Zusammenfassung: Logik für die Informatik

Rico Klimpel

February 11, 2020

Contents

Informationen

Zusammenfassung der Vorlesung Logik für die Informatik an der CAU Kiel aus dem Wintersemester 2019/2020, gehalten von Prof. Dr. Thomas Wilke. Ein Versuch die wichtigsten Aussagen ohne enorm lange Formalitäten drum herum knapp zu Papier zu bringen. Kein Anspruch auf Vollständigkeit. Geschrieben in LaTeX.

Part I
Aussagenlogik

Part II
Prädikatenlogik

1 Syntax & Semantik

1.1 Signatur

`https://lili.informatik.uni-kiel.de/llocs/Signatures`

Eine Signatur $\mathcal{S}$ besteht aus eine Menge S von Symbolen und einer Funktion $\Sigma\colon S \to \mathbb{N} \cup \mathbb{N} \times \{1\}$.
The Elemente von S werden Symbole genannt und wie folgt eingeteilt:

- Ein Symbol f mit $\Sigma(f) = \langle n, 1\rangle$ für $n > 0$ ist eine Funktionssymbol.
  Menge dieser Symbole: $\mathcal{F}_\Sigma$ oder einfach $\mathcal{F}$.

- Ein Symbol R mit $\Sigma(R) = n$ für $n > 0$ ist ein Relationssymbol.
  Menge dieser Symbole: $\mathcal{R}_\Sigma$ oder $\mathcal{R}$.

- Ein Symbol c mit $\Sigma(c) = \langle 0, 1\rangle$ ist ein Symbol für eine Konstante.
  Menge dieser Symbole: $\mathcal{C}_\Sigma$ oder $\mathcal{C}$.

- Symbol b mit $\Sigma(b) = 0$ ist ein Symbol für einen boolschen Wert.
  Menge dieser Symbole: $\mathcal{B}_\Sigma$ oder $\mathcal{B}$.

Im allgemeinen werden Signaturen mit $\mathcal{B} \neq \emptyset$ ignoriert (Signaturen ohne boolsche Werte). Keine Ahnung warum er das sagt.

Beispiele:

$$S = \{\text{zero}, \text{one}, \text{add}, \text{mult}\}$$
$$\Sigma = \{\text{zero} \mapsto \langle 0,1\rangle, \text{one} \mapsto \langle 0,1\rangle, \text{add} \mapsto \langle 2,1\rangle, \text{mult} \mapsto \langle 2,1\rangle\}$$

Vereinfacht aufgeschrieben sieht das ganze so aus:

$$\mathcal{S} = \{\text{zero}, \text{one}, \text{add}//2, \text{mult}//2\}$$

1.2 Struktur

`https://lili.informatik.uni-kiel.de/llocs/Structures`

Sei $\mathcal{S}$ eine Signatur. Eine $\mathcal{S}$-Struktur $\mathcal{A}$ besteht aus:

- Univserum A mit $A \neq \emptyset$

- Für jedes Symbol eine Konstanten $c \in \mathcal{S}$ eine Interpretation $c^{\mathcal{A}} \in A$ von c.

- Für jedes Funktionssymbol $f//n \in \mathcal{S}$ eine Interpretation $f^{\mathcal{A}}\colon A^n \to A$

- Für jedes Relationssymbol $R/n \in \mathcal{S}$ eine Interpretation $R^{\mathcal{A}} \subseteq A^n$

Hier ein Beispiel das ungefähr zu der Signatur oben passt:

$$\begin{aligned}
A &= \{0, 1, 2, 3\} \\
\text{zero}^{\mathcal{A}} &= 3 \\
\text{one}^{\mathcal{A}} &= 2 \\
\text{add}^{\mathcal{A}}(a, b) &= 0 && \text{for } a, b \in A \\
\text{mult}^{\mathcal{A}}(a, b) &= a + b \text{ rest } 4 && \text{for } a, b \in A \\
\text{Lt}^{\mathcal{N}} &= \{\langle a, a\rangle : a \in A\}
\end{aligned}$$

1.3 Terme

`https://lili.informatik.uni-kiel.de/llocs/Syntax_of_first-order_logic#Formal_definition_of_terms`

Induktive Defintion für alle Terme über eine Signatur $\mathcal{S}$, die auch $\mathcal{S}$-terms genannt wird: Basiselemente:

- Ein Baum mit nur einem Element das eine Variable der Prädikatenlogik enthält ist ein $\mathcal{S}$-term.
- Ein Baum mit nur einem Element das eine Konstante $c \in \mathcal{S}$ enthält ist ein $\mathcal{S}$-term.

Diese werden die atomaren $\mathcal{S}$-terme genannt. Induktionsregeln:

- Wenn $f//n \in \mathcal{S}$ eine Funktion und $t_0, \ldots, t_{n-1}$ $\mathcal{S}$-terms sind, dann ist der Baum mit der Wurzel f und den n Teilbäumen t_0, ..., t_{n-1} ein $\mathcal{S}$-term.

1.4 Formeln

`https://lili.informatik.uni-kiel.de/llocs/Syntax_of_first-order_logic#Formal_definition_of_formulas`

Induktive Defintion für alle Formeln über eine Signatur $\mathcal{S}$, die auch $\mathcal{S}$-formulas genannt wird: Basiselemente:

- Der einelementige Baum in dem das einzige Element eines der konstanten Symbole $\top$ oder $\perp$ ist, ist eine (prädikatenlogische) Formel.
- Wenn t_0, t_1 Terme sind, dann ist der Baum mit der Wurzel $\doteq$ und den Teilbäumen t_0 und t_1 eine Formel.
- Wenn $R/n \in \mathcal{S}$ eine Relation ist und $t_0, \ldots, t_{n-1}$ Terme sind dann ist der Baum mit der Wurzel R und den n Teilbäumenm $t_0, \ldots, t_{n-1}$ eine Formel.

Diese werden die atomaren Formeln genannt. Induktionsregeln:

- Wenn C ein n-stelliger Junktor ist und $\varphi_0, \ldots, \varphi_{n-1}$ Formeln sind, dann ist der Baum mit der Wurzel C und den n Teilbäumen $\varphi_0, \ldots, \varphi_{n-1}$ eine Formel.
- Wenn x_i eine Variable ist und φ eine Formel, dann ist der Baum mit der Wurzel $\exists x_i$ oder der Wurzel $\forall x_i$ und dem Teilbaum φ eine Formel.

1.5 Interpretation von Termen

`https://lili.informatik.uni-kiel.de/llocs/Semantics_of_first-order_logic#Interpretation_of_terms`

Sei $\mathcal{S}$ eine Signatur und $\mathcal{A}$ eine $\mathcal{S}$-Struktur. Für eine Belegung (A-Belegung) β, ist der Wert von jedem $\mathcal{S}$-term t in $\mathcal{A}$ unter β: $[\![t]\!]^{\mathcal{A}}_{\beta}$ defniert durch folgender Induktion. Basiselemente:

- Für alle $i \in \mathbb{N}$ gilt: $[\![x_i]\!]^{\mathcal{A}}_{\beta} = \beta(x_i)$.
  Variablen bekommen den ihnen unter β zugewiesenen Wert bei der alleinigen Auswertung.
- Für jedes $c \in C$ gilt: $[\![c]\!]^{\mathcal{A}}_{\beta} = c^{\mathcal{A}}$
  Konstante Symbole werden wie in der Struktur beschriegben ausgwertet wenn sie alleine stehen.

Induktionsregel:

- Für alle $f//n \in \mathcal{F}$ und die $\mathcal{S}$-terms $t_0, \dots, t_{n-1}$ gilt: $[\![f(t_0, \dots, t_{n-1})]\!]^{\mathcal{A}}_{\beta} = f^{\mathcal{A}}([\![t_0]\!]^{\mathcal{A}}_{\beta}, \dots, [\![t_{n-1}]\!]^{\mathcal{A}}_{\beta})$

1.6 Interpretation von Formlen

`https://lili.informatik.uni-kiel.de/llocs/Semantics_of_first-order_logic#Interpretation_of_formulas`

So dieses mal einfach direkt die Induktive Definition: Basiselemente:

- Für die boolschen konstannten Symbole gilt:

$$[\![\bot]\!]^{\mathcal{A}}_{\beta} = 0 \tag{1}$$

$$[\![\top]\!]^{\mathcal{A}}_{\beta} = 1 \tag{2}$$

- Für alle Terme t_0, t_1 gilt:

$$[\![t_0 \doteq t_1]\!]^{\mathcal{A}}_{\beta} = \begin{cases} 1 & \text{if } [\![t_0]\!]^{\mathcal{A}}_{\beta} = [\![t_1]\!]^{\mathcal{A}}_{\beta} \\ 0 & \text{sonst} \end{cases} \tag{3}$$

- Für alle Relationen $R/n \in \mathcal{R}$ und Terme $t_0, \dots, t_{n-1}$ gilt:

$$[\![R(t_0, \dots, t_n)]\!] = \begin{cases} 1 & \text{if } \langle [\![t_0]\!]^{\mathcal{A}}_{\beta}, \dots, [\![t_{n-1}]\!]^{\mathcal{A}}_{\beta} \rangle \in R^{\mathcal{A}} \\ 0 & \text{sonst} \end{cases} \tag{4}$$

Induktionsregeln:

- Für jeden n-stelligen Junktor C und die Formlen $\varpi_0, ..., \varphi_{n-1}$ gilt:

$$[\![C(\varphi_0, \dots, \varphi_{n-1})]\!]^{\mathcal{A}}_{\beta} = f_C([\![\varphi_0]\!]^{\mathcal{A}}_{\beta}, \dots, [\![\varphi_{n-1}]\!]^{\mathcal{A}}_{\beta})$$

- Für jede Formel φ und $i \in \mathbb{N}$ gilt:

$$[\![\exists x_i \varphi]\!]^{\mathcal{A}}_{\beta} = \begin{cases} 1 & \text{wenn ein } a \in A \text{ existiert,} \\ & \text{für das gilt } [\![\varphi]\!]^{\mathcal{A}}_{\beta[\frac{x_i}{a}]} = 1 \\ 0 & \text{sonst} \end{cases}$$

$$[\![\forall x_i \varphi]\!]^{\mathcal{A}}_{\beta} = \begin{cases} 1 & \text{wenn für alle } a \in A \text{ gilt:} \\ & [\![\varphi]\!]^{\mathcal{A}}_{\beta[\frac{x_i}{a}]} = 1 \\ 0 & \text{sonst} \end{cases}$$

$[\![\varphi]\!]^{\mathcal{A}}_{\beta} = 1$ wird auch als $\mathcal{A}, \beta \models \varphi$ geschrieben und es heißt $\mathcal{A}$ zusammen mit β ist ein Modell für φ. Dies gilt auch für eine Menge von Formeln ϕ mit $\mathcal{A}, \beta \models \phi$ ist dann gemeint das es ein Modell für alle Formeln der Menge ist.

1.7 Freie Variablen

`https://lili.informatik.uni-kiel.de/llocs/Free_variables`

Freie Variablen sind genau die, deren Belegung für die Auswertung eiens Terms von relevanz sind. Auch die freien Variablen wurden Induktiv definiert, dabei ist $\mathcal{S}$ eine Signatur:

Basiselemente:

- $\text{fvars}(\top) = \emptyset$ und $\text{fvars}(\bot) = \emptyset$.
- Für alle Terme t_0 und t_1, $\text{fvars}(t_0 \doteq t_1) = \text{vars}(t_0) \cup \text{vars}(t_1)$.
- Für alle Relationssymbole $R/n \in \mathcal{S}$ und Terme $t_0, \dots, t_n$ gilt $\text{fvars}(R(t_0, \dots, t_{n-1})) = \bigcup_{i<n} \text{vars}(t_i)$.

Induktionsregeln:

- Für jeden n-stelligen Junktor C und Formlen $\varphi_0, \dots, \varphi_{n-1}$ gilt $\text{fvars}(C(\varphi_0, \dots, \varphi_{n-1})) = \bigcup_{i<n} \text{fvars}(\varphi_i)$.
- Für alle $i \in \mathbf{N}$ und alle Formeln φ gilt $\text{fvars}(\exists x_i \varphi) = \text{fvars}(\varphi) \setminus \{x_i\}$ und $\text{fvars}(\forall x_i \varphi) = \text{fvars}(\varphi) \setminus \{x_i\}$.

Freie Variablen sind demnoch genau die ganzen Variablen in einer Formel, die nicht durch ein $\forall$ oder $\exists$ in genau der Formel φ dahinter gebunden sind.

1.8 Koinzidenzlemma

`https://lili.informatik.uni-kiel.de/llocs/Coincidence_lemma_(first-order_logic)`

Sei $\mathcal{S}$ eine Signatur, φ eine Formel über $\mathcal{S}$, $\mathcal{A}$ eine $\mathcal{S}$-Struktur und β_0 und β_1 Belegungen für Variablen. Wenn $\beta_0|_{\text{fvars}(\varphi)} = \beta_1|_{\text{fvars}(\varphi)}$, dann gilt:

$$[\![\varphi]\!]_{\beta_0} = [\![\varphi]\!]_{\beta_1}$$

Das heißt, das für die Auswertung einer Formel nur genau die Variablen von Relevanz sind, die auch in der Formel vorkommen, und zwei verschiedene Belegungen, die aber in genau den vorkommenden Variablen gleich sind. Auch beim Auswerten von φ auf das gleiche Ergebnis kommen.

2 Modelierung

2.1 Relationen in Strukturen defnieren?

`https://lili.informatik.uni-kiel.de/llocs/Definable_relations`

2.2 Erfüllbarkeit einer Formel

https://lili.informatik.uni-kiel.de/llocs/Satisfiability_(first-order_logic)

3 Äquivalenz

3.0.1 Plenex Normalform

https://lili.informatik.uni-kiel.de/llocs/Prenex_normal_form

3.0.2 Konjunktive Normalform

https://lili.informatik.uni-kiel.de/llocs/Conjunctive_normal_form_(first-order_logic)

4 Folgerungsbeziehungen (Entailment)

4.1 Folgerungsbeziehung

4.2 Äquivalenz von Formeln

https://lili.informatik.uni-kiel.de/llocs/Formula_equivalence_(first-order_logic)

4.3 Regeln der Prädikatenlogik

4.4 Quantorenregeln

https://lili.informatik.uni-kiel.de/llocs/Quantifier_laws

4.5 Umbenennen von gebundenen Variablen

https://lili.informatik.uni-kiel.de/llocs/Quantifier_laws

4.6 Namenskonflikte

https://lili.informatik.uni-kiel.de/llocs/Term_substitution_and_substitution_lemma#Substitution_lemma

Sei φ eine prädikatenlogische Formel und σ eine Term Substitution. Es gibt einen Namenskonflikt in φ für σ wenn ein $x_i \in \mathrm{dom}(\sigma)$ existiert und $x_j \in \mathrm{vars}(\sigma(x_i))$ so dass $x_i \in \mathrm{scope}(x_j, \varphi)$.
Das bedeutet: Die Domain einer Termsubstitution sind genau die Variablen die durch die Substitution ersetzt werden sollen. Wenn nun in der Formel genau im Scope einer Variablen aus dieser Domain die gleiche Variable (ungebunden) auftraucht, wie auch in dem ersetzten Teil der Substitution vorkommt Dann ist es ein Namenskonflikt. Immer noch zu kompliziert? Dann hilft vielleicht nochmal die Vereinfachte Formel:

$$\exists x_i \in \mathrm{dom}(\sigma) \exists x_j \in \mathrm{vars}(\sigma(x_i)) : x_i \in \mathrm{scope}(x_j, \varphi)$$

Dann ist da ein Namenskonflikt.

4.7 Scope von Quantoren

https://lili.informatik.uni-kiel.de/llocs/Scope_of_quantifiers_in_first-order_logic

Die Menge der freien Variablen im "scope" einer durch einen Quantor gebundenen Variablen x_i einer prädikatenlogischen Formel φ wird aufgeschrieben als $\mathrm{scope}(x_i, \varphi)$ und ist über die folgende Induktion definiert:

Basiselemente:

- $\mathrm{scope}(x_i, \bot) = \emptyset$ und $\mathrm{scope}(x_i, \top) = \emptyset$.
- $\mathrm{scope}(x_i, R(t_0, \dots, t_{n-1})) = \emptyset$ für jedes n-stelliges Relationssymbol R und die Terme $t_0, \dots, t_{n-1}$.
- $\mathrm{scope}(x_i, t_0 \doteq t_1) = \emptyset$ für alle Terme t_0 und t_1.

Anders gesagt: $\mathrm{scope}(x_i, \varphi) = \emptyset$ für jede atomare Formel φ. Nicht kompliziert.

Induktionsregeln:

- $\mathrm{scope}(x_i, C(\varphi_0, \dots, \varphi_{n-1})) = \bigcup_{i<n} \mathrm{scope}(x_i, \varphi_j)$ für jeden n-stelligen Junktor C und alle Formeln $\varphi_0, \dots, \varphi_{n-1}$.

  Das heißt wenn man scope auf einen Junktor aufruft, so betrachtet man alle Formeln die in dem Junktor vorkommen und ruft auf diese scope auf.

- Dies ist der entscheidende Teil:
  $\mathrm{scope}(x_i, Qx_j\varphi) = \begin{cases} \mathrm{fvars}(\varphi) \setminus \{x_i\} & \text{if } i = j \\ \mathrm{scope}(x_i, \varphi) \setminus \{x_j\} & \text{sonst} \end{cases}$ für $Q \in \{\exists, \forall\}$ und die Formel φ.
  Das heißt: Ruft man scope auf einen Quantoren $\exists$ oder $\forall$ auf, so enstpricht scope genau dann den freien Variablen der Formel hinter dem Quantor, wenn die in der Scope Funktion angegeben Variable der vor dem Quantor entspricht. Ansonsten rufen wir auf die Formel die zu dem Quantor gehört erneut scope auf, lassen aber die durch den Quantor gebundene Variable außenvor.

4.8 Normalformen

4.8.1 Boolsche Normalform

https://lili.informatik.uni-kie

4.9 Beziehung zwischen Erfüllbarkeit und Folgerungsbeziehung

---

5 Beweissysteme

5.1 Natürliches Beweissystem

5.1.1 Beweisregeln

5.1.2 Korrektheit & Vollständigkeit

5.2 Resolutionsbeweise

5.2.1 Korrektheit & Vollständigkeit

5.2.2 Verbindung zwischen Resolution und Logik-Programmierung

---

6 Kompaktheit

---

7 Klausuraufgabentypen

Prädikatenlogik:

- Ankreuzaufgaben richtig/falsch
- Angeben ob Formeln erfüllbar/unerfüllbar/allgmeingültig (mit Beweis)
- Formeln für bestimmte Problemstellungen angeben
- Äquivalenz von Formeln Beweisen/Widerlegen
- Modell für erfüllbare Formlen angeben
- Natürlicher Beweis für Allgmeingültigkeit
- Resultionsbeweis für Folgerungsbeziehung