Propositional Logic

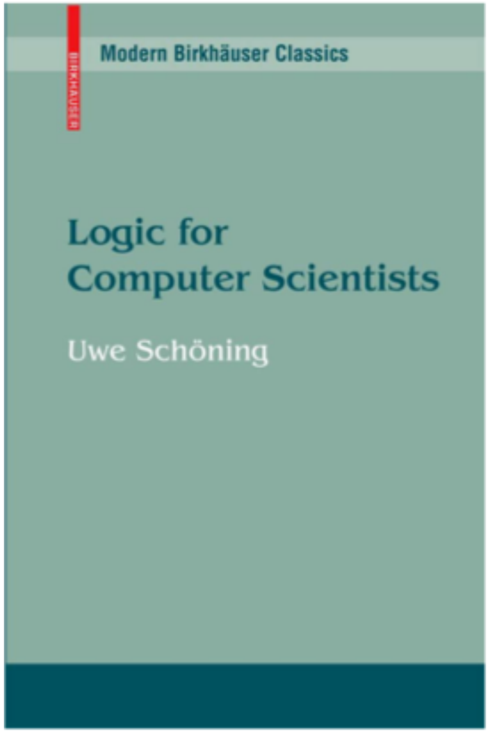
Testing, Quality Assurance, and Maintenance
Winter 2018

Prof. Arie Gurfinkel

UNIVERSITY OF WATERLOO

References

- Chpater 1 of Logic for Computer Scientists
  http://www.springerlink.com/content/978-0-8176-4762-9/



What is Logic

According to Merriam-Webster dictionary logic is:

a (1) : a science that deals with the principles and criteria of validity of inference and demonstration

d :the arrangement of circuit elements (as in a computer) needed for computation; *also*: the circuits themselves

What is Formal Logic

Formal Logic consists of

- syntax – what is a legal sentence in the logic
- semantics – what is the meaning of a sentence in the logic
- proof theory – formal (syntactic) procedure to construct valid/true sentences

Formal logic provides

- a language to precisely express knowledge, requirements, facts
- a formal way to reason about consequences of given facts rigorously

Propositional Logic (or Boolean Logic)

Explores simple grammatical connections such as *and*, *or*, and *not* between simplest "atomic sentences"

A = "Paris is the capital of France"

B = "mice chase elephants"

The subject of propositional logic is to declare formally the truth of complex structures from the truth of individual atomic components

A and B

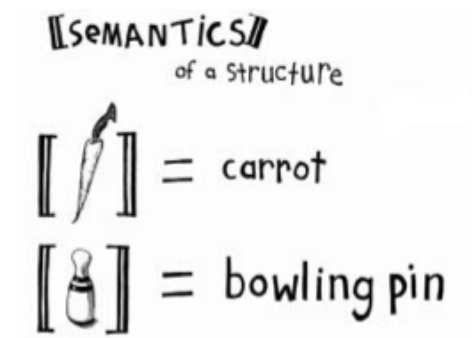
A or B

if A then B

Syntax and Semantics

Syntax

- MW: the way in which linguistic elements (such as words) are put together to form constituents (such as phrases or clauses)
- Determines and restricts how things are written



Semantics

- MW: the study of meanings
- Determines how syntax is interpreted to give meaning

Syntax of Propositional Logic

An *atomic formula* has a form A_i , where i = 1, 2, 3 …

Formulas are defined inductively as follows:

- All atomic formulas are formulas
- For every formula F, ¬F (called not F) is a formula
- For all formulas F and G, F ∧ G (called and) and F ∨ G (called or) are formulas

Abbreviations

- use A, B, C, … instead of A_1, A_2, …
- use $F_1 \rightarrow F_2$ instead of $\neg F_1 \vee F_2$ (implication)
- use $F_1 \leftrightarrow F_2$ instead of $(F_1 \rightarrow F_2) \wedge (F_2 \rightarrow F_1)$ (iff)

Syntax of Propositional Logic (PL)

$$
\begin{aligned}
\text{truth\_symbol} &::= \top(\text{true}) \mid \bot(\text{false}) \\
\text{variable} &::= p, q, r, \ldots \\
\text{atom} &::= \text{truth\_symbol} \mid \text{variable} \\
\text{literal} &::= \text{atom} | \neg\text{atom} \\
\text{formula} &::= \text{literal} \mid \\
&\quad \neg\text{formula} \mid \\
&\quad \text{formula} \land \text{formula} \mid \\
&\quad \text{formula} \lor \text{formula} \mid \\
&\quad \text{formula} \rightarrow \text{formula} \mid \\
&\quad \text{formula} \leftrightarrow \text{formula}
\end{aligned}
$$

UNIVERSITY OF WATERLOO

Example

$$F = \neg((A_5 \wedge A_6) \vee \neg A_3)$$

Sub-formulas are

$$\begin{aligned} F, ((A_5 \wedge A_6) \vee \neg A_3), \\ A_5 \wedge A_6, \neg A_3, \\ A_5, A_6, A_3 \end{aligned}$$

UNIVERSITY OF WATERLOO

Semantics of propositional logic

For an atomic formula A_i in **D**: **A'**(A_i) = **A**(A_i)

A'((F ∧ G))	= 1	if **A'**(F) = 1 and A'(G) = 1
	= 0	otherwise
A'((F ∨ G))	= 1	if **A'**(F) = 1 or A'(G) = 1
	= 0	otherwise
A'(¬F)	= 1	if **A'**(F) = 0
	= 0	otherwise

Example

$$F = \neg(A \wedge B) \vee C$$

$$\mathcal{A}(A) = 1$$
$$\mathcal{A}(B) = 1$$
$$\mathcal{A}(C) = 0$$

Truth Tables for Basic Operators

$\mathcal{A}(F)$	$\mathcal{A}(G)$	$\mathcal{A}((F \wedge G))$
0	0	0
0	1	0
1	0	0
1	1	1

$\mathcal{A}(F)$	$\mathcal{A}(\neg F)$
0	1
1	0

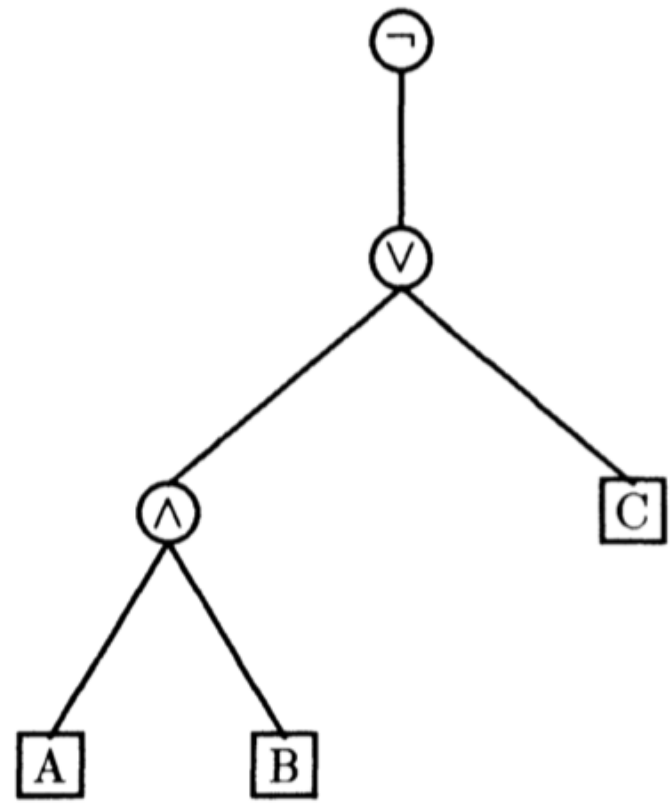
$\mathcal{A}(F)$	$\mathcal{A}(G)$	$\mathcal{A}((F \vee G))$
0	0	0
0	1	1
1	0	1
1	1	1

$$F = \neg(A \wedge B) \vee C$$

$$\mathcal{A}(A) = 1$$

$$\mathcal{A}(B) = 1$$

$$\mathcal{A}(C) = 0$$



Propositional Logic: Semantics

An assignment A is *suitable* for a formula F if A assigns a truth value to every atomic proposition of F

An assignment A is a *model* for F, written $A \models F$, iff

- A is suitable for F
- A(F) = 1, i.e., F *holds* under A

A formula F is *satisfiable* iff F has a model, otherwise F is *unsatisfiable* (or contradictory)

A formula F is *valid* (or a tautology), written $\models F$, iff every suitable assignment for F is a model for F

UNIVERSITY OF WATERLOO

Determining Satisfiability via a Truth Table

A formula F with n atomic sub-formulas has 2^n suitable assignments

Build a truth table enumerating all assignments

F is satisfiable iff there is at least one entry with 1 in the output

	A_1	A_2	$\cdots$	A_{n-1}	A_n	F
$\mathcal{A}_1$:	0	0		0	0	$\mathcal{A}_1(F)$
$\mathcal{A}_2$:	0	0		0	1	$\mathcal{A}_2(F)$
$\vdots$			$\ddots$			$\vdots$
$\mathcal{A}_{2^n}$:	1	1		1	1	$\mathcal{A}_{2^n}(F)$

UNIVERSITY OF WATERLOO

An example

$$F = (\neg A \to (A \to B))$$

A	B	$\neg A$	$(A \to B)$	F
0	0	1	1	1
0	1	1	1	1
1	0	0	0	1
1	1	0	1	1

Validity and Unsatisfiability

Theorem:

A formula F is valid if and only if ¬F is unsatifsiable

Proof:

F is valid ⇔ every suitable assignment for F is a model for F

⇔ every suitable assignment for ¬ F is not a model for ¬ F

⇔ ¬ F does not have a model

⇔ ¬ F is unsatisfiable

Normal Forms: CNF and DNF

A *literal* is either an atomic proposition v or its negation ~v

A *clause* is a disjunction of literals

- e.g., (v1 || ~v2 || v3)

A formula is in *Conjunctive Normal Form* (CNF) if it is a conjunction of disjunctions of literals (i.e., a conjunction of clauses):

- e.g., (v1 || ~v2) && (v3 || v2)

$$\bigwedge_{i=1}^{n} (\bigvee_{j=1}^{m_i} L_{i,j})$$

A formula is in *Disjunctive Normal Form* (DNF) if it is a disjuction of conjunctions of literals

$$\bigvee_{i=1}^{n} (\bigwedge_{j=1}^{m_i} L_{i,j})$$

UNIVERSITY OF WATERLOO

From Truth Table to CNF and DNF

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

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

A	B	C	F
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	0
1	0	0	1
1	0	1	1
1	1	0	0
1	1	1	0

Normal Form Theorem

Theorem: For every formula F, there is an equivalent formula F_1 in CNF and F_2 in DNF

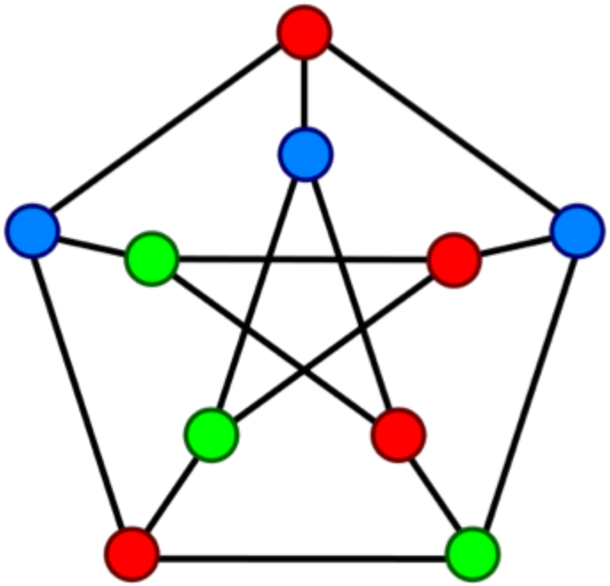
Proof: (by induction on the structure of the formula F)

UNIVERSITY OF WATERLOO

ENCODING PROBLEMS INTO CNF-SAT

Graph k-Coloring

Given a graph *G = (V, E)*, and a natural number *k > 0* is it possible to assign colors to vertices of *G* such that no two adjacent vertices have the same color.



Formally:

- does there exists a function f : V → [0..k) such that
- for every edge (u, v) in E, f(u) != f(v)

Graph coloring for k > 2 is NP-complete

Problem: Encode k-coloring of G into CNF

- construct CNF *C* such that C is SAT iff G is k-colorable

https://en.wikipedia.org/wiki/Graph_coloring

k-coloring as CNF

Let a Boolean variable $f_{v,i}$ denote that vertex *v* has color *i*

- if $f_{v,i}$ is true if and only if f(v) = i

Every vertex has at least one color

$$\bigvee_{0 \le i < k} f_{v,i} \qquad (v \in V)$$

No vertex is assigned two colors

$$\bigwedge_{0 \le i < j < k} (\neg f_{v,i} \vee \neg f_{v,j}) \qquad (v \in V)$$

No two adjacent vertices have the same color

$$\bigwedge_{0 \le i < k} (\neg f_{v,i} \vee \neg f_{u,i}) \qquad ((v,u) \in E)$$

Propositional Resolution

Pivot

$$\frac{C \vee p \qquad\qquad D \vee \neg p}{C \vee D}$$

Resolvent

Res({C, p}, {D, !p}) = {C, D}

Given two clauses (C, p) and (D, !p) that contain a literal p of different polarity, create a new clause by taking the union of literals in C and D

Resolution Lemma

Lemma:

Let F be a CNF formula. Let R be a resolvent of two clauses X and Y in F. Then, $F \cup \{R\}$ is equivalent to F

Proof System

$$P_1, \ldots, P_n \vdash C$$

An inference rule is a tuple (P_1, …, P_n, C)

- where, P_1, …, P_n, C are formulas
- P_i are called premises and C is called a conclusion
- intuitively, the rules says that the conclusion is true if the premises are

A proof system P is a collection of inference rules

A proof in a proof system P is a tree (or a DAG) such that

- nodes are labeled by formulas
- for each node *n*, (parents(n), n) is an inference rule in P

Propositional Resolution

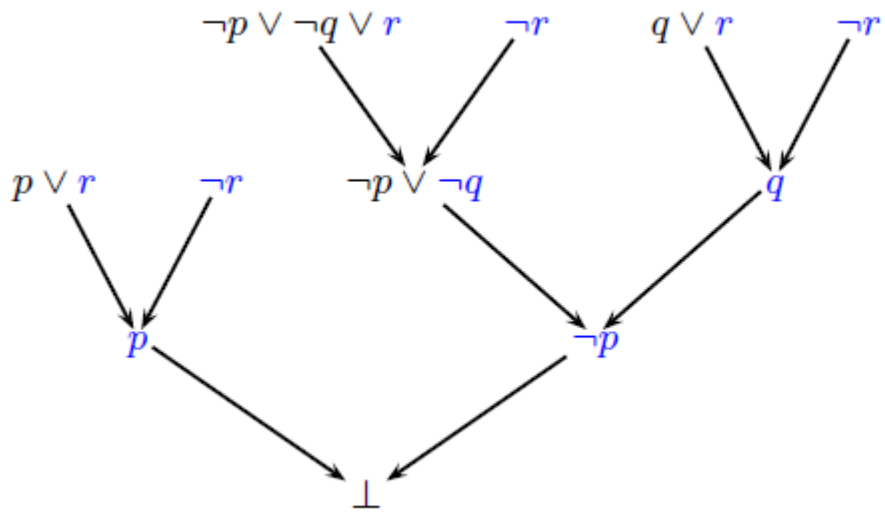
$$\frac{C \vee p \qquad\qquad D \vee \neg p}{C \vee D}$$

Propositional resolution is a sound inference rule

Proposition resolution system consists of a single propositional resolution rule

Example of a resolution proof

A refutation of $\neg p \vee \neg q \vee r$, $p \vee r$, $q \vee r$, $\neg r$:



UNIVERSITY OF WATERLOO

Resolution Proof Example

Show by resolution that the following CNF is UNSAT

$$\neg b \wedge (\neg a \vee b \vee \neg c) \wedge a \wedge (\neg a \vee c)$$

$$\dfrac{\dfrac{\dfrac{\neg a \vee b \vee \neg c \qquad a}{b \vee \neg c} \qquad b}{\neg c} \qquad \dfrac{a \qquad \neg a \vee c}{c}}{\bot}$$

Entailment and Derivation

A set of formulas F entails a set of formulas G iff every model of F and is a model of G

$$F \models G$$

A formula G is derivable from a formula F by a proof system P if there exists a proof whose leaves are labeled by formulas in F and the root is labeled by G

$$F \vdash_P G$$

UNIVERSITY OF WATERLOO

Soundness and Completeness

A proof system P is sound iff

$$(F \vdash_P G) \implies (F \models G)$$

A proof system P is complete iff

$$(F \models G) \implies (F \vdash_P G)$$

Completeness of Propositional Resolution

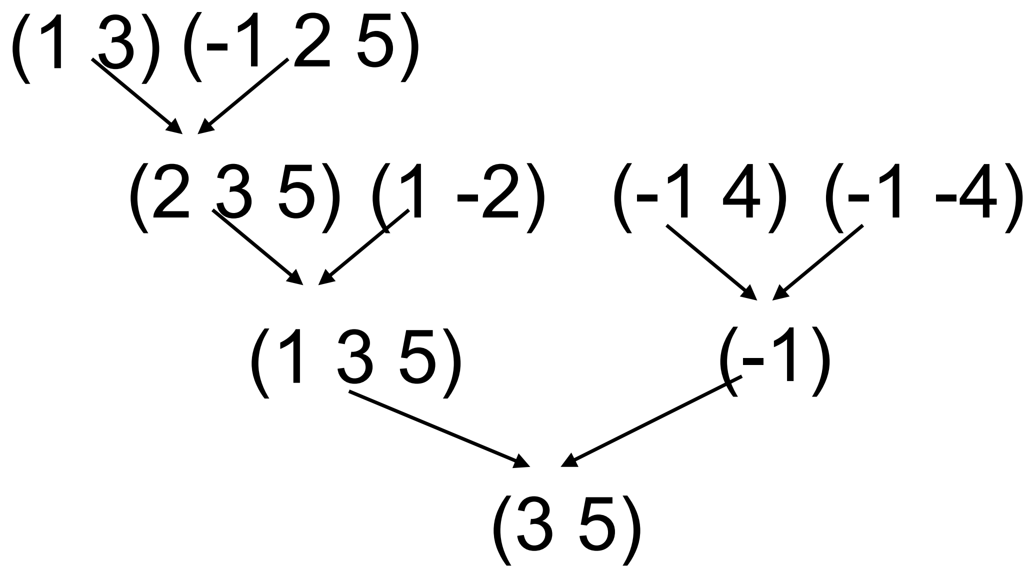
Theorem: Propositional resolution is sound and complete for propositional logic

Proof by resolution

Notation: positive numbers mean variables, negative mean negation

Let φ = (1 3) ∧ (-1 2 5) ∧ (-1 4) ∧ (-1 -4)

We'll try to prove φ → (3 5)



Resolution

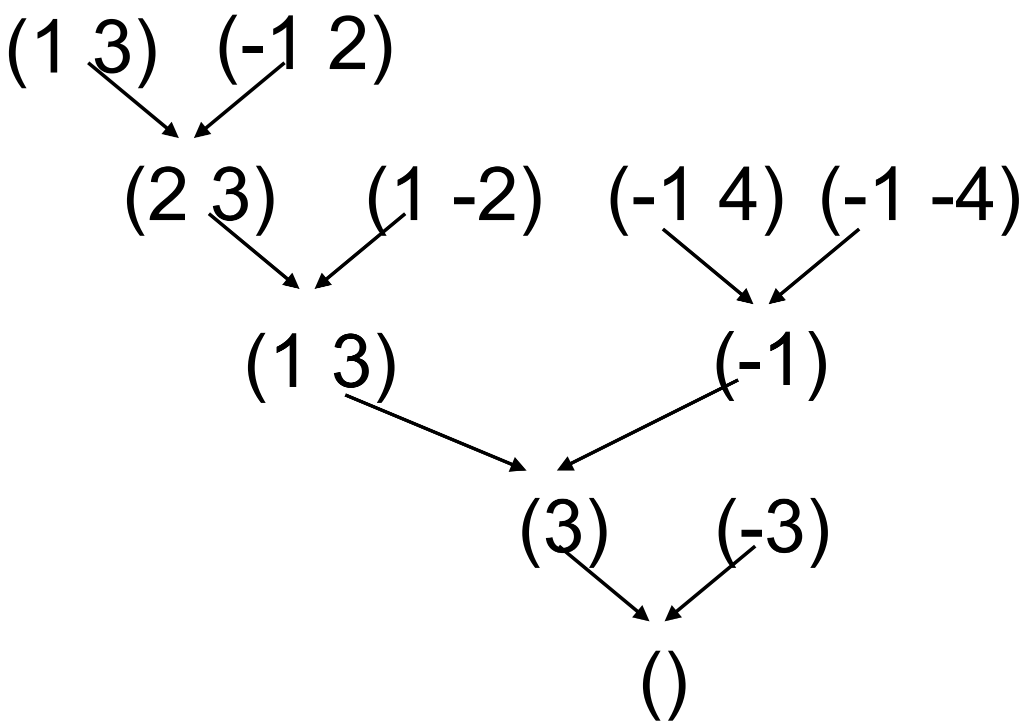
Resolution is a sound and complete inference system for CNF

If the input formula is unsatisfiable, there exists a proof of the empty clause

Example: UNSAT Derivation

Notation: positive numbers mean variables, negative mean negation

Let φ = (1 3) ∧ (-1 2) ∧ (-1 4) ∧ (-1 -4) ∧ (-3)



Logic for Computer Scientists: Ex. 33

Using resolution show that

$$A \wedge B \wedge C$$

is a consequence of

$$\neg A \vee B$$

$$\neg B \vee C$$

$$A \vee \neg C$$

$$A \vee B \vee C$$

Logic for Computer Scientists: Ex. 34

Show using resolution that F is valid

$$F = (\neg B \wedge \neg C \wedge D) \vee (\neg B \wedge \neg D) \vee (C \wedge D) \vee B$$

$$\neg F = (B \vee C \vee \neg D) \wedge (B \vee D) \wedge (\neg C \vee \neg D) \wedge \neg B$$