MCSP is Hard for Read-Once Nondeterministic Branching Programs

Ludmila Glinskih

Boston University

Artur Riazanov

EPFL

LATIN 2022
Guanajuato, Mexico, November 8

Outline

- Minimum Circuit Size Problem
- Branching Programs
- Our result: every 1-NBP computing MCSP has superpolynomial size
- Technique

Minimum Circuit Size Problem

Input:

- truth table of a Boolean function $f: \{0,1\}^n \to \{0,1\}$

1	0	0	1	0	1	1	0	…	1

Truth table of f of length $N = 2^n$

Minimum Circuit Size Problem

Input:

- truth table of a Boolean function $f: \{0, 1\}^n \to \{0, 1\}$
- size parameter s

1	0	0	1	0	1	1	0	…	1

Truth table of f of length $N = 2^n$

Minimum Circuit Size Problem

Input:

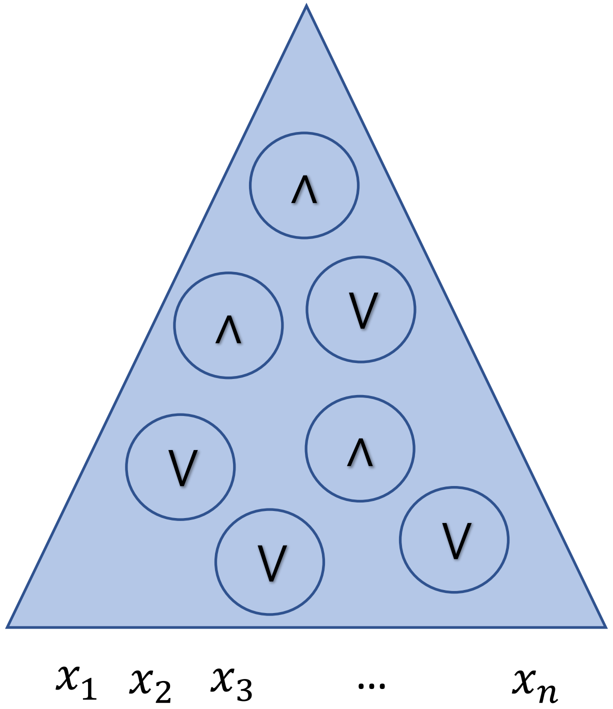
- truth table of a Boolean function $f: \{0,1\}^n \to \{0,1\}$
- size parameter s

1	0	0	1	0	1	1	0	…	1

Truth table of f of length $N = 2^n$

Output:

yes, if f can be computed by a circuit of size at most s



Hardness of MCSP

- MCSP is in NP
  Guess a circuit and check, whether it computes f or not

Hardness of MCSP

- MCSP is in NP
  Guess a circuit and check, whether it computes f or not

- MCSP $\in P \Rightarrow$ no strong PRGs [Razborov, Rudich, 1994]

Hardness of MCSP

- MCSP is in NP
  Guess a circuit and check, whether it computes f or not

- MCSP $\in P \Rightarrow$ no strong PRGs [Razborov, Rudich, 1994]

- MCSP is NP-complete $\Rightarrow EXP \neq ZPP$ [Murray, Williams, 2015]

Hardness of MCSP

- MCSP is in NP
  Guess a circuit and check, whether it computes f or not

- MCSP $\in P \Rightarrow$ no strong PRGs [Razborov, Rudich, 1994]

- MCSP is NP-complete $\Rightarrow EXP \neq ZPP$ [Murray, Williams, 2015]

- Complexity of MCSP in restricted classes is important too:
  If MCSP cannot be computed by
    - a branching program of size $N^{2.01}$
    - formula of size $N^{3.01}$
    - circuit of size $N^{1.01}$

  Then NP $\not\subset$ C-SIZE[n^k] for all k [Chen, Jin, Williams, 2019]

MCSP is hard in certain computational models

In multiple computational models MCSP was shown to be hard

MCSP is hard in certain computational models

In multiple computational models MCSP was shown to be hard

- AC^0(MCSP)= $2^{\Omega(N^{\frac{1}{d}})}$ [Cheraghchi, Kabanets, Lu, Myrisiotis, 2019]

MCSP is hard in certain computational models

In multiple computational models MCSP was shown to be hard

- AC^0(MCSP)= $2^{\Omega(N^{\frac{1}{d}})}$ [Cheraghchi, Kabanets, Lu, Myrisiotis, 2019]

- AC^0[mod p](MCSP)= $2^{\Omega(N^{\frac{0.49}{d}})}$ [Golovnev, Ilango, Impagliazzo, Kabanets, Kolokolova, Tal, 2019]

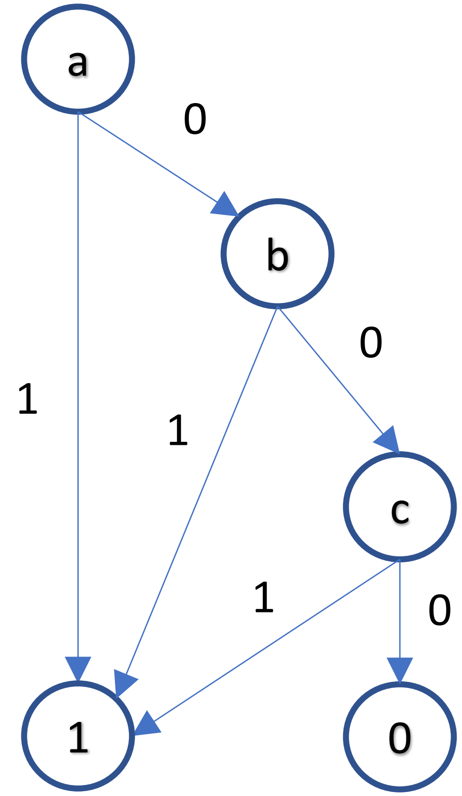
MCSP is hard in certain computational models

In multiple computational models MCSP was shown to be hard

- AC^0(MCSP)= $2^{\Omega(N^{\frac{1}{d}})}$ [Cheraghchi, Kabanets, Lu, Myrisiotis, 2019]
- AC^0[mod p](MCSP)= $2^{\Omega(N^{\frac{0.49}{d}})}$ [Golovnev, Ilango, Impagliazzo, Kabanets, Kolokolova, Tal, 2019]
- 1-coNBP(MCSP)= $2^{\Omega(N)}$ [Cheraghchi, Hirahara, Myrisiotis, Yoshida, 2019]

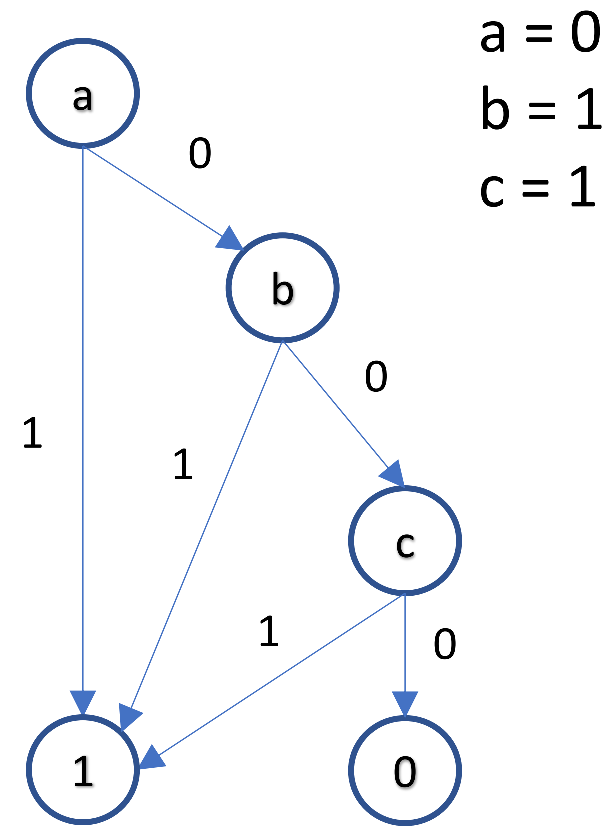
Branching program

- BP is a way to represent Boolean function:
  - directed graph without cycles
  - one source
  - two sinks: labeled with 0 and 1
  - all other vertices labeled with variables
  - values of variables on edges
- Size of a BP is a number of vertices



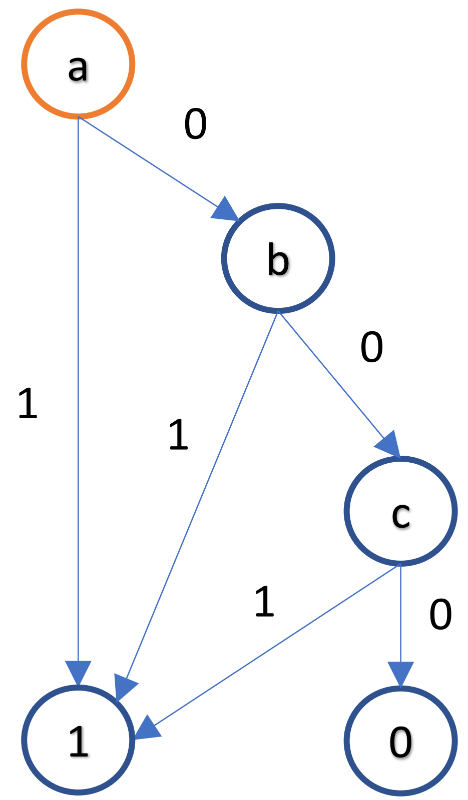
Branching program

- BP is a way to represent Boolean function:
    - directed graph without cycles
    - one source
    - two sinks: labeled with 0 and 1
    - all other vertices labeled with variables
    - values of variables on edges
- Size of a BP is a number of vertices



Branching program

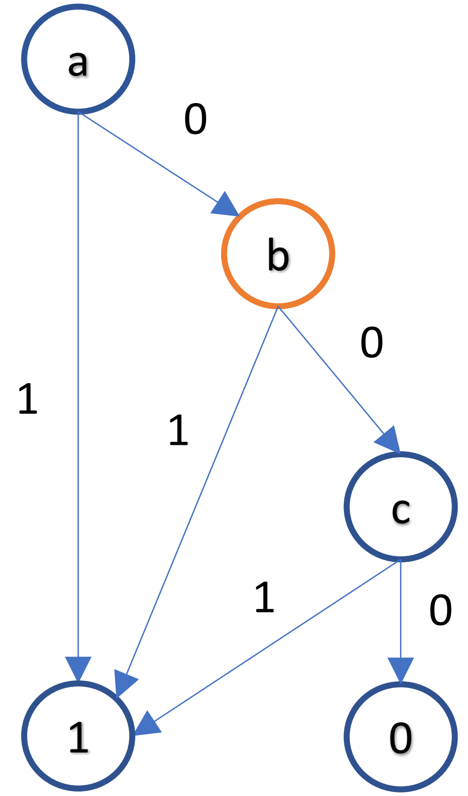
- BP is a way to represent Boolean function:
  - directed graph without cycles
  - one source
  - two sinks: labeled with 0 and 1
  - all other vertices labeled with variables
  - values of variables on edges
- Size of a BP is a number of vertices



a = 0
b = 1
c = 1

Branching program

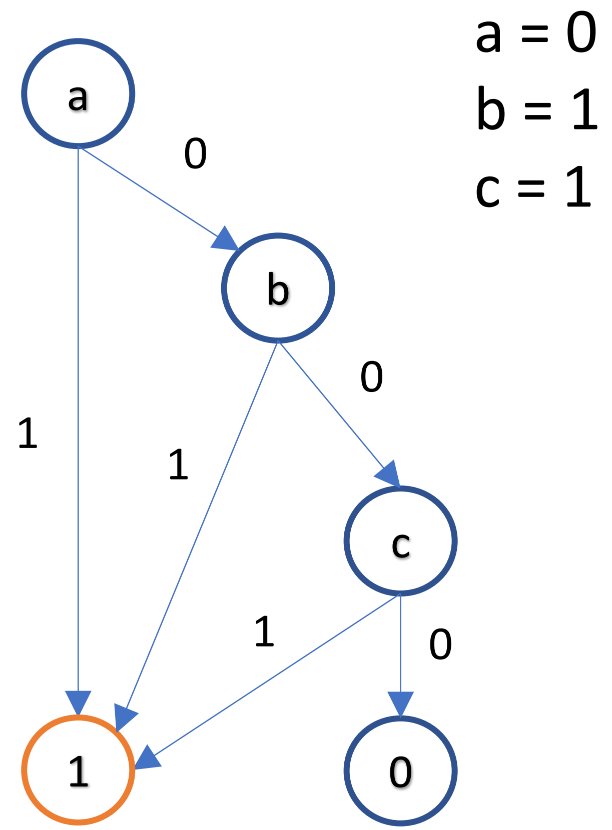
- BP is a way to represent Boolean function:
    - directed graph without cycles
    - one source
    - two sinks: labeled with 0 and 1
    - all other vertices labeled with variables
    - values of variables on edges
- Size of a BP is a number of vertices



a = 0
b = 1
c = 1

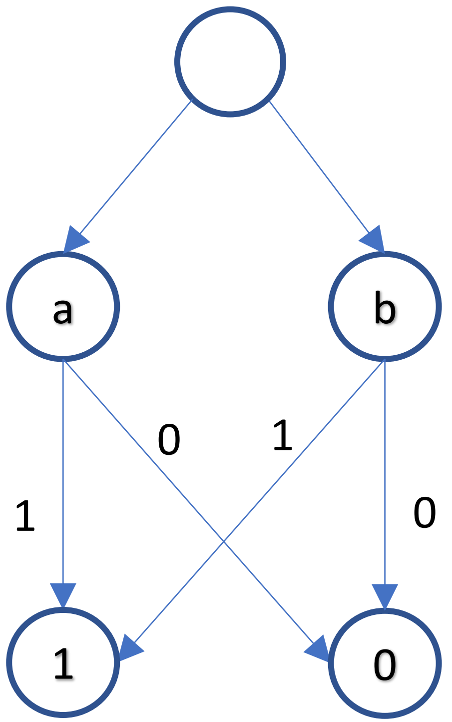
Branching program

- BP is a way to represent Boolean function:
  - directed graph without cycles
  - one source
  - two sinks: labeled with 0 and 1
  - all other vertices labeled with variables
  - values of variables on edges
- Size of a BP is a number of vertices



Non-deterministic branching program

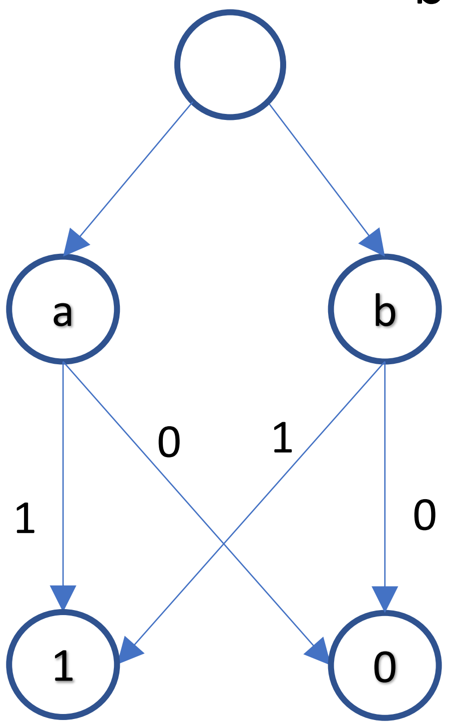
- NBP additionally has non-deterministic nodes:
  - non-deterministic nodes are unlabeled
  - the value equals 1 $\Leftrightarrow$ exists a path to 1-sink



Non-deterministic branching program

- NBP additionally has non-deterministic nodes:
  - non-deterministic nodes are unlabeled
  - the value equals 1 $\Leftrightarrow$ exists a path to 1-sink

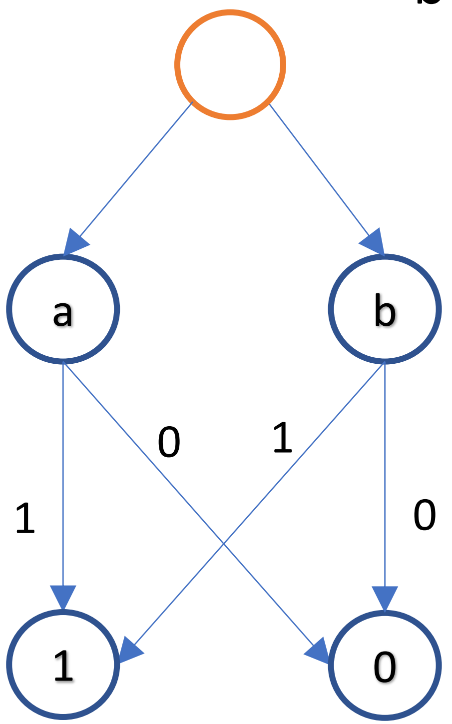
a = 1
b = 0



Non-deterministic branching program

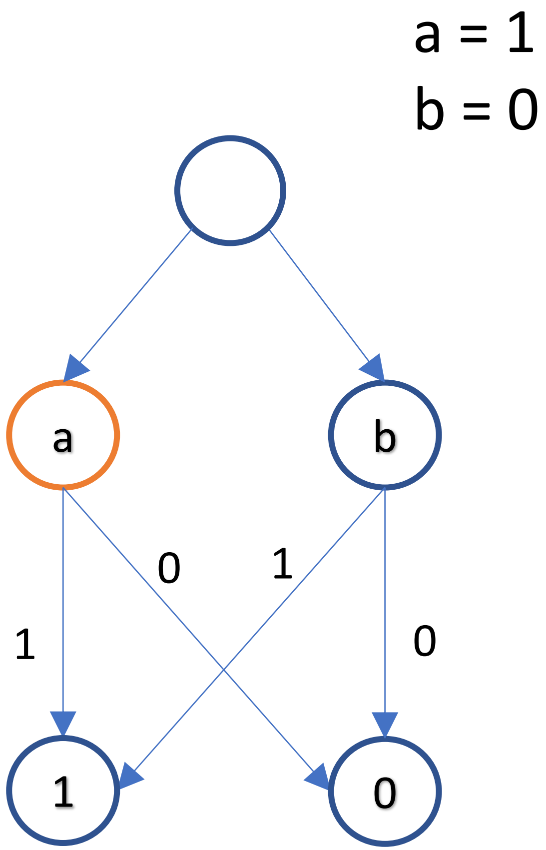
- NBP additionally has non-deterministic nodes:
  - non-deterministic nodes are unlabeled
  - the value equals 1 $\Leftrightarrow$ exists a path to 1-sink

a = 1
b = 0



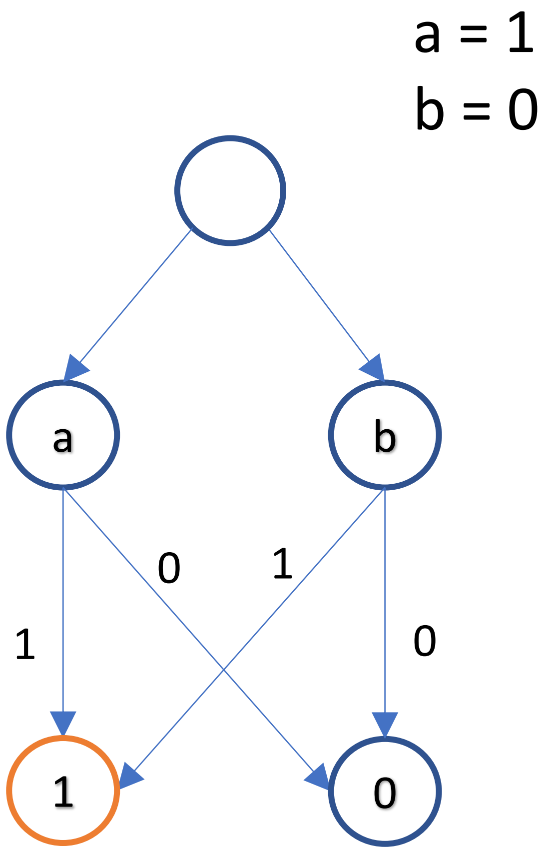
Non-deterministic branching program

- NBP additionally has non-deterministic nodes:
    - non-deterministic nodes are unlabeled
    - the value equals 1 $\Leftrightarrow$ exists a path to 1-sink



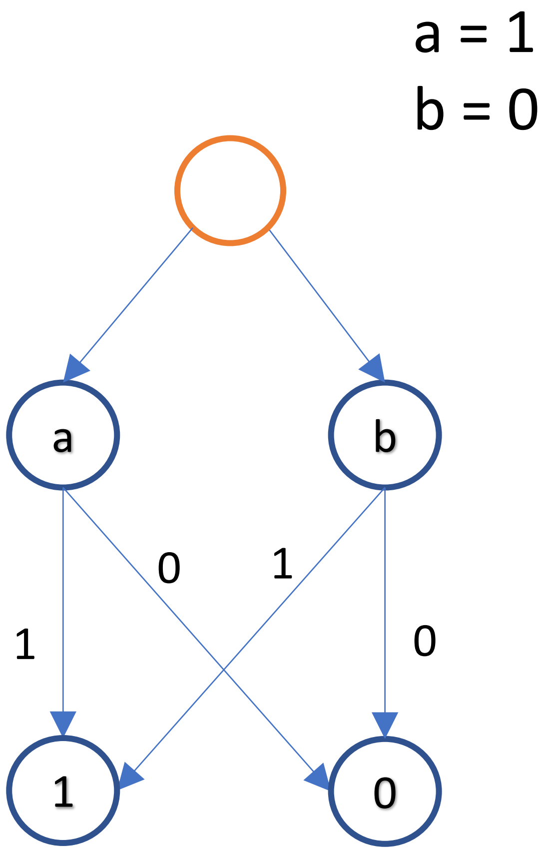
Non-deterministic branching program

- NBP additionally has non-deterministic nodes:
  - non-deterministic nodes are unlabeled
  - the value equals 1 $\Leftrightarrow$ exists a path to 1-sink



Non-deterministic branching program

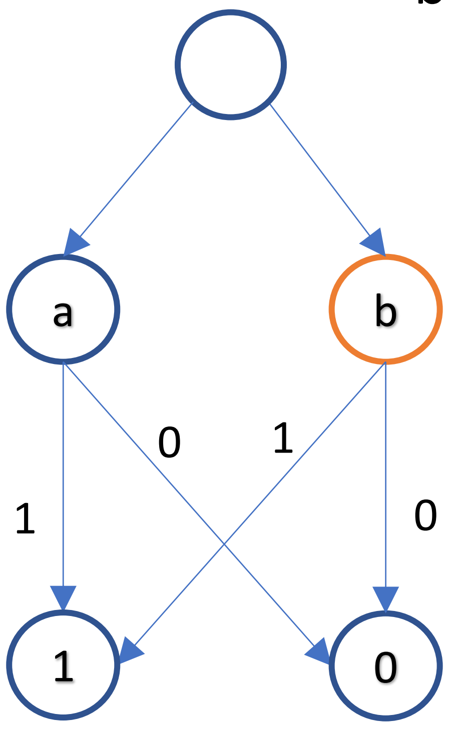
- NBP additionally has non-deterministic nodes:
  - non-deterministic nodes are unlabeled
  - the value equals 1 $\Leftrightarrow$ exists a path to 1-sink



Non-deterministic branching program

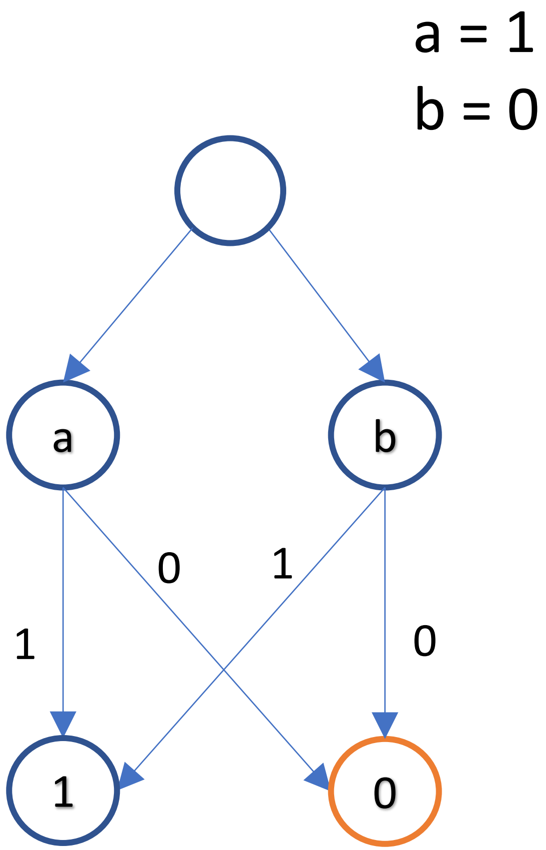
- NBP additionally has non-deterministic nodes:
  - non-deterministic nodes are unlabeled
  - the value equals 1 $\Leftrightarrow$ exists a path to 1-sink

a = 1
b = 0

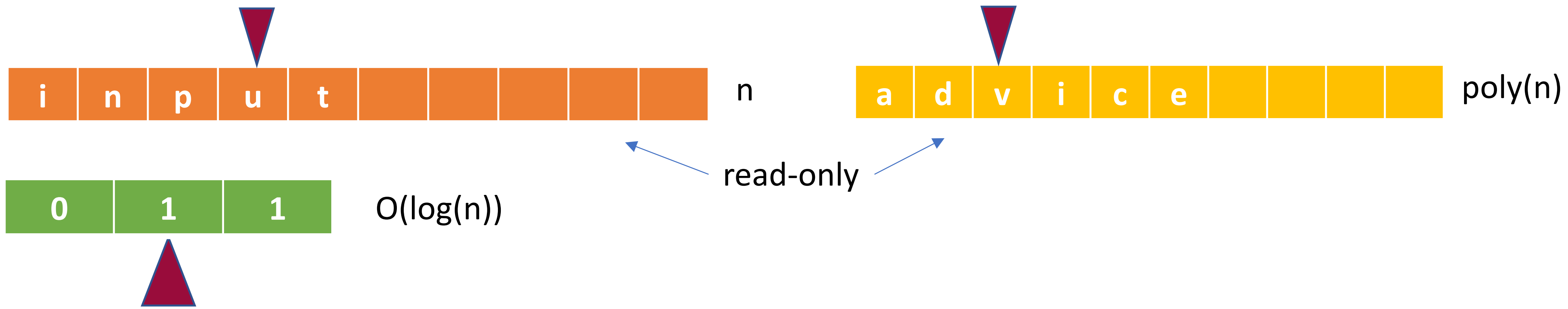


Non-deterministic branching program

- NBP additionally has non-deterministic nodes:
  - non-deterministic nodes are unlabeled
  - the value equals 1 $\Leftrightarrow$ exists a path to 1-sink

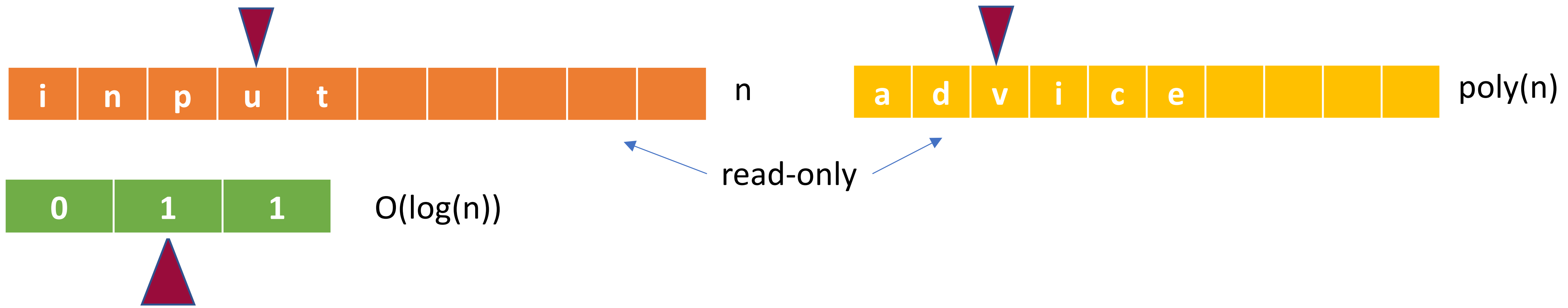


Complexity class with logarithmic space



- BP(f)=poly $\Leftrightarrow$ f is in L/poly

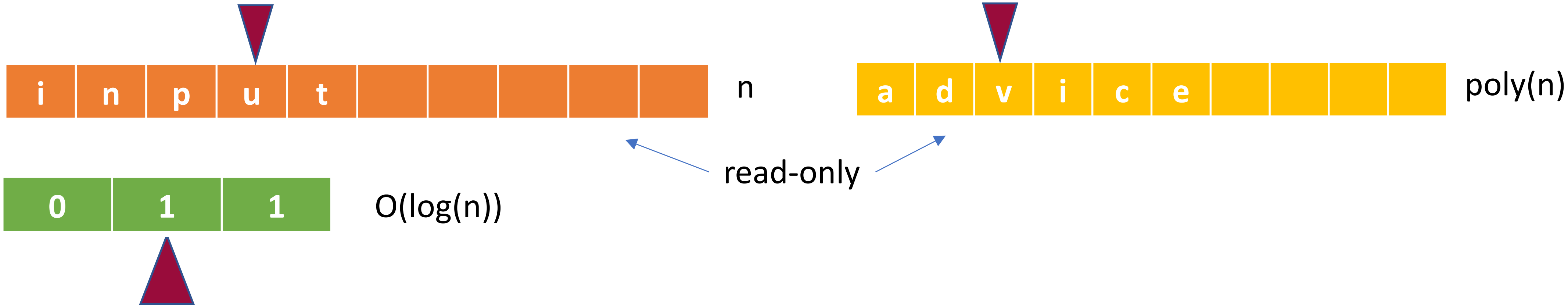
Complexity class with logarithmic space



- BP(f)=poly $\Leftrightarrow$ f is in L/poly

BP(f) is a BP complexity of f

Complexity class with logarithmic space



- BP(f)=poly $\Leftrightarrow$ f is in L/poly

BP(f) is a BP complexity of f

- NBP corresponds to NL/poly

Best lower bounds for branching programs

- At least a $1 - \frac{1}{2^n}$ fraction of functions require BP size $\frac{2^n}{4n}$

Best lower bounds for branching programs

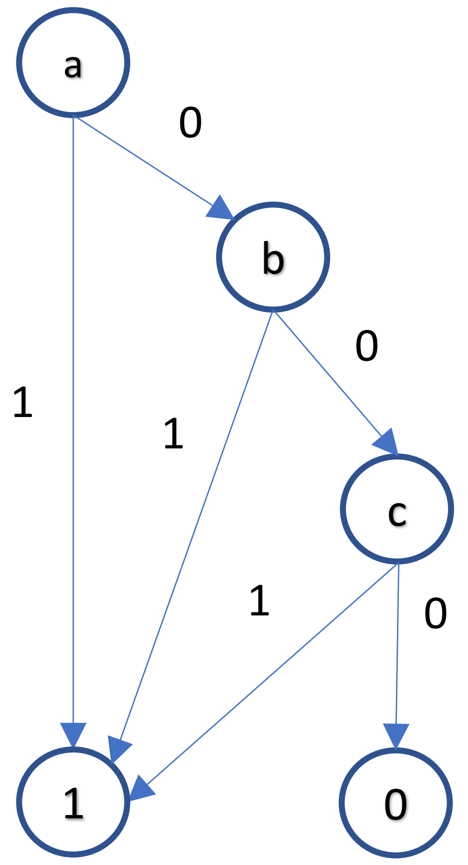
- At least a $1-\frac{1}{2^n}$ fraction of functions require BP size $\frac{2^n}{4n}$
- The best lower bound: BP(ED)=$\Omega\left(\frac{n^2}{\log^2 n}\right)$ [Nechiporuk, 1966]

Best lower bounds for branching programs

- At least a $1-\frac{1}{2^n}$ fraction of functions require BP size $\frac{2^n}{4n}$
- The best lower bound: BP(ED)=$\Omega\left(\frac{n^2}{\log^2 n}\right)$ [Nechiporuk, 1966]
- Recent results:
  - BP(MCSP)=$\widetilde{\Omega}(N^2)$ [Cheraghchi, Kabanets, Lu, Myrisiotis, 2019]
  - Barrier on proving better than $\widetilde{\Omega}(N^2)$ for MCSP [Chen, Jin, Williams, 2019]

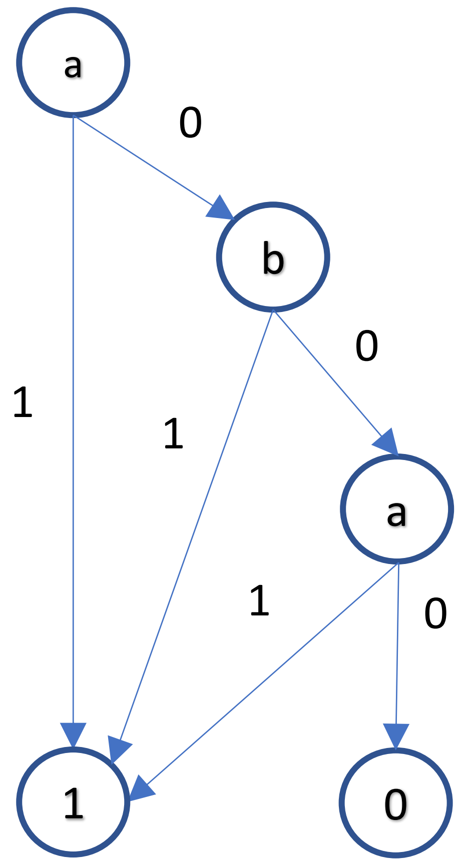
Read-once branching programs

1-BP (1-NBP) if for every path every variable occurs no more than 1 time



Read-once branching programs

1-BP (1-NBP) if for every path every variable occurs no more than 1 time



Known lower bounds for 1-NBPs

- 1-NBP(CLIQUE_ONLY) = $2^{\Omega(\sqrt{n})}$ [Borodin, Razborov, Smolensky, 1993]

Known lower bounds for 1-NBPs

- 1-NBP(CLIQUE_ONLY) = $2^{\Omega\left(\sqrt{n}\right)}$ [Borodin, Razborov, Smolensky, 1993]
- 1-NBP($\bigoplus_{\Delta}$)= $2^{\Omega(n)}$ [Duris, Hromkovic, Jukna, Sauerhoff, Schnitger, 2004]
  - $\bigoplus_{\Delta}$ parity of triangles in a graph

Known lower bounds for 1-NBPs

- 1-NBP(CLIQUE_ONLY) = $2^{\Omega(\sqrt{n})}$ [Borodin, Razborov, Smolensky, 1993]

- 1-NBP($\oplus_\Delta$)= $2^{\Omega(n)}$ [Duris, Hromkovic, Jukna, Sauerhoff, Schnitger, 2004]
  - $\oplus_\Delta$ parity of triangles in a graph

- 1-NBP(coMCSP) = $2^{\Omega(N)}$ [Cheraghchi, Kabanets, Lu, Myrisiotis, 2019]

Known lower bounds for 1-NBPs

- 1-NBP(CLIQUE_ONLY) = $2^{\Omega(\sqrt{n})}$ [Borodin, Razborov, Smolensky, 1993]
- 1-NBP($\oplus_\Delta$)= $2^{\Omega(n)}$ [Duris, Hromkovic, Jukna, Sauerhoff, Schnitger, 2004]
  - $\oplus_\Delta$ parity of triangles in a graph
- 1-NBP(coMCSP) = $2^{\Omega(N)}$ [Cheraghchi, Kabanets, Lu, Myrisiotis, 2019]

MCSP naturally a nondeterministic problem, so it is harder to prove a lower bound against NBP

Main result

Theorem: size of 1-NBP computing MCSP is $N^{\Omega(\log\log N)}$

Main result

This result is tight for MCSP with linear size parameter

Theorem: size of 1-NBP computing MCSP is $N^{\Omega(\log\log N)}$

Main result

This result is tight for MCSP with linear size parameter

Theorem: size of 1-NBP computing MCSP is $N^{\Omega(\log\log N)}$

Theorem [Ilango'20]: assuming Exponential Time Hypothesis every Turing machine computing MCSP* requires time $N^{\Omega(\log\log N)}$

Main result

This result is tight for MCSP with linear size parameter

Theorem: size of 1-NBP computing MCSP is $N^{\Omega(\log\log N)}$

Theorem [Ilango'20]: assuming Exponential Time Hypothesis every Turing machine computing MCSP* requires time $N^{\Omega(\log\log N)}$

(n x n)-Bipartite Independent Set Problem

Main result

This result is tight for MCSP with linear size parameter

Theorem: size of 1-NBP computing MCSP is $N^{\Omega(\log\log N)}$

Theorem [Ilango'20]: assuming Exponential Time Hypothesis every Turing machine computing MCSP* requires time $N^{\Omega(\log\log N)}$

(n x n)-Bipartite Independent Set Problem

ETH-hard

Main result

This result is tight for MCSP with linear size parameter

Theorem: size of 1-NBP computing MCSP is $N^{\Omega(\log\log N)}$

Theorem [Ilango'20]: assuming Exponential Time Hypothesis every Turing machine computing MCSP* requires time $N^{\Omega(\log\log N)}$

Main result

This result is tight for MCSP with linear size parameter

Theorem: size of 1-NBP computing MCSP is $N^{\Omega(\log\log N)}$

Theorem [Ilango'20]: assuming Exponential Time Hypothesis every Turing machine computing MCSP* requires time $N^{\Omega(\log\log N)}$

In MCSP* input is a truth table of a partial function

(n x n)-Bipartite Independent Set Problem $\longrightarrow$ MCSP*

ETH-hard

Main result

This result is tight for MCSP with linear size parameter

Theorem: size of 1-NBP computing MCSP is $N^{\Omega(\log\log N)}$

Theorem [Ilango'20]: assuming Exponential Time Hypothesis every Turing machine computing MCSP* requires time $N^{\Omega(\log\log N)}$

In MCSP* input is a truth table of a partial function

(n x n)-Bipartite Independent Set Problem

ETH-hard

Exp-time reduction

MCSP*

Main result

This result is tight for MCSP with linear size parameter

Theorem: size of 1-NBP computing MCSP is $N^{\Omega(\log\log N)}$

Theorem [Ilango'20]: assuming Exponential Time Hypothesis every Turing machine computing MCSP* requires time $N^{\Omega(\log\log N)}$

In MCSP* input is a truth table of a partial function

(n x n)-Bipartite Independent Set Problem

Exp-time reduction

MCSP*

ETH-hard

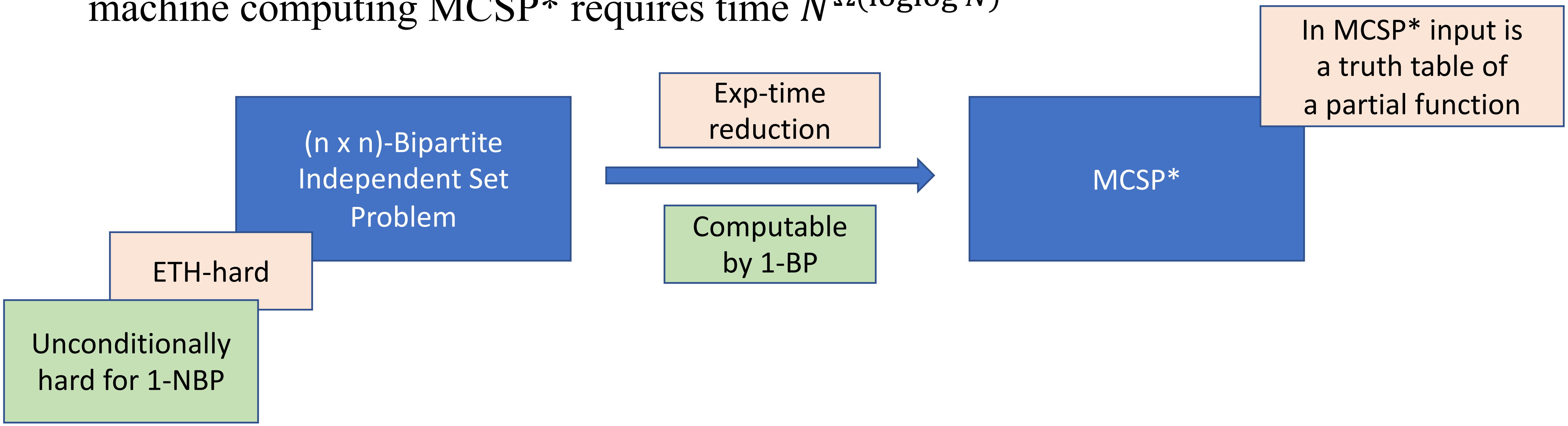
Unconditionally hard for 1-NBP

Main result

This result is tight for MCSP with linear size parameter

Theorem: size of 1-NBP computing MCSP is $N^{\Omega(\text{loglog } N)}$

Theorem [Ilango'20]: assuming Exponential Time Hypothesis every Turing machine computing MCSP* requires time $N^{\Omega(\text{loglog } N)}$



Main result

This result is tight for MCSP with linear size parameter

Theorem: size of 1-NBP computing MCSP is $N^{\Omega(\log\log N)}$

Theorem [Ilango'20]: assuming Exponential Time Hypothesis every Turing machine computing MCSP* requires time $N^{\Omega(\log\log N)}$

In MCSP* input is a truth table of a partial function

(n x n)-Bipartite Independent Set Problem

ETH-hard

Unconditionally hard for 1-NBP

Exp-time reduction

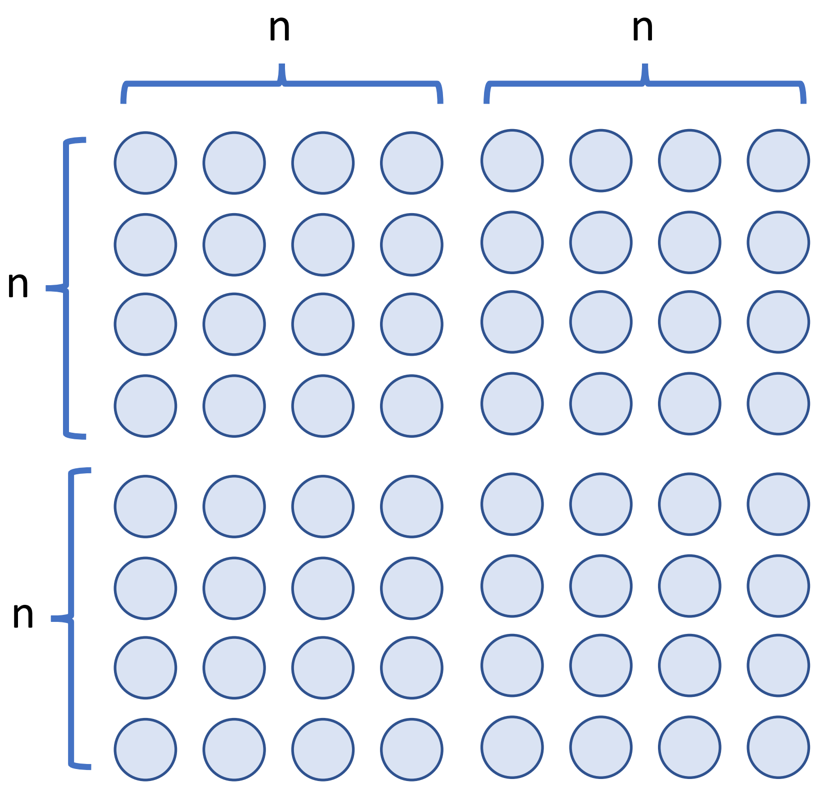
Computable by 1-BP

MCSP*

MCSP

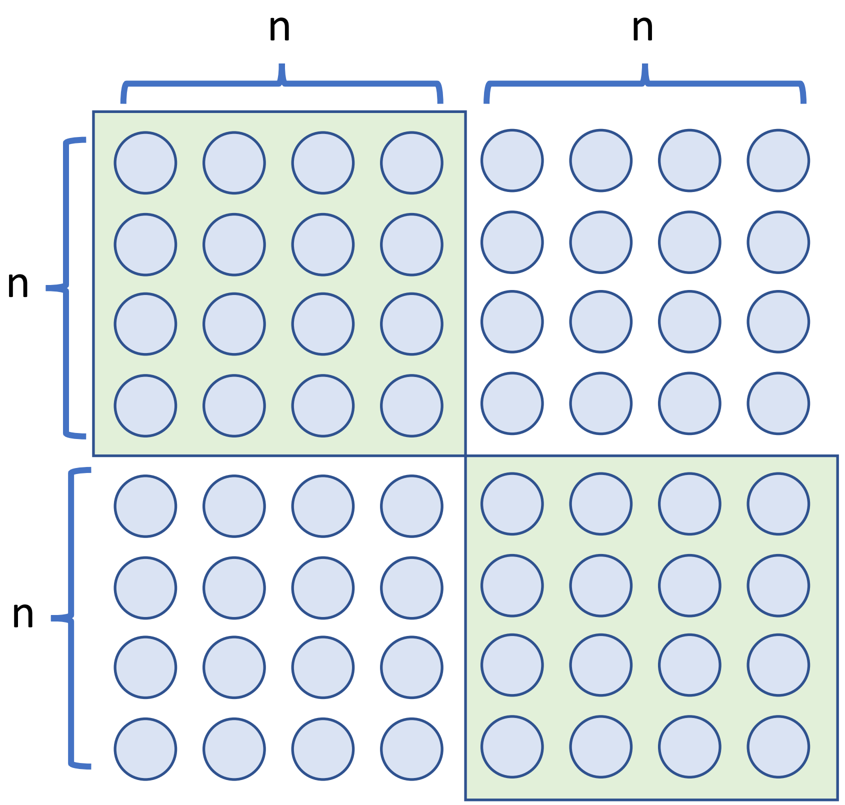
Have the same 1-NBP complexity

(n x n)-Bipartite Permutation Independent Set (BPIS)



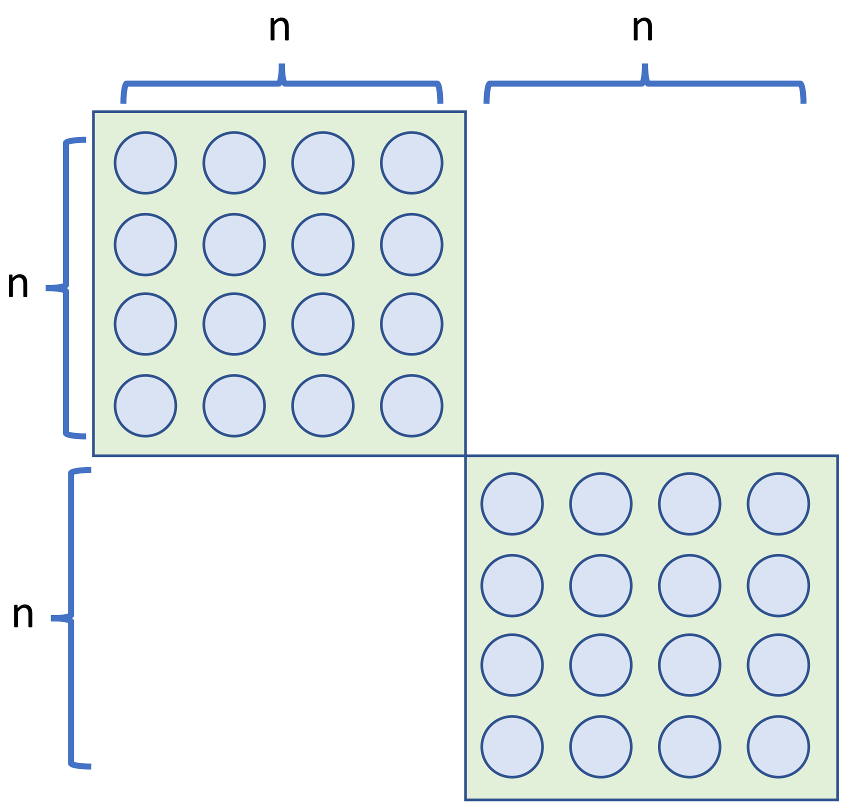
- Graph with 2n x 2n vertices,
- Edges exist only between vertices from two quadrants
- Need to find exactly one vertex from every row, and exactly one vertex from every column, such that
  - These vertices are from the two quadrants
  - These vertices form independent set

(n x n)-Bipartite Permutation Independent Set (BPIS)



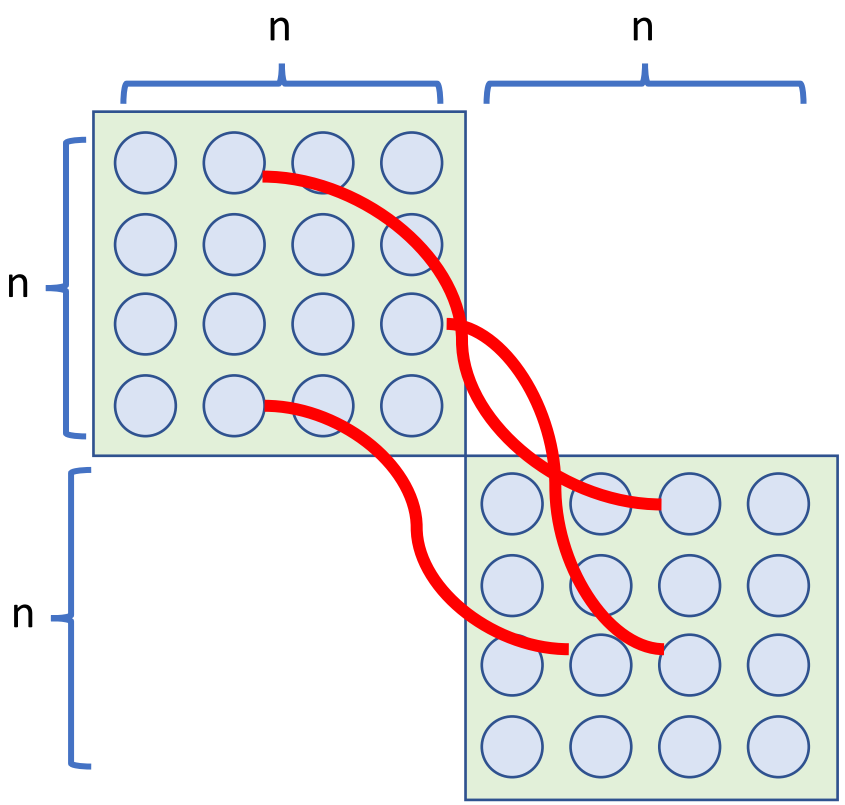
- Graph with 2n x 2n vertices,
- Edges exist only between vertices from two quadrants
- Need to find exactly one vertex from every row, and exactly one vertex from every column, such that
  - These vertices are from the two quadrants
  - These vertices form independent set

(n x n)-Bipartite Permutation Independent Set (BPIS)



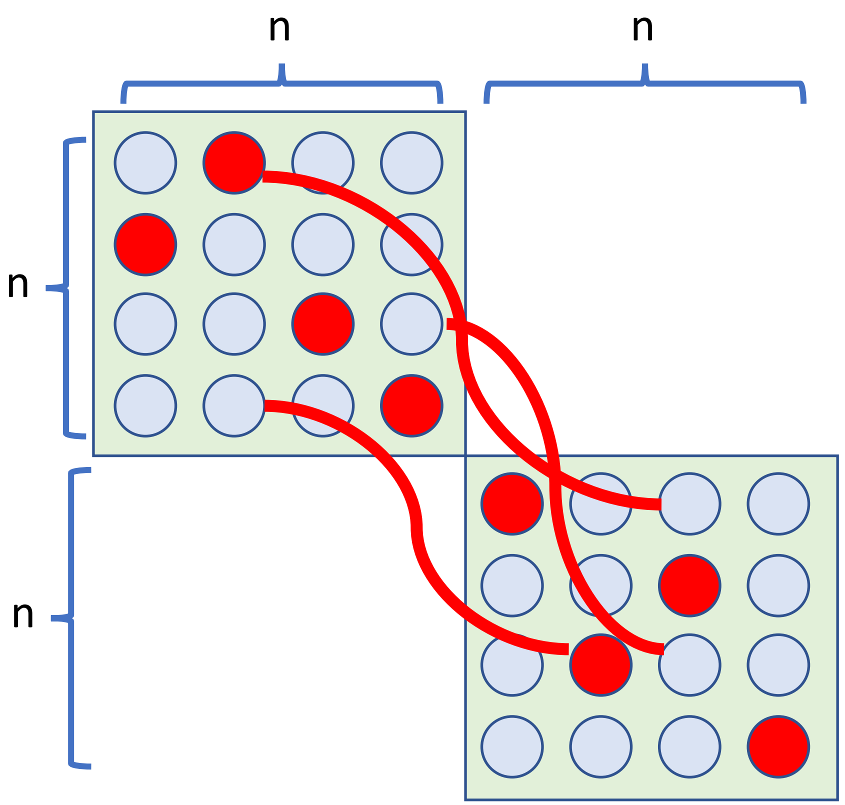
- Graph with 2n x 2n vertices,
- Edges exist only between vertices from two quadrants
- Need to find exactly one vertex from every row, and exactly one vertex from every column, such that
  - These vertices are from the two quadrants
  - These vertices form independent set

(n x n)-Bipartite Permutation Independent Set (BPIS)



- Graph with 2n x 2n vertices,
- Edges exist only between vertices from two quadrants
- Need to find exactly one vertex from every row, and exactly one vertex from every column, such that
  - These vertices are from the two quadrants
  - These vertices form independent set

(n x n)-Bipartite Permutation Independent Set (BPIS)



- Graph with 2n x 2n vertices,
- Edges exist only between vertices from two quadrants
- Need to find exactly one vertex from every row, and exactly one vertex from every column, such that
  - These vertices are from the two quadrants
  - These vertices form independent set

(n x n)-BPIS is hard for 1-NBP

Lemma: size of 1-NBP computing an (n x n)-BPIS is $2^{\Omega(n \log n)}$

(n x n)-BPIS is hard for 1-NBP

Lemma: size of 1-NBP computing an (n x n)-BPIS is $2^{\Omega(n \log n)}$

Idea of the proof:

- Show that the minimum 1-NBP for Bipartite Permutation Independent Set has the same size as the minimum 1-NBP for Bipartite Permutation Clique

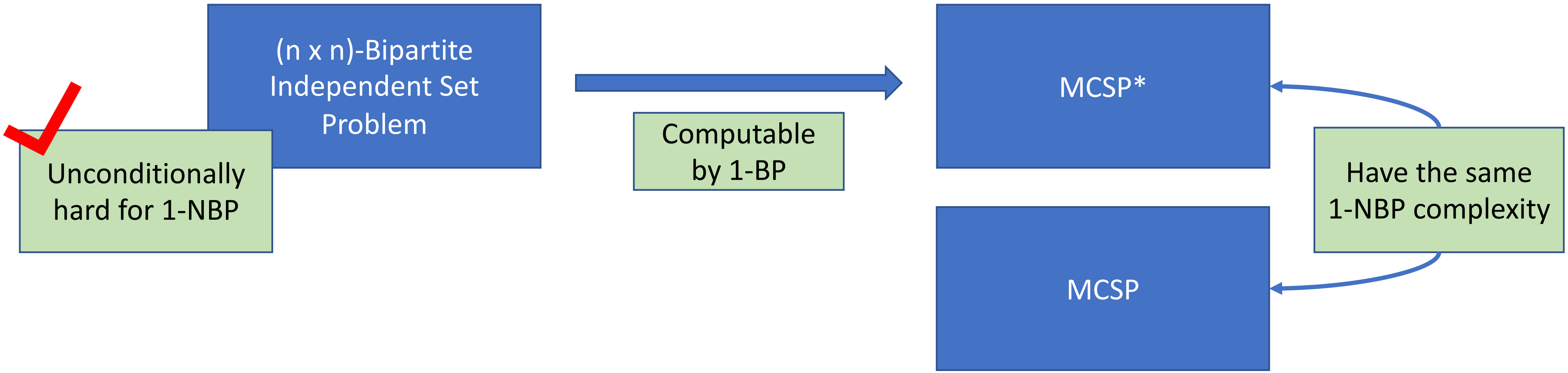
(n x n)-BPIS is hard for 1-NBP

Lemma: size of 1-NBP computing an (n x n)-BPIS is $2^{\Omega(n \log n)}$

Idea of the proof:

- Show that the minimum 1-NBP for Bipartite Permutation Independent Set has the same size as the minimum 1-NBP for Bipartite Permutation Clique
- Adapt the proof of the lower bound on 1-NBP for CLIQUE_ONLY to get a lower bound on BPC

Progress so far



1-NBP for MCSP* can be transformed to 1-NBP for BPIS

$$\gamma(x,y,z) = \begin{cases} \bigvee_{i\in[2n]}(y_i \wedge z_i) & \text{, if } x = 0^{2n} \\ \bigvee_{i\in[2n]} z_i & \text{, if } x = 1^{2n} \\ \bigvee_{i\in[2n]}(x_i \vee y_i) & \text{, if } z = 1^{2n} \\ 0 & \text{, if } z = 0^{2n} \\ \mathsf{OR}_n(x_1,\dots,x_n) & \text{, if } z = 1^n0^n \text{ and } y = 0^{2n} \\ \mathsf{OR}_n(x_{n+1},\dots,x_{2n}) & \text{, if } z = 0^n1^n \text{ and } y = 0^{2n} \\ 1 & \text{, if } \exists\ ((j,k),(j',k')) \in E \text{ such that } (x,y,z) = (\overline{e_k e_{k'}}, 0^{2n}, e_j e_{j'}) \\ \star & \text{, otherwise} \end{cases}$$

1-NBP for MCSP* can be transformed to 1-NBP for BPIS

$$\gamma(x,y,z) = \begin{cases} \bigvee_{i\in[2n]}(y_i \wedge z_i) & \text{, if } x = 0^{2n} \\ \bigvee_{i\in[2n]} z_i & \text{, if } x = 1^{2n} \\ \bigvee_{i\in[2n]}(x_i \vee y_i) & \text{, if } z = 1^{2n} \\ 0 & \text{, if } z = 0^{2n} \\ \mathsf{OR}_n(x_1,\ldots,x_n) & \text{, if } z = 1^n0^n \text{ and } y = 0^{2n} \\ \mathsf{OR}_n(x_{n+1},\ldots,x_{2n}) & \text{, if } z = 0^n1^n \text{ and } y = 0^{2n} \\ \boxed{1} & \text{,if } \exists\, ((j,k),(j',k')) \in E \text{ such that } (x,y,z) = (\overline{e_k e_{k'}}, 0^{2n}, e_j e_{j'}) \\ \star & \text{, otherwise} \end{cases}$$

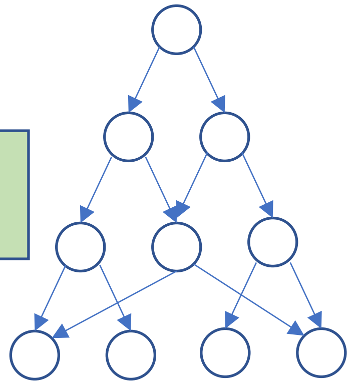
Only these bits of the truth table depend on the input bits of BPIS

1-NBP for MCSP* can be transformed to 1-NBP for BPIS

$$\gamma(x,y,z) = \begin{cases} \bigvee_{i\in[2n]}(y_i \wedge z_i) & \text{, if } x = 0^{2n} \\ \bigvee_{i\in[2n]} z_i & \text{, if } x = 1^{2n} \\ \bigvee_{i\in[2n]}(x_i \vee y_i) & \text{, if } z = 1^{2n} \\ 0 & \text{, if } z = 0^{2n} \\ \mathsf{OR}_n(x_1,\ldots,x_n) & \text{, if } z = 1^n0^n \text{ and } y = 0^{2n} \\ \mathsf{OR}_n(x_{n+1},\ldots,x_{2n}) & \text{, if } z = 0^n1^n \text{ and } y = 0^{2n} \\ 1 & \text{, if } \exists\, ((j,k),(j',k')) \in E \text{ such that } (x,y,z) = (\overline{e_k e_{k'}}, 0^{2n}, e_j e_{j'}) \\ \star & \text{, otherwise} \end{cases}$$

Only these bits of the truth table depend on the input bits of BPIS

1-NBP for MCSP*

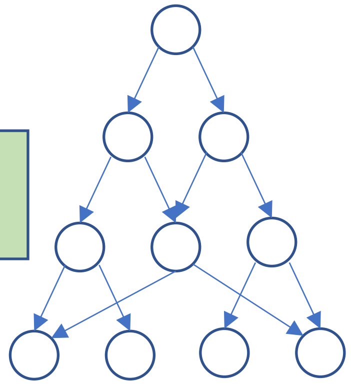


1-NBP for MCSP* can be transformed to 1-NBP for BPIS

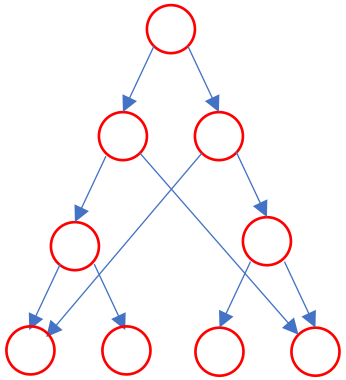
$$\gamma(x,y,z) = \begin{cases} \bigvee_{i\in[2n]}(y_i \wedge z_i) & \text{, if } x = 0^{2n} \\ \bigvee_{i\in[2n]} z_i & \text{, if } x = 1^{2n} \\ \bigvee_{i\in[2n]}(x_i \vee y_i) & \text{, if } z = 1^{2n} \\ 0 & \text{, if } z = 0^{2n} \\ \mathsf{OR}_n(x_1,\dots,x_n) & \text{, if } z = 1^n0^n \text{ and } y = 0^{2n} \\ \mathsf{OR}_n(x_{n+1},\dots,x_{2n}) & \text{, if } z = 0^n1^n \text{ and } y = 0^{2n} \\ 1 & \text{, if } \exists\, ((j,k),(j',k')) \in E \text{ such that } (x,y,z) = (\overline{e_k e_{k'}}, 0^{2n}, e_j e_{j'}) \\ \star & \text{, otherwise} \end{cases}$$

Only these bits of the truth table depend on the input bits of BPIS

1-NBP for MCSP*



Substitute bits of the truth table of γ that do not depend on BPIS' input

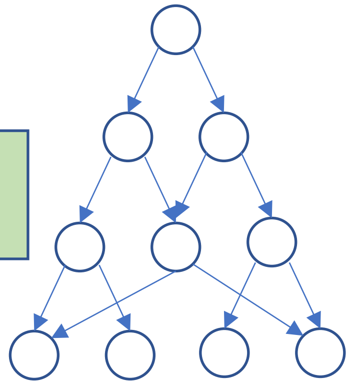


1-NBP for MCSP* can be transformed to 1-NBP for BPIS

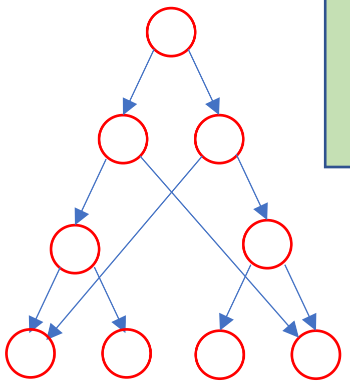
$$\gamma(x,y,z) = \begin{cases} \bigvee_{i\in[2n]}(y_i \wedge z_i) & \text{, if } x = 0^{2n} \\ \bigvee_{i\in[2n]} z_i & \text{, if } x = 1^{2n} \\ \bigvee_{i\in[2n]}(x_i \vee y_i) & \text{, if } z = 1^{2n} \\ 0 & \text{, if } z = 0^{2n} \\ \mathsf{OR}_n(x_1,\ldots,x_n) & \text{, if } z = 1^n0^n \text{ and } y = 0^{2n} \\ \mathsf{OR}_n(x_{n+1},\ldots,x_{2n}) & \text{, if } z = 0^n1^n \text{ and } y = 0^{2n} \\ \boxed{1} & \text{,if } \exists\, ((j,k),(j',k')) \in E \text{ such that } (x,y,z) = (\overline{e_k e_{k'}}, 0^{2n}, e_j e_{j'}) \\ \star & \text{, otherwise} \end{cases}$$

Only these bits of the truth table depend on the input bits of BPIS

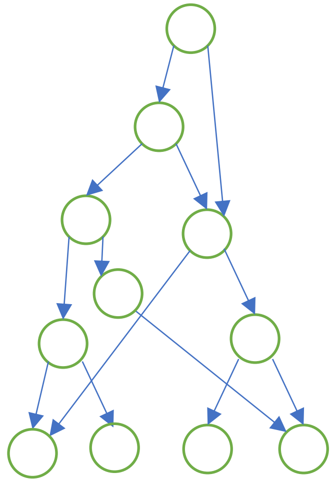
1-NBP for MCSP*



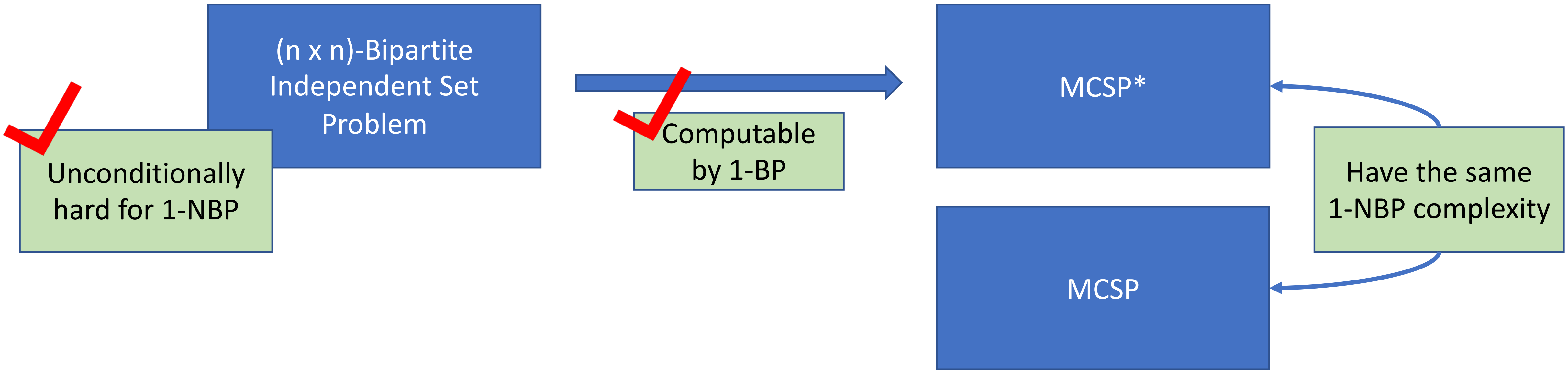
Substitute bits of the truth table of γ that do not depend on BPIS' input



Substitute 1-BPs that computes dependency on the edges of BPIS



Almost finished



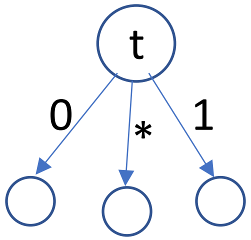
MCSP* and MCSP have the same 1-NBP complexity

Lemma: the size of the minimal 1-NBP computing MCSP* equals the size of the minimal 1–NBP computing MCSP

MCSP* and MCSP have the same 1-NBP complexity

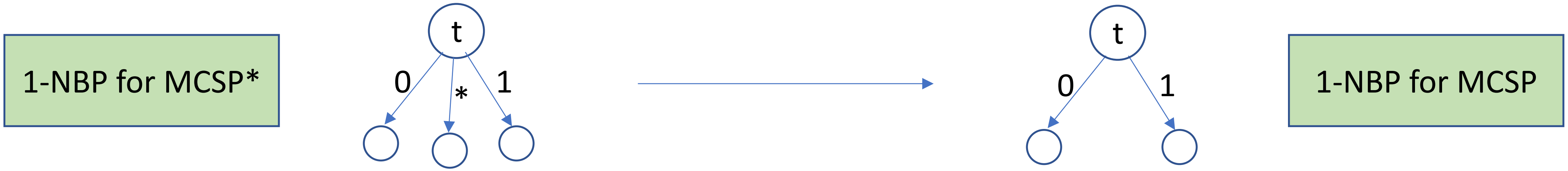
Lemma: the size of the minimal 1-NBP computing MCSP* equals the size of the minimal 1–NBP computing MCSP

1-NBP for MCSP*



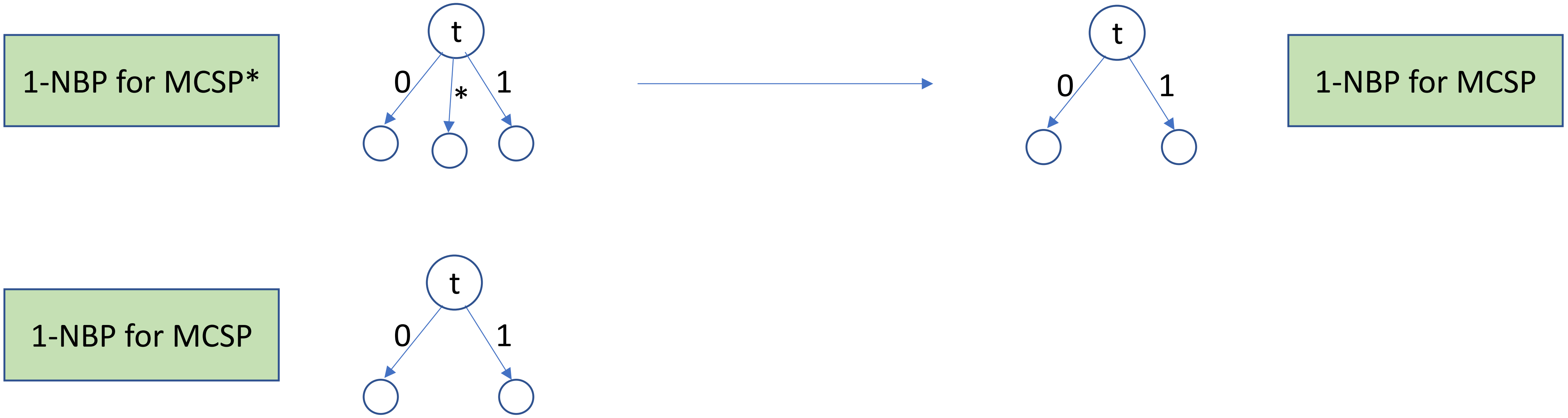
MCSP* and MCSP have the same 1-NBP complexity

Lemma: the size of the minimal 1-NBP computing MCSP* equals the size of the minimal 1–NBP computing MCSP



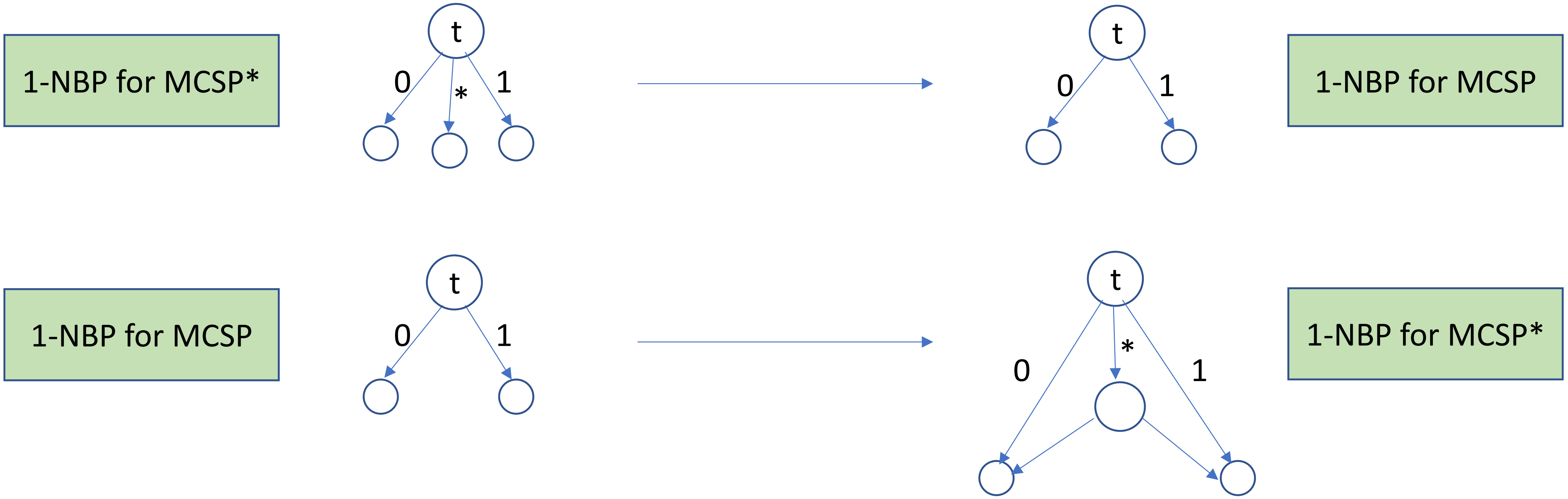
MCSP* and MCSP have the same 1-NBP complexity

Lemma: the size of the minimal 1-NBP computing MCSP* equals the size of the minimal 1–NBP computing MCSP

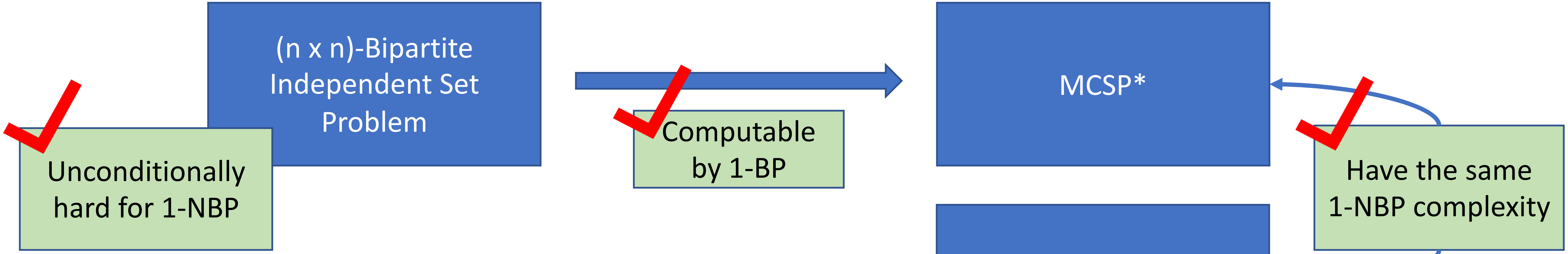


MCSP* and MCSP have the same 1-NBP complexity

Lemma: the size of the minimal 1-NBP computing MCSP* equals the size of the minimal 1–NBP computing MCSP



Putting all together



If MCSP could be computed by a small 1-NBP, then (n x n)-BPIS could be computed by a 1-NBP of even smaller size.
That leads to a contradiction.

Upper bound

Lemma: MCSP on an input of length 2^n with a size parameter s can be computed by a 1-NBP of size $2^n 2^{O(s \log s)}$

Upper bound

Simple guess
and check strategy

Lemma: MCSP on an input of length 2^n with a size parameter s can be computed by a 1-NBP of size $2^n 2^{O(s \log s)}$

Upper bound

Simple guess
and check strategy

Lemma: MCSP on an input of length 2^n with a size parameter s can be computed by a 1-NBP of size $2^n 2^{O(s \log s)}$

Corollary: our $2^{\Omega(n \log n)}$ lower bound is tight for inputs with a linear size parameter

Open questions

- Show tight lower bound for MCSP with higher size parameters
  - The same technique cannot work, as we cannot construct a truth table of a function with higher than linear circuit complexity

Open questions

- Show tight lower bound for MCSP with higher size parameters
  - The same technique cannot work, as we cannot construct a truth table of a function with higher than linear circuit complexity
- Extend this result to other models of computations
  - For any model in which (n x n)-BPIS is hard and the reduction to the truth table is efficiently computable the same size lower bound will hold

Partial Minimum Circuit Size Problem

Input:

- truth table of a partial Boolean function
  $f: \{0,1\}^n \to \{0,1,*\}$
- size parameter s

1	*	*	1	*	1	1	0	…	1

Truth table of f of length $N = 2^n$

Output:

yes, if exists a total function g that is consistent with f and can be computed by a circuit of size at most s