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

9. Januar 2026

Logik in der Informatik

Wintersemester 2025/2026

Übungsblatt 11

Abgabe: *bis 19. Januar 2026, 13.^{00} Uhr*

Aufgabe 1: (Moodle-Quiz)

Absolvieren Sie das Quiz 11 auf der Moodle-Plattform.

Aufgabe 2: (Präsenzaufgabe)

Sei $\varphi \in \mathsf{FO}[\sigma]$, wobei σ eine Signatur ist, die das 1-stellige Relationssymbol P enthält. Seien x und y zwei verschiedene Variablen. Leiten Sie ähnlich wie in Beispiel 4.19 aus dem Skript die folgenden beiden Sequenzen im Sequenzenkalkül $\mathfrak{K}_S$ ab.

(a) $\neg\neg\varphi \ \vdash \ \varphi$

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

Aufgabe 3: (40 Punkte)

(a) 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 Sequenzenregeln:

(i) $\vee$-Einführung im Antezedens $(\vee\text{A})$:

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

(ii) $\wedge$-Einführung im Sukzedens $(\wedge\text{S})$:

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

(iii) $\exists$-Einführung im Antezedens $(\exists\text{A})$:

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

— auf der nächsten Seite geht's weiter —

(b) Wir betrachten in dieser Teilaufgabe Kalküle über der Menge $M := \mathsf{AL}(\{\neg, \to\})$.
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 $\mathfrak{K}_{\mathrm{Syl}}$ und $\mathfrak{K}_{\mathrm{Abd}}$ die beiden folgenden Kalküle über der Menge M: Beide Kalküle enthalten für jede *allgemeingültige* aussagenlogische Formel $\varphi \in M$ das Axiom $\frac{}{\varphi}$.
Außerdem enthält

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

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

die *Abduktion* genannt wird, und

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

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

die *Syllogismus* genannt wird.

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

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

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

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

Aufgabe 4: (20 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.*

(a) Importieren Sie im Modul `dpll` Ihrer Abgabe `blatt11.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`, sodass 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]]
```

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

erfolgreich 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 dieselbe Anfrage jedoch scheitern.
Hinweise: Implementieren Sie dazu den *DPLL-Algorithmus*, wie er auf den Seiten 93/94 des Skripts beschrieben ist. Definieren Sie geeignete Hilfsprädikate. Nutzen Sie insbesondere die bereits auf Blatt 9 und 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.

(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 $P :=$

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

Bilden Sie für die kontextfreie Grammatik G eine *Definite Clause Grammar (DCG)*, sodass 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.