Humboldt-Universität zu Berlin
Lehrstuhl Logik in der Informatik
Dr. Christoph Berkholz

22. Januar 2019

Logik in der Informatik

Wintersemester 2018/2019

Übungsblatt 12

Abgabe: *bis 29. 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))*

Aufgabe 1: (24 Punkte)

(a) Sei σ eine Signatur und sei $\varphi \in \mathsf{FO}[\sigma]$. Leiten Sie ähnlich wie in Beispiel 4.19 die Sequenz

$$\varphi \vdash \neg\neg\varphi$$

im Sequenzenkalkül $\mathfrak{K}_S$ ab.

(b) Sei σ eine Signatur, die ein einstelliges Relationssymbol P enthält. Seien x und y zwei verschiedene Variablen. Leiten Sie die Sequenz

$$P(x),\ \forall x \forall y\, x{=}y \ \vdash\ \forall y\, P(y)$$

im Sequenzenkalkül $\mathfrak{K}_S$ ab.

Aufgabe 2: (26 Punkte)

Sei $\sigma := \{E\}$ die Signatur, die aus einem 2-stelligen Relationssymbol E besteht.

(a) Zeigen Sie, dass die Klasse aller azyklischen (endlichen oder unendlichen) Graphen erststufig axiomatisierbar ist.

(b) Nutzen Sie den Endlichkeitssatz der Logik erster Stufe, um zu zeigen, dass die Klasse aller *nicht* azyklischen (endlichen oder unendlichen) Graphen *nicht* erststufig axiomatisierbar ist.

Zur Erinnerung: Ein gerichteter Graph ist azyklisch, falls er keinen Kreis endlicher Länge besitzt.

— *auf der nächsten Seite geht's weiter* —

Aufgabe 3: **(25 Punkte)**

Wir betrachten in dieser Aufgabe Kalküle über der Menge $M := \mathsf{AL}(\{\neg, \rightarrow\})$.

Ein Kalkül $\mathfrak{K}$ über M heißt *korrekt*, falls für jede Menge $\Phi \subseteq M$ und jede Formel $\psi \in M$ gilt: Wenn $\psi \in \mathrm{abl}_{\mathfrak{K}}(\Phi)$, dann gilt $\Phi \models \psi$. Ein Kalkül $\mathfrak{K}$ über M heißt *vollständig*, falls für jede Menge $\Phi \subseteq M$ und jede Formel $\psi \in M$ gilt: Wenn $\Phi \models \psi$, dann ist $\psi \in \mathrm{abl}_{\mathfrak{K}}(\Phi)$.

Seien $\mathfrak{K}_{Syl}$ und $\mathfrak{K}_{Abd}$ die beiden folgenden Kalküle über der Menge M: Beide Kalküle enthalten für jede *allgemeingültige* aussagenlogische Formel $\varphi \in M$ das Axiom $\dfrac{}{\varphi}$.

Außerdem enthält

- $\mathfrak{K}_{Abd}$ für alle $\varphi, \psi \in M$ die Ableitungsregel

$$\frac{\psi \qquad (\varphi \rightarrow \psi)}{\varphi} \ ,$$

  die *Abduktion* genannt wird,

- $\mathfrak{K}_{Syl}$ für alle $\varphi, \psi, \chi \in M$ die Ableitungsregel

$$\frac{(\varphi \rightarrow \psi) \qquad (\psi \rightarrow \chi)}{(\varphi \rightarrow \chi)} \ ,$$

  die *Syllogismus* genannt wird.

(a) Geben Sie für $\varphi := \neg(A_0 \rightarrow A_0)$ und $\Phi := \emptyset$ eine möglichst kurze Ableitung von φ aus Φ in $\mathfrak{K}_{Abd}$ an.

(b) Beweisen Sie, dass $\mathfrak{K}_{Abd}$ vollständig, aber nicht korrekt ist.

(c) Beweisen Sie, dass $\mathfrak{K}_{Syl}$ korrekt, aber nicht vollständig ist.

(d) Betrachten Sie den Kalkül $\mathfrak{K}$ über M, der alle Ableitungsregeln aus $\mathfrak{K}_{Syl}$ und alle Ableitungsregeln aus $\mathfrak{K}_{Abd}$ enthält. Ist $\mathfrak{K}$ korrekt bzw. vollständig? Begründen Sie Ihre Antwort.

— auf der nächsten Seite geht's weiter —

Aufgabe 4: (25 Punkte)

Lesen Sie Kapitel 7 aus dem Buch „Learn Prolog Now!“.
Achtung: *Die Bearbeitung der Aufgabe ist unter Beachtung der bekannten Abgabehinweise über Moodle abzugeben! Analog zu früheren Blättern finden Sie die benötigten Dateien auf der Seite zur Prolog-Übung. Machen Sie sich auch mit den neuen Prolog-Modulen* `unit_propagation` *und* `pure_literal` *vertraut.*[1]

(a) Importieren Sie im Modul `dpll` Ihrer Abgabe `blatt12.pl` die Prädikate aus den Prolog-Modulen, die Sie für die Lösung der folgenden Teilaufgabe benötigen.

(b) Wir kodieren Klauselmengen, wie gewohnt, 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 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]]
```

erfüllt sein. 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 scheitern.
Hinweise: Implementieren Sie dazu den *DPLL-Algorithmus*, wie er auf Seite **??** des Skripts beschrieben ist. Definieren Sie geeignete Hilfsprädikate. Nutzen Sie insbesondere die bereits auf Blatt 10 und 11 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.

(c) 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 $P :=$

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

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 beispielsweise für die Liste

```
X = [ if, e1, then, if, e2, then, s1, else, s2] .
```

Fügen Sie Ihre Definite Clause Grammar der Datei `blatt11.pl` hinzu.

[1] Verfügbar ab 22.01.19.