Automated Reasoning Systems

Finite Models and Counterexamples: Mace4

Temur Kutsia

RISC, Johannes Kepler University of Linz, Austria
kutsia@risc.uni-linz.ac.at

March 29, 2011

Mace4

- A program that searches finite models
- Author: Bill McCune
- Theory: First-order logic with equality
- Can serve as a complement of Prover9, looking for counterexamples before (or at the same time as) using Prover9 to search for a proof.
- Implemented in C.
- Web page: http://www.cs.unm.edu/~mccune/mace4/

What Mace Does

1. Mace4 produces interpretations which are models of the input formulas.
2. The interpretations have finite domain.
3. By default, Mace4 starts searching for a structure of domain size 2, and then it increments the size until it succeeds or reaches some limit.

How Mace Works

1. Fixed domain size, say n. The members of the domain are named $\{0, \ldots, n-1\}$.

How Mace Works

1. Fixed domain size, say n. The members of the domain are named $\{0, \ldots, n-1\}$.
2. Tables for the function and predicate symbols are set up.

How Mace Works

1. Fixed domain size, say n. The members of the domain are named $\{0, \ldots, n-1\}$.
2. Tables for the function and predicate symbols are set up.
3. All ground instances of the input clauses (over the domain) are generated.

How Mace Works

1. Fixed domain size, say n. The members of the domain are named $\{0, \ldots, n-1\}$.
2. Tables for the function and predicate symbols are set up.
3. All ground instances of the input clauses (over the domain) are generated.
4. In a systematic way, a recursive backtracking procedure fills in the cells of the tables and uses the ground clauses to propagate the effects of the assignments

How Mace Works

1. Fixed domain size, say n. The members of the domain are named $\{0, \ldots, n-1\}$.
2. Tables for the function and predicate symbols are set up.
3. All ground instances of the input clauses (over the domain) are generated.
4. In a systematic way, a recursive backtracking procedure fills in the cells of the tables and uses the ground clauses to propagate the effects of the assignments
5. When contradictions are encountered, backtracking occurs, the propagations and assignments are undone, and other assignments are attempted.

How Mace Works

1. Fixed domain size, say n. The members of the domain are named $\{0, \ldots, n-1\}$.
2. Tables for the function and predicate symbols are set up.
3. All ground instances of the input clauses (over the domain) are generated.
4. In a systematic way, a recursive backtracking procedure fills in the cells of the tables and uses the ground clauses to propagate the effects of the assignments
5. When contradictions are encountered, backtracking occurs, the propagations and assignments are undone, and other assignments are attempted.
6. If all the tables become full, with no contradictions, a model has been found.

How Mace Works

1. Fixed domain size, say n. The members of the domain are named $\{0, \ldots, n-1\}$.
2. Tables for the function and predicate symbols are set up.
3. All ground instances of the input clauses (over the domain) are generated.
4. In a systematic way, a recursive backtracking procedure fills in the cells of the tables and uses the ground clauses to propagate the effects of the assignments
5. When contradictions are encountered, backtracking occurs, the propagations and assignments are undone, and other assignments are attempted.
6. If all the tables become full, with no contradictions, a model has been found.
7. If Mace4 is iterating through domain sizes, this procedure applies, separately, to each domain size.

Running Mace4

1. Prepare the input file containing the logical specification of a conjecture and the search parameters.
2. Issue a command that runs Mace4 on the input file and produces an output file.
3. Look at the output.
4. If a model is found, be happy.
5. If no model found, try either to increase the domain size, or run Prover9 to find a contradiction.

How to Run Mace4

From the command line:

- Run: `mace4 -f inputfile`
  or `mace4 -f inputfile > outputfile`
  or `mace4 < inputfile`
  or `mace4 < inputfile > outputfile`
- If the input file contains commands that Mace4 does not understand (e.g., Prover9 commands), use the option `-c` to ignore them:
  `mace4 -c -f inputfile > outputfile`
- Help:
  `mace4 -help`

Using GUI.

Syntax

- The same as for Prover9.
- Mace preprocesses the input also in the same way as Prover9.

Syntax

Example (to compute a counterexample for the goal):

```
formulas(assumptions).
  E * x = x.
  x' * x = E.
  (x * y) * z = x * (y * z).
end_of_list.

formulas(goals).
 A * B = B * A.
end_of_list.
```

Syntax

Example (to compute a model of the input clauses):

```
clauses(sos).
  E * x = x.
  x' * x = E.
  (x * y) * z = x * (y * z).
  A * B != B * A.
end_of_list.
```