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

23. Januar 2020

Logik in der Informatik

Wintersemester 2019/2020

Übungsblatt 13

Abgabe: *bis 4. Februar 2020, 11.15 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: (24 Punkte)

Sei $\sigma := \{R, f_0, f_1, c\}$, wobei c ein Konstantensymbol, R ein 2-stelliges Relationssymbol und f_0, f_1 zwei 1-stellige Funktionssymbole sind.

Beweisen Sie folgende Aussagen aus Korollar 4.41 aus dem Vorlesungsskript:

(a) Das Unerfüllbarkeitsproblem für $\mathsf{FO}[\sigma]$ ist nicht entscheidbar.

(b) Das Erfüllbarkeitsproblem für $\mathsf{FO}[\sigma]$ ist nicht semi-entscheidbar.

(c) Das Folgerungsproblem für $\mathsf{FO}[\sigma]$ ist nicht entscheidbar.

Aufgabe 2: (26 Punkte)

(a) Sei R ein 2-stelliges Relationssymbol, f ein 1-stelliges Funktionssymbol und seien c und d Konstantensymbole.

Im Folgenden ist für jedes $i \in \{1, 2\}$ eine Signatur σ_i und ein $\mathsf{FO}[\sigma_i]$-Satz φ_i gegeben.

(1) Sei $\sigma_1 := \{R, f, c\}$ und sei φ_1 der folgende $\mathsf{FO}[\sigma_1]$-Satz:

$$\forall x \forall y \forall z \left(R(x, f(x)) \ \wedge \ \Big(\big(R(x,y) \wedge R(y,z) \big) \ \rightarrow \ R(x,z) \Big) \right)$$

(2) Sei $\sigma_2 := \{R, c, d\}$ und sei φ_2 der folgende $\mathsf{FO}[\sigma_2]$-Satz:

$$\exists x \, R(c,x) \ \wedge \ \forall x \forall y \Big(\neg x{=}y \ \rightarrow R(x,y) \Big)$$

Geben Sie für jedes $i \in \{1, 2\}$ eine σ_i-Herbrandstruktur $\mathcal{A}_i$ und eine σ_i-Herbrandstruktur $\mathcal{B}_i$ an, so dass gilt:

$$\mathcal{A}_i \models \varphi_i \qquad \text{und} \qquad \mathcal{B}_i \not\models \varphi_i.$$

Begründen Sie jeweils, warum $\mathcal{A}_i \models \varphi_i$ bzw. $\mathcal{B}_i \not\models \varphi_i$ gilt.

(b) Sei $\sigma := \{f, c\}$, wobei f ein 1-stelliges Funktionssymbol ist und c ein Konstantensymbol.
Zeigen Sie, dass Satz 4.46 aus dem Vorlesungsskript im Allgemeinen *nicht* für $\mathsf{FO}[\sigma]$-Sätze in Skolemform gilt, die *nicht* gleichheitsfrei sind.
Geben Sie dazu einen $\mathsf{FO}[\sigma]$-Satz φ in Skolemform an, so dass gilt:

$$\varphi \text{ ist erfüllbar,} \qquad \text{aber} \qquad \varphi \text{ besitzt kein Herbrand-Modell.}$$

Aufgabe 3: **(25 Punkte)**

Sei $\sigma := \{R, f\}$, wobei R ein 2-stelliges Relationssymbol und f ein 1-stelliges Funktionssymbol ist. Transformieren Sie die $\mathsf{FO}[\sigma]$-Formel

$$\varphi(z) \;:=\; \exists x \,\Big(\forall y \; R(x,y) \;\vee\; f(x) = z\Big)$$

in einen zu φ erfüllbarkeitsäquivalenten gleichheitsfreien $\mathsf{FO}[\hat{\sigma}]$-Satz $\hat{\varphi}$ in Skolemform.

Gehen Sie dabei vor wie im Beweis von Satz 4.52 im Vorlesungsskript. Geben Sie insbesondere auch die Formel an, die nach jedem der Schritte 1, 2 und 3 des Beweises entsteht.

— auf der nächsten Seite geht's weiter —

Aufgabe 4: (25 Punkte)

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. Machen Sie sich auch mit den neuen Prolog-Modulen* ***`unit_propagation`*** *und* ***`pure_literal`*** *vertraut.*[1]

(a) Importieren Sie im Modul `dpll` Ihrer Abgabe `blatt13.pl` die Prädikate aus den Prolog-Modulen, die Sie für die Lösung der folgenden Teilaufgabe benötigen.

(b) Wir kodieren Klauselmengen, wie gewohnt, 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 94 des Skripts beschrieben ist. Definieren Sie geeignete Hilfsprädikate. Nutzen Sie insbesondere die bereits auf Blatt 11 und 12 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) Implementieren Sie ein Prädikat `sat/1`, so dass eine Anfrage

```
?- sat(F).
```

für eine aussagenlogische Formel `F` genau dann erfolgreich ist, wenn `F` erfüllbar ist.

Hinweise: Ihr Prädikat soll zu der Formel `F` zuerst eine *erfüllbarkeitsäquivalente* 3-*KNF* konstruieren, und anschließend deren Erfüllbarkeit mit dem DPLL-Algorithmus testen. Es macht nichts, wenn Ihr Prädikat für eine erfüllbare Formel mehrfach

```
true.
```

ausgibt.

Beispielsweise, sollte Prolog auf die Anfrage

```
?- sat((p => ~q) \/ (~ (p /\ q) /\ r), L).
```

mit `true.` antworten.

[1] Verfügbar ab 28.01.20.