Distributed Systems

ALI KAMANDI, PH.D.

SCHOOL OF ENGINEERING SCIENCE

COLLEGE OF ENGINEERING

UNIVERSITY OF TEHRAN

KAMANDI@UT.AC.IR

توصیف رسمی

نمایش ریاضی سیستم

- جبر

$+, -, *, /$

- منطق

$\wedge \quad \vee \quad \neg \quad \Rightarrow \quad \equiv$

- مجموعه ها

$\cap$ intersection $\cup$ union $\subseteq$ subset $\setminus$ set difference

Propositional Logic

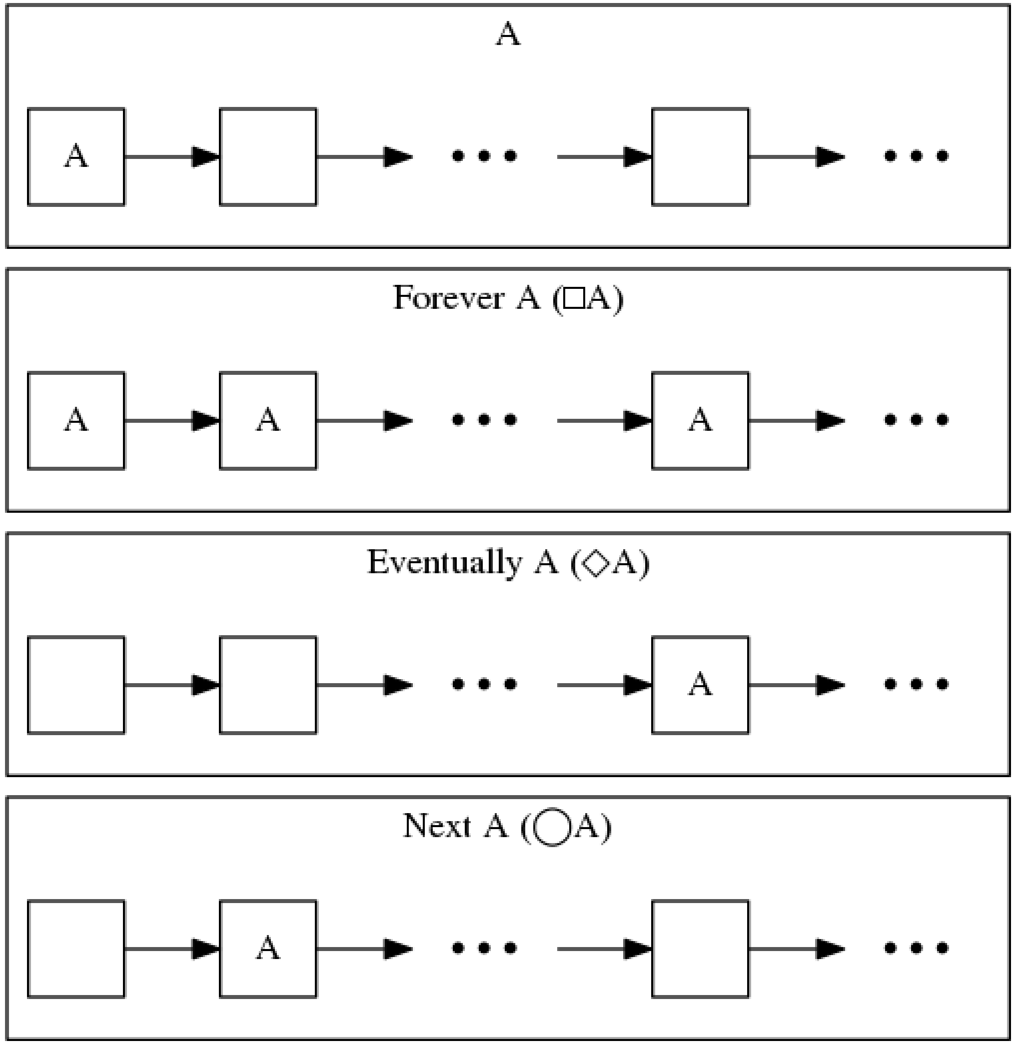
$\wedge$	conjunction (and)	$\Rightarrow$	implication (implies)
$\vee$	disjunction (or)	$\equiv$	equivalence (is equivalent to)
$\neg$	negation (not)		

Predicate Logic

Predicate logic extends propositional logic with the two quantifiers:

$\forall$ universal quantification (for all)
$\exists$ existential quantification (there exists)

Temporal Logic



Specifying a simple clock

An hour clock (HC)

A typical behavior of the clock is the sequence of states:

$$[hr = 11] \;\rightarrow\; [hr = 12] \;\rightarrow\; [hr = 1] \;\rightarrow\; [hr = 2] \;\rightarrow\; \cdots$$

$[hr = 11]$ is a state in which the variable *hr* has the value 11

A pair of successive states, such as $[hr = 1] \rightarrow [hr = 2]$, is called a step.

To specify the hour clock, we describe **all its possible behaviors**.

We write an **initial predicate** that species the possible initial values of *hr* , and a **next-state** relation that species how the value of *hr* can change in any step.

$$HCini \triangleq hr \in \{1, \ldots, 12\}$$

… is informal.

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

The temporal formula $\Box F$ asserts that formula F is always true.

In particular, $\Box HCnxt$ is the assertion that $HCnxt$ is true for every step in the behavior.

Weather station:

$$\begin{bmatrix} hr & = & 11 \\ tmp & = & 23.5 \end{bmatrix} \rightarrow \begin{bmatrix} hr & = & 12 \\ tmp & = & 23.5 \end{bmatrix} \rightarrow \begin{bmatrix} hr & = & 12 \\ tmp & = & 23.4 \end{bmatrix} \rightarrow$$

$$\begin{bmatrix} hr & = & 12 \\ tmp & = & 23.3 \end{bmatrix} \rightarrow \begin{bmatrix} hr & = & 1 \\ tmp & = & 23.3 \end{bmatrix} \rightarrow \cdots$$

$$HCini \wedge \Box HCnxt$$

$$HCini \wedge \Box(HCnxt \vee (hr' = hr))$$

$$HCini \wedge \Box[HCnxt]_{hr}$$

TLA+

- Reserved words that appear in small upper-case letters (like EXTENDS) are written in ASCII with ordinary upper-case letters.
- When possible, symbols are represented pictorially in ASCII—for example, $\Box$ is typed as `[]` and $\neq$ as `#`. (You can also type $\neq$ as `/=`.)
- When there is no good ASCII representation, TEX notation is used—for example, $\in$ is typed as `\in`. The major exception is $\triangleq$, which is typed as `==`.

MODULE *HourClock*

EXTENDS *Naturals*

VARIABLE hr

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

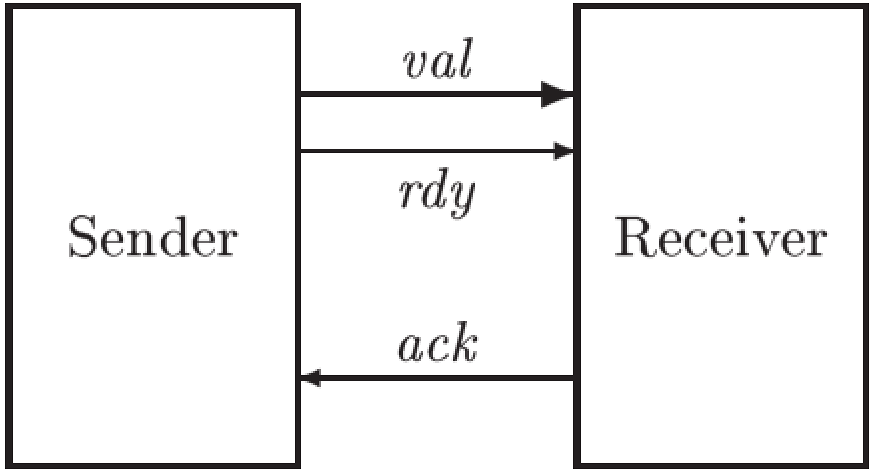
$$HCnxt \triangleq hr' = \text{IF } hr \neq 12 \text{ THEN } hr + 1 \text{ ELSE } 1$$

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

THEOREM $HC \Rightarrow \Box HCini$

```
----------------------- MODULE HourClock ----------------------
EXTENDS Naturals
VARIABLE hr
HCini  ==  hr \in (1 .. 12)
HCnxt  ==  hr' = IF hr # 12 THEN hr + 1 ELSE 1
HC  ==  HCini /\ [][HCnxt]_hr
--------------------------------------------------------------
THEOREM  HC => []HCini
==============================================================
```

An Asynchronous Interface



$$\begin{bmatrix} val & = & 26 \\ rdy & = & 0 \\ ack & = & 0 \end{bmatrix} \xrightarrow{Send\ 37} \begin{bmatrix} val & = & 37 \\ rdy & = & 1 \\ ack & = & 0 \end{bmatrix} \xrightarrow{Ack} \begin{bmatrix} val & = & 37 \\ rdy & = & 1 \\ ack & = & 1 \end{bmatrix} \xrightarrow{Send\ 4}$$

$$\begin{bmatrix} val & = & 4 \\ rdy & = & 0 \\ ack & = & 1 \end{bmatrix} \xrightarrow{Ack} \begin{bmatrix} val & = & 4 \\ rdy & = & 0 \\ ack & = & 0 \end{bmatrix} \xrightarrow{Send\ 19} \begin{bmatrix} val & = & 19 \\ rdy & = & 1 \\ ack & = & 0 \end{bmatrix} \xrightarrow{Ack} \cdots$$

MODULE *AsynchInterface*

EXTENDS *Naturals*
CONSTANT *Data*
VARIABLES *val*, *rdy*, *ack*

$$TypeInvariant \triangleq \land\ val \in Data$$
$$\land\ rdy \in \{0, 1\}$$
$$\land\ ack \in \{0, 1\}$$

$$Init \triangleq \land\ val \in Data$$
$$\land\ rdy \in \{0, 1\}$$
$$\land\ ack = rdy$$

$$Send \triangleq \land\ rdy = ack$$
$$\land\ val' \in Data$$
$$\land\ rdy' = 1 - rdy$$
$$\land\ \text{UNCHANGED}\ ack$$

$$Rcv \triangleq \land\ rdy \neq ack$$
$$\land\ ack' = 1 - ack$$
$$\land\ \text{UNCHANGED}\ \langle val, rdy \rangle$$

$$Next \triangleq Send \lor Rcv$$
$$Spec \triangleq Init \land \Box[Next]_{\langle val, rdy, ack \rangle}$$

THEOREM $Spec \Rightarrow \Box TypeInvariant$

$$\begin{bmatrix} big & = 0 \\ small & = 0 \end{bmatrix}$$

The big jug is filled from the faucet. $\downarrow$

$$\begin{bmatrix} big & = 5 \\ small & = 0 \end{bmatrix}$$

The small jug is filled from the big one. $\downarrow$

$$\begin{bmatrix} big & = 2 \\ small & = 3 \end{bmatrix}$$

The small jug is emptied (onto the ground). $\downarrow$

$$\begin{bmatrix} big & = 2 \\ small & = 0 \end{bmatrix}$$

A little thought reveals that there are three kinds of steps in a behavior:

- Filling a jug.
- Emptying a jug.
- Pouring from one jug to the other. There are two cases:
  - This empties the first jug.
  - This fills the second jug, possibly leaving water in the first jug.

EXTENDS *Integers*

VARIABLES *big*, *small*

$$Init \triangleq \land big = 0 \\ \qquad\quad \land small = 0$$

$$\begin{aligned} Next \triangleq\ & \lor FillSmall \\ & \lor FillBig \\ & \lor EmptySmall \\ & \lor EmptyBig \\ & \lor SmallToBig \\ & \lor BigToSmall \end{aligned}$$

$$FillSmall \triangleq small' = 3$$

$$\begin{bmatrix} big & = 2 \\ small & = 1 \end{bmatrix} \rightarrow \begin{bmatrix} big & = 2 \\ small & = 3 \end{bmatrix}$$

$$\begin{bmatrix} big & = 2 \\ small & = 1 \end{bmatrix} \rightarrow \begin{bmatrix} big & = \sqrt{42} \\ small & = 3 \end{bmatrix}$$

$$FillSmall \triangleq \begin{array}{l} \land\ small' = 3 \\ \land\ big' = big \end{array}$$

FillBig

$$FillBig \triangleq \land\, big' = 5$$
$$\land\, small' = small$$

$$EmptySmall \triangleq \land\, small' = 0$$
$$\land\, big' = big$$

$$EmptyBig \triangleq \land\, big' = 0$$
$$\land\, small' = small$$

SmallToBig

$$
\begin{aligned}
SmallToBig \triangleq\ & \lor \land big + small > 5 \\
& \quad \land big' = 5 \\
& \quad \land small' = small - (5 - big) \\
& \lor \land big + small \leq 5 \\
& \quad \land big' = big + small \\
& \quad \land small' = 0
\end{aligned}
$$

$$Min(m, n) \triangleq \text{IF } m < n \text{ THEN } m \text{ ELSE } n$$

$$\begin{aligned} SmallToBig \triangleq\ & \land big' = Min(big + small, 5) \\ & \land small' = small - (Min(big + small, 5) - big) \end{aligned}$$

$$\begin{aligned} & SmallToBig \triangleq \\ & \quad \text{LET } poured \triangleq Min(big + small, 5) - big \\ & \quad \text{IN } \quad \land big' \quad = big + poured \\ & \quad \qquad\ \ \land small' = small - poured \end{aligned}$$

BigToSmall

$$
\begin{array}{l}
BigToSmall \triangleq \\
\quad \text{LET } poured \triangleq Min(big + small, 3) - small \\
\quad \text{IN} \quad \land\ big' \quad = big - poured \\
\quad \qquad\ \ \land\ small' = small + poured
\end{array}
$$

Invariant:

$$\mathit{TypeOK} \triangleq \begin{array}{l} \land\ \mathit{big} \quad\ \ \in 0\,..\,5 \\ \land\ \mathit{small} \in 0\,..\,3 \end{array}$$

منابع

Leslie Lamport, Specifying Systems, The TLA+ Language and Tools for Hardware and Software Engineers, Addison-Wesley, 2002.