Trabajo Práctico
Verificación de Software

LCC – FCEIA – UNR

Maximiliano Cristiá

2025

- El trabajo práctico (TP) es individual.
- Se debe entregar por correo electrónico (cristia@cifasis-conicet.gov.ar) a más tardar el día que se presenten a rendir IS 2.
- Antes de empezar a resolverlo cada alumno debe seleccionar o inventar un problema, me lo debe comunicar por correo electrónico (cristia@cifasis-conicet.gov.ar) y debe esperar mi respuesta para comenzar. La idea es que yo voy a determinar si el problema es razonable o no.

  El problema que elijan o inventen no tiene por qué ser ni grande ni complejo. La especificación Z tiene que tener al menos dos operaciones y al menos un invariante de estado. Pueden elegir problemas de la práctica y de los exámenes finales pero yo voy a controlar que dos alumnos no estén resolviendo el mismo problema.
- Una vez establecido el problema el alumno deberá:
  1. Formalizar el problema en Z.
  2. Traducir la especificación Z a un programa {*log*} (setlog) que esté correctamente tipado (es decir se debe usar el *typechecker* de {*log*}). Van a aprender {*log*} al final de Ingeniería de Software 1.
     - Para ejecutar {*log*} necesitan instalar SWI-Prolog 9.3 o superior (SWI-Prolog es un intérprete de Prolog): http://www.swi-prolog.org/.
     - El sitio oficial de {*log*} es: https://www.clpset.unipr.it/setlog.Home.html.
     - Apunte de clase para aprender a traducir de Z a {*log*}: https://www.fceia.unr.edu.ar/ingsoft/tp/setlog.pdf.
     - Presentación/tutorial (en inglés) sobre {*log*}: https://www.fceia.unr.edu.ar/ingsoft/tp/abz.pdf.
     - Manual de usuario (en inglés) de {*log*}: https://www.clpset.unipr.it/SETLOG/setlog-man.pdf.
  3. Ejecutar dos simulaciones no triviales sobre el prototipo usando el entorno Next. Las simulaciones deben involucrar a todas las operaciones del programa {*log*} y deben apuntar a validar escenarios funcionales de la aplicación. Las simulaciones deben mostrar las trazas de ejecución y deben chequear los invariantes. Si en alguna simulación un invariante no se verifica *no* es un error. Pueden reportar esa situación en el informe, corregir la especificación y volver a ejecutar la simulación. Justamente la idea de hacer simulaciones es ver si la especificación está razonablemente bien antes de intentar demostrar los lemas de invarianza.

     La ejecución de simulaciones en entorno Next está explicada en el apunte de clase.

4. Usar el VCG para generar las condiciones de verificación. El VCG está explicado en el apunte de clase.
5. Descargar todas las condiciones de verificación generadas por el VCG. Esto también está explicado en el apunte de clase.
6. Generar casos de prueba a partir del programa {*log*} usando {*log*}-TTF. Se deben generar casos de prueba para todas las operaciones del programa y usando al menos dos tácticas en cada operación. La generación de casos de prueba con {*log*}-TTF está explicada en el apunte de clase.

- El TP se debe entregar en un archivo `.zip` o `.tar.gz` que deberá contener lo siguiente.
  - Un archivo LaTeX que contenga el código de la especificación Z. La especificación debe pasar el control de tipos de Z/EVES o Fuzz.
  - Un archivo `.pdf` con el informe del TP (ver más abajo). El archivo LaTeX del informe puede ser el mismo que el de la especificación.
  - El archivo con el código fuente del prototipo {*log*}.
  - El archivo generado por el VCG (es de la forma `*-vc.slog`).
  - El código {*log*} debe pasar el control de tipos de {*log*}, todas las obligaciones de prueba generadas por el VCG deben ser descargadas y debe haber un caso de prueba para cada hoja de los árboles de prueba que se generen.
- El informe del TP debe describir lo siguiente.
  1. El problema inventado o elegido.
  2. La especificación Z incluyendo las designaciones que correspondan, más breves comentarios informales que expliquen la especificación.
  3. Las trazas de ejecución de las dos simulaciones ejecutadas en entorno Next. El código de las simulaciones tiene que estar dentro de uno o dos entornos LaTeX del tipo `verbatim` y tiene que ser solo texto.
  4. Explicar cómo fue la interacción con el VCG: ¿Tuvieron que agregar hipótesis? ¿Usaron los comandos `vcgce`, `vcacg` o `findh`?
  5. Los comandos {*log*}-TTF que se usaron para generar los casos de prueba de cada operación. Los comandos tienen que estar dentro de un entorno LaTeX del tipo `verbatim` y tiene que ser solo texto.
  6. Los archivos generados por {*log*}-TTF que contienen los casos de prueba.

  En todos los casos se debe explicar qué es lo que se hace. Por ejemplo, se debe explicar coloquialmente las simulaciones, la generación de casos de prueba, etc.
- Yo voy a corregir cada TP. Una de las cosas que voy a tener en cuenta es cómo está organizado, presentado y redactado el informe (además, obviamente, de las cuestiones técnicas).

> Si los aspectos técnicos están bien pero la organización, presentación o redacción del informe (incluye, por caso, la ortografía) no me satisfacen van a tener que mejorarlo. **Esto lo van a poder hacer solo una vez, caso contrario deberán volver a rendir IS 2 (todo, final y TP).**
> **Claramente si los aspectos técnicos están mal, el TP y el examen quedan desaprobados y los deberán hacer nuevamente.**