Reverse engineerable LaTeXexamples

Steffan Sølvsten

Aarhus University
soelvsten@cs.au.dk

February 17, 2020

Abstract

This document is created with the intention to make the learning curve of switching to LaTeX simpler. It does this by giving an example or two of all types of writing usually done as part of a handin, such as math, code, proofs and much more. The code for all examples is made as easy to read as possible.

Contents

1 Math

Most basically you use *equation* to write a mathematical equation on a single new and centered line, such as the following

$$x = y \tag{1}$$

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. You can also use the * in the \begin{align} to not include line numbers

$$\begin{aligned}(R' \circ R)^{-1} &= (\{(a,b) \in S \times U | \exists t \in T : (a,t) \in R \wedge (t,b) \in R'\})^{-1} \\ &= \{(p,q) \in U \times S | (q,p) \in R \circ R'\} \\ &= \{(p,q) \in U \times S | \exists t \in T : (p,t) \in R'^{-1} \wedge (t,q) \in R^{-1}\} \\ &= R'^{-1} \circ R^{-1}\end{aligned}$$

In the following example with *alignat*, we get the ability to have multiple alignments in the same column, such that both the arrows on the left and the equals sign are aligned nicely. Here is a reference to line 3.

$$\begin{aligned}&\cos x = \cos x * \cos y && (2)\\ \Updownarrow\quad &\\ &0 = \cos x * \cos y - \cos x && (3)\\ \Updownarrow\quad &\\ &0 = \cos x \cdot (\cos y - 1) && (4)\end{aligned}$$

1.1 Math fonts and symbols

In mathmode you can also use different styles called alphabets. Bold **b** and italic text such as functions can be written with *function*. Furthermore the real numbers $\mathbb{R}$ has a shortcut \R and the same goes for $\mathbb{C}$, $\mathbb{Q}$, $\mathbb{F}$and more. All these macros ensure to go into math mode, so you will not get any compile errors in plain writing. Finally with *mathcal* you can get caligraphic writing.

In general you would like to define macros for whenever you are going to reuse the same notation over and over again. For example if you have to write about the NP-complete problem SAT multiple times, you may want to define SAT as a local macro, since it is not part of the preamble.

1.2 Linear Algebra

With *pmatrix* matrices can also be made, such as in the following. For the row operations I have made the *ero* macro to make life easier.

$$\left(\begin{array}{cc|c} 2 & 1 & 1 \\ 1 & 1 & 0 \end{array}\right) \overset{\substack{R_1 - R_2 \\ \sim \\ 2R2}}{} \left(\begin{array}{cc|c} 1 & 0 & 1 \\ 2 & 2 & 0 \end{array}\right)$$

Other common (Linear) Algebra operators that I have made macros for are $\operatorname{Det}(A)$, the equivalence class $[v]_{\mathcal{V}}$, the matrix representation ${}_{\mathcal{V}}[L]_{\mathcal{W}}$, the $\operatorname{sgn}(\sigma)$ and the $\operatorname{Span}(v_1, v_2, \ldots, v_n)$ can be made easily with the macros coded. The vectors $\mathbf{1}$ and $\mathbf{0}$ also have a shortcut.

1.3 Other math macros

Things like sequences and sets we use all the time too, which is why I've made macros. For a sequence you can write $a_1, a_3, \ldots, a_n$ 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 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 lemma below has a label, which is 2.1 and can be used to reference it.

Lemma 2.1 (Martin 2.11)**.** $\forall x \in \Sigma^* \forall (p,q) \in Q : \delta^*((p,q),x) = (\delta_1^*(p,x), \delta_2^*(q,x))$

Proof. We prove this by induction in the construction of x.

In the base case of x being the empty string Λ

$$\begin{aligned} \delta^*((p,q),\Lambda) &= (p,q) && \text{def. 2.12} \\ &= (\delta_1^*(p,\Lambda), \delta_2^*(q,\Lambda)) && \text{def. 2.12} \end{aligned}$$

Now assume per induction that for $y \in \Sigma^*$ we have that $\delta^*((p,q),y) = (\delta_1^*(p,y), \delta_2^*(q,y))$ and consider $x = y\sigma$ for $\sigma \in \Sigma$.

$$\begin{aligned} \delta^*((p,q),y\sigma) &= \delta(\delta^*((p,q),y),\sigma) && \text{def. 2.12} \\ &= \delta((\delta_1^*(p,y), \delta_2^*(q,y)),\sigma) && \text{I.H.} \\ &= (\delta_1(\delta_1^*(p,y),\sigma), \delta_2(\delta_2^*(q,y),\sigma)) && \text{def. af } \delta \\ &= (\delta_1^*(p,y\sigma), \delta_2^*(q,y\sigma)) && \text{def. 2.12} \end{aligned}$$

□

3 Formal proofs

3.1 proof trees

The `bussproof` package allows to create proof trees. Usually this is done with the command *prooftree*, but should it not have a line break after, then there also is created the command *bprooftree* as shown in figure 1.

$$\frac{}{\mathsf{leaf} \sim\!=\!\sim \mathsf{leaf}}\ (\text{MLeaf}) \qquad \frac{c_1 \sim\!=\!\sim c_1' \qquad c_2 \sim\!=\!\sim c_2'}{\mathsf{node}\ c_1\ x\ c_2 \sim\!=\!\sim \mathsf{node}\ c_1'\ x'\ c_2'}\ (\text{MNode})$$

Figure 1: Inductive relation for trees of the same shape

3.2 logic proofs

There exists various packages to 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 learn quite a lot of new syntax. Most importantly notice, that a proof and subproof has two arguments, the assumption and the conclusions.

1. presumption	
2. conclussion	
3. assumption	
4. conclussion	argument
5. conclussion	
6. conclussion	

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

$$\backslash pline[\text{linenumber}]\{\text{formula}\}[\text{justification}]$$

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 3.1.

Lemma 3.1. $(p \wedge q) \wedge r \vdash p \wedge (q \wedge r)$

Proof.

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

□

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

Theorem 3.2. $\forall x(P(x) \to Q(x) \vdash (\forall x \neg Q(x)) \to (\forall x \neg P(x)))$

Proof.

1. ...	
2. $\boxed{\mathsf{x}_0}$	$\neg\mathsf{Q}(\mathsf{x}_0)$
3. ...	
4. ...	

□

4 Tables and figures

The following is a figure with an image in it, and its label is 2. Figures can contain pretty much everything, so just experiment, it will most likely work. 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.

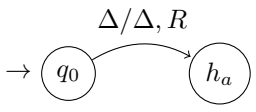


Figure 3: Transition diagram of a small Turing machine

Figure 2: A picture

4.1 Tables

Tables are also put into a float similar to figures, which makes it possible to add captions and references to it similar to before, such as the table in 1. The table itself is made with *tabular*.

l-column	r-column	c-column	gray column	light blue column
a	b	c	d	e
f	g	h	i	j
$k = \frac{1}{2}$	l	m	n	o

Table 1: A table with this being the caption

4.2 Graphs

Here is a graph from our first Calculus handin, kindly sponsored by the wonderful Rasmus Skovdal. This can also be inserted into a figure, with which there are also captions and reference options, such as in figure 4.

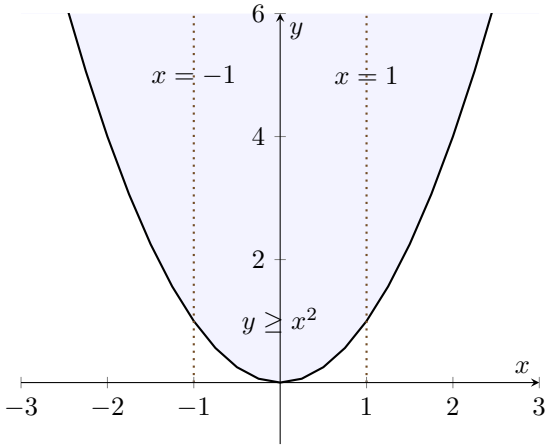
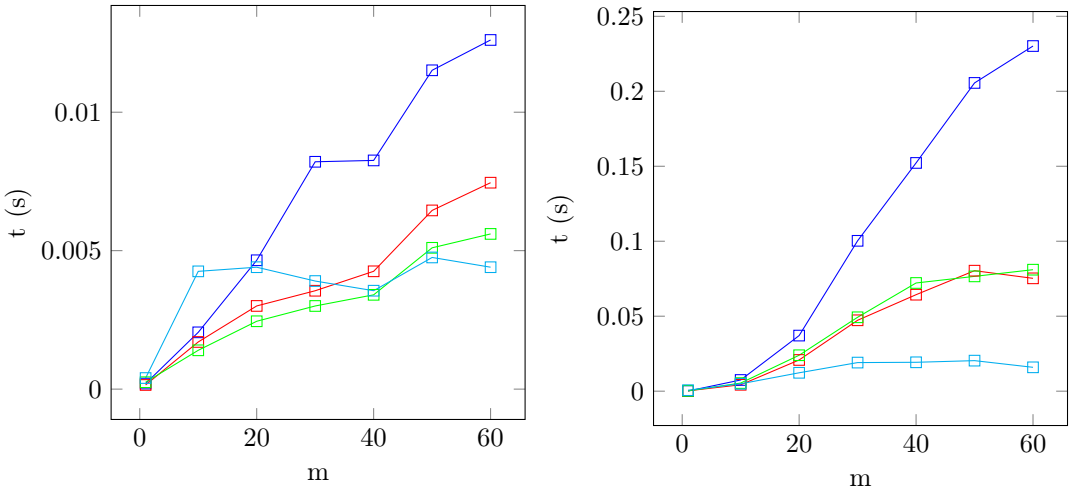


Figure 4: A pretty graph by Rasmus Skovdal

In Figure 5 you can find timing of a Simplex Algorithm, where sub Figure 5a contains the data for $n = 10$. Since the code for each figure also contains all the data, we have chosen to

move the code into the sub folder *figures*. From experience I will heavily endorse everyone else to do the same.



(a) $n = 10$ (b) $n = 30$

Figure 5: Performance with Bland's rule (blue), largest coefficient rule (red), largest increase rule (green), and SciPy (cyan).

4.3 Graphs and automata

In Figure 6 you can find a small game gadget used by Hansen and Sølvsten in [HS20]. This figure was written by hand and the code is separated into *polynomia.tex* in the *figures* folder. I highly encourage you to also make your figures manually rather than using any tool that autogenerates it (IPE being the only exception). This makes it much easier to make modifications later and your document ends up more easier to cope with in general.

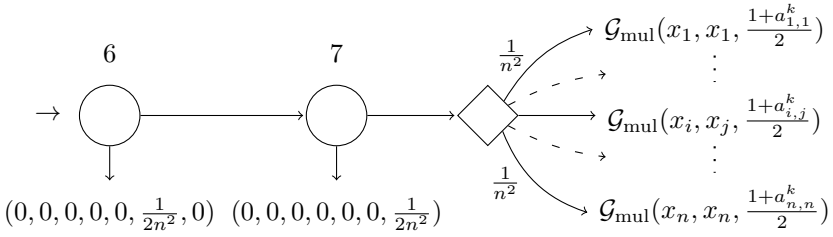


Figure 6: A game between multiple players on a graph

While the above already gives you enough tools in your toolkit to create an automata, the preamble also includes the *automata* package to make it much easier. An example is shown in Figure 7

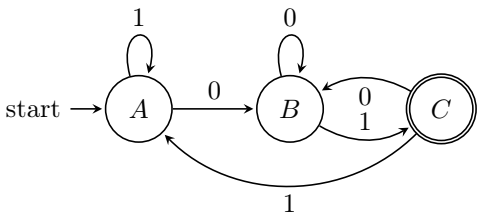


Figure 7: A finite automoata, that accepts strings ending with "01"

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 used *title* in the first place. The following code has the label 1

```
1 //This is a comment, nordic letters are supported (æ, œ, å)
2 public static String example(int n) {
3   return "You wrote: "+n;
4 }
```

Code 1: I'm a caption

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 also is math with the $ symbols and you can also escape to all of LaTeX with *@ and @*.

```
1 *@ \textbf{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
```

Code 2: The algorithm *linear exponentiation*

6 Referencing and Citing

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 prior compilation. For example this section has the number 6.

If you wan to reference the bibliography, then you want to use the *cite* operation instead. Here is a reference to [HS20], 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 you just seperate them with commas.

References

[HS20] Kristoffer Arnsfelt Hansen and Steffan Christ Sølvsten. "$\exists\mathbb{R}$-Completeness of Stationary Nash Equilibria in Perfect Information Stochastic Games". In: *Yet in review* (Feb. 2020).

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.