Práctica 3
Verificación de Especificaciones Z con Z/EVES

Maximiliano Cristiá
Análisis de Sistemas
F.C.E.I.A. - U.N.R.

Junio 2005

1. Problemas

1. Discuta las diferencias entre validación y verificación y explique por qué la validación es un proceso particularmente difícil.

2. Considere la siguiente especificación.

State

$$x : X$$
$$y : Y$$
$$f : X \pfun Y$$

Operation

$$\Delta State$$
$$x? : X$$
$$y? : Y$$

$$f' = f \cup \{x? \mapsto y?\}$$
$$x' = x$$
$$y' = y$$

Compruebe si *Operation* satisface o no el siguiente invariante:

Invariant

$$State$$

$$x \mapsto y \notin f$$

Si no es posible probarlo, modifique *Operation* de manera tal que lo verifique.

3. La especificación que se incluye a continuación, describe mínimamente el estado de un sistema de archivos y la operación de apertura de un archivo en modo de lectura. En este sistema de archivos cada archivo se representa por un entero y tiene asociado otro entero que representa de una forma u otra la información de control de acceso (llamada *ac*).

   La política de seguridad usada en este sistema de archivos requiere, entre otras cosas, que para que se pueda abrir un archivo en modo de lectura todos los archivos abiertos previamente en modo de escritura deben tener una *ac* superior a la que posee el nuevo archivo.

   La variable de estado *ac* representa la asociación entre nombres de archivo e información de control de acceso. La variable *secmat* representa los archivos abiertos así como también en qué modo han sido abiertos.

   $$MODE ::= READ \mid WRITE$$

   State

   $$ac : \mathbb{Z} \pfun \mathbb{Z}$$
   $$secmat : \mathbb{Z} \pfun MODE$$

   OpenObjectReadConf

   $$\Delta State$$
   $$o? : \mathbb{Z}$$

   $$o? \in \mathrm{dom}\ ac$$
   $$\forall\, o : \mathrm{dom}\ secmat \mid secmat\ o = WRITE \bullet ac\ o? \leq ac\ o$$
   $$secmat' = secmat \oplus \{o? \mapsto READ\}$$
   $$ac' = ac$$

   Confinement

   $$State$$

   $$\forall\, o_1, o_2 : \mathrm{dom}\ secmat \bullet$$
   $$secmat\ o_1 = READ \land secmat\ o_2 = WRITE \Rightarrow ac\ o_1 \leq ac\ o2$$

   a) Descargue las obligaciones de prueba de *OpenObjectReadConf* y *Confinement*.

   b) Explique la razón por la cual Z/EVES propone esas obligaciones de prueba.

c) Pruebe que *OpenObjectReadConf* preserva *Confinement*.

4. Considere la siguiente especificación Z y utilice Z/EVES para responder a la consigna que se detalla más abajo.

$$[DNI, DOMICILIO]$$

SeguridadSocial

$$viveEn : DNI \pfun DOMICILIO$$
$$ingresosPorDomicilio : DOMICILIO \pfun \mathbb{Z}$$

SeguridadSocialInv

$$SeguridadSocial$$

$$\mathrm{ran}\ viveEn = \mathrm{dom}\ ingresosPorDomicilio$$

CambioDeDomicilio

$$\Delta SeguridadSocial$$
$$x? : DNI;\ nuevoDomicilio? : DOMICILIO;\ z? : \mathbb{Z}$$

$$x? \in \mathrm{dom}\ viveEn$$
$$nuevoDomicilio? \neq viveEn\ x?$$
$$viveEn' = viveEn \oplus \{x? \mapsto nuevoDomicilio?\}$$
$$ingresosPorDomicilio' = ingresosPorDomicilio \oplus \{nuevoDomicilio? \mapsto z?\}$$

Determine formalmente si la operación *CambioDeDomicilio* preserva o no el invariante de la especificación. En caso de que no lo haga explique la razón de la falla y modifique las postcondiciones de la operación de manera tal que la operación corregida verifique el invariante pero que al mismo tiempo mantenga la forma operativa. (No es necesario efectuar la prueba formal de que la nueva versión efectivamente verifica el invariante.)

5. Considere la siguiente especificación Z del problema 4 con los cambios introducidos a continuación y utilice Z/EVES para responder a las consignas que se detallan más abajo.

CambioDeDomicilio

$\Delta SeguridadSocial$
$x? : DNI;\ nuevoDomicilio? : DOMICILIO;\ z? : \mathbb{Z}$

$x? \in \operatorname{dom} viveEn$
$nuevoDomicilio? \neq viveEn\ x?$
$viveEn' = viveEn \oplus \{x? \mapsto nuevoDomicilio?\}$
$ingresosPorDomicilio'$
$\quad = \textbf{if}\ viveEn\ x? \notin \operatorname{ran}(\{x?\} \lhd viveEn)$
$\quad \textbf{then}\ \{viveEn\ x?\} \lhd ingresosPorDomicilio \oplus \{nuevoDomicilio? \mapsto z?\}$
$\quad \textbf{else}\ ingresosPorDomicilio \oplus \{nuevoDomicilio? \mapsto z?\}$

a) Descargue la obligación de prueba que Z/EVES plantea en el esquema *CambioDeDomicilio*.

b) Explique por qué Z/EVES plantea la obligación de prueba mencionada en el ítem anterior

c) Demuestre (utilizando Z/EVES) que *CambioDeDomicilio* preserva el invariante *SeguridadSocialInv*. **Ayuda:** deberá probar algunos lemas sobre teoría de conjuntos.

6. Considere los esquemas mostrados más abajo. Probar que *AddItem* preserva *SysInv*.

System

$list : \operatorname{seq} \mathbb{Z}$

AddItem

$\Delta System$
$z? : \mathbb{Z}$

$\neg\ \langle z? \rangle\ \mathsf{in}\ list$
$list' = list \frown \langle z? \rangle$

SysInv

$System$

$\forall\ l_1 : \operatorname{seq} \mathbb{Z};\ z : \mathbb{Z} \bullet list = l_1 \frown \langle z \rangle \Rightarrow \neg\ \langle z \rangle\ \mathsf{in}\ l_1$

7. Considere la siguiente especificación de parte de un sistema para reserva de pasajes de una línea aérea.

$[PASSPORT]$

$$maxSeats : \mathbb{Z}$$
$$maxSeats > 0$$

El sistema se representa por medio de dos secuencias: *seats*, que almacena los números de pasaporte que tienen asiento reservado en el avión; y *waitingList*, que guarda los números de pasaporte que han solicitado una reserva pero están en lista de espera.

$$ARS$$
$$seats, waitingList : \operatorname{seq} PASSPORT$$

Se modela la operación para solicitar una reserva. Si aun hay lugar en el avión se le asigna un asiento al siguiente pasaporte. Hasta ese momento la lista de espera debe estar vacía.

$$ReservationOK1$$
$$\Delta ARS$$
$$p? : PASSPORT$$
$$\#seats < maxSeats$$
$$seats' = seats \frown \langle p? \rangle$$
$$waitingList' = waitingList$$

Si no hay más asientos pero se siguen haciendo reservaciones, se debe poner a los siguientes pasaportes en la lista de espera (pero esta no puede ser muy larga).

$$ReservationOK2$$
$$\Delta ARS$$
$$p? : PASSPORT$$
$$\#seats = maxSeats$$
$$\#waitingList < 2 * maxSeats$$
$$seats' = seats$$
$$waitingList' = waitingList \frown \langle p? \rangle$$

$$Reservation \mathrel{\widehat=} ReservationOK1 \lor ReservationOK2$$

El invariante que se desea preservar es el siguiente:

ReservationInv

$$ARS$$

$$\#seats + \#waitingList \leq 3 * maxSeats$$

Usted debe:

a) Explicar formalmente si la operación *Reservation* preserva o no el invariante. **Ayuda:** Puede utilizar Z/EVES para responder este punto.

b) Hacer las mínimas modificaciones posibles (según los requerimientos informales) para lograr que la operación preserve el invariantes.

c) Demostrar, usando Z/EVES, que la operación modificada preserva el invariante.