Unit 1
Intro to propositional logic

Computer Science 1002

Introduction to Logic for Computer Scientists

1

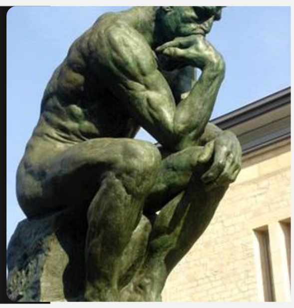
Logic is Everywhere!

- In the computer systems that reason and make decisions
- In the hardware circuits on which these systems are in built
- In the specifications describing precisely what these systems should and should not do
- In proofs that they are secure
- In the communication protocols on which Internet runs…

2

2

Do we ourselves think logically?



3

3

Do We Think Logically?
Let's try to solve this card puzzle

You see the following cards. Each has a letter on one side and a number on the other.

B 5 2 J

Which cards do you have to turn to check that for the cards in front of you if a card has a J on it then it has a 5 on the other side?

4

4

5

Do We Think Logically?
Let's try to solve this card puzzle

You see the following cards. Each has a letter on one side and a number on the other.

B 5 2 J

Which cards do you have to turn to check that for the cards in front of you if a card has a J on it then it has a 5 on the other side?

6

6

Card puzzle

Which cards do we have to flip to verify this:
- If a card has a J then it has a 5 on the other side

B 5 2 5

We do need to turn the card with J, since if there is no 5 on the other side, then "if J then 5 on the other side" would be false.

7

Card puzzle

Which cards do we have to flip to verify this:
- If a card has a J then it has a 5 on the other side

B C 2 J

Suppose we turn the card with 5 on it and find a C. This does not tell us anything about cards with J having 5 on the other side.

8

Card puzzle

Which cards do we have to flip to verify this:
- If a card has a J then it has a 5 on the other side

B 5 J J

Suppose we turn a card with 2 on it, and see an J.
Then we got a card with a J, but without 5!
So "If a card has a J then it has to have a 5" would not be true.

9

Card puzzle

Which cards do we need to flip to verify this:
- If a card has a J then it has a 5 on the other side

B 5 2 J

- All cards where J is visible
- And all cards with numbers other than 5 visible.

10

"If … Then" in Logic

The sentence in this puzzle has a logical structure:

"if A then B" J → 5

What circumstances make this true?

A is true and B is true	J 5	👍
A is true and B is false	J 2	👎
A is false and B is true	B 5	👍
A is false and B is false	B 2	👍

11

If A then B

J 5

- Sometimes people think that "if … then" goes both ways…
  - But usually it doesn't!
- If you live in NL, you must pay HST tax.
  - John lives in BC.
  - Does John pay HST?

- A politician says: " If I am elected I will build a regional transit system"
  - When would you say that she did not deliver on her promise?
  - Would you hold her accountable if she is not elected?

12

13

Why not just use English?

- Natural languages are ambiguous.
- For example, the word "any" can have different meanings depending on the context:

- Any = some
  - Can I solve **any** puzzle?
  - Can I solve **some (even one)** puzzle?
- Any = all
  - **Any** student knows this.
  - **Every** student knows this.

- **Any** student can get an A
- Would **any** student get an A?

14

14

What is Logic?

- Very informally, a logic system has the following parts:
  - A **language (syntax)**: what symbols are allowed, and how to combine them into correctly built statements
  - The **meaning (semantics)**: what are possible values of symbols (here, true and false), and how they combine to give values to statements.
  - The **reasoning (rules of inference)**: rules for deriving new statements: constructing arguments and proofs

15

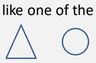
Logic is ancient!

- Already known in Babylon, 11 century BC
  - Esagil-kin-apli's "Sakikkū" ("Symptoms")
- Several schools of logic in India in 6th – 5th century BC
- Mohism in China
  - And then politics interfered…
- Aristotle, etc in ancient Greece
  - This is considered the origin of western formal logic.

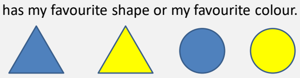
Soon after humans started to make sense of the world, they started inventing the rules of logic.

16

Shapes and colours puzzle

- I like one of the shapes.
- 
- I like one of the colours.

- I like a figure if it has my favourite shape or my favourite colour.



- I like  . What can you say about the rest?

17

18

Language of logic: propositions

- **Proposition**: A sentence stating a fact that can be *true* or *false*.
  - "Ice is colder than water"
  - "MUN is a university"
  - "2+2=7"
  - But not "Hi!" or "what is x?" or "Open this book!"

When we say "proposition", we usually mean a sentence stating a single fact, rather than a "compound proposition" such as "Sky is blue and snow is white".

For the first two weeks, we will study the logic which operates with propositions and their combinations: **propositional logic**.

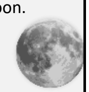
19

Propositional variables

- **Propositional variables:**
  - A, B, C (or p, q, r)
  - Variables that stand for propositions, and evaluate to TRUE or FALSE depending whether the corresponding proposition is true or false.

Similar to arithmetic variables:

- If you weight x kg on Earth, then you weight $x/6$ kg on the Moon.
  - Here, the arithmetic variable x stands for your weight on Earth in kg.
  - If you weight 60kg, then the value of x is 60
  - In this case, arithmetic expression $x/6$ evaluates to 10:
    - you weight 10kg on the Moon.



20

Language of logic: connectives

Pronunciation	Notation	Meaning
Not A (negation)	¬ A	Opposite of A is true, ¬A is true when A is false
A and B (conjunction)	A ∧ B	True if both A and B are true
A or B (disjunction)	A ∨ B	True if either A or B are true (or both)
If A then B (implication)	A → B	True whenever if A is true, then B is also true

21

Language of logic: connectives

Pronunciation	Notation	Meaning
Not A (negation)	¬ A	Opposite of A is true, ¬A is true when A is false
A and B (conjunction)	A ∧ B	True if both A and B are true
A or B (disjunction)	A ∨ B	True if either A or B are true (or both)
If A then B (implication)	A → B	True whenever if A is true, then B is also true

- A: "It is sunny"
- B: "it is cold"

- ¬ A: It is not sunny
- A ∧ B: It is sunny and cold
- A ∨ B: It is sunny or cold
- A → B: If it is sunny, then it is cold

22

Building logic formulas

- In arithmetic you build formulas out of numbers, variables and operations
  - $x + 3 * z$
- In logic we build formulas out of propositions, TRUE, FALSE and connectives.
  - The simplest formulas are propositions, propositional variables, TRUE and FALSE: "sky is blue", A, B, $TRUE$, $FALSE$ …
  - If A and B are logic formulas, then so are $\neg A \quad A \wedge B \quad A \vee B \quad A \rightarrow B$

Connective	Notation
A and B	A ∧ B
A or B	A ∨ B
If A then B	A → B
Not A	¬A

23

Translating logic formulas

- Let A be "It is sunny", B be "it is cold", C be "It's snowing"
  - $(B \wedge C) \rightarrow (\neg A)$ IF (AND) THEN NOT
    - If it is cold and snowing, then it is not sunny
  - $B \rightarrow (C \vee A)$ IF THEN (OR)
    - If it is cold, then it is snowing or sunny
  - $((\neg A) \wedge A) \rightarrow C$ IF (NOT AND) THEN
    - If it is not sunny, and also sunny, then it is snowing.

Connective	Notation
A and B	A ∧ B
A or B	A ∨ B
If A then B	A → B
Not A	¬A

24

25

We now know how to write logic sentences

But what do they actually mean?

26

The truth

- A sentence can be true or false when the values of all its propositions are known.
  - To know whether it is sunny and cold outside, need to know both whether it is sunny, and whether it is cold.
- **Truth assignment:** setting all propositional variables to either true or false.
  - Is it sunny? – Yes! A=true
  - Is it cold? – No! , B=false
  - Is it snowing? – Yes! C= true
  - Truth assignment: A=true , B=false , C= true

27

The truth

- **Satisfying assignment** for a sentence: assignment to propositions that makes this sentence true.
  - Otherwise, **falsifying** assignment.
  - Truth assignment A=true, B=true **satisfies** A ∧ B
    - Is it both sunny and cold? – Yes!
  - Truth assignment A=true, B=false **falsifies** A ∧ B
    - Is it both sunny and cold? – No!

28

Truth value of a sentence

- Truth assignment A=true, B=false **falsifies** A ∧ B
  - Is it both sunny and cold? – No! Because it is not cold.
  - The only way to satisfy $A \land B$ is for both A and B to be true.
- But A=true, B=false satisfies $A \lor B$
  - Is it either sunny or cold? – Yes! Because it is sunny.
  - The only way to falsify $A \lor B$ is for both A and B to be false.

29

Truth of "or"

- Note: the "or" we use in logic, ∨, is not exclusive.
  - A=true, B=true assignment satisfies $A \lor B$
  - Is it cold or snowing outside (and so I need a jacket)?
    - Yes, it is actually both cold and snowing outside, so you definitely need a jacket!

30

When is "if … then" true?

- Recall the cards puzzle: If there is J on one side, then there is 5 on the other

"if A then B" J → 5

- What circumstances (truth assignments) make this true?
  - A is true and B is true — J 5 — Satisfying
  - A is true and B is false — J 2 — Falsifying
  - A is false and B is true — K 5 — Satisfying
  - A is false and B is false — K 2 — Satisfying

31

32

Twins Puzzle

- There are two identical twin brothers, Dave and Jim.
- One of them always lies; another always tells the truth.
- Suppose you see one of them and you want to find out his name.
- How can you learn if you met Dave or Jim by asking just one short yes-no question? You don't know which one of them is the liar.

33

33

Twins puzzle

- Let us look at how different questions get answered in all possible scenarios.
- You could be talking to Jim (or not), and Jim could be the liar (or not)
  - Gives us 4 possible scenarios.

This is Jim	Jim is a liar	Question
Yes	**Yes**	Answer
Yes	**No**	Answer
No	**Yes**	Answer
No	**No**	Answer

34

Twins puzzle

- Suppose you ask directly "Are you Jim?"
  - In this case both Jim and Dave can say "yes" and can say "no".
  - So you cannot find out who you are talking to, but you do learn something:
  - If the answer is "Yes", then Dave is the liar

This is Jim	Jim is a liar	Are you Jim?
Yes	**Yes**	No
Yes	**No**	Yes
No	**Yes**	No
No	**No**	Yes

35

Twins puzzle

- Suppose you ask "Is 2+2=4?"
  - Again both Jim and Dave can say "yes" and can say "no", so you cannot find out who you are talking to
  - But again you do learn something: if the answer is "No", then you are talking to the liar

This is Jim	Jim is a liar	Is 2+2=4?
Yes	**Yes**	No
Yes	**No**	Yes
No	**Yes**	Yes
No	**No**	No

36

Twins puzzle

- How about asking "Is Dave a liar?"
  - Now, you can see from the table that Jim would have to say "yes" no matter if he is a liar or not. And Dave would always say "No".
  - So if the answer is "Yes", you are talking to Jim, and if "No", to Dave.

This is Jim	Jim is a liar	Is Dave a liar?
Yes	**Yes**	Yes
Yes	**No**	Yes
No	**Yes**	No
No	**No**	No

37

Twins puzzle

- So "Is Dave a liar?" lets you determine who you are talking to. But this time you only learn the name, not who is the liar.
  - This is unavoidable: you need two questions to single out each specific scenario out of 4.
  - If you are allowed to ask any two questions we looked at, you would know both who is the liar and who you are talking to.

This is Jim	Jim is a liar	Are you Jim?	Is 2+2=4?	Is Dave a liar?
Yes	**Yes**	No	No	Yes
Yes	**No**	Yes	Yes	Yes
No	**Yes**	No	Yes	No
No	**No**	Yes	No	No

38

39

Truth tables

A	B	not A $\neg A$	A and B $A \wedge B$	A or B $A \vee B$	if A then B $A \rightarrow B$
True	***True***	False	True	True	True
True	***False***	False	False	True	False
False	***True***	True	False	True	True
False	***False***	True	False	False	True

- Write down truth value for every possible scenario (assignment)
  - All combinations of true/false values of propositions: one per table line
  - Every new variable doubles table size.
    - With 2 variables, have 4 lines. With 3 variables, 8, etc.

40

- Let
  - A be "It's sunny"
  - B be "it's cold"

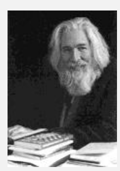
A	B	not A $\neg A$	A and B $A \wedge B$	A or B $A \vee B$	if A then B $A \rightarrow B$
True	***True***	False	True	True	True
True	***False***	False	False	True	False
False	***True***	True	False	True	True
False	***False***	True	False	False	True

- When it is true that it's sunny and true that it's cold, it is true that it's sunny or cold.
  - When A is true, B is true, then $A \vee B$ is true
- When it's sunny is true, but it's cold is false, then "it is both sunny and cold" is false.
  - When A is true, B is false, then $A \wedge B$ is false.

41

Knights and knaves

- On a mystical island, there are two kinds of people: knights and knaves.
-  Knights always tell the truth.
- Knaves always lie. 



Raymond Smullyan

42

Knights and knaves

- On a mystical island, there are two kinds of people: knights and knaves. Knights always tell the truth. Knaves always lie.

- Puzzle 1:
  - You meet two people on the island, Arnold and Bob.
  - Arnold says “I am a knave, or Bob is a knight”.
  - Is Arnold a knight or a knave?
  - What about Bob? Is Bob a knight or a knave?

43

44

Evaluating longer formulas

- We now know how to determine when $\neg A$, $A \vee B$, $A \wedge B$, and $A \rightarrow B$ are true
  - As long as we know what are the truth values of A and B

Pronunciation	Notation	True when
Not A	¬ A	Opposite of A is true
A and B	A ∧ B	Both A and B are true
A or B	A ∨ B	Either A or B is true (or both)
If A then B	A → B	if A is true, then B is also true; Also true when A is false, for any B.

- What if we have a longer formula?

In what order should we apply the connectives?

$$A \wedge \neg B \vee \neg(C \rightarrow A) \rightarrow A$$

45

Order of precedence

- Remember arithmetic: PEMDAS/BODMAS…
  - 6 + -5 * 7 + 8 = (6+ ((-5)* x)) + 8
    - First negate 5, then multiply -5 and x, then add this to 6, add result to 8.
    - Order: parentheses, unary -, then *, then +.

- In logic formulas
  1. Parentheses as in arithmetic formulas
  2. Then negation (¬) : like unary minus
  3. then AND (∧): like times (*)
  4. then OR (∨): like plus (+)
  5. Only then "if … then" (→)
  6. When several of the same, go from left to right
     - except →, which is right to left

Pronunciation	Notation	True when
Not A	¬ A	Opposite of A is true
A and B	A ∧ B	Both A and B are true
A or B	A ∨ B	Either A or B is true (or both)
If A then B	A → B	if A is true, then B is also true; Also true when A is false, for any B.

46

Order of precedence

$$A \wedge \neg B \vee \neg(C \rightarrow A) \rightarrow A$$

$$\big((A \wedge (\neg B)) \vee (\neg(C \rightarrow A))\big) \rightarrow A$$

- Get the second formula by *fully parenthesizing* first

Pronunciation	Notation	True when
Not A	¬ A	Opposite of A is true
A and B	A ∧ B	Both A and B are true
A or B	A ∨ B	Either A or B is true (or both)
If A then B	A → B	if A is true, then B is also true; Also true when A is false, for any B.

- Make sure to keep track of parentheses and order of operations!!!
  - $A \vee B \wedge C$ is not the same as $(A \vee B) \wedge C$
    - Just like $2 + 3 * 4 \neq (2 + 3) * 4$
  - When A is true, but both B and C are false,
    - $A \vee (B \wedge C)$ is true,
    - but $(A \vee B) \wedge C$ is false.

47

48

Better way: syntax trees

- Syntax tree: visualize a formula
  - The last operation on top
  - The numbers/variables at the bottom.
  - Branch points (nodes) marked by operations.

2+3*4 is a sum of two formulas
- 2 and 3*4
- 3*4 is a product of two formulas
  - 3 and 4.

To compute a value at a node
- compute all values below it
- then apply the operation marking the node.

When computed the value at the top, done

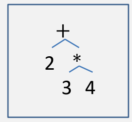
49

Better way: syntax trees

- Syntax tree: visualize a formula
  - The last operation on top
  - The numbers/variables at the bottom.
  - Branch points (nodes) marked by operations.

2+3*4 is a sum of two formulas
- 2 and 3*4
- 3*4 is a product of two formulas
  - 3 and 4.

To compute a value at a node
- compute all values below it
- then apply the operation marking the node.



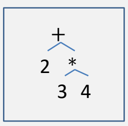
When computed the value at the top, done

50

Better way: syntax trees +

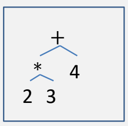
2 + 3 * 4 = 2 + (3 * 4)

2+3*4 is a sum of
- 2 and 3*4
- 3*4 is a product of
  - 3 and 4.



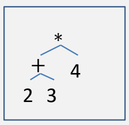
2 * 3 + 4 = (2 * 3) + 4

2*3+4 is a sum of
- 2*3 and 4
- 2*3 is a product of
  - 2 and 3.



(2 + 3) * 4

(2+3)*4 is a product of
- 2+3 and 4
- 2+3 is a sum of
  - 2 and 3.

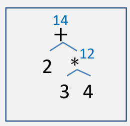


51

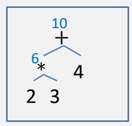
Evaluating formulas with syntax trees

- To compute a value at a node, compute all values below it,
- then apply the operation marking the node.
- When reached the top, done.

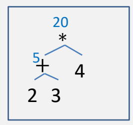
2 + 3 * 4 = 2 + (3 * 4)



2 * 3 + 4 = (2 * 3) + 4



(2 + 3) * 4



52

Building syntax trees

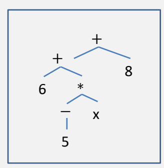
- Find the last operation to do
  - Such as the rightmost +
    - outside of parentheses
    - If no + outside, rightmost outside *
- Split the formula into left and right of that operation.
  - Build syntax trees for left and right formulas separately.
  - Connect the last operation (top) to syntax trees for left and right formulas.

- 6 + -5 * x + 8

53

+ Building syntax trees

- Find the last operation to do
  - Such as the rightmost +
    - outside of parentheses
    - If no + outside, rightmost outside *
- Split the formula into left and right of that operation.
  - Build syntax trees for left and right formulas separately.
  - Connect the last operation (top) to syntax trees for left and right formulas.



- 6 + -5 * x + 8 = (6+ ((-5)* x)) + 8

54

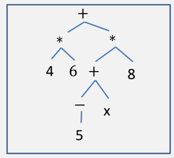
Building syntax trees

- Find the last operation to do
  - Such as the rightmost +
    - outside of parentheses
    - If no + outside, rightmost outside *
- Split the formula into left and right of that operation.
  - Build syntax trees for left and right formulas separately.
    - A single variable or constant is the smallest tree that cannot be split further
  - Connect the last operation (top) to syntax trees for left and right formulas.
- 4* 6 + (-5 + x) * 8

55

Building syntax trees

- Find the last operation to do
  - Such as the rightmost +
    - outside of parentheses
    - If no + outside, rightmost outside *
- Split the formula into left and right of that operation.
  - Build syntax trees for left and right formulas separately.
  - Connect the last operation (top) to syntax trees for left and right formulas.



- 4* 6 +(-5 + x) * 8= (4*6)+ (((-5)+ x)* 8)

56

57

Syntax trees for logic formulas

- Precedence:
  - ¬ first, then ∧, then ∨, → last
  - ¬ is like a unary minus, ∧ like * and ∨ like +
- Syntax tree:
  - The last operation on top
  - Variables on the bottom.
  - Branch points marked by connectives.
- $A \vee B \wedge C$ is $A \vee (B \wedge C)$
  - which is different from $(A \vee B) \wedge C$
  - Like 2+3*4 is different from (2+3)*4

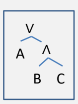
$A \vee (B \wedge C)$

$(A \vee B) \wedge C$

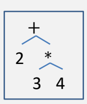
58

Syntax trees for logic formulas

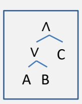
- Precedence:
  - ¬ first, then ∧, then ∨, → last
  - ¬ is like a unary minus, ∧ like * and ∨ like +
- Syntax tree:
  - The last operation on top
  - Variables on the bottom.
  - Branch points marked by connectives.
- $A \vee B \wedge C$ is $A \vee (B \wedge C)$
  - which is different from $(A \vee B) \wedge C$



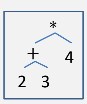
$A \vee (B \wedge C)$



2+3*4=14



$(A \vee B) \wedge C$



(2+3)*4=20

59

Syntax trees for logic formulas

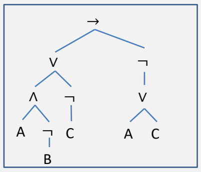
- Precedence:
  - Parentheses
  - then ¬ (like unary -)
  - then ∧ (like *)
  - then ∨ (like +)
  - Finally →
  - Left to right for ∨,∧
    - right to left for →

$A \wedge \neg B \vee \neg C \rightarrow \neg(A \vee C)$

60

Syntax trees for logic formulas

- Precedence:
  - Parentheses
  - then ¬ (like unary -)
  - then ∧ (like *)
  - then ∨ (like +)
  - Finally →
  - Left to right for ∨,∧
    - right to left for →



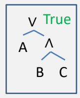
$A \wedge \neg B \vee \neg C \rightarrow \neg(A \vee C)$

61

Evaluating logic formulas with trees

Symbol	True when
¬ A	A is false
A ∧ B	Both A and B are true
A ∨ B	At least one of A, B true
A → B	A false or B true or both

- Let A be true, B false, C false



$A \vee (B \wedge C)$

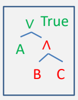
1. B and C are both false, "false and false" is false, so $B \wedge C$ is false
2. A is true, $B \wedge C$ is false. True or false is true.
3. So $A \vee (B \wedge C)$ is true

62

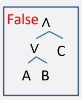
Evaluating logic formulas with trees

Symbol	True when
¬ A	A is false
A ∧ B	Both A and B are true
A ∨ B	At least one of A, B true
A → B	A false or B true or both

- Let A be true, B false, C false



$A \vee (B \wedge C)$



$(A \vee B) \wedge C$

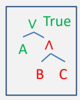
1. A is true, B is false. True or false is true.
2. . $A \vee B$ is true, but C is false. True and false is false.
3. So $(A \vee B) \vee C$ is false

63

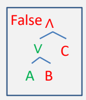
Evaluating logic formulas with trees

Symbol	True when
¬ A	A is false
A ∧ B	Both A and B are true
A ∨ B	At least one of A, B true
A → B	A false or B true or both

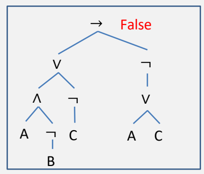
- Let A be true, B false, C false



$A \vee (B \wedge C)$



$(A \vee B) \wedge C$



$A \wedge \neg B \vee \neg C \rightarrow \neg(A \vee C)$

64

65

Knights and knaves

- On a mystical island, there are two kinds of people: knights and knaves. Knights always tell the truth. Knaves always lie.

- Puzzle 1:
  - You meet two people on the island, Arnold and Bob.
  - Arnold says "Either I am a knave, or Bob is a knight".
  - Is Arnold a knight or a knave?
  - What about Bob? Is Bob a knight or a knave?

66

Knights and knaves

- You meet two people on the island, Arnold and Bob. Arnold says "Either I am a knave, or Bob is a knight".
  - Propositional variable A:
    - True when Arnold is a knight
    - False when Arnold is a knave
  - Propositional variable B:
    - True when Bob is a knight,
    - False when Bob is a knave.
  - Statement $\neg A \vee B$: "Either Arnold is a knave, or Bob is a knight"
- Want: scenarios where
  - either both Arnold is a knight and what he said, $\neg A \vee B$, is true
  - or Arnold is a knave and $\neg A \vee B$ is false.

67

Iff (if and only if): a new connective

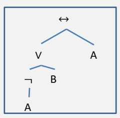
- Want: scenarios where
  - either both Arnold is a knight and what he said, $\neg A \vee B$, is true
  - or Arnold is a knave and $\neg A \vee B$ is false.
- Can write this using "if and only if" ("iff") notation: $(\neg A \vee B) \leftrightarrow A$.
- $G \leftrightarrow H$ is true when G and H have same value
  - Either both G and H are true,
  - Or both G and H are false
- Note: we do not use "=" between logic formulas!
- The symbol $\leftrightarrow$ is called "biconditional"
- Precedence order of $\leftrightarrow$ is even lower than $\rightarrow$

G	H	G if and only if H $G \leftrightarrow H$
True	*True*	*True*
True	*False*	*False*
False	*True*	*False*
False	*False*	*True*

68

Knights and knaves

- Arnold says "Either I am a knave, or Bob is a knight".
  - A: True when Arnold is a knight, false when Arnold is a knave
  - B: True when Bob is a knight, false when Bob is a knave.
  - Arnold said: $\neg A \vee B$: "Either Arnold is a knave, or Bob is a knight"
  - By rules of what it means to be a knight or a knave
    - $(\neg A \vee B) \leftrightarrow A$ must be true
  - Let's try to see when this happens.

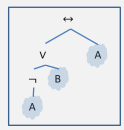


69

Knights and knaves

- Arnold says "Either I am a knave, or Bob is a knight".
  - A: True when Arnold is a knight, false when Arnold is a knave
  - B: True when Bob is a knight, false when Bob is a knave.
  - Arnold said: $\neg A \vee B$: "Either Arnold is a knave, or Bob is a knight"
  - By rules of what it means to be a knight or a knave
    - $(\neg A \vee B) \leftrightarrow A$ must be true
  - Let's try to see when this happens.

A	B			
True	*True*			
True	*False*			
False	*True*			
False	*False*			

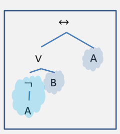


70

Knights and knaves

- Arnold says "Either I am a knave, or Bob is a knight".
  - A: True when Arnold is a knight, false when Arnold is a knave
  - B: True when Bob is a knight, false when Bob is a knave.
  - Arnold said: $\neg A \vee B$: "Either Arnold is a knave, or Bob is a knight"
  - By rules of what it means to be a knight or a knave
    - $(\neg A \vee B) \leftrightarrow A$ must be true
  - Let's try to see when this happens.

A	B	$\neg A$		
True	*True*	False		
True	*False*	False		
False	*True*	True		
False	*False*	True		

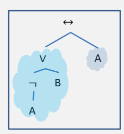


71

Knights and knaves

- Arnold says "Either I am a knave, or Bob is a knight".
  - A: True when Arnold is a knight, false when Arnold is a knave
  - B: True when Bob is a knight, false when Bob is a knave.
  - Arnold said: $\neg A \vee B$: "Either Arnold is a knave, or Bob is a knight"
  - By rules of what it means to be a knight or a knave
    - $(\neg A \vee B) \leftrightarrow A$ must be true
  - Let's try to see when this happens.

A	B	$\neg A$	$\neg A \vee B$	
True	*True*	False	True	
True	*False*	False	False	
False	*True*	True	True	
False	*False*	True	True	

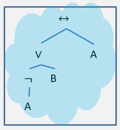


72

Knights and knaves

- Arnold says "Either I am a knave, or Bob is a knight".
  - A: True when Arnold is a knight, false when Arnold is a knave
  - B: True when Bob is a knight, false when Bob is a knave.
  - Arnold said: $\neg A \lor B$: "Either Arnold is a knave, or Bob is a knight"
  - By rules of what it means to be a knight or a knave
    - $(\neg A \lor B) \leftrightarrow A$ must be true
  - Let's try to see when this happens.

A	B	$\neg A$	$\neg A \lor B$	$(\neg A \lor B) \leftrightarrow A$
True	*True*	False	True	True
True	*False*	False	False	False
False	*True*	True	True	False
False	*False*	True	True	False



73

Knights and knaves

- Arnold says "Either I am a knave, or Bob is a knight".
  - By rules of what it means to be a knight or a knave
    - $(\neg A \lor B) \leftrightarrow A$ must be true
  - The only scenario when the rules of the puzzle are satisfied is the very first one, when both A and B are true.
  - So Arnold is a knight

And Bob is also a knight

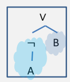
A	B	$\neg A$	$\neg A \lor B$	$(\neg A \lor B) \leftrightarrow A$
True	*True*	False	True	True
True	*False*	False	False	False
False	*True*	True	True	False
False	*False*	True	True	False

74

75

Truth tables

- List all possible scenarios: All the ways of setting variables to true, false.
  - Each time we add a new variable the number of possible scenarios doubles:
    - We need to look at all possible truth assignments to the previous variables twice: once for the new variable being true, and another for the new variable being false.
    - So a truth table has 2 rows for 1 variable, 4 for 2, 8 for 3, and in general 2^n for n variables.
- For each of them, find the value of the whole formula by evaluating bottom-up
  - Use the syntax tree.
  - Write down the value of each subformula (node of the subtree) as a column in the truth table.
    - Label the column with the formula at the subtree.
- Example: $(\neg A) \lor B$

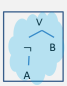


A	B	$\neg A$	
True	*True*	*False*	
True	*False*	*False*	
False	*True*	*True*	
False	*False*	*True*	

76

Truth tables

- List all possible scenarios: All the ways of setting variables to true, false.
  - Each time we add a new variable the number of possible scenarios doubles:
    - We need to look at all possible truth assignments to the previous variables twice: once for the new variable being true, and another for the new variable being false.
    - So a truth table has 2 rows for 1 variable, 4 for 2, 8 for 3, and in general 2^n for n variables.
- For each of them, find the value of the whole formula by evaluating bottom-up
  - Use the syntax tree.
  - Write down the value of each subformula (node of the subtree) as a column in the truth table.
    - Label the column with the formula at the subtree.
- Example: $(\neg A) \lor B$



A	B	$\neg A$	$\neg A \lor B$
True	*True*	*False*	*True*
True	*False*	*False*	*False*
False	*True*	*True*	*True*
False	*False*	*True*	*True*

77

Logical equivalence

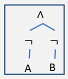
- Example: $(\neg A) \lor B$
  - Notice: its truth table's last column is the same as for $A \rightarrow B$
  - So $(\neg A) \lor B$ and $A \rightarrow B$ are **logically equivalent.**

A	B	$\neg A$	$\neg A \lor B$	$A \rightarrow B$
True	*True*	*False*	*True*	*True*
True	*False*	*False*	*False*	*False*
False	*True*	*True*	*True*	*True*
False	*False*	*True*	*True*	*True*

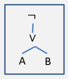
78

Logical equivalence

- Two formulas (on the same variables) are **logically equivalent** if all values in the columns of the truth tables labeled by them are the same.
  - $(\neg A) \vee B$ and $\mathrm{A} \rightarrow \mathrm{B}$ are logically equivalent.
  - So are $\neg(A \vee B)$ and $\neg A \wedge \neg B$, as well as $\neg(A \wedge B)$ and $\neg A \vee \neg B$
    - The last two equivalences are known as DeMorgan's laws.



A	B	$\neg A$	$\neg B$	$\neg A \wedge \neg B$	$A \vee B$	$\neg(A \vee B)$
True	*True*	*False*	*False*	*False*	*True*	*False*
True	*False*	*False*	*True*	*False*	*True*	*False*
False	*True*	*True*	*False*	*False*	*True*	*False*
False	*False*	*True*	*True*	*True*	*False*	*True*



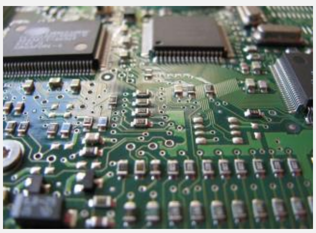
We will talk a lot more about logical equivalences next week

79

80

Logic in hardware

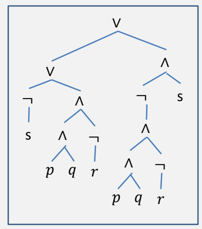
- When you study computer architecture, you will hear of "logic design" and "combinatorial circuits"
- So how is all that hardware related to the logic we talked about?



81

Circuits

- Circuits are a generalization of formulas that (can be) smaller.
  - More precisely, a generalization of syntax trees for formulas.
  - Main trick: draw a repeated subtree only once, connect everywhere it belongs.
  - Second trick: ∧ and ∨ nodes allow for more than two connections
    - Like summation and product in arithmetic.

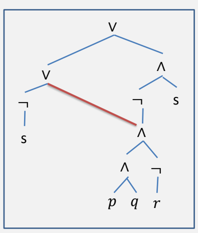


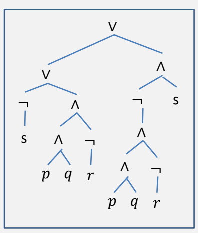
$\neg s \vee \ p \wedge q \wedge \neg\ r \ \vee \neg(p \wedge q \wedge \neg r) \wedge s$

82

Circuits

- Main trick: draw a repeated subtree only once, connect everywhere it belongs.

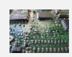




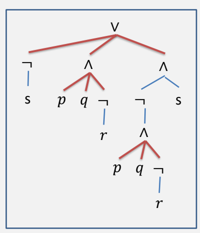
$\neg s \vee \ p \wedge q \wedge \neg\ r \ \vee \neg(p \wedge q \wedge \neg r) \wedge s$

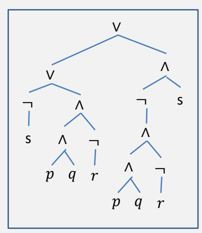
83

Circuits



- Second trick: ∧ and ∨ nodes allow for more than two connections
  - Like summation and product in arithmetic.



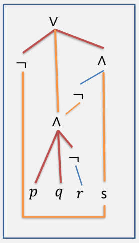


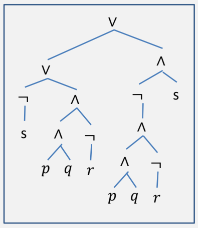
$\neg s \vee \ p \wedge q \wedge \neg\ r \ \vee \neg(p \wedge q \wedge \neg r) \wedge s$

84

Circuits

- Putting them together: much smaller picture!





$\neg s \vee p \wedge q \wedge \neg r \vee \neg(p \wedge q \wedge \neg r) \wedge s$

85

Circuits

Every formula's syntax tree is (essentially) a circuit.

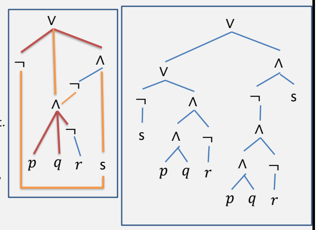
- Just write variables once and connect to corresponding places.

A circuit can be "unwound" into a syntax tree which can be then written as a formula

- the resulting tree can be much bigger than the circuit.

Circuit can be evaluated in same way as a formula

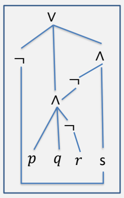
- Start from the bottom, and compute all values below a node before computing the value of a node.
  - Label truth table columns with expressions at the nodes
    - In fact, even for truth tables for formulas we do not recompute columns if we already computed them before: so we are implicitly using the "main trick"



$\neg s \vee p \wedge q \wedge \neg r \vee \neg(p \wedge q \wedge \neg r) \wedge s$

86

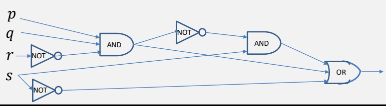
Circuits



- In digital logic circuit design, change in terminology:

∨	∧	¬	Node	Edge	TRUE	FALSE
OR	AND	NOT	Gate	Wire	1	0

  - Also often write the circuits sideways (left to right)
  - Variables are often called **inputs**; value of the formula is the **output**.



87

88

What do circuits compute?

- We can view evaluating a formula or a circuit as computing a function
  - inputs are values of propositional variables, each of which is true or false.
  - output is either *true* or *false.*
- A variable is **Boolean** if it takes only values 0 (false) or 1 (true):
  - A **Boolean function** takes a list of Boolean variables and outputs 0 or 1.
- Example: $Majority(x, y, z)$ is a Boolean function, with $Majority(x, y, z) = 1$ when at least two out of its inputs x, y, z are 1, and 0 otherwise.
  - Here, each of x, y, z must be either 0 or 1.
  - So $Majority(x, y, z)$ is 1 when at most one of its inputs is a 0.

89

Boolean functions

- A Boolean function is fully described by its truth table
  - For every possible combination of values of its inputs, the truth table says what the output is.

$Majority(x, y, z) = 1$
when at least two out of x, y, z are 1
and $Majority(x, y, z) = 0$ otherwise.

$x = 1?$	$y = 1?$	$z = 1?$	$Majority\,(x, y, z) = 1?$
True	True	True	True
True	True	False	True
True	False	True	True
True	False	False	False
False	True	True	True
False	True	False	False
False	False	True	False
False	False	False	False

90

Boolean functions

- A Boolean function is fully described by its truth table
  - For every possible values of its inputs, the truth table says what the output is.
  - Changing TRUE to 1 and FALSE to 0

$Majority(x, y, z) = 1$
when at least two out of x, y, z are 1
and $Majority(x, y, z) = 0$ otherwise.

x	y	z	$Majority(x,y,z)$
1	1	1	1
1	1	0	1
1	0	1	1
1	0	0	0
0	1	1	1
0	1	0	0
0	0	1	0
0	0	0	0

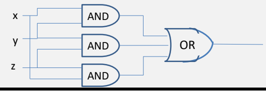
91

Computing Boolean functions

- Both formulas and circuits compute Boolean functions
  - that is, can tell what the value of a function is in a given input scenario (assignment of Boolean/truth values to variables).

Though technically propositional variables and logic formulas have values TRUE/FALSE, whereas Boolean variables and functions have values 1/0 (and syntax for connectives is different), we will often abuse the notation and use them interchangeably.

- Formula for $Majority(x, y, z)$:
  - Let A: "$x = 1$", B: "$y = 1$", C: "$z = 1$".
  - Then $Majority(x, y, z) = 1$ if and only if $(A \wedge B) \vee (A \wedge C) \vee (B \wedge C)$ is true
- Circuit for $Majority(x, y, z)$ is
  - Much more on circuits in other computer science courses.



92

93

Constructing a formula from a truth table

- For every formula (or circuit) there is a truth table.
- But also from every truth table we can construct a formula
  - and every formula can be directly converted into a circuit.
- So for every Boolean function we can construct a formula which is true exactly in the scenarios when this function is supposed to output 1.
- Moreover, we can construct formulas of a special form:
  - If we use only the second trick to represent them (multi-input ∨,∧)
  - Then the resulting generalized syntax trees have at most four layers

94

From a Boolean function to a formula

- Suppose there is a Boolean function f such that $f(1,0,0) = 1$.
  - Let's propositions for its inputs x, y, z respectively being 1 be A, B, C
    - A denotes "$x = 1$", if $x = 1$ then A is true, otherwise A is false; same for B,C
  - The input to f above is 1, 0, 0
    - $x = 1, y = 0, z = 0$, so A = True, B = False, C=False,
    - Then the formula F_f encoding function f above should be true on A=True, B=False, C=False.
  - We can now write a formula which is true only on this assignment:
    - $A \wedge \neg B \wedge \neg C$

95

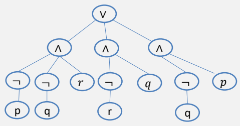
From a Boolean function to a formula

- To write a formula encoding the whole Boolean function f, write a formula encoding every assignment that makes f output 1.
  - Say $f(1,0,0) = 1$, $f(1,0,1) = 1$, and for any other input $f(x, y, z) = 0$
  - Then the corresponding assignments are $A \wedge \neg B \wedge \neg C$ and $A \wedge \neg B \wedge C$
- And finally take an OR of these formulas.
  - So the resulting formula would say "Either the formula is true because we are in the first scenario where f outputs 1, or the second, etc…"
  - $(A \wedge \neg B \wedge \neg C) \vee (A \wedge \neg B \wedge C)$

96

DNF (Sum of Products)

- **Disjunctive Normal Form**, or **DNF:** formulas that are an OR of ANDs of (possibly negated) variables.
  - Essentially the same as Sum of Products
    - With yet another notation for connectives: ∨ is +, ∧ is ⋅, 0 is FALSE, 1 is TRUE
  - Call a variable or negated variable a **literal**: $x, \neg y,\ p, \neg q, A, \neg B$
  - Call an ∧ of literals a **term**: $(x \wedge \neg y \wedge z),\ (\neg p \wedge \neg q),\ (A), (A \wedge B \wedge C)$

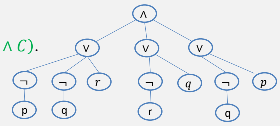


- $(A \wedge \neg C) \vee (\neg B) \vee (B \wedge C)$ is a DNF
- A∨ $\neg B \vee C$ is a DNF. So is $A \wedge \neg B \wedge C$.
- $\neg(A \wedge \ \neg B) \vee C$ is not a DNF

97

CNF (Product of Sums)

- **Conjunctive Normal Form**, or **CNF:** formulas that are an AND of ORs of (possibly negated) variables.
  - Also known as Product of Sums (after switching notation)
  - As for DNF, a variable or negated variable is a **literal**: $x, \neg y,\ p, \neg q, A, \neg B$
  - Call an ∨ of literals a **clause**: $(x \vee \neg y \vee z),\ (\neg p \vee \neg q), (A), (A \vee B \vee C)$

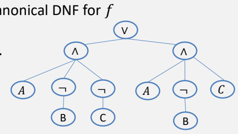


- $(B \vee \neg A) \wedge (\neg B) \wedge (B \vee C)$ is a CNF
- $(A \vee \neg B \vee C)$ is a CNF. So is $(A \wedge \neg B \wedge C)$.
- $(A \vee \ \neg B \wedge C)$ is not a CNF

98

Canonical DNF

- The resulting formula would say "Either the formula is true because we are in the first scenario where f outputs 1, or the second, etc…"
  - $(A \wedge \neg B \wedge \neg C) \vee (A \wedge \neg B \wedge C)$
- This formula is of the form OR of ANDs of literals: it is a DNF
- Such formula is called the **canonical DNF** of f if it has a term for every possible list of its input values (line of the truth table) on which f outputs 1.
  - That is, the construction we just did produced canonical DNF for f
- Every Boolean function f has a canonical DNF.
  - Can do this construction with any truth table.



99

Canonical DNF

x	y	$f(x,y)$
1	1	1
1	0	0
0	1	0
0	0	1

- Let's try with $f(x,y)$ which outputs 1 when either both $x = 1$ and $y = 1$, or $x = 0$ and $y = 0$
- Let A: "$x = 1$", B: "$y = 1$"
- Check when $f(x,y)$ outputs 1:
  - $x = 1, y = 1$, as well as $x = 0, y = 0$
  - Corresponds to $A \wedge B$, and $\neg A \wedge \neg B$
- Now, state that at least one of these cases happens:

$$(\boldsymbol{A} \wedge \boldsymbol{B}) \vee (\neg \boldsymbol{A} \wedge \neg \boldsymbol{B})$$

This is the canonical DNF for the $f(x,y)$ above.

100

Canonical CNF

- To construct a canonical CNF for f
  - Take every falsifying assignment $(f(x,y,z) = 0)$
    - Say, A = False, B = True, C=False.
  - Write a formula which is true only on this assignment:
    - $\neg A \wedge B \wedge \neg C$
  - To say that this assignment does not happen, say that at least one of the variables takes the opposite value of what it has in this assignment:
    - $A \vee \neg B \vee C$
  - Take an AND of these clauses for all falsifying assignments
    - "First bad thing does not happen, and second bad thing does not happen…"

101

Canonical CNF

x	y	$f(x,y)$
1	1	1
1	0	0
0	1	0
0	0	1

- Let's try with $f(x,y)$ which outputs 1 when either both $x = 1$ and $y = 1$, or $x = 0$ and $y = 0$
- Let A: "$x = 1$", B: "$y = 1$"
- First, look at when $f(x,y)$ outputs 0:
  - $x = 1, y = 0$, as well as $x = 0, y = 1$
  - Corresponds to $A \wedge \neg B$, and $\neg A \wedge B$
- State that they do *not* happen
  - $A \wedge \neg B$ does not happen: $(\neg A \vee B)$
  - $\neg A \wedge B$ does not happen: $(A \vee \neg B)$
- Finally, say that both do not happen: $(\neg \boldsymbol{A} \vee \boldsymbol{B}) \wedge (\boldsymbol{A} \vee \neg \boldsymbol{B})$
  - This is our canonical CNF for $f(x,y)$

102

103