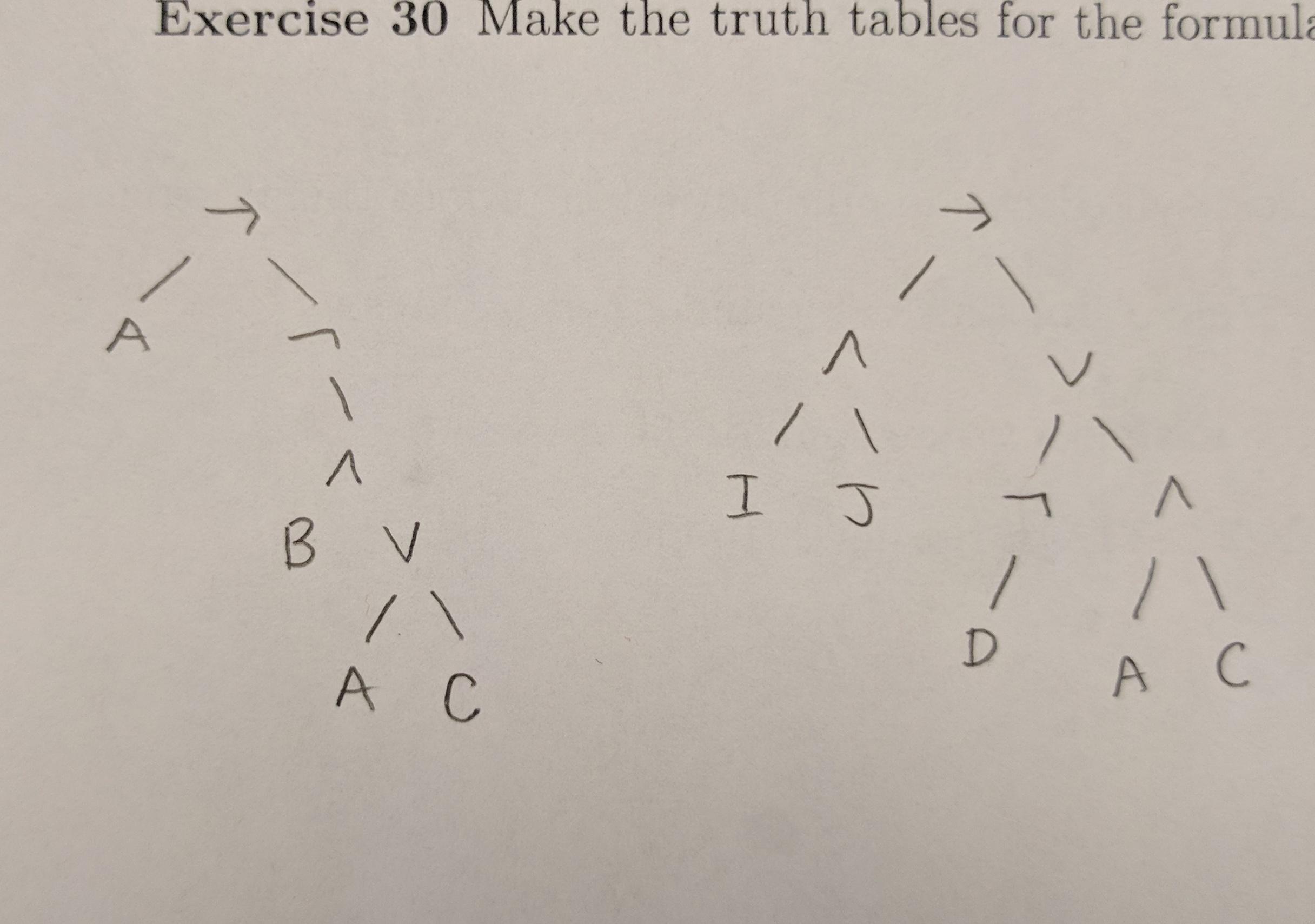
¬ ∧ ∨ →

Notes and Information for Test 2

Subformulas

* break formula down to smallest entity
* Ex: subformula of A→¬(B∧(A∨C)) is: A→¬(B∧(A∨C)), A, ¬(B∧(A∨C)), B∧(A∨C), B, A∨C, C
* its tree is:



Truth values

* To find truth values, can either make a truth table or do the following:

Let A(A) = A(B) = 0 and A(C) = 1

*A*(¬((A ∧ B) ∨ C)) = { 1 if *A*((A ∧ B) ∨ C) = 0 }

{ 0 otherwise }

= { 1 if *A*(A ∧ B) = 0 and *A*(C) = 0 }

{ 0 otherwise }

= { 1 if [*A*(A) = 0 or *A*(B) = 0] and *A*(C) = 0 }

{ 0 otherwise }

= 0

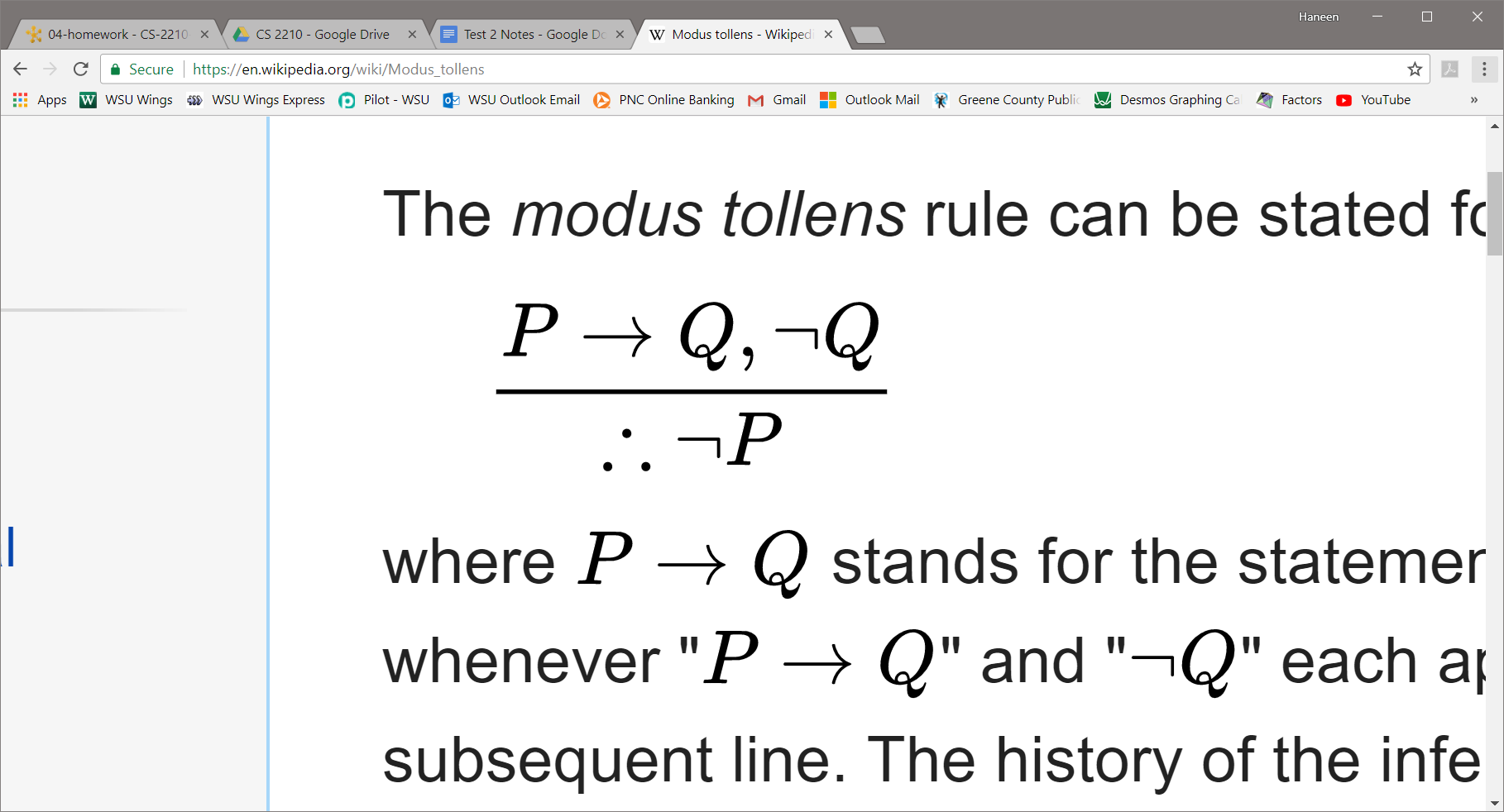
Model

* To find a model, find what makes the formula equal 1.
* Ex: a model for ((a -> ¬b) ∧ c) ∨ a is A(a)=1, A(b) = A(c) = 0

Satisfiable vs unsatisfiable:

* Make truth table and if all the values in the result are 0’s, then it’s unsatisfiable. Else, it is satisfiable.

Converting to propositional logic

* Example:  converts to ((P -> Q) ∧ ¬Q) ⊨ ¬P in propositional logic.
* To prove they are valid, show that the result column is all 1’s (true).

Determining v(P) of the Datalog program P

* example: r(a, b)

r(x, y) -> s(x, y)

s(x, y) ∧ s(y, z) -> s(x, z)

ra,b

{rx,y -> sx,y| x, y ϵ {a, b, c}}

{sx,y ∧ sy,z -> sx,z | x, y, z ϵ {a, b, c}}

Translating to formulas

* Example:

“if I'm guilty, I must be punished."

“I'm not guilty"

“Thus, I must not be punished."

* Answer:

guilty -> punished

¬guilty

¬punished

* Example:

“if Carlo won the competition, the either Mario came second or Sergio came third."

“Sergio did not come third."

“Thus, if Mario did not come second, then Carlo did not win the competition."

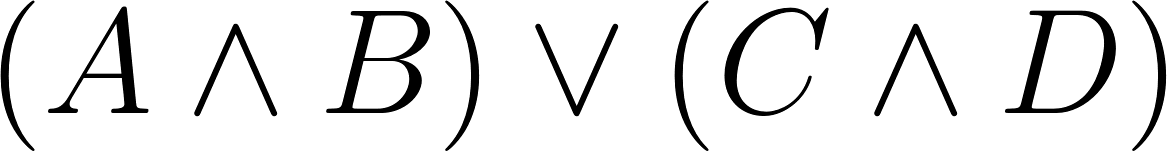
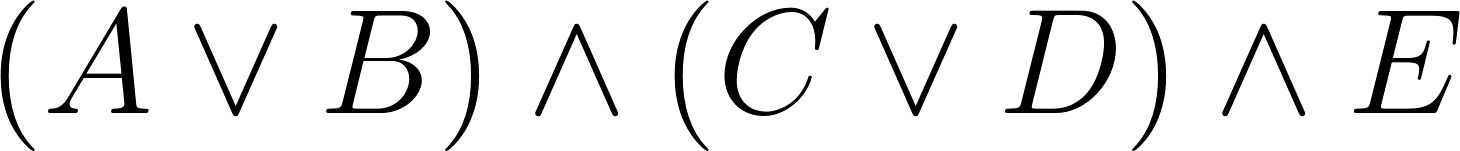
* Answer:

woncarlo -> secondmario ∨ thirdsergio

¬thirdsergio

¬secondmario -> ¬woncarlo

Converting formulas into CNF and DNF

* DNF
  + You need to shape the formula to have the or (∨) as the top level and the and (∧) inside. Example: [](https://www.codecogs.com/eqnedit.php?latex=%20(%20A%20%5Cwedge%20B%20)%20%5Cvee%20(%20C%20%5Cwedge%20D%20)%20%0)
* CNF
  + You need to shape the formula to have the and (∧) as the top level and the or (∨) inside. Example: [](https://www.codecogs.com/eqnedit.php?latex=%20(%20A%20%5Cvee%20B%20)%20%5Cwedge%20(%20C%20%5Cvee%20D%20)%20%5Cwedge%20E%20%0)

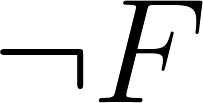
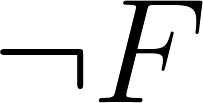
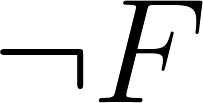
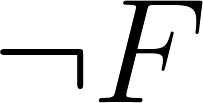
Dealing with Disjunctions and Conjunctions

* For creating the tableau of a given formula
  + If you are dealing with a conjunction (or) you branch off and split
  + If you are dealing with a disjunction (and) you branch straight down

Satisfiable vs Unsatisfiable

* Satisfiable if there are is at least one branch with no contradictions.
* Unsatisfiable otherwise.

Is it a tautology?

* Running a tableau on F
  + If the tableau closes (all branches contain contradictions)
    - F is unsatisfiable
    - F is not a tautology (not valid)
  + Else the tableau does not close (at least one branch is not contradictory)
    - F is satisfiable
    - Check tableau on [](https://www.codecogs.com/eqnedit.php?latex=%20%5Cneg%20F%20%0) to find if it is a tautology (i.e. valid)
* Tableau on [](https://www.codecogs.com/eqnedit.php?latex=%20%5Cneg%20F%20%0)
  + If the tableau closes (all branches contain contradictions)
    - [](https://www.codecogs.com/eqnedit.php?latex=%20%5Cneg%20F%20%0) is unsatisfiable
    - F is a tautology (valid)
  + Else the tableau does not close
    - [](https://www.codecogs.com/eqnedit.php?latex=%20%5Cneg%20F%20%0) is satisfiable
    - F is not a tautology (not valid)

Representing the “entails” operator (⊧) in a tableau:

* Proving that {F1,...,Fn} ⊧ G is equivalent to proving that 
  + This can be done using a tableaux
* Class example for exam review
  + (A /\ B), (not A \/ not B) ⊧ (A /\ C)