Tokeneer en Fastest™: Derivación automática de casos de prueba a partir de la especificación formal de un sistema seguro

Brenda Lieber

Tesina de Grado
Director: Maximiliano Cristiá
Licenciatura en Ciencias de la Computación
Facultad de Ciencias Exactas, Ingeniería y Agrimensura
Universidad Nacional de Rosario
Argentina

Noviembre 2009

Resumen

Fastest^{TM} es una herramienta que genera casos de prueba a partir de especificaciones formales en Z. El proyecto Tokeneer, desarrollado por la empresa Praxis High Integrity Systems, define un sistema seguro usando métodos formales. En particular, la especificación está escrita en Z. El trabajo realizado en esta tesina consiste en adaptar la especificación de Tokeneer para poder generar casos de prueba con Fastest^{TM} y comparar estos resultados con los obtenidos en el proyecto Tokeneer.

Índice general

Capítulo 1

Introducción

El testing es una etapa crítica en el ciclo de vida del software. Puede ser muy efectiva si se lleva a cabo rigurosamente y las especificaciones formales proveen las bases para este tipo de prácticas. Especificar sistemas formalmente tiene muchas ventajas, siendo el testing de software uno de sus usos más prácticos e inmediatos. Al automatizar este proceso, se logra reducir los costos de desarrollo de software significativamente.

Las especificaciones formales son descripciones matemáticas de software o hardware que pueden ser usadas para desarrollar una implementación. Describen *que* debe hacer el sistema, y no necesariamente *como* lo hace. Dada una especificación, es posible usar técnicas de verificación formal para demostrar que el diseño de un sistema es correcto con respecto a la especificación. Ésto tiene la ventaja de que el diseño de un sistema puede ser revisado antes de invertir en su implementación. Un diseño no puede ser declarado "correcto" por si mismo, pero solo "correcto con respecto a su especificación". Si la especificación formal describe el problema a resolver correctamente es un tema distinto. Es también un problema difícil de resolver, ya que se debe construir una representación formal abstracta de un dominio de problema informal concreto, y tal paso de abstracción no es sencillo de probar formalmente. Sin embargo, es posible validar una especificación probando teoremas que involucran propiedades que la especificación debería cumplir. Si es correcta, estos teoremas refuerzan la comprensión del especificador acerca de la especificación y su relación con el dominio del problema subyacente. Si no, la especificación probablemente debe ser modificada para reflejar mejor la comprensión del dominio de aquellos involucrados en producir (e implementar) la especificación. La notación Z es un ejemplo de uno de los principales lenguajes de especificaciones formales.

El testing es el proceso de validación (se está construyendo el software correcto?) y verificación (se está construyendo el software correctamente?) que un programa de software:

1. cumple con los requerimientos de negocio y técnicos que guían su diseño y desarrollo;
2. funciona como se espera; y
3. puede ser implementado con las mismas características.

El testing de software, dependiendo del método de testing empleado, puede ser implementado en cualquier momento del proceso de desarrollo. Sin embargo, la mayor parte del esfuerzo ocurre después de que los requerimientos han sido identificados y el proceso de codificación ha sido completado. Por lo tanto, la metodología de testing es gobernada por la metodología de desarrollo de software adoptada.

El testing generalmente se agrupa por nivel de especificidad de la prueba. El **testing de unidad** se refiere a las pruebas que verifican la funcionalidad de una sección específica de código, normalmente a nivel de las funciones. Este tipo de pruebas son generalmente escritas por desarrolladores mientras trabajan en el código para asegurarse que una función específica funciona como se espera. El testing de unidad por si solo no puede verificar la funcionalidad del código, pero es usado para asegurarse que los bloques que construyen el software funcionan independientemente. El **testing de sistema** prueba un sistema integrado completamente para verificar que cumple con sus requerimientos.

Los métodos de testing se dividen tradicionalmente en dos enfoques: **testing funcional** o de **caja negra**, donde no se tiene conocimiento de la implementación interna del sistema y **testing estructural** o de **caja blanca**, donde el tester tiene acceso a las estructuras de datos internas y algoritmos que implementan el sistema.

El **testing basado en especificaciones** es un tipo de testing de caja negra que apunta a testear la funcionalidad del software con respecto a los requerimientos. La especificación de una operación puede ser usada para derivar sistemáticamente casos de prueba abstractos y evaluar automáticamente los resultados que resultan de testear con ellos al programa que supuestamente implementa la operación [2]. Este enfoque ha sido puesto en práctica con especificaciones escritas en el lenguaje de especificación Z.

Fastest^{TM} es una herramienta para derivar casos de prueba a partir de una especificación formal escrita en lenguaje Z que implementa las técnicas de testing basado en modelos descripto en [1], [2] y [3].

Tokeneer es un sistema desarrollado dentro del marco de un proyecto de investigación cuyo fin es demostrar que es posible desarrollar sistemas seguros rigurosamente de una manera rentable.

El resto de este informe explica como se usó Fastest^{TM} para generar el testing del sistema Tokeneer automáticamente.

En el capítulo 2 se describe brevemente Fastest^{TM} que es la herramienta de automatización de testing analizada en este trabajo.

El capítulo 3 explica el proyecto Tokeneer, y la especificación formal de este sistema en particular.

El capítulo 4 explica las adaptaciones necesarias realizadas sobre la especificación de Tokeneer al estándar de la notación soportado por Fastest^{TM} y al conjunto de operaciones aceptado actualmente por la herramienta.

En el capítulo 5 se muestra como se derivaron los casos de prueba generados por Fastest^{TM} para la especificación del proyecto Tokeneer.

Finalmente, en el capítulo 6 se presentan las conclusiones y posibles trabajos futuros para continuar esta investigación.

En los apéndices A, B y C se incluyen la especificación completa adaptada de Tokeneer y las clases y casos de prueba generados en este trabajo.

Capítulo 2

Fastest^{TM}

Fastest^{TM} es una herramienta para derivar casos de prueba a partir de una especificación formal escrita en lenguaje Z. El primer prototipo de esta herramienta fue desarrollado por Maximiliano Cristiá y Pablo Rodriguez Monetti para el proyecto Flowx, financiado en conjunto por la empresa Flowgate Consulting y FONTAR (Fondo Tecnológico Argentino). Actualmente, la herramienta es mantenida por Flowgate Consulting y puede descargarse desde
http://www.flowgate.net/tools/Fastest.tar.gz. Para el trabajo descripto en este documento, la versión utilizada de Fastest^{TM} fue la 1.3.2, instalada sobre un entorno Windows XP con Java SE Runtime Environment 1.6.

Fastest^{TM} implementa las técnicas de testing basado en modelos descripto en [1], [2] y [3]. Utiliza una arquitectura de cliente-servidor e invocación implícita, lo cual implica un mejor rendimiento y la capacidad de introducir cambios de una manera simple.

La especificación que Fastest^{TM} toma como entrada debe estar escrita en Latex, siendo éste el formato abierto más utilizado para la documentación de especificaciones en Z. Fastest^{TM} ha sido integrado con el proyecto CZT (Community Z Tools, http://czt.sourceforge.net) para evitar esfuerzos duplicados en el desarrollo de módulos centrales. La versión de Latex que soporta Fastest^{TM} es la soportada por CZT y se basa en el basado en el ISO de Z [9].

La versión utilizada de la herramienta no soporta el lenguaje Z completo, como por ejemplo, composición de esquemas, canalización de esquemas, el operador *Theta*, el tipo esquema y definiciones axiomáticas. Actualmente Fastest^{TM} puede utilizarse para realizar testing de unidad. En el futuro, se tratará de extender a testing de integración y de sistema.

Los pasos para generar casos de prueba con Fastest^{TM} son lo siguientes:

1. Abrir una ventana de comandos, moverse hasta el directorio donde se encuentra instalada la herramienta y ejecutar el siguiente comando:

   ```
   java -jar fastest.jar
   ```

2. Cargar la especificación:

   ```
   loadspec <archivo>.tex
   ```

3. Seleccionar las operaciones a testear:

```
selop <operación>
```

4. [Opcional] Seleccionar tácticas de testing:

   ```
   addtactic <operación> <táctica> <parámetros>
   ```

5. Generar árboles de prueba (uno por cada operación):

   ```
   genalltt
   ```

6. Explorar los arboles y las clases de prueba:

   ```
   showtt
   ```

   ```
   showsch -tcl
   ```

7. [Opcional] Podar los árboles:

   ```
   prunefrom | prunebelow <clase>
   ```

8. [Opcional] Definir modelos finitos:

   ```
   setfinitemodel <clase> <opciones>
   ```

9. Calcular casos abstractos:

   ```
   genalltca
   ```

10. [Opcional] Iterar sobre los últimos 3 pasos todas las veces que se quiera en cualquier orden

11. Explorar los casos de prueba:

    ```
    showsch -tca
    ```

Para ver los comandos de Fastest™ en detalle, referirse al comando de ayuda o al manual de usuario que viene acompañado con la herramienta [5].

Capítulo 3

Tokeneer

Para este trabajo se ha utilizado la especificación del sistema existente Tokeneer ID Station (TIS), desarrollado en conjunto por la empresa Praxis High Integrity Systems y la NSA (Agencia de Seguridad Nacional de los Estados Unidos) [6]. El desarrollo de este sistema se realizó dentro del marco de un proyecto de investigación cuyo fin es demostrar que es posible desarrollar sistemas seguros rigurosamente de una manera rentable.

TIS es el componente de un sistema mas grande, el sistema Tokeneer. El sistema completo provee protección a información segura ubicada en una red de estaciones de trabajo situadas en un enclave físicamente seguro.

Una Estación de Matriculación emite un token a un usuario aprobado. Para generar el token, la Estación de Matriculación depende de una Autoridad de Certificación para generar un certificado de identidad X.509 firmado y una Autoridad de Autorización para generar certificados de autorización X.509 firmados conteniendo información privilegiada y de autorización (Certificado de Privilegio) e información biométrica de identificación y autenticación (Certificado I&A).

X.509 es un estándar en criptografía para infraestructura de clave pública (PKI, por sus siglas en inglés) para identificación única e infraestructura de administración de privilegios.

El TIS es una entidad autónoma "confiable" responsable de realizar la verificación biométrica de un usuario. Para realizar esta tarea, usa la información biométrica en el Certificado IA en el token del usuario y una lectura de la huella digital del usuario. Si la identificación es exitosa, asumiendo que el usuario tiene suficiente autorización almacenada en el Certificado de Privilegio, el TIS agrega un Certificado de Autorización firmado al token del usuario y libera el cerrojo en la puerta del enclave para permitir al usuario acceder al enclave.

La Estación de Trabajo chequea el Certificado de Autorización para determinar si el usuario está actualmente autorizado a usar las facilidades que provee.

El software desarrollado en el proyecto Tokeneer fue dividido entre Funciones Centrales y Funciones de Soporte. Las Funciones de Soporte imitan a los controladores de los periféricos proveyendo comunicación a simuladores de periféricos externos. En un sistema real serían remplazados por controladores y periféricos desarrollados independientemente.

El Controlador del Lector de Tarjetas modela dos lectores de tarjetas, uno localizado afuera del enclave (el lector de token de usuario) y otro localizado dentro del enclave (el lector de token de

administrador). TIS lee el token de un usuario desde el lector de token de usuario y puede escribir un Certificado de Autorización adicional al token de usuario a través de este lector de tarjetas. TIS lee el token de un administrador desde el lector de token de administrador.

El Controlador de la Alarma modela una alarma audible localizada dentro del enclave. Esta alarma es designada para notificar a un guardia del peligro de una brecha en la seguridad. TIS controla el estado de la alarma audible (silencio o alarmando).

El Controlador de la Puerta y el Cerrojo modela un sensor en la puerta del enclave indicando si la puerta está abierta o cerrada, y el cerrojo de la puerta, que puede estar bloqueado o desbloqueado. TIS monitorea el sensor de la puerta para determinar si la puerta está cerrada o abierta. TIS controla el cerrojo de la puerta seteándolo a bloqueado o desbloqueado.

El Controlador del Visor modela la interfaz de un visor localizado fuera del enclave, que provee información a un usuario que desea entrar al enclave. TIS controla los datos presentados al usuario en este visor.

El Stub de la Biblioteca Biométrica modela una biblioteca biométrica y la interfaz al lector de huellas digitales, que típicamente sería realizada a través de la biblioteca biométrica. TIS interroga a la biblioteca biométrica para determinar si hay una lectura de huella digital disponible para análisis y pide validación de la lectura de huella digital actual contra información de plantilla proporcionada.

La Interfaz de Usuario es una simple consola y una facilidad para importar datos de configuración desde un archivo. TIS muestra información en la consola y lee simples entradas de teclado desde la consola. Actualizaciones de configuración compleja, que pueden ser típicamente realizadas a través de una GUI, son realizadas proveyendo una facilidad para importar datos de configuración desde un archivo.

El proceso de desarrollo de TIS se divide en las siguientes fases:

1. Análisis de Requerimientos
2. Especificación Formal (usando el lenguaje formal Z)
3. Diseño
4. Implementación (en el lenguaje SPARK Ada)
5. Verificación (usando herramientas de SPARK)
6. Testing

En esta tesina se utilizará la especificación formal del sistema Tokeneer para hacer la derivación automática de casos de prueba para demostrar que Fastest^{TM} puede ser utilizado en sistemas existentes con especificaciones extensas. La especificación orginal puede consultarse en [7]. La especificación adaptada para Fastest^{TM} se incluye completa en el apéndice A.

Tokeneer describe en su especificación formal de modo inequívoco lo que el sistema hace. Es importante el nivel de abstracción elegido. La especificación formal no debería tratar como el sistema es implementado, y en particular los detalles internos son deliberadamente muy abstractos. Se especifican las interacciones con el entorno externo, pero pueden ser abstraídas. Por ejemplo, se provee un modelo abstracto del Registro de Auditoría (ya que éste es exportado como parte del archivo de auditoría), pero no se proveen detalles de la estructura, formato o contenido del registro.

La especificación fue escrita en lenguaje Z acompañada de texto en lenguaje natural. La notación Z usa tipos de datos y lógica de predicados para describir la manera en que el sistema se comportará; Z es particularmente poderoso porque usa esquemas para descomponer la especificación en componentes pequeños que pueden ser analizados individualmente y luego combinados para describir el sistema como un todo. La notación Z provee un lenguaje de especificación inequívoco mientras que la narrativa en lenguaje natural asiste a los lectores, escritores y revisores para entender la intención del Z proveyendo una descripción secundaria del sistema y aconsejar en como interpretar el Z.

La especificación Z fue verificada usando el verificador de tipos Fuzz, un verificador de tipos veloz que verifica la consistencia de tipos en todas las expresiones.

La especificación formal fue desarrollada identificando estados y operaciones. Todos los estados asociados con el sistema fueron identificados: pueden ser estados mantenidos por el sistema (por ejemplo dentro de TIS los datos de configuración son parte del estado del sistema) o estados modelando el entorno externo del sistema (por ejemplo TIS hace uso de tiempo provisto externamente, y puede modificar el estado del cerrojo de la puerta). Estas dos partes del modelo están capturando distintos tipos de cosas. El estado mantenido por el sistema es una descripción del estado deseado, mientras que el modelado del entorno externo es una descripción del estado actual del mundo. Esto refleja la división identificada durante la captura de requerimientos entre los requerimientos del sistema a construir y las expectativas del entorno. Un ejemplo es la restricción explícita de la fuente del tiempo provista externamente: ésta fue designada como siempre creciente. Ésto identifica explícitamente una dependencia del entorno de proveer una fuente de tiempo confiable que nunca decrezca - si el tiempo decrece, no se requiere que el sistema responda sólidamente.

Mientras el estado era identificado se lo fue modularizando, capturando componentes de estados relacionados fuertemente dentro de un esquema, y agregando invariantes de estado mientras fuera necesario. Por ejemplo, para manejar la relación entre los componentes de la puerta, el cerrojo y la alarma, fueron identificadas invariantes que describen exactamente las condiciones bajo las cuales el cerrojo debería ser bloqueado por el sistema y la alarma activada por el sistema.

Se desarrollaron operaciones basadas en los escenarios identificados durante el análisis de requerimientos. En la mayoría de los casos estas operaciones fueron identificadas involucrando varias fases. Por ejemplo, el escenario de la entrada de un usuario al enclave involucra la recepción de un token, validarlo, posiblemente leerlo y y validar datos biométricos, escribir en el token y finalmente desbloquear la puerta. En cada etapa las cosas pueden salir mal. Se usaron esquemas de operación para describir comportamientos exitosos primero. Una vez que las condiciones para un comportamiento exitoso eran identificadas, las condiciones de error podían ser deducidas y la salida del error descripta formalmente. Por ejemplo, los datos biométricos solo pueden ser validados si coinciden con una plantilla en el token (especificado por el esquema *ValidateFingerOK*). Luego se consideraron las condiciones de error: o el token ya no puede realizar la comparación o no hay coincidencia (especificado por el esquema *ValidateFingerFail* y *UserTokenTorn* respectivamente). El proceso completo de validación de datos dactilares puede ser entonces presentado como una combinación de estos tres esquemas.

De esta manera se realizó la descripción del sistema completo. Finalmente, se definió un conjunto viable de valores para el estado inicial del sistema. Este conjunto de valores debe satisfacer todas las invariantes, y debe representar un sistema seguro.

Capítulo 4

Adaptaciones de Tokeneer al estandar Z soportado por Fastest^{TM}

Para realizar este trabajo, las siguientes herramientas fueron instaladas sobre un entorno Windows XP:

1. Java SE Runtime Environment 1.6, para correr Fastest^{TM}
2. Fastest^{TM} 1.3.2, descargado desde http://www.flowgate.net
3. Eclipse 3.3.2, como entorno para integrar las herramientas de Latex
4. MikTeX (http://miktex.org basic-miktex-2.7.3248.exe), para compilar el Latex de las especificaciones Z
5. TeXlipse (http://texlipse.sourceforge.net net.sourceforge.texlipse_1.2.2_src.zip), un plugin para Eclipse para editar Latex
6. CZT (http://czt.sourceforge.net czt_1_5_0_eclipse_plugin.zip), plugin para Eclipse de CZT (Community Z Tools) para verificación de tipos en Z
7. Fuente CZT (CZTSans.ttf, incompatible con Office 2007)

Se creó una carpeta nombrada Tokeneer4Fastest, y se copiaron los .tex de la especificación exclusivamente, renombrándolos de <x>.tex a f<x>.tex. Se creo un archivo principal .tex que solo incluye los f<x>.tex, usando el paquete CZT. Se borraron todas las narraciones y vínculos, dejando solo el Z. Finalmente, se creó un script de ant que concatena todos los .tex de la especificación en un .zed para hacer el type-checking con el plugin de CZT para Eclipse y un único .tex para pasarlo como entrada a Fastest^{TM}

El chequeo de tipos de Tokeneer se hizo con Fuzz. Fuzz es un sistema de impresión y chequeo de tipos para especificaciones Z que funciona con Latex. El estilo de Latex que utiliza Fuzz se basa en el lenguaje Z definido por Spivey en [4]. Fastest^{TM} usa CZT, que está basado en el ISO de Z [9]. A continuación se listan los comandos que hubo que remplazar y las adaptaciones que hubo que realizar por esta diferencia.

4.1. Comando `defs`

Tokeneer usa para la definición horizontal de esquemas el comando `\defs`. En CZT se usa ==, el mismo comando que se usa para abreviaturas. Todas las ocurrencias de \defs se remplazaron por ==. Por ejemplo:

```
\begin{zed}
RealWorld \defs TISControlledRealWorld \land TISMonitoredRealWorld
\end{zed}
```

se remplazó por

```
\begin{zed}
RealWorld == TISControlledRealWorld \land TISMonitoredRealWorld
\end{zed}
```

4.2. Comando `syntax`

En la especificación de Tokeneer, se uso el comando `\begin{syntax}` en lugar de `\begin{zed}` en tres definiciones de tipos. Esto no es soportado por Fastest™. Se remplazaron estos `{syntax}` por `{zed}`. Por ejemplo:

```
\begin{syntax}
ADMINOP ::=  archiveLog | updateConfigData | overrideLock | shutdownOp
\end{syntax}
```

se remplazó por

```
\begin{zed}
ADMINOP ::=  archiveLog | updateConfigData | overrideLock | shutdownOp
\end{zed}
```

4.3. Esquemas primados

El verificador de tipos CZT indicaba 11 ocurrencias de error en el uso del decorador prima en esqumas. La solución a estos errores fue remplazar el símbolo ' por ~'. Abajo se muestra un ejemplo. El siguiente esquema

```
\begin{schema}{StartContext}
\Delta IDStation
\\ RealWorldChanges
\\ \Xi Config
```

```
\\ \Xi KeyStore
\\ InitDoorLatchAlarm'
\\ InitStats'
\\ InitAdmin'
\\ \Xi UserToken
\\ \Xi AdminToken
\\ \Xi Finger
\\ \Xi Floppy
\\ \Xi Keyboard
\end{schema}
```

se remplazó por

```
\begin{schema}{StartContext}
\Delta IDStation
\\ RealWorldChanges
\\ \Xi Config
\\ \Xi KeyStore
\\ InitDoorLatchAlarm~'
\\ InitStats~'
\\ InitAdmin~'
\\ \Xi UserToken
\\ \Xi AdminToken
\\ \Xi Finger
\\ \Xi Floppy
\\ \Xi Keyboard
\end{schema}
```

4.4. Palabra reservada *id*

En Tokeneer, *id* se usa como variable pero es una palabra reservada del lenguaje Z que indica la relación identidad. Se remplazaron todas las ocurrencias de *id* por *iden*. Por ejemplo:

Certificate

$id : CertificateId$
$validityPeriod : \mathbb{P}\ TIME$
$isValidatedBy : KEYPART$

se remplazó por

Certificate

$iden : CertificateId$
$validityPeriod : \mathbb{P}\ TIME$
$isValidatedBy : \mathbb{P}\ KEYPART$

4.5. Conjunto finitos

Los conjuntos finitos no son soportados por Fastest^{TM} en su versión actual. Las 45 ocurrencias de este tipo de conjunto en Tokeneer se remplazaron por conjuntos infinitos. Esta restricción no produce ninguna diferencia en la derivación de casos de prueba.

Un ejemplo de este remplazo se muestra a continuación:

AuditLog

$$auditLog : \mathbb{F}\ Audit$$
$$auditAlarm : ALARM$$

AuditLog

$$auditLog : \mathbb{P}\ Audit$$
$$auditAlarm : ALARM$$

4.6. Relación inversa

Todas las ocurrencias de la relación inversa ~ se encerraron entre paréntesis para que el Latex compile. Por ejemplo:

EnrolmentDataOKInv

$$EnrolmentDataOK$$
$$KeyStoreInv$$
$$enrolmentFile : ValidEnrol \pfun FLOPPY$$

$$currentFloppy \in \mathrm{ran}\ enrolmentFile$$
$$(\exists\ ValidEnrol \bullet \theta\, ValidEnrol = (enrolmentFile^{\sim})\, currentFloppy)$$

4.7. Expresiones condicionales

Fastest^{TM} no soporta la expresión condicional **if then else**. Existían 7 ocurrencias de esta expresión en la especificación de Tokeneer. Se hizo la transformación dividiendo el esquema que contenía la expresión en dos esquemas con el siguiente criterio:

- Sea el esquema S con la expresión **if** *cond* **then** *val1* **else** *val2*.
- Crear 2 esquemas S1 y S2 iguales a S.
- En S1, reemplazar la expresión **if then else** por *cond* y usar *val1*.
- En S2, reemplazar la expresión **if then else** por $\neg$ *cond* y usar *val2*.
- Finalmente, remplazar la definición de S por $S == S1 \lor S2$.

Por ejemplo:

PollUserToken

$\Delta UserToken$
$RealWorld$

$userTokenPresence' = present \Leftrightarrow userToken \neq noT$
$currentUserToken' = \textbf{if } userToken \neq noT \textbf{ then } userToken \textbf{ else } currentUserToken$

se remplazó por

*PollUserToken*1

$\Delta UserToken$
$RealWorld$

$userTokenPresence' = present \Leftrightarrow userToken \neq noT$
$userToken \neq noT$
$currentUserToken' = userToken$

*PollUserToken*2

$\Delta UserToken$
$RealWorld$

$userTokenPresence' = present \Leftrightarrow userToken \neq noT$
$userToken = noT$
$currentUserToken' = currentUserToken$

$PollUserToken == PollUserToken1 \lor PollUserToken2$

Cuando existiera más de un **if**, el esquema se divide en 2 por la cantidad de **if** existentes.

4.8. Definiciones axiomáticas

Fastest™ no soporta definiciones axiomáticas. Todas las definiciones axiomáticas de Tokeneer se remplazaron por esquemas, excepto la definición de *ISSUER*.

Abajo se presenta un ejemplo de remplazo de una definición axiomática por un esquema:

$minClearance : Clearance \times Clearance \rightarrow Clearance$

MinClearance

$minClearance : Clearance \times Clearance \rightarrow Clearance$

Donde se usa la función *minClearance*, se agregó el esquema *MinClearance*.

Para *ISSUER*, se borró la siguiente definición:

$ISSUER : \mathbb{P}\ USER$

y se agregó a cada esquema que la usa. Son estos dos:

CertificateId

$issuer : ISSUER$

remplazado por

CertificateId

$ISSUER : \mathbb{P}\ USER$
$issuer : USER$

$issuer \in ISSUER$

y

KeyStore

$issuerKey : ISSUER \pfun KEYPART$
$ownName : \mathbb{F}\ ISSUER$
$theISSUER : \mathbb{F}\ ISSUER \fun ISSUER$

$ownName \neq \varnothing \Rightarrow theISSUER\ ownName \in \mathrm{dom}\ issuerKey$
$\#ownName \leq 1$

remplazado por

KeyStore

$ISSUER : \mathbb{P}\ USER$
$issuerKey : USER \pfun KEYPART$
$ownName : \mathbb{F}\ USER$
$theISSUER : \mathbb{F}\ USER \fun USER$

$ownName \neq \varnothing \Rightarrow theISSUER\ ownName \in \mathrm{dom}\ issuerKey$
$\#ownName \leq 1$
$\mathrm{dom}\ issuerKey = ISSUER$
$ownName \subseteq ISSUER$
$\mathrm{dom}\ theISSUER \subseteq \mathbb{F}\ ISSUER$
$\mathrm{ran}\ theISSUER = ISSUER$

4.9. Definición de tipos opcionales

Para poder definir items opcionales, en Tokeneer se hicieron las siguientes definciones:

$optional\ X == \{x : \mathbb{F}\ X \mid \#x \leq 1\}$
$nil[X] == \varnothing[X]$
$the[X] == \{\ x : X \bullet \{x\} \mapsto x\}$

Estas definiciones no son soportadas por Fastest^{TM}. Las variables de estos tipos se adaptaron de la siguiente manera:

$x : optional\ Y$ se remplazó por $x : \mathbb{P}\ Y \mid \#x \leq 1$ en la definición de cada esquema.

Por ejemplo, la declaración de la variable *isValidatedBy*

Certificate

$id : CertificateId$
$validityPeriod : \mathbb{P}\ TIME$
$isValidatedBy : optional\ KEYPART$

se remplazó por

Certificate

$iden : CertificateId$
$validityPeriod : \mathbb{P}\ TIME$
$isValidatedBy : \mathbb{P}\ KEYPART$

$\#isValidatedBy \leq 1$

$x = nil$ se remplazó por $x = \varnothing$ en la definición de cada esquema.

Por ejemplo:

AdminLogout

$\Delta Admin$

$rolePresent \neq nil$
$rolePresent' = nil$
$currentAdminOp' = nil$

se remplazó por

AdminLogout

$\Delta Admin$

$rolePresent \neq \varnothing$
$rolePresent' = \varnothing$
$currentAdminOp' = \varnothing$

the $x : Y$ se remplazó por $the : \mathbb{P}\ Y \rightarrow Y$ en la definición de cada esquema.

La función *the* tiene 40 ocurrencias aproximadamente en la especificación de Tokeneer. Para que no haya conflicto en la herencia, hubo que definir una función *the* por cada tipo. Muchos de los esquemas que usan la función *the* de un tipo en particular, la heredan de un esquema incluido. En esos casos, no hizo falta definir la función nuevamente. En los tres esquemas zed (cálculo de esquemas) que usan *the*, tampoco hizo falta definirlos porque los heredan.

Los esquemas donde se definió la función *the* son los siguientes (solo se incluyen las declaraciones):

AdminLogon

$\Delta Admin$
$AdminToken$
$theAuthCert : \mathbb{F}\ AuthCert \rightarrow AuthCert$

AddAuthCertToUserToken

$\Delta UserToken$
$KeyStore$
$Config$
$currentTime : TIME$
$theAuthCert : \mathbb{F}\ AuthCert \rightarrow AuthCert$

Admin

$rolePresent : \mathbb{F}\ ADMINPRIVILEGE$
$availableOps : \mathbb{P}\ ADMINOP$
$currentAdminOp : \mathbb{F}\ ADMINOP$
$theADMINPRIVILEGE : \mathbb{F}\ ADMINPRIVILEGE \rightarrow ADMINPRIVILEGE$
$theADMINOP : \mathbb{F}\ ADMINOP \rightarrow ADMINOP$

KeyStore

$issuerKey : ISSUER \nrightarrow KEYPART$
$ownName : \mathbb{F}\ ISSUER$
$theISSUER : \mathbb{F}\ ISSUER \rightarrow ISSUER$

ValidEnrol

$Enrol$
$theKEYPART : \mathbb{F}\ KEYPART \rightarrow KEYPART$

TokenWithValidAuth

$Token$
$theAuthCert : \mathbb{F}\ AuthCert \rightarrow AuthCert$

Debajo se incluye un ejemplo completo con declaraciones y restricciones de como se remplazaron los esquemas que usan *the*

TokenWithValidAuth

$Token$

$authCert \neq$
$\quad \wedge\ (the\ authCert).tokenID = tokenID$
$\quad \wedge\ (the\ authCert).baseCertId = idCert.id$

se remplazó por

TokenWithValidAuth

$$Token$$
$$theAuthCert : \mathbb{P}\, AuthCert \rightarrow AuthCert$$

$$authCert \neq \varnothing$$
$$\wedge\ (theAuthCert\ authCert).tokenID = tokenID$$
$$\wedge\ (theAuthCert\ authCert).baseCertId = idCert.iden$$

Al agregar la definición de *the* dentro de los esquemas, se generaba el siguiente conflicto. La función *goodT* toma un *Token*. En la especificación original, se utilizaba también con *TokenWithValidAuth*, que incluye solo a *Token*. Al agregar la definición de *the* a *TokenWithValidAuth*, se generaba un conflicto de tipos al aplicar la función *goodT* sobre *TokenWithValidAuth*. Entonces se definió una segunda función *goodThe*, que toma un *TokenWithValidAuth*, y para ese parámetro se remplazaron todas las ocurrencias de *goodT* por *goodThe*:

Esto es, se remplazó

$$TOKENTRY ::= noT \mid badT \mid goodT \langle\langle Token \rangle\rangle$$

por

$$TOKENTRY ::= noT \mid badT \mid goodT \langle\langle Token \rangle\rangle \mid goodThe \langle\langle TokenWithValidAuth \rangle\rangle$$

y

$$goodT\,(\theta\, TokenWithValidAuth)$$

por

$$goodThe\,(\theta\, TokenWithValidAuth)$$

4.10. Inclusión de esquemas en el predicado

Al correr el verificador de tipos de CZT, se presentan en la especificación original 7 ocurrencias de error al incluir esquemas en el predicado de un esquema. Todas estas ocurrencias se dan por el remplazo de definiciones axiomáticas por esquemas.

A continuación se muestra la solución con un ejemplo. En el siguiente esquema:

AddAuthCertToUserToken

$\Delta UserToken$
$KeyStore$
$Config$
$currentTime : TIME$
$theAuthCert : \mathbb{P}\, AuthCert \rightarrow AuthCert$
$goodT : Token \nrightarrow TOKENTRY$

$userTokenPresence = present$
$currentUserToken \in \operatorname{ran} goodT$
$\exists\, ValidToken;\ ValidToken\,' \bullet$
$\quad \theta\, ValidToken = ((goodT^{\sim})\, currentUserToken)$
$\quad \land\ \theta\, ValidToken\,' = ((goodT^{\sim})\, currentUserToken')$
$\quad \land\ (\exists\, newAuthCert : AuthCert \bullet$
$\qquad theAuthCert\ authCert' = newAuthCert \land NewAuthCert)$
$\quad \land\ tokenID' = tokenID$
$\quad \land\ idCert' = idCert$
$\quad \land\ privCert' = privCert$
$\quad \land\ iandACert' = iandACert$
$userTokenPresence' = userTokenPresence$

el error se daba en la sentencia

$\exists\, newAuthCert : AuthCert$

por el esquema *NewAuthCert* que usa una función (*minClearance*) de un esquema de los que remplazan definiciones axiomáticas. Entonces se agregó la definición de la función en el cuantificador:

$\exists\, newAuthCert : AuthCert;$
$minClearance : Clearance \times Clearance \rightarrow Clearance$

AddAuthCertToUserToken

$\Delta UserToken$
$KeyStore$
$Config$
$currentTime : TIME$
$theAuthCert : \mathbb{P}\, AuthCert \rightarrow AuthCert$
$goodT : Token \nrightarrow TOKENTRY$

$userTokenPresence = present$
$currentUserToken \in \operatorname{ran} goodT$
$\exists\, ValidToken;\ ValidToken\,' \bullet$
$\quad \theta\, ValidToken = ((goodT^{\sim})\, currentUserToken)$
$\quad \land\ \theta\, ValidToken\,' = ((goodT^{\sim})\, currentUserToken')$
$\quad \land\ (\exists\, newAuthCert : AuthCert;$
$\qquad minClearance : Clearance \times Clearance \rightarrow Clearance \bullet$
$\qquad\quad theAuthCert\ authCert' = newAuthCert \land NewAuthCert)$
$\quad \land\ tokenID' = tokenID$
$\quad \land\ idCert' = idCert$
$\quad \land\ privCert' = privCert$
$\quad \land\ iandACert' = iandACert$
$userTokenPresence' = userTokenPresence$

Un remplazo similar se hizo en el esquema *NoChange*:

NoChange

$\Delta IDStation$

$currentDoor = currentDoor'$
$currentLatch' = currentLatch$
$doorAlarm = doorAlarm'$
$auditAlarm = auditAlarm'$
$currentDisplay' = currentDisplay$
$currentScreen'.screenMsg = currentScreen.screenMsg$
$AddElementsToLog \lor \Xi AuditLog$

NoChange

$\Delta IDStation$
$sizeElement : Audit \to \mathbb{N}$
$maxSupportedLogSize : \mathbb{N}$
$sizeLog : \mathbb{F}\, Audit \to \mathbb{N}$
$oldestLogTime : \mathbb{F}\, Audit \to TIME$
$newestLogTime : \mathbb{F}\, Audit \to TIME$

$currentDoor = currentDoor'$
$currentLatch' = currentLatch$
$doorAlarm = doorAlarm'$
$auditAlarm = auditAlarm'$
$currentDisplay' = currentDisplay$
$currentScreen'.screenMsg = currentScreen.screenMsg$
$AddElementsToLog \lor \Xi AuditLog$

y en los restantes esquemas que daban error al incluir esquemas en el predicado.

4.11. Operador *Theta*

La versión utilizada de Fastest™ no soporta el operador *Theta*. El uso de este operador en poscondiciones no da error porque las poscodiciones no se usan para derivar casos de prueba. El problema se da al expandir precondiciones con este operador.

Hay 23 ocurrencias de *Theta* en Tokeneer: 18 en invariantes, 2 en poscondiciones y 3 en precondiciones dentro de cuantificadores.

En el esquema *IDStation* existen 5 ocurrencias de Theta. Este esquema es un invariante de estado que no deriva casos casos de prueba, por lo tanto no da error en Fastest™ y no hace falta hacer ningún remplazo. Lo mismo ocurre con los esquemas *UserTokenWithOKAuthCert*, *UserTokenOK*, *UserAllowedEntry*, *EnrolmentDataOK* y *AdminTokenOK*. En el esquema *AddAuthCertToUserToken* existen 2 ocurrencias de Theta. Una en una poscondición y una en una precondición pero dentro de un cuantificador, que tampoco presenta errores. Los esquemas *UpdateKeyStoreFromFloppy* y *AdminLogon* usan el operador *Theta* en una precondición dentro de un cuantificador también. El esquema *FinishUpdateConfigDataOK* presenta una ocurrencia de *Theta* en una poscondición y como se mencionó arriba, no presenta problemas para Fastest™

4.12. Composición de esquemas

Fastest™ no soporta la composición de esquemas a esta altura. Hay tres esquemas que usan composición (;) en Tokeneer. Se comentaron ya que no agregan funcionalidad y los casos de prueba se pueden derivar de los módulos que integran la composición. Uno de estos esquemas es:

$$TISWriteUserToken == (WriteUserToken \fatsemi UpdateUserToken) \\ \lor [\ UserTokenTorn \mid status = waitingUpdateToken\ ]$$

Como se incluye en este otro esquema, también hubo que comentarlo, pero con el mismo argumento pueden derivarse casos de cada uno de los esquemas que lo componen:

$$TISUserEntryOp == \\ TISReadUserToken \\ \lor\ TISValidateUserToken \\ \lor\ TISReadFinger \\ \lor\ TISValidateFinger \\ \lor\ TISWriteUserToken \\ \lor\ TISValidateEntry \\ \lor\ TISUnlockDoor \\ \lor\ TISCompleteFailedAccess$$

Los siguientes esquemas se comentaron por las mismas razones:

$$StartArchiveLog == \\ (StartArchiveLogOK \fatsemi UpdateFloppy) \\ \lor\ StartArchiveLogWaitingFloppy \\ \lor [\ BadAdminLogout \mid \\ enclaveStatus = waitingStartAdminOp \\ \land\ theADMINOP\ currentAdminOp = archiveLog\ ]$$

$$ClearLogThenAddElements == ClearLog \fatsemi AddElementsToLog$$

$$TISArchiveLogOp == StartArchiveLog \lor FinishArchiveLog$$

FinishArchiveLogOK

$$AdminOpFinishContext \\ \Xi Config \\ \Xi Floppy \\ \Xi DoorLatchAlarm$$

$$theADMINOP\ currentAdminOp = archiveLog \\ floppyPresence = present \\ writtenFloppy = currentFloppy \\ (\exists\ archive : \mathbb{F}\ Audit;\ sizeElement : Audit \to \mathbb{N};\ sizeLog : \mathbb{F}\ Audit \to \mathbb{N}; \\ \quad oldestLogTime : \mathbb{F}\ Audit \to TIME;\ newestLogTime : \mathbb{F}\ Audit \to TIME \bullet \\ \quad\quad ClearLogThenAddElements \land writtenFloppy = auditFile\ archive) \\ currentScreen'.screenMsg = requestAdminOp$$

$$
\begin{array}{l}
FinishArchiveLogFail == \\
\quad FinishArchiveLogBadMatch \\
\quad \lor\ FinishArchiveLogNoFloppy \\
FinishArchiveLog == \\
\quad FinishArchiveLogOK \\
\quad \lor\ FinishArchiveLogFail \\
\quad \lor\ [\ BadAdminLogout\ | \\
\quad enclaveStatus = waitingFinishAdminOp \\
\quad \land\ theADMINOP\ currentAdminOp = archiveLog\ ]
\end{array}
$$

$$
\begin{array}{l}
TISAdminOp == \\
\quad TISOverrideDoorLockOp \\
\quad \lor\ TISShutdownOp \\
\quad \lor\ TISUpdateConfigDataOp \\
\quad \lor\ TISArchiveLogOp
\end{array}
$$

$$
\begin{array}{l}
TISProcessing == (\\
\quad TISEnrolOp \\
\quad \lor\ TISUserEntryOp \\
\quad \lor\ TISAdminLogon \\
\quad \lor\ TISStartAdminOp \\
\quad \lor\ TISAdminOp \\
\quad \lor\ TISAdminLogout \\
\quad \lor\ TISIdle) \\
\quad \land\ LogChange
\end{array}
$$

4.13. Tipos inductivos

Los tipos libres inductivos se adaptaron para ser simples tipos enumerados ya que hasta el momento Fastest™ no soporta los primeros. Para esta adaptación se quitó el parámetro del tipo inductivo y se remplazó la función por una variable. A continuación se listan los tipos que usan inducción, indicando previamente que uso se da a sus funciones:

keyedOps usa la imagen relacional 3 veces y la inversa 1 vez, haciendo que su reemplazo sea más complejo.

$$KEYBOARD ::= noKB \mid badKB \mid keyedOps\langle\langle ADMINOP\rangle\rangle$$

goodFP usa el rango 1 vez (remplazo complejo).

$$FINGERPRINTTRY ::= noFP \mid badFP \mid goodFP\langle\langle FINGERPRINT\rangle\rangle$$

goodT usa theta 4 veces, el rango 4 veces y la inversa 2 veces. Su remplazo es complejo. Distinto es el caso de la función *goodThe*, que solo usa *Theta* (5 veces) y tiene un remplazo simple.

$$
\begin{array}{l}
TOKENTRY ::= noT \mid badT \mid goodT\langle\langle Token\rangle\rangle \mid \\
\quad goodThe\langle\langle TokenWithValidAuth\rangle\rangle
\end{array}
$$

enrolmentFile usa la imagen relacional 2 veces y la inversa 2 veces (remplazo complejo). *auditFile* aplica la función una vez (remplazo simple). *configFile* usa la imagen relacional 1 vez y la inversa 2 veces (remplazo complejo).

$$
\begin{aligned}
&FLOPPY ::= noFloppy \mid emptyFloppy \mid badFloppy \mid \\
&\quad enrolmentFile\langle\!\langle ValidEnrol \rangle\!\rangle \mid \\
&\quad auditFile\langle\!\langle \mathbb{F}\, Audit \rangle\!\rangle \mid \\
&\quad configFile\langle\!\langle Config \rangle\!\rangle
\end{aligned}
$$

displayStats y *displayConfigData* usan theta 1 vez (remplazo simple).

$$
\begin{aligned}
&SCREENTEXT ::= clear \mid welcomeAdmin \mid busy \mid removeAdminToken \mid closeDoor \mid \\
&\quad requestAdminOp \mid doingOp \mid invalidRequest \mid invalidData \mid \\
&\quad insertEnrolmentData \mid validatingEnrolmentData \mid enrolmentFailed \mid \\
&\quad archiveFailed \mid insertBlankFloppy \mid insertConfigData \mid \\
&\quad displayStats\langle\!\langle Stats \rangle\!\rangle \mid displayConfigData\langle\!\langle Config \rangle\!\rangle
\end{aligned}
$$

En los casos identificados como de remplazo simple, solo hay que eliminar el parametro.

Por ejemplo:

$$
\begin{aligned}
&SCREENTEXT ::= clear \mid welcomeAdmin \mid busy \mid removeAdminToken \mid closeDoor \mid \\
&\quad requestAdminOp \mid doingOp \mid invalidRequest \mid invalidData \mid \\
&\quad insertEnrolmentData \mid validatingEnrolmentData \mid enrolmentFailed \mid \\
&\quad archiveFailed \mid insertBlankFloppy \mid insertConfigData \mid \\
&\quad displayStats\langle\!\langle Stats \rangle\!\rangle \mid displayConfigData\langle\!\langle Config \rangle\!\rangle
\end{aligned}
$$

se remplaza por

$$
\begin{aligned}
&SCREENTEXT ::= clear \mid welcomeAdmin \mid busy \mid removeAdminToken \mid closeDoor \mid \\
&\quad requestAdminOp \mid doingOp \mid invalidRequest \mid invalidData \mid \\
&\quad insertEnrolmentData \mid validatingEnrolmentData \mid enrolmentFailed \mid \\
&\quad archiveFailed \mid insertBlankFloppy \mid insertConfigData \mid \\
&\quad displayStats \mid displayConfigData
\end{aligned}
$$

En los casos complejos, se remplaza el tipo parametrizado por una ocurrencia por elemento del dominio de la función en la definición, agregando la definición de la función en cada esquema que lo usa y una restricción indicando el mapeo:

Sean

$$
\begin{aligned}
&R = a \mid b \mid c \\
&S = e \mid f \mid g << S >>
\end{aligned}
$$

remplazo S por:

$$S = e \mid f \mid ga \mid gb \mid gc$$

y en el equema que se usa agrego:

$$
\begin{aligned}
&g : R \pfun S \mid \\
&g = \{a \mapsto ga, b \mapsto gb, c \mapsto gc\}
\end{aligned}
$$

Por ejemplo, para *keyedOps*:

$$KEYBOARD ::= noKB \mid badKB \mid keyedOps\langle\langle ADMINOP\rangle\rangle$$

se remplaza por:

$$KEYBOARD ::= noKB \mid badKB \mid keyedOpsArchiveLog \mid keyedOpsUpdateConfigData \mid keyedOpsOverrideLock \mid keyedOpsShutdownOp$$

y en el esquema que lo usa agrego:

$$\begin{array}{|l}
\hline
.. \\
keyedOps : ADMINOP \pfun KEYBOARD \\
\hline
keyedOps = \{archiveLog \mapsto keyedOpsArchiveLog, \\
\quad updateConfigData \mapsto keyedOpsUpdateConfigData, \\
\quad overrideLock \mapsto keyedOpsOverrideLock, \\
\quad shutdownOp \mapsto keyedOpsShutdownOp\} \\
... \\
\hline
\end{array}$$

Ésto en el caso de parámetros de tipo enumerado. Cuando se usan parámetros de tipo básico, se remplaza la definición del tipo básico por un tipo enumerado y se usa el mismo criterio.

Ejemplo: se remplaza

$$[FINGERPRINT]$$

por

$$FINGERPRINT ::= FP1 \mid FP2 \mid FP3$$

y

$$FINGERPRINTTRY ::= noFP \mid badFP \mid goodFP\langle\langle FINGERPRINT\rangle\rangle$$

por

$$FINGERPRINTTRY ::= noFP \mid badFP \mid goodFP1 \mid goodFP2 \mid goodFP3$$

usándose de la siguiente manera:

$$\begin{array}{|l}
\hline
FingerOK \\
\hline
Finger \\
UserToken \\
goodFP : FINGERPRINT \pfun FINGERPRINTTRY \\
\hline
goodFP = \{FP1 \mapsto goodFP1, FP2 \mapsto goodFP2, FP3 \mapsto goodFP3\} \\
currentFinger \in \operatorname{ran} goodFP \\
\hline
\end{array}$$

Para este caso particular, este esquema se incluye en el el predicado de otro, *ValidateFingerOK*. *goodFP* da error de identificador no declarado en este caso, así que se incluye la definición aquí también.

En los otros casos el parámetro es un esquema. Para éstos, se remplaza la función por tres ocurrencias en la definición del tipo y se agrega la definición de la función en cada esquema que los usa.

Ejemplo:

$$TOKENTRY ::= noT \mid badT \mid goodT1 \mid goodT2 \mid goodT3 \mid goodThe$$

AdminTokenOKInv

$$\begin{array}{l} AdminTokenOK \\ KeyStoreInv \\ goodT : Token \pfun TOKENTRY \\ \hline currentAdminToken \in \operatorname{ran} goodT \\ \exists\, TokenWithValidAuth \bullet \\ \quad (goodThe = currentAdminToken \\ \quad \land\, (\exists\, IDCert \bullet \theta IDCert = idCert \land CertOK) \\ \quad \land\, (\exists\, AuthCert \bullet \theta AuthCert = theAuthCert\ authCert \land AuthCertOK) \\ \quad \land\, (theAuthCert\ authCert).role \in ADMINPRIVILEGE \\ \quad \land\, currentTime \in (theAuthCert\ authCert).validityPeriod) \end{array}$$

4.14. Tipo esquema

El tipos esquema no está soportado en Fastest™. El esquema *Screen* solo define un tupla que se puede remplazar por un producto cartesiano, y luego remplazar la aplicación de las funciones proyección por los números de la tupla.

Screen

$$\begin{array}{l} screenStats : SCREENTEXT \\ screenMsg : SCREENTEXT \\ screenConfig : SCREENTEXT \end{array}$$

se remplazó por:

$$Screen == SCREENTEXT \times SCREENTEXT \times SCREENTEXT$$

y por ejemplo:

$$screen'.screenConfig = clear$$

se remplazó por

$$screen'.3 = clear$$

4.15. Errores al cargar la especificación en Fastest™

Al intentar cargar la especificación por partes, a diferencia de CZT, Fastest™ arroja errores al encontrar definiciones de tipos después de su uso. Por lo tanto, se modificó el orden de la especificación para que todas las definiciones de tipo se hagan al principio y las demás en orden.

Otro error que se presentó fue con el siguiente tipo de esquema:

$$
\begin{aligned}
&TISEarlyUpdate == UpdateLatch \land UpdateAlarm \\
&\quad \land [\ RealWorldChanges \mid screen' = screen \land display' = display\ ] \\
&\quad \land [\Delta IDStation \mid currentDisplay = currentDisplay'] \\
&\quad \land \Xi UserToken \land \Xi AdminToken \land \Xi Finger \\
&\quad \land \Xi Floppy \land \Xi Keyboard \land \Xi Config \land \Xi Stats \\
&\quad \land \Xi KeyStore \land \Xi Admin \land \Xi Internal \\
&\quad \land (AddElementsToLog \lor \Xi AuditLog)
\end{aligned}
$$

Hubo que reemplazarlo de la siguiente forma para que Fastest™ lo interprete correctamente:

$$
\begin{aligned}
&TISEarlyUpdate == UpdateLatch \land UpdateAlarm \\
&\quad \land [\ RealWorldChanges \mid screen' = screen \land display' = display\ ] \\
&\quad \land [\Delta IDStation \mid currentDisplay = currentDisplay'] \\
&\quad \land [\Xi UserToken;\ \Xi AdminToken;\ \Xi Finger;\ \Xi Floppy; \\
&\quad \Xi Keyboard;\ \Xi Config;\ \Xi Stats; \\
&\quad \Xi KeyStore;\ \Xi Admin;\ \Xi Internal; \\
&\quad (AddElementsToLog \lor \Xi AuditLog) \mid true]
\end{aligned}
$$

Este remplazo se hizo en 2 esquemas. El uso del esquema *true* provoca que Fastest™ arroje el siguiente error inofensivo al cargar la especificación:

```
Fastest> loadspec t.tex
Loading specification..
Schema no encontrado!
Schema no encontrado!
Specification loaded.
```

4.16. Invariantes de estado

A pesar de no presentan un error para Fastest™ se remplazaron todos los esquemas de estado siguiendo un estilo de inclusión de invariantes usado en el lenguaje formal TLA y no el normalmente usado en especificaciones Z. Éste consiste en definir un esquema de estado normalizado sin ningún predicado y un esquema de estado que incluye al esquema original y que contiene su invariante. Los esquemas de operación no incluyen el esquema del invariante. Todas las precondiciones son explícitas.

Ejemplo:

$$
\begin{array}{|l}
\hline
CAIdCert \\
IDCert \\
\hline
isValidatedBy = \{\ subjectPubK\ \} \\
\hline
\end{array}
$$

se remplazó por:

$$
\begin{array}{|l}
\hline
CAIdCert \\
IDCert \\
\hline
\end{array}
$$

CAIdCertInv

CAIdCert

$isValidatedBy = \{\ subjectPubK\ \}$

Capítulo 5

Testing de Tokeneer en Fastest^{TM}

Una vez realizadas todas las adaptaciones necesarias para que la especificación de Tokeneer pase la verificación de tipos de CZT y cargue exitosamente en Fastest^{TM} se procedió a realizar la derivación de casos de prueba.

La versión actual de Fastest^{TM} no hace poda automática del árbol de pruebas, pero se dispone de un script que utiliza la herramienta de chequeo de tipos ZEVES para hacer estas podas, como se detalla en la sección 5.8. ZEVES es un verificador de tipos distribuido por la empresa ORA Canada hasta el año 2005, basado en el estilo de Z de Spivey [4]. En las primeras secciones de este capítulo se analizan los árboles de prueba obtenidos para cada operación y se explica como podar los árboles manualmente.

Para poder cargar la especificación de Tokeneer en la actual versión de Fastest^{TM} sin que dé errores de memoria, la herramienta debe lanzarse con los siguientes parámetros:

```
java -Xss8M -Xms512m -Xmx1024m -jar fastest.jar
```

Se ejecutó el siguiente comando para obtener una lista de todas las operaciones presentes en la especificación:

```
showloadedops
```

$RealWorldChanges$
$PollTime$
$PollDoor$
$PollUserToken1$
$PollUserToken2$
$PollUserToken$
$PollAdminToken1$
$PollAdminToken2$
$PollAdminToken$
$PollFinger1$
$PollFinger2$
$PollFinger$
$PollFloppy1$
$PollFloppy2$

PollFloppy
*PollKeyboard*1
*PollKeyboard*2
PollKeyboard
UpdateLatch
UpdateAlarm
UpdateDisplay
*UpdateScreen*1
*UpdateScreen*2
*UpdateScreen*3
*UpdateScreen*4
UpdateScreen
UpdateUserToken
UpdateFloppy
AddElementsToLog
TISEarlyUpdate
TISUpdate
ArchiveLog
ClearLog
AuditDoor
AuditLatch
AuditAlarm
AuditLogAlarm
AuditDisplay
AuditScreen
NoChange
LogChange
TISPoll
AddSuccessfulEntryToStats
AddFailedEntryToStats
AddSuccessfulBioCheckToSt
AddFailedBioCheckToStats
UnlockDoor
LockDoor
AddAuthCertToUserToken
UpdateKeyStore
UpdateKeyStoreFromFloppy
AdminLogon
AdminLogout
AdminStartOp
AdminFinishOp
ResetScreenMessage
UserEntryContext
UserTokenTorn
ReadUserToken
TISReadUserToken
BioCheckNotRequired
BioCheckRequired
ValidateUserTokenOK
ValidateUserTokenFail

TISValidateUserToken
ReadFingerOK
NoFinger
FingerTimeout
TISReadFinger
ValidateFingerOK
ValidateFingerFail
TISValidateFinger
WriteUserTokenOK
WriteUserTokenFail
WriteUserToken
EntryOK
EntryNotAllowed
TISValidateEntry
UnlockDoorOK
WaitingTokenRemoval
TokenRemovalTimeout
TISUnlockDoor
FailedAccessTokenRemoved
TISCompleteFailedAccess
EnclaveContext
EnrolContext
RequestEnrolment
ReadEnrolmentFloppy
ReadEnrolmentData
ValidateEnrolmentDataOK
ValidateEnrolmentDataFail
ValidateEnrolmentData
FailedEnrolFloppyRemoved
WaitingFloppyRemoval
CompleteFailedEnrolment
TISEnrolOp
AdminTokenTear
BadAdminTokenTear
BadAdminLogout
LoginAborted
LoginContext
ReadAdminToken
TISReadAdminToken
ValidateAdminTokenOK
ValidateAdminTokenFail
TISValidateAdminToken
FailedAdminTokenRemoved
WaitingAdminTokenRemoval
TISCompleteFailedAdminLog
TISAdminLogon
TokenRemovedAdminLogout
AdminTokenTimeout
TISCompleteTimeoutAdminLo
TISAdminLogout

AdminOpContext
AdminOpStartedContext
AdminOpFinishContext
StartOpContext
ValidateOpRequestOK
ValidateOpRequestFail
NoOpRequest
ValidateOpRequest
TISStartAdminOp
StartArchiveLogOK
StartArchiveLogWaitingFlo
FinishArchiveLogNoFloppy
FinishArchiveLogBadMatch
StartUpdateConfigOK
StartUpdateConfigWaitingF
StartUpdateConfigData
FinishUpdateConfigDataOK
FinishUpdateConfigDataFai
FinishUpdateConfigData
TISUpdateConfigDataOp
ShutdownOK
ShutdownWaitingDoor
TISShutdownOp
OverrideDoorLockOK
TISOverrideDoorLockOp
StartContext
StartNonEnrolledStation
StartEnrolledStation
TISStartup
TISIdle

Las 144 operaciones listadas son las posibles candidatas para derivar casos de prueba.

5.1. Operaciones con solo poscondiciones

Los tres primeros esquemas, *RealWorldChanges*, *PollTime* y *PollDoor* solo contienen poscondiciones, por lo cual no se pueden derivar casos de prueba de estas operaciones. Cuando se tratan de generar casos con Fastest^{TM} para este tipo de esquemas, el resultado es un árbol vacío. Ejemplo:

```
Fastest> selop RealWorldChanges
Fastest> genalltt
Generating test tree for 'RealWorldChanges' operation..
Fastest> showtt

RealWorldChanges_VIS
```

Las siguientes 25 operaciones presentan el mismo comportamiento, creando un total de 28 esquemas que no derivan casos de prueba, por lo tanto la lista de posibles candidatos se reduce a

116.

UpdateLatch
UpdateDisplay
UpdateUserToken
UpdateFloppy
AddElementsToLog
ArchiveLog
ClearLog
AuditDoor
AuditLatch
AuditAlarm
AuditLogAlarm
AuditDisplay
AuditScreen
NoChange
AddSuccessfulEntryToStats
AddFailedEntryToStats
AddSuccessfulBioCheckToStats
AddFailedBioCheckToStats
UnlockDoor
LockDoor
ResetScreenMessage
EnclaveContext
LoginContext
AdminOpContext
StartContext

5.2. Operaciones de sondeo

Las siguientes operaciones a testear son *PollUserToken1* y *PollUserToken2*. Éstas son operaciones parciales que se componen en el esquema total *PollUserToken*. Por lo tanto, se derivaran los casos de la operación total.

Para esto, con la especificación cargada, lo primero es seleccionar la operación:

```
selop PollUserToken
```

Las tácticas de testing de Forma Normal Disyuntiva (DNF, por sus siglas en inglés) y Tipos Libres (FT, por sus siglas en inglés) aplican perfectamente a la operación *PollUserToken*, no así la táctica de Particion Estándar (SP, por sus siglas en inglés), ya que no se usa un operador matemático en este esquema.

La táctica DNF se aplica por defecto. La táctica FT se agrega de la siguiente manera sobre la variable *userToken* que es la que se involucra en la precondición de esta operación:

```
addtactic PollUserToken FT userToken
```

Luego se ejecutan las sentencias para generar y visualizar el árbol de prueba:

```
genalltt

showtt
```

```
PollUserToken_VIS
  !______PollUserToken_DNF_1
  |       !______PollUserToken_FT_1
  |       !______PollUserToken_FT_2
  |       !______PollUserToken_FT_3
  |       !______PollUserToken_FT_4
  |       !______PollUserToken_FT_5
  |       !______PollUserToken_FT_6
  |
  !______PollUserToken_DNF_2
  |       !______PollUserToken_FT_7
  |       !______PollUserToken_FT_8
  |       !______PollUserToken_FT_9
  |       !______PollUserToken_FT_10
  |       !______PollUserToken_FT_11
  |       !______PollUserToken_FT_12
  |
  !______PollUserToken_DNF_3
  |       !______PollUserToken_FT_13
  |       !______PollUserToken_FT_14
  |       !______PollUserToken_FT_15
  |       !______PollUserToken_FT_16
  |       !______PollUserToken_FT_17
  |       !______PollUserToken_FT_18
  |
  !______PollUserToken_DNF_4
  |       !______PollUserToken_FT_19
  |       !______PollUserToken_FT_20
  |       !______PollUserToken_FT_21
  |       !______PollUserToken_FT_22
  |       !______PollUserToken_FT_23
  |       !______PollUserToken_FT_24
  |
  !______PollUserToken_DNF_5
  |       !______PollUserToken_FT_25
  |       !______PollUserToken_FT_26
  |       !______PollUserToken_FT_27
  |       !______PollUserToken_FT_28
  |       !______PollUserToken_FT_29
  |       !______PollUserToken_FT_30
  |
  !______PollUserToken_DNF_6
  |       !______PollUserToken_FT_31
  |       !______PollUserToken_FT_32
```

```
  |       !______PollUserToken_FT_33
  |       !______PollUserToken_FT_34
  |       !______PollUserToken_FT_35
  |       !______PollUserToken_FT_36
  |
  !______PollUserToken_DNF_7
  |       !______PollUserToken_FT_37
  |       !______PollUserToken_FT_38
  |       !______PollUserToken_FT_39
  |       !______PollUserToken_FT_40
  |       !______PollUserToken_FT_41
  |       !______PollUserToken_FT_42
  |
  !______PollUserToken_DNF_8
          !______PollUserToken_FT_43
          !______PollUserToken_FT_44
          !______PollUserToken_FT_45
          !______PollUserToken_FT_46
          !______PollUserToken_FT_47
          !______PollUserToken_FT_48
```

Si mostramos los esquemas que corresponden a las clases de test del árbol de prueba con la sentencia:

```
showsch -tcl
```

vemos que la primer clase es una contradicción, por lo tanto puede podarse del árbol de prueba:

```
\begin{schema}{PollUserToken\_ DNF\_ 1}\\
 PollUserToken\_ VIS
\where
 \lnot userToken \neq noT \\
 userToken \neq noT
\end{schema}
```

La clase *PollUserToken_DNF_3* también puede podarse ya que es igual a *PollUserToken_DNF_1*.

Las clases *PollUserToken_DNF_2* y *PollUserToken_DNF_4* son iguales. Por lo tanto, puede podarse una de ellas:

```
\begin{schema}{PollUserToken\_ DNF\_ 2}\\
 PollUserToken\_ VIS
\where
 userToken \neq noT
\end{schema}

\begin{schema}{PollUserToken\_ DNF\_ 4}\\
 PollUserToken\_ VIS
```

```
\where
 userToken \neq noT
\end{schema}
```

Lo mismo para las clases *PollUserToken_DNF_5* y *PollUserToken_DNF_6*.

Las clases *PollUserToken_DNF_3* y *PollUserToken_DNF_8* también son contradicciones así que también se eliminan:

```
\begin{schema}{PollUserToken\_ DNF\_ 7}\\
 PollUserToken\_ VIS
\where
 userToken \neq noT \\
 \lnot userToken \neq noT \\
 userToken = noT
\end{schema}

\begin{verbatim}
\begin{schema}{PollUserToken\_ DNF\_ 8}\\
 PollUserToken\_ VIS
\where
 userToken \neq noT \\
 userToken = noT
\end{schema}
```

La clase *PollUserToken_FT_7* se contradice con la clase *PollUserToken_DNF_2*:

```
\begin{schema}{PollUserToken\_ FT\_ 7}\\
 PollUserToken\_ DNF\_ 2
\where
 userToken = noT
\end{schema}
```

Y las clases *PollUserToken_FT_32*, *PollUserToken_FT_33*, *PollUserToken_FT_34*, *PollUserToken_FT_35* y *PollUserToken_FT_36* se contradicen con la clase *PollUserToken_DNF_6*:

```
\begin{schema}{PollUserToken\_ DNF\_ 6}\\
 PollUserToken\_ VIS
\where
 userToken = noT
\end{schema}

\begin{schema}{PollUserToken\_ FT\_ 32}\\
 PollUserToken\_ DNF\_ 6
\where
 userToken = badT
```

```
\end{schema}

\begin{schema}{PollUserToken\_ FT\_ 33}\\
 PollUserToken\_ DNF\_ 6
\where
 userToken = goodT1
\end{schema}

\begin{schema}{PollUserToken\_ FT\_ 34}\\
 PollUserToken\_ DNF\_ 6
\where
 userToken = goodT2
\end{schema}

\begin{schema}{PollUserToken\_ FT\_ 35}\\
 PollUserToken\_ DNF\_ 6
\where
 userToken = goodT3
\end{schema}

\begin{schema}{PollUserToken\_ FT\_ 36}\\
 PollUserToken\_ DNF\_ 6
\where
 userToken = goodThe
\end{schema}
```

Para hacer estas podas, se ejecutan los siguientes comandos:

```
prunefrom PollUserToken_DNF_1
prunefrom PollUserToken_DNF_3
prunefrom PollUserToken_DNF_4
prunefrom PollUserToken_DNF_5
prunefrom PollUserToken_DNF_7
prunefrom PollUserToken_DNF_8
prunefrom PollUserToken_FT_7
prunefrom PollUserToken_FT_32
prunefrom PollUserToken_FT_33
prunefrom PollUserToken_FT_34
prunefrom PollUserToken_FT_35
prunefrom PollUserToken_FT_36
```

Finalmente se generan los casos de prueba abstractos:

```
Fastest> genalltca
PollUserToken_FT_8 evaluations to perform: at most 1
PollUserToken_FT_8 test case generation -> SUCCESS.
PollUserToken_FT_9 evaluations to perform: at most 1
PollUserToken_FT_9 test case generation -> SUCCESS.
```

```
PollUserToken_FT_11 evaluations to perform: at most 1
PollUserToken_FT_10 evaluations to perform: at most 1
PollUserToken_FT_11 test case generation -> SUCCESS.
PollUserToken_FT_10 test case generation -> SUCCESS.
PollUserToken_FT_12 evaluations to perform: at most 1
PollUserToken_FT_12 test case generation -> SUCCESS.
PollUserToken_FT_31 evaluations to perform: at most 1
PollUserToken_FT_31 test case generation -> SUCCESS.
Elapsed time: 1563 miliseconds.
```

Para mostrar el árbol de prueba, las clases de prueba y los casos de prueba abstractos se ejecutan los siguientes comandos:

```
showtt
showsch -tcl
showsch -tca
```

La salida se muestra en los apéndices B y C.

Las siguientes operaciones de sondeo son similares y los casos y clases de prueba para éstas también se pueden encontrar en los apéndices.

PollAdminToken
PollFinger
PollFloppy
PollKeyboard

En total, entre esquemas parciales y totales, las operaciones de sondeo son 15. Restan probar 101 operaciones.

5.3. Operación de actualización de alarma

Los siguientes comandos se ejecutan sobre la operación *UpdateAlarm*. Los casos y clases de prueba derivados se listan en los apéndices.

```
selop UpdateAlarm
addtactic UpdateAlarm FT doorAlarm
addtactic UpdateAlarm FT auditAlarm
genalltt
prunefrom UpdateAlarm_DNF_2
prunefrom UpdateAlarm_DNF_4
prunefrom UpdateAlarm_FT_2
prunefrom UpdateAlarm_FT_4
prunefrom UpdateAlarm_FT_13
prunefrom UpdateAlarm_FT_15
```

```
prunefrom UpdateAlarm_FT_17
prunefrom UpdateAlarm_FT_27
prunefrom UpdateAlarm_FT_28
prunefrom UpdateAlarm_FT_29
genalltca
showtt
showsch -tcl
showsch -tca

UpdateAlarm_FT_3 evaluations to perform: at most 1
UpdateAlarm_FT_3 test case generation -> SUCCESS.
UpdateAlarm_FT_18 evaluations to perform: at most 1
UpdateAlarm_FT_30 evaluations to perform: at most 1
UpdateAlarm_FT_18 test case generation -> SUCCESS.
UpdateAlarm_FT_30 test case generation -> SUCCESS.
UpdateAlarm_FT_25 evaluations to perform: at most 1
UpdateAlarm_FT_25 test case generation -> SUCCESS.
Elapsed time: 1265 miliseconds.
```

5.4. Operaciones que generan árboles vacíos

Se encuentran algunos esquemas que generan árboles de prueba vacíos al tratar de derivar casos usando la táctica por defecto de Forma Normal Disyuntiva (DNF). Por ejemplo:

```
Fastest> selop AddElementsToLog
Fastest> genalltt
Generating test tree for 'AddElementsToLog' operation..
Fastest> showtt

AddElementsToLog_VIS
```

Los siguientes esquemas presentan el mismo comportamiento:

AddElementsToLog
ArchiveLog
ClearLog
AddAuthCertToUserToken
UpdateKeyStore
UpdateKeyStoreFromFloppy
NoFinger
EnrolContext
ReadAdminToken
FailedAdminTokenRemoved
AdminOpStartedContext
StartOpContext
ValidateOpRequestOK

ValidateOpRequestFail
NoOpRequest
WaitingAdminTokenRemoval
TISReadAdminToken
ValidateAdminTokenOK
ValidateAdminTokenFail
FinishArchiveLogNoFloppy
ShutdownOK
RequestEnrolment
OverrideDoorLockOK
StartNonEnrolledStation
StartEnrolledStation
ReadEnrolmentFloppy
ValidateEnrolmentDataFail
FailedEnrolFloppyRemoved
WaitingFloppyRemoval
TISIdle
AdminLogon
AdminLogout
AdminStartOp
AdminFinishOp
AdminOpFinishContext
StartArchiveLogOK
StartArchiveLogWaitingFloppy
FinishArchiveLogBadMatch
StartUpdateConfigOK
StartUpdateConfigWaitingFloppy
ShutdownWaitingDoor
FinishUpdateConfigDataOK
FinishUpdateConfigDataFail
ValidateEnrolmentDataOK

5.5. Operaciones cuyas clases generan millones de evaluaciones

Al tratar de derivar clases de prueba de la siguiente operación total, *UpdateScreen*, compuesta por las cuatro operaciones parciales *UpdateScreen1*, *UpdateScreen2*, *UpdateScreen3* y *UpdateScreen4*, se presenta un problema:

```
Fastest> selop UpdateScreen
Fastest> genalltt
Generating test tree for 'UpdateScreen' operation..
Fastest> showtt

UpdateScreen_VIS
  !______UpdateScreen_DNF_1
  !______UpdateScreen_DNF_2
```

```
  !______UpdateScreen_DNF_3
  !______UpdateScreen_DNF_4

Fastest> genalltca
UpdateScreen_DNF_4 evaluations to perform: at most 225710080
```

Hay millones de evaluaciones a realizar. Esto se debe a que en el predicado se usan varias funciones. Cuando se combinan los modelos finitos de varias funciones el número crece exponencialmente.

El conjunto finito para todas las variables de tipos estructurados (como funciones, relaciones, conjunto de partes, ect) son generados por los conjuntos finitos de los tipos más básicos, esencialmente haciendo todas las combinaciones posibles entre los elementos de los conjuntos finitos definidos por los tipos básicos. Por ejemplo, el conjunto finito de $\mathbb{P}\,\mathbb{Z}$ será {{}, {-1},{0},{1},{-1,0},{-1,1},{0,1},{-1,0,1}} si el conjunto finito de $\mathbb{Z}$ es {-1,0,1}.

Restringiendo el tamaño del modelo finito de los tipos básicos a 2 o incluso a 1 se debería reducir notablemente el número de combinaciones:

```
Fastest> setfinitemodel UpdateScreen_DNF_1 -btsize 2
Fastest> setfinitemodel UpdateScreen_DNF_2 -btsize 2
Fastest> setfinitemodel UpdateScreen_DNF_3 -btsize 2
Fastest> setfinitemodel UpdateScreen_DNF_4 -btsize 2
Fastest> genalltca
UpdateScreen_DNF_1 evaluations to perform: at most 120932352

Fastest> setfinitemodel UpdateScreen_DNF_1 -btsize 1
Fastest> setfinitemodel UpdateScreen_DNF_2 -btsize 1
Fastest> setfinitemodel UpdateScreen_DNF_3 -btsize 1
Fastest> setfinitemodel UpdateScreen_DNF_4 -btsize 1
Fastest> genalltca
UpdateScreen_DNF_1 evaluations to perform: at most 419904
```

Evidentemente se logra reducir el número de evaluaciones pero siguen siendo mayores a 5000, que no generan casos en un tiempo aceptable.

5.5.1. Casos manejables

Otra operación que presenta un comportamiento similar es *LogChange*, aunque cuando se restringe el tamaño del modelo finito, las evaluaciones se reducen a un número manejable:

```
Fastest> selop LogChange
Fastest> genalltt
Generating test tree for 'LogChange' operation..
Fastest> showtt

LogChange_VIS
```

```
  !______LogChange_DNF_1
  !______LogChange_DNF_2
  !______LogChange_DNF_3

Fastest> setfinitemodel LogChange_DNF_1 -btsize 2
Fastest> setfinitemodel LogChange_DNF_2 -btsize 2
Fastest> setfinitemodel LogChange_DNF_3 -btsize 2

Fastest> genalltca
LogChange_DNF_3 evaluations to perform: at most 2304
LogChange_DNF_3 test case generation -> FAILED.
LogChange_DNF_1 evaluations to perform: at most 2304
LogChange_DNF_1 test case generation -> SUCCESS.
LogChange_DNF_2 evaluations to perform: at most 2304
LogChange_DNF_2 test case generation -> SUCCESS.
Elapsed time: 779015 miliseconds.
Fastest> showtt
Fastest> showsch -tcl
Fastest> showsch -tca
```

Los casos y clases de prueba obtenidos se listan en los apéndices.

De las siguientes operaciones se pueden derivar casos y clases de prueba de manera similar, aunque el tamaño del modelo finito debe ser 1 en la mayoría de los casos para lograr casos en un tiempo manejable. Los mismos se pueden encontrar en los apéndices.

```
Fastest> selop UserEntryContext
Fastest> genalltt
Generating test tree for 'UserEntryContext' operation..
Fastest> showtt

UserEntryContext_VIS
  !______UserEntryContext_DNF_1
  !______UserEntryContext_DNF_2
  !______UserEntryContext_DNF_3
  !______UserEntryContext_DNF_4
  !______UserEntryContext_DNF_5
  !______UserEntryContext_DNF_6
  !______UserEntryContext_DNF_7
  !______UserEntryContext_DNF_8
  !______UserEntryContext_DNF_9
  !______UserEntryContext_DNF_10

Fastest> setfinitemodel UserEntryContext_DNF_1 -btsize 1
Fastest> setfinitemodel UserEntryContext_DNF_2 -btsize 1
Fastest> setfinitemodel UserEntryContext_DNF_3 -btsize 1
Fastest> setfinitemodel UserEntryContext_DNF_4 -btsize 1
Fastest> setfinitemodel UserEntryContext_DNF_5 -btsize 1
Fastest> setfinitemodel UserEntryContext_DNF_6 -btsize 1
```

```
Fastest> setfinitemodel UserEntryContext_DNF_7 -btsize 1
Fastest> setfinitemodel UserEntryContext_DNF_8 -btsize 1
Fastest> setfinitemodel UserEntryContext_DNF_9 -btsize 1
Fastest> setfinitemodel UserEntryContext_DNF_10 -btsize 1
Fastest> genalltca
UserEntryContext_DNF_1 evaluations to perform: at most 64
UserEntryContext_DNF_1 test case generation -> SUCCESS.
UserEntryContext_DNF_10 evaluations to perform: at most 8
UserEntryContext_DNF_10 test case generation -> SUCCESS.
UserEntryContext_DNF_5 evaluations to perform: at most 64
UserEntryContext_DNF_9 evaluations to perform: at most 64
UserEntryContext_DNF_5 test case generation -> SUCCESS.
UserEntryContext_DNF_9 test case generation -> SUCCESS.
UserEntryContext_DNF_6 evaluations to perform: at most 8
UserEntryContext_DNF_6 test case generation -> SUCCESS.
UserEntryContext_DNF_7 evaluations to perform: at most 64
UserEntryContext_DNF_7 test case generation -> SUCCESS.
UserEntryContext_DNF_8 evaluations to perform: at most 8
UserEntryContext_DNF_8 test case generation -> SUCCESS.
UserEntryContext_DNF_3 evaluations to perform: at most 64
UserEntryContext_DNF_3 test case generation -> SUCCESS.
UserEntryContext_DNF_4 evaluations to perform: at most 8
UserEntryContext_DNF_4 test case generation -> SUCCESS.
UserEntryContext_DNF_2 evaluations to perform: at most 8
UserEntryContext_DNF_2 test case generation -> SUCCESS.
Elapsed time: 204671 miliseconds.

Fastest> selop UserTokenTorn
Fastest> genalltt
Generating test tree for 'UserTokenTorn' operation..
Fastest> showtt

UserTokenTorn_VIS
  !______UserTokenTorn_DNF_1
  !______UserTokenTorn_DNF_2
  !______UserTokenTorn_DNF_3
  !______UserTokenTorn_DNF_4
  !______UserTokenTorn_DNF_5
  !______UserTokenTorn_DNF_6
  !______UserTokenTorn_DNF_7
  !______UserTokenTorn_DNF_8
  !______UserTokenTorn_DNF_9
  !______UserTokenTorn_DNF_10

Fastest> setfinitemodel UserTokenTorn_DNF_1 -btsize 1
Fastest> setfinitemodel UserTokenTorn_DNF_2 -btsize 1
Fastest> setfinitemodel UserTokenTorn_DNF_3 -btsize 1
Fastest> setfinitemodel UserTokenTorn_DNF_4 -btsize 1
Fastest> setfinitemodel UserTokenTorn_DNF_5 -btsize 1
Fastest> setfinitemodel UserTokenTorn_DNF_6 -btsize 1
```

```
Fastest> setfinitemodel UserTokenTorn_DNF_7 -btsize 1
Fastest> setfinitemodel UserTokenTorn_DNF_8 -btsize 1
Fastest> setfinitemodel UserTokenTorn_DNF_9 -btsize 1
Fastest> setfinitemodel UserTokenTorn_DNF_10 -btsize 1
Fastest> genalltca
UserTokenTorn_DNF_1 evaluations to perform: at most 64
UserTokenTorn_DNF_1 test case generation -> SUCCESS.
UserTokenTorn_DNF_10 evaluations to perform: at most 64
UserTokenTorn_DNF_10 test case generation -> SUCCESS.
UserTokenTorn_DNF_8 evaluations to perform: at most 64
UserTokenTorn_DNF_8 test case generation -> SUCCESS.
UserTokenTorn_DNF_9 evaluations to perform: at most 64
UserTokenTorn_DNF_9 test case generation -> SUCCESS.
UserTokenTorn_DNF_4 evaluations to perform: at most 64
UserTokenTorn_DNF_4 test case generation -> SUCCESS.
UserTokenTorn_DNF_7 evaluations to perform: at most 64
UserTokenTorn_DNF_7 test case generation -> SUCCESS.
UserTokenTorn_DNF_6 evaluations to perform: at most 64
UserTokenTorn_DNF_6 test case generation -> SUCCESS.
UserTokenTorn_DNF_5 evaluations to perform: at most 64
UserTokenTorn_DNF_5 test case generation -> SUCCESS.
UserTokenTorn_DNF_3 evaluations to perform: at most 64
UserTokenTorn_DNF_3 test case generation -> SUCCESS.
UserTokenTorn_DNF_2 evaluations to perform: at most 64
UserTokenTorn_DNF_2 test case generation -> SUCCESS.
Elapsed time: 240313 miliseconds.

Fastest> selop ReadUserToken
Fastest> genalltt
Generating test tree for 'ReadUserToken' operation..
Fastest> showtt

ReadUserToken_VIS
  !______ReadUserToken_DNF_1
  !______ReadUserToken_DNF_2
  !______ReadUserToken_DNF_3
  !______ReadUserToken_DNF_4
  !______ReadUserToken_DNF_5
  !______ReadUserToken_DNF_6
  !______ReadUserToken_DNF_7
  !______ReadUserToken_DNF_8
  !______ReadUserToken_DNF_9
  !______ReadUserToken_DNF_10

Fastest> setfinitemodel ReadUserToken_DNF_1 -btsize 1
Fastest> setfinitemodel ReadUserToken_DNF_2 -btsize 1
Fastest> setfinitemodel ReadUserToken_DNF_3 -btsize 1
Fastest> setfinitemodel ReadUserToken_DNF_4 -btsize 1
Fastest> setfinitemodel ReadUserToken_DNF_5 -btsize 1
Fastest> setfinitemodel ReadUserToken_DNF_6 -btsize 1
```

```
Fastest> setfinitemodel ReadUserToken_DNF_7 -btsize 1
Fastest> setfinitemodel ReadUserToken_DNF_8 -btsize 1
Fastest> setfinitemodel ReadUserToken_DNF_9 -btsize 1
Fastest> setfinitemodel ReadUserToken_DNF_10 -btsize 1
Fastest> genalltca
ReadUserToken_DNF_1 evaluations to perform: at most 72
ReadUserToken_DNF_1 test case generation -> FAILED.
ReadUserToken_DNF_10 evaluations to perform: at most 72
ReadUserToken_DNF_10 test case generation -> SUCCESS.
ReadUserToken_DNF_7 evaluations to perform: at most 72
ReadUserToken_DNF_7 test case generation -> FAILED.
ReadUserToken_DNF_8 evaluations to perform: at most 72
ReadUserToken_DNF_8 test case generation -> SUCCESS.
ReadUserToken_DNF_9 evaluations to perform: at most 72
ReadUserToken_DNF_9 test case generation -> FAILED.
ReadUserToken_DNF_6 evaluations to perform: at most 72
ReadUserToken_DNF_6 test case generation -> SUCCESS.
ReadUserToken_DNF_5 evaluations to perform: at most 72
ReadUserToken_DNF_5 test case generation -> FAILED.
ReadUserToken_DNF_4 evaluations to perform: at most 72
ReadUserToken_DNF_4 test case generation -> SUCCESS.
ReadUserToken_DNF_2 evaluations to perform: at most 72
ReadUserToken_DNF_2 test case generation -> SUCCESS.
ReadUserToken_DNF_3 evaluations to perform: at most 72
ReadUserToken_DNF_3 test case generation -> FAILED.
Elapsed time: 417953 miliseconds.

Fastest> selop TISReadUserToken
Fastest> genalltt
Generating test tree for 'TISReadUserToken' operation..
Fastest> showtt

TISReadUserToken_VIS
  !______TISReadUserToken_DNF_1
  !______TISReadUserToken_DNF_2
  !______TISReadUserToken_DNF_3
  !______TISReadUserToken_DNF_4
  !______TISReadUserToken_DNF_5
  !______TISReadUserToken_DNF_6
  !______TISReadUserToken_DNF_7
  !______TISReadUserToken_DNF_8
  !______TISReadUserToken_DNF_9
  !______TISReadUserToken_DNF_10

Fastest> setfinitemodel TISReadUserToken_DNF_1 -btsize 1
Fastest> setfinitemodel TISReadUserToken_DNF_2 -btsize 1
Fastest> setfinitemodel TISReadUserToken_DNF_3 -btsize 1
Fastest> setfinitemodel TISReadUserToken_DNF_4 -btsize 1
Fastest> setfinitemodel TISReadUserToken_DNF_5 -btsize 1
Fastest> setfinitemodel TISReadUserToken_DNF_6 -btsize 1
```

```
Fastest> setfinitemodel TISReadUserToken_DNF_7 -btsize 1
Fastest> setfinitemodel TISReadUserToken_DNF_8 -btsize 1
Fastest> setfinitemodel TISReadUserToken_DNF_9 -btsize 1
Fastest> setfinitemodel TISReadUserToken_DNF_10 -btsize 1
Fastest> genalltca
TISReadUserToken_DNF_1 evaluations to perform: at most 72
TISReadUserToken_DNF_1 test case generation -> FAILED.
TISReadUserToken_DNF_10 evaluations to perform: at most 72
TISReadUserToken_DNF_10 test case generation -> SUCCESS.
TISReadUserToken_DNF_8 evaluations to perform: at most 72
TISReadUserToken_DNF_8 test case generation -> SUCCESS.
TISReadUserToken_DNF_9 evaluations to perform: at most 72
TISReadUserToken_DNF_9 test case generation -> FAILED.
TISReadUserToken_DNF_3 evaluations to perform: at most 72
TISReadUserToken_DNF_3 test case generation -> FAILED.
TISReadUserToken_DNF_6 evaluations to perform: at most 72
TISReadUserToken_DNF_6 test case generation -> SUCCESS.
TISReadUserToken_DNF_5 evaluations to perform: at most 72
TISReadUserToken_DNF_5 test case generation -> FAILED.
TISReadUserToken_DNF_7 evaluations to perform: at most 72
TISReadUserToken_DNF_7 test case generation -> FAILED.
TISReadUserToken_DNF_4 evaluations to perform: at most 72
TISReadUserToken_DNF_4 test case generation -> SUCCESS.
TISReadUserToken_DNF_2 evaluations to perform: at most 72
TISReadUserToken_DNF_2 test case generation -> SUCCESS.
Elapsed time: 433844 miliseconds.

Fastest> selop ReadFingerOK
Fastest> genalltt
Generating test tree for 'ReadFingerOK' operation..
Fastest> showtt

ReadFingerOK_VIS
  !______ReadFingerOK_DNF_1
  !______ReadFingerOK_DNF_2
  !______ReadFingerOK_DNF_3
  !______ReadFingerOK_DNF_4
  !______ReadFingerOK_DNF_5
  !______ReadFingerOK_DNF_6
  !______ReadFingerOK_DNF_7
  !______ReadFingerOK_DNF_8
  !______ReadFingerOK_DNF_9
  !______ReadFingerOK_DNF_10

Fastest> setfinitemodel ReadFingerOK_DNF_1 -btsize 1
Fastest> setfinitemodel ReadFingerOK_DNF_2 -btsize 1
Fastest> setfinitemodel ReadFingerOK_DNF_3 -btsize 1
Fastest> setfinitemodel ReadFingerOK_DNF_4 -btsize 1
Fastest> setfinitemodel ReadFingerOK_DNF_5 -btsize 1
Fastest> setfinitemodel ReadFingerOK_DNF_6 -btsize 1
```

```
Fastest> setfinitemodel ReadFingerOK_DNF_7 -btsize 1
Fastest> setfinitemodel ReadFingerOK_DNF_8 -btsize 1
Fastest> setfinitemodel ReadFingerOK_DNF_9 -btsize 1
Fastest> setfinitemodel ReadFingerOK_DNF_10 -btsize 1
Fastest> genalltca
ReadFingerOK_DNF_1 evaluations to perform: at most 8
ReadFingerOK_DNF_1 test case generation -> FAILED.
ReadFingerOK_DNF_4 evaluations to perform: at most 8
ReadFingerOK_DNF_4 test case generation -> SUCCESS.
ReadFingerOK_DNF_7 evaluations to perform: at most 8
ReadFingerOK_DNF_7 test case generation -> FAILED.
ReadFingerOK_DNF_9 evaluations to perform: at most 8
ReadFingerOK_DNF_9 test case generation -> FAILED.
ReadFingerOK_DNF_6 evaluations to perform: at most 8
ReadFingerOK_DNF_6 test case generation -> SUCCESS.
ReadFingerOK_DNF_8 evaluations to perform: at most 8
ReadFingerOK_DNF_8 test case generation -> SUCCESS.
ReadFingerOK_DNF_5 evaluations to perform: at most 8
ReadFingerOK_DNF_5 test case generation -> FAILED.
ReadFingerOK_DNF_10 evaluations to perform: at most 8
ReadFingerOK_DNF_10 test case generation -> SUCCESS.
ReadFingerOK_DNF_3 evaluations to perform: at most 8
ReadFingerOK_DNF_3 test case generation -> FAILED.
ReadFingerOK_DNF_2 evaluations to perform: at most 8
ReadFingerOK_DNF_2 test case generation -> SUCCESS.
Elapsed time: 48328 miliseconds.

Fastest> selop FingerTimeout
Fastest> genalltt
Generating test tree for 'FingerTimeout' operation..
Fastest> showtt

FingerTimeout_VIS
  !______FingerTimeout_DNF_1
  !______FingerTimeout_DNF_2
  !______FingerTimeout_DNF_3
  !______FingerTimeout_DNF_4
  !______FingerTimeout_DNF_5
  !______FingerTimeout_DNF_6
  !______FingerTimeout_DNF_7
  !______FingerTimeout_DNF_8
  !______FingerTimeout_DNF_9
  !______FingerTimeout_DNF_10

Fastest> setfinitemodel FingerTimeout_DNF_1 -btsize 1
Fastest> setfinitemodel FingerTimeout_DNF_2 -btsize 1
Fastest> setfinitemodel FingerTimeout_DNF_3 -btsize 1
Fastest> setfinitemodel FingerTimeout_DNF_4 -btsize 1
Fastest> setfinitemodel FingerTimeout_DNF_5 -btsize 1
Fastest> setfinitemodel FingerTimeout_DNF_6 -btsize 1
```

```
Fastest> setfinitemodel FingerTimeout_DNF_7 -btsize 1
Fastest> setfinitemodel FingerTimeout_DNF_8 -btsize 1
Fastest> setfinitemodel FingerTimeout_DNF_9 -btsize 1
Fastest> setfinitemodel FingerTimeout_DNF_10 -btsize 1
Fastest> genalltca
FingerTimeout_DNF_1 evaluations to perform: at most 8
FingerTimeout_DNF_1 test case generation -> FAILED.
FingerTimeout_DNF_5 evaluations to perform: at most 8
FingerTimeout_DNF_5 test case generation -> FAILED.
FingerTimeout_DNF_9 evaluations to perform: at most 8
FingerTimeout_DNF_9 test case generation -> FAILED.
FingerTimeout_DNF_8 evaluations to perform: at most 8
FingerTimeout_DNF_8 test case generation -> SUCCESS.
FingerTimeout_DNF_10 evaluations to perform: at most 8
FingerTimeout_DNF_10 test case generation -> SUCCESS.
FingerTimeout_DNF_3 evaluations to perform: at most 8
FingerTimeout_DNF_3 test case generation -> FAILED.
FingerTimeout_DNF_4 evaluations to perform: at most 8
FingerTimeout_DNF_4 test case generation -> SUCCESS.
FingerTimeout_DNF_2 evaluations to perform: at most 8
FingerTimeout_DNF_2 test case generation -> SUCCESS.
FingerTimeout_DNF_7 evaluations to perform: at most 8
FingerTimeout_DNF_7 test case generation -> FAILED.
FingerTimeout_DNF_6 evaluations to perform: at most 8
FingerTimeout_DNF_6 test case generation -> SUCCESS.
Elapsed time: 45906 miliseconds.

Fastest> selop ValidateFingerFail
Fastest> genalltt
Generating test tree for 'ValidateFingerFail' operation..
Fastest> showtt

ValidateFingerFail_VIS
  !______ValidateFingerFail_DNF_1
  !______ValidateFingerFail_DNF_2
  !______ValidateFingerFail_DNF_3
  !______ValidateFingerFail_DNF_4
  !______ValidateFingerFail_DNF_5
  !______ValidateFingerFail_DNF_6
  !______ValidateFingerFail_DNF_7
  !______ValidateFingerFail_DNF_8
  !______ValidateFingerFail_DNF_9
  !______ValidateFingerFail_DNF_10

Fastest> setfinitemodel ValidateFingerFail_DNF_1 -btsize 1
Fastest> setfinitemodel ValidateFingerFail_DNF_2 -btsize 1
Fastest> setfinitemodel ValidateFingerFail_DNF_3 -btsize 1
Fastest> setfinitemodel ValidateFingerFail_DNF_4 -btsize 1
Fastest> setfinitemodel ValidateFingerFail_DNF_5 -btsize 1
Fastest> setfinitemodel ValidateFingerFail_DNF_6 -btsize 1
```

```
Fastest> setfinitemodel ValidateFingerFail_DNF_7 -btsize 1
Fastest> setfinitemodel ValidateFingerFail_DNF_8 -btsize 1
Fastest> setfinitemodel ValidateFingerFail_DNF_9 -btsize 1
Fastest> setfinitemodel ValidateFingerFail_DNF_10 -btsize 1
Fastest> genalltca
ValidateFingerFail_DNF_1 evaluations to perform: at most 8
ValidateFingerFail_DNF_1 test case generation -> FAILED.
ValidateFingerFail_DNF_2 evaluations to perform: at most 8
ValidateFingerFail_DNF_2 test case generation -> SUCCESS.
ValidateFingerFail_DNF_6 evaluations to perform: at most 8
ValidateFingerFail_DNF_6 test case generation -> SUCCESS.
ValidateFingerFail_DNF_4 evaluations to perform: at most 8
ValidateFingerFail_DNF_4 test case generation -> SUCCESS.
ValidateFingerFail_DNF_8 evaluations to perform: at most 8
ValidateFingerFail_DNF_8 test case generation -> SUCCESS.
ValidateFingerFail_DNF_7 evaluations to perform: at most 8
ValidateFingerFail_DNF_7 test case generation -> FAILED.
ValidateFingerFail_DNF_5 evaluations to perform: at most 8
ValidateFingerFail_DNF_5 test case generation -> FAILED.
ValidateFingerFail_DNF_3 evaluations to perform: at most 8
ValidateFingerFail_DNF_3 test case generation -> FAILED.
ValidateFingerFail_DNF_10 evaluations to perform: at most 8
ValidateFingerFail_DNF_10 test case generation -> SUCCESS.
ValidateFingerFail_DNF_9 evaluations to perform: at most 8
ValidateFingerFail_DNF_9 test case generation -> FAILED.
Elapsed time: 45641 miliseconds.

Fastest> selop UnlockDoorOK
Fastest> genalltt
Generating test tree for 'UnlockDoorOK' operation..
Fastest> showtt

UnlockDoorOK_VIS
  !______UnlockDoorOK_DNF_1
  !______UnlockDoorOK_DNF_2
  !______UnlockDoorOK_DNF_3
  !______UnlockDoorOK_DNF_4
  !______UnlockDoorOK_DNF_5
  !______UnlockDoorOK_DNF_6
  !______UnlockDoorOK_DNF_7
  !______UnlockDoorOK_DNF_8
  !______UnlockDoorOK_DNF_9
  !______UnlockDoorOK_DNF_10

Fastest> setfinitemodel UnlockDoorOK_DNF_1 -btsize 1
Fastest> setfinitemodel UnlockDoorOK_DNF_2 -btsize 1
Fastest> setfinitemodel UnlockDoorOK_DNF_3 -btsize 1
Fastest> setfinitemodel UnlockDoorOK_DNF_4 -btsize 1
Fastest> setfinitemodel UnlockDoorOK_DNF_5 -btsize 1
Fastest> setfinitemodel UnlockDoorOK_DNF_6 -btsize 1
```

```
Fastest> setfinitemodel UnlockDoorOK_DNF_7 -btsize 1
Fastest> setfinitemodel UnlockDoorOK_DNF_8 -btsize 1
Fastest> setfinitemodel UnlockDoorOK_DNF_9 -btsize 1
Fastest> setfinitemodel UnlockDoorOK_DNF_10 -btsize 1
Fastest> genalltca
UnlockDoorOK_DNF_1 evaluations to perform: at most 8
UnlockDoorOK_DNF_1 test case generation -> FAILED.
UnlockDoorOK_DNF_6 evaluations to perform: at most 8
UnlockDoorOK_DNF_6 test case generation -> SUCCESS.
UnlockDoorOK_DNF_9 evaluations to perform: at most 8
UnlockDoorOK_DNF_9 test case generation -> FAILED.
UnlockDoorOK_DNF_10 evaluations to perform: at most 8
UnlockDoorOK_DNF_10 test case generation -> SUCCESS.
UnlockDoorOK_DNF_7 evaluations to perform: at most 8
UnlockDoorOK_DNF_7 test case generation -> FAILED.
UnlockDoorOK_DNF_5 evaluations to perform: at most 8
UnlockDoorOK_DNF_5 test case generation -> FAILED.
UnlockDoorOK_DNF_8 evaluations to perform: at most 8
UnlockDoorOK_DNF_8 test case generation -> SUCCESS.
UnlockDoorOK_DNF_4 evaluations to perform: at most 8
UnlockDoorOK_DNF_4 test case generation -> SUCCESS.
UnlockDoorOK_DNF_2 evaluations to perform: at most 8
UnlockDoorOK_DNF_2 test case generation -> SUCCESS.
UnlockDoorOK_DNF_3 evaluations to perform: at most 8
UnlockDoorOK_DNF_3 test case generation -> FAILED.
Elapsed time: 45782 miliseconds.

Fastest> selop WaitingTokenRemoval
Fastest> genalltt
Generating test tree for 'WaitingTokenRemoval' operation..
Fastest> showtt

WaitingTokenRemoval_VIS
  !______WaitingTokenRemoval_DNF_1
  !______WaitingTokenRemoval_DNF_2

Fastest> setfinitemodel WaitingTokenRemoval_DNF_2 -btsize 1
Fastest> setfinitemodel WaitingTokenRemoval_DNF_1 -btsize 1
Fastest> genalltca
WaitingTokenRemoval_DNF_1 evaluations to perform: at most 64
WaitingTokenRemoval_DNF_1 test case generation -> SUCCESS.
WaitingTokenRemoval_DNF_2 evaluations to perform: at most 576
WaitingTokenRemoval_DNF_2 test case generation -> SUCCESS.
Elapsed time: 146859 miliseconds.

Fastest> selop TokenRemovalTimeout
Fastest> genalltt
Generating test tree for 'TokenRemovalTimeout' operation..
Fastest> showtt
```

```
TokenRemovalTimeout_VIS
  !______TokenRemovalTimeout_DNF_1
  !______TokenRemovalTimeout_DNF_2
  !______TokenRemovalTimeout_DNF_3
  !______TokenRemovalTimeout_DNF_4
  !______TokenRemovalTimeout_DNF_5
  !______TokenRemovalTimeout_DNF_6
  !______TokenRemovalTimeout_DNF_7
  !______TokenRemovalTimeout_DNF_8
  !______TokenRemovalTimeout_DNF_9
  !______TokenRemovalTimeout_DNF_10

Fastest> setfinitemodel TokenRemovalTimeout_DNF_1
Invalid parameters.  Try 'help'.
Fastest> setfinitemodel TokenRemovalTimeout_DNF_1 -btsize 1
Fastest> setfinitemodel TokenRemovalTimeout_DNF_2 -btsize 1
Fastest> setfinitemodel TokenRemovalTimeout_DNF_3 -btsize 1
Fastest> setfinitemodel TokenRemovalTimeout_DNF_4 -btsize 1
Fastest> setfinitemodel TokenRemovalTimeout_DNF_5 -btsize 1
Fastest> setfinitemodel TokenRemovalTimeout_DNF_6 -btsize 1
Fastest> setfinitemodel TokenRemovalTimeout_DNF_7 -btsize 1
Fastest> setfinitemodel TokenRemovalTimeout_DNF_8 -btsize 1
Fastest> setfinitemodel TokenRemovalTimeout_DNF_9 -btsize 1
Fastest> setfinitemodel TokenRemovalTimeout_DNF_10 -btsize 1
Fastest> genalttca
Invalid command.  Try 'help'.
Fastest> genalltca
TokenRemovalTimeout_DNF_4 evaluations to perform: at most 72
TokenRemovalTimeout_DNF_4 test case generation -> SUCCESS.
TokenRemovalTimeout_DNF_7 evaluations to perform: at most 72
TokenRemovalTimeout_DNF_7 test case generation -> FAILED.
TokenRemovalTimeout_DNF_10 evaluations to perform: at most 72
TokenRemovalTimeout_DNF_10 test case generation -> SUCCESS.
TokenRemovalTimeout_DNF_8 evaluations to perform: at most 72
TokenRemovalTimeout_DNF_8 test case generation -> SUCCESS.
TokenRemovalTimeout_DNF_5 evaluations to perform: at most 72
TokenRemovalTimeout_DNF_5 test case generation -> FAILED.
TokenRemovalTimeout_DNF_9 evaluations to perform: at most 72
TokenRemovalTimeout_DNF_9 test case generation -> FAILED.
TokenRemovalTimeout_DNF_6 evaluations to perform: at most 72
TokenRemovalTimeout_DNF_6 test case generation -> SUCCESS.
TokenRemovalTimeout_DNF_1 evaluations to perform: at most 72
TokenRemovalTimeout_DNF_1 test case generation -> FAILED.
TokenRemovalTimeout_DNF_2 evaluations to perform: at most 72
TokenRemovalTimeout_DNF_2 test case generation -> SUCCESS.
TokenRemovalTimeout_DNF_3 evaluations to perform: at most 72
TokenRemovalTimeout_DNF_3 test case generation -> FAILED.
Elapsed time: 342625 miliseconds.

Fastest> selop TISUnlockDoor
```

```
Fastest> genalltt
Generating test tree for 'TISUnlockDoor' operation..
Fastest> showtt

TISUnlockDoor_VIS
  !______TISUnlockDoor_DNF_1
  !______TISUnlockDoor_DNF_2
  !______TISUnlockDoor_DNF_3
  !______TISUnlockDoor_DNF_4
  !______TISUnlockDoor_DNF_5
  !______TISUnlockDoor_DNF_6
  !______TISUnlockDoor_DNF_7
  !______TISUnlockDoor_DNF_8
  !______TISUnlockDoor_DNF_9
  !______TISUnlockDoor_DNF_10

Fastest> setfinitemodel TISUnlockDoor_DNF_1 -btsize 1
Fastest> setfinitemodel TISUnlockDoor_DNF_2 -btsize 1
Fastest> setfinitemodel TISUnlockDoor_DNF_3 -btsize 1
Fastest> setfinitemodel TISUnlockDoor_DNF_4 -btsize 1
Fastest> setfinitemodel TISUnlockDoor_DNF_5 -btsize 1
Fastest> setfinitemodel TISUnlockDoor_DNF_6 -btsize 1
Fastest> setfinitemodel TISUnlockDoor_DNF_7 -btsize 1
Fastest> setfinitemodel TISUnlockDoor_DNF_8 -btsize 1
Fastest> setfinitemodel TISUnlockDoor_DNF_9 -btsize 1
Fastest> setfinitemodel TISUnlockDoor_DNF_10 -btsize 1
Fastest> genalltca
TISUnlockDoor_DNF_3 evaluations to perform: at most 72
TISUnlockDoor_DNF_3 test case generation -> FAILED.
TISUnlockDoor_DNF_7 evaluations to perform: at most 72
TISUnlockDoor_DNF_7 test case generation -> FAILED.
TISUnlockDoor_DNF_9 evaluations to perform: at most 72
TISUnlockDoor_DNF_9 test case generation -> FAILED.
TISUnlockDoor_DNF_8 evaluations to perform: at most 72
TISUnlockDoor_DNF_8 test case generation -> SUCCESS.
TISUnlockDoor_DNF_6 evaluations to perform: at most 72
TISUnlockDoor_DNF_6 test case generation -> SUCCESS.
TISUnlockDoor_DNF_2 evaluations to perform: at most 72
TISUnlockDoor_DNF_2 test case generation -> SUCCESS.
TISUnlockDoor_DNF_5 evaluations to perform: at most 72
TISUnlockDoor_DNF_5 test case generation -> FAILED.
TISUnlockDoor_DNF_10 evaluations to perform: at most 72
TISUnlockDoor_DNF_10 test case generation -> SUCCESS.
TISUnlockDoor_DNF_1 evaluations to perform: at most 72
TISUnlockDoor_DNF_1 test case generation -> FAILED.
TISUnlockDoor_DNF_4 evaluations to perform: at most 72
TISUnlockDoor_DNF_4 test case generation -> SUCCESS.
Elapsed time: 341359 miliseconds.

Fastest> selop FailedAccessTokenRemoved
```

```
Fastest> genalltt
Generating test tree for 'FailedAccessTokenRemoved' operation..
Fastest> showtt

FailedAccessTokenRemoved_VIS
  !______FailedAccessTokenRemoved_DNF_1
  !______FailedAccessTokenRemoved_DNF_2
  !______FailedAccessTokenRemoved_DNF_3
  !______FailedAccessTokenRemoved_DNF_4
  !______FailedAccessTokenRemoved_DNF_5
  !______FailedAccessTokenRemoved_DNF_6
  !______FailedAccessTokenRemoved_DNF_7
  !______FailedAccessTokenRemoved_DNF_8
  !______FailedAccessTokenRemoved_DNF_9
  !______FailedAccessTokenRemoved_DNF_10

Fastest> setfinitemodel FailedAccessTokenRemoved_DNF_1
Invalid parameters.  Try 'help'.
Fastest> setfinitemodel FailedAccessTokenRemoved_DNF_1 -btsize 1
Fastest> setfinitemodel FailedAccessTokenRemoved_DNF_2 -btsize 1
Fastest> setfinitemodel FailedAccessTokenRemoved_DNF_3 -btsize 1
Fastest> setfinitemodel FailedAccessTokenRemoved_DNF_4 -btsize 1
Fastest> setfinitemodel FailedAccessTokenRemoved_DNF_5 -btsize 1
Fastest> setfinitemodel FailedAccessTokenRemoved_DNF_6 -btsize 1
Fastest> setfinitemodel FailedAccessTokenRemoved_DNF_7 -btsize 1
Fastest> setfinitemodel FailedAccessTokenRemoved_DNF_8 -btsize 1
Fastest> setfinitemodel FailedAccessTokenRemoved_DNF_9 -btsize 1
Fastest> setfinitemodel FailedAccessTokenRemoved_DNF_10 -btsize 1
Fastest> genalltca
FailedAccessTokenRemoved_DNF_2 evaluations to perform: at most 8
FailedAccessTokenRemoved_DNF_2 test case generation -> SUCCESS.
FailedAccessTokenRemoved_DNF_9 evaluations to perform: at most 8
FailedAccessTokenRemoved_DNF_9 test case generation -> FAILED.
FailedAccessTokenRemoved_DNF_8 evaluations to perform: at most 8
FailedAccessTokenRemoved_DNF_8 test case generation -> SUCCESS.
FailedAccessTokenRemoved_DNF_4 evaluations to perform: at most 8
FailedAccessTokenRemoved_DNF_4 test case generation -> SUCCESS.
FailedAccessTokenRemoved_DNF_6 evaluations to perform: at most 8
FailedAccessTokenRemoved_DNF_6 test case generation -> SUCCESS.
FailedAccessTokenRemoved_DNF_5 evaluations to perform: at most 8
FailedAccessTokenRemoved_DNF_5 test case generation -> FAILED.
FailedAccessTokenRemoved_DNF_7 evaluations to perform: at most 8
FailedAccessTokenRemoved_DNF_7 test case generation -> FAILED.
FailedAccessTokenRemoved_DNF_3 evaluations to perform: at most 8
FailedAccessTokenRemoved_DNF_3 test case generation -> FAILED.
FailedAccessTokenRemoved_DNF_10 evaluations to perform: at most 8
FailedAccessTokenRemoved_DNF_10 test case generation -> SUCCESS.
FailedAccessTokenRemoved_DNF_1 evaluations to perform: at most 8
FailedAccessTokenRemoved_DNF_1 test case generation -> FAILED.
Elapsed time: 45469 miliseconds.
```

```
Fastest> selop ReadEnrolmentData
Fastest> genalltt
Generating test tree for 'ReadEnrolmentData' operation..
Fastest> showtt

ReadEnrolmentData_VIS
  !______ReadEnrolmentData_DNF_1
  !______ReadEnrolmentData_DNF_2

Fastest> setfinitemodel ReadEnrolmentData_DNF_1 -btsize 2
Fastest> setfinitemodel ReadEnrolmentData_DNF_2 -btsize 2
Fastest> genalltca
ReadEnrolmentData_DNF_2 evaluations to perform: at most 2304
ReadEnrolmentData_DNF_2 test case generation -> SUCCESS.
ReadEnrolmentData_DNF_1 evaluations to perform: at most 2304
ReadEnrolmentData_DNF_1 test case generation -> SUCCESS.
Elapsed time: 249656 miliseconds.

Fastest> selop CompleteFailedEnrolment
Fastest> genalltt
Generating test tree for 'CompleteFailedEnrolment' operation..
Fastest> showtt

CompleteFailedEnrolment_VIS
  !______CompleteFailedEnrolment_DNF_1
  !______CompleteFailedEnrolment_DNF_2

Fastest> setfinitemodel CompleteFailedEnrolment_DNF_2 -btsize 2
Fastest> setfinitemodel CompleteFailedEnrolment_DNF_1 -btsize 2
Fastest> genalltca
CompleteFailedEnrolment_DNF_1 evaluations to perform: at most 2304
CompleteFailedEnrolment_DNF_1 test case generation -> SUCCESS.
CompleteFailedEnrolment_DNF_2 evaluations to perform: at most 2304
CompleteFailedEnrolment_DNF_2 test case generation -> SUCCESS.
Elapsed time: 246312 miliseconds.

Fastest> selop AdminTokenTear
Fastest> genalltt
Generating test tree for 'AdminTokenTear' operation..
Fastest> showtt

AdminTokenTear_VIS
  !______AdminTokenTear_DNF_1
  !______AdminTokenTear_DNF_2
  !______AdminTokenTear_DNF_3
  !______AdminTokenTear_DNF_4
  !______AdminTokenTear_DNF_5

Fastest> setfinitemodel AdminTokenTear_DNF_1 -btsize 1
```

```
Fastest> setfinitemodel AdminTokenTear_DNF_2 -btsize 1
Fastest> setfinitemodel AdminTokenTear_DNF_3 -btsize 1
Fastest> setfinitemodel AdminTokenTear_DNF_4 -btsize 1
Fastest> setfinitemodel AdminTokenTear_DNF_5 -btsize 1
Fastest> genalltca
AdminTokenTear_DNF_3 evaluations to perform: at most 8
AdminTokenTear_DNF_3 test case generation -> SUCCESS.
AdminTokenTear_DNF_1 evaluations to perform: at most 8
AdminTokenTear_DNF_1 test case generation -> SUCCESS.
AdminTokenTear_DNF_4 evaluations to perform: at most 8
AdminTokenTear_DNF_4 test case generation -> SUCCESS.
AdminTokenTear_DNF_2 evaluations to perform: at most 8
AdminTokenTear_DNF_2 test case generation -> SUCCESS.
AdminTokenTear_DNF_5 evaluations to perform: at most 8
AdminTokenTear_DNF_5 test case generation -> SUCCESS.
Elapsed time: 16109 miliseconds.

Fastest> selop BadAdminTokenTear
Fastest> genalltt
Generating test tree for 'BadAdminTokenTear' operation..
Fastest> showtt

BadAdminTokenTear_VIS
  !______BadAdminTokenTear_DNF_1
  !______BadAdminTokenTear_DNF_2
  !______BadAdminTokenTear_DNF_3
  !______BadAdminTokenTear_DNF_4
  !______BadAdminTokenTear_DNF_5

Fastest> setfinitemodel BadAdminTokenTear_DNF_1 -btsize 1
Fastest> setfinitemodel BadAdminTokenTear_DNF_2 -btsize 1
Fastest> setfinitemodel BadAdminTokenTear_DNF_3 -btsize 1
Fastest> setfinitemodel BadAdminTokenTear_DNF_4 -btsize 1
Fastest> setfinitemodel BadAdminTokenTear_DNF_5 -btsize 1
Fastest> genalltca
BadAdminTokenTear_DNF_1 evaluations to perform: at most 72
BadAdminTokenTear_DNF_1 test case generation -> SUCCESS.
BadAdminTokenTear_DNF_4 evaluations to perform: at most 72
BadAdminTokenTear_DNF_4 test case generation -> SUCCESS.
BadAdminTokenTear_DNF_3 evaluations to perform: at most 72
BadAdminTokenTear_DNF_3 test case generation -> SUCCESS.
BadAdminTokenTear_DNF_2 evaluations to perform: at most 72
BadAdminTokenTear_DNF_2 test case generation -> SUCCESS.
BadAdminTokenTear_DNF_5 evaluations to perform: at most 72
BadAdminTokenTear_DNF_5 test case generation -> SUCCESS.
Elapsed time: 128563 miliseconds.

Fastest> selop LoginAborted
Fastest> genalltt
Generating test tree for 'LoginAborted' operation..
```

```
Fastest> showtt

LoginAborted_VIS
  !______LoginAborted_DNF_1
  !______LoginAborted_DNF_2
  !______LoginAborted_DNF_3
  !______LoginAborted_DNF_4
  !______LoginAborted_DNF_5

Fastest> setfinitemodel LoginAborted_DNF_1
Invalid parameters.  Try 'help'.
Fastest> setfinitemodel LoginAborted_DNF_1 -btsize 1
Fastest> setfinitemodel LoginAborted_DNF_2 -btsize 1
Fastest> setfinitemodel LoginAborted_DNF_3 -btsize 1
Fastest> setfinitemodel LoginAborted_DNF_4 -btsize 1
Fastest> setfinitemodel LoginAborted_DNF_5 -btsize 1
Fastest> genalltca
LoginAborted_DNF_1 evaluations to perform: at most 8
LoginAborted_DNF_1 test case generation -> SUCCESS.
LoginAborted_DNF_5 evaluations to perform: at most 8
LoginAborted_DNF_5 test case generation -> SUCCESS.
LoginAborted_DNF_2 evaluations to perform: at most 8
LoginAborted_DNF_2 test case generation -> SUCCESS.
LoginAborted_DNF_4 evaluations to perform: at most 8
LoginAborted_DNF_3 evaluations to perform: at most 8
LoginAborted_DNF_4 test case generation -> SUCCESS.
LoginAborted_DNF_3 test case generation -> SUCCESS.
Elapsed time: 20281 miliseconds.

Fastest> selop TISValidateAdminToken
Fastest> genalltt
Generating test tree for 'TISValidateAdminToken' operation..
Fastest> showtt

TISValidateAdminToken_VIS
  !______TISValidateAdminToken_DNF_1
  !______TISValidateAdminToken_DNF_2
  !______TISValidateAdminToken_DNF_3
  !______TISValidateAdminToken_DNF_4
  !______TISValidateAdminToken_DNF_5
  !______TISValidateAdminToken_DNF_6
  !______TISValidateAdminToken_DNF_7

Fastest> setfinitemodel TISValidateAdminToken_DNF_1
Invalid parameters.  Try 'help'.
Fastest> setfinitemodel TISValidateAdminToken_DNF_1 -btsize 1
Fastest> setfinitemodel TISValidateAdminToken_DNF_2 -btsize 1
Fastest> setfinitemodel TISValidateAdminToken_DNF_3 -btsize 1
Fastest> setfinitemodel TISValidateAdminToken_DNF_4 -btsize 1
Fastest> setfinitemodel TISValidateAdminToken_DNF_5 -btsize 1
```

```
Fastest> setfinitemodel TISValidateAdminToken_DNF_6 -btsize 1
Fastest> setfinitemodel TISValidateAdminToken_DNF_7 -btsize 1
Fastest> genalltca
TISValidateAdminToken_DNF_1 evaluations to perform: at most 8
TISValidateAdminToken_DNF_1 test case generation -> FAILED.
TISValidateAdminToken_DNF_7 evaluations to perform: at most 8
TISValidateAdminToken_DNF_7 test case generation -> SUCCESS.
TISValidateAdminToken_DNF_4 evaluations to perform: at most 8
TISValidateAdminToken_DNF_4 test case generation -> SUCCESS.
TISValidateAdminToken_DNF_5 evaluations to perform: at most 8
TISValidateAdminToken_DNF_5 test case generation -> SUCCESS.
TISValidateAdminToken_DNF_6 evaluations to perform: at most 8
TISValidateAdminToken_DNF_6 test case generation -> SUCCESS.
TISValidateAdminToken_DNF_2 evaluations to perform: at most 8
TISValidateAdminToken_DNF_2 test case generation -> FAILED.
TISValidateAdminToken_DNF_3 evaluations to perform: at most 8
TISValidateAdminToken_DNF_3 test case generation -> SUCCESS.
Elapsed time: 28062 miliseconds.

Fastest> selop TISCompleteFailedAdminLogon
Fastest> genalltt
Generating test tree for 'TISCompleteFailedAdminLogon' operation..
Fastest> showtt

TISCompleteFailedAdminLogon_VIS
  !______TISCompleteFailedAdminLogon_DNF_1
  !______TISCompleteFailedAdminLogon_DNF_2

Fastest> setfinitemodel TISCompleteFailedAdminLogon_DNF_1 -btsize 1
Fastest> setfinitemodel TISCompleteFailedAdminLogon_DNF_2 -btsize 1
Fastest> genalltca
TISCompleteFailedAdminLogon_DNF_2 evaluations to perform: at most 8
TISCompleteFailedAdminLogon_DNF_2 test case generation -> SUCCESS.
TISCompleteFailedAdminLogon_DNF_1 evaluations to perform: at most 8
TISCompleteFailedAdminLogon_DNF_1 test case generation -> SUCCESS.
Elapsed time: 9063 miliseconds.

Fastest> selop TISAdminLogon
Fastest> genalltt
Generating test tree for 'TISAdminLogon' operation..
Fastest> showtt

TISAdminLogon_VIS
  !______TISAdminLogon_DNF_1
  !______TISAdminLogon_DNF_2
  !______TISAdminLogon_DNF_3
  !______TISAdminLogon_DNF_4
  !______TISAdminLogon_DNF_5
  !______TISAdminLogon_DNF_6
  !______TISAdminLogon_DNF_7
```

```
  !______TISAdminLogon_DNF_8
  !______TISAdminLogon_DNF_9
  !______TISAdminLogon_DNF_10

Fastest> setfinitemodel TISAdminLogon_DNF_1 -btsize 1
Fastest> setfinitemodel TISAdminLogon_DNF_2 -btsize 1
Fastest> setfinitemodel TISAdminLogon_DNF_3 -btsize 1
Fastest> setfinitemodel TISAdminLogon_DNF_4 -btsize 1
Fastest> setfinitemodel TISAdminLogon_DNF_5 -btsize 1
Fastest> setfinitemodel TISAdminLogon_DNF_6 -btsize 1
Fastest> setfinitemodel TISAdminLogon_DNF_7 -btsize 1
Fastest> setfinitemodel TISAdminLogon_DNF_8 -btsize 1
Fastest> setfinitemodel TISAdminLogon_DNF_9 -btsize 1
Fastest> setfinitemodel TISAdminLogon_DNF_10 -btsize 1
Fastest> genalltca
TISAdminLogon_DNF_1 evaluations to perform: at most 64
TISAdminLogon_DNF_1 test case generation -> FAILED.
TISAdminLogon_DNF_5 evaluations to perform: at most 8
TISAdminLogon_DNF_5 test case generation -> SUCCESS.
TISAdminLogon_DNF_4 evaluations to perform: at most 8
TISAdminLogon_DNF_4 test case generation -> SUCCESS.
TISAdminLogon_DNF_9 evaluations to perform: at most 8
TISAdminLogon_DNF_9 test case generation -> SUCCESS.
TISAdminLogon_DNF_6 evaluations to perform: at most 8
TISAdminLogon_DNF_6 test case generation -> SUCCESS.
TISAdminLogon_DNF_10 evaluations to perform: at most 8
TISAdminLogon_DNF_10 test case generation -> SUCCESS.
TISAdminLogon_DNF_8 evaluations to perform: at most 8
TISAdminLogon_DNF_8 test case generation -> SUCCESS.
TISAdminLogon_DNF_7 evaluations to perform: at most 8
TISAdminLogon_DNF_7 test case generation -> SUCCESS.
TISAdminLogon_DNF_3 evaluations to perform: at most 8
TISAdminLogon_DNF_3 test case generation -> FAILED.
TISAdminLogon_DNF_2 evaluations to perform: at most 8
TISAdminLogon_DNF_2 test case generation -> FAILED.
Elapsed time: 63016 miliseconds.

Fastest> selop BadAdminLogout
Fastest> genalltt
Generating test tree for 'BadAdminLogout' operation..
Fastest> showtt

BadAdminLogout_VIS
  !______BadAdminLogout_DNF_1
  !______BadAdminLogout_DNF_2
  !______BadAdminLogout_DNF_3
  !______BadAdminLogout_DNF_4
  !______BadAdminLogout_DNF_5

Fastest> setfinitemodel BadAdminLogout_DNF_1 -btsize 1
```

```
Fastest> setfinitemodel BadAdminLogout_DNF_2 -btsize 1
Fastest> setfinitemodel BadAdminLogout_DNF_3 -btsize 1
Fastest> setfinitemodel BadAdminLogout_DNF_4 -btsize 1
Fastest> setfinitemodel BadAdminLogout_DNF_5 -btsize 1
Fastest> genalltca
BadAdminLogout_DNF_3 evaluations to perform: at most 576
BadAdminLogout_DNF_3 test case generation -> SUCCESS.
BadAdminLogout_DNF_4 evaluations to perform: at most 576
BadAdminLogout_DNF_4 test case generation -> SUCCESS.
BadAdminLogout_DNF_1 evaluations to perform: at most 576
BadAdminLogout_DNF_1 test case generation -> SUCCESS.
BadAdminLogout_DNF_5 evaluations to perform: at most 576
BadAdminLogout_DNF_5 test case generation -> SUCCESS.
BadAdminLogout_DNF_2 evaluations to perform: at most 576
BadAdminLogout_DNF_2 test case generation -> SUCCESS.
Elapsed time: 1531750 miliseconds.

Fastest> selop TokenRemovedAdminLogout
Fastest> genalltt
Generating test tree for 'TokenRemovedAdminLogout' operation..
Fastest> showtt

TokenRemovedAdminLogout_VIS
  !______TokenRemovedAdminLogout_DNF_1
  !______TokenRemovedAdminLogout_DNF_2
  !______TokenRemovedAdminLogout_DNF_3
  !______TokenRemovedAdminLogout_DNF_4
  !______TokenRemovedAdminLogout_DNF_5

Fastest> setfinitemodel TokenRemovedAdminLogout_DNF_1 -btsize 1
Fastest> setfinitemodel TokenRemovedAdminLogout_DNF_2 -btsize 1
Fastest> setfinitemodel TokenRemovedAdminLogout_DNF_3 -btsize 1
Fastest> setfinitemodel TokenRemovedAdminLogout_DNF_4 -btsize 1
Fastest> setfinitemodel TokenRemovedAdminLogout_DNF_5 -btsize 1
Fastest> genalltca
TokenRemovedAdminLogout_DNF_1 evaluations to perform: at most 64
TokenRemovedAdminLogout_DNF_1 test case generation -> SUCCESS.
TokenRemovedAdminLogout_DNF_2 evaluations to perform: at most 64
TokenRemovedAdminLogout_DNF_2 test case generation -> SUCCESS.
TokenRemovedAdminLogout_DNF_5 evaluations to perform: at most 64
TokenRemovedAdminLogout_DNF_5 test case generation -> SUCCESS.
TokenRemovedAdminLogout_DNF_3 evaluations to perform: at most 64
TokenRemovedAdminLogout_DNF_3 test case generation -> SUCCESS.
TokenRemovedAdminLogout_DNF_4 evaluations to perform: at most 64
TokenRemovedAdminLogout_DNF_4 test case generation -> SUCCESS.
Elapsed time: 48860 miliseconds.

Fastest> selop TISCompleteTimeoutAdminLogout
Fastest> genalltt
Generating test tree for 'TISCompleteTimeoutAdminLogout' operation..
```

```
Fastest> showtt

TISCompleteTimeoutAdminLogout_VIS
  !______TISCompleteTimeoutAdminLogout_DNF_1
  !______TISCompleteTimeoutAdminLogout_DNF_2

Fastest> setfinitemodel TISCompleteTimeoutAdminLogout_DNF_1 -btsize 1
Fastest> setfinitemodel TISCompleteTimeoutAdminLogout_DNF_2 -btsize 1
Fastest> genalltca
TISCompleteTimeoutAdminLogout_DNF_2 evaluations to perform: at most 8
TISCompleteTimeoutAdminLogout_DNF_2 test case generation -> SUCCESS.
TISCompleteTimeoutAdminLogout_DNF_1 evaluations to perform: at most 8
TISCompleteTimeoutAdminLogout_DNF_1 test case generation -> SUCCESS.
Elapsed time: 66578 miliseconds.
```

Las siguientes operaciones generan varias clases pero ningún caso de prueba. La razón por la cual no se encuentran casos es que Fastest^{TM} no soporta la referencia a esquemas en el predicado de otro esquema y estas operaciones contienen referencias a esquemas en los predicados. Las clases de prueba se listan en el apéndice B.

```
Fastest> selop BioCheckRequired
Fastest> genalltt
Generating test tree for 'BioCheckRequired' operation..
Fastest> showtt

BioCheckRequired_VIS
  !______BioCheckRequired_DNF_1
  !______BioCheckRequired_DNF_2
  !______BioCheckRequired_DNF_3
  !______BioCheckRequired_DNF_4
  !______BioCheckRequired_DNF_5
  !______BioCheckRequired_DNF_6
  !______BioCheckRequired_DNF_7
  !______BioCheckRequired_DNF_8
  !______BioCheckRequired_DNF_9
  !______BioCheckRequired_DNF_10

Fastest> setfinitemodel BioCheckRequired_DNF_1 -btsize 1
Fastest> setfinitemodel BioCheckRequired_DNF_2 -btsize 1
Fastest> setfinitemodel BioCheckRequired_DNF_3 -btsize 1
Fastest> setfinitemodel BioCheckRequired_DNF_4 -btsize 1
Fastest> setfinitemodel BioCheckRequired_DNF_5 -btsize 1
Fastest> setfinitemodel BioCheckRequired_DNF_6 -btsize 1
Fastest> setfinitemodel BioCheckRequired_DNF_7 -btsize 1
Fastest> setfinitemodel BioCheckRequired_DNF_8 -btsize 1
Fastest> setfinitemodel BioCheckRequired_DNF_9 -btsize 1
Fastest> setfinitemodel BioCheckRequired_DNF_10 -btsize 1
Fastest> genalltca
BioCheckRequired_DNF_1 evaluations to perform: at most 8
```

```
BioCheckRequired_DNF_1 test case generation -> FAILED.
BioCheckRequired_DNF_5 evaluations to perform: at most 8
BioCheckRequired_DNF_5 test case generation -> FAILED.
BioCheckRequired_DNF_3 evaluations to perform: at most 8
BioCheckRequired_DNF_3 test case generation -> FAILED.
BioCheckRequired_DNF_7 evaluations to perform: at most 8
BioCheckRequired_DNF_7 test case generation -> FAILED.
BioCheckRequired_DNF_4 evaluations to perform: at most 8
BioCheckRequired_DNF_4 test case generation -> FAILED.
BioCheckRequired_DNF_2 evaluations to perform: at most 8
BioCheckRequired_DNF_2 test case generation -> FAILED.
BioCheckRequired_DNF_6 evaluations to perform: at most 8
BioCheckRequired_DNF_6 test case generation -> FAILED.
BioCheckRequired_DNF_10 evaluations to perform: at most 8
BioCheckRequired_DNF_10 test case generation -> FAILED.
BioCheckRequired_DNF_9 evaluations to perform: at most 8
BioCheckRequired_DNF_9 test case generation -> FAILED.
BioCheckRequired_DNF_8 evaluations to perform: at most 8
BioCheckRequired_DNF_8 test case generation -> FAILED.
Elapsed time: 31875 miliseconds.

Fastest> selop BioCheckNotRequired
Fastest> genalltt
Generating test tree for 'BioCheckNotRequired' operation..
Fastest> showtt

BioCheckNotRequired_VIS
  !______BioCheckNotRequired_DNF_1
  !______BioCheckNotRequired_DNF_2
  !______BioCheckNotRequired_DNF_3
  !______BioCheckNotRequired_DNF_4
  !______BioCheckNotRequired_DNF_5
  !______BioCheckNotRequired_DNF_6
  !______BioCheckNotRequired_DNF_7
  !______BioCheckNotRequired_DNF_8
  !______BioCheckNotRequired_DNF_9
  !______BioCheckNotRequired_DNF_10

Fastest> setfinitemodel BioCheckNotRequired_DNF_1 -btsize 2
Fastest> setfinitemodel BioCheckNotRequired_DNF_2 -btsize 2
Fastest> setfinitemodel BioCheckNotRequired_DNF_3 -btsize 2
Fastest> setfinitemodel BioCheckNotRequired_DNF_4 -btsize 2
Fastest> setfinitemodel BioCheckNotRequired_DNF_5 -btsize 2
Fastest> setfinitemodel BioCheckNotRequired_DNF_6 -btsize 2
Fastest> setfinitemodel BioCheckNotRequired_DNF_7 -btsize 2
Fastest> setfinitemodel BioCheckNotRequired_DNF_8 -btsize 2
Fastest> setfinitemodel BioCheckNotRequired_DNF_9 -btsize 2
Fastest> setfinitemodel BioCheckNotRequired_DNF_10 -btsize 2
Fastest> genalltca
BioCheckNotRequired_DNF_1 evaluations to perform: at most 2304
```

```
BioCheckNotRequired_DNF_1 test case generation -> FAILED.
BioCheckNotRequired_DNF_8 evaluations to perform: at most 2304
BioCheckNotRequired_DNF_8 test case generation -> FAILED.
BioCheckNotRequired_DNF_3 evaluations to perform: at most 2304
BioCheckNotRequired_DNF_3 test case generation -> FAILED.
BioCheckNotRequired_DNF_9 evaluations to perform: at most 2304
BioCheckNotRequired_DNF_9 test case generation -> FAILED.
BioCheckNotRequired_DNF_7 evaluations to perform: at most 2304
BioCheckNotRequired_DNF_7 test case generation -> FAILED.
BioCheckNotRequired_DNF_5 evaluations to perform: at most 2304
BioCheckNotRequired_DNF_5 test case generation -> FAILED.
BioCheckNotRequired_DNF_10 evaluations to perform: at most 2304
BioCheckNotRequired_DNF_10 test case generation -> FAILED.
BioCheckNotRequired_DNF_2 evaluations to perform: at most 2304
BioCheckNotRequired_DNF_2 test case generation -> FAILED.
BioCheckNotRequired_DNF_4 evaluations to perform: at most 2304
BioCheckNotRequired_DNF_4 test case generation -> FAILED.
BioCheckNotRequired_DNF_6 evaluations to perform: at most 2304
BioCheckNotRequired_DNF_6 test case generation -> FAILED.
Elapsed time: 7854032 miliseconds.

Fastest> selop ValidateUserTokenOK
Fastest> genalltt
Generating test tree for 'ValidateUserTokenOK' operation..
Fastest> showtt

ValidateUserTokenOK_VIS
  !______ValidateUserTokenOK_DNF_1
  !______ValidateUserTokenOK_DNF_2
  !______ValidateUserTokenOK_DNF_3
  !______ValidateUserTokenOK_DNF_4
  !______ValidateUserTokenOK_DNF_5
  !______ValidateUserTokenOK_DNF_6
  !______ValidateUserTokenOK_DNF_7
  !______ValidateUserTokenOK_DNF_8
  !______ValidateUserTokenOK_DNF_9
  !______ValidateUserTokenOK_DNF_10
  !______ValidateUserTokenOK_DNF_11
  !______ValidateUserTokenOK_DNF_12
  !______ValidateUserTokenOK_DNF_13
  !______ValidateUserTokenOK_DNF_14
  !______ValidateUserTokenOK_DNF_15
  !______ValidateUserTokenOK_DNF_16
  !______ValidateUserTokenOK_DNF_17
  !______ValidateUserTokenOK_DNF_18
  !______ValidateUserTokenOK_DNF_19
  !______ValidateUserTokenOK_DNF_20

Fastest> setfinitemodel ValidateUserTokenOK_DNF_1 -btsize 1
Fastest> setfinitemodel ValidateUserTokenOK_DNF_2 -btsize 1
```

```
Fastest> setfinitemodel ValidateUserTokenOK_DNF_3 -btsize 1
Fastest> setfinitemodel ValidateUserTokenOK_DNF_4 -btsize 1
Fastest> setfinitemodel ValidateUserTokenOK_DNF_5 -btsize 1
Fastest> setfinitemodel ValidateUserTokenOK_DNF_6 -btsize 1
Fastest> setfinitemodel ValidateUserTokenOK_DNF_7 -btsize 1
Fastest> setfinitemodel ValidateUserTokenOK_DNF_8 -btsize 1
Fastest> setfinitemodel ValidateUserTokenOK_DNF_9 -btsize 1
Fastest> setfinitemodel ValidateUserTokenOK_DNF_10 -btsize 1
Fastest> setfinitemodel ValidateUserTokenOK_DNF_11 -btsize 1
Fastest> setfinitemodel ValidateUserTokenOK_DNF_12 -btsize 1
Fastest> setfinitemodel ValidateUserTokenOK_DNF_13 -btsize 1
Fastest> setfinitemodel ValidateUserTokenOK_DNF_14 -btsize 1
Fastest> setfinitemodel ValidateUserTokenOK_DNF_15 -btsize 1
Fastest> setfinitemodel ValidateUserTokenOK_DNF_16 -btsize 1
Fastest> setfinitemodel ValidateUserTokenOK_DNF_17 -btsize 1
Fastest> setfinitemodel ValidateUserTokenOK_DNF_18 -btsize 1
Fastest> setfinitemodel ValidateUserTokenOK_DNF_19 -btsize 1
Fastest> setfinitemodel ValidateUserTokenOK_DNF_20 -btsize 1
Fastest> genalltca
ValidateUserTokenOK_DNF_4 evaluations to perform: at most 8
ValidateUserTokenOK_DNF_4 test case generation -> FAILED.
ValidateUserTokenOK_DNF_20 evaluations to perform: at most 8
ValidateUserTokenOK_DNF_20 test case generation -> FAILED.
ValidateUserTokenOK_DNF_19 evaluations to perform: at most 8
ValidateUserTokenOK_DNF_19 test case generation -> FAILED.
ValidateUserTokenOK_DNF_15 evaluations to perform: at most 8
ValidateUserTokenOK_DNF_15 test case generation -> FAILED.
ValidateUserTokenOK_DNF_17 evaluations to perform: at most 8
ValidateUserTokenOK_DNF_17 test case generation -> FAILED.
ValidateUserTokenOK_DNF_12 evaluations to perform: at most 8
ValidateUserTokenOK_DNF_12 test case generation -> FAILED.
ValidateUserTokenOK_DNF_13 evaluations to perform: at most 8
ValidateUserTokenOK_DNF_13 test case generation -> FAILED.
ValidateUserTokenOK_DNF_14 evaluations to perform: at most 8
ValidateUserTokenOK_DNF_14 test case generation -> FAILED.
ValidateUserTokenOK_DNF_18 evaluations to perform: at most 8
ValidateUserTokenOK_DNF_18 test case generation -> FAILED.
ValidateUserTokenOK_DNF_16 evaluations to perform: at most 8
ValidateUserTokenOK_DNF_16 test case generation -> FAILED.
ValidateUserTokenOK_DNF_7 evaluations to perform: at most 8
ValidateUserTokenOK_DNF_7 test case generation -> FAILED.
ValidateUserTokenOK_DNF_11 evaluations to perform: at most 8
ValidateUserTokenOK_DNF_11 test case generation -> FAILED.
ValidateUserTokenOK_DNF_10 evaluations to perform: at most 8
ValidateUserTokenOK_DNF_10 test case generation -> FAILED.
ValidateUserTokenOK_DNF_9 evaluations to perform: at most 8
ValidateUserTokenOK_DNF_9 test case generation -> FAILED.
ValidateUserTokenOK_DNF_6 evaluations to perform: at most 8
ValidateUserTokenOK_DNF_6 test case generation -> FAILED.
ValidateUserTokenOK_DNF_8 evaluations to perform: at most 8
```

```
ValidateUserTokenOK_DNF_5 evaluations to perform: at most 8
ValidateUserTokenOK_DNF_8 test case generation -> FAILED.
ValidateUserTokenOK_DNF_5 test case generation -> FAILED.
ValidateUserTokenOK_DNF_3 evaluations to perform: at most 8
ValidateUserTokenOK_DNF_1 evaluations to perform: at most 8
ValidateUserTokenOK_DNF_3 test case generation -> FAILED.
ValidateUserTokenOK_DNF_1 test case generation -> FAILED.
ValidateUserTokenOK_DNF_2 evaluations to perform: at most 8
ValidateUserTokenOK_DNF_2 test case generation -> FAILED.
Elapsed time: 75890 miliseconds.

Fastest> selop ValidateUserTokenFail
Fastest> genalltt
Generating test tree for 'ValidateUserTokenFail' operation..
Fastest> showtt

ValidateUserTokenFail_VIS
  !______ValidateUserTokenFail_DNF_1
  !______ValidateUserTokenFail_DNF_2
  !______ValidateUserTokenFail_DNF_3
  !______ValidateUserTokenFail_DNF_4
  !______ValidateUserTokenFail_DNF_5
  !______ValidateUserTokenFail_DNF_6
  !______ValidateUserTokenFail_DNF_7
  !______ValidateUserTokenFail_DNF_8
  !______ValidateUserTokenFail_DNF_9
  !______ValidateUserTokenFail_DNF_10

Fastest> setfinitemodel ValidateUserTokenFail_DNF_1 -btsize 1
Fastest> setfinitemodel ValidateUserTokenFail_DNF_2 -btsize 1
Fastest> setfinitemodel ValidateUserTokenFail_DNF_3 -btsize 1
Fastest> setfinitemodel ValidateUserTokenFail_DNF_4 -btsize 1
Fastest> setfinitemodel ValidateUserTokenFail_DNF_5 -btsize 1
Fastest> setfinitemodel ValidateUserTokenFail_DNF_6 -btsize 1
Fastest> setfinitemodel ValidateUserTokenFail_DNF_7 -btsize 1
Fastest> setfinitemodel ValidateUserTokenFail_DNF_8 -btsize 1
Fastest> setfinitemodel ValidateUserTokenFail_DNF_9 -btsize 1
Fastest> setfinitemodel ValidateUserTokenFail_DNF_10 -btsize 1
Fastest> genalltca
ValidateUserTokenFail_DNF_8 evaluations to perform: at most 8
ValidateUserTokenFail_DNF_8 test case generation -> FAILED.
ValidateUserTokenFail_DNF_3 evaluations to perform: at most 8
ValidateUserTokenFail_DNF_3 test case generation -> FAILED.
ValidateUserTokenFail_DNF_5 evaluations to perform: at most 8
ValidateUserTokenFail_DNF_5 test case generation -> FAILED.
ValidateUserTokenFail_DNF_1 evaluations to perform: at most 8
ValidateUserTokenFail_DNF_1 test case generation -> FAILED.
ValidateUserTokenFail_DNF_2 evaluations to perform: at most 8
ValidateUserTokenFail_DNF_2 test case generation -> FAILED.
ValidateUserTokenFail_DNF_6 evaluations to perform: at most 8
```

```
ValidateUserTokenFail_DNF_6 test case generation -> FAILED.
ValidateUserTokenFail_DNF_4 evaluations to perform: at most 8
ValidateUserTokenFail_DNF_4 test case generation -> FAILED.
ValidateUserTokenFail_DNF_7 evaluations to perform: at most 8
ValidateUserTokenFail_DNF_7 test case generation -> FAILED.
ValidateUserTokenFail_DNF_10 evaluations to perform: at most 8
ValidateUserTokenFail_DNF_10 test case generation -> FAILED.
ValidateUserTokenFail_DNF_9 evaluations to perform: at most 8
ValidateUserTokenFail_DNF_9 test case generation -> FAILED.
Elapsed time: 37531 miliseconds.

Fastest> selop ValidateFingerOK
Fastest> genalltt
Generating test tree for 'ValidateFingerOK' operation..
Fastest> showtt

ValidateFingerOK_VIS
  !______ValidateFingerOK_DNF_1
  !______ValidateFingerOK_DNF_2
  !______ValidateFingerOK_DNF_3
  !______ValidateFingerOK_DNF_4
  !______ValidateFingerOK_DNF_5
  !______ValidateFingerOK_DNF_6
  !______ValidateFingerOK_DNF_7
  !______ValidateFingerOK_DNF_8
  !______ValidateFingerOK_DNF_9
  !______ValidateFingerOK_DNF_10

Fastest> setfinitemodel ValidateFingerOK_DNF_1 -btsize 1
Fastest> setfinitemodel ValidateFingerOK_DNF_2 -btsize 1
Fastest> setfinitemodel ValidateFingerOK_DNF_3 -btsize 1
Fastest> setfinitemodel ValidateFingerOK_DNF_4 -btsize 1
Fastest> setfinitemodel ValidateFingerOK_DNF_5 -btsize 1
Fastest> setfinitemodel ValidateFingerOK_DNF_6 -btsize 1
Fastest> setfinitemodel ValidateFingerOK_DNF_7 -btsize 1
Fastest> setfinitemodel ValidateFingerOK_DNF_8 -btsize 1
Fastest> setfinitemodel ValidateFingerOK_DNF_9 -btsize 1
Fastest> setfinitemodel ValidateFingerOK_DNF_10 -btsize 1
Fastest> genalltca
ValidateFingerOK_DNF_1 evaluations to perform: at most 8
ValidateFingerOK_DNF_1 test case generation -> FAILED.
ValidateFingerOK_DNF_6 evaluations to perform: at most 8
ValidateFingerOK_DNF_6 test case generation -> FAILED.
ValidateFingerOK_DNF_4 evaluations to perform: at most 8
ValidateFingerOK_DNF_4 test case generation -> FAILED.
ValidateFingerOK_DNF_9 evaluations to perform: at most 8
ValidateFingerOK_DNF_9 test case generation -> FAILED.
ValidateFingerOK_DNF_8 evaluations to perform: at most 8
ValidateFingerOK_DNF_8 test case generation -> FAILED.
ValidateFingerOK_DNF_10 evaluations to perform: at most 8
```

```
ValidateFingerOK_DNF_10 test case generation -> FAILED.
ValidateFingerOK_DNF_7 evaluations to perform: at most 8
ValidateFingerOK_DNF_7 test case generation -> FAILED.
ValidateFingerOK_DNF_2 evaluations to perform: at most 8
ValidateFingerOK_DNF_2 test case generation -> FAILED.
ValidateFingerOK_DNF_5 evaluations to perform: at most 8
ValidateFingerOK_DNF_5 test case generation -> FAILED.
ValidateFingerOK_DNF_3 evaluations to perform: at most 8
ValidateFingerOK_DNF_3 test case generation -> FAILED.
Elapsed time: 30031 miliseconds.

Fastest> selop EntryOK
Fastest> genalltt
Generating test tree for 'EntryOK' operation..
Fastest> showtt

EntryOK_VIS
  !______EntryOK_DNF_1
  !______EntryOK_DNF_2
  !______EntryOK_DNF_3
  !______EntryOK_DNF_4
  !______EntryOK_DNF_5
  !______EntryOK_DNF_6
  !______EntryOK_DNF_7
  !______EntryOK_DNF_8
  !______EntryOK_DNF_9
  !______EntryOK_DNF_10

Fastest> setfinitemodel EntryOK_DNF_1 -btsize 1
Fastest> setfinitemodel EntryOK_DNF_2 -btsize 1
Fastest> setfinitemodel EntryOK_DNF_3 -btsize 1
Fastest> setfinitemodel EntryOK_DNF_4 -btsize 1
Fastest> setfinitemodel EntryOK_DNF_5 -btsize 1
Fastest> setfinitemodel EntryOK_DNF_6 -btsize 1
Fastest> setfinitemodel EntryOK_DNF_7 -btsize 1
Fastest> setfinitemodel EntryOK_DNF_8 -btsize 1
Fastest> setfinitemodel EntryOK_DNF_9 -btsize 1
Fastest> setfinitemodel EntryOK_DNF_10 -btsize 1
Fastest> genalltca
EntryOK_DNF_8 evaluations to perform: at most 8
EntryOK_DNF_8 test case generation -> FAILED.
EntryOK_DNF_6 evaluations to perform: at most 8
EntryOK_DNF_6 test case generation -> FAILED.
EntryOK_DNF_5 evaluations to perform: at most 8
EntryOK_DNF_5 test case generation -> FAILED.
EntryOK_DNF_4 evaluations to perform: at most 8
EntryOK_DNF_4 test case generation -> FAILED.
EntryOK_DNF_2 evaluations to perform: at most 8
EntryOK_DNF_2 test case generation -> FAILED.
EntryOK_DNF_3 evaluations to perform: at most 8
```

```
EntryOK_DNF_3 test case generation -> FAILED.
EntryOK_DNF_10 evaluations to perform: at most 8
EntryOK_DNF_10 test case generation -> FAILED.
EntryOK_DNF_1 evaluations to perform: at most 8
EntryOK_DNF_1 test case generation -> FAILED.
EntryOK_DNF_9 evaluations to perform: at most 8
EntryOK_DNF_9 test case generation -> FAILED.
EntryOK_DNF_7 evaluations to perform: at most 8
EntryOK_DNF_7 test case generation -> FAILED.
Elapsed time: 30015 miliseconds.

Fastest> selop EntryNotAllowed
Fastest> genalltt
Generating test tree for 'EntryNotAllowed' operation..
Fastest> showtt

EntryNotAllowed_VIS
  !______EntryNotAllowed_DNF_1
  !______EntryNotAllowed_DNF_2
  !______EntryNotAllowed_DNF_3
  !______EntryNotAllowed_DNF_4
  !______EntryNotAllowed_DNF_5
  !______EntryNotAllowed_DNF_6
  !______EntryNotAllowed_DNF_7
  !______EntryNotAllowed_DNF_8
  !______EntryNotAllowed_DNF_9
  !______EntryNotAllowed_DNF_10

Fastest> setfinitemodel EntryNotAllowed_DNF_1 -btsize 1
Fastest> setfinitemodel EntryNotAllowed_DNF_2 -btsize 1
Fastest> setfinitemodel EntryNotAllowed_DNF_3 -btsize 1
Fastest> setfinitemodel EntryNotAllowed_DNF_4 -btsize 1
Fastest> setfinitemodel EntryNotAllowed_DNF_5 -btsize 1
Fastest> setfinitemodel EntryNotAllowed_DNF_6 -btsize 1
Fastest> setfinitemodel EntryNotAllowed_DNF_7 -btsize 1
Fastest> setfinitemodel EntryNotAllowed_DNF_8 -btsize 1
Fastest> setfinitemodel EntryNotAllowed_DNF_9 -btsize 1
Fastest> setfinitemodel EntryNotAllowed_DNF_10 -btsize 1
Fastest> genalltca
EntryNotAllowed_DNF_1 evaluations to perform: at most 8
EntryNotAllowed_DNF_1 test case generation -> FAILED.
EntryNotAllowed_DNF_6 evaluations to perform: at most 8
EntryNotAllowed_DNF_6 test case generation -> FAILED.
EntryNotAllowed_DNF_7 evaluations to perform: at most 8
EntryNotAllowed_DNF_7 test case generation -> FAILED.
EntryNotAllowed_DNF_2 evaluations to perform: at most 8
EntryNotAllowed_DNF_2 test case generation -> FAILED.
EntryNotAllowed_DNF_8 evaluations to perform: at most 8
EntryNotAllowed_DNF_8 test case generation -> FAILED.
EntryNotAllowed_DNF_3 evaluations to perform: at most 8
```

```
EntryNotAllowed_DNF_3 test case generation -> FAILED.
EntryNotAllowed_DNF_10 evaluations to perform: at most 8
EntryNotAllowed_DNF_10 test case generation -> FAILED.
EntryNotAllowed_DNF_9 evaluations to perform: at most 8
EntryNotAllowed_DNF_9 test case generation -> FAILED.
EntryNotAllowed_DNF_4 evaluations to perform: at most 8
EntryNotAllowed_DNF_4 test case generation -> FAILED.
EntryNotAllowed_DNF_5 evaluations to perform: at most 8
EntryNotAllowed_DNF_5 test case generation -> FAILED.
Elapsed time: 29703 miliseconds.

Fastest> selop AdminTokenTimeout
Fastest> genalltt
Generating test tree for 'AdminTokenTimeout' operation..
Fastest> showtt

AdminTokenTimeout_VIS
  !______AdminTokenTimeout_DNF_1
  !______AdminTokenTimeout_DNF_2
  !______AdminTokenTimeout_DNF_3
  !______AdminTokenTimeout_DNF_4
  !______AdminTokenTimeout_DNF_5

Fastest> setfinitemodel AdminTokenTimeout_DNF_1 -btsize 1
Fastest> setfinitemodel AdminTokenTimeout_DNF_2 -btsize 1
Fastest> setfinitemodel AdminTokenTimeout_DNF_3 -btsize 1
Fastest> setfinitemodel AdminTokenTimeout_DNF_4 -btsize 1
Fastest> setfinitemodel AdminTokenTimeout_DNF_5 -btsize 1
Fastest> genalltca
AdminTokenTimeout_DNF_1 evaluations to perform: at most 64
AdminTokenTimeout_DNF_1 test case generation -> FAILED.
AdminTokenTimeout_DNF_2 evaluations to perform: at most 64
AdminTokenTimeout_DNF_2 test case generation -> FAILED.
AdminTokenTimeout_DNF_3 evaluations to perform: at most 64
AdminTokenTimeout_DNF_3 test case generation -> FAILED.
AdminTokenTimeout_DNF_4 evaluations to perform: at most 64
AdminTokenTimeout_DNF_4 test case generation -> FAILED.
AdminTokenTimeout_DNF_5 evaluations to perform: at most 64
AdminTokenTimeout_DNF_5 test case generation -> FAILED.
Elapsed time: 110140 miliseconds.

Fastest> selop TISShutdownOp
Fastest> genalltt
Generating test tree for 'TISShutdownOp' operation..
Fastest> showtt

TISShutdownOp_VIS
  !______TISShutdownOp_DNF_1
  !______TISShutdownOp_DNF_2
```

```
Fastest> setfinitemodel TISShutdownOp_DNF_1 -btsize 1
Fastest> setfinitemodel TISShutdownOp_DNF_2 -btsize 1
Fastest> genalltca
TISShutdownOp_DNF_2 evaluations to perform: at most -128
TISShutdownOp_DNF_2 test case generation -> FAILED.
TISShutdownOp_DNF_1 evaluations to perform: at most -1024
TISShutdownOp_DNF_1 test case generation -> FAILED.
Elapsed time: 1015 miliseconds.

Fastest> selop TISStartup
Fastest> genalltt
Generating test tree for 'TISStartup' operation..
Fastest> showtt

TISStartup_VIS
  !______TISStartup_DNF_1
  !______TISStartup_DNF_2

Fastest> setfinitemodel TISStartup_DNF_1 -btsize 1
Fastest> setfinitemodel TISStartup_DNF_2 -btsize 1
Fastest> genalltca
TISStartup_DNF_1 evaluations to perform: at most 8
TISStartup_DNF_1 test case generation -> FAILED.
TISStartup_DNF_2 evaluations to perform: at most 4
TISStartup_DNF_2 test case generation -> FAILED.
Elapsed time: 6250 miliseconds.
```

5.6. Operaciones que incluyen funciones con esquemas en el dominio

La version actual de Fastest^{TM} no soporta la inclusión de funciones cuyo dominio sea de tipo esquema. Aquí se muestra un ejemplo del error que arroja la herramienta al tratar de derivar casos de prueba a partir de una operación que incluye una función de este tipo:

```
Fastest> selop TISEnrolOp
Fastest> genalltt
Generating test tree for 'TISEnrolOp' operation..
Fastest> showtt

TISEnrolOp_VIS
  !______TISEnrolOp_DNF_1
  !______TISEnrolOp_DNF_2
  !______TISEnrolOp_DNF_3
  !______TISEnrolOp_DNF_4
  !______TISEnrolOp_DNF_5
  !______TISEnrolOp_DNF_6
```

```
Fastest> setfinitemodel TISEnrolOp_DNF_1 -btsize 1
Fastest> setfinitemodel TISEnrolOp_DNF_2 -btsize 1
Fastest> setfinitemodel TISEnrolOp_DNF_3 -btsize 1
Fastest> setfinitemodel TISEnrolOp_DNF_4 -btsize 1
Fastest> setfinitemodel TISEnrolOp_DNF_5 -btsize 1
Fastest> setfinitemodel TISEnrolOp_DNF_6 -btsize 1
Fastest> genalltca
TISEnrolOp_DNF_1 evaluations to perform: at most 8
TISEnrolOp_DNF_1 test case generation -> SUCCESS.
TISEnrolOp_DNF_2 evaluations to perform: at most 8
TISEnrolOp_DNF_2 test case generation -> SUCCESS.
TISEnrolOp_DNF_6 evaluations to perform: at most 8
TISEnrolOp_DNF_6 test case generation -> SUCCESS.
TISEnrolOp_DNF_5 evaluations to perform: at most 8
TISEnrolOp_DNF_5 test case generation -> SUCCESS.
TISEnrolOp_DNF_4 evaluations to perform: at most 8
TISEnrolOp_DNF_4 test case generation -> FAILED.
Exception in thread "Thread-2" java.lang.NullPointerException
        at compserver.tcasegen.fm.TClassFiniteModel.<init>(TClassFiniteModel.java:229)
        at compserver.tcasegen.strategies.IterativeTCaseStrategy.generateAbstractTCase
        at compserver.tcasegen.TCaseGen.generateAbstractTCase(TCaseGen.java:27)
        at client.blogic.testing.tcasegen.TCaseGenClientRunner.run(TCaseGenClientRunner
        at java.lang.Thread.run(Unknown Source)
```

La clase de prueba *TISEnrolOp_DNF_3* contiene en su predicado una variable de este tipo no soportado. La variable es $enrolmentFile : ValidEnrol \pfun FLOPPY$. Podando esta clase de prueba se pueden derivar casos de las clases restantes:

```
Fastest> selop TISEnrolOp
Fastest> genalltt
Generating test tree for 'TISEnrolOp' operation..
Fastest> showtt

TISEnrolOp_VIS
  !______TISEnrolOp_DNF_1
  !______TISEnrolOp_DNF_2
  !______TISEnrolOp_DNF_3
  !______TISEnrolOp_DNF_4
  !______TISEnrolOp_DNF_5
  !______TISEnrolOp_DNF_6

Fastest> prunefrom TISEnrolOp_DNF_3

Fastest> setfinitemodel TISEnrolOp_DNF_1 -btsize 1
Fastest> setfinitemodel TISEnrolOp_DNF_2 -btsize 1
Fastest> setfinitemodel TISEnrolOp_DNF_4 -btsize 1
Fastest> setfinitemodel TISEnrolOp_DNF_5 -btsize 1
Fastest> setfinitemodel TISEnrolOp_DNF_6 -btsize 1
Fastest> genalltca
```

```
TISEnrolOp_DNF_2 evaluations to perform: at most 8
TISEnrolOp_DNF_2 test case generation -> SUCCESS.
TISEnrolOp_DNF_1 evaluations to perform: at most 8
TISEnrolOp_DNF_1 test case generation -> SUCCESS.
TISEnrolOp_DNF_5 evaluations to perform: at most 8
TISEnrolOp_DNF_5 test case generation -> SUCCESS.
TISEnrolOp_DNF_4 evaluations to perform: at most 8
TISEnrolOp_DNF_4 test case generation -> FAILED.
TISEnrolOp_DNF_6 evaluations to perform: at most 8
TISEnrolOp_DNF_6 test case generation -> SUCCESS.
Elapsed time: 17797 miliseconds.
```

Las clases y casos de prueba obtenidas se listan en los apéndices.

Las operación *ValidateEnrolmentData* se compone a partir de tres operaciones que ya fueron testeadas.

Las tres siguientes operaciones:

WriteUserTokenOK
WriteUserTokenFail
WriteUserToken

usan una función con dominio esquema: $goodT : Token \pfun TOKENTRY$. El problema con estas operaciones es que usan la variable $goodT$ en todas las clases del árbol y no puede aplicarse la solución utilizada en el ejemplo anterior.

5.7. Operaciones que arrojan errores

Fastest™ genera una excepción al tratar de generar casos de prueba para algunos esquemas. Se analizó el problema y la conclusión es que es casi imposible de solucionar porque la forma normal disyuntiva genera cientos o miles de millones de hojas. Esto hace que la máquina se quede sin memoria. Estos esquemas no especifican una unidad de código sino todo un subsistema en si, y Fastest™ está pensada para testing de unidad.

Ejemplo:

```
Fastest> selop TISEarlyUpdate
Fastest> genalltt
Generating test tree for 'TISEarlyUpdate' operation..
java.lang.reflect.InvocationTargetException
        at sun.reflect.NativeMethodAccessorImpl.invoke0(Native Method)
        at sun.reflect.NativeMethodAccessorImpl.invoke(Unknown Source)
        at sun.reflect.DelegatingMethodAccessorImpl.invoke(Unknown Source)
        at java.lang.reflect.Method.invoke(Unknown Source)
        at client.blogic.management.ii.IIOrder.run(IIOrder.java:84)
        at java.lang.Thread.run(Unknown Source)
```

```
Caused by: java.lang.NullPointerException
        at client.blogic.testing.ttree.strategies.
IterativeTTreeStrategy.generateTTree(IterativeTTreeStrategy.java:36)
        at client.blogic.testing.ttree.TTreeGen.generateTTree2(TTreeGen.java:151)
        at client.blogic.testing.ttree.TTreeGen.manageEvent(TTreeGen.java:82)
        ... 6 more
```

Las operaciones que se listan abajo arrojan el mismo error. Componen esquemas que ya han sido testeados y por lo tanto no agregarían casos nuevos:

TISEarlyUpdate
TISUpdate
TISPoll
TISValidateUserToken
TISReadFinger
TISValidateFinger
TISValidateEntry
TISCompleteFailedAccess

5.8. Eliminación automática de clases de prueba vacías

Una mejora de Fastest^{TM} es la eliminación automática de clases de prueba vacías. Al realizar este trabajo, la versión utilizada de la herramienta no contaba con esa funcionalidad, pero sí se disponía de scripts que realizan esta tarea; lo que mejora notablemente la eficiencia de Fastest^{TM} ya que no tiene que buscar en clases vacías y aumenta la productividad del tester, quien no necesita verificar si las clases están vacías o no.

Son dos scritps: `czt2zeves` que traduce el Latex de CZT en el Latex de ZEVES; y `fze` que llama al anterior y a ZEVES para tratar de eliminar las clases de prueba vacías. Estos scripts funcionan solo sobre Linux y requieren tener instalado el programa ZEVES (en modo línea de comandos; la interfaz gráfica no es necesaria).

Para poder utilizar estos scripts, se instalaron junto con ZEVES en un servidor de Flowgate Consulting y se eliminaron las siguientes operaciones de la especificación ya que ZEVES no las puede interpretar (ésto no presenta un problema porque estas operaciones no agregan casos de prueba):

$$
\begin{aligned}
&TISEarlyUpdate == UpdateLatch \land UpdateAlarm \\
&\quad \land [\ RealWorldChanges \mid screen' = screen \land display' = display\ ] \\
&\quad \land [\Delta IDStation \mid currentDisplay = currentDisplay'] \\
&\quad \land [\Xi UserToken;\ \Xi AdminToken;\ \Xi Finger;\ \Xi Floppy; \\
&\qquad \Xi Keyboard;\ \Xi Config;\ \Xi Stats; \\
&\qquad \Xi KeyStore;\ \Xi Admin;\ \Xi Internal; \\
&\qquad (AddElementsToLog \lor \Xi AuditLog) \mid true]
\end{aligned}
$$

$$
\begin{aligned}
&TISUpdate == \\
&UpdateLatch \land UpdateAlarm \land UpdateDisplay \land UpdateScreen \\
&\quad \land [\Xi UserToken;\ \Xi AdminToken;\ \Xi Finger;\ \Xi Floppy; \\
&\quad \Xi Keyboard;\ \Xi Config;\ \Xi Stats; \\
&\quad \Xi KeyStore;\ \Xi Admin;\ \Xi Internal; \\
&\quad (AddElementsToLog \lor \Xi AuditLog) \mid true]
\end{aligned}
$$

Los pasos a seguir para eliminar clases de prueba vacías fueron los siguientes:

1. Ejecutar Fastest^{TM} en la máquina local hasta generar las clases de prueba
2. Exportar las clases de prueba con *showsch -tcl -o clases.tex*
3. Copiar este archivo al servidor
4. Invocar `fze` pasándole dos parámetros: primero el nombre del archivo Latex que tiene la especificación y segundo el archivo donde se exportaron las clases de prueba. Para esto se ejecutan los siguientes comandos en el servidor:
   `(time fze especificacion.tex clases.tex) < poda.txt 2< tiempos.txt`
   Se antepone el comando `time` para poder medir el tiempo que demora calcular las podas, estos tiempos se guardan en el archivo `tiempos.txt`. Las podas necesarias se obtienen en el archivo `poda.txt`. La salida de `fze` es una lista de comandos prunefrom, cada uno seguido del nombre de una clase de prueba que ZEVES dice que es vacía.
5. Copiar y pegar esa lista de comandos en Fastest
6. Correr genalltca

`fze` tira un error si ZEVES no puede chequear los tipos de la especificación o de las clases de prueba. Esto en general se debe a las diferencias que hay entre CZT y ZEVES. El error es arrojado en un archivo que luego se analiza para modificar la especificación y volver a ejecutar los pasos enunciados arriba. Aun así puede haber clases de prueba para las cuales Fastest^{TM} no encuentra un caso de prueba. Esto puede deberse a: (a) hay algún error en Fastest^{TM} o CZT; (b) la clase es vacía pero ZEVES no pudo demostrarlo; (c) la clase no es vacía pero el modelo finito que usa Fastest^{TM} no permite encontrar un caso. Cada caso se analiza individualmente.

A continuación se muestra la ejecución de Fastest^{TM} en conjunto con los scripts de eliminacipon automática de clases de prueba vacías para las operaciones de sondeo y actualización de alarma:

```
loadspec D:\t_zeves.tex
selop PollUserToken
addtactic PollUserToken FT userToken
selop PollAdminToken
addtactic PollAdminToken FT adminToken
selop PollFinger
addtactic PollFinger FT finger
selop PollFloppy
addtactic PollFloppy FT floppy
selop PollKeyboard
```

```
addtactic PollKeyboard FT keyboard
selop UpdateAlarm
addtactic UpdateAlarm FT doorAlarm
addtactic UpdateAlarm FT auditAlarm
genalltt
showsch -tcl -o clases.tex
blieber@localhost ~ $ time fze t_zeves.tex clases.tex
Filtrando texto Z en la especificacion...
Filtrando texto Z en las clases de prueba...
Traduciendo a ZEVES la especificacion...
Escribiendo los teoremas de eliminacion...
Invocando ZEVES para que realice las pruebas de los teoremas...
Generando los comandos de poda...
prunefrom PollUserToken_DNF_1
prunefrom PollUserToken_FT_1
prunefrom PollUserToken_FT_2
prunefrom PollUserToken_FT_3
prunefrom PollUserToken_FT_4
prunefrom PollUserToken_FT_5
prunefrom PollUserToken_FT_6
prunefrom PollUserToken_FT_7
prunefrom PollUserToken_DNF_3
prunefrom PollUserToken_FT_13
prunefrom PollUserToken_FT_14
prunefrom PollUserToken_FT_15
prunefrom PollUserToken_FT_16
prunefrom PollUserToken_FT_17
prunefrom PollUserToken_FT_18
prunefrom PollUserToken_FT_19
prunefrom PollUserToken_FT_26
prunefrom PollUserToken_FT_27
prunefrom PollUserToken_FT_28
prunefrom PollUserToken_FT_29
prunefrom PollUserToken_FT_30
prunefrom PollUserToken_FT_32
prunefrom PollUserToken_FT_33
prunefrom PollUserToken_FT_34
prunefrom PollUserToken_FT_35
prunefrom PollUserToken_FT_36
prunefrom PollUserToken_DNF_7
prunefrom PollUserToken_FT_37
prunefrom PollUserToken_FT_38
prunefrom PollUserToken_FT_39
prunefrom PollUserToken_FT_40
prunefrom PollUserToken_FT_41
prunefrom PollUserToken_FT_42
prunefrom PollUserToken_DNF_8
prunefrom PollUserToken_FT_43
prunefrom PollUserToken_FT_44
prunefrom PollUserToken_FT_45
```

```
prunefrom PollUserToken_FT_46
prunefrom PollUserToken_FT_47
prunefrom PollUserToken_FT_48
prunefrom PollFloppy_DNF_1
prunefrom PollFloppy_FT_1
prunefrom PollFloppy_FT_2
prunefrom PollFloppy_FT_3
prunefrom PollFloppy_FT_4
prunefrom PollFloppy_FT_5
prunefrom PollFloppy_FT_6
prunefrom PollFloppy_FT_7
prunefrom PollFloppy_FT_8
prunefrom PollFloppy_FT_9
prunefrom PollFloppy_FT_10
prunefrom PollFloppy_FT_11
prunefrom PollFloppy_DNF_3
prunefrom PollFloppy_FT_21
prunefrom PollFloppy_FT_22
prunefrom PollFloppy_FT_23
prunefrom PollFloppy_FT_24
prunefrom PollFloppy_FT_25
prunefrom PollFloppy_FT_26
prunefrom PollFloppy_FT_27
prunefrom PollFloppy_FT_28
prunefrom PollFloppy_FT_29
prunefrom PollFloppy_FT_30
prunefrom PollFloppy_FT_31
prunefrom PollFloppy_FT_42
prunefrom PollFloppy_FT_43
prunefrom PollFloppy_FT_44
prunefrom PollFloppy_FT_45
prunefrom PollFloppy_FT_46
prunefrom PollFloppy_FT_47
prunefrom PollFloppy_FT_48
prunefrom PollFloppy_FT_49
prunefrom PollFloppy_FT_50
prunefrom PollFloppy_FT_52
prunefrom PollFloppy_FT_53
prunefrom PollFloppy_FT_54
prunefrom PollFloppy_FT_55
prunefrom PollFloppy_FT_56
prunefrom PollFloppy_FT_57
prunefrom PollFloppy_FT_58
prunefrom PollFloppy_FT_59
prunefrom PollFloppy_FT_60
prunefrom PollFloppy_DNF_7
prunefrom PollFloppy_FT_61
prunefrom PollFloppy_FT_62
prunefrom PollFloppy_FT_63
prunefrom PollFloppy_FT_64
```

```
prunefrom PollFloppy_FT_65
prunefrom PollFloppy_FT_66
prunefrom PollFloppy_FT_67
prunefrom PollFloppy_FT_68
prunefrom PollFloppy_FT_69
prunefrom PollFloppy_FT_70
prunefrom PollFloppy_DNF_8
prunefrom PollFloppy_FT_71
prunefrom PollFloppy_FT_72
prunefrom PollFloppy_FT_73
prunefrom PollFloppy_FT_74
prunefrom PollFloppy_FT_75
prunefrom PollFloppy_FT_76
prunefrom PollFloppy_FT_77
prunefrom PollFloppy_FT_78
prunefrom PollFloppy_FT_79
prunefrom PollFloppy_FT_80
prunefrom PollFinger_DNF_1
prunefrom PollFinger_FT_1
prunefrom PollFinger_FT_2
prunefrom PollFinger_FT_3
prunefrom PollFinger_FT_4
prunefrom PollFinger_FT_5
prunefrom PollFinger_FT_6
prunefrom PollFinger_DNF_3
prunefrom PollFinger_FT_11
prunefrom PollFinger_FT_12
prunefrom PollFinger_FT_13
prunefrom PollFinger_FT_14
prunefrom PollFinger_FT_15
prunefrom PollFinger_FT_16
prunefrom PollFinger_FT_22
prunefrom PollFinger_FT_23
prunefrom PollFinger_FT_24
prunefrom PollFinger_FT_25
prunefrom PollFinger_FT_27
prunefrom PollFinger_FT_28
prunefrom PollFinger_FT_29
prunefrom PollFinger_FT_30
prunefrom PollFinger_DNF_7
prunefrom PollFinger_FT_31
prunefrom PollFinger_FT_32
prunefrom PollFinger_FT_33
prunefrom PollFinger_FT_34
prunefrom PollFinger_FT_35
prunefrom PollFinger_DNF_8
prunefrom PollFinger_FT_36
prunefrom PollFinger_FT_37
prunefrom PollFinger_FT_38
prunefrom PollFinger_FT_39
```

```
prunefrom PollFinger_FT_40
prunefrom UpdateAlarm_FT_4
prunefrom UpdateAlarm_FT_2
prunefrom UpdateAlarm_FT_5
prunefrom UpdateAlarm_FT_6
prunefrom UpdateAlarm_DNF_2
prunefrom UpdateAlarm_FT_7
prunefrom UpdateAlarm_FT_9
prunefrom UpdateAlarm_FT_10
prunefrom UpdateAlarm_FT_8
prunefrom UpdateAlarm_FT_11
prunefrom UpdateAlarm_FT_12
prunefrom UserTokenTorn_DNF_5
prunefrom UpdateAlarm_FT_20
prunefrom UpdateAlarm_FT_23
prunefrom UpdateAlarm_FT_24
prunefrom UpdateAlarm_FT_27
prunefrom UpdateAlarm_FT_29
prunefrom PollKeyboard_DNF_1
prunefrom PollKeyboard_FT_1
prunefrom PollKeyboard_FT_2
prunefrom PollKeyboard_FT_3
prunefrom PollKeyboard_FT_4
prunefrom PollKeyboard_FT_5
prunefrom PollKeyboard_FT_6
prunefrom PollKeyboard_DNF_2
prunefrom PollKeyboard_FT_7
prunefrom PollKeyboard_FT_8
prunefrom PollKeyboard_FT_9
prunefrom PollKeyboard_FT_10
prunefrom PollKeyboard_FT_11
prunefrom PollKeyboard_FT_12
prunefrom PollKeyboard_DNF_3
prunefrom PollKeyboard_FT_13
prunefrom PollKeyboard_FT_14
prunefrom PollKeyboard_FT_15
prunefrom PollKeyboard_FT_16
prunefrom PollKeyboard_FT_17
prunefrom PollKeyboard_FT_18
prunefrom PollKeyboard_FT_19
prunefrom PollKeyboard_FT_26
prunefrom PollKeyboard_FT_27
prunefrom PollKeyboard_FT_28
prunefrom PollKeyboard_FT_29
prunefrom PollKeyboard_FT_30
prunefrom PollKeyboard_DNF_6
prunefrom PollKeyboard_FT_31
prunefrom PollKeyboard_FT_32
prunefrom PollKeyboard_FT_33
prunefrom PollKeyboard_FT_34
```

```
prunefrom PollKeyboard_FT_35
prunefrom PollKeyboard_FT_36
prunefrom PollKeyboard_DNF_7
prunefrom PollKeyboard_FT_37
prunefrom PollKeyboard_FT_38
prunefrom PollKeyboard_FT_39
prunefrom PollKeyboard_FT_40
prunefrom PollKeyboard_FT_41
prunefrom PollKeyboard_FT_42
prunefrom PollKeyboard_DNF_8
prunefrom PollKeyboard_FT_43
prunefrom PollKeyboard_FT_44
prunefrom PollKeyboard_FT_45
prunefrom PollKeyboard_FT_46
prunefrom PollKeyboard_FT_47
prunefrom PollKeyboard_FT_48
prunefrom PollAdminToken_DNF_1
prunefrom PollAdminToken_FT_1
prunefrom PollAdminToken_FT_2
prunefrom PollAdminToken_FT_3
prunefrom PollAdminToken_FT_4
prunefrom PollAdminToken_FT_5
prunefrom PollAdminToken_FT_6
prunefrom PollAdminToken_FT_7
prunefrom PollAdminToken_DNF_3
prunefrom PollAdminToken_FT_13
prunefrom PollAdminToken_FT_14
prunefrom PollAdminToken_FT_15
prunefrom PollAdminToken_FT_16
prunefrom PollAdminToken_FT_17
prunefrom PollAdminToken_FT_18
prunefrom PollAdminToken_FT_19
prunefrom PollAdminToken_FT_26
prunefrom PollAdminToken_FT_27
prunefrom PollAdminToken_FT_28
prunefrom PollAdminToken_FT_29
prunefrom PollAdminToken_FT_30
prunefrom PollAdminToken_FT_32
prunefrom PollAdminToken_FT_33
prunefrom PollAdminToken_FT_34
prunefrom PollAdminToken_FT_35
prunefrom PollAdminToken_FT_36
prunefrom PollAdminToken_DNF_7
prunefrom PollAdminToken_FT_37
prunefrom PollAdminToken_FT_38
prunefrom PollAdminToken_FT_39
prunefrom PollAdminToken_FT_40
prunefrom PollAdminToken_FT_41
prunefrom PollAdminToken_FT_42
prunefrom PollAdminToken_DNF_8
```

```
prunefrom PollAdminToken_FT_43
prunefrom PollAdminToken_FT_44
prunefrom PollAdminToken_FT_45
prunefrom PollAdminToken_FT_46
prunefrom PollAdminToken_FT_47
prunefrom PollAdminToken_FT_48

real    4m23.440s
user    3m57.940s
sys     0m5.080s
```

`fze` detecta todas las podas identificadas manualmente en los capítulos 5.2 y 5.3 y muchas más, lo que demuestra la eficiencia de ejecutar este paso en la derivación de casos de prueba. El tiempo que demora la herramienta en calcular estas podas es menos de 5 minutos.

Capítulo 6

Conclusión y Trabajos Futuros

En el proyecto Tokeneer no se realizó testing de unidad. La razón por la cual Praxis no realiza testing de unidad o testing de módulos es que consideran que hay suficiente superposición en la clase de errores que se detectan efectivamente a través del análisis estático y el testing de unidad [6]. Como realizan análisis estático, no encuentran beneficios en realizar testing de unidad. Sin embargo, el análisis estático no elimina la necesidad de llevar adelante testing de sistema [10]. El testing de sistema debería encontrar los errores que no hubiesen sido detectados por el análisis estático.

Praxis considera el testing de sistema un ejercicio muy valioso ya que determina si el sistema está integrado satisfactoriamente y prueba el comportamiento del sistema contra la especificación y en última instancia, los requerimientos. La dependencia en el testing de sistema conlleva a desarrollar sistemas incrementalmente de tal manera de obtener un sistema integrado en una etapa temprana, aunque no totalmente funcional. Este enfoque presenta el beneficio de minimizar los riesgos del desarrollo realizando la integración anticipadamente.

El testing de sistema fue derivado a mano a partir del diseño formal, construyendo escenarios que cubriesen todas las condiciones encontradas en el diseño. No se utilizó ninguna herramienta para soportar esta etapa. Se prepararon 33 casos de test que combinan los varios módulos del diseño formal [8].

Por el momento Fastest^{TM} solo soporta testing de unidad pero en una etapa futura está contemplado aumentar el poder de la herramienta para cubrir testing de integración y de sistema. Por otro lado, la especificación de Tokeneer está escrita de manera incremental, agrupando submódulos que definen partes del sistema hasta especificar el sistema total. Por lo tanto, al probar los módulos que definen subsistemas se están probando los escenarios preparados en el testing de sistema realizado para el proyecto Tokeneer. No se han podido probar algunas de las operaciones totales que agrupan los submódulos de Tokeneer y poder comparar éstos con el testing derivado por Praxis. Fundamentalmente la razón por la cual se encuentran errores al tratar de generar estos casos de prueba es que Fastest^{TM} está pensada para testing de unidad y estos casos no especifican una unidad de código sino todo un subsistema.

Fastest^{TM} automatiza el testing y obvia la necesidad de realizar la tarea de construir casos de prueba manualmente, reduciendo los costos de desarrollo significativamente. En este trabajo se ha logrado derivar casos de prueba para la mayoría de las operaciones parciales que componen la especificación del sistema Tokeneer, aumentando el testing producido para Tokeneer y demostrando que la herramienta puede ser utilizada en especificaciones de gran tamaño.

A continuación se muestra una tabla con algunas de las operaciones a partir de las cuales se realizaron podas manuales y con ayuda de los scripts que usan ZEVES. Se muestra el número de clases que fueron podadas a mano y con ZEVES; y los tiempos que demoraron Fastest y ZEVES en generar los casos de prueba y las podas:

Operación	**Hojas Iniciales**	**Hojas Podadas a Mano**	**Hojas Podadas con ZEVES**	**Tiempo en Fastest (s)**	**Tiempo en ZEVES (s)**
PollUserToken	48	24	40	1.563	42.3
PollAdminToken	48	24	40	1.563	42.3
PollFinger	40	24	34	1.563	42.3
PollFloppy	80	36	64	1.563	42.3
PollKeyboard	48	24	34	1.563	42.3
UpdateAlarm	30	12	16	1.265	42.3

Abajo se muestra una tabla con todas las operaciones a partir de las cuales se derivaron clases y casos de prueba y el tiempo que demoró la herramienta en generar éstos. El número total de clases de pruebas obtenido es 550, y el número de casos 168. El tiempo promedio que demora Fastest^{TM} en derivar casos de prueba para las operaciones de Tokeneer es de 372.246 segundos (aproximadamente 6 minutos).

Nro	Operación	Clases	Casos	Tiempo en segundos
1	PollUserToken	48	6	1.563
2	PollAdminToken	48	6	1.563
3	PollFinger	40	5	1.563
4	PollFloppy	80	10	1.563
5	PollKeyboard	48	5	1.563
6	UpdateAlarm	30	4	1.265
7	LogChange	3	2	779.015
8	UserEntryContext	10	10	204.671
9	UserTokenTorn	10	10	240.313
10	ReadUserToken	10	5	417.953
11	TISReadUserToken	10	5	433.844
12	ReadFingerOK	10	5	48.328
13	FingerTimeout	10	5	45.906
14	ValidateFingerFail	10	5	45.641
15	UnlockDoorOK	10	5	45.782
16	WaitingTokenRemoval	2	2	146.859
17	TokenRemovalTimeout	10	5	342.625
18	TISUnlockDoor	10	5	341.359
19	FailedAccessTokenRemoved	10	5	45.469
20	ReadEnrolmentData	2	2	249.656
21	CompleteFailedEnrolment	2	2	246.312
22	AdminTokenTear	5	5	161.09
23	BadAdminTokenTear	5	5	128.563
24	LoginAborted	5	5	20.281
25	TISValidateAdminToken	7	5	2.8062
26	TISCompleteFailedAdminLogon	2	2	9.063
27	TISAdminLogon	17	7	63.016
28	BadAdminLogout	5	5	1531.750
29	TokenRemovedAdminLogout	5	5	48.860
30	TISCompleteTimeoutAdminLogout	2	2	66.578
31	BioCheckRequired	10	0	31.875
32	BioCheckNotRequired	0	10	7854.032
33	ValidateUserTokenOK	20	0	75.890
34	ValidateUserTokenFail	10	0	37.531
35	ValidateFingerOK	10	0	30.031
36	EntryOK	10	0	30.015
37	EntryNotAllowed	10	0	29.703
38	AdminTokenTimeout	5	0	110.140
39	TISShutdownOp	2	2	1.015
40	TISStartup	2	2	6.250
41	TISEnrolOp	5	4	17.797

Trabajos futuros para ampliar el poder de Fastest^{TM} podrían incluir alternativas en el algoritmo que usa la herramienta para calcular la táctica de forma normal disyuntiva más eficientemente orientadas a especificaciones Z. También sería útil automatizar cada uno de los pasos realizados en la adaptación de la especificación mostrados en este trabajo o adaptar la herramienta para aceptar distintos estilos de especificaciones Z.

Apéndice A

Especificación de Tokeneer adaptada para Fastest^{TM}

$[USER]$

$[FINGERPRINTTEMPLATE]$

$[KEYPART]$

$[TOKENID]$

$[Audit]$

$FINGERPRINT ::= FP1 \mid FP2 \mid FP3$

$PRESENCE ::= present \mid absent$

$CLASS ::= unmarked \mid unclassified \mid restricted \mid confidential \mid secret \mid topsecret$

$PRIVILEGE ::= userOnly \mid guard \mid securityOfficer \mid auditManager$

$DOOR ::= open \mid closed$
$LATCH ::= unlocked \mid locked$
$ALARM ::= silent \mid alarming$

$DISPLAYMESSAGE ::= blank \mid welcome \mid insertFinger \mid openDoor \mid wait \mid$
$removeToken \mid tokenUpdateFailed \mid doorUnlocked$

$ADMINOP ::= archiveLog \mid updateConfigData \mid overrideLock \mid shutdownOp$

$$KEYBOARD ::= noKB \mid badKB \mid keyedOpsArchiveLog \mid keyedOpsUpdateConfigData \mid keyedOpsOverrideLock \mid keyedOpsShutdownOp$$

$$STATUS ::= quiescent \mid gotUserToken \mid waitingFinger \mid gotFinger \mid waitingUpdateToken \mid waitingEntry \mid waitingRemoveTokenSuccess \mid waitingRemoveTokenFail$$

$$ENCLAVESTATUS ::= notEnrolled \mid waitingEnrol \mid waitingEndEnrol \mid enclaveQuiescent \mid gotAdminToken \mid waitingRemoveAdminTokenFail \mid waitingStartAdminOp \mid waitingFinishAdminOp \mid shutdown$$

$$SCREENTEXT ::= clear \mid welcomeAdmin \mid busy \mid removeAdminToken \mid closeDoor \mid requestAdminOp \mid doingOp \mid invalidRequest \mid invalidData \mid insertEnrolmentData \mid validatingEnrolmentData \mid enrolmentFailed \mid archiveFailed \mid insertBlankFloppy \mid insertConfigData \mid displayStats \mid displayConfigData$$

$$FLOPPY ::= noFloppy \mid emptyFloppy \mid badFloppy \mid enrolmentFile1 \mid enrolmentFile2 \mid enrolmentFile3 \mid auditFile \mid configFile1 \mid configFile2 \mid configFile3$$

$$FINGERPRINTTRY ::= noFP \mid badFP \mid goodFP1 \mid goodFP2 \mid goodFP3$$
$$TOKENTRY ::= noT \mid badT \mid goodT1 \mid goodT2 \mid goodT3 \mid goodThe$$

$$TIME == \mathbb{N}$$

$$zeroTime == 0$$

$$ADMINPRIVILEGE == \{guard, auditManager, securityOfficer\}$$

$$Screen == SCREENTEXT \times SCREENTEXT \times SCREENTEXT$$

Clearance

$$class : CLASS$$

MinClearance

$$minClearance : Clearance \times Clearance \rightarrow Clearance$$

FingerprintTemplate

$$template : FINGERPRINTTEMPLATE$$

CertificateId

$ISSUER : \mathbb{P}\ USER$
$issuer : USER$

$issuer \in ISSUER$

Certificate

$iden : CertificateId$
$validityPeriod : \mathbb{P}\ TIME$
$isValidatedBy : \mathbb{P}\ KEYPART$

CertificateInv

$Certificate$

$\#isValidatedBy \leq 1$

IDCert

$Certificate$
$subject : USER$
$subjectPubK : KEYPART$

CAIdCert

$IDCert$

CAIdCertInv

$CAIdCert$

$isValidatedBy = \{\ subjectPubK\ \}$

AttCertificate

$Certificate$
$baseCertId : CertificateId$
$tokenID : TOKENID$

PrivCert

$AttCertificate$
$role : PRIVILEGE$
$clearance : Clearance$

AuthCert

$AttCertificate$
$role : PRIVILEGE$
$clearance : Clearance$

IandACert

$AttCertificate$
$template : FingerprintTemplate$

Token

$tokenID : TOKENID$
$idCert : IDCert$
$privCert : PrivCert$
$iandACert : IandACert$
$authCert : \mathbb{P}\, AuthCert$

TokenInv

$Token$

$\#authCert \leq 1$

ValidToken

$Token$

ValidTokenInv

$ValidToken$

$privCert.baseCertId = idCert.iden$
$iandACert.baseCertId = idCert.iden$
$privCert.tokenID = tokenID$
$iandACert.tokenID = tokenID$

TokenWithValidAuth

$Token$
$theAuthCert : \mathbb{P}\, AuthCert \rightarrow AuthCert$

TokenWithValidAuthInv

$TokenWithValidAuth$

$authCert \neq \varnothing$
$\quad \wedge\ (theAuthCert\ authCert).tokenID = tokenID$
$\quad \wedge\ (theAuthCert\ authCert).baseCertId = idCert.iden$

CurrentToken

$ValidToken$
$now : TIME$

CurrentTokenInv

CurrentToken

$now \in idCert.validityPeriod$
$\quad \cap\ privCert.validityPeriod$
$\quad \cap\ iandACert.validityPeriod$

Enrol

$idStationCert : IDCert$
$issuerCerts : \mathbb{P}\ IDCert$

EnrolInv

Enrol

$idStationCert \in issuerCerts$

ValidEnrol

Enrol
$theKEYPART : \mathbb{P}\ KEYPART \rightarrow KEYPART$

ValidEnrolInv

ValidEnrol

$issuerCerts \cap CAIdCert \neq \varnothing$
$\forall\ cert : issuerCerts \bullet$
$\quad cert.isValidatedBy \neq \varnothing$
$\quad \land\ (\exists\ issuerCert : issuerCerts \bullet issuerCert \in CAIdCert$
$\quad\quad \land\ theKEYPART\ \ cert.isValidatedBy = issuerCert.subjectPubK$
$\quad\quad \land\ cert.iden.issuer = issuerCert.subject)$

MaxSupportedLogSize

$maxSupportedLogSize : \mathbb{N}$

Config

MaxSupportedLogSize
$alarmSilentDuration, latchUnlockDuration : TIME$
$tokenRemovalDuration : TIME$
$enclaveClearance : Clearance$
$authPeriod : PRIVILEGE \rightarrow TIME \rightarrow \mathbb{P}\ TIME$
$entryPeriod : PRIVILEGE \rightarrow CLASS \rightarrow \mathbb{P}\ TIME$
$minPreservedLogSize : \mathbb{N}$
$alarmThresholdSize : \mathbb{N}$

ConfigInv

Config

$alarmThresholdSize < minPreservedLogSize$
$minPreservedLogSize \leq maxSupportedLogSize$

SizeAx

$$sizeElement : Audit \rightarrow \mathbb{N}$$
$$sizeLog : \mathbb{P}\, Audit \rightarrow \mathbb{N}$$

SizeAxInv

SizeAx

$$sizeLog\varnothing = 0$$
$$\forall\, log : \mathbb{P}\, Audit;\ entry : Audit \bullet$$
$$\quad entry \in log \Rightarrow sizeLog\ log =$$
$$\quad sizeLog\ (log \setminus \{entry\}) + sizeElement\ entry$$

AuditLog

$$auditLog : \mathbb{P}\, Audit$$
$$auditAlarm : ALARM$$

LogAx

$$oldestLogTime : \mathbb{P}\, Audit \rightarrow TIME$$
$$newestLogTime : \mathbb{P}\, Audit \rightarrow TIME$$

LogAxInv

LogAx

$$\forall\, A, B : \mathbb{P}\, Audit \bullet$$
$$\quad newestLogTime(A \cup B) \geq newestLogTime\ A$$
$$\quad \land\ oldestLogTime(A \cup B) \leq oldestLogTime\ A$$

KeyStore

$$ISSUER : \mathbb{P}\, USER$$
$$issuerKey : USER \pfun KEYPART$$
$$ownName : \mathbb{P}\, USER$$
$$theISSUER : \mathbb{P}\, USER \rightarrow USER$$

$$\mathrm{dom}\, issuerKey = ISSUER$$
$$ownName \subseteq ISSUER$$
$$\mathrm{dom}\, theISSUER \subseteq \mathbb{P}\, ISSUER$$
$$\mathrm{ran}\, theISSUER = ISSUER$$

KeyStoreInv

KeyStore

$$ownName \neq \varnothing \Rightarrow theISSUER\ ownName \in \mathrm{dom}\, issuerKey$$
$$\#ownName \leq 1$$

Stats

$successEntry : \mathbb{N}$
$failEntry : \mathbb{N}$
$successBio : \mathbb{N}$
$failBio : \mathbb{N}$

TISControlledRealWorld

$latch : LATCH$
$alarm : ALARM$
$display : DISPLAYMESSAGE$
$screen : Screen$

TISMonitoredRealWorld

$now : TIME$
$door : DOOR$
$finger : FINGERPRINTTRY$
$userToken, adminToken : TOKENTRY$
$floppy : FLOPPY$
$keyboard : KEYBOARD$

$$RealWorld == TISControlledRealWorld \land TISMonitoredRealWorld$$

Admin

$rolePresent : \mathbb{P}\, ADMINPRIVILEGE$
$availableOps : \mathbb{P}\, ADMINOP$
$currentAdminOp : \mathbb{P}\, ADMINOP$
$theADMINPRIVILEGE : \mathbb{P}\, ADMINPRIVILEGE \rightarrow ADMINPRIVILEGE$
$theADMINOP : \mathbb{P}\, ADMINOP \rightarrow ADMINOP$

AdminInv

Admin

$rolePresent = \varnothing \Rightarrow availableOps = \varnothing$
$(rolePresent \neq \varnothing \land theADMINPRIVILEGE\ rolePresent = guard) \Rightarrow$
$\quad availableOps = \{overrideLock\}$
$(rolePresent \neq \varnothing \land theADMINPRIVILEGE\ rolePresent = auditManager) \Rightarrow$
$\quad availableOps = \{archiveLog\}$
$(rolePresent \neq \varnothing \land theADMINPRIVILEGE\ rolePresent = securityOfficer) \Rightarrow$
$\quad availableOps = \{updateConfigData, shutdownOp\}$
$currentAdminOp \neq \varnothing \Rightarrow$
$\quad (theADMINOP\ currentAdminOp \in availableOps \land rolePresent \neq \varnothing)$
$\#rolePresent \leq 1$
$\#currentAdminOp \leq 1$

DoorLatchAlarm

$currentTime : TIME$
$currentDoor : DOOR$
$currentLatch : LATCH$
$doorAlarm : ALARM$
$latchTimeout : TIME$
$alarmTimeout : TIME$

DoorLatchAlarmInv

$DoorLatchAlarm$

$currentLatch = locked \Leftrightarrow currentTime \geq latchTimeout$
$doorAlarm = alarming \Leftrightarrow$
$\quad (currentDoor = open$
$\quad\quad \wedge\ currentLatch = locked$
$\quad\quad \wedge\ currentTime \geq alarmTimeout)$

UserToken

$currentUserToken : TOKENTRY$
$userTokenPresence : PRESENCE$

AdminToken

$currentAdminToken : TOKENTRY$
$adminTokenPresence : PRESENCE$

Finger

$currentFinger : FINGERPRINTTRY$
$fingerPresence : PRESENCE$

Floppy

$currentFloppy : FLOPPY$
$writtenFloppy : FLOPPY$
$floppyPresence : PRESENCE$

Keyboard

$currentKeyedData : KEYBOARD$
$keyedDataPresence : PRESENCE$

Internal

$status : STATUS$
$enclaveStatus : ENCLAVESTATUS$
$tokenRemovalTimeout : TIME$

IDStation

$$
\begin{array}{l}
UserToken \\
AdminToken \\
Finger \\
DoorLatchAlarm \\
Floppy \\
Keyboard \\
Config \\
Stats \\
KeyStore \\
Admin \\
AuditLog \\
Internal \\
currentDisplay : DISPLAYMESSAGE \\
currentScreen : Screen
\end{array}
$$

IDStationInv

$$
\begin{array}{l}
IDStation \\
goodT : Token \pfun TOKENTRY \\
\hline
status \in \{\ gotFinger, waitingFinger, waitingUpdateToken, waitingEntry\ \} \Rightarrow \\
\quad ((\exists\, ValidToken \bullet goodT(\theta\, ValidToken) = currentUserToken) \\
\qquad \lor (\exists\, TokenWithValidAuth \bullet goodThe = currentUserToken)) \\
rolePresent \neq \varnothing \Rightarrow \\
\quad (\exists\, TokenWithValidAuth \bullet goodThe = currentAdminToken) \\
enclaveStatus \notin \{\ notEnrolled, waitingEnrol, waitingEndEnrol\ \} \Rightarrow \\
\quad (ownName \neq \varnothing) \\
enclaveStatus \in \{\ waitingStartAdminOp, waitingFinishAdminOp\ \} \Leftrightarrow \\
\quad currentAdminOp \neq \varnothing \\
(currentAdminOp \neq \varnothing \\
\quad \land\, theADMINOP\ currentAdminOp \in \{\ shutdownOp, overrideLock\ \}) \\
\qquad \Rightarrow enclaveStatus = waitingStartAdminOp \\
enclaveStatus = gotAdminToken \Rightarrow rolePresent = \varnothing \\
currentScreen.1 = displayStats \\
currentScreen.3 = displayConfigData
\end{array}
$$

RealWorldChanges

$$
\begin{array}{l}
\Delta RealWorld \\
\hline
now' \geq now
\end{array}
$$

PollTime

$$
\begin{array}{l}
\Delta DoorLatchAlarm \\
RealWorld \\
\hline
currentTime' = now
\end{array}
$$

PollDoor

$\Delta DoorLatchAlarm$
$RealWorld$

$currentDoor' = door$
$latchTimeout' = latchTimeout$
$alarmTimeout' = alarmTimeout$

*PollUserToken*1

$\Delta UserToken$
$RealWorld$

$userTokenPresence' = present \Leftrightarrow userToken \neq noT$
$userToken \neq noT$
$currentUserToken' = userToken$

*PollUserToken*2

$\Delta UserToken$
$RealWorld$

$userTokenPresence' = present \Leftrightarrow userToken \neq noT$
$userToken = noT$
$currentUserToken' = currentUserToken$

$PollUserToken == PollUserToken1 \lor PollUserToken2$

*PollAdminToken*1

$\Delta AdminToken$
$RealWorld$

$adminTokenPresence' = present \Leftrightarrow adminToken \neq noT$
$adminToken \neq noT$
$currentAdminToken' = adminToken$

*PollAdminToken*2

$\Delta AdminToken$
$RealWorld$

$adminTokenPresence' = present \Leftrightarrow adminToken \neq noT$
$adminToken = noT$
$currentAdminToken' = currentAdminToken$

$PollAdminToken == PollAdminToken1 \lor PollAdminToken2$

*PollFinger*1

$\Delta Finger$
$RealWorld$

$fingerPresence' = present \Leftrightarrow finger \neq noFP$
$finger \neq noFP$
$currentFinger' = finger$

*PollFinger*2

$\Delta Finger$
$RealWorld$

$fingerPresence' = present \Leftrightarrow finger \neq noFP$
$finger = noFP$
$currentFinger' = currentFinger$

$PollFinger == PollFinger1 \lor PollFinger2$

*PollFloppy*1

$\Delta Floppy$
$RealWorld$

$floppyPresence' = present \Leftrightarrow floppy \neq noFloppy$
$floppy \neq noFloppy$
$currentFloppy' = floppy$
$writtenFloppy' = writtenFloppy$

*PollFloppy*2

$\Delta Floppy$
$RealWorld$

$floppyPresence' = present \Leftrightarrow floppy \neq noFloppy$
$floppy = noFloppy$
$currentFloppy' = currentFloppy$
$writtenFloppy' = writtenFloppy$

$PollFloppy == PollFloppy1 \lor PollFloppy2$

*PollKeyboard*1

$\Delta Keyboard$
$RealWorld$

$keyedDataPresence = present \Leftrightarrow keyboard \neq noKB$
$keyboard \neq noKB$
$currentKeyedData' = keyboard$

*PollKeyboard*2

$\Delta Keyboard$
$RealWorld$

$keyedDataPresence = present \Leftrightarrow keyboard \neq noKB$
$keyboard = noKB$
$currentKeyedData' = currentKeyedData$

$PollKeyboard == PollKeyboard1 \lor PollKeyboard2$

UpdateLatch

$\Xi DoorLatchAlarm$
$RealWorldChanges$

$latch' = currentLatch$

UpdateAlarm

$\Xi DoorLatchAlarm$
$AuditLog$
$RealWorldChanges$

$alarm' = alarming \Leftrightarrow$
$\quad doorAlarm = alarming \lor$
$\quad auditAlarm = alarming$

UpdateDisplay

$\Delta IDStation$
$RealWorldChanges$

$display' = currentDisplay$
$currentDisplay' = currentDisplay$

*UpdateScreen*1

$\Delta IDStation$
$\Xi Admin$
$RealWorldChanges$

$screen'.2 = currentScreen.2$
$theADMINPRIVILEGE\ rolePresent = securityOfficer$
$rolePresent \neq \varnothing$
$screen'.3 = currentScreen.3$
$screen'.1 = currentScreen.1$

*UpdateScreen*2

$\Delta IDStation$
$\Xi Admin$
$RealWorldChanges$

$screen'.2 = currentScreen.2$
$theADMINPRIVILEGE\ rolePresent \neq securityOfficer$
$rolePresent \neq \varnothing$
$screen'.3 = clear$
$screen'.1 = currentScreen.1$

UpdateScreen3

$\Delta IDStation$
$\Xi Admin$
$RealWorldChanges$

$screen'.2 = currentScreen.2$
$theADMINPRIVILEGE\ rolePresent = securityOfficer$
$rolePresent = \varnothing$
$screen'.3 = currentScreen.3$
$screen'.1 = clear$

UpdateScreen4

$\Delta IDStation$
$\Xi Admin$
$RealWorldChanges$

$screen'.2 = currentScreen.2$
$theADMINPRIVILEGE\ rolePresent \neq securityOfficer$
$rolePresent = \varnothing$
$screen'.3 = clear$
$screen'.1 = clear$

$$UpdateScreen == UpdateScreen1 \lor UpdateScreen2 \lor UpdateScreen3 \lor UpdateScreen4$$

UpdateUserToken

$\Xi IDStation$
$RealWorldChanges$
$\Xi TISControlledRealWorld$

$userToken' = currentUserToken$

UpdateFloppy

$\Delta IDStation$
$RealWorldChanges$
$\Xi UserToken$
$\Xi AdminToken$
$\Xi Finger$
$\Xi DoorLatchAlarm$
$\Xi Keyboard$
$\Xi Config$
$\Xi Stats$
$\Xi KeyStore$
$\Xi Admin$
$\Xi AuditLog$
$\Xi Internal$
$\Xi TISControlledRealWorld$

$floppy' = writtenFloppy$
$currentFloppy' = badFloppy$
$floppyPresence' = floppyPresence$
$currentDisplay' = currentDisplay$
$currentScreen' = currentScreen$

AddElementsToLog

$Config$
$SizeAx$
$LogAx$
$\Delta AuditLog$

$$\exists\, newElements : \mathbb{P}_1\, Audit \bullet$$
$$oldestLogTime\ newElements \geq newestLogTime\ auditLog$$
$$\land\ (auditLog' = auditLog \cup newElements$$
$$\land\ (sizeLog\ auditLog' < alarmThresholdSize \land auditAlarm' = auditAlarm$$
$$\lor\ sizeLog\ auditLog' \geq alarmThresholdSize \land auditAlarm' = alarming)$$
$$\lor$$
$$sizeLog\ auditLog + sizeLog\ newElements > minPreservedLogSize$$
$$\land\ (\exists\, oldElements : \mathbb{P}\, Audit \bullet$$
$$oldElements \cup auditLog' = auditLog \cup newElements$$
$$\land\ oldestLogTime\ auditLog' \geq newestLogTime\ oldElements)$$
$$\land\ sizeLog\ auditLog' \geq minPreservedLogSize$$
$$\land\ auditAlarm' = alarming)$$

$$TISEarlyUpdate == UpdateLatch \land UpdateAlarm$$
$$\land\ [\ RealWorldChanges \mid screen' = screen \land display' = display\ ]$$
$$\land\ [\Delta IDStation \mid currentDisplay = currentDisplay']$$
$$\land\ [\Xi UserToken;\ \Xi AdminToken;\ \Xi Finger;\ \Xi Floppy;$$
$$\Xi Keyboard;\ \Xi Config;\ \Xi Stats;$$
$$\Xi KeyStore;\ \Xi Admin;\ \Xi Internal;$$
$$(AddElementsToLog \lor \Xi AuditLog) \mid true]$$

$$
\begin{aligned}
&TISUpdate == \\
&UpdateLatch \land UpdateAlarm \land UpdateDisplay \land UpdateScreen \\
&\quad \land [\Xi UserToken;\ \Xi AdminToken;\ \Xi Finger;\ \Xi Floppy; \\
&\quad \Xi Keyboard;\ \Xi Config;\ \Xi Stats; \\
&\quad \Xi KeyStore;\ \Xi Admin;\ \Xi Internal; \\
&\quad (AddElementsToLog \lor \Xi AuditLog) \mid true]
\end{aligned}
$$

ArchiveLog

$$
\begin{aligned}
&Config \\
&LogAx \\
&\Delta AuditLog \\
&archive : \mathbb{P}\ Audit \\
&\hline
&\exists\ notArchived, \\
&newElements : \mathbb{P}\ Audit;\ sizeElement : Audit \to \mathbb{N}; \\
&sizeLog : \mathbb{P}\ Audit \to \mathbb{N} \bullet \\
&\quad archive \subseteq auditLog \cup newElements \\
&\quad \land\ auditLog' \subseteq archive \cup notArchived \\
&\quad \land\ newestLogTime\ archive \leq oldestLogTime\ notArchived \\
&\quad \land\ AddElementsToLog
\end{aligned}
$$

ClearLog

$$
\begin{aligned}
&Config \\
&SizeAx \\
&LogAx \\
&\Delta AuditLog \\
&archive : \mathbb{P}\ Audit \\
&\hline
&(\exists\ sinceArchive, lostSinceArchive : \mathbb{P}\ Audit \bullet \\
&\quad archive \cup sinceArchive = lostSinceArchive \cup auditLog \\
&\quad \land\ oldestLogTime\ sinceArchive \geq newestLogTime\ archive \\
&\quad \land\ newestLogTime\ lostSinceArchive \leq oldestLogTime\ auditLog \\
&\quad \land\ auditLog' = sinceArchive) \\
&(sizeLog\ auditLog' < alarmThresholdSize \land auditAlarm' = silent \\
&\quad \lor\ sizeLog\ auditLog' \geq alarmThresholdSize \land auditAlarm' = alarming)
\end{aligned}
$$

AuditDoor

$$
\begin{aligned}
&\Delta DoorLatchAlarm \\
&AddElementsToLog \\
&\hline
&currentDoor \neq currentDoor'
\end{aligned}
$$

AuditLatch

$$
\begin{aligned}
&\Delta DoorLatchAlarm \\
&AddElementsToLog \\
&\hline
&currentLatch' \neq currentLatch
\end{aligned}
$$

AuditAlarm

$\Delta DoorLatchAlarm$
$AddElementsToLog$

$doorAlarm \neq doorAlarm'$

AuditLogAlarm

$AddElementsToLog$

$auditAlarm \neq auditAlarm'$

AuditDisplay

$AddElementsToLog$
$\Delta IDStation$

$currentDisplay' \neq currentDisplay$

AuditScreen

$\Delta IDStation$
$AddElementsToLog$

$currentScreen'.2 \neq currentScreen.2$

NoChange

$\Delta IDStation$
$sizeElement : Audit \rightarrow \mathbb{N}$
$maxSupportedLogSize : \mathbb{N}$
$sizeLog : \mathbb{P}\, Audit \rightarrow \mathbb{N}$
$oldestLogTime : \mathbb{P}\, Audit \rightarrow TIME$
$newestLogTime : \mathbb{P}\, Audit \rightarrow TIME$

$currentDoor = currentDoor'$
$currentLatch' = currentLatch$
$doorAlarm = doorAlarm'$
$auditAlarm = auditAlarm'$
$currentDisplay' = currentDisplay$
$currentScreen'.2 = currentScreen.2$
$AddElementsToLog \lor \Xi AuditLog$

$LogChange == AuditAlarm \lor AuditLatch \lor AuditDoor$
$\quad \lor AuditLogAlarm \lor AuditScreen \lor AuditDisplay$
$\quad \lor NoChange$

TISPoll

$\Delta IDStation$
$\Xi RealWorld$
$PollTime$
$PollDoor$
$PollUserToken$
$PollAdminToken$
$PollFinger$
$PollFloppy$
$PollKeyboard$
$LogChange$
$\Xi Config$
$\Xi KeyStore$
$\Xi Admin$
$\Xi Stats$
$\Xi Internal$

$currentScreen' = currentScreen$
$currentDisplay = doorUnlocked \land currentLatch' = locked$
$\quad \land (status \neq waitingRemoveTokenFail \land currentDisplay' = welcome$
$\quad\quad \lor status = waitingRemoveTokenFail \land currentDisplay' = removeToken)$
$\lor \neg (currentDisplay = doorUnlocked \land currentLatch' = locked)$
$\quad \land currentDisplay' = currentDisplay$

AddSuccessfulEntryToStats

$\Delta Stats$

$failEntry' = failEntry$
$successEntry' = successEntry + 1$
$failBio' = failBio$
$successBio' = successBio$

AddFailedEntryToStats

$\Delta Stats$

$failEntry' = failEntry + 1$
$successEntry' = successEntry$
$failBio' = failBio$
$successBio' = successBio$

AddSuccessfulBioCheckToStats

$\Delta Stats$

$failEntry' = failEntry$
$successEntry' = successEntry$
$failBio' = failBio$
$successBio' = successBio + 1$

AddFailedBioCheckToStats

$\Delta Stats$

$failEntry' = failEntry$
$successEntry' = successEntry$
$failBio' = failBio + 1$
$successBio' = successBio$

UnlockDoor

$\Delta DoorLatchAlarm$
$Config$

$latchTimeout' = currentTime + latchUnlockDuration$
$alarmTimeout' = currentTime + latchUnlockDuration +$
$alarmSilentDuration$
$currentTime' = currentTime$
$currentDoor' = currentDoor$

LockDoor

$\Delta DoorLatchAlarm$

$currentLatch' = locked$
$latchTimeout' = currentTime$
$alarmTimeout' = currentTime$
$currentTime' = currentTime$
$currentDoor' = currentDoor$

CertIssuerKnown

$KeyStore$
$Certificate$

CertIssuerKnownInv

$CertIssuerKnown$
$KeyStoreInv$
$CertificateInv$

$iden.issuer \in \mathrm{dom}\, issuerKey$

CertOK

$CertIssuerKnown$

CertOKInv

$CertOK$
$CertIssuerKnownInv$

$issuerKey(iden.issuer) \in isValidatedBy$

CertIssuerIsThisTIS

$KeyStore$
$Certificate$

CertIssuerIsThisTISInv

$CertIssuerIsThisTIS$
$KeyStoreInv$
$CertificateInv$

$ownName \neq \varnothing$
$iden.issuer = theISSUER\ ownName$

$AuthCertOK == CertIssuerIsThisTIS \land CertOK$

NewAuthCert

$ValidToken$
$KeyStore$
$Config$
$MinClearance$
$newAuthCert : AuthCert$
$currentTime : TIME$

NewAuthCertInv

$NewAuthCert$

$ownName \neq \varnothing$
$newAuthCert.iden.issuer = theISSUER\ ownName$
$newAuthCert.validityPeriod = authPeriod\ privCert.role\ currentTime$
$newAuthCert.baseCertId = idCert.iden$
$newAuthCert.tokenID = tokenID$
$newAuthCert.role = privCert.role$
$newAuthCert.clearance = minClearance(enclaveClearance, privCert.clearance)$
$newAuthCert.isValidatedBy = \{\ issuerKey(theISSUER\ ownName)\ \}$

AddAuthCertToUserToken

$\Delta UserToken$
$KeyStore$
$Config$
$currentTime : TIME$
$theAuthCert : \mathbb{P}\, AuthCert \rightarrow AuthCert$
$goodT : Token \nrightarrow TOKENTRY$

$userTokenPresence = present$
$currentUserToken \in \operatorname{ran} goodT$
$\exists\, ValidToken;\ ValidToken\,' \bullet$
$\quad \theta\, ValidToken = ((goodT^{\sim})\, currentUserToken)$
$\quad \land\ \theta\, ValidToken\,' = ((goodT^{\sim})\, currentUserToken')$
$\quad \land\ (\exists\, newAuthCert : AuthCert;$
$\qquad minClearance : Clearance \times Clearance \rightarrow Clearance \bullet$
$\qquad\quad theAuthCert\ authCert' = newAuthCert \land NewAuthCert)$
$\quad \land\ tokenID' = tokenID$
$\quad \land\ idCert' = idCert$
$\quad \land\ privCert' = privCert$
$\quad \land\ iandACert' = iandACert$
$userTokenPresence' = userTokenPresence$

UpdateKeyStore

$\Delta KeyStore$
$ValidEnrol$

$theISSUER\ ownName' = idStationCert.subject$
$issuerKey' = issuerKey \oplus \{c : issuerCerts \bullet c.subject \mapsto c.subjectPubK\}$
$\quad \oplus\ \{theISSUER\ ownName \mapsto idStationCert.subjectPubK\}$

UpdateKeyStoreFromFloppy

$\Delta KeyStore$
$Floppy$
$enrolmentFile : ValidEnrol \nrightarrow FLOPPY$

$currentFloppy \in \operatorname{ran} enrolmentFile$
$(\exists\, ValidEnrol \bullet \theta\, ValidEnrol = (enrolmentFile^{\sim})\, currentFloppy$
$\quad \land\ UpdateKeyStore)$

AdminLogon

$\Delta Admin$
$AdminToken$
$theAuthCert : \mathbb{P}\, AuthCert \rightarrow AuthCert$
$goodT : Token \nrightarrow TOKENTRY$

$rolePresent = \varnothing$
$\exists\, ValidToken \bullet$
$\quad (goodT(\theta\, ValidToken) = currentAdminToken$
$\quad \land\ theADMINPRIVILEGE\ rolePresent' = (theAuthCert\ authCert).role)$
$currentAdminOp' = \varnothing$

AdminLogout

$\Delta Admin$

$rolePresent \neq \varnothing$
$rolePresent' = \varnothing$
$currentAdminOp' = \varnothing$

AdminStartOp

$\Delta Admin$
$Keyboard$
$keyedOps : ADMINOP \pfun KEYBOARD$

$keyedOps = \{archiveLog \mapsto keyedOpsArchiveLog,$
$\quad updateConfigData \mapsto keyedOpsUpdateConfigData,$
$\quad overrideLock \mapsto keyedOpsOverrideLock,$
$\quad shutdownOp \mapsto keyedOpsShutdownOp\}$
$rolePresent \neq \varnothing$
$currentAdminOp = \varnothing$
$currentKeyedData \in keyedOps(\!|\ availableOps\ |\!)$
$rolePresent' = rolePresent$
$theADMINOP\ \ currentAdminOp' = (keyedOps^{\sim})currentKeyedData$

AdminFinishOp

$\Delta Admin$

$rolePresent \neq \varnothing$
$currentAdminOp \neq \varnothing$
$rolePresent' = rolePresent$
$currentAdminOp' = \varnothing$

ResetScreenMessage

$\Delta Internal$
$\Delta Admin$
$currentScreen, currentScreen' : Screen$

$status' \notin \{quiescent, waitingRemoveTokenFail\}$
$\quad \land\ currentScreen'.2 = busy$
$\lor$
$status' \in \{quiescent, waitingRemoveTokenFail\}$
$\quad \land (enclaveStatus' = enclaveQuiescent \land rolePresent' = \varnothing$
$\qquad \land\ currentScreen'.2 = welcomeAdmin$
$\quad \lor\ enclaveStatus' = enclaveQuiescent \land rolePresent' \neq \varnothing$
$\qquad \land\ currentScreen'.2 = requestAdminOp$
$\quad \lor\ enclaveStatus' = waitingRemoveAdminTokenFail$
$\qquad \land\ currentScreen'.2 = removeAdminToken$
$\quad \lor\ enclaveStatus' \notin \{enclaveQuiescent, waitingRemoveAdminTokenFail\}$
$\qquad \land\ currentScreen'.2 = currentScreen.2)$

UserEntryContext

$\Delta IDStation$
$RealWorldChanges$
$\Xi Config$
$\Xi AdminToken$
$\Xi KeyStore$
$\Xi Admin$
$\Xi Keyboard$
$\Xi Floppy$
$\Xi Finger$
$\Xi TISControlledRealWorld$
$ResetScreenMessage$

$enclaveStatus' = enclaveStatus$
$status \neq waitingEntry \Rightarrow tokenRemovalTimeout' = tokenRemovalTimeout$

UserTokenTorn

$UserEntryContext$
$\Xi UserToken$
$\Xi DoorLatchAlarm$
$AddFailedEntryToStats$
$AddElementsToLog$

$status \in \{gotUserToken, waitingUpdateToken,$
$\quad waitingFinger, gotFinger, waitingEntry\}$
$userTokenPresence = absent$
$currentDisplay' = welcome$
$status' = quiescent$

ReadUserToken

$UserEntryContext$
$\Xi UserToken$
$\Xi DoorLatchAlarm$
$\Xi Stats$
$AddElementsToLog$

$enclaveStatus \in \{enclaveQuiescent, waitingRemoveAdminTokenFail\}$
$status = quiescent$
$userTokenPresence = present$
$currentDisplay' = wait$
$status' = gotUserToken$

$TISReadUserToken == ReadUserToken$

UserTokenWithOKAuthCert

$KeyStore$
$UserToken$
$currentTime : TIME$

UserTokenWithOKAuthCertInv

$$
\begin{array}{l}
UserTokenWithOKAuthCert \\
KeyStoreInv \\
goodT : Token \pfun TOKENTRY \\
\hline
currentUserToken \in \operatorname{ran} goodT \\
\exists\, TokenWithValidAuth \bullet \\
\quad (goodThe = currentUserToken \\
\quad \land\ currentTime \in (theAuthCert\ authCert).validityPeriod \\
\quad \land\ (\exists\, IDCert \bullet \theta IDCert = idCert \land CertOK) \\
\quad \land\ (\exists\, AuthCert \bullet \theta AuthCert = theAuthCert\ authCert \land AuthCertOK))
\end{array}
$$

UserTokenOK

$$
\begin{array}{l}
KeyStore \\
UserToken \\
currentTime : TIME
\end{array}
$$

UserTokenOKInv

$$
\begin{array}{l}
UserTokenOK \\
KeyStoreInv \\
goodT : Token \pfun TOKENTRY \\
\hline
currentUserToken \in \operatorname{ran} goodT \\
\exists\, CurrentToken \bullet \\
\quad (goodT(\theta ValidToken) = currentUserToken \\
\quad \land\ now = currentTime \\
\quad \land\ (\exists\, IDCert \bullet \theta IDCert = idCert \land CertOK) \\
\quad \land\ (\exists\, PrivCert \bullet \theta PrivCert = privCert \land CertOK) \\
\quad \land\ (\exists\, IandACert \bullet \theta IandACert = iandACert \land CertOK))
\end{array}
$$

BioCheckNotRequired

$$
\begin{array}{l}
UserEntryContext \\
\Xi UserToken \\
\Xi DoorLatchAlarm \\
\Xi Stats \\
AddElementsToLog \\
\hline
status = gotUserToken \\
userTokenPresence = present \\
UserTokenWithOKAuthCert \\
status' = waitingEntry \\
currentDisplay' = wait
\end{array}
$$

BioCheckRequired

$UserEntryContext$
$\Xi UserToken$
$\Xi DoorLatchAlarm$
$\Xi Stats$
$AddElementsToLog$

$status = gotUserToken$
$userTokenPresence = present$
$\neg\ UserTokenWithOKAuthCert \land UserTokenOK$
$currentDisplay' = insertFinger$
$status' = waitingFinger$

$ValidateUserTokenOK == BioCheckRequired \lor BioCheckNotRequired$

ValidateUserTokenFail

$UserEntryContext$
$\Xi UserToken$
$\Xi DoorLatchAlarm$
$\Xi Stats$
$AddElementsToLog$

$status = gotUserToken$
$userTokenPresence = present$
$\neg\ UserTokenWithOKAuthCert \land \neg\ UserTokenOK$
$currentDisplay' = removeToken$
$status' = waitingRemoveTokenFail$

$TISValidateUserToken ==$
$\quad ValidateUserTokenOK \lor ValidateUserTokenFail$
$\quad \lor [\ UserTokenTorn \mid status = gotUserToken]$

ReadFingerOK

$UserEntryContext$
$\Xi DoorLatchAlarm$
$\Xi UserToken$
$\Xi Stats$
$AddElementsToLog$

$status = waitingFinger$
$fingerPresence = present$
$userTokenPresence = present$
$currentDisplay' = wait$
$status' = gotFinger$

NoFinger

$\Xi IDStation$
$RealWorldChanges$
$\Xi TISControlledRealWorld$

$status = waitingFinger$
$fingerPresence = absent$
$userTokenPresence = present$

FingerTimeout

$UserEntryContext$
$\Xi UserToken$
$\Xi DoorLatchAlarm$
$\Xi Stats$
$AddElementsToLog$

$status = waitingFinger$
$userTokenPresence = present$
$currentDisplay' = removeToken$
$status' = waitingRemoveTokenFail$

$$TISReadFinger == ReadFingerOK \lor FingerTimeout \lor NoFinger \lor [\ UserTokenTorn \mid status = waitingFinger\ ]$$

FingerOK

$Finger$
$UserToken$
$goodFP : FINGERPRINT \pfun FINGERPRINTTRY$

$goodFP = \{FP1 \mapsto goodFP1, FP2 \mapsto goodFP2, FP3 \mapsto goodFP3\}$
$currentFinger \in \operatorname{ran} goodFP$

ValidateFingerOK

$UserEntryContext$
$\Xi DoorLatchAlarm$
$\Xi UserToken$
$AddSuccessfulBioCheckToStats$
$AddElementsToLog$
$goodFP : FINGERPRINT \pfun FINGERPRINTTRY$

$status = gotFinger$
$userTokenPresence = present$
$FingerOK$
$status' = waitingUpdateToken$
$currentDisplay' = wait$

ValidateFingerFail

$UserEntryContext$
$\Xi UserToken$
$\Xi DoorLatchAlarm$
$AddFailedBioCheckToStats$
$AddElementsToLog$

$status = gotFinger$
$userTokenPresence = present$
$currentDisplay' = removeToken$
$status' = waitingRemoveTokenFail$

$$TISValidateFinger == ValidateFingerOK \lor ValidateFingerFail \lor [\ UserTokenTorn \mid status = gotFinger\ ]$$

WriteUserTokenOK

$UserEntryContext$
$\Xi DoorLatchAlarm$
$\Xi Stats$
$AddElementsToLog$
$AddAuthCertToUserToken$

$status = waitingUpdateToken$
$userTokenPresence = present$
$status' = waitingEntry$
$currentDisplay' = wait$

WriteUserTokenFail

$UserEntryContext$
$\Xi DoorLatchAlarm$
$\Xi Stats$
$AddElementsToLog$
$AddAuthCertToUserToken$

$status = waitingUpdateToken$
$userTokenPresence = present$
$status' = waitingEntry$
$currentDisplay' = tokenUpdateFailed$

$$WriteUserToken == WriteUserTokenOK \lor WriteUserTokenFail$$

UserAllowedEntry

$UserToken$
$Config$
$currentTime : TIME$

UserAllowedEntryInv

$$
\begin{array}{l}
UserAllowedEntry \\
ConfigInv \\
goodT : Token \pfun TOKENTRY \\
\hline
(\exists\, ValidToken \bullet \\
\quad goodT(\theta\, ValidToken) = currentUserToken \\
\quad \land\ currentTime \in entryPeriod\ privCert.role\ privCert.clearance.class) \\
\lor\ (\exists\, TokenWithValidAuth \bullet \\
\quad goodThe = currentUserToken \\
\quad \land\ currentTime \in \\
entryPeriod\ (theAuthCert\ authCert).role\ (theAuthCert).clearance.class)
\end{array}
$$

EntryOK

$$
\begin{array}{l}
UserEntryContext \\
\Xi DoorLatchAlarm \\
\Xi UserToken \\
\Xi Stats \\
AddElementsToLog \\
\hline
status = waitingEntry \\
userTokenPresence = present \\
UserAllowedEntry \\
currentDisplay' = openDoor \\
status' = waitingRemoveTokenSuccess \\
tokenRemovalTimeout' = currentTime + tokenRemovalDuration
\end{array}
$$

EntryNotAllowed

$$
\begin{array}{l}
UserEntryContext \\
\Xi DoorLatchAlarm \\
\Xi UserToken \\
\Xi Stats \\
AddElementsToLog \\
\hline
status = waitingEntry \\
userTokenPresence = present \\
\neg\ UserAllowedEntry \\
currentDisplay' = removeToken \\
status' = waitingRemoveTokenFail \\
tokenRemovalTimeout' = tokenRemovalTimeout
\end{array}
$$

$$
\begin{array}{l}
TISValidateEntry == EntryOK \\
\qquad\qquad \lor\ EntryNotAllowed \\
\qquad\qquad \lor\ [\ UserTokenTorn \mid status = waitingEntry\ ]
\end{array}
$$

UnlockDoorOK

$UserEntryContext$
$\Xi UserToken$
$UnlockDoor$
$AddSuccessfulEntryToStats$
$AddElementsToLog$

$status = waitingRemoveTokenSuccess$
$userTokenPresence = absent$
$currentDisplay' = doorUnlocked$
$status' = quiescent$

WaitingTokenRemoval

$\Xi IDStation$
$RealWorldChanges$
$\Xi TISControlledRealWorld$

$status \in$
$\quad \{waitingRemoveTokenSuccess, waitingRemoveTokenFail\}$
$status = waitingRemoveTokenSuccess \Rightarrow$
$\quad currentTime \leq tokenRemovalTimeout$
$userTokenPresence = present$

TokenRemovalTimeout

$UserEntryContext$
$\Xi DoorLatchAlarm$
$\Xi UserToken$
$\Xi Stats$
$AddElementsToLog$

$status = waitingRemoveTokenSuccess$
$currentTime > tokenRemovalTimeout$
$userTokenPresence = present$
$status' = waitingRemoveTokenFail$
$currentDisplay' = removeToken$

$TISUnlockDoor == UnlockDoorOK$
$\quad \lor [WaitingTokenRemoval \mid status = waitingRemoveTokenSuccess]$
$\quad \lor TokenRemovalTimeout$

FailedAccessTokenRemoved

$UserEntryContext$
$\Xi UserToken$
$\Xi DoorLatchAlarm$
$AddFailedEntryToStats$
$AddElementsToLog$

$status = waitingRemoveTokenFail$
$userTokenPresence = absent$
$currentDisplay' = welcome$
$status' = quiescent$

$$TISCompleteFailedAccess == FailedAccessTokenRemoved \lor [WaitingTokenRemoval \mid status = waitingRemoveTokenFail]$$

EnclaveContext

$\Delta IDStation$
$RealWorldChanges$
$\Xi TISControlledRealWorld$
$\Xi UserToken$
$\Xi AdminToken$
$\Xi Finger$
$\Xi Stats$

$tokenRemovalTimeout' = tokenRemovalTimeout$

EnrolContext

$EnclaveContext$
$\Xi Keyboard$
$\Xi Admin$
$\Xi DoorLatchAlarm$
$\Xi Config$
$\Xi Floppy$

RequestEnrolment

$EnrolContext$
$\Xi KeyStore$
$\Xi AuditLog$
$\Xi Internal$

$enclaveStatus = notEnrolled$
$floppyPresence = absent$
$currentScreen'.2 = insertEnrolmentData$
$currentDisplay' = blank$

ReadEnrolmentFloppy

$EnrolContext$
$\Xi KeyStore$

$enclaveStatus = notEnrolled$
$floppyPresence = present$
$currentScreen'.2 = validatingEnrolmentData$
$enclaveStatus' = waitingEnrol$
$status' = status$
$currentDisplay' = blank$

$$ReadEnrolmentData == ReadEnrolmentFloppy \lor RequestEnrolment$$

EnrolmentDataOK

$Floppy$
$KeyStore$

EnrolmentDataOKInv

$EnrolmentDataOK$
$KeyStoreInv$
$enrolmentFile : ValidEnrol \pfun FLOPPY$

$currentFloppy \in \operatorname{ran} enrolmentFile$
$(\exists\, ValidEnrol \bullet \theta\, ValidEnrol = (enrolmentFile^{\sim})\, currentFloppy)$

ValidateEnrolmentDataOK

$EnrolContext$
$UpdateKeyStoreFromFloppy$
$AddElementsToLog$

$enclaveStatus = waitingEnrol$
$EnrolmentDataOK$
$currentScreen'.2 = welcomeAdmin$
$enclaveStatus' = enclaveQuiescent$
$status' = quiescent$
$currentDisplay' = welcome$

ValidateEnrolmentDataFail

$EnrolContext$
$\Xi KeyStore$
$AddElementsToLog$

$enclaveStatus = waitingEnrol$
$\neg\, EnrolmentDataOK$
$currentScreen'.2 = enrolmentFailed$
$enclaveStatus' = waitingEndEnrol$
$status' = status$
$currentDisplay' = blank$

$ValidateEnrolmentData == ValidateEnrolmentDataOK$
$\quad \lor\ ValidateEnrolmentDataFail$

FailedEnrolFloppyRemoved

$EnrolContext$
$\Xi KeyStore$

$enclaveStatus = waitingEndEnrol$
$floppyPresence = absent$
$currentScreen'.2 = insertEnrolmentData$
$enclaveStatus' = notEnrolled$
$status' = status$
$currentDisplay' = blank$

WaitingFloppyRemoval

$EnclaveContext$
$\Xi IDStation$

$enclaveStatus = waitingEndEnrol$
$floppyPresence = present$

$CompleteFailedEnrolment == FailedEnrolFloppyRemoved$
$\quad \lor\ WaitingFloppyRemoval$

$TISEnrolOp == ReadEnrolmentData \lor ValidateEnrolmentData$
$\quad \lor\ CompleteFailedEnrolment$

AdminTokenTear

$EnclaveContext$
$\Xi Config$
$\Xi Floppy$
$\Xi Keyboard$
$\Xi DoorLatchAlarm$
$\Xi KeyStore$
$ResetScreenMessage$

$adminTokenPresence = absent$
$status' = status$
$currentDisplay' = currentDisplay$
$enclaveStatus' = enclaveQuiescent$

BadAdminTokenTear

$AdminTokenTear$
$AddElementsToLog$

$enclaveStatus \in \{gotAdminToken, waitingStartAdminOp, waitingFinishAdminOp\}$

BadAdminLogout

$BadAdminTokenTear$
$AdminLogout$

$enclaveStatus \in \{waitingStartAdminOp, waitingFinishAdminOp\}$

LoginAborted

$BadAdminTokenTear$
$\Xi Admin$

$enclaveStatus = gotAdminToken$

LoginContext

$$
\begin{array}{l}
EnclaveContext \\
\Xi Keyboard \\
\Xi KeyStore \\
\Xi DoorLatchAlarm \\
\Xi Config \\
\Xi Floppy \\
\hline
status' = status \\
currentDisplay' = currentDisplay
\end{array}
$$

ReadAdminToken

$$
\begin{array}{l}
LoginContext \\
\Xi Admin \\
AddElementsToLog \\
\hline
status \in \{\ quiescent, waitingRemoveTokenFail\ \} \\
enclaveStatus = enclaveQuiescent \\
rolePresent = \varnothing \\
adminTokenPresence = present \\
enclaveStatus' = gotAdminToken \\
currentScreen' = currentScreen
\end{array}
$$

$$TISReadAdminToken == ReadAdminToken$$

AdminTokenOK

$$
\begin{array}{l}
AdminToken \\
KeyStore \\
currentTime : TIME
\end{array}
$$

AdminTokenOKInv

$$
\begin{array}{l}
AdminTokenOK \\
KeyStoreInv \\
goodT : Token \pfun TOKENTRY \\
\hline
currentAdminToken \in \operatorname{ran} goodT \\
\exists\, TokenWithValidAuth \bullet \\
\quad (goodThe = currentAdminToken \\
\quad \land (\exists\, IDCert \bullet \theta IDCert = idCert \land CertOK) \\
\quad \land (\exists\, AuthCert \bullet \theta AuthCert = theAuthCert\ authCert \land AuthCertOK) \\
\quad \land (theAuthCert\ authCert).role \in ADMINPRIVILEGE \\
\quad \land currentTime \in (theAuthCert\ authCert).validityPeriod)
\end{array}
$$

ValidateAdminTokenOK

$$
\begin{array}{l}
LoginContext \\
AdminLogon \\
AddElementsToLog \\
\hline
enclaveStatus = gotAdminToken \\
adminTokenPresence = present \\
AdminTokenOK \\
currentScreen'.2 = requestAdminOp \\
enclaveStatus' = enclaveQuiescent
\end{array}
$$

ValidateAdminTokenFail

$$
\begin{array}{l}
LoginContext \\
\Xi Admin \\
AddElementsToLog \\
\hline
enclaveStatus = gotAdminToken \\
adminTokenPresence = present \\
\neg\ AdminTokenOK \\
currentScreen'.2 = removeAdminToken \\
enclaveStatus' = waitingRemoveAdminTokenFail
\end{array}
$$

$$
\begin{array}{l}
TISValidateAdminToken == ValidateAdminTokenOK \lor \\
\quad ValidateAdminTokenFail \\
\quad \lor\ LoginAborted
\end{array}
$$

FailedAdminTokenRemoved

$$
\begin{array}{l}
LoginContext \\
\Xi Admin \\
AddElementsToLog \\
\hline
enclaveStatus = waitingRemoveAdminTokenFail \\
adminTokenPresence = absent \\
currentScreen'.2 = welcomeAdmin \\
enclaveStatus' = enclaveQuiescent \\
currentDisplay' = currentDisplay
\end{array}
$$

WaitingAdminTokenRemoval

$$
\begin{array}{l}
EnclaveContext \\
\Xi IDStation \\
\hline
enclaveStatus = waitingRemoveAdminTokenFail \\
adminTokenPresence = present
\end{array}
$$

$$
\begin{array}{l}
TISCompleteFailedAdminLogon == \\
\quad FailedAdminTokenRemoved \\
\quad \lor\ WaitingAdminTokenRemoval
\end{array}
$$

$$TISAdminLogon == TISReadAdminToken \lor TISValidateAdminToken \lor TISCompleteFailedAdminLogon$$

TokenRemovedAdminLogout

$$AdminTokenTear$$
$$AdminLogout$$
$$AddElementsToLog$$

$$enclaveStatus = enclaveQuiescent$$
$$rolePresent \neq \varnothing$$

AdminTokenTimeout

$$LoginContext$$
$$AdminLogout$$
$$AddElementsToLog$$
$$ResetScreenMessage$$

$$enclaveStatus = enclaveQuiescent$$
$$adminTokenPresence = present$$
$$rolePresent \neq \varnothing$$
$$\neg\ AdminTokenOK$$
$$enclaveStatus' = waitingRemoveAdminTokenFail$$

$$TISCompleteTimeoutAdminLogout == TISCompleteFailedAdminLogon$$

$$TISAdminLogout == TokenRemovedAdminLogout \lor AdminTokenTimeout \lor TISCompleteTimeoutAdminLogout$$

AdminOpContext

$$EnclaveContext$$
$$\Xi Keyboard$$
$$\Xi KeyStore$$

AdminOpStartedContext

$$AdminOpContext$$

$$enclaveStatus = waitingStartAdminOp$$
$$adminTokenPresence = present$$
$$status' = status$$

AdminOpFinishContext

$$
\begin{array}{l}
AdminOpContext \\
AdminFinishOp \\
\hline
enclaveStatus = waitingFinishAdminOp \\
adminTokenPresence = present \\
status' = status \\
currentDisplay' = currentDisplay \\
enclaveStatus' = enclaveQuiescent
\end{array}
$$

StartOpContext

$$
\begin{array}{l}
EnclaveContext \\
\Xi DoorLatchAlarm \\
\Xi Keyboard \\
\Xi Config \\
\Xi Floppy \\
\Xi KeyStore \\
\hline
enclaveStatus = enclaveQuiescent \\
adminTokenPresence = present \\
rolePresent \neq \varnothing \\
status \in \{\ quiescent, waitingRemoveTokenFail\ \} \\
status' = status \\
currentDisplay' = currentDisplay
\end{array}
$$

ValidateOpRequestOK

$$
\begin{array}{l}
StartOpContext \\
AdminStartOp \\
AddElementsToLog \\
keyedOps : ADMINOP \pfun KEYBOARD \\
\hline
keyedOps = \{archiveLog \mapsto keyedOpsArchiveLog, \\
\quad updateConfigData \mapsto keyedOpsUpdateConfigData, \\
\quad overrideLock \mapsto keyedOpsOverrideLock, \\
\quad shutdownOp \mapsto keyedOpsShutdownOp\} \\
keyedDataPresence = present \\
currentKeyedData \in keyedOps(\!|\ availableOps\ |\!) \\
currentScreen'.2 = doingOp \\
enclaveStatus' = waitingStartAdminOp
\end{array}
$$

ValidateOpRequestFail

$$
\begin{array}{l}
StartOpContext \\
\Xi Admin \\
AddElementsToLog \\
keyedOps : ADMINOP \pfun KEYBOARD \\
\hline
keyedOps = \{archiveLog \mapsto keyedOpsArchiveLog, \\
\quad updateConfigData \mapsto keyedOpsUpdateConfigData, \\
\quad overrideLock \mapsto keyedOpsOverrideLock, \\
\quad shutdownOp \mapsto keyedOpsShutdownOp\} \\
keyedDataPresence = present \\
currentKeyedData \notin keyedOps(\!| availableOps |\!) \\
currentScreen'.2 = invalidRequest \\
enclaveStatus' = enclaveStatus
\end{array}
$$

NoOpRequest

$$
\begin{array}{l}
StartOpContext \\
\Xi IDStation \\
\hline
keyedDataPresence = absent
\end{array}
$$

$$
\begin{array}{l}
ValidateOpRequest == \\
\quad ValidateOpRequestOK \\
\quad \lor\ ValidateOpRequestFail \\
\quad \lor\ NoOpRequest
\end{array}
$$

$$
TISStartAdminOp == ValidateOpRequest
$$

StartArchiveLogOK

$$
\begin{array}{l}
AdminOpStartedContext \\
\Xi Config \\
\Xi Admin \\
\Xi DoorLatchAlarm \\
\hline
theADMINOP\ currentAdminOp = archiveLog \\
floppyPresence = present \\
floppyPresence' = floppyPresence \\
currentFloppy' = currentFloppy \\
currentScreen'.2 = doingOp \\
currentDisplay' = currentDisplay \\
enclaveStatus' = waitingFinishAdminOp \\
(\exists\, archive : \mathbb{P}\, Audit;\ oldestLogTime : \mathbb{P}\, Audit \to TIME; \\
\quad newestLogTime : \mathbb{P}\, Audit \to TIME \bullet \\
\qquad ArchiveLog \land writtenFloppy' = auditFile)
\end{array}
$$

StartArchiveLogWaitingFloppy

$AdminOpStartedContext$
$\Xi Config$
$\Xi Admin$
$\Xi DoorLatchAlarm$
$\Xi Floppy$

$theADMINOP\ currentAdminOp = archiveLog$
$floppyPresence = absent$
$currentScreen'.2 = insertBlankFloppy$
$currentDisplay' = currentDisplay$
$enclaveStatus' = enclaveStatus$

FinishArchiveLogNoFloppy

$AdminOpFinishContext$
$\Xi Config$
$\Xi Floppy$
$\Xi DoorLatchAlarm$
$AddElementsToLog$

$theADMINOP\ currentAdminOp = archiveLog$
$floppyPresence = absent$
$currentScreen'.2 = archiveFailed$

FinishArchiveLogBadMatch

$AdminOpFinishContext$
$\Xi Config$
$\Xi Floppy$
$\Xi DoorLatchAlarm$
$AddElementsToLog$

$theADMINOP\ currentAdminOp = archiveLog$
$floppyPresence = present$
$writtenFloppy \neq currentFloppy$
$currentScreen'.2 = archiveFailed$

StartUpdateConfigOK

$AdminOpStartedContext$
$\Xi Floppy$
$\Xi Config$
$\Xi Admin$
$\Xi DoorLatchAlarm$

$theADMINOP\ currentAdminOp = updateConfigData$
$floppyPresence = present$
$currentScreen'.2 = doingOp$
$currentDisplay' = currentDisplay$
$enclaveStatus' = waitingFinishAdminOp$

StartUpdateConfigWaitingFloppy

$AdminOpStartedContext$
$\Xi Config$
$\Xi Admin$
$\Xi Floppy$
$\Xi DoorLatchAlarm$

$theADMINOP\ currentAdminOp = updateConfigData$
$floppyPresence = absent$
$currentScreen'.2 = insertConfigData$
$currentDisplay' = currentDisplay$
$enclaveStatus' = enclaveStatus$

$$StartUpdateConfigData ==$$
$$StartUpdateConfigOK$$
$$\lor\ StartUpdateConfigWaitingFloppy$$
$$\lor\ [\ BadAdminLogout \mid enclaveStatus = waitingStartAdminOp$$
$$\land\ theADMINOP\ currentAdminOp = updateConfigData\ ]$$

FinishUpdateConfigDataOK

$AdminOpFinishContext$
$\Xi Floppy$
$\Xi DoorLatchAlarm$
$AddElementsToLog$
$configFile : Config \pfun FLOPPY$

$theADMINOP\ currentAdminOp = updateConfigData$
$currentFloppy \in \operatorname{ran} configFile$
$\theta Config\ ' = (configFile^{\sim})currentFloppy$
$currentScreen'.2 = requestAdminOp$

FinishUpdateConfigDataFail

$AdminOpFinishContext$
$\Xi Config$
$\Xi Floppy$
$\Xi DoorLatchAlarm$
$AddElementsToLog$
$configFile : Config \pfun FLOPPY$

$theADMINOP\ currentAdminOp = updateConfigData$
$currentFloppy \notin \operatorname{ran} configFile$
$currentScreen'.2 = invalidData$

$$FinishUpdateConfigData$$
$$== FinishUpdateConfigDataOK$$
$$\lor\ FinishUpdateConfigDataFail$$
$$\lor\ [\ BadAdminLogout \mid enclaveStatus = waitingFinishAdminOp$$
$$\land\ theADMINOP\ currentAdminOp = updateConfigData\ ]$$

$$TISUpdateConfigDataOp == StartUpdateConfigData \lor FinishUpdateConfigData$$

ShutdownOK

$$AdminOpContext$$
$$\Xi Config$$
$$\Xi Floppy$$
$$AddElementsToLog$$
$$LockDoor$$
$$AdminLogout$$

$$enclaveStatus = waitingStartAdminOp$$
$$theADMINOP\ currentAdminOp = shutdownOp$$
$$currentDoor = closed$$
$$currentScreen'.2 = clear$$
$$enclaveStatus' = shutdown$$
$$currentDisplay' = blank$$

ShutdownWaitingDoor

$$AdminOpContext$$
$$\Xi Config$$
$$\Xi Floppy$$
$$\Xi DoorLatchAlarm$$
$$\Xi Admin$$

$$enclaveStatus = waitingStartAdminOp$$
$$theADMINOP\ currentAdminOp = shutdownOp$$
$$currentDoor = open$$
$$currentScreen'.2 = closeDoor$$
$$enclaveStatus' = enclaveStatus$$
$$currentDisplay' = currentDisplay$$

$$TISShutdownOp == ShutdownOK \lor ShutdownWaitingDoor$$

OverrideDoorLockOK

$$AdminOpStartedContext$$
$$\Xi Floppy$$
$$\Xi Config$$
$$AddElementsToLog$$
$$AdminFinishOp$$
$$UnlockDoor$$

$$theADMINOP\ currentAdminOp = overrideLock$$
$$currentScreen'.2 = requestAdminOp$$
$$currentDisplay' = doorUnlocked$$
$$enclaveStatus' = enclaveQuiescent$$

$$TISOverrideDoorLockOp == OverrideDoorLockOK \lor [\ BadAdminLogout \mid enclaveStatus = waitingStartAdminOp \land theADMINOP\ currentAdminOp = overrideLock\ ]$$

InitDoorLatchAlarm

DoorLatchAlarm

$currentTime = zeroTime$
$currentDoor = closed$
$latchTimeout = zeroTime$
$alarmTimeout = zeroTime$

InitKeyStore

KeyStore

$issuerKey = \varnothing$
$ownName = \varnothing$

InitConfig

Config

$alarmSilentDuration = 10$
$latchUnlockDuration = 150$
$tokenRemovalDuration = 100$
$enclaveClearance.class = unmarked$
$authPeriod = PRIVILEGE \times \{\{t : TIME \bullet t \mapsto t\,..\,t + 72000\}\}$
$entryPeriod = PRIVILEGE \times \{CLASS \times \{TIME\}\}$

InitAdmin

Admin

$rolePresent = \varnothing$
$currentAdminOp = \varnothing$

InitStats

Stats

$successEntry = 0$
$failEntry = 0$
$successBio = 0$
$failBio = 0$

InitAuditLog

AuditLog

$auditLog = \varnothing$
$auditAlarm = silent$

InitIDStation

$IDStation$
$InitDoorLatchAlarm$
$InitConfig$
$InitKeyStore$
$InitStats$
$InitAuditLog$
$InitAdmin$

$currentScreen.2 = clear$
$currentDisplay = blank$
$enclaveStatus = notEnrolled$
$status = quiescent$

StartContext

$\Delta IDStation$
$RealWorldChanges$
$\Xi Config$
$\Xi KeyStore$
$InitDoorLatchAlarm'$
$InitStats'$
$InitAdmin'$
$\Xi UserToken$
$\Xi AdminToken$
$\Xi Finger$
$\Xi Floppy$
$\Xi Keyboard$

StartNonEnrolledStation

$StartContext$

$ownName = \varnothing$
$currentScreen'.2 = insertEnrolmentData$
$currentDisplay' = blank$
$enclaveStatus' = notEnrolled$
$status' = quiescent$
$(\exists\, newElements : \mathbb{P}\, Audit;\ startUnenrolledTISElement : Audit;$
$\quad sizeElement : Audit \to \mathbb{N};\ sizeLog : \mathbb{P}\, Audit \to \mathbb{N};$
$\quad oldestLogTime : \mathbb{P}\, Audit \to TIME;\ newestLogTime : \mathbb{P}\, Audit \to TIME \bullet$
$\quad\quad AddElementsToLog \land startUnenrolledTISElement \in newElements)$

StartEnrolledStation

$StartContext$

$ownName \neq \varnothing$

$currentScreen'.2 = welcomeAdmin$

$currentDisplay' = welcome$

$enclaveStatus' = enclaveQuiescent$

$status' = quiescent$

$(\exists\, newElements : \mathbb{P}\, Audit;\ startEnrolledTISElement : Audit;$

$\quad sizeElement : Audit \rightarrow \mathbb{N};\ sizeLog : \mathbb{P}\, Audit \rightarrow \mathbb{N};$

$\quad oldestLogTime : \mathbb{P}\, Audit \rightarrow TIME;\ newestLogTime : \mathbb{P}\, Audit \rightarrow TIME \bullet$

$\quad\quad AddElementsToLog \land startEnrolledTISElement \in newElements)$

$TISStartup == StartEnrolledStation \lor StartNonEnrolledStation$

TISIdle

$\Xi IDStation$

$\Xi TISControlledRealWorld$

$status = quiescent$

$enclaveStatus = enclaveQuiescent$

$userTokenPresence = absent$

$adminTokenPresence = absent$

$rolePresent = \varnothing$

Apéndice B

Clases de prueba generadas a partir de la especificación de Tokeneer

```
PollUserToken_VIS
  !______PollUserToken_DNF_2
  |       !______PollUserToken_FT_8
  |       |       !______PollUserToken_FT_8_TCASE
  |       |
  |       !______PollUserToken_FT_9
  |       |       !______PollUserToken_FT_9_TCASE
  |       |
  |       !______PollUserToken_FT_10
  |       |       !______PollUserToken_FT_10_TCASE
  |       |
  |       !______PollUserToken_FT_11
  |       |       !______PollUserToken_FT_11_TCASE
  |       |
  |       !______PollUserToken_FT_12
  |               !______PollUserToken_FT_12_TCASE
  |
  |
  !______PollUserToken_DNF_6
          !______PollUserToken_FT_31
                  !______PollUserToken_FT_31_TCASE
```

PollUserToken_VIS

$currentUserToken : TOKENTRY$
$userTokenPresence : PRESENCE$
$latch : LATCH$
$alarm : ALARM$
$display : DISPLAYMESSAGE$
$screen : Screen$
$now : TIME$
$door : DOOR$
$finger : FINGERPRINTTRY$
$userToken, adminToken : TOKENTRY$
$floppy : FLOPPY$
$keyboard : KEYBOARD$

$(\neg\ userToken \neq noT$
$userToken \neq noT) \lor userToken \neq noT \lor (userToken \neq noT$
$\neg\ userToken \neq noT$
$userToken \neq noT) \lor (userToken \neq noT$
$userToken \neq noT) \lor (\neg\ userToken \neq noT$
$userToken = noT) \lor userToken = noT \lor (userToken \neq noT$
$\neg\ userToken \neq noT$
$userToken = noT) \lor (userToken \neq noT$
$userToken = noT)$

PollUserToken_DNF_2

PollUserToken_VIS

$userToken \neq noT$

PollUserToken_FT_8

PollUserToken_DNF_2

$userToken = badT$

PollUserToken_FT_9

PollUserToken_DNF_2

$userToken = goodT1$

PollUserToken_FT_10

PollUserToken_DNF_2

$userToken = goodT2$

PollUserToken_FT_11

PollUserToken_DNF_2

$userToken = goodT3$

PollUserToken_FT_12

PollUserToken_DNF_2

$userToken = goodThe$

PollUserToken_DNF_6

PollUserToken_VIS

$userToken = noT$

PollUserToken_FT_31

PollUserToken_DNF_6

$userToken = noT$

...

Las clases de prueba generadas a partir de la especificación de Tokeneer son muy extensas y no se incluyen en su totalidad en la versión impresa de este trabajo. Consultar la versión digital del Apéndice B.

Apéndice C

Casos de prueba generados a partir de la especificación de Tokeneer

PollUserToken_FT_8_TCASE

PollUserToken_FT_8

$userTokenPresence = present$
$adminToken = noT$
$finger = noFP$
$keyboard = noKB$
$now = 0$
$display = blank$
$screen = (clear, clear, clear)$
$userToken = badT$
$latch = unlocked$
$currentUserToken = noT$
$floppy = noFloppy$
$alarm = silent$
$door = open$

PollUserToken_FT_9_TCASE

PollUserToken_FT_9

$userTokenPresence = present$
$adminToken = noT$
$finger = noFP$
$keyboard = noKB$
$now = 0$
$display = blank$
$screen = (clear, clear, clear)$
$userToken = goodT1$
$latch = unlocked$
$currentUserToken = noT$
$floppy = noFloppy$
$alarm = silent$
$door = open$

PollUserToken_FT_10_TCASE

PollUserToken_FT_10

$userTokenPresence = present$
$adminToken = noT$
$finger = noFP$
$keyboard = noKB$
$now = 0$
$display = blank$
$screen = (clear, clear, clear)$
$userToken = goodT2$
$latch = unlocked$
$currentUserToken = noT$
$floppy = noFloppy$
$alarm = silent$
$door = open$

PollUserToken_FT_11_TCASE

PollUserToken_FT_11

$userTokenPresence = present$
$adminToken = noT$
$finger = noFP$
$keyboard = noKB$
$now = 0$
$display = blank$
$screen = (clear, clear, clear)$
$userToken = goodT3$
$latch = unlocked$
$currentUserToken = noT$
$floppy = noFloppy$
$alarm = silent$
$door = open$

PollUserToken_FT_12_TCASE

PollUserToken_FT_12

$userTokenPresence = present$
$adminToken = noT$
$finger = noFP$
$keyboard = noKB$
$now = 0$
$display = blank$
$screen = (clear, clear, clear)$
$userToken = goodThe$
$latch = unlocked$
$currentUserToken = noT$
$floppy = noFloppy$
$alarm = silent$
$door = open$

PollUserToken_FT_31_TCASE

PollUserToken_FT_31

$userTokenPresence = present$
$adminToken = noT$
$finger = noFP$
$keyboard = noKB$
$now = 0$
$display = blank$
$screen = (clear, clear, clear)$
$userToken = noT$
$latch = unlocked$
$currentUserToken = noT$
$floppy = noFloppy$
$alarm = silent$
$door = open$

...

Los casos de prueba generados a partir de la especificación de Tokeneer son muy extensos y no se incluyen en su totalidad en la versión impresa de este trabajo. Consultar la versión digital del Apéndice C.

Bibliografía

[1] P. Stocks. *"Applying formal methods to software testing"*. Ph.D. dissertation, Department of Computer Science, University of Queensland, 1993.

[2] H. M. Hörcher y J. Peleska. *"Using formal specifications to support software testing"*. Software Quality Journal, vol. 4, pp. 309–327, 1995.

[3] P. Stocks y D. Carrington. *"A framework for specification-based testing"*. IEEE Transactions on Software Engineering, vol. 22, no. 11, pp. 777–793, Nov. 1996.

[4] J. M. Spivey. *"The Z Notation: A Reference Manual"*. Programming Research Group, University of Oxford, Second Edition, 1998.

[5] M. Cristiá y P. Rodriguez Monetti. *"The Fastest 1.3 User's Guide, Automating Software Testing"*. Flowgate Consulting, Apr. 2009.

[6] Praxis High Integrity Systems Ltd. *"Tokeneer ID Station: EAL5 Demonstrator: Summary Report"*. Praxis High Integrity Systems Ltd, S.P1229.81.1., 2008.

[7] Praxis High Integrity Systems Ltd. *"TIS Formal Specification"*. Praxis High Integrity Systems Ltd, S.P1229.41.2., 2008.

[8] Praxis High Integrity Systems Ltd. *"System Test Specification "*. Praxis High Integrity Systems Ltd, S.P1229.63.1., 2008.

[9] ISO/IEC 13568:2002(E) *"Information technology - Z formal specification notation - Syntax, type system and semantics"*.

[10] S. King, J. Hammond, R. Chapman y A. Pryor. *"Is Proof More Cost Effective Than Testing?"*. IEEE Transactions on Software Engineering, Volume 26 Number 8.