Código Promela en LaTeX Métodos Formales Otoño 2012 Sección 101

José de Jesús Lavalle Martínez

13 de septiembre de 2012

Resumen

Este documento sirve para aprender a escribir código Promela en LaTeX.

En general siempre hay al menos tres maneras de presentar en LaTeX código de lenguajes de programación, la primera es usar el ambiente `verbatim`, la segunda usando el ambiente `alltt`, para la tercera se debe usar el ambiente `lstlisting`.

1. verbatim

En la primera manera se escribe el código entre `\begin{verbatim}` y `\end{verbatim}`, note que en este ambiente LaTeX no interpreta los comandos, no se necesita incluir algún paquete, así obtendrá:

Código Promela 1.1. Programa para hallar el cociente y el residuo de dos números naturales, presentado usando el ambiente `verbatim`.

```
/* Copyright 2007 by Moti Ben-Ari
under the GNU GPL; see readme.txt */
```

```
active proctype P() {
   int dividend = 15, divisor  = 4;
   int quotient = 0, remainder = 0;
   int n = dividend;

   assert (dividend >= 0 && divisor > 0);

   do
      :: n != 0 ->
         assert(dividend == quotient * divisor + remainder + n);
         assert(0 <= remainder && remainder < divisor);

         if
            :: remainder + 1 == divisor ->
               quotient++;
               remainder = 0
            :: else ->
               remainder++
         fi;
         n--
      :: else ->
         break
   od;
   printf("%d divided by %d = %d, remainder = %d\n",
             dividend, divisor, quotient, remainder);
   assert (dividend == quotient * divisor + remainder);
   assert (0 <= remainder && remainder < divisor);
}
```

2. alltt

En la segunda manera se escribe el código entre `\begin{alltt}` y `\end{alltt}`, note que en este ambiente LaTeX sí interpreta los comandos, necesita poner en el preámbulo `\usepackage{alltt}`, para obtener:

Código Promela 2.1. Programa para hallar el cociente y el residuo de dos números naturales, presentado usando el ambiente `alltt`.

```
/* Copyright 2007 by Moti Ben-Ari
under the GNU GPL; see readme.txt */

active proctype P()
   int dividend = 15, divisor  = 4;
   int quotient = 0, remainder = 0;
   int n = dividend;

   assert (dividend >= 0 && divisor > 0);

   do
      :: n != 0 ->
         assert(dividend == quotient * divisor + remainder + n);
         assert(0 <= remainder && remainder < divisor);

         if
            :: remainder + 1 == divisor ->
               quotient++;
               remainder = 0
            :: else ->
               remainder++
         fi;
         n--
      :: else ->
         break
   od;
   printf("%d divided by%d =%d, remainder =%d \n ",
             dividend, divisor, quotient, remainder);
   assert (dividend == quotient * divisor + remainder);
   assert (0 <= remainder && remainder < divisor);
```

3. lstlisting

La tercera forma es escribiendo el código entre `\begin{lstlisting}` y `\end{lstlisting}` con opción `[language=Promela]`, en este ambiente tampoco se interpretan los comandos, lo único que hace el paquete es formatear

el código de acuerdo al lenguaje seleccionado y poner en negras las palabras reservadas del lenguaje, ponga en el preámbulo `\usepackage{listings}`.

Código Promela 3.1. Programa para hallar el cociente y el residuo de dos números naturales, presentado usando el ambiente `lstlisting`.

```
/* Copyright 2007 by Moti Ben-Ari
under the GNU GPL; see readme.txt */

active proctype P() {
    int dividend = 15, divisor  = 4;
    int quotient = 0, remainder = 0;
    int n = dividend;

    assert (dividend >= 0 && divisor > 0);

    do
        :: n != 0 ->
            assert(dividend == quotient * divisor + remainder + n);
            assert(0 <= remainder && remainder < divisor);

            if
                :: remainder + 1 == divisor ->
                    quotient++;
                    remainder = 0
                :: else ->
                    remainder++
            fi;
            n--
        :: else ->
            break
    od;
    printf("\%d divided by %d = %d, remainder = %d\n", dividend, divisor
    assert (dividend == quotient * divisor + remainder);
    assert (0 <= remainder && remainder < divisor);
}
```

Observe que las líneas largas se salen del margen derecho de la hoja. Ponga el código entre `\begin{lstlisting}` y `\end{lstlisting}` con opción

`[language=Promela,numbers=left,breaklines=true]` si quiere que el paquete `listings` numere las líneas de código y parta automáticamente las líneas muy grandes, obtendrá:

Código Promela 3.2. Programa para hallar el cociente y el residuo de dos números naturales, presentado usando el ambiente `lstlisting` con las líneas numeradas y ruptura automática de líneas largas.

```
1  /* Copyright 2007 by Moti Ben-Ari
2  under the GNU GPL; see readme.txt */
3
4  active proctype P() {
5     int dividend = 15, divisor  = 4;
6     int quotient = 0, remainder = 0;
7     int n = dividend;
8
9     assert (dividend >= 0 && divisor > 0);
10
11    do
12       :: n != 0 ->
13          assert(dividend == quotient * divisor +
               remainder + n);
14          assert(0 <= remainder && remainder < divisor);
15
16          if
17             :: remainder + 1 == divisor ->
18                quotient++;
19                remainder = 0
20             :: else ->
21                remainder++
22          fi;
23          n--
24       :: else ->
25          break
26    od;
27    printf("%d divided by %d = %d, remainder = %d\n",
         dividend, divisor, quotient, remainder);
28    assert (dividend == quotient * divisor + remainder);
29    assert (0 <= remainder && remainder < divisor);
```

```
30 }
```

Si además quiere encerrar el código en una caja sombreada agregue la opción `frame=shadowbox`.

Código Promela 3.3. Programa para hallar el cociente y el residuo de dos números naturales, presentado usando el ambiente `lstlisting` con las líneas numeradas, ruptura automática de líneas largas y caja sombreada.

```
1  /* Copyright 2007 by Moti Ben-Ari
2  under the GNU GPL; see readme.txt */
3
4  active proctype P() {
5     int dividend = 15, divisor  = 4;
6     int quotient = 0, remainder = 0;
7     int n = dividend;
8
9     assert (dividend >= 0 && divisor > 0);
10
11    do
12       :: n != 0 ->
13          assert(dividend == quotient * divisor +
               remainder + n);
14          assert(0 <= remainder && remainder < divisor);
15
16          if
17             :: remainder + 1 == divisor ->
18                quotient++;
19                remainder = 0
20             :: else ->
21                remainder++
22          fi;
23          n--
24       :: else ->
25          break
26    od;
27    printf("%d divided by %d = %d, remainder = %d\n",
         dividend, divisor, quotient, remainder);
28    assert (dividend == quotient * divisor + remainder);
```

```
29      assert (0 <= remainder && remainder < divisor);
30  }
```

Como puede observar en la línea 27 se imprimen mal los carácteres de formato, para solucionarlo utilizamos la opción `escapeinside=‘’` para que lo que encerremos entre estos dos carácteres sea interpretado por LaTeX, es decir `"‘\%d divided by \%d = \%d, remainder = \%d \textbackslash\!n’"`.

Código Promela 3.4. Programa para hallar el cociente y el residuo de dos números naturales, presentado usando el ambiente `lstlisting` con las líneas numeradas, ruptura automática de líneas largas y carácteres interpretados por LaTeX.

```
1  /* Copyright 2007 by Moti Ben-Ari
2  under the GNU GPL; see readme.txt */
3
4  active proctype P() {
5      int dividend = 15, divisor  = 4;
6      int quotient = 0, remainder = 0;
7      int n = dividend;
8
9      assert (dividend >= 0 && divisor > 0);
10
11     do
12         :: n != 0 ->
13             assert(dividend == quotient * divisor +
                   remainder + n);
14             assert(0 <= remainder && remainder < divisor);
15
16             if
17                 :: remainder + 1 == divisor ->
18                     quotient++;
19                     remainder = 0
20                 :: else ->
21                     remainder++
22             fi;
23             n--
24         :: else ->
```

```
25              break
26      od;
27      printf(" %d divided by %d = %d, remainder = %d
            \n", dividend, divisor, quotient, remainder);
28      assert (dividend == quotient * divisor + remainder);
29      assert (0 <= remainder && remainder < divisor);
30  }
```

Lo malo es que al habilitar a LaTeX para que interprete el código, el paquete `listings` pierde el control, si ahora ponemos el código en una caja el paquete se equivoca al pintarla, obtenemos:

Código Promela 3.5. Programa para hallar el cociente y el residuo de dos números naturales, presentado usando el ambiente `lstlisting` con las líneas numeradas, ruptura automática de líneas largas, carácteres especiales interpretados por LaTeX y caja sombreada.

```
1  /* Copyright 2007 by Moti Ben-Ari
2  under the GNU GPL; see readme.txt */
3
4  active proctype P() {
5      int dividend = 15, divisor  = 4;
6      int quotient = 0, remainder = 0;
7      int n = dividend;
8
9      assert (dividend >= 0 && divisor > 0);
10
11     do
12         :: n != 0 ->
13            assert(dividend == quotient * divisor +
                  remainder + n);
14            assert(0 <= remainder && remainder < divisor);
15
16            if
17               :: remainder + 1 == divisor ->
18                  quotient++;
19                  remainder = 0
20               :: else ->
21                  remainder++
```

```
22              fi;
23              n--
24          :: else ->
25              break
26      od;
27      printf("%d divided by %d = %d, remainder = %d
            \n", dividend, divisor, quotient, remainder);
28      assert (dividend == quotient * divisor + remainder);
29      assert (0 <= remainder && remainder < divisor);
30  }
```

Como no se puede tener todo, debemos tomar una decisión, es decir, si queremos la caja deshabilitamos la ruputura automática de líneas, nosotros partimos las líneas, para obtener:

Código Promela 3.6. Programa para hallar el cociente y el residuo de dos números naturales, presentado usando el ambiente `lstlisting` con las líneas numeradas, ruptura manual de líneas largas, carácteres especiales interpretados por LaTeX y caja sombreada.

```
1  /* Copyright 2007 by Moti Ben-Ari
2  under the GNU GPL; see readme.txt */
3
4  active proctype P() {
5      int dividend = 15, divisor  = 4;
6      int quotient = 0, remainder = 0;
7      int n = dividend;
8
9      assert (dividend >= 0 && divisor > 0);
10
11     do
12         :: n != 0 ->
13             assert(dividend == quotient * divisor +
14                     remainder + n);
15             assert(0 <= remainder && remainder < divisor);
16
17             if
18                 :: remainder + 1 == divisor ->
19                     quotient++;
```

```
20                  remainder = 0
21              :: else ->
22                  remainder++
23          fi;
24          n--
25      :: else ->
26          break
27  od;
28  printf("%d divided by %d = %d, remainder = %d \n",
29          dividend, divisor, quotient, remainder);
30  assert (dividend == quotient * divisor + remainder);
31  assert (0 <= remainder && remainder < divisor);
32 }
```