Some Closure Results for Polynomial Factorization and Applications

Mrinal Kumar

Simons Institute -> University of Toronto

Joint work with Chi-Ning Chou (Harvard) and Noam Solomon (MIT)

Multivariate Polynomials

$$P = \sum_{\mathbf{e}} \alpha_{\mathbf{e}} \mathbf{X}^{\mathbf{e}}$$

$$\mathbf{e} = (e_1, e_2, \ldots, e_n), \sum_i e_i \leq d \qquad \mathbf{X}^{\mathbf{e}} = X_1^{e_1} X_2^{e_2} \cdots X_n^{e_n}$$

$\alpha_{\mathbf{e}}$- field elements

Polynomial on n variables, of degree d.

Multivariate Polynomials in CS

Multivariate Polynomials in CS

Ubiquitous in Computer Science

Multivariate Polynomials in CS

Ubiquitous in Computer Science

- Algorithm design (Bipartite matching, Subgraph Isomorphism)

Multivariate Polynomials in CS

Ubiquitous in Computer Science

- Algorithm design (Bipartite matching, Subgraph Isomorphism)
- Coding theory (BCH, Reed-Solomon, Reed-Muller, PCPs)

Multivariate Polynomials in CS

Ubiquitous in Computer Science

- Algorithm design (Bipartite matching, Subgraph Isomorphism)
- Coding theory (BCH, Reed-Solomon, Reed-Muller, PCPs)
- Derandomization (Worst Case to Average Case reductions)

Multivariate Polynomials in CS

Ubiquitous in Computer Science

- Algorithm design (Bipartite matching, Subgraph Isomorphism)
- Coding theory (BCH, Reed-Solomon, Reed-Muller, PCPs)
- Derandomization (Worst Case to Average Case reductions)
- Boolean Circuit Complexity (Razborov-Smolensky)

Multivariate Polynomials in CS

Ubiquitous in Computer Science

- Algorithm design (Bipartite matching, Subgraph Isomorphism)
- Coding theory (BCH, Reed-Solomon, Reed-Muller, PCPs)
- Derandomization (Worst Case to Average Case reductions)
- Boolean Circuit Complexity (Razborov-Smolensky)
- Polynomial Method in Combinatorics (Kakeya sets, Distinct distances, Joints problem, Cap sets)

Computation with Multivariate Polynomials

Computation with Multivariate Polynomials

Given a polynomial P, do something…

Computation with Multivariate Polynomials

Given a polynomial P, do something…

- Is P non-zero ?

Computation with Multivariate Polynomials

Given a polynomial P, do something…

- Is P non-zero ?
- What does P evaluate to at the origin ?

Computation with Multivariate Polynomials

Given a polynomial P, do something…

- Is P non-zero ?
- What does P evaluate to at the origin ?
- Output P + Q, P x Q, for some polynomial Q

Computation with Multivariate Polynomials

Given a polynomial P, do something…

- Is P non-zero ?
- What does P evaluate to at the origin ?
- Output P + Q, P x Q, for some polynomial Q
- Output the first order partial derivatives of P

Computation with Multivariate Polynomials

Given a polynomial P, do something…

- Is P non-zero ?
- What does P evaluate to at the origin ?
- Output P + Q, P x Q, for some polynomial Q
- Output the first order partial derivatives of P
- Output an f, which divides P

Computation with Multivariate Polynomials

Given a polynomial P, do something…

- Is P non-zero ?
- What does P evaluate to at the origin ?
- Output P + Q, P x Q, for some polynomial Q
- Output the first order partial derivatives of P
- Output an f, which divides P

But wait, how is P given ?

Representing Multivariate Polynomials

Representing Multivariate Polynomials

Sparse representation : as a sum of monomials $P = \sum_{\mathbf{e}} \alpha_{\mathbf{e}} \mathbf{X}^{\mathbf{e}}$

Representing Multivariate Polynomials

Sparse representation : as a sum of monomials $P = \sum_{\mathbf{e}} \alpha_{\mathbf{e}} \mathbf{X}^{\mathbf{e}}$

Intuitive and natural

Representing Multivariate Polynomials

Sparse representation : as a sum of monomials $P = \sum_{\mathbf{e}} \alpha_{\mathbf{e}} \mathbf{X}^{\mathbf{e}}$

Intuitive and natural

Many operations are easy (sum, product, derivatives…)

Representing Multivariate Polynomials

Sparse representation : as a sum of monomials $P = \sum_{\mathbf{e}} \alpha_{\mathbf{e}} \mathbf{X}^{\mathbf{e}}$

Intuitive and natural

Many operations are easy (sum, product, derivatives…)

But, highly non-succinct (exp(n) monomials, hard to evaluate)

Representing Multivariate Polynomials

Sparse representation : as a sum of monomials $P = \sum_{\mathbf{e}} \alpha_{\mathbf{e}} \mathbf{X}^{\mathbf{e}}$

Intuitive and natural

Many operations are easy (sum, product, derivatives…)

But, highly non-succinct (exp(n) monomials, hard to evaluate)

$$P = \sum_{S \subseteq [n]} \prod_{i \in S} X_i$$

Representing Multivariate Polynomials

Sparse representation : as a sum of monomials $P = \sum_{\mathbf{e}} \alpha_{\mathbf{e}} \mathbf{X}^{\mathbf{e}}$

Intuitive and natural

Many operations are easy (sum, product, derivatives…)

But, highly non-succinct (exp(n) monomials, hard to evaluate)

$$P = \sum_{S \subseteq [n]} \prod_{i \in S} X_i \qquad\qquad P = \prod_{i=[n]} (X_i + 1)$$

Representing Multivariate Polynomials

Sparse representation : as a sum of monomials $P = \sum_{\mathbf{e}} \alpha_{\mathbf{e}} \mathbf{X}^{\mathbf{e}}$

Intuitive and natural

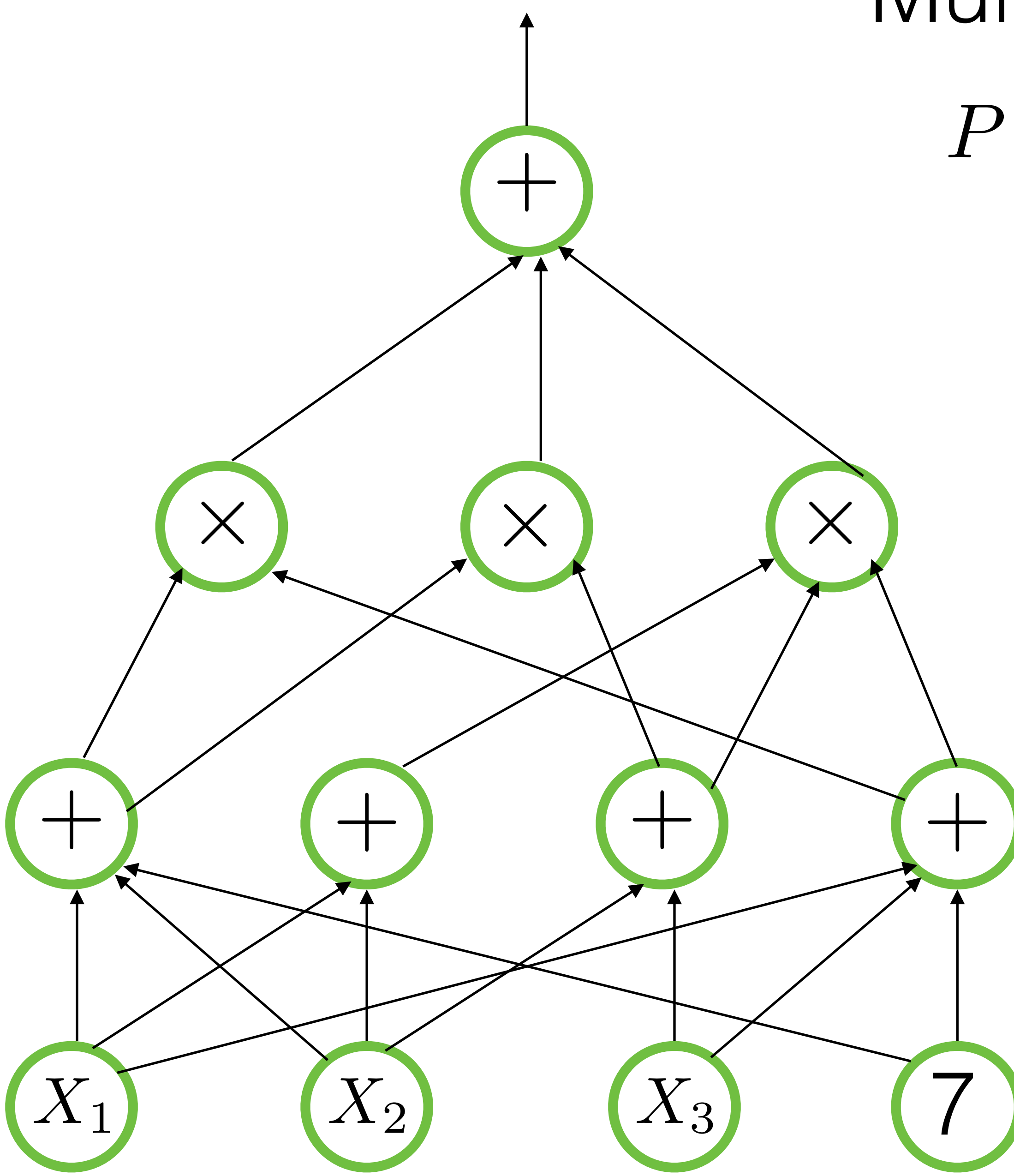
Many operations are easy (sum, product, derivatives…)

But, highly non-succinct (exp(n) monomials, hard to evaluate)

$$P = \sum_{S \subseteq [n]} \prod_{i \in S} X_i \qquad\qquad P = \prod_{i=[n]} (X_i + 1)$$

Is there a representation which is more succinct than sum of monomials?

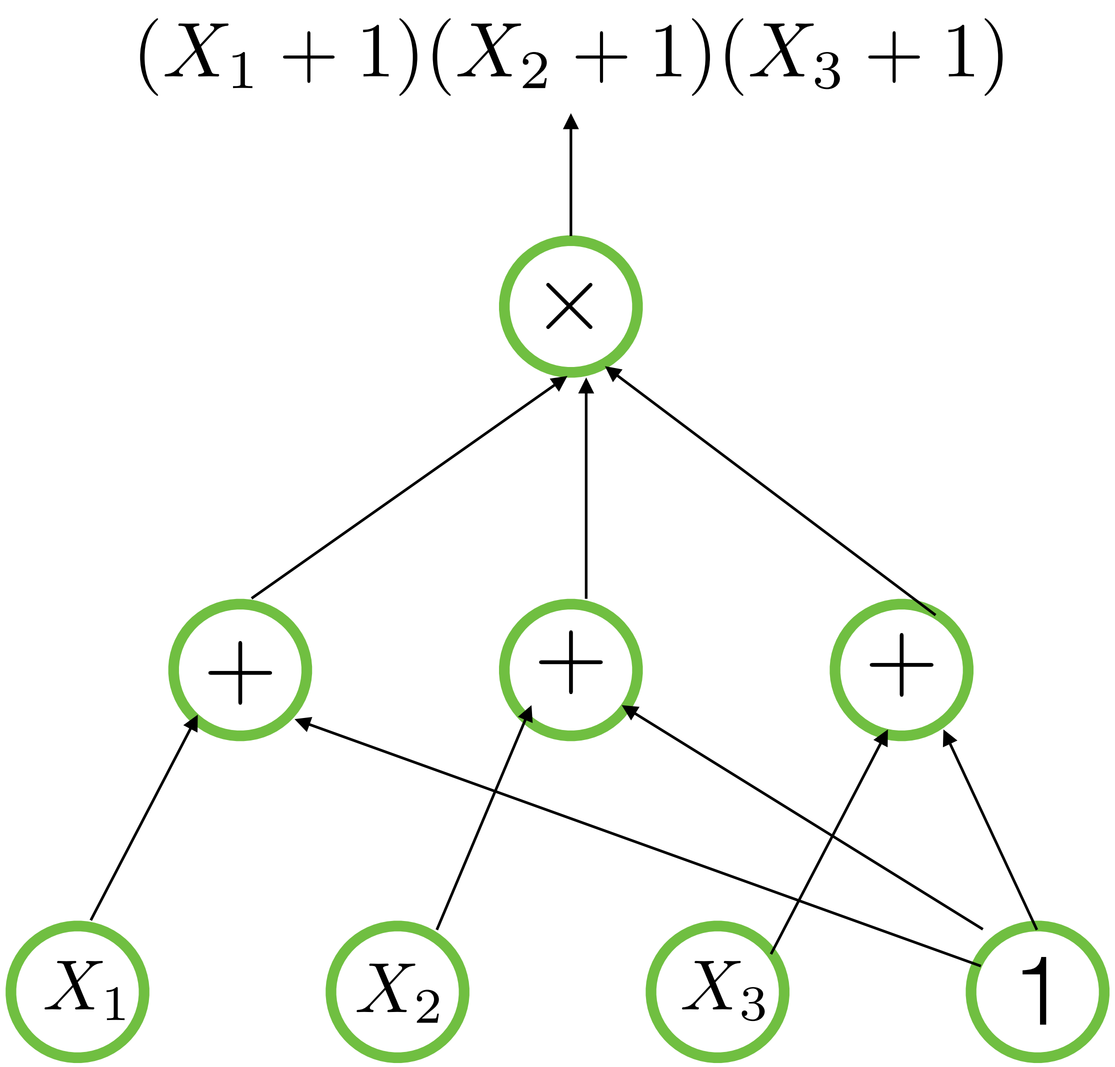
Arithmetic circuits



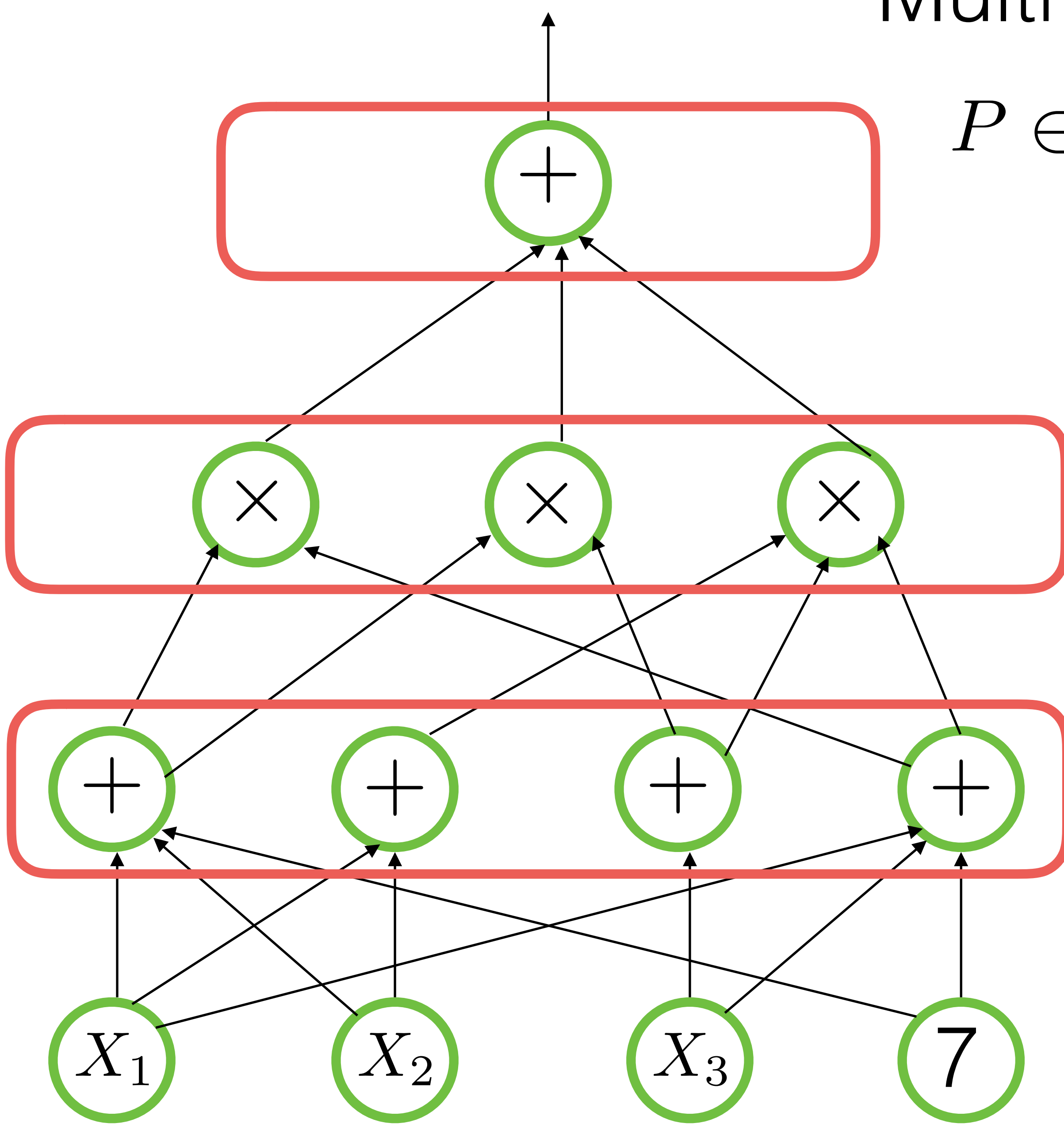
Multivariate polynomial

$$P \in \mathbb{F}[X_1, X_2, \ldots, X_n]$$

Arithmetic circuits



Arithmetic circuits



Multivariate polynomial

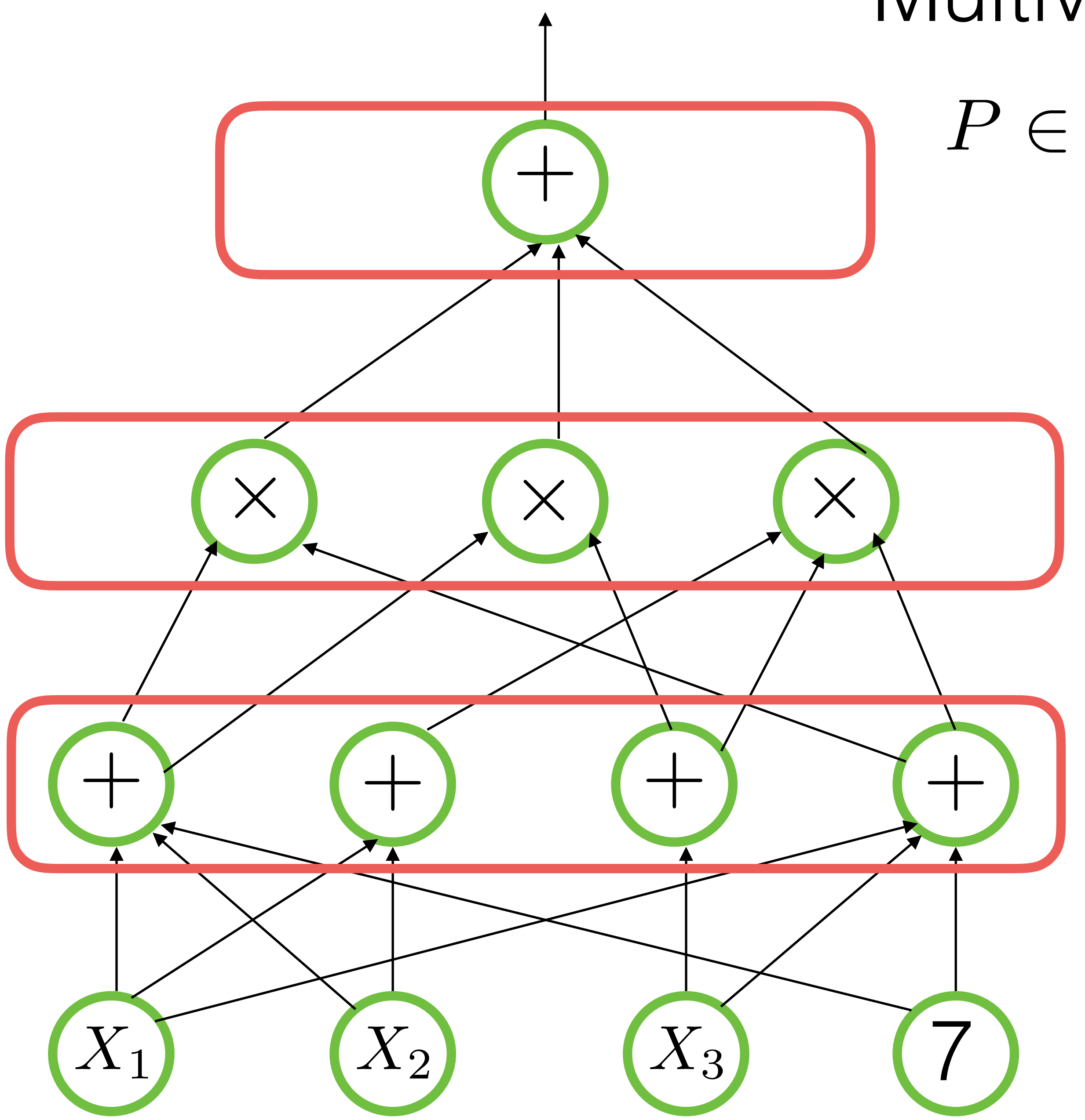
$$P \in \mathbb{F}[X_1, X_2, \ldots, X_n]$$

Depth - 3

Arithmetic circuits

Multivariate polynomial

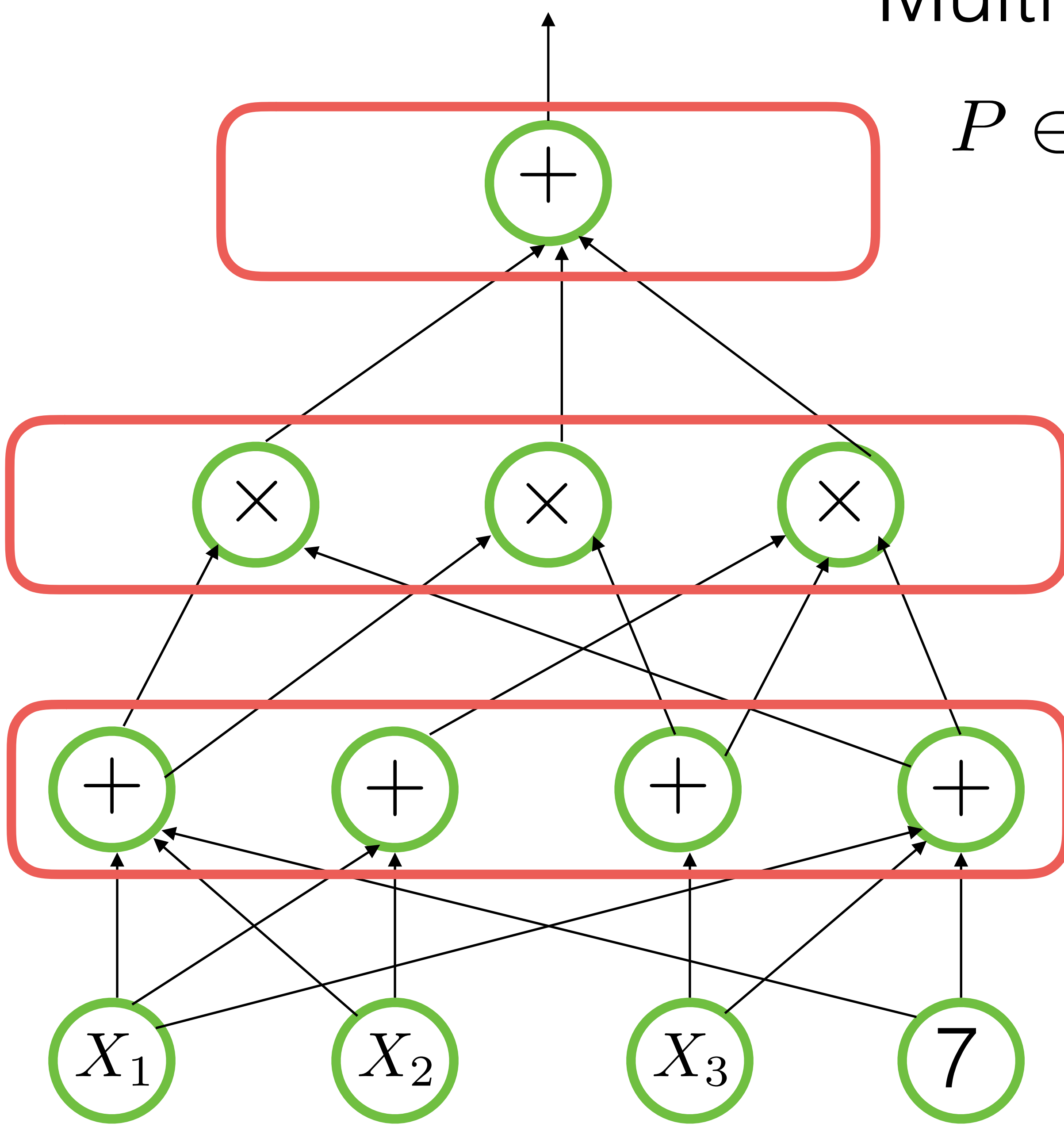
$P \in \mathbb{F}[X_1, X_2, \ldots, X_n]$



Depth - 3

Size - 8

Arithmetic circuits



Multivariate polynomial

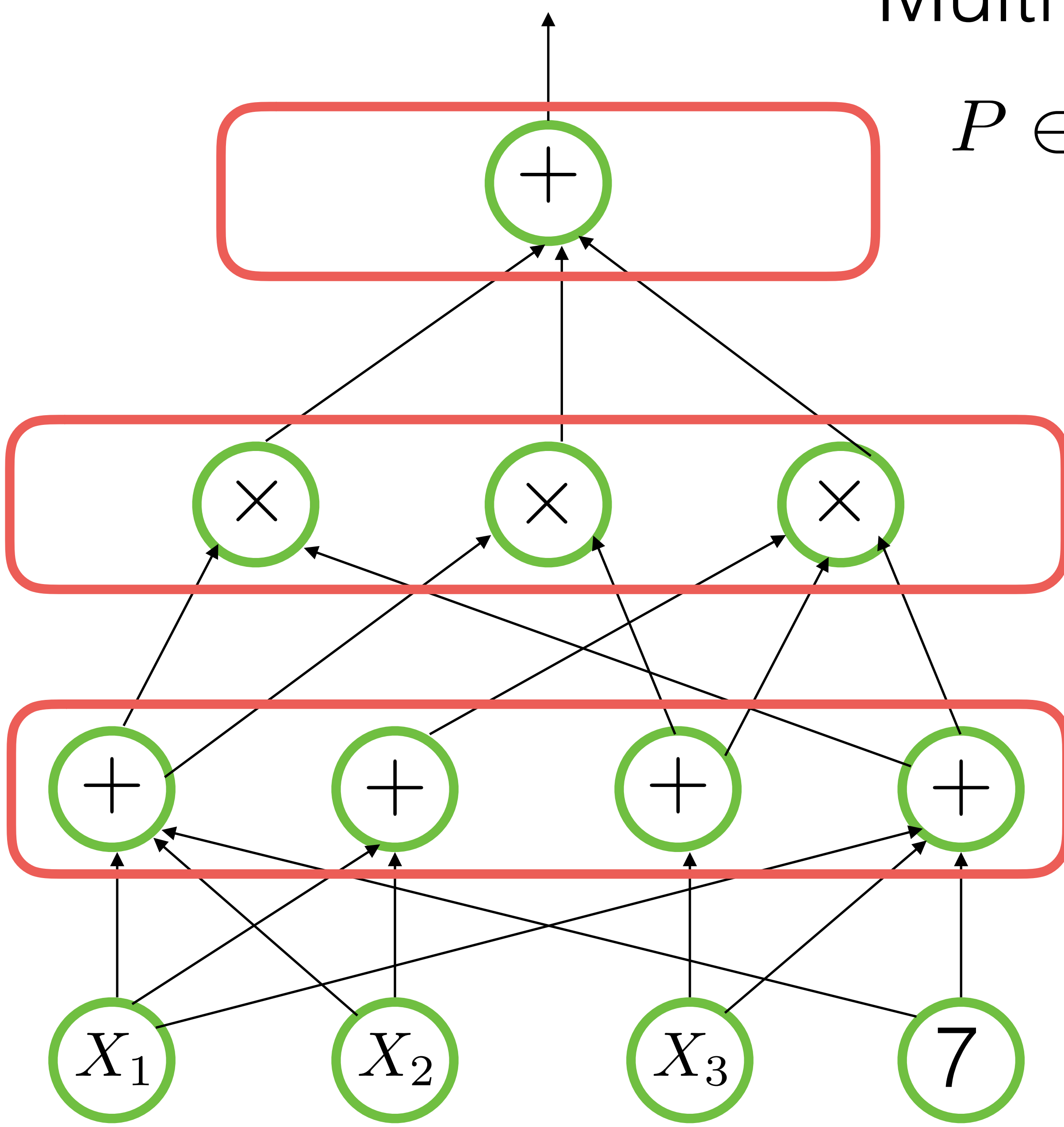
$$P \in \mathbb{F}[X_1, X_2, \ldots, X_n]$$

Depth - 3

Size - 8

An underlying field

Arithmetic circuits



Multivariate polynomial

$P \in \mathbb{F}[X_1, X_2, \ldots, X_n]$

Depth - 3

Size - 8

An underlying field

A circuit is called a formula if the underlying graph is a tree.

Arithmetic circuits

Arithmetic circuits

- Succinctly encode multivariate polynomials

Arithmetic circuits

- Succinctly encode multivariate polynomials
- Evaluation, Sum, Product etc are easy

Arithmetic circuits

- Succinctly encode multivariate polynomials
- Evaluation, Sum, Product etc are easy
- Identity Testing is Efficient (with randomness)

Arithmetic circuits

- Succinctly encode multivariate polynomials
- Evaluation, Sum, Product etc are easy
- Identity Testing is Efficient (with randomness)
- Can efficiently extract low degree components, can compute first order derivatives

Arithmetic circuits

- Succinctly encode multivariate polynomials
- Evaluation, Sum, Product etc are easy
- Identity Testing is Efficient (with randomness)
- Can efficiently extract low degree components, can compute first order derivatives
- Most natural algorithms for computing polynomials are in fact arithmetic circuits for computing them

Arithmetic circuits

- Succinctly encode multivariate polynomials
- Evaluation, Sum, Product etc are easy
- Identity Testing is Efficient (with randomness)
- Can efficiently extract low degree components, can compute first order derivatives
- Most natural algorithms for computing polynomials are in fact arithmetic circuits for computing them

But, can we compute their factors efficiently ?

Polynomial Factorization

Polynomial Factorization

Given an arithmetic circuit C, n-variate, degree d = poly(n), size(C) = poly(n), output the circuits for irreducible factors of C.

Polynomial Factorization

Given an arithmetic circuit C, n-variate, degree d = poly(n), size(C) = poly(n), output the circuits for irreducible factors of C.

Polynomial Factorization

Given an arithmetic circuit C, n-variate, degree d = poly(n), size(C) = poly(n), output the circuits for irreducible factors of C.

Is this even reasonable : is the output of size poly(n) ?

Polynomial Factorization

Given an arithmetic circuit C, n-variate, degree d = poly(n), size(C) = poly(n), output the circuits for irreducible factors of C.

Is this even reasonable : is the output of size poly(n) ?

Do factors of polynomials with 'small' arithmetic circuits have 'small' arithmetic circuits ?

Polynomial Factorization

Given an arithmetic circuit C, n-variate, degree d = poly(n), size(C) = poly(n), output the circuits for irreducible factors of C.

Is this even reasonable : is the output of size poly(n) ?

Do factors of polynomials with 'small' arithmetic circuits have 'small' arithmetic circuits ?

Not true for sparse representation!

Factors of sparse polynomials

$$P = y \prod_{i \in [n]} (X_i - 1) + \prod_{i \in [n]} (X_i^n - 1)$$

Factors of sparse polynomials

$$P = y \prod_{i \in [n]} (X_i - 1) + \prod_{i \in [n]} (X_i^n - 1)$$

$$P = \prod_{i \in [n]} (X_i - 1) \left(y + \prod_{i \in [n]} \left(X_i^{n-1} + X_i^{n-2} + \cdots + 1 \right) \right)$$

Factors of sparse polynomials

$$P = y \prod_{i \in [n]} (X_i - 1) + \prod_{i \in [n]} (X_i^n - 1)$$

$$P = \prod_{i \in [n]} (X_i - 1) \left(y + \prod_{i \in [n]} \left(X_i^{n-1} + X_i^{n-2} + \cdots + 1 \right) \right)$$

Sparsity of P = s = $O(2^n)$

Factors of sparse polynomials

$$P = y \prod_{i \in [n]} (X_i - 1) + \prod_{i \in [n]} (X_i^n - 1)$$

$$P = \prod_{i \in [n]} (X_i - 1) \left(y + \prod_{i \in [n]} \left(X_i^{n-1} + X_i^{n-2} + \cdots + 1 \right) \right)$$

Sparsity of P = s = $O(2^n)$

Sparsity of the irreducible factor = s' = $\Theta(n^n)$ >>> poly(s)

Factors of sparse polynomials

$$P = y \prod_{i \in [n]} (X_i - 1) + \prod_{i \in [n]} (X_i^n - 1)$$

$$P = \prod_{i \in [n]} (X_i - 1) \left(y + \prod_{i \in [n]} \left(X_i^{n-1} + X_i^{n-2} + \cdots + 1 \right) \right)$$

Sparsity of P = s = $O(2^n)$

Sparsity of the irreducible factor = s' = $\Theta(n^n)$ >>> poly(s)

Another reason why this representation is not so nice…

Polynomial Factorization

Given an arithmetic circuit C, n-variate, degree d = poly(n), size(C) = poly(n), output the circuits for irred. factors of C.

Polynomial Factorization

Given an arithmetic circuit C, n-variate, degree d = poly(n), size(C) = poly(n), output the circuits for irred. factors of C.

- Does the factor even have a small circuit ?

Polynomial Factorization

Given an arithmetic circuit C, n-variate, degree d = poly(n), size(C) = poly(n), output the circuits for irred. factors of C.

- Does the factor even have a small circuit ?

Theorem [Kaltofen]

Let P be an n-variate polynomial of degree d, which can be computed by a size s circuit. Then, any factor of P can be computed by a circuit of size poly(s, n, d).

Polynomial Factorization

Given an arithmetic circuit C, n-variate, degree d = poly(n), size(C) = poly(n), output the circuits for irred. factors of C.

- Does the factor even have a small circuit ?

Theorem [Kaltofen]

Let P be an n-variate polynomial of degree d, which can be computed by a size s circuit. Then, any factor of P can be computed by a circuit of size poly(s, n, d).

Moreover, a circuit for the factors can be computed in time poly(s, n, d) with randomness, even with only query access to circuit for P.

Polynomial Factorization

Given an arithmetic circuit C, n-variate, degree d = poly(n), size(C) = poly(n), output the circuits for irred. factors of C.

- Does the factor even have a small circuit ?

Theorem [Kaltofen]

Let P be an n-variate polynomial of degree d, which can be computed by a size s circuit. Then, any factor of P can be computed by a circuit of size poly(s, n, d).

Moreover, a circuit for the factors can be computed in time poly(s, n, d) with randomness, even with only query access to circuit for P.

The complexity class VP is uniformly closed under taking factors.

Detour : Algebraic P and NP

VP - algebraic P

n-variate polynomials of degree d = poly(n) which can be computed by circuits of size poly(n)

VP - algebraic P

n-variate polynomials of degree d = poly(n) which can be computed by circuits of size poly(n)

Determinant = $\sum_{\pi \in S_m} (-1)^{\mathrm{sign}(\pi)} \prod_{j=1}^{m} X_{j,\pi(j)}$

VP - algebraic P

n-variate polynomials of degree d = poly(n) which can be computed by circuits of size poly(n)

Determinant = $\sum_{\pi \in S_m} (-1)^{\mathrm{sign}(\pi)} \prod_{j=1}^{m} X_{j,\pi(j)}$

[Csanky, Berkowitz] Determinant is in VP.

VP - algebraic P

n-variate polynomials of degree d = poly(n) which can be computed by circuits of size poly(n)

Determinant = $\sum_{\pi \in S_m} (-1)^{\mathrm{sign}(\pi)} \prod_{j=1}^{m} X_{j,\pi(j)}$

[Csanky, Berkowitz] Determinant is in VP.

Esym(m, d) = $\sum_{S \in \binom{[m]}{d}} \prod_{j \in S} X_j$

VP - algebraic P

n-variate polynomials of degree d = poly(n) which can be computed by circuits of size poly(n)

Determinant = $\sum_{\pi \in S_m} (-1)^{\mathrm{sign}(\pi)} \prod_{j=1}^{m} X_{j,\pi(j)}$

[Csanky, Berkowitz] Determinant is in VP.

Esym(m, d) = $\sum_{S \in \binom{[m]}{d}} \prod_{j \in S} X_j$

[Folklore, Ben-Or] Esym(m,d) is in VP.

VNP - algebraic NP

n-variate polynomials of degree $d = \text{poly}(n)$ which are 'explicit'

VNP - algebraic NP

n-variate polynomials of degree $d = \text{poly}(n)$ which are 'explicit' - coefficient of any monomial can be efficiently determined

VNP - algebraic NP

n-variate polynomials of degree d = poly(n) which are 'explicit' - coefficient of any monomial can be efficiently determined

Permanent = $\sum_{\pi \in S_m} \prod_{j=1}^{m} X_{j,\pi(j)}$

VNP - algebraic NP

n-variate polynomials of degree d = poly(n) which are 'explicit' - coefficient of any monomial can be efficiently determined

Permanent = $\sum_{\pi \in S_m} \prod_{j=1}^{m} X_{j,\pi(j)}$

[Valiant] Permanent is complete for VNP.

Algebraic P vs Algebraic NP

Algebraic P vs Algebraic NP

- Valiant's hypothesis : VNP is not contained in VP.

Algebraic P vs Algebraic NP

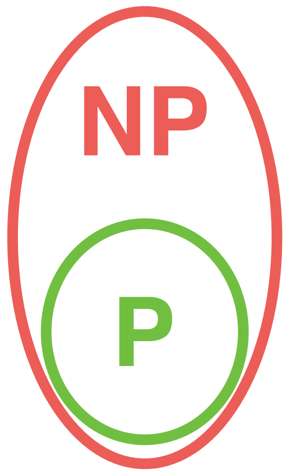
- Valiant's hypothesis : VNP is not contained in VP.
- In particular, he conjectured that Permanent does not have poly(m) sized arithmetic circuits.

Algebraic P vs Algebraic NP

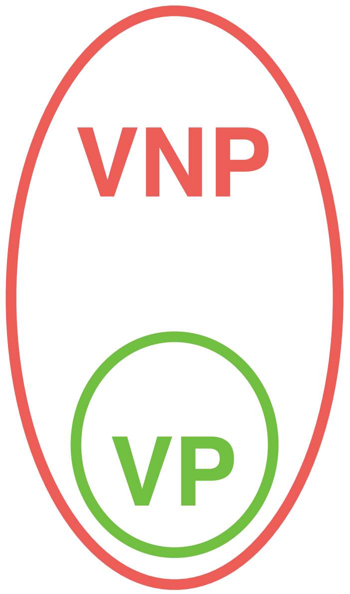
- Valiant's hypothesis : VNP is not contained in VP.
- In particular, he conjectured that Permanent does not have poly(m) sized arithmetic circuits.
- Algebraic analogue of the P vs NP question.

Cook's vs Valiant's hypothesis

P vs NP

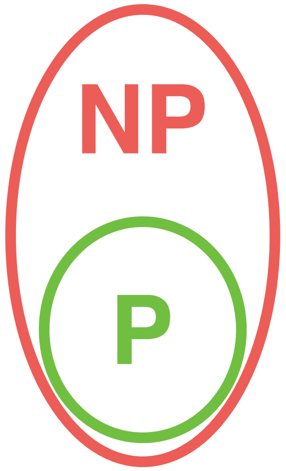


VP vs VNP

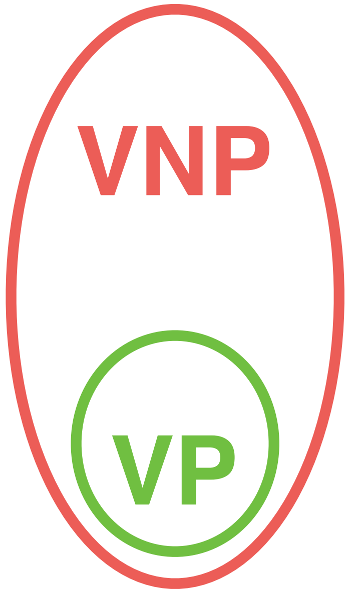


Cook's vs Valiant's hypothesis

P vs NP



VP vs VNP



[Burgisser] Under GRH, VP = VNP implies non-uniform P = non-uniform NP.

Algebraic P vs Algebraic NP

Are there explicit polynomial families which cannot be computed by polynomial sized arithmetic circuits ?

Why do we care

Why do we care

- A fundamental question in computer science.

Why do we care

- A fundamental question in computer science.
- Necessary for P vs NP, and potentially easier.

Why do we care

- A fundamental question in computer science.
- Necessary for P vs NP, and potentially easier.
- Applications to derandomization.

Closure under Factorization

Polynomial Factorization

Theorem [Kaltofen]

Let P be an n-variate polynomial of degree d, which can be computed by a size s circuit. Then, any factor of P can be computed by a circuit of size poly(s, n, d).

Moreover, a circuit for the factors can be computed in time poly(s, n, d) with randomness, even with only query access to circuit for P.

The complexity class VP is uniformly closed under taking factors!

What about closure of other classes ?

- If a polynomial is in VNP, are the factors in VNP ?

- If a polynomial has small formulas, do its factors have small formulas ?

- If a polynomial has small constant depth circuits, do the factors have small constant depth circuits ?

Why do we care ?

Why do we care ?

- Closure under taking factors is a natural algebraic requirement, which any algebraically nice model of computation would have. So, natural to ask.

Why do we care ?

- Closure under taking factors is a natural algebraic requirement, which any algebraically nice model of computation would have. So, natural to ask.

- If VNP is not closed under taking factors, then VP is different from VNP.

Why do we care ?

- Closure under taking factors is a natural algebraic requirement, which any algebraically nice model of computation would have. So, natural to ask.

- If VNP is not closed under taking factors, then VP is different from VNP.

- The road from Hardness to Randomness goes via polynomial factorization.

Why do we care ?

- Closure under taking factors is a natural algebraic requirement, which any algebraically nice model of computation would have. So, natural to ask.
- If VNP is not closed under taking factors, then VP is different from VNP.
- The road from Hardness to Randomness goes via polynomial factorization.

Conjecture [Burgisser]

The complexity class VNP is closed under taking factors.

Results I : Closure results

Factors of polynomials in VNP

Factors of polynomials in VNP

Theorem [Chou-K-Solomon]

The complexity class VNP is closed under taking factors.

Factors of polynomials in VNP

Theorem [Chou-K-Solomon]

The complexity class VNP is closed under taking factors.

$$P = \sum_{\mathbf{Y} \in \{0,1\}^m} Q(\mathbf{X}, \mathbf{Y})$$

Factors of polynomials in VNP

Theorem [Chou-K-Solomon]

The complexity class VNP is closed under taking factors.

$$\mathrm{size}(Q) = \mathrm{poly}(n) = s$$

$$P = \sum_{\mathbf{Y} \in \{0,1\}^m} Q(\mathbf{X}, \mathbf{Y})$$

Factors of polynomials in VNP

Theorem [Chou-K-Solomon]

The complexity class VNP is closed under taking factors.

$$P = \sum_{\mathbf{Y} \in \{0,1\}^m} Q(\mathbf{X}, \mathbf{Y})$$

$$\text{size}(Q) = \text{poly}(n) = s$$

$$m = \text{poly}(n)$$

Factors of polynomials in VNP

Theorem [Chou-K-Solomon]

The complexity class VNP is closed under taking factors.

$$P = \sum_{\mathbf{Y} \in \{0,1\}^m} Q(\mathbf{X}, \mathbf{Y})$$

$$\text{size}(Q) = \text{poly}(n) = s$$

$$m = \text{poly}(n)$$

f is a factor of P

Factors of polynomials in VNP

Theorem [Chou-K-Solomon]

The complexity class VNP is closed under taking factors.

$$P = \sum_{\mathbf{Y} \in \{0,1\}^m} Q(\mathbf{X}, \mathbf{Y})$$

$$\text{size}(Q) = \text{poly}(n) = s$$

$$m = \text{poly}(n)$$

f is a factor of P

$$f = \sum_{\mathbf{Y} \in \{0,1\}^{m'}} Q'(\mathbf{X}, \mathbf{Y})$$

Factors of polynomials in VNP

Theorem [Chou-K-Solomon]

The complexity class VNP is closed under taking factors.

$$P = \sum_{\mathbf{Y} \in \{0,1\}^m} Q(\mathbf{X}, \mathbf{Y})$$

$$\text{size}(Q) = \text{poly}(n) = s$$

$$m = \text{poly}(n)$$

f is a factor of P

$$\text{size}(Q') = \text{poly}(n, s)$$

$$f = \sum_{\mathbf{Y} \in \{0,1\}^{m'}} Q'(\mathbf{X}, \mathbf{Y})$$

Factors of polynomials in VNP

Theorem [Chou-K-Solomon]

The complexity class VNP is closed under taking factors.

$$P = \sum_{\mathbf{Y} \in \{0,1\}^m} Q(\mathbf{X}, \mathbf{Y})$$

$\text{size}(Q) = \text{poly}(n) = s$

$m = \text{poly}(n)$

f is a factor of P

$$f = \sum_{\mathbf{Y} \in \{0,1\}^{m'}} Q'(\mathbf{X}, \mathbf{Y})$$

$\text{size}(Q') = \text{poly}(n, s)$

$m' = \text{poly}(n, s)$

Factors of polynomials in VNP

Theorem [Chou-K-Solomon]

The complexity class VNP is closed under taking factors.

$$P = \sum_{\mathbf{Y} \in \{0,1\}^m} Q(\mathbf{X}, \mathbf{Y})$$

$$\text{size}(Q) = \text{poly}(n) = s$$

$$m = \text{poly}(n)$$

f is a factor of P

$$f = \sum_{\mathbf{Y} \in \{0,1\}^{m'}} Q'(\mathbf{X}, \mathbf{Y})$$

$$\text{size}(Q') = \text{poly}(n, s)$$

$$m' = \text{poly}(n, s)$$

Improves a quasi-polynomial upper bound of Dutta-Saxena-Sinhababu.

Factors of polynomials with small formulas

Factors of polynomials with small formulas

Theorem [Chou-K-Solomon, Dutta-Saxena-Sinhababu]

Let P be an n-variate degree D polynomial computable by a formula of size s, and let f be a factor of degree d of P. Then, f can be computed by a formula of size $\mathbf{d}^{\mathbf{O}(\log \mathbf{d})} \cdot \mathrm{poly}(\mathbf{n}, \mathbf{s}, \mathbf{D})$

Factors of polynomials with small formulas

Theorem [Chou-K-Solomon, Dutta-Saxena-Sinhababu]

Let P be an n-variate degree D polynomial computable by a formula of size s, and let f be a factor of degree d of P. Then, f can be computed by a formula of size
$$\mathbf{d}^{\mathbf{O}(\log \mathbf{d})} \cdot \mathrm{poly}(\mathbf{n}, \mathbf{s}, \mathbf{D})$$

For low, but growing degree factors, this is still poly(n).

Factors of polynomials with shallow circuits

Factors of polynomials with shallow circuits

Theorem [Chou-K-Solomon]

Let P be an n-variate degree D polynomial computable by a depth k circuit of size s, and let f be a factor of degree d of P. Then, f can be computed by depth k + O(1) circuits of size $\mathbf{d}^{\mathbf{O}(\mathbf{d}^{\epsilon})} \cdot \mathrm{poly}(\mathbf{n}, \mathbf{s}, \mathbf{D})$

Factors of polynomials with shallow circuits

Theorem [Chou-K-Solomon]

Let P be an n-variate degree D polynomial computable by a depth k circuit of size s, and let f be a factor of degree d of P. Then, f can be computed by depth k + O(1) circuits of size $\mathbf{d^{O(d^{\epsilon})}} \cdot \mathrm{poly}(\mathbf{n}, \mathbf{s}, \mathbf{D})$

Again, for low, but growing degree factors, this is still poly(n).

Factors of polynomials with shallow circuits

Theorem [Chou-K-Solomon]

Let P be an n-variate degree D polynomial computable by a depth k circuit of size s, and let f be a factor of degree d of P. Then, f can be computed by depth k + O(1) circuits of size $\mathbf{d}^{\mathbf{O}(\mathbf{d}^{\epsilon})} \cdot \mathrm{poly}(\mathbf{n}, \mathbf{s}, \mathbf{D})$

Again, for low, but growing degree factors, this is still poly(n).

A bound of $\mathbf{n}^{\mathbf{O}(\mathbf{d}^{\epsilon})} \cdot \mathrm{poly}(\mathbf{n}, \mathbf{s}, \mathbf{D})$ **follows from Kaltofen's result and standard structure theorems, but this is not poly(n,s) as long as d is growing.**

Results II : Applications to Hardness vs Randomness

Hardness vs Randomness

Hardness vs Randomness

Hardness vs Randomness

Two fundamental family of questions in computational complexity

Hardness vs Randomness

Two fundamental family of questions in computational complexity

Hardness vs Randomness

Two fundamental family of questions in computational complexity

Lower bounds : ‘explicit’ hard functions

Hardness vs Randomness

Two fundamental family of questions in computational complexity

Lower bounds : ‘explicit’ hard functions

Hardness vs Randomness

Two fundamental family of questions in computational complexity

Lower bounds : 'explicit' hard functions

Derandomization : does every problem with an efficient randomized algorithm have an efficient deterministic algorithm

Hardness vs Randomness

Two fundamental family of questions in computational complexity

Lower bounds : ‘explicit’ hard functions

Derandomization : does every problem with an efficient randomized algorithm have an efficient deterministic algorithm

Hardness vs Randomness

Two fundamental family of questions in computational complexity

Lower bounds : 'explicit' hard functions

Derandomization : does every problem with an efficient randomized algorithm have an efficient deterministic algorithm

Seemingly unrelated, but have rich, insightful and deep connections to each other [Nisan, Wigderson, Impagliazzo, Kabanets….]

Hardness vs Randomness

Two fundamental family of questions in computational complexity

Lower bounds : 'explicit' hard functions

Derandomization : does every problem with an efficient randomized algorithm have an efficient deterministic algorithm

Seemingly unrelated, but have rich, insightful and deep connections to each other [Nisan, Wigderson, Impagliazzo, Kabanets….]

Hardness vs Randomness

Two fundamental family of questions in computational complexity

Lower bounds : ‘explicit’ hard functions

Derandomization : does every problem with an efficient randomized algorithm have an efficient deterministic algorithm

Seemingly unrelated, but have rich, insightful and deep connections to each other [Nisan, Wigderson, Impagliazzo, Kabanets….]

Here, we focus on this phenomenon for algebraic computation

Hardness vs Randomness

Two fundamental family of questions in computational complexity

Lower bounds : ‘explicit’ hard functions

Derandomization : does every problem with an efficient randomized algorithm have an efficient deterministic algorithm

Seemingly unrelated, but have rich, insightful and deep connections to each other [Nisan, Wigderson, Impagliazzo, Kabanets….]

Here, we focus on this phenomenon for algebraic computation

Hardness vs Randomness

Two fundamental family of questions in computational complexity

Lower bounds : ‘explicit’ hard functions

Derandomization : does every problem with an efficient randomized algorithm have an efficient deterministic algorithm

Seemingly unrelated, but have rich, insightful and deep connections to each other [Nisan, Wigderson, Impagliazzo, Kabanets….]

Here, we focus on this phenomenon for algebraic computation

Hardness vs Randomness

Two fundamental family of questions in computational complexity

Lower bounds : ‘explicit’ hard functions

Derandomization : does every problem with an efficient randomized algorithm have an efficient deterministic algorithm

Seemingly unrelated, but have rich, insightful and deep connections to each other [Nisan, Wigderson, Impagliazzo, Kabanets….]

Here, we focus on this phenomenon for algebraic computation

Polynomial Identity Testing

Polynomial Identity Testing

Input : An arithmetic circuit C of size s, degree d in n variables.

Polynomial Identity Testing

Input : An arithmetic circuit C of size s, degree d in n variables.

Question : Is the polynomial computed by C identically zero ?

Polynomial Identity Testing

Input : An arithmetic circuit C of size s, degree d in n variables.

Question : Is the polynomial computed by C identically zero ?

Polynomial Identity Testing

Input : An arithmetic circuit C of size s, degree d in n variables.

Question : Is the polynomial computed by C identically zero ?

A natural question on its own, but some unexpected and remarkable connections to lower bounds and algorithm design.

A simple randomized algorithm

Lemma [Ore, Schwartz, Zippel, DeMillo, Lipton]

Let S be a subset of the field. Then,

$$\Pr_{\mathbf{a}\in\mathbf{S}^{\mathbf{n}}}[\mathbf{C}(\mathbf{a}) = \mathbf{0}] \leq \frac{\mathbf{d}}{|\mathbf{S}|}$$

A simple randomized algorithm

Lemma [Ore, Schwartz, Zippel, DeMillo, Lipton]

Let S be a subset of the field. Then,

$$\Pr_{\mathbf{a} \in \mathbf{S}^{\mathbf{n}}}[\mathbf{C}(\mathbf{a}) = \mathbf{0}] \leq \frac{\mathbf{d}}{|\mathbf{S}|}$$

So, querying the circuit at a random point from a large enough grid works with high probability.

A simple randomized algorithm

Lemma [Ore, Schwartz, Zippel, DeMillo, Lipton]

Let S be a subset of the field. Then,

$$\Pr_{\mathbf{a} \in \mathbf{S}^{\mathbf{n}}}[\mathbf{C}(\mathbf{a}) = \mathbf{0}] \leq \frac{\mathbf{d}}{|\mathbf{S}|}$$

So, querying the circuit at a random point from a large enough grid works with high probability.

And, we didn't even have to look inside the circuit.

A simple randomized algorithm

Lemma [Ore, Schwartz, Zippel, DeMillo, Lipton]

Let S be a subset of the field. Then,

$$\Pr_{\mathbf{a} \in \mathbf{S}^{\mathbf{n}}}[\mathbf{C}(\mathbf{a}) = \mathbf{0}] \leq \frac{\mathbf{d}}{|\mathbf{S}|}$$

So, querying the circuit at a random point from a large enough grid works with high probability.

And, we didn't even have to look inside the circuit.

Also, gives an *exp(n log d)* time deterministic algorithm. We are interested in doing anything better than this!

Deterministic Polynomial Identity Testing

Deterministic Polynomial Identity Testing

Deterministic polynomial identity testing has some remarkable consequences.

Deterministic Polynomial Identity Testing

Deterministic polynomial identity testing has some remarkable consequences.

Deterministic Polynomial Identity Testing

Deterministic polynomial identity testing has some remarkable consequences.

- Lower bounds closely related to VP vs VNP [Kabanets-Impagliazzo]

Deterministic Polynomial Identity Testing

Deterministic polynomial identity testing has some remarkable consequences.

- Lower bounds closely related to VP vs VNP [Kabanets-Impagliazzo]
- Deterministic polynomial factorization [Kopparty-Saraf-Shpilka]

Deterministic Polynomial Identity Testing

Deterministic polynomial identity testing has some remarkable consequences.

- Lower bounds closely related to VP vs VNP [Kabanets-Impagliazzo]
- Deterministic polynomial factorization [Kopparty-Saraf-Shpilka]
- Deterministic parallel algorithms for bipartite matching [Lovasz, Mulmuley-Vazirani-Vazirani]

Deterministic Polynomial Identity Testing

Deterministic polynomial identity testing has some remarkable consequences.

- Lower bounds closely related to VP vs VNP [Kabanets-Impagliazzo]
- Deterministic polynomial factorization [Kopparty-Saraf-Shpilka]
- Deterministic parallel algorithms for bipartite matching [Lovasz, Mulmuley-Vazirani-Vazirani]
- Deterministic algorithms for many subgraph isomorphism problems

Deterministic Polynomial Identity Testing

Deterministic polynomial identity testing has some remarkable consequences.

- **Lower bounds closely related to VP vs VNP [Kabanets-Impagliazzo]**
- Deterministic polynomial factorization [Kopparty-Saraf-Shpilka]
- Deterministic parallel algorithms for bipartite matching [Lovasz, Mulmuley-Vazirani-Vazirani]
- Deterministic algorithms for many subgraph isomorphism problems

Hardness and Randomness

Hardness and Randomness

Deterministic Polynomial Identity Testing

Hardness and Randomness

Deterministic Polynomial Identity Testing

Explicit Lower Bounds for Arithmetic Circuits

Hardness and Randomness

Deterministic Polynomial Identity Testing

Explicit Lower Bounds for Arithmetic Circuits

Hardness and Randomness

Deterministic Polynomial Identity Testing

derandomization to hardness

Heintz-Schnorr
Kabanets-Impagliazzo

Explicit Lower Bounds for Arithmetic Circuits

Hardness and Randomness

Deterministic Polynomial Identity Testing

derandomization to hardness

Heintz-Schnorr
Kabanets-Impagliazzo

Explicit Lower Bounds for Arithmetic Circuits

Hardness and Randomness

Deterministic Polynomial Identity Testing

derandomization to hardness

Heintz-Schnorr
Kabanets-Impagliazzo

hardness to derandomization

Kabanets-Impagliazzo

Explicit Lower Bounds for Arithmetic Circuits

Hardness and Randomness

Deterministic Polynomial Identity Testing

derandomization to hardness

Heintz-Schnorr
Kabanets-Impagliazzo

hardness to derandomization

Kabanets-Impagliazzo

Explicit Lower Bounds for Arithmetic Circuits

Randomness from hardness

Theorem [Kabanets-Impagliazzo]

Super-polynomial lower bounds for arithmetic circuits imply non-trivial deterministic PIT for polynomial size arithmetic circuits.

Randomness from hardness

Theorem [Kabanets-Impagliazzo]

Super-polynomial lower bounds for arithmetic circuits imply non-trivial deterministic PIT for polynomial size arithmetic circuits.

Crucially, this proof uses Kaltofen's result about closure of VP under factorization,

Randomness from hardness

Theorem [Kabanets-Impagliazzo]

Super-polynomial lower bounds for arithmetic circuits imply non-trivial deterministic PIT for polynomial size arithmetic circuits.

Crucially, this proof uses Kaltofen's result about closure of VP under factorization.

And thus, does not extend to formulas or low depth circuits, where we do not know closure results.

Scaled down versions of this result ?

Randomness from hardness at low depth

Question [Shpilka-Yehudayoff]

Do lower bounds for low depth circuits imply deterministic PIT for them ?

Randomness from hardness at low depth

Question [Shpilka-Yehudayoff]

Do lower bounds for low depth circuits imply deterministic PIT for them ?

Theorem [Dvir-Shpilka-Yehudayoff]

Lower bounds for low depth circuits imply deterministic PIT for low depth circuits with bounded individual degree.

Randomness from hardness at low depth

Theorem [Chou-K-Solomon]

Super-polynomial lower bounds for low depth arithmetic circuits for poly(log n) degree polynomials imply non-trivial deterministic PIT for them.

Randomness from hardness at low depth

Theorem [Chou-K-Solomon]

Super-polynomial lower bounds for low depth arithmetic circuits for poly(log n) degree polynomials imply non-trivial deterministic PIT for them.

Thus, we get rid of the low individual degree assumption of Dvir et al. at the cost of asking for lower bounds for low degree polynomials.

Randomness from hardness at low depth

Theorem [Chou-K-Solomon]

Super-polynomial lower bounds for low depth arithmetic circuits for poly(log n) degree polynomials imply non-trivial deterministic PIT for them.

Thus, we get rid of the low individual degree assumption of Dvir et al. at the cost of asking for lower bounds for low degree polynomials.

For depth k PIT, we need lower bounds for depth k+5 circuits, which as of now, renders this result unusable.

Randomness from hardness for formulas

Theorem [Chou-K-Solomon]

An $\mathbf{n^{2+\epsilon}}$ lower bound for (border of) formulas for a constant degree polynomial implies sub-exponential PIT for linear size formulas.

Randomness from hardness for formulas

Theorem [Chou-K-Solomon]

An $\mathbf{n^{2+\epsilon}}$ lower bound for (border of) formulas for a constant degree polynomial implies sub-exponential PIT for linear size formulas.

Currently, we don't even know non-trivial PIT for linear size depth-4 formulas.

Randomness from hardness for formulas

Theorem [Chou-K-Solomon]

An $\mathbf{n^{2+\epsilon}}$ lower bound for (border of) formulas for a constant degree polynomial implies sub-exponential PIT for linear size formulas.

Currently, we don't even know non-trivial PIT for linear size depth-4 formulas.

Currently, we know $\mathbf{n^{2-\epsilon}}$ lower bounds of this kind.

Randomness from hardness for formulas

Theorem [Chou-K-Solomon]

An $\mathbf{n^{2+\epsilon}}$ lower bound for (border of) formulas for a constant degree polynomial implies sub-exponential PIT for linear size formulas.

Currently, we don't even know non-trivial PIT for linear size depth-4 formulas.

Currently, we know $\mathbf{n^{2-\epsilon}}$ lower bounds of this kind.

So, (seemingly) small improvement in the state of lower bounds for formulas has extremely interesting consequences for the PIT question.

To summarize

To summarize

- VNP (the algebraic analog of NP) is closed under taking factors.

To summarize

- VNP (the algebraic analog of NP) is closed under taking factors.
- Low (but growing) degree factors of small formulas, low depth circuits have small formulas, low depth circuits respectively.

To summarize

- VNP (the algebraic analog of NP) is closed under taking factors.
- Low (but growing) degree factors of small formulas, low depth circuits have small formulas, low depth circuits respectively.
- Even somewhat non-trivial lower bounds for formulas, low depth circuits imply sub exponential time deterministic Identity Testing algorithms for them.

Snippets of the proof

Key lemma : structure of factors

Key lemma : structure of factors

Lemma (informal)

Let P be an n-variate polynomial of degree D, which can be computed by a size s circuit. Let f be a factor of P of degree d.

Key lemma : structure of factors

Lemma (informal)

Let P be an n-variate polynomial of degree D, which can be computed by a size s circuit. Let f be a factor of P of degree d.

Then, there exist polynomials $P_1, P_2, \ldots, P_d$ whose complexity is closely related to that of P

Key lemma : structure of factors

Lemma (informal)

Let P be an n-variate polynomial of degree D, which can be computed by a size s circuit. Let f be a factor of P of degree d.

Then, there exist polynomials $P_1, P_2, \ldots, P_d$ whose complexity is closely related to that of P, and a d-variate polynomial Q of degree d

Key lemma : structure of factors

Lemma (informal)

Let P be an n-variate polynomial of degree D, which can be computed by a size s circuit. Let f be a factor of P of degree d.

Then, there exist polynomials $P_1, P_2, \ldots, P_d$ whose complexity is closely related to that of P, and a d-variate polynomial Q of degree d which is computable by a size poly(d) circuit

Key lemma : structure of factors

Lemma (informal)

Let P be an n-variate polynomial of degree D, which can be computed by a size s circuit. Let f be a factor of P of degree d.

Then, there exist polynomials $P_1, P_2, \ldots, P_d$ whose complexity is closely related to that of P, and a d-variate polynomial Q of degree d which is computable by a size poly(d) circuit, such that $f = Q\left(P_1, P_2, \ldots, P_d\right)$.

Key lemma : structure of factors

Lemma (informal)

Let P be an n-variate polynomial of degree D, which can be computed by a size s circuit. Let f be a factor of P of degree d.

Then, there exist polynomials $P_1, P_2, \ldots, P_d$ whose complexity is closely related to that of P, and a d-variate polynomial Q of degree d which is computable by a size poly(d) circuit, such that $f = Q(P_1, P_2, \ldots, P_d)$.

'Normal form' for factors.

Key lemma : structure of factors

Lemma (informal)

Let P be an n-variate polynomial of degree D, which can be computed by a size s circuit. Let f be a factor of P of degree d.

Then, there exist polynomials $P_1, P_2, \ldots, P_d$ whose complexity is closely related to that of P, and a d-variate polynomial Q of degree d which is computable by a size poly(d) circuit, such that $f = Q\left(P_1, P_2, \ldots, P_d\right)$.

'Normal form' for factors.

Would be helpful in arguing about their structure.

Structure of factors

P, n-variate, degree D size s circuit.

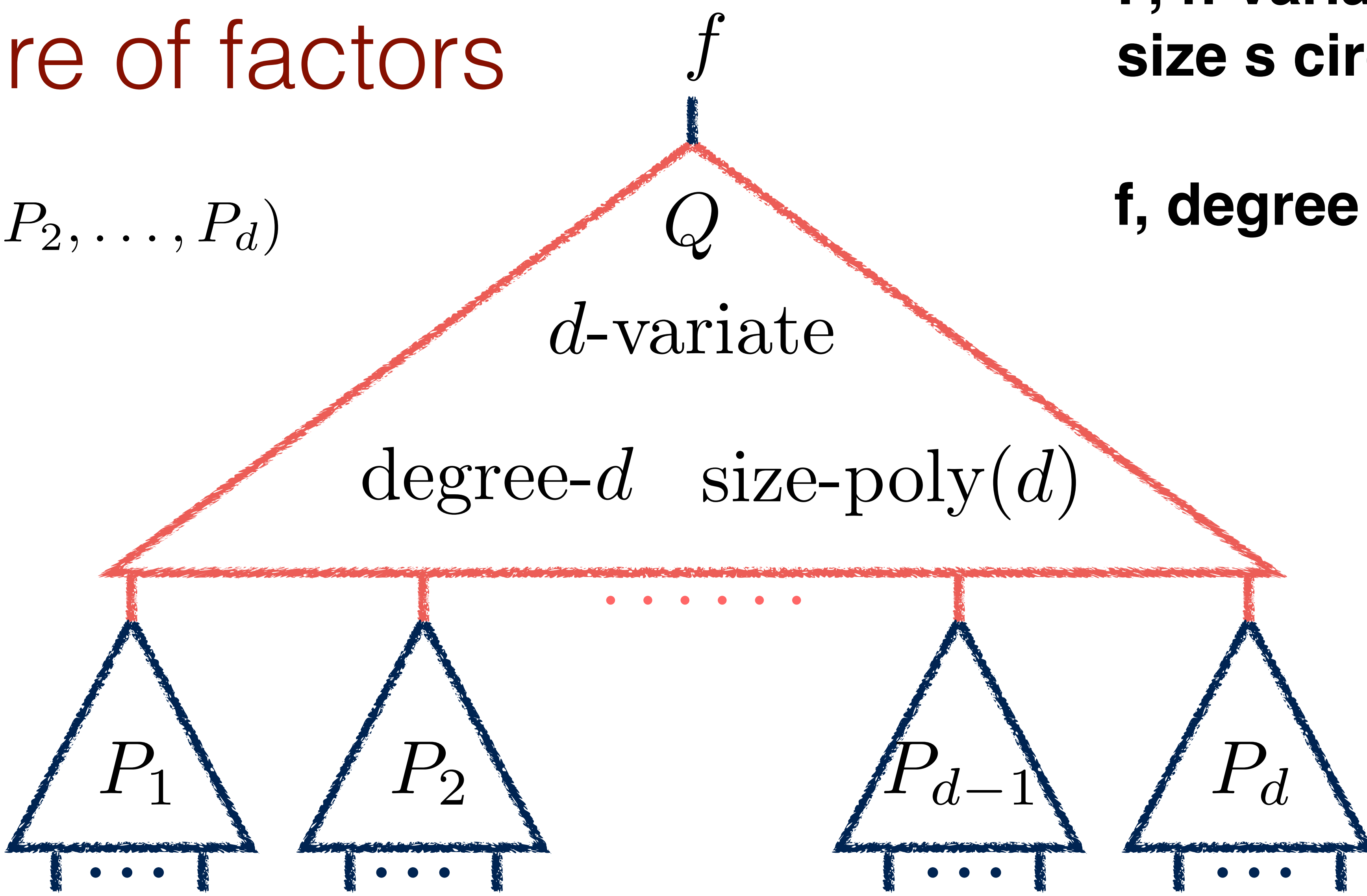
f, degree d factor of P.

Structure of factors

P, n-variate, degree D size s circuit.

f, degree d factor of P.

$$f = Q(P_1, P_2, \ldots, P_d)$$

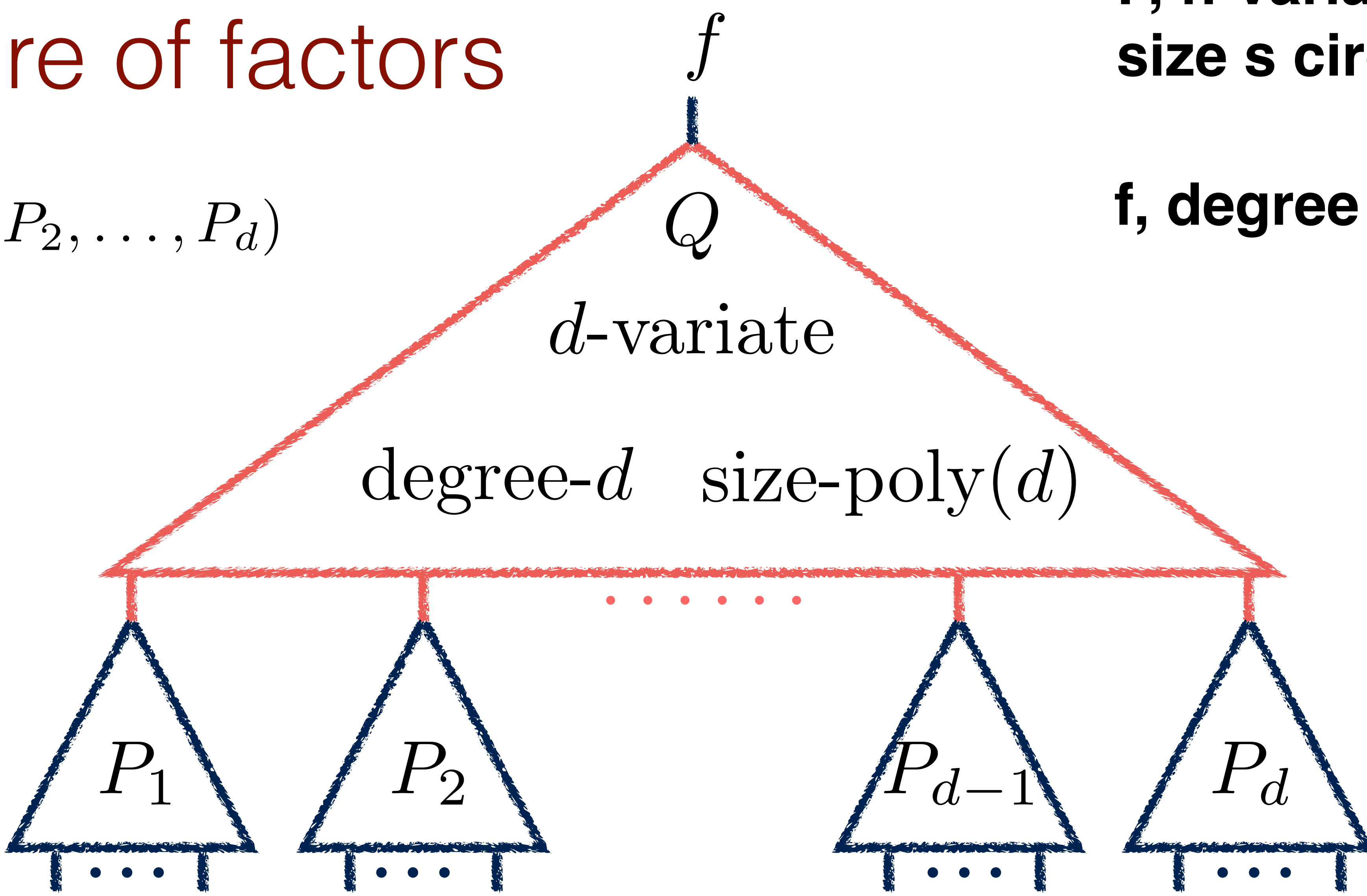


Structure of factors

P, n-variate, degree D size s circuit.

f, degree d factor of P.

$f = Q(P_1, P_2, \ldots, P_d)$



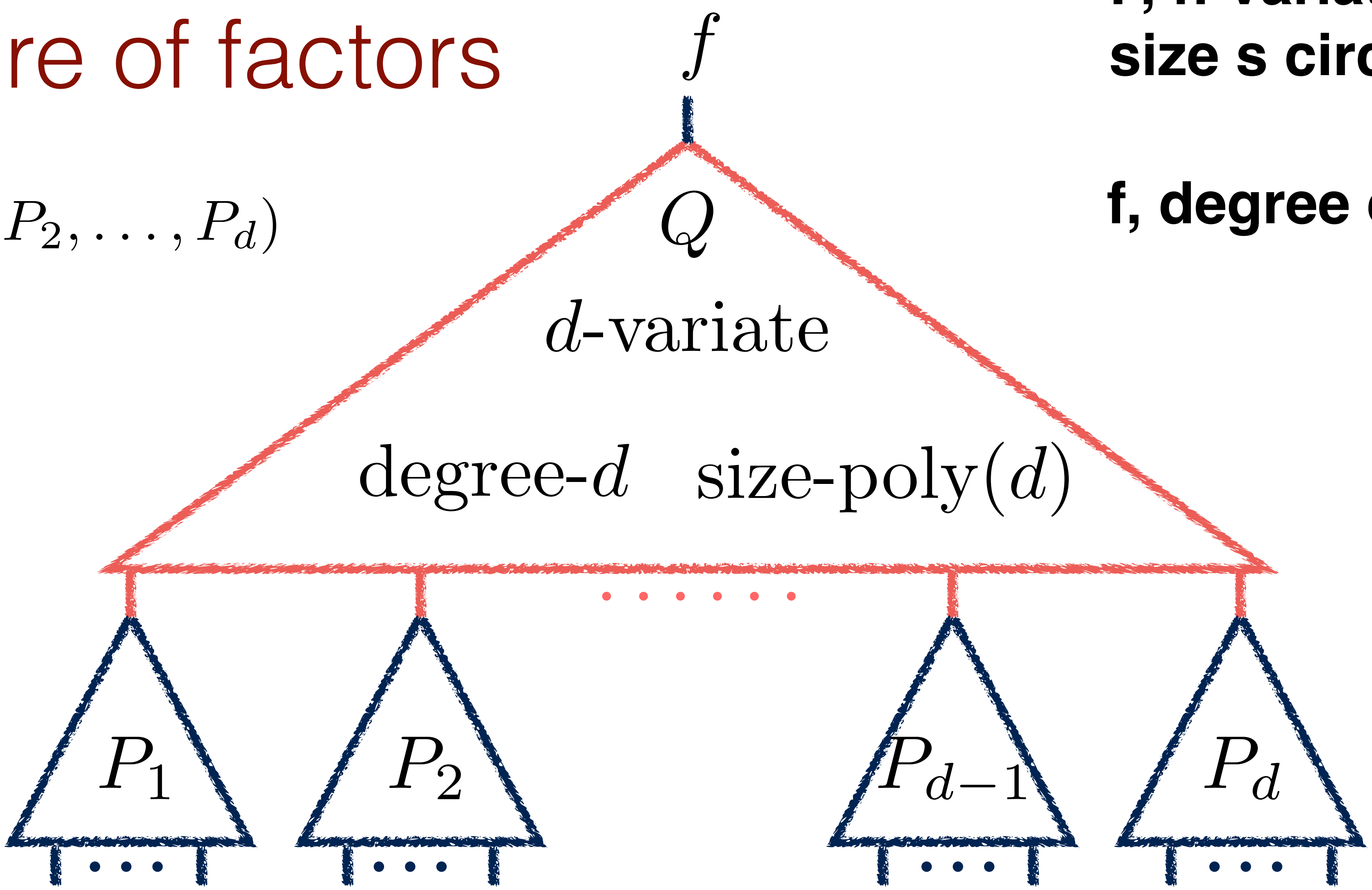
$\text{size}(P_i) = \text{poly}(s, D)$

Structure of factors

P, n-variate, degree D size s circuit.

f, degree d factor of P.

$f = Q(P_1, P_2, \ldots, P_d)$



$\text{size}(P_i) = \text{poly}(s, D)$

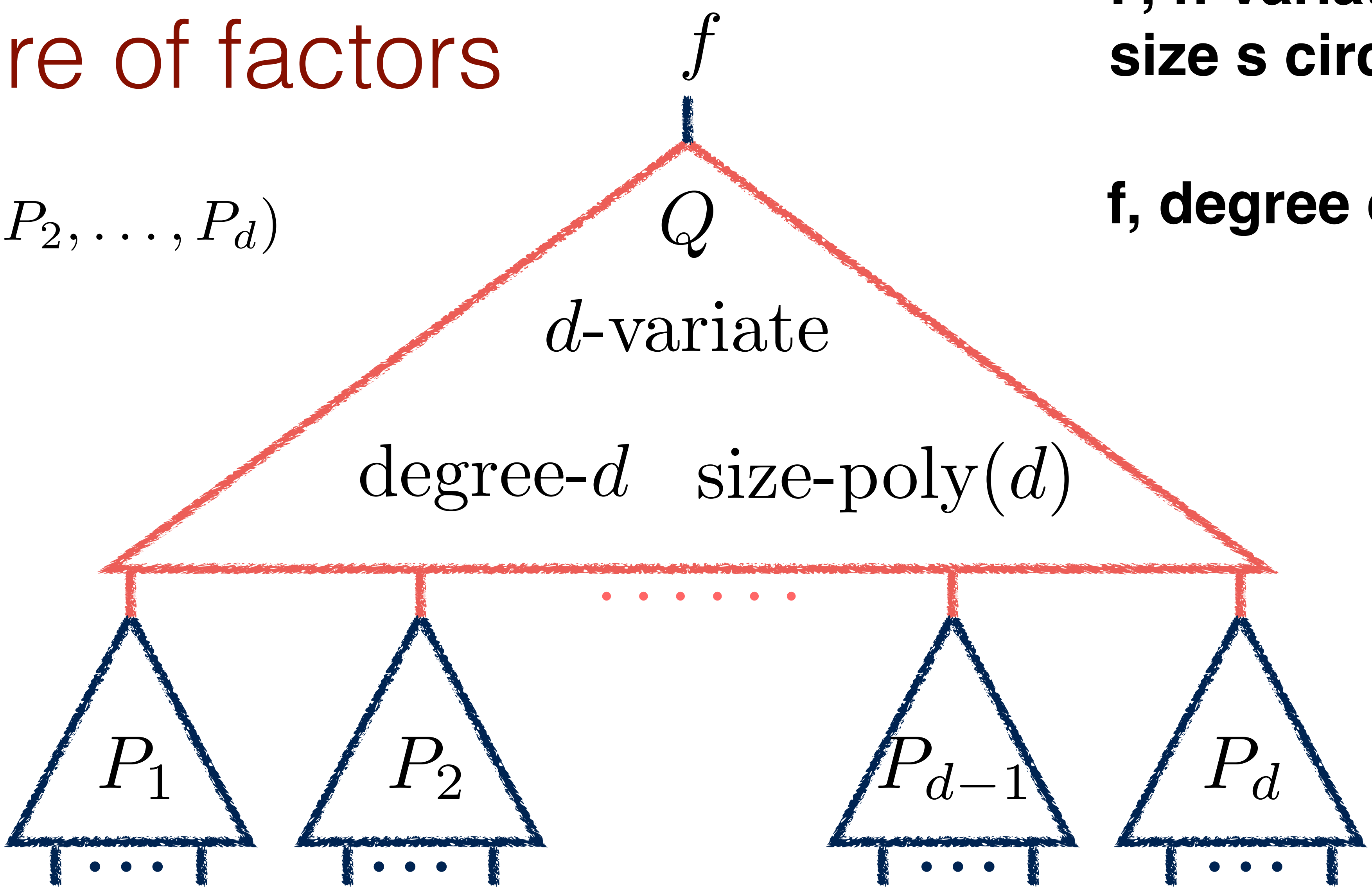
structure preserved

Structure of factors

P, n-variate, degree D size s circuit.

f, degree d factor of P.

$f = Q(P_1, P_2, \ldots, P_d)$



size(P_i) = poly(s, D)

structure preserved

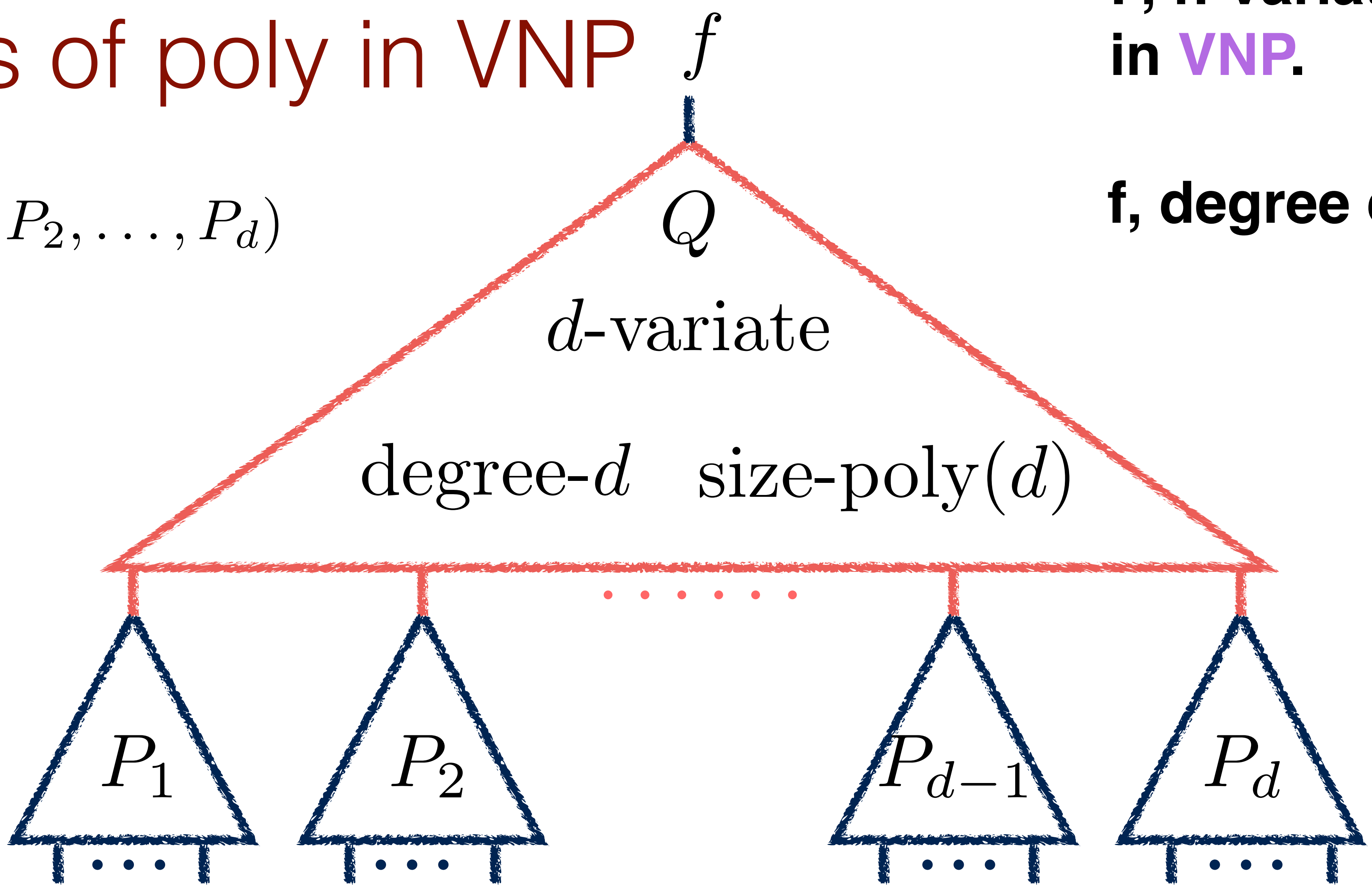
low depth $\rightarrow$ low depth

formula $\rightarrow$ formula

VNP $\rightarrow$ VNP

Factors of poly in VNP

$f = Q(P_1, P_2, \ldots, P_d)$



P, n-variate, degree D in VNP.

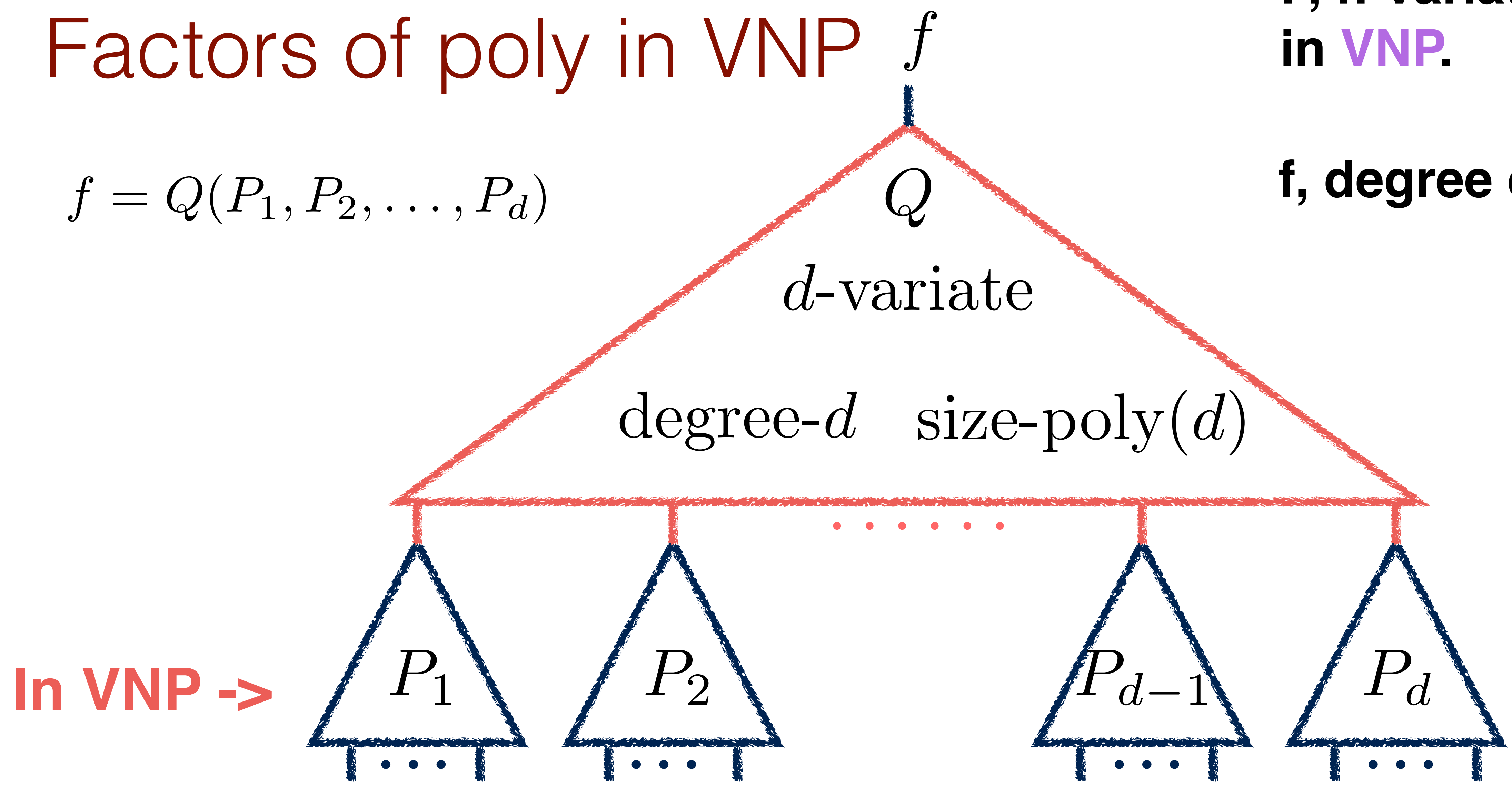
f, degree d factor of P.

Factors of poly in VNP

$f = Q(P_1, P_2, \ldots, P_d)$

P, n-variate, degree D in VNP.

f, degree d factor of P.



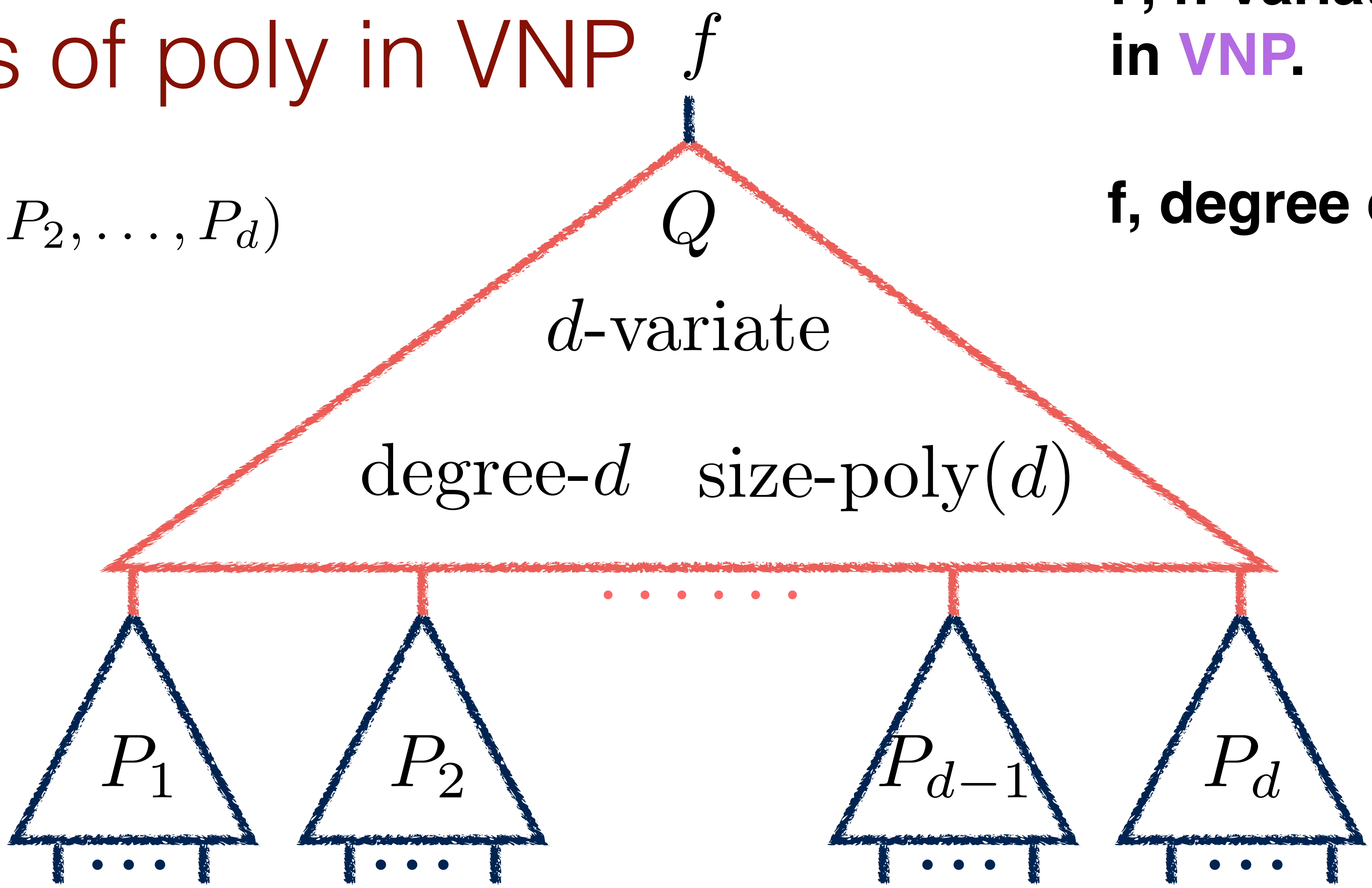
Factors of poly in VNP

P, n-variate, degree D in VNP.

f, degree d factor of P.

$f = Q(P_1, P_2, \ldots, P_d)$

In VNP ->



Theorem [Valiant]

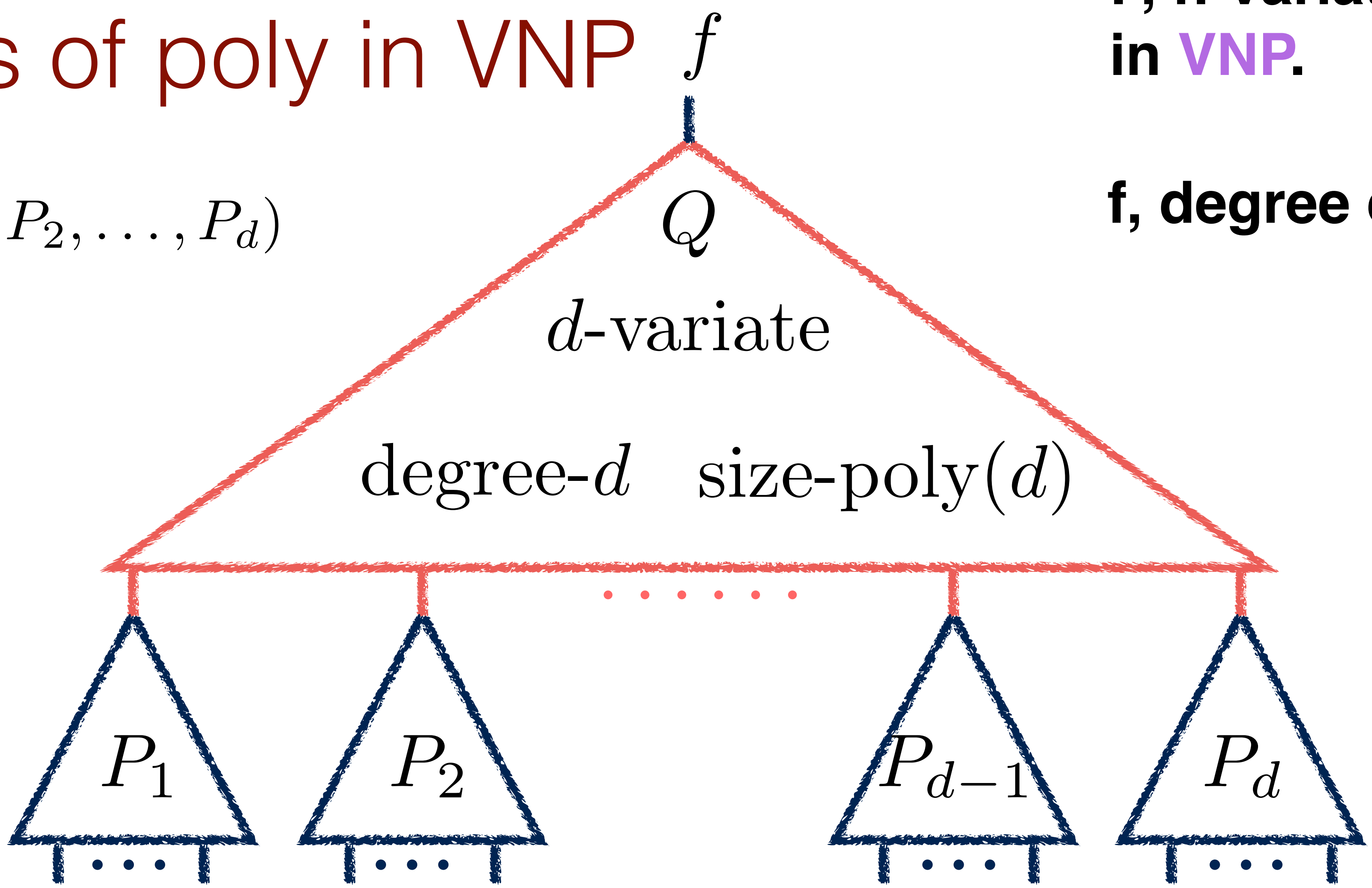
If each $\mathbf{P_i}$ is in VNP, then $\mathbf{Q(P_1, P_2, \ldots, P_d)}$ is in VNP.

Factors of poly in VNP

P, n-variate, degree D in VNP.

f, degree d factor of P.

$f = Q(P_1, P_2, \ldots, P_d)$



In VNP ->

Theorem [Valiant]

If each $\mathbf{P_i}$ is in VNP, then $\mathbf{Q(P_1, P_2, \ldots, P_d)}$ is in VNP.

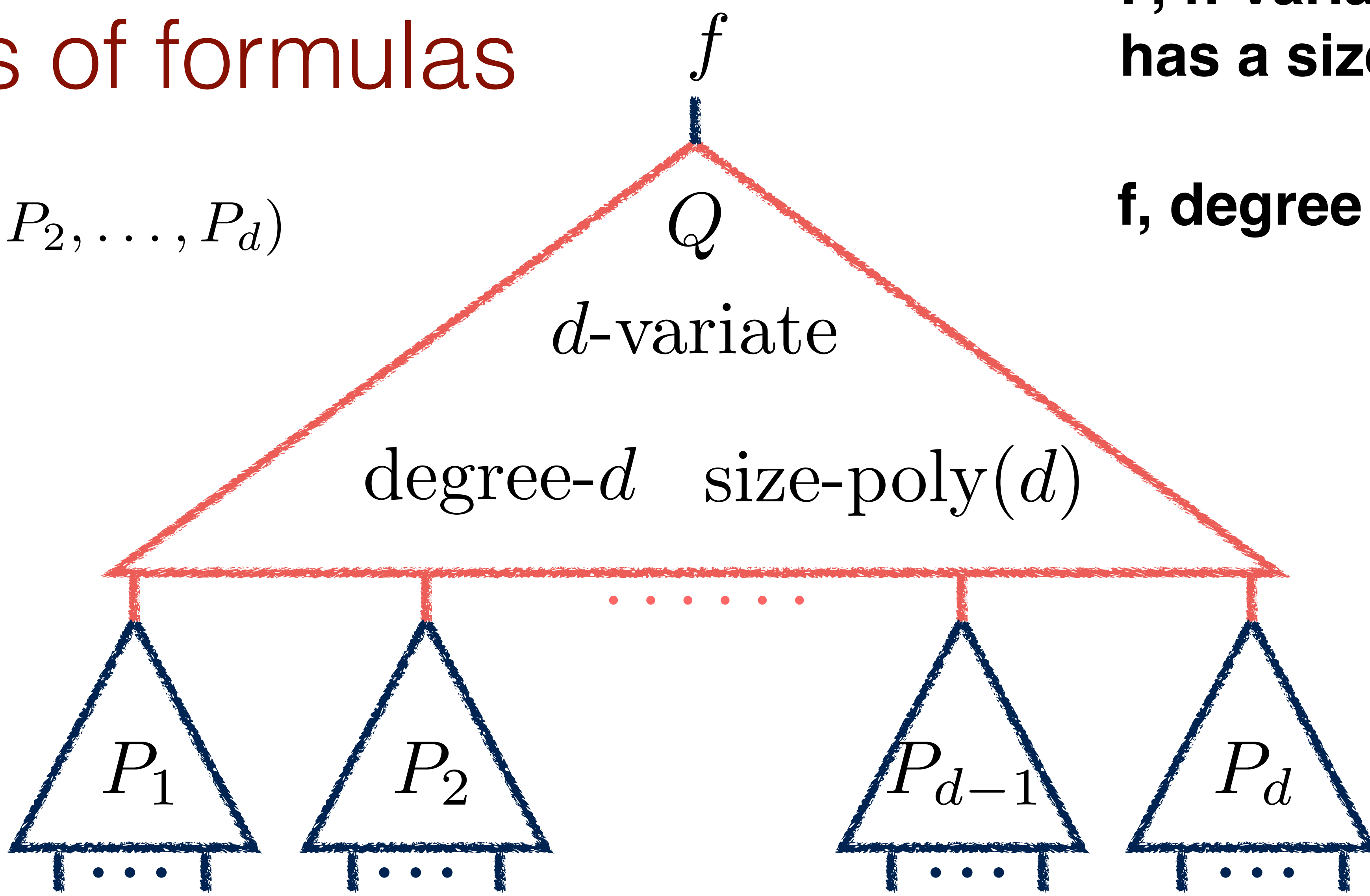
Thus, VNP is closed under taking factors.

Factors of formulas

P, n-variate, degree D has a size s formula.

f, degree d factor of P.

$f = Q(P_1, P_2, \ldots, P_d)$

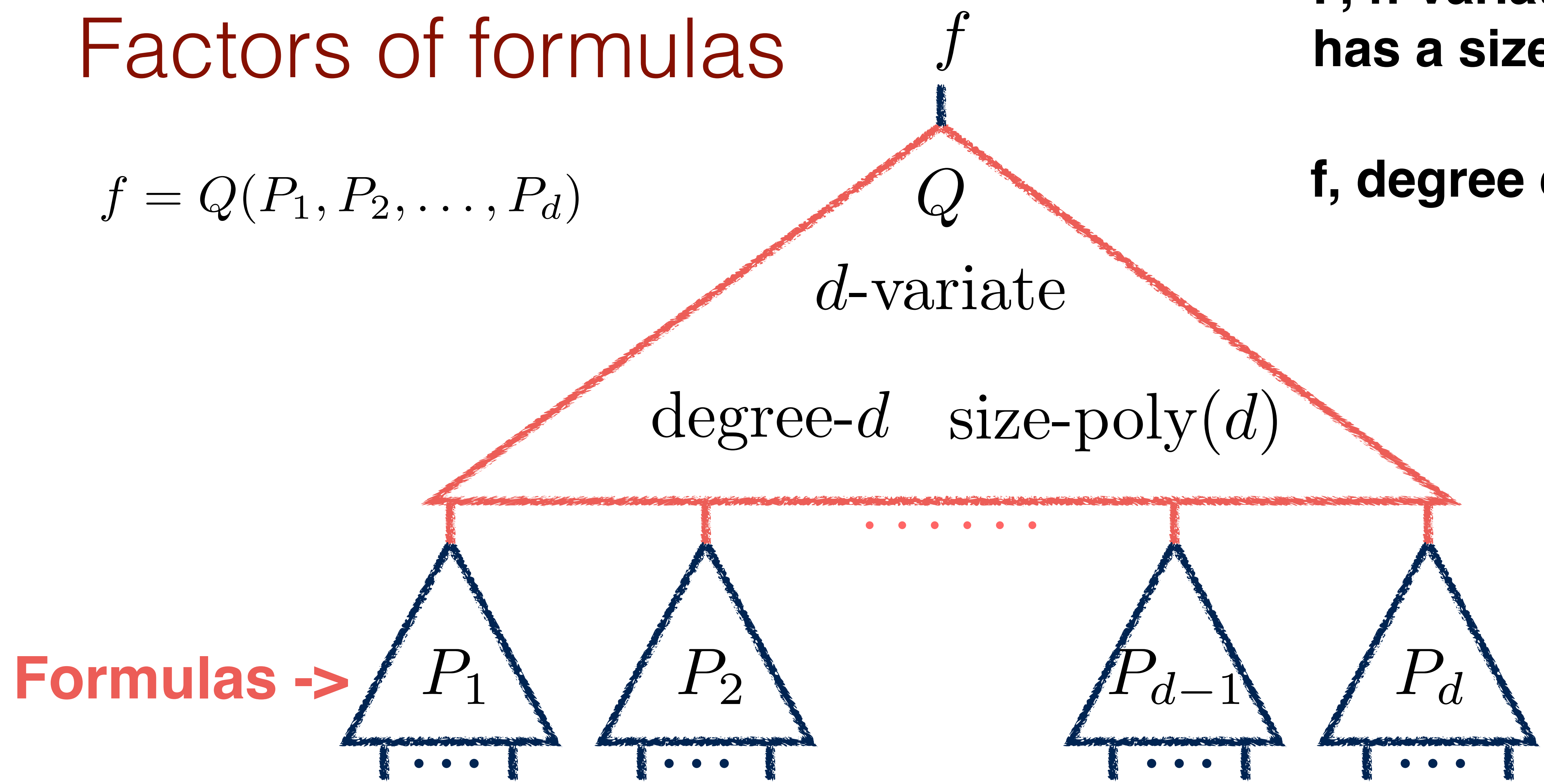


Factors of formulas

P, n-variate, degree D has a size s formula.

f, degree d factor of P.

$f = Q(P_1, P_2, \ldots, P_d)$

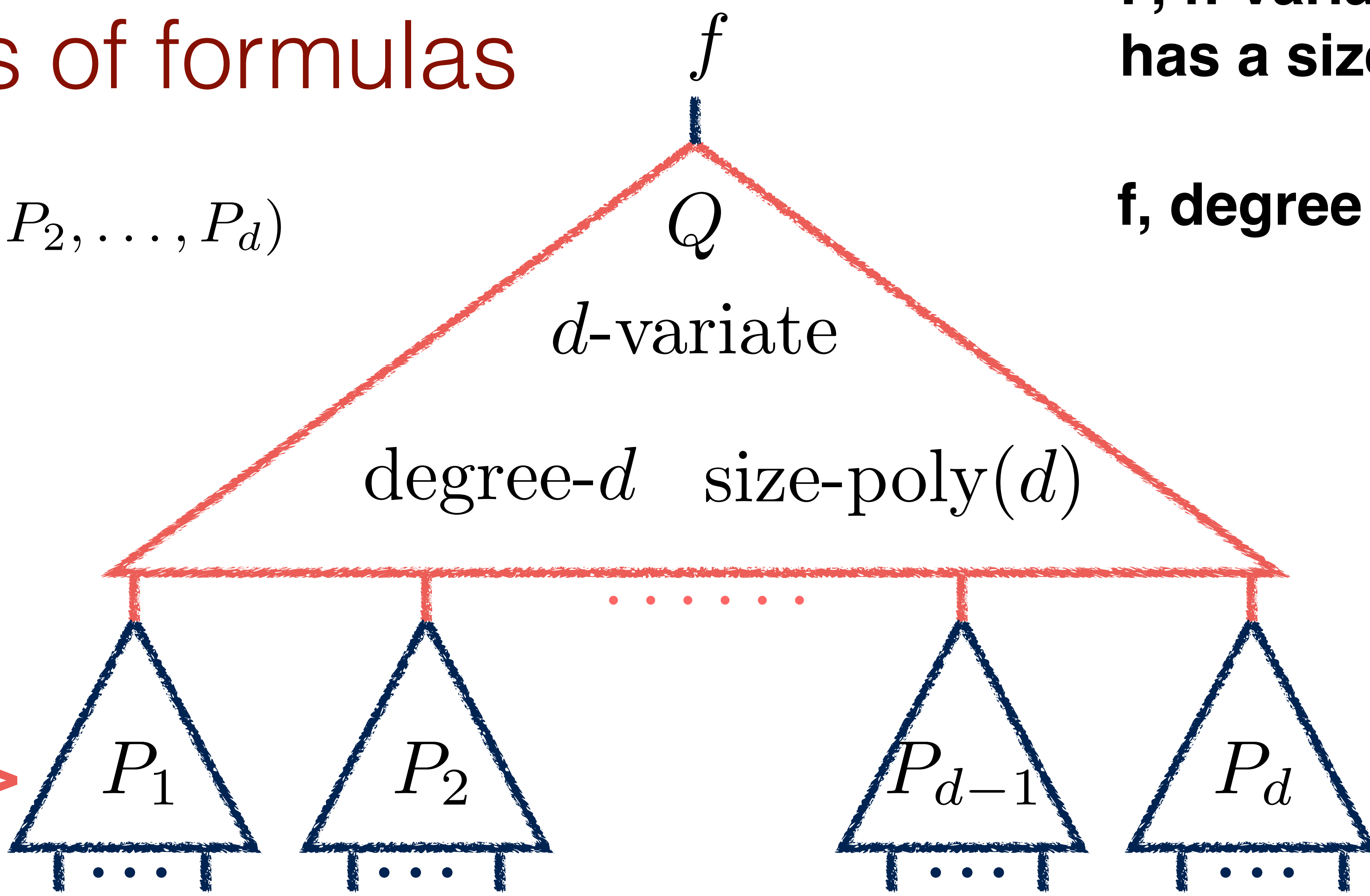


Factors of formulas

P, n-variate, degree D has a size s formula.

f, degree d factor of P.

$f = Q(P_1, P_2, \ldots, P_d)$



Formulas ->

Theorem [Valiant-Skyum-Berkowitz-Rackoff]

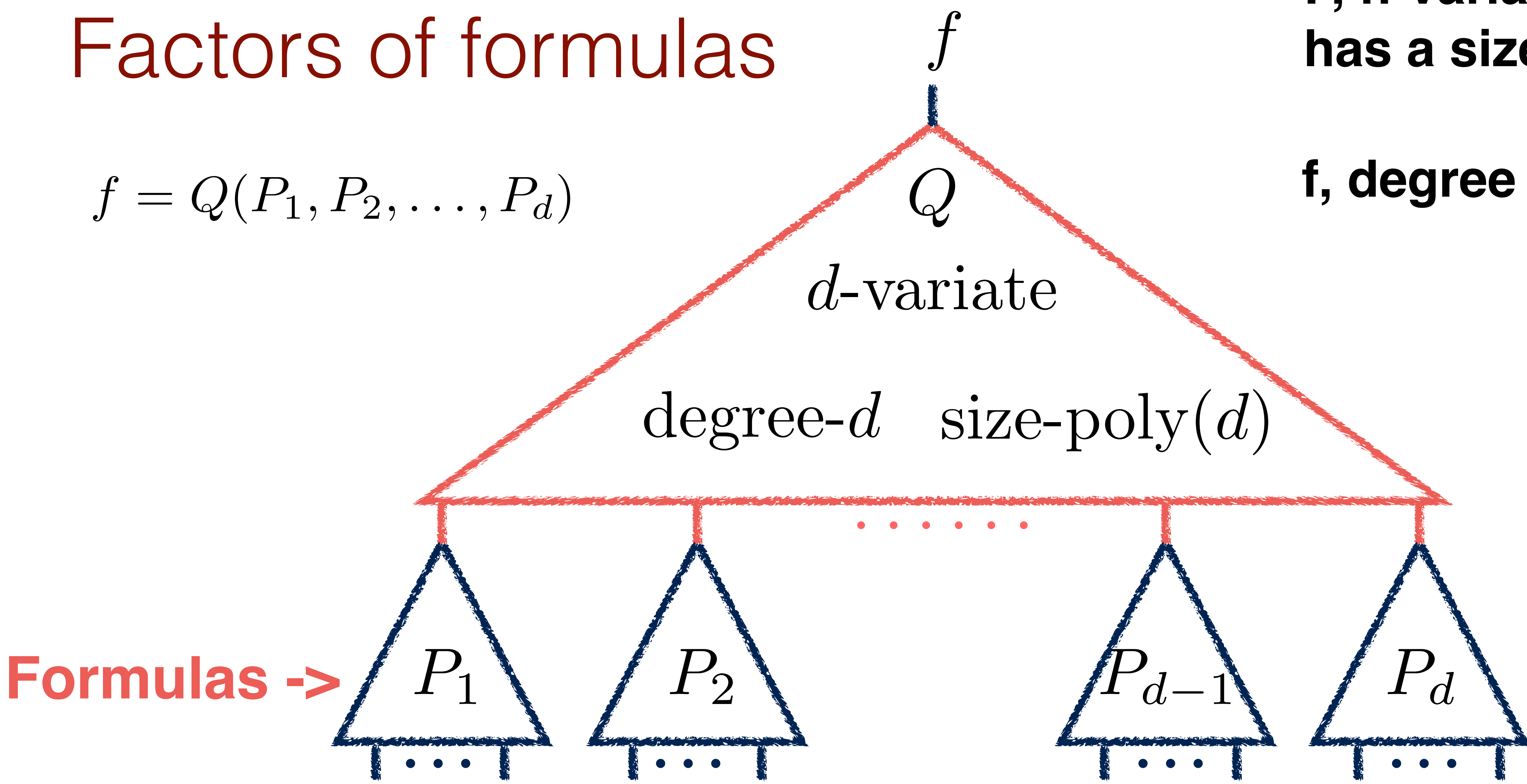
Q has a formula of size $\mathbf{d^{O(\log d)}}$.

Factors of formulas

P, n-variate, degree D has a size s formula.

f, degree d factor of P.

$$f = Q(P_1, P_2, \ldots, P_d)$$



Theorem [Valiant-Skyum-Berkowitz-Rackoff]

Q has a formula of size $\mathbf{d^{O(\log d)}}$.

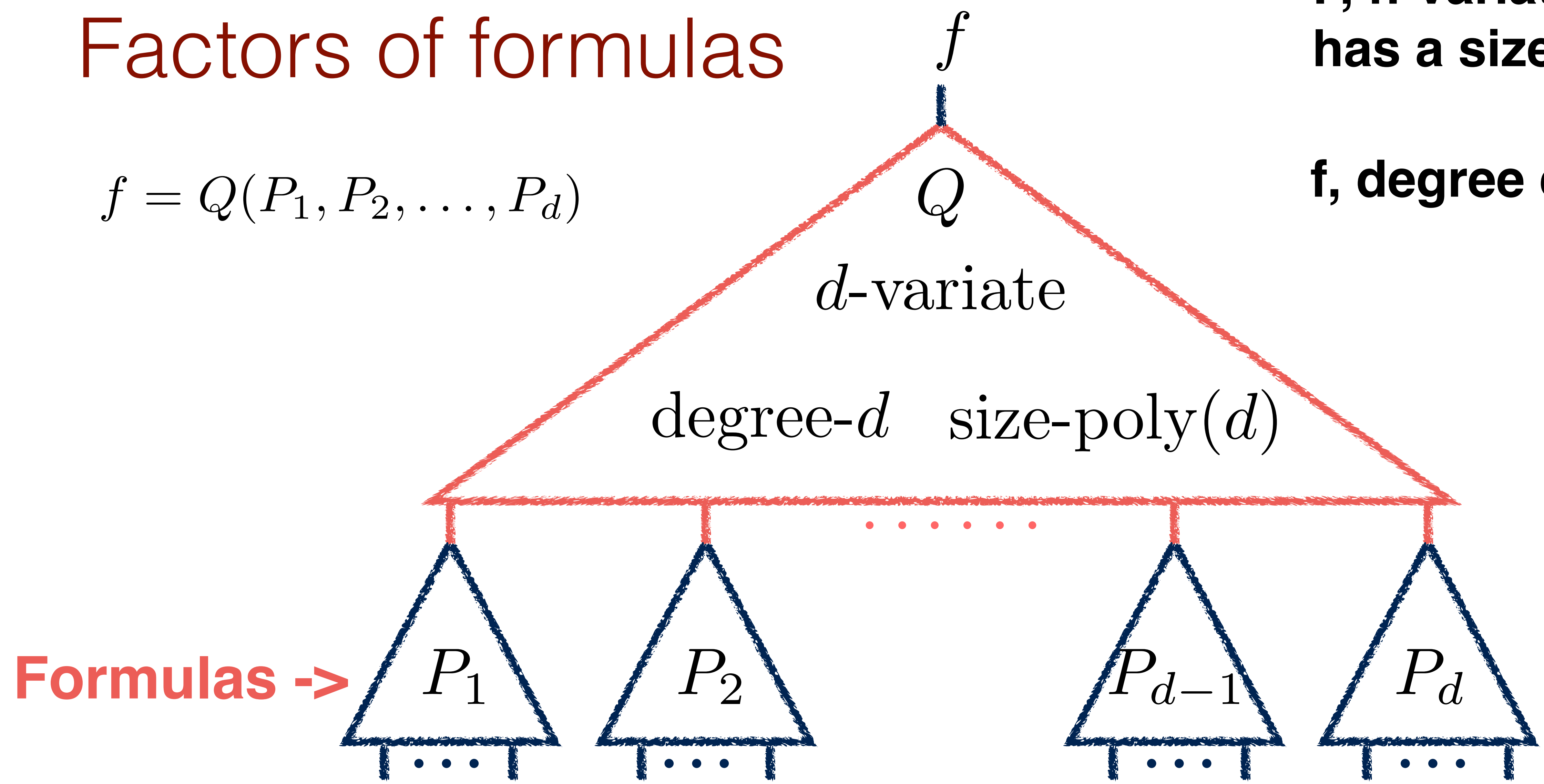
Take the formula for $\mathbf{Q}$, and paste a formula for each $\mathbf{P_i}$ at the leaves.

Factors of formulas

P, n-variate, degree D has a size s formula.

f, degree d factor of P.

$f = Q(P_1, P_2, \ldots, P_d)$



Theorem [Valiant-Skyum-Berkowitz-Rackoff]

Q has a formula of size $\mathbf{d^{O(\log d)}}$.

Take the formula for $\mathbf{Q}$, and paste a formula for each $\mathbf{P_i}$ at the leaves.

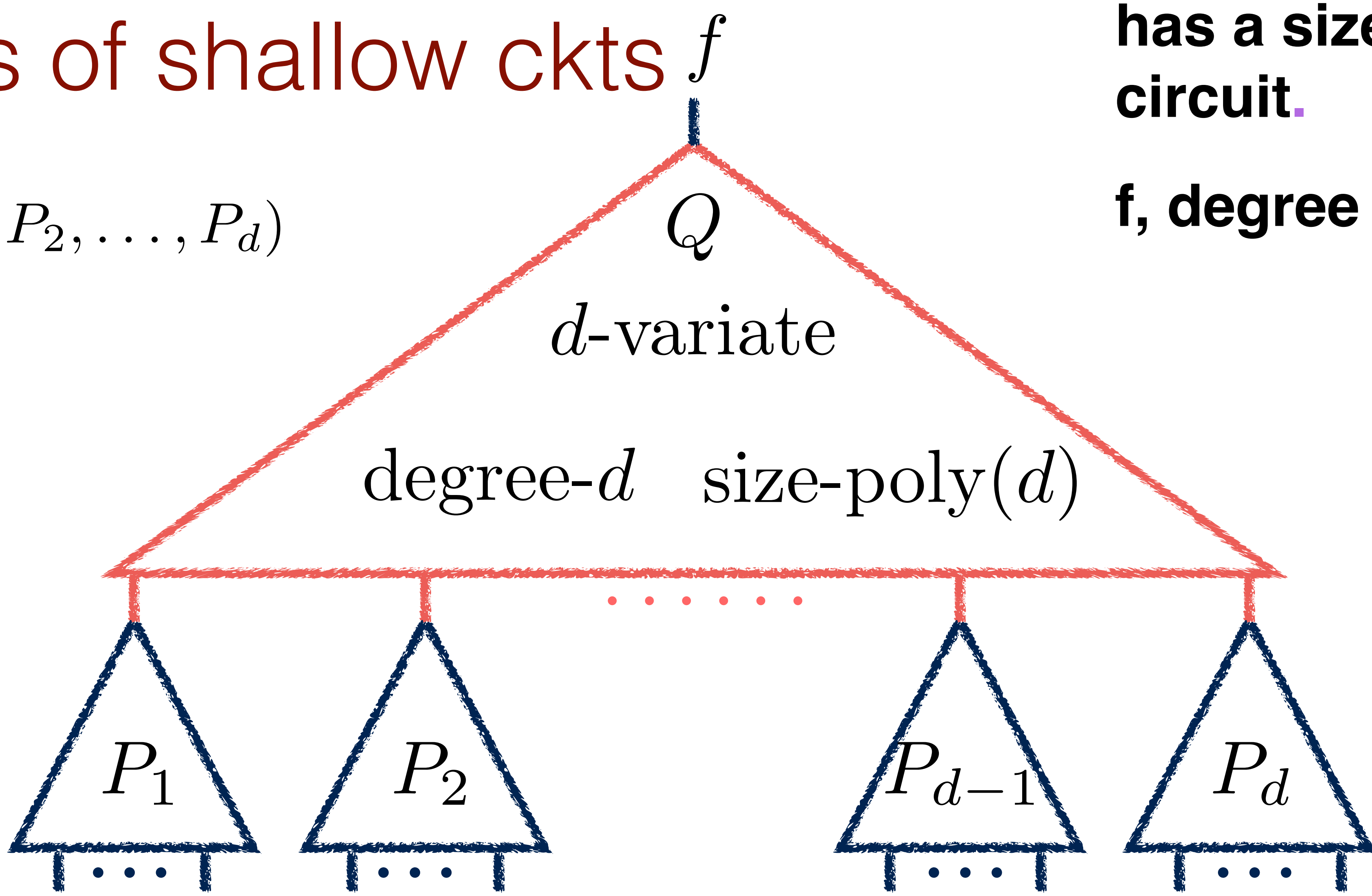
We get a formula for f of size $\mathbf{d^{O(\log d)}} \cdot \mathrm{poly}(\mathbf{n}, \mathbf{s}, \mathbf{D})$.

Factors of shallow ckts

P, n-variate, degree D has a size s, depth-k circuit.

f, degree d factor of P.

$f = Q(P_1, P_2, \ldots, P_d)$

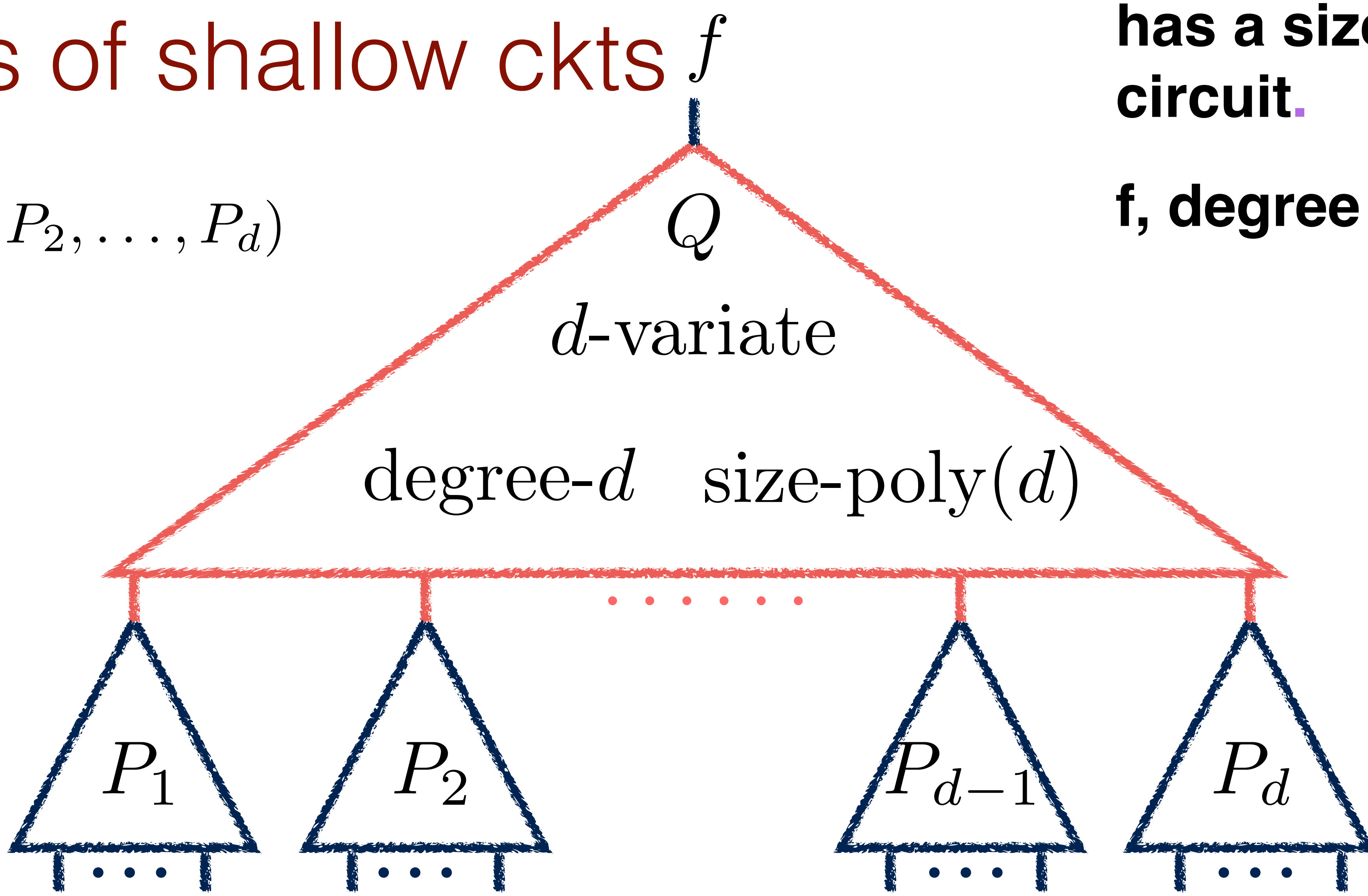


Factors of shallow ckts

P, n-variate, degree D has a size s, depth-k circuit.

f, degree d factor of P.

$f = Q(P_1, P_2, \ldots, P_d)$



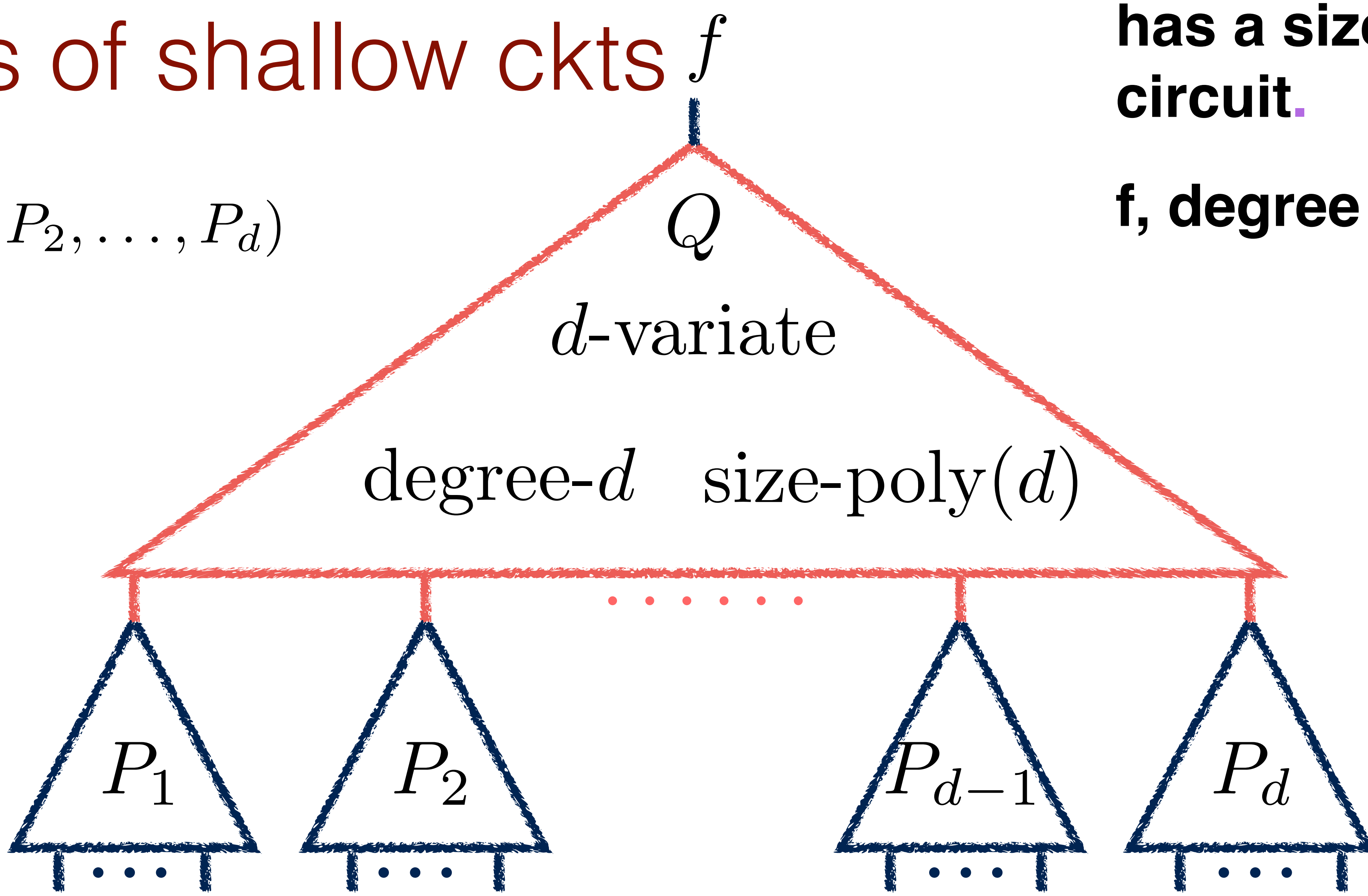
Depth k + 1 ckt ->

Factors of shallow ckts

P, n-variate, degree D has a size s, depth-k circuit.

f, degree d factor of P.

$f = Q(P_1, P_2, \ldots, P_d)$



Depth k + 1 ckt ->

Theorem [Agrawal-Vinay, Tavenas]

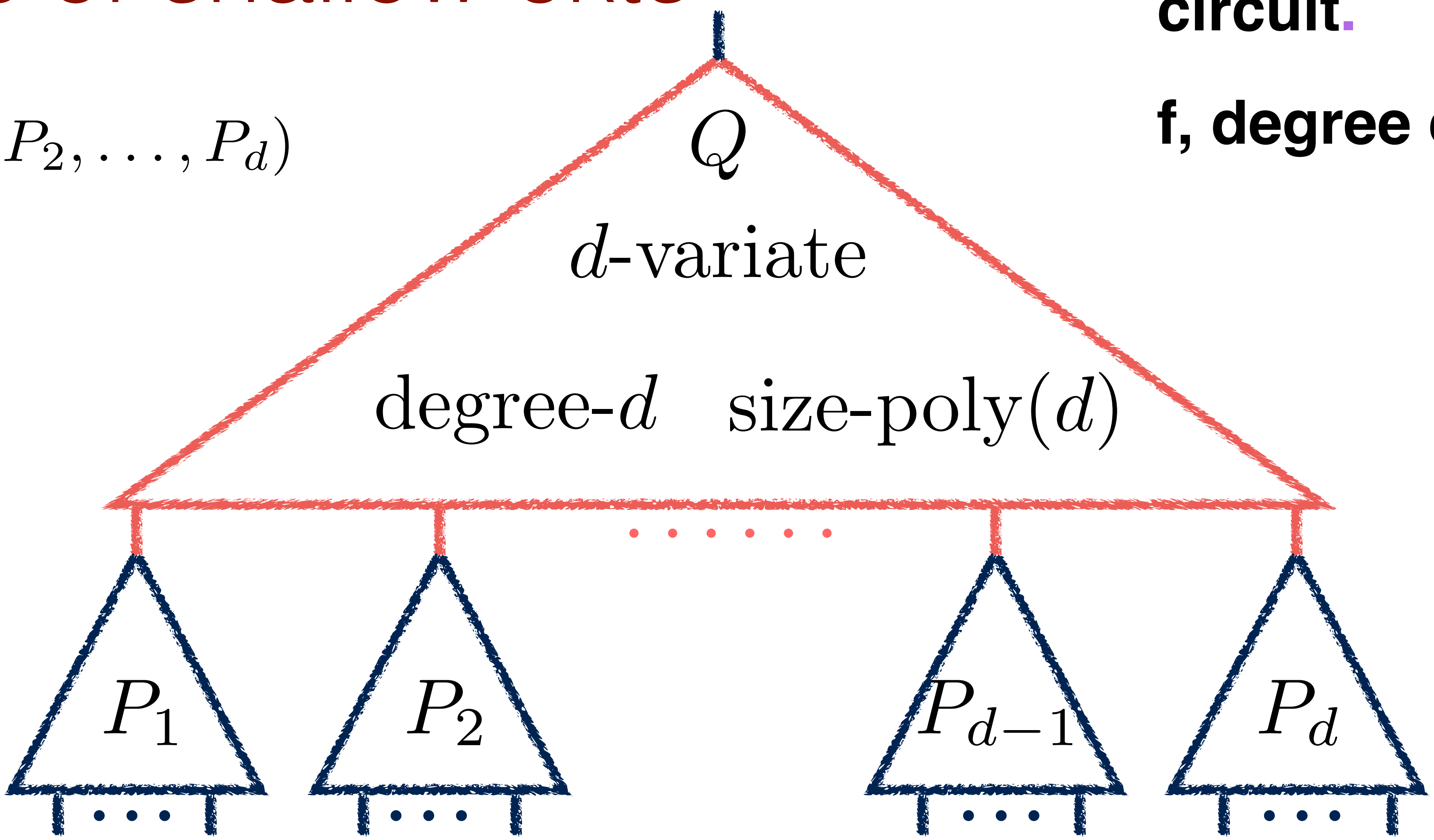
Q has a depth 2c circuit of size $\mathbf{d^{O(d^{1/c})}}$.

Factors of shallow ckts f

P, n-variate, degree D has a size s, depth-k circuit.

f, degree d factor of P.

$f = Q(P_1, P_2, \ldots, P_d)$



Depth k + 1 ckt ->

Theorem [Agrawal-Vinay, Tavenas]

Q has a depth 2c circuit of size $\mathbf{d^{O(d^{1/c})}}$.

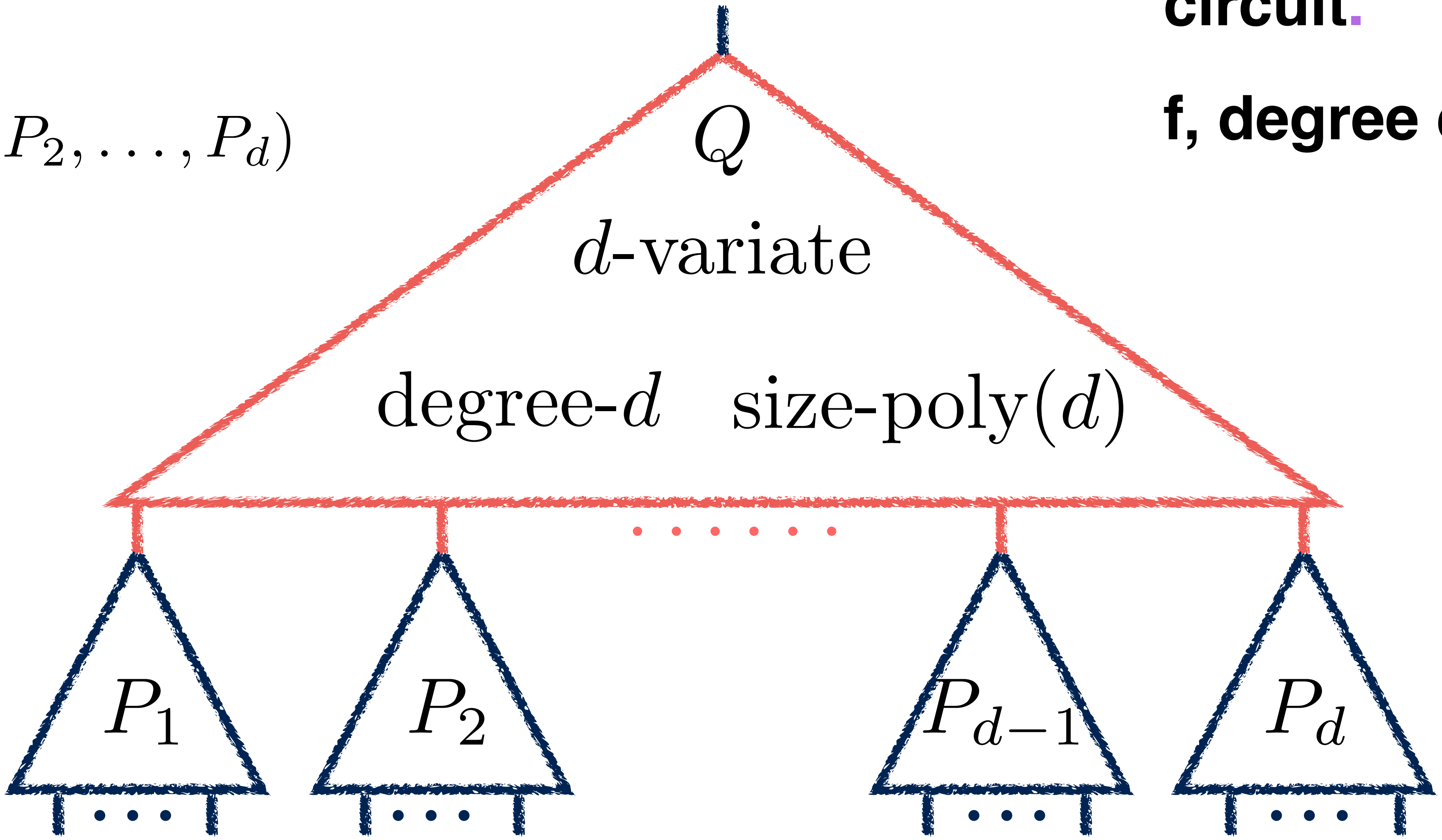
Take the shallow circuit for $\mathbf{Q}$, and paste the shallow circuit for each $\mathbf{P_i}$ at the leaves.

Factors of shallow ckts f

P, n-variate, degree r has a size s, depth-k circuit.

f, degree d factor of P.

$f = Q(P_1, P_2, \ldots, P_d)$



Depth k + 1 ckt ->

Theorem [Agrawal-Vinay, Tavenas]

Q has a depth 2c circuit of size $\mathbf{d^{O(d^{1/c})}}$.

Take the shallow circuit for $\mathbf{Q}$, and paste the shallow circuit for each $\mathbf{P_i}$ at the leaves.

We get a circuit for f of depth $k + 2c + O(1)$ and size $\mathbf{d^{O(d^{\epsilon})}} \cdot \mathrm{poly}(\mathbf{n}, \mathbf{s}, \mathbf{D})$

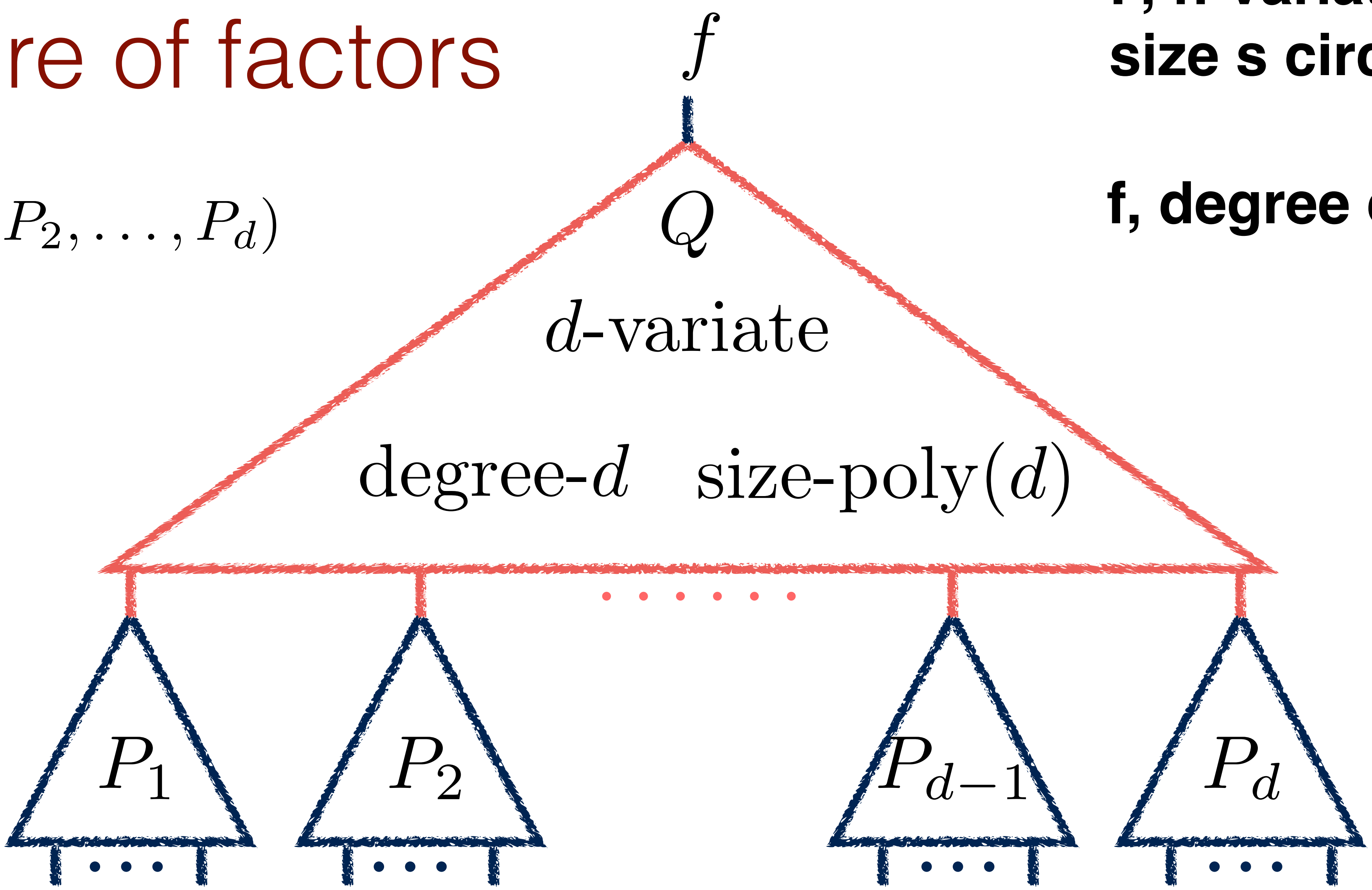
Proving the structural lemma

Structure of factors

P, n-variate, degree D size s circuit.

f, degree d factor of P.

$f = Q(P_1, P_2, \ldots, P_d)$



size(P_i) = poly(s, D)

structure preserved

low depth $\rightarrow$ low depth

formula $\rightarrow$ formula

VNP $\rightarrow$ VNP

Structure of roots

Structure of roots

The lemma for ‘roots’ of P.

Structure of roots

The lemma for 'roots' of P.

$$P(X_1, X_1, \ldots, X_{n-1}, Y) \qquad \text{degree}(P) = r \qquad \text{size}(P) = s$$

Structure of roots

The lemma for 'roots' of P.

$P(X_1, X_1, \ldots, X_{n-1}, Y)$ $\quad \text{degree}(P) = r \quad \text{size}(P) = s$

$f(X_1, X_1, \ldots, X_{n-1})$ $\quad \text{degree}(f) = d \quad P(\mathbf{X}, f) = 0$

Structure of roots

The lemma for ‘roots’ of P.

$P(X_1, X_1, \ldots, X_{n-1}, Y)$ $\quad$ $\text{degree}(P) = r$ $\quad$ $\text{size}(P) = s$

$f(X_1, X_1, \ldots, X_{n-1})$ $\quad$ $\text{degree}(f) = d$ $\quad$ $P(\mathbf{X}, f) = 0$

$Q(P_1, P_2, \ldots, P_d) = f$ $\quad$ $\text{degree}(Q) = d$ $\quad$ $\text{size}(Q) = \text{poly}(d)$

Structure of roots

The lemma for 'roots' of P.

$P(X_1, X_1, \ldots, X_{n-1}, Y)$ $\quad \text{degree}(P) = r$ $\quad \text{size}(P) = s$

$f(X_1, X_1, \ldots, X_{n-1})$ $\quad \text{degree}(f) = d$ $\quad P(\mathbf{X}, f) = 0$

$Q(P_1, P_2, \ldots, P_d) = f$ $\quad \text{degree}(Q) = d$ $\quad \text{size}(Q) = \text{poly}(d)$

Preprocessing :

Structure of roots

The lemma for ‘roots’ of P.

$P(X_1, X_1, \ldots, X_{n-1}, Y)$ $\quad$ $\text{degree}(P) = r$ $\quad$ $\text{size}(P) = s$

$f(X_1, X_1, \ldots, X_{n-1})$ $\quad$ $\text{degree}(f) = d$ $\quad$ $P(\mathbf{X}, f) = 0$

$Q(P_1, P_2, \ldots, P_d) = f$ $\quad$ $\text{degree}(Q) = d$ $\quad$ $\text{size}(Q) = \text{poly}(d)$

Preprocessing :

$$\frac{\partial P}{\partial y}(\mathbf{X}, f) \neq 0$$

Structure of roots

The lemma for ‘roots’ of P.

$P(X_1, X_1, \ldots, X_{n-1}, Y)$ $\quad \text{degree}(P) = r$ $\quad \text{size}(P) = s$

$f(X_1, X_1, \ldots, X_{n-1})$ $\quad \text{degree}(f) = d$ $\quad P(\mathbf{X}, f) = 0$

$Q(P_1, P_2, \ldots, P_d) = f$ $\quad \text{degree}(Q) = d$ $\quad \text{size}(Q) = \text{poly}(d)$

Preprocessing :

$$\frac{\partial P}{\partial y}(\mathbf{X}, f) \neq 0$$

Else, we work with a derivative of P. They also have ‘small’ circuits.

Structure of roots

The lemma for ‘roots’ of P.

$P(X_1, X_1, \ldots, X_{n-1}, Y)$ $\quad$ $\text{degree}(P) = r$ $\quad$ $\text{size}(P) = s$

$f(X_1, X_1, \ldots, X_{n-1})$ $\quad$ $\text{degree}(f) = d$ $\quad$ $P(\mathbf{X}, f) = 0$

$Q(P_1, P_2, \ldots, P_d) = f$ $\quad$ $\text{degree}(Q) = d$ $\quad$ $\text{size}(Q) = \text{poly}(d)$

Preprocessing :

$$\frac{\partial P}{\partial y}(\mathbf{X}, f) \neq 0$$

Else, we work with a derivative of P. They also have ‘small’ circuits.

$$\frac{\partial P}{\partial y}(\mathbf{0}, f(0)) = \delta \neq 0$$

Structure of roots

The lemma for 'roots' of P.

$P(X_1, X_1, \ldots, X_{n-1}, Y)$ $\quad \text{degree}(P) = r \quad \text{size}(P) = s$

$f(X_1, X_1, \ldots, X_{n-1})$ $\quad \text{degree}(f) = d \quad P(\mathbf{X}, f) = 0$

$Q(P_1, P_2, \ldots, P_d) = f$ $\quad \text{degree}(Q) = d \quad \text{size}(Q) = \text{poly}(d)$

Preprocessing :

$$\frac{\partial P}{\partial y}(\mathbf{X}, f) \neq 0$$

Else, we work with a derivative of P. They also have 'small' circuits.

$$\frac{\partial P}{\partial y}(\mathbf{0}, f(0)) = \delta \neq 0$$

Else, we translate the origin to ensure this.

Defining the generators

Defining the generators

$$P(\mathbf{X}, f(\mathbf{0}) + Y) = P(\mathbf{X}, f(\mathbf{0})) + Y\frac{\partial P}{\partial Y}(\mathbf{X}, f(\mathbf{0})) + \cdots + Y^r \cdot \frac{1}{r!} \cdot \frac{\partial^r P}{\partial Y^r}(\mathbf{X}, f(\mathbf{0})$$

Defining the generators

$$P(\mathbf{X}, f(\mathbf{0}) + Y) = P(\mathbf{X}, f(\mathbf{0})) + Y\frac{\partial P}{\partial Y}(\mathbf{X}, f(\mathbf{0})) + \cdots + Y^r \cdot \frac{1}{r!} \cdot \frac{\partial^r P}{\partial Y^r}(\mathbf{X}, f(\mathbf{0})$$

$$P_i(\mathbf{X}) = \frac{1}{i!} \cdot \frac{\partial^i P}{\partial Y^i}(\mathbf{X}, f(\mathbf{0})) - \frac{1}{i!} \cdot \frac{\partial^i P}{\partial Y^i}(\mathbf{0}, f(\mathbf{0}))$$

Defining the generators

$$P(\mathbf{X}, f(\mathbf{0}) + Y) = P(\mathbf{X}, f(\mathbf{0})) + Y \frac{\partial P}{\partial Y}(\mathbf{X}, f(\mathbf{0})) + \cdots + Y^r \cdot \frac{1}{r!} \cdot \frac{\partial^r P}{\partial Y^r}(\mathbf{X}, f(\mathbf{0})$$

$$P_i(\mathbf{X}) = \frac{1}{i!} \cdot \frac{\partial^i P}{\partial Y^i}(\mathbf{X}, f(\mathbf{0})) - \frac{1}{i!} \cdot \frac{\partial^i P}{\partial Y^i}(\mathbf{0}, f(\mathbf{0}))$$

$$P_i(\mathbf{0}) = 0$$

Defining the generators

$$P(\mathbf{X}, f(\mathbf{0}) + Y) = P(\mathbf{X}, f(\mathbf{0})) + Y \frac{\partial P}{\partial Y}(\mathbf{X}, f(\mathbf{0})) + \cdots + Y^r \cdot \frac{1}{r!} \cdot \frac{\partial^r P}{\partial Y^r}(\mathbf{X}, f(\mathbf{0})$$

$$P_i(\mathbf{X}) = \frac{1}{i!} \cdot \frac{\partial^i P}{\partial Y^i}(\mathbf{X}, f(\mathbf{0})) - \frac{1}{i!} \cdot \frac{\partial^i P}{\partial Y^i}(\mathbf{0}, f(\mathbf{0}))$$

$$P_i(\mathbf{0}) = 0$$

$$P(\mathbf{X}, f(\mathbf{0}) + Y) = P(\mathbf{X}, f(\mathbf{0})) + Y \cdot (P_1 + \alpha_1) + Y^2 \cdot (P_2 + \alpha_2) + \cdots + Y^r \cdot (P_r + \alpha_r)$$

Defining the generators

$$P(\mathbf{X}, f(\mathbf{0}) + Y) = P(\mathbf{X}, f(\mathbf{0})) + Y \frac{\partial P}{\partial Y}(\mathbf{X}, f(\mathbf{0})) + \cdots + Y^r \cdot \frac{1}{r!} \cdot \frac{\partial^r P}{\partial Y^r}(\mathbf{X}, f(\mathbf{0})$$

$$P_i(\mathbf{X}) = \frac{1}{i!} \cdot \frac{\partial^i P}{\partial Y^i}(\mathbf{X}, f(\mathbf{0})) - \frac{1}{i!} \cdot \frac{\partial^i P}{\partial Y^i}(\mathbf{0}, f(\mathbf{0})) \qquad P_i(\mathbf{0}) = 0$$

$$P(\mathbf{X}, f(\mathbf{0}) + Y) = P(\mathbf{X}, f(\mathbf{0})) + Y \cdot (P_1 + \alpha_1) + Y^2 \cdot (P_2 + \alpha_2) + \cdots + Y^r \cdot (P_r + \alpha_r)$$

$$P(\mathbf{X}, f(\mathbf{0}) + Y) = P(\mathbf{X}, f(\mathbf{0})) + Y \cdot (P_1 + \delta) + Y^2 \cdot (P_2 + \alpha_2) + \cdots + Y^r \cdot (P_r + \alpha_r)$$

Constructing the root : Newton iteration

Constructing the root : Newton iteration

Constructing the root : Newton iteration

We construct the root iteratively.

Constructing the root : Newton iteration

We construct the root iteratively.

We start with the degree 0 term of the root.

Constructing the root : Newton iteration

We construct the root iteratively.

We start with the degree 0 term of the root.

At the end of iteration i, we will be able to recover the homogeneous components of f of degree up to i.

Newton iteration : base case

Newton iteration : base case

$$P(\mathbf{X}, f) = 0$$

Newton iteration : base case

$$P(\mathbf{X}, f) = 0$$

$$\mathrm{Linear}(P(\mathbf{X}, f)) = 0$$

Newton iteration : base case

$$P(\mathbf{X}, f) = 0$$

$$\text{Linear}(P(\mathbf{X}, f)) = 0$$

$$\text{Linear}(P(\mathbf{X}, f(0) + L)) = 0$$

$$L := \text{homogeneous component of } f \text{ of degree equal to } 1$$

Newton iteration : base case

Base case : Getting a circuit for the linear term

$$P(\mathbf{X}, f(\mathbf{0}) + L) = P(\mathbf{X}, f(\mathbf{0})) + L \cdot (P_1 + \delta) + L^2 \cdot (P_2 + \alpha_2) + \cdots + L^r \cdot (P_r + \alpha_r)$$

Newton iteration : base case

Base case : Getting a circuit for the linear term

$$P(\mathbf{X}, f(\mathbf{0}) + L) = P(\mathbf{X}, f(\mathbf{0})) + L \cdot (P_1 + \delta) + L^2 \cdot (P_2 + \alpha_2) + \cdots + L^r \cdot (P_r + \alpha_r)$$

$$\text{Linear}\,(P(\mathbf{X}, f(\mathbf{0}) + L)) = 0$$

Newton iteration : base case

Base case : Getting a circuit for the linear term

$$P(\mathbf{X}, f(\mathbf{0}) + L) = P(\mathbf{X}, f(\mathbf{0})) + L \cdot (P_1 + \delta) + L^2 \cdot (P_2 + \alpha_2) + \cdots + L^r \cdot (P_r + \alpha_r)$$

$$\text{Linear}\,(P(\mathbf{X}, f(\mathbf{0}) + L)) = 0$$

$$\text{Linear}\,\left(P(\mathbf{X}, f(\mathbf{0})) + L \cdot (P_1 + \delta) + L^2 \cdot (P_2 + \alpha_2) + \cdots + L^r \cdot (P_r + \alpha_r)\right) = 0$$

Newton iteration : base case

Base case : Getting a circuit for the linear term

$$P(\mathbf{X}, f(\mathbf{0}) + L) = P(\mathbf{X}, f(\mathbf{0})) + L \cdot (P_1 + \delta) + L^2 \cdot (P_2 + \alpha_2) + \cdots + L^r \cdot (P_r + \alpha_r)$$

$$\text{Linear}\,(P(\mathbf{X}, f(\mathbf{0}) + L)) = 0$$

$$\text{Linear}\left(P(\mathbf{X}, f(\mathbf{0})) + L \cdot (P_1 + \delta) + L^2 \cdot (P_2 + \alpha_2) + \cdots + L^r \cdot (P_r + \alpha_r))\right) = 0$$

L : homogeneous degree 1

Newton iteration : base case

Base case : Getting a circuit for the linear term

$$P(\mathbf{X}, f(\mathbf{0}) + L) = P(\mathbf{X}, f(\mathbf{0})) + L \cdot (P_1 + \delta) + L^2 \cdot (P_2 + \alpha_2) + \cdots + L^r \cdot (P_r + \alpha_r)$$

$$\text{Linear}\,(P(\mathbf{X}, f(\mathbf{0}) + L)) = 0$$

$$\text{Linear}\,\big(P(\mathbf{X}, f(\mathbf{0})) + L \cdot (P_1 + \delta) + L^2 \cdot (P_2 + \alpha_2) + \cdots + L^r \cdot (P_r + \alpha_r))\big) = 0$$

L : homogeneous degree 1

$$\forall j > 1, \text{Linear}(L^j \cdot A) = 0$$

Newton iteration : base case

Base case : Getting a circuit for the linear term

$$P(\mathbf{X}, f(\mathbf{0}) + L) = P(\mathbf{X}, f(\mathbf{0})) + L \cdot (P_1 + \delta) + L^2 \cdot (P_2 + \alpha_2) + \cdots + L^r \cdot (P_r + \alpha_r)$$

$$\text{Linear}\,(P(\mathbf{X}, f(\mathbf{0}) + L)) = 0$$

$$\text{Linear}\,\big(P(\mathbf{X}, f(\mathbf{0})) + L \cdot (P_1 + \delta) + L^2 \cdot (P_2 + \alpha_2) + \cdots + L^r \cdot (P_r + \alpha_r)\big) = 0$$

L : homogeneous degree 1 $\forall j > 1, \text{Linear}(L^j \cdot A) = 0$

$$0 = \text{Linear}(P(\mathbf{X}, f(\mathbf{0}))) + L \cdot \text{Constant Term}(P_1 + \delta)$$

Newton iteration : base case

Base case : Getting a circuit for the linear term

$$P(\mathbf{X}, f(\mathbf{0}) + L) = P(\mathbf{X}, f(\mathbf{0})) + L \cdot (P_1 + \delta) + L^2 \cdot (P_2 + \alpha_2) + \cdots + L^r \cdot (P_r + \alpha_r)$$

$$\text{Linear}\,(P(\mathbf{X}, f(\mathbf{0}) + L)) = 0$$

$$\text{Linear}\,\big(P(\mathbf{X}, f(\mathbf{0})) + L \cdot (P_1 + \delta) + L^2 \cdot (P_2 + \alpha_2) + \cdots + L^r \cdot (P_r + \alpha_r))\big) = 0$$

L : homogeneous degree 1 $\forall j > 1, \text{Linear}(L^j \cdot A) = 0$

$$0 = \text{Linear}(P(\mathbf{X}, f(\mathbf{0}))) + L \cdot \text{Constant Term}(P_1 + \delta)$$

$$= \text{Linear}(P(\mathbf{X}, f(\mathbf{0}))) + L \cdot \delta$$

Newton iteration : base case

Base case : Getting a circuit for the linear term

$$P(\mathbf{X}, f(\mathbf{0}) + L) = P(\mathbf{X}, f(\mathbf{0})) + L \cdot (P_1 + \delta) + L^2 \cdot (P_2 + \alpha_2) + \cdots + L^r \cdot (P_r + \alpha_r)$$

$$\text{Linear}\left(P(\mathbf{X}, f(\mathbf{0}) + L)\right) = 0$$

$$\text{Linear}\left(P(\mathbf{X}, f(\mathbf{0})) + L \cdot (P_1 + \delta) + L^2 \cdot (P_2 + \alpha_2) + \cdots + L^r \cdot (P_r + \alpha_r)\right) = 0$$

L : homogeneous degree 1

$$\forall j > 1, \text{Linear}(L^j \cdot A) = 0$$

$$0 = \text{Linear}(P(\mathbf{X}, f(\mathbf{0}))) + L \cdot \text{Constant Term}(P_1 + \delta)$$

$$= \text{Linear}(P(\mathbf{X}, f(\mathbf{0}))) + L \cdot \delta$$

$$\delta \neq 0$$

Newton iteration : base case

Base case : Getting a circuit for the linear term

$$P(\mathbf{X}, f(\mathbf{0}) + L) = P(\mathbf{X}, f(\mathbf{0})) + L \cdot (P_1 + \delta) + L^2 \cdot (P_2 + \alpha_2) + \cdots + L^r \cdot (P_r + \alpha_r)$$

$$\mathrm{Linear}\,(P(\mathbf{X}, f(\mathbf{0}) + L)) = 0$$

$$\mathrm{Linear}\left(P(\mathbf{X}, f(\mathbf{0})) + L \cdot (P_1 + \delta) + L^2 \cdot (P_2 + \alpha_2) + \cdots + L^r \cdot (P_r + \alpha_r)\right) = 0$$

L : homogeneous degree 1

$$\forall j > 1, \mathrm{Linear}(L^j \cdot A) = 0$$

$$0 = \mathrm{Linear}(P(\mathbf{X}, f(\mathbf{0}))) + L \cdot \mathrm{Constant\ Term}(P_1 + \delta)$$

$$= \mathrm{Linear}(P(\mathbf{X}, f(\mathbf{0}))) + L \cdot \delta$$

$$\delta \neq 0$$

$$\frac{-1}{\delta} \cdot \mathrm{Linear}(P(\mathbf{X}, f(\mathbf{0}))) = L$$

Newton iteration : base case

Base case : Getting a circuit for the linear term

$$P(\mathbf{X}, f(\mathbf{0}) + L) = P(\mathbf{X}, f(\mathbf{0})) + L \cdot (P_1 + \delta) + L^2 \cdot (P_2 + \alpha_2) + \cdots + L^r \cdot (P_r + \alpha_r)$$

$$\text{Linear}\,(P(\mathbf{X}, f(\mathbf{0}) + L)) = 0$$

$$\text{Linear}\left(P(\mathbf{X}, f(\mathbf{0})) + L \cdot (P_1 + \delta) + L^2 \cdot (P_2 + \alpha_2) + \cdots + L^r \cdot (P_r + \alpha_r)\right) = 0$$

L : homogeneous degree 1

$$\forall j > 1, \text{Linear}(L^j \cdot A) = 0$$

$$0 = \text{Linear}(P(\mathbf{X}, f(\mathbf{0}))) + L \cdot \text{Constant Term}(P_1 + \delta)$$

$$= \text{Linear}(P(\mathbf{X}, f(\mathbf{0}))) + L \cdot \delta$$

$$\delta \neq 0$$

$$\frac{-1}{\delta} \cdot \text{Linear}(P(\mathbf{X}, f(\mathbf{0}))) = L$$

$$f(0) + L = A_1(P_0)$$

Newton iteration : base case

Base case : Getting a circuit for the linear term

$$P(\mathbf{X}, f(\mathbf{0}) + L) = P(\mathbf{X}, f(\mathbf{0})) + L \cdot (P_1 + \delta) + L^2 \cdot (P_2 + \alpha_2) + \cdots + L^r \cdot (P_r + \alpha_r)$$

$$\text{Linear}\left(P(\mathbf{X}, f(\mathbf{0}) + L)\right) = 0$$

$$\text{Linear}\left(P(\mathbf{X}, f(\mathbf{0})) + L \cdot (P_1 + \delta) + L^2 \cdot (P_2 + \alpha_2) + \cdots + L^r \cdot (P_r + \alpha_r)\right) = 0$$

L : homogeneous degree 1

$$\forall j > 1, \text{Linear}(L^j \cdot A) = 0$$

$$0 = \text{Linear}(P(\mathbf{X}, f(\mathbf{0}))) + L \cdot \text{Constant Term}(P_1 + \delta)$$

$$= \text{Linear}(P(\mathbf{X}, f(\mathbf{0}))) + L \cdot \delta$$

$$\delta \neq 0$$

$$\frac{-1}{\delta} \cdot \text{Linear}(P(\mathbf{X}, f(\mathbf{0}))) = L$$

$$f(0) + L = A_1(P_0)$$

Newton iteration : general case

Newton iteration : general case

At the end of step k-1 :

Newton iteration : general case

At the end of step k-1 :

$\forall j < k, \mathrm{Deg}_j(f(\mathbf{0}) + h) = \mathrm{Deg}_j(f)$

$h = A_{k-1}(P_0, P_1, P_2, \ldots, P_d)$

Newton iteration : general case

At the end of step k-1 :

$\forall j < k, \mathrm{Deg}_j(f(\mathbf{0}) + h) = \mathrm{Deg}_j(f)$ $\quad$ $h = A_{k-1}(P_0, P_1, P_2, \ldots, P_d)$

$\forall j < k + 1, \mathrm{Deg}_j(f(\mathbf{0}) + h') = \mathrm{Deg}_j(f)$ $\quad$ $h' = A_k(P_0, P_1, P_2, \ldots, P_d)$

Newton iteration : general case

At the end of step k-1 :

$$\forall j < k, \mathrm{Deg}_j(f(\mathbf{0}) + h) = \mathrm{Deg}_j(f) \qquad h = A_{k-1}(P_0, P_1, P_2, \ldots, P_d)$$

$$\forall j < k+1, \mathrm{Deg}_j(f(\mathbf{0}) + h') = \mathrm{Deg}_j(f) \qquad h' = A_k(P_0, P_1, P_2, \ldots, P_d)$$

$$h' = h + \text{ monomials of degree } k$$

$$\mathrm{Deg}_k(P(\mathbf{x}), f(\mathbf{0}) + h + (h' - h)) = 0 \qquad h' = h - \mathrm{Deg}_k\left(\frac{1}{\delta}\left(P(\mathbf{X}, f(\mathbf{0}) + h)\right)\right)$$

Newton iteration : general case

At the end of step k-1 :

$\forall j < k, \mathrm{Deg}_j(f(\mathbf{0}) + h) = \mathrm{Deg}_j(f)$ $\quad h = A_{k-1}(P_0, P_1, P_2, \ldots, P_d)$

$\forall j < k+1, \mathrm{Deg}_j(f(\mathbf{0}) + h') = \mathrm{Deg}_j(f)$ $\quad h' = A_k(P_0, P_1, P_2, \ldots, P_d)$

$h' = h +$ monomials of degree k

$\mathrm{Deg}_k(P(\mathbf{x}), f(\mathbf{0}) + h + (h' - h)) = 0$ $\quad h' = h - \mathrm{Deg}_k\left(\frac{1}{\delta}\left(P(\mathbf{X}, f(\mathbf{0}) + h)\right)\right)$

$P(\mathbf{X}, f(\mathbf{0}) + h) = P_0 + P_1 \cdot h + P_2 \cdot h^2 + \ldots + P_r \cdot h^r$

Newton iteration : general case

At the end of step k-1 :

$$\forall j < k, \mathrm{Deg}_j(f(\mathbf{0}) + h) = \mathrm{Deg}_j(f) \qquad h = A_{k-1}(P_0, P_1, P_2, \ldots, P_d)$$

$$\forall j < k+1, \mathrm{Deg}_j(f(\mathbf{0}) + h') = \mathrm{Deg}_j(f) \qquad h' = A_k(P_0, P_1, P_2, \ldots, P_d)$$

$$h' = h + \text{ monomials of degree } k$$

$$\mathrm{Deg}_k(P(\mathbf{x}), f(\mathbf{0}) + h + (h' - h)) = 0 \qquad h' = h - \mathrm{Deg}_k\left(\frac{1}{\delta}\left(P(\mathbf{X}, f(\mathbf{0}) + h)\right)\right)$$

$$P(\mathbf{X}, f(\mathbf{0}) + h) = P_0 + P_1 \cdot h + P_2 \cdot h^2 + \ldots + P_r \cdot h^r$$

But, we are interested in monomials of degree at most d.

Newton iteration : general case

At the end of step k-1 :

$\forall j < k, \mathrm{Deg}_j(f(\mathbf{0}) + h) = \mathrm{Deg}_j(f)$

$h = A_{k-1}(P_0, P_1, P_2, \ldots, P_d)$

$\forall j < k+1, \mathrm{Deg}_j(f(\mathbf{0}) + h') = \mathrm{Deg}_j(f)$

$h' = A_k(P_0, P_1, P_2, \ldots, P_d)$

$h' = h + \text{ monomials of degree } k$

$\mathrm{Deg}_k(P(\mathbf{x}), f(\mathbf{0}) + h + (h' - h)) = 0$

$h' = h - \mathrm{Deg}_k\left(\frac{1}{\delta}\left(P(\mathbf{X}, f(\mathbf{0}) + h)\right)\right)$

$P(\mathbf{X}, f(\mathbf{0}) + h) = P_0 + P_1 \cdot h + P_2 \cdot h^2 + \ldots + P_r \cdot h^r$

But, we are interested in monomials of degree at most d.

$\mathrm{Deg}_k\left(P(\mathbf{X}, f(\mathbf{0}) + h)\right) = \mathrm{Deg}_k\left(P_0 + P_1 \cdot h + P_2 \cdot h^2 + \ldots + P_k \cdot h^k\right)$

Newton iteration : general case

At the end of step k-1 :

$\forall j < k, \mathrm{Deg}_j(f(\mathbf{0}) + h) = \mathrm{Deg}_j(f)$ $\qquad h = A_{k-1}(P_0, P_1, P_2, \ldots, P_d)$

$\forall j < k+1, \mathrm{Deg}_j(f(\mathbf{0}) + h') = \mathrm{Deg}_j(f)$ $\qquad h' = A_k(P_0, P_1, P_2, \ldots, P_d)$

$h' = h + \text{ monomials of degree } k$

$\mathrm{Deg}_k(P(\mathbf{x}), f(\mathbf{0}) + h + (h' - h)) = 0$ $\qquad h' = h - \mathrm{Deg}_k\left(\frac{1}{\delta}(P(\mathbf{X}, f(\mathbf{0}) + h))\right)$

$P(\mathbf{X}, f(\mathbf{0}) + h) = P_0 + P_1 \cdot h + P_2 \cdot h^2 + \ldots + P_r \cdot h^r$

But, we are interested in monomials of degree at most d.

$\mathrm{Deg}_k\,(P(\mathbf{X}, f(\mathbf{0}) + h)) = \mathrm{Deg}_k\left(P_0 + P_1 \cdot h + P_2 \cdot h^2 + \ldots + P_k \cdot h^k\right)$

So, we never look beyond the first d terms. f is a function of

$P_0, P_1, \ldots, P_d$

The details we skipped

The details we skipped

- General factors, and not just roots

The details we skipped

- General factors, and not just roots
- The circuit complexity of the generators

The details we skipped

- General factors, and not just roots
- The circuit complexity of the generators
- The shape/depth of the circuit of the generators

The details we skipped

- General factors, and not just roots
- The circuit complexity of the generators
- The shape/depth of the circuit of the generators
- The degree of and circuit size of the ‘top level’ computation

The details we skipped

- General factors, and not just roots
- The circuit complexity of the generators
- The shape/depth of the circuit of the generators
- The degree of and circuit size of the ‘top level’ computation
- These can be addressed using some standard ideas (interpolation, homogenization etc.)

Summary

Summary

- VNP is closed under taking factors.

Summary

- VNP is closed under taking factors.
- Low (but growing) degree factors of small formulas, low depth circuits have small formulas, low depth circuits respectively.

Summary

- VNP is closed under taking factors.

- Low (but growing) degree factors of small formulas, low depth circuits have small formulas, low depth circuits respectively.

- Even somewhat non-trivial lower bounds for formulas, low depth circuits imply sub exponential time deterministic Identity Testing algorithms for them.

Open problems

Open problems

Factors of formulas, low depth circuits : Are formula truly closed under taking factors ? What about constant depth circuits ?

Open problems

Factors of formulas, low depth circuits : Are formula truly closed under taking factors ? What about constant depth circuits ?

Factor conjecture [Kaltofen, Burgisser] : can a degree d factor of a polynomial P with a size s circuit be computed by a circuit of size poly(s,d) ? The current bounds look like poly(s, d, degree (P)), and degree(P) could be superpoly(s).

Open problems

Factors of formulas, low depth circuits : Are formula truly closed under taking factors ? What about constant depth circuits ?

Factor conjecture [Kaltofen, Burgisser] : can a degree d factor of a polynomial P with a size s circuit be computed by a circuit of size poly(s,d) ? The current bounds look like poly(s, d, degree (P)), and degree(P) could be superpoly(s).

Arithmetic formula lower bounds : Proving better than Quadratic lower bounds for arithmetic formula ? Resulting PIT applications ?

Open problems

Factors of formulas, low depth circuits : Are formula truly closed under taking factors ? What about constant depth circuits ?

Factor conjecture [Kaltofen, Burgisser] : can a degree d factor of a polynomial P with a size s circuit be computed by a circuit of size poly(s,d) ? The current bounds look like poly(s, d, degree (P)), and degree(P) could be superpoly(s).

Arithmetic formula lower bounds : Proving better than Quadratic lower bounds for arithmetic formula ? Resulting PIT applications ?

Thank You!