

The `eventB` package*

Thai Son Hoang
ETH-Zurich
<htson at inf dot ethz dot ch>

January 15, 2014

Abstract

This class provides a template for typesetting Event-B models. It was developed at the Swiss Federal Institute of Technology Zurich (ETH-Zurich).

Contents

1	Introduction	1
2	Usage	1
2.1	Package Options	2
3	Implementation	2
3.1	Package Loading	2
3.2	Internal Helper Macros	2
3.3	Declaration of Options for the Package	6
3.3.1	Option for bounding boxes	7
3.3.2	Options for font size and spacing	8
3.3.3	Options for colouring	9
3.4	Meta-macros for creating macros for modelling elements	11
3.5	Commands for Pretty-Print Event-B Models	11

1 Introduction

This package was developed in order to ease the typesetting of Event-B models in \LaTeX .

2 Usage

See `sample-eventB.tex` for an example of how to use the package.

*This document corresponds to `eventB` v2.0, dated 2014/01/14.

2.1 Package Options

The package offers the following options:

- `nobox`: to disable to bounding boxes for the Event-B modelling elements,
- `small`, `compact`, `tiny`: options for font size,
- `colour` (or `color`): to colour several modelling elements.

3 Implementation

3.1 Package Loading

We begin by loading the required package `xspace`, `xcolor`, and `etoolbox`.

```
1 \RequirePackage{xspace}
2 \RequirePackage{xcolor}
3 \RequirePackage{etoolbox}
```

3.2 Internal Helper Macros

We define same basic internal helper macros that will be used to defined other macros.

<code>\B@ifstrequal</code>	A utility wrapper for ake sure that the first argument is properly expanded. 4 <code>\newcommand{\B@ifstrequal}{\expandafter\ifstrequal\expandafter}</code>
<code>\B@keywordbase</code>	Basic macro for Event-B keywords. 5 <code>\newcommand{\B@keywordbase}[1]{\mathbf{#1}}</code>
<code>\B@identifierbase</code>	Basic macro for Event-B identifiers. 6 <code>\newcommand{\B@identifierbase}[1]{\mathit{#1}}</code>
<code>\B@labelbase</code>	Basic macro for Event-B labels. 7 <code>\newcommand{\B@labelbase}[2][]{\%</code> 8 <code>\ifstrequal{#1}{\}%</code> 9 <code>\mathsf{#2}</code> 10 <code>}{\%</code> 11 <code>\mathit{#2}</code> 12 <code>}</code> 13 <code>}</code>
<code>\B@constructbase</code>	Basic macro for Event-B constructs. 14 <code>\newcommand{\B@constructbase}[1]{\mathsf{#1}}</code>
<code>\B@pobase</code>	Basic macro for Event-B proof obligations. 15 <code>\newcommand{\B@pobase}[1]{\mathsf{#1}}</code>
<code>\B@declarationbase</code>	Basic macro for Event-B declarations (e.g., constants, variables, etc.). 16 <code>\newcommand{\B@declarationbase}[2]{\%</code> 17 <code>\begin{array}{@{}l@{\B@tab}l@{}}</code> 18 <code>\B@keyword{#1:} & #2</code> 19 <code>\end{array}</code> 20 <code>}</code>

`\B@sectionbase` Basic macro for Event-B sections (e.g., axioms, invariants, etc.). The optional argument is to produce the keywords or not

```

21 \newcommand{\B@sectionbase}[3] [] {%
22   \ifstrequal{#1}{%{
23     \begin{array}{@{}l@{}}
24       \B@keyword{#2:} \\
25       \begin{array}{@{\B@tab}l@{\B@tab}l@{\B@tab}}
26         #3
27       \end{array}
28     \end{array}
29   }{
30     \begin{array}{@{\B@tab}l@{\B@tab}l@{\B@tab}}
31       #3
32     \end{array}
33   }
34 }

```

`\B@eventbase` Basic macro for pretty-print Event-B events.

```

35 \newcommand{\B@eventbase}[7] [] {%
36   { % BEGIN group

```

We first save the arguments to local variables.

```

37   \newcommand\evt@sts{#1}% Event status
38   \newcommand\evt@label{#2}% Event label
39   \newcommand\evt@absevt{#3}% Abstract event
40   \newcommand\evt@pars{#4}% Event parameters
41   \newcommand\evt@grds{#5}% Event guards
42   \newcommand\evt@wits{#6}% Event witnesses
43   \newcommand\evt@acts{#7}% Event actions

```

The convergence status is skipped if empty.

```

44   \B@ifstrequal{\evt@sts}{%{
45     \newcommand\pretty@sts{}
46   }{
47     \newcommand\pretty@sts{\B@tab\Bstatus \B@tab \evt@sts \\\}
48   }

```

The refines clause is skipped if there are no abstract events.

```

49   \B@ifstrequal{\evt@absevt}{%{
50     \newcommand\pretty@absevt{}
51   }{
52     \newcommand\pretty@absevt{\B@tab \Brefines \B@tab \evt@absevt{} \\\}
53   }

```

The parameters is skipped if there are none.

```

54   \B@ifstrequal{\evt@pars}{%{
55     \newcommand\pretty@pars{}
56   }{
57     \newcommand\pretty@pars{\B@tab \Bany \B@tab \evt@pars \B@tab \Bwhere \\\}
58   }

```

The keywords for guards also depends on if there are parameters or not.

```

59   \B@ifstrequal{\evt@grds}{%{
60     \newcommand\pretty@grds{}
61   }{
62     \newcommand\pretty@grds@tmp{

```

```

63     \begin{array}{@{\B@tab\B@tab}l@{\B@tab}l}
64     \evt@grds
65     \end{array}\\
66   }
67   \B@ifstrequal{\evt@pars}{}{
68     \newcommand\pretty@grds{
69       \B@tab \Bwhen \\
70       \pretty@grds@tmp
71     }
72   }{
73     \newcommand\pretty@grds{\pretty@grds@tmp}
74   }
75 }

```

The witnesses are skipped if there are none.

```

76   \B@ifstrequal{\evt@wits}{}{
77     \newcommand\pretty@wits{
78   }{
79     \newcommand\pretty@wits{
80       \B@tab\Bwith\\
81       \begin{array}{@{\B@tab\B@tab}l}
82         \evt@wits
83       \end{array}\\
84     }
85   }

```

When there are no actions, SKIP is used. The keyword is changed depending on whether the event has parameters or not.

```

86   \B@ifstrequal{\evt@acts}{}{
87     \renewcommand\evt@acts{\SKIP}
88   }{}
89   \newcommand\pretty@acts@tmp{
90     \begin{array}{@{\B@tab\B@tab}l@{\B@tab}l}
91       \evt@acts
92     \end{array}\\
93   }
94   \newcommand\pretty@acts@keyword{\B@tab\Bthen \\}
95   \B@ifstrequal{\evt@pars}{}{
96     \B@ifstrequal{\evt@grds}{}{
97       \renewcommand\pretty@acts@keyword{\B@tab\Bbegin \\}
98     }{}
99   }{}
100  \newcommand\pretty@acts{
101    \pretty@acts@keyword
102    \pretty@acts@tmp
103  }

```

Finally we put all the pretty-print pieces together.

```

104  \begin{array}{@{}l@{}}
105    \Bevt{\evt@label} \\
106    \pretty@sts
107    \pretty@absepts
108    \pretty@pars
109    \pretty@grds
110    \pretty@wits

```

```

111     \pretty@acts
112     \B@tab\Bend
113   \end{array}
114 } % END group
115 }

```

`\B@inlineeventbase` Basic macro for pretty-print Event-B events inline.

```

116 \newcommand{\B@inlineeventbase}[7] [] {
117   { % BEGIN group
118     We first save the arguments to local variables.
119     \newcommand\evt@sts{#1}% Event status
120     \newcommand\evt@label{#2}% Event label
121     \newcommand\evt@absevt{#3}% Abstract event
122     \newcommand\evt@pars{#4}% Event parameters
123     \newcommand\evt@grds{#5}% Event guards
124     \newcommand\evt@wits{#6}% Event witnesses
125     \newcommand\evt@acts{#7}% Event actions
126
127     The convergence status is skipped if empty.
128     \B@ifstrequal{\evt@sts}{}{
129       \newcommand\pretty@sts{}
130     }{
131       \newcommand\pretty@sts{(\evt@sts)}
132     }
133
134     The refines clause is skipped if there are no abstract events.
135     \B@ifstrequal{\evt@absevt}{}{
136       \newcommand\pretty@absevt{}
137     }{
138       \newcommand\pretty@absevt{~\Brefines~\evt@absevt}
139     }
140
141     The parameters is skipped if there are none.
142     \B@ifstrequal{\evt@pars}{}{
143       \newcommand\pretty@pars{}
144     }{
145       \newcommand\pretty@pars{\Bany~\evt@pars~\Bwhere~}
146     }
147
148     The keywords for guards also depends on if there are parameters or not.
149     \B@ifstrequal{\evt@grds}{}{
150       \newcommand\pretty@grds{}
151     }{
152       \newcommand\pretty@grds@tmp{
153         \evt@grds~
154       }
155       \B@ifstrequal{\evt@pars}{}{
156         \newcommand\pretty@grds{\Bwhen~\pretty@grds@tmp}
157       }{
158         \newcommand\pretty@grds{\pretty@grds@tmp}
159       }
160     }
161
162     The witnesses are skipped if there are none.
163     \B@ifstrequal{\evt@wits}{}{

```

```

153     \newcommand\pretty@wits{}
154   }{
155     \newcommand\pretty@wits{
156       \Bwith~
157       \evt@wits~
158     }
159   }

```

When there are no actions, SKIP is used. The keyword is changed depending on whether the event has parameters or not.

```

160   \B@ifstrequal{\evt@acts}{}{
161     \renewcommand\evt@acts{\SKIP}
162   }{}
163   \newcommand\pretty@acts@tmp{
164     \evt@acts
165   }
166   \newcommand\pretty@acts@keyword{\Bthen}
167   \B@ifstrequal{\evt@pars}{}{
168     \B@ifstrequal{\evt@grds}{}{
169       \renewcommand\pretty@acts@keyword{\Bbegin}
170     }{}
171   }{}
172   \newcommand\pretty@acts{
173     \pretty@acts@keyword~
174     \pretty@acts@tmp~
175   }

```

Finally we put all the pretty-print pieces together.

```

176   \Bevt{\evt@label}\pretty@sts\pretty@absepts~\widehat{=}~
177   \pretty@pars
178   \pretty@grds
179   \pretty@wits
180   \pretty@acts
181   \Bend
182 } % END group
183 }

```

\B@makebox A wrapper macro to make a fbox with the boundary adjusted.

```

184 \newlength{\B@tmp@length}
185 \newcommand{\B@makebox}[1]{
186   {
187     \setlength{\B@tmp@length}{\fboxsep}
188     \setlength{\fboxsep}{2ex}
189     \fbox{#1}
190     \setlength{\fboxsep}{\B@tmp@length}
191   }
192 }

```

3.3 Declaration of Options for the Package

In this part various options for the package are defined.

3.3.1 Option for bounding boxes

By default, Event-B modelling elements, e.g., invariants, events, etc., are displayed in a bounding box. This `nobox` option enables them to be displayed without the bounding box.

`\B@event` Default definition displays Event-B events in a box.

```
193 \newcommand{\B@event}[7] [] {
194   \B@makebox{
195     \ensuremath{
196       \B@eventbase[#1]{#2}{#3}{#4}{#5}{#6}{#7}
197     }
198   }
199 }
```

`\B@declaration` Default definition displays Event-B declarations in a box.

```
200 \newcommand{\B@declaration}[2] {
201   \B@makebox{%
202     \ensuremath{%
203       \B@declarationbase[#1]{#2}
204     }
205   }
206 }
```

`\B@section` Default definition displays Event-B sections in a box

```
207 \newcommand{\B@section}[3] [] {
208   \B@makebox{%
209     \ensuremath{%
210       \B@sectionbase[#1]{#2}{#3}
211     }
212   }
213 }
```

Option “nobox” The above commands are redefined accordingly when option `nobox` is enabled.

```
214 \DeclareOption{nobox}{
```

`\B@event` Redefine the definition without the bounding box.

```
215   \renewcommand{\B@event}[7] [] {%
216     \B@eventbase[#1]{#2}{#3}{#4}{#5}{#6}{#7}
217   }
```

`\B@declaration` Redefine the definition without the bounding box.

```
218   \renewcommand{\B@declaration}[2] {
219     \B@declarationbase[#1]{#2}
220   }
```

`\B@section` Redefine the definition without the bounding box.

```
221   \renewcommand{\B@section}[3] [] {
222     \B@sectionbase[#1]{#2}{#3}
223   }
224 }
```

3.3.2 Options for font size and spacing

We define the default values for font size and some spacing commands, and how they are redefined according to options `small`, `compact`, and `tiny`. In particular, option `compact` and `tiny` implies option `nobox`.

`\B@fontsize` The font size used in the `Bcode` environment (defined later).

```
225 \newcommand{\B@fontsize}{\normalsize}
```

`\B@vspace` A vertical rule for spacing, defaulted to be `2ex`.

```
226 \newcommand{\B@vspace}[1][2ex]{\[#1]}
```

`\B@hspace` A horizontal rule for spacing, defaulted to be `2em`.

```
227 \newcommand{\B@hspace}[1][2em]{\hspace{#1}}
```

`\B@tab` A small tab for spacing, defaulted to be `\quad`.

```
228 \newcommand{\B@tab}{\quad} % A small separation space
```

We subsequently redefined the above spacing commands when one of the options `small`, `compact`, `tiny` is enabled.

Option “small” For option `small` they are adjusted as follows.

```
229 \DeclareOption{small}{
```

`\B@fontsize` Redefine to be `\small` for option `small`.

```
230 \renewcommand{\B@fontsize}{\small}
```

`\B@vspace` Redefine to be `1ex` for option `small`.

```
231 \renewcommand{\B@vspace}[1][1ex]{\[#1]}
```

`\B@hspace` Redefine to be `1em` for option `small`.

```
232 \renewcommand{\B@hspace}[1][1em]{\hspace{#1}}
```

`\B@tab` Redefine to be `\` for option `small`.

```
233 \renewcommand{\B@tab}{\ }
```

```
234 }
```

Option “compact” For option `compact` the commands are adjusted as follows.

```
235 \DeclareOption{compact}{
```

`\B@fontsize` Redefine to be `\footnotesize` for option `compact`.

```
236 \renewcommand{\B@fontsize}{\footnotesize}
```

`\B@vspace` Redefine to be `0ex` for option `compact`.

```
237 \renewcommand{\B@vspace}[1][0ex]{\[#1]}
```

`\B@hspace` Redefine to be `0.5em` for option `compact`.

```
238 \renewcommand{\B@hspace}[1][0.5em]{\hspace{#1}}
```

`\B@tab` Redefine to be `\` for option `compact`.

```
239 \renewcommand{\B@tab}{\ }
```

Option `nobox` is enabled.

```
240 \ExecuteOptions{nobox}
```

```
241 }
```


Option “tiny” For option `tiny` the commands are adjusted as follows.

```

242 \DeclareOption{tiny}{
\B@fontsize  Redefine to be \scriptsize for option tiny.
243   \renewcommand{\B@fontsize}{\scriptsize}

\B@vspace    Redefine to be -0.5ex for option tiny.
244   \renewcommand{\B@vspace}[1][-0.5ex]{\{ \{ #1 \}}

\B@hspace    Redefine to be 0.5em for option compact.
245   \renewcommand{\B@hspace}[1][0.5em]{\hspace{#1}}

\B@tab       Redefine to be \ for option compact.
246   \renewcommand{\B@tab}{\ }

Option nobox is enabled.
247   \ExecuteOptions{nobox}
248 }
```

3.3.3 Options for colouring

Keywords, labels and identifiers in Event-B can be coloured. We define several commands and redefine them accordingly for colouring. When `colour` (or `color`) option is enabled, one can customise the colours for Event-B keywords, labels or identifier or proof obligation labels. We proceed with some definitions that can be redefined by these options.

```

\B@keyword    Macro for Event-B keywords.
249 \newcommand{\B@keyword}[1]{\ensuremath{\B@keywordbase{#1}}\xspace}

\B@identifier Macro for Event-B identifiers.
250 \newcommand{\B@identifier}[1]{\ensuremath{\B@identifierbase{#1}}\xspace}

\B@label      Macro for Event-B labels.
251 \newcommand{\B@label}[2][ ]{\ensuremath{\B@labelbase{#1}{#2}}\xspace}

\B@construct  Macro for Event-B constructs.
252 \newcommand{\B@construct}[1]{\ensuremath{\B@constructbase{#1}}\xspace}

\B@po         Macro for Event-B proof obligations.
253 \newcommand{\B@po}[1]{\ensuremath{\B@pobase{#1}}\xspace}
```

We redefine the above commands if option `colour` or `color` is enabled. Furthermore, we define some commands for setting colour for various modelling elements.

Option 'colour' The option colour is declared as follows.

```
254 \DeclareOption{colour}{
```

\setBKeywordColour Utility macro for defining Event-B keywords colour.

```
255 \newcommand{\setBKeywordColour}[1]{\colorlet{B@keywordcolor}{#1}}
256 \setBKeywordColour{blue}
```

\setBIdentifierColour Utility macro for defining Event-B identifiers colour.

```
257 \newcommand{\setBIdentifierColour}[1]{\colorlet{B@identifiercolor}{#1}}
258 \setBIdentifierColour{blue!50!red}
```

\setBLabelColour Utility macro for defining Event-B labels colour.

```
259 \newcommand{\setBLabelColour}[1]{\colorlet{B@labelcolor}{#1}}
260 \setBLabelColour{green!50!black}
```

\setBConstructColour Utility macro for defining Event-B constructs colour.

```
261 \newcommand{\setBConstructColour}[1]{\colorlet{B@constructcolor}{#1}}
262 \setBConstructColour{orange!75!black}
```

\setBP0Colour Utility macro for defining Event-B proof obligations colour.

```
263 \newcommand{\setBP0Colour}[1]{\colorlet{B@pocolor}{#1}}
264 \setBP0Colour{red}
```

We redefine the commands with colours.

```
265 \renewcommand{\B@keyword}[1]{
266   \ensuremath{\textcolor{B@keywordcolor}{\B@keywordbase{#1}}}\xspace
267 }
268 \renewcommand{\B@identifier}[1]{
269   \ensuremath{\textcolor{B@identifiercolor}{\B@identifierbase{#1}}}\xspace
270 }
271 \renewcommand{\B@label}[2][ ]{
272   \ensuremath{\textcolor{B@labelcolor}{\B@labelbase{#1}{#2}}}\xspace
273 }
274 \renewcommand{\B@construct}[1]{
275   \ensuremath{\textcolor{B@constructcolor}{\B@constructbase{#1}}}\xspace
276 }
277 \renewcommand{\B@po}[1]{
278   \ensuremath{\textcolor{B@pocolor}{\B@pobase{#1}}}\xspace
279 }
280 }
```

Option 'color' This option is a pointer to colour.

```
281 \DeclareOption{color}{
282   \ExecuteOptions{colour}
283 }
```

After declaration of options, we execute them accordingly.

```
284 \ProcessOptions
```

3.4 Meta-macros for creating macros for modelling elements

We define meta-macros to create macros for different modelling elements.

```

285 \newcommand{\B@newmacro}[3][]{
286   \ifstrequal{#1}{}{
287     \expandafter\def\csname #2\endcsname{#3{#2}}
288   }{
289     \expandafter\def\csname #1\endcsname{#3{#2}}
290   }
291 }

292 \newcommand{\newBctx}[2][]{\B@newmacro[#1]{#2}{\Bctx}}
293 \newcommand{\newBset}[2][]{\B@newmacro[#1]{#2}{\Bset}}
294 \newcommand{\newBcst}[2][]{\B@newmacro[#1]{#2}{\Bcst}}
295 \newcommand{\newBaxm}[2][]{\B@newmacro[#1]{#2}{\Baxm}}
296 \newcommand{\newBthm}[2][]{\B@newmacro[#1]{#2}{\Bthm}}
297 \newcommand{\newBmch}[2][]{\B@newmacro[#1]{#2}{\Bmch}}
298 \newcommand{\newBvrb}[2][]{\B@newmacro[#1]{#2}{\Bvrb}}
299 \newcommand{\newBinvt}[2][]{\B@newmacro[#1]{#2}{\Binvt}}
300 \newcommand{\newBevt}[2][]{\B@newmacro[#1]{#2}{\Bevt}}
301 \newcommand{\newBpar}[2][]{\B@newmacro[#1]{#2}{\Bpar}}
302 \newcommand{\newBgrd}[2][]{\B@newmacro[#1]{#2}{\Bgrd}}
303 \newcommand{\newBact}[2][]{\B@newmacro[#1]{#2}{\Bact}}
304 \newcommand{\newBkeyword}[2][]{\B@newmacro[#1]{#2}{\Bkeyword}}

```

3.5 Commands for Pretty-Print Event-B Models

We start with the definition of the `\eventB` macro.

```

305 \newcommand{\eventB}{Event-B\xspace}

```

The `Bcode` environment for displaying Event-B models. The environment has an optional argument for specifying the font size. By default, it is the same as the `\B@fontsize` controlled by the package option.

```

306 \newenvironment{Bcode}[1][\B@fontsize]{\begin{center}#1{\end{center}}

```

Declarations and Collections Event-B modelling elements are organised into declarations (e.g., variables, constants, etc.) or collections (e.g., invariants, axioms). For each declaration, the input is a comma-separated list of elements. For each collection, the input is a newly(`\`)-separated list of elements.

```

307 \newcommand{\carriersets}[1]{%
308   \B@declaration{sets}{#1}
309 }

310 \newcommand{\constants}[1]{%
311   \B@declaration{constants}{#1}
312 }

```

```

313 \newcommand{\axioms}[2][]{\%
314   \B@section[#1]{axioms}{#2}
315 }

316 \newcommand{\variables}[1]{
317   \B@declaration{variables}{#1}
318 }

319 \newcommand{\invariants}[2][]{
320   \B@section[#1]{invariants}{#2}
321 }

322 \newcommand{\variant}[1]{
323   \B@declaration{variant}{#1}
324 }

```

Event-B keywords We define the keywords for pretty-print Event-B models.

```

325 \newBkeyword[Bany]{any}
326 \newBkeyword[Bbegin]{begin}
327 \newBkeyword[Brefines]{refines}
328 \newBkeyword[Bstatus]{status}
329 \newBkeyword[Bthen]{then}
330 \newBkeyword[Bwhen]{when}
331 \newBkeyword[Bwhere]{where}
332 \newBkeyword[Bwith]{with}
333 \newBkeyword[Bend]{end}

```

Event-B modelling elements We define several macros for pretty-print Event-B modelling elements.

```

334 \newcommand{\Bctx}[1]{\B@construct{#1}}
335 \newcommand{\Bset}[1]{\B@identifier{#1}}
336 \newcommand{\Bcst}[1]{\B@identifier{#1}}
337 \newcommand{\Baxm}[1]{\B@label{#1}}
338 \newcommand{\Bthm}[1]{\B@label[thm]{#1}}
339
340 \newcommand{\Bmch}[1]{\B@construct{#1}}
341 \newcommand{\Bvrb}[1]{\B@identifier{#1}}
342 \newcommand{\Binv}[1]{\B@label{#1}}
343 \newcommand{\Bevt}[1]{\B@label{#1}}
344 \newcommand{\Bpar}[1]{\B@identifier{#1}}
345 \newcommand{\Bact}[1]{\B@label{#1}}
346 \newcommand{\Bgrd}[1]{\B@label{#1}}
347 \newcommand{\Bbap}[1]{\hbox{\sl\bfseries #1}}
348
349 %%%% Theorem Proof Obligation
350 %%%% Print the theorem proof obligation, given the theorem label.
351 %%%% Arguments:
352 %%%% 1. Theorem label
353 %%%%
354 %%%% Usage:
355 %%%% - \thmpo{thm} will produce "thm/THM"
356 \newcommand{\thmpo}[1]{\Bthm{#1}/\B@po{THM}}
357

```

```

358 %%%% Axiom Well-definedness Proof Obligation
359 %%%% Print the axiom well-definedness proof obligation, given the
360 %%%% axiom label.
361 %%%% Arguments:
362 %%%% 1. Axiom label
363 %%%%
364 %%%% Usage:
365 %%%% - \axmwdpo{axm} will produce "axm/WD"
366 \newcommand{\axmwdpo}[1]{\Baxm{#1}/\B@po{WD}}
367
368 %%%% Invariant Proof Obligation
369 %%%% Print the invariant proof obligation, given the event name and
370 %%%% invariant label
371 %%%% Arguments:
372 %%%% 1. Event name
373 %%%% 2. Invariant label
374 %%%%
375 %%%% Usage:
376 %%%% - \invpo{evt}{inv} will produce "evt/inv/INV"
377 \newcommand{\invpo}[2]{\Bevt{#1}/\Binv{#2}/\B@po{INV}}
378
379 %%%% Theorem (in guard) Proof Obligation
380 %%%% Print the simulation proof obligation, given the event name and
381 %%%% the theorem (in guard) label.
382 %%%% Arguments:
383 %%%% 1. Event name
384 %%%% 2. Theorem (in guard) label
385 %%%%
386 %%%% Usage:
387 %%%% - \grdthmpo{evt}{thm} will produce "evt/thm/THM"
388 \newcommand{\grdthmpo}[2]{\Bevt{#1}/\Bthm{#2}/\B@po{THM}}
389
390 %%%% Feasibility Proof Obligation
391 %%%% Print the feasibility proof obligation, given the event name and
392 %%%% the action label
393 %%%% Arguments:
394 %%%% 1. Event name
395 %%%% 2. Action label
396 %%%%
397 %%%% Usage:
398 %%%% - \fispo{evt}{act} will produce "evt/act/FIS"
399 \newcommand{\fispo}[2]{\Bevt{#1}/\Bact{#2}/\B@po{FIS}}
400
401 %%%% Variant finiteness Proof Obligation
402 %%%% Print the Variant finiteness proof obligation
403 %%%% Arguments: No arguments
404 %%%%
405 %%%% Usage:
406 %%%% - \finpo will produce "FIN"
407 \newcommand{\finpo}{\B@po{FIN}}
408
409 %%%% Variant Proof Obligation

```

```

410 %%%% Print the guard strengthen proof obligation, given the event name
411 %%%% Arguments:
412 %%%% 1. Event name
413 %%%%
414 %%%% Usage:
415 %%%% - \grdpo{evt} will produce "evt/VAR"
416 \newcommand{\varpo}[1]{\Bevt{#1}/\B@po{VAR}}
417
418 %%%% Simulation Proof Obligation
419 %%%% Print the simulation proof obligation, given the event name and
420 %%%% the action label.
421 %%%% Arguments:
422 %%%% 1. Event name
423 %%%% 2. Action label
424 %%%%
425 %%%% Usage:
426 %%%% - \simpo{evt}{act} will produce "evt/act/SIM"
427 \newcommand{\simpo}[2]{\Bevt{#1}/\Bact{#2}/\B@po{SIM}}
428
429 %%%% Guard Strengthen Proof Obligation
430 %%%% Print the guard strengthen proof obligation, given the event
431 %%%% name and the guard label
432 %%%% Arguments:
433 %%%% 1. (Abstract) Event name
434 %%%% 2. (Abstract) Guard label
435 %%%%
436 %%%% Usage:
437 %%%% - \grdpo{evt}{grd} will produce "evt/grd/GRD"
438 \newcommand{\grdpo}[2]{\Bevt{#1}/\Bgrd{#2}/\B@po{GRD}}
439
440 %%%% Variant Natural Number Proof Obligation
441 %%%% Print the Variant Natural Number proof obligation, given the event name
442 %%%% Arguments:
443 %%%% 1. Event name
444 %%%%
445 %%%% Usage:
446 %%%% - \natpo{evt} will produce "evt/NAT"
447 \newcommand{\natpo}[1]{\Bevt{#1}/\B@po{NAT}}
448

```

\inlineevent

```

449 \newcommand{\inlineevent}[7][]{
450   \B@inlineeventbase[#1]{#2}{#3}{#4}{#5}{#6}{#7}
451 }
452
453
454
455
456
457
458

```

```

459 %%%% (BEGIN) Macros for Pretty-Print Event-B Components %%%
460 \newcommand{\SKIP}{\textsc{skip}\xspace}
461 %

```

```

462 %%%% INITIALISATION label
463 \newBvt{INITIALISATION}

464
465 %%%% Pretty print the initialisation: no ‘refines’ clause. no parameters, no
466 %%%% guards
467 %%%% Arguments:
468 %%%% 1. (Newline(\)-separated) list of assignments.
469 %%%%
470 %%%% Usage: \initialisation{S1(v,x,y)\S2(w,x,y)}
471 %%%% will produce the following
472 %%%%
473 %%%% init
474 %%%% begin
475 %%%% S1(v, x, y)
476 %%%% S2(w, x, y)
477 %%%% end
478 %%%%
479 \newcommand{\initialisation}[1]{
480 \event{\INITIALISATION}{}{}{}{}{#1}
481 }
482
483 %\newcommand{\event}[7][{}]{
484 % \B@event[#1]{#2}{#3}{#4}{#5}{#6}{#7}
485 %}
486
487 \let\event\B@event
488 \let\Bvspace\B@vspace
489 \let\Bhspace\B@hspace
490 \let\Bpo\B@po
491

```

Numbers written in *italic* refer to the page where the corresponding entry is described; numbers underlined refer to the code line of the definition; numbers in **roman** refer to the code lines where the entry is used.

.. 4 , 44, 49, 54, 59, 67, 76, 86, 95, 96, 125, 130, 135, 140, 146, 152, 160, 167, 168	\Bstatus 47	\newBcst 294
\B@inlineeventbase 116 , 450	\Bthen 94, 166	\newBctx 292
\B@keyword 18, 24, 249 , 265, 304	\Bthm . 296, 338, 356, 388	\newBevt 300, 463
\B@keywordbase 5 , 249, 266	\Bvrb 298, 341	\newBgrd 302
\B@label 251 , 271, 337, 338, 342, 343, 345, 346	\Bvspace 488	\newBinvs 299
\B@labelbase 7 , 251, 272	\Bwhen 69, 147	\newBkeyword 304, 325–333
\B@makebox 184 , 194, 201, 208	\Bwhere 57, 138	\newBmch 297
\B@newmacro 285, 292–304	\Bwith 80, 156	\newBpar 301
\B@po . 253 , 277, 356, 366, 377, 388, 399, 407, 416, 427, 438, 447, 490	C	\newBset 293
\B@pobase . 15 , 253, 278	\carriersets 307	\newBthm 296
\B@section 207 , 221 , 314, 320	\constants 310	\newBvrb 298
\B@sectionbase 21 , 210, 222	E	P
\B@tab 17, 25, 30, 47, 52, 57, 63, 69, 80, 81, 90, 94, 97, 112, 228 , 233 , 239 , 246	\event 480, 483, 487	\pretty@absevt 50, 52, 107, 131, 133, 176
\B@vspace 226 , 231 , 237 , 244 , 488	\eventB 305	\pretty@acts 100, 111, 172, 180
\Bact . 303, 345, 399, 427	\evt@absevt 39, 49, 52, 120, 130, 133	\pretty@acts@keyword 94, 97, 101, 166, 169, 173
\Bany 57, 138	\evt@acts 43, 86, 87, 91, 124, 160, 161, 164	\pretty@acts@tmp .. . 89, 102, 163, 174
\Baxm 295, 337, 366	\evt@grds 41, 59, 64, 96, 122, 140, 144, 168	\pretty@grds 60, 68, 73, 109, 141, 147, 149, 178
\Bbap 347	\evt@label 38, 105, 119, 176	\pretty@grds@tmp 62, 70, 73, 143, 147, 149
\Bbegin 97, 169	\evt@pars ... 40, 54, 57, 67, 95, 121, 135, 138, 146, 167	\pretty@pars . 55, 57, 108, 136, 138, 177
\Bcst 294, 336	\evt@sts 37, 44, 47, 118, 125, 128	\pretty@sts .. 45, 47, 106, 126, 128, 176
\Bctx 292, 334	\evt@wits 42, 76, 82, 123, 152, 157	\pretty@wits . 77, 79, 110, 153, 155, 179
\Bend 112, 181	F	S
\Bevt 105, 176, 300, 343, 377, 388, 399, 416, 427, 438, 447	\finpo 406, 407	\setBConstructColour 261
\bfseries 347	\fispo 398, 399	\setBIdentifierColour 257
\Bgrd 302, 346, 438	G	\setBKeywordColour . 255
\Bhspace 489	\grdpo 415, 437, 438	\setBLabelColour .. 259
\Binvs 299, 342, 377	\grdthmpo 387, 388	\setBPOColour 263
\Bmch 297, 340	I	\simpo 426, 427
\Bpar 301, 344	\INITIALISATION 462 , 480	\SKIP 87, 161, 460
\Bpo 490	\initialisation 470, 479	\sl 347
\Brefines 52, 133	\inlineevent 449	
\Bset 293, 335	\invariants 319	
	\invpo 376, 377	
	L	T
	\let 487–490	\thmpo 355, 356
	N	V
	\natpo 446, 447	\variables 316
	\newBact 303	\variant 322
	\newBaxm 295	\varpo 416

Change History

v1.0		v1.1.1	
General: Initial version	1	General: Updated documentation . .	1
v1.0.1		v2.0	
\B@declaration: Ensure math-		General: Major re-implementation,	
mode	7	use etoolbox instead of ifthen	1
\B@event: Ensure math-mode	7	Updated documentation, added	
\B@section: Ensure math-mode . .	7	DoNotIndex	1
v1.1		\B@makebox: Added	6
General: Re-implement how op-		\INITIALISATION: Renamed from	
tions are defined, added options		init	15
'nobox'	1	\inlineevent: Renamed from	
		eventinline	14