LaTeX Style for Uppaal Models

Marius Mikučionis
marius@cs.aau.dk

January 29, 2018

1 Introduction

The intent of this manual is to document the features of `uppaal.sty` LaTeX style package. The `uppaal.sty` package has been developed over many years of publishing papers related to Uppaal. The initial intent was to gather common definitions used throughout papers. The common definitions outgrew into package and finally became a piece of software it is now.

1.1 Example

Listing 1 shows the simplest LaTeX code to typeset Uppaal code:

Listing 1: The simplest Uppaal typesetting in LaTeX.

```
\begin{uppaalcode}
int lIZERO = 0;   // note the characters l, I, 0 and O
int distance = 5; // approximated distance between cars
int velocityEgo, velocityFront; /* approximated velocities */
int accelerationEgo, accelerationFront; /** acceleration */
void updateDiscrete() {
    int newVel, oldVel = velocityFront - velocityEgo;
    velocityEgo = velocityEgo + accelerationEgo;
    velocityFront = velocityFront + accelerationFront;
    newVel = velocityFront - velocityEgo;
    if (distance > maxSensorDistance) {
        distance = maxSensorDistance + 1;
    } else { // $d \approx \sum_i \frac{v_i+v_{i+1}}{2}\Delta t$
        distance += (oldVel + newVel)/2;
    }
}
\end{uppaalcode}
```

The result is the following minimalistic listing:

```
int lIZERO = 0;   // note the characters l, I, 0 and O
int distance = 5; // approximated distance between cars
int velocityEgo, velocityFront; /* approximated velocities */
int accelerationEgo, accelerationFront; /** acceleration */
void updateDiscrete() {
    int newVel, oldVel = velocityFront - velocityEgo;
    velocityEgo = velocityEgo + accelerationEgo;
    velocityFront = velocityFront + accelerationFront;
    newVel = velocityFront - velocityEgo;
    if (distance > maxSensorDistance) {
        distance = maxSensorDistance + 1;
    } else { // d ≈ ∑_i (v_i+v_{i+1})/2 Δt
        distance += (oldVel + newVel)/2;
    }
}
```

Listing 3 shows the same listing with customized **uppaalcode** environment. Properties can also be typeset as inline code using Listing 4:

Listing 2: LaTeX code.

```
\begin{uppaalcode}[caption={Heavily customized listing.},
  label={lst:example2},
  captionpos=b, % put caption at the bottom of the listing
  float, % make it "float", prevents page-breaks in listing
  frame=shadowbox, rulesepcolor=\color{lightgray},
  numbers=left,numberstyle=\tiny,numbersep=3mm,
  xleftmargin=5mm]
int lIZERO = 0;   // note the characters l, I, 0 and O
int distance = 5; // approximated distance between cars
int velocityEgo, velocityFront; /* approximated velocities */
int accelerationEgo, accelerationFront; /** acceleration */
void updateDiscrete() {
    int velNew, velOld = velocityFront - velocityEgo;
    velocityEgo = velocityEgo + accelerationEgo;
    velocityFront = velocityFront + accelerationFront;
    velNew = velocityFront - velocityEgo;
    if (distance > maxSensorDistance) {
        distance = maxSensorDistance + 1;
    } else { // $d \approx \sum_i \frac{v_i+v_{i+1}}{2}\Delta t$
        distance += (velOld + velNew)/2;
    }
}
\end{uppaalcode}
```

```
int lIZERO = 0;   // note the characters l, I, 0 and O
int distance = 5; // approximated distance between cars
int velocityEgo, velocityFront; /* approximated velocities */
int accelerationEgo, accelerationFront; /** acceleration */
void updateDiscrete() {
    int velNew, velOld = velocityFront - velocityEgo;
    velocityEgo = velocityEgo + accelerationEgo;
    velocityFront = velocityFront + accelerationFront;
    velNew = velocityFront - velocityEgo;
    if (distance > maxSensorDistance) {
        distance = maxSensorDistance + 1;
    } else { // d ≈ ∑_i (v_i+v_{i+1})/2 Δt
        distance += (velOld + velNew)/2;
    }
}
```

Listing 3: Heavily customized listing.

Table 1: Query text embedded into table.

Purpose	**Query**
Deadlock check	`A[] not deadlock || P.done`
Deadlock check	`A[] ! deadlock || P.done`
Find trajectory	`simulate 10 [<=100]{x}:1:P.done`
Find trajectory	`simulate 10 [<=100]{x, y, z}:1:P.done`

1.2 Installation Dependencies

The style package depends on the following LaTeX packages:

listings – to implement the actual listings typesetting.

xcolor – to define and customize colors.

xspace – to omit trailing space for tool names.

beramono – Bitstream Vera Mono fonts, the directory is usually called `bera` and the font files are prefixed with `fvm`.

The packages above are common and usually distributed by TeXLive and MikTex. For example, Debian GNU/Linux puts them into `texlive-latex-recommended`, `texlive-latex-recommended-doc` and `texlive-fonts-extra` packages.

Listing 4: LaTeX code for queries in a table.

```
\begin{tabular}{ll}
\toprule
{\bf Purpose} & {\bf Query} \\
\midrule
Deadlock check & \uppProp{A[] not deadlock || P.done} \\
Deadlock check & \uppAG{! deadlock || P.done} \\
Find trajectory & \uppProp{simulate 10 [<=100]\{x\}:1:P.done} \\
Find trajectory & \uppSimC{10}{<=100}{x, y, z}{1}{P.done} \\
\bottomrule
\end{tabular}
```

2 Package Options

None is implemented. It would be nice to turn on and off the coloring to gray scale printing, change the typewriter fonts as some proceedings demand using their own.

3 Environments

The `uppaal.sty` defines Uppaal language styles for `listings.sty` package, thus all the hard lifting is done by `listings.sty` and the details can be found in **listings** documentation.

lstlisting environment is the way to typeset any listing, and one needs to specify `language={[GUI]Uppaal}` parameter to turn on syntax highlighting similar to Uppaal editor.

uppaalcode environment is a shorthand to **lstlisting** environment with the language predefined to `[GUI]Uppaal`.

Here is the list of commonly used parameters used to customize the listing:

language the style package defines the following Uppaal code variants:

- `Uppaal` basic Uppaal keywords and font styles for them, only slanting and bold is used, no colors.
- `[GUI]Uppaal` the same font styles as above plus colors similar to Uppaal editor.
- `[LIT]Uppaal` the same features as above except some characters are replaced by mathematical notation.

caption specifies the text title for the listing.

captionpos specifies the placemant of the caption relative to the listing, possible values: `t` – at the top (default), and `b` – at the bottom.

label defines a label to refered to by the `\ref` command. `lst:` prefix can be useful to distinguish listings from figures and other floats.

float specifies that the listing must be treated as a float, i.e. the layout must into onto one page and it cannot be broken by a page break.

numbers adds a number column on the left or right, possible values: `none` (default), `left`, `right`.

numberstyle specifies the style for numbers, e.g. `\tiny`.

numbersep specifies the space size between numbers and listing.

xleftmargin specifies the size of the left margin.

frame draws some frame elements around the listing, possible values: `none` (default), `leftline`, `topline`, `bottomline`, `lines` (top and bottom), `single` (whole frame), `shadowbox` or a subset of `trblTRBL` characters where upper case denotes double lines.

frameround

There are are many many more options, please see the documentation for `listings.sty` package. It also plays nicely with `tcolorbox` package for fancy listings in `beamer` presentations.

4 Inline Code

Apart from explicit listings Uppaal code can also be typeset inside regular text flow using style similar to the default scheme in Uppaal graphical user interface. Table 2 shows a list of available commands.

5 Colors and Styles

Table 3 enumerates the colors and styles which can be customized.

Acknowledgements

I thank my colleagues for testing and suggesting: Brian Nielsen, Ulrik Nyman, Danny B. Poulsen, and others.

Table 2: List of supported LaTeX commands.

Description	**Example**	**Result**
Labels associated with **locations**:		
Location name	\uppLoc{Done}	Done
Invariant expression	\uppInv{x<=7 && y'==0}	x<=7 && y'==0
Exponential rate expr.	\uppRate{1:7}	1:7
Labels associated with **edges**:		
Select statement	\uppSelect{i:int[1,7]}	i:**int**[1,7]
Guard expression	\uppGuard{x>=3}	x>=3
Plain synchronization	\uppSync{message!}	message!
Input synchronization	\uppIn{message}	message?
Output synchronization	\uppOut{message}	message!
Update statement	\uppUpdate{x=5,y=7}	x=5,y=7
Weight expression	\uppWeight{i*10}	i*10
Variable and type names from **declarations**:		
Variable name	\uppVar{count}	count
Constant name	\uppConst{MAX_N}	MAX_N
Clock name	\uppClock{time}	time
Channel name	\uppChan{message}	message
Type name	\uppType{int32_t}	int32_t
Function name	\uppFunc{enqueue()}	*enqueue()*
Template name	\uppTemp{Train}	Train
Process name	\uppProc{Train(3)}	Train(3)
Properties and queries from **verifier**:		
Any property text	\uppProp{E<> deadlock}	E<> deadlock
Exists path with Future	\uppEF{deadlock}	E<> deadlock
Exists path with Global	\uppEG{counter<=9}	E[] counter<=9
All paths with Future	\uppAF{counter<=9}	A<> counter<=9
All paths with Global	\uppAG{counter<=9}	A[] counter<=9
Estimate probability	\uppPr{[<= 7](<> done)}	Pr[<=7](<> done)
Probability of Future	\uppPrF{<= 7}{done}	Pr[<=7](<> done)
Probability of Global	\uppPrG{<= 7}{good}	Pr[<=7]([] good)
Simulate and project	\uppSim{5}{<= 7}{x,y}	simulate 5 [<=7]{x,y}
Simulate with check	\uppSimC{5}{<= 7}{x}{3}{done}	simulate 5 [<=7]{x}:3:done
Tool names:		
Uppaal	\uppaal	Uppaal
Uppaal Tron	\uppaaltron	Uppaal Tron
Uppaal Tiga	\uppaaltiga	Uppaal Tiga
Uppaal SMC	\uppaalsmc	Uppaal SMC
Uppaal Stratego	\stratego	Uppaal Stratego

Table 3: Color definitions.

Color	Name	RGB	Sample
\uppCommentColor	darker red	0.4 , 0 , 0	
\uppKeywordColor	dark green	0 , 0.4 , 0	
\uppTypeColor	darker green	0 , 0.3 , 0	
\uppLocColor	dark red	0.5 , 0 , 0	
\uppInvColor	dark magenta	0.4 , 0 , 0.4	
\uppRateColor	pink	0.875, 0.25, 0.5	
\uppSelectColor	dark yellow	0.4 , 0.4 , 0	
\uppGuardColor	dark green	0 , 0.3 , 0	
\uppSyncColor	dark cyan	0 , 0.4 , 0.4	
\uppUpdateColor	dark blue	0 , 0 , 0.4	
\uppWeightColor	dark orange	0.4 , 0.2 , 0	

Table 4: Style definitions.

Style	Definition	Purpose	Sample
\uppBasicStyle	\uppPlainStyle	basic text	text
\uppKeywordStyle	\uppBfStyle	keywords	**return**
\uppTypeStyle	\uppBfStyle	type names	**clock**
\uppConstStyle	\uppBfStyle	constant names	**true**
\uppFuncStyle	\uppSlStyle	function names	*sin*
\uppCommentStyle	\uppSlStyle	simple comment	*comment*
\uppFCommentStyle	\uppSlBfStyle	fancy comment	***comment***
\uppPlainStyle	(plain font)	plain text	text
\uppBfStyle	\bfseries	bold text	**text**
\uppSlStyle	\slshape	slanted text	*text*
\uppSlBfStyle	\slshape\bfseries	slanted bold text	***text***