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

2. Januar 2026

Logik in der Informatik

Wintersemester 2025/2026

Übungsblatt 10

Abgabe: *bis 12. Januar 2026, 13.^{00} Uhr*

Aufgabe 1: (Moodle-Quiz)

Absolvieren Sie das Quiz 10 auf der Moodle-Plattform.

Aufgabe 2: (Präsenzaufgabe)

(a) Sei σ eine beliebige Signatur. Zeigen Sie, dass es Formeln $\varphi, \psi \in \mathsf{FO}[\sigma]$ gibt, für die gilt:

$$(\,\varphi \;\wedge\; \forall x\,\psi\,) \;\not\equiv\; \forall x\,(\,\varphi \;\wedge\; \psi\,)$$

(b) Katzen äußern sich bekanntlich mithilfe der Laute „M“, „I“ und „U“.
Die *Katzensprache* K ist eine Menge von Worten über dem Alphabet $A := \{\mathtt{M}, \mathtt{I}, \mathtt{U}\}$, die durch die folgenden Regeln rekursiv definiert ist:

Basisregel: **(B)** $\mathtt{MI} \in K$.

Rekursive Regeln: Für alle $v, w \in A^*$ gilt:

(R1) ist $v\mathtt{I} \in K$, so ist auch $v\mathtt{IU} \in K$;

(R2) ist $\mathtt{M}v \in K$, so ist auch $\mathtt{M}vv \in K$;

(R3) ist $v\mathtt{III}w \in K$, ist auch $v\mathtt{U}w \in K$; und

(R4) ist $v\mathtt{UU}w \in K$, ist auch $vw \in K$.

(i) Beweisen Sie durch Induktion über den Aufbau der Menge K, dass für jedes Wort $w \in K$ gilt:

> Die Anzahl $|w|_{\mathtt{I}}$ der Vorkommen des Lauts $\mathtt{I}$ in w ist *nicht* durch 3 teilbar (d. h. es gibt eine Zahl $k \in \mathbb{N}$, sodass gilt: $|w|_{\mathtt{I}} = 3k+1$ oder $|w|_{\mathtt{I}} = 3k+2$).

(ii) Geben Sie einen Kalkül $\mathfrak{K}$ über der Menge A^* an, der die Sprache K definiert, d. h. es soll gelten: $\mathrm{abl}_{\mathfrak{K}} = K$.

(iii) Geben Sie für jedes der folgenden Worte entweder eine Ableitung des Wortes in $\mathfrak{K}$ an oder beweisen Sie, dass es nicht in der Menge $\mathrm{abl}_{\mathfrak{K}}$ liegt.

(i) $\mathtt{MIU}$ **(ii)** $\mathtt{MUII}$ **(iii)** $\mathtt{MUUU}$

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

Aufgabe 3: (40 Punkte)

(a) Welche der folgenden beiden Aussagen ist für jede Signatur σ und jede $\mathsf{FO}[\sigma]$-Formel φ korrekt, welche nicht? Beweisen Sie, dass ihre Antworten korrekt sind.

(i) $\exists x \forall y\, \varphi \models \forall y \exists x\, \varphi$ **(ii)** $\forall y \exists x\, \varphi \models \exists x \forall y\, \varphi$

(b) Sei $\sigma := \{ E \}$ die Signatur mit dem 2-stelligen Relationssymbol E. Betrachten Sie die $\mathsf{FO}[\sigma]$-Formel

$$\varphi(x,z) \;:=\; \exists y \Big(E(z,y) \to \big(\forall y\, E(x,y) \;\wedge\; \neg \exists x\, E(x,y) \big) \Big)$$

(i) Berechnen Sie eine zu φ äquivalente $\mathsf{FO}[\sigma]$-Formel in Negationsnormalform.

(ii) Berechnen Sie eine zu φ äquivalente $\mathsf{FO}[\sigma]$-Formel in Pränex-Normalform.

Gehen Sie hierbei ähnlich wie in Beispiel 3.70 vor. Machen Sie pro Zwischenschritt nur eine Umformung und kommentieren Sie Ihre Zwischenschritte.

(c) Beweisen Sie Satz 3.67 aus der Vorlesung:

> Jede $\mathsf{FO}[\sigma]$-Formel φ ist äquivalent zu einer Formel in NNF.

(d) Sei E ein 2-stelliges Relationssymbol und sei $\mathsf{2\text{-}COL}$ die Klasse aller gerichteten, 2-färbbaren Graphen, d. h. aller $\{ E \}$-Strukturen $\mathcal{A} = (A, E^{\mathcal{A}})$ für die gilt:

> Es gibt eine Funktion $f \colon A \to \{ \mathit{rot}, \mathit{blau} \}$ mit $f(a) \neq f(b)$ für alle $(a,b) \in E^{\mathcal{A}}$.

Zeigen Sie mittels **logischer Reduktion** (ähnlich wie im Beweis von Satz 3.58), dass die Klasse $\mathsf{2\text{-}COL}$ *nicht* FO-definierbar ist.

Aufgabe 4: (20 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 Veranstaltung.*

(a) Machen Sie sich mit den Prolog-Modulen `al_def`, `al_literals` und `al_nf` vertraut und laden Sie diese Prolog-Module in ein Verzeichnis Ihrer Wahl.

(b) Erstellen Sie (im selben Verzeichnis) in einer Datei `blatt10.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 9 als Listen von Listen von Literalen.

Implementieren Sie das Prädikat `knf_shell/0`, so dass eine Anfrage

```
?- knf_shell.
```

— auf der nächsten Seite geht's weiter —

eine Eingabeaufforderung zur Konstruktion von Klauselmengen aus aussagenlogischen Formeln startet.

Das heißt, 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) *Zur Erinnerung: Pure Literal Rule*

> Literale λ, deren Negat $\overline{\lambda}$ nirgendwo in der Klauselmenge auftaucht, können auf 1 gesetzt werden. Alle Klauseln, die ein solches Literal enthalten, sind dann wahr und können gestrichen werden. Wiederhole dies, so lange es Literale gibt, deren Negat nirgendwo in der Klauselmenge auftaucht.

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 solange anwendet, wie es Literale gibt, deren Negat nirgendwo in der Klauselmenge auftaucht 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.