

The `eventB` package*

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

February 27, 2013

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	Helper Macros	2
3.3	Declaration of Options for the Package	6
3.3.1	Option for bounding boxes	6
3.3.2	Options for font size and spacing	8
3.3.3	Options for colouring	9
3.4	Commands for Pretty-Print Event-B Models	10

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` v1.1.1, dated 2012/02/21.

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 Helper Macros

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

```
\B@keywordbase Basic macro for Event-B keywords.
4 \newcommand{\B@keywordbase}[1]{\mathbf{#1}}
```

```
\B@identifierbase Basic macro for Event-B identifiers.
5 \newcommand{\B@identifierbase}[1]{\mathit{#1}}
```

```
\B@labelbase Basic macro for Event-B labels.
6 \newcommand{\B@labelbase}[2][]{
7   \ifstrequal{#1}{}{
8     \mathsf{#2}
9   }{
10    \mathit{#2}
11  }
12 }
```

```
\B@pobase Basic macro for Event-B proof obligations.
13 \newcommand{\B@pobase}[1]{\ensuremath{\mathsf{#1}}\xspace}
```

```
\B@declarationbase Basic macro for Event-B declarations (e.g., variables, constants, etc.).
14 \newcommand{\B@declarationbase}[2]{
15   \begin{array}{l@{\B@tab}l}
16     \B@keyword{#1:} & #2
17   \end{array}
18 }
```

```
\B@sectionbase Basic macro for Event-B sections (e.g., invariants, axioms, etc.).
19 \newcommand{\B@sectionbase}[3][]{
20   \ifstrequal{#1}{}{
21     \begin{array}{l}
22       \B@keyword{#2:} \\

```

```

23     \begin{array}{l@{\B@tab}l}
24         #3
25     \end{array}
26 \end{array}
27 }{
28     \begin{array}{l@{\B@tab}l}
29         #3
30     \end{array}
31 }
32 }

```

`\B@ifstrequal` A wrapper for ake sure that the first argument is properly expanded.

```

33 \newcommand{\B@ifstrequal}{\expandafter\ifstrequal\expandafter}

```

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

```

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

```

We first save the arguments to local variables.

```

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

```

The convergence status is skipped if empty.

```

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

```

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

```

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

```

The parameters is skipped if there are none.

```

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

```

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

```

58     \B@ifstrequal{\evt@grds}{}{
59         \newcommand\pretty@grds{}
60     }{
61         \newcommand\pretty@grds@tmp{
62             \begin{array}{@{\B@tab\B@tab}l@{\B@tab}l}
63                 \evt@grds
64             \end{array}\\\}

```

```

65     }
66     \B@ifstrequal{\evt@pars}{-}{
67         \newcommand\pretty@grds{
68             \B@tab \Bwhen \\\
69             \pretty@grds@tmp
70         }
71     }{
72         \newcommand\pretty@grds{\pretty@grds@tmp}
73     }
74 }

```

The witnesses are skipped if there are none.

```

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

```

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

```

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

```

Finally we put all the pretty-print pieces together.

```

103     \begin{array}{l}
104         \Bevt{\evt@label} \\\
105         \pretty@sts
106         \pretty@absepts
107         \pretty@pars
108         \pretty@grds
109         \pretty@wits
110         \pretty@acts
111         \B@tab\Bend
112     \end{array}

```

```

113 } % END group
114 }

```

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

```

115 \newcommand{\B@inlineeventbase}[7][]{
116 { % BEGIN group
    We first save the arguments to local variables.
117   \newcommand\evt@sts{#1}% Event status
118   \newcommand\evt@label{#2}% Event label
119   \newcommand\evt@absevt{#3}% Abstract event
120   \newcommand\evt@pars{#4}% Event parameters
121   \newcommand\evt@grds{#5}% Event guards
122   \newcommand\evt@wits{#6}% Event witnesses
123   \newcommand\evt@acts{#7}% Event actions

```

The convergence status is skipped if empty.

```

124   \B@ifstrequal{\evt@sts}{}{
125     \newcommand\pretty@sts{}
126   }{
127     \newcommand\pretty@sts{(\evt@sts)}
128   }

```

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

```

129   \B@ifstrequal{\evt@absevt}{}{
130     \newcommand\pretty@absevt{}
131   }{
132     \newcommand\pretty@absevt{\Brefines~\evt@absevt}
133   }

```

The parameters is skipped if there are none.

```

134   \B@ifstrequal{\evt@pars}{}{
135     \newcommand\pretty@pars{}
136   }{
137     \newcommand\pretty@pars{\Bany~\evt@pars~\Bwhere~}
138   }

```

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

```

139   \B@ifstrequal{\evt@grds}{}{
140     \newcommand\pretty@grds{}
141   }{
142     \newcommand\pretty@grds@tmp{
143       \evt@grds~
144     }
145     \B@ifstrequal{\evt@pars}{}{
146       \Bwhen~\pretty@grds@tmp
147     }{
148       \newcommand\pretty@grds{\pretty@grds@tmp}
149     }
150   }

```

The witnesses are skipped if there are none.

```

151   \B@ifstrequal{\evt@wits}{}{
152     \newcommand\pretty@wits{}
153   }{
154     \newcommand\pretty@wits{

```

```

155     \Bwith~
156     \evt@wits~
157   }
158 }

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

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

```

Finally we put all the pretty-print pieces together.

```

175 \begin{array}{l}
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{array}
183 } % END group
184 }

```

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

```

185 \newlength{\B@tmp@length}
186 \newcommand{\B@makebox}[1]{
187   {
188     \setlength{\B@tmp@length}{\fboxsep}
189     \setlength{\fboxsep}{2ex}
190     \fbox{#1}
191     \setlength{\fboxsep}{\B@tmp@length}
192   }
193 }

```

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.

```
194 \newcommand{\B@event}[7][]{
195   \B@makebox{
196     \ensuremath{
197       \B@eventbase[#1]{#2}{#3}{#4}{#5}{#6}{#7}
198     }
199   }
200 }
```

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

```
201 \newcommand{\B@declaration}[2]{
202   \B@makebox{
203     \ensuremath{
204       \B@declarationbase{#1}{#2}
205     }
206   }
207 }
```

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

```
208 \newcommand{\B@section}[3][]{
209   \B@makebox{
210     \ensuremath{
211       \B@sectionbase[#1]{#2}{#3}
212     }
213   }
214 }
```

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

```
215 \DeclareOption{nobox}{
```

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

```
216 \renewcommand{\B@event}[7][]{
217   \B@eventbase[#1]{#2}{#3}{#4}{#5}{#6}{#7}
218 }
```

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

```
219 \renewcommand{\B@declaration}[2]{
220   \B@declarationbase{#1}{#2}
221 }
```

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

```
222 \renewcommand{\B@section}[3][]{
223   \B@sectionbase[#1]{#2}{#3}
224 }
225 }
```

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).

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

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

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

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

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

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

```
229 \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.

```
230 \DeclareOption{small}{
```

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

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

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

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

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

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

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

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

```
235 }
```

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

```
236 \DeclareOption{compact}{
```

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

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

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

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

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

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

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

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

Option `nobox` is enabled.

```
241 \ExecuteOptions{nobox}
```

```
242 }
```


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

```

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

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

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

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

Option nobox is enabled.
248   \ExecuteOptions{nobox}
249 }
```

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.
250 \newcommand{\B@keyword}[1]{\ensuremath{\B@keywordbase{#1}}\xspace}

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

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

```

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

```

261 \newcommand{\setBP0Colour}[1]{\colorlet{B@pocolor}{#1}}
262 \setBP0Colour{red}

```

We redefine the commands with colours.

```

263 \renewcommand{\B@keyword}[1]{
264   \ensuremath{\textcolor{B@keywordcolor}{\B@keywordbase{#1}}}\xspace
265 }
266 \renewcommand{\B@identifier}[1]{
267   \ensuremath{\textcolor{B@identifiercolor}{\B@identifierbase{#1}}}\xspace
268 }
269 \renewcommand{\B@label}[2][ ]{
270   \ensuremath{\textcolor{B@labelcolor}{\B@labelbase{#1}{#2}}}\xspace
271 }
272 \renewcommand{\B@po}[1]{
273   \ensuremath{\textcolor{B@pocolor}{\B@pobase{#1}}}\xspace
274 }
275 }

```

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

```

276 \DeclareOption{color}{
277   \ExecuteOptions{colour}
278 }

```

After declaration of options, we execute them accordingly.

```

279 \ProcessOptions

```

3.4 Commands for Pretty-Print Event-B Models

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

```

280 \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.

```

281 \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.

```

282 \newcommand{\carriersets}[1]{
283   \B@declaration{sets}{#1}
284 }

```

```

285 \newcommand{\constants}[1]{
286   \B@declaration{constants}{#1}
287 }

288 \newcommand{\axioms}[2][]{
289   \B@section[#1]{axioms}{#2}
290 }

291 \newcommand{\variables}[1]{
292   \B@declaration{variables}{#1}
293 }

294 \newcommand{\invariants}[2][]{
295   \B@section[#1]{invariants}{#2}
296 }

297 \newcommand{\variant}[1]{
298   \B@declaration{variant}{#1}
299 }

```

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

```

300 \newcommand{\Bany}{\B@keyword{any}}
301 \newcommand{\Bbegin}{\B@keyword{begin}}
302 \newcommand{\Bend}{\B@keyword{end}}
303 \newcommand{\Brefines}{\B@keyword{refines}}
304 \newcommand{\Bstatus}{\B@keyword{status}}
305 \newcommand{\Bthen}{\B@keyword{then}}
306 \newcommand{\Bwhen}{\B@keyword{when}}
307 \newcommand{\Bwhere}{\B@keyword{where}}
308 \newcommand{\Bwith}{\B@keyword{with}}

```

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

```

309 \newcommand{\Bctx}[1]{\ensuremath{\mathbf{#1}}\xspace}
310 \newcommand{\Bset}[1]{\B@identifier{#1}}
311 \newcommand{\Bcst}[1]{\B@identifier{#1}}
312 \newcommand{\Baxm}[1]{\B@label{#1}}
313 \newcommand{\Bthm}[1]{\B@label{thm}{#1}}
314
315 \newcommand{\Bmch}[1]{\ensuremath{\mathbf{#1}}\xspace}
316 \newcommand{\Bvrb}[1]{\B@identifier{#1}}
317 \newcommand{\Binv}[1]{\B@label{#1}}
318 \newcommand{\Bvt}[1]{\B@label{#1}}
319 \newcommand{\Bpar}[1]{\B@identifier{#1}}
320 \newcommand{\Bact}[1]{\B@label{#1}}
321 \newcommand{\Bgrd}[1]{\B@label{#1}}
322 \newcommand{\Bbap}[1]{\hbox{\sl\bfseries #1}}

```

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

```

323 \newcommand{\B@newmacro}[3][]{
324   \ifstrequal{#1}{\B@newmacro}{
325     \expandafter\def\csname #2\endcsname{#3{#2}}
326   }{

```

```

327 \expandafter\def\csname #1\endcsname{#3{#2}}
328 }
329 }

330 \newcommand{\newBctx}[2] [] {\B@newmacro{#1}{#2}{\Bctx}}
331 \newcommand{\newBset}[2] [] {\B@newmacro{#1}{#2}{\Bset}}
332 \newcommand{\newBcst}[2] [] {\B@newmacro{#1}{#2}{\Bcst}}
333 \newcommand{\newBaxm}[2] [] {\B@newmacro{#1}{#2}{\Baxm}}
334 \newcommand{\newBthm}[2] [] {\B@newmacro{#1}{#2}{\Bthm}}
335 \newcommand{\newBmch}[2] [] {\B@newmacro{#1}{#2}{\Bmch}}
336 \newcommand{\newBvrb}[2] [] {\B@newmacro{#1}{#2}{\Bvrb}}
337 \newcommand{\newBinvt}[2] [] {\B@newmacro{#1}{#2}{\Binvt}}
338 \newcommand{\newBevt}[2] [] {\B@newmacro{#1}{#2}{\Bevt}}
339 \newcommand{\newBpar}[2] [] {\B@newmacro{#1}{#2}{\Bpar}}
340 \newcommand{\newBgrd}[2] [] {\B@newmacro{#1}{#2}{\Bgrd}}
341 \newcommand{\newBact}[2] [] {\B@newmacro{#1}{#2}{\Bact}}

342
343 %%%% Theorem Proof Obligation
344 %%%% Print the theorem proof obligation, given the theorem label.
345 %%%% Arguments:
346 %%%% 1. Theorem label
347 %%%%
348 %%%% Usage:
349 %%%% - \thmpo{thm} will produce "thm/THM"
350 \newcommand{\thmpo}[1]{\Bthm{#1}/\B@po{THM}}
351

352 %%%% Axiom Well-definedness Proof Obligation
353 %%%% Print the axiom well-definedness proof obligation, given the
354 %%%% axiom label.
355 %%%% Arguments:
356 %%%% 1. Axiom label
357 %%%%
358 %%%% Usage:
359 %%%% - \axmwdpo{axm} will produce "axm/WD"
360 \newcommand{\axmwdpo}[1]{\Baxm{#1}/\B@po{WD}}
361

362 %%%% Invariant Proof Obligation
363 %%%% Print the invariant proof obligation, given the event name and
364 %%%% invariant label
365 %%%% Arguments:
366 %%%% 1. Event name
367 %%%% 2. Invariant label
368 %%%%
369 %%%% Usage:
370 %%%% - \invpo{evt}{inv} will produce "evt/inv/INV"
371 \newcommand{\invpo}[2]{\Bevt{#1}/\Binvt{#2}/\B@po{INV}}
372

```

```

373 %%%% Theorem (in guard) Proof Obligation
374 %%%% Print the simulation proof obligation, given the event name and
375 %%%% the theorem (in guard) label.
376 %%%% Arguments:
377 %%%% 1. Event name
378 %%%% 2. Theorem (in guard) label
379 %%%%
380 %%%% Usage:
381 %%%% - \grdthmpo{evt}{thm} will produce "evt/thm/THM"
382 \newcommand{\grdthmpo}[2]{\Bevt{#1}/\Bthm{#2}/\B@po{THM}}
383
384 %%%% Feasibility Proof Obligation
385 %%%% Print the feasibility proof obligation, given the event name and
386 %%%% the action label
387 %%%% Arguments:
388 %%%% 1. Event name
389 %%%% 2. Action label
390 %%%%
391 %%%% Usage:
392 %%%% - \fispo{evt}{act} will produce "evt/act/FIS"
393 \newcommand{\fispo}[2]{\Bevt{#1}/\Bact{#2}/\B@po{FIS}}
394
395 %%%% Variant finiteness Proof Obligation
396 %%%% Print the Variant finiteness proof obligation
397 %%%% Arguments: No arguments
398 %%%%
399 %%%% Usage:
400 %%%% - \finpo will produce "FIN"
401 \newcommand{\finpo}{\B@po{FIN}}
402
403 %%%% Variant Proof Obligation
404 %%%% Print the guard strengthen proof obligation, given the event name
405 %%%% Arguments:
406 %%%% 1. Event name
407 %%%%
408 %%%% Usage:
409 %%%% - \grdpo{evt} will produce "evt/VAR"
410 \newcommand{\varpo}[1]{\Bevt{#1}/\B@po{VAR}}
411
412 %%%% Simulation Proof Obligation
413 %%%% Print the simulation proof obligation, given the event name and
414 %%%% the action label.
415 %%%% Arguments:
416 %%%% 1. Event name
417 %%%% 2. Action label
418 %%%%
419 %%%% Usage:
420 %%%% - \simpo{evt}{act} will produce "evt/act/SIM"
421 \newcommand{\simpo}[2]{\Bevt{#1}/\Bact{#2}/\B@po{SIM}}
422
423 %%%% Guard Strengthen Proof Obligation
424 %%%% Print the guard strengthen proof obligation, given the event

```

```

425 %%%% name and the guard label
426 %%%% Arguments:
427 %%%% 1. (Abstract) Event name
428 %%%% 2. (Abstract) Guard label
429 %%%%
430 %%%% Usage:
431 %%%% - \grdpo{evt}{grd} will produce "evt/grd/GRD"
432 \newcommand{\grdpo}[2]{\Bevt{#1}/\Bgrd{#2}/\B@po{GRD}}
433
434 %%%% Variant Natural Number Proof Obligation
435 %%%% Print the Variant Natural Number proof obligation, given the event name
436 %%%% Arguments:
437 %%%% 1. Event name
438 %%%%
439 %%%% Usage:
440 %%%% - \natpo{evt} will produce "evt/NAT"
441 \newcommand{\natpo}[1]{\Bevt{#1}/\B@po{NAT}}
442
\inlineevent
443 \newcommand{\inlineevent}[7][]{
444   \B@inlineeventbase[#1]{#2}{#3}{#4}{#5}{#6}{#7}
445 }
446
447
448
449
450
451
452
453 %%%% (BEGIN) Macros for Pretty-Print Event-B Components %%%
454 \newcommand{\SKIP}{\textsc{skip}\xspace}
455 %
\INITIALISATION
456 %%%% INITIALISATION label
457 \newBevt{INITIALISATION}
458
459 %%%% Pretty print the initialisation: no ‘‘refines’’ clause. no parameters, no
460 %%%% guards
461 %%%% Arguments:
462 %%%% 1. (Newline(\))-separated) list of assignments.
463 %%%%
464 %%%% Usage: \initialisation{S1(v,x,y)\S2(w,x,y)}
465 %%%% will produce the following
466 %%%%
467 %%%% init
468 %%%% begin
469 %%%% S1(v, x, y)
470 %%%% S2(w, x, y)
471 %%%% end

```


E		\initialisation 464, 473	\pretty@acts@tmp ..
\event 474, 477, 481		\inlineevent <u>443</u>	. 88, 101, 162, 173
\eventB 280		\invariants 294	\pretty@grds
\evt@absevtS ... 38,		\invpo 370, 371	... 59, 67, 72,
48, 51, 119, 129, 132			108, 140, 148, 178
\evt@acts	L		\pretty@grds@tmp 61,
. 42, 85, 86, 90,	\let 481–484		69, 72, 142, 146, 148
123, 159, 160, 163	N		\pretty@pars . 54, 56,
\evt@grds	\natpo 440, 441		107, 135, 137, 177
. 40, 58, 63, 95,	\newBact 341		\pretty@sts .. 44, 46,
121, 139, 143, 167	\newBaxm 333		105, 125, 127, 176
\evt@label	\newBcst 332		\pretty@wits . 76, 78,
. 37, 104, 118, 176	\newBctx 330		109, 152, 154, 179
\evt@pars ... 39, 53,	\newBevt 338, 457	S	
56, 66, 94, 120,	\newBgrd 340	\setBIdentifierColour	
134, 137, 145, 166	\newBinv 337 <u>257</u>	
\evt@sts 36,	\newBmch 335	\setBKeywordColour . <u>255</u>	
43, 46, 117, 124, 127	\newBpar 339	\setBLabelColour .. <u>259</u>	
\evt@wits 41,	\newBset 331	\setBPOColour <u>261</u>	
75, 81, 122, 151, 156	\newBthm 334	\simpo 420, 421	
	\newBvrb 336	\SKIP 86, 160, 454	
F	P	\sl 322	
\finpo 400, 401	\pretty@absevtS ...	T	
\fispo 392, 393 49, 51,	\thmpo 349, 350	
	106, 130, 132, 176	V	
G	\pretty@acts	\variables 291	
\grdp 409, 431, 432	. 99, 110, 171, 180	\variant 297	
\grdthmpo 381, 382	\pretty@acts@keyword	\varpo 410	
 93, 96,		
I	100, 165, 168, 172		
\INITIALISATION <u>456</u> , 474			

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 14
'nobox' 1	\inlineevent: Renamed from
	eventinline 14