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

12. Januar 2016

Logik in der Informatik

Wintersemester 2015/2016

Übungsblatt 11

Abgabe: *bis 21. 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: **(24 Punkte)**

Sei σ eine Signatur, sei $\Gamma \subseteq_e \mathsf{FO}[\sigma]$, seien $\varphi, \psi, \chi \in \mathsf{FO}[\sigma]$ und seien $x, y \in \mathsf{VAR}$.

Beweisen Sie die Korrektheit der folgenden Regeln des Sequenzenkalküls.

(a) $\wedge$-Einführung im Sukzedens ($\wedge$S):

$$\frac{\begin{array}{l}\Gamma \vdash \varphi \\ \Gamma \vdash \psi\end{array}}{\Gamma \vdash (\varphi \wedge \psi)}$$

(b) $\vee$-Einführung im Antezedens ($\vee$A):

$$\frac{\begin{array}{l}\Gamma, \varphi \vdash \chi \\ \Gamma, \psi \vdash \chi\end{array}}{\Gamma, (\varphi \vee \psi) \vdash \chi}$$

(c) $\exists$-Einführung im Antezedens ($\exists$A):

$$\frac{\Gamma, \varphi\frac{y}{x} \vdash \psi}{\Gamma, \exists x\varphi \vdash \psi} \qquad \text{falls } y \notin \text{frei}(\Gamma, \exists x\varphi, \psi)$$

Aufgabe 2: **(24 Punkte)**

Sei σ eine Signatur.

Beweisen Sie das Substitutionslemma (Lemma 4.14 im Skript). Gehen Sie dabei wie folgt vor:

(a) Beweisen Sie das Substitutionslemma für σ-Terme,
d.h. beweisen Sie die folgende *Behauptung (a)*:

Für alle σ-Terme t und u, jede Variable $x \in \mathsf{VAR}$ und jede σ-Interpretation $\mathcal{I}$ gilt:

$$[\![u\tfrac{t}{x}]\!]^{\mathcal{I}} \quad = \quad [\![u]\!]^{\mathcal{I}\frac{t}{x}} .$$

(b) Beweisen Sie das Substitutionslemma für $\mathsf{FO}[\sigma]$-Formeln.

Bemerkung: Die im Skript auf Seite 193 für eine $\mathsf{FO}[\sigma]$-Formel φ, einen σ-Term t und eine Variable $x \in \mathsf{VAR}$ beschriebene Formel $\varphi\frac{t}{x}$ ist nur bis auf Äquivalenz eindeutig definiert.

Wir definieren aus diesem Grund für jede $\mathsf{FO}[\sigma]$-Formel φ, jeden σ-Term t und jede Variable $x \in \mathsf{VAR}$ die Menge $M(\varphi, t, x)$ aller gemäß der Definition auf Seite 193 des Skripts gebildeten $\mathsf{FO}[\sigma]$-Formeln $\varphi\frac{t}{x}$.

Beweisen Sie nun die folgende *Behauptung (b)*:

Sei φ eine $\mathsf{FO}[\sigma]$-Formel, t ein σ-Term, $x \in \mathsf{VAR}$ und $\psi \in M(\varphi, x, t)$.
Dann gilt für jede σ-Interpretation $\mathcal{I}$, dass

$$\mathcal{I} \models \psi \quad \Longleftrightarrow \quad \mathcal{I}\tfrac{t}{x} \models \varphi.$$

Hinweis: Benutzen Sie Behauptung (a).

Aufgabe 3: **(27 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: $\text{abl}_{\mathfrak{K}} = B$.

Aufgabe 4: **(25 Punkte)**

Lesen Sie Kapitel 7 aus dem Buch „Learn Prolog Now!“.

Achtung: *Geben Sie Ihre Lösungsansätze für die Teilaufgaben (b)–(e) in einer Datei mit dem Namen* `dpll.pl` *über das GOYA-System ab!* ***Es gilt:*** *Lösungsansätze, die von SWI-Prolog nicht geladen werden können, werden nicht bewertet!*

(a) Machen Sie sich mit den Prolog-Modulen vertraut, die Sie unter der URL `http://www2.informatik.hu-berlin.de/logik/lehre/WS15-16/Logik/downloads/al/` finden können. Laden Sie diese Prolog-Module in ein Verzeichnis Ihrer Wahl.

(b) Erstellen Sie (im selben Verzeichnis) in einer Datei `dpll.pl` ein Modul mit dem Namen `dpll`, welches das Prädikat `dpll/1` exportiert.

(c) Importieren Sie im Modul `dpll` die Prädikate aus den Prolog-Modulen aus Teilaufgabe (a), die Sie für die Lösung der folgenden Teilaufgabe benötigen.

(d) Wir kodieren Klauselmengen wie auf den Blättern 9 und 10 als Listen von Listen von Literalen. Implementieren Sie das Prädikat `dpll/1`, so dass eine Anfrage `?- dpll(KM).` für eine Klauselmenge `KM` genau dann erfolgreich ist, wenn die Klauselmenge erfüllbar ist.

Beispielsweise sollte die Anfrage `?- dpll(KM)` für die Klauselmenge

```
KM = [[x1,~x5,~x6,x7], [~x1,x2,~x5], [~x1,~x2,~x3,~x5,~x6], [x1,x2,~x4,x7],
      [~x4,~x6,~x7], [x3,~x5,x7], [x3,~x4,~x5], [x5,~x6], [x5,x4,~x8],
      [x1,x3,x5,x6,x7], [~x7,x8], [~x6,~x7,~x8]]
```

mit `true.` beantwortet werden. Es macht hierbei nichts, wenn die Antwort `true.` durch das Backtracking mehrfach ausgegeben werden kann. Für die Klauselmenge

```
KM = [[~r,t,w], [~r,~s,~w], [~r,~t], [~q,s,t], [~q,r,~s], [r,s,w],
      [r,~t,~w], [q,u], [s,~u,~w], [q,w], [q,~s,~u]]
```

sollte die selbe Anfrage jedoch mit `false.` beantwortet werden.

Hinweise: Implementieren Sie dazu den *DPLL-Algorithmus*, wie er auf Seite 89 des Skripts beschrieben ist. Definieren Sie geeignete Hilfsprädikate. Nutzen Sie insbesondere die bereits in Blatt 9 und Blatt 10 implementierten Vereinfachungsheuristiken *Unit Propagation* und *Pure Literal Rule*, die Sie aus den Modulen des entsprechenden Namens importieren können. Die Streichung von Klauseln, die Obermengen von anderen Klauseln sind, müssen Sie nicht implementieren.

(e) Gegeben sei die kontextfreie Grammatik $G = (\Sigma, V, S, P)$ mit den Terminalsymbolen $\Sigma := \{\texttt{if}, \texttt{then}, \texttt{else}, \texttt{e1}, \texttt{e2}, \texttt{s1}, \texttt{s2}\}$, den Nichtterminalsymbolen $V := \{\texttt{stmt}, \texttt{expr}\}$, dem Startsymbol $S := \texttt{stmt}$, und den Produktionen $P :=$

$$\{ \ \texttt{stmt} \longrightarrow \texttt{if expr then stmt}, \quad \texttt{stmt} \longrightarrow \texttt{if expr then stmt else stmt}, \quad \texttt{stmt} \longrightarrow \texttt{s1}, \quad \texttt{stmt} \longrightarrow \texttt{s2}, \quad \texttt{expr} \longrightarrow \texttt{e1}, \quad \texttt{expr} \longrightarrow \texttt{e2} \ \}$$

Bilden Sie für die kontextfreie Grammatik G eine *Definite Clause Grammar (DCG)*, so dass die Anfrage `?- stmt(X, []).` genau dann erfüllt wird, wenn `X` eine Liste von Terminalsymbolen aus Σ ist, die einem Wort der durch G beschriebenen Sprache entspricht. Dies gilt bspw. für die Liste `X = [ if, e1, then, if, e2, then, s1, else, s2]` . Fügen Sie Ihre Definite Clause Grammar der Datei `dpll.pl` hinzu.

(f) Untersuchen Sie mit der Anfrage `?- listing.` die interne Darstellung Ihrer DCG aus Teilaufgabe (e) in SWI-Prolog. Erklären Sie die Ausgabe von SWI-Prolog bei der Anfrage

```
?- stmt([ if, e1, then, if, e2, then, s1, else, s2], []).
```