CS228 Logic for Computer Science 2020

Lecture 9: Resolution

Instructor: Ashutosh Gupta

IITB, India

Compile date: 2020-02-05

Topic 9.1

Resolution

Clauses as sets and CNF formulas as set of sets

Definition 9.1 (clause redefined)

A clause is a finite set of literals $\{\ell_1, \ldots, \ell_n\}$ *and interpreted as* $\ell_1 \vee .. \vee \ell_n$.

For a clause C and a literal ℓ, we will write $\ell \cup C$ to denote $\{\ell\} \cup C$.

Definition 9.2 (CNF formula redefined)

A CNF formula is a finite set of clauses $\{C_1, \ldots, C_n\}$ *and interpreted as* $C_1 \wedge .. \wedge C_n$.

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 Σ 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

$$\textsc{Assumption}\frac{}{\Sigma \vdash C} C \in \Sigma$$

- derive new clauses using resolution

$$\textsc{Resoluton}\frac{\Sigma \vdash F \vee G \qquad \Sigma \vdash \neg F \vee H}{\Sigma \vdash G \vee H}$$

(We derived the above proof rule)

Resolution proof rule

Typically Σ 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.

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

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

Attendance quiz

Exercise 9.1

Which of the following are correct resolution derivations? Let / be the derivation line.

$(p), (\neg p)/(\bot)$; $(\neg p), (p)/(\bot)$; $(p \vee q), (\neg p)/q$; $(\neg p \vee q), (p)/q$;
$(p \vee q \vee \neg r), (\neg p)/(q \vee \neg r)$; $(\neg p \vee q \vee \neg r), (p)/(q \vee \neg r)$;
$(p \vee \neg p), (\neg p \vee q)/(\neg p \vee q)$; $(p \vee q), (\neg p \vee q)/(q)$;
$(p \vee q), (\neg p \vee \neg q)/(\neg q \vee q)$;

$(p), (p)/(\bot)$; $(\neg p), (p)/(p)$; $(p \vee q), (\neg p)/\neg q$; $(\neg pp \vee q), (p)/\bot$;
$(p \vee q \vee \neg r), (\neg p)/(q \vee r)$; $(\neg p \vee q \vee \neg r), (p)/(\neg r)$; $(p \vee \neg p), (\neg p \vee q)/(q)$;
$(p \vee q), (\neg p \vee q)/(\neg q)$; $(p \vee q), (\neg p \vee \neg q)/(\neg q \vee p)$;

Non-unique resolvents

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

Example 9.1

The following resolutions are between two clauses, with different pivots

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

Exercise 9.2

a. There is something wrong with the above resolvents. What is it?

b. If there are multiple choices for resolution, should we do it at all?

Resolution proof method

Resolution proof method takes a set of clauses Σ and produces a forest of clauses as a proof.

Clauses in the proof are either from Σ or consequences of previous clauses.

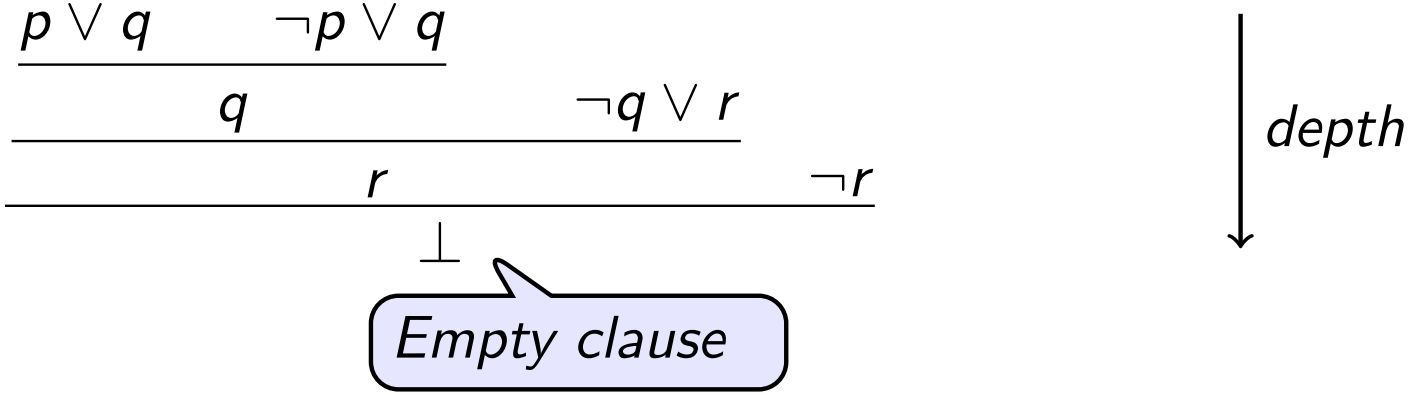
The goal of the proof method is to find the empty clause, which stands for inconsistency.

Resolution Proofs

Example 9.2

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

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.

Formal proofs and $\bot$

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

We may observe that just before deriving empty clause we have to 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 9.1

If resolution proof system can derive $\Sigma \vdash \bot$, Σ is unsatisfiable.

Proof.

Since we have proven that formal derivation is sound in lecture 5, Σ is unsatisfiable.

Using resolution to prove statements

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

We assume Σ is finite. We will relax this by the next lecture.

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

We apply the resolution proof method on Σ'.

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

Exercise 9.3

Convert the above steps into a formal derivation.

Topic 9.2

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 have to worry about a single proof 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.

Superset clauses are redundant

Theorem 9.2

For clauses C and D, if $D \subset C$ and the empty clause can be derived using C then it can be derived using D.

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

Exercise 9.4

Prove the above theorem.

Ignore valid clauses in resolution

Definition 9.3

If a clause contains both p and $\neg p$ then the clause is syntactically valid.

If a syntactically valid clause contributes in deriving the empty clause, the descendents clause must participate in some resolution with pivot p.

However, that is impossible.

Example 9.3

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

Note that the resolution fails to remove p in the consequence.

If a syntactically valid clause is generated then we can ignore it for any further derivations, without loss of completeness.

Pure literals

Definition 9.4

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

Theorem 9.3

The removal of clauses containing the pure literals in a CNF preserves satisfiability.

Exercise 9.5

Prove the above theorem

Unit clause propagation

If $\{F\}$ occurs in a resolution proof, we can remove $\neg F$ from every clause, which is valid because of the following resolutions.

$$\frac{\{F\} \quad \neg F \cup D}{\{D\}}\text{Resolution}$$

Prefer resolving similar clauses

Our goal is to remove all literals.

In the following we removed p and brought in D

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

If most of the literals in D are in C, we will have less expansion.

Topic 9.3

Problems

Resolution: redundancy in resolution proofs

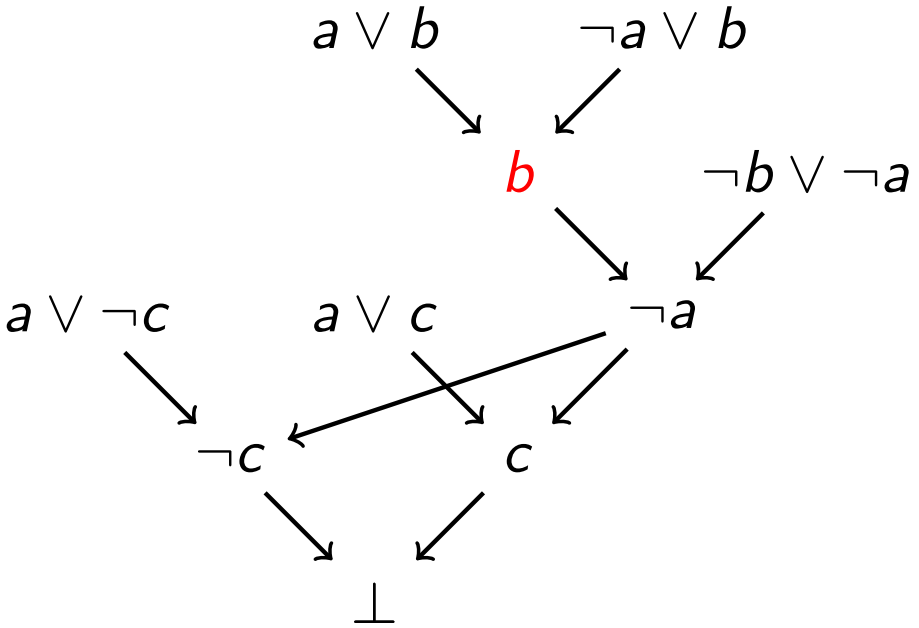
Exercise 9.6

Let us suppose we have a resolution proof deriving $\bot$. We have discussed that valid clauses should not be used for resolution. However, no one 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 9.7

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



The resolution producing b is redundant in both the paths to $\bot$.

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

End of Lecture 9