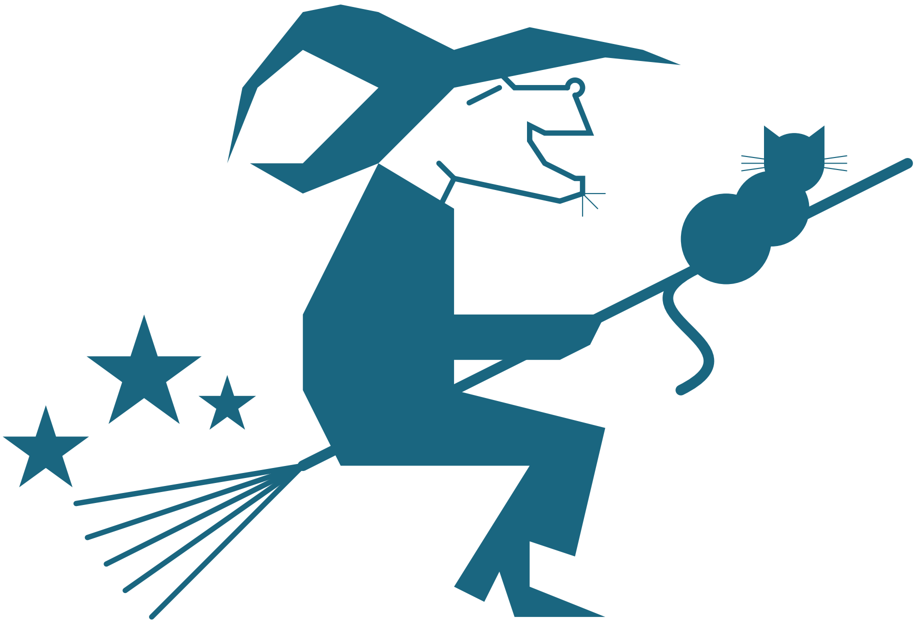
# The **halloweenmath** package: scary and creepy math symbols with $\mathcal{AMS}$-LaTeX integration*

G. Mezzetti

November 1, 2019

**Abstract**

This document describes a LaTeX package that originates from a question asked for enjoyment on *TeX – LaTeX Stack Exchange* (see [1]) by the user cfr; this package defines a handful of commands for typesetting mathematical symbols of various kinds (including "large" operators, extensible arrow-like relations, and growing arrow-like math accents) that all draw from the classic Halloween-related iconography (pumpkins, witches, ghosts, cats, and so on) while being, at the same time, seamlessly integrated within the rest of the mathematics produced by LaTeX/$\mathcal{AMS}$-LaTeX.



---

[*] This file has version 0.11, release date 2019/11/01; last documentation update: 2019/11/01.

# Contents

## Copyright notice

This document has been produced by feeding to a LaTeX typesetting engine the file `halloweenmath.dtx` version 0.11 (2019/11/01). This file is part of a work named "halloweenmath package", and may be distributed and/or modified only as a part thereof. Subsection 3.4 on page 13 explains the conditions under which the halloweenmath package may be distributed and/or modified.

The halloweenmath package has the following LPPL maintenance status: "author-maintained".

Permission is granted to make new printed copies of this document, by any means (*e.g.*, photocopying, direct printing from the electronic version, and so on).

Note, though, that distribution of this document alone is "distribution of only part of the Work" in the sense of the LaTeX Project Public License, and should therefore be avoided: please distribute, in addition, the source file `halloweenmath.dtx` together with all other files listed in `manifest.txt`, as explained as well in Subsection 3.4 and in the file `manifest.txt` itself.

## Author's addresses

The recommended way to contact the author is by e-mail; his address is:

gustavo dot mezzetti at istruzione dot it

Please include the *exact* phrase `halloweenmath␣package` in the *subject* of your message; otherwise, your message could be thrown away by a mail filter. In this regard, the weakness of the laws concerning the protection of privacy forces me to explicitly forbid the use of the above address for any kind of advertising, marketing, polling, or similar commercial or statistical contact.

If you have no e-mail access, write to:

Gustavo Mezzetti
L. S. U. "A. di Savoia Duca d'Aosta"
Via del Santo, 57
I–35123 PADOVA PD
Italy

# Introduction

On October 31, 2016, around 3 in the morning (GMT), user cfr asked a question [1] on the *TeX – LaTeX Stack Exchange* network, challenging other users to contribute LaTeX documents that showcased "the spookiest, creepiest images" ever possible. I was aware of a method—well known, nonetheless—to integrate arbitrary pictures into math formulas as though they were mathematical symbols, so, on the pretext "that, since (La)TeX is a language primarily designed to handle text with math, the lack of a 'math-flavored' answer . . . was a gap that ought to be filled", I wrote an answer implementing a command that drew a (not particularly spooky or creepy) little witch astride of her broom, but with the silly distinctive feature of being seen by LaTeX as a mathematical "large" operator like `\sum`, `\prod`, `\int`, or `\bigcup`; this first command was named `\mathwitch` and was "intended for denoting the operation of applying black magic to the ensuing subformula".

I subsequently added an extensible math accent similar to `\overrightarrow`, but with the arrow replaced by a broom carrying the same witch and a black cat, and my answer ended up winning the contest (the presence of the cat turned out to be a decisive factor); besides, some users were so enthusiast of the new commands that they requested that they were re-implemented as a new, stand-alone package. Well, here it is.

In the writing of the package, more features, and creatures, have been added. Also available now are:

- ghosts, both still and hovering (or, if you prefer, acting both as ordinary symbols and as math accents similar to `\overrightarrow`);
- a `\pumpkin` binary operator, with an associated "large" operator named `\bigpumpkin`[1] (cf. `\cup` vs. `\bigcup`, `\cap` vs. `\bigcap`, `\vee` vs. `\bigvee`, `\oplus` vs. `\bigoplus`, . . . );
- extensible "arrow-like" brooms that act as relation symbols;
- clouds, for denoting unscrutable subformulas (this one is even sensible!);
- fluttering bats, skulls, and more.

All the above are available for the `bold` math version as well as for the `normal` one. The symbols provided should prove sufficient for preparing lecture notes, slides, tests, or even exams to amuse (?) your students with during the Halloween period. Of course, any suggestion for additions and/or improvements are most welcome: you can find my addresses on page 3.

As for the name of this new package, cfr initially proposed mathwitch (from the name of the main command); Guilherme Zanotelli then suggested mathemagics (a truly befitting pun!); but in the end I decided to choose halloweenmath because (1) it instantly and clearly communicates the context in which the package arose, as well as the purpose for which it was written, and (2) it is 13 characters long.

This work is dedicated to cfr, without whom it would have never seen the light, and to whom I am offering it "as an Epiphany present"—or rather, as one would say in Italian, "*come regalo per la Befana*", where the last word bears not-so-subtle connections with the world of witches. . . [2]

---

[1] `\greatpumpkin` is available as a synonym, in homage to Linus van Pelt.

[2] Indeed, the first version of this package was released on January 6, 2017.

## Disclaimer

Although a LaTeX package is a piece of software that executes in a rather protected environment and should therefore be relatively incapable of doing serious damages, no warranty is given, as usual with computer software, that this code will work properly, or even that it will work at all. To be on the safe side, be aware that using this code can even result in loss of data and/or damages to your disks and your storage media. If you use this code, you do so exclusively at your own risk!

See Subsection 3.4 and the LaTeX Project Public Licence for more information.

## About this document

This document, like many documentation files distributed for LaTeX, is divided into two parts.

The first part, *User's Manual*, contains all the documentation you need to be able to *use* the halloweenmath package. Every user of this package will probably want to read this part.

The second part, *Implementation*, can be omitted if you want (explanations on how to omit it are given at the beginning of the part itself), and contains information that concerns only "hackers" who want to learn how the package that this document presents is *implemented*. Actually, this part contains the complete listing of the LaTeX code that implements the macros, in the usual DocStrip format; if you are not a LaTeX hacker, this part will probably have little meaning to you.

In this release too, alas, I have to ask to be excused for not having completed the comments in the *Implementation* part. Some 35% of the code is still uncommented, and, as I had already said in the *Introduction* to version 0.01, at present I cannot yet make any reliable guess as to when (or even if!) I'll be able to finish the work. It is precisely for this reason that I decided to release it in this incomplete form, to avoid deferring indefinitely the availability of the new, horrifying features that this version introduces.

## Part I

# User's Manual

## 1 About this part

This part tells you how to use the halloweenmath package. It also contains detailed instructions on how to install it and how to generate this same documentation, but normally you need not bother reading this information, either because your TeX distribution already includes both the package code and its documentation, or because package installation is implicitly dealt with by the automatic update mechanism provided by the distribution itself.

## 2 How to use the halloweenmath package

This section explains everything you need to know in order to be able to use the halloweenmath package. However, to keep this document independent of the package itself, the output produced by the various commands is not shown here: for this, you are referred to [2].

### 2.1 Package loading

The usual `\usepackage` declaration is used, as in

```
\usepackage{halloweenmath}
```

It should be noted, however, that the halloweenmath package requires the amsmath package, and loads it (without specifying any option) if it is not already loaded. If you want to pass options to amsmath, load it before halloweenmath.

The halloweenmath package defines no options by itself; nevertheless, it does honor the [no]`sumlimits` options from the amsmath package (see below).

### 2.2 Scary symbols

This package defines some commands *for use in mathematical mode*, which provide a handful of symbols, all of which honor the `bold` math version, when in force. They can be loosely grouped as follows.

#### 2.2.1 Pumpkins

`\pumpkin`
`\bigpumpkin`
`\greatpumpkin`

It is even too obvious to begin with pumpinks. In this regard, the package provides two interrelated commands:

- `\pumpkin` yields a binary operator symbol in the shape of a (little) pumpkin;
- `\bigpumpkin` is the "large" operator that is to `\pumpkin` as `\bigoplus` is to `\oplus`; it typesets a somewhat bigger pumpkin in in-line formulas, which becomes significantly larger in displayed formulas.

The [no]`sumlimits` options from the amsmath package also affect the `\bigpumpkin` command. As a homage to Linus van Pelt, `\greatpumpkin` is available as a perfect synonym for `\bigpumpkin`.

### 2.2.2 Witches, brooms, and black cats

All this arose from a witch.

`\mathwitch`
`\reversemathwich`

The `\mathwitch` command yields a “large” operator symbol, similar to `\sum`, representing a witch astride of a broom (in my answers to [1], I claimed that this symbol denotes the operation of applying black magic to the ensuing subformula, a fundamental operation in so many mathematical proofs. . . ). There is also a `\reversemathwich` command, which produces an operator that looks like a mirror image of the above, for denoting the inverse operation of undoing the black magic. Both commands honor the [`no`]`sumlimits` options from the `amsmath` package.

`\xrightwitchonbroom`
`\xleftwitchonbroom`

Witches infest the field of extensible “arrow-like” symbols too: the commands `\xrightwitchonbroom` and `\xleftwitchonbroom` are syntactically identical to, *e.g.*, `\xrightarrow`, but produce an extensible broom with a witch sitting on it. That is to say, the syntax for `\xrightwitchonbroom` is as follows

```
\xrightwitchonbroom[⟨under text⟩]{⟨over text⟩}
```

and `\xleftwitchonbroom` is similar.

`\xrightwitchonpitchfork`
`\xleftwitchonpitchfork`

In version 0.01 of this package, the `\x` $*$ `witchonbroom` commands drew under the witch a “broom” that, actually, looked more like a pitchfork (`cfr` had already complained about this in one of her comments to my original answer on TeX.SX); in the present version this has been corrected, but the output produced by the first version (which I liked) can still be obtained by means of these two commands, whose name is, indeed, more appropriate to the look of the symbols they yield. Obviously, their syntax remains identical, *e.g.*,

```
\xleftwitchonpitchfork[⟨under text⟩]{⟨over text⟩}
```

`\overrightwitchonbroom`
`\overleftwitchonbroom`
`\underrightwitchonbroom`
`\underleftwitchonbroom`

Finally, the four commands

```
\overrightwitchonbroom {⟨subformula⟩}
\overleftwitchonbroom  {⟨subformula⟩}
\underrightwitchonbroom{⟨subformula⟩}
\underleftwitchonbroom {⟨subformula⟩}
```

`\overrightwitchonpitchfork`
`\overleftwitchonpitchfork`
`\underrightwitchonpitchfork`
`\underleftwitchonpitchfork`

are similar to, *e.g.*, `\overrightarrow`, and typeset a broom, with a witch sitting on it, over or under the subformula specified in their argument.[3] Also in this case, the look of the symbols they draw has changed since version 0.01 of this package, but the old shapes have been kept available under different, and more appropriate, names: just use

```
\overrightwitchonpitchfork {⟨subformula⟩}
\overleftwitchonpitchfork  {⟨subformula⟩}
\underrightwitchonpitchfork{⟨subformula⟩}
\underleftwitchonpitchfork {⟨subformula⟩}
```

instead of the above.

**Important note:** All the commands described in this subsection have a $*$-form that adds a black cat on the broomstick. The syntax should be easy to remember, since the additional asterisk can be thought of as reminiscent of the cat.

---

[3] It should be noted that, in my answers to [1], only the command that now corresponds to `\overrightwitchonbroom` was provided, but under the different name `\overrightbroom`.

### 2.2.3 Ghosts

Of course, on Halloween, ghosts cannot be missing either.

`\mathghost`
`\mathrightghost`
`\mathleftghost`

There are three similar commands (without arguments) that produce ordinary symbols, that is, symbols that are treated like letters: the `\mathghost` command yield a symbol resembling a ghost standing upfront; the `\mathrightghost` (resp., `\mathleftghost`) command gives a symbol that looks like a ghost moving toward the right (resp., the left).

`\xrightswishingghost`
`\xleftswishingghost`

Once again, ghosts too haunt the region of extensible "arrow-like" symbols:

```
\xrightswishingghost[⟨under text⟩]{⟨over text⟩}
\xleftswishingghost [⟨under text⟩]{⟨over text⟩}
```

are analogous to, *e.g.*, `\xrightarrow`, but, in place of the arrow, draw a ghost swishing, respectively, toward the right and toward the left.

`\overrightswishingghost`
`\overleftswishingghost`
`\underrightswishingghost`
`\underleftswishingghost`

It is now easy to guess what the following four commands do:

```
\overrightswishingghost {⟨subformula⟩}
\overleftswishingghost  {⟨subformula⟩}
\underrightswishingghost{⟨subformula⟩}
\underleftswishingghost {⟨subformula⟩}
```

### 2.2.4 Bats

For some people, bats are even more horrifying than ghosts. The halloweenmath package provides a collection of commands for drawing bat-like symbols that are closely analogous to the ghost-related ones.

`\mathbat`
`\mathrightbat`
`\mathleftbat`

You can get an ordinary symbol resembling a bat flying, respectively, toward you, toward the right, and toward the left with the three argumentless commands `\mathbat`, `\mathrightbat`, and `\mathleftbat`. The height of these symbols has intentionally been made pretty small, to facilitate their use in superscripts.

`\xrightflutteringbat`
`\xleftflutteringbat`

You can get extensible "arrow-like" relation symbols that depict a fluttering bat by means of the following constructs:

```
\xrightflutteringbat[⟨under text⟩]{⟨over text⟩}
\xleftflutteringbat [⟨under text⟩]{⟨over text⟩}
```

`\overrightflutteringbat`
`\overleftflutteringbat`
`\underrightflutteringbat`
`\underleftflutteringbat`

And these are the associated "over-/under-arrow-like" commands:

```
\overrightflutteringbat {⟨subformula⟩}
\overleftflutteringbat  {⟨subformula⟩}
\underrightflutteringbat{⟨subformula⟩}
\underleftflutteringbat {⟨subformula⟩}
```

`\overbat`
`\underbat`

There are, finally, two more commands that have no parallel in the collection of ghostly symbols (because ghost are to tall!): they are

```
\overbat ∗{⟨subformula⟩}
\underbat∗{⟨subformula⟩}
```

where $*$ indicates an optional `*` denoting, as usual, their $*$-form. These constructs typeset ⟨*subformula*⟩ respectively surmounted by the bat produced by `\mathbat`, or with that symbol underneath. The normal (*i.e.*, unstarred) form pretends that the bat has zero width (but some height), so that, say, `$x+\overbat{y}+z$` has the same horizontal spacing as `$x+y+z$`; if you prefer that the actual width of the bat be taken into account instead, then use the starred variants.

### 2.2.5 Skulls

`\skull`
`\bigskull`

For skulls, I imagined a rôle very similar to that of pumpkins. Indeed, anything that is shaped like a human head reminds me of symbols like `\oplus` ($\oplus$) and `\otimes` ($\otimes$), so that my fantasy immediately goes to a binary operator and its associated "large" operator. In this case, we have:

- `\skull`, which is the binary operator (similar to `\oplus`);
- `\bigskull`, which is the "large" operator (similar to `\bigoplus`) and which, like all symbols of this kind, becomes significantly larger in displayed math.

As for all "large" operators defined by this package, the [`no`]`sumlimits` options from the amsmath package affect the `\bigskull` command too.

### 2.2.6 Other symbols

The halloweenmath package implements one more symbol that could loosely be connected with the world of witches.

`\mathcloud`
`\reversemathcloud`

Mathemagicians often deal with unscrutable formulas. To represent them, the `\mathcloud` command (without arguments) is available, which produces a symbol, with the form of a cloud, that behaves to a certain extent like a fraction, in that it is typeset in a significantly larger size in displayed formulas than in in-line ones. The companion command `\reversemathcloud` yields a mirror image of the same symbol.

## 2.3 More "arrow-like" symbols (these don't scare!)

If designing the first symbol, the `\mathwitch`, could be relatively entertaining, the implementation of the other operators and ordinary symbols like `\pumpkin`, `\mathghost`, `\mathcloud`, or `\bigskull` soon became a rather dull and repetitive task, that essentialy consisted of nothing more than placing lots of dots on a grid. The part related to the extensible "arrow-like" commands was a bit more amusing, since it involved dealing with at least a few typical TeXnical aspects. Probably, it was for this reason that this field grew so much in passing from the first release of the package to the second.

### 2.3.1 Script-style version of standard over/under arrows

In order to implement our over/under "arrow-like" symbols, we had to develop an extension of some internal macros of the amsmath package. These extensions can be applied to produce variants of the standard over/under arrows too.

`\overscriptrightarrow`
`\overscriptleftarrow`
`\overscriptleftrightarrow`
`\underscriptrightarrow`
`\underscriptleftarrow`
`\underscriptleftrightarrow`

The following six commands are analogous to their standard counterparts whose names do not contain the `script` substring, but the arrows they put above or below the given subformulas are typeset in their relative script style, instead of in the same style as the subformulas themselves (as the standard commands do):

```
\overscriptrightarrow     {⟨subformula⟩}
\overscriptleftarrow      {⟨subformula⟩}
\overscriptleftrightarrow {⟨subformula⟩}
\underscriptrightarrow    {⟨subformula⟩}
\underscriptleftarrow     {⟨subformula⟩}
\underscriptleftrightarrow{⟨subformula⟩}
```

### 2.3.2 "Unwitched" brooms and pitchforks

In the last years, the use of unmanned aircrafts has become increasingly important, and it has in parallel become more and more common to see flying devices of this kind even across the skies of our cities (just think of drones and of how they tend to be employed ubiquitously for the most heterogeneous kinds of jobs and services). Needless to say, it was inevitable that this aeronautical revolution propagated even to an inherently conservative world like that of sorcery, and it should come as no great surprise to learn that the halloweenmath package now features "unwitched" versions of all the "arrow-like" extensible symbols too, both of those that depict brooms and of those which stick to the original pitchfork-like shape. The useful side of these is that the absence of the witch allows for a reduction of the vertical space they take up.

`\xrightbroom`
`\xleftbroom`

To begin with, we have the extensible "arrow-like" relations, those modeled after `\xrightarrow`: they are

```
\xrightbroom[⟨under text⟩]{⟨over text⟩}
\xleftbroom [⟨under text⟩]{⟨over text⟩}
```

`\xrightpitchfork`
`\xleftpitchfork`

and produce extensible brooms *without* a witch sitting on them. The commands

```
\xrightpitchfork[⟨under text⟩]{⟨over text⟩}
\xleftpitchfork [⟨under text⟩]{⟨over text⟩}
```

are almost identical, only they yield pitchforks instead of brooms—or, if you prefer, brooms with the shape that was used in version 0.01 of this package.

`\rightbroom`
`\leftbroom`
`\hmrightpitchfork`
`\hmleftpitchfork`

When both the ⟨*under text*⟩ and the ⟨*over text*⟩ are void, the `\xrightbroom`... `\xleftpitchfork` commands yield a symbol that is, on purpose, more or less as large as what `\xrightarrow{}` yields (*i.e.*, $\rightarrow$); but it turns out that such a short broom (or pitchfork) completely lacks the broomstick. For this reason, the two (argumentless) commands `\rightbroom` and `\leftbroom` has been provided too: they typeset a broom the same length of a `\longrightarrow`/`\longleftarrow` ($\longrightarrow$/$\longleftarrow$). It would now seem logical to infer that the two companion commands that typeset pitchforks should be named `\rightpitchfork` and `\leftpitchfork`; unfortunately, these names already denote two different (albeit vaguely similar) symbols supplied by the MnSymbol package. Because of this potential name clash, we have chosen instead to call them `\hmrightpitchfork` and `\hmleftpitchfork`, respectively (the `hm` prefix, of course, stands for "Halloween math").

Before getting to the group of the "over-/under-arrow-like" extensible brooms, a preliminary statement needs to be made. For the "witched" version of all these, we had chosen to typeset them in the relative script style associated to the style of the affected subformula. For example, in a construct like

```
$W_{\overrightwitchonbroom{x_{1}+\dots+x_{n}}}$
```

the subformula `x_{1}+\dots+x_{n}`, argument of the `\overrightwitchonbroom` command, is set in script style, therefore the style applied to the covering broom (and witch) will be the scriptscript style. We already remarked in Subsection 2.3.1 that this is not what the standard "over-/under-arrow" commands do: instead, they use for the arrow the same style as that in which the argument gets typeset. The reason for this different behavior was that, with the same style of the affected subformula, the witch would have looked too large; but when the witch isn't there, both solutions are viable, and we do provide both.

We have, in the first place, a set of commands for placing brooms or pitchforks (a.k.a. version-0.01-styled brooms) over or under a subformula, setting them in the same style as the subformula itself. There are four “over-/under-arrow-like” extensible brooms…

`\overrightbroom`
`\overleftbroom`
`\underrightbroom`
`\underleftbroom`

```
\overrightbroom {⟨subformula⟩}
\overleftbroom  {⟨subformula⟩}
\underrightbroom{⟨subformula⟩}
\underleftbroom {⟨subformula⟩}
```

`\overrightpitchfork`
`\overleftpitchfork`
`\underrightpitchfork`
`\underleftpitchfork`

…and four corresponding “over-/under-arrow-like” extensible pitchforks:

```
\overrightpitchfork {⟨subformula⟩}
\overleftpitchfork  {⟨subformula⟩}
\underrightpitchfork{⟨subformula⟩}
\underleftpitchfork {⟨subformula⟩}
```

Then, for each of the commands just listed, a corresponding variant is provided that sets the broom, or the pitchfork, in the relative script style of the style applied to ⟨*subformula*⟩, and which is named following the same convention we already used for the commands of Subsection 2.3.1, that is, inserting the `script` substring. Thus, again, we find four “over-/under-arrow-like” extensible brooms…

`\overscriptrightbroom`
`\overscriptleftbroom`
`\underscriptrightbroom`
`\underscriptleftbroom`

```
\overscriptrightbroom {⟨subformula⟩}
\overscriptleftbroom  {⟨subformula⟩}
\underscriptrightbroom{⟨subformula⟩}
\underscriptleftbroom {⟨subformula⟩}
```

`\overscriptrightpitchfork`
`\overscriptleftpitchfork`
`\underscriptrightpitchfork`
`\underscriptleftpitchfork`

…and four matching “over-/under-arrow-like” extensible pitchforks:

```
\overscriptrightpitchfork {⟨subformula⟩}
\overscriptleftpitchfork  {⟨subformula⟩}
\underscriptrightpitchfork{⟨subformula⟩}
\underscriptleftpitchfork {⟨subformula⟩}
```

**Please note:** Since all of the commands listed in this subsection omit the witch, none of them has got a $*$-form for adding a cat (the cat can only go with the witch: no witch, no cat).

### 2.3.3 Summary of the naming scheme

It is worth noting that the commands for the extensible brooms/pitchforks follow a consistent naming scheme: the name is of the form

\\⟨*x-over-under*⟩⟨*opt-script*⟩⟨*left-right*⟩⟨*opt-witch*⟩⟨*broom-pitchf*⟩

where

⟨*x-over-under*⟩ ⟶ `x` | `over` | `under`
⟨*opt-script*⟩ ⟶ ⟨*empty*⟩ | `script`
⟨*left-right*⟩ ⟶ `left` | `right`
⟨*opt-witch*⟩ ⟶ ⟨*empty*⟩ | `witchon`
⟨*broom-pitchf*⟩ ⟶ `broom` | `pitchfork`

with the constraint that ⟨*opt-script*⟩ must be ⟨*empty*⟩ if either ⟨*x-over-under*⟩ = `x` or ⟨*opt-witch*⟩ ≠ ⟨*empty*⟩.

# 3 Installing the package code and documentation[4]

As usual for LaTeX distributions, the halloweenmath package is distributed in the form of a `.dtx` file, namely `halloweenmath.dtx`, and an accompanying `.ins` file, namely `halloweenmath.ins`.

## 3.1 Installation

To extract the code, run LaTeX (or plain TeX) once on the file `halloweenmath.ins`. This will generate the following LaTeX input file:

```
halloweenmath.sty
```

To finish the installation, move it to a LaTeX input directory. The above listing of the files you need to move is also displayed on the terminal at the end of the run of the file `halloweenmath.ins`. The documentation of your TeX installation should tell you how to find the LaTeX input directory/ies, and probably also how to create new LaTeX input directories reserved to contain your personal packages. The accepted TDS-compliant directory where the above file should be put is

```
$TEXMF/tex/latex/halloweenmath/
```

Please note: before using the halloweenmath package you must read its license (see Subsection 3.4) to see whether its terms are acceptable for you, especially for what concerns the lack of any warranty; if they are not, don't use it.

## 3.2 Documentation

To produce this documentation, run LaTeX three times (for the ToC to get right) on the file `halloweenmath.dtx`. This won't give you the index and the change history, though: for those, you should run *MakeIndex* on the files `halloweenmath.idx` and `halloweenmath.glo`, obtained during the last of the aforesaid three LaTeX runs, with styles `gind.ist` and `gglo.ist`, respectively (which are part of the standard LaTeX distribution); then pass `halloweenmath.dtx` through LaTeX twice more.

The documentation produced by default includes the *Implementation* part too; see the beginning of this part for instructions on how to omit it.

## 3.3 The makefile

A (quite trivial!) makefile, named `Makefile`, that "knows" how to carry out the installation, is offered along with the package. Although it is not a required part of the distribution (as per LPPL), you may find that some distributors supply it all the same (I hope the CTAN sites will do so). With this makefile in the same directory as the two source files `halloweenmath.dtx` and `halloweenmath.ins`, the simple shell command, given inside that same directory,

```
make
```

will *both* extract the code file *and* typeset the full documentation (in PDF). Type

```
make help
```

for more information.

---

[4] Recall that you need to read this section only in the unusual case that you have to carry out a manual installation from the sources; normally, you'll be using a prepackaged TeX distribution like TeX Live or MiKTeX, which can automatically manage installation as well as updates.

## 3.4 License

The halloweenmath package is *not* in the public domain: its author, Gustavo Mezzetti, owns the copyright, and in general retains all the rights therein; but as a special exception, the author grants you the permissions indicated below.

### 3.4.1 Distribution and/or modification

The halloweenmath package may be distributed and/or modified under the conditions of the LaTeX Project Public License, either version 1.3 of this license or (at your option) any later version. The latest version of this license is in

`http://www.latex-project.org/lppl.txt`

and version 1.3 or later is part of all distributions of LaTeX version 2005/12/01 or later.

The halloweenmath package has the LPPL maintenance status "author-maintained".

The file `manifest.txt` included along with this file specifies what the halloweenmath package consists of; more precisely, it explains how the locutions "Work" and "Compiled Work", used in the LaTeX Project Public License, are to be interpreted in the case of this work.

### 3.4.2 Use

The use of the halloweenmath package is unrestricted, provided that you accept the terms and conditions of the LaTeX Project Public License and of the following subsection for what concerns the absence of any warranty.

### 3.4.3 No warranty

There is absolutely no warranty for the halloweenmath package. The Copyright Holder provides the halloweenmath package "as is", without warranty of any kind, either expressed or implied, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The entire risk as to the quality and performance of the halloweenmath package is with you. Should the halloweenmath package prove defective, you assume the cost of all necessary servicing, repair, or correction.

In no event will The Copyright Holder, or any other party who may distribute and/or modify the halloweenmath package as permitted by the LaTeX Project Public License, be liable to you for damages, including any general, special, incidental or consequential damages arising out of any use of the halloweenmath package or out of inability to use it (including, but not limited to, loss of data, data being rendered inaccurate, or losses sustained by anyone as a result of any failure of the halloweenmath package to operate with any other programs), even if The Copyright Holder or said other party has been advised of the possibility of such damages.

# References

[1] *Seasonal Challenge (Contributions from TEXing Dead Welcome)*, https://tex.stackexchange.com/q/336768/69818

[2] *User's manual for the* halloweenmath *package*, included in the documentation of the halloweenmath package, $TEXMF/doc/latex/halloweenmath/halloweenmath-man.pdf

# Change History

0.01
- General: First writing of the code . . . 1

0.10
- General: Added “Boo” options . . . 1
  - Added “unwitched” extensible brooms and pitchforks . . . . . . . 1
  - Added bats (ordinary symbols and extensible “arrow-like” series) . . . . . . . . . . . . . . . . . . 1
  - Added skulls (binary operators and “large” operators) . . . . . . 1
  - Changed the look of the broom in the extensible “arrow-like” symbols, but retained the old one in newly added pitchforks . 1

# Index

Numbers in italics refer to the page where the corresponding entry is described, the ones underlined to the code line of the definition, the rest to the code lines where the entry is used.