Examen: Relaxed Concurrency

Deadline: 18 de Agosto, 2017

Examen de la materia Relaxed Concurrency in the 21st century: It's Weak! Para la corrección, incluya en un archivo comprimido un scan o foto de las respuestas a las preguntas del ejercicio 1, y el código del ejercicio 2. Incluya su nombre en cada pagina y en cada archivo fuente. Envíe el archivo adjunto a gpetri@irif.fr a más tardar el 18 de Agosto. Incluya en el encabezado del email su nombre completo.

Ejercicio 1

Para cada uno de los siguientes ejemplos, decir si el comportamiento en cuestión es posible en los modelos: SC, TSO, PSO y PowerPC. Se recomienda utilizar la herramienta `herd` con las formalizaciones de TSO[1] y PowerPC[2]. Para el modelo de PSO se recomienda modificar la formalización de TSO.

[1] `http://diy.inria.fr/doc/tso-02.cat`

[2] `https://github.com/herd/herdtools7/blob/master/herd/libdir/ppc.cat`

1) Dekker (Store Buffering)

$x = 0 \ \wedge \ y = 0$		
$x = 1;$	$\|$	$y = 1;$
$r_1 = y;$		$r_2 = x;$
$r_1 = 0 \ \wedge \ r_2 = 0$		

2) Load Buffering[3]

$x = 0 \ \wedge \ y = 0$		
$r_1 = y;$	$\|$	$r_2 = x;$
$x = 1;$		$y = 1;$
$r_1 = 1 \ \wedge \ r_2 = 1$		

3) Thin Air Reads

$x = 0 \ \wedge \ y = 0$		
$r_1 = y;$	$\|$	$r_2 = x;$
$x = r_1;$		$y = r_2;$
$r_1 = 1 \ \wedge \ r_2 = 1$		

4) 1-Fence Dekker

$x = 0 \ \wedge \ y = 0$		
$x = 1;$		$y = 1;$
$\texttt{fence};$	$\|$	$r_2 = x;$
$r_1 = y;$		
$r_1 = 0 \ \wedge \ r_2 = 0$		

5) 2-Fence Dekker

$x = 0 \ \wedge \ y = 0$		
$x = 1;$		$y = 1;$
$\texttt{fence};$	$\|$	$\texttt{fence};$
$r_1 = y;$		$r_2 = x;$
$r_1 = 0 \ \wedge \ r_2 = 0$		

6) Write Read Causality[4]

$x = 0 \ \wedge \ y = 0$				
$x = 1;$	$\|$	$r_1 = x;$ $y = 1;$	$\|$	$r_2 = y;$ $r_3 = x;$
$r_1 = 1 \ \wedge \ r_2 = 1 \ \wedge \ r_3 = 0$				

7) Independent Readers Independent Writers (IRIW)[4]

$x = 0 \ \wedge \ y = 0$						
$x = 1;$	$\|$	$y = 1;$	$\|$	$r_1 = x;$ $r_2 = y;$	$\|$	$r_3 = y;$ $r_4 = x;$
$r_1 = r_3 = 1 \ \wedge \ r_2 = r_4 = 0$						

8) Sync IRIW[4]

$x = 0 \ \wedge \ y = 0$						
$x = 1;$	$\|$	$y = 1;$	$\|$	$r_1 = x;$	$\|$	$r_3 = y;$
$\texttt{sync};$		$\texttt{sync};$		$r_2 = y;$		$r_4 = x;$
$r_1 = r_3 = 1 \ \wedge \ r_2 = r_4 = 0$						

[3] Modificado el 7/8/2017.
[4] Modificado el 2/8/2017.

9) Sync IRIW 2^4

$\mathtt{x} = 0 \ \wedge \ \mathtt{y} = 0$							
			$\mathtt{r}_1 = \mathtt{x};$			$\mathtt{r}_3 = \mathtt{y};$
$\mathtt{x} = 1;$	$\|$	$\mathtt{y} = 1;$	$\mathtt{sync};$	$\|$	$\mathtt{sync};$	
			$\mathtt{r}_2 = \mathtt{y};$			$\mathtt{r}_4 = \mathtt{x};$
$\mathtt{r}_1 = \mathtt{r}_3 = 1 \ \wedge \ \mathtt{r}_2 = \mathtt{r}_4 = 0$							

Ejercicio 2

1) Escriba un programa en Java para representar el litmus test Dekker del ejercicio 1. Se recomienda utilizar un bucle que repita el test hasta encontrar una ocurrencia del comportamiento $\mathsf{r}_1 = \mathsf{r}_2 = 0$ del test.

2) Corrija el programa Java sin modificar su estructura para evitar que el resultado ocurra. Explique brevemente en un comentario cuales son las garantías del lenguaje que aseguran la corrección.

3) Escriba Dekker en C++ garantizando la ausencia de *data races*, pero sin impedir el comportamiento $\mathsf{r}_1 = \mathsf{r}_2 = 0$.

4) Corrija el programa anterior para evitar el comportamiento en C++, nuevamente sin modificar la estructura del programa. Explique brevemente su solución en los comentarios.