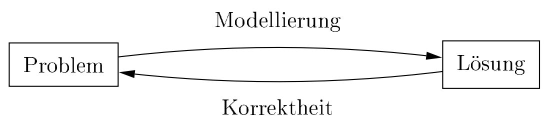
Prof. Dr. Hans-Jörg Kreowski
Studiengang Informatik
Universität Bremen
MZH 3260
Tel.: 2956, 3697 (Sekr.), Fax: 4322
E-Mail: kreo@informatik.uni-bremen.de

Oktober 2002

Theoretische Informatik II
Sommersemester 2003

Zum Sinn Theoretischer Informatik am Beispiel einer korrekten Modellierung mit Hilfe endlicher Automaten

In vielen Gebieten der Informatik geht es um die Modellierung von Datenverarbeitungsproblemen und ihren (korrekten) Lösungen.



Am Beispiel eines sehr einfachen eingebetteten Systems soll illustriert werden, was das konkret heißen kann. Ziel ist die Modellierung einer Heizung (bzw. ihrer Steuerung), die heizt, wenn die Raumtemperatur zu niedrig ist. Etwas detaillierter ist folgende Aufgabe zu bewältigen:

Modelliere eine Heizung,

(1) die angeschaltet werden kann und dann betriebsbereit ist,

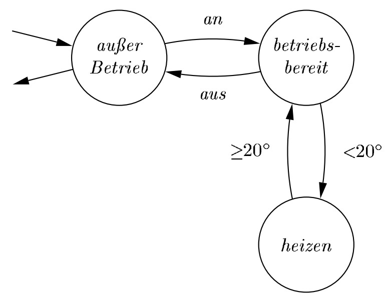
(2) die bei einer Temperatur unter 20° heizt,

(3) die wieder betriebsbereit wird, wenn die Temperatur mindestens 20° erreicht hat, und

(4) die ausgeschaltet werden kann, wenn sie nicht gerade heizt.

Ziel der Modellierung ist, diese informelle textuelle Beschreibung so zu präzisieren und zu explizieren, dass eine computerausführbare Lösung entsteht.

Der textuellen Beschreibung ist zu entnehmen, dass die Heizung mindestens drei verschiedene Zustände haben kann: *heizen, betriebsbereit* und *außer Betrieb*, wobei das den Zustand vor dem Anschalten und nach dem Ausschalten bezeichnet. Außerdem ist von vier Ereignissen die Rede, die eintreten können: anschalten (*an*), ausschalten (*aus*), Temperatur sinkt unter 20° (<20°), Temperatur steigt auf mindestens 20° (≥20°). Die Ereignisse ändern Zustände, was in der folgenden visuellen Darstellung festgehalten ist:



Die visuelle Darstellung ist ein sogenannter Zustandsgraph mit den Zuständen als Knoten, zu erkennen an den eingekreisten Namen, und den durch Ereignisse eintretenden Zustandsübergängen als Kanten, die als Pfeile gezeichnet sind mit dem Ereignis als Markierung am Pfeil, dem Pfeilende beginnend am Zustand vor dem Ereignis und der Pfeilspitze endend am Zustand nach dem Ereignis. Der Zustandsgraph präzisiert die textuelle Beschreibung in zweierlei Hinsicht:

(5) Der Zustand vor dem Anschalten ist derselbe wie nach dem Ausschalten.

(6) Dieser Zustand wird als Anfang und Ende aller Vorgänge im Zusammenhang mit der Heizung betrachtet, was einerseits durch den Pfeil, der von keinem Zustand ausgeht, und andererseits durch den Pfeil, der auf keinen Zustand zeigt, graphisch dargestellt ist.

Der Zustandsgraph spezifiziert die bei der modellierten Heizung *zulässigen* Abläufe, die aus allen Ereignisfolgen bestehen, die vom Anfangs- zum Endzustand führen. Das sind alle Folgen $u_1 u_2 \dots u_n$ für $n \geq 1$, die aus *an*-*aus*-Zyklen u_i für $i =$

$1, \ldots, n$ bestehen. Dabei ist ein *an*-*aus*-Zyklus eine Ereignisfolge, die mit *an* beginnt, mit *aus* endet und dazwischen immer abwechselnd $<20°$ und $\geq 20°$ durchläuft, d.h. die Form

$$an <20° \geq 20° <20° \geq 20° \cdots <20° \geq 20° \, aus$$

hat. Außerdem wird die leere Folge als *zulässig* betrachtet.

Bezeichnet man nun den Zustandsgraphen als *heating* und die Menge aller zulässigen Abläufe als $L(heating)$, dann kann man *heating* als Modell betrachten, dessen Verhalten durch $L(heating)$ festgelegt ist und das damit die gestellte Aufgabe löst.

Aber ist die Lösung auch korrekt? Wird wirklich das gestellte Problem gelöst? Genau genommen ist *heating* einfach konstruiert und dann zur Lösung erklärt worden. Dass alles richtig ist, bleibt der Intuition überlassen, was bei kleinen Problemen noch angeht. Aber was passiert bei Hunderten von Zuständen und Tausenden von Übergängen? Da kann die Intuition schnell versagen. Glücklicherweise kann man aber noch einen Schritt weiterkommen, indem man dem Lösungsmodell noch ein explizites Modell des Problems gegenüberstellt.

Um das von der bisherigen Beschreibung abzusetzen, soll es dadurch geschehen, dass angegeben wird, welche Ereignisfolgen bei der Heizung verboten sein sollen. Eine Ereignisfolge gilt als *verboten*, wenn sie mindestens eine der folgenden Teilfolgen enthält:

(a) *an* $\geq 20°$

(b) $<20°$ *aus*

(c) $<20°$ $<20°$

(d) $\geq 20°$ $\geq 20°$

(e) *an u an*

(f) *aus u aus*

wobei u eine Ereignisfolge ist, die nur $<20°$ und $\geq 20°$ enthält. *Verboten* sind auch Folgen, die (g) nicht mit *an* beginnen oder (h) nicht mit *aus* enden. Wenn $L_{forbidden}$ die Menge aller verbotenen Ereignisfolgen bezeichnet, dann lässt sich diese Menge als Präzisierung des Problems durch negative Anforderungen auffassen, durch die ausgedrückt wird, was nicht passieren soll. Bezogen auf diese Anforderungen ist *heating* ein korrektes Modell, wenn keine zulässige Ereignisfolge verboten ist. Mit anderen Worten, *heating* ist *korrekt* bzgl. $L_{forbidden}$, wenn $L(heating) \cap L_{forbidden} = \emptyset$. Tatsächlich trifft das auch zu, denn vom Start aus muss man mit *an* beginnen (g) und zum Ende kommt man nur mit *aus* (h). Und

$\geq 20°$ kann nur eintreten, wenn $<20°$ direkt vorausgeht (a & d), so wie nach $<20°$ nur $\geq 20°$ eintreten kann (b & c). Schließlich kann man *an* nur erhalten, wenn man mit *aus* zum Anfang zurückgekehrt ist (e), und ein weiteres *aus* setzt ein vorheriges *an* voraus (f).

Insgesamt ist damit eine Modellierung gelungen, bei der das Problem durch die Menge $L_{forbidden}$ der verbotenen Ereignisfolgen präzisiert und die Lösung durch den Zustandsgraphen *heating* modelliert ist, dessen Verhalten als die Menge $L(heating)$ der zulässigen Ereignisfolgen bestimmt ist. Die Lösung ist in dem Sinne korrekt, dass keine bei *heating* zulässige Ereignisfolge in der Problembeschreibung verboten ist.

Eine interessante Frage an dieser Stelle ist, ob Korrektheit automatisch überprüft werden kann. Es wird sich im Laufe der Lehrveranstaltung herausstellen, dass diese Frage bejaht werden kann, wenn man die negativen Anforderungen wie die Lösung mit Hilfe von Zustandsgraphen beschreibt. Tatsächlich definieren Zustandsgraphen endliche Automaten, wie sie demnächst näher untersucht werden. Endliche Automaten gehören zu den einfachsten algorithmischen Instrumenten, die in der Informatik eine wichtige Rolle spielen. Endliche Automaten erzeugen sogenannte Sprachen wie beispielsweise die Mengen der zulässigen und verbotenen Ereignisfolgen. Es wird unter anderem gezeigt, dass der Durchschnitt zweier solcher Sprachen wieder eine solche Sprache ist und dass die Leerheit einer solchen Sprache algorithmisch festgestellt werden kann, womit das Korrektheitsproblem in diesem Falle gelöst wäre. Diese Art der Verifikation wird Modelchecking genannt und seit einigen Jahren äußerst erfolgreich in der Praxis eingesetzt. In anderen Zusammenhängen ist es allerdings häufig sehr viel schwieriger Korrektheit sicherzustellen, so dass darauf oft verzichtet wird. Es muss jedoch beachtet werden, dass Modelle, die nicht sicher korrekt sind, Fehler machen können — teure Fehler oder vielleicht sogar fatale Fehler. Sich um Korrektheit zu bemühen, kann sich also lohnen. Sie setzt allerdings immer den Einsatz mathematischer Methoden voraus, weil nur dann ein echter Nachweis möglich ist.