Introducción a la demostración automática de teoremas

J.A. Alonso, J. Borrego, A. Chávez y F.J. Martín

Dpto. Ciencias de la Computación e Inteligencia Artificial

Universidad de Sevilla

¿Qué es el razonamiento automático?

El razonamiento automático se dedica a estudiar cómo usar un ordenador para ayudar en la parte de resolución de problemas que requiere razonamiento. Algunas cuestiones que surgen durante dicho estudio son la representación del conocimiento, las reglas para derivar nuevo conocimiento del que se tiene, y las estrategias para controlar dichas reglas. Otras cuestiones se refieren a la implementación de la teoría resultante y a las aplicaciones para las cuales el correspondiente software puede ser usado. Teoría, implementación y aplicaciones juegan papeles vitales para el razonamiento automático a la hora de intentar alcanzar uno de sus principales objetivos: proporcionar un asistente de razonamiento automático.

L. Wos: *What is Automated Reasoning?*
Journal of Automated Reasoning, Vol. 1.

¿Qué es el razonamiento automático?

El razonamiento automático se dedica al desarrollo de programas de ordenador que sean capaces de demostrar que una *conjetura* es una *consecuencia lógica* de un conjunto de *axiomas* o *hipótesis*.

- El lenguage en el que la conjetura, las hipótesis y los axiomas son escritos es una lógica, a menudo de primer orden, pero también puede ser no clásica o de orden superior.
- Las pruebas producidas por un sistema de razonamiento automático describen cómo y por qué la conjetura es una consecuencia de los axiomas y las hipótesis, utilizando para ello las reglas de derivación.

Utilización de un sistema de RA

- El usuario formaliza el problema y lo pasa al sistema de demostración automática como entrada
- El sistema intenta resolver el problema
- Si el sistema tiene éxito, se obtendrá una solución al problema
- Si el sistema no tiene éxito, entonces el usuario puede proporcionar cierta ayuda, intentar demostrar un resultado intermedio o revisar la formalización.

Algunos sistemas de RA

- Otter: http://www-unix.mcs.anl.gov/AR/otter/
- Mace: http://www-unix.mcs.anl.gov/AR/mace/
- EQP: http://www-unix.mcs.anl.gov/AR/eqp/
- Vampire: http://www.cs.man.ac.uk/ riazanoa/Vampire/
- Waldmeister: http://agent.informatik.uni-kl.de/waldmeister/
- Nqthm: http://www.cs.utexas.edu/users/boyer/ftp/nqthm/
- ACL2: http://www.cs.utexas.edu/users/moore/acl2/
- Coq: http://coq.inria.fr/
- HOL: http://www.cl.cam.ac.uk/Research/HVG/HOL/
- PVS: http://pvs.csl.sri.com/
- Nuprl: http://www.cs.cornell.edu/Info/Projects/NuPrl/

Campos de aplicación

- Matemáticas
  - “Problema de Robbins” - EQP
  - Problemas de quasigrupos - OTTER
- Síntesis de programas
- Verificación de software
- Verificación de hardware
  - FM9001 - NQTHM
  - AMD5K86 - ACL2
- Web semántica

Ejemplos de problema

- Rompecabezas:
  - Tres misioneros y tres caníbales están en la orilla derecha de un río. Hay una barca que puede transportar sólo dos personas. Si el número de misioneros nunca pude ser inferior al de caníbales en ninguna de las orillas, ¿cómo puden cruzar todos a la orilla izquierda?. (La barca no puede cruzar vacía).
- Diseño de circuitos
  - Usando cualquier número de puertas AND y OR, pero no más de dos puertas NOT, construir un circuito de acuerdo con la siguiente especificación: Hay tres entradas (e_1, e_2 y e_3) y tres salidas (s_1, s_2 y s_3) de forma que se cumpla $s_i = \texttt{not}(e_i)$.
- Problemas matemáticos
  - Sea G un grupo y e su elemento neutro. Demostrar que si, para todo x de G, $x^2 = e$, entonces G es conmutativo.

Problema elemental de grupos

Sea G un grupo y e su elemento neutro. Demostrar que si, para todo x de G, $x^2 = e$, entonces G es conmutativo.

- Axiomas de grupo:
  - $(\forall x)[e.x = x]$
  - $(\forall x)[x.e = x]$
  - $(\forall x)[x^{-1}.x = e]$
  - $(\forall x)[x.x^{-1} = e]$
  - $(\forall x)(\forall y)(\forall z)[(x.y).z = x.(y.z)]$
- Hipótesis:
  - $(\forall x)[x.x = e]$
- Conclusión:
  - $(\forall x)(\forall y)[x.y = y.x]$

Formalización en OTTER

”grupos1.in”

```
1  op(400, xfy, *).
2  op(300, yf, ^).
3
4  set(auto2).
5
6  list(usable).
7  e * x = x.                          % Ax. 1
8  x * e = x.                          % Ax. 2
9  x^ * x = e.                         % Ax. 3
10 x * x^ = e.                         % Ax. 4
11 (x * y) * z = x * (y * z).          % Ax. 5
12 x = x.                              % Ax. 6
13 x * x = e.
14 b * a != a * b.
15 end_of_list.
```

Ejecución en OTTER

- Ejecución: `otter < grupos1.in > grupos1.out`

”Prueba obtenida”

```
----> UNIT CONFLICT at  0.01 sec ----> 38 [binary,37.1,1.1] $F.

---------------- PROOF ----------------------------------
1 []                                          b*a!=a*b.
3,2 []                                        e*x=x.
5,4 []                                        x*e=x.
10 []                                         (x*y)*z=x*(y*z).
13 []                                         x*x=e.
17 [para_into,10.1.1.1,13.1.1,demod,3,flip.1] x*(x*y)=y.
23 [para_into,10.1.1,13.1.1,flip.1]           x*(y*(x*y))=e.
33 [para_from,23.1.1,17.1.1.2,demod,5,flip.1] x*(y*x)=y.
37 [para_from,33.1.1,17.1.1.2]                x*y=y*x.
38 [binary,37.1,1.1]                          $F.
------------ end of proof ----------------------------------
```

Problema elemental de búsqueda de modelos

Buscar un grupo no conmutativo.

- Axiomas de grupo:
  - $(\forall x)[e.x = x]$
  - $(\forall x)[x.e = x]$
  - $(\forall x)[x^{-1}.x = e]$
  - $(\forall x)[x.x^{-1} = e]$
  - $(\forall x)(\forall y)(\forall z)[(x.y).z = x.(y.z)]$
- Conclusión:
  - $b.a \neq a.b$

Formalización en MACE

”grupos2.in”

```
1  op(400, xfy, *).
2  op(300, yf, ^).
3
4  set(auto2).
5
6  list(usable).
7  e * x = x.                      % Ax. 1
8  x * e = x.                      % Ax. 2
9  x^ * x = e.                     % Ax. 3
10 x * x^ = e.                     % Ax. 4
11 (x * y) * z = x * (y * z).      % Ax. 5
12 b * a != a * b.
13 end_of_list.
```

Ejecución en MACE

- Ejecución: `mace2 -n2 -N6 -p < grupos2.in > grupos2.out`

”Prueba obtenida”

```
========================== Model #1 at 0.01 seconds:

 e: 0                     |    ^ :
                          |         0 1 2 3 4 5
 * :                      |       ---------------
      | 0 1 2 3 4 5       |         0 1 2 4 3 5
    --+------------       |
    0 | 0 1 2 3 4 5       |   a: 1
    1 | 1 0 3 2 5 4       |
    2 | 2 4 0 5 1 3       |   b: 2
    3 | 3 5 1 4 0 2       |
    4 | 4 2 5 0 3 1       |
    5 | 5 3 4 1 2 0       |
```

Bibliografía

- M. Davis. *The early history of automated deduction* En *Handbook of automated reasoning*
  (`http://www.cs.nyu.edu/cs/faculty/davism/early.ps`)
- G. Kolata. *Computer math proof shows reasoning power* (The New York Times, 10 de diciembre de 1996)
  (`http://www.nytimes.com/library/cyber/week/1210math.html`)
- G. Sutcliffe. *What is automated theorem proving?*
  (`http://www.cs.miami.edu/~tptp/OverviewOfATP.html`)