Universität Paderborn
Prof. Dr. Johannes Blömer
Prof. Dr. Hans Kleine Büning

04. Dezember 2015

Modellierung – WS 2015/2016

Präsenzübung 7
7. - 11. Dezember 2015
(Dieser Übungszettel besteht aus 4 Aufgaben)

Hinweis: In der Präsenzübung haben Sie die Möglichkeit unter Anleitung Ihres Tutors, das Entwickeln von Lösungen zu üben und Ihre Fragen zu klären. Jeder Präsenzübungszettel enthält eine große Auswahl an Aufgaben, von denen ein Teil in der Präsenzübung besprochen wird. Es ist *nicht* das Ziel der Präsenzübung "Musterlösungen" zu verteilen.

Aufgabe 1 (Notationen von Termen)
Geben Sie für die folgenden Terme in Infixdarstellung die passende Baum-, Postfix- und Präfixform an.

1. $(a + ((x + 5) * x)) - (y + z)$
2. $a + ((y * (x + z)) - (x + 3))$
3. $((a * b) - (c - d)) + ((5 + x) * (y - c))$

Aufgabe 2 (Abstrakte Algebra)
Die Datenstruktur *Schlange* verwaltet Zahlen in der Reihenfolge ihres Eintreffens. Operationen auf einer Schlange erlauben es, Zahlen hinten anzufügen (*enqueue*) oder vorne die am längsten wartende Zahl zu entnehmen (*dequeue*). Zusätzlich gibt es Operationen, um eine neue leere Schlange zu erzeugen (*createQueue*), die erste Zahl zurückzugeben (*front*) sowie eine Methode, die anzeigt, ob die Schlange leer ist (*empty*).

Die folgende abstrakte Algebra beschreibt eine Schlange (τ, Σ, Q) mit der **Signatur**

$$
\begin{array}{llrll}
\Sigma & = (\mathcal{S}, \mathcal{F}) \text{ mit} & & & \\
\mathcal{S} & = \{ \text{ Queue, Nat0, BOOL } \} & & & \\
\mathcal{F} & = \{ \; \mathit{createQueue}: & & \to \text{Queue}, & (F_1) \\
 & \quad \mathit{enqueue}: & \text{Queue} \times \text{Nat0} & \to \text{Queue}, & (F_2) \\
 & \quad \mathit{dequeue}: & \text{Queue} & \to \text{Queue}, & (F_3) \\
 & \quad \mathit{front}: & \text{Queue} & \to \text{Nat0}, & (F_4) \\
 & \quad \mathit{empty}: & \text{Queue} & \to \text{BOOL} \;\} & (F_5)
\end{array}
$$

Die Sorte Queue stellt eine Schlange dar. Die Sorte Nat0 beschreibt Elemente der Schlange und stellt somit Zahlen dar. Für die Sorte BOOL sind die Konstanten *true* und *false* definiert. *createQueue* ist eine 0-stellige Operation und deshalb eine Konstante.

Gegeben ist die **Menge von Axiomen**

$$
\begin{array}{lll}
Q = \{ & & \\
\quad Q_1: & dequeue(enqueue(createQueue, x)) & \rightarrow createQueue, \\
\quad Q_2: & dequeue(enqueue(enqueue(q, y), x)) & \rightarrow enqueue(dequeue(enqueue(q, x)), y), \\
\quad Q_3: & front(enqueue(createQueue, x)) & \rightarrow x, \\
\quad Q_4: & front(enqueue(q, x)) & \rightarrow front(q), \\
\quad Q_5: & empty(createQueue) & \rightarrow true, \\
\quad Q_6: & empty(enqueue(createQueue, x)) & \rightarrow false \\
\} & &
\end{array}
$$

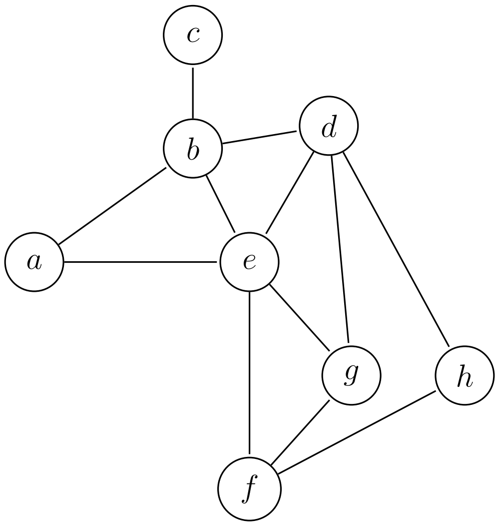
wobei x, y Variablen der Sorte Nat0 sind und q eine Variable der Sorte Queue ist.
Im Folgenden seien x, y Terme der Sorte Nat0.

1. Prüfen Sie, ob die folgenden Terme entsprechend der Signatur korrekt sind. Zeichnen Sie die Terme als Baum und notieren Sie an jedem Blatt und an jedem inneren Knoten die entsprechende Sorte. Kennzeichnen Sie die Stellen, an denen die Terme nicht korrekt gebildet wurden.
   a) $front(enqueue(createQueue, front(enqueue(createQueue, x))))$
   b) $empty(dequeue(enqueue(createQueue, x), y))$
   c) $enqueue(dequeue(createQueue), front(enqueue(createQueue, x))$
   d) $enqueue(enqueue(createQueue, x), y)$
   e) $front(dequeue(enqueue(createQueue, x)))$
   f) $front(createQueue, x)$
   g) $enqueue(a, createQueue)$
2. Formen Sie die folgenden Terme mit Hilfe der Axiome so um, dass Sie als Ergebnis eine einzelne Variable oder Konstante erhalten. Geben Sie in jedem Schritt das verwendete Axiom an.
   a) $empty(dequeue(enqueue(createQueue, y)))$
   b) $empty(enqueue(createQueue, front(enqueue(enqueue(createQueue, y), x))))$
   c) $front(enqueue(dequeue(enqueue(createQueue, y)), x))$
   d) $dequeue(enqueue(createQueue, z))$
   e) $empty(enqueue(createQueue, y))$
   f) $front(dequeue(enqueue(enqueue(createQueue, z), x)))$
   g) $front(enqueue(enqueue(createQueue, x), z))$
3. Erweitern Sie die Algebra um eine zusätzliche Operation *greatest*, die die größte Zahl in der Schlange zurückgibt. Wenn die Schlange beispielsweise die Zahlen (2,3,8,5,7) enthält, soll *greatest* für diese Schlange 8 ausgeben.
   - Nehmen Sie an, dass für die Sorte *Nat0* die Operation $max : Nat0 \times Nat0 \rightarrow Nat0$ gegeben ist.
   - $max(x, y)$ gibt x zurück, falls $x \geq y$. Andernfalls wird y zurückgegeben.
   - Sie können davon ausgehen, dass keine Zahl mehrfach in der Schlange vorhanden ist und dass die kleinste Zahl in der Schlange größer als 0 ist.

- Für die Sorte *Nat0* ist eine Konstante *zero* definiert, die dem Wert 0 entspricht.

Geben Sie die nötigen Ergänzungen der Signatur sowie der Axiome an.

Aufgabe 3 (Darstellung, Begriffe, Wegprobleme)
Gegeben sei der Graph G:



1. Stellen Sie den ungerichteten Graph $G = (V, E)$ als Menge von Knoten und Kanten dar.

2. Geben Sie den Grad der Knoten an.

3. Gibt es in dem durch die Menge $I_1 = \{a, b, c, d, e\}$ induzierten Teilgraph von G einen Euler-Weg? Zeichnen Sie den Teilgraph und begründen Sie ihre Antwort.

4. Gibt es in dem durch die Menge $I_2 = \{a, b, d, e, f, h\}$ induzierten Teilgraph von G einen Euler-Kreis? Zeichnen Sie den Teilgraph und begründen Sie ihre Antwort.

5. Gibt es in dem durch die Menge $I_3 = \{a, b, d, e, f, g, h\}$ induzierten Teilgraph von G einen Hamilton-Kreis? Zeichnen Sie den Teilgraph und begründen Sie ihre Antwort.

Aufgabe 4 (Graphen,Modellierung)

1. Es sei S die Menge der Studenten der Vorlesung Modellierung. Manche Studenten sind untereinander befreundet. Wie kann dieser Sachverhalt geeignet durch einen Graphen modelliert werden?

2. Die Bearbeitung der Modellierungsübungszettel darf in Gruppen mit maximal drei Teilnehmern erfolgen. Dabei geben Studenten nur in einer Gruppe ab, wenn sie mit jedem anderen Mitglied der Gruppe befreundet sind.

   Gibt es Studenten, die die Voraussetzungen für eine Abgabe in einer Gruppen mit 3 Mitgliedern erfüllen? Wie kann diese Frage allgemein mittels des Graphen aus der vorherigen Teilaufgabe modelliert werden?

3. Um von jemandem Lösungen zu kopieren, ist erfahrungsgemäß keine Freundschaft notwendig. Damit $x \in S$ von $y \in S$ die Lösung kopieren würde, reicht es aus, wenn x mit jemandem befreundet ist, der mit jemandem befreundet ist, der mit jemanden befreundet ist, $\dots$, der mit y befreundet ist.

   Wie kann man die Frage, ob x von y kopieren würde, mittels des Graphen aus der ersten Teilaufgabe modellieren?

4. Nun nehmen wir an, dass $K \subseteq S$ die Teilmenge der sehr klugen Studenten ist, in der jeder Student jeweils alle Aufgaben lösen kann.

   Wie kann man die Frage, ob jeder Student durch kopieren der gesamten Lösung volle Punktzahl erreichen könnte, mittels des Graphen aus der ersten Teilaufgabe modellieren?