Verificador para PCTL

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:

SintaxisPCTL. En él se define la sintaxis de las fórmulas de PCTL que se podrán procesar.

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

Rela. Contiene funciones auxiliares para manejar la relación de transición con probabilidades.

Modelo. Contiene las definiciones de tipos de datos del modelo.

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.

ArrayUtil. Contiene funciones auxiliares para facilitar el manejo de arreglos y matrices.

Asimismo, se utilizó la biblioteca Haskell DSP como auxiliar en el manejo de matrices.
Todos los archivos fuente utilizan la codificación UTF-8.

2. Distribución

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

```
VerificadorPCTL
|-- prueba.dat
`-- src
    |-- ArrayUtil.hs
    |-- Lector.hs
    |-- Main.hs
    |-- Matrix
    |   |-- LU.hs
    |   `-- Matrix.hs
    |-- Modelo.hs
    |-- Rela.hs
    |-- SintaxisPCTL.hs
    `-- VerificadorPCTL.hs
```

3. Ejecución

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.
[1 of 9] Compiling SintaxisPCTL     ( SintaxisPCTL.hs, interpreted )
```

```
[2 of 9] Compiling Matrix.Matrix    ( Matrix/Matrix.hs, interpreted )
[3 of 9] Compiling Matrix.LU        ( Matrix/LU.hs, interpreted )
[4 of 9] Compiling Modelo           ( Modelo.hs, interpreted )
[5 of 9] Compiling Lector           ( Lector.hs, interpreted )
[6 of 9] Compiling Rela             ( Rela.hs, interpreted )
[7 of 9] Compiling ArrayUtil        ( ArrayUtil.hs, interpreted )
[8 of 9] Compiling VerificadorPCTL  ( VerificadorPCTL.hs, interpreted )
[9 of 9] Compiling Main             ( Main.hs, interpreted )
Ok, modules loaded: VerificadorPCTL, Main, ArrayUtil, Modelo, Matrix.LU,
Matrix.Matrix, Lector, SintaxisPCTL, Rela.
*Main>
```

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 especificaciónn del modelo M y la
fórmula a verificar
../prueba.dat
Los estados s tales que M,s|=a son:
fromList [2,3,4]
```

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 átomos.
- Número de estados.
- Relación de transición con probabilidades.
- Función de etiquetamiento.
- Fórmula.

A continuación se incluye un ejemplo de dicho archivo:

```
1  {not, try,  fail ,  succ}
2  4
3  (2,  1.0)
4  (2,   0.01),(3,   0.01),(4,   0.98)
5  (1,  1.0  )
6  (4,  1.0  )
7  #1>{not}#2>{try}#3>{fail}#4>{succ}
8  (P >= 0.9 [X (Or (Not(Lit try))(Lit succ))])
```

Ejemplos de otras fórmulas:

```
1  ********** X **********
2  (P >= 0.9 [X (Or (Not(Lit try))(Lit succ))])
3  (P >= 0.9 [X (Not (And (Lit try)(Not (Lit succ))))])
4  (P <= 0.99 [X (Not (And (Lit try)(Not (Lit succ))))])
5  (P < 0.99 [X (Not (And (Lit try)(Not (Lit succ))))])
6  ********** Uk **********
7  (P > 0.98 [Uk 2 (Top) (Lit succ)])
8  (P >= 1.0 [Uk 2 (Top) (Lit succ)])
9  ********** U **********
10 (P > 0.99 [U (Lit try) (Lit  succ)])
11 (P < 0.99 [U (Lit try) (Lit  succ)])
12 ********** Props **********
13 (Not (And (Lit try)(Not (Lit succ))))
14 (Lit  succ)
15 (Top)
16 (Not (Lit try))
17 (And (Lit try) (Lit  fail ))
18 (Lit  try)
```