Preparing Z specifications using LaTeX mark-up

/Tutorial guides

This tutorial won't teach you how to write Z, but if you already have some idea on that, then it will show you how to mark-up that Z so that cadiz will be able to process it and LaTeX will be able to typeset it. Specific examples are given here; for general definitions, see the reference manual.

1. Contents of this page

2. Choosing a style file

There is a choice of two style files for typesetting a Z specification with LaTeX: `cadiZ.sty` and `ltcadiz.sty`. New users should probably choose `ltcadiz.sty`. Both define environments for drawing outlines around Z paragraphs, and define commands for typesetting mathematical symbols. The mark-ups they define are the same (inevitably with an exception). The differences between the style files are as follows.

The `cadiZ.sty` package sets the mathematical symbols using the `oxsy` fonts, which produce good results in many sizes, but may not be available to you. (I believe they are distributed only with the `fuzz` typechecker). It defines commands

for typesetting Z mathematical symbols, but no others. Paragraph outlines are scaled to fit around the formal text that they contain. Paragraphs that are longer than a page are divided automatically. Formal text is typeset in an erect font.

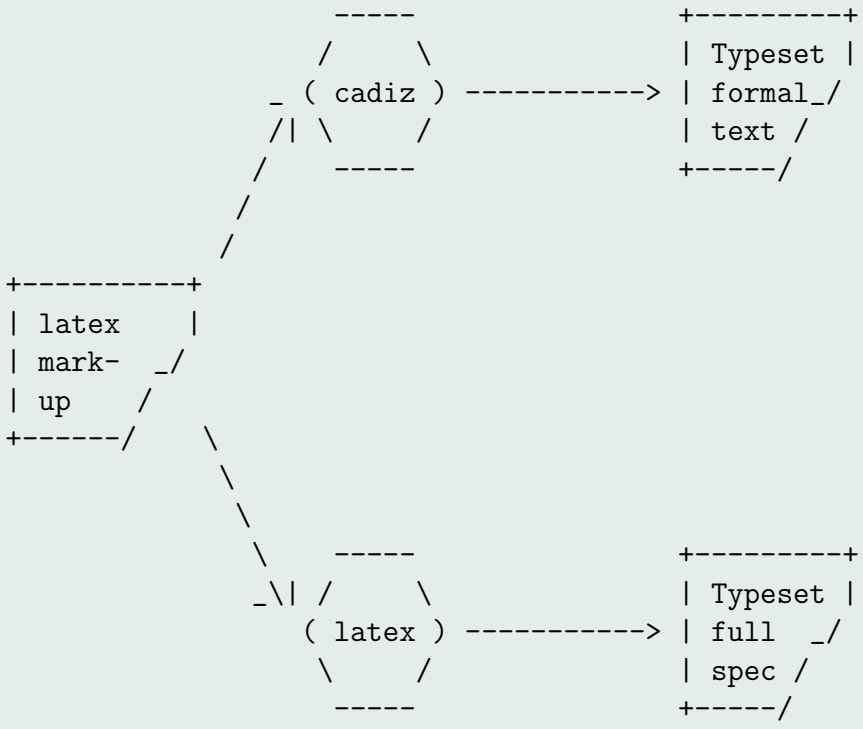
The `ltcadiz.sty` package extends the `oz.sty` package, which should come with CADiℤ. However, note that `ltcadiz.sty` requires some things to be marked-up differently from `oz.sty` in order to be checked by `cadiz` and in order that `cadiZ.sty` can be substituted. The `ltcadiz.sty` package sets the mathematical symbols by overstriking existing characters, which works fine in the particular font-size combination for which it was designed, but produces poor results otherwise. It defines commands for typesetting Z mathematical symbols and many more. Paragraph outlines are always the full width of the page. Paragraphs that are longer than a page fall off the bottom of the page unless the user marks-up where a paragraph should be divided. Formal text is typeset in an italic font.

To use the `ltcadiz.sty` package, for example, your mark-up must have the following structure.

```
\documentclass{article}
\usepackage{ltcadiz}
\begin{document}
Formal & Informal Text
\end{document}
```

3. Typesetting LaTeX mark-up

This section may seem premature here, but it is important to realise that a specification prepared using LaTeX mark-up can be typeset in two different ways, one via cadiz and the other directly using LaTeX.



The cadiz route cannot typeset informal text, because troff is used to generate

the display, and cadiz does not even try to translate arbitrary LaTeX mark-up to troff mark-up.

The LaTeX route is appropriate for the final document, but the cadiz route is necessary for interactive browsing. It is important to use cadiz at some stage, as otherwise the specification would not be typechecked.

The decision about which style file to use affects only the LaTeX route; the cadiz route uses troff, and hence always uses an erect font (Helvetica) with outlines scaled to fit around the formal text.

Precise details of how to use the two routes is given in printing LaTeX specifications on paper.

4. Formal environments

The formal paragraphs of a Z specification are delimited from the informal paragraphs by being enclosed within particular LaTeX environments, i.e. between the commands `\begin{environment}` and `\end{environment}`. The various environments and the kinds of Z paragraphs that are expected to appear within them are as follows.

`\begin{axdef}` ... `\end{axdef}`	Axiomatic description
`\begin{gendef}` ... `\end{gendef}`	Generic axiomatic description
`\begin{schema}` ... `\end{schema}`	[Generic] schema paragraph
`\begin{theorem}` ... `\end{theorem}`	[Generic] conjecture
`\begin{zed}` ... `\end{zed}`	All other non-outlined Z

These environments should be entered only from the top-level document environment, otherwise they might not typeset properly. Only formal Z notation should appear within these environments, otherwise cadiz will not be able to parse their contents.

There is one further formal environment in addition to those above of standard Z.

`\begin{syntax}` ... `\end{syntax}`	All other non-outlined Z

The `syntax` environment is a tabulated version of the `zed` environment, in which columns can be aligned using the `&` character. This effect is achieved only when typesetting via the LaTeX route; cadiz treats `&` as white space in this environment.

The formal environments are illustrated in the example specification below.

5. Informal environments

All the facilities of LaTeX may be exploited for typesetting the informal material between Z paragraphs. In particular, the style files contribute definitions of

additional environments for typesetting formal-like material that is not to be processed by cadiz. As this isn't a LaTeX tutorial, refer to the style files for details.

6. Directives to cadiz

cadiz ignores the informal text in a specification, recognising only the formal Z paragraphs and the directives that affect the tool's behaviour. These directives are all marked-up as individual lines, starting with `%%` at the beginning of the line, and immediately followed by the name of the directive, then any arguments. For example,

```
%%Zinchar \subseteq U+2286
```

directs cadiz to map the LaTeX command `\subseteq` to the UCS character 2286.

The first `%` is sufficient to make LaTeX see the line as a comment and so ignore it. Most directives must be written between formal paragraphs, not within them. cadiz considers any line in informal text beginning with `%%` to be a directive, so don't start your own LaTeX comments like that or cadiz will moan at you.

There are directives for typesetting indexes, turning off typesetting, and so on. A full list of directives is in the reference manual.

7. An example specification

The following example specification is intended to illustrate the LaTeX mark-up that cadiz can recognise and the typesetting of formal text performed by latex. It is not intended to be indicative of good style, but merely to provide examples of LaTeX mark-up and typesetting. The specification is of a bicycle gearing system.

7.1. Begin document

This preamble is necessary for LaTeX (not for cadiz). It does not correspond to anything being typeset.

```
\documentclass{article}
\usepackage{ltcadiz}
\begin{document}
```

7.2. Section heading

section *gearsystem* parents *standard_toolkit*

```
\begin{zsection}
\SECTION gearsystem \parents standard\_toolkit
\end{zsection}
```

7.3. Free types paragraph

$$gearing ::= Single \mid Hub\langle\langle \mathbb{N} \rangle\rangle \mid Derailleurs\langle\langle \mathbb{N} \times \mathbb{N} \rangle\rangle$$

```
\begin{zed}
gearing ::= Single | Hub \ldata \nat \rdata
    | Derailleurs \ldata \nat \cross \nat \rdata
\end{zed}
```

7.4. Horizontal definition paragraph

$$Wellgeared == [gears : gearing \mid gears \in ran\ Derailleurs]$$

```
\begin{zed}
Wellgeared == [ gears : gearing | gears \in \ran Derailleurs ]
\end{zed}
```

7.5. Generic axiomatic description paragraph

$[X, Y]$

$$First : X \times Y \to X$$
$$Second : X \times Y \to Y$$

$$\forall x : X; y : Y \bullet First\ (x, y) = x \land Second\ (x, y) = y$$

```
\begin{gendef}[X,Y]
First : X \cross Y \fun X \\
Second : X \cross Y \fun Y
\where
\forall x : X; y : Y @ \\
\t1     First~(x,y) = x \land \\
\t1     Second~(x,y) = y
\end{gendef}
```

7.6. Schema definition paragraph

Gear

$$Wellgeared$$
$$ingear : \mathbb{N} \times \mathbb{N}$$

$$\exists r, s : \mathbb{N} \mid gears = Derailleurs(r, s) \bullet First\ ingear < r \land Second\ ingear < s$$


```
\begin{schema}{Gear}
Wellgeared \\
ingear : \nat \cross \nat
\where
\exists r, s : \nat | gears = Derailleurs (r,s) @\\
\t1 First~ingear < r \land Second~ingear < s
\end{schema}
```

7.7. Generic schema definition paragraph

$$Bicycle\,[G]$$
$$gears : \mathbb{P}\, G$$

```
\begin{schema}{Bicycle}[G]
gears:\power G
\end{schema}
```

7.8. Axiomatic description paragraph

$$
\begin{array}{|l}
numgears : gearing \to \mathbb{N} \\
\hline
\forall\, g : gearing \bullet \\
\quad (g = Single \Rightarrow numgears\ g = 1) \land \\
\quad (\exists\, x : \mathbb{N} \bullet g = Hub\ x \Rightarrow numgears\ g = x) \land \\
\quad (\exists\, r, s : \mathbb{N} \bullet g = Derailleurs\ (r, s) \Rightarrow numgears\ g = r * s)
\end{array}
$$

```
\begin{axdef}
numgears : gearing \fun \nat
\where
\forall g : gearing @ \\
\t1 (g = Single \implies numgears~g = 1) \land \\
\t1 (\exists x : \nat @ g = Hub~x \implies numgears~g = x) \land \\
\t1 (\exists r, s :\nat @ g = Derailleurs~(r, s) \implies numgears~g = r*
\end{axdef}
```

7.9. Conjecture paragraph

Lemma1 ==

$$\vdash ?\ numgears(Derailleurs(3, 7)) = 21$$

```
\begin{theorem}{Lemma1}
\thrm numgears (Derailleurs (3,7)) = 21
\end{theorem}
```

The argument is the name of the conjecture. A conjecture need not have a name, in which case the argument must be some white space, not omitted. There is a tutorial on proving conjectures using cadiz.

7.10. End document

This postamble is necessary for LaTeX (not for cadiz). It does not correspond to anything being typeset.

```
\end{document}
```

8. Mathematics within paragraphs

The mark-up of specific mathematical symbols may be found by referring to the following tables in the reference manual,

- standard core Z notation
- non-standard extensions
- toolkit symbols

It can be helpful to bear in mind that the LaTeX mark-up is first converted to a sequence of UCS characters, and that it is that sequence which cadiz lexes

and parses. To be sure that some characters are lexed as separate lexical tokens instead of being taken together as a single lexical token, you sometimes have to ask for space. For example, in the juxtaposed application $f\ x$ of a function f (not an operator) to an argument x, the space is necessary to avoid this being lexed as fx. Spaces, tabs and newlines in the mark-up do not generate any characters; LaTeX commands like ~ and \t1 generate *SPACE*, and \\ and \also generate *NLCHAR*.

When cadiz rejects your mark-up and you need to work out why, it is worth considering the sequence of UCS characters to which cadiz converts your mark-up. The prep tool may be helpful, as it does the same translation to UCS characters as cadiz but then just displays them in a window.

When you want to introduce a new mathematical symbol of your own, you will need not just its Z operator template, Z definition and a LaTeX rendering macro, but also a mark-up directive to tell cadiz how to convert it to a sequence of UCS characters. Here are some examples of mark-up directives for symbols that are single UCS characters.

```
%%Zchar \Delta U+0394
%%Zchar \arithmos U-0001D538
%%Zinchar \rel U+2194
%%Zprechar \finset U-0001D53D
%%Zpostchar \rangle U+3009
```

Notice that four-character encodings are used for symbols on the basic multilingual plane of UCS, and eight-character encodings are used for other symbols.

Also, the directive name reflects the fixity of the symbol, and affects the generation of spaces around the conversion. A similar choice of directives exists for multiple character symbols.

```
%%Zinword \partition partition
%%Zpreword \dom dom
%%Zpostword \star ^*
```

When defining rendering macros for operator symbols, you should be more precise about how much space surrounds the symbol: relation symbols should have more space than function symbols, to reflect their lower precedence. It is your responsibility to ensure that the rendering by LaTeX is consistent with the conversion by CADiZ, as required by the Z standard.

Subscripts are marked-up using the usual LaTeX mark-up of an _ character followed by either a single character or a braced sequence of characters. Superscripts are marked-up using the usual LaTeX mark-up of an ^ character followed by either a single character or a braced sequence of characters.

IT 22-Jan-2002