Dpto. de Ciencias de la Computación e Inteligencia Artificial
UNIVERSIDAD DE SEVILLA
Dpto. de Lenguajes y Sistemas Informáticos
UNIVERSIDAD DE CADIZ

Automatic Verification of Polynomial Rings Fundamental Properties in ACL2

Inmaculada Medina Bulo et al.
ACL2 Workshop 2000.

Introduction

- Goals
  - Formalization of multivariate *polynomials* over a coefficient field, $\mathbb{Q}$, and their basic operations in ACL2
  - Verification of their main properties
  - Computation by using their operations
- Main findings
  - Polynomial formalization
  - Automatic verification of fundamental properties that structure them as a ring
    - $\langle K[X], +, -, 0 \rangle$ form a commutative group
    - $\langle K[X], \cdot, 1 \rangle$ form a commutative monoid
    - $\cdot$ is distributive over $+$ on the right and on the left
  - Computation by using the operations
- Potential applications
  - The formalization of Buchberger's algorithm

Polynomial Representation Problems

- Normalized/Unnormalized Representation

  1. Normalized Representation and Syntactic Equality

     – Advantages

       Equality is syntactic and ACL2 handles it directly (`EQUAL`)

     – Disadvantages

       We have to work in normal form. This complicates the proofs

  2. Unnormalized Representation and Semantic Equality

     – Advantages

       It spares the operations from the need to work with normal forms. The computation done by the algorithm is separated from the normalization process

     – Disadvantages

       Equality must work module normal form and the prover does not manage it directly

Polynomial Representation Problems

- Dense/Sparse Representation

1. Dense
   - Advantages

     Simple algorithms
   - Disadvantages

     Unsuitable for the case of multiple variables
2. Sparse
   - Advantages

     Suitable for multivariate polynomials
   - Disadvantages

     More complex algorithms

Polynomial Representation

We have chosen

- Initially, a sparse normalized representation
- Finally, a sparse unnormalized representation
- Formalization
  - A polynomial is a finite sum of monomials
  - A semantic equality predicate
  - Necessary operations (addition, negation, multiplication)
  - Verification of the fundamental properties of polynomials
  - A monomial is a product between a coefficient and a term

Formalization of Terms

- Definition

  A term on a set of variables X is a finite power product of the form

$$x_1^{e_1} \dots x_n^{e_n} = X^{\langle e_1, \dots, e_n \rangle} = \prod_{i=1}^{n} x_i^{e_i}$$

- Representation

  A list of natural numbers, once we have determined X and an order $<_X$ over X.

  For example, $<_X = \{(x_i, x_j) : i < j\}$

$$x_1^{e_1} \cdot \dots \cdot x_n^{e_n} = X^{\langle e_1, \dots, e_n \rangle} \longrightarrow \langle e_1, \dots, e_n \rangle$$

- Null Term

$$1 = x_1^0 \cdot \dots \cdot x_n^0 = X^{\langle 0, \dots, 0 \rangle} \longrightarrow \langle 0, \dots, 0 \rangle$$

Terms in ACL2

- Recognizer of terms

```
(defmacro termp (a)
  `(natural-listp ,a))
```

- Null term

```
(defconst *null*
  nil)

(defun nullp (a)
  (cond ((atom a)
         (equal a *null*))
        (t
         (and (equal (first a) 0)
              (nullp (rest a))))))
```

- Compatibility and equality relation

```
(defmacro compatiblep (a b)
  `(equal (len ,a) (len ,b)))

(defmacro = (a b)
  `(equal ,a ,b))
```

Multiplication of Terms

- Definition

$$X^{\langle a_1,\ldots,a_n\rangle} \cdot X^{\langle b_1,\ldots,b_n\rangle} = X^{\langle a_1+b_1,\ldots,a_n+b_n\rangle}$$

```
(defun * (a b)
  (cond ((and (not (termp a)) (not (termp b)))
         *null*)
        ((not (termp a)) b)
        ((not (termp b)) a)
        ((endp a) b)
        ((endp b) a)
        (t
         (cons (LISP::+ (first a) (first b))
               (* (rest a) (rest b))))))
```

- Commutative Monoid Structure

```
(defthm *-identity-1
  (implies (and (nullp a) (termp b)
                (compatiblep a b))
           (= (* a b) b)))

(defthm *-identity-2
  (implies (and (termp a) (nullp b)
                (compatiblep a b))
           (= (* a b) a)))

(defthm associativity-of-*
  (= (* (* a b) c) (* a (* b c))))

(defthm commutativity-of-*
  (= (* a b) (* b a)))
```

Total and Strict Order on Terms

- Definition (lexicographical ordering)

$$X^{\langle a_1,\ldots,a_n \rangle} < X^{\langle b_1,\ldots,b_n \rangle} \equiv$$

$$\langle a_1,\ldots,a_n \rangle < \langle b_1,\ldots,b_n \rangle \equiv$$

$$\exists i \left(a_i < b_i \wedge \forall j < i \;\; a_j = b_j \right)$$

```
(defun < (a b)
  (cond ((or (endp a) (endp b))
         (not (endp b)))
        ((equal (first a) (first b))
         (< (rest a) (rest b)))
        (t
         (LISP::< (first a) (first b)))))
```

- Properties of the order

```
(defthm irreflexivity-of-<
  (not (< a a)))

(defthm transitivity-of-<
  (implies (and (< a b) (< b c))
           (< a c)))

(defthm trichotomy-of-<
  (implies (and (termp a) (termp b))
           (or (< a b) (< b a) (= a b)))
  :rule-classes nil)
```

Term Embedding in ϵ_0-ordinals

- Formalization

$$X^{\langle a_1,\ldots,a_n\rangle} \longmapsto \omega^{\omega^n + a_1} + \cdots + \omega^{\omega + a_n}$$

$$\underbrace{x}_{\texttt{(1)}} \longmapsto \underbrace{\omega^{\omega+1}}_{\texttt{((1 . 1) . 0)}}$$

$$\underbrace{x^8 \cdot y^0}_{\texttt{(8 0)}} \longmapsto \underbrace{\omega^{\omega^2+8} + \omega^{\omega}}_{\texttt{((2 . 8) (1 . 0) . 0)}}$$

$$\underbrace{x^4 \cdot y^3 \cdot z^5}_{\texttt{(4 3 5)}} \longmapsto \underbrace{\omega^{\omega^3+4} + \omega^{\omega^2+3} + \omega^{\omega+5}}_{\texttt{((3 . 4) (2 . 3) (1 . 5) . 0)}}$$

- Definition

```
(defun term->e0-ordinal (a)
  (cond ((endp a)
         0)
        (t
         (cons (cons (len a) (first a))
               (term->e0-ordinal (rest a))))))
```

Well-founded Order

```
(defthm e0-ordinalp-term->e0-ordinal
  (implies (termp a)
           (e0-ordinalp (term->e0-ordinal a)))
  :hints (("Goal" ...)))

(defthm well-ordering-of-<
  (and (implies (termp a)
                (e0-ordinalp
                   (term->e0-ordinal a)))
       (implies (and (termp a) (termp b)
                     (< a b))
             (e0-ord-< (term->e0-ordinal a)
                       (term->e0-ordinal b))))
  :rule-classes :well-founded-relation)
```

- Problem

```
(< '(3 1) '(1 2 1)) ⟶ nil

(term->e0-ordinal '(3 1))
⟶ ((2 . 3) (1 . 1) . 0)

(term->e0-ordinal '(1 2 1))
⟶ ((3 . 1) (2 . 2) (1 . 1) . 0)

(e0-ord-< '((2 . 3) (1 . 1) . 0)
          '((3 . 1) (2 . 2) (1 . 1) . 0))
⟶ t
```

- Solution

```
(defun < (a b)
  (cond ((LISP::< (len a) (len b))
         t)
        ((LISP::> (len a) (len b))
         nil)
        (...
```

Admissibility of the Order

- Definition
  - $\forall a \in [X] \backslash \{1\} \; 1 = x_1^0 \cdot \cdots \cdot x_n^0 < a$
  - $\forall a, b, c \in [X] \; (a < b \implies ac < bc)$
- Formalization
  - The order has a first element

```
(defthm <-has-first
  (implies (and (termp a) (termp b)
                (compatiblep a b)
                (nullp a) (not (nullp b)))
           (< a b)))
```

  - The order is compatible with the multiplication

```
(defthm <-compatible-*-1
  (implies (and (termp a) (termp b)
                (termp c)
                (compatiblep a c)
                (compatiblep b c)
                (< a b))
           (< (* a c) (* b c))))

(defthm <-compatible-*-2
  (implies (and (termp a) (termp b)
                (termp c)
                (compatiblep a c)
                (compatiblep b c)
                (< a b))
           (< (* c a) (* c b))))
```

Formalization of Monomials

- Definition

  A monomial on X is a product of the form

$$c \cdot X^{\langle e_1, \ldots, e_n \rangle}$$

- Representation

  A list whose first element is its coefficient and whose rest is its term

$$c \cdot X^{\langle e_1, \ldots, e_n \rangle} \longrightarrow (c \;\; (e_1 \; \ldots \; e_n))$$

- Identity Monomial

$$1 \cdot X^{\langle 0, \ldots, 0 \rangle} \longrightarrow (1 \;\; (0 \; \ldots \; 0))$$

Monomials in ACL2

- Recognizer of monomials

```
(defmacro monomialp (a)
  ‘(and (consp ,a) (rationalp (first ,a))
        (termp (rest ,a))))
```

- Identity and Null Monomial

```
(defconst *one* (monomial 1 TER::*null*))

(defmacro onep (a)
  ‘(and (equal (coefficient ,a) 1)
        (TER::nullp (term ,a))))

(defconst *null* (monomial 0 TER::*null*))

(defmacro nullp (a)
  ‘(equal (coefficient ,a) 0))
```

- Compatibility and equality relation

```
(defun compatiblep (a b)
  (TER::compatiblep (term a) (term b)))

(defun = (a b)
  (or (and (nullp a) (nullp b))
      (and (LISP::= (coefficient a)
                    (coefficient b))
           (TER::= (term a) (term b)))))
```

Multiplication of Monomials

- Definition

$$aX^{\langle a_1,\ldots,a_n\rangle} \cdot bX^{\langle b_1,\ldots,b_n\rangle} =$$

$$(a \cdot b)X^{\langle a_1+b_1,\ldots,a_n+b_n\rangle}$$

```
(defun * (a b)
  (monomial (LISP::* (coefficient a)
                     (coefficient b))
            (TER::* (term a) (term b))))
```

- Commutative Monoid Structure

```
(defthm *-identity-1
  (implies (and (onep a) (monomialp b)
                (compatiblep a b))
           (= (* a b) b)))

(defthm *-identity-2
  (implies (and (monomialp a) (onep b)
                (compatiblep a b))
           (= (* a b) a)))

(defthm associativity-of-*
  (= (* (* a b) c) (* a (* b c))))

(defthm commutativity-of-*
  (= (* a b) (* b a)))
```

Formalization of Polynomials

- Definition

  A polynomial on X is a finite sum of monomials

$$c_1 \cdot X^{\langle e_{11}, \ldots, e_{1n} \rangle} + \cdots + c_m \cdot X^{\langle e_{m1}, \ldots, e_{mn} \rangle}$$

- Representation

$$((c_1 \ (e_{11} \ \ldots \ e_{1n})) \ \ldots \ (c_m \ (e_{m1} \ \ldots \ e_{mn})))$$

- Recognizer

```
(defmacro polynomialp (p)
  `(monomial-listp ,p))
```

- Null polynomial

```
(defconst *null* nil)

(defmacro nullp (p) `(endp ,p))
```

- Identity polynomial

```
(defconst *one*
  (polynomial MON::*one* *null*))

(defmacro onep (p) `(= ,p *one*))
```

Polynomial Semantic Equality

The equality relation defined on polynomials must verify the following properties

1. The reflexive property
2. The symmetrical property
3. The transitive property
4. $p_1 +_p (m +_m p_2) =_p m +_m (p_1 +_p p_2)$
5. $p_1 =_p p_2 \wedge q_1 =_p q_2 \implies p_1 +_p q_1 =_p p_2 +_p q_2$
6. $(k_1, t) +_m ((k_2, t) +_m p) =_p ((k_1 +_k k_2), t) +_m p$
7. $m = 0 \implies m +_m p =_p p$

Polynomial Equality in ACL2

- Semantic equality

```
(defun = (p1 p2)
  (equal (nf p1) (nf p2)))
```

- Normal form
  - Monomials are strictly ordered
  - Null monomials do not appear

  This implies that monomials with identical terms can not appear

```
(defun nf (p)
  (cond ((or (not (polynomialp p)) (nullp p))
         *null*)
        (t
         (+-monomial (first p) (nf (rest p))))))

(defun +-monomial (m p)
  (cond
   ((MON::nullp m) p)
   ((nullp p) (polynomial m *null*))
   ((TER::= (term m) (term (first p)))
    (let ((c (LISP::+ (coefficient m)
                      (coefficient (first p)))))
      (if (equal c 0) (rest p)
        (polynomial (monomial c (term m))
                    (rest p)))))
   ((TER::< (term (first p)) (term m))
    (polynomial m p))
   (t
    (polynomial (first p)
                (+-monomial m (rest p))))))
```

Addition & Negation of Polynomials

- Polynomial Addition

```
(defun + (p1 p2)
  (cond ((and (not (polynomialp p1))
              (not (polynomialp p2))) *null*)
        ((not (polynomialp p1)) p2)
        ((not (polynomialp p2)) p1)
        (t (append p1 p2))))
```

- Polynomial Negation

```
(defun - (p)
  (cond ((or (not (polynomialp p)) (nullp p))
         *null*)
        (t (polynomial
            (monomial (LISP::- (coefficient
                                (first p)))
                      (term (first p)))
            (- (rest p))))))
```

- Commutative Group with Addition and Negation

```
(defthm --distributes-+
  (= (- (+ p1 p2)) (+ (- p1) (- p2))))

(defthm +-identity-1 (= (+ p *null*) p))

(defthm +-identity-2 (= (+ *null* p) p))

(defthm associativity-of-+
  (= (+ (+ p1 p2) p3) (+ p1 (+ p2 p3))))

(defthm commutativity-of-+
  (= (+ p1 p2) (+ p2 p1)))

(defthm +-- (= (+ p (- p)) *null*)))
```

Multiplication of Polynomials

- Definition

```
(defun *-monomial (m p)
  (cond ((or (nullp p) (not (monomialp m))
             (not (polynomialp p)))
         *null*)
        (t (polynomial (MON::* m (first p))
                       (*-monomial m (rest p))))))

(defun * (p1 p2)
  (cond ((or (nullp p1)
             (not (polynomialp p1)))
         *null*)
        (t
         (+ (*-monomial (first p1) p2)
            (* (rest p1) p2)))))
```

- Commutative Monoid with Multiplication

```
(defthm *-identity-1 (= (* *one* p) p))

(defthm *-identity-2 (= (* p *one*) p))

(defthm associativity-of-*
  (= (* p1 (* p2 p3)) (* (* p1 p2) p3)))

(defthm commutativity-of-*
  (= (* p1 p2) (* p2 p1)))

(defthm *-cancellative-1
  (= (* *null* p) *null*))

(defthm *-cancellative-2
  (= (* p *null*) *null*))
```

Distributivity Property

- Distributivity of Multiplication over Addition

```
(defthm *-distributes-+-1
  (= (* p1 (+ p2 p3))
     (+ (* p1 p2) (* p1 p3))))

(defthm *-distributes-+-2
  (= (* (+ p1 p2) p3)
     (+ (* p1 p3) (* p2 p3))))
```

This completes the proof that polynomials have a ring structure

Congruences

- Polynomial constructor

```
(defcong MON::= = (polynomial m p) 1)

(defcong = = (polynomial m p) 2)
```

- Negation and addition of polynomials

```
(defcong = = (- p) 1))

(defcong = = (+ p1 p2) 2)
(defcong = = (+ p1 p2) 1)
```

- Multiplication between monomials and polynomials

```
(defcong MON::= = (*-monomial m p) 1)

(defthm =-implies-=-*-monomial-2
 (implies (and (monomialp m)
               (polynomialp p1)
               (polynomialp p1-equiv)
               (MON::compatiblep m (first p1))
               (compatiblep p1 p1-equiv)
               (= p1 p1-equiv))
          (= (*-monomial m p1)
             (*-monomial m p1-equiv))))
```

Congruences

- Multiplication of polynomials

```
(defthm =-implies-=-*-2
  (implies (and (polynomialp p1)
                (polynomialp p2)
                (polynomialp p2-equiv)
                (compatiblep p1 p2)
                (compatiblep p1 p2-equiv)
                (= p2 p2-equiv))
           (= (* p1 p2) (* p1 p2-equiv))))

(defthm =-implies-=-*-1
  (implies (and (polynomialp p1)
                (polynomialp p1-equiv)
                (polynomialp p2)
                (compatiblep p1 p2)
                (compatiblep p1-equiv p2)
                (= p1 p1-equiv))
           (= (* p1 p2) (* p1-equiv p2))))
```

Conclusions and Future Work

- Conclusions
  - A formalization of multivariate polynomials rings with rational coefficients in ACL2
  - It is interesting to note some of the advantages exposed by ACL2 in comparison with NQTHM
  - Compatibility relation complicate the proofs
  - Guards: Operations can be executed on any platform
- Future Work
  - Abstraction of the coefficient field
  - Obtaining an automatic verification of Buchberger's algorithm for Gröbner bases
  - There are many applications of Gröbner bases