2 Unity: Grundlage für den Entwurf nebenläufiger Programme

Chandy, K. M.; Misra, J.: Parallel Program Design - A Foundation. Addison-Wesley Publishing Company, Reading, Massachusetts, 1989.

2.1 Grundidee, Syntax und Semantik

2.1.1 Grundidee

1. Beschreibung der “Aufgabe” reaktiver Programmsysteme durch Eigenschaften ihrer Abläufe und der zugehörigen Zustandsfolgen
   Trennung von Lösung des eigentlichen Problems und Fragen der Implementierung:
   - Befehlszähler ersetzen durch allgemeine Forderung an nebenläufige Aktionen, alle Aktivitäten als zyklische Prozesse.
   - Ablaufplan “Fairness”, d. h. in einem unendlichen Ablauf tritt jede Zuweisung unendlich oft auf.
2. Ziel: Bessere Verständlichkeit von Programmen und “Wiederverwendbarkeit” für verschiedene Architekturen
3. Programmentwicklung erfolgt stufenweise.
   Nach der Lösung des Grundproblems schrittweise Optimierung zur Nutzung der gegebenen Hardware-Möglichkeiten durch “Transformationen”

4. Zentrale Eigenschaften:
   - Nichtdeterminismus
     In frühen Phasen der Programmentwicklung keine Aussage über die Abarbeitungsreihenfolge.
   - Keine Kontrollflußkontrolle
     Sicht auf Programme ist nicht mehr von vorneherein sequentiell.
   - Synchronität und Asynchronität
     Beides muß im Modell erfaßbar sein.
   - Zustände und Zuweisungen
     Modell basiert auf Theorie der Zustandsübergänge und auf Grundelementen von Programmiersprachen.
   - **Aber: Die traditionelle von-Neumann-Sicht (Befehlszähler, nur ein Speichertransfer zu jedem Zeitpunkt) wird aufgegeben.**

2.1.2 Syntax von Unity-Programmen

program →	**Program**	program-name	
	declare	declare-section	// Deklaration der Zustandsvariablen
	always	always-section	// Definition von Variablen als Funktion anderer
	initially	initially-section	// Prädikat, das der Anfangszustand erfüllen muß
	assign	assign-section	// Zuweisungen, die die Zustandsübergänge beschreiben
	end		

2.1.3 Vereinbarungen

Syntax ähnlich Pascal
Beispiel: declare z: array[0 .. n] of integer

2.1.4 Zuweisungen

- Mehrfachzuweisungen, zwei Schreibweisen:
  $x, y, z := f(z, y, z), g(x, y, z), h(x, y, z)$ oder
  $x := f(x, y, z) \parallel y := g(x, y, z) \parallel z := h(x, y, z)$
- Abkürzung durch “Quantifizierung”:
  Anstelle von $A[0] := B[0] \parallel A[1] := B[1] \parallel \ldots \parallel A[N] := B[N]$
  quantifiziert $\langle \parallel i: 0 \le i \le N :: A[i] := B[i] \rangle$
  Beispiel: Vorbesetzung einer Einheitsmatrix
  $\langle \parallel i, j: 0 \le i \le N \wedge 0 \le j \le N \wedge i \ne j :: U[i, j] := 0 \rangle$
  $\parallel \langle \parallel i: 0 \le i \le N :: U[i, j] := 1 \rangle$
- Bedingte Zuweisung
  $$x := \begin{array}{lll} -1 & \text{if} & y < 0 \quad \sim \\ 0 & \text{if} & y = 0 \quad \sim \\ 1 & \text{if} & y > 0 \end{array}$$
  Beispiel: Vorbesetzung einer Einheitsmatrix
  $\langle \parallel i, j: 0 \le i \le N \wedge 0 \le j \le N :: U[i, j] := 0 \ \text{if} \ i \ne j \sim 1 \ \text{if} \ i = j \rangle$

- Nichtdeterministische Auswahl beschrieben durch Operator []. Mit [] getrennte Zuweisungen werden nichtdeterministisch zur Ausführung ausgewählt unter Einhaltung der schwachen Fairneß.
  Beispiel: Vorbesetzung einer Einheitsmatrix
  $\langle [] \; i, j: 0 \le i \le N \wedge 0 \le j \le N :: U[i, j] := 0 \;\text{ if }\; i \ne j \;\sim\; 1 \;\text{ if }\; i = j \rangle$
  oder
  $\langle || \; i, j: 0 \le i \le N \wedge 0 \le j \le N \wedge i \ne j :: U[i, j] := 0 \rangle$
  $[] \langle || \; i: 0 \le i \le N :: U[i, i] := 1 \rangle$

2.1.5 Initialisierung

- Syntax analog Zuweisung, wobei anstelle von := das Gleichheitszeichen = steht
- Die Gleichungen können so angeordnet werden, daß der zugewiesene Wert eine Funktion von Konstanten oder Anfangswerten vorher initialisierter Variablen ist.
  Beispiel:
  $B[0] = 0 \;||\; N = 2$
  $[] \langle ||i: 0 < i \le N :: A[i] = B[i-1] [] B[i] = A[i] \rangle$

2.1.6 Der ALWAYS-Abschnitt

- Syntax analog Initialisierung
- Dient der Definition von Variablen als Funktion von anderen.
- Legt Invarianten des Programms fest.
- Dient lediglich der übersichtlicheren und verständlicheren Darstellung.

2.1.7 Beispiel: Terminplanung

- N Beteiligte suchen Termin für das nächstmögliche ganztägige Treffen.
- Kalenderfunktion i.f(x) ermittelt zu einem Datum x das früheste, nicht vor x liegende, zu dem der Terminkalender des i-ten Beteiligten noch keinen Eintrag enthält.
- **Program** P2
  declare r : integer;
  initially r = 0
  assign $\langle []i: 0 \le i \le N :: r := i.f(r) \rangle$
  end {P2}
- Ergebnis ist der sich einstellende Fixpunkt, falls er existiert.