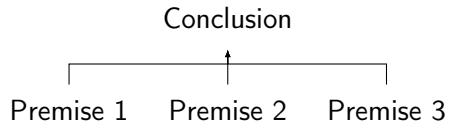
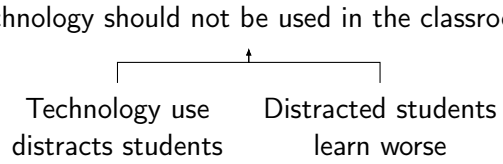
LRP Week 5-1: Informal Argument Mapping

The basic building block of an informal argument map is, like with logic machine systems, a **one-step argument**, that goes from some premises to a conclusion.

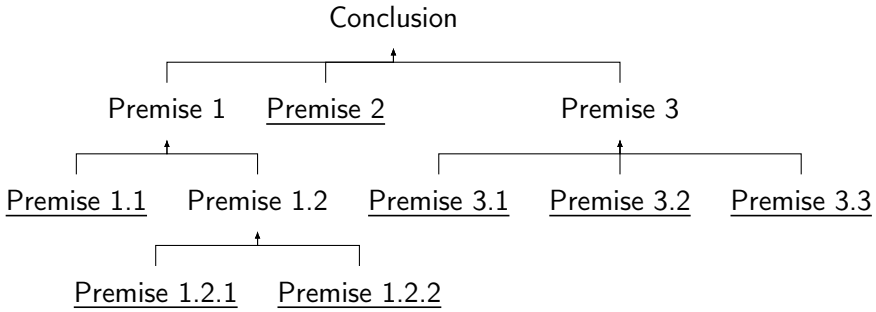
One-Step Argument



For example:

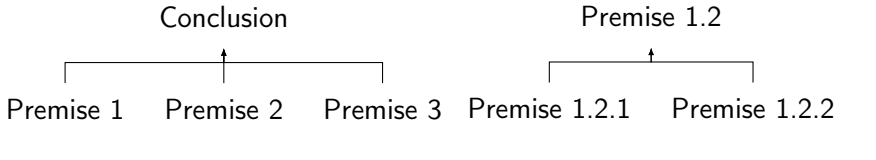


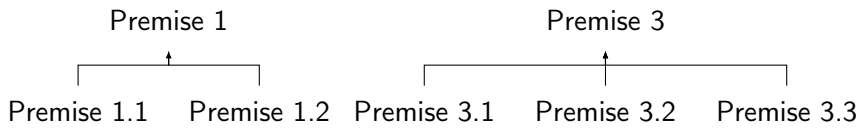
Each premise can itself be the conclusion of a one-step argument: that is in fact how we build multi-step arguments:



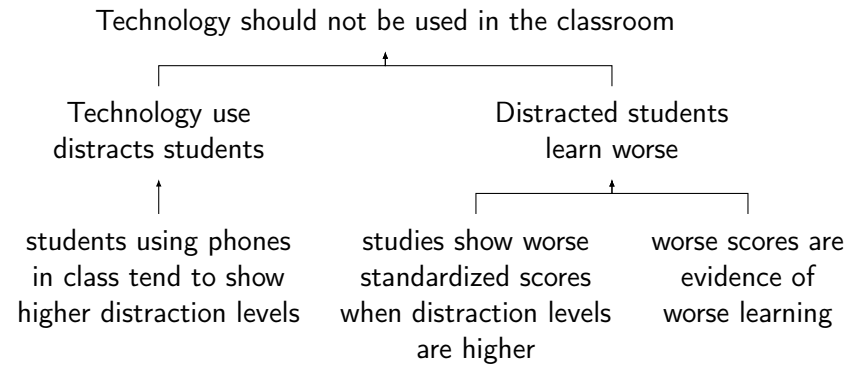
Here, the underlined premises are ones that aren't themselves the conclusions of arguments (i.e. left undefended).

If we split up this large multi-step argument into the single-step ones that compose it, we get four single-step arguments:

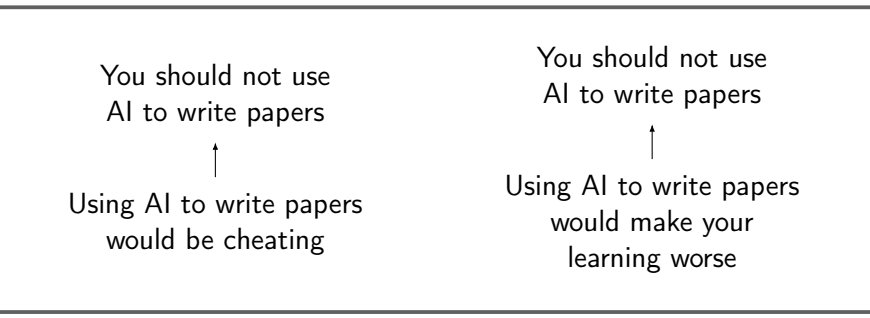




To make the technology argument into a multi-step argument, we ask if any of the premises could use further defense. For example:

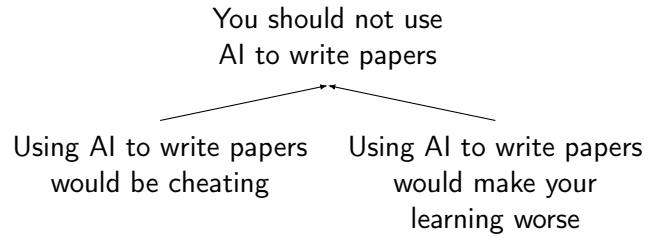


As we've been noting throughout the course, the same conclusion can be supported by multiple arguments:



In these cases, we can write the conclusion just once:

Same Conclusion via Multiple Arguments



Note that this is *not* the same as having a single two-premise argument. Below, on the left, we have a conclusion supported by one two-premise argument. On the right,

we have a conclusion begin supported by two different one-premise arguments.

1 | Support

Good arguments are not necessarily truth-preserving. The argument that technology use distracts students and therefore should not be used in the class may be a good argument even if there could be some scenarios in which the premise is true but the conclusion is false (for instance, if what is being learned requires technology). In these cases, what we ask of the premises is not that they *guarantee* the truth of the conclusion, but that they *support* the truth of the conclusion.

In the case of the technology argument, the fact that technology use in classrooms is *in general* distracting can give us reason *in general* to think we should not use technology in the classroom, even if it doesn't guarantee that we should never use technology in the classroom.

Support

In a good argument, the premises **support** the conclusion in that *if* the premises are true, we have good reason to think that the conclusion is true.[a]

a. **Note**: In a truth-preserving argument, the premises *guarantee* the conclusion. Thus, if the premises are true, we have very good reason to think the conclusion is true (since it *must* be true). So in a truth-preserving argument, the premises *support* the conclusion.

2 | Constructing Good Argument Maps

The greatest virtue of an argument is that the premises support the conclusion. When constructing argument maps, we should always be checking to see if the premises support the conclusion. But there are two additional rules that allow us to construct good argument maps that clarify the structure of complicated arguments.

Rules of Argument Mapping

1. **Simple Statements**: each premise should be a simple statement.
   (a) It should be a **statement**: something that can be true or false.
   (b) It should be as **simple** as possible.
2. **Mutual Dependence**: the premises in a one-step argument (or each step of a multi-step argument) should be mutually dependent. That is, they should able to support the conclusion *only* together with all of the other premises.

Let's discuss these rules in turn.[1]

2.1 | Each Premise Should Be a Simple Statement

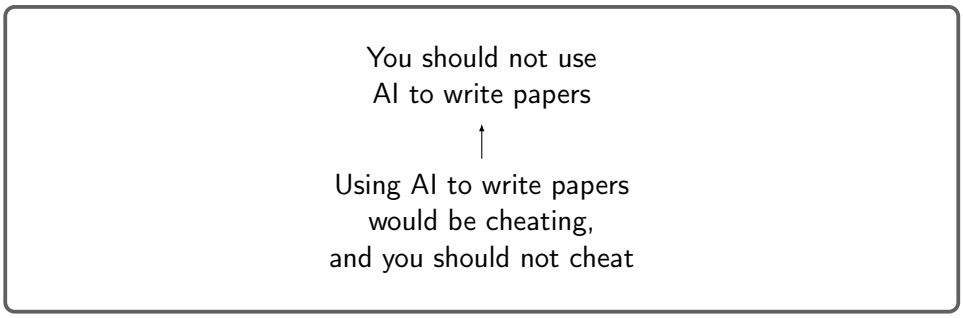
Recall that a **statement** is something you can write or say that can be true or false.[2] Premises need to be statements because we need things that can be true or false as the building blocks of our arguments. The rule of simple statements says not only that our premises must be statements, but also that they are *simple*.[3] What does **simple** mean? It's hard to be precise about this. Nonetheless, we can have some general rules of thumb:

Rules of Thumb for Simple Statements

In general, a simple statement

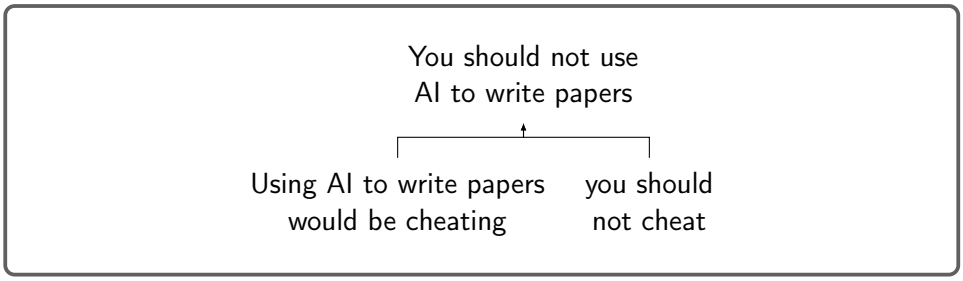
1. is as short as possible.
2. has no *linking* words, like "and", "furthermore," "but," and "however" (if there is a linking word, you can probably split the sentence into two different statements at that word, and put them at the same level).
3. has no *reasoning* words, like "because," "since," and "as" (if there is a reasoning word, you can probably split the sentence into two different statements, and put one of the statements at another level).

Let's see some examples of making argument maps follow these rules. Consider the following argument map:



This doesn't follow the first rule, because it has a statement that is not simple. The statement "Using AI to write papers would be cheating, and you should not cheat" has the linking word "and", thus combing two statements. If we split these out into two statements, we can see how we actually have *two* premises working together:

1. Note: argument mapping is *informal*, and different people do it different ways. It's hard to come up with a systematic algorithm to follow. I take inspiration from Shamik Dasgupta's "Brief Guide to Argument Mapping. I recommend taking a look at his method if you're interested. However, his method has eleven rules to remember, and I think only the three that I've formulated above are necessary. But it can be a fun exercise to compare my three rules to Dasgupta's eleven, and see what the differences are.
2. Remember also that we can use the statement test to check whether a sentence is a statement or not: If you can add "It is true that" to the beginning of a sentence, and it still makes sense (whether it is true or false), then the sentence is a statement. Otherwise, it is not a statement.
3. **Note**: this is different than when we did argument mapping with logic machines. Then, we allowed premises to be complex.

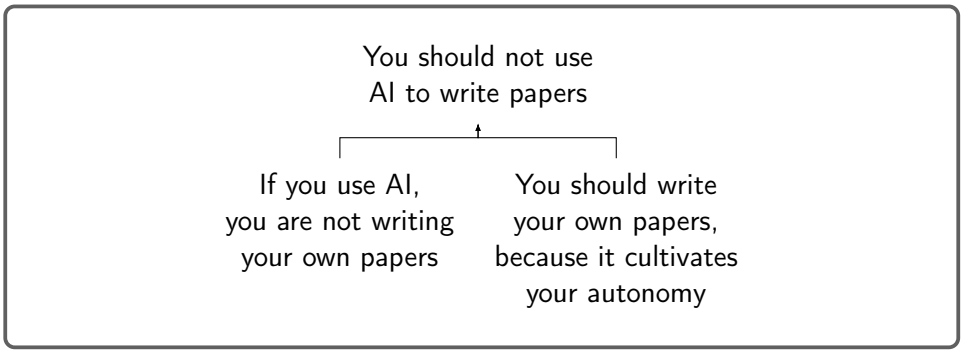


Notice also that once we separate out these premises, we can analyze (if we wish) this argument as an instance of the normative reverse implication machine!

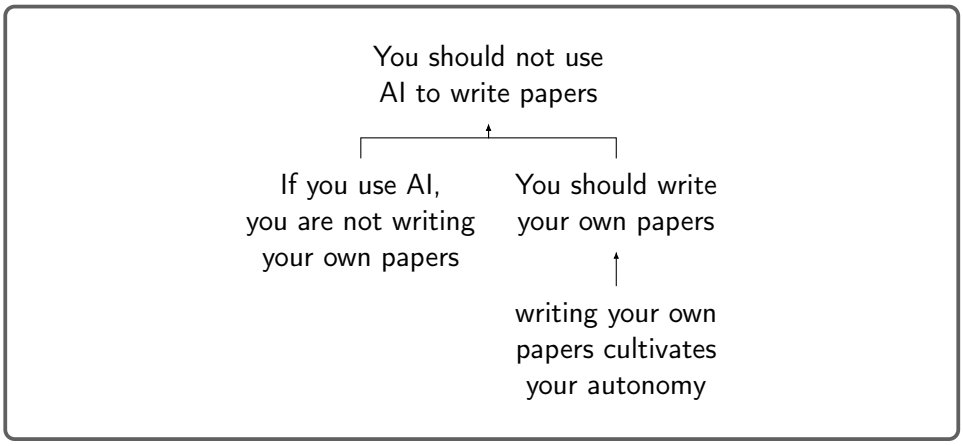
1. You should not cheat
2. **If** you use AI to write papers, **then** you are cheating.

→ You should not use AI to write papers.

Let's now consider an example of a *reasoning* word occuring in an argument:



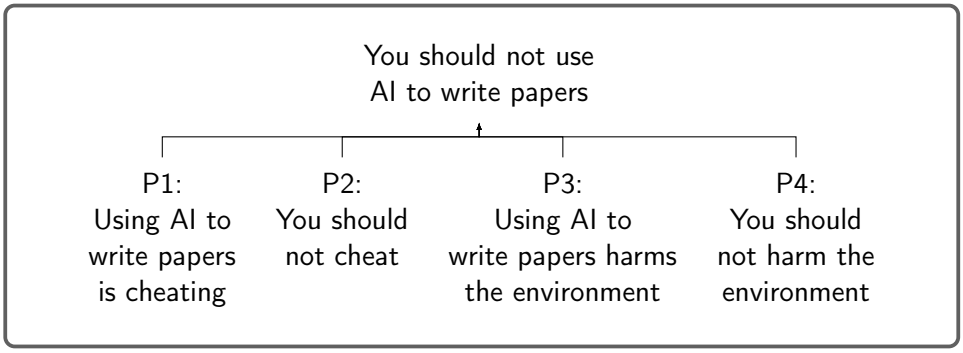
What's the problem with this? The second premise is not simple enough: it not only claims that you should write your own papers, but also tries to give further reasoning to defend the premise. So we should split it up into two statements: "You should write your own papers" and "it cultivates your autonomy." And then, because what "it" refers to might be unclear, we should replace it with what it refers to: "[Writing your own papers] cultivates your autonomy." Finally, since the second statement, "[Writing your own papers] cultivates your autonomy," serves to *defend* the statement "You should write your own papers", we should put it as a premise that defends that statement, one level down. In all, we have the following:



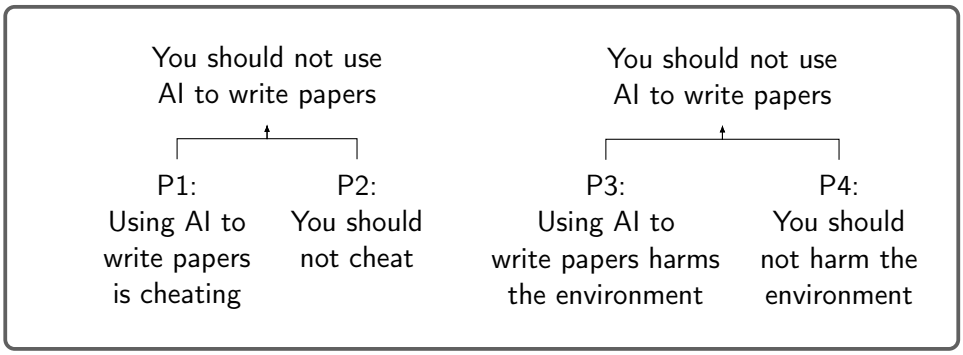
In general, when we have a linking word in a premise, then when we separate it out into two premises, the two premises stay at the same level. When we have a reasoning word in a premise, then when we separate it out into two premises, one of the premises goes one level down.

2.2 | Premises in a Step Should be Mutually Dependent

How do we check for mutual dependence? Suppose we have a set of premises in a step. For example:



Our strategy will be to see if we can group the premises into groups such that each group of premises gives a good argument for the conclusion without needing the other premises. Here, two of the premises, P1 and P2, are about cheating and work together (if we were doing this with logic machines, it would be in the form of the normative implication machine). The other two, P3 and P4, similar are about the environment and work together. Notice that neither of the pairs depend on the other pair in order to support the conclusion: the considerations about cheating don't need the considerations about the environment, and the considerations about the environment don't require the ones about cheating. So these two pairs of premises are **not** mutually dependent, and we can consider them separate arguments:



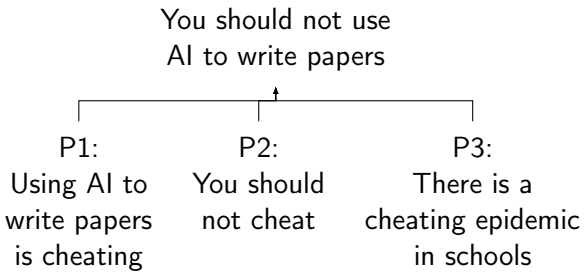
Considering each of the two arguments separately, we see that the pairs of premises **are** mutually dependent. P1, "Using AI to write papers is cheating" only supports the conclusion that you should not use AI to write papers if P2 is true: you should not cheat. Likewise, P2 only supports the conclusion together with P1: each needs the other. (The same applies to P3 and P4).

The "mutual dependence" rule can be hard to evaluate. Here are some rules of thumb:

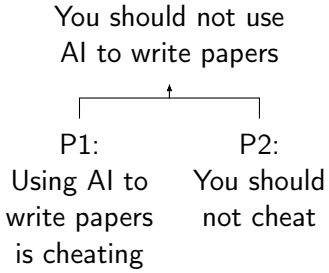
Rules of Thumb for Mutual Dependence

1. Mutually dependent statements tend to share some key words or a topic: after all, if they aren't using the same words, it's harder them to be on a related topic and thus dependent on each other.
2. If a premise seems unnecessary for an argument, and also seems to support the conclusion on its own, it probably can be moved to its own argument. (The same applies to groups of unnecessary premises.)
3. If a premise seems unnecessary for an argument, but it also doesn't support the conclusion on its own, it probably can be removed entirely. (The same applies to groups of unnecessary premises.)
4. If you end up splitting an argument into two different arguments, but both of the arguments require one of the same premises, you can make two copies of the same premise (one for each argument).

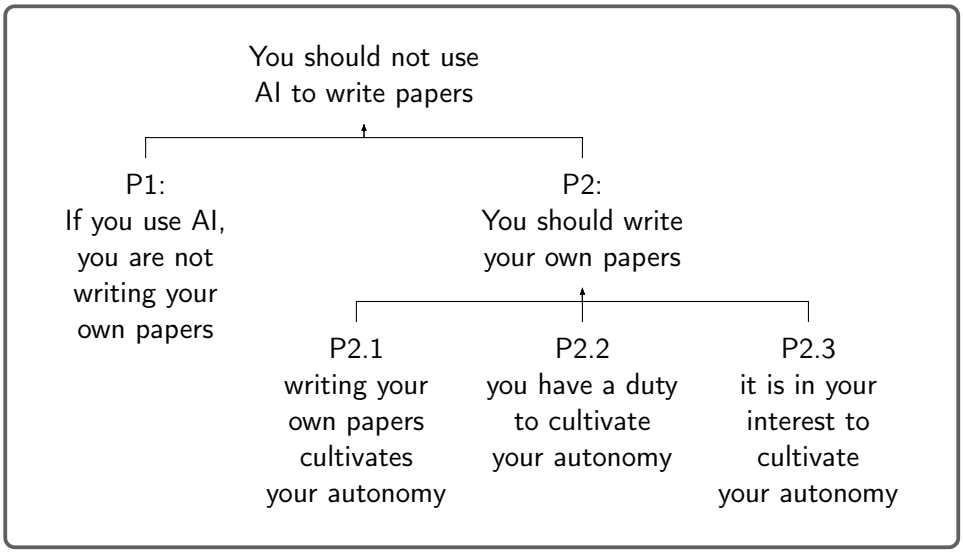
We saw above an example of the first and second rules of thumb. P1 and P2 shared the topic of cheating, and P3 and P4 shared the topic of harming the environment. And each of the pairs were unnecessary for the argument that the other was making. Below is an example of the third rule of thumb.



Here, P1 and P2 are enough to support the conclusion, and the fact that there is a cheating epidemic isn't really relevant. If you should not cheat, then you should not cheat, whether or not there is a cheating epidemic. So we should delete P3:



Now the fourth rule of thumb: making copies of premises. Consider the following argument:



P2.2 and P2.3 both support P2 together with P2.1. But they do so in different ways. So P2.2 and P2.3 should be separated into two different arguments. But which argument does P2.1 go with? We notice that both of the arguments needs P2.1. Therefore, we should copy P2.1 and write it twice, and have two separate arguments for P2 (here I'm going to hide P1 and the conclusion for space reasons).