EASLLC 2014

Logic and Computation

Phokion G. Kolaitis

University of California Santa Cruz
&
IBM Research - Almaden
There are many other complexity classes. For a comprehensive catalog, visit the Complexity Zoo at qwiki.stanford.edu/wiki/Complexity_Zoo
Complete Problems for Complexity Classes

Definition: A decision problem $Q$ is NP-complete if
- $Q$ is in NP.
- $Q$ is NP-hard: If $Q'$ is in NP, then there is a polynomial-time computable function $f$ such that for every string $x$, we have that $x \in Q' \iff f(x) \in Q$.
  - Such an $f$ is a polynomial-time reduction of $Q'$ to $Q$ ($Q' \preceq_p Q$).

Definition: A decision problem $Q$ is P-complete if
- $Q$ is in P.
- $Q$ is P-hard: If $Q'$ is in P, then there is a logarithmic-space computable function $f$ such that for every string $x$, we have that $x \in Q' \iff f(x) \in Q$.
  - Such an $f$ is a logarithmic-space reduction of $Q'$ to $Q$ ($Q' \preceq_{\log} Q$).
Complete Problems for Complexity Classes

Fact:
- Each of the complexity classes NL, P, NP, PSPACE possesses complete problems.
- Moreover, logic provides “natural” complete problems for each of these classes.

Theorem: (all undefined notions to be explained later on)
- QBF (Quantified boolean formulas) is PSPACE-complete.
- SAT (Satisfiability of CNF formulas) is NP-complete.
- Horn SAT (Satisfiability of Horn formulas) is P-complete.
- 2-SAT (Satisfiability of 2CNF-formulas) is NL-complete.
NP-Completeness

Definition:
- **CNF-formula**: a propositional formula in conjunctive normal form.
- **SAT (Satisfiability)**: Given a CNF-formula, is it satisfiable?

**Theorem (S. Cook – 1971)**
SAT is NP-complete

**Hint of Proof:**
- Membership in NP is easy (uses the fact that the model checking problem for propositional formulas is in P).
- NP-hardness: Encoding of non-deterministic Turing machines running in polynomial-time.
Complements of Complexity Classes

Definition: Let $\mathcal{C}$ be a complexity class. Then the complexity class $\text{co}\mathcal{C}$ consists of all languages $L$ such that the complement ($\Sigma^*-L$) of $L$ is in $\mathcal{C}$.

Example:
- $\text{coNP}$ consists of all languages whose complement is in NP.
- In particular, $\text{UNSAT} = \{\varphi : \varphi$ is a contradiction$\}$ is a member of $\text{coNP}$.

Fact: If $L$ is $\mathcal{C}$-complete, then ($\Sigma^*-L$) is $\text{co}\mathcal{C}$-complete. In particular, $\text{UNSAT}$ is $\text{coNP}$-complete.
Algorithmic Problems in Propositional Logic

- **The Model Checking Problem (Formula Evaluation Problem):**
  Given a propositional formula $\varphi$ and a truth assignment $s$, does $s$ satisfy $\varphi$?
  **Fact:** The Model Checking Problem is in P.

- **The Satisfiability Problem:**
  Given a propositional formula $\varphi$, is it satisfiable?
  **Fact:** The Satisfiability Problem is NP-complete
  **Reason:** It is in NP, and a special case of it (namely, SAT) is NP-complete.

- **The Tautology Problem:**
  Given a propositional formula $\varphi$, is it a tautology?
  **Fact:** The Tautology Problem is coNP-complete.
  **Reason:** $\varphi$ is a tautology if and only if ($\neg \varphi$) is unsatisfiable.
Algorithmic Problems in Propositional Logic

Facts:

- Unless $P = NP$, there is **no** polynomial-time algorithm for testing if a CNF-formula is satisfiable.

- Unless $P = NP$, there is **no** polynomial-time algorithm for testing if a DNF-formula is a tautology.

- There is a polynomial-time algorithm for testing if a DNF-formula is satisfiable (*why?*).

- There is a polynomial-time algorithm for testing if a CNF-formula is a tautology (*why?*).

- Unless $P = NP$,
  - there is **no** polynomial-time algorithm for converting a CNF-formula to a DNF-formula.
  - there is **no** polynomial-time algorithm for converting a DNF-formula to a CNF-formula.
Question: Suppose that a decision problem Q is NP-complete. What ways do we have to cope with the fact that, in all likelihood, no polynomial-time algorithm for Q exists?

Answer: There are three main possible approaches:

- Design heuristic algorithms for this problem that work well in “practice”.
- Identify special cases (restrictions) of the problem for which polynomial-time algorithms exist.
- Study the average-case complexity of the problem under a suitable distribution on the space of the inputs.
Fact:
- The Satisfiability Problem is NP-complete.
- The Tautology Problem is coNP-complete.

We also have to address a question raised earlier, namely:

Question: Find a notion of “proof” for propositional logic and use it to prove all tautologies (all true propositional logic formulas).

We will see that the answer to this question is closely related to the problem of finding a good heuristic for satisfiability.
Resolution in Propositional Logic

- The **Resolution Method** is a proof system for proving all tautologies of propositional logic.

- The **Davis-Putnam Procedure** is a widely used heuristic algorithm for the satisfiability problem (hence also for the tautology problem).

- The Resolution Method and the Davis-Putnam procedure are intimately related.
Literals, Clauses, and Sets of Clauses

Note: Recall that a literal is a propositional variable $P_i$ or a negated propositional variable ($\neg P_i$).

Definition:
- If $L$ is a literal, then the complement $L'$ of $L$ is
  - $L' = (\neg P_i)$, if $L = P_i$
  - $L' = P_i$, if $L = (\neg P_i)$.
- A clause is a finite set of literals.
- A truth assignment $s$ satisfies a clause $C$ if there is a literal $L$ in $C$ such that $s^*(L) = 1$.
- A truth assignment $s$ satisfies a set $F$ of clauses if it satisfies every clause in $F$.

Fact:
- The empty clause $\{\}$ is unsatisfiable.
- Every non-empty clause is satisfiable.
- Every CNF formula $\varphi$ can be identified with a finite set $F$ of clauses so that $\varphi$ is satisfiable if and only if $F$ is satisfiable.
Resolvents

Definition: Let C, D, and R be three clauses. We say that R is a resolvent of C and D if there is a literal L such that
L ∈ C, L’ ∈ D, and R = (C – {L}) ∪ (D – {L’}).

This is often denoted as a rule (the resolution rule):

\[ C^* \cup \{L\}, \ D^* \cup \{L'\} \]
\[ \hspace{1cm} \hline \]
\[ \hspace{2cm} C^* \cup D^* \]

Examples:

- \{¬ P, Q\}, \{P, R\}  \quad \{P, ¬ Q, R\}, \{¬ P, S, T\}
  \[ \hspace{1cm} \hline \]
  \[ \{Q, R\} \]  \quad \[ \hspace{1cm} \hline \]
  \[ \{¬ Q, R, S, T\} \]
The Resolution Operator

Definition:
- The resolution operator is the function Res that, when applied to a set F of clauses, returns the set Res(F) of clauses, where Res(F) = F ∪ { R: R is a resolvent of two clauses of F }.

- Res*(F) is the result of applying Res repeatedly. More formally,
  - Res^0(F) = F
  - Res^{n+1}(F) = Res(Res^n(F))
  - Res*(F) = \bigcup_{n \geq 0} Res^n(F)

Note:
F = Res^0(F) \subseteq Res^1(F) \subseteq ... \subseteq Res^n(F) \subseteq Res^{n+1}(F) \subseteq ...
Basic Properties of the Resolution Operator

Proposition: The following statements are true:

- If \( F \subseteq G \), then \( \text{Res}(F) \subseteq \text{Res}(G) \) (monotonicity of Res)
- \( \text{Res}(\text{Res}^*(F)) = \text{Res}^*(F) \)
- If \( F \) is a finite set of clauses, then there is a natural number \( m \) such that \( \text{Res}^*(F) = \text{Res}^m(F) \).

Proof of the last part:
If \( F \) is a finite set of clauses, then the number of distinct propositional variables occurring in \( F \) is finite, say it is \( k \). Since there are \( 2k \) literals arising from these variables, the total number of different clauses involving these variables is at most \( 2^{2k} \). Since the sequence \( \text{Res}^n(F) \), \( n \geq 0 \), is increasing and since each \( \text{Res}^n(F) \) is a set of clauses, the iteration has to come to an end after at most \( 2^{2k} \) many steps.
The Resolution Operator

Example:
Let $F = \{ \{P, Q, T\}, \{\neg P, Q, T\}, \{\neg Q, T\} \}$
The Resolution Operator

Example:
Let \( F = \{ \{P, Q, T\}, \{\neg P, Q, T\}, \{\neg Q, T\} \} \)

Then

- \( \text{Res}^0(F) = F \)
- \( \text{Res}^1(F) = F \cup \{ \{Q, T\}, \{P, T\}, \{\neg P, T\} \} \)
- \( \text{Res}^2(F) = \text{Res}^1(F) \cup \{ \{T\} \} \)
- \( \text{Res}^3(F) = \text{Res}^2(F) \)

Hence,
- \( \text{Res}^*(F) = \text{Res}^2(F). \)

Note: \( F \) is satisfiable (e.g., by \( s(P) = 1, s(Q) = 1, s(T) = 1 \)).
The Resolution Operator

Example:
Let $F = \{\{P, Q\}, \{\neg P, Q\}, \{P, \neg Q\}, \{\neg P, \neg Q\}\}$. 
The Resolution Operator

Example:
Let $F = \{ \{P, Q\}, \{\neg P, Q\}, \{P, \neg Q\}, \{\neg P, \neg Q\}\}$.

Then
- $\text{Res}^0(F) = F$
- $\text{Res}^1(F) = F \cup \{ \{Q\}, \{P\}, \{P, \neg P\}, \{Q, \neg Q\}, \{\neg P\}, \{\neg Q\}\}$
- $\text{Res}^2(F) = \text{Res}^1(F) \cup \{ \{\} \}$
- $\text{Res}^3(F) = \text{Res}^2(F)$

Hence,
- $\text{Res}^*(F) = \text{Res}^2(F)$

Note: $F$ is unsatisfiable.
**Resolution Derivation**

**Definition:** Let F be a set of clauses. A **resolution derivation** from F is a finite sequence $C_1, C_2, \ldots, C_n$ of clauses such that
- each $C_i$ is a member of F
or
- $C_i$ is a resolvent of two earlier clauses in the sequence.

**Example:** $F = \{ \{P, Q, T\}, \{\neg P, Q, T\}, \{\neg Q, T\} \}$
1. $\{P, Q, T\}$ in F
2. $\{\neg P, Q, T\}$ in F
3. $\{Q, T\}$ resolvent of 1. and 2.
4. $\{\neg Q, T\}$ in F
5. $\{T\}$ resolvent of 3. and 4.
Resolution Derivation

Definition: Let \( F \) be a set of clauses.
- A resolution derivation from \( F \) is a finite sequence \( C_1, C_2, \ldots, C_n \) of clauses such that
  - each \( C_i \) is a member of \( F \)
  - \( C_i \) is a resolvent of two earlier clauses in the sequence.

- A resolution derivation of a clause \( C \) from \( F \) is a resolution derivation \( C_1, \ldots, C_n \) from \( C \) such that \( C = C_n \).

Fact: The following are equivalent for a set \( F \) of clauses and a clause \( C \).
- \( C \in \text{Res}^*(F) \)
- There is a resolution derivation of \( C \) from \( F \).

Note:
- The notion of resolution derivation is a purely syntactic notion (no semantics involved, just manipulation of strings).
- A resolution derivation can be thought of as a “proof” in propositional logic.
The Resolution Theorem

Resolution Theorem (J.A. Robinson – 1965)

Let $F$ be a finite set of clauses. Then the following are equivalent:

1. $F$ is unsatisfiable
2. $\emptyset \in \text{Res}^*(F)$
3. There is a resolution derivation of $\emptyset$ from $F$.

Proof:

- $2. \iff 3.$ is easy (and was discussed earlier).
- $2. \implies 1.$ is the “soundness” of the resolution method.
- $1. \implies 2.$ is the “completeness” of the resolution method.

Note: In general, proving “completeness” is harder than proving “correctness”.
Illustration of the Resolution Theorem

Example:  \( F = \{ \{P, Q\}, \{\neg P, Q\}, \{P, \neg Q\}, \{\neg P, \neg Q\} \} \)

- \( F \) is unsatisfiable. So, the Resolution Theorem tells that a resolution derivation of \( \{\} \) from \( F \) exists.

- Resolution derivation of \( \{\} \) from \( F \).
  1. \( \{P, Q\} \) in \( F \)
  2. \( \{\neg P, Q\} \) in \( F \)
  3. \( \{Q\} \) resolvent of 1. and 2.
  4. \( \{P, \neg Q\} \) in \( F \)
  5. \( \{\neg P, \neg Q\} \) in \( F \)
  6. \( \{\neg Q\} \) resolvent of 4. and 5.
  7. \( \{\}\) resolvent of 3. and 6.

Note: Resolution derivations of \( \{\} \) from \( F \) need not be unique.
Illustration of the Resolution Theorem

Example: Let \( F = \{ \{P, Q, T\}, \{\neg P, Q, T\}, \{\neg Q, T\} \} \)

- \( F \) is satisfiable by \( s(P) = 1, s(Q) = 1, s(T) = 1 \). So, the Resolution Theorem tells that \( \emptyset \) is not a member of \( \text{Res}^*(F) \).

- \( \text{Res}^0(F) = F \)
- \( \text{Res}^1(F) = F \cup \{ \{Q, T\}, \{P, T\}, \{\neg P, T\} \} \)
- \( \text{Res}^2(F) = \text{Res}^1(F) \cup \{ \{T\} \} \)
- \( \text{Res}^3(F) = \text{Res}^2(F) \)

Hence,

- \( \text{Res}^*(F) = \text{Res}^2(F) = F \cup \{ \{Q, T\}, \{P, T\}, \{\neg P, T\}, \{T\} \} \)
- \( \emptyset \) is not a member of \( \text{Res}^*(F) \).
The Resolution Theorem

Resolution Theorem (J.A. Robinson – 1965)
Let F be a finite set of clauses. Then the following are equivalent:
1. F is unsatisfiable
2. \{\} \in \text{Res}^*(F)
3. There is a resolution derivation of \{\} from F.

Proof:

2. \Rightarrow 1. “Soundness”

We will prove something stronger, namely, that F and \text{Res}^*(F) are tautologically equivalent. Consequently, if \{\} \in \text{Res}^*(F), then \text{Res}^*(F) is unsatisfiable; hence F is unsatisfiable.
The Resolution Lemma

Resolution Lemma: Let $F$ be a set of clauses and $R \in \text{Res}(F)$. Then $F$ is tautologically equivalent to $F \cup \{R\}$.
(in symbols, $F \equiv F \cup \{R\}$)
The Resolution Lemma

Resolution Lemma: Let $F$ be a set of clauses and $R \in \text{Res}(F)$. Then $F$ is tautologically equivalent to $F \cup \{R\}$. (in symbols, $F \equiv F \cup \{R\}$)

Proof:
1. If $s$ satisfies $F \cup \{R\}$, then clearly $s$ satisfies $F$.
2. Assume that $s$ satisfies $F$. We have to show that $s$ satisfies $R$. Since $R \in \text{Res}(F)$, there are two clauses of the form $C \cup \{L\}$ and $D \cup \{L'\}$ in $F$ such that $R = C \cup D$.
   - If $s^*(L) = 0$, then $s^*(C) = 1$, hence $s^*(R) = 1$.
   - If $s^*(L) = 1$, then $s^*(D) = 1$, hence $s^*(R) = 1$.

Corollary: $F$ is tautologically equivalent to $\text{Res}^*(F)$ (in symbols, $F \equiv \text{Res}^*(F)$).
The Resolution Theorem

Resolution Theorem (J.A. Robinson – 1965)
Let F be a finite set of clauses. Then the following are equivalent:
1. F is unsatisfiable
2. \( \{ \} \in \text{Res}^*(F) \)
3. There is a resolution derivation of \( \{ \} \) from F.

Proof:
2. \( \Rightarrow \) 1. “Soundness” follows from the Resolution Lemma.
1. \( \Rightarrow \) 2. “Completeness”
This direction requires more work.


The Resolution Theorem

**Note:** It remains to show the following:
If $F$ is a finite unsatisfiable set of clauses, then $\{ \} \in \text{Res}^*(F)$.
- This will be shown using strong induction on the number of literals occurring in $F$ and the following lemma.

**The Splitting Lemma:** Let $F$ be a set of clauses and let $P$ be a propositional variable occurring in $F$. Let
  - $F(P/0) = \text{set of clauses obtained from } F \text{ by setting } P \text{ to } 0$
  - $F(P/1) = \text{set of clauses obtained from } F \text{ by setting } P \text{ to } 1$

Then the following statements are equivalent:
1. $F$ is unsatisfiable
2. Both $F(P/0)$ and $F(P/1)$ are unsatisfiable.
Splitting a Set of Clauses

Let F be a set of clauses and P be a prop. variable occurring in F.

- **F(P/0)** = set of clauses obtained from F by setting P to 0.
  
  More formally, F(P/0) is obtained from F by:
  
  - Keeping all clauses in F in which neither P nor \( \neg P \) occurs.
  - Removing all clauses in F containing \( \neg P \).
  - Removing P from each clause of F that contains P.

- **F(P/1)** = set of clauses obtained from F by setting P to 1.
  
  More formally, F(P/1) is obtained from F by:
  
  - Keeping all clauses in F in which neither P nor \( \neg P \) occurs.
  - Removing all clauses in F containing P.
  - Removing P from each clause of F that contains \( \neg P \).

**Note:** Here, we assume without loss of generality that no clause of F contains both P and \( \neg P \).
Splitting a Set of Clauses

Example:
\[ F = \{ \{P, Q, T\}, \{\neg P, \neg Q\}, \{\neg P, Q, T\}, \{\neg Q, \neg T\}, \{Q, \neg T\}\} \]

Splitting on P:
- \[ F(P/0) = \{ \{Q, T\}, \{\neg Q, \neg T\}, \{Q, \neg T\}\} \]
- \[ F(P/1) = \{ \{P, Q\}, \{Q, T\}, \{\neg Q, \neg T\}, \{Q, \neg T\}\} \]

Splitting on T:
- \[ F(T/0) = \{ \{P, Q\}, \{\neg P, \neg Q\}, \{\neg P, Q\}\} \]
- \[ F(T/1) = \{ \{\neg P, \neg Q\}, \{\neg Q\}, \{Q\}\} \]
The Splitting Lemma: Let $F$ be a set of clauses and let $P$ be a propositional variable occurring in $F$. Let

- $F(P/0)$ = set of clauses obtained from $F$ by setting $P$ to 0
- $F(P/1)$ = set of clauses obtained from $F$ by setting $P$ to 1.

Then the following statements are equivalent:

1. $F$ is unsatisfiable
2. Both $F(P/0)$ and $F(P/1)$ are unsatisfiable.

Proof:

1. $\Rightarrow$ 2. If $F$ is unsatisfiable, but, say, $F(P/0)$ is satisfiable by some assignment $s$, then extend $s$ to a satisfying assignment $t$ for $F$ by setting $t(P) = 0$.

2. $\Rightarrow$ 1. If $F(P/0)$ and $F(P/1)$ are unsatisfiable, then also $F$ is unsatisfiable, since if $s$ satisfies $F$, then

- $s$ satisfies $F(P/0)$, if $s(P) = 0$
- $s$ satisfies $F(P/1)$, if $s(P) = 1$. 
The Resolution Theorem

**Theorem:** If $F$ is a finite unsatisfiable set of clauses, then $\{ \} \in \text{Res}^*(F)$.

**Proof:** By strong induction on the number of literals in $F$. Assume $F$ has $n$ literals and that the result holds for all finite unsatisfiable sets of clauses with fewer than $n$ literals. Take a propositional variable $P$ in $F$. By the Splitting Lemma, both $F(P/0)$ and $F(P/1)$ are unsatisfiable and have fewer than $n$ literals. By induction hypothesis, $\{ \} \in \text{Res}^*(F(P/0))$ and $\{ \} \in \text{Res}^*(F(P/1))$. So, we have that:

- There is a resolution derivation $C_1, \ldots, C_m$ of $\{ \}$ from $F(P/0)$
- There is a resolution derivation $D_1, \ldots, D_s$ of $\{ \}$ from $F(P/1)$.

If each $C_i$ is a clause of $F$ or if each $D_j$ is a clause of $F$, then we have a derivation of $\{ \}$ from $F$. Otherwise,
The Resolution Theorem

- If $C_i$ is a clause in the resolution derivation $C_1, \ldots, C_m$ of $\{\}$ from $F(P/0)$ such that $C_i$ is not in $F$, then replace $C_i$ by $C_i \cup \{P\}$. This way we obtain a resolution derivation of $\{P\}$ from $F$.

- If $D_j$ is a clause in the resolution derivation $D_1, \ldots, D_s$ of $\{\}$ from $F(P/1)$ such that $D_j$ is not in $F$, then replace $D_j$ by $D_j \cup \{\neg P\}$. This way we obtain a resolution derivation of $\{\neg P\}$ from $F$.

- By concatenating these two resolution derivations from $F$ and then applying the resolution rule
  \[
  \{P\}, \{\neg P\}
  \]
  we obtain a resolution derivation of $\{\}$ from $F$.

This completes the proof of the Resolution Theorem.
Illustration and Exercise

- \( F = \{ \{P, Q, T\}, \{\neg P, Q, T\}, \ldots, \{\neg P, \neg Q, \neg T\} \} \) (8 clauses)
  - \( F(P/0) = \{ \{Q, T\}, \{\neg Q, T\}, \{Q, \neg T\}, \{\neg Q, \neg T\} \} \)
  - \( F(P/1) = \{ \{Q, T\}, \{\neg Q, T\}, \{Q, \neg T\}, \{\neg Q, \neg T\} \} \)

- Find a resolution derivation of \( \{ \} \) from \( F(P/0) \)
  - Insert \( P \) in every clause of this derivation (none of these is in \( F \)).
  - This yields a resolution derivation of \( \{ P \} \) from \( F \).

- Find a resolution derivation of \( \{ \} \) from \( F(P/1) \)
  - Insert \( \neg P \) in every clause of this derivation (none of these is in \( F \)).
  - This yields a resolution derivation of \( \{ \neg P \} \) from \( F \).

- Concatenate these two resolution derivations to derive a resolution derivation of \( \{ \} \) from \( F \).
Resolution Theorem (J.A. Robinson – 1965)

Let $F$ be a finite set of clauses. Then the following are equivalent:

1. $F$ is unsatisfiable
2. $\{ \} \in \text{Res}^*(F)$
3. There is a resolution derivation of $\{ \}$ from $F$.

Proof:

2. $\Rightarrow$ 1. “Soundness”

Key ingredient: Resolution Lemma.

1. $\Rightarrow$ 2. “Completeness”

Key ingredients: Splitting Lemma + Strong Induction.
Resolution Theorem (J.A. Robinson – 1965)
Let F be an arbitrary set of clauses. Then the following are equivalent:

1. F is unsatisfiable
2. \{ \} \in \text{Res}^*(F)
3. There is a resolution derivation of \{ \} from F.

Proof:

2. \implies 1. “Soundness”

Key ingredient: Resolution Lemma.

1. \implies 2. “Completeness”

Resolution Theorem for finite sets + Compactness Theorem.
Resolution as a Proof System for Tautologies

- Let $\varphi$ be a formula in DNF. Then, the Resolution Theorem implies that the following statements are equivalent:
  1. $\varphi$ is a tautology (semantic notion)
  2. $\neg \varphi$ is unsatisfiable (semantic notion)
  3. There is a resolution derivation of $\{\}$ from $\neg \varphi$. (syntactic notion)

Note:

- The Resolution Theorem asserts that a purely semantic notion coincides with a purely syntactic notion.
- Resolution derivation is a notion of proof in propositional logic.
- The Resolution Theorem asserts that this notion of proof is sound and complete: it allows us to prove all tautologies and nothing else. (the whole truth and nothing but the truth).
  - **Soundness**: If there is a resolution derivation of $\{\}$ from $\neg \varphi$, then $\varphi$ is a tautology (nothing but the truth).
  - **Completeness**: If $\varphi$ is a tautology, then there is a resolution derivation of $\{\}$ from $\neg \varphi$. (the whole truth).
Resolution as an Algorithm for Satisfiability

Resolution Algorithm I:
Given a CNF-formula $\varphi$:

- Compute $\text{Res}^*(\varphi)$, where $\varphi$ is viewed as a finite set of clauses.
  - If $\{\}$ is not in $\text{Res}^*(\varphi)$, then $\varphi$ is satisfiable.
  - If $\{\}$ is in $\text{Res}^*(\varphi)$, then $\varphi$ is unsatisfiable.

Note:
- Termination of the algorithm follows from the fact that there is some $m$ such that $\text{Res}^*(\varphi) = \text{Res}^m(\varphi)$.
- Correctness of the algorithm follows from the Resolution Theorem.
Resolution as an Algorithm for SAT

Resolution Algorithm I:
Given a CNF-formula $\phi$:

- Compute $\text{Res}^*(\phi)$, where $\phi$ is viewed as a finite set of clauses.
  - If $\{\}$ is not in $\text{Res}^*(\phi)$, then $\{\}$ is satisfiable.
  - If $\{\}$ is in $\text{Res}^*(\phi)$, then $\{\}$ is unsatisfiable.

Fact:

- In the worst case, Resolution Algorithm I may require exponentially many steps (more on this later).
- However, Resolution Algorithm I runs in polynomial time in an important special case.
Resolution and 2SAT

Definition: 2SAT is the following decision problem: Given a 2CNF-formula, is it satisfiable?

Theorem: (Krom – 1967) Resolution Algorithm I is a polynomial-time algorithm for 2SAT.
Resolution and 2SAT

**Theorem:** (Krom – 1967)
Resolution Algorithm I is a polynomial-time algorithm for 2SAT.

**Proof:**
- Resolution Algorithm I is an algorithm for 2SAT since it is an algorithm for SAT.
- Resolution Algorithm I is a polynomial-time algorithm for 2SAT, because
  - Every resolvent of two clauses of size at most 2 is a clause of size at most 2. Hence, \( \text{Res}^*(\varphi) \) consists of sets of clauses of size at most 2.
  - If a 2CNF-formula \( \varphi \) has \( k \) propositional variables, then there are \( O(k^2) \) clauses of size at most 2. Hence, the computation of \( \text{Res}^*(\varphi) \) terminates within \( O(k^2) = O(|\varphi|^2) \) steps.
On the Difference between 2 and 3

Fact:
- 3SAT is NP-complete
- 2SAT is in P.

Fact:
2 is the only natural number $k > 1$ such that a resolvent of two clauses of size at most $k$ is a clause of size at most $k$.
($2+2-2 = 2$, while $3+3-2 = 4 > 3$, $4+4-2 = 6$, etc.).
Computational Complexity of 2SAT

- Recall that \( \text{LOGSPACE} \subseteq \text{NLOGSPACE} \subseteq \text{P} \subseteq \text{NP} \subseteq \text{PSPACE} \).
- Recall also that \( \text{NLOGSPACE} = \text{coNLOGSPACE} \) (Immerman - Szelepcsényi Theorem)

**Fact:** PATH is NLOGSPACE-complete, where PATH is the following decision problem:
Given a directed graph \( G = (V,E) \) and two nodes \( s \) and \( t \) in \( V \), is there a path from \( s \) to \( t \) along the edges of \( G \)? (in symbols: does \( \text{PATH}(G,s,t) \) hold?)

**Theorem:** 2SAT is NLOGSPACE-complete.
**Proof:** We will show that
- 2SAT is in NLOGSPACE
- \( \text{PATH} \preceq_{\log} 2\text{SAT} \).
Computational Complexity of 2SAT

- Given a 2CNF formula $\varphi$, construct a directed graph $G_{\varphi}$ as follows:
  - The nodes of $G_{\varphi}$ are the literals occurring in $\varphi$.
  - For every conjunct $(L_1 \lor L_2)$ of $\varphi$, we form the edges $E(L_1', L_2)$ and $E(L_2', L_1)$
    (recall that $(L_1 \lor L_2) \equiv (L_1' \rightarrow L_2) \equiv (L_2' \rightarrow L_1)$).

Note: There is a logarithmic space algorithm that, given a 2CNF formula $\varphi$, the algorithm constructs the graph $G_{\varphi}$.

Claim 1: The following statements are equivalent:
1. $\varphi$ is satisfiable
2. There is no propositional variable $P$ such that both $\text{PATH}(G_{\varphi}, P \rightarrow P)$ and $P(G_{\varphi}, \neg P, P)$ hold.

Proof of Claim 1: Exercise
Illustration of the Construction

- Formula $\varphi$: $(P \lor Q) \land (\neg P \lor Q) \land (P \lor \neg Q) \land (\neg P \lor \neg Q)$
  - Graph $G_\varphi$
    - $P \leftrightarrow Q$
    - $\neg P \leftrightarrow \neg Q$
    - Path: $P \rightarrow \neg Q \rightarrow \neg P$
    - Path: $\neg P \rightarrow Q \rightarrow P$

- Formula $\varphi$: $(P \lor Q) \land (\neg P \lor Q) \land (P \lor \neg Q)$
  - Graph $G_\varphi$
    - $P \leftrightarrow Q$
    - $\neg P \leftrightarrow \neg Q$
    - Path: $\neg P \rightarrow Q \rightarrow P$
    - No path from $P$ to $\neg P$
Computational Complexity of 2SAT

Claim 2: 2SAT is in NLOGSPACE
Proof of Claim 2:

- Since there is a logspace reduction of 2UNSAT to PATH and since NLOGSPACE is closed under logspace reductions, it follows that 2UNSAT is in NLOGSPACE.
- Since NLOGSPACE is closed under complements, it follows that 2SAT is in NLOGSPACE.

Claim 3: 2SAT is NLOGSPACE-hard.
Proof of Claim 3:

Given a graph G and nodes s and t, construct the following 2CNF formula $\varphi_G$

- For every edge $(a,b)$ of $E$, add the conjunct ($\neg a \lor b$)
- Add also the conjuncts $s$ and ($\neg t$).

Then it is easy to see that PATH(G,s,t) holds if and only if $\varphi_G$ is unsatisfiable.
# Complexity of Satisfiability and its Variants

<table>
<thead>
<tr>
<th>Problem</th>
<th>Computational Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>SAT</td>
<td>NP-complete</td>
</tr>
<tr>
<td>3SAT</td>
<td>NP-complete</td>
</tr>
<tr>
<td>1-in-3 SAT</td>
<td>NP-complete</td>
</tr>
<tr>
<td>2SAT</td>
<td>NLOGSPACE-complete (hence in P)</td>
</tr>
</tbody>
</table>
Question: Let $F$ be an unsatisfiable set of clauses. How long can the shortest resolution derivations of $\emptyset$ be?

Note:
- A resolution derivation $C_1, \ldots, C_m$ of $\emptyset$ from $F$ is a proof that $F$ is unsatisfiable.
- Moreover, one can check in polynomial time whether a given sequence $C_1, \ldots, C_m$ is a resolution derivation of $\emptyset$ from $F$.
- This seems to suggest that UNSAT is in NP ("guess a candidate resolution derivation and verify that it is one"), which would imply that NP = coNP.
Question: Let F be an unsatisfiable set of clauses. How long can the shortest resolution derivations of \{ \} be?

Note:
- A resolution derivation $C_1, \ldots, C_m$ of \{ \} from F is a proof that F is unsatisfiable.
- Moreover, one can check in polynomial time whether a given sequence $C_1, \ldots, C_m$ is a resolution derivation of \{ \} from F.
- This seems to suggest that UNSAT is in NP (“guess a candidate resolution derivation and verify that it is one”), which would imply that NP = coNP.
- However, this is not true, because it is not true that if F is unsatisfiable, then there is always a “short” (i.e., polynomial in the size of F) resolution derivation of \{ \} from F.
Worst-Case Behavior of the Resolution Algorithm

Fact: There are sequences $F_1, F_2, \ldots, F_n, \ldots$ of sets of clauses such that each $F_n$ is unsatisfiable, but every resolution derivation of $\emptyset$ from $F_n$ has length $\Omega(2^{|F_n|})$.

The Pigeonhole Principle $P(n)$:
If $n+1$ letters are placed into $n$ mailboxes, then one of these mailboxes will have at least two letters.

Fact: The Pigeonhole Principle $P(n)$ can be “expressed” in propositional logic:
- Introduce propositional variable $P_{ij}$, $1 \leq i \leq n+1$, $1 \leq j \leq n$.
  Intuitively, $P_{ij}$ is true means that letter $i$ is placed in mailbox $j$.
- Let $F_n$ be the set consisting of the following clauses
  - $\{ P_{i1}, \ldots, P_{in} \}$, for $1 \leq i \leq n+1$ (every letter is placed in a mailbox)
  - $\{ \neg P_{ik}, \neg P_{jk} \}$, for $i \neq j$ and $1 \leq k \leq n$ (no mailbox gets two letters)
- $F_n$ is an unsatisfiable set of clauses, for every $n \geq 1$. 
Worst-Case Behavior of the Resolution Algorithm

Examples:

- $F_1 = \{ \{P_{11}\}, \{P_{21}\}, \{\neg P_{11}, \neg P_{21}\} \}$
- $F_2 = \{ \{P_{11}, P_{12}\}, \{P_{21}, P_{22}\}, \{P_{31}, P_{32}\},$
  \[ \{\neg P_{11}, \neg P_{21}\}, \{\neg P_{11}, \neg P_{31}\}, \{\neg P_{21}, \neg P_{31}\}, \{\neg P_{12}, \neg P_{22}\}, \{\neg P_{12}, \neg P_{32}\}, \{\neg P_{22}, \neg P_{32}\} \}$

Fact: Each $F_n$ has $O(n^3)$-many clauses in $F_n$.

Theorem (Haken – 1985):
Every resolution derivation of $\{ \} \text{ from } F_n$ has length $\Omega(2^n)$. 