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

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`.

```
\RequirePackage{xspace}
\RequirePackage{xcolor}
\RequirePackage{etoolbox}
```

3.2 Internal Helper Macros

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

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

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

`\B@keywordbase` Basic macro for Event-B keywords.

```
\newcommand{\B@keywordbase}[1]{\mathbf{#1}}
```

`\B@identifierbase` Basic macro for Event-B identifiers.

```
\newcommand{\B@identifierbase}[1]{\mathit{#1}}
```

`\B@labelbase` Basic macro for Event-B labels.

```
\newcommand{\B@labelbase}[2][]{%
  \ifstrequal{#1}{}{%
    \mathsf{#2}
  }{%
    \mathit{#2}
  }
}
```

`\B@constructbase` Basic macro for Event-B constructs.

```
\newcommand{\B@constructbase}[1]{\mathsf{#1}}
```

`\B@pobase` Basic macro for Event-B proof obligations.

```
\newcommand{\B@pobase}[1]{\mathsf{#1}}
```

`\B@declarationbase` Basic macro for Event-B declarations (e.g., constants, variables, etc.).

```
\newcommand{\B@declarationbase}[2]{%
  \begin{array}{@{}l@{\B@tab}l@{}}
    \B@keyword{#1:} & #2
  \end{array}
}
```

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

```
\newcommand{\B@sectionbase}[3][]{%
  \ifstrequal{#1}{}{
    \begin{array}{@{}l@{}}
      \B@keyword{#2:} \\
      \begin{array}{@{\B@tab}l@{\B@tab}l@{}}
        #3
      \end{array}
    \end{array}
  }{
    \begin{array}{@{\B@tab}l@{\B@tab}l@{\B@tab}}
      #3
    \end{array}
  }
}
```

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

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

We first save the arguments to local variables.

```
    \newcommand\evt@sts{#1}% Event status
    \newcommand\evt@label{#2}% Event label
    \newcommand\evt@absevts{#3}% Abstract event
    \newcommand\evt@pars{#4}% Event parameters
    \newcommand\evt@grds{#5}% Event guards
    \newcommand\evt@wits{#6}% Event witnesses
    \newcommand\evt@acts{#7}% Event actions
```

The convergence status is skipped if empty.

```
    \B@ifstrequal{\evt@sts}{}{
      \newcommand\pretty@sts{}
    }{
      \newcommand\pretty@sts{\B@tab\Bstatus \B@tab \evt@sts \\}
    }
```

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

```
    \B@ifstrequal{\evt@absevts}{}{
      \newcommand\pretty@absevts{}
    }{
      \newcommand\pretty@absevts{\B@tab \Brefines \B@tab \evt@absevts{} \\}
    }
```

The parameters is skipped if there are none.

```
    \B@ifstrequal{\evt@pars}{}{
      \newcommand\pretty@pars{}
    }{
      \newcommand\pretty@pars{\B@tab \Bany \B@tab \evt@pars \B@tab \Bwhere \\}
    }
```

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

```
    \B@ifstrequal{\evt@grds}{}{
      \newcommand\pretty@grds{}
    }{
      \newcommand\pretty@grds@tmp{
```

```
      \begin{array}{@{\B@tab\B@tab}l@{\B@tab}l}
        \evt@grds
      \end{array}\\
    }
    \B@ifstrequal{\evt@pars}{}{
      \newcommand\pretty@grds{
        \B@tab \Bwhen \\
        \pretty@grds@tmp
      }
    }{
      \newcommand\pretty@grds{\pretty@grds@tmp}
    }
  }
```

The witnesses are skipped if there are none.

```
  \B@ifstrequal{\evt@wits}{}{
    \newcommand\pretty@wits{}
  }{
    \newcommand\pretty@wits{
      \B@tab\Bwith\\
      \begin{array}{@{\B@tab\B@tab}ll}
        \evt@wits
      \end{array}\\
    }
  }
```

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

```
  \B@ifstrequal{\evt@acts}{}{
    \renewcommand\evt@acts{\SKIP}
  }{}
  \newcommand\pretty@acts@tmp{
    \begin{array}{@{\B@tab\B@tab}l@{\B@tab}l}
      \evt@acts
    \end{array}\\
  }
  \newcommand\pretty@acts@keyword{\B@tab\Bthen \\}
  \B@ifstrequal{\evt@pars}{}{
    \B@ifstrequal{\evt@grds}{}{
      \renewcommand\pretty@acts@keyword{\B@tab\Bbegin \\}
    }{}
  }{}
  \newcommand\pretty@acts{
    \pretty@acts@keyword
    \pretty@acts@tmp
  }
```

Finally we put all the pretty-print pieces together.

```
  \begin{array}{@{}l@{}}
    \Bevt{\evt@label} \\
    \pretty@sts
    \pretty@absevts
    \pretty@pars
    \pretty@grds
    \pretty@wits
```

```
      \pretty@acts
      \B@tab\Bend
    \end{array}
  } % END group
}
```

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

```
\newcommand{\B@inlineeventbase}[7][]{
  { % BEGIN group
```

We first save the arguments to local variables.

```
    \newcommand\evt@sts{#1}% Event status
    \newcommand\evt@label{#2}% Event label
    \newcommand\evt@absevts{#3}% Abstract event
    \newcommand\evt@pars{#4}% Event parameters
    \newcommand\evt@grds{#5}% Event guards
    \newcommand\evt@wits{#6}% Event witnesses
    \newcommand\evt@acts{#7}% Event actions
```

The convergence status is skipped if empty.

```
    \B@ifstrequal{\evt@sts}{}{
      \newcommand\pretty@sts{}
    }{
      \newcommand\pretty@sts{(\evt@sts)}
    }
```

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

```
    \B@ifstrequal{\evt@absevts}{}{
      \newcommand\pretty@absevts{}
    }{
      \newcommand\pretty@absevts{~\Brefines~\evt@absevts}
    }
```

The parameters is skipped if there are none.

```
    \B@ifstrequal{\evt@pars}{}{
      \newcommand\pretty@pars{}
    }{
      \newcommand\pretty@pars{\Bany~\evt@pars~\Bwhere~}
    }
```

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

```
    \B@ifstrequal{\evt@grds}{}{
      \newcommand\pretty@grds{}
    }{
      \newcommand\pretty@grds@tmp{
        \evt@grds~
      }
      \B@ifstrequal{\evt@pars}{}{
        \newcommand\pretty@grds{\Bwhen~\pretty@grds@tmp}
      }{
        \newcommand\pretty@grds{\pretty@grds@tmp}
      }
    }
```

The witnesses are skipped if there are none.

```
    \B@ifstrequal{\evt@wits}{}{
```

```
    \newcommand\pretty@wits{}
  }{
    \newcommand\pretty@wits{
      \Bwith~
      \evt@wits~
    }
  }
```

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

```
  \B@ifstrequal{\evt@acts}{}{
    \renewcommand\evt@acts{\SKIP}
  }{}
  \newcommand\pretty@acts@tmp{
    \evt@acts
  }
  \newcommand\pretty@acts@keyword{\Bthen}
  \B@ifstrequal{\evt@pars}{}{
    \B@ifstrequal{\evt@grds}{}{
      \renewcommand\pretty@acts@keyword{\Bbegin}
    }{}
  }{}
  \newcommand\pretty@acts{
    \pretty@acts@keyword~
    \pretty@acts@tmp~
  }
```

Finally we put all the pretty-print pieces together.

```
    \Bevt{\evt@label}\pretty@sts\pretty@absevts~\widehat{=}~
    \pretty@pars
    \pretty@grds
    \pretty@wits
    \pretty@acts
    \Bend
  } % END group
}
```

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

```
\newlength{\B@tmp@length}
\newcommand{\B@makebox}[1]{
  {
    \setlength{\B@tmp@length}{\fboxsep}
    \setlength{\fboxsep}{2ex}
    \fbox{#1}
    \setlength{\fboxsep}{\B@tmp@length}
  }
}
```

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.

```
\newcommand{\B@event}[7][]{
  \B@makebox{
    \ensuremath{
      \B@eventbase[#1]{#2}{#3}{#4}{#5}{#6}{#7}
    }
  }
}
```

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

```
\newcommand{\B@declaration}[2]{
  \B@makebox{%
    \ensuremath{%
      \B@declarationbase{#1}{#2}
    }
  }
}
```

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

```
\newcommand{\B@section}[3][]{
  \B@makebox{%
    \ensuremath{%
      \B@sectionbase[#1]{#2}{#3}
    }
  }
}
```

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

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

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

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

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

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

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

```
  \renewcommand{\B@section}[3][]{
    \B@sectionbase[#1]{#2}{#3}
  }
}
```

3.3.2 Options for font size and spacing

We define the default values for font size and some spacing commands, and how the 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).

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

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

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

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

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

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

```
\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.

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

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

Option `nobox` is enabled.

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

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

```
\DeclareOption{tiny}{
```

`\B@fontsize` Redefine to be `\scriptsize` for option `tiny`.

```
  \renewcommand{\B@fontsize}{\scriptsize}
```

`\B@vspace` Redefine to be `-0.5ex` for option `tiny`.

```
  \renewcommand{\B@vspace}[1][-0.5ex]{\\[#1]}
```

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

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

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

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

Option `nobox` is enabled.

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

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.

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

`\B@identifier` Macro for Event-B identifiers.

```
\newcommand{\B@identifier}[1]{\ensuremath{\B@identifierbase{#1}}\xspace}
```

`\B@label` Macro for Event-B labels.

```
\newcommand{\B@label}[2][]{\ensuremath{\B@labelbase[#1]{#2}}\xspace}
```

`\B@construct` Macro for Event-B constructs.

```
\newcommand{\B@construct}[1]{\ensuremath{\B@constructbase{#1}}\xspace}
```

`\B@po` Macro for Event-B proof obligations.

```
\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.

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

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

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

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

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

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

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

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

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

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

```
  \newcommand{\setBPOColour}[1]{\colorlet{B@pocolor}{#1}}
  \setBPOColour{red}
```

We redefine the commands with colours.

```
  \renewcommand{\B@keyword}[1]{
    \ensuremath{\textcolor{B@keywordcolor}{\B@keywordbase{#1}}}\xspace
  }
  \renewcommand{\B@identifier}[1]{
    \ensuremath{\textcolor{B@identifiercolor}{\B@identifierbase{#1}}}\xspace
  }
  \renewcommand{\B@label}[2][]{
    \ensuremath{\textcolor{B@labelcolor}{\B@labelbase[#1]{#2}}}\xspace
  }
  \renewcommand{\B@construct}[1]{
    \ensuremath{\textcolor{B@constructcolor}{\B@constructbase{#1}}}\xspace
  }
  \renewcommand{\B@po}[1]{
    \ensuremath{\textcolor{B@pocolor}{\B@pobase{#1}}}\xspace
  }
}
```

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

```
\DeclareOption{color}{
  \ExecuteOptions{colour}
}
```

After declaration of options, we execute them accordingly.

```
\ProcessOptions
```

3.4 Meta-macros for creating macros for modelling elements

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

```
\newcommand{\B@newmacro}[3][]{
  \ifstrequal{#1}{}{
    \expandafter\def\csname #2\endcsname{#3{#2}}
  }{
    \expandafter\def\csname #1\endcsname{#3{#2}}
  }
}
\newcommand{\newBctx}[2][]{\B@newmacro[#1]{#2}{\Bctx}}
\newcommand{\newBset}[2][]{\B@newmacro[#1]{#2}{\Bset}}
\newcommand{\newBcst}[2][]{\B@newmacro[#1]{#2}{\Bcst}}
\newcommand{\newBaxm}[2][]{\B@newmacro[#1]{#2}{\Baxm}}
\newcommand{\newBthm}[2][]{\B@newmacro[#1]{#2}{\Bthm}}
\newcommand{\newBmch}[2][]{\B@newmacro[#1]{#2}{\Bmch}}
\newcommand{\newBvrb}[2][]{\B@newmacro[#1]{#2}{\Bvrb}}
\newcommand{\newBinv}[2][]{\B@newmacro[#1]{#2}{\Binv}}
\newcommand{\newBevt}[2][]{\B@newmacro[#1]{#2}{\Bevt}}
\newcommand{\newBpar}[2][]{\B@newmacro[#1]{#2}{\Bpar}}
\newcommand{\newBgrd}[2][]{\B@newmacro[#1]{#2}{\Bgrd}}
\newcommand{\newBact}[2][]{\B@newmacro[#1]{#2}{\Bact}}
\newcommand{\newBkeyword}[2][]{\B@newmacro[#1]{#2}{\B@keyword}}
```

3.5 Commands for Pretty-Print Event-B Models

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

```
\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.

```
\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.

```
\newcommand{\carriersets}[1]{%
  \B@declaration{sets}{#1}
}
\newcommand{\constants}[1]{%
  \B@declaration{constants}{#1}
}
```

```
\newcommand{\axioms}[2][]{%
  \B@section[#1]{axioms}{#2}
}
\newcommand{\variables}[1]{
  \B@declaration{variables}{#1}
}
\newcommand{\invariants}[2][]{
  \B@section[#1]{invariants}{#2}
}
\newcommand{\variant}[1]{
  \B@declaration{variant}{#1}
}
```

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

```
\newBkeyword[Bany]{any}
\newBkeyword[Bbegin]{begin}
\newBkeyword[Brefines]{refines}
\newBkeyword[Bstatus]{status}
\newBkeyword[Bthen]{then}
\newBkeyword[Bwhen]{when}
\newBkeyword[Bwhere]{where}
\newBkeyword[Bwith]{with}
\newBkeyword[Bend]{end}
```

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

```
\newcommand{\Bctx}[1]{\B@construct{#1}}
\newcommand{\Bset}[1]{\B@identifier{#1}}
\newcommand{\Bcst}[1]{\B@identifier{#1}}
\newcommand{\Baxm}[1]{\B@label{#1}}
\newcommand{\Bthm}[1]{\B@label[thm]{#1}}

\newcommand{\Bmch}[1]{\B@construct{#1}}
\newcommand{\Bvrb}[1]{\B@identifier{#1}}
\newcommand{\Binv}[1]{\B@label{#1}}
\newcommand{\Bevt}[1]{\B@label{#1}}
\newcommand{\Bpar}[1]{\B@identifier{#1}}
\newcommand{\Bact}[1]{\B@label{#1}}
\newcommand{\Bgrd}[1]{\B@label{#1}}
\newcommand{\Bbap}[1]{\hbox{\sl\bfseries #1}}

%%%%% Theorem Proof Obligation
%%%%% Print the theorem proof obligation, given the theorem label.
%%%%% Arguments:
%%%%% 1. Theorem label
%%%%%
%%%%% Usage:
%%%%% - \thmpo{thm} will produce "thm/THM"
\newcommand{\thmpo}[1]{\Bthm{#1}/\B@po{THM}}

```

```
%%%% Axiom Well-definedness Proof Obligation
%%%% Print the axiom well-definedness proof obligation, given the
%%%% axiom label.
%%%% Arguments:
%%%% 1. Axiom label
%%%%
%%%% Usage:
%%%% - \axmwdpo{axm} will produce "axm/WD"
\newcommand{\axmwdpo}[1]{\Baxm{#1}/\B@po{WD}}

%%%% Invariant Proof Obligation
%%%% Print the invariant proof obligation, given the event name and
%%%% invariant label
%%%% Arguments:
%%%% 1. Event name
%%%% 2. Invariant label
%%%%
%%%% Usage:
%%%% - \invpo{evt}{inv} will produce "evt/inv/INV"
\newcommand{\invpo}[2]{\Bevt{#1}/\Binv{#2}/\B@po{INV}}

%%%% Theorem (in guard) Proof Obligation
%%%% Print the simulation proof obligation, given the event name and
%%%% the theorem (in guard) label.
%%%% Arguments:
%%%% 1. Event name
%%%% 2. Theorem (in guard) label
%%%%
%%%% Usage:
%%%% - \grdthmpo{evt}{thm} will produce "evt/thm/THM"
\newcommand{\grdthmpo}[2]{\Bevt{#1}/\Bthm{#2}/\B@po{THM}}

%%%% Feasibility Proof Obligation
%%%% Print the feasibility proof obligation, given the event name and
%%%% the action label
%%%% Arguments:
%%%% 1. Event name
%%%% 2. Action label
%%%%
%%%% Usage:
%%%% - \fispo{evt}{act} will produce "evt/act/FIS"
\newcommand{\fispo}[2]{\Bevt{#1}/\Bact{#2}/\B@po{FIS}}

%%%% Variant finiteness Proof Obligation
%%%% Print the Variant finiteness proof obligation
%%%% Arguments: No arguments
%%%%
%%%% Usage:
%%%% - \finpo will produce "FIN"
\newcommand{\finpo}{\B@po{FIN}}

%%%% Variant Proof Obligation
```

```
%%%% Print the guard strengthen proof obligation, given the event name
%%%% Arguments:
%%%% 1. Event name
%%%%
%%%% Usage:
%%%% - \grdpo{evt} will produce "evt/VAR"
\newcommand{\varpo}[1]{\Bevt{#1}/\B@po{VAR}}

%%%% Simulation Proof Obligation
%%%% Print the simulation proof obligation, given the event name and
%%%% the action label.
%%%% Arguments:
%%%% 1. Event name
%%%% 2. Action label
%%%%
%%%% Usage:
%%%% - \simpo{evt}{act} will produce "evt/act/SIM"
\newcommand{\simpo}[2]{\Bevt{#1}/\Bact{#2}/\B@po{SIM}}

%%%% Guard Strengthen Proof Obligation
%%%% Print the guard strengthen proof obligation, given the event
%%%% name and the guard label
%%%% Arguments:
%%%% 1. (Abstract) Event name
%%%% 2. (Abstract) Guard label
%%%%
%%%% Usage:
%%%% - \grdpo{evt}{grd} will produce "evt/grd/GRD"
\newcommand{\grdpo}[2]{\Bevt{#1}/\Bgrd{#2}/\B@po{GRD}}

%%%% Variant Natural Number Proof Obligation
%%%% Print the Variant Natural Number proof obligation, given the event name
%%%% Arguments:
%%%% 1. Event name
%%%%
%%%% Usage:
%%%% - \natpo{evt} will produce "evt/NAT"
\newcommand{\natpo}[1]{\Bevt{#1}/\B@po{NAT}}
```

\inlineevent

```
\newcommand{\inlineevent}[7][]{
  \B@inlineeventbase[#1]{#2}{#3}{#4}{#5}{#6}{#7}
}
```

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

\INITIALISATION

```
%%%% INITIALISATION label
\newBevt{INITIALISATION}

%%%% Pretty print the initialisation: no ‘‘refines’’ clause. no parameters, no
%%%% guards
%%%% Arguments:
%%%% 1. (Newline(\\)-separated) list of assignments.
%%%%
%%%% Usage: \initialisation{S1(v,x,y)\\S2(w,x,y)}
%%%%       will produce the following
%%%%
%%%%       init
%%%%       begin
%%%%         S1(v, x, y)
%%%%         S2(w, x, y)
%%%%       end
%%%%
\newcommand{\initialisation}[1]{
  \event{\INITIALISATION}{}{}{}{}{#1}
}

%\newcommand{\event}[7][]{
%  \B@event[#1]{#2}{#3}{#4}{#5}{#6}{#7}
%}

\let\event\B@event
\let\Bvspace\B@vspace
\let\Bhspace\B@hspace
\let\Bpo\B@po

```

Index

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.

C

E

F

G

I

L

N

P

S

T

V

Change History