Formale Modellierung
Vorlesung 1 vom 24.04.14: Einführung

Serge Autexier & Christoph Lüth

Universität Bremen

Sommersemester 2014

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 1100
  Donnerstag, 14 – 16, MZH 1100

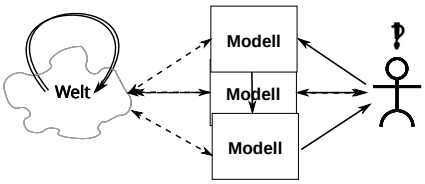
- Webseite:

Ariane-5

Die Vasa

10. August 1628

Modellierung — Das Prinzip



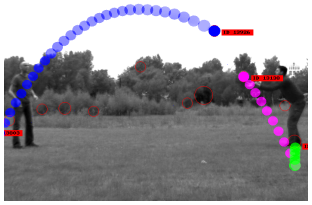
- Grundlegendes Prinzip der Naturwissenschaften

Modellierung — Beispiele



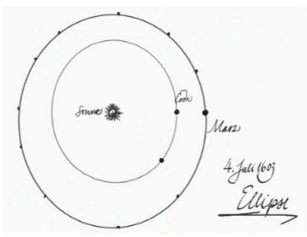
$$2Mg \quad + \quad O_2 \quad \longrightarrow \quad 2MgO$$

Modellierung — Beispiele



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

Modellierung — Beispiele



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

Arten der Modellierung

- Computer — diskrete Mathematik, formale Logik
- Physikalische Systeme — kontinuierliche Mathematik, DGL
- Eingebette Systeme (CPS) — beides

Lernziele

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

Themen

- Formale Logik:
  - Aussagenlogik ($A \wedge B$, $A \longrightarrow B$), Prädikatenlogik ($\forall x.P$)
  - Formales Beweisen: natürliches Schließen
  - Induktion, induktive Datentypen, Rekursion
  - Die Gödel-Theoreme
- Spezifikation und Verifikation:
  - Formale Modellierung mit der UML und OCL
  - Temporale Logik
  - Hybride Systeme

Der Theorembeweiser Isabelle

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

Formale Logik

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

Geschichte

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

Grundbegriffe der formalen Logik

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

Unvollständigkeit

- Gödels 1. Unvollständigkeitssatz:
  - Jede Logik, die Peano-Arithmetik formalisiert, ist entweder inkonsistent oder unvollständig.
- 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 28.04.14: Aussagenlogik und natürliches Schließen

Serge Autexier & Christoph Lüth

Universität Bremen

Sommersemester 2014

Organisatorisches

- Montagstermin?
- Keine Übung am Donnerstag (01. Mai)
- Dafür Übung nächsten Montag (05. Mai)
- Nächste VL am Donnerstag (08. Mai)

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
  - Beschreibungslogiken
  - FOL mit induktiven Datentypen
  - FOL mit Induktion und Rekursion
  - Die Unvollständigkeitssätze von Gödel
- Teil II: Spezifikation und Verifikation

Formalisierung von Aussagen

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

Formale Logik

- Ziel: Formalisierung von Folgerungen wie
  - Wenn es regnet, wird die Straße nass.
  - 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) $V \subseteq \mathcal{P}rop$ (Menge V gegeben)
  2. $\bot \in \mathcal{P}rop$
  3. Wenn $\phi, \psi \in \mathcal{P}rop$, dann
     - $\phi \wedge \psi \in \mathcal{P}rop$
     - $\phi \vee \psi \in \mathcal{P}rop$
     - $\phi \longrightarrow \psi \in \mathcal{P}rop$
     - $\phi \longleftrightarrow \psi \in \mathcal{P}rop$
  4. Wenn $\phi \in \mathcal{P}rop$, dann $\neg\phi \in \mathcal{P}rop$.
- NB. Präzedenzen: $\neg$ vor $\wedge$ vor $\vee$ vor $\longrightarrow, \longleftrightarrow$

Wann ist eine Formel gültig?

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

Semantik

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

Definition (Semantik aussagenlogischer Formeln)

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

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

Semantische Gültigkeit und Folgerung

- Semantische Gültigkeit: $\models \phi$

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

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

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

Beweisen mit semantischer Folgerung

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

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

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

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

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

Die fehlenden Konnektive

- Einführung als Abkürzung:

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

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

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

- Ableitungsregeln als Theoreme.

Die fehlenden Schlußregeln

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

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

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

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

Formale Modellierung
Vorlesung 3 vom 05.05.14: Konsistenz & Vollständigkeit der Aussagenlogik

Serge Autexier & Christoph Lüth

Universität Bremen

Sommersemester 2014

Organisatorisches

- Übung am Donnerstag 08.05.14 muss ausfallen.
- Ersatztermin?

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
  - Beschreibungslogiken
  - FOL mit induktiven Datentypen
  - FOL mit Induktion und Rekursion
  - Die Unvollständigkeitssätze von Gödel
- Teil II: Spezifikation und Verifikation

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:

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

- $(\wedge, \neg)$ und $(\vee, \bot)$ sind ausreichend (functional complete)
- Anwendung: konjunktive und disjunktive Normalformen (CNF/DNF)
- Ein Operator reicht: $A \mid B$ (Sheffer-Strich), $A \downarrow B$ (weder-noch)

Korrektheit (Soundness)

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

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

Beweis: Induktion über der Ableitung $\Gamma \vdash \phi$
- Nützliches Korollar: $\Gamma \not\models \phi$ dann $\Gamma \nvdash \phi$

Konsistenz

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

Definition 2 (Konsistenz)

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

Lemma 3 (Charakterisierung von Konsistenz)

Folgende Aussagen sind äquivalent:

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

Definition 4 (Maximale Konsistenz)

Γ maximal konsistent gdw.
(i) Γ konsistent, und
(ii) wenn $\Gamma \subseteq \Gamma'$ und Γ' konsistent, dann $\Gamma = \Gamma'$

Lemma 5 (Konstruktion maximal konsistenter Mengen)

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

Eigenschaften maximal konsistenter Mengen

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

Lemma 6

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

- Wenn Γ maximal konsistent ist, dann:

(i) entweder $\phi \in \Gamma$ oder $\neg\phi \in \Gamma$
(ii) $\phi \wedge \psi \in \Gamma$ gdw. $\phi, \psi \in \Gamma$
(iii) $\phi \longrightarrow \psi \in \Gamma$ gdw. (wenn $\phi \in \Gamma$ dann $\psi \in \Gamma$)

Vollständigkeit

Lemma 7

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

Damit:

- Wenn $\Gamma \nvdash \phi$ dann gibt es v so dass $[\![\psi]\!]_v = 1$ für $\psi \in \Gamma$, $[\![\phi]\!]_v = 0$.
- Wenn $\Gamma \nvdash \phi$ dann $\Gamma \not\models \phi$.

Theorem 8 (Vollständigkeit 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: Prädikatenlogik

Formale Modellierung
Vorlesung 4 vom 12.05.14: Prädikatenlogik erster Stufe

Serge Autexier & Christoph Lüth

Universität Bremen

Sommersemester 2014

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
  - Beschreibungslogiken
  - FOL mit induktiven Datentypen
  - FOL mit Induktion und Rekursion
  - Die Unvollständigkeitssätze von Gödel
- Teil II: Spezifikation und Verifikation

Das Tagesmenü

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

Beschränkungen der Aussagenlogik

- Beschränkung der Aussagenlogik:
  - Eine Zahl n ist eine Primzahl genau dann wenn sie nicht 1 ist und nur durch 1 und sich selbst teilbar ist.
  - Eine Zahl m ist durch eine Zahl n teilbar genau dann wenn es eine Zahl p gibt, so dass $m = n \cdot p$.
  - Nicht in Aussagenlogk formalisierbar.
- Ziel: Formalisierung von Aussagen wie
  - Alle Zahlen sind ein Produkt von Primfaktoren.
  - Es gibt keine größte Primzahl.

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: $\vee, \longleftrightarrow, \neg, \longleftrightarrow, \exists$
- Die Trennung zwischen Termen und Formeln ist der wesentliche Abstraktionsschritt in der Prädikatenlogik.

Terme

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

Formeln

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

Freie und gebundene Variable

Definition (Freie und gebundene Variablen)

Variablen in $t \in \mathcal{T}erm, p \in \mathcal{F}orm$ sind frei, gebunden, oder bindend:

(i) x bindend in $\forall x.\phi$, $\exists x.\psi$

(ii) Für $\forall x.\phi$ und $\exists x.\phi$ ist x in Teilformel ϕ gebunden

(iii) Ansonsten ist x frei

- $FV(\phi)$: Menge der freien Variablen in ϕ
- Beispiel:
$$(q(x) \vee \exists x.\forall y.p(f(x), z) \wedge q(a)) \vee \forall r(x, z, g(x))$$
- Formel (Term) s geschlossen, wenn $FV(s) = \emptyset$
- Abschluss einer Formel: $Cl(\phi) = \forall z_1 \ldots z_k.\phi$ für $FV(\phi) = \{z_1, \ldots, z_k\}$

Semantik: Strukturen

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

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

(i) A nicht-leere Menge (Universum)

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

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

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

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

Semantische Gültigkeit

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

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

$$[\![P(t_1, \ldots, t_n)]\!]_{\mathfrak{A}} = \begin{cases} 1 & \langle [\![t_1]\!]_{\mathfrak{A}}, \ldots, [\![t_n]\!]_{\mathfrak{A}} \rangle \in P_{\mathfrak{A}} \\ 0 & \text{sonst} \end{cases}$$

$$[\![t_1 \doteq t_2]\!]_{\mathfrak{A}} = \begin{cases} 1 & [\![t_1]\!]_{\mathfrak{A}} = [\![t_2]\!]_{\mathfrak{A}} \\ 0 & \text{sonst} \end{cases}$$

$$\begin{array}{rl} [\![\forall x.\phi]\!]_{\mathfrak{A}} & = \min(\{[\![\phi[^{\overline{a}}_x]]\!]_{\mathfrak{A}} \mid a \in A\}) \\ [\![\exists x.\phi]\!]_{\mathfrak{A}} & = \max(\{[\![\phi[^{\overline{a}}_x]]\!]_{\mathfrak{A}} \mid a \in A\}) \end{array}$$

- 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:

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

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 19.05.14: Eigenschaften der Prädikatenlogik erster Stufe

Serge Autexier & Christoph Lüth

Universität Bremen

Sommersemester 2014

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
  - Beschreibungslogiken
  - FOL mit induktiven Datentypen
  - FOL mit Induktion und Rekursion
  - Die Unvollständigkeitssätze von Gödel
- Teil II: Spezifikation und Verifikation

Das Tagesmenü

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

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

Vorbereitende Definitionen

Definition 2 (Theorien, Henkin-Theorien)

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

(ii) Henkin-Theorie: Für jedes $\exists x.\phi(x) \in T$ gibt es Witness c mit $\exists x.\phi(x) \longrightarrow \phi(c) \in T$

Definition 3

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

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

Maximal konsistente Theorien

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* $\Delta \subseteq \Gamma$ *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 26.05.14: Beschreibungslogiken

Serge Autexier & Christoph Lüth

Universität Bremen

Sommersemester 2014

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
  - Beschreibungslogiken
  - FOL mit induktiven Datentypen
  - FOL mit Induktion und Rekursion
  - Die Unvollständigkeitssätze von Gödel
- Teil II: Spezifikation und Verifikation

Beschreibungslogiken

- Entscheidbare Fragmente von FOL
- Zusammenhang zu Notation
- Beschreibungslogik, ALC Logik
- ND Kalkül
- Korrektheit & Vollständigkeit
- Logik ALCQI
- Anwendung
- ND Kalkül

Entscheidbare Fragmente

- Aussagenlogik

$$\begin{aligned}\mathcal{F}orm_\Sigma := \bot|\top|A|\neg\mathcal{F}orm_\Sigma\\ |\mathcal{F}orm_\Sigma \wedge \mathcal{F}orm_\Sigma|\mathcal{F}orm_\Sigma \vee \mathcal{F}orm_\Sigma\\ |\mathcal{F}orm_\Sigma \longrightarrow \mathcal{F}orm_\Sigma|\mathcal{F}orm_\Sigma \longleftrightarrow \mathcal{F}orm_\Sigma\end{aligned}$$

- Beschreibungslogik
  - Nur ein- und zweistellige Prädikate,
  - Nur 2 Variablen für Quantoren linear verwendet, nur Konstanten für Termen
  - $\longrightarrow$ und $\longleftrightarrow$ nie unterhalb von anderen Konnektiven
- Prädikatenlogik

$$\begin{aligned}\mathcal{T}erm_\Sigma :=& f(\mathcal{T}erm_\Sigma,\ldots,\mathcal{T}erm_\Sigma)\\ \mathcal{F}orm_\Sigma :=& \bot|\top|P(\mathcal{T}erm_\Sigma,\ldots,\mathcal{T}erm_\Sigma)|\neg\mathcal{F}orm_\Sigma\\ &|\mathcal{F}orm_\Sigma \wedge \mathcal{F}orm_\Sigma|\mathcal{F}orm_\Sigma \vee \mathcal{F}orm_\Sigma\\ &|\mathcal{F}orm_\Sigma \longrightarrow \mathcal{F}orm_\Sigma|\mathcal{F}orm_\Sigma \longleftrightarrow \mathcal{F}orm_\Sigma\\ &|\forall x.\mathcal{F}orm_\Sigma|\exists x.\mathcal{F}orm_\Sigma\end{aligned}$$

Beschreibungslogik

- Nur ein- und zweistellige Prädikate,

  Parent($Steve$)

  hasChild($Steve$, $John$)
- Nur 2 Variablen für Quantoren linear verwendet, nur Konstanten für Termen

  $\forall x.\,\mathsf{Parent}(x) \longleftrightarrow \mathsf{Human}(x) \wedge$
  $\exists y.\,\mathsf{hasChild}(x,y) \wedge \mathsf{Human}(y)$
- $\longrightarrow$ und $\longleftrightarrow$ nie unterhalb von anderen Konnektiven

- Nur ein- und zweistellige Prädikate,

  $\underbrace{\mathsf{Parent}}_{\text{Konzepte}}(Steve), \underbrace{\mathsf{hasChild}}_{\text{Rollen}}(Steve, John)$
- Nur 2 Variablen für Quantoren linear verwendet, nur Konstanten für Termen

  $\mathsf{Parent} \equiv \mathsf{Human} \sqcap$
  $\exists\,\mathsf{hasChild}\,.\,\mathsf{Human}$
- $\top, \bot, \wedge, \vee, \longrightarrow$ und $\longleftrightarrow$ werden zu $\top, \bot, \sqcap, \sqcup, \sqsubseteq$ und $\equiv$

ALC-Formalisierungen

- Menge aller ALC-Formeln ist ϕ_c
- Wird verwendet um Weltwissen zu beschreiben
- Grundlage von OWL, RDF (Semantic Web)
- Werkzeugunterstützung Protégé zum Beispiel
- Formalisierung besteht aus Terminologie (TBOX) und Annahmen (Assertions, ABOX):
  - TBOX:
    - Inklusionen $C \sqsubseteq D$
    - Definitionen $C \equiv \alpha$, C Name
    - Es darf maximal eine Definition für einen Namen geben
  - ABOX:

$$Parent(Steve), hasChild(Steve, John)$$

Beispiel TBOX

$$\begin{aligned}Man &\sqsubseteq Human\\ Woman &\sqsubseteq Human\\ \mathsf{Parent} &\equiv \mathsf{Human} \sqcap \exists\,\mathsf{hasChild}\,.\,\mathsf{Human}\\ Father &\equiv Parent \sqcap Man\\ Mother &\equiv Parent \sqcap Woman\end{aligned}$$

Familie von Beschreibungslogiken

- ALC: nur atomare Rollen
- ALCN: Zahleneinschränkungen für Rollen, unqualifiziert

$$\leq nR, \geq nR$$

- ALCQ: Zahleneinschränkungen für Rollen, qualifiziert

$$\leq nR.C, \geq nR.C$$

- ALCI: Inverse Rollen

$$\forall R^-.C, \exists R^-.C, \ldots$$

Semantik

Interpretation $\mathcal{I} = (\Delta^{\mathcal{I}}, \_^{\mathcal{I}})$

- $\Delta^{\mathcal{I}}$ domäne (Universum), nicht-leer.
- $\_^{\mathcal{I}}$ Abbildung von
  - Individuen auf Elemente von $\Delta^{\mathcal{I}}$,
  - Konzepten auf Teilmengen von $\Delta^{\mathcal{I}}$,
  - Rollen auf Teilmengen von $\Delta^{\mathcal{I}} \times \Delta^{\mathcal{I}}$

Abbildung

$$
\begin{aligned}
\top^{\mathcal{I}} &= \Delta^{\mathcal{I}} \\
\bot^{\mathcal{I}} &= \emptyset \\
(\neg C)^{\mathcal{I}} &= \Delta^{\mathcal{I}} \setminus C^{\mathcal{I}} \\
(C \sqcap D)^{\mathcal{I}} &= C^{\mathcal{I}} \cap D^{\mathcal{I}} \\
(C \sqcup D)^{\mathcal{I}} &= C^{\mathcal{I}} \cup D^{\mathcal{I}} \\
(\exists R.C)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} \mid \exists b.(a,b) \in R^{\mathcal{I}} \wedge b \in C^{\mathcal{I}}\} \\
(\forall R.C)^{\mathcal{I}} &= \{a \in \Delta^{\mathcal{I}} \mid \forall b.(a,b) \in R^{\mathcal{I}} \rightarrow b \in C^{\mathcal{I}}\}
\end{aligned}
$$

$(\leq nR)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid |\{b \mid (a,b$

$(\geq nR)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid |\{b \mid (a,b$

$(\leq nR.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid |\{b \mid (a,b$

$(\geq nR.C)^{\mathcal{I}} = \{a \in \Delta^{\mathcal{I}} \mid |\{b \mid (a,b$

Modell

Sei $\mathcal{I} = (\Delta^{\mathcal{I}}, \_^{\mathcal{I}})$ eine Interpretation.

- $\mathcal{I} \models C(a)$ gdw. $a^{\mathcal{I}} \in C^{\mathcal{I}}$
- $\mathcal{I} \models R(a,b)$ gdw. $(a^{\mathcal{I}}, b^{\mathcal{I}}) \in R^{\mathcal{I}}$

ND Kalkül für ALC

Alexandre Rademaker. *A Proof Theory for Description Logics*, PhD Thesis, PUC-Rio, Bresil, March 2010

Axiomatisierung von ALC

$$
\begin{aligned}
\forall R.(\alpha \sqcap \beta) &\equiv \forall R.\alpha \sqcap \forall R.\beta && (1)\\
\forall R.\top &\equiv \top && (2)\\
\exists R.(\alpha \sqcup \beta) &\equiv \exists R.\alpha \sqcup \exists R.\beta && (3)\\
\exists R.\bot &\equiv \bot && (4)
\end{aligned}
$$

- Einige Fakten
  - Falls $\vdash \alpha$ gilt, dann auch $\vdash \forall R.\alpha$ (Necessitation)
  - If $C \sqsubseteq D$ then $\exists R.C \sqsubseteq \exists R.D$
  - If $C \sqsubseteq D$ then $\forall R.C \sqsubseteq \forall R.D$

Labelled Formel

$$
\begin{aligned}
L &:= \forall R, L \mid \exists R, L \mid \epsilon \\
\phi_{lc} &:= {}^{L}\phi c
\end{aligned}
$$

Aus labelled Formel kann immer die normale Formel wieder berechnet werden

$$
\begin{aligned}
\sigma({}^{\epsilon}\alpha) &= \alpha \\
\sigma({}^{\forall R, L}\alpha) &= \forall R.\sigma({}^{L}\alpha) \\
\sigma({}^{\exists R, L}\alpha) &= \exists R.\sigma({}^{L}\alpha)
\end{aligned}
$$

- Notation

$${}^{L\forall}\alpha, {}^{L\exists}\alpha,$$

Wenn alle Labels der Form $\forall R$ bzw. $\exists R$ sind

Kalkül des natürlichen Schließen für ALC

$$\frac{{}^{L\forall}(\alpha \sqcap \beta)}{{}^{L\forall}\alpha}\ \sqcap\text{-e} \qquad \frac{{}^{L\forall}(\alpha \sqcap \beta)}{{}^{L\forall}\beta}\ \sqcap\text{-e} \qquad \frac{{}^{L\forall}\alpha \quad {}^{L\forall}\beta}{{}^{L\forall}(\alpha \sqcap \beta)}\ \sqcap\text{-i}$$

$$\frac{{}^{L\exists}(\alpha \sqcup \beta) \quad \begin{matrix}[{}^{L\exists}\alpha] \\ \vdots \\ \gamma\end{matrix} \quad \begin{matrix}[{}^{L\exists}\beta] \\ \vdots \\ \gamma\end{matrix}}{\gamma}\ \sqcup\text{-e} \qquad \frac{{}^{L\exists}\alpha}{{}^{L\exists}(\alpha \sqcup \beta)}\ \sqcup\text{-i} \qquad \frac{{}^{L\exists}\beta}{{}^{L\exists}(\alpha \sqcup \beta)}\ \sqcup\text{-i}$$

Kalkül des natürlichen Schließen für ALC

$$\frac{{}^{L}\alpha \quad {}^{\neg L}\neg\alpha}{\bot}\ \neg\text{-e} \qquad \frac{\begin{matrix}[{}^{L}\alpha] \\ \vdots \\ \bot\end{matrix}}{{}^{\neg L}\neg\alpha}\ \neg\text{-i} \qquad \frac{\begin{matrix}[{}^{\neg L}\neg\alpha] \\ \vdots \\ \bot\end{matrix}}{{}^{L}\alpha}\ \bot_c$$

$$\frac{{}^{L}\exists R.\alpha}{{}^{L,\exists R}\alpha}\ \exists\text{-e} \qquad \frac{{}^{L,\exists R}\alpha}{{}^{L}\exists R.\alpha}\ \exists\text{-i} \qquad \frac{{}^{L}\forall R.\alpha}{{}^{L,\forall R}\alpha}\ \forall\text{-e}$$

Kalkül des natürlichen Schließen für ALC

$$\frac{{}^{L,\forall R}\alpha}{{}^{L}\forall R.\alpha}\ \forall\text{-i} \qquad \frac{{}^{L_1}\alpha \quad {}^{L_1}\alpha \sqsubseteq {}^{L_2}\beta}{{}^{L_2}\beta}\ \sqsubseteq\text{-e} \qquad \frac{\begin{array}{c}[{}^{L_1}\alpha]\\ \vdots\\ {}^{L_2}\beta\end{array}}{{}^{L_1}\alpha \sqsubseteq {}^{L_2}\beta}\ \sqsubseteq\text{-i}$$

$$\frac{{}^{L}\alpha}{{}^{\forall R,L}\alpha}\ Gen$$

Korrektheit & Vollständigkeit

- $ND_{\mathcal{ALC}}$ ist korrekt
- $ND_{\mathcal{ALC}}$ ist vollständig
- Gegeben Annahmen T und zu beweisende ALC Formel α und ein voll-expandierter ND-Ableitungsbaum P:
  - Falls P kein Beweis ist, dann kann daraus ein Gegenbeispiel für $T \vdash \alpha$ extrahiert werden.
  - Entscheidbarkeit

Die Logik ALCQI

Familie von Beschreibungslogiken

- ALC: nur atomare Rollen
- ALCN: Zahleneinschränkungen für Rollen, unqualifiziert

$$\leq nR, \geq nR$$

- ALCQ: Zahleneinschränkungen für Rollen, qualifiziert

$$\leq nR.C, \geq nR.C$$

- ALCI: Inverse Rollen

$$\forall R^{-}.C, \exists R^{-}.C, \ldots$$

Die Logik ALCQI

- Konzepte und Rollen

$$\alpha := \bot | A | \neg\alpha | \alpha_1 \sqcap \alpha_2 | \alpha_1 \sqcup \alpha_2 | \forall P.\alpha | \exists P.\alpha | \leq nP.\alpha \geq nP.\alpha$$
$$P := R | R^{-}$$

- TBox wie gehabt, ABox auch
- Labeled Formeln

$$L := \forall P, L | \exists P, L | \leq nP, L | \geq nP, L | \epsilon$$
$$\phi_{cl} := {}^{L}\phi_c$$

Kalkül des natürlichen Schließen für ALCQI

$$\frac{{}^{L^{\forall\geq}}(\alpha \sqcap \beta)}{{}^{L^{\forall\geq}}\alpha}\ \sqcap\text{-e} \qquad \frac{{}^{L^{\forall\geq}}(\alpha \sqcap \beta)}{{}^{L^{\forall\geq}}\beta}\ \sqcap\text{-e} \qquad \frac{{}^{L^{\forall\leq}}\alpha \quad {}^{L^{\forall\leq}}\beta}{{}^{L^{\forall\leq}}(\alpha \sqcap \beta)}\ \sqcap\text{-i}$$

$$\frac{{}^{L^{\exists\leq}}(\alpha \sqcup \beta) \quad \begin{array}{c}[{}^{L^{\exists\leq}}\alpha]\\ \vdots\\ \gamma\end{array} \quad \begin{array}{c}[{}^{L^{\exists\leq}}\beta]\\ \vdots\\ \gamma\end{array}}{\gamma}\ \sqcup\text{-e} \qquad \frac{{}^{L^{\exists\geq}}\alpha}{{}^{L^{\exists\geq}}(\alpha \sqcup \beta)}\ \sqcup\text{-i} \qquad \frac{{}^{L^{\exists\geq}}\beta}{{}^{L^{\exists\geq}}(\alpha \sqcup \beta)}\ \sqcup\text{-i}$$

Kalkül des natürlichen Schließen für ALCQI

$$\frac{{}^{L^{\forall\exists}}\alpha \quad \neg{}^{L^{\forall\exists}}\neg\alpha}{\bot}\ \neg\text{-e} \qquad \frac{\begin{array}{c}[{}^{L^{\forall\exists}}\alpha]\\ \vdots\\ \bot\end{array}}{\neg{}^{L^{\forall\exists}}\neg\alpha}\ \neg\text{-i} \qquad \frac{\begin{array}{c}[\neg{}^{L^{\forall\exists}}\neg\alpha]\\ \vdots\\ \bot\end{array}}{{}^{L^{\forall\exists}}\alpha}\ \bot_c$$

$$\frac{{}^{L}\exists R.\alpha}{{}^{L,\exists R}\alpha}\ \exists\text{-e} \qquad \frac{{}^{L,\exists R}\alpha}{{}^{L}\exists R.\alpha}\ \exists\text{-i} \qquad \frac{{}^{L}\forall R.\alpha}{{}^{L,\forall R}\alpha}\ \forall\text{-e}$$

Kalkül des natürlichen Schließen für ALCQI

$$\frac{{}^{L,\forall R}\alpha}{{}^{L}\forall R.\alpha}\ \forall\text{-i} \qquad \frac{{}^{L}\leq nR.\alpha}{{}^{L,\leq nR}\alpha}\ \leq\text{-e} \qquad \frac{{}^{L,\leq nR}\alpha}{{}^{L}\leq nR.\alpha}\ \leq\text{-i}$$

$$\frac{{}^{L}\geq nR.\alpha}{{}^{L,\geq nR}\alpha}\ \geq\text{-e} \qquad \frac{{}^{L,\geq nR}\alpha}{{}^{L}\geq nR.\alpha}\ \geq\text{-i}$$

$$\frac{{}^{\exists R,L}\alpha}{{}^{\geq 1R,L}\alpha}\ \geq\exists \qquad \frac{{}^{\geq nR,L}\alpha}{{}^{\exists R,L}\alpha}\ \exists\geq (n \geq 1)$$

Kalkül des natürlichen Schließen für ALCQI

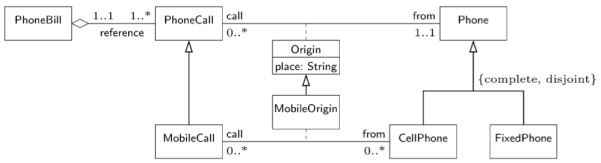
$$\frac{\geq^{mR,L}\alpha}{\geq^{nR,L}\alpha} - \geq (m \geq n) \qquad \frac{\leq^{mR,L}\alpha}{\leq^{nR,L}\alpha} + \geq (m \leq n) \qquad \frac{^{L}\alpha}{\forall R,^{L}\alpha}\ Gen$$

$$\frac{^{L_1}\alpha \quad ^{L_1}\alpha \sqsubseteq {}^{L_2}\beta}{^{L_2}\beta} \sqsubseteq\text{-e} \qquad \frac{\begin{array}{c}[^{L_1}\alpha]\\ \vdots \\ ^{L_2}\beta\end{array}}{^{L_1}\alpha \sqsubseteq {}^{L_2}\beta} \sqsubseteq\text{-i} \qquad \frac{\exists R,^{L_1}\alpha \sqsubseteq {}^{L_2}\beta}{^{L_1}\alpha \sqsubseteq \forall R^-,^{L_2}\beta}\ inv$$

Korrektheit und Vollständigkeit

- $ND_{\mathcal{ALCQI}}$ ist korrekt
- $ND_{\mathcal{ALCQI}}$ ist vollständig
- Gegeben Annahmen T und zu beweisende ALCQI Formel α und ein voll-expandierter ND-Abeltiungsbaum P:
  - Falls P kein Beweis ist, dann kann daraus ein Gegenbeispiel für $T \vdash \alpha$ extrahiert werden.
  - Entscheidbarkeit

Anwendung: UML



Anwendung: UML Diagramm als TBOX

$$\begin{aligned}
\texttt{Origin} &\sqsubseteq \forall \texttt{place.String}\\
\texttt{Origin} &\sqsubseteq \exists \texttt{place.}\top \sqcap (\leq 1\ \texttt{place})\\
\texttt{Origin} &\sqsubseteq \exists \texttt{call.PhoneCall} \sqcap (\leq 1\ \texttt{call}) \sqcap \exists \texttt{from.Phone} \sqcap (\leq 1\ \texttt{from})\\
\texttt{MobileOrigin} &\sqsubseteq \exists \texttt{call.MobileCall} \sqcap (\leq 1 \texttt{call}) \sqcap \exists \texttt{from.CellPhone} \sqcap (\leq 1\ \texttt{from})\\
\texttt{PhoneCall} &\sqsubseteq (\geq 1\ \texttt{call}^-\texttt{.Origin}) \sqcap (\leq 1\ \texttt{call}^-\texttt{.Origin})\\
\top &\sqsubseteq \forall \texttt{reference}^-\texttt{.PhoneBill} \sqcap \forall \texttt{reference.PhoneCall}\\
\texttt{PhoneBill} &\sqsubseteq (\geq 1\ \texttt{reference}^-)\\
\texttt{PhoneCall} &\sqsubseteq (\geq 1\ \texttt{reference}) \sqcap (\leq 1\ \texttt{reference})\\
\texttt{MobileCall} &\sqsubseteq \texttt{PhoneCall}\\
\texttt{MobileOrigin} &\sqsubseteq \texttt{Origin}\\
\texttt{CellPhone} &\sqsubseteq \texttt{Phone}\\
\texttt{FixedPhone} &\sqsubseteq \texttt{Phone}\\
\texttt{CellPhone} &\sqsubseteq \neg\texttt{FixedPhone}\\
\texttt{Phone} &\sqsubseteq \texttt{CellPhone} \sqcup \texttt{FixedPhone}
\end{aligned}$$

Anwendung: Beweis von Eigenschaften des UML Diagramms

$$\dfrac{\dfrac{[\geq 2\ \texttt{c}^-\texttt{.MO}]^2 \qquad \dfrac{\texttt{MO} \sqsubseteq 0}{\geq 2\ \texttt{c}^-\texttt{.MO} \sqsubseteq \geq 2\ \texttt{c}^-\texttt{.0}} \qquad \dfrac{[\texttt{MC}]^1 \quad \texttt{MC} \sqsubseteq \texttt{PC}}{\texttt{PC}} \quad \texttt{PC} \sqsubseteq \geq 1\ \texttt{c}^-\texttt{.0} \sqcap \leq 1\ \texttt{c}^-\texttt{.0}}{\geq 2\ \texttt{c}^-\texttt{.0}} \qquad \geq 1\ \texttt{c}^-\texttt{.0} \sqcap \leq 1\ \texttt{c}^-\texttt{.0}}{\leq 1\ \texttt{c}^-\texttt{.0}}$$

$$\dfrac{\dfrac{\bot}{\neg \geq 2\ \texttt{c}^-\texttt{.MO}}\ 2}{\texttt{MC} \sqsubseteq \neg \geq 2\ \texttt{c}^-\texttt{.MO}}\ 1$$

- ND-Beweis, dass jeder *MobileCall* maximal einen *MobileOrigin* hat.

Anwendung: Konsistenz des UML Diagramms

$$\dfrac{\dfrac{\texttt{Cell} \sqsubseteq \neg\texttt{Fixed} \qquad [\texttt{Cell}]^1}{\neg\texttt{Fixed}} \qquad \dfrac{\texttt{Cell} \sqsubseteq \texttt{Fixed} \qquad [\texttt{Cell}]^1}{\texttt{Fixed}}}{\dfrac{\bot}{\texttt{Cell} \sqsubseteq \bot}\ 1}$$

- Neues Axiom *CellPhone* $\sqsubseteq$ *FixedPhone*
- Inkonsistenz

Eigenschaften von Beschreibungslogiken

Complexity of reasoning in Description Logics

Note: the information here is (always) incomplete and updated often

Base description logic: $\mathcal{A}$ttributive $\mathcal{L}$anguage with $\mathcal{C}$omplements

$\mathcal{ALC}$ ::= $\bot$ | $\top$ | A | $\neg C$ | $C \sqcap D$ | $C \sqcup D$ | $\exists R.C$ | $\forall R.C$

Concept constructors:

- $\mathcal{F}$ – functionality[4]: $(\leq 1\ R)$
- $\mathcal{N}$ – (unqualified) number restrictions: $(\geq n\ R)$, $(\leq n\ R)$
- $\mathcal{Q}$ – qualified number restrictions: $(\geq n\ R.C)$, $(\leq n\ R.C)$
- $\mathcal{O}$ – nominals: $\{a\}$ or $\{a_1, \ldots, a_n\}$ ("one-of")
- μ – least fixpoint operator: $\mu X.C$
- Forbid complex roles[5] in number restrictions[6]

Role constructors: trans reg

- $\mathcal{I}$ – role inverse: R^-
- $\cap$ – role intersection[3]: $R \cap S$
- $\cup$ – role union: $R \cup S$
- $\neg$ – role complement: $\neg R$ full
- $\circ$ – role chain (composition): $R \circ S$
- * – reflexive-transitive closure[4]: R^*
- id – concept identity: $id(C)$

TBox (concept axioms):

- empty TBox
- acyclic TBox ($A \equiv C$, A is a concept name; no cycles)
- general TBox ($C \sqsubseteq D$, for arbitrary concepts C and D)

RBox (role axioms): OWL-Lite OWL-DL OWL 1.1

- $\mathcal{S}$ – role transitivity: $\mathsf{Tr}(R)$
- $\mathcal{H}$ – role hierarchy: $R \sqsubseteq S$
- $\mathcal{R}$ – complex role inclusions: $R \circ S \sqsubseteq R$, $R \circ S \sqsubseteq S$
- $\mathcal{I}$ – some additional features (click to see them)

Reset — You have selected a Description Logic: $\mathcal{ALC}$

Complexity[7] of reasoning problems[8]

Concept satisfiability	PSpace-complete	• Hardness for $\mathcal{ALC}$: see [80]. • Upper bound for $\mathcal{ALCQ}$: see [12, Theorem 4.6].
ABox consistency	PSpace-complete	• Hardness follows from that for concept satisfiability. • Upper bound for $\mathcal{ALCQO}$: see [17, Appendix A].

Important properties of the Description Logic

Finite model property	Yes	$\mathcal{ALC}$ is a notational variant of the multi-modal logic $\mathbf{K_m}$ (cf. [77]), for which the finite model property can be found in [4, Sect. 2.3].

http://www.cs.man.ac.uk/~ezolin/dl/

http://dl.kr.org

Zusammenfassung und Nächste Woche

- Fragmente von Prädikatenlogik, die noch entscheidbar sind
- Beispiel: Familie der Beschreibungslogiken
- Grundlegene Beschreibungslogik ALC
- Fortgeschrittene Beschreibungslogik ALCQI
  - Modellierung von UML-Diagrammen
- Prädikatenlogik mit Induktion

Formale Modellierung
Vorlesung 7 vom 02.06.14: Prädikatenlogik mit induktiven Datentypen

Serge Autexier & Christoph Lüth

Universität Bremen

Sommersemester 2014

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
  - Beschreibungslogiken
  - FOL mit induktiven Datentypen
  - FOL mit Induktion und Rekursion
  - Die Unvollständigkeitssätze von Gödel
- Teil II: Spezifikation und Verifikation

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 $\mathbb{N}$

$$\forall x.\, s(x) \neq 0 \quad (N1)$$
$$\forall x.\forall y.\, s(x) = s(y) \longrightarrow x = y \quad (N2)$$
$$\forall x.\, 0 + x = x \quad (A1)$$
$$\forall x.\forall y.\, s(x) + y = s(x + y) \quad (A2)$$

- Beweise in ND

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

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

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:

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

Beweise in ND

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

In Isabelle.

Wie sehen unsere Zahlen eigtl. aus?

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

Modelle

- Füge hinzu:

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

- Füge weiter hinzu:

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

- "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:

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

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

$$P(t) := P(\$)\left[\begin{smallmatrix} t \\ \$ \end{smallmatrix}\right]$$

- Abgeleitete ND Regeln:

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

Hilft das Induktionsschema zum Beweisen?

- Es gelten:

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

- Beweise in ND

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

- Definiere

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

Und was ist mit den Modellen?

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

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

- Jede endliche Teilmenge von PA^∞ hat Modell

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

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

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

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

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

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

Allgemein

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

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

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

- Und allgemein für frei erzeugte Datentypen.

Zusammenfassung

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

Formale Modellierung
Vorlesung 8 vom 07.06.14: FOL mit Induktion und Rekursion

Serge Autexier & Christoph Lüth

Universität Bremen

Sommersemester 2014

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
  - Beschreibungslogiken
  - FOL mit induktiven Datentypen
  - FOL mit Induktion und Rekursion
  - Die Unvollständigkeitssätze von Gödel
- Teil II: Spezifikation und Verifikation

Das Tagesmenü

- Beweis von Eigenschaften von Funktionen mit FOL-ND
  - Wohlfundierte Induktion und rekursive Funktionen
- Axiomatische Definition von Theorien ist gefährlich
- Prädikatenlogik mit mehreren Typen
- Konservative Erweiterungen als sicheres Theorie Definitionsprinzip
  - Typdefinitionen
  - Wohlfundierte rekursive Funktionen/Prädikate
- Terminierende Funktionen und abgeleitete Induktionsschemata

Mehr Beweise

- Definiere $\leq$ und half:

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

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

- Beweise

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

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

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

- Beweise $<$ ist wohlfundiert
-

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

$$\frac{\begin{array}{ccc} \begin{bmatrix} \forall x.x < c \\ \text{half}(x) \leq x \\ c = 0 \end{bmatrix} & \begin{bmatrix} \forall x.x < c \\ \text{half}(x) \leq x \\ c = s(0) \end{bmatrix} & \begin{bmatrix} \forall x.x < c \\ \text{half}(x) \leq x \\ \exists u.c = s(s(u)) \end{bmatrix} \\ c = 0 \vee \\ c = s(0) \vee \\ \exists u.c = s(s(u)) & \text{half}(c) \leq c & \text{half}(c) \leq c \end{array}}{\forall x.\, \text{half}(x) \leq x}$$

Mehr Information

- Besser zum beweisen wäre wenn man gleich hätte

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

- Vergleiche:

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

- Generiere Induktionschema aus rekursiven Funktionsdefinitionen

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

Weitere Beispiele

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

- Sortieren

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

- Induktionsschema

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

Weitere Beispiele

- Fibonacci:

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

$$\frac{P(0) \quad P(\mathrm{s}(0)) \quad \begin{bmatrix} P(\mathrm{s}(c)), P(c) \\ \vdots \\ P(\mathrm{s}(\mathrm{s}(c))) \end{bmatrix}}{\forall x.P(x)}$$

Weitere Beispiele

- GGT:

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

$$\frac{\forall y.P(0, y) \quad \forall x.P(\mathrm{s}(x), 0) \quad \begin{bmatrix} x \leq y \\ P(x, y - x) \\ \vdots \end{bmatrix} \quad \begin{bmatrix} \neg(x \leq y) \\ P(x - y, x) \\ \vdots \end{bmatrix}}{\forall x, y.P(x, y)} \qquad \begin{matrix} P(x, y) \\ \\ P(x, y) \end{matrix}$$

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

- 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)

Definition von Theorien

- Was alles schiefgehen kann und wie man das vermeidet
- Axiomatische Definition von Theorien ist gefährlich
- Bisher können wir Listen nicht unterscheiden von nat. Zahlen, von binären Bäumen, etc.
- Wir brauchen so etwas wie Typen für die jeweiligen Objekte, die disjunkt voneinander sind

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 $t_i \in \mathcal{T}erm_\Sigma^{\tau_i}, 1 \le i \le n$, dann $f(t_1, \ldots, t_n) \in \mathcal{T}erm_\Sigma^{\tau_0}$
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 $t_i \in \mathcal{T}erm_\Sigma^{\tau_i}, 1 \le i \le n$, 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	$\forall x.\phi \in \mathcal{F}orm_\Sigma, x \in X$	$\forall x_\tau.\phi \in \mathcal{F}orm_\Sigma$
	$\exists x.\phi \in \mathcal{F}orm_\Sigma, x \in X$	$\exists x_\tau.\phi \in \mathcal{F}orm_\Sigma$

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

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

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:

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

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)
- (†) $t \in \mathcal{T}erm_\Sigma^\tau$; Ggf. Umbenennung durch Substitution

Der Existenzquantor

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

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

- (*) 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 $\mathcal{F}orm_\Sigma$ 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 $\mathrm{Abs}_t : r \to t, \mathrm{Rep}_t : t \to r$ axiomatisch Beschrieben

Typdefinitionen sind konservative 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.

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

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 $c_i : \tau^i_1 \times \tau^i_{n_i} \to t \in \mathcal{F}, 1 \le i \le k$. Dann ist t frei erzeugt in S durch Konstruktoren $c_1, \ldots, c_k$ gdw.

- $S \vdash \forall x_t. \bigvee_{i=1\ldots k} \exists y^1_{\tau^i_1}, \ldots y^{n_k}_{\tau^i_{n_i}}.x = c_i(y^1, \ldots, y^{n_i})$
- $S \vdash \forall y^1_{\tau^i_1}, \ldots y^{n_i}_{\tau^i_{n_i}}.\forall z^1_{\tau^i_1}, \ldots z^{n_i}_{\tau^i_{n_i}}.c_i(y^1, \ldots, y^{n_i}) = c_i(z^1, \ldots, z^{n_i}) \longrightarrow ((y^1 = z^1 \wedge \cdots \wedge y^{n_i} = y^{n_i}))$ für alle c_i
- $S \vdash \forall y^1_{\tau^i_1}, \ldots y^{n_i}_{\tau^i_{n_i}}.\forall z^1_{\tau^j_1}, \ldots z^{n_j}_{\tau^j_{n_j}}.c_i(y^1, \ldots, y^{n_i}) = c_j(z^1, \ldots, z^{n_j})$ 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

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

so daß beweisbar

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

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:
  - 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

Erweiterung um totale, terminierende Funktionen ist konservativ

Definition 7 (Funktions- und Prädikatsdefinitionen)

Sei $S = ((\mathcal{T}, \mathcal{F}, \mathcal{P}), \Phi)$ eine Spezifikation, $f : \tau_1 \times \dots \times \tau_n \to \tau_0 \notin \mathcal{F}$ ($\tau_i \in \mathcal{T}$) 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
- $\Phi' = \Phi \cup \Psi$
- $\mathcal{F}' = \mathcal{F} \cup \{f : \tau_1 \times \dots \times \tau_n \to \tau_0\}$

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 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).

Abgeleitete Induktionsschemata

- fib(x)

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

- half(x)

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

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

Formale Modellierung
Vorlesung 9 vom 12.06.14: Die Unvollständigkeitssätze von Gödel

Serge Autexier & Christoph Lüth

Universität Bremen

Sommersemester 2014

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
  - Beschreibungslogiken
  - FOL mit induktiven Datentypen
  - FOL mit Induktion und Rekursion
  - Die Unvollständigkeitssätze von Gödel
- Teil II: Spezifikation und Verifikation

Das Tagesmenü

- 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.

Gödels erster Unvollständigkeitssatz

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

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

Gödel Kodierung

Folgende Funktion ist definierbar in PA:

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

Eigenschaften: Es gibt eindeutige Projektionen

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

Gödel Kodierung für Terme

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

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

$$\ulcorner x_i\urcorner := (0, i)$$

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

$$\ulcorner f_i\urcorner := (1, i)$$

- Terme

$$\ulcorner f_i(t_1, \ldots t_n)\urcorner := \langle \ulcorner f_i\urcorner, \ulcorner t_1\urcorner, \ldots \ulcorner t_n\urcorner \rangle$$

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

$$\ulcorner p_i\urcorner := (2, i)$$

- Atome

$$\ulcorner p_i(t_1, \ldots t_n)\urcorner := \langle \ulcorner p_i\urcorner, \ulcorner t_1\urcorner, \ldots \ulcorner t_n\urcorner \rangle$$

- Konnektive und Quantoren

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

Gödel Kodierung für Formeln II

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

- $\ulcorner\neg\varphi\urcorner = (\ulcorner\neg\urcorner, \ulcorner\varphi\urcorner)$
- $\ulcorner\psi \circ \varphi\urcorner = \langle \ulcorner\circ\urcorner, \ulcorner\psi\urcorner, \ulcorner\varphi\urcorner \rangle$
- $\ulcorner Qx_i.\varphi\urcorner = \langle \ulcorner Q\urcorner, \ulcorner x_i\urcorner, \ulcorner\varphi\urcorner \rangle$

Lemma 1 (Facts)

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

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

Gödel Kodierung für Ableitungen

- Gödel Kodierung für Hypothesen Liste:

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

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

- Definition von Konkatenation $*$ und Streichen von Hypothesen

Gödel Kodierung für Ableitungen

$$\left\ulcorner \dfrac{\begin{matrix}D_1 & D_2\\ \phi & \psi\end{matrix}}{\phi \wedge \psi}\ \wedge I \right\urcorner = \langle(5,\ulcorner\wedge\urcorner),\left\ulcorner\begin{matrix}D_1\\ \phi\end{matrix}\right\urcorner,\left\ulcorner\begin{matrix}D_2\\ \psi\end{matrix}\right\urcorner,\ulcorner\phi\wedge\psi\urcorner\rangle$$

$$\left\ulcorner \dfrac{\begin{matrix}D\\ \phi \wedge \psi\end{matrix}}{\phi}\ \wedge E_L \right\urcorner = \langle(6,\ulcorner\wedge\urcorner),\left\ulcorner\begin{matrix}D\\ \phi\wedge\psi\end{matrix}\right\urcorner,\ulcorner\phi\urcorner\rangle$$

Gödel Kodierung für Ableitungen

$$\left\ulcorner \dfrac{\begin{matrix}D\\ \psi\end{matrix}}{\phi \longrightarrow \psi}\ \longrightarrow I \right\urcorner = \langle(5,\ulcorner\longrightarrow\urcorner),\left\ulcorner\begin{matrix}D\\ \psi\end{matrix}\right\urcorner,\ulcorner\phi\longrightarrow\psi\urcorner\rangle$$

$$\left\ulcorner \dfrac{\begin{matrix}D_1 & D_2\\ \phi & \phi\longrightarrow\psi\end{matrix}}{\psi}\ \longrightarrow E \right\urcorner = \langle(6,\ulcorner\longrightarrow\urcorner),\left\ulcorner\begin{matrix}D_1\\ \phi\end{matrix}\right\urcorner,\left\ulcorner\begin{matrix}D_2\\ \phi\longrightarrow\psi\end{matrix}\right\urcorner,\ulcorner\psi\urcorner\rangle$$

Gödel Kodierung für Ableitungen

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

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

Beweisbarkeit

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

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

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

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

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

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

Fixpoint Theorem

Theorem 2 (Fixpoint Theorem)

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

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.

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

- Es gilt $\mathbb{N} \models \varphi \longleftrightarrow \neg\,\mathrm{Thm}(\ulcorner\varphi\urcorner)$
- Annahme $\mathbb{N} \models \mathrm{Thm}(\ulcorner\varphi\urcorner)$

$\Leftrightarrow \mathbb{N} \models \exists x.\,\mathrm{Prov}(x,\ulcorner\varphi\urcorner)$	$\Leftrightarrow \mathbb{N} \models \mathrm{Prov}(n,\ulcorner\varphi\urcorner)$ for some n
$\Leftrightarrow \vdash \mathrm{Prov}(n,\ulcorner\varphi\urcorner)$ for some n	$\Leftrightarrow \vdash \varphi$
$\Rightarrow \vdash \neg\,\mathrm{Thm}(\ulcorner\varphi\urcorner)$	$\Rightarrow \mathbb{N} \models \neg\,\mathrm{Thm}(\ulcorner\varphi\urcorner)$

- 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 10 vom 24.06.14: Formale Modellierung mit UML und OCL

Serge Autexier & Christoph Lüth

Universität Bremen

Sommersemester 2014

Fahrplan

- Teil I: Formale Logik
- Teil II: Spezifikation und Verifikation
  - Formale Modellierung mit der UML und OCL
  - Lineare Temporale Logik
  - Temporale Logik und Modellprüfung
  - Hybride Systeme
  - Zusammenfassung, Rückblick, Ausblick

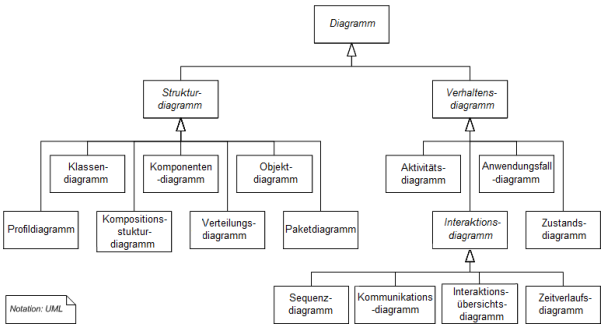
Unified Modeling Language (UML)

- Allgemeine Modellierungssprache
- Spezifikation problemorientiert
- Übersetzung in verschiedene Programmiersprachen möglich
- Nur bestimme Aspekte sind formal

UML als formale Spezifikationssprache

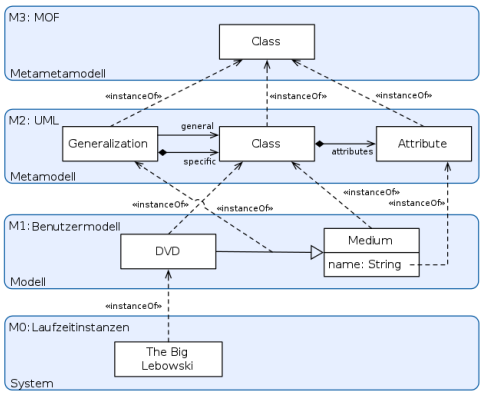
Diagrammtyp	Modellierte Aspekte	Formal
Klassendiagramm	Statische Systemstruktur	Ja
Paketdiagramm	Pakete, Namensräume	Nein
Objektdiagramm	Zustand von Objekten	(Ja)
Kompositionsstrukturdiagramm	Kollaborationen	Nein
Komponentendiagramm	Dynamische Systemstruktur	(Nein)
Verteilungsdiagramm	Implementierungsaspekte	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
- Entwickelt in den 90ern
- Formale Constraints an UML-Diagrammen
  - Datentypen gegeben durch UML

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 oder Widersprüche
  - Aussagen, die unter allen Belegungen zu 1 bzw. 0 auswerten
- Es gilt bspw. $[\![\neg A \vee B]\!]_v = [\![A \longrightarrow B]\!]_v$,
  aber nicht $(\neg A \vee B) \longleftrightarrow (A \longrightarrow B)$

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

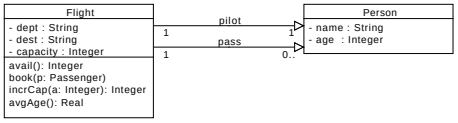
- Aus Attribute, Operationen, Assoziationen des Modells
- Navigation entlang der Assoziationen
- Für Kardinalität 1 Typ `T`, sonst `Set(T)`
- Benutzerdefinierte Operationen in Ausdrücken müssen zustandsfrei sein (Stereotyp `<<query>>`)

Beispiel

Ein Flugbuchungssystem

Jeder Flug hat ein Start, ein Ziel, eine Kapazität (Anzahl verfügbarer Sitze), einen Piloten sowie eine Menge von zugeordneten Passagieren; Piloten und Passagiere sind Personen.
Jede Person hat einen Namen und ein Alter.

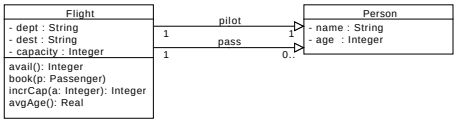
OCL im Beispiel



Start und Ziel sind immer unterschiedlich.

```
context Flight
inv:    self.dept <> self.dest
```

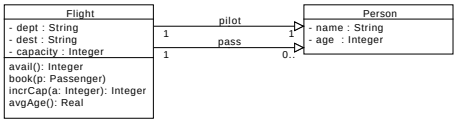

OCL im Beispiel



Der Pilot muss über 25 sein.

```
context Flight
inv:    self.pilot.age >= 25
```

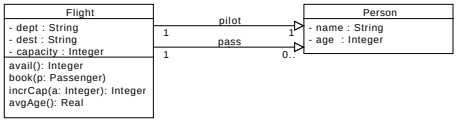

OCL im Beispiel



Es gibt nie mehr Passagiere als Kapazität.

```
context Flight
inv:    self.pass->size() <= self.capacity
```

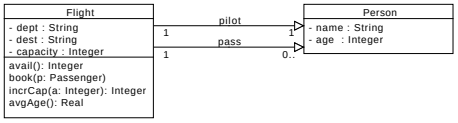

OCL im Beispiel



Jeder Flug hat mindestens einen Passagier über 18.

```
context Flight
inv:    self.pass->exists(p|p.age >= 18)
```

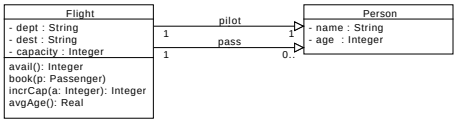

OCL im Beispiel



Der Pilot ist kein Passagier.

```
context Flight
inv:    not (self.pass->contains(self.pilot))
```

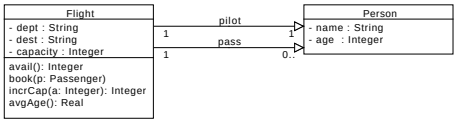

OCL im Beispiel



Jeder Passagier unter 18 wird von einem über 18 mit dem gleichen Namen (Elternteil, Geschwister) begleitet.

```
context Flight
inv: self.pass->all(p|p.age < 18 implies
                    self.pass->exists(q|q.name= p.name
                                      and q.age >= 18))
```

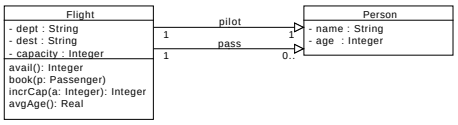

OCL im Beispiel



Die Operation avail gibt die Anzahl der noch freien Plätze zurück.

```
context Flight::avail() : Integer
post:   result = self.capacity - self.pass->size()
```

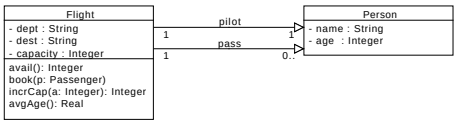
OCL im Beispiel



Wenn noch ein Platz frei ist, soll book den Passagier zu diesem Flug hinzufügen.

```
context Flight::book(p: Person)
pre:    self.capacity - self.pass->size() > 0
post:   self.pass->contains(p)
```

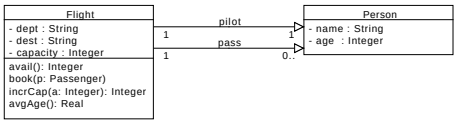

OCL im Beispiel



Die Operation incrCap erhöht die Kapazität des Fluges.

```
context Flight::incrCap(a: Integer) : Integer
pre:    self.capacity+ a- self.pass->size() >= 0
post:   self.capacity = @pre(self.capacity)+ a
        result = self.capacity
```


OCL im Beispiel



Die Operation avgAge soll das Durchschnittsalter der Fluggäste berechnen.

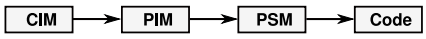
```
context Flight::avgAge() : Real
pre:    self.pass->exists(p| True)
post:   result =
          self.pass->iterate(p:Passenger; sum: Real= 0
                             | sum+ p.age)
                            / self.pass->size()
```


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

- OCL erlaubt Einschränkungen auf Modellen
- Programmbegriff: abstrakter Zustandsübergang
  - Relation zwischen Vor- und Nachtzustand
- Erlaubt mathematisch präzisere Modellierung
- Frage:
  - Werkzeugunterstützung?
  - Ziel: Beweise, Codegenerierung, . . . ?
- Kritik UML:
  - "OO built-in"
  - Adäquat für eingebettete Systeme, CPS, . . . ?

Formale Modellierung
Vorlesung 11 vom 30.06.2014: Lineare Temporale Logik

Serge Autexier & Christoph Lüth

Universität Bremen

Sommersemester 2014

Fahrplan

- Teil I: Formale Logik
- Teil II: Spezifikation und Verifikation
  - Formale Modellierung mit der UML und OCL
  - Lineare Temporale Logik
  - Temporale Logik und Modellprüfung
  - Hybride Systeme
  - Zusammenfassung, Rückblick, Ausblick

Tagesmenu: Lineare Temporale Logik

Logik	Programmbegriff	Beweisprinzip
HOL	Rekursive Funktion	Induktion
OCL[1]	Zustandsübergang	Vor/Nachbedingung
TL	Zustandsmaschine	Modelchecking

- Endliche Zustandsmaschinen
- Pfadausdrücke
- Ausdrücke über Pfaden: LTL

[1] Und andere

Endliche Zustandsmaschine

Definition (Finite State Machine (FSM))

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

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

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

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

Ein Einfaches Beispiel

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

Linear Temporal Logic (LTL) and Pfade

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

Definition (Pfade)

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

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

Lineare Temporale Logik (LTL)

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

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

Erfüllung und Modelle für LTL

Die Erfüllbarkeitsrelation für einen Pfad p und eine LTL-Formel ϕ ist induktiv wie folgt definiert:

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

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

Definition (Modell einer LTL-Formel)

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

Äquivalenzen

Definition (Äquivalenz)

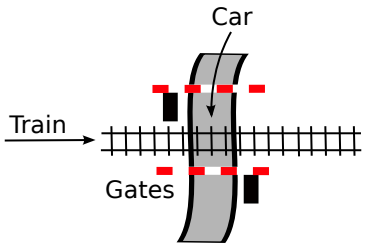
Zwei Formeln sind äquivalent, $\phi \equiv \psi$ gdw. für alle FSM $\mathcal{M}$ und Pfade p in $\mathcal{M}$, $p \models \phi \longleftrightarrow p \models \psi$

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

$$\begin{array}{lcl} \mathsf{F}(\phi \vee \psi) & \equiv & \mathsf{F}\,\phi \vee \mathsf{F}\,\psi \\ \mathsf{G}(\phi \wedge \psi) & \equiv & \mathsf{G}\,\phi \wedge \mathsf{G}\,\psi \end{array} \qquad \begin{array}{lcl} \neg\mathsf{F}\,\phi & \equiv & \mathsf{G}(\neg\phi) \\ \neg\mathsf{G}\,\phi & \equiv & \mathsf{F}(\neg\phi) \\ \neg\mathsf{X}\,\phi & \equiv & \mathsf{X}(\neg\phi) \end{array} \qquad \begin{array}{lcl} \mathsf{FGF}\phi & \equiv & \mathsf{GF}\phi \\ \mathsf{GFG}\phi & \equiv & \mathsf{FG}\phi \end{array}$$

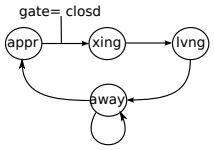
$$\begin{array}{lcl} \mathsf{X}\,\mathsf{F}\,\phi & \equiv & \mathsf{F}\,\mathsf{X}\,\phi \\ \mathsf{X}\,\mathsf{G}\,\phi & \equiv & \mathsf{G}\,\mathsf{X}\,\phi \\ \mathsf{X}(\phi\ U\ \psi) & \equiv & \mathsf{X}\,\phi\ U\ \mathsf{X}\,\psi \end{array} \qquad \begin{array}{lcl} \mathsf{F}\,\phi & \equiv & \phi \vee \mathsf{X}\,\mathsf{F}\,\phi \\ \mathsf{G}\,\phi & \equiv & \phi \wedge \mathsf{X}\,\mathsf{G}\,\phi \\ \phi\ U\ \psi & \equiv & \psi \vee (\phi \wedge \mathsf{X}(\phi\ U\ \psi)) \end{array}$$

Längeres Beispiel: der Bahnübergang

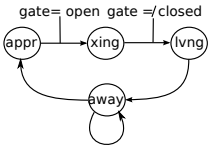


Modellierung des Bahnübergangs

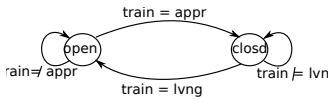
Zustände des Zuges:



Zustände des Autos:



Zustände der Schranke:



Die FSM

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

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

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

- Zustandsübergang komponentenweise, bspw:

$$\langle away, open, away\rangle \rightarrow \langle appr, open, away\rangle$$
$$\langle appr, open, away\rangle \rightarrow \langle xing, open, away\rangle$$
$$\ldots$$

Bahnübergang — Formalisierung von Eigenschaften

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

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

- Ein Auto kann den Übergang immer wieder verlassen:

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

- Ein annähernder Zug darf irgendwann den Bahnübergang passieren:

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

- Es gibt Autos, die den Bahnübergang passieren:

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

  - Nicht in LTL auszudrücken!

Zusammenfassung

- LTL: Logik über Pfade in Zustandsautomaten
- Aussagenlogik plus modale Operatoren (X, G, F, U)
- Man kann eine Axiomatisierung und Schlussregeln angeben
  - Dann ist LTL konsistent und vollständig.
- In der Praxis wird LTL über Modellprüfung (model checking) bewiesen.
  - Modellierung des Systems als FSM $\mathcal{M}$, Eigenschaften als LTL-Formel ϕ, Überprüfung ob $\mathcal{M} \models \phi$.
- LTL ist für Sicherheitseigenschaften, keine Verfügbarkeit.
- Dazu mehr in der nächsten Vorlesugn

Formale Modellierung
Vorlesung 12 vom 07.07.2014: Temporale Logik und Modellprüfung

Serge Autexier & Christoph Lüth

Universität Bremen

Sommersemester 2014

Organisatorisches

- Übung am Donnerstag kann verspätet anfangen (ca. 14:30).

Fahrplan

- Teil I: Formale Logik
- Teil II: Spezifikation und Verifikation
  - Formale Modellierung mit der UML und OCL
  - Lineare Temporale Logik
  - Temporale Logik und Modellprüfung
  - Hybride Systeme
  - Zusammenfassung, Rückblick, Ausblick

Computational Tree Logic (CTL)

- Grenzen der LTL: Quantifikation über Pfaden
  - z.B. Existenz eines Pfades mit einer bestimmten Eigenschaft
- Computational Tree Logic (CTL): Erweiterung der LTL um existentielle/universelle Quantoren über modalen Pfadoperatoren.
  - Modale Operatoren: die Zustandsübergänge betreffend
- Name: Pfade im Berechnungsbaum durch Auffalten der FSM.
  - Beispiel Berechnungsbäume für die Getränkemaschine

CTL

Die Formeln der CTL sind gegeben durch:

$$
\begin{array}{lll}
\phi ::= & \top \mid \bot \mid p & \text{— True, false, atomic} \\
\mid & \neg\phi \mid \phi_1 \wedge \phi_2 \mid \phi_1 \vee \phi_2 \mid \phi_1 \longrightarrow \phi_2 & \text{— Propositional formulae} \\
\mid & \mathsf{AX}\,\phi \mid \mathsf{EX}\,\phi & \text{— All or some next state} \\
\mid & \mathsf{AF}\,\phi \mid \mathsf{EF}\,\phi & \text{— All or some future states} \\
\mid & \mathsf{AG}\,\phi \mid \mathsf{EG}\,\phi & \text{— All or some global future} \\
\mid & \mathsf{A}[\phi_1\ U\ \phi_2] \mid \mathsf{E}[\phi_1\ U\ \phi_2] & \text{— Until all or some}
\end{array}
$$

Erfüllbarkeit

- CTL-Formeln: wie LTL, aber mit Quantoren (A or E) über den Temporaloperatoren.
- Ganz grob: A heißt Temporaloperator gilt für alle Pfade von hier; E bedeutet, Temporaloperator gilt für mindestens ein Pfad von hier.
  - Nicht ganz: Temporaloperatoren sind wieder CTL-Formeln, deshalb Rekursion
- In conclusio: Erfüllbarkeitsrelation nicht für einzelne Pfade p oder Bäume t, sondern immer in Bezug auf bestimmten Zustand der FSM.

Erfüllbarkeit für CTL

Für eine FSM $\mathcal{M} = \langle \Sigma, \rightarrow \rangle$, $s \in \Sigma$ und eine CTL-Formel ϕ, die Erfüllbarkeitsrelation $\mathcal{M}, s \models \phi$ ist induktiv wie folgt definiert:

$$
\begin{array}{lll}
\mathcal{M}, s \models \top & & \\
\mathcal{M}, s \not\models \bot & & \\
\mathcal{M}, s \models p & \text{gdw} & p(s) \\
\mathcal{M}, s \models \phi \wedge \psi & \text{gdw} & \mathcal{M}, s \models \phi \text{ und } \mathcal{M}, s \models \psi \\
\mathcal{M}, s \models \phi \vee \psi & \text{gdw} & \mathcal{M}, s \models \phi \text{ oder } \mathcal{M}, s \models \psi \\
\mathcal{M}, s \models \phi \longrightarrow \psi & \text{gdw} & \text{wenn } \mathcal{M}, s \models \phi \text{ dann } \mathcal{M}, s \models \psi \\
\ldots & &
\end{array}
$$

Erfüllbarkeit für CTL

Für eine FSM $\mathcal{M} = \langle \Sigma, \rightarrow \rangle$, $s \in \Sigma$ und eine CTL-Formel ϕ, die Erfüllbarkeitsrelation $\mathcal{M}, s \models \phi$ ist induktiv wie folgt definiert:

$$
\begin{array}{lll}
\ldots & & \\
\mathcal{M}, s \models \mathsf{AX}\,\phi & \text{gdw} & \text{für alle } s_1 \text{ mit } s \rightarrow s_1 \text{ gibt es } \mathcal{M}, s_1 \models \phi \\
\mathcal{M}, s \models \mathsf{EX}\,\phi & \text{gdw} & \text{es gibt } s_1 \text{ mit } s \rightarrow s_1 \text{ und } \mathcal{M}, s_1 \models \phi \\
\mathcal{M}, s \models \mathsf{AG}\,\phi & \text{gdw} & \text{für alle Pfade } p \text{ mit } p_1 = s \\
 & & \text{gilt } \mathcal{M}, p_i \models \phi \text{ für alle } i \geq 2 \\
\mathcal{M}, s \models \mathsf{EG}\,\phi & \text{gdw} & \text{es gibt einen Pfad } p \text{ mit } p_1 = s \\
 & & \text{und } \mathcal{M}, p_i \models \phi \text{ für alle } i \geq 2 \\
\mathcal{M}, s \models \mathsf{AF}\,\phi & \text{gdw} & \text{für alle Pfade } p \text{ mit } p_1 = s \\
 & & \text{gilt } \mathcal{M}, p_i \models \phi \text{ für ein } i \\
\mathcal{M}, s \models \mathsf{EF}\,\phi & \text{gdw} & \text{es gibt einen Pfad } p \text{ mit } p_1 = s \\
 & & \text{und } \mathcal{M}, p_i \models \phi \text{ für ein } i \\
\mathcal{M}, s \models \mathsf{A}[\phi\ U\ \psi] & \text{gdw} & \text{für alle Pfade } p \text{ mit } p_1 = s \text{ gibt es } i \\
 & & \text{mit } \mathcal{M}, p_i \models \psi \text{ und für alle } j < i,\ \mathcal{M}, p_j \models \phi \\
\mathcal{M}, s \models \mathsf{E}[\phi\ U\ \psi] & \text{gdw} & \text{es gibt einen Pfad } p \text{ mit } p_1 = s \text{ und es gibt } i \\
 & & \text{mit } \mathcal{M}, p_i \models \psi \text{ und für alle } j < i,\ \mathcal{M}, p_j \models \phi
\end{array}
$$

Spezifikationsmuster

- Etwas schlechtes (p) darf nicht auftreten: AG $\neg p$ (Sicherheit)
- p tritt unendlich oft auf: AG(AF p)
- p tritt irgendwann auf: AF p (Verfügbarkeit)
- In der Zukunft, p wird irgenwann für immer gelten: AF AG p
- Wann immer p gilt, wird q irgendwann auch gelten: AG($p \longrightarrow$ AF q)
- In allen Zuständen ist p immer eine Möglichkeit: AG(EF p)

LTL und CTL

- CTL ist ausdrucksstärker als LTL, aber das gilt auch anders herum!
  - D.h. es gibt Eigenschaften, die in LTL ausgedrückt werden können, aber nicht in CTL.
- Beispiel: in allen Pfade, in denen p auftritt, tritt auch q auf.
- LTL: F $p \longrightarrow$ F q
- CTL: Weder AF $p \longrightarrow$ AF q noch AG($p \longrightarrow$ AF q)
- Die Logik CTL^* kombiniert die Mächtigkeit von LTL and CTL.

Äquivalenzen

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

$$\begin{array}{lcllcl} \neg(\mathsf{AF}\,\phi) & \equiv & \mathsf{EG}(\neg\phi) & \mathsf{AF}(\phi \vee \psi) & \equiv & \mathsf{AF}\,\phi \vee \mathsf{AF}\,\psi \\ \neg(\mathsf{EF}\,\phi) & \equiv & \mathsf{AG}(\neg\phi) & \mathsf{AG}(\phi \wedge \psi) & \equiv & \mathsf{AG}\,\phi \wedge \mathsf{AG}\,\psi \end{array}$$

$$\mathsf{A}[\phi\ U\ \psi] \equiv \neg(\mathsf{E}[\neg\psi\ U\ \neg\phi \wedge \neg\psi] \vee \mathsf{EG}\,\neg\psi)$$

Theorem (Funktionale Vollständigkeit von CTL)

Eine Menge von CTL-Operatoren ist funktional vollständig für CTL gdw. sie mind. jeweils einen der folgenden Mengen enthält: AX *oder* EX*;* EG*,* AF *oder* AU*; und* EU*.*

Modellprüfung (Model-Checking)

- Das Model-Checking Problem:

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

Theorem (Komplexität von Modellprüfung)

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

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

Skizze eines Model-Checking-Algorithmus für CTL

- Die Denotation einer CTL-Formel ϕ in einem Modell $\mathcal{M}$ ist definiert:
  $$[\![\phi]\!]_{\mathcal{M}} \stackrel{def}{=} \{s \mid \mathcal{M}, s \models \phi\}$$
- Wir definieren $[\![\phi]\!]_{\mathcal{M}}$ durch Rekursion über ϕ:
  - Die aussagenlogischen Fälle sind einfach, z.B. $[\![\phi \vee \psi]\!] = [\![\phi]\!] \cup [\![\psi]\!]$
  - Die temporalen Operatoren werden durch die Äquivalenzen zu EX, EG, EU reduziert, z.B. $[\![\mathsf{AF}\,\phi]\!] = [\![\neg\,\mathsf{EG}\,\neg\phi]\!]$
- Für Menge von Zuständen $Y \subseteq S$, definiere:
  $$pre_\exists(Y) = \{s \in S \mid \exists s'.(s \to s', s' \in Y)\}$$
  und damit rekursive Formulierung für EG, EU:

$$\begin{array}{rcl} [\![\mathsf{EX}\,\phi]\!] & = & pre_\exists([\![\phi]\!]) \\ [\![\mathsf{EG}\,\phi]\!] & = & [\![\phi]\!] \cap pre_\exists([\![\mathsf{EG}\,\phi]\!]) \\ [\![\mathsf{E}[\phi\ U\ \psi]]\!] & = & [\![\psi]\!] \cup ([\![\phi]\!] \cap pre_\exists([\![\mathsf{E}[\phi\ U\ \psi]]\!])) \end{array}$$

- Basis für funktionale Implementation oder Korrektheitsbeweis.

Model-Checking Werkzeuge

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

Zusammenfassung

- LTL und CTL sind temporale Logiken, die Aussagen über das Verhalten eines als FSM modellierten Systems erlauben.
  - Unterschiedliche Mächtigkeiten
  - LTL für Sicherheitseigenschaften, CTL für Verfügbarkeit.
- Modellprüfung (Model-Checking):
  - Entscheidbar, aber mit hoher Komplexität (Zustandsexplosion)
  - Zustandsabstraktion und Zustandskompression machen Model-Checking handhabbar.
- Model-Checker wie NuSMV entscheiden das Model-Checking-Problem.
  - Bei negativer Antwort Gegenbeispiel.
  - Vertrauenswürdigkeit: bei positiver Antwort? Wie gut ist das Modell?

Formale Modellierung
Vorlesung 13 vom 14.07.2014: Hybride Systeme

Serge Autexier & Christoph Lüth

Universität Bremen

Sommersemester 2014

Fahrplan

- Teil I: Formale Logik
- Teil II: Spezifikation und Verifikation
  - Formale Modellierung mit der UML und OCL
  - Lineare Temporale Logik
  - Temporale Logik und Modellprüfung
  - Hybride Systeme
  - Zusammenfassung, Rückblick, Ausblick

*Thanks to Andreas Nonnengart for the slides

What are Hybrid Systems?

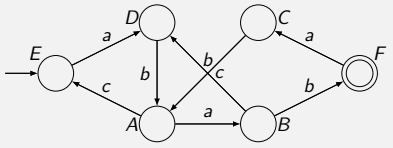
Alur, Henzinger et al

A hybrid system is a digital real-time system that is embedded in an analog environment. It interacts with the physical world through sensors and actuators.

Wikipedia

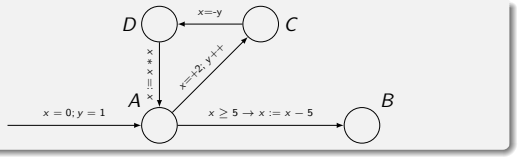
A hybrid system is a system that exhibits both continuous and discrete dynamic behavior – a system that can both flow (described by differential equations) and jump (described by a difference equation).

Finite Automata



- There are vertices (states, locations) and edges (transitions)
- and maybe some input alphabet
- and maybe some "accepting" state

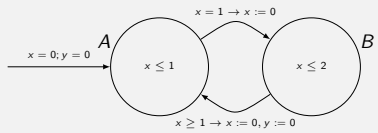
Discrete Automata



- there are variables involved, and they can be manipulated
- transitions may be guarded
- in general not finite state

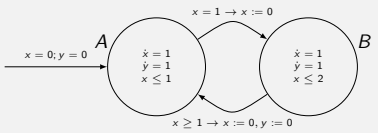
Timed Automata

- additional *clock variables*
- they continuously increase their value in locations
- all of them behave identically
- only operation: reset to 0



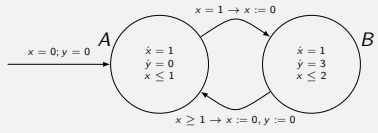
Timed Automata

- additional *clock variables*
- they continuously increase their value in locations
- all of them behave identically
- only operation: reset to 0



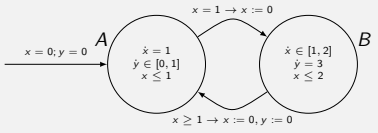
Multi-Phase Automata

- additional variables with a fixed rate, not only clocks
- they increase their value according to the rate
- thus not all of them behave identically
- arbitrary operations

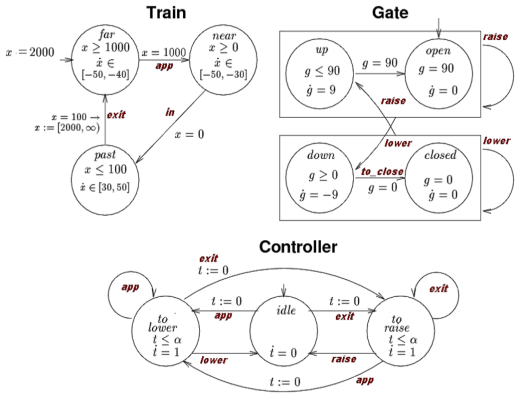


Rectangular Automata

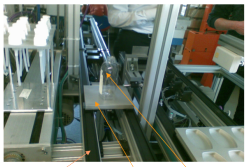
- additional variables with a *bounded* rate
- they increase their value according to these bounds
- they represent arbitrary functions wrt/ bounds
- arbitrary operations



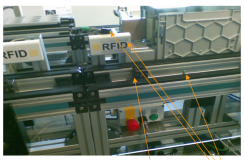
Railroad Gate Controller



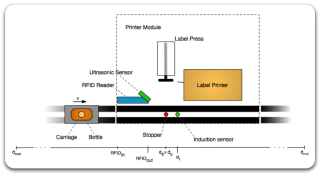
Smart Factory



transportation belt, carriage, bottle

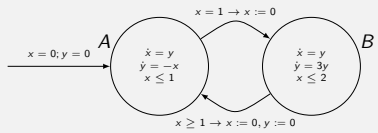


Labeling Section with stoppers and sensors



Affine Automata

- additional variables with arbitrary rate
- the rate may be in terms of the (other) variables
- they represent in general non-linear functions
- arbitrary operations



Temporal Logic - operators $\Box$ and $\Diamond$

Linear Temporal Logic

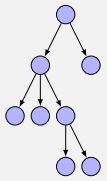
Interpret $\Box$ as *Always, Henceforth, from now on*
Interpret $\Diamond$ as *Eventually, Unavoidable*

Branching Temporal Logic

Interpret $\Box$ as *Always, Henceforth, from now on*
Interpret $\Diamond$ as *Eventually in a possible future*

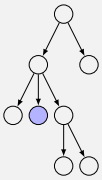
Computation Tree Logic Illustrated

$\forall\Box$ *for each path - always*



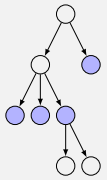
Computation Tree Logic Illustrated

$\exists\Diamond$ *for some path - eventually*



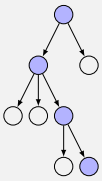
Computation Tree Logic Illustrated

$\forall\Diamond$ *for each path - eventually*



Computation Tree Logic Illustrated

$\exists\Box$ *for some path - always*



Timed (Integrator) CTL

- add clock variables
- these may be used in formulas
- restrict these clocks to certain locations (stopwatches)

$$z.\exists\Diamond\{A \wedge z \leq 5\}$$
$$c^{\{N,M\}}.\forall\Box\{P \rightarrow c \geq 12\}$$

Safety Properties

A safety property is of the form

$$\forall\Box\Phi$$

where Φ is a classical logic formula (with arithmetics)
We call a state s safe if $\Phi(s)$ is true

It has to be shown that all reachable states are safe (forward reachability)

or, equivalently,

It has to be shown that no unsafe state is reachable (backward reachability)

Forward Reachability

The Operator $post(S)$

Given a set S of states

$$post(S) = \{s \mid \exists s' \in S : s' \mapsto_\delta \mapsto_{tr} s\}$$

Fixpoint Iteration

Start with S a the initial states
repeat until $post(S) \subseteq S$: $S := S \cup post(S)$

Finally

Check whether $\Phi(S)$ holds

Backward Reachability

The Operator $pre(S)$

Given a set S of states

$$pre(S) = \{s \mid \exists s' \in S : s \mapsto_{tr} \mapsto_\delta s'\}$$

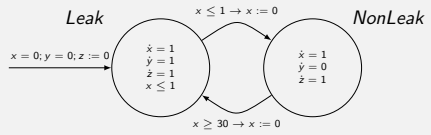
Fixpoint Iteration

Start with $S = \{s \mid \neg\Phi(s)\}$
repeat until $pre(S) \subseteq S$: $S := S \cup pre(S)$

Finally

Check whether the initial state is contained in S

Example: Leaking Gas Burner

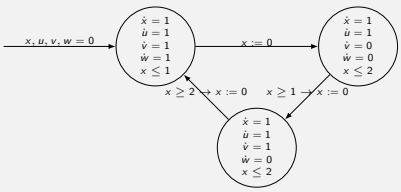


Safety Property

$$\forall\Box\ z \geq 60 \rightarrow 20 * y \leq z$$

$I = \{Leak(0,0,0)\}$
$post(I) = \{Leak(x,y,z) \mid 0 \leq x \leq 1, y = x, z = x\}$
$\cup\ \{NonLeak(0,y,z) \mid 0 \leq y \leq 1, z = y\}$

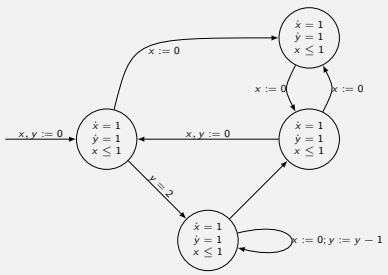
Problem: Long Loops



Property (many iterations)

$$\forall\Box\ (u \geq 154 \rightarrow 5.9 * w \leq u + v)$$

Another Problem: Termination

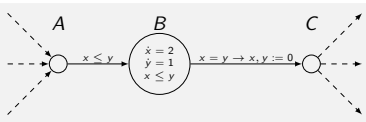


Location Elimination

General Idea

- Compute the responsibility for a location once and for all
- thereby compute a definition for this location
- insert this definition into the automaton
- delete the location (and all the transitions to and fro)

Elimination Example



$\forall\Box\ x + y \leq 10$

Reachability Theory for B

$A(x,y) \rightarrow x \leq y \rightarrow B(x,y)$
$B(x,y) \rightarrow x \leq y$
$B(x,y) \rightarrow x + y \leq 10$
$B(x,y) \rightarrow \forall\delta\ 0 \leq \delta \wedge x' = x + 2\delta \wedge y' = y + \delta \wedge x' \leq y' \rightarrow B(x',y')$
$B(x,y) \rightarrow x = y \rightarrow C(0,0)$

Elimination Approach

Reachability Theory simplified

$A(x,y) \rightarrow x \leq y \rightarrow B(x,y)$
$B(x,y) \rightarrow x \leq y$
$B(x,y) \rightarrow x + y \leq 10$
$B(x,y) \rightarrow x \leq x' \wedge x + 2 * y' = x' + 2 * y \wedge x' \leq y' \rightarrow B(x',y')$
$B(x,y) \rightarrow x = y \rightarrow C(0,0)$

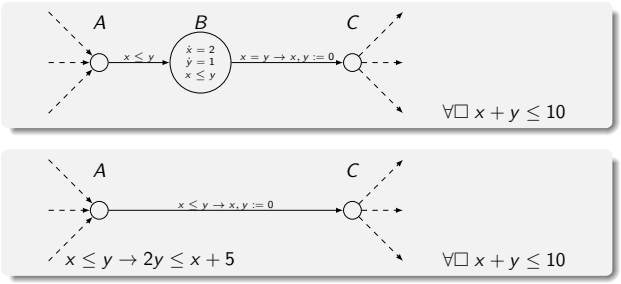
Fixpoint Computation (Definition for B)

$B(x,y) \rightarrow x \leq y \rightarrow C(0,0)$
$B(x,y) \rightarrow x \leq y \rightarrow 2 * y \leq x + 5$

Insertion (in A)

$A(x,y) \rightarrow x \leq y \rightarrow C(0,0)$
$A(x,y) \rightarrow x \leq y \rightarrow 2 * y \leq x + 5$

Elimination Result

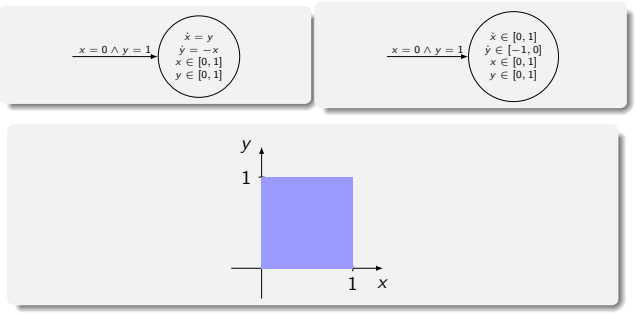


Elimination Approach

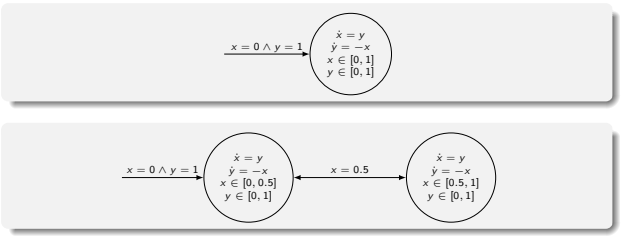
Advantages

- with each elimination the verification problem decreases
- no need for multiple turns through the automaton
- in a sense mixes (and generalizes) standard reachability approaches

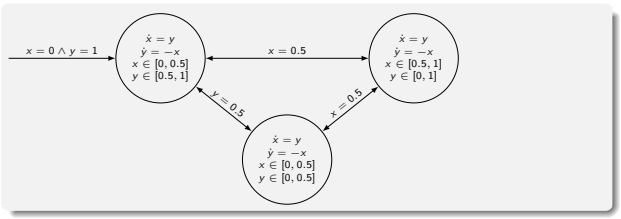
Approximation of Affine Behavior



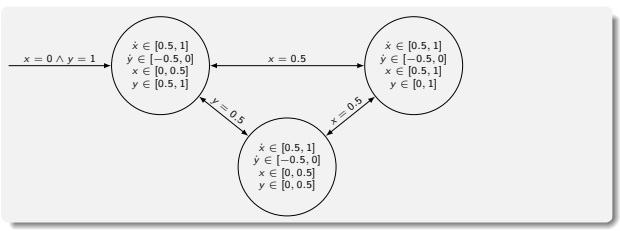
Location Splitting



One More Splitting



One More Splitting



Eliminating A

Positive A-clauses

$x = 0 \wedge y = 1 \to A(x, y)$	initial state
$B(x, y) \to x = 0.5 \wedge y \in [0.5, 1] \to A(x, y)$	from B to A
$C(x, y) \to y = 0.5 \wedge x \in [0, 0.5] \to A(x, y)$	from C to A
$A(x, y) \to y' \le y \wedge x' \in [0, 0.5] \wedge y' \in [0.5, 1] \wedge x + y \le x' + y' \to A(x', y')$	continuous change

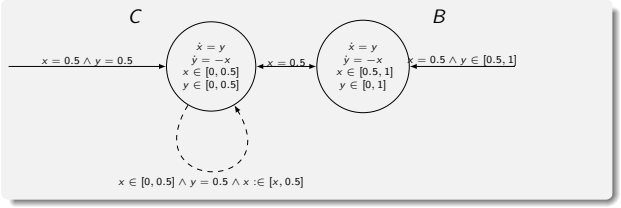
Fixpoint Computation and Definition of A

$x \in [0, 0.5] \wedge y \in [0.5, 1] \wedge 1 \le x + y \to A(x, y)$
$C(x, y) \to y = 0.5 \wedge y' = 0.5 \wedge x \in [0, 0.5] \wedge x \le x' \wedge x' \in [0, 0.5] \to A(x', y')$

Insertion of A's Definition

$x = 0.5 \wedge y \in [0.5, 1] \to B(x, y)$
$x = 0.5 \wedge y = 0.5 \to C(x, y)$
$C(x, y) \to x \in [0, 0.5] \wedge y = 0.5 \wedge x' \in [x, 0.5] \wedge y' = y \to C(x', y')$

After Eliminating A



Eliminating C

Positive C-clauses

$x = 0.5 \wedge y = 0.5 \rightarrow C(x, y)$
$B(x, y) \rightarrow x = 0.5 \wedge y \in [0, 0.5] \rightarrow C(x, y)$
$C(x, y) \rightarrow x \leq x' \wedge y' \leq y \wedge x' \in [0, 0.5] \wedge y' \in [0, 0.5] \rightarrow C(x', y')$

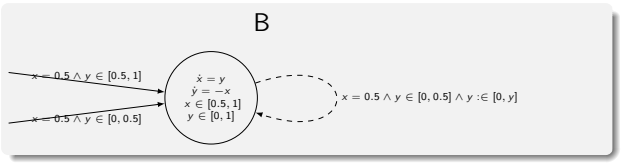
Fixpoint Computation and Definition of C

$x = 0.5 \wedge y \in [0, 0.5] \rightarrow C(x, y)$
$B(x, y) \rightarrow x = 0.5 \wedge y \in [0, 0.5] \wedge x' = 0.5 \wedge y' \in [0, y] \rightarrow C(x', y')$

Insertion of C's Definition

$x = 0.5 \wedge y \in [0, 0.5] \rightarrow B(x, y)$
$B(x, y) \rightarrow x = 0.5 \wedge y \in [0, 0.5] \wedge x' = 0.5 \wedge y' \in [0, y] \rightarrow B(x', y')$

After Eliminating C



Eliminating B

Positive B-clauses

$x = 0.5 \wedge y \in [0.5, 1] \rightarrow B(x, y)$
$x = 0.5 \wedge y \in [0, 0.5] \rightarrow B(x, y)$
$B(x, y) \rightarrow x \leq x' \wedge y' \leq y \wedge x' + 2y' \leq x + 2y \wedge x' \in [0.5, 1] \wedge y' \in [0, 1] \rightarrow B(x', y')$

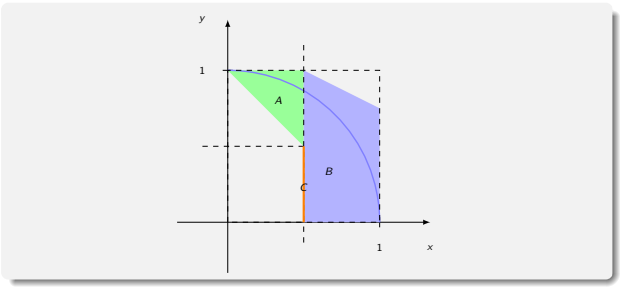
Fixpoint Computation and Definition of B

$x + 2y \leq 2.5 \wedge x \in [0.5, 1] \wedge y \in [0, 1] \rightarrow B(x, y)$

Final Insertion and Result

$x \in [0, 0.5] \wedge y \in [0.5, 1] \wedge 1 \leq x + y \rightarrow A(x, y)$
$x + 2y \leq 2.5 \wedge x \in [0.5, 1] \wedge y \in [0, 1] \rightarrow B(x, y)$
$x = 0.5 \wedge y \in [0, 0.5] \rightarrow C(x, y)$

After Eliminating All



Summary

- Modelling of systems with continuous state changes requires different techniques
- Inspired by state machines, but with continuous behaviour in states expressed by first derivatives
- Different aspects
  - Timed Automata
  - Multi-Phase Automata
  - Rectangular Automata
  - Affine Automata
- Properties formulated using CTL;
- Verification approaches beyond forward/bachward reachability analysis

Formale Modellierung
Vorlesung 14 vom 21.07.2014: Zusammenfassung, Rückblick, Ausblick

Serge Autexier & Christoph Lüth

Universität Bremen

Sommersemester 2014

Fahrplan

- Teil I: Formale Logik
- Teil II: Spezifikation und Verifikation
  - Formale Modellierung mit der UML und OCL
  - Lineare Temporale Logik
  - Temporale Logik und Modellprüfung
  - Hybride Systeme
  - Zusammenfassung, Rückblick, Ausblick

Heute in diesem Theater

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

Unsere Reise durch die Logik

	Entscheidbar?	Vollständig?	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, PVS

Aussagenlogik

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

Prädikatenlogik

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

Induktion und Logik höherer Stufe

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

UML

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

Temporallogik

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

Modellierung, formale Modellierung, Programme und formale Methoden

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

Formale Modellierung: Geschichtlicher Rückblick

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

Formale Methoden: Geschichtlicher Rückblick

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

Automatisches Theorembeweisen

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

Formale Modellierung und Formale Methoden

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

Formale Methoden

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

Formale Modellierung in der Mathematik

- Perelmann und Poincaré; Andrew Wiles und Fermat's letztes Theorem; die Riemannsche Vermutung
- Vierfarbenproblem (Appel-Haken, 1970)
- Vierfarbenproblem in Coq (Gonthier, 2005)
- Die Keplersche Vermutung und Flyspeck (Hales, ab 2002?)

Stand der Kunst

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

... und jetzt?

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

Tschüß!