# PNL: Prozesse und Nebenläufigkeit

Korrekturen zum Skript Wintersemester 2001/2002
Stand vom 11.2.2002

**Kapitel 2**

Seite 7, erster Punkt: Jedes $a \in A$ ist ein BPA (*atomare Aktion*).

Seite 13: Da $gew(s_1) > gew(s_2) > \ldots > gew(s_q) > \ldots$ für jede Ableitung ...

Seite 13: erster Punkt: Falls $n$ einen Summanden der Form $a$ enthält, dann gilt $n \stackrel{a}{\rightarrow} \surd$ und wegen $n \underline{\leftrightarrow} n'$ auch $n' \stackrel{a}{\rightarrow} \surd$. Also ist $a$ auch in $n'$ als Summand enthalten.

Seite 27: In der Zeichnung sind zwei unterbrochene Linien zu ergänzen.

**Kapitel 4**

Seite 84, 5. Punkt: $Guards = \{guard_t \mid t \in T\}$ ordnet jeder Transition $t \in T$ ein Prädikat $guard_t$ mit Variablen aus Var zu.

Seite 93: *Funktionen* :

$$MINE : DBM \rightarrow Bag(MB)$$

**Kapitel 8**

Seite 199:

$$(1 - \frac{1}{n})^{n-1} > \frac{1}{e} > (1 - \frac{1}{n})^n$$

Seite 210:

**Satz 8.10** *Gegeben sei ein Algorithmus für n Prozessoren, der wechselseitigen Ausschluß ohne Verklemmung mit Speichersynchronisation so realisiert, dass kein Prozess mehr als k-mal (für ein $k \in \mathbb{N}$) in den kritischen Abschnitt eintritt, während ein anderer darauf wartet. Dann wird dazu eine gemeinsame Variable mit mindestens n Werten (d.h. $\Omega(logn)$ Bit) benötigt.*

Dieses Ergebnis kann durch probabilistische Algorithmen verbessert werden:

**Satz 8.11** *Ein probabilistischer Algorithmus kann das Problem mit einer gemeinsamen Variablen mit konstanter Wertezahl (d.h. $\Omega(1)$ Bit) lösen, wobei jeder Prozessor mit der Wahrscheinlichkeit $\Theta(\frac{1}{n})$ Zutritt hat.*

**Kapitel 9**

Seite: 209: Korrekte Interaktion:

In jeder Rechnung $\sigma$ enthält $\sigma|i$ eine Folge von Aurufen und zugehörigen Antworten ( in dieser Reihenfolge) beginnend mit einem Aufruf.

**Beispiel 0.1** Beispielfolge $\sigma_1$, obere Zeile: $\sigma|0$ , untere Zeile: $\sigma|1$

$$\sigma_1 = \left\{ \begin{smallmatrix} x = 0 \wedge \\ y = 0 \end{smallmatrix} \right\} \begin{array}{l} \overbrace{write_0(x, 1) \qquad ack_0(x)}^{w_0} \qquad \overbrace{read_0(y) \qquad return_0(y, 1)}^{r_0} \\ \qquad \underbrace{write_1(y, 1) \qquad ack_1(y)}_{w_1} \qquad \underbrace{read_1(x) \qquad return_1(x, 1)}_{r_1} \end{array}$$

**Lebendigkeit:** ...

Seite 212: Zeile 1 des *Sequentially consistent local write algorithm*:

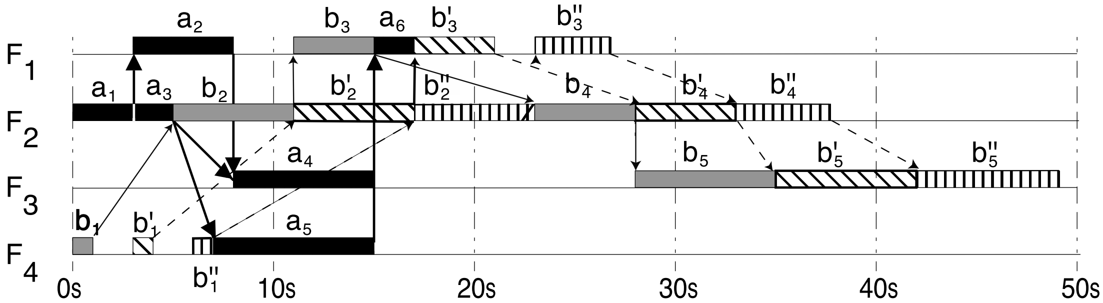
1: when $read_i(x)$ occurs:

**Kapitel 11**

Seite 271 Formel (11.13) lies:

$$\overline{y}(t_1, t_2) = \frac{1}{n_a(t_2) - n_a(t_1)} \cdot \sum_{i=n_z(t_1)+1}^{n_z(t_2)} y_i = \frac{1}{n_a(t_2) - n_a(t_1)} \int_{t_1}^{t_2} f(t)dt$$

Seite 278: Abbildung 11.14 b):




Seite 279 Mitte: Typisches Beispiel : Durchführung eines „Benchmarktests“, in dem auf ein zu Beginn und zu Ende des Tests leeres System ...

Seite 285, Formel (11.40):

$$\rho_i = \frac{E[V_i] \cdot E[D_S]}{c_i} = \frac{E[V_i] \cdot E[B_i]}{m_i} \cdot E[D_S]$$

Seite 286, 2. Formel von unten: $E[B_{S'}] = 100 \cdot 120ms + 100 \cdot 20ms = 14s$