# CS228 Logic for Computer Science 2020

## Lecture 10: Completeness

Instructor: Ashutosh Gupta

IITB, India

Compile date: 2020-02-05

Topic 10.1

# Completeness

# Completeness

Now let us ask the daunting question!!!!!

Is resolution proof system complete?

In other words,
if $\Sigma$ is unsatisfiable, are we guaranteed to derive $\Sigma \vdash \perp$ via resolution?

We need a notion of not able to derive something.

# Clauses derivable with proofs of depth $n$

We define the set $Res^n(\Sigma)$ of clauses that are derivable via resolution proof of depth $n$ from the set of clauses $\Sigma$.

## Definition 10.1

*Let $\Sigma$ be a set of clauses.*

$$Res^0(\Sigma) \triangleq \Sigma$$

$$Res^{n+1}(\Sigma) \triangleq Res^n(\Sigma) \cup \{C | C \text{ is a resolvent of clauses } C_1, C_2 \in Res^n(\Sigma)\}$$

## Example 10.1

*Let* $\Sigma = \{(p \vee q), (\neg p \vee q), (\neg q \vee r), \neg r\}$.

$Res^0(\Sigma) = \Sigma$

$Res^1(\Sigma) = \Sigma \cup \{q, p \vee r, \neg p \vee r, \neg q\}$

$Res^2(\Sigma) = Res^1(\Sigma) \cup \{r, q \vee r, p, \neg p, \perp\}$

# All derivable clauses

Since there are only finitely many variables in $\Sigma$, we can only derive finitely many clauses. $Res^n(\Sigma)$ must saturate at some time point.

### Definition 10.2

*Let $\Sigma$ be a set of clauses. There must be some $m$ such that*

$$Res^{m+1}(\Sigma) = Res^m(\Sigma).$$

*Let $Res^*(\Sigma) \triangleq Res^m(\Sigma)$.*

# Completeness

### Theorem 10.1
*If a finite set of clauses $\Sigma$ is unsatisfiable, $\bot \in Res^*(\Sigma)$.*

### Proof.
We prove the theorem using induction over number of variables in $\Sigma$.
Wlog, We assume that there are no tautology clauses in $\Sigma$.(why?)

**base case:**
$p_1$ is the only variable in $\Sigma$.
Assume $\Sigma$ is unsat. Therefore, $\{p, \neg p\} \in \Sigma$.

We have the following derivation of $\bot$.

$$\frac{\Sigma \vdash p \qquad \Sigma \vdash \neg p}{\bot}$$

...

# Completeness (contd.)

## Proof(contd.)

**induction step:**

Assume: theorem holds for all the formulas containing variables $p_1, ..., p_n$.
Consider an unsatisfiable set $\Sigma$ of clauses containing variables $p_1, ...p_n, p$.
Let

- $\Sigma_0 \triangleq$ the set of clauses from $\Sigma$ that have $p$.
- $\Sigma_1 \triangleq$ be the set of clauses from $\Sigma$ that have $\neg p$.
- $\Sigma_* \triangleq$ be the set of clauses from $\Sigma$ that have neither $p$ nor $\neg p$.

$\Sigma = \Sigma_0 \wedge \Sigma_1 \wedge \Sigma_*$

Furthermore, let

- $\Sigma'_0 \triangleq \{C - \{p\} | C \in \Sigma_0\}$
- $\Sigma'_1 \triangleq \{C - \{\neg p\} | C \in \Sigma_1\}$

Now consider formula

$$\underbrace{(\Sigma'_0 \wedge \Sigma_*) \vee (\Sigma'_1 \wedge \Sigma_*)}_{\text{p is not in the formula}}$$

## Exercise 10.1

*Show* $\Sigma'_0 \models \Sigma_0$ *and* $\Sigma'_1 \models \Sigma_1$

...

# Example: projections

Example 10.2

*Consider* $\Sigma = \{p_1 \vee p, p_2, \neg p_1 \vee \neg p_2 \vee p, \neg p_2 \vee \neg p\}$

$\Sigma_0 = \{p_1 \vee p, \neg p_1 \vee \neg p_2 \vee p\}$
$\Sigma_1 = \{\neg p_2 \vee \neg p\}$
$\Sigma_* = \{p_2\}$

$\Sigma'_0 = \{p_1, \neg p_1 \vee \neg p_2\}$
$\Sigma'_1 = \{\neg p_2\}$

$(\Sigma'_0 \wedge \Sigma_*) \vee (\Sigma'_1 \wedge \Sigma_*) = \{p_1, \neg p_1 \vee \neg p_2, p_2\} \vee \{\neg p_2, p_2\}$

# Completeness (contd.)

## Proof(contd.)

**claim:** If $(\Sigma_0' \wedge \Sigma_*) \vee (\Sigma_1' \wedge \Sigma_*)$ is sat then $\Sigma$ is sat

- Assume $m \models (\Sigma_0' \wedge \Sigma_*) \vee (\Sigma_1' \wedge \Sigma_*)$
- Therefore, $m \models \Sigma_{*(\text{why?})}$
- Case 1: $m \models (\Sigma_1' \wedge \Sigma_*)$.
  Since all the clauses of $\Sigma_0$ have $p$, $m[p \mapsto 1] \models \Sigma_{0(\text{why?})}$.
  Since $\Sigma_1'$ has no $p$, $m[p \mapsto 1] \models \Sigma_1'$.
  Since $\Sigma_1' \models \Sigma_1$, $m[p \mapsto 1] \models \Sigma_1$.
- Case 2: $m \models (\Sigma_0' \wedge \Sigma_*)$.
  Symmetrically, $m[p \mapsto 0] \models \Sigma_0$ and $m[p \mapsto 0] \models \Sigma_1$.
- Therefore, there is a model to $\Sigma_0 \wedge \Sigma_1 \wedge \Sigma_*$.

Since $\Sigma$ is unsat, $(\Sigma_0' \wedge \Sigma_*) \vee (\Sigma_1' \wedge \Sigma_*)$ is unsat. ...

## Exercise 10.2

*Show $\Sigma$ and $(\Sigma_0' \wedge \Sigma_*) \vee (\Sigma_1' \wedge \Sigma_*)$ are equivsatisfiable*

# Completeness (contd.)

## Proof(contd.)

Now we apply the induction hypothesis.
Since $(\Sigma'_0 \wedge \Sigma_*) \vee (\Sigma'_1 \wedge \Sigma_*)$ is unsat and has no $p$, therefore

$$\bot \in Res^*(\Sigma'_0 \wedge \Sigma_*) \qquad \text{and} \qquad \bot \in Res^*(\Sigma'_1 \wedge \Sigma_*).$$

Choose a derivation of $\bot$ from both. ...

## Example 10.3

*Recall our example* $\Sigma_* = \{p_2\}$, $\Sigma'_0 = \{p_1, \neg p_1 \vee \neg p_2\}$, $\Sigma'_1 = \{\neg p_2\}$.
*Proofs for our running example*

$$\dfrac{\dfrac{p_1 \quad \neg p_1 \vee \neg p_2}{\neg p_2} \qquad p_2}{\bot} \qquad\qquad \dfrac{\neg p_2 \qquad p_2}{\bot}$$

*The above proofs do not start from clauses that are from* $\Sigma$.

# Completeness (contd.)

## Proof(contd.)

Now there are two cases.

Case 1: $\perp$ was derived using only clauses from $\Sigma_*$ in any of the two.
Therefore, $\perp \in Res^*(\Sigma_*)$. Therefore, $\perp \in Res^*(\Sigma_0 \wedge \Sigma_1 \wedge \Sigma_*)$.

Case 2: In both the derivations $\Sigma_0'$ are $\Sigma_1'$ are involved respectively.
Therefore, $p \in Res^*(\Sigma_0 \wedge \Sigma_*)$ and $\neg p \in Res^*(\Sigma_1 \wedge \Sigma_*)$.(why?)
Therefore, $\perp \in Res^*(\Sigma_0 \wedge \Sigma_1 \wedge \Sigma_*)$(why?). $\square$

## Example 10.4

*Recall proofs.*

$$\dfrac{\dfrac{\dfrac{p_1 \vee \mathbf{p} \qquad \neg p_1 \vee \neg p_2 \vee \mathbf{p}}{\neg p_2 \vee \mathbf{p}} \qquad p_2}{\perp \vee \mathbf{p}} \qquad \dfrac{\neg p_2 \vee \neg \mathbf{p} \qquad p_2}{\perp \vee \neg \mathbf{p}}}{\perp}$$

# Completeness so far

## Theorem 10.2

*Let $\Sigma$ be a finite set of formulas and $F$ be a formula. The following statements are equivalent.*

- $\Sigma \vdash F$
- $\emptyset \in Res^*(\Sigma')$, *where $\Sigma'$ is CNF of $\bigwedge \Sigma \wedge \neg F$*
- $\Sigma \models F$

## Proof.

$$\begin{tikzcd}
 & \Sigma \vdash F \arrow[dr, "\text{Proof rules are sound}"] & \\
\emptyset \in Res^*(\Sigma') \arrow[ur, "\text{Proof can be generated}"] & & \Sigma \models F \arrow[ll, "\text{Last theorem}"']
\end{tikzcd}$$

$\square$

## Exercise 10.3

*How last theorem is applicable here?*

Topic 10.2

# Finite to Infinite

# How do we handle $\Sigma'' \models F$ if $\Sigma''$ is an infinite set?

There is an interesting argument.

$$\begin{tikzcd}
 & \Sigma \vdash F \arrow[dr, "\text{Proof rules are sound}"] & \\
\emptyset \in Res^*(\Sigma') \arrow[ur, "\text{Proof can be generated}"] & & \Sigma \models F \arrow[ll, "\text{Last theorem}"'] \arrow[d, leftrightarrow, "\text{For some finite } \Sigma \subseteq \Sigma''"'] \\
 & & \Sigma'' \models F
\end{tikzcd}$$

We prove that if an infinite set implies some formula, then there is a finite subset also implies the formula.

# A theorem on strings

## Theorem 10.3

*Consider an infinite set $S$ of finite binary strings. There exists an infinite string $w$ such that the following holds.*

$$\forall n. \quad |\{w' \in S | w_n \text{ is prefix of } w'\}| = \infty$$

*where $w_n$ is prefix of $w$ of length $n$.*

## Proof.

We inductively construct $w$ and we will keep shrinking $S$. Initially $w := \epsilon$.

**base case:**

- Let $S_0 := \{u \in S | u \text{ starts with } 0\}$.
- Let $S_1 := \{u \in S | u \text{ starts with } 1\}$.
- Let $S_\epsilon := S \cap \{w\}$.

Clearly, $S = S_\epsilon \cup S_0 \cup S_1$. Either $S_0$ or $S_1$ are infinite.(why?)
If $S_0$ is infinite, $w := 0$ and $S := S_0$. Otherwise, $w := 1$ and $S := S_1$.
$w$ is prefix of all strings in the shrunk $S$. ...

# A theorem on strings (contd.)

## Proof(contd.)

**induction step:**

Let us suppose we have $w$ of length $n$ and $w$ is prefix of all strings in $S$.

- Let $S_0 := \{u \in S | u \text{ has } 0 \text{ at } n+1th \text{ position}\}$.
- Let $S_1 := \{u \in S | u \text{ has } 1 \text{ at } n+1th \text{ position}\}$.
- Let $S_\epsilon := S \cap \{w\}$.

Clearly, $S = S_\epsilon \cup S_0 \cup S_1$. Either $S_0$ or $S_1$ are infinite.(why?)

If $S_0$ is infinite, $w := w0$ and $S := S_0$. Otherwise, $w := w1$ and $S := S_1$. $w$ is prefix of all strings in the again shrunk $S$.

Therefore, we can construct the required $w$. □

## Exercise 10.4

*Is the above construction of $w$ practical?*

# Compactness

## Theorem 10.4

*A set $\Sigma$ of formulas is satisfiable iff every finite subset of $\Sigma$ is satisfiable.*

## Proof.

Forward direction is trivial.(why?)

Reverse direction:

We order formulas of $\Sigma$ in some order, i. e., $\Sigma = \{F_1, F_2, ......\}$.

Let $\{p_1, p_2, ...\}$ be ordered list of variables from $\mathbf{Vars}(\Sigma)$ such that

- variables in $\mathbf{Vars}(F_1)$ followed by
- the variables in $\mathbf{Vars}(F_2) - \mathbf{Vars}(F_1)$, and so on.

Due to the rhs, we have assignments $m_n$ such that $m_n \models \bigwedge_{i=1}^{n} F_i$.

We need to construct an assignment $m$ such that $m \models \Sigma$. Let us do it! ...

# Compactness (contd.) II

## Proof(contd.)

We assume $m_n : \textbf{Vars}(\bigwedge_{i=1}^{n} F_i) \rightarrow \mathcal{B}$.

We may see $m_n$ as finite binary strings since variables are ordered $p_1, p_2, ...$ and $m_n$ is assigning values to some first $k$ variables.

Let $S = \{m_n \text{ as a string } | n > 0\}$

Due to the previous theorem, there is an infinite binary string $m$ such that each prefix of $m$ is prefix of infinitely many strings in $S$. ...

# Compactness (contd.) III

## Proof(contd.)

**claim:** if we interpret $m$ as an assignment(how?), then $m \models F$.

- Consider a formula $F_n \in \Sigma$.
- Let $\bigwedge_{i=1}^{n} F_i$ has $k$ variables.
- Consider $m'$ be the prefix of length $k$ of $m$.

$$\overbrace{\underbrace{\underbrace{101....}_{m'}....}_{m_j}.....}^{m}$$

- There must be $m_j \in S$, such that $m'$ is prefix of $m_j$ and $j > n$(why?)
- Since $m_j \models \bigwedge_{i=1}^{j} F_i$, $m_j \models F_n$.
- Therefore, $m' \models F_n$.
- Therefore, $m \models F_n$. $\square$

# Implication is effectively enumerable.

### Theorem 10.5
*If $\Sigma$ is a finite set of formulas, then $\Sigma \models F$ is decidable.*

Proof. Due to truth tables. □

### Theorem 10.6
*If $\Sigma$ is effectively enumerable, then $\Sigma \models F$ is semi-decidable.*

Proof.
Due to compactness if $\Sigma \models F$, there is a finite set $\Sigma_0 \subseteq \Sigma$ such that $\Sigma_0 \models F$.
Since $\Sigma$ is effectively enumerable, let $G_1, G_2, ....$ be the enumeration of $\Sigma$.
Let $S_n \triangleq \{G_1, \ldots, G_n\}$.
There must be a $\Sigma_0 \subseteq S_{k\text{(why?)}}$.
Therefore, $S_k \models F$.
We may enumerate $S_n$ and check $S_n \models F$, which is decidable.
Therefore, eventually we will say yes if $\Sigma \models F$. □

**Commentary:** If $\Sigma \models F$ does not hold, the above procedure will not terminate.

Topic 10.3

# Problems

# Resolution: size upper bound

### Exercise 10.5

*Let $F$ an unsatisfiable CNF formula with $n$ variables. Show that there is a resolution proof of $\perp$ from $F$ that is smaller than $2^{n+1} - 1$.*

# End of Lecture 10