Fast algorithms for factoring polynomials
A selection

Erich Kaltofen 寒爐
North Carolina State University
`google->kaltofen`
`google->han lu`

Overview of my work

Theorem. Factorization in $\mathsf{K}[x]$ is undecidable even if K is an effective field [van der Waerden '35, Fröhlich and Shepherdson '55].

Theorem. Factorization in $\mathbb{Z}_p[x]$ [Berlekamp '67] and $\mathbb{Q}[x]$ [LLL] is polynomial-time.

If factorization in $\mathsf{K}[x]$ is polynomial-time then factorization in $\mathsf{K}[x_1, \ldots, x_n]$ is polynomial-time [Kaltofen '82].

If arithmetic in K is polynomial-time then factorization in $\overline{\mathsf{K}}[x_1, \ldots, x_n]$ is polynomial-time [Kaltofen '85, '91].

Overview of my work

Theorem. Factorization in $\mathsf{K}[x]$ is undecidable even if K is an effective field [van der Waerden '35, Fröhlich and Shepherdson '55].

Theorem. Factorization in $\mathbb{Z}_p[x]$ [Berlekamp '67] and $\mathbb{Q}[x]$ [LLL] is polynomial-time.

If factorization in $\mathsf{K}[x]$ is polynomial-time then factorization in $\mathsf{K}[x_1, \ldots, x_n]$ is polynomial-time [Kaltofen '82].

If arithmetic in K is polynomial-time then factorization in $\overline{\mathsf{K}}[x_1, \ldots, x_n]$ is polynomial-time [Kaltofen '85, '91].

Best arithm. complexity: Lecerf '06 d^3 with LinBox linear algebra

Division-free straight-line program example

$v_1 \leftarrow c_1 \times x_1;$

$v_2 \leftarrow y - c_2;$ Comment: c_1, c_2 are constants in K

$v_3 \leftarrow v_2 \times v_2;$

$v_4 \leftarrow v_3 + v_1;$

$v_5 \leftarrow v_4 \times x_3;$

$\vdots$

$v_{101} \leftarrow v_{100} + v_{51};$

The variable v_{101} holds a polynomial in $\mathsf{K}[x_1, x_2, \ldots]$

Straight-line programs [Kaltofen '85] and black box programs [Kaltofen & Trager '88] for irreducible factors can be computed in random polynomial time in the input size and total degree.

Division-free straight-line program example

$$
\begin{aligned}
&\upsilon_1 \leftarrow c_1 \times x_1;\\
&\upsilon_2 \leftarrow y - c_2; \qquad \text{Comment: } c_1, c_2 \text{ are constants in } \mathsf{K}\\
&\upsilon_3 \leftarrow \upsilon_2 \times \upsilon_2;\\
&\upsilon_4 \leftarrow \upsilon_3 + \upsilon_1;\\
&\upsilon_5 \leftarrow \upsilon_4 \times x_3;\\
&\vdots\\
&\upsilon_{101} \leftarrow \upsilon_{100} + \upsilon_{51};
\end{aligned}
$$

The variable υ_{101} holds a polynomial in $\mathsf{K}[x_1, x_2, \ldots]$

Straight-line programs [Kaltofen '85] and black box programs [Kaltofen & Trager '88] for irreducible factors can be computed in random polynomial time in the input size and total degree.

$\longrightarrow$ used by V. Kabernets [2003] for complexity lower bounds.

Subquadratic complexity

Theorem. We have two algorithms that factor in $\mathbb{Z}_2[x]$ in $O(n^{1.81})$ bit complexity [Kaltofen & Shoup '95].

Subquadratic complexity

Theorem. We have two algorithms that factor in $\mathbb{Z}_2[x]$ in $O(n^{1.81})$ bit complexity [Kaltofen & Shoup '95].

Unfortunately, remains best-known complexity today
Note: no complexity model tricks (output size, field operation count, etc.) possible

Approximate multivariate factorization

Conclusion on my exact algorithm [JSC 1(1)'85]

"*D. Izraelevitz at Massachusetts Institute of Technology has already implemented a version of algorithm 1 using complex floating point arithmetic. Early experiments indicate that the linear systems computed in step (L) tend to be* ***numerically ill-conditioned.*** *How to overcome this numerical problem is an important question which we will investigate.*"

Approximate multivariate factorization

Conclusion on my exact algorithm [JSC 1(1)'85]

"*D. Izraelevitz at Massachusetts Institute of Technology has already implemented a version of algorithm 1 using complex floating point arithmetic. Early experiments indicate that the linear systems computed in step (L) tend to be* ***numerically ill-conditioned.*** *How to overcome this numerical problem is an important question which we will investigate.*"

Gao, Kaltofen, May, Yang, Zhi 2004: practical algorithms to find the factorization of a nearby factorizable polynomial given any f

especially "noisy" f:
Given $\underline{f} = \underline{f_1 \cdots f_s} + f_{\text{noise}}$,
we find $\bar{f}_1, \ldots \bar{f}_s$ s.t. $\| f_1 \cdots f_s - \bar{f}_1 \cdots \bar{f}_s \| \approx \| f_{\text{noise}} \|$
even for large noise: $\| f_{\text{noise}} \| / \| f \| \geq 10^{-3}$

Kaltofen & Koiran '06: supersparse (lacunary) polynomials

$f = \sum_i c_i \overline{X}^{\overline{\alpha_i}} \in \mathsf{K}[\overline{X}]$ where $\overline{X}^{\overline{\alpha_i}} = X_1^{\alpha_{i,1}} \cdots X_n^{\alpha_{i,n}}$

Input: $\varphi(\zeta) \in \mathbb{Z}[\zeta]$ monic irred.; let $\mathsf{K} = \mathbb{Q}[\zeta]/(\varphi(\zeta))$
a supersparse $f(\overline{X}) = \sum_{i=1}^{t} c_i \overline{X}^{\overline{\alpha_i}} \in \mathsf{K}[\overline{X}]$
a factor degree bound d

Output: a list of all irreducible factors of f over K of degree $\leq d$
and their multiplicities (which is $\leq t$ except for any X_j)

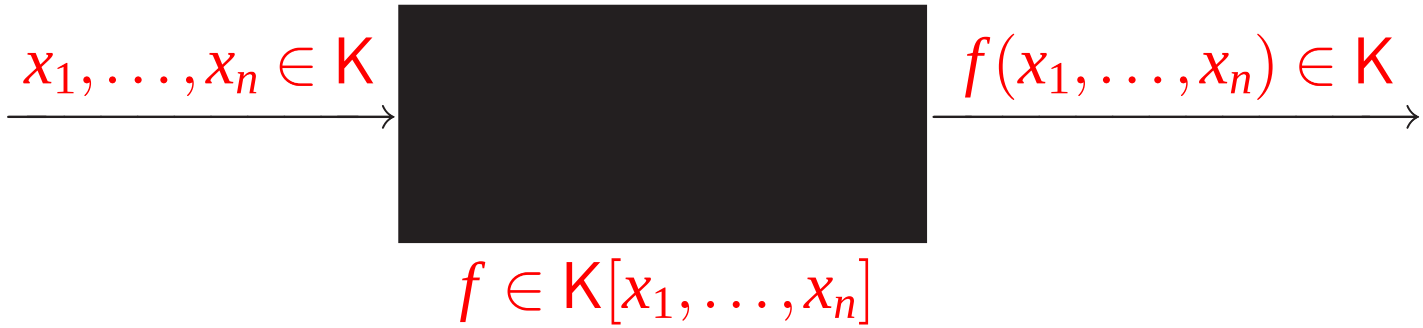
Bit complexity is:

$\big(\mathrm{size}(f) + d + \deg(\varphi) + \log\|\varphi\|\big)^{O(n)}$ (sparse factors)

$\big(\mathrm{size}(f) + d + \deg(\varphi) + \log\|\varphi\|\big)^{O(1)}$ (blackbox factors)

where $\mathrm{size}(f) = \sum_{i=1}^{t} (\text{dense-size}(c_i) + \lceil \log_2(\alpha_{i,1} \cdots \alpha_{i,n} + 2) \rceil)$

Black box polynomials

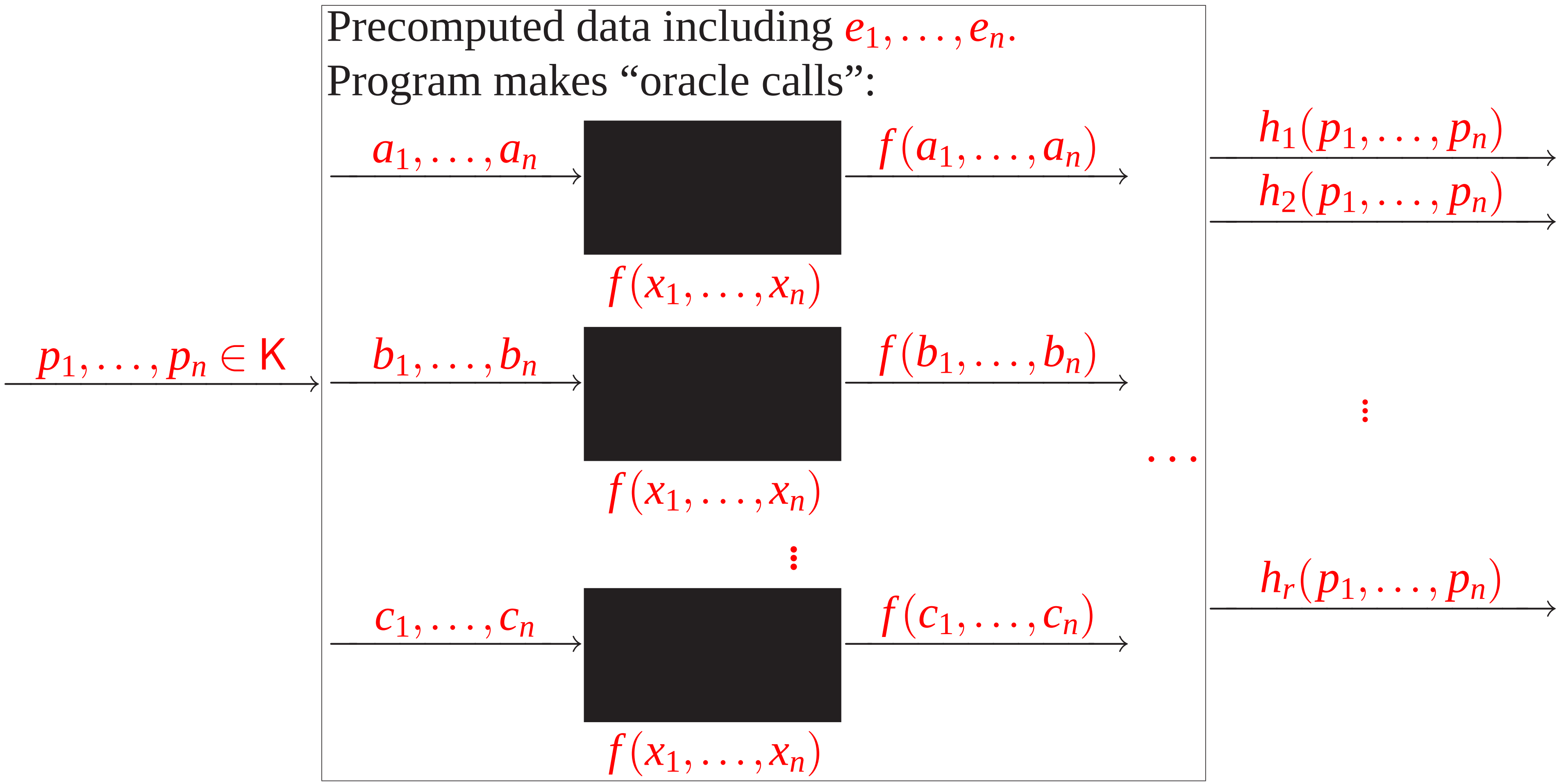


K an arbitrary field, e.g., rationals, reals, complexes

Perform polynmial algebra operations, e.g., factorization with

$n^{O(1)}$ black box calls,
$n^{O(1)}$ arithmetic operations in K and
$n^{O(1)}$ randomly selected elements in K

Kaltofen and Trager (1988) efficiently construct the following efficient program:



$$f(x_1, \ldots, x_n) = h_1(x_1, \ldots, x_n)^{e_1} \cdots h_r(x_1, \ldots, x_n)^{e_r}$$

$h_i \in \mathsf{K}[x_1, \ldots, x_n]$ irreducible.

Characterization of Factor Evaluation Program

- Always evaluates the same associate of each factor

$$x\,y \quad \text{vs.} \quad (\frac{1}{2}x)\,(2y)$$

- Construction of program is Monte-Carlo (might produce incorrect program with probability $\le \varepsilon$), and requires a factorization procedure for $\mathsf{K}[y]$, but the program itself is deterministic
- Program contains positive integer constants of value bounded by $2^{\deg(f)^{1+o(1)}}/\varepsilon$
- Program makes

$$O(\deg(f)^2) \text{ oracle calls,}$$

  none of whose inputs depends on another one's output, $\longrightarrow$ parallel version
- Furthermore, program performs $\deg(f)^{2+o(1)}$ arithmetic operations in K

Given a black box

$f(x_1, \ldots, x_n) \in \mathsf{K}[x_1, \ldots, x_n]$
K a field

compute by multiple evaluation of this black box the sparse representation of f

$$f(x_1, \ldots, x_n) = \sum_{i=1}^{t} a_i x_1^{e_{i,1}} \cdots x_n^{e_{i,n}}, \quad a_i \neq 0$$

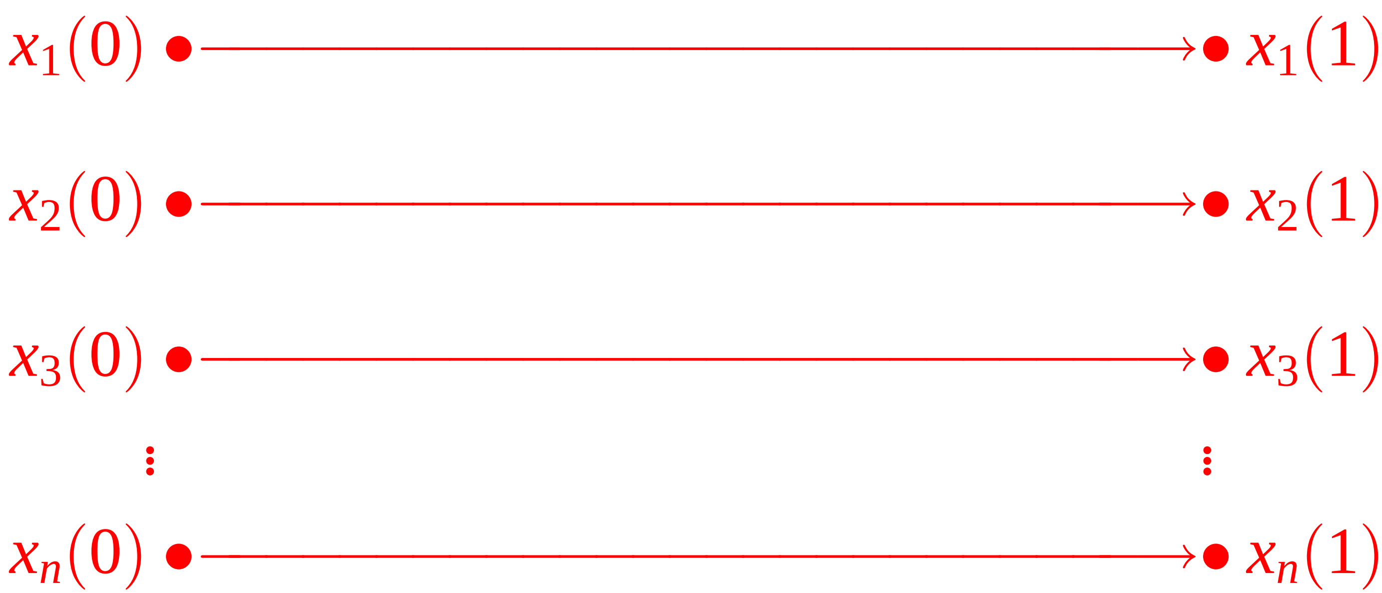
Several solutions that are polynomial-time in n and t:

Zippel (1979, 1988), Ben-Or, Tiwari (1988)
Kaltofen, Lakshman (1988)
Grigoriev, Karpinski, Singer (1988)
Mansour (1992)
Kaltofen and Lee (2000)

Homotopy Method for Solving $F(X) = 0$

Known:
Solution to
$G(X) = 0$

Wanted:
Solution to
$F(X) = 0$



Follow from $y = 0$ to $y = 1$ the solutions of

$$H(X(y)) = (1-y)G(X(y)) + yF(X(y))$$

Our Homotopy

For $f(x_1,\dots,x_n) \in \mathsf{K}[x_1,\dots,x_n]$ consider

$$\bar{f}(X,Y) = f(X+b_1, Y(p_2-a_2(p_1-b_1)-b_2)+a_2X+b_2,\\ \dots, Y(p_n-a_n(p_1-b_1)-b_n)+a_nX+b_n)$$

The field elements $a_2,\dots,a_n,b_1,\dots,b_n$ are pre-chosen (“known”)
The field elements $p_1,\dots,p_n$ are input

Notice: The polynomial $\bar{f}(X,0)$ is independent of $p_1,\dots,p_n$ and can be factored into

$$\bar{f}(X,0) = \prod_{i=1}^{r} g_i(X)^{e_i}, \quad g_i(X) \in \mathsf{K}[X] \text{ irreducible}$$

Our Homotopy

For $f(x_1,\ldots,x_n) \in \mathsf{K}[x_1,\ldots,x_n]$ consider

$$\bar{f}(X,Y) = f(X+b_1, Y(p_2-a_2(p_1-b_1)-b_2)+a_2X+b_2, \ldots, Y(p_n-a_n(p_1-b_1)-b_n)+a_nX+b_n)$$

The field elements $a_2,\ldots,a_n,b_1,\ldots,b_n$ are pre-chosen ("known")
The field elements $p_1,\ldots,p_n$ are input

Notice: The polynomial $\bar{f}(X,0)$ is independent of $p_1,\ldots,p_n$ and can be factored into

$$\bar{f}(X,0) = \prod_{i=1}^{r} g_i(X)^{e_i}, \quad g_i(X) \in \mathsf{K}[X] \text{ irreducible}$$

By an ***effective Hilbert Irreducibility Theorem*** one can guarantee that the g_i are distinct images of the factors of f

$$g_i(X) = h_i(X+b_1,\ldots,a_nX+b_n),\ f(x_1,\ldots,x_n) = \prod_{i=1}^{r} h(x_1,\ldots,x_n)^{e_i}$$

$\longrightarrow$ enters randomization

Our Homotopy

For $f(x_1,\dots,x_n) \in \mathsf{K}[x_1,\dots,x_n]$ consider

$$\bar{f}(X,Y) = f(X+b_1, Y(p_2 - a_2(p_1-b_1)-b_2)+a_2X+b_2,\\ \dots, Y(p_n - a_n(p_1-b_1)-b_n)+a_nX+b_n)$$

The field elements $a_2,\dots,a_n,b_1,\dots,b_n$ are pre-chosen (“known”)
The field elements $p_1,\dots,p_n$ are input

Notice: The polynomial $\bar{f}(X,0)$ is independent of $p_1,\dots,p_n$ and can be factored into

$$\bar{f}(X,0) = \prod_{i=1}^{r} g_i(X)^{e_i}, \quad g_i(X) \in \mathsf{K}[X] \text{ irreducible}$$

By ***Hensel Lifting*** we can follow the factorization to

$$\bar{f}(X,Y) = \prod_{i=1}^{r} \bar{h}_i(X,Y)^{e_i}$$

Now

$$\bar{f}(p_1-b_1,1) = f(p_1,\dots,p_n), \quad \forall i: \bar{h}_i(p_1-b_1,1) = h_i(p_1,\dots,p_n)$$

Four Corollaries

Corollary 1: (Parallel Factorization)
For $\mathsf{K} = \mathbb{Q}$, we can compute in Monte Carlo $\mathcal{NC}$ all sparse factors of f of fixed degree and with no more than a given number t terms

Corollary 2: (Sparse Rational Interpolation)
Given a degree bound

$$b \geq \max(\deg(f), \deg(g))$$

and a bound t for the maximum number of non-zero terms in both f and g, we can in **Las Vegas** polynomial-time in b and t compute from a black box for f/g the sparse representations of f and g

Four Corollaries

Corollary 1: (Parallel Factorization)
For $\mathsf{K} = \mathbb{Q}$, we can compute in Monte Carlo $\mathscr{NC}$ all sparse factors of f of fixed degree and with no more than a given number t terms

Corollary 2' [Kaltofen & Yang '07]: (Sparse Rational Interpol.)
Given a degree bound

$$b \geq \max(\deg(f), \deg(g))$$

we can in **Monte Carlo** polynomial-time in b and t_f, t_g (number of terms in f and g) compute the sparse representations of f, g.

Four Corollaries

Corollary 1: (Parallel Factorization)
For $\mathsf{K} = \mathbb{Q}$, we can compute in Monte Carlo $\mathcal{NC}$ all sparse factors of f of fixed degree and with no more than a given number t terms

Corollary 2′ [Kaltofen & Yang '07]: (Sparse Rational Interpol.)
Given a degree bound

$$b \geq \max(\deg(f), \deg(g))$$

we can in **Monte Carlo** polynomial-time in b and t_f, t_g (number of terms in f and g) compute the sparse representations of f, g.

Uses **early termination** [Kaltofen & Lee '03]; our algorithm is practical. **Hybrid** version based on [Giesbrecht, Labahn, Lee '06] and [Kaltofen, Yang, Zhi '05].

Corollary 3: (Greatest Common Divisor)
From a black box for

$$f_1(x_1,\ldots,x_n),\ldots,f_r(x_1,\ldots,x_r) \in \mathsf{K}[x_1,\ldots,x_n]$$

we can efficiently produce a feasible program with oracle calls that allows to evaluate one and the same associate of

$$\mathrm{GCD}(f_1,\ldots,f_r).$$

Corollary 4: (Factors as Straight-Line Programs)
Let $f \in \mathsf{K}[x_1, \ldots, x_n]$ be given by a straight-line program of size s, e.g.,

$$\upsilon_1 \leftarrow c_1 \times x_1;$$
$$\upsilon_2 \leftarrow x_2 - c_2;$$

Comment: c_1, c_2 are constants in K

$$\upsilon_3 \leftarrow \upsilon_2 \times \upsilon_2;$$
$$\upsilon_4 \leftarrow \upsilon_3 + \upsilon_1;$$
$$\upsilon_5 \leftarrow \upsilon_4 \times x_3;$$
$$\vdots$$
$$\upsilon_{101} \leftarrow \upsilon_{100} + \upsilon_{51};$$

The variable υ_{101} holds a polynomial in $\mathbb{F}_q[x_1, \ldots]$ of degree $\leq 2^{101}$. Then one can compute in polynomial-time in $s + \deg(f)$ straight-line programs of **polynomial-size** for all irreducible factors.

谢

谢

THANK YOU!