CS 577- Intro to Algorithms

Computational Intractability (Part 2)

Dieter van Melkebeek

November 19, 2020

Outline

Outline

Motivation

Outline

Motivation

- Recognizing infeasible approaches

Outline

Motivation

- Recognizing infeasible approaches
- P vs NP problem

Outline

Motivation

- Recognizing infeasible approaches
- P vs NP problem

P, NP, and NPC

Outline

Motivation

- Recognizing infeasible approaches
- P vs NP problem

P, NP, and NPC

- P: decision problems that have polynomial-time algorithms

Outline

Motivation

- Recognizing infeasible approaches
- P vs NP problem

P, NP, and NPC

- P: decision problems that have polynomial-time algorithms
- NP: decision problems with yes-instances that have polynomial-time verifiable certificates

Outline

Motivation

- Recognizing infeasible approaches
- P vs NP problem

P, NP, and NPC

- P: decision problems that have polynomial-time algorithms
- NP: decision problems with yes-instances that have polynomial-time verifiable certificates
- Fact: $P \subseteq NP$

Outline

Motivation

- Recognizing infeasible approaches
- P vs NP problem

P, NP, and NPC

- P: decision problems that have polynomial-time algorithms
- NP: decision problems with yes-instances that have polynomial-time verifiable certificates
- Fact: $\mathrm{P} \subseteq \mathrm{NP}$
- Conjecture: $\mathrm{P} \neq \mathrm{NP}$

Outline

Motivation

- Recognizing infeasible approaches
- P vs NP problem

P, NP, and NPC

- P: decision problems that have polynomial-time algorithms
- NP: decision problems with yes-instances that have polynomial-time verifiable certificates
- Fact: P $\subseteq$ NP
- Conjecture: P $\neq$ NP
- NP-complete (NPC): hardest problems in NP

Outline

Motivation

- Recognizing infeasible approaches
- P vs NP problem

P, NP, and NPC

- P: decision problems that have polynomial-time algorithms
- NP: decision problems with yes-instances that have polynomial-time verifiable certificates
- Fact: $\mathrm{P} \subseteq \mathrm{NP}$
- Conjecture: $\mathrm{P} \neq \mathrm{NP}$
- NP-complete (NPC): hardest problems in NP
- Assume $\mathrm{P} \neq \mathrm{NP}$. If $B \in \mathrm{NPC}$ then $B \notin \mathrm{P}$.

Outline

Motivation

- Recognizing infeasible approaches
- P vs NP problem

P, NP, and NPC

- P: decision problems that have polynomial-time algorithms
- NP: decision problems with yes-instances that have polynomial-time verifiable certificates
- Fact: P $\subseteq$ NP
- Conjecture: P $\neq$ NP
- NP-complete (NPC): hardest problems in NP
- Assume P $\neq$ NP. If $B \in$ NPC then $B \notin$ P.

Today: Satisfiability

NP Decision/Search/Optimization

NP Decision/Search/Optimization

- Parameters:
  - $c \in \mathbb{N}$
  - $V : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ in P
  - $f : \{0,1\}^* \times \{0,1\}^* \to \mathbb{R}$ in P

NP Decision/Search/Optimization

- Parameters:
  - $c \in \mathbb{N}$
  - $V : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ in P
  - $f : \{0,1\}^* \times \{0,1\}^* \to \mathbb{R}$ in P
- Solution set for input $x \in \{0,1\}^n$:

$$S_x \doteq \{y \in \{0,1\}^{n^c} : V(x,y) = 1\}$$

NP Decision/Search/Optimization

- Parameters:
  - $c \in \mathbb{N}$
  - $V : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ in P
  - $f : \{0,1\}^* \times \{0,1\}^* \to \mathbb{R}$ in P
- Solution set for input $x \in \{0,1\}^n$:

$$S_x \doteq \{y \in \{0,1\}^{n^c} : V(x,y) = 1\}$$

- Goal:

NP Decision/Search/Optimization

- Parameters:
  - $c \in \mathbb{N}$
  - $V : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ in P
  - $f : \{0,1\}^* \times \{0,1\}^* \to \mathbb{R}$ in P
- Solution set for input $x \in \{0,1\}^n$:

$$S_x \doteq \{y \in \{0,1\}^{n^c} : V(x,y) = 1\}$$

- Goal:

  Decision Is $S_x \neq \emptyset$?

NP Decision/Search/Optimization

- Parameters:
  - $c \in \mathbb{N}$
  - $V : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ in P
  - $f : \{0,1\}^* \times \{0,1\}^* \to \mathbb{R}$ in P
- Solution set for input $x \in \{0,1\}^n$:

$$S_x \doteq \{y \in \{0,1\}^{n^c} : V(x,y) = 1\}$$

- Goal:

  Decision Is $S_x \neq \emptyset$?

  Search Find $y \in S_x$ or report that no such y exists.

NP Decision/Search/Optimization

- Parameters:
  - $c \in \mathbb{N}$
  - $V : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ in P
  - $f : \{0,1\}^* \times \{0,1\}^* \to \mathbb{R}$ in P
- Solution set for input $x \in \{0,1\}^n$:

$$S_x \doteq \{y \in \{0,1\}^{n^c} : V(x,y) = 1\}$$

- Goal:

Decision	Is $S_x \neq \emptyset$?
Search	Find $y \in S_x$ or report that no such y exists.
Optimization	Find $y^* \in S_x$ such that $f(x,y^*) = \min_{y \in S_x}(f(x,y))$ respectively $f(x,y^*) = \max_{y \in S_x}(f(x,y))$

NP Decision/Search/Optimization

- Parameters:
  - $c \in \mathbb{N}$
  - $V : \{0,1\}^* \times \{0,1\}^* \to \{0,1\}$ in P
  - $f : \{0,1\}^* \times \{0,1\}^* \to \mathbb{R}$ in P
- Solution set for input $x \in \{0,1\}^n$:

$$S_x \doteq \{y \in \{0,1\}^{n^c} : V(x,y) = 1\}$$

- Goal:

Decision: Is $S_x \neq \emptyset$?

Search: Find $y \in S_x$ or report that no such y exists.

Optimization: Find $y^* \in S_x$ such that
$f(x,y^*) = \min_{y \in S_x}(f(x,y))$ respectively
$f(x,y^*) = \max_{y \in S_x}(f(x,y))$

- Examples: Independent Set, Satisfiability, Graph Coloring, Traveling Salesperson Problem, ...

NP-Hardness and NP-Completeness

NP-Hardness and NP-Completeness

Definition

B is NP-hard if $(\forall A \in \mathrm{NP})\, A \leq^p B$.

NP-Hardness and NP-Completeness

Definition
B is NP-hard if $(\forall A \in \mathrm{NP})\, A \leq^p B$.

Definition
B is NP-complete if B is NP-hard and $B \in \mathrm{NP}$.

NP-Hardness and NP-Completeness

Definition

B is NP-hard if $(\forall\, A \in \mathrm{NP})\, A \leq^p B$.

Definition

B is NP-complete if B is NP-hard and $B \in \mathrm{NP}$.

Proposition

Suppose B is NP-complete. Then $B \in \mathrm{P} \Leftrightarrow \mathrm{P} = \mathrm{NP}$.

NP-Hardness and NP-Completeness

Definition

B is NP-hard if $(\forall\, A \in \mathrm{NP})\, A \leq^p B$.

Definition

B is NP-complete if B is NP-hard and $B \in \mathrm{NP}$.

Proposition

Suppose B is NP-complete. Then $B \in \mathrm{P} \Leftrightarrow \mathrm{P} = \mathrm{NP}$.

Proof

NP-Hardness and NP-Completeness

Definition
B is NP-hard if $(\forall\, A \in \mathrm{NP})\, A \leq^p B$.

Definition
B is NP-complete if B is NP-hard and $B \in \mathrm{NP}$.

Proposition
Suppose B is NP-complete. Then $B \in \mathrm{P} \Leftrightarrow \mathrm{P} = \mathrm{NP}$.

Proof
$\Leftarrow$ $B \in \mathrm{NP}$ and $\mathrm{P} = \mathrm{NP}$ implies $B \in \mathrm{P}$.

NP-Hardness and NP-Completeness

Definition

B is NP-hard if $(\forall\, A \in \mathrm{NP})\, A \leq^p B$.

Definition

B is NP-complete if B is NP-hard and $B \in \mathrm{NP}$.

Proposition

Suppose B is NP-complete. Then $B \in \mathrm{P} \Leftrightarrow \mathrm{P} = \mathrm{NP}$.

Proof

$\Leftarrow$ $B \in \mathrm{NP}$ and $\mathrm{P} = \mathrm{NP}$ implies $B \in \mathrm{P}$.

$\Rightarrow$ Consider any $A \in \mathrm{NP}$.

NP-Hardness and NP-Completeness

Definition

B is NP-hard if $(\forall A \in \mathrm{NP})\, A \leq^p B$.

Definition

B is NP-complete if B is NP-hard and $B \in \mathrm{NP}$.

Proposition

Suppose B is NP-complete. Then $B \in \mathrm{P} \Leftrightarrow \mathrm{P} = \mathrm{NP}$.

Proof

- $\Leftarrow$ $B \in \mathrm{NP}$ and $\mathrm{P} = \mathrm{NP}$ implies $B \in \mathrm{P}$.
- $\Rightarrow$ Consider any $A \in \mathrm{NP}$.
  $A \leq^p B$ and $B \in \mathrm{P}$ implies $A \in \mathrm{P}$.

NP-Hardness and NP-Completeness

Definition
B is NP-hard if $(\forall\, A \in \mathrm{NP})\, A \leq^p B$.

Definition
B is NP-complete if B is NP-hard and $B \in \mathrm{NP}$.

Proposition
Suppose B is NP-complete. Then $B \in \mathrm{P} \Leftrightarrow \mathrm{P} = \mathrm{NP}$.

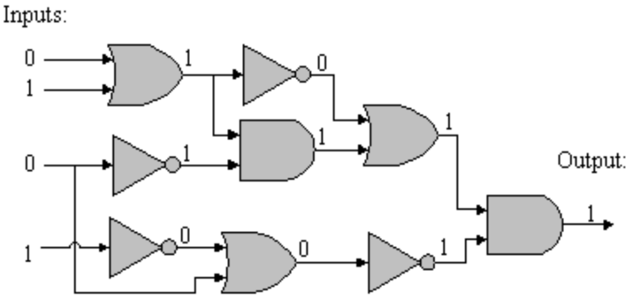
Proof
$\Leftarrow$ $B \in \mathrm{NP}$ and $\mathrm{P} = \mathrm{NP}$ implies $B \in \mathrm{P}$.

$\Rightarrow$ Consider any $A \in \mathrm{NP}$.
$A \leq^p B$ and $B \in \mathrm{P}$ implies $A \in \mathrm{P}$.

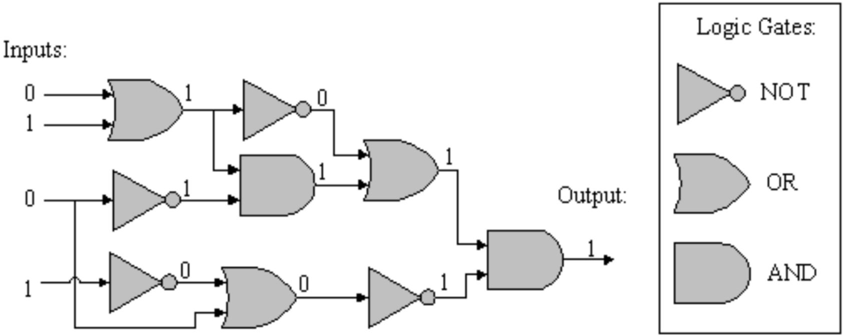
Corollary
Assume $\mathrm{P} \neq \mathrm{NP}$. If B is NP-hard then $B \notin \mathrm{P}$.

Circuit Satisfiability

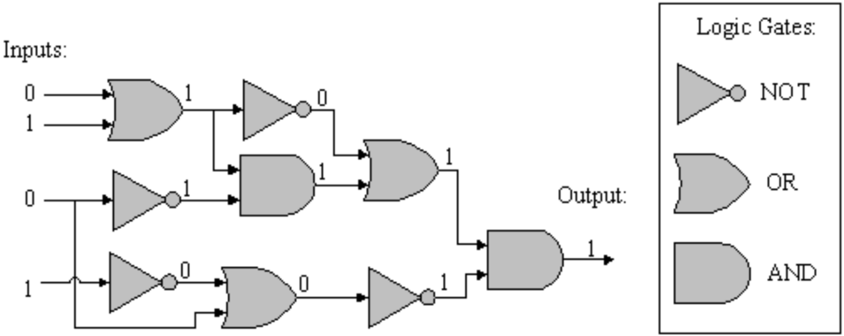
Circuit Satisfiability



Circuit Satisfiability

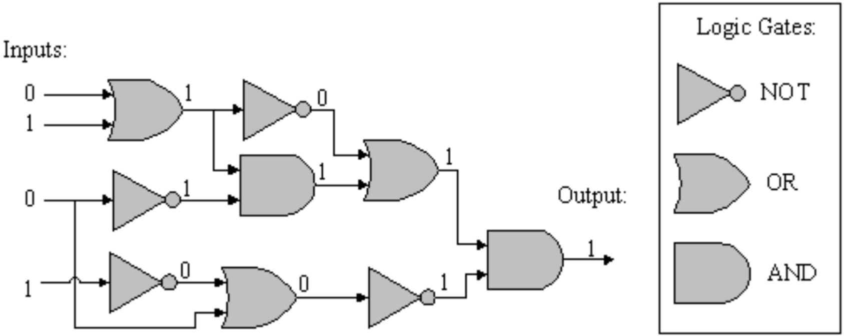


Circuit Satisfiability



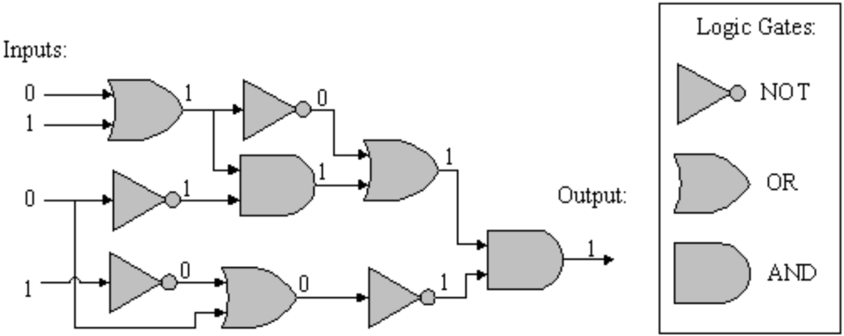
Input: Boolean circuit C on inputs $y_1, y_2, \ldots, y_m$

Circuit Satisfiability



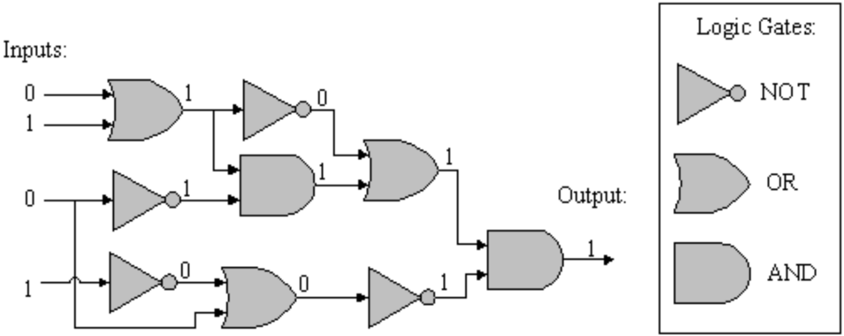
Input: Boolean circuit C on inputs $y_1, y_2, \ldots, y_m$, i.e, DAG where each leaf has label in $\{y_1, y_2, \ldots, y_m\}$, and each other vertex a label in $\{\wedge, \vee, \neg\}$

Circuit Satisfiability



Input: Boolean circuit C on inputs $y_1, y_2, \ldots, y_m$, i.e, DAG where each leaf has label in $\{y_1, y_2, \ldots, y_m\}$, and each other vertex a label in $\{\wedge, \vee, \neg\}$; output vertex

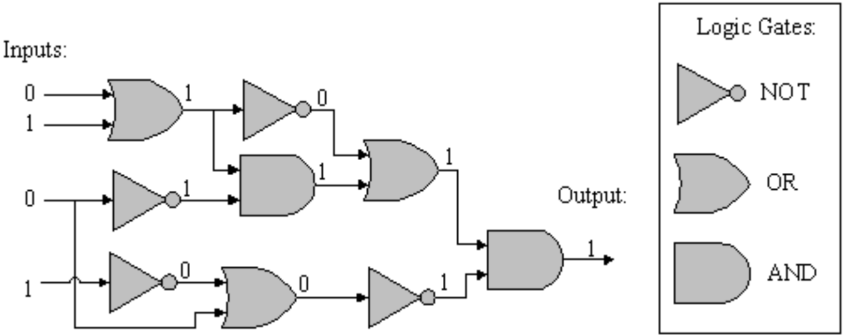
Circuit Satisfiability



Input: Boolean circuit C on inputs $y_1, y_2, \ldots, y_m$, i.e, DAG where each leaf has label in $\{y_1, y_2, \ldots, y_m\}$, and each other vertex a label in $\{\wedge, \vee, \neg\}$; output vertex

Output: satisfying assignment, i.e., a setting of the inputs of C to true/false that makes the output of C true

Circuit Satisfiability



Input: Boolean circuit C on inputs $y_1, y_2, \ldots, y_m$, i.e, DAG where each leaf has label in $\{y_1, y_2, \ldots, y_m\}$, and each other vertex a label in $\{\wedge, \vee, \neg\}$; output vertex

Output: satisfying assignment, i.e., a setting of the inputs of C to true/false that makes the output of C true; or report that none exists.

Circuit-SAT is NP-Hard

Circuit-SAT is NP-Hard

Proof

Circuit-SAT is NP-Hard

Proof

- Need to show $A \leq^p$ Circuit-SAT for each A in NP.

Circuit-SAT is NP-Hard

Proof

- Need to show $A \leq^p$ Circuit-SAT for each A in NP.
- A is specified by $c \in \mathbb{N}$ and $V \in \mathrm{P}$ such that valid solutions on input $x \in \{0,1\}^n$ are $S_x \doteq \{y \in \{0,1\}^{n^c} : V(x,y) = 1\}$.

Circuit-SAT is NP-Hard

Proof

- Need to show $A \leq^p$ Circuit-SAT for each A in NP.
- A is specified by $c \in \mathbb{N}$ and $V \in \mathrm{P}$ such that valid solutions on input $x \in \{0,1\}^n$ are $S_x \doteq \{y \in \{0,1\}^{n^c} : V(x,y) = 1\}$.

Lemma

A Boolean circuit C_x on $m \doteq n^c$ inputs such that $C_x(y) = V(x,y)$ for all $y \in \{0,1\}^m$ can be constructed in time $n^{O(1)}$.

Circuit-SAT is NP-Hard

Proof

- Need to show $A \leq^p$ Circuit-SAT for each A in NP.
- A is specified by $c \in \mathbb{N}$ and $V \in \mathrm{P}$ such that valid solutions on input $x \in \{0,1\}^n$ are $S_x \doteq \{y \in \{0,1\}^{n^c} : V(x,y) = 1\}$.

Lemma

A Boolean circuit C_x on $m \doteq n^c$ inputs such that $C_x(y) = V(x,y)$ for all $y \in \{0,1\}^m$ can be constructed in time $n^{O(1)}$.

Reduction

Circuit-SAT is NP-Hard

Proof

- Need to show $A \leq^p$ Circuit-SAT for each A in NP.
- A is specified by $c \in \mathbb{N}$ and $V \in \mathrm{P}$ such that valid solutions on input $x \in \{0,1\}^n$ are $S_x \doteq \{y \in \{0,1\}^{n^c} : V(x,y) = 1\}$.

Lemma

A Boolean circuit C_x on $m \doteq n^c$ inputs such that $C_x(y) = V(x,y)$ for all $y \in \{0,1\}^m$ can be constructed in time $n^{O(1)}$.

Reduction

A Circuit-SAT

Circuit-SAT is NP-Hard

Proof

- Need to show $A \leq^p$ Circuit-SAT for each A in NP.
- A is specified by $c \in \mathbb{N}$ and $V \in$ P such that valid solutions on input $x \in \{0,1\}^n$ are $S_x \doteq \{y \in \{0,1\}^{n^c} : V(x,y) = 1\}$.

Lemma

A Boolean circuit C_x on $m \doteq n^c$ inputs such that $C_x(y) = V(x,y)$ for all $y \in \{0,1\}^m$ can be constructed in time $n^{O(1)}$.

Reduction

A	Circuit-SAT
x	

Circuit-SAT is NP-Hard

Proof

- Need to show $A \leq^p$ Circuit-SAT for each A in NP.
- A is specified by $c \in \mathbb{N}$ and $V \in$ P such that valid solutions on input $x \in \{0,1\}^n$ are $S_x \doteq \{y \in \{0,1\}^{n^c} : V(x,y) = 1\}$.

Lemma

A Boolean circuit C_x on $m \doteq n^c$ inputs such that $C_x(y) = V(x,y)$ for all $y \in \{0,1\}^m$ can be constructed in time $n^{O(1)}$.

Reduction

$$\begin{array}{ccc} A & & \text{Circuit-SAT} \\ x & \rightarrow & x' = C_x \end{array}$$

Circuit-SAT is NP-Hard

Proof

- Need to show $A \leq^p$ Circuit-SAT for each A in NP.
- A is specified by $c \in \mathbb{N}$ and $V \in$ P such that valid solutions on input $x \in \{0,1\}^n$ are $S_x \doteq \{y \in \{0,1\}^{n^c} : V(x,y) = 1\}$.

Lemma

A Boolean circuit C_x on $m \doteq n^c$ inputs such that $C_x(y) = V(x,y)$ for all $y \in \{0,1\}^m$ can be constructed in time $n^{O(1)}$.

Reduction

$$\begin{array}{ccl} A & & \text{Circuit-SAT} \\ x & \to & x' = C_x \\ & & \downarrow \text{[blackbox]} \\ & & y' \text{ with } C_x(y') = 1 \end{array}$$

Circuit-SAT is NP-Hard

Proof

- Need to show $A \leq^p$ Circuit-SAT for each A in NP.
- A is specified by $c \in \mathbb{N}$ and $V \in \mathrm{P}$ such that valid solutions on input $x \in \{0,1\}^n$ are $S_x \doteq \{y \in \{0,1\}^{n^c} : V(x,y) = 1\}$.

Lemma

A Boolean circuit C_x on $m \doteq n^c$ inputs such that $C_x(y) = V(x,y)$ for all $y \in \{0,1\}^m$ can be constructed in time $n^{O(1)}$.

Reduction

$$\begin{array}{ccl} & A & \text{Circuit-SAT} \\ & x & \rightarrow \quad x' = C_x \\ & & \downarrow \text{[blackbox]} \\ y = y' & \leftarrow & y' \text{ with } C_x(y') = 1 \end{array}$$

Satisfiability

Satisfiability

Specification

Input: Boolean formula φ

Satisfiability

Specification

Input: Boolean formula φ

Output: satisfying assignment of φ, or report that none exists

Satisfiability

Specification

Input: Boolean formula φ

Output: satisfying assignment of φ, or report that none exists

Restrictions

- CNF-SAT: φ is CNF, i.e., a conjunction of clauses.

Satisfiability

Specification

Input: Boolean formula φ

Output: satisfying assignment of φ, or report that none exists

Restrictions

- CNF-SAT: φ is CNF, i.e., a conjunction of clauses.
  - Clause: disjunction of literals

Satisfiability

Specification

Input: Boolean formula φ

Output: satisfying assignment of φ, or report that none exists

Restrictions

- CNF-SAT: φ is CNF, i.e., a conjunction of clauses.
  - Clause: disjunction of literals
  - Literal: variable or negated variable

Satisfiability

Specification

Input: Boolean formula φ

Output: satisfying assignment of φ, or report that none exists

Restrictions

- CNF-SAT: φ is CNF, i.e., a conjunction of clauses.
  - Clause: disjunction of literals
  - Literal: variable or negated variable
- k-SAT for fixed $k \in \mathbb{N}$: φ is k-CNF, i.e., each clause contains at most k literals.

Satisfiability

Specification

Input: Boolean formula φ

Output: satisfying assignment of φ, or report that none exists

Restrictions

- CNF-SAT: φ is CNF, i.e., a conjunction of clauses.
  - Clause: disjunction of literals
  - Literal: variable or negated variable
- k-SAT for fixed $k \in \mathbb{N}$: φ is k-CNF, i.e., each clause contains at most k literals.

$$\ell_1 \vee \ell_2 \vee \cdots \vee \ell_{k-1} \vee \ell_k$$

Satisfiability

Specification

Input: Boolean formula φ

Output: satisfying assignment of φ, or report that none exists

Restrictions

- CNF-SAT: φ is CNF, i.e., a conjunction of clauses.
  - Clause: disjunction of literals
  - Literal: variable or negated variable
- k-SAT for fixed $k \in \mathbb{N}$: φ is k-CNF, i.e., each clause contains at most k literals.

$$\ell_1 \vee \ell_2 \vee \cdots \vee \ell_{k-1} \vee \ell_k \quad \equiv \quad \overline{\ell_1} \wedge \overline{\ell_2} \wedge \overline{\ell_{k-1}} \Rightarrow \ell_k$$

Satisfiability

Specification

Input: Boolean formula φ

Output: satisfying assignment of φ, or report that none exists

Restrictions

- CNF-SAT: φ is CNF, i.e., a conjunction of clauses.
  - Clause: disjunction of literals
  - Literal: variable or negated variable
- k-SAT for fixed $k \in \mathbb{N}$: φ is k-CNF, i.e., each clause contains at most k literals.

$$\ell_1 \vee \ell_2 \vee \cdots \vee \ell_{k-1} \vee \ell_k \quad \equiv \quad \overline{\ell_1} \wedge \overline{\ell_2} \wedge \overline{\ell_{k-1}} \Rightarrow \ell_k$$

Results

Satisfiability

Specification

Input: Boolean formula φ

Output: satisfying assignment of φ, or report that none exists

Restrictions

- CNF-SAT: φ is CNF, i.e., a conjunction of clauses.
  - Clause: disjunction of literals
  - Literal: variable or negated variable
- k-SAT for fixed $k \in \mathbb{N}$: φ is k-CNF, i.e., each clause contains at most k literals.

$$\ell_1 \vee \ell_2 \vee \cdots \vee \ell_{k-1} \vee \ell_k \quad \equiv \quad \overline{\ell_1} \wedge \overline{\ell_2} \wedge \overline{\ell_{k-1}} \Rightarrow \ell_k$$

Results

- 3-SAT is NP-hard.

Satisfiability

Specification

Input: Boolean formula φ

Output: satisfying assignment of φ, or report that none exists

Restrictions

- CNF-SAT: φ is CNF, i.e., a conjunction of clauses.
  - Clause: disjunction of literals
  - Literal: variable or negated variable
- k-SAT for fixed $k \in \mathbb{N}$: φ is k-CNF, i.e., each clause contains at most k literals.

$$\ell_1 \vee \ell_2 \vee \cdots \vee \ell_{k-1} \vee \ell_k \quad \equiv \quad \overline{\ell_1} \wedge \overline{\ell_2} \wedge \overline{\ell_{k-1}} \Rightarrow \ell_k$$

Results

- 3-SAT is NP-hard.
- 2-SAT can be solved in polynomial time.

Proving NP-Hardness

Proving NP-Hardness

Strategy

Proving NP-Hardness

Strategy

To show a new problem C is NP-hard:

Proving NP-Hardness

Strategy

To show a new problem C is NP-hard:

- Find a known NP-complete problem B.

Proving NP-Hardness

Strategy

To show a new problem C is NP-hard:

- Find a known NP-complete problem B.
- Show that $B \leq^p C$.

Proving NP-Hardness

Strategy

To show a new problem C is NP-hard:

- Find a known NP-complete problem B.
- Show that $B \leq^p C$.

Justification

Proving NP-Hardness

Strategy

To show a new problem C is NP-hard:

- Find a known NP-complete problem B.
- Show that $B \leq^p C$.

Justification

Consider any $A \in \mathrm{NP}$.

Proving NP-Hardness

Strategy

To show a new problem C is NP-hard:

- Find a known NP-complete problem B.
- Show that $B \leq^p C$.

Justification

Consider any $A \in \mathrm{NP}$.

- By the NP-hardness of B, $A \leq^p B$.

Proving NP-Hardness

Strategy

To show a new problem C is NP-hard:

- Find a known NP-complete problem B.
- Show that $B \leq^p C$.

Justification

Consider any $A \in \mathrm{NP}$.

- By the NP-hardness of B, $A \leq^p B$.
- We show that $B \leq^p C$.

Proving NP-Hardness

Strategy

To show a new problem C is NP-hard:

- Find a known NP-complete problem B.
- Show that $B \leq^p C$.

Justification

Consider any $A \in \mathrm{NP}$.

- By the NP-hardness of B, $A \leq^p B$.
- We show that $B \leq^p C$.
- Therefore $A \leq^p B \leq^p C$.

Proving NP-Hardness

Strategy

To show a new problem C is NP-hard:

- Find a known NP-complete problem B.
- Show that $B \leq^p C$.

Justification

Consider any $A \in \text{NP}$.

- By the NP-hardness of B, $A \leq^p B$.
- We show that $B \leq^p C$.
- Therefore $A \leq^p B \leq^p C$.
- By transitivity $A \leq^p C$.

3-SAT is NP-Hard

3-SAT is NP-Hard

Strategy

Mapping reduction Circuit-SAT $\leq^p$ 3-SAT

3-SAT is NP-Hard

Strategy

Mapping reduction Circuit-SAT $\leq^p$ 3-SAT

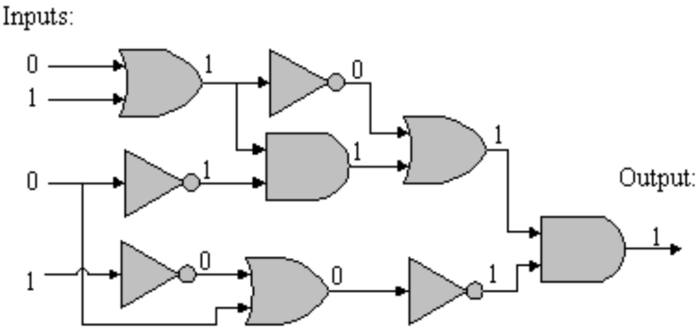
Gadget reduction

3-SAT is NP-Hard

Strategy

Mapping reduction Circuit-SAT $\leq^p$ 3-SAT

Gadget reduction



Reduction Circuit-SAT $\leq^p$ 3-SAT

Reduction Circuit-SAT $\leq^p$ 3-SAT

- Introduce a variable for each input of C.

Reduction Circuit-SAT $\leq^p$ 3-SAT

- Introduce a variable for each input of C.
- Introduce a variable for each gate of C.

Reduction Circuit-SAT $\leq^p$ 3-SAT

- Introduce a variable for each input of C.
- Introduce a variable for each gate of C.
- For each gate g, include clauses with at most 3 literals each that combined force the variable of the gate to the value of the gate on the input.

Reduction Circuit-SAT $\leq^p$ 3-SAT

- Introduce a variable for each input of C.
- Introduce a variable for each gate of C.
- For each gate g, include clauses with at most 3 literals each that combined force the variable of the gate to the value of the gate on the input.
    - $g' =$ NOT g

Reduction Circuit-SAT $\leq^p$ 3-SAT

- Introduce a variable for each input of C.
- Introduce a variable for each gate of C.
- For each gate g, include clauses with at most 3 literals each that combined force the variable of the gate to the value of the gate on the input.
  - $g' = \text{ NOT } g \rightarrow \begin{cases} g \Rightarrow \overline{g'} \\ \overline{g} \Rightarrow g' \end{cases}$

Reduction Circuit-SAT $\leq^p$ 3-SAT

- Introduce a variable for each input of C.
- Introduce a variable for each gate of C.
- For each gate g, include clauses with at most 3 literals each that combined force the variable of the gate to the value of the gate on the input.
  - $g' = \text{ NOT } g \rightarrow \begin{cases} g \Rightarrow \overline{g'} \\ \overline{g} \Rightarrow g' \end{cases} \equiv \begin{cases} \overline{g} \vee \overline{g'} \\ g \vee g' \end{cases}$

Reduction Circuit-SAT $\leq^p$ 3-SAT

- Introduce a variable for each input of C.
- Introduce a variable for each gate of C.
- For each gate g, include clauses with at most 3 literals each that combined force the variable of the gate to the value of the gate on the input.
  - $g' = \text{ NOT } g \rightarrow \begin{cases} g \Rightarrow \overline{g'} \\ \overline{g} \Rightarrow g' \end{cases} \equiv \begin{cases} \overline{g} \vee \overline{g'} \\ g \vee g' \end{cases}$
  - $g' = g_1 \text{ AND } g_2$

Reduction Circuit-SAT $\leq^p$ 3-SAT

- Introduce a variable for each input of C.
- Introduce a variable for each gate of C.
- For each gate g, include clauses with at most 3 literals each that combined force the variable of the gate to the value of the gate on the input.
  - $g' = \text{ NOT } g \rightarrow \begin{cases} g \Rightarrow \overline{g'} \\ \overline{g} \Rightarrow g' \end{cases} \equiv \begin{cases} \overline{g} \vee \overline{g'} \\ g \vee g' \end{cases}$
  - $g' = g_1 \text{ AND } g_2 \rightarrow \begin{cases} \overline{g_1} \Rightarrow \overline{g'} \\ \overline{g_2} \Rightarrow \overline{g'} \\ g_1 \wedge g_2 \Rightarrow g' \end{cases}$

Reduction Circuit-SAT $\leq^p$ 3-SAT

- Introduce a variable for each input of C.
- Introduce a variable for each gate of C.
- For each gate g, include clauses with at most 3 literals each that combined force the variable of the gate to the value of the gate on the input.
  - $g' = \text{ NOT } g \rightarrow \begin{cases} g \Rightarrow \overline{g'} \\ \overline{g} \Rightarrow g' \end{cases} \equiv \begin{cases} \overline{g} \vee \overline{g'} \\ g \vee g' \end{cases}$
  - $g' = g_1 \text{ AND } g_2 \rightarrow \begin{cases} \overline{g_1} \Rightarrow \overline{g'} \\ \overline{g_2} \Rightarrow \overline{g'} \\ g_1 \wedge g_2 \Rightarrow g' \end{cases} \equiv \begin{cases} g_1 \vee \overline{g'} \\ g_2 \vee \overline{g'} \\ \overline{g_1} \vee \overline{g_2} \vee g' \end{cases}$

Reduction Circuit-SAT $\leq^p$ 3-SAT

- Introduce a variable for each input of C.
- Introduce a variable for each gate of C.
- For each gate g, include clauses with at most 3 literals each that combined force the variable of the gate to the value of the gate on the input.
  - $g' = \text{ NOT } g \rightarrow \begin{cases} g \Rightarrow \overline{g'} \\ \overline{g} \Rightarrow g' \end{cases} \equiv \begin{cases} \overline{g} \vee \overline{g'} \\ g \vee g' \end{cases}$
  - $g' = g_1 \text{ AND } g_2 \rightarrow \begin{cases} \overline{g_1} \Rightarrow \overline{g'} \\ \overline{g_2} \Rightarrow \overline{g'} \\ g_1 \wedge g_2 \Rightarrow g' \end{cases} \equiv \begin{cases} g_1 \vee \overline{g'} \\ g_2 \vee \overline{g'} \\ \overline{g_1} \vee \overline{g_2} \vee g' \end{cases}$
- Add unit clause consisting of the variable for the output gate.

Reduction Circuit-SAT $\leq^p$ 3-SAT

- Introduce a variable for each input of C.
- Introduce a variable for each gate of C.
- For each gate g, include clauses with at most 3 literals each that combined force the variable of the gate to the value of the gate on the input.
  - $g' = \text{ NOT } g \to \begin{cases} g \Rightarrow \overline{g'} \\ \overline{g} \Rightarrow g' \end{cases} \equiv \begin{cases} \overline{g} \vee \overline{g'} \\ g \vee g' \end{cases}$
  - $g' = g_1 \text{ AND } g_2 \to \begin{cases} \overline{g_1} \Rightarrow \overline{g'} \\ \overline{g_2} \Rightarrow \overline{g'} \\ g_1 \wedge g_2 \Rightarrow g' \end{cases} \equiv \begin{cases} g_1 \vee \overline{g'} \\ g_2 \vee \overline{g'} \\ \overline{g_1} \vee \overline{g_2} \vee g' \end{cases}$
- Add unit clause consisting of the variable for the output gate.

Correctness

Reduction Circuit-SAT $\leq^p$ 3-SAT

- Introduce a variable for each input of C.
- Introduce a variable for each gate of C.
- For each gate g, include clauses with at most 3 literals each that combined force the variable of the gate to the value of the gate on the input.
  - $g' = \text{ NOT } g \rightarrow \begin{cases} g \Rightarrow \overline{g'} \\ \overline{g} \Rightarrow g' \end{cases} \equiv \begin{cases} \overline{g} \vee \overline{g'} \\ g \vee g' \end{cases}$
  - $g' = g_1 \text{ AND } g_2 \rightarrow \begin{cases} \overline{g_1} \Rightarrow \overline{g'} \\ \overline{g_2} \Rightarrow \overline{g'} \\ g_1 \wedge g_2 \Rightarrow g' \end{cases} \equiv \begin{cases} g_1 \vee \overline{g'} \\ g_2 \vee \overline{g'} \\ \overline{g_1} \vee \overline{g_2} \vee g' \end{cases}$
- Add unit clause consisting of the variable for the output gate.

Correctness

- C has a satisfying assignment
  $\Leftrightarrow \varphi$ has a satisfying assignment.

Reduction Circuit-SAT $\leq^p$ 3-SAT

- Introduce a variable for each input of C.
- Introduce a variable for each gate of C.
- For each gate g, include clauses with at most 3 literals each that combined force the variable of the gate to the value of the gate on the input.
  - $g' = \text{ NOT } g \to \begin{cases} g \Rightarrow \overline{g'} \\ \overline{g} \Rightarrow g' \end{cases} \equiv \begin{cases} \overline{g} \vee \overline{g'} \\ g \vee g' \end{cases}$
  - $g' = g_1 \text{ AND } g_2 \to \begin{cases} \overline{g_1} \Rightarrow \overline{g'} \\ \overline{g_2} \Rightarrow \overline{g'} \\ g_1 \wedge g_2 \Rightarrow g' \end{cases} \equiv \begin{cases} g_1 \vee \overline{g'} \\ g_2 \vee \overline{g'} \\ \overline{g_1} \vee \overline{g_2} \vee g' \end{cases}$
- Add unit clause consisting of the variable for the output gate.

Correctness

- C has a satisfying assignment
  $\Leftrightarrow \varphi$ has a satisfying assignment.
- Each satisfying assignment for φ includes one for C.

Reduction Circuit-SAT $\leq^p$ 3-SAT

- Introduce a variable for each input of C.
- Introduce a variable for each gate of C.
- For each gate g, include clauses with at most 3 literals each that combined force the variable of the gate to the value of the gate on the input.
  - $g' =$ NOT $g \rightarrow \begin{cases} g \Rightarrow \overline{g'} \\ \overline{g} \Rightarrow g' \end{cases} \equiv \begin{cases} \overline{g} \vee \overline{g'} \\ g \vee g' \end{cases}$
  - $g' = g_1$ AND $g_2 \rightarrow \begin{cases} \overline{g_1} \Rightarrow \overline{g'} \\ \overline{g_2} \Rightarrow \overline{g'} \\ g_1 \wedge g_2 \Rightarrow g' \end{cases} \equiv \begin{cases} g_1 \vee \overline{g'} \\ g_2 \vee \overline{g'} \\ \overline{g_1} \vee \overline{g_2} \vee g' \end{cases}$
- Add unit clause consisting of the variable for the output gate.

Correctness

- C has a satisfying assignment
  $\Leftrightarrow \varphi$ has a satisfying assignment.
- Each satisfying assignment for φ includes one for C.

Polynomial running time

Polynomial-Time Algorithm for 2-SAT

Polynomial-Time Algorithm for 2-SAT

Reduction to digraph reachability

Polynomial-Time Algorithm for 2-SAT

Reduction to digraph reachability

Construction of the digraph G

Polynomial-Time Algorithm for 2-SAT

Reduction to digraph reachability

Construction of the digraph G

- Introduce a vertex for each variable x_i that occurs in φ, and another one for its negation $\overline{x_i}$.

Polynomial-Time Algorithm for 2-SAT

Reduction to digraph reachability

Construction of the digraph G

- Introduce a vertex for each variable x_i that occurs in φ, and another one for its negation $\overline{x_i}$.
- Interpret each clause $\ell_1 \vee \ell_2$ as the implications $\overline{\ell_1} \Rightarrow \ell_2$ and $\overline{\ell_2} \Rightarrow \ell_1$. Include edges $(\overline{\ell_1}, \ell_2)$ and $(\overline{\ell_2}, \ell_1)$ in G.

Polynomial-Time Algorithm for 2-SAT

Reduction to digraph reachability

Construction of the digraph G

- Introduce a vertex for each variable x_i that occurs in φ, and another one for its negation $\overline{x_i}$.
- Interpret each clause $\ell_1 \vee \ell_2$ as the implications $\overline{\ell_1} \Rightarrow \ell_2$ and $\overline{\ell_2} \Rightarrow \ell_1$. Include edges $(\overline{\ell_1}, \ell_2)$ and $(\overline{\ell_2}, \ell_1)$ in G.

Claim

φ has a satisfying assignment
$\Leftrightarrow$ for no variable x_i there are paths $x_i \rightsquigarrow \overline{x_i}$ and $\overline{x_i} \rightsquigarrow x_i$ in G.

Polynomial-Time Algorithm for 2-SAT

Reduction to digraph reachability

Construction of the digraph G

- Introduce a vertex for each variable x_i that occurs in φ, and another one for its negation $\overline{x_i}$.
- Interpret each clause $\ell_1 \vee \ell_2$ as the implications $\overline{\ell_1} \Rightarrow \ell_2$ and $\overline{\ell_2} \Rightarrow \ell_1$. Include edges $(\overline{\ell_1}, \ell_2)$ and $(\overline{\ell_2}, \ell_1)$ in G.

Claim

φ has a satisfying assignment
$\Leftrightarrow$ for no variable x_i there are paths $x_i \rightsquigarrow \overline{x_i}$ and $\overline{x_i} \rightsquigarrow x_i$ in G.

Proof

$\Rightarrow$ By contraposition.

Polynomial-Time Algorithm for 2-SAT

Reduction to digraph reachability

Construction of the digraph G

- Introduce a vertex for each variable x_i that occurs in φ, and another one for its negation $\overline{x_i}$.
- Interpret each clause $\ell_1 \vee \ell_2$ as the implications $\overline{\ell_1} \Rightarrow \ell_2$ and $\overline{\ell_2} \Rightarrow \ell_1$. Include edges $(\overline{\ell_1}, \ell_2)$ and $(\overline{\ell_2}, \ell_1)$ in G.

Claim

φ has a satisfying assignment
$\Leftrightarrow$ for no variable x_i there are paths $x_i \rightsquigarrow \overline{x_i}$ and $\overline{x_i} \rightsquigarrow x_i$ in G.

Proof

$\Rightarrow$ By contraposition.

$\Leftarrow$ Pick a literal ℓ such that there is no path $\ell \rightsquigarrow \overline{\ell}$ in G, set ℓ to true, simplify the formula/digraph, and repeat until there are no variables left.