Ingeniera del Software 2

Taller #5 – Verificación de Sistemas Concurrentes

Lecciones: Lección #12, #13, #14, #15.
DEADLINE: Jueves 28 de octubre a las 23:59 hs.
Fecha de Reentrega: Jueves 25 de noviembre a las 23:59 hs.

Formato de Entrega

El taller debe ser entregado a través del campus en la fecha de entrega indicada en el documento. Entregar un archivo **pdf** sólo con las respuestas para las preguntas de los ejercicios entregables.

IMPORTANTE: No todos los ejercicios de este documento son para entregar. Muchos son de entrada en calor para que puedan resolver los ejercicios entregables. También sirven como práctica para el segundo parcial.

1. Análisis de Bisimilaridad

La herramienta MTSA puede ser utilizada para analizar bisimulación de dos procesos P y Q mediante el siguiente procedimiento:

- Introduzca un proceso nuevo SYS que compone P y Q.
- Seleccione la opción "Compile" (o haga click en el boton "C").
- En la ventana que aparecerá, seleccione P y Q.
- En "semantics" elija Strong o Weak según sea el caso.
- Haga click en "Check".

La herramienta reportará una relación de bisimulación en el caso de que exista.

El uso de la herramienta es para que puedan corregirse a sí mismos. Se espera que puedan responder preguntas de bisimulación sin la herramienta.

Ejercicio 1

Muestre, proveyendo una relación de bisimulación que P∼Q

```
P = (a -> (b -> c  -> P |
           b -> c -> P)
    ).

Q = (a -> b -> R |
     a -> b -> R
    ),
R = (c -> Q).
```

Ejercicio 2

(Entregable)

Muestre que P $\approx$ Q.

```
PAux = (a -> (b -> STOP |
           t -> c -> STOP)
    ).
QAux = ( a -> (b -> STOP |
           t -> c -> STOP)
    | a -> c -> STOP
    ).

||Q = QAux\{t}.
||P = PAux\{t}.
```

Ejercicio 3

(Entregable) Muestre que P $\not\approx$ Q explicando la estrategia ganadora del atacante en el juego de bisimulación. Recuerde que una estrategia ganadora debería contemplar cualquier jugada del otro jugador. Es posible que un árbol sea una buena forma de representar la estrategia: los nodos que representan el turno del defensor tienen multiples hijos, uno para cada posible jugada. Los turnos del atacant solo tienen un hijo, representando la jugada de la estrategia ganadora.

```
PAux = (a -> (b -> STOP |
           t -> c -> STOP)
    ).
QAux = (a -> (c -> STOP |
           t -> b -> STOP)
    ).

||Q = QAux\{t}.
||P = PAux\{t}.
```

Ejercicio 4

Escriba una demostración de que no puede existir una relación de bisimulación débil para P y Q del ejercicio anterior. Puede usar la notación P_i para referirse al LTS resultante de cambiar el estado inicial del LTS P al estado i. Suponga la existencia de una relación de bisimulación débil que contenga (P_0, Q_0) y llegue a una contradicción. Llegue a la contradicción utilizando la misma lógica que utilizó para construir la estrategia ganadora.

Ejercicio 5

(Entregable) Utilice MTSA para verificar si el proceso PRIMES(4) con un ocultamiento adecuado es débilmente bisimilar a la especificación de su comportamiento observable dado por :

```
PRIMES_SPEC = Q0,
Q0        = (filter[0].prime[2] -> Q1),
Q1        = (filter[1].prime[3] -> Q2),
Q2        = (filter[2].prime[5] -> Q3),
Q3        = (filter[3].prime[7] -> Q4),
Q4        = (end -> Q0) + {filter[0..3].prime[2..15]}.
```

Explique cómo hizo y qué concluyó.

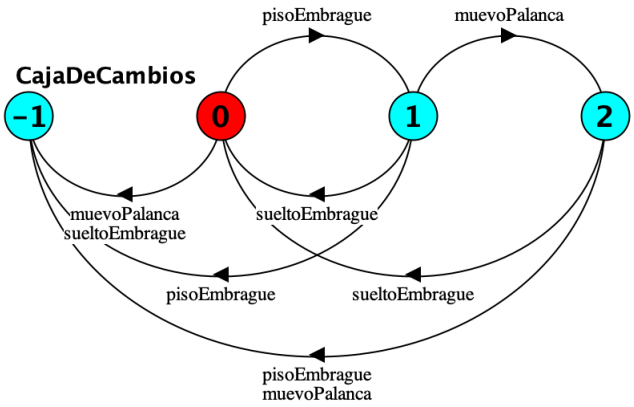
Ejercicio 6

De un contrajemplo a la siguiente afirmación: Dos LTS P y Q que tienen las mismas trazas son fuertemente bisimilares.

2. Propiedades de Safety con Observadores

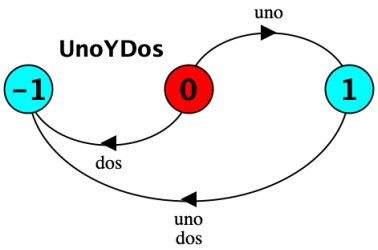
Ejercicio 7

Para el siguiente LTS, dar una expresión en FSP (lo más compacta posible, usando la palabra reservada "property") cuya semántica es equivalente.



Ejercicio 8

Para el siguiente LTS, dar una expresión en FSP (lo más compacta posible, usando la palabra reservada "property") cuya semántica es equivalente.



Ejercicio 9

(Entregable) Considere el ejercicio del Museo del taller anterior. Imagínese que está haciendo test driven development y aún no realizó el modelo del museo, y que debe hacer observadores que testeen exáctamente una sola propiedad del modelo a contruir.

1. Escriba un observador que lleve a un estado de error sólo si es posible entrar cuando el museo está cerrado. Utilice la palabra clave *property*. Importante: use el alfabeto más chico posible y haga que el observador permita el mayor número de trazas posible que no violen la propiedad. El observador no debería dar error si el modelo violara otras propiedades deseables del museo.

2. Escriba un observador que lleve a un estado de error si es posible salir del museo cuando está cerrado. Nuevamente utilice la palabra clave *property* y el alfabeto más chico posible.

3. Usando la función de menú "Check", "Safety" compruebe que el Museo satisface la propiedad "No se puede entrar cuando el museo está cerrado" pero viola la propiedad "No se puede salir cuando el museo está cerrado." Informe el modelo de Museo que utilizó y el contraejemplo reportado por la herramienta.

Ejercicio 10

(Entregable) La elección de un líder en un sistema distribuido un proceso mediante el cual se designa a un nodo de la red (o más bien a un proceso ejecutando en un nodo de la red) como el organizador de alguna tarea que se realizará entre varias computadoras.

La elección comienza con todos los nodos desconociendo quién es el líder y termina con todos identificando al mismo nodo como líder.

A continuación se presenta un modelo de un procesos de elección de líder distribuido para un conjunto de nodos en una red sincrónica con forma de anillo. Se trata del algoritmo "Lehann-Chang Robots" (LCR). Notar que red sincrónica hacer referencia a que las comunicaciones avanzan de a pasos coordinados (todos pueden mandar hasta un paquete a algún nodo en cada ronda).

Cada nodo envía su identificador único (UID) a su vecino y recibe un UID de su vecino. Si el UID recibido es menor que el propio lo descarta. Si es mayor al propio entonces lo reenvía a su vecino en la próxima ronda. Si el UID recibido es igual al propio se declara como líder.

Notar que el FSP que se muestra a continuación introduce sintaxis nueva. Se utilza:

- la definición de un conjunto (ej. `UID`),
- la dimensión del conjunto (ej. `#UID` = 4),
- indexación de eventos usando conjuntos (ej. `M` contiene los eventos `null`, `msg[19]`,`msg[14]`,`msg[7]`,`msg[2]`),
- utiliza un `if-then-else`,
- se utiliza el apóstrofo (`'`) para indicar que lo que sigue es un literal y no una variable (ej, `DSTATE[status]['null]` hace referencia a la variable `status` con rango `Rango` y `'null` hace referencia al evento `null` y no a una variable `null` que en este caso estaría indefinida).

```
//leader election in a synchronous ring
set      UID  = {[19],[14],[7],[2]}
const    N    = #UID
set      M    = {null,msg.UID}
set Status    = {unknown, leader}

//links or channels hold at most a single message
CHAN  = (put[m:M] -> get[m] -> CHAN).

//the synchronous model proceeds in two steps
ROUND = (step1 -> step2 -> ROUND).

//the processes
PROCESS(U=1)
  = (init[U]->STATE['unknown]['msg[U]]),
STATE[status:Status][send:M]
  = (step1 -> put[send] -> STATE[status][send]
    |step2 ->
      (get.null  -> DSTATE[status]['null]
      |get.msg[v:UID] ->
       if (v>U) then
         DSTATE[status]['msg[v]]
       else if (v==U) then
         DSTATE['leader]['null]
       else
                 DSTATE[status]['null]
       )
    ),
DSTATE[status:Status][send:M]
  = ([send]->[status]->STATE[status][send]).
||LCR = ( ROUND
        || chan[1..N]:CHAN
        || proc[i:1..N]:PROCESS(@(UID,i-1))
        )/{
        forall[i:1..N] {
           proc[i].get/chan[i].get,
           proc[i].put/chan[i%N+1].put,
           step1/proc[i].step1,
           step2/proc[i].step2
           }
        }.
```

Verificar, con un observador y la palabra reservada "property"que una vez declarado un líder, ningún otro nodo se declara como líder. Verificar que el modelo LCR efectivamente cumple la propiedad deseada.

Ejercicio 11

Considere las siguientes definiciones:

```
A = (a -> b -> A).
property Obs = (a -> b-> Obs) + {c}.
||AObs = (A || Obs).
```

Existen trazas a error en AObs? Por qué?

3. Progreso

Ejercicio 12

(Entregable) Reconsidere el protocolo de elección presentado anteriormente. Cómo chequearía que en toda traza con fair choice algún proceso se declarar líder? Como chequearía que en toda traza un proceso particular (ej. proc[2]) se declara líder?

Ejercicio 13

(Entregable) De un contraejemplo a la siguiente afirmación: Si Q satisface una propiedad de progreso Prop entonces para todo R, (Q|| R) satisface Prop.

4. Lógica Temporal Lineal

En esta sección las preguntas se refieren a formulas de LTL que se evalúan sobre estructuras de Kripke lineales donde múltiples proposiciones pueden ser verdaderas en un estado.

Ejercicio 14

(Entregable) Imaginemos que tenemos un modelo de una aspiradora robot. El robot sale de la base, automáticamente recorre la casa, limpia y vuelve a la base cuando se quedá sin batería. Escribir en LTL los siguientes requerimientos detallando si son una propiedad de safety o de liveness:

a) El robot está en la base infinitamente.
b) Si la batería está baja, entra en modo ahorro hasta volver a la base.
c) Si el robot detecta una pared en frente suyo, gira a la izquierda hasta que el sensor no la vea más.

Utilice las siguientes variables proposicionales: girandoAIzquierda,, bateriaBaja, modoAhorro, enBase, paredDelante.

Ejercicio 15

Indique si estos pares de fórmulas son equivalentes. Probar la equivalencia o dar una estructura de Kripke que sea un contraejemplo.

a)	$\Diamond(p \wedge q)$	vs	$\Diamond p \wedge \Diamond q$
b)	$\Box(p \wedge q)$	vs	$\Box p \wedge \Box q$
c)	$\Box(p \vee q)$	vs	$\Box p \vee \Box q$
d)	$(a \vee b)\ \mathbf{U}\ b$	vs	$a\ \mathbf{U}\ b$
e)	$(a\ \mathbf{U}\ b)\ \mathbf{U}\ c$	vs	$a\ \mathbf{U}\ (b\ \mathbf{U}\ c)$

5. Lógica Temporal Lineal para LTS

En esta sección las preguntas se refieren a formulas de LTL que se evalúan sobre trazas de un LTS. En este contexto, cada traza es representada por una estructura de kripke lineal donde cada estado tiene exactamente una proposición verdadera. Esa proposición indica qué evento acaba de suceder en la traza.

Ejercicio 16

Explique por qué no existen trazas en las que fórmula $\Box(a \wedge b)$ sea satisfecha.

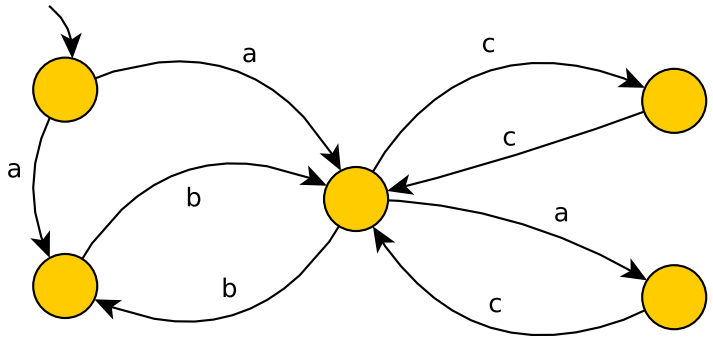
Ejercicio 17

(Entregable) Reescriba las propiedades del ejercicio anterior con variables proposicionales que hacen referencia al último evento que sucedió. Puede utilizar las siguientes proposiciones/eventos:

- entróABase: sucede cuando el robot pasa de estar fuera de la base a estar dentro de la base.
- salióDeBase: sucede cuando el robot pasa de estar dentro de la base a fuera de la base.
- bateríaBaja: sucede en cuanto la batería pasa por debajo del umbral del 10 % de carga.
- modoAhorroOff: sucede cuando se sale del modo ahorro.
- modoAhorroOn: sucede cuando se entra en el modo ahorro.
- paredDelanteDetectado: sucede cuando el sensor reporta una pared por delante.
- gira1GradoIzquierda: emisión del comando que gira el robot 1 grado a la izquierda.

Ejercicio 18

Dado el siguiente LTS con alfabeto $\{a, b, c\}$.



¿Cuáles de las siguientes fórmulas son válidas en el LTS? De contraejemplos cuando corresponda.

Verifique sus respuestas modelando el LTS en MTSA y usando la funcionalidad de verificación de LTL que provee la herramienta.

1. $\Box(a \vee b \vee c)$
2. $\Diamond\Box c$
3. $\Box a$
4. $a \ \mathbf{U} \ (b \vee c)$
5. $\Box(a) \vee \Box(b) \vee \Box(c)$
6. $\mathbf{X}\ (\neg c) \rightarrow \mathbf{X}\ (\ \mathbf{X}\ (c))$

Ejercicio 19

Retomando el modelo de pipes y filters para el cálculo de números primos, escriba una propiedad que asevere que el filtro 2 eventualmente declara el número 5 como primo y nunca ningún otro número. Escriba otra que dice que el filto 2 declara el número 4 como primo.

Utilice MTSA para verificar sus respuesta.

Ejercicio 20

(Entregable)

Retomando el modelo del museo, reescriba las dos propiedades escritas con observadores usando LTL.

Utilice MTSA para verificar su respuesta.

Para estas propiedades, ¿qué resultó más compacto y claro, usar LTL u observadores?