

# CS228 Logic for Computer Science 2023

## Lecture 7: Conjunctive normal form and Resolution

Instructor: Ashutosh Gupta

IITB, India

Compile date: 2023-01-24

# Normal forms

- ▶ Grammar of propositional logic is too complex.
- ▶ If one builds a tool, one will prefer to handle fewer connectives and simpler structure
- ▶ We transform given formulas into normal forms before handling them.

We will look at the following two normal forms

- ▶ Negation normal form (seen in the previous lecture)
- ▶ Conjunctive normal forms

**Commentary:** Building a software for handling formulas with the complexity is undesirable. We aim to reduce the complexity by applying transformations to obtain a normalized form. The normalization results in standardization and interoperability of tool.

## Removing $\oplus$ , $\Rightarrow$ , and $\Leftrightarrow$ .

Please note the following equivalences that remove  $\oplus$ ,  $\Rightarrow$ , and  $\Leftrightarrow$  from a formula.

- ▶  $(p \Rightarrow q) \equiv (\neg p \vee q)$
- ▶  $(p \oplus q) \equiv (p \vee q) \wedge (\neg p \vee \neg q)$
- ▶  $(p \Leftrightarrow q) \equiv \neg(p \oplus q)$

For the ease of presentation, we will assume you can remove them at will.

**Commentary:** Removing  $\Rightarrow$  is common and desirable. The removal of  $\oplus$  and  $\Leftrightarrow$ , however, blows up the formula size. Their straight up removal is not desirable. We can avoid the blow up in some contexts. However, in our presentation we will skip the issue.

## Topic 7.1

### Conjunctive normal form

## Some terminology

- ▶ Propositional variables are also referred as **atoms**
- ▶ A **literal** is either an atom or its negation
- ▶ A **clause** is a disjunction of literals.

Since  $\vee$  is associative, commutative, and absorbs multiple occurrences, a clause may be referred as a set of literals.

### Example 7.1

- ▶  *$p$  is an atom but  $\neg p$  is not.*
- ▶  *$\neg p$  and  $p$  both are literals.*
- ▶  *$p \vee \neg p \vee p \vee q$  is a clause.*
- ▶  *$\{p, \neg p, q\}$  is the same clause.*

# Conjunctive normal form(CNF)

## Definition 7.1

A formula is in **CNF** if it is a conjunction of clauses.

Since  $\wedge$  is associative, commutative and absorbs multiple occurrences, a CNF formula may be referred as a set of clauses

## Example 7.2

- ▶  $\neg p$  and  $p$  both are in CNF.
- ▶  $(p \vee \neg q) \wedge (r \vee \neg q) \wedge \neg r$  in CNF.
- ▶  $\{(p \vee \neg q), (r \vee \neg q), \neg r\}$  is the same CNF formula.
- ▶  $\{\{p, \neg q\}, \{r, \neg q\}, \{\neg r\}\}$  is the same CNF formula.

**Commentary:** A set of formulas is interpreted depending on the context. There is no requirement that we apply conjunction among the elements. A clause is a set of literals. We interpret it as disjunction of literals. A CNF formula is a set of clauses, which is set of sets of literals. We interpret it as conjunction of clauses. The inner part is interpreted as disjunction and the outer set is interpreted as conjunction.

## Exercise 7.1

- Write a formal grammar for CNF
- How can we represent true and false using CNF formulas?

# CNF conversion

## Theorem 7.1

*For every formula  $F$  there is another formula  $F'$  in CNF such that  $F \equiv F'$ .*

## Proof.

Let us suppose we have

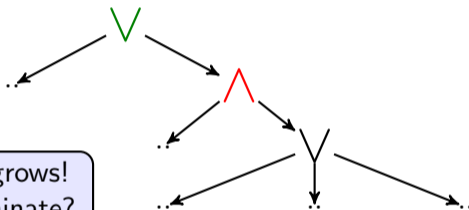
- ▶ removed  $\oplus, \Rightarrow, \Leftrightarrow$  using the standard equivalences,
- ▶ converted the formula in NNF, and
- ▶ flattened  $\wedge$  and  $\vee$ .

...

# CNF conversion (contd.)

## Proof(contd.)

Now the formulas have the following form with literals at leaves.



After the push formula size grows!  
Why should the method terminate?

Since  $\vee$  distributes over  $\wedge$ , we can push  $\vee$  inside  $\wedge$ . Eventually, we obtain a CNF formula.  $\square$

## Example 7.3

Are we done?

*Conversion to CNF*

$$(p \Rightarrow (\neg q \wedge r)) \wedge (p \Rightarrow \neg q) \equiv (\neg p \vee (\neg q \wedge r)) \wedge (\neg p \vee \neg q) \equiv (\neg p \vee \neg q) \wedge (\neg p \vee r) \wedge (\neg p \vee \neg q)$$

**Commentary:** The above is a good example of an algorithm that has intuitively clear but formally non-trivial termination argument. Termination proof is presented at the end of this deck.

# Formal derivation for CNF

## Theorem 7.2

*Let  $F'$  be the CNF of  $F$ . If we have  $\Sigma \vdash F$ , then we can derive  $\Sigma \vdash F'$ .*

## Proof.

We combine the following pieces of proofs for each step of the transformations.

- ▶ Derivations for NNF
- ▶ Derivations for substitutions that remove  $\Rightarrow$ ,  $\oplus$ , and  $\Leftrightarrow$
- ▶ Derivations for substitutions that flatten  $\wedge$  and  $\vee$
- ▶ Derivations for substitutions that apply distributivity

Therefore, we have the derivations. □

## Conjunctive normal form(CNF) more notation

- ▶ A **unit clause** contains only one literal.
- ▶ A **binary clause** contains two literals.
- ▶ A **ternary clause** contains three literals.
- ▶ We extend the definition of clauses to the empty set of literals. Say,  $\perp$  is the empty clause.

For a clause  $C$  and a literal  $\ell$ , we may write  $\ell \cup C$  or  $\ell \vee C$  to denote  $\{\ell\} \cup C$ .

### Example 7.4

- ▶  $(p \wedge q \wedge \neg r)$  has three unit clauses
- ▶  $(p \vee \neg q \vee \neg s) \wedge (p \vee q) \wedge \neg r$  has a ternary, a binary, and a unit clause

### Exercise 7.2

- Give a linear time algorithm to prove validity of a CNF formula
- What is the interpretation of the empty set of clauses?

## Topic 7.2

### Tseitin encoding

# CNF is desirable

- ▶ Fewer connectives
- ▶ Simple structure
- ▶ Many problems naturally encode into CNF.

We will see this in couple of lectures.

## How do we get to CNF?

- ▶ The transformation using distributivity **explodes** the formula
- ▶ Is there a way **to avoid** the explosion?
- ▶ **Yes!** there is a way.

## Tseitin encoding

But, with a **cost**.

## Tseitin encoding : intuition

### Example 7.5

Consider formula  $p \vee (q \wedge r)$ , which is not in CNF.

We replace offending  $(q \wedge r)$  by a fresh  $x$  and add clauses to encode that  $x$  behaves like  $(q \wedge r)$ .

$$(p \vee x) \wedge (x \Rightarrow (q \wedge r))$$

After simplification,

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

### Exercise 7.3

- Ideally, we should have introduced  $(x \Leftrightarrow (q \wedge r))$ . Why is the above with implication correct?
- Show that transformation from  $(F \vee \neg G)$  to  $(F \vee \neg x) \wedge (x \Rightarrow G)$  will not preserve satisfiability.
- Show that transformation from  $(F \vee \neg G)$  to  $(F \vee \neg x) \wedge (G \Rightarrow x)$  preserves satisfiability.

## Tseitin encoding (Plaisted-Greenbaum optimization included)

By introducing fresh variables, Tseitin encoding can translate every formula into an equisatisfiable CNF formula **without** exponential explosion.

1. Assume input formula  $F$  is NNF without  $\oplus$ ,  $\Rightarrow$ , and  $\Leftrightarrow$ .
2. Find a  $G_1 \wedge \dots \wedge G_n$  that is just below an  $\vee$  in  $F(G_1 \wedge \dots \wedge G_n)$
3. Replace  $F(G_1 \wedge \dots \wedge G_n)$  by  $F(p) \wedge \underbrace{(\neg p \vee G_1) \wedge \dots \wedge (\neg p \vee G_n)}_{p \Rightarrow G_1 \wedge \dots \wedge G_n}$ , where  $p$  is a fresh variable
4. goto 2

### Exercise 7.4

*Modify the encoding such that it works without the assumptions at step 1*

**Commentary:** If you read wikipedia about the encoding, you will find that Tseitin encoding adds more clauses. They add clauses for  $p \Leftrightarrow G_1 \wedge \dots \wedge G_n$ . Our translation includes Plaisted-Greenbaum optimization, which drops  $G_1 \wedge \dots \wedge G_n \Rightarrow p$  while preserving satisfiability. Solve the above exercise to understand to see if we do not apply the optimization then we do not need assumption 1. Hint: Download sat solver `$wget http://fmv.jku.at/limboole/limboole1.1.tar.gz` look for function `tseitin` in file `limboole.c`

## Example: linear cost of Tseitin encoding

### Example 7.6

Consider formula  $(p_1 \wedge \dots \wedge p_n) \vee (q_1 \wedge \dots \wedge q_m)$

Using distributivity, we obtain the following CNF containing  $mn$  clauses.

$$\bigwedge_{i \in 1..n, j \in 1..m} (p_i \vee q_j)$$

Using Tseitin encoding, we obtain the following CNF containing  $m + n + 1$  clauses, where  $x$  and  $y$  are the fresh Boolean variables.

$$(x \vee y) \wedge \bigwedge_{i \in 1..n} (\neg x \vee p_i) \wedge \bigwedge_{j \in 1..m} (\neg y \vee q_j)$$

### Exercise 7.5

Give a model to the original formula that is not a model of the transformed formula

## Tseitin encoding preserves satisfiability

Let us prove one direction of the equisatisfiability.

### Theorem 7.3

if  $m \models F(p) \wedge (\neg p \vee G_1) \wedge \cdots \wedge (\neg p \vee G_n)$  then  $m \models F(G_1 \wedge \cdots \wedge G_n)$

### Proof.

Assume  $m \models F(p) \wedge (\neg p \vee G_1) \wedge \cdots \wedge (\neg p \vee G_n)$ . We have three cases.

First case  $m \models p$ :

- ▶ Therefore,  $m \models G_i$  for all  $i \in 1..n$ .
- ▶ Therefore,  $m \models G_1 \wedge \cdots \wedge G_n$ .
- ▶ Due to the substitution theorem,  $m \models F(G_1 \wedge \cdots \wedge G_n)$ .

Second case  $m \not\models p$  and  $m \not\models G_1 \wedge \cdots \wedge G_n$ :

- ▶ Due to the substitution theorem,  $m \models F(G_1 \wedge \cdots \wedge G_n)$

# Tseitin encoding preserves satisfiability(contd.)

## Proof(contd.)

Third case  $m \not\models p$  and  $m \models G_1 \wedge \dots \wedge G_n$ :

- ▶ Since  $F(G_1 \wedge \dots \wedge G_n)$  is in NNF,  $p$  occurs only positively in  $F(p)$ .
- ▶ Let us consider model  $m[p \mapsto 1]$ .
- ▶ Naturally,  $m[p \mapsto 1] \models p$ .
- ▶ Therefore,  $m[p \mapsto 1] \models F(p)$  (why?).
- ▶ Since  $p$  does not occur in  $G_i$ s,  $m[p \mapsto 1] \models G_1 \wedge \dots \wedge G_n$ .
- ▶ Due to the substitution theorem,  $m[p \mapsto 1] \models F(G_1 \wedge \dots \wedge G_n)$ .
- ▶ Therefore,  $m \models F(G_1 \wedge \dots \wedge G_n)$ . □

**Commentary:** We have introduced  $p$ , which is replacing  $G_1 \wedge \dots \wedge G_n$ . Since the formula is in NNF, the negation symbols are only on variables. Therefore, they cannot be above  $G_1 \wedge \dots \wedge G_n$  in  $F(G_1 \wedge \dots \wedge G_n)$ . Therefore,  $p$  occurs positively in  $F(p)$ .

We leave the other direction of equisatisfiability as the following exercise.

## Exercise 7.6

Show if  $\not\models F(p) \wedge (\neg p \vee G_1) \wedge \dots \wedge (\neg p \vee G_n)$  then  $\not\models F(G_1 \wedge \dots \wedge G_n)$

**Commentary:** Hint: The formulas in the above exercise are not equivalent but equisatisfiable. To prove, we need to choose a model  $m$  that satisfies  $F(G_1 \wedge \dots \wedge G_n)$ . Since  $p$  does not occur in  $F(G_1 \wedge \dots \wedge G_n)$ , we should be able to freely change the value of  $p$  in  $m$  without affecting  $m \models F(G_1 \wedge \dots \wedge G_n)$ . We need to choose the value of  $p$  appropriately such that  $m[p \mapsto ?] \models F(p) \wedge (\neg p \vee G_1) \wedge \dots \wedge (\neg p \vee G_n)$ .

## Topic 7.3

### Resolution proof system

## Derivations starting from CNF

We assumed that we have a set of formulas in the lhs, which was treated as conjunction of the formulas.

$$\Sigma \vdash F$$

The conjunction of CNF formulas is also a CNF formula.

If all formulas are CNF, we may **assume  $\Sigma$  as a set of clauses**.

# Derivations from CNF formulas

How many rules do we need?

Answer: We need only two rules

- ▶ derive clauses from the CNF formula

$$\text{ASSUMPTION} \frac{}{\Sigma \vdash C} C \in \Sigma$$

- ▶ derive new clauses using resolution

$$\text{RESOLUTION} \frac{\Sigma \vdash F \vee G \quad \Sigma \vdash \neg F \vee H}{\Sigma \vdash G \vee H}$$

(We derived the above proof rule)

## Resolution proof rule

Typically  $\Sigma$  is clear from the context, so we may not write it explicitly again and again.

Since we are deriving only clauses, we apply resolution rule as follows.

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

- ▶ clauses  $p \vee C$  and  $\neg p \vee D$  are called **antecedents**
- ▶ variable  $p$  is called **pivot**
- ▶ clause  $C \vee D$  is called **resolvent**

# Resolution proof system

Resolution proof method takes a set of clauses  $\Sigma$  and produces a **forest of clauses** as a proof.

Clauses in the proof are either from  $\Sigma$  or consequences of previous clauses.

The aim of the proof method is to find the empty clause, which stands for inconsistency.

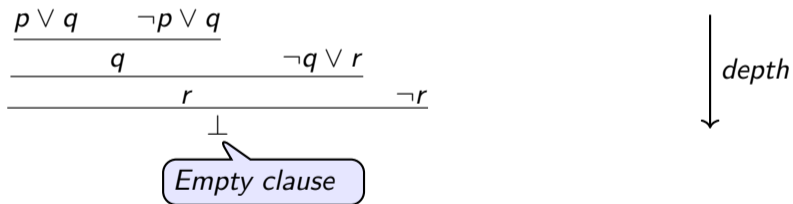
The proof systems that aim to derive false are called **refutation proof systems**.

# Resolution Proofs

## Example 7.7

Consider  $F = (p \vee q) \wedge (\neg p \vee q) \wedge (\neg q \vee r) \wedge \neg r$ ,

We will consider the context of our derivation to be  $\Sigma = \{(p \vee q), (\neg p \vee q), (\neg q \vee r), \neg r\}$



**Wait!** we never derive empty formula in formal proofs. Is it allowed?  
It will make sense in a minute.

## Topic 7.4

Formal proof system vs. resolution proof system

## Clauses as sets in resolution proof system

In formal proofs, we needed to show every small steps for example  $\vee$ -Symm.

The resolution proof system avoids the mundane work by taking the input in CNF, which a clause as a finite set of literals not as a list of literals.

The shortcuts are supported by equivalences learned earlier lectures.

If need be we can always construct a formal proof from a resolution poof.

**Commentary:** Please note that the resolution proof system is not the same as the "formal proof system". They have similarities. They achieve the same goal (demonstrated in the slides) but they play by different rules. In the exams, you will be explicitly asked to write a proof in a given proof system. You have to honor the rules of the proof system.

## Formal proofs and $\perp$

Recall, formal proof system does not refer to  $\perp$ . It encodes  $\perp$  using  $F \wedge \neg F$  for some formula  $F$ .

Observe that just before deriving empty clause we derive  $\Sigma \vdash r$  and  $\Sigma \vdash \neg r$ , for some variable  $r$ .

We translate the last resolution as the following derivation

1.  $\Sigma \vdash \neg r$
2.  $\Sigma \vdash r$
3.  $\Sigma \vdash \neg r \wedge r$  ( $\wedge$ -intro applied to 2 and 1)

### Theorem 7.4

*If resolution proof system can derive  $\Sigma \vdash \perp$ ,  $\Sigma$  is unsatisfiable.*

### Proof.

Since we have proven that formal derivation is sound in lecture 5,  $\Sigma$  is unsatisfiable.  $\square$

## Using resolution to prove statements

Let us suppose we are asked to derive  $\Sigma \vdash F$ .

We assume  $\Sigma$  is **finite**. We will relax this by the next lecture.

We will convert  $\bigwedge \Sigma \wedge \neg F$  into a set of clauses  $\Sigma'$ .

We apply the resolution proof method on  $\Sigma'$ .

If we derive  $\perp$  clause,  $\Sigma \vdash F$  is derivable.

### Exercise 7.7

*Convert the above steps into a formal derivation.*

## Example: using resolution to prove statements

### Example 7.8

Let us suppose we want to show  $\{\neg(\neg r \wedge (s \vee t))\} \vdash (s \Rightarrow r)$  is derivable.

We write negated formula  $\underbrace{\neg(\neg r \wedge (s \vee t))}_{\wedge \Sigma} \wedge \underbrace{\neg(s \Rightarrow r)}_{\neg F}$

We convert the above into a CNF formula.

$$\underbrace{(r \vee \neg s) \wedge (r \vee \neg t)}_{\wedge \Sigma} \wedge \underbrace{s \wedge \neg r}_{\neg F}$$

The following resolution proof shows that the statement is derivable.

$$\begin{array}{c} r \vee \neg s \quad s \\ \hline r \\ \hline \perp \end{array} \quad \neg r$$

We can translate the resolution proof into a formal proof. (Please try)

# Topic 7.5

## Problems

# Convert into CNF

## Exercise 7.8

*Give a CNF formula equivalent to  $p \oplus q \oplus \neg r$ .*

## Exercise: disjunctive normal form(DNF)

### Definition 7.2

A formula is in **DNF** if it is a disjunction of conjunctions of literals.

### Exercise 7.9

- Prove: for every formula  $F$  there is another formula  $F'$  in DNF such that  $F \equiv F'$ .
- Give the formal grammar of DNF
- Give a linear time algorithm to prove satisfiability of a DNF formula

# CNF and DNF

## Exercise 7.10

*Give an example of a non-trivial formula that is both CNF and DNF*

## Exercise 7.11

*Convert the following formulas into CNF with/without introducing fresh variables*

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

# P=NP argument

## Exercise 7.12

*What is wrong with the following proof of  $P=NP$ ? Give counterexample.*

*Tseitin encoding does not explode and proving validity of CNF formulas has a linear time algorithm. Therefore, we can convert every formula into CNF in polynomial time and check validity in linear time. As a consequence, we can check satisfiability of  $F$  in linear time by checking validity of  $\neg F$  in linear time.*

# Validity\*\*

## Exercise 7.13

*Give a procedure like Tseitin encoding that converts a formula into another equi-valid DNF formula. Prove correctness of your transformation.*

## Algebraic normal form(ANF)\*\*

ANF formulas are defined using the following grammar.

$$\begin{aligned} A &::= \top \mid \perp \mid p \\ C &::= A \wedge C \mid A \\ ANF &::= C \oplus ANF \mid C \end{aligned}$$

### Exercise 7.14

- Give an efficient algorithm to covert any formula into equivalent ANF formula.*
- Give an efficient algorithm to covert any formula into equisatisfiable ANF formula.*

## CNF vs. DNF\*\*\*

### Exercise 7.15

*Give a class of Boolean functions that can be represented using linear size DNF formula but can only be represented by an exponential size CNF formula.*

### Exercise 7.16

*Give a class of Boolean functions that can be represented using linear size CNF formula but can only be represented by an exponential size DNF formula.*

# Probability of satisfiability\*\*\*

## Exercise 7.17

- a. What is the probability that the conjunction of a random **multiset** of literals of size  $k$  over  $n$  Boolean variables is unsatisfiable?*
- b. What is the probability that the conjunction of a random **set** of literals of size  $k$  over  $n$  Boolean variables is unsatisfiable?*

## And inverter graphs (AIG)\*\*

AIG formulas are defined using the following grammar.

$$A ::= A \wedge A \mid \neg A \mid p$$

### Exercise 7.18

*Give heuristics to minimize the number of inverters in an AIG formula without increasing the size of the formula.*

**Commentary:** Example of such heuristics: Local Two-Level And-Inverter Graph Minimization without Blowup. Robert Brummayer and Armin Biere, 2006.

# Resolution proof

## Exercise 7.19

*Give resolution proofs of the following formulas.*

1.  $p_{11} \wedge p_{21} \wedge (\neg p_{11} \vee \neg p_{21})$
2.  $(p_{11} \vee p_{12}) \wedge (p_{21} \vee p_{22}) \wedge (p_{31} \vee p_{32}) \wedge (\neg p_{11} \vee \neg p_{21}) \wedge (\neg p_{21} \vee \neg p_{31}) \wedge (\neg p_{31} \vee \neg p_{11}) \wedge (\neg p_{12} \vee \neg p_{22}) \wedge (\neg p_{22} \vee \neg p_{32}) \wedge (\neg p_{32} \vee \neg p_{12})$

**Commentary:** If you can not do it by hand, try using a solver to generate the proof.

## Non-unique resolvents and valid clauses

Between two clauses we may need to choose the pivot to apply the resolution. We may have multiple choices applicable.

### Example 7.9

*The following resolutions are between two clauses, with different pivots*

$$\frac{p \vee q \vee r \quad \neg p \vee \neg q \vee r}{q \vee \neg q \vee r} \qquad \frac{p \vee q \vee r \quad \neg p \vee \neg q \vee r}{p \vee \neg p \vee r}$$

### Exercise 7.20

- There is something wrong with the above resolvents. What is it?*
- If there are multiple choices for resolution, should we do it at all?*
- Prove that if a valid clause (a clause that contains both both  $p$  and  $\neg p$  for some  $p$ ) is generated, we can **ignore** it for any further derivations **without loss of completeness**.*

# Pure literals

## Definition 7.3

*If a literal occurs in a CNF formula and its negation does not then it is a **pure literal**.*

## Exercise 7.21

*Show that the removal of clauses containing the pure literals in a CNF preserves satisfiability.*

# Superset clauses are redundant

## Exercise 7.22

For clauses  $C$  and  $D$ , show that if  $D \subset C$  and  $\perp$  can be derived using  $C$  then it can be derived using  $D$ .

If clause  $C$  is superset of clause  $D$ , then  $C$  is **redundant**.

## Example 7.10

Consider  $\{q, \neg q \vee r, r, \neg r\}$ . We say  $\neg q \vee r$  is redundant because  $r \subset \neg q \vee r$ .

A proof using  $\neg q \vee r$ :

$$\frac{\frac{q \quad \neg q \vee r}{r} \quad \neg r}{\perp}$$

A modified proof using the shorter clause.

$$\frac{r \quad \neg r}{\perp}$$

## Resolution: redundancy in resolution proofs

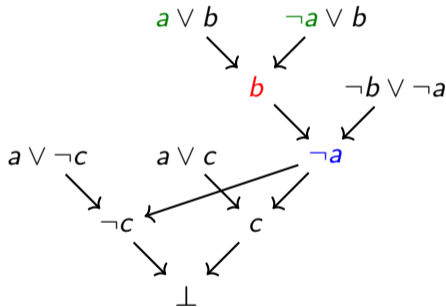
### Exercise 7.23

*Let us suppose we have a resolution proof deriving  $\perp$ . We have discussed that valid clauses should not be used for resolution. However, nobody stops us in producing them and then further using them for the resolution. Let us suppose we have valid clauses occurring somewhere in the middle of our proof. Give a linear (or close to linear) time algorithm in terms of the size of the proof that removes the valid clauses from the proof.*

## More redundancies

### Exercise 7.24

Each resolution removes a single literal. Therefore, if downstream resolutions reintroduce the literal, then purpose the earlier resolution is defeated. For example,



The resolution producing  $b$  is redundant in both the paths to  $\perp$ , because  $\neg a$  was reintroduced.

Therefore, our proof may be unnecessarily large. Give an algorithm to remove the redundancies.

## Topic 7.6

Extra slides: termination of CNF conversion

# CNF conversion terminates

## Theorem 7.5

*The procedure of converting a formula into CNF terminates.*

### Proof.

For a formula  $F$ , let  $\nu(F) \triangleq$  the maximum height of  $\vee$  to  $\wedge$  alternations in  $F$ .

Consider a formula  $F(G)$  such that

$$G = \bigvee_{i=0}^m \bigwedge_{j=0}^{n_i} G_{ij}.$$

After the push we obtain  $F(G')$ , where

$$G' = \bigwedge_{j_1=0}^{n_1} \dots \bigwedge_{j_m=0}^{n_m} \underbrace{\bigvee_{i=0}^m G_{ij_i}}_{\nu(\quad) < \nu(G)}$$

### Observations

- ▶  $G'$  is either the top formula or the parent connective is  $\wedge$
- ▶  $G_{ij}$  is either a literal or an  $\vee$  formula

**Commentary:**  $\vee$  formula means that the top symbol in the formula is  $\vee$

We need to apply flattening to keep  $F(G')$  in the form (like the previous proof).

# CNF conversion terminates (contd.)

(contd.)

Due to König lemma, the procedure terminates. [König lemma slides are at the end.] (why?) □

## Exercise 7.25

*Consider a set of balls that are labelled with positive numbers. We can replace a  $k$  labelled ball with any number of balls with labels less than  $k$ . Using König lemma, show that the process always terminates.*

*Hint: in the above exercise, the bag is the subformulas of  $F(G)$ .*

## Exercise 7.26

*Show  $F'$  obtained from the procedure may be exponentially larger than  $F$ .*

**Commentary:** It is slightly involved to see the application of König lemma on our theorem. We can understand the application via the above exercise. In the above exercise, we are removing balls with large labels and replacing with balls that have smaller labels. This process will eventually hit label 1 for all balls. Once the balls with label 1 are removed, we can not add any more balls. In a similar way in our theorem, we are removing subformulas with larger  $\nu$  and replacing with many subformulas with smaller  $\nu$ . Therefore, the process will terminate. The formal construction is left for the exercise.

# König Lemma

## Theorem 7.6

*For an infinite connected graph  $G$ , if degree of each node is finite then there is an infinite simple path in  $G$  from each node.*

### Proof.

We construct an infinite simple path  $v_1, v_2, v_3, \dots$  as follows.

#### base case:

Choose any  $v_1 \in G$ . Let  $G_1 \triangleq G$ .

#### induction step:

1. Assume path  $v_1, \dots, v_i$  and an infinite connected graph  $G_i$  such that  $v_i \in G_i$  and  $v_1 \dots v_{i-1} \notin G_i$ .
2. In  $G_i$ , there is a neighbour  $v_{i+1} \in G_i$  of  $v_i$  such that infinite nodes are reachable from  $v_{i+1}$  without visiting  $v_i$ . (why?)
3. Let  $S$  be the reachable nodes. Let  $G_{i+1} \triangleq G_i|_S$ . □

## Exercise 7.27

*Prove that any finitely-branching infinite tree must have an infinite branch.*

## Topic 7.7

Extra slides: Implementation issues in resolution

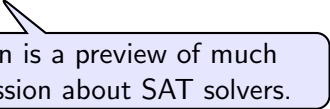
# Efficient implementation of a proof method

A proof method implicitly defines a non-deterministic proof search algorithm

In implementing such an algorithm, one needs to ensure that one is not doing unnecessary work.

Now we only worry about a single rule. We may be more effective in finding the proof strategy.

We will discuss some simple observations that may cut huge search spaces.



This discussion is a preview of much detailed discussion about SAT solvers.

End of Lecture 7