Deterministic identity testing for sum of read-once oblivious arithmetic branching programs

Rohit Gurjar, Arpita Korwar, Nitin Saxena,
IIT Kanpur
Thomas Thierauf
Aalen University

June 18, 2015

Polynomial Identity Testing

- PIT: given a polynomial $P(\boldsymbol{x}) \in \mathbb{F}[x_1, x_2, \ldots, x_n]$, $P(\boldsymbol{x}) = 0$?

Polynomial Identity Testing

- PIT: given a polynomial $P(\boldsymbol{x}) \in \mathbb{F}[x_1, x_2, \ldots, x_n]$, $P(\boldsymbol{x}) = 0$?
- Input Models:
  - Arithmetic Circuits
  - Arithmetic Branching Programs

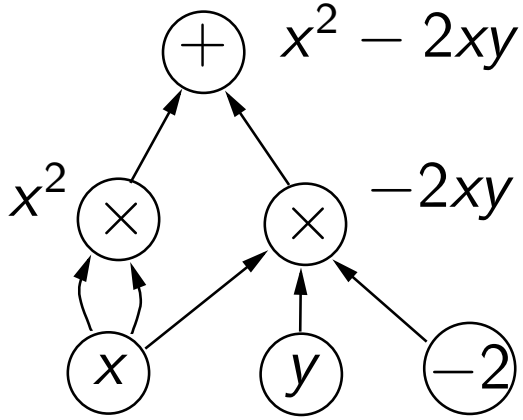


Figure: An Arithmetic circuit

Randomized Test

- Rephrasing the question: Given an arithmetic circuit decide if it computes the zero polynomial.
- Randomized PIT: evaluate $P(\boldsymbol{x})$ at a random point [Demillo and Lipton, 1978, Zippel, 1979, Schwartz, 1980].

Randomized Test

- Rephrasing the question: Given an arithmetic circuit decide if it computes the zero polynomial.
- Randomized PIT: evaluate $P(\boldsymbol{x})$ at a random point [Demillo and Lipton, 1978, Zippel, 1979, Schwartz, 1980].
- There is no efficient deterministic test known.

Randomized Test

- Rephrasing the question: Given an arithmetic circuit decide if it computes the zero polynomial.
- Randomized PIT: evaluate $P(\boldsymbol{x})$ at a random point [Demillo and Lipton, 1978, Zippel, 1979, Schwartz, 1980].
- There is no efficient deterministic test known.
- Two Paradigms:
  - Whitebox: one can see the input circuit.
  - Blackbox: circuit is hidden, only evaluations are allowed.

Randomized Test

- Rephrasing the question: Given an arithmetic circuit decide if it computes the zero polynomial.
- Randomized PIT: evaluate $P(\boldsymbol{x})$ at a random point [Demillo and Lipton, 1978, Zippel, 1979, Schwartz, 1980].
- There is no efficient deterministic test known.
- Two Paradigms:
  - Whitebox: one can see the input circuit.
  - Blackbox: circuit is hidden, only evaluations are allowed.
- Derandomizing PIT has connections with circuit lower bounds [Kabanets and Impagliazzo, 2003, Agrawal, 2005].

Randomized Test

- Rephrasing the question: Given an arithmetic circuit decide if it computes the zero polynomial.
- Randomized PIT: evaluate $P(\boldsymbol{x})$ at a random point [Demillo and Lipton, 1978, Zippel, 1979, Schwartz, 1980].
- There is no efficient deterministic test known.
- Two Paradigms:
  - Whitebox: one can see the input circuit.
  - Blackbox: circuit is hidden, only evaluations are allowed.
- Derandomizing PIT has connections with circuit lower bounds [Kabanets and Impagliazzo, 2003, Agrawal, 2005].
- An efficient test is known only for restricted class of circuits, e.g., Sparse polynomials, set-multilinear circuits, ROABP.

Arithmetic Branching Program

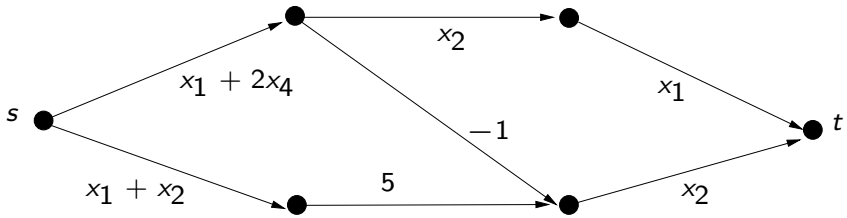


Figure: An Arithmetic branching program.

- ABP: a directed acyclic graph G with a start node and an end node.
- Each edge has a weight from $\mathbb{F}[\boldsymbol{x}]$.

Arithmetic Branching Program

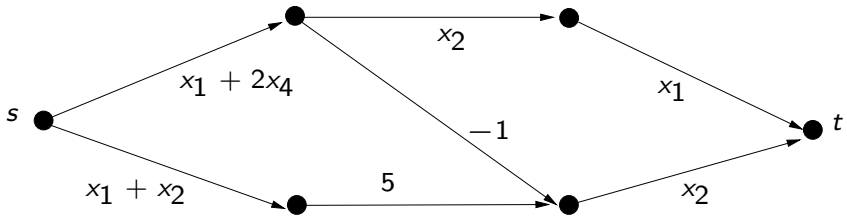


Figure: An Arithmetic branching program.

- ABP: a directed acyclic graph G with a start node and an end node.
- Each edge has a weight from $\mathbb{F}[\boldsymbol{x}]$.

$$C(\boldsymbol{x}) = \sum_{p \in \text{paths}(s,t)} W(p), \text{ where } W(p) = \prod_{e \in p} W(e).$$

Arithmetic Branching Program

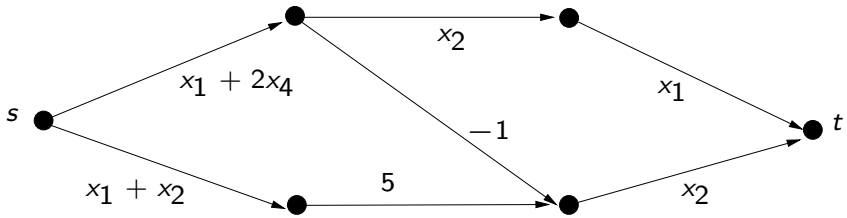


Figure: An Arithmetic branching program.

- ABP: a directed acyclic graph G with a start node and an end node.
- Each edge has a weight from $\mathbb{F}[\boldsymbol{x}]$.

$$C(\boldsymbol{x}) = \sum_{p \in \text{paths}(s,t)} W(p), \text{ where } W(p) = \prod_{e \in p} W(e).$$

- $C(\boldsymbol{x}) = (x_1 + 2x_4)x_2x_1 - (x_1 + 2x_4)x_2 + (x_1 + x_2)5x_2$

Arithmetic Branching Program

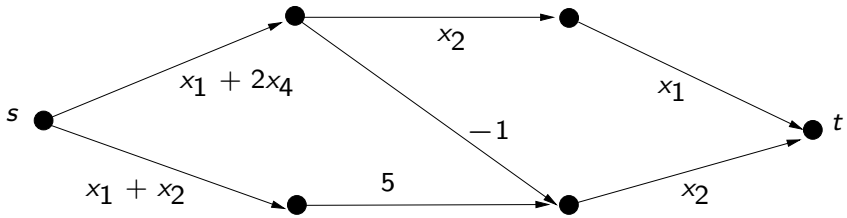


Figure: An Arithmetic branching program.

- ABP: a directed acyclic graph G with a start node and an end node.
- Each edge has a weight from $\mathbb{F}[\boldsymbol{x}]$.

$$C(\boldsymbol{x}) = \sum_{p \in \text{paths}(s,t)} W(p), \text{ where } W(p) = \prod_{e \in p} W(e).$$

- $C(\boldsymbol{x}) = (x_1 + 2x_4)x_2x_1 - (x_1 + 2x_4)x_2 + (x_1 + x_2)5x_2$
- Width: maximum number of nodes in a layer.

Read-once Oblivious ABP

- Any variable occurs in at most one layer.

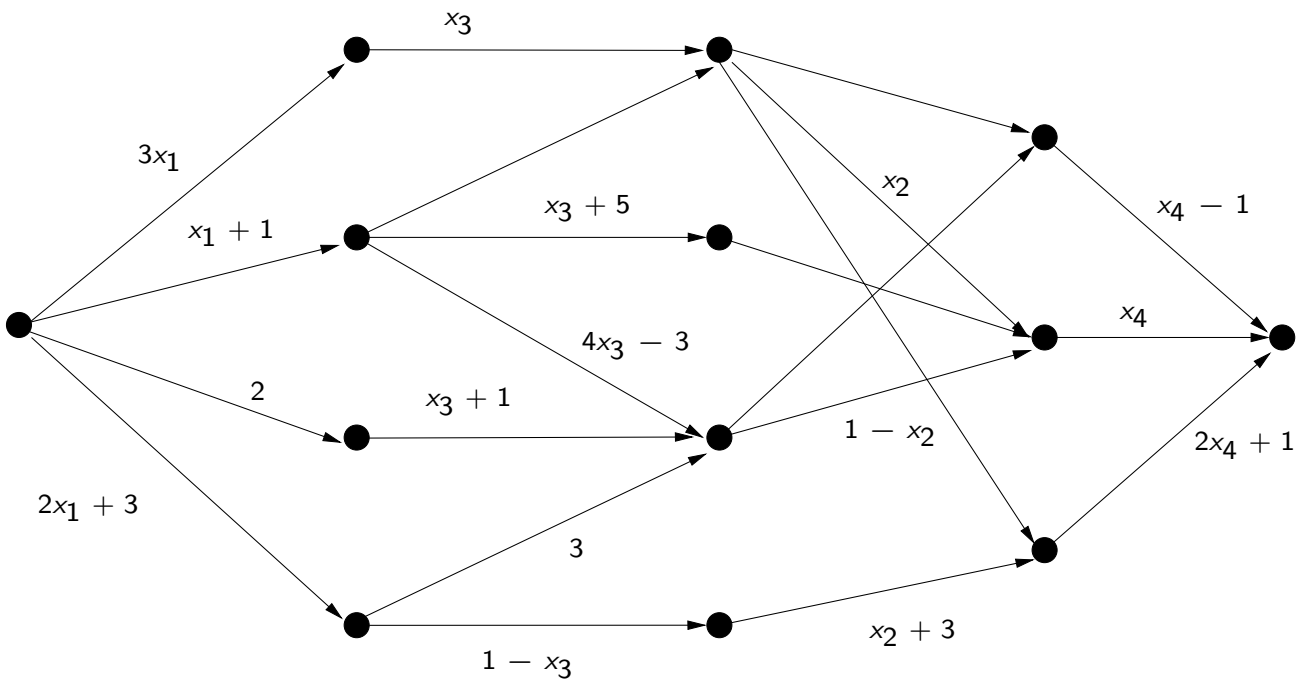


Figure: A Read-once oblivious ABP with variable order (x_1, x_3, x_2, x_4)

Previous Work

- [Raz and Shpilka, 2005] gave a polynomial time whitebox test for ROABP.

Previous Work

- [Raz and Shpilka, 2005] gave a polynomial time whitebox test for ROABP.
- Later, quasi-polynomial time blackbox tests were obtained [Forbes and Shpilka, 2013, Forbes et al., 2014, Agrawal et al., 2014].

Previous Work

- [Raz and Shpilka, 2005] gave a polynomial time whitebox test for ROABP.
- Later, quasi-polynomial time blackbox tests were obtained [Forbes and Shpilka, 2013, Forbes et al., 2014, Agrawal et al., 2014].
- Sum of two ROABPs: we give the first polynomial time whitebox test (different variable orders)

Previous Work

- [Raz and Shpilka, 2005] gave a polynomial time whitebox test for ROABP.
- Later, quasi-polynomial time blackbox tests were obtained [Forbes and Shpilka, 2013, Forbes et al., 2014, Agrawal et al., 2014].
- Sum of two ROABPs: we give the first polynomial time whitebox test (different variable orders)

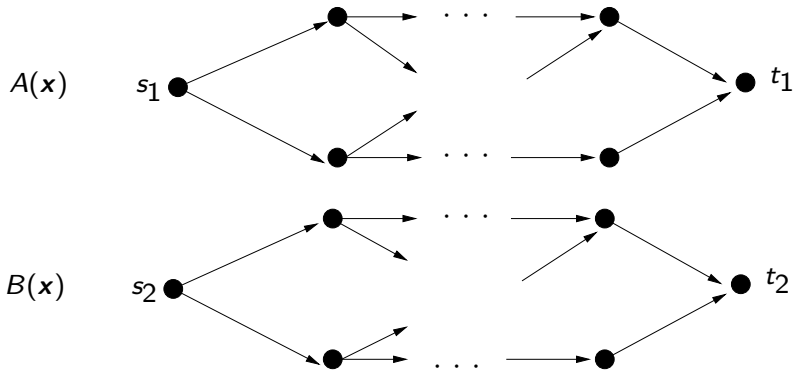


Figure: Sum of two ROABPs

Previous Work

- [Raz and Shpilka, 2005] gave a polynomial time whitebox test for ROABP.
- Later, quasi-polynomial time blackbox tests were obtained [Forbes and Shpilka, 2013, Forbes et al., 2014, Agrawal et al., 2014].
- Sum of two ROABPs: we give the first polynomial time whitebox test (different variable orders)

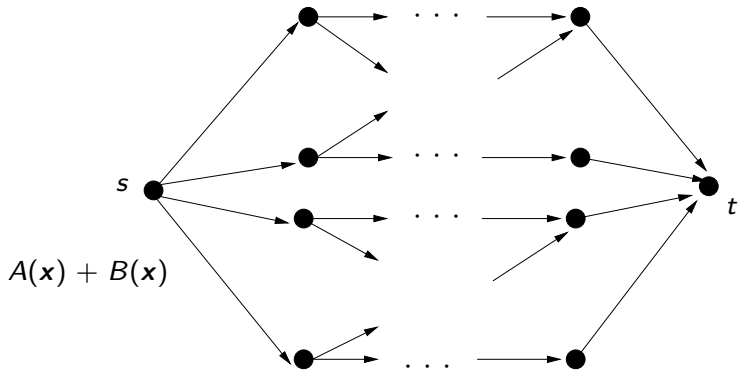


Figure: Sum of two ROABPs

Previous Work

- [Raz and Shpilka, 2005] gave a polynomial time whitebox test for ROABP.
- Later, quasi-polynomial time blackbox tests were obtained [Forbes and Shpilka, 2013, Forbes et al., 2014, Agrawal et al., 2014].
- Sum of two ROABPs: we give the first polynomial time whitebox test (different variable orders)

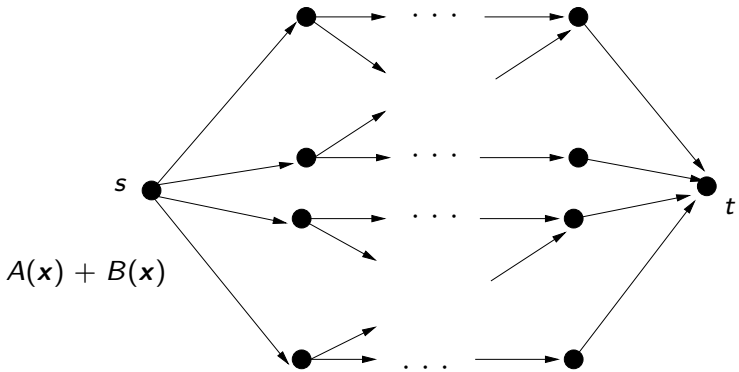


Figure: Sum of two ROABPs

- Sum of two ROABPs not captured by ROABP [Nair and Saha, 2014].

Evaluation Dimension or Partial Coefficient Dimension [Nisan, 1991]

- $\mathbb{F}$-linear dependence of polynomials:

$$\begin{aligned} P_1 &= 1 + x_1 \\ P_2 &= x_1 x_2 \\ P_3 &= 1 + x_1 + 2x_1 x_2 \end{aligned}$$

$$P_1 + 2P_2 - P_3 = 0$$

Evaluation Dimension or Partial Coefficient Dimension [Nisan, 1991]

- Any multilinear polynomial $A(\boldsymbol{x})$ can be written as:

$$A = A_0 + x_1 A_1,$$

where $A_0, A_1 \in \mathbb{F}[x_2, x_3, \ldots, x_n]$.

Evaluation Dimension or Partial Coefficient Dimension [Nisan, 1991]

- Any multilinear polynomial $A(\boldsymbol{x})$ can be written as:

$$A = A_0 + x_1 A_1,$$

where $A_0, A_1 \in \mathbb{F}[x_2, x_3, \ldots, x_n]$.

- Similarly,

$$A = A_{00} + x_1 A_{10} + x_2 A_{01} + x_1 x_2 A_{11},$$

where $A_{00}, A_{10}, A_{01}, A_{11} \in \mathbb{F}[x_3, \ldots, x_n]$.

Evaluation Dimension or Partial Coefficient Dimension [Nisan, 1991]

- Any multilinear polynomial $A(\boldsymbol{x})$ can be written as:

$$A = A_0 + x_1 A_1,$$

where $A_0, A_1 \in \mathbb{F}[x_2, x_3, \ldots, x_n]$.

- Similarly,

$$A = A_{00} + x_1 A_{10} + x_2 A_{01} + x_1 x_2 A_{11},$$

where $A_{00}, A_{10}, A_{01}, A_{11} \in \mathbb{F}[x_3, \ldots, x_n]$.

$$A = \sum_{\boldsymbol{e} \in \{0,1\}^i} x_1^{e_1} x_2^{e_2} \cdots x_i^{e_i} A_{\boldsymbol{e}}, \text{ where } A_{\boldsymbol{e}} \in \mathbb{F}[x_{i+1}, \ldots, x_n]$$

Coefficient Dimension = Minimum width [Nisan, 1991]

- A multilinear polynomial A is computed by an ROABP of width-w and variable order $(x_1, x_2, \ldots, x_n)$ if and only if

$$\dim\{A_{\boldsymbol{e}} \mid \boldsymbol{e} \in \{0,1\}^i\} \leq w, \ \forall i \in [n].$$

Coefficient Dimension $\leq$ Width

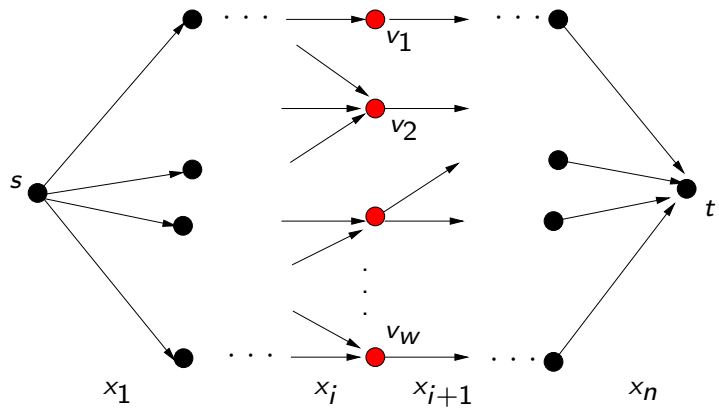
- A multilinear polynomial A is computed by an ROABP of width-w and variable order $(x_1, x_2, \ldots, x_n)$ $\Longrightarrow$

$$\dim\{A_{\boldsymbol{e}} \mid \boldsymbol{e} \in \{0,1\}^i\} \leq w, \ \forall i \in [n].$$

Coefficient Dimension $\leq$ Width

- A multilinear polynomial A is computed by an ROABP of width-w and variable order $(x_1, x_2, \ldots, x_n)$ $\Longrightarrow$

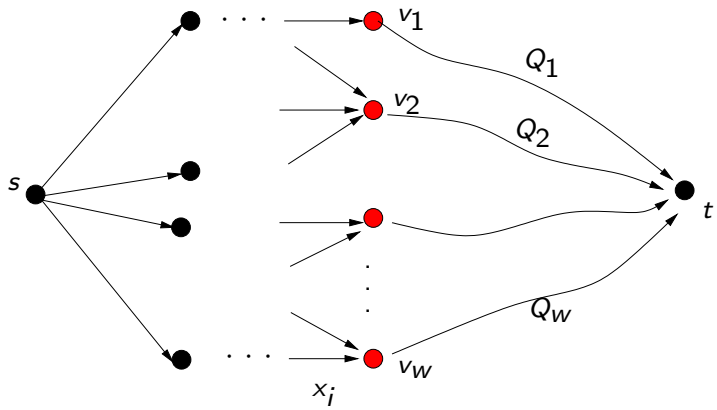
$$\dim\{A_{\boldsymbol{e}} \mid \boldsymbol{e} \in \{0,1\}^i\} \leq w, \ \forall i \in [n].$$



Coefficient Dimension $\leq$ Width

- A multilinear polynomial A is computed by an ROABP of width-w and variable order $(x_1, x_2, \ldots, x_n)$ $\Longrightarrow$

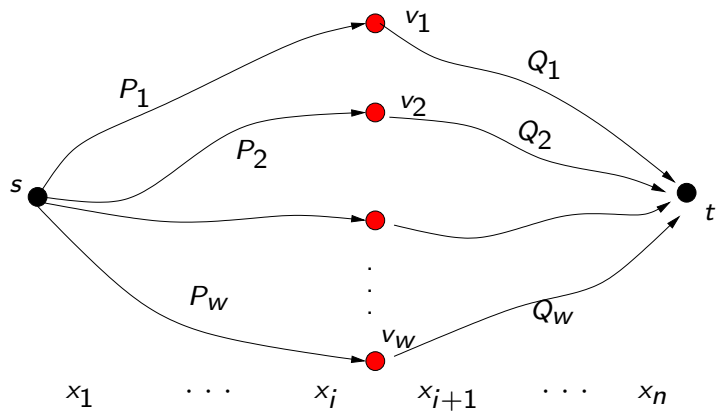
$$\dim\{A_{\boldsymbol{e}} \mid \boldsymbol{e} \in \{0,1\}^i\} \leq w, \ \forall i \in [n].$$



Coefficient Dimension $\leq$ Width

- A multilinear polynomial A is computed by an ROABP of width-w and variable order $(x_1, x_2, \ldots, x_n)$ $\Longrightarrow$

$$\dim\{A_{\boldsymbol{e}} \mid \boldsymbol{e} \in \{0,1\}^i\} \leq w, \ \forall i \in [n].$$

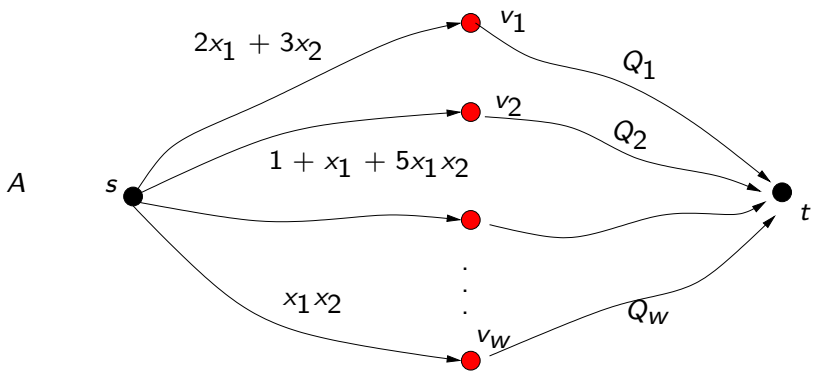


- $A = \sum_{j=1}^{w} P_j Q_j$, where $P_j \in \mathbb{F}[x_1, \ldots, x_i]$ and $Q_j \in \mathbb{F}[x_{i+1}, \ldots, x_n]$ for each j.

Coefficient Dimension $\leq$ width

- A multilinear polynomial A is computed by an ROABP of width-w and variable order $(x_1, x_2, \ldots, x_n)$ $\Longrightarrow$

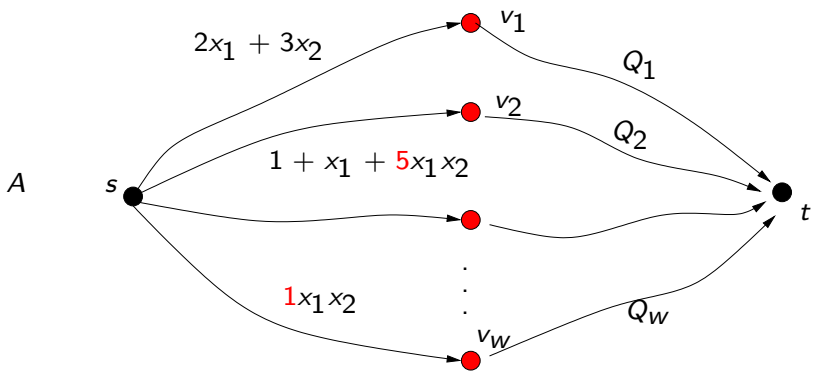
$$\dim\{A_{\boldsymbol{e}} \mid \boldsymbol{e} \in \{0,1\}^i\} \leq w, \ \forall i \in [n].$$



Coefficient Dimension $\leq$ width

- A multilinear polynomial A is computed by an ROABP of width-w and variable order $(x_1, x_2, \ldots, x_n)$ $\Longrightarrow$

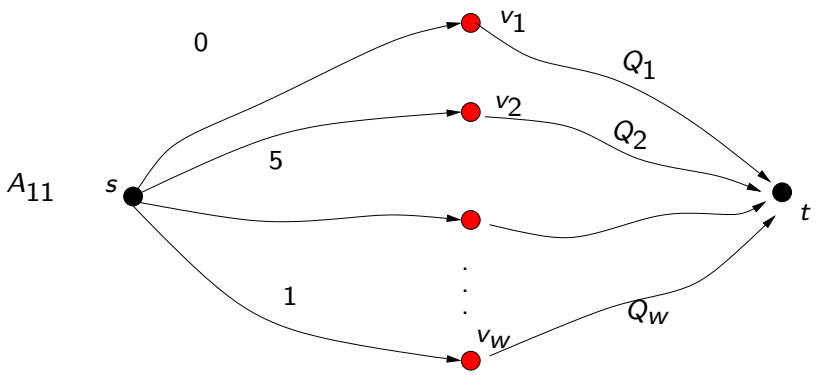
$$\dim\{A_{\boldsymbol{e}} \mid \boldsymbol{e} \in \{0,1\}^i\} \leq w, \ \forall i \in [n].$$



Coefficient Dimension $\leq$ width

- A multilinear polynomial A is computed by an ROABP of width-w and variable order $(x_1, x_2, \ldots, x_n)$ $\Longrightarrow$

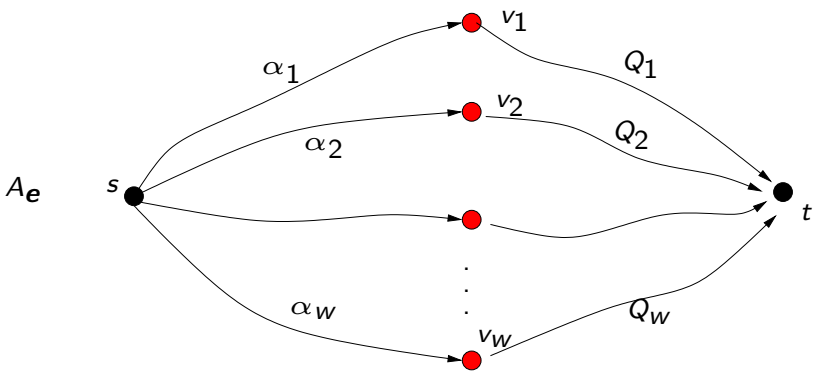
$$\dim\{A_{\boldsymbol{e}} \mid \boldsymbol{e} \in \{0,1\}^i\} \leq w, \ \forall i \in [n].$$



Coefficient Dimension $\leq$ width

- A multilinear polynomial A is computed by an ROABP of width-w and variable order $(x_1, x_2, \ldots, x_n)$ $\Longrightarrow$

$$\dim\{A_{\boldsymbol{e}} \mid \boldsymbol{e} \in \{0,1\}^i\} \leq w, \ \forall i \in [n].$$



$$A_{\boldsymbol{e}} = \sum_{j=1}^{w} \alpha_j Q_j,$$

where $\alpha_j \in \mathbb{F}$ and $Q_j \in \mathbb{F}[x_{i+1}, \ldots, x_n]$ for each j.

Coefficient Dimension = Minimum width [Nisan, 1991]

- A multilinear polynomial A is computed by an ROABP of width-w and variable order $(x_1, x_2, \ldots, x_n)$ $\Longleftarrow$

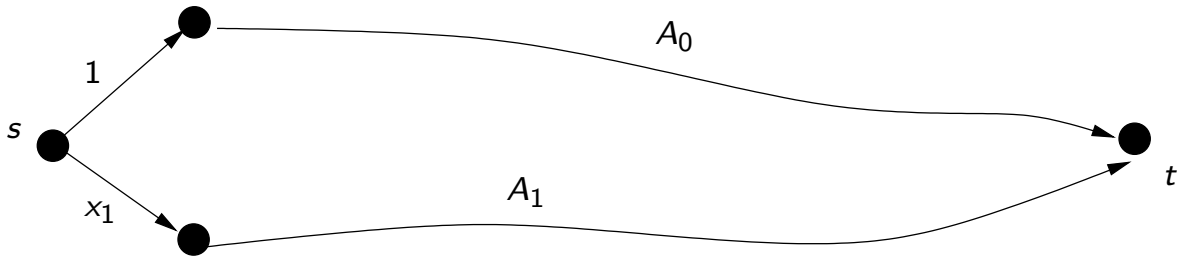
$$\dim\{A_{\boldsymbol{e}} \mid \boldsymbol{e} \in \{0,1\}^i\} \leq w, \ \forall i \in [n].$$

Width $\leq$ Coefficient Dimension

- For a polynomial A, small Coefficient dimension $\implies$ small width ROABP.

Width $\leq$ Coefficient Dimension

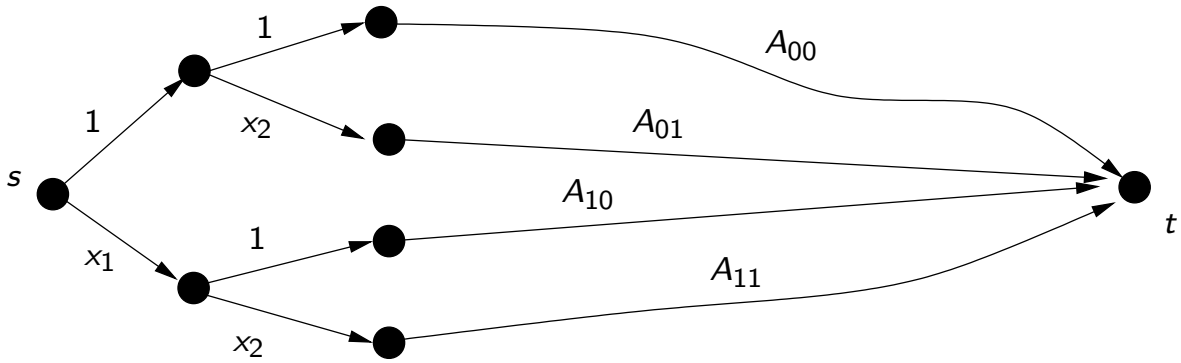
- For a polynomial A, small Coefficient dimension $\implies$ small width ROABP.



$$A = A_0 + x_1 A_1$$

Width $\leq$ Coefficient Dimension

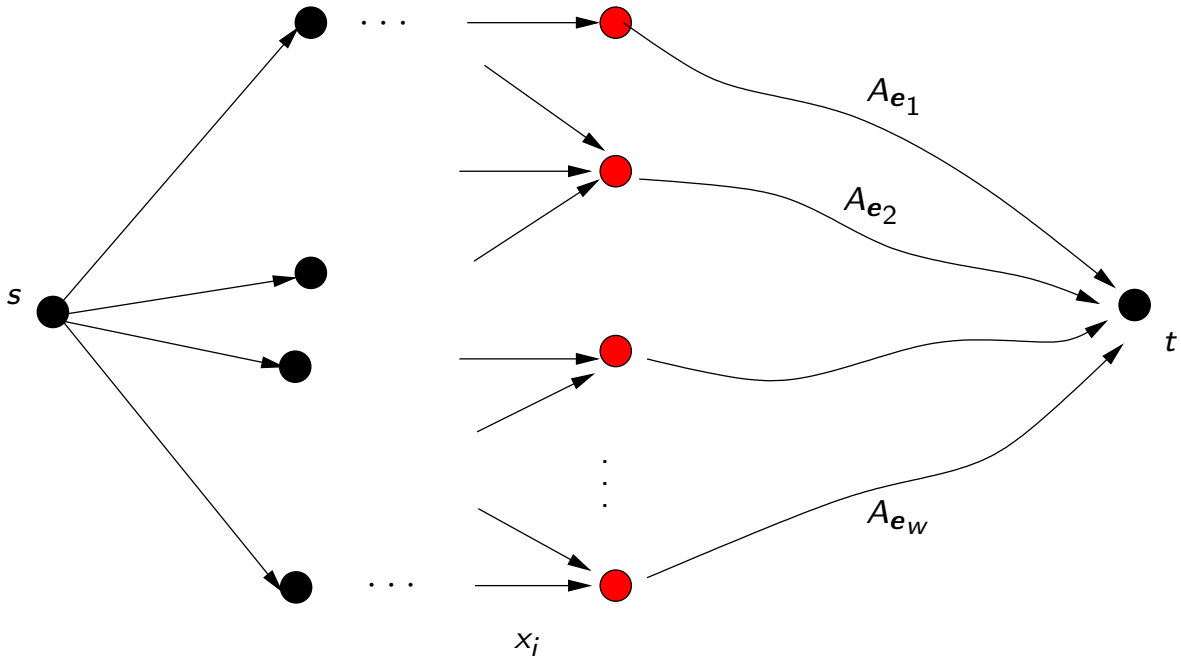
- For a polynomial A, small Coefficient dimension $\implies$ small width ROABP.



$$A = A_{00} + x_1 A_{10} + x_2 A_{01} + x_1 x_2 A_{11}$$

Width $\leq$ Coefficient Dimension

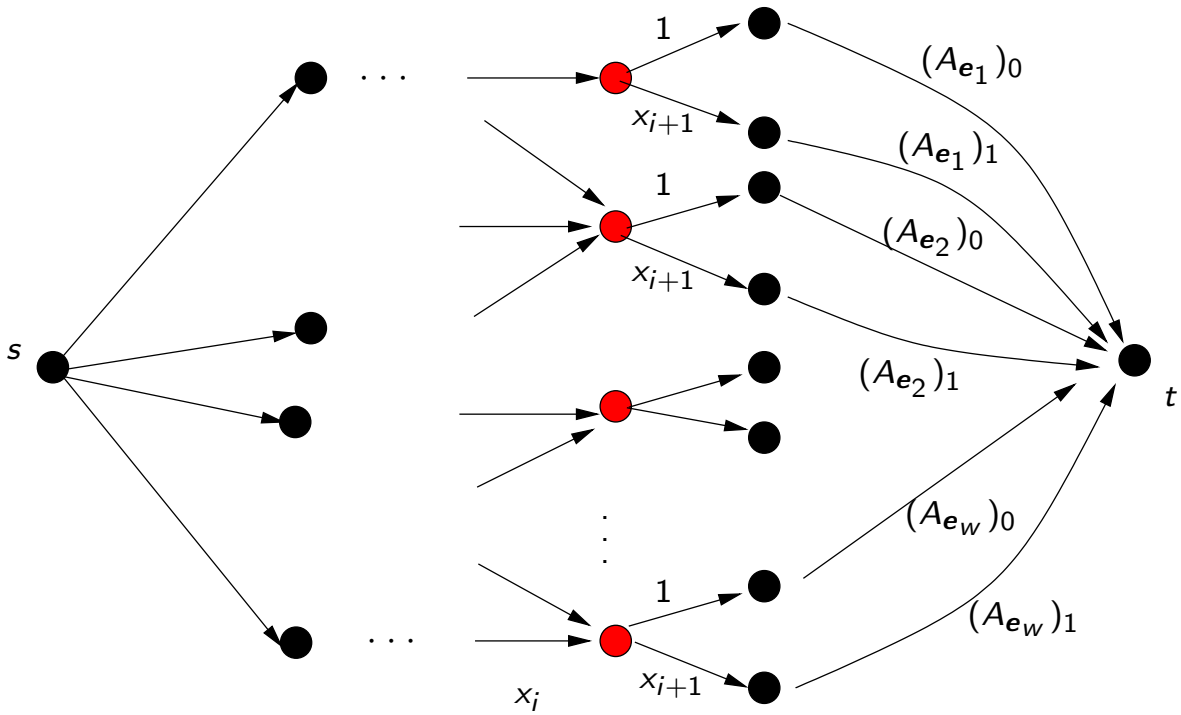
- For a polynomial A, small Coefficient dimension $\implies$ small width ROABP.



$$A = P_1 A_{e_1} + P_2 A_{e_2} + \cdots + P_w A_{e_w}$$

Width $\leq$ Coefficient Dimension

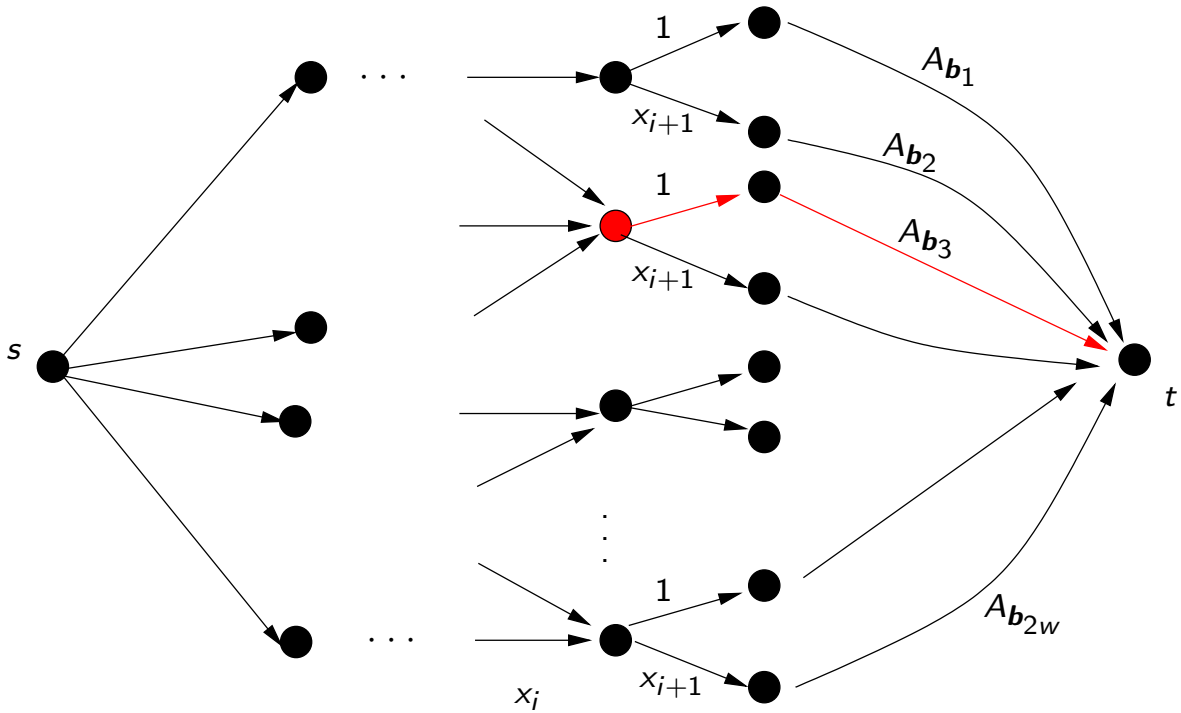
- For a polynomial A, small Coefficient dimension $\Longrightarrow$ small width ROABP.



$$A_{e_j} = (A_{e_j})_0 + x_{i+1}(A_{e_j})_1$$

Width $\leq$ Coefficient Dimension

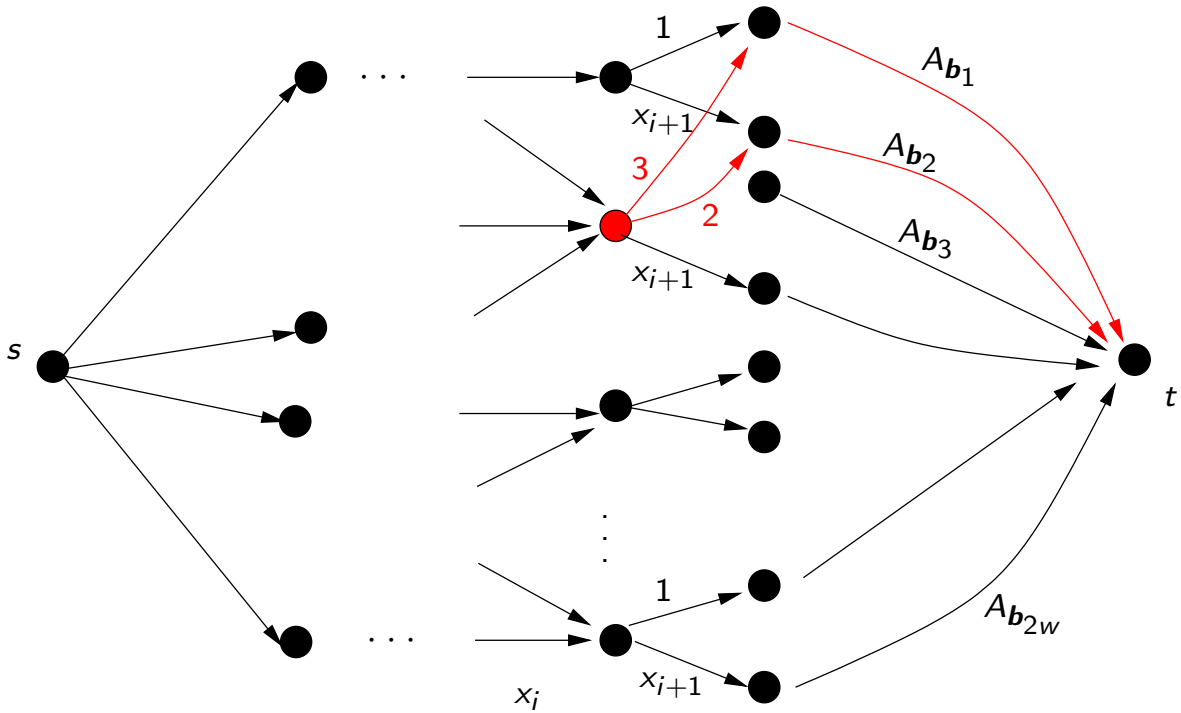
- For a polynomial A, small Coefficient dimension $\implies$ small width ROABP.



Suppose $A_{b_3} = 3A_{b_1} + 2A_{b_2}$

Width $\leq$ Coefficient Dimension

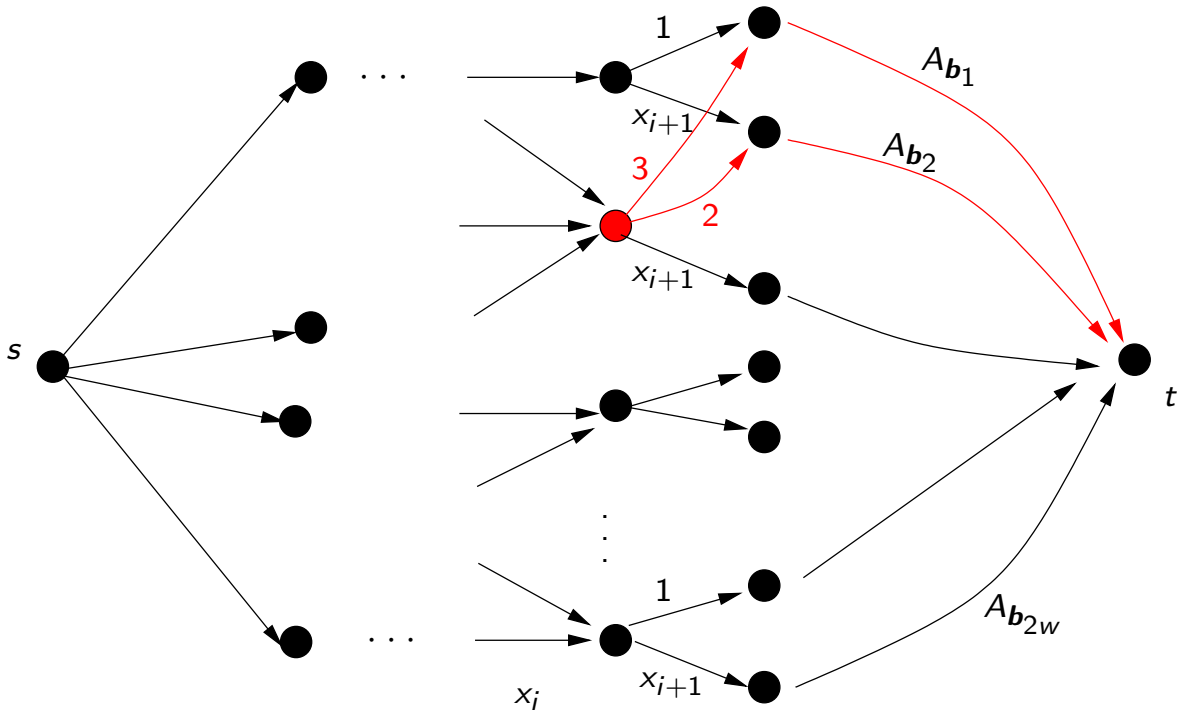
- For a polynomial A, small Coefficient dimension $\implies$ small width ROABP.



Suppose $A_{b_3} = 3A_{b_1} + 2A_{b_2}$

Width $\leq$ Coefficient Dimension

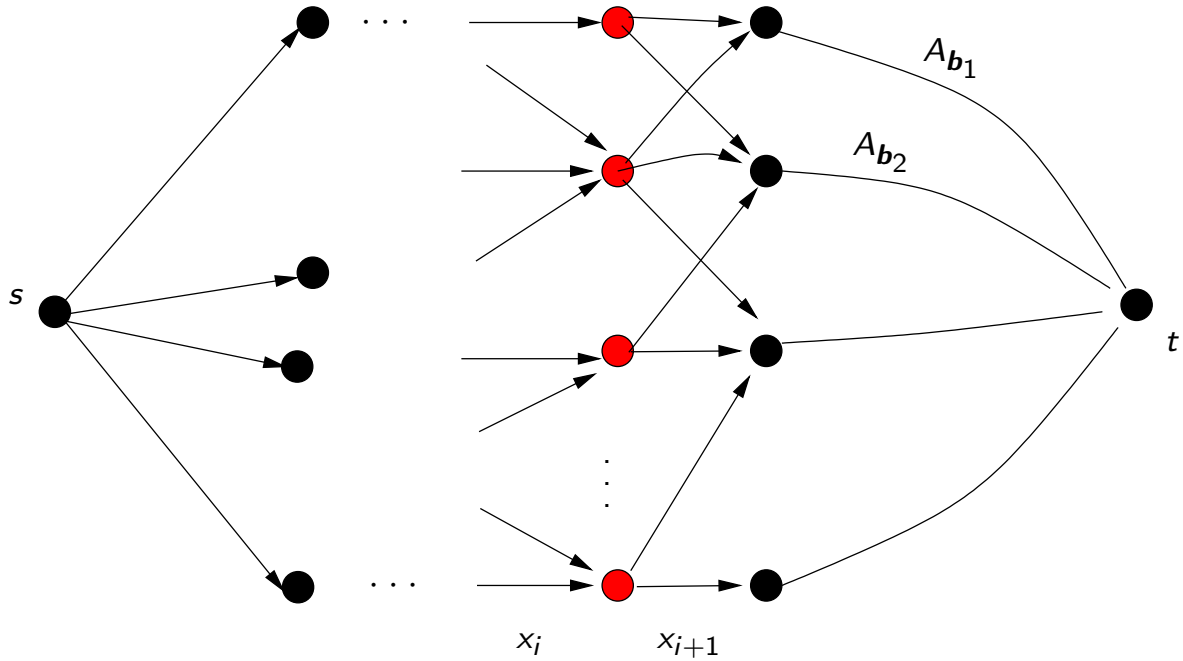
- For a polynomial A, small Coefficient dimension $\implies$ small width ROABP.



Suppose $A_{b_3} = 3A_{b_1} + 2A_{b_2}$

Width $\leq$ Coefficient Dimension

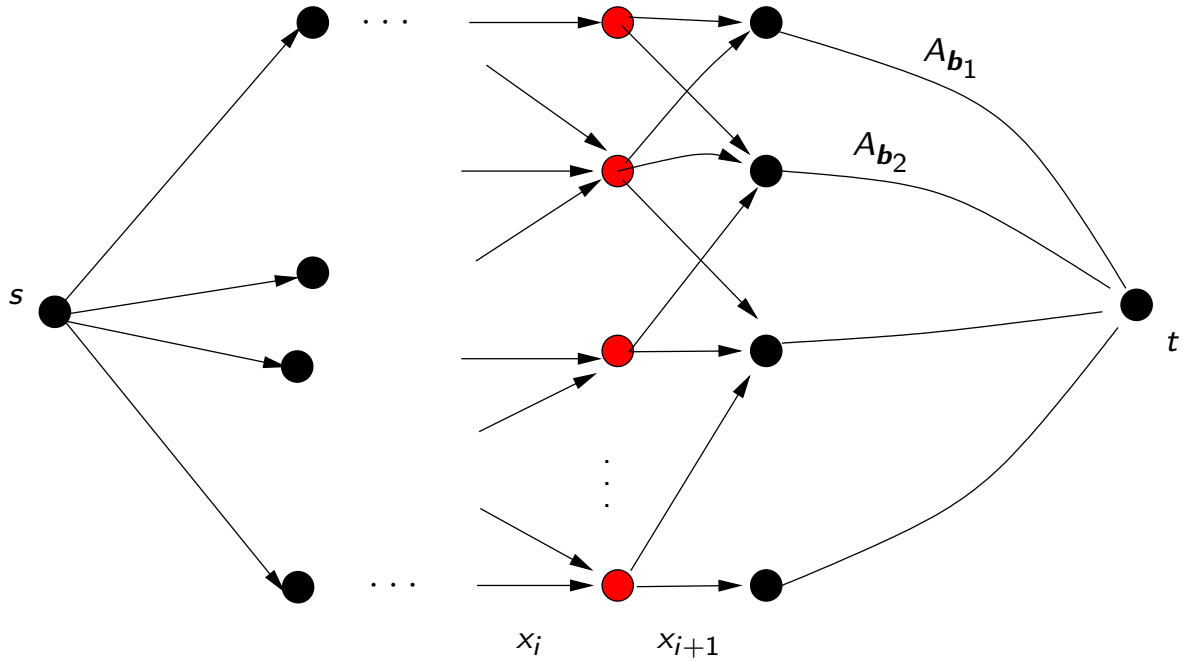
- For a polynomial A, small Coefficient dimension $\implies$ small width ROABP.



- Note that these dependencies essentially define the ROABP.

Width $\leq$ Coefficient Dimension

- For a polynomial A, small Coefficient dimension $\implies$ small width ROABP.



- Dependency support at most $w + 1$.

Equivalence of two ROABPs

- $A = B$?, where A has an ROABP in variable order $(x_1, x_2, \ldots, x_n)$.

Equivalence of two ROABPs

- $A = B$?, where A has an ROABP in variable order $(x_1, x_2, \ldots, x_n)$.
- Step 1: Find the linear dependencies among partial coefficients of A.
- Step 2: Verify if the same dependencies hold among the corresponding partial coefficients of B.

Equivalence of two ROABPs

- $A = B$?, where A has an ROABP in variable order $(x_1, x_2, \ldots, x_n)$.
- Step 1: Find the linear dependencies among partial coefficients of A.
- Step 2: Verify if the same dependencies hold among the corresponding partial coefficients of B.
- For example, if $A_{\boldsymbol{b}_3} = 3A_{\boldsymbol{b}_1} + 2A_{\boldsymbol{b}_2}$, then verify whether

$$B_{\boldsymbol{b}_3} = 3B_{\boldsymbol{b}_1} + 2B_{\boldsymbol{b}_2}.$$

Equivalence of two ROABPs

- $A = B$?, where A has an ROABP in variable order $(x_1, x_2, \ldots, x_n)$.
- Step 1: Find the linear dependencies among partial coefficients of A.
- Step 2: Verify if the same dependencies hold among the corresponding partial coefficients of B.
- For example, if $A_{\boldsymbol{b}_3} = 3A_{\boldsymbol{b}_1} + 2A_{\boldsymbol{b}_2}$, then verify whether
$$B_{\boldsymbol{b}_3} = 3B_{\boldsymbol{b}_1} + 2B_{\boldsymbol{b}_2}.$$
- If any dependency fails, then $A \neq B$.

Equivalence of two ROABPs

- $A = B$?, where A has an ROABP in variable order $(x_1, x_2, \ldots, x_n)$.
- Step 1: Find the linear dependencies among partial coefficients of A.
- Step 2: Verify if the same dependencies hold among the corresponding partial coefficients of B.
- For example, if $A_{\boldsymbol{b}_3} = 3A_{\boldsymbol{b}_1} + 2A_{\boldsymbol{b}_2}$, then verify whether

$$B_{\boldsymbol{b}_3} = 3B_{\boldsymbol{b}_1} + 2B_{\boldsymbol{b}_2}.$$

- If any dependency fails, then $A \neq B$.
- If all the dependencies hold then $A = cB$ for some constant c.

Equivalence of two ROABPs

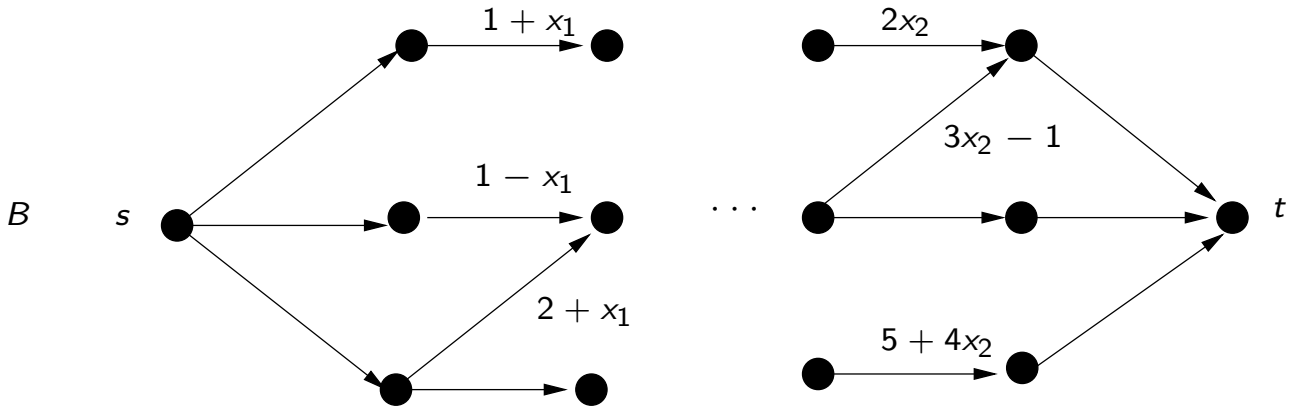
- $A = B$?, where A has an ROABP in variable order $(x_1, x_2, \ldots, x_n)$.
- Step 1: Find the linear dependencies among partial coefficients of A.
- Step 2: Verify if the same dependencies hold among the corresponding partial coefficients of B.
- For example, if $A_{\boldsymbol{b}_3} = 3A_{\boldsymbol{b}_1} + 2A_{\boldsymbol{b}_2}$, then verify whether

$$B_{\boldsymbol{b}_3} = 3B_{\boldsymbol{b}_1} + 2B_{\boldsymbol{b}_2}.$$

- If any dependency fails, then $A \neq B$.
- If all the dependencies hold then $A = cB$ for some constant c.
- Question: how to verify dependencies for B?

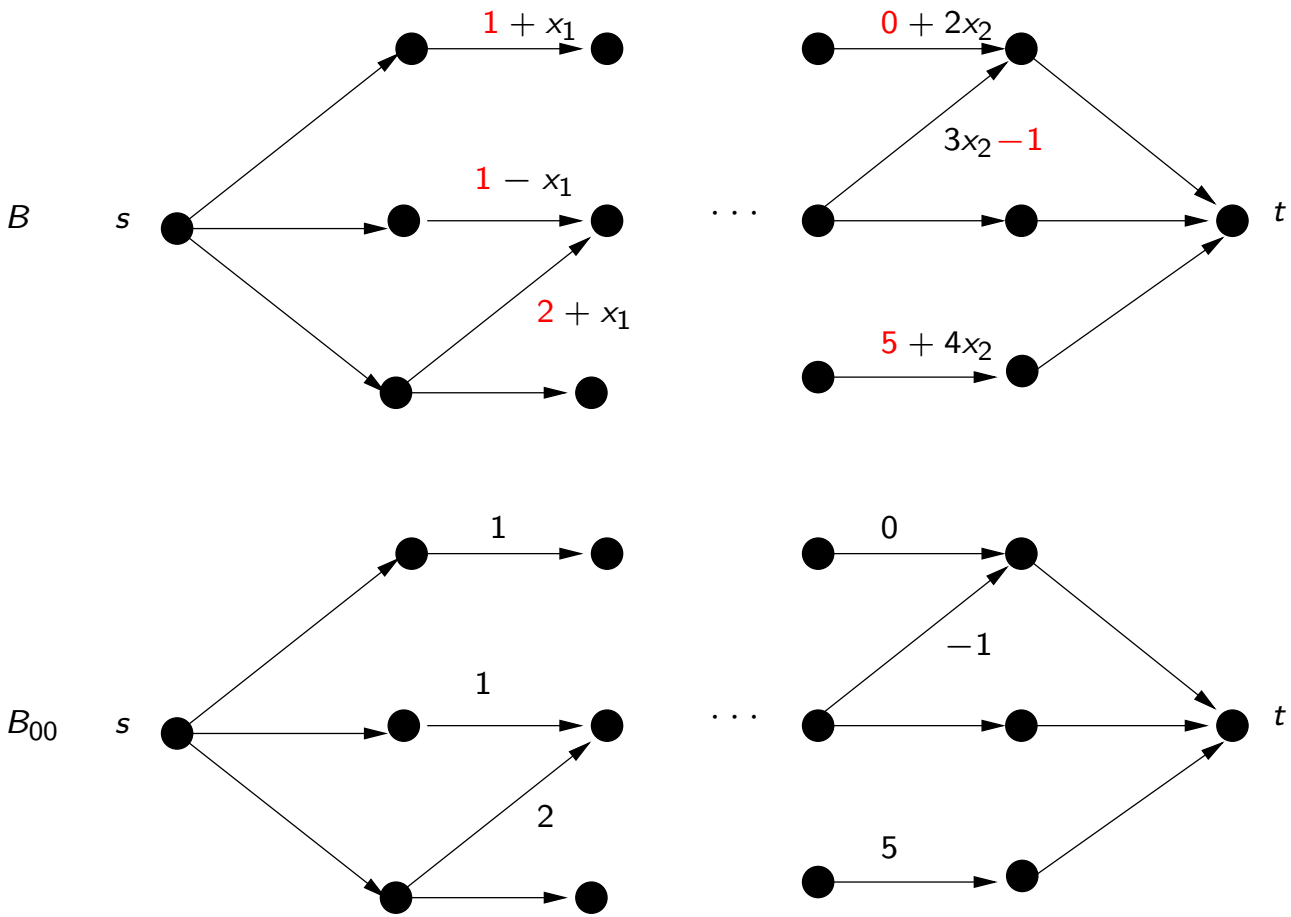
Verifying Dependencies for B

- For example $B_{00} + 2B_{10} - B_{11} = 0$.



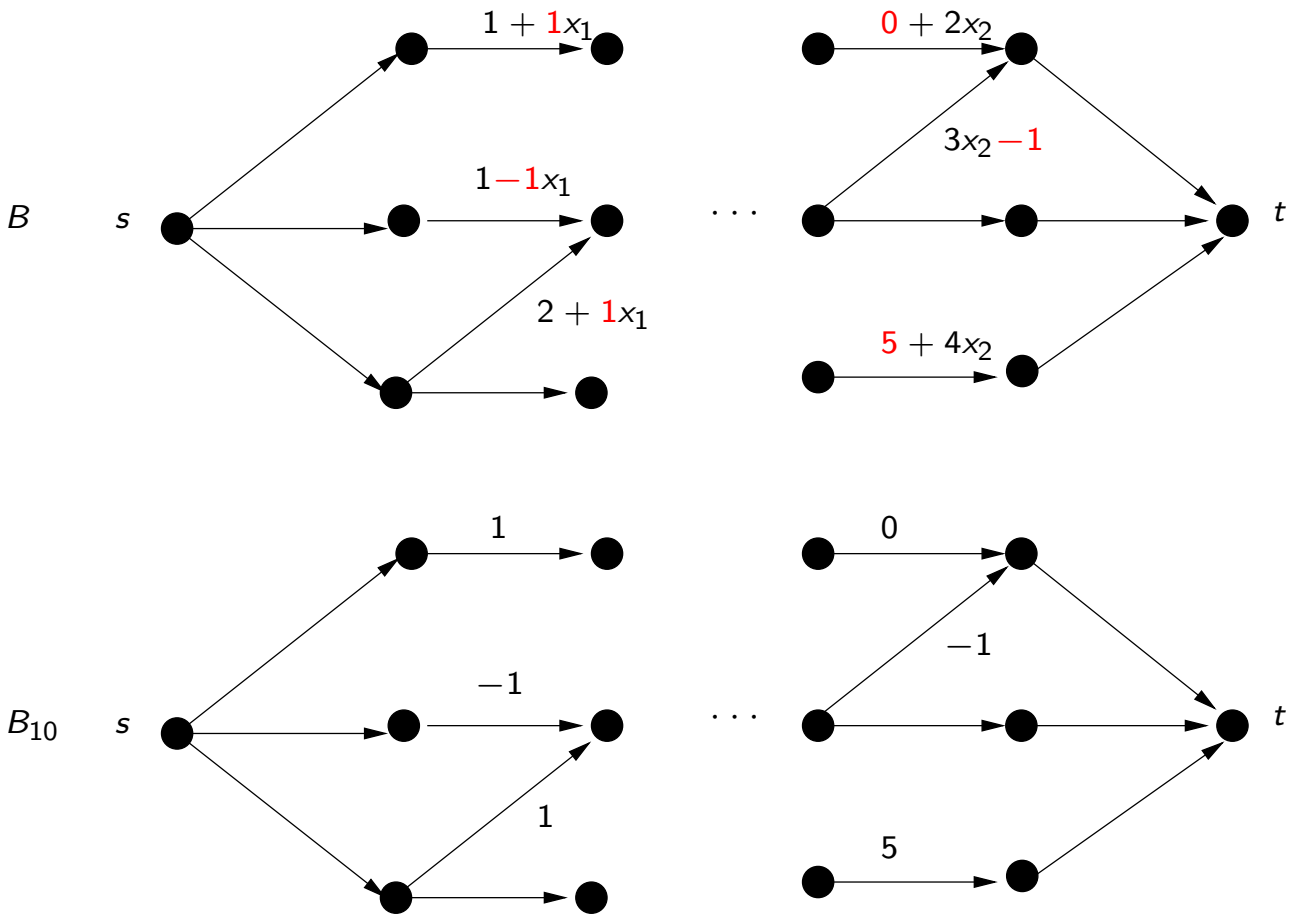
Verifying Dependencies for B

- For example $B_{00} + 2B_{10} - B_{11} = 0$.



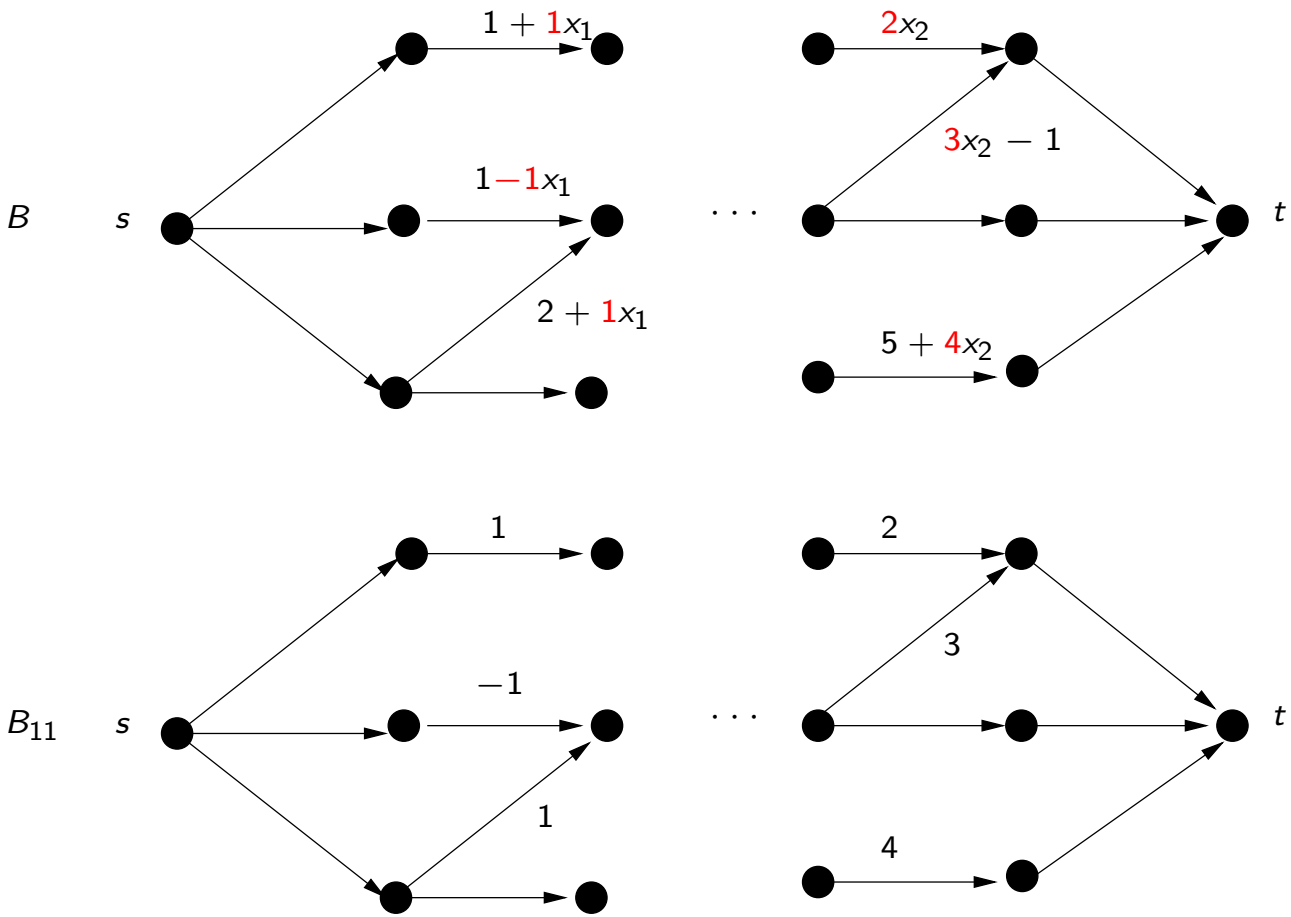
Verifying Dependencies for B

- For example $B_{00} + 2B_{10} - B_{11} = 0$.



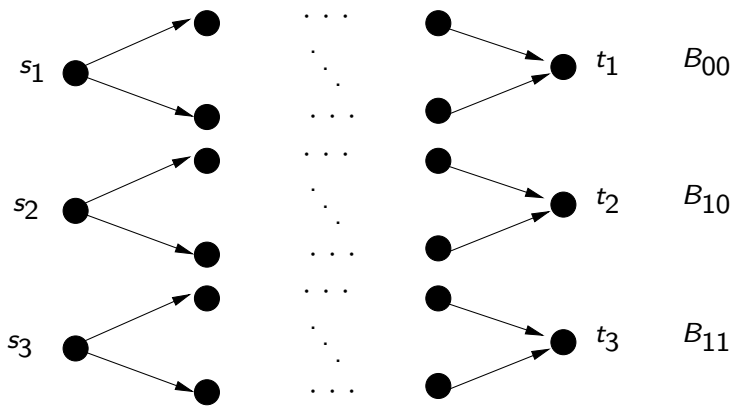
Verifying Dependencies for B

- For example $B_{00} + 2B_{10} - B_{11} = 0$.



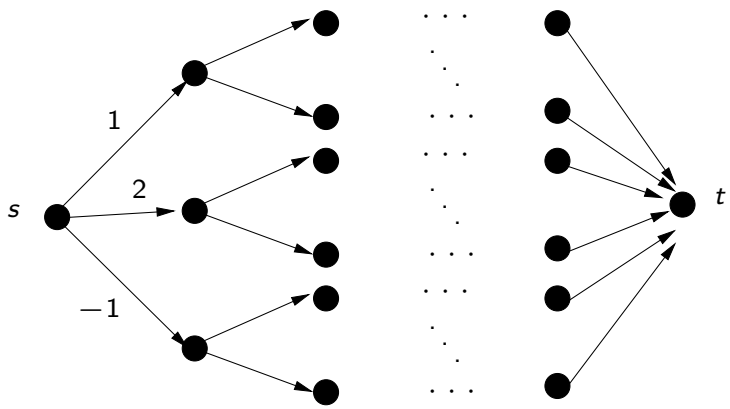
Verifying Dependencies

- Verifying dependencies for B?
- For example $B_{00} + 2B_{10} - B_{11} = 0$.
- B_{00}, B_{10} and B_{11} all three have an ROABP in the same variable order.



Verifying Dependencies

- Verifying dependencies for B?
- For example $B_{00} + 2B_{10} - B_{11} = 0$.
- B_{00}, B_{10} and B_{11} all three have an ROABP in the same variable order.



- It reduces to zero testing for an ROABP of a larger width ($\leq w(w+1)$).

Generalizations

- We generalize this test to sum of c ROABPs with time complexity $\text{poly}(w^{2^c} n^c)$.

Generalizations

- We generalize this test to sum of c ROABPs with time complexity $\text{poly}(w^{2^c} n^c)$.
- We also make this test blackbox but with quasi-polynomial cost $(nw)^{c \cdot 2^c \log nw}$.
- based on the ideas of *least basis isolation* and *low-support rank-concentration*.

Generalizations

- We generalize this test to sum of c ROABPs with time complexity $\text{poly}(w^{2^c} n^c)$.
- We also make this test blackbox but with quasi-polynomial cost $(nw)^{c \cdot 2^c \log nw}$.
- based on the ideas of *least basis isolation* and *low-support rank-concentration*.
- The same techniques work in the case of higher individual degree, with essentially the same time complexity.

Discussion

- Can one bring down the time complexity from $w^{O(2^c)}$ to $w^{O(c)}$?
- Can we use these ideas to solve depth-3 multilinear circuits?

Discussion

- Can one bring down the time complexity from $w^{O(2^c)}$ to $w^{O(c)}$?
- Can we use these ideas to solve depth-3 multilinear circuits?
- Our proof was inspired from a question in the boolean setting – equivalence of two ROBPs. Are there more connections?

Thank you

Generalizations

- We generalize this test to sum of c ROABPs with time complexity $\text{poly}(w^{2^c} n^c)$.

Generalizations

- We generalize this test to sum of c ROABPs with time complexity $\text{poly}(w^{2^c} n^c)$.
- $A_1 = A_2 + A_3 + \cdots + A_{c-1}$?

Generalizations

- We generalize this test to sum of c ROABPs with time complexity $\text{poly}(w^{2^c} n^c)$.
- $A_1 = A_2 + A_3 + \cdots + A_{c-1}$?
- Take dependencies of A_1 and verify for $A_2 + A_3 + \cdots A_{c-1}$.

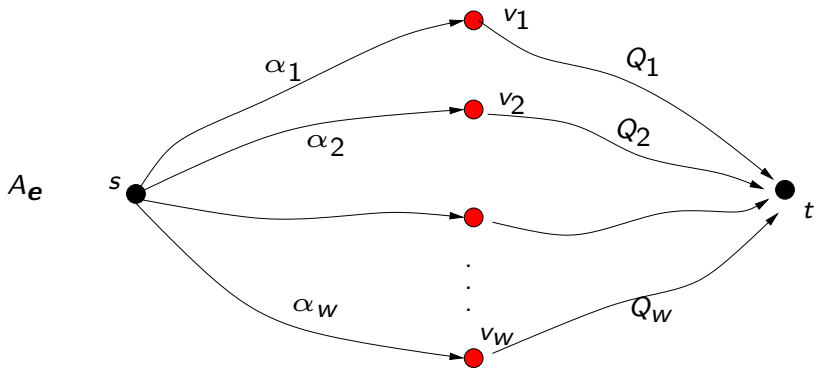
Generalizations

- We generalize this test to sum of c ROABPs with time complexity $\text{poly}(w^{2^c} n^c)$.
- $A_1 = A_2 + A_3 + \cdots + A_{c-1}$?
- Take dependencies of A_1 and verify for $A_2 + A_3 + \cdots A_{c-1}$.
- Reduces to PIT for sum of $c - 1$ ROABPs.

$$T(w, c) = nwT(c - 1, w^2) + \text{poly}(n, w)$$

Dependencies for A

- Say, A has variable order $(x_1, x_2, \ldots, x_n)$.

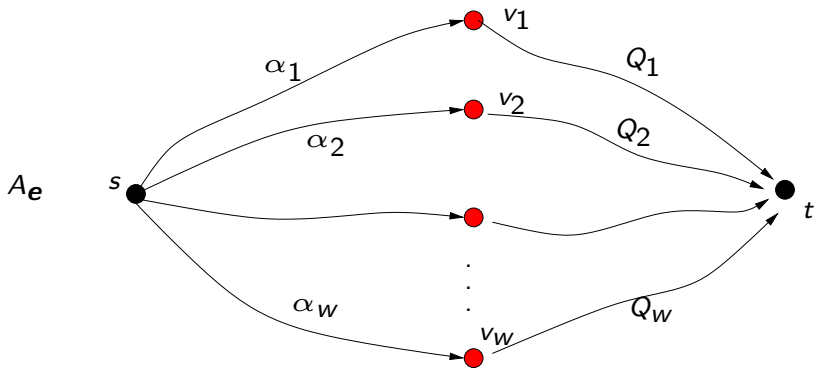


- Recall that for any $\boldsymbol{e} \in \{0,1\}^i$,

$$A_{\boldsymbol{e}} = \sum_{j=1}^{w} \alpha_j Q_j, \text{ where } \alpha_j \in \mathbb{F}$$

Dependencies for A

- Say, A has variable order $(x_1, x_2, \ldots, x_n)$.



- Recall that for any $\boldsymbol{e} \in \{0,1\}^i$,

$$A_{\boldsymbol{e}} = \sum_{j=1}^{w} \alpha_j Q_j, \text{ where } \alpha_j \in \mathbb{F}$$

- Consider the vectors $(\alpha_1, \alpha_2, \ldots, \alpha_w)$, and find the dependencies.

Agrawal, M. (2005).

Proving lower bounds via pseudo-random generators.
In *FSTTCS*, volume 3821 of *Lecture Notes in Computer Science*, pages 92–105.

Agrawal, M., Gurjar, R., Korwar, A., and Saxena, N. (2014).

Hitting-sets for ROABP and sum of set-multilinear circuits.
Electronic Colloquium on Computational Complexity (ECCC), 21:85.
(to appear in SICOMP, 2015).

Demillo, R. A. and Lipton, R. J. (1978).

A probabilistic remark on algebraic program testing.
Information Processing Letters, 7(4):193 – 195.

Forbes, M. A., Saptharishi, R., and Shpilka, A. (2014).

Hitting sets for multilinear read-once algebraic branching programs, in any order.
In *Symposium on Theory of Computing, STOC 2014, New York, NY, USA, May 31 - June 03, 2014*, pages 867–875.

Forbes, M. A. and Shpilka, A. (2013).

Quasipolynomial-time identity testing of non-commutative and read-once oblivious algebraic branching programs.
In *FOCS*, pages 243–252.

Kabanets, V. and Impagliazzo, R. (2003).

Derandomizing polynomial identity tests means proving circuit lower bounds.
STOC, pages 355–364.

Nair, V. and Saha, C. (2014).

Personal communication.

Nisan, N. (1991).

Lower bounds for non-commutative computation (extended abstract).
In *Proceedings of the 23rd ACM Symposium on Theory of Computing, ACM Press*, pages 410–418.

Raz, R. and Shpilka, A. (2005).

Deterministic polynomial identity testing in non-commutative models.
Computational Complexity, 14(1):1–19.

Schwartz, J. T. (1980).

Fast probabilistic algorithms for verification of polynomial identities.
J. ACM, 27(4):701–717.

Zippel, R. (1979).

Probabilistic algorithms for sparse polynomials.
In *Proceedings of the International Symposium on on Symbolic and Algebraic Computation*, EUROSAM '79, pages 216–226, London, UK, UK. Springer-Verlag.