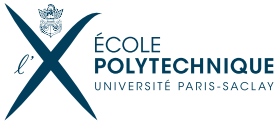
Fast polynomial reduction for generic bivariate ideals

Joris van der Hoeven, Robin Larrieu

Laboratoire d'Informatique de l'Ecole Polytechnique (LIX)



CARAMBA Seminar – Nancy
23 / 05 / 2019

Introduction

Let $\langle A, B \rangle$ be the ideal generated by A and B ($A, B \in \mathbb{K}[X, Y]$).

- Given $P \in \mathbb{K}[X, Y]$, check if $P \in \langle A, B \rangle$.
  (ideal membership test)
- Compute a normal form of $\bar{P} \in \mathbb{K}[X, Y]/\langle A, B \rangle$.
  (computation in the quotient algebra)

Introduction

Let $\langle A, B \rangle$ be the ideal generated by A and B ($A, B \in \mathbb{K}[X, Y]$).

- Given $P \in \mathbb{K}[X, Y]$, check if $P \in \langle A, B \rangle$.
  (ideal membership test)
- Compute a normal form of $\bar{P} \in \mathbb{K}[X, Y]/\langle A, B \rangle$.
  (computation in the quotient algebra)

Classical solution using *Gröbner bases*.

Introduction

Let $\langle A, B \rangle$ be the ideal generated by A and B ($A, B \in \mathbb{K}[X, Y]$).

- Given $P \in \mathbb{K}[X, Y]$, check if $P \in \langle A, B \rangle$.
  (ideal membership test)
- Compute a normal form of $\bar{P} \in \mathbb{K}[X, Y]/\langle A, B \rangle$.
  (computation in the quotient algebra)

Classical solution using *Gröbner bases*.

- Fast Gröbner basis algorithms rely on linear algebra (ex: F4, F5...)
- Can we do it with polynomial arithmetic?

Introduction

Let $\langle A, B \rangle$ be the ideal generated by A and B ($A, B \in \mathbb{K}[X, Y]$).

- Given $P \in \mathbb{K}[X, Y]$, check if $P \in \langle A, B \rangle$.
  (ideal membership test)
- Compute a normal form of $\bar{P} \in \mathbb{K}[X, Y]/\langle A, B \rangle$.
  (computation in the quotient algebra)

Classical solution using *Gröbner bases*.

- Fast Gröbner basis algorithms rely on linear algebra (ex: F4, F5...)
- Can we do it with polynomial arithmetic?
  - Given a Gröbner basis G, can we reduce P modulo G faster?

Introduction

Let $\langle A, B \rangle$ be the ideal generated by A and B ($A, B \in \mathbb{K}[X, Y]$).

- Given $P \in \mathbb{K}[X, Y]$, check if $P \in \langle A, B \rangle$.
  (ideal membership test)
- Compute a normal form of $\bar{P} \in \mathbb{K}[X, Y]/\langle A, B \rangle$.
  (computation in the quotient algebra)

Classical solution using *Gröbner bases*.

- Fast Gröbner basis algorithms rely on linear algebra (ex: F4, F5...)
- Can we do it with polynomial arithmetic?
  - Given a Gröbner basis G, can we reduce P modulo G faster?
  - Are these ideas useful to compute G faster?

Introduction

Main result

For generic ideals in two variables, reduction is possible with quasi-optimal complexity.
If A,B are given in total degree and if we use the degree-lexicographic order, then the Gröbner basis can also be computed efficiently.

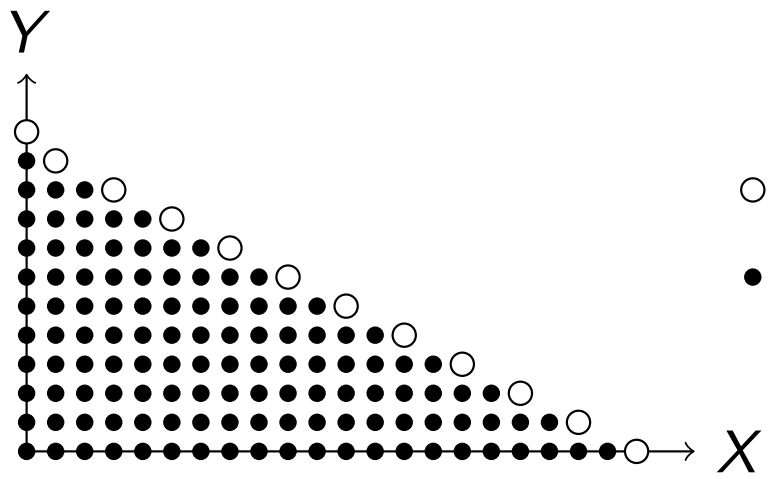
References

- van der Hoeven, L. *Fast reduction of bivariate polynomials with respect to sufficiently regular Gröbner bases* (ISSAC '18).
- van der Hoeven, L. *Fast Gröbner basis computation and polynomial reduction for generic bivariate ideals* (to appear in AAECC).

Outline

Outline

Presentation of the problem



- ○ lead. monom. of G
- • $\mathbb{K}$-basis of $\mathbb{K}[X, Y]/I$

- A, B: $O(n^2)$ coefficients
- $\mathbb{K}[X, Y]/I$: dimension $O(n^2)$
- G: $O(n^3)$ coefficients ($O(n^2)$ for each G_i)

Reduction using G needs at least $O(n^3)$ $\Longrightarrow$ reduction with less information?

Presentation of the problem

Theorem (van der Hoeven – ACA 2015)

The extended reduction of P modulo G can be computed in quasi-linear time for the size of the equation

$$P = \sum_i Q_i G_i + R$$

Presentation of the problem

Theorem (van der Hoeven – ACA 2015)

The extended reduction of P modulo G can be computed in quasi-linear time for the size of the equation

$$P = \sum_i Q_i G_i + R$$

- But this equation has size $\Theta(n^3)$ and we would like to achieve $\tilde{O}(n^2)$ complexity...

Presentation of the problem

Theorem (van der Hoeven – ACA 2015)

The extended reduction of P modulo G can be computed in quasi-linear time for the size of the equation

$$P = \sum_i Q_i G_i + R$$

- But this equation has size $\Theta(n^3)$ and we would like to achieve $\tilde{O}(n^2)$ complexity...
- $\Longrightarrow$ Somehow reduce the size of the equation.

Dichotomic selection strategy

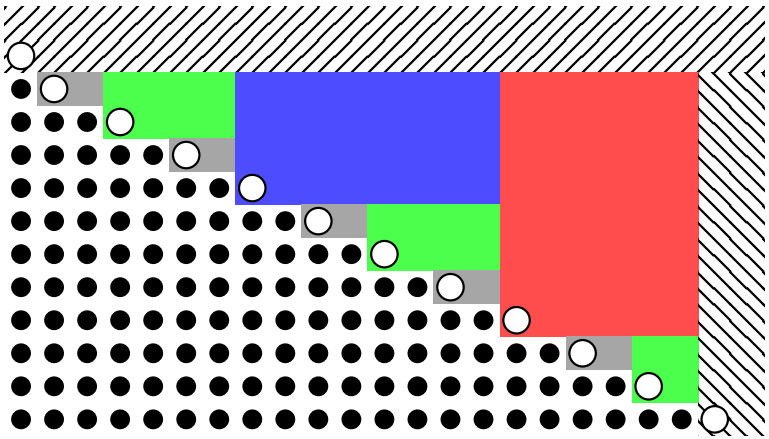
The extended reduction is not unique: several ways to reduce each term.

- The remainder is unique if G is a Gröbner basis.
- The quotients depend on a *selection strategy*.

Dichotomic selection strategy

The extended reduction is not unique: several ways to reduce each term.

- The remainder is unique if G is a Gröbner basis.
- The quotients depend on a *selection strategy*.



- $n/2$ quotients of degree d
- $n/4$ quotients of degree $2d$
- $n/8$ quotients of degree $4d$
- ...

$\Longrightarrow$ The degree of the quotients is controlled.

Truncated basis elements

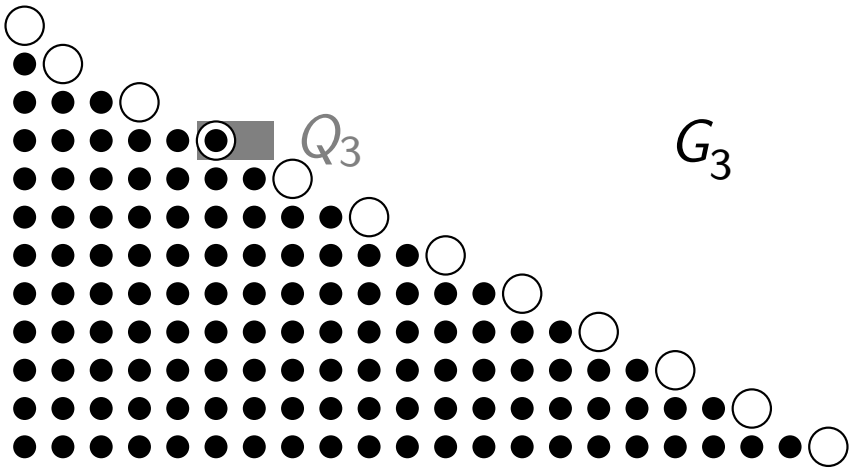
What is $125\,231\,546\,432$ quo $12\,358\,748\,151$?

Truncated basis elements

What is 125 231 546 432 quo 12 358 748 151 ?
If we know the size of the quotient, then only a few head terms are relevant.

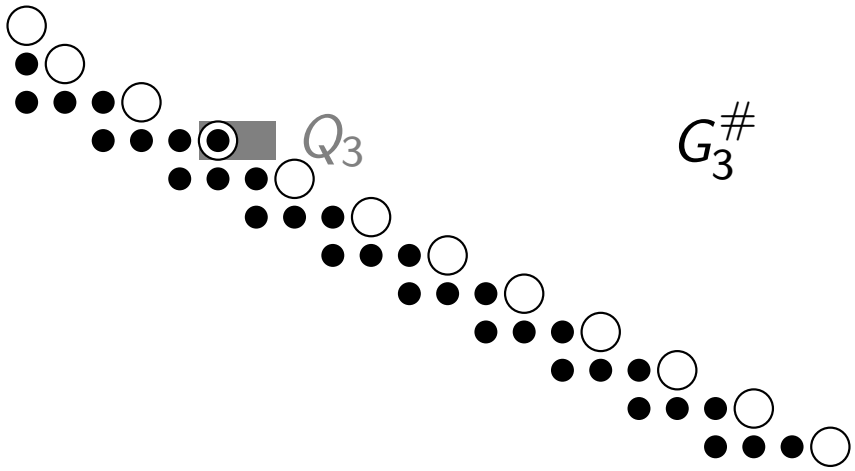
Truncated basis elements

What is 125 231 546 432 quo 12 358 748 151 ?
If we know the size of the quotient, then only a few head terms are relevant.



Truncated basis elements

What is 125 231 546 432 quo 12 358 748 151 ?
If we know the size of the quotient, then only a few head terms are relevant.



With the dichotomic selection strategy, $G^{\#} := (G_0^{\#}, \ldots, G_n^{\#})$ requires only $\tilde{O}(n^2)$ coefficients.

Rewriting the equation

$P - \sum Q_i G_i \approx P - \sum Q_i G_i^{\#}$ up to a certain precision. We need to increase this precision to continue the computation.

Rewriting the equation

$P - \sum Q_i G_i \approx P - \sum Q_i G_i^{\#}$ up to a certain precision. We need to increase this precision to continue the computation.

Remark

The Gröbner basis is generated by A and B $\implies$ redundant information.
There must be some relations between the G_i.

Rewriting the equation

$P - \sum Q_i G_i \approx P - \sum Q_i G_i^{\#}$ up to a certain precision. We need to increase this precision to continue the computation.

Remark

The Gröbner basis is generated by A and $B \implies$ redundant information.
There must be some relations between the G_i.

Assume there is $\mathcal{I}_i \subset \{0, \ldots, n\} \setminus \{i\}$ and (small) polynomials a_j such that $G_i = \sum_{j \in \mathcal{I}_i} a_j G_j$.

Rewriting the equation

$P - \sum Q_i G_i \approx P - \sum Q_i G_i^{\#}$ up to a certain precision. We need to increase this precision to continue the computation.

Remark

The Gröbner basis is generated by A and $B \implies$ redundant information.
There must be some relations between the G_i.

Assume there is $\mathcal{I}_i \subset \{0, \ldots, n\} \setminus \{i\}$ and (small) polynomials a_j such that $G_i = \sum_{j \in \mathcal{I}_i} a_j G_j$.
Assume also that for $j \in \mathcal{I}_i$, $G_j^{\#}$ has higher precision than $G_i^{\#}$.

Rewriting the equation

$P - \sum Q_i G_i \approx P - \sum Q_i G_i^{\#}$ up to a certain precision. We need to increase this precision to continue the computation.

Remark

The Gröbner basis is generated by A and $B \implies$ redundant information.
There must be some relations between the G_i.

Assume there is $\mathcal{I}_i \subset \{0, \ldots, n\} \setminus \{i\}$ and (small) polynomials a_j such that $G_i = \sum_{j \in \mathcal{I}_i} a_j G_j$.

Assume also that for $j \in \mathcal{I}_i$, $G_j^{\#}$ has higher precision than $G_i^{\#}$.

Then replacing $Q_i G_i^{\#}$ by $\sum_{j \in \mathcal{I}_i} Q_i a_j G_j^{\#}$ increases the precision.

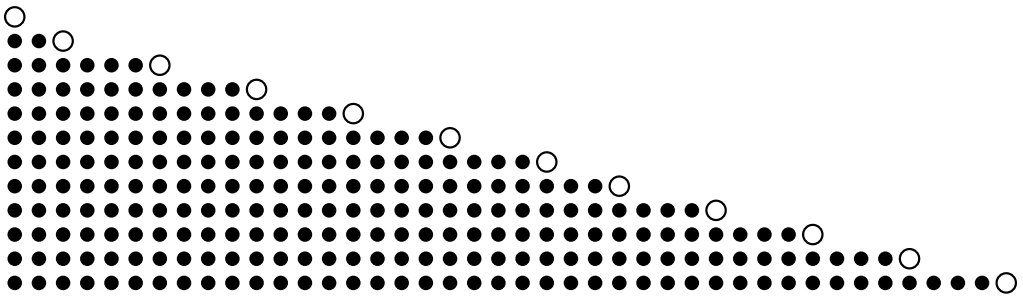
Outline

Definition: Vanilla Gröbner stairs

We consider the term orders $\prec_k$ ($k \in \mathbb{N}^*$) as the weighted-degree lexicographic order with weights $(X : 1, Y : k)$.

Vanilla Gröbner stairs

The monomials below the stairs are the minimal elements with respect to $\prec_k$



Example for $k = 4$ and an ideal I of degree $D = 237$

Definition: Retractive property

Retractive property

let $\mathcal{I} := \{0, 1, n\}$. The retractive property means that for any $i \leqslant n$ we have a linear combination

$$G_i = \sum_{j \in \mathcal{I}} C_{i,j} \; G_j \qquad .$$

Definition: Retractive property

Retractive property

For $\ell \in \mathbb{N}^*$, let $\mathcal{I}_\ell := \{0, 1, n\} \cup \ell\mathbb{N} \cap (0, n)$. The retractive property means that for any $i, \ell \leqslant n$ we have a linear combination

$$G_i = \sum_{j \in \mathcal{I}_\ell} C_{i,j,\ell} G_j \text{ with } \deg_k C_{i,j,\ell} = O(k\ell).$$

Definition: Retractive property

Retractive property

For $\ell \in \mathbb{N}^*$, let $\mathcal{I}_\ell := \{0, 1, n\} \cup \ell\mathbb{N} \cap (0, n)$. The retractive property means that for any $i, \ell \leqslant n$ we have a linear combination

$$G_i = \sum_{j \in \mathcal{I}_\ell} C_{i,j,\ell} G_j \text{ with } \deg_k C_{i,j,\ell} = O(k\ell).$$

More precisely, $\deg_k C_{i,j,\ell} < k(2\ell - 1)$.

Definition: Retractive property

Retractive property

For $\ell \in \mathbb{N}^*$, let $\mathcal{I}_\ell := \{0, 1, n\} \cup \ell\mathbb{N} \cap (0, n)$. The retractive property means that for any $i, \ell \leqslant n$ we have a linear combination

$$G_i = \sum_{j \in \mathcal{I}_\ell} C_{i,j,\ell} G_j \text{ with } \deg_k C_{i,j,\ell} = O(k\ell).$$

More precisely, $\deg_k C_{i,j,\ell} < k(2\ell - 1)$.

A Gröbner basis for the k-order is vanilla if it is a vanilla Gröbner stairs and has the retractive property.

Conjecture: vanilla Gröbner bases are generic

Experimentally, for generators chosen at random, and for various term orders, the Gröbner basis is vanilla.

Terse representation

Proposition

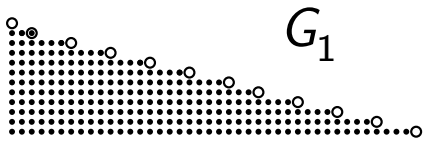
With the dichotomic selection strategy and with $p_i = \max(2^l \text{ dividing } i)$, we have $\deg_Y Q_i < p_i$ and $\deg_X Q_i < kp_i$ so $\deg_k Q_i < 2kp_i$.

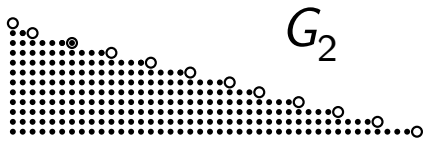
Terse representation

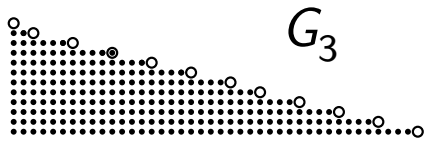
Vanilla Gröbner bases admit a terse representation constituted of:

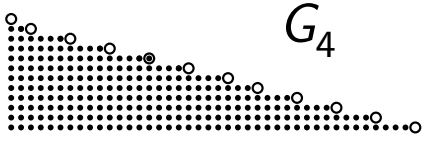
- $G_i^\# := G_i$ for $i \in \{0, 1, n\}$.
- $G_i^\#$ is the truncation at precision $2kp_i$ of G_i for $1 < i < n$.
- the retraction coefficients $C_{i,j,\ell}$ for $\ell = 2, 4, \ldots$, $j \in I_\ell$, i a multiple of $\ell/2$.

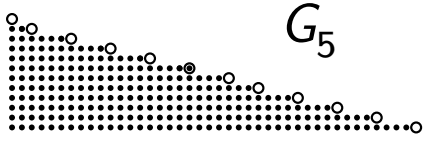
Terse representation – Example

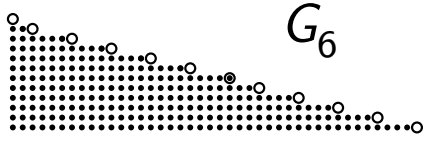


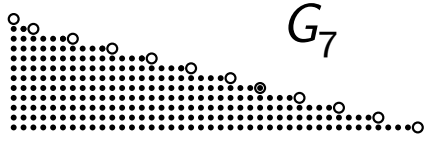


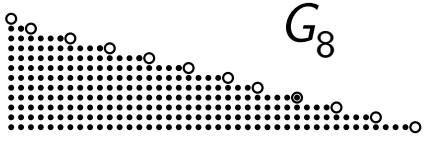


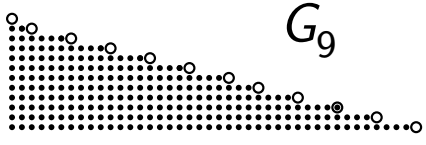


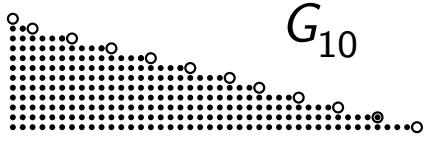


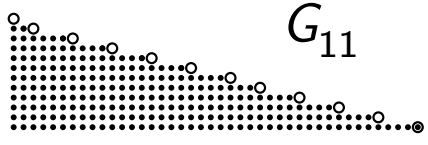




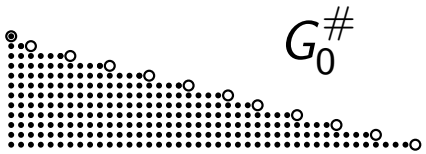


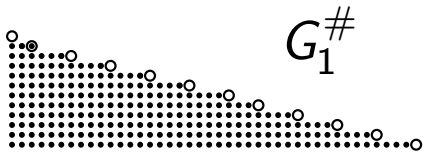


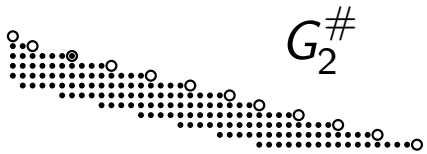




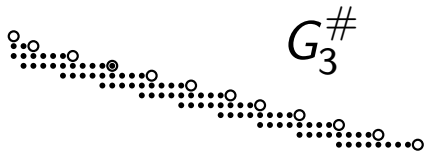
Terse representation – Example



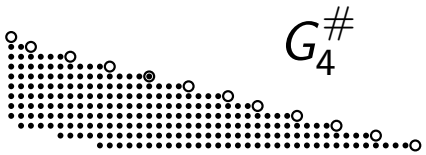




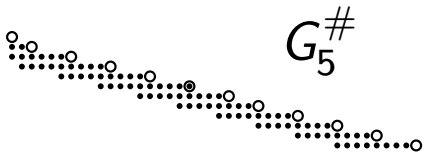
+ the linear combination
$G_2 = f_2(G_i, i \in \{0, 1, 4, 8, 11\})$
(5 polynomials of degree 27)



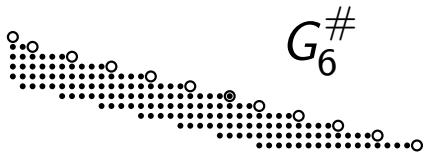
+ the linear combination
$G_3 = f_3(G_i, i \in \{0, 1, 2, 4, 6, 8, 10, 11\})$
(8 polynomials of degree 11)



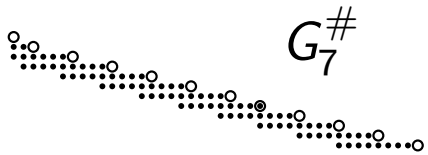
+ the linear combination
$G_4 = f_4(G_i, i \in \{0, 1, 8, 11\})$
(4 polynomials of degree 59)



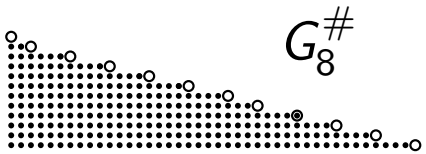
+ the linear combination
$G_5 = f_5(G_i, i \in \{0, 1, 2, 4, 6, 8, 10, 11\})$
(8 polynomials of degree 11)



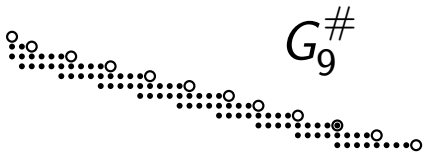
+ the linear combination
$G_6 = f_6(G_i, i \in \{0, 1, 4, 8, 11\})$
(5 polynomials of degree 27)



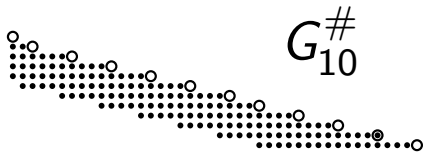
+ the linear combination
$G_7 = f_7(G_i, i \in \{0, 1, 2, 4, 6, 8, 10, 11\})$
(8 polynomials of degree 11)



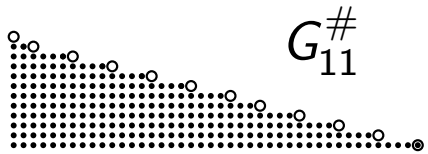
+ the linear combination
$G_8 = f_8(G_i, i \in \{0, 1, 11\})$
(3 polynomials of degree 123)



+ the linear combination
$G_9 = f_9(G_i, i \in \{0, 1, 2, 4, 6, 8, 10, 11\})$
(8 polynomials of degree 11)



+ the linear combination
$G_{10} = f_{10}(G_i, i \in \{0, 1, 4, 8, 11\})$
(5 polynomials of degree 27)



Reduction algorithm

Theorem

Let $P = \sum_i Q_i G_i^{\#} + \tilde{R}$ be an extended reduction of P w.r.t. $G^{\#}$. Then $P = \sum_i Q_i G_i$ is in normal form with respect to G.

This is because the monomials in normal form are minimal w.r.t. $\prec_k$.

Reduction algorithm

Theorem

Let $P = \sum_i Q_i G_i^{\#} + \tilde{R}$ be an extended reduction of P w.r.t. $G^{\#}$. Then $P = \sum_i Q_i G_i$ is in normal form with respect to G.

This is because the monomials in normal form are minimal w.r.t. $\prec_k$.

Algorithm

- Compute an extended reduction w.r.t. $G^{\#}$

Reduction algorithm

Theorem

Let $P = \sum_i Q_i G_i^{\#} + \tilde{R}$ be an extended reduction of P w.r.t. $G^{\#}$. Then $P = \sum_i Q_i G_i$ is in normal form with respect to G.

This is because the monomials in normal form are minimal w.r.t. $\prec_k$.

Algorithm

- Compute an extended reduction w.r.t. $G^{\#}$
- Use the retraction coefficients to find S_0, S_1, S_n such that $\sum_i Q_i G_i = S_0 G_0 + S_1 G_1 + S_n G_n$.

Reduction algorithm

Theorem

Let $P = \sum_i Q_i G_i^{\#} + \tilde{R}$ be an extended reduction of P w.r.t. $G^{\#}$. Then $P = \sum_i Q_i G_i$ is in normal form with respect to G.

This is because the monomials in normal form are minimal w.r.t. $\prec_k$.

Algorithm

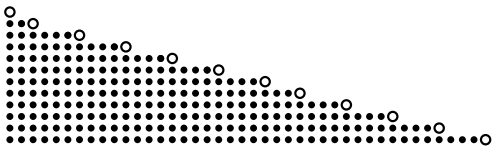
- Compute an extended reduction w.r.t. $G^{\#}$
- Use the retraction coefficients to find S_0, S_1, S_n such that $\sum_i Q_i G_i = S_0 G_0 + S_1 G_1 + S_n G_n$.
- Set $R := P - S_0 G_0 - S_1 G_1 - S_n G_n$.

Applications

Multiplication in $\mathbb{A} := \mathbb{K}[X, Y]/I$: Multiply-then-reduce in time $\tilde{O}(n^2)$.

Change of basis:

$P \in \mathbb{A}^{[4]}$



$\updownarrow$

$P \in \mathbb{A}^{[42]}$

Perform a Gröbner walk ($\log n$ steps in time $\tilde{O}(n^2)$ each).

$$\mathbb{A}^{[4]} \longleftrightarrow \mathbb{A}^{[8]} \longleftrightarrow \mathbb{A}^{[16]} \longleftrightarrow \mathbb{A}^{[32]} \longleftrightarrow \mathbb{A}^{[42]}$$

(assuming these terse representations have been precomputed).

Outline

Presentation of the setting

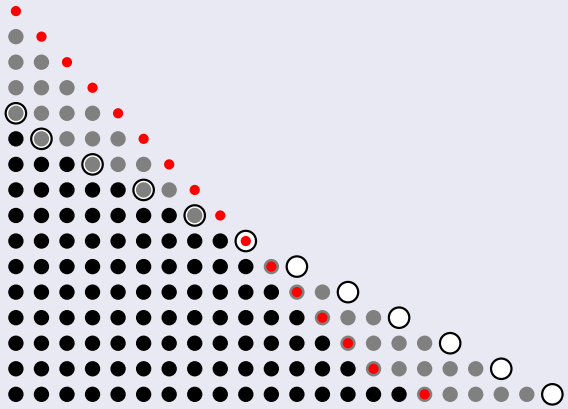
- $I = \langle A, B \rangle$ with generic $A, B \in \mathbb{K}[X, Y]$ given in total degree.
- Use the degree reverse lexicographic order to compute G.
- $\deg A = \deg B = n$ (also works if $\deg B = m \geqslant n$).
- We want to reduce P with $\deg P = d$.

Presentation of the setting

- $I = \langle A, B \rangle$ with generic $A, B \in \mathbb{K}[X, Y]$ given in total degree.
- Use the degree reverse lexicographic order to compute G.
- $\deg A = \deg B = n$ (also works if $\deg B = m \geqslant n$).
- We want to reduce P with $\deg P = d$.

Remark

In this case, G is not vanilla: the shape of the stairs do not match.



○ lead. monom. of G

• vanilla stairs

Concise representation

- Reduced Gröbner basis:
  $G_{i+2}^{\mathrm{red}} = \mathit{Spol}(G_i^{\mathrm{red}}, G_{i+1}^{\mathrm{red}}) \text{ rem } G_0^{\mathrm{red}}, \ldots, G_{i+1}^{\mathrm{red}}$

Concise representation

- Reduced Gröbner basis:
  $G_{i+2}^{\mathrm{red}} = \mathit{Spol}(G_i^{\mathrm{red}}, G_{i+1}^{\mathrm{red}}) \text{ rem } G_0^{\mathrm{red}}, \ldots, G_{i+1}^{\mathrm{red}}$
- Remark: $G_{i+2} = \mathit{Spol}(G_i, G_{i+1}) \text{ rem } G_i, G_{i+1}$ also gives a Gröbner basis.

Concise representation

- Reduced Gröbner basis:
  $G_{i+2}^{\mathrm{red}} = \mathit{Spol}(G_i^{\mathrm{red}}, G_{i+1}^{\mathrm{red}}) \text{ rem } G_0^{\mathrm{red}}, \ldots, G_{i+1}^{\mathrm{red}}$
- Remark: $G_{i+2} = \mathit{Spol}(G_i, G_{i+1}) \text{ rem } G_i, G_{i+1}$ also gives a Gröbner basis.

$$\begin{pmatrix} G_{i+1} \\ G_{i+2} \end{pmatrix} = M_i \begin{pmatrix} G_i \\ G_{i+1} \end{pmatrix}$$

Concise representation

- Reduced Gröbner basis:
  $G_{i+2}^{\mathrm{red}} = \mathit{Spol}(G_i^{\mathrm{red}}, G_{i+1}^{\mathrm{red}}) \text{ rem } G_0^{\mathrm{red}}, \ldots, G_{i+1}^{\mathrm{red}}$
- Remark: $G_{i+2} = \mathit{Spol}(G_i, G_{i+1}) \text{ rem } G_i, G_{i+1}$ also gives a Gröbner basis.

$$\begin{pmatrix} G_{i+k} \\ G_{i+k+1} \end{pmatrix} = M_{i,k} \begin{pmatrix} G_i \\ G_{i+1} \end{pmatrix}$$

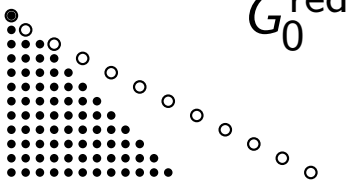
Concise representation

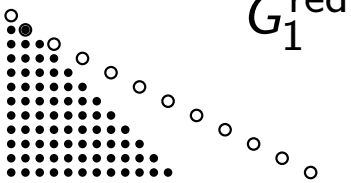
- Reduced Gröbner basis:
  $G_{i+2}^{\mathrm{red}} = \mathit{Spol}(G_i^{\mathrm{red}}, G_{i+1}^{\mathrm{red}}) \text{ rem } G_0^{\mathrm{red}}, \ldots, G_{i+1}^{\mathrm{red}}$
- Remark: $G_{i+2} = \mathit{Spol}(G_i, G_{i+1}) \text{ rem } G_i, G_{i+1}$ also gives a Gröbner basis.

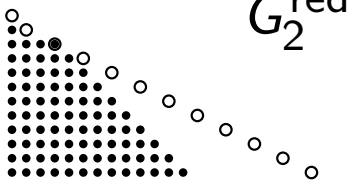
$$\begin{pmatrix} G_{i+k} \\ G_{i+k+1} \end{pmatrix} = M_{i,k} \begin{pmatrix} G_i \\ G_{i+1} \end{pmatrix}$$

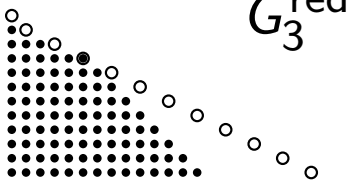
$G_0 \cong A$, $G_1 \cong B$ and well-chosen $M_{i,k}$ hold all information about G. Also, little information is required to compute the $M_{i,k}$.

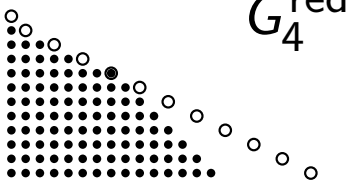
Concise representation – Example

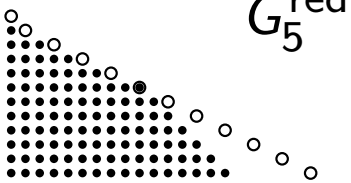
G_0^{red} 

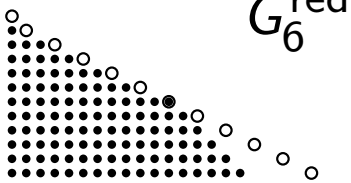
G_1^{red} 

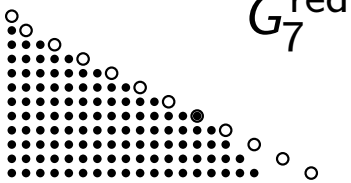
G_2^{red} 

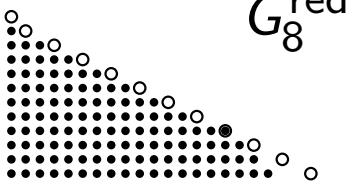
G_3^{red} 

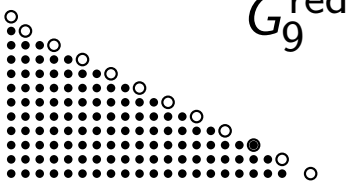
G_4^{red} 

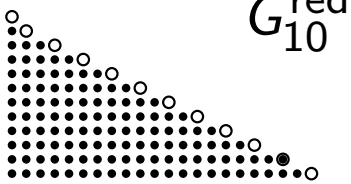
G_5^{red} 

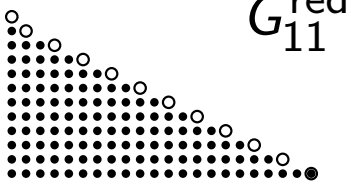
G_6^{red} 

G_7^{red} 

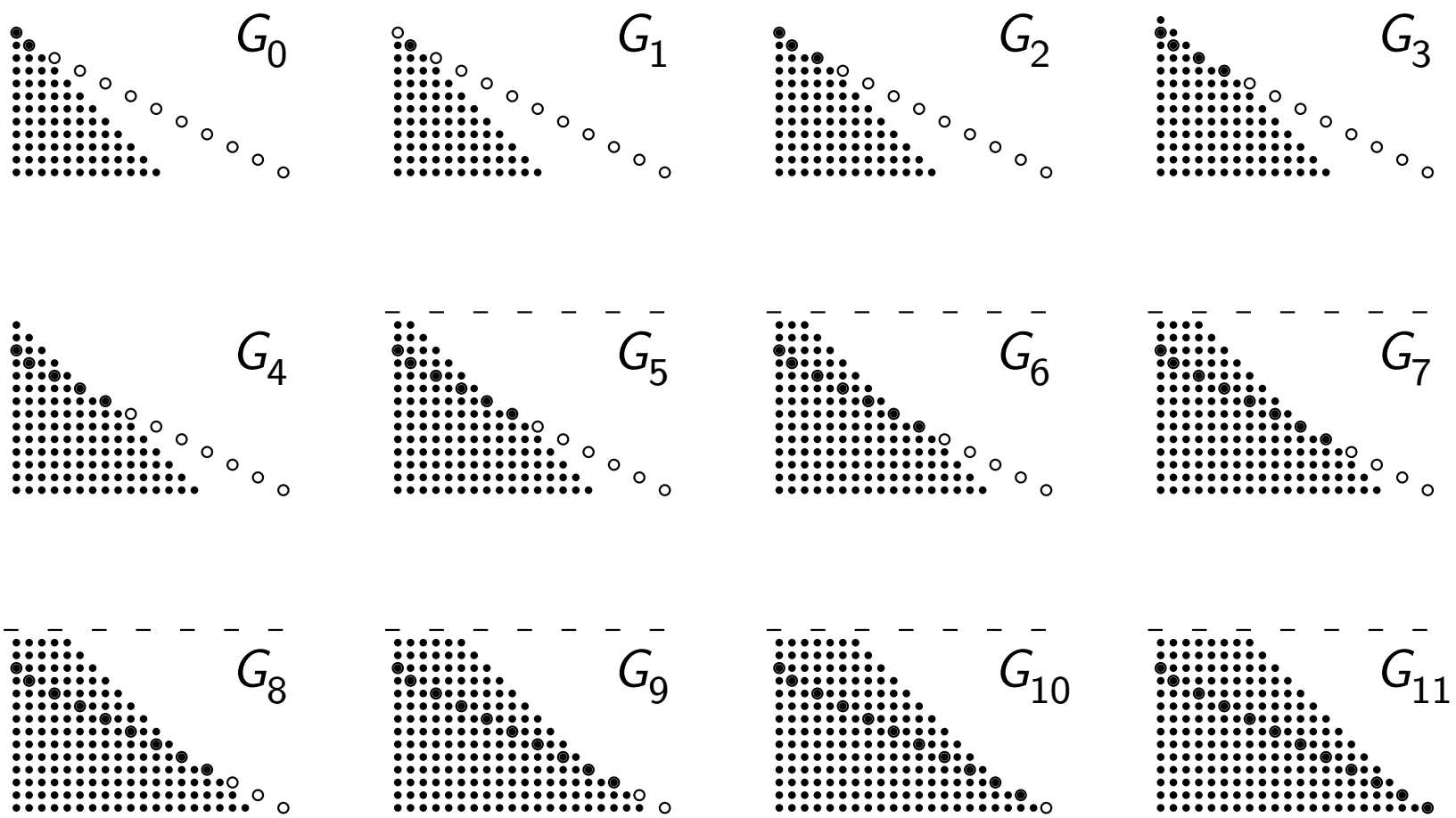
G_8^{red} 

G_9^{red} 

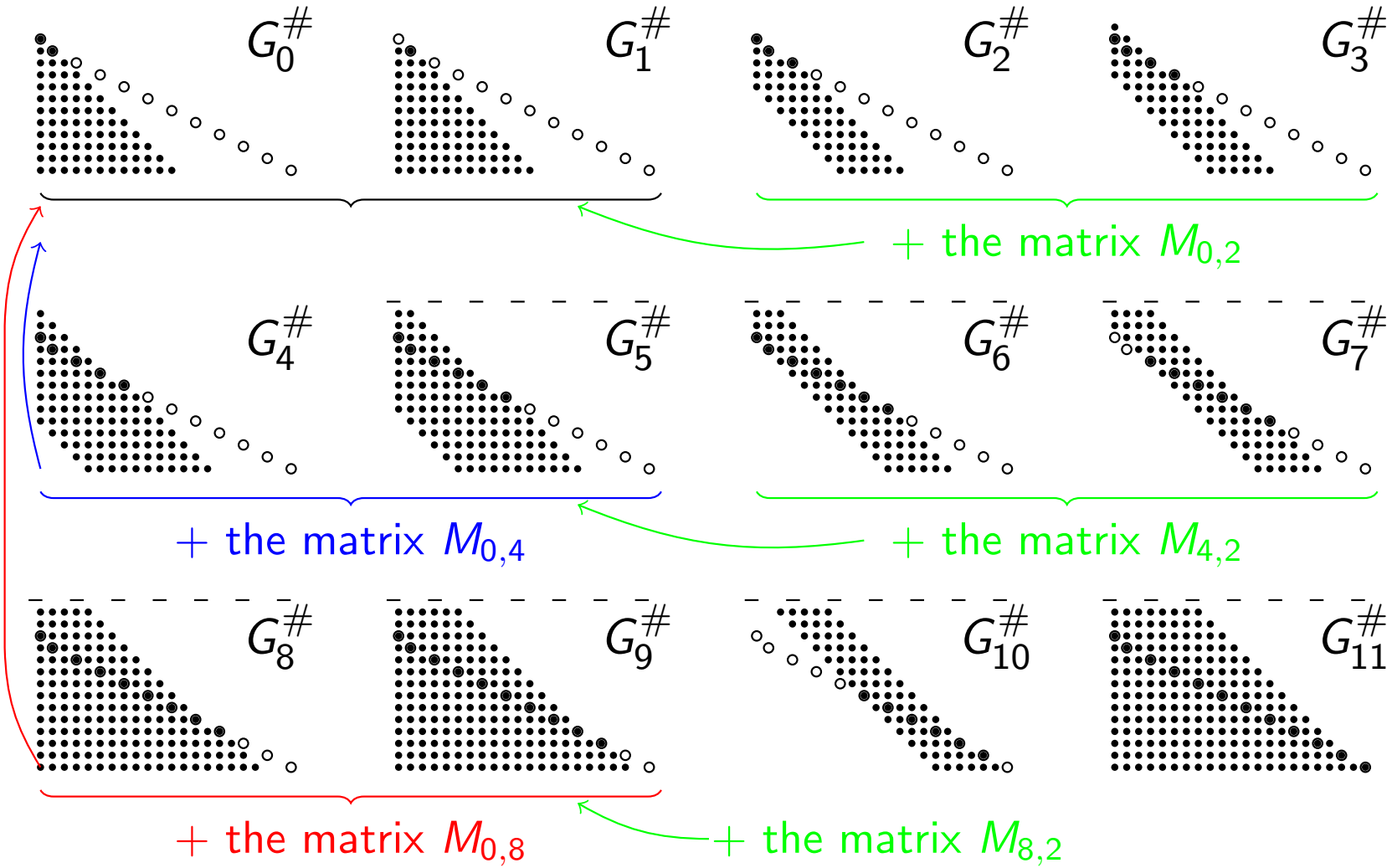
G_{10}^{red} 

G_{11}^{red} 

Concise representation – Example



Concise representation – Example



Reduction algorithm

Reminder

⚠ monomials in normal form are **NOT** the minimal monomials w.r.t. $\prec$

$\Longrightarrow$ Cannot simply reduce w.r.t. $G^{\#}$.

Reduction algorithm

Reminder

⚠ monomials in normal form are **NOT** the minimal monomials w.r.t. $\prec$

$\Longrightarrow$ Cannot simply reduce w.r.t. $G^{\#}$.

Must perform the substitutions on the fly during the computation

- Start reducing using $G_i^{\#}$.

Reduction algorithm

Reminder

⚠ monomials in normal form are **NOT** the minimal monomials w.r.t. $\prec$

$\Longrightarrow$ Cannot simply reduce w.r.t. $G^{\#}$.

Must perform the substitutions on the fly during the computation

- Start reducing using $G_i^{\#}$.
- The precision of $G_i^{\#}$ is chosen (by definition) sufficient to compute Q_i.

Reduction algorithm

Reminder

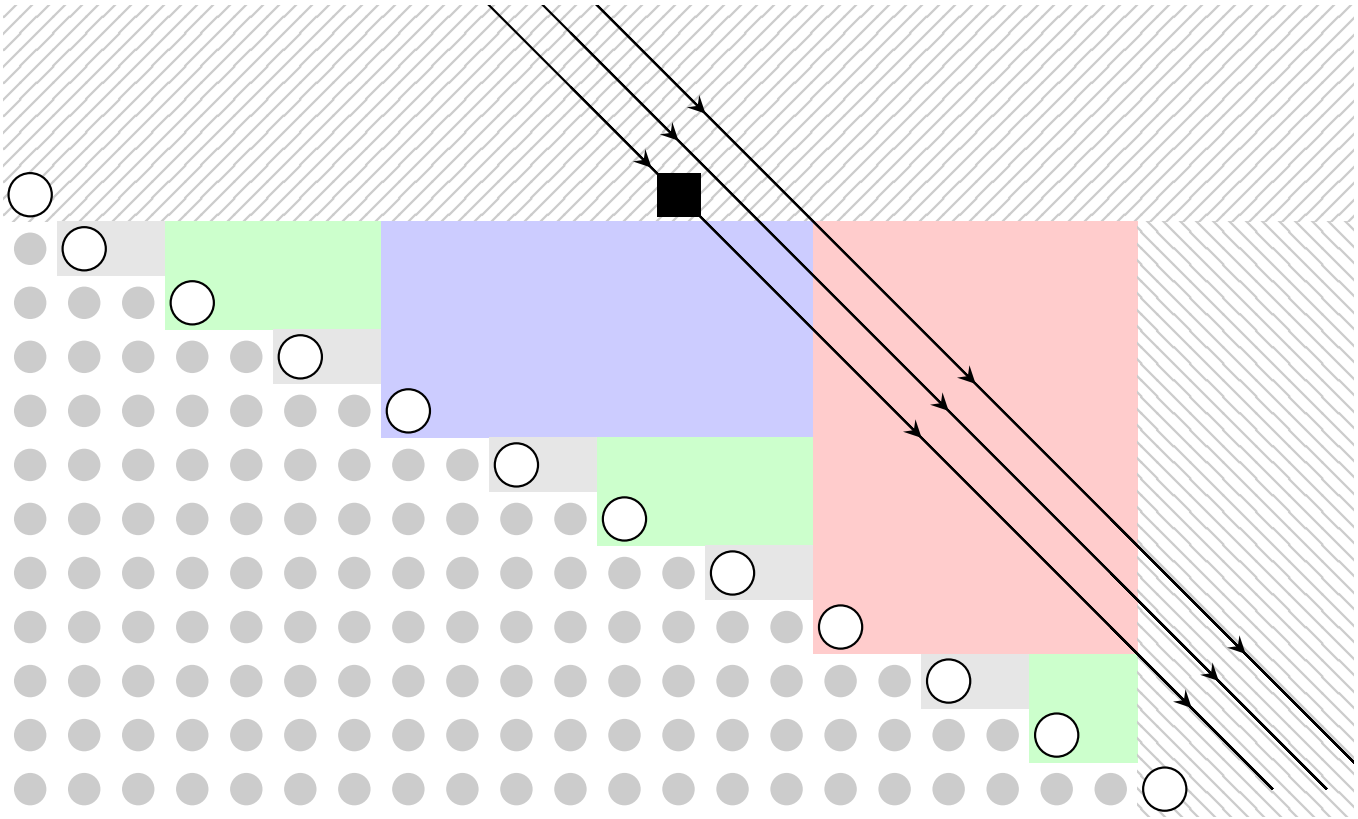
⚠ monomials in normal form are **NOT** the minimal monomials w.r.t. $\prec$

$\Longrightarrow$ Cannot simply reduce w.r.t. $G^{\#}$.

Must perform the substitutions on the fly during the computation

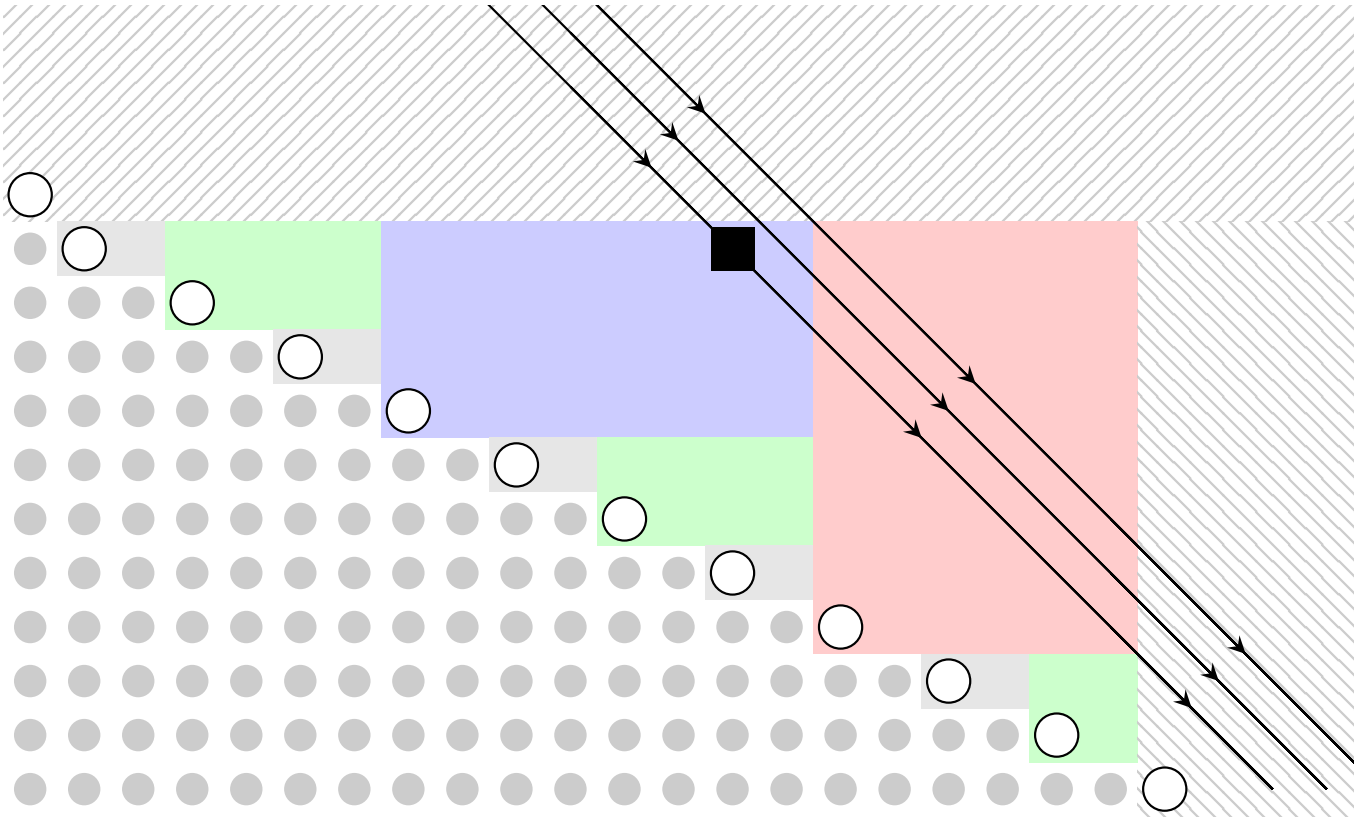
- Start reducing using $G_i^{\#}$.
- The precision of $G_i^{\#}$ is chosen (by definition) sufficient to compute Q_i.
- Once Q_i is known, replace $Q_i G_i$ by $S_k G_k + S_{k+1} G_{k+1}$ to increase precision.

Reduction algorithm – Example



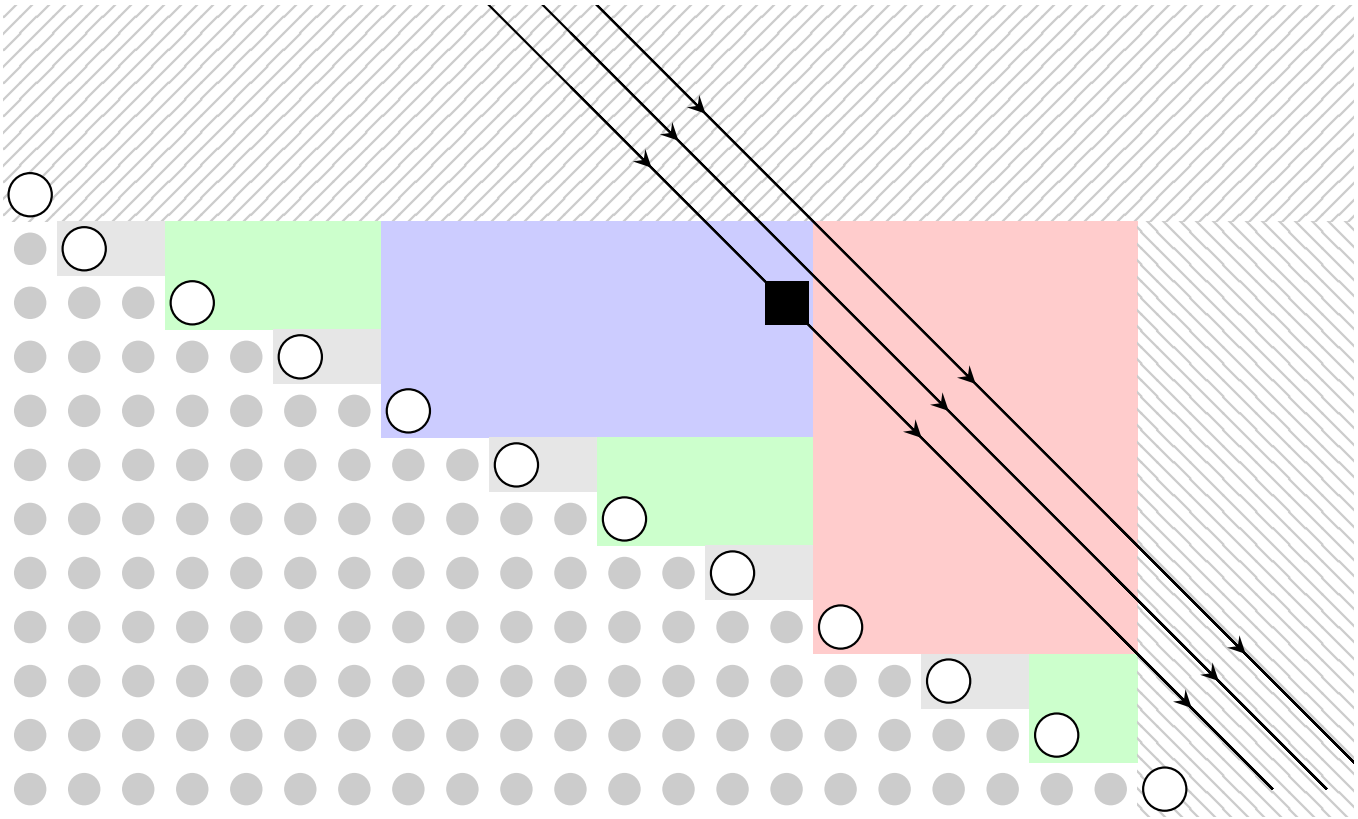
$$P = \bullet X^{12} Y^{11} + \cdots$$

Reduction algorithm – Example



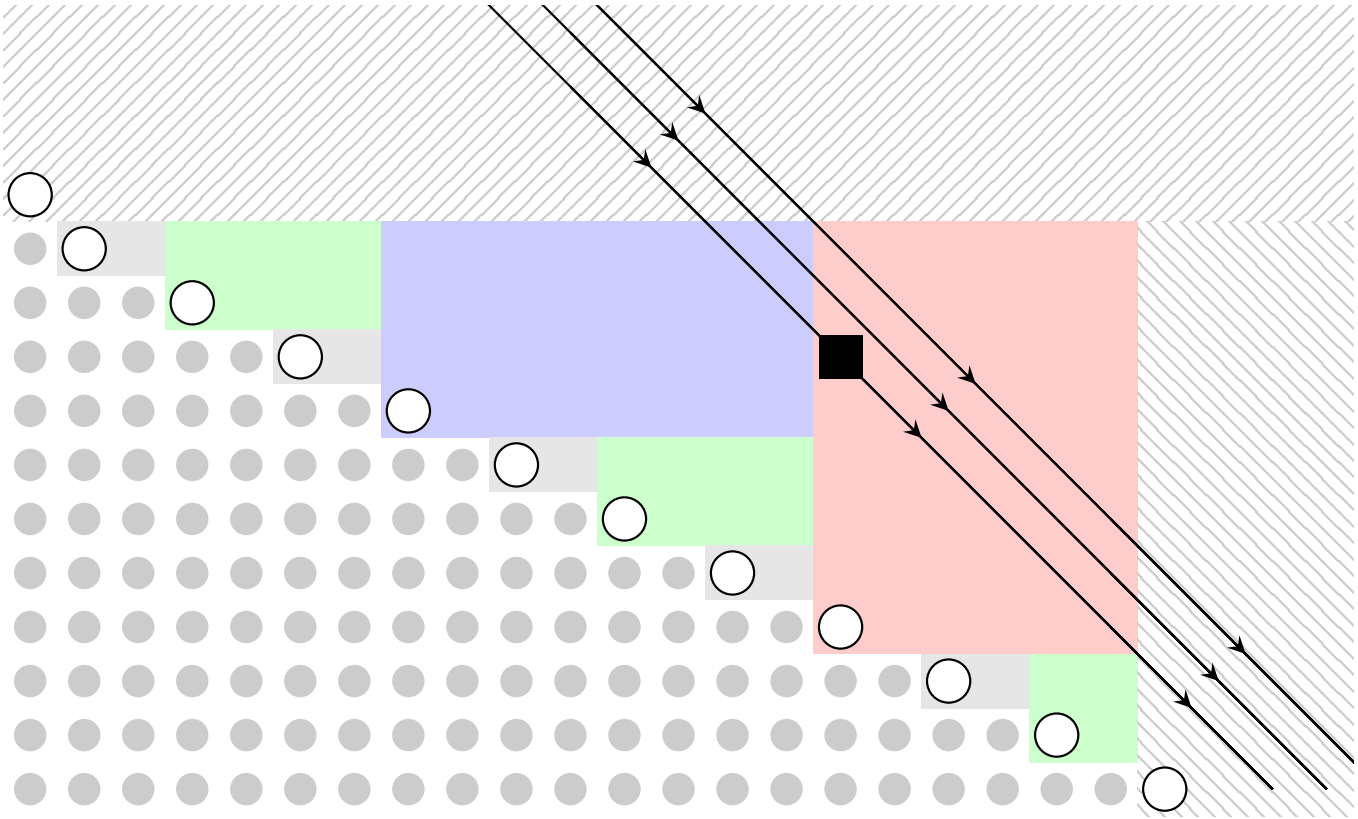
$$P = \bullet X^{13} Y^{10} + \cdots$$

Reduction algorithm – Example



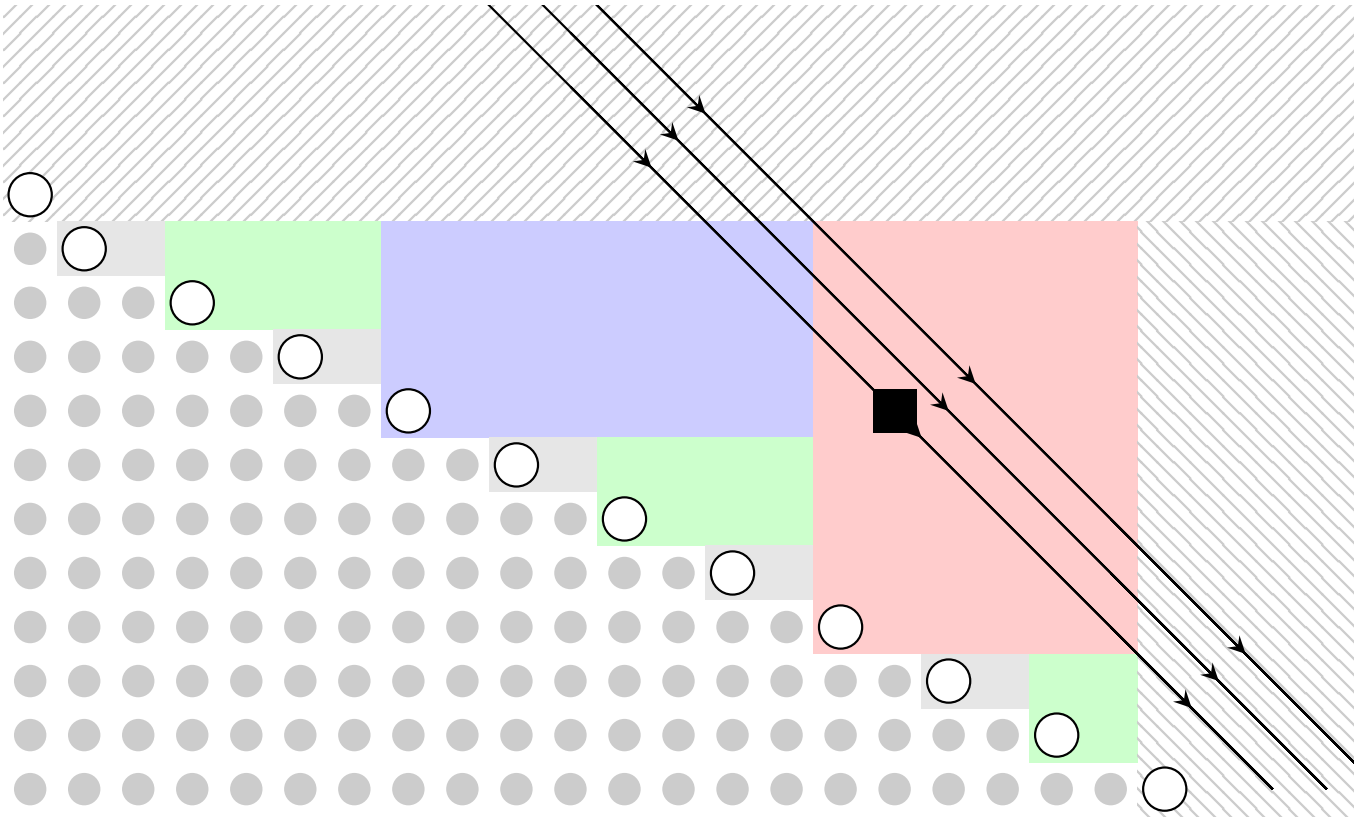
$$P = \bullet X^{14} Y^9 + \cdots$$

Reduction algorithm – Example



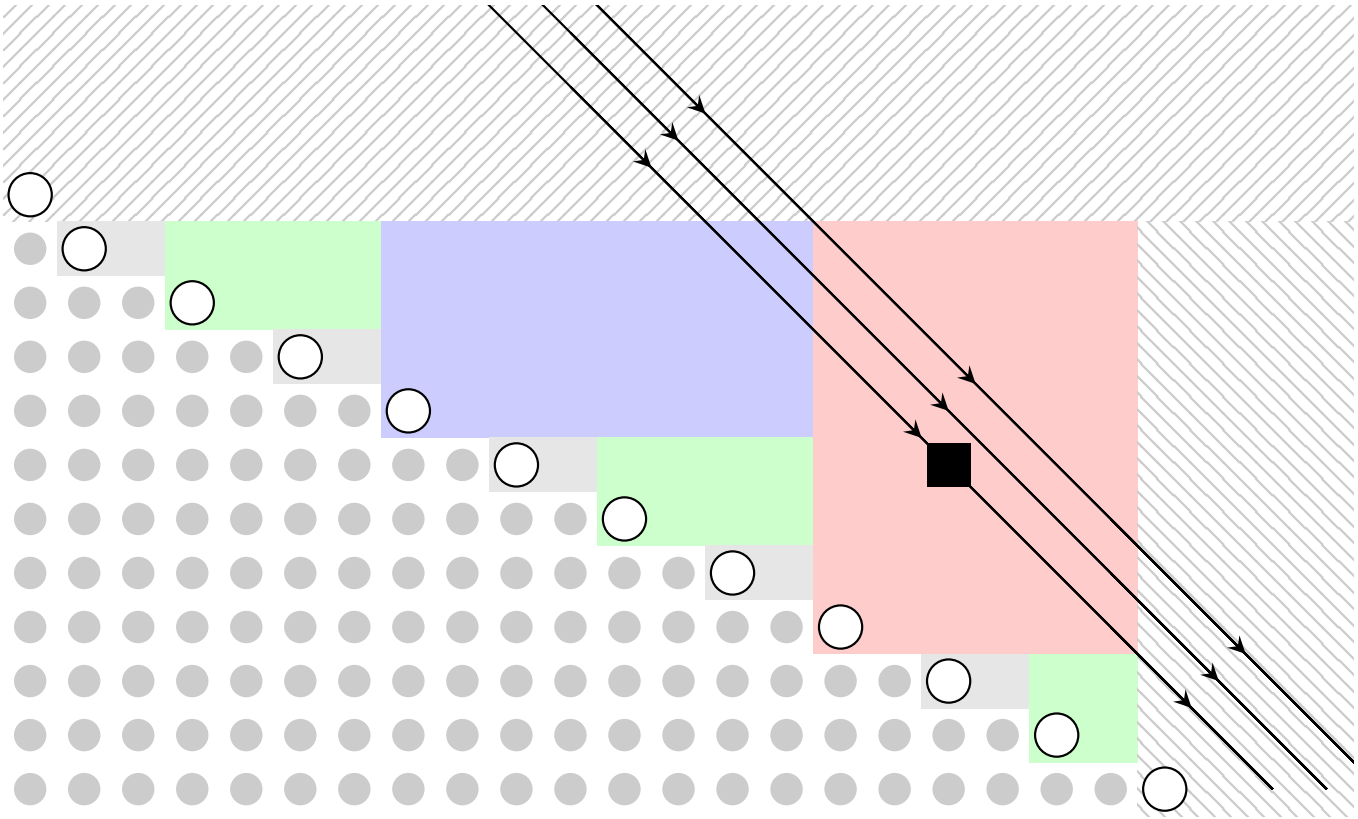
$$P = \bullet X^{15} Y^{8} + \cdots$$

Reduction algorithm – Example



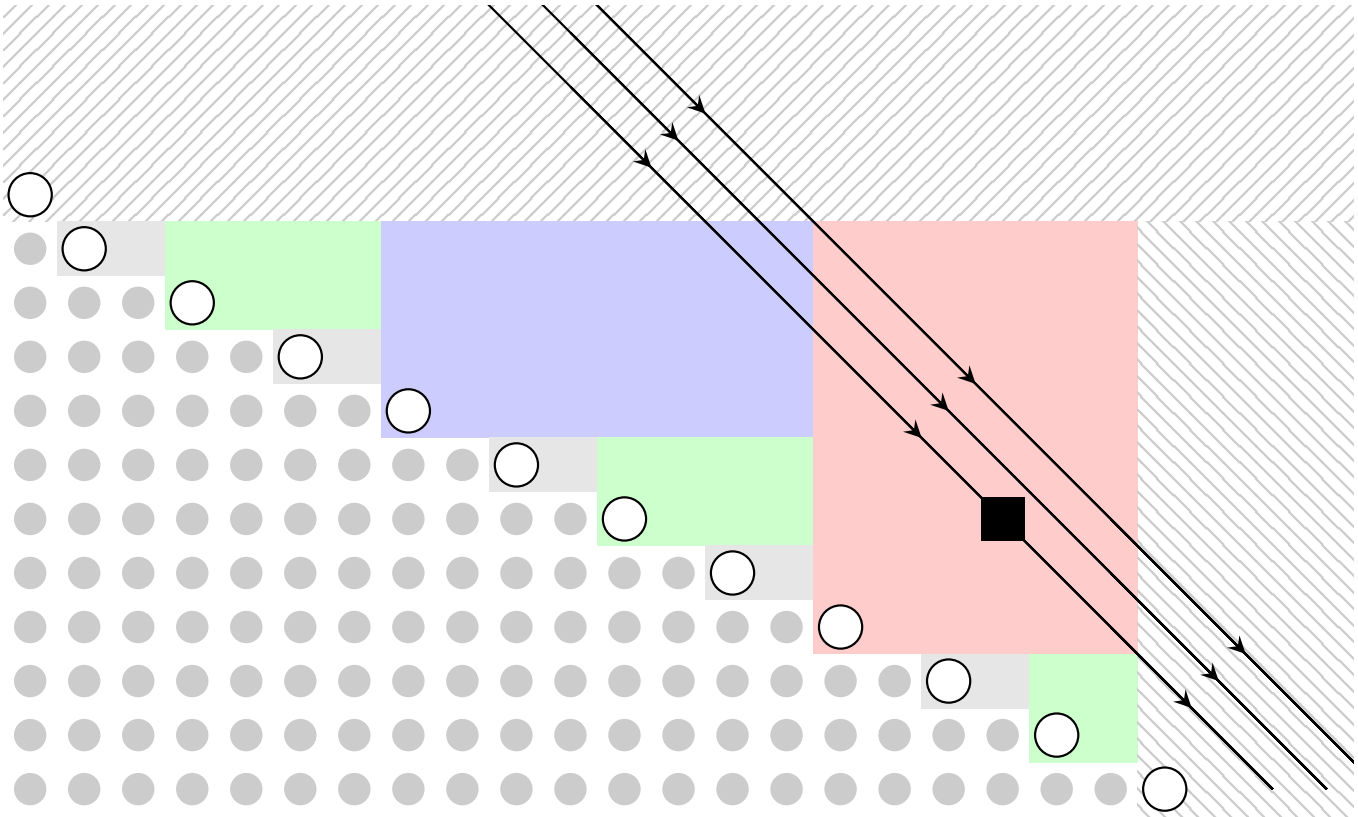
$$P = \bullet X^{16} Y^7 + \cdots$$

Reduction algorithm – Example



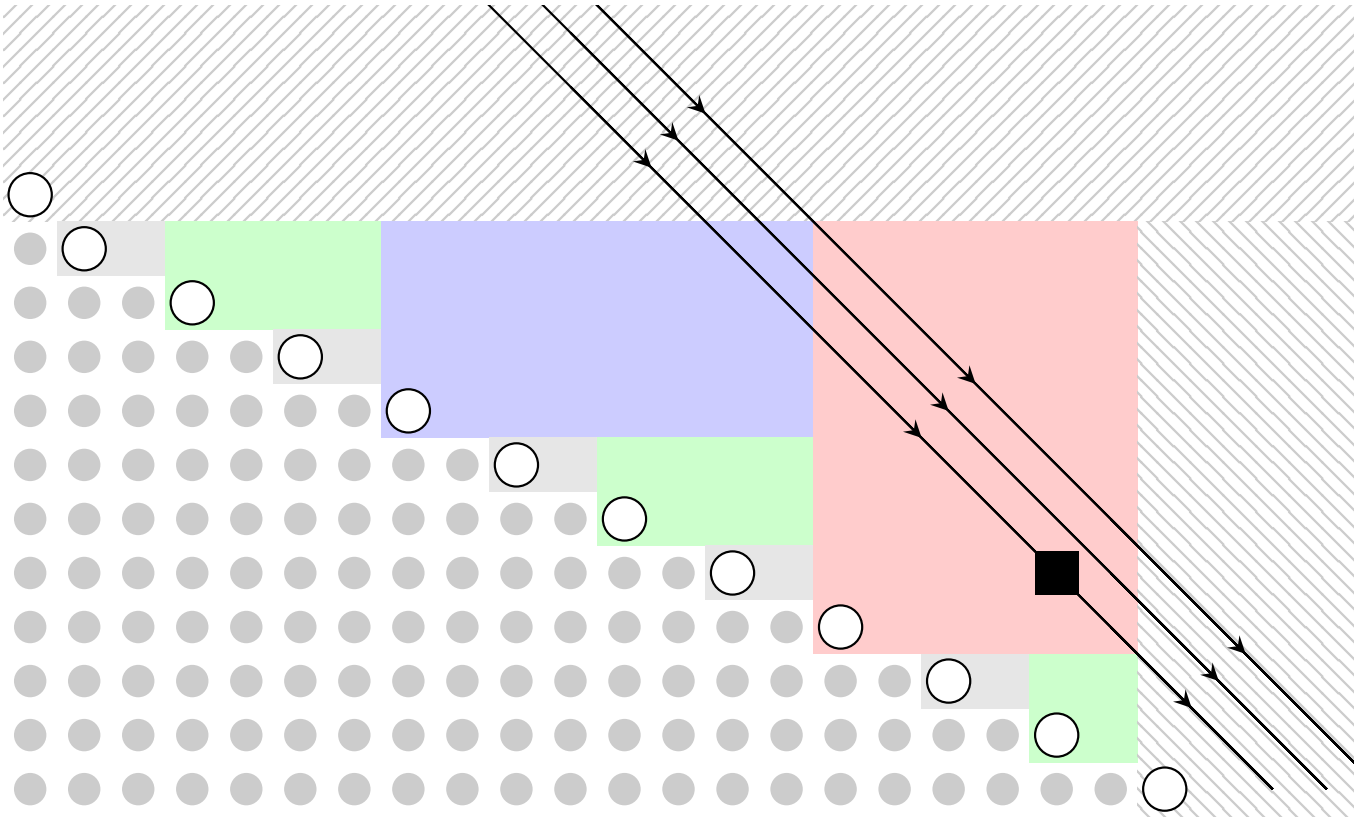
$$P = \bullet X^{17} Y^6 + \cdots$$

Reduction algorithm – Example



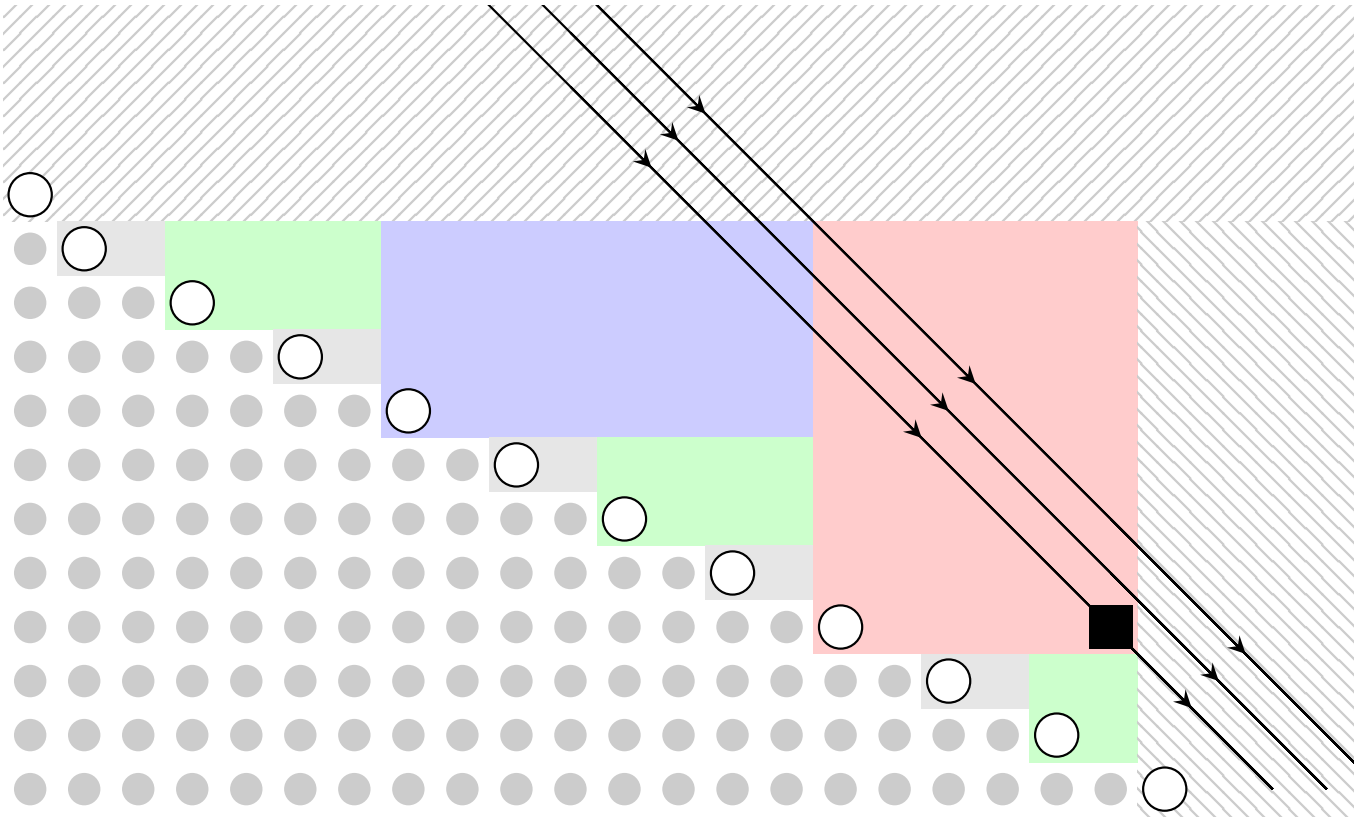
$$P = \bullet X^{18} Y^5 + \cdots$$

Reduction algorithm – Example



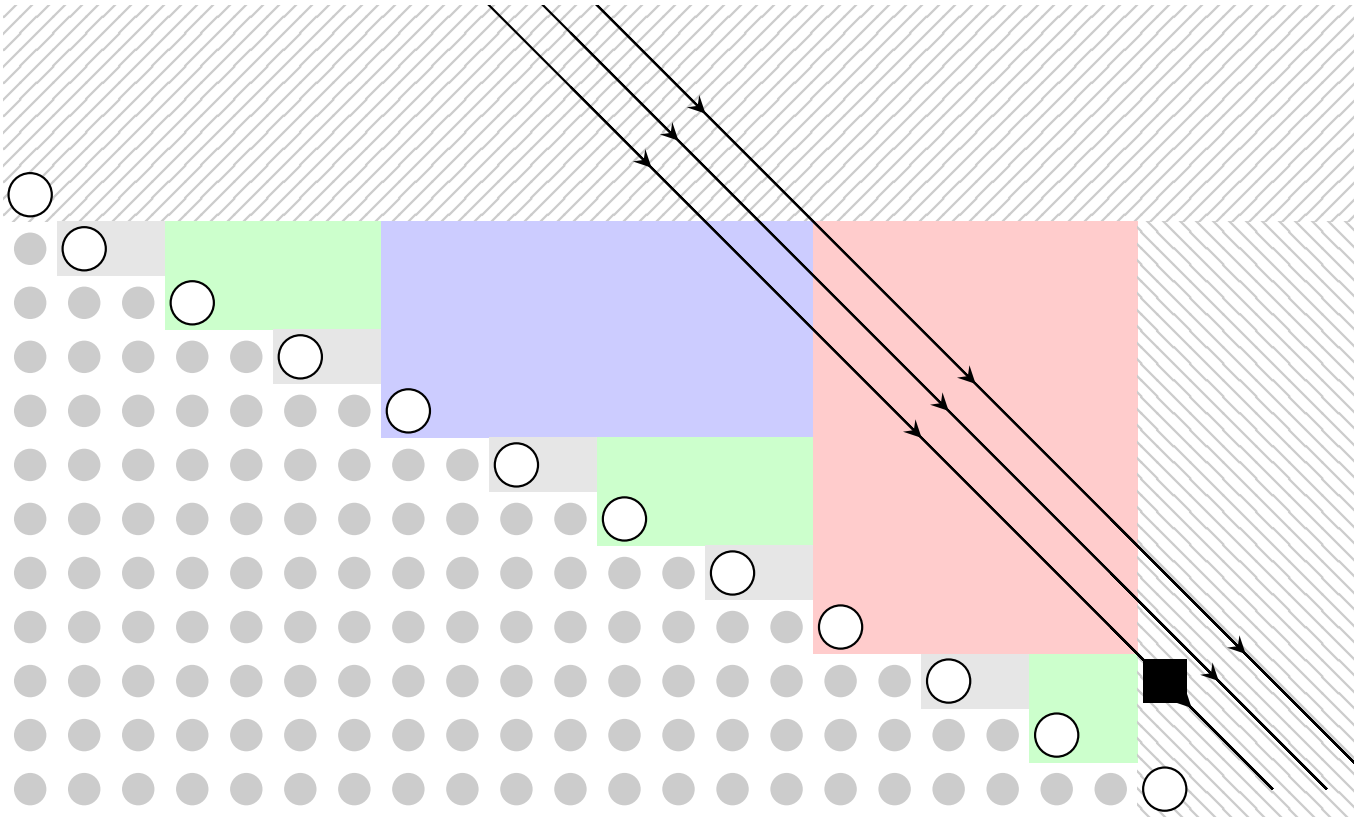
$$P = \bullet X^{19} Y^4 + \cdots$$

Reduction algorithm – Example



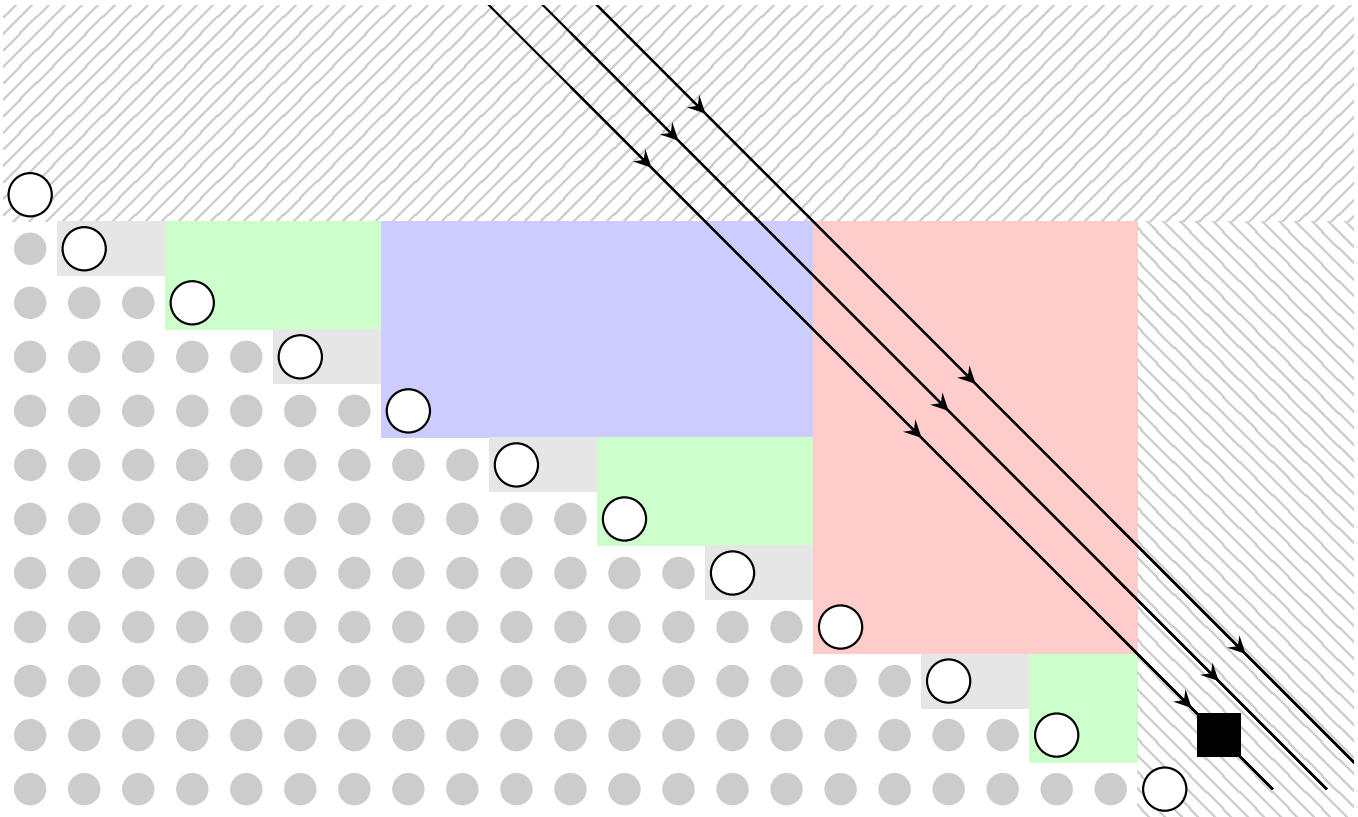
$$P = \bullet X^{20} Y^3 + \cdots$$

Reduction algorithm – Example



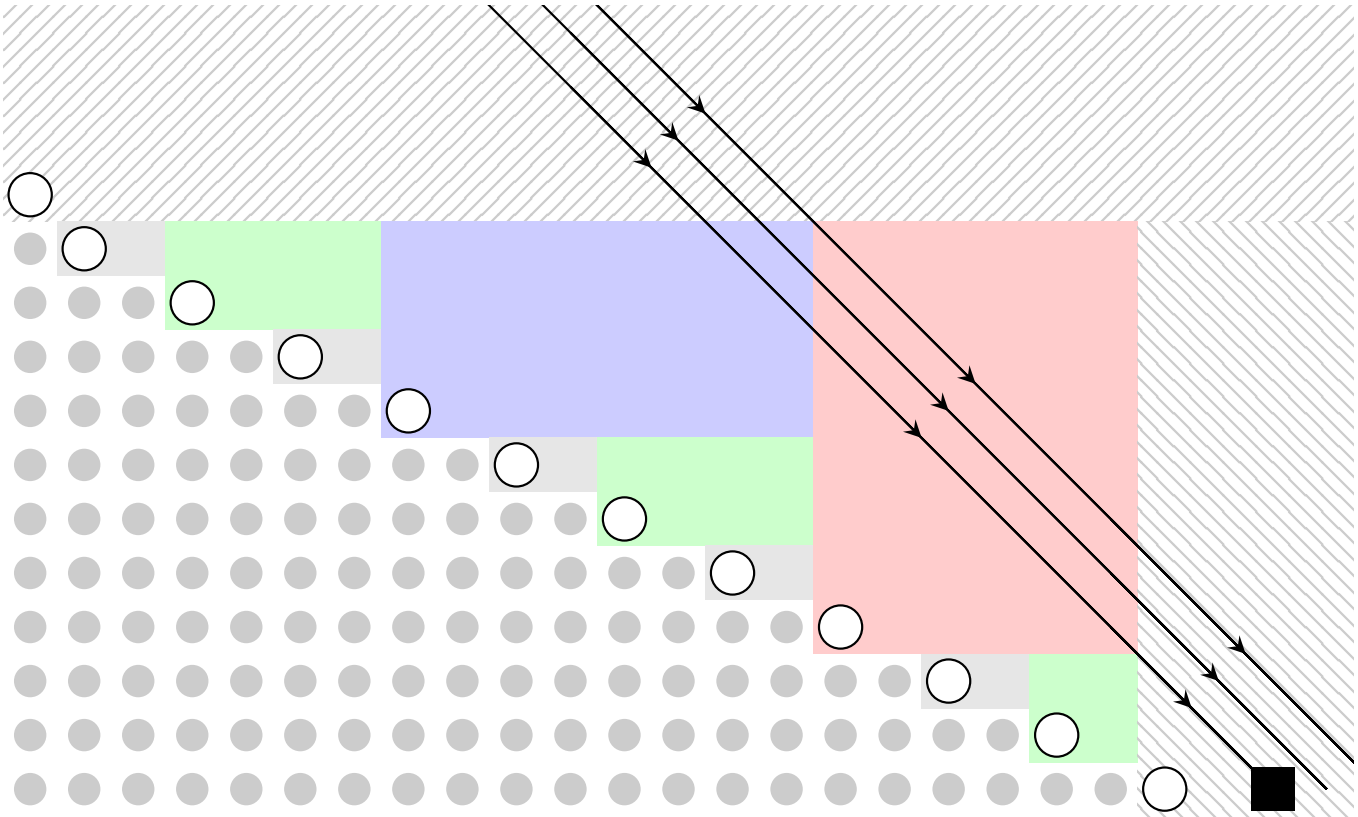
$$P = \bullet X^{21} Y^2 + \cdots$$

Reduction algorithm – Example



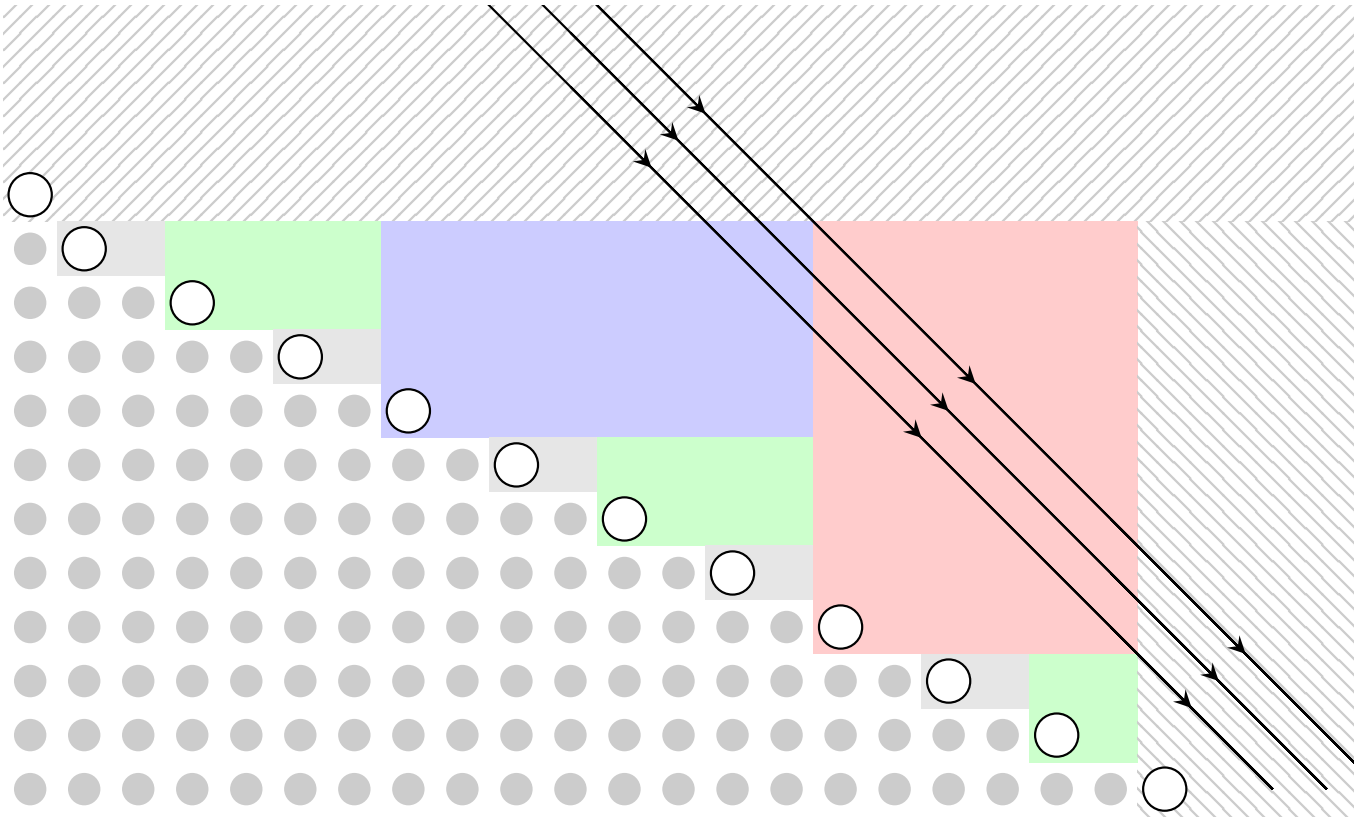
$$P = \bullet X^{22} Y^{1} + \cdots$$

Reduction algorithm – Example



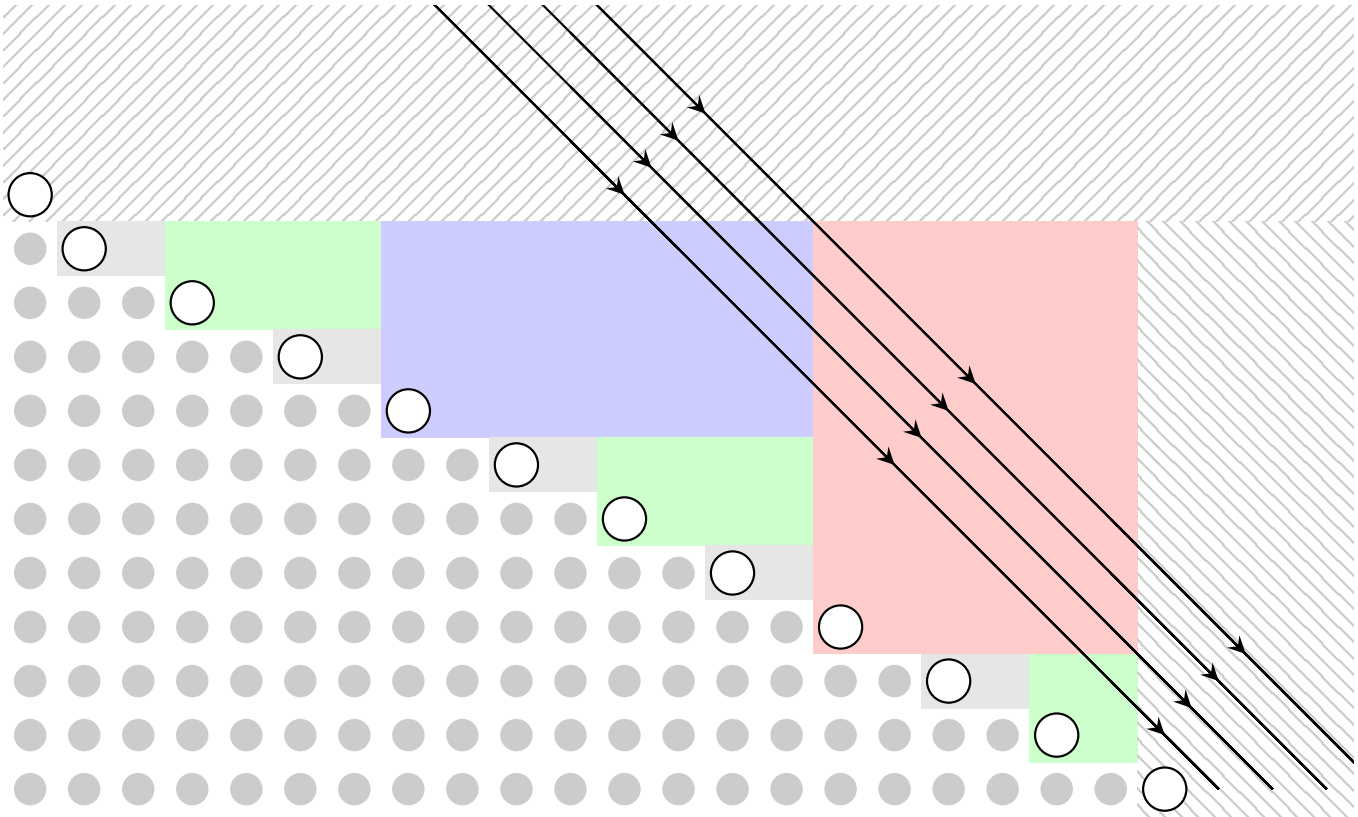
$$P = \bullet X^{23} Y^{0} + \cdots$$

Reduction algorithm – Example



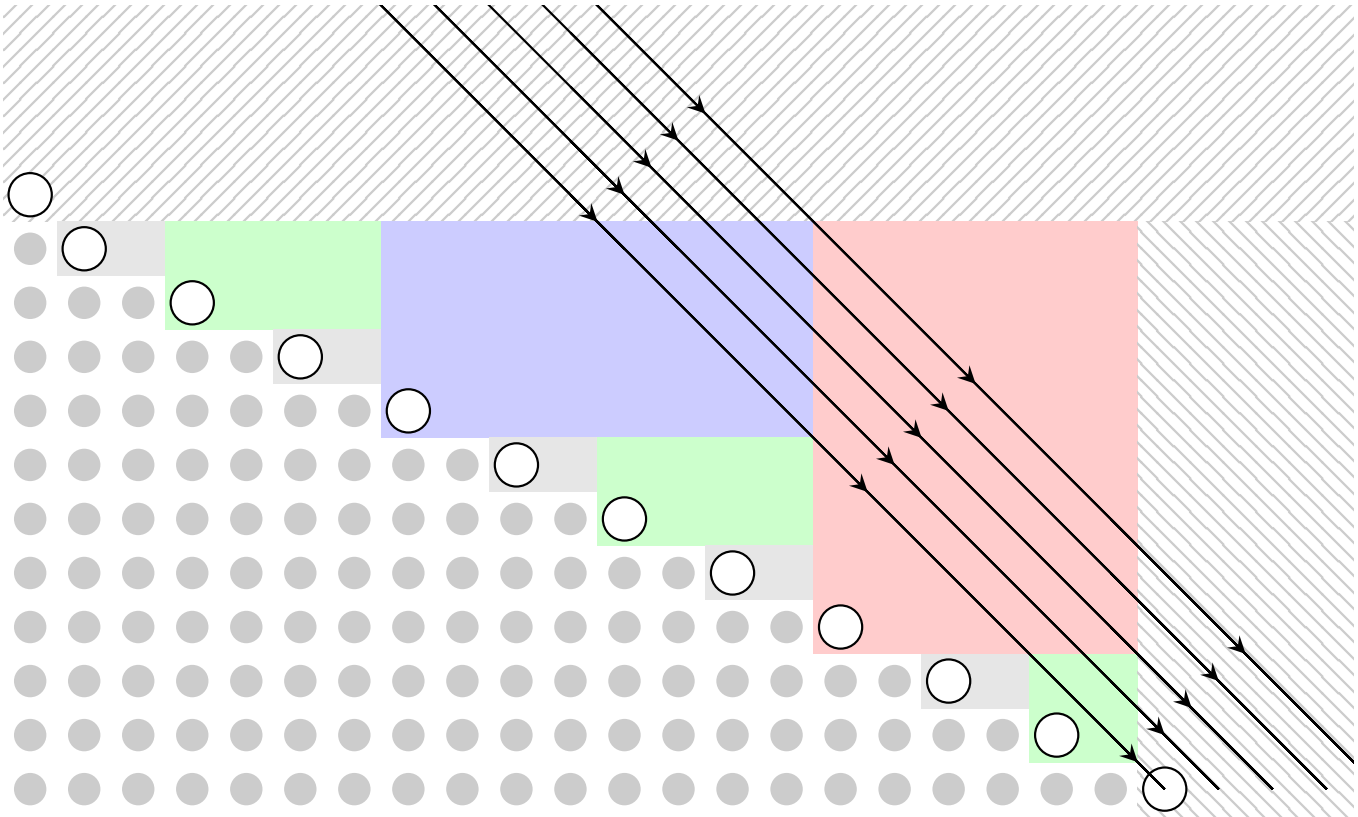
$$P = \bullet Y^{22} + \cdots$$

Reduction algorithm – Example



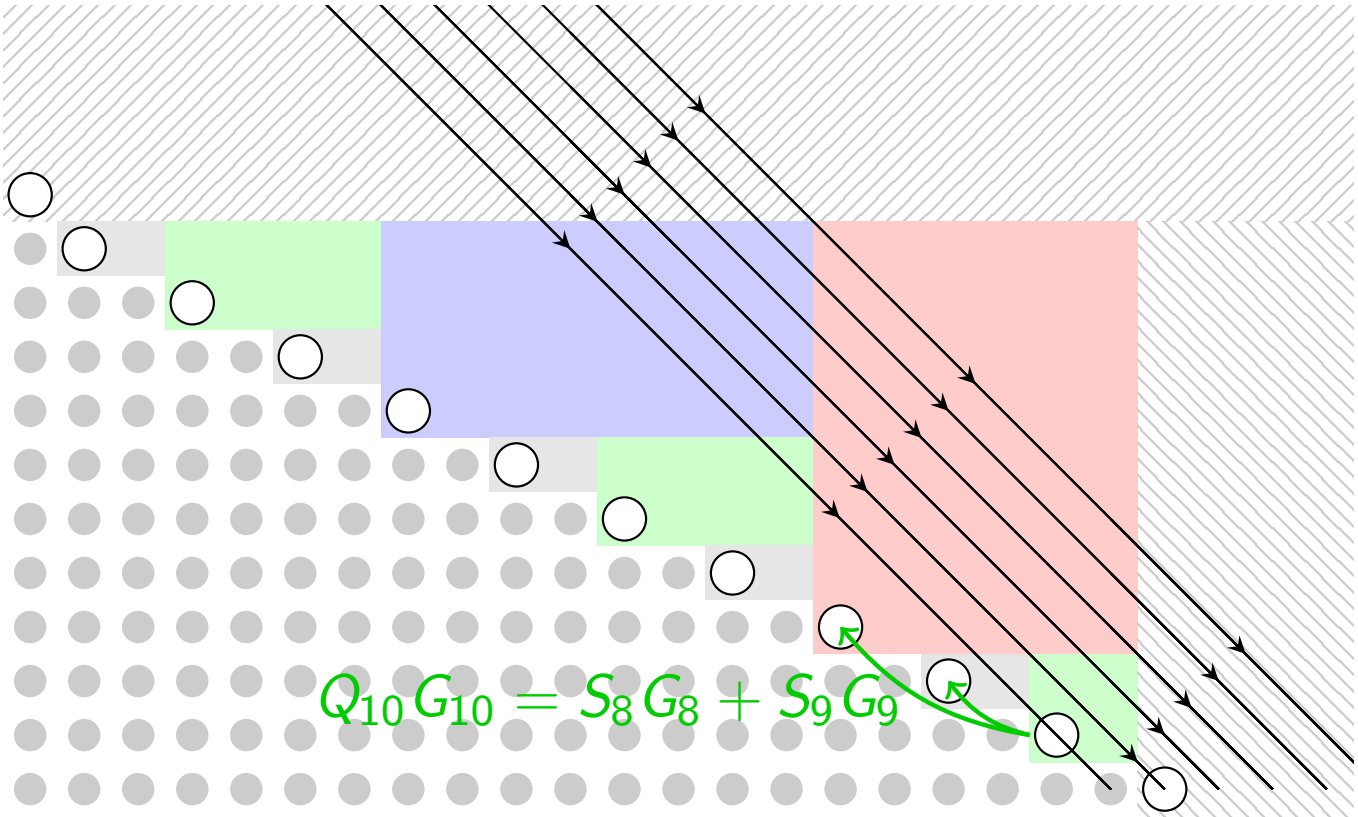
$$P = \bullet Y^{21} + \cdots$$

Reduction algorithm – Example



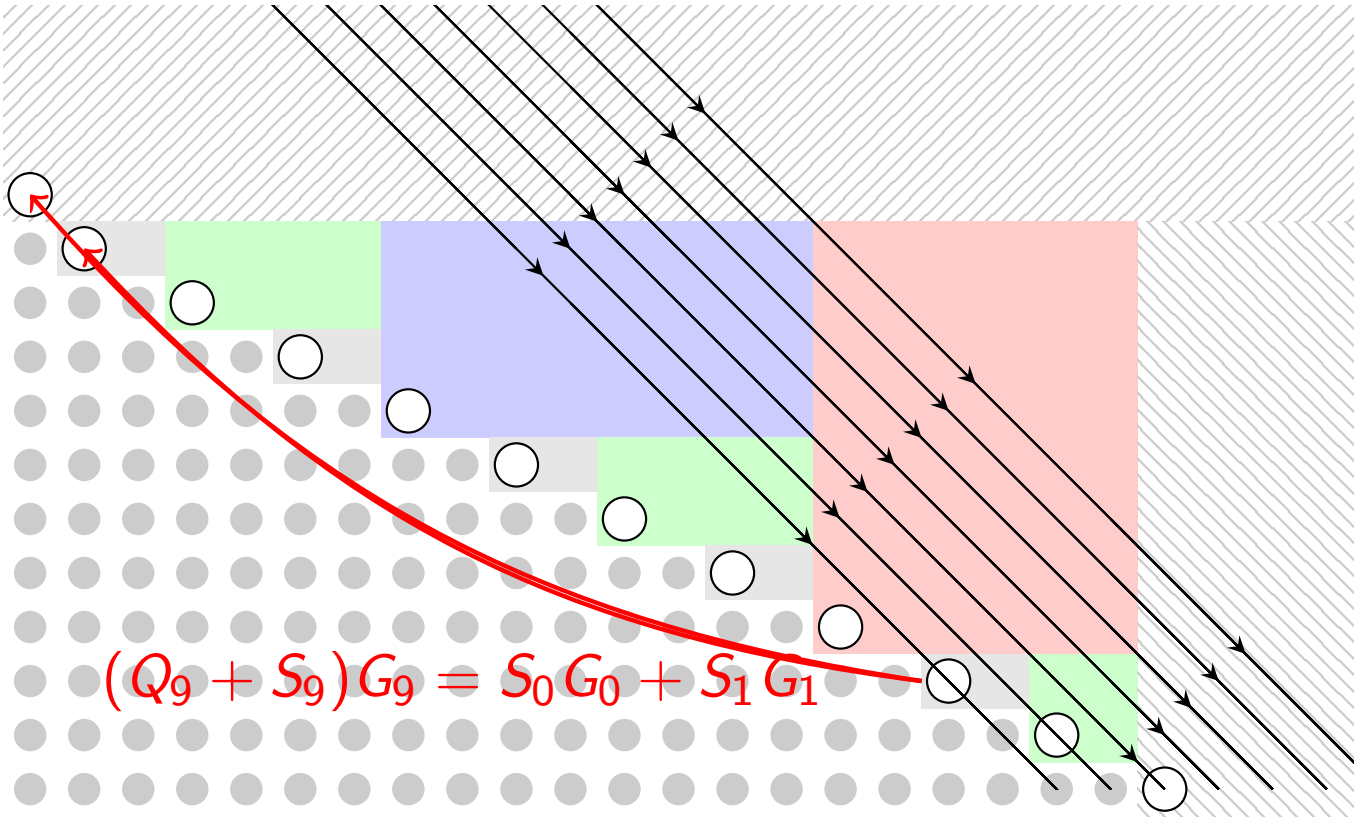
$$P = \bullet Y^{20} + \cdots$$

Reduction algorithm – Example



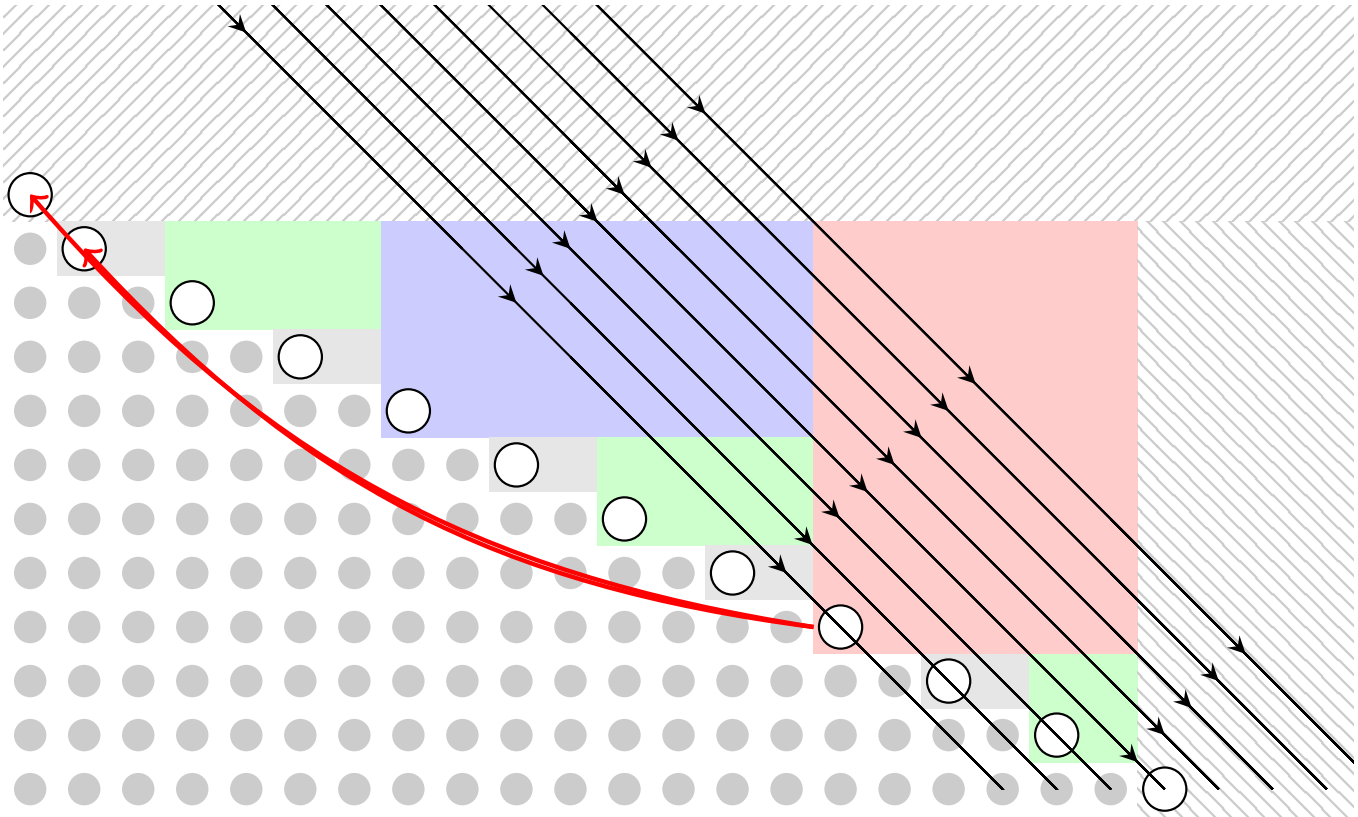
$$P = \bullet Y^{19} + \cdots$$

Reduction algorithm – Example



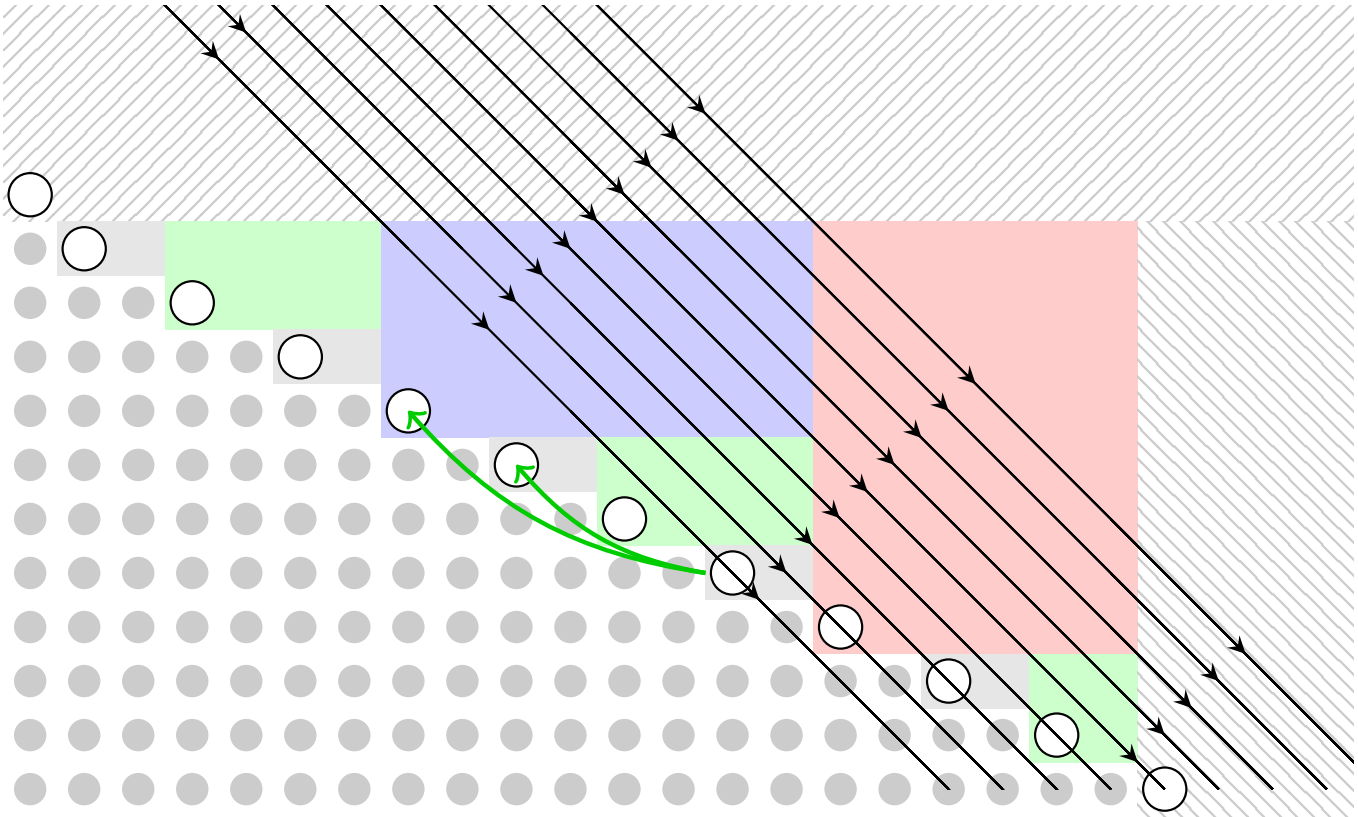
$$P = \bullet Y^{18} + \cdots$$

Reduction algorithm – Example



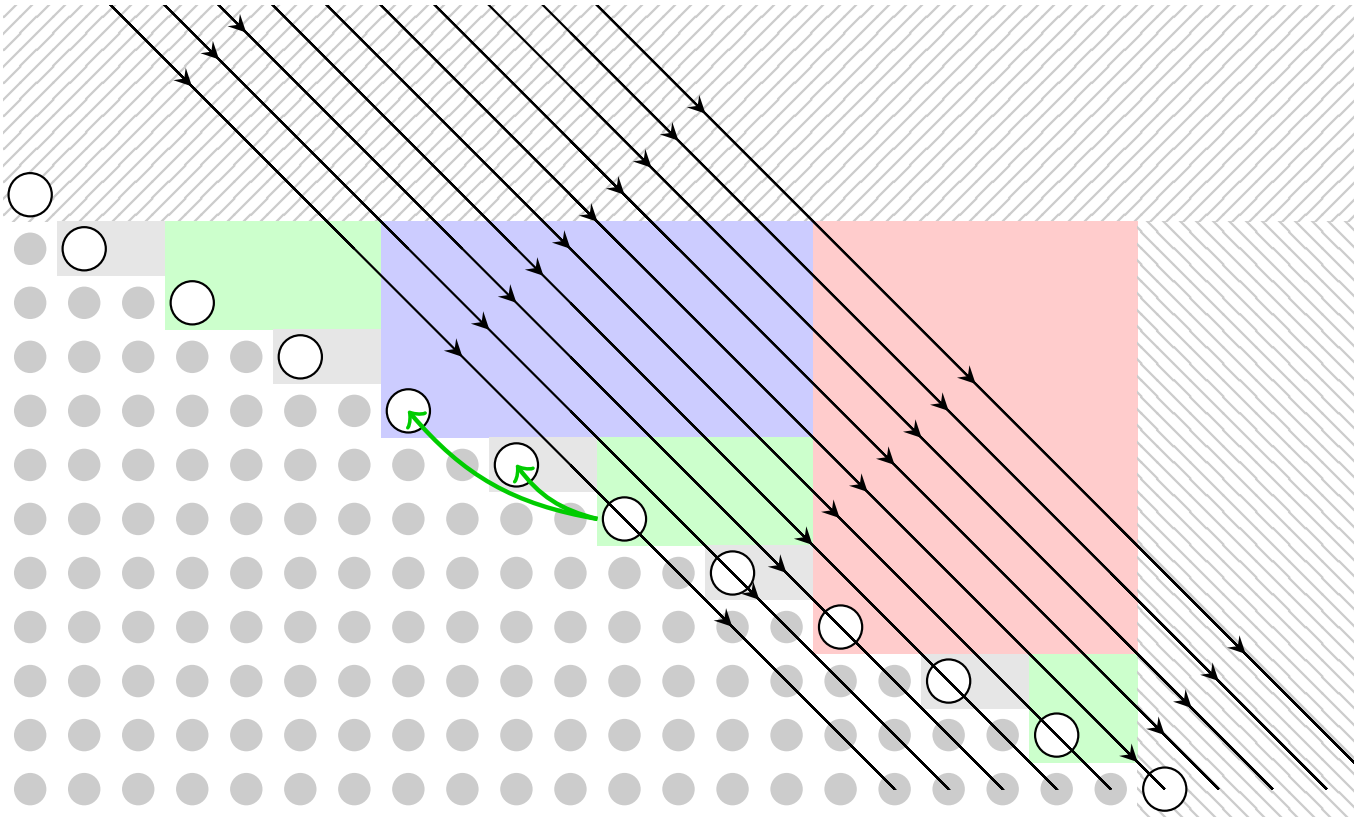
$$P = \bullet Y^{17} + \cdots$$

Reduction algorithm – Example



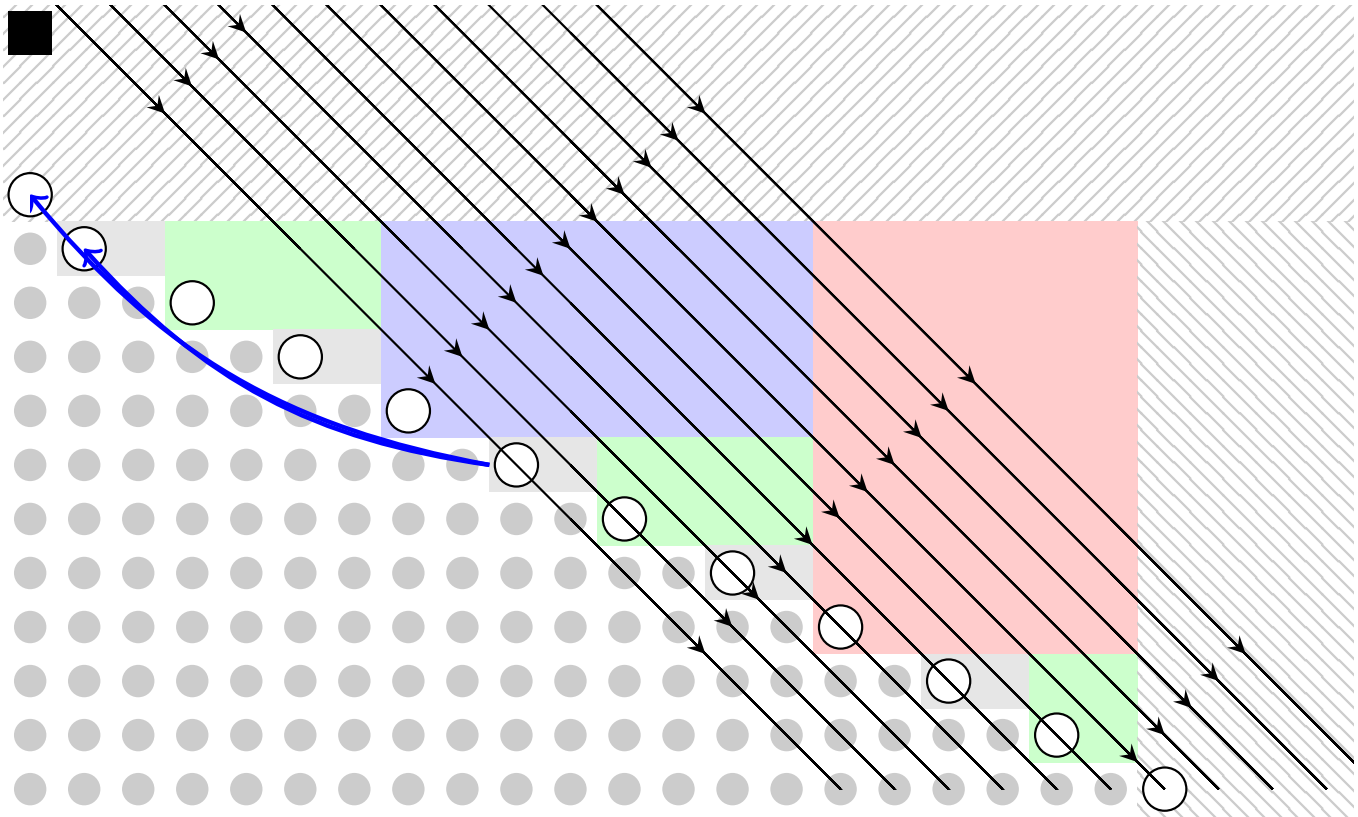
$$P = \bullet Y^{16} + \cdots$$

Reduction algorithm – Example



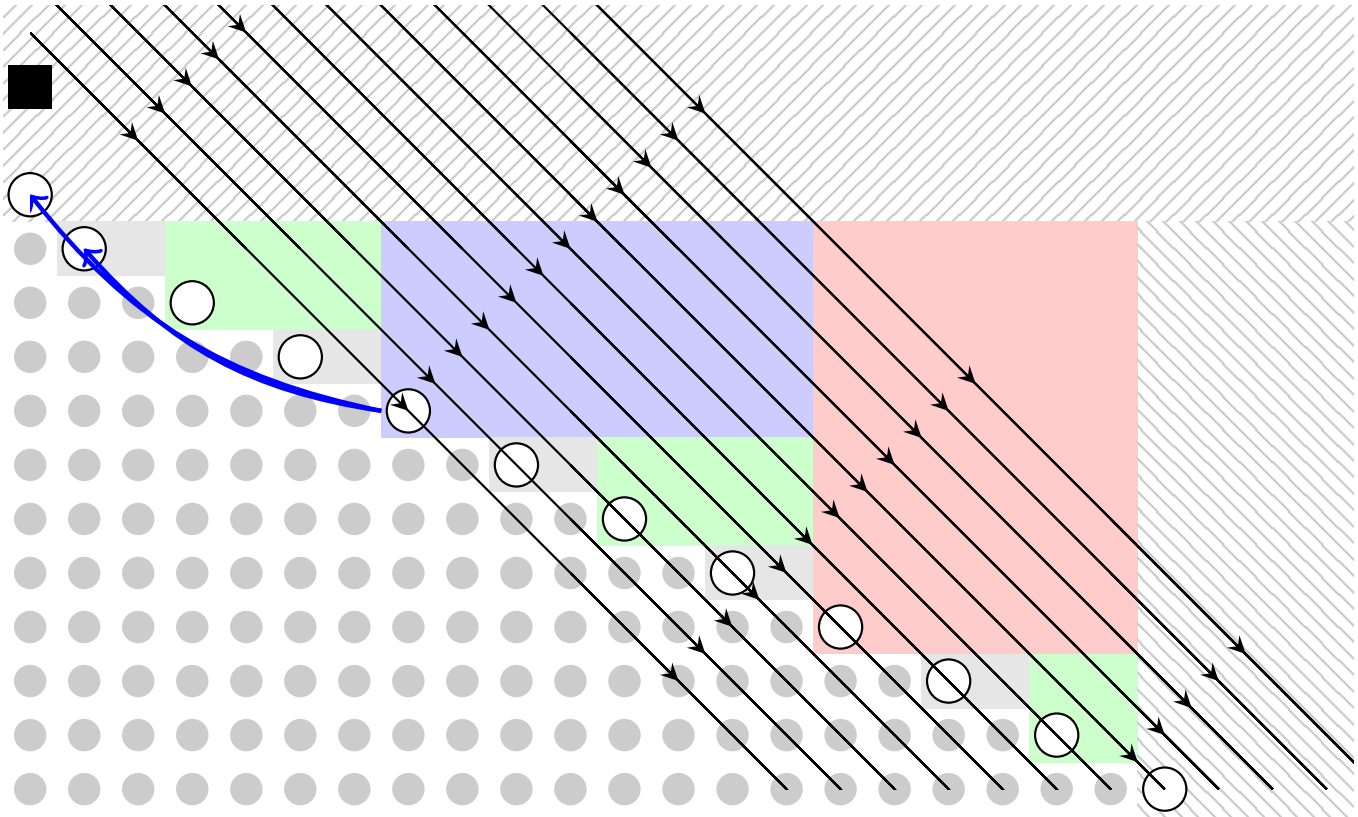
$$P = \bullet Y^{15} + \cdots$$

Reduction algorithm – Example



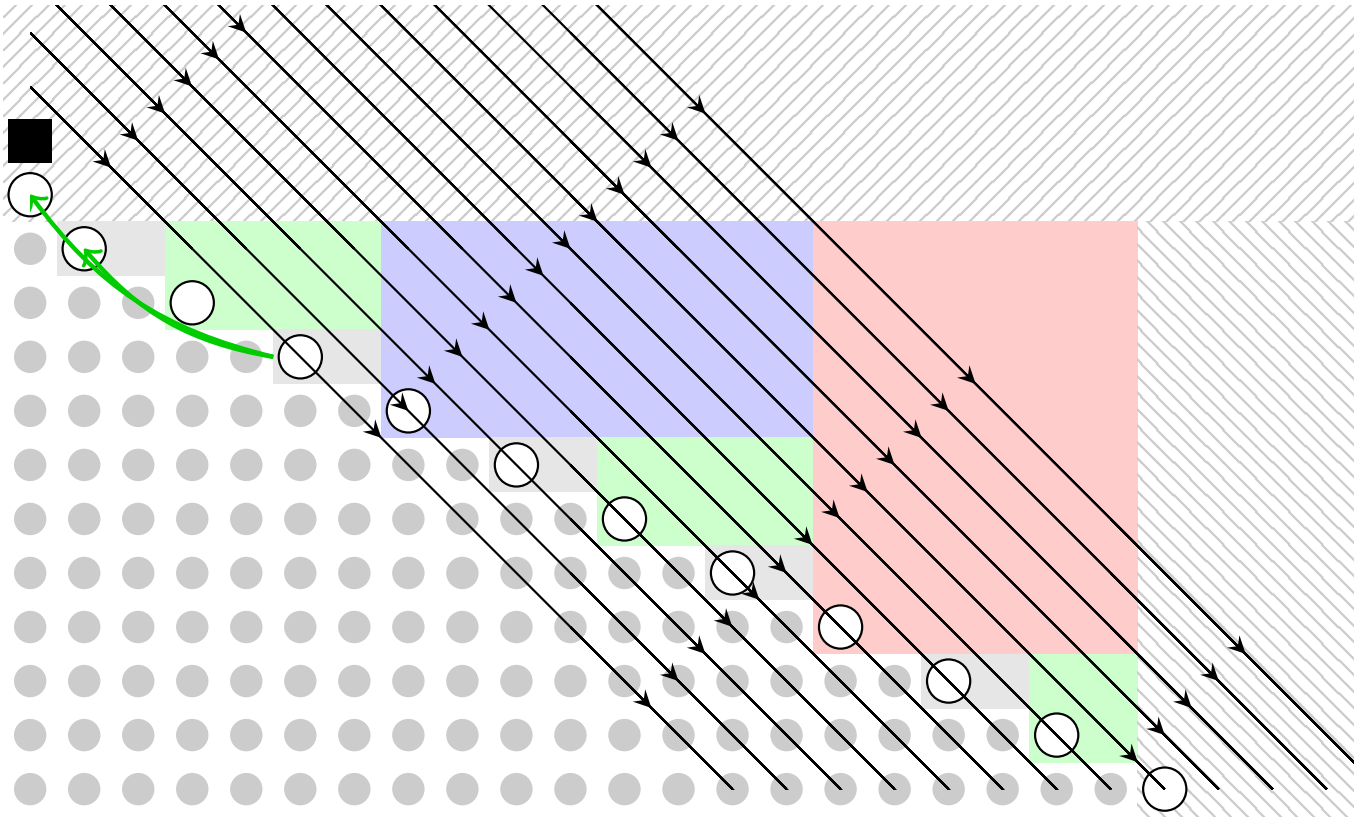
$$P = \bullet Y^{14} + \cdots$$

Reduction algorithm – Example



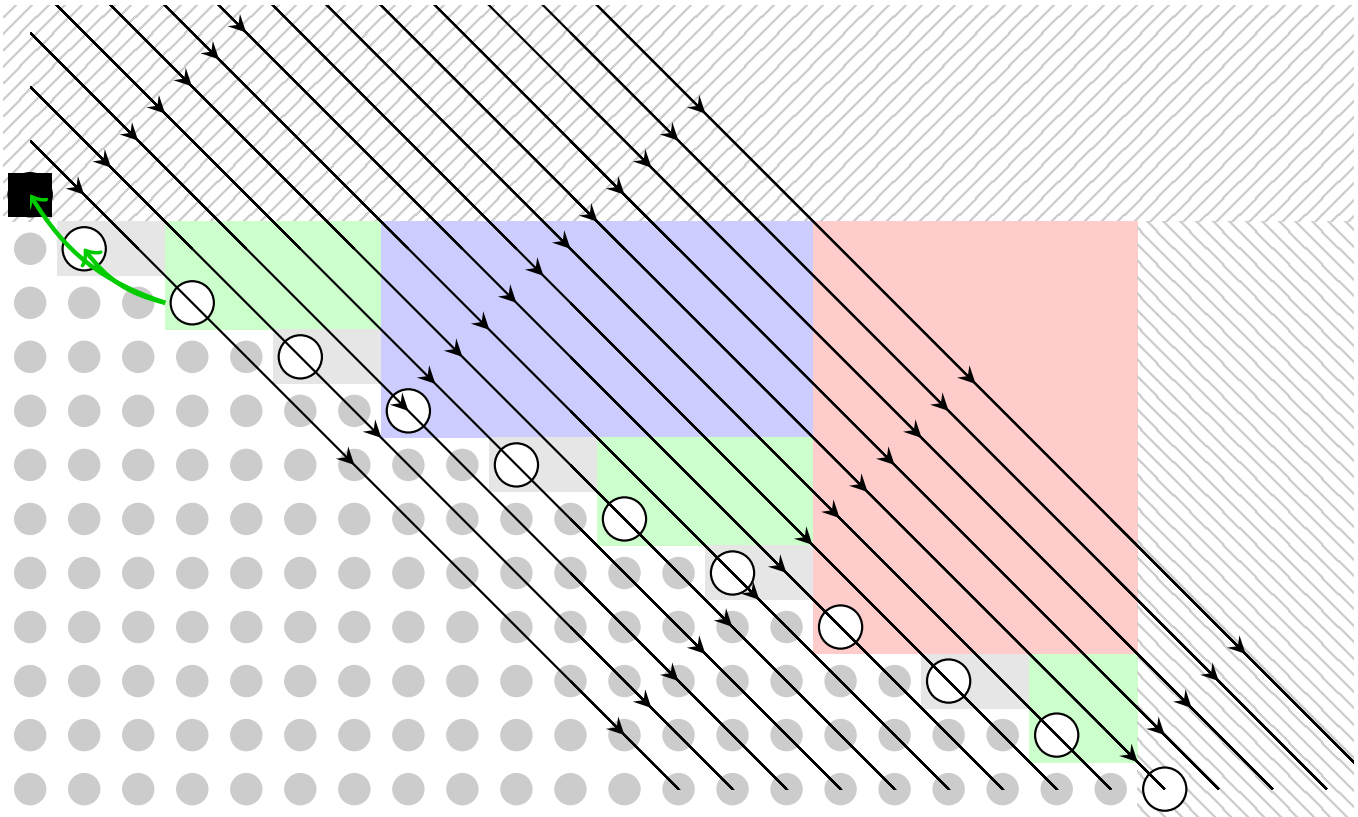
$$P = \bullet Y^{13} + \cdots$$

Reduction algorithm – Example



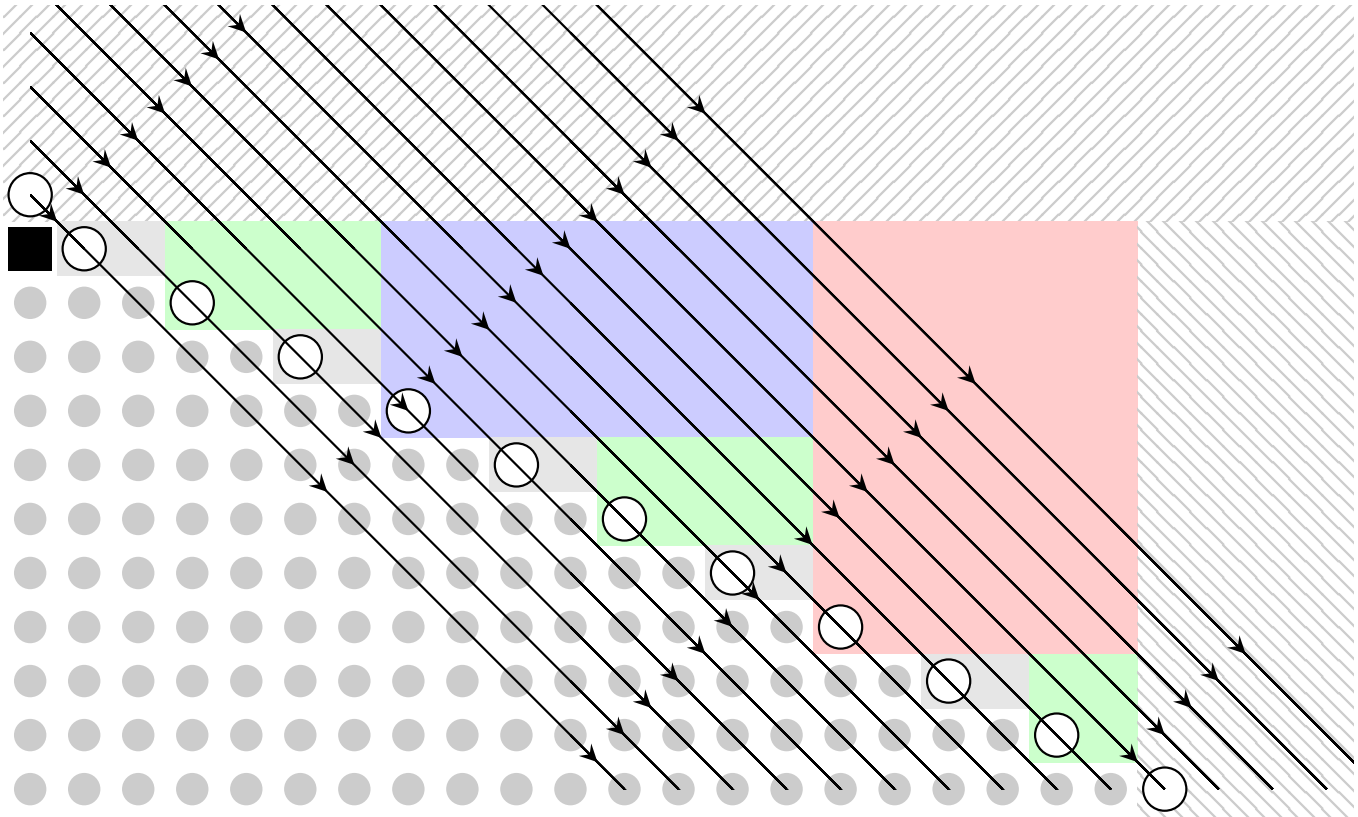
$$P = \bullet Y^{12} + \cdots$$

Reduction algorithm – Example



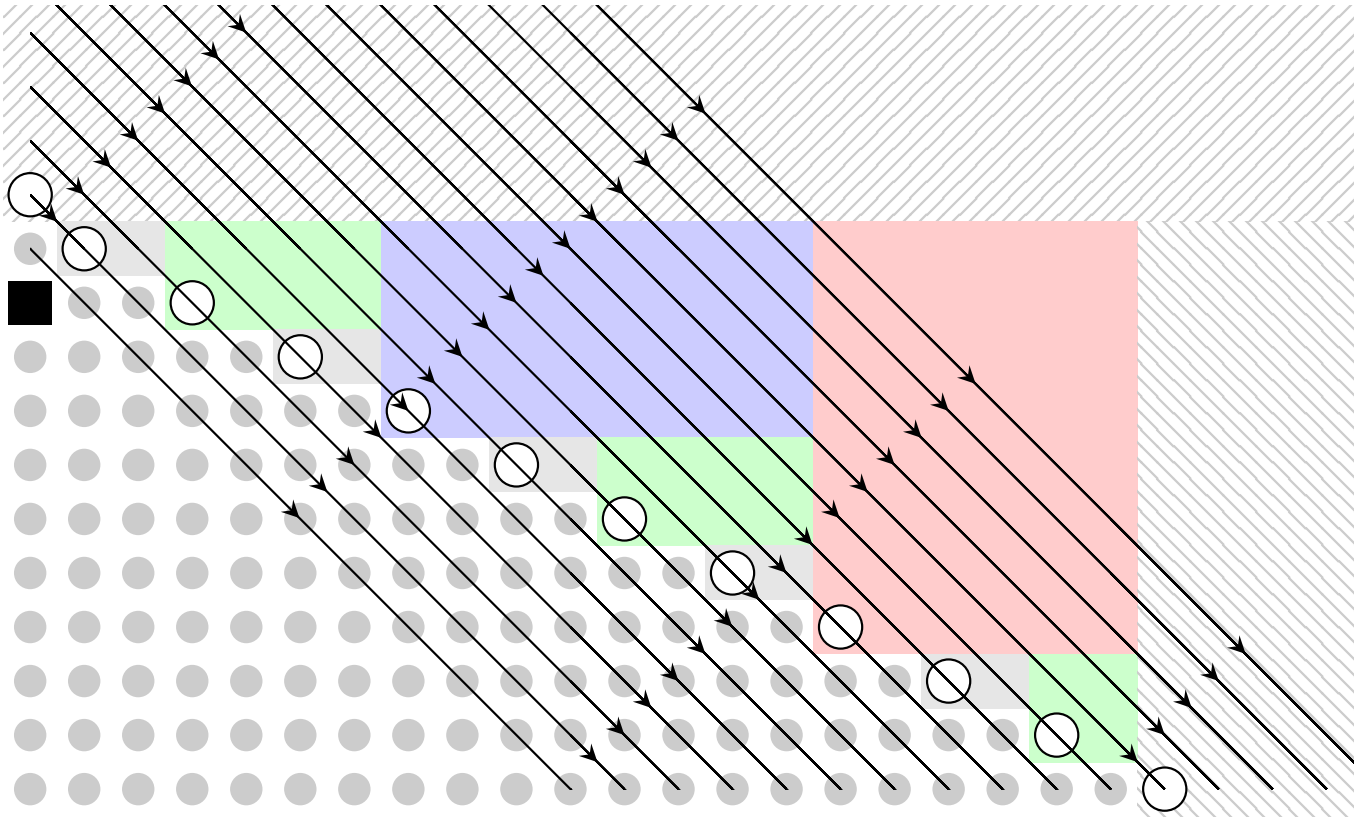
$$P = \bullet Y^{11} + \cdots$$

Reduction algorithm – Example



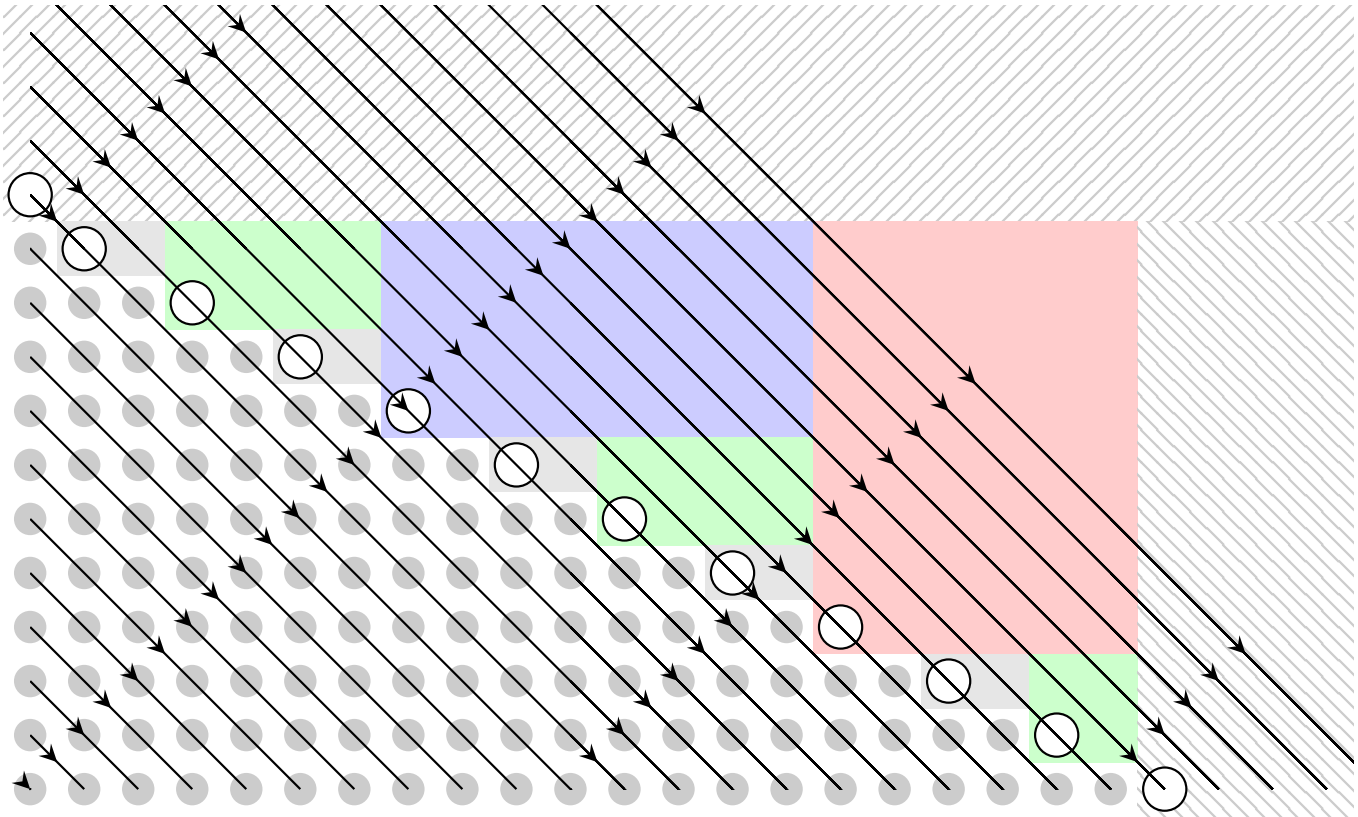
$$P = \bullet Y^{10} + \cdots$$

Reduction algorithm – Example



$$P = \bullet Y^9 + \cdots$$

Reduction algorithm – Example



$$P = 0$$

Implementation

- Proof-of-concept implementation in SageMath (`https://www.lix.polytechnique.fr/~larrieu/`)

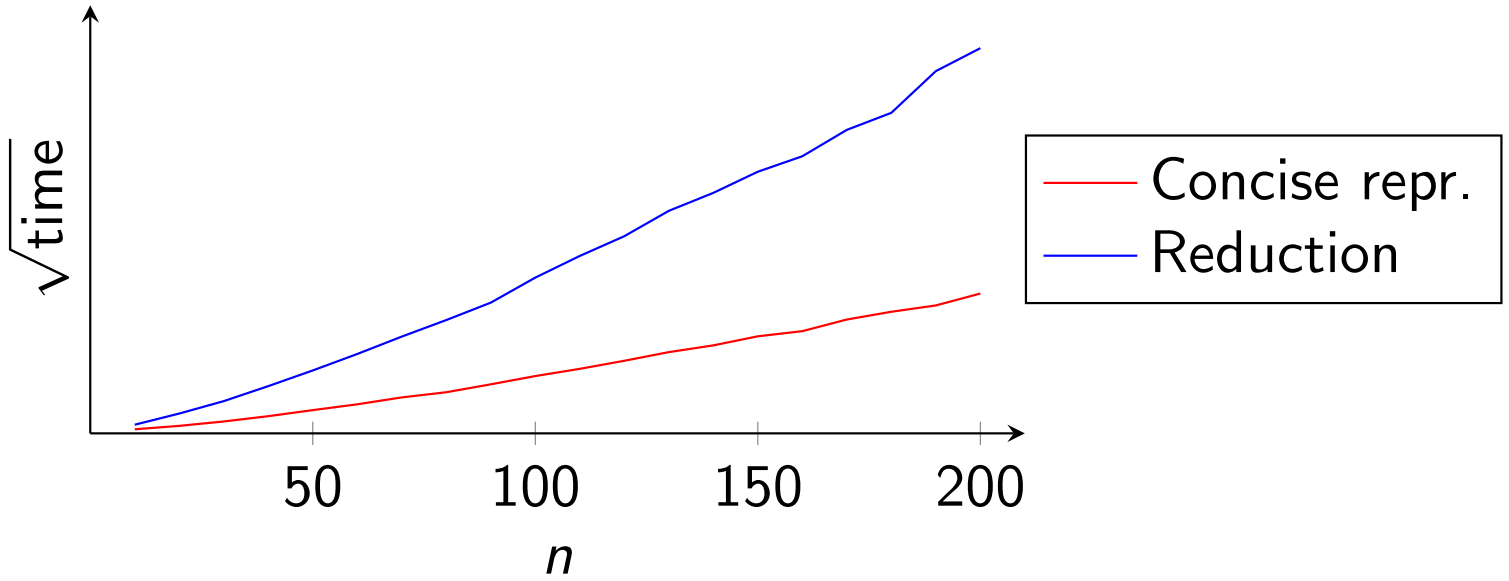
Implementation

- Proof-of-concept implementation in SageMath (`https://www.lix.polytechnique.fr/~larrieu/`)
- More serious implementation in Mathemagix (package larrix)

Implementation

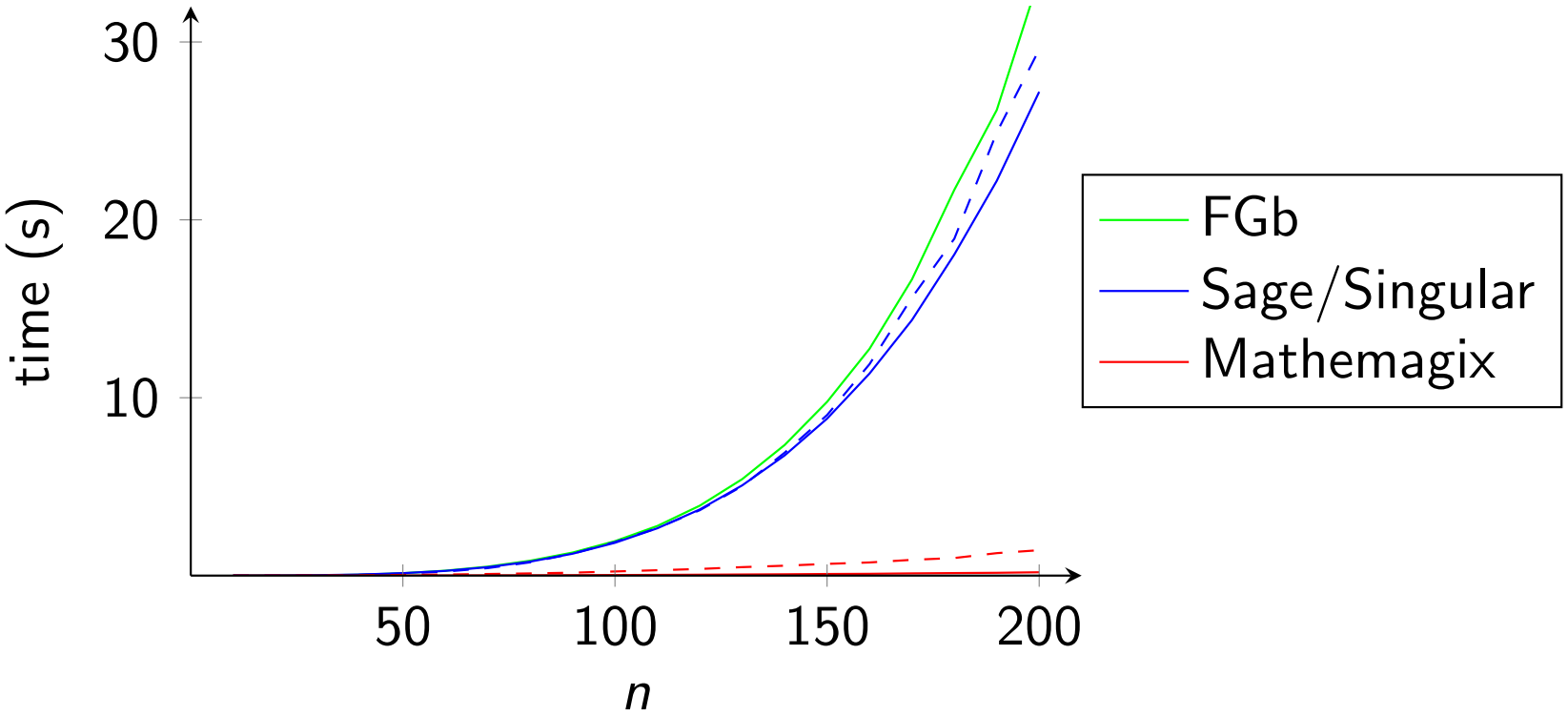
- Proof-of-concept implementation in SageMath (`https://www.lix.polytechnique.fr/~larrieu/`)
- More serious implementation in Mathemagix (package larrix)

Actually achieves $\tilde{O}(n^2)$ complexity.



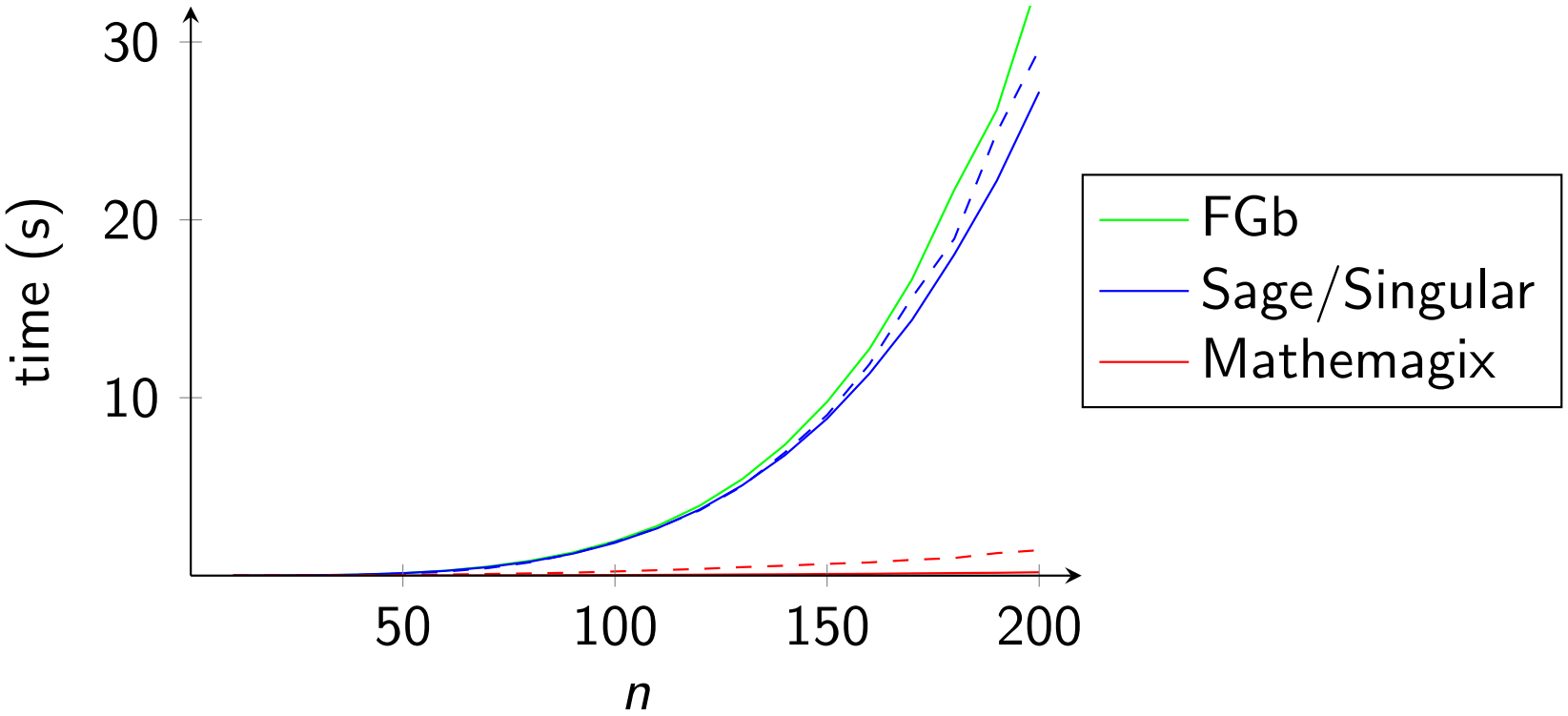
Benchmarks

Outperforms state-of-the-art libraries



Benchmarks

Outperforms state-of-the-art libraries



Note: those libraries support > 2 variables and non-generic settings while we do not.

Applications

Concise representation

Structure of $\mathbb{K}[X, Y]/\langle A, B\rangle$ in time $\tilde{O}(n^2)$.

Reduction

- Quasi-optimal ideal membership test $P \in^? \langle A, B\rangle$.
- Quasi-optimal multiplication in $\mathbb{K}[X, Y]/\langle A, B\rangle$.
- Compute the reduced Gröbner basis in time $\tilde{O}(n^3)$.
  (In general, the concise representation is sufficient)

Applications

Concise representation

Structure of $\mathbb{K}[X, Y]/\langle A, B\rangle$ in time $\tilde{O}(n^2)$.

Reduction

- Quasi-optimal ideal membership test $P \in^? \langle A, B\rangle$.
- Quasi-optimal multiplication in $\mathbb{K}[X, Y]/\langle A, B\rangle$.
- Compute the reduced Gröbner basis in time $\tilde{O}(n^3)$.
  (In general, the concise representation is sufficient)

Other consequence

The resultant of A and B can be computed in $O(n^{2+\varepsilon})$ over $\mathbb{F}_q$.
See *Fast computation of generic bivariate resultants* (van der Hoeven and Lecerf)

Conclusion

Vanilla Gröbner bases

Assuming sufficient genericity:

- Can precompute a *terse representation* (space $\tilde{O}(n^2)$).
- Can reduce in time $\tilde{O}(n^2)$ using this representation.

Grevlex order, generators given in total degree

Assuming sufficient genericity:

- Can compute efficiently a *concise representation* (space $\tilde{O}(n^2)$).
- Can reduce in time $\tilde{O}(n^2)$ using this representation.

Conclusion

Vanilla Gröbner bases

Assuming sufficient genericity:

- Can precompute a *terse representation* (space $\tilde{O}(n^2)$).
- Can reduce in time $\tilde{O}(n^2)$ using this representation.

Grevlex order, generators given in total degree

Assuming sufficient genericity:

- Can compute efficiently a *concise representation* (space $\tilde{O}(n^2)$).
- Can reduce in time $\tilde{O}(n^2)$ using this representation.

Generalization:

- Relax the genericity assumptions ?
- More than 2 variables ?

Conclusion

Vanilla Gröbner bases

Assuming sufficient genericity:

- Can precompute a *terse representation* (space $\tilde{O}(n^2)$).
- Can reduce in time $\tilde{O}(n^2)$ using this representation.

Grevlex order, generators given in total degree

Assuming sufficient genericity:

- Can compute efficiently a *concise representation* (space $\tilde{O}(n^2)$).
- Can reduce in time $\tilde{O}(n^2)$ using this representation.

Generalization:

- Relax the genericity assumptions ? $\rightarrow$ seems feasible.
- More than 2 variables ? $\rightarrow$ not so clear.

Conclusion

Vanilla Gröbner bases

Assuming sufficient genericity:

- Can precompute a *terse representation* (space $\tilde{O}(n^2)$).
- Can reduce in time $\tilde{O}(n^2)$ using this representation.

Grevlex order, generators given in total degree

Assuming sufficient genericity:

- Can compute efficiently a *concise representation* (space $\tilde{O}(n^2)$).
- Can reduce in time $\tilde{O}(n^2)$ using this representation.

Generalization:

- Relax the genericity assumptions ? $\rightarrow$ seems feasible.
- More than 2 variables ? $\rightarrow$ not so clear.

Thank you for your attention