

# Logik in der Informatik

Wintersemester 2018/2019

## Übungsblatt 11

**Abgabe:** bis 22. Januar 2019, 11.<sup>15</sup> 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:

(20 Punkte)

**Beweisen oder widerlegen** Sie die folgenden Aussagen:

(a) Sei  $\sigma = \emptyset$ . Es gibt einen FO[ $\sigma$ ]-Satz  $\varphi_a$ , so dass für jede  $\sigma$ -Struktur  $\mathcal{A}$  gilt:

$$\mathcal{A} \models \varphi_a \iff |A| \text{ ist eine Primzahl.}$$

(b) Sei  $\sigma = \{E/2, F/2\}$  eine relationale Signatur. Es gibt einen FO[ $\sigma$ ]-Satz  $\varphi_b$ , so dass für jede  $\sigma$ -Struktur  $\mathcal{A} = (A, E^{\mathcal{A}}, F^{\mathcal{A}})$ , bei der  $\mathcal{G} := (A, E^{\mathcal{A}})$  ein endlicher gerichteter Pfad ist, gilt:

$$\mathcal{A} \models \varphi_b \iff F^{\mathcal{A}} \text{ ist der reflexive und transitive Abschluss von } E^{\mathcal{A}}.$$

Dabei nutzen wir folgende Begriffe:

- Ein Graph  $\mathcal{G} = (V^{\mathcal{G}}, E^{\mathcal{G}})$  ist ein endlicher gerichteter Pfad, falls es ein  $n \in \mathbb{N}$  mit  $n \geq 1$  gibt, so dass  $|V^{\mathcal{G}}| = n$ ,  $V^{\mathcal{G}} = \{v_1, \dots, v_n\}$  und  $E^{\mathcal{G}} = \{(v_i, v_{i+1}) : 1 \leq i < n\}$ .
- Seien  $E^{\mathcal{A}}, F^{\mathcal{A}} \subseteq A \times A$ .  $F^{\mathcal{A}}$  heißt transitiver und reflexiver Abschluss von  $E^{\mathcal{A}}$ , wenn für alle  $(a, a') \in A \times A$  gilt:

$$(a, a') \in F^{\mathcal{A}} \iff a = a' \text{ oder es gibt im gerichteten Graphen } \mathcal{G} = (A, E^{\mathcal{A}}) \text{ einen Weg vom Knoten } a \text{ zum Knoten } a'.$$

### Aufgabe 2:

(30 Punkte)

Katzen äußern sich bekanntlich mit Hilfe der Laute „M“, „I“ und „U“.

Die *Katzensprache*  $K$  ist eine Menge von Worten über dem Alphabet  $A := \{M, I, U\}$ , die durch die folgenden Regeln rekursiv definiert ist:

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

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

(R1) Ist  $vI \in K$ , so ist auch  $vIU \in K$ .

(R2) Ist  $Mv \in K$ , so ist auch  $Mvv \in K$ .

(R3) Ist  $vIIIw \in K$ , so ist auch  $vUw \in K$ .

(R4) Ist  $vUUw \in K$ , so ist auch  $vw \in K$ .

(a) Geben Sie für jedes der folgenden Worte aus  $A^*$  an, ob es zur Menge  $K$  gehört oder nicht. Begründen Sie jeweils Ihre Antwort!

- (i) MIU
- (iii) MUII
- (ii) UMII
- (iv) MU

- (b) Beweisen Sie mit einer Induktion über den Aufbau der Menge  $K$ , dass für jedes Wort  $w \in K$  gilt: Die Anzahl  $|w|_I$  der Vorkommen des Lauts  $I$  in  $w$  ist *nicht* durch 3 teilbar (d.h. es gibt eine Zahl  $k \in \mathbb{N}$ , so dass gilt:  $|w|_I = 3k + 1$  oder  $|w|_I = 3k + 2$ ).
- (c) Kann eine Katze „MUUU“ machen? D.h., ist  $MUUU \in K$ ?
- (d) Geben Sie einen Kalkül  $\mathfrak{K}$  über der Menge  $A^*$  an, welcher die Sprache  $K$  definiert, d.h. insbesondere gilt:  $\text{abl}_{\mathfrak{K}} = K$ .

### Aufgabe 3:

(25 Punkte)

Sei  $\sigma$  eine Signatur, sei  $\Gamma \subseteq_e \text{FO}[\sigma]$ , seien  $\varphi, \psi, \chi \in \text{FO}[\sigma]$  und seien  $x, y \in \text{VAR}$ .

- (a) Beweisen Sie die Korrektheit der Sequenzenregel  $\exists$ -Einführung im Antezedens ( $\exists A$ ):

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

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

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

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

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

#### 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 Blatt 7 und 9 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 `blatt11.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.
```

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.