SBMLPkgSpec: A LaTeX Class for SBML Level 3 Package Specification Documents

Michael Hucka
mhucka@caltech.edu
Computing and Mathematical Sciences
California Institute of Technology
Pasadena, CA, USA

5 October 2011

Version 1.0.1

Contents

1 Introduction

The SBMLPKGSPEC document class for LaTeX provides a standard format for SBML Level 3 *package specification* documents. In the paragraphs below, I explain how to use this document class. I assume readers are already familiar with LaTeX. (There are many tutorials available on LaTeX today, and readers should have not trouble finding them on the Internet if they need more information than what is provided here.)

The document before you is itself formatted using the SBMLPKGSPEC class, with a minor change to omit some SBML package-specific text normally placed on the front page. (For example, the front page of this document does not announce it is an "SBML Level 3 Package Specification", nor does it provide a package URL and other similar information.) In all other respects, "what you see is what you get"—this is the appearance of an SBML Level 3 package specification document when it is formatted with SBMLPKGSPEC.

Incidentally, in the rest of this document, I usually refer to the SBMLPKGSPEC class as simply "SBMLPKGSPEC".

1.1 Prerequisites and installation

The SBMLPKGSPEC class itself consists of only a single file, `sbmlpkgspec.cls`. It comes with some accompanying documentation (specifically, the file you are reading, along with the source files to produce it), but the actual class definition is in the file `sbmlpkgspec.cls`. Its use should require only a recent and relatively complete installation of LaTeX 2e. I developed and tested SBMLPKGSPEC with the TeX Live 2009 distribution on a Mac OS X 10.6.x system. Table 1 lists the Internet locations from where you may obtain SBMLPKGSPEC.

Item	Location
Distribution archive	`http://sourceforge.net/project/sbml/files/specifications/tex`
Web page	`http://sbml.org/Documents/Specifications/LaTeX`
Source tree (SVN)	`https://sbml.svn.sourceforge.net/svnroot/sbml/trunk/project/tex`

Table 1: *Where to find* SBMLPKGSPEC *on the Internet.*

To use SBMLPKGSPEC, you will need to inform your copy of LaTeX 2ε where to find `sbmlpkgspec.cls`. This can be done in a variety of ways. Here are two common ones:

- *Per-document installation.* This is arguably the simplest approach. Download SBMLPKGSPEC from a distribution site (see Table 1), copy the contents (specifically, `sbmlpkgspec.cls` and the folder named "`logos`") to the folder where you keep the other files for the SBML Level 3 package specification you are authoring, and you are done. The next time you run LaTeX in that folder (assuming you declare the document class as explained in Section 2), it will find the `.cls` file in the current directory and be on its merry way.
- *"Central" installation.* In this approach, you install `sbmlpkgspec.cls` in a folder where you keep other LaTeX class files, and configure your copy of LaTeX to find things there. Configuring a TeX system in this way on Unix-type systems (Linux, etc.) usually requires setting the environment variable `TEXINPUTS` and possibly others. Please consult the documentation for your TeX installation to determine how to do this.

1.2 Special notation in this document

☞ Some paragraphs in this document include a hand pointer in the left margin (illustrated at the left). These are meant to call attention to paragraphs containing significant points that may be too easily missed during the first reading. Readers may wish to revisit those paragraphs once they are actually using SBMLPKGSPEC in practice.

1.3 Where to send bug reports and Feedback

Please report problems you encounter with SBMLPKGSPEC. You can contact the author directly, at the email address given on the cover page, or you can file a bug report using the tracker at `http://sbml.org/issue-tracker`.

2 Creating documents with SBMLPkgSpec

This section provides a summary of the main features and capabilities of SBMLPkgSpec, and serves as a guide to getting started with this LaTeX class.

2.1 Basic document structure

The following fragment illustrates the basic structure of a simple input file.

```
\documentclass{sbmlpackage}
\begin{document}

\packageTitle{Example}
\packageVersion{Version 1 (Draft)}
\packageVersionDate{14 August 2011}
\packageGeneralURL{http://sbml.org/Documents/Specifications/Example}
\packageThisVersionURL{http://sbml.org/Documents/Specifications/Example_14_August_2011}

\author{Michael Hucka\\[0.25em]
  \mailto{mhucka@caltech.edu}\\[0.25em]
  Computing and Mathematical Sciences\\
  California Institute of Technology\\
  Pasadena, CA, USA
}

\maketitlepage
\maketableofcontents

\section{...}
...
\end{document}
```

The fragment above illustrates the general structure expected by SBMLPkgSpec. First, several commands all beginning with the characters `\package` set various internal variables that are used by SBMLPkgSpec to produce the final package specification document. For example, there is a title for the package (`\packageTitle{}`), a version number for the package (`\packageVersion{}`), and more. Next, the author is declared. After that, the commands `\maketitlepage` and `\maketableofcontents` instruct LaTeX to produce a title page and table of contents, respectively. Then comes the real body of the document, with section headings and so on. Finally, the document is closed with the standard LaTeX command `\end{document}`.

☞ If your document is a draft version, make sure to add the special argument `[draftspec]` to the `\documentclass` command. This causes the front page of your document to have the word "DRAFT" printed on it in large gray type, and for every page footer to mention "(DRAFT)" in it. Here is an example of how to use this option:

```
\documentclass[draftspec]{sbmlpackage}
\begin{document}
...
```

Not shown here, but useful to know, is that SBMLPkgSpec provides an additional command for putting a prominent notice on the front page. Writing `\notice{`*text*`}` will put *text* in a red box in the middle of the front page.

SBMLPkgSpec does not define special commands for formatting author information beyond the `\author` command (which is actually a standard LaTeX command from the `article` document class). It does, however, provide the command `\mailto`, which is designed to turn email addresses into `mailto:` hyperlinks. Any additional formatting of author and affiliation information is up to you, using standard LaTeX commands. Of course, there are many times when multiple authors *are* involved, so it is useful to have an approach for handling that situation. An easy approach is to embed a `tabular` environment inside the `\author` command. The following is an example taken directly from an actual SBML Level 3 package specification document:

```
\author{%
  \begin{tabular}{c>{\hspace{20pt}}c}
    Lucian P. Smith                     & Michael Hucka\\[0.25em]
    \mailto{lpsmith@u.washington.edu}   & \mailto{mhucka@caltech.edu}\\[0.25em]
    Department of Bioengineering        & Computing and Mathematical Sciences\\
    University of Washington            & California Institute of Technology\\
    Seattle, WA, US                     & Pasadena, CA, US\\
  \end{tabular}
}
```

☞ A final point about SBMLPKGSPEC is worth mentioning right at the outset. When you run LaTeX (typically using the `pdflatex` variant) and look at the output, you will often find that section references, page references, and line numbers do not get refreshed correctly after one invocation of the command. With SBMLPKGSPEC, you will typically have to run the command not twice, but *three* times, to get the correct, final numbers, because it uses the `varioref` package. This is typically not a problem during actual writing; the edit-run-preview cycle is such that you usually only care to see the results of content changes, and for that, running LaTeX only once is enough, even if it leaves reference unadjusted. However, when you *are* interested in checking (e.g.) figure and table references, then it is important to keep in mind the fact that you need to run LaTeX three times in succession to get all the reference updates to propagate through. The rule of thumb is: if you run LaTeX and the references look wrong, run it again.

2.2 Tables and figures

SBMLPKGSPEC preloads the LaTeX `graphicx` and `xcolor` packages, giving authors immediate access to the functionality of these extensions without having to include them manually. For example, if you had an illustration in a file named "`example.pdf`", the following fragment would generate a simple figure containing it:

```
\begin{figure}
  \includegraphics{example}
  \caption{The figure caption.}
  \label{fig:example}
\end{figure}
```

The LaTeX `graphicx` package is extremely powerful; it offers many options for controlling the size/scale and other characteristics of graphics files. Please refer to its documentation for help on how to use it fully.

To produce the most pleasing-looking documents, create your figures using 8 point Helvetica for the text font. SBMLPKGSPEC redefines figures and tables to use Helvetica as the font family and an 8 point size by default, so creating included figures with match characteristics will lead to more consistent-looking documents. The stylistic choices here were made not only for aesthetic reasons; the tighter letter spacing of the sans serif font, and the smaller font size, makes it easier for authors to fit material into tables and figures. The specific choice of Helvetica is also driven in part by consideration of the tools available to authors. In particular, it is today common to find online drawing tools that offer Helvetica as a font choice.

SBMLPKGSPEC also preloads the `booktabs` package. This provides `\toprule`, `\midrule` and `\bottomrule` (among others), which can be used to produce attractive tables. The following text is what produced Figure 1 on page 3:

```
\begin{table}[hb]
  \begin{edtable}{tabular}{ll}                % From the lineno package; see text, Section 2.2
    \toprule                                  % From booktabs -- generates line at top
    \textbf{Item}        & \textbf{Location} \\
    \midrule                                  % From booktabs -- generates middle line
    Distribution archive & \url{\distURL} \\
    Web page             & \url{\webURL} \\
    Source tree (SVN)    & \url{\srcURL} \\
    \bottomrule                               % From booktabs -- generates line at bottom
  \end{edtable}
  \caption{Where to find \textsc{sbmlpackage} on the Internet.}
  \label{where}
\end{table}
```

The table and figure environments in SBMLPKGSPEC have been redefined to place their content inside a LaTeX `centering` environment. This causes the content to be centered on the page, and consequently, you do not need to add centering commands to your floats.

To refer to figures, tables and other elements in the document, please use the special commands listed in Table 2 instead of writing the usual idioms "Figure~\ref{...}". The commands in Table 2 will not only produce content and page references that are automatically hyperlinked to the appropriate locations in the finished document; they will also take care of adding the tie for you, and they use the package `varioref` instead of the regular LaTeX `ref` command to produce more informative references to figures and tables.

Command	Purpose	Output
`\fig{`*label*`}`	Figure reference	Figure X
`\tab{`*label*`}`	Table reference	Table X
`\sect{`*label*`}`	Section reference	Section X

Table 2: *Commands for referring to figures and other entities in an* SBMLPKGSPEC *document. Use the commands with an argument consisting of the label being referenced. For example:* `\fig{myfig}`.

Finally, in the case of long tables, readability is often enhanced by adding a background color to every other row. Once again, SBMLPKGSPEC preloads a LaTeX package (in this case, `xcolor` with the `[table]` option) that provides a convenient facility for automatically coloring alternate rows in a table. Although many variations are possible, for consistency between SBML package specification documents, I recommend using one in particular:

```
\rowcolors{2}{sbmlrowgray}{}
```

Simply insert the literal line above after the opening `\begin{table}` of your table, and proceed as usual. The resulting table will have a light gray background color for every other row. This is demonstrated in Figure 5 on page 9, which was produced using the following opening sequence:

```
\begin{table}[hbt]
  \rowcolors{2}{sbmlrowgray}{}
  \begin{edtable}{tabular}{ll}
  ...
```

☞ Note that tables are *not* defined by SBMLPKGSPEC to use alternate-row background coloring by default, because in some situations (such as short tables, or tables containing color), it is unnecessary and distracting. You must add the `\rowcolors` manually, when it's appropriate.

2.3 Hyperlinks

In the last example in the previous section, you may have noticed the use of the command `\url{}`. This command comes as part of the LaTeX `hyperref` package, and like the other packages mentioned in this section, it is also preloaded by SBMLPKGSPEC. It provides a number of facilities that are used to implement features of SBMLPKGSPEC. Table 3 lists some commands you may find useful in writing SBML specification documents.

Command	Purpose
`\url{`*URL*`}`	Produce a hyperlinked reference to *URL*
`\nolinkurl{`*URL*`}`	Format *URL* in the same way as `\url`, without making it a hyperlink
`\href{`*URL*`}{`*text*`}`	Make *text* a hyperlink to *URL*

Table 3: *Commands provided by* `hyperref` *for creating hyperlinks.*

2.4 Examples and literal text

A document about file formats and programming often includes passages meant to be literal text. Conventionally, these are typeset in a monospaced type face resembling the output of typewriter. There are two ways of accomplishing this for SBML package specifications. The first is to use LaTeX's standard `\texttt{`*text*`}` command. This causes "*text*" to be output in a fixed-width font, like so: "`text`".

The second method is to use an environment defined by SBMLPkgSpec and implemented using the LaTeX package `listings`. This environment is called `example`. Wrapping any text with `\begin{example}` and `\end{example}` will cause the text to be output by itself with a gray box behind it, as in this example:

```
This is an example of content placed within an "example" environment.
```

The `example` environment is particularly powerful. Anything placed inside it will be taken literally—even LaTeX commands will be ignored, except for `\end{example}`. In fact, the entire contents of the document fragment shown in Section 2.1, including the `\begin{document}` and `\end{document}`, were all left unchanged within the `example` environment used to produce the example.

Of course, sometimes you *do* want an embedded LaTeX command to be interpreted inside the `example` environment. For those situations, surround the LaTeX sequence with vertical bar (`|`) characters. The vertical bar is defined by SBMLPkgSpec as the escape character for the `example` environment.

Finally, for those cases when it is more convenient to put the example contents in a separate file, SBMLPkgSpec provides the command `\exampleFile{`*file*`}`. It will insert the contents of *file* at the point where it is invoked in the text, and format the contents in the same style as the `example` environment.

2.5 Color

To help increase the style consistency between SBML specification documents, SBMLPkgSpec defines a number of custom color names that authors may use with commands such as `color`, `colorbox`, etc. Table 4 provides a list of these color names.

Color name	Color sample	RGB color value		
`sbmlblue`		red = 0.08	blue = 0.51	green = 0.77
`sbmlgray`		red = 0.7	blue = 0.7	green = 0.7
`sbmlrowgray`		red = 0.94	blue = 0.94	green = 0.94
`sbmlchangedcolor`		red = 0.69	blue = 0.19	green = 0.376
`extremelylightgray`		red = 0.97	blue = 0.97	green = 0.97
`verylightgray`		red = 0.9	blue = 0.9	green = 0.9
`lightgray`		red = 0.8	blue = 0.8	green = 0.8
`mediumgray`		red = 0.5	blue = 0.5	green = 0.5
`darkgray`		red = 0.3	blue = 0.3	green = 0.3
`almostblack`		red = 0.2	blue = 0.2	green = 0.2
`darkblue`		red = 0.1	blue = 0.4	green = 0.55
`lightyellow`		red = 0.995	blue = 0.97	green = 0.93
`mediumgreen`		red = 0.1	blue = 0.6	green = 0.3

Table 4: *Color names defined by* SBMLPkgSpec. *These names may be used in addition to the names of colors defined by the LaTeX package* `xcolor`.

Some of the colors in Table 4 are used by SBMLPkgSpec itself, for some of the design elements such as section dividers and background colors. Others are colors that sometimes prove useful in different contexts of a specification document, and still others are colors that were used in past SBML documentation and are carried over in case they prove useful again in the future.

When a specification document is a revision of a previous document, it is the convention in SBML documentation to indicate text changes by coloring them in red. For this purpose, SBMLPkgSpec defines a command and an

environment. The command `\changed{`*text*`}` causes *text* to be written in red (or more precisely, the color defined by `\sbmlchangedcolor`, which is a dark purple red), like this. It can be used for long stretches of text and include embedded spaces and formatting commands. Alternatively, for coloring even longer stretches of text and multiple paragraphs, you may prefer to use the environment `blockChanged`.

2.6 Commands for SBML constructs

SBML defines a number of commands for referring to objects defined in the main SBML specifications, such as **Species**, **Reaction**, etc. Within a given main SBML specification document—which are all in PDF format—the mention of an object name is hyperlinked to the definition of that object elsewhere in the document, such that clicking on the name causes the PDF reading program to jump to the definition in the file. Unfortunately, while it is technically possible to hyperlink from SBML package documents to specific parts within an external PDF file located somewhere on the Internet, the result is more confusing and annoying than helpful.

SBMLPKGSPEC therefore defines two sets of commands: one set predefines each of the SBML Level 3 Core object names, but without hyperlinking, and the second set lets package authors define new object names, with hyperlinking. The result is that references to Core SBML objects are displayed in black and without linking, while package objects appear in blue, as hyperlinks. The linking is implemented using the standard LaTeX `hyperref` package.

In addition to these commands, SBMLPKGSPEC provides other commands for typesetting primitive type names (such as `SId`, `double`, and so on) and other entities and XML fragments. The need for using these occurs routinely when writing SBML specifications. The commands are described in Section 2.6.3 and Section 2.6.4.

2.6.1 Predefined SBML Core object and type names

Table 5 on the next page lists the commands to typeset the names of SBML Level 3 Core objects. They are designed to be as convenient to use as possible, requiring only one additional character (i.e., the leading backslash character) to be typed beyond the name of the object itself.

2.6.2 Defining new object names and types in package specifications

When an SBML package specification document defines new object classes, it is useful to make all mentions of the class name be hyperlinks to the class definition in the document. To this end, SBMLPKGSPEC provides commands that can be used to create new LaTeX commands to define hyperlinked name references. The purpose of these commands is to let package authors define commands of the form `\ObjectName` that print the name `ObjectName` and simultaneously make it a hyperlink to the sections in the document where **ObjectName** is defined. They are best used conjunction with LaTeX's `\newcommand` command:

`\defRef{`*name*`}{`*section*`}`
: Create a hyperlinked reference to the section labeled *section* and call it *name*. The reference appears at the point in the text where the `\defRef` command is invoked. This command is intended for references to regular (not abstract) classes; see the next command for abstract classes.

`\absDefRef{`*name*`}{`*section*`}`
: Create a hyperlinked reference to the section labeled *section* and call it *name*. The reference appears at the point in the text where the `\defRef` command is invoked. This command is intended for references to abstract classes; see the previous command for the corresponding command for non-abstract classes.

The following is an example of how the commands above may be used; this is taken straight from the source files of the SBML Level 3 Version 1 Core specification document:

```
\newcommand{\SBase}{\absDefRef{SBase}{sec:sbase}\xspace}
\newcommand{\SBML} {\defRef{SBML}{sec:sbml}\xspace}
\newcommand{\Model}{\defRef{Model}{sec:model}\xspace}
```

The section labels "`sec:base`", "`sec:sbml`", etc., are defined using LaTeX's `\label` command at the beginning of each section where the corresponding objects are defined. (The command `\xspace` is discussed in Section 3.2.)

Command	Object
\AlgebraicRule	**AlgebraicRule**
\Annotation	**Annotation**
\AssignmentRule	**AssignmentRule**
\Compartment	**Compartment**
\Constraint	**Constraint**
\Delay	**Delay**
\EventAssignment	**EventAssignment**
\Event	**Event**
\FunctionDefinition	**FunctionDefinition**
\InitialAssignment	**InitialAssignment**
\KineticLaw	**KineticLaw**
\ListOfCompartments	**ListOfCompartments**
\ListOfConstraints	**ListOfConstraints**
\ListOfEventAssignments	**ListOfEventAssignments**
\ListOfEvents	**ListOfEvents**
\ListOfFunctionDefinitions	**ListOfFunctionDefinitions**
\ListOfInitialAssignments	**ListOfInitialAssignments**
\ListOfLocalParameters	**ListOfLocalParameters**
\ListOfModifierSpeciesReferences	**ListOfModifierSpeciesReferences**
\ListOfPackages	**ListOfPackages**
\ListOfParameters	**ListOfParameters**
\ListOfReactions	**ListOfReactions**
\ListOfRules	**ListOfRules**
\ListOfSpeciesReferences	**ListOfSpeciesReferences**
\ListOfSpecies	**ListOfSpecies**
\ListOfUnitDefinitions	**ListOfUnitDefinitions**
\ListOfUnits	**ListOfUnits**
\LocalParameter	**LocalParameter**
\Message	**Message**
\Model	**Model**
\ModifierSpeciesReference	**ModifierSpeciesReference**
\Notes	**Notes**
\Package	**Package**
\Parameter	**Parameter**
\Priority	**Priority**
\RateRule	**RateRule**
\Reaction	**Reaction**
\Rule	**Rule**
\SBML	**SBML**
\SBase	***SBase***
\SimpleSpeciesReference	**SimpleSpeciesReference**
\SpeciesReference	**SpeciesReference**
\Species	**Species**
\StoichiometryMath	**StoichiometryMath**
\Trigger	**Trigger**
\UnitDefinition	**UnitDefinition**
\Unit	**Unit**

Table 5: *Commands for the names of object classes defined in the SBML Level 3 Core specification.*

Sometimes it is desirable to write the names of object classes without introducing hyperlinks. A common situation is when mentioning the names of the object classes or types in section headings. In these situations, instead of using the `\ObjectName` commands, you may use the following commands provided by SBMLPKGSPEC:

`\class{`*name*`}`
: Typesets *name* in the same font style as used by `\defRef{`*name*`}{`*section*`}`, without a hyperlink.

`\abstractclass{`*name*`}`
: Typesets *name* in the same font style as used by `\absDefRef{`*name*`}{`*section*`}`, without a hyperlink.

2.6.3 Commands for formatting the names of primitive data types

A convention that has evolved over the years of writing SBML specifications is to typeset the names of primitive data types (such as `SId`) in a monospaced, typewriter-like type face, without hyperlinks. SBML packages may define their own new primitive types. To format the names of these types in a style consistent with the SBML Level 3 Core specification document, package authors should use the `\primtype` and `\primtypeNC` commands:

`\primtype{`*name*`}`
: Typesets the name of a primitive data type *name*. SBML Level 3 Core also defines and uses a number of primitive data types, but these do not have separate commands in SBMLPKGSPEC. Instead, they should be written using the command `\primtype{`*type*`}`, where *type* is one of the following names:

`boolean`	`int`	`SId`	`UnitSId`
`double`	`positiveInteger`	`SIdRef`	`UnitSIdRef`
`ID`	`SBOTerm`	`string`	

`\primtypeNC{`*name*`}`
: Like `\primtype`, but does not force the color of the text to be black. The main use of this variant of the command is when writing the names of primitive types in the arguments to LaTeX sectioning commands (e.g., `\section`, `\subsection` and the like), to avoid the change in color that would otherwise occur in the document's table of contents. (The `\primtype` command sets the color of the text to pure black, to make the text stand out more in the document body. The change of color would occur because the entries in the table of contents are all hyperlinks to the beginning of the sections, and hyperlinks are colored blue.)

For example, the command sequence "`\primtype{SId}`" written somewhere in the body of a document produces "`SId`" in the formatted output. On the other hand, the following illustrates how to write the name of a type in a section heading:

```
\subsection{This is the documentation for \primtypeNC{MySpecialSId}}
```

2.6.4 Commands for formatting in-text XML descriptions

In addition to formatting the names of SBML object classes and primitive data types, SBMLPKGSPEC provides commands for formatting text meant to be literal examples of XML. These special commands are provided so in-text descriptions of XML constructs can be formatted in a way and attractive way:

`\token{`*text*`}`
: Formats literal XML tokens, such as attribute names. Do not use this to format text with embedded spaces; instead, use multiple `\token` commands separated by spaces.

`\val{`*text*`}`
: Format the value of an attribute. This is essentially `\token` but surrounded by double quotes.

Here is an example of using the commands above. Suppose you wanted to write that a certain attribute named `attrib1` is to be assigned the value "`10`". The following input text,

```
\token{attrib1}=\val{10}
```

will result in the following formatted output:

`attrib1`="`10`"

Note that the command `\token` and others above are only meant to be used for single in-text mentions of tokens and XML attribute-value pairs. They are not suitable for longer content; for those cases, use the `example` environment described in Section 2.4.

2.7 Line numbers

SBMLPkgSpec preloads the LaTeX package `lineno` and configures it to produce the line numbers that you see in the right column of every page. These line numbers are important for specification documents because they allow discussion and bug reports to refer to specific portions of the text more easily. Crucially, lists of errata and other issues recognized *after* a specification is released need ways of referring to precise locations in the specification document, and line numbers are invaluable for that purpose.

☞ For the most part, you needn't do anything to get line numbers to be added to your document. There are exceptions: certain content such as tabular material and floats are not handled by `lineno` very well, and require manual intervention. Some cases cannot be fixed at all, notably figures incorporated from external files, but tabular material is fixable. To get line numbers to be displayed in tables, you will need to wrap all uses of `tabular` with the special environment `edtable`. The basic idiom is the following:

```
\begin{edtable}{tabular}{...normal tabular column specifiers...}
  ...tabular content...
\end{edtable}
```

The `edtable` environment is able to wrap a number of standard LaTeX environments, of which `tabular` is probably the most useful for most kinds of tables. Practical examples of using `edtable` appear below in other sections.

2.8 SBML validation rules

A convention developed while writing the core SBML specification documents is to define validation and consistency rules that must or should be satisfied in order for a document to conform to the specification. SBML package specifications should likewise follow this convetion and define their own validation and consistency rules. The convention identifies different degrees of rule strictness. Formally, the differences are expressed in the statement of a rule: either a rule states that a condition *must* be true, or a rule states that it *should* be true. Rules of the former kind are strict SBML validation rules—a model encoded in SBML must conform to all of them in order to be considered valid. Rules of the latter kind are consistency rules.

To help highlight these differences, the SBML specification documents and SBMLPkgSpec provide commands to format the three kinds of rules, with three different symbols:

- ☑ A checked box indicates a *requirement* for SBML conformance. If a model does not follow this rule, it does not conform to the specification. (Mnemonic intention behind the choice of symbol: "This must be checked.")
- ▲ A triangle indicates a *recommendation* for model consistency. If a model does not follow this rule, it is not considered strictly invalid as far as the specification is concerned; however, it indicates that the model contains a physical or conceptual inconsistency. (Mnemonic intention behind the choice of symbol: "This is a cause for warning.")
- ★ A star indicates a strong recommendation for good modeling practice. This rule is not strictly a matter of SBML encoding, but the recommendation comes from logical reasoning. As in the previous case, if a model does not follow this rule, it is not strictly considered an invalid SBML encoding. (Mnemonic intention behind the choice of symbol: "You're a star if you heed this.")

SBMLPkgSpec defines three commands for writing these rules in SBML package specifications documents:

`\validRule{`*number*`}{`*text*`}`
: Format *number* as a validation rule with the description *text*.

`\consistencyRule{`*number*`}{`*text*`}`
: Format *number* as a consistency rule with the description *text*.

`\modelingRule{`*number*`}{`*text*`}`
: Format *number* as a modeling rule with the description *text*.

3 Additional features of SBMLPkgSpec

This section describes some additional aspects of SBMLPkgSpec.

3.1 Options understood by SBMLPkgSpec

A number of options may be given to SBMLPkgSpec in the `\documentclass` command that begins a document. Here follows a complete list:

`draftspec`
: This option causes the front page of the document to contain the word "DRAFT" in large gray letters, and the footer of every page of the document to contain the word "(DRAFT)". Authors should use this option until such time as the specification document is considered a release candidate or a final release.

`finalspec`
: (Default) This option causes the large "DRAFT" on the front page and "(DRAFT)" in the footers to be omitted. It is the opposite of the `draftspec` option.

`toc`
: (Default) This option causes SBMLPkgSpec to include a table of contents as the second page of the document. Whether the table has one wide column or two narrow columns is then controlled by the options `twocolumntoc` and `singlecolumntoc`, described below.

`notoc`
: This option causes the table of contents to be omitted. (It is unclear under what circumstances one would want to omit the table of contents, but since some other LaTeX classes include this feature, SBMLPkgSpec follows suit.)

`twocolumntoc`
: This option causes SBMLPkgSpec to produce a two-column table of contents rather than the default one-column version. (That is, unless the `notoc` option is also given, in which case, no table of contents is produced.) This is useful when a document is long, with many sections, because the two-column version is more compact. (However, beware that since the columns are narrower than the single-column version, long section names may become wrapped, leading to an aesthetically less pleasing result. If you need to use two column output, you may alway want to examine whether you can shorten your section titles.)

`singlecolumntoc`
: (Default) This option is the opposite of `twocolumntoc`; it causes SBMLPkgSpec to produce a single column table of contents as the second page of the document. (Again, if `notoc` is also given, then no table of contents is produced at all.)

3.2 Notable LaTeX packages preloaded by SBMLPkgSpec

As mentioned above, SBMLPkgSpec preloads many common LaTeX packages. Some are used to implement the features of SBMLPkgSpec itself; others are preloaded so that authors do not have to load them explicitly. Knowing what SBMLPkgSpec provides upfront also makes it easier to explain how to write SBML specification documents.

The following is a list of the LaTeX packages preloaded by SBMLPkgSpec. It is beyond the scope of this document to explain their features and capabilities in detail; readers are urged to consult the documentation for each one to learn more about them. Documentation is available at CTAN (`http://www.tug.org/ctan.html`).

- `amssymb`: This package defines many symbols and special characters. In SBMLPkgSpec, it is used to get the symbols defined by the validation rule commands `\validRule`, `\consistencyRule` and `\modelingRule` described in Section 2.8.

- `array`: This provides a new and extended implementation of the LaTeX `array` environment and `tabular` environments. It is particularly useful for the column formatting options it provides for the `tabular` environment.
- `booktabs`: Mentioned in Section 2.2, this LaTeX package defines the commands `\toprule`, `\bottomrule`, and `\midrule` for produced more attractive and professional-looking tables.
- `enumitem`: This relatively new package adds facilities for more easily adjusting the look of LaTeX list environments, including the `description` environment—something that is difficult to do in plain LaTeX. (Oh sure, there *are* variables and commands for doing it in LaTeX, but they are limited and sometimes produce unexpected consequences elsewhere in a document.)
- `graphicx`: A powerful and rich system for working with external graphics files in LaTeX documents.
- `hyperref`: Mentioned in Section 2.3, this package defines commands for creating hyperlinks within and between documents. SBMLPkgSpec uses it to define commands such as `\fig` and `\tab`.
- `lineno`: Used by SBMLPkgSpec to implement line numbers on the page. As mentioned in Section 2.7, not all line numbering can be accomplished completely without user intervention; in some cases, authors must add special commands to get line numbers into content on the page.
- `listings`: Used by SBMLPkgSpec to implement the `example` environment and `\exampleFile` command.
- `multicol`: This provides an environment for putting multiple columns of text on a page. In SBMLPkgSpec, it is used to produce two-column table of contents when the `[twocolumntoc]` option is given as described in Section 3.1.
- `natbib`: This reimplements the normal LaTeX `\cite` command to work with author-year citations and add a number of useful features. It provides variants of `\cite` such as `\citep` and `\citeauthor`. The features of `natbib` are worth learning and using.
- `varioref`: This defines commands such as `\vref`, which is similar to `\ref` but includes a page reference such as "on the next page" or "on page 27" when the corresponding `\label` is not on the same page. (The way that the page references are generated requires multiple passes of running LaTeX, which is one of the reasons why using SBMLPkgSpec requires 2–3 runs of LaTeX to generate final cross-references.)
- `wasysym`: Similar to `amssymb`, this LaTeX package defines another set of symbols and special characters.
- `xcolor`: This package defines a large number of color names and associated commands.
- `xspace`: A marvelous little extension that provides just one very useful command, `\xspace`. The `\xspace` command can be used at the end of a macro designed to be expanded into text; it adds a space unless the macro is followed by a punctuation symbol. This makes it possible to invoke the macro in text without adding an empty `{}` pair after it—something that would otherwise be a problem because LaTeX would consume the space character following the macro invocation, leading to a missing space in the final output. The commands such as `\SBML` are defined using `\xspace`, so that in running text, you can write "one two `\SBML` three four" and it will produce "one two **SBML** three four" rather than "one two **SBML**three four".

4 Acknowledgments

This work was made possible by grant R01 GM070923 from the NIH National Institute of General Medical Sciences (USA) for continued development and support of SBML and related software infrastructure.