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

11. Januar 2018

Logik in der Informatik

Wintersemester 2017/2018

Übungsblatt 11

Abgabe: *bis 23. Januar 2018, 11.10 Uhr (vor der Vorlesung oder im Briefkasten zwischen den Räumen 3.401 und 3.402 im Johann von Neumann-Haus (Rudower Chaussee 25))*

Aufgabe 1: (25 Punkte)

Beweisen oder widerlegen Sie die folgenden Aussagen:

(a) Sei $\sigma = \emptyset$. Es gibt einen $\mathsf{FO}[\sigma]$-Satz φ_a, so dass für jede σ-Struktur $\mathcal{A}$ gilt:

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

(b) Sei $\sigma = \{E/2, F/2\}$ eine relationale Signatur. Es gibt einen $\mathsf{FO}[\sigma]$-Satz φ_b, so dass für jede σ-Struktur $\mathcal{A} = (A, E^{\mathcal{A}}, F^{\mathcal{A}})$, bei der $\mathcal{G} := (A, E^{\mathcal{A}})$ ein endlicher gerichteter Pfad ist, gilt:

$$\mathcal{A} \models \varphi_b \quad \Longleftrightarrow \quad F^{\mathcal{A}} \text{ ist der reflexive und transitive Abschluss von } E^{\mathcal{A}}.$$

Dabei nutzen wir folgende Begriffe:

- Ein Graph $\mathcal{G} = (V^{\mathcal{G}}, E^{\mathcal{G}})$ ist ein endlicher gerichteter Pfad, falls es ein $n \in \mathbb{N}$ mit $n \geq 1$ gibt, so dass $|V^{\mathcal{G}}| = n$, $V^{\mathcal{G}} = \{v_1, \ldots, v_n\}$ und $E^{\mathcal{G}} = \{\, (v_i, v_{i+1}) \,:\, 1 \leq i < n \,\}$.
- Seien $E^{\mathcal{A}}, F^{\mathcal{A}} \subseteq A \times A$. $F^{\mathcal{A}}$ heißt transitiver und reflexiver Abschluss von $E^{\mathcal{A}}$, wenn für alle $(a, a') \in A \times A$ gilt:

$$(a, a') \in F^{\mathcal{A}} \quad \Longleftrightarrow \quad a = a' \text{ oder es gibt im gerichteten Graphen } \mathcal{G} = (A, E^{\mathcal{A}}) \text{ einen Weg vom Knoten } a \text{ zum Knoten } a'.$$

Aufgabe 2: (25 Punkte)

(a) Welche der beiden folgenden Aussagen ist für jede Signatur σ und jede $\mathsf{FO}[\sigma]$-Formel φ korrekt, welche nicht? Beweisen Sie, dass ihre Antworten korrekt sind.

(i) $\exists x \forall y \varphi \models \forall y \exists x \varphi$ **(ii)** $\forall y \exists x \varphi \models \exists x \forall y \varphi$

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

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

(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.

(c) 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.

Aufgabe 3: **(25 Punkte)**

(a) Sei $\sigma := \{E\}$ die Signatur, die aus dem 2-stelligen Relationssymbol E besteht. Betrachten Sie das Alphabet $A = A_{\mathsf{FO}[\sigma]}$ und die Menge $M := A^*$. Geben Sie einen Kalkül $\mathfrak{K}$ über der Menge M an, so dass gilt: $\mathrm{abl}_{\mathfrak{K}} = \mathsf{FO}[\sigma]$.

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

-
$$\frac{}{mi}$$
- für alle $w \in A^*$ enthält $\mathfrak{K}$ die Regel
$$\frac{mw}{mww}$$
- für alle $w, w' \in A^*$ enthält $\mathfrak{K}$ die Regeln
$$\frac{wiiiw'}{wuw'} \quad \text{und} \quad \frac{wuuw'}{ww'}$$

(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? Beweisen Sie jeweils, dass Ihre Antwort korrekt ist.

(I) $miiii$

(II) $miiuu$

(III) mu

— auf der nächsten Seite geht's weiter —

Aufgabe 4: (25 Punkte)

Lesen Sie Kapitel 7 aus dem Buch „Learn Prolog Now!“.

Achtung: *Die Bearbeitung der Aufgabe ist unter Beachtung der bekannten Abgabehinweise über Moodle abzugeben! Analog zu Blatt 7, 9 und 10 finden Sie die benötigten Dateien auf der Seite zur Prolog-Übung. Machen Sie sich auch mit den neuen Prolog-Modulen* ***`unit_propagation`*** *und* ***`pure_literal`*** *vertraut.*[1]

(a) Importieren Sie im Modul `dpll` Ihrer Abgabe `blatt11.pl` die Prädikate aus den Prolog-Modulen, die Sie für die Lösung der folgenden Teilaufgabe benötigen.

(b) Wir kodieren Klauselmengen, wie auf den Blättern 9 und 10, als Listen von Listen von Literalen. Implementieren Sie das Prädikat `dpll/1`, so dass eine Anfrage

```
?- dpll(KM).
```

für eine Klauselmenge `KM` genau dann erfolgreich ist, wenn die Klauselmenge erfüllbar ist.

Beispielsweise sollte die Anfrage für die Klauselmenge

```
KM = [[x1,~x5,~x6,x7], [~x1,x2,~x5], [~x1,~x2,~x3,~x5,~x6],
      [x1,x2,~x4,x7], [~x4,~x6,~x7], [x3,~x5,x7], [x3,~x4,~x5],
      [x5,~x6], [x5,x4,~x8], [x1,x3,x5,x6,x7], [~x7,x8],
      [~x6,~x7,~x8]]
```

erfüllt sein. Es macht hierbei nichts, wenn die Antwort `true.` durch das Backtracking mehrfach ausgegeben werden kann. Für die Klauselmenge

```
KM = [[~r,t,w], [~r,~s,~w], [~r,~t], [~q,s,t], [~q,r,~s],
      [r,s,w], [r,~t,~w], [q,u], [s,~u,~w], [q,w], [q,~s,~u]]
```

sollte die selbe Anfrage jedoch scheitern.

Hinweise: Implementieren Sie dazu den *DPLL-Algorithmus*, wie er auf Seite 89 des Skripts beschrieben ist. Definieren Sie geeignete Hilfsprädikate. Nutzen Sie insbesondere die bereits auf Blatt 9 und 10 implementierten Vereinfachungsheuristiken *Unit Propagation* und *Pure Literal Rule*, die Sie aus den Modulen des entsprechenden Namens importieren können. Die Streichung von Klauseln, die Obermengen von anderen Klauseln sind, müssen Sie nicht implementieren.

(c) Gegeben sei die kontextfreie Grammatik $G = (\Sigma, V, S, P)$ mit den Terminalsymbolen $\Sigma := \{\texttt{if}, \texttt{then}, \texttt{else}, \texttt{e1}, \texttt{e2}, \texttt{s1}, \texttt{s2}\}$, den Nichtterminalsymbolen $V := \{\texttt{stmt}, \texttt{expr}\}$, dem Startsymbol $S := \texttt{stmt}$, und den Produktionen $P :=$

$$\{\ \texttt{stmt} \longrightarrow \texttt{if expr then stmt},\quad \texttt{stmt} \longrightarrow \texttt{if expr then stmt else stmt},$$
$$\texttt{stmt} \longrightarrow \texttt{s1},\quad \texttt{stmt} \longrightarrow \texttt{s2},\quad \texttt{expr} \longrightarrow \texttt{e1},\quad \texttt{expr} \longrightarrow \texttt{e2}\ \}$$

Bilden Sie für die kontextfreie Grammatik G eine *Definite Clause Grammar (DCG)*, so dass die Anfrage

```
?- stmt(X, []).
```

genau dann erfüllt wird, wenn `X` eine Liste von Terminalsymbolen aus Σ ist, die einem Wort der durch G beschriebenen Sprache entspricht. Dies gilt beispielsweise für die Liste

```
X = [ if, e1, then, if, e2, then, s1, else, s2] .
```

Fügen Sie Ihre Definite Clause Grammar der Datei `blatt11.pl` hinzu.

[1] Verfügbar ab 16.01.18.