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

19. Dezember 2019

Logik in der Informatik

Wintersemester 2019/2020

Übungsblatt 10

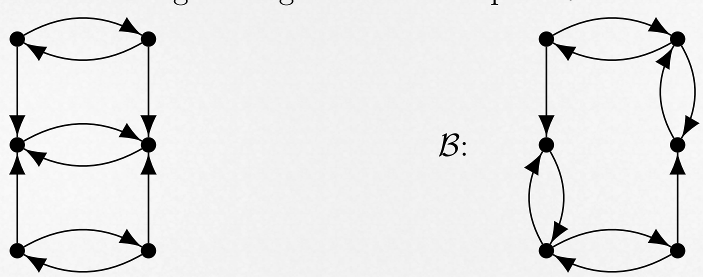
Abgabe: *bis 14. Januar 2019, 11.15 Uhr (vor der Vorlesung oder im Briefkasten zwischen den Räumen 3.401 und 3.402 im Johann von Neumann-Haus (Rudower Chaussee 25))*

Achtung: Wir möchten Sie daran erinnern, dass das Vorrechnen in den Übungsgruppen Voraussetzung für die Prüfungszulassung ist.

Aufgabe 1: (30 Punkte)

Sei $\sigma := \{E/2\}$. Betrachten Sie die folgenden gerichteten Graphen $\mathcal{A}$ und $\mathcal{B}$:

$\mathcal{A}$: $\mathcal{B}$:



Für

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

gilt $\mathcal{A} \models \varphi$ und $\mathcal{B} \not\models \varphi$.

(a) Leiten Sie aus dem $\mathsf{FO}[\sigma]$-Satz φ eine Gewinnstrategie für Spoiler im EF-Spiel auf $\mathcal{A}$ und $\mathcal{B}$ her. Geben Sie an, wie viele Runden Spoiler benötigt, wenn er dieser Strategie folgt. Beschreiben Sie die Strategie ähnlich wie in der in der Vorlesung behandelten Beweisidee zu Satz 3.52 im Vorlesungsskript.

(b) Existiert eine bessere Gewinnstrategie für Spoiler? D.h. eine Strategie, mit der er in weniger Runden das Spiel gewinnt? Wenn ja, dann beschreiben Sie eine solche Strategie. Wenn nein, dann begründen Sie dieses.

Aufgabe 2: **(20 Punkte)**

Zeigen Sie, dass das Lemma 3.57 aus dem Skript im folgenden Sinne optimal ist:

Seien $\mathcal{A}$ und $\mathcal{B}$ endliche lineare Ordnungen, sei $k := 2$, und sei $\bar{a} := a_1, a_2$ und $\bar{b} := b_1, b_2$, wobei a_1, b_1 die kleinsten und a_2, b_2 die größten Elemente in A und B bezüglich $\leq^{\mathcal{A}}$ und $\leq^{\mathcal{B}}$ sind.

Falls $|A| < |B|$ und $|A| \leq 2^m$, so hat *Spoiler* eine Gewinnstrategie im m-Runden EF-Spiel auf $\mathcal{A}$ und $\mathcal{B}$.

Aufgabe 3: **(25 Punkte)**

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

(a) 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 größer ist als die Anzahl der in ihnen vorkommenden bs.

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: $w \in L \iff \mathcal{A}_w \models \varphi$.

(b) Sei 2-*COL* die Klasse aller gerichteten zweifärbbaren Graphen, d.h. aller $\{E/2\}$-Strukturen $\mathcal{A} = (A, E^{\mathcal{A}})$ für die gilt:

> Es gibt eine Funktion $f : A \to \{rot, blau\}$, so dass für jede Kante (a, b) in $E^{\mathcal{A}}$ gilt: $f(a) \neq f(b)$.

Zeigen Sie: Die Klasse 2-*COL* ist *nicht FO-definierbar*.

Aufgabe 4: **(25 Punkte)**

Lesen Sie Kapitel 10 aus dem Buch „Learn Prolog Now!“.
Achtung: *Die Bearbeitung der Teilaufgaben* **(b)**, **(c)** *und* **(d)** *ist unter Beachtung der bekannten Abgabehinweise über Moodle abzugeben! Importieren Sie in Ihrer Abgabedatei* ***`blatt10.pl`****, wie in Blatt 9, die Datei* ***`al.pl`****.*

(a) Gegeben sei das folgende Prolog-Programm:

```
a(X, Y) :- b(X, Y).
a(1, 1).
b(X, X) :- c(X).
b(X, Y) :- c(X), !, c(Y).
b(X, Y) :- c(X), c(Y).
c(2).
c(3).
```

Zeichnen Sie einen Suchbaum für die folgende Anfrage: `?- a(X, Y).`

(b) Schreiben Sie in `blatt10.pl` ein Prädikat `not_member/2`, so dass `not_member(X, L)` für einen Term `X` und eine Liste `L` genau dann erfüllt ist, wenn `X` mit *keinem* Element von `L` unifiziert werden kann. Verwenden Sie dabei abgesehen vom Cut und dem in SWI-Prolog vordefinierten Prädikat `fail/0` keine weiteren Prädikate, und insbesondere nicht `\=/2`.

(c) Führen Sie in `blatt10.pl` einen neuen Operator `<=>` für die Biimplikation $\leftrightarrow$ ein, der den gleichen Typ und die gleiche Präzedenz wie der in `al.pl` definierte Operator `=>` hat.

(d) Implementieren Sie in `blatt10.pl`, analog zu Beispiel 2.54 im Vorlesungsskript, Schritt 1 des Tseitin-Verfahrens. D.h., schreiben Sie ein Prädikat `tseitin/2`, so dass die Anfrage `tseitin(F, L)` für eine aussagenlogische Formel `F` eine Liste `L` aussagenlogischer Formeln ausgibt, die die folgenden Eigenschaften hat:

- Die Konjunktion der Formeln in der Liste `L` ist erfüllbarkeitsäquivalent zu `F`.
- Die Liste `L` enthält für jede Teilformel von `F` (abgesehen von Literalen) genau eine Formel.
- In jeder Formel aus `L` kommen höchstens 3 verschiedene Aussagensymbole vor.

Beispielsweise sollte Prolog auf die Anfrage:

```
tseitin((p => ~q) \/ (~ (p /\ q) /\ r), L).
```

wie folgt antworten:

```
L = [x1, x1<=>x2\/x3, x2<=> (p=> ~q), x3<=>x4/\r, x4<=> ~x5, x5<=>p/\q].
```

Hierbei sind die konkrete Wahl der neuen Aussagensymbole sowie die Reihenfolge der Formeln in der Repräsentation der Menge unwesentlich.

Hinweise:

- Benutzen Sie zur Erzeugung neuer Aussagensymbole das in SWI-Prolog eingebaute Prädikat `gensym/2`. Das Prädikat `gensym/2` instantiiert bei dem Aufruf `gensym(x, A)` die Variable `A` mit einem Atom der Form xn, wobei eine Zahl n so gewählt wird, dass das Atom xn in diesem Lauf von SWI-Prolog noch nicht verwendet wurde.
- Benutzen Sie den in Teilaufgabe (c) definierten Operator `<=>`.
- Nutzen Sie ggf. Cut oder Negation. Führen Sie bei Bedarf Hilfsprädikate ein.