Complexity - theoretic consequences of the existence of optimal proof systems

Zenon Sadowski
University of Bialystok, Poland

http://math.uwb.edu.pl/~sadowski/

The theory of propositional proof systems		Structural complexity theory
Optimal proof systems	$\longrightarrow$	Complete sets for NP$\cap$co-NP and for UP

Key observation:

There is a connection between the existence of optimal proof systems for Propositional Logic and the existence of complete languages for NP$\cap$co-NP and for UP.

Four consequences in Structural Complexity Theory

(I)

An optimal proof system for TAUT exists if and only if there is a recursive enumeration of machines covering all easy (recognizable in polynomial time) subsets of TAUT.

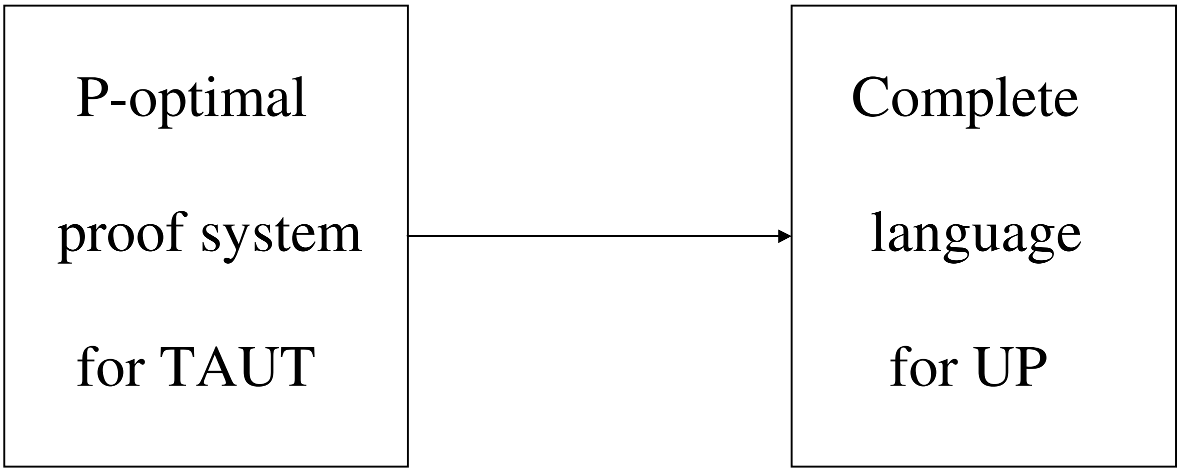
TAUT -- the set of all propositional tautologies

(II)

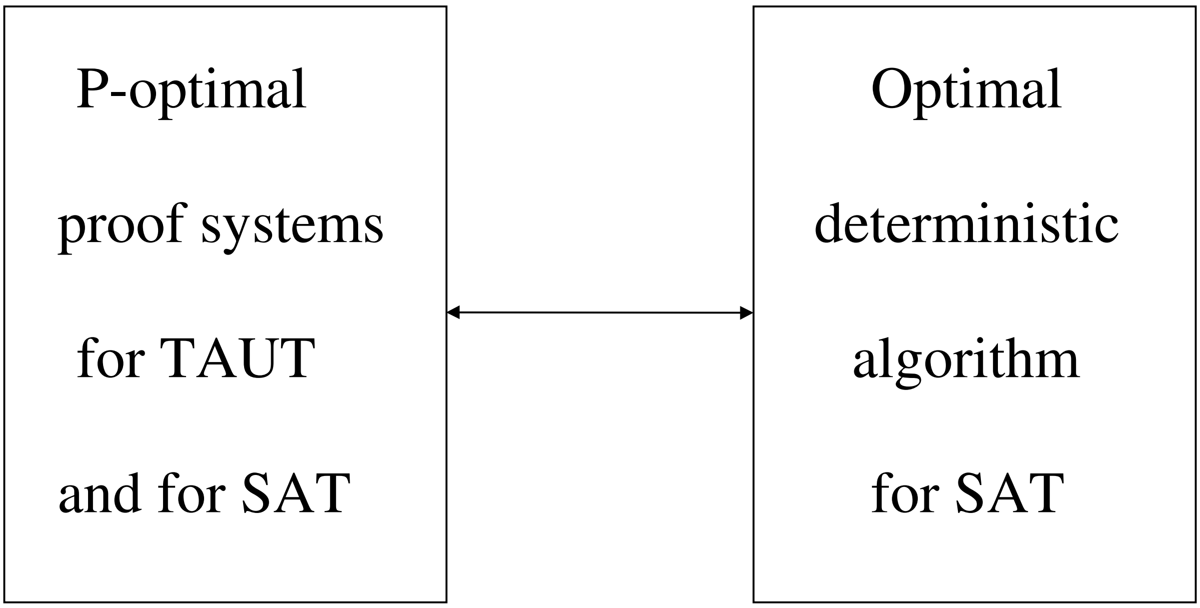
P-optimal proof systems for TAUT and for SAT	$\longrightarrow$	Complete language for NP$\cap$co-NP

SAT -- the set of all satisfiable boolean formulas

(III)


P-optimal
proof system
for TAUT
Complete
language
for UP

(IV)


P-optimal
proof systems
for TAUT
and for SAT
Optimal
deterministic
algorithm
for SAT

Relevant works:

- Cook, Reckhow 1979
  – Definition of a propositional proof system (a proof system for TAUT)

- Kowalczyk 1984
  – NP$\cap$co-NP has a complete language if and only if there exists a recursive enumeration of machines covering all languages from NP$\cap$co-NP.

- Hartmanis, Hemachandra 1988
  – UP has a complete language if and only if there exists a recursive enumeration of machines covering all languages from UP.

- Krajíček, Pudlák 1989
  – Whether an optimal proof system for TAUT exists ?
  – Whether a p-optimal proof system for TAUT exists ?

- Messner, Torán 1997
  – If there exists a p-optimal proof system for TAUT then there exists a complete language for UP.

- Köbler, Messner 1998
  – Whether a p-optimal proof system for SAT exists ?
  – If both p-optimal proof systems for TAUT and for SAT exist then NP$\cap$co-NP has a complete language.

Outline of the presentation

1. Optimal proof systems
2. Complete sets for NP$\cap$co-NP and for UP
3. The structure of easy subsets of TAUT
4. A connection between optimal proof systems and complete sets
5. An optimal deterministic algorithm for SAT

Preliminaries

Σ – a certain fixed finite alphabet
$x \in \Sigma^*$ - x is a string over the alphabet Σ
$|x|$ - the length of x
TIME(M, w) - computing time of a Turing machine M on w
PTM - polynomial time clocked deterministic Turing machine ($n^k + k$ its polynomial time bound)

1. Optimal proof systems

Definition (Cook, Reckhow)
A propositional proof system (a proof system for TAUT)...

$$f:\quad \underset{\text{(proofs)}}{\Sigma^*} \xrightarrow{\text{onto}} \underset{\text{(formulas)}}{\mathrm{TAUT}}$$

f - computable by a deterministic Turing machine in time bounded by a polynomial in the length of the input

A string w such that $f(w) = \alpha$ we call a proof of a formula α.

Definition (Cook, Reckhow)
A propositional proof system is polynomially bounded if and only if there exists a polynomial p such that every $\alpha \in \mathrm{TAUT}$ has a proof of length no more than $p(|\alpha|)$ in this system.

Fact (Cook, Reckhow)
NP = co-**NP** if and only if there exists a polynomially bounded propositional proof system.

L -- a language, e.g., L = SAT, L = TAUT

Definition (Köbler, Messner)
A proof system for a language L is a polynomial time computable function $h: \Sigma^* \xrightarrow{\text{onto}} L$.

How does one compare the efficiency of proof systems ?

h, h' - proof systems for a language L
proof system h p - simulates proof system h'

Definition
We say that h <u>p - simulates</u> h' if there exists a polynomial time computable function $\gamma : \Sigma^* \to \Sigma^*$ such that for every $x \in L$ and every $w \in \Sigma^*$, if w is a proof of x in h', then $\gamma(w)$ is a proof of x in h.

$$\begin{tikzcd}
w \arrow[r, "h'"] \arrow[d, "\gamma"'] & x \\
\gamma(w) \arrow[ur, "h"'] &
\end{tikzcd}$$

w - a proof of x in h'
$\gamma(w)$ - a proof of x in h

g, g' - propositional proof systems (proof systems for TAUT)

Definition
We say that g <u>simulates</u> g' if there exists a polynomial p such that for every tautology α, if α has a proof of length n in g', then α has a proof of length $\leq p(n)$ in g.

$$\begin{tikzcd}
g \text{ p-simulates } g' \arrow[d] \\
g \text{ simulates } g'
\end{tikzcd}$$

	Character of the translation
p-simulation	deterministic
Simulation	nondeterministic

L - a language

Definition
A proof system is p - optimal for L if it p - simulates any proof system for L.

Definition
A proof system is optimal for L if it simulates any proof system for L.

Open problems (Krajíček , Pudlák)

Whether there exists a p - optimal proof system for TAUT ?

Whether there exists an optimal proof system for TAUT ?

Open problem (Köbler, Messner)

Whether there exists a p-optimal proof system for SAT ?

2. Complete languages for NP∩co-NP and for UP

NP∩co-NP - class of languages we are interested in

Languages in **NP** have succinct certificates.
Languages in **co-NP** have succinct disqualifications.
Languages in **NP∩co-NP** have both.

Definition

A language L is said to be complete for **NP∩co-NP** if L∈ **NP∩co-NP** and any language M in **NP∩co-NP** is many-one polynomial time reducible to L.

Problem

Whether **NP∩co-NP** has complete languages?

This problem relativizes both ways.

Theorem (Immerman, Hartmanis)

There is an oracle E such that $\mathbf{NP}^{\mathbf{E}}\cap\mathbf{co\text{-}NP}^{\mathbf{E}}$ has complete languages.

Theorem (Sipser)

There is an oracle X for which $\mathbf{NP}^{\mathbf{X}}\cap\mathbf{co\text{-}NP}^{\mathbf{X}}$ does not have a complete language.

Definition

A Turing machine M is called a YES-NO nondeterministic Turing machine if M is nondeterministic and possesses two distinguished states, the accepting state YES, and the rejecting state NO.

Definition

A YES-NO nondeterministic Turing machine is called strong nondeterministic if conditions (i) - (ii) hold.

(i) For any input x there exists a computation path of M accepting x or there exists a computation path rejecting x.

(ii) For any input x there does not exist simultaneously a computation path of M accepting x and a computation path of M rejecting x.

$N_1, N_2, N_3, N_4, \ldots$ standard enumeration of all nondeterministic polynomial time clocked Turing machines

NP$\cap$**co-NP** = { $L(N_i)$: N_i is strong nondeterministic }

Theorem (Kowalczyk)
There exists a complete language for **NP$\cap$co-NP** if and only if there exists a recursively enumerable list of strong nondeterministic polynomial time clocked Turing machines N_{i_1} , N_{i_2} , N_{i_3} , ... such that $\{L(N_{i_k}): \ k \geq 1 \}$ = **NP$\cap$co-NP**

r. e. list of machines naming all languages from **NP$\cap$co-NP**
„ nice" presentation of **NP$\cap$co-NP**

T - formal theory

(1) the set of all theorems of T is recursively enumerable
(2) only true sentences about Turing machines can be proved in T (T is sound)

Theorem (Kowalczyk)
If **NP$\cap$co-NP** has no complete languages, then for any theory T there exists $L \in$ **NP$\cap$co-NP** such that for no nondeterministic polynomial time clocked N_i with $L(N_i) = L$ can it be proven in T, that N_i is strong nondeterministic.

UP - languages

Definition

A nondeterministic Turing machine is categorical if for any input x there is at most one accepting computation .

categorical = unambiguous

$$\mathbf{UP} = \{ L(N_i) : N_i \text{ is categorical } \}$$

Problem

Whether **UP** has complete languages ?

This problem relativizes both ways.

Theorem (Hartmanis, Hemachandra)

There exists an oracle A such that $\mathbf{UP}^{\mathbf{A}}$ has no complete languages.

Theorem (Hartmanis, Hemachandra)

There exists an oracle B such that $\mathbf{UP}^{\mathbf{B}}$ has complete languages.

Theorem (Hartmanis, Hemachandra)

There exists a complete language for **UP** if and only if there exists a recursively enumerable list of categorical nondeterministic polynomial time clocked Turing machines N_{i_1} , N_{i_2} , N_{i_3} , ... such that $\{L(N_{i_k}): \ k \geq 1\}$ = **UP**

„nice" presentation of **UP**

Theorem (Hartmanis, Hemachandra)

If **UP** has no complete languages , then for any theory T there exists $L \in$ **UP** such that for no nondeterministic polynomial time clocked N_i with $L(N_i) = L$ can it be proven in T, that N_i is categorical.

3. The structure of easy subsets of TAUT

Definition

A transducer N is called sound if N produces only propositional tautologies.

USEFUL TOOL :
Formulas expressing the soundness of Turing machines

polynomial time clocked transducer

M

$$A_M = \{\ Sound_M^1,\ Sound_M^2,\ Sound_M^3,\ \ldots\ \}$$

the set of propositional formulas

(1) $Sound_M^n$ is a propositional tautology if and only if for every input of length n, the machine M outputs a tautology.

(2) $Sound_M^n$ can be constructed in time polynomial in n.

Definition

By a NP-easy subset of TAUT we mean a set A such that $A \subset$ TAUT and $A \in$ **NP**.

Definition

An optimal nondeterministic algorithm for TAUT is a nondeterministic Turing machine M which accepts TAUT and such that for every nondeterministic Turing machine M' which accepts TAUT there exists a polynomial p such that for every $\alpha \in$ TAUT

$$\boxed{\text{TIME}(M, \alpha) \leq p(|\alpha|, \text{TIME}(M', \alpha)}$$

Theorem

Statements (i) -- (iii) are equivalent.

(i) There exists an optimal propositional proof system.

(ii) There exists an optimal nondeterministic algorithm for TAUT.

(iii) The class of all NP-easy subsets of TAUT possesses a recusive NP-presentation.

NP - presentation

There exists a r.e. list of nondeterministic polynomial time clocked Turing machines N_{i_1}, N_{i_2}, N_{i_3}, ... such that

(1) $L(N_{i_j}) \subset$ TAUT for every j

(2) For every NP-easy A there exists k such that $A = L(N_{i_k})$.

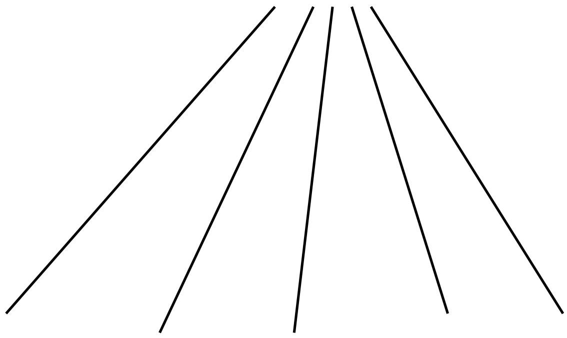
Proof

(i) → (ii)

propositional proof system	→	nondeterministic algorithm for TAUT - „guess and verify”
nondeterministic algorithm for TAUT	→	propositional proof system

(ii) → (ii)

M - an optimal nondeterministic algorithm for TAUT



F_1 F_2 F_3 F_4 ... Turing machines

F_i simulates M and accepts w if and only if M accepts w in no more than $n^i + i$ steps, where $n = |w|$

NP - cover

(1) For every i, $L(F_i) \subset$ TAUT
(2) For every A, NP-easy subset of TAUT, there exists i such that $A \subset L(F_i)$.

↓

NP - presentation

$N_1, N_2, N_3, N_4, \ldots$ the list of all nondeterministic polynomial time clocked Turing machines

Let $j = \langle i, k \rangle$, w - input, $|w| = n$

K_j simulates $n^k + k$ steps of F_i and N_k and accepts w if and only if both F_i and N_k accepts w.

$\boxed{K_1,\ K_2,\ K_3,\ K_4, \ldots \quad \text{NP - presentation}}$

(iii) $\rightarrow$ (i)

We define Opt: $\Sigma^* \xrightarrow{\text{onto}} \text{TAUT}$

What does a proof of α in Opt look like ?

w — PTM M (Turing machine) — α

$|w| = n$

M - sound $\longleftrightarrow$ α - propositional tautology

M n-sound $\longleftrightarrow$ α - propositional tautology

Turing machine M is n-sound if and only if for any input of length n, M produces a propositional tautology.

< M, w, certificate >

A proof of α in Opt

The certificate states that M is n-sound, where $n = |w|$.

We use NP-presentation and Sound_M^n formulas to produce certificates

Let G be the Turing machine generating the codes of the machines N_{i_1}, N_{i_2}, N_{i_3}, ... forming NP-presentation

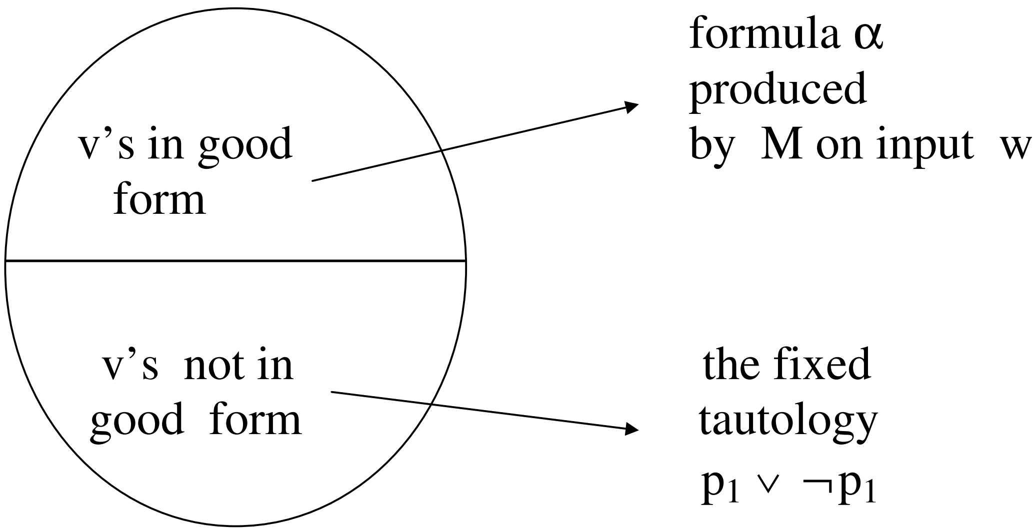
A string v is in good form if

$$v = < M,\ w,\ \text{Comp-G},\ \text{Comp-Sound}_M{}^n,\ 0^{n^k+k} >$$

where:

M - PTM with time bound $n^k + k$
n - the length of w
Comp-G - a computation of the machine G. This computation produces a code of a certain machine N_{i_j} .
$\text{Comp-Sound}_M{}^n$ - a computation of the machine N_{i_j} accepting the formula $\text{Sound}_M{}^n$
0^{n^k+k} - the sequence of zeros (padding)

Opt:



1 $\quad \mathrm{Opt}: \Sigma^* \xrightarrow{\text{onto}} \mathrm{TAUT}$

G produces a code of N_{i_j}

$\downarrow$

N_{i_j} accepts Sound_M^n

$\downarrow$

Sound_M^n is a propositional tautology

$\downarrow$

M is n-sound

$\downarrow$

α is a propositional tautology

2 $\quad$ Opt is polynomial time computable

3 $\quad$ Opt simulates any propositional proof system

Let h be a propositional proof system

$$h: \Sigma^* \xrightarrow{\text{onto}} \mathrm{TAUT}$$

$$K \text{ - PTM of the system } h \text{ with time bound } n^l + l$$

$$\downarrow$$

$$A_K = \{ \text{Sound}_K^{\,1},\ \text{Sound}_K^{\,2},\ \text{Sound}_K^{\,3},\ \ldots \}$$

$$\downarrow$$

N_{i_j} - the machine from the NP-presentation naming the set A_K

$t: \Sigma^* \rightarrow \Sigma^*$

$$\boxed{t(x) = \langle K,\ x,\ \text{Comp-G},\ \text{Comp-Sound}_K^{\,n},\ 0^{n^l+1} \rangle}$$

n - the length of x
Comp-G - a computation of G producing N_{i_j}
$\text{Comp-Sound}_K^{\,n}$ - a computation of N_{i_j} accepting $\text{Sound}_K^{\,n}$

t translates proofs in h into proofs in Opt.

□

Definition

By an easy subset of TAUT we mean a set A such that $A \subset \text{TAUT}$ and $A \in \mathbf{P}$

Definition

An almost optimal deterministic algorithm for TAUT is a deterministic Turing machine M which recognizes TAUT and such that for every Turing machine M' which recognizes TAUT there exists a polynomial p such that

$$\boxed{\begin{array}{c}\text{for every } \alpha \in \text{TAUT} \\ \text{TIME}(M, \alpha) \leq p(|\alpha|, \text{TIME}(M', \alpha))\end{array}}$$

Theorem

Statements (i) - (iii) are equivalent.

(i) There exists a p-optimal propositional proof system.

(ii) There exists an almost optimal deterministic algorithm for TAUT

(iii) The class of all easy subsets of TAUT possesses a recursive P-presentation.

P-presentation

There exists a r.e. list of deterministic polynomial time clocked Turing machines $D_{i_1}, D_{i_2}, D_{i_3}, D_{i_4}, \ldots$ such that

(1) $L(D_{i_j}) \subset \text{TAUT}$ for every j

(2) For every easy A there exists k such that $A = L(D_{i_j})$

Definition

A deterministic (nondeterministic) Turing machine M is T-sound if $T \vdash$ „ $L(M) \subset TAUT$ ”

Definition

A set $A \subset TAUT$ is T-provably NP-easy if there exists a nondeterministic polynomial time clocked Turing machine M fulfilling (1) - (2)

(1) M is T-sound

(2) $L(M) = A$

Definition

... A is T-provably easy ...

Theorem

If there does not exist an optimal proof system, then for every theory T there exists an NP-easy subset of TAUT which is not T-provably NP-easy.

Proof

For every NP-easy A there exists an N_i such that $L(N_i) = A$ and N_i is T-sound.

$\downarrow$

$\Omega_T = \{ N_i : N_i$ is T-sound $\}$ is an NP-presentation of NP-easy subsets of TAUT.

$\downarrow$

There exists an optimal propositional proof system.

$\downarrow$

Contradiction

Theorem

If there does not exist a p-optimal propositional proof system, then for every theory T there exists an easy subset of TAUT which is not T-provably easy.

4. A connection between optimal proof systems and complete sets

NP $\cap$ co-NP, **UP** ———— promise classes

This classes are defined using nondeterministic polynomial time clocked Turing machines which obey special conditions (promises).

Class	Promise
NP $\cap$ co-NP	... N_i is strong nondeterministic ...
UP	... N_i is categorical ...

B. Borchert

predicate classes versus promise classes

Ch. Papadimitriou

syntactic classes versus semantic classes

The existence of optimal proof systems implies the existence of complete languages for certain promise classes.

Analogies

One language is polynomial time many-one reducible to the second language	A proof system p-simulates another proof system
complete language	optimal proof system

P-optimal proof system for TAUT $\longrightarrow$ Complete language for UP

Messner, Torán 1997

P-optimal proof systems for TAUT and for SAT $\longrightarrow$ Complete language for NP$\cap$co-NP

Köbler , Messner 1998

Theorem (Köbler, Messner)

If TAUT and SAT have p-optimal proof systems, then **NP**$\cap$ co-**NP** has a complete language.

Proof

L - **NP**$\cap$ co-**NP** complete language

$L = \{< M, x, \alpha_x^M, \beta_n^M, Proof_1, Proof_2, 0^{n^k+k} > : x \in L(M)\}$

where:

M - polynomial time clocked nondeterministic YES-NO Turing machine, $n^k + k$ its time bound

$|x| = n$

α_x^M - boolean formula

α_x^M is satisfiable if and only if M is x-complete

M is x-complete if and only if there exists a computation of M accepting x or there exists a computation of M rejecting x

β_n^M - boolean formula

β_n^M is a propositional tautology if and only if M is n-consitent

M is n-consistent if and only if for any input of length n there does not exist simultaneously a computation path of M accepting x and a computation path rejecting x.

Proof_1 - a proof of α_x^M in the p-optimal proof system for SAT

Proof_2 - a proof of β_n^M in the p-optimal proof system for TAUT

0^{n^k+k} - sequence of zeros (padding)

5. An optimal deterministic algorithm for SAT

L - a language, e. g., L = SAT, L = TAUT

Definition

An optimal deterministic algorithm for L is a deterministic Turing machine M which recognizes L and such that for every deterministic Turing machine M' which recognizes L there exists a polynomial p such that

$$\boxed{\begin{array}{l}\text{for every } x \in \Sigma^* \\ \text{TIME}(M, x) \leq p(|x|, \text{TIME}(M', x))\end{array}}$$

Optimal algorithm	Almost optimal algorithm
optimality property stated on both YES and NO instances	optimality property stated only on YES instances

P-optimal proof system for TAUT $\longleftrightarrow$ Almost optimal deterministic algorithm for TAUT

Krajíček, Pudlák 1989

What about an <u>optimal</u> deterministic algorithm for SAT (TAUT) ??

P- optimal proof system for SAT $\longleftrightarrow$ Almost optimal deterministic algorithm for SAT

Theorem

The following statements are equivalent.

(i) TAUT and SAT have p-optimal proof systems.

(ii) There exists an optimal deterministic algorithm for SAT.

(iii) There exists an optimal deterministic algorithm for TAUT.