The eventB package*

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

February 26, 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

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.

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.

*This document corresponds to eventB v1.1.1, dated 2012/02/21.

3 Implementation

3.1 Package Loading

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

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

3.2 Declaration of Options for the Package

In this part various options for the package are defined.

Option for rounding boxes By default, Event-B modelling elements, e.g., invariants, events, etc., are displayed in a rounding box. This option enables them to be displayed without the rounding box.

```
\newlength{\B@oldfboxsep}
\newcommand{\event}[7][]{
  \setlength{\B@oldfboxsep}{\fboxsep}
  \setlength{\fboxsep}{2ex}
  \fbox{
    \ensuremath{
      \B@event[#1]{#2}{#3}{#4}{#5}{#6}{#7}
    }
  }
  \setlength{\fboxsep}{\B@oldfboxsep}
}

\newcommand{\B@declaration}[2]{
  \setlength{\B@oldfboxsep}{\fboxsep}
  \setlength{\fboxsep}{2ex}
  \fbox{
    \ensuremath{
      \B@declarationbase{#1}{#2}
    }
  }
  \setlength{\fboxsep}{\B@oldfboxsep}
}

\newcommand{\B@section}[3][]{
  \setlength{\B@oldfboxsep}{\fboxsep}
  \setlength{\fboxsep}{2ex}
  \fbox{
    \ensuremath{
      \B@sectionbase[#1]{#2}{#3}
    }
  }
  \setlength{\fboxsep}{\B@oldfboxsep}
}

\DeclareOption{nobox}{
  \renewcommand{\event}[7][]{
    \B@event[#1]{#2}{#3}{#4}{#5}{#6}{#7}
  }

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

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

```

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`.

```
\newcommand{\B@fontsize}{\normalsize} % The font size used in Bcode environment
\newcommand{\Bvspace}[1][2ex]{\\[#1]} % Vertical space
\newcommand{\Bhspace}[1][2em]{\hspace{#1}} % Horizontal space
\newcommand{\B@tab}{\quad} % A small separation space

\DeclareOption{small}{
  \renewcommand{\B@fontsize}{\small}
  \renewcommand{\Bvspace}[1][1ex]{\\[#1]}
  \renewcommand{\Bhspace}[1][1em]{\hspace{#1}}
  \renewcommand{\B@tab}{\ }
}
\DeclareOption{compact}{
  \renewcommand{\B@fontsize}{\footnotesize}
  \renewcommand{\Bvspace}[1][0ex]{\\[#1]}
  \renewcommand{\Bhspace}[1][0.5em]{\hspace{#1}}
  \renewcommand{\B@tab}{\ }
  \ExecuteOptions{nobox}
}
\DeclareOption{tiny}{
  \renewcommand{\B@fontsize}{\scriptsize}
  \renewcommand{\Bvspace}[1][-0.5ex]{\\[#1]}
  \renewcommand{\Bhspace}[1][0.5em]{\hspace{#1}}
  \renewcommand{\B@tab}{\ }
  \ExecuteOptions{nobox}
}

```

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.

```
\newcommand{\B@keyword}[1]{\ensuremath{\B@keywordbase{#1}}\xspace}
\newcommand{\Bidentifier}[1]{\ensuremath{\B@identifier{#1}}\xspace}
\newcommand{\Blabel}[2][]{\ensuremath{\B@label[#1]{#2}}\xspace}
\newcommand{\Bpo}[1]{\ensuremath{\B@po{#1}}\xspace}
```

```
\DeclareOption{colour}{
  \newcommand{\setBKeywordColour}[1]{\colorlet{B@keywordcolor}{#1}}
  \setBKeywordColour{blue}
  \newcommand{\setBIdentifierColour}[1]{\colorlet{B@identifiercolor}{#1}}
  \setBIdentifierColour{blue!50!red}
  \newcommand{\setBLabelColour}[1]{\colorlet{B@labelcolor}{#1}}
  \setBLabelColour{green!50!black}
  \newcommand{\setBPOColour}[1]{\colorlet{B@pocolor}{#1}}
  \setBPOColour{red}
  \renewcommand{\B@keyword}[1]{
    \ensuremath{\textcolor{B@keywordcolor}{\B@keywordbase{#1}}}\xspace
  }
  \renewcommand{\Bidentifier}[1]{
    \ensuremath{\textcolor{B@identifiercolor}{\B@identifier{#1}}}\xspace
  }
  \renewcommand{\Blabel}[2][]{
    \ensuremath{\textcolor{B@labelcolor}{\B@label[#1]{#2}}}\xspace
  }
  \renewcommand{\Bpo}[1]{
    \ensuremath{\textcolor{B@pocolor}{\B@po{#1}}}\xspace
  }
}
\DeclareOption{color}{
  \ExecuteOptions{colour}
}

```

After declaration of options, we execute them accordingly.

```
\ProcessOptions
```

3.3 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.

```
\newcommand{\Bany}{\B@keyword{any}}
\newcommand{\Bbegin}{\B@keyword{begin}}
\newcommand{\Bend}{\B@keyword{end}}
\newcommand{\Brefines}{\B@keyword{refines}}
\newcommand{\Bstatus}{\B@keyword{status}}
\newcommand{\Bthen}{\B@keyword{then}}
\newcommand{\Bwhen}{\B@keyword{when}}
\newcommand{\Bwhere}{\B@keyword{where}}
\newcommand{\Bwith}{\B@keyword{with}}
```

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

```
\newcommand{\Bctx}[1]{\ensuremath{\mathbf{#1}}\xspace}
\newcommand{\Bset}[1]{\Bidentifier{#1}}
\newcommand{\Bcst}[1]{\Bidentifier{#1}}
\newcommand{\Baxm}[1]{\Blabel{#1}}
\newcommand{\Bthm}[1]{\Blabel[thm]{#1}}

\newcommand{\Bmch}[1]{\ensuremath{\mathbf{#1}}\xspace}
\newcommand{\Bvrb}[1]{\Bidentifier{#1}}
\newcommand{\Binv}[1]{\Blabel{#1}}
\newcommand{\Bevt}[1]{\Blabel{#1}}
\newcommand{\Bpar}[1]{\Bidentifier{#1}}
\newcommand{\Bact}[1]{\Blabel{#1}}
\newcommand{\Bgrd}[1]{\Blabel{#1}}
\newcommand{\Bbap}[1]{\hbox{\sl\bfseries #1}}
```

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}{}{
  %\ifthenelse{\equal{#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}}

%%%%% 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}/\Bpo{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}/\Bpo{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}/\Bpo{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}/\Bpo{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}/\Bpo{FIS}}

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

\newcommand{\B@keywordbase}[1]{\mathbf{#1}}
\newcommand{\B@identifier}[1]{\mathit{#1}}
\newcommand{\B@label}[2][]{
  \ifthenelse{\equal{#1}{}}{
    \mathsf{#2}
  }{
    \mathit{#2}
  }
}



\newcommand{\eventinline}[7][]{
  \setlength{\B@oldfboxsep}{\fboxsep}
  \setlength{\fboxsep}{2ex}
  \fbox{
    \ensuremath{
      \B@eventinline[#1]{#2}{#3}{#4}{#5}{#6}{#7}
    }
  }
  \setlength{\fboxsep}{\B@oldfboxsep}
}


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

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

```
}

\newcommand{\B@po}[1]{\ensuremath{\mathsf{#1}}\xspace}

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


%%%%% Pretty print an general Event-B event
%%%%% Arguments:
%%%%% 1. (Optional) convergence status.
%%%%% 2. Name of the event.
%%%%% 3. Name of the abstract event.
%%%%% 4. (Comma-separated) list of parameters.
%%%%% 5. (Newline(\\)-separated) list of guards.
%%%%% 6. (Newline(\\)-separated) list of witness predicates.
%%%%% 7. (Newline(\\)-separated) list of assignments.
%%%%%
%%%%% Usage: \B@event[conv]{conc}{abs}{x,y}{G1(x,y)\\G2(x,y)}{W1\\W2}{S1(v,x,y)\\S2(w,x,y)}
%%%%%       will produce the following
%%%%%
%%%%%       conc
%%%%%       refines abs
%%%%%       status conv
%%%%%       any x, y where
%%%%%         G1(x, y)
%%%%%         G2(x, y)
%%%%%       with
%%%%%         W1
%%%%%         W2
%%%%%       then
%%%%%         S1(v, x, y)
%%%%%         S2(w, x, y)
%%%%%       end
%%%%%
%%%%% Special case:
%%%%% - Empty abstract event --> refines clause is omitted.
%%%%% - Empty convergence status --> status clause is omitted.
%%%%% - Empty witness --> with clause is omitted.
%%%%% - Empty parameters, empty guards --> begin ... end
%%%%% - Empty parameters --> when ... then ... end
%%%%% - Empty actions --> \SKIP
\newcommand{\B@event}[7][]{
  \def\evt@sts{#1}
  \def\evt@name{#2}
  \def\evt@absevts{#3}
  \def\evt@pars{#4}
  \def\evt@grds{#5}
  \def\evt@wits{#6}
  \def\evt@acts{#7}
  %% Pretty-print convergence status
  \ifthenelse{\equal{\evt@sts}{}}{
    \def\pretty@sts{}
  }{
    \def\pretty@sts{\B@tab\Bstatus \B@tab \evt@sts \\}
  }
  % Pretty-print abstract events
  \ifthenelse{\equal{\evt@absevts}{}}{
    \def\pretty@absevts{}
  }{
    \def\pretty@absevts{\B@tab\Brefines \B@tab \evt@absevts \\}
  }
  % Pretty-print parameters
  \ifthenelse{\equal{\evt@pars}{}}{
    \def\pretty@pars{}
  }{
    \def\pretty@pars{\B@tab\Bany \B@tab \evt@pars \B@tab \Bwhere \\}
  }
  % Pretty-print guards
  \ifx\evt@grds\@empty
  \def\pretty@grds{}
  \else
  \def\evt@grds@tmp{
    \begin{array}{@{\B@tab\B@tab}l@{\B@tab}l}
      \evt@grds
    \end{array}\\
  }
  \ifthenelse{\equal{\evt@pars}{}}{
    \def\pretty@grds{
      \B@tab \Bwhen \\
      \evt@grds@tmp
    }
  }{
    \def\pretty@grds{\evt@grds@tmp}
  }
  \fi
  % Pretty-print witnesses
  \ifx\evt@wits\@empty
  \def\pretty@wits{}
  \else
  \def\pretty@wits{
    \B@tab\Bwith\\
    \begin{array}{@{\B@tab\B@tab}ll}
      \evt@wits
    \end{array}\\
  }
  \fi
  % Pretty-print actions
  \ifx\evt@acts\@empty
  \def\evt@acts{\SKIP}
  \else
  \fi
  \def\evt@acts@tmp{
    \begin{array}{@{\B@tab\B@tab}l@{\B@tab}l}
      \evt@acts
    \end{array}\\
  }
```

```
  \def\evt@acts@keyword{\B@tab\Bthen \\}
  \ifx\evt@pars\@empty
  \ifx\evt@grds\@empty
  \def\evt@acts@keyword{\B@tab\Bbegin \\}
  \else
  \fi
  \else
  \fi
  \def\pretty@acts{
    \evt@acts@keyword
    \evt@acts@tmp
  }
  % Really do it now
  \begin{array}{l}
    \Bevt{\evt@name} \\
    \pretty@sts
    \pretty@absevts
    \pretty@pars
    \pretty@grds
    \pretty@wits
    \pretty@acts
    \B@tab\Bend
  \end{array}
}

%%%%% Pretty print an general Event-B event
%%%%% Arguments:
%%%%% 1. (Optional) convergence status.
%%%%% 2. Name of the event.
%%%%% 3. Name of the abstract event.
%%%%% 4. (Comma-separated) list of parameters.
%%%%% 5. (Newline(\\)-separated) list of guards.
%%%%% 6. (Newline(\\)-separated) list of witness predicates.
%%%%% 7. (Newline(\\)-separated) list of assignments.
%%%%%
%%%%% Usage: \B@event[conv]{conc}{abs}{x,y}{G1(x,y)\\G2(x,y)}{W1\\W2}{S1(v,x,y)\\S2(w,x,y)}
%%%%%       will produce the following
%%%%%
%%%%%       conc
%%%%%       refines abs
%%%%%       status conv
%%%%%       any x, y where
%%%%%         G1(x, y)
%%%%%         G2(x, y)
%%%%%       with
%%%%%         W1
%%%%%         W2
%%%%%       then
%%%%%         S1(v, x, y)
%%%%%         S2(w, x, y)
%%%%%       end
%%%%%
%%%%% Special case:
%%%%% - Empty abstract event --> refines clause is omitted.
```

```
%%%%% - Empty convergence status --> status clause is omitted.
%%%%% - Empty witness --> with clause is omitted.
%%%%% - Empty parameters, empty guards --> begin ... end
%%%%% - Empty parameters --> when ... then ... end
%%%%% - Empty actions --> \SKIP
\newcommand{\B@eventinline}[7][]{
  \def\evt@sts{#1}
  \def\evt@name{#2}
  \def\evt@absevts{#3}
  \def\evt@pars{#4}
  \def\evt@grds{#5}
  \def\evt@wits{#6}
  \def\evt@acts{#7}
  %% Ignore convergence status
  \def\pretty@sts{}
  % Pretty-print abstract events
  \ifx\evt@absevts\@empty
  \def\pretty@absevts{}
  \else
  \def\pretty@absevts{\Brefines~\evt@absevts~}
  \fi
  % Pretty-print parameters
  \ifx\evt@pars\@empty
  \def\pretty@pars{}
  \else
  \def\pretty@pars{\Bany~\evt@pars~\Bwhere~}
  \fi
  % Pretty-print guards
  \ifx\evt@grds\@empty
  \def\pretty@grds{}
  \else
  \def\evt@grds@tmp{
      \evt@grds
  }
  \ifx\evt@pars\@empty
  \def\pretty@grds{
    \Bwhen~
    \evt@grds@tmp~
  }
  \else
  \def\pretty@grds{\evt@grds@tmp~}
  \fi
  \fi
  % Pretty-print witnesses
  \ifx\evt@wits\@empty
  \def\pretty@wits{}
  \else
  \def\pretty@wits{
    \Bwith~
    \evt@wits~
  }
  \fi
  % Pretty-print actions
  \ifx\evt@acts\@empty
  \def\evt@acts{\SKIP}
  \else
  \fi
  \def\evt@acts@tmp{
    \evt@acts
  }
  \def\evt@acts@keyword{\Bthen}
  \ifx\evt@pars\@empty
  \ifx\evt@grds\@empty
  \def\evt@acts@keyword{\Bbegin}
  \else
  \fi
  \else
  \fi
  \def\pretty@acts{
    \evt@acts@keyword~
    \evt@acts@tmp~
  }
  % Really do it now
  \begin{array}{l}
    \Bevt{\evt@name}~\widehat{=}~
    \pretty@sts
    \pretty@absevts
    \pretty@pars
    \pretty@grds
    \pretty@wits
    \pretty@acts
    \Bend
  \end{array}
}

%%%%% INITIALISATION label
\newBevt{init}

%%%%% Pretty print the initialisation: no ‘‘refines’’ clause. no parameters, no
%%%%% guards
%%%%% Arguments:
%%%%% 1. (Newline(\\)-separated) list of assignments.
%%%%%
%%%%% Usage: \init{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{\init}{}{}{}{}{#1}
}
```

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.

Change History

v1.0

General: Initial version 1

v1.0.1

General: Ensure that the keywords, labels are in math-mode 1

v1.1

General: Re-implement how options are defined, added options 'box' 1

v1.1.1

General: Updated documentation . 1