Karatsuba's algorithm for polynomials over rings without decidable equality

Max Zeuner

University of Stockholm

Workshop on Homotopy Type Theory/ Univalent Foundations
Vienna, 23.04.2023

A brief history of fast multiplication

- **1960:** In a seminar, Kolmogorov conjectures:

  The standard elementary school, $\mathcal{O}(n^2)$-fast multiplication algorithm is optimal
  (n being the number of bits of the numbers multiplied)

A brief history of fast multiplication

- **1960:** In a seminar, Kolmogorov conjectures:

  The standard elementary school, $\mathcal{O}(n^2)$-fast multiplication algorithm is optimal
  (n being the number of bits of the numbers multiplied)

- **One week later:** Anatoly Karatsuba (23 at the time) finds a faster algorithm

A brief history of fast multiplication

- **1960:** In a seminar, Kolmogorov conjectures:

  The standard elementary school, $\mathcal{O}(n^2)$-fast multiplication algorithm is optimal
  (n being the number of bits of the numbers multiplied)

- **One week later:** Anatoly Karatsuba (23 at the time) finds a faster algorithm

- **1962:** Kolmogorov publishes the result under Karatsuba's name: Karatsuba and Ofman [1962]

Karatsuba for polynomials I

Following [Abdeljaoued and Lombardi, 2004, Sec. 6.1]:

$$\begin{aligned} p(x) &= a_0 + a_1 x + \cdots + a_n x^n \\ &= \underbrace{p_e}_{a_0 + a_2 x + \ldots}(x^2) + x \cdot \underbrace{p_o}_{a_1 + a_3 x + \ldots}(x^2) \end{aligned}$$

$\Rightarrow$ we can write $p(x) \cdot q(x)$ as:

$$p_e q_e(x^2) + x \cdot \big[(p_e + p_o)(q_e + q_o) - p_e q_e - p_o q_o\big](x^2) + x^2 \cdot p_o q_o(x^2)$$

Karatsuba for polynomials II

Algorithm: `karatsuba`

Input: polynomials p, q

if *p or q constant* **then**

  scalar multiplication with constant

  `-- remark:` $p = a_0 \Rightarrow p = p_e$

else

  $r \leftarrow$ `karatsuba` $p_e\ q_e$

  $s \leftarrow$ `karatsuba` $p_o\ q_o$

  $t \leftarrow$ `karatsuba` $(p_e + p_o)\ (q_e + q_o)$

  return $r(x^2) + x \cdot \left[t - r - s\right](x^2) + x^2 \cdot s(x^2)$

end

Karatsuba for polynomials II

Algorithm: `karatsuba`

Input: polynomials p, q

if *p or q constant* **then**

  scalar multiplication with constant

  `-- remark:` $p = a_0 \Rightarrow p = p_e$

else

  $r \leftarrow$ `karatsuba` $p_e\ q_e$

  $s \leftarrow$ `karatsuba` $p_o\ q_o$

  $t \leftarrow$ `karatsuba` $(p_e + p_o)\ (q_e + q_o)$

  return $r(x^2) + x \cdot \left[t - r - s\right](x^2) + x^2 \cdot s(x^2)$

end

$\Rightarrow$ 3 recursive calls with input roughly half the size

$\Rightarrow$ runs in $\mathcal{O}(n^{\log_2 3}) \approx \mathcal{O}(n^{1.585})$ instead of $\mathcal{O}(n^{\log_2 4}) = \mathcal{O}(n^2)$ time

Problems w/ verification I

Termination-checker complains: p_e, p_o *not structurally smaller than* p

Problems w/ verification I

Termination-checker complains: p_e, p_o *not structurally smaller than* p

Solution (e.g. Dénès et al. [2012]):
provide **fuel**, additional argument of type $\mathbb{N}$:

- $\texttt{karatsubaRec}\ 0\ p\ q := p \cdot q$
- $\texttt{karatsubaRec}\ (n+1)\ p\ q :=$ above algorithm w/ recursive calls
  - $\texttt{karatsubaRec}\ n\ p_e\ q_e$
  - $\texttt{karatsubaRec}\ n\ p_o\ q_o$
  - $\texttt{karatsubaRec}\ n\ (p_e + p_o)\ (q_e + q_o)$

Problems w/ verification I

Termination-checker complains: *p_e, p_o not structurally smaller than p*

Solution (e.g. Dénès et al. [2012]):
provide **fuel**, additional argument of type $\mathbb{N}$:

- `karatsubaRec` $0\ p\ q := p \cdot q$
- `karatsubaRec` $(n+1)\ p\ q :=$ above algorithm w/ recursive calls
  - `karatsubaRec` $n\ p_e\ q_e$
  - `karatsubaRec` $n\ p_o\ q_o$
  - `karatsubaRec` $n\ (p_e + p_o)\ (q_e + q_o)$

$\Rightarrow$ `karatsuba` $p\ q$:= `karatsubaRec` $\max\{\ell(p), \ell(q)\}\ p\ q$.
(ℓ something like length/size/degree...)

Problems w/ verification I

Termination-checker complains: *p_e, p_o not structurally smaller than p*

Solution (e.g. Dénès et al. [2012]):
provide **fuel**, additional argument of type $\mathbb{N}$:

- `karatsubaRec` $0\ p\ q := p \cdot q$
- `karatsubaRec` $(n+1)\ p\ q :=$ above algorithm w/ recursive calls
  - `karatsubaRec` $n\ p_e\ q_e$
  - `karatsubaRec` $n\ p_o\ q_o$
  - `karatsubaRec` $n\ (p_e + p_o)\ (q_e + q_o)$

$\Rightarrow$ `karatsuba` $p\ q$:= `karatsubaRec` $\max\{\ell(p), \ell(q)\}\ p\ q$.
(ℓ something like length/size/degree...)

Today: stick with this solutions

Problems w/ verification II

Goal: Prove `karatsubaRec` n is multiplication for any n.
(don't prove that fuel $\max\{\ell(p), \ell(q)\}$ bigger than no. of recursions!)

Problems w/ verification II

Goal: Prove `karatsubaRec` n is multiplication for any n.
(don't prove that fuel $\max\{\ell(p), \ell(q)\}$ bigger than no. of recursions!)

- Base Case: for free
- Inductive Step: show algebraic identity

Problems w/ verification II

Goal: Prove `karatsubaRec` n is multiplication for any n.
(don't prove that fuel $\max\{\ell(p), \ell(q)\}$ bigger than no. of recursions!)

- Base Case: for free
- Inductive Step: show algebraic identity

Problem: List of coefficients not unique representation of polynomials (no ring structure on lists!!!)

Problems w/ verification II

Goal: Prove `karatsubaRec` n is multiplication for any n.
(don't prove that fuel $\max\{\ell(p), \ell(q)\}$ bigger than no. of recursions!)

- Base Case: for free
- Inductive Step: show algebraic identity

Problem: List of coefficients not unique representation of polynomials
(no ring structure on lists!!!)

Solution 1 (path of pain): bite the bullet and prove `karatsuba` for lists
(See bachelor's thesis of Sahlin [2022])

Problems w/ verification II

Goal: Prove `karatsubaRec` n is multiplication for any n.
(don't prove that fuel $\max\{\ell(p), \ell(q)\}$ bigger than no. of recursions!)

- Base Case: for free
- Inductive Step: show algebraic identity

Problem: List of coefficients not unique representation of polynomials (no ring structure on lists!!!)

Solution 1 (path of pain): bite the bullet and prove `karatsuba` for lists (See bachelor's thesis of Sahlin [2022])

Solution 2: assume decidable equality on base ring
$\Rightarrow$ represent $p(x)$ by list of coefficients $[a_0, \ldots, a_n]$ with $a_n \neq 0$

Problems w/ verification II

Goal: Prove `karatsubaRec` n is multiplication for any n.
(don't prove that fuel $\max\{\ell(p), \ell(q)\}$ bigger than no. of recursions!)

- Base Case: for free
- Inductive Step: show algebraic identity

Problem: List of coefficients not unique representation of polynomials
(no ring structure on lists!!!)

Solution 1 (path of pain): bite the bullet and prove `karatsuba` for lists
(See bachelor's thesis of Sahlin [2022])

Solution 2: assume decidable equality on base ring
$\Rightarrow$ represent $p(x)$ by list of coefficients $[a_0, \ldots, a_n]$ with $a_n \neq 0$

Today: Use HITs to get best of both worlds! (see this `PR`)

A HIT for Polynomials

Idea: Polynomials as lists of coefficients modulo trailing zeros:

A HIT for Polynomials

Idea: Polynomials as lists of coefficients modulo trailing zeros:

```
data R[X] : Type where
  []     : R[X]
  _::_   : R → R[X] → R[X]
  drop0 : 0r :: [] ≡ []
```

A HIT for Polynomials

Idea: Polynomials as lists of coefficients modulo trailing zeros:

```
data R[X] : Type where
  []    : R[X]
  _::_  : R → R[X] → R[X]
  drop0 : 0r :: [] ≡ []
```

One can prove (without assuming that R has decidable equality):

- R[X] has a commutative ring structure
  (See bachelor's thesis of Åkerman Rydbeck [2022])
- It has the universal property of the polynomial ring

Karatsuba in Cubical Agda

Because of drop0, need **mutually recursive** definition:

$$\text{karatsubaRec} : \mathbb{N} \to \text{R[X]} \to \text{R[X]} \to \text{R[X]}$$
$$\text{karatsubaRec}{\equiv} : \forall\ (n : \mathbb{N})\ (p\ q : \text{R[X]}) \to \text{karatsubaRec}\ n\ p\ q \equiv p \cdot q$$

Karatsuba in Cubical Agda

Because of drop0, need **mutually recursive** definition:

```
karatsubaRec : ℕ → R[X] → R[X] → R[X]
karatsubaRec≡ : ∀ (n : ℕ) (p q : R[X]) → karatsubaRec n p q ≡ p · q
```

Base algorithm as presented:

```
karatsubaRec zero p q = p · q -- base case

karatsubaRec (suc n) [] q = {!!} -- mult. w/ 0
karatsubaRec (suc n) [ a₀ ] q = {!!} -- mult. w/ scalar a₀
karatsubaRec (suc n) (a₀ :: a₁ :: p) [] = {!!} -- mult. w/ 0
karatsubaRec (suc n) (a₀ :: a₁ :: p) [ b₀ ] = {!!} -- mult. w/ scalar b₀
karatsubaRec (suc n) (a₀ :: a₁ :: p) (b₀ :: b₁ :: q) = {!!} -- rec. call
```

Filling in the paths

For the higher constructor:

```
karatsubaRec (suc n) (a0 :: a1 :: p) (b0 :: drop0 i) = {!!}
  -- assume (karatsubaRec≡ n)

karatsubaRec (suc n) (a0 :: a1 :: p) (drop0 i) = {!!} -- easy

karatsubaRec (suc n) (a0 :: drop0 i) q = {!!}
  -- assume (karatsubaRec≡ n)

karatsubaRec (suc n) (drop0 i) q = {!!} -- easy
```

Filling in the paths

For the higher constructor:

```
karatsubaRec ( suc n ) ( a0 :: a1 :: p ) ( b0 :: drop0 i ) = {!!}
  -- assume (karatsubaRec≡ n)

karatsubaRec ( suc n ) ( a0 :: a1 :: p ) ( drop0 i ) = {!!} -- easy

karatsubaRec ( suc n ) ( a0 :: drop0 i ) q = {!!}
  -- assume (karatsubaRec≡ n)

karatsubaRec ( suc n ) ( drop0 i ) q = {!!} -- easy
```

Then, since $\forall\ n\ p\ q \to$ isProp (karatsubaRec≡ $n\ p\ q$)

$\Rightarrow$ only cumbersome but straightforward algebra remains

The fuel problem

Want: set karatsuba $p\ q :=$ karatsubaRec $n\ p\ q$
for some n (depending on p, q)

The fuel problem

Want: set karatsuba $p\ q :=$ karatsubaRec $n\ p\ q$
for some n (depending on p, q)

1. elements of R[X] don't have length
   drop0 $\Rightarrow$ [],[0],[0,0],... would need to be assigned same value
2. unless R has decidable equality,
   can't compute the degree of a polynomial either

The fuel problem

Want: set karatsuba $p\ q :=$ karatsubaRec $n\ p\ q$
for some n (depending on p, q)

1. elements of R[X] don't have length
   drop0 $\Rightarrow$ [],[0],[0,0],... would need to be assigned same value
2. unless R has decidable equality,
   can't compute the degree of a polynomial either

But we can define something like length:

$$\text{truncLength} : R[X] \to \| \mathbb{N} \|$$
$$\text{truncLength } [] = |\ 0\ |$$
$$\text{truncLength } (a :: p) = \text{map suc } (\text{truncLength } p)$$
$$\text{truncLength } (\text{drop0 } i) = \text{squash } |\ 1\ |\ |\ 0\ |\ i$$

Factor through $|\_| : \mathbb{N} \to \| \mathbb{N} \|$

Lemma (Kraus [2015])

Let A be a type and B a set. Given a function $f : A \to B$ with a proof of

$$\forall\ (x\ y : A) \to f\,x \equiv f\,y$$

we get a function $|f| : \| A \| \to B$ s.t. $|f|\ |a| = f\,a$ definitionally, for all $a : A$.

Factor through $|\_| : \mathbb{N} \to \| \mathbb{N} \|$

Lemma (Kraus [2015])

Let A be a type and B a set. Given a function $f : A \to B$ with a proof of

$$\forall\ (x\ y : A) \to f\ x \equiv f\ y$$

we get a function $|f| : \| A \| \to B$ s.t. $|f|\ |a| = f\ a$ definitionally, for all $a : A$.

Now, we have

- isSet (R[X] → R[X] → R[X])
- $\forall\ n\ m \to$ karatsubaRec $n \equiv \_\cdot\_ \equiv$ karatsubaRec m

And thus get

karatsubaTruncRec : $\| \mathbb{N} \| \to$ R[X] → R[X] → R[X]

Factor through $|\_| : \mathbb{N} \to \| \mathbb{N} \|$

Lemma (Kraus [2015])

Let A be a type and B a set. Given a function $f : A \to B$ with a proof of

$$\forall\ (x\ y : A) \to f\ x \equiv f\ y$$

we get a function $|f| : \| A \| \to B$ s.t. $|f|\ |a| = f\ a$ definitionally, for all $a : A$.

Now, we have

- isSet (R[X] → R[X] → R[X])
- $\forall\ n\ m \to$ karatsubaRec $n \equiv \_\cdot\_ \equiv$ karatsubaRec m

And thus get

$$\text{karatsubaTruncRec} : \| \mathbb{N} \| \to R[X] \to R[X] \to R[X]$$

Exercise: Define a different karatsubaTruncRec' using the contractibility of the singleton type of $\_\cdot\_$.
How does their computational behavior compare?

Putting it all together

$$\text{karatsuba} : \text{R[X]} \to \text{R[X]} \to \text{R[X]}$$
$$\text{karatsuba}\ p\ q = \textbf{let}\ \mathit{fuel} = \text{map2 max}\ (\text{truncLength}\ p)\ (\text{truncLength}\ q)$$
$$\textbf{in}\ \text{karatsubaTruncRec}\ \mathit{fuel}\ p\ q$$

Putting it all together

$$\text{karatsuba} : \text{R[X]} \to \text{R[X]} \to \text{R[X]}$$
$$\text{karatsuba}\ p\ q = \text{let}\ \mathit{fuel} = \text{map2 max (truncLength}\ p)\ (\text{truncLength}\ q)$$
$$\text{in karatsubaTruncRec}\ \mathit{fuel}\ p\ q$$

Exercises/sanity checks:

1. Convince yourself that this computes as it should (on list polynomials without drop0)
2. Prove karatsuba $\equiv$ _·_

The general scheme
(programming with HITs and fuel)

1. Implement a provably correct version of some algorithm
   - In our setting: *Fuel* for the termination checker
   - Bound for fuel depends on implementation of input data

The general scheme (programming with HITs and fuel)

1. Implement a provably correct version of some algorithm
   - In our setting: *Fuel* for the termination checker
   - Bound for fuel depends on implementation of input data

2. HITs for our input data that make the correctness proof easier but break the algorithm
   - (Exception in our setting: decidable equality case)

The general scheme (programming with HITs and fuel)

1. Implement a provably correct version of some algorithm
   - In our setting: *Fuel* for the termination checker
   - Bound for fuel depends on implementation of input data
2. HITs for our input data that make the correctness proof easier but break the algorithm
   - (Exception in our setting: decidable equality case)
3. Factor through $|\_| : \mathbb{N} \to \| \mathbb{N} \|$ to be able to program with HITs!

The general scheme (programming with HITs and fuel)

1. Implement a provably correct version of some algorithm
   - In our setting: *Fuel* for the termination checker
   - Bound for fuel depends on implementation of input data
2. HITs for our input data that make the correctness proof easier but break the algorithm
   - (Exception in our setting: decidable equality case)
3. Factor through $|\_| : \mathbb{N} \to \| \mathbb{N} \|$ to be able to program with HITs!

Future Work:

1. make a runable program using `--erased-cubical`
2. more algorithm examples

Thank You

Jounaidi Abdeljaoued and Henri Lombardi. *Méthodes matricielles-Introduction à la complexité algébrique*. Springer Science & Business Media, 2004.

Maxime Dénès, Anders Mörtberg, and Vincent Siles. A Refinement-Based Approach to Computational Algebra in Coq. In Lennart Beringer and Amy Felty, editors, *Interactive Theorem Proving*, volume 7406 of *Lecture Notes in Computer Science*, pages 83–98. Springer Berlin Heidelberg, 2012. ISBN 978-3-642-32346-1. doi: 10.1007/978-3-642-32347-8_7. URL http://dx.doi.org/10.1007/978-3-642-32347-8_7.

Anatolii Alekseevich Karatsuba and Yu P Ofman. Multiplication of many-digital numbers by automatic computers. In *Doklady Akademii Nauk*, volume 145, pages 293–294. Russian Academy of Sciences, 1962.

Nicolai Kraus. The general universal property of the propositional truncation. In Hugo Herbelin, Pierre Letouzey, and Matthieu Sozeau, editors, *20th International Conference on Types for Proofs and Programs (TYPES 2014)*, volume 39 of *Leibniz International Proceedings in Informatics (LIPIcs)*, pages 111–145, Dagstuhl, Germany, 2015. ISBN 978-3-939897-88-0. doi:

http://dx.doi.org/10.4230/LIPIcs.TYPES.2014.111. URL http://drops.dagstuhl.de/opus/volltexte/2015/5494.

Gustav Sahlin. A formal proof of karatsuba's algorithm in agda. Bachelor's thesis, Stockholm University, 2022. https://github.com/gustav5/Karatsuba-for-Polynomials-in-Agda/blob/main/2022_thesis.pdf.

Carl Åkerman Rydbeck. Formalisation of polynomials in cubical type theory using cubical agda. Bachelor's thesis, Stockholm University, 2022. https://su.diva-portal.org/smash/record.jsf?pid=diva2%3A1686693.