Exámenes teóricos y prácticos de Z

2 de mayo de 2019

Índice

1. Ejercicios teóricos

1. Considere la siguiente especificación Z:

   $A == [\ldots]$

   $Op_A^1 == [\Delta A \ldots]$

   $Op_A^2 == [\Delta A \ldots]$

   $E == [f : \mathbb{N} \pfun A]$

   Especifique una operación, utilizando promoción de operaciones y composición, sobre E que tome dos A, aplique Op_A^1 sobre la primera instancia, Op_A^2 sobre la segunda y agrega a f las instancias de A luego de haberle aplicado las operaciones respectivas.

   Solución $Op == Op_1 \fatsemi Op_2$

   Op_1

   ΔE
   Op_A^1
   $A? : A$
   $n? : \mathbb{N}$

   $\Theta A = A?$
   $f' = f \oplus \{n? \mapsto \Theta A'\}$

   Op_2

   ΔE
   Op_A^2
   $A?? : A$
   $m? : \mathbb{N}$

   $\Theta A = A??$
   $f' = f \oplus \{m? \mapsto \Theta A'\}$

2. En un sistema operativo existe un conjunto de procesos que esperan ser ejecutados por el scheduler. Este debe seleccionar uno de ellos y, al hacerlo, lo debe eliminar de ese conjunto. Suponga que no dese especificar en detalle la política de scheduling. Especifique en Z la operación que describe como el scheduler selecciona el proceso.

Solución $[PROCESO]$

$next : \mathbb{P} PROCESO \rightarrow PROCESO$

Sistema

$$
\begin{array}{l}
\vdots \\
esperando : \mathbb{P} PROCESO \\
activo : PROCESO \\
\vdots
\end{array}
$$

CambiarProceso

$\Delta Sistema$

$$
\begin{array}{l}
\vdots \\
esperando' = esperando \setminus \{next\,(esperando)\} \cup \{activo\} \\
activo' = next\,(esperando) \\
\vdots
\end{array}
$$

3. En Z no existe el tipo booleano (es decir el tipo que contiene solo los dos valores de verdad, *true* y *false*) y no se puede definir. Más aun, se desaconseja utilizar tipos similares (es decir, tipos que intentan capturar el valor de verdad de cierto predicado). Mediante un ejemplo explique como lo reemplazaría. El ejemplo debe mostrar primero una especificación que usa un tipo equivalente al booleano y luego la misma especificación pero sin utilizarlo.

Solución incorrecta $[PERSONA]$

$edad ::= mayor \mid menor$

$mayor : PERSONA \rightarrow edad$

Solución correcta $[PERSONA]$

$mayores : \mathbb{P} PERSONA$

4. Sean A y B dos esquemas Z y $\otimes$ un conector lógico básico. Explique formalmente el significado del esquema $A \otimes B$.

Solución Sean $A == [D_A \mid P_A]$ y $B == [D_B \mid P_B]$ entonces dicho esquema será: $A \otimes B == [D_A;\ D_B \mid (P_A) \otimes (P_B)]$.

5. Determine el tipo de la expresión $f \cup \{x \mapsto y\}$ siendo $f : X \to Y$; $x : X$; $y : Y$; justifique su respuesta.

   Solución Observemos primero que las funciones no son un tipo, sino un conjunto. Luego el tipo de dicha expresión como así también el de f es $X \leftrightarrow Y$, es decir, relación de X a Y. Vale la pena observar que el tipo $X \leftrightarrow Y$ es simplemente un sinónimo de $\mathbb{P}(X \times Y)$.

6. Defina formalmente el operador Z de composición $\fatsemi$.

   Solución Sean e_i las variables que declaran X e Y, entonces su composición será: $X \fatsemi Y == (X\,[e_i''/e_i'] \land Y\,[e_i''/e_i]) \setminus (e_i'')$.

7. Describa cómo puede reemplazarse el tipo bool en Z.

   Solución Un predicado $P : X \to \{true, false\}$ puede ser reemplazado por una definición axiomática

   $$\mid Q : \mathbb{P}X$$

   designada como: x satisface el predicado $P \approx x \in Q$.

8. Explique la semántica del operador Θ de Z.

   Solución COMPLETAR.

9. El siguiente modelo Z tiene un error de estilo: detéctelo y corríjalo.

   $[X, Y]$

   $Bool ::= F \mid V$

Estado
$existen : X \pfun Bool$

*Oper*1

$\Delta Estado$
$x? : X$

$x? \notin dom\,(existen)$
$existen' = existen \cup \{x? \mapsto F\}$

*Oper*2

$\Delta Estado$
$x? : X$

$x? \in dom\,(existen)$
$existen' = existen \cup \{x? \mapsto V\}$

Solución $[X]$

Estado

$existen : \mathbb{P} X$
$noexisten : \mathbb{P} X$

*Oper*1

$\Delta Estado$
$x? : X$

$x? \notin existen \land x? \notin noexisten$
$existen' = existen$
$noexisten' = noexisten \cup \{x?\}$

*Oper*2

$\Delta Estado$
$x? : X$

$x? \in existen \lor x? \in noexisten$
$existen' = existen \cup \{x?\}$
$noexisten' = noexisten$

10. Explique y ejemplifique las dos formas vistas en clase para formalizar los invariantes de estado de una especificación Z. Analice las ventajas y desventajas de ambas formas.

Solución COMPLETAR.

11. Determine si las funciones parciales en Z constituyen un tipo o no. En cualquier caso justifique la respuesta.

Solución No son un tipo sino un conjunto pues las operaciones definidas sobre ellas pueden resultar en una instancia que no es una función, Por ejemplo $f \cup \{1 \mapsto 2, 1 \mapsto 2\}$.

12. Explique la diferencia entre requerimientos y especificaciones.

Solución COMPLETAR.

2. Ejercicios prácticos

2.1. Administración de usuarios de un SO

2.1.1. Requerimientos

Especificar en Z la interfaz de administración de usuarios de un sistema operativo UNIX según los siguientes requerimientos:

- Se deben especificar las operaciones: creación de un usuario, eliminación de un usuario, login de un usuario presentando su contraseña, logout de un usuario, consulta de los usuarios logueados, agregar un usuario a su subgrupo de usuarios.
- Cada usuario puede estar en exactamente un grupo de usuarios. Los nombres de los grupos de usuarios son fijos.
- Tener en cuenta que un usuario siempre debe tener una contraseña la cual se guarda encriptada (piense cuidadosamente cómo especificar esto sin describir un algoritmos criptográfico particular).
- Existe un usuario especial, *root*, que es el único que puede agregar y borrar usuarios y asignar un usuario a su grupo, pero para ello debe estar logueado.

Especificar solo los casos exitosos excepto para la operación login, de la cual se deben especificar todos los casos.

2.1.2. Designaciones

- u es un nombre de usuario $\approx u \in USUARIO$.
- Existe un super-usuario $\approx root$.
- c es una contraseña $\approx u \in CONTRASEÑA$.
- Existe una contraseña vacía $\approx empty$.
- e es una contraseña encriptada $\approx e \in ENCRIPTADA$.
- La contraseña c encriptada es $\approx encrypt\,(e)$.
- g es un grupo $\approx g \in GRUPO$.

- La operación resulto satisfactoria $\approx ok$.
- Se produjo un error en la operación $\approx error$.
- En el estado actual se guarda información sobre los usuarios/contraseñas us, usuarios/grupos gs y los usuarios logueados son $ls \approx Sistema\,(us, gs, ls)$.
- El usuario $u?$ agrega el usuario $n?$ con contraseña $c?$ y se informa el resultado $r!$ de la operación $\approx AgregarUsuario\,(u?, n?, c?, r!)$.
- El usuario $u?$ borra el usuario $n?$ y se informa el resultado $r!$ de la operación $\approx BorrarUsuario\,(u?, n?, r!)$.
- El usuario $u?$ agrega el usuario $n?$ al grupo $g?$ y se informa el resultado $r!$ de la operación $\approx AgregarGrupo\,(u?, n?, g?, r!)$.
- El usuario $u?$ consulta los usuarios logueados $us!$ y se informa el resultado $r!$ de la operación $\approx ConsultaUsuarios\,(u?, us!, r!)$.
- El usuario $u?$ se desloguea y se informa el resultado $r!$ de la operación $\approx LogOut\,(u?, r!)$.
- El usuario $u?$ intenta loguearse con contraseña $c?$ y se informa el resultado $r!$ de la operación $\approx Login\,(u?, c?, r!)$.

2.1.3. Tipos

$$\left[USUARIO, CONTRASEÑA, ENCRIPTADA, GRUPO\right]$$
$$result ::= error \mid ok$$

2.1.4. Definiciones axiomáticas

$$root : USUARIO$$
$$empty : CONTRASEÑA$$
$$encrypt : CONTRASEÑA \rightarrowtail ENCRIPTADA$$

2.1.5. Esquemas

Sistema

$$us : USUARIO \pfun ENCRIPTADA$$
$$gs : USUARIO \pfun GRUPO$$
$$ls : \mathbb{P} USUARIO$$

SistemaInit

$$Sistema$$
$$us = \{root \mapsto encrypt\,(empty)\}$$
$$gs = \varnothing$$
$$ls = \varnothing$$

AgregarUsuarioOk

$$\Delta Sistema$$
$$u? : USUARIO$$
$$n? : USUARIO$$
$$c? : CONTRASEÑA$$
$$r! : result$$

$$u? = root \land u? \in ls$$
$$n? \notin dom\,(us)$$
$$us' = us \cup \{n? \mapsto encrypt\,(c?)\}$$
$$gs' = gs$$
$$ls' = ls$$
$$r! = ok$$

BorrarUsuarioOk

$$\Delta Sistema$$
$$u? : USUARIO$$
$$n? : USUARIO$$
$$r! : result$$

$$u? = root \land u? \in ls$$
$$n? \in dom\,(us) \land n? \neq root$$
$$us' = \{n?\} \ndres us$$
$$gs' = \{n?\} \ndres gs$$
$$ls' = ls \setminus \{n?\}$$
$$r! = ok$$

AgregarGrupoOk

$\Delta Sistema$
$u? : USUARIO$
$n? : USUARIO$
$g? : GRUPO$
$r! : result$

$u? = root \land u? \in ls$
$n \in dom\,(us) \land n? \notin dom\,(gs)$
$gs' = gs \cup \{n? \mapsto g?\}$
$us' = us$
$ls' = ls$
$r! = ok$

ConsultaUsuariosOk

$\Xi Sistema$
$u? : USUARIO$
$us! : \mathbb{P} USUARIO$
$r! : result$

$u? \in ls$
$us! = ls$
$r! = ok$

LogOutOk

$\Delta Sistema$
$u? : USUARIO$
$r! : result$

$u? \in ls$
$ls' = ls \setminus \{u?\}$
$gs' = gs$
$us' = us$
$r! = ok$

LoginOk

$\Delta Sistema$
$u? : USUARIO$
$c? : CONTRASEÑA$
$r! : result$

$u? \in dom\,(us) \land u? \notin ls$
$encrypt\,(c?) = us\,(u?)$
$ls' = ls \cup \{u?\}$
$us' = us$
$gs' = gs$
$r! = ok$

WrongUser

$\Xi Sistema$
$u? : USUARIO$
$r! : result$

$u? \notin dom\,(us)$
$r! = error$

AlreadyLogged

$\Xi Sistema$
$u? : USUARIO$
$r! : result$

$u? \in ls$
$r! = error$

InvalidLogin

$\Xi Sistema$
$u? : USUARIO$
$c? : CONTRASEÑA$
$r! : result$

$encrypt\,(c?) \neq us\,(u?)$
$r! = error$

$Login == LoginOk \lor WrongUser \lor AlreadyLogged \lor InvalidLogin$

2.1.6. Invariantes

$InvRoot \land BorrarUsuario \Rightarrow InvRoot'$

InvRoot

$Sistema$

$root \in dom\,(us)$

$$InvSistema \land BorrarUsuario \land Logout \land Login \land AgregarGrupo \Rightarrow InvSistema'$$

$InvSistema$

$Sistema$

$dom\,(gs) \subseteq dom\,(us)$

$ls \subseteq dom\,(us)$

2.2. Sistema de gestión de procesos

2.2.1. Requerimientos

Especifique en Z usando promoción de operaciones los siguientes requerimientos:

Un proceso puede estar en tres estado: activo, pasivo o muerto. La señal *Activate* pasa el proceso de pasivo a activo; la señal *Kill* lo pasa de pasivo a muerto; la señal *Suspend* lo pasa de activo a pasivo. La señal *KillNow* hace que un proceso activo pasa a estar muerto. El estado inicial de un proceso es pasivo.

En un cierto sistema cada proceso se identifica por un identificador de proceso. El sistema presenta una interfaz que permite suscribir un proceso a una de las señales mencionadas en el párrafo anterior. Es decir, si una de las señales aparece, el sistema la comunica a todos los procesos suscritos a esa señal (lo que implica que todos ellos cambian de estado de acuerdo a las reglas mencionadas más arriba). Puede suscribirse un proceso a más de una señal simplemente utilizando la interfaz varias veces. El usuario de la interfaz es quien determina el identificador para el proceso que se esté suscribiendo; el sistema rechazará la suscripción si el identificador ya está usado para la misma señal.

El sistema establece que uno de los procesos es el primario; cuando aun no se ha suscrito ningún proceso el primario es un proceso especial llamado *idle*. Cada vez que llega una señal *Activate*, el sistema elige aleatoriamente entre los procesos afectados al que será el nuevo proceso primario.

Nota Para modelar la aleatoriedad es suficiente la parte más básica de la teoría de conjuntos.

2.2.2. Designaciones

- i es un identificador de proceso $\approx i \in ID$.

- El proceso se encuentra en estado activo $\approx activo$.
- El proceso se encuentra en estado pasivo $\approx pasivo$.
- El proceso se encuentra en estado muerto $\approx muerto$.
- El proceso esta suscito a la señal activate $\approx activate$.
- El proceso esta suscito a la señal kill $\approx kill$.
- El proceso esta suscito a la señal suspend $\approx suspend$.
- Existe un proceso inicial especial $\approx idle$.
- El proceso esta suscripto a las señales s y se encuentra en el estado e $\approx PROCESO\,(e, s)$.
- El proceso recibe la señal α y obra en consecuencia $\approx PROCESO\alpha$.
- El proceso se suscribe a la señal $s?$ $\approx PROCESOSuscribir\,(s?)$.
- El proceso actual del sistema es a y la lista de procesos es ps $\approx Sistema\,(ps, a)$.
- Se crea un nuevo proceso $i?$ $\approx SistemaNuevoProceso\,(i?)$.
- El sistema suscribe al proceso $i?$ $\approx SistemaSuscribir\,(i?)$.
- El sistema recibe la señal activate y actualiza sus procesos $\approx SistemaActivate$.
- El sistema recibe la señal kill y actualiza sus procesos $\approx SistemaKill$.
- El sistema recibe la señal suspend y actualiza sus procesos $\approx SistemaSuspend$.

2.2.3. Tipos

$[ID]$

$estado ::= activo \mid pasivo \mid muerto$

$señales ::= activate \mid kill \mid suspend$

$PROCESO$

$e : estado$

$s : \mathbb{P} señales$

2.2.4. Definiciones axiomáticas

$$idle : ID$$

2.2.5. Esquemas

PROCESOInit

$$PROCESO$$

$$e = pasivo$$
$$s = \varnothing$$

PROCESOActivateOk

$$\Delta PROCESO$$

$$e = pasivo$$
$$activate \in s$$
$$e' = activo$$
$$s' = s$$

PROCESOKillOk

$$\Delta PROCESO$$

$$e = pasivo$$
$$kill \in s$$
$$e' = muerto$$
$$s' = \varnothing$$

PROCESOSuspendOk

$$\Delta PROCESO$$

$$e = activo$$
$$suspend \in s$$
$$e' = pasivo$$
$$s' = s$$

PROCESOSuscribirOk

$$\Delta PROCESO$$
$$s? : señales$$

$$s? \notin s$$
$$e' = e$$
$$s' = s \cup \{s?\}$$

Sistema

$$ps : ID \pfun PROCESO$$
$$a : ID$$

SistemaInit

$$Sistema$$
$$PROCESOInit$$
$$ps = \{idle \mapsto \Theta PROCESO\}$$
$$a = idle$$

SistemaNuevoProcesoOk

$$\Delta Sistema$$
$$PROCESOInit$$
$$i? : ID$$
$$i? \notin dom\,(ps)$$
$$ps' = ps \cup \{i? \mapsto \Theta PROCESO\}$$
$$a' = a$$

SistemaSuscribirOk

$$\Delta Sistema$$
$$PROCESOSuscribirOk$$
$$i? : ID$$
$$i? \in dom\,(ps) \land i? \neq idle \land ps\,(i?)\,.e \neq muerto$$
$$ps\,(i?) = \Theta PROCESO$$
$$ps' = ps \oplus \{i? \mapsto \Theta PROCESO'\}$$
$$a' = a$$

SistemaActivateOk

$$\Delta Sistema$$
$$ps' = (\lambda\, i : ID \mid ps\,(i) = \Theta PROCESO \land PROCESOActivateOk \bullet \Theta PROCESO')$$
$$a' \in \{i : ID \mid i \in dom\,(ps') \land ps'\,(i)\,.e = activo\}$$

SistemaKillOk

$$\Delta Sistema$$
$$ps' = (\lambda\, i : ID \mid ps\,(i) = \Theta PROCESO \land PROCESOKillOk \bullet \Theta PROCESO')$$
$$a' = a$$

SistemaSuspendOk

$$\Delta Sistema$$
$$ps' = (\lambda\, i : ID \mid ps\,(i) = \Theta PROCESO \land PROCESOSuspendOk \bullet \Theta PROCESO')$$
$$a' = a$$

$$PROCESOKillNowOk == PROCESOSuspendOk \fatsemi PROCESOKillOk$$
$$SistemaKillNowOk == SistemaSuspendOk \fatsemi SistemaKillOk$$

2.2.6. Invariantes

$InvActive \land SistemaActivate \Rightarrow InvActive'$

$InvActive$

$Sistema$

$a \in dom\,(ps)$

2.3. Gestión de Hoteles

2.3.1. Requerimientos

Un hotel tiene N habitaciones con diversas capacidades (entre 2 y 8 camas) para huéspedes. A las habitaciones se accede utilizando una llave electrónica (con forma de tarjeta de crédito) que se configura en el momento en que el huésped hace el check-in. El conserje inserta la llave en un dispositivo conectado a una computadora y luego debe usar el sistema de gestión del hotel para configurar la llave. La operación de check-in incluye, entonces, tomar los datos del huésped (entre ellos el período en que permanecerá en el hotel), asignarle una habitación y configurarle la llave de la habitación. Si bien la habitación se le asigna a un huésped, esta puede ser ocupada por otras personas (familiares, amigos, etc.) y es importante registrar ese número para que sea tenido en cuenta en la logística general del hotel. Obviamente no se puede asignar una habitación que ya está ocupada (total o parcialmente) dentro del período en que permanecerá el huésped, ni se pueden registrar más personas en la habitación que la capacidad de esta.

1. Usando promoción de operaciones especifique en Z la asignación de una habitación a un huésped.
2. Especifique la configuración de una llave para una habitación.
3. Especifique la operación de check-in usando de la forma más conveniente el lenguaje de especificación.
4. Especifique una operación que liste todas las habitaciones ocupadas y la fecha en que se desocuparán.

2.3.2. COMPLETAR

2.3.3. Tipos

$[FECHA, DNI, ID]$

HABITACION

$capacidad : \mathbb{N}_1$
$ocupantes : \mathbb{P} DNI$
$fechas : \mathbb{P} FECHA$

LLAVE

$id : ID$
$fechas : \mathbb{P} FECHA$

2.3.4. Definiciones Axiomáticas

$ultimo : \mathbb{P}_1 FECHA \rightarrow FECHA$

2.3.5. Esquemas

HABITACIONAsignarOk

$\Delta HABITACION$
$o? : \mathbb{P} DNI$
$f? : \mathbb{P} FECHA$

$\# o? \leq capacidad$
$fechas = \varnothing$
$ocupantes' = o?$
$fechas' = f?$
$capacidad' = capacidad$

LLAVEConfigurarOk

$\Delta LLAVE$
$i : ID$
$f? : \mathbb{P} FECHA$

$id' = i$
$fechas' = f?$

Sistema

$habitaciones : ID \pfun HABITACION$
$reservadas : ID \pfun DNI$
$huespedes : \mathbb{N}$

SistemaAsignarOk

$\Delta Sistema$
$HABITACIONAsignarOk$
$d? : DNI$
$i : ID$

$i \in dom\,(habitaciones \rhd \{h : HABITACION \mid \exists\, HABITACIONAsignarOk\})$
$habitaciones\,(i) = \Theta HABITACION$
$habitaciones' = habitaciones \oplus \{i \mapsto \Theta HABITACION'\}$
$reservadas' = reservadas \cup \{i \mapsto d?\}$
$huespedes' = huespedes + \#\, o?$

SistemaConfigurarLlaveOk

$LLAVEConfigurarOk$
$ll! : LLAVE$

$ll! = \Theta LLAVE$

$CheckInOk == SistemaAsignarOk \fatsemi SistemaConfigurarLlaveOk$

SistemaListarOk

$\Xi Sistema$
$r! = ID \pfun FECHA$

$r! = (\lambda\, i : ID \mid habitaciones\,(i)\,.fecha \neq \varnothing \bullet ultimo\,(habitaciones\,(i)\,.fecha))$

2.3.6. COMPLETAR

2.4. Lista de control de acceso

2.4.1. Requerimientos

El equipo de diseño ha definido la siguiente interfaz para un módulo que representa una *lista de control de acceso* (ACL).

- $AddUserRight(u, r)$: agrega al usuario u en la ACL con el permiso r.
- $AddGrpRight(g, r)$: agrega al grupo g en la ACL con el permiso r.
- $IsReader(u)$: determina si u es un lector de la ACL, es decir si u tiene permiso de lectura o pertenece a un grupo que tiene permiso de lectura.
- $IsWriter(u)$: determina si u es un escritor de la ACL, es decir si u tiene permiso de escritura o pertenece a un grupo que tiene permiso de escritura.
- Los permisos posibles son *Read* y *Write*.

- La notación refiere al uso habitual en DOO, es decir si $a : ACL$ y $u1$ es un usuario, entonces $a.AddUserRight(u1, Read)$ agregará $u1$ con permiso $Read$ en la ACL a.
- La función que asocia grupos con usuarios está disponible en el sistema.
- Se requiere un adecuado tratamiento de los errores.

Especifique en Z este problema.

2.4.2. Designaciones (COMPLETAR)

2.4.3. Tipos

$$[USUARIO, GRUPO]$$
$$permiso ::= write \mid read$$
$$respuesta ::= yes \mid no$$

2.4.4. Definiciones axiomáticas

$$in : GRUPO \leftrightarrow USUARIO$$

2.4.5. Esquemas

ACL

$$us : premiso \leftrightarrow USUARIO$$
$$gs : permiso \leftrightarrow GRUPO$$

ACLInit

$$us = \varnothing$$
$$gs = \varnothing$$

AddUserOk

$$\Delta ACL$$
$$u? : USUARIO$$
$$p? : permiso$$
$$(p?, u?) \notin us$$
$$us' = us \cup \{(p?, u?)\}$$
$$gs' = gs$$

UsuarioTienePermiso

ΞACL
$u? : USUARIO$
$p? : permiso$

$p? \mapsto u? \in us$

$AddUser == AddUserOk \lor UsuarioTienePermiso$

AddGrpOk

ΔACL
$g? : GRUPO$
$p? : permiso$

$p? \mapsto g? \notin gs$
$gs' = gs \cup \{(p?, g?)\}$
$us' = us$

GrupoTienePermiso

ΞACL
$g? : GRUPO$
$p? : permiso$

$(p?, g?) \in gs$

$AddGrp == AddGrpOk \lor GrupoTienePermiso$

IsReaderOk

ΞACL
$u? : USUARIO$
$r! : respuesta$

$(reader, u?) \in us \lor gs(\!|\{reader\}|\!) \lhd in \rhd \{u?\} \neq \varnothing$
$r! = yes$

IsReaderError

ΞACL
$u? : USUARIO$
$r! : respuesta$

$(reader, u?) \notin us \land gs(\!|\{reader\}|\!) \lhd in \rhd \{u?\} = \varnothing$
$r! = no$

$IsReader == IsReaderOk \lor IsReaderError$

$IsWriterOk$

ΞACL

$u? : USUARIO$

$r! : respuesta$

$(writer, u?) \in us \lor gs(\!|\{writer\}|\!) \lhd in \rhd \{u?\} \neq \varnothing$

$r! = yes$

$IsWriterError$

ΞACL

$u? : USUARIO$

$r! : respuesta$

$(writer, u?) \notin us \land gs(\!|\{writer\}|\!) \lhd in \rhd \{u?\} = \varnothing$

$r! = no$

$IsWriter == IsWriterOk \lor IsWriterError$

2.4.6. Invariantes (COMPLETAR)

2.5. Gestión de Clubes (COMPLETAR)

2.5.1. Requerimientos

Un club está formado por socios y una comisión encargada de tomar las decisiones y mantener un estatuto. En el estatuto se describen un conjunto de reglas que los socios deben cumplir. Esta medida debe quedar registrada en un acta indicando el artículo del estatuto que fue violado por el socio. Por ejemplo: un socio tiene una cierta cantidad de cuotas atrasadas, entonces la comisión decide suspenderlo hasta que regularice su situación. De esta manera constará en acta dicha decisión haciendo referencia al artículo correspondiente.

Los miembros de la comisión deben ser socios del club y uno de ellos debe ser el presidente.

Las cuotas del club deben ser liquidadas una vez por mes. El club debe llevar un registro del estado de cuenta de cada socio, donde se identifique las cuotas pagas y las cuotas impagas.

El club tiene la posibilidad de reestructurar el estado de cuentas de un socio. Esto quiere decir que se pueden cancelar y crear nuevas deudas mensuales para conformar un nuevo plan de pago. Esta decisión es facultad de la comisión y debe quedar registrado en acta con el mismo procedimiento anteriormente mencionado.

Por último, cada socio puede consultar su estado de cuenta y cuál es el mínimo de cuotas que debe adeudar para hacer uso de los beneficios del club. Para esto se deberá consultar el estatuto y las actas correspondientes para verificar si el socio no tiene suspensiones efectivas.

Especifique en Z el problema.

2.5.2. COMPLETAR

2.5.3. COMPLETAR

2.5.4. COMPLETAR

2.5.5. COMPLETAR

2.5.6. COMPLETAR

2.6. Balanza de camiones (COMPLETAR)

2.6.1. Requerimientos

Una empresa posee una balanza para pesar camiones cargados con materia prima. El camión debe ubicarse más o menos sobre el centro de la balanza para que la pesada sea correcta. Con este fin la empresa instaló cuatro sensores en los vértices de un rectángulo imaginario de forma tal que cuando detectan que el camión está dentro de ese rectángulo, se debe bajar una barrera detrás del camión. Si el camión rebasa alguno de los laterales del rectángulo se enciende una (de dos) luz ubicada delante del camión que indica qué lado está rebasado.

Una vez que el camión está correctamente ubicado y se bajaron las barreras, el chofer debe deslizar una tarjeta magnética que lo identifica. Si la tarjeta es válida, se activa la balanza. Cuando el pesaje finaliza, se debe imprimir un tique con los datos del conductor y el peso. Luego se levantan las barreras.

Modele en Z las siguientes operaciones relacionadas con los requerimientos enunciados arriba:

1. Ingresa el camión a la balanza.

2. Un par de sensores emite la señal que indica que el camión está mal ubicado. Especificar ambos pares.

3. Encender una luz para indicarle al chofer qué lado está rebasado. Especificar ambas luces.

4. Un par de sensores emite la señal que indica que ese lado no está rebasado. Esta señal se emite si previamente se rebasó ese lado. Especificar ambos pares.

5. Apagar una luz para indicarle al chofer que ese lado ahora está bien ubicado.

6. Pasan 5 segundos desde que el camión ingresó a la balanza o desde que se detectó el último «lado no rebasado», en ambos casos sin que se haya detectado un "lado rebasado". En este caso se debe bajar la barrera.

7. Validar la tarjeta recibiendo su número.

8. Activar la balanza.

9. Imprimir el tique.

2.6.2. COMPLETAR

2.6.3. COMPLETAR

2.6.4. COMPLETAR

2.6.5. COMPLETAR

2.6.6. COMPLETAR

2.7. Sistema de archivos

2.7.1. Requerimientos

Un archivo consta de un nombre que lo identifica y de un contenido que consiste en una secuencia de caracteres. Existen operaciones para agregar un caracter al final de un archivo y borrar el último caracter. Un sistema de archivos consta de un conjunto de archivos. Debe ser posible cambiar el nombre de un archivo del sistema de archivos por otro que no exista en el sistema. Dando el nombre de un archivo, es posible agregar un caracter al final de aquel solo si el tamaño del archivo no supera cierta cota máxima permitida para el sistema de archivos. De igual forma debe ser posible eliminar el último caracter de un archivo dado.

Modele en Z utilizando promoción de operaciones los requerimientos anteriores. Designe los fenómenos de interés.

2.7.2. Designaciones (COMPLETAR)

2.7.3. Definiciones Axiomáticas

$$maximo : \mathbb{N}_1$$

2.7.4. Tipos

$$[NOMBRE, CARACTER]$$

$ARCHIVO$

$$nombre : NOMBRE$$
$$contenido : seq CARACTER$$

2.7.5. Esquemas

$Sistema$

$$archivos : \mathbb{P} ARCHIVO$$
$$\forall x, y : ARCHIVO \mid x \in archivos \land y \in archivos \bullet x.nombre \neq y.nombre$$

$ARCHIVORenombrarOk$

$$\Delta ARCHIVO$$
$$nuevo? : NOMBRE$$
$$nombre \neq nuevo?$$
$$nombre' = nuevo?$$
$$contenido' = contenido$$

$ARCHIVOAgregarCaracterOk$

$$\Delta ARCHIVO$$
$$c? : CARACTER$$
$$\# contenido < maximo$$
$$contenido' = contenido^\frown \langle c? \rangle$$
$$nombre' = nombre$$

$ARCHIVOBorrarCaracterOk$

$$\Delta ARCHIVO$$
$$\# contenido > 0$$
$$contenido' = front\,(contenido)$$
$$nombre' = nombre$$

$$
\begin{array}{|l}
\textit{ARCHIVOaSistema} \\
\hline
\Delta SISTEMA \\
\Delta ARCHIVO \\
n? : NOMBRE \\
\hline
\Theta ARCHIVO \in archivos \\
n? = nombre \\
archivos' = archivos \setminus \{\Theta ARCHIVO\} \cup \{\Theta ARCHIVO'\} \\
\hline
\end{array}
$$

$$SistemaRenombrarOk == ARCHIVOaSISTEMA \land ARCHIVORenombrarOk$$
$$SistemaAgregarOk == ARCHIVOaSISTEMA \land ARCHIVOAgregarCaracterOk$$
$$SistemaBorrarOk == ARCHIVOaSISTEMA \land ARCHIVOBorrarCaracterOk$$

2.7.6. Invariantes (COMPLETAR)

2.8. Transferencia de programas por red (COMPLETAR)

2.8.1. Requerimientos

Especifique un esquema de estado y una operación Z que cumplan con los siguientes requerimientos:

- Una operación carga en la memoria de una computadora un programa que le es enviado desde otra computadora.
- Como el programa puede ser muy largo, puede ocurrir que sea enviado en más de un paquete de datos, lo que implica que la operación a especificar será invocada varias veces. Esto se denomina secuencia de comandos.
- La interfaz de la operación consta de:

  Entradas: La secuencia de bytes del programa (posiblemente incompleto, dada su longitud), la dirección de memoria a partir de la cual debe cargarse el programa y el número de secuencia de paquetes de datos (CSC). Todos estos elementos forman un paquete de datos.

  Salidas: Un código de error y, posiblemente, un número natural.
- El CSC inicial debe ser mayor que cero; la carga finaliza cuando el CSC que se recibe es cero. El CSC de cada paquete de datos que se recibe debe ser una unidad menor al anterior.

- La dirección inicial debe estar dentro del sector de memoria disponible para programas y la carga no puede desbordar dicho sector.
- Se debe controlar que un paquete de datos no pise un área de memoria modificada durante la misma secuencia de comandos.
- En caso de que todo vaya bien la operación debe retornar el mensaje «Comando recibido» y modificar la memoria.
- En caso de error se debe retornar "Error de carga" más un dato según se indica a continuación. Si el CSC de un paquete está fuera de secuencia debe retornar el CSC. Si la dirección inicial implica que se sobrescribirá un área de memoria modificada en la misma secuencia de paquetes o si no está dentro del sector adecuado, se debe retornar la dirección recibida. Si se desborda la memoria se debe retornar la longitud de la secuencia de bytes.

2.8.2. COMPLETAR

2.8.3. COMPLETAR

2.8.4. COMPLETAR

2.8.5. COMPLETAR

2.8.6. COMPLETAR

2.9. Creación de personajes para videojuegos (COMPLETAR)

2.9.1. Requerimientos

Un juego de computadora permite crear diferentes monstruos para luego enfrentarse a ellos. Para crear un monstruo se le debe seleccionar una cabeza, un cuerpo y las extremidades. El jugador puede probar tantas veces como quiera los diferentes tipos de partes disponibles hasta obtener un monstruo de su agrado (o de su desagrado). Cuando un monstruo está completo se puede guardar en un repositorio de monstruos ya creados, del cual se pueden ir seleccionando monstruos para enfrentarse.

Especifique en Z las diferentes etapas de la creación/selección de un monstruo y el repositorio de monstruos según se describe arriba.

2.9.2. COMPLETAR

2.9.3. COMPLETAR

2.9.4. COMPLETAR

2.9.5. COMPLETAR

2.9.6. COMPLETAR

2.10. Fixture de liga de futbol (COMPLETAR)

2.10.1. Requerimientos

Se debe armar el «fixture» de partidos de una liga de equipos de fútbol. Los equipos inscriptos juegan todos contra todos una sola vez. Se juega solo los domingos. Los partidos se juegan siempre en cancha neutral. Se puede suponer que cada equipo posee una cancha propia. Cada equipo juega a lo sumo un partido por día. Cada equipo puede indicar en forma previa al comienzo del torneo a lo sumo 5 domingos en los cuales no puede intervenir.

Especifique en Z las operaciones (totales) que se listan a continuación, en relación a los requerimientos que se enuncian arriba:

1. Inscripción de un nuevo equipo.

2. Inclusión en el «fixture» de un nuevo partido entre dos equipos.

3. Eliminación de un equipo de la lista con la consiguiente eliminación de todos los partidos en los que participara.

2.10.2. COMPLETAR

2.10.3. COMPLETAR

2.10.4. COMPLETAR

2.10.5. COMPLETAR

2.10.6. COMPLETAR

2.11. Control de ascensores

2.11.1. Requerimientos

Un edificio contará con varios ascensores controlados desde un programa. No puede haber más de dos ascensores moviéndose al mismo tiempo. Cada ascensor puede ser llamado desde cada piso mediante un botón. Dentro de cada ascensor hay una botonera para dirigirlo hacia el piso al que se desee ir. En cada piso y en cada túnel hay un sensor que avisa del paso del ascensor por ese piso. Las puertas son manuales.

1. Especifique en Z las siguientes operaciones básicas referidas al problema de arriba: detectar que se haya pulsado un botón dentro de un ascensor, ordenar a un ascensor moverse hacia abajo o arriba, detectar que el ascensor haya pasado por un piso, detener el ascensor.

2. Promover las operaciones anteriores a nivel de un edificio que cuente con p pisos y n ascensores.

2.11.2. Designaciones (COMPLETAR)

2.11.3. Definiciones axiomáticas

$$
\begin{array}{|l}
p : \mathbb{N}_1 \\
n : \mathbb{N}_1 \\
maxA : 2
\end{array}
$$

2.11.4. Tipos

$$ESTADO ::= quieto \mid esperando \mid bajando \mid subiendo$$
$$PISO == 0..p$$
$$ID == 1..n$$

ASCENSOR

$estado : ESTADO$
$piso : PISO$
$boton : PISO$

2.11.5. Esquemas

ASCENSORInit

ASCENSOR

$estado = quieto$
$piso = 0$
$boton = 0$

ASCENSORBotonOk

$\Delta ASCENSOR$
$b? : BOTON$

$estado = quieto \land piso \neq b?$
$boton' = b?$
$estado' = esperando$
$piso' = piso$

ASCENSORSubirOk

$\Delta ASCENSOR$

$estado = esperando \land piso < boton$
$estado' = subiendo$
$piso' = piso$
$boton' = boton$

ASCENSORBajarOk

$\Delta ASCENSOR$

$estado = esperando \land piso > boton$
$estado' = bajando$
$piso' = piso$
$boton' = boton$

ASCENSORSeñalOk

$\Delta ASCENSOR$
$señal? : PISO$

$estado \in \{subiendo, bajando\} \land señal? \neq boton$
$piso' = señal?$
$estado' = estado$
$boton' = boton$

ASCENSORSeñalDetenerOk

$\Delta ASCENSOR$
$señal? : PISO$

$estado \in \{subiendo, bajando\} \land señal? = boton$
$piso' = señal?$
$estado' = quieto$
$boton' = boton$

Edificio

$ascensores : ID \pfun ASCENSOR$
$moviendo : 0..maxA$

EdificioInit

Edificio

$ascensores = (\lambda\, i : ID \bullet ASCENSORInit)$
$moviendo = 0$

ASCENSORaEdificio

$\Delta Edificio$
$\Delta ASCENSOR$
$i? : ID$

$ascensores\,(i?) = \Theta ASCENSOR$
$ascensores' = ascensores \oplus \{i? \mapsto \Theta ASCENSOR'\}$
$moviendo' = moviendo$

$EdificioBotonOk == (ASCENSORaEdificio \land ASCENSORBotonOk) \fatsemi Actualizar$
$EdificioDetener == (ASCENSORaEdificio \land ASCENSORSeñalDetener) \fatsemi Restar \fatsemi Actualizar$
$EdificioSubirOk == (ASCENSORaEdificio \land ASCENSORSubirOk) \fatsemi Sumar$
$EdificioBajarOk == (ASCENSORaEdificio \land ASCENSORBajarOk) \fatsemi Sumar$
$Restar == [\Delta EDIFICIO \mid moviendo' = moviendo - 1 \land ascensores' = ascensores]$
$Sumar == [\Delta EDIFICIO \mid moviendo' = moviendo + 1 \land ascensores' = ascensores]$
$Actualizar == Subir \lor Bajar \lor Maximo \lor SinEspera$
$Maximo == [\Xi EDIFICIO \mid moviendo = maxA]$
$SinEspera == [\Xi EDIFICIO \mid \{i : ID \mid ascensores\,(i).estado = esperando\} = \varnothing]$

Subir

EdificioSubirOk

$i? \in \{i : ID \mid ascensores\,(i).estado = esperando\}$
$moviendo < maxA$

$Bajar$

$EdificioBajarOk$

$i? \in \{i : ID \mid ascensores\,(i)\,.estado = esperando\}$
$moviendo < maxA$

2.11.6. Invariantes (COMPLETAR)

2.12. Compañía de seguros

2.12.1. Requerimientos

Una solicitud de póliza de seguro de una cierta compañía de seguros debe pasar por varias etapas antes de ser comunicada la decisión de la empresa al solicitante. La solicitud consta de varios datos que debe llenar el solicitante y de otros que son completados por las diferentes áreas de la empresa a medida que la solicitud es procesada. Las etapas son las siguientes:

1. La solicitud es recibida y se controla que los datos consignados por el solicitante sean coherentes. Si lo son se acepta la solicitud y pasa a la siguiente etapa; caso contrario la solicitud se rechaza. Los datos son coherentes si se solicita un seguro para un auto y se consigna la patente o si se solicita un seguro para una casa y se consigna el domicilio.

2. En esta etapa las solicitudes rechazadas no sufren cambios y pasan a la siguiente etapa. Las solicitudes aceptadas son evaluadas y se le asigna un precio anual por el seguro solicitado o se las puede rechazar, en cuyo caso no se asigna un valor. En cualquier caso pasan a la siguiente etapa.

3. En esta etapa las solicitudes rechazadas en las etapas anteriores no sufren cambio y pasan a la siguiente etapa. Las solicitudes aceptadas en ambas etapas son analizadas por el departamento de legales y este les adjunta un contrato, en cuyo caso es aceptada de forma definitiva.

Modele en Z utilizando composición de operaciones el proceso anterior.

2.12.2. COMPLETAR

2.12.3. Tipos

$[PATENTE, DOMICILIO, CONTRATO]$
$SEGURO ::= auto \mid casa$
$ID ::= patente \ll PATENTE \gg \mid domiclio \ll DOMICILIO \gg$

2.12.4. Definiciones Axiomáticas

$$precio : ID \pfun \mathbb{N}$$
$$contrato : ID \times \mathbb{N} \pfun CONTRATO$$

2.12.5. Esquemas

Sistema

$$e1 : \mathbb{P} ID$$
$$e2 : ID \pfun \mathbb{N}$$
$$e3 : ID \pfun CONTRATO$$
$$r : \mathbb{P} ID$$

SistemaInit

Sistema

$$e1 = \varnothing$$
$$e2 = \varnothing$$
$$e3 = \varnothing$$
$$r = \varnothing$$

SistemaSolicitudOk

$$\Delta Sistema$$
$$s? : SEGURO$$
$$i? : ID$$

$$(s? = auto \land dom\,(i?) \subseteq PATENTE) \lor (s? = casa \land dom\,(i?) \subseteq DOMICILIO)$$
$$e1' = e1 \cup \{i?\}$$
$$e2' = e2 \land e3' = e3 \land r' = r$$

SistemaSolicitudError

$$\Delta Sistema$$
$$s? : SEGURO$$
$$i? : ID$$

$$(s? = auto \land dom\,(i?) \subseteq DOMICILIO) \lor (s? = casa \land dom\,(i?) \subseteq PATENTE)$$
$$r' = r \cup \{i?\}$$
$$e1' = e1 \land e2' = e2 \land e3' = e3$$

$$Etapa1 == SistemaSolicitudOk \lor SistemaSolicitudError$$

$Etapa2$

$\Delta Sistema$

$e1' = \varnothing$

$e2' = e2 \cup (\lambda\, i : ID \mid i \in e1 \land i \in dom\,(precio) \bullet precio\,(i))$

$e3' = e3$

$r' = r \cup \{i : ID \mid i \in e1 \land i \notin dom\,(precio)\}$

$Etapa3$

$\Delta Sistema$

$e1' = e1$

$e2' = \varnothing$

$e3' = e3 \cup (\lambda\, i : ID \mid i \in dom\,(e2) \land (i, e2\,(i)) \in dom\,(contrato) \bullet contrato\,(i, e2\,(i)))$

$r' = r \cup \{i : ID;\ n : \mathbb{N} \mid \{i \mapsto n\} \in e2 \land \{i \mapsto n\} \notin dom\,(contrato)\}$

$Etapas == Etapa1 \fatsemi Etapa2 \fatsemi Etapa3$

2.12.6. COMPLETAR

2.13. Sistema judicial (COMPLETAR)

2.13.1. Requerimientos

Una medida judicial puede referirse a uno o más gravámenes. Cada medida judicial se identifica por el número de juez que la emite, las dos partes intervinientes (personas físicas o jurídicas) y la fecha de emisión de la medida. Se espera que los jueces o sus ayudantes puedan ingresar una medida o modificarla según se explica más abajo. Las medidas no pueden darse de baja.

Cada gravamen es de un cierto tipo (por ejemplo, embargo o inhibición) del cual dependen el código, el nombre y la caducidad del gravamen (es decir, una vez transcurrido el tiempo de caducidad desde la fecha de emisión de la medida que lo contiene, cesa el efecto del gravamen, lo que produce un cambio de estado en el gravamen). Además, el tipo del gravamen determina si este afecta o bien a una persona o bien a algunos o todos los inmuebles que posea una persona. Cada gravamen puede estar en uno de varios estados según se describe a continuación: activo, desde el momento en que se carga la medida en el sistema; cancelado, desde el momento en que la persona afectada toma alguna acción judicial que el juez interviniente entienda que justifica cancelarle el gravamen que pese sobre aquella o alguno de sus inmuebles; caduco, desde el momento en que, no habiendo sido cancelado, ha transcurrido el período de caducidad del mismo (si bien este cambio de estado no es iniciado

por un usuario, puede suponerse que hay un componente externo que emite una señal cada vez que transcurre un día).

Modele en Z utilizando promoción de operaciones los requerimientos anteriores. Designe los fenómenos de interés.

2.13.2. COMPLETAR

2.13.3. COMPLETAR

2.13.4. COMPLETAR

2.13.5. COMPLETAR

2.13.6. COMPLETAR

2.14. Formularios web

2.14.1. Requerimientos

Para que una persona pueda utilizar un sistema debe completar un formulario web donde se le pide nombre completo y dirección de correo electrónico. Luego el sistema agrega a la persona una base temporal. Después alguien autoriza a la persona cargando en el sistema una contraseña para ese usuario. Una vez autorizado, el sistema le comunica a la persona la contraseña por correo electrónico y desde ese momento pasa a la base definitiva.

Utilice composición de operaciones para modelar en Z todo el proceso de alta de un usuario.

2.14.2. Designaciones (COMPLETAR)

2.14.3. Tipos

$$[NOMBRE, CORREO, CONTRASEÑA, MAIL]$$

2.14.4. Definiciones Axiomáticas

$$send : CORREO \times CONSTRASEÑA \rightarrow MAIL$$

2.14.5. Esquemas

Sistema

$$solicitudes : CORREO \pfun NOMBRE$$
$$autorizados : CORREO \pfun NOMBRE \times CONTRASEÑA$$
$$definitivos : CORREO \pfun NOMBRE \times CONTRASEÑA$$

SistemaInit

Sistema

$$solicitudes = \varnothing$$
$$autorizados = \varnothing$$
$$definitivos = \varnothing$$

SistemaSolicitarOk

$\Delta Sistema$

$$c1? : CORREO$$
$$n1? : NOMBRE$$

$$c1? \notin dom\,(solicitudes)$$
$$solicitudes' = solicitudes \cup \{c1? \mapsto n1?\}$$
$$autorizados' = autorizados$$
$$definitivos' = definitivos$$

SistemaAutorizarOk

$\Delta Sistema$

$$c2? : CORREO$$
$$p2? : CONTRASEÑA$$

$$c2? \in dom\,(solicitudes) \land c2? \notin dom\,(autorizados)$$
$$solicitudes' = \{c2?\} \ndres solicitudes$$
$$autorizados' = autorizados \cup \{c2? \mapsto (solicitudes\,(c2?)\,, p2?)\}$$
$$definitivos' = definitivos$$

SistemaComunicarOk

$\Delta Sistema$

$$c2? : CORREO$$
$$p2? : CONTRASEÑA$$
$$r! : MAIL$$

$$solicitudes' = solicitudes$$
$$autorizados' = \{c2?\} \ndres autorizados$$
$$definitivos' = definitivos \cup \{autorizados\,(c2?)\}$$
$$r! = send\,(c2?, p2?)$$

$SistemaAltaOk == SistemaSolicitarOk \fatsemi SistemaAutorizarOk \fatsemi SistemaComunicarOk$

2.14.6. Invariantes (COMPLETAR)

2.15. Sistema de documentación

2.15.1. Requerimientos

Un documento tiene un nombre que lo identifica, un cuerpo de texto y una fecha de última modificación. Al modificar el documento se debe actualizar la fecha correspondiente. Además, puede buscarse una cadena de caracteres dada en el cuerpo de un documento comunicándole al usuario todos los números de línea donde se encuentra. Modificar un documento significa cambiar la línea por otra o agregar o borrar una; las líneas están numeradas. Si el documento tiene las lineas numero i y j con $i > j + 1$, entonces debe tener todas las líneas numeradas entre i y j . Si el usuario no las agregó, el sistema debe poner líneas en blanco. Cualquier modificación se lleva a cabo dando el número de línea y la cadena de caracteres para esa línea.

Un sistema de documentación consiste en un conjunto de documentos y un conjunto de usuarios que pueden modificar o buscar en los documentos. Si un usuario registrado quiere modificar un documento debe dar su nombre, número de línea y cadena de reemplazo (si es necesario). Si un usuario quiere buscar una cadena en la base de documentación, el sistema le responderá dándole el nombre de cada documento donde la cadena aparece junto a todos los números de línea donde se encuentra la cadena.

Modele en Z utilizando promoción de operaciones los requerimientos anteriores.

2.15.2. Designaciones (COMPLETAR)

2.15.3. Tipos

$[ID, CARACTER, FECHA, USR]$
$CADENA == seq CARACTER$
$error ::= si \mid no$

2.15.4. Definiciones Axiomáticas

$usuarios : \mathbb{P} USR$

2.15.5. Esquemas

$$
\begin{array}{|l}
DOCUMENTO \\
\hline
cuerpo : seq\,(CADENA) \\
fecha : FECHA \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
DOCUMENTOBuscarOk \\
\hline
\Xi DOCUMENTO \\
usuario? : USR \\
cadena? : CADENA \\
lineas! : \mathbb{P}\,\mathbb{N}_1 \\
error! : error \\
\hline
usuario? \in usuarios \\
lineas! = \{i : dom\,(cuerpo) \mid cadena?\ infix\ cuerpo\,(i)\} \\
error! = no \\
\hline
\end{array}
$$

$$
\begin{array}{|l}
DOCUMENTOCambiarOk \\
\hline
\Delta DOCUMENTO \\
usuario? : USR \\
cadena? : CADENA \\
linea? : \mathbb{N}_1 \\
fecha? : FECHA \\
error! : error \\
\hline
usuario? \in usuarios \\
cuerpo' = (cuerpo \oplus \{linea? \mapsto cadena?\}) \cup \{i : \mathbb{N}_1 \mid i \in \#\,cuerpo + 1..linea? - 1 \bullet i \mapsto \langle\rangle\} \\
fecha' = fecha? \\
error! = no \\
\hline
\end{array}
$$

$$DOCUMENTOAgregarOk == [DOCUMENTOCambiarOk \mid linea? \notin dom\,(cuerpo)]$$

$$DOCUMENTOModificarOk == [DOCUMENTOCambiarOk \mid linea? \in dom\,(cuerpo)]$$

$$
\begin{array}{|l}
DOCUMENTOBorrarOk \\
\hline
\Delta DOCUMENTO \\
usuario? : USR \\
linea? : \mathbb{N}_1 \\
fecha? : FECHA \\
error! : error \\
\hline
usuario? \in usuarios \land linea? \in dom\,(cuerpo) \\
cuerpo' = cuerpo \oplus \{linea? \mapsto \langle\rangle\} \\
fecha' = fecha? \\
error! = no \\
\hline
\end{array}
$$

$$Sistema$$
$$s : ID \pfun DOCUMENTO$$

$$SistemaBuscarOk$$
$$\Xi Sistema$$
$$DOCUMENTOBuscarOk$$
$$r! : ID \pfun \mathbb{P}\mathbb{N}_1$$
$$r! = \{(x, y) : ID \times \mathbb{P}\mathbb{N}_1 \mid x \in dom\,(s) \land \exists\, DOCUMENTOBuscarOk \land s\,(x) = \Theta DOCUMENTO \land y = lineas!\}$$

$$DOCUMENTOaSISTEMA$$
$$\Delta Sistema$$
$$i? : ID$$
$$i? \in dom\,(s) \land s\,(i?) = \Theta DOCUMENTO$$
$$s' = s \oplus \{i? \mapsto \Theta DOCUMENTO'\}$$

$$SistemaModificarOk == DOCUMENTOaSISTEMA \land DOCUMENTOModificarOk$$
$$SistemaAgregarOk == DOCUMENTOaSISTEMA \land DOCUMENTOAgregarOk$$
$$SistemaBorrarOk == DOCUMENTOaSISTEMA \land DOCUMENTOBorrarOk$$

2.15.6. Invariantes (COMPLETAR)