

# Übung 9 – Objekte und Türme in Maude

Formale Techniken in der Software-Entwicklung

SoSe 2012

Christian Kroiß



## Aufgabe 6-1

```
(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 > .
```



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



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

**Gebraucht wird zusätzlich ein Puffer für empfangene Nachrichten:**

```
class RBuffer | q : NatList .
```

**Die Regel `write` soll nun jeweils den Wert im Speicher des Receivers nehmen und in den Rbuffer schreiben:**

```
rl [write] : < R : Receiver | cell : E, rbuffer : RB >  
            < RB : RBuffer | q : Es >  
=> < R : Receiver | cell : mt >  
    < RB : RBuffer | q : E Es > .
```



b) Modifizieren Sie das Modul so, dass die Pakete immer in der richtigen Reihenfolge in den Empfangspuffer gestellt werden. Beschreiben Sie in einem Kommentar, wie Sie das bewerkstelligen.

Es reicht bei receive darauf zu achten, dass die Nachricht mit der korrekten Sequenznummer gelesen wird:

```
cr1 [receive] : < R : Receiver | cell : mt, cnt : N >  
              to R from S val E cnt M  
              => < R : Receiver | cell : E, cnt : N + 1 >  
if M = N + 1 .
```



## a) nicht objektorientiertes Systemmodul

```
mod HANOI is
  protecting NAT .
  protecting BOOL .
  sorts Tower Disc Config .
  sort Disc < Tower .
  op empty : -> Tower [ctor] .
  op _ _ : Tower Tower -> Tower [assoc comm ctor id: empty] .
  op <_,_,_> : Tower Tower Tower -> Config [ctor] .
  op d(_) : Nat -> Disc [ctor] .
  op init_tower : Nat -> Tower .
  op init : Nat -> Config .
  op maxdisc : Tower -> Nat .
```

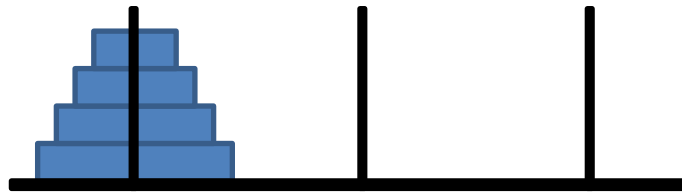


```
var N : Nat .  
vars T1 T2 T3 : Tower .  
var D : Disc .  
  
eq init_tower(1) = d(1) .  
ceq init_tower(N) = init_tower(sd(N,1)) d(N) if N > 1 .  
eq init(N) = < init_tower(N), empty, empty > .  
  
eq maxdisc(empty) = 0 .  
eq maxdisc(T1 d(N)) = max(maxdisc(T1),N) .
```

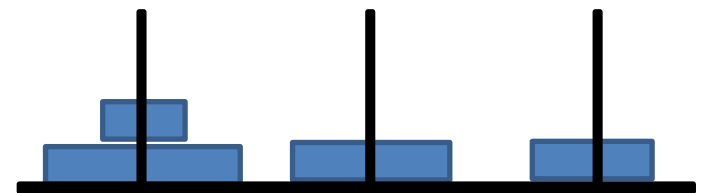
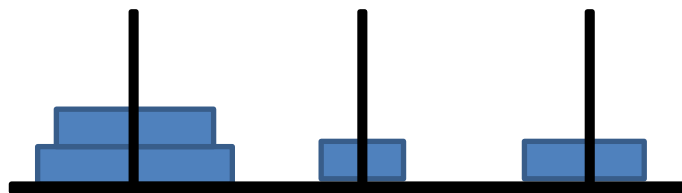
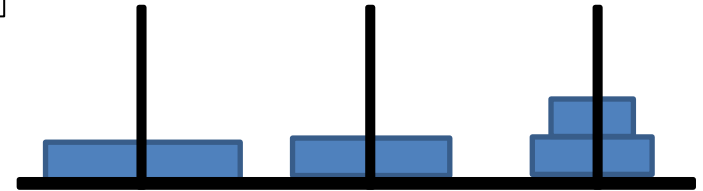
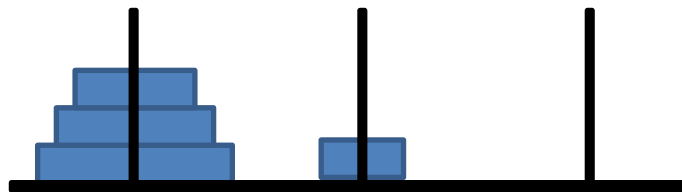
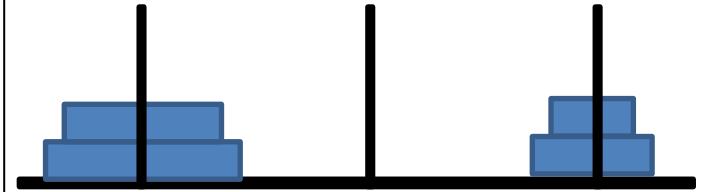


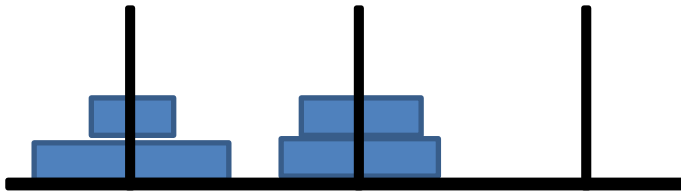


```
crl [A_B] : < T1 d(N), T2, T3 > => < T1, T2 d(N), T3 >  
         if maxdisc(T1) < N /\ maxdisc(T2) < N .  
crl [A_C] : < T1 d(N), T2, T3 > => < T1, T2, T3 d(N) >  
         if maxdisc(T1) < N /\ maxdisc(T3) < N .  
crl [B_A] : < T1, T2 d(N), T3 > => < T1 d(N), T2, T3 >  
         if maxdisc(T2) < N /\ maxdisc(T1) < N .  
crl [B_C] : < T1, T2 d(N), T3 > => < T1, T2, T3 d(N) >  
         if maxdisc(T2) < N /\ maxdisc(T3) < N .  
crl [C_A] : < T1, T2, T3 d(N) > => < T1 d(N), T2, T3 >  
         if maxdisc(T3) < N /\ maxdisc(T1) < N .  
crl [C_B] : < T1, T2, T3 d(N) > => < T1, T2 d(N), T3 >  
         if maxdisc(T3) < N /\ maxdisc(T2) < N .  
  
endm
```

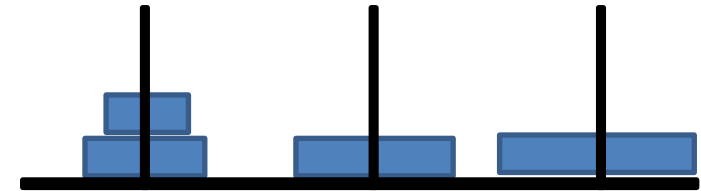
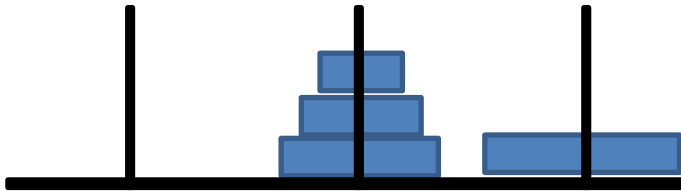
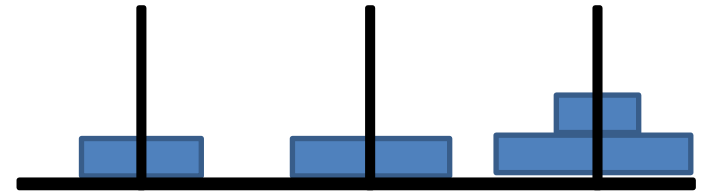
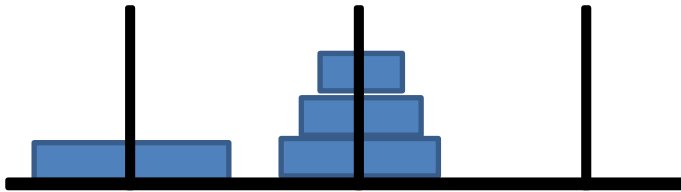
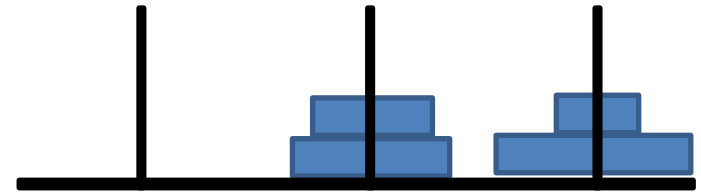


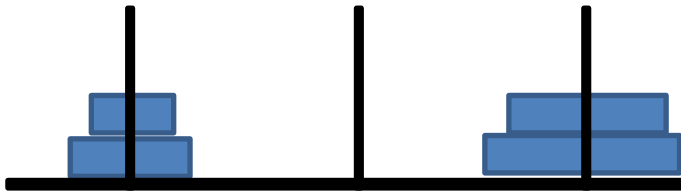
- 1)  $A \rightarrow B$
- 2)  $A \rightarrow C$
- 3)  $B \rightarrow C$
- 4)  $A \rightarrow B$
- 5)  $C \rightarrow A$



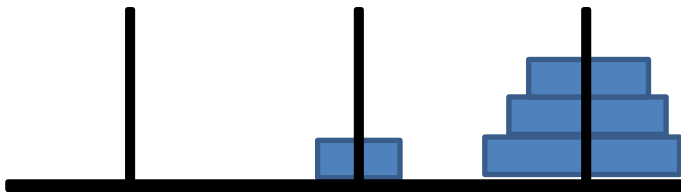
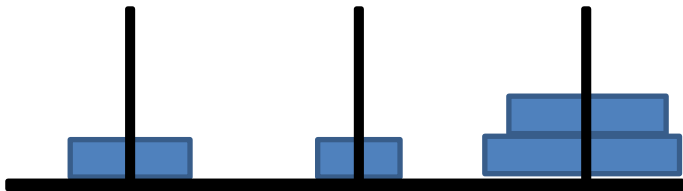
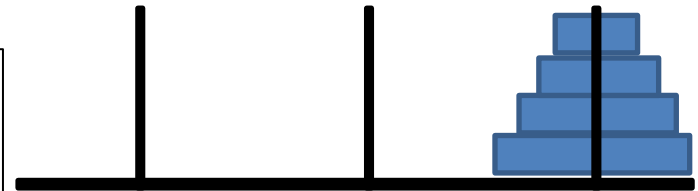


- 6)  $C \rightarrow B$
- 7)  $A \rightarrow B$
- 8)  $A \rightarrow C$
- 9)  $B \rightarrow C$
- 10)  $B \rightarrow A$
- 11)  $C \rightarrow A$





- 12)  $B \rightarrow C$
- 13)  $A \rightarrow B$
- 14)  $A \rightarrow C$
- 15)  $B \rightarrow C$





## b) Objektorientiertes Modul

```
(omod OHANOI is
  protecting QID .
  protecting INT .

  protecting SORTABLE-LIST{Int<} .

  subsort Qid < Oid .

  class Tower | c : List{Int<} .
```



```
msg move : Oid Oid -> Msg .  
op initial : -> Configuration .
```

```
eq initial = < 'A : Tower | c : 1 2 3 4 >  
            < 'B : Tower | c : nil > < 'C : Tower | c : nil > .
```

```
op max : List{Int<} -> Nat .
```

```
vars A B : Oid .  
var N : Nat .  
vars L1 L2 : List{Int<} .
```

```
eq max(nil) = 0 .  
eq max(L1 N) = max(max(L1),N) .
```



```
cr1 [move] :  
    move (A, B)  
    < A : Tower | c : L1 N >  
    < B : Tower | c : L2 >  
=>  
    < A : Tower | c : L1 >  
    < B : Tower | c : L2 N >  
    if max(L1) < N /\ max(L2) < N .  
  
endom)
```