2. Übung zur Vorlesung Software-Produktlinien

TECHNISCHE UNIVERSITÄT DARMSTADT

Variabilitätsmodellierung im Lösungsraum

Aufgabe 1 Annotative Variabilitätsmodellierung

Aufgabe 1.1 Erstellen eines annotierten Modells

Gegeben seien die State Machine Modellvarianten in Abbildung 1 für die drei Produktkonfigurationen $P1$, $P2$ und $P3$. Geben Sie hierfür ein (möglichst kleines) 150% State Machine Modell mit annotierten Transitionen an, aus dem sich genau diese Varianten ableiten lassen.

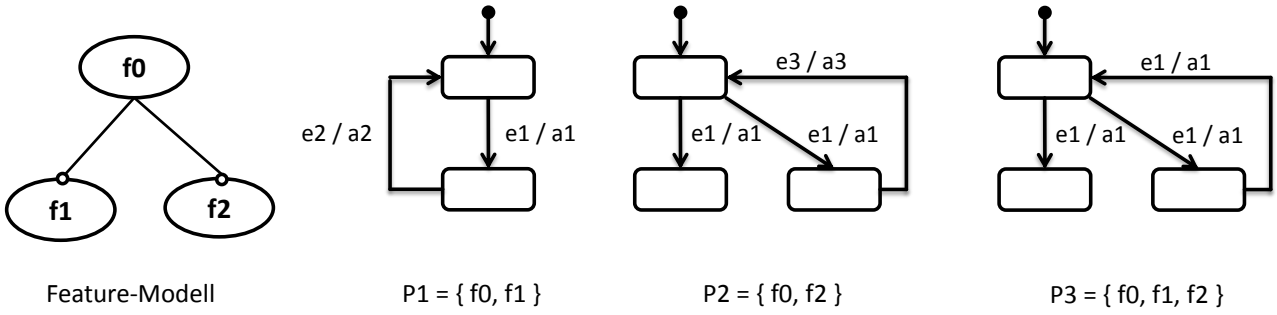

Abbildung 1: State Machine Modellvarianten und zugehöriges Feature-Modell

Aufgabe 1.2 Analyse annotierter Modelle

- Prüfen Sie die annotierten State Machine Modelle in Abbildung 2 auf Wohlgeformtheit.
- Komplettieren Sie, falls möglich, die partiell annotierten State Machine Modelle in Abbildung 3.
- Das annotierte State Machine Modell in Abbildung 4 soll in einem Tool bearbeitet werden, das nur die Annotationen von Transitionen als Teil der Transitions-Guards (Variability Encoding) erlaubt. Transformieren Sie das Modell entsprechend.
- Neben den in der Vorlesung betrachteten Wohlgeformtheitseigenschaften für State Machine Modelle kann eine weitere Forderung darin bestehen, dass jeder Zustand in einem Unterautomaten einer Modellvariante über mindestens einen Transitionspfad innerhalb dieser Modellvariante erreichbar ist.
  a) Formulieren Sie eine Regel zur Prüfung dieser Eigenschaften auf einem 150% Modell.
  b) Prüfen Sie diese Eigenschaft für das Modell in Abbildung 5.

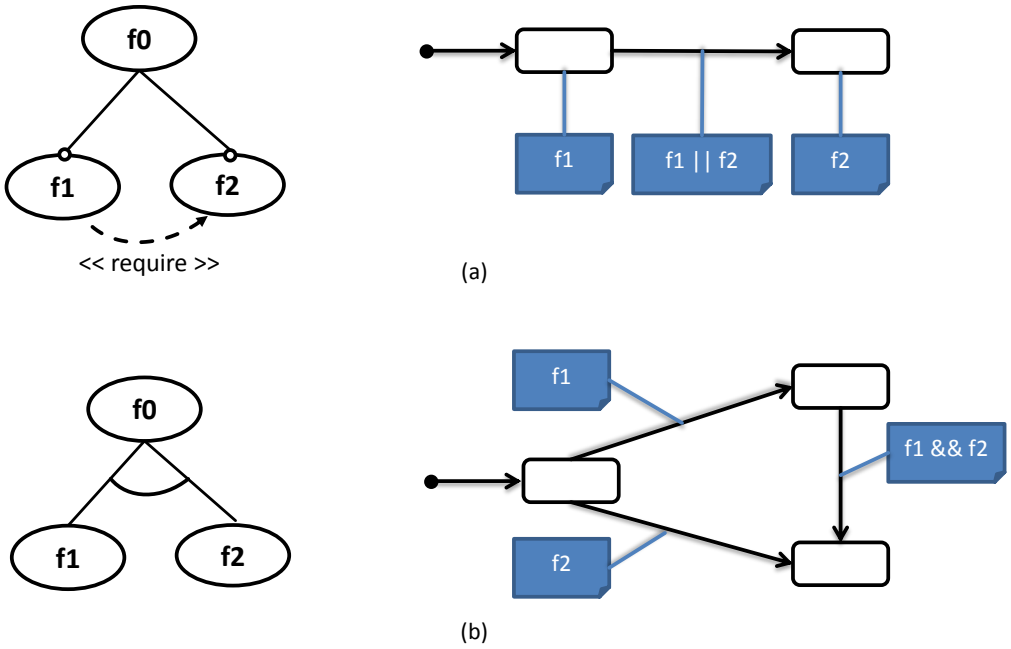

Abbildung 2: Beispiele für annotierte State Machine Modelle

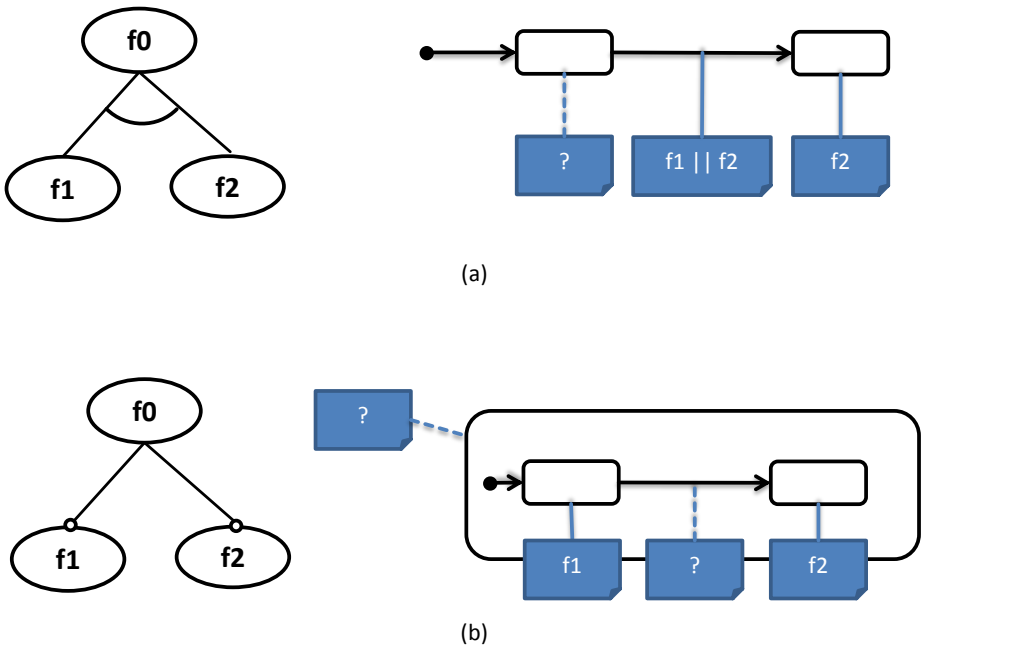

Abbildung 3: Beispiele für partiell annotierte State Machine Modelle

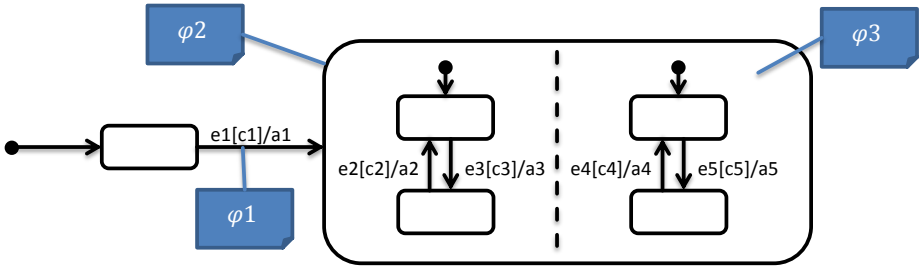

Abbildung 4: Beispiel für ein annotiertes State Machine Modell

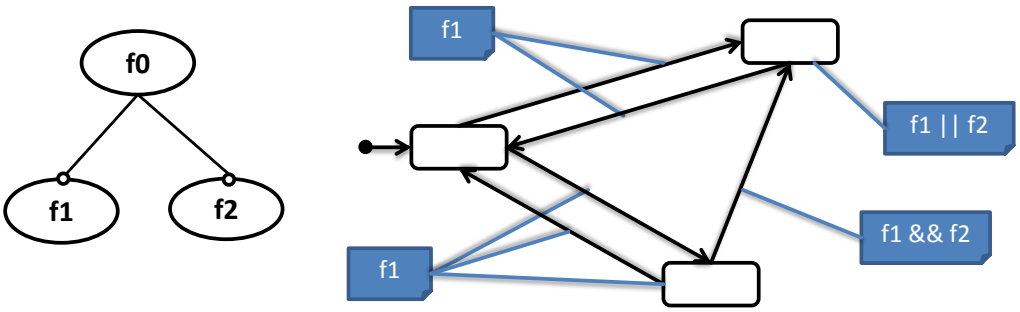

Abbildung 5: Beispiele für ein annotiertes State Machine Modell

Aufgabe 1.3 Entwurf annotierter Modellierungssprachen

Für die Spezifikation von Interaktionens-Szenarien zwischen Systemkomponenten werden in der UML *Sequenzdiagramme* verwendet, die auf *Basic Message Sequence Charts* (BMSC) basieren. Abbildung 6 zeigt den grundlegenden Aufbau eines BMSC. Für die Modellierung va-

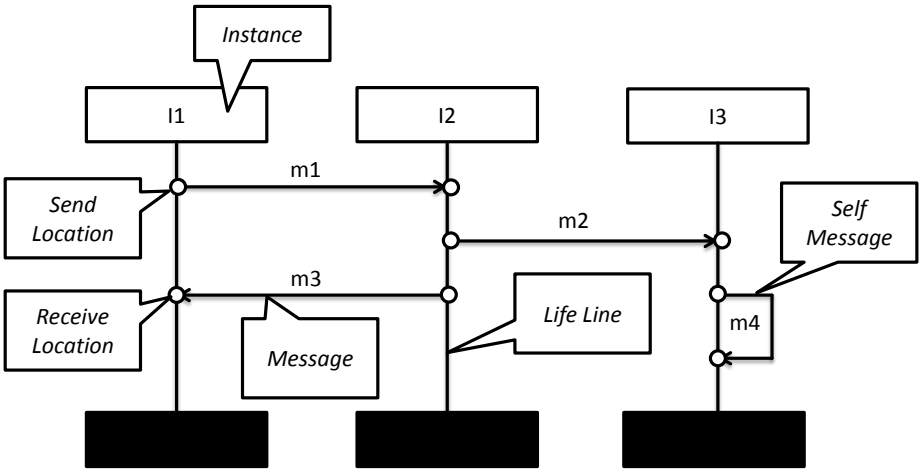

Abbildung 6: Bestandteile eines Basic Message Sequence Charts

riabler Interaktions-Szenarien einer Produktlinie sollen BMSC um Annotationen mit Feature-Bedindungen erweitert werden.

- Erstellen Sie eine Liste der Modellelemente, aus denen sich BMSC zusammensetzen und beschreiben Sie die Beziehung zwischen diesen Elementen.
- Überlegen Sie, welche Modellelemente eines BMSC für die Variblitätsmodellierung annotiert werden sollten.
- Welche Wohlgeformtheitskriterien gelten für annotierte BMSC? Welche der in Abbildung 7 abgebildeten annotierten BMSC sind wohlgeformt?

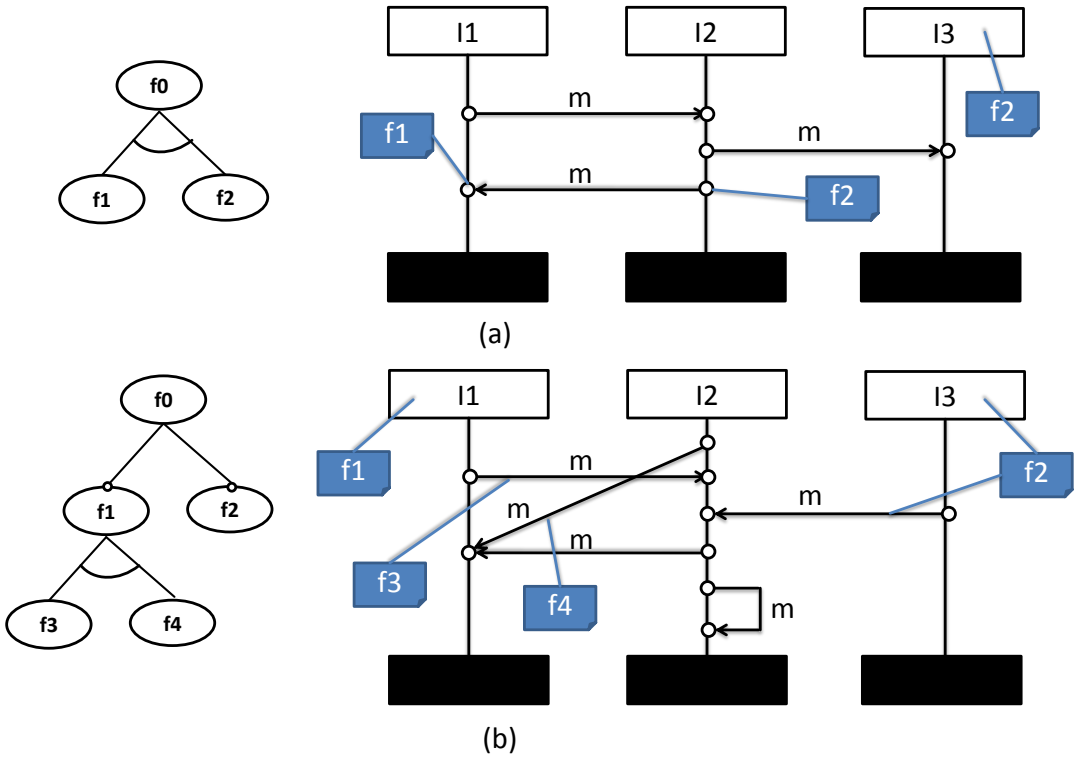

Abbildung 7: Beispiele für annotierte BMSC

Aufgabe 2 Choice Calculus

a) Definieren Sie eine variable Funktion `optimum` in C-Syntax mithilfe des Choice-Kalküls, die zwei Zahlen `x` und `y` als Parameter übernimmt und das Optimum dieser beiden Zahlen zurückgibt. Die Funktion soll zwei unabhängige Variabilitätsdimensionen haben:

- Schnittstellenvariabilität: Die Funktion soll entweder Zahlen vom Typ `int` oder vom Typ `float` verarbeiten.
- Funktionalitätsvariabilität: Die Funktion soll entweder die maximale oder die minimale der beiden Zahlen ermitteln und zurückgeben.

Geben Sie zum Choice-Term auch eine entsprechende Baumdarstellung an.

b) Wenden Sie auf den Term aus Aufgabe a) die *Choice Elimination Semantics* einmal vor und einmal nach der Anwendung der Distributions-Regel an.

c) Deklarieren und verwenden Sie für den Term aus Aufgabe a) nun entsprechende Dimensionen `Type[i,f]` und `Impl[min,max]`. Binden Sie die Dimension `Type[i,f]` and die Variable `v`.

d) Ändern Sie den Term so, dass für zwei Zahlen vom Typ `float` stets das maximum berechnet wird und für zwei Zahlen vom Typ `int` stets das minimum berechnet wird. Wenden Sie auf diesen Term die *Choice Elimination Semantics* erneut an.

Aufgabe 3 Delta-orientierte Variabilitätsmodellierung

- Erstellen Sie für das 150% State Machine Modell aus Aufgabe 1.1 ein entsprechendes Delta State Machine Modell mit der Modellvariante von $P1$ als Kernmodell.
- In Abbildung 8 ist das Feature-Modell eines einfachen Betriebssystem-Schedulers sowie eine State Machine Variante abgebildet, die den Lebenszyklus eines Scheduler-Tasks für eine Scheduler-Konfiguration beschreibt. Definieren Sie State Machine Deltas für die Ableitung der weiteren Varianten aus der gegebenen Variante. Hierfür soll gelten:
  - Ein aktiver Task in einem *Multi-Threading* System wird vom Scheduler vom Zustand `Ready` in Zustand `Running` und wieder zurück versetzt, um dem Task Rechenressourcen zu erteilen bzw. wieder zu entziehen.
  - In einem System mit *Preemptive Scheduling* kann ein Task durch einen höherprioren Task vom Zustand `Running` in den Zustand `Suspended` versetzt werden. Dieser Task muss zunächst wieder in den Zustand `Ready` wechseln, um seine Ausführung fortzusetzen.
  - Im Falle von *Fail Safe* verbleibt ein Task nach Auftreten eines Fehlers im Zustand `Fail` und erfordert einen Neustart des Systems.
  - Im Falle von *Fail Recover* wechselt ein Task nach Auftreten eines Fehlers zurück in den Zustand `Idle`.
- Definieren Sie für die annotierten BMSC aus Aufgabe 1.3 eine entsprechende *Delta BMSC* Sprache. Wie sollte die Delta-Ordnung auf BMSC gewählt werden, um Konflikte aufzulösen?

Aufgabe 4 Abstract Delta Modeling

Gegeben sei das abstrakte Delta Model $(D, \prec)$ mit $D = \{x_1, x_2, x_3, x_4\}$. Es gelte $x_1 \prec x_2$, $x_1 \prec x_4$ und $x_3 \prec x_4$ sowie $x_2 \cdot x_4 \neq x_4 \cdot x_2$.

- Enthält das Delta Modell Konflikte? Falls ja, lösen sie diesen Konflikt durch (1) Erweiterung der Relation $\prec$, (2) durch Hinzunahme eines konfliktlösenden Deltas x_5 auf.
- Für das inverse Delta x^{-1} eines Deltas x gilt: $x \cdot x^{-1} = x^{-1} \cdot x = \epsilon$ Beweisen Sie: Falls im Deltoid $(\mathscr{D}, \cdot, \epsilon)$ für jedes Delta $x \in \mathscr{D}$ das Inverse existiert, dann kann jedes beliebige Modell $c = y(\mathbf{0})$ als Kernmodell gewählt werden.

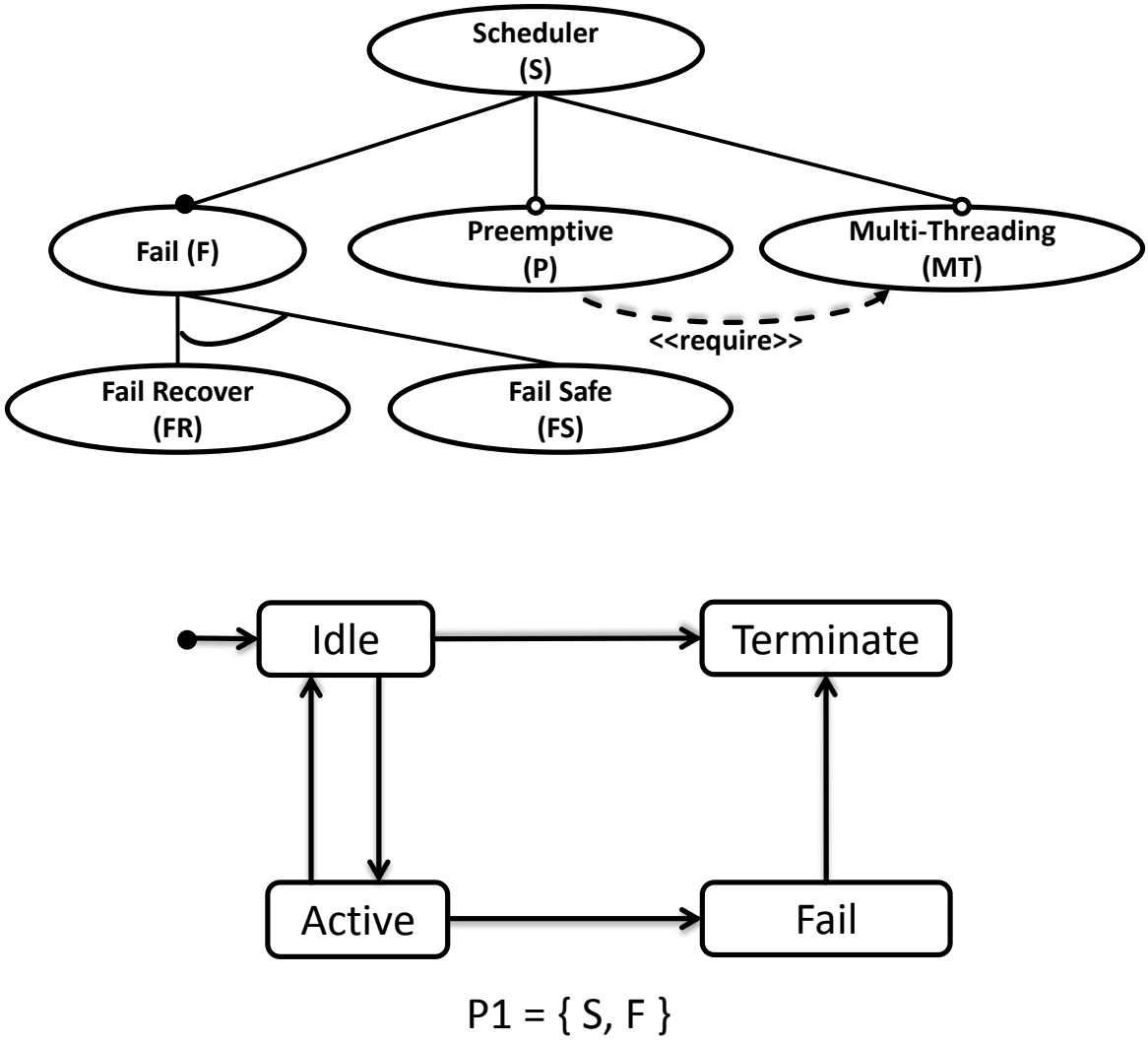

Abbildung 8: State Machine Modellvariante eines Schedulers und zugehöriges Feature-Modell