Foundations of System Development

Martin Wirsing

in cooperation with
Axel Rauschmayer

WS 05/06

Vergleich am Beispiel der Uhr: Maude und TLA

1. Uhr in TLA

- **TLA spezifiziert die Menge aller möglichen (unendlichen) Abläufe der Uhr:**

$$
\begin{aligned}
IClk &\equiv hr \in \{0,\ldots,23\} \wedge min \in \{0,\ldots,59\} \\
Min &\equiv min < 59 \wedge min' = min + 1 \wedge hr' = hr \\
Hr &\equiv min = 59 \wedge min' = 0 \wedge hr' = (hr + 1) \bmod 24 \\
Tick &\equiv Min \vee Hr \\
Clock &\equiv IClk \wedge \Box[Tick]_{hr,min} \wedge \mathrm{WF}_{hr,min}(Tick)
\end{aligned}
$$

2. Uhr in Maude (algebraische Spez.)

- **Maude spezifiziert alle möglichen Zustände der Uhr:**

```
fmod CLOCK is
  protecting NAT .
  sort State .

  op init : -> State [ctor] .
  op tick : State  -> State [ctor] .
  op hr   : State -> Nat .
  op min : State -> Nat .

  var s : State .
  eq min(init) = 0 .
  eq min(tick(s)) = if min(s) < 59 then min(s) + 1 else 0 fi  .
  eq hr(init) = 0 .
  eq hr(tick(s)) =
       if min(s) < 59
                  then hr(s)
                  else if hr(s) < 23 then hr(s) + 1  else 0 fi
       fi .
endfm
```

2. Testen der Uhr in Maude (algebraische Spez.)

- **Hilfsfunktion zum Generieren des Zustands**
  `genTick(S, M)` generiert den Zustand nach `M` „ticks“:

```
op genTick : State Nat -> State .
var S : State . var M : Nat .
eq genTick(S, 0) = S .
eq genTick(S, s M) = tick(genTick(S, M)) .
```

- **Testbeispiele**
  Beobachtung des Zustands nach 112 „ticks“)

```
reduce in CLOCK : min(genTick(init, 112)) .
rewrites: 561 in 9287ms cpu (4ms real) (60
    rewrites/second)
result NzNat: 52
==========================================
reduce in CLOCK : hr(genTick(init, 112)) .
rewrites: 25377 in 3148026001ms cpu (244ms real) (0
    rewrites/second)
result NzNat: 1
```


2. Uhr mit Sekundenangabe in Maude (algebraische Spez.)

```
fmod CLOCK2 is
  protecting NAT .
  sort State2 .

  op init2 : -> State2 [ctor] .
  op tick2 : State2  -> State2 [ctor] .
  op hr  : State2 -> Nat .
  op min : State2 -> Nat .
  op sec : State2 -> Nat .

  var s : State2 .
  eq sec(init2) = 0 .
  eq sec(tick2(s)) = if sec(s) < 59 then sec(s) + 1 else 0 fi  .

  eq min(init2) = 0 .
  eq min(tick2(s)) =
    if sec(s) < 59 then min(s)
            else  if min(s) < 59 then min(s) + 1 else 0 fi
    fi .
. . .
```

2. Uhr mit Sekundenangabe in Maude (algebraische Spez.)

```
. . .

  eq hr(init2) = 0 .
  eq hr(tick2(s)) =
       if (sec(s) < 59 or min(s) < 59)
           then hr(s)
             else if hr(s) < 23 then hr(s) + 1 else 0 fi
       fi .
endfm
```


3. Uhr in Maude (Termersetzungssystem)

- **Das Ersetzungssystem spezifiziert alle möglichen Zustandsübergange der Uhr:**

```
mod CLOCK-MACHINE is
     protecting NAT .
     including CONFIGURATION .
     op CLOCK : -> Cid .
     op min :_ : Nat  -> Attribute .
     op hr :_ : Nat  -> Attribute .
     op tick : -> Msg .

     var C : Oid .  vars M M1 H H1 : Nat .

     crl [tick] :
     tick < C : CLOCK | hr : H  , min : M  >
     => < C : CLOCK | hr : H1 , min : M1 >
     if M1 := (if M < 59 then M + 1 else 0 fi ) /\
        H1 := (if M < 59 then H else
                  if H < 23 then H + 1 else 0 fi
               fi) .
endm
```

3. Testen der Uhr in Maude (Termersetzungssystem)

- **Hilfsfunktion zum Generieren von „tick“**

```
op genTick : Nat -> Configuration .
eq genTick(0) = none .
eq genTick(s M) = tick genTick(M) .
```

- **Testbeispiel**

```
rewrite in CLOCK-MACHINE :
  genTick(137) < C : CLOCK | min : 10, hr : 7 > .
rewrites: 827 in 9999749470ms cpu (23ms real) (0
  rewrites/second)
result Object: < C : CLOCK | min : 27, hr : 9 >
```


3. Verfeinerung

- **MTLA: Verfeinerung als Implikation**
  - CLOCK2 => CLOCK
- **Maude: Wechsel der Datenstruktur**
  - **CLOCK2 als Verfeinerung von CLOCK:**
    - Erweitere CLOCK2 um passendes “Minuten-Tick” zu CLOCK-BY-CLOCK2
    - Konstruiere Umbenennung σ: sig(CLOCK) –> sig(CLOCK-BY-CLOCK2)
    - Dann gilt: CLOCK-BY-CLOCK2 simuliert bzgl. σ , d.h. CLOCK-BY-CLOCK2 $\sim\sim>_{\sigma}$ CLOCK

3. Verfeinerung

- **Erweiterung von CLOCK2**

```
fmod CLOCK-BY-CLOCK2 is
      protecting NAT .
      protecting CLOCK2 .
      op tick : State2  -> State2 .
      var s : State2 .
      eq tick(s) = tick2(tick2( … tick2(s) …)) .
  endfm
```

60-mal

- **Umbenennung**

  σ: sig(CLOCK) –> sig(CLOCK-BY-CLOCK2)

  σ(State) = State2, σ(init) = init2

- **Simulationsbeweis**

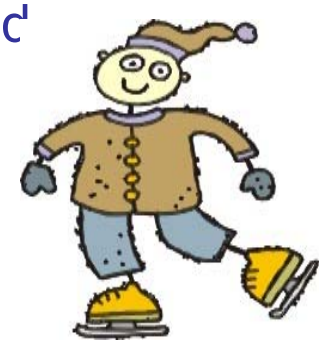
  Ax(CLOCK-BY-CLOCK2) => Ax(CLOCK[State2/State, init2/init])

Zusammenfassung: TLA vs. Maude

- **TLA –Temporal Logic of Actions :**
  - Spezifikation der Menge aller möglichen (unendlichen) Abläufe der Abläufe der Uhr
  - Verfeinerung als Implikation
- **Maude – Algebraische Spezifikation**
  - Spezifikation der möglichen Zustände der Uhr
  - Ablesen der Zeit durch Beobachtungsfunktionen (min, hr)
  - Verfeinerung als Wechsel der Datenstruktur
- **Maude – Dynamisches Verhalten**
  - Spezifikation der Zustandsübergange der Uhr
  - Testen endlicher Abläufe

Vielen Dank für Ihre Aufmerksamkeit und Mitarbeit!

Axel Rauschmayer und ich wünschen Ihnen **viel Erfolg in der Klausur** und eine **angenehme vorlesungsfreie Zeit!**



Auf Wiedersehen!