Propositional Logic

Bow-Yaw Wang

August 7, 2025

Section 1

Introduction

Outline

Logic and Reasoning

- Consider the following arguments:

Example

若火車誤點且車站沒有計程車，則小明開會就遲到。小明開會並沒有遲到，而火車誤點。那麼車站就有計程車。

Example

如果下雨而且小華沒帶雨傘，則小華會淋溼。小華並沒有淋溼，而外面正在下雨。那麼小華一定帶了雨傘。

- Both examples have the same structure:

p	火車誤點	下雨
q	車站有計程車	小華帶雨傘
r	小明開會遲到	小華淋溼

If p and not q, then r. Not r. p. Hence q.
（若 p 且 非 q，則 r。非 r， p。 則 q）

Propositions

- We will develop a language to reason such arguments.
- Our langauge is based on propositions (or declarative sentences).
- Examples:
  - The sum of 3 and 5 equals 8.
  - Every even natural number greater than two is the sum of two prime numbers (Goldbach's conjecture).
  - All hobbits like mushrooms in their soup.
- A proposition can either be "true" or "false."
- Non-examples:
  - When will we have lunch?
  - Run!

Atomic Sentences

- Certain sentences are the basic blocks of our language.
  - They are called atomic (or indecomposable) sentences.
- We will use $p, q, r, \ldots$ (possibly with sub- or super-scripts) to denote sentences.
- Examples:
  - Let p denote "I won the lottery last week."
  - Let q denote "I bought a lottery ticket."
  - Let r denote "I won last week's grand prize."
- In fact, p, q, and r are all atomic sentences.

Sentences

- Let $p, q, r, \ldots$ be sentences.
  - p : "I won the lottery last week."
  - q : "I bought a lottery ticket."
  - r : "I won last week's grand prize."
- We construct new sentences by the following connectives:
  - The negation of p (denoted by $\neg p$).
    - It is **not** true that "I won the lottery last week."
  - The disjunction of p and q (denoted by $p \vee q$).
    - "I won the lottery last week" **or** "I won last week's grand prize."
  - The conjunction of p and q (denoted by $p \wedge q$).
    - "I won the lottery last week" **and** "I bought a lottery ticket."
  - The implication of r and p (denoted by $r \implies p$).
    - "I won last week's grand prize" **implies** "I won the lottery last week."

Binding Priorities

- If p, q, r are sentences, $p \wedge q$ and $(\neg r) \vee q$ are sentences.
- $(p \wedge q) \implies ((\neg r) \vee q)$ is also a sentence.
- To reduce the number of parentheses, we adopt the following conventions:
  Convention.

strong		weak
$\neg$	$\{\vee, \wedge\}$	$\implies$

- Hence $p \wedge q \implies \neg r \vee q$ is indeed $(p \wedge q) \implies ((\neg r) \vee q)$.

Examples, Examples, Examples

- Let us rewrite our examples:

Example

若火車誤點且車站沒有計程車，則小明開會就遲到。小明開會並沒有遲到，而火車誤點。那麼車站就有計程車。

- We have the following atomic sentences:

p : 火車誤點	q : 車站有計程車	r : 小明開會遲到

- In our language, we write:
  - $p \land \neg q \implies r$ （若火車誤點且車站沒有計程車，則小明開會就遲到）
  - $\neg r$ （小明開會並沒有遲到）
  - p （火車誤點）
  - Hence q （車站就有計程車）

Examples, Examples, Examples

- Let us rewrite our examples:

Example

如果下雨而且小華沒帶雨傘，則小華會淋溼。小華並沒有淋溼，而外面正在下雨。那麼小華一定帶了雨傘。

- We have the following atomic sentences:

p : 下雨	q : 小華帶雨傘	r : 小華淋溼

- In our language, we write:
  - $p \wedge \neg q \implies r$ （如果下雨而且小華沒帶雨傘，則小華會淋溼）
  - $\neg r$ （小華並沒有淋溼）
  - p （外面正在下雨）
  - Hence q （小華一定帶了雨傘）

Section 2

Natural Deduction

Outline

Natural Deduction

- In our examples, we (informally) infer new sentences.
- In natural deduction, we have a collection of proof rules.
  - These proof rules allow us to infer new sentences logically followed from existing ones.
- Supose we have a set of sentences: $\phi_1, \phi_2, \ldots, \phi_n$ (called premises), and another sentence ψ (called a conclusion).
- The notation

$$\phi_1, \phi_2, \ldots, \phi_n \vdash \psi$$

  is called a sequent.
- A sequent is valid if a proof (built by the proof rules) can be found.
- We will try to build a proof for our examples. Namely,

$$p \wedge \neg q \implies r, \neg r, p \vdash q.$$

Proof Rules for Natural Deduction – Conjunction

- Suppose we want to prove a conclusion $\phi \land \psi$. What do we do?
  - Of course, we need to prove both ϕ and ψ so that we can conclude $\phi \land \psi$.
- Hence the proof rule for conjunction is

$$\frac{\phi \quad \psi}{\phi \land \psi} \land i$$

  - Note that premises are shown above the line and the conclusion is below. Also, $\land i$ is the name of the proof rule.
  - This proof rule is called "conjunction-introduction" since we introduce a conjunction ($\land$) in the conclusion.

Proof Rules for Natural Deduction – Conjunction

- For each connective, we have introduction proof rule(s) and also elimination proof rule(s).
- Suppose we want to prove a conclusion ϕ from the premise $\phi \land \psi$. What do we do?
  - We don't do any thing since we know ϕ already!
- Here are the elimination proof rules:

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

- The rule $\land e_1$ says: if you have a proof for $\phi \land \psi$, then you have a proof for ϕ by applying this proof rule.
- Why do we need two rules?
  - Because we want to manipulate syntax only.

Examples

Example

Prove $p \wedge q, r \vdash q \wedge r$.

Proof.

We are looking for a proof of the form:

$$\begin{array}{c} p \wedge q \quad r \\ \vdots \\ q \wedge r \end{array}$$

□

Examples

Example

Prove $p \land q, r \vdash q \land r$.

Proof.

We are looking for a proof of the form:

$$\cfrac{\cfrac{p \land q}{q}\ \land e_2 \qquad r}{q \land r}\ \land i$$

□

Examples

Example

Prove $p \wedge q, r \vdash q \wedge r$.

Proof.

We will write proofs in lines:

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

□

Proof Rules for Natural Deduction – Double Negation

- Suppose we want to prove ϕ from a proof for $\neg\neg\phi$. What do we do?
  - There is no difference between ϕ and $\neg\neg\phi$. The same proof suffices!
- Hence we have the following proof rules:

$$\frac{\phi}{\neg\neg\phi}\ \neg\neg i \qquad\qquad \frac{\neg\neg\phi}{\phi}\ \neg\neg e$$

Examples

Example

Prove $p, \neg\neg(q \wedge r) \vdash \neg\neg p \wedge r$.

Proof.

We are looking for a proof like:

$$\begin{array}{c} p \quad \neg\neg(q \wedge r) \\ \vdots \\ \neg\neg p \wedge r \end{array}$$

□

Examples

Example

Prove $p, \neg\neg(q \wedge r) \vdash \neg\neg p \wedge r$.

Proof.

We are looking for a proof like:

$$\dfrac{\dfrac{p}{\neg\neg p}\ \neg\neg i \qquad \dfrac{\dfrac{\neg\neg(q \wedge r)}{q \wedge r}\ \neg\neg e}{r}\ \wedge e_2}{\neg\neg p \wedge r}\ \wedge i$$

□

Examples

Example

Prove $p, \neg\neg(q \wedge r) \vdash \neg\neg p \wedge r$.

Proof.

We will write proofs in lines:

1	p	premise
2	$\neg\neg(q \wedge r)$	premise
3	$\neg\neg p$	$\neg\neg i$ 1
4	$q \wedge r$	$\neg\neg e$ 2
5	r	$\wedge e_2$ 4
6	$\neg\neg p \wedge r$	$\wedge i$ 3, 5

□

Proof Rules for Natural Deduction – Implication

- Suppose we want to prove ψ from proofs for ϕ and $\phi \implies \psi$. What do we do?
  - We just put the two proofs for ϕ and $\phi \implies \psi$ together.
- Here is the proof rule:

$$\frac{\phi \quad \phi \implies \psi}{\psi} \implies e$$

- This proof rule is also called *modus ponens*.
- Here is another proof rule related to implication:

$$\frac{\phi \implies \psi \quad \neg\psi}{\neg\phi} \; MT$$

- This proof rule is called *modus tollens*.

Example

Example

Prove $p \implies (q \implies r), p, \neg r \vdash \neg q$.

Proof.

1	$p \implies (q \implies r)$	premise
2	p	premise
3	$\neg r$	premise
4	$q \implies r$	$\implies e$ 2, 1
5	$\neg q$	MT 4, 3

□

Proof Rules for Natural Deduction – Implication

- Suppose we want to prove $\phi \implies \psi$. What do we do?
  - We assume ϕ to prove ψ. If succeed, we conclude $\phi \implies \psi$ without any assumption.
  - Note that ϕ is added as an assumption and then removed so that $\phi \implies \psi$ does not depend on ϕ.
- We use a "box" to simulate this strategy.
- Here is the proof rule:

$$\frac{\boxed{\begin{array}{c}\phi\\ \vdots\\ \psi\end{array}}}{\phi \implies \psi} \implies i$$

- At any point in a box, you can only use a sentence ϕ before that point. Moreover, no box enclosing the occurrence of ϕ has been closed.

Example

Example

Prove $\neg q \implies \neg p \vdash p \implies \neg\neg q$.

Proof.

$$\dfrac{\boxed{\dfrac{\neg q \implies \neg p \qquad \dfrac{p}{\neg\neg p}\,\neg\neg i}{\neg\neg q}\,MT}}{p \implies \neg\neg q}\implies i$$

1	$\neg q \implies \neg p$	premise
2	p	assumption
3	$\neg\neg p$	$\neg\neg i$ 2
4	$\neg\neg q$	MT 1, 3
5	$p \implies \neg\neg q$	$\implies i$ 2-4

□

Theorems

Example

Prove $\vdash p \implies p$.

Proof.

1	p	assumption
2	$p \implies p$	$\implies i$ 1 - 1

□

In the box, we have $\phi \equiv \psi \equiv p$.

Definition

A sentence ϕ such that $\vdash \phi$ is called a theorem.

Examples

Example

Prove $p \wedge q \implies r \vdash p \implies (q \implies r)$.

Proof.

1	$p \wedge q \implies r$	premise	
2	p	assumption	⎤
3	q	assumption	⎤ ⎢
4	$p \wedge q$	$\wedge i$ 2, 3	⎢ ⎢
5	r	$\implies e$ 4, 1	⎦ ⎢
6	$q \implies r$	$\implies i$ 3-5	⎦
7	$p \implies (q \implies r)$	$\implies i$ 2-6	

□

Proof Rules for Natural Deduction – Disjunction

- Suppose we want to prove $\phi \vee \psi$. What do we do?
  - We can either prove ϕ or ψ.
- Here are the proof rules:

$$\frac{\phi}{\phi \vee \psi} \vee i_1 \qquad \frac{\psi}{\phi \vee \psi} \vee i_2$$

  - Note the symmetry with $\wedge e_1$ and $\wedge e_2$.

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

  - Can we have a corresponding symmetric elimination rule for disjunction? Recall

$$\frac{\phi \quad \psi}{\phi \wedge \psi} \wedge i$$

Proof Rules for Natural Deduction – Disjunction

- Suppose we want to prove χ from $\phi \vee \psi$. What do we do?
  - We assume ϕ to prove χ and then assume ψ to prove χ.
  - If both succeed, χ is proved from $\phi \vee \psi$ without assuming ϕ and ψ.
- Here is the proof rule:

$$\frac{\phi \vee \psi \qquad \boxed{\begin{array}{c}\phi\\ \vdots\\ \chi\end{array}} \qquad \boxed{\begin{array}{c}\psi\\ \vdots\\ \chi\end{array}}}{\chi}\ \vee e$$

- In addition to nested boxes, we may have parallel boxes in our proofs.

Example

Recall that our syntax does not admit commutativity.

Example

Prove $p \vee q \vdash q \vee p$.

Proof.

$$\dfrac{p \vee q \qquad \boxed{\dfrac{p}{q \vee p}\ \vee i_2} \qquad \boxed{\dfrac{q}{q \vee p}\ \vee i_1}}{q \vee p}\ \vee e$$

1	$p \vee q$	premise	
2	p	assumption	]
3	$q \vee p$	$\vee i_2$ 2	]
4	q	assumption	]
5	$q \vee p$	$\vee i_1$ 4	]
6	$q \vee p$	$\vee e$ 1, 2-3, 4-5	

□

Example

Example

Prove $q \implies r \vdash p \vee q \implies p \vee r$.

Proof.

1	$q \implies r$	premise	
2	$p \vee q$	assumption	⌉
3	p	assumption	⌉
4	$p \vee r$	$\vee i_1$ 3	⌋
5	q	assumption	⌉
6	r	$\implies e$ 5, 1	
7	$p \vee r$	$\vee i_2$ 6	⌋
8	$p \vee r$	$\vee e$ 2, 3-4, 5-7	⌋
9	$p \vee q \implies p \vee r$	$\implies i$ 2-8	

□

Example

Example

Prove $p \wedge (q \vee r) \vdash (p \wedge q) \vee (p \wedge r)$.

Proof.

1	$p \wedge (q \vee r)$	premise	
2	p	$\wedge e_1$ 1	
3	$q \vee r$	$\wedge e_2$ 1	
4	q	assumption	⌉
5	$p \wedge q$	$\wedge i$ 2, 4	|
6	$(p \wedge q) \vee (p \wedge r)$	$\vee i_1$ 5	⌋
7	r	assumption	⌉
8	$p \wedge r$	$\wedge i$ 2, 7	|
9	$(p \wedge q) \vee (p \wedge r)$	$\vee i_2$ 8	⌋
10	$(p \wedge q) \vee (p \wedge r)$	$\vee e$ 3, 4-6, 7-9	

□

Example

Example

Prove $(p \wedge q) \vee (p \wedge r) \vdash p \wedge (q \vee r)$.

Proof.

1	$(p \wedge q) \vee (p \wedge r)$	premise	
2	$p \wedge q$	assumption	⌉
3	p	$\wedge e_1$ 2	|
4	q	$\wedge e_2$ 2	|
5	$q \vee r$	$\vee i_1$ 4	|
6	$p \wedge (q \vee r)$	$\wedge i$ 3, 5	⌋
7	$p \wedge r$	assumption	⌉
8	p	$\wedge e_1$ 7	|
9	r	$\wedge e_2$ 7	|
10	$q \vee r$	$\vee i_2$ 9	|
11	$p \wedge (q \vee r)$	$\wedge i$ 8, 10	⌋
12	$p \wedge (q \vee r)$	$\vee e$ 1, 2-6, 7-11	

□

Contradiction

Definition

Contradictions are sentences of the form $\phi \land \neg\phi$ or $\neg\phi \land \phi$.

- Examples:
  - $p \land \neg p$, $\neg(p \lor q \implies r) \land (p \lor q \implies r)$.
- Logically, any sentence can be proved from a contradiction.
  - If $0 = 1$, then $100 \neq 100$.
- Particularly, if ϕ and ψ are contradictions, we have $\phi \dashv\vdash \psi$.
  - $\phi \dashv\vdash \psi$ means $\phi \vdash \psi$ and $\psi \vdash \phi$ (called provably equivalent).
- Since all contradictions are equivalent, we will use the symbol $\bot$ (called "bottom") for them.
- We are now ready to discuss proof rules for negation.

Proof Rules for Natural Deduction – Negation

- Since any sentence can be proved from a contradiction, we have

$$\frac{\bot}{\phi}\ \bot e$$

- When both ϕ and $\neg\phi$ are proved, we have a contradiction.

$$\frac{\phi \quad \neg\phi}{\bot}\ \neg e$$

  - The proof rule could be called $\bot i$. We use $\neg e$ because it eliminates a negation.

Example

Example

Prove $\neg p \vee q \vdash p \implies q$.

Proof.

1	$\neg p \vee q$	premise	
2	$\neg p$	assumption	⌉
3	p	assumption	⌉ |
4	$\bot$	$\neg e$ 3, 2	| |
5	q	$\bot e$ 4	⌋ |
6	$p \implies q$	$\implies i$ 3-5	⌋
7	q	assumption	⌉
8	p	assumption	⌉ |
9	q	copy 7	⌋ |
10	$p \implies q$	$\implies i$ 8-9	⌋
11	$p \implies q$	$\vee e$ 1, 2-6, 7-10	

□

Proof Rules for Natural Deduction – Negation

- Suppose we want to prove $\neg\phi$. What do we do?
  - We assume ϕ and try to prove a contradiction. If succeed, we prove $\neg\phi$.
- Here is the proof rule:

$$\frac{\boxed{\begin{array}{c}\phi \\ \vdots \\ \bot\end{array}}}{\neg\phi}\ \neg i$$

Example

Example

Prove $p \implies q, p \implies \neg q \vdash \neg p$.

Proof.

1	$p \implies q$	premise
2	$p \implies \neg q$	premise
3	p	assumption
4	q	$\implies e$ 3, 1
5	$\neg q$	$\implies e$ 3, 2
6	$\bot$	$\neg e$ 4, 5
7	$\neg p$	$\neg i$ 3-6

□

Example

Example

Prove $p \land \neg q \implies r, \neg r, p \vdash q$.

Proof.

1	$p \land \neg q \implies r$	premise	
2	$\neg r$	premise	
3	p	premise	
4	$\neg q$	assumption	⌉
5	$p \land \neg q$	$\land i$ 3, 4	|
6	r	$\implies e$ 5, 1	|
7	$\bot$	$\neg e$ 6, 2	⌋
8	$\neg\neg q$	$\neg i$ 4-7	
9	q	$\neg\neg e$ 8	

□

Derived Rules

- Some rules can actually be derived from others.

Examples

Prove $p \implies q, \neg q \vdash \neg p$ (modus tollens).

Proof.

1	$p \implies q$	premise	
2	$\neg q$	premise	
3	p	assumption	⌉
4	q	$\implies e$ 3, 1	|
5	$\bot$	$\neg e$ 4, 2	⌋
6	$\neg p$	$\neg i$ 3-5	

□

Derived Rules

Examples

Prove $p \vdash \neg\neg p$ $(\neg\neg i)$

Proof.

1	p	premise	
2	$\neg p$	assumption	⌉
3	$\bot$	$\neg e$ 1, 2	⌋
4	$\neg\neg p$	$\neg i$ 2-3	

□

- These rules can be replaced by their proofs and are not necessary.
  - They are just macros to help us write shorter proofs.

Reductio ad absurdum (RAA)

Example

Prove $\neg p \implies \bot \vdash p$ (RAA).

Proof.

1	$\neg p \implies \bot$	premise	
2	$\neg p$	assumption	⌉
3	$\bot$	$\implies e$ 2, 1	⌋
4	$\neg\neg p$	$\neg i$ 2-3	
5	p	$\neg\neg e$ 4	

□

Tertium non datur, Law of the Excluded Middle (LEM)

Example

Prove $\vdash p \vee \neg p$.

Proof.

1	$\neg(p \vee \neg p)$	assumption	
2	p	assumption	
3	$p \vee \neg p$	$\vee i_1$ 2	
4	$\bot$	$\neg e$ 3, 1	
5	$\neg p$	$\neg i$ 2-4	
6	$p \vee \neg p$	$\vee i_2$ 5	
7	$\bot$	$\neg e$ 6, 1	
8	$\neg\neg(p \vee \neg p)$	$\neg i$ 1-7	
9	$p \vee \neg p$	$\neg\neg e$ 8	

□

Proof Rules for Natural Deduction (Summary)

Conjunction ($\wedge$)

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

Disjunction ($\vee$)

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

Implication ($\implies$)

$$\frac{\boxed{\begin{array}{c}\phi \\ \vdots \\ \psi\end{array}}}{\phi \implies \psi} \implies i \qquad \frac{\phi \quad \phi \implies \psi}{\psi} \implies e$$

Proof Rules for Natural Deduction (Summary)

Negation ($\neg$)

$$\frac{\boxed{\begin{array}{c}\phi\\ \vdots\\ \bot\end{array}}}{\neg\phi}\ \neg i \qquad\qquad \frac{\phi \quad \neg\phi}{\bot}\ \neg e$$

Contradiction ($\bot$)

(no introduction rule)

$$\frac{\bot}{\phi}\ \bot e$$

Double negation ($\neg\neg$)

(no introduction rule)

$$\frac{\neg\neg\phi}{\phi}\ \neg\neg e$$

Useful Derived Proof Rules

$$\frac{\phi \implies \psi \quad \neg\psi}{\neg\phi}\ MT \qquad \frac{\phi}{\neg\neg\phi}\ \neg\neg i$$

$$\frac{\boxed{\begin{array}{c}\neg\phi \\ \vdots \\ \bot\end{array}}}{\phi}\ RAA \qquad \frac{}{\phi \lor \neg\phi}\ LEM$$

Provable Equivalence

- Recall $p \dashv\vdash q$ means $p \vdash q$ and $q \vdash p$.
- Here are some provably equivalent sentences:

$$
\begin{array}{rcl}
\neg(p \land q) & \dashv\vdash & \neg q \lor \neg p \\
\neg(p \lor q) & \dashv\vdash & \neg q \land \neg p \\
p \implies q & \dashv\vdash & \neg q \implies \neg p \\
p \implies q & \dashv\vdash & \neg p \lor q \\
p \land q \implies p & \dashv\vdash & r \lor \neg r \\
p \land q \implies r & \dashv\vdash & p \implies (q \implies r)
\end{array}
$$

- Try to prove them.

Proof by Contradiction

- Although it is very useful, the proof rule RAA is a bit puzzling.

$$\frac{\boxed{\begin{array}{c}\neg\phi \\ \vdots \\ \bot\end{array}}}{\phi}\ \mathit{RAA}$$

- Instead of proving ϕ directly, the proof rule allows indirect proofs.
  - If $\neg\phi$ leads to a contradiction, then ϕ must hold.
- Note that indirect proofs are not "constructive."
  - We do not show why ϕ holds; we only know $\neg\phi$ is impossible.
- In early 20th century, some logicians and mathematicians chose not to prove indirectly. They are intuitionistic logicians or mathematicians.
- For the same reason, intuitionists also reject

$$\frac{}{\phi \vee \neg\phi}\ \mathit{LEM} \qquad\qquad \frac{\neg\neg\phi}{\phi}\ \neg\neg e$$

Proof by Contradiction

Theorem

There are $a, b \in \mathbb{R} \setminus \mathbb{Q}$ such that $a^b \in \mathbb{Q}$.

Proof.

Let $b = \sqrt{2}$. There are two cases:

- If $b^b \in \mathbb{Q}$, we are done since $\sqrt{2} \in \mathbb{R} \setminus \mathbb{Q}$.
- If $b^b \notin \mathbb{Q}$, choose $a = b^b = \sqrt{2}^{\sqrt{2}}$. Then $a^b = (b^b)^b = \sqrt{2}^{\sqrt{2} \cdot \sqrt{2}} = \sqrt{2}^2 = 2$. Since $\sqrt{2}^{\sqrt{2}}, \sqrt{2} \in \mathbb{R} \setminus \mathbb{Q}$, we are done. □

- An intuitionist would criticize the proof since it does not tell us what a, b give $a^b \in \mathbb{Q}$.
  - We know (a, b) is either $(\sqrt{2}, \sqrt{2})$ or $(\sqrt{2}^{\sqrt{2}}, \sqrt{2})$.

Section 3

Propositional logic as a formal language

Outline

Well-Formedness

Definition

A well-formed formula is constructed by applying the following rules finitely many times:

- atom: Every propositional atom $p, q, r, \ldots$ is a well-formed formula;
- $\neg$: If ϕ is a well-formed formula, so is $(\neg\phi)$;
- $\wedge$: If ϕ and ψ are well-formed formulae, so is $(\phi \wedge \psi)$;
- $\vee$: If ϕ and ψ are well-formed formulae, so is $(\phi \vee \psi)$;
- $\implies$: If ϕ and ψ are well-formed formulae, so is $(\phi \implies \psi)$.

- More compactly, well-formed formulae are defined by the following grammar in Backus Naur form (BNF):

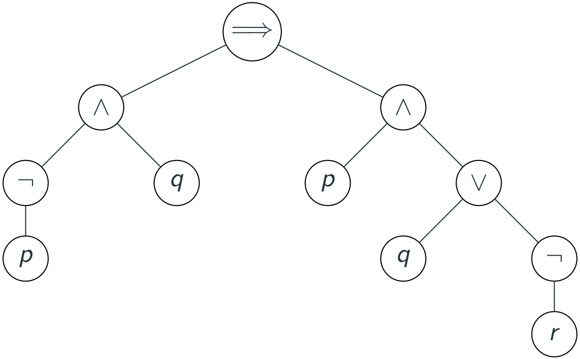
$$\phi ::= p \mid (\neg\phi) \mid (\phi \wedge \phi) \mid (\phi \vee \phi) \mid (\phi \implies \phi)$$

Inversion Principle

- How do we check if $(((\neg p) \wedge q) \implies (p \wedge (q \vee (\neg r))))$ is well-formed?
- Although a well-formed formula needs five grammar rules to construct, the construction process can always be inverted.
  - This is called inversion principle.
- To show $(((\neg p) \wedge q) \implies (p \wedge (q \vee (\neg r))))$ is well-formed, we need to show both $((\neg p) \wedge q)$ and $(p \wedge (q \vee (\neg r)))$ are well-formed.
- To show $((\neg p) \wedge q)$ is well-formed, we need to show both $(\neg p)$ and q are well-formed.
  - q is well-formed since it is an atom.
- To show $(\neg p)$ is well-formed, we need to show p is well-formed.
  - p is well-formed since it is an atom.
- Similarly, we can show $(p \wedge (q \vee (\neg r)))$ is well-formed.

Parse Tree

- The easiest way to decide whether a formula is well-formed is perhaps by drawing its parse tree.



Subformulae

- Given a well-formed formula, its subformulae are the well-formed formulae corresponding to its parse tree.
- For instance, the subformulae of the well-formed formulae $(((\neg p) \wedge q) \implies (p \wedge (q \vee (\neg r))))$ are

$$
\begin{array}{l}
p \\
q \\
r \\
(\neg p) \\
(\neg r) \\
((\neg p) \wedge q) \\
(q \vee (\neg r)) \\
(p \wedge (q \vee (\neg r))) \\
(((\neg p) \wedge q) \implies (p \wedge (q \vee (\neg r))))
\end{array}
$$

Section 4

Semantics of propositional logic

Outline

From $\vdash$ to $\models$

- We have developed a calculus to determine whether $\phi_1, \phi_2, \ldots, \phi_n \vdash \psi$ is valid.
  - That is, from the premises $\phi_1, \phi_2, \ldots, \phi_n$, we can conclude ψ.
  - Our calculus is syntactic. It depends on the syntactic structures of $\phi_1, \phi_2, \ldots, \phi_n$, and ψ.
- We will introduce another relation between premises $\phi_1, \phi_2, \ldots, \phi_n$ and a conclusion ψ.

$$\phi_1, \phi_2, \ldots, \phi_n \models \psi.$$

  - The new relation is defined by 'truth values' of atomic formulae and the semantics of logical connectives.

Truth Values and Models

Definition

The set of truth values is $\{F, T\}$ where F represents 'false' and T represents 'true.'

Definition

A valuation or model of a formula ϕ is an assignment from each proposition atom in ϕ to a truth value.

Truth Values of Formulae

Definition

Given a valuation of a formula ϕ, the truth value of ϕ is defined inductively by the following truth tables:

ϕ	ψ	$\phi \wedge \psi$
F	F	F
F	T	F
T	F	F
T	T	T

ϕ	ψ	$\phi \vee \psi$
F	F	F
F	T	T
T	F	T
T	T	T

ϕ	ψ	$\phi \implies \psi$
F	F	T
F	T	T
T	F	F
T	T	T

ϕ	$\neg\phi$
F	T
T	F

$\top$
T

$\bot$
F

Example

- $\phi \land \psi$ is T when ϕ and ψ are T.
- $\phi \lor \psi$ is T when ϕ or ψ is T.
- $\bot$ is always F; $\top$ is always T.
- $\phi \implies \psi$ is T when ϕ "implies" ψ.

Example

Consider the valuation $\{q \mapsto \mathsf{T}, p \mapsto \mathsf{F}, r \mapsto \mathsf{F}\}$ of $(q \land p) \implies r$. What is the truth value of $(q \land p) \implies r$?

Proof.

Since the truth values of q and p are T and F respectively, the truth value of $q \land p$ is F. Moreover, the truth value of r is F. The truth value of $(q \land p) \implies r$ is T. □

Truth Tables for Formulae

- Given a formula ϕ with propositional atoms $p_1, p_2, \ldots, p_n$, we can construct a truth table for ϕ by listing 2^n valuations of ϕ.

Example

Find the truth table for $(p \implies \neg q) \implies (q \vee \neg p)$.

Proof.

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

□

Outline

Validity of Sequent Revisited

- Informally $\phi_1, \phi_2, \ldots, \phi_n \vdash \psi$ is valid if we can derive ψ with assumptions $\phi_1, \phi_2, \ldots, \phi_n$.
  - We have formalized "deriving ψ with assumptions $\phi_1, \phi_2, \ldots, \phi_n$" by "constructing a proof in a formal calculus."
- We can give another interpretation by valuations and truth values.
- Consider a valuation ν over all propositional atoms in $\phi_1, \phi_2, \ldots, \phi_n, \psi$.
  - By "assumptions $\phi_1, \phi_2, \ldots, \phi_n$," we mean "$\phi_1, \phi_2, \ldots, \phi_n$ are T under the valuation ν.
  - By "deriving ψ,", we mean ψ is also T under the valuation ν.
- Hence, "we can derive ψ with assumptions $\phi_1, \phi_2, \ldots, \phi_n$" actually means "if $\phi_1, \phi_2, \ldots, \phi_n$ are T under a valuation, then ψ must be T under the same valuation.

Semantic Entailment

Definition

We say

$$\phi_1, \phi_2, \ldots, \phi_n \models \psi$$

holds if for every valuations where $\phi_1, \phi_2, \ldots, \phi_n$ are T, ψ is also T. In this case, we also say $\phi_1, \phi_2, \ldots, \phi_n$ semantically entail ψ.

- Examples
  - $p \land q \models p$. For every valuation where $p \land q$ is T, p must be T. Hence $p \land q \models p$.
  - $p \lor q \not\models q$. Consider the valuation $\{p \mapsto \mathrm{T}, q \mapsto \mathrm{F}\}$. We have $p \lor q$ is T but q is F. Hence $p \lor q \not\models q$.
  - $\neg p, p \lor q \models q$. Consider any valuation where $\neg p$ and $p \lor q$ are T. Since $\neg p$ is T, p must be F under the valuation. Since p is F and $p \lor q$ is T, q must be T under the valuation. Hence $\neg p, p \lor q \models q$.
- The validity of $\phi_1, \phi_2, \ldots, \phi_n \vdash \psi$ is defined by syntactic calculus. $\phi_1, \phi_2, \ldots, \phi_n \models \psi$ is defined by truth tables. Do these two relations coincide?

Soundness Theorem for Propositional Logic

Theorem (Soundness)

Let $\phi_1, \phi_2, \ldots, \phi_n$ and ψ be propositional logic formulae. If $\phi_1, \phi_2, \ldots, \phi_n \vdash \psi$ is valid, then $\phi_1, \phi_2, \ldots, \phi_n \models \psi$ holds.

Proof.

Consider the assertion $M(k)$:

"For all sequents $\phi_1, \phi_2, \ldots, \phi_n \vdash \psi (n \geq 0)$ that have a proof of length k, then $\phi_1, \phi_2, \ldots, \phi_n \models \psi$ holds."

$k = 1$. The only possible proof is of the form

1	ϕ	premise

This is the proof of $\phi \vdash \phi$. For every valuation such that ϕ is T, ϕ must be T. Thus, $\phi \models \phi$.

Soundness Theorem for Propositional Logic

Proof (cont'd).

Assume $M(i)$ for $i < k$. Consider a proof of the form

1	ϕ_1	premise
	$\dots$	
n	ϕ_n	premise
	$\dots$	
k	ψ	justification

We have the following possible cases for justification:

i $\wedge i$. Then ψ is $\psi_1 \wedge \psi_2$. In order to apply $\wedge i$, ψ_1 and ψ_2 must appear in the proof. That is, we have $\phi_1, \phi_2, \dots, \phi_n \vdash \psi_1$ and $\phi_1, \phi_2, \dots, \phi_n \vdash \psi_2$. By inductive hypothesis, $\phi_1, \phi_2, \dots, \phi_n \vDash \psi_1$ and $\phi_1, \phi_2, \dots, \phi_n \vDash \psi_2$. Hence $\phi_1, \phi_2, \dots, \phi_n \vDash \psi_1 \wedge \psi_2$ (Why?).

Soundness Theorem for Propositional Logic

Proof (cont'd).

ii $\vee e$. Recall the proof rule for $\vee e$:

$$\frac{\eta_1 \vee \eta_2 \qquad \boxed{\begin{array}{c}\eta_1\\ \vdots \\ \psi\end{array}} \qquad \boxed{\begin{array}{c}\eta_2\\ \vdots \\ \psi\end{array}}}{\psi}\ \vee e$$

In order to apply $\vee e$, $\eta_1 \vee \eta_2$ must appear in the proof. We have $\phi_1, \phi_2, \ldots, \phi_n \vdash \eta_1 \vee \eta_2$. By turning "assumptions" η_1 and η_2 to "premises," we obtain proofs for $\phi_1, \phi_2, \ldots, \phi_n, \eta_1 \vdash \psi$ and $\phi_1, \phi_2, \ldots, \phi_n, \eta_2 \vdash \psi$. By inductive hypothesis, $\phi_1, \phi_2, \ldots, \phi_n \models \eta_1 \vee \eta_2$, $\phi_1, \phi_2, \ldots, \phi_n, \eta_1 \models \psi$, and $\phi_1, \phi_2, \ldots, \phi_n, \eta_2 \models \psi$. Consider any valuation such that $\phi_1, \phi_2, \ldots, \phi_n$ evaluates to T. $\eta_1 \vee \eta_2$ must be T. If η_1 is T under the valuation, ψ is also T (Why?). Similarly for η_2 is T. Thus $\phi_1, \phi_2, \ldots, \phi_n \models \psi$.

Soundness Theorem for Propositional Logic

Proof (cont'd).

iii Other cases are similar. Prove the case of $\Longrightarrow e$ to see if you understand the proof.

□

- The soundness theorem shows that our calculus does not go wrong.
- If there is a proof of a sequent, then the conclusion must be true for all valuations where all premises are true.
- The theorem also allows us to show the non-existence of proofs.
- Given a sequent $\phi_1, \phi_2, \ldots, \phi_n \vdash \psi$, how do we prove there is no proof for the sequent?
  - Try to find a valuation where $\phi_1, \phi_2, \ldots, \phi_n$ are T but ψ is F.

Outline

Completeness Theorem for Propositional Logic

- "$\phi_1, \phi_2, \ldots, \phi_n \vdash \psi$ is valid" and "$\phi_1, \phi_2, \ldots, \phi_n \models \psi$ holds" are very different.
  - "$\phi_1, \phi_2, \ldots, \phi_n \vdash \psi$ is valid" requires proof search (syntax);
  - "$\phi_1, \phi_2, \ldots, \phi_n \models \psi$ holds" requires a truth table (semantics).
- If "$\phi_1, \phi_2, \ldots, \phi_n \models \psi$ holds" implies "$\phi_1, \phi_2, \ldots, \phi_n \vdash \psi$ is valid," then our natural deduction proof system is complete.
- The natural deduction proof system is both sound and complete. That is

  $\phi_1, \phi_2, \ldots, \phi_n \vdash \psi$ is valid iff $\phi_1, \phi_2, \ldots, \phi_n \models \psi$ holds.

Completeness Theorem for Propositional Logic

- We will show the natural deduction proof system is complete.
- That is, if $\phi_1, \phi_2, \ldots, \phi_n \models \psi$ holds, then there is a natural deduction proof for the sequent $\phi_1, \phi_2, \ldots, \phi_n \vdash \psi$.
- Assume $\phi_1, \phi_2, \ldots, \phi_n \models \psi$. We proceed in three steps:
  1. $\models \phi_1 \implies (\phi_2 \implies (\ldots(\phi_n \implies \psi)))$ holds;
  2. $\vdash \phi_1 \implies (\phi_2 \implies (\ldots(\phi_n \implies \psi)))$ is valid;
  3. $\phi_1, \phi_2, \ldots, \phi_n \vdash \psi$ is valid.

Completeness Theorem for Propositional Logic (Step 1)

Lemma

$\phi_1, \phi_2, \ldots, \phi_n \models \psi$ *holds iff* $\models \phi_1 \implies (\phi_2 \implies (\ldots(\phi_n \implies \psi)))$ *holds.*

Proof.

Suppose $\models \phi_1 \implies (\phi_2 \implies \cdots \implies (\phi_n \implies \psi))$ holds. If ϕ_1, ϕ_2, ..., ϕ_n evaluate to T under any valuation, ψ must also evaluate to T since $\models \phi_1 \implies (\phi_2 \implies \cdots \implies (\phi_n \implies \psi))$. Hence $\phi_1, \phi_2, \ldots, \phi_n \models \psi$.

Suppose $\models \phi_1 \implies (\phi_2 \implies (\ldots(\phi_n \implies \psi)))$ does not hold. Then there is valuation where $\phi_1, \phi_2, \ldots, \phi_n$ is T but ψ is F. Contradiction. □

Definition

Let ϕ be a propositional logic formula. We say ϕ is a tautology if $\models \phi$.

- A tautology is a propositional logic formula that evaluates to T for all of its valuations.

Completeness Theorem for Propositional Logic (Step 2)

- Our goal is to show the following theorem:

Theorem

If $\models \eta$ holds, then $\vdash \eta$ is valid.

- Similar to tautologies, we introduce the following definition:

Definition

Let ϕ be a propositional logic formula. We say ϕ is a theorem if $\vdash \phi$.

- Two types of theorems:
  - If $\vdash \phi$, ϕ is a theorem proved by the natural deduction proof system.
  - The soundness theorem for propositional logic is another type of theorem proved by mathematical reasoning (less formally).

Completeness Theorem for Propositional Logic (Step 2)

Proposition

Let ϕ be a formula with propositional atoms $p_1, p_2, \ldots, p_n$. Let l be a line in ϕ's truth table. For all $1 \leq i \leq n$, let $\hat{p}_i$ be p_i if p_i is T in l; otherwise $\hat{p}_i$ is $\neg p_i$. Then

1. $\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_n \vdash \phi$ *is valid if the entry for ϕ at l is T;*
2. $\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_n \vdash \neg\phi$ *is valid if the entry for ϕ at l is F.*

Proof.

We prove by induction on the height of the parse tree of ϕ.

- ϕ is a propositional atom p. Then $p \vdash p$ or $\neg p \vdash \neg p$ have one-line proof.
- ϕ is $\neg\phi_1$.
  - If ϕ is T at l. Then ϕ_1 is F. By IH, $\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_n \vdash \neg\phi_1 (\equiv \phi)$.
  - If ϕ is F at l. Then ϕ_1 is T. By IH, $\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_n \vdash \phi_1$. Using $\neg\neg i$, we have $\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_n \vdash \neg\neg\phi_1 (\equiv \neg\phi)$.

Completeness Theorem for Propositional Logic (Step 2)

Proof (cont'd).

- ϕ is $\phi_1 \implies \phi_2$.
  - If ϕ is F at l, then ϕ_1 is T and ϕ_2 is F at l. By IH, $\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_n \vdash \phi_1$ and $\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_n \vdash \neg\phi_2$. Consider

1	$\phi_1 \implies \phi_2$	assumption
	$\vdots$	
i	ϕ_1	IH
i + 1	ϕ_2	$\implies e$ i, 1
	$\vdots$	
j	$\neg\phi_2$	IH
j + 1	$\bot$	$\neg$ e i+1, j
j + 2	$\neg(\phi_1 \implies \phi_2)$	$\neg$ i 1-(j+1)

Completeness Theorem for Propositional Logic (Step 2)

Proof (cont'd).

- ϕ is $\phi_1 \implies \phi_2$.
  - If ϕ is T at l, we have three subcases. Consider the case where ϕ_1 and ϕ_2 are F at l. Then

1	ϕ_1	assumption	⌉
	$\vdots$		|
i	$\neg\phi_1$	IH	|
i + 1	$\bot$	$\neg$ e 1, i	|
i + 2	ϕ_2	$\bot$ e (i+1)	⌋
i + 3	$\phi_1 \implies \phi_2$	$\implies$ i 1-(i+2)	

  The other two subcases are simple exercises.

Completeness Theorem for Propositional Logic (Step 2)

Proof (cont'd).

- ϕ is $\phi_1 \land \phi_2$.
  - If ϕ is T at l, then ϕ_1 and ϕ_2 are T at l. By IH, we have $\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_n \vdash \phi_1$ and $\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_n \vdash \phi_2$. Using $\land$ i, we have $\hat{p}_1, \hat{p}_2, \ldots, \hat{p}_n \vdash \phi_1 \land \phi_2$.
  - If ϕ is F at l, there are three subcases. Consider the subcase where ϕ_1 and ϕ_2 are F at l. Then

1	$\phi_1 \land \phi_2$	assumption
2	ϕ_1	$\land$ e$_1$ 1
	$\vdots$	
i	$\neg\phi_1$	IH
i + 1	$\bot$	$\neg$ e 2, i
i + 2	$\neg(\phi_1 \land \phi_2)$	$\neg$ i 1-(i+1)

    The other two subcases are simple exercises.

Completeness Theorem for Propositional Logic (Step 2)

Proof.

- ϕ is $\phi_1 \vee \phi_2$.
  - If ϕ is F at I, then ϕ_1 and ϕ_2 are F at I. Then

1	$\phi_1 \vee \phi_2$	assumption
2	ϕ_1	assumption
	$\cdots$	
i	$\neg\phi_1$	IH
i + 1	$\bot$	$\neg$ e 2, i
i + 2	ϕ_2	assumption
	$\cdots$	
j	$\neg\phi_2$	IH
j + 1	$\bot$	$\neg$ e i+2, j
j + 2	$\bot$	$\vee$ e 2-(i+1), (i+2)-(j+1)
j + 3	$\neg(\phi_1 \vee \phi_2)$	$\neg$i 1-(j+2)

  - If ϕ is T at I, there are three subcases. All of them are simple exercises. □

Completeness Theorem for Propositional Logic (Step 2)

Theorem

If ϕ is a tautology, then ϕ is a theorem.

Proof.

Let ϕ have propositional atoms $p_1, p_2, \ldots, p_n$. Since ϕ is a tautology, each line in ϕ's truth table is T. By the above proposition, we have the following 2^n proofs for ϕ:

$$\begin{array}{rcl}
\neg p_1, \neg p_2, \ldots, \neg p_n & \vdash & \phi \\
p_1, \neg p_2, \ldots, \neg p_n & \vdash & \phi \\
\neg p_1, p_2, \ldots, \neg p_n & \vdash & \phi \\
& \vdots & \\
p_1, p_2, \ldots, p_n & \vdash & \phi
\end{array}$$

We apply the rule LEM and the $\vee$e rule to obtain a proof for $\vdash \phi$ (see the next example). □

Completeness Theorem for Propositional Logic (Step 2)

Example

Observe that $\models p \implies (q \implies p)$. Prove $\vdash p \implies (q \implies p)$.

Proof.

1	$p \vee \neg p$	LEM
2	p	assumption
3	$q \vee \neg q$	LEM
4	q	assumption
	$\ldots$	
i	$p \implies (q \implies p)$	$p, q \vdash p \implies (q \implies p)$
i + 1	$\neg q$	assumption
	$\ldots$	
j	$p \implies (q \implies p)$	$p, \neg q \vdash p \implies (q \implies p)$
j + 1	$p \implies (q \implies p)$	$\vee$e 3, 4-i, (i+1)-j
j + 2	$\neg p$	assumption
j + 3	$q \vee \neg q$	LEM
j + 4	q	assumption
	$\ldots$	
k	$p \implies (q \implies p)$	$\neg p, q \vdash p \implies (q \implies p)$
k + 1	$\neg q$	assumption
	$\ldots$	
l	$p \implies (q \implies p)$	$\neg p, \neg q \vdash p \implies (q \implies p)$
l + 1	$p \implies (q \implies p)$	$\vee$e (j+3), (j+4)-k, (k+1)-l
l + 2	$p \implies (q \implies p)$	$\vee$e 1, 2-(j+1), (j+2)-(l+1)

□

Completeness Theorem for Propositional Logic (Step 3)

Lemma

If $\phi_1 \implies (\phi_2 \implies (\cdots(\phi_n \implies \psi)))$ *is a theorem, then* $\phi_1, \phi_2, \ldots, \phi_n \vdash \psi$ *is valid.*

Proof.

Consider

1	ϕ_1	premise
2	ϕ_2	premise
	$\cdots$	
n	ϕ_n	premise
	$\cdots$	
i	$\phi_1 \implies (\phi_2 \implies (\cdots(\phi_n \implies \psi)))$	theorem
i + 1	$\phi_2 \implies (\cdots(\phi_n \implies \psi))$	$\implies$ e 1, i
i + 2	$\phi_3 \implies (\cdots(\phi_n \implies \psi))$	$\implies$ e 2, (i+1)
	$\cdots$	
i + n - 1	$\phi_n \implies \psi$	$\implies$ e (n-1), (i+n-2)
i + n	ψ	$\implies$ e n, (i+n-1)

□

Section 5

Normal Forms

Outline

Semantically Equivalence and Validity

- Consider two formulae $\phi_1 \wedge \phi_2$ and $\phi_2 \wedge \phi_1$.
- Intuitively, $\phi_1 \wedge \phi_2$ and $\phi_2 \wedge \phi_1$ should have the same "meaning."
- More formally, two formulae ϕ and ψ have the same meaning if their truth tables coincide.

Definition

Let ϕ and ψ be propositional logic formulae. ϕ and ψ are semantically equivalent (written $\phi \equiv \psi$) if both $\phi \models \psi$ and $\psi \models \phi$ hold.

- Examples:
$$p \implies q \equiv \neg q \implies \neg p \qquad p \implies q \equiv \neg p \vee q$$
$$p \wedge q \implies p \equiv r \vee \neg r \qquad p \wedge q \implies r \equiv p \implies (q \implies r)$$
- A formula ϕ is valid if it is a tautology.

Definition

Let ϕ be a propositional logic formula. ϕ is valid if $\models \phi$.

Conjunctive Normal Form (CNF)

Definition

A literal L is either an atom p or its negation $\neg p$. A clause D is a disjunction of literals. A formula C is in conjunctive normal form (CNF) if it is a conjunction of clauses.

$$
\begin{array}{rcl}
L & ::= & p \mid \neg p \\
D & ::= & L \mid L \vee D \\
C & ::= & D \mid D \wedge C
\end{array}
$$

- Examples: $(\neg q \vee p \vee r) \wedge (\neg p \vee r) \wedge q$, $(p \vee r) \wedge (\neg p \vee r) \wedge (p \vee \neg r)$

Validity of CNF Formulae

Lemma

A clause $L_1 \vee L_2 \vee \cdots \vee L_m$ is valid iff there is a propositional atom p such that L_i is p and L_j is $\neg p$ for some $1 \leq i, j \leq m$.

Proof.

Without loss of generality, assume $L_1 = p$ and $L_2 = \neg p$. Then $p \vee \neg p \vee L_3 \vee \cdots \vee L_m$ evaluates to T for any valuation. The clause is valid.
Conversely, consider the valuation where all literals evaluate to F. This is possible since every literal L_i has no negation in the clause. The clause evaluates to F under the valuation. □

- Examples:
  - $p \vee q \vee q \vee \neg p \vee r$ is valid;
  - $p \vee \neg q \vee r \vee \neg q$ is not valid (consider $\{p \mapsto \mathrm{F}, q \mapsto \mathrm{T}, r \mapsto \mathrm{F}\}$).
- The validity of any propositional logic formula ϕ can be checked in linear time.

From Truth Tables to Conjunctive Normal Form

- Suppose we have the truth table for a formula ϕ with propositional atoms $p_1, p_2, \ldots, p_n$.
- For each line l where ϕ evaluates to F, construct a clause ψ_l as follows.
  - $\psi_l = L_{l,1} \vee L_{l,2} \vee \cdots \vee L_{l,n}$ where $L_{l,j} = \neg p_j$ if p_j is T at line l; otherwise $L_{l,j} = p_j$.
- Then $\phi \equiv \psi_1 \wedge \psi_2 \wedge \cdots \psi_m$ where ψ_l's are contructed for every line evaluating ϕ to F.
- Observe that $\psi_1 \wedge \psi_2 \wedge \cdots \psi_m$ is F iff ψ_l is F for some $1 \leq l \leq m$. $\psi_l = L_{l,1} \vee L_{l,2} \vee \cdots \vee L_{l,n}$ is F iff $L_{l,j}$ is F for every $1 \leq j \leq n$. $L_{l,j}$ is F iff p_j has its truth value at line l.
- In other words, $\psi_1 \wedge \psi_2 \wedge \cdots \psi_m$ is F under a valuation iff the valuation evaluates ϕ to F in ϕ's truth table.

From Truth Tables to Conjunctive Normal Form

Example

Translate $p \vee q \implies q \wedge \neg r$ into CNF.

Proof.

p	q	r	$p \vee q \implies q \wedge \neg r$	p	q	r	$p \vee q \implies q \wedge \neg r$
F	F	F	T	T	F	F	F
F	F	T	T	T	F	T	F
F	T	F	T	T	T	F	T
F	T	T	F	T	T	T	F

p	q	r	ψ_l	p	q	r	ψ_l
F	T	T	$p \vee \neg q \vee \neg r$	T	F	F	$\neg p \vee q \vee r$
T	F	T	$\neg p \vee q \vee \neg r$	T	T	T	$\neg p \vee \neg q \vee \neg r$

$p \vee q \implies q \wedge \neg r \equiv (p \vee \neg q \vee \neg r) \wedge (\neg p \vee q \vee r) \wedge (\neg p \vee q \vee \neg r) \wedge (\neg p \vee \neg q \vee \neg r)$. □

Outline

Validity Checking

- Given a propositional logic formula in conjunctive normal form, we can check the validity of the formula in linear time.
- Recall that a formula is valid iff it is a theorem.
- If we can translate any propositional logic formula into conjunctive normal form, we can check the validity of the formula!
- We know how to translate any logic formula to conjunctive normal form by its truth table.
  - This is not satisfactory. If we have to construct its truth table, we can check validity already.
- We will give an algorithm $\mathtt{CNF}(\phi)$ to convert any propositional logic formula into conjunctive normal form without building its truth table.

From Formula to Conjunctive Normal Form

- Any propositional logic formula can be transformed to conjunctive normal form by the following equivalences:

$$\phi \implies \psi \equiv \neg\phi \vee \psi$$

$$\neg(\phi \wedge \psi) \equiv \neg\phi \vee \neg\psi \qquad \neg(\phi \vee \psi) \equiv \neg\phi \wedge \neg\psi$$

$$\phi \wedge (\psi_1 \vee \psi_2) \equiv (\phi \wedge \psi_1) \vee (\phi \wedge \psi_2)$$

$$\phi \vee (\psi_1 \wedge \psi_2) \equiv (\phi \vee \psi_1) \wedge (\phi \vee \psi_2)$$

- The algorithm `CNF`(ϕ) hence consists of three steps:
  - Remove every implication ($\implies$) from ϕ (Algorithm `IMPL_FREE`(ϕ));
  - Push every negation ($\neg$) to literals (Algorithm `NNF`(ϕ));
  - Apply law of distribution (Algorithm `PUSHDISJ`(ϕ)).

Algorithm IMPL_FREE(ϕ)

Input: ϕ : a logic formula
Output: ϕ' : all implications ($\implies$) in ϕ' are removed and $\phi' \equiv \phi$
switch ϕ **do**
- **case** ϕ is a literal: **do return** ϕ;
- **case** ϕ is $\neg\phi_1$: **do return** $\neg$IMPL_FREE(ϕ_1);
- **case** ϕ is $\phi_1 \wedge \phi_2$: **do return** IMPL_FREE(ϕ_1) $\wedge$ IMPL_FREE(ϕ_2);
- **case** ϕ is $\phi_1 \vee \phi_2$: **do return** IMPL_FREE(ϕ_1) $\vee$ IMPL_FREE(ϕ_2);
- **case** ϕ is $\phi_1 \implies \phi_2$: **do return** IMPL_FREE($\neg\phi_1 \vee \phi_2$);
- **otherwise do** assert(0);

Algorithm 1: IMPL_FREE(ϕ)

Algorithm $\texttt{NNF}(\phi)$

Input: ϕ : a logic formula without implication ($\implies$)
Output: ϕ' : only propositional atoms in ϕ' are negated and $\phi' \equiv \phi$
switch ϕ **do**
- **case** ϕ is a literal: **do return** ϕ;
- **case** ϕ is $\neg\neg\phi_1$: **do return** $\texttt{NNF}(\phi_1)$;
- **case** ϕ is $\phi_1 \land \phi_2$: **do return** $\texttt{NNF}(\phi_1) \land \texttt{NNF}(\phi_2)$;
- **case** ϕ is $\phi_1 \lor \phi_2$: **do return** $\texttt{NNF}(\phi_1) \lor \texttt{NNF}(\phi_2)$;
- **case** ϕ is $\neg(\phi_1 \land \phi_2)$: **do return** $\texttt{NNF}(\neg\phi_1 \lor \neg\phi_2)$;
- **case** ϕ is $\neg(\phi_1 \lor \phi_2)$: **do return** $\texttt{NNF}(\neg\phi_1 \land \neg\phi_2)$;
- **otherwise do** $\texttt{assert}(0)$;

Algorithm 2: $\texttt{NNF}(\phi)$

Definition

If only propositional atoms in ϕ are negated, ϕ is in negation normal form.

Algorithm $\texttt{PUSHDISJ}(\phi)$

Input: ϕ : an NNF formula without implication ($\implies$)
Output: ϕ' : ϕ' is in CNF and $\phi' \equiv \phi$
switch ϕ **do**
  case ϕ is a literal: **do return** ϕ;
  case ϕ is $\phi_1 \wedge \phi_2$: **do return** $\texttt{PUSHDISJ}(\phi_1) \wedge \texttt{PUSHDISJ}(\phi_2)$;
  case ϕ is $\phi_1 \vee \phi_2$: **do return** $\texttt{DISTR}(\texttt{PUSHDISJ}(\phi_1), \texttt{PUSHDISJ}(\phi_2))$;

Algorithm 3: $\texttt{PUSHDISJ}(\phi)$

Input: η_1, η_2 : η_1, η_2 are in CNF
Output: ϕ' : ϕ' is in CNF and $\phi' \equiv \eta_1 \vee \eta_2$
if η_1 is $\eta_{11} \wedge \eta_{12}$ **then** **return** $\texttt{DISTR}(\eta_{11}, \eta_2) \wedge \texttt{DISTR}(\eta_{12}, \eta_2)$;
else if η_2 is $\eta_{21} \wedge \eta_{22}$ **then** **return** $\texttt{DISTR}(\eta_1, \eta_{21}) \wedge \texttt{DISTR}(\eta_1, \eta_{22})$;
else **return** $\eta_1 \vee \eta_2$;

Algorithm 4: $\texttt{DISTR}(\eta_1, \eta_2)$

Checking Validity Revisited

- Let ϕ be a propositional logic formula. Consider the following algorithm for checking its validity.
  1. Compute a CNF formula $\psi = \texttt{CNF}(\phi)$.
  2. Check the validity of ψ.

Validity and Satisfiability

Definition

Let ϕ be a propositional logic formula. ϕ is satisfiable if it evaluates to T under some valuation.

- Example: $p \vee q \implies p$ is satisfiable (consider $\{p \mapsto \mathrm{T}, q \mapsto \mathrm{T}\}$); it is not valid (consider $\{p \mapsto \mathrm{F}, q \mapsto \mathrm{T}\}$).

Proposition

Let ϕ be a propositional logic formula. ϕ is satisfiable iff $\neg\phi$ is not valid.

Proof.

Suppose ϕ evaluates to T under a valuation. Then $\neg\phi$ evaluates to F under the valuation. $\neg\phi$ is not valid.

Conversely, suppose $\neg\phi$ is not valid. Hence $\neg\phi$ evaluates to F under a valuation. Thus ϕ evaluates to T under the valuation. ϕ is satisfiable. □

Satisfiability of Propositional Logic Formulae

- Let ϕ be a propositional logic formula. Consider the following algorithm for checking its satisfiability.
  1. Compute a CNF formula $\psi = \mathtt{CNF}(\neg\phi)$.
  2. Check the validity of ψ.
  3. Return "ϕ is satisfiable" if ψ is not valid; Return "ϕ is not satisfiable" if ψ is valid.
- Recall that satisfiability of propositional logic formulae is an NP-complete problem.
- Is the above algorithm in polynomial time? Why?

Outline

Horn Clauses

- Given a propositional logic formula in CNF, it is easy to check its validity; it is "hard" to check its satisfiability.
- We will consider a subclass of CNF formulae whose satisfiability can be checked efficiently.

Definition

A Horn formula is a propositional logic formula ϕ of the following form:

$$\begin{array}{rcl} P & ::= & \bot \mid \top \mid p \\ A & ::= & P \mid P \land A \\ C & ::= & A \implies P \\ H & ::= & C \mid C \land H. \end{array}$$

A clause of the form C is called a Horn clause.

- Example: $(p \land q \land s \implies \bot) \land (q \land r \implies p) \land (\top \implies s)$
- Nonexample: $(p \land \neg q \land s \implies \bot) \land (q \land r \implies p \land s) \land (p \lor r \implies s)$

Satisfiability of Horn Formulae

- Consider a Horn clause $P_1 \wedge P_2 \wedge \cdots \wedge P_n \implies Q$.
- If $P_1, P_2, \ldots, P_n$ are assigned to T, then Q must be T; otherwise, Q can be an arbitrary truth value.
- We hence have the following (informal) algorithm:
  1. Mark $\top$ if it occurs in the Horn formula ϕ;
  2. If there is a Horn clause $P_1 \wedge P_2 \wedge \cdots \wedge P_n \implies Q$ in ϕ such that all P_j for $1 \leq j \leq n$ are marked, mark Q;
  3. If $\bot$ is marked, print "The Horn formula ϕ is unsatisfiable."
  4. Print "The Horn formula ϕ is satisfiable."

Algorithm Horn(ϕ)

Input: ϕ: ϕ is a Horn formula
Output: "unsatisfiable" if ϕ is unsatisfiable; otherwise "satisfiable."
mark all occurrences of $\top$ in ϕ;
while there is a Horn clause $P_1 \wedge P_2 \wedge \cdots \wedge P_n \implies Q$ in ϕ such that P_j are all marked but Q is not **do**
    mark Q;
if $\bot$ is marked **then** **return** "unsatisfiable" ;
else **return** "satisfiable" ;

Algorithm 5: Horn(ϕ)

Satisfiability of Horn Formulae

Theorem

Let ϕ be a Horn formula with n propositional atoms. $\texttt{Horn}(\phi)$ *runs at most $n + 1$ iterations and decides the satisfiability of ϕ correctly.*

Proof.

At each iteration, an unmarked atom will be marked. Since there are n atoms, there are at most $n + 1$ iterations.

By induction on the number of iterations, we show that "all marked P are true for all valuations where ϕ evaluates to T." At iteration 0 (before entering the loop), only $\top$ are marked. Clearly, $\top$ must be true for any valuation. At iteration $k + 1$, consider a Horn clause $P_1 \land P_2 \land \cdots \land P_n \implies Q$ where $P_1, P_2, \ldots, P_n$ are marked but not Q. For a valuation ν where ϕ evaluates to T, $P_1 \land P_2 \land \cdots \land P_n \implies Q$ must evaluate to T. Since $P_1, P_2, \ldots, P_n$ are true in ν (by IH), Q must be true in ν.

Satisfiability of Horn Formulae

Proof (cont'd).

We now prove $\texttt{Horn}(\phi)$ answers correctly. When $\texttt{Horn}(\phi)$ returns "unsatisfiable," there is a Horn clause $P_1 \wedge P_2 \wedge \cdots \wedge P_n \implies \bot$ where $P_1, P_2, \ldots, P_n$ are all marked. Suppose ν is a valuation where ϕ evaluates to T. Then $P_1, P_2, \ldots, P_n$ must be true in ν. Hence $P_1 \wedge P_2 \wedge \cdots \wedge P_n \implies \bot$ evaluates to F. ϕ cannot evaluate to T under ν. A contradiction. When $\texttt{Horn}(\phi)$ returns "satisfiable," define a valuation ν where all marked propositional atoms are assigned to T and all unmarked atoms are F. We claim ϕ evaluates to T in ν. Suppose not. There is a Horn clause $P_1 \wedge P_2 \wedge \cdots P_n \implies Q$ in ϕ which evaluates to F under ν. That is, $P_1, P_2, \ldots, P_n$ are T but Q is F under ν. By the definition of ν, $P_1, P_2, \ldots, P_n$ are marked by the algorithm. Hence Q must also be marked by the algorithm. Q cannot be F in ν. A contradiction. □

Section 6

SAT Solvers

Outline

SAT Solvers

- The satisfiability problem for propositional logic is to decide whether a propositional logic formula ϕ is satisfiable.
  - The satisfiability problem for propositional logic is an NP-complete problem (Cook's Theorem).
- Many SAT solvers are available for the problem.
- We will discuss algorithms for the satisfiability problem.
- For this topic, most materials are copied or modified from Prof. Sharad Malik's and Prof. Chung-Yang Huang's lecture notes.

Equisatisfiable Propositional Logic Formulae

- Let ϕ and ψ be propositional logic formulae.
- ϕ and ψ are equisatisfiable if

$$\phi \text{ is satisfiable if and only if } \psi \text{ is satisfiable.}$$

- What is the difference between semantic equivalence and equisatisfiability?

Tseitin Transformation

Theorem

For every propositional logic formula ϕ, there is a propositional logic formula ψ in CNF such that ϕ and ψ are equisatisfiable.

Proof.

For every subformula α of ϕ, x_α is p if α is the atomic proposition p; x_α is P_α otherwise. For every non-atomic subformula α, define C_α as follows.

α	C_α	Remark
$\neg\beta$	$(x_\alpha \lor x_\beta) \land (\neg x_\alpha \lor \neg x_\beta)$	$x_\alpha \Leftrightarrow \neg x_\beta$
$\beta_0 \lor \beta_1$	$(x_\alpha \lor \neg x_{\beta_0}) \land (x_\alpha \lor \neg x_{\beta_1}) \land (\neg x_\alpha \lor x_{\beta_0} \lor x_{\beta_1})$	$x_\alpha \Leftrightarrow x_{\beta_0} \lor x_{\beta_1}$
$\beta_0 \land \beta_1$	$(\neg x_\alpha \lor x_{\beta_0}) \land (\neg x_\alpha \lor x_{\beta_1}) \land (x_\alpha \lor \neg x_{\beta_0} \lor \neg x_{\beta_1})$	$x_\alpha \Leftrightarrow x_{\beta_0} \land x_{\beta_1}$

Let $\psi = x_\phi \land \bigwedge\{C_\alpha : \alpha \text{ is a non-atomic subformula of } \phi\}$. ϕ and ψ are equisatisfiable. □

Example

Example

Transform $\phi = \neg(p \land \neg q)$ into an equisatisfiable propositional logic formula in CNF.

Proof.

$\neg q, p \land \neg q, \phi$ are non-atomic subformulae of ϕ.

$$\begin{aligned}
C_{\neg q} &\triangleq (P_{\neg q} \lor q) \land (\neg P_{\neg q} \lor \neg q) \\
C_{p \land \neg q} &\triangleq (\neg P_{p \land \neg q} \lor p) \land (\neg P_{p \land \neg q} \lor P_{\neg q}) \land (P_{p \land \neg q} \lor \neg p \lor \neg P_{\neg q}) \\
C_{\phi} &\triangleq (P_{\phi} \lor P_{p \land \neg q}) \land (\neg P_{\phi} \lor \neg P_{p \land \neg q})
\end{aligned}$$

ϕ and $P_{\phi} \land C_{\phi} \land C_{p \land \neg q} \land C_{\neg q} = P_{\phi} \land (P_{\phi} \lor P_{p \land \neg q}) \land (\neg P_{\phi} \lor \neg P_{p \land \neg q}) \land (\neg P_{p \land \neg q} \lor p) \land (\neg P_{p \land \neg q} \lor P_{\neg q}) \land (P_{p \land \neg q} \lor \neg p \lor \neg P_{\neg q}) \land (P_{\neg q} \lor q) \land (\neg P_{\neg q} \lor \neg q)$ are equisatisfiable. □

Example

Example

Transform $\phi = p \lor \neg(q \land (r \lor \neg s))$ into an equisatisfiable propositional logic formula in CNF.

Proof.

$\neg s, r \lor \neg s, q \land (r \lor \neg s), \neg(q \land (r \lor \neg s)), \phi$ are non-atomic subformulae of ϕ.

$$
\begin{array}{rcl}
C_{\neg s} & \triangleq & (P_{\neg s} \lor s) \land (\neg P_{\neg s} \lor \neg s) \\
C_{r \lor \neg s} & \triangleq & (P_{r \lor \neg s} \lor \neg r) \land (P_{r \lor \neg s} \lor \neg P_{\neg s}) \land (\neg P_{r \lor \neg s} \lor r \lor P_{\neg s}) \\
C_{q \land (r \lor \neg s)} & \triangleq & (\neg P_{q \land (r \lor \neg s)} \lor q) \land (\neg P_{q \land (r \lor \neg s)} \lor P_{r \lor \neg s}) \land \\
 & & (P_{q \land (r \lor \neg s)} \lor \neg q \lor \neg P_{r \lor \neg s}) \\
C_{\neg(q \land (r \lor \neg s))} & \triangleq & (P_{\neg(q \land (r \lor \neg s))} \lor P_{q \land (r \lor \neg s)}) \land (\neg P_{\neg(q \land (r \lor \neg s))} \lor \neg P_{q \land (r \lor \neg s)}) \\
C_{\phi} & \triangleq & (P_{\phi} \lor \neg p) \land (P_{\phi} \lor \neg P_{\neg(q \land (r \lor \neg s))}) \land (\neg P_{\phi} \lor p \lor P_{\neg(q \land (r \lor \neg s))})
\end{array}
$$

Then ϕ and $P_{\phi} \land C_{\phi} \land C_{\neg(q \land (r \lor \neg s))} \land C_{q \land (r \lor \neg s)} \land C_{r \lor \neg s} \land C_{\neg s}$ is equisatisfiable. □

Properties of Tseitin Transformation

- Let ϕ be a propositional logic formula.
- The size $|\phi|$ of ϕ is the number of symbols (atomic propositions, $\wedge$, $\vee$, $\neg$) in ϕ.
  - Parentheses ("(" and ")") do not count.
  - $|\phi|$ is the number of nodes in the parsing tree of ϕ.
- Tseitin Transformation of ϕ has
  - $|\phi|$ atomic propositions;
  - Each C_α has at most 3 clauses;
  - Each clause of C_α has at most 3 literals.
- Let $\phi = (p_{11} \wedge p_{12} \wedge \cdots p_{1m_1}) \vee (p_{21} \wedge p_{22} \wedge \cdots p_{2m_2}) \vee \cdots \vee (p_{n1} \wedge p_{n2} \wedge \cdots p_{nm_n})$. We obtain a semantic equivalent propositional logic formula ψ by distributive law. What is the size of ψ?

DIMACS SAT Format i

- DIMACS SAT format is a standard text format for CNF formulae.
- Most SAT solvers accept a simplified version of DIMACS SAT format.
- Here is the example from the SAT Competition home page.

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

DIMACS SAT Format ii

- From the SAT Competition home page:
  - The file can start with comments, that is lines begining with the character `c`.
  - Right after the comments, there is the line `p cnf` *nbvar nbclauses* indicating that the instance is in CNF format; *nbvar* is the exact number of variables appearing in the file; *nbclauses* is the exact number of clauses contained in the file.
  - Then the clauses follow. Each clause is a sequence of distinct non-null numbers between -*nbvar* and *nbvar* ending with `0` on the same line; it cannot contain the opposite literals i and $-i$ simultaneously. Positive numbers denote the corresponding variables. Negative numbers denote the negations of the corresponding variables.
- Following DIMACS SAT format, we will represent a propositional logic formula in CNF by a set of clauses.
  - $(\neg a \vee b \vee \neg c) \wedge (\neg b \vee d) \wedge (a \vee c \vee \neg d)$ is represented as

$$(\neg a \vee b \vee \neg c) \quad (\neg b \vee d) \quad (a \vee c \vee \neg d)$$

SAT Algorithms

- Davis, Putnam, 1960
  - Explicit resolution based.
  - May explode in memory.
- Davis, Logemann, Loveland, (DLL) 1962
  - Search based.
  - Most successful, basis for almost all modern SAT solvers.
  - Learning and non-chronological backtracking, 1996.
- Stålmarcks algorithm, 1980s
  - Proprietary algorithm. Patented.
  - Commercial versions available
- Stochastic Methods, 1992
  - Unable to prove unsatisfiability, but may find solutions for a satisfying problem quickly.
  - Local search and hill climbing

Resolution

- Consider the proof rule:

$$\frac{l_0 \lor l_1 \lor \cdots \lor l_m \lor k \quad \overline{k} \lor l'_0 \lor l'_1 \lor \cdots \lor l'_n}{l_0 \lor l_1 \lor \cdots \lor l_m \lor l'_0 \lor l'_1 \lor \cdots \lor l'_n} \textit{ resolution}$$

  - k and $\overline{k}$ are complementary literals.
  - We assume $\lor$ is commutative.
- For instance, we obtain $(a \lor \neg b \lor \neg d \lor f)$ from $(a \lor \neg b \lor \neg c)$ and $(\neg d \lor c \lor f)$ by resolution.

Davis Putnam Algorithm

- Select a variable for resolution iteratively.
- Consider the following set of clauses:

$$\begin{array}{lll} (a \vee b \vee c) & (b \vee \neg c \vee f) & (\neg b \vee e) \\ \hline (a \vee c \vee e) & (\neg c \vee e \vee f) & \\ \hline (a \vee e \vee f) & & \end{array}$$

SAT!

- Consider the following set of clauses:

$$\begin{array}{cccc} (a \vee b) & (a \vee \neg b) & (\neg a \vee c) & (\neg a \vee \neg c) \\ \hline (a) & (\neg a \vee c) & (\neg a \vee \neg c) & \\ \hline (c) & (\neg c) & & \\ \hline () & & & \end{array}$$

UNSAT!

Basic DLL i

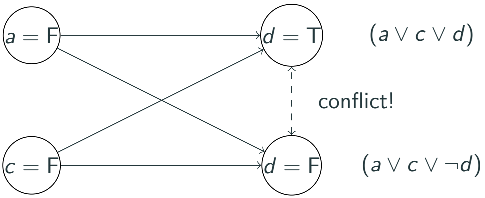
- Consider the following set of clauses:

$$(\neg a \vee b \vee c) \quad (a \vee c \vee d) \quad (a \vee c \vee \neg d) \quad (a \vee \neg c \vee d)$$
$$(a \vee \neg c \vee \neg d) \quad (\neg b \vee \neg c \vee d) \quad (\neg a \vee b \vee \neg c) \quad (\neg a \vee \neg b \vee c)$$

- We perform depth first search:

$$a = \mathrm{F} \quad b = \mathrm{F} \quad c = \mathrm{F}$$

and obtain a conflict:



  - This is an implication graph.

Basic DLL ii

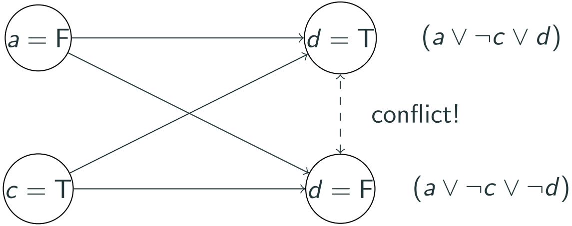
- Consider the following set of clauses:

$$\begin{array}{cccc} (\neg a \vee b \vee c) & (a \vee c \vee d) & (a \vee c \vee \neg d) & (a \vee \neg c \vee d) \\ (a \vee \neg c \vee \neg d) & (\neg b \vee \neg c \vee d) & (\neg a \vee b \vee \neg c) & (\neg a \vee \neg b \vee c) \end{array}$$

- Backtrack!

$$\begin{array}{lll} a = \mathrm{F} & b = \mathrm{F} & \cancel{c = \mathrm{F}} \\ & & c = \mathrm{T}(\text{forced}) \end{array}$$

and obtain a conflict:



Basic DLL iii

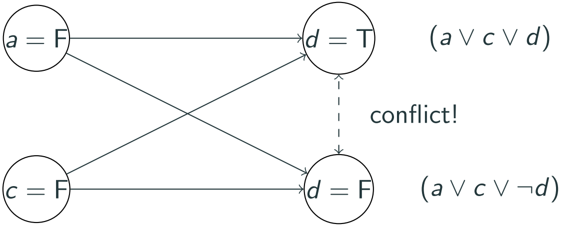
- Consider the following set of clauses:

$$(\neg a \vee b \vee c) \quad (a \vee c \vee d) \quad (a \vee c \vee \neg d) \quad (a \vee \neg c \vee d)$$
$$(a \vee \neg c \vee \neg d) \quad (\neg b \vee \neg c \vee d) \quad (\neg a \vee b \vee \neg c) \quad (\neg a \vee \neg b \vee c)$$

- Backtrack!

$$a = \mathrm{F} \quad \sout{b = \mathrm{F}}$$
$$b = \mathrm{T}(\text{forced}) \quad c = \mathrm{F}$$

and obtain a conflict:



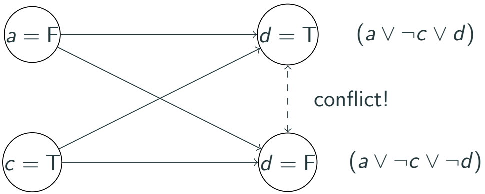
- Consider the following set of clauses:

$$(\neg a \vee b \vee c) \quad (a \vee c \vee d) \quad (a \vee c \vee \neg d) \quad (a \vee \neg c \vee d)$$
$$(a \vee \neg c \vee \neg d) \quad (\neg b \vee \neg c \vee d) \quad (\neg a \vee b \vee \neg c) \quad (\neg a \vee \neg b \vee c)$$

- Backtrack!

$a = \mathrm{F}$ ~~$b = \mathrm{F}$~~
$b = \mathrm{T}(\text{forced})$ ~~$c = \mathrm{F}$~~
$c = \mathrm{T}(\text{forced})$

and obtain a conflict:



Basic DLL v

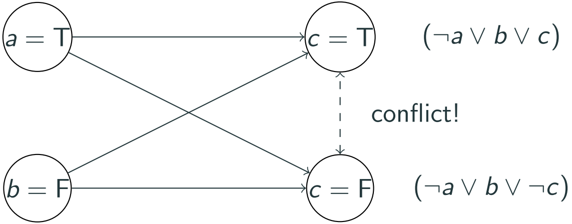
- Consider the following set of clauses:

$$
\begin{array}{cccc}
(\neg a \vee b \vee c) & (a \vee c \vee d) & (a \vee c \vee \neg d) & (a \vee \neg c \vee d) \\
(a \vee \neg c \vee \neg d) & (\neg b \vee \neg c \vee d) & (\neg a \vee b \vee \neg c) & (\neg a \vee \neg b \vee c)
\end{array}
$$

- Backtrack!

$$
\begin{array}{ll}
\cancel{a = \mathrm{F}} & \\
a = \mathrm{T}(\text{forced}) & b = \mathrm{F}
\end{array}
$$

and obtain a conflict:



Basic DLL vi

- Consider the following set of clauses:

$$(\neg a \lor b \lor c) \quad (a \lor c \lor d) \quad (a \lor c \lor \neg d) \quad (a \lor \neg c \lor d)$$
$$(a \lor \neg c \lor \neg d) \quad (\neg b \lor \neg c \lor d) \quad (\neg a \lor b \lor \neg c) \quad (\neg a \lor \neg b \lor c)$$

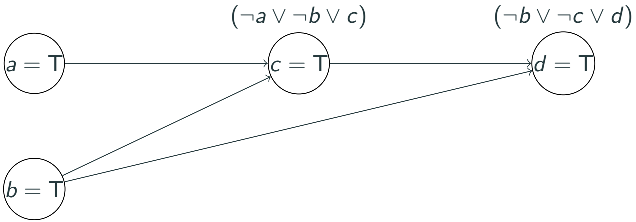
- Backtrack!

~~$a = \mathrm{F}$~~

$a = \mathrm{T}(\text{forced})$ ~~$b = \mathrm{F}$~~

$b = \mathrm{T}(\text{forced})$

and SAT!

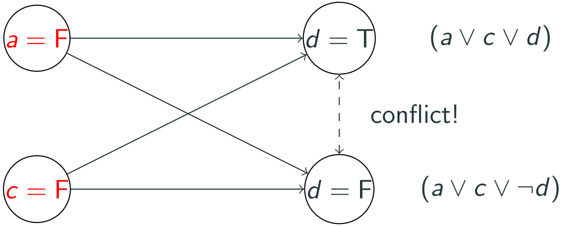


Conflict Driven Learning

- When a conflict is encountered, add its cause to prevent it.
- Consider the following set of clauses:

$$\begin{array}{cccc} (\neg a \vee b \vee c) & (a \vee c \vee d) & (a \vee c \vee \neg d) & (a \vee \neg c \vee d) \\ (a \vee \neg c \vee \neg d) & (\neg b \vee \neg c \vee d) & (\neg a \vee b \vee \neg c) & (\neg a \vee \neg b \vee c) \end{array}$$

  and the following valuation: $a = \mathrm{F} \quad b = \mathrm{F} \quad c = \mathrm{F}$
- The conflict is caused by $a = \mathrm{F}$ and $c = \mathrm{F}$:



- Hence we add a learned clause: $(a \vee c)$.

Non-Chronological Backtracking

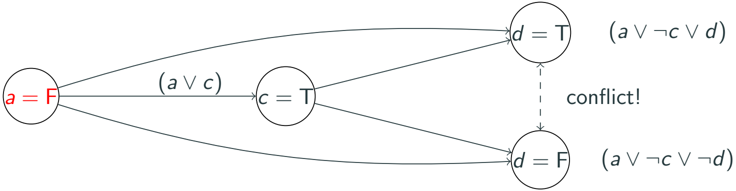
- When a clause is learned, backtrack to the next-to-the-last variable in the clause.
- Consider the following set of clauses:

$$(\neg a \vee b \vee c) \quad (a \vee c \vee d) \quad (a \vee c \vee \neg d) \quad (a \vee \neg c \vee d)$$
$$(a \vee \neg c \vee \neg d) \quad (\neg b \vee \neg c \vee d) \quad (\neg a \vee b \vee \neg c) \quad (\neg a \vee \neg b \vee c)$$
$$(a \vee c)$$

backtrack to a:

$$a = \mathrm{F}$$

obtain a conflict, and add a learned clause (a):



DPLL+CDCL i

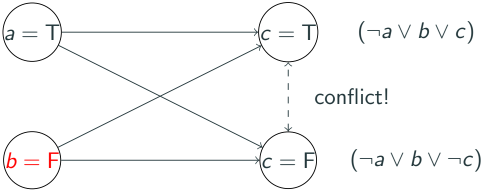
- Consider the following set of clauses:

$$
\begin{array}{cccc}
(\neg a \vee b \vee c) & (a \vee c \vee d) & (a \vee c \vee \neg d) & (a \vee \neg c \vee d) \\
(a \vee \neg c \vee \neg d) & (\neg b \vee \neg c \vee d) & (\neg a \vee b \vee \neg c) & (\neg a \vee \neg b \vee c) \\
(a \vee c) & (a) & &
\end{array}
$$

- backtrack all variables:

$$b = \mathrm{F}$$

obtain a conflict, and add a learned clause (b):

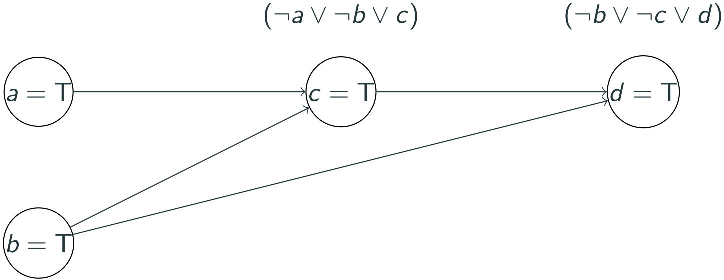


DPLL+CDCL ii

- Consider the following set of clauses:

$$(\neg a \vee b \vee c) \quad (a \vee c \vee d) \quad (a \vee c \vee \neg d) \quad (a \vee \neg c \vee d)$$
$$(a \vee \neg c \vee \neg d) \quad (\neg b \vee \neg c \vee d) \quad (\neg a \vee b \vee \neg c) \quad (\neg a \vee \neg b \vee c)$$
$$(a \vee c) \quad (a) \quad (b)$$

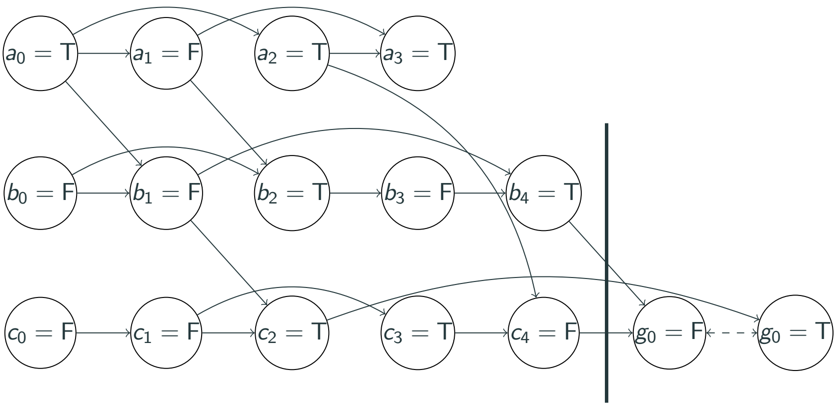
- and SAT!



Conflict Analysis i

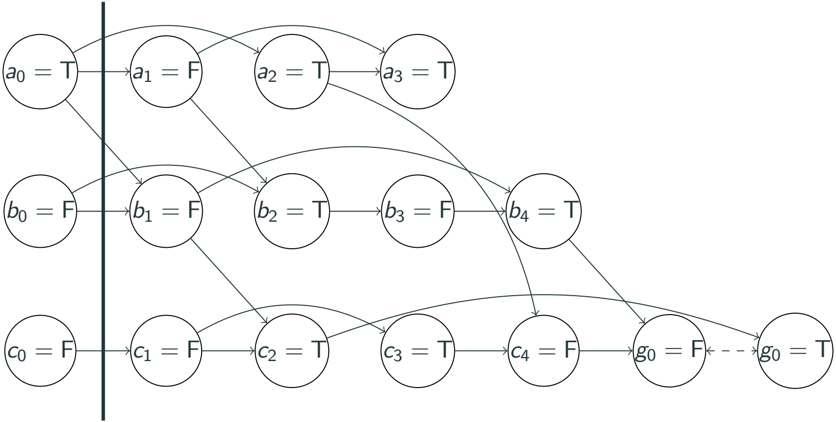
- When a conflict occurs, we would like to “learn” a clause to prevent the conflict from reoccurring.
    - CDCL = Conflict-Driven Clause Learning
- We hence would like to know what valuations cause the conflict.
- In an implication graph, any cut from root assignments to conflicting assignments is such a cause.

Conflict Analysis ii



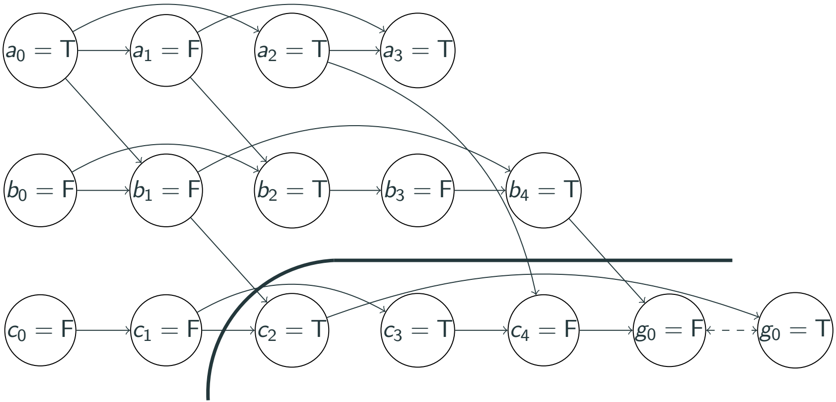
- $b_4 = \mathrm{T}$, $c_2 = \mathrm{T}$, and $c_4 = \mathrm{F}$ cause the conflict.
- Hence we can learn $(\neg b_4 \vee \neg c_2 \vee c_4)$.

Conflict Analysis iii



- $a_0 = \mathrm{T}$, $b_0 = \mathrm{F}$, and $c_0 = \mathrm{F}$ cause the conflict.
- Hence we can learn $(\neg a_0 \vee b_0 \vee c_0)$.

Conflict Analysis iv

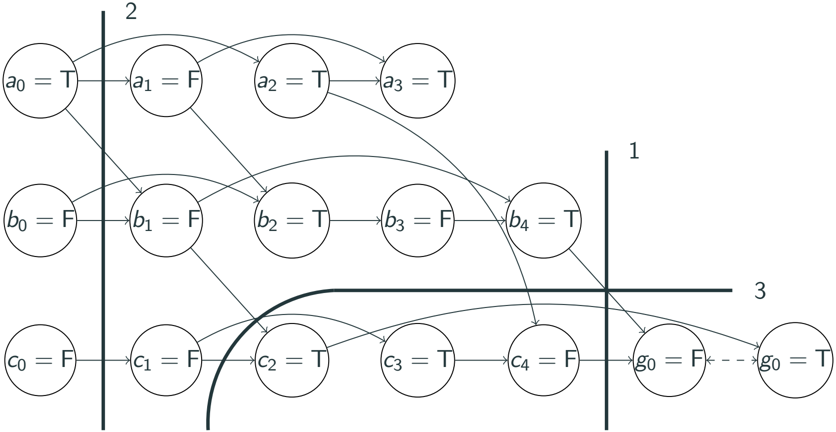


- $a_2 = T$, $b_1 = F$, $b_4 = T$, and $c_1 = F$ cause the conflict.
- Hence we can learn $(\neg a_2 \lor b_1 \lor \neg b_4 \lor c_1)$.

Conflict Analysis v

- Clearly, there are several cuts in an implication graph.
- Each cut corresponds to a learned clause.
- Which one is the best?
- Idea: reverse exactly one truth value.
- Consider only cuts that has only one node in the same level as the conflict.
  - This is called a unique implication point (UIP).

Conflict Analysis vi



- Cut 1 is not UIP. Cut 2 is the last UIP. Cut 3 is the first UIP.
  - Emperically, the first UIP is the best.

Where to go?

- Contrary to general belief, SAT solvers are not impractical.
- Engineering ideas are essential to the real world.
  - When cleverly applied, they can even tackle hard theoretical problems such as satisfiability.
- We only discuss a clever idea in SAT solvers very briefly.
- There are certainly many more.
  - There are many interesting ideas for propagation, memory management, etc.
- MiniSAT is an open-sourced fast SAT solver.
- Read its code. You will learn a lot more!