

# 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<sup>1</sup> y PowerPC<sup>2</sup>. Para el modelo de PSO se recomienda modificar la formalización de TSO.

---

<sup>1</sup><http://diy.inria.fr/doc/tso-02.cat>

<sup>2</sup><https://github.com/herd/herdtools7/blob/master/herd/libdir/ppc.cat>

1) Dekker (Store Buffering)

$$\frac{\frac{x = 0 \wedge y = 0}{\begin{array}{c} x = 1; \quad y = 1; \\ r_1 = y; \quad \parallel \quad r_2 = x; \end{array}}}{r_1 = 0 \wedge r_2 = 0}$$

2) Load Buffering

$$\frac{\frac{x = 0 \wedge y = 0}{\begin{array}{c} r_1 = y; \quad r_2 = x; \\ x = 1; \quad \parallel \quad y = 1; \end{array}}}{r_1 = 0 \wedge r_2 = 0}$$

3) Thin Air Reads

$$\frac{\frac{x = 0 \wedge y = 0}{\begin{array}{c} r_1 = y; \quad r_2 = x; \\ x = r_1; \quad \parallel \quad y = r_2; \end{array}}}{r_1 = 1 \wedge r_2 = 1}$$

4) 1-Fence Dekker

$$\frac{\frac{x = 0 \wedge y = 0}{\begin{array}{c} x = 1; \quad y = 1; \\ \text{fence}; \quad \parallel \quad r_2 = x; \\ r_1 = y; \end{array}}}{r_1 = 0 \wedge r_2 = 0}$$

5) 2-Fence Dekker

$$\frac{\frac{x = 0 \wedge y = 0}{\begin{array}{c} x = 1; \quad y = 1; \\ \text{fence}; \quad \parallel \quad \text{fence}; \\ r_1 = y; \quad r_2 = x; \end{array}}}{r_1 = 0 \wedge r_2 = 0}$$

6) Write Read Causality<sup>3</sup>

$$\frac{\frac{\frac{x = 0 \wedge y = 0}{\begin{array}{c} x = 1; \quad \parallel \quad r_1 = x; \quad \parallel \quad r_2 = y; \\ y = 1; \quad \parallel \quad r_3 = x; \end{array}}}{r_1 = 1 \wedge r_2 = 1 \wedge r_3 = 0}}$$

7) Independent Readers Independent Writers (IRIW)<sup>3</sup>

$$\frac{\frac{x = 0 \wedge y = 0}{\begin{array}{c} x = 1; \quad \parallel \quad y = 1; \quad \parallel \quad r_1 = x; \quad \parallel \quad r_3 = y; \\ r_2 = y; \quad \parallel \quad r_4 = x; \end{array}}}{r_1 = r_3 = 1 \wedge r_2 = r_4 = 0}$$

8) Sync IRIW<sup>3</sup>

$$\frac{\frac{x = 0 \wedge y = 0}{\begin{array}{c} x = 1; \quad \parallel \quad y = 1; \quad \parallel \quad r_1 = x; \quad \parallel \quad r_3 = y; \\ \text{sync}; \quad \parallel \quad \text{sync}; \quad \parallel \quad r_2 = y; \quad \parallel \quad r_4 = x; \end{array}}}{r_1 = r_3 = 1 \wedge r_2 = r_4 = 0}$$

---

<sup>3</sup> Modificado el 2/8/2017.

9) Sync IRIW 2<sup>3</sup>

$$\begin{array}{c}
 \hline
 x = 0 \wedge y = 0 \\
 \hline
 \begin{array}{ccccccc}
 & & & & r_1 = x; & & r_3 = y; \\
 x = 1; & \parallel & y = 1; & \parallel & \text{sync}; & \parallel & \text{sync}; \\
 & & & & r_2 = y; & & r_4 = x;
 \end{array} \\
 \hline
 r_1 = r_3 = 1 \wedge r_2 = r_4 = 0
 \end{array}$$

## 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  $r_1 = r_2 = 0$  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  $r_1 = r_2 = 0$ .
- 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.