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

16. Januar 2026

Logik in der Informatik

Wintersemester 2025/2026

Übungsblatt 12

Abgabe: *bis 26. Januar 2026, 13.^{00} Uhr*

Aufgabe 1: (Moodle-Quiz)

Absolvieren Sie das Quiz 12 auf der Moodle-Plattform.

Aufgabe 2: (Präsenzaufgabe)

(a) Sei $\sigma := \{ E \}$ die Signatur, die aus dem 2-stelligen Relationssymbol E besteht.

- **(i)** Zeigen Sie, dass die Klasse aller azyklischen (endlichen oder unendlichen) Graphen erststufig axiomatisierbar ist.
- **(ii)** 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.

Hinweis: Ein gerichteter Graph ist azyklisch, falls es kein $\ell \in \mathbb{N}_{\geqslant 1}$ und keine Knoten $a_1, \ldots, a_\ell$ gibt, sodass (a_ℓ, a_1) und (a_i, a_{i+1}) für alle $i \in [\ell - 1]$ Kanten im Graphen bilden.

(b) Sei $\sigma := \{R, f_0, f_1, c\}$, wobei c ein Konstantensymbol, R ein 2-stelliges Relationssymbol und f_0, f_1 zwei 1-stellige Funktionssymbole sind.

Beweisen Sie folgende Aussagen aus Korollar 4.41 aus dem Vorlesungsskript:

- **(i)** Das Unerfüllbarkeitsproblem für $\mathsf{FO}[\sigma]$ ist nicht entscheidbar.
- **(ii)** Das Erfüllbarkeitsproblem für $\mathsf{FO}[\sigma]$ ist nicht semi-entscheidbar.

Aufgabe 3: (40 Punkte)

(a) Sei $\sigma := \{ E \}$ die Signatur, die aus dem 2-stelligen Relationssymbol E besteht und seien $\varphi, \psi \in \mathsf{FO}[\sigma]$. Seien $x, y \in \mathsf{VAR}$ zwei verschiedene Variablen. Leiten Sie ähnlich wie in Beispiel 4.19 aus dem Skript die folgenden beiden Sequenzen im Sequenzenkalkül $\mathfrak{K}_S$ ab.

- **(i)** $\varphi \vdash \neg\neg\varphi$
- **(ii)** $\varphi, (\neg\varphi \vee \psi) \vdash \psi$

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

(b) Sei $\sigma := \{ E, f \}$ die Signatur, die aus dem 2-stelligen Relationssymbol E und dem 1-stelligen Funktionssymbol f besteht. Betrachten Sie das Alphabet $A := A_{\mathsf{FO}[\sigma]}$ und die Menge $M := A^*$.

Definieren Sie $\mathsf{FO}[\sigma]$ mithilfe eines geeigneten Kalküls $\mathfrak{K}$. Das heißt, $\mathrm{abl}_{\mathfrak{K}}$ soll aus genau denjenigen Elementen von M bestehen, die gemäß Definition 3.15 syntaktisch korrekte Formeln der Logik erster Stufe über der Signatur σ sind, ohne dass Sie $\mathsf{FO}[\sigma]$ in der Definition des Kalküls $\mathfrak{K}$ verwenden.

(c) Sei $\Sigma := \{ a, b \}$ und $\sigma_\Sigma := \{ \leqslant, P_a, P_b \}$ die Signatur mit dem 2-stelligen Relationssymbol $\leqslant$ und den zwei 1-stelligen Relationssymbolen P_a und P_b.

Definition: Eine Sprache $L \subseteq \Sigma^*$ ist *erststufig axiomatisierbar*, falls es eine (möglicherweise unendliche) Menge Φ von $\mathsf{FO}[\sigma_\Sigma]$-Sätzen gibt, sodass für alle nicht-leeren Worte $w \in \Sigma^*$ gilt:

$$w \in L \quad \Longleftrightarrow \quad \mathcal{A}_w \models \Phi$$

Zur Erinnerung: $\mathcal{A}_w$ ist die zu w gehörige Wortstruktur.

(i) Zeigen Sie, dass die durch den regulären Ausdruck $(ba)^*$ beschriebene Sprache $L_1 \subseteq \Sigma^*$ erststufig axiomatisierbar ist.

(ii) Sei für ein Wort $u \in \Sigma^*$ die Sprache $L(u)$ definiert als die Menge $\{ w \in \Sigma^* \ : \ w \neq u \}$. Zeigen Sie, dass die Sprache $L(u)$ für jedes $u \in \Sigma^*$ erststufig axiomatisierbar ist.

(iii) Nutzen Sie **(ii)** um zu zeigen, dass die Sprache $L_2 \subseteq \Sigma^*$, die durch den regulären Ausdruck $(aa)^*$ beschrieben wird, erststufig axiomatisierbar ist.
Beachten Sie, dass L_2 *nicht* FO-definierbar ist.

(iv) Zeigen Sie mithilfe von **(ii)**, dass *jede* Sprache $L \subseteq \Sigma^*$ erststufig axiomatisierbar ist.

Aufgabe 4: (20 Punkte)

Lesen Sie Kapitel 8 aus dem Buch „Learn Prolog Now!“.
Achtung: *Die Bearbeitung der Aufgabe ist unter Beachtung der bekannten Abgabehinweise über Moodle abzugeben! Sie können zur Lösung dieser Aufgabe und der Zusatzaufgabe alle Prolog-Module verwenden, die Sie unter*

https://hu.berlin/prolog

vorfinden. Dies gilt insbesondere für die Module `tseitin.pl` *und* `dpll.pl`.[1]

(a) Implementieren Sie ein Prädikat `sat/1`, so dass eine Anfrage

```
?- sat(F).
```

für eine aussagenlogische Formel `F` genau dann erfolgreich ist, wenn `F` erfüllbar ist.

Hinweise: Ihr Prädikat soll zu der Formel `F` zuerst eine *erfüllbarkeitsäquivalente* 3-*KNF* konstruieren, und anschließend deren Erfüllbarkeit mit dem DPLL-Algorithmus testen. Es macht nichts, wenn Ihr Prädikat für eine erfüllbare Formel mehrfach

```
true.
```

ausgibt.

[1] Verfügbar ab 19.01.26 ca. 20:00 Uhr.

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

(b) Für die folgenden Teilaufgaben betrachten wir *geordnete und beschriftete Binärbäume* (analog zu Blatt 6) mit einer leicht geänderten Termrepräsentation.[2] Das folgende Prädikat implementiert eine *pre-order-Traversierung* eines gegebenen Binärbaums `B` in Prolog und veranschaulicht die leicht geänderte Termrepräsentation:

```
preorder(node(T, nil, nil), [T]) :- !.
preorder(node(T, L, R), [T|X]) :-
    preorder(L, LX), preorder(R, RX), append(LX, RX, X).
```

Schreiben Sie eine *Definite Clause Grammar mit einem zusätzlichen Argument*, so dass die Anfrage

```
?- pt(B, X, []).
```

für einen Binärbaum `B` und eine Liste `X` genau dann erfüllt ist, wenn `X` eine pre-order-Traversierung von `B` ist.

(c) Wir nennen einen Binärbaum einen *binären Suchbaum*, wenn seine Knoten nur mit Ganzzahlen beschriftet sind und für jeden seiner inneren Knoten `node(Z, L, R)` gilt: Jeder Knoten in `L` ist mit einer Zahl kleiner `Z`, und jeder Knoten in `R` mit einer Zahl größer `Z` beschriftet.

Schreiben Sie eine *Definite Clause Grammar mit drei zusätzlichen Argumenten*, so dass die Anfrage

```
?- bst(B, Min, Max, X, []).
```

genau dann erfüllt ist, wenn `X` die pre-order-Traversierung für den *binären Suchbaum* `B` ist, und zusätzlich `Min` und `Max` die kleinste, beziehungsweise größte Ganzzahl ist, die als Beschriftung in `B` vorkommt.

Hinweis: Nutzen Sie die Möglichkeit, Ihrer Definite Clause Grammar mit Hilfe der Notation {...} zusätzliche Ziele hinzuzufügen. Verwenden Sie gegebenfalls das eingebaute Prädikat `integer/1`.

Aufgabe 5: (20 Zusatzpunkte)

Für Vergleiche von SAT-Solvern werden 3-KNF oft im sogenannten DIMACS-Format angegeben.[3] Implementieren Sie ein Prädikat `sat_dimacs/1`, welches als Argument den Namen einer Datei erhält, so dass beispielsweise die Anfrage

```
?- sat_dimacs('knf.cnf').
```

genau dann erfolgreich ist, wenn die in der Datei `knf.cnf` repräsentierte 3-KNF erfüllbar ist. Ist diese nicht erfüllbar, soll das Ergebnis `false.` sein.

Sie können in Ihrer Implementation davon ausgehen, dass die aufgerufene Datei im aktuellen Verzeichnis existiert und dem DIMACS-Standard entspricht.

Unter https://hu.berlin/prolog finden Sie auch Beispieldateien im DIMACS-Format.

[2] Dennoch haben innere Knoten weiterhin exakt zwei Kinder.

[3] So waren auch auf der SAT Competition 2025 [Link: http://www.satcompetition.org/] die 2011 festgelegten Regeln gültig, welche auch für uns das DIMACS-Format definieren sollen (vgl. https://satcompetition.github.io/2025/benchmarks.html).