

Theorie der Programmiersprachen

8. Übung

1. Aufgabe: Man zeige mittels *Grundresolution* und *prädikatenlogischer Resolution*, dass sowohl

$$F = \forall x (\neg P(x) \wedge P(f(a)))$$

als auch

$$G = \forall x \left(\left((P(f(x)) \rightarrow P(x)) \wedge P(f(f(a))) \right) \wedge \neg P(a) \right)$$

unerfüllbar sind.

2. Aufgabe: Formalisieren Sie die Aussagen (a) und (b) als prädikatenlogische Formeln. Verwenden Sie die Notation $S(x, y)$ – x ist Student von y , $G(x)$ – x ist glücklich, $M(x)$ – x mag Logik.

(a) $A =$ „Der Professor ist glücklich, wenn alle seine Studenten Logik mögen.“

(b) $B =$ „Der Professor ist glücklich, wenn er keine Studenten hat.“

Zeigen Sie durch *Grundresolution* und *prädikatenlogische Resolution*, dass (b) eine Folgerung von (a) ist. Formulieren Sie dazu $A \wedge \neg B$ in Klauselform.

Gegeben seien zwei Strukturen \mathcal{A} und \mathcal{B} mit:

$U_{\mathcal{A}} = \{p, s_1, \dots, s_n\}$	$U_{\mathcal{B}} = \mathbb{N} = \{1, 2, 3, \dots\}$
$I_{\mathcal{A}}(G) = \{x \mid x \text{ ist glücklich}\}$	$I_{\mathcal{B}}(G) = \{x \mid x = 1\}$
$I_{\mathcal{A}}(M) = \{x \mid x \text{ mag Logik}\}$	$I_{\mathcal{B}}(M) = \{x \mid x > 1\}$
$I_{\mathcal{A}}(S) = \{(x, y) \mid x \text{ ist Student von } y\}$	$I_{\mathcal{B}}(S) = \{(x, y) \mid x > y\}$
$I_{\mathcal{A}}(p) = \{p\}$	$I_{\mathcal{B}}(p) = \{1\}$
$I_{\mathcal{A}}(q) = \{s_1\}, \quad q \text{ Skolemkonstante}$	$I_{\mathcal{B}}(q) = \{2\}$

Zeigen Sie durch „Hochgehen“ in Ihren Beweisen, dass beide Strukturen kein Modell für die entsprechende Formel sind.

3. Aufgabe: Man drücke folgende Tatsachen als prädikatenlogische Formeln aus:

$A =$ „Jeder Drache ist glücklich, wenn alle seine Kinder fliegen können.“

$B =$ „Grüne Drachen können fliegen.“

$C =$ „Ein Drache ist grün, wenn er Kind mindestens eines grünen Drachen ist.“

Man zeige durch *Grundresolution* und *prädikatenlogische Resolution*, dass aus A , B und C folgt, dass *alle grünen Drachen glücklich sind*.

4. Aufgabe: Geben Sie *alle* prädikatenlogischen Resolventen von

$$\{P(f(x)), \neg Q(z), P(z)\} \quad \text{und} \quad \{\neg P(x), R(g(x), a)\}$$

an. (x, y, z sind Variablen, a ist eine Konstante)

5. Aufgabe: Man wende den Unifikationsalgorithmus auf die Literalmenge

$$L = \{P(x, y), P(f(a), g(x)), P(f(z), g(f(z)))\}$$

an.

6. Aufgabe: Aus Effizienzgründen wird in manchen Implementierungen des Unifikationsalgorithmus' auf den Test „kommt x in t vor“ (*occur check*) verzichtet. Man gebe ein Beispiel einer *nicht unifizierbaren*, zweielementigen Literalmenge L_1, L_2 an, so dass L_1 und L_2 keine gemeinsamen Variablen enthalten, und ein Unifikationsalgorithmus ohne *occur check* – je nach Implementierung – in eine unendliche Schleife gerät oder fälschlicherweise „unifizierbar“ konstatiert.

7. Aufgabe: Zeigen Sie, dass der Unifikationsalgorithmus (naiv implementiert) exponentielle Laufzeit haben kann.

Hinweis: Man betrachte das Beispiel:

$$L = \left\{ P(x_1, x_2, \dots, x_n), P(f(x_0, x_0), f(x_1, x_1), \dots, f(x_{n-1}, x_{n-1})) \right\}$$

Man überlege sich eine geeignete Datenstruktur für Literale bzw. Literalmenge, so dass das Unifizieren effizienter durchgeführt werden kann.