Towards abstract and executable multivariate polynomials in Isabelle

Florian Haftmann

Andreas Lochbihler

Wolfgang Schreiner

Institute for Informatics TU Munich Institute of Information Security ETH Zurich Jo

RISC Johannes Kepler University Linz

Isabelle 2014

A cultural "gap" between two communities.

- Theorem proving:
 - Sound formal development of theories on top of a small trusted kernel.
 - Computations reduced to logical inferences.
 - Correct but inconvenient to use and painfully slow.
- Computer algebra:
 - > Elaboration of mathematics by paper-and-pencil or TP software.
 - Separate implementation in mathematical software systems.
 - Convenient to use and reasonably fast but highly untrustworthy.

How can we bridge this gap?

An Isabelle package in which the working mathematician can develop

- Mathematical theories based on an abstract view of polynomials.
 - Type-checked definitions and theorems.
 - Computer-supported/mechanically verified proofs.
- Algorithms based on the defined mathematical notions.
 - Executable with "reasonable" efficiency (rapid prototyping).
 - Formal specification and computer-supported verification.

A single computer-supported formal framework for proving and computing with (multivariate) polynomials.

Polynomials

What is the polynomial written as $2x^3 - 5x + 7$?

Traditional: the symbolic expression itself.

$$\left(\sum_{i=0}^{n} a_{i} x^{i}\right) \cdot \left(\sum_{j=0}^{m} b_{j} x^{j}\right) = \sum_{k=0}^{m+n} \left(\sum_{i \in \mathbb{N}_{0}, j \in \mathbb{N}_{0}}^{i+j=k} a_{i} \cdot b_{j}\right) \cdot x^{k}$$

► Computer science: an array [7, 5, 0, 3]

```
int[] mult(int[] a, int[] b)
{
    int m = a.length-1; int n = b.length-1;
    int[] c = new int[m+n+1];
    for (int i = 0; i <= m; i++)
        for (int j = 0; j <= n; j++)
            c[i+j] += a[i]*b[j];
    return c;
}</pre>
```

Two representations of a more fundamental concept.

Polynomials

The more fundamental concept is the modern view of polynomials.

▶ Polynomial: a function $[0 \mapsto 7, ..., 3 \mapsto 3, 4 \mapsto \underline{0}, 5 \mapsto \underline{0}, ...]$

Let R be a ring. A (univariate) polynomial over R is a mapping $p : \mathbb{N}_0 \to R, n \mapsto p_n$, such that $p_n = 0$ nearly everywhere, i.e., for all but finitely many values of n.

Elegant mathematics:

Polynomial ring: R[x]

The set of polynomials with (+) and (·); variable x just denotes the polynomial $[0 \mapsto 0, 1 \mapsto 1, 2 \mapsto 0, 3 \mapsto 0, \ldots]$.

See e.g. [Winkler, 1996].

Multivariate Polynomials

What is a polynomial $3x^2y + 5yz$ in variables x, y, z?

▶ Polynomial: a function $[(1,1,0) \mapsto 3, (0,1,1) \mapsto 5, (0,0,0) \mapsto 0, \ldots]$

An n-variate polynomial over the ring R is a mapping $p : \mathbb{N}_0^n \to R, (i_1, \ldots, i_n) \mapsto p_{i_1, \ldots, i_n}$, such that $p_{i_1, \ldots, i_n} = 0$ nearly everywhere.

• Polynomial ring: $R[x_1, \ldots, x_n]$

The set of all n-variate polynomials over R; variable x_i denotes the polynomial $[\ldots, i \mapsto 1, \ldots]$.

► Isomorphism: $R[x_1, \ldots, x_n] \simeq (R[x_1, \ldots, x_{n-1}])[x_n].$

Recursive algorithms may be devised for many (not all) computational problems on multivariate polynomials.

- Polynomial division is defined on K[x] where K is a field.
- But $K[x_1, \ldots, x_{n-1}]$ is only a ring.
- Multivariate polynomials thus only support "pseudo-division".

Prune mapping:

Represent only exponents/monomials with non-zero coefficients.

- Univariate polynomial representations:
 - ▶ Dense: coefficient sequence [*c*₀,...,*c*_n]
 - Sparse: exponent/coeff. sequence $[(e_0, c_0), \ldots, (e_r, c_r)]$ with $e_i < e_{i+1}$.
- *n*-variate polynomial representations:
 - ► Recursive: univariate polynomial whose coefficients are (n − 1)-variate polynomials (represented densely or sparsely).
 - ► Distributive: monomial/coefficient sequence [(m₀, c₀), ..., (m_r, c_r)] (typically represented sparsely).
 - > Total order on monomials required for unique representation.

Algorithmic efficiency:

- Recursive algorithms based on isomorphism operate most efficiently with recursive representation.
- Buchberger's Gröbner bases algorithm processes terms in any given "admissible" order and profits from distributive rep. in that order.

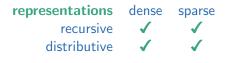
abstract type

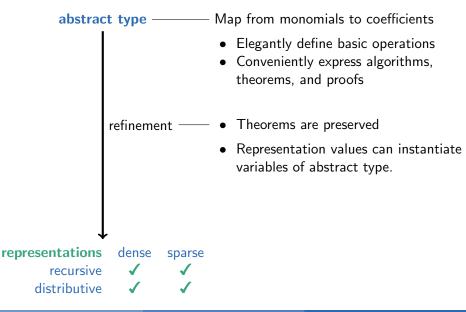
representationsdensesparserecursive✓✓distributive✓✓

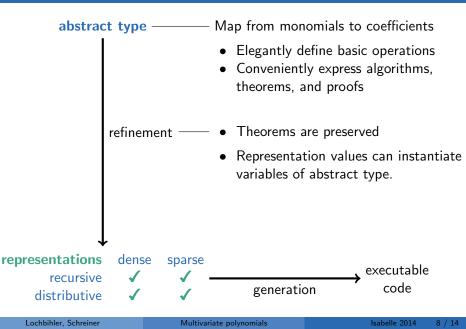
Lochbihler, Schreiner

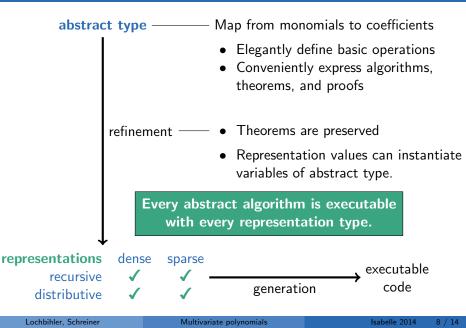
abstract type ------ Map from monomials to coefficients

- Elegantly define basic operations
- Conveniently express algorithms, theorems, and proofs









typedef $a \Rightarrow_0 b = \{ f :: a \Rightarrow b \mid almost-everywhere-zero f \}$

'a mpoly

'a poly-rec

'a poly-distr

typedef $a \Rightarrow_0 b = \{ f :: a \Rightarrow b \mid almost-everywhere-zero f \}$

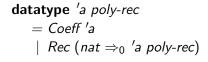
$$(\textit{nat} \Rightarrow_0 \textit{nat}) \Rightarrow_0 'a \ \cong \ 'a \textit{mpoly}$$

'a poly-rec

'a poly-distr

typedef $a \Rightarrow_0 b = \{ f :: a \Rightarrow b \mid almost-everywhere-zero f \}$

$$(\textit{nat} \Rightarrow_0 \textit{nat}) \Rightarrow_0 'a \ \cong \ 'a \textit{mpoly}$$

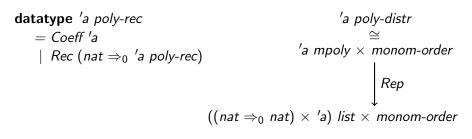


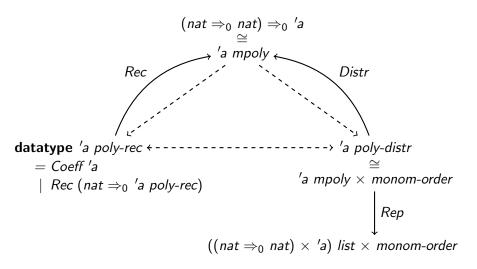
'a poly-distr

$$(\textit{nat} \Rightarrow_0 \textit{nat}) \Rightarrow_0 'a \ \cong \ 'a \textit{mpoly}$$

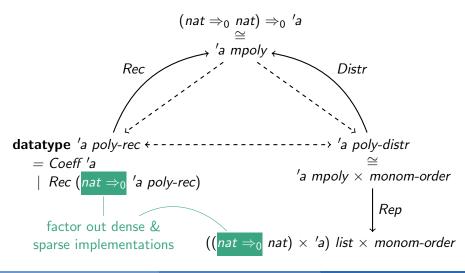


$$(\textit{nat} \Rightarrow_0 \textit{nat}) \Rightarrow_0 'a \ \cong \ 'a \textit{mpoly}$$





typedef $a \Rightarrow_0 b = \{ f :: a \Rightarrow b \mid almost-everywhere-zero f \}$



Lochbihler, Schreiner

Should the number of variables show up in the type?

'a mpoly vs. 'a poly poly . . . poly vs. ('a, 7) mpoly

Should the number of variables show up in the type?

- 'a mpoly vs. 'a poly poly . . . poly vs. ('a, 7) mpoly
- Algorithms change number of variables dynamically.
- No computation on types

 $('a, 4 + 3) mpoly \neq ('a, 2 + 5) mpoly$

Should the number of variables show up in the type?

- Algorithms change number of variables dynamically.
- No computation on types

 $('a, 4 + 3) mpoly \neq ('a, 2 + 5) mpoly$

Should the number of variables show up in the type?

- Algorithms change number of variables dynamically.
- No computation on types

 $('a, 4 + 3) mpoly \neq ('a, 2 + 5) mpoly$

Should the number of variables show up in the type?

- Algorithms change number of variables dynamically.
- No computation on types

$$('a, 4 + 3) mpoly \neq ('a, 2 + 5) mpoly$$

 Polynomials over an unbounded number of variables
 Derived notion variable number: the highest index of a variable with non-zero coefficient.

Implicitly extend polynomials as needed.

Example:

- Gröbner bases algorithm depends on a **monomial order**.
- Efficiency relies on fast access to leading monomial in that order.

'a mpoly vs. 'a mpoly \times monom-order

Example:

- Gröbner bases algorithm depends on a **monomial order**.
- Efficiency relies on fast access to leading monomial in that order.

'a mpoly vs. 'a mpoly \times monom-order

► Algebraic type classes require uniqueness of polynomials.

Example:

- Gröbner bases algorithm depends on a **monomial order**.
- Efficiency relies on fast access to leading monomial in that order.

'a mpoly vs. 'a mpoly × monom-order

- Algebraic type classes require uniqueness of polynomials.
- > Algorithm receives representational details as parameter.
- If polynomial's representation fits to the parameter, execution is fast.
 Otherwise, convert polynomial ... or search ...
- No static checks, no efficiency guarantees!

Open Problem: Controlling Representations

What happens when we combine two polynomials?

+	Rec	Distr.
Rec	Rec	???
Distr	???	Distr

How can we make contextual information available?

Open Problem: Controlling Representations

What happens when we combine two polynomials?

+	Rec	Distr.
Rec	Rec	???
Distr	???	Distr

How can we make contextual information available?

How can the user specify the representations?

value (2 :: int poly) * 3

Recursive or distributive? Dense or Sparse? Which monomial order?

In CAS, the user declares his choice as a configuration option. Can we mimick this in Isabelle?

Lochbihler, Schreiner

The Ubiquituous Type Class zero

typedef $'a \Rightarrow_0 'b = \{ f :: 'a \Rightarrow 'b :: zero \mid almost-everywhere-zero f \}$

There is no map function for 'b that satisfies

 $map \ f \circ map \ g = map \ (f \circ g)$

The Ubiquituous Type Class zero

typedef $'a \Rightarrow_0 'b = \{ f :: 'a \Rightarrow 'b :: zero \mid almost-everywhere-zero f \}$

There is no map function for 'b that satisfies

 $map \ f \circ map \ g = map \ (f \circ g)$

 $\begin{array}{l} \mathsf{BNF} \Rightarrow_0 \text{ is not a BNF!} \\ \mathsf{Must construct }'a \ \textit{poly-rec manually} \end{array}$

- Lifting Quotient theorem only for relations that respect *zero* No parametrised correspondence relations
- Transfer Transfer rules must restrict function space ===> is too weak
 - Library Re-implement finite maps with the invariant $0 \notin ran m$ How can we improve reuse?

Current state of multivariate polynomials in Isabelle:

- Design seems good
- Prototype of abstract and representation types with minimal set of operations
- Lemmas and algorithm implementations are still missing

Current state of multivariate polynomials in Isabelle:

- Design seems good
- Prototype of abstract and representation types with minimal set of operations
- Lemmas and algorithm implementations are still missing

Up for discussion:

- ► User-friendliness/convenience for the working mathematician.
- Control of representations
- Better integration with Isabelle packages