Algoritmos polinómicos en ACL2
(Una aproximación al algoritmo de Buchberger) *

Inmaculada Medina[1], José A. Alonso[2] y Francisco Palomo[1]

[1] Depto. de Lenguajes y Sistemas Informáticos. Univ. de Cádiz.
Esc. Superior de Ingeniería de Cádiz. C/ Chile, s/n. 11003 Cádiz. España.
{Inmaculada.Medina, Francisco.Palomo, }@uca.es
[2] Depto. de Ciencias de la Computación e Inteligencia Artificial. Univ. de Sevilla.
E.T.S. de Ingeniería Informática Avda. Reina Mercedes, s/n. 41012 Sevilla. España.
Jose-Antonio.Alonso@cs.us.es

Resumen Presentamos algunos resultados sobre la formalización en ACL2 de elementos del álgebra de polinomios. Éstos incluyen el desarrollo y verificación, en paralelo, de procedimientos de cálculo algebraico efectivos sobre polinomios. Además, presentamos una aproximación a la verificación en ACL2 del algoritmo de Buchberger para calcular bases de Gröbner.

1 Introducción y objetivos

El Álgebra Computacional ha sufrido un gran desarrollo en los última década con la proliferación de numerosos sistemas. Éstos son la culminación de los resultados teóricos obtenidos en el último medio siglo. Uno de los descubrimientos centrales fue debido a B. Buchberger cuando en 1965 proporcionó un algoritmo para construir bases de Gröbner. El algoritmo de Buchberger, aunque inicialmente no tuvo mucha difusión, supuso una revolución en áreas muy diversas. Se emplea fundamentalmente para resolver el problema de la pertenencia al ideal en un anillo de polinomios y para decidir la congruencia inducida por el ideal.

Sorprende la riqueza de estructuras algebraicas subyacentes a la propia definición del algoritmo de Buchberger. Hay que emplear un cuerpo conmutativo de coeficientes para definir los monomios con los que construir polinomios multivariables, los monomios deben poseer un orden admisible con el que inducir otro sobre los polinomios. Luego hay que construir funciones de reducción apropiadas entre los polinomios.

Pero asegurar su corrección aumenta la variedad: hay que definir el concepto de ideal polinómico finitamente generado y su congruencia inducida. Por otro lado, las funciones de reducción conviene estudiarlas en el marco, más general, de las relaciones de reducción abstractas que proporciona la teoría general de la reescritura. Todo esto lleva a poder definir con precisión la noción de base

* Este trabajo ha sido parcialmente financiado por el proyecto TIC2000–1368–CO3–02 del Ministerio de Ciencias y Tecnología.

de Gröbner y, en este esfuerzo adicional, radica la diferencia entre *programar* el algoritmo y *demostrar* que es correcto.

Pero ¿es posible aunar programación y demostración de manera que su desarrollo pueda acometerse en paralelo con un alto grado de fiabilidad? La respuesta nos la dan los sistemas de razonamiento automatizado.

Nuestro objetivo final es obtener una implementación verificada del algoritmo de Buchberger en el sistema ACL2 [4]. Presentaremos aquí un plan para alcanzarlo, las dificultades existentes y cómo solucionarlas. Este objetivo ha sido ya alcanzado por L. Théry [11] en el sistema COQ. Existe también un proyecto en MIZAR [10] encaminado a la formalización de las bases de Gröbner.

No obstante, las lógicas subyacentes a COQ y MIZAR son muy distintas a la de ACL2. La lógica ACL2 es, con mucho, más primitiva y no presenta distinción entre el lenguaje de programación y el de especificación. Como contrapartida se obtiene directamente la ejecutabilidad y es posible alcanzar un grado de automatización elevado.

Tres aspectos definen a ACL2. Primero, ACL2 es una lógica de primer orden con igualdad —sin cuantificadores ni tipos— que emplea funciones recursivas totales; consta además de un principio de extensión denominado *encapsulado*[1] que permite cierta expresividad de orden superior. ACL2 también es un lenguaje de programación aplicativo, un subconjunto funcional puro de LISP y por ende, ejecutable. Por último, ACL2 es un sistema de razonamiento automatizado: un demostrador de teoremas heurístico con reescritura ordenada e inducción ordinal.

2 El problema de la pertenencia al ideal

Uno de los problemas principales del Álgebra Conmutativa es el de la pertenencia al ideal. Las siguientes definiciones generales nos permitirán acordar la notación que emplearemos. En lo sucesivo, sea A un anillo conmutativo.

Definición 1. *$I \subseteq A$ es un ideal de A si permanece cerrado bajo la suma y bajo el producto por elementos de A.*

Definición 2. *El ideal generado por $B \subseteq A$, escrito $\langle B \rangle$, es el conjunto de las combinaciones lineales de los elementos de B con coeficientes en A. Se dice que B es la base de $\langle B \rangle$ y que un ideal está finitamente generado si posee base finita.*

Definición 3. *La congruencia inducida por un ideal I, escrito $\equiv_I$, se define por $f \equiv_I g \iff f - g \in I$*

Proposición 1. *El problema de la pertenencia a un ideal I es decidible si, y sólo si, la congruencia del ideal $\equiv_I$ es decidible.*

Demostración. Inmediata a partir de la definición de $\equiv_I$.

[1] Este principio permite introducir una función restringida por axiomas siempre que se proporcione un modelo para la teoría generada.

3 Anillos de polinomios

Considérese un anillo $A = K[x_1, \ldots, x_k]$ de polinomios de $k \in \mathbb{N}^*$ variables sobre un cuerpo conmutativo arbitrario K. Los elementos de A son polinomios en las indeterminadas $x_1, \ldots, x_k$ con coeficientes en K. Éstos están constituidos por monomios de A, que son productos de potencias de la forma $c \cdot x_1^{a_1} \cdots x_n^{a_k}$, donde $c \in K$ es el coeficiente y $x_1^{a_1} \cdots x_k^{a_k}$ es el término, con $a_1, \ldots, a_k \in \mathbb{N}$.

Como se observa existe toda una serie de estructuras algebraicas que es necesario formalizar para llegar al concepto de polinomio. Se ha desarrollado en [6] una teoría computacional en ACL2 de los polinomios de múltiples variables sobre un cuerpo de coeficientes. Esta formalización incluye sus operaciones y propiedades fundamentales que establecen, principalmente, la existencia de la estructura de anillo. El objetivo aquí ha sido desarrollar una biblioteca reutilizable sobre polinomios con el que realizar ulteriores trabajos.

Pese a lo que pueda pensarse, ni siquiera la demostración de las propiedades básicas resulta trivial. No parece que esto se deba a la simplicidad de la lógica ACL2. En [10] se reconoce que desarrollar la asociatividad del producto en MIZAR supuso un reto. Se sorprenden sus autores de que en tratados de Álgebra se deje la demostración como ejercicio o se desarrolle sólo en el caso univariable y de manera incompleta, recurriendo a un razonamiento por analogía. De igual forma, el autor de [11] vio incrementado su esfuerzo por problemas técnicos surgidos durante la formalización de los polinomios en COQ. Otra formalización del anillo de polinomios se encuentra en [2].

Con respecto a la representación de los polinomios, se ha evolucionado hacia una representación dispersa, normalizada e uniforme. Es decir, una vez fijado el número de variables, se puede asociar una forma canónica a cada polinomio en la que los monomios están en orden estrictamente decreciente, no existe ninguno que sea nulo y todos contienen el mismo número de variables. La principal ventaja de esta representación se encuentra a la hora de decidir la igualdad.

Internamente se emplean listas de monomios, que a su vez son listas cuyo primer elemento es su coeficiente y cuyo resto es su término. Cada término se representa unívocamente mediante una lista de números naturales una vez que se han determinado dos cosas: el conjunto de variables y su orden.

El orden de monomios induce un orden entre polinomios, que también ha sido desarrollado en ACL2, véase [7]. Son imprescindibles aquí las propiedades de buena fundamentación de los órdenes. Por razones técnicas de la lógica de ACL2, su demostración se ha llevado a cabo mediante inmersión ordinal en ϵ_0.

4 Ideales polinómicos

La noción de ideal polinómico finitamente generados puede desarrollarse en ACL2 a partir de este marco. En lo sucesivo, sean p y q polinomios, y C y F listas de polinomios.

La pertenencia de p a $\langle F \rangle$ puede expresarse con el predicado $\exists C\ p = cl(C, F)$, donde cl es una función recursiva que calcula la combinación lineal de los elementos de F con coeficientes en C. Pero ACL2 no posee cuantificadores, así que

para ocultar el cuantificador introducimos una función de Skolem restringida por axiomas a devolver una lista de coeficientes testigo de la pertenencia al ideal.

```
(defun-sk en-ideal (p F)
  (exists (C)
    (and (polinomiosp C)
         (equal p (cl C F)))))
```

Internamente, este evento hace uso del principio de encapsulado para extender la lógica conservativamente con esta nueva función y su teoría asociada. Entonces es posible demostrar las propiedades fundamentales que debe satisfacer un ideal:

```
(defthm |p en <F> & q en <F> => p + q en <F>|
  (implies (and (polinomiop p) (polinomiop q) (polinomiosp F))
           (implies (and (en-ideal p F) (en-ideal q F))
                    (en-ideal (+ p q) F))))

(defthm |q en <F> => p * q en <F>|
  (implies (and (polinomiop p) (polinomiosp F))
           (implies (en-ideal q F)
                    (en-ideal (* p q) F))))
```

5 Reducciones polinómicas

Sea $<_M$ un orden bien fundamentado sobre los monomios y p un polinomio no nulo. El monomio principal de p con respecto a $<_M$ se denominará $mp(p)$. El coeficiente del monomio principal será $cp(p)$ y su término $tp(p)$. Para referirnos al resto del polinomio se empleará $resto(p)$.

Definición 4. *Sea f un polinomio no nulo. La relación de reducción $\rightarrow_f$ sobre polinomios inducida por f se define de manera que $p \rightarrow_f q$ si p contiene un monomio m, no nulo, tal que $\exists c\ m = -c \cdot mp(f)$ y $q = p + c \cdot f$.*

Definición 5. *Sea $F = \{f_1, \ldots, f_k\}$ un conjunto finito de polinomios no nulos. La relación de reducción inducida por F se define como $\rightarrow_F\ = \bigcup_{i=1}^{k} \rightarrow_{f_i}$*

A continuación se presenta la formalización de la reducción sobre polinomios dentro del marco suministrado por [8,9] donde se formalizan las reducciones abstractas como funciones binarias que a partir de un objeto de un cierto dominio, y de un *operador*, devuelven otro objeto del mismo dominio, llevando a cabo un *paso de reducción*. Un operador no puede ser aplicado a cualquier objeto, por lo que se introduce también un predicado binario que indicará cuándo es *válida* dicha aplicación.

Posteriormente, esto nos permitirá traspasar mediante instanciación funcional propiedades bien conocidas de las reducciones abstractas al caso particular de las reducciones polinómicas, sin necesidad de demostrarlas partiendo de cero.

Esta formalización requiere definir tres funciones. Un predicado unario que especifica el conjunto sobre el cual pretendemos que esté definida la reducción, en

nuestro caso particular, el de los polinomios. Una función binaria, `reduccion`, que representa la aplicación de un operador a un objeto, donde un operador será representado por una estructura que constan de los valores m, c y f de la definición 4. Y, por último, un predicado binario, `valido`, que comprobará si es válido aplicar un operador a un objeto.

Para que sea válido aplicar un operador a p, p debe ser un polinomio que contenga al monomio m, f debe ser un polinomio no nulo perteneciente a F y $c = -m/mp(f)$. Para esto último es necesario que $mp(f)$ divida a m.

El siguiente paso es definir la relación $\leftrightarrow_F^*$ inducida por $\rightarrow_F$. Debido a las limitaciones de la lógica de ACL2, se añade un parámetro extra a la función para evitar la aparición del cuantificador existencial. Este nuevo parámetro tiene como cometido almacenar la secuencia de pasos de reducción que se realizan.

Un paso de prueba se representa mediante una estructura con cuatro campos: un campo booleano que indica si el paso es directo o no, el operador que se aplica y los elementos que se conectan (`elt1`, `elt2`).

Una vez tenemos la función `paso-valido` que comprueba la validez de un paso de prueba, podemos definir qué entendemos por equivalencia entre dos polinomios uniformes respecto la reducción polinómica que hemos definido (el parámetro `prueba` contiene los pasos de prueba que demuestran que $p \leftrightarrow_F^* q$).

```
(defun <->* (p q prueba F)
  (if (endp prueba)
      (and (equal p q) (polinomiop p))
    (and (polinomiop p)
         (paso-valido (first prueba) F)
         (equal p (elt1 (first prueba)))
         (<->* (elt2 (first prueba)) q (rest prueba) F))))
```

Por último se aborda la demostración de la noetherianidad de la relación de reducción. En este caso $<$ representa el orden de polinomios y está probada su buena fundamentación. Así que esa relación se puede utilizar para enunciar la noetherianidad de la reducción polinómica previamente presentada.

```
(defthm |reduccion(p, o) < p|
  (implies (and (polinomiop p) (polinomiosp F))
           (implies (valido p o F)
                    (< (reduccion p o) p))))
```

En cuanto a la función de reducción polinómica, `red*`, que se utiliza en el algoritmo de Buchberger, se modela a partir de la clausura de la extensión a conjuntos de una función de reducción que recibe dos polinomios y devuelve el resultado de reducir su primer parámetro respecto del segundo.

Se demuestra la estabilidad del ideal respecto a la función de reducción, $q \in F \implies (p \in \langle F \rangle \iff red_F^*(p, q) \in \langle F \rangle)$

Por último, el siguiente teorema establece que la congruencia del ideal es subconjunto de la relación de equivalencia (en realidad son iguales, pero sólo necesitaremos la contención en un sentido).

Teorema 1. *Si* $F = \{f_1, \ldots, f_k\}$ *es un conjunto finito de polinomios e* $I = \langle F \rangle$ *entonces* $p \equiv_I q \implies p \leftrightarrow_F^* q$

```
(defthm |p =<F> q => p <->F* q|
  (implies (and (polinomiop p) (polinomiop q) (polinomiosp F))
           (implies (=<> p q F)
                    (<->* p q (prueba-<->* p q F) F))))
```

6 El algoritmo de Buchberger

Definición 6. *Sea* $F = \{f_1, \ldots, f_k\}$ *un conjunto finito de polinomios.* F *es una base de Gröbner de un ideal* I *si* $I = \langle F \rangle$ *y* $\rightarrow_F$ *es confluente.*

Definición 7. *Sean* $f_i = mp(f_i) + resto(f_i)$ *y* $f_j = mp(f_j) + resto(f_j)$, $m = \text{mcm}(mp(f_i), mp(f_j))$ *el mínimo común múltiplo de los monomios principales de* f_i *y* f_j, *y* m_i *y* m_j *monomios tales que* $m_i \cdot mp(f_i) = m = m_j \cdot mp(f_j)$. *El s-polinomio inducido por los polinomios* f_i *y* f_j *se define como:*

$$s\text{-}polinomio(f_i, f_j) = m_i f_i - m_j f_j$$

Teorema 2. *Sea* $F = \{f_1, \ldots, f_k\}$ *un conjunto finito de polinomios. Entonces* G *es una base de Gröbner de* F *si, y sólo si, todos los s-polinomios inducidos por los polinomios de* G *se reducen a* 0.

Para formalizar en ACL2 el algoritmo de Buchberger es necesario previamente definir varias funciones. La función `parejas` devuelve las parejas formadas por un elemento y todos los de otro conjunto; y la función `parejas-iniciales` devuelve todas las parejas que se pueden formar con los elementos de un conjunto.

Seguidamente se define la función principal que lleva a cabo todo el proceso de cálculo de una base de Gröbner a partir de una base inicial y que recibe como parámetros la base inicial `F`.

```
(defun Buchberger-aux (F C)
  (declare (xargs :measure (medida-Buchberger F C)))
  (if (and (polinomiosp F)
           (polinomiosp (aplana C)))
      (if (endp C)
          F
        (let* ((p (first (first C)))
               (q (second (first C)))
               (h (red* (s-polinomio p q) F)))
          (if (equal h (nulo))
              (Buchberger-aux F (rest C))
            (Buchberger-aux (cons h F)
                            (append (parejas h F) (rest C))))))
    F))

(defun Buchberger (F)
  (Buchberger-aux F (parejas-iniciales F)))
```

Obsérvese que se ha de suministrar una medida para poder demostrar la parada del algoritmo. Esta cuestión será estudiada a continuación.

6.1 Parada

Los argumentos de terminación en ACL2 se basan en la noción de estructura bien fundamentada. Los ordinales hasta ϵ_0 con su relación de orden habitual constituyen la única estructura bien fundamentada disponible primitivamente en el sistema y son necesarios para demostrar la terminación de las funciones introducidas.

Para abordar la terminación del algoritmo de Buchberger empleamos un orden lexicográfico sobre pares de listas de polinomios, ya que se puede observar que en la primera rama recursiva el primer parámetro permanece inalterado mientras el segundo parámetro decrece estructuralmente —se elimina un polinomio de la lista— y que en la otra rama, el primer parámetro decrece en un cierto sentido pese a que se le añade un nuevo polinomio. Esto es consecuencia del siguiente lema.

Lema 1 (de Dickson). *Dada una sucesión* $\{s_n\}$ *en* $\mathbb{N}^k$*, existen* i *y* j *tales que* $i < j \wedge s_i \leq_k s_j$ *donde* $\langle a_1, \ldots, a_k \rangle \leq_k \langle b_1, \ldots, b_k \rangle$ *si* $\forall 1 \leq i \leq k\ a_i \leq b_i$.

Consideremos la secuencia de términos líderes de los polinomios del parámetro y la relación de divisibilidad entre términos —los términos pueden verse como elementos de $\mathbb{N}^k$ donde $\leq_k$ juega el papel de la relación de divisibilidad— el lema anterior implica que no se pueden seguir añadiendo a la lista eternamente polinomios cuyos términos líderes no son divisibles por los anteriores.[2]

En este lema subyace un orden bien fundamentado sobre listas de polinomios. El problema está en construir ese orden, combinarlo con el orden estructural del otro parámetro mediante un producto lexicográfico, y demostrar mediante inmersión en los ϵ_0-ordinales que está bien fundamentado.

Es decir, dada una lista de términos que tiene la propiedad que ningún término divide a otro más allá en la secuencia, hay que construir una inmersión de la secuencia de términos en los ordinales, de manera que la secuencia de ordinales sea estrictamente decreciente. Ésta es una reformulación «finita» del lema de Dickson, pero es justo lo que se necesita si un determinado argumento de terminación confía en esta propiedad de las secuencias de términos. Y éste es el caso del algoritmo de Buchberger.

Siguiendo estas líneas se ha definido una medida para demostrar la terminación del algoritmo de Buchberger a partir de la que se emplea para demostrar el lema de Dickson en [5].

[2] Esto es así porque el polinomio que se añade a la base, h, no es nulo y es irreducible por F. En consecuencia, su término líder no es divisible por ninguno de los términos líderes de los polinomios de F.

6.2 Corrección parcial

Algunas de las ideas necesarias para llevar a cabo la demostración de corrección del algoritmo de Buchberger aparecen en [1,3,12], pero dado que una demostración completamente formal requiere un nivel de detalle mucho mayor del que aparece en un texto estándar, hemos establecido una serie de etapas, a modo de plan o estrategia de prueba, que nos conducirán al resultado final.

1. Demostrar la estabilidad del ideal respecto al algoritmo de Buchberger.

$$p \in \langle F \rangle \iff p \in \langle Buchberger(F) \rangle$$

2. Demostrar que la relación de reducción es normalizadora. Para ello definimos una función fn_F que calcula la forma normal o irreducible de un polinomio respecto a F y otra que construye una prueba que nos conduce desde un polinomio hasta su forma normal.
3. Demostrar que $fn_F(p) = red^*_F(p)$, y por lo tanto que $p \to^*_F red^*_F(p)$, donde red^*_F es la función de reducción que se utiliza en el algoritmo de Buchberger.
4. Demostrar que $\Phi(F)$ implica la confluencia local de la relación de reducción:

$$\Phi(F) \implies \forall p, q, r \ (p \to_F q \wedge p \to_F r \implies q \downarrow^*_F r)$$

   donde $\Phi(F)$ expresa el hecho de que los s-polinomios formados a partir de polinomios de una base F se reducen a 0.

$$\Phi(F) \equiv \forall p, q \in F \ s\text{-}polinomio(p, q) \to^*_F 0$$

   La confluencia local se reformula mediante los conceptos de prueba con forma de pico local y prueba con forma de valle. Una reducción es localmente confluente si, y sólo si, para toda prueba con forma de pico local existe una prueba equivalente con forma de valle. Para evitar la cuantificación existencial se definirá una función que construirá una prueba con forma de valle a partir de otra con forma de pico local.
5. Demostrar que si la relación de reducción inducida por F es localmente confluente y noetheriana entonces la relación de equivalencia inducida es decidible.

$$\forall p, q, r \ (p \to_F q \wedge p \to_F r \implies q \downarrow^*_F r) \wedge \ \to_F \text{ es noetheriana} \implies$$
$$\forall p, q \ (p \leftrightarrow^*_F q \iff fn_F(p) = fn_F(q))$$

   Esto se obtiene por instanciación funcional de los resultados presentados en [8,9] en el que se asume la existencia de una relación de reducción convergente[3] y se demuestra que su equivalencia inducida es decidible. Nuestra reducción es convergente, por tanto definimos un procedimiento de decisión para la equivalencia inducida, demostrando que es correcto y completo. Éste se limita a comprobar la igualdad de las formas irreducibles.

[3] Es decir, localmente confluente y noetheriana.

6. Demostrar que se cumple $\Phi(Buchberger(F))$.
7. Demostrar que la relación de reducción inducida por $G = Buchberger(F)$ es localmente confluente. Este resultado y el hecho de que sea normalizadora nos permiten demostrar que la equivalencia inducida es decidible por instanciación funcional.

Una vez llevado a cabo este plan, podemos obtener el resultado final demostrando el siguiente teorema que expresa el hecho de que el algoritmo de Buchberger devuelve una base de Gröbner.

Teorema 3. *Sea* $G = Buchberger(F)$, *entonces* $p \in \langle F\rangle \iff p \to_G^* 0$

Demostración. Demostremos primero que $p \in \langle G\rangle \implies p \to_G^* 0$.

$$\begin{aligned}
p \in \langle F\rangle &\implies p \in \langle G\rangle && \text{(ya que } p \in \langle F\rangle \iff p \in \langle G\rangle\text{)}\\
&\implies p \equiv_G 0 && \text{(por definición de } \equiv_G\text{)}\\
&\implies p \leftrightarrow_G^* 0 && \text{(ya que } p \equiv_G q \implies p \leftrightarrow_G^* q\text{)}
\end{aligned}$$

Recordemos que se cumple $\Phi(G)$, que implica que $\to_G$ es localmente confluente. Además $\to$ es normalizadora, independientemente de G, por lo que sabemos que $\leftrightarrow_G^*$ es decidible: basta comparar las formas normales.

$$\begin{aligned}
p \leftrightarrow_G^* 0 &\implies fn_G(p) = fn_G(0) && \text{(por decidibilidad de } \leftrightarrow_G^*\text{)}\\
&\implies red_G^*(p) = red_G^*(0) && \text{(ya que } fn_G(p) = red_G^*(p)\text{)}\\
&\implies red_G^*(p) = 0 && \text{(por definición de } red_G^*\text{)}\\
&\implies p \to_G^* 0 && \text{(ya que } p \to_G^* red_G^*(p)\text{)}
\end{aligned}$$

A continuación, demostremos que $p \to_G^* 0 \implies p \in \langle F\rangle$.

$$\begin{aligned}
p \to_G^* 0 &\implies p \leftrightarrow_G^* 0 && \text{(ya que } \to^* \subseteq \leftrightarrow^*\text{)}\\
&\implies fn_G(p) = fn_G(0) && \text{(por decidibilidad de } \leftrightarrow_G^*\text{)}\\
&\implies red_G^*(p) = red_G^*(0) && \text{(ya que } fn_G(p) = red_G^*(p)\text{)}\\
&\implies red_G^*(p) = 0 && \text{(por definición de } red_G^*\text{)}\\
&\implies red_G^*(p) \in \langle G\rangle && \text{(ya que } 0 \in \langle G\rangle\text{)}\\
&\implies p \in \langle G\rangle && \text{(ya que } p \in \langle G\rangle \iff red_G^*(p) \in \langle G\rangle\text{)}\\
&\implies p \in \langle F\rangle && \text{(ya que } p \in \langle F\rangle \iff p \in \langle G\rangle\text{)}
\end{aligned}$$

Los teoremas ACL2 correspondientes serían los siguientes, donde *prueba-fn* es la función que construye una prueba que conduce desde un polinomio hasta su forma normal.

```
(defthm |G = Buchberger(F) & p en <F> => p ->*G 0|
  (let* ((G (Buchberger F)) (prueba (prueba-fn p G)))
    (implies (and (polinomiop p) (polinomiosp F))
             (implies (en-ideal p F) (->* p (nulo) prueba G)))))
```

```
(defthm |G = Buchberger(F) & p ->*G 0 => p en <F>|
  (implies (and (polinomiop p) (polinomiosp F))
           (implies (->* p (nulo) prueba (Buchberger F)) (en-ideal p F))))
```

7 Conclusiones y trabajo futuro

Hemos mostrado cómo es posible aunar programación y demostración en el desarrollo de algoritmos algebraicos en el sistema ACL2.

El objetivo final es obtener una implementación verificada del algoritmo de Buchberger en ACL2 siguiendo los pasos de la estrategia presentada. Queda por demostrar parte del cuarto apartado del plan —esta propiedad equivale, en este contexto, al teorema de pares críticos de Knuth y Bendix— y parte del teorema $\Phi(Buchberger(F))$ del apartado sexto.

En el enfoque seguido no nos limitamos a demostrar que el algoritmo de Buchberger devuelve una base de Gröbner sino que finalmente proporcionaremos un procedimiento de decisión verificado para el problema de la pertenencia al ideal exhibiendo las funciones computables necesarias para que se cumpla que $p \in \langle F \rangle \iff red^*_G(p) = 0$, siendo $G = Buchberger(F)$.

Por otro lado queremos destacar la característica de ejecutabilidad en ACL2 de los algoritmos obtenidos y la reutilización de trabajos previamente desarrollados en dicho sistema, que creemos muy positivo.

Referencias

1. Baader, F., Nipkow, T.: *Term Rewriting and All That.* Cambridge University Press (1998)
2. Barja, J. M., Pérez, G.: *Demostración en Implementaciones Concretas de Anillos de Polinomios.* RSMAE (1999)
3. Geddes, K.O.; Czapor, S. R. y Labahn, G.: *Algorithms for Computer Algebra.* Kluwer (1992)
4. Kaufmann, M. y Manolios, P., Moore, J S.: *Computer-Aided Reasoning: An Approach.* Kluwer Academic Publishers (2000)
5. Martín, F. J.: *Formalization and proof of Dickson's lemma in ACL2.* Universidad de Sevilla. Informe Técnico (2002)
6. Medina, I.; Alonso, J. A. y Palomo, F.: *Automatic Verification of Polynomial Rings Fundamental Properties in ACL2.* ACL2 Workshop 2000 Proceedings (2000)
7. Medina, I.; Palomo, F. y Alonso, J.A.: *Implementation in ACL2 of Well-Founded Polynomial Orderings.* ACL2 Workshop 2002 Proceedings (2002)
8. Ruiz, J. L.: *Una teoría computacional acerca de la lógica ecuacional.* Universidad de Sevilla. Tesis doctoral (2001)
9. Ruiz, J. L.; Alonso, J. A.; Hidalgo, M. J. y Martín, F. J.: *Formal proofs about rewriting using ACL2.* Annals of Mathematics and Artificial Intelligence **36** (2002)
10. Rudnicki, P.; Schwarzweller, C. y Trybulec, A.: *Commutative Algebra in the Mizar System.* J. Symbolic computation **32** (2001)
11. Théry, L.: *A Machine-Checked Implementation of Buchberger's Algorithm.* J. Automated Reasoning **26** (2001)
12. Winkler, F. *Polynomial Algorithms in Computer Algebra.* Springer-Verlag (1996)