ACRO

v2.0a   2015/08/25

Typeset Acronyms

Clemens Niederberger

https://bitbucket.org/cgnieder/acro/

contact@mychemistry.eu

ACRO not only allows you to create acronyms in a simple way but also lets you add them to different classes of acronyms. Lists can be created of separate classes wherever you want the list to appear.

ACRO also provides an option `single` which ignores acronyms that are used only once in the whole document.

As an experimental feature ACRO also offers the option `sort` which automatically sorts the list created by `\printacronyms`.

Table of Contents

1. Licence and Requirements

Permission is granted to copy, distribute and/or modify this software under the terms of the LaTeX Project Public License (LPPL), version 1.3 or later (`http://www.latex-project.org/lppl.txt`). The software has the status "maintained."

ACRO loads and needs the following packages: expl3,[1] xparse, xtemplate, l3keys2e,[2] zref-abspage[3] and translations[4] [Nie13].

2. About

> Acronyms and initialisms are abbreviations formed from the initial components in a phrase or a word. These components may be individual letters (as in CEO) or parts of words (as in Benelux and Ameslan). There is no universal agreement on the precise definition of the various terms nor on written usage. *[Wik12a]*

After WIKIPEDIA told us what acronyms are and we won't confuse them with units or other kinds of abbreviations – why would we need another package for them? There are several already: acronym [Oet12], acromake [Tho95], acroterm [Vos10], the abbreviations package abbrevs [Swi01] (the current version 1.4 has a bug,[5] though), the nomenclature package nomencl [Vey+05], and of course the mighty glossaries [Tal13]. So there is really no *need* for a new package.

On the other hand acronym, the best of the acronym specific packages, has one or two shortcomings and sometimes using glossaries seems a bit of an overkill (or simply inconvenient as one has to run `makeglossaries`, `makeindex` or `xindy`, then[6]). So ACRO stands somewhere in between (but closer to acronym).

The main reason for the existance of ACRO is a question on TeX.sx[7] which intrigued me and in consequence led to ACRO and it's option `single`.

ACRO has many similarities with the acronym package. In fact, quite some macros have the same name and meaning.[8]

---

1. on CTAN as l3kernel: `http://mirrors.ctan.org/macros/latex/contrib/l3kernel/`

2. on CTAN as l3packages: `http://mirrors.ctan.org/macros/latex/contrib/l3packages/`

3. on CTAN as oberdiek: `http://mirrors.ctan.org/macros/latex/contrib/oberdiek/`

4. on CTAN as translations: `http://mirrors.ctan.org/macros/latex/contrib/translations/`

5. see `http://tex.stackexchange.com/q/59840/5049` for solutions.

6. Rumour has it there is going to be a version that can be used without running an external program

7. `http://tex.stackexchange.com/q/59449/5049`

8. *Not* in the sense of `\meaning`!

Please take a minute to think and decide which package will suit your needs best. Are you planning to add a glossary to your book? You should probably go with glossaries, then. Are you planning to add a nomenclature? You may want to choose nomencl (or again: glossaries) and so on. ACRO does a good job for lists of abbreviations.

3. Basics

3.1. Creating New Acronyms

Acronyms are created with the command `\DeclareAcronym`.

`\DeclareAcronym{`⟨*id*⟩`}{`⟨*list of keys*⟩`}`
The basic command for declaring an acronym.

This command understands a number of keys which are listed below. Some of them are not described immediately but at appropriate places in the documentation.

`short` = {⟨*text*⟩} (required)
the short form of the acronym. This option is required: an acronym must have a short form. If this is set it *must* be set as first option! If another option is set first and notices the `short` option missing it assumes that the ⟨*id*⟩ should be used as short version and sets it accordingly. A warning will be written to the log then.

`long` = {⟨*text*⟩} (required)
the long form of the acronym. This option is required: an acronym must have a description.

`short-plural` = {⟨*text*⟩} Default: s
the plural ending appended to the short form.

`short-plural-form` = {⟨*text*⟩}
theplural short form of the acronym; replaces the short form when used instead of appending the plural ending.

`long-plural` = {⟨*text*⟩} Default: s
the plural ending appended to the long form.

`long-plural-form` = {⟨*text*⟩}
plural long form of the acronym; replaces the long form when used instead of appending the plural ending.

`alt-plural` = {⟨*text*⟩} Default: s
the plural ending appended to the alternative form.

`alt-plural-form` = {⟨*text*⟩}
theplural alternative form of the acronym; replaces the alternative form when used instead of appending the plural ending.

`list` = {⟨*text*⟩}

    if specified this will be written in the list as description instead of the long form.

`short-indefinite` = {⟨*text*⟩}   Default: a

    indefinite article for the short form.

`long-indefinite` = {⟨*text*⟩}   Default: a

    indefinite article for the long form.

`long-pre` = {⟨*text*⟩}

    ⟨*text*⟩ is prepended to the long form in the text but not in the list of acronyms.

`long-post` = {⟨*text*⟩}

    ⟨*text*⟩ is appended to the long form in the text but not in the list of acronyms.

`alt` = {⟨*text*⟩}
    alternative short form.

`alt-indefinite` = {⟨*text*⟩}   Default: a

    indefinite article for the alternative form.

`extra` = {⟨*text*⟩}
    extra information to be added in the list of acronyms.

`foreign` = {⟨*original long form*⟩}

    can be useful when dealing with acronyms in foreign languages, see section 3.6 for details.

`sort` = {⟨*text*⟩}
    if used the acronym will be sorted according to this key instead of its ID.

`class` = {⟨*text*⟩}
    the class the acronym belongs to.

`cite` = {[⟨*prenote*⟩][⟨*postnote*⟩]{⟨*citation keys*⟩}}
    a citation that is printed to the acronym according to an option explained later.

`short-format` = {⟨*T_EX code*⟩}
    the format used for the short form of the acronym.

`long-format` = {⟨*T_EX code*⟩}
    the format used for the long form of the acronym.

`first-long-format` = {⟨*T_EX code*⟩}
    the format used for the first long form of the acronym as set with \ac, \acf or \acflike and
    their uppercase, plural and indefinite forms.

`pdfstring` = {{⟨*text*⟩/⟨*plural ending*⟩}}
    used as PDF string replacement in bookmarks when used together with the hyperref package.
    The appended plural ending is optional. If you leave it (*and* the /) the default ending is used.

accsupp = {⟨*text*⟩}
> sets the `ActualText` key as presented by the accsupp package for the acronym.

index-sort = {⟨*text*⟩}

> If you use the package option `index` every occurrence of an acronym is recorded to the index
> and sorted by its ID or (if set) by the value of the `sort` key. This key allows to set an individual
> sorting option for the index. See section 4.7 for details.

index = {⟨*text*⟩}

> This key allows to overwrite the automatic index entry with an arbitrary one. See section 4.7
> for details.

index-cmd = {⟨*text*⟩}

> This key let's you set an individual index creating command for this acronym. It should be a
> command that takes one mandatory argument. See section 4.7 for details.

In its simplest form an acronym needs a short and a long form. Please note that both keys
must be set and that the `short` key *must* always be the *first* key that is set.

```
1 % preamble:
2 \DeclareAcronym{test}{
3   short = ST ,
4   long  = Some Test
5 }
```

This creates the acronym "ST" with the ID "test" and the long form "Some Test."

The `format` key allows you to choose a specific format for the short form of an acronym:

```
1 % preamble:
2 \DeclareAcronym{ot}{
3   short        = ot ,
4   long         = Other Test ,
5   short-format = \scshape
6 }
```

The short form now looks like this: OT.

The `cite` key needs a bit explaining. It expects arguments like the standard `\cite` command,
i.e., two optional arguments setting the ⟨*prenote*⟩ and ⟨*postnote*⟩ and one mandatory argument
setting the citation key.

```
1 % preamble:
2 \DeclareAcronym{ny}{
3   short       = NY ,
4   short-plural = ,
5   long        = New York ,
6   long-plural  = ,
7   cite        = {NewYork}
8 }
```

```
1 % bib file for use with biber/biblatex:
2 @online{NewYork,
3   author  = {Wikipedia},
4   title   = {New York City},
5   urldate = {2012-09-27},
6   url     = {http://en.wikipedia.org/wiki/New_York_City},
7   year    = {2012}
8 }
```

The first appearance now looks as follows[9]: New York (NY) [Wik12b].

3.2. Using the Acronyms – the Commands

Acronyms are used with one of the following commands:

\ac*{⟨*id*⟩}
  basic command; the first output is different from subsequent ones.

\Ac*{⟨*id*⟩}
  same as \ac but capitalizes the first letter of the long form.

\acs*{⟨*id*⟩}
  short form; the actual acronym.

\acl*{⟨*id*⟩}
  long form; the meaning of the acronym.

\Acl*{⟨*id*⟩}
  same as \acl but capitalizes first letter.

---

9. The appearance of the citation of course depends on the citation style you're using.

`\aca*{`⟨*id*⟩`}`
> **a**lternative short form as specified in the `alt` key of `\DeclareAcronym`; if it hasn't been specified this is identical to `\acs`.

`\acf*{`⟨*id*⟩`}`
> first form; output like the first time `\ac` is output.

`\Acf*{`⟨*id*⟩`}`
> same as `\acf` but capitalizes first letter of the long form.

`\acp*{`⟨*id*⟩`}`
> **p**lural form of `\ac`;

`\Acp*{`⟨*id*⟩`}`
> same as `\acp` but capitalizes first letter of the long form.

`\acsp*{`⟨*id*⟩`}`
> plural form of `\acs`;

`\aclp*{`⟨*id*⟩`}`
> plural form of `\acl`;

`\Aclp*{`⟨*id*⟩`}`
> same as `\aclp` but capitalizes first letter.

`\acap*{`⟨*id*⟩`}`
> plural form of `\aca`;

`\acfp*{`⟨*id*⟩`}`
> plural form of `\acf`;

`\Acfp*{`⟨*id*⟩`}`
> same as `\acfp` but capitalizes first letter of the long form.

If an acronym is used the first time with `\ac` its output is different from subsequent uses. To be clear on this: the first time! If the acronym has been used with *any* of the output commands before it is *not* the first time any more.

If you use the starred variant an acronym will not be marked as used. This proves useful if an acronym is typeset in a section title, for example, since then the appearance in the table of contents won't mark it as used.

```
1 % preamble:
2 % \DeclareAcronym{cd}{
3 %   short       = cd ,
4 %   long        = Compact Disc ,
5 %   short-format = \scshape
6 % }
7 first time: \ac{cd} \\
8 second time: \ac{cd} \\
9 short: \acs{cd} \\
10 alternative: \aca{cd} \\
11 first again: \acf{cd} \\
12 long: \acl{cd} \\
13 short plural: \acsp{cd} \\
14 long plural: \aclp{cd}
```

first time: Compact Disc (CD)
second time: CD
short: CD
alternative: CD
first again: Compact Disc (CD)
long: Compact Disc
short plural: CDs
long plural: Compact Discs

3.3. Plural Forms

If an acronym is defined in the standard way ACRO uses an 's' that's appended to both the short and the long form when one of the plural commands is used. However, that is not always the best solution. For one thing not all acronyms may have a plural form. Second, the plural form especially of the long forms may be formed differently. And third, other languages can have other plural endings.

For these reasons \DeclareAcronym can get the following keys:

short-plural = {⟨*text*⟩}   Default: s
  The plural ending of the short form.

long-plural = {⟨*text*⟩}   Default: s
  The plural ending of the long form.

long-plural-form = {⟨*text*⟩}
  An alternative plural form for the long form.

These keys are optional. If they're not used, the default setting is s. If you use long-plural-form the long form will be replaced by the specified plural form when necessary.

Suppose we define the following acronyms:

```
1 \DeclareAcronym{cd}{
2   short        = cd ,
3   long         = Compact Disc ,
4   short-format = \scshape
5 }
6 \DeclareAcronym{ny}{
7   short        = NY ,
8   short-plural = ,
```

```
 9    long         = New York ,
10    long-plural  =
11 }
12 \DeclareAcronym{sw}{
13    short        = SW ,
14    long         = Sammelwerk ,
15    long-plural = e
16 }
17 \DeclareAcronym{MP}{
18    short           = MP ,
19    long            = Member of Parliament ,
20    long-plural-form = Members of Parliament
21 }
```

These acronyms now have the following plural appearances:

```
1 \acsp{cd}, \aclp{cd} \\        CDs, Compact Discs
2 \acsp{ny}, \aclp{ny} \\        NY, New York
3 \acsp{sw}, \aclp{sw} \\        SWs, Sammelwerke
4 \acsp{MP}, \aclp{MP}           MPs, Members of Parliament
```

3.4. Alternative Short Forms

For some acronyms it might be useful to have alternative forms. For this `\DeclareAcronym` has another key:

alt = {⟨*text*⟩}
  Alternative short form.

```
1 % preamble:
2 % \DeclareAcronym{jpg}{
3 %    short = JPEG ,
4 %    alt   = JPG ,
5 %    long  = Joint Photographic Experts Group
6 % }
7 default: \acs{jpg} \\
8 alt.: \aca{jpg}
```

    default: JPEG
alt.: JPG

The alternative form uses the same plural ending as the default short form and is formatted in the same way.

3.5. Extra Information for the List Entry

Of course you can print a list of acronyms where their meaning is explained. Sometimes it can be useful to add additional information there. This is done with another key to `\DeclareAcronym`:

extra = {⟨*text*⟩}
Additional information for the list of acronyms.

These information will only be displayed in the list. See section 5 for the impact of the following example.

```
1 % preamble:
2 % \DeclareAcronym{nato}{
3 %   short        = nato ,
4 %   long         = North Atlantic Treaty Organization ,
5 %   extra        = \textit{deutsch}: Organisation des Nordatlantikvertrags ,
6 %   short-format = \scshape
7 % }
8 The \ac{nato} is an intergovernmental military alliance based on the
9 North Atlantic Treaty which was signed on 4~April 1949. \ac{nato}
10 headquarters are in Brussels, Belgium, one of the 28 member states
11 across North America and Europe, the newest of which, Albania and
12 Croatia, joined in April 2009.
```

The North Atlantic Treaty Organization (NATO) is an intergovernmental military alliance based on the North Atlantic Treaty which was signed on 4 April 1949. NATO headquarters are in Brussels, Belgium, one of the 28 member states across North America and Europe, the newest of which, Albania and Croatia, joined in April 2009.

3.6. Foreign Language Acronyms

Introduced in version 1.3

I repeatedly read the wish for being able to add translations to acronyms when the acronyms stem from another language than the document language, *i.e.*, something like the following in a German document:

```
1 \ac{ecu}\\                          Steuergerät (Electronic Control Unit, ECU)
2 \ac{ecu}                            ECU
```

That's why I decided to add the following key:

foreign = {⟨*original long form*⟩}
> A description for an acronym originating in another language than the document language.

> Here is the definition of the above mentioned ECU acronym:

```
1 \DeclareAcronym{ecu}{
2   short   = ECU ,
3   long    = Steuerger\"at ,
4   foreign = Electronic Control Unit
5 }
```

As you have seen this adds the foreign entry to the first appearance of an acronym. It is also added in parentheses to the list of acronyms after the long entry. Actually the entry there is the argument to the following command:

\acroenparen{⟨*argument*⟩}
> Places ⟨*argument*⟩ in parentheses: \acroenparen{example}: (example). See page 20 for a way to customize this other than redefining it.

4. Additional Commands and Possibilities

4.1. Indefinite Forms

Introduced in version 1.2

Unlike many other languages [10] in English the indefinite article is not determined by the grammatical case, gender or number but by the pronounciation of the following word. This means that the short and the long form of an acronym can have different indefinite articles. For these cases ACRO offers the keys short-indefinite, alt-indefinite and long-indefinite whose default is a. For every lowercase singular command two alternatives exist, preceded by i and I, respectively, which output the lowercase and uppercase version of the corresponding indefinite article.

```
1 % preamble:
2 % \DeclareAcronym{ufo}{
3 %   short         = UFO ,
4 %   long          = unidentified flying object ,
5 %   long-indefinite = an
6 % }
7 \Iac{ufo}; \iacs{ufo}; \iacl{ufo}
```

---

10. Let's better say: unlike the other languages where I know at least the basics.

> An unidentified flying object (UFO); a UFO; an unidentified flying object

4.2. Uppercasing

`\acfirstupper{⟨token list⟩}`

This command uppercases the first token in ⟨*token list*⟩. The command is less powerful than `\makefirstuc` that is provided by the mfirstuc package [Tal12] but it is expandable. Obvious downsides are for example that it does not uppercase accented letters.

4.3. Simulating the First Appearance

Users told me [11] that there are cases when it might be useful to have the the acronym typeset according to the `first-style` but with another text than the long form. For such cases ACRO offers the following commands.

`\acflike*{⟨id⟩}{⟨instead of long form⟩}`
Write some alternative long form for acronym ⟨*id*⟩ as if it were the first time the acronym was used.

`\acfplike*{⟨id⟩}{⟨instead of long form⟩}`
Plural form of `\acflike`.

```
1 \acsetup{first-style=footnote}
2 \acflike{ny}{the big apple}
```

NY[a]
_____

a. the big apple [Wik12b]

The plural ending in `\acfplike` is only appended to the short form. It makes no sense to append it to the text that is inserted manually anyway. Note that whatever text you're inserting might be gobbled depending on the `first-style` you're using.

4.4. Using Classes

The acronyms of ACRO can be divided into different classes. This doesn't change the output but allows different acronym lists, see section 5. For this `\DeclareAcronym` has an additional key:

`class = {⟨text⟩}`
Associated class for an acronym.

This might be useful if you can and want to divide your acronyms into different types, technical and grammatical ones, say, that shall be listed in different lists.

---

11. Well – one, to be precise ;)

```
1 % preamble:
2 % \DeclareAcronym{la}{
3 %    short        = LA ,
4 %    short-plural = ,
5 %    long         = Los Angeles ,
6 %    long-plural  = ,
7 %    class        = city
8 % }
9 % \DeclareAcronym{ny}{
10 %    short        = NY ,
11 %    short-plural = ,
12 %    long         = New York ,
13 %    long-plural  = ,
14 %    class        = city ,
15 %    cite         = NewYork
16 % }
17 \acl{la} (\acs{la}) \\
18 \acl{ny} (\acs{ny})
```

Los Angeles (LA)
New York (NY)

4.5. Reset or Mark as Used, Test if Acronym Has Been Used

If you want for some reason to fool ACRO into thinking that an acronym is used for the first time you can call one of these commands:

\acreset{⟨*comma separated list of ids*⟩}

Introduced in version 0.5

This will reset a used acronym such that the next use of \ac will again print it as if it were used the first time. This will *not* remove an acronym from being printed in the list if it actually *has* been used before.

\acresetall

Reset all acronyms.

\acifused{⟨*id*⟩}{⟨*true*⟩}{⟨*false*⟩}

Introduced in version 1.3e

This command tests if the acronym with ID ⟨*id*⟩ has already been used and either puts true or false in the input stream.

```
1 \acreset{ny}\ac{ny}
```

New York (NY) [Wik12b]

Beware that both commands act *globally*! There are also commands that effectively do the opposite of \acreset, *i.e.*, mark acronyms as used:

\acuse{⟨*comma separated list of ids*⟩}

Introduced in version 0.5

This has the same effect as if an acronym had been used twice, that is, further uses of \ac will

print the short form and the acronym will in any case be printed in the list (as long as its class is not excluded).

`\acuseall`

Mark all acronyms as used.

4.6. `\ac` and Friends in P D F Bookmarks

ACRO's commands usually are not expandable which means they'd leave unallowed tokens in P D F bookmarks. hyperref offers `\texorpdfstring` to circumvent that issue manually but that isn't really a nice solution. What's the point of having macros to get output for you if you have to specify it manually after all?

That is why ACRO offers a preliminary solution for this. In a bookmark every `\ac` like command falls back to a simple text string typesetting what `\acs` would do (or `\acsp` for plural forms). These text strings both can accessed manually and can be modified to an output reserved for P D F bookmarks.

`\acpdfstring{`⟨*id*⟩`}`

Access the text string used in P D F bookmarks.

`\acpdfstringplural{`⟨*id*⟩`}`

Access the plural form of the text string used in P D F bookmarks.

`pdfstring` = `{{`⟨*pdfstring*⟩`/`⟨*plural ending*⟩`}}`

Key for `\DeclareAcronym` to declare a custom text string for P D F bookmarks. The plural ending can be set optionally.

`accsupp` = `{`⟨*text*⟩`}`

Key for `\DeclareAcronym` to set the `ActualText` property of `\BeginAccSupp` (see accsupp's documentation for details) to be used for an acronym. It only has an effect when the package option `accsupp` is used, too.

For example the P D F acronym used in the title for this section is defined as follows:

```
1 \DeclareAcronym{pdf}{
2   short     = pdf ,
3   long      = Portable Document Format ,
4   format    = \scshape ,
5   pdfstring = PDF ,
6   accsupp   = PDF
7 }
```

This also demonstrates the `accsupp` key. For this to work you need to use the *package option* `accsupp`, too, which will load the package accsupp. Then the key `accsupp` will set the

ActualText property of \BeginAccSupp. Please refer to accsupp's documentation for details. To see its effect copy P D F and paste it into a text file. You should get uppercase letters instead of lowercase ones.

4.7. Adding Acronyms to the Index

A C R O has the package option index. If it is used an index entry will be recorded every time an *unstarred* acronym command is used. The index entry will be ⟨*id*⟩@⟨*short*⟩, ⟨*sort*⟩@⟨*short*⟩ if the sort key has been set, ⟨*index-sort*⟩@⟨*short*⟩ if the index-sort has been set, or ⟨*index*⟩ if the key index has been set for the specific acronym. The short versions appearing there are formatted according to the chosen format of the corresponding acronym, of course.

This document demonstrates the feature. You can find every acronym that has been declared in the index. In order to allow flexibility the indexing command can be chosen both globally via package option and individually for every acronym. This would allow to add acronyms to a specific index if more than one index is used, for example with help of the imakeidx package.

I'm not yet convinced this is a feature many people if anyone needs and if they do if it is flexible enough. If you have any thoughts on this I'd appreciate an email.

5. Printing the List

Printing the whole list of acronyms is easy: just place \printacronyms where ever you want the list to be.

\printacronyms[⟨*options*⟩]
  Print the list of acronyms.

The commands takes a few options, namely the following ones:

include-classes = {⟨*list of classes*⟩}
  Takes a comma-separated list of the classes of acronyms that should be in the list.

exclude-classes = {⟨*list of classes*⟩}
  Takes a comma-separated list of the classes of acronyms that should *not* be in the list.

name = {⟨*name of the list*⟩}
  sets the name for the list.

heading = {⟨*sectioning command without leading backslash*⟩}   Default: section*

  Sets the sectioning command for the heading of the list. A special value is none which suppresses the heading.

sort = <u>true</u>|false   Default: true

  Set sorting for this list only.

```
1 \acsetup{extra-style=comma}
2 \printacronyms[exclude-classes=city]
3
4 \printacronyms[include-classes=city,name={City Acronyms}]
```

Acronyms

CD Compact Disc

CTAN Comprehensive TeX Archive Network

ECU Steuergerät (Electronic Control Unit)

ID identification string

JPEG Joint Photographic Experts Group

MP Member of Parliament

NATO North Atlantic Treaty Organization, *deutsch*: Organisation des Nordatlantikvertrags

PDF Portable Document Format

SW Sammelwerk

ST Some Test

UFO unidentified flying object

City Acronyms

LA Los Angeles

NY New York

You can see that the default layout is a `description` list with a `\section*` title. Both can be changed, see section 6.

The command `\printacronyms` needs two LaTeX runs. This is a precaution to avoid error messages with a possibly empty list. But since almost all documents need at least two runs and often are compiled much more often than that, this fact shouldn't cause too much inconvenience.

6. Options and Customization

6.1. General Options

There are a few options which change the general behaviour of ACRO. Underlined values are used if no value is given.

messages = silent|loud   Default: loud

Setting messages = {silent} will turn all of ACRO's error messages into warnings and all of ACRO's warnings into info messages. Be sure to check the log file carefully if you decide to set this option.

single = true|false   Default: false

If set to true an acronym that's used only once (with \ac) in a document will only print the acronym in a specified form and will not be printed in the list.

single-form = long|short|alt|first   Default: long

Determines how a single appearance of an acronym is printed if single = {true} has been chosen.

hyperref = true|false   Default: false

If set to true the short forms of the acronyms will be linked to their list entry.

label = true|false   Default: false

If set to true this option will place \label{⟨*prefix*⟩⟨*id*⟩} the first time the acronym with ID ⟨*id*⟩ is used.

label-prefix = {⟨*text*⟩}   Default: ac:

The prefix for the \label that is placed when option label = {true} is used.

record-pages = true|false   Default: true

Since ACRO can handle arabic, roman and Roman page numbers but *not* any other kind of numbering this option allows to turn the page number recording off for these cases as it would lead to errors else. This affects the whole document and can only be set in the preamble! It means you cannot have page numbers in the list of acronyms in this case. Or rather: you can if you use pages = {first}.

only-used = true|false   Default: true

This option is true as default. It means that only acronyms that are actually used in the document are printed in the list. If false, all acronyms defined with \DeclareAcronym will be written to the list.

mark-as-used = first|any   Default: any

This option determines wether an acronym is mark as used when the *first* form is used the first time (with \ac, \acf or \acflike and their uppercase, plural and indefinite forms) or when any of the \ac-like commands is used.

macros = <u>true</u>|false   Default: false
: If set to true this option will create a macro \\$\langle id \rangle$ for each acronym as a shortcut for \ac{$\langle id \rangle$}. Already existing macros will *not* be overwritten.

xspace = <u>true</u>|false   Default: false

: If set to true this option will append \xspace from the xspace package to the commands created with the macros option.

strict = <u>true</u>|false   Default: false
: If set to true and the option macros = {true} is in effect then already existing macros will be overwritten.

sort = <u>true</u>|false   Default: true
: If set to true the acronym list will be sorted automatically. The entries are sorted by their ID ignoring upper and lower case. This option needs the experimental package l3sort (from the l3experimental bundle) and can only be set in the preamble.

cite = <u>all</u>|first|none   Default: first
: This option decides whether citations that are added via cite are added to each first, every or no appearance of an acronym.

cite-cmd = {$\langle control\ sequence \rangle$}   Default: \cite
: This option determines which command is used for the citation. Each citation command that takes the cite key as argument is valid, for example biblatex's \footcite.

cite-connect = {$\langle code \rangle$}   Default: \nobreakspace
: Depending on the citation command in use a space should be inserted before the citation or maybe not (e.g. \footcite...). This option allows you to set this. Actually it can be used to place arbitrary code right before the citation.

group-citation = <u>true</u>|false   Default: false

: If set to true the short form (or the long form) and the citation of an acronym will be printed together in parentheses when an acronym is used the first time.

group-cite-cmd = {$\langle control\ sequence \rangle$}   Default: \cite

: This option determines which command is used for the citation when an acronym is used the first time *and* group-citation = {true}. Each citation command that takes the cite key as argument is valid, for example biblatex's \footcite.

index = <u>true</u>|false   Default: false

: If set to true an index entry will be recorded every time an *unstarred* acronym command is used for the corresponding acronym.

index-cmd = {$\langle control\ sequence \rangle$}   Default: \index

: Chooses the index command that is used when option index has been set to true.

accsupp = <u>true</u>|false   Default: false

: Activates the access support as provided by the accsupp package.

uc-cmd = {⟨*control sequence*⟩}   Default: \acfirstupper

The command that is used to capitalize the first word in the \Ac and the like commands. You can change it to another one like for example \makefirstuc [12] or \MakeTextUppercase.[13]

All options of this and the following sections can be set up either as package options or via the setup command:

\acsetup{⟨*options*⟩}

Set up ACRO anywhere in the document. Or separate package loading from setup.

```
1 % with \acsetup{macros}
2 we could have used these before: \nato, \ny
```

we could have used these before: NATO, NY

6.2. Options Regarding Acronyms

The options described in this section all influence the layout of one of the possible output forms of the acronyms.

short-format = {⟨*format*⟩}   (initially empty)

Sets a format for all short forms. For example short-format = {\scshape} would print all short forms in small caps.

long-format = {⟨*format*⟩}   (initially empty)

The same for the long forms.

foreign-format = {⟨*format*⟩}   (initially empty)

Introduced in
version 1.3

The format for the foreign entry when it appears as part of the first appearance of an acronym.

first-long-format = {⟨*format*⟩}   (initially empty)

Introduced in
version 1.2

The format for the long form on first usage (with \ac, \acf or \acflike and their uppercase, plural and indefinite forms).

list-short-format = {⟨*format*⟩}   (initially empty)

Introduced in
version 1.1

An extra format for the short entries in the list. If not used this is the same as short-format. Please be aware that a call of short-format after this one will overwrite it again.

list-long-format = {⟨*format*⟩}   (initially empty)

An extra format for the long entries in the list. If not used this is the same as long-format. Please be aware that a call of long-format after this one will overwrite it again.

---

12. from the mfirstuc package
13. from the textcase package

list-foreign-format = {⟨*format*⟩}   Default: \acroenparen

  The format for the foreign entry as it appears in the list. This may be code that ends with a macro that takes a mandatory argument.

extra-format = {⟨*format*⟩}   (initially empty)

  The same for the extra information.

first-style = default|plain|empty|square|short|reversed|plain-reversed|footnote|sidenote

Default: default

Changed in
version 1.1
  The basic style of the first appearance of an acronym. The value sidenote needs the command \sidenote to be defined for example by the sidenotes package.

extra-style = default|plain|comma|paren|bracket   Default: default

  Defines the way the extra information is printed in the list.

plural-ending = {⟨*tokenlist*⟩}   Default: s

  With this option the default plural ending can be set.

```
1 % (Keep in mind that we're in
2 % a minipage here!)
3 \acsetup{first-style=empty}
4 \acf{ny} \\
5 \acsetup{first-style=footnote}
6 \acf{ny} \\
7 \acsetup{first-style=square}
8 \acf{ny} \\
9 \acsetup{first-style=short}
10 \acf{ny} \\
11 \acsetup{first-style=reversed}
12 \acf{ny} \\
13 \acsetup{first-style=plain}
14 \acf{ny} \\
15 \acsetup{first-style=plain-reversed}
16 \acf{ny}
```

NY
NY[a]
New York [NY] [Wik12b]
NY [Wik12b]
NY (New York) [Wik12b]
New York – NY [Wik12b]
NY – New York [Wik12b]

───────────────
  a. New York [Wik12b]

6.3. Options Regarding the List

page-ref = none|plain|comma|paren   Default: none

  If this option is set to a value other than none the page numbers of the an acronym appeared on are printed in the list. Please note that this is an experimental feature and might fail in quite a number of cases. If you notice anything please send me an email!

pages = all|first   Default: all

Introduced in
version 1.5
  If the option page-ref has any value other than none this option determines wether all usages of the acronyms are listed or only the first time. Implicitly sets label = {true}.

6. Options and Customization

page-name = {⟨*page name*⟩}   Default: `p.\@\,`
The "name" of the page label. This is automatically translated to the active language. However for the time being there are many translations missing, yet. Please notify me if you find your language missing.

pages-name = {⟨*page name plural*⟩}   Default: `pp.\@\,`

Introduced in version 1.0

The "name" of the page label when there are more than one page. This is automatically translated to the active language. However for the time being there are many translations missing, yet. Please notify me if you find your language missing.

following-page = <u>true</u>|false   Default: `false`

Introduced in version 1.3

If set to `true` a page range in the list of acronyms that consists of two pages will be written by the first page and an appended `f`. This depends on the option `next-page`.

following-pages = <u>true</u>|false   Default: `false`

Introduced in version 1.3

If set to `true` a page range in the list of acronyms that set consists of more than two pages will be written by the first page and an appended `ff`. This depends on the option `next-pages`.

next-page = {⟨*text*⟩}   Default: `\,f.\@`

Introduced in version 1.0

Appended to a page number when `following-page` is set to `true` and the range is only 2 pages long. This is automatically translated to the active language. However, for the time being there are many translations missing, yet. Please notify me if you find your language missing.

next-pages = {⟨*text*⟩}   Default: `\,ff.\@`

Introduced in version 1.0

Appended to a page number when `following-pages` is set to `true` and the range is more than 2 pages long. This is automatically translated to the active language. However, for the time being there are many translations missing, yet. Please notify me if you find your language missing.

list-type = table|⟨*list*⟩   Default: `description`
This option let's you choose how the list is printed. ⟨*list*⟩ can be any valid list like `itemize` or `description`.

list-style = list|tabular|longtable|extra-tabular|extra-longtable|extra-tabular-rev|
extra-longtable-rev   Default: `list`
If you choose `list-type` = {table} you have to specify which kind of table should be used. If you choose `longtable`, `extra-longtable` or `extra-longtable-rev` you have to load longtable in your preamble. The values `extra-longtable` and `extra-longtable-rev` put the extra information in a column of its own.

list-heading = chapter|chapter*|section|section*|subsection|subsection*|subsubsection|
subsubsection*|addchap|addsec|none

Default: `section*`

Changed in version 2.0

The heading type of the list. The last two only work with a KOMA-Script class that also defines the appropriate command. A special value is `none` which suppresses the heading.

list-name = {⟨*list name*⟩}   Default: `Acronyms`
The name of the list. This is what's written in the list-heading. This is automatically translated

21

to the active language. However, for the time being there are many translations missing, yet. Please notify me if you find your language missing.

`list-table-width` = {⟨*dimension*⟩}   Default: `.7\columnwidth`
    This has only an effect if you chose `list-type` = {⟨*table*⟩}. The *second* column (or the third if you choose one of the `-rev` styles) of the table is a `p` column whose width can be specified with this option.

`list-caps` = <u>true</u>|false   Default: `false`
    Print the first letters of the long form capitalized.

    If you for example have loaded enumitem you can define a custom list for the acronym list:

```
1  % preamble:
2  % \usepackage{enumitem}
3  \newlist{acronyms}{description}{1}
4  \newcommand*\addcolon[1]{#1:}
5  \setlist[acronyms]{
6    labelwidth=3em,
7    leftmargin=3.5em,
8    noitemsep,
9    itemindent=0pt,
10   font=\addcolon}
11 \acsetup{list-type=acronyms,hyperref=false,extra-style=comma}
12 \printacronyms
```

Acronyms

CD: Compact Disc
CTAN: Comprehensive TEX Archive Network
ECU: Steuergerät (Electronic Control Unit)
ID: identification string
JPEG: Joint Photographic Experts Group
LA: Los Angeles
MP: Member of Parliament
NATO: North Atlantic Treaty Organization, *deutsch*: Organisation des Nordatlantikvertrags
NY: New York
PDF: Portable Document Format
SW: Sammelwerk
ST: Some Test
UFO: unidentified flying object

6.4. Other Heading Command

If you want a `list-heading` which is not yet provided you can add your own option:

```
1 \DeclareInstance{acro-title}{myname}{sectioning}
2   { name-format = command that uses #1 }
```

After that you could use `list-heading` = {myname}.

7. Trailing Tokens and Special Action

ACRO has the possibility to look ahead for certain tokens and switch a boolean if it finds them. Per default ACRO knows about three tokens: the "`dot`" (.), the "`dash`" (-) and the "`babel-hyphen`" (\babelhyphen).

A token is made known to ACRO with the following macro:

`\AcroRegisterTrailing`⟨*token*⟩{⟨*name*⟩}
This registers the token ⟨*token*⟩ so ACRO looks if it follows directly after an acronym macro. ⟨*name*⟩ is the internal name for this token.

The ACRO package already registers the above mentioned tokens:

```
1 \AcroRegisterTrailing . {dot}
2 \AcroRegisterTrailing - {dash}
3 \AcroRegisterTrailing \babelhyphen {babel-hyphen}
```

If a token is registered it doesn't mean that ACRO looks for it. The token must first be activated for this:

`activate-trailing-tokens` = {⟨*csv list of token names*⟩}
Tell ACRO to look for trailing tokens. This is done by giving a csv list of the internal *names* of the tokens. Per default only `dot` is activated.

`deactivate-trailing-tokens` = {⟨*csv list of token names*⟩}
Tell ACRO not to look for trailing tokens. This is done by giving a csv list of the internal *names* of the tokens.

All the above on its own does nothing visible. However: inside of an acronym, *i. e.*, for example inside the long or the short form it can be tested for those trailing options:

\aciftrailing{⟨*csv list of token names*⟩}{⟨*true*⟩}{⟨*false*⟩}

Check if one of the tokens listed in ⟨*csv list of token names*⟩ is following and either place ⟨*true*⟩ or ⟨*false*⟩ in the input stream.

ACRO uses this to define to further macros:

\acdot

Inserts a . if no dot follows.

\acspace

Inserts a \space if no dash or babel-hyphen follows.

The definitions are equivalent to the following code:

```
1 \newcommand*\acdot{\aciftrailing{dot}{}{.}}
2 \newcommand*\acspace{\aciftrailing{dash,babel-hyphen}{}{\space}}
```

This could be used to define an acronym as follows:

```
1 \DeclareAcronym{etc}{
2   short = {\textit{etc}\acdot} ,
3   long  = {\textit{et cetera}} ,
4   short-plural = , long-plural =
5 }
```

If now you somewhere use

```
1 \ac{etc}.
```

there won't be two dots printed.

The command \acspace is used already in the definition of the first appearance of a macro. Let's say you're a German chemist and you have

```
1 \DeclareAcronym{PU}{
2   long = Polyurethan ,
3   long-plural = e
4 }
```

and you use it the first time like this:

```
1 \ac{PU}-Hartschaum
```

then according to German orthography and typesetting rules this should be printed as

"Polyurethan(PU)-Hartschaum"

i. e., with *no* space between long and short form. This is exactly what happens it you say

```
1 \acsetup{activate-trailing-tokens = {dash,babel-hyphen}}
```

in the preamble.

8. About Page Ranges

If you enable the `page-ref` option ACRO adds page numbers to the list of acronyms. In version 0.* it would add a page reference for an acronym in the list of acronyms that used \pageref to refer to the first appearance of an acronym. This is retained using `pages = {first}`. Version 1.0 uses a different approach that doesn't use a label but instead will list *all* pages an acronym appeared on. With hyperref the pages are referenced using \hyperpage.

There are some options that control how this list will be typeset, e.g., `following-page`, `next-pages` or the option `page-ref` itself. It is important to mention that the page list will always take at least two compilation runs until changes in the options or the actual page numbers affect it. This is due to the fact that the updated sequence is first written to the aux file and only read in during the next run.

9. Language Support

ACRO detects if packages babel or polyglossia are being loaded and tries to adapt certain strings to match the chosen language. However, due to my limited language knowledge only a few translations are provided. I'll show how the English translations are defined so you can add the translations to your preamble if needed. Even better would be you'd send me a short email at contact@mychemistry.eu with the appropriate translations for your language and I'll add them to ACRO.

```
1 \DeclareTranslation{English}{acronym-list-name}{Acronyms}
2 \DeclareTranslation{English}{acronym-page-name}{p.}
3 \DeclareTranslation{English}{acronym-pages-name}{pp.}
4 \DeclareTranslation{English}{acronym-next-page}{f.}
5 \DeclareTranslation{English}{acronym-next-pages}{ff.}
```

10. hyperref Support

The option `hyperref` = {`true`} adds internal links from all short (or alternative) forms to their respective list entries. Of course this only works if you have loaded the hyperref package in your preamble. You should use this option with care: if you don't use `\printacronyms` anywhere this option will result in loads of hyperref warnings. Also printing several lists can result in warnings if don't clearly separate the lists into different classes. If an acronym appears in more than one list there will also be more than one hypertarget for this acronym.

Using hyperref will also add `\hyperpage` to the page numbers in the list (provided they are displayed in the style chosen). Like with an index the references will thus not point to the acronyms directly but to the page they're on.

11. Defining Own Acronym Macros

Introduced in
version 2.0

The commands explained in section 3.2 have all been defined with a dedicated command – there is a family of dedicated commands, actually:

`\NewAcroCommand{`⟨*cs*⟩`}{`⟨*code*⟩`}`
  Defines a new ACRO acronym command ⟨*cs*⟩. This sets up the necessary framework needed by acronym commands and defines ⟨*cs*⟩ with an optional star argument and a mandatory argument for the acronym id using xparse's `\NewDocumentCommand`. Inside ⟨*code*⟩ one can refer to the id with #1.

`\RenewAcroCommand{`⟨*cs*⟩`}{`⟨*code*⟩`}`
  Defines a new ACRO acronym command ⟨*cs*⟩. This sets up the necessary framework needed by acronym commands and defines ⟨*cs*⟩ with an optional star argument and a mandatory argument for the acronym id using xparse's `\RenewDocumentCommand`. Inside ⟨*code*⟩ one can refer to the id with #1.

`\DeclareAcroCommand{`⟨*cs*⟩`}{`⟨*code*⟩`}`
  Defines a new ACRO acronym command ⟨*cs*⟩. This sets up the necessary framework needed by acronym commands and defines ⟨*cs*⟩ with an optional star argument and a mandatory argument for the acronym id using xparse's `\DeclareDocumentCommand`. Inside ⟨*code*⟩ one can refer to the id with #1.

`\ProvideAcroCommand{`⟨*cs*⟩`}{`⟨*code*⟩`}`
  Defines a new ACRO acronym command ⟨*cs*⟩. This sets up the necessary framework needed by

acronym commands and defines ⟨*cs*⟩ with an optional star argument and a mandatory argument for the acronym id using xparse's `\ProvideDocumentCommand`. Inside ⟨*code*⟩ one can refer to the id with #1.

Inside these macros one can use a number of low-level expl3 commands.[14]

Acronym fetching commands

`\acro_use:n` {⟨*id*⟩}
Fetches the acronym using either the first or the long form depending on earlier uses.

`\acro_short:n` {⟨*id*⟩}
Fetches the short form of the acronym.

`\acro_long:n` {⟨*id*⟩}
Fetches the long form of the acronym.

`\acro_alt:n` {⟨*id*⟩}
Fetches the alternative short form of the acronym.

Acronym setup commands

`\acro_first_upper:`
ACRO setup command which tells the macros above that we want to uppercase the first letter of the long version. Should be used *before* one of the acronym fetching commands.

`\acro_plural:`
ACRO setup command which tells the macros above that we want to use plural forms. Should be used *before* one of the acronym fetching commands.

`\acro_indefinite:`
ACRO setup command which tells the macros above that we want to add the indefinite article. Should be used *before* one of the acronym fetching commands.

`\acro_cite:`
ACRO setup command which tells the macros above that we want to add the citation in any case independent of the option `cite`. Should be used *before* one of the acronym fetching commands.

`\acro_no_cite:`
ACRO setup command which tells the macros above that we want to have no citation independent of the option `cite`. Should be used *before* one of the acronym fetching commands.

`\acro_index:`
ACRO setup command which tells the macros above that we want to add an index in any case independent of the option `index`. Should be used *before* one of the acronym fetching commands.

---

14. Which is why you need to use them inside an expl3 programming environment. This means in the preamble surround the definitions with `\ExplSyntaxOn` and `\ExplSyntaxOff`.

Additional macros for further uses

`\acro_begin:`

When an acronym macro is defined "by hand", *i. e.*, not using `\NewAcroCommand` then this should be the first macro in the code. Must have a matching `\acro_end:`.

`\acro_end:`

When an acronym macro is defined "by hand", *i. e.*, not using `\NewAcroCommand` then this should be the last macro in the code. Must have a matching `\acro_begin:`.

`\acro_check_and_mark_if:nn` {⟨*boolean expression*⟩} {⟨*id*⟩}

Checks if the acronym ⟨*id*⟩ exists and marks it as used when ⟨*boolean*⟩ expression evaluates to `true`. This macro is used inside `\NewAcroCommand` and friends.

`\acro_check_acronym:nn` {⟨*id*⟩} {true|false}

Checks if the acronym ⟨*id*⟩ exists and marks it as used if `true` or doesn't. This macro is used inside `\acro_check_and_mark_if:nn`.

`\acro_use_acronym:n` {true|false}

Mark an acronym as used or don't. This macro is used inside `\acro_check_acronym:nn`. An acronym marked as used cannot be unmarked.

∗ `\acro_if_acronym_used:n`*TF* {⟨*id*⟩} {⟨*true*⟩} {⟨*false*⟩}

The code-level version of `\acifused`. This macro is expandable.

Examples The usage of above macros is best explained with a few examples. The following definition is done by ACRO:

```
1 \NewAcroCommand \ac { \acro_use:n {#1} }
```

An equivalent definition for `\ac` would be

```
1 \NewDocumentCommand \ac {sm}
2   {
3     \acro_begin:
4       \acro_check_and_mark_if:nn {#1} {#2}
5       \acro_use:n {#2}
6     \acro_end:
7   }
```

which should explain what the actual framework is which `\NewAcroCommand` adds.

Other definitions by ACRO are for example the following ones:

```
1 \NewAcroCommand \Ac
2   {
3     \acro_first_upper:
4     \acro_use:n {#1}
5   }
6 \NewAcroCommand \acp
7   {
8     \acro_plural:
9     \acro_use:n {#1}
10  }
11 \NewAcroCommand \Acp
12  {
13    \acro_plural:
14    \acro_first_upper:
15    \acro_use:n {#1}
16  }
17 \NewAcroCommand \Aclp
18  {
19    \acro_plural:
20    \acro_first_upper:
21    \acro_long:n {#1}
22  }
```

12. About Plural Forms, Possessive Forms and Similar Constructs – the Concept of Endings

ACRO has a concept of *endings*. All of ACRO's plural options are defined by saying

```
1 \ProvideAcroEnding {plural} {s} {s}
```

The command's syntax and what it does is as follows:

\ProvideAcroEnding{⟨*name*⟩}{⟨*short default*⟩}{⟨*long default*⟩}
  This macro defines the options

  • ⟨*name*⟩-ending,

  • short-⟨*name*⟩-ending,

  • alt-⟨*name*⟩-ending and

- `long-`⟨*name*⟩`-ending`.

It also defines the acronym properties

- `short-`⟨*name*⟩,
- `short-`⟨*name*⟩`-form`,
- `alt-`⟨*name*⟩,
- `alt-`⟨*name*⟩`-form`,
- `long-`⟨*name*⟩ and
- `long-`⟨*name*⟩`-form`.

Additionally it defines a setup macro as described in section 11, `\acro_`⟨*name*⟩`:`. If ⟨*name*⟩ contains a - (dash) it is replaced by _ before `\acro_`⟨*name*⟩`:` is built. So if you choose `my-name` the corresponding macro is named `\acro_my_name:`. If you use any other non-letters you are on your own.

Note that you *must use* `\ProvideAcroEnding` *before any acronym definition*!

This could be used together with the macros described in section 11 for adding support for possessive forms:

```
1 \ExplSyntaxOn
2 % this now only works because I've use the same already in the preamble so
3 % it does nothing here:
4 \ProvideAcroEnding {possessive} {'s} {'s}
5
6 \NewAcroCommand \acg
7   {
8     \acro_possessive:
9     \acro_use:n {#1}
10   }
11 \ExplSyntaxOff
12 The \acg{cd} booklet says\ldots
```

The CD's booklet says...

Please note that different endings are cumulative which you probably want to avoid! Imagine a macro

```
1 \NewAcroCommand \acgp
2   {
3     \acro_possessive:
```

```
4      \acro_plural:
5      \acro_use:n {#1}
6    }
```

This would give "CDs's" instead of "CDs'". To solve this you might want to consider

```
1  \ProvideAcroEnding {possessive-singular} {'s} {'s}
2  \ProvideAcroEnding {possessive-plural}   {s'} {s'}
```

A. All Acronyms Used in this Documentation

```
1 \acsetup{
2   list-type   = table,
3   list-style  = longtable,
4   list-heading = subsection*,
5   extra-style  = comma,
6   page-ref    = comma
7 }
8 \printacronyms[name=All Acronyms]
```

All Acronyms

CD	Compact Disc, pp. 8, 9, 30, 31
CTAN	Comprehensive TeX Archive Network, p. 2
ECU	Steuergerät (Electronic Control Unit), pp. 10, 11
ID	identification string, pp. 4, 5, 13, 17, 18
JPEG	Joint Photographic Experts Group, p. 9
LA	Los Angeles, p. 13
MP	Member of Parliament, p. 9
NATO	North Atlantic Treaty Organization, *deutsch*: Organisation des Nordatlantikvertrags, pp. 10, 19
NY	New York, pp. 6, 9, 12, 13, 19, 20
PDF	Portable Document Format, pp. 4, 14, 15
SW	Sammelwerk, p. 9
ST	Some Test, p. 5
UFO	unidentified flying object, p. 12

```
1 \acsetup{
2   list-type    = table,
3   list-style   = longtable,
4   list-heading = subsection*,
5   page-ref     = comma
6 }
7 \printacronyms[include-classes=city,name=City Acronyms]
```

City Acronyms

LA   Los Angeles, p. 13
NY   New York, pp. 6, 9, 12, 13, 19, 20

B. References

[Nie13] Clemens NIEDERBERGER. translations. version 1.1a, Sept. 30, 2013.
URL: http://mirror.ctan.org/macros/latex/contrib/translations/.

[Oet12] Tobias OETIKER.
acronym. *An Acronym Environment for LATEX 2ε*. version 1.38, Oct. 29, 2012.
URL: http://mirror.ctan.org/macros/latex/contrib/acronym/.

[Swi01] Matt SWIFT. abbrevs. version 1.4, Sept. 8, 2001. URL:
http://mirror.ctan.org/macros/latex/contrib/frankenstein/abbrevs/.

[Tal12] Nicola L.C. TALBOT. mfirstuc. version 1.05, May 21, 2012.
URL: http://mirror.ctan.org/macros/latex/contrib/glossaries/.

[Tal13] Nicola L.C. TALBOT. glossaries. version 3.05, Apr. 21, 2013.
URL: http://mirror.ctan.org/macros/latex/contrib/glossaries/.

[Tho95] Paul A. THOMPSON. acromake. version na, July 16, 1995.
URL: http://mirror.ctan.org/macros/latex/contrib/acromake/.

[Vey+05] Boris VEYTSMAN et al.
nomencl. *A Package to Create a Nomenclature*. version 4.2, Sept. 22, 2005.
URL: http://mirror.ctan.org/macros/latex/contrib/nomencl/.

[Vos10] Jakob VOSS. acroterm. version 0.1, Nov. 19, 2010.
URL: http://mirror.ctan.org/macros/latex/contrib/acroterm/.

[Wik12a] WIKIPEDIA. *Acronym and initialism*. 2012.
URL: http://en.wikipedia.org/wiki/Acronyms (visited on 06/21/2012).

[Wik12b] WIKIPEDIA. *New York City*. 2012.
URL: http://en.wikipedia.org/wiki/New_York_City (visited on 09/27/2012).

C. Index

Index