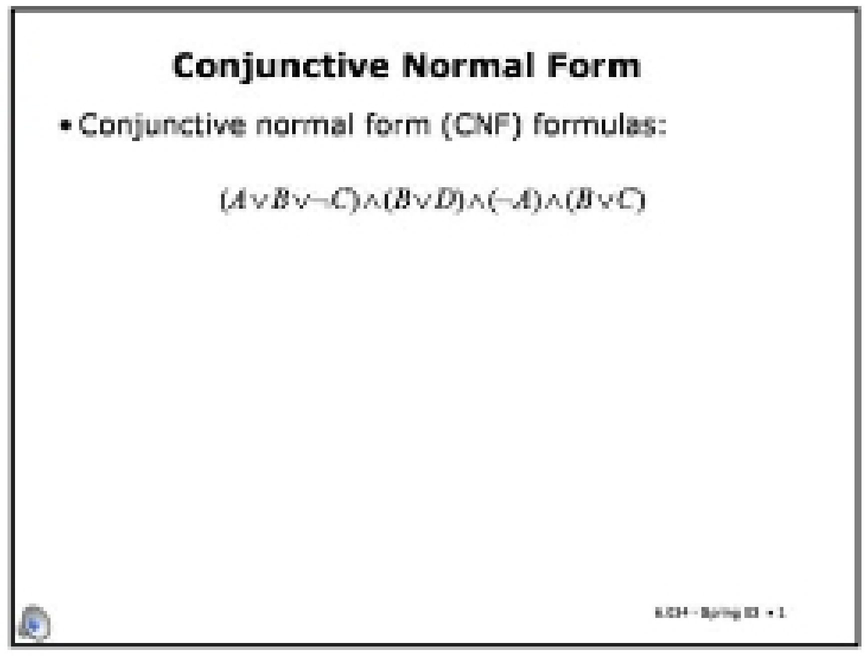
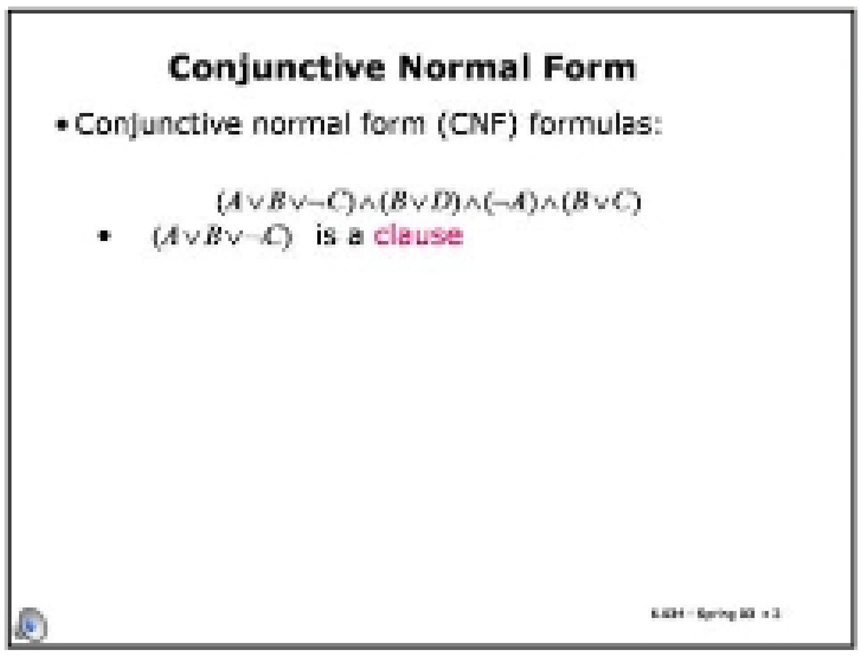
6.034 Notes: Section 10.1

Slide 10.1.1

A sentence written in conjunctive normal form looks like ((A or B or not C) and (B or D) and (not A) and (B or C)).



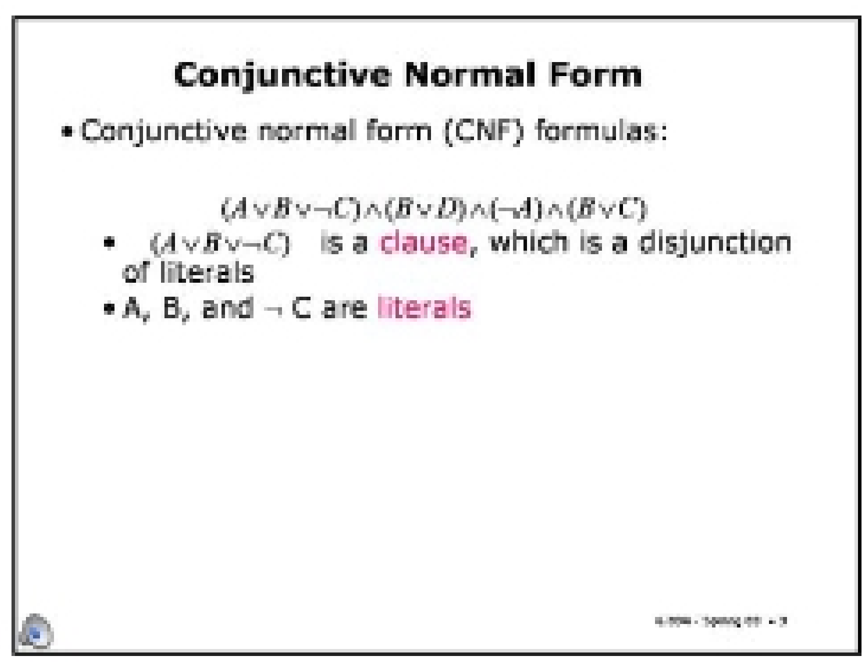


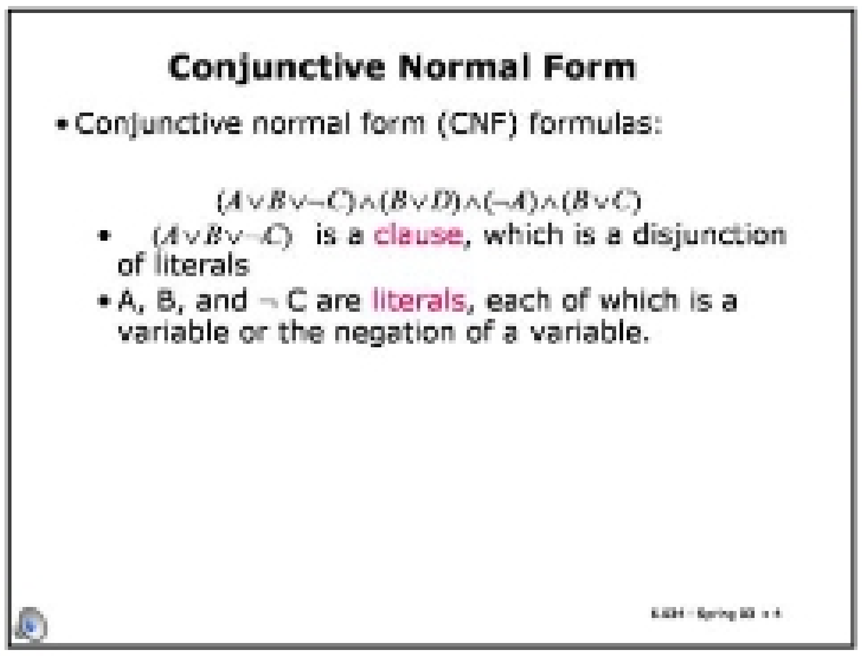
Slide 10.1.2

Its outermost structure is a conjunction. It's a conjunction of multiple units. These units are called "clauses."

Slide 10.1.3

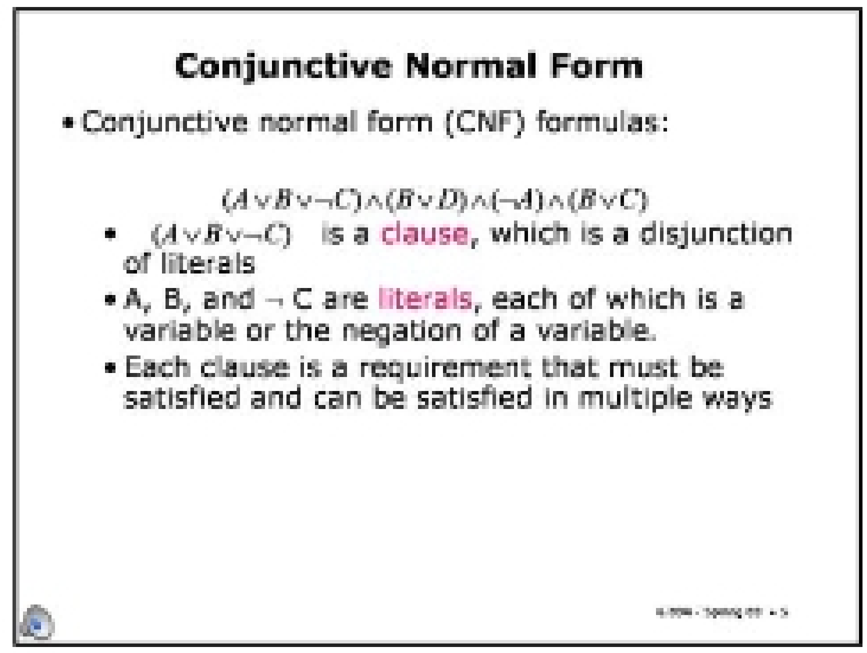
A clause is the disjunction of many things. The units that make up a clause are called literals.

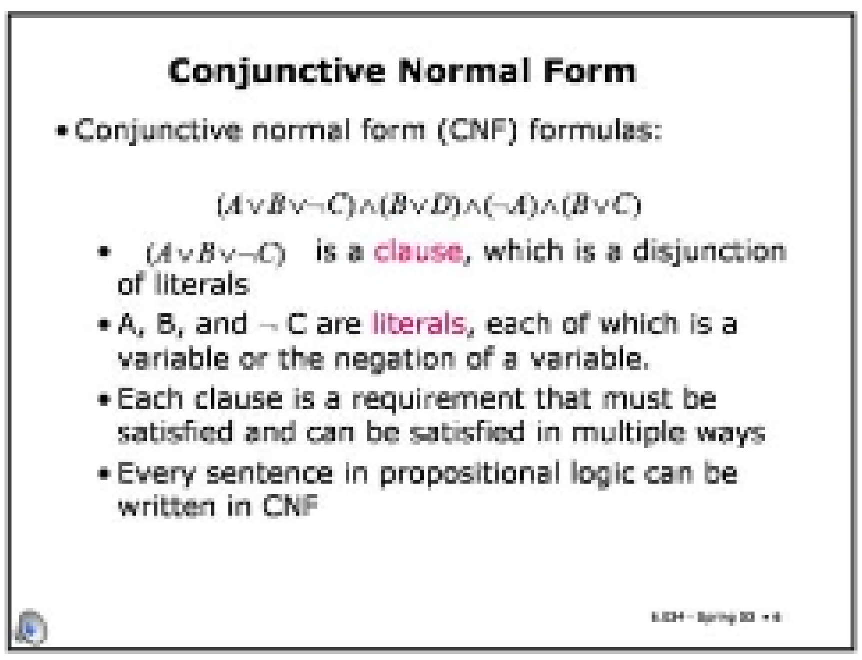




Slide 10.1.4
And a literal is either a variable or the negation of a variable.

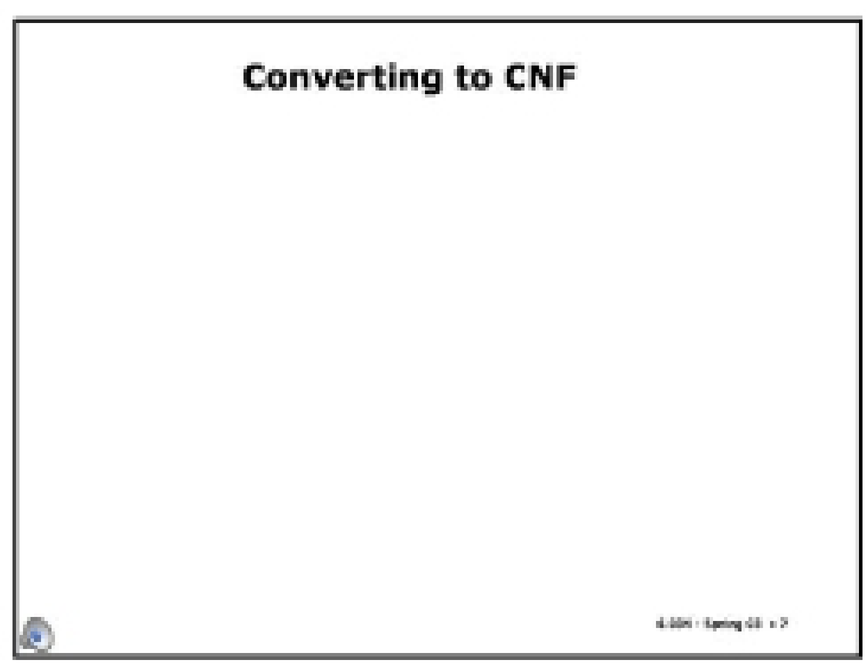
Slide 10.1.5
So you get an expression where the negations are pushed in as tightly as possible, then you have ors, then you have ands. This is like saying that every assignment has to meet each of a set of requirements. You can think of each clause as a requirement. So somehow, the first clause has to be satisfied, and it has different ways that it can be satisfied, and the second one has to be satisfied, and the third one has to be satisfied, and so on.

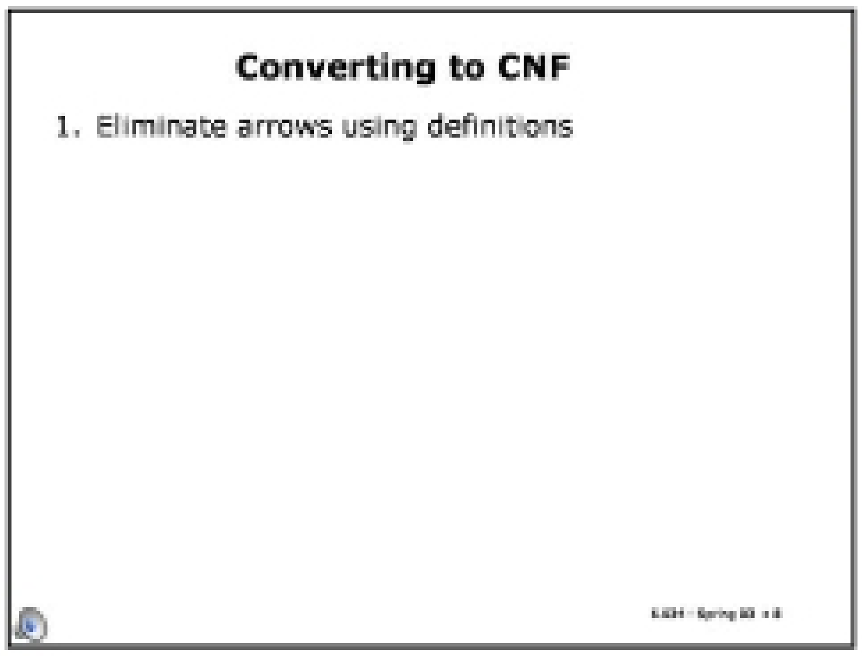




Slide 10.1.6
You can take any sentence in propositional logic and write it in conjunctive normal form.

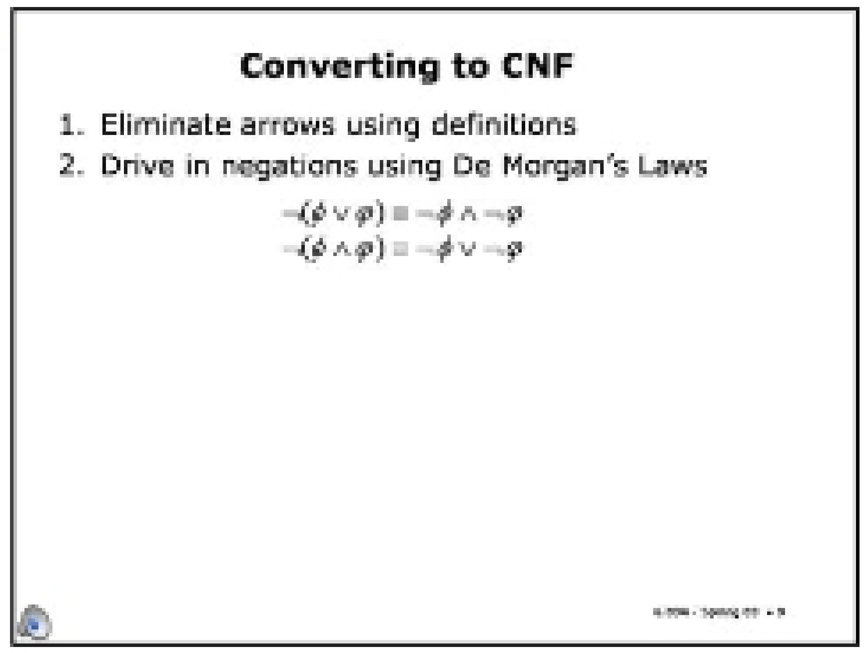
Slide 10.1.7
Here's the procedure for converting sentences to conjunctive normal form.

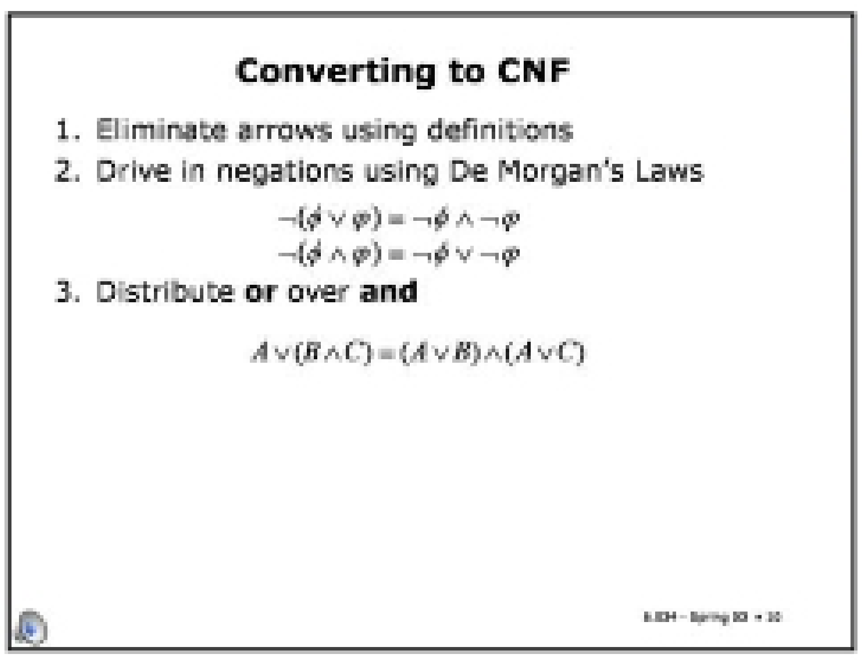




Slide 10.1.8
The first step is to eliminate single and double arrows using their definitions.

Slide 10.1.9
The next step is to drive in negation. We do it using DeMorgan's Laws. You might have seen them in a digital logic class. Not (phi or psi) is equivalent to (not phi and not psi). And, Not (phi and psi) is equivalent to (not phi or not psi). So if you have a negation on the outside, you can push it in and change the connective from and to or, or from or to and.





Slide 10.1.10
The third step is to distribute or over and. That is, if we have (A or (B and C)) we can rewrite it as (A or B) and (A or C). You can prove to yourself, using the method of truth tables, that the distribution rule (and DeMorgan's laws) are valid.

Slide 10.1.11
One problem with conjunctive normal form is that, although you can convert any sentence to conjunctive normal form, you might do it at the price of an exponential increase in the size of the expression. Because if you have A and B and C OR D and E and F, you end up making the cross-product of all of those things.

For now, we'll think about satisfiability problems, which are generally fairly efficiently converted into CNF.