Intermediate-Level Issues in LaTeX2e

List of Figures

Defining commands

Section 3.4.1 of Lamport spends a lot of space on dealing with curly braces in macro definitions. Luckily, that is rarely an issue in practice.

I usually use `\newcommand` to define a macro. If it already exists, and you want to redefine it, use `\renewcommand`. If there are parameters, you specify how many by a square-bracket-enclosed argument. Then come a pair of curly braces with the body of the macro definition. The arguments are indicated by `#1`, `#2`, etc.

I won't say much else about defining macros, except that the ifthen package, documented by Lamport in Section C.8.5 is great on paper, but it just doesn't work for me. I end up using the raw TeX commands for conditional compilation. I'm not going to go into them here, but feel free to ask about them.

Defining environments and theorem-like environments

Sections 3.4.2 and 3.4.3 of Lamport are pretty straightforward.

One question that often comes up is whether to draw numbers for theorems, lemmas, corollaries from one big pool or instead number theorems separately from everything else, lemmas separately from everything else, etc. In other words, should a paper have

- Lemma 1, Lemma 2, Theorem 3, Corollary 4, Lemma 5, ..., or
- Lemma 1, Lemma 2, Theorem 1, Corollary 1, Lemma 3, ...?

Although you can make a decent argument for either way, I favor the one-big-pool approach (i.e., the first way above).

Do use labels for your theorem-like numbers. For example:

Theorem 1 *All numbers are interesting.*

Proof: We prove the theorem by contradiction. Suppose that there exists a number x that is not interesting. Without loss of generality, let x be the lowest such number. Then x is the smallest non-interesting number, which is in itself interesting. Hence, x is interesting, thus contradicting our supposition that x is not interesting. ■

Corollary 2 *No number is boring.*

Proof: Immediate from Theorem 1. ■

Hence, by Corollary 2, we cannot use any numbers as drill bits.

When the qed symbol appears at the end of a math display (including an eqnarray), the `\qed` macro does not quite work:

Proof: We have

$$\begin{aligned} T(n) &= 2T(n/2) + \Theta(n) \\ &= \Theta(n \lg n) \ . \end{aligned}$$

■

We'd really like the qed symbol to appear on the last line of the display. We can do that, with varying degrees of effort. The biggest change would require us to redefine all the math display environments. Rather than doing that, let's explicitly ask for a qed symbol in math display modes.

Now we get what we want:

Lemma 3 *The running time $T(n)$ is $\Theta(n \lg n)$.*

Proof: We have

$$\begin{aligned} T(n) &= 2T(n/2) + \Theta(n) \\ &= \Theta(n \lg n) \ . \qquad \blacksquare \end{aligned}$$

Corollary 4 *The running time $T(n)$ is polynomial in n.*

Proof: Since $T(n)$ is $\Theta(n \lg n)$, it is also $O(n^2)$. ■

With more macro hacking, we can eliminate the need for the negative vspace when a math display ends a proof. Again, that would require us to redefine all the math display environments.

Figures

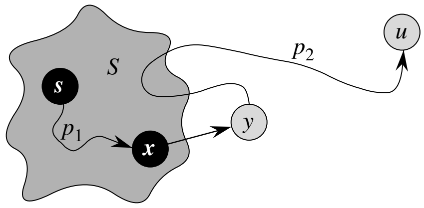
Of course you want to put figures in your papers! There are several aspects to including figures, some of which we'll cover later in the course.

Produce .eps (Encapsulated PostScript) files for your figures. Make sure that the bounding box is correct in each .eps file. You'll see the bounding box in a line like

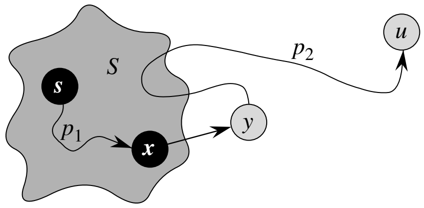
```
%%BoundingBox: 85 490 503 694
```

near the top of the .eps file. You can use gv to verify that you've got the right bounding box.

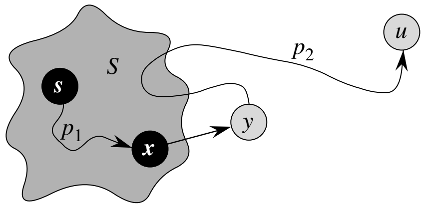
You then can include the figure via the `\includegraphics` command. Make sure you use the graphics package in your preamble!



You can also make your figure be flush left against the page margin:

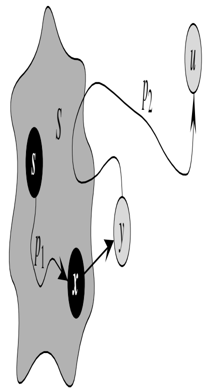


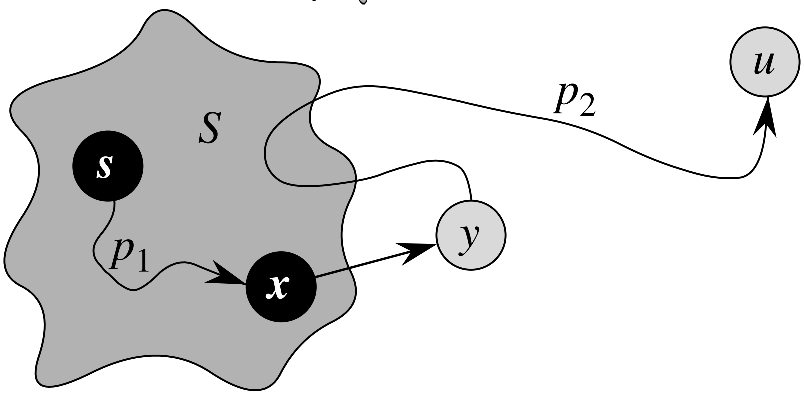
Or, if you're from another planet, flush right:

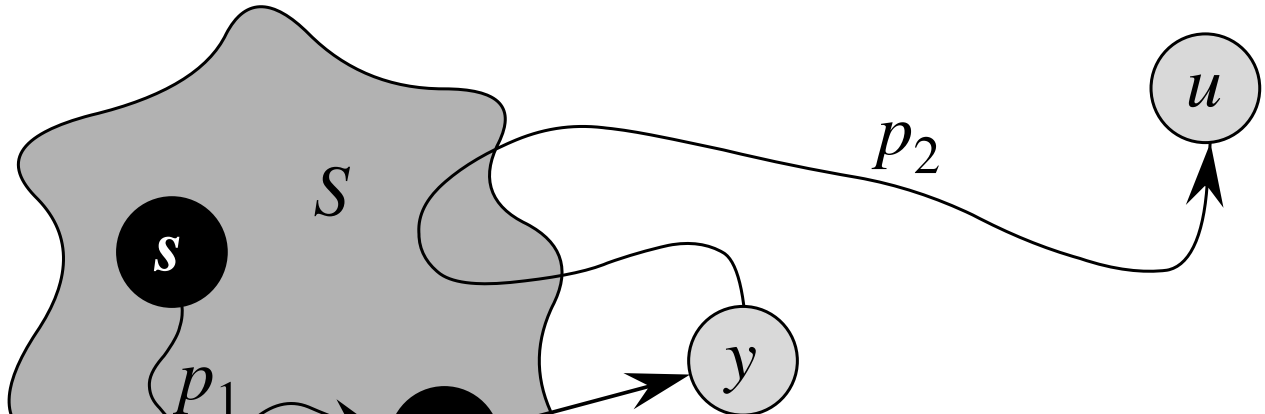


Getting the bounding box right in the .eps file is the key.

You can scale the figure easily by using the `\resizebox` command. Its first two arguments give the width and height of the figure. If you use `!` as one of the arguments, that dimension scales automatically to preserve the aspect ratio of the original figure. If you use `\textwidth` as your first argument, your figure will be exactly the width of the text page.







I almost always use the figure environment for figures in papers (and books). The figure environment numbers the figure, allows the figure to get a symbolic label, and it allows a caption. The label allows me to refer to Figure 1.

LaTeX2e has its own notions of where it will put figures. Pages 199–200 of Lamport list several parameters that LaTeX2e uses to decide whether it will put a figure at the top of a page or on a page of only figures. I do not particularly like the default values; they tend to favor pages of only figures. I have redefined several

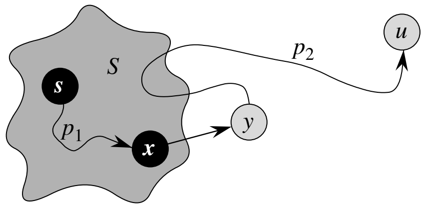


Figure 1: Why Dijkstra’s algorithm works.

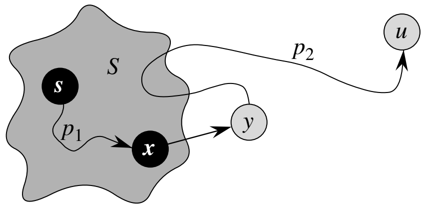


Figure 2: Why Dijkstra’s algorithm works and works.

of these values in ~thc/mytex/page.tex.

Nor am I particularly fond of how captions appear by default. I like captions to be in a smaller font than the main text. I like the text “Figure X” to be in boldface.

There’s another issue with the `\caption` macro. By default, it takes the argument and puts it in a list of figures, assuming that you ask for a list of figures. Who wants a list of figures? Not me! I’ve asked for one in this file, and I don’t think I ever have before. Nor do I expect to ever again. But here’s the problem. Suppose you have a loooong caption. Then by default you get a loooong entry is your list of figures. Here’s an example:

To solve all of these problems, I have my own `\figcaption` macro. It puts “Figure X” in bold, uses a small font, and gives an empty optional argument to `\caption` so that there’ll be no entry in the list of figures:

That looks better. I also have a `\figpart` macro for when a figure has multiple parts:

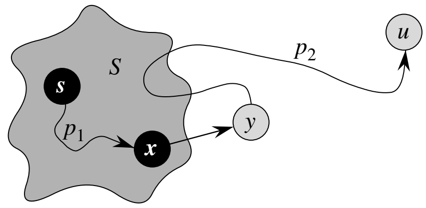
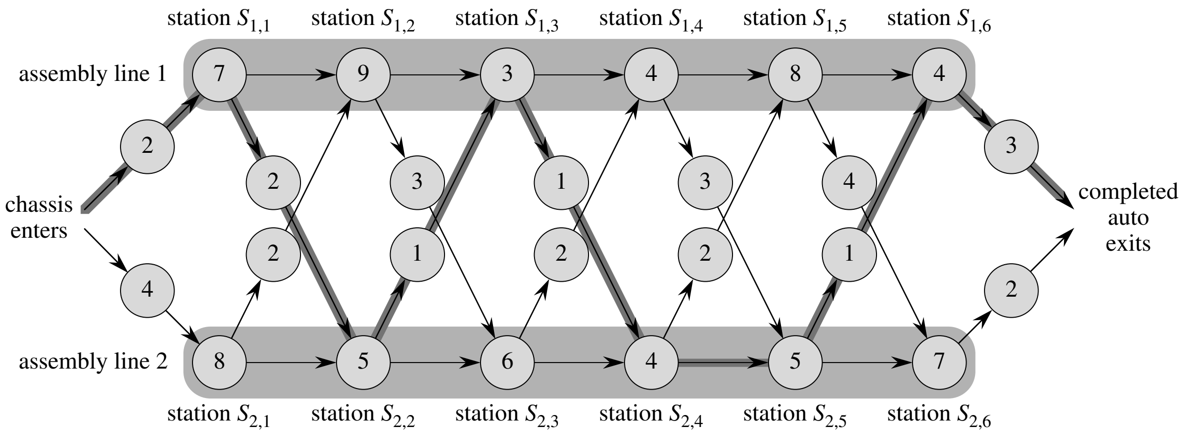


Figure 3: Why Dijkstra's algorithm works.



(a)

j	1	2	3	4	5	6
$f_1[j]$	9	18	20	24	32	35
$f_2[j]$	12	16	22	25	30	37

$f^* = 38$

j	2	3	4	5	6
$l_1[j]$	1	2	1	1	2
$l_2[j]$	1	2	1	2	2

$l^* = 1$

(b)

Figure 4: (a) An instance of the assembly-line problem with costs e_i, $a_{i,j}$, $t_{i,j}$, and x_i indicated. The heavily shaded path indicates the fastest way through the factory. **(b)** The values of $f_i[j]$, f^*, $l_i[j]$, and l^* for the instance in part (a).