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>

March 29, 2021

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.

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
`<<->`	$\nleftrightarrow$	Total relations
`<->>`	$\twoheadleftarrow$	Surjective relations
`<<->>`	$\twoheadleftarrow\!\!\!\!\!\rightarrow$	Total surjective relations
`!circ`	$\circ$	Backward composition
`!id`	id	Identity
`<|`	$\lhd$	Domain restriction
`<<|`	$\not\lhd$	Domain subtraction
`|>`	$\rhd$	Range restriction
`|>>`	$\not\rhd$	Range subtraction
`~`	$\sim$	Inverse
`<+`	$\leftarrow\!\!\!\!+$	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
`->>`	$\surj$	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

```
machine Sensor_m0_SNSR
variables
 SNSR
invariants
 theorem @thm0_1: SNSR ∈ BOOL
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
```

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

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 extends SNSR_on
16  end
17
18  SNSR_off extends SNSR_off
19  begin
20   @act2: DEP := DEP + 1
21  end
22
23 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
7  variables
```

```
8
9   SNSR
10
11  DEP
12
13  Snsr_01
14
15  Snsr_10
16
17  ctrl_snsr
18
19  ctrl_dep
20
21  ctrl_snsr_01
22
23  ctrl_snsr_10
24
25 invariants
26
27  @inv2_1:
28  Snsr_01 = FALSE ∧ Snsr_10 = FALSE ∧ ctrl_snsr_01 = FALSE ∧ ctrl_snsr_10 = FALSE ⇒
        ctrl_snsr = SNSR
29
30  @inv2_2: ctrl_dep ∈ ℕ
31
32  @inv2_3: Snsr_10 = FALSE ∧ ctrl_snsr_10 = FALSE ⇒ ctrl_dep = DEP
33
34  @inv2_4: Snsr_10 = TRUE ∨ ctrl_snsr_10 = TRUE ⇒ ctrl_dep = DEP − 1
35
36  @inv2_5: ctrl_snsr_01 = TRUE ⇒ SNSR = TRUE
37
38  @inv2_6: ctrl_snsr_10 = TRUE ⇒ SNSR = FALSE
39
40  @inv2_7: ctrl_snsr_01 = TRUE ⇒ Snsr_01 = FALSE
41
42  @inv2_8: ctrl_snsr_10 = TRUE ⇒ Snsr_10 = FALSE
43
44 events
45
46  INITIALISATION extends INITIALISATION
47  begin
48   @act5: ctrl_snsr := FALSE
49   @act6: ctrl_dep := 0
50   @act7: ctrl_snsr_01 := FALSE
51   @act8: ctrl_snsr_10 := FALSE
52  end
53
54  SNSR_on extends SNSR_on
55  when
56   @grd3: ctrl_snsr_10 = FALSE
57  end
58
59  SNSR_off extends SNSR_off
60  when
61   @grd3: ctrl_snsr_01 = FALSE
62  end
```

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