Statecharts Parametrizados

Maximiliano Cristiá
Ingeniería de Software 1
LCC – FCEIA – UNR
Rosario – Argentina
2024

1. Regla fundamental para el uso de Statecharts parametrizados

Los Statecharts parametrizados son una forma de expresar Statecharts complejos, esto significa que no hay una semántica específica para ellos sino que se debe usar la semántica de un Statechart no parametrizado. En consecuencia la regla para el uso de Statecharts parametrizados es la siguiente: un Statechart parametrizado es correcto si puede ser traducido automáticamente (mecánicamente, es decir con un algoritmo) a un Statechart no parametrizado. La posibilidad de traducción automática está directamente ligada a que la traducción no implique cuestiones semánticas sino únicamente sintácticas.

2. Requerimientos informales

Una habitación posee varias puertas de ingreso/egreso. En cada puerta se han instalado dos sensores ópticos uno al lado del otro (cada uno es semejante a los que se instalan en las puertas de los ascensores para evitar que se cierren si hay alguien entrando o saliendo). Estos sensores envían una señal cada vez que algo interrumpe el haz de luz que se establece entre cada extremo. Como hay dos por puerta es posible determinar si una persona ingresa a la habitación o sale de ella. Notar que esta inteligencia NO es parte de la funcionalidad de los sensores.

La habitación cuenta con un sistema de ventilación electrónico. Es posible aumentar o disminuir discretamente la cantidad de aire que el sistema hace circular.

Se requiere un programa que aumente/disminuya paulatinamente la circulación de aire desde cero hasta el máximo posible, cada vez que ingresan/egresan tres personas a la sala. Es decir la potencia de circulación debe ser la misma, por ejemplo, si hay 3, 4 o 5 personas en la sala.

3. Desginaciones

Como en este apunte solo modelaremos parte del conocimiento de dominio (DK) y la especificación (S) daremos únicamente las designaciones necesarias para describir esos documentos.

Se aumenta en una unidad la potencia del sistema de ventilación $\approx A$

Se disminuye en una unidad la potencia del sistema de ventilación $\approx D$

El sensor i de la puerta p emite una señal $\approx s(i, p)$

La cantidad de puertas de la habitación $\approx P$

La máxima potencia del sistema de ventilación $\approx MP$

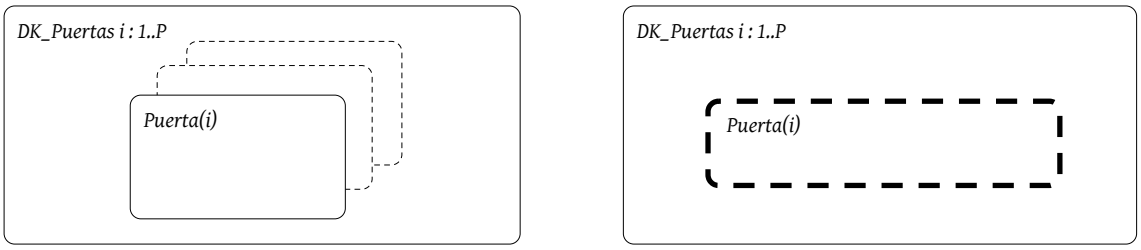


Figura 1: P puertas en paralelo usando la sugerencia de Harel, a la izquierda; a la derecha, usando la notación alternativa (usualmente más fácil de dibujar con herramientas de edición).

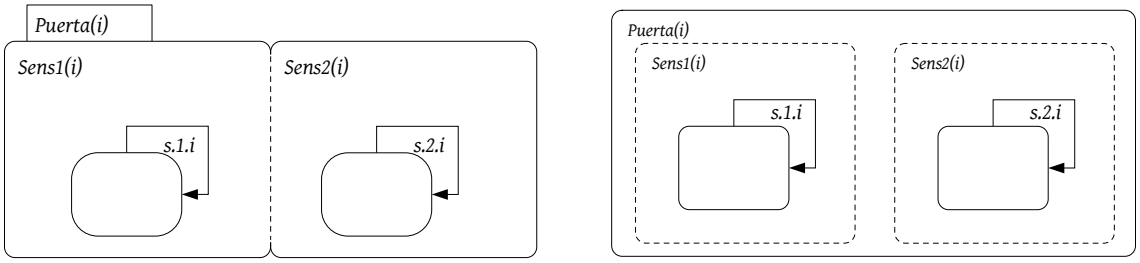


Figura 2: Un modelo simple para el DK de los sensores de cada puerta. A la izquierda la notación de Harel; a la derecha la notación alternativa.

4. El conocimiento del dominio (DK)

Como mencionamos más arriba sólo modelaremos una parte del conocimiento del dominio; el resto queda como ejercicio para el alumno. Comenzamos modelando P puertas en paralelo. Para esto usamos un Statechart parametrizado paralelo (o de tipo AND), ver Figura 1.

Notar que también hemos modificado la forma de nombrar los estados parametrizados, con *Puerta*(i), en lugar de con *Puerta i*.

Seguimos en la Figura 2 con el modelo de cada puerta la cual consiste en dos sensores que emiten una señal cada vez que algo los activa. Ambos sensores están en paralelo. Sin embargo, se espera que haya cierta serialización en las señales: si *Sens*1 se activa entonces luego se activa *Sens*2 y viceversa; dos activaciones consecutivas de uno de los sensores significa que la persona se arrepintió de entrar o salir, pero es imposible que dos personas activen una única vez cualquiera de los sensores ni que se active dos veces consecutivas uno de ellos sin que en el medio se active el otro. Para modelar todo esto o bien hay que modificar el modelo actual o bien agregar más conocimiento de dominio que indique que se está asumiendo tal comportamiento. Sin embargo, para mantener el problema simple no modelarmos ninguna de estas propiedades del domino del problema (el resto lo dejamos como ejercicio para el lector).

Notar que se utilizan los eventos designados para modelar las señales de los sensores.

Ahora pasamos al conocimiento de dominio referente al sistema de ventilación. Aquí utilizamos un Statechart parametrizado de tipo OR como se muestra en las Figuras 3 y 4. Harel, en lugar de elipses, utiliza rombos. El conjunto de estados parametrizados (*Pot*(j)) comprende los estados *Pot*(j) con j entre 1 y $MP-1$, lo que implica que ni *Pot*(0) ni *Pot*(MP) están en ese conjunto.

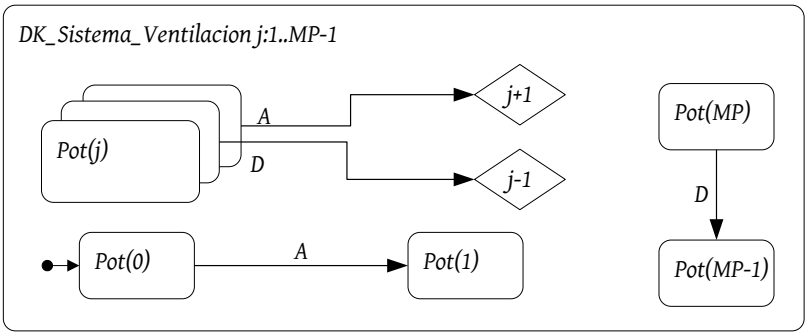


Figura 3: El conocimiento de dominio del sistema de ventilación, usando la notación de Harel

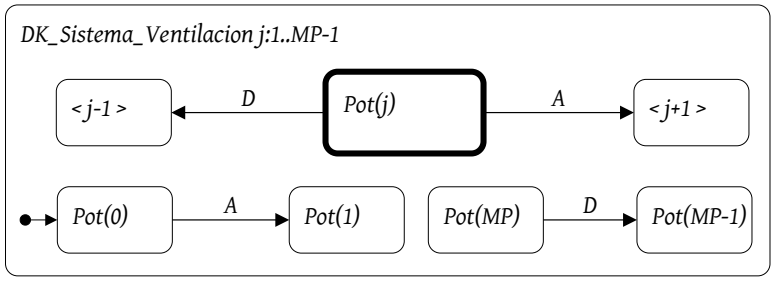


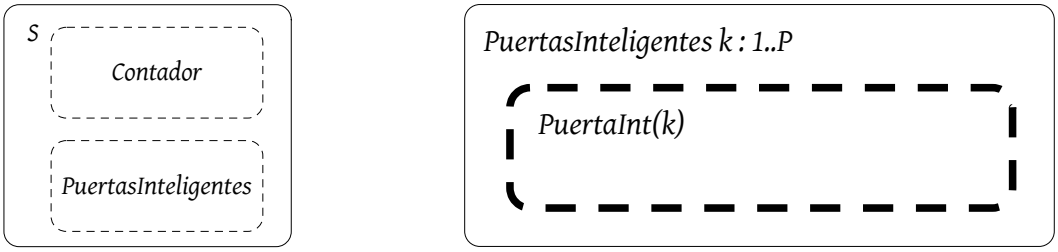
Figura 4: El conocimiento de dominio del sistema de ventilación, notación alternativa.

Una flecha saliendo del conjunto de estados parametrizados ($Pot(j)$) hacia un estado elipse significa que de cada estado (j) del conjunto sale una flecha idéntica al estado cuyo índice es el indicado en el elipse. En nuestro ejemplo, tenemos que de cada $Pot(j)$ sale una flecha etiquetada con A a $Pot(j+1)$ y una etiquetada con D a $Pot(j-1)$. Notar que si j vale 1 hay una flecha etiquetada con D a $Pot(0)$ aunque este último no es parte del conjunto; algo similar vale para $Pot(MP)$. Es responsabiliad del especificador que tales estados ($Pot(0)$ y $Pot(MP)$) existan en el Statechart. También, aunque no aparece en el ejemplo, una flecha saliente del conjunto de estados parametrizados hacia un estado común tiene el mismo significado: de cada estado parametrizado sale una flecha hacia ese estado con la misma etiqueta. Aunque tampoco aparece en este ejemplo, es posible tener flechas que entren al conjunto de estados parametrizados desde estados no parametrizados (comunes); en este caso el significado es similar: una flecha desde el estado común hacia cada uno de los estados parametrizados con la misma etiqueta (lo que es no determinístico).

No tiene mucho sentido tener flechas entrantes al conjunto provenientes desde un elipse.

5. La especificación (S)

Hemos especificado el sistema como dos máquinas en paralelo como se muestra en la Figura 5a. Comenzamos expandiendo *PuertasInteligentes* (Figuras 5b y 6) que, precisamente, hace lo que los sensores de cada puerta no pueden hacer por sí sólos: determinar cuándo una persona entra y cuándo sale. Para ellos pensamos en un ”proceso” por puerta que atiende las señales de



(a) La especificación al más alto nivel de abstracción

(b) *PuertasInteligentes* es simplemente la composición en paralelo de cada puerta

Figura 5: La especificación a alto nivel

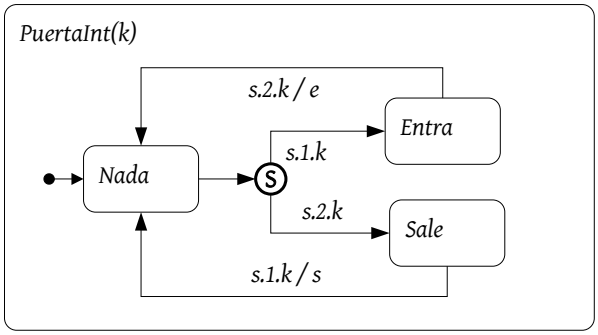


Figura 6: El modelo para cada puerta determina si una persona entra o sale según el orden de las señales provenientes de sus sensores.

sus sensores y determina si por esa puerta sale o entra una persona. Las salidas o entradas se comunican al resto del sistema por medio de los eventos s y e, respectivamente. Notar que en *PuertasInteligentes* el índice es k mientras que en *DK_Puertas* es i. Esto no causa problemas pues como indica la regla para la construcción de Statecharts parametrizados, luego de la traducción de ambos a Statecharts no parametrizados los índices desaparecen siendo reemplazados por constantes numéricas, por lo que, por ejemplo, los eventos de la forma $s.1.i$ y $s.1.k$ (de *DK_Puertas* y *PuertasInteligentes*) pasan a ser los mismos: $s.1.1$, $s.1.2$, etc.

En la Figura 7 mostramos la parte del sistema que cuenta las entradas y salidas según lo indicado por cada *PuertaInt* y comunica al sistema de ventilación si debe disminuir o aumentar la potencia cada tres salidas o entradas, respectivamente.

5.1. Un problema con la especificación

Si entra una cantidad suficiente de gente el sistema de ventilación llegará a su máxima potencia. Si continúa entrando gente, el sistema de software continuará enviando señales al sistema de ventilación para que este siga aumentando la potencia pero serán ignoradas. Ahora, si comienza a salir gente, el sistema de software pedirá al sistema de ventilación que reduzca la potencia cuando en realidad no debería hacerlo pues aun hay demasiada gente.

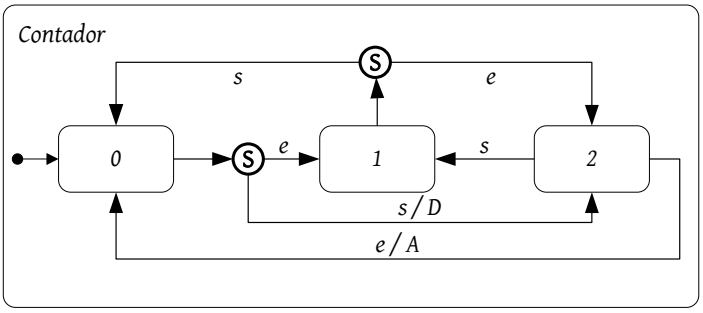


Figura 7: Contador de personas que entran o salen por cualquier puerta.

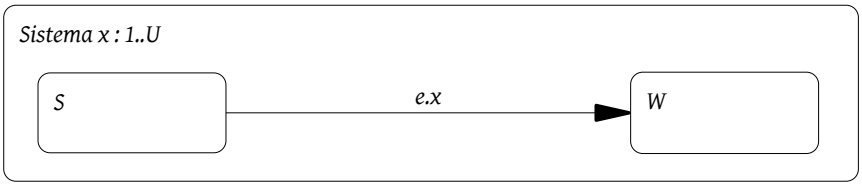


Figura 8: Parametrización de transiciones.

6. Más sobre Statecharts parametrizados

En esta sección mostramos algunos casos de Statecharts parametrizados que no aparecen en el ejemplo que hemos desarrollado más arriba.

Una forma de parametrización que no apareció en el ejemplo se da cuando es necesario que una máquina transicione debido a un gran número de eventos. Graficamos una situación así en la Figura 8. Ese Statechart parametrizado es equivalente a un Statechart no parametrizado en el cual salen U flechas desde el estado S hacia el estado W cada una de las cuales está etiquetada por un evento de la forma $e.i$ para $i \in [1, U]$.

Un ejemplo más avanzado lo vemos en la Figura 9a. En este caso tenemos un estado-OR parametrizado con p, del cual sale una flecha etiquetada con un evento parametrizado con q. Si bien en este caso ambos parámetros van de 1 a n, en general la cuantificación de cada parámetro podría ser sobre rangos diferentes. El estado $piso(1)$ es el estado inicial. Luego, de cada estado $piso(p)$ sale una flecha etiquetada $g.q$ que entra al estado $piso(q)$, para cada $q \in 1..n$. Es decir que si $n = 3$, de $piso(2)$ sale la flecha $g.1$ hacia el estado $piso(1)$, la flecha $g.2$ hacia el estado $piso(2)$ y la flecha $g.3$ hacia el estado $piso(3)$. En la Figura 9b podemos ver el Statechart no parametrizado correspondiente al de la Figura 9a, para $n = 3$.

También es posible incluir restricciones *in* relativas a estados parametrizados y relaciones aritméticas entre los parámetros en las condiciones de las transiciones, como se muestra en la Figura 10. Observar las transiciones que parten del conector condicional hacia los estados *Subiendo* y *Bajando*. En ellas usamos las restricciones $in\ PisoE(p)$, $in\ PisoA(q)$, $p > q$ y $p < q$. Los estados $PisoE(p)$ y $PisoA(q)$ están definidos en máquinas paralelas a *Mover*. Cuando el Statechart parametrizado se reescribe como uno no parametrizado, las variables p y q se sustituyen por valores, en cada flecha. De esta forma si $n = 3$ la flecha etiquetada con $[in\ PisoE(p) \wedge in\ PisoA(q) \wedge$

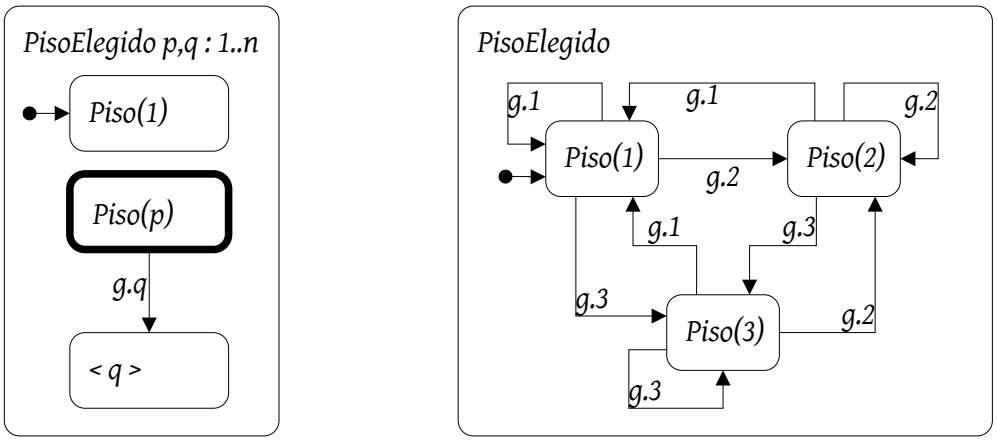


Figura 9: En (a), parametrización-OR combinada con parametrización de transiciones usando notación alternativa. En (b), Statechart no parametrizado equivalente para $n = 3$.

$p > q] \ / \ c \wedge up$ se reescribe como nueve flechas etiquetadas de la siguiente forma:

$[in\ PisoE(1) \wedge in\ PisoA(1) \wedge 1 > 1] \ / \ c \wedge up$
$[in\ PisoE(1) \wedge in\ PisoA(2) \wedge 1 > 2] \ / \ c \wedge up$
$[in\ PisoE(1) \wedge in\ PisoA(3) \wedge 1 > 3] \ / \ c \wedge up$
$[in\ PisoE(2) \wedge in\ PisoA(1) \wedge 2 > 1] \ / \ c \wedge up$
$[in\ PisoE(2) \wedge in\ PisoA(2) \wedge 2 > 2] \ / \ c \wedge up$
$[in\ PisoE(2) \wedge in\ PisoA(3) \wedge 2 > 3] \ / \ c \wedge up$
$[in\ PisoE(3) \wedge in\ PisoA(1) \wedge 3 > 1] \ / \ c \wedge up$
$[in\ PisoE(3) \wedge in\ PisoA(2) \wedge 3 > 2] \ / \ c \wedge up$
$[in\ PisoE(3) \wedge in\ PisoA(3) \wedge 3 > 3] \ / \ c \wedge up$

En un segundo paso, las flechas con condiciones imposible se descartan. En este caso sobreviven las siguientes:

$[in\ PisoE(2) \wedge in\ PisoA(1) \wedge 2 > 1] \ / \ c \wedge up$
$[in\ PisoE(3) \wedge in\ PisoA(1) \wedge 3 > 1] \ / \ c \wedge up$
$[in\ PisoE(3) \wedge in\ PisoA(2) \wedge 3 > 2] \ / \ c \wedge up$

También es interesante la flecha que parte del conector condicional y llega al estado *Detenido*. En este caso en la condición se usa un único parámetro: $[in\ PisoE(p) \wedge in\ PisoA(p)]$. Por lo tanto, la transición se reescribe como tres transiciones etiquetadas de la siguiente forma:

$[in\ PisoE(1) \wedge in\ PisoA(1)]$
$[in\ PisoE(2) \wedge in\ PisoA(2)]$
$[in\ PisoE(3) \wedge in\ PisoA(3)]$

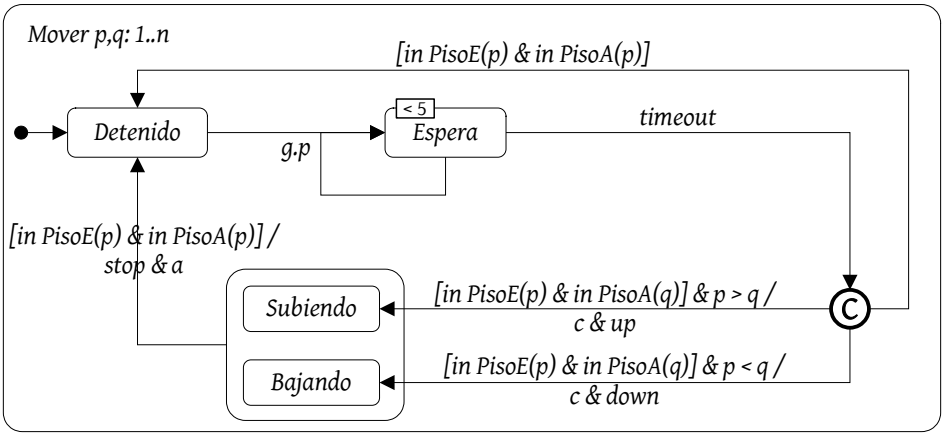


Figura 10: En la condición de una transición se pueden incluir relaciones aritméticas entre los parámetros. Los estados $PisoE(p)$ y $PisoA(p)$ son de máquinas que están en paralelo con *Mover*.

Lo que es equivalente a haber etiquetado la transición parametrizada con: $[in\ PisoE(p) \wedge in\ PisoA(q) \wedge p = q]$. Notar que esta transición complementa a las otras dos que parten del conector condicional. De esta forma las transiciones contemplan todas las posibles condiciones.

Dado que los estados $PisoE(p)$ y $PisoA(p)$ son de máquinas que están en paralelo con *Mover*, es responsabilidad del ingeniero definir rangos de cuantificación para p y q que sean subconjuntos de los rangos usados en las máquinas paralelas donde se definen $PisoE(p)$ y $PisoA(p)$. En el caso de la Figura 10, si $1..n$ no es un subconjunto de los rangos de definición de $PisoE(p)$ o $PisoA(p)$, habrá transiciones con condiciones que involucran estados inexistentes.

7. Nota final

Los problemas como el mostrado en este ejemplo llevan a Statecharts a su límite expresivo. Es posible que sea conveniente utilizar lenguajes como CSP o TLA, o combinar Statecharts con Z para este tipo de problemas.

Ejercicio 1 *Modificar el conocimiento de dominio respecto de los sensores de cada puerta de forma tal que se cumplan las siguientes propieades:*

- *Dos activaciones consecutivas de un sensor significa que la persona se arrepintió de entrar o salir.*
- *Es imposible que se active dos veces consecutivas uno, sin que en el medio se active el otro.*
- *El primero en activarse siempre será Sens*1.

Ejercicio 2 *Busque una solución para el problema planteado en la Sección 5.1.*