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

15. Dezember 2015

Logik in der Informatik

Wintersemester 2015/2016

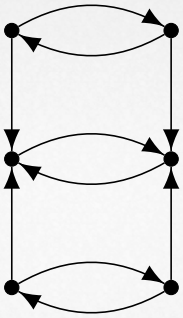
Übungsblatt 9

Abgabe: *bis 7. Januar, 13.¹⁵ Uhr (vor der Vorlesung oder im Briefkasten zwischen den Räumen 3.401 und 3.402 im Johann von Neumann-Haus (Rudower Chaussee 25))*

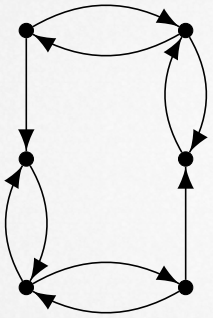
Aufgabe 1: (25 Punkte)

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

$\mathcal{A}$:



$\mathcal{B}$:



Für

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

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: (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 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 $f : A \to \{\mathit{rot}, \mathit{blau}\}$, 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

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

(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.

Aufgabe 4 finden Sie auf der Rückseite

Aufgabe 4: (25 Punkte)

Lesen Sie Kapitel 11 aus dem Buch „Learn Prolog Now!“.

Achtung: *Geben Sie Ihre Lösungsansätze in einer Datei* `blatt09.pl` *über das GOYA-System ab!* ***Außerdem gilt:*** *Lösungsansätze, die von SWI-Prolog nicht geladen werden können, werden nicht bewertet!*

(a) Das Ziel dieser Aufgabe ist es, eine Verwaltung für Übungspunkte in Prolog zu implementieren. Wir repräsentieren Übungsteilnehmer/innen durch ihren Namen als Zeichenkette. Die Information, dass `S` auf Übungsblatt Nr. `N` genau `P` Punkte erhält, repräsentieren wir durch einen Fakt `punkte(S, N, P)` in der Wissensbasis. Damit das durch solche Fakten definierte Prädikat `punkte/3` während der Laufzeit von SWI-Prolog verändert werden kann, müssen Sie die Datei `blatt09.pl` mit der Zeile

```
:- dynamic punkte/3.
```

beginnen.

Schreiben Sie ein Prädikat `bepunkten/3`, so dass eine Anfrage `?- bepunkten(S, N, P).` die folgende Wirkung hat:

(1) Wurde Übungsblatt Nr. `N` für den/die Übungsteilnehmer/in `S` bereits bepunktet, d.h. also, existiert ein Fakt `punkte(S, N, P2)` mit irgendeiner Punktzahl `P2` in der Wissensbasis, dann wird dieser Fakt aus der Wissensbasis gelöscht.

(2) Der Fakt `punkte(S, N, P)` wird der Wissensbasis *am Ende* hinzugefügt.

Beispiel: Die Anfrage `?- bepunkten("Mara", 2, 0).` fügt der Wissensbasis den Fakt `punkte("Mara", 2, 0)` hinzu. Eine folgende Anfrage `?- bepunkten("Mara", 2, 99).` löscht den Fakt `punkte("Mara", 2, 0)` aus der Wissensbasis und ergänzt die Wissensbasis anschließend durch den Fakt `punkte("Mara", 2, 99)`.

(b) Schreiben Sie ein Prädikat `bepunktet/2`, so dass die Anfrage `?- bepunktet(N, L).` für eine Zahl `N` eine Liste der Namen aller Übungsteilnehmer/innen zurückgibt, deren Übungsblatt Nr. `N` der Wissensbasis zufolge bereits bepunktet wurde.

(c) Laden Sie die Datei `al_v2.pl` von der URL `http://www2.informatik.hu-berlin.de/logik/lehre/WS15-16/Logik/downloads/al_v2.pl` in *das selbe Verzeichnis* wie die Datei `blatt09.pl`. Setzen Sie an den Anfang der Datei `blatt09.pl` die Zeile

```
:- ensure_loaded(['al_v2']).
```

Wir repräsentieren im Folgenden Klauseln als Listen von Literalen und Klauselmengen als Listen von Klauseln. Beispielsweise repräsentiert der Prolog-Term

```
[[~x1, ~x2, ~x3, x4], [x1, ~x2], [x2]]
```

die Klauselmenge $\{\{\neg X_1, \neg X_2, \neg X_3, X_4\}, \{X_1, \neg X_2\}, \{X_2\}\}$.

Schreiben Sie ein Prädikat `unit_propagation/2`, das die Vereinfachungsheuristik *Unit Propagation* des DPLL-Algorithmus implementiert. D.h., ist `K` eine Klauselmenge, dann sollte die Anfrage `?- unit_propagation(K, K2).` in `K2` die Klauselmenge zurückgeben, die aus `K` entsteht, indem die Unit Propagation so lange auf `K` angewendet wird, bis keine „Einerklauseln“ mehr vorhanden sind. Beispielsweise sollte die Anfrage

```
?- unit_propagation([[~x1, ~x2, ~x3, x4], [x1, ~x2], [x2]], K2).
```

zu der Antwort

```
?- K2 = [[~x3, x4]].
```

oder einer äquivalenten Antwort führen. *Hinweis:* Führen Sie geeignete Hilfsprädikate ein!