Mathematics for CS II (CS202: Logic)

Subhajit Roy
subhajit@cse.iitk.ac.in

Computer Science and Engineering,
Indian Institute of Technology, Kanpur
http://web.cse.iitk.ac.in/~subhajit/cs202/

What is logic?

What is logic?

- *logic* puzzles...

What is logic?

- *logic* puzzles...
- *logic* circuits...

What is logic?

- *logic* puzzles...
- *logic* circuits...
- program *logic*...

What is logic?

- *logic* puzzles...
- *logic* circuits...
- program *logic*...
- logic in conversation: tere baat mein koi *logic* nahin hai...

What is logic?

- *logic* puzzles...
- *logic* circuits...
- program *logic*...
- logic in conversation: tere baat mein koi *logic* nahin hai...

ponder...

What is this course about?

Mathematics?

What is this course about?

Mathematics + **Applications**

What is this course about?

Mathematics + **Applications**

Mathematical Logic + **Computational Logic**

Course Logistics

Course Component	**Percentage**
Assignments	30%
Quizzes	$2 \times 10\%$
End-Sem Exam	50%

Course Logistics

Course Component	Percentage
Assignments	30%
Quizzes	$2 \times 10\%$
End-Sem Exam	50%

- *Start assignments early*: they might look easy but will take some doing. No deadline extensions.
- *Attend classes and keep in sync with the lectures*: logic is an easy course, but can get difficult if you don't give it enough time.
- *Seek help from me and TAs*: we are there to help you.
- *No unfair means*: CSE policy on unfair means is quite strict; the same will be followed in this course.

Course Logistics

Course webpage:

https://web.cse.iitk.ac.in/users/subhajit/courses/CS335_Jul2018/CS202.php

(link from my homepage)

Course Logistics

Course webpage:
https://web.cse.iitk.ac.in/users/subhajit/courses/CS335_Jul2018/CS202.php

(link from my homepage)

Assugnment submission through: *moodle.cse.iitk.ac.in*

Books

- Michael Huth and Mark Ryan. Logic in Computer Science: Modelling and Reasoning about Systems. 2nd Edition.
- Mordechai Ben-Ari. Mathematical Logic for Computer Science. 3rd ed. 2012 Edition. [Acknowledgement: some figures are taken from the teaching material provided]
- Herbert B. Enderton. A Mathematical Introduction to Logic. 2nd Edition.
- Madhavan Mukund and S P Suresh. Introduction to Logic.

Logic: The language of mathematics

“The aim of logic in computer science is to *develop languages to model the situations* we encounter as computer science professionals, in such a way that *we can reason about them formally*.”

- Huth and Ryan

Logic: The language of mathematics

"The aim of logic in computer science is to *develop languages to model the situations* we encounter as computer science professionals, in such a way that *we can reason about them formally*."

- Huth and Ryan

Formal Reasoning

Construct arguments that can be defended rigorously or checked by a machine.

Why this formalism?

- Premise: *Some cars* rattle.
- Premise: My car is *some car*.
- Conclusion: My car rattles.

– Smullyan (1978, p. 183)

Why this formalism?

- Premise: *Some cars* rattle.
- Premise: My car is *some car*.
- Conclusion: My car rattles.

– Smullyan (1978, p. 183)

Informal natural reasoning can be flawed!

Propositional Logic

If the train arrives late and there are no taxis at the station, then John is late for his meeting. John is not late for his meeting. The train did arrive late. Therefore, there were taxis at the station.

Propositional Logic

If the train arrives late and there are no taxis at the station, then John is late for his meeting. John is not late for his meeting. The train did arrive late. Therefore, there were taxis at the station.

Is the above argument correct?

Propositional Logic

If the train arrives late and there are no taxis at the station, then John is late for his meeting. John is not late for his meeting. The train did arrive late. Therefore, there were taxis at the station.

If it is raining and Jane does not have her umbrella with her, then she will get wet. Jane is not wet. It is raining. Therefore, Jane has her umbrella with her.

Propositional Logic

If the train arrives late and there are no taxis at the station, then John is late for his meeting. John is not late for his meeting. The train did arrive late. Therefore, there were taxis at the station.

If it is raining and Jane does not have her umbrella with her, then she will get wet. Jane is not wet. It is raining. Therefore, Jane has her umbrella with her.

Compare the above two statements.

Propositional Logic

If the train arrives late and there are no taxis at the station, then John is late for his meeting. John is not late for his meeting. The train did arrive late. Therefore, there were taxis at the station.

If it is raining and Jane does not have her umbrella with her, then she will get wet. Jane is not wet. It is raining. Therefore, Jane has her umbrella with her.

Same structure!

If **p** *and not* **q**, *then* **r**. *Not* **r**. **p** . *Therefore*, **q**.

Symbolic Logic

If the train arrives late and there are no taxis at the station, then John is late for his meeting. John is not late for his meeting. The train did arrive late. Therefore, there were taxis at the station.

Symbolic Logic

If <u>the train arrives late</u> and there are no <u>taxis at the station</u> , then <u>John is late for his meeting</u> . John is not late for his meeting. The train did arrive late. Therefore, there were taxis at the station.

If **p** *and not* **q**, *then* **r**. *Not* **r**. **p** *.* *Therefore*, **q**.

Symbolic Logic

If the train arrives late and there are no taxis at the station , then John is late for his meeting . John is not late for his meeting. The train did arrive late. Therefore, there were taxis at the station.

If **p** *and not* **q**, *then* **r**. *Not* **r**. **p** . *Therefore*, **q**.

Propositions

- **p** $\equiv$ *the train arrives late*
- **q** $\equiv$ *there are taxis at the station*
- **r** $\equiv$ *John is late for his meeting*

Make it easier to write: more symbols

Make it easier to write: more symbols

If **p** *and not* **q**, *then* **r**. *Not* **r**. **p** *.* *Therefore*, **q**.

Make it easier to write: more symbols

If **p** *and not* **q**, *then* **r**. *Not* **r**. **p** *.* *Therefore*, **q**.

Operators

- $\wedge \equiv$ *and*
- $\neg \equiv$ *not*
- $\rightarrow \equiv$ *then*

Make it easier to write: more symbols

If **p** *and not* **q**, *then* **r**. *Not* **r**. **p** . *Therefore*, **q**.

Operators

- $\wedge \equiv$ *and*
- $\neg \equiv$ *not*
- $\rightarrow \equiv$ *then*

$\mathbf{p} \wedge \neg \mathbf{q} \rightarrow \mathbf{r}$ *;* $\neg \mathbf{r}$. **p** . *Therefore*, **q**.

Make it easier to write: more symbols

If **p** *and not* **q**, *then* **r**. *Not* **r**. **p** *.* *Therefore*, **q**.

Operators

- $\wedge \equiv$ *and*
- $\neg \equiv$ *not*
- $\rightarrow \equiv$ *then*

$\mathbf{p} \wedge \neg\mathbf{q} \rightarrow \mathbf{r}$ *;* $\neg\mathbf{r}$. **p** *.* *Therefore*, **q**.

We can *compose* these operators to create arguments of arbitrary complexity!

Understanding the language formally

What is a language?

Understanding the language formally

What is a language?

Language

A set of strings.

Understanding the language formally

What is a language?

Language

A set of strings.

Syntax + **Semantics**

Understanding the language formally

What is a language?

Language

A set of strings.

Syntax + **Semantics**

Syntax

Defines the **structure** of strings in the language

Understanding the language formally

What is a language?

Language

A set of strings.

Syntax + **Semantics**

Syntax

Defines the **structure** of strings in the language

Semantics

Defines the **meaning** of strings in the language

Propositional Logic: Syntax

Countably infinite set of atomic propositions, $\mathcal{P} = \{p_0, p_{1,} \ldots\}$ and three logical connectives $\neg, \vee, \wedge$

Propositional Formulas

The set Φ of formulas of propositional logic is the smallest set satisfying the following conditions:

- Every atomic proposition $p \in \Phi$
- If $\alpha \in \Phi$, then $(\neg\alpha) \in \Phi$
- If $\alpha, \beta \in \Phi$, then $\alpha \vee \beta \in \Phi$
- If $\alpha, \beta \in \Phi$, then $\alpha \wedge \beta \in \Phi$

Propositional Logic: Syntax

Countably infinite set of atomic propositions, $\mathcal{P} = \{p_0, p_{1,} \dots\}$ and three logical connectives $\neg, \vee, \wedge$

Propositional Formulas

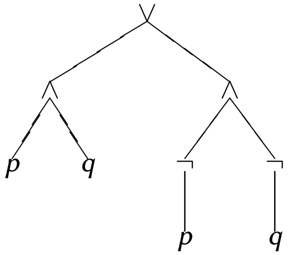
The set Φ of formulas of propositional logic is the smallest set satisfying the following conditions:

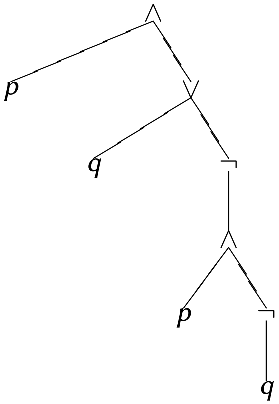
- Every atomic proposition $p \in \Phi$
- If $\alpha \in \Phi$, then $(\neg\alpha) \in \Phi$
- If $\alpha, \beta \in \Phi$, then $\alpha \vee \beta \in \Phi$
- If $\alpha, \beta \in \Phi$, then $\alpha \wedge \beta \in \Phi$

Backus–Naur (BNF) form

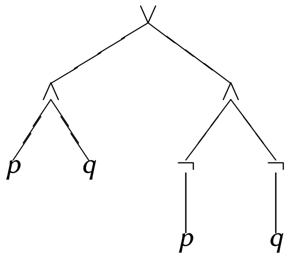
$\Phi ::= p \mid \neg\Phi \mid \Phi \wedge \Phi \mid \Phi \vee \Phi$

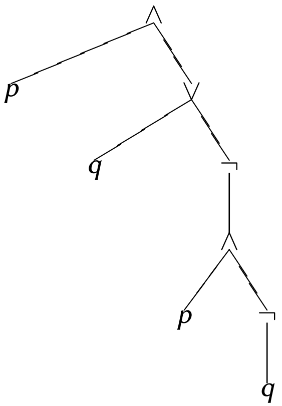
(Abstract) Syntax Trees





(Abstract) Syntax Trees





What are the formula (strings) corresponding to the two trees?

Selecting ASTs

Precedence: What operator binds “tigher”? [$\neg$, $\wedge$, $\vee$]

Selecting ASTs

Precedence: What operator binds “tigher”? [$\neg$, $\wedge$, $\vee$]

Associativity: How do the operators associate?

Additional Symbols

- Implication ($\rightarrow$)
- Biconditional ($\leftrightarrow$)

Additional Symbols

- Implication ($\rightarrow$)
- Biconditional ($\leftrightarrow$)

Excercise: Modify the syntax to include these additional symbols

Formal Semantics

Interpretation

An **interpretation** for a formula $A \in \mathscr{F}$ is a *total function* $\mathscr{I}_{\mathrm{A}} \mapsto \{T, F\}$ that assigns a truth value to *every* atom in the set of propositions $\mathscr{P}_A$.

Formal Semantics

Interpretation

An **interpretation** for a formula $A \in \mathscr{F}$ is a *total function* $\mathscr{I}_{\text{A}} \mapsto \{T, F\}$ that assigns a truth value to *every* atom in the set of propositions $\mathscr{P}_A$.

Intuition: Every row in a truth table is an interpretation

Formal Semantics

Interpretation

An **interpretation** for a formula $A \in \mathscr{F}$ is a *total function* $\mathscr{I}_{\mathrm{A}} \mapsto \{T, F\}$ that assigns a truth value to *every* atom in the set of propositions $\mathscr{P}_A$.

Intuition: Every row in a truth table is an interpretation

Interpretation for set of formulas

An **interpretation** can be lifted to a set of formulas $S = \{A_1, \ldots\}$ over propositions $\mathscr{P}_S = \bigcup_i \mathscr{P}_{A_i}$ as a function $\mathscr{I}_A \mapsto \{T, F\}$

Formal Semantics

Interpretation

An **interpretation** for a formula $A \in \mathscr{F}$ is a *total function* $\mathscr{I}_{\mathrm{A}} \mapsto \{T, F\}$ that assigns a truth value to *every* atom in the set of propositions $\mathscr{P}_A$.

Intuition: Every row in a truth table is an interpretation

Interpretation for set of formulas

An **interpretation** can be lifted to a set of formulas $S = \{A_1, \dots\}$ over propositions $\mathscr{P}_S = \bigcup_i \mathscr{P}_{A_i}$ as a function $\mathscr{I}_A \mapsto \{T, F\}$

Valuation

The **valuation** of $A \in \mathscr{F}$ is defined as a total function $v_{\mathscr{I}_A} \mapsto \{T, F\}$ that assigns a truth value to A under $\mathscr{I}_A$. Note that $v_{\mathscr{I}_A}$ can be defined inductively on the structure of A.

Semantics (as truth table)

A	$v(A_1)$	$v(A_2)$	$v(A)$
$\neg A_1$	T		F
$\neg A_1$	F		T
$A_1 \vee A_2$	F	F	F
$A_1 \vee A_2$	otherwise		T
$A_1 \wedge A_2$	T	T	T
$A_1 \wedge A_2$	otherwise		F

Satisfiability, Validity

Satisfiability

$A \in \mathscr{F}$ is satisfiable iff $v_{\mathscr{I}_A}(A) = T$ for *some* interpretation of $\mathscr{I}$
A satisfying interpretation is called a **model** for A

Satisfiability, Validity

Satisfiability

$A \in \mathscr{F}$ is satisfiable iff $v_{\mathscr{I}_A}(A) = T$ for *some* interpretation of $\mathscr{I}$
A satisfying interpretation is called a **model** for A

Validity

$A \in \mathscr{F}$ is valid (denoted $\models A$) iff $v_{\mathscr{I}}(A) = T$ for all interpretations of $\mathscr{I}$
A valid propositional formula is called a **taulology**

Satisfiability, Validity

Satisfiability

$A \in \mathscr{F}$ is satisfiable iff $v_{\mathscr{I}_A}(A) = T$ for *some* interpretation of $\mathscr{I}$
A satisfying interpretation is called a **model** for A

Validity

$A \in \mathscr{F}$ is valid (denoted $\models A$) iff $v_{\mathscr{I}}(A) = T$ for all interpretations of $\mathscr{I}$
A valid propositional formula is called a **taulology**

Unsatisfiable

$A \in \mathscr{F}$ is unsatisfiable iff $v_{\mathscr{I}}(A) = F$ for *all* interpretations of $\mathscr{I}$
Also referred to as a **contradiction**
[*not satisfiable*]

Satisfiability, Validity

Falsifiable

$A \in \mathscr{F}$ is falsifiable (denoted $\not\models A$) iff $v_{\mathscr{I}}(A) = F$ for *some* interpretations of $\mathscr{I}$
[*not valid*]

Satisfiability, Validity

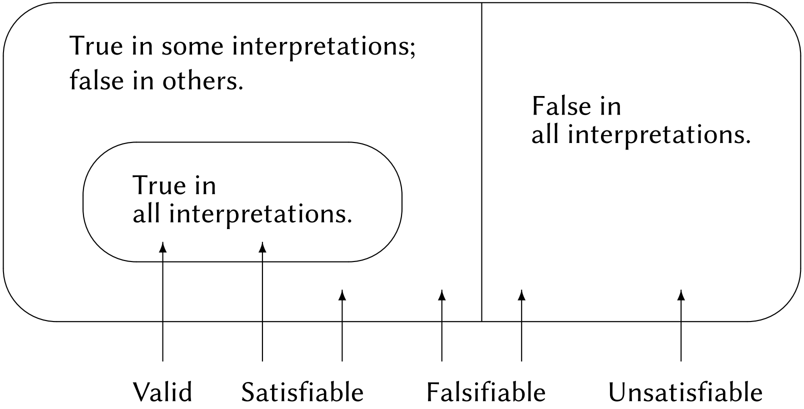
Falsifiable

$A \in \mathscr{F}$ is falsifiable (denoted $\not\models A$) iff $v_{\mathscr{I}}(A) = F$ for *some* interpretations of $\mathscr{I}$
[*not valid*]

All these concepts can be lifted to a set of formulas $A_i \in \mathscr{F}$ instead of a single formula.

For example, a set of formulas $A_i \in \mathscr{F}$ is satisfiable iff $v_{\mathscr{I}}(A_i) = T$ on *all* A_i for *some* interpretation of $\mathscr{I}$

Satisfiability, Validity



Solving computer science problems using SAT solving

Graph Coloring

Given a graph G and (atmost) k colors, assign colors to the nodes such that no two neighbours share the same color.

Solving computer science problems using SAT solving

Graph Coloring

Given a graph G and (atmost) k colors, assign colors to the nodes such that no two neighbours share the same color.

Considered a hard problem!

Solving computer science problems using SAT solving

Graph Coloring

Given a graph G and (atmost) k colors, assign colors to the nodes such that no two neighbours share the same color.

Considered a hard problem!

Encoding in SAT

- Variables:
  e_{ij}: an edge exists between the vertices i and j
  a_{ij}: the i^{th} vertex is assigned the j^{th} color
- Constraints (for k colors):
  - Each node v is assigned a color: $\forall_v.(a_{v1} \lor a_{v2} \lor \cdots \lor a_{vk})$
  - Each node v is given exactly one color: $\forall_v \neg(a_{vi} \land a_{vj})$
  - Two neighboring nodes don't have same color: $\forall_{v,w} \neg(e_{vw} \land a_{vi} \land a_{wi})$

Conjunctive Normal Form (CNF)

For many applications, it is required to use restricted syntactic forms.

Conjunctive Normal Form (CNF)

For many applications, it is required to use restricted syntactic forms.

CNF

- *CNF* ≡ $\underline{\text{conjunction}}$ of clauses (set of clauses)
- *clause* ≡ $\underline{\text{disjunction}}$ of literals (set of literals)
- *literal* ≡ $\underline{\text{proposition}}$ or negated proposition

AND of ORs

Conjunctive Normal Form (CNF)

For many applications, it is required to use restricted syntactic forms.

CNF

- *CNF*$\equiv$ $\underline{\text{conjunction}}$ of clauses (set of clauses)
- *clause* $\equiv$ $\underline{\text{disjunction}}$ of literals (set of literals)
- *literal* $\equiv$ $\underline{\text{proposition}}$ or negated proposition

AND of ORs

Example: $(p_1 \vee \neg p_2 \vee p_3) \wedge (p_1 \vee p_5) \wedge \ldots$

Conjunctive Normal Form (CNF)

For many applications, it is required to use restricted syntactic forms.

CNF

- *CNF* ≡ $\underline{\text{conjunction}}$ of clauses (set of clauses)
- *clause* ≡ $\underline{\text{disjunction}}$ of literals (set of literals)
- *literal* ≡ $\underline{\text{proposition}}$ or negated proposition

AND of ORs

Example: $(p_1 \vee \neg p_2 \vee p_3) \wedge (p_1 \vee p_5) \wedge \ldots$

Can be seen as a list of list of literals!

Disjunctive Normal Form (CNF)

DNF

- *DNF* $\equiv$ disjunction of cubes (set of cubes)
- *clause* $\equiv$ conjunction of literals (set of literals)
- *literal* $\equiv$ proposition or negated proposition

OR of ANDs

Disjunctive Normal Form (CNF)

DNF

- *DNF* $\equiv$ $\underline{\text{disjunction}}$ of cubes (set of cubes)
- *clause* $\equiv$ $\underline{\text{conjunction}}$ of literals (set of literals)
- *literal* $\equiv$ $\underline{\text{proposition}}$ or negated proposition

OR of ANDs

Example: $(p_1 \wedge \neg p_2 \wedge p_3) \vee (p_1 \wedge p_5) \vee \ldots$

Disjunctive Normal Form (CNF)

DNF

- *DNF* $\equiv$ disjunction of cubes (set of cubes)
- *clause* $\equiv$ conjunction of literals (set of literals)
- *literal* $\equiv$ proposition or negated proposition

OR of ANDs

Example: $(p_1 \wedge \neg p_2 \wedge p_3) \vee (p_1 \wedge p_5) \vee \ldots$

Can be seen as a list of list of literals!

Negation Normal Form (NNF)

NNF

the negation operator ($\neg$) is applied only to variables

Negation Normal Form (NNF)

NNF

the negation operator ($\neg$) is applied only to variables

Not a cannonical form

Converting to Conjunctive Normal Form

Conversion

1. Convert to NNF: push negations into the formula (De Morgan's Law):
   $\neg(p \lor q)$ to $(\neg p) \land (\neg q)$
   $\neg(p \land q)$ to $(\neg p) \lor (\neg q)$
2. Repeatedly apply the distributive law where a disjunction occurs over a conjunction:
   $p \lor (q \land r)$ to $(p \lor q) \land (p \lor r)$

Converting to Conjunctive Normal Form

Conversion

1. Convert to NNF: push negations into the formula (De Morgan's Law):
   $\neg(p \lor q)$ to $(\neg p) \land (\neg q)$
   $\neg(p \land q)$ to $(\neg p) \lor (\neg q)$
2. Repeatedly apply the distributive law where a disjunction occurs over a conjunction:
   $p \lor (q \land r)$ to $(p \lor q) \land (p \lor r)$

For DNF, use distribution of $\land$ over $\lor$

Converting to Conjunctive Normal Form

Conversion

1. Convert to NNF: push negations into the formula (De Morgan's Law):
   $\neg(p \lor q)$ to $(\neg p) \land (\neg q)$
   $\neg(p \land q)$ to $(\neg p) \lor (\neg q)$
2. Repeatedly apply the distributive law where a disjunction occurs over a conjunction:
   $p \lor (q \land r)$ to $(p \lor q) \land (p \lor r)$

For DNF, use distribution of $\land$ over $\lor$

What can be the blowup in the worst case?

SAT solvers

- Given a formula ϕ, a SAT solver checks the satisfiability of the formula. It provides either:
  - SAT: a model
  - UNSAT: proof of unsatisfiability

SAT solvers

- Given a formula ϕ, a SAT solver checks the satisfiability of the formula. It provides either:
  - SAT: a model
  - UNSAT: proof of unsatisfiability
- How to check validity?

SAT solvers

- Given a formula ϕ, a SAT solver checks the satisfiability of the formula. It provides either:
  - SAT: a model
  - UNSAT: proof of unsatisfiability
- How to check validity?
- It accepts the formula as a CNF formula in the DIMACS representation:

```
c
c start with comments
c
c
p cnf 5 3
1 -5 4 0
-1 5 3 4 0
-3 -4 0
```

Assignment 1: Sudoku

Write an encoding for:

- Sudoku solving, and
    - Additional constraint: The numbers in the main diagonals are also unique
- Sudoku generation
    - Use the above encoding as a black box

Implement it using the minisat solver.

Logical Equivalence $\equiv$

Logical Equivalence

$A_1 \equiv A_2$ if and only if $v_{\mathscr{I}}(A_1) = v_{\mathscr{I}}(A_2)$ for all interpretations $\mathscr{I}$.

Mathematical Induction

Mathematical Induction

- Assume an **induction hypothesis**
- Prove the **base case**
- Prove the **inductive step** for *k+1, assuming induction hypothesis on k*

Mathematical Induction

Mathematical Induction

- Assume an **induction hypothesis**
- Prove the **base case**
- Prove the **inductive step** for *k+1, assuming induction hypothesis on k*

Example: Prove that the sum of n numbers is $\frac{n(n+1)}{2}$

Structural Induction

Structural Induction

To show that some property ϕ holds for all logical formula ψ
Base case: Show that ϕ holds for all propositions p
Induction Hypothesis: Assume that the formula holds for formula A_1 and A_2
Inductive Step: Do a case analysis to show that the formula holds for $\neg A_1$, $A_1 \wedge A_2$ and $A_1 \vee A_2$

Structural Induction

Structural Induction

To show that some property ϕ holds for all logical formula ψ
Base case: Show that ϕ holds for all propositions p
Induction Hypothesis: Assume that the formula holds for formula A_1 and A_2
Inductive Step: Do a case analysis to show that the formula holds for $\neg A_1$, $A_1 \wedge A_2$ and $A_1 \vee A_2$

Proof: By mathematical induction of the height of the syntax trees of $\neg A_1$, $A_1 \wedge A_2$ and $A_1 \vee A_2$

Structural Induction

Structural Induction

To show that some property ϕ holds for all logical formula ψ

Base case: Show that ϕ holds for all propositions p

Induction Hypothesis: Assume that the formula holds for formula A_1 and A_2

Inductive Step: Do a case analysis to show that the formula holds for $\neg A_1$, $A_1 \wedge A_2$ and $A_1 \vee A_2$

Proof: By mathematical induction of the height of the syntax trees of $\neg A_1$, $A_1 \wedge A_2$ and $A_1 \vee A_2$

prove larger formula by induction on its subformulas

Structural Induction

Theorem

For every well-formed propositional logic formula, the number of left brackets is equal to the number of right brackets.
$\Phi ::= p \mid (\neg\Phi) \mid (\Phi \wedge \Phi) \mid (\Phi \vee \Phi)$

Structural Induction

Theorem

For every well-formed propositional logic formula, the number of left brackets is equal to the number of right brackets.
$\Phi ::= p \mid (\neg\Phi) \mid (\Phi \wedge \Phi) \mid (\Phi \vee \Phi)$

Proof: Base Case: For the case of propositions p, #left(p)=#right(p)=0
Inductive Step: Consider the case for $(\Phi_1 \wedge \Phi_2)$, #left(Φ_i)=#right(Φ_i)=n, i=1,2 (induction hypothesis)
So, #left$(\Phi_1 \wedge \Phi_2)$= #left(Φ_1) + #left(Φ_2) + 1 = $n_1 + n_2 + 1$
And, #right$(\Phi_1 \wedge \Phi_2)$= #right(Φ_1) + #right(Φ_2) + 1 = $n_1 + n_2 + 1$
So, proved for all conjunctions. SImilarly prove for negation and disjunction.

Proof System

Proof System (Deductive System)

- a set (possibly infinite) set of formulas, called *axioms*
- a set of *inference rules*

used to deduce **tautologies**

Proof System

Proof System (Deductive System)

- a set (possibly infinite) set of formulas, called *axioms*
- a set of *inference rules*

used to deduce **tautologies**

Proofs

- A sequent $\phi_1, \phi_2, \ldots, \phi_n \vdash \psi_1, \psi_2, \ldots, \psi_m$ is valid, if a proof for it can be found. ϕ_i are premises/antecedents, ψ_i are conclusions/consequents.
- A proof can be denoted:
  - As a proof tree, where each node is an axiom (leaf node) or inference rule (non-leaf node)
  - As a sequence of valid sequents $\gamma_1, \gamma_2, \ldots \gamma_i, \ldots, \gamma_n$. For a sequent γ_i to be valid, its premise(s) is either some ϕ_i, an axiom, or a conclusion proven in some preceeding γ_j, $j < i$

Proof System

Proof System (Deductive System)

- a set (possibly infinite) set of formulas, called *axioms*
- a set of *inference rules*

used to deduce **tautologies**

Proofs

- A sequent $\phi_1, \phi_2, \ldots, \phi_n \vdash \psi_1, \psi_2, \ldots, \psi_m$ is valid, if a proof for it can be found. ϕ_i are premises/antecedents, ψ_i are conclusions/consequents.
- A proof can be denoted:
  - As a proof tree, where each node is an axiom (leaf node) or inference rule (non-leaf node)
  - As a sequence of valid sequents $\gamma_1, \gamma_2, \ldots \gamma_i, \ldots, \gamma_n$. For a sequent γ_i to be valid, its premise(s) is either some ϕ_i, an axiom, or a conclusion proven in some preceeding γ_j, $j < i$

[illegible]

Proof System

Proof System (Deductive System)

- a set (possibly infinite) set of formulas, called *axioms*
- a set of *inference rules*

used to deduce **tautologies**

Proofs

- A sequent $\phi_1, \phi_2, \ldots, \phi_n \vdash \psi_1, \psi_2, \ldots, \psi_m$ is valid, if a proof for it can be found. ϕ_i are premises/antecedents, ψ_i are conclusions/consequents.
- A proof can be denoted:
  - As a proof tree, where each node is an axiom (leaf node) or inference rule (non-leaf node)
  - As a sequence of valid sequents $\gamma_1, \gamma_2, \ldots \gamma_i, \ldots, \gamma_n$. For a sequent γ_i to be valid, its premise(s) is either some ϕ_i, an axiom, or a conclusion proven in some preceeding γ_j, $j < i$

[illegible]

Natural Deduction

Natural Deduction (ND) is one of the proof systems for propositional logic

Natural Deduction

Natural Deduction (ND) is one of the proof systems for propositional logic

$$\frac{premises}{conclusion}\,rule\_name$$

Natural Deduction

Natural Deduction (ND) is one of the proof systems for propositional logic

$$\frac{premises}{conclusion}\,rule\_name$$

To Prove: $p \wedge q, r \vdash q \wedge r$

$$\cfrac{\cfrac{p \wedge q}{q}\wedge e_2 \quad r}{q \wedge r}\wedge i$$

proof tree

1. $p \wedge q$ (premise)
2. r (premise)
3. q ($\wedge e_2$ 1)
4. $q \wedge r$ ($\wedge i$ 3,2)

flattened proof

Natural Deduction

Natural Deduction (ND) is one of the proof systems for propositional logic

$$\frac{premises}{conclusion}\,rule\_name$$

To Prove: $p \land q, r \vdash q \land r$

$$\cfrac{\cfrac{p \land q}{q}\land e_2 \quad r}{q \land r}\land i$$ *proof tree*	1. $p \land q$ (premise) 2. r (premise) 3. q ($\land e_2$ 1) 4. $q \land r$ ($\land i$ 3,2) *flattened* proof

How to check a proof:

Natural Deduction

Natural Deduction (ND) is one of the proof systems for propositional logic

$$\frac{premises}{conclusion}\, rule\_name$$

To Prove: $p \land q, r \vdash q \land r$

$$\cfrac{\cfrac{p \land q}{q}\land e_2 \quad r}{q \land r}\land i$$ *proof tree*	1. $p \land q$ (premise) 2. r (premise) 3. q ($\land e_2$ 1) 4. $q \land r$ ($\land i$ 3,2) *flattened* proof

How to check a proof:

- *Check that each line corresponds to a valid rule, axiom on a set of premises which have already been established*

Natural Deduction

Conjunction

$$\frac{\phi \quad \psi}{\phi \land \psi}\land i \qquad \frac{\phi \land \psi}{\phi}\land e_1 \qquad \frac{\phi \land \psi}{\psi}\land e_2$$

Natural Deduction

Conjunction

$$\frac{\phi \quad \psi}{\phi \wedge \psi}\wedge i \qquad \frac{\phi \wedge \psi}{\phi}\wedge e_1 \qquad \frac{\phi \wedge \psi}{\psi}\wedge e_2$$

Procedural Interpretation:

1. $\wedge i$: first prove ϕ and ψ, and use them for $\phi \wedge \psi$
2. $\wedge e$: to prove ϕ, prove $\phi \wedge \psi$ (strengthen proof); it is useful when $\phi \wedge \psi$ is already lying around

Natural Deduction

Disjunction

$$\frac{\phi}{\phi \lor \psi}\lor i_1 \qquad \frac{\psi}{\phi \lor \psi}\lor i_2 \qquad \frac{\phi \lor \psi \quad \boxed{\begin{array}{c}\phi\\ \dots\\ \gamma\end{array}} \quad \boxed{\begin{array}{c}\psi\\ \dots\\ \gamma\end{array}}}{\gamma}\lor e$$

Natural Deduction

Disjunction

$$\frac{\phi}{\phi \vee \psi}\vee i_1 \qquad \frac{\psi}{\phi \vee \psi}\vee i_2 \qquad \frac{\phi \vee \psi \quad \boxed{\begin{array}{c}\phi\\ \dots\\ \gamma\end{array}} \quad \boxed{\begin{array}{c}\psi\\ \dots\\ \gamma\end{array}}}{\gamma}\vee e$$

Procedural Interpretation:

1. $\vee i$: to prove $\phi \vee \psi$, try proving ϕ (or ψ)
2. $\vee e$: to prove γ from $\phi \wedge \psi$ try proving it from both ϕ and ψ

Natural Deduction

Disjunction

$$\frac{\phi}{\phi \vee \psi}\vee i_1 \qquad \frac{\psi}{\phi \vee \psi}\vee i_2 \qquad \frac{\phi \vee \psi \quad \boxed{\begin{array}{c}\phi\\ \dots\\ \gamma\end{array}} \quad \boxed{\begin{array}{c}\psi\\ \dots\\ \gamma\end{array}}}{\gamma}\vee e$$

Procedural Interpretation:

1. $\vee i$: to prove $\phi \vee \psi$, try proving ϕ (or ψ)
2. $\vee e$: to prove γ from $\phi \wedge \psi$ try proving it from both ϕ and ψ

Note the *scope* (shown by the box): the scope contains *local assumptions*, and hence the derived formulas are not valid outside the proof

Natural Deduction

Negation

$$\dfrac{\boxed{\begin{array}{c}\phi\\ \cdots\\ \bot\end{array}}}{\neg\phi}\,\neg i \qquad \dfrac{\phi \quad \neg\phi}{\bot}\,\neg e \qquad \dfrac{\neg\neg\phi}{\phi}\,\neg\neg$$

Natural Deduction

Negation

$$\dfrac{\boxed{\begin{array}{c}\phi \\ \dots \\ \bot\end{array}}}{\neg\phi}\,\neg i \qquad \dfrac{\phi \quad \neg\phi}{\bot}\,\neg e \qquad \dfrac{\neg\neg\phi}{\phi}\,\neg\neg$$

Procedural Interpretation:

1. $\neg\phi$: to prove $\neg\phi$, assume ϕ and try to derive a contradiction (proof by contradiction)

Natural Deduction

$\perp$

$\perp$ (no intro rule) $\dfrac{\perp}{\phi}\perp$

Natural Deduction

$\perp$

$\perp$ (no intro rule) $\dfrac{\perp}{\phi}\perp$

Procedural Interpretation:

1. $\perp$: anything goes in a mad world (a world with contradictions)

Natural Deduction

$\rightarrow$

$$\frac{\boxed{\begin{array}{c}\phi\\ \dots\\ \psi\end{array}}}{\phi \rightarrow \psi} \rightarrow i_1 \qquad \frac{\phi \quad \phi \rightarrow \psi}{\psi} \rightarrow e^{a} \qquad \frac{\phi \rightarrow \psi \quad \neg\psi}{\neg\phi} \rightarrow e^{b}$$

[a] modus ponens
[b] modus tollens

Natural Deduction

$\rightarrow$

$$\frac{\boxed{\begin{array}{c}\phi \\ \dots \\ \psi\end{array}}}{\phi \rightarrow \psi} \rightarrow i_1 \qquad \frac{\phi \quad \phi \rightarrow \psi}{\psi} \rightarrow e^{a} \qquad \frac{\phi \rightarrow \psi \quad \neg\psi}{\neg\phi} \rightarrow e^{b}$$

[a] modus ponens
[b] modus tollens

Procedural Interpretation:

1. $\rightarrow i$: to prove $\phi \rightarrow \psi$, assume ϕ and prove ψ

Natural Deduction

Bi-implication

$$\frac{\boxed{\begin{array}{c}\phi\\ \ldots\\ \psi\end{array}} \quad \boxed{\begin{array}{c}\psi\\ \ldots\\ \phi\end{array}}}{\phi \leftrightarrow \psi} \rightarrow i_1 \qquad \frac{\phi \quad \phi \leftrightarrow \psi}{\psi} \rightarrow e1 \qquad \frac{\psi \quad \phi \leftrightarrow \psi}{\phi} \rightarrow e2$$

ND Proofs

To Prove: $(q \longrightarrow r) \vdash ((\neg q \longrightarrow \neg p) \longrightarrow (p \longrightarrow r))$

1. $q \longrightarrow r$	(premise)
$((\neg q \longrightarrow \neg p) \longrightarrow (p \longrightarrow r))$	

Proofs

1. $q \longrightarrow r$ (premise)
2. $\neg q \longrightarrow \neg p$ (assumption)
3. p (assumption)

7. r ($\longrightarrow$e 1, 6)
8. $p \longrightarrow r$ ($\longrightarrow$i 3–7)
9. $(\neg q \longrightarrow \neg p) \longrightarrow (p \longrightarrow r)$ ($\longrightarrow$i 2–8)

Proofs

1. $q \longrightarrow r$ (premise)

2. $\neg q \longrightarrow \neg p$ (assumption)

3. p (assumption)

7. r ($\longrightarrow$e 1, 6)

8. $p \longrightarrow r$ ($\longrightarrow$i 3–7)

9. $(\neg q \longrightarrow \neg p) \longrightarrow (p \longrightarrow r)$ ($\longrightarrow$i 2–8)

How to go about a proof

1. This “structure” of the proof can be deduced from the formula to be proved.
2. The left out part needs “cleverness”.

Proofs

1.	$q \longrightarrow r$	(premise)
2.	$\neg q \longrightarrow \neg p$	(assumption)
3.	p	(assumption)
4.	$\neg\neg p$	($\neg\neg$i 3)
5.	$\neg\neg q$	(MT 2, 4)
6.	q	($\neg\neg$e 5)
7.	r	($\longrightarrow$e 1, 6)
8.	$p \longrightarrow r$	($\longrightarrow$i 3–7)
9.	$(\neg q \longrightarrow \neg p) \longrightarrow (p \longrightarrow r)$	($\longrightarrow$i 2–8)

How to go about a proof

1. This "structure" of the proof can be deduced from the formula to be proved.
2. The left out part needs "cleverness".

Premises versus Assumptions

- The proof is subject to the provided premises; they are globally valid and can be assumed everywhere.
- The assumptions are made during the proof (eg. to do case-split for $\vee$); lemmas proved with assumptions are valid only within the respective scope.

ND Proofs

General heuristic:

1. Start with applying the introduction rules, backward from the conclusion. Ex., to prove $A \longrightarrow B$, add A as an assumption to get to B. To prove $A \bigwedge B$, use $\wedge i$ to prove A, and then prove B. (often dictated by the structure of the formula to be proved)
2. When you run out, use elimination rules to go forward. Ex. If you have $A \longrightarrow B$ and A, derive B. If you have $A \bigvee B$, split on cases, considering A in one case and B in the other.

ND Proofs

General heuristic:

1. Start with applying the introduction rules, backward from the conclusion. Ex., to prove $A \longrightarrow B$, add A as an assumption to get to B. To prove $A \bigwedge B$, use $\wedge i$ to prove A, and then prove B. (often dictated by the structure of the formula to be proved)
2. When you run out, use elimination rules to go forward. Ex. If you have $A \longrightarrow B$ and A, derive B. If you have $A \bigvee B$, split on cases, considering A in one case and B in the other.

Structure of the proof

1. The structure of the proof can be formed from the syntactic structure of the formula to be proved
2. The "gaps" in this structure needs to be deduced in a clever way

Practice on certain proofs

1. $\neg q \to \neg p \vdash p \to \neg\neg q$
2. $p \to (q \to r) \vdash p \land q \to r$
3. $p \to q \vdash p \land r \to q \land r$
4. $(p \lor q) \lor r \vdash p \lor (q \lor r)$
5. $p \land (q \lor r) \vdash (p \land q) \lor (p \land r)$
6. $p \to q, p \to \neg q \vdash \neg p$
7. $p \to \neg p \vdash \neg \mathrm{p}$

Semantics: Logical Consequence

- For a set of formulas U and a formula A, A is *logical consequence* of U (denoted $U \vDash A$) if and only if U is a model of A .
  - A is true in all interpretations that U is true
  - Example: $\{p, \neg q\} \models A$

Semantics: Logical Consequence

- For a set of formulas U and a formula A, A is *logical consequence* of U (denoted $U \vDash A$) if and only if U is a model of A .
    - A is true in all interpretations that U is true
    - Example: $\{p, \neg q\} \models A$
- Note the difference between $\vdash$ (derives, syntactic) and $\vDash$ (models, semantic)

Semantics: Logical Consequence

- For a set of formulas U and a formula A, A is *logical consequence* of U (denoted $U \vDash A$) if and only if U is a model of A .
  - A is true in all interpretations that U is true
  - Example: $\{p, \neg q\} \models A$
- Note the difference between $\vdash$ (derives, syntactic) and $\vDash$ (models, semantic)
- Note the difference between $\models$ and $\rightarrow$ (also $\equiv$ and $\leftrightarrow$)
  - $\rightarrow$ and $\leftrightarrow$ are symbols in the language of propositional logic
  - $\models$ and $\equiv$ are symbols in the matalanguage—the language used to describe properties of the language of propositional language

Theories

Closure under $\models$

A set of formulas $\mathscr{T}$ is closed under logical consequence iff for all formulas A, if $\mathscr{T} \models A$.

Theories

Closure under $\models$

A set of formulas $\mathscr{T}$ is closed under logical consequence iff for all formulas A, if $\mathscr{T} \models A$.

Theory

A set of formulas $\mathscr{T}$ closed under logical consequence is a theory. The elements of $\mathscr{T}$ are theorems.

Theories

Closure under $\models$

A set of formulas $\mathscr{T}$ is closed under logical consequence iff for all formulas A, if $\mathscr{T} \models A$.

Theory

A set of formulas $\mathscr{T}$ closed under logical consequence is a theory. The elements of $\mathscr{T}$ are theorems.

Theories are constructed by selecting *axioms* (A) and deducing their logical consequence.

Axiomatization

Axiomatizable

A theory $\mathscr{T}$ is *axiomatizable* iff there exist axioms U such that $\mathscr{T} = \{\text{formulas } A \text{ such that } U \models A\}$. If U is finite, $\mathscr{T}$ is *finitely* axoiomatizable.

Axiomatization

Axiomatizable

A theory $\mathscr{T}$ is *axiomatizable* iff there exist axioms U such that $\mathscr{T} = \{\text{formulas } A \text{ such that } U \models A\}$. If U is finite, $\mathscr{T}$ is *finitely* axoiomatizable.

- Tautologies are not interesting. It is more interesting to construct theories (logical consequences under a given set of axioms).

Axiomatization

Axiomatizable

A theory $\mathscr{T}$ is *axiomatizable* iff there exist axioms U such that $\mathscr{T} = \{\text{formulas } A \text{ such that } U \models A\}$. If U is finite, $\mathscr{T}$ is *finitely* axoiomatizable.

- Tautologies are not interesting. It is more interesting to construct theories (logical consequences under a given set of axioms).
- Arithmetic is axiomatizable with *Peano's axioms*. However, it is *not finitely axiomatizable* as the induction axiom is not a single axiom but a axiom schema (includes one axiom per predicate definable in the first-order language of Peano arithmetic).

Decision Procedures in Propositional Logic

Decision Procedures

An *algorithm* is a decision procedure for a given set of formulas $\mathscr{U} \subseteq \mathscr{F}$ if, given an arbitrary formula $A \in \mathscr{F}$ it *always terminates*, and returns answer **yes** if $A \in \mathscr{U}$ and **no** if $A \notin \mathscr{U}$.

Decision Procedures in Propositional Logic

Decision Procedures

An *algorithm* is a decision procedure for a given set of formulas $\mathscr{U} \subseteq \mathscr{F}$ if, given an arbitrary formula $A \in \mathscr{F}$ it *always terminates*, and returns answer **yes** if $A \in \mathscr{U}$ and **no** if $A \notin \mathscr{U}$.

- If $\mathscr{U}$ is a set of satisfiable formulas, the decision procedure for $\mathscr{U}$ is called a *decision procedure for satisfiability* (similarly for validity)

Decision Procedures in Propositional Logic

Decision Procedures

An *algorithm* is a decision procedure for a given set of formulas $\mathscr{U} \subseteq \mathscr{F}$ if, given an arbitrary formula $A \in \mathscr{F}$ it *always terminates*, and returns answer **yes** if $A \in \mathscr{U}$ and **no** if $A \notin \mathscr{U}$.

- If $\mathscr{U}$ is a set of satisfiable formulas, the decision procedure for $\mathscr{U}$ is called a *decision procedure for satisfiability* (similarly for validity)
- A decision procedure for satisfiability can be used for validity by checking the negation of the formula for satisfiability. Such a procedure is called *refutation procedure* (we are proving validity by refuting its negation).

Decision Procedures in Propositional Logic

Decision Procedures

An *algorithm* is a decision procedure for a given set of formulas $\mathscr{U} \subseteq \mathscr{F}$ if, given an arbitrary formula $A \in \mathscr{F}$ it *always terminates*, and returns answer **yes** if $A \in \mathscr{U}$ and **no** if $A \notin \mathscr{U}$.

- If $\mathscr{U}$ is a set of satisfiable formulas, the decision procedure for $\mathscr{U}$ is called a *decision procedure for satisfiability* (similarly for validity)
- A decision procedure for satisfiability can be used for validity by checking the negation of the formula for satisfiability. Such a procedure is called *refutation procedure* (we are proving validity by refuting its negation).
- Truth table construction is an (inefficient) decision procedure.

Truth Tables as Decision Procedure

p	q	$p \to q$	$\neg p$	$\neg q$	$\neg q \to \neg p$	$(p \to q) \leftrightarrow (\neg q \to \neg p)$
T	T	T	F	F	T	T
T	F	F	F	T	F	T
F	T	T	T	F	T	T
F	F	T	T	T	T	T

Truth Tables as Decision Procedure

p	q	$p \to q$	$\neg p$	$\neg q$	$\neg q \to \neg p$	$(p \to q) \leftrightarrow (\neg q \to \neg p)$
T	T	T	F	F	T	T
T	F	F	F	T	F	T
F	T	T	T	F	T	T
F	F	T	T	T	T	T

It can also be tabulated as follows:

$(p$	$\to$	$q)$	$\leftrightarrow$	$(\neg$	q	$\to$	$\neg$	$p)$
T		F			F			T
T		F		T	F			T
T		F		T	F		F	T
T		F		T	F	F	F	T
T	F	F		T	F	F	F	T
T	F	F	T	T	F	F	F	T

Truth Tables as Decision Procedure

a compact tabulation

Truth Tables as Decision Procedure

a compact tabulation

p	q	$(p$	$\rightarrow$	$q)$	$\leftrightarrow$	$(\neg$	q	$\rightarrow$	$\neg$	$p)$
T	T	T	T	T	T	F	T	T	F	T
T	F	T	F	F	T	T	F	F	F	T
F	T	F	T	T	T	F	T	T	T	F
F	F	F	T	F	T	T	F	T	T	F

Is $p \vee q \equiv q \vee p$?

Is $p \vee q \equiv q \vee p$?

p	q	$p \vee q$	$\neg p$	$\neg q$	$(p \vee q) \wedge \neg p \wedge \neg q$
T	T	T	F	F	F
T	F	T	F	T	F
F	T	T	T	F	F
F	F	F	T	T	F

Is $p \vee q \equiv q \vee p$?

p	q	$p \vee q$	$\neg p$	$\neg q$	$(p \vee q) \wedge \neg p \wedge \neg q$
T	T	T	F	F	F
T	F	T	F	T	F
F	T	T	T	F	F
F	F	F	T	T	F

Don't be fooled by the simplicity of truth tables; it is incredibly powerful tool!

Semantic tableaux

An efficient decision procedure (for satisfiability)

Semantic tableaux

An efficient decision procedure (for satisfiability)

Basic Algorithm

- Recursively *decompose* the formula down to literals

Semantic tableaux

An efficient decision procedure (for satisfiability)

Basic Algorithm

- Recursively *decompose* the formula down to literals
- Check for *contradiction*: existance of positive and negative literal on the same node

Semantic tableaux

An efficient decision procedure (for satisfiability)

Basic Algorithm

- Recursively *decompose* the formula down to literals
- Check for *contradiction*: existance of positive and negative literal on the same node
- Formula is satisfiable if for atleast one leaf node:
  - there does *not exist a pending* subformula (pending subformula is one which has never been picked for decomposition)
  - there does n *ot exist a contradiction* on the leaf

Formula Decomposition

- Formulas classified into α and β formulas
  - *α formulas* require both the components to be satisfied; both components from the decomposition get added to the current list of subformulas
  - *β formulas* require any one of components to be satisfied; leads to splitting of the current path, each path inheriting one of the components from the decomposition

Semantic tableau for $p\wedge (\ngg q\vee \ngg p)$$

$$\begin{tikzcd}[column sep=small]
 & p \wedge (\neg q \vee \neg p) \arrow[d] & \\
 & p, \neg q \vee \neg p \arrow[dl] \arrow[dr] & \\
p, \neg q & & p, \neg p \\
\odot & & \times
\end{tikzcd}$$

Semantic tableau for $p\wedge (\ngg q\vee \ngg p)$

$$\begin{tikzcd}
& p \land (\neg q \lor \neg p) \arrow[d] & \\
& p, \neg q \lor \neg p \arrow[dl] \arrow[dr] & \\
p, \neg q & & p, \neg p \\
\odot & & \times
\end{tikzcd}$$

Model: $p, \neg q$

Semantic tableaux $(p \lor q) \land (\neg p \land \neg q)$

$$(p \lor q) \land (\neg p \land \neg q)$$

Semantic tableaux $(p \vee q) \wedge (\neg p \wedge \neg q)$

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

$$\downarrow$$

Semantic tableaux $(p \vee q) \wedge (\neg p \wedge \neg q)$

$$\begin{tikzcd}[row sep=small] (p \vee q) \wedge (\neg p \wedge \neg q) \\ \checkmark \arrow[d] \\ p \vee q, \neg p \wedge \neg q \end{tikzcd}$$

Semantic tableaux $(p \vee q) \wedge (\neg p \wedge \neg q)$

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

$$\checkmark$$

$$\downarrow$$

$$p \vee q, \neg p \wedge \neg q$$

$$\downarrow$$

Semantic tableaux $(p \vee q) \wedge (\neg p \wedge \neg q)$

$$\begin{tikzcd}[row sep=large]
(p \vee q) \wedge (\neg p \wedge \neg q) \arrow[d, "\checkmark"'] \\
p \vee q, \neg p \wedge \neg q \arrow[d, "\checkmark"'] \\
p \vee q, \neg p, \neg q
\end{tikzcd}$$

Semantic tableaux $(p \lor q) \land (\neg p \land \neg q)$

$$\begin{tikzcd}
& (p \lor q) \underset{\checkmark}{\land} (\neg p \land \neg q) \arrow[d] & \\
& p \lor q, \neg p \underset{\checkmark}{\land} \neg q \arrow[d] & \\
& p \lor q, \neg p, \neg q \arrow[dl] \arrow[dr] & \\
{} & & {}
\end{tikzcd}$$

Semantic tableaux $(p \vee q) \wedge (\neg p \wedge \neg q)$

$$\begin{tikzcd}
 & \underset{\checkmark}{(p \vee q) \wedge (\neg p \wedge \neg q)} \arrow[d] & \\
 & p \vee q, \underset{\checkmark}{\neg p \wedge \neg q} \arrow[d] & \\
 & \underset{\checkmark}{p \vee q}, \neg p, \neg q \arrow[dl] \arrow[dr] & \\
p, \neg p, \neg q & & q, \neg p, \neg q
\end{tikzcd}$$

Semantic tableaux $(p \vee q) \wedge (\neg p \wedge \neg q)$

$$\begin{tikzcd}
& \underset{\checkmark}{(p \vee q) \wedge (\neg p \wedge \neg q)} \arrow[d] & \\
& p \vee q, \underset{\checkmark}{\neg p \wedge \neg q} \arrow[d] & \\
& \underset{\checkmark}{p \vee q}, \neg p, \neg q \arrow[dl] \arrow[dr] & \\
\underset{\times}{p, \neg p, \neg q} & & \underset{\times}{q, \neg p, \neg q}
\end{tikzcd}$$

Semantic tableaux $(p \lor q) \land (\neg p \land \neg q)$

Semantic tableaux may **not** be unique!

Semantic tableaux $(p \lor q) \land (\neg p \land \neg q)$

Semantic tableaux may **not** be unique!

Another tableaux for $(p \lor q) \land (\neg p \land \neg q)$

$$\begin{tikzcd}
& (p \lor q) \land (\neg p \land \neg q) \arrow[d] & \\
& p \lor q, \neg p \land \neg q \arrow[dl] \arrow[dr] & \\
p, \neg p \land \neg q \arrow[d] & & q, \neg p \land \neg q \arrow[d] \\
p, \neg p, \neg q & & q, \neg p, \neg q \\
\times & & \times
\end{tikzcd}$$

Semantic tableaux $(p \lor q) \land (\neg p \land \neg q)$

Semantic tableaux may **not** be unique!

Another tableaux for $(p \lor q) \land (\neg p \land \neg q)$

$$\begin{tikzcd}
& (p \lor q) \land (\neg p \land \neg q) \arrow[d] & \\
& p \lor q, \neg p \land \neg q \arrow[dl] \arrow[dr] & \\
p, \neg p \land \neg q \arrow[d] & & q, \neg p \land \neg q \arrow[d] \\
p, \neg p, \neg q & & q, \neg p, \neg q \\
\times & & \times
\end{tikzcd}$$

Which one is better?

Semantic tableaux $(p \vee q) \wedge (\neg p \wedge \neg q)$

Semantic tableaux may **not** be unique!

Another tableaux for $(p \vee q) \wedge (\neg p \wedge \neg q)$

$$\begin{tikzcd}
 & (p \vee q) \wedge (\neg p \wedge \neg q) \arrow[d] & \\
 & p \vee q, \neg p \wedge \neg q \arrow[dl] \arrow[dr] & \\
p, \neg p \wedge \neg q \arrow[d] & & q, \neg p \wedge \neg q \arrow[d] \\
p, \neg p, \neg q & & q, \neg p, \neg q \\
\times & & \times
\end{tikzcd}$$

Which one is better?

Heuristic: *Process conjunctions before disjunctions*

α-formulas

α	α_1	α_2
$\neg\neg A_1$	A_1	
$A_1 \wedge A_2$	A_1	A_2
$\neg(A_1 \vee A_2)$	$\neg A_1$	$\neg A_2$
$\neg(A_1 \rightarrow A_2)$	A_1	$\neg A_2$
$A_1 \leftrightarrow A_2$	$A_1 \rightarrow A_2$	$A_2 \rightarrow A_1$

β-formulas

β	β_1	β_2
$\neg(B_1 \wedge B_2)$	$\neg B_1$	$\neg B_2$
$B_1 \vee B_2$	B_1	B_2
$B_1 \rightarrow B_2$	$\neg B_1$	B_2
$\neg(B_1 \leftrightarrow B_2)$	$\neg(B_1 \rightarrow B_2)$	$\neg(B_2 \rightarrow B_1)$

Termination of Tableau Construction

Completed Tableau

- A tableau whose construction has terminated is a *completed tableau*.
- A completed tableau is *closed* is all leaves are closed;
- A completed tableau is *open* if some leaves are open.

Termination of Tableau Construction

Completed Tableau

- A tableau whose construction has terminated is a *completed tableau*.
- A completed tableau is *closed* is all leaves are closed;
- A completed tableau is *open* if some leaves are open.

Theorem

The construction of a tableau for any formula ψ terminates. When the construction terminates, all the leaves are marked $\times$ or $\odot$

Termination of Tableau Construction

How to do proofs of termination?

Termination of Tableau Construction

How to do proofs of termination?

Ranking Function

Come up with a *ranking function* such it:

- The function has a *lower boun*d (say it is always non-negative)
- The function *decreases* with each step of the algorithm

Termination of Tableau Construction

Proof

- For the set of (sub)formulas in the pending list, let
  - b(l): total number of binary operators
  - n(l): total number of negations
- Let us use a ranking function over the subformulas in list l in a node: $W(l) = 3.b(l) + n(l)$
- For each α-formula, argue that $W(l)$ decreases
- For each β-formula, argue that each child has a lower $W(l)$ value than the parent

Analytic tableaux (Smullyan, 1968)

Two alternations for efficiency:

- Subformulas not copied from parent to child
- Subformulas for decomposition selected from all nodes from root to the current node (not only from the parent node)
- Check for contradiction on the path (not only on the node)

Analytic tableaux (Smullyan, 1968)

Two alternations for efficiency:

- Subformulas not copied from parent to child
- Subformulas for decomposition selected from all nodes from root to the current node (not only from the parent node)
- Check for contradiction on the path (not only on the node)

$$(p \lor q) \land (\neg p \land \neg q)$$

$$\downarrow$$

Analytic tableaux (Smullyan, 1968)

Two alternations for efficiency:

- Subformulas not copied from parent to child
- Subformulas for decomposition selected from all nodes from root to the current node (not only from the parent node)
- Check for contradiction on the path (not only on the node)

$$\begin{tikzcd} \underset{\checkmark}{(p \vee q) \wedge (\neg p \wedge \neg q)} \arrow[d] \\ p \vee q, \neg p \wedge \neg q \end{tikzcd}$$

Analytic tableaux (Smullyan, 1968)

Two alternations for efficiency:

- Subformulas not copied from parent to child
- Subformulas for decomposition selected from all nodes from root to the current node (not only from the parent node)
- Check for contradiction on the path (not only on the node)

$$\begin{tikzcd}
\overset{}{(p \vee q) \wedge (\neg p \wedge \neg q)} \\
\checkmark \arrow[d] \\
p \vee q, \neg p \wedge \neg q \arrow[d] \\
{}
\end{tikzcd}$$

Analytic tableaux (Smullyan, 1968)

Two alternations for efficiency:

- Subformulas not copied from parent to child
- Subformulas for decomposition selected from all nodes from root to the current node (not only from the parent node)
- Check for contradiction on the path (not only on the node)

$$\begin{tikzcd}[row sep=large] (p \vee q) \wedge (\neg p \wedge \neg q) \\ \overset{\checkmark}{} \arrow[d] \\ p \vee q, \neg p \wedge \neg q \\ \overset{\checkmark}{} \arrow[d] \\ \neg p, \neg q \end{tikzcd}$$

Analytic tableaux (Smullyan, 1968)

Two alternations for efficiency:

- Subformulas not copied from parent to child
- Subformulas for decomposition selected from all nodes from root to the current node (not only from the parent node)
- Check for contradiction on the path (not only on the node)

$$\begin{tikzcd}
& \underset{\checkmark}{(p \lor q) \land (\neg p \land \neg q)} \arrow[d] & \\
& p \lor q, \underset{\checkmark}{\neg p \land \neg q} \arrow[d] & \\
& \neg p, \neg q \arrow[dl] \arrow[dr] & \\
{} & & {}
\end{tikzcd}$$

Analytic tableaux (Smullyan, 1968)

Two alternations for efficiency:

- Subformulas not copied from parent to child
- Subformulas for decomposition selected from all nodes from root to the current node (not only from the parent node)
- Check for contradiction on the path (not only on the node)

$$\begin{tikzcd}
 & (p \lor q) \land (\neg p \land \neg q) \arrow[d, "\checkmark"'] & \\
 & p \lor q, \neg p \land \neg q \arrow[d, "\checkmark"] & \\
 & \neg p, \neg q \arrow[dl, "\checkmark"'] \arrow[dr] & \\
p & & q
\end{tikzcd}$$

Analytic tableaux (Smullyan, 1968)

Two alternations for efficiency:

- Subformulas not copied from parent to child
- Subformulas for decomposition selected from all nodes from root to the current node (not only from the parent node)
- Check for contradiction on the path (not only on the node)

$$\begin{tikzcd}
& (p \lor q) \land (\neg p \land \neg q) \arrow[d, "\checkmark"'] & \\
& p \lor q, \neg p \land \neg q \arrow[d, "\checkmark"] & \\
& \neg p, \neg q \arrow[dl, "\checkmark"'] \arrow[dr] & \\
p & & q
\end{tikzcd}$$

Soundness

(verus valuation functions or semantic tableau)

Soundness

(verus valuation functions or semantic tableau)

Completeness

Compactness

Normal forms

CNF, NNF

What cannot be done in propositional logic

Need for more powerful logics