Environments and Tools to Facilitate Proving Hard Conjectures

Robert Veroff
Computer Science Department
University of New Mexico

AWARD-2004 Workshop
August 6, 2004

Perspectives

- Automatic deduction: If I am as successful at proving theorems in skew Boolean logic as Matthew, then something is *right*.
- Automated deduction: If I am as successful at proving theorems in skew Boolean logic as Matthew, then something is *wrong*.

Let's focus for the moment not on the capabilities of a theorem proving program but on the more general problem of providing an environment for proving hard conjectures.

Issues

- Accessibility and usability (including interface)
  - mathematician friendly
- Support tools
- Mathematician’s project notebook

Where is the (human) time spent?

- Mathematical knowledge and insight
- Ingenuity and creativity
- Determination and tenacity

One objective is to minimize time spent on the last, so we can concentrate on the first.

The middle is just the current reality of today's systems ...

A Painful Example
Skew Boolean Logic

```
x => (y => x)                                       (A1)
(x => (y => z)) => ((x => y) => (x => z))           (A2)
((x => y) => x) => x                                (A3)
x => (y -> x)                                       (A4)
(x -> (y -> z)) => ((x -> y) -> (x -> z))           (A5)
(x -> (y -> z)) => (y -> (x -> z))                  (A6)
((x -> y) -> x) => x                                (A7)
((x => y) -> y) => ((y => x) -> x)                  (A8)
(x => y) -> (x -> y)                                (A9)

x -> (x v y)                                        (A11)
y => (x v y)                                        (A12)
(x -> z) -> ((y -> z) -> ((x v y) -> z))            (A13)
(x ^ y) => x                                        (A14)
(x ^ y) -> y                                        (A15)
(x => y) => ((x => z) => (x => (y ^ z)))            (A16)
```

Challenge Theorems

```
(x => y) -> ((x v z) => (y v z))                (Step 8a)
(x => y) -> ((z v x) => (z v y))                (Step 8b)
(x => y) -> ((y => x) -> ((x ^ z) => (y ^ z)))  (Step 8c)
(x => y) -> ((z ^ x) => (z ^ y))                (Step 8d)
```

Approach

Take a very deep breath ...

- Prove equational form
  - theorem T becomes equation T = 1
  - additional axioms:

```
x => x = 1
x != 1 | x -> y != 1 | y = 1
x => y != 1 | y => x != 1 | x = y
```

  - high-level sketch provided by Matthew (hence “Step 8”)
  - typical application of proof sketches, not especially difficult

Approach (continued)

- Prove equational form, but with resolution and equality axioms

  ```
  x != y | x -> z = y -> z.
  ```

  Should be a “just do it” ...

- Translate problem and all accumulated hints to P/CD form

  t = 1 goes to P(t)

  t1 = t2 goes to

  ```
  P(t1 => t2)
  P(t1 -> t2)
  ```

Approach (continued)

- Prove hybrid form that still includes equality axioms and a substitution rule for predicate P

  ```
  x != y | -P(x) | P(y)
  ```

  issue with constant 1 (two versions)

- Eliminate equality, but including P-form analogs

  ```
  -P(x => y) | -P(y => x) | -P(x -> z) | -P(y -> z).
  ```

  note this is first-level only

- Eliminate P-form analogs of equality

  Mostly a typical application of proof sketches ...

  ```
  -P(x => y) | -P(y => z) | P(x => z).
  ```

Observations

- Mind bogglingly tedious with isolated spurts of creativity.
- Editor macros, shell scripts, and Otter mods made a dramatic difference.
- Automate!!!

Discussion: What works for you?