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

5. Januar 2017

Logik in der Informatik

Wintersemester 2016/17

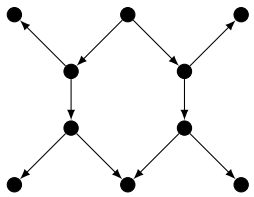
Übungsblatt 10

Abgabe: *17. Januar 2017*

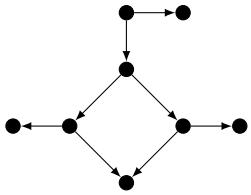
Aufgabe 1: (25 Punkte)

Betrachten Sie die folgenden Graphen $\mathcal{A}$ und $\mathcal{B}$:

$\mathcal{A}$:



$\mathcal{B}$:



(a) Welches ist das kleinste m, so dass Spoiler eine Gewinnstrategie im m-Runden Ehrenfeucht-Fraïssé Spiel auf $\mathcal{A}$ und $\mathcal{B}$ hat? Begründen Sie Ihre Antwort, indem Sie eine Gewinnstrategie für Spoiler im m-Runden Ehrenfeucht-Fraïssé Spiel und eine Gewinnstrategie für Duplicator im $(m-1)$-Runden Ehrenfeucht-Fraïssé Spiel beschreiben.

(b) Finden Sie für Ihre Antwort m aus Teil **(a)** einen $\mathsf{FO}[\sigma]$-Satz ψ der Quantorentiefe m, so dass $\mathcal{A} \models \psi$ und $\mathcal{B} \models \neg\psi$.

Aufgabe 2: (26 Punkte)

(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 gerade ist.

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 $f : A \to \{\textit{rot}, \textit{blau}\}$, 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 3: **(24 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

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

(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.

— auf der nächsten Seite geht's weiter —

Aufgabe 4: (25 Punkte)

Lesen Sie Kapitel 12 aus dem Buch „Learn Prolog Now!“.

Achtung: *Geben Sie Ihre Lösungsansätze für die Teilaufgaben* **(b)**–**(e)** *in einer Datei mit dem Namen* `pure_literal.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 `al_def`, `al_literals` und `al_nf` 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.

(b) Erstellen Sie (im selben Verzeichnis) in einer Datei `pure_literal.pl` ein Modul mit dem Namen `pure_literal`, das die Prädikate `knf_shell/0` und `pure_literal/2` exportiert.

(c) Importieren Sie im Modul `pure_literal` genau die Prädikate aus den Modulen `al_def`, `al_literals` und `al_nf`, die Sie zum lösen der folgenden beiden Teilaufgaben benötigen.

(d) Wir kodieren Klauselmengen wie auf Blatt 9 als Listen von Listen von Literalen.

Implementieren Sie das Prädikat `knf_shell/0`, so dass eine Anfrage

```
?- knf_shell.
```

eine Eingabeaufforderung zur Konstruktion von Klauselmengen aus aussagenlogischen Formeln startet. D.h., wenn über die Tastatur eine aussagenlogische Formel als Prolog-Term eingegeben wird, dann soll nach Ende der Eingabe (durch `.` und die Taste „Enter“) eine zu der Formel äquivalente Klauselmenge ausgegeben werden. Dies soll so lange wiederholt werden, bis statt einer aussagenlogischen Formel das Atom `bye` (wieder gefolgt durch `.` und die Taste „Enter“) eingegeben wird.

Hinweise: Definieren Sie sich gegebenenfalls geeignete Hilfsprädikate. Verwenden Sie für die Eingabe das Prädikat `read/1` und für die Ausgabe das Prädikat `write/1`. Beide Prädikate sind in SWI-Prolog vordefiniert. Sie müssen sich nicht um die Behandlung von Eingabefehlern kümmern.

(e) Implementieren Sie ein Prädikat `pure_literal/2`, so dass eine Anfrage von der Form

```
?- pure_literal(KM, KM2).
```

auf die Klauselmenge `KM` die *Pure Literal Rule* des DPLL-Algorithmus anwendet und die entstehende Klauselmenge in `KM2` zurückgibt. Beispielsweise sollte die Anfrage

```
?- pure_literal([[~x1, x2, ~x5], [x1, x2, ~x4, x7], [x3, ~x5, x7],
                 [x3, ~x4, ~x5], [x5,x4,~x8], [x1,x3,x5,x7],
                 [~x7,x8]], KM2).
```

zu der Antwort

```
KM2 = [].
```

führen.

Hinweise: Definieren Sie geeignete Hilfsprädikate. *Beispielsweise* bietet es sich an, Prädikate `is_literal/2` und `is_pure_literal/2` einzuführen, so dass das Ziel `is_literal(L, KM)` für jedes in der Klauselmenge `KM` vorkommende Literal `L` erfüllt ist, und so dass das Ziel `is_pure_literal(L, KM)` für jedes in der Klauselmenge `KM` vorkommende Literal `L` erfüllt ist, dessen Negat nicht in `KM` vorkommt.