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

12. Januar 2017

Logik in der Informatik

Wintersemester 2016/17

Übungsblatt 11

Abgabe: *24. Januar 2017*

Aufgabe 1: (24 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: (26 Punkte)

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

Die *Hundesprache* H ist eine Menge von Worten über dem Alphabet $\Sigma := \{\mathrm{W}, \mathrm{A}, \mathrm{U}\}$, die durch die folgenden Regeln rekursiv definiert ist:

Basisregel: (B) $\mathrm{WA} \in H$.

Rekursive Regeln: Für alle $v \in \Sigma^*$ und alle $w \in \Sigma^*$ gilt:

(R1) Ist $v \in H$, so ist auch $vv \in H$.

(R2) Ist $v\mathrm{A}w \in H$, so ist auch $v\mathrm{AU}w \in H$.

(R3) Ist $v\mathrm{UU}w \in H$, so ist auch $vAAAw \in H$.

(R4) Ist $v\mathrm{AAA}w \in H$, so ist auch $vw \in H$.

(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 $\mathrm{WAAA} \in H$?

(d) Geben Sie einen Kalkül $\mathfrak{K}$ über der Menge Σ^* an, so dass gilt: $\mathrm{abl}_{\mathfrak{K}} = H$.

Aufgabe 3: (25 Punkte)

Sei σ eine Signatur und seien $\varphi, \psi \in \mathsf{FO}[\sigma]$.

(a) Leiten Sie ähnlich wie in Beispiel 4.19 aus dem Skript die folgende Sequenz im Sequenzenkalkül $\mathfrak{K}_S$ ab.

$$\varphi,\ (\neg\varphi \vee \psi)\ \vdash\ \psi$$

(b) Sei $\Gamma \subseteq_e \mathsf{FO}[\sigma]$.

Beweisen Sie die Korrektheit der $\wedge$-Einführung im Sukzedens ($\wedge$S):

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

— *auf der nächsten Seite geht's weiter* —

Aufgabe 4: (25 Punkte)

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

Achtung: *Geben Sie Ihre Lösungsansätze für die Teilaufgaben* **(a)**-**(d)** *in einer Datei mit dem Namen* `dpll.pl` *über Moodle ab!* ***Es gilt:*** *Lösungsansätze, die von SWI-Prolog nicht geladen werden können, werden nicht bewertet!*

(a) Machen Sie sich mit den Prolog-Modulen vertraut, die Sie unter der URL `http://www2.informatik.hu-berlin.de/logik/lehre/WS16-17/Logik/downloads/al/` finden können. Laden Sie diese Prolog-Module in ein Verzeichnis Ihrer Wahl. Erstellen Sie (im selben Verzeichnis) in einer Datei `dpll.pl` ein Modul mit dem Namen `dpll`, welches das Prädikat `dpll/1` exportiert.

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

(c) 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.

(d) 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 `dpll.pl` hinzu.

(e) Untersuchen Sie mit der Anfrage

```
?- listing.
```

die interne Darstellung Ihrer DCG aus Teilaufgabe **(d)** in SWI-Prolog. Erklären Sie die Ausgabe von SWI-Prolog bei der Anfrage

```
?- stmt([ if, e1, then, if, e2, then, s1, else, s2], []).
```