CS6480:
Real-Time and Composition

Robbert van Renesse
Cornell University

Based on Chapters 9 and 10 of “Specifying Systems” by Leslie Lamport

Recall: HourClock

MODULE $HourClock$

EXTENDS $Naturals$

VARIABLE hr

$$HCini \triangleq hr \in (1 \,..\, 12)$$

$$HCnxt \triangleq hr' = \text{IF } hr = 12 \text{ THEN } 1 \text{ ELSE } hr + 1$$

$$HC \triangleq HCini \land \Box[HCnxt]_{hr} \land \mathrm{WF}_{hr}(HCnxt)$$

Recall: HourClock

$$\text{MODULE } \mathit{HourClock}$$

$$\text{EXTENDS } \mathit{Naturals}$$
$$\text{VARIABLE } hr$$
$$\mathit{HCini} \triangleq hr \in (1 \,..\, 12)$$
$$\mathit{HCnxt} \triangleq hr' = \text{IF } hr = 12 \text{ THEN } 1 \text{ ELSE } hr + 1$$
$$\mathit{HC} \triangleq \mathit{HCini} \land \Box[\mathit{HCnxt}]_{hr} \land \text{WF}_{hr}(\mathit{HCnxt})$$

Can we create an HourClock that ticks (approximately) once an hour?

Specifying Real-Time

MODULE *RealTime*

EXTENDS *Reals*

VARIABLE *now*

$$RTini \triangleq now \in Real$$

$$RTnxt \triangleq now' \in \{r \in Real : r > now\}$$

$$RT \triangleq \land RTini$$
$$\land \Box[RTnxt]_{now}$$
$$\land \forall r \in Real : \mathrm{WF}_{now}(RTnxt \land (now' > r))$$

Note: takes discrete steps

Specifying Real-Time

MODULE *RealTime*

EXTENDS *Reals*

VARIABLE *now*

$$RTini \triangleq now \in Real$$

$$RTnxt \triangleq now' \in \{r \in Real : r > now\}$$

$$\begin{aligned} RT \triangleq\ & \land\ RTini \\ & \land\ \Box[RTnxt]_{now} \\ & \land\ \forall\, r \in Real : \mathrm{WF}_{now}(RTnxt \land (now' > r)) \end{aligned}$$

Why this?

Composing HourClock and RealTime

Can we create a spec that extends HourClock to "tick" at (approximately) regular intervals, like a physical clock?

Allowed steps in composition:

$$\begin{bmatrix} hr & = & 12 \\ now & = & \sqrt{2.47} \end{bmatrix} \rightarrow \begin{bmatrix} hr & = & 1 \\ now & = & \sqrt{2.47} \end{bmatrix}$$

Clock ticks are instantaneous

$$\begin{bmatrix} hr & = & 11 \\ now & = & 23.4 \end{bmatrix} \rightarrow \begin{bmatrix} hr & = & 11 \\ now & = & 23.5 \end{bmatrix}$$

Time progresses between ticks

Real-time HourClock

- Want time between HCnxt steps to be approximately one hour on the real-time clock
  - *Real clocks drift!!*
- If t is the time in seconds between two steps, then we want
  - $3600 - \rho \leq t \leq 3600 + \rho$
  - We call ρ the “drift” of a clock (not to be confused with “skew” δ)

Bounding time between HCnxt steps

CONSTANT Rho A positive real number.

MODULE $Inner$

VARIABLE t — t is the elapsed time since the last $HCnxt$ step.

$$TNext \triangleq t' = \text{IF } HCnxt \text{ THEN } 0 \text{ ELSE } t + (now' - now)$$

$$Timer \triangleq (t = 0) \land \Box[TNext]_{\langle t, hr, now \rangle}$$

$$MaxTime \triangleq \Box(t \leq 3600 + Rho)$$

t is always at most $3600 + Rho$.

$$MinTime \triangleq \Box[HCnxt \Rightarrow t \geq 3600 - Rho]_{hr}$$

An $HCnxt$ step can occur only if $t \geq 3600 - Rho$.

$$HCTime \triangleq Timer \land MaxTime \land MinTime$$

Bounding time between HCnxt steps

We're going to want to hide *t*

CONSTANT Rho — A positive real number.

MODULE *Inner*

VARIABLE t — t is the elapsed time since the last $HCnxt$ step.

$$TNext \triangleq t' = \text{IF } HCnxt \text{ THEN } 0 \text{ ELSE } t + (now' - now)$$

$$Timer \triangleq (t = 0) \land \Box[TNext]_{\langle t, hr, now \rangle}$$

$$MaxTime \triangleq \Box(t \leq 3600 + Rho)$$

t is always at most $3600 + Rho$.

$$MinTime \triangleq \Box[HCnxt \Rightarrow t \geq 3600 - Rho]_{hr}$$

An $HCnxt$ step can occur only if $t \geq 3600 - Rho$.

$$HCTime \triangleq Timer \land MaxTime \land MinTime$$

Real-Time HourClock

MODULE *RealTimeHourClock*

EXTENDS *Reals*, *HourClock*

VARIABLE *now* — The current time, measured in seconds.

CONSTANT *Rho* — A positive real number.

ASSUME $(Rho \in Real) \land (Rho > 0)$

$I(t) \triangleq$ INSTANCE *Inner*

$NowNext \triangleq \land\ now' \in \{r \in Real : r > now\}$
$\qquad\qquad\quad \land$ UNCHANGED hr

A *NowNext* step can advance *now* by any amount while leaving *hr* unchanged.

$RTnow \triangleq \land\ now \in Real$
$\qquad\qquad \land\ \Box[NowNext]_{now}$
$\qquad\qquad \land\ \forall r \in Real : \mathrm{WF}_{now}(NowNext \land (now' > r))$

RTnow specifies how time may change.

$RTHC \triangleq HC \land RTnow \land (\exists\, t : I(t)!HCTime)$

The complete specification.

Real-Time HourClock

MODULE *RealTimeHourClock*

EXTENDS *Reals*, *HourClock*

VARIABLE *now* — The current time, measured in seconds.

CONSTANT *Rho* — A positive real number.

ASSUME $(Rho \in Real) \land (Rho > 0)$

$I(t) \triangleq$ INSTANCE *Inner*

$$NowNext \triangleq \land\ now' \in \{r \in Real : r > now\} \quad \land\ \text{UNCHANGED } hr$$

A *NowNext* step can advance *now* by any amount while leaving *hr* unchanged.

$$RTnow \triangleq \land\ now \in Real \quad \land\ \Box[NowNext]_{now} \quad \land\ \forall r \in Real : \text{WF}_{now}(NowNext \land (now' > r))$$

RTnow specifies how time may change.

$$RTHC \triangleq HC \land RTnow \land (\exists t : I(t)!HCTime)$$

The complete specification.

Why do we need this?

Composition of Specifications

- Given two or more specifications, looking for set of behaviors that satisfy all specifications

➔Composition is the conjunction of specifications

Let’s compose two instantiations of HourClock and see what happens…

Rewriting HourClock a bit

MODULE *HourClock*

EXTENDS *Naturals*

VARIABLE *hr*

$$HCN(h) \triangleq h' = (h \% 12) + 1$$

$$HCini \triangleq hr \in (1 \,..\, 12)$$

$$HCnxt \triangleq HCN(hr)$$

$$HC \triangleq HCini \land \Box[HCnxt]_{hr}$$

TwoClocks Spec

$$TwoClocks \triangleq \begin{array}{l} \land (x \in 1 \,..\, 12) \land \Box[HCN(x)]_x \\ \land (y \in 1 \,..\, 12) \land \Box[HCN(y)]_y \end{array}$$

TwoClocks Spec

$$TwoClocks \triangleq \begin{array}{l} \land\ (x \in 1 \,..\, 12) \land \Box[HCN(x)]_x \\ \land\ (y \in 1 \,..\, 12) \land \Box[HCN(y)]_y \end{array}$$

Not in the "standard" form $Init \land \Box[Next]_{vars}$

TwoClocks Spec

$$TwoClocks \triangleq \begin{array}{l} \land \ (x \in 1 \,..\, 12) \land \Box[HCN(x)]_x \\ \land \ (y \in 1 \,..\, 12) \land \Box[HCN(y)]_y \end{array}$$

$$\equiv \begin{array}{l} \land \ (x \in 1 \,..\, 12) \land (y \in 1 \,..\, 12) \\ \land \ \Box \left([HCN(x)]_x \land [HCN(y)]_y \right) \end{array}$$

Because $\Box(F \land G) \equiv (\Box F) \land (\Box G)$.

$$\equiv \begin{array}{l} \land \ (x \in 1 \,..\, 12) \land (y \in 1 \,..\, 12) \\ \land \ \Box \left(\begin{array}{l} \land \ HCN(x) \lor x' = x \\ \land \ HCN(y) \lor y' = y \end{array} \right) \end{array}$$

By definition of $[\ldots]_x$ and $[\ldots]_y$.

Cont'd

$$
\begin{aligned}
\equiv\ & \land\ (x \in 1 \,..\, 12) \land (y \in 1 \,..\, 12) \\
& \land\ \Box(\ \land\ HCN(x) \lor x' = x \\
& \qquad\ \ \land\ HCN(y) \lor y' = y\ )
\end{aligned}
$$

$$
\begin{aligned}
\equiv\ & \land\ (x \in 1 \,..\, 12) \land (y \in 1 \,..\, 12) \\
& \land\ \Box(\lor\ HCN(x) \land HCN(y) \\
& \qquad \lor\ HCN(x) \land (y' = y) \\
& \qquad \lor\ HCN(y) \land (x' = x) \\
& \qquad \lor\ (x' = x) \land (y' = y)\ )
\end{aligned}
$$

Because:

$$
\begin{pmatrix} \land \lor A_1 \\ \quad \lor A_2 \\ \land \lor B_1 \\ \quad \lor B_2 \end{pmatrix} \equiv \begin{pmatrix} \lor A_1 \land B_1 \\ \lor A_1 \land B_2 \\ \lor A_2 \land B_1 \\ \lor A_2 \land B_2 \end{pmatrix}
$$

TwoClocks Spec

$$TwoClocks \triangleq \land (x \in 1 \,..\, 12) \land \Box[HCN(x)]_x$$
$$\land (y \in 1 \,..\, 12) \land \Box[HCN(y)]_y$$

$$\equiv \land (x \in 1 \,..\, 12) \land (y \in 1 \,..\, 12)$$
$$\land \Box[\lor HCN(x) \land HCN(y)$$
$$\lor HCN(x) \land (y' = y)$$
$$\lor HCN(y) \land (x' = x)\ ]_{\langle x, y \rangle}$$

By definition of $[\ldots]_{\langle x, y \rangle}$.

"standard" form $Init \land \Box[TCNxt]_{vars}$

TwoClocks Spec

$$TwoClocks \triangleq \land (x \in 1 \,..\, 12) \land \Box[HCN(x)]_x$$
$$\land (y \in 1 \,..\, 12) \land \Box[HCN(y)]_y$$

$$\equiv \land (x \in 1 \,..\, 12) \land (y \in 1 \,..\, 12)$$
$$\land \Box [\lor HCN(x) \land HCN(y)$$
$$\lor HCN(x) \land (y' = y)$$
$$\lor HCN(y) \land (x' = x)]_{\langle x, y \rangle}$$

By definition of $[\ldots]_{\langle x, y \rangle}$.

Clocks can progress simultaneously!

TwoClocks Spec

$$TwoClocks \triangleq \begin{array}{l} \land\ (x \in 1 \,..\, 12) \land \Box[HCN(x)]_x \\ \land\ (y \in 1 \,..\, 12) \land \Box[HCN(y)]_y \end{array}$$

$$\equiv \begin{array}{l} \land\ (x \in 1 \,..\, 12) \land (y \in 1 \,..\, 12) \\ \land\ \Box\left[\begin{array}{l} \lor\ HCN(x) \land HCN(y) \\ \lor\ HCN(x) \land (y' = y) \\ \lor\ HCN(y) \land (x' = x) \end{array}\right]_{\langle x,\, y \rangle} \end{array}$$

By definition of $[\ldots]_{\langle x,\, y \rangle}$.

Clocks can progress simultaneously!

If we don't want this, can write: $TwoClocks \land \Box[(x' = x) \lor (y' = y)]_{\langle x,\, y \rangle}$

Performance properties

1. Step must complete within δ time: *safety* property
   - "hard real-time"
2. Step must complete within δ time on average: *hyperproperty*
   - Implied by 1
3. Step must eventually occur: *liveness* property
   - Implied by 1 or 2

TLA+ only allows specifying *properties*

- A *property* is a set of behaviors (infinite traces) each satisfying some predicate
- "response time < δ" is a predicate over a single behavior
- "average response time < δ" is a predicate over a set of behaviors

Tools for checking hyperproperties

- Some hyperproperties just involve small sets of behaviors
- 2-Safety: two behaviors provide a counterexample
- Security example: “Observational Determinism”
  - Behavior of public variables is deterministic
  - Independent of behavior of private variables or scheduler
  - *Bad*: pair of traces that cause system to look nondeterministic to low observer
- Can be handled in TLA+ using “self-composition”
  - Like TwoClocks
  - Can be model-checked, TLAPS, …
- Still can’t handle average response time…
  - *Good*: average time over all behaviors is low enough
- Alternative tools: HyperLTL, HyperCTL, Hyper modal μ-calculus