Formale Techniken in der Software-Entwicklung: Besprechung Blatt 1

Christian Kroiß

27. April 2012

Nachtrag: Struktur...

Definition: Struktur

Sei $\Sigma = (S, F, P)$ eine mehrsortige Signatur mit einer Menge S an Sorten, einer $S^* \times S$-sortierten Familie F an *Funktionssymbolen* und einer S^*-sortierten Familie P an *Prädikaten*.

Eine Σ-Struktur A besteht aus

- Einer Familie $(A_s)_{s \in S}$ von nicht-leeren Trägermengen.
- Funktionen $f^A : A_{s1} \times \ldots \times A_{sn} \to A_s$ für alle $f \in F_{(s1,\ldots,sn,s)}$.
- Relationen $p^A \subseteq A_{s1} \times \ldots \times A_{sn}$ für alle $p \in P_{(s1,\ldots,sn)}$.

Nachtrag: Struktur...

Definition: Struktur

Sei $\Sigma = (S, F, P)$ eine mehrsortige Signatur mit einer Menge S an Sorten, einer $S^* \times S$-sortierten Familie F an *Funktionssymbolen* und einer S^*-sortierten Familie P an *Prädikaten*.

Eine Σ-Struktur A besteht aus

- Einer Familie $(A_s)_{s \in S}$ von nicht-leeren Trägermengen.
- Funktionen $f^A : A_{s1} \times \ldots \times A_{sn} \to A_s$ für alle $f \in F_{(s1,\ldots,sn,s)}$.
- Relationen $p^A \subseteq A_{s1} \times \ldots \times A_{sn}$ für alle $p \in P_{(s1,\ldots,sn)}$.

Eine Struktur ist also eine Algebra mit Prädikaten.

Aufgabe 1-a) Signatur

Es ist $MAP0 = (S, F, P)$ mit Sorten $S = \{Map, Index, Data\}$, Operationenmengen

$$\begin{aligned} F_{\langle\langle\rangle, Map\rangle} &= \{empty\} \\ F_{\langle\langle Map, Index, Data\rangle, Map\rangle} &= \{update\} \\ F_{\langle\langle Map, Index\rangle, Data\rangle} &= \{get\} \\ F_{\langle\langle Map, Index\rangle, Map\rangle} &= \{remove\} \\ F_{\langle w, s\rangle} &= \emptyset \qquad \text{sonst} \end{aligned}$$

und Prädikaten

$$\begin{aligned} P_{\langle Map, Index\rangle} &= \{isdef\} \\ P_w &= \emptyset \qquad \text{sonst} \end{aligned}$$

Aufgabe 1-b): Struktur Variante 1: Mengen von Tupeln

- Trägermengen:
  - $A_{Index} = \mathbb{N}$
  - A_{Data} = Eine Menge von beliebigen Objekten
  - $A_{Map} = \mathcal{P}(A_{Index} \times A_{Data})$
    (Potenzmenge von $A_{Index} \times A_{Data}$, d.h. jedes Element von A_{Map} ist eine Menge von Tupeln aus $A_{Index} \times A_{Data}$.)

- Funktionen:

Aufgabe 1-b): Struktur Variante 1: Mengen von Tupeln

- Trägermengen:
  - $A_{Index} = \mathbb{N}$
  - A_{Data} = Eine Menge von beliebigen Objekten
  - $A_{Map} = \mathcal{P}(A_{Index} \times A_{Data})$
    (Potenzmenge von $A_{Index} \times A_{Data}$, d.h. jedes Element von A_{Map} ist eine Menge von Tupeln aus $A_{Index} \times A_{Data}$.)

- Funktionen:
  - $empty^A = \emptyset$

Aufgabe 1-b): Struktur Variante 1: Mengen von Tupeln

- Trägermengen:
  - $A_{Index} = \mathbb{N}$
  - A_{Data} = Eine Menge von beliebigen Objekten
  - $A_{Map} = \mathcal{P}(A_{Index} \times A_{Data})$
    (Potenzmenge von $A_{Index} \times A_{Data}$, d.h. jedes Element von A_{Map} ist eine Menge von Tupeln aus $A_{Index} \times A_{Data}$.)

- Funktionen:
  - $empty^A = \emptyset$
  - $update^A(m, i, d) = remove^A(m, i) \cup \{(i, d)\}$

Aufgabe 1-b): Struktur Variante 1: Mengen von Tupeln

- Trägermengen:
  - $A_{Index} = \mathbb{N}$
  - A_{Data} = Eine Menge von beliebigen Objekten
  - $A_{Map} = \mathcal{P}(A_{Index} \times A_{Data})$
    (Potenzmenge von $A_{Index} \times A_{Data}$, d.h. jedes Element von A_{Map} ist eine Menge von Tupeln aus $A_{Index} \times A_{Data}$.)

- Funktionen:
  - $empty^A = \emptyset$
  - $update^A(m, i, d) = remove^A(m, i) \cup \{(i, d)\}$
  - $get^A(m, i) = \begin{cases} undefined & \text{falls } \neg\exists d \in A_{Data}.\ (i, d) \in m \\ d & \text{falls } (i, d) \in m \end{cases}$

Aufgabe 1-b): Struktur Variante 1: Mengen von Tupeln

- Trägermengen:
  - $A_{Index} = \mathbb{N}$
  - A_{Data} = Eine Menge von beliebigen Objekten
  - $A_{Map} = \mathcal{P}(A_{Index} \times A_{Data})$
    (Potenzmenge von $A_{Index} \times A_{Data}$, d.h. jedes Element von A_{Map} ist eine Menge von Tupeln aus $A_{Index} \times A_{Data}$.)

- Funktionen:
  - $empty^A = \emptyset$
  - $update^A(m, i, d) = remove^A(m, i) \cup \{(i, d)\}$
  - $get^A(m, i) = \begin{cases} undefined & \text{falls } \neg\exists d \in A_{Data}.\ (i, d) \in m \\ d & \text{falls } (i, d) \in m \end{cases}$
  - $remove^A(m, i) = \{(i', d) \in m \mid i' \neq i\}$

Aufgabe 1-b): Struktur Variante 1: Mengen von Tupeln

- Trägermengen:
  - $A_{Index} = \mathbb{N}$
  - A_{Data} = Eine Menge von beliebigen Objekten
  - $A_{Map} = \mathcal{P}(A_{Index} \times A_{Data})$
    (Potenzmenge von $A_{Index} \times A_{Data}$, d.h. jedes Element von A_{Map} ist eine Menge von Tupeln aus $A_{Index} \times A_{Data}$.)

- Funktionen:
  - $empty^A = \emptyset$
  - $update^A(m, i, d) = remove^A(m, i) \cup \{(i, d)\}$
  - $get^A(m, i) = \begin{cases} undefined & \text{falls } \neg\exists d \in A_{Data}.\ (i, d) \in m \\ d & \text{falls } (i, d) \in m \end{cases}$
  - $remove^A(m, i) = \{(i', d) \in m \mid i' \neq i\}$
- Relation:
  $isdef^A$ = $\{(m, i) \mid m$ enthält ein Paar (i, d) für ein $d \in A_{Data}\}$

Aufgabe 1-b): Struktur Variante 2: Sequenz von Tupeln

- Trägermengen:
  - $A_{Index} = \mathbb{N}$
  - A_{Data} = Eine Menge von beliebigen Objekten
  - $A_{Map} = \{\langle (i_1, d_1), \ldots, (i_n, d_n) \rangle \mid i_1, \ldots, i_n \in A_{Index}, d_1, \ldots, d_n \in A_{Data}, n \in \mathbb{N}_0\}$
- Funktionen:

Aufgabe 1-b): Struktur Variante 2: Sequenz von Tupeln

- Trägermengen:
  - $A_{Index} = \mathbb{N}$
  - $A_{Data} =$ Eine Menge von beliebigen Objekten
  - $A_{Map} = \{\langle (i_1, d_1), \ldots, (i_n, d_n) \rangle \mid i_1, \ldots, i_n \in A_{Index}, d_1, \ldots, d_n \in A_{Data}, n \in \mathbb{N}_0\}$
- Funktionen:
  - $empty^A = \langle\rangle$

Aufgabe 1-b): Struktur Variante 2: Sequenz von Tupeln

- Trägermengen:
  - $A_{Index} = \mathbb{N}$
  - $A_{Data} =$ Eine Menge von beliebigen Objekten
  - $A_{Map} = \{\langle (i_1, d_1), \ldots, (i_n, d_n) \rangle \mid i_1, \ldots, i_n \in A_{Index}, d_1, \ldots, d_n \in A_{Data}, n \in \mathbb{N}_0\}$
- Funktionen:
  - $empty^A = \langle\rangle$
  - $update^A(\langle (i_1, d_1), \ldots, (i_n, d_n) \rangle, i, d) = \langle (i, d), (i_1, d_1), \ldots, (i_n, d_n) \rangle$

Aufgabe 1-b): Struktur Variante 2: Sequenz von Tupeln

- Trägermengen:
  - $A_{Index} = \mathbb{N}$
  - A_{Data} = Eine Menge von beliebigen Objekten
  - $A_{Map} = \{\langle (i_1, d_1), \ldots, (i_n, d_n) \rangle \mid i_1, \ldots, i_n \in A_{Index}, d_1, \ldots, d_n \in A_{Data}, n \in \mathbb{N}_0\}$
- Funktionen:
  - $empty^A = \langle\rangle$
  - $update^A(\langle (i_1, d_1), \ldots, (i_n, d_n) \rangle, i, d) = \langle (i, d), (i_1, d_1), \ldots, (i_n, d_n) \rangle$
  - $get^A(\langle (i_1, d_1), \ldots, (i_n, d_n) \rangle, i) =$

$$\begin{cases} undefined & \text{falls } n = 0 \\ d_1 & \text{falls } i_1 = i \\ get^A(\langle (i_2, d_2), \ldots, (i_n, d_n) \rangle, i) & \text{sonst} \end{cases}$$

Aufgabe 1-b): Struktur Variante 2: Sequenz von Tupeln

- Trägermengen:
  - $A_{Index} = \mathbb{N}$
  - A_{Data} = Eine Menge von beliebigen Objekten
  - $A_{Map} = \{\langle (i_1, d_1), \ldots, (i_n, d_n) \rangle \mid i_1, \ldots, i_n \in A_{Index}, d_1, \ldots, d_n \in A_{Data}, n \in \mathbb{N}_0\}$
- Funktionen:
  - $empty^A = \langle\rangle$
  - $update^A(\langle (i_1, d_1), \ldots, (i_n, d_n) \rangle, i, d) = \langle (i, d), (i_1, d_1), \ldots, (i_n, d_n) \rangle$
  - $get^A(\langle (i_1, d_1), \ldots, (i_n, d_n) \rangle, i) =$

$$\begin{cases} undefined & \text{falls } n = 0 \\ d_1 & \text{falls } i_1 = i \\ get^A(\langle (i_2, d_2), \ldots, (i_n, d_n) \rangle, i) & \text{sonst} \end{cases}$$

  - $remove^A(\langle (i_1, d_1), \ldots, (i_n, d_n) \rangle, i) =$

$$\begin{cases} \langle\rangle & \text{falls } n = 0 \\ \langle (i_2, d_2), \ldots, (i_n, d_n) \rangle & \text{falls } i_1 = i \\ \langle (i_1, d_1) \rangle + remove^A(\langle (i_2, d_2), \ldots, (i_n, d_n) \rangle, i) & \text{sonst.} \end{cases}$$

Aufgabe 1-b): Struktur Variante 2: Sequenz von Tupeln

- Trägermengen:
  - $A_{Index} = \mathbb{N}$
  - A_{Data} = Eine Menge von beliebigen Objekten
  - $A_{Map} = \{\langle (i_1, d_1), \ldots, (i_n, d_n)\rangle \mid i_1, \ldots, i_n \in A_{Index}, d_1, \ldots, d_n \in A_{Data}, n \in \mathbb{N}_0\}$
- Funktionen:
  - $empty^A = \langle\rangle$
  - $update^A(\langle (i_1, d_1), \ldots, (i_n, d_n)\rangle, i, d) = \langle (i, d), (i_1, d_1), \ldots, (i_n, d_n)\rangle$
  - $get^A(\langle (i_1, d_1), \ldots, (i_n, d_n)\rangle, i) =$
    $$\begin{cases} undefined & \text{falls } n = 0 \\ d_1 & \text{falls } i_1 = i \\ get^A(\langle (i_2, d_2), \ldots, (i_n, d_n)\rangle, i) & \text{sonst} \end{cases}$$
  - $remove^A(\langle (i_1, d_1), \ldots, (i_n, d_n)\rangle, i) =$
    $$\begin{cases} \langle\rangle & \text{falls } n = 0 \\ \langle (i_2, d_2), \ldots, (i_n, d_n)\rangle & \text{falls } i_1 = i \\ \langle (i_1, d_1)\rangle + remove^A(\langle (i_2, d_2), \ldots, (i_n, d_n)\rangle, i) & \text{sonst.} \end{cases}$$
- Relation:
  $isdef^A = \{(m, i) \mid m \text{ enthält ein Paar } (i, d) \text{ für ein } d \in A_{Data}\}$

Aufgabe 1: Anmerkung

Ein charakteristischer Unterschied zwischen den beiden MAP0-Algebren ist, dass in der ersten die Formel

$$i \neq j \vee x = y \Rightarrow$$
$$update(update(m, i, x), j, y) = update(update(m, j, y), i, x)$$

gilt, in der zweiten aber nicht.

Aufgabe 2: Signatur

```
fmod PARAFFIN is
   sorts Radikal Paraffin .
   op H : -> Radikal [ctor] .
   op rad : Radikal Radikal Radikal -> Radikal [ctor] .
   op para : Radikal Radikal Radikal Radikal -> Paraffin [ctor] .
endfm
```

(Kohlenstoffatome sind implizit!)

Aufgabe 2: Beispiele

Methan (CH_4): `para(H,H,H,H)`

Butan (C_4H_{10}):

```
para(H,H,H,
   rad(H,H,
      rad(H,H,
         rad(H,H,H))))
```

Iso-Butan:

```
para(H,H,H,
   rad(H,
      rad(H,H,H),
      rad(H,H,H)))
```

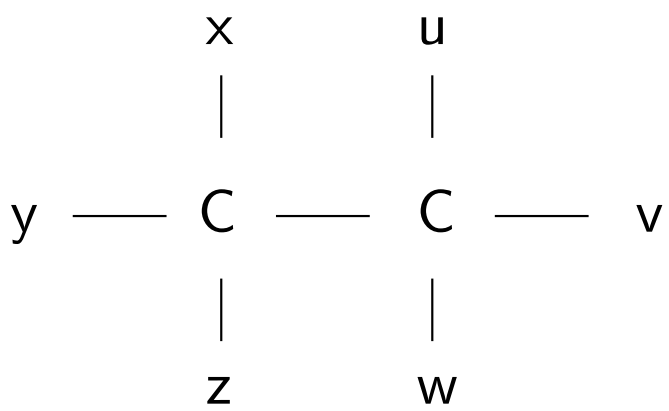
Alternative für Butan:

```
para(rad(H,H,H),
   H, H,
   rad(H, rad(H,H,H), H))
```

Aufgabe 2: Äquivalenz-Klassen

Die folgenden Gesetze präzisieren die Bedingung, dass die vier Valenzen eines Kohlenstoffatoms innerhalb eines Paraffins ununterscheidbar sind.

(UMK)	$para(x, y, z, rad(u, v, w))$	Vertauschen von Paraffin-
	$= para(rad(x, y, z), u, v, w)$	und Radikalbindung
(ROT)	$para(u, v, w, x) = para(v, w, x, u)$	Rotation der Radikale
(SWP)	$para(u, v, w, x) = para(v, u, w, x)$	Vertauschen von Radikalen



Aufgabe 2: Äquivalenz-Beweis

Mit diesen Gesetzen kann die oben behauptete Äquivalenz nachgewiesen werden:

$$
\begin{aligned}
& \text{para(H, H, H, rad(H, H, rad(H, H, rad(H, H, H))))} && \\
= \; & \text{para(rad(H, H, H), H, H, rad(H, H, rad(H, H, H)))} && \text{(UMK)} \\
= \; & \text{para(rad(rad(H, H, H), H, H), H, H, rad(H, H, H))} && \text{(UMK)} \\
= \; & \text{para(H, H, rad(H, H, H), rad(rad(H, H, H), H, H))} && \text{(ROT)} \\
= \; & \text{para(H, rad(H, H, H), rad(rad(H, H, H), H, H), H)} && \text{(ROT)} \\
= \; & \text{para(rad(H, H, H), H, rad(rad(H, H, H), H, H), H)} && \text{(SWP)} \\
= \; & \text{para(H, rad(rad(H, H, H), H, H), H, rad(H, H, H))} && \text{(ROT)} \\
= \; & \text{para(rad(rad(H, H, H), H, H), H, rad(H, H, H), H)} && \text{(ROT)} \\
= \; & \text{para(rad(H, H, H), H, H, rad(H, rad(H, H, H), H))} && \text{(UMK)}
\end{aligned}
$$

Die Rotations- und Vertauschungsgesetze könnten zusätzlich auch für Radikale formuliert werden, dann hätte man in obigem Beweis auf das doppelte Umklammern verzichten können.

Tipps: Makros für Schreibfaule

```
CH3 : -> Radikal .
CH2 : Radikal -> Radikal .
CH3p : Radikal -> Paraffin .
Methan Butan IsoButan Butan2 : -> Paraffin .
eq CH3 = rad(H, H, H) .
eq CH3p(R) = para(H, H, H, R) .
var R : Radikal .
eq CH2(R) = rad(H, H, R) .
eq Methan = CH3p(H) .
eq Butan = CH3p(CH2(CH2(CH3))).
```

- Nicht-Konstruktor Konstanten verwenden
- Nicht-Konstruktor Funktionen verwenden
- Gleichungen zur Auflösung der Konstanten und Funktionen definieren