Softwareparadigmen SS 2018, Übungsblatt 3

Abgabe: 6. Juni 2018, bis 18:00 Uhr in Ihrem Gruppen-SVN des Institutes.
Ordnerstruktur: <Repository-URL>/sol3_xx.odf → xx = Gruppennummer
Es sind lediglich Abgaben im PDF Format gultig. Bitte geben Sie nur **ein** Dokument mit dem Namen "sol3_xx.pdf" ab, das alle Lösungen für das dritte Aufgabenblatt beinhaltet. Die Übung findet in den bereits beim ersten Aufgabenblatt registrierten 3er-Gruppen statt.

Allgemeine Hinweise:

- EXP-Ausdrücke werden zur besseren Lesbarkeit in `Festbreitenschrift` statt unterstrichen geschrieben.
- Die Beweise müssen in der Form, wie sie in der Vorlesung präsentiert wurden, durchgeführt werden. (vollständige bzw. strukturelle Induktion).
- Beispiele mit konkreten Werten stellen alleine keinen allgemeinen Beweis dar und werden zu 0 Punkten auf das jeweilige Beispiel führen.
- Wenn Sie Annahmen treffen, müssen diese kurz beschrieben werden.
- Achten Sie darauf, welche Funktionen und Prädikate verwendet werden dürfen. Sollten Sie Hilfsfunktionen schreiben bzw. Lemmata verwenden, muss auch deren Korrektheit gezeigt werden!

Beispiel 1 (3.0 P.)

Gegeben sei der Datentyp der *rationalen Zahlen*, der folgendermaßen definiert ist:

- Datentyp: $Q = (A, f_1, f_2, p_1)$
- Wertebereich: $A = \mathbb{Z} \times (\mathbb{Z} \setminus \{0\})$, Paare (n, d) aus ganzen Zahlen, wobei der Nenner nicht Null sein darf
- Funktionen:
  1. f_1: mult : $mult((a, b), (c, d)) = (a \cdot c, b \cdot d)$
     Symbol auf syntaktischer Ebene: `mult`
  2. f_2: inverse : $inverse((n, d)) = (d, n)$ wenn $n \neq 0$ sonst undefiniert
     Symbol auf syntaktischer Ebene: `inverse`
- Prädikate:
  1. p_1: equal : $eq?((a, b), (c, d)) \leftrightarrow \frac{a}{b} = \frac{c}{d}$
  2. Symbol auf syntaktischer Ebene: `eq?`
- Konstanten: zur Vereinfachung des Beispiels nicht vorhanden, für die Lösung können auf semantischer Ebene alle rationalen Zahlen verwendet werden

Bestimmen Sie für die nachfolgend angegebene Codierung der rationalen Zahlen in den Datentyp der *Listen + Integer*, für welche codierten Funktionen/Prädikate die Codierung gültig ist. Beweisen Sie für

gültige Codierungen, dass die Codierungseigenschaften erfüllt sind, beziehungsweise geben Sie für nicht gültige Codierungen ein Beispiel an, dass die Codierungseigenschaften verletzt.

Codierung für Elemente aus A:

$$\pi((n,d)) = build(n, build(d,[])) = [n,d]$$

Codierung der Funktionen:

$$\pi[mult](x,y) = build(first(rest(x)) \cdot first(rest(y)), build(first(x) \cdot first(y),[]))$$
$$\pi[inverse](x) = build(first(rest(x)), build(first(x),[])) \dots \text{ wenn } first(x) \neq 0$$

Codierung des Prädikats:

$$\pi[eq](x,y) = eq?(x,y)$$

Beispiel 2 (2.5 P.)

Gegeben sei der Datentyp *IntegerSet*, mit dem es möglich ist, Mengen von Ganzzahlen zu speichern. Definiert ist dieser Datentyp wie folgt:

- Datentyp: $IntegerSet = (A, f_1, f_2, p_1, p_2, c_1)$,
- Notation auf semantischer Ebene: Zur Unterscheidung von Listen verwenden wir zur Darstellung von Mengen geschwungene Klammern statt eckigen Klammern. Die leere Mengen soll durch $\{\}$ und eine Mengen mit n Elementen durch $\{i_1, \dots, i_n\}$ dargestellt werden.
- Wertebereich A: Der Wertebereich kann induktiv definiert werden:
  $A = \{\{\}\} \cup \{insert(s,i) | s \in A, i \in \mathbb{Z}\}$ Die leere Mengen ist also eine Mengen und jede Mengen kann mittels der *insert*-Funktion aus anderen Mengen erzeugt werden.
- Funktionen:
  1. f_1: $insert(s,i) = s'$ mit $s, s' \in A$ und $i \in \mathbb{Z}$
     - Fügt ein Element in die Menge ein
     - $insert(\{\}, i) = \{i\}$
     - $insert(\{i_1, \dots, i_n\}, i_{n+1}) = \{i_1, \dots, i_n, i_{n+1}\}$
     - $insert(\{i_1, \dots, i, \dots i_n\}, i) = \{i_1, \dots, i, \dots i_n\}$
     - Symbol auf syntaktischer Ebene: `insert`
  2. f_2: $remove(s,i) = s'$ mit $s, s' \in A$
     - Löscht das nächste Element
     - $remove(\{\}, i) = \{\}$
     - $remove(\{i_1, \dots, i_k, \dots i_n\}, i_k) = \{i_1 \dots i_{k-1}, i_{k+1}, \dots, i_n\}$
     - $remove(\{i_1, \dots, i_n\}, i_{n+1}) = \{i_1, \dots, i_n\}$
     - Symbol auf syntaktischer Ebene: `remove`
- Prädikate:
  1. p_1: $isEmpty?(s) = b$ mit $q \in A$ und $b \in \{T, F\}$
     - Ist wahr, wenn die leere Menge übergeben wird
     - Wenn $s = \{\}$ dann $isEmpty?(s) = T$
     - Sonst $isEmpty?(s) = F$
     - Symbol auf syntaktischer Ebene: `isEmpty?`
  2. p_2: $isElement?(s,i) = b$ mit $q \in A$ und $b \in \{T, F\}$
     - Ist wahr, wenn i in s enthalten ist.
     - Wenn $s = \{\dots i \dots\}$ dann $isElement?(s,i) = T$

  - Sonst $isElement?(s) = F$
  - Symbol auf syntaktischer Ebene: `isElement?`

- Konstanten:

  1. c_1: {} für die leere Menge + Konstanten aus dem Datentyp Integer
  2. Symbol auf syntaktischer Ebene: `emptyS`

Geben Sie eine Codierung des Datentyps *IntegerSet* in den Datentyp *Listen + Integer* an.

Beispiel 3 (4.0 P.)

Gegeben ist folgendes AL-Programm. Hier steht α für ein beliebiges Statement.

```
1 begin
2   begin
3     x := add(1,2);
4     if eq?(x,3) then
5       α
6     else
7       x := add(x,2)
8   end;
9   α
10 end
```

(a) Ersetzten Sie α durch das konkrete Statement `a := a + a`. Verwenden Sie die Interpretationsfunktion für AL-Programme, um das Programm auszuwerten. Der Wert von `a` soll anfänglich 42 sein. Geben Sie auch die Zwischenschritte sowie das resultierende ω Environment an.

(b) Beweisen Sie unter Verwendung der Interpretationsfunktion für AL-Programme:
Das oben stehende AL-Programm ist für alle α ident mit dem folgenden AL-Programm.

```
1 begin
2   x := 3;
3   begin
4     α;
5     α
6   end
7 end
```

Beispiel 4 (3.0 P.)

Interpretieren Sie die gegebenen prädikatenlogischen Ausdrücke über dem angegebenen Datentyp und bestimmen Sie deren semantischen Status (gültig, erfüllbar, unerfüllbar). Geben Sie die Zwischenschritte für I_{PL} an, sowie die Belegung durch die ω Environments. Die Interpretation der Terme (I_T) dürfen Sie abkürzen. Beachten Sie, dass für die Operatoren folgende Ordnung der Bindungsstärke (= operator precedence) gilt:

- $\neg$
- $\wedge$
- $\vee$
- $\rightarrow$

(a) Datentyp der ganzen Zahlen
$\exists x \forall y \; eq?(add(x, y), 0)$

(b) Datentyp der Listen
$\exists a \; \neg empty?(l) \rightarrow eq?(first(l), a)$

(c) Datentyp der ganzen Zahlen
$\forall x \; eq?(mult(x, 2), y) \vee gt?(mult(x, x), y)$

Viel Erfolg!