Completeness of Resolution and Recap of computability
Soundness Theorem. If a set of clauses $\Gamma$ has a resolution refutation then $\Gamma$ is unsatisfiable.

Proof. Suppose $C_1, C_2, \ldots, C_m$ is a refutation of $\Gamma$. Consider:

$\Delta_0 = \Gamma$

$\Delta_i = \Delta_{i-1} \cup \exists C_i^3$

$\Delta_m = \exists \overline{\exists} \exists C_1, C_2, \ldots, C_m^3$

$C_m = \exists \overline{\exists} \exists \implies \Delta_m$ is unsatisfiable

Resolution Lemma. If $C$ is the resolvent of two clauses in $\Gamma$ then if $\Gamma \cup \exists C^3$ is unsatisfiable then $\Gamma$ is unsatisfiable.

Proof. Assume $\models \neg \Gamma$ and $C$ is the resolvent $DU\exists P^3$ and $EU\exists \neg P^3$.

$\models DU\exists P^3 \land EU\exists \neg P^3$

WLOG $\models C(p)$ = $T$

$\exists \overline{\exists}$ set of $\models [C] = T$

Therefore $\models C$ (since $C \subseteq C$)

Completeness Theorem. If $\Gamma$ is an unsatisfiable set of clauses then $\Gamma$ has a refutation.
Proof [Davis–Putnam]

Assume \( T \) is finite set and is unsatisfiable

\( T \) is non-empty

By induction on the number of propositions that appear in \( T \).

**Base Case:** The set of propositions appearing in \( T \) is empty.

\[ \exists \exists \in T \]

Refutation for \( T \) is \( \exists \exists \)

**Induction Step:** Let \( \phi \) be a proposition that appears in \( T \).

If a clause \( C \) contains both \( \phi \) and \( \neg \phi \), then \( C \) is satisfied. WLOG assume \( T \) does not contain such clauses.

We partition \( T \) into

\[ T_0^+ = \{ \exists \in T \mid \exists \phi, \neg \phi \wedge C = \phi \} \]

\[ T_0^- = \{ \exists \in T \mid \phi \in C \} \]

\[ T_0^0 = \{ \exists \in T \mid \neg \phi \in C \} \]

\[ T_0^- = \{ \exists \in T \mid \neg \phi \in C \} \]

\[ T_0^+ = T_0^+ \cup \exists \in C \cup D \mid C \cup \exists \phi \in T_0^+, D \cup \exists \neg \phi \in T_0^- \]

Refutation \( T \) is going to be do all resolutions and refute \( T_0^+ \).

**Lemma** \( T_0^+ \) is unsatisfiable.
Assume for contradiction that

\[ \neg T_p \]

is satisfiable.

Let \( v \models T_p \).

Let \( v' \) be the valuation that agrees with \( v \) on all propositions except \( \top \).

\[ v' \not\models T_p \]

WLOG \( v(\top) = T \) and \( v'(\top) = F \).

\[ T_p = T_0 \cup \{ \text{CUD} \mid \text{CU} \notin \rho_3 \in T_+ \}, DU \notin \rho_3 \leq T_- \]

\[ v \not\models T_0 \quad v' \not\models T_0 \]

\[ v \not\models T_+ \quad v' \not\models T_- \]

If \( v \not\models T_- \) or \( v' \not\models T_+ \), then \( T \) is satisfiable.

Assume \( v \not\models T_- \) and \( v' \not\models T_+ \).

\[ \exists \rho_3 \in T_+ \cup DU \notin \rho_3 \in T_- \]

s.t. \( v \not\models DU \notin \rho_3 \quad v' \not\models \text{CU} \rho_3 \).

\[ v, v' \not\models \text{CUD} \quad \text{and} \quad v, v' \not\models \text{C} \quad v, v' \not\models \text{CUD} \in T_p \]

Contradicts \( v, v' \not\models T_p \).

If \( T \) is an infinite set of unsatisfiable clauses, then Compactness Theorem says that \( \exists \) a finite subset \( \Delta \subseteq T \) which is unsatisfiable.
At the beginning
- Input tape contains input
- All other tapes are blank.
- State is initial state 0.

At any time, Turing machine reads the input tape and each work-tape. Based on its current state
- Change its state
- Write Symbols on each work-tape
- Move input/work tape heads either left or right.
- It may choose to write a symbol on the output tape.

Given an input, the TM can do one
of 3 things
- Run forever
- Halts but it does not accept
- Halts and accepts.

Language \( L(M) = \{ w \mid M \text{ accepts } w \} \)

M recognizes A iff \( A = L(M) \).

Church-Turing Thesis Any mechanical procedure can be implemented on Turing machine.

- For TM M there is \( \text{Sing}(M) \) that has only one work-tape s.t
  \( L(M) = L(\text{Sing}(M)) \)

- Nondeterministic: On any input the machine may have more than one computation.
  A NTM N accepts input x if N accepts x on some computation.

- For any NTM N there is a deterministic \( L(\text{det}(N)) \) s.t \( L(N) = L(\text{det}(N)) \)

Recursively Enumerable A language A is r.e. if \( \exists \) TM M s.t. \( L(M) = A \).
Recursive: A language \( A \) is recursive if there exists a machine \( M \) that halts on all inputs and \( L(M) = A \).

**Proposition:** If \( A \) is recursive then it is also r.e.

**Proposition:** If \( A \) is recursive then \( \overline{A} \) is also recursive.

**Proof:** If \( A \) is recursive then \( \exists M \) that halts on all inputs and \( L(M) = A \). Consider \( \overline{M} \): Runs \( M \) and flips \( M \)'s answer.

\[ \overline{M} \text{ halts on all inputs and } L(\overline{M}) = \overline{A}. \]

**Theorem:** \( A \) is recursive if and only if \( A \) is r.e. and \( \overline{A} \) is r.e.

**Proof:**

\( \Rightarrow \): \( \exists A \text{ r.e. } \Rightarrow \exists \overline{A} \text{ r.e. } \)

\( \Leftarrow \): \( A \) is recognized by \( M_1 \) and \( \overline{A} \) is recognized by \( M_2 \).

**Algorithm for \( A \):**

On input \( x \) (dovetailing)

Run \( M_1 \) on \( x \) and \( \overline{M_2} \) on \( x \).

Stop if \( M_1 \) or \( \overline{M_2} \) halts.
Run $M_{12}$ on $x$.

If $M_{12}$ accepts, then answer Yes.