

PL - Resolution with Completeness

HE Mingxin, Max

CS104: program07 @ yeah.net CS108:
mxhe1 @ yeah.net

I2ML(H) Spring 2023 (CS104|CS108)

Exercises 09 : Reading and More

Record your time spent (in 0.1 hours) with brief tasks and durations in your learning log by hand writing!

- 1) Read [textB-ch01-1.6-SAT-solvers.pdf](#) (cont.)
- 2) Work on Assignment 3&4... (cont.)

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 $\{\ell_1, \dots, \ell_n\}$ and interpreted as $\ell_1 \vee \dots \vee \ell_n$.

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 $\{C_1, \dots, C_n\}$ and interpreted as $C_1 \wedge \dots \wedge C_n$.

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

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

- ▶ 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 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.

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

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

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

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

Thinking Exercise 9.1

- There is something wrong with the above resolvents. What is it?*
- 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 aim of the proof method is to find the empty clause, which stands for inconsistency.

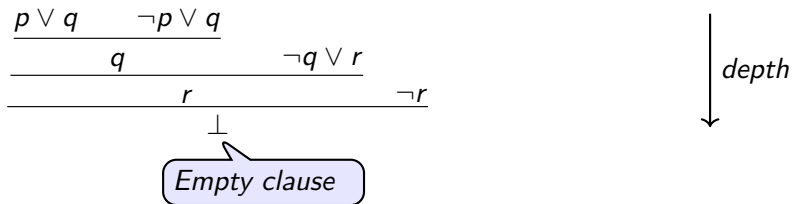
The proof systems that aim to derive false are called **refutation proof systems**.

Resolution Proofs

Example 9.2

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.

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 9.1

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

Proof.

Since we have proven that formal derivation is sound in lecture 5 & 6, Σ is unsatisfiable. \square

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 in Topic 5 this 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 \perp clause, $\Sigma \vdash F$ is derivable.

Thinking Exercise 9.2

Convert the above steps into a formal derivation.

Example: using resolution to prove statements

Example 9.3

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.

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

Topic 9.2

Implementation Issues in Resolution

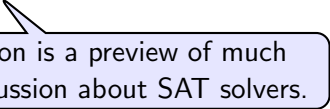
Efficient implementation of a proof method

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 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.

Superset clauses are redundant

Theorem 9.2

For clauses C and D , 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 9.4

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.

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

Thinking Exercise 9.3

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 *valid*.

If a valid clause *contributes* in deriving \perp , the descendants must participate in some resolution step with pivot p .

The resolution step is *counterproductive*, i.e., resolvent is superset of some antecedent.

Example 9.5

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

*Note that resolvent $p \vee C \vee D \supset p \vee C$, which makes the resolution step *counterproductive*.*

If a valid clause is generated, 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.

Thinking Exercise 9.4

Prove the above theorem

Unit clause propagation

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

$$\frac{\ell \quad \neg\ell \vee D}{D}$$

Prefer resolving similar clauses

Our goal is to remove all literals.

In the following we removed p and brought in D

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

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

Topic 9.3

Problems A

Resolution Proof

Thinking Exercise 9.5

Give resolution proofs of the following formulas.

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

Resolution: redundancy in resolution proofs

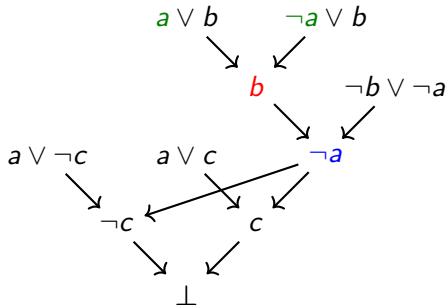
Thinking Exercise 9.6

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

Thinking Exercise 9.7

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 9.4

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 9.5

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 9.6

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

Definition 9.6

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 9.4

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 $\neg p$.

Furthermore, let

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

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

...

Thinking Exercise 9.8

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

Example: Projections

Example 9.7

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

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

$$\Sigma_1 = \{\neg p_2 \vee \neg p\}$$

$$\Sigma_* = \{p_2\}$$

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

$$\Sigma'_1 = \{\neg p_2\}$$

Let us get familiar with an important formula:

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

Completeness (contd.)

Proof(contd.)

Now consider formula

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

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.

...

Thk. Exe. 9.9 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 9.8

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 9.9

Recall proofs.

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

Thinking Exercise 9.10

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

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 $\neg p$ anywhere in $\Sigma_0 \wedge \Sigma_*$, we are guaranteed a leftover p . We need a symmetric argument for deriving $\neg p$ from $\Sigma_1 \wedge \Sigma_*$.

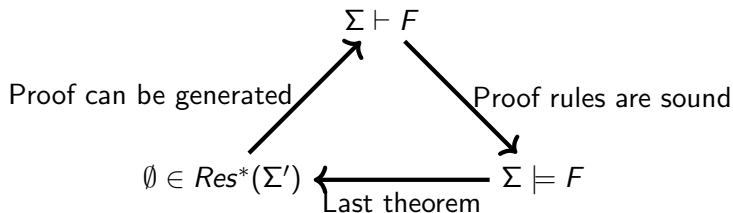
Completeness So Far

Theorem 9.5

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.



Thinking Exercise 9.11

How is the last theorem applicable here?

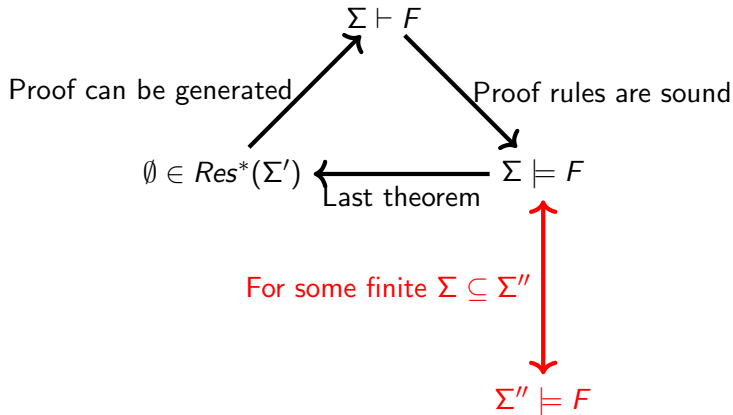


Topic 9.5

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 9.6

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 $S_\epsilon := S \cap \{w\}$.

Commentary: ϵ is the empty string.

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

If S_0 is **infinite**, $w := w0$ and $S := S_0$. Otherwise, $w := w1$ and $S := S_1$.
 w of length $n+1$ is prefix of all strings in the shrunk S .

Therefore, we can construct the required w .



Thinking Exercise 9.12

- Is the above construction of w **practical**?*
- Construct infinite w for set S containing words of form 0^*1*

Compactness

Theorem 9.7

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.)

Commentary: Notation alert: we assumed our models assign values to all variables. Here we are defining a different object that maps only finitely many variables.

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 .

...

Example : some m_n may not be a prefix of m

Example 9.10

Consider $\Sigma = \{p \vee q, \neg p \wedge r, \dots\}$

Let $m_1 = \{p \mapsto 1, q \mapsto 0\}$

Let $m_2 = \{p \mapsto 0, q \mapsto 1, r \mapsto 1\}$

Note that $m_1 \not\models \neg p \wedge r$. Therefore, m_1 will not be prefix of any m_n and consequently not prefix of m .

Thinking Exercise 9.13

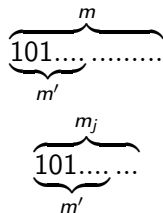
Construct Σ , m_n s, and m following the construction of previous slide such that no m_n is prefix of m ?

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 k be the number of variables appearing in $\bigwedge_{i=1}^n F_i$.
- ▶ 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, $m \models F_n$. □



Commentary: m' may not be m_n as in the example 9.10. The theorem is about showing that even if m_n is not there, there is some other model that satisfies F_n . Furthermore, m_j may also be not a prefix of m . Surprised! Georg Cantor lost his mind thinking about ∞ . Lookout for BBC documentary Dangerous Knowledge.

Implication is decidable for finite lhs.

Theorem 9.8

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 9.7

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

Example 9.11

- ▶ *The set of all programs effectively enumerable, since they are finite strings*
- ▶ *The set of all terminating programs is not effectively enumerable.*

Definition 9.7

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

Implication is semi-decidable

Theorem 9.9

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 $S_k \supseteq \Sigma_0$ (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 9.6

Problems B *

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 $\text{slimest}(F) \triangleq \min(\{\text{width}(R) \mid R \text{ is resolution proof of unsatisfiability of } F\})$.

Thinking Exercise 9.14

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 $\text{slimest}(F|_{\ell}) \leq k$ then $\text{slimest}(F) \leq k$.

Exercise: connect finite and infinite

Thinking Exercise 9.15

Consider an infinite set S of finite binary strings. Prove/disprove: For each infinite binary string w the following holds.

$$\forall n. |\{w' \in S \mid w_n \text{ is prefix of } w'\}| > 0 \quad \text{iff} \quad \forall n. |\{w' \in S \mid w_n \text{ is prefix of } w'\}| = \infty$$

where w_n is prefix of w of length n .

End of Lecture 9

Commentary: Solution: Reverse direction is trivial. Forward direction: w'_n be the word in S such that w_n is prefix of w'_n . Since w_n is prefix of w_{n+k} , w_n is prefix of w'_{n+k} for each $k \geq 0$. Therefore, $w'_{n+k} \in |\{w' \in S \mid w_n \text{ is prefix of } w'\}|$. Therefore, $|\{w' \in S \mid w_n \text{ is prefix of } w'\}| = \infty$.