Verificador para CTL

Laura Alicia Leonides Jiménez

En el presente documento se especifica cómo fue implementado el programa, cuáles archivos se entregan en la distribución anexa a la versión electrónica del documento y cómo debe ejecutarse el programa.

Asimismo, se incluyen algunos ejemplos de los archivos en los que debe especificarse el modelo y la fórmula a verificar.

1. Implementación

El programa fue realizado en Haskell 98 y contiene los siguientes módulos:

SintaxisCTL. En él se define la sintaxis de las fórmulas de CTL que se podrán procesar.

VerificadorCTL. Se encarga de verificar un modelo y una fórmula.

Rela. En él se define la relación de transición.

Modelo. Contiene las definiciones de tipos de datos

Main. Módulo principal del sistema. Encargado de obtener el archivo con las definiciones y de procesar la información que ahí se encuentra (modelo y fórmula) necesarias para especificar un modelo.

Lector. Se encarga de leer el archivo en el que se encuentra la definición del modelo y la fórmula. Parsea cada una de las partes involucradas.

Todos los archivos fuente utilizan la codificación UTF-8.

2. Distribución

Junto con la versión electrónica de este documento, se entrega el archivo `VerificadorCTL.tar.gz`. Dicho archivo debe descomprimirse y entonces se tendrá la siguiente estructura de directorios:

```
VerificadorCTL
|-- VerificadorCTL
|-- modeloFórmula.dat
‘-- src
    |-- Lector.hs
    |-- Main.hs
    |-- Modelo.hs
    |-- Rela.hs
    |-- SintaxisCTL.hs
    ‘-- VerificadorCTL.hs
```

3. Ejecución

Para ejecutar el programa tenemos tres opciones:

Con el ejecutable incluido

En la distribución se incluye el archivo VerificadorCTL, que es un ejecutable generado para Linux Mandriva. Se tiene que ejecutar de la siguiente manera:

```
[cotupha@ronja VerificadorCTL]$ ./VerificadorCTL
```

Con un intérprete

Basta con cargar el módulo `Main` en un intérprete de Haskell (las pruebas se realizaron utilizando GHCi Versión 6.8.2) e invocar a la función `main`, de la siguiente manera:

```
[cotupha@ronja src]$ ghci Main
GHCi, version 6.8.2: http://www.haskell.org/ghc/  :? for help
Loading package base ... linking ... done.
Compiling SintaxisCTL   (SintaxisCTL.hs, interpreted)
```

```
Compiling Rela          (Rela.hs, interpreted)
Compiling Modelo        (Modelo.hs, interpreted)
Compiling Lector        (Lector.hs, interpreted)
Compiling VerificadorCTL(VerificadorCTL.hs, interpreted)
Compiling Main          (Main.hs, interpreted)
Ok,modules loaded:VerificadorCTL,Main,Lector,Modelo,Rela,SintaxisCTL.
*Main> main
```

Con un ejecutable generado para otro sistema

Si se requiere de un ejecutable para otro sistema operativo, debe compilarse el código, generar el nuevo ejecutable y, posteriormente correrlo.

Una vez que se ha ejecutado el programa (con alguno de los tres métodos antes mencionados), éste solicita al usuario que proporcione el nombre del archivo en el que se encuentra almacenada la definición del modelo y la fórmula. Es importante señalar que debe proporcionarse la ruta relativa al archivo desde el directorio en el que se está ejecutando el programa.

Después de especificar el nombre del archivo, el programa parsea dicho archivo y muestra en pantalla los estados que satisfacen a la fórmula dada en el modelo especificado, como se muestra a continuación:

```
Dame el nombre del archivo con la especificación del modelo M
y la fórmula a
../modeloFórmula.dat
Los estados s tales que M,s|=a son:
fromList ["s0","s1","s2"]
```

4. Definición del modelo y fórmula

Es necesario proporcionar al programa un archivo en el que se encuentren las definiciones necesarias para determinar el modelo y la fórmula a verificar. Deben especificarse, en el orden presentado, los siguientes componentes:

- Conjunto de estados.
- Conjunto de átomos.

- Relación de transición.
- Función de etiquetamiento.
- Fórmula.

A continuación se incluye un ejemplo de dicho archivo:

Listing 1: Ejemplo

```
1  {s0, s1, s2}
2  {p, q, r}
3  {(s0, s1), (s1, s0), (s0,s2), (s2, s2), (s1, s2)}
4  #s0>{p,q}#s1>{q,r}#s2>{r}
5  (And(Lit r)(Lit q))
```

Ejemplos de otras fórmulas:

Listing 2: Fórmulas

```
1  (AG (Then (Or (Or (Lit p)(Lit q))(Lit r))(EF (EG (Lit r)))))
2  (AU (Lit p)(Lit r))
3  (EU (And (Lit p)(Lit q))(Lit r))
4  (AF (Lit r))
5  (EG (Lit r))
6  (Not (EF (And (Lit p)(Lit r))))
7  (Not (AX (And (Lit q)(Lit r))))
8  (EX (And (Lit q)(Lit r)))
9  (Not (Lit r))
10 (And (Lit p)(Lit q))
```