8 Unity: Grundlage für den Entwurf nebenläufiger Programme

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

8.1 Grundidee, Syntax und Semantik

8.1.1 Grundidee

- Beschreibung der "Aufgabe" reaktiver Programmsysteme durch Eigenschaften ihrer Abläufe und der zugehörigen Zustandsfolgen.
- Trennung zwischen Lösung des eigentlichen Problems und Fragen der Implementierung
  - Befehlszähler durch allgemeine Forderung an nebenläufige Aktionen ersetzen, alle Aktivitäten als zyklische Prozesse.
  - Ablaufplan "Fairness", d. h. in einem unendlichen Ablauf tritt jede Zuweisung unendlich oft auf.

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

- Ziel: Bessere Verständlichkeit von Programmen und "Wiederverwendbarkeit" für verschiedene Architekturen
  - Programmentwicklung erfolgt stufenweise.
  - Nach der Lösung des Grundproblems schrittweise Optimierung zur Nutzung der gegebenen Hardware-Möglichkeiten durch "Transformationen"
- Zentrale Eigenschaften:
  - **`Indeterminismus`**
    In frühen Phasen der Programmentwicklung keine Aussage über die Abarbeitungsreihenfolge.
  - **`Keine Kontrollflußkontrolle`**
    Sicht auf Programme ist nicht mehr von vornherein sequentiell.
  - **`Synchronität und Asynchronität`**
    Beides kann im Modell erfaßt werden.
  - **`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.**

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

8.1.2 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

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

8.1.3 Vereinbarungen

Syntax ähnlich Pascal

Beispiel: `declare z: array[0 .. n] of integer`

8.1.4 Zuweisungen

- **Mehrfachzuweisungen, zwei Schreibweisen:**

  $x, y, z := f(x, y, z), g(x, y, z), h(x, y, z)$ **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 $\langle \parallel i : 0 \leq i \leq N :: A[i] := B[i] \rangle$

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

Beispiel: Vorbesetzung einer Einheitsmatrix

$$\langle \| \; i, j: 0 \le i \le N \wedge 0 \le j \le N \wedge i \neq j :: U[i, j] := 0 \rangle$$
$$\| \; \langle \| \; i: 0 \le i \le N :: U[i, i] := 1 \rangle$$

- **Bedingte Zuweisung**

$$x := \begin{array}{llll} -1 & \text{if} & y < 0 & \sim \\ 0 & \text{if} & y = 0 & \sim \\ 1 & \text{if} & y > 0 & \end{array}$$

Beispiel: Vorbesetzung einer Einheitsmatrix

$$\langle \| \; i, j: 0 \le i \le N \wedge 0 \le j \le N :: U[i, j] := 0 \;\; \text{if} \;\; i \neq j \;\; \sim \;\; 1 \;\; \text{if} \;\; i = j \rangle$$

- **Nichtdeterministische Auswahl beschrieben durch Operator [].**

Mit [] getrennte Zuweisungen werden nichtdeterministisch zur Ausführung ausgewählt unter Einhaltung der schwachen Fairneß.

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

Beispiel: Vorbesetzung einer Einheitsmatrix

$$\langle [] \; i, j: 0 \le i \le N \wedge 0 \le j \le N :: U[i, j] := 0 \;\; \text{if} \;\; i \neq j \;\; \sim \;\; 1 \;\; \text{if} \;\; i = j \rangle$$

oder

$$\langle \| \; i, j: 0 \le i \le N \wedge 0 \le j \le N \wedge i \neq j :: U[i, j] := 0 \rangle$$
$$[] \; \langle \| \; i: 0 \le i \le N :: U[i, i] := 1 \rangle$$

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

8.1.5 Initialisierung

- Syntax analog Zuweisung, wobei anstelle von := das Gleichheitszeichen = steht
- Die Gleichungen müssen so angeordnet werden können, daß der zugewiesene Wert eine Funktion von Konstanten oder Anfangswerten vorher initialisierter Variablen ist.

Beispiel:

$$\mathbf{B}[\mathbf{0}] = \mathbf{0} \;\|\; \mathbf{N} = \mathbf{2}$$

$$[] \langle \;\|\mathbf{i}: \mathbf{0} < \mathbf{i} \leq \mathbf{N} :: \mathbf{A}[\mathbf{i}] = \mathbf{B}[\mathbf{i}-\mathbf{1}][] \; \mathbf{B}[\mathbf{i}] = \mathbf{A}[\mathbf{i}] \rangle$$

8.1.6 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.

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

8.1.7 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 Datum, zu dem der Terminkalender des i-ten Beteiligten noch keinen Eintrag enthält.
- Programm

```
Program P2
   declare    r: integer;
   initially  r = 0
   assign     ⟨[]i: 0 ≤ i ≤ N :: r := i.f(r)⟩
end {P2}
```

- Ergebnis ist der sich einstellende Fixpunkt, falls er existiert.

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

8.2 Basisrelationen zur Beschreibung des Programmverhaltens

8.2.1 Zusicherungstechnik nach Floyd-Hoare

$\{p\}s\{q\}$ besagt, daß die Ausführung der Anweisung s in einem Zustand, der das Prädikat p erfüllt, zu einem Zustand führt, der das Prädikat q erfüllt.

Einige wichtige Aussagen:

- $\{p\}s\{true\}$
- $\{false\}s\{q\}$
- $$\frac{\{p\}s\{false\}}{\neg p}$$
- $$\frac{\{p\}s\{q\},\ \{p'\}s\{q'\}}{\{p \vee p'\}s\{q \vee q'\},\ \{p \wedge p'\}s\{q \vee q'\}}$$
- $$\left\{(b_0 \Rightarrow q^x_{e_0}) \wedge \ \ldots \wedge (b_n \Rightarrow q^x_{e_n}) \wedge ((\neg b_0 \wedge \ldots \wedge \neg b_n) \Rightarrow q)\right\}$$
  $$x := e_0 \ \text{if} \ b_0 \ \sim \ldots \sim \ e_n \ \text{if} \ b_n \quad \{q\}$$

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

8.2.2 Die Relation unless

❑ **Definition der Relation unless**

◆ **Quantifizierung über die Anweisungen eines Programms**

F sei ein Programm

- $\langle \forall s(s \text{ in } F \Rightarrow \{p\}s\{q\})\rangle$ **soll zum Ausdruck bringen, daß für alle Anweisung s des Programms F die Aussage** $\{p\}s\{q\}$ **zutrifft.**
- $\langle \exists s(s \text{ in } F \Rightarrow \{p\}s\{q\})\rangle$ **soll zum Ausdruck brinngen, daß auf wenigstens eine Anweisung des Programms F die Aussage** $\{p\}s\{q\}$ **zutrifft.**

◆ **F sei ein Programm, s eine Anweisung**

Definition: $p \text{ unless } q \equiv \langle \forall s: s \text{ in } F :: \{p \wedge \neg q\}s\{p \vee q\}\rangle$

Informell: Wenn p erst einmal gültig ist, dann bleibt es mindestens bis zum Eintreten von q gültig.

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

- **Beispiel: Der Wert der Variablen x ist unbegrenzt monoton wachsend**

  $\mathbf{x = k}$ **unless** $\mathbf{x > k}$

  - **Spezialfälle:**

    stable $\mathbf{p \equiv p}$ **unless false**

    invariant $\mathbf{q \equiv (}$**initial condition** $\mathbf{\Rightarrow q) \wedge q}$ **is stable**

  - **Bemerkung:**

    stable p ist äquivalent zu $\forall$**s(s in F** $\Rightarrow \{\mathbf{p}\}\mathbf{s}\{\mathbf{p}\}$**),**
    d. h. p wird von jeder Anweisung des Programms bewahrt.

  - **Der Always-Abschnitt eines Programms kann als Definition einer Invarianten angesehen werden.**

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

8.2.3 Die Relation ensures

- **Definition der Relation ensures**

  p ensures $\mathbf{q \equiv (p}$ **unless** $\mathbf{q \wedge \langle \exists s}$**: s in F ::** $\{\mathbf{p \wedge \neg q}\}\mathbf{s}\{\mathbf{q}\}\rangle)$

- **Beispiel: Der Wert der Variablen x ist unbegrenzt monoton wachsend**

  $\mathbf{x = k}$ **ensures** $\mathbf{x > k}$

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

8.2.4 **Die Relation leads-to ($\mapsto$)**

Definition der Relation leads-to

Es gilt $p \mapsto q$ genau dann, wenn es auf Grund des folgenden Regelsystems ableitbar ist:

$$\frac{p \text{ ensures } q}{p \mapsto q}$$

(Transitivität)

$$\frac{p \mapsto q,\ q \mapsto r}{p \mapsto r}$$

(Disjunktivität)

$$\frac{\forall m(m \in W \Rightarrow p(m) \mapsto q)}{\langle \exists m(m \in W \wedge p(m)) \rangle \mapsto q}$$

wobei W eine Menge möglicher Argumente für p(x) ist

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

8.2.5 **Fixpunkt**

- **Sei R eine zu einer Ausführungsfolge gehörige Zustandsfolge.**

  Dann besitzt jedes Element R_i von R eine Komponente

  R_i.state, die den Zustand der Variablen darstellt, und eine Komponente

  R_i.label, die die Zuweisung charakterisiert, die zu R_{i+1}.state führt.

  R_0.state bezeichnet den Anfangszustand.

- **Das Fixpunktprädikat FP eines Programms G ist folgendermaßen definiert:**

  $\forall s(s \text{ in } G \wedge s \text{ is } X := E \Rightarrow (X = E))$

  oder im Zustandsmodell

  $FP[R_i] \equiv \langle \forall j(j \geq i \Rightarrow R_i.state = R_j.state) \rangle$.

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

8.2.6 Die Relationen detects und until

❑ **Definitionen**

$$\mathbf{p} \text{ detects } \mathbf{q} \equiv (\mathbf{p} \Rightarrow \mathbf{q}) \wedge (\mathbf{q} \ \mapsto \mathbf{p})$$

$$\mathbf{p} \text{ until } \mathbf{q} \equiv (\mathbf{p} \text{ unless } \mathbf{q}) \wedge (\mathbf{p} \ \mapsto \mathbf{q})$$

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

8.2.7 Wichtige Theoreme

❑ **unless**

- **Reflexivität**

$$\mathbf{p} \text{ unless } \mathbf{p}$$

- **Antireflexivität**

$$\mathbf{p} \text{ unless } \neg\mathbf{p}$$

- **Abschwächung der Nachbedingung**

$$\frac{\mathbf{p} \text{ unless } \mathbf{q},\ \mathbf{q} \Rightarrow \mathbf{r}}{\mathbf{p} \text{ unless } \mathbf{r}}$$

- **Konjunktion und Disjunktion**

$$\frac{\mathbf{p} \text{ unless } \mathbf{q},\ \mathbf{p}' \text{ unless } \mathbf{q}'}{\begin{array}{c}(\mathbf{p} \wedge \mathbf{p}') \text{ unless } (\mathbf{p} \wedge \mathbf{q}') \vee (\mathbf{p}' \wedge \mathbf{q}) \vee (\mathbf{q} \wedge \mathbf{q}') \\ (\mathbf{p} \vee \mathbf{p}) \text{ unless } (\neg\mathbf{p} \wedge \mathbf{q}') \vee (\neg\mathbf{p}' \wedge \mathbf{q}) \vee (\mathbf{q} \wedge \mathbf{q}')\end{array}}$$

- **Kürzung**

$$\frac{\mathbf{p} \text{ unless } \mathbf{q},\ \mathbf{q} \text{ unless } \mathbf{r}}{\mathbf{p} \vee \mathbf{q} \text{ unless } \mathbf{r}}$$

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

ensures

- **Reflexivität**

$$\mathbf{p \text{ ensures } p}$$

- **Abschwächung der Nachbedingung**

$$\frac{\mathbf{p \text{ ensures } q},\ \mathbf{q} \Rightarrow \mathbf{r}}{\mathbf{p \text{ ensures } r}}$$

- **Unmöglichkeit**

$$\frac{\mathbf{p \text{ ensures false}}}{\neg \mathbf{p}}$$

- **Konjunktion**

$$\frac{\mathbf{p \text{ unless } q},\ \mathbf{p'} \text{ ensures } \mathbf{q'}}{\mathbf{p} \wedge \mathbf{p'} \text{ ensures } (\mathbf{p} \wedge \mathbf{q'}) \vee (\mathbf{p'} \wedge \mathbf{q}) \vee (\mathbf{q} \wedge \mathbf{q'})}$$

- **Disjunktion**

$$\frac{\mathbf{p \text{ ensures } q}}{\mathbf{p} \vee \mathbf{r} \text{ ensures } \mathbf{q} \vee \mathbf{r}}$$

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

leads-to

- **Implikation**

$$\frac{\mathbf{p} \Rightarrow \mathbf{q}}{\mathbf{p} \mapsto \mathbf{q}}$$

- **Unmöglichkeit**

$$\frac{\mathbf{p} \mapsto \mathbf{false}}{\neg \mathbf{p}}$$

- **Disjunktion**

  Für jede Menge W gilt:

$$\frac{\langle \forall \mathbf{m} (\mathbf{m} \in \mathbf{W} \Rightarrow \mathbf{p}(\mathbf{m}) \mapsto \mathbf{q}(\mathbf{m})) \rangle}{\langle \exists \mathbf{m} (\mathbf{m} \in \mathbf{W} \wedge \mathbf{p}(\mathbf{m})) \rangle \mapsto \langle \exists \mathbf{m} (\mathbf{m} \in \mathbf{W} \wedge \mathbf{q}(\mathbf{m})) \rangle}$$

- **Kürzung**

$$\frac{\mathbf{p} \mapsto \mathbf{q} \vee \mathbf{b},\ \mathbf{b} \mapsto \mathbf{r}}{\mathbf{p} \mapsto \mathbf{q} \vee \mathbf{r}}$$

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

leads-to (Fortsetzung)

- **PSP-Theorem (Progress-Safety-Progress)**

$$\frac{p \mapsto q,\ r \text{ unless } b}{p \wedge r \mapsto (q \wedge r) \vee b}$$

- **Fertigstellung**

  Gegeben seien endliche Mengen von Prädikaten p_i und q_i

$$\frac{\langle \forall i (p_i \mapsto q_i \vee r) \rangle \quad \langle \forall i (q_i \text{ unless } r) \rangle}{\langle \wedge i (p_i) \rangle \mapsto \langle \wedge i (q_i) \rangle \vee r}$$

Fixpunkt

- **Stabilität im Fixpunkt**

  Für jedes Prädikat p ist $(FP \wedge p)$ stabil.

- **Korollar**

$$\frac{p \mapsto q}{FP \Rightarrow (p \Rightarrow q)}$$

8.3 **Erfassung der Systemarchitektur**

Mappings beschreiben informal Eigenschaften der Architektur des Rechensystems und ihre Auswirkung auf UNITY-Programme.

8.3.1 **Asynchrone Prozessoren mit lokal gemeinsamen Speichern (Multiprozessoren)**

1. **Jede Anweisung wird einem bestimmten Prozessor zugeordnet.**
2. **Jede Variable wird einem bestimmten Speicher zugeordnet.**
3. **Für jeden Prozessor wird die Reihenfolge festgelegt, in der er die Anweisungen ausführt.**

❑ **Diese Abbildung unterliegt folgenden Einschränkungen:**

1. **Alle Variablen auf der linken Seite einer Zuweisung sind in Speichern hinterlegt, deren Inhalt von dem Prozessor, dem die Zuweisung zugeordnet ist, modifiziert werden kann.**
2. **Alle Variablen auf der rechten Seite einer Zuweisung sind in Speichern hinterlegt, deren Inhalt von dem Prozessor, dem die Zuweisung zugeordnet ist, gelesen werden kann.**
3. **Für jeden Prozessor stellt die Ausführungsreihenfolge sicher, daß jede ihm zugewiesene Anweisung unendlich oft zur Ausführung kommt.**

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

8.3.2 **Verteilte Systeme (Prozessoren mit lokalem Speicher und Verbindungskanälen)**

1. **Jede Zuweisung wird einem bestimmten Prozessor zugeordnet.**
2. **Jede Variable wird dem lokalen Speicher oder einem Kanal zugeordnet.**
3. **Für jeden Prozessor wird die Reihenfolge festgelegt, in der er die Anweisungen ausführt.**

❑ **Diese Abbildung unterliegt folgenden Einschränkungen:**

1. **Jedem Kanal ist höchstens eine Variable zugeordnet und diese Variable ist vom Typ Sequenz (FIFO-Kanal).**
2. **Eine Variable, die einem Kanal zugeordnet ist, kommt nur in Anweisungen zweier Prozessoren vor und diese haben eine der folgenden Formen:**
   - **Anweisungen eines der Prozessoren modifizieren die Variable durch Anfügen eines Wertes, solange die Sequenz eine bestimmte Länge (Pufferlänge) nicht überschreitet.**
   - **Anweisungen des anderen Prozessors modifizieren die Variable durch Entfernen des ersten Wertes der Sequenz, falls ihre Länge größer als null ist.**
   - **Die Variable wird auf keine andere Weise benutzt.**

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

8.4 Anwendungsbeispiel: Verteilte Terminierungsentdeckung

8.4.1 Definition des generischen Entdeckungsproblems

8.4.1.1 Programmüberlagerungen

❑ Gegeben sei ein Programm F und eine stabile Eigenschaft W dieses Programms.

Durch **Überlagerung** entstehe daraus ein Programm mit einer neuen Variablen *claim*, so daß gilt: claim detects W.

Unter Überlagerung wird verstanden:

Hinzufügen von Variablen, einschließlich ihrer Initialisierung,

Erweitern der Zuweisungen durch neue parallele Zuweisungen und

Hinzufügen neuer Zuweisungen,

wobei die neuen Zuweisungen keine Variablen von W verändern.

F wird als das **unterliegende** Programm bezeichnet.

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

❑ **Überlagerungssatz**

Alle in den Basisrelationen ausdrückbaren gültigen Prädikate des unterliegenden Programms sind auch in jeder Überlagerung erfüllt.

8.4.1.2 Beispiel zur Motivation

W ≡ (die Anzahl der ausgeführten Befehle übersteigt 10)

```
Program {superposition} P1
      initially  count = 0 [] claim = false
      transform
         each statement s in the underlying program to:
            s || count := count + 1
      add   claim := (count > 10)
end {P1}
```

- Folgerung

  claim *detects* (count > 10)

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

8.4.2 Heuristiken

1. Direkte Implementierung der Entdeckung

(I) x detects e, wobei x eine überlagerte Variable ist und e ein stabiles Prädikat über dem zugrundeliegenden Programm.

- Anfangs: x = false
- Genau ein hinzugefügter Befehl: x := e

Beweis:

a) Nachweis für **invariant x** $\Rightarrow$ **e**

Aussage am Anfang richtig.

Kann nur falsch werden, wenn e falsch wird oder x wahr.

Da e stabil, ersteres nicht möglich.

Letzteres nur durch Befehl x := e;

Nachbedingung ist x = e, also gilt **x** $\Rightarrow$ **e**.

Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

b) Nachweis für **e** $\mapsto$ **x**

(**stable e** $\Leftrightarrow$ **e unless false**) $\Rightarrow$ **e unless x** und somit

$\{\mathbf{e} \wedge \neg \mathbf{x}\}\mathbf{x} := \mathbf{e}\{\mathbf{x}\}$,

also **e ensures x** und damit **e** $\mapsto$ **x**.

(II) Implementierung von $(\mathbf{x} \geq \mathbf{k})$ **detects** $(\mathbf{e} \geq \mathbf{k})$, wobei x überlagerte Variable ist und e ein monoton wachsender ganzzahliger Ausdruck.

- Anfangs **x** = (**Anfangswert von e**).
- Genau ein hinzugefügter Befehl: x := e

Beweis:

Mit der Bemerkung, daß die Spezifikation die Eigenschaft **invariant x** $\leq$ **e** und $(\mathbf{e} = \mathbf{k}) \mapsto (\mathbf{x} \geq \mathbf{k})$ besitzt, erfolgt der Beweis analog dem vorangehenden.

Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

2. Nutzung der Transitivität von detects

(III) Implementierung von **claim detects W(d)**, wobei es ineffizient ist, **W(d)** auf der gegebenen Architektur direkt festzustellen
(z. B. weil d eine verteilte Datenstruktur ist)

- Auffinden eines günstiger festzustellenden Prädikats **p(d')** über einer geeigneten Datenstruktur d'.
- Gestaltung des Programms so, daß

  claim detects p(d') und **p(d') detects W(d)**.

Die Invariante, die d' beschreibt, ist normalerweise schwächer als die von d, die Fortschritt-Bedingung stärker.

Beweis:

Transitivität von detects folgt aus der von $\mapsto$.

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

8.4.3 Anwendungsbeispiel für die Heuristiken

❑ **Ausführung des vorangehenden Programms in einem verteilten System**

Aufspaltung des Zählers nach (III), dazu:

1. **Einführung einer überlagerten Variablen u.m für jeden Prozessor u zur Zählung der lokalen Befehlsausführungen.**
2. **Einführung einer weiteren überlagerten Variablen n, der von Zeit zu Zeit die Summe der u.m zugewiesen wird unter Verwendung von (II).**
3. **Mittels claim wird count > 10 festgestellt nach (I). count > 10 ist stabil, weil n monoton steigend ist.**

❑ **Formale Beschreibung**

invariant **u.m =** (**Anzahl der Befehlsausführungen in Prozeß u**)

(**count** > **k**) **detects** (⟨**+ u :: u.m**⟩ > **k**)

claim detects (**count** > **10**)

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

- **Formale Beschreibung (Wiederholung)**

  invariant **u.m = (Anzahl der Befehlsausführungen in Prozeß u)**

  (count > k) detects (⟨+ u :: u.m⟩ > k)

  claim detects (count > 10)

- **Korrektheitsbeweis der Lösungsstrategie**

  Aus claim detects (count > 10)

  und (count > k) detects (⟨+ u :: u.m⟩ > k)

  folgt wegen der Transitivität von detects

  claim detects (⟨+ u :: u.m⟩ > 10)

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

Hier wird deutlich wie die Invariante abgeschwächt und die Fortschritt-Bedingung verstärkt wird:

claim detects (⟨+ u :: u.m⟩ > 10)

ist äquivalent zu

claim ⇒ (⟨+ u :: u.m⟩ > 10)

und (⟨+ u :: u.m⟩ > 10) ↦ (count > 10)

Ersteres wird abgeschwächt zu **claim detects (count > 10)**

letzteres verstärkt zu **(⟨+ u :: u.m⟩ > k) ↦ (count > k)**

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

❑ **Ableitung des Programms**

```
Program {superposition} P2
   initially  count = 0 [] claim = false [] ⟨[] u :: u.m = 0⟩
   transform
      for each processor u transform each statement s
      in processor u to
         s || u.m := u.m + 1
   add   count := ⟨+ u :: u.m⟩ [] claim := (count > 10)
end {P2}
```


Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

8.4.4 Weitere Anwendung der Heuristiken

❑ **Vorgehensweise**

1. **Aufspaltung der Summation über nicht-lokale Variablen in eine Summation über lokale Kopien der Variablen. Dazu:**
2. **Einführung weiterer überlagerter Variablen u.r, die Kopien der Variablen u.m darstellen (II) und lokal zum entdeckenden Prozessor sind.**
3. **Variable count erhält die Summe dieser Zwischenvariablen.**
4. **Dadurch Entdeckung mit Zwischenschritt (III).**

❑ **Formale Beschreibung**

invariant u.m = (Anzahl der Befehlsausführungen in Prozeß u)

(u.r > k) detects (u.m > k) **(für alle u)**

claim detects (count > 10)

Folgerung: (⟨+ u :: u.r⟩ > k) detects (⟨+ u :: u.m⟩ > k)

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

- **Korrektheitsbeweis der Lösungsstrategie**

  Zweimalige Anwendung der Transitivität von *detects*.

- **Ableitung des Programms**

```
Program {superposition} P3
   initially  n = 0 [] claim = false [] ⟨[] u :: u.m, u.r = 0, 0⟩
   transform
      for each processor u, transform each statement s
      in processor u to
         s || u.m := u.m + 1
   add   n := ⟨+ u :: u.r⟩ [] claim := (n > 10)
         [] ⟨[] u :: u.r := u.m⟩
end {P3}
```


Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

8.4.5 Spezifikation der Terminierungsentdeckung

- **Das zugrunde gelegte Programm wird repräsentiert durch einen gerichteten Graphen $G = (V, E)$, wobei V eine (konstante) Menge von Knoten ist, die die Prozesse darstellen, und E eine (konstante) Menge von Pfeilen, die unidirektionale Verbindungskanäle zwischen den Prozessen darstellen.**
- **Mit jedem Knoten u ist ein Prädikat u.idle verbunden, das anzeigt, ob der entsprechende Prozeß passiv ist.**
- **Programmstruktur (synchrones Modell!)**

```
Program {structure of the underlying program} R0
   assign      ⟨[] u, v :: (u, v) ∈ E :: v.idle := u.idle ∧ v.idle⟩
           [] ⟨[] u :: u ∈ V :: u.idle := true⟩
end {R0}
```

- **Stabiles Prädikat W, das entdeckt werden soll:**

  $W \equiv \langle \wedge u: u \in V :: u.idle \rangle$

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

8.4.6 Sequentielle Architektur

❑ **Heuristik I**

```
Program {superposition on R0} R1
   initially  claim = false
   add        claim := ⟨∧ u :: u.idle⟩
end {R1}
```


Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

8.4.7 Asynchrone Architektur mit lokal gemeinsamen Speichern

❑ **Informelle Beschreibung**

1. Einführung einer Variablen d, definiert als die Menge der inaktiven Prozesse.
2. Bei Änderungen des Prozeßzustandes wird d entsprechend aktualisiert.
3. Wenn d alle Prozesse enthält, ist das Programm terminiert.

❑ **Formale Beschreibung**

invariant $d = \{u \mid u.idle\}$

claim detects $(d = V)$

❑ **Korrektheitsbeweis**

Trivial mit $W \equiv (d = V)$ **.**

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

❑ **Ableitung des Programms**

```
Program {the transformed program} R2
   initially  claim = false [] d = {u | u.idle}
   assign       ⟨[] (u, v) :: v.idle := u.idle ∧ v.idle
                              || d := d - {v} if  ¬u.idle
                ⟩
              [] ⟨[] u ::u.idle := true
                        || d := d ∪ {u} if  u.idle
                ⟩
              [] claim := (d = V)
end {R2}
```


Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

8.4.7.1 **Erste Entkopplung der Modifikationen**

❑ **Informelle Beschreibung**

1. **Abschwächung der Invariante dahingehend, daß d eine Teilmenge der inaktiven Prozesse ist (III).**
2. **Verstärkung der Fortschritt-Bedingung: Endliche Zeit, nachdem alle Prozesse inaktiv sind, enthält d alle Prozesse.**
3. **Prozesse werden zu d asynchron hinzugefügt, nachdem sie inaktiv geworden sind.**

❑ **Formale Beschreibung**

invariant $d \subseteq \{u \mid u.idle\}$

$W \mapsto (d = V)$ **(Fortschrittbedingung)**

claim detects $(d = V)$ **(Entdeckungsbedingung)**

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

❑ **Korrektheitsbeweis**

$(d = V) \Rightarrow W$, wegen der ersten Invariante

$(d = V)$ detects W , wegen der Fortschrittbedingung

claim detects W , wegen der Entdeckungsbed. und der Transitivität von *detects*

❑ **Ableitung des Programms**

```
Program {the transformed program} R2'
   initially  claim = false [] d = {u | u.idle}
   assign     ⟨[] (u, v) :: v.idle := u.idle ∧ v.idle
                          || d := d - {v}   if  ¬u.idle
              ⟩
              [] ⟨[] u ::u.idle := true
                       [] d := d ∪ {u}   if  u.idle
              ⟩
              [] claim := (d = V)
end {R2'}
```


Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

8.4.8 Weitere Entkopplung der unterliegenden und der überlagerten Berechnung

❑ **Informelle Beschreibung**

1. Einführung einer neuen Variablen b zusammen mit lokalen Variablen u.delta.
2. Dadurch Entdeckung mit Zwischenschritt (III).

❑ **Formale Beschreibung**

Die Relation *chain* zwischen Prozessen sei zu jedem Zeitpunkt der Berechnung definiert durch:

u chain v $\equiv$ Im unterliegenden Graphen G existiert ein Pfad von u nach v derart, daß für jede Kante (u', v') auf diesem Pfad $v' \in u'.delta$ ist.

invariant $v \in b \Rightarrow [(v.delta = empty) \wedge v.idle] \vee \langle \exists u: u \notin b :: u \text{ chain } v \rangle$

stable $b = V$

$W \mapsto (b = V)$ (Fortschrittbedingung)

claim detects $(b = V)$ (Entdeckungsbedingung)

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

❑ **Korrektheitsbeweis der Lösungsstrategie**

(b = V) ⇒ W	wegen der Invariante, da die zweite Alternative nicht zutreffen kann
(b = V) detects W	wegen der vorangehenden Beziehung und der Fortschritt-Bedingung
claim detects W	wegen der Transitivität von *detects*

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

❑ **Ableitung des Programms**

```
Program {the transformed program} R3
  initially  claim = false [] b = empty [] ⟨[] u :: u.delta = empty⟩
  assign     ⟨[] (u, v) :: v.idle := u.idle ∧ v.idle
                        || u.delta := u.delta ∪ {v}
                              if  ¬u.idle
             ⟩
           [] ⟨[] u :: u.idle := true⟩
           [] ⟨[] u :: b, u.delta := b ∪ {u} - u.delta, empty
                              if  u.idle
             ⟩
           [] claim := (b = V)
  end {R3}
```


Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

❑ **Korrektheitsbeweis des Programms**

(i) invariant $\mathbf{v} \in \mathbf{b} \Rightarrow [(\mathbf{v.delta} = \mathbf{empty}) \wedge \mathbf{v.idle}] \vee \langle \exists \mathbf{u}: \mathbf{u} \notin \mathbf{b} :: \mathbf{u}\ \mathbf{chain}\ \mathbf{v} \rangle$

Im Anfangszustand erfüllt, da dort b = empty gilt.

Beweisstruktur:

Feststellung der Zustandsübergänge, die die Invariante ungültig machen könnten und Untersuchung entsprechender Anweisungen

	Vorbedingung	Nachbedingung
(1)	$\mathbf{v} \notin \mathbf{b}$	$\mathbf{v} \in \mathbf{b}$
(2)	$\mathbf{v.delta} = \mathbf{empty} \wedge \mathbf{v.idle}$	$\mathbf{v.delta} \neq \mathbf{empty}$
(3)	$\mathbf{v.delta} = \mathbf{empty} \wedge \mathbf{v.idle}$	$\neg \mathbf{v.idle}$
(4)	$\exists \mathbf{u}(\mathbf{u} \notin \mathbf{b} \wedge \mathbf{u}\ \mathbf{chain}\ \mathbf{v})$	$\forall \mathbf{u}(\mathbf{u} \notin \mathbf{b} \Rightarrow \neg(\mathbf{u}\ \mathbf{chain}\ \mathbf{v}))$

(1)Nur durch Befehl

b, v.delta := b ∪ {v} - v.delta, emptyif v.idle .

Erhält die Invariante.

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

(2)Nur durch Befehl

u.idle = v.idle ∧ u.idle || v.delta := v.delta ∪ {u}if ¬v.idle

Nach Vorbedingung **v.idle**, also Zustand unverändert.

(3)Nur durch Befehl

v.idle = u.idle ∧ v.idle || u.delta := u.delta ∪ {v}if ¬u.idle

Vor Ausführung u.idle: Zustand unverändert.

Vor Ausführung ¬u.idle: Wegen der Invariante

$(\mathbf{u} \notin \mathbf{b}) \vee (\exists \mathbf{u'}: \mathbf{u'} \notin \mathbf{b} :: \mathbf{u'}\ \mathbf{chain}\ \mathbf{u})$.

Also nach dem Befehl $\mathbf{v} \in \mathbf{u.delta}$ und somit **u chain v**.

Aus $\mathbf{u} \notin \mathbf{b}$ folgt mit $\mathbf{u} = \mathbf{u'}$: $\exists \mathbf{u'}: \mathbf{u'} \notin \mathbf{b} :: \mathbf{u'}\ \mathbf{chain}\ \mathbf{v}$.

Aus $\exists \mathbf{u'}: \mathbf{u'} \notin \mathbf{b} :: \mathbf{u'}\ \mathbf{chain}\ \mathbf{u}$ folgt dies mit

$\mathbf{u'}\ \mathbf{chain}\ \mathbf{u} \wedge \mathbf{u}\ \mathbf{chain}\ \mathbf{v} \Rightarrow \mathbf{u'}\ \mathbf{chain}\ \mathbf{v}$.

Also bleibt die Invariante erhalten.

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

(4)Zwei Fälle

1. Kette hat Verbindungsstück das unterbrochen wird:

   Vor Übergang $(\mathbf{u\ chain\ x}) \wedge (\mathbf{y} \in \mathbf{x.delta}) \wedge (\mathbf{y\ chain\ v})$,

   danach $(\mathbf{y} \notin \mathbf{x.delta}) \wedge (\mathbf{y\ chain\ v})$.

   Nur durch Befehl **b, x.delta := b ∪ {x} - x.delta, empty if x.idle**.

   Danach $\mathbf{y} \notin \mathbf{b}$ und damit $\mathbf{v} \notin \mathbf{b}$ für $\mathbf{y} = \mathbf{v}$ und

   $\mathbf{y} \notin \mathbf{b} \wedge (\mathbf{y\ chain\ v})$ für $\mathbf{y} \neq \mathbf{v}$.

   Also Invariante gesichert.

2. Kettenanfang u wird in b aufgenommen:

   Vor Übergang: $(\mathbf{y} \in \mathbf{delta}) \wedge (\mathbf{y\ chain\ v})$.

   Nach Übergang: $(\mathbf{y} \in \mathbf{b}) \wedge (\mathbf{y\ chain\ v})$.

   Nur durch Befehl **b, u.delta := b ∪ {u} - u.delta, emptyif u.idle**.

   Nach Ausführung $\mathbf{y} \in \mathbf{b}$ und damit $\mathbf{v} \notin \mathbf{b}$ für $\mathbf{y} = \mathbf{v}$ und

   $\mathbf{y} \notin \mathbf{b} \wedge (\mathbf{y\ chain\ v})$ für $\mathbf{y} \neq \mathbf{v}$.

   Also Invariante gesichert.

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

(ii) Fortschritt-Bedingung $\mathbf{W} \mapsto (\mathbf{b = V})$

Wenn W gilt, dann können nur die Befehle

⟨[] u :: b, u.delta := b ∪ {u} - u.delta, emptyif u.idle⟩

[] claim := (b = V)

den Zustand ändern.

Daraus folgt:

(a) $\mathbf{W} \mapsto (\mathbf{v.delta = empty})$

(b) $\mathbf{stable\ W} \wedge (\mathbf{v.delta = empty})$

(c) $(\mathbf{W} \wedge \forall \mathbf{v} :: \mathbf{v.delta = empty}) \mapsto (\mathbf{u} \in \mathbf{b})$

(d) $\mathbf{stable\ W} \wedge (\forall \mathbf{v} :: \mathbf{v.delta = empty}) \wedge \mathbf{u} \in \mathbf{b}$

Aus (a) und (b): $\mathbf{W} \mapsto (\forall \mathbf{v} :: \mathbf{v.delta = empty})$

und daraus mit (c) und (d): $\mathbf{W} \mapsto (\forall \mathbf{u} :: \mathbf{u} \in \mathbf{b})$.

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

(iii) Stabilität stable b = V

(**b = V**) $\wedge$ **Invariante** $\Rightarrow$ ($\forall$**v :: v.idle** $\wedge$ **v.delta = empty**).

Damit kann nur die letzte Zeile des Programms den Zustand ändern, also bleibt b auf jeden Fall unverändert.

(iv) claim detects (**b = V**)

Trivial nach Heuristik 1.

8.4.9 Verteilte Architekturen

8.4.9.1 Problembeschreibung

- **Grundlegendes Modell wie bei lokal gemeinsamem Speicher.**
- **Zusätzlich: Ein Kanal von u nach v wird dargestellt durch die Variable (u, v).c.**

  Da der Inhalt der Nachrichten des unterliegenden Systems belanglos ist, werden alle seine Nachrichten durch den Wert m dargestellt.
- **Anfänglich ist genau der Prozeß *init* aktiv.**

Unity: Terminierungsentdeckung

- **Programmstruktur (asynchrones verteiltes Modell!)**

```
Program {structure of the underlying program} DS
   assign
        { u sends message m along channel (u, v).c
          if u is active}
        ⟨[] u, v : (u, v) ∈ E :: (u, v).c := (u, v).c; m
                                  if  ¬u.idle ⟩
        { v becomes active by receiving a message
          along channel (u, v)}
      [] ⟨[] u, v : (u, v) ∈ E ::v.idle, m, (u, v).c := false,
                               head((u, v).c), tail((u, v).c)
                                  if  (u, v).c ≠ null ⟩
      [] ⟨[] u :: u.idle := true ⟩
end {DS}
```


Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

Unity: Terminierungsentdeckung

- **Stabiles Prädikat W, das entdeckt werden soll:**

$$W \equiv \langle \wedge u: u \in V :: u.idle \wedge (u, v).c = null \rangle$$

- **Kanalüberlagerung**

  Rückwirkungsfreie Mitbenutzung der vorhandenen Kanäle

  - **Durch getypte Nachrichten (vergleichbar den ‘message queues’ von Unix), der Typ legt fest, ob eine Nachricht zum überlagerten oder darunterliegenden Programm gehört.**
  - **Die unterliegende Programmausführung ignoriert überlagerte Nachrichten.**
  - **Es wird sichergestellt, daß eine Nachricht des unterliegenden Programms nicht unendlich lange durch überlagerte Nachrichten blockiert wird.**

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

❑ **Informelle Beschreibung der Lösungsstrategie**

- **Jeder Prozeß des Systems mit gemeinsamem Speicher wird einem Prozeß des verteilten Systems samt seiner sendenden Kanäle gleichgesetzt.**
- **Ein inaktiver Prozeß des Systems mit gemeinsamem Speicher entspricht im verteiltemSystem einem inaktiven Prozeß mit leeren sendenden Kanälen.**
- **Um festzustellen, ob ein Kanal leer ist, werden Bestätigungsnachrichten verwendet.**
- **Für jeden Prozeß u wird ein Prädikat u.e vorausgesetzt mit**

  u.e detects $\forall$**v**((**u**, **v**) $\in$ **E** $\Rightarrow$ ((**u**, **v**).**c** = **null**)),

  d. h. jeder Prozeß kann feststellen, ob alle seine Nachrichten angekommen sind.
- **Einführung der Variablen b analog vorangehender Vorgehensweise mit folgender Invarianten:**

  invariant v $\in$ **b** $\Rightarrow$ ([(**v.delta = empty**) $\wedge$ **v.idle** $\wedge$ **v.e**] $\vee$ $\langle$$\exists$**u: u** $\notin$ **b :: u chain v**$\rangle$)

❑ **Abgeleitetes Programm**

Program `{distributed termination detection - transformed program} R4`

initially

`claim = false[] b = empty [] ⟨[] u :: u.delta = empty ⟩`

assign

```
{u sends message m along channel (u,v).c if u is active,
 and u.delta is updated}
⟨[] (u, v) ::
      (u, v).c, u.delta := (u,v).c;m, u.delta ∪ {v}
                                          if  ¬u.idle ⟩
{u receives a message along channel (u, v)
 after which it becomes active}
[] ⟨[] (u, v) ::
     v.idle, m, (u, v).c := false, head((u, v).c), tail((u, v).c)
                                          if (u, v).c = null ⟩
[] ⟨[] u :: u.idle := true ⟩
   {added statements}
[] ⟨[] u :: b, u.delta := b ∪ {u} - u.delta, empty
                                  if u.idle ∧ u.e ⟩
[] claim := (b = V)
```

end `{R4}`

8.5

Beschreibung (Spezifikation) eines Problems durch Gleichungen

Beispiel: Byzantinische Verständigung

❑ **Notation:**

- u, v, w bezeichnen fehlerfreie Prozessoren
- x, y, z bezeichnen fehlerhafte Prozessoren
- g bezeichnet den Anführer

❑ **Spezifikation 1**

```
byz[u] = byz[v]
byz[u] = d⁰[g] if reliable(g)
```

Dabei ist $d^0[g]$ der Anfangswert einer bestimmten Variablen des Anführers.

❑ Zu konstruieren ist ein Gleichungssystem, das den Bedingungen des initially-Abschnitts genügt und somit unmittelbar ein Programm zur Lösung des Verständigungsproblems liefert, das im Hinblick auf verschiedene Architekturen modifiziert werden kann.

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig

❑ **Spezifikation 2**

- t Anzahl der fehlerhaften Prozesse
- $con^r[u, x]$ enthält den Wert, den in der r-ten Runde x an u gesandt hat.
- $d^r[u]$ ist der Wert, den u in der r-ten Runde für den Einigungswert hält.

Definition: $con^r[u, *] = \langle +x : con^r[u, x] :: 1 \rangle$

(B1) $\langle \wedge u: u \neq g :: \neg d^0[u] \rangle \wedge \langle \wedge u, x :: \neg con^0[u, x] \rangle$

(A1) $con^r[u, v] = d^{r-1}[v]$

(A2) $con^{r-1}[u, x] \Rightarrow con^r[v, x]$

(E1) $d^r[u] = [d^{r-1}[u] \vee (con^r[u, *] \geq r \wedge con^r[u, g])]$

(E2) $byz[u] = d^{t+1}[u]$

◆ **Nachweisbar ist: Spezifikation 2 $\Rightarrow$ Spezifikation 1**

◆ **Spezifikation 2 kann als Lösung mittels beglaubigter Nachrichten angesehen werden.**

Universität Erlangen-Nürnberg, IMMD IV, F. Hofmann
Reproduktion jeder Art oder Verwendung dieser Unterlage zu Lehrzwecken außerhalb der Universität Erlangen-Nürnberg ist ohne Genehmigung des Autors unzulässig