Computing Finite Models by Reduction to Function-Free Clause Logic

Peter Baumgartner[1]
Alexander Fuchs[2]
Cesare Tinelli[2]

[1]National ICT Australia (NICTA)

[2]The University of Iowa

University of Koblenz
June 08, 2006

Outline

Outline

Automated Theorem Proving

- Solving Problems
  Mathematical proofs, software and hardware verification, . . .
- Formulated in Logic
  Propositional logic, first-order logic, . . .
  Here: first-order logic with equality
- Automatically
  User interaction consists at best only of problem formulation.

Example Proving - Group Theory

- Do the following axioms specify a group?

$$
\begin{array}{llcll}
\forall x, y, z & : & (x * y) * z & = & x * (y * z) & (\text{associativity}) \\
\forall x & : & e * x & = & x & (\text{left} - \text{identity}) \\
\forall x & : & i(x) * x & = & e & (\text{left} - \text{inverse})
\end{array}
$$

- Hypothesis: Does right-identity hold?

$$
\forall x \quad : \quad x * e \quad = \quad x \qquad (\text{right} - \text{identity})
$$

- Yes, it does (also right-inverse).

Outline

Disproving

- First-order logic is only semi-decidable.
- A prover might not terminate when trying to prove a theorem that does not hold.
- Disproving as complementary task: Detect satisfiability and provide a model / counterexample.

Example Disproving - Group Theory

- Does commutativity hold in a group?

$$\begin{array}{llcll} \forall x, y, z & : & (x * y) * z & = & x * (y * z) & (\text{associativity}) \\ \forall x & : & e * x & = & x & (\text{left} - \text{identity}) \\ \forall x & : & i(x) * x & = & e & (\text{left} - \text{inverse}) \end{array}$$

- Hypothesis:

$$\forall x, y \quad : \quad x * y \quad = \quad y * x \qquad (\text{commutat.})$$

- No, it does not.

Example Disproving Cont. - Group Theory

Counterexample: a group with finite domain of size 6, where the elements 2 and 3 are not commutative:

Domain: $\{1, 2, 3, 4, 5, 6\}$

$e : 1$

$i :$

1	2	3	4	5	6
1	2	3	5	4	6

$* :$

	1	2	3	4	5	6
1	1	2	3	4	5	6
2	2	1	4	3	6	5
3	3	5	1	6	2	4
4	4	6	2	5	1	3
5	5	3	6	1	4	2
6	6	4	5	2	3	1

Outline

Disproving

Finite Domain Model Finding

- Approaches
- FM-Darwin

Evaluation

Idea

- Assume a fixed domain size n.
- Use a tool to decide if there exists a model with domain size n for a given problem.
- Do this starting with $n = 1$ with increasing n until a model is found.
- Note: domain of size n will consist of $\{1, \ldots, n\}$.

1. Approach: SEM-style

- Tools: SEM, Finder, Mace4
- Specialized constraint solvers.
- For a given domain generate all ground instances of the clause.
- Example: For domain size 2 and clause $p(a, g(x))$ the instances are $p(a, g(1))$ and $p(a, g(2))$.

1. Approach: SEM-style

- Set up multiplication tables for all symbols with the whole domain as cell values.
- Example: For domain size 2 and function symbol g with arity 1 the cells are $g(1) = \{1, 2\}$ and $g(2) = \{1, 2\}$.
- Try to restrict each cell to exactly 1 value.
- The clauses are the constraints guiding the search and propagation.
- Example: if the cell of a contains $\{1\}$, the clause $a = b$ forces the cell of b to be $\{1\}$ as well.

2. Approach: Mace-style

- Tools: Mace2, Paradox
- For given domain size n transform first-order clause set into equisatisfiable propositional clause set.
- Original problem has a model of domain size n iff the transformed problem is satisfiable.
- Run SAT solver on transformed problem and translate model back.

Paradox - Example

Domain: $\{1, 2\}$

Clauses: $\{p(a) \vee f(x) = a\}$

Flattened: $p(y) \vee f(x) = y \vee a \neq y$

Instances:
$$p(1) \vee f(1) = 1 \vee a \neq 1$$
$$p(2) \vee f(1) = 1 \vee a \neq 2$$
$$p(1) \vee f(2) = 1 \vee a \neq 1$$
$$p(2) \vee f(2) = 1 \vee a \neq 2$$

Totality:
$$a = 1 \vee a = 2$$
$$f(1) = 1 \vee f(1) = 2$$
$$f(2) = 1 \vee f(2) = 2$$

Functionality:
$$a \neq 1 \vee a \neq 2$$
$$f(1) \neq 1 \vee f(1) \neq 2$$
$$f(2) \neq 1 \vee f(2) \neq 2$$

Difficult Example

- Consider the clause set consisting of the $n \cdot (n-1)/2 + 1$ unit clauses:

$$p(c_1, \ldots, c_n)$$
$$\neg p(x_1, \ldots, x_{i-1}, x, x_{i+1}, \ldots, x_{j-1}, x, x_{j+1}, \ldots, x_n)$$

- Example for $n = 3$:

$Clauses$	$Model$
$p(c_1, c_2, c_3)$	$c_1 = 1$
$\neg p(x_1, x_1, x_3)$	$c_2 = 2$
$\neg p(x_1, x_2, x_1)$	$c_3 = 3$
$\neg p(x_1, x_2, x_2)$	$p(1, 2, 3)$

- Guess: For which n do Mace4 and Paradox give up?

Difficult Example

- Mace4 and Paradox give up for $n = 8$.
- There are n^{n-1} instances of the clause $\neg p(x_1, \ldots, x_{i-1}, x, x_{i+1}, \ldots, x_{j-1}, x, x_{j+1}, \ldots, x_n)$.
- Memory consumption is the main bottleneck.
- Our approach does not have this problem.

Outline

Disproving

Finite Domain Model Finding

- Approaches
- FM-Darwin

Evaluation

Paradox vs. FM-Darwin

Our approach is inspired by Paradox:

Paradox	**FM-*Darwin***
Clause set	Clause set
↓ Flattening	↓ Flattening’
Flat clause set	Flat clause set’
↓ Grounding wrt. domain	↓ Relational form wrt. domain
Propositional clause set	Function-free clause set
↓	↓
SAT solver	Decision procedure (*Darwin*)

FM-Darwin - Flattening

- A flat literal is one of:
  - $(\neg)\ p(x_1, \ldots, x_m)$ for predicate symbol p of arity m
  - $\neg f(x_1, \ldots, x_m) = y$ for function symbol f of arity m
  - $x = y$ or $x = x$
- Transformation into flat literals:
  - Extract subterms:

    Example: $p(a) \vee f(x) = c$ becomes $p(y) \vee y \neq a \vee f(x) = z \vee z \neq c$.
  - Remove trivial disequations:

    Example: $q(x, y) \vee x \neq y$ becomes $q(x, x)$.

FM-Darwin - Elimination of Function Symbols

- After exhaustive application of flattening all literals are flat and function symbols occur only in disequalities.
- Transform disequalities into a relations: $\neg f(x_1, \ldots, x_m) = y$ becomes $r_f(x_1, \ldots, x_m, y)$.
- Example: $f(x, y) = z$ becomes $r_f(x, y, z)$.
- The resulting clause sets contains no function symbols and no disequalities.

FM-Darwin - Totality Axioms

- Need to ensure that each relation r_f representing a function is left-total, i.e. defined for all arguments.

- For domain size n add for each function symbol f the axiom $R_f(x_1, \ldots, x_m, 1) \vee \ldots \vee R_f(x_1, \ldots, x_m, n)$.

FM-Darwin - Functionality Axioms

- No functionality axioms are needed.
- Any model can be transformed into one obeying right-uniqueness.
- All positive literals over r_f occur only in the totality axiom, nowhere else.
- Example: Say $r_f(1, 1, 1)$ and $r_f(1, 1, 2)$ are true in a model. After setting $r_f(1, 1, 2)$ to false the totality axiom is still satisfied, and any other clause which was satisfied previously is still satisfied.

FM-Darwin - Equality Axiomatization

- Add the axioms $d \neq d'$ and $d' \neq d$ for all different domain elements.

- Example: For domain size 2 the axioms are $1 \neq 2$ and $2 \neq 1$.

FM-Darwin - Example

Domain:	$\{1, 2\}$
Clauses:	$\{p(a) \vee f(x) = a\}$
Flattened:	$p(y) \vee f(x) = y \vee a \neq y$
Relational:	$p(y) \vee r_f(x, y) \vee \neg r_a(y)$
Totality:	$r_a(1) \vee r_a(2)$ $r_f(x, 1) \vee r_f(x, 2)$
Equality:	$1 \neq 2$ $2 \neq 1$

Space Complexity

While Paradox generates exponentially many clauses, n^k for a clause containing k variables, FM-Darwin generates only a quadratic number.

Outline

Setup

- Compared the two best model finders according to the last CASC competition (Mace4 and Paradox) with FM-Darwin
- Used all satisfiable clausal TPTP 3.1.1 problems as a benchmark.
- Configuration: Xeon 2.4Ghz CPU, a limit of 5 minutes and 500MB of RAM for a process.

Results

Type Horn	Type Equ.	Total	FM-Darwin Solv.	FM-Darwin Time	Mace4 Solv.	Mace4 Time	Paradox 1.3 Solv.	Paradox 1.3 Time
no	no	400	383	2.5	272	3.8	372	1.0
no	yes	154	120	6.0	63	3.9	101	0.7
yes	no	65	37	8.3	37	0.2	59	2.2
yes	yes	196	135	3.7	181	3.7	182	5.3
all		815	675	3.7	553	3.5	714	2.1

Type: split into Horn problems and problems containing equality.
Total: number of problems per category.
Solv.: number of problems solved by a system.
Time: average time (in s) needed for the solved problems.

Observations

- While Mace4 and Paradox are in general noticeably faster than FM-Darwin, the difference is not dramatic.
- Memory consumption limits the scalability of Mace4 and Paradox.
- For the 101 problems that Paradox can not solve in 5 minutes, it gives up for all but 15.
- FM-Darwin solves 64 resp. 54 problems on which Mace4 resp. Paradox give up.
- FM-Darwin solves more problems than Mace4, and more non-Horn problems than any other system.