Time-Space Tradeoffs for Counting NP Solutions Modulo Integers

Ryan Williams

Carnegie Mellon University

Introduction

Recent progress in understanding the time complexity of hard problems in the space-bounded setting

Introduction

Recent progress in understanding the time complexity of hard problems in the space-bounded setting

- Time-space tradeoffs for SAT and QUANTIFIED BOOLEAN FORMULAS

[Santhanam'01, Fortnow-Lipton-Van Melkebeek-Viglas'05, W'05, Diehl-Van Melkebeek'06]

Introduction

Recent progress in understanding the time complexity of hard problems in the space-bounded setting

- Time-space tradeoffs for SAT and QUANTIFIED BOOLEAN FORMULAS

[Santhanam'01, Fortnow-Lipton-Van Melkebeek-Viglas'05, W'05, Diehl-Van Melkebeek'06]

- SAT requires $\Omega(n^{2\cos(\pi/7)}) \approx n^{1.801}$ time on RAMs using $n^{o(1)}$ space

Also holds for VERTEX COVER, HAMILTONIAN PATH, MAX CUT, *etc.*

Introduction

Recent progress in understanding the time complexity of hard problems in the space-bounded setting

- Time-space tradeoffs for SAT and QUANTIFIED BOOLEAN FORMULAS

[Santhanam'01, Fortnow-Lipton-Van Melkebeek-Viglas'05, W'05, Diehl-Van Melkebeek'06]

- SAT requires $\Omega(n^{2\cos(\pi/7)}) \approx n^{1.801}$ time on RAMs using $n^{o(1)}$ space

Also holds for VERTEX COVER, HAMILTONIAN PATH, MAX CUT, *etc.*

Can similar limitations be proved for other problems?

Introduction

Recent progress in understanding the time complexity of hard problems in the space-bounded setting

• Time-space tradeoffs for SAT and QUANTIFIED BOOLEAN FORMULAS

[Santhanam'01, Fortnow-Lipton-Van Melkebeek-Viglas'05, W'05, Diehl-Van Melkebeek'06]

• SAT requires $\Omega(n^{2\cos(\pi/7)}) \approx n^{1.801}$ time on RAMs using $n^{o(1)}$ space

Also holds for VERTEX COVER, HAMILTONIAN PATH, MAX CUT, *etc.*

Can similar limitations be proved for other problems?

[Diehl-Van Melkebeek], [Viola] Lower bounds for QSAT with randomized algorithms

Introduction

Introduction

Let m be a fixed integer and Π be a combinatorial problem.

Is the number of solutions to a given instance of Π divisible by m?

Introduction

Let m be a fixed integer and Π be a combinatorial problem.

Is the number of solutions to a given instance of Π divisible by m?

MOD$_m$SAT: For formula F, is the number of solutions divisible by m?

[Cai-Hemachandra'92]

Define MOD$_m$P, for which MOD$_m$SAT is a complete problem.

Introduction

Let m be a fixed integer and Π be a combinatorial problem.

Is the number of solutions to a given instance of Π divisible by m?

MOD_mSAT: For formula F, is the number of solutions divisible by m?

[Cai-Hemachandra'92]

Define MOD_mP, for which MOD_mSAT is a complete problem.

Recent Attention: Fueled by Valiant's *accidental algorithms*

– For some $\#$P-complete problems,

- Can determine in P if the number of solutions is divisible by 7,
- But MOD_2P-hard to determine if the number of solutions is divisible by 2.

Complexity of MOD_m P

Complexity of MOD_m P

How difficult is MOD_m SAT, in general?

(Is it just as hard as SAT? Is it harder??)

Complexity of MOD_m P

How difficult is MOD_m SAT, in general?

(Is it just as hard as SAT? Is it harder??)

- (Valiant-Vazirani)

  RP-reduction from SAT to MOD_m SAT

- (Naik-Regan-Sivakumar)

  $O(n \cdot \text{poly}(\log n))$-time randomized reduction from SAT to MOD_2 SAT

- (Toda-Ogihara, Gupta)

  $O(n^{k+1})$-time randomized 2-sided reduction from Σ_kSAT to MOD_2 SAT

Complexity of MOD_m P

How difficult is MOD_m SAT, in general?

(Is it just as hard as SAT? Is it harder??)

- (Valiant-Vazirani)

  RP-reduction from SAT to MOD_m SAT

- (Naik-Regan-Sivakumar)

  $O(n \cdot \mathrm{poly}(\log n))$-time randomized reduction from SAT to MOD_2 SAT

- (Toda-Ogihara, Gupta)

  $O(n^{k+1})$-time randomized 2-sided reduction from Σ_kSAT to MOD_2 SAT

Naive Idea For MOD_m SAT Lower Bounds:

Prove lower bounds for MOD_m SAT *by applying above reduction(s) and appealing to known* SAT *lower bounds.*

MAJOR OBSTACLE: RANDOMNESS

MAJOR OBSTACLE: RANDOMNESS

- Can't use Valiant-Vazirani reduction or its offshoots–
  since they're randomized and space-inefficient

MAJOR OBSTACLE: RANDOMNESS

- Can't use Valiant-Vazirani reduction or its offshoots–
  since they're randomized and space-inefficient
- Even if we could make them space-efficient, we could only say:

  MOD$_m$ SAT has a fast det. alg. $\Longrightarrow$ SAT has a fast rand. alg.

  But we don't know nontrivial randomized time lower bounds for SAT(!)

MAJOR OBSTACLE: RANDOMNESS

- Can't use Valiant-Vazirani reduction or its offshoots–
  since they're randomized and space-inefficient
- Even if we could make them space-efficient, we could only say:

  $$\text{MOD}_m \text{ SAT has a fast det. alg.} \implies \text{SAT has a fast rand. alg.}$$

  But we don't know nontrivial randomized time lower bounds for SAT(!)
- Could we get rid of the randomness?

MAJOR OBSTACLE: RANDOMNESS

- Can't use Valiant-Vazirani reduction or its offshoots–
  since they're randomized and space-inefficient

- Even if we could make them space-efficient, we could only say:

  MOD_m SAT has a fast det. alg. $\Longrightarrow$ SAT has a fast rand. alg.

  But we don't know nontrivial randomized time lower bounds for SAT(!)

- Could we get rid of the randomness?

- **Derandomizations? We don't know no stinking derandomizations!**
  (But we have good reason to believe they exist [Klivans-Van Melkebeek])

Main Result

Main Result

Time-Space LBs for NP $\longmapsto$ Time-Space LBs for $\mathsf{MOD}_m\mathsf{P}$

Main Result

Time-Space LBs for NP $\longmapsto$ Time-Space LBs for $\mathrm{MOD}_m\mathrm{P}$

Transfer Principle for Time-Space Lower Bounds:

If there is an *alternation-trading proof* that

SAT cannot be solved in t time and s space,

Main Result

Time-Space LBs for NP $\quad \mapsto \quad$ Time-Space LBs for $\mathsf{MOD}_m\mathsf{P}$

Transfer Principle for Time-Space Lower Bounds:

If there is an *alternation-trading proof* that

SAT cannot be solved in t time and s space,

then for every $\varepsilon > 0$ and primes $p \neq q$, there is a proof that:

One of MOD_p SAT or MOD_q SAT can't be solved
in $t^{1-\varepsilon}$ time and $s^{1-\varepsilon}$ space.

Outline of Talk

- Introduction
- Some Notation
- Some Preliminaries
- Alternation-Trading Proofs
- Transfer Principle

Some Notation

Some Notation

Recall $\mathsf{DTISP}[t, s]$ is the class of problems solvable in time t and space s.

Define $\mathsf{DTS}[t] := \mathsf{DTISP}[t^{1+o(1)}, n^{o(1)}]$.

Some Notation

Recall $\mathsf{DTISP}[t, s]$ is the class of problems solvable in time t and space s.

Define $\mathsf{DTS}[t] := \mathsf{DTISP}[t^{1+o(1)}, n^{o(1)}]$.

Notation for Alternating Classes:

- $(\exists\ f(n))\ C$ = class of problems solved by a machine that: $\exists$-guesses $f(n)^{1+o(1)}$ bits, then runs a C-machine.
  Example: $\mathsf{NTIME}[n^{1+o(1)}] = (\exists\ n)\mathsf{DTIME}[n^{1+o(1)}]$.
- $(\forall\ f(n))\ C$ defined similarly.

Some Notation

Recall $\mathsf{DTISP}[t, s]$ is the class of problems solvable in time t and space s.

Define $\mathsf{DTS}[t] := \mathsf{DTISP}[t^{1+o(1)}, n^{o(1)}]$.

Notation for Alternating Classes:

- $(\exists\ f(n))\ C$ = class of problems solved by a machine that:
  $\exists$-guesses $f(n)^{1+o(1)}$ bits, then runs a C-machine.
  Example: $\mathsf{NTIME}[n^{1+o(1)}] = (\exists\ n)\mathsf{DTIME}[n^{1+o(1)}]$.

- $(\forall\ f(n))\ C$ defined similarly.

- $(\mathsf{MOD}_m\ f(n))\ C$ = class of problems solved by a machine that:
  guesses $y : |y| = f(n)^{1+o(1)}$, runs a C-machine N,
  accepts iff the number of y that make N accept is divisible by m.
  Example: $\mathsf{MOD}_m\mathsf{TIME}[n^{1+o(1)}] = (\mathsf{MOD}_m\ n)\mathsf{DTIME}[n^{1+o(1)}]$.

Some Preliminaries

Alternation Speedup Theorem *(Trading Time For Alternations)*

Some Preliminaries

Alternation Speedup Theorem *(Trading Time For Alternations)*

[Kannan] $\mathsf{DTISP}[t, s] \subseteq (\exists\, b \cdot s)(\forall\ \log b)\mathsf{DTISP}[t/b, s]$

[Fortnow-Van Melkebeek] $\mathsf{DTISP}[t, s] \subseteq (\forall\, b \cdot s)(\exists\ \log b)\mathsf{DTISP}[t/b, s]$

Some Preliminaries

Alternation Speedup Theorem *(Trading Time For Alternations)*

[Kannan] $\mathsf{DTISP}[t, s] \subseteq (\exists\, b \cdot s)(\forall\ \log b)\mathsf{DTISP}[t/b, s]$

[Fortnow-Van Melkebeek] $\mathsf{DTISP}[t, s] \subseteq (\forall\, b \cdot s)(\exists\ \log b)\mathsf{DTISP}[t/b, s]$

Proof Sketches:

[Kannan]

Existentially guess configs $C_1, \ldots, C_{b+1}$ of a DTISP machine.

Universally guess $i \in [b]$.

Accept iff $C_i \vdash^{t/b} C_{i+1}$ and C_1 is initial and C_{b+1} is accepting.

Some Preliminaries

Alternation Speedup Theorem *(Trading Time For Alternations)*

[Kannan] $\mathsf{DTISP}[t,s] \subseteq (\exists\, b \cdot s)(\forall\ \log b)\mathsf{DTISP}[t/b,s]$

[Fortnow-Van Melkebeek] $\mathsf{DTISP}[t,s] \subseteq (\forall\, b \cdot s)(\exists\ \log b)\mathsf{DTISP}[t/b,s]$

Proof Sketches:

[Kannan]

Existentially guess configs $C_1, \ldots, C_{b+1}$ of a DTISP machine.

Universally guess $i \in [b]$.

Accept iff $C_i \vdash^{t/b} C_{i+1}$ and C_1 is initial and C_{b+1} is accepting.

[Fortnow-van Melkebeek]

Universally guess configs $C_1, \ldots, C_{b+1}$ of a DTISP machine.

Existentially guess $i \in [b]$.

Accept iff ($\neg(C_i \vdash^{t/b} C_{i+1})$ or C_1 is not initial or C_{b+1} is not rejecting).

A Slowdown Lemma *(Trading Alternations For Time)*

Idea: The assumption $\mathsf{NTIME}[n] \subseteq \mathsf{DTS}[n^c]$

lets you remove alternations from a computation at little time cost

A Slowdown Lemma *(Trading Alternations For Time)*

Idea: The assumption $\mathsf{NTIME}[n] \subseteq \mathsf{DTS}[n^c]$

lets you remove alternations from a computation at little time cost

Let $c \geq 1$.

Theorem: For all $b \geq a \geq 1$,

$\mathsf{NTIME}[n] \subseteq \mathsf{DTS}[n^c] \Longrightarrow (\exists n^a)(\forall n^b)\mathsf{DTS}[n^b] \subseteq (\exists n^a)\mathsf{DTS}[n^{bc}]$

and $(\forall n^a)(\exists n^b)\mathsf{DTS}[n^b] \subseteq (\forall n^a)\mathsf{DTS}[n^{bc}]$.

Alternation-Trading Proofs

Alternation-Trading Proofs

An *alternation-trading proof* that $\textsf{SAT} \notin \textsf{DTS}[n^c]$ works by:

- Showing $\textsf{NTIME}[n] \not\subseteq \textsf{DTS}[n^c]$
- Appealing to strong completeness properties of SAT

Alternation-Trading Proofs

An *alternation-trading proof* that $\textsf{SAT} \notin \textsf{DTS}[n^c]$ works by:

- Showing $\textsf{NTIME}[n] \not\subseteq \textsf{DTS}[n^c]$
- Appealing to strong completeness properties of SAT

Show $\textsf{NTIME}[n] \not\subseteq \textsf{DTS}[n^c]$ by assuming the opposite, and applying three rules in a way that $\textsf{DTS}[t] \subseteq \textsf{DTS}[t^{1-\varepsilon}]$ (a contradiction) can be derived:

Alternation-Trading Proofs

An *alternation-trading proof* that $\mathsf{SAT} \notin \mathsf{DTS}[n^c]$ works by:

- Showing $\mathsf{NTIME}[n] \not\subseteq \mathsf{DTS}[n^c]$
- Appealing to strong completeness properties of SAT

Show $\mathsf{NTIME}[n] \not\subseteq \mathsf{DTS}[n^c]$ by assuming the opposite, and applying three rules in a way that $\mathsf{DTS}[t] \subseteq \mathsf{DTS}[t^{1-\varepsilon}]$ (a contradiction) can be derived:

1. (Speedup) $\mathsf{DTS}[n^b] \subseteq (\exists\, n^a)(\forall\, \log n)\mathsf{DTS}[n^{b-a}]$
   $\mathsf{DTS}[n^b] \subseteq (\forall\, n^a)(\exists\, \log n)\mathsf{DTS}[n^{b-a}]$

Alternation-Trading Proofs

An *alternation-trading proof* that $\text{SAT} \notin \mathsf{DTS}[n^c]$ works by:

- Showing $\mathsf{NTIME}[n] \not\subseteq \mathsf{DTS}[n^c]$
- Appealing to strong completeness properties of SAT

Show $\mathsf{NTIME}[n] \not\subseteq \mathsf{DTS}[n^c]$ by assuming the opposite, and applying three rules in a way that $\mathsf{DTS}[t] \subseteq \mathsf{DTS}[t^{1-\varepsilon}]$ (a contradiction) can be derived:

1. (Speedup) $\mathsf{DTS}[n^b] \subseteq (\exists\, n^a)(\forall\, \log n)\mathsf{DTS}[n^{b-a}]$

   $\mathsf{DTS}[n^b] \subseteq (\forall\, n^a)(\exists\, \log n)\mathsf{DTS}[n^{b-a}]$

2. (Slowdown) $(\exists\, n^a)(\forall\, n^b)\mathsf{DTS}[n^b] \subseteq (\exists\, n^a)\mathsf{DTS}[n^{bc}]$

   and $(\forall\, n^a)(\exists\, n^b)\mathsf{DTS}[n^b] \subseteq (\forall\, n^a)\mathsf{DTS}[n^{bc}]$.

Alternation-Trading Proofs

An *alternation-trading proof* that $\textsf{SAT} \notin \textsf{DTS}[n^c]$ works by:

- Showing $\textsf{NTIME}[n] \not\subseteq \textsf{DTS}[n^c]$
- Appealing to strong completeness properties of SAT

Show $\textsf{NTIME}[n] \not\subseteq \textsf{DTS}[n^c]$ by assuming the opposite, and applying three rules in a way that $\textsf{DTS}[t] \subseteq \textsf{DTS}[t^{1-\varepsilon}]$ (a contradiction) can be derived:

1. (Speedup) $\textsf{DTS}[n^b] \subseteq (\exists\, n^a)(\forall\, \log n)\textsf{DTS}[n^{b-a}]$
   $\textsf{DTS}[n^b] \subseteq (\forall\, n^a)(\exists\, \log n)\textsf{DTS}[n^{b-a}]$
2. (Slowdown) $(\exists\, n^a)(\forall\, n^b)\textsf{DTS}[n^b] \subseteq (\exists\, n^a)\textsf{DTS}[n^{bc}]$
   and $(\forall\, n^a)(\exists\, n^b)\textsf{DTS}[n^b] \subseteq (\forall\, n^a)\textsf{DTS}[n^{bc}]$.
3. (Combination) $(\exists\, n^a)(\exists\, n^b)\textsf{DTS}[n^d] \subseteq (\exists\, n^a + n^b)\textsf{DTS}[n^d]$
   $(\forall\, n^a)(\forall\, n^b)\textsf{DTS}[n^d] \subseteq (\forall\, n^a + n^b)\textsf{DTS}[n^d]$

Example: Alternation-Trading Proofs

There is an alternation-trading proof of

$$\textsf{SAT} \notin \textsf{DTS}[n^{\sqrt{2}-\varepsilon}] \quad \text{[Lipton-Viglas'99]}$$

because if $\textsf{NTIME}[n] \subseteq \textsf{DTS}[n^{\sqrt{2}-\varepsilon}]$, then

Example: Alternation-Trading Proofs

There is an alternation-trading proof of

$$\textsf{SAT} \notin \textsf{DTS}[n^{\sqrt{2}-\varepsilon}] \quad \text{[Lipton-Viglas'99]}$$

because if $\textsf{NTIME}[n] \subseteq \textsf{DTS}[n^{\sqrt{2}-\varepsilon}]$, then

$\textsf{DTS}[n^2] \subseteq (\exists\, n)(\forall\, \log n)\textsf{DTS}[n]$ (Speedup)

Example: Alternation-Trading Proofs

There is an alternation-trading proof of

$$\textsf{SAT} \notin \textsf{DTS}[n^{\sqrt{2}-\varepsilon}] \quad \text{[Lipton-Viglas'99]}$$

because if $\textsf{NTIME}[n] \subseteq \textsf{DTS}[n^{\sqrt{2}-\varepsilon}]$, then

$$\textsf{DTS}[n^2] \subseteq (\exists\, n)(\forall\ \log n)\textsf{DTS}[n] \text{ (Speedup)}$$

$$\subseteq (\exists\, n)\textsf{DTS}[n^c] \text{ (Slowdown)}$$

Example: Alternation-Trading Proofs

There is an alternation-trading proof of

$$\textsf{SAT} \notin \textsf{DTS}[n^{\sqrt{2}-\varepsilon}] \quad \text{[Lipton-Viglas'99]}$$

because if $\textsf{NTIME}[n] \subseteq \textsf{DTS}[n^{\sqrt{2}-\varepsilon}]$, then

$$\textsf{DTS}[n^2] \subseteq (\exists\, n)(\forall\ \log n)\textsf{DTS}[n] \text{ (Speedup)}$$

$$\subseteq (\exists\, n)\textsf{DTS}[n^c] \text{ (Slowdown)}$$

$$\subseteq \textsf{DTS}[n^{c^2}] \text{ (Slowdown)}$$

Example: Alternation-Trading Proofs

There is an alternation-trading proof of

$$\text{SAT} \notin \mathsf{DTS}[n^{\sqrt{2}-\varepsilon}] \quad \text{[Lipton-Viglas'99]}$$

because if $\mathsf{NTIME}[n] \subseteq \mathsf{DTS}[n^{\sqrt{2}-\varepsilon}]$, then

$$\begin{aligned} \mathsf{DTS}[n^2] &\subseteq (\exists\, n)(\forall\, \log n)\mathsf{DTS}[n] \text{ (Speedup)} \\ &\subseteq (\exists\, n)\mathsf{DTS}[n^c] \text{ (Slowdown)} \\ &\subseteq \mathsf{DTS}[n^{c^2}] \text{ (Slowdown)} \end{aligned}$$

Contradiction when $c^2 < 2$.

Example: Alternation-Trading Proofs

There is an alternation-trading proof of

$$\textsf{SAT} \notin \textsf{DTS}[n^{\sqrt{2}-\varepsilon}] \quad \text{[Lipton-Viglas'99]}$$

because if $\textsf{NTIME}[n] \subseteq \textsf{DTS}[n^{\sqrt{2}-\varepsilon}]$, then

$$\begin{aligned} \textsf{DTS}[n^2] &\subseteq (\exists\, n)(\forall\, \log n)\textsf{DTS}[n] \text{ (Speedup)} \\ &\subseteq (\exists\, n)\textsf{DTS}[n^c] \text{ (Slowdown)} \\ &\subseteq \textsf{DTS}[n^{c^2}] \text{ (Slowdown)} \end{aligned}$$

Contradiction when $c^2 < 2$.

Can prove $\textsf{SAT} \notin \textsf{DTS}[n^{2\cos(\pi/7)-\varepsilon}]$ using alternation-trading.

Transfer Principle for Time-Space Lower Bounds

Transfer Principle for Time-Space Lower Bounds

Let $c > 1$. If there is an *alternation-trading proof* that

Transfer Principle for Time-Space Lower Bounds

Let $c > 1$. If there is an *alternation-trading proof* that

$$\text{SAT} \notin \text{DTS}[n^c],$$

then for every $\varepsilon > 0$ and primes $p \neq q$, there is a proof that:

$$\text{Either } \text{MOD}_p\,\text{SAT} \notin \text{DTS}[n^{c-\varepsilon}],$$
$$\text{or } \text{MOD}_q\,\text{SAT} \notin \text{DTS}[n^{c-\varepsilon}].$$

Transfer Principle for Time-Space Lower Bounds

Let $c > 1$. If there is an *alternation-trading proof* that

$$\textsc{Sat} \notin \mathsf{DTS}[n^c],$$

then for every $\varepsilon > 0$ and primes $p \neq q$, there is a proof that:

$$\text{Either } \mathsf{MOD}_p\,\textsc{Sat} \notin \mathsf{DTS}[n^{c-\varepsilon}],$$
$$\text{or } \mathsf{MOD}_q\,\textsc{Sat} \notin \mathsf{DTS}[n^{c-\varepsilon}].$$

Corollary: For every prime p (except for possibly one of them)

$$\mathsf{MOD}_p\,\textsc{Sat} \notin \mathsf{DTS}[n^{1.8}].$$

Transfer Principle for Time-Space Lower Bounds

Let $c > 1$. If there is an *alternation-trading proof* that

$$\text{SAT} \notin \text{DTS}[n^c],$$

then for every $\varepsilon > 0$ and primes $p \neq q$, there is a proof that:

$$\text{Either } \text{MOD}_p\,\text{SAT} \notin \text{DTS}[n^{c-\varepsilon}],$$
$$\text{or } \text{MOD}_q\,\text{SAT} \notin \text{DTS}[n^{c-\varepsilon}].$$

Corollary: For every prime p (except for possibly one of them)

$$\text{MOD}_p\,\text{SAT} \notin \text{DTS}[n^{1.8}].$$

Corollary: $\text{MOD}_6\,\text{SAT} \notin \text{DTS}[n^{1.8}]$.

Proof: If not, then both $\text{MOD}_2\,\text{SAT}$ and $\text{MOD}_3\,\text{SAT}$ are in $\text{DTS}[n^{1.8}]$, a contradiction.

Transfer Principle: Proof Idea

Transfer Principle: Proof Idea

- $\mathsf{MOD}_p\,\textsc{Sat} \in \mathsf{DTS}[n^c] \Rightarrow \mathsf{MOD}_p\mathsf{TIME}[n] \subseteq \mathsf{DTS}[n^c]$

  (easy– uses reduction from $\mathsf{NTIME}[n]$ *to* $\textsc{Sat}$*)*

Transfer Principle: Proof Idea

- $\mathsf{MOD}_p\,\textsc{Sat} \in \mathsf{DTS}[n^c] \Rightarrow \mathsf{MOD}_p\mathsf{TIME}[n] \subseteq \mathsf{DTS}[n^c]$

  (easy– uses reduction from $\mathsf{NTIME}[n]$ *to* $\textsc{Sat}$*)*

- The alternation-trading rules have "modular counting counterparts":

For all primes $p \neq q$ and $\varepsilon > 0$,

(Speedup) $\mathsf{DTS}[n^b] \subseteq (\mathsf{MOD}_p\; n^a)(\mathsf{MOD}_q\; \log n)\mathsf{DTS}[n^{b-a+\varepsilon}]$

(Slowdown) $\mathsf{MOD}_q\mathsf{TIME}[n] \subseteq \mathsf{DTS}[n^c]$ implies

$(\mathsf{MOD}_p\; n^a)(\mathsf{MOD}_q\; n^b)\mathsf{DTS}[n^b] \subseteq (\mathsf{MOD}_p\; n^a)\mathsf{DTS}[n^{bc}]$

(Combination)

$(\mathsf{MOD}_p\; n^a)(\mathsf{MOD}_p\; n^b)\mathsf{DTS}[n^d] \subseteq (\mathsf{MOD}_p\; n^a + n^b)\mathsf{DTS}[n^d]$

Transfer Principle: Proof Idea

- $\mathsf{MOD}_p\,\mathsf{SAT} \in \mathsf{DTS}[n^c] \Rightarrow \mathsf{MOD}_p\mathsf{TIME}[n] \subseteq \mathsf{DTS}[n^c]$

  (easy– uses reduction from $\mathsf{NTIME}[n]$ *to* SAT*)*

- The alternation-trading rules have "modular counting counterparts":

For all primes $p \neq q$ and $\varepsilon > 0$,

(Speedup) $\mathsf{DTS}[n^b] \subseteq (\mathsf{MOD}_p\, n^a)(\mathsf{MOD}_q\ \log n)\mathsf{DTS}[n^{b-a+\varepsilon}]$

(Slowdown) $\mathsf{MOD}_q\mathsf{TIME}[n] \subseteq \mathsf{DTS}[n^c]$ implies
$(\mathsf{MOD}_p\, n^a)(\mathsf{MOD}_q\, n^b)\mathsf{DTS}[n^b] \subseteq (\mathsf{MOD}_p\, n^a)\mathsf{DTS}[n^{bc}]$

(Combination)
$(\mathsf{MOD}_p\, n^a)(\mathsf{MOD}_p\, n^b)\mathsf{DTS}[n^d] \subseteq (\mathsf{MOD}_p\, n^a + n^b)\mathsf{DTS}[n^d]$

Informally, if $\mathsf{SAT} \in \mathsf{DTS}[n^c]$ implies $\mathsf{DTS}[t] \subseteq \mathsf{DTS}[t^{1-\varepsilon}]$, then $\mathsf{MOD}_p\,\mathsf{SAT}$ and $\mathsf{MOD}_q\,\mathsf{SAT}$ are in $\mathsf{DTS}[n^{c-\varepsilon}]$ also implies it.

Speedup of DTISP via Modular Counting

Speedup of DTISP via Modular Counting

Theorem: For all $\varepsilon > 0$ and $b \geq a \geq 1$,

$$\mathsf{DTS}[n^b] \subseteq (\mathsf{MOD}_p\, n^a)(\mathsf{MOD}_q\ \log n)\mathsf{DTS}[n^{b-a+\varepsilon}]$$

Speedup of DTISP via Modular Counting

Theorem: For all $\varepsilon > 0$ and $b \geq a \geq 1$,

$$\mathsf{DTS}[n^b] \subseteq (\mathsf{MOD}_p\ n^a)(\mathsf{MOD}_q\ \log n)\mathsf{DTS}[n^{b-a+\varepsilon}]$$

Proof Steps:

1. Convert $\mathsf{DTS}[n^b]$ machine into a "canonical form" that runs in time $n^{b+\varepsilon}$ and space $n^{o(1)}$

2. By essentially replacing "$\exists$" with "MOD_p" and "$\forall$" with "MOD_q", the Alternating Speedup Theorem works on a canonical machine.

$$\mathsf{DTS}[n^b] \subseteq (\mathsf{MOD}_p\, n^a)(\mathsf{MOD}_q\ \log n)\mathsf{DTS}[n^{b-a+\varepsilon}]$$

1. Convert $\mathsf{DTS}[n^b]$ machine into a "canonical form"

$$\mathsf{DTS}[n^b] \subseteq (\mathsf{MOD}_p\, n^a)(\mathsf{MOD}_q\ \log n)\mathsf{DTS}[n^{b-a+\varepsilon}]$$

1. Convert $\mathsf{DTS}[n^b]$ machine into a "canonical form"

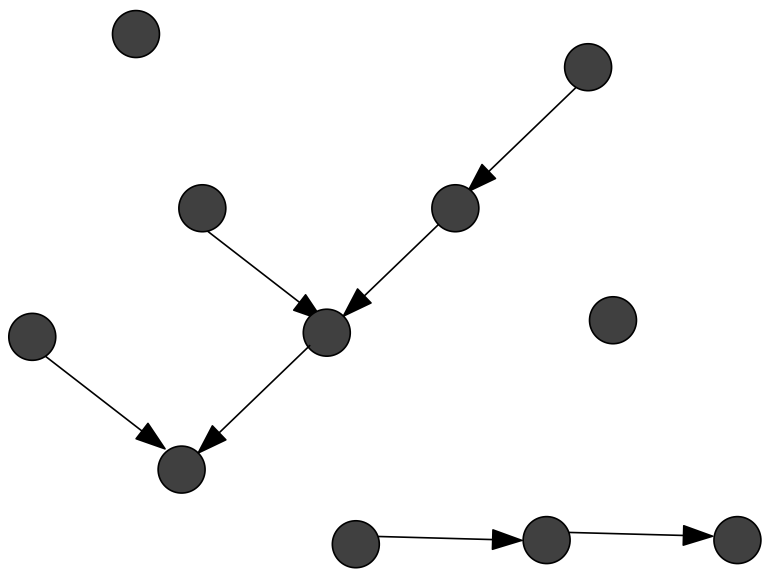
Recall the notion of a configuration graph for M on x:

$$\mathsf{DTS}[n^b] \subseteq (\mathsf{MOD}_p\, n^a)(\mathsf{MOD}_q\, \log n)\mathsf{DTS}[n^{b-a+\varepsilon}]$$

1. Convert $\mathsf{DTS}[n^b]$ machine into a "canonical form"

Recall the notion of a configuration graph for M on x:

$G_{M,x}$:

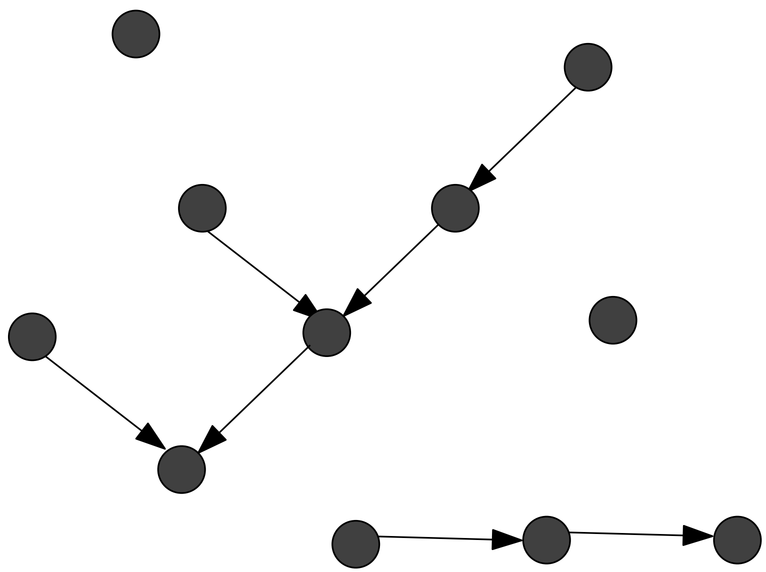


$\mathsf{DTS}[n^b] \subseteq (\mathsf{MOD}_p\, n^a)(\mathsf{MOD}_q\ \log n)\mathsf{DTS}[n^{b-a+\varepsilon}]$

1. Convert $\mathsf{DTS}[n^b]$ machine into a "canonical form"

Recall the notion of a configuration graph for M on x:

$G_{M,x}$:



Nodes = Configurations C of $M(x)$, Edge $(C, C') \iff C \vdash C'$

M is deterministic $\Longrightarrow outdeg(G_{M,x}) \leq 1$.

$$\mathsf{DTS}[n^b] \subseteq (\mathsf{MOD}_p\, n^a)(\mathsf{MOD}_q\, \log n)\mathsf{DTS}[n^{b-a+\varepsilon}]$$

1. Convert $\mathsf{DTS}[n^b]$ machine into a "canonical form"

A deterministic machine M is *canonical* if, for every input x,

$$indeg(G_{M,x}) = outdeg(G_{M,x}) = 1.$$

$$\mathsf{DTS}[n^b] \subseteq (\mathsf{MOD}_p\, n^a)(\mathsf{MOD}_q\, \log n)\mathsf{DTS}[n^{b-a+\varepsilon}]$$

1. Convert $\mathsf{DTS}[n^b]$ machine into a "canonical form"

A deterministic machine M is *canonical* if, for every input x,

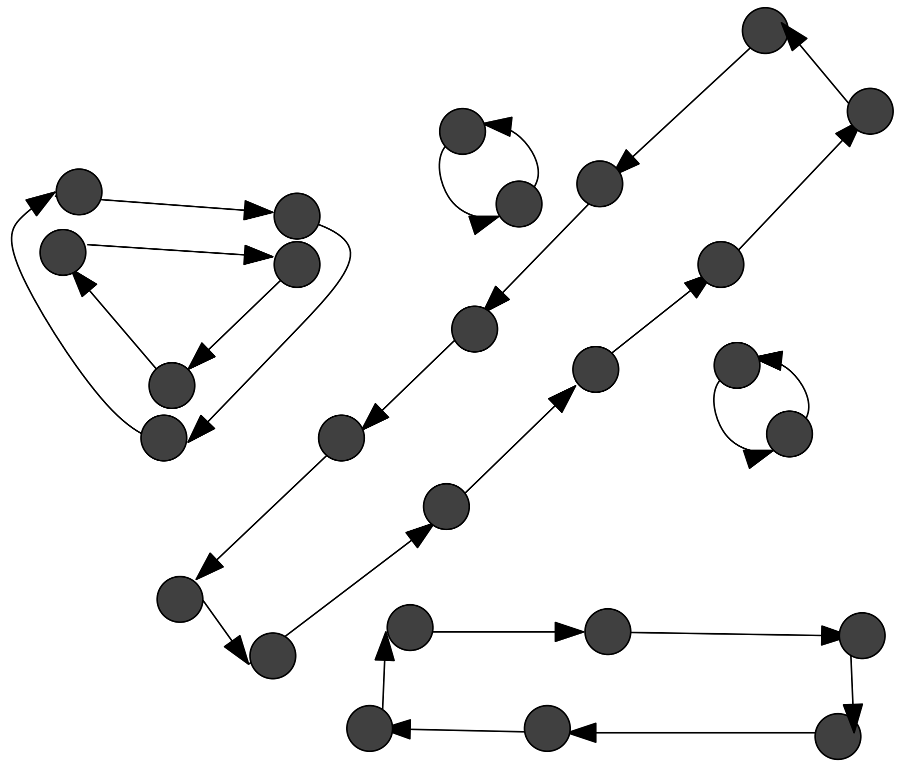
$$indeg(G_{M,x}) = outdeg(G_{M,x}) = 1.$$

$\mathsf{DTS}[n^b] \subseteq (\mathsf{MOD}_p\, n^a)(\mathsf{MOD}_q\, \log n)\mathsf{DTS}[n^{b-a+\varepsilon}]$

1. Convert $\mathsf{DTS}[n^b]$ machine into a "canonical form"

A deterministic machine M is *canonical* if, for every input x,

$$indeg(G_{M,x}) = outdeg(G_{M,x}) = 1.$$



$$\mathsf{DTS}[n^b] \subseteq (\mathsf{MOD}_p\, n^a)(\mathsf{MOD}_q\, \log n)\mathsf{DTS}[n^{b-a+\varepsilon}]$$

1. Convert $\mathsf{DTS}[n^b]$ machine into a "canonical form"

More Details:

How can we make a machine canonical?

$$\mathsf{DTS}[n^b] \subseteq (\mathsf{MOD}_p\, n^a)(\mathsf{MOD}_q\, \log n)\mathsf{DTS}[n^{b-a+\varepsilon}]$$

1. Convert $\mathsf{DTS}[n^b]$ machine into a “canonical form”

More Details:

How can we make a machine canonical?

First, make it reversible

$$\mathsf{DTS}[n^b] \subseteq (\mathsf{MOD}_p\, n^a)(\mathsf{MOD}_q\ \log n)\mathsf{DTS}[n^{b-a+\varepsilon}]$$

1. Convert $\mathsf{DTS}[n^b]$ machine into a "canonical form"

More Details:

How can we make a machine canonical?

First, make it reversible

A deterministic machine M is *reversible* if, for every input x,

$$indeg(G_{M,x}) \leq 1.$$

$$\mathsf{DTS}[n^b] \subseteq (\mathsf{MOD}_p\, n^a)(\mathsf{MOD}_q\ \log n)\mathsf{DTS}[n^{b-a+\varepsilon}]$$

1. Convert $\mathsf{DTS}[n^b]$ machine into a "canonical form"

More Details:

How can we make a machine canonical?

First, make it reversible

A deterministic machine M is *reversible* if, for every input x,

$$indeg(G_{M,x}) \leq 1.$$

Define

rTISP$[t, s]$ =

problems solvable by reversible machines in time t, space s.

$\mathsf{DTS}[n^b] \subseteq (\mathsf{MOD}_p\, n^a)(\mathsf{MOD}_q\ \log n)\mathsf{DTS}[n^{b-a+\varepsilon}]$

1. Convert $\mathsf{DTS}[n^b]$ machine into a "canonical form"

More Details:

How can we make a machine canonical?

First, make it reversible

A deterministic machine M is *reversible* if, for every input x,

$$indeg(G_{M,x}) \leq 1.$$

Define

rTISP$[t, s]$ =

problems solvable by reversible machines in time t, space s.

Theorem: [Bennett'89] $\mathsf{DTISP}[t, s] \subseteq$ rTISP$[t^{1+\varepsilon}, s \log t]$.

$$\mathsf{DTS}[n^b] \subseteq (\mathsf{MOD}_p\, n^a)(\mathsf{MOD}_q\ \log n)\mathsf{DTS}[n^{b-a+\varepsilon}]$$

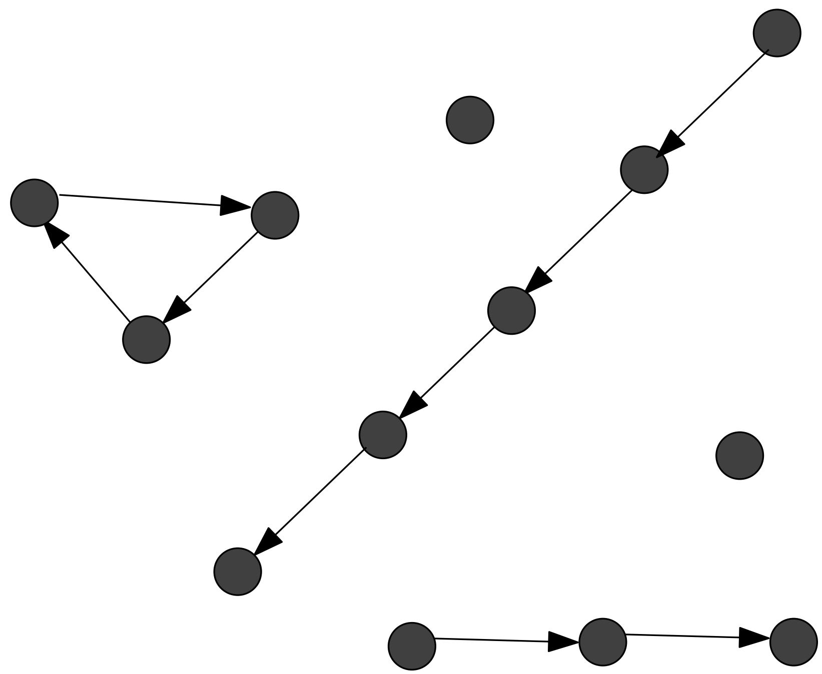
Conversion to Canonical Machines

$$\mathsf{DTS}[n^b] \subseteq (\mathsf{MOD}_p\, n^a)(\mathsf{MOD}_q\ \log n)\mathsf{DTS}[n^{b-a+\varepsilon}]$$

Conversion to Canonical Machines

Take a $\mathsf{DTS}[n^b]$ machine M and make it reversible using Bennett.

On an input, its config graph looks like:



$$\mathsf{DTS}[n^b] \subseteq (\mathsf{MOD}_p\, n^a)(\mathsf{MOD}_q\ \log n)\mathsf{DTS}[n^{b-a+\varepsilon}]$$

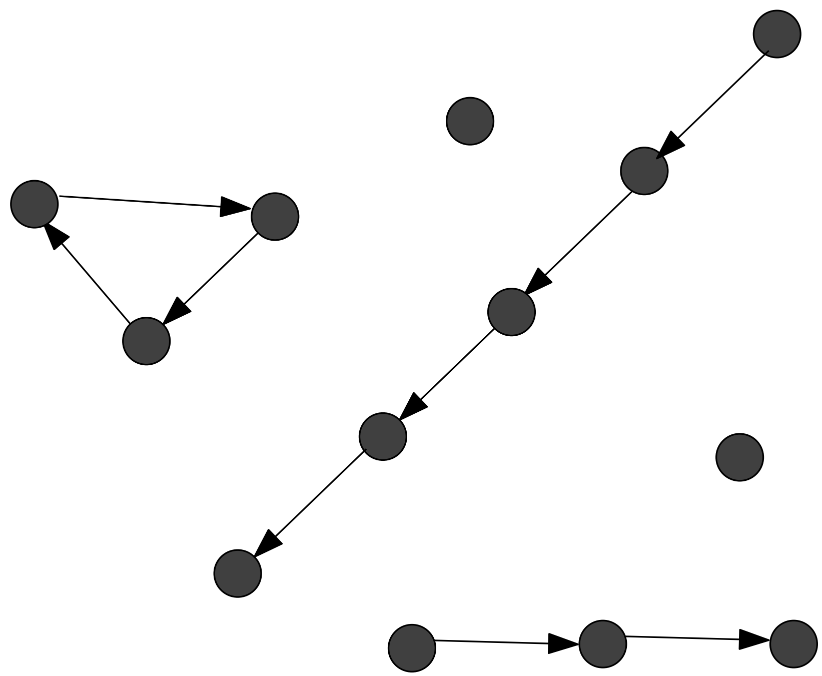
From Reversible Machine to Canonical Machine

To make M canonical, we make another machine that simulates it, but starts running “backwards” when it reaches a dead end.

$$\mathsf{DTS}[n^b] \subseteq (\mathsf{MOD}_p\, n^a)(\mathsf{MOD}_q\; \log n)\mathsf{DTS}[n^{b-a+\varepsilon}]$$

From Reversible Machine to Canonical Machine

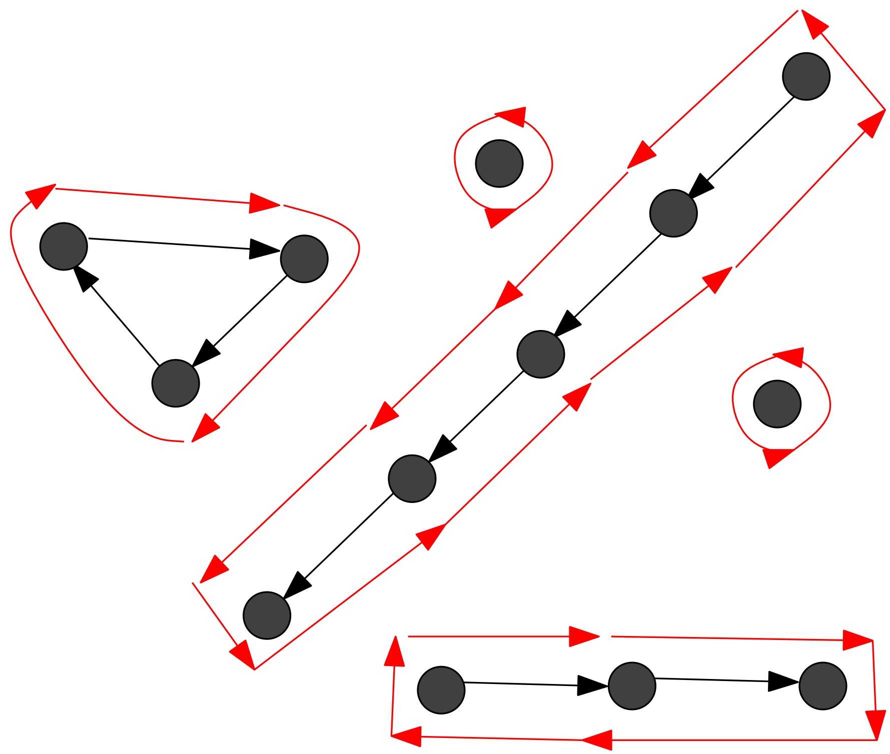
To make M canonical, we make another machine that simulates it, but starts running “backwards” when it reaches a dead end.



$$\mathsf{DTS}[n^b] \subseteq (\mathsf{MOD}_p\; n^a)(\mathsf{MOD}_q\; \log n)\mathsf{DTS}[n^{b-a+\varepsilon}]$$

From Reversible Machine to Canonical Machine

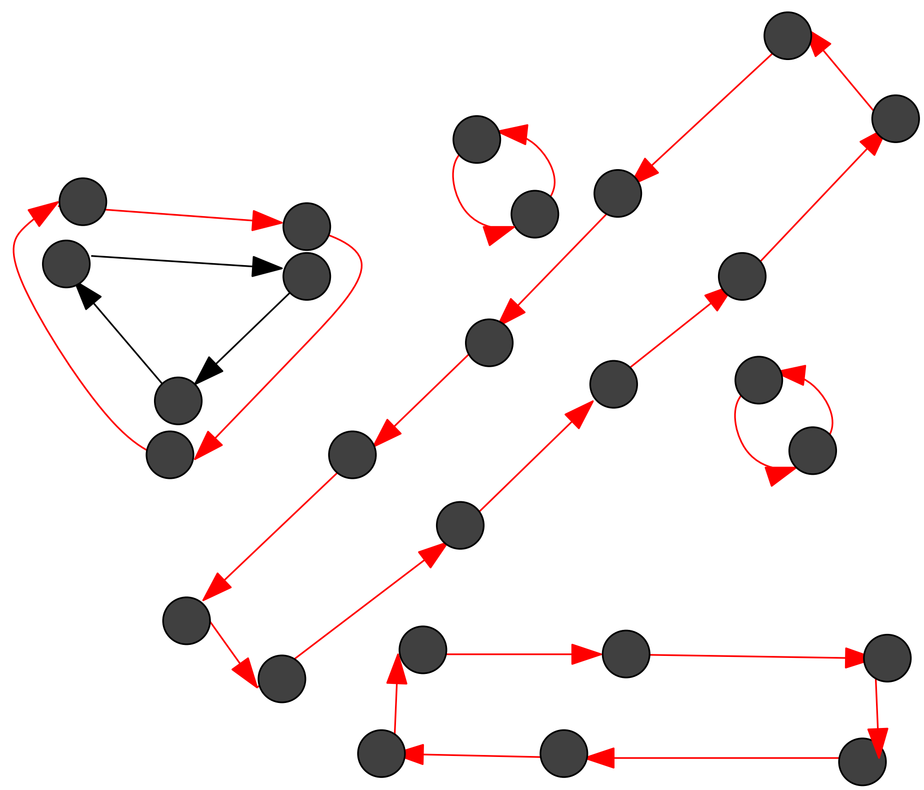
To make M canonical, we make another machine that simulates it, but starts running “backwards” when it reaches a dead end.



$$\mathsf{DTS}[n^b] \subseteq (\mathsf{MOD}_p\, n^a)(\mathsf{MOD}_q\ \log n)\mathsf{DTS}[n^{b-a+\varepsilon}]$$

From Reversible Machine to Canonical Machine

To make M canonical, we make another machine that simulates it, but starts running "backwards" when it reaches a dead end.



The Speedup Theorem on a Canonical Machine

The Speedup Theorem on a Canonical Machine

1. Convert $\mathsf{DTS}[n^b]$ machine into a "canonical form"

2. By replacing "$\exists$" with "MOD_p" and "$\forall$" with "MOD_q", the Alternating Speedup Theorem works on a canonical machine.

The Speedup Theorem on a Canonical Machine

1. Convert $\mathsf{DTS}[n^b]$ machine into a "canonical form"

2. By replacing "$\exists$" with "MOD_p" and "$\forall$" with "MOD_q", the Alternating Speedup Theorem works on a canonical machine.

More Details: Let T be the runtime of the canonical machine.

The Speedup Theorem on a Canonical Machine

1. Convert $\mathsf{DTS}[n^b]$ machine into a "canonical form"

2. By replacing "$\exists$" with "MOD_p" and "$\forall$" with "MOD_q", the Alternating Speedup Theorem works on a canonical machine.

More Details: Let T be the runtime of the canonical machine.

- If we count (mod p) the number of config sequences $C_1, \ldots, C_{b+1}$ where the number of i satisfying $C_i \vdash^{T/b} C_{i+1}$ is divisible by q, this mod-p residue is *different* in the accepting and rejecting cases.

The Speedup Theorem on a Canonical Machine

1. Convert $\mathsf{DTS}[n^b]$ machine into a "canonical form"

2. By replacing "$\exists$" with "MOD_p" and "$\forall$" with "MOD_q", the Alternating Speedup Theorem works on a canonical machine.

More Details: Let T be the runtime of the canonical machine.

- If we count (mod p) the number of config sequences $C_1, \ldots, C_{b+1}$ where the number of i satisfying $C_i \vdash^{T/b} C_{i+1}$ is divisible by q, this mod-p residue is *different* in the accepting and rejecting cases.

- Can pre-compute the mod-p residue for the accepting case.

The Key to the Counting Simulation

The Key to the Counting Simulation

For a canonical machine, input x, integer ℓ, and given configuration C:

- **There are unique configurations D and E s.t. $C \vdash^{\ell} D$ and $E \vdash^{\ell} C$.**

The Key to the Counting Simulation

For a canonical machine, input x, integer ℓ, and given configuration C:

- **There are unique configurations D and E s.t. $C \vdash^{\ell} D$ and $E \vdash^{\ell} C$.**

A configuration sequence $C_1, \ldots, C_{b+1}$ has k mistakes for $M(x)$ iff there are exactly k pairs (C_i, C_{i+1}) s.t. $\neg(C_i \vdash^{T/b} C_{i+1})$.

The Key to the Counting Simulation

For a canonical machine, input x, integer ℓ, and given configuration C:

- **There are unique configurations D and E s.t. $C \vdash^{\ell} D$ and $E \vdash^{\ell} C$.**

A configuration sequence $C_1, \ldots, C_{b+1}$ has k mistakes for $M(x)$ iff there are exactly k pairs (C_i, C_{i+1}) s.t. $\neg(C_i \vdash^{T/b} C_{i+1})$.

Can show that the number of configuration sequences with k mistakes:

- depends *only* on time-space usage and accepting/rejecting condition
- *not* on subtle properties of the machine's behavior

Formal Statement

Let a *complete configuration sequence* $C_1, \ldots, C_{b+1}$ have C_1 as initial and C_{b+1} as accepting.

Formal Statement

Let a *complete configuration sequence* $C_1, \ldots, C_{b+1}$ have C_1 as initial and C_{b+1} as accepting.

Counting Lemma: Let n be an integer, let $k \in \{0, 1, \ldots, b+1\}$, and let $\hat{M}$ be canonical. Then there are positive integers $N_A(n, k, b)$ and $N_R(n, k, b)$ such that, for all inputs x of length n:

1. If $\hat{M}(x)$ accepts, then the number of complete b-configuration sequences for $\hat{M}(x)$ with k mistakes is $N_A(n, k, b)$.
2. If $\hat{M}(x)$ rejects, then the number of complete b-configuration sequences for $\hat{M}(x)$ with k mistakes is $N_R(n, k, b)$.
3. $N_A(n, k, b) - N_R(n, k, b) = (-1)^k \binom{b+1}{k}$.

Conclusions and Questions

Conclusions and Questions

- A mapping from time-space lower bounds for nondeterminism to analogous lower bounds for modular counting of solutions

Conclusions and Questions

- A mapping from time-space lower bounds for nondeterminism to analogous lower bounds for modular counting of solutions
- Relies on properties of alternation-trading proofs and determinism

Conclusions and Questions

- A mapping from time-space lower bounds for nondeterminism to analogous lower bounds for modular counting of solutions
- Relies on properties of alternation-trading proofs and determinism
- Can we prove better time lower bounds for MAJORITY SAT?

  (It's PP*-complete...)*
- How far can alternation-trading proofs go?

Advertisement for Upcoming Work

Advertisement for Upcoming Work

- **Automated Time Lower Bounds:**

  Formalization of alternation-trading proofs

  $\Longrightarrow$ Implementation of a theorem prover

  Proofs found by a combo of exhaustive search and linear programming

Advertisement for Upcoming Work

- **Automated Time Lower Bounds:**

  Formalization of alternation-trading proofs

  $\Longrightarrow$ Implementation of a theorem prover

  Proofs found by a combo of exhaustive search and linear programming

- Experiments suggest that our $\Omega(n^{2\cos(\pi/7)})$ time lower bound for SAT is the best possible with the current tools(!)

Advertisement for Upcoming Work

- **Automated Time Lower Bounds:**

  Formalization of alternation-trading proofs

  $\Longrightarrow$ Implementation of a theorem prover

  Proofs found by a combo of exhaustive search and linear programming

- Experiments suggest that our $\Omega(n^{2\cos(\pi/7)})$ time lower bound for SAT is the best possible with the current tools(!)

If this is correct, then some REALLY NEW IDEAS will be required

to make further progress on time-space lower bounds

Thank you!