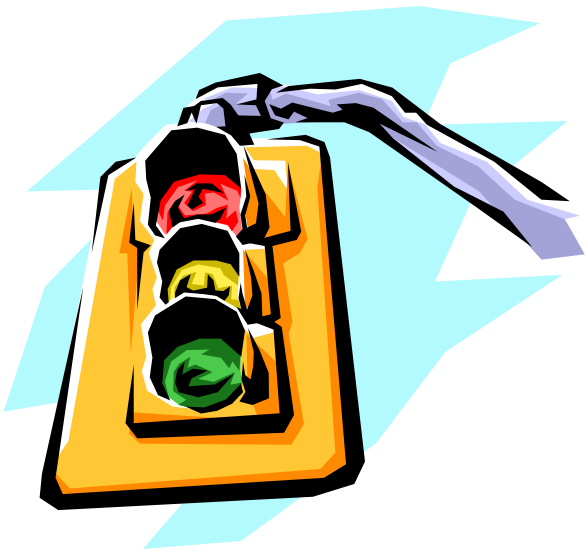
# Softwarepraktikum

## Teil: Eingebettete Systeme

Sommersemester 2003

# Analyse II:

# Verhalten (Zustandsautomaten)



Aufgabe 2

# Analyse II: Verhalten (Zustandsautomaten)

**Umfang:** **2 Wochen**
**Punkte:** **100 P.**

Nachdem in der ersten Aufgabe die Systemstruktur mit Hilfe von Klassen- und Instanzendiagrammen beschrieben wurde und eine erste Analyse des Systemverhaltens zur Aufstellung von Sequenzdiagrammen geführt hat, soll in dieser Aufgabe das Verhalten vollständig spezifiziert werden. Anstelle von Sequenzdiagrammen, die nur ausgewählte Szenarien aufzeigen können, werden daher **Zustandsdiagramme** (**Statecharts**) [Har98][For01] verwendet, welche Zustandsautomaten beschreiben. Zunächst soll das lokale Verhalten der einzelnen Lichtsignalanlagen modelliert werden. Um das Konzept der „Grünen Welle“ realisieren zu können, wird anschließend das globale Verhalten beschrieben. Diese Vorgehensweise erleichtert diesen Analyseschritt, da weniger komplexe Teilsysteme separat betrachtet werden können („**Teile-und-Beherrsche**“-Prinzip).

## 1 Lokales Verhalten: Zustandsdiagramme

Ein Zustandsdiagramm besteht aus der Beschreibung von

- **Zuständen** (**States**), in denen sich das (Teil)-System befinden kann,
- **Ereignissen** (**Events**) oder **Signalen**, die Reaktionen des (Teil)-Systems auslösen,
- **Bedingungen** (**Conditions**), die eine Einschränkung für einen Zustandswechsel beschreiben, und
- **Transitionen**, die Reaktionen auf Ereignisse beschreiben, indem sie Beziehungen zwischen den Zuständen mit eventuellen Bedingungen festlegen.

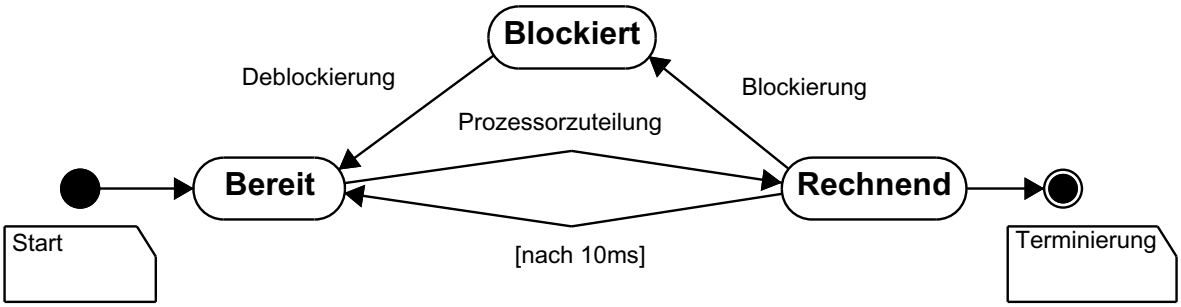



Abbildung 12 Zustandsdiagramm eines einfachen Prozessmodells

Abbildung 12 zeigt als Beispiel ein Zustandsdiagramm eines einfachen Prozessmodells. Ein Prozess kann sich demnach in den drei Zuständen *Bereit*, der Prozess wartet auf Zuteilung eines Prozessors, *Rechnend*, dem Prozess ist momentan ein Prozessor zugeordnet, und *Blockiert*, der Prozess hat einen blockierenden Aufruf (z. B. eine I/O-Operation) ausgeführt, befinden. Nach der Erzeugung des Prozesses befindet sich dieser im Zustand *Bereit* (der ausgefüllte Kreis kennzeichnet das **Startsymbol**). Dem Prozess wird nach Ablauf einer Zeitscheibe von 10 ms der Prozessor entzogen (modelliert durch die Bedingung *[nach 10ms]*), sofern er den Prozessor nicht bereits freiwillig mittels Aufruf einer blockierenden Operation abgegeben hat. Ist eine Transition z. B. mit *E[B]* versehen, so wird bei Eintritt des Ereignisses *E* überprüft, ob Bedingung *B* wahr ist. Nur in diesem Fall wird die Transition vollzogen. Ist eine Transition wie in Abbildung 12 lediglich mit einer Bedingung versehen, so findet der Zustandsübergang statt, sobald die Bedingung wahr ist. Insbesondere können so zeitliche Bezüge modelliert werden. Wenn der Prozess seine Auf-

gabe erledigt hat, so terminiert er, was durch die Transition zum **Terminierungssymbol** ausgedrückt ist.

In Steuerungssystemen treten häufig Ereignisse ein, auf die erst später reagiert werden kann. Solche Ereignisse müssen dann zwischengespeichert werden. Diese Zwischenspeicherung kann mittels Einführung neuer Zustände erfolgen, die neben ihrer eigentlichen Zustandsinformation auch den Eintritt der entsprechenden Ereignisse beinhalten. Um eine Zustandsexplosion zu vermeiden, können **„parallele“ Zustandsdiagramme** genutzt werden. Für jedes zu speichernde Ereigniss wird ein separates Zustandsdiagramm modelliert, dessen Zustände festlegen, ob ein Ereignis eingetreten ist, auf welches noch reagiert werden muss.

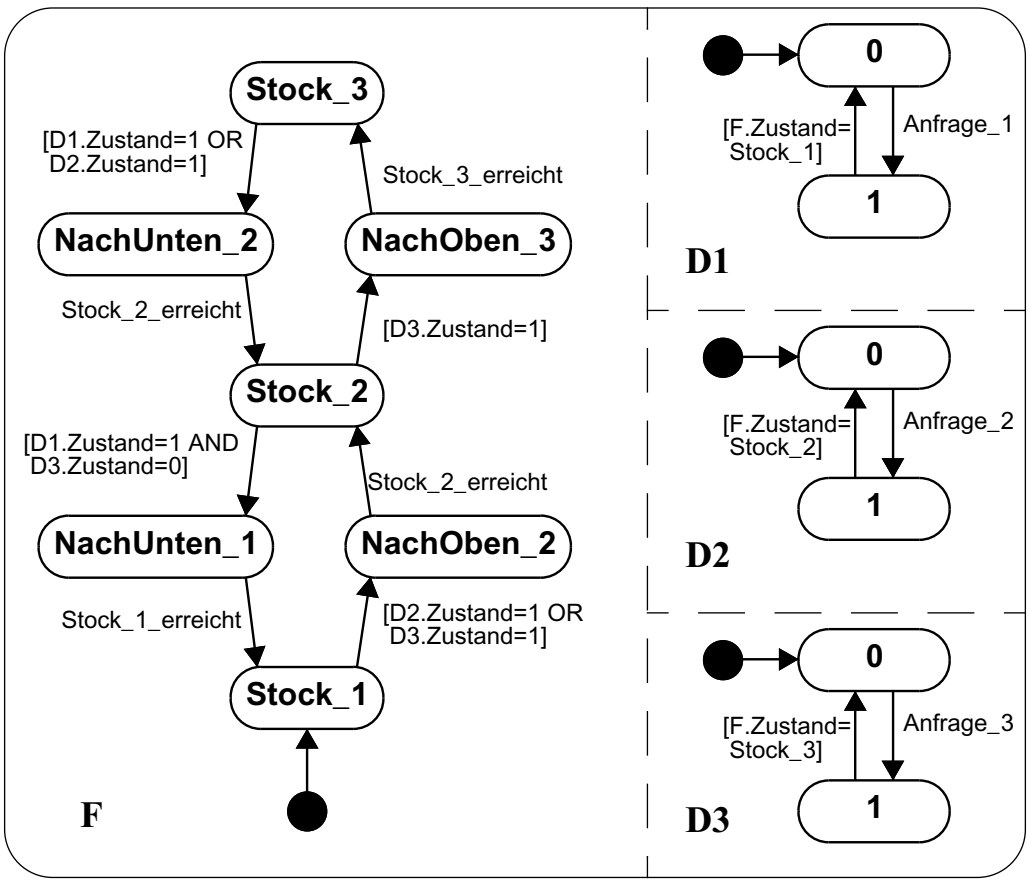



Abbildung 13 „Paralleles“ Zustandsdiagramm einer Fahrstuhlsteuerung

Abbildung 14 zeigt als Beispiel das Verhalten einer einfachen Fahrstuhlsteuerung unter Verwendung der „parallelen“ Zustandsdiagramme F, D1, D2 und D3. D1, D2 und D3 halten fest, ob eine Anfrage im entsprechenden Stockwerk vorliegt, auf die noch reagiert werden muss. F selbst reagiert nicht auf die Anfrage-Ereignisse, sondern überprüft, in welchen Zuständen sich D1, D2 und D3 befinden. Das dazugehörige Klassendiagramm zeigt Abbildung 14.

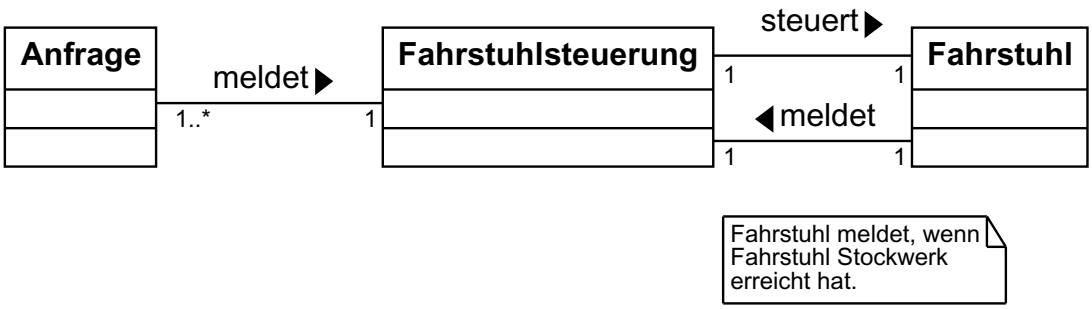
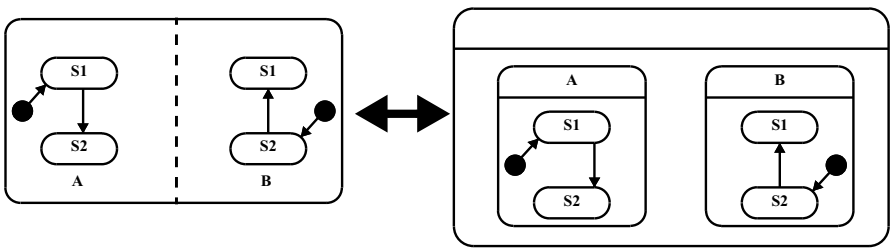



Abbildung 14 Klassendiagramm eines einfachen Fahrstuhlsystems

### 1.1 Aufgabe (50 Punkte)

Modellieren Sie das lokale Verhalten der einzelnen Lichtsignalanlagen mit Hilfe von Zustandsdiagrammen[1]. Überlegen Sie sich zunächst mögliche Phasen und Phasenfolgen. Jede Phase kann direkt auf einen Zustand abgebildet werden, wobei in den Zustandssymbolen anstelle der Bezeichnung der einzelnen Zustände direkt deren Ausgaben und Aktivitäten geschrieben werden können. Mittels Transitionen werden dann Zustandsübergänge entsprechend der Phasenfolgen modelliert. Für die eigentliche Steuerung stehen Detektor- und

1. Telelogic Tau erlaubt leider nicht, „parallele“ Zustandsdiagramme wie in Abbildung 13 zu modellieren. Daher soll die folgende, als äquivalent definierte Notation eingesetzt werden:



Timer-Ereignisse zur Verfügung. Beachten Sie, dass Signalgeber ausfallen können. Solche Fehlfunktionen werden mittels Ereignissen der Steuerung mitgeteilt.

## 2 Globales Verhalten: „Kommunizierende“ Zustandsdiagramme

Abbildung 15 zeigt die Zustandsdiagramme eines modulo-16-Zählers. Für jedes Bit des Zählers wurde ein separates Zustandsdiagramm erstellt. Die Überträge werden anhand von Ereignissen an das jeweilige nächste Bit weitergeleitet, wobei das Versenden eines Ereignisses mit „^“ gekennzeichnet wird. Die Zustandsdiagramme „kommunizieren“ somit über das Versenden von Ereignissen. Im Gegensatz zu „parallelen“ Zustandsdiagrammen, welche die Zustände *eines* Objekttyps in einer „parallelen“ Notation beschreiben, wird bei „kommunizierenden“ Zustandsdiagrammen das Verhalten *verschiedener* Objekttypen beschrieben[1].

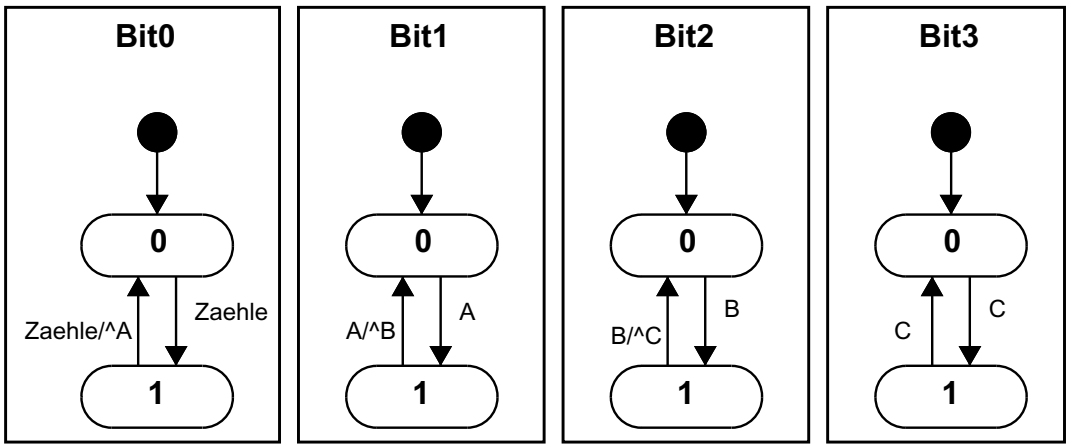



Abbildung 15 Zustandsdiagramme der verschiedenen (Bit-)Objekttypen eines Modulo-16-Zählers

### 2.1 Aufgabe (50 Punkte)

Modellieren Sie das Konzept der „Grünen Welle“, indem Sie die Zustandsdiagramme der einzelnen Lichtsignalanlagen erweitern. Beachten Sie diesmal, dass für das Erstellen von Verkehrsstatistiken

1. „Parallele“ und „kommunizierende“ Zustandsdiagramme implizieren verschiedene Entwurfsentscheidungen. „Kommunizierende“ Zustandsdiagramme lassen die Umsetzung der Objekte in verschiedenen Prozessen und sogar deren räumliche Trennung zu.

und die Anzeige von Fehlfunktionen Informationen zur Benutzerkonsole geschickt werden müssen.

## 3 Literatur

[Har98] D. Harel, M. Politi. *Modeling Reactive Systems with Statecharts: The Statemate Approach*. New York: McGraw-Hill. 1998

[For01] P. Forbig. *Objektorientierte Softwareentwicklung mit UML*. Leipzig: Fachbuchverlag Leipzig. 2001