A straightforward proof of Köbler-Messner's result

Zenon Sadowski
University of Bialystok, Poland

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

Theorem (Köbler, Messner)

Optimal proof systems for SAT and for TAUT $\longrightarrow$ Complete sets for $\mathbf{NP} \cap$ co-$\mathbf{NP}$

TAUT - set of all propositional tautologies
SAT - set of all satisfiable boolean formulas

Our Contribution

Another proof of this theorem

Relevant works:

- Cook, Reckhow 1979
  - Definition of a propositional proof system (a proof system for TAUT)
- Krajíček, Pudlák 1989
  - Does a p-optimal proof system for TAUT exist ?
- Sadowski 1997
  - If there exists a p-optimal proof system for the tautologies of the quantified propositional logic (QBF) then **NP** $\cap$ co-**NP** has a complete language.
- Köbler, Messner 1998
  - Does a p-optimal proof system for SAT exist ?
  - If both p-optimal proof systems for TAUT and for SAT exist then **NP** $\cap$ co-**NP** has a complete language.

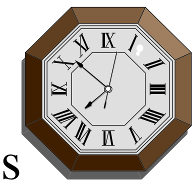
Outline of the presentation

1. Optimal proof systems
2. Complete sets for NP $\cap$ co-NP
3. Our proof of Köbler–Messner's result
4. Two bridges between optimal proof systems and Structural Complexity Theory

Preliminaries

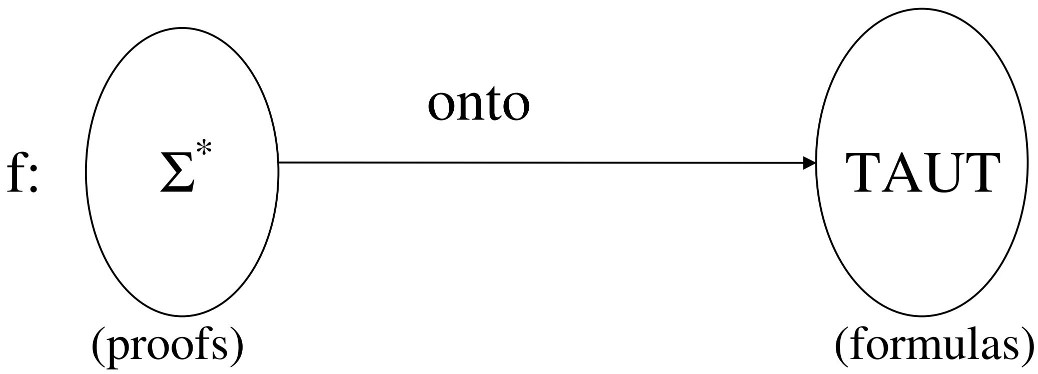
Σ – a certain fixed finite alphabet
$x \in \Sigma^*$ – x is a string over the alphabet Σ
$|x|$ – the length of x

basic computational model – deterministic and nondeterministic Turing machines

polynomial time clocked Turing machines 

1. Optimal proof systems

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

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, g – proof systems for a language L

Definition

We say that h p-simulates g if there exists a polynomial time computable function $\gamma: \Sigma^* \to \Sigma^*$ translating proofs in g into proofs in h.

$$\begin{tikzcd}
\text{(proofs)} & & \text{(formulas)} \\[-2em]
w \arrow[rr, "g"] \arrow[d, "\gamma"'] & & \alpha \\
\gamma(w) \arrow[urr, "h"] & &
\end{tikzcd}$$

w – a proof of α in g

γ(w) – a proof of α in h

L – a language

Definition (Krajíček, Pudlák)
A proof system is p-optimal for L if it p-simulates any proof system for L.

Open problem (Krajíček, Pudlák)
Does there exist a p-optimal proof system for TAUT ?

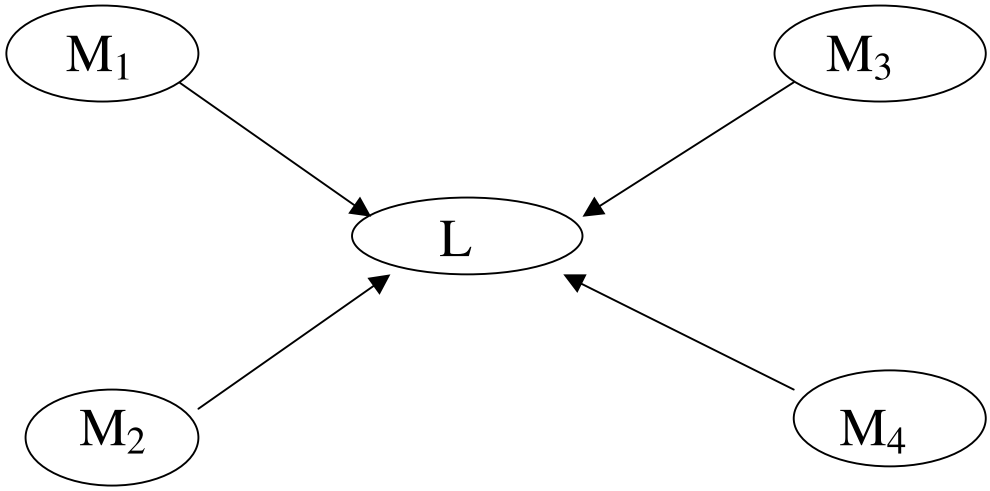
Open problem (Köbler, Messner)
Does there exist a p-optimal proof system for SAT ?

2. **Complete languages for NP $\cap$ co-NP**

NP$\cap$co-**NP** – class of languages
Languages in **NP** have succinct certificates
Languages in co-NP **have succinct disqualifications**
Languages in **NP**$\cap$co-**NP** have both

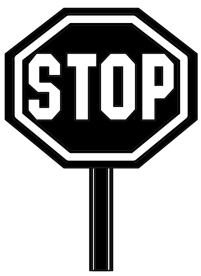
Definition

A language L is complete for $\mathbf{NP} \cap$ co-$\mathbf{NP}$ if ...



Open problem

Does $\mathbf{NP} \cap$ co-$\mathbf{NP}$ have complete languages ?



BAD NEWS

Theorem (Immerman, Hartmanis)

Exists oracle A: $\mathbf{NP}^A \cap$ co-$\mathbf{NP}^A$ has complete languages

Theorem (Sipser)

Exists oracle B: $\mathbf{NP}^B \cap$ co-$\mathbf{NP}^B$ does not have a complete language

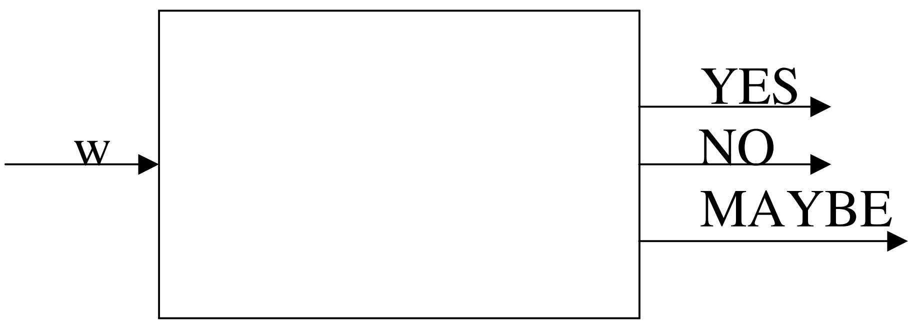
What does a machine model for NP$\cap$ **co-**NP **look like ?**

L$\in$ **NP**$\cap$ co-**NP** $\longleftrightarrow$ a pair of polynomial time clocked Turing machines which accept complementary languages

NP$\cap$ co-**NP** machine = polynomial time clocked Turing machine which obeys the special condition (promise)

What does this promise look like ?

Let M be a nondeterministic YES-NO Turing machine



(1) M behaves well on input x iff there exists a computation of M accepting x or there exists a computation of M rejecting x.

(2) M is n-consistent iff for any input x of length n there does not exist simultaneously a computation path of M accepting x and a computation path of M rejecting x.

Definition

A polynomial time clocked nondeterministic Turing machine M is strong nondeterministic if conditions (i)–(ii) hold:

(i) M behaves well on any input x
(ii) For any n the machine M is n-consistent

N_1, N_2, N_3, N_4, ... standard enumeration of all nondeterministic polynomial time clocked Turing machines

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

3. **Our proof of Köbler –Messner's result**

USEFFUL TOOLS:

Two families of boolean formulas expressing the promise: „M is strong nondeterministic"

M – polynomial time clocked Turing machine

$$\Big\downarrow$$

$$\{ \alpha_x^M : x \in \Sigma^* \}$$
$$\{ \beta_n^M : n = 1, 2, 3, \ldots \}$$

α_x^M is satisfiable iff M behaves well on input x
β_n^M is a propositional tautology iff M is n-consistent

The following conditions are equivalent:

(1) For any $x \in \Sigma^*$ the formula α_x^M is satisfiable and for any n the formula β_n^M is a propositional tautology

(2) M is strong nondeterministic

Theorem (Köbler, Messner)

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

Proof

We define the language L complete for **NP**$\cap$co-**NP**:

$L=\{<M, x, \alpha_x^M, \beta_n^M, \text{Proof}_1, \text{Proof}_2, 0^{n^k+k}>: x\in L(M)\}$

Where:

M is a polynomial time clocked nondeterministic T. M.

n^k+k is its polynomial time bound

x – a string such that $|x|=n$

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} the sequence of zeros (padding)

(i) $L\in$ **NP**

(ii) $L\in$ co-**NP**

(iii) L is a complete language for $\mathbf{NP} \cap$ co-$\mathbf{NP}$

L' – any language in $\mathbf{NP} \cap$ co-$\mathbf{NP}$
L' is accepted by the strong nondeterministic polynomial time clocked Turing machine N ($n^l + 1$ is its polynomial time bound)

f: $\Sigma^* \longrightarrow \Sigma^*$

$$\boxed{f(x) = \langle N, x, \alpha_x^N, \beta_n^N, \text{Proof}_1, \text{Proof}_2, 0^{n^l+1} \rangle}$$

The only task is to construct in polynomial time the strings Proof_1, Proof_2

There exists a proof system for SAT in which the string x is a proof of α_x^N

There exists a proof system for TAUT in which the string 0^n is a proof of β_n^N

4. **Two bridges ...**

(1) Optimal proof systems and complete languages for promise classes

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

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

Open problem

Do there exist complete languages for promise classes?

Theorem (Köbler, Messner)

The following conditions are equivalent

(i) A promise class C has a test set with a p-optimal proof system.

(ii) A promise class C has a complete language.

(2) The natural proof system for SAT and decision versus search in ***NP***

Open problem

Is the natural proof system for SAT p-optimal?

Theorem (Fenner, Fortnow, Naik, Rogers)

The following are equivalent:

(i) The natural proof system for SAT is p-optimal.

(ii) For all $L \in P$ and for all machines M that accept L, $\exists f_M \in$ PF such that $\forall x \in L$ $f_M(x)$ outputs an accepting computation of M on x.

Corollary

If the natural proof system for SAT is p-optimal then for any language $L \in NP$ the following conditions are equivalent:

(i) The decision version of L can be solved in polynomial time.

(ii) Every search problem associated to L can be solved in polynomial time.

decision versus search