Generación Automática de Tests - Año 2018
Trabajo Práctico 2
Herramientas de Generación Automática de Tests

La resolución de este trabajo será grupal para alumnos de grado (no más de tres personas), e individual para alumnos de posgrado. La evaluación constará de una instancia de defensa oral individual. Todo el código fuente que debe utilizar para la realización de este trabajo se encuentra disponible en Moodle, en el archivo TP2-src.zip

Ejercicio 1. `CubiGA` es una biblioteca Java de código abierto para implementar algoritmos genéticos. Entre otras clases, podemos encontrar una representación de individuos como secuencias de bits en la clase `BitChromosome.java` (paquete `es.unileon.rnag.cubiga.chromosome`).
Utilice el código fuente provisto de `BitChromosome` y genere tests usando `EvoSuite` y `Randoop`, que maximicen cobertura y puntaje de mutación (con ambas herramientas).
Ayuda: Puede modificar el código para eliminar no determinismo, y agregar métodos para generar mejores aserciones si lo considera necesario.

Ejercicio 2. `NodeCachingLinkedList` de `Apache Collections` es una lista doblemente encadenada circular, que además almacena un caché de nodos eliminados para reutilizarlos en futuras inserciones, y así reducir el tiempo requerido para la alocación de memoria para los nodos.
Utilice `Korat` para generar objetos para la implementación provista de `NodeCachingLinkedList` (por ejemplo, listas de enteros). Desarrolle un generador para teorías `JUnit` basado en las instancias producidas para `Korat`. Provea teorías para testear los métodos `add` y `remove` de la clase con el generador implementado, hasta lograr máxima cobertura y matar todos los mutantes para `add` y `remove` y todos los invocados por estos.

Ejercicio adicional de posgrado: Averigue y explique como utilizar la herramienta de generación exhaustiva acotada `TestEra`, basada en especificaciones declarativas (en lenguaje `Alloy`), para generar entradas para testear `NodeCachingLinkedList`.

Ejercicio 3. Utilice la herramienta `Pex` para generar tests que maximicen cobertura, para los métodos `patternIndex` y `cal` de `EjerciciosPex.cs` (ver especificación en los comentarios del código), y para `add` y `remove` de la clase `ArrayList.cs`. Anote las clases con contratos e invariantes si lo considera necesario para mejorar los tests generados.