The `checkcites`* script

Enrico Gregorio
`Enrico dot Gregorio at univr dot it`

Paulo Roberto Massa Cereda
`cereda at users dot sf dot net`

Contents

1 Introduction

`checkcites` is a Lua script written for the sole purpose of detecting unused or undefined references from both LaTeX auxiliary or bibliography files. We use *unused reference* to refer to the reference present the bibliography file – with the `.bib` extension – but not cited in the `.tex` file. The *undefined reference* is exactly the opposite, that is, the items cited in the `.tex` file, but not present in the `.bib` file.

The original idea came from a question posted in the TeX.sx community about how to check which bibliography entries were not used. We decided to write a script to check references. We opted for Lua, since it's a very straightforward language and it has an interpreter available on every modern TeX distribution.

Attention!

`checkcites` is known to run with the most recent `texlua` and `lua` interpreters. Unfortunately, the code is incompatible with interpreters prior to the Lua 5.1 language specification.

2 How the script works

`checkcites` uses the generated `.aux` file to start the analysis. The first step is to extract all citations found, in the form of `\citation{a}`. For every `\citation` line found,

*Version 1.0i from December 18, 2012.

`checkcites` will extract the citations and add them to a table, even for multiple citations separated by commas, like `\citation{a,b,c}`. Then the citations table has all duplicate values removed – in other words, the table becomes a set. Let's call A the set of citations.

Attention!

If `\citation{*}` is found, `checkcites` will issue a message telling that `\nocite{*}` is in the `.tex` document, but the script will do the check nonetheless.

At the same time `checkcites` also looks for bibliography data, in the form of `\bibdata{a}`. Similarly, for every `\bibdata` line found, the script will extract the bibliography data and add them to a table, even if they are separated by commas, like `\bibdata{d,e,f}`. The table has all duplicate values removed.

Attention!

If no `\bibdata` command is found, the script ends. There's nothing to do in this case.

Now, `checkcites` will extract all entries from the bibliography files found in the previous step. For every element in the bibliography data table, the script will look for entries like `@BOOK`, `@ARTICLE` and so forth – we actually use pattern matching for this – and add their identifiers to a table. The script treats all `.bib` files as if they were only one. After all files have been analyzed and all entries' identifiers extracted, the table has all duplicate values removed. Let's call B the set of bibliography entries.

Attention!

If `checkcites` cannot find a certain bibliography file – that is, a `.bib` file – the script ends. Make sure to put the correct name of the bibliography file in your `.tex` file.

Now we have both sets A and B. In order to get all unused references in the `.bib` files, we compute the set difference

$$B - A = \{x : x \in B, x \notin A\}.$$

Similarly, in order to get all undefined references in the `.tex` file, we compute the set difference

$$A - B = \{x : x \in A, x \notin B\}.$$

If there are either unused or undefined references, `checkcites` will print them in a list format. In Section 3 there's a more complete explanation on how to use the script.

3 Usage

`checkcites` is very easy to use. First of all, let's define two files that will be used here to explain the script usage. Here's our sample bibliography file `example.bib`, with five fictional entries.

Bibliography file

```
@BOOK{foo:2012a,
  title = {My Title One},
  publisher = {My Publisher One},
  year = {2012},
  editor = {My Editor One},
  author = {Author One}
}

@BOOK{foo:2012b,
  title = {My Title Two},
  publisher = {My Publisher Two},
  year = {2012},
  editor = {My Editor Two},
  author = {Author Two}
}

@BOOK{foo:2012c,
  title = {My Title Three},
  publisher = {My Publisher Three},
  year = {2012},
  editor = {My Editor Three},
  author = {Author Three}
}

@BOOK{foo:2012d,
  title = {My Title Four},
  publisher = {My Publisher Four},
  year = {2012},
  editor = {My Editor Four},
  author = {Author Four}
}

@BOOK{foo:2012e,
  title = {My Title Five},
  publisher = {My Publisher Five},
  year = {2012},
  editor = {My Editor Five},
  author = {Author Five}
}
```

The second file is our main LaTeX document, `document.tex`.

Main document

```
\documentclass{article}

\begin{document}

Hello world \cite{foo:2012a,foo:2012c}, how are you \cite{foo:2012f},
and goodbye \cite{foo:2012d,foo:2012a}.

\bibliographystyle{plain}
\bibliography{example}

\end{document}
```

Open a terminal and run `checkcites`:

```
$ checkcites
```

```
checkcites.lua -- a reference checker script (v1.0i)
Copyright (c) 2012 Enrico Gregorio, Paulo Roberto Massa Cereda

Usage: checkcites.lua [--all | --unused | --undefined] file.aux

--all         Lists all unused and undefined references.
--unused      Lists only unused references in your 'bib' file.
--undefined   Lists only undefined references in your 'tex' file.

If no flag is provided, '--all' is set by default.
Be sure to have all your 'bib' files in the same directory.
```

If you don't have `checkcites` installed with your TEX distribution, you can run the standalone script `checkcites.lua` with either `texlua` or `lua`. We recommend to use `texlua`, as it's shipped with all the modern TEX distributions:

```
$ texlua checkcites.lua
```

When you run `checkcites` without providing any argument to it, the script usage will be printed, as seen in the previous output. The only required argument is the auxiliary file – with the `.aux` extension – which is generated when you compile your main `.tex` file. For example, if your main document is named `foo.tex`, you probably have a `foo.aux` file too. Let's compile our sample document `document.tex`:

```
$ pdflatex document.tex
```

After running `pdflatex` on our `.tex` file, there's now a `document.aux` file in our work directory.

Auxiliary file

```
\relax
\citation{foo:2012a}
\citation{foo:2012c}
\citation{foo:2012f}
\citation{foo:2012d}
\citation{foo:2012a}
\bibstyle{plain}
\bibdata{example}
```

Now we can run `checkcites` on the `document.aux` file:

```
$ checkcites document.aux
```

```
checkcites.lua -- a reference checker script (v1.0i)
Copyright (c) 2012 Enrico Gregorio, Paulo Roberto Massa Cereda

I found 4 citation(s).
Great, there's only one 'bib' file. Let me check it.
I found 5 reference(s).

Unused reference(s) in your bibliography file(s): 2
- foo:2012b
- foo:2012e

Undefined reference(s) in your TeX file: 1
- foo:2012f
```

As we can see in the script output, `checkcites` analyzed both `.aux` and `.bib` files and found two unused references in the bibliography file – `foo:2012b` and `foo:2012e` – and one undefined reference in the document – `foo:2012f`.

`checkcites` allows a command line switch that will tell it how to behave. For example,

```
$ checkcites --unused document.aux
```

The `--unused` flag will make the script only look for unused references in the `.bib` file. The argument order doesn't matter, you can also run

```
$ checkcites document.aux --unused
```

The script will behave the same. Similarly, you can use

```
$ checkcites --undefined document.aux
```

The `--undefined` flag will make the script only look for undefined references in the `.tex` file. If you want `checkcites` to look for both unused and undefined references, run:

```
$ checkcites --all document.aux
```

If no special argument is provided, the `--all` flag is set as default.

4 License

This script is licensed under the LaTeX Project Public License. If you want to support LaTeX development by a donation, the best way to do this is donating to the TeX Users Group.

Official code repository

http://github.com/cereda/checkcites