Examples Document

Reverse engineerable LaTeX

Steffan Sølvsten

201505832@post.au.dk

January 31, 2017

Abstract

The following document is meant to show the use of the various packages used in the preamble. They are all created with the intent to be reverse engineerable.

Contents

1 Math

Most basically you use *equation* to write a mathematical equation on a single new and centered line or you can write math in the text by writing $ x_0 $ = x_0. If you want to write math over more lines, it is better to use the *align* or even better the *alignat* commands. In the following example with *alignat*, we get the ability to align on more things, such that both the arrows on the left and the equals sign are aligned nicely. Here is a reference to line 2.

$$\begin{aligned} & cosx = cosx * cosy && (1)\\ \Updownarrow & \\ & 0 = cosx * cosy - cosx && (2) \end{aligned}$$

Her er én linje midt i *alignat* environment, hvormed linjer stadig er aligned på tværs af indskydende tekst. Denne linje får dog også et linjenummer, hvilket er fjernet med *nonumber*

$$\begin{aligned} \Updownarrow & \\ & 0 = cosx(cosy - 1) && (3) \end{aligned}$$

With *pmatrix* matrices can also be made, such as the general matrix below

$$\begin{pmatrix} v_1 \cdot v_1 & v_1 \cdot v_2 & \dots & v_1 \cdot v_n \\ v_2 \cdot v_1 & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ v_n \cdot v_1 & \dots & \dots & v_n \cdot v_n \end{pmatrix}$$

Things like sequences and sets we use all the time too, which is why I've made macros. For a sequence you can write $a_1, a_3, \dots, a_n$ and for sets $\{x \in \Sigma^* \mid |x| \geq 42\}$. Both of these shortcuts are overloaded, such that they can take fewer arguments than currently shown here.

2 Logic and proofs

With logic and more you want to make a *theorem*, *lemma*, *proposition*, *corollary* or *conjecture* followed possibly by a *proof*. These are all currently implemented in the localized preamble to associate the same command to both languages. The proof has a label, which is 2.1.

Lemma 2.1 (Reference to [2])**.** Here we have a lemma, which is currently set up to start with the sectionnumbering and then followed by a unique number. This can be changed in the language specific preamble file

Proof. In here we can write a proof, which is automatically going to be appended a tombstone. Here we could also add an equation a list.

$$2 + 2 = 4$$

You can also place a fitch-style proof as described below, though there possibly needs to be a linebreak after it to place the tombstone correctly at the end. □

2.1 lplfitch

There exists various packages to make make logic proofs, but after some time looking around I've chosen to focus on the *lplfitch* package, since its output seems to be the generally preferred and used style. This package is very easy to work with, when first learned, but until then you need to keep learn quite a lot of new syntax. Most importantly notice, that a proof and subproof has two arguments, the assumption and the conclussions.

```
| 1. presumption
|-
| 2. conclussion
|   | 3. assumption
|   |-
|   | 4. conclussion                    argument
|   | 5. conclussion
| 6. konklussion
```

All the commands can be found in the documentation, which can be found *here*, but most shortcuts are of the form: $"l" + type + "i/e"$. Argumentation is also done using these shortcuts. Every line is of the form

$$pline[hlinenumberi]\{hformulai\}[hjustificationi]$$

Notice, that in fitch-style there isn't normally written assumption or premise, but it is merely shown by the horizontal line. You can write it if you want. The proof to exercise 1.2.1 c from TØ class is in the proof to lemma 2.2.

Lemma 2.2. $(p \land q) \land r \vdash p \land (q \land r)$

Proof.

1.	$(p \land q) \land r$	**Premise**
2.	r	$\land$**Elim:** 1
3.	$p \land q$	$\land$**Elim:** 1
4.	p	$\land$**Elim:** 3
5.	q	$\land$**Elim:** 3
6.	$q \land r$	$\land$**Intro:** 2, 5
7.	$p \land (q \land r)$	$\land$**Intro:** 4, 6

□

If you need a fresh variable, you don't create a subproof, but instead a boxed subproof. The solution to an exercise in one of the handins is then

Theorem 2.1. $\forall x(P(x) \to Q(x) \vdash (\forall x \neg Q(x)) \to (\forall x \neg P(x)))$

Proof.

1.	$\forall x\,(P(x) \to Q(x))$	
2.	$\forall x\,\neg Q(x)$	
3.	$\boxed{x_0}\ \neg Q(x_0)$	
4.	$P(x_0) \to Q(x_0)$	$\forall$**Elim:** 1, 3
5.	$\neg P(x_0)$	**Modus Tollens:** 3, 4
6.	$\forall x\,\neg P(x)$	$\forall$**Intro:** 3–5
7.	$(\forall x\,\neg Q(x)) \to (\forall x\,\neg P(x))$	$\to$**Intro:** 2–6

□

2.2 logicproof

Alternatively you can use the *logicproof* package, which uses a syntax more similar to other LaTeX. The output also resembles the style in the Logic book used in the course dBerLog. [3] You have to choose one of the two packages due to conflicts. Currently this one is turned off, but the proof from theorem 2.1 should look something like this, but I couldn't get it to compile.

```
\begin{logicproof}{2} %Amount of subproofs
  \forall x (P(x) \to Q(x))          & premise
\\ \begin{subproof}
    \forall x \lnot Q(x)             & assume
      \begin{subproof}
        \llap{x_0} \lnot Q(x_0)      & fresh
      \\  P(x_0) \to Q(x_0)          & \forall x \,
          \mathrm{e} 1, 3
      \\  \lnot P(x_0)               & Modus Tollens
      \end{subproof}
    \forall x \, \lnot P(x)          & \forall x \,
       \mathrm{i} 3--5
    \end{subproof}
  (\forall x \, \lnot Q(x)) \to (\forall x \, \lnot P(x))
                                     & \to \mathrm{i} 2--6
\end{logicproof}
```

3 Figures

The following is a figure with an image in it, and its label is 1. Figures can contain pretty much everything, so just experiment, it will most likely work.

Figure 1: A picture

If you want to have your figure beside your text you need to put it into a *wrapfigure* instead of a normal *figure*. Place the text at the line, on which you want the figure to start. The first variable are the amount of lines the box is high, the second is left or right, while the last is the width.

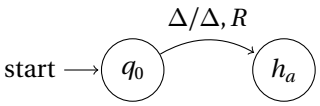
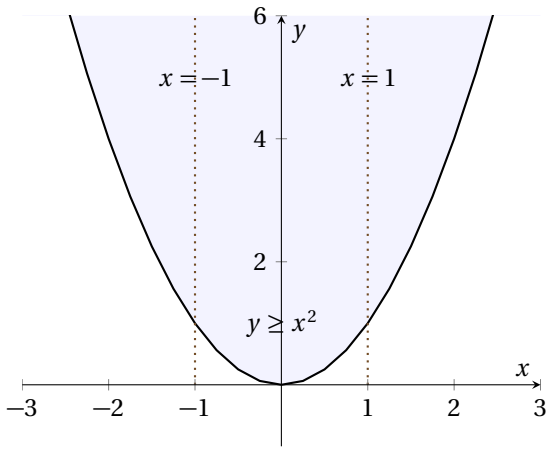


Figure 2: Transition diagram of a small Turing machine

4 Graphs

Here is a graph from our first Calculus handin, kindly sponsored by the wonderful Rasmus Skovdal. [1] This can also be inserted into a figure, with which there are also captions and reference options.



5 Code

The following is sourcecode for some non-aweinspiring Java method. By using a caption you also give it a number, but alternatively it can be given a *title* instead of a *caption*. Using *title* does break the ability to make and reference a *label*, but that is your intent anyways, if you do this. The following code has the label 1

Code 1: I'm a caption

```
1 //This is a comment, nordic letters are not supported
2 public static String example(int n) {
3         return "You wrote: "+n;
4 }
```

If the language has to be something else than is the standard specified, then it has to be declared as an argument. In the following pseudocode there are also used escape character *@ and @* to insert LaTeX math.

Code 2: The algorithm *linear exponentiation*

```
1  Algorithm:  Linear Exponentiation (x,p)
2  Input     :  p ≥ 0
3  Output    :  r = x^p
4  Method    :  r ← 1
5               q ← p
6               {I} while  q > 0  do
7                       r ← r * x
8                       q ← q − 1
```

6 Trees

With qtree we have a quick and easy way to draw trees. Notice, that you need to have a space between an element and a closing bracket.

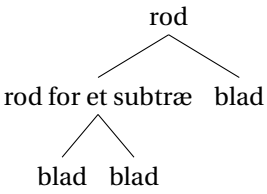


Figure 3: A tree

If the trees need to be a bit more complex, then you need to use tikz, where it is currently in the preamble set up to create red-black trees.

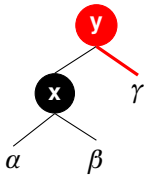


Figure 4: A nicer tree

7 Automata

By using Tikz creating automota by hand is easy, and I highly encourage you do it manually, to make modifications later much easier and your document much nicer in general. In the following the only complication has been the longer edge from (C) to (A) over 1, where in the code commented out it is explicitly defined by guiding points, but it could also be done by changing the angle, which is by default 30 degrees.

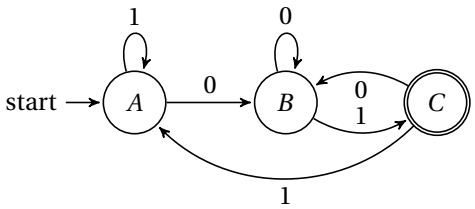
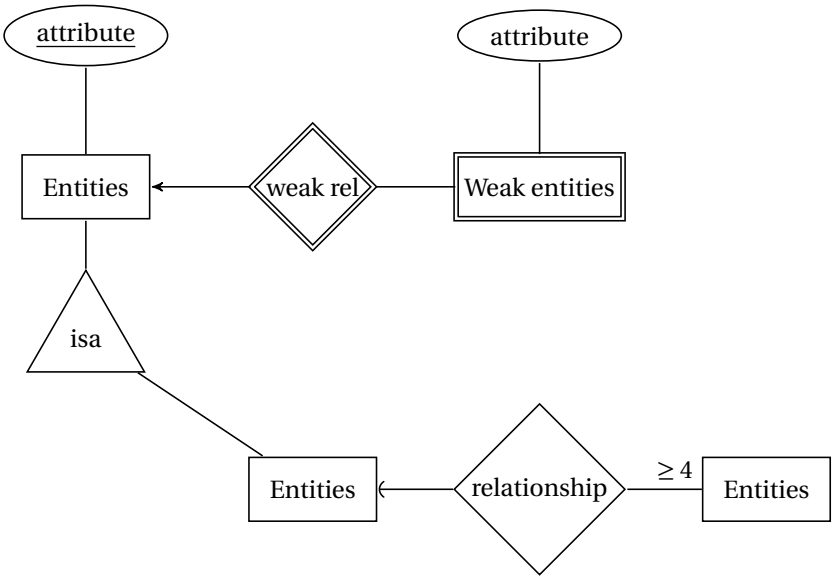


Figure 5: An automoata, that accepts strings ending with "01"

8 E/R Diagrams

By using *er* and *shapes* packages a E/R diagrammer can be drawn. These are made in the exact same way as the automota.



9 References

If you need to reference equations, figures, sections or anything with a label you want to use the *ref* operation. A *ref* will always point to a *label* defined and saved in the earlier compilation. For example this section has the number 9.

If you wan to reference the bibliography, then you want to use the *cite* operation instead. Here is a reference to [3], which is written in the *bibliography* further down. This is redefined to litteratur with the danish preamble, preamble_dk.tex. If you want more references simultaneously the you just seperate them with commas: [3, 2]

10 Quotes

With the *csquotes* package it is possible to make nice quotes, such as Steffan's remark "We are computer scientists, not vampires" [2, s. 12]. By using *blockquote* you get automatically a quote, which is on its own line, if it is 4+ lines long.

> Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat. Duis aute irure dolor in reprehenderit in voluptate velit esse cillum dolore eu fugiat nulla pariatur. [3, s. 1]

Notice, that both quotes used a *c* as a prefix to *quote*. This gives the possibility to reference to the bibliography. If you don't need that you can safely remove it.

11 Columns

Here is some text split into two columns, which can be used for various things. I have not used this very often, so I'm not sure if it works well with various stuff prior shown, such as figures, code or equations. If you want to use code, you will have to at least not include a header and remove the centering option as set up in the preamble.

Here is some more text in the other column. Over here we could have the index, while on the other side we have the title and also the abstract?

References

[1] Skovdal, Rasmus: *Calculus 1*, 2015

[2] Jørgensen, Steffan: *101 quotes*, 2015

[3] Huth, Michael og Ryan, Mark: *Logic in Computer Science*, second edition, 2004

[4] You can either write the bibliography in this document or have an extra references.bib file with all the information in the following form:

@bookFilVri97, Author = Filar, Jerzy and Vrieze, Koos, Publisher = Springer, Title = Competitive Markov Decision Processes, Year = 1997

A An Appendix

Lorem ipsum dolor sit amet, consectetur adipiscing elit, sed do eiusmod tempor incididunt ut labore et dolore magna aliqua. Ut enim ad minim veniam, quis nostrud exercitation ullamco laboris nisi ut aliquip ex ea commodo consequat.