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

16. Januar 2020

Logik in der Informatik

Wintersemester 2019/2020

Übungsblatt 12

Abgabe: *bis 28. 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: (24 Punkte)

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

(a) Leiten Sie ähnlich wie in Beispiel 4.19 aus dem Skript die folgende Sequenz im Sequenzenkalkül $\mathfrak{K}_S$ ab.

$$\varphi,\ (\neg\varphi \vee \psi)\ \vdash\ \psi$$

(b) Beweisen Sie die Korrektheit der $\wedge$-Einführung im Sukzedens ($\wedge$S):

$$\frac{\Gamma \vdash \varphi \qquad \Gamma \vdash \psi}{\Gamma \vdash (\varphi \wedge \psi)}$$

(c) **Beweisen oder widerlegen** Sie die Korrektheit der folgenden Sequenzenregel

$$\frac{\Gamma, \psi \vdash \varphi \qquad \Gamma, \varphi \vdash \psi}{\Gamma \vdash \varphi \vee \psi}$$

(d) **Beweisen oder widerlegen** Sie die Korrektheit der folgenden Sequenzenregel

$$\frac{\Gamma, \varphi \vdash \psi}{\Gamma \vdash \neg\varphi \vee \psi}$$

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 12 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.*

(a) Machen Sie sich mit den Prolog-Modulen `al_def`, `al_literals` und `al_nf` vertraut, welche Sie auf der Seite zur Prolog-Übung finden können. Laden Sie diese Prolog-Module in ein Verzeichnis Ihrer Wahl.

(b) Erstellen Sie (im selben Verzeichnis) in einer Datei `blatt12.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 11 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.