CS2209A 2017

Applied Logic for Computer Science

Lecture 2

Propositional Logic:

Syntax, semantics, truth table

Instructor: Yu Zhen Xie

Language of logic: building blocks

- **Proposition**:
  A sentence that can be *true* or *false*.
  – A: "It is raining in St. John's right now".
  – B: "2+2=7"
  – But not "Hi!" or "x is an even number"
- **Propositional variables:**
  – A, B, C (or p, q, r)
  – It is a shorthand to denote propositions:
    - "B is true", for the B above, means "2+2=7" is true.

Language of logic: connectives

Pronunciation	Notation	Meaning
A and B (conjunction)	**A $\wedge$ B**	True if both A and B are true
A or B (disjunction)	**A $\vee$ B**	True if either A or B are true (or both)
If A then B (implication)	**A $\rightarrow$ B**	True whenever if A is true, then B is also true
Not A (negation)	$\neg$ **A**	Opposite of A is true, $\neg$A is true when A is false

- Let A be “It is sunny” and B be “it is cold”
  - A $\wedge$ B: It is sunny and cold
  - A $\vee$ B: It is either sunny or cold
  - A $\rightarrow$ B: If it is sunny, then it is cold
  - $\neg$ A : It is not sunny

Language of logic: syntax

Pronunciation	Notation	True when
A and B	A $\wedge$ B	Both A and B must be true
A or B	A $\vee$ B	Either A or B must be true (or both)
If A then B	A $\rightarrow$ B	if A is true, then B is also true
Not A	$\neg$ A	Opposite of A is true

- Now we can combine these operations to make longer formulas
- $\boldsymbol{A} \wedge \neg\, \boldsymbol{B} \vee \neg \boldsymbol{C} \rightarrow \boldsymbol{A}$
  - If it is either sunny and not cold or not snowing, then it is sunny.
- How is the precedence of the connectives?

Language of logic: syntax

Pronunciation	Notation	True when
A and B	A ∧ B	Both A and B must be true
A or B	A ∨ B	Either A or B must be true (or both)
If A then B	A → B	if A is true, then B is also true
Not A	¬ A	Opposite of A is true

- **Precedence**: ¬ first, then ∧, then ∨, → last

  ¬ is like a unary minus, ∧ like *, ∨ like +

  ¬A ∨ B ∧ C is like -5+3*8

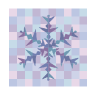
- When in doubt or need a different order, use parentheses

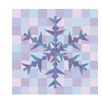
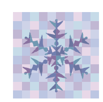
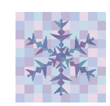
  $\boldsymbol{A} \land \neg \boldsymbol{B} \lor \neg \boldsymbol{C} \rightarrow \boldsymbol{A}$ is $\big((A \land (\neg B)) \lor (\neg C)\big) \rightarrow A$

  $A \lor B \land C$ is **not** the same as $(A \lor B) \land C$

Language of logic: semantics

Pronunciation	Notation	True when
A and B	A ∧ B	Both A and B must be true
A or B	A ∨ B	Either A or B must be true (or both)
If A then B	A → B	if A is true, then B is also true
Not A	¬ A	Opposite of A is true

- Let
  - A be "It is sunny", 
  - B be "it is cold", 
  - C be "It's snowing" 

- What are the translations of:
  - $B \wedge C \rightarrow \neg A$ 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 either snowing or sunny
  - $\neg A \wedge A \rightarrow C$ IF (NOT  AND ) THEN 
    - If it is sunny and not sunny, then it is snowing.

The truth

The truth

- We talk about a sentence being true or false when the values of the variables are **known**.
  - If we didn't know whether it is sunny, we would not know whether **$A \wedge B \rightarrow C$** is true or false.
- **Truth assignment:** setting values of variables to true/false.

  e.g. A=true, B=false, C=false
- **Satisfying assignment** for a sentence: assignment that makes it true.
  - (Otherwise, **falsifying** assignment).
  - A=true, B=false, C= false satisfies $A \wedge B \rightarrow C$
  - A=true, B=true, C=false falsifies $A \wedge B \rightarrow C$

Truth assignment

Let
- A be “It is sunny” 
- B be “It is cold” 
- C be “It is snowing 

A
True
False

- It is sunny
- It is not sunny

B
True
False

- It is cold
- It is not cold

C
True
False

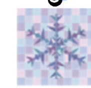
- It is snowing
- It is not snowing

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

- It is sunny and cold
- It is sunny and not cold
- It is not sunny and cold
- It is neither sunny nor cold

Truth assignment

Let

- A be “It is sunny” 
- B be “It is cold” 
- C be “It is snowing 

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

Given **n** variables, how many different truth assignments will there be?

Answer: $\mathbf{2^n}$

Truth tables

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

- Let
  - A be “It is sunny” 
  - B be “it is cold” 
- It is sunny and cold.
- It is sunny and not cold
- It is not sunny and cold
- It is neither sunny nor cold

A	B	not A	A and B
True	*True*	False	True
True	*False*	False	False
False	*True*	True	False
False	*False*	True	False

Truth table: V

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

- "It is raining or I am a dolphin"

Truth table: $\mathbf{p \rightarrow q}$

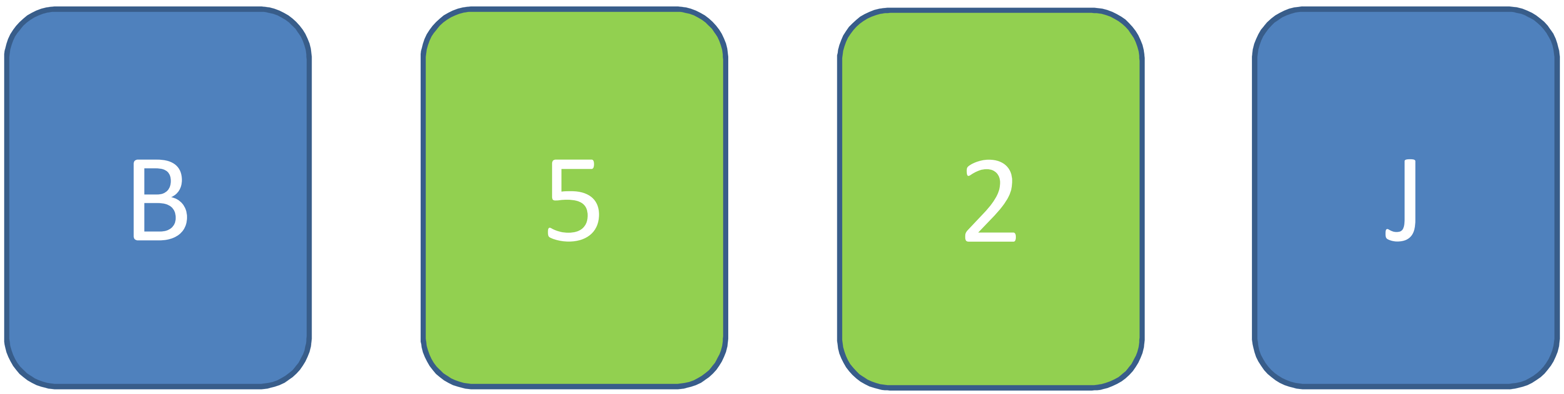
p	q	if p then q
True	*True*	True
True	*False*	**False**
False	*True*	**True**
False	*False*	**True**

- Let
  - p be "*It is raining*"
  - q be "*It is cloudy*"
  - **"If p then q"**
  - **"p implies q"**

- The implication is only **false** if its left hand side (i.e., ***p***) is true while the right hand side (***q***) is false.
- That is, "*if it is raining then it is cloudy*" is **false** only when it is raining out of blue sky. If it is not raining, this propositional formula is true no matter whether it is cloudy or not.

The fun game

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

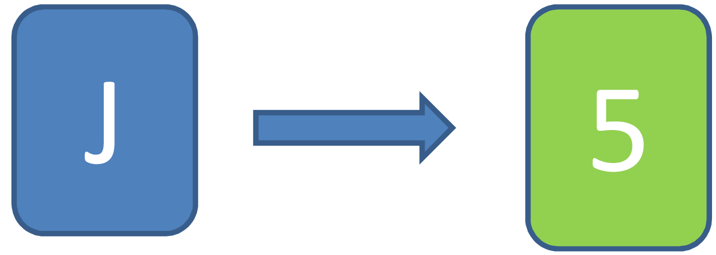


- Which cards do you need to turn to check that **“if a card has a J on it then it has a 5 on the other side”?**

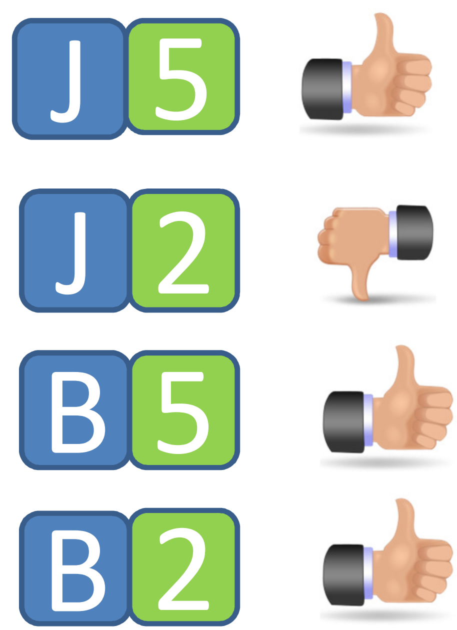
“if … then” in logic

- This puzzle has a logical structure:

“if A then B”



- What circumstances make this true?
  - A is true and B is true
  - A is true and B is false
  - A is false and B is true
  - A is false and B is false



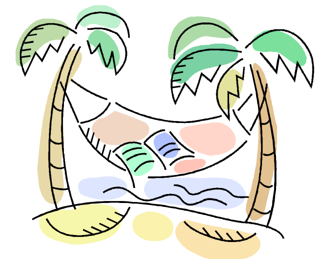
Language of logic

Pronunciation	Notation	Meaning
A and B (conjunction)	**A** $\wedge$ **B**	True if both A and B are true
A or B (disjunction)	**A** $\vee$ **B**	True if either A or B are true (or both)
If A then B (implication)	**A** $\rightarrow$ **B**	True whenever if A is true, then B is also true
Not A (negation)	$\neg$ **A**	Opposite of A is true, $\neg$A is true when A is false

- "If and only if", **iff**, $\leftrightarrow$
  - A $\leftrightarrow$ B:
    - A $\rightarrow$ B and B $\rightarrow$ A
    - A if and only if B
    - A and B either both true or both false

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?

Knights and knaves

- Puzzle 1: Arnold says "Either I am a knave, or Bob is a knight". **Is Arnold a knight or a knave? What about Bob?**
  To solve:
  - ***A***: Arnold is a knight
  - ***B***: Bob is a knight
  - Formula: $\neg \boldsymbol{A} \vee \boldsymbol{B}$: "Either Arnold is a knave, or Bob is a knight"
  - Want: scenarios where either both **A is a knight** and **the formula is true**, or **A is a knave** and **the formula is false**.
  - Use "if and only if" notation: $(\neg \boldsymbol{A} \vee \boldsymbol{B}) \leftrightarrow \boldsymbol{A}$
    True if both formulas have same value.

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

Truth tables: equivalence

A	B	not A	if A then B	(not A) or B
True	*True*	False	True	True
True	*False*	False	False	False
False	*True*	True	True	True
False	*False*	True	True	True

- Now, $\neg \boldsymbol{A} \vee \boldsymbol{B}$ is the same as $\mathrm{A} \rightarrow \mathrm{B}$
  - So $\neg A \vee B$ and $\mathrm{A} \rightarrow \mathrm{B}$ are **equivalent.**

❖ "if it rains it must be cloudy" is equivalent to say "it can't happen that both it's not cloudy and raining".

Special types of sentences

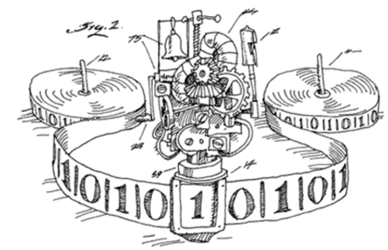
- A sentence that has a satisfying assignment is **satisfiable.**
  - *Some* row in the truth table ends with *True*.
  - Example: $\mathbf{B} \rightarrow \mathbf{A}$
- Sentence is a **contradiction:**
  - All assignments are falsifying.
  - *All* rows end with *False*.
  - Example: $\boldsymbol{A} \wedge \neg \boldsymbol{A}$
- Sentence is a **tautology:**
  - All assignments are satisfying
  - *All* rows end with *True*.
  - Example: $\mathbf{B} \rightarrow \mathbf{A} \vee \boldsymbol{B}$

A	B	$\mathrm{B} \rightarrow \mathrm{A}$
True	*True*	*True*
True	*False*	*True*
False	*True*	*False*
False	*False*	*True*

A	$\mathrm{A} \wedge \neg\mathrm{A}$
True	*False*
False	*False*

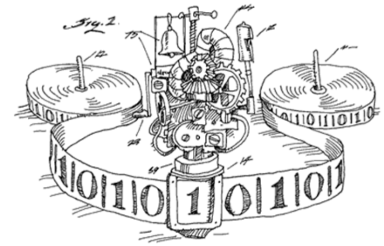
A	B	$\mathrm{A} \vee \boldsymbol{B}$	$\mathrm{B} \rightarrow \mathrm{A} \vee \boldsymbol{B}$
True	*True*	*True*	*True*
True	*False*	*True*	*True*
False	*True*	*True*	*True*
False	*False*	*False*	*True*

Determining formula type

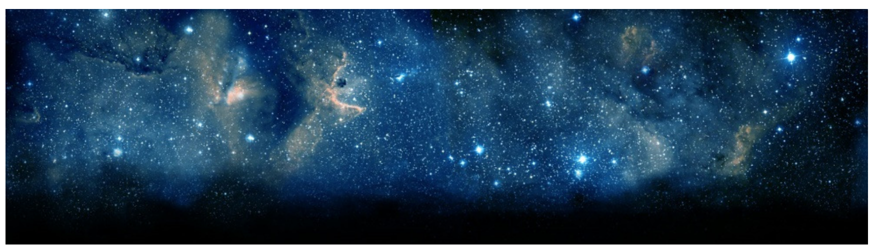


- How long does it take to check if a formula is satisfiable?
  - If somebody gives you a satisfying assignment, then in time roughly the size of the formula.
    - On a *m*-symbol formula, take time O(*m*) = constant * *m*, for some constant depending on the computer/software.
  - What if you don't know a satisfying assignment? How hard it is to find it?
    - Using a truth table: in time $O(m * 2^n)$ on a length *m* *n*-variable formula.
    - Is it efficient?...

Complexity of computation

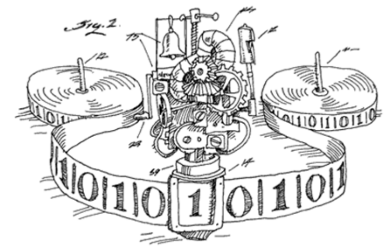


- Would you still consider a problem really solvable if it takes very long time?
  - Say 10^n steps on an n-symbol string?
  - At a billion (10^9) steps per second (~1GHz)?
  - To process a string of length 100…
  - will take $10^{100}/10^9$ seconds, or ~$3x10^{72}$ centuries.

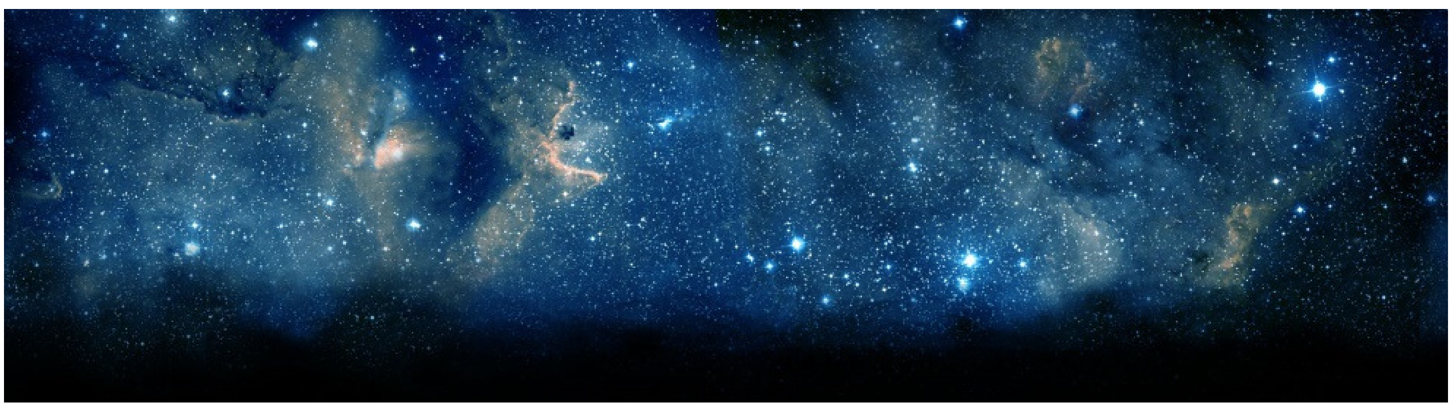


  - Age of the universe: about $1.38x10^{10}$ years.
  - Atoms in the observable universe: 10^{78}-10^{82}.

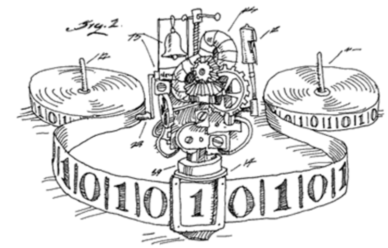
Complexity of computation



- What strings do we work with in real life?
  - A DNA string has 3.2×10^9 base pairs
  - A secure key in crypto: 128-256 bits
  - Number of Walmart transactions per day: 10^6.
  - URLs searched by Google in 2012: 3×10^{12}.



Determining formula type



- How long does it take to check if a formula is satisfiable?
  - Using a truth table: in time $O(m * 2^n)$ on a length *m*, *n*-variable formula.
  - Is it efficient?
    - Not really!
    - Formula with 100 variables is already too big!
    - In software verification: millions of variables!
  - Can we do better?

A million-dollar question!