2 & \text{— Bis}
\end{array}
$$

- Präzendenzen: 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-Stuktur).
- 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 ϕ ist induktiv wie folgt definiert:

$$
\begin{array}{lcll}
p & \models & \top & \\
p & \not\models & \bot & \\
p & \models & q & \text{gdw} \quad q(p_1) \\
p & \models & \neg\phi & \text{gdw} \quad p \not\models \phi
\end{array}
\qquad
\begin{array}{lcll}
p & \models & \phi \wedge \psi & \text{gdw} \quad p \models \phi \text{ und } p \models \psi \\
p & \models & \phi \vee \psi & \text{gdw} \quad p \models \phi \text{ oder } p \models \psi \\
p & \models & \phi \longrightarrow \psi & \text{gdw} \quad \text{wenn } p \models \phi \\
 & & & \quad \text{dann } p \models \psi
\end{array}
$$

$$
\begin{array}{lcll}
p & \models & \mathsf{X}\,\phi & \text{gdw} \quad p^2 \models \phi \\
p & \models & \mathsf{G}\,\phi & \text{gdw} \quad \text{für alle } i \text{ gilt } p^i \models \phi \\
p & \models & \mathsf{F}\,\phi & \text{gdw} \quad \text{es gibt } i \text{ mit } p^i \models \phi \\
p & \models & \phi\, U\, \psi & \text{gdw} \quad \text{es gibt } i\ p^i \models \psi \text{ und für } j = 1, \ldots, i-1,\ \ p^j \models \phi
\end{array}
$$

Definition (Modell einer LTL-Formel)

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

Äquivalenzen

Definition (Äquivalenz)

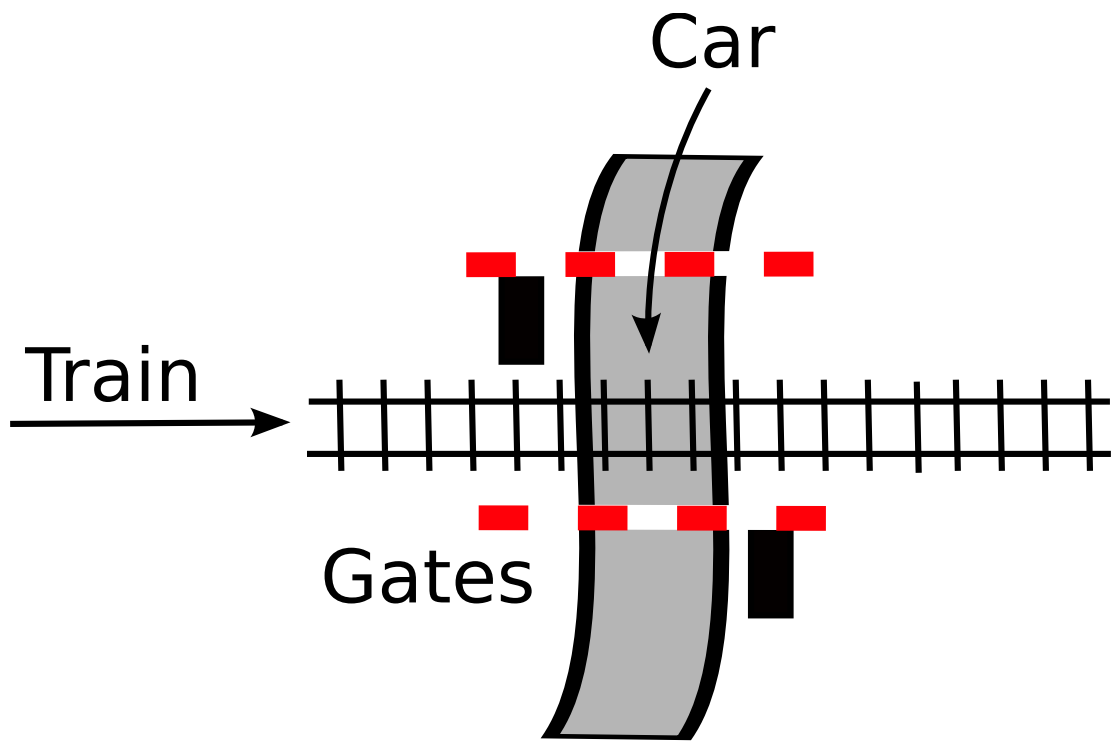
Zwei Formeln sind äquivalent, $\phi \equiv \psi$ gdw. für alle FSM $\mathcal{M}$ und Pfade p in $\mathcal{M}$, $p \models \phi \longleftrightarrow p \models \psi$

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

$$\mathsf{F}(\phi \vee \psi) \equiv \mathsf{F}\,\phi \vee \mathsf{F}\,\psi \qquad \neg\,\mathsf{F}\,\phi \equiv \mathsf{G}(\neg\phi) \qquad \mathit{FGF}\,\phi \equiv \mathit{GF}\,\phi$$
$$\mathsf{G}(\phi \wedge \psi) \equiv \mathsf{G}\,\phi \wedge \mathsf{G}\,\psi \qquad \neg\,\mathsf{G}\,\phi \equiv \mathsf{F}(\neg\phi) \qquad \mathit{GFG}\,\phi \equiv \mathit{FG}\,\phi$$
$$\neg\,\mathsf{X}\,\phi \equiv \mathsf{X}(\neg\phi)$$

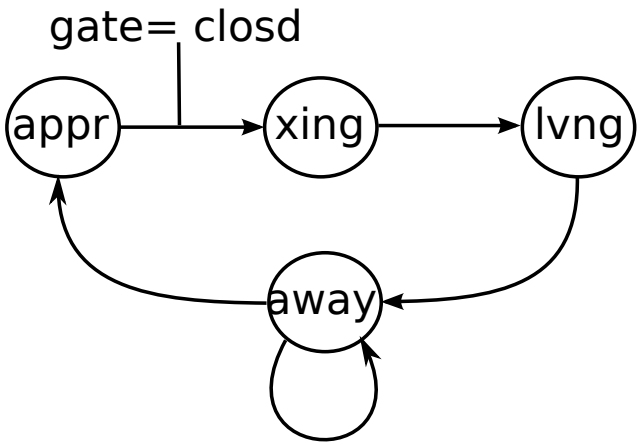
$$\mathsf{X}\,\mathsf{F}\,\phi \equiv \mathsf{F}\,\mathsf{X}\,\phi \qquad \mathsf{F}\,\phi \equiv \phi \vee \mathsf{X}\,\mathsf{F}\,\phi$$
$$\mathsf{X}\,\mathsf{G}\,\phi \equiv \mathsf{G}\,\mathsf{X}\,\phi \qquad \mathsf{G}\,\phi \equiv \phi \wedge \mathsf{X}\,\mathsf{G}\,\phi$$
$$\mathsf{X}(\phi\,U\,\psi) \equiv \mathsf{X}\,\phi\,U\,\mathsf{X}\,\psi \qquad \phi\,U\,\psi \equiv \psi \vee (\phi \wedge \mathsf{X}(\phi\,U\,\psi))$$

Längeres Beispiel: der Bahnübergang

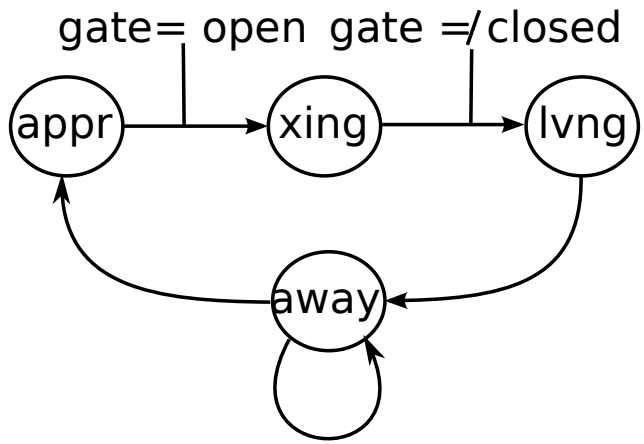


Modellierung des Bahnübergangs

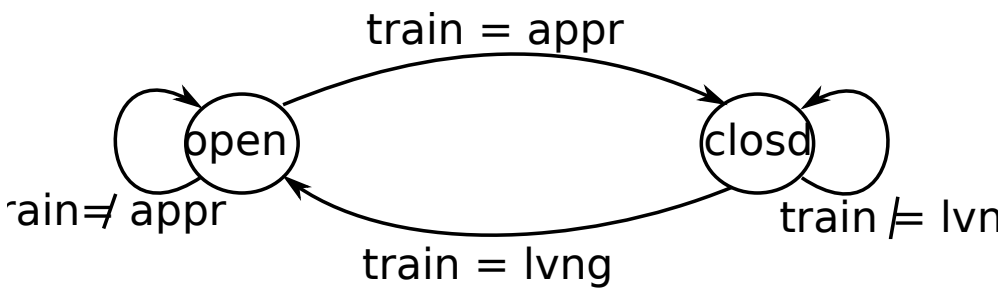
Zustände des Zuges:



Zustände des Autos:



Zustände der Schranke:



Die FSM

- Zustände sind eine endliche Abbildung der Variablen *Car*, *Train*, *Gate* auf Wertebereiche:

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

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

- Zustandsübergang komponentenweise, bspw:

$$\begin{aligned} &\langle away, open, away\rangle \rightarrow \langle appr, open, away\rangle \\ &\langle appr, open, away\rangle \rightarrow \langle xing, open, away\rangle \\ &\ldots \end{aligned}$$

Bahnübergang — Formalisierung von Eigenschaften

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

Bahnübergang — Formalisierung von Eigenschaften

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

$$\mathsf{G}\,\neg(\mathit{car} = \mathit{xing} \wedge \mathit{train} = \mathit{xing})$$

- Ein Auto kann den Übergang immer wieder verlassen:

Bahnübergang — Formalisierung von Eigenschaften

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

$$\mathsf{G}\,\neg(\mathit{car} = \mathit{xing} \wedge \mathit{train} = \mathit{xing})$$

- Ein Auto kann den Übergang immer wieder verlassen:

$$\mathsf{G}(\mathit{car} = \mathit{xing} \longrightarrow \mathsf{F}(\mathit{car} = \mathit{lvng}))$$

- Ein annähernder Zug darf irgendwann den Bahnübergang passieren:

Bahnübergang — Formalisierung von Eigenschaften

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

$$\mathsf{G}\,\neg(\mathit{car} = \mathit{xing} \wedge \mathit{train} = \mathit{xing})$$

- Ein Auto kann den Übergang immer wieder verlassen:

$$\mathsf{G}(\mathit{car} = \mathit{xing} \longrightarrow \mathsf{F}(\mathit{car} = \mathit{lvng}))$$

- Ein annähernder Zug darf irgendwann den Bahnübergang passieren:

$$\mathsf{G}(\mathit{train} = \mathit{appr} \longrightarrow \mathsf{F}(\mathit{train} = \mathit{xing}))$$

- Es gibt Autos, die den Bahnübergang passieren:

Bahnübergang — Formalisierung von Eigenschaften

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

$$\mathsf{G}\,\neg(\mathit{car} = \mathit{xing} \wedge \mathit{train} = \mathit{xing})$$

- Ein Auto kann den Übergang immer wieder verlassen:

$$\mathsf{G}(\mathit{car} = \mathit{xing} \longrightarrow \mathsf{F}(\mathit{car} = \mathit{lvng}))$$

- Ein annähernder Zug darf irgendwann den Bahnübergang passieren:

$$\mathsf{G}(\mathit{train} = \mathit{appr} \longrightarrow \mathsf{F}(\mathit{train} = \mathit{xing}))$$

- Es gibt Autos, die den Bahnübergang passieren:

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

  - Nicht in LTL auszudrücken!

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: $\mathsf{AF}\, p, \mathsf{EG}\, q, \mathsf{A}[p\, U\, q]$
- Name: Pfade im Berechnungsbaum durch Auffalten der FSM.
  - Beispiel Berechnungsbäume für die Getränkemaschine

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 Pfade, in denen p auftritt, tritt auch q auf.
- LTL: $\mathsf{F}\, p \longrightarrow \mathsf{F}\, q$
- CTL: Weder $\mathsf{AF}\, p \longrightarrow \mathsf{AF}\, q$ noch $\mathsf{AG}(p \longrightarrow \mathsf{AF}\, q)$
- Die Logik CTL^* kombiniert die Mächtigkeit von LTL and CTL.

Modellprüfung (Model-Checking)

- Das Model-Checking Problem:

  Gegeben Modell $\mathcal{M}$ und Eigenschaft ϕ, 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

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

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 ϕ, 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?

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

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:
  - $\mathrm{Thm}(f)$ gdw. $\lfloor f \rfloor$ ist ein Theorem
  - Fixpunktsatz: zu Formel $\varphi(x)$ gibt es ψ so dass $\vdash \varphi(\lceil \psi \rceil) \longleftrightarrow \psi$
  - *Gödel-sentence*: $\varphi(x) \stackrel{def}{=} \neg\, \mathrm{Thm}(x)$

UML

- Was ist formal an der UML?

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?

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?

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

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

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: Dijsktra, Gries und andere (1972 ff)
- Problem: sehr viele, größtenteils triviale Beweise

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.

Formale Modellierung und Formale Methoden

- Das LCF System (Robin Milner: Stanford LCF, Cambride 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

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)
- Hardwareverifkation: Intel, AMD, Infineon, ...
- L4.verified

Formale Modellierung in der Mathematik

- Offene mathematische Probleme und ihre Lösung:
  - Perelmann 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)

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)

. . . und jetzt?

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

Tschüß!