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

22. Januar 2021

Logik in der Informatik

Wintersemester 2020/2021

Übungsblatt 10

Abgabe: *bis 1. Februar 2021, 13.^{00} Uhr via Moodle*

*Um unseren Tutor*innen die Arbeit etwas zu erleichtern, notieren Sie bitte auf dem ersten Blatt jeder schriftlich erstellten Abgabedatei oben rechts eingerahmt folgende Daten:*
<Vorname Nachname>, <Matrikelnr.>, <Moodle-Loginname>

Aufgabe 1: (Moodle-Quiz)

Absolvieren Sie das Quiz 10 auf der Moodle-Plattform.

Aufgabe 2: (Präsenzaufgabe)

Hunde äußern sich bekanntlich mit Hilfe der Laute „W“, „A“ und „U“.

Sei $\Sigma := \{\mathrm{W}, \mathrm{A}, \mathrm{U}\}$ und sei die *Hundesprache* H definiert durch $H := \mathrm{abl}_{\mathfrak{K}}$, wobei $\mathfrak{K}$ der folgende Kalkül über der Menge Σ^* ist:

$$
\begin{aligned}
\mathfrak{K} := \quad & \left\{ \frac{}{\mathrm{WA}} \right\} \\
\cup \quad & \left\{ \frac{v}{vv} : \text{ für alle } v \in \Sigma^* \right\} \\
\cup \quad & \left\{ \frac{v\mathrm{A}w}{v\mathrm{AU}w} : \text{ für alle } v, w \in \Sigma^* \right\} \\
\cup \quad & \left\{ \frac{v\mathrm{UU}w}{v\mathrm{AAA}w} : \text{ für alle } v, w \in \Sigma^* \right\} \\
\cup \quad & \left\{ \frac{v\mathrm{AAA}w}{vw} : \text{ für alle } v, w \in \Sigma^* \right\}
\end{aligned}
$$

(a) Geben Sie für jedes der folgenden Worte aus Σ^* an, ob es zur Menge H gehört oder nicht. Begründen Sie jeweils Ihre Antwort!

(i) WA **(ii)** UWAA **(iii)** WAWAUU **(iv)** WU

(b) Zeigen Sie, dass für jedes Wort $w \in H$ gilt:
Die Anzahl $|w|_A$ der Vorkommen des Lauts A in w ist *nicht* durch 3 teilbar
(d.h., es gibt eine Zahl $k \in \mathbb{N}$, so dass gilt: $|w|_A = 3k + 1$ oder $|w|_A = 3k + 2$).

(c) Kann ein Hund „WAAA“ machen? D.h., ist WAAA $\in H$?

Aufgabe 3: (40 Punkte)

(a) Sei $\sigma = \{E/2\}$. Betrachten Sie die $\mathsf{FO}[\sigma]$-Formel

$$\varphi(x) \;:=\; \neg\,\exists y\,\forall z\,\Big(E(y,z) \to (E(x,z) \wedge \exists x\, E(x,x))\Big)$$

(i) Berechnen Sie eine zu φ äquivalente $\mathsf{FO}[\sigma]$-Formel in Negationsnormalform.

(ii) Berechnen Sie eine zu φ äquivalente $\mathsf{FO}[\sigma]$-Formel in Pränex-Normalform.

Gehen Sie hierbei wie in Beispiel 3.71 vor. Machen Sie pro Zwischenschritt nur eine Umformung und kommentieren Sie Ihre Zwischenschritte.

(b) Beweisen Sie Satz 3.68 aus der Vorlesung, das heißt zeigen Sie:

Jede $\mathsf{FO}[\sigma]$-Formel φ ist äquivalent zu einer Formel in NNF.

(c) Nutzen Sie zur Lösung dieser Aufgabe die Methode der logischen Reduktion (ähnlich wie im Beweis von Satz 3.59).

Sei $\Sigma = \{a, b\}$ und $\sigma_\Sigma = \{\leqslant, P_a, P_b\}$ die Signatur, die aus dem 2-stelligen Relationssymbol $\leqslant$, sowie zwei 1-stelligen Relationssymbolen P_a und P_b besteht. Beweisen Sie, dass es keinen $\mathsf{FO}[\sigma_\Sigma]$-Satz gibt, der die Sprache aller Worte aus $\{a, b\}^*$ beschreibt, in denen die Anzahl der in ihnen vorkommenden as durch drei teilbar ist.

Zur Erinnerung: Ein $\mathsf{FO}[\sigma_\Sigma]$-Satz φ *beschreibt* eine Sprache $L \subseteq \Sigma^*$, falls für jedes nicht-leere Wort $w \in \Sigma^*$ gilt: $w \in L \iff \mathcal{A}_w \models \varphi$.

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 `blatt10.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 9 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) 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 anwendet 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.