

CS228 Logic for Computer Science 2021

Lecture 10: Completeness

Instructor: Ashutosh Gupta

IITB, India

Compile date: 2021-01-29

Topic 10.1

Completeness

Completeness

Now let us ask the daunting question!!!!

Is resolution proof system complete?

In other words,
if Σ 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 proofs of depth n from the set of clauses Σ .

Definition 10.1

Let Σ be a set of clauses.

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

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

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 Σ , we can only derive finitely many clauses. $\text{Res}^n(\Sigma)$ must saturate at some time point.

Definition 10.2

Let Σ be a set of clauses. There must be some m such that

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

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

Completeness

Theorem 10.1

If a finite set of clauses Σ is unsatisfiable, $\perp \in \text{Res}^*(\Sigma)$.

Proof.

We prove the theorem using induction over number of variables in Σ .

Wlog, We assume that there are no tautology clauses in Σ ._(why?)

base case:

p is the only variable in Σ .

Assume Σ is unsat. Therefore, $\{p, \neg p\} \subseteq \Sigma$.

We have the following derivation of \perp .

$$\frac{\Sigma \vdash p \quad \Sigma \vdash \neg p}{\perp}$$

Completeness (contd.)

Proof(contd.)

induction step:

Assume: theorem holds for all the formulas containing variables p_1, \dots, p_n .

Consider an unsatisfiable set Σ of clauses containing variables p_1, \dots, p_n, p .

Let

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

$$\Sigma_0 \wedge \Sigma_1 = \psi$$

[No valid clauses]

Furthermore, let

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

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

...

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.)

Now consider formula

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

claim: If $(\Sigma'_0 \wedge \Sigma_*) \vee (\Sigma'_1 \wedge \Sigma_*)$ is sat then Σ is sat.

► Assume for some m , $m \models (\Sigma'_0 \wedge \Sigma_*) \vee (\Sigma'_1 \wedge \Sigma_*)$.

► Therefore, $m \models \Sigma_*$. (why?)

► Case 1: $m \models (\Sigma'_1 \wedge \Sigma_*)$.

Since all the clauses of Σ_0 have p , $m[p \mapsto 1] \models \Sigma_0$ (why?).

Since Σ'_1 and Σ_* have no p , $m[p \mapsto 1] \models \Sigma'_1$ and $m[p \mapsto 1] \models \Sigma_*$.

Since $\Sigma'_1 \models \Sigma_1$, $m[p \mapsto 1] \models \Sigma_1$.

► Case 2: $m \models (\Sigma'_0 \wedge \Sigma_*)$. Symmetrically, $m[p \mapsto 0] \models \Sigma_0 \wedge \Sigma_1 \wedge \Sigma_*$.

► Therefore, $\Sigma_0 \wedge \Sigma_1 \wedge \Sigma_*$ is sat.

...

Exercise 10.2 Show Σ and $(\Sigma'_0 \wedge \Sigma_*) \vee (\Sigma'_1 \wedge \Sigma_*)$ are equisatisfiable but not equivalent.

Completeness (contd.)

Proof(contd.)

Since Σ is unsat, $(\Sigma'_0 \wedge \Sigma_*) \vee (\Sigma'_1 \wedge \Sigma_*)$ is unsat.

Now we apply the induction hypothesis.

Since $(\Sigma'_0 \wedge \Sigma_*) \vee (\Sigma'_1 \wedge \Sigma_*)$ is unsat and has no p , $\perp \in \text{Res}^*(\Sigma'_0 \wedge \Sigma_*)$ and $\perp \in \text{Res}^*(\Sigma'_1 \wedge \Sigma_*)$.

Choose a derivation of \perp from both. Now there are two cases.

Case 1: \perp was derived using only clauses from Σ_* in any of the two proofs.

Therefore, $\perp \in \text{Res}^*(\Sigma_*)$. Therefore, $\perp \in \text{Res}^*(\Sigma_0 \wedge \Sigma_1 \wedge \Sigma_*)$.

Case 2: In both the derivations Σ'_0 are Σ'_1 are involved respectively.

...

Example: choosing derivations

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

$$\frac{\frac{p_1 \quad \neg p_1 \vee \neg p_2}{\neg p_2} \quad p_2}{\perp}$$

$$\frac{\neg p_2 \quad p_2}{\perp}$$

The above proofs belong to the case 2.

*The above proofs **do not start** from clauses that are from Σ . So we cannot use them immediately. We need **a construction**.*

Completeness (contd.)

Proof(contd.)

Case 2: In both the derivations Σ'_0 are Σ'_1 are involved respectively.(contd.)

Therefore, $p \in \text{Res}^*(\Sigma_0 \wedge \Sigma_*)$ and $\neg p \in \text{Res}^*(\Sigma_1 \wedge \Sigma_*)$.(why?)[needs thinking; look at the example to understand.]

Therefore, $\perp \in \text{Res}^*(\Sigma_0 \wedge \Sigma_1 \wedge \Sigma_*)$ (why?).



Example 10.4

Recall proofs.

$$\frac{\frac{\frac{p_1 \vee p \quad \neg p_1 \vee \neg p_2 \vee p}{\neg p_2 \vee p}}{\perp \vee p} \quad p_2 \quad \frac{\frac{\neg p_2 \vee \neg p}{\perp \vee \neg p} \quad p_2}{\perp}}$$

Exercise 10.3

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

Commentary: By inserting p in Σ'_0 clauses of the left proof we obtain clauses of Σ_0 . Therefore, the proof transforms into a proof from $\Sigma_0 \wedge \Sigma_*$. Since there are no $\neg p$ any where in $\Sigma_0 \wedge \Sigma_*$, we are guaranteed a leftover p . We need symmetric argument for deriving $\neg p$ from $\Sigma_1 \wedge \Sigma_*$.

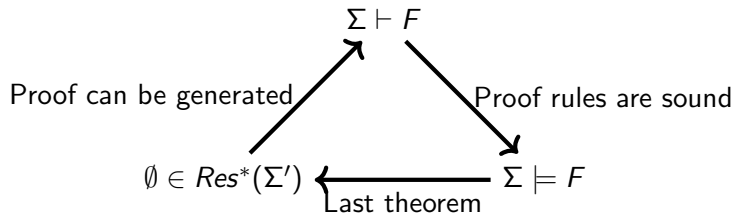
Completeness so far

Theorem 10.2

Let Σ be a finite set of formulas and F be a formula. The following statements are equivalent.

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

Proof.



Exercise 10.4

How is the last theorem applicable here?

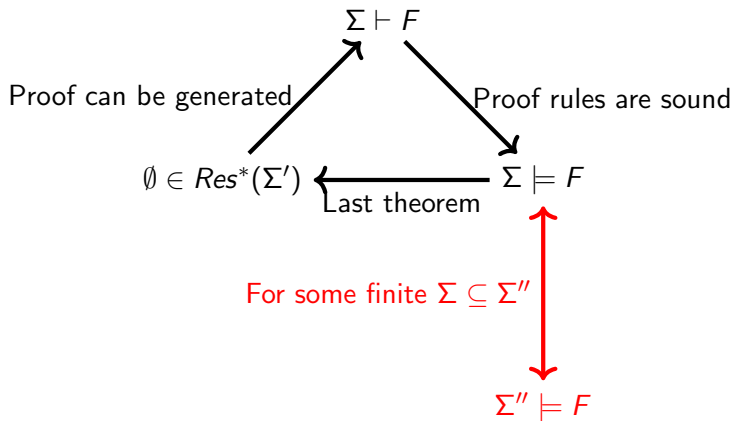


Topic 10.2

Finite to Infinite

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

There is an interesting argument.



We prove that if an infinite set implies a formula, then 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.

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

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 $S_0 := \{u \in S \mid u \text{ starts with } 0\}$.
- ▶ Let $S_1 := \{u \in S \mid u \text{ starts with } 1\}$.
- ▶ 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 $S_0 := \{u \in S \mid u \text{ has } 0 \text{ at } n+1 \text{th position}\}$.
- ▶ Let $S_1 := \{u \in S \mid u \text{ has } 1 \text{ at } n+1 \text{th position}\}$.
- ▶ 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 shrunk S .

Therefore, we can construct the required w . □

Exercise 10.5

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

Compactness

Theorem 10.4

A set Σ of formulas is satisfiable *iff* every finite subset of Σ is satisfiable.

Proof.

Forward direction is trivial._(why?)

Reverse direction:

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

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

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

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

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

...

Compactness (contd.) II

Proof(contd.)

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

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

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

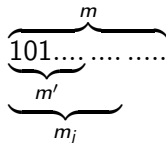
Due to the previous theorem, there is an infinite binary string m such that each prefix of m is prefix of infinitely many strings in S

Compactness (contd.) III

Proof(contd.)

claim: if we interpret m as a model_(how?), then $m \models \Sigma$.

- ▶ 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 .



- ▶ 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, $m \models F_n$.



Implication is decidable for finite lhs.

Theorem 10.5

If Σ is a finite set of formulas, then $\Sigma \models F$ is decidable.

Proof.

Due to truth tables.



Two definitions: effectively enumerable and semi-decidable

Definition 10.3

If we can enumerate a set using an algorithm, then it is called effectively enumerable.

Example 10.5

The set of all terminating programs is not effectively enumerable.

Definition 10.4

A yes/no problem is semi-decidable, if we have an algorithm for only one side of the problem.

Implication is effectively enumerable.

Theorem 10.6

If Σ 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 Σ is effectively enumerable, let G_1, G_2, \dots be the enumeration of Σ .

Let $S_n \triangleq \{G_1, \dots, G_n\}$.

There must be a $\Sigma_0 \subseteq S_k$ (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 $\Sigma \models F$.



Topic 10.3


Problems

Slim proofs

For an unsatisfiable CNF formula F , a resolution proof R is a sequence of clauses such that:

- ▶ Each clause in R is either from F or derived by resolution from the earlier clauses in R .
- ▶ The last clause in R is \perp .

Consider the following definitions

- ▶ For a clause C and literal ℓ , let $C|_{\ell} \triangleq \begin{cases} \top & \ell \in C \\ C - \{\bar{\ell}\} & \text{otherwise.} \end{cases}$ 
- ▶ Let $F|_{\ell} \triangleq \bigwedge_{C \in F} C|_{\ell}$.
- ▶ Let $\text{width}(R)$ and $\text{width}(F)$ be the length of the longest clause in R and F , respectively.
- ▶ Let $\text{slimest}(F) \triangleq \min(\{\text{width}(R) \mid R \text{ is resolution proof of unsatisfiability of } F\})$.

Exercise 10.6

Prove the following facts.

1. if $F|_{\ell}$ has an unsatisfiability proof, then $F \wedge \ell$ has an unsatisfiability proof.
2. if $k \geq \text{width}(F)$, $\text{slimest}(F|_{\ell}) \leq k - 1$, and $\text{slimest}(F|_{\bar{\ell}}) \leq k$ then $\text{slimest}(F) \leq k$.

End of Lecture 10

