More Reductions

Abhiram Ranade

March 28, 2016

Outline for today:

Outline for today:

Today we prove:

$IS \leq_K ILP \leq_K CSAT \leq_K CNFSAT \leq_K IS$ Decision versions

Outline for today:

Today we prove:
$IS \leq_K ILP \leq_K CSAT \leq_K CNFSAT \leq_K IS$ Decision versions

CSAT : "Circuit satisfiability"

Outline for today:

Today we prove:

$IS \leq_K ILP \leq_K CSAT \leq_K CNFSAT \leq_K IS$ Decision versions

CSAT : "Circuit satisfiability"

CNFSAT : "Conjunctive normal form satisfiability"

Outline for today:

Today we prove:
$IS \leq_K ILP \leq_K CSAT \leq_K CNFSAT \leq_K IS$ Decision versions

CSAT : "Circuit satisfiability"

CNFSAT : "Conjunctive normal form satisfiability"

Thus all above problems are equivalent for the purposes of designing polynomial time algorithms.

(0-1)ILP : The decision version

(0-1)ILP : The decision version

Original Input: $m \times n$ Matrix A, m vector b, n vector c.

(0-1)ILP : The decision version

Original Input: $m \times n$ Matrix A, m vector b, n vector c.
0-1 Optimization problem: $\max c^T x$ s.t. $x \in \{0, 1\}^n$, $Ax \leq b$

(0-1)ILP : The decision version

Original Input: $m \times n$ Matrix A, m vector b, n vector c.
0-1 Optimization problem: $\max c^T x$ s.t. $x \in \{0, 1\}^n$, $Ax \leq b$

Natural existence version: Does there exist an $x \in \{0, 1\}^n$ s.t. $Ax \leq b$ and $c^T x \geq t$, where t is an additional input.

(0-1)ILP : The decision version

Original Input: $m \times n$ Matrix A, m vector b, n vector c.
0-1 Optimization problem: $\max c^T x$ s.t. $x \in \{0,1\}^n$, $Ax \leq b$

Natural existence version: Does there exist an $x \in \{0,1\}^n$ s.t. $Ax \leq b$ and $c^T x \geq t$, where t is an additional input.

But the condition $c^T x \geq t$ can be included into $Ax \leq b$

(0-1)ILP : The decision version

Original Input: $m \times n$ Matrix A, m vector b, n vector c.
0-1 Optimization problem: $\max c^T x$ s.t. $x \in \{0,1\}^n$, $Ax \leq b$

Natural existence version: Does there exist an $x \in \{0,1\}^n$ s.t. $Ax \leq b$ and $c^T x \geq t$, where t is an additional input.

But the condition $c^T x \geq t$ can be included into $Ax \leq b$
Add row $-c^T$ to A, and append $-t$ to b.

(0-1)ILP : The decision version

Original Input: $m \times n$ Matrix A, m vector b, n vector c.
0-1 Optimization problem: $\max c^T x$ s.t. $x \in \{0,1\}^n$, $Ax \leq b$

Natural existence version: Does there exist an $x \in \{0,1\}^n$ s.t. $Ax \leq b$ and $c^T x \geq t$, where t is an additional input.

But the condition $c^T x \geq t$ can be included into $Ax \leq b$
Add row $-c^T$ to A, and append $-t$ to b.

(0-1) ILP Existence version:

(0-1)ILP : The decision version

Original Input: $m \times n$ Matrix A, m vector b, n vector c.
0-1 Optimization problem: $\max c^T x$ s.t. $x \in \{0,1\}^n$, $Ax \leq b$

Natural existence version: Does there exist an $x \in \{0,1\}^n$ s.t. $Ax \leq b$ and $c^T x \geq t$, where t is an additional input.

But the condition $c^T x \geq t$ can be included into $Ax \leq b$
Add row $-c^T$ to A, and append $-t$ to b.

(0-1) ILP Existence version:
Input: Matrix A, vector b.

(0-1)ILP : The decision version

Original Input: $m \times n$ Matrix A, m vector b, n vector c.
0-1 Optimization problem: $\max c^T x$ s.t. $x \in \{0,1\}^n$, $Ax \leq b$

Natural existence version: Does there exist an $x \in \{0,1\}^n$ s.t. $Ax \leq b$ and $c^T x \geq t$, where t is an additional input.

But the condition $c^T x \geq t$ can be included into $Ax \leq b$
Add row $-c^T$ to A, and append $-t$ to b.

(0-1) ILP Existence version:
Input: Matrix A, vector b.
Output: Does there exist $x \in \{0,1\}^n$ s.t. $Ax \leq b$?

(0-1)ILP : The decision version

Original Input: $m \times n$ Matrix A, m vector b, n vector c.
0-1 Optimization problem: $\max c^T x$ s.t. $x \in \{0,1\}^n$, $Ax \leq b$

Natural existence version: Does there exist an $x \in \{0,1\}^n$ s.t. $Ax \leq b$ and $c^T x \geq t$, where t is an additional input.

But the condition $c^T x \geq t$ can be included into $Ax \leq b$
Add row $-c^T$ to A, and append $-t$ to b.

(0-1) ILP Existence version:
Input: Matrix A, vector b.
Output: Does there exist $x \in \{0,1\}^n$ s.t. $Ax \leq b$?

Note:(0-1) ILP-Existence problem is a decision problem.

IS $\leq_K$ ILP

IS(G,k) : Does a graph G have an independent set of size k?
ILP(A,b): Does there exist a 0-1 vector x s.t. $Ax \leq b$?

IS $\leq_K$ ILP

IS(G,k) : Does a graph G have an independent set of size k?
ILP(A,b): Does there exist a 0-1 vector x s.t. $Ax \leq b$?

Instance map:

IS $\leq_K$ ILP

IS(G,k) : Does a graph G have an independent set of size k?
ILP(A,b): Does there exist a 0-1 vector x s.t. $Ax \leq b$?

Instance map:
x_u : 0-1 variable for each vertex u.

IS $\leq_K$ ILP

IS(G,k) : Does a graph G have an independent set of size k?
ILP(A,b): Does there exist a 0-1 vector x s.t. $Ax \leq b$?

Instance map:
x_u : 0-1 variable for each vertex u.
$\forall (u, v) \in E(G) : x_u + x_v \leq 1$

IS $\leq_K$ ILP

IS(G,k) : Does a graph G have an independent set of size k?
ILP(A,b): Does there exist a 0-1 vector x s.t. $Ax \leq b$?

Instance map:
x_u : 0-1 variable for each vertex u.
$\forall (u, v) \in E(G) : x_u + x_v \leq 1$
$\sum_u x_u = k$

IS $\leq_K$ ILP

IS(G,k) : Does a graph G have an independent set of size k?
ILP(A,b): Does there exist a 0-1 vector x s.t. $Ax \leq b$?

Instance map:
x_u : 0-1 variable for each vertex u.
$\forall (u,v) \in E(G) : x_u + x_v \leq 1$
$\sum_u x_u = k$ Equivalent: $\sum_u x_u \leq k, \quad -\sum_u x_u \leq -k$

IS $\leq_K$ ILP

IS(G,k) : Does a graph G have an independent set of size k?
ILP(A,b): Does there exist a 0-1 vector x s.t. $Ax \leq b$?

Instance map:
x_u : 0-1 variable for each vertex u.
$\forall (u, v) \in E(G) : x_u + x_v \leq 1$
$\sum_u x_u = k$ Equivalent: $\sum_u x_u \leq k, \quad -\sum_u x_u \leq -k$
Form A, b from the above inequalities.

IS $\leq_K$ ILP

IS(G,k) : Does a graph G have an independent set of size k?
ILP(A,b): Does there exist a 0-1 vector x s.t. $Ax \leq b$?

Instance map:
x_u : 0-1 variable for each vertex u.
$\forall (u, v) \in E(G) : x_u + x_v \leq 1$
$\sum_u x_u = k$ Equivalent: $\sum_u x_u \leq k, \quad -\sum_u x_u \leq -k$
Form A, b from the above inequalities.

Instance map function runs in time polynomial in size of G.

IS $\leq_K$ ILP

IS(G,k) : Does a graph G have an independent set of size k?
ILP(A,b): Does there exist a 0-1 vector x s.t. $Ax \leq b$?

Instance map:
x_u : 0-1 variable for each vertex u.
$\forall (u, v) \in E(G) : x_u + x_v \leq 1$
$\sum_u x_u = k$ Equivalent: $\sum_u x_u \leq k, \quad -\sum_u x_u \leq -k$
Form A, b from the above inequalities.

Instance map function runs in time polynomial in size of G.

IS(G,k) = true $\Rightarrow$ ILP(A,b) = true.

IS $\leq_K$ ILP

IS(G,k) : Does a graph G have an independent set of size k?
ILP(A,b): Does there exist a 0-1 vector x s.t. $Ax \leq b$?

Instance map:
x_u : 0-1 variable for each vertex u.
$\forall(u, v) \in E(G) : x_u + x_v \leq 1$
$\sum_u x_u = k$ Equivalent: $\sum_u x_u \leq k, \quad -\sum_u x_u \leq -k$
Form A, b from the above inequalities.

Instance map function runs in time polynomial in size of G.

IS(G,k) = true $\Rightarrow$ ILP(A,b) = true.

ILP(A,b) = true $\Rightarrow$ IS(G,k) = true.

Circuit Satisfiability

Circuit Satisfiability

Input: C = a combinational circuit with inputs $1, \ldots, n$ and a single output.

Circuit Satisfiability

Input: $C =$ a combinational circuit with inputs $1, \ldots, n$ and a single output.

C is specified by giving its DAG, and vertex labels: AND, OR, NOT

Circuit Satisfiability

Input: $C =$ a combinational circuit with inputs $1, \ldots, n$ and a single output.

C is specified by giving its DAG, and vertex labels: AND, OR, NOT

Output: Does there exist $z = (z_1, \ldots, z_n) \in \{0, 1\}^n$ s.t. if for all i input i is set to value z_i the output will take the value true.

0 means false/NO, 1 means true/YES.

Circuit Satisfiability

Input: C = a combinational circuit with inputs $1, \ldots, n$ and a single output.

C is specified by giving its DAG, and vertex labels: AND, OR, NOT

Output: Does there exist $z = (z_1, \ldots, z_n) \in \{0,1\}^n$ s.t. if for all i input i is set to value z_i the output will take the value true.

0 means false/NO, 1 means true/YES.

If such a z exists, then C is said to be satisfiable.

Circuit Satisfiability

Input: C = a combinational circuit with inputs $1, \ldots, n$ and a single output.

C is specified by giving its DAG, and vertex labels: AND, OR, NOT

Output: Does there exist $z = (z_1, \ldots, z_n) \in \{0, 1\}^n$ s.t. if for all i input i is set to value z_i the output will take the value true.

0 means false/NO, 1 means true/YES.

If such a z exists, then C is said to be satisfiable.
If no such z exists, then C is unsatisfiable.

$ILP \leq_K CSAT$

$ILP \leq_K CSAT$

ILP(A,b) : Does there exist 0-1 vector x s.t. $Ax \leq b$.

$ILP \leq_K CSAT$

ILP(A,b) : Does there exist 0-1 vector x s.t. $Ax \leq b$.

CSAT(C) : Does there exist 0-1 vector z which if given as input to C produces output 1?

$ILP \leq_K CSAT$

ILP(A,b) : Does there exist 0-1 vector x s.t. $Ax \leq b$.

CSAT(C) : Does there exist 0-1 vector z which if given as input to C produces output 1?

Reduction idea: (1) $z \equiv x$.

$ILP \leq_K CSAT$

ILP(A,b) : Does there exist 0-1 vector x s.t. $Ax \leq b$.
CSAT(C) : Does there exist 0-1 vector z which if given as input to C produces output 1?

Reduction idea: (1) $z \equiv x$. (2) Make the circuit check $Az \leq b$.

$ILP \leq_K CSAT$

ILP(A,b) : Does there exist 0-1 vector x s.t. $Ax \leq b$.
CSAT(C) : Does there exist 0-1 vector z which if given as input to C produces output 1?

Reduction idea: (1) $z \equiv x$. (2) Make the circuit check $Az \leq b$.

Instance map:

$ILP \leq_K CSAT$

ILP(A,b) : Does there exist 0-1 vector x s.t. $Ax \leq b$.
CSAT(C) : Does there exist 0-1 vector z which if given as input to C produces output 1?

Reduction idea: (1) $z \equiv x$. (2) Make the circuit check $Az \leq b$.

Instance map: IM(A,b){

$ILP \leq_K CSAT$

ILP(A,b) : Does there exist 0-1 vector x s.t. $Ax \leq b$.
CSAT(C) : Does there exist 0-1 vector z which if given as input to C produces output 1?

Reduction idea: (1) $z \equiv x$. (2) Make the circuit check $Az \leq b$.

Instance map: IM(A,b){
Construct a circuit with n inputs. $z_1, \ldots, z_n$ = values on inputs.

$ILP \leq_K CSAT$

ILP(A,b) : Does there exist 0-1 vector x s.t. $Ax \leq b$.
CSAT(C) : Does there exist 0-1 vector z which if given as input to C produces output 1?

Reduction idea: (1) $z \equiv x$. (2) Make the circuit check $Az \leq b$.

Instance map: IM(A,b){
Construct a circuit with n inputs. $z_1, \ldots, z_n =$ values on inputs.
Subcircuit C_1 for checking condition $a_{11}z_1 + a_{12}z_2 + \ldots + a_{1n} \leq b_1$

$ILP \leq_K CSAT$

ILP(A,b) : Does there exist 0-1 vector x s.t. $Ax \leq b$.
CSAT(C) : Does there exist 0-1 vector z which if given as input to C produces output 1?

Reduction idea: (1) $z \equiv x$. (2) Make the circuit check $Az \leq b$.

Instance map: IM(A,b){
Construct a circuit with n inputs. $z_1, \ldots, z_n$ = values on inputs.
Subcircuit C_1 for checking condition $a_{11}z_1 + a_{12}z_2 + \ldots + a_{1n} \leq b_1$
"AND bits of a_{1j} with z_j, add the results and feed to comparator having other input b_1"

$ILP \leq_K CSAT$

ILP(A,b) : Does there exist 0-1 vector x s.t. $Ax \leq b$.
CSAT(C) : Does there exist 0-1 vector z which if given as input to C produces output 1?

Reduction idea: (1) $z \equiv x$. (2) Make the circuit check $Az \leq b$.

Instance map: IM(A,b){
Construct a circuit with n inputs. $z_1, \ldots, z_n =$ values on inputs.
Subcircuit C_1 for checking condition $a_{11}z_1 + a_{12}z_2 + \ldots + a_{1n} \leq b_1$
"AND bits of a_{1j} with z_j, add the results and feed to comparator having other input b_1"
Overall circuit : AND together outputs of all C_i.
}

$ILP \leq_K CSAT$

ILP(A,b) : Does there exist 0-1 vector x s.t. $Ax \leq b$.
CSAT(C) : Does there exist 0-1 vector z which if given as input to C produces output 1?

Reduction idea: (1) $z \equiv x$. (2) Make the circuit check $Az \leq b$.

Instance map: IM(A,b){
Construct a circuit with n inputs. $z_1, \ldots, z_n$ = values on inputs.
Subcircuit C_1 for checking condition $a_{11}z_1 + a_{12}z_2 + \ldots + a_{1n} \leq b_1$
"AND bits of a_{1j} with z_j, add the results and feed to comparator having other input b_1"
Overall circuit : AND together outputs of all C_i.
}

- Construction happens in polytime in n

$ILP \leq_K CSAT$

ILP(A,b) : Does there exist 0-1 vector x s.t. $Ax \leq b$.
CSAT(C) : Does there exist 0-1 vector z which if given as input to C produces output 1?

Reduction idea: (1) $z \equiv x$. (2) Make the circuit check $Az \leq b$.

Instance map: IM(A,b){
Construct a circuit with n inputs. $z_1, \ldots, z_n$ = values on inputs.
Subcircuit C_1 for checking condition $a_{11}z_1 + a_{12}z_2 + \ldots + a_{1n} \leq b_1$
"AND bits of a_{1j} with z_j, add the results and feed to comparator having other input b_1"
Overall circuit : AND together outputs of all C_i.
}

- Construction happens in polytime in n
- ILP(A,b) has a solution x

$ILP \leq_K CSAT$

ILP(A,b) : Does there exist 0-1 vector x s.t. $Ax \leq b$.
CSAT(C) : Does there exist 0-1 vector z which if given as input to C produces output 1?

Reduction idea: (1) $z \equiv x$. (2) Make the circuit check $Az \leq b$.

Instance map: IM(A,b){
Construct a circuit with n inputs. $z_1, \ldots, z_n$ = values on inputs.
Subcircuit C_1 for checking condition $a_{11}z_1 + a_{12}z_2 + \ldots + a_{1n} \leq b_1$
"AND bits of a_{1j} with z_j, add the results and feed to comparator having other input b_1"
Overall circuit : AND together outputs of all C_i.
}

- Construction happens in polytime in n
- ILP(A,b) has a solution x
  $\Rightarrow$ if x = circuit input, then output = 1.

$ILP \leq_K CSAT$

ILP(A,b) : Does there exist 0-1 vector x s.t. $Ax \leq b$.
CSAT(C) : Does there exist 0-1 vector z which if given as input to C produces output 1?

Reduction idea: (1) $z \equiv x$. (2) Make the circuit check $Az \leq b$.

Instance map: IM(A,b){
Construct a circuit with n inputs. $z_1, \ldots, z_n =$ values on inputs.
Subcircuit C_1 for checking condition $a_{11}z_1 + a_{12}z_2 + \ldots + a_{1n} \leq b_1$
"AND bits of a_{1j} with z_j, add the results and feed to comparator having other input b_1"
Overall circuit : AND together outputs of all C_i.
}

- Construction happens in polytime in n
- ILP(A,b) has a solution x
  $\Rightarrow$ if $x =$ circuit input, then output $= 1$. $\Rightarrow$ C is satisfiable

$ILP \leq_K CSAT$

ILP(A,b) : Does there exist 0-1 vector x s.t. $Ax \leq b$.
CSAT(C) : Does there exist 0-1 vector z which if given as input to C produces output 1?

Reduction idea: (1) $z \equiv x$. (2) Make the circuit check $Az \leq b$.

Instance map: IM(A,b){
Construct a circuit with n inputs. $z_1, \ldots, z_n$ = values on inputs.
Subcircuit C_1 for checking condition $a_{11}z_1 + a_{12}z_2 + \ldots + a_{1n} \leq b_1$
"AND bits of a_{1j} with z_j, add the results and feed to comparator having other input b_1"
Overall circuit : AND together outputs of all C_i.
}

- Construction happens in polytime in n
- ILP(A,b) has a solution x
  $\Rightarrow$ if x = circuit input, then output = 1. $\Rightarrow$ C is satisfiable
- C is satisfiable for inputs z

$ILP \leq_K CSAT$

ILP(A,b) : Does there exist 0-1 vector x s.t. $Ax \leq b$.
CSAT(C) : Does there exist 0-1 vector z which if given as input to C produces output 1?

Reduction idea: (1) $z \equiv x$. (2) Make the circuit check $Az \leq b$.

Instance map: IM(A,b){
Construct a circuit with n inputs. $z_1, \ldots, z_n$ = values on inputs.
Subcircuit C_1 for checking condition $a_{11}z_1 + a_{12}z_2 + \ldots + a_{1n} \leq b_1$
"AND bits of a_{1j} with z_j, add the results and feed to comparator having other input b_1"
Overall circuit : AND together outputs of all C_i.
}

- Construction happens in polytime in n
- ILP(A,b) has a solution x
  $\Rightarrow$ if x = circuit input, then output = 1. $\Rightarrow$ C is satisfiable
- C is satisfiable for inputs z
  $\Rightarrow Az \leq b$

$ILP \leq_K CSAT$

ILP(A,b) : Does there exist 0-1 vector x s.t. $Ax \leq b$.
CSAT(C) : Does there exist 0-1 vector z which if given as input to C produces output 1?

Reduction idea: (1) $z \equiv x$. (2) Make the circuit check $Az \leq b$.

Instance map: IM(A,b){
Construct a circuit with n inputs. $z_1, \ldots, z_n$ = values on inputs.
Subcircuit C_1 for checking condition $a_{11}z_1 + a_{12}z_2 + \ldots + a_{1n} \leq b_1$
"AND bits of a_{1j} with z_j, add the results and feed to comparator having other input b_1"
Overall circuit : AND together outputs of all C_i.
}

- Construction happens in polytime in n
- ILP(A,b) has a solution x
  $\Rightarrow$ if x = circuit input, then output = 1. $\Rightarrow$ C is satisfiable
- C is satisfiable for inputs z
  $\Rightarrow Az \leq b \Rightarrow$ ILP instance has a solution.

CNFSAT

CNFSAT

Input: Circuit in conjunctive normal form:

CNFSAT

Input: Circuit in conjunctive normal form:

- Circuit has inputs $z_1, \ldots, z_n$.

CNFSAT

Input: Circuit in conjunctive normal form:

- Circuit has inputs $z_1, \ldots, z_n$.
- There are m OR gates, each with k inputs. Circuit inputs or their complements are connected to inputs of OR gates.

CNFSAT

Input: Circuit in conjunctive normal form:

- Circuit has inputs $z_1, \ldots, z_n$.
- There are m OR gates, each with k inputs. Circuit inputs or their complements are connected to inputs of OR gates.
- Outputs of OR gates are connected to an AND gate.

CNFSAT

Input: Circuit in conjunctive normal form:

- Circuit has inputs $z_1, \ldots, z_n$.
- There are m OR gates, each with k inputs. Circuit inputs or their complements are connected to inputs of OR gates.
- Outputs of OR gates are connected to an AND gate.
- Output of AND gate is output of overall circuit.

CNFSAT

Input: Circuit in conjunctive normal form:

- Circuit has inputs $z_1, \ldots, z_n$.
- There are m OR gates, each with k inputs. Circuit inputs or their complements are connected to inputs of OR gates.
- Outputs of OR gates are connected to an AND gate.
- Output of AND gate is output of overall circuit.

Output: Is it possible to assign values to inputs s.t. circuit output is 1?

CNFSAT

Input: Circuit in conjunctive normal form:

- Circuit has inputs $z_1, \ldots, z_n$.
- There are m OR gates, each with k inputs. Circuit inputs or their complements are connected to inputs of OR gates.
- Outputs of OR gates are connected to an AND gate.
- Output of AND gate is output of overall circuit.

Output: Is it possible to assign values to inputs s.t. circuit output is 1?

Obvious result: CNFSAT $\leq_K$ CSAT

CNFSAT

Input: Circuit in conjunctive normal form:

- Circuit has inputs $z_1, \ldots, z_n$.
- There are m OR gates, each with k inputs. Circuit inputs or their complements are connected to inputs of OR gates.
- Outputs of OR gates are connected to an AND gate.
- Output of AND gate is output of overall circuit.

Output: Is it possible to assign values to inputs s.t. circuit output is 1?

Obvious result: CNFSAT $\leq_K$ CSAT
Instance map: Identity.

CNFSAT

Input: Circuit in conjunctive normal form:

- Circuit has inputs $z_1, \ldots, z_n$.
- There are m OR gates, each with k inputs. Circuit inputs or their complements are connected to inputs of OR gates.
- Outputs of OR gates are connected to an AND gate.
- Output of AND gate is output of overall circuit.

Output: Is it possible to assign values to inputs s.t. circuit output is 1?

Obvious result: CNFSAT $\leq_K$ CSAT
Instance map: Identity.

If R is a special case of Q, then $R \leq_K Q$ with instance map $=$ identity.

CNFSAT

Input: Circuit in conjunctive normal form:

- Circuit has inputs $z_1, \ldots, z_n$.
- There are m OR gates, each with k inputs. Circuit inputs or their complements are connected to inputs of OR gates.
- Outputs of OR gates are connected to an AND gate.
- Output of AND gate is output of overall circuit.

Output: Is it possible to assign values to inputs s.t. circuit output is 1?

Obvious result: CNFSAT $\leq_K$ CSAT
Instance map: Identity.

If R is a special case of Q, then $R \leq_K Q$ with instance map $=$ identity.

Note: CNFSAT input can also be given as a formula in propositional logic: conjunction of "clauses", where clause $=$ disjunctions of variables or their negations.

CSAT $\leq_K$ CNFSAT

CSAT $\leq_K$ CNFSAT

Theorem: Every combinational circuit can be expressed as a CNF circuit.

CSAT $\leq_K$ CNFSAT

Theorem: Every combinational circuit can be expressed as a CNF circuit.

False start:

Instance map IM(C){ Return CNF circuit equivalent to C}

CSAT $\leq_K$ CNFSAT

Theorem: Every combinational circuit can be expressed as a CNF circuit.

False start:
Instance map IM(C){ Return CNF circuit equivalent to C}

This does not work, because resulting CNF circuit can be exponentially large! Instance map will not run in polytime.

CSAT $\leq_K$ CNFSAT

Theorem: Every combinational circuit can be expressed as a CNF circuit.

False start:
Instance map IM(C){ Return CNF circuit equivalent to C}

This does not work, because resulting CNF circuit can be exponentially large! Instance map will not run in polytime.

Key idea: Instance map should return circuit C' that checks if C "works correctly" and is satisfiable.

CSAT $\leq_K$ CNFSAT

Theorem: Every combinational circuit can be expressed as a CNF circuit.

False start:
Instance map IM(C){ Return CNF circuit equivalent to C}

This does not work, because resulting CNF circuit can be exponentially large! Instance map will not run in polytime.

Key idea: Instance map should return circuit C' that checks if C "works correctly" and is satisfiable.

- C is satisfiable

CSAT $\leq_K$ CNFSAT

Theorem: Every combinational circuit can be expressed as a CNF circuit.

False start:
Instance map IM(C){ Return CNF circuit equivalent to C}

This does not work, because resulting CNF circuit can be exponentially large! Instance map will not run in polytime.

Key idea: Instance map should return circuit C' that checks if C "works correctly" and is satisfiable.

- C is satisfiable $\Rightarrow$ C' will return 1

CSAT $\leq_K$ CNFSAT

Theorem: Every combinational circuit can be expressed as a CNF circuit.

False start:
Instance map IM(C){ Return CNF circuit equivalent to C}

This does not work, because resulting CNF circuit can be exponentially large! Instance map will not run in polytime.

Key idea: Instance map should return circuit C' that checks if C "works correctly" and is satisfiable.

- C is satisfiable $\Rightarrow$ C' will return 1 $\Rightarrow$ C' is satisfiable

CSAT $\leq_K$ CNFSAT

Theorem: Every combinational circuit can be expressed as a CNF circuit.

False start:
Instance map IM(C){ Return CNF circuit equivalent to C}

This does not work, because resulting CNF circuit can be exponentially large! Instance map will not run in polytime.

Key idea: Instance map should return circuit C' that checks if C "works correctly" and is satisfiable.

- C is satisfiable $\Rightarrow$ C' will return 1 $\Rightarrow$ C' is satisfiable
- C' is satisfiable

CSAT $\leq_K$ CNFSAT

Theorem: Every combinational circuit can be expressed as a CNF circuit.

False start:
Instance map IM(C){ Return CNF circuit equivalent to C}

This does not work, because resulting CNF circuit can be exponentially large! Instance map will not run in polytime.

Key idea: Instance map should return circuit C' that checks if C "works correctly" and is satisfiable.

- C is satisfiable $\Rightarrow$ C' will return 1 $\Rightarrow$ C' is satisfiable
- C' is satisfiable $\Rightarrow$ C' can return 1

CSAT $\leq_K$ CNFSAT

Theorem: Every combinational circuit can be expressed as a CNF circuit.

False start:
Instance map IM(C){ Return CNF circuit equivalent to C}

This does not work, because resulting CNF circuit can be exponentially large! Instance map will not run in polytime.

Key idea: Instance map should return circuit C' that checks if C "works correctly" and is satisfiable.

- C is satisfiable $\Rightarrow$ C' will return 1 $\Rightarrow$ C' is satisfiable
- C' is satisfiable $\Rightarrow$ C' can return 1 $\Rightarrow$ C can return 1

CSAT $\leq_K$ CNFSAT

Theorem: Every combinational circuit can be expressed as a CNF circuit.

False start:
Instance map IM(C){ Return CNF circuit equivalent to C}

This does not work, because resulting CNF circuit can be exponentially large! Instance map will not run in polytime.

Key idea: Instance map should return circuit C' that checks if C "works correctly" and is satisfiable.

- C is satisfiable $\Rightarrow$ C' will return 1 $\Rightarrow$ C' is satisfiable
- C' is satisfiable $\Rightarrow$ C' can return 1 $\Rightarrow$ C can return 1 $\Rightarrow$ C is satisfiable.

The reduction

The reduction

CSAT(C) : Does there exist vector $z \in \{0, 1\}^n$ which if given as input to C produces output 1?

The reduction

CSAT(C) : Does there exist vector $z \in \{0,1\}^n$ which if given as input to C produces output 1?
CNFSAT(CNF circuit C') : Does there exist 0-1 vector $z \in \{0,1\}^m$ which if given as input to C produces output 1?

The reduction

CSAT(C) : Does there exist vector $z \in \{0,1\}^n$ which if given as input to C produces output 1?
CNFSAT(CNF circuit C') : Does there exist 0-1 vector $z \in \{0,1\}^m$ which if given as input to C produces output 1?

IM(C){

The reduction

CSAT(C) : Does there exist vector $z \in \{0,1\}^n$ which if given as input to C produces output 1?
CNFSAT(CNF circuit C') : Does there exist 0-1 vector $z \in \{0,1\}^m$ which if given as input to C produces output 1?

IM(C){
$C^* =$ Circuit equivalent to C, having only 2 input gates.

The reduction

CSAT(C) : Does there exist vector $z \in \{0,1\}^n$ which if given as input to C produces output 1?
CNFSAT(CNF circuit C') : Does there exist 0-1 vector $z \in \{0,1\}^m$ which if given as input to C produces output 1?

IM(C){
$C^* =$ Circuit equivalent to C, having only 2 input gates.
Inputs to C': Inputs to C^*, internal nodes of C^*

The reduction

CSAT(C) : Does there exist vector $z \in \{0,1\}^n$ which if given as input to C produces output 1?
CNFSAT(CNF circuit C') : Does there exist 0-1 vector $z \in \{0,1\}^m$ which if given as input to C produces output 1?

IM(C){
$C^* =$ Circuit equivalent to C, having only 2 input gates.
Inputs to C': Inputs to C^*, internal nodes of C^*
Clauses: Union of clause sets L_i to assert that Gate i in C^* "works correctly" $\cup \{z\}$ $z =$ output of C^*

The reduction

CSAT(C) : Does there exist vector $z \in \{0,1\}^n$ which if given as input to C produces output 1?
CNFSAT(CNF circuit C') : Does there exist 0-1 vector $z \in \{0,1\}^m$ which if given as input to C produces output 1?

IM(C){
$C^* =$ Circuit equivalent to C, having only 2 input gates.
Inputs to C': Inputs to C^*, internal nodes of C^*
Clauses: Union of clause sets L_i to assert that Gate i in C^* "works correctly" $\cup \{z\}$ $z =$ output of C^*
L_i for AND gate with inputs u, v output w :
$w = uv$

The reduction

CSAT(C) : Does there exist vector $z \in \{0,1\}^n$ which if given as input to C produces output 1?
CNFSAT(CNF circuit C') : Does there exist 0-1 vector $z \in \{0,1\}^m$ which if given as input to C produces output 1?

IM(C){
$C^* =$ Circuit equivalent to C, having only 2 input gates.
Inputs to C': Inputs to C^*, internal nodes of C^*
Clauses: Union of clause sets L_i to assert that Gate i in C^* "works correctly" $\cup \{z\}$ $z =$ output of C^*
L_i for AND gate with inputs u, v output w :
$w = uv \equiv (w \rightarrow uv)(uv \rightarrow w)$

The reduction

CSAT(C) : Does there exist vector $z \in \{0,1\}^n$ which if given as input to C produces output 1?
CNFSAT(CNF circuit C') : Does there exist 0-1 vector $z \in \{0,1\}^m$ which if given as input to C produces output 1?

IM(C){
$C^* =$ Circuit equivalent to C, having only 2 input gates.
Inputs to C': Inputs to C^*, internal nodes of C^*
Clauses: Union of clause sets L_i to assert that Gate i in C^* "works correctly" $\cup \{z\}$ $\quad z =$ output of C^*
L_i for AND gate with inputs u, v output w :
$w = uv \equiv (w \rightarrow uv)(uv \rightarrow w) \equiv (uv + w')((uv)' + w)$

The reduction

CSAT(C) : Does there exist vector $z \in \{0,1\}^n$ which if given as input to C produces output 1?
CNFSAT(CNF circuit C') : Does there exist 0-1 vector $z \in \{0,1\}^m$ which if given as input to C produces output 1?

IM(C){
$C^* =$ Circuit equivalent to C, having only 2 input gates.
Inputs to C': Inputs to C^*, internal nodes of C^*
Clauses: Union of clause sets L_i to assert that Gate i in C^* "works correctly" $\cup \{z\}$ $\quad$ $z =$ output of C^*
L_i for AND gate with inputs u, v output w :
$w = uv \equiv (w \rightarrow uv)(uv \rightarrow w) \equiv (uv + w')((uv)' + w)$
$\equiv (u + w')(v + w')(u' + v' + w)$ $\quad$ Clause set L_i

The reduction

CSAT(C) : Does there exist vector $z \in \{0,1\}^n$ which if given as input to C produces output 1?
CNFSAT(CNF circuit C') : Does there exist 0-1 vector $z \in \{0,1\}^m$ which if given as input to C produces output 1?

IM(C){
$C^* =$ Circuit equivalent to C, having only 2 input gates.
Inputs to C': Inputs to C^*, internal nodes of C^*
Clauses: Union of clause sets L_i to assert that Gate i in C^* "works correctly" $\cup \{z\}$ $\quad$ $z =$ output of C^*
L_i for AND gate with inputs u, v output w :
$w = uv \equiv (w \to uv)(uv \to w) \equiv (uv + w')((uv)' + w)$
$\equiv (u + w')(v + w')(u' + v' + w)$ $\quad$ Clause set L_i
L_i for OR, NOT gates : similar.
}

The reduction continued

The reduction continued

- IM runs in time polynomial in size of C.

The reduction continued

- *IM* runs in time polynomial in size of C.
- CSAT(C) $=$ true
  $\Rightarrow \exists$ input values for C that produce 1 at output.

The reduction continued

- IM runs in time polynomial in size of C.
- CSAT(C) = true
  $\Rightarrow \exists$ input values for C that produce 1 at output.
  Supply values at inputs of C + values at internal nodes to C',
  this should produce 1 at output of C'

The reduction continued

- IM runs in time polynomial in size of C.
- CSAT(C) = true
  $\Rightarrow \exists$ input values for C that produce 1 at output.
  Supply values at inputs of C + values at internal nodes to C',
  this should produce 1 at output of C'
  $\Rightarrow CNFSAT(C') =$ true.

The reduction continued

- IM runs in time polynomial in size of C.
- CSAT(C) = true
  $\Rightarrow \exists$ input values for C that produce 1 at output.
  Supply values at inputs of C + values at internal nodes to C',
  this should produce 1 at output of C'
  $\Rightarrow CNFSAT(C') =$ true.
- $CNFSAT(C') =$ true
  $\Rightarrow \exists$ input values for C' that produce 1 at output.

The reduction continued

- IM runs in time polynomial in size of C.
- CSAT(C) = true
  $\Rightarrow \exists$ input values for C that produce 1 at output.
  Supply values at inputs of C + values at internal nodes to C', this should produce 1 at output of C'
  $\Rightarrow CNFSAT(C') =$ true.
- $CNFSAT(C') =$ true
  $\Rightarrow \exists$ input values for C' that produce 1 at output.
  Inputs to C' include inputs to C. Pick out those and supply to C. Should produce 1 at output of C.

The reduction continued

- IM runs in time polynomial in size of C.
- CSAT(C) = true
  $\Rightarrow \exists$ input values for C that produce 1 at output.
  Supply values at inputs of C + values at internal nodes to C',
  this should produce 1 at output of C'
  $\Rightarrow CNFSAT(C') =$ true.
- $CNFSAT(C') =$ true
  $\Rightarrow \exists$ input values for C' that produce 1 at output.
  Inputs to C' include inputs to C. Pick out those and supply
  to C. Should produce 1 at output of C.
  $\Rightarrow$ CSAT(C) = true.

The reduction continued

- IM runs in time polynomial in size of C.
- CSAT(C) = true
  $\Rightarrow \exists$ input values for C that produce 1 at output.
  Supply values at inputs of C + values at internal nodes to C', this should produce 1 at output of C'
  $\Rightarrow CNFSAT(C') =$ true.
- $CNFSAT(C') =$ true
  $\Rightarrow \exists$ input values for C' that produce 1 at output.
  Inputs to C' include inputs to C. Pick out those and supply to C. Should produce 1 at output of C.
  $\Rightarrow$ CSAT(C) = true.

Remark: Each clause in C' in the above reduction has at most 3 variables/negations. Can get exactly 3 with slight modification.

The reduction continued

- IM runs in time polynomial in size of C.
- CSAT(C) = true
  $\Rightarrow \exists$ input values for C that produce 1 at output.
  Supply values at inputs of C + values at internal nodes to C', this should produce 1 at output of C'
  $\Rightarrow CNFSAT(C') =$ true.
- $CNFSAT(C') =$ true
  $\Rightarrow \exists$ input values for C' that produce 1 at output.
  Inputs to C' include inputs to C. Pick out those and supply to C. Should produce 1 at output of C.
  $\Rightarrow$ CSAT(C) = true.

Remark: Each clause in C' in the above reduction has at most 3 variables/negations. Can get exactly 3 with slight modification.

3CNFSAT or 3SAT: Satisfiablilty when each clause has exactly 3 variables or their negations.

The reduction continued

- IM runs in time polynomial in size of C.
- CSAT(C) = true
  $\Rightarrow \exists$ input values for C that produce 1 at output.
  Supply values at inputs of C + values at internal nodes to C', this should produce 1 at output of C'
  $\Rightarrow CNFSAT(C') =$ true.
- $CNFSAT(C') =$ true
  $\Rightarrow \exists$ input values for C' that produce 1 at output.
  Inputs to C' include inputs to C. Pick out those and supply to C. Should produce 1 at output of C.
  $\Rightarrow$ CSAT(C) = true.

Remark: Each clause in C' in the above reduction has at most 3 variables/negations. Can get exactly 3 with slight modification.

3CNFSAT or 3SAT: Satisfiablilty when each clause has exactly 3 variables or their negations.

CSAT $\leq_K$ 3SAT

3SAT $\leq_K$ IS

3SAT(3CNF formula C) : Is C satisfiable

C = conjunction of clauses

IS(G,k) : Does G have an IS of size k?

3SAT $\leq_K$ IS

3SAT(3CNF formula C) : Is C satisfiable

C = conjunction of clauses

IS(G,k) : Does G have an IS of size k?

The questions in the two problems:

3SAT $\leq_K$ IS

3SAT(3CNF formula C) : Is C satisfiable

C = conjunction of clauses

IS(G,k) : Does G have an IS of size k?

The questions in the two problems:

3SAT: Should a literal be set to true?

3SAT $\leq_K$ IS

3SAT(3CNF formula C) : Is C satisfiable

C = conjunction of clauses

IS(G,k) : Does G have an IS of size k?

The questions in the two problems:
3SAT: Should a literal be set to true?
IS: Should a vertex be put in the IS?

3SAT $\leq_K$ IS

3SAT(3CNF formula C) : Is C satisfiable

C = conjunction of clauses

IS(G,k) : Does G have an IS of size k?

The questions in the two problems:

3SAT: Should a literal be set to true?

IS: Should a vertex be put in the IS?

Constraints:

3SAT $\leq_K$ IS

3SAT(3CNF formula C) : Is C satisfiable

C = conjunction of clauses

IS(G,k) : Does G have an IS of size k?

The questions in the two problems:

3SAT: Should a literal be set to true?

IS: Should a vertex be put in the IS?

Constraints:

3SAT: A literal and its complement should both not be true.

3SAT $\leq_K$ IS

3SAT(3CNF formula C) : Is C satisfiable

C = conjunction of clauses

IS(G,k) : Does G have an IS of size k?

The questions in the two problems:

3SAT: Should a literal be set to true?

IS: Should a vertex be put in the IS?

Constraints:

3SAT: A literal and its complement should both not be true.

3SAT: At least one literal in a clause should be set true.

3SAT $\leq_K$ IS

3SAT(3CNF formula C) : Is C satisfiable

C = conjunction of clauses

IS(G,k) : Does G have an IS of size k?

The questions in the two problems:

3SAT: Should a literal be set to true?

IS: Should a vertex be put in the IS?

Constraints:

3SAT: A literal and its complement should both not be true.

3SAT: At least one literal in a clause should be set true.

IS: Vertices connected by an edge should both not be selected.

3SAT $\leq_K$ IS

3SAT(3CNF formula C) : Is C satisfiable

C $=$ conjunction of clauses

IS(G,k) : Does G have an IS of size k?

The questions in the two problems:

3SAT: Should a literal be set to true?

IS: Should a vertex be put in the IS?

Constraints:

3SAT: A literal and its complement should both not be true.

3SAT: At least one literal in a clause should be set true.

IS: Vertices connected by an edge should both not be selected.

At least k vertices must be selected.

3SAT $\leq_K$ IS

3SAT(3CNF formula C) : Is C satisfiable

C = conjunction of clauses

IS(G,k) : Does G have an IS of size k?

The questions in the two problems:

3SAT: Should a literal be set to true?

IS: Should a vertex be put in the IS?

Constraints:

3SAT: A literal and its complement should both not be true.

3SAT: At least one literal in a clause should be set true.

IS: Vertices connected by an edge should both not be selected.

At least k vertices must be selected.

Ideas for reduction:

3SAT $\leq_K$ IS

3SAT(3CNF formula C) : Is C satisfiable

C = conjunction of clauses

IS(G,k) : Does G have an IS of size k?

The questions in the two problems:
3SAT: Should a literal be set to true?
IS: Should a vertex be put in the IS?

Constraints:
3SAT: A literal and its complement should both not be true.
3SAT: At least one literal in a clause should be set true.
IS: Vertices connected by an edge should both not be selected.
At least k vertices must be selected.

Ideas for reduction:
Each literal should be represented by a vertex?

3SAT $\leq_K$ IS

3SAT(3CNF formula C) : Is C satisfiable

C = conjunction of clauses

IS(G,k) : Does G have an IS of size k?

The questions in the two problems:
3SAT: Should a literal be set to true?
IS: Should a vertex be put in the IS?

Constraints:
3SAT: A literal and its complement should both not be true.
3SAT: At least one literal in a clause should be set true.
IS: Vertices connected by an edge should both not be selected.
At least k vertices must be selected.

Ideas for reduction:
Each literal should be represented by a vertex?
The vertices corresponding to a literal and its complement should be connected by an edge?

The reduction

IM(CNF formula C){

The reduction

IM(CNF formula C){
Let L_i be the clauses.

The reduction

IM(CNF formula C){
Let L_i be the clauses.
Create vertex for each literal in the formula.

The reduction

IM(CNF formula C){

Let L_i be the clauses.

Create vertex for each literal in the formula.

Connect all vertices corresponding to a clause by edges.

The reduction

IM(CNF formula C){
Let L_i be the clauses.
Create vertex for each literal in the formula.
Connect all vertices corresponding to a clause by edges.

If u appears in one clause, and u' in another, connect corresponding vertices by a edge.

The reduction

IM(CNF formula C){
Let L_i be the clauses.
Create vertex for each literal in the formula.
Connect all vertices corresponding to a clause by edges.

If u appears in one clause, and u' in another, connect corresponding vertices by a edge.

$k =$ number of clauses.
Return Graph constructed, k }

The reduction

IM(CNF formula C){
Let L_i be the clauses.
Create vertex for each literal in the formula.
Connect all vertices corresponding to a clause by edges.

If u appears in one clause, and u' in another, connect corresponding vertices by a edge.

$k =$ number of clauses.
Return Graph constructed, k }

- *IM* runs in time polynomial in formula size.

The reduction

IM(CNF formula C){
Let L_i be the clauses.
Create vertex for each literal in the formula.
Connect all vertices corresponding to a clause by edges.

If u appears in one clause, and u' in another, connect corresponding vertices by a edge.

$k =$ number of clauses.
Return Graph constructed, k }

- *IM* runs in time polynomial in formula size.
- Formula C is satisfiable $\Rightarrow$ every clause contains a true literal

The reduction

IM(CNF formula C){
Let L_i be the clauses.
Create vertex for each literal in the formula.
Connect all vertices corresponding to a clause by edges.

If u appears in one clause, and u' in another, connect corresponding vertices by a edge.

$k =$ number of clauses.
Return Graph constructed, k }

- *IM* runs in time polynomial in formula size.
- Formula C is satisfiable $\Rightarrow$ every clause contains a true literal $\Rightarrow$ Corresponding vertices form an IS of size k

The reduction

IM(CNF formula C){
Let L_i be the clauses.
Create vertex for each literal in the formula.
Connect all vertices corresponding to a clause by edges.

If u appears in one clause, and u' in another, connect corresponding vertices by a edge.

$k =$ number of clauses.
Return Graph constructed, k }

- *IM* runs in time polynomial in formula size.
- Formula C is satisfiable $\Rightarrow$ every clause contains a true literal $\Rightarrow$ Corresponding vertices form an IS of size k
- IS(G,k)=true $\Rightarrow$ IS of k vertices present.

The reduction

IM(CNF formula C){

Let L_i be the clauses.

Create vertex for each literal in the formula.

Connect all vertices corresponding to a clause by edges.

If u appears in one clause, and u' in another, connect corresponding vertices by a edge.

$k =$ number of clauses.

Return Graph constructed, k }

- *IM* runs in time polynomial in formula size.
- Formula C is satisfiable $\Rightarrow$ every clause contains a true literal $\Rightarrow$ Corresponding vertices form an IS of size k
- IS(G,k)=true $\Rightarrow$ IS of k vertices present.
  Each vertex must come from different clause.

The reduction

IM(CNF formula C){
Let L_i be the clauses.
Create vertex for each literal in the formula.
Connect all vertices corresponding to a clause by edges.

If u appears in one clause, and u' in another, connect corresponding vertices by a edge.

$k =$ number of clauses.
Return Graph constructed, k }

- *IM* runs in time polynomial in formula size.
- Formula C is satisfiable $\Rightarrow$ every clause contains a true literal $\Rightarrow$ Corresponding vertices form an IS of size k
- IS(G,k)=true $\Rightarrow$ IS of k vertices present.
  Each vertex must come from different clause.
  Vertex for $u \in IS \Rightarrow$ Vertex for $u' \notin IS$

The reduction

IM(CNF formula C){
Let L_i be the clauses.
Create vertex for each literal in the formula.
Connect all vertices corresponding to a clause by edges.

If u appears in one clause, and u' in another, connect corresponding vertices by a edge.

$k =$ number of clauses.
Return Graph constructed, k }

- *IM* runs in time polynomial in formula size.
- Formula C is satisfiable $\Rightarrow$ every clause contains a true literal $\Rightarrow$ Corresponding vertices form an IS of size k
- IS(G,k)=true $\Rightarrow$ IS of k vertices present.
  Each vertex must come from different clause.
  Vertex for $u \in IS \Rightarrow$ Vertex for $u' \notin IS$
  Set all literals associated with IS vertices true.

The reduction

IM(CNF formula C){
Let L_i be the clauses.
Create vertex for each literal in the formula.
Connect all vertices corresponding to a clause by edges.

If u appears in one clause, and u' in another, connect corresponding vertices by a edge.

k = number of clauses.
Return Graph constructed, k }

- *IM* runs in time polynomial in formula size.
- Formula C is satisfiable $\Rightarrow$ every clause contains a true literal $\Rightarrow$ Corresponding vertices form an IS of size k
- IS(G,k)=true $\Rightarrow$ IS of k vertices present.
  Each vertex must come from different clause.
  Vertex for $u \in IS \Rightarrow$ Vertex for $u' \notin IS$
  Set all literals associated with IS vertices true.
  If this does not set truth of both u, u' for some u, set u=true.

The reduction

IM(CNF formula C){
Let L_i be the clauses.
Create vertex for each literal in the formula.
Connect all vertices corresponding to a clause by edges.

If u appears in one clause, and u' in another, connect corresponding vertices by a edge.

$k =$ number of clauses.
Return Graph constructed, k }

- IM runs in time polynomial in formula size.
- Formula C is satisfiable $\Rightarrow$ every clause contains a true literal $\Rightarrow$ Corresponding vertices form an IS of size k
- IS(G,k)=true $\Rightarrow$ IS of k vertices present.
  Each vertex must come from different clause.
  Vertex for $u \in IS \Rightarrow$ Vertex for $u' \notin IS$
  Set all literals associated with IS vertices true.
  If this does not set truth of both u, u' for some u, set u=true.
  $\Rightarrow$ C is true.