The Logics of Program Verification

Testing, Quality Assurance, and Maintenance
Fall 2023

Prof. Arie Gurfinkel

UNIVERSITY OF WATERLOO

```
method factorial (n: int) returns (v:int)
  requires n >= 0;
  ensures v = fact(n);
{
  v := 1;
  if (n <= 1) { return v; }
  var i := 2;
  while (i <= n)
    invariant i <= n + 1
    invariant v = fact(i - 1)
  {
    v := i * v;
    i := i + 1;
  }
  return v;
}
```

Specification

Inductive Invariant

Program Verification

How can we *argue* that a given program is correct

- i.e., satisfies its formal specifications?

Such an argument must combine

- Operational Semantics – to understand different programming constructs
- Propositional Reasoning – to break the problem into sub-goals that can be reasoned individually and combined later
- Mathematical Reasoning – properties of numbers, arithmetic, factorial, etc…
- Formal argument style – to mechanically check the flow of reasoning

All of this requires a **LOGIC**

- A formal language with well-defined semantics and strict reasoning rules

Three Logics of Program Verification

Program Verifier
(Dafny)

Hoare Logic
(logic of programs)

SMT Solver
(Z3)

First Order Logic
(logic of mathematical theories)

SAT Solver
(Z3)

Propositional Logic
(logic of Boolean circuits)

UNIVERSITY OF WATERLOO

Plan for the next few weeks

Week	Monday	Friday
Week 7 (Oct 30)	Propositional Logic (1)	Propositional Logic (2)
Week 8 (Nov 6)	First Order Logic (Part 1)	First Order Logic (Part 2)
Week 9 (Nov 13)	Hoare Logic (Part 1)	Hoare Logic (Part 2)

Understanding formal logic can be ~~boring~~ hard.
Don’t ignore suggested reading material!!!

Propositional Logic

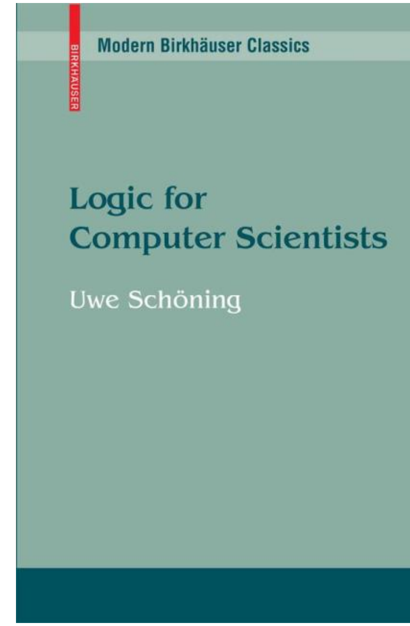
Testing, Quality Assurance, and Maintenance
Fall 2023

Prof. Arie Gurfinkel

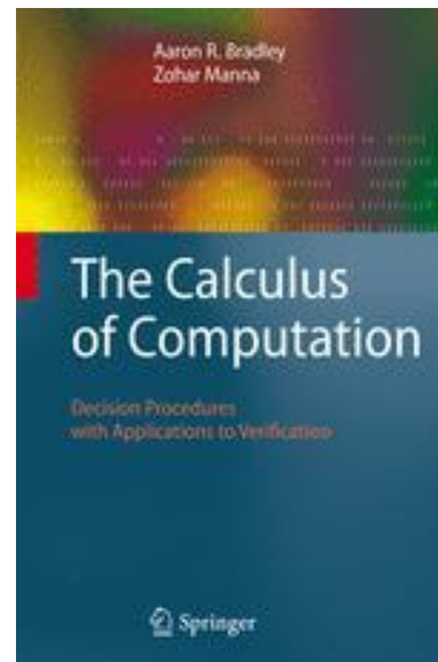
UNIVERSITY OF WATERLOO

References

- Chpater 1 of Logic for Computer Scientists
  https://link.springer.com/book/10.1007/978-0-8176-4763-6



- Chapter 1 of Calculus of Computation
  https://link.springer.com/book/10.1007/978-3-540-74113-8



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

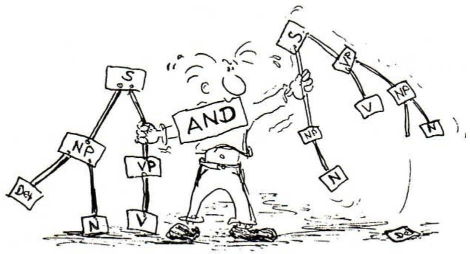
A and B

A or B

if A then B

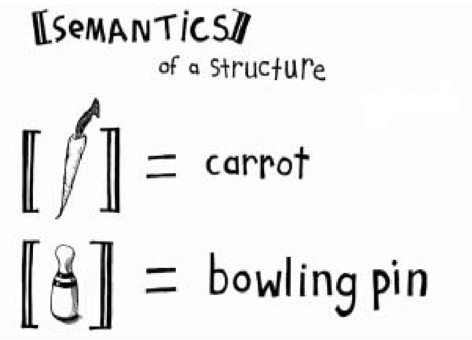
A and not A

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, $\neg F$ (called not F) is a formula
- For all formulas F and G, $F \wedge G$ (called and) and $F \vee G$ (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)

UNIVERSITY OF WATERLOO

Formal Syntax of Propositional Logic

```
constant    ::= true | false | 0 | 1 | ⊤ | ⊥

variable    ::= p | q | r | A | B | C | A₀ | A₁ | …

atom        ::= constant | variable

literal     ::= atom | ¬ atom

formula     ::= literal             |
                ¬ formula           |
                formula ∧ formula |
                formula ∨ formula
```

Example

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

Sub-formulas are

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

Semantics of propositional logic 1/2

Start with two truth values: {0, 1}

- 0 stands for false, and 1 stands for true

Let **D** be any subset of the *atomic* formulas

An *assignment* **A** is a map **D** → {0, 1}

- **A** assigns true/false to every atomic in **D**

Let **E** ⊇ **D** be a set of formulas built from **D** using propositional connectives

Extended assignment **A**': **E** → {0, 1} extends **A** from atomic formulas to all formulas

UNIVERSITY OF WATERLOO

Semantics of propositional logic 2/2

For an atomic formula A_i in **D**: $\mathbf{A'}(A_i) = \mathbf{A}(A_i)$

$\mathbf{A'}(F \wedge G)$ = 1 if $\mathbf{A'}(F) = 1$ and $A'(G) = 1$
= 0 otherwise

$\mathbf{A'}(F \vee G)$ = 1 if $\mathbf{A'}(F) = 1$ or $A'(G) = 1$
= 0 otherwise

$\mathbf{A'}(\neg F)$ = 1 if $\mathbf{A'}(F) = 0$
= 0 otherwise

Exercise: Define Extended Assignment

$$F = \neg((A \land B) \lor C)$$

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

Is F true or false under A'?

Truth Tables for Basic Operators

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

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

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

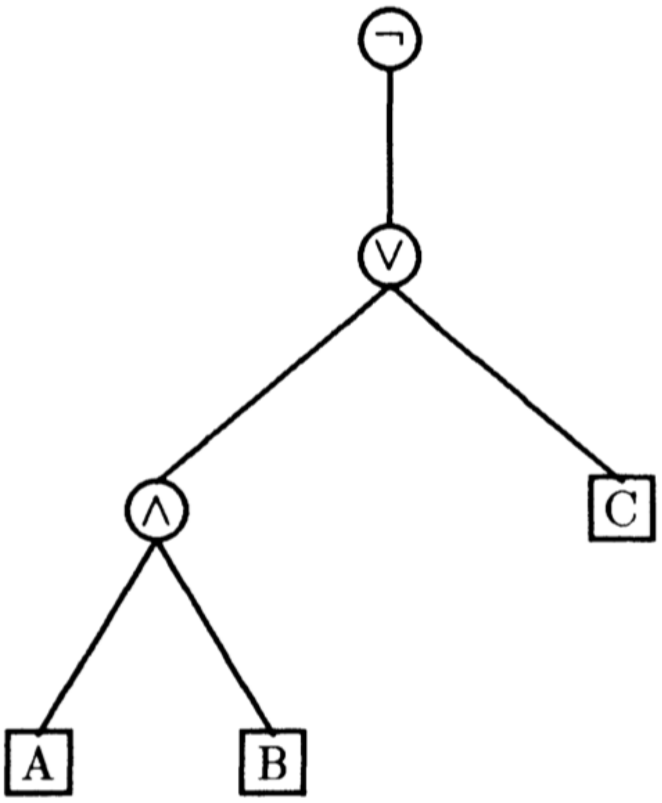
Formula

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

Assignment

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

Abstract Syntax Tree (AST)



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⊨ 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 ⊨ F, iff every suitable assignment for F is a model for F

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	$\mathcal{A}_{2^n}(F)$

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

Semantic Equivalence

Two formulas F and G are *(semantically) equivalent*, written **F ≡ G**, iff for every assignment A that is suitable for both F and G, A'(F) = A'(G)

For example, (F ∧ G) is equivalent to (G ∧ F)

Formulas with different atomic propositions can be equivalent

- e.g., all tautologies are equivalent to true
- e.g., all unsatisfiable formulas are equivalent to false

Substitution Theorem

Theorem: Let F and G be equivalent formulas. Let H be a formula in which F occurs as a sub-formula. Let H' be a formula obtained from H by replacing every occurrence of F by G. Then, H and H' are equivalent.

In symbols:

$$F \equiv G \qquad \Rightarrow \qquad H \equiv H[F \to G]$$

Proof:

(Let's talk about proof by induction first…)

UNIVERSITY OF WATERLOO

Structural Induction on the formula structure

The definition of a syntax of a formula is an *inductive* definition

- first, define atomic formulas; second, define more complex formulas from simple ones, each next definition uses previous definition recursively

The definition of the semantics of a formula is also inductive

- first, determine value of atomic propositions; second, define values of more complex formulas

The same principle works for proving properties of formulas!

- To show that every formula F satisfies some property S:
- (base case) show that S holds for atomic formulae
- (induction step) assume S holds for an arbitrary fixed formulas F and G. Show that S holds for $(F \wedge G)$, $(F \vee G)$, and $(\neg F)$

Substitution Theorem

Theorem: Let F and G be equivalent formulas. Let H be a formula in which F occurs as a sub-formula. Let H' be a formula obtained from H by replacing every occurrence of F by G. Then, H and H' are equivalent.

Proof: by induction on formula structure

(base case) if H is atomic, then F = H, H' = G, and $F \equiv G$

(inductive step)

(case 1) $H = \neg H_1$

(case 2) $H = H_1 \wedge H_2$

(case 3) $H = H_1 \vee H_2$

UNIVERSITY OF WATERLOO

Useful Equivalences (1/ 2)

$$
\begin{aligned}
(F \wedge F) &\equiv F \\
(F \vee F) &\equiv F && \text{(Idempotency)} \\[1ex]
(F \wedge G) &\equiv (G \wedge F) \\
(F \vee G) &\equiv (G \vee F) && \text{(Commutativity)} \\[1ex]
((F \wedge G) \wedge H) &\equiv (F \wedge (G \wedge H)) \\
((F \vee G) \vee H) &\equiv (F \vee (G \vee H)) && \text{(Associativity)} \\[1ex]
(F \wedge (F \vee G)) &\equiv F \\
(F \vee (F \wedge G)) &\equiv F && \text{(Absorption)} \\[1ex]
(F \wedge (G \vee H)) &\equiv ((F \wedge G) \vee (F \wedge H)) \\
(F \vee (G \wedge H)) &\equiv ((F \vee G) \wedge (F \vee H)) && \text{(Distributivity)} \\[1ex]
\neg\neg F &\equiv F && \text{(Double Negation)}
\end{aligned}
$$

Useful Equivalences (2/ 2)

$$
\begin{array}{rcll}
\neg(F \wedge G) & \equiv & (\neg F \vee \neg G) & \\
\neg(F \vee G) & \equiv & (\neg F \wedge \neg G) & \text{(deMorgan's Laws)} \\
 & & & \\
(F \vee G) & \equiv & F, \text{ if } F \text{ is a tautology} & \\
(F \wedge G) & \equiv & G, \text{ if } F \text{ is a tautology} & \text{(Tautology Laws)} \\
 & & & \\
(F \vee G) & \equiv & G, \text{ if } F \text{ is unsatisfiable} & \\
(F \wedge G) & \equiv & F, \text{ if } F \text{ is unsatisfiable} & \text{(Unsatisfiability Laws)}
\end{array}
$$

Don’t believe in these laws. Prove them …

UNIVERSITY OF WATERLOO

Bool: Exercise 18: Children and Doctors

Formalize and show that the two statements are equivalent

- If the child has temperature or has a bad cough and we reach the doctor, then we call him

$$((T \vee C) \wedge R) \Rightarrow D$$

- If the child has temperature, then we call the doctor, provided we reach him, and, if we reach the doctor then we call him, if the child has a bad cough

$$(R \Rightarrow (T \Rightarrow D)) \wedge (C \Rightarrow (R \Rightarrow D))$$

Equivalence proof

$((T \vee C) \wedge R) \Rightarrow D$

$((T \wedge R) \vee (C \wedge R) \Rightarrow D)$

$((T \wedge R) \Rightarrow D) \wedge ((C \wedge R) \Rightarrow D)$

$(R \Rightarrow (T \Rightarrow D)) \wedge (C \Rightarrow (R \Rightarrow D))$

Law:

$(a \vee b) \Rightarrow c$

$(a \Rightarrow c) \wedge (b \Rightarrow c)$

Merriam-Webster SINCE 1828

JOIN MWU | GAMES | BROWSE THESAURUS | WORD OF THE DAY | WORDS AT PLAY

normal form

DICTIONARY | THESAURUS

normal form noun

Definition of *normal form*

logic

: a canonical or standard fundamental form of a statement to which others can be reduced

especially **:** a compound statement in the propositional calculus consisting of nothing but a conjunction of disjunctions whose disjuncts are either elementary statements or negations thereof

https://www.merriam-webster.com/dictionary/normal%20form

Negation Normal Form (NNF)

A formula F is in Negation Normal Form (NNF) if every occurrence of negation (¬) in F is applied to an atomic sub-formula of F.

For example,

- $\neg a \vee \neg b \vee c$ is in NNF
- $\neg (a \wedge b \wedge \neg c)$ is NOT in NNF (why?)

Theorem (NNF): For every formula F there is a semantically equivalent form G in NNF. In symbols

- For every F, exists G in NNF, such that $F \equiv G$

UNIVERSITY OF WATERLOO

Normal Form: DNF

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

A *cube* is a conjunction of literals

- e.g., (v1 ∧ ¬ v2 ∧ v3)

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

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

(Fun) Fact: determining whether a DNF formula F is satisfiable is easy

- easy == linear in the size of the formula

Normal Form: CNF

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 F is in *Conjunctive Normal Form* (CNF) if F is a conjunction of disjunctions of literals

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

(Fun) Fact: determining whether a CNF formula F is satisfiable is hard

- hard == NP-complete

Normal Form Theorem

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

That is, CNF and DNF are normal forms:

- Every propositional formula can be converted to CNF and to DNF without affecting its meaning (i.e., semantics)!

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

Details are left as an exercise!

Example: From Truth Table to CNF and DNF

DNF

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

CNF

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

Truth table

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

see the book for detailed algorithm

From Truth Table to DNF / CNF

From a truth table T to DNF formula F

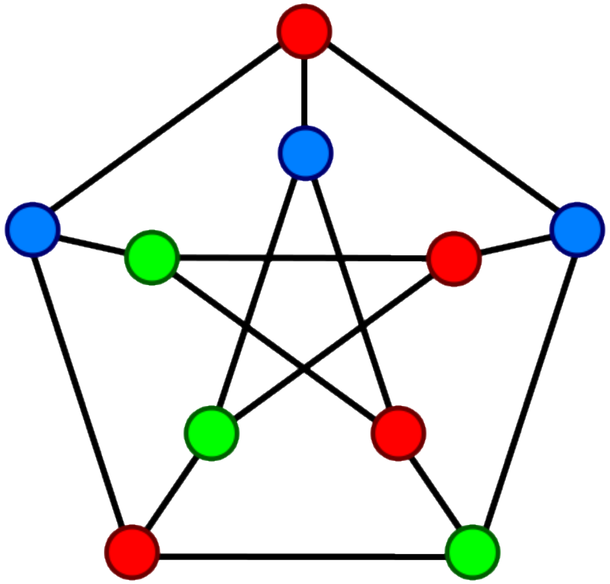
- For every row i of T with value 1, construct a conjunction F_i that characterizes the row. That is, F_i is a formula that is true **exactly** for the assignment of row I
- Let $F = \vee F_i$, then F is equivalent to T and is in DNF
- Fact: F has as many disjuncts as rows with value 1 in T
- Question: How big can F be? How small can F be?

From a truth table T to CNF formula G

- For every row i of T with value 0, construct a conjunction F_i that characterizes the row
- Let G_i be NNF of $\neg F_i$
- Let $G = \wedge G_i$, then G is equivalent to T and is in CNF
- Fact: F has as many disjuncts as rows with value 0 in T
- Question: How big can F be? How small can F be?

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 REASONING

Explaining Satisfiability and Unsatisfiability

Let F be a propositional formula (large)

Assume that F is satisfiable. What is a short proof / certificate to establish satisfiability without a doubt?

- provide a model. The model is linear in the size of the formula

Assume that F is unsatisfiable. What is a short proof / certificate to establish **UNSATISFIABILITY** without a doubt?

For example, is the following formula SAT or UNSAT? How do you explain your answer?

$$\neg b \land (\neg a \lor b \lor \neg c) \land a \land (\neg a \lor c)$$

UNIVERSITY OF WATERLOO

What is a proof?

Egoroff's theorem in Royden Fitzpatrick (comparison with lemma 10)

I find it little clear, but still I am unable to understand the proof of the lemma 10,

Egoroff's Theorem *Assume E has finite measure. Let $\{f_n\}$ be a sequence of measurable functions on E that converges pointwise on E to the real-valued function f. Then for each $\epsilon > 0$, there is a closed set F contained in E for which*

$$\{f_n\} \to f \text{ uniformly on } F \text{ and } m(E \sim F) < \epsilon.$$

Lemma 10 *Under the assumptions of Egoroff's Theorem, for each $\eta > 0$ and $\delta > 0$, there is a measurable subset A of E and an index N for which*

$$|f_n - f| < \eta \text{ on } A \text{ for all } n \geq N \text{ and } m(E \sim A) < \delta.$$

Proof For each k, the function $|f - f_k|$ is properly defined, since f is real-valued, and it is measurable, so that the set $\{x \in E \mid |f(x) - f_k(x)| < \eta\}$ is measurable. The intersection of a countable collection of measurable sets is measurable. Therefore

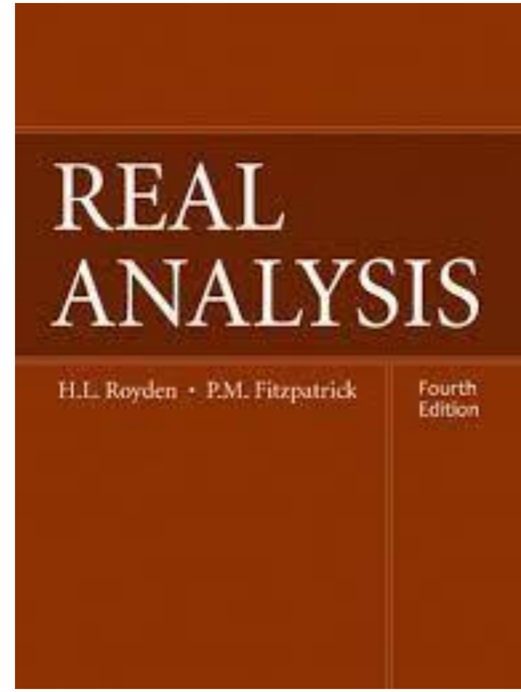
$$E_n = \{x \in E \mid |f(x) - f_k(x)| < \eta \text{ for all } k \geq n\} \quad ?$$

is a measurable set. Then $\{E_n\}_{n=1}^{\infty}$ is an ascending collection of measurable sets, and $E = \cup_{n=1}^{\infty} E_n$, since $\{f_n\}$ converges pointwise to f on E. We infer from the continuity of measure that

$$m(E) = \lim_{n \to \infty} m(E_n).$$

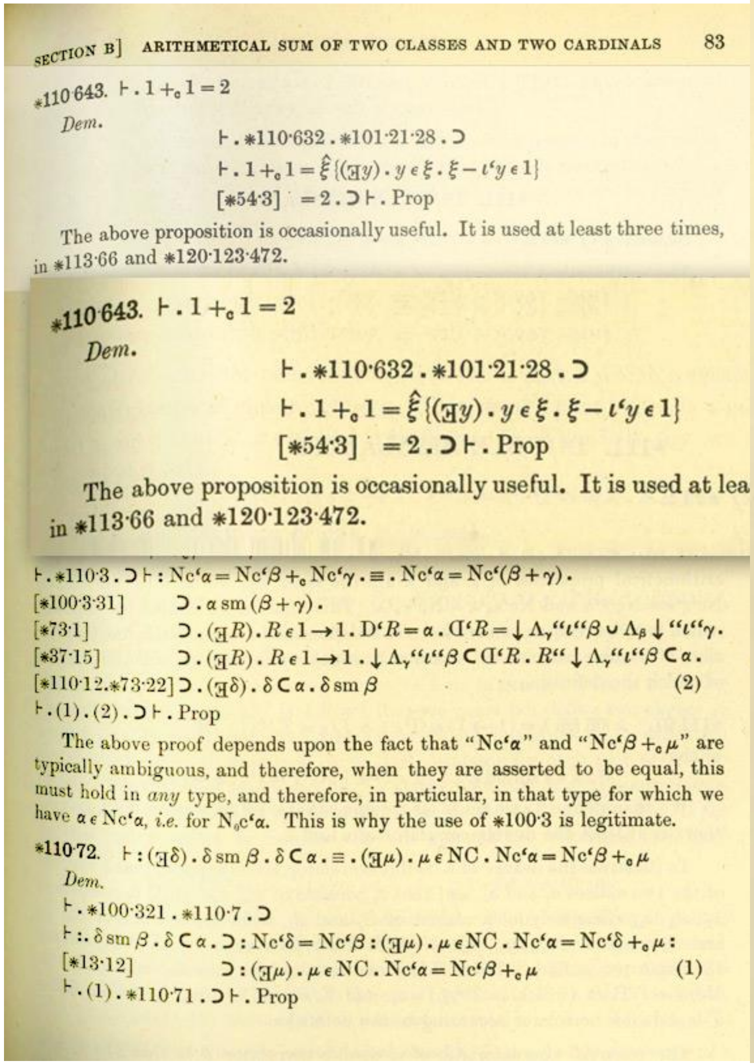
? Since $m(E) < \infty$, we may choose an index N for which $m(E_N) > m(E) - \epsilon$. Define $A = E_n$ and observe that, by the excision property of measure, $m(E \sim A) = m(E) - m(E_N) < \epsilon$. $\square$

Any help guys!



https://math.stackexchange.com/questions/2154189/proof-of-egoroffs-theorem-by-using-lemma-10-royden-real-analysis

What is a proof?



SECTION B] ARITHMETICAL SUM OF TWO CLASSES AND TWO CARDINALS 83

$*110{\cdot}643.\ \vdash . 1 +_c 1 = 2$

Dem.

$\vdash . *110{\cdot}632 . *101{\cdot}21{\cdot}28 . \supset$

$\vdash . 1 +_c 1 = \hat{\xi}\{(\exists y) . y \in \xi . \xi - \iota`y \in 1\}$

$[*54{\cdot}3] \quad = 2 . \supset \vdash . \text{Prop}$

The above proposition is occasionally useful. It is used at least three times, in $*113{\cdot}66$ and $*120{\cdot}123{\cdot}472$.

$*110{\cdot}643.\ \vdash . 1 +_c 1 = 2$

Dem.

$\vdash . *110{\cdot}632 . *101{\cdot}21{\cdot}28 . \supset$

$\vdash . 1 +_c 1 = \hat{\xi}\{(\exists y) . y \in \xi . \xi - \iota`y \in 1\}$

$[*54{\cdot}3] \quad = 2 . \supset \vdash . \text{Prop}$

The above proposition is occasionally useful. It is used at lea
in $*113{\cdot}66$ and $*120{\cdot}123{\cdot}472$.

$\vdash . *110{\cdot}3 . \supset \vdash : \text{Nc}`\alpha = \text{Nc}`\beta +_c \text{Nc}`\gamma . \equiv . \text{Nc}`\alpha = \text{Nc}`(\beta + \gamma) .$

$[*100{\cdot}3{\cdot}31] \quad \supset . \alpha \,\text{sm}\, (\beta + \gamma) .$

$[*73{\cdot}1] \quad \supset . (\exists R) . R \in 1 \to 1 . \text{D}`R = \alpha . \text{Cl}`R = \downarrow \Lambda_\gamma``\iota``\beta \cup \Lambda_\beta \downarrow ``\iota``\gamma .$

$[*37{\cdot}15] \quad \supset . (\exists R) . R \in 1 \to 1 . \downarrow \Lambda_\gamma``\iota``\beta \subset \text{Cl}`R . R`` \downarrow \Lambda_\gamma``\iota``\beta \subset \alpha .$

$[*110{\cdot}12 . *73{\cdot}22] \supset . (\exists \delta) . \delta \subset \alpha . \delta \,\text{sm}\, \beta \quad (2)$

$\vdash . (1) . (2) . \supset \vdash . \text{Prop}$

The above proof depends upon the fact that "$\text{Nc}`\alpha$" and "$\text{Nc}`\beta +_c \mu$" are typically ambiguous, and therefore, when they are asserted to be equal, this must hold in *any* type, and therefore, in particular, in that type for which we have $\alpha \in \text{Nc}`\alpha$, *i.e.* for $\text{N}_0\text{c}`\alpha$. This is why the use of $*100{\cdot}3$ is legitimate.

$*110{\cdot}72.\ \vdash : (\exists \delta) . \delta \,\text{sm}\, \beta . \delta \subset \alpha . \equiv . (\exists \mu) . \mu \in \text{NC} . \text{Nc}`\alpha = \text{Nc}`\beta +_c \mu$

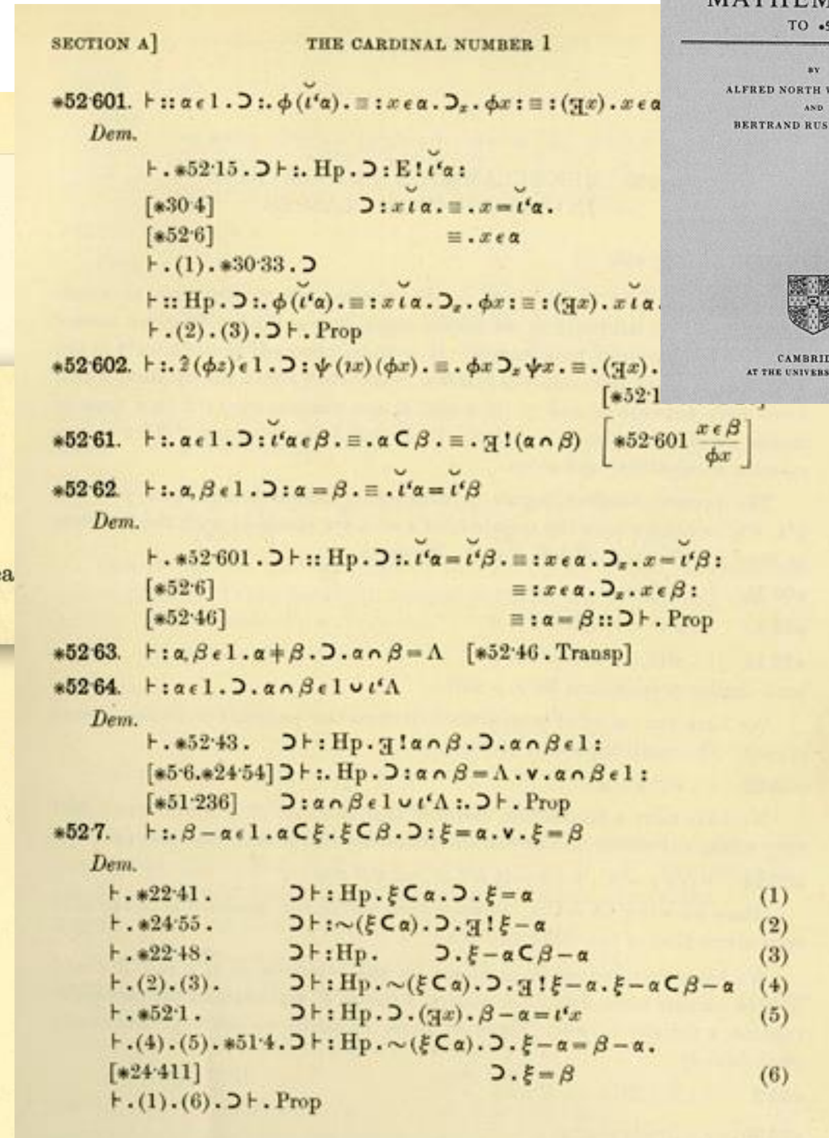
Dem.

$\vdash . *100{\cdot}321 . *110{\cdot}7 . \supset$

$\vdash :. \delta \,\text{sm}\, \beta . \delta \subset \alpha . \supset : \text{Nc}`\delta = \text{Nc}`\beta : (\exists \mu) . \mu \in \text{NC} . \text{Nc}`\alpha = \text{Nc}`\delta +_c \mu :$

$[*13{\cdot}12] \quad \supset : (\exists \mu) . \mu \in \text{NC} . \text{Nc}`\alpha = \text{Nc}`\beta +_c \mu \quad (1)$

$\vdash . (1) . *110{\cdot}71 . \supset \vdash . \text{Prop}$



SECTION A] THE CARDINAL NUMBER 1

$*52{\cdot}601.\ \vdash :: \alpha \in 1 . \supset :. \phi(\breve{\iota}`\alpha) . \equiv : x \in \alpha . \supset_x . \phi x : \equiv : (\exists x) . x \in \alpha$

Dem.

$\vdash . *52{\cdot}15 . \supset \vdash :. \text{Hp} . \supset : \text{E}!\,\breve{\iota}`\alpha :$

$[*30{\cdot}4] \quad \supset : x \breve{\iota} \alpha . \equiv . x = \breve{\iota}`\alpha .$

$[*52{\cdot}6] \quad \equiv . x \in \alpha$

$\vdash . (1) . *30{\cdot}33 . \supset$

$\vdash :: \text{Hp} . \supset :. \phi(\breve{\iota}`\alpha) . \equiv : x \breve{\iota} \alpha . \supset_x . \phi x : \equiv : (\exists x) . x \breve{\iota} \alpha .$

$\vdash . (2) . (3) . \supset \vdash . \text{Prop}$

$*52{\cdot}602.\ \vdash :. \hat{z}(\phi z) \in 1 . \supset : \psi(\imath x)(\phi x) . \equiv . \phi x \supset_x \psi x . \equiv . (\exists x) .$

$[*52{\cdot}1$

$*52{\cdot}61.\ \vdash :. \alpha \in 1 . \supset : \breve{\iota}`\alpha \in \beta . \equiv . \alpha \subset \beta . \equiv . \exists!(\alpha \cap \beta) \quad \left[*52{\cdot}601 \frac{x \in \beta}{\phi x}\right]$

$*52{\cdot}62.\ \vdash :. \alpha, \beta \in 1 . \supset : \alpha = \beta . \equiv . \breve{\iota}`\alpha = \breve{\iota}`\beta$

Dem.

$\vdash . *52{\cdot}601 . \supset \vdash :: \text{Hp} . \supset :. \breve{\iota}`\alpha = \breve{\iota}`\beta . \equiv : x \in \alpha . \supset_x . x = \breve{\iota}`\beta :$

$[*52{\cdot}6] \quad \equiv : x \in \alpha . \supset_x . x \in \beta :$

$[*52{\cdot}46] \quad \equiv : \alpha = \beta :: \supset \vdash . \text{Prop}$

$*52{\cdot}63.\ \vdash : \alpha, \beta \in 1 . \alpha \neq \beta . \supset . \alpha \cap \beta = \Lambda \quad [*52{\cdot}46 . \text{Transp}]$

$*52{\cdot}64.\ \vdash : \alpha \in 1 . \supset . \alpha \cap \beta \in 1 \cup \iota`\Lambda$

Dem.

$\vdash . *52{\cdot}43 . \quad \supset \vdash : \text{Hp} . \exists! \alpha \cap \beta . \supset . \alpha \cap \beta \in 1 :$

$[*5{\cdot}6 . *24{\cdot}54] \supset \vdash :. \text{Hp} . \supset : \alpha \cap \beta = \Lambda . \text{v} . \alpha \cap \beta \in 1 :$

$[*51{\cdot}236] \quad \supset : \alpha \cap \beta \in 1 \cup \iota`\Lambda :. \supset \vdash . \text{Prop}$

$*52{\cdot}7.\ \vdash :. \beta - \alpha \in 1 . \alpha \subset \xi . \xi \subset \beta . \supset : \xi = \alpha . \text{v} . \xi = \beta$

Dem.

$\vdash . *22{\cdot}41 . \quad \supset \vdash : \text{Hp} . \xi \subset \alpha . \supset . \xi = \alpha \quad (1)$

$\vdash . *24{\cdot}55 . \quad \supset \vdash : \sim(\xi \subset \alpha) . \supset . \exists! \xi - \alpha \quad (2)$

$\vdash . *22{\cdot}48 . \quad \supset \vdash : \text{Hp} . \quad \supset . \xi - \alpha \subset \beta - \alpha \quad (3)$

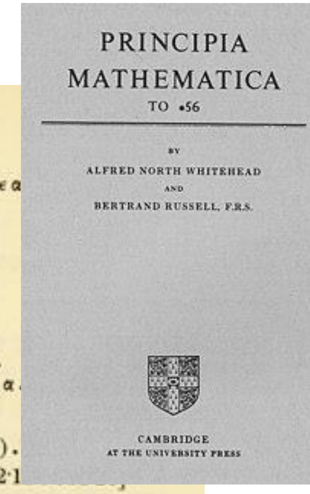
$\vdash . (2) . (3) . \quad \supset \vdash : \text{Hp} . \sim(\xi \subset \alpha) . \supset . \exists! \xi - \alpha . \xi - \alpha \subset \beta - \alpha \quad (4)$

$\vdash . *52{\cdot}1 . \quad \supset \vdash : \text{Hp} . \supset . (\exists x) . \beta - \alpha = \iota`x \quad (5)$

$\vdash . (4) . (5) . *51{\cdot}4 . \supset \vdash : \text{Hp} . \sim(\xi \subset \alpha) . \supset . \xi - \alpha = \beta - \alpha .$

$[*24{\cdot}411] \quad \supset . \xi = \beta \quad (6)$

$\vdash . (1) . (6) . \supset \vdash . \text{Prop}$



PRINCIPIA MATHEMATICA
TO *56

BY
ALFRED NORTH WHITEHEAD
AND
BERTRAND RUSSELL, F.R.S.

CAMBRIDGE
AT THE UNIVERSITY PRESS

https://writings.stephenwolfram.com/2010/11/100-years-since-principia-mathematica/

Propositional Resolution Inferenc

Pivot

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

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

F is a CNF formula; X and Y are two clauses in F

R be a resolvent of X and Y

Then,

$F \cup \{ R \}$ is semantically equivalent to F

- R is implied by F
- Any model that makes F true, also makes R true
- Adding R to F does not make F any harder to satisfy

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

Resolution Theorem

F be a set of clauses (i.e., a formula in CNF)

$Res(F) = F \cup \{R \mid R \text{ is a resolvent of two clauses in } F\}$

Res^n is defined recursively as follows:

$$Res^0(F) = F$$

$$Res^{n+1}(F) = Res(Res^n(F)), \text{ for } n \geq 0$$

$$Res^*(F) = \bigcup_{n \geq 0} Res^n(F)$$

Theorem: A CNF F is UNAT iff Res*(F) contains an empty clause

Exercise from LCS

For the following set of clauses determine Res^n for n=0, 1, 2

$$A \vee \neg B \vee C$$

$$B \vee C$$

$$\neg A \vee C$$

$$B \vee \neg C$$

$$\neg C$$

UNIVERSITY OF WATERLOO

Resolution Proof Example

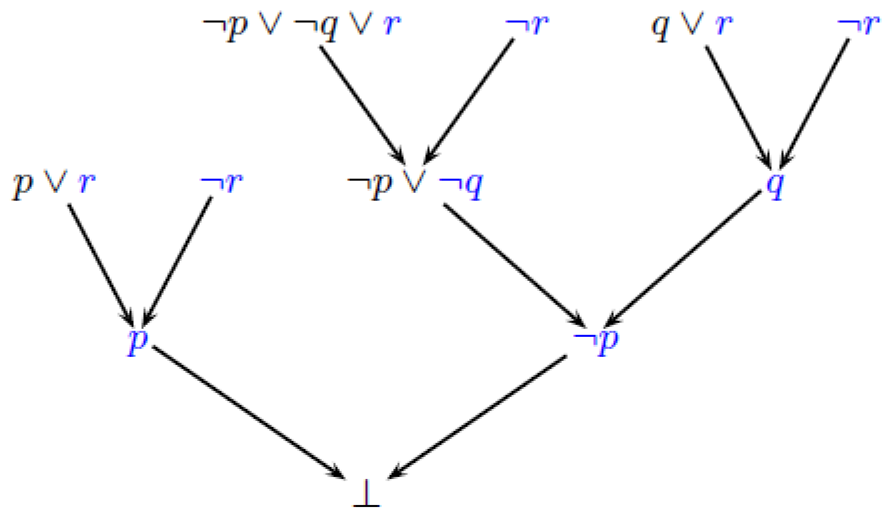
Show by resolution that the following CNF is UNSAT

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

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

Example of a resolution proof

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



Proof of the Resolution Theorem 1/3

(Soundness) By Resolution Lemma, F is equivalent to $Res^i(F)$ for any i.

Let n be such that $Res^{n+1}(F)$ contains an empty clause, but $Res^n(F)$ does not

- such n must exist because an empty clause was added at some point

Then, $Res^n(F)$ must contain two unit clauses L and $\neg L$

- because the only way to construct an empty clause is to resolve two unit clauses

Hence, F is UNSAT

- every clause added by resolution is implied (entailed) by F
- hence, $F \rightarrow L$ and $F \rightarrow \neg L$
- Therefore, $F \rightarrow (L \land \neg L)$, and $F \rightarrow$ False

Proof of the Resolution Theorem 2/3

(Completeness) By **induction** on the number of different atomic propositions in F.

(base case) if F has 0 atomic propositions and has a clause, then F contains an empty clause

- empty clause is the only clause without any atomic propositions

Proof of the Resolution Theorem 3/3

(inductive case):

Assume F is UNSAT and F has atomic propositions $A_1, \ldots A_{n+1}$

Let F_0 be the result of replacing atomic proposition A_{n+1} by 0

Let F_1 be the result of replacing atomic proposition A_{n+1} by 1

Since F is UNSAT, so are F_0 and F_1

- e.g., if F_0 is SAT with assignment M, then extend M to $A_{n+1} \rightarrow 0$, …

By IH, both F_0 and F_1 derive an empty clause

- Hence, Res*(F) contains (A_{n+1}) (or empty clause) and Res*(F) contains $(\neg A_{n+1})$ (or empty clause)

Therefore, Res*(F) contains an empty clause!

Example for the last step of Pf of Res Theorem

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

$F_0 = (a) \wedge (\neg a) \wedge (\neg c)$

- $Res^*(F_0)$ contains an empty clause
- By following the same resolution steps in F, we show that $Res^*(F)$ contains the clause (b)

$F_1 = (a) \wedge (c) \wedge (\neg c)$

- $Res^*(F_1)$ contains an empty clause
- By following the same resolution steps in F, we show that $Res^*(F)$ contains the clause $(\neg b)$

Therefore, $Res^*(F)$ contains an empty clause!

Propositional Resolution as an Inference Rule

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

Propositional resolution is a sound inference rule

Proposition resolution proof system consists of a single propositional resolution rule

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

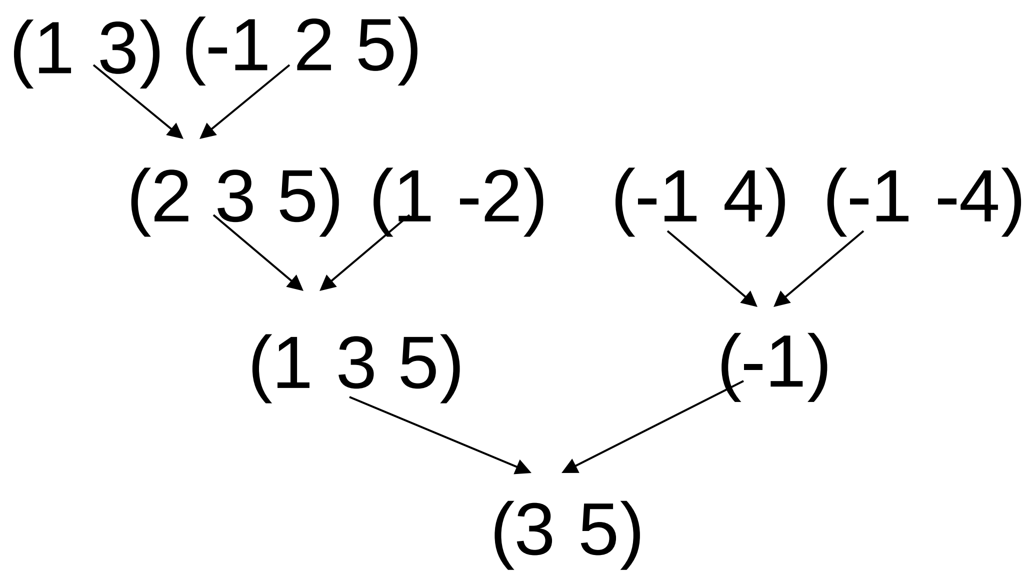
Theorem: Propositional resolution is sound and complete proof system for propositional logic!

Proof by resolution

Notation: positive numbers mean variables, negative mean negation

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

We'll try to prove φ → (3 5)



Example: UNSAT Derivation

Notation: positive numbers mean variables, negative mean negation
Let φ = (1 3) ∧ (-1 2) ∧ (1 -2) ∧ (-1 4) ∧ (-1 -4) ∧ (-3)

$$
\begin{tikzcd}
(1\ 3) \arrow[dr] & & (-1\ 2) \arrow[dl] & & & & & \\
& (2\ 3) \arrow[dr] & & (1\ {-2}) \arrow[dl] & & ({-1}\ 4) \arrow[dr] & & ({-1}\ {-4}) \arrow[dl] \\
& & (1\ 3) \arrow[drr] & & & & ({-1}) \arrow[dll] & \\
& & & & (3) \arrow[dr] & & ({-3}) \arrow[dl] & \\
& & & & & () & &
\end{tikzcd}
$$

Logic for Computer Scientists: Ex. 33

Using resolution show that

$$A \wedge B \wedge C$$

is a consequence of

$$\neg A \vee B$$

$$\neg B \vee C$$

$$A \vee \neg C$$

$$A \vee B \vee C$$

UNIVERSITY OF WATERLOO

Logic for Computer Scientists: Ex. 34

Show using resolution that F is valid

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

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