# 2. Übung zur Vorlesung Software-Produktlinien

TECHNISCHE UNIVERSITÄT DARMSTADT

**Lösungsvorschläge**

## Lösung 1

### Lösung 1.1

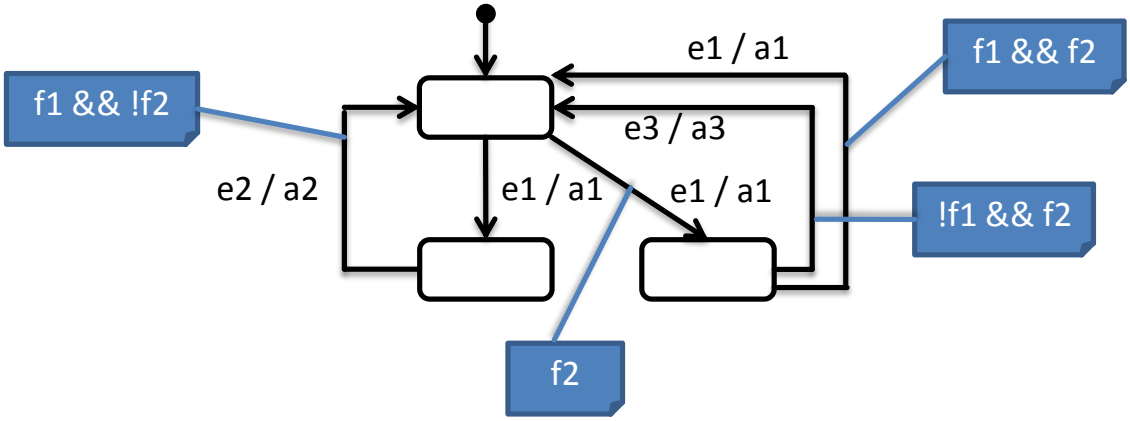


**Abbildung 1:** 150% State Machine Modell

*Hinweise:*

- Die Lösung ist nicht eindeutig und hängt u.a. davon ab, wie die Transitionen, die mit $e1/a1$ beschriftet sind, identifiziert werden.
- Für die nicht gegebene Modellvariante der Konfiguration $P0 = \{f0\}$ wird angenommen, dass sie die Gemeinsamkeiten der drei gegebenen Modellvarianten enthält.

### Lösung 1.2

- (a) nicht wohlgeformt:
  - Für die Konfiguration $\{f0, f2\}$ hat die annotierte Transition keinen Startzustand.
  - Für die Konfiguration $\{f0\}$ hat die Initial-Transition keinen Zielzustand. Alternativ könnte auch angenommen werden, dass die Initial-Transition nur vorhanden ist, wenn auch der Default-Zustand vorhanden ist.

(b) nicht wohlgeformt:

- Die Annotation der Transition ganz rechts ist nicht erfüllbar.

- (a) nicht möglich:
  - Da $f1$ und $f2$ in Konflikt stehen, kann keine passende Annotation gefunden werden, sodass die Transition immer einen Start- und Zielzustand hat.

  (b) möglich:
  - Umgebenden Zustand annotieren mit $f1 \&\& f2$
  - Transition annotieren mit $f1 \&\& f2$

- Lösung siehe Abbildung 2. *Hinweis:* Falls das Tool keine Guards für Default-Transitionen zulässt, müssen alle Transitionen des rechten Unterautomaten um den Guard $\varphi 3$ erweitert werden.

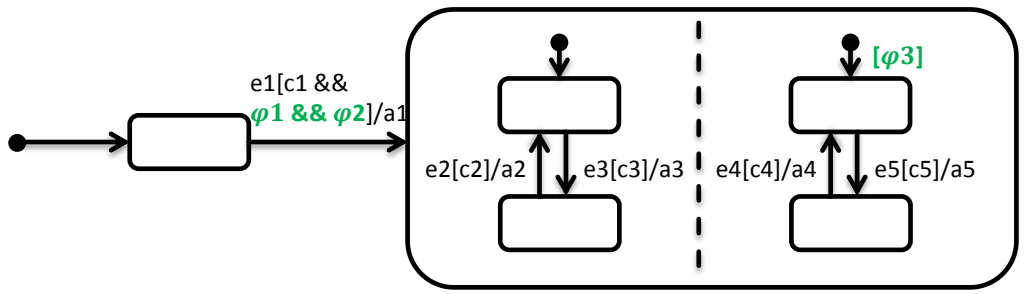


**Abbildung 2:** Transformiertes State Machine Modell mit Variability Encoding

- Zusammenhängender Zustands-Transitions-Graph:

  a) Für alle Zustände $e \in$ *Zustände* in Unterautomat $e' \in$ *Unterautomaten* mit Initialzustand $e_0 \in$ *Zustände* betrachte die Menge der Transitionspfade:

$$\begin{aligned} e_0^0 \xrightarrow{e_1^0} e_2^0 \xrightarrow{e_3^0} \cdots \xrightarrow{e_{k-1}^0} e_k^0 &= e \\ e_0^1 \xrightarrow{e_1^1} e_2^1 \xrightarrow{e_3^1} \cdots \xrightarrow{e_{k-1}^1} e_k^1 &= e \\ \vdots &\quad \vdots \\ e_0^l \xrightarrow{e_1^l} e_2^l \xrightarrow{e_3^l} \cdots \xrightarrow{e_{k-1}^l} e_k^l &= e \end{aligned}$$

  mit

$$\widetilde{FM} \wedge \left( \bigwedge_{0 \le i \le k} \varphi_i^j \right) \not\models \texttt{false}$$

für alle $0 \leq j \leq l$. Dann muss gelten:

$$\vdash \varphi \rightarrow \left( \bigvee_{0 \leq j \leq l} \left( \bigwedge_{0 \leq i \leq k} \varphi_i^j \right) \right)$$

*Anmerkung:* Die Anzahl $k$ an Modellsegmenten muss nicht zwingend für die verschiedenen Pfade gleich sein und wurde hier nur mehrfach verwendet, um die Definition kompakt zu halten.

*Hinweis zur Implementierung:* Die Suche nach einem Pfad kann durch (wiederholte) Tiefensuche implementiert werden. Aufgrund der Wohlgeformtheitseigenschaften für Transitionen ist es ausreichend, entweder nur die Bedingungen der Zustände oder der Transitionen zu betrachten.

b) Die Eigenschaft gilt nicht: In der Konfiguration $\{f\,0, f\,2\}$ ist der obere Zustand vorhanden, aber nicht erreichbar.

## Lösung 1.3

- Modellelemente von BMSC und deren Beziehungen:
  - *Instanzen* mit Namen und Lebenslinien.
  - Menge von *Locations* auf einer Lebenslinie, unterschieden in Send- und Receive-Locations, mit vertikaler Totalordnung.
  - *Nachrichten* mit Namen. Nachrichten sind eindeutig definiert durch ihre Sende- und Empfangs-Locations. Verschiedene Nachrichten mit gleichen Nachrichtennamen sind zulässig. Jede Location ist entweder Sende- oder Empfangs-Location genau einer Nachricht, Selbst-Nachrichten sind erlaubt (Sender- und Empfänger-Instanz sind identisch, nicht aber die Locations).
  - Globale Wohlgeformtheit: die transitive Hülle der Vereinigung der Totalordnung der Locations pro Instanz-Linie und der Ordnung der Locations verschiedener Instanzen durch die Nachrichten-Ordnung (Senden einer Nachricht passiert vor dessen Empfang) ergibt eine Halbordnung (Zyklenfreiheit der globalen Ordnung von Locations).
- Instanzen, Locations, Nachrichten. Ähnlich wie bei Transitionen von State Machine Modellen wäre es ausreichend, lediglich Nachrichten zu annotieren:
  - Locations sind nur vorhanden, wenn die zugehörige Nachricht vorhanden ist.
  - Instanzen sind nur vorhanden, wenn mindestens eine zugehörige Location vorhanden ist.
- Wohlgeformtheitseigenschaften:
  - Alle Annotationen erfüllen das Feature-Modell (optional).
  - Instanzen sind nur vorhanden, wenn mindestens eine Location auf der zugehörigen Lebenslinie vorhanden ist.
  - Locations sind nur vorhanden, wenn die zugehörige Instanz vorhanden ist.

- Locations sind nur vorhanden, wenn die zugehörige Nachricht vorhanden ist.
- Nachrichten sind nur vorhanden, wenn die zugehörigen Locations vorhanden sind.
- Die globale Location-Ordnung aller BMSC Varianten sind zyklenfrei.

Beispiele:

- (a) nicht wohlgeformt.
- (b) wohlgeformt.

## Lösung 2

a) Choice-Term:

```
[i:int,f:float] optimum ([i:int,f:float] x, [i:int,f:float] y) {
  if (x [min:<,max:>] y)
    return x;
  else
    return y;
}
```

Hinweis zu Baumdarstellung in Abbildung 3: Der Baum ist zwecks Übersichtlichkeit an einigen Stellen vereinfacht dargestellt.

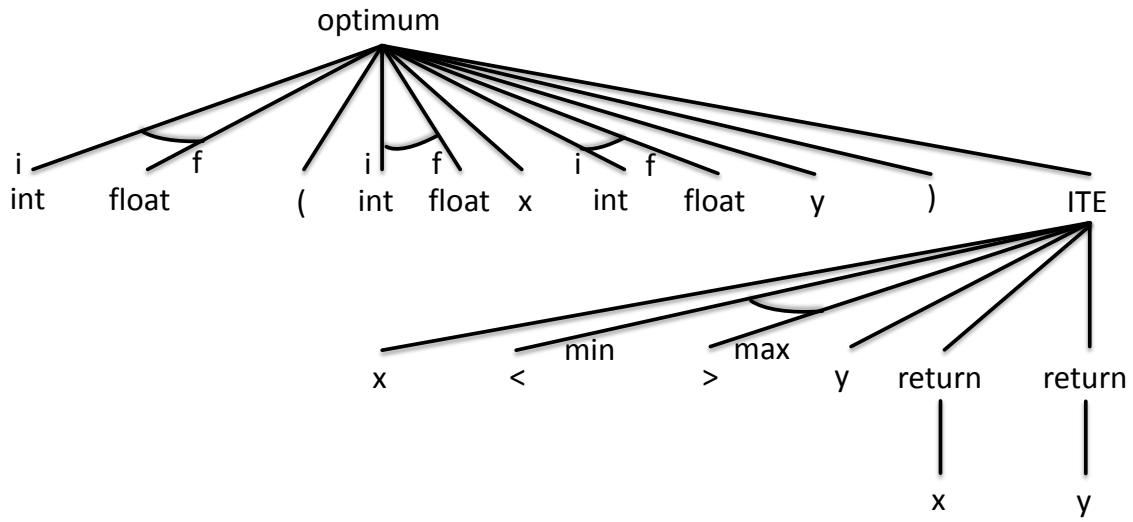


**Abbildung 3:** Baumdarstellung des Choice-Terms

b) Erste Anwendung der Choice Elimination Semantics ergibt 4 Varianten:

Variante (i,min):

```
int optimum (int x, int y) {
  if (x < y)
```

```
    return x;
  else
    return y;
}
```

Variante (f,min):

```
float optimum (float x, float y) {
  if (x < y)
    return x;
  else
    return y;
}
```

Variante (i,max):

```
int optimum (int x, int y) {
  if (x > y)
    return x;
  else
    return y;
}
```

Variante (f,max):

```
float optimum (float x, float y) {
  if (x > y)
    return x;
  else
    return y;
}
```

Erste Anwendung der Distributions-Regel auf die erste Dimension:

```
[i:int,f:float] optimum ([i:int x,f:float x], [i:int y,f:float y]) {
  if (x [min:<,max:>] y)
    return x;
  else
    return y;
}
```

Anwendung der Distributions- und Merge-Regel auf die erste Dimension:

```
[i:int,f:float] optimum ([i:int x, int y,f:float x, float y]) {
  if (x [min:<,max:>] y)
    return x;
  else
    return y;
}
```

Anwendung der Distributions-Regel auf die zweite Dimension:

```
[i:int,f:float] optimum ([i:int,f:float] x, [i:int,f:float] y) {
  if ([min:x<y,max:x>y])
    return x;
  else
    return y;
}
```

usw.

Per Definition ergibt die erneute Auswertung des Terms durch die Choice Elimination Semantics nach jeder Anwendung der Distributions-Regel die gleiche Varianten-Menge wie vor der Anwendung.

c)
```
dim Type[i,f] in
dim Impl[min,max] in
let v = Type[int,float] in
v optimum (v x, v y) {
  if (x Impl[<,>] y)
    return x;
  else
    return y;
}
```

d) Gemeinsame Auswahl-Labels für beide Dimensionen:

```
[i:int,f:float] optimum ([i:int,f:float] x, [i:int,f:float] y) {
  if (x [i:<,f:>] y)
    return x;
  else
    return y;
}
```

Anwendung der Choice Elimination Semantics ergibt nun 2 Varianten:

Variante i:

```
int optimum (int x, int y) {
  if (x < y)
    return x;
  else
    return y;
}
```

Variante f:

```
float optimum (float x, float y) {
  if (x > y)
```

```
    return x;
  else
    return y;
}
```

## Lösung 3

- Lösung siehe Abbildung 4. Vorgehensweise:
  - Für alle annotierten Elemente, die nicht im Kern enthalten sind, `add` Delta mit Application Condition entsprechend der annotierten Presence Condition einfügen.
  - Für alle annotierten Elemente, die im Kern enthalten sind, `rem` Delta mit Application Condition entsprechend der *negierten* Presence Condition einfügen.
  - Application Conditions gegebenenfalls vereinfachen
  - Zustandsnamen einführen, um den Anwendungskontext zu definieren.

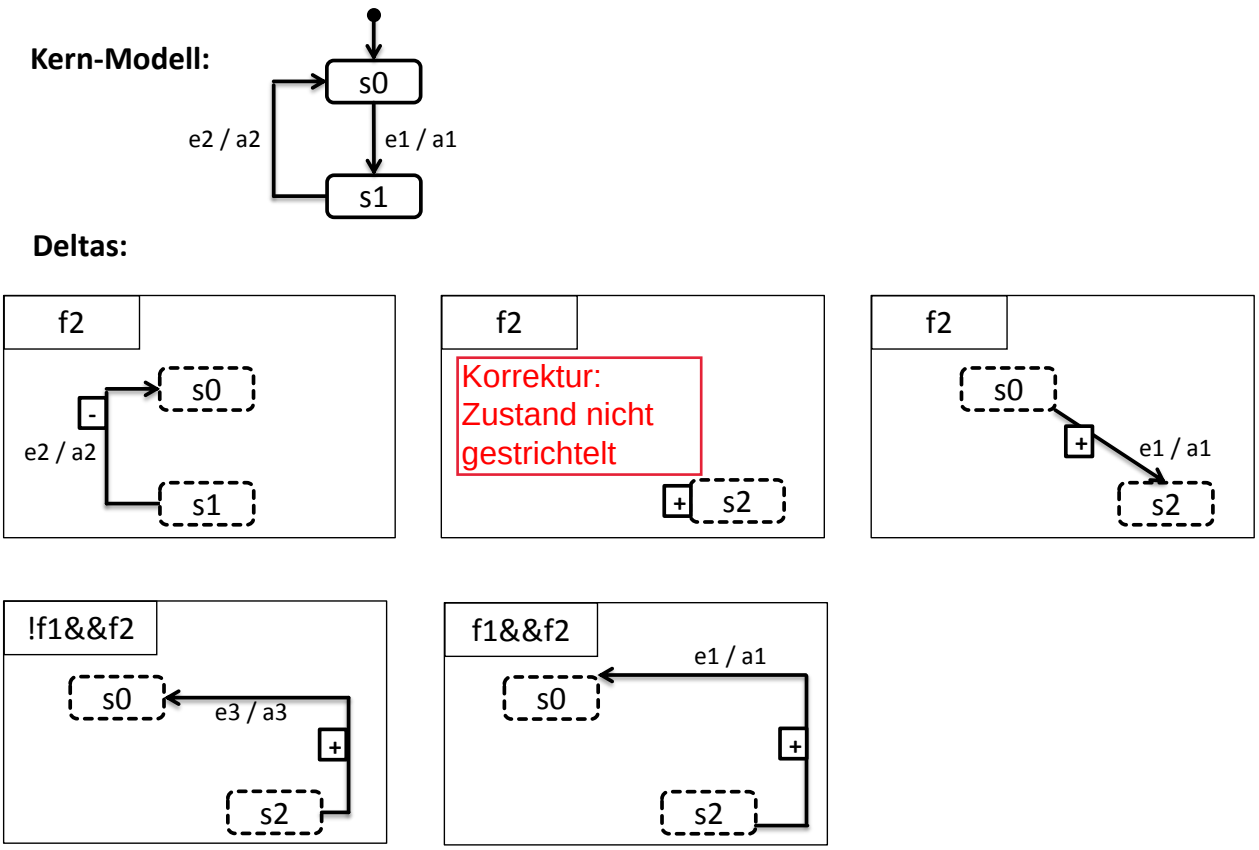


**Abbildung 4:** Delta State Machine Modell für eine 150% State Machine

- Lösung siehe Abbildung 5. Konflikte: Das Delta `MT` muss vor dem Delta für `P` angewendet werden.

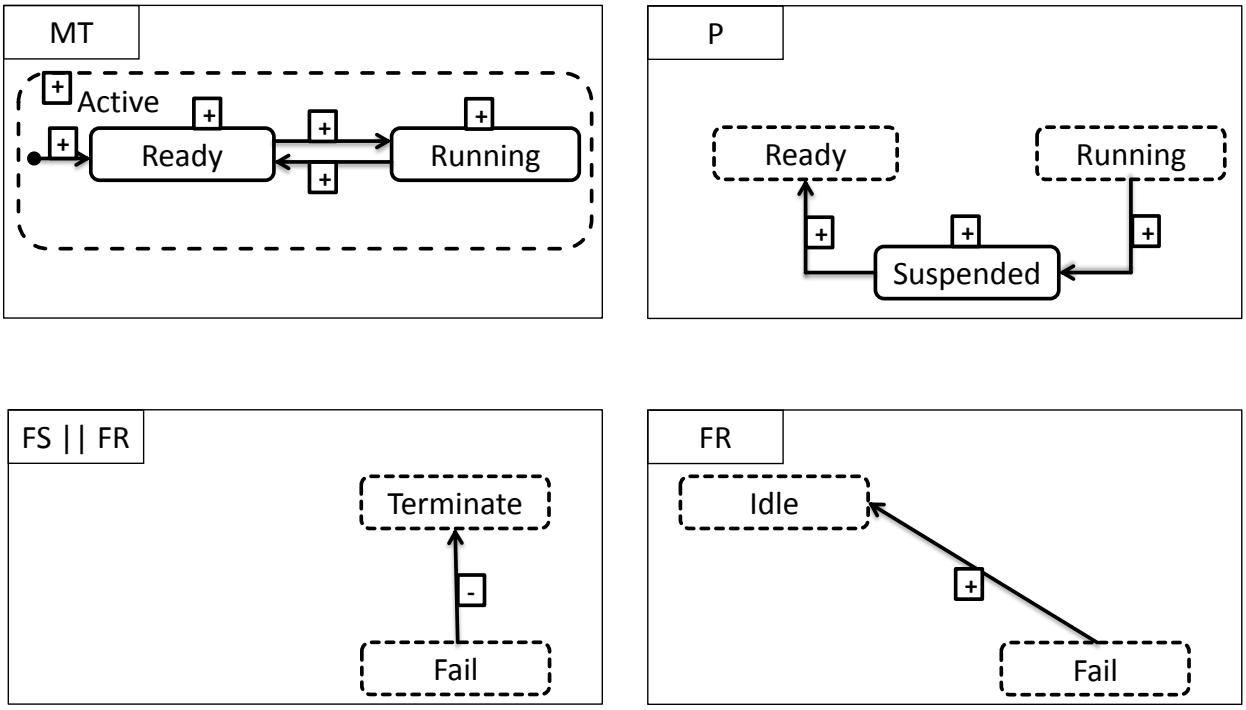


**Abbildung 5:** Delta State Machine Modell für eine Scheduler-Produktlinie

- Definition von `add` und `remove` Deltas für:
  - Instanzen (kein Kontext notwendig)
  - Locations (Kontext ist direkte Vorgänger- und Nachfolge-Location auf der Instanzlinie. Hierfür z.B. zusätzliche (Pseudo-)Start- und End-Locations pro Instanz einführen).
  - Nachrichten (Kontext ist Sende- und Empfangs-Location)

  Reihenfolge zur Auflösung von Konflikten:
  - Einfügen: Instanzen, Locations, Nachrichten
  - Löschen: Nachrichten, Locations, Instanzen

## Lösung 4

- Die Deltas $x_2$ und $x_4$ stehen in Konflikt, da sie nicht geordnet sind unter $\prec$.
  (1) Ordnung zwischen $x_2$ und $x_4$ erzwingen: $x_2 \prec x_4$ oder $x_4 \prec x_2$ möglich.
  (2) Setze $D := D \cup \{x_5\}$ mit $x_2 \prec x_5$, $x_4 \prec x_5$ und
  $$x_5 \cdot x_2 \cdot x_4 = x_5 \cdot x_4 \cdot x_2.$$
- Sei $c = y(\mathbf{0})$ mit $y = y_1 \cdot y_2 \cdots y_k$. Dann soll für jede beliebige Modellvariante $p = z(\mathbf{0})$ mit $z = z_1 \cdot z_2 \cdots z_l$ gelten, dass ein Delta $x$ existiert mit $p = x(c)$. *Konstruktiver Beweis:*
  - Sei $Y = \{y_1, y_2, \ldots, y_k\}$ und $Z = \{z_1, z_2, \ldots, z_l\}$.
  - Setze $X_Y = (Y \setminus Z)$, $X_Z = (Z \setminus Y)$ und $X = X_Z \cup \{x^{-1} | x \in X_Y\}$.
  - Dann gilt: $p = x(c)$ wobei $x = x_1 \cdot x_2 \cdots x_m$ eine Linearisierung von $X$ ist.