Verificación de software

Trabajo Práctico Final - Ingeniería de Software II

Autor:

Juan Ignacio Farizano

Departamento de Ciencias de la Computación
Facultad de Ciencias Exactas, Ingeniería y Agrimensura
Universidad Nacional de Rosario
Rosario, Santa Fe, Argentina

19 de febrero de 2025

1. Enunciado del problema

Queremos especificar el funcionamiento del sistema *Mis Estudiantes* que utilizan directivos de escuelas primarias y secundarias para inscribir y reinscribir alumnos (por ej. promoción de grado, repitente, etc) en la base de datos del Ministerio de Educación de la provincia de Buenos Aires, lo descrito es similar al sistema existente pero simplificado para mantener una dificultad razonable para este trabajo práctico.

Un directivo desea inscribir o reinscribir a un alumno en su escuela. A cada alumno se le asigna una inscripción actual que consiste en un grado escolar y un estado que representa una de las tres siguientes situaciones:

- **Inscripto**: el alumno fue inscripto y es habilitado a cursar el grado registrado
- **Promovido**: el alumno cumplió los requisitos para promocionar el grado al cual se encuentra inscripto y es habilitado a ser inscripto en el grado siguiente, o a graduarse si se encontraba en el 12avo grado
- **Repitente**: el alumno no cumplió los requisitos para promocionar de grado y debe ser inscripto en el mismo grado

Se deben especificar las siguientes operaciones:

1. **Inscribir alumno**: se registra por primera vez a un alumno y se lo inscribe a primer grado
2. **Reinscribir alumno**: se inscribe un alumno previamente registrado al grado que corresponda según su último estado de inscripción
3. **Cerrar inscripción**: la inscripción actual es cerrada y se registra una promoción o repitencia
4. **Consultar graduado**: se desea consultar si un alumno se graduó

No se diferencia entre primaria y secundaria, registramos desde 1er grado hasta 12avo, el último del secundario. Los requisitos para promocionar de grado se encuentran por fuera del sistema y no deben ser tenidos en cuenta.

2. Especificación en Z

Para comenzar, damos las designaciones:

alumno es un alumno $\approx alumno \in ALUMNO$

grado es un grado correspondiente a un año escolar $\approx grado \in GRADO$

estado es un estado de inscripción $\approx estado \in ESTADO$

rep es un mensaje de respuesta $\approx rep \in REP$

Estado de inscripción actual correspondiente al alumno *alumno* $\approx inscripciones(alumno)$

Luego, definimos los tipos básicos y enumerados utilizados:

$[ALUMNO]$
$GRADO == \mathbb{N}$
$ESTADO ::= inscripto \mid promueve \mid repite$
$REP ::= alumnoEsGraduado \mid alumnoNoEsGraduado \mid alumnoNoEncontrado$

Además, damos la siguiente definición axiomática que define el último grado que debe cursar y promocionar un alumno para graduarse:

$$maximoGrado : GRADO$$
$$maximoGrado = 12$$

Definimos nuestro estado, donde *registrados* es el conjunto de alumnos que fueron registrados en el sistema, e *inscripciones* es la asignación de cada alumno a su inscripción actual, siendo esta una tupla compuesta por un grado y un estado de inscripción según lo requerido.

MisEstudiantes
$$registrados : \mathbb{P}\, ALUMNO$$
$$inscripciones : ALUMNO \pfun (GRADO \times ESTADO)$$

Definimos dos invariantes de estado. La primera indica que todos los alumnos registrados en el sistema deben tener un estado de inscripción actual y viceversa.

InscripcionesInv
$$MisEstudiantes$$
$$\mathrm{dom}\, inscripciones = registrados$$

La segunda invariante representa que para todo alumno registrado su grado de inscripción no debe superar el máximo grado posible.

MaximoGradoInv

$$MisEstudiantes$$

$$\forall\, alumno : registrados \bullet (inscripciones\ alumno).1 \leq maximoGrado$$

Definimos el estado inicial del sistema:

MisEstudiantesInicial

$$MisEstudiantes$$

$$registrados = \varnothing$$
$$inscripciones = \varnothing$$

La primera operación a especificar es *InscribirAlumno*. En el caso exitoso se debe dar la condición de que el alumno especificado no se encuentre registrado en el sistema, luego se lo registra y se los inscribe a primer grado.

InscribirAlumnoOk

$$\Delta MisEstudiantes$$
$$alumno? : ALUMNO$$

$$alumno? \notin registrados$$
$$inscripciones' = inscripciones \cup \{alumno? \mapsto (1, inscripto)\}$$
$$registrados' = registrados \cup \{alumno?\}$$

El único caso de error de esta operación se da cuando el alumno ya se encontraba registrado en el sistema.

InscribirAlumnoRegistradoE

$$\Xi MisEstudiantes$$
$$alumno? : ALUMNO$$

$$alumno? \in registrados$$

$$InscribirAlumno == InscribirAlumnoOk \lor InscribirAlumnoRegistradoE$$

La siguiente operación a modelar es *ReinscribirAlumno*, esta cuenta con dos casos exitosos. El primero se da cuando el alumno se encuentra en estado de promoción y se lo inscribe en el siguiente grado a menos que haya promovido el último grado; el segundo caso exitoso se da cuando el alumno se encuentra en estado de repitencia y se lo debe inscribir en el mismo grado que se encontraba.

ReinscribirAlumnoPromovidoOk

$\Delta MisEstudiantes$

$alumno? : ALUMNO$

$alumno? \in registrados$

$(inscripciones\ alumno?).1 < maximoGrado$

$(inscripciones\ alumno?).2 = promueve$

$inscripciones' = inscripciones \oplus \{alumno? \mapsto ((inscripciones\ alumno?).1 + 1, inscripto)\}$

$registrados' = registrados$

ReinscribirAlumnoRepitenteOk

$\Delta MisEstudiantes$

$alumno? : ALUMNO$

$alumno? \in registrados$

$(inscripciones\ alumno?).1 \leq maximoGrado$

$(inscripciones\ alumno?).2 = repite$

$inscripciones' = inscripciones \oplus \{alumno? \mapsto ((inscripciones\ alumno?).1, inscripto)\}$

$registrados' = registrados$

Los casos de error posibles son 3: estos se dan cuando el alumno no fue registrado, si ya se ha graduado o si se intenta reinscribirlo cuando ya se encontraba inscripto.

ReinscribirAlumnoNoEncontradoE

$\Xi MisEstudiantes$

$alumno? : ALUMNO$

$alumno? \notin registrados$

ReinscribirAlumnoGraduadoE

$\Xi MisEstudiantes$

$alumno? : ALUMNO$

$alumno? \in registrados$

$((inscripciones\ alumno?)).1 = maximoGrado$

$((inscripciones\ alumno?)).2 = promueve$

ReinscribirAlumnoDobleInscripE

$\Xi MisEstudiantes$

$alumno? : ALUMNO$

$alumno? \in registrados$

$(inscripciones\ alumno?).2 = inscripto$

$ReinscribirAlumnoE == ReinscribirAlumnoGraduadoE \lor ReinscribirAlumnoDobleInscripE$
$\lor ReinscribirAlumnoNoEncontradoE$

$ReinscribirAlumnoOk == ReinscribirAlumnoPromovidoOk \lor ReinscribirAlumnoRepitenteOk$

$ReinscribirAlumno == ReinscribirAlumnoOk \lor ReinscribirAlumnoE$

La tercera operación es *CerrarInscripción*, en el caso exitoso se debe dar que el alumno se encuentra registrado y su estado de inscripción sea *inscripto*, luego este estado es reemplazado por *promueve* o *repite* según indique el usuario y el grado se mantiene sin modificaciones. Se permite volver a cerrar una inscripción cerrada para permitir modificaciones o correcciones de errores del usuario.

CerrarInscripcionOk

$\Delta MisEstudiantes$

$alumno? : ALUMNO$

$estado? : ESTADO$

$alumno? \in registrados$

$estado? \neq inscripto$

$inscripciones' = inscripciones \oplus \{alumno? \mapsto ((inscripciones\ alumno?).1, estado?)\}$

$registrados' = registrados$

Los casos de error son dos: se intenta cerrar una inscripción modificando su estado por *inscripto* o el alumno indicado no se encuentra registrado en el sistema.

CerrarInscripcionEstadoInvalidoE

$$\Xi MisEstudiantes$$
$$estado? : ESTADO$$
$$estado? = inscripto$$

CerrarInscripcionAlumnoNoEncontradoE

$$\Xi MisEstudiantes$$
$$alumno? : ALUMNO$$
$$alumno? \notin registrados$$

$$CerrarInscripcionE == CerrarInscripcionEstadoInvalidoE \lor CerrarInscripcionAlumnoNoEncontradoE$$
$$CerrarInscripcion == CerrarInscripcionOk \lor CerrarInscripcionE$$

La última operación es *AlumnoEsGraduado* donde hay tres casos posible y cada uno tiene un mensaje de respuesta posible: el alumno está registrado y es graduado, el alumno está registrado y no es graduado o el alumno no está registrado en el sistema.

AlumnoEsGraduadoSiOk

$$\Xi MisEstudiantes$$
$$alumno? : ALUMNO$$
$$rep! : REP$$
$$alumno? \in registrados$$
$$(inscripciones\ alumno?).1 = maximoGrado$$
$$(inscripciones\ alumno?).2 = promueve$$
$$rep! = alumnoEsGraduado$$

AlumnoEsGraduadoNoOk

$\Xi MisEstudiantes$

$alumno? : ALUMNO$

$rep! : REP$

$alumno? \in registrados$

$(inscripciones\ alumno?).1 \neq maximoGrado \lor (inscripciones\ alumno?).2 \neq promueve$

$rep! = alumnoNoEsGraduado$

AlumnoEsGraduadoNoEncontradoE

$\Xi MisEstudiantes$

$alumno? : ALUMNO$

$rep! : REP$

$alumno? \notin registrados$

$rep! = alumnoNoEncontrado$

$AlumnoEsGraduadoOk == AlumnoEsGraduadoSiOk \lor AlumnoEsGraduadoNoOk$

$AlumnoEsGraduado == AlumnoEsGraduadoOk \lor AlumnoEsGraduadoNoEncontradoE$

3. {log}

La traducción de la especificación Z a un programa *{log}* se puede encontrar en el archivo **misestudiantes.pl**, para ejecutar este programa no es necesario consultar ninguna librería.

3.1. Simulaciones

Para realizar las siguientes simulaciones se cargó el programa con el type checker activado y una vez verificado que el programa tipa correctamente se desactivó para simplificar la ejecución de las simulaciones.

Primera simulación

```
misEstudiantesInicial(R0, I0) &
```

```
inscribirAlumno(R0, I0, pepe, R1, I1) &
cerrarInscripcion(R1, I1, pepe, promueve, R2, I2) &
reinscribirAlumno(R2, I2, pepe, R3, I3) &
cerrarInscripcion(R3, I3, pepe, repite, R4, I4) &
cerrarInscripcion(R4, I4, pepe, promueve, R5, I5) &
inscribirAlumno(R5, I5, pepe, R6, I6) &
alumnoEsGraduado(R6, I6, pepe, Rep1, R6, I6) &
alumnoEsGraduado(R6, I6, pepito, Rep2, R6, I6).
```

El objetivo de esta simulación es probar los usos habituales del programa, comenzando desde el estado inicial se inscribe un alumno *Pepe* y se da una serie de inscripciones cerradas y reinscripciones, incluso una ocasión donde se cerró una inscripción como repitente y luego se modificó a promoción (por ejemplo, el alumno puede haber recuperado materias luego del cierre o se corrigió una equivocación al cerrar por primera vez). Además se consulta si *Pepe* está graduado después de promover segundo grado, lo cual no sería posible, y por último se consulta por un alumno que no se encuentra registrado.

```
R0 = {},
I0 = {},
R1 = {pepe},
I1 = {[pepe,[1,inscripto]]},
R2 = {pepe},
I2 = {[pepe,[1,promueve]]},
R3 = {pepe},
I3 = {[pepe,[2,inscripto]]},
R4 = {pepe},
I4 = {[pepe,[2,repite]]},
R5 = {pepe},
I5 = {[pepe,[2,promueve]]},
R6 = {pepe},
I6 = {[pepe,[2,promueve]]},
Rep1 = alumnoNoEsGraduado,
Rep2 = alumnoNoEncontrado
```

Obtuvimos el resultado esperado, los cambios de estado fueron satisfactorios y la operación de consulta funcionó correctamente.

Segunda simulación

```
misEstudiantesInicial(R0, I0) &
inscribirAlumno(R0, I0, pepe, R1, I1) &
cerrarInscripcion(R1, I1, pepe, promueve, R2, I2) &
reinscribirAlumno(R2, I2, pepe, R3, I3) &
cerrarInscripcion(R3, I3, pepe, promueve, R4, I4) &
reinscribirAlumno(R4, I4, pepe, R5, I5) &
cerrarInscripcion(R5, I5, pepe, promueve, R6, I6) &
reinscribirAlumno(R6, I6, pepe, R7, I7) &
cerrarInscripcion(R7, I7, pepe, promueve, R8, I8) &
reinscribirAlumno(R8, I8, pepe, R9, I9) &
cerrarInscripcion(R9, I9, pepe, promueve, R10, I10) &
reinscribirAlumno(R10, I10, pepe, R11, I11) &
cerrarInscripcion(R11, I11, pepe, promueve, R12, I12) &
reinscribirAlumno(R12, I12, pepe, R13, I13) &
cerrarInscripcion(R13, I13, pepe, promueve, R14, I14) &
reinscribirAlumno(R14, I14, pepe, R15, I15) &
cerrarInscripcion(R15, I15, pepe, promueve, R16, I16) &
reinscribirAlumno(R16, I16, pepe, R17, I17) &
cerrarInscripcion(R17, I17, pepe, promueve, R18, I18) &
reinscribirAlumno(R18, I18, pepe, R19, I19) &
cerrarInscripcion(R19, I19, pepe, promueve, R20, I20) &
reinscribirAlumno(R20, I20, pepe, R21, I21) &
cerrarInscripcion(R21, I21, pepe, promueve, R22, I22) &
reinscribirAlumno(R22, I22, pepe, R23, I23) &
alumnoEsGraduado(R23, I23, pepe, Rep1, R23, I23) &
cerrarInscripcion(R23, I23, pepe, promueve, R24, I24) &
alumnoEsGraduado(R24, I24, pepe, Rep2, R24, I24) &
reinscribirAlumno(R24, I24, pepe, R25, I25) &
alumnoEsGraduado(R25, I25, pepe, Rep3, R25, I25) &
cerrarInscripcion(R25, I25, pepe, repite, R26, I26) &
alumnoEsGraduado(R26, I26, pepe, Rep4, R26, I26).
```

Esta segunda simulación consiste en inscribir al alumno *Pepe* partiendo del estado inicial y realizar el ciclo cerrar inscripción/reinscribir hasta que se encuentre cursando el último grado, una vez aquí se

consulta si está graduado a lo que se debería obtener una respuesta negativa, se cierra la inscripción con una promoción para graduarlo, se consulta de nuevo y el resultado esta vez debería ser positivo. Luego, se intenta reinscribir al alumno graduado pero esto no es permitido, por lo que consultando de nuevo se debería ver que mantiene su estado de graduado. Este estado de graduado es perdido solo cuando se decide modificar el estado del último cierre de inscripción por una repitencia, y consultado por última vez el resultado sería negativo.

```
R0 = {},
I0 = {},
R1 = {pepe},
I1 = {[pepe,[1,inscripto]]},
R2 = {pepe},
I2 = {[pepe,[1,promueve]]},
.
.
.
I22 = {[pepe,[11,promueve]]},
R23 = {pepe},
I23 = {[pepe,[12,inscripto]]},
Rep1 = alumnoNoEsGraduado,
R24 = {pepe},
I24 = {[pepe,[12,promueve]]},
Rep2 = alumnoEsGraduado,
R25 = {pepe},
I25 = {[pepe,[12,promueve]]},
Rep3 = alumnoEsGraduado,
R26 = {pepe},
I26 = {[pepe,[12,repite]]},
Rep4 = alumnoNoEsGraduado
```

Los resultados fueron acortados ya que contaban con muchos estados intermedios similares. Se obtuvieron los resultados esperados tanto en los cambios de estado satisfactorios, usos inválidos de las operaciones que no modificaron el estado y consultas respondidas correctamente.

3.2. Demostración automática de propiedades

Invocando el generador de condiciones de verificación (VCG) con el comando *vcg('misestudiantes.pl').* es generado el archivo **misestudiantes-vc.pl**, consultando este archivo y ejecutando el comando *check_vcs_misestudiantes.* es posible descargar las condiciones de verificación.

En un principio todas las condiciones son verificadas correctamente excepto por la condición **inscribirAlumno_pi_inscripcionesPfunInv**, por lo cual fue necesario agregar la hipótesis **inscripcionesInv(Registrados, Inscripciones)** y luego esta condición fue descargada exitosamente. No fue necesario utilizar el comando *findh.*

4. Demostración de preservación de invariantes en Z/Eves

Utilizando la interfaz gráfica de *Z/Eves* se definió **InscribirAlumnoPI** el cual verifica que la operación **InscribirAlumno** preserva el invariante de estado **InscripcionesInv**.

theorem InscribirAlumnoPI

$$InscripcionesInv \land InscribirAlumno \Rightarrow InscripcionesInv'$$

Y su demostración es la siguiente:

proof[*InscribirAlumnoPI*]

invoke InscribirAlumno;

split InscribirAlumnoOk;

simplify;

cases;

invoke InscribirAlumnoOk;

invoke InscripcionesInv;

equality substitute;

reduce;

next;

invoke InscribirAlumnoRegistradoE;

invoke Ξ *MisEstudiantes*;

reduce;

next;

■

5. Generación de casos de prueba con Fastest

Utilizando Fastest se generarán casos de prueba para la operación *ReinscribirAlummno*, los comandos ejecutados para generar los casos de prueba fueron los siguientes:

```
loadspec misestudiantes_fastest.tex
selop CerrarInscripcion
genalltt
addtactic CerrarInscripcion_DNF_1 FT estado?
addtactic CerrarInscripcion_DNF_1 SP \oplus inscripciones
    \oplus \{ alumno? \mapsto ( ( inscripciones(alumno?) ).1 , estado? ) \}
addtactic CerrarInscripcion_DNF_2 FT estado?
addtactic CerrarInscripcion_DNF_3 SP \notin alumno? \notin registrados
genalltt
genalltca

CerrarInscripcion_VIS
  !______CerrarInscripcion_DNF_1
  |  !______CerrarInscripcion_FT_1
  |  |  !______CerrarInscripcion_SP_4
  |  |  !______CerrarInscripcion_SP_5
  |  |  !______CerrarInscripcion_SP_6
  |  |  !______CerrarInscripcion_SP_7
  |  |  !______CerrarInscripcion_SP_8
  |  |  !______CerrarInscripcion_SP_9
  |  |  !______CerrarInscripcion_SP_10
  |  |  !______CerrarInscripcion_SP_11
  |  |
  |  !______CerrarInscripcion_FT_2
  |  |  !______CerrarInscripcion_SP_12
  |  |  !______CerrarInscripcion_SP_13
  |  |  !______CerrarInscripcion_SP_14
  |  |  !______CerrarInscripcion_SP_15
  |  |  !______CerrarInscripcion_SP_16
  |  |  !______CerrarInscripcion_SP_17
  |  |  !______CerrarInscripcion_SP_18
```

```
|  |  !______CerrarInscripcion_SP_19
|  |
|  !______CerrarInscripcion_FT_3
|    !______CerrarInscripcion_SP_20
|    !______CerrarInscripcion_SP_21
|    !______CerrarInscripcion_SP_22
|    !______CerrarInscripcion_SP_23
|    !______CerrarInscripcion_SP_24
|    !______CerrarInscripcion_SP_25
|    !______CerrarInscripcion_SP_26
|    !______CerrarInscripcion_SP_27
|
|
!______CerrarInscripcion_DNF_2
|  !______CerrarInscripcion_FT_28
|  !______CerrarInscripcion_FT_29
|  !______CerrarInscripcion_FT_30
|
!______CerrarInscripcion_DNF_3
  !______CerrarInscripcion_SP_31
  !______CerrarInscripcion_SP_32
```

asdasdadasdsad

Acá explicar por qué no generó para algunas hojas del árbol de prueba

Esquemas de casos de pruebas generados

CerrarInscripcion_SP_15_TCASE

CerrarInscripcion_SP_15

CerrarInscripcion_SP_16_TCASE

CerrarInscripcion_SP_16

CerrarInscripcion_SP_23_TCASE

CerrarInscripcion_SP_23

CerrarInscripcion_SP_24_TCASE

CerrarInscripcion_SP_24

CerrarInscripcion_FT_28_TCASE

CerrarInscripcion_FT_28

$registrados = \varnothing$

$inscripciones = \varnothing$

$estado? = inscripto$

$alumno? = aLUMNO1$

CerrarInscripcion_SP_31_TCASE

CerrarInscripcion_SP_31

$registrados = \varnothing$

$inscripciones = \varnothing$

$estado? = inscripto$

$alumno? = aLUMNO1$

CerrarInscripcion_SP_32_TCASE

CerrarInscripcion_SP_32

$registrados = \{aLUMNO1\}$

$inscripciones = \varnothing$

$estado? = inscripto$

$alumno? = aLUMNO1$