A Sample Document for the Usages of lstEventB Package

Thai Son Hoang
ECS, University of Southampton
<T dot S dot Hoang at ecs dot soton dot ac dot uk>

August 14, 2018

For convenient, we define macro `\EventB` for Event-B.

We start first with some inline Event-B code by embedding them using a pair of `|`, for example `|@grd1: "SNSR = FALSE"|` gives @grd1: "SNSR = FALSE". Any Event-B formulae including Unicode symbols will be typeset using the `bsymb` package accordingly.

ASCII	Symbols	Explanation
`!:`	$\in$	Set membership
`/:`	$\notin$	Set non-membership
`<:`	$\subseteq$	Subset
`/<:`	$\not\subseteq$	Not a subset
`<<:`	$\subset$	Proper subset
`/<<:`	$\not\subset$	Not a proper subset
`!finite`	finite	Finite
`!partition`	partition	Partition

Table 1: Set predicates

ASCII	Symbols	Explanation
`!BOOL`	BOOL	BOOL set
`!TRUE`	TRUE	TRUE
`!FALSE`	FALSE	FALSE
`!bool`	bool	bool predicate

Table 2: BOOL and bool

More complete piece of code (including the Unicode symbols) can be typeset using the `EventBcode` environment. Below is the typesetting of an Event-B machine.

```
1 machine Sensor_m0_SNSR
```

ASCII	Symbols	Explanation
!INT	$\mathbb{Z}$	Set of integer numbers
!NAT	$\mathbb{N}$	Set of natural numbers
!NAT1	$\mathbb{N}_1$	Set of positive natural numbers
!min	min	Mininum
!max	max	Maximum
-	$-$	Difference
*	$*$	Product
!/	$\div$	Quotient
!mod	mod	Remainder
..	..	Interval

Table 3: Numbers

ASCII	Symbols	Explanation
>	$>$	Greater
<	$<$	Less
>=	$\geq$	Greater or equal
<=	$\leq$	Less or equal

Table 4: Number predicates

ASCII	Symbols	Explanation
<->	$\leftrightarrow$	Relations
!dom	dom	Domain
!ran	ran	Range
<<->	$\leftarrow\!\!\!\leftrightarrow$	Total relations
<->>	$\leftrightarrow\!\!\!\rightarrow$	Surjective relations
<<->>	$\leftarrow\!\!\!\leftrightarrow\!\!\!\rightarrow$	Total surjective relations
!circ	$\circ$	Backward composition
!id	id	Identity
<|	$\lhd$	Domain restriction
<<|	$\lhd\!\!\!-$	Domain subtraction
|>	$\rhd$	Range restriction
|>>	$\rhd\!\!\!-$	Range subtraction
~	$^{-1}$	Inverse
<+	$\lhd\!\!\!+$	Overriding
><	$\otimes$	Direct product
||	$\parallel$	Parallel product
!prj1	prj_1	First projection
!prj2	prj_2	Second projection

Table 5: Relations

ASCII	Symbols	Explanation
+->	$\pfun$	Partial functions
-->	$\fun$	Total functions
>+>	$\pinj$	Partial injections
>->	$\inj$	Total injections
+>>	$\psur$	Partial surjections
->>	$\sur$	Total surjections
>->>	$\bij$	Bijections
%	λ	Lambda abstraction

Table 6: Functions

ASCII	Symbols	Explanation
:=	$:=$	Becomes equal to
::	$:\in$	Choice from a set
:|	$:\mid$	Choice by predicate

Table 7: Functions

```
variables
 SNSR
invariants
 @thm0_1: "SNSR ∈ BOOL" theorem
events

 INITIALISATION
 begin
  @act1: "SNSR := FALSE"
 end

 SNSR_on
 when
  @grd1: "SNSR = FALSE"
 then
  @act1: "SNSR := TRUE"
 end

 SNSR_off
 when
  @grd1: "SNSR = TRUE"
 then
  @act1: "SNSR := FALSE"
 end

end
```

One can change the different colour options. For example, `\EventBSetKeywordColour{blue!50!black}` will change the keyword colour to dark blue. (This has effects only when

---

```
machine Sensor_m0_SNSR
variables
 SNSR
```

```
4 invariants
5  @thm0_1: "SNSR ∈ BOOL" theorem
```

One can includes external file containing Event-B code using the `\EventBinputlisting` command. For example the following is the result of including the code in the file `Sensor_m1_DEP.bumx` using `\EventBinputlisting{Sensor_m1_DEP.bumx}`.

```
1  machine Sensor_m1_DEP
2  refines Sensor_m0_SNSR
3  variables
4   SNSR
5   DEP
6  invariants
7   @inv0_1: "DEP ∈ ℕ"
8  events
9
10  INITIALISATION extended
11  begin
12   @act2: "DEP := 0"
13  end
14
15  SNSR_on extended
16  refines SNSR_on
17  end
18
19  SNSR_off extended
20  refines SNSR_off
21  begin
22   @act2: "DEP := DEP + 1"
23  end
24
25 end
```

More specifically, one can specify more details on the inclusion, e.g., the ranges, as the following example
`\EventBinputlisting[firstline=16,lastline=20]{Sensor_m2_snsr.bumx}`
gives

```
1  @inv1_1: "Snsr_01 = TRUE ⇒ SNSR = TRUE"
2
3  @inv1_2: "Snsr_10 = TRUE ⇒ SNSR = FALSE"
4
5  @inv1_3: "Snsr_01 = FALSE ∨ Snsr_10 = FALSE"
```

```
1 machine Sensor_m3_Ctrl
2
3 refines
4
5  Sensor_m2_Snsr
6
```

```
variables

 SNSR

 DEP

 Snsr_01

 Snsr_10

 ctrl_snsr

 ctrl_dep

 ctrl_snsr_01

 ctrl_snsr_10

invariants

 @inv2_1:
 "Snsr_01 = FALSE ∧ Snsr_10 = FALSE ∧ ctrl_snsr_01 = FALSE ∧ ctrl_snsr_10 = FALSE
      ⇒ ctrl_snsr = SNSR"

 @inv2_2: "ctrl_dep ∈ ℕ"

 @inv2_3: "Snsr_10 = FALSE ∧ ctrl_snsr_10 = FALSE ⇒ ctrl_dep = DEP"

 @inv2_4: "Snsr_10 = TRUE ∨ ctrl_snsr_10 = TRUE ⇒ ctrl_dep = DEP − 1"

 @inv2_5: "ctrl_snsr_01 = TRUE ⇒ SNSR = TRUE"

 @inv2_6: "ctrl_snsr_10 = TRUE ⇒ SNSR = FALSE"

 @inv2_7: "ctrl_snsr_01 = TRUE ⇒ Snsr_01 = FALSE"

 @inv2_8: "ctrl_snsr_10 = TRUE ⇒ Snsr_10 = FALSE"

events

 INITIALISATION extended
 refines INITIALISATION
 begin
  @act5: "ctrl_snsr := FALSE"
  @act6: "ctrl_dep := 0"
  @act7: "ctrl_snsr_01 := FALSE"
  @act8: "ctrl_snsr_10 := FALSE"
 end

 SNSR_on extended
 refines SNSR_on
 when
  @grd3: "ctrl_snsr_10 = FALSE"
 end

 SNSR_off extended
```

```
62  refines SNSR_off
63  when
64   @grd3: "ctrl_snsr_01 = FALSE"
65  end
66
67  ctrl_Senses_Snsr_01 extended
68  refines ctrl_Senses_Snsr_01
69  begin
70   @act2: "ctrl_snsr_01 := TRUE"
71  end
72
73  ctrl_Senses_Snsr_10 extended
74  refines ctrl_Senses_Snsr_10
75  begin
76   @act2: "ctrl_snsr_10 := TRUE"
77  end
78
79  ctrl_on
80  when
81   @grd1: "ctrl_snsr_01 = TRUE"
82  then
83   @act1: "ctrl_snsr_01 := FALSE"
84   @act2: "ctrl_snsr := TRUE"
85  end
86
87  ctrl_off
88  when
89   @grd1: "ctrl_snsr_10 = TRUE"
90  then
91   @act1: "ctrl_snsr_10 := FALSE"
92   @act2: "ctrl_snsr := FALSE"
93   @act3: "ctrl_dep := ctrl_dep + 1"
94  end
95
96 end
```