

Universidad Nacional de Rosario

FACULTAD DE CIENCIAS EXACTAS, INGENIERÍA Y AGRIMENSURA

PARCIALES STATECHARTS+CSP

Autor:
Joaquin Arroyo

Índice

1. Teoría	2
1.1. Ejercicio 1	2
1.2. Ejercicio 2	3
1.3. Ejercicio 3	3
1.4. Ejercicio 4	4
1.5. Ejercicio 5	4
1.6. Ejercicio 6	5
1.7. Ejercicio 7	5
1.8. Ejercicio 8	5
1.9. Ejercicio 9	5
1.10. Ejercicio 10	5
1.11. Ejercicio 11	6
1.12. Ejercicio 12	6
2. Práctica Statecharts	8
2.1. Ejercicio 1	8
2.2. Ejercicio 2	9
2.3. Ejercicio 3	11
2.4. Ejercicio 4	12
2.5. Ejercicio 5	12
2.6. Ejercicio 6	13
2.7. Ejercicio 7	13
2.8. Ejercicio 8	14
2.9. Ejercicio 9	15
2.10. Ejercicio 10	15
2.11. Ejercicio 12	16
2.12. Ejercicio 14	16
2.13. Ejercicio 15	17
3. Práctica CSP	18
3.1. Ejercicio 1	18
3.2. Ejercicio 2	19
3.3. Ejercicio 4	20
3.4. Ejercicio 5	21
3.5. Ejercicio 6	22
3.6. Ejercicio 9	23
3.7. Ejercicio 10	23
3.8. Ejercicio 11	24
3.9. Ejercicio 12	25
3.10. Ejercicio 13	26
3.11. Ejercicio 15	28
3.12. Ejercicio 16	28
3.13. Ejercicio 17	29

1. Teoría

1.1. Ejercicio 1

Para cada uno de los procesos siguientes, encuentre el proceso secuencial equivalente, justificando cada paso de su cálculo.

$$a) \quad P = a \rightarrow b \rightarrow STOP \| (a \rightarrow a \rightarrow STOP \sqcap a \rightarrow c \rightarrow STOP)$$

$$b) \quad Q = (b \rightarrow b \rightarrow STOP \square a \rightarrow b \rightarrow STOP) \| a \rightarrow c \rightarrow STOP$$

$$c) \quad R = c \rightarrow a \rightarrow STOP \| (a \rightarrow b \rightarrow STOP \square b \rightarrow b \rightarrow STOP)$$

a)

$$a \rightarrow b \rightarrow STOP_{ab} \| (a \rightarrow a \rightarrow STOP_a \sqcap a \rightarrow c \rightarrow STOP_{ac}) \quad (1)$$

$$\equiv < e \rightarrow P \sqcap e \rightarrow Q = e \rightarrow (P \sqcap Q) >$$

$$a \rightarrow b \rightarrow STOP_{ab} \| a \rightarrow (a \rightarrow STOP_a \sqcap c \rightarrow STOP_{ac}) \quad (2)$$

$$\equiv < Synchronization >$$

$$a \rightarrow (b \rightarrow STOP_{ab} \| (a \rightarrow STOP_a \sqcap c \rightarrow STOP_{ac})) \quad (3)$$

$$\equiv < Distributive >$$

$$a \rightarrow ((b \rightarrow STOP_{ab} \| a \rightarrow STOP_a) \sqcap (b \rightarrow STOP_{ab} \| c \rightarrow STOP_{ac})) \quad (4)$$

$$\equiv < Ley.indp >$$

$$a \rightarrow ((b \rightarrow (a \rightarrow (STOP_{ab} \| STOP_a)) \sqcap (b \rightarrow STOP_{ab} \| c \rightarrow STOP_{ac}))) \quad (5)$$

$$\equiv < Interleaving >$$

$$a \rightarrow ((b \rightarrow (a \rightarrow (STOP_{ab} \| STOP_a)) \sqcap (b \rightarrow (STOP_{ab} \| c \rightarrow STOP_{ac}) \square (c \rightarrow (b \rightarrow STOP_{ab} \| STOP_{ac})))) \quad (6)$$

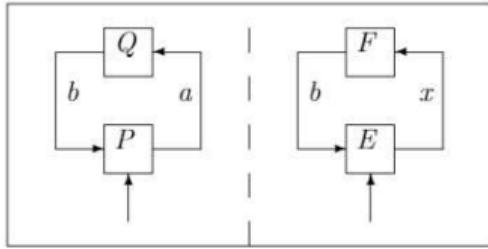
$$\equiv < [\{a, b\} \cap \{a\} \neq \emptyset], [\{a, b\} \cap \{a, c\} \neq \emptyset], [\{a, c\} \cap \{a, b\} \neq \emptyset] >$$

$$a \rightarrow (b \rightarrow STOP_{ab} \sqcap (b \rightarrow STOP_{abc} \square c \rightarrow STOP_{acb})) \quad (7)$$

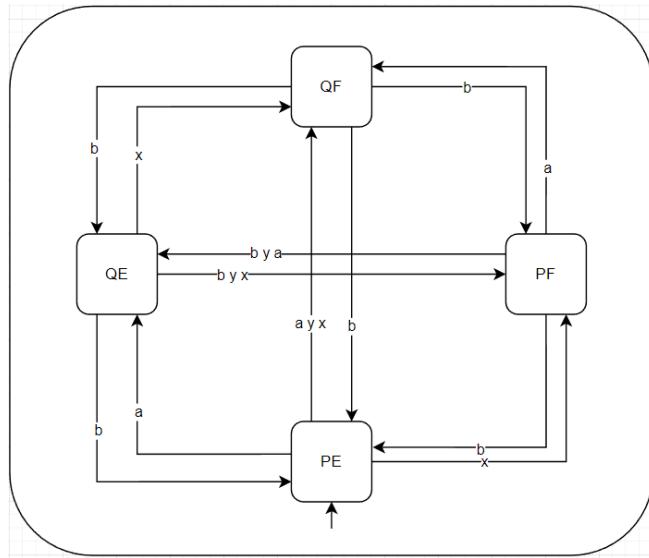
Luego, el proceso es secuencial.

1.2. Ejercicio 2

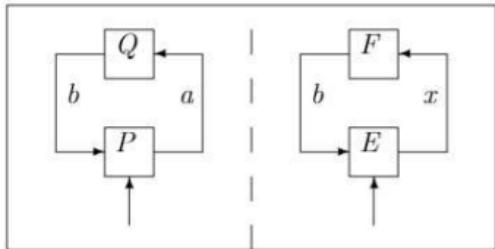
Muestre un Statechart equivalente al que se muestra en la figura que no haga uso de los estados AND.



Se hace un producto cartesiano entre los estados de ambos procesos en paralelo.



1.3. Ejercicio 3



Tomado en: 2015.06.10(r2).

Traduzca el Statechart de la figura anterior en un proceso CSP equivalente. Si no puede hacerlo, justifique su respuesta.

$$PR = P \parallel E$$

$$P = a \rightarrow Q$$

$$Q = b \rightarrow P$$

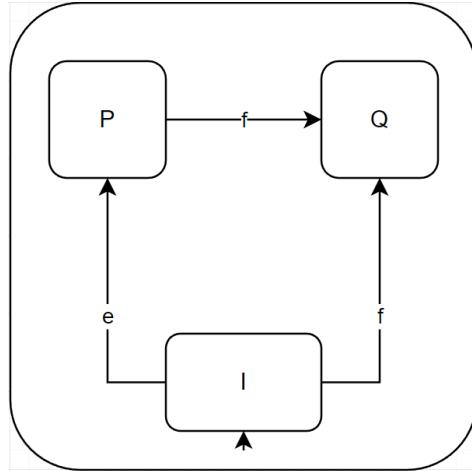
$$E = x \rightarrow F$$

$$F = b \rightarrow E$$

1.4. Ejercicio 4

Explique la diferencia entre los operadores \square y ∇ . Luego muestre cómo representar ∇ en Statecharts.

La diferencia entre $e \rightarrow P \square f \rightarrow Q$ y $e \rightarrow P \nabla f \rightarrow Q$ es que, el primero, ofrece al entorno poder interactuar con e y f , pero cuando entró en alguno de los dos, ya no puede ir por el otro. En cambio, en el segundo caso, también se ofrece poder interactuar con e y f , pero, en caso de que se aparezca el evento e , y luego aparezca el evento f , el proceso comienza a comportarse como Q ; si llega primero el evento f , directamente el proceso se comporta como Q . Es decir, el segundo operador ofrece la posibilidad de interrupción sobre el proceso de la izquierda.



1.5. Ejercicio 5

Modele un proceso CSP que debe recibir cien señales en cualquier orden y, una vez que las ha recibido a todas, debe comportarse como el proceso W .

$$COUNTER = startC?n \rightarrow C(n)$$

$$C(0) = \overline{all} \rightarrow COUNTER$$

$$C(n + 1) = signal \rightarrow C(n)$$

$$P = (startC!100 \rightarrow all \rightarrow W) \parallel COUNTER$$

1.6. Ejercicio 6

Explique en pocas líneas el concepto semántico utilizado en CSP que permite formalizar la idea de no determinismo.

El concepto semántico que permite formalizar el no determinismo en CSP son los rechazos. Estos son eventos que un proceso puede no ejecutar sin importar por cuanto tiempo el entorno los ofrezca.

$a \rightarrow P \square b \rightarrow Q$, el entorno ofrece el evento a o b , y el proceso debe ejecutar.

$a \rightarrow P \sqcap b \rightarrow Q$, el entorno ofrece el evento a o b , y el proceso puede ejecutar, o puede rechazar.

$\{a\}$ y $\{b\}$ son rechazos iniciales del proceso.

Rechazo \Rightarrow no determinismo.

1.7. Ejercicio 7

Explique las diferencias y semejanzas entre los operadores CSP: $|$, \square y \sqcap .

La semántica entre $|$ y \square es similar, ambos ofrecen al entorno la posibilidad de comportarse como alguno de los procesos que ofrecen. La diferencia entre $|$ y \square es puramente sintáctica. El primero, solo permite expresiones de la forma $e \rightarrow P | f \rightarrow Q$, es decir, cada alternativa etiquetada que se utilice, debe estar explícitamente precedida por su etiqueta (primer evento). En cambio, en el segundo operador, esto no es necesario, es legal escribir cosas como $P = e \rightarrow Q \square R$, $e \rightarrow P \square e \rightarrow Q$, $P = \square_{i=1}^n e_i \rightarrow P_i$, $P = \square i \in [1, n] \bullet e_i \rightarrow P_i$, o $Q \square R$.

La diferencia de estos dos operadores, con el operador \sqcap es que este operador decide por si mismo si se comporta como el proceso a su izquierda, o a su derecha. Tenemos un NO determinismo, ya que el entorno no puede decidir como se comporta este proceso.

1.8. Ejercicio 8

Explique con la debida precisión la semántica del modelo de concurrencia de CSP.

Hay que explicar el tema de la sincronización y las leyes fundamentales. (Volver a consultar)

1.9. Ejercicio 9

Muestre un ejemplo sencillo donde se pueda apreciar la diferencia de los modelos de concurrencia de Statecharts y CSP.

En Statecharts esta relación entre BELTL y WL estaba regida por el modelo broadcast: BELTL anunciaría i y WL transicionaría si el evento llegaba en el momento preciso, caso contrario se perdería. CSP, por el contrario, se rige por un modelo sincrónico: si dos procesos comparten un evento, ambos deben estar dispuestos a consumirlo al mismo tiempo, caso contrario uno de ellos deberá esperar a que el otro proceso esté en condiciones de hacerlo; es decir, los eventos no se pierden, sino que los procesos se bloquean.

1.10. Ejercicio 10

Escriba en CSP y explique cada una de las siguientes leyes: “Interleaving”, “Deadlock”, Sincronización y “IEOF”.

1. $e \notin \alpha P, f \notin \alpha Q \Rightarrow f \rightarrow P || e \rightarrow Q = f \rightarrow (P || e \rightarrow Q) \square e \rightarrow (f \rightarrow P || Q)$ (interleaving).

Esta ley dice que si un proceso es el resultado de la composición paralela de dos procesos cuyos primeros eventos no son comunes entre ellos, entonces, desde el entorno, el proceso se comportará como si ofreciera ambas posibilidades.

2. $e \rightarrow P \parallel e \rightarrow Q = e \rightarrow (P \parallel Q)$ (Ley de sincronización). Esta ley dice que si un proceso es el resultado de la composición paralela de dos procesos cuyos primeros eventos son iguales, entonces, desde el entorno, se ofrece dicho proceso, y se sigue con la parallelización de los procesos correspondientes.
3. $e \in \alpha Q, f \notin \alpha P \Rightarrow e \rightarrow P \parallel f \rightarrow Q = f \rightarrow (e \rightarrow P \parallel Q)$ (Ley de eventos independientes). Esta ley establece que si un proceso es el resultado de la composición paralela de dos procesos, donde uno de sus dos primeros eventos es común entre ellos, entonces, el entorno ofrece en primer momento el proceso que NO es común, y luego el proceso que es común, seguido por la parallelización de los procesos correspondientes.
4. $. e \in \alpha Q, f \in \alpha P \Rightarrow e \rightarrow P \parallel f \rightarrow Q = STOP$ (Ley de Deadlock). Esta ley establece que si un proceso es el resultado de la composición paralela de dos procesos cuyos primeros eventos son comunes entre ellos, entonces, tenemos un deadlock. STOP es un proceso que forma parte del lenguaje, es como una constante. Este proceso no ejecuta ningún evento, no hace nada, indica terminación anormal.

1.11. Ejercicio 11

Determine si la siguiente proposición es correcta o no (es decir si la igualdad es correcta o no) y justifique su respuesta.

$$c?x \rightarrow P; Q = c?x \rightarrow (P; Q)$$

Esta proposición es ley sí y sólo sí x no es una variable libre en Q , ya que como CSP es un lenguaje declarativo, una expresión como $c?x \rightarrow P$ declara un identificador x que almacenará el mismo valor sólo durante la vida de P , pues una vez que este haya terminado se habrá abandonado el ámbito de x . Por ejemplo, si tenemos:

$P = c?x \rightarrow SKIP; d!x \rightarrow STOP$ y $Q = c?x \rightarrow (SKIP; d!x \rightarrow STOP)$
en P , la variable x en c y en d son independientes, en cambio en Q , van a ser la misma variable, por lo que no se cumple la igualdad.

1.12. Ejercicio 12

Describa el modelo de fallas y divergencias de CSP, explicando y definiendo los conceptos de traza, rechazo, falla y divergencia.

Las fallas se dividen en **trazas** y **rechazos**.

Trazas: Secuencias de eventos comunicados por un proceso. Conjunto de trazas de $P \rightarrow tP$ o $t(P)$. Pueden ser secuencias finitas o infinitas.

$$\begin{aligned} P = a \rightarrow b \rightarrow STOP &\Rightarrow tP = \{\langle \rangle, \langle a \rangle, \langle a, b \rangle\} \\ P = e \rightarrow P &\Rightarrow tP = \{\langle \rangle, \langle e \rangle, \langle e, e \rangle, \dots\} \end{aligned}$$

Rechazos: Eventos que un proceso puede no ejecutar sin importar por cuanto tiempo el entorno los ofrezca.

$a \rightarrow P \square b \rightarrow Q$, el entorno ofrece el evento a o b , y el proceso debe ejecutar.
 $a \rightarrow P \sqcap b \rightarrow Q$, el entorno ofrece el evento a o b , y el proceso puede ejecutar, o puede rechazar.
 $\{a\}$ y $\{b\}$ son rechazos iniciales del proceso.

Rechazo \Rightarrow no determinismo.

Se define el conjunto de conjuntos de rechazos iniciales de un proceso P como rP o $r(P)$.

Fallas(Trazas × Rechazos): Definimos P/s como el proceso P luego de ejecutar la traza $s \in tP$. $r(P/s)$ es el conjunto de conjuntos de rechazos iniciales de P/s .

Las fallas de un proceso P se definen como fP o $f(P)$, y $fP = \{(s, A) \mid s \in tP \wedge A \in r(P/s)\}$, luego, esta es la semántica de las fallas.

Divergencia: Es la posibilidad de que un proceso entre en una secuencia infinita de acciones internas. Claramente un proceso que diverge es inútil, incluso más que STOP, puesto que no tiene comunicación con el entorno y ni siquiera rechaza nada (en el sentido de $r(P)$).

CSP considera que una vez que un proceso diverge no es posible saber con precisión qué es lo que hace después por lo que se considera que dos procesos que divergen son equivalentes (iguales).

2. Práctica Statecharts

2.1. Ejercicio 1

Tomado en: 2009.08.07(r2).

Un sistema debe controlar un aparato para efectuar electroencefalogramas simples. El análisis consiste en estudiar el voltaje que emiten 10 electrodos que permiten conocer la actividad bioeléctrica cerebral (cada uno comunica un valor al sistema). Es necesario tomar 5 muestras por segundo, espaciadas uniformemente. En cada una de las 5 muestras se lee el valor de los 10 electrodos. Notar que si el cerebro del paciente no presenta actividad en las cercanías de un electrodo, este no emitirá señal alguna. Por lo tanto, el sistema no puede esperar indefinidamente por la señal del electrodo. Finalmente, el sistema debe enviar secuencialmente a una impresora el valor obtenido en cada electrodo (que haya retornado uno o nada en caso de que no se haya registrado ninguno).

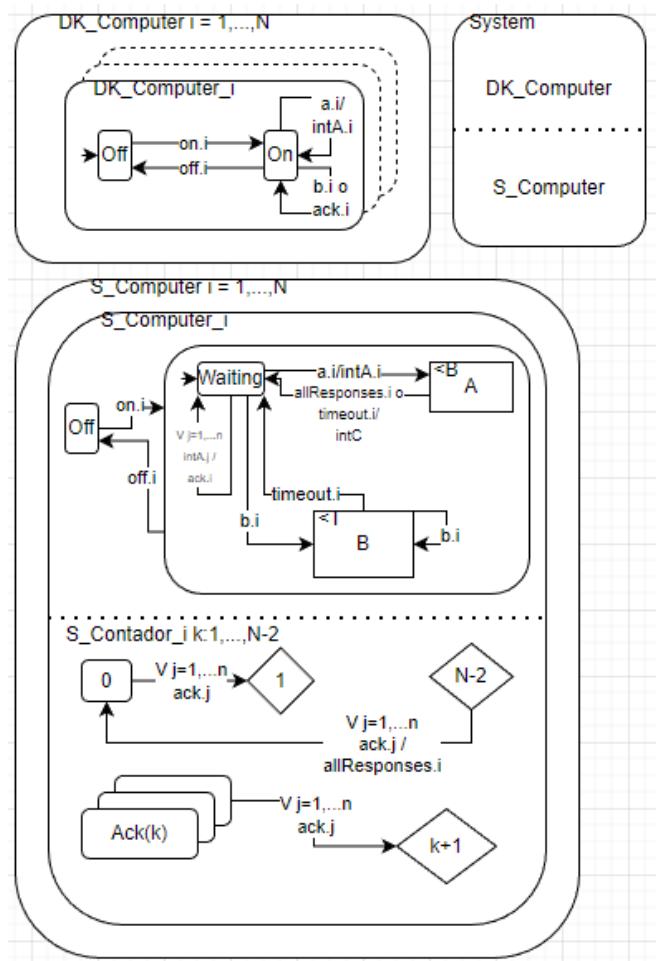
- a) Modele en Statecharts la especificación y el conocimiento del dominio del problema anterior.
- b) Escriba las designaciones y modele en CSP el conocimiento de dominio y la especificación de los requerimientos enunciados en el problema. Para las cuestiones temporales puede asumir la existencia de un temporizador como el que se mostró en clase.

2.2. Ejercicio 2

Ejercicio 2 Tomado en: 2010.06.15(p2).

Un sistema distribuido consta de N computadoras cada una de las cuales ejecuta el mismo programa; estas computadoras pueden estar encendidas o apagadas. Este programa recibe dos eventos desde el exterior (en cada computadora), a y b . Cuando el programa en una computadora recibe a , debe comunicarles a todas las otras computadoras, por medio de un evento interno, que lo ha recibido. Luego de comunicar esta situación esa computadora debe esperar a que todas le respondan. Cada computadora encendida responde con un evento diferente pero solo si está en su estado inicial. Como puede haber computadoras apagadas o que estén haciendo otra cosa, el programa esperará un tiempo B cada respuesta. Cuando recibe todas las respuestas o han transcurrido B unidades de tiempo, debe emitir el evento c y volver al estado inicial. Si el programa recibe el evento b debe esperar un tiempo t hasta poder recibir el evento a a menos que se dé otro evento b luego de T unidades de tiempo, en cuyo caso debe reiniciar el tiempo de espera.

- a) Designe los términos más importantes del enunciado.
 - b) Describa en Statecharts la especificación del programa mencionado en el problema.
 - c) Describa en CSP la especificación del programa mencionado en el problema.
-
- La computadora c se prende $\approx \text{on}(c)$. EC, S
 - La computadora c se apaga $\approx \text{off}(c)$. EC, S
 - La computadora c recibe la señal a $\approx a(c)$. EC, S
 - La computadora c recibe la señal b $\approx b(c)$. EC, S
 - La computadora c envia la señal interna a $\approx \text{intA}(c)$. MC, S
 - La computadora c envia la señal interna c $\approx \text{intC}(c)$. MC, S
 - La computadora c envia ack indicando que recibio intA $\approx \text{ack}(c)$. MC, S
 - B es el tiempo en el que todas las computadoras esperan el evento ack de todas las demás computadoras.
 - El numero total de computadoras (prendidas y apagadas) es N .
 - T es el tiempo que espera la computadora c luego de recibir b , antes de poder realizar alguna otra acción.
 - El sistema le indica a la computadora c que recibio todas las respuestas ack $\approx \text{allresponses}(c)$. MC, S



2.3. Ejercicio 3

Tomado en: 2015.06.10(p2).

Un sistema de alarma hogareño simple consta de un teclado (contiene los 10 dígitos y las dos primeras letras del abecedario), dos sensores de movimiento y dos magnéticos (para aberturas) y una bocina. Todo el sistema será controlado por un programa.

El sistema puede estar armado o no. Armado significa que si se detecta una intrusión debe sonar la bocina que se teclee el código de seguridad. Si no está armado no se responderá a las intrusiones. Una intrusión se detecta cuando alguno de los sensores envía una señal. Si está armado, al teclear el código de seguridad, el sistema se desarma; si está desarmado, al teclear el código de seguridad, se arma.

El código de seguridad viene de fábrica y consta de 2 dígitos.

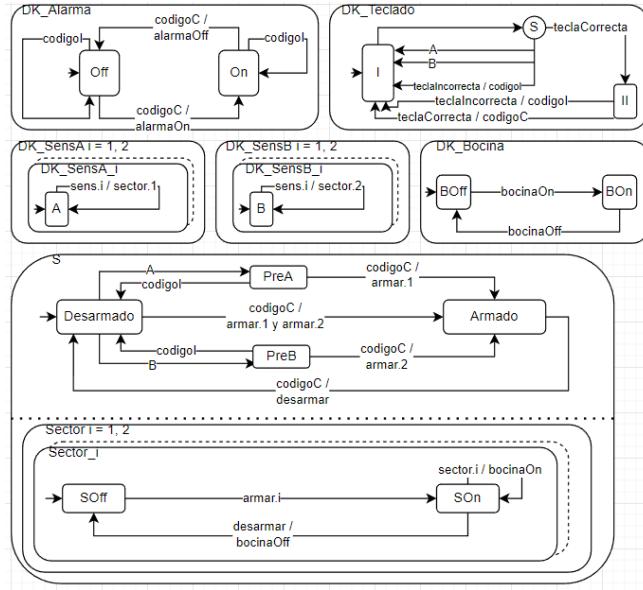
El sistema agrupa un sensor de movimiento y uno magnético en el sector A y a los sensores restantes en el sector B. Cada sector se puede armar independientemente del otro. Para hacerlo se teclea su letra y el código de seguridad. Si un sector no está armado no se responderá a las intrusiones en ese sector. Si solo se teclea el código de seguridad se arman todos los sectores; si algún sector está armado y se teclea el código de seguridad, se desarmán todos los sectores.

- a) Designe los fenómenos de interés, escribir la tabla de control y modelar en Statecharts el conocimiento de dominio y la especificación de los requerimientos anteriores, abstrayendo convenientemente el teclado y el código de seguridad.

14

- b) Utilizando las mismas designaciones modele en CSP el conocimiento de dominio y la especificación de los requerimientos enunciados en el problema.

- Se ingresa el código correcto $\approx \text{codigoC}$. EC, S
- Se ingresa el código incorrecto $\approx \text{codigoI}$. EC, S
- El sistema activa la alarma $\approx \text{alarmaOn}$. MC, S
- El sistema desactiva la alarma $\approx \text{alarmaOff}$. MC, S
- El usuario presiona la tecla A $\approx \text{teclaA}$. EC, S
- El usuario presiona la tecla B $\approx \text{teclaB}$. EC, S
- El usuario presiona la tecla correcta $\approx \text{teclaCorrecta}$. EC, S (Se entiende por tecla correcta al próximo dígito del código)
- El usuario presiona la tecla incorrecta $\approx \text{teclaIncorrecta}$. EC, S
- El teclado indica al sistema que se ingresó el código correcto $\approx \text{codigoC}$. EC, S
- El teclado indica al sistema que se ingresó el código incorrecto $\approx \text{codigoI}$. EC, S
- El sensor s detecta un movimiento $\approx \text{sens(s)}$
- El sensor del sector i indica al sistema que detectó movimiento $\approx \text{sector(i)}$
- El sistema enciende la bocina $\approx \text{bocinaOn}$. MC, S
- El sistema apaga la bocina $\approx \text{bocinaOff}$. MC, S
- El sistema arma el sector s $\approx \text{armar(s)}$. MC, S
- El sistema desarma los sectores $\approx \text{desarmar}$. MC, S



2.4. Ejercicio 4

Tomado en: 2015.06.17(r2).

Se cuenta con un proceso *S* que envía mensajes, un medio *E* que los transmite y un proceso *R* que los recibe. *E* puede almacenar solo un mensaje a la vez, puede corromper a lo sumo uno de cada tres mensajes consecutivos y, por simplicidad, se supone que cada mensaje es un 0 o un 1; *E* está fuera del control del sistema, es decir es parte del entorno. Se espera que *S* y *R* colaboren para evitar la corrupción producida por *E*. Para ello utilizarán un mecanismo simple: *S* enviará cada mensaje por triplicado y *R* decidirá cuál es el válido de acuerdo a una elección por mayoría simple.

Especifique en Statecharts los procesos *S*, *E* y *R* descriptos arriba.

2.5. Ejercicio 5

Tomado en: 2015.06.17(r2).

Se requiere el software de control para un robot industrial con las siguientes características. El robot puede moverse hacia la izquierda o derecha una cierta distancia y consta de un rociador y termómetro. El robot está ubicado al borde de una cinta transportadora que acarrea piezas metálicas. El requerimiento básico es que si el sensor detecta que la temperatura de la pieza que está en su cercanía es superior a *maxtemp*, el robot debe abrir el rociador y debe moverse en el sentido de la cinta hasta que la temperatura de la pieza sea inferior a *maxtemp*. Si el robot llega hasta su límite de desplazamiento debe cerrar el rociador. En cualquier caso debe retornar a la posición inicial.

Tener en cuenta lo siguiente:

- El termómetro envía una señal con la temperatura medida.
 - El motor del robot envía una señal indicando que se ha alcanzado el extremo derecho o izquierdo.
 - El software de control puede encender el motor en uno u otro sentido y detenerlo.
 - No se puede poner en funcionamiento el motor en el mismo paso en que se recibe una medición de temperatura.
- a) Describa en Statecharts la especificación y el conocimiento del dominio del problema que se enumera arriba. Abstraiga convenientemente las diferentes temperaturas que puede comunicar el termómetro.
- b) Describa en CSP la especificación y el conocimiento del dominio de los requerimientos enumerados en el problema.

2.6. Ejercicio 6

Un productor produce de a un mensaje a la vez a razón de 3 segundos cada uno. Existen otros dos productores que se comportan de la misma forma pero trabajan con tiempos de 1 y 4 segundos, respectivamente. Todos los productores tienen un error en sus relojes de $\pm\delta$ segundos con $0 < \delta < 1$.

Los mensajes enviados por los tres productores son recibidos por un componente que puede recibir de a un mensaje a la vez. El componente cuenta con un “buffer” con capacidad para MB mensajes. Poner o sacar un mensaje del “buffer” le toma 0,1 segundos.

Por otro lado, hay dos consumidores de mensajes que pueden consumir de a un mensaje a la vez cada uno, a razón de 2 segundos por mensaje. El componente comunica los mensajes según una política FIFO.

Tener en cuenta que los productores y consumidores son las entidades activas.

- a) Escriba las designaciones y la tabla de control y visibilidad del problema anterior.
- b) Modele en Statecharts el conocimiento del dominio (productores y consumidores) y la especificación del componente de almacenamiento.
- c) Modele en CSP el conocimiento del dominio (productores y consumidores) y la especificación del componente de almacenamiento.

15



- d) Explique claramente (mostrando un ejemplo si es necesario) si el modelo semántico “broadcast” de Statecharts hace que el sistema completo se comporte de forma diferente a como lo hace la especificación CSP, que responde a un modelo semántico sincrónico.

2.7. Ejercicio 7

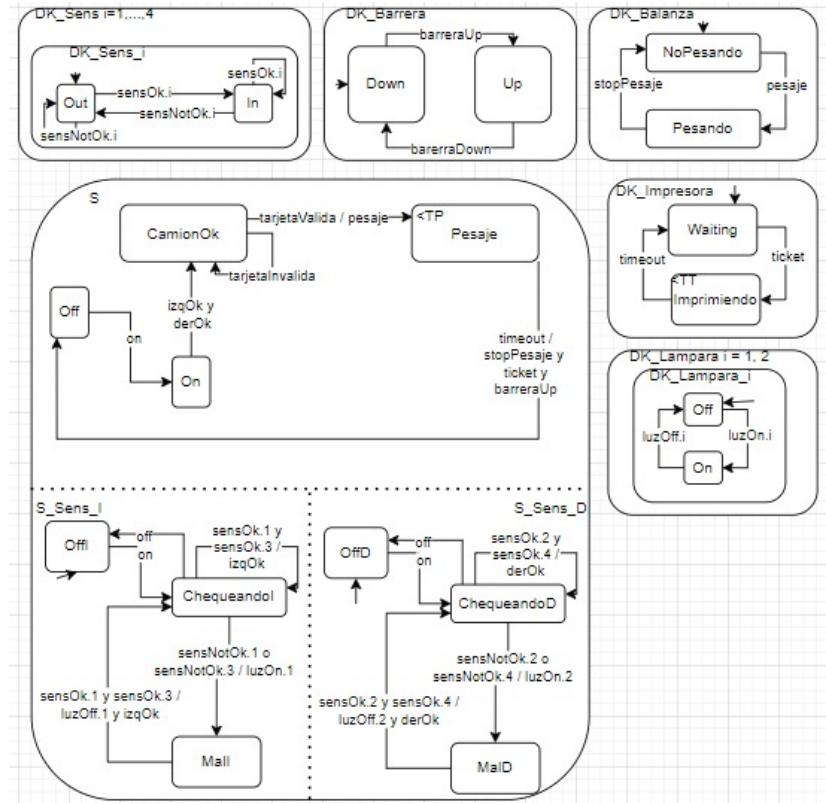
Una empresa posee una balanza para pesar camiones cargados con materia prima. El camión debe ubicarse más o menos sobre el centro de la balanza para que la pesada sea correcta. Con este fin la empresa instaló cuatro sensores en los vértices de un rectángulo imaginario de forma tal que cuando detectan que el camión está dentro de ese rectángulo, se debe bajar una barrera detrás del camión. Si el camión rebasa alguno de los laterales del rectángulo se enciende una (de dos) luz ubicada delante del camión que indica qué lado está rebasado.

Una vez que el camión está correctamente ubicado y se bajaron las barreras, el chofer debe deslizar una tarjeta magnética que lo identifica. Si la tarjeta es válida, se activa la balanza. Cuando el pesaje finaliza, se debe imprimir un tique con los datos del conductor y el peso. Luego se levantan las barreras.

Modele en Statecharts el conocimiento de dominio y la especificación de los requerimientos anteriores. Abstraiga adecuadamente la validación de la tarjeta y la impresión del tique con sus datos.

- Ingresa un camion y se inicia el sistema \approx on. EC, S
- El sistema enciende la luz de la lampara l \approx luzOn(l). MC, S
- El sistema apaga la luz de la lampara l \approx luzOff(l). MC, S
- El sistema activa la balanza y pesa al camion \approx pesaje. MC, S
- El sistema desactiva la balanza y guarda el peso del camion \approx stopPesaje. MC, S
- TP el tiempo que tarda el pesaje
- El chofer ingresa una tarjeta valida \approx tarjetaValida. EC, S
- El chofer ingresa una tarjeta invalida \approx tarjetaInvalida. EC, S
- El sistema manda a imprimir el ticket \approx ticker. MC, S

- TT el tiempo que tarda en imprimirse el ticket.
- El sistema levanta las barreras \approx barrerasUp. MC, S
- El sistema baja las barreras \approx barrerasDown. MC, S
- El sensor s indica que el camion esta dentro de la balanza \approx sensOk(s). EC, S
- El sensor s indica que el camion esta fuera de la balanza \approx sensNotOk(s). EC, S



2.8. Ejercicio 8

Tomado en: 2010.06.04.

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/dismine 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. Además, si la habitación permanece vacía por más de 12 horas se debe encender el sistema de ventilación por 1 hora.

Modele en Statecharts el conocimiento de dominio y la especificación de los siguientes requerimientos.

2.9. Ejercicio 9

Un edificio contará con varios ascensores controlados desde un programa. No puede haber más de dos ascensores moviéndose al mismo tiempo. Las puertas de los ascensores solo deben abrirse cuando estos están detenidos en un piso. Cada ascensor puede ser llamado desde cada piso mediante un botón. Dentro de cada ascensor hay una botonera para dirigirlo hacia el piso al que se desee ir. En cada piso y en cada túnel hay un sensor que avisa el paso del ascensor por ese piso. Las puertas son manuales.

- a) Describa en Statecharts el conocimiento de dominio y la especificación del sistema de ascensores que se enuncia arriba, para un edificio de p pisos y n ascensores. No tener en cuenta los botones de los pisos. Designe los fenómenos de interés.
- b) Modele en CSP el conocimiento de dominio y la especificación del problema.

2.10. Ejercicio 10

Una máquina expendedora de bebidas funciona de la siguiente manera. La máquina está inactiva hasta que se libera una traba de seguridad. Los clientes pueden seleccionar una bebida pulsando el botón correspondiente. Hay cinco bebidas diferentes. Cuando la máquina no está inactiva se pueden insertar monedas mientras se muestra en una pequeña pantalla el total hasta el momento. (Se aceptan monedas de 25 y 50 centavos y de 1 peso. La máquina detecta el valor de la moneda.) Si se selecciona una bebida y el total de dinero es igual al precio del producto, entonces la máquina entrega la bebida correspondiente. Cada bebida

16

puede tener un precio diferente. Si el total es mayor que el precio del producto, se entrega la bebida y la pantalla se actualiza poniendo como nuevo total la diferencia. La máquina no puede ponerse en modo inactivo si el total mostrado no es cero. El usuario puede pulsar un botón para que la máquina le retorne el total mostrado en pantalla, en cuyo caso el total mostrado se pone a cero.

- a) Modele en Statecharts el conocimiento de dominio y la especificación de los requerimientos anteriores. Aunque no es necesario mostrar el total de dinero ingresado por el usuario, el modelo debe contemplar una abstracción adecuada. También abstraiga adecuadamente el tema de los precios de las bebidas.
- b) Modele en CSP el conocimiento de dominio y la especificación de los requerimientos enunciados en el problema.

2.11. Ejercicio 12

Un software controlará el funcionamiento de un minicomponente musical. El equipo consta de reproductor de CD y radio AM/FM. El panel de control cuenta con la interfaz de usuario habitual: encendido/apagado, “play”, “prev”, “next”, “stop”, “pause”, “eject” y selector CD-radio. Los botones como “play”, “prev”, etc. sirven para todos los componentes dependiendo del selector (por ejemplo, “prev” sirve para mover el sintonizador de la radio hacia las estaciones más bajas). Si el CD está seleccionado y se pulsa de forma continua por 2 segundos o más cualquiera de los botones “prev” o “next”, entonces se debe avanzar dentro de la misma canción y no saltar de canción. En la misma situación si se pulsan dos veces con diferencia de menos de 1 décima de segundo, entonces se pasa a la última o primera pista. Además, hay un sensor que detecta la presencia de un CD. Si se pulsa el botón de apagado mientras el CD está en funcionamiento primero se debe apagar el CD y luego el equipo. El minicomponente cuenta con una memoria que permite que al re-encenderse se active el último componente usado, en particular si es la radio se encenderá AM o FM.

Las órdenes recibidas desde los botones, si el CD está activado, se deben traducir en órdenes al motor (giro normal o giro rápido hacia atrás o adelante, parar, arrancar, expulsar) o el lector láser (encender, apagar, leer).

Las órdenes recibidas desde los botones, si la radio está activada, se deben traducir en órdenes a un sintonizador (aumentar/disminuir frecuencia y cambiar rango de frecuencia).

- a) Liste las designaciones y confeccione la tabla de control y visibilidad para los fenómenos de interés de los requerimientos anteriores.
- b) Describa en CSP un modelo para el conocimiento de dominio del sistema de control.
- c) Describa en CSP un modelo para el sistema de control.
- d) Describa en Statecharts un modelo para el conocimiento de dominio del sistema de control.
- e) Describa en Statecharts un modelo para el sistema de control.

2.12. Ejercicio 14

Una cinta transportadora acarrea ítems de dos tipos, A y B , que luego son capturados por las máquinas M_A y M_B , respectivamente. Un software controla la cinta y ambas máquinas.

El software debe encender cada una de las máquinas M_A y M_B , mediante los eventos *turnona* y *turnonb*, respectivamente. Cada máquina necesita r unidades de tiempo para estar operativa.

Cada vez que en la cinta se detecta un ítem de un tipo, el software debe emitir el evento *takea* o *takeb* para que la máquina correspondiente capture el ítem en cuestión, a menos que a continuación se indique otra cosa.

Si entre la aparición de dos o más ítems transcurren menos de n unidades de tiempo, no se debe emitir la orden de captura correspondiente a los ítems posteriores al primero; se deberá emitir las órdenes de captura cuando aparezca un ítem n unidades de tiempo después del anterior.

Si transcurren más de t unidades de tiempo entre dos ítems, se debe apagar todo el sistema mediante el evento *abort*.

Especifique en Statecharts el conocimiento de dominio y la especificación de los requerimientos que se enuncian arriba.

2.13. Ejercicio 15

Protocolo CSMA/CD. Para transmitir datos entre terminales de trabajo conectadas en red se debe hacer uso de algún protocolo. En algunas redes *broadcast* con un único bus la clave está en cómo asignar el uso de este cuando varias terminales compiten por él.

Uno de los protocolos que resuelven esta cuestión es el *Carrier Sense, Multiple Access with Collision Detection*, o simplemente CSMA/CD. Una breve descripción del funcionamiento de este protocolo es la siguiente.

Una terminal transmite al sistema (es decir a la implementación del protocolo) un mensaje que debe ser enviado por la red. El sistema, si el bus está disponible (esto es, no hay otra terminal transmitiendo), comienza a enviar su mensaje. Sin embargo, si detecta que el bus está ocupado, espera un tiempo aleatorio y vuelve a intentar transmitir el mensaje. Esto lo hará tantas veces como sea necesario hasta que pueda empezar a transmitir.

Aun tomando estas precauciones puede ocurrir que dos terminales usen el bus al mismo tiempo, lo que da lugar a una *colisión*. Cuando una colisión ocurre el bus comunica esta situación a todas las terminales. Esto implica que todas las terminales abortan inmediatamente las transmisiones y, nuevamente, esperan un tiempo aleatorio para empezar a transmitir de nuevo.

Los mensajes que colisionan se pierden. Una vez que el mensaje ha sido transmitido, la terminal que inició la transmisión es notificada.

Se supone que todos los mensajes (sin importar tamaño) demoran exactamente λ unidades de tiempo en ser transmitidos.

En resumen, en cada terminal corre una implementación del protocolo y todas las terminales comparten el bus. La interfaz que provee el bus consta de: determinar si el bus está libre o no, enviar un mensaje, comunicar que un mensaje se transmitió, comunicar a todas las terminales que hay colisión.

Modele la especificación del protocolo CSMA/CD que se describe arriba en:

18



- a) Statecharts
- b) CSP

3. Práctica CSP

$TIMER = start?n \rightarrow (TIMER'(n)[n > 0] \parallel TIMER) \vee STOP \rightarrow TIMER$

$TIMER'(0) = \overline{timeout} \rightarrow TIMER$

$TIMER'(n + 1) = tick \rightarrow TIMER'(n)$

$DKRT = \overline{tick} \rightarrow DKRT$

$RTR = DKRT \parallel TIMER$

$RTR(n) = DKRT \parallel H[\parallel_{i=1}^n i : TIMER]$

donde $H = \{x.tick \rightarrow tick \mid x \in [1, n]\}$

3.1. Ejercicio 1

Tomado en: 2009.08.07(r2).

Un sistema debe controlar un aparato para efectuar electroencefalogramas simples. El análisis consiste en estudiar el voltaje que emiten 10 electrodos que permiten conocer la actividad bioeléctrica cerebral (cada uno comunica un valor al sistema). Es necesario tomar 5 muestras por segundo, espaciadas uniformemente. En cada una de las 5 muestras se lee el valor de los 10 electrodos. Notar que si el cerebro del paciente no presenta actividad en las cercanías de un electrodo, este no emitirá señal alguna. Por lo tanto, el sistema no puede esperar indefinidamente por la señal del electrodo. Finalmente, el sistema debe enviar secuencialmente a una impresora el valor obtenido en cada electrodo (que haya retornado uno o nada en caso de que no se haya registrado ninguno).

- a) Modele en Statecharts la especificación y el conocimiento del dominio del problema anterior.
- b) Escriba las designaciones y modele en CSP el conocimiento de dominio y la especificación de los requerimientos enunciados en el problema. Para las cuestiones temporales puede asumir la existencia de un temporizador como el que se mostró en clase.

- El electrodo e recibe su muestra por el canal elec \approx elec(e). EC, U
- El electrodo comunica la muestra al sistema por el canal send \approx send(e). EC, S
- El sistema envia a la impresora el valor del electrodo e por el canal print \approx print(e). MC, S
- El sistema indica que recibio todas las muestras \approx muestras. MC, S
- El sistema indica que recibio todos los valores de una muestra \approx valores. MC, S
- Se espera como maximo 1 segundo para tomar las 5 muestras de los 10 electrodos.
- $T < 1/5$ es el tiempo que el sistema espera el valor de todos los electrodos por muestra.

$DK_ELECTRODO = elec?v \rightarrow send!v \rightarrow DK_ELECTRODO$

$DK_IMPRESORA = print?v \rightarrow DK_IMPRESORA$

$S = 1.start!1 \rightarrow 2.start!T \rightarrow COUNTER(5, 10) \vee (muestras \rightarrow \overline{1.stop} \rightarrow S \mid 1.timeout \rightarrow S)$
 $\parallel RTR(2)$

$COUNTER(0, _) = \overline{2.stop} \rightarrow \overline{muestras} \rightarrow SKIP$

$COUNTER(n + 1, 0) = \overline{2.stop} \rightarrow 2.start!T \rightarrow COUNTER(n, 10)$

$COUNTER(n + 1, m + 1) = \square_{i=1}^{10} i.send?v \rightarrow print!v \rightarrow COUNTER(n + 1, m)$
 $\mid 2.timeout \rightarrow 2.start!T \rightarrow COUNTER(n, 10)$

$SYSTEM = S \parallel DK_IMPRESORA \parallel H[\parallel_{i=1}^{10} i : DK_ELECTRODO]$

3.2. Ejercicio 2

Ejercicio 2 Tomado en: 2010.06.15(p2).

Un sistema distribuido consta de N computadoras cada una de las cuales ejecuta el mismo programa; estas computadoras pueden estar encendidas o apagadas. Este programa recibe dos eventos desde el exterior (en cada computadora), a y b . Cuando el programa en una computadora recibe a , debe comunicarles a todas las otras computadoras, por medio de un evento interno, que lo ha recibido. Luego de comunicar esta situación esa computadora debe esperar a que todas le respondan. Cada computadora encendida responde con un evento diferente pero solo si está en su estado inicial. Como puede haber computadoras apagadas o que están haciendo otra cosa, el programa esperará un tiempo B cada respuesta. Cuando recibe todas las respuestas o han transcurrido B unidades de tiempo, debe emitir el evento c y volver al estado inicial. Si el programa recibe el evento b debe esperar un tiempo t hasta poder recibir el evento a a menos que se dé otro evento b luego de T unidades de tiempo, en cuyo caso debe reiniciar el tiempo de espera.

- a) Designe los términos más importantes del enunciado.
- b) Describa en Statecharts la especificación del programa mencionado en el problema.
- c) Describa en CSP la especificación del programa mencionado en el problema.

- La computadora c recibe el evento $a \approx a(c)$. EC, S
- La computadora c recibe el evento $b \approx b(c)$. EC, S
- La computadora c comunica que recibió el evento $a \approx \text{intA}(c)$. MC, S
- La computadora c comunica que recibió el evento $\text{intA} \approx \text{ack}(c)$. MC, S
- La computadora c emite el evento $c \approx \text{intC}(c)$. MC, S
- Se enciende la computadora $c \approx \text{on}(c)$. EC, S
- Se apaga la computadora $c \approx \text{off}(c)$. EC, S
- B es el tiempo que espera una computadora a que todas le respondan
- T es el tiempo que espera la computadora cuando recibe el evento b , para recibir otros eventos
- N es la cantidad de computadoras totales (apagadas y encendidas)
- start es el canal por donde el proceso P se comunica con el proceso TIMER
- startC es el canal por donde el proceso P se comunica con el proceso COUNTER

$$DK_COMPUTER = \text{on} \rightarrow \text{off} \rightarrow DK_COMPUTER$$

$$COUNTER = \text{startC}?n \rightarrow C(n-1)$$

$$C(0) = \overline{\text{allresponses}} \rightarrow COUNTER$$

$$C(n+1) = H[\Box_{i=1}^N i.\text{ack}] \rightarrow C(n)$$

$$S_COMPUTER = \text{on} \rightarrow WAITING \vee \text{off} \rightarrow S_COMPUTER$$

$$WAITING = a \rightarrow A \mid b \rightarrow B \mid H[\Box_{i=1}^N i.\text{intA}] \rightarrow \text{ack} \rightarrow WAITING$$

$$A = \overline{\text{intA}} \rightarrow WAITRESPONSES$$

$$WAITRESPONSES = \text{start}!B \rightarrow \text{startC}!N \rightarrow (\text{timeout} \rightarrow \overline{\text{intC}} \rightarrow WAITING \mid \text{allresponses} \rightarrow \overline{\text{intC}} \rightarrow WAITING)$$

$$\parallel RTR \parallel COUNTER$$

$$B = \text{start}!T \rightarrow (\text{timeout} \rightarrow WAITING \mid b \rightarrow STOP \rightarrow B)$$

$$SYSTEM = \parallel_{i=1}^N (S_COMPUTER \parallel DK_COMPUTER)$$

$$H = \{j.i.\text{ack} \rightarrow i.\text{ack}, j.i.\text{intA} \rightarrow i.\text{intA} \mid i, j \in [1, N]\}$$

3.3. Ejercicio 4

Tomado en: 2015.06.17(r2).

Se cuenta con un proceso S que envía mensajes, un medio E que los transmite y un proceso R que los recibe. E puede almacenar solo un mensaje a la vez, puede corromper a lo sumo uno de cada tres mensajes consecutivos y, por simplicidad, se supone que cada mensaje es un 0 o un 1; E está fuera del control del sistema, es decir es parte del entorno. Se espera que S y R colaboren para evitar la corrupción producida por E . Para ello utilizarán un mecanismo simple: S enviará cada mensaje por triplicado y R decidirá cuál es el válido de acuerdo a una elección por mayoría simple.

Especifique en Statecharts los procesos S , E y R descriptos arriba.

- Se quiere enviar un 1 \approx send1. EC, S
- Se quiere enviar un 0 \approx send0. EC, S
- S envia 1 \approx s1. MC, S
- S envia 0 \approx s0. MC, S
- E envia un 1 \approx e1. EC, S
- E envia un 0 \approx e0. EC, S
- R decide que 1 es el valor valido \approx valid1. MC, S
- R decide que 0 es el valor valido \approx valid0. MC, S
- El sistema conto 3 mensajes \approx done. MC, S
- startC es el canal por donde se comunica el proceso P con el contador de mensajes

$$S = \text{send1} \rightarrow S1(2) \mid \text{send0} \rightarrow S0(2)$$

$$S1(0) = \overline{s1} \rightarrow S$$

$$S1(n+1) = \overline{s1} \rightarrow S1(n)$$

$$S0(0) = \overline{s0} \rightarrow S$$

$$S0(n+1) = \overline{s0} \rightarrow S0(n)$$

$$E = (s1 \rightarrow \overline{e1} \rightarrow E \mid s0 \rightarrow \overline{e0} \rightarrow E)$$

$$R = (\text{startC}!3 \rightarrow (R1(2) \parallel R0(2)) \nabla \text{done} \rightarrow R) \parallel C$$

$$R1(0) = \overline{\text{valid1}} \rightarrow \text{SKIP}$$

$$R1(n+1) = 1 \rightarrow R1(n)$$

$$R0(0) = \overline{\text{valid0}} \rightarrow \text{SKIP}$$

$$R0(n+1) = 0 \rightarrow R0(n)$$

$$C = \text{startC}?n \rightarrow C(n)$$

$$C(0) = \overline{\text{done}} \rightarrow C$$

$$C(n+1) = (e0 \rightarrow C(n)) \mid (e1 \rightarrow C(n))$$

3.4. Ejercicio 5

Tomado en: 2015.06.17(r2).

Se requiere el software de control para un robot industrial con las siguientes características. El robot puede moverse hacia la izquierda o derecha una cierta distancia y consta de un rociador y termómetro. El robot está ubicado al borde de una cinta transportadora que acarrea piezas metálicas. El requerimiento básico es que si el sensor detecta que la temperatura de la pieza que está en su cercanía es superior a $maxtemp$, el robot debe abrir el rociador y debe moverse en el sentido de la cinta hasta que la temperatura de la pieza sea inferior a $maxtemp$. Si el robot llega hasta su límite de desplazamiento debe cerrar el rociador. En cualquier caso debe retornar a la posición inicial.

Tener en cuenta lo siguiente:

- El termómetro envía una señal con la temperatura medida.
 - El motor del robot envía una señal indicando que se ha alcanzado el extremo derecho o izquierdo.
 - El software de control puede encender el motor en uno u otro sentido y detenerlo.
 - No se puede poner en funcionamiento el motor en el mismo paso en que se recibe una medición de temperatura.
- a) Describa en Statecharts la especificación y el conocimiento del dominio del problema que se enumera arriba. Abstraiga convenientemente las diferentes temperaturas que puede comunicar el termómetro.
- b) Describa en CSP la especificación y el conocimiento del dominio de los requerimientos enunciados en el problema.

- El sistema inicia el motor del robot hacia la izquierda \approx left. MC, S
- El sistema inicia el motor del robot hacia la derecha \approx right. MC, S
- El sistema frena el motor del robot \approx stop. MC, S
- El sistema le indica al motor que vuelva a su posición inicial \approx initP. MC, S
- El sistema abre el rociador del robot \approx open. MC, S
- El sistema cierra el rociador del robot \approx close. MC, S
- El termómetro del robot envía la temperatura al sistema \approx temp. EC, S
- MAXTEMP es la temperatura máxima permitida para una pieza.
- El motor indica al sistema que llegó a su límite de desplazamiento izquierdo \approx ilim. EC, S
- El motor indica al sistema que llegó a su límite de desplazamiento derecho \approx dlim. EC, S
- Se enciende el sistema \approx on. EC, S
- Se apaga el sistema \approx off. EC, S

$$DK_ROBOT = on \rightarrow (DK_ROBOTON || DK_ROCIADOR || DK_TERMOMETRO) \\ \nabla off \rightarrow DK_ROBOT$$

$$DK_ROBOTON = (left \rightarrow DK_ROBOTON \\ | right \rightarrow DK_ROBOTON \\ | \overline{ilim} \rightarrow DK_ROBOTON \\ | dlim \rightarrow DK_ROBOTON \\ | initP \rightarrow DK_ROBOTON \\ | stop \rightarrow DK_ROBOTON)$$

$$DK_ROCIADOR = open \rightarrow DK_ROCIADOR | close \rightarrow DK_ROCIADOR$$

$$DK_TERMOMETRO = temp!v \rightarrow DK_TERMOMETRO$$

$$S = on \rightarrow S_ON \nabla off \rightarrow S$$

$$S_ON = temp?v \rightarrow ((\overline{lleft} \rightarrow \overline{lopen} \rightarrow S_ROBOT_L \parallel \overline{rright} \rightarrow \overline{ropen} \rightarrow S_ROBOT_R); S_ON [v \geq MAXTEMP] S_ON)$$

$$S_ROBOT_L = ltemp?v \rightarrow (S_ROBOT_L[v \geq MAXTEMP] \overline{lclose} \rightarrow \overline{lstop} \rightarrow \overline{linitP} \rightarrow SKIP) \\ \nabla ilim \rightarrow \overline{lclose} \rightarrow \overline{linitP} \rightarrow SKIP$$

$$S_ROBOT_R = rtemp?v \rightarrow (S_ROBOT_R[v \geq MAXTEMP] \overline{rclose} \rightarrow \overline{rstop} \rightarrow \overline{rinitP} \rightarrow SKIP) \\ \nabla dlim \rightarrow \overline{rclose} \rightarrow \overline{rinitP} \rightarrow SKIP$$

$$L = \{left \rightarrow lleft, right \rightarrow bright, initP \rightarrow linitP, stop \rightarrow lstop, open \rightarrow lopen, close \rightarrow lclose, temp \rightarrow ltemp\}$$

$$R = \{left \rightarrow rleft, right \rightarrow rrigh, initP \rightarrow rinitP, stop \rightarrow rstop, open \rightarrow ropen, close \rightarrow rclose, temp \rightarrow rtemp\}$$

$$SYSTEM = L[DK_ROBOT] \parallel R[DK_ROBOT] \parallel S$$

3.5. Ejercicio 6

Un productor produce de a un mensaje a la vez a razón de 3 segundos cada uno. Existen otros dos productores que se comportan de la misma forma pero trabajan con tiempos de 1 y 4 segundos, respectivamente. Todos los productores tienen un error en sus relojes de $\pm\delta$ segundos con $0 < \delta < 1$.

Los mensajes enviados por los tres productores son recibidos por un componente que puede recibir de a un mensaje a la vez. El componente cuenta con un “buffer” con capacidad para MB mensajes. Poner o sacar un mensaje del “buffer” le toma 0,1 segundos.

Por otro lado, hay dos consumidores de mensajes que pueden consumir de a un mensaje a la vez cada uno, a razón de 2 segundos por mensaje. El componente comunica los mensajes según una política FIFO.

Tener en cuenta que los productores y consumidores son las entidades activas.

- a) Escriba las designaciones y la tabla de control y visibilidad del problema anterior.
- b) Modele en Statecharts el conocimiento del dominio (productores y consumidores) y la especificación del componente de almacenamiento.
- c) Modele en CSP el conocimiento del dominio (productores y consumidores) y la especificación del componente de almacenamiento.

15

- d) Explique claramente (mostrando un ejemplo si es necesario) si el modelo semántico “broadcast” de Statecharts hace que el sistema completo se comporte de forma diferente a como lo hace la especificación CSP, que responde a un modelo semántico sincrónico.

- El productor1 produce un mensaje $\approx msg1$. EC, U
- El productor 2 o 3 produce un mensaje $\approx msg23$. EC, U
- El consumidor c consume un mensaje $\approx consume(c)$. EC, S
- El buffer recibe un mensaje $m \approx newmsg(m)$. EC, S
- El buffer envia el primer mensaje $m \approx sendmsg(m)$. MC, S

$$DK_PRODUCTOR = msg1?m \rightarrow DK_PRODUCTOR1(m) \parallel msg23?m \rightarrow DK_PRODUCTOR23(m)$$

$$DK_PRODUCTOR1(m) = newmsg!m \rightarrow^3 DK_PRODUCTOR$$

$$DK_PRODUCTOR23(m) = newmsg!m \rightarrow^{1 < t < 4} DK_PRODUCTOR \text{ (DUDOSO, quizas usar } \sqcap \text{)}$$

$$DK_CONSUMIDOR = \overline{consume} \rightarrow^2 DK_CONSUMIDOR \mid receive?v \rightarrow DK_CONSUMIDOR$$

$$S_BUFFER = startB?(MB + 1) \rightarrow BUFFER(MB + 1, \langle \rangle)$$

$$BUFFER(MB + 1, \langle \rangle) = newmsg?m \rightarrow^{0,1} BUFFER(MB, \langle m \rangle)$$

$$BUFFER(MB + 1, s) = newmsg?m \rightarrow^{0,1} BUFFER(MB, \langle m \rangle \cap s)$$

$$\mid \Box_{i=1}^2 i.consume \rightarrow i.receive!head(s) \rightarrow^{0,1} BUFFER(MB + 2, tail(s))$$

$$BUFFER(0, s) = consume \rightarrow^{0,1} BUFFER(1, tail(s))$$

$$SYSTEM = startB!MB \rightarrow (S_BUFFER \parallel DK_PRODUCTOR \parallel \\ (\parallel_{i=1}^2 i : DK_CONSUMIDOR))$$

3.6. Ejercicio 9

Un edificio contará con varios ascensores controlados desde un programa. No puede haber más de dos ascensores moviéndose al mismo tiempo. Las puertas de los ascensores solo deben abrirse cuando estos están detenidos en un piso. Cada ascensor puede ser llamado desde cada piso mediante un botón. Dentro de cada ascensor hay una botonera para dirigirlo hacia el piso al que se desee ir. En cada piso y en cada túnel hay un sensor que avisa el paso del ascensor por ese piso. Las puertas son manuales.

- a) Describa en Statecharts el conocimiento de dominio y la especificación del sistema de ascensores que se enuncia arriba, para un edificio de p pisos y n ascensores. No tener en cuenta los botones de los pisos. Designe los fenómenos de interés.
- b) Modele en CSP el conocimiento de dominio y la especificación del problema.

3.7. Ejercicio 10

Una máquina expendedora de bebidas funciona de la siguiente manera. La máquina está inactiva hasta que se libera una traba de seguridad. Los clientes pueden seleccionar una bebida pulsando el botón correspondiente. Hay cinco bebidas diferentes. Cuando la máquina no está inactiva se pueden insertar monedas mientras se muestra en una pequeña pantalla el total hasta el momento. (Se aceptan monedas de 25 y 50 centavos y de 1 peso. La máquina detecta el valor de la moneda.) Si se selecciona una bebida y el total de dinero es igual al precio del producto, entonces la máquina entrega la bebida correspondiente. Cada bebida

16

puede tener un precio diferente. Si el total es mayor que el precio del producto, se entrega la bebida y la pantalla se actualiza poniendo como nuevo total la diferencia. La máquina no puede ponerse en modo inactivo si el total mostrado no es cero. El usuario puede pulsar un botón para que la máquina le retorne el total mostrado en pantalla, en cuyo caso el total mostrado se pone a cero.

- a) Modele en Statecharts el conocimiento de dominio y la especificación de los requerimientos anteriores. Aunque no es necesario mostrar el total de dinero ingresado por el usuario, el modelo debe contemplar una abstracción adecuada. También abstraiga adecuadamente el tema de los precios de las bebidas.
- b) Modele en CSP el conocimiento de dominio y la especificación de los requerimientos enunciados en el problema.

- Se activa la máquina \approx on. EC, S
- Se desactiva la máquina \approx off. EC, S
- El cliente selecciona la bebida $i \approx \text{select}(i)$. EC, S
- El cliente inserta una moneda de 25 \approx c25. EC, S
- El cliente inserta una moneda de 50 \approx c50. EC, S

- El cliente inserta una moneda de 1 peso $\approx p1$. EC, S
- El cliente indica que quiere el dinero $\approx \text{wantCash}$. EC, S
- El sistema da la bebida $i \approx \text{give}(i)$. MC, S
- El sistema da vuelto $\approx \text{giveCash}$. MC, S
- El sistema indica que al usuario no le alcanza el dinero para la bebida seleccionada $\approx \text{moneyLeft}$. MC, S
- El sistema indica a la pantalla que muestre el dinero actual $\approx \text{showMoney}$. MC, S
- $\text{price}(i)$ indica el precio de la bebida i

$$\begin{aligned} DK_MAQUINA &= on \rightarrow DK_MAQUINA_ON \\ DK_MAQUINA_ON &= giveCash \rightarrow DK_MAQUINA_ON \mid \\ &\quad \square_{i=1}^5 i.give \rightarrow DK_MAQUINA_ON \mid \\ &\quad off \rightarrow DK_MAQUINA \end{aligned}$$

$$\begin{aligned} DK_PANTALLA &= on \rightarrow DK_PANTALLA_ON \\ DK_PANTALLA_ON &= showMoney \rightarrow DK_PANTALLA_ON \mid \\ &\quad off \rightarrow DK_PANTALLA \end{aligned}$$

$$\begin{aligned} S &= on \rightarrow WAITCASH(0) \nabla off \rightarrow S \\ WAITCASH(c) &= \\ &\quad c25 \rightarrow \overline{\text{showMoney}} \rightarrow WAITCASH(c + 0,25) \\ &\quad | \quad c50 \rightarrow \overline{\text{showMoney}} \rightarrow WAITCASH(c + 0,50) \\ &\quad | \quad p1 \rightarrow \overline{\text{showMoney}} \rightarrow WAITCASH(c + 1) \\ &\quad | \quad wantCash \rightarrow giveCash \rightarrow WAITCASH(0) \\ &\quad | \quad \square_{i=1}^5 i.select \rightarrow GIVE(i, \text{price}(i), c) \nabla \text{moneyLeft} \rightarrow WAITCASH(c) \end{aligned}$$

$$GIVE(i, p, c) = (\overline{i.give} \rightarrow WAITCASH(c - p))[p \leq c](\overline{\text{moneyLeft}} \rightarrow SKIP)$$

$$SYSTEM = S \parallel DK_MAQUINA$$

3.8. Ejercicio 11

Se trata de mantener cierta temperatura dentro de un automóvil. La temperatura solo puede ser regulada ajustando el funcionamiento del acondicionador de aire (AA); el AA solo se apaga o se prende, no es posible hacer que el aire salga a mayor o menor temperatura. Entonces, cuanto más funciona el AA más se enfriá el habitáculo. El conductor puede seleccionar la temperatura que deseé mantener por medio de una perilla giratoria. La temperatura deseada puede estar entre 18 y 30 °C; si la temperatura descendea es de 18 °C, el AA no se detiene nunca. El sistema estará en funcionamiento solo si el conductor ha pulsado un botón de activación; el sistema sale de funcionamiento cuando el botón se vuelve a pulsar o el motor se detiene. Existen dos termómetros dentro del habitáculo (uno en la parte delantera y el otro en la parte trasera) que, una vez encendido el sistema, envían señales de incremento o decremento de la temperatura sensada. Los termómetros inicialmente se encuentran a 18 °C; los termómetros no envían señales si la temperatura excede su rango de funcionamiento.

Designe los fenómenos de interés, armar la tabla de visibilidad y control y modele en CSP el conocimiento de dominio y la especificación de los requerimientos anteriores.

3.9. Ejercicio 12

Un software controlará el funcionamiento de un minicomponente musical. El equipo consta de reproductor de CD y radio AM/FM. El panel de control cuenta con la interfaz de usuario habitual: encendido/apagado, “play”, “prev”, “next”, “stop”, “pause”, “eject” y selector CD-radio. Los botones como “play”, “prev”, etc. sirven para todos los componentes dependiendo del selector (por ejemplo, “prev” sirve para mover el sintonizador de la radio hacia las estaciones más bajas). Si el CD está seleccionado y se pulsa de forma continua por 2 segundos o más cualquiera de los botones “prev” o “next”, entonces se debe avanzar dentro de la misma canción y no saltar de canción. En la misma situación si se pulsan dos veces con diferencia de menos de 1 décima de segundo, entonces se pasa a la última o primera pista. Además, hay un sensor que detecta la presencia de un CD. Si se pulsa el botón de apagado mientras el CD está en funcionamiento primero se debe apagar el CD y luego el equipo. El minicomponente cuenta con una memoria que permite que al re-encenderse se active el último componente usado, en particular si es la radio se encenderá AM o FM.

Las órdenes recibidas desde los botones, si el CD está activado, se deben traducir en órdenes al motor (giro normal o giro rápido hacia atrás o adelante, parar, arrancar, expulsar) o el lector láser (encender, apagar, leer).

Las órdenes recibidas desde los botones, si la radio está activada, se deben traducir en órdenes a un sintonizador (aumentar/disminuir frecuencia y cambiar rango de frecuencia).

- a) Liste las designaciones y confeccione la tabla de control y visibilidad para los fenómenos de interés de los requerimientos anteriores.
- b) Describa en CSP un modelo para el conocimiento de dominio del sistema de control.
- c) Describa en CSP un modelo para el sistema de control.
- d) Describa en Statecharts un modelo para el conocimiento de dominio del sistema de control.
- e) Describa en Statecharts un modelo para el sistema de control.

- Se enciende el sistema \approx on. EC, S
- Se apaga el sistema \approx off. EC, S
- Se presiona el botón play \approx play. EC, S
- Se presiona el botón prev \approx prev. EC, S
- Se presiona el botón next \approx next. EC, S
- Se presiona el botón stop \approx stop. EC, S
- Se presiona el botón pause \approx pause. EC, S
- Se presiona el botón eject \approx eject. EC, S
- Se selecciona radio \approx radio. EC, S
- Se selecciona radio AM \approx am. EC, S
- Se selecciona radio FM \approx fm. EC, S
- Se selecciona CD \approx CD. EC, S
- Un sensor detecta el CS \approx hayCD. EC, S
- El CD salta de canción \approx adelante. MC, S
- El CD vuelve de canción \approx atrás. MC, S
- El CD se pausa \approx parar. MC, S
- El CD arranca \approx arrancar. MC, S
- El CD se expulsa \approx expulsar. MC, S
- El CD adelanta la canción actual \approx adelantar. MC, S
- El CD atrasa la canción actual \approx atrasar. MC, S

- El sintonizador aumenta la frecuencia \approx aumentar. MC, S
- El sintonizador disminuye la frecuencia \approx disminuir. MC, S
- El sintonizador cambia rango de frecuencia a fm \approx freqFM. MC, S
- El sintonizador cambia rango de frecuencia a am \approx freqAM. MC, S

COMPLETAR

$$DK_SISTEMA = on \rightarrow DK_SISTEMA_ON$$

$$DK_SISTEMA_ON = off \rightarrow DK_SISTEMA$$

$$DK_SINTONIZADOR = on \rightarrow DK_SINTONIZADOR_ON \vee off \rightarrow DK_SINTONIZADOR$$

$$DK_SINTONIZADOR_ON =$$

$$prev \rightarrow \overline{disminuir} \rightarrow DK_SINTONIZADOR_ON |$$

$$next \rightarrow \overline{aumentar} \rightarrow DK_SINTONIZADOR_ON |$$

$$am \rightarrow \overline{freqAM} \rightarrow DK_SINTONIZADOR_ON |$$

$$fm \rightarrow \overline{freqFM} \rightarrow DK_SINTONIZADOR_ON$$

$$S = on \rightarrow S_CONTROL \vee off \rightarrow S$$

$$S_CONTROL = radio \rightarrow RADIO | cd \rightarrow CD$$

$$RADIO = freqAM \rightarrow AM |$$

$$freqFM \rightarrow FM$$

$$cd \rightarrow CD$$

$$AM = disminuir \rightarrow AM |$$

$$aumentar \rightarrow AM |$$

$$freqFM \rightarrow FM$$

$$FM = disminuir \rightarrow FM |$$

$$aumentar \rightarrow FM |$$

$$freqAM \rightarrow AM$$

3.10. Ejercicio 13

Un proceso, B , debe almacenar elementos en dos *buffers* de la misma capacidad finita y conocida, N . Por otro lado, existen procesos (llamados productores) que envían datos a B para que este los almacene en los *buffers*, y existen procesos (llamados consumidores) que le piden a B los datos que tiene almacenados (lo que hace que los *buffers* se vayan vaciando).

Cuando un consumidor quiere un dato que B tiene, el proceso le debe indicar el *buffer* del cual lo quiere; en cambio los productores no pueden seleccionar el *buffer*.

Obviamente B no puede poner elementos en un *buffer* lleno y no puede sacar elementos de un *buffer* vacío. Lo que sí debe hacer B es balancear el uso de los *buffers*: cuando almacena datos lo debe hacer en el *buffer* más vacío y si un *buffer* se está vaciando más rápido que el otro, debe pasar elementos del último al primero.

Describa en CSP la especificación y el conocimiento de dominio de los requerimientos anteriores. Designe los términos básicos de su modelo.

- El productor p envía un valor a B \approx send(p). EC, S
- El consumidor c pide un valor del buffer 1 \approx give1(c). EC, S
- El consumidor c pide un valor del buffer 2 \approx give2(c). EC, S
- N es el tamaño máximo de los dos buffers
- T es el tiempo que esperan los consumidores para recibir un valor

$$DK_CONSUMIDOR = (start!T \rightarrow (\overline{give1} \rightarrow wait?v \rightarrow \overline{stop} \rightarrow DK_CONSUMIDOR \sqcap give2 \rightarrow wait?v \rightarrow \overline{stop} \rightarrow DK_CONSUMIDOR) \sqcap timeout \rightarrow DK_CONSUMIDOR) \parallel RTR$$

$$DK_PRODUCTOR = send!v \rightarrow DK_PRODUCTOR$$

$$B = BUFFER(\langle \rangle, \langle \rangle)$$

$$BUFFER(\langle \rangle, \langle \rangle) = \square_{i=1}^N i.send?v \rightarrow BUFFER(\langle v \rangle, \langle \rangle)$$

$$BUFFER(b1, \langle \rangle) = \square_{i=1}^N i.send?v \rightarrow BUFFER(b1, \langle v \rangle) \\ | \square_{i=1}^N i.give1 \rightarrow i.wait!head(b1) \rightarrow BALANCE(tail(b1), \langle \rangle)$$

$$BUFFER(\langle \rangle, b2) = \square_{i=1}^N i.send?v \rightarrow BUFFER(\langle v \rangle, b2) \\ | \square_{i=1}^N i.give2 \rightarrow i.wait!head(b2) \rightarrow BALANCE(\langle \rangle, tail(b2))$$

$$BUFFER(b1, b2) = \square_{i=1}^N send?v \rightarrow (CHECK2(v, b1, b2)[length(b1) > length(b2)]CHECK1(v, b1, b2)) \\ | \square_{i=1}^N i.give1 \rightarrow i.wait!head(b1) \rightarrow BALANCE(tail(b1), b2) \\ | \square_{i=1}^N i.give2 \rightarrow i.wait!head(b2) \rightarrow BALANCE(b1, tail(b2))$$

$$CHECK1(v, b1, b2) = BUFFER(b1, b2)[length(b1) = N]BALANCE(\langle v \rangle \cap b1, b2)$$

$$CHECK2(v, b1, b2) = BUFFER(b1, b2)[length(b2) = N]BALANCE(b1, \langle v \rangle \cap b2)$$

$$BALANCE(b1, b2) = BALANCE'(b1, b2)[abs(length(b1) - length(b2)) > 1]BUFFER(b1, b2)$$

$$BALANCE'(b1, b2) = BALANCE(tail(b1), head(b1) \cap b2)[length(b1) > length(b2)] \\ BALANCE(head(b2) \cap b1, tail(b2))$$

$$SYSTEM = (\parallel_{i=1}^N DK_CONSUMIDOR) \parallel (\parallel_{i=1}^N DK_PRODUCTOR) \parallel B$$

Donde

$$length : \forall X \bullet Seq(X) \rightarrow N$$

$$length(\langle \rangle) = 0$$

$$\forall x : X, s : Seq(X) \bullet length(\langle x \rangle \cap s) = 1 + length(s)$$

3.11. Ejercicio 15

Protocolo CSMA/CD. Para transmitir datos entre terminales de trabajo conectadas en red se debe hacer uso de algún protocolo. En algunas redes *broadcast* con un único bus la clave está en cómo asignar el uso de este cuando varias terminales compiten por él.

Uno de los protocolos que resuelven esta cuestión es el *Carrier Sense, Multiple Access with Collision Detection*, o simplemente CSMA/CD. Una breve descripción del funcionamiento de este protocolo es la siguiente.

Una terminal transmite al sistema (es decir a la implementación del protocolo) un mensaje que debe ser enviado por la red. El sistema, si el bus está disponible (esto es, no hay otra terminal transmitiendo), comienza a enviar su mensaje. Sin embargo, si detecta que el bus está ocupado, espera un tiempo aleatorio y vuelve a intentar transmitir el mensaje. Esto lo hará tantas veces como sea necesario hasta que pueda empezar a transmitir.

Aun tomando estas precauciones puede ocurrir que dos terminales usen el bus al mismo tiempo, lo que da lugar a una *colisión*. Cuando una colisión ocurre el bus comunica esta situación a todas las terminales. Esto implica que todas las terminales abortan inmediatamente las transmisiones y, nuevamente, esperan un tiempo aleatorio para empezar a transmitir de nuevo.

Los mensajes que colisionan se pierden. Una vez que el mensaje ha sido transmitido, la terminal que inició la transmisión es notificada.

Se supone que todos los mensajes (sin importar tamaño) demoran exactamente λ unidades de tiempo en ser transmitidos.

En resumen, en cada terminal corre una implementación del protocolo y todas las terminales comparten el bus. La interfaz que provee el bus consta de: determinar si el bus está libre o no, enviar un mensaje, comunicar que un mensaje se transmitió, comunicar a todas las terminales que hay colisión.

Modele la especificación del protocolo CSMA/CD que se describe arriba en:

18

- a) Statecharts
- b) CSP

3.12. Ejercicio 16

Se trata del software que controla un teléfono celular muy simple. El teléfono cuenta con un teclado numérico, una tecla “enviar” y otra “cortar”. Además posee una bocina que puede ser encendida o apagada y una pantalla que puede mostrar una cadena de dígitos; existe una operación que borra el contenido de la pantalla. El teléfono se enciende si se pulsa la tecla “cortar” durante más de 2 segundos. Una vez encendido el teléfono espera que el usuario teclee un número o que llegue una llamada.

Los números a los cuales se puede llamar poseen una cantidad de dígitos variable. Cada vez el usuario pulsa un dígito el número correspondiente debe mostrarse en la pantalla. Una vez que el usuario pulsa “enviar” el sistema debe esperar a que se pulse “cortar” (no se modela la comunicación en sí). Mientras tanto no se pueden recibir llamadas, es decir se pierden. Cuando se pulsa “cortar” se debe borrar la pantalla.

Si el teléfono puede recibir una llamada y esto ocurre, entonces el sistema debe hacer sonar la bocina de manera intermitente hasta que el usuario pulse “enviar” o “cortar”. La intermitencia es: 1 segundo de ruido seguido de un segundo de silencio. Si el usuario pulsa “cortar” el teléfono podrá recibir o hacer nuevas llamadas. Si el usuario pulsa “enviar” no se podrán hacer ni recibir nuevas llamadas hasta que pulse “cortar”. Si el número que se recibe coincide con uno que se mantiene en la agenda del teléfono, entonces se debe mostrar en la pantalla el nombre que le corresponda.

Escriba las designaciones y modele en CSP el conocimiento de dominio y la especificación de los requerimientos que se enunciaron arriba.

Nota: puede utilizar la especificación del temporizador visto en clase.

3.13. Ejercicio 17

Tomado en: 2016.07.01 (r2).

Designar los fenómenos de interés de los siguientes requerimientos, construir la tabla de control y visibilidad y modelar en CSP el conocimiento de dominio (incluir el mecanismo de comunicación entre los sensores) y la especificación. Considerar que la máquina a construir es el software que controla a D .

El dispositivo D debe recibir datos de N sensores conectados en serie, y con esos datos realizar diversas estadísticas. El mecanismo debe ser el siguiente: D debe pedir al primer sensor que le envíe el dato que tenga disponible, este sensor, a su vez, luego de enviar el dato le "avisa" al siguiente sensor que es su turno, y así sucesivamente hasta el sensor N . D espera un cierto tiempo a que arribe el siguiente dato, tras el cual considera que no existen más sensores (el hecho de que estén en serie y que se "avisen" entre ellos hace que N sea transparente para D). Cuando D termina de recibir los valores debe enviarle el promedio de los valores recibidos a otro dispositivo, y comenzar el ciclo nuevamente en cuto caso se reinicia el arreglo de sensores de manera tal que el primero es el que posiblemente enviará el primer valor en el nuevo ciclo. En caso de no haber recibido ningún valor luego de efectuar el primer pedido (porque N es cero o porque el primer sensor se demoró más de lo esperado), debe enviarle al otro dispositivo un mensaje de error.

- El sistema indica al dispositivo que comience a recibir datos de los sensores $\approx \text{start}$. MC, S
- El sistema se enciende $\approx \text{on}$. EC, S
- El sistema se apaga $\approx \text{off}$. EC, S
- El dispositivo le pide al sensor i que se active $\approx \text{sens}(i)$. MC, U
- El sensor i le envía al dispositivo su dato $\approx \text{send}(i)$. MC, S
- T es el tiempo que espera el dispositivo el siguiente dato.
- El dispositivo envía el promedio de los datos al otro dispositivo por el canal $\text{sendP} \approx \overline{\text{send}}$. MC, S
- El dispositivo envía error al otro dispositivo $\approx \text{error}$. MC, S

COMPLETAR

$$DK_SENS = (||_{i=1}^N i : \text{sens}?j) \rightarrow \overline{\text{send}} \rightarrow (j+1).\text{sens}!(j+1) \rightarrow DK_SENS$$

$$\begin{aligned} S &= \text{on} \rightarrow S_ON \nabla \text{off} \rightarrow S \\ S_ON &= \text{start} \rightarrow \text{COUNTER}(1) \end{aligned}$$

$$\begin{aligned} \text{COUNTER}(n) &= \text{start}!T \rightarrow n.\text{sens}!n \rightarrow \\ &\quad (\text{timeout} \rightarrow (\overline{\text{sendP}} \rightarrow S_ON[n > 1 \vee n == N] \overline{\text{error}} \rightarrow S_ON) \mid \\ &\quad \text{send} \rightarrow \text{COUNTER}(n+1)) \\ &\quad \parallel RTR \end{aligned}$$