Reverse engineerable LaTeXexamples

Steffan Sølvsten*

Aarhus University
soelvsten@cs.au.dk

December 21, 2021

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.

*Together with numerous many other beautiful people at Aarhus Univesity.

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 * at the end of the environment name 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.

$$\cos x = \cos x * \cos y \tag{2}$$

$$\Updownarrow$$

$$0 = \cos x * \cos y - \cos x \tag{3}$$

$$\Updownarrow$$

$$0 = \cos x \cdot (\cos y - 1) \tag{4}$$

1.1 Math fonts and symbols

In mathmode you can also use different styles called alphabets. Bold $\mathbf{b}$ and italic text such as functions can be written with *function*. Finally with *mathcal* you can get calligraphic writing, such as $\mathcal{G}$ and with *mathbb* you get math symbols for fields, like $\mathbb{F}$, and other things such as $\nvdash$.

Furthermore there are quite a few symbols are you normally have to write, such as the real numbers $\mathbb{R}$. Hence, the preamble defines multiple shortcuts in *math_macros_symb* for $\mathbb{C}$, $\mathbb{Q}$, $\mathbb{F}$ and many more. All these macros ensure to go into math mode, so you will not get any compile errors in plain writing. The file is easy to read and is well documented.

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 PARTITION multiple times, you may want to define PARTITION 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 How to easily find symbols

One good place to find the LaTeX command for a symbol is at detexify.kirelabs.org/classify.html.

1.4 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*. To not have an immediate line break after, then the preamble also defines the environment *bprooftree* that can be used as shown in Figure 1.

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

Figure 1: Inductive relation for trees of the same shape

To fit more complicated typing rules, such as the *T-ODot* typing rule of Zhang and Kifer [ZK17], onto a single page or on a beamer slide one may may need to use the `array` environment inside the text math. An example can be seen in Figure 2.

$$\frac{\begin{array}{l}\Gamma \vdash e_1 : num_{\mathbb{d}_1} \\ \\ \Gamma \vdash e_2 : num_{\mathbb{d}_2}\end{array} \qquad \Phi \implies \begin{pmatrix} e_1 \odot e_2 \\ \Updownarrow \\ e_1 + \mathbb{d}_1 \odot e_2 + \mathbb{d}_2 \end{pmatrix}}{\Gamma \vdash e_1 \odot e_2 : bool_0}$$

Figure 2: The *T-ODot* typing rule for comparisons.

One is of course not restricted to writing mere proof rules, but can also write bigger proof trees, such as the small examples in Figure 3.

$$\dfrac{\dfrac{\dfrac{}{\Gamma \vdash 0 : num_0}\,\text{(T-Num)} \qquad \Gamma \vdash e : num_{\mathbb{d}}}{\Gamma \vdash 0 - e : num_{0-\mathbb{d}}}\,\text{(T-OPlus)}}{\Gamma \vdash -e : num_{-\mathbb{d}}}$$

Figure 3: Proof of `T-UMinus`

3.2 logic proofs

There exists various packages to make logic proofs, and due to coursework at Aarhus University this preamble includes the *logicproof* package. A basic proof is in this

1.	*mathenvironment*	rule
2.	p	assumption
3.	q	...
4.	q	rule i_2

A full example of a proof for $(p \vee q) \vee r \vdash p \vee (q \vee r)$ would then be

1.	$(p \vee q) \vee r$	premise
2.	$(p \vee q)$	assumption
3.	p	assumption
4.	$p \vee (q \vee r)$	$\vee$i$_1$, 3
5.	q	assumption
6.	$q \vee r$	$\vee$i$_1$, 5
7.	$p \vee (q \vee r)$	$\vee$i$_2$, 6
8.	$p \vee (q \vee r)$	$\vee$e, 2, 3–4, 5–7
9.	r	assumption
10.	$q \vee r$	$\vee$i$_2$, 9
11.	$p \vee (q \vee r)$	$\vee$i$_2$, 10
12.	$p \vee (q \vee r)$	$\vee$e, 1, 2–8, 9–11

3.2.1 lplfitch

At first this preamble was using the *lplfitch* package, since its output seems to be the generally preferred and used style compared to the box proves above. The syntax for this package is

much less intuitive to use and has a much steeper learning curve. See the older versions of this document for how it looks and how it is works.

4 Tables and figures

(a) An image

(b) Another image

Figure 4: Some pictures in subfigures

The following is a figure with subfigures with an image in it, and its label is 4. 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.

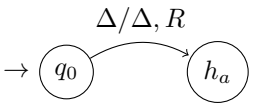


Figure 5: Transition diagram of a small Turing machine

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 in Table 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 6.

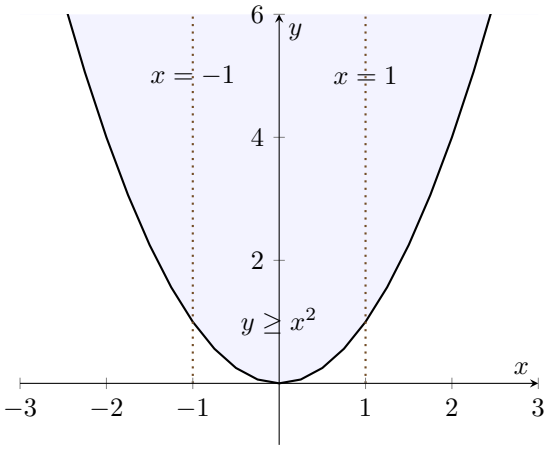
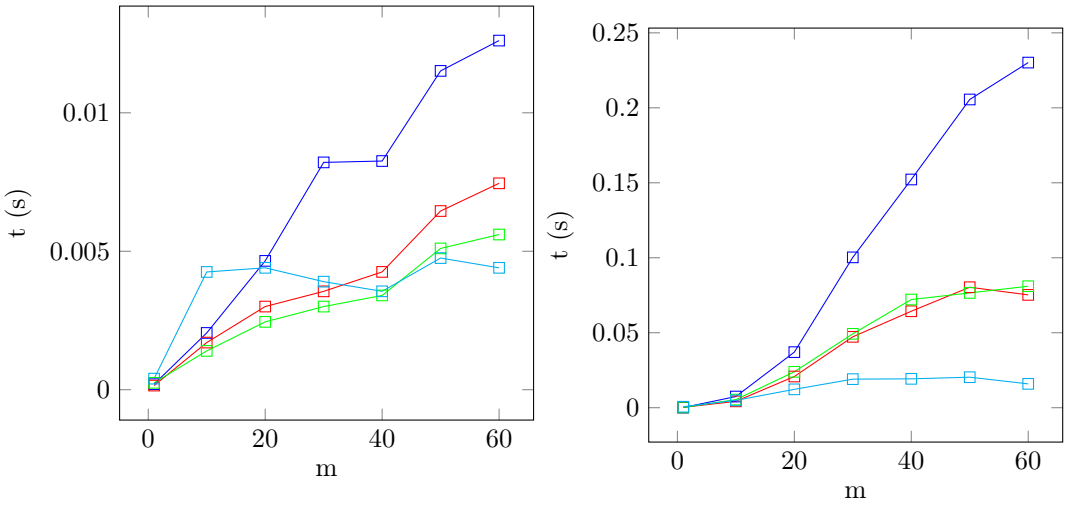


Figure 6: A pretty graph by Rasmus Skovdal

In Figure 7 you can find timing of a Simplex Algorithm, where sub Figure 7a 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 7: Performance with Bland's rule (blue), largest coefficient rule (red), largest increase rule (green), and SciPy (cyan).

4.3 Graphs and automata

In Figure 8 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.

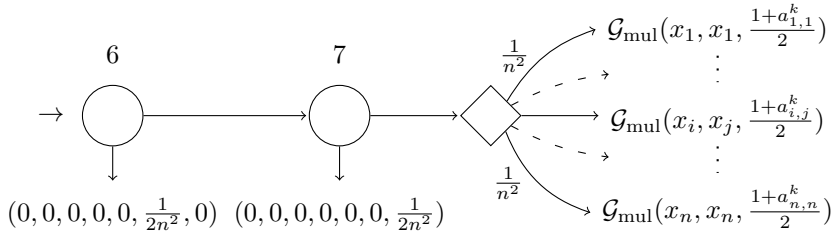


Figure 8: A game between multiple players on a graph

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 9

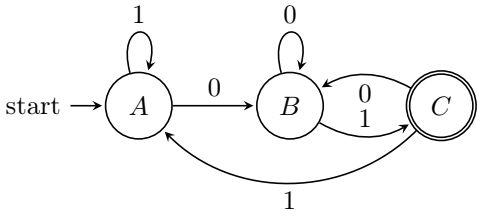


Figure 9: A finite automoata, that accepts strings ending with "01"

4.4 Derivation trees

Simple trees can be drawn using the *qtree* package. It is really only good enough for derivation trees

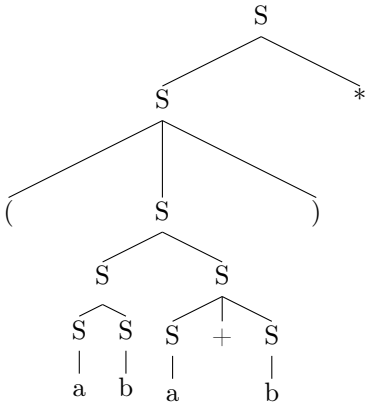


Figure 10: A derivation of $(abc + a)*$ on the grammar $S \mapsto SS|S + S|S * |(S)|a|b|0$

4.5 Math and grammars

In some cases, one may want to include a grammar for a language within a figure, such as the context-sensitive language in the figure below. This is merely done, by placing an *equation* or *align* within a figure.

$$\begin{aligned} S &\rightarrow TD_2 \mid \Lambda \\ T &\rightarrow ABTAB \mid D_1 \\ bDa &\rightarrow ba \\ BA &\rightarrow AB \\ BD_i &\rightarrow bD_i \\ Bb &\rightarrow bb \\ Ab &\rightarrow ab \\ Aa &\rightarrow aa \end{aligned}$$

Figure 11: A context-sensitive grammar for the language $\{a^n b^n a^n b^n \mid n \geq 0\}$

5 Lists and enumeration

You can make a list by use of *itemize* and enumerated lists with *enumerate*. These can also be nested within each other easily.

- First item
  - First subitem
  - Second subitem
- Second item
  1. First enumerated subitem
  2. Second enumerated subitem
- Third item

Finally, the choice of bullet points can be customized, such as the following with parenthesis

(a) First option

(b) Second option

6 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   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 1: The algorithm *linear exponentiation*

The preamble contains the ability to place code into floats similar to figures and tables, which can be seen above. For this you want to use the custom *blstlisting* environment rather than the default *lstlisting* one inside a *lstfloat*. The only immediate problem is, that the float counter is separate from the listing, which is why both are assigned 1.

Finally, here is a Coq proof

```
1   Fixpoint nexec {S Q : Type}
2                  (qs : list Q) (d : S → Q → list Q) (s : list S) :=
3     match s with
4     | []       ⇒ qs
5     | s' :: sx ⇒ nexec (qs >>= (d s')) d sx
6     end.
7
8   Lemma nexec_distr :
9     forall {S Q} (qs1 qs2 : list Q)
10                 (d : S → Q → list Q) (s : list S),
11      (nexec qs1 d s) ++ (nexec qs2 d s) = (nexec (qs1 ++ qs2) d s).
12  Proof.
13    intros S Q qs1 qs2 d s.
14    generalize dependent qs2; generalize dependent qs1.
15    induction s as [| s sx].
16    - reflexivity.
17    - intros qs1 qs2; simpl. rewrite list_bind_distr. apply IHsx.
18  Qed.
```

Code 2: Definition of NFA execution using the list monad

7 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 7.

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* file *references.bib*. This is redefined to litteratur with the danish preamble, preamble_dk.tex. If you want more references simultaneously you just add another entry in the references file.

The file contains three references, but the *biber* tool resolves what references you actually use and only displays the ones that you cite. Here is a citation of another paper by my supervisor Jaco van de Pol [BDP19]

8 TODO notes

This is a small correction

When working on bigger projects or when collaborating it can be useful to be able to add todo notes for coordination, such as the one on the right. All TODO notes are hidden, when the documentclass at the very top is also given the *final* argument.

Sølvsten: This is a bigger todo, why it is in its own paragraph. It is possible to write math inside all of these, such as $\sum_i^n i = \frac{n(n+1)}{n}$. With these you can sketch out all major points you want to write up as part of a section or exercise.

It is possible to create a "list of todos" as below

List of TODO Notes

Finally if you are missing citations you can can do [??] or [unknown] .

References

[BDP19] Vincent Bloemen, Alexandre Duret-Lutz, and Jaco van de Pol. "Model checking with generalized Rabin and Fin-less automata". In: *International Journal on Software Tools for Technology Transfer* 21.3 (June 2019), pp. 307–324.

[HS20] Kristoffer Arnsfelt Hansen and Steffan Christ Sølvsten. "$\exists\mathbb{R}$-Completeness of Stationary Nash Equilibria in Perfect Information Stochastic Games". In: vol. 47. submitted. 2020.

[ZK17] Danfeng Zhang and Daniel Kifer. "LightDP: towards automating differential privacy proofs". In: *Proceedings of the 44th ACM SIGPLAN Symposium on Principles of Programming Languages*. POPL 2017. ACM Press, 2017.

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.