Especificación de sistemas concurrentes con TLA$^+$

Temporal Logic of Actions

- Lenguaje no tipado
- Fuertemente orientado a concurrencia
- Teoría de conjuntos
- Lógica de temporal
- Máquinas de estados
  - El estado se define por medio de variables
  - Se definen operaciones (acciones) que modifican ese estado y por lo tanto definen la relación de transición.

Concurrencia, TL y TLA

- La lógica temporal, en principio, permite describir un sistema concurrente con una única fórmula (A. Pneuli, 1977).
- Según Lamport, en la práctica, no resulta muy conveniente; por este motivo él introduce TLA.
- La mayor parte de TLA no son fórmulas temporales, las cuales sólo aparecen en el momento necesario.

TLA y TLA$^+$

- TLA por sí sola es matemática y lógica temporal, es el fundamento teórico del lenguaje.
- TLA es algo así como Z sin los esquemas: lógica.
- TLA$^+$ provee el *azúcar sintáctico* necesario para escribir especificaciones largas.
  - Modularización
  - Parametrización
  - Instanciación

¿Qué es un estado?

- Sea
  - *Val*, la colección de todos los valores posibles (de cualquier tipo), excepto TRUE y FALSE.
  - *Var*, el conjunto de nombres de variables.
- Un estado es una función de *Var* en *Val*.
- Entonces si *s* es un estado y *x* es una variable (cualquiera, es decir se haya usado o no para definir *s*), *s(x)* es el valor de *x* en *s*.
- En TLA, un estado puede considerarse un estado del Universo.

Designaciones para un *timer*

- El timer debe detenerse o emitir una señal *limit* unidades de tiempo luego de haber sido iniciado ≈ *Set(limit)*
- Se enciende el timer ≈ *Start*
- El timer alcanza el límite de tiempo prefijado ≈ *Timeout*
- El tiempo real avanza una unidad de tiempo ≈ *Tick*
- El timer se detiene ≈ *Stop*

Modelando el tiempo real

MODULE *DK_RealTime*

EXTENDS *Naturals*

VARIABLES *now*

$TypeInv \triangleq now \in Nat$

$InitRealTime \triangleq \quad now = 0$

$Tick \triangleq \quad now' = now + 1$

$RealTimeSpec \triangleq InitRealTime \wedge \Box[Tick]_{now} \wedge WF_{now}(Tick)$

El *timer*

MODULE *Timer*

EXTENDS *Naturals* , *DK _ RealTime*

VARIABLES *running* , *realtime* , *limit*

$sv \triangleq \langle realtime , running , limit \rangle$

$TypeInv \triangleq realtime , limit \in Nat \wedge running \in \{\overline{\text{yes}} , \overline{\text{no}}\}$

$$
\begin{aligned}
InitTimer \triangleq \quad & \wedge realtime = 0 \\
& \wedge running = \overline{\text{no}}
\end{aligned}
$$

$$
\begin{aligned}
Set(l) \triangleq \quad & \wedge running = \overline{\text{no}} \\
& \wedge l > 0 \\
& \wedge limit' = l \\
& \wedge \text{UNCHANGED} \langle now , realtime , running \rangle
\end{aligned}
$$

El *timer* (2)

$$
\begin{aligned}
Start \triangleq \quad & \wedge running = \overline{\text{no}} \\
& \wedge limit > 0 \\
& \wedge realtime' = now \\
& \wedge running' = \overline{\text{yes}} \\
& \wedge \text{UNCHANGED} \langle now, limit \rangle \\
Stop \triangleq \quad & \wedge running = \overline{\text{yes}} \\
& \wedge running' = \overline{\text{no}} \\
& \wedge \text{UNCHANGED} \ \langle realtime, limit, now \rangle \\
Timeout \triangleq \quad & \wedge running = \overline{\text{yes}} \\
& \wedge now - realtime \geq limit \\
& \wedge running' = \overline{\text{no}} \\
& \wedge \text{UNCHANGED} \ \langle realtime, limit, now \rangle
\end{aligned}
$$

now no va a cambiar mientras se "ejecute" una operación; en otras palabras las operaciones no consumen tiempo.

El *timer* (3)

$$TimerNext \triangleq Start \lor Stop \lor Timeout \lor (\exists\, l \in \mathbb{N} \bullet Set(l))$$

$$TimerSpec \triangleq InitTimer \land \Box[TimerNext]_{sv} \land WF_{sv,\, now}(Timeout)$$

THEOREM $\quad Spec \Rightarrow \Box TypeInv$

Módulos

- La unidad de especificación de TLA$^+$ son los módulos; llevan un nombre para futuras referencias.
- Un módulo puede incluir:
  - variables: VARIABLE, VARIABLES
  - constantes: CONSTANT, CONSTANTS
  - hipótesis sobre las constantes: ASSUME
  - definiciones: $\triangleq$
  - teoremas: THEOREM
  - otros módulos: EXTENDS, INSTANCE

Un timer a límite constante

EXTENDS e INSTANCE

- EXTENDS permite utilizar todas las definiciones efectuadas en los módulos que se extienden.
  - Excepto BOOLEAN, todos los demás "tipos" deben ser incluidos usando esta cláusula.
- *T1* $\triangleq$ INSTANCE *Timer* WITH *realtime* $\leftarrow$ *t1*, *limit* $\leftarrow$ *Tp*, *running* $\leftarrow$ *r1*
  - Define una instancia de *Timer* donde *time*, *Limit* y *running* son renombradas; *t1*, *Tp* y *r1* deben estar definidas.
  - De esta forma, *T1!Timeout* es

$$r1 = \bar{\text{yes}} \wedge now - t1 \geq Tp \wedge r1' = \bar{\text{no}} \wedge \text{UNCHANGED } \langle t1, Tp, now \rangle$$

VARIABLE

- Esta cláusula se utiliza para definir las variables de estado que interesan al módulo.
- Las variables no tienen tipo. Esto significa que si definimos la variable *n*, luego podemos escribir cosas como $n = Tail(s)$ o $n = 5$.
- Sin embargo, es común que se incluya un predicado sobre los valores posibles que pueden tomar las variables y luego se "obligue" al especificador a probar que ese predicado es un invariante.

Definiciones

- Las definiciones permiten estructurar la lógica que se utiliza dentro de un módulo.
- El nombre de una definición se puede utilizar en otras definiciones; y como vimos, las definiciones de un módulo pueden accederse desde otros.
- Las definiciones pueden ser paramétricas:

$$
\begin{aligned}
Set(l) \triangleq \quad & \wedge running = \overline{\text{no}} \\
& \wedge l > 0 \\
& \wedge limit' = l \\
& \wedge \text{UNCHANGED} \langle now, realtime, running \rangle
\end{aligned}
$$

Acciones

- Las acciones son las operaciones del sistema.
- Formalmente, una acción es un predicado que depende de dos estados; por tanto, contiene variables primadas y no-primadas.
- Si A es una acción y s y t son dos estados tales que $s \llbracket A \rrbracket t$ entonces el par (s,t) es un paso-A.
  - $s \llbracket A \rrbracket t \mathrel{\widehat{=}} A(\forall v: s(v) / v,\ t(v) / v')$
- Si $\langle s_0, s_1, \ldots \rangle$ es una ejecución, se define el valor de verdad de la acción A como:
  - $\langle s_0, s_1, \ldots \rangle \llbracket A \rrbracket \mathrel{\widehat{=}} s_0 \llbracket A \rrbracket s_1$

Acciones (2)

- Las acciones son atómicas. Es decir, no existen estados intermedios entre el estado de partida y el de llegada de una acción.
- Sean *A* y *B* acciones que se inician en *s* y finalizan en *t* y *v*, respectivamente. Entonces, el sistema pasa de *s* a *t* o de *s* a *v*, pero no ambos ni a ningún otro.
- Esto significa que si se espera que las ejecuciones del sistema verifiquen cierta propiedad, habrá que considerar ambas.

Ejemplo

MODULE S

EXTENDS *Naturals*

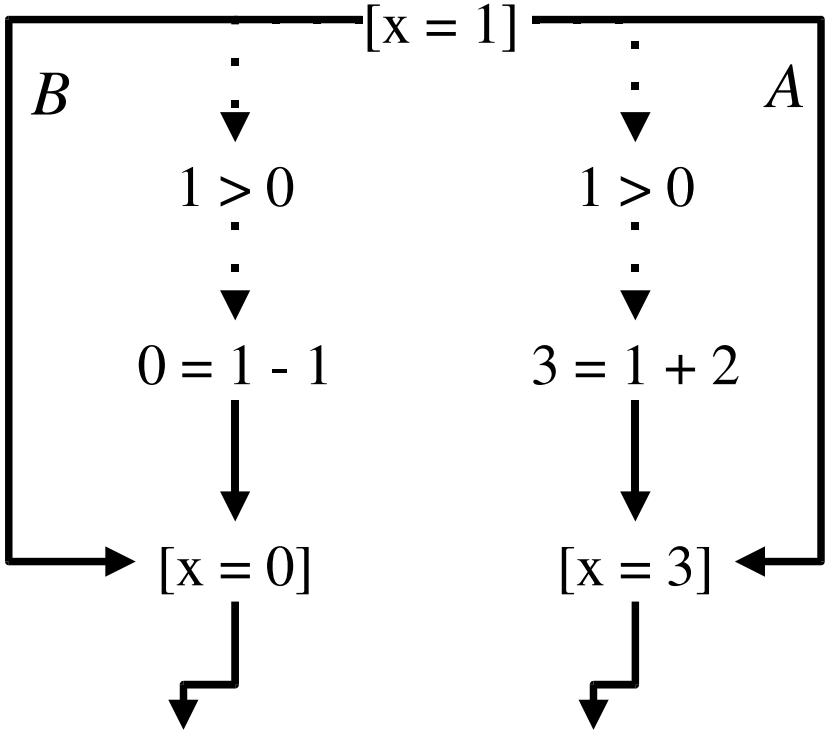
VARIABLE x

$$Init \triangleq x = 1$$

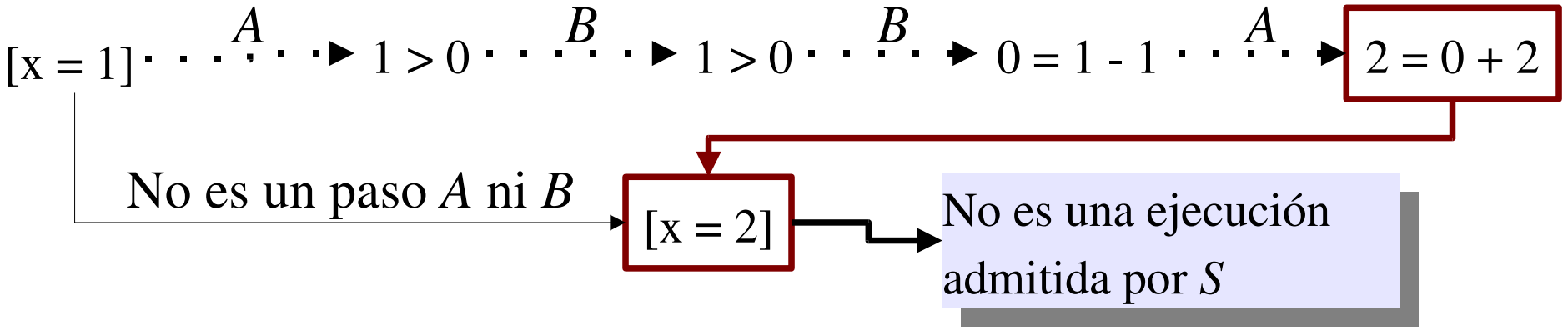
$$A \triangleq \land x > 0 \land x' = x + 2$$

$$B \triangleq \land x > 0 \land x' = x - 1$$

$$Spec \triangleq Init \land \Box[A \lor B]_x$$



S admite las dos ejecuciones



Interleaving model of concurrency

- La semántica de TLA está orientada a abordar el problema de la concurrencia por medio del IMC.
- Dado un estado *s* y *n* acciones que pueden iniciarse en *s*, se deben considerar las *n* ejecuciones que se originan partiendo de *s*.
- Si se espera que una especificación verifique cierta propiedad, deben tenerse en cuenta todas las ejecuciones producto de intercalar de todas las formas posibles la ejecución de sus acciones.

La forma de una especificación

- En TLA las especificaciones tiene una forma sintáctica restringida: $Init \wedge \Box[Next]_{vars} \wedge Fairness_{vars}$

  donde

  - *Next* es por lo general la disyunción de las acciones permitidas en el sistema;
  - *vars* son todas las variables del sistema;
  - *Fairness* es una conjunción numerable de fórmulas de WF y/o SF.

La forma de... (2)

- Esta sintaxis obedece a seis criterios:
  - Sirve para la mayoría de los sistemas
  - Dice que el primer estado de cualquier ejecución debe verificar *Init*, y cualquier par de estados de una ejecución deben estar relacionados por una de las acciones definidas
  - Las acciones controladas por la máquina se ejecutarán equitativamente
  - Siempre se obtienen máquinas cerradas
  - La fórmula resultante es invariante con respecto a los pasos intrascendentes
  - Respeta el teorema de Alpern-Schneider

$$
\begin{aligned}
&\langle s_0, s_1, \ldots \rangle [\![\mathit{Init} \wedge \Box [\mathit{Next}]_{\mathit{vars}} \wedge \mathit{Fairness}_v]\!] \\
&\quad \equiv \quad \langle s_0, s_1, \ldots \rangle [\![\mathit{Init} \wedge \Box (\mathit{Next} \vee \mathit{vars}' = \mathit{vars}) \wedge \mathit{Fairness}_{\mathit{vars}}]\!] \\
&\quad \equiv \quad \langle s_0, s_1, \ldots \rangle [\![\mathit{Init} \wedge \Box (A_1 \vee A_2 \vee \mathit{vars}' = \mathit{vars}) \wedge \mathit{Fairness}_{\mathit{vars}}]\!] \\
&\quad \equiv \quad \wedge \langle s_0, s_1, \ldots \rangle [\![\mathit{Init}]\!] \\
&\qquad\quad \wedge \langle s_0, s_1, \ldots \rangle [\![\Box (A_1 \vee A_2 \vee \mathit{vars}' = \mathit{vars})]\!] \\
&\qquad\quad \wedge \langle s_0, s_1, \ldots \rangle [\![\mathit{Fairness}_{\mathit{vars}}]\!] \\
&\quad \equiv \quad \wedge s_0 [\![\mathit{Init}]\!] \\
&\qquad\quad \wedge \forall n : \langle s_n, \ldots \rangle [\![A_1 \vee A_2 \vee \mathit{vars}' = \mathit{vars}]\!] \\
&\qquad\quad \wedge \langle s_0, s_1, \ldots \rangle [\![\mathit{Fairness}_{\mathit{vars}}]\!] \\
&\quad \equiv \quad \wedge \mathit{Init}(\forall v : s_0(v)/v) \\
&\qquad\quad \wedge \forall n : \quad \vee s_n [\![A_1]\!] s_{n+1} \\
&\qquad\qquad\qquad\quad \vee s_n [\![A_2]\!] s_{n+1} \\
&\qquad\qquad\qquad\quad \vee s_n [\![\mathit{vars}' = \mathit{vars}]\!] s_{n+1} \\
&\qquad\quad \wedge \langle s_0, s_1, \ldots \rangle [\![\mathit{Fairness}_{\mathit{vars}}]\!]
\end{aligned}
$$

$$
\begin{aligned}
&\equiv \\
&\wedge \mathit{Init}(\forall v : s_0(v)/v) \\
&\wedge \forall n : \quad \vee A_1(\forall v : s_n(v)/v, s_{n+1}(v)/v') \\
&\qquad\quad \vee A_2(\forall v : s_n(v)/v, s_{n+1}(v)/v') \\
&\qquad\quad \vee s_n [\![\mathit{vars}' = \mathit{vars}]\!] s_{n+1} \\
&\wedge \langle s_0, s_1, \ldots \rangle [\![\mathit{Fairness}_{\mathit{vars}}]\!]
\end{aligned}
$$

$Fairness_{vars}$

- Debe ser una conjunción numerable de fórmulas de la forma:
  - $\mathrm{WF}_{vars}(A) \triangleq \Box(\Box\mathrm{ENABLED}\ \langle A\rangle_{vars} \Rightarrow \Diamond\langle A\rangle_{vars})$
  - $\mathrm{SF}_{vars}(A) \triangleq \Box\Diamond\mathrm{ENABLED}\ \langle A\rangle_{vars} \Rightarrow \Box\Diamond\langle A\rangle_{vars}$

  donde
    - A es una *subacción* de *Next*, es decir $\Rightarrow$ *Next*
    - $\Diamond\langle A\rangle_{vars}$ es igual a $\neg\Box[\neg A]_{vars}$

$$[\neg A]_{vars} \equiv \neg A \vee vars' = vars \equiv \neg(A \wedge \neg(vars' = vars)) \equiv \neg(A \wedge vars' \neq vars)$$
$$\Rightarrow \Diamond\langle A\rangle_{vars} \equiv \Diamond(A \wedge vars' \neq vars)$$

$$
\begin{array}{lll}
\langle s_0, \ldots \rangle [\![\mathit{TimerSpec}]\!] & & \\
\quad \equiv & \langle s_0, \ldots \rangle [\![\mathit{InitTimer} \wedge \Box [\mathit{TimerNext}]_{sv} \wedge \mathit{WF}_{sv,\, now}(\mathit{Timeout})]\!] & \\
\quad \equiv & \wedge \quad s_0(\mathit{realtime}) \in \mathit{Nat} \wedge s_0(\mathit{running}) = \overline{\text{no}} & \\
& \wedge \quad \forall\, n: \quad \vee \quad \mathit{Start}(\forall\, v : s_n(v)/v, s_{n+1}(v)/v') & \\
& \qquad\qquad\quad \vee \quad \exists\, l \in \mathbb{N} \bullet \mathit{Set}(l)(\forall\, v : s_n(v)/v, s_{n+1}(v)/v') & \\
& \qquad\qquad\quad \vee \quad \mathit{Stop}(\forall\, v : s_n(v)/v, s_{n+1}(v)/v') & \\
& \qquad\qquad\quad \vee \quad \mathit{Timeout}(\forall\, v : s_n(v)/v, s_{n+1}(v)/v') & \\
& \qquad\qquad\quad \vee \quad s_{n+1}(sv) = s_n(sv) & \\
& \wedge \quad \forall\, n: \quad (\forall\, m: \quad \wedge \quad m \geqslant n & \\
& \qquad\qquad\qquad\qquad\quad \wedge \quad s_m(\mathit{running}) = \overline{\text{yes}} & \\
& \qquad\qquad\qquad\qquad\quad \wedge \quad s_m(\mathit{now} - \mathit{realtime}) \geq s_m(\mathit{limit})) & \\
& \qquad\qquad\quad \Rightarrow (\exists\, k: \quad \wedge \quad k \geqslant n & \\
& \qquad\qquad\qquad\qquad\quad \wedge \quad s_k(\mathit{running}) = \overline{\text{yes}} & \\
& \qquad\qquad\qquad\qquad\quad \wedge \quad s_k(\mathit{now} - \mathit{realtime}) \geq s_m(\mathit{limit})) & \\
& \qquad\qquad\qquad\qquad\quad \wedge \quad s_{k+1}(\mathit{running}) = \overline{\text{no}} & \\
& \qquad\qquad\qquad\qquad\quad \wedge \quad s_k(\langle \mathit{realtime}, \mathit{limit}, \mathit{now} \rangle) = s_{k+1}(\langle \mathit{realtime}, \mathit{limit}, \mathit{now} \rangle) & \\
& \qquad\qquad\qquad\qquad\quad \wedge \quad s_k(sv) \neq s_{k+1}(sv)) & \\
\end{array}
$$

Regla de conjunción para WF

- Si A_1 , ... , A_n son acciones tales que, para $i \neq j$, siempre que la acción A_i está habilitada, A_j no puede habilitarse hasta que se da un paso-A_i, entonces $\mathrm{WF}_v(A_1) \wedge \ldots \wedge \mathrm{WF}_v(A_n)$ es equivalente a $\mathrm{WF}_v(A_1 \vee \ldots \vee A_n)$.
  - Como la conjunción es una forma particular de la cuantificación universal, existe una regla similar para una cantidad numerable de acciones.
- Hay una regla igual para SF.

La visión externa del *timer*

MODULE *ETimer*

$timer(t, r, l) \triangleq$

INSTANCE *Timer* WITH $realtime \leftarrow t, running \leftarrow r, limit \leftarrow l$

$Spec \triangleq \exists t, r, l : timer(t, r, l)!TimerSpec$

- La cuantificación existencial utilizada se denomina *cuantificación existencial temporal*.
- $\exists v : F$ es cierta para una ejecución sí y sólo sí existe alguna secuencia de valores -uno en cada estado- que puede ser asignada a la variable v de manera de hacer F cierta.

Definición de la c.e.t

$$s \overset{x}{=} t \triangleq \forall\, v \neq x : s(v) = t(v)$$

Asignan lo mismo excepto en x

$$\not\downarrow\langle s_0, s_1, \ldots\rangle \triangleq \quad \textbf{\textit{if}}\ \forall\, n : s_n = s_0$$

$$\textbf{\textit{then}}\ \langle s_0, s_0, \ldots\rangle$$

$$\textbf{\textit{else}}\quad \textbf{\textit{if}}\ s_1 = s_0$$

$$\textbf{\textit{then}}\ \not\downarrow\langle s_1, s_2, \ldots\rangle$$

$$\textbf{\textit{else}}\ \langle s_0\rangle \circ \not\downarrow\langle s_1, s_2, \ldots\rangle$$

$\not\downarrow$ remueve los pasos intrascendentes excepto los finales

$$\sigma, \rho, \tau \in S^{\infty}$$

$$\sigma[\![\bar{\exists}\, x : F]\!] \triangleq \exists\, \rho, \tau : \not\downarrow\sigma = \not\downarrow\rho \wedge \rho \overset{x}{=} \tau \wedge \tau[\![F]\!]$$

Invariancia con respecto a los pasos intrascendentes