Humboldt-Universität zu Berlin
Lehrstuhl Logik in der Informatik
Prof. Dr. Nicole Schweikardt

13. Januar 2023

Logik in der Informatik

Wintersemester 2022/2023

Übungsblatt 11

Abgabe: *bis 23. Januar 2023, 13.00 Uhr*

Aufgabe 1: (Moodle-Quiz)

Absolvieren Sie das Quiz 11 auf der Moodle-Plattform.

Aufgabe 2: (Präsenzaufgabe)

Wir betrachten in dieser Aufgabe Kalküle über der Menge $M := \mathsf{AL}(\{\neg, \rightarrow\})$.

Ein Kalkül $\mathfrak{K}$ über M heißt *korrekt*, falls für jede Menge $\Phi \subseteq M$ und jede Formel $\psi \in M$ gilt: Wenn $\psi \in \mathrm{abl}_{\mathfrak{K}}(\Phi)$, dann gilt $\Phi \models \psi$. Ein Kalkül $\mathfrak{K}$ über M heißt *vollständig*, falls für jede Menge $\Phi \subseteq M$ und jede Formel $\psi \in M$ gilt: Wenn $\Phi \models \psi$, dann ist $\psi \in \mathrm{abl}_{\mathfrak{K}}(\Phi)$.

Seien $\mathfrak{K}_{Syl}$ und $\mathfrak{K}_{Abd}$ die beiden folgenden Kalküle über der Menge M: Beide Kalküle enthalten für jede *allgemeingültige* aussagenlogische Formel $\varphi \in M$ das Axiom $\dfrac{}{\varphi}$.

Außerdem enthält

- $\mathfrak{K}_{Abd}$ für alle $\varphi, \psi \in M$ die Ableitungsregel

$$\frac{\psi \qquad (\varphi \rightarrow \psi)}{\varphi},$$

  die *Abduktion* genannt wird,

- $\mathfrak{K}_{Syl}$ für alle $\varphi, \psi, \chi \in M$ die Ableitungsregel

$$\frac{(\varphi \rightarrow \psi) \qquad (\psi \rightarrow \chi)}{(\varphi \rightarrow \chi)},$$

  die *Syllogismus* genannt wird.

(a) Geben Sie für $\varphi := \neg(A_0 \rightarrow A_0)$ und $\Phi := \emptyset$ eine möglichst kurze Ableitung von φ aus Φ in $\mathfrak{K}_{Abd}$ an.

(b) Beweisen Sie, dass $\mathfrak{K}_{Abd}$ vollständig, aber nicht korrekt ist.

(c) Beweisen Sie, dass $\mathfrak{K}_{Syl}$ korrekt, aber nicht vollständig ist.

(d) Betrachten Sie den Kalkül $\mathfrak{K}$ über M, der alle Ableitungsregeln aus $\mathfrak{K}_{Syl}$ und alle Ableitungsregeln aus $\mathfrak{K}_{Abd}$ enthält. Ist $\mathfrak{K}$ korrekt? Ist $\mathfrak{K}$ vollständig? Begründen Sie jeweils Ihre Antwort.

Aufgabe 3: (40 Punkte)

(a) **Beweisen oder widerlegen** Sie die folgende Aussage:
Sei $\sigma = \emptyset$. Es gibt einen $\mathsf{FO}[\sigma]$-Satz φ, so dass für jede endliche σ-Struktur $\mathcal{A}$ gilt:

$$\mathcal{A} \models \varphi \quad \Longleftrightarrow \quad |A| \text{ ist eine Primzahl.}$$

(b) Betrachten Sie das Alphabet $A = \{\mathtt{M}, \mathtt{I}, \mathtt{U}\}$ und die Menge $M := A^*$.
Sei $\mathfrak{K}$ der Kalkül über M, der aus den folgenden Regeln besteht:

- $\mathfrak{K}$ enthält das Axiom

$$\frac{}{\mathtt{MI}}$$

- Für alle $w, w' \in A^*$ enthält $\mathfrak{K}$ die Regeln

$$\frac{w\mathtt{I}}{w\mathtt{IU}} \quad \text{und} \quad \frac{\mathtt{M}w}{\mathtt{M}ww} \quad \text{und} \quad \frac{w\mathtt{III}w'}{w\mathtt{U}w'} \quad \text{und} \quad \frac{w\mathtt{UU}w'}{ww'}$$

(i) Geben Sie eine rekursive Definition der Menge $\mathrm{abl}_{\mathfrak{K}}$ an.

(ii) Welche der folgenden Worte sind aus $\mathfrak{K}$ ableitbar, welche nicht? Begünden Sie Ihre Antwort, indem Sie entweder ein Ableitung des Wortes in $\mathfrak{K}$ angeben oder erläutern, warum es keine solche Ableitung geben kann.

(i) `MIU` **(iii)** `MUII`
(ii) `UMII` **(iv)** `MU`

(iii) Beweisen Sie durch Nutzen des Induktionsprinzips, dass für jedes Wort $w \in \mathrm{abl}_{\mathfrak{K}}$ gilt:

Die Anzahl $|w|_{\mathtt{I}}$ der Vorkommen des Buchstabens `I` in w ist *nicht* durch 3 teilbar.

(c) Sei σ eine Signatur, sei $\Gamma \subseteq_e \mathsf{FO}[\sigma]$, seien $\varphi, \psi, \chi \in \mathsf{FO}[\sigma]$ und seien $x, y \in \mathsf{VAR}$.
Beweisen Sie die Korrektheit der folgenden Sequenzenregeln:

(i) $\wedge$-Einführung im Sukzedens ($\wedge$S):

$$\frac{\Gamma \vdash \quad \varphi \qquad \Gamma \vdash \quad \psi}{\Gamma \vdash (\varphi \wedge \psi)}$$

(ii) $\exists$-Einführung im Antezedens ($\exists$A):

$$\frac{\Gamma, \varphi^{\frac{y}{x}} \vdash \psi}{\Gamma, \exists x \varphi \vdash \psi} \qquad \text{falls} \quad y \notin \mathrm{frei}(\Gamma, \exists x \varphi, \psi)$$

Aufgabe 4: (20 Punkte)

Lesen Sie Kapitel 12 aus dem Buch „Learn Prolog Now!“.
Achtung: *Die Bearbeitung der Aufgabe ist unter Beachtung der bekannten Abgabehinweise über Moodle abzugeben! Analog zu früheren Blättern finden Sie die benötigten Dateien auf der Seite zur Prolog-Übung.*

(a) Machen Sie sich mit den Prolog-Modulen `al_def`, `al_literals` und `al_nf` vertraut, welche Sie auf der Seite zur Prolog-Übung finden können. Laden Sie diese Prolog-Module in ein Verzeichnis Ihrer Wahl.

(b) Erstellen Sie (im selben Verzeichnis) in einer Datei `blatt11.pl` ein Modul mit dem Namen `pure_literal`, das die Prädikate `knf_shell/0` und `pure_literal/2` exportiert.

(c) Importieren Sie im Modul `pure_literal` genau die Prädikate aus den Modulen `al_def`, `al_literals` und `al_nf`, die Sie zum Lösen der folgenden beiden Teilaufgaben benötigen.

(d) Wir kodieren Klauselmengen wie auf Blatt 10 als Listen von Listen von Literalen.

Implementieren Sie das Prädikat `knf_shell/0`, so dass eine Anfrage

```
?- knf_shell.
```

eine Eingabeaufforderung zur Konstruktion von Klauselmengen aus aussagenlogischen Formeln startet. D.h., wenn über die Tastatur eine aussagenlogische Formel als Prolog-Term eingegeben wird, dann soll nach Ende der Eingabe (durch `.` und die Taste „Enter“) eine zu der Formel äquivalente Klauselmenge ausgegeben werden. Dies soll so lange wiederholt werden, bis statt einer aussagenlogischen Formel das Atom `bye` (wieder gefolgt durch `.` und die Taste „Enter“) eingegeben wird.

Hinweise: Definieren Sie sich gegebenenfalls geeignete Hilfsprädikate. Verwenden Sie für die Eingabe das Prädikat `read/1` und für die Ausgabe das Prädikat `write/1`. Beide Prädikate sind in SWI-Prolog vordefiniert. Sie müssen sich nicht um die Behandlung von Eingabefehlern kümmern.

(e) *Zur Erinnerung: Pure Literal Rule*

> Literale λ, deren Negat $\overline{\lambda}$ nirgendwo in der Klauselmenge auftaucht, können auf 1 gesetzt werden. Alle Klauseln, die ein solches Literal enthalten, sind dann wahr und können gestrichen werden. Wiederhole dies, so lange es Literale gibt, deren Negat nirgendwo in der Klauselmenge auftaucht.

Implementieren Sie ein Prädikat `pure_literal/2`, so dass eine Anfrage von der Form

```
?- pure_literal(KM, KM2).
```

auf die Klauselmenge `KM` die *Pure Literal Rule* des DPLL-Algorithmus solange anwendet, wie es Literale gibt, deren Negat nirgendwo in der Klauselmenge auftaucht und die entstehende Klauselmenge in `KM2` zurückgibt. Beispielsweise sollte die Anfrage

```
?- pure_literal([[~x1, x2, ~x5], [x1, x2, ~x4, x7], [x3, ~x5, x7],
                 [x3, ~x4, ~x5], [x5,x4,~x8], [x1,x3,x5,x7],
                 [~x7,x8]], KM2).
```

zu der Antwort

```
KM2 = [].
```

führen.

Hinweise: Definieren Sie geeignete Hilfsprädikate. *Beispielsweise* bietet es sich an, Prädikate `is_literal/2` und `is_pure_literal/2` einzuführen, so dass das Ziel `is_literal(L, KM)` für jedes in der Klauselmenge `KM` vorkommende Literal `L` erfüllt ist, und so dass das Ziel

`is_pure_literal(L, KM)` für jedes in der Klauselmenge `KM` vorkommende Literal `L` erfüllt ist, dessen Negat nicht in `KM` vorkommt.