Propuesta de Tesis de Grado de Ingeniería en Informática

Detección de Deadlocks en Rust en tiempo de compilación mediante Redes de Petri

Director: Ing. Pablo A. Deymonnaz

Alumno: Horacio Lisdero Scaffino, *(Padrón # 100.132)*
hlisdero@fi.uba.ar

Facultad de Ingeniería, Universidad de Buenos Aires

17 de febrero de 2023

Índice

1. Introducción

En el área de computación concurrente, uno de los desafíos principales es probar la correctitud de un programa concurrente. A diferencia de un programa secuencial donde para cada entrada se obtiene siempre la misma salida, en un programa concurrente la salida puede depender de cómo se intercalaron las instrucciones de los diferentes procesos o threads durante la ejecución.

La correctitud de un programa concurrente se define entonces en términos de propiedades del cómputo realizado y no en términos del resultado obtenido. En la literatura se definen dos tipos de propiedades de correctitud[1][2][3]:

- Propiedades de *safety*: Propiedades que se deben cumplir *siempre*.
- Propiedades de *liveness*: Propiedades que se deben cumplir *eventualmente*.

Dos de las propiedades de tipo *safety* deseables en un programa concurrente son:

- **Exclusión mutua**: dos procesos no deben acceder a recursos compartidos al mismo tiempo.
- **Ausencia de *deadlock***: un sistema en ejecución debe poder continuar realizando su tarea, es decir, avanzar produciendo trabajo útil.

Usualmente se utilizan primitivas de sincronización tales como mutexes, semáforos, monitores y *condition variables* para implementar el acceso coordinado de los procesos o hilos a los recursos compartidos. No obstante, el uso correcto de estas primitivas es difícil de lograr en la práctica.

Los deadlocks son uno de los grandes problemas de los sistemas distribuidos. Su aparición puede dejar sistemas enteros fuera de uso y causar graves daños a personas y equipos. Numerosas teorías y métodos se han propuesto para prevenir los deadlocks.

1.1. Motivación

2. Estado del arte / Literatura relacionada

3. Objetivos

El objetivo general del trabajo es ...
Los objetivos particulares son:

1. 1
2. 2
3. 3
    a) 3.1
    b) 3.2

4. Metodologías propuestas

5. Conjuntos de datos (dependiendo del área de la tesis)

6. Recursos informáticos (opcional)

6.1. Hardware

6.2. Software

6.2.1. Otras herramientas

7. Cronograma de trabajo

Cantidad de horas: [768-1000]

Tareas	Meses											
	1	2	3	4	5	6	7	8	9	10	11	12
Tarea A												
Tarea B												
Tarea C												
Tarea D												
Tarea E												

- Tarea A [100 horas]: (descripción)
- Tarea B [180 horas]:
- ...

Referencias

[1] Ben-Ari, M. *Principles of Concurrent and Distributed Programming*, 2nd ed. Pearson Education, 2006.

[2] Coulouris, G., Dollimore, J., Kindberg, T., and Blair, G. *Distributed Systems, Concepts and Design*, 5th ed. Pearson Education, 2012.

[3] van Steen, M., and Tanenbaum, A. S. *Distributed Systems*, 3rd ed. Pearson Education, 2017.