1 Prozeßsysteme und Koordinierung

F. Hofmann: "Betriebssysteme: Grundkonzepte und Modellvorstellungen", 2. Auflage, Leitfäden der angewandten Informatik, B. G. Teubner Stuttgart, 1991.

1.1 Formalisierung des Begriffs "Prozeßsystem"

- ❑ Programmentwicklung erfolgt über mehrere Abstraktionsstufen

  Frage: Welcher Zusammenhang muß zwischen Darstellungen auf unterschiedlichen Abstraktionsebenen bestehen?

- ❑ Antwort in zwei Schritten:
  - Relationale Programme: Gewünscht sind gleiche Endergebnisse
    Typisch sind numerische Berechnungen
  - Reaktive Programme: Gewünscht ist gleiches “Verhalten”
    Typisch sind Gerätesteuerungen (Verkehrssignalanlagen, Roboter)
- ❑ Prozesse bestehen aus der Abarbeitung von atomaren Aktivitäten.
- ❑ Aktivitäten werden - als Verallgemeinerungen von Befehlen - durch eine Nichtblockierungs- (oder Ausführbarkeits-)Bedingung und eine Effektbeschreibung charakterisiert.

- ❑ Beispiel: P-Operation für Semaphor s
  - Nichblockierungsbedingung: $'s > 0$
  - Effektbeschreibung: $s = 's - 1$

- ❑ Beispiel: Spezifikation und Implementation eines endlichen Puffers mit 10 Plätzen


```
CLASS puffer;

  DECLARATIONS
    inhalt: SEQUENCE OF INTEGER;

  INITIALLY
    inhalt = λ;

  OPERATIONS
    send(sinf: INTEGER);
      NBL
        LENGTH(’inhalt) < 10;
      EFFECTS
        inhalt = ’inhalt CONCAT sinf;
    receive() → rinf: INTEGER;
      NBL
        LENGTH(’inhalt) > 0;
      EFFECTS
        rinf = ’inhalt.FIRST()
        AND inhalt = ’inhalt.TAIL();

END_CLASS
```



```
CLASS puffer;

  DECLARATIONS
    ringpuffer:  ARRAY [0..9] OF INTEGER;
    geschrieben: INTEGER;
      /* Zählt die ausgeführten send-Operationen */
    gelesen:     INTEGER;
      /* Zählt die ausgeführten receive-Operationen */

  INITIALLY
    geschrieben = 0 AND gelesen = 0;
```

```
  OPERATIONS
      send(sinf: INTEGER);
         NBL
            'geschrieben - 'gelesen < 10;
         EFFECTS
            FOR_ALL i (0≤i<10 AND i ≠ ('geschrieben % 10)
                  IMPL ringpuffer[i] = 'ringpuffer[i])
            AND ringpuffer['geschrieben % 10] = sinf
            AND geschrieben = 'geschrieben + 1
            AND gelesen = 'gelesen;
      receive() → rinf: INTEGER;
         NBL
            'geschrieben - 'gelesen > 0;
         EFFECTS
            rinf = 'ringpuffer['gelesen % 10]
            AND FOR_ALL i(0≤i<10
                  IMPL ringpuffer[i] = 'ringpuffer[i])
            AND geschrieben = 'geschrieben
            AND gelesen = 'gelesen + 1;
END_CLASS
```


❑ Bezeichnungen

- 'x bzw. x bezeichne den Zustand des Datenbereichs auf der niedrigeren Abstraktionsebene vor bzw. nach Ausführung einer Aktivität.
- 'y bzw. y bezeichne den Zustand des Datenbereichs auf der höheren Abstraktionsebene vor bzw. nach Ausführung einer Aktivität.
- $INIT_h(y)$ bzw. $INIT_n(x)$ beschreibe die Menge der möglichen Startwerte einer Aktion auf der höheren bzw. niedrigeren Abstraktionsebene.
- $N_h('y)$ bzw. $E_h('y, y)$ seien die NBL- bzw. EFFEKT-Beschreibung einer Aktivität der höheren Abstraktionsebene.
- $N_n('x)$ bzw. $E_n('x, x)$ seien die NBL- bzw. EFFEKT-Beschreibung einer Aktivität der niedrigeren Abstraktionsebene.
- rep sei eine Abbildung der Zustände der niedrigeren Abstraktionsebene auf Zustände der höheren.

❑ Eine Aktion der niedrigeren Abstraktionsebene ist eine korrekte Implementation einer Aktion der höheren, wenn sie folgenden Zusammenhang besitzen:

- $\forall x, y\ (INIT_n(x) \wedge rep(x) = y \Rightarrow INIT_h(y))$

  d. h., wenn in der Implementation *x* den Zustand y repräsentiert und x in der Implementation ein zulässiger Anfangszustand ist, dann ist y ein zulässiger Anfangszustand in der Spezifikation.
- $\forall 'x, x, 'y, y\ (\ (rep('x) = 'y) \wedge (rep(x) = y) \wedge E_n('x, x) \Rightarrow E_h('y, y))$

  d. h., wenn 'y bzw. y durch 'x bzw. x repräsentiert wird und eine Aktivität in der Implementation von 'x nach x führen kann, dann kann die zugeordnete Aktivität der Spezifikation von 'y nach y führen. Es wird aber nicht verlangt, daß jeder laut Spezifikation mögliche Übergang auch eine Entsprechung in der Implementation besitzt.
- $\forall x, y\ (\ (rep(x) = y) \Rightarrow (N_n(x) \Leftrightarrow N_h(y))\ )$

  d. h. die Nichtblockierungsbedingung einer Aktivität ist im Zustand x der Implementation genau dann erfüllt, wenn die Nichtblockierungsbedingung der entsprechenden Aktivität der Spezifikation im Zustand `rep(x)` erfüllt ist.

Definition der Funktion rep:

`rep: (ARRAY[0..9] OF INTEGER)× INTEGER × INTEGER → SEQUENCE OF INTEGER`

rep(ringpuffer, gelesen, geschrieben)
$=_{df}$ ringpuffer[gelesen % 10] • ringpuffer[(gelesen+1) % 10] •
... • ringpuffer[(geschrieben-1) % 10]

E_n:

geschrieben = 'geschrieben+1
gelesen = 'gelesen

$$\text{ringpuffer}[i] = \begin{cases} \text{'ringpuffer}[i] & \text{falls } i \neq \text{'geschrieben \% 10} \\ \text{sinf} & \text{falls } i = \text{'geschrieben \% 10} \end{cases}$$

E_h:

inhalt = 'inhalt • sinf

Zu zeigen ist

E_n(ringpuffer, gelesen, geschrieben, 'ringpuffer, 'gelesen, 'geschrieben)
$\rightarrow E_h$(rep(ringpuffer, gelesen, geschrieben), rep('ringpuffer, 'gelesen, 'geschrieben))

rep(ringpuffer, gelesen, geschrieben)
= ringpuffer[gelesen % 10] • ringpuffer[(gelesen+1) % 10] •
... • ringpuffer[(geschrieben-1) % 10]
= ringpuffer['gelesen % 10] • ringpuffer[('gelesen+1) % 10] •
... • ringpuffer[('geschrieben) % 10]
= ringpuffer['gelesen % 10] • ringpuffer[('gelesen+1) % 10] •
... • ringpuffer[('geschrieben-1) % 10] • sinf
= rep('ringpuffer, 'gelesen, 'geschrieben) • sinf

Prozeßsysteme

❑ Bestandteile sequentieller Prozesse:

1. A, eine endliche Menge von Aktionen.
2. $D = D_1 \times D_2$, die Zustandsmenge. Dabei bezeichnet D_1 die Menge der *Aktionenidentifikatoren*, D_2 den Wertevorrat an Datenzuständen.
3. $adr: A \rightarrow D_1$, eine injektive Abbildung, die jeder Aktion einen Identifikator zuordnet.
4. $w \in D$, der Anfangszustand des Prozesses.

❑ Bezeichnungen

- Ein n-Tupel $(x_1, x_2, ..., x_n)$ wird *typisches Element* des kartesischen Produktes $M_1 \times M_2 \times ... \times M_n$ genannt, wenn x_i für alle i mit $1 \le i \le n$ eine Variable mit Wertebereich M_i ist.
- Ist $y = (a_1, a_2, ..., a_n) \in M_1 \times M_2 \times ... \times M_n$ und $(x_1, x_2, ..., x_n)$ typisches Element von $M_1 \times M_2 \times ... \times M_n$, dann bezeichnet $x_i[y]$ den Wert a_i.

- Die einzelnen Komponenten einer Aktion und ihr Identifikator werden durch folgende Schreibweise festgelegt:

  a: WHEN B_a('v) DO v = F_a('v).

  Dabei ist

  a) a der Identifikator der Aktion,
  b) v typisches Element von D und 'v der Wert, den v unmittelbar vor Ausführung der Aktion hatte,
  c) $B_a: D \rightarrow \{TRUE, FALSE\}$ die *Ausführbarkeitsbedingung* der Aktion und
  d) $F_a: D \rightarrow D$ die *Übergangsfunktion*.

  Ist $v = (L, G)$, so wird durch $G[F_a('v)]$ die Änderung des Datenzustandes charakterisiert und durch $L[F_a('v)]$ die Nachfolgeaktion festgelegt. Weiter wird vorausgesetzt, daß $L[F_a('v)] \ne L['v]$ ist für alle Werte von D. Dies entspricht der *generellen Annahme*, daß die Ausführung einer Aktion eine Änderung am Befehlszähler bewirkt.

- Jeder Aktion a wird eine Abbildung $Z_a: D \rightarrow D$ zugeordnet durch die Vorschrift:

$$Z_a(v) = \begin{cases} F_a(v) & B_a(v) = TRUE \\ & \text{falls} \\ v & B_a(v) = FALSE \end{cases} .$$

Definition 1.1

Das Quintupel $P = (A, D, bz, adr, w)$ heißt *Prozeßsystem*, wenn gilt:

1. $D = D_1 \times D_2 \times ... \times D_n \times H$ ist ein endliches, kartesisches Produkt und heißt Zustandsmenge. Im weiteren sei $(L_1, L_2, ..., L_n, G)$ typisches Element von D. L_1 bis L_n bezeichnen die *Befehlszähler* der n Prozesse, aus denen sich das Prozeßsystem zusammensetzt. G bezeichnet den Anteil, der die Zustände aller relevanten Modulinstanzen darstellt.
2. $w \in D$ ist der Anfangszustand des Systems.
3. A ist eine endliche Menge von Aktionen.
4. $bz: A \to \{1, 2, ..., n\}$ ordnet jede Aktion einem Prozeß (Befehlszähler) zu.
5. $adr: A \to \bigcup_{i=1}^{n} D_i$ ordnet jeder Aktion eine Aktionsadresse zu, wobei:
   a) $adr(a) \in D_{bz(a)}$ ist und
   b) aus $adr(a) = adr(b)$ und $bz(a) = bz(b)$ folgt, daß $a = b$ ist (d. h. eine Aktion ist durch Prozeßzugehörigkeit und Aktionsadresse eindeutig bestimmt).

6. Die Aktionen haben die Struktur

   $a_{i,j}$: WHEN $'L_i = j$ AND $B_{i,j}('G)$
   DO $L_i = N_{i,j}('G)$
   AND FORALL $k(1 \le k \le n$ AND $i \ne k$ IMPL $L_k = 'L_k)$
   AND $G = F_{i,j}('G)$

   mit $i = bz(a_{i,j})$ und $j = adr(a_{i,j})$.

   Außerdem wird $N_{i,j}('G) \ne j$ gefordert,
   d. h. jede Aktion bewirkt bei Ausführung eine Änderung des Befehlszählers.

Definition 1.2

1. $TP_i = (A_i, D_i', adr_i', w_i')$ heißt i-ter *(Teil-)Prozeß* von $P = (A, D, bz, adr, w)$, wenn gilt:
   a) $A_i = \{a \mid a \in A \wedge bz(a) = i\}$, $D_i' = D_i \times H$
   b) $adr_i' = adr|A_i$ und
   c) $w_i' = (L_i[w], G[w])$.
2. Ablauf $a \Leftrightarrow a \in A^*$

Definition 1.3

Sei $P = (A, D, bz, adr, w)$ ein Prozeßsystem. Dann ist $val_P: A^* \to D$ definiert durch

$$val_P(\alpha) = \begin{cases} w & \text{falls } \alpha = \lambda \\ Z_{a_{i,j}}(val_P(\beta)) & \text{falls } \alpha = \beta \bullet a_{i,j} \end{cases}.$$

Definition 1.4

Sei $P = (A, D, bz, adr, w)$ ein Prozeßsystem und $(L_1, L_2, ..., L_n, G)$ typisches Element von D. Dann bezeichnet:

1. $proc_P(f, g)$ das Prädikat $f, g \in A \wedge bz(f) = bz(g)$, also die Eigenschaft, daß f und g Aktionen des gleichen Prozesses sind,
2. $bef_P(\alpha)$ die Aktionenmenge $\{a \mid a \in A \wedge adr(a) = L_{bz(a)}[val_P(\alpha)]\}$,

   d. h. $bef_P(\alpha)$ ist die Menge der Aktionen, die nach Ablauf von α zur Ausführung anstehen, wobei in α nicht ausführbare Aktionen als wirkungslos betrachtet werden, und
3. $lf_P(\alpha)$ die Aktionenmenge $\{a \mid a \in A \wedge val_P(\alpha \bullet a) \ne val_P(\alpha)\}$,

   d. h. $lf_P(\alpha)$ ist die Menge der Aktionen, die nach Ablauf von α ausführbar sind, wobei in α nicht ausführbare Aktionen als wirkungslos betrachtet werden.

Satz 1.1

Es sei $P = (A, D, bz, adr, w)$ ein Prozeßsystem und $(L_1, L_2, ..., L_n, G)$ typisches Element von D. A_i sei die Aktionenmenge des i-ten Prozesses. Dann gilt:

1. $\forall\alpha\,(\alpha \in A^* \Rightarrow lf(\alpha) \subseteq bef(\alpha))$
2. $\forall i, \alpha\,(1 \le i \le n \wedge \alpha \in A^* \Rightarrow |bef(\alpha) \cap A_i| \le 1)$
3. $\forall i, \alpha, \beta(1 \le i \le n \wedge \alpha \in A^* \wedge \beta \in \left(\bigcup_{\substack{1 \le j \le n \\ i \ne j}} A_j\right)^* \Rightarrow (bef(\alpha) \cap A_i = bef(\alpha \bullet \beta) \cap A_i))$

Definition 1.5

Ein Ablauf heißt aktiv, wenn gilt:
$\forall i\,(1 \le i \le \text{Länge}(\alpha) \Rightarrow \alpha_i \in lf_P({}_{i-1}\alpha))$.

Satz 1.2

1. $(val_P(\alpha) = val_P(\beta)) \Rightarrow (val_P(\alpha \bullet \gamma) = val_P(\beta \bullet \gamma))$
2. $aktiv(\alpha \bullet f \bullet \beta) \wedge aktiv(\alpha \bullet \beta) \Rightarrow \forall i\,(1 \le i \le \text{Länge}(\beta) \Rightarrow (bz(f) \ne bz(\beta_i)))$