Formale Modellierung
Vorlesung 1 vom 03.04.13: Einführung

Serge Autexier & Christoph Lüth

Universität Bremen

Sommersemester 2013

Organisatorisches

- Veranstalter:

Serge Autexier
serge.autexier@dfki.de
MZH 3120, Tel. 59834

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

- Termine:

Montag, 16 – 18, MZH 1110
Donnerstag, 14 – 16, MZH 1110

- Webseite:

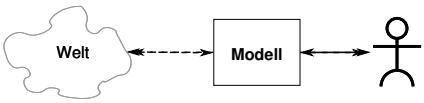
`http://www.informatik.uni-bremen.de/~cxl/lehre/foma.ss13`

Ariane-5

Die Vasa

10. August 1628

Modellierung — Das Problem



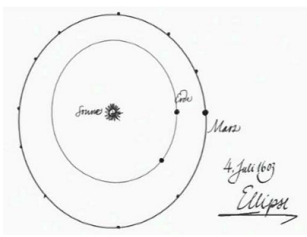
Modellierung — Das Problem

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

Modellierung — Das Problem

$$x = at^2 + bt + c$$

Modellierung — Das Problem



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

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 und der Sequenzenkalkül
  - Induktion, induktive Datentypen, Rekursion
  - Die Gödel-Theoreme
- Spezifikation und Verifikation:
  - Die Spezifikationssprache Z
  - Programme in Z
  - Beispiel, Anwendung

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: $\mathcal{T}h \not\vdash \bot$
  - 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.
- 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 08.04.13: Formale Logik und natürliches Schließen

Serge Autexier & Christoph Lüth

Universität Bremen

Sommersemester 2013

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: Syntax und Semantik, Natürliches Schließen
  - Konsistenz & Vollständigkeit der Aussagenlogik
  - Prädikatenlogik (FOL): Syntax und Semantik
  - Konsistenz & Vollständigkeit von FOL
  - FOL mit induktiven Datentypen
  - FOL mit Induktion und Rekursion
  - Die Gödel-Theoreme
  - Weitere Datentypen: Mengen, Multimengen, Punkte
- Teil II: Spezifikation und Verifikation
- Teil III: Schluß

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.
  - 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:

$$\frac{\diamondsuit}{\clubsuit}\,\alpha \qquad \frac{\diamondsuit}{\spadesuit}\,\beta \qquad \frac{\clubsuit \quad \spadesuit}{\heartsuit}\,\gamma \qquad \frac{}{\diamondsuit}\,\delta$$

- Beispielableitung: $\heartsuit$

Aussagenlogik

- Sprache $\mathcal{P}rop$ gegeben durch:
  1. Variablen (Atome) $V \Subset \mathcal{P}rop$ (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 $\Gamma \Subset \mathit{Prop}$, 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
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

Natürliches Schließen (ND)

- 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

Beispiel für 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 — Die Regeln

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

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

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

Die fehlenden Konnektive

- Einführung als Abkürzung:

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

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

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

- Ableitungsregeln als Theoreme.

Die fehlenden Schlußregeln

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

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

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

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 $[\![\cdot]\!]_\nu : \mathit{Prop} \to \{0, 1\}$
- Natürliches Schließen: intuitiver Kalkül
- Nächste Woche:
  - Sequenzenkalkül
  - Konsistenz und Vollständigkeit von Aussagenlogik

Formale Modellierung
Vorlesung 3 vom 15.04.13: Aussagenlogik: Konsistenz & Vollständigkeit

Serge Autexier & Christoph Lüth

Universität Bremen

Sommersemester 2013

Organisatorisches

Vorlesung und Übung nächste Woche (22.04, 25.04.) fallen aus!

Fahrplan

- Teil I: Formale Logik
  - Einführung
  - Aussagenlogik: Syntax und Semantik, Natürliches Schließen
  - Konsistenz & Vollständigkeit der Aussagenlogik
  - Prädikatenlogik (FOL): Syntax und Semantik
  - Konsistenz & Vollständigkeit von FOL
  - FOL mit induktiven Datentypen
  - FOL mit Induktion und Rekursion
  - Die Gödel-Theoreme
  - Weitere Datentypen: Mengen, Multimengen, Punkte
- Teil II: Spezifikation und Verifikation
- Teil III: Schluß

Das Tagesmenü

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

Eigenschaften der Aussagenlogik

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

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

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: $\models (\phi \longrightarrow (\psi \longrightarrow \sigma)) \longleftrightarrow (\phi \wedge \psi \longrightarrow \sigma)$
    $\models \phi \longrightarrow \psi \longrightarrow \phi$

Eigenschaften der Aussagenlogik

- Operatoren durch andere definierbar:

$$\begin{aligned}
&\models (\phi \longleftrightarrow \psi) \longleftrightarrow (\phi \longrightarrow \psi) \wedge (\psi \longrightarrow \phi)\\
&\models (\phi \longrightarrow \psi) \longleftrightarrow (\neg\phi \vee \psi)\\
&\models \phi \vee \psi \longleftrightarrow (\neg\phi \longrightarrow \psi)\\
&\models \phi \vee \psi \longleftrightarrow \neg(\neg\phi \wedge \neg\psi)\\
&\models \phi \wedge \psi \longleftrightarrow \neg(\neg\phi \vee \neg\psi)\\
&\models \neg\phi \longleftrightarrow (\phi \longrightarrow \bot)\\
&\models \bot \longleftrightarrow (\phi \wedge \neg\phi)
\end{aligned}$$

- $(\wedge, \neg)$ und $(\vee, \bot)$ sind genug (functional complete)
- Anwendung: konjunktive und disjunktive Normalformen (CNF/DNF)
- Gleichfalls: $A \mid B$ (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)

Lemma 1 (Korrektheit von ND)
Wenn $\Gamma \vdash \phi$, *dann* $\Gamma \models \phi$

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

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* $\Gamma \nvdash \phi$
(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 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 $\Gamma \Subset \Gamma'$ und Γ' konsistent, dann $\Gamma = \Gamma'$

Lemma 5 (Konstruktion maximal konsistenter Mengen)

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

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 der Aussagenlogik)

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

- Deshalb: Aussagenlogik entscheidbar

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 (29.04.13): Prädikatenlogik

Formale Modellierung
Vorlesung 4 vom 29.04.13: Prädikatenlogik erster Stufe

Serge Autexier & Christoph Lüth

Universität Bremen

Sommersemester 2013

Fahrplan

- Teil I: Formale Logik
  - Einführung
  - Aussagenlogik: Syntax und Semantik, Natürliches Schließen
  - Konsistenz & Vollständigkeit der Aussagenlogik
  - Prädikatenlogik (FOL): Syntax und Semantik
  - Konsistenz & Vollständigkeit von FOL
  - FOL mit induktiven Datentypen
  - FOL mit Induktion und Rekursion
  - Die Gödel-Theoreme
  - Weitere Datentypen: Mengen, Multimengen, Punkte
- Teil II: Spezifikation und Verifikation
- Teil III: Schluß

Das Tagesmenü

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

Beispiel: `Make`

The `make` utility automatically determines which pieces of a large program need to be recompiled, and issues commands to recompile them.

- Abhängigkeiten werden durch Regeln formalisiert
- Wenn Ziel älter ist als Abhängigkeit wird es neu erzeugt.

```
lecture-01.pdf: lecture-01.tex prelude.sty
        pdflatex lecture-01.tex

lecture-02.pdf: lecture-02.tex prelude.sty diagram.pdf
        pdflatex lecture-02.tex

diagram.pdf: diagram.svg
        inkscape -A diagram.pdf diagram.svg
```


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: $\vee, \longleftrightarrow, \neg, \longleftrightarrow, \exists$

Terme

- Menge $\mathcal{T}erm_\Sigma$ der Terme (zur Signatur Σ) gegeben durch:
  - Variablen: $X \Subset \mathcal{T}erm_\Sigma$
  - 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 $\phi \wedge \psi \in \mathcal{F}orm_\Sigma$, $\phi \vee \psi \in \mathcal{F}orm_\Sigma$, $\phi \longrightarrow \psi \in \mathcal{F}orm_\Sigma$, $\phi \longleftrightarrow \psi \in \mathcal{F}orm_\Sigma$
  - 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 $t_1, \ldots, t_m \in \mathcal{T}erm$, 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 $P_{\mathfrak{A}} \Subset A^n$

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

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

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

- Damit semantische Gültigkeit (Wahrheit):
  $\mathfrak{A} \models \phi$ gdw. $[\![Cl(\phi)]\!]_{\mathfrak{A}} = 1$, $\models \phi$ gdw. $\mathfrak{A} \models \phi$ für alle $\mathfrak{A}$

Substitution

- $t[^s_x]$ ist Ersetzung von x durch s in t
- Definiert durch strukturelle Induktion:

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

Natürliches Schließen mit Quantoren

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

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

Der Existenzquantor

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

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

- (*) 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: Vollständigkeit und natürliche Zahlen

Formale Modellierung
Vorlesung 5 vom 06.05.13: Eigenschaften der Prädikatenlogik erster Stufe

Serge Autexier & Christoph Lüth

Universität Bremen

Sommersemester 2013

Fahrplan

- Teil I: Formale Logik
  - Einführung
  - Aussagenlogik: Syntax und Semantik, Natürliches Schließen
  - Konsistenz & Vollständigkeit der Aussagenlogik
  - Prädikatenlogik (FOL): Syntax und Semantik
  - Konsistenz & Vollständigkeit von FOL
  - FOL mit induktiven Datentypen
  - FOL mit Induktion und Rekursion
  - Die Gödel-Theoreme
  - Weitere Datentypen: Mengen, Multimengen, Punkte
- Teil II: Spezifikation und Verifikation
- Teil III: Schluß

Das Tagesmenü

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

Natürliches Schließen mit Quantoren

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

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

Der Existenzquantor

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

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

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

Regeln für die Gleichheit

- Reflexivität, Symmetrie, Transitivität:

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

- Kongruenz:

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

- Substitutivität:

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

Die natürlichen Zahlen

- Verschiedene Axiomatisierungen:
- Presburger-Arithmetik
  - 5 Axiome
  - Konsistent und vollständig
  - Entscheidbar (Aufwand $2^{2^{cn}}$, n Länge der Aussage)
  - Enthält Nichtstandardmodelle
- Peano-Arithmetik
  - 8 Axiome
  - Konsistent
  - Unvollständig (bzgl. Standard-Modellen)
  - Nicht entscheidbar

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 $\forall x.\phi(x)$

Vorbereitende Definitionen

Definition 2 (Theorien, Henkin-Theorien)

(i) Eine Theorie ist eine unter Ableitbarkeit geschlossene Menge $T \Subset \mathcal{F}orm_\Sigma$

(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

Konstruktion einer maximal konsistenten Theorie

Definition 4

Sei T Theorie zur Signatur Σ:

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

Lemma 5

T^* *konservative Erweiterung von* T

Konstruktion maximal konsistenter Theorien

Lemma 6

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 7 (Lindenbaum)

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

Vollständigkeit von ND

Lemma 8 (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 9 (Vollständigkeit von ND)

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

Entscheidbarkeit

Theorem 10 (Kompaktheit)

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

- Aus Vollständigkeit folgt nicht Entscheidbarkeit:

Theorem 11 (Church)

Prädikatenlogik ist unentscheidbar.

- Beweis durch Kodierung von FOL in unentscheidbare Theorie

Zusammenfassung

- Natürliches Schließen in FOL: Substitution und Eigenvariablenbedingung.
- FOL ist vollständig, aber nicht entscheidbar

Formale Modellierung
Vorlesung 6 vom 13.05.13: Prädikatenlogik mit induktiven Datentypen

Serge Autexier & Christoph Lüth

Universität Bremen

Sommersemester 2013

Fahrplan

- Teil I: Formale Logik
  - Einführung
  - Aussagenlogik: Syntax und Semantik, Natürliches Schließen
  - Konsistenz & Vollständigkeit der Aussagenlogik
  - Prädikatenlogik (FOL): Syntax und Semantik
  - Konsistenz & Vollständigkeit von FOL
  - FOL mit induktiven Datentypen
  - FOL mit Induktion und Rekursion
  - Die Gödel-Theoreme
  - Weitere Datentypen: Mengen, Multimengen, Punkte
- Teil II: Spezifikation und Verifikation
- Teil III: Schluß

Das Tagesmenü

- Standard und Nichtstandardmodelle
- Kann man nichtstandard modell ausschliessen?
- Beweis von Eigenschaften von Funktionen mit FOL-ND
  - Induktive Datentypen mit einfacher, struktureller Induktion
  - Wohlfundierte Induktion und rekursive Funktionen

Beweisen mit Natürlichen Zahlen

- Axiome der Natürlichen Zahlen $\mathbb{N}$

$$\forall x.\, \mathsf{s}(x) \neq 0 \qquad \text{(N1)}$$
$$\forall x.\forall y.\, \mathsf{s}(x) = \mathsf{s}(y) \longrightarrow x = y \qquad \text{(N2)}$$
$$\forall x.\, x + 0 = x \qquad \text{(A1)}$$
$$\forall x.\forall y.\, x + \mathsf{s}(y) = \mathsf{s}(x + y) \qquad \text{(A2)}$$

- Beweise in ND

$$\text{(N1)(N2)(A1)(A2)} \vdash \forall x.0 + x = x$$

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 Schlußregeln

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

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

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

Natürliches Schließen mit Quantoren

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

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

Der Existenzquantor

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

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

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

Wie sehen unsere Zahlen eigtl. aus?

- Angefangen mit "0" und "s"
- Axiome $N1$ und $N2$

Modelle

- Füge hinzu:

$$\forall x.x \neq 0 \longrightarrow \exists y.x = \mathsf{s}(y) \qquad (\mathsf{N3})$$

- Füge weiter hinzu:

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

- "Mehrere" Kopien von $\mathbb{N}$ weg, Zyklen weg... $\mathbb{Z}$ bleibt.
- $\mathbb{N}$ is das Standardmodell. Alle anderen Strukturen $\mathbb{N}+\mathbb{Z}$, $\mathbb{N}+\mathbb{Z}+\mathbb{Z}$, ... die mehr als nur $\mathbb{N}$ enthalten sind Nichtstandardmodelle

Induktionsschema

- Induktionsschema für Natürliche Zahlen:

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

- $P(\$)$ Formelschema: \$ ausgezeichnetes, neues Symbol ("Variable") und

$$P(t) := P(\$)\begin{bmatrix} t \\ \$ \end{bmatrix}$$

- Abgeleitete ND Regeln:

$$\frac{P(0) \quad \forall x.P(x) \longrightarrow P(\mathsf{s}(x))}{\forall x.P(x)}\ \mathit{ISNat} \qquad \frac{\begin{array}{c}[P(c)] \\ \vdots \\ P(0) \quad P(\mathsf{s}(c))\end{array}}{\forall x.P(x)}\ \mathit{IS}^c,\ c \text{ Eigenvariable}$$

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

$$(\mathsf{N1})(\mathsf{N2})(\mathsf{A1})(\mathsf{A2})(\mathsf{ISNat}) \vdash \forall x.0 + x = x$$
... und auch
$$(\mathsf{N1})(\mathsf{N2})(\mathsf{A1})(\mathsf{A2})(\mathsf{ISNat}) \vdash \forall x.\forall y.\,\mathsf{s}(x) + y = \mathsf{s}(x + y)$$
... und auch
$$(\mathsf{N1})(\mathsf{N2})(\mathsf{A1})(\mathsf{A2})(\mathsf{ISNat}) \vdash \forall x.\forall y.x + y = y + x$$

- Definiere

$$(\mathsf{N1})(\mathsf{N2})(\mathsf{A1})(\mathsf{A2})(\mathsf{ISNat}) \qu=:\quad (\mathsf{Presburger})$$

Und was ist mit den Modellen?

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

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

- Jede endliche Teilmenge von PA^∞ hat Modell

Theorem 1 (Kompaktheit)
Γ hat ein Modell gdw. jede endliche Teilmenge $\Delta \Subset \Gamma$ hat ein Modell

- Also hat PA^∞ Modell, das aber größer ist als $\mathbb{N}$
- Es kann keine Axiomenmenge geben für $\mathbb{N}$ geben, die nicht auch noch Nichtstandartmodelle hat

Allgemein

- Alle natürlichen Zahlen sind konstruiert aus 0 und s:

$$\mathbb{N} := 0 \mid \mathsf{s}(\mathbb{N})$$

$$P(0) \wedge (\forall x_{\mathbb{N}}.P(x) \longrightarrow P(\mathsf{s}(x))) \longrightarrow \forall x_{\mathbb{N}}.P(x) \qquad (\mathsf{ISNat})$$

- Alle natürlichen Listen über Zahlen sind konstruiert aus Nil und cons:

$$\mathbb{LIST} := \mathsf{Nil} \mid \mathsf{cons}(\mathbb{N}, \mathbb{LIST})$$

$$P(\mathsf{Nil}) \wedge (\forall x_{\mathbb{LIST}}.P(x) \longrightarrow \forall n_{\mathbb{N}}.P(\mathsf{cons}(n, x))) \longrightarrow \forall x_{\mathbb{LIST}}.P(x) \qquad (\mathsf{ISList})$$

Allgemein

- Alle Binärbäume über Zahlen sind konstruiert aus Leaf und Node:

$$\mathbb{TREE} := \mathsf{Leaf}(\mathbb{N}) \mid \mathsf{Node}(\mathbb{TREE}, \mathbb{TREE})$$

$$\forall n_{\mathbb{N}}.P(\mathsf{Leaf}(n)) \wedge (\forall x_{\mathbb{TREE}}.\forall y_{\mathbb{TREE}}.(P(x) \wedge P(y)) \longrightarrow P(\mathsf{Node}(x, y))) \longrightarrow \forall x_{\mathbb{TREE}}.P(x) \qquad (\mathsf{ISTree})$$

- Und allgemein für frei erzeugte Datentypen.

Mehr Beweise

- Definiere $\leq$ und half:

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

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

- Beweise

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

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
-

$$\frac{\left[\ \forall x.x < c \wedge P(x)\ \right]\quad\vdots\quad P(c)}{\forall x.P(x) \leq x}$$

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

Mehr Information

- Besser zum beweisen wäre wenn man gleich hätte

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

- Vergleiche:

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

- Generiere Induktionschema aus rekursiven Funktionsdefinitionen

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

Weitere Beispiele

$$\mathbb{LIST} := \mathsf{Nil} \mid \mathsf{cons}(\mathbb{N}, \mathbb{LIST})$$

- Sortieren

$\forall x.\ \mathsf{sort}(\mathsf{Nil}) = \mathsf{Nil}$
$\forall s, t.m = \min(\mathsf{cons}(n, l))$
$\quad \longrightarrow \mathsf{sort}(\mathsf{cons}(n, l)) = \mathsf{cons}(m, \mathsf{sort}(\mathsf{cons}(n, l) - m))$
$\forall n.\ \min(\mathsf{cons}(n, \mathsf{Nil})) = n$
$\forall n, l.\ \min(\mathsf{cons}(m, l)) < n \longrightarrow \min(\mathsf{cons}(n, \mathsf{cons}(m, l))) = \min(\mathsf{cons}(m, l))$
$\forall n, l.\neg(\min(\mathsf{cons}(m, l)) < n) \longrightarrow \min(\mathsf{cons}(n, \mathsf{cons}(m, l))) = n$

- Induktionsschema

$$\frac{P(\mathsf{Nil}) \quad \forall m, n.m = \min(\mathsf{cons}(n, l)) \wedge P(\mathsf{cons}(n, l) - m) \longrightarrow P(\mathsf{cons}(n, l))}{\forall l.P(l)}$$

Weitere Beispiele

- Fibonacci:

$\mathsf{fib}(0) = 0$
$\mathsf{fib}(\mathsf{s}(0)) = s(0)$
$\forall n.\ \mathsf{fib}(s(s(n))) = \mathsf{fib}(s(n)) + \mathsf{fib}(n)$

$$\frac{P(0) \quad P(\mathsf{s}(0)) \quad \left[\ P(\mathsf{s}(c)), P(c)\ \right]\quad\vdots\quad P(c)}{\forall x.P(x)}$$

Weitere Beispiele

- GGT:

$\forall y.\ \mathsf{ggt}(0, y) = y$
$\forall x.\ \mathsf{ggt}(\mathsf{s}(x), 0) = \mathsf{s}(x)$
$\forall x, y.x \leq y \longrightarrow \mathsf{ggt}(x, y) = \mathsf{ggt}(x, y - x)$
$\forall x, y.\neg(x \leq y) \longrightarrow \mathsf{ggt}(x, y) = \mathsf{ggt}(x - y, y)$

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

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

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

- 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.

```
f(99) = f(f(110))
      = f(100)
      = f(f(111))
      = f(101)
      = 91
```

```
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)
        .... Pattern continues
      = f(99)
        (same as on the left)
      = 91
```


Zusammenfassung

- Jede Axiomenmenge zur Formalisierung der Natürlichen Zahlen hat Nichtstandardmodelle
- Induktionsschema für erzeugte Datentypen
- 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)

Formale Modellierung
Vorlesung 7 vom 23.05.13: FOL mit Induktion und Rekursion

Serge Autexier & Christoph Lüth

Universität Bremen

Sommersemester 2013

Fahrplan

- Teil I: Formale Logik
  - Einführung
  - Aussagenlogik: Syntax und Semantik, Natürliches Schließen
  - Konsistenz & Vollständigkeit der Aussagenlogik
  - Prädikatenlogik (FOL): Syntax und Semantik
  - Konsistenz & Vollständigkeit von FOL
  - FOL mit induktiven Datentypen
  - FOL mit Induktion und Rekursion
  - Die Gödel-Theoreme
  - Weitere Datentypen: Mengen, Multimengen, Punkte
- Teil II: Spezifikation und Verifikation
- Teil III: Schluß

Das Tagesmenü

- Axiomatische Definition von Theorien ist gefährlich
- Prädikatenlogik mit mehreren Typen
- Konservative Erweiterungen als sicheres Theorie Definitionsprinzip
  - Typdefinitionen
  - Wohlfundierte rekursive Funktionen/Prädikate

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 Schlußregeln

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

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

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

Natürliches Schließen mit Quantoren

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

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

Der Existenzquantor

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

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

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

Regeln für die Gleichheit

- Reflexivität, Symmetrie, Transitivität:

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

- Kongruenz:

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

- Substitutivität:

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

Motivation

- Typen müssen nicht-leere Trägermengen haben — Korrektheit
- Neue Typen axiomatisch zu spezifizieren gefährlich
- Konservative Erweiterungen
  - Typdefinitionen sind konservative Erweiterungen
  - Terminierende totale rekursive Funktionen/Prädikate sind konservative Erweiterungen

Getypte Prädikatenlogik – Signatur

	Ungetypt	Getypt
Signatur Σ		
- Typen $\mathcal{T}$	–	$i, \mathbb{N}, \mathbb{Z}$
- Funktionssymbole $\mathcal{F}$	$f, ar(f) = n$	$f : \tau_1 \times \cdots \times \tau_n \to \tau_0, \tau_i \in \mathcal{T}$
- Prädikatssymbole $\mathcal{P}$	$P, ar(P) = n$	$P : \tau_1 \times \cdots \times \tau_n, \tau_i \in \mathcal{T}$
	$\doteq, ar(\doteq) = 2$	$\doteq_\tau : \tau \times \tau, \tau \in \mathcal{T}$
Variablen X	abz. unendlich	abz. unendlich X_τ für jedes $\tau \in \mathcal{T}$
		$x_i, x_{\mathbb{N}}, x_{\mathbb{Z}}, \ldots$

Getypte Prädikatenlogik – Terme & Formeln

	Ungetypt	Getypt
Terme $\mathcal{T}erm_\Sigma$		$\mathcal{T}erm_\Sigma{}^{\tau_1} \cup \cdots \cup \mathcal{T}erm_\Sigma(\tau_n), \tau \in \mathcal{T}$
- Variablen	$x \in \mathcal{T}erm_\Sigma\ x \in X$	$x \in \mathcal{T}erm_\Sigma{}^{\tau}, x \in X_\tau$
- Funktionen	$f \in \mathcal{F}$ mit $ar(f) = n$ und $t_1, \ldots, t_n \in \mathcal{T}erm_\Sigma$, dann $f(t_1, \ldots, t_n) \in \mathcal{T}erm_\Sigma$	$f : \tau_1 \times \cdots \times \tau_n \to \tau_0 \in \mathcal{F}$ und $t_i \in \mathcal{T}erm^{\tau_i}_\Sigma, 1 \leq i \leq n$, dann $f(t_1, \ldots, t_n) \in \mathcal{T}erm^{\tau_0}_\Sigma$
Formeln $\mathcal{F}orm_\Sigma$		
- Atome	$P \in \mathcal{P}$ mit $ar(P) = n$ und $t_1, \ldots, t_n \in \mathcal{T}erm_\Sigma$, dann $P(t_1, \ldots, t_n) \in \mathcal{F}orm_\Sigma$	$P : \tau_1 \times \cdots \times \tau_n \in \mathcal{P}$ und $t_i \in \mathcal{T}erm^{\tau_i}_\Sigma, 1 \leq i \leq n$, dann $P(t_1, \ldots, t_n) \in \mathcal{F}orm_\Sigma$
- PL Konnective	$\neg\psi, \varphi \wedge \psi, \varphi \vee \psi, \varphi \longrightarrow \psi, \varphi \longleftrightarrow \psi \ldots$	
- Quantoren	$\forall x.\phi \in \mathcal{F}orm_\Sigma, x \in X$	$\forall x_\tau.\phi \in \mathcal{F}orm_\Sigma$
	$\exists x.\phi \in \mathcal{F}orm_\Sigma, x \in X$	$\exists x_\tau.\phi \in \mathcal{F}orm_\Sigma$

Getypte Prädikatenlogik – ND Regeln

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

- (*) Eigenvariablenbedingung:
  x_τ nicht frei in offenen Vorbedingungen von ϕ (x_τ beliebig)
- (†) $t \in \mathcal{T}erm^\tau_\Sigma$; Ggf. Umbenennung durch Substitution

Der Existenzquantor

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

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

- (*) Eigenvariablenbedingung:
  x_τ nicht frei in ψ, oder einer offenenen Vorbedingung außer ϕ
- (†) $t \in \mathcal{T}erm^\tau_\Sigma$; Ggf. Umbenennung durch Substitution

Regeln für die Gleichheit

- Reflexivität, Symmetrie, Transitivität:

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

- Kongruenz:

$$\frac{x_1 =_{t_1} y_1, \ldots, x_n =_{t_n} y_n}{f(x_1, \ldots, x_n) =_t f(y_1, \ldots, y_n)}\ \text{cong}$$

- Substitutivität:

$$\frac{x_1 =_{t_1} y_1, \ldots, x_m =_{t_m} y_m \quad P(x_1, \ldots, x_m)}{P(y_1, \ldots, y_m)}\ \text{subst}$$

Basic Definitions

Definition 1 (Loose Spezifikationen)

Sei $\Sigma = (\mathcal{T}, \mathcal{F}, \mathcal{P})$ eine getypte Signature und $\Phi \in \mathcal{F}orm_\Sigma$. Dann ist $S = (\Sigma, \Phi)$ eine lose Spezifikation.
Die Theorie einer Spezifikation S ist $\mathrm{Th}(S) := \{\varphi \in \mathcal{F}orm_\Sigma | \Phi \vdash \varphi\}$.

Definition 2 (Konsistenz)

Eine lose Spezifikation S ist konsistent wenn $\bot$ nicht beweisbar in S: $\bot \notin \mathrm{Th}(S)$.

- Insbesondere müssen dann alle Typen nicht-leere Trägermengen haben

Spezifikations Erweiterungen

Definition 3 (Erweiterungen)

Eine Spezifikation $S' = (\Sigma', \Phi')$ ist eine Erweiterung einer Spezifikation $S = (\Sigma, \Phi)$ genau dann wenn

- $\Sigma \subseteq \Sigma'$
- $\Phi \subseteq \Phi'$

S' is eine konservative Erweiterung von S genau dann wenn

$$\mathrm{Th}(S) = \mathrm{Th}(S')_{|\Sigma}$$

wobei die $|\Sigma$ die Einschränkung auf Formeln aus $\mathcal{T}erm_\Sigma$ ist

Lemma 4

Jede konservative Erweiterung einer konsistenten Theorie ist konsistent.

Typdefinition

- Spezifiziere nicht-leere Teilmenge eines gegebenen Typs r
- Deklariere neuen Typ t mit Trägermenge isomorph zu Werten in spezifizierter Teilmenge
- Isomorphie wird durch inverse Funktionen $\mathsf{Abs}_t : r \to t, \mathsf{Rep}_t : t \to r$ axiomatisch Beschrieben

Typdefinitionen sind Erweiterungen

Definition 5 (Typdefinitionen)

Sei $S = ((\mathcal{T}, \mathcal{F}, \mathcal{P}), \Phi)$ eine Spezifikation, $r \in \mathcal{T}$ und $P \in \mathcal{F}orm_\Sigma$ mit genau einer freien Variable vom Typ r. Dann ist eine Erweiterung $S' = ((\mathcal{T}', \mathcal{F}', \mathcal{P}'), \Phi')$ eine Typdefinition für einen Typ $t \notin \mathcal{T}$ gdw.

- $\mathcal{T}' = \mathcal{T} \cup \{t\}$
- $\mathcal{F}' = \mathcal{F} \cup \{\mathsf{Abs}_t : r \to t, \mathsf{Rep}_t : t \to r\}$
- $\mathcal{P}' = \mathcal{P} \cup \{=_t : t \times t\}$
- $\Phi' = \Phi \cup \{\ \forall x_t.\, \mathsf{Abs}_t(\mathsf{Rep}_t(x)) =_t x,$
  $\forall x_r.P(x_r) \longrightarrow \mathsf{Rep}_t(\mathsf{Abs}_t(x)) =_r x\}$
- Man kann beweisen $S \vdash \exists x_r.P(x)$ (bzw. es gilt $\exists x_r.P(x) \in \mathsf{Th}(S)$)

Teminierende, totale Funktionen

- Spezifiziere Funktionen/Prädikate die beweisbar total, eindeutig und terminierend sind
- Theorie-Erweiterungen um beweisbar total, eindeutig und terminierende Funktionen/Prädikate sind konservative Erweiterungen
- Syntaktische Kriterien für eindeutige und totale Deklarationen
- Beweisverfahren für terminierende Funktionen

Frei Erzeugte Typen

Definition 6 (Frei Erzeugte Typen)

Sei $S = ((\mathcal{T}, \mathcal{F}, \mathcal{P}), \Phi)$ eine Spezifikation, $t \in \mathcal{T}$ and $c_i : \tau_1^i \times \tau_{n_i}^i \to t \in \mathcal{F}, 1 \leq i \leq k$. Dann ist t frei erzeugt in S durch Konstruktoren $c_1, \ldots, c_k$ gdw.

- $S \vdash \forall x_t.\, \bigvee_{i=1\ldots k} \exists y^1_{\tau_1^i}, \ldots y^{n_k}_{\tau_{n_i}^i}.x = c_i(y^1, \ldots, y^{n_i})$
- $S \vdash \forall y^1_{\tau_1^i}, \ldots y^{n_i}_{\tau_{n_i}^i}.\forall z^1_{\tau_1^i}, \ldots z^{n_i}_{\tau_{n_i}^i}.c_i(y^1, \ldots, y^{n_i}) = c_i(z^1, \ldots, z^{n_i}) \longrightarrow ((y^1 = z^1 \wedge \cdots \wedge y^{n_i} = y^{n_i}))$ für alle c_i
- $S \vdash \forall y^1_{\tau_1^i}, \ldots y^{n_i}_{\tau_{n_i}^i}.\forall z^1_{\tau_1^j}, \ldots z^{n_j}_{\tau_{n_j}^j}.c_i(y^1, \ldots, y^{n_i}) = c_j(z^1, \ldots, z^{n_j})$ für alle $i \neq j$

Kriterien für eindeutig und total

- Sei t Typ
- Definitionsgleichungen für Funktion f sind Menge von bedingten geschlossene Gleichungen der Form

$$\forall x_{1_{\tau_1}} \ldots x_{n_{\tau_n}} \ldots . P_0 \longrightarrow f(x_1, \ldots, x_n) = t_0$$
$$\vdots$$
$$\forall x_{1_{\tau_1}} \ldots x_{n_{\tau_n}} \ldots . P_n \longrightarrow f(x_1, \ldots, x_n) = t_n$$

  so daß beweisbar

  - $S \vdash \forall x_{1_{\tau_1}} \ldots x_{n_{\tau_n}}.P_i \wedge P_j \longleftrightarrow \bot, \forall i \neq j$
  - $S \vdash \forall x_{1_{\tau_1}} \ldots x_{n_{\tau_n}}.P_1 \vee \ldots \vee P_n$

Terminierungsbeweise – Idee

- Die natürlichen Zahlen sind frei erzeugt über 0 und s:
- Jedem Grundterm über $\mathbb{N}$ kann eine größe zugeordnet werden über die Anzahl der Konstruktoren.
- Zeige für rekursiv definierte Funktionen auf $\mathbb{N}$, dass die rekursiven Argument in rekursiven Funktionsaufrufen kleiner sind bezüglich der Ordnung auf den natürlichen Zahlen unter der entsprechenden Bedingung P_i.

Terminierung

- Beispiele:
  - $\mathsf{half}(x)$ eine Hypothese pro Rekursionsgleichung
  - $\mathsf{fib}(x)$: mehrere Hypothesen pro Rekursionsgleichung
  - $\gcd(x, y)$: lexicographische Ordnung
- Beweise alle Hypothesen im Kalkül. Terminierung gilt relativ zur Terminierung der anderen involvierten Funktionen und Prädikate.
- Analog für Prädikate auf $\mathbb{N}$ mit bedingten Äquivalenzen
- Allgemeine Typen: für frei erzeugte Datentypen kann Abbildung in natürliche Zahlen definiert werden, die die Anzahl der Konstruktoren zählt. Damit lässt sich das Terminierungsverfahren auf all frei erzeugten Datentypen erweitern

Erweiterung um Totale, Terminierende Funktionen is Konservativ

Definition 7 (Funktions- und Prädikatsdefinitionen)

Sei $S = ((\mathcal{T}, \mathcal{F}, \mathcal{P}), \Phi)$ eine Spezifikation, $f : \tau_1 \times \cdots \times \tau_n \to \tau_0 \notin \mathcal{F}$ ($\tau_i \in \mathcal{T}$) und $\Psi \in \mathcal{F}orm_{\Sigma \cup \{f\}}$. Dann ist eine Erweiterung $S' = ((\mathcal{T}, \mathcal{F}', \mathcal{P}), \Phi')$ eine Funktionsdefinition gdw.

- Ψ ist eine eindeutig und totale Definition für f
- f ist terminierend und alle in der Definition von f vorkommenden Funktionen und Prädikate sind terminierend
- $\Phi' = \Phi \cup \Psi$
- $\mathcal{F}' = \mathcal{F} \cup \{f : \tau_1 \times \cdots \times \tau_n \to \tau_0\}$

Analog für Prädikatsdefinitionen.

Lemma 8

Funktionsdefinitionen bzw. Prädikatsdefinitionen sind konservativ

Sicheres Spezifikationsprinzip

- Beginne mit Basistheorie mit $\mathbb{N}$ und wohlfundiertem Induktionsschemata für $\mathbb{N}$ (getypte Prädikatenlogik mit Typ $\mathbb{N}$ und Induktionsschemata!)
  - $\mathbb{N}$ hat beweisbar nicht-leere Trägermenge
- Erweitere nur konservativ um
  - totale, terminierende Funktionen und Prädikate
  - Typdefinitionen (ausgehend von $\mathbb{N}$)
    - Erbt Induktionsprinzip über Umweg über $\mathbb{N}$
- Erlaubt Definition von Konstruktoren für neue Typen
  - Terminierung: Abbildung der Termgröße auf $\mathbb{N}$ mittels geschachtelter Anwendung von $\mathrm{Rep}_{\mathsf{t}}$
  - Wenn freie Erzeugtheit des neuen Typs beweisbar, dann folgt Induktionsschema direkt auf dem neuen Typ
- Damit hat man garantiert immer konsistente Spezifikationen (= Modellierung).

Formale Modellierung
Vorlesung 8 vom 27.05.13: Die Gödel-Theoreme

Serge Autexier & Christoph Lüth

Universität Bremen

Sommersemester 2013

Fahrplan

- Teil I: Formale Logik
  - Einführung
  - Aussagenlogik: Syntax und Semantik, Natürliches Schließen
  - Konsistenz & Vollständigkeit der Aussagenlogik
  - Prädikatenlogik (FOL): Syntax und Semantik
  - Konsistenz & Vollständigkeit von FOL
  - FOL mit induktiven Datentypen
  - FOL mit Induktion und Rekursion
  - Die Gödel-Theoreme
  - Weitere Datentypen: Mengen, Multimengen, Punkte
- Teil II: Spezifikation und Verifikation
- Teil III: Schluß

Das Tagesmenü

- Terminierende Funktionen und abgeleitete Induktionsschemata
- 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.

Sicheres Spezifikationsprinzip

- Beginne mit Basistheorie mit $\mathbb{N}$ und wohlfundiertem Induktionsschemata für $\mathbb{N}$ (getypte Prädikatenlogik mit Typ $\mathbb{N}$ und Induktionsschemata!)
  - $\mathbb{N}$ hat beweisbar nicht-leere Trägermenge
- Erweitere nur konservativ um
  - totale, terminierende Funktionen und Prädikate
  - Typdefinitionen (ausgehend von $\mathbb{N}$)
    - Erbt Induktionsprinzip über Umweg über $\mathbb{N}$
- Erlaubt Definition von Konstruktoren für neue Typen
  - Terminierung: Abbildung der Termgröße auf $\mathbb{N}$ mittels geschachtelter Anwendung von Rep_t
  - Wenn freie Erzeugtheit des neuen Typs beweisbar, dann folgt Induktionsschema direkt auf dem neuen Typ
- Damit hat man garantiert immer konsistente Spezifikationen (= Modellierung).

Abgeleitetete Induktionsschemata

- $\mathrm{fib}(x)$

$$\frac{P(0) \quad P(s(0)) \quad \forall x.P(x) \wedge P(\mathrm{s}(x)) \longrightarrow P(\mathrm{s}(\mathrm{s}(x)))}{\forall x.P(x)}$$

- $\mathrm{half}(x)$

$$\frac{P(0) \quad P(s(0)) \quad \forall x.P(x) \longrightarrow P(\mathrm{s}(\mathrm{s}(x)))}{\forall x.P(x)}$$

- $\mathrm{gcd}(x)$

$$\frac{\begin{array}{l} P(0,y) \\ x > 0 \longrightarrow P(x,0) \\ \forall x,y.x > y \wedge P(x-y,y) \longrightarrow P(x,y) \\ \forall x,y.\neg(x > y) \wedge P(x,y-x) \longrightarrow P(x,y) \end{array}}{\forall x.\forall y.P(x,y)}$$

Abgeleitetete Induktionsschemata besser zum Beweisen

- Abgeleitete Induktionsschemata erzeugen Fälle, in denen die Rekursionsgleichungen der Funktion/Prädikate direkt anwendbar sind
- Abgeleitete Induktionsschemata hilfreich wenn Induktion über Variablen gemacht wird, die als Argument der entsprechenden Funktion vorkommen.

$$\forall x.\varphi(\mathrm{half}(x))$$

- Fälle:
  1. $\varphi(\mathrm{half}(0)) \rightsquigarrow \varphi(0)$
  2. $\varphi(\mathrm{half}(\mathrm{s}(0))) \rightsquigarrow \varphi(0)$
  3. $\varphi(\mathrm{half}(x) \longrightarrow \varphi(\mathrm{half}(\mathrm{s}(\mathrm{s}(x)))) \rightsquigarrow \varphi(\mathrm{half}(x) \longrightarrow \varphi(\mathrm{s}(\mathrm{half}(x)))$

Gödels erster Unvollständigkeitssatz

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

Gödel Kodierung

Folgende Funktion ist definierbar in PA:

$$(n, m) = 2^n \times 3^m$$

Eigenschaften: Es gibt eindeutige Projektionen

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

Gödel Kodierung für Terme

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

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

$$\lceil x_i \rceil := (0, i)$$

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

$$\lceil f_i \rceil := (1, i)$$

- Terme

$$\lceil f_i(t_1, \ldots t_n) \rceil := \langle \lceil f_i \rceil, \lceil t_1 \rceil, \ldots \lceil t_n \rceil \rangle$$

wobei

$$\langle n_1, \ldots, n_k \rangle := \begin{cases} n_1 & \text{if } k = 1 \\ (n_1, \langle n_2, \ldots, n_k \rangle) & \text{if k} > 1 \end{cases}$$

Gödel Kodierung für Formeln

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

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

$$\lceil p_i \rceil := (2, i)$$

- Atome

$$\lceil p_i(t_1, \ldots t_n) \rceil := \langle \lceil p_i \rceil, \lceil t_1 \rceil, \ldots \lceil t_n \rceil \rangle$$

- 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 $\Sigma = (\mathcal{F}, \mathcal{P})$, Variables X

- $\lceil \neg \varphi \rceil = (\lceil \neg \rceil, \lceil \varphi \rceil)$
- $\lceil \psi \circ \varphi \rceil = \langle \lceil \circ \rceil, \lceil \psi \rceil, \lceil \varphi \rceil \rangle$
- $\lceil Q x_i . \varphi \rceil = \langle \lceil Q \rceil, \lceil x_i \rceil, \lceil \varphi \rceil \rangle$

Lemma 1 (Facts)

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

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

Gödel Kodierung für Ableitungen

- Gödel Kodierung für Hypothesen Liste:

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

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

- Definition von Konkatenation $*$ und Streichen von Hypothesen

Gödel Kodierung für Ableitungen

$$\left\lceil \begin{array}{c} \overset{D_1}{\phi} \quad \overset{D_2}{\psi} \\ \hline \phi \wedge \psi \end{array} \wedge I \right\rceil = \langle (5, \lceil \wedge \rceil), \left\lceil \overset{D_1}{\phi} \right\rceil, \left\lceil \overset{D_2}{\psi} \right\rceil, \lceil \phi \wedge \psi \rceil \rangle$$

$$\left\lceil \begin{array}{c} \overset{D}{\phi \wedge \psi} \\ \hline \phi \end{array} \wedge E_L \right\rceil = \langle (6, \lceil \wedge \rceil), \left\lceil \overset{D}{\phi \wedge \psi} \right\rceil, \lceil \phi \rceil \rangle$$

Gödel Kodierung für Ableitungen

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

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

Gödel Kodierung für Ableitungen

- Basisfall: $\lceil [\phi] \rceil := \langle (4, \lceil \phi \rceil) \rangle$
- Entsprechend für RAA, $\forall I$, $\forall E$
- Definiere $\text{Der}(p, h, z)$: $\lfloor p \rfloor$ ist Beweis für $\lfloor z \rfloor$ aus Hypothesen $\lfloor h \rfloor$

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

Beweisbarkeit

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

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

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

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

- Thm(f): $\lfloor f \rfloor$ ist ein PA Theorem

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

Fixpoint 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 die natürlichen Zahlen zu Formalisieren erlaubt die Formulierung von wahren Aussagen, die weder beweisbar noch widerlegbar sind.

Thm$(f) \longleftrightarrow \exists p.\, \text{Prov}(p, f)$
existiert φ so dass $\vdash \varphi \longleftrightarrow \neg\, \text{Thm}(\lceil \varphi \rceil)$ Fixpoint auf $\neg\, \text{Thm}(f)$
φ bedeutet: "Ich bin nicht beweisbar"

- Es gilt $\mathbb{N} \models \varphi \longleftrightarrow \neg\, \text{Thm}(\lceil \varphi \rceil)$
- Annahme $\mathbb{N} \models \text{Thm}(\lceil \varphi \rceil)$

$\Leftrightarrow \mathbb{N} \models \exists x.\, \text{Prov}(x, \lceil \varphi \rceil)$	$\Leftrightarrow \mathbb{N} \models \text{Prov}(n, \lceil \varphi \rceil)$ for some n
$\Leftrightarrow \vdash \text{Prov}(n, \lceil \varphi \rceil)$ for some n	$\Leftrightarrow \vdash \varphi$
$\Rightarrow \vdash \neg\, \text{Thm}(\lceil \varphi \rceil)$	$\Rightarrow \mathbb{N} \models \neg\, \text{Thm}(\lceil \varphi \rceil)$

- Contradiction, hence φ is true in $\mathbb{N}$, but not provable

Zusammenfassung

- Terminierende Funktionen und abgeleitete Induktionsschemata
  - Hilfreich bei Induktion über Variablen in Argumenten von terminierenden Funktionen um Rekursionsgleichungen anwendbar zu machen
- 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.

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

Formale Modellierung
Vorlesung 9 vom 03.06.13: Weitere Datentypen: Mengen, Multimengen, Punkte

Serge Autexier & Christoph Lüth

Universität Bremen

Sommersemester 2013

Fahrplan

- Teil I: Formale Logik
  - Einführung
  - Aussagenlogik: Syntax und Semantik, Natürliches Schließen
  - Konsistenz & Vollständigkeit der Aussagenlogik
  - Prädikatenlogik (FOL): Syntax und Semantik
  - Konsistenz & Vollständigkeit von FOL
  - FOL mit induktiven Datentypen
  - FOL mit Induktion und Rekursion
  - Die Gödel-Theoreme
  - Weitere Datentypen: Mengen, Multimengen, Punkte
- Teil II: Spezifikation und Verifikation
- Teil III: Schluß

Das Tagesmenü

- Isabelle/HOL: sicheres spezifizieren (konservative Erweiterung)
- typedecl, definition, typedef, datatype, primrec, fun, function
- typedef
  - Mengen, Multimengen, Rat
- datatype
  - Listen, Listen ohne Wiederholung, Punkte & Vektoren 3D

Mengen

- Mengen über beliebigen Typ t: Set_t
- Characterisiert über Prädikat $P : t \to o$

$$\begin{aligned} \mathsf{Collect} &: (t \to o) \to \mathsf{Set}_t \\ \in &: t \times \mathsf{Set}_t \to o \\ x \in (\mathsf{Collect}\, P) &= Px \end{aligned}$$

- Set_t nicht leer wenn Funktionenraum $t \to o$ nicht leer.

$$\begin{aligned} &\bot : t \to o \\ &\forall x_t.\neg\bot(x) \\ &\emptyset_t \in \mathsf{Set}_t \\ &\subseteq : \mathsf{Set}_t \times \mathsf{Set}_t \to o \\ &A \subseteq B \longleftrightarrow \forall x_t.x \in A \longrightarrow x \in B \\ &\mathsf{Power} : \mathsf{Set}_t \to \mathsf{Set}_{\mathsf{Set}_t} \\ &\mathsf{Power}\, A = \underbrace{\mathsf{Collect}(\lambda x_{\mathsf{Set}_t}.x \subseteq A)}_{\{x:\mathsf{Set}_t \mid x \subseteq A\}} \end{aligned}$$

In Isabelle

```
typedecl 'a set
axiomatization Collect :: "('a => bool) => 'a set" -- "comprehension"
  and member :: "'a => 'a set => bool" -- "membership"
where
  mem_Collect_eq [iff, code_unfold]: "member a (Collect P) = P a"
  and Collect_mem_eq [simp]: "Collect (%x. member x A) = A"
```


Definitionen

- Definiere nicht-rekursive Konstanten

$$\mathsf{nand} : o \to o \to o \qquad \mathsf{nand}\, xy = (\neg x) \wedge (\neg y)$$

- Definitionen sind konservative Erweiterungen
- In Isabelle/HOL

```
definition nand :: "bool => bool => bool" where
  "nand x y == (~ x) & ~ y"
```


Typdefinition

- Spezifiziere nicht-leere Teilmenge eines gegebenen Typs r
- Deklariere neuen Typ t mit Trägermenge isomorph zu Werten in spezifizierter Teilmenge
- Isomorphie wird durch inverse Funktionen $\mathsf{Abs}_t : r \to t, \mathsf{Rep}_t : t \to r$ axiomatisch Beschrieben

Typdefinitionen sind Erweiterungen

Definition 1 (Typdefinitionen)

Sei $S = ((\mathcal{T}, \mathcal{F}, \mathcal{P}), \Phi)$ eine Spezifikation, $r \in \mathcal{T}$ und $P \in \mathcal{F}orm_\Sigma$ mit genau einer freien Variable vom Typ r. Dann ist eine Erweiterung $S' = ((\mathcal{T}', \mathcal{F}', \mathcal{P}'), \Phi')$ eine Typdefinition für einen Typ $t \notin \mathcal{T}$ gdw.

- $\mathcal{T}' = \mathcal{T} \cup \{t\}$
- $\mathcal{F}' = \mathcal{F} \cup \{\mathsf{Abs}_t : r \to t, \mathsf{Rep}_t : t \to r\}$
- $\mathcal{P}' = \mathcal{P} \cup \{=_t : t \times t\}$
- $\Phi' = \Phi \cup \{\ \forall x_t.\, \mathsf{Abs}_t(\mathsf{Rep}_t(x)) =_t x,$
  $\forall x_r.P(x_r) \longrightarrow \mathsf{Rep}_t(\mathsf{Abs}_t(x)) =_r x\}$
- Man kann beweisen $S \vdash \exists x_r.P(x)$ (bzw. es gilt $\exists x_r.P(x) \in \mathsf{Th}(S)$)

Beispiel: Paare

- Paare über Typen s und t

$$\text{Pair\_Rep} : s \to t \to (s \to t \to o)$$
$$\text{Pair\_Rep}(a, b) = \lambda x_s \lambda y_t . x = a \wedge y = b$$

- In Isabelle/HOL

```
definition Pair_Rep :: "'a => 'b => 'a => 'b => bool" where
  "Pair_Rep a b = (%x y. x = a & y = b)"

definition "prod = {f.  ex a b. f = Pair_Rep (a::'a) (b::'b)}"

typedef ('a, 'b) prod (infixr "*" 20) =
   "prod :: ('a => 'b => bool) set"

type_notation (xsymbols)
  "prod"  ("(_ x/ _)" [21, 20] 20)
type_notation (HTML output)
  "prod"  ("(_ x/ _)" [21, 20] 20)

definition Pair :: "'a => 'b => 'a 'b prod"  where
  "Pair a b = Abs_prod (Pair_Rep a b)"
```


Datentypen

- Datentypen sind Typdefinition mit Definition der Konstruktoren

```
datatype 'a option = None | Some 'a
```

- Case-Expressions für Datentypen

```
case o of None -> 1 | o of Some(n) -> x
```

- Induktive Datentypen

```
datatype 'a list =
    Nil    ("[]")
  | Cons 'a  "'a list"    (infixr "#" 65)
```

- Generierte Taktiken:
  - `case_tac`: Strukturelle Fallunterscheidung
  - `induct_tac`: Strukturelle Induktion

Rekursive Funktionen

- Drei Möglichkeiten rekursive Funktionen zu definieren:
  - `primrec`
    Strukturelle Fallunterscheidung und Rekursion
  - `fun`
    Totale, terminierende rekursive Funktionen mit Terminierungsbeweis

Erweiterung um Totale, Terminierende Funktionen is Konservativ

Definition 2 (Funktions- und Prädikatsdefinitionen)

Sei $S = ((\mathcal{T}, \mathcal{F}, \mathcal{P}), \Phi)$ eine Spezifikation, $f : \tau_1 \times \cdots \times \tau_n \to \tau_0 \notin \mathcal{F}$ ($\tau_i \in \mathcal{T}$) und $\Psi \in \mathcal{F}orm_{\Sigma \cup \{f\}}$. Dann ist eine Erweiterung $S' = ((\mathcal{T}, \mathcal{F}', \mathcal{P}), \Phi')$ eine Funktionsdefinition gdw.

- Ψ ist eine eindeutig und totale Definition für f
- f ist terminierend und alle in der Definition von f vorkommenden Funktionen und Prädikate sind terminierend
- $\Phi' = \Phi \cup \Psi$
- $\mathcal{F}' = \mathcal{F} \cup \{f : \tau_1 \times \cdots \times \tau_n \to \tau_0\}$

Analog für Prädikatsdefinitionen.

Lemma 3

Funktionsdefinitionen bzw. Prädikatsdefinitionen sind konservativ

Strukturelle Rekursive Funktionen

- case-artige definitions patterns
- Genau eine Schicht Konstruktoren

```
primrec append :: "'a list => 'a list => 'a list" (infixr "@" 65) where
append_Nil: "[] @ ys = ys" |
append_Cons: "(x#xs) @ ys = x # xs @ ys"
```

- Nicht aber

```
primrec myhalf :: "nat => nat" where
"myhalf 0 = 0" |
"myhalf (Suc 0) = 0" |
"myhalf (Suc (Suc x)) = Suc (myhalf x)"
```


Allgemeine Totale rekursive Funktionen

- `fun` zur Definition allgemeiner rekursiver Funktionen

```
fun myhalf :: "nat => nat" where
"myhalf 0 = 0" |
"myhalf (Suc 0) = 0" |
"myhalf (Suc (Suc x)) = Suc (myhalf x)"
```

- Automatischer Terminierungsbeweiser
- Definition wird verweigert, falls Terminierungsbeweis mislingt

Terminierung

- Beispiele:
  - half(x) eine Hypothese pro Rekursionsgleichung
  - fib(x): mehrere Hypothesen pro Rekursionsgleichung
  - gcd(x, y): lexicographische Ordnung
- Beweise alle Hypothesen im Kalkül. Terminierung gilt relativ zur Terminierung der anderen involvierten Funktionen und Prädikate.
- Analog für Prädikate auf $\mathbb{N}$ mit bedingten Äquivalenzen
- Allgemeine Typen: für frei erzeugte Datentypen kann Abbildung in natürliche Zahlen definiert werden, die die Anzahl der Konstruktoren zählt. Damit lässt sich das Terminierungsverfahren auf all frei erzeugten Datentypen erweitern

Quotienten Typen

- Gegeben natürliche Zahlen $\mathbb{N}$
- Rationale Zahl ist Paar nat. Zahlen (n, m) so dass $m \neq 0$

```
definition israt :: "(nat * nat) => bool" where
"israt = (%x . -((snd x) = 0)"
```

- typedef führt nicht zu gewünschtem Ergebnis

```
typedef Rat = "{x . israt(x)}"
```

- Braucht Gleichheit der Paare zusätzlich

```
definition ratrel :: "(nat * nat) =>  (nat * nat) => bool" where
"ratrel =  (%x y .-(snd x = 0) & -(snd y = 0) &
                  snd x * fst y = fst x * snd y)"
```

- Quotienten Typen:

```
quotient_type rat = "nat * nat" / partial: "ratrel"
```

Integers

```
definition intrel :: "(nat * nat) => (nat * nat) => bool" where
  "intrel = (%(x, y) (u, v). x + v = u + y)"

quotient_type int = "nat * nat" / "intrel"
```


Multimengen

```
typedef 'a mset = "{f::'a => nat . True}"
 apply (auto)
done

definition emptyMSet :: "'a mset" where
"emptyMSet = Abs_mset(%x.0)"

definition unionMSet :: "'a mset => 'a mset => 'a mset" where
"unionMSet =
 (%m1 m2. Abs_mset (%n.((Rep_mset m1 n) + (Rep_mset m2 n))))"

definition interMSet :: "'a mset => 'a mset => 'a mset" where
"interMSet =
 (%m1 m2. Abs_mset (%n.(min (Rep_mset m1 n) (Rep_mset m2 n))))"
```


Punkte und Mehr

- Punkte im 3D

```
typedef point3D = "{p::(int * int * int).True}"
 apply (auto)
done

definition origin :: "point3D" where
"origin = (Abs_point3D (0,0,0))"
```

- Vektoren 3D

```
datatype Vektor3D = Vektor int * int * int

definition skalarProdukt :: "Vektor3D => Vektor3D => Vektor3D" where
"skalarProdukt = (% v1 v2 .
  case (v1,v2) of ((Vektor x y z), (Vektor u v w))
    => .."

definition point2Vektor :: "point3D => point3D => Vektor3D" where
...
```


Listen ohne doppelte Vorkommen

- Listen ohne doppelte Vorkommen

```
primrec contains :: "'a list => 'a => bool" where
"contains Nil x = False" |
"contains (Cons x xs) y = ((x = y) or contains xs x)"

fun count :: "'a => 'a list => nat" where
"count x Nil = 0" |
"count x (Cons y xs) = (if (x=y) then (Suc (count x xs))
                        else (count x xs))"


definition noDuplicates :: "'a list => bool" where
"noDuplicates = (%l::'a list.\<forall>x::'a.(contains l x)
                                --> (count x l) = (Suc 0))"

typedef 'a setlist = "{l::'a list . noDuplicates l}"
```


Zusammenfassung

- Isabelle/HOL
- Primitiven vordefiniert zur sicheren Einführung neuer Typen und Funktionen
  - `typedef, datatype, quotient_type`
  - `primrec, fun`
- Typen
  - Paare
  - Option, Listen
  - Rat, Int
  - Multimengen
  - Punkte und Vektoren im 3D
  - Listen ohne doppelte Vorkommen

Formale Modellierung
Vorlesung 10 vom 10.06.13: Modellierung von Programmen

Serge Autexier & Christoph Lüth

Universität Bremen

Sommersemester 2013

Fahrplan

- Teil I: Formale Logik
- Teil II: Spezifikation und Verifikation
  - Modellierung von Programmen
  - Die Z-Notation
  - Formale Modellierung mit der UML und OCL
- Teil III: Schluß

Das Tagesmenü

- Bis jetzt:
  1. Grundlagen — formale Logik
  2. Statische Datentypen
- Heute: Programme
  1. Funktional,
  2. imperativ,
  3. objektorientiert.

Funktionale Programme

- Wir haben:
  1. Logik höherer Stufe
  2. terminierende rekursive Funktionen
  3. induktive (generierte) Datentypen
- Funktionale Programme:
  1. Funktionen höherer Ordnung
  2. Beliebige Rekursion
  3. Beliebige Datentypen

Modellierung beliebiger Rekursion

Definition 1 (Partielle Ordnung)

$(X, \sqsubseteq)$ mit $\sqsubseteq$ antisymmetrisch, reflexiv und transitiv.

Definition 2 (CPO)

$(X, \sqsubseteq)$ mit $\sqsubseteq$ partielle Ordnung und jede Kette C

$$x_1 \sqsubseteq x_2 \sqsubseteq \ldots x_n \sqsubseteq \ldots$$

hat kleinste obere Schranke $\bigsqcup_{i<\omega} x_i$

Definition 3 (Stetige Funktion)

Funktion $f : (X, \sqsubseteq) \to (Y, \preccurlyeq)$ ist Scott-stetig, wenn f die Ordnung $\sqsubseteq$ und $\bigsqcup$ bewahrt.

Eigenschaften von CPOs

Theorem 4 (Kleene-Tarski-Fixpunktsatz)

Sei $f : (X, \sqsubseteq) \to (X, \sqsubseteq)$ Scott-stetig und $\bot \in X$ kleinstes Element, dann hat f einen kleinsten Fixpunkt lfp_f mit $f(lfp_f) = lfp_f$.

- CPOs und stetige Funktionen modellieren funktionale Programme (Logic of computable functions, LCF)
- CPOs modellieren Domänen (Datentypen für funktionale Programme)
- Möglich: konservative Einbettung in HOL (HOLCF)

Ein Beispiel

Beispiel: Flugbuchungssystem

- Ein Flug hat eine eindeutige Flugnummer, Start und Ziel, sowie die Anzahl Sitze (verfügbar und insgesamt).
- Das Flugbuchungssystem identifiziert Flüge anhand ihrer Flugnummer.
- Das Flugbuchungsystem soll folgende Operationen unterstützen:
  - Anfrage nach Verbindung (Start und Ziel) sowie Anzahl Plätze;
  - Buchung mit Flugnummer, Anzahl Plätze

- Modellierung funktional (Isabelle/HOL): axiomatisch vs. konservativ
- Modellierung imperativ: Zustandsübergang

Die Z Notation

- Basiert auf getypter Mengenlehre
- Entwickelt seit 1980 (Jean-Claude Abrial, Oxford PRG)
- Industriell genutzt (IBM, Altran Praxis (vorm. Praxis Critical Systems))
- LaTeX-Notation und Werkzeugunterstützung (Community Z Tools, HOL-Z, ProofPower)

Modellierung in Z (1)

$[FLIGHTID, STRING]$

Flight

$flightid : FLIGHTID$
$dept : STRING$
$arr : STRING$
$avail : \mathbb{N}$
$total : \mathbb{N}$

FlightDB

$flights : \mathbb{P}\, FLIGHTID$
$data : FLIGHTID \pfun Flight$

$flights = \mathrm{dom}\, data$
$\forall i : FLIGHTID \bullet (data(i)).flightid = i$

Modellierung in Z (2)

Lookup

$FlightDB$
$flid? : FLIGHTID$
$flight! : Flight$

$flid? \in flights$
$flight! = data(flid?)$

Modellierung in Z (3)

SuccessfulQuery

$FlightDB$
$from? : STRING$
$to? : STRING$
$seats? : \mathbb{N}$
$fid! : FLIGHTID$

$fid! \in flights$
$(data(fid!)).avail \geq seats?$
$(data(fid!)).dept = from?$
$(data(fid!)).arr = to?$

Modellierung in Z (4)

FailedQuery

$FlightDB$
$from? : STRING$
$to? : STRING$
$seats? : \mathbb{N}$
$fid! : FLIGHTID$

$fid! \notin flights$

Zusammenfassung

- Modellierung funktionaler Programme: cpos (LCF) für beliebige Rekursion
  - Datenmodellierung: erzeugte Datentypen
- Modellierung imperativer Programme: Zustandsübergang
  - Datenmodellierung: als Mengen (Z) oder als Klassen (UML/OCL)
- Nächste Woche: Z im Gesamtüberblick

Formale Modellierung
Vorlesung 11 vom 17.06.13: Die Z-Notation

Serge Autexier & Christoph Lüth

Universität Bremen

Sommersemester 2013

Fahrplan

- Teil I: Formale Logik
- Teil II: Spezifikation und Verifikation
  - Modellierung von Programmen
  - Die Z-Notation
  - Formale Modellierung mit der UML und OCL
- Teil III: Schluß

Das Tagesmenü

- Die Z-Notation
  - Grundlagen
  - Der Schemakalkül
  - Die Bücherei
- Das Beispiel
  - Der sichere autonome Roboter

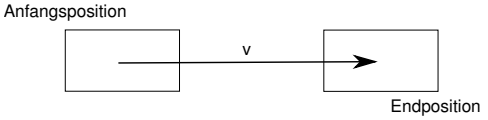
Was ist Z?

- Spezifikationssprache, basierend auf getypter Mengenlehre
  - Alles ist eine Menge (Mengen sind Typen)
  - Viel syntaktische Konvention
- Spezifikation von imperativen Programmen
  - Zustand und Zustandsänderung integraler Bestandteil
- Entwickelt Ende 80er, Oxford UCL und IBM UK

Mengenlehre

- Begründet 1874–1884 durch Georg Cantor (Naive Mengenlehre)
  - Leider inkonsistent (Burali-Forte Paradox, Russelsches Paradox)
- Axiomatisierungen durch Zermelo (inkonsistent), später Fränkel (ZF), von Neumann, Gödel, Bernays
- Axiome:
  Extensionalität, Separation, Paarbildung, Vereinigung, Potenzmenge, Wohlfundiertheit, Ersetzung, Leere Menge, Unendlichkeit, Auswahl
  - Auswahlaxiom unabhängig vom Rest
- Getypte Mengenlehre: HOL
  - Mengen sind Prädikate sind Funktionen nach `bool`

Beispiel: Fahrzeug in der Ebene



- Bremszeit und Bremsstrecke:

$$v(t) = v_0 - a_{brk}\,t \qquad s(t) = v_0 t - \tfrac{a_{brk}}{2} t^2 \qquad T = \tfrac{v_0}{a_{brk}} \qquad S = \tfrac{v_0^2}{2 a_{brk}}$$

- Modellierung in Z: Berechnung des Bremsweges

$$brk : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$$
$$\forall\, v, a : \mathbb{N} \bullet brk\,(v, a) = (v * v) \operatorname{div} (2 * a)$$

Modellierung: Punkte und Polygone

Schematyp für Punkte (Vektoren):

$$\begin{array}{|l} VEC \\ \hline x : \mathbb{Z} \\ y : \mathbb{Z} \\ \hline \end{array}$$

Typ für Polygone und Strecken:

$$POLY == \{s : \operatorname{seq} VEC \mid \#s > 3 \land head\ s = last\ s\}$$
$$SEG == VEC \times VEC$$

Addition und Skalarmultiplikation von Vektoren

$$add : VEC \times VEC \to VEC$$
$$smult : R \times VEC \to VEC$$
$$\forall\, p, q : VEC \bullet add\,(p, q) = (p.x + q.x, p.y + q.y)$$
$$\forall\, n : R;\ p : VEC \bullet smult\,(n, p) = (n * p.x, n * p.y)$$

Mehr zu Punkten und Polygonen

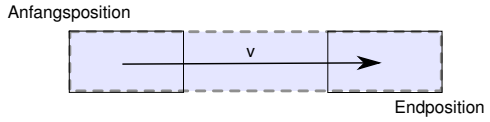
Durch eine Strecke definierte linke Halbebene

$$left : SEG \to \mathbb{P}\, VEC$$
$$\forall\, a, b : VEC \bullet left\,(a, b) = \{p : VEC \mid (b.y - a.y) * (p.x - b.x) - (p.y - b.y) * (b.x - a.x) < 0\}$$

Fläche eines Polygons:

$$sides : POLY \to \mathbb{P}\, SEG$$
$$area : POLY \to \mathbb{P}\, VEC$$
$$\forall\, p : POLY \bullet sides\ p = \{s : SEG \mid \langle s.1, s.2 \rangle \text{ in } p\}$$
$$\forall\, p : POLY \bullet area\ p = \bigcap \{s : SEG \mid s \in sides\ p \bullet left\ s\}$$

Bewegung von Polygonen



Bewegung eines Polygons um einen Vektor

$$move : POLY \times VEC \to POLY$$

$$\forall p : POLY;\ v : VEC \bullet move\,(p, v) = (\lambda x : VEC \bullet add\,(x, v)) \circ p$$

Durch eine Bewegung überstrichene Fläche

$$cov : POLY \times VEC \to \mathbb{P}\, VEC$$

$$\forall p : POLY;\ v : VEC \bullet cov\,(p, v) = \bigcup \{\tau : R \mid 0 \leq \tau \leq 1 \bullet area\,(move\,(p, \tau * v))\}$$

Der Autonome Roboter

RobotParam

$cont : POLY$
$a_{brk} : \mathbb{Z}$
$T : \mathbb{Z}$

World

RobotParam
$obs : \mathbb{P}\, VEC$

Robot

RobotParam
$vel : \mathbb{Z}$
$\omega : \mathbb{Z}$
$v : VEC$
$o : VEC$
$a? : \mathbb{Z}$
$\delta\omega? : \mathbb{Z}$
$c : POLY$

$c = move\,(cont, o)$
$v = cart\,(vel, \omega)$

Der Autonome Roboter in Bewegung

RobotMoves

$\Delta Robot$
$\Xi World$

$vel' = vel + a? * T$
$\omega' = \omega + \delta\omega? * T$
$o' = add\,(o, v')$

- Wann werden Änderungen effektiv?
- Wann ist der Roboter sicher?

Der Autonome Roboter Bremst

- Verhalten beim Bremsen:

RobotBrakes

$\Delta Robot$
$\Xi World$

$vel' = vel - a_{brk} * T$
$\omega' = \omega$
$o' = add\,(o, v')$

- Invariante: Bremsen muß immer sicher sein.

Der Sichere Roboter: Anforderung

- Jetzt und in unmittelbarer Zukunft sicher

RobotSafeMove

RobotMoves

$cov\,(c, v') \cap obs = \emptyset$

- Sicheres Bremsen: Bremsweg frei

RobotSafeBrake

RobotBrakes

$cov\,(c, brk\,(v, \omega, a_{brk})) \cap obs = \emptyset$

- Sicher: Roboter kann sicher fahren oder bremsen

$RobotSafe == RobotSafeMove \lor RobotSafeBrake$

Der Sichere Roboter: Implementation

Sicheres Fahren: Weg ist frei, Bremsweg ist frei

RobotMovesSafely

$\Delta Robot$
$\Xi World$

$(cov\,(c, v') \cup cov\,(move\,(c, v'), brk\,(v', \omega', a_{brk}))) \cap obs = \emptyset$
$vel' = vel + a? * T$
$\omega' = \omega + \delta\omega? * T$
$o' = add\,(o, v')$

Implementation des sicheren Roboterverhaltens:

$RobotDrivesSafely = RobotMovesSafely \lor RobotBrakes$

Zu zeigen:

$RobotDrivesSafely \Rightarrow RobotSafe$
$RobotDrivesSafely \Rightarrow RobotSafe'$

Zusammenfassung

- Z basiert auf getypter Mengenlehre
- Elemente der Sprache:
  - Axiomatische Definitionen
  - Schema, Schemakalkül
  - Mathematical Toolkit (Standardbücherei)
- Modellbasierte und Zustandsbasierte Spezifikationen

Formale Modellierung
Vorlesung 12 vom 24.06.13: Formale Modellierung mit UML und OCL

Serge Autexier & Christoph Lüth

Universität Bremen

Sommersemester 2013

Fahrplan

- Teil I: Formale Logik
- Teil II: Spezifikation und Verifikation
  - Modellierung von Programmen
  - Die Z-Notation
  - Formale Modellierung mit der UML und OCL
- Teil III: Schluß

Das Tagesmenü

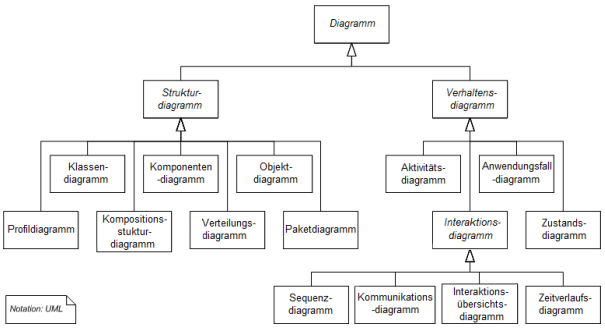
	Datenmodellierung	Programmbegriff
Isabelle/HOL	induktive Datentypen	totale rekursive Funktionen
Z	(induktive) Mengen	Vor/Nachzustand
UML	?	?

- Formale Modellierung mit der UML
- . . . mit der OCL: Object Constraint Language

UML als formale Spezifikationssprache

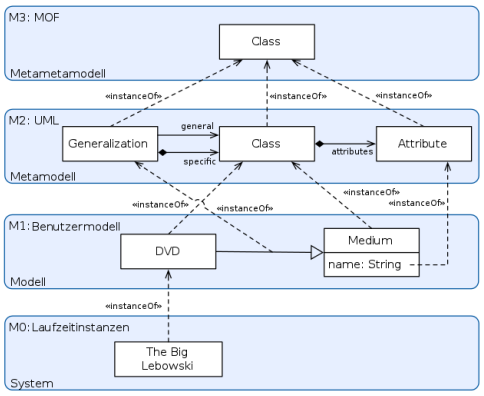
Diagrammtyp	Modellierte Aspekte	Formal
Klassendiagramm	Statische Systemstruktur	Ja
Paketdiagramm	Pakete, Namensräume	Nein
Objektdiagramm	Zustand von Objekten	(Ja)
Kompositionsstrukturdiagramm	Kollaborationen	Nein
Komponentendiagramm	Dynamische Systemstruktur	(Nein)
Verteilungsdiagramm	Implementierungsaspekte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

Semantik der UML: Metamodellierung



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) → OCL-Std. §7.5.11
- 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` ($\mathbb{Z}$) → OCL-Std. §11.5.2
- `Real` ($\mathbb{R}$) → OCL-Std. §11.5.1
  - `Integer` Subklasse von `Real`
  - `round`, `floor` von `Real` nach `Integer`
- `String` (Zeichenketten) → OCL-Std. §11.5.3
  - `substring`, `toReal`, `toInteger`, `characters` etc.
- `Boolean` (Wahrheitswerte) → OCL-Std. §11.5.4
  - `or`, `xor`, `and`, `implies`
  - Sowie Relationen auf `Real`, `Integer`, `String`

Collection Types

- `Set`, `OrderedSet`, `Bag`, `Sequence`
- Operationen auf allen Kollektionen: → OCL-Std. §11.7.1
  - `size`, `includes`, `count`, `isEmpty`, `flatten`
  - Kollektionen werden immer flachgeklopft
- `Set` → OCL-Std. §11.7.2
  - `union`, `intersection`,
- `Bag` → OCL-Std. §11.7.3
  - `union`, `intersection`, `count`
- `Sequence` → OCL-Std. §11.7.4
  - `first`, `last`, `reverse`, `prepend`, `append`

Collection Types: Iteratoren

- Iteratoren: Funktionen höherer Ordnung
- Alle definiert über `iterate` → OCL-Std. §7.7.6:

```
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

Modelltypen

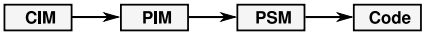
- Aus Attribute, Operationen, Assoziationen des Modells
- Navigation entlang der Assoziationen
- Für Kardinalität 1 Typ `T`, sonst `Set(T)`
- Benutzerdefinierte Operationen in Ausdrücken müssen zustandsfrei sein (Stereotyp `<<query>>`)

Style Guide

- Komplexe Navigation vermeiden (“Loose coupling”)
- Adäquaten Kontext auswählen
- “Use of allInstances is discouraged”
- Invarianten aufspalten
- Hilfsoperationen definieren

MDA + OCL

- MDA: Model-driven architecture
- Entwicklung durch Modelltransformation



- Rolle der OCL:
  - Metasprache
  - Codegenerierung
  - Laufzeitchecks
- Beispiele für Werkzeuge: MDT/OCL
  - MDT/OCL: EMF mit OCL-Unterstützung

Zusammenfassung

- Kritik UML:
  - “OO built-in”
  - Adäquat für eingebettete Systeme, CPS, . . . ?
- OCL erlaubt Einschränkungen auf Modellen
- Erlaubt mathematisch präzisere Modellierung
- Frage:
  - Werkzeugunterstützung?
  - Ziel: Beweise, Codegenerierung, . . . ?

Formale Modellierung
Vorlesung 13 vom 01.07.13: Rückblick und Ausblick

Serge Autexier & Christoph Lüth

Universität Bremen

Sommersemester 2013

Fahrplan

- Teil I: Formale Logik
- Teil II: Spezifikation und Verifikation
- Teil III: Schluß
  - Rückblich & Ausblick

Das Tagesmenü

- Zur Abrundung: JML
- Rückblick
- Ausblick

JML

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()
}
```


JML: Wichtigste Schlüsselworte

- Für Klassen und Interfaces:
  - `invariant`
  - `spec_public`
  - `nullable`
- Für Methoden und Konstruktoren:
  - `requires`
  - `ensures`
  - `assignable`
  - `pure`
- Im Code:
- `assert`
- `invariant`, `assignable`

JML: Längeres Beispiel

Zusammenfassung JML

- Modellierung der Daten durch Java
- Zusätzliche Annotation der Programme mit Korrektheitsbedingungen
- Dadurch leichtgewichtig — erleichtert Einführung
- Benefits:
  - Generierung von Testfällen
  - Statische Analyse ("erweiterte Typprüfung")
  - Korrektheitsbeweis (automatisch oder interaktiv)
- Nachteile:
  - Umständliche Modellierung — kein Abstraktionsgewinn

Rückblick

Logiken und Spezifikationsformalismen

	Datenmodellierung	Programmbegriff
Isabelle/HOL	induktive Datentypen	totale rekursive Funktionen
Z	(induktive) Mengen	Vor/Nachzustand[1]
UML	Klassendiagramme	OCL
JML	Java	Vor/Nachzustand, Invarianten

[1] Invarianten durch Ξ

Unsere Reise durch die Logik

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

Ausblick

Wohin von hier?

- Verifikation von Hardware und Software
  - Formalisierung von Hardware oder Software (Programmiersprache)
  - Nachweis der Eigenschaften wird Beweis
- Weitere Ausdrucksmächtigkeit:
  - Nebenläufigkeit: Prozesskalküle (CSP), Modallogik
  - Zeitliche Aspekte (Temporallogik)

Prozesskalküle (Prozessalgebren)

- Modellierung nebenläufige Systeme
- Werkzeugunterstützung: Modelchecker FDR
- Beispiel (CSP): ein Flugbuchungssystem

$$\begin{aligned} SERVER = \;& query \to result \to SERVER \\ & \Box\ booking \to (ok \to SERVER \sqcap fail \to SERVER) \\ & \Box\ cancel \to ok \to SERVER \end{aligned}$$

$$\begin{aligned} CLIENT = query \to result \to \;& (booking \to \; (ok \to CLIENT \\ & \qquad \Box\ fail \to CLIENT) \\ & \sqcap\ CLIENT) \end{aligned}$$

$$SYSTEM = CLIENT \parallel SERVER$$

Problem: Deadlock

Temporale Logiken

- System wird beschrieben als FSM
  - Ggf. mit Übergangszeiten
  - Graphische Notation oder DSL (Promela für SPIN)
- Spezifikation als temporallogische Formel:
  - "Irgendwann muss ein Event HALT (H) auftreten": $\mathbf{F}H$
  - "Es darf nie ein STOP (S) ohne eine WARNUNG (W) zuvor auftreten":

$$\mathbf{G}((W \longrightarrow \mathbf{F}S) \vee \neg S)$$

- Lineare vs. baumartige Zeit (LTL, CTL), TLA
- Werkzeuge: SPIN, UPPAAL

Fazit

Dafür und Dagegen

Was spricht für formale Modellierung in der Entwicklung?

- Anforderungen werden eindeutig formuliert
- Randbedingungen werden seltener (gar nicht) übersehen
- Fehler werden früher gefunden
- Werkzeugunterstützung möglich
  - Von erweitertem Typcheck bis automatischen/interaktiven Beweis
- Normen (IEC 61508:3, CENELEC EN50128, DO 178B) fordern formale Methoden für hohe Sicherheitsstufen (SIL 3, Level B)

Was spricht gegen formale Modellierung?

- Höherer Zeitaufwand
- Qualifikation des Personals
- Verlust an Agilität

Zusammenfassung

Formale Modellierung

Beschreibung der Welt durch Mittel der mathematischen Logik

- Beispiele: HOL, die Z Notation, UML/OCL, JML
- Vorteile:
  - Spezifikationen eindeutig formuliert
  - Nachweis von Eigenschaften möglich
  - Formale Verifikation möglich