Ludwig-Maximilians-Universität München **SoSe 2012**
Institut für Informatik
Prof. Dr. C. Prehofer, C. Kroiß

Formale Methoden des Softwareengineering
Übungsblatt 6
Besprechung am 15.06.2012

Aufgabe 1: Objekt-orientiertes Modellieren in Maude

Gegeben sei das folgende Modul in Full Maude (auch auf der Homepage der Vorlesung erhältlich). Eine Beschreibung des vordefinierten Moduls `NAT-LIST` gibt es im Maude Manual unter `http://maude.cs.uiuc.edu/maude2-manual/html/maude-manualch7.html`.

Listing 1: COMM.maude

```
(omod COMM is
    protecting NAT-LIST .
    protecting QID .
    sort Nat? .
    subsort Nat < Nat? .
    subsort Qid < Oid .

    class Buffer | q : NatList, reader : Oid .
    class Sender | cell : Nat?, cnt : Nat, receiver : Oid .
    class Receiver | cell : Nat?, cnt : Nat .

    op mt : -> Nat? .

    msg to_from_val_cnt_ : Oid Oid Nat Nat -> Msg .
    op start : Oid Oid Oid NatList -> Configuration .

    vars E N M : Nat .
    var L : NatList .
    vars X Y Z : Oid .

    --- Beachte: Nicht immer tauchen alle Attribute auf

    rl [read] : < X : Buffer | q : L E, reader : Y >
                < Y : Sender | cell : mt, cnt : N >
        => < X : Buffer | q : L, reader : Y >
           < Y : Sender | cell : E, cnt : N + 1 > .
    rl [send] : < Y : Sender | cell : E, cnt : N, receiver : Z >
        => < Y : Sender | cell : mt, cnt : N, receiver : Z >
           to Z from Y val E cnt N .
    rl [receive] : < Z : Receiver | cell : mt, cnt : N >
                   to Z from Y val E cnt M
        => < Z : Receiver | cell : E, cnt : N + 1 > .
```

```
33
    eq start(X, Y, Z, L) =
        < X : Buffer | q : L, reader : Y >
        < Y : Sender | cell : mt, cnt : 0, receiver : Z >
37      < Z : Receiver | cell : mt, cnt : 0 > .
endom)

--- Maude manual: 16.5 Tracing commands
41 set trace on .
set trace body off .

--- Interaktion:
45 --- (red start('x, 'y, 'z, 2 1) .)
--- (rew start('x, 'y, 'z, 2 1) .)
```

(a) Geben Sie eine Regel `write` an, mit deren Hilfe der Empfang gepuffert wird.

(b) Modifizieren Sie das Modul so, dass die Pakete immer in der richtigen Reihenfolge in den Empfangspuffer gestellt werden.

Aufgabe 2: Türme von Hanoi in Maude

In dieser Aufgabe soll das Problem der Türme von Hanoi mit Maude gelöst werden.

(a) Erstellen Sie ein **nicht objektorientiertes** Systemmodul `HANOI`, das die Regeln und Abläufe des Spiels gemäß der Aufgabe vom letzten Blatt modelliert. Verwenden Sie dabei geeignete Labels für die Regeln, um später nachvollziehen zu können, welche Schritte durchgeführt werden.

Die Anzahl der Türme sei wie immer festgesetzt auf drei. Die Anzahl der Scheiben soll jedoch flexibel sein. Integrieren Sie dazu einen geeigneten Konstruktor, der die Anzahl N an Scheiben als Parameter erhält und daraus einen geeigneten Anfangszustand aufbaut.

(b) Geben Sie an, wie anhand des Moduls `HANOI` mit Maude automatisch eine Lösung für das Rätsel mit 4 Scheiben gefunden werden kann, d.h. einen sequentiellen Ablauf von Schritten, so dass der Turm mit 4 Scheiben gemäß den Regeln vollständig von A nach C transportiert wird. Kopieren Sie dazu einfach die durchgeführten Maude-Befehle und die dazugehörige Ausgabe in eine Text-Datei.

(c) Erstellen Sie jetzt ein objektorientiertes Modul in Full-Maude, in dem die Türme als Objekte repräsentiert werden und das Versetzen der Scheiben durch geeigenete Nachrichten.

Erstellen Sie nun eine Konfiguration, die den Anfangszustand enthält und eine Reihe von Nachrichten, die genau die Schritte der in der letzten Teilaufgabe gefundenen Lösung repräsentieren. Überprüfen Sie anschließend, dass aus dieser Anfangskonfiguration durch Term Rewriting tatsächlich der gewünschte Zielzustand erzeugt werden kann.

Tipp: Verwenden Sie dazu das `search`-Kommando und nutzen Sie die Tatsache, dass sich Konfigurationen wie andere Terme auch mit dem Operator `==` vergleichen lassen.