# Simulación – Proyecto 2015

**Especificación del sistema descripto utilizando el formalismo DEVS**

**Astorga, Darío**

**Bernal, Matías**

**Isoardi, Bruno**

$\boldsymbol{Generador} = \{X, Y, S, \delta_{int}, \delta_{ext}, \lambda, ta\}$

$X = \emptyset$

$Y = arrive \times \{0\}$

$S = IR^{+} + \{0\}$

$\lambda(s) = \lambda(\sigma) = (arrive, 0)$

$\delta_{\text{int}}(\text{s}) = \delta_{\text{int}}(\sigma) = \text{get\_next\_arrive}$

$\delta_{\text{ext}}(\text{s, e, x}) = \delta_{\text{ext}}(\sigma, \text{e}, \text{v}_{\text{x}}) = \sigma$

$ta(s) = ta(\sigma) = \sigma$

$S_0 = S = \text{get\_next\_arrive} \wedge t = 0$

*DATOS ADICIONALES:*

- get_next_arrive es una función que calcula el proximo arribo cada 10 unidades de tiempo en promedio distribuido exponencialmente.
- S es un real positivo o cero que utilizamos para generar salidas.

$\boldsymbol{Cinta} = \{X, Y, S, \delta_{int}, \delta_{ext}, \lambda, ta\}$

$X = \{arrive, start, stop\} \times \{0, 1\}$

$Y = \{detect, leave\} \times \{0\}$

$S = [IR^+ + \{0\}] \times \{true, false\} \times IR^+ + \{0\} \times IR^+ + \{0\}$

$$\lambda(s) = \lambda(list, running, rejects, \sigma) = \begin{cases} list.head = l \rightarrow (\text{detect}, 0) \\ list.head = d * l \rightarrow (leave. 0) \end{cases}$$

$\delta_{\text{int}}(\text{s}) = \delta_{\text{int}}(\text{list, running, rejects, } \sigma) =$

$$\begin{cases} list.head = l + d \rightarrow (sort\_list(list.tail, \sigma * vc), running, check\_next\_event(list)) \\ list.head = l \rightarrow (sort\_list(list, \sigma * vc), running, check\_next\_event(list)) \\ list.head = [\,] \rightarrow (list, running, \infty) \end{cases}$$

$\delta_{\text{ext}}(\text{s, e, x}) = \delta_{\text{ext}}(\text{list, running, rejects, } \sigma), \text{e, } (v_x, \text{port}) =$

$$\begin{cases} \neg running \wedge v_x = arrive \rightarrow (list, running, rejected + 1, \infty) \\ running \wedge v_x = arrive \\ \rightarrow (add\_to\_list(sort\_list(list.head,\ vc * e),\ \frac{l}{vc}),\ running,\ rejected, check\_next\_event(list)) \\ running \wedge v_x = stop \rightarrow (sort\_list(list,\ vc * e)),\ false,\ rejected, \infty) \\ \neg running \wedge v_x = start \rightarrow (list,\ true,\ rejected, check\_next\_event(list)) \end{cases}$$

$ta(s) = ta(\text{list, running, rejected, } \sigma) = \sigma$

$S_0 = S = (\text{[ ], false, 0, } \infty)$

*DATOS ADICIONALES:*

- sort_list es una función que ordena la lista de objetos de la cinta dejando en el head el más proximo a generar un evento.
- check_next_event es una función que de la lista de objetos devuelve el tiempo del evento más proximo.

- La implementación de la lista posee las funciones tail y head que devuelven el primero de la lista y el resto de la lista sin el primer elemento respectivamente.
- S es el producto cartesiano de una lista de reales positivos con el cero, que son las distancias recorridas de los objetos de la lista, un booleano que indica si la lista esta corriendo, un real positivo con cero que indica el número de rechazos de la cinta y real positivo o cero que utilizamos para generar salidas.

$\boldsymbol{Mesa} = \{X, Y, S, \delta_{int}, \delta_{ext}, \lambda, ta\}$

$X = \{rotate\_left, rotate\_right, move\_up, move\_down, arrive, pick\} \times \{0\}$

$Y = \{up, down, right, left, picked\} \times \{0\}$

$S = \{up, down\} \times \{right, left\} \times \{up, down, right, left, picked\} \times IR^{+} + \{0\}$

$\lambda(s) = \lambda(elevation, rotation, last, \sigma) = (last, 0)$

$\delta_{\text{int}}(\text{s}) = \delta_{\text{int}}(elevation, rotation, last, \sigma) = (elevation, rotation, last, \infty)$

$\delta_{\text{ext}}(\text{s, e, x}) = \delta_{\text{ext}}((elevation, rotation, last, \sigma), \text{e}, (\text{v}_\text{x}, \text{port})) =$

$$\begin{cases} \text{v}_\text{x} = move\_down \rightarrow (down, rotation, down, t_{\text{mov}}) \\ \text{v}_\text{x} = rotate\_right \rightarrow (elevation,\ right,\ right, t_{\text{mov}}) \\ \text{v}_\text{x} = move\_up \rightarrow (up, rotation,\ up, t_{\text{mov}}) \\ \text{v}_\text{x} = left \rightarrow (elevation,\ left,\ left, t_{\text{mov}}) \\ \text{v}_\text{x} = pick \rightarrow (elevation,\ rotation,\ picked, uniform(6, 16)) \end{cases}$$

$ta(s) = ta(\sigma) = \sigma$

$S_0 = S = (down, left, start, \infty)$

*DATOS ADICIONALES:*

- uniform es una función que devuelve el tiempo que tarda en hacer pick la mesa, con distribucion uniforme U [6,16].
- S es el producto cartesiano de un conjunto que indica si la mesa esta arriba o abajo, un conjunto que indica si la mesa esta a la izquierda o la derecha, un conjunto que indica cual fue la ultima entrada y un real positivo con cero que indica el número de rechazos de la cinta y real positivo o cero que utilizamos para generar salidas.

$$\boldsymbol{Unidad\ de\ Control} = \{X, Y, S, \delta_{int}, \delta_{ext}, \lambda, ta\}$$

$$X = \{detect, up, down, right, left, picked, leave\} \times \{0, 1\}$$

$$Y = \{start, stop, rotate\_left,\ rotate\_right, move\_up, move, down, arrive, pick\} \times \{0,1\}$$

$$S = \{true, false\} \times \{up, down\} \times \{left, right\} \times \{start, stop, detect, up, down, right, left, picked, leave\} \times IR^{+} + \{0\}$$

$$\lambda(s) = \lambda(running\_conveyor,\ table.elevation,\ table.rotation, last\_order, \sigma) =$$

$$\begin{cases} \neg running\_conveyor \wedge last\_order = start \rightarrow (start, 1) \\ running\_conveyor \wedge last\_order = stop \rightarrow (start, 1) \\ table.rotation = right \wedge last\_order = left \rightarrow (rotate\_left, 0) \\ table.rotation = left \wedge last\_order = right \rightarrow (rotate\_right, 0) \\ table.elevation = down \wedge last\_order = up \rightarrow (move\_up, 0) \\ table.elevation = up \wedge last\_order = down \rightarrow (move\_down, 0) \\ last\_order = leave \rightarrow (arrive, 0) \\ last\_order = picked \rightarrow (pick, 0) \end{cases}$$

$$\delta_{\text{int}}(\text{s}) = \delta_{\text{int}}(running\_conveyor, table.elevation, table.rotation, last\_order, \sigma) =$$

$$\begin{cases}
last_order = start \rightarrow (true, table.elevation, table.rotation, last_order, \infty) \\
last_order = stop \rightarrow (false, table.elevation, table.rotation, last_order, \infty) \\
last_order = up \rightarrow (running_conveyor, table.elevation,\ table.rotation, right, 0) \\
last_order = down \\
\quad \wedge \neg running_conveyor \rightarrow (running_conveyor, table.elevation, table.rotation, start, 0) \\
last_order = down \\
\quad \wedge running_conveyor \rightarrow (running_conveyor, table.elevation, table.rotation, start, \infty) \\
last_order = left \rightarrow (running_conveyor, table.elevation,\ table.rotation, down, 0) \\
last_order = right \rightarrow (running_conveyor, table.elevation,\ table.rotation, pick, 0) \\
last_order = picked \rightarrow (running_conveyor, table.elevation,\ table.rotation, left, 0) \\
last_order = leave \rightarrow (running_conveyor, table.elevation, table.rotation,\ up, 0) \\
last_order = detect \wedge (elevation = up \vee rotation = right) \\
\qquad \rightarrow (running_conveyor, table.elevation, table.rotation, stop, 0)
\end{cases}$$

$$last\_order = detect \wedge elevation = down \wedge rotation = left \\ \rightarrow (running\_conveyor, table.elevation, table.rotation, start, \infty)$$

$\delta_{ext}(s, e, x) = \delta_{ext}(running\_conveyor, table.elevation, table.rotation, last\_order, \sigma)$, e, ($v_x$, port)) =

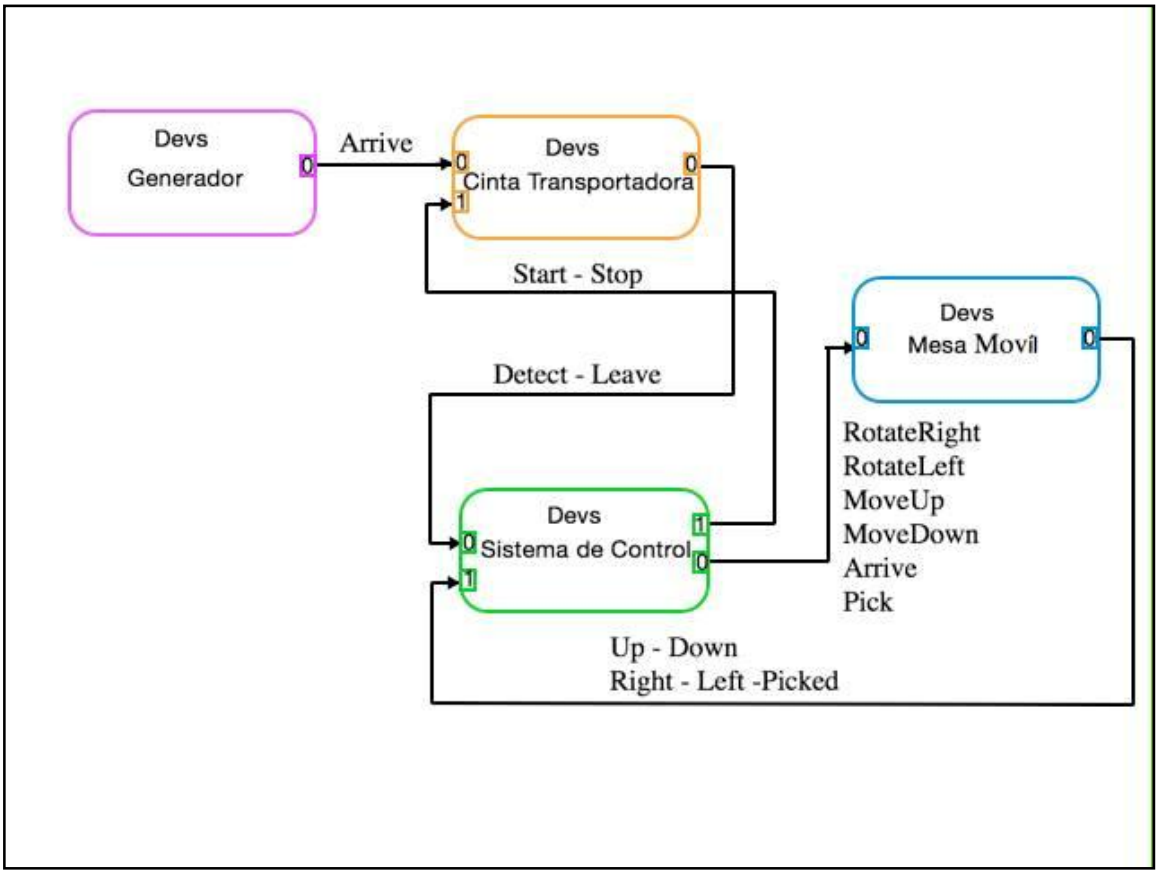
$$
\begin{cases}
v_x = detect \wedge (elevation = up \vee rotation = right) \\
\qquad \rightarrow (running\_conveyor, table.elevation, table.rotation, stop, 0) \\
v_x = detect \wedge table.elevation = down \wedge table.rotation = left \\
\qquad \rightarrow (running\_conveyor, table.elevation, table.rotation, start, \infty) \\
v_x = up \wedge table.elevation = down \rightarrow (running\_conveyor, up, table.rotation, up, 0) \\
v_x = down \wedge table.elevation = up \rightarrow (running\_conveyor, down, table.rotation, down, 0) \\
v_x = right \wedge table.rotation = left \rightarrow (running\_conveyor, table.elevation, right, right, 0) \\
v_x = left \wedge table.rotation = right \rightarrow (running\_conveyor, table.elevation, left, left, 0) \\
v_x = picked \rightarrow (running\_conveyor, table.elevation, table.rotation, picked, 0) \\
v_x = leave \rightarrow (running\_conveyor, table.elevation, table.rotation, up, 0)
\end{cases}
$$

$ta(s) = ta(running\_conveyor, table.elevation, table.rotation, last\_order, \sigma) = \sigma$

$S_0 = S = (false, table.down, table.left, start, \sigma)$

*DATOS ADICIONALES:*

- S es el producto cartesiano de un conjunto que indica si la cinta esta corriendo o no, un conjunto que indica si la mesa esta a arriba o abajo, un conjunto que indica si la mesa esta a la izquierda o la derecha, un conjunto que indica cual fue la ultima entrada y un real positivo con cero que indica el número de rechazos de la cinta y real positivo o cero que utilizamos para generar salidas.


Devs
Generador
0
Arrive
0
1
Devs
Cinta Transportadora
0
Start - Stop
Detect - Leave
Devs
Mesa Movîl
0
0
RotateRight
RotateLeft
MoveUp
MoveDown
Arrive
Pick
Devs
Sistema de Control
0
1
1
0
Up - Down
Right - Left -Picked