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

9. Januar 2020

Logik in der Informatik

Wintersemester 2019/2020

Übungsblatt 11

Abgabe: *bis 21. Januar 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: (25 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

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

(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 2: (20 Punkte)

Beweisen oder widerlegen Sie die folgenden Aussagen:

(a) Sei $\sigma = \emptyset$. Es gibt einen $\mathsf{FO}[\sigma]$-Satz φ_a, so dass für jede σ-Struktur $\mathcal{A}$ gilt:

$$\mathcal{A} \models \varphi_a \quad \Longleftrightarrow \quad |A| \text{ ist eine Primzahl.}$$

(b) Sei $\sigma = \{E/2, F/2\}$ eine relationale Signatur. Es gibt einen $\mathsf{FO}[\sigma]$-Satz φ_b, so dass für jede σ-Struktur $\mathcal{A} = (A, E^{\mathcal{A}}, F^{\mathcal{A}})$, bei der $\mathcal{G} := (A, E^{\mathcal{A}})$ ein endlicher gerichteter Pfad ist, gilt:

$$\mathcal{A} \models \varphi_b \quad \Longleftrightarrow \quad F^{\mathcal{A}} \text{ ist der reflexive und transitive Abschluss von } E^{\mathcal{A}}.$$

Dabei nutzen wir folgende Begriffe:

- Ein Graph $\mathcal{G} = (V^{\mathcal{G}}, E^{\mathcal{G}})$ ist ein endlicher gerichteter Pfad, falls es ein $n \in \mathbb{N}$ mit $n \geq 1$ gibt, so dass $|V^{\mathcal{G}}| = n$, $V^{\mathcal{G}} = \{v_1, \ldots, v_n\}$ und $E^{\mathcal{G}} = \{\, (v_i, v_{i+1}) \,:\, 1 \leq i < n \,\}$.
- Seien $E^{\mathcal{A}}, F^{\mathcal{A}} \subseteq A \times A$. $F^{\mathcal{A}}$ heißt transitiver und reflexiver Abschluss von $E^{\mathcal{A}}$, wenn für alle $(a, a') \in A \times A$ gilt:

$$(a,a') \in F^{\mathcal{A}} \quad \Longleftrightarrow \quad a = a' \text{ oder es gibt im gerichteten Graphen } \mathcal{G} = (A, E^{\mathcal{A}}) \text{ einen Weg vom Knoten } a \text{ zum Knoten } a'.$$

Aufgabe 3: **(30 Punkte)**

Bernard's lustige Burger-Braterei™ ist zu einem weltweit operierenden Unternehmen gewachsen. Neuestes Produkt ist ein *Online-Burger-Konfigurator*™, mit dessen Hilfe der Kunde sich seinen persönlichen Lieblings-Burger zusammenstellen kann.

Schnell stellt sich jedoch heraus, dass gewisse Beschränkungen eingeführt werden müssen, um die Kunden vor ihrer eigenen Phantasie zu schützen. Kern des *Online-Burger-Konfigurators*™ ist eine rekursiv definierte Menge B aller bestellbaren Burger. B ist über dem Alphabet

$$A \;:=\; \{\, R,\; S,\; K,\; [\,,\;)\,\}$$

definiert, wobei R einen *Rindfleisch-Patty*, S eine Portion *Salat*, K eine *Käsescheibe*, sowie die Zeichen [und) jeweils eine *untere* und eine *obere Brötchenhälfte* repräsentieren. Die Regeln für die Menge B lauten wie folgt:

Basisregel: (B) $[\,) \in B$.

Rekursive Regeln: Für alle $u, v \in A^*$ gilt:

(R1) Ist $[uv) \in B$, so ist auch $[uRv) \in B$.

(R2) Ist $uRv \in B$, so ist auch $uRKv \in B$.

(R3) Ist $ua) \in B$ und $a \in \{R, K\}$, so ist auch $uaS) \in B$.

(R4) Ist $[u) \in B$ und ist $[v) \in B$, so ist auch $[u[v) \in B$.

(a) Geben Sie für jedes der folgenden Worte aus A^* an, ob es zu der Menge B gehört oder nicht. Begründen Sie Ihre Antwort.

(i) $[\,[\,)\,)$

(ii) $[RKKSSR)$

(iii) $[RR[RKS[RS)$

(b) Im Folgenden bezeichnen wir für alle Wörter $v, w \in A^*$ mit $|w|_v$ die Anzahl der Vorkommen von v als Teilwort in w.

Beispiel: Für das Wort $w = [RKKSSR)$ ist $|w|_R = |w|_S = 2$, $|w|_{R)} = 1$ und $|w|_{K)} = 0$.

Beweisen Sie mit einer Induktion über den Aufbau der Menge B, dass

$$|w|_R \quad \geq \quad |w|_S \;+\; |w|_{R)} \;+\; |w|_{K)}$$

für jedes $w \in B$. Daraus folgt sofort, dass jeder in *Bernard's lustiger Burger-Braterei*™ bestellbare Burger mindestens so viel Rindfleisch wie Salat enthält.

(c) Geben Sie einen Kalkül $\mathfrak{K}$ über der Menge A^* an, so dass gilt: $\mathrm{abl}_{\mathfrak{K}} = B$.

— auf der nächsten Seite geht's weiter —

Aufgabe 4: (25 Punkte)

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

Achtung: *Die Bearbeitung der Aufgabe ist unter Beachtung der bekannten Abgabehinweise über Moodle abzugeben! Importieren Sie in Ihrer Abgabedatei* `blatt11.pl`*, analog zu Blatt 9, die Dateien* `al.pl` *und* `kinodb.pl`*.*

(a) Schreiben Sie ein Prädikat `george/1`, so dass die Anfrage

```
?- george(R).
```

in der Liste `R` die Menge aller Tupel `(K, Z)` zurückgibt, so dass gilt: Zum Zeitpunkt `Z` läuft im Kino `K` ein Film, in dem George Clooney Regie geführt hat.

(b) Wir implementieren eine Reservierungsverwaltung für das Kino Babylon (in dem zur Zeit aus technischen Gründen nur ein Saal in Betrieb ist). Der Umstand, dass eine Person `P` für den Film, der um `Z` Uhr beginnt, Sitzplatz `S` reserviert hat, soll durch einen Fakt `belegt(P, Z, S)` in der Wissensbasis ausgedrückt werden. Stellen Sie zu diesem Zweck Ihrer Datei `blatt11.pl` die Zeile

```
:- dynamic belegt/3.
```

voran. Schreiben Sie ein Prädikat `reservieren/3`, so dass die Anfrage

```
?- reservieren(P, Z, S).
```

den Sitzplatz `S` für Person `P` und `Z` Uhr reserviert, d.h., der Wissensbasis einen Fakt `belegt(P, Z, S)` hinzufügt. Dies soll allerdings nur möglich sein, wenn um `Z` Uhr im Babylon tatsächlich ein Film läuft, und wenn der Sitzplatz für diese Uhrzeit noch nicht belegt ist. Anderenfalls soll die Anfrage scheitern.

(c) Schreiben Sie ein Prädikat `stornieren/2`, so dass die Anfrage

```
?- stornieren(Z, S).
```

die Reservierung für den Sitzplatz `S` zur Zeit `Z` aufhebt, d.h., einen entsprechenden Fakt in der Wissensbasis löscht. Gibt es eine solche Reservierung nicht, so soll die Anfrage scheitern.

(d) Wir repräsentieren im Folgenden Klauseln als Listen von Literalen und Klauselmengen als Listen von Klauseln. So repräsentiert `[[~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 „Einerklausel“ mehr vorhanden ist. Beispielsweise sollte die Anfrage

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

zu der folgenden (oder einer äquivalenten) Antwort führen:

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

Hinweis: Führen Sie geeignete Hilfsprädikate ein.