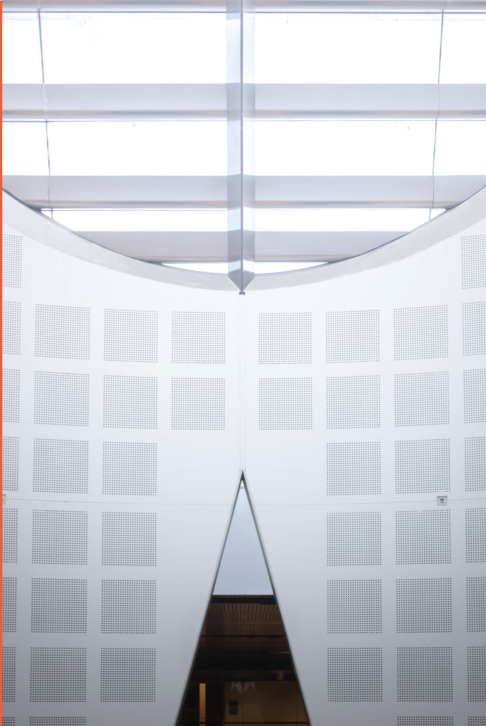
COMP2022|2922 Models of Computation

Propositional Logic

Sasha Rubin

September 15, 2022

THE UNIVERSITY OF SYDNEY



Agenda

1. Syntax
2. Semantics
3. Validity and Satisfiability

Why study logic?

Logics are used for representing, computing and reasoning.

- Electronics: design and simplification of digital circuits
- Databases: basis for query languages (SQL)
- Programming languages: DATALOG, PROLOG
- Algorithms: P vs NP
- AI: Knowledge-bases, explainable AI, specifications of agent goals and environments (planning in AI)
- Foundations of mathematics: provability
- Programming: formally reasoning about the correctness of programs.
- Automated reasoning: automated theorem proving
- Reasoning: as an aid to identify valid arguments

Which logics do we study?

1. Propositional logic

   about *propositions*, e.g., statements such as conditionals in programming, digital circuits.

2. Predicate logic (aka First-order logic)

   about *relations and functions*, e.g., the correctness of programs, knowledge-bases in AI, declarative programs, database queries.

There are also logics of *time*, *knowledge*, *belief* ...

What makes up a logic?

1. Syntax tells us the structure of expressions, or the rules for putting symbols together to form an expression.
2. Semantics refers to the meaning of expressions, or how they are evaluated.
3. Deduction is a syntactic mechanism for deriving new true expressions from existing true expressions.

Logic is like algebra!

- Algebra is for expressing and reasoning about arithmetic.
- Propositional Logic is for expressing and reasoning about propositions (aka statements).

Name	Propositional Logic	Algebra
variables	$p, q, r \ldots$	$x, y, z, \ldots$
values	$0, 1$	$1, 2, 3, \ldots$
operations	$\vee, \wedge, \neg, \ldots$	$+, -, \times, \ldots$
expressions	$\neg(p \wedge q)$	$1 + (x - y)$
equivalences	$p \wedge q \equiv q \wedge p$	$x + y = y + x$
evaluation	$\neg(p \wedge q) = 0$ if $p, q = 1$	$x \times y = 3$ if $x = 1, y = 3$
deduction	$p \wedge q$ true implies p true	$x = 2$ implies $x \times x = 4$

Propositional logic

- You can write propositional logic formulas in most programming languages.
- In `Python` and `Java` these are called Boolean expressions
- They are usually used as conditions for control flow, e.g., inside an `if` or a `while`.

Self test

Do the following two `Java` Boolean expressions say the same thing about integer variables x and y?

1. `!(x >= 5 && x != y)`
2. `(x < 5 || x == y)`

Vote now! (on mentimeter)

Propositions

A proposition is a sentence that declares a fact that is either true, or false, but not both.

It is not always easy to tell if a sentence is a proposition.

Which of the following are propositions?

1. The sun is hot.
2. The moon is made of cheese.
3. `2 < 5`
4. if $x = 7$ then $x < 5$.
5. Are you happy?
6. x is an integer.

Vote now! (on mentimeter)

Agenda

1. Syntax
2. Semantics
3. Validity and Satisfiability

Syntax

- Java and Python each have their own syntax for writing propositional formulas.
- We will use another syntax which is the standard in computer science and mathematics.

Name	Prop. Logic	Python	Java
conjunction	$\wedge$	`and`	`&&`
disjunction	$\vee$	`or`	`||`
negation	$\neg$	`not`	`!`
implication	$\rightarrow$		
bi-implication	$\leftrightarrow$	`==`	`==`
top/verum	$\top$	`True`	`true`
bottom/falsum	$\bot$	`False`	`false`
atoms	$p, q, r, \ldots$	Boolean variables	Boolean variables
formulas	$F, G, H, \ldots$	Boolean expressions	Boolean expressions

Propositional Logic: Syntax

We now precisely define the syntactic rules for definining formulas of Propositional Logic.

Definition

An atom is a variable of the form $p_1, p_2, p_3, \ldots, p, q, r, \ldots$
A formula is defined by the following recursive process:

F1. Every atom is a formula

F2. If F is a formula then $\neg F$ is a formula.

F3. If F, G are formulas then so are $(F \vee G)$ and $(F \wedge G)$.

Reading guide:

$\neg$ is the negation symbol, read ”it is not the case that”

$\wedge$ is the conjunction symbol, read "and"

$\vee$ is the disjunction symbol, read "or"

Recursive definition

This is another example of a recursive definition.

- The base case specifies the simplest objects, and the recursive cases specify how to build complex objects from ones we already built.
- Think of the definition as a construction manual!
- Let's build $(p \wedge (q \vee \neg p))$ using this definition (bottom-up).[1]

[1] We can also show this in a top-down manner.

Recursive definition

Note. We can also represent a formula as a tree.

Example $(p \wedge (q \vee \neg p))$

Self test

Which of the following are formulas according to the syntax before?

Vote now! (on mentimeter)

1. $(\neg p)$
2. $\neg\neg\neg p$
3. $\neg p \neg q$
4. $(r \wedge \neg(p \vee q))$

Propositional Logic: Syntax

Why do we need this mathematical definition of formula?

1. It specifies what we mean by a propositional formula.
   - If we disagree on whether something is a formula, we just consult the definition
2. It allows one to design algorithms that process/manipulate formulas using recursion as follows:
   - The base case is rule 1 of the definition
   - The recursive case are rules 2 and 3 of the definition
3. One can prove things about propositional formulas and about code that processes formulas.

Propositional Logic: Syntax

Tutorial problem

A formula which occurs in another formula is called a subformula. Give a recursive process that defines the set **SubForms**(G) of all subformulas of G.

Propositional Logic: Syntax

We use abbreviations:

- $(\bigvee_{i=1}^{n} F_i)$ instead of $(\ldots((F_1 \vee F_2) \vee F_3) \cdots \vee F_n)$
- $(\bigwedge_{i=1}^{n} F_i)$ instead of $(\ldots((F_1 \wedge F_2) \wedge F_3) \cdots \wedge F_n)$
- We may drop outermost parentheses.

E.g., we may write $(p \vee q \vee \neg r) \wedge r$ instead of $(((p \vee q) \vee \neg r) \wedge r)$.

Reading logical formulas

It is not always grammatically correct or elegant to read formulas by simply inserting the names of the connectives.

- We may read $p \wedge q$ as "p and q" and $\neg p$ as "not p".
- But if we know that p stands for "The earth is flat" and q stands for "The earth is round" ...
- then we may read $p \wedge q$ as "The earth is flat and the earth is round", or even as "The earth is flat and round".
- and we don't read $\neg p$ as "not the earth is flat" in English, but rather "it is not the case that the earth is flat" or even "the earth is not flat".

Agenda

1. Syntax
2. Semantics
3. Validity and Satisfiability

Semantics: truth values

Recall: Semantics refers to how you derive the value of a formula based on the values of its atomic subformulas.

- The elements of the set $\{0, 1\}$ are called truth values
- We read 1 as "true", and 0 as "false"
- After we assign truth values to atoms we can give truth values to formulas.
- Because of the recursive structure of formulas, we only need to give the rules for each connective ($\wedge, \vee, \neg$)

Here are the rules for doing this . . .

Semantics: truth tables

The formula $\neg F$ is true if and only if the formula F is false.

F	$\neg F$
0	1
1	0

Semantics: truth tables

The formula $(F \wedge G)$ is true if and only if both the formulas F and G are true.

F	G	$(F \wedge G)$
0	0	0
0	1	0
1	0	0
1	1	1

Semantics: truth tables

The formula $(F \vee G)$ is true if and only if either F or G or both are true.

F	G	$(F \vee G)$
0	0	0
0	1	1
1	0	1
1	1	1

Semantics

Example

Evaluate the formula $F = \neg((p \wedge q) \vee r)$ under the assignment $p = 1, q = 1, r = 0$.

p	q	r	$\neg((p \wedge q) \vee r)$
1	1	0	

Semantics

- An assignment is a function from atoms to truth values.
- On three variables p, q, r there are $2^3 = 8$ assignments.

Exercise: Evaluate the formula $F = \neg((p \wedge q) \vee r)$ under all assignments of the atoms p, q, r.

p	q	r	$\neg((p \wedge q) \vee r)$
0	0	0	
0	0	1	
0	1	0	
0	1	1	
1	0	0	
1	0	1	
1	1	0	
1	1	1	

Semantics: definition

– An assignment is a function α from atoms to truth values.

– The truth value of a formula under α is defined by the following recursive process:

TV1. $\mathrm{tv}(p, \alpha) = \alpha(p)$ for every atom p

TV2. $$\mathrm{tv}(\neg F, \alpha) = \begin{cases} 0 & \text{if } \mathrm{tv}(F, \alpha) = 1 \\ 1 & \text{if } \mathrm{tv}(F, \alpha) = 0 \end{cases}$$

TV3. $$\mathrm{tv}(F \wedge G, \alpha) = \begin{cases} 1 & \text{if } \mathrm{tv}(F, \alpha) = 1 \text{ and } \mathrm{tv}(G, \alpha) = 1 \\ 0 & \text{otherwise} \end{cases}$$

TV4. $$\mathrm{tv}(F \vee G, \alpha) = \begin{cases} 1 & \text{if } \mathrm{tv}(F, \alpha) = 1 \text{ or } \mathrm{tv}(G, \alpha) = 1 \\ 0 & \text{otherwise} \end{cases}$$

Self-test

For which assignments does $\mathrm{tv}((p \wedge (q \vee \neg p)), \alpha)$ equal 1?

Vote now! (on mentimeter)

1. $p = 0, q = 0.$
2. $p = 0, q = 1.$
3. $p = 1, q = 0.$
4. $p = 1, q = 1.$

Semantics

Why do we need this precise definition of semantics?

1. It specifies how to evaluate formulas.
   - If we disagree on the truth value of F under α, we just consult the definition.
2. This definition is implemented by the runtime environment of `Python` and `Java` to compute the values of Boolean expressions.
3. One can prove things about formulas or about code that processes formulas.

Semantics: Terminology

- We sometimes shorten $\mathrm{tv}(F, \alpha)$ and simply write $\alpha(F)$.
  e.g., $\mathrm{tv}(F \wedge G, \alpha) = 1$ may be written $\alpha(F \wedge G) = 1$.
- In case $\alpha(F) = 1$, we say any of the following:
  - α makes F true
  - α satisfies F
  - α models F, which is written $\alpha \models F$.
- The symbol $\models$ is called the “double-turnstile”.

What about other logical symbols?

Many times it is convenient to use an extended syntax:

1. $\top$ and $\bot$ are formulas.
   - These symbols are called the propositional constants.
   - $\top$ is read top and $\bot$ is read bottom
2. if F, G are formulas then so are $(F \rightarrow G)$ and $(F \leftrightarrow G)$.

   $\rightarrow$ is called the conditional (aka implication)

   $\leftrightarrow$ is called the bi-conditional (aka bi-implication)

Here are the semantics given informally:

1. the formula $\top$ is always true, and the formula $\bot$ is always false.
2. The formula $(F \rightarrow G)$ is false if and only if F is true and G is false.
3. The formula $(F \leftrightarrow G)$ is true if and only if F and G have the same truth values.

Exercise

Convince yourself that here is an equivalent (and more succinct) way to define the truth value of formulas under an assignment α, also for the extended syntax:

TV1. $\mathrm{tv}(p, \alpha) = \alpha(p)$ for atoms p.

TV2. $\mathrm{tv}(\neg F, \alpha) = 1 - \mathrm{tv}(F, \alpha)$.

TV3. $\mathrm{tv}(F \wedge G, \alpha) = \min\{\mathrm{tv}(F, \alpha), \mathrm{tv}(G, \alpha)\}$.

TV4. $\mathrm{tv}(F \vee G, \alpha) = \max\{\mathrm{tv}(F, \alpha), \mathrm{tv}(G, \alpha)\}$.

TV5. $\mathrm{tv}(F \to G, \alpha) = \mathrm{tv}(\neg F \vee G, \alpha)$.

TV6. $\mathrm{tv}(F \leftrightarrow G, \alpha) = \mathrm{tv}((F \to G) \wedge (G \to F), \alpha)$.

TV7. $\mathrm{tv}(\top, \alpha) = 1$, $\mathrm{tv}(\bot, \alpha) = 0$.

Semantics: conditional

F	G	$(F \to G)$
0	0	1
0	1	1
1	0	0
1	1	1

Semantics: conditional

- We now discuss why $(F \rightarrow G)$ has the same truth table as $(\neg F \vee G)$.
- When is the formula $(F \rightarrow G)$ true?
- It is true whenever, if F is true, then also G is true.
- So, if F is not true, then it doesn't matter whether G is true or not, the formula $(F \rightarrow G)$ will still be true.

F	G	$(F \rightarrow G)$
0	0	1
0	1	1
1	0	0
1	1	1

Note.

- Conditional is not the same as the programming construct "If condition is true then do instruction".
- Conditional does not mean that F causes G to be true.
- It might be useful to think of the formula $F \rightarrow G$ to mean that if F is true then I promise to make G true ...

Self test

Suppose I promise you that:

"if I am elected, then I will lower taxes".

Under what conditions do I break my promise?

Vote now! (on mentimeter)

1. I am elected and I lower taxes
2. I am elected and I do not lower taxes
3. I am not elected and I lower taxes
4. I am not elected and I do not lower taxes

Self test

- A robot has two sensor variables:

  `raining` and `carrying_umbrella`.
- Your job is to assign a truth value to the Boolean variable

  `ready`

  that says whether or not the robot can go outside and not get wet.

Vote now! (on mentimeter)

1. `ready = raining and carrying_umbrella`
2. `ready = not raining or carrying_umbrella`

Connection with natural language

What is the connection between propositional logic and natural language? Although logic can be used to model propositions of natural language, there are important differences.

- In logic, semantics is determined by the assignment and the syntactic structure of the formula.
- In natural language, the semantics also depends on the context.

Connection with natural language

Be careful formalising English statements into logic!

1. "or" in English ...
   - Sometimes it is inclusive (“I will sing or I will dance”), and so we would write $p \vee q$
   - Sometimes it is exclusive ("I win or I die"), and so we would write $(p \vee q) \wedge \neg(p \wedge q)$,
   - and sometimes it is unclear ("your error is in the program or the data").
2. "and" in English ...
   - "He threw the stone and the window broke" means something quite different in common English to "The window broke and he threw the stone".
   - But in formal logic, they have the same meaning.

Agenda

1. Syntax
2. Semantics
3. Validity and Satisfiability

Valid formulas

Definition
A formula F is valid if every assignment satisfies F.

i.e., "if its truth-table always has value 1".

Examples.

1. $p \vee \neg p$ is valid.
2. $p \vee \neg q$ is not valid

Valid formulas represent logical laws. . . in the same way that $x + 0 = x$ is an arithmetic law.

Satisfiable formulas

Definition
A formula F is satisfiable if at least one assignment satisfies F.
i.e., "if its truth-table has at least one 1".

Examples.

1. $p \vee \neg q$ is satisfiable.
2. $p \wedge \neg p$ is not satisfiable.

Why is satisfiability interesting?

Satisfiability problem

The satisfiability problem is to decide if a given propositional formula F is satisfiable.

This can be solved, in principle, by checking all the rows of the truth-table for F.

Satisfiability problem

Question. How efficient is this truth-table test for satisfiability?

- Suppose F has n atoms.
- Then the truth-table has 2^n rows.
- So this algorithm runs in worst-case exponential time (in the size of F).
- Suppose you can generate a table at the rate of one row per second.
- If $n = 80$, you will need 2^{80} seconds to write out the table.
- This is about 2.8 million times the age of the universe.
- To think about: what if you could generate, e.g., a billion rows per second?

Question. Is there a substantially faster method? Polynomial time in the size of F?

Satisfiability problem

The satisfiability problem is extremely important! Why?

- If the satisfiability problem is in **P** then **P** = **NP**!
  - In fact, there are many problems like this, not only in logic. See `COMP3027:Algorithm Design`.
- Many many problems can be efficiently reduced to the satisfiability problem.
- There are mature tools for solving the satisfiability problem in practice, called SAT solvers.