Automated Deduction with Otter and MACE

William McCune

Mathematics and Computer Science Division
Argonne National Laboratory

October 1995

Outline

- Application
- How the programs work
- How to use the programs
- Some results

Application

- Both programs apply to first-order predicate logic with equality.
  - Otter searches for proofs.
  - MACE (Models And Counter-Examples) searches for finite models.
- Practical application
  - Otter – simple theories, equational theories, Horn theories.
  - MACE – simple formulas, small models.

Otter Example

A simple theorem: $xx = e$ groups are commutative.

Otter input:

```
----------------------------------------------
  op(400, xfx, *).
  set(auto). clear(detailed_history).

  list(usable).
  e * x = x.          % A basis for group theory
  i(x) * x = e.
  (x * y) * z = x * (y * z).
  x * x = e.          % Hypothesis
  B * A != A * B.     % Denial of commutativity
  end_of_list.
----------------------------------------------
```

Otter proof:

```
  ---- PROOF at   0.11 sec -----

  1 [] B*A!=A*B.
  3,2 [] e*x=x.
  6 [] (x*y)*z=x* (y*z).
  8 [] x*x=e.
  11 [para_into,6,8,demod,3,flip.1] x* (x*y)=y.
  15 [para_into,6,8,flip.1] x* (y* (x*y))=e.
  18,17 [para_into,11,8] x*e=x.
  25 [para_from,15,11,demod,18,flip.1] x* (y*x)=y.
  29 [para_from,25,11] x*y=y*x.
  30 [binary,29,1] $F.
```

MACE Example

Find a noncommutative group.

MACE input:

```
----------------------------------------------
  op(400, xfx, *).

  list(usable).
  0 * x = x.          % A basis for group theory
  i(x) * x = 0.
  (x * y) * z = x * (y * z).

  1 * 2 != 2 * 1.     % Denial of commutativity
  end_of_list.
----------------------------------------------
```

MACE output:

```
  ================= Model #1 at 7.43 seconds:

  * :                    i :
      | 0 1 2 3 4 5        0 1 2 3 4 5
    --+------------      ---------------
    0 | 0 1 2 3 4 5        0 1 2 4 3 5
    1 | 1 0 3 2 5 4
    2 | 2 4 0 5 1 3
    3 | 3 5 1 4 0 2
    4 | 4 2 5 0 3 1
    5 | 5 3 4 1 2 0
```


How Otter Works

- The program is not interactive—the user supplies a conjecture and a search strategy.
- All proofs are by contradiction.
- Input statements are made into *clauses*.
  - Conjunctive normal form.
  - Replace existentially quantified variables with *Skolem functions*.
- The search is driven by a simple loop.
  - Apply inference rules selected by the user.
  - Simplify derived clauses and test for subsumption.
  - Apply search strategies specified by the user.
  - Stop when a contradiction is found or when resources are exhausted.

Otter's Inference Rules

- Paramodulation for equations:

$$\frac{\alpha_1 = \beta, \quad \gamma[\alpha_2] = \delta \quad \text{(with } \alpha_1\sigma \equiv \alpha_2\sigma\text{)}}{(\gamma[\beta] = \delta)\sigma}$$

  σ is a substitution of terms for variables that makes α_1 identical to α_2.

- Example:

$$\frac{\boxed{x' * x} = e, \quad (\boxed{y * z'}) * z = y}{e * z = z''}$$

  The terms $x * x'$ and $y * z'$ are *unified*, instantiating y to z''.

- Hyperresolution for implications:

$$\alpha_1 = \beta_1, \ \cdots, \ \alpha_n = \beta_n \ \rightarrow \ \gamma = \delta$$

- Other inference rules.
- All instantiation of variables happens by unification during application of inference rules.

Strategies for Otter

- Where to start the search.
- Restrict application of inference rules.
- Selecting the next clause for application of inference rules.
  - Use the smallest available clause (best first).
  - Use a queue (breadth first).
  - A combination of these (`pick_given_ratio`).
- Processing derived clauses.
  - Simplify with rewrite rules (demodulation).
  - Orient equations.
  - Discard subsumed clauses.
  - Discard “bad” clauses.

Knuth-Bendix Searches in Otter

- Knuth-Bendix completion
  - Start with a set of equations Σ.
  - Apply paramodulation (superposition) and simplification in a particular way, and make all derived equations, if possible, into rewrite rules.
  - If the procedure terminates without failing, the result is a terminating set of rewrite rules R (a *complete set of reductions*) that produces canonical forms.
  - Rewriting (in any way) with R is a decision procedure for the word problem in the theory Σ.
- This is easy to do with Otter.
- But a complete set of reductions does not exist for most interesting theories.
- However, it is a very useful search strategy.
  - If an equation can be used to simplify terms with respect to some term ordering $\succ$, apply it wherever possible.

How MACE Works

- The user doesn't need to know.
- For the curious:
  - Input statements are made into clauses.

$$(x * y) * z = x * (y * z)$$

  - The clauses are made into a relational form.

$$x * y = u,\ y * z = v,\ u * z = w \ \rightarrow\ x * v = w$$
$$x * y = u,\ y * z = v,\ x * v = w \ \rightarrow\ u * z = w$$

  - A domain size n is given, and all instances over the domain are constructed.

$$2 \times 6^6 = 93312 \text{ instances for } n{=}6.$$

  - Clauses about functions are included.
  - Now we have a propositional satisfiability problem.
  - Apply a Davis-Putnam procedure.
  - Propositional models are transformed into first-order models.

How to Use Otter

- Automatic mode — the user supplies a statement of the conjecture and Otter does the rest. Examples:

  Group theory, $x^3 = e \ \Rightarrow\ [[x,y],y] = e$ (1 second).

  Ring theory, $xx = x \ \Rightarrow\ xy = yx$ (4 seconds).

  Lattice theory, distrib. $\Rightarrow$ dual(distrib.) (4 seconds).
- "Interactive" mode.
  - Interact with Otter by making a sequence of searches.
  - Run Otter; if it fails to find a proof, examine the output file, and adjust the search parameters; repeat.
- "Find-a-theorem" mode.
  - Usually for finding a basis with particular properties (single axiom, self-dual basis).
  - Generate a large set of candidate bases.
  - Automatically run a separate Otter search with each candidate.
  - Go away for the weekend.

Otter's "Interactive" Mode

- Weight limit of derived clauses.
  - If the limit is too high, Otter uses too much memory to store clauses.
  - If the limit is too low, Otter discards too many interesting clauses and runs out of work ("sos empty").
- Raising or lowering the priority of clauses. Examples:
  - Assign high priority to clauses containing Skolem constants
  - Assign low priority (or discard) clauses containing terms of the form $t_1 \wedge t_2 \wedge t_3 \wedge t_4$.
- Restricting application of inference rules. Example:
  - Do not apply paramodulation to clauses that are implications.
  - Do not apply paramodulation from the "heavy" side of the equation.
- Change representation.
- Include lemmas.
- Use a different basis for the theory.

Using Otter – Steiner Law

Uniqueness of the binary Steiner law on a cubic curve.

$$\left\{\begin{array}{l} f(x,y)=f(y,x) \\ f(x,f(y,x))=y \\ f(e,e)=e \\ g(x,y)=g(y,x) \\ g(x,g(y,x))=y \\ g(e,e)=e \end{array}\right\} =\!(gL)\!\Rightarrow\ \{f(x,y)=g(x,y)\}$$

($=\!(gL)\!\Rightarrow$ is an inference rule for algebraic geometry.)

- Our first proof took 38 hours and was found with a strategy that prefers clauses with variables.
- We were then interested in the analogous conjectures for 5-ary and for 8-ary operations, with full symmetry.
  - We studied the proof of the binary case and refined the search strategy, then we tried arities 3 and 4.
  - We separated two lemmas from the main proof.
  - Restricted use of symmetry.
  - Success for the 5-ary and 8-ary cases (with searches of a few minutes).
  - The strategy was further refined into an algorithm, which led us to a proof (by hand) for n-ary operations, $n \geq 2$.

Using Otter – CS Conjecture

$$\left\{\begin{array}{l} \text{cancellative semigroup} \\ xy^3=y^3x \end{array}\right\} \Rightarrow \{(xy)^9=x^9y^9\}$$

- True for groups (Narain Gupta, 1969).
- CS? Yes (Otter, 1994).
  Strategy: max_weight=37, ratio=3, cancellation as rewrite rules. Proof (7 hours) is bidirectional.
- Without cancellation?
  ($\{\mathsf{S}, xy^2=y^2x\} \Rightarrow \{(xy)^4=x^4y^4\}$ is true.)
  - restrict rewriting
  - adjust forward/backward balance
  - success (6 hours) – humanized proof:

$$\begin{array}{ll} xxxxxxxxxyyyyyyyyy & \\ xxxxxxyyyyyyyyxxxy & [xxx(yyyyyyyy)] \\ xxxxxxyyyyyyyyxxxyy & [yxxx] \\ xxxxxxyyyyxxyyyxyy & [yyy(xx)] \\ xxxxyyyxxyxxyyyxyy & [(xx)yyy] \\ xyyxxxyxxyxxyyyxyy & [xxx(yy)] \\ xyyxyyxyxxyxxyxxyy & [(xxy)(xxy)(xxy)(yyxy)] \\ xyyxyyxyxyxxyxxyxy & [(xyx)(xyx)(xyx)(xy)] \\ xyyxyxyxxyxyxyxyxy & [(yx)(yx)(yx)(xyxxyx)] \\ xyxyxyxyxyxyxyxyxy & [y(yx)(yx)(yx)] \end{array}$$

CS Conjecture (continued)

- change representation: $P((AB)^9)) \rightarrow P(A^9B^9)$.
  - adjust forward/backward balance
  - proof in a few minutes.
- $\{\mathsf{S}, xy^4=y^4x\} \Rightarrow \{(xy)^{16}=x^{16}y^{16}\}$??
  - failure
  - conjecture: *not a theorem*

The Robbins Problem

$$\begin{array}{lll} x+y=y+x & (C) & \\ (x+y)+z=x+(y+z) & (A) & \\ n(n(x)+y)+n(n(x)+n(y))=x & (H) & \text{Huntington} \\ n(n(x+n(y))+n(x+y))=x & (R) & \text{Robbins} \end{array}$$

- $\{C,A,H\}$ is a basis for Boolean algebra.
- Is $\{C,A,R\}$ basis for Boolean algebra? Still open.
- Theorem (Steve Winker):

$$\left\{\begin{array}{l} C,\ A,\ R \\ \exists x\exists y,\ x+y=x \end{array}\right\} \Rightarrow \{H\}.$$

The Robbins Problem (continued)

This is a difficult problem for Otter — 25 minutes with a specialized strategy:

```
assign(max_weight, 30).
clear(eq_units_both_ways).
set(dynamic_demod_lex_dep).
assign(max_distinct_vars, 3).
assign(pick_given_ratio, 3).
lex([A, B, c, d, n(x), +(x,x)]).

weight_list(purge_gen).
weight(+($(0),+($(0),+($(0),$(0)))), 500).
weight(n(n($(0))), 500).

weight(n(x+x) = $(0), 500).
weight(n(x+c) = $(0), 500).
weight(n(c+x) = $(0), 500).
weight(n(x+d) = $(0), 500).
weight(n(d+x) = $(0), 500).

weight($(0) = $(0)+  ($(0)+$(0)), 500).
weight($(0) = n($(0)+  ($(0)+$(0))), 500).
weight($(0) = n($(0)+ n($(0)+$(0))), 500).
weight($(0) = n(n($(0)+$(0))+$(0)), 500).
end_of_list.

weight_list(pick_given).
weight($(1) = $(2), 1).
end_of_list.
```


SAM's Lemma

Theorem. Let L be a modular lattice with 0 and 1. For all $x, y \in L$, if z_1 is a complement of $x \vee y$, and if z_2 is a complement of $x \wedge y$, then

$$(z_1 \vee (x \wedge z_2)) \wedge (z_1 \vee (y \wedge z_2)) = z_1.$$

- How to formulate the denial of the conclusion?

```
% C1 is a complement of A v B
C1 v (A v B) = 1.    C1 ^ (A v B) = 0.

% C2 is a complement of A ^ B
C2 v (A ^ B) = 1.    C2 ^ (A ^ B) = 0.

% denial 1:
(C1 v (A ^ C2)) ^ (C1 v (B ^ C2)) != C1.

% denial 2:
C1 v (A ^ C2) = D.
C1 v (B ^ C2) = E.
D ^ E != C1.

% denial 3 (this one works best):
A ^ C2 = R1.    C1 v R1 = D.
B ^ C2 = R2.    C1 v R2 = E.
D ^ E != C1.
```

- Proof in 18 seconds (constants assigned weight 0).

Theorem Finding With Otter

Theorem (Padmanabhan)

$$\left\{ \begin{array}{l} p(x,y,y) = y \\ p(y,x,y) = y \\ p(y,y,x) = y \\ f(y) = y \end{array} \right\} \Leftrightarrow p(p(x,y,y),u,p(p(x,y,y),f(y),z)) = y.$$

- Problem: find a single equation with *two* slots.
- Use Otter to generate many (7236) candidates.
- Run a separate Otter search with each candidate.
- Success:

Theorem (Otter)

$$\left\{ \begin{array}{l} p(x,y,y) = y \\ p(y,x,y) = y \\ p(y,y,x) = y \\ f(y) = y \\ g(y) = y \end{array} \right\} \Leftrightarrow p(p(x,y,y),p(x,p(y,z,f(y)),g(y)),u) = y.$$

Weaknesses

- Otter
  - Theories with many operations.
  - Defined operations.
  - Non-Horn theories, set theory.
  - Associative-commutative operations.
  - Proofs are longer than necessary.
- MACE
  - Complex statements.
  - Large models.
  - Too many isomorphic models.

Soundness of Otter Proofs

From a proof about distributivity in QLT:

427,426	$x \vee (y \wedge (((z \wedge y) \vee x) \wedge z)) = (z \wedge y) \vee x$	[26 → 340]
433,432	$((x \vee y) \wedge x) \vee y = x \vee y$	[9 → 340]
439	$(((x \wedge y) \vee x) \wedge y) \vee x = (x \vee y) \wedge x$	[186 :433]
476	$(x \wedge (y \wedge ((x \wedge y) \vee z))) \vee z = (x \wedge y) \vee z$	[5 → 342]
504	$x \vee ((y \wedge z) \vee ((x \vee (y \wedge z)) \wedge y)) = x \vee (y \wedge z)$	[14 → 52 :11,119]
509,508	$((x \vee y) \wedge z) \vee (z \wedge y) = z \wedge (y \vee x)$	[4 → 52]
522	$(x \wedge (y \vee (x \wedge z))) \vee (x \wedge z) = x \wedge ((x \wedge z) \vee y)$	[21 → 52]
563,562	$(x \vee y) \wedge (z \wedge (y \vee x)) = (x \vee y) \wedge z$	[354 → 44 :37, flip]
566	$x \wedge (y \vee z) = (z \vee y) \wedge ((y \vee z) \wedge x)$	[354 → 80]
567	$x \wedge (y \vee z) = (z \vee y) \wedge x$	[354 → 26 :563]
570,569	$(x \vee y) \wedge ((y \vee x) \wedge z) = (x \vee y) \wedge z$	[354 → 5, flip]
571	$(x \vee y) \wedge z = z \wedge (y \vee x)$	[flip 566 :570]
582	$x \wedge (y \wedge (z \vee u)) = (u \vee z) \wedge (x \wedge y)$	[359 → 26]
584	$x \wedge (y \wedge (z \vee u)) = y \wedge (x \wedge (u \vee z))$	[359 → 20]
592,591	$((x \vee y) \wedge y) \vee x = y \vee x$	[375 → 375 :570,341, flip]
618,617	$((x \vee y) \wedge y) \vee (y \vee x) = y \vee x$	[375 → 40 :592]
620,619	$x \vee (y \vee ((x \vee z) \wedge z)) = x \vee (y \vee z)$	[375 → 48 :47, flip]
632	$x \vee ((x \vee y) \wedge y) = x \vee y$	[375 → 28 :41, flip]
666	$x \vee (y \vee (y \wedge x)) = y \vee x$	[376 → 16 :355,11,41,618,3]
717	$x \vee (y \wedge (z \wedge (x \vee (y \wedge z)))) = x \vee (y \wedge z)$	[5 → 399]
767	$x \wedge (y \vee (z \vee u)) = (z \vee (y \vee u)) \wedge x$	[139 → 567 :11]
774	$(x \vee y) \wedge z = (y \vee x) \wedge z$	[4 → 567]
839	$(x \wedge y) \vee (y \wedge x) = y \wedge x$	[7 → 56 :8]
843	$x \vee ((y \wedge z) \vee ((x \vee y) \wedge x)) = x \vee (y \wedge z)$	[14 → 56 :11,123]
864,863	$(((x \wedge y) \vee z) \wedge x) \vee (x \wedge y) = x \wedge ((x \wedge y) \vee z)$	[21 → 56]
869	$(x \wedge y) \vee ((y \vee z) \wedge x) = x \wedge (y \vee z)$	[9 → 56]
966	$(x \wedge y) \vee (y \vee x) = x \vee y$	[33 → 666]
969,968	$x \vee ((x \wedge y) \vee y) = x \vee y$	[30 → 666]
1030,1029	$(x \wedge y) \vee (z \vee (y \wedge x)) = (x \wedge y) \vee z$	[839 → 46 :41, flip]
1033	$x \vee (y \wedge z) = (y \wedge z) \vee ((z \wedge y) \vee x)$	[839 → 139]
1036	$x \vee (y \wedge z) = (z \wedge y) \vee x$	[839 → 27 :1030]
1038,1037	$(x \wedge y) \vee ((y \wedge x) \vee z) = (y \wedge x) \vee z$	[839 → 10, flip]
1039	$(x \wedge y) \vee z = z \vee (y \wedge x)$	[flip 1033 :1038]

Soundness (continued)

- Otter proofs are difficult to for humans to check.
  - unnatural
  - usually much longer than necessary
- However, Otter can output proofs that are easy for computers to check.
  - An independent, very simple proof checking program exists.
  - Formal verification of the proof checker is feasible.

MACE Results – OMTS

A question from Frank Bennett: is there a pair of orthogonal Mendelsohn triple systems of order 9?

This corresponds to a model of the following clauses.

```
f(x,x) = x
h(x,x) = x
f(x,f(y,x))=y
h(x,h(y,x))=y
f(x,y)=u, f(z,w)=u, h(x,y)=v, h(z,w)=v -> x=z, y=w
```

MACE produced the following model.

```
Model #1 at 54.58 seconds:

 f: | 0 1 2 3 4 5 6 7 8      h: | 0 1 2 3 4 5 6 7 8
-----------------------     -----------------------
  0 | 0 8 5 2 7 4 3 6 1       0 | 0 2 6 4 3 1 8 5 7
  1 | 8 1 7 6 3 2 5 4 0       1 | 5 1 0 8 2 3 7 6 4
  2 | 3 5 2 8 6 0 4 1 7       2 | 1 4 2 6 7 8 0 3 5
  3 | 6 4 0 3 8 7 1 5 2       3 | 4 5 7 3 0 6 2 8 1
  4 | 5 7 6 1 4 8 2 0 3       4 | 3 8 1 0 4 7 5 2 6
  5 | 2 6 1 7 0 5 8 3 4       5 | 7 0 8 1 6 5 3 4 2
  6 | 7 3 4 0 2 1 6 8 5       6 | 2 7 3 5 8 4 6 1 0
  7 | 4 2 8 5 1 3 0 7 6       7 | 8 6 4 2 5 0 1 7 3
  8 | 1 0 3 4 5 6 7 2 8       8 | 6 3 5 7 1 2 4 0 8
```


MACE Results – TNL

Transitive Near Lattice (TNL)

$$
\begin{array}{ll}
x \wedge x = x, & x \vee x = x, \\
x \wedge y = y \wedge x, & x \vee y = y \vee x, \\
x \wedge (x \vee y) = x, & x \vee (x \wedge y) = x, \\
x \wedge (y \vee (x \vee z)) = x, & x \vee (y \wedge (x \wedge z)) = x.
\end{array}
$$

If $(\vee, \wedge)$ and $(\vee, *)$ are both transitive near lattices, then $x \wedge y = x * y$.

Counterexample:

```
v | 0 1 2 3 4
--+----------
0 | 0 3 0 3 0
1 | 3 1 1 3 1
2 | 0 1 2 3 4
3 | 3 3 3 3 3
4 | 0 1 4 3 4


^ | 0 1 2 3 4          * | 0 1 2 3 4
--+----------          --+----------
0 | 0 2 2 0 4          0 | 0 4 2 0 4
1 | 2 1 2 1 4          1 | 4 1 2 1 4
2 | 2 2 2 2 2          2 | 2 2 2 2 2
3 | 0 1 2 3 4          3 | 0 1 2 3 4
4 | 4 4 2 4 4          4 | 4 4 2 4 4
```

Network Addresses

- Supplements to McCune/Padmanabhan monograph *Automated Equational Deduction with Otter*:

  `http://www.mcs.anl.gov/home/mccune/ar/monograph/`

- Automated Deduction at Argonne:

  `http://www.mcs.anl.gov/home/mccune/ar/`

- Otter

  ```
  ftp://info.mcs.anl.gov/pub/Otter/otter.tar.Z   (unix)
  ftp://info.mcs.anl.gov/pub/Otter/MACOT304.HQX   (Mac)
  ftp://info.mcs.anl.gov/pub/Otter/DOSOT304.EXE   (DOS)
  ```

- MACE (UNIX):

  `ftp://info.mcs.anl.gov/pub/Otter/mace.tar.Z`

- Me:

  `mccune@mcs.anl.gov`

- These slides:

`ftp://info.mcs.anl.gov/pub/Otter/Slides/manitoba-95.ps.Z`