Marco Kunze (`makunze@cs.tu-berlin.de`) **SS 2002**
Sebastian Nowozin (`nowozin@cs.tu-berlin.de`) **07.07.2002**

Lösungen zum Zusatz-. Übungsblatt TheGI2

Lösung zu Aufgabe 1:

(a) $T_{Keller/ker(eval(A))}$:

- Sorten:

$$\left(T_{Keller/ker(eval(A))}\right)_{Data} = \left\{ [t_d]_{ker(eval(A))} \middle| t_d \in A_{Data} \right\}$$

$$\left(T_{Keller/ker(eval(A))}\right)_{Stack} = \left\{ [t_s]_{ker(eval(A))} \middle| t_s \in A_{Stack} \right\}$$

- Operationen:

$$d0_{T_{Keller/ker(eval(A))}} = [d0_A]_{ker(eval(A))}$$

$d1$ bis $d9$ sind analog definiert.

$$empty_{T_{Keller/ker(eval(A))}} = [empty_A]_{ker(eval(A))}$$

Für die folgenden Definitionen gilt gemeinsam: $t_d \in A_{Data}$ und $t_s \in A_{Stack}$.

$$push_{T_{Keller/ker(eval(A))}}\left([t_d]_{ker(eval(A))}, [t_s]_{ker(eval(A))}\right) = [push_A(t_d, t_s)]_{ker(eval(A))}$$

$$pop_{T_{Keller/ker(eval(A))}}\left([t_s]_{ker(eval(A))}\right) = [pop_A(t_s)]_{ker(eval(A))}$$

$$clean_{T_{Keller/ker(eval(A))}}\left([t_s]_{ker(eval(A))}\right) = [clean_A(t_s)]_{ker(eval(A))}$$

$$top_{T_{Keller/ker(eval(A))}}\left([t_s]_{ker(eval(A))}\right) = [top_A(t_s)]_{ker(eval(A))}$$

(b)

$$\left(ker(eval(A))|_R\right)_{Data} = \{0,1,2,3,4,5,6,7,8,9\}$$

$$\left(ker(eval(A))|_R\right)_{Stack} = \{0,1,2,3,4,5,6,7,8,9\}^*$$

(c)

- $r : R \to T_{Keller/ker(eval(A))}$

$$r_{Data}(r_d) = [r_d]_{ker(eval(A))} \text{ mit } r_d \in R_{Data}$$

$$r_{Stack}(r_s) = [r_s]_{ker(eval(A))} \text{ mit } r_s \in R_{Stack}$$

r ist bijektiv, also linkseindeutig und rechtstotal. Rechtstotal, da alle Äquivalenzklassen erreicht werden:

$$\forall [r_d]_{ker(eval(A))} \in \left(T_{Keller/ker(eval(A))}\right)_{Data} : \exists p_d \in R_{Data} : r_{Data}(p_d) = [r_d]_{ker(eval(A))}$$

und

$$\forall [r_s]_{ker(eval(A))} \in \left(T_{Keller/ker(eval(A))}\right)_{Stack} : \exists p_s \in R_{Stack} : r_{Stack}(p_s) = [r_s]_{ker(eval(A))}$$

Linkseindeutig, da R_{Data} und $\left(T_{Keller/ker(eval(A))}\right)_{Data}$, beziehungsweise R_{Stack} und $\left(T_{Keller/ker(eval(A))}\right)_{Stack}$ gleich mächtig sind, und direkt elementweise aufeinander abgebildet werden können. Dies ist durch die Definition ersichtlich. Also ist r bijektiv.

- $norm : T_{Keller} \to R$, $norm : r^{-1} \circ nat(T_{Keller}, ker(eval(A)))$

$$norm_{Data}(t_d) = r_d,\ \text{mit } r_d \in R_{Data} \text{ und } r_d \in [t_d]_{ker(eval(A))}$$

$$norm_{Stack}(t_s) = r_s,\ \text{mit } r_s \in R_{Stack} \text{ und } r_s \in [t_s]_{ker(eval(A))}$$

  Die Operationen definieren sich daraus folgend durch $norm$. Es sei im folgenden $t_d \in (T_{Keller})_{Data}$ und $t_s \in (T_{Keller})_{Stack}$.
  - $d0_R = norm_{Data}(d0) = d0$
  - $d1_R$ bis $d9_R$ analog
  - $empty_R = norm_{Stack}(empty) = empty$
  - $push_R(t_d, t_s) = norm_{Stack}(push(t_d, t_s)) = push(t_d, t_s)$
  - $pop_R(t_s) = norm_{Stack}(pop(t_s)) = pop(t_s)$
  - $clean_R(t_s) = norm_{Stack}(clean(t_s)) = clean(t_s)$
  - $top_R(t_s) = norm_{Stack}(top(t_s)) = r_d$, mit $r_d \in R_{Data}$ und $(r_d, top(t_s)) \in ker(eval(A))$

(d) $norm$ lässt sich alternativ wie folgt definieren:

$$norm = r^{-1} \circ nat(T_{Keller}, ker(eval(A)))$$

Da r bijektiv ist, gibt es einen Umkehrhomomorphismus r^{-1}.

- $d0_R = norm_{Data}(d0) = r^{-1} \circ nat(T_{Keller}, ker(eval(A)))(d0) = r^{-1}([d0]_{ker(eval(A))}) = d0$
- analog für $d1_R$ bis $d9_R$ und $empty_R$
- $push_R(t_d, t_s) = norm_{Stack}(push(t_d, t_s)) = r^{-1} \circ nat(T_{Keller}, ker(eval(A)))(push(t_d, t_s)) = r^{-1}([push(t_d, t_s)]_{ker(eval(A))}) = r_s$, mit $r_s \in R_{Stack}$ und $(r_s, push(t_d, t_s)) \in ker(eval(A))$
- analog für die anderen Operationen

Lösung zu Aufgabe 2:

Es genügen drei Gleichungen, um die Eindeutigkeit zu erreichen. Im folgenden sei $s \in (T_{Keller})_{Stack}$ und $d \in (T_{Keller})_{Data}$.

$$\begin{array}{lcl} \text{(E1)} & clean(s) & = s \\ \text{(E2)} & top(push(d, s)) & = d \\ \text{(E3)} & pop(push(d, s)) & = s \end{array}$$

Lösung zu Aufgabe 3:

(a) A ist operationserzeugt, also $eval(A) : T_{Keller} \to A$ ist surjektiv (no-junk Prinzip).

Beweis:

- Data:

$$\{eval(A)(d1), eval(A)(d2), \ldots, eval(A)(d9)\} = \{d1_A, d2_A, \ldots, d9_A\} = A_{Data}$$

- Stack: Sei $s \in (T_{Keller})_{Stack}$. Wir Beweisen mittels vollständiger Induktion über die Länge n von s.
  - Induktionsanfang: $n = 0 \Rightarrow s = \lambda$
    Also für $t = empty \Rightarrow eval(A)_{Stack}(t) = empty_A = \lambda$.
  - Induktionsschritt:

* Induktionsvoraussetzung: $\forall s \in A_{Stack}$ mit Länge n:

$$\exists t \in (T_{Keller})_{Stack} : eval(A)_{Stack}(t) = s$$

* Induktionsbehauptung: $\forall s' \in A_{Stack}$ mit Länge $n+1$:

$$\exists t' \in (T_{Keller})_{Stack} : eval(A)_{Stack}(t') = s'$$

* Induktionsbeweis: Sei $s' = sa$ mit $a \in A_{Data}$, $s \in A_{Stack}$.

$$\begin{array}{ll} \Rightarrow & s \text{ hat die Laenge } n \\ \overset{IV}{\Rightarrow} & \exists t \in (T_{Keller})_{Stack} : eval(A)_{Stack}(t) = s \quad (1) \\ \Rightarrow & \exists d \in (T_{Keller})_{Data} : eval(A)_{Data}(d) = a \quad (2) \end{array}$$

Sei $t = push(d, t)$.

$$\Rightarrow eval(A)_{Stack}(t') = push_A(eval(A)_{Data}(d), eval(A)_{Stack}(s)) \overset{1+2}{=} push_A(a, s) \overset{Def.A}{=} sa = s'$$

Damit ist die Behauptung bewiesen.

(b) $A \models \sim^E$. Also $\sim^E \subseteq ker(eval(A))$.

Beweis:

- E1.

$$\begin{array}{rcl} eval(A)(clean(s)) & = & eval(A)(s) \\ \Leftrightarrow clean_A(eval(A)(s)) & = & eval(A)(s) \\ \Leftrightarrow eval(A)(s) & = & eval(A)(s) \end{array}$$

- E2.

$$\begin{array}{rcl} eval(A)(top(push(d, s))) & = & eval(A)(d) \\ \Leftrightarrow top_A(eval(push(d, s))) & = & eval(A)(d) \\ \Leftrightarrow top_A(push_A(eval(A)(d), eval(A)(s))) & = & eval(A)(d) \\ \Leftrightarrow top_A(eval(A)(s)eval(A)(d)) & = & eval(A)(d) \\ \Leftrightarrow eval(A)(d) & = & eval(A)(d) \end{array}$$

- E3.

$$\begin{array}{rcl} eval(A)(pop(push(d, s))) & = & eval(A)(s) \\ \Leftrightarrow pop_A(eval(A)(push(d, s))) & = & eval(A)(s) \\ \Leftrightarrow pop_A(push_A(eval(A)(d), eval(A)(s))) & = & eval(A)(s) \\ \Leftrightarrow pop_A(eval(A)(s)eval(A)(d)) & = & eval(A)(s) \\ \Leftrightarrow eval(A)(s) & = & eval(A)(s) \end{array}$$

Also $A \models \sim^E$.

(c) (i) A. $\forall t \in T_{Keller} \exists r \in R : t \sim^E r$

- $t_d \in (T_{Keller})_{Data}$:
  Fallunterscheidung:
  - $t_d \in \{d0, d1, \ldots, d9\}$
    Dann $t_d \sim^E_{Data} r$ für $r = eval(A)(t_d)$.
  - $t_d \in \{top_R(t_s)\}$
    Dann $t_d \sim^E_{Data} r$ für $r = eval(A)(t_d)$.

- $t_s \in (T_{Keller})_{Stack}$:
  <u>Fallunterscheidung:</u>
  - $t_s \in \{empty\}$
    Dann $t_s \sim^E_{Stack}$ für $r = empty_R$.
  - $t_s \in \{push(t_d, t_t) | t_d \in (T_{Keller})_{Data}, t_t \in (T_{Keller})_{Stack}\}$
    Es gilt (siehe Aufgabe 1c):

    $$r = norm(push(t_d, t_t)) = push_R(t_d, t_t)$$

    Dann ist für dieses r: $t_s \sim^E_{Stack} r$
  - $t_s \in \{clean(t_t) | t_t \in (T_{Keller})_{Stack}\}$ und für $r \stackrel{1c}{=} clean_R(t_t) \stackrel{E1}{=} empty_R$:

    $$t_s \sim^E_{Stack} r$$

  - $t_s \in \{pop(t_t) | t_t \in (T_{Keller})_{Stack}\}$ und für $r \stackrel{1c}{=} pop_R(t_t)$:

    $$t_s \sim^E_{Stack} r$$

B. (???) (Macht für uns keinen Sinn, da $R \approx A$ ist (bis auf Umbenennung stimmen die Repräsentantenmengen mit den Mengen aus A überein). Daher ist das zu zeigende für feste Repräsentanten trivial, da $r_1 = r_2 \Rightarrow r_1 \sim^E r_2$ per Definition gilt. r_1, r_2 sind ja feste Werte, keine Terme, daher muss $\sim^E$ ihre Gleichheit kennen. Die Aufgabenstellung ist für uns nicht klar.)

(ii) Wir wissen, dass r bijektiv ist, also umkehrbar ist, und r^{-1} damit auch surjektiv ist. i ist bijektiv, und es gilt:

$$i = eval(A)|_R \circ r^{-1}$$

Mit Definition 8.5.7 (Epi-Mono-Faktorisierung eines Homomorphismus) ist damit der eingeschränkte Auswertungshomomorphismus $eval(A)|_R$ injektiv.

Es gilt also $T_{Keller/\sim^E} \simeq A$.