Automatización de la Aritmética

J.A. Alonso, J.J. Arrabal, A. Fernández y M. J. Pérez
Facultad de Matemáticas – Universidad de Sevilla
E–mail: {jalonso,arrabal,alexf,marper}@obelix.cica.es

Knowledge representation and automated reasoning have endlessly been, along the last centuries, aims of man. From Logic and Computation, Automated Deduction (**A.D.**) emerges as a treat of reaching these aims. Between the achievements of **A.D.** excels the solving of some conjectures in simple mathematical theories. In Arithmetic, which plays a central role in Mathematics, appear old and interesting conjectures. It is shown in this paper a system which furnishes main provings in Arithmetic and that shows a way to try unsolved problems related to this field.

1 Introducción.

Hacia finales del siglo XVII, Leibniz formula dos cuestiones que sintetizan viejas aspiraciones del hombre:

- la necesidad de disponer de un lenguaje universal (*lingua characteristica*) en el que poder expresar cualquier idea; y
- la posibilidad de mecanizar cualquier tipo de razonamiento (*calculus ratiocinator*).

Con el nacimiento de la Lógica Matemática moderna, en 1847, con los trabajos de G. Boole y A. De Morgan, aparece un primer "rational calculus", en el sentido de Leibniz. Posteriormente, en 1869, S. Jevons construye una máquina para verificar identidades booleanas.

La publicación en 1879 de los *Begriffsschrift* por parte de G. Frege, marca el inicio de la formalización de la Lógica. Dicho texto se considera precursor no sólo de los sistemas actuales de Lógica Matemática sino también de todos los lenguajes formales. Paradójicamente, a pesar de su importancia, estos trabajos pasaron desapercibidos.

Independientemente, G. Peano publica en 1889 sus *Arithmetices Principia* en el que utiliza una nueva notación simbólica para formalizar la teoría de números que R. Dedekind presentó el año anterior. La Aritmética de Peano (**PA**) es una teoría de primer orden que recoge las propiedades de los números naturales.

A principios del presente siglo, D. Hilbert propone usar las ideas de Frege y Peano para fundamentar las Matemáticas. En 1910, B. Russell y A. Whitehead publican el primer volumen de sus *Principia Mathematica* en la línea propuesta por Hilbert.

La primera cuestión formulada por Leibniz, la necesidad de construir un lenguaje universal, empezaba a tener respuestas parcialmente satisfactorias con los lenguajes formales. Hacia finales de los cincuenta, con la irrupción de los ordenadores electrónicos y con el soporte de los muchos logros obtenidos en el campo de la Lógica Matemática, se retoma con fuerza el segundo objetivo de Leibniz: la automatización del razonamiento, que se convierte en uno de los problemas iniciales de la Inteligencia Artificial.

En 1957, A. Newell, J.C. Shaw y H.A. Simon presentan su "máquina lógica" que consiste en un programa de ordenador capaz de probar 38 teoremas de la Lógica proposicional de los Principia Mathematica (tres años antes, M. Davis había escrito un programa para demostrar teoremas de la Aritmética de Presburger que no fue publicado en su momento). En 1960, Hao Wang con sus programas consiguió demostrar 9 capítulos de los Principia Mathematica en 9 minutos, incluyendo teoremas del cálculo de predicados usando el sistema deductivo de Gentzen.

En el mismo año, Gilmore presenta otra forma de tratar el problema de la deducción en la Lógica de primer orden basándose en un resultado de Skolem y Herbrand que reduce la demostración de una fórmula de primer orden, F, a la de una fórmula proposicional, F', asociada a ella. En esta reducción aparecen dos problemas: (1) La búsqueda de un "candidato" proposicional, F', y (2) la demostración de F'.

El artículo "*A computing procedure for quantification theory*", publicado por M. Davis y H. Putnam en 1960, supuso un avance cualitativo en la solución del problema (2) al introducir el concepto de cláusula y reducir la demostración de primer orden al estudio de la consistencia de conjuntos finitos de cláusulas proposicionales, proporcionando reglas que simplifican dicho estudio.

Los intentos de obtener una solución satisfactoria al problema (1) hacen emerger el concepto de unificación, que subyace en los trabajos de Prawitz, y culmina en el trabajo "*A machine oriented logic based on the resolution principle*" de J.A. Robinson publicado en 1965. En el mismo, presenta el principio de resolución en el que se combina la regla de corte proposicional y la unificación. Este trabajo supuso un hito en la demostración automática de teoremas y, en cierto modo, puede considerarse como el inicio de la deducción automática.

Desde entonces se han introducido diversos refinamientos del principio de resolución encaminados a mejorar la potencia del mismo en ciertos casos, y se han proporcionado estrategias para orientar el proceso y acotar el espacio de búsqueda. Entre los refinamientos y estrategias, destacamos los siguientes:

- **Estrategia del conjunto soporte** (L. Wos, D. Carson y G. Robinson, 1965).
- **Hiper–resolución** (J.A. Robinson, 1965).
- **Demodulación** (L. Wos, G. Robinson, D. Carson y L. Shalla, 1967).

– **Paramodulación** (G. Robinson y L. Wos, 1969).

– **Resolución UR** y **estrategia de pesos** (J. McCharen, R. Overbeek y L. Wos, 1976).

2 Representación de la Aritmética.

2.1 El sistema de Peano.

El lenguaje de la Aritmética, $\mathbf{L}$, es un lenguaje de primer orden con igualdad que consta de los siguientes símbolos no lógicos: un símbolo de constante ($\mathbf{0}$), uno de función 1–aria ($\mathbf{S}$), dos símbolos de funciones binarias ($+$, $\cdot$) y un predicado binario ($<$).

La Aritmética de Peano, **PA**, es la teoría de primer orden sobre $\mathbf{L}$ cuyos axiomas no lógicos son, básicamente, de dos tipos: los que corresponden a la parte no negativa de los anillos ordenados discretos, $\mathbf{P}^-$, y un esquema de axiomas de inducción.

Concretamente, los axiomas no lógicos son

(PA1): La función sucesor es inyectiva.

$\forall x \forall y(\mathbf{S}x = \mathbf{S}y \to x = y)$

(PA2): El $\mathbf{0}$ no es sucesor.

$\forall x(\mathbf{S}x \neq 0)$

(PA3) y (PA4): Definición recursiva de la suma.

$\forall x(0 + x = 0)$

$\forall x \forall y(\mathbf{S}x + y = \mathbf{S}(x + y))$

(PA5) y (PA6): Definición recursiva del producto.

$\forall x(0 \cdot x = 0)$

$\forall x \forall y(\mathbf{S}x \cdot y = y + x \cdot y)$

(PA7): Esquema de axiomas de inducción: para cada fórmula $\varphi(x)$,

$\varphi(0) \wedge \forall x(\varphi(x) \to \varphi(\mathbf{S}x)) \to \forall x \varphi(x)$

2.2 El sistema de Peano–Quaife.

A. Quaife, en su tesis doctoral de 1990, propuso una nueva representación del sistema de Peano con objeto de su automatización. En adelante, nos referiremos al mismo como la Aritmética de Peano–Quaife, **PQA**.

El lenguaje de la teoría **PQA** es un lenguaje clausal de primer orden cuyos símbolos no lógicos son: dos símbolos de constantes (`0`,`1`) y dos símbolos de funciones binarias (`+` y `*`).

Si $A_1, \ldots, A_n, B_1, \ldots, B_m$ son fórmulas atómicas, entonces la expresión

$$A_1, \ldots, A_n \to B_1, \ldots, B_m$$

es una cláusula que se corresponde con el cierre universal de la fórmula

$$A_1 \wedge \ldots \wedge A_n \to B_1 \vee \ldots \vee B_m$$

En el caso $n = m = 0$, se obtiene la cláusula vacía ($\to$) que se interpreta como **falso**.

La Aritmética de Peano–Quaife es una teoría de primer orden con igualdad sobre el lenguaje descrito anteriormente cuyos axiomas son:

(PQA1): La función sucesor es inyectiva.

```
1 + x = 1 + y -> x = y.
```

(PQA2): El `0` no es sucesor.

```
x = 0 ->.
```

(PQA3) y (QPA4): Definición recursiva de la suma.

```
-> 0 + x = x.
```

```
-> (1 + x) + y = 1 + (x + y).
```

(PQA5) y (PQA6): Definición recursiva del producto.

```
-> 0 * x = 0.
```

```
-> (1 + x) * y = y + (x * y).
```

(PQA7): Esquema de axiomas de inducción: para cada fórmula `F(x)`,

```
F(0) -> F(a).
```

```
F(0), F(1 + a) -> F(x).
```

(PQA8): Reflexividad de la igualdad.

```
 -> x=x.
```

3 Razonamiento en la Aritmética.

Las reglas de inferencia del sistema **PQA** son las siguientes:

1. **Resolución binaria**: Dados los conjuntos finitos de fórmulas atómicas $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$, $\mathcal{D}$ y dos fórmulas atómicas B y C tales que existe un unificador, σ, de máxima generalidad de B y C, la cláusula $(\mathcal{A}, \mathcal{C} \to \mathcal{B}, \mathcal{D})\sigma$ es una resolvente binaria de las cláusulas $\mathcal{A} \to \mathcal{B}, B$ y $\mathcal{C}, C \to \mathcal{D}$.

2. **Resolución UR**: Dada la cláusula $A_1, \ldots, A_n \to B_1, \ldots, B_m$, las cláusulas unitarias positivas $\to C_1, \ldots, \to C_n$ y las cláusulas unitarias negativas $D_1 \to, \ldots, D_{m-1} \to$ tales que existe un unificador, σ, de máxima generalidad verificando $A_i\sigma = C_i\sigma$ $(1 \leq i \leq n)$ y $B_j\sigma = D_j\sigma$ $(1 \leq j \leq m-1)$, la cláusula unitaria positiva $\to B_m\sigma$ es una resolvente **UR** de las de partida. De manera análoga se define la resolvente **UR**

cuya conclusión sea de la forma $A_i\theta \rightarrow (1 \leq i \leq n)$ ó $\rightarrow B_j\rho$ $(1 \leq j \leq m-1)$, respectivamente.

3. **Hiper–resolución**: Dada la cláusula $A_1, \ldots, A_n \rightarrow \mathcal{B}$ y las cláusulas positivas $\rightarrow \mathcal{C}_i, C_i$ $(1 \leq i \leq n)$ tales que existe un unificador, σ, de máxima generalidad verificando $A_i\sigma = C_i\sigma$ $(1 \leq i \leq n)$, la cláusula positiva $\rightarrow (\mathcal{B}, \mathcal{C}_1, \ldots, \mathcal{C}_n)\sigma$ es una hiper–resolvente de las dadas.

4. **Demodulación**: Dada la igualdad $\rightarrow s = t$ y la cláusula $\mathcal{A}[r] \rightarrow \mathcal{B}$, donde $\mathcal{A}[r]$ representa un conjunto de fórmulas atómicas que contiene al término r en una posición determinada, tales que existe un unificador, σ, de máxima generalidad de r y s, diremos que la cláusula $(\mathcal{A}[t] \rightarrow \mathcal{B})\sigma$ se ha obtenido por demodulación a partir de las dadas.

5. **Paramodulación**: Dadas las cláusulas $\mathcal{A} \rightarrow s = t, \mathcal{B}$ y $\mathcal{C}[r] \rightarrow \mathcal{D}$ tales que existe un unificador, σ, de máxima generalidad de r y s, diremos que la cláusula $(\mathcal{A}, \mathcal{C}[t] \rightarrow \mathcal{B}, \mathcal{D})\sigma$ se ha obtenido por paramodulación a partir de las dadas. Análogamente, se definen los casos en los que la unificación se hace entre r y t, o bien cuando r ocurre en $\mathcal{D}$.

Dado un conjunto de cláusulas $\mathcal{C} = C_1, \ldots, C_n$, una **refutación** de $\mathcal{C}$ es una sucesión de cláusulas, en la que cada una es un elemento de $\mathcal{C}$ o se obtiene de los anteriores mediante una de las reglas de inferencias antes citadas y, además, el último elemento de la sucesión es la cláusula vacía.

En el procedimiento de Quaife no se usa resolución binaria. En primer lugar, se intenta obtener la prueba mediante resolución UR, paramodulación y demodulación. En caso de fallo, se aplica hiper–resolución.

Las principales estrategias del sistema **PQA** son las siguientes:

1. **Estrategia del conjunto soporte**: En esta estrategia se divide el conjunto, $\mathcal{C}$, de cláusulas en dos conjuntos disjuntos, $\mathcal{D}$ y $\mathcal{C} - \mathcal{D}$, de forma que $\mathcal{C} - \mathcal{D}$ es satisfacible. El conjunto $\mathcal{C} - \mathcal{D}$ se llama conjunto usable y $\mathcal{D}$ se denomina conjunto soporte. La estrategia sólo permite obtener resolventes en el caso en que, al menos una de las cláusulas sobre las que se resuelve, esté soportada por $\mathcal{D}$; es decir, que ella o bien algunos de sus antecesores pertenecen a $\mathcal{D}$.

   Quaife utiliza como conjunto soporte las cláusulas obtenidas a partir de la negación de la fórmula que se quiere demostrar.

2. **Pesos:** En el proceso de deducción, las cláusulas se eligen en función del peso asignado (de menor a mayor peso). Por defecto, el peso de una cláusula es la suma del número de variables, constantes, funciones y predicados que en ella aparecen. No obstante, dicho peso puede modificarse con el fin de orientar el proceso deductivo.

   Quaife propone asignar peso 1 a todos los términos y subtérminos cerrados que aparecen en las cláusulas del soporte.

El sistema de razonamiento en el que Quaife ha desarrollado demostraciones automáticas en la Aritmética es OTTER (**O**rganized **T**echniques for **T**heorem–proving and **E**ffective **R**esearch). Dicho programa es un demostrador de teoremas para la Lógica de primer orden con igualdad, similar al AURA y LMA/ITP, desarrollado por W. McCune en el Argonnne National Laboratory en 1988.

El bucle principal del sistema en el proceso de búsqueda de una refutación es el siguiente:

```
Mientras que el conjunto soporte sea no vacio y no contenga a la clausula
             vacia, hacer:
  1. Elegir la clausula del soporte que tiene menor peso.
  2. Mover la clausula elegida del soporte al usable.
  3. Inferir y procesar nuevas clausulas usando las reglas de inferencia
     disponibles; cada nueva clausula  debe obtenerse a partir de la clau-
     sula seleccionada y elementos del conjunto de usables; las nuevas
     clausulas que superen las pruebas de eliminacion se agregan al con-
     junto soporte.
```

4 Resultados obtenidos por Quaife.

En el desarrollo de la Aritmética **PQA**, a partir de los axiomas del sistema, Quaife va ampliando, justificadamente, la teoría introduciendo nuevos símbolos de funciones y de predicados que corresponden a las operaciones y relaciones usuales de la Aritmética. En este proceso conviene resaltar el tratamiento de las listas dentro de la Aritmética, que son necesarias para la obtención de algunos resultados fundamentales como el de la factorización única.

Entre los resultados obtenidos por Quaife destacamos los siguientes:

- Primer teorema de Euclides: si x es un número primo que divide a $y \cdot z$, entonces divide a y o divide a z.
- Teorema aritmético de Pitágoras: la raíz cuadrada de un número primo es irracional.
- Segundo teorema de Euclides: existen infinitos números primos.
- Teorema de la factorización en números primos.
- Teorema de Euler: si x e y son primos entre sí, entonces $x^{\varphi(y)} \equiv 1 (\mathrm{mod}\ y)$ (en donde $\varphi(y)$ es la función de Euler: el cardinal del conjunto de números menores o iguales que y primos con y).
- Teorema de Fermat: si y es primo, entonces $x^y \equiv x (\mathrm{mod}\ y)$.

Asímismo, establece la equivalencia entre las siguientes conjeturas:

- GOLDBACH: todo número par distinto de 0 es suma de dos primos.
- VINOGRADOV: todo número impar mayor que 5 es suma de tres primos.
- SIERPINSKI: todo número par mayor que 4 es suma de tres primos.
- QUAIFE: todo número mayor que 5 es suma de tres primos.

5 Teoremas de ilustración.

En esta última sección presentamos la demostración de dos teoremas que ilustran algunas de las técnicas y procedimientos anteriormente descritos.

5.1 Segundo teorema de Euclides.

Teorema: *Existen infinitos números primos.*

En primer lugar, damos una demostración natural con objeto de reproducirla automáticamente con OTTER.

> Hemos de probar que para cada número natural, x, existe un número primo y que es mayor que x. Lo haremos por reducción al absurdo.
>
> Supongamos que existe un número natural a tal que cualquier número primo es menor o igual que a. Sea x un número arbitrario. Puesto que el menor factor primo de $1 + x!$ es primo, debe ser menor o igual que a y, por tanto, divide al factorial de a. En particular, el menor factor primo de $1 + a!$ divide al factorial de a y, puesto que también divide a $1 + a!$, tiene que dividir a la diferencia y, por tanto, debe ser igual a 1. De donde se sigue que $a! = 0$. Lo que es una contradicción.

A continuación veamos cómo OTTER reproduce dicha prueba. Para ello, se necesita un fichero de entrada en el que se recoja las cláusulas correspondientes a las hipótesis del teorema así como las correspondientes a la negación de la tesis. Además, dicho fichero debe incorporar las reglas de inferencia y posibles estrategias a utilizar en la demostración.

En esta prueba concreta utilizamos los siguientes símbolos no lógicos con los significados que se indican:

`fact (x)`	factorial de x
`mfp (x)`	menor factor primo de x, si $x > 1$ y x en caso contrario
`PR (x)`	x es un número primo
`x < y`	x es menor que y
`DIV (x,y)`	x divide a y

Las hipótesis necesarias son las siguientes:

- Si un número divide a una suma y a uno de los sumandos, entonces divide al otro.
- El menor factor primo de un número divide a dicho número.
- Si el menor factor primo de un número es 1, entonces dicho número es 1.
- El menor factor primo de $1 + x$ es distinto de 0.
- El menor factor primo de $1 + x!$ es primo.
- Si $y \neq 0$ e $y \leq x$, entonces y divide a $x!$.
- El factorial de cualquier número es distinto de 0.

Las cláusulas correspondientes a estas fórmulas constituyen la lista de usables.

La tesis del teorema es $\forall x \exists y(x < y \wedge PR(y))$. Su negación es $\exists x \forall y \neg(x < y \wedge PR(y))$. Introduciendo la variable de Skolem `a`, la cláusula correspondiente se transforma en `a < y , PR(y) ->.` que se incluye en el conjunto soporte.

Como reglas de inferencia vamos a utilizar resolución UR y demodulación. Los demoduladores utilizados son:

- Transformar la suma $x + y = x$ en $y = 0$.
- Transformar expresiones del tipo `DIV(x,1)` en `x=1`.

Las cláusulas correspondientes a estas fórmulas constituyen la lista de demoduladores.

El contenido del fichero de entrada es el siguiente.

```
list(usable).
DIV(x, y + z), DIV(x, z) -> DIV(x, y).
-> DIV(mfp(x), x).
mfp(x) = 1 -> x = 1.
mfp(1 + x) = 0 ->.
-> PR(mfp(1 + fact(x))).
-> y = 0, x < y, DIV(y, fact(x)).
fact(x) = 0 ->.
end_of_list.

list(demodulators).
-> (x + y = x) = (y = 0).
-> DIV(x,1) = (x = 1).
end_of_list.

list(sos).
a < y , PR(y) ->.
end_of_list.
```

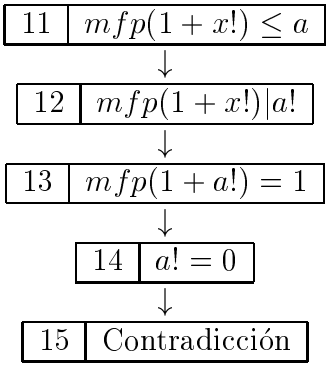
```
set(ur_res).
```

Aplicando OTTER a dicho fichero obtenemos la siguiente demostración:

```
----> UNIT CONFLICT at   0.08 sec ----> 15 [binary,14.1,7.1]  -> .

Length of proof is 4.  Level of proof is 4.

---------------- PROOF ----------------

1 [] DIV(x,y+z), DIV(x,z) -> DIV(x,y).
2 []  -> DIV(mfp(x),x).
3 [] mfp(x)=1 -> x=1.
4 [] mfp(1+x)=0 -> .
5 []  -> PR(mfp(1+fact(x))).
6 []  -> y=0, x<y, DIV(y,fact(x)).
7 [] fact(x)=0 -> .
8 []  -> (x+y=x)= (y=0).
9 []  -> DIV(x,1)= (x=1).
10 [] a<y, PR(y) -> .
11 [ur,10,5] a<mfp(1+fact(x)) -> .
12 [ur,11,6,4]  -> DIV(mfp(1+fact(x)),fact(a)).
13 [ur,12,1,2,demod,9]  -> mfp(1+fact(a))=1.
14 [ur,13,3,demod,8]  -> fact(a)=0.
15 [binary,14.1,7.1]  -> .

------------ end of proof -------------
```

En las dos primeras líneas del fichero de salida se indican el tiempo de ejecución (0.08 segundos), la longitud de la demostración (4) y su profundidad (4). Cada una de las cláusulas de la demostración va precedida de un número que se usa como referencia en los sucesivos pasos del proceso deductivo. A continuación, entre corchetes, se indica la regla de inferencia utilizada y las cláusulas a las que se aplica.

El grafo correspondiente a la demostración anterior es el siguiente:



5.2 Teorema aritmético de Pitágoras.

Teorema: *La raíz cuadrada de un número primo es irracional.*

Siguiendo el esquema del teorema anterior, comenzamos dando una demostración natural.

> Sea p un número primo y supongamos que su raíz cuadrada es un número racional, b/a. Entonces $p \cdot a^2 = b^2$. Dividiendo ambos miembros por el máximo común divisor de a y b se obtiene que $p \cdot a'^2 = b'^2$, donde $a' = a/mcd(a,b)$ y $b' = b/mcd(a,b)$. Por tanto, a'^2 divide a b'^2; luego b'^2 es divisible por a'. Teniendo presente que a' y b' son primos entre sí, resulta que b' es divisible por a'. Luego $a' = mcd(a',b') = 1$. Por otra parte, de la relación $p \cdot a'^2 = b'^2$ se deduce que si $p \cdot a'^2$ es primo, entonces b'^2 sería primo y, por tanto, $b' = 1$. Como hemos probado que $a' = 1$ y estamos suponiendo que p es primo, resulta que $p \cdot a'^2$ es primo. Luego, $b' = 1$. De donde se concluye que $p = 1$. Lo que es una contradicción.

A continuación veamos cómo OTTER reproduce dicha prueba. En este caso únicamente necesitamos añadir un nuevo símbolo no lógico a los usados en el teorema anterior: `mcd(x,y)` que se interpreta como el máximo común divisor de x e y, si $x \neq 0$ e y en caso contrario.

Las hipótesis necesarias son las siguientes:

- La divisibilidad es transitiva.
- Un número divide a cualquier otro en el que intervenga como factor.
- Si $x|y$, entonces el máximo común divisor de x e y es x.
- Si x e y son números distintos de cero, entonces los números $x/mcd(x,y)$ e $y/mcd(x,y)$ son primos entre sí.
- Si $u \cdot x^2 = y^2$, entonces $u \cdot (\frac{x}{mcd(x,y)})^2 = (\frac{y}{mcd(x,y)})^2$.
- Primer teorema de Euclides.
- 1 no es primo.
- Si el producto de dos números es primo, entonces alguno de ellos es 1.

Las cláusulas correspondientes a estas fórmulas constituyen la lista de usables.

La tesis del teorema es $\forall x(PR(x) \rightarrow \neg\exists y\exists z(mcd(y,z) \neq 0 \wedge x \cdot y^2 = z^2)$. Su negación es $\exists x(PR(x) \wedge \exists y\exists z(mcd(y,z) \neq 0 \wedge x \cdot y^2 = z^2)$. Introduciendo variables de Skolem, se obtienen las tres cláusulas que colocamos en el conjunto soporte.

Como reglas de inferencia vamos a utilizar resolución UR y paramodulación y demodulación. Los demoduladores utilizados son:

- Escribir los productos en orden lexicográfico.
- Transformar la expresión `1 * x` en `x`.

Las cláusulas correspondientes a estas fórmulas constituyen la lista de demoduladores.

Usaremos las estrategias de eliminación de unidades, demodulación hacia atrás y pesos. Además de los pesos correspondientes a los subtérminos del soporte ha sido necesario considerar los subtérminos de la expresión $p \cdot (\frac{a}{mcd(a,b)})^2 = (\frac{b}{mcd(a,b)})^2$.

El contenido del fichero de entrada es el siguiente.

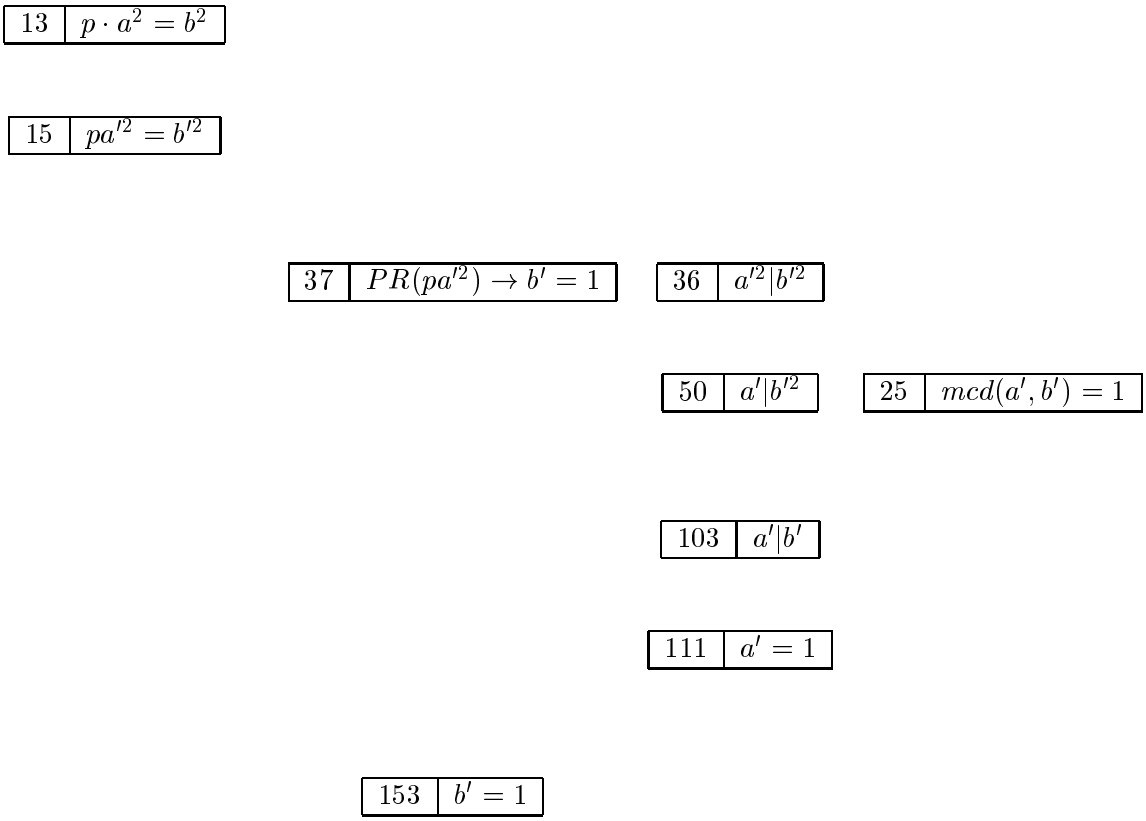
```
list(usable).
-> x = x.
DIV(x, y), DIV(y, z) -> DIV(x, z).
-> DIV(x, (y*x)).
DIV(x, y) -> mcd(x, y) = x.
-> mcd(x, y) = 0, mcd((x / mcd(x, y)), (y / mcd(x, y))) = 1.
u * (x * x) = y * y -> u * ((x / mcd(x, y)) * (x / mcd(x, y))) =
        (y / mcd(x, y)) * (y / mcd(x, y)).
mcd(x, y) = 1, DIV(x, (y * z)) -> DIV(x, z).
PR(1) ->.
PR(y * z) -> y = 1, z = 1.
end_of_list.

list(demodulators).
-> x * y = y * x.
-> 1 * x = x.
end_of_list.

list(sos).
-> PR(p).
-> p * (a * a) = b * b.
mcd(a, b) = 0 ->.
end_of_list.

set(ur_res).
set(para_into).
set(para_from).
set(back_demod).
set(unit_deletion).

weight_list(pick_and_purge).
weight(PR(p),1).
weight(p * (a * a),1).
weight(a * a,1).
weight(b * b,1).
weight(mcd(a, b),1).
```

```
weight(p * (a / mcd(a, b)) * a / mcd(a, b), 1).
weight((a / mcd(a, b)) * a / mcd(a, b), 1).
weight(a / mcd(a, b), 1).
weight((b / mcd(a, b)) * b / mcd(a, b), 1).
weight(b / mcd(a, b), 1).
end_of_list.
```

Aplicando OTTER a dicho fichero obtenemos la siguiente demostración:

```
----> UNIT CONFLICT at   1.41 sec ----> 198 [binary,197.1,8.1]  -> .

Length of proof is 10.  Level of proof is 8.

---------------- PROOF ----------------

2 [] DIV(x,y), DIV(y,z) -> DIV(x,z).
3 []  -> DIV(x,y*x).
4 [] DIV(x,y) -> mcd(x,y)=x.
5 []  -> mcd(x,y)=0, mcd(x/mcd(x,y),y/mcd(x,y))=1.
6 [] u*x*x=y*y -> u* (x/mcd(x,y))*x/mcd(x,y)= (y/mcd(x,y))*y/mcd(x,y).
7 [] mcd(x,y)=1, DIV(x,y*z) -> DIV(x,z).
8 [] PR(1) -> .
9 [] PR(y*z) -> y=1, z=1.
10 []  -> x*y=y*x.
11 []  -> 1*x=x.
12 []  -> PR(p).
13 []  -> p*a*a=b*b.
14 [] mcd(a,b)=0 -> .
15 [ur,13,6]  -> p* (a/mcd(a,b))*a/mcd(a,b)= (b/mcd(a,b))*b/mcd(a,b).
25 [ur,14,5]  -> mcd(a/mcd(a,b),b/mcd(a,b))=1.
36 [para_from,15.1.1,3.1.2]
   -> DIV((a/mcd(a,b))*a/mcd(a,b),(b/mcd(a,b))*b/mcd(a,b)).
37 [para_from,15.1.2,9.1.1] PR(p* (a/mcd(a,b))*a/mcd(a,b)) -> b/mcd(a,b)=1.
50 [ur,36,2,3]  -> DIV(a/mcd(a,b),(b/mcd(a,b))*b/mcd(a,b)).
103 [ur,25,7,50]  -> DIV(a/mcd(a,b),b/mcd(a,b)).
112,111 [para_into,25.1.1,4.2.1,unit_del,103]  -> a/mcd(a,b)=1.
154,153 [back_demod,37,demod,112,112,11,10,11,unit_del,12]  -> b/mcd(a,b)=1.
162 [back_demod,15,demod,112,112,11,10,11,154,154,11]  -> p=1.
197 [para_from,162.1.1,12.1.1]  -> PR(1).
198 [binary,197.1,8.1]  -> .

------------ end of proof -------------
```

El grafo correspondiente a la demostración anterior es el siguiente:



6 Conclusiones.

El sistema **PQA** ha proporcionado demostraciones de numerosos resultados de la Aritmética. Muchas de las pruebas son "equiparables" en tamaño y estructura a las que se encuentran en libros clásicos de Matemáticas.

Sin embargo, no ha conseguido resolver ninguna de las muchas conjeturas que existen sobre teoría de números. Existen monografías que recopilan problemas no resueltos de la Aritmética; por ejemplo, en la de Guy se encuentran más de 200 de tales problemas. Por otra parte, Quaife aporta una lista de 32 problemas abiertos formulados en el sistema

PQA.

No perdemos la esperanza de que mediante la deducción automática podamos resolver algunas conjeturas de la Aritmética, como ya ha ocurrido en otros campos matemáticos.

7 Bibliografía.

References

[1] Davis, M. *A computer program for Presburger's algorithm.* **Summer Inst. for Symbolic Logic**, Cornell U.V. (1957), 215–233.

[2] Davis, M. *The prehistory and early of automated deduction.* **Automation of Reasoning: Classical Papers on Computational Logic**, Vol 1. Ed. by J. Siekmann and G. Wrightson. Springer Verlag (1983), 1–28.

[3] Davis, M.; Putnam, H. *A computing procedure for quantification theory.* **Journal of the ACM**, 7, 3 (1960), 201–215.

[4] Gilmore, M. *A proof method for quantification theory, its justification and realization.* **IBM J. Res. Develop.** (1960), 28–35.

[5] Loveland, D.W.. *Automated theorem proving: a quarter century review.* **Contemporary Mathematics. Automated theorem proving: after 25 years**. American Mathematical Society, Vol 29 (1983), 1–46.

[6] McCharen, J.; Overbeek, R.; Wos, L *Complexity and related enhancement for automated theorem–proving program.* **Computer and Mathematics with applications**. (1976), 1–16.

[7] McCune, W. *OTTER User's guide".* **ANL** (1988).

[8] Newell, A.; Shaw, J.C.; Simon H.A. *Empirical explorations with the Logic Theory Machine: A case study in heuristics.* **Proc. Western Joint Computer Conf.** (1956), 218–239.

[9] Peano, G. **Aritmetices Principia, Nova Methodo Exposita**. Bocca (1889).

[10] Prawitz, D. *An improved proof procedure.* **Theoria** (1960), 102–139.

[11] Quaife, A. *Unsolved problems in elementary number theory.* **Journal Automated Reasoning**. (1991), 287–300.

[12] Quaife, A. **Automated development of fundamental Mathematical Theories**. Kluwer (1992).

[13] Robinson, G; Wos, L. *Paramodulation and theorem–proving in first–order theories with equality.* **Machine Intelligence** (B. Meltzer and D. Michie, eds.), Edinburg University Press (1969), 135–150.

[14] Robinson, J.A. *A machine oriented logic based on the Resolution Principle.* **Journal of the ACM** 12 (1965) 23–41.

[15] Robinson, J.A. *Automatic deduction with Hyper–Resolution.* **Journal of Computer Mathematics**, 1 (1965), 227–334.

[16] Wang, H. *Proving theorem by pattern recognition, Part I..* **Comm. ACM** (1960), 220–234.

[17] Whitehead, A.N.; Russell, B.. **Prinicipia Mathematica**. Vol 1. Cambridge University Press (1910).

[18] Wos, L.; Carson, D.; Robinson, G.. *Eficiency and completeness of the support strategy in theorem proving.* **Journal of the ACM** (1965), 536–541.

[19] Wos, L.; Henschen, L.. *Automated theorem proving 1965–1970.* **Automation of reasoning: classical papers on computational logic**, Vol 2. Ed. by J. Siekmann and G. Wrightson. Springer Verlag (1983), 1–24.

[20] Wos, L.; Robinson, G.; Carson, D.; Shala, L.. *The concept of demodulation in theorem proving.* **Journal of the ACM** (1967), 698–709.