Manual para realizar *testing* de TADs especificados en Maude e implementados en C++.

TECHNICAL REPORT: SIC-06/11.
Isabel Pita
Dept. Sistemas Informáticos y Computación
Universidad Complutense de Madrid.
(Última modificación 02-02-2011)

1. Introducción

El presente manual es una guía para ayudar a los estudiantes de la asignatura de Estructuras de Datos y de la Información de segundo curso de Ingeniería Superior en Informática e Ingeniería Técnica en Informática de Sistemas de la UCM a comprobar la corrección de las implementaciones de los tipos de datos realizadas en C++ respecto a una especificación realizada en Maude mediante la generación de casos de prueba. Todo ello se realiza de forma integrada bajo el entorno Eclipse.

La herramienta se encuentra en fase de pruebas. Por ello se ruega que se sigan correctamente las instrucciones para su uso.

Dada una especificación escrita en Maude y una implementación en C++, la herramienta genera los casos de prueba a partir de la especificación, a continuación transforma los términos en Maude a secuencias de instrucciones en C++ que construyen objetos cuyo estado coincide con el especificado. Las instrucciones en C++ componen un programa `main.cpp` que se incorpora al proyecto `Testing` creado en Eclipse para realizar las pruebas. Si ya existía este fichero, se borra y se incorpora el creado para esta prueba. Una vez creado el archivo `main.cpp` se debe tratar como un programa en C++, compilarlo y ejecutarlo. Si el caso de prueba es correcto se muestra en la consola de C++ un punto y se continua con el caso siguiente. Si algún caso falla se muestra un mensaje con el término Maude que produce como resultado la ejecución de la especificación y el resultado obtenido de ejecutar en C++ el caso de prueba.

Cuando se produce un fallo puede ser que exista un error en la implementación en C++, o que el resultado de la especificación no sea el que supone el usuario (error en la especificación). También podría ocurrir que el fallo esté en la implementación de las constructoras, en la implementación de la sobrecarga de los operadores de asignación o comparación, en la implementación del método `mostrar()` o en la traducción del término Maude a las instrucciones en C++. Resumiendo, el alumno debe comprobar si el término que proporciona Maude como resultado es el resultado previsto por el usuario. Si lo es, revisar la implementación realizada. Si el error continúa existiendo comprobar que las instrucciones generadas en el `main.cpp` son correctas. Por último revisar la implementación de las constructoras y de los métodos que sobrecargan la igualdad y la asignación, así como el método de mostrar.

El manual se compone de:

1. Adaptación del entorno Eclipse para la generación de los casos de prueba.
2. Restricciones al lenguaje Maude para utilizar la herramienta de *testing*.
3. Normas que deben seguirse al realizar una implementación en C++ para poder ejecutar los casos de prueba.
4. Como ejecutar los casos de prueba.
5. Problemas más frecuentes.

2. Adaptación del entorno Eclipse para la generación de los casos de prueba.

Para poder ejecutar la herramienta de *testing* se debe tener instalado el entorno Eclipse IDE for C/C++ Developers y el ejecutable de Maude según se explica en el manual *Guía rápida sobre ejecución de especificaciones algebraicas en Maude bajo el entorno Eclipse para estudiantes de Estructuras de Datos. (Actualizado para poder utilizar la herramienta de testing.)*. Se debe cargar el fichero `dd.maude` en el directorio en que se encuentra el ejecutable de Maude, y el plugin `es.ucm.dsic.fadoss.maudeCtesting` en el directorio de plugins de Eclipse.

Al configurar Eclipse para que ejecute Maude (ver el manual de ejecución de especificaciones en Maude) se debe de poner el fichero `dd.maude` como ejecutable de *Full Maude* y marcar la opción de ejecutar con *Full Maude*. En las opciones avanzadas se debe marcar *Other* en *Operating System* y escribir el siguiente comando `cmd.exe , /C , bash -c .ecʰo $$; exec %maudebin% %fullmaude% -interactive -no-tecla -no-wrap"`. Sobre el comando puesto por defecto únicamente hay que agregar al final la instrucción `-no-wrap` (antes de las comillas).

3. Restricciones al lenguaje Maude para utilizar la herramienta de *testing*.

En esta sección se enumeran una serie de normas que se deben seguir para poder ejecutar una especificación Maude en el programa de *Testing*. Algunas de estas restricciones están motivadas por deficiencias en la herramienta que se irán solventando.

Restricciones sobre la definición de los módulos:

- Las especificaciones deben escribirse en Full Maude, esto es, cada módulo que se defina debe ir entre paréntesis. Ademas debe darse un identificador diferente a cada ecuación.
- La herramienta solo trata módulos funcionales (`fmod`).
- Puede escribirse más de un módulo en cada fichero, pero el testing se realiza sobre el último módulo que se haya cargado. Si este módulo incluye a otros, también se puede realizar testing sobre las operaciones de estos.
- Si se quieren incluir módulos definidos en ficheros distintos de aquel en que se encuentra el TAD que se está definiendo (fichero actual), se debe incluir el comando *in nombre-archivo* al principio del fichero actual. El fichero debe encontrarse también en el espacio de trabajo dentro del proyecto *Testing*.

  Hay que tener en cuenta que si se carga el fichero con el sistema Maude (no con el *testing*), los ficheros incluidos no se buscarán en el espacio de trabajo sino en el directorio en que esté cargado Maude y por lo tanto no se encontrarán. En este caso es necesario cargar los módulos como se hace en el sistema Maude. Si no se comenta la línea en que se incluye el módulo en el fichero, se producirá un warning de que el fichero no se encuentra, pero no tendrá más consecuencias.

Restricciones sobre la definición de las operaciones (constructoras, modificadoras y observadoras):

- No utilizar notación infija en las especificaciones. Todas las operaciones deben estar en notación prefija.

- No se pueden utilizar operaciones parciales, que utilicen el símbolo $\sim>$.
- Las constructoras de todos los tipos que se utilicen a la vez deben tener nombre diferente.
- La constructora del tipo no puede ser privada, y no puede utilizar tipos complejos como parámetros (esto elimina por el momento el poder probar el TAD de las secuencias). Sin embargo si pueden utilizarse tipos definidos por el usuario en parámetros y resultados de las operaciones modificadoras y observadoras, como ocurre en la práctica de los árboles de búsqueda.
- El primer parámetro utilizado en la definición de las operaciones modificadoras y observadoras debe ser el sistema (TAD que se está especificando). No se puede hacer testing sobre operaciones cuyo primer parámetro no sea el sistema, por ejemplo operaciones cuyo resultado no dependa del sistema que se está definiendo, esto elimina la operación `unit` de las listas. No importa el orden de los parámetros en las operaciones constructoras.
- No se hace testing sobre los casos erroneos. Si un término se reduce a error, no se genera caso de prueba sobre él.

Constantes: Se han definido las siguientes constantes. No se puede hacer testing sobre especificaciones que superen estas constantes:

- Número máximo de parámetros en una operación (constructoras, observadoras y modificadoras) 5;
- Parametros definidos en el módulo 3;
- Número máximo de constructoras en la especificación 5;
- Número máximo de operaciones modificadoras y observadoras en la especificación, excluidas las constructoras 20;
- Número máximo de tipos utilizados en la especificación 15; debe observarse que este máximo se refiere también al número de tipos definidos en el fichero de *mapping* entre los identificadores de Maude y de C++. Deben incluirse todos los tipos y subtipos que puedan aparecer como resultado.
- Número máximo de tipos que se definen en una teoría utilizada como parámetro 3;

4. Normas que deben seguirse al realizar una implementación en C++ para poder ejecutar los casos de prueba.

Para una especificación se pueden realizar varias implementaciones, por ejemplo, una con memoria estática, otra con memoria dinámica etc. Para cada implementación se proporciona al sistema un fichero de *mapping* en el que se indica como se relacionan los nombres de las operaciones en Maude y en la implementación C++ así como los ficheros que es necesario incluir en la implementación.

1. Debe existir un fichero de interfaz (`.h`) y uno de implementación (`.cpp`) por cada implementación que se quiera definir. Ademas deben denominarse igual que el fichero de mapping asociado a esa implementación.
2. Todos los TAD que se implementen deben tener definida la sobrecarga del operador de igualdad, sobrecarga del operador de asignación y un método `mostrar` que muestre en la consola el contenido del TAD.

3. Las constructoras de la especificación pueden implementarse como constructoras del tipo o por medio de un método. La forma de implementación se indica en el fichero de *mapping*: si existe nombre para el método en C++ entonces la implementación se realiza con el método, si no se proporciona nombre para el método se asume que se implementa mediante la constructora.

   No se puede implementar una constructora al mismo tiempo como constructora C++ y con un método.

   Solo puede implementarse una constructora, esto es, no se puede definir una constructora sin parámetros y otra con parámetros, ni varias constructoras con distinto número de parámetros.

4. Los métodos que implementan operaciones modificadoras, esto es, aquellas que en la especificación cambian el estado, deben implementarse con funciones `void` que cambien el estado del objeto. No deben devolver el objeto modificado.

5. Las operaciones observadoras deben implementarse con métodos que devuelvan un valor del tipo resultado. Este tipo no puede ser un puntero, debe ser una copia del valor.

6. No se pueden utilizar tipos enumerados en las implementaciones salvo el tipo definido en la práctica de la *Cesta de la compra*.

5. Como ejecutar los casos de prueba.

Una vez abierto Eclipse, se debe crear un proyecto denominado `Testing` (El sistema diferencia mayúsculas de minúsculas). En este proyecto se definirán los archivos `.maude` con las especificaciones algebraicas y los archivos `.h` y `.cpp` con la interfaz y la implementación de los TADs en C++ sobre los que se quiera realizar el *testing*. No se puede realizar *testing* sobre archivos que se encuentren en proyectos que no se denominen `Testing`, aunque aparezcan los botones correspondientes.

Para poder ejecutar los casos de prueba hay que definir un fichero de texto (`.txt`) con la información necesaria para crear el programa principal en C++ a partir de los casos de prueba. En particular se define la correspondencia entre los identificadores utilizados en Maude y los utilizados en C++, así como el nombre de los archivos que se deben importar. El identificador del fichero debe coincidir con él de la implementacion en C++, aunque su extensión es distinta. La estructura de este fichero se proporciona al final del capítulo.

Acciones. Estando abierto en el editor de Eclipse el archivo `.maude` con la especificación que se desea probar, aparecen en la barra de herramientas tres iconos que son los utilizados en el testing.

1.  (`init`). Se utiliza para iniciar la herramienta. Esta opción debe utilizarse sólo una vez al comienzo de la sesión. El proceso toma bastante tiempo (2-3 minutos).

2.  (`exec`). Se utiliza cada vez que se quiere probar una operación de una especificación. Antes de utilizarla por primera vez debe inicializarse el proceso con la opción anterior.

3.  (`stop`). Se utiliza cuando se quiere finalizar la sesión. Si se utiliza y luego quiere realizarse otra vez el testing hará falta volver a inicializarlo.

Proceso de *testing*.

1. Una vez que se ha inicializado el proceso y estando abierto en el editor el fichero `.maude` con la especificación de la implementación que se desea probar, se pulsa el botón de ejecutar (`Exec`).

2. Si es la primera operación que se prueba de este TAD, la herramienta pregunta por el fichero utilizado en la interfaz de la implementación. (El identificador de este fichero debe coincidir con el fichero de texto donde se encuentra el *mapping* de las funciones y tipos.). Este fichero debe encontrarse en el proyecto `Testing`. La herramienta muestra una lista con todos los ficheros de interfaz existentes en el proyecto, se debe seleccionar uno y pulsar el botón `OK`. No escribir el nombre en la caja, sino marcar con el ratón el fichero y pulsar el botón `OK`.

3. Si se acaba de probar una operación que pertenece al mismo fichero, el sistema pregunta si se desea modificar el fichero de interfaz. Si esta nueva operación se desea probar con la misma implementación que la operación anterior no es necesario modificar el fichero y basta con pulsar el botón `OK` ya que se proporciona por defecto la opción `No`. Si se desea modificar el nombre del fichero en que está realizada la implementación (por ejemplo porque se haya probado una implementación estática y ahora se desee probar una dinámica) entonces se escribe en la caja `Si` se pulsa el botón `OK` y el sistema presentará la lista de todas las interfaces del proyecto `Testing` para que se seleccione una.

4. A continuación muestra una lista con las operaciones sobre las que se puede hacer el testing. Seleccionar una de ellas con el ratón y pulsar el botón `OK` o bien simplemente *enter*. Como en el caso anterior no escribir el nombre en la caja. Las operaciones las selecciona el sistema teniendo en cuenta el fichero de *mapping*. Una operación definida en la especificación, que no aparece en el fichero de *mapping*, se considera como privada de la especificación y por lo tanto no implementada.

5. La última pregunta que realiza la herramienta es el número de casos de prueba que se quieren generar. Se recomienda utilizar el valor que se indica en las prácticas, ya que hay casos en que debido a deficiencias en la herramienta el tiempo de respuesta es inaceptable. No se recomienda superar los 200 casos por el momento.

6. Cuando la herramienta termina de ejecutarse (no debe tardar mas de uno o dos minutos) se muestra un mensaje por pantalla. La herramienta ha generado un fichero `main.cpp` en el proyecto `Testing` con los casos de prueba. Este fichero debe tratarse como un programa C++, puede consultarse si se desea aunque no es necesario. Compilarlo y ejecutarlo como se hace normalmente en Eclipse.

El fichero de *mapping*. En el archivo de *mapping* se declaran los archivos que se deben incluir para compilar el programa C++, si estos archivos son diferentes en diferentes implementaciones del tipo, entonces se definirán diferentes ficheros `.txt` para la misma especificación.

En el archivo se declara el tipo que se está definiendo y la instanciación que se hace de él en el caso de ser parametrizado.Tambien se declaran las instanciaciones de los tipos parametrizados que se utilizan en la definición del tipo. Por lo tanto si se quiere probar el tipo con diversas instanciaciones es necesario definir un fichero `.txt` distinto por cada instanciación que se quiera probar.

La estructura por líneas del fichero es la siguiente:

- Identificador del tipo en Maude que se está definiendo e identificador del tipo en la implementación en C++. Si el tipo es parametrizado debe escribirse el tipo en Maude tal como lo denota el sistema: identificador del tipo, ‘{, identificador de la vista utilizada para instanciar el tipo, ‘} y el tipo en C++ tal como se declaran los objetos del tipo. Por ejemplo: `Lista‘{vInt‘} Lista<int>`

- En líneas sucesivas los diferentes tipos que se utilizan en el modulo. Primero se nombra el tipo en Maude y separado por un blanco el tipo en C++.

- Para terminar la definición de los tipos se escribe en una línea la palabra `finsort`

- En líneas sucesivas todas las constructoras del tipo. Para cada constructora se escribe el identificador en Maude y a continuación, separado por un blanco el identificador en C++. Si la constructora no está implementada mediante ningún método C++, sino que se implementa en la declaración del objeto se da solo el identificador en Maude.

- Para terminar la definición de las constructoras se escribe en una línea la palabra `finconstructora`
- En líneas sucesivas todas las operaciones sobre las que se puede desear hacer testing. Se escribe el identificador en Maude y a continuación el identificador en C++. Todas las operaciones deben estar especificadas e implementadas. Las operaciones privadas de la especificación o privadas de la implementación no aparecen en el fichero.
- Para terminar la definición de las operaciones se escribe una línea con la palabra `finoperaciones`.
- Cabecera del programa, con los comentarios iniciales, y los includes que hagan falta. Por ejemplo:

```
/* testing del TAD Pilas  */
#include<iostream>
using namespace std;
#include "PilaEstatica.cpp"
```

- Para indicar el fin de la cabecera del programa una línea con la palabra `fincabecera`.
- El tratamiento de errores que se quiere realizar. Por ejemplo

```
catch(int n)
```

- Para indicar el final de los errores una línea con la palabra `finerrores`.

6. Problemas más frecuentes.

Como se ha indicado en la introducción, la herramienta está en pruebas.
Situaciones conflictivas que se pueden producir son:

1. Si se produce el mensaje: `Inconsistence in the operations defined in the mapping file`, comprobar que el identificador de las operaciones modificadoras y observadoras definidas en el fichero de *mapping* coincide con el del fichero `.maude`. Si coincide reportar el error a la profesora.
2. Si se produce el mensaje: `Maude module parameters inconsistent with C++ TAD instantiation` reportar directamente el error.
3. Si se produce el mensaje: `The ctor was not found in the Maude module. Check consistence of Maude module and the mapping file`, comprobar que el identificador de las constructoras en Maude coincide en el fichero de *mapping* y en el módulo Maude. Comprobar que se están cumpliendo las restricciones sobre las constructoras tanto en Maude como en C++.
4. Si se produce el mensaje: `The operation was not found in the Maude module`, actuar igual que en el primer caso.
5. Si al pulsar el botón `cancel` en alguna ventana se produce un error, se puede volver a utilizar la herramienta sin necesidad de volver a inicializarla.
6. Si al pulsar el botón `exec` la herramienta tarda mas de 5 minutos en acabar lo mas probable es que se haya quedado colgada. Esto puede ser debido a que el número de casos de prueba sea mayor que lo que puede tratar el sistema. Dependiendo de la sintaxis de la operación este valor será uno u otro y no tiene porque ser un valor elevado. Se ha detectado que algunas operaciones de los árboles pueden dar problemas con valores de 20.

   En este caso debemos cerrar Eclipse y finalizar manualmente el proceso de ejecución de Maude.