

Logik in der Informatik

Wintersemester 2017/2018

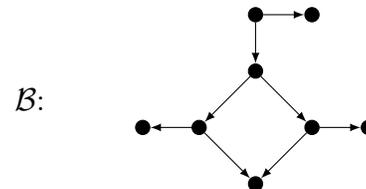
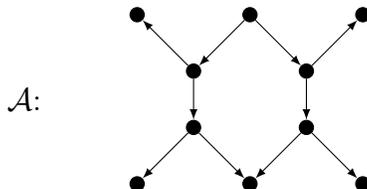
Übungsblatt 10

Abgabe: bis 16. Januar 2018, 11.¹⁰ 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:

(25 Punkte)

Sei $\sigma := \{E/2\}$. Betrachten Sie die folgenden gerichteten Graphen \mathcal{A} und \mathcal{B} :

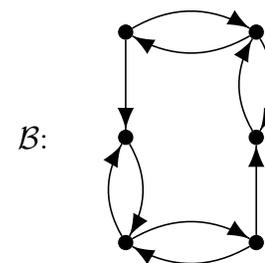
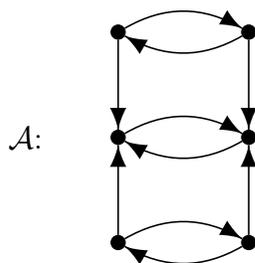


- (a) Welches ist das kleinste m , so dass Spoiler eine Gewinnstrategie im m -Runden Ehrenfeucht-Fraïssé Spiel auf \mathcal{A} und \mathcal{B} hat? Begründen Sie Ihre Antwort, indem Sie eine Gewinnstrategie für Spoiler im m -Runden Ehrenfeucht-Fraïssé Spiel und eine Gewinnstrategie für Duplicator im $(m-1)$ -Runden Ehrenfeucht-Fraïssé Spiel beschreiben.
- (b) Finden Sie für Ihre Antwort m aus Teil (a) einen FO[σ]-Satz ψ der Quantortiefe m , so dass $\mathcal{A} \models \psi$ und $\mathcal{B} \models \neg\psi$.

Aufgabe 2:

(25 Punkte)

Sei $\sigma := \{E/2\}$. Betrachten Sie die folgenden gerichteten Graphen \mathcal{A} und \mathcal{B} :



Für

$$\varphi := \exists x \exists y \forall z (E(z, x) \vee E(z, y))$$

gilt $\mathcal{A} \models \varphi$ und $\mathcal{B} \not\models \varphi$.

- (a) Leiten Sie aus dem FO[σ]-Satz φ eine Gewinnstrategie für Spoiler im EF-Spiel auf \mathcal{A} und \mathcal{B} her. Geben Sie an, wie viele Runden Spoiler benötigt, wenn er dieser Strategie folgt. Beschreiben Sie die Strategie ähnlich wie in der in der Vorlesung behandelten Beweisidee zu Satz 3.52 im Vorlesungsskript.
- (b) Existiert eine bessere Gewinnstrategie für Spoiler? D.h. eine Strategie, mit der er in weniger Runden das Spiel gewinnt? Wenn ja, dann beschreiben Sie eine solche Strategie. Wenn nein, dann begründen Sie dieses.

Aufgabe 3:**(25 Punkte)**

Nutzen Sie zur Lösung dieser Aufgabe die Methode der logischen Reduktion (ähnlich wie im Beweis von Satz 3.59).

- (a) Sei $\Sigma = \{a, b\}$ und $\sigma_\Sigma = \{\leq, P_a, P_b\}$ die Signatur, die aus dem 2-stelligen Relationssymbol \leq , sowie zwei 1-stelligen Relationssymbolen P_a und P_b besteht. Beweisen Sie, dass es keinen FO[σ_Σ]-Satz gibt, der die Sprache aller Worte aus $\{a, b\}^*$ beschreibt, in denen die Anzahl der in ihnen vorkommenden a s größer ist als die Anzahl der in ihnen vorkommenden b s.

Zur Erinnerung: Ein FO[σ_Σ]-Satz φ beschreibt eine Sprache $L \subseteq \Sigma^*$, falls für jedes nicht-leere Wort $w \in \Sigma^*$ gilt: $w \in L \iff \mathcal{A}_w \models \varphi$.

- (b) Sei 2-COL die Klasse aller gerichteten zweifärbbaren Graphen, d.h. aller $\{E/2\}$ -Strukturen $\mathcal{A} = (A, E^{\mathcal{A}})$ für die gilt:

Es gibt eine Funktion $f : A \rightarrow \{\text{rot}, \text{blau}\}$, so dass für jede Kante (a, b) in $E^{\mathcal{A}}$ gilt:
 $f(a) \neq f(b)$.

Zeigen Sie: Die Klasse 2-COL ist *nicht FO-definierbar*.

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 `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.
```

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.