172 Notes – Motivation, and Logic chapters

Introduce Briggs:

3 classrooms in basement – not operational yet, new locations for 141, 142

2 classrooms on main floor – are working now

1 auditorium on main floor – new room for big talks

upstairs – all prof offices, hardware lab for 231, student research lab, breakout, relaxing, work space for students

Why 172?

* Show venn diagram – CS is intersection of three things, science, engineering, and math.
* Engineering – building and inventing things (algorithms, data structures, hardware)
* Science – studying the things we build (testing our algs, DS, hardware to see how they work in various situations, which are better)
* Math – the tools we need to use to help out with the science or the engineering, prove how much time or space algs use

Math behind CS.

* Computer scientists need a universal language to talk about things. English is not universal, even programming languages like Python/C++ are not universal because not everyone knows them. Pseudocode is not universal.
* Math is universal.
  + Two basic components of this course that we will learn.
  + Mathematical formalisms that enable us as computer scientists to communicate ideas to one another while eliminating all ambiguity about what we mean.
    - Consider in 141 where you learned how to take a list of integers and find the largest number in that list. You can do this in lots of different ways – front to back, back to front, by initializing biggest so far to 100 or 10. Lots of variations.
    - But the mathematical way to describe the largest element in a list is very straightforward, consise, and there’s pretty much only one way to do it.
    - (exists x in L) (forall y in L) x >= y
  + Learning how to prove things about those mathematical formalisms.
* Basic idea is learning to prove things.
* Why do we want to learn to prove things?
* In CS, we often create and study complex things (algs, data structures, ways of organizing information, or things that have rather unpredictable properties like the internet)
* We can use our intuition to deduce how these things work, but sometimes our intuition is just plain wrong or it conflicts with someone else’s intuition.
* So we can use logic to prove things unambiguously, to convince ourselves and other people about the way something works.
* Things people often care about proving:
  + Prove an algorithm actually does what we claim it does.
  + Prove an algorithm takes a certain amount of time or memory to run.
  + Prove that an algorithm works efficiently all the time or in certain situations.
  + Prove a data structure allows us to access information efficiently.
  + One of the most famous unsolved problems in computer science is a proof-type problem.
    - P vs NP problem. Asks (informally): If the answer to a computational problem is easy to CHECK, is the problem also easy to SOLVE?
    - Use example with TSP or SUBSET-SUM.

Admin stuff – pass out syllabus

* Textbook – either is fine, keep up with reading (more than 141), do reading before class.
* Important things – homework, in class quizzes, 2 midterms (evenings)
* Class conduct – Main idea is RESPECT each other, what that means is:
  + Being on time
  + Raise your hand to ask a question
  + Don’t be on your cell phone or computer in class.
  + Ask to go to the bathroom (I’ll always say yes)
* Introductory game – One book you either loved or hated in high school or college english class?

START PROPOSITIONAL LOGIC [HW read syllabus, read articles]

2.1

* Propositional logic is a tool that lets formally prove that something is true.
* It is a formal mathematical system that has symbols and rules, just like algebra.
* In algebra, the smallest item we usually care about is a variable or a number, and we can combine variables and numbers using operations like plus, times, exponents, and so on, into larger expressions.
* The smallest unit we use in propositional logic is the proposition, or statement.
  + A proposition is a sentence that is true or false but not both.
  + Example: “It is sunny outside.” FALSE
  + Example: “Prof Kirlin is wearing a checkered shirt.” TRUE
  + Example: “2 + 2 = 5” FALSE
  + Example: “2 + x = 5” AMBIGUOUS – not a statement
  + Example: “She is wearing black shoes.” AMBIG – not a statement
  + Example: “This sentence is false” –PARADOX, not a statement
* We can use variables to stand for statements.
  + Example: “CS 172 is a fun class.” Let that be p.
  + Let q be “CS 172 meets at 1pm” Let that be q.
  + Let r be “CS 172 is a required class”
* We can combine statements into compound statements.
  + There are three symbols from which we can build compound statements.
  + AND ^
  + OR v
  + NOT ~
  + Order of operations: ~ has higher prec than ^ or v, which have equal precedence.
    - P ^ q v r is ambiguous
  + Can also use parentheses to override.
* Do some examples.
  + p ^ q
  + p v q
  + ~p
* So we know how to determine if a single proposition is true, but how do we determine if a larger proposition is true? Like is p ^ q true?
* We know what these symbols mean intuitively, but we have to define them formally.
* Truth tables show exactly how these symbols work, and let us assign truth values to compound statements.

START HERE AUG 25  
  
Show all 3 truth tables

**TT table for (p and q) or not r**

* Show truth tables for all 3.
  + Example with **or being exclusive**.
  + You may order asparagus or spinach.
  + Make truth table
* Translating the word “but”
  + “I like apples but not bananas.” = “I like apples and I don’t like bananas”
* Translating “neither/nor”
  + I like neither apples nor bananas = I do not like apples and I don’t like bananas.
* Logic in algebra:
  + X <= y means “x is less than y or x is = y”
  + X < y < z” means x is less than y and y is less than z.”
* Translation practice  
    
  **EX**
  + **K = I like Coke.**
  + **P = I like Pepsi.**
  + **S = I like Sprite.**
  + **I like Coke and Pepsi but not Sprite. K and p and not s**
  + **I like Sprite and at least one of the other two. S and (p or s)**
  + **I don’t like any of Coke, Pepsi, or Sprite. Not p and not q and not r**
* More terminology
  + AND = conjunction
  + OR = disjunction
* OR in logic is inclusive or (review truth table)
  + What would truth table look like for exclusive or?
  + How would you write exclusive or in terms of and/or/not
* **TT table for (p and q) or not r**
* **Logical equivalence**
  + If two statements have the same TT, they are logically equiv
  + Use triple equals
  + E.g., p and q === q and p
  + To prove that two statements are equivalent, draw a truth table for each and make sure their corresponding columns match 100%
  + To prove that two statements are not equivalent, draw a truth table for each and find at least one entry in the two columns that doesn’t match.
* DeMorgan’s laws = show equiv in TT
  + Write negation of: It is raining today and I forgot my umbrella
  + Write negation of: if (x < 3) and (y >= 2)
* Terminology
  + Tautology = statement that is always true
  + Contradition = statement that is always false.
  + Lowercase t and c in statements.
    - P and t === p
    - P and c === c
    - What is p or t
    - What is p or c
* Show table of logical equivalences.

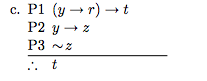
2.2 Conditional statements

* Another common type of proposition is an implication or conditional statement.
  + If I eat all my vegetables, then I get dessert.
  + We use the arrow symbol.
  + P = I eat all my vegetables.
  + Q = I get dessert.
  + P -> q
  + OK to switch tenses of English around to make it grammatically correct.
* Show TT for implies.
  + P is called the hypothesis, and q is called the conclusion.
  + p q p->q
  + T T T
  + T F F
  + F T T
  + F F T
* Reason is because the logic behind p -> q means if p is true, q must be true. We don’t know what to do if q is false, so we define the overall statement to be true.
* If 1+1=17, then I ate catfood for lunch.
  + TRUE OR FALSE?
* Order of operations --- always use parentheses with implies
  + P or not q implies not p, [[[ this is ambiguous ]]]
* Can rewrite p -> q as not p or q.
* Way to remember this is to think of it as something a parent says to you.
  + Either you get home by ten oclock, or you’re grounded.
  + P = you do not get home by 10
  + Not p = you get home by 10
  + Q = grounded
  + Therefore the sentence is not p or q.
  + Equal to p -> q.
* Another example:
  + Either you get to work on time or you are fired.
  + Let ~p be “you get to work on time”
  + Let q be “you are fired”
  + Then the given statement is ~p or q.
* Negation of conditional.
  + Not(p -> q) = p and not q.
  + Negation of a conditional is NOT a conditional, it’s an “and” statement.
* Contrapositive.
  + P -> q contrapositive is “not q implies not p”
  + Contrapositive is always equivalent to the original statement.
  + “If you sleep through your alarm, you will be late for class.”
  + Contrapos = “If you are not late for class, you didn’t sleep through your alarm”
* Converse
  + q -> p
* Inverse
  + ~p -> ~q.
* Just because p implies q is true does not mean that the converse or inverse are necessarily true. (they might be, but not guaranteed).
  + A conditional statement and its converse are not logically equivalent.
  + A conditional statement and its inverse are not logically equivalent.
  + The converse and inverse of a conditional statement ARE logically equivalent to each other.
* IF in the middle of a sentence; ONLY IF (formal logic/math definitions; these mean specific things)
  + So far we are used to having conditional statements phrased as IF-THEN statements, with the IF at the beginning of the sentence and THEN in the middle (sometimes we omit the word then).
    - Such as “If I sleep through my alarm [then] I will miss the exam.”
  + But the word “if” can also appear in the middle of the sentence, and you have to be a little careful with the meaning in that situation.
  + The first kind of “if” you’ll see in the middle of the sentence is something of the form “*p* if *q*.”
    - For example, “I will miss the exam if I sleep through my alarm.”
    - This is logically equivalent to the original statement.
    - In other words, “p if q” === “if q, then p” === q 🡪 p
  + However, it is common to see the phrase “only if” in the middle of the sentence, which formally means something very different.
    - For example, “You will pass the course ONLY IF you turn in the final report.”
    - This DOES NOT mean that IF you turn in the final report, you pass the course.”
      * Because there are other requirements to pass the course as well.
    - This means, rather, that IF you DON’T turn in the final report, you will not pass.”
    - In other words, “p only if q” === “if ~q, then ~p” === “~q 🡪 ~p” === “p 🡪 q”
  + Adding the “only” changes the meaning so that the only thing that can cause you to miss the exam is if you sleep through the alarm.
  + I wear a hat if it’s sunny. VERSUS. I wear a hat only if it’s sunny.
* IF and only IF
  + Sometimes we will encounter statements that use if and only if. I wear a hat if and only if it’s sunny.
  + A number is even IF AND ONLY IF it is divisible by 2.
  + Means both the “if” case and the “only if” case are true.
  + Called biconditional.
  + New symbol p <-> q
  + P <-> q === (p -> q) and (q -> p)
  + Show TT
  + iff
* Practice with converse, contrapositive, inverse
  + Easiest to do when translated into if then statement.
  + Statement = If it is snowing, then Rhodes is closed.
  + Contrapositive = If Rhodes is not closed, then it is not snowing.
  + Converse = If Rhodes is closed, then it is snowing.
  + Inverse = If it is not snowing, then Rhodes is not closed.
  + **Practice: If I don’t study for the test, I will get a bad grade.**
* Necessary and sufficient
  + P is necessary for q Ex: “Being 35 years old is necessary to be president”
  + If q happened, p must have happened.
  + Q -> p
  + P is sufficient for q Ex: A number being divisible by 4 is sufficient for the # to be even.”
  + Q will happen if p happens
  + P ->q
  + Necessary and sufficient == IFF

DAY 3 – start with biconditional truth table.

P <-> q === (p -> q) and (q -> p)

2.3 Valid and invalid arguments

* Motivation: There are many times when we want to prove something or draw some conclusion about an algorithm or data structure where the statement we want to prove boils down to a proposition.
* “Algorithm A runs in less than 30 seconds.”
* “If I switch my program to use this data structure, my program twice as fast.”
* “If I have a list with an odd numbers of items, the algorithm works correctly, but if the list has an even number of items, the output is wrong.”
* Building blocks of proofs are called arguments. An argument is a set of statements which we are given at the beginning of the proof to be true (called premises) and a single statement at the end (called the conclusion).
* We write arguments like this.
  + We write these argument forms like this
  + Premise
  + Premise
  + Premise
  + ------------
  + [therefore dots] conclusion
* An argument is called “valid” if no matter what T/F values are substituted for the variables in the premises, if the premises are true, then the conclusion is also true.
* If there are T/F values that we can put in that make the premises true but the conclusion false, then the argument is invalid.
* Ex: premises are [P1] P->Q plus [P2] (R v ~Q). Conclusion is p 🡪 r.
  + Draw truth table, show critical rows, conclude it’s a valid argument.
  + Show critical rows.
* Ex: Same premises. Conclusion is r -> p.
  + Same critical rows, but shows it’s an invalid argument.
* Put up process of testing an arg for validity:
  + Identify the premises and theh concusion.
  + Construct TT showing truth values of all premises and the concusion.
  + Identify critical rows. (rows with all premises true)
    - If all critical rows also have a true conclusion, valid arg.
    - If there is at least one crit row with a false conclusion, invalid arg.
* Now the problem is that figuring out whether an argument is valid/invalid through truth tables is a tedious process, especially when the number of variables starts going up. So we have what are called rules of inference that allow us to immediately know from some combination of premises, what kinds of conclusions what might be true, and which ones might be false.
* We’re going to look over the rules of inference.
* We will eventually use these rules to
* Modus ponens
  + P -> q, p THEN q
  + If I type a negative number into my program, the program crashes  
    I type a negative number in
  + ----
  + The program crashes
* Modus tollens
  + P -> q, not q, THEN not p
* If time, converse/inverse errors.
* Disjunctive addition
  + My algorithm runs in less than 30 seconds.
  + …or it is raining outside.
  + Can add true or false statement.
* Conjunctive addition
  + Ex: I own a dog. I own a cat. Therefore, I own a dog and a cat.
* Conjunctive simplification
  + Ex: I own a dog and a cat. Therefore, I own a dog.
* *No disjunctive simplification!*
  + *Fallacy: I own a dog or a cat. Therefore, I own a dog.*
* Instead, disjunctive syllogism.
  + Syllogism – another word for an argument
  + I own a dog or a cat. I don’t own a cat. Therefore, I own a dog.
* Hypothetical syllogism
  + Ex: If it snows more than an inch, Rhodes will close.  
    If Rhodes closes, then I will sleep until noon.  
    Conclusion: If it snows more than an inch, I will sleep until noon.
* Division into cases or dilemma
  + Ex: Alice is in either 141 or 142.
  + If Alice is in 141, then she has a MWF class.
  + If Alice is in 142, then she has a MWF class.
  + Conclusion: Alice has a MWF class.
* Contradiction rule.
  + Not used directly that much, but later we’ll see how to use this rule indirectly.
  + Ex: If Bob likes hot dogs, then 1+1 = 3. Conclusion: Bob does not like hot dogs.
* Sometimes hear these called rules of inference.
* Together with the logical equivalences, these are all the rules you need to do any proof.
  + Logical equivalences work both ways.
  + Rules of inference work one way only.
* Both rules of logical equivalence and rules of inference work when the variables stand for compound propositions.
  + Example: P1: p AND q. P2: (p AND q) -> r. Conclusion: r.
* Introduce proof method (table, line #s)
  + Premises = p, p->q, ~q or r, concl=r
    - Steps are easy: MP to get q, DS to get r.
  + Premises=p->r, q->r, conc=(p or q) -> r
    - Steps: def of impl separately, combine with conj add, distribute, def of impl
  + Premises = ~(p -> q) ; ~p or s ; Conclusion=~(s -> q)
    - steps: def of implication to get p and ~q, disj/syll, combine back together
  + 



General strategies

* Look for obvious things first. (obvious MP, MT, DS)
* Simplify complicated expressions using logical Equivs.
  + Especially things with implications, as these can mask simple statements.
  + ~( (~p -> q) -> r) = ~( (p or q) -> r) = ~( ~(p or q) or r) = (p or q) and not r
* Use WANT list to work backwards

How to prove each kind of statement

How to prove an AND statement

Prove each side separately, combine with conj addition

How to prove an OR statement

See if you can prove one side, then add the other with disj addition

**Day 4 - Conditional Worlds**

Start with warmup problem, #41 above (hint, there is a step where you use disj add)

Cover general strategies above, including how to prove an and statement, how to prove an OR/IMPLICATION statement.

A conditional world is a proof technique that is almost always used to prove an implication.

* If it’s sunny, I’m going to Shelby Farms. (s -> f)
* If I go to Shelby farms, I will either bike or run. (f -> b OR r)
* Biking makes me happy (b -> h).
* Running makes me happy (r -> h)
* I’d like to conclude that if it’s sunny, I’m going to be happy.
* When doing a proof, we are always given some premises to start that we know are true.
* Inside of our world, we know all these things are true.
* (Draw picture of the world, list our premises inside the world).
* We know that in our world, it’s either going to be sunny this weekend or not, but we don’t know which one.
* But we are being asked to prove something about only what happens if it’s sunny. We’re not being asked to prove anything about what happens if it’s not sunny.
* So let’s hypothesize about what would happen if it’s going to be sunny.
  + Inside this box, this represents a world where it is sunny – this is a hypothesis, a statement that for the moment, we are assuming to be true, but only inside this world. It’s not true in the larger, outside world.
  + Called a conditional world. Inside a conditional world, all the same rules from the outside world apply, but not vice versa.
  + Show how a conditional world is represented in a formal proof tabular format.
  + Finish proof.
  + Close conditional world.

Rule of thumb: when asked to prove an implication, immediately jump into a conditional world.

2nd example: p->r, q->r, conc=(p or q) -> r

earlier, we did this one without a conditional world. Much easier with one.

3rd example:

P1: ~x -> (p and q)

P2: r -> ~p

P3: ~r -> y

Prove: x or y

Show by assuming ~x ; redo by assuming ~y

**Proof by contradiction**

* Sometimes you’ll be doing a proof and you just get stuck. You try lots of different things and nothing is working.
* One thing to try is a proof by contradiction.
* Proof by contradiction uses the same conditional world technique, but it can be used to prove anything, not just a conditional statement.
* The key behind the idea is that you cannot have a conditional world where a contradiction is true. Contradictions are false by definition. If you ever are inside a conditional world and derive a contradiction, then you know the world cannot exist, and the hypothesis that you used to create the world MUST BE FALSE.
  + Shortcut for the contradiction rule: ~p -> c; therefore p.
* Start by creating a conditional world where the NEGATION of what you want to prove is true.
* Derive a contradiction inside this world. Must be of the form p and ~p.
* Close the world, and your original statement may be negated outside the world.

Do this one assuming ~(x or y)

* P1: ~x -> (p and q)
* P2: r -> ~p
* P3: ~r -> y
* Prove: x or y

2nd example:

P1: (p -> q) -> r

P2: ~s

P3: (~q -> s) or ~p

Prove: r

Do this with and without contra

PRACTICE

P1 r OR (~y -> z)

P2 x -> (~y and s)

P3 (z and s) -> q

P4 ~(r and x)

Prove x -> q longish, but not hard, 2/5 in forward direction

proving ~q -> ~x is pretty challenging, 4/5.

proof by contra is 3/5 difficulty (assuming ~(x -> q))

P1: (a and ~b) -> d

P2: d or a

P3: e -> (~b or f)

Prove: ~(d or f) -> ~e Pretty challenging. 3/5 difficulty; 4/5 in opposite direction