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

25. Januar 2018

Logik in der Informatik

Wintersemester 2017/2018

Übungsblatt 13

Abgabe: *bis 6. Februar 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: (30 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:

(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: (25 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: (20 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

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

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.

Aufgabe 4: (25 Punkte)

Lesen Sie Kapitel 8 aus dem Buch „Learn Prolog Now!“.

Achtung: *Geben Sie Ihre Lösungsansätze in einer Datei mit dem Namen* `blatt13.pl` *über Moodle ab!* ***Es gilt:*** *Lösungsansätze, die von SWI-Prolog nicht geladen werden können, werden nicht bewertet!*

(a) 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.

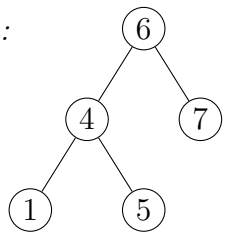
Sie können zur Lösung dieser Aufgabe alle Prolog-Module verwenden, die Sie unter `https://www2.informatik.hu-berlin.de/logik/lehre/WS17-18/Logik/prolog-uebung.shtml` vorfinden. Dies gilt insbesondere für die Module `tseitin.pl` und `dpll.pl`.

In den folgenden Teilaufgaben betrachten wir *geordnete und beschriftete Binärbäume*, die folgende Eigenschaften erfüllen:

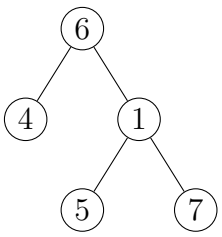
- Jeder Binärbaum enthält mindestens einen Knoten.
- Jeder Knoten in einem Binärbaum ist entweder ein *Blattknoten*, d.h. er besitzt keine Kinderknoten, oder er ist ein *innerer Knoten*, d.h. er besitzt *sowohl einen linken als auch einen rechten Teilbaum*, die beide nicht leer sind.
- Jeder Knoten in einem Binärbaum ist mit einem Prolog-Term beschriftet.

Betrachten Sie beispielsweise folgende Binärbäume:

Binärbaum 1:



Binärbaum 2:



Wir repräsentieren Binärbäume wie folgt durch Prolog-Terme: Ist `T` ein Prolog-Term, dann entspricht `node(T, nil, nil)` dem Binärbaum, der aus genau einem, mit `T` beschrifteten, Blattknoten besteht. Repräsentieren `L` und `R` Binärbäume, dann ist `node(T, L, R)` der Binärbaum, der aus einem mit `T` beschrifteten inneren Knoten besteht, an den als linker Teilbaum `L` und als rechter Teilbaum `R` angehängt ist. Beispielsweise wird *Binärbaum 1* repräsentiert durch den Term

```
node(6,
     node(4,
          node(1, nil, nil),
          node(5, nil, nil)),
     node(7, nil, nil))
```

(b) Eine Liste `X` nennen wir die *pre-order-Traversierung* eines gegebenen Binärbaums `B`, wenn das folgende Prädikat `preorder/2` die Anfrage

```
?- preorder(B, X).
```

erfüllt:

```
preorder(node(T, nil, nil), [T]) :- !.
preorder(node(T, L, R), [T|X]) :-
    preorder(L, LX), preorder(R, RX), append(LX, RX, X).
```

Obwohl jeder Binärbaum eine eindeutige pre-order-Traversierung besitzt, gibt es umgekehrt Listen, die pre-order-Traversierungen von keinem oder auch von mehr als einem Binärbaum sind. Beispielsweise ist die Liste `[6, 4, 1, 5, 7]` sowohl für *Binärbaum 1* als auch für *Binärbaum 2* eine pre-order-Traversierung.

Schreiben Sie eine *Definite Clause Grammar mit einem zusätzlichen Argument*, so dass die Anfrage

```
?- bt(B, X, []).
```

für einen Binärbaum `B` und eine Liste `X` genau dann erfüllt ist, wenn `X` eine pre-order-Traversierung von `B` ist.

(c) Wir nennen einen Binärbaum einen *binären Suchbaum*, wenn seine Knoten nur mit Ganzzahlen beschriftet sind und für jeden seiner inneren Knoten `node(Z, L, R)` gilt: Jeder Knoten in `L` ist mit einer Zahl $< $ `Z`, und jeder Knoten in `R` mit einer Zahl $>$ `Z` beschriftet. Beispielsweise ist *Binärbaum 1* ein binärer Suchbaum, *Binärbaum 2* jedoch nicht.

Schreiben Sie eine *Definite Clause Grammar mit drei zusätzlichen Argumenten*, so dass die Anfrage

```
?- bst(B, Min, Max, X, []).
```

genau dann erfüllt ist, wenn `X` die pre-order-Traversierung für den *binären Suchbaum* `B` ist, und zusätzlich `Min` und `Max` die kleinste, beziehungsweise größte Ganzzahl ist, die als Beschriftung in `B` vorkommt.

Hinweis: Nutzen Sie die Möglichkeit, Ihrer Definite Clause Grammar mit Hilfe der Notation {...} zusätzliche Ziele hinzuzufügen. Verwenden Sie gegebenfalls das eingebaute Prädikat `integer/1`.