Theorem Proving with Vampire for Rigorous Systems Engineering

Finite Models

Giles Reger and Martin Suda

University of Manchester and TU Wein

Introduction

In the standard saturation mode we only get saturated sets when the formula is satisfied. This is not always useful.

Additionally, saturation can find it difficult to find saturated sets even if there are small finite models

So we implemented finite model building

Finite Model Building

- Based on ideas in Paradox (and earlier, MACE)
- Useful for establishing non-theorems i.e. satisfiability checking

- *Idea:* For a domain size n create a ground problem that is satisfiable if the original problem has a finite model of size n.

- The ground literals can be (consistently) named/translated into SAT variables and the ground problem decided by a SAT solver

- We can just check for bigger and bigger values of n

Preparing the Problem

- **Definition Introduction.** This reduces the size of clauses produced by flattening. A clause $p(f(a,b), g(f(a,b)))$ becomes $p(t_1, t_2)$ and we introduce the definition clauses $t_1 = f(a,b)$ and $t_2 = g(t_1)$

- **Flattening.** This is necessary for the technique in general. A clause $p(f(a,b), g(f(a,b)))$ becomes

$$p(x_1, x_2) \vee x_1 \neq f(x_3, x_4) \vee x_2 \neq g(x_1) \vee x_3 \neq a \vee x_4 \neq b$$

- **Splitting.** This can reduce the number of variables in clauses (important later). The clause $p(x,y) \vee q(y,z)$ is transformed to the two clauses $p(x,y) \vee s(y)$ and $\neg s(y) \vee q(y,z)$.

The Constraints

- **Groundings.** For each (flattened) clause $C[\mathbf{x}]$ and each vector of domain constants $\mathbf{d}$ translate and add $C[\mathbf{d}]$
- **Functionality.** For each function symbol f with arity a, vector of domain constants $\mathbf{d}$ of length a and distinct domain constants d_1 and d_2 translate and add $f(\mathbf{d}) \neq d_1 \vee f(\mathbf{d}) \neq d_2$
- **Totality.** For each function symbol f with arity a and vector of domain constants $\mathbf{d}$ of length a translate and add $f(\mathbf{d}) = d_1 \vee \ldots \vee f(\mathbf{d}) = d_n$ for (all) the domain constants d_i
- Note the <u>exponential</u> nature of these constraint sets

Symmetry Breaking

- Any model will be symmetrical in ordering of domain constants
- So the SAT solver will be checking the same model multiple times
- We can (partly) break these symmetries by ordering ground terms
- Pick and order n ground terms (include all constants at the front)
- For term t_i and domain size n add the clauses

$$t_i \neq d_m \vee t_1 = d_{m-1} \vee \ldots \vee t_{i-1} = d_{m-1}$$

  for $m \leq n$ and if $i \leq n$ add

$$t_i = d_1 \vee \ldots \vee t_i = d_i$$

Sort Inference

- Separate constants and function positions into different distinct <u>sorts</u>
- Under certain conditions we can detect a maximum size for a sort
- This information can render certain constraints redundant i.e. if a sort does not need to be bigger than a certain size then we drop constraints talking about those larger sizes
- Sort inference in the multi-sorted case is much more involved, see the SAT 2015 paper

Importance of the SAT Solver

- The majority of time is spent inside the SAT solver
- Therefore, making the SAT solver faster can improve this method.
- **Variable Elimination.** As implemented in e.g. MiniSAT. Idea is to apply all resolutions on a variable to eliminate it. Only do this if it will reduce the size. Removes pure variables.
    - Can help a lot
    - Can make things worse

Deciding Non-Non-Theorems (e.g. Unsat)

- This is a decision procedure for EPR i.e. we stop at n where n is the number of constants in the problem
- The input can restrict the size of the domain, then we can detect the absence of a model i.e.

  $$X = Y \lor X = Z \lor Y = Z$$

  means $n \leq 2$
- Or less obscurely,

  $$X = a \lor X = b$$

  also means $n \leq 2$

Starting from more than 1

- Conversely
$$a \neq b \land a \neq c \land b \neq c$$
means that $n \geq 3$
- We detect cliques in constant disequalities and use this number as the starting point

Group Theory Example

Consider this problem from group theory

```
fof(left_identity,axiom, ! [X] : mult(e,X) = X).
fof(left_inverse,axiom, ! [X] : ?[Y] : mult(Y,X) = e).
fof(associativity,axiom,
    ! [X,Y,Z] : mult(mult(X,Y),Z) = mult(X,mult(Y,Z))).
fof(group_of_order_2,axiom,! [X] : mult(X,X) = e).

fof(a,axiom,a!=e).
```

We get a model of size 2

Group Theory Example

Consider this problem from group theory

```
fof(left_identity,axiom, ! [X] : mult(e,X) = X).
fof(left_inverse,axiom, ! [X] : ?[Y] : mult(Y,X) = e).
fof(associativity,axiom,
    ! [X,Y,Z] : mult(mult(X,Y),Z) = mult(X,mult(Y,Z))).
fof(group_of_order_2,axiom,! [X] : mult(X,X) = e).
```

```
fof(a,axiom,$distinct(a,b,c)).
```

We get a model of size 4

Group Theory Example

Consider this problem from group theory

```
fof(left_identity,axiom, ! [X] : mult(e,X) = X).
fof(left_inverse,axiom, ! [X] : ?[Y] : mult(Y,X) = e).
fof(associativity,axiom,
    ! [X,Y,Z] : mult(mult(X,Y),Z) = mult(X,mult(Y,Z))).
fof(group_of_order_2,axiom,! [X] : mult(X,X) = e).
```

```
fof(a,axiom,$distinct(a,b,c)).
fof(a,axiom,![X]:(X = a | X = b | X = c)).
```

We don't get any models

Handling Multi-Sorted Input

If we have multiple (user-defined) sorts then those sorts may have different sizes in the model. This means that the standard approach no longer works.

We could enumerate all possible size combinations. But this is inefficient (demonstrated next) so we have a better way of searching.

Organized Monkey Village example

Example with three sorts: *monkey*, *banana*, *tree*

$$(\forall M : \mathit{monkey})(\mathrm{owns}(M, \mathrm{b}_1(M)) \wedge \mathrm{owns}(M, \mathrm{b}_2(M)) \wedge \mathrm{b}_1(M) \neq \mathrm{b}_2(M))$$

$$(\forall M_1, M_2 : \mathit{monkey})(\forall B : \mathit{banana})(\mathrm{owns}(M_1, B) \wedge \mathrm{owns}(M_2, B) \rightarrow M_1 = M_2)$$

Organized Monkey Village example

Example with three sorts: *monkey*, *banana*, *tree*

$$(\forall M : \mathit{monkey})(\mathrm{owns}(M, \mathrm{b}_1(M)) \wedge \mathrm{owns}(M, \mathrm{b}_2(M)) \wedge \mathrm{b}_1(M) \neq \mathrm{b}_2(M))$$

$$(\forall M_1, M_2 : \mathit{monkey})(\forall B : \mathit{banana})(\mathrm{owns}(M_1, B) \wedge \mathrm{owns}(M_2, B) \rightarrow M_1 = M_2)$$

$$(\forall T : \mathit{tree})(\exists M_1, M_2, M_3 : \mathit{monkey})((\bigwedge_{i=1}^{3} \mathrm{sits}(M_i) = T) \wedge \mathrm{dist}(M_1, M_2, M_3))$$

$$(\forall M_1, M_2, M_3, M_4 : \mathit{monkey})(\forall T : \mathit{tree})((\bigwedge_{i=1}^{4} \mathrm{sits}(M_i) = T) \Rightarrow \neg\mathrm{dist}(M_1, M_2, M_3,$$

Organized Monkey Village example

Example with three sorts: *monkey*, *banana*, *tree*

$$(\forall M : \mathit{monkey})(\mathrm{owns}(M, \mathrm{b}_1(M)) \wedge \mathrm{owns}(M, \mathrm{b}_2(M)) \wedge \mathrm{b}_1(M) \neq \mathrm{b}_2(M))$$

$$(\forall M_1, M_2 : \mathit{monkey})(\forall B : \mathit{banana})(\mathrm{owns}(M_1, B) \wedge \mathrm{owns}(M_2, B) \rightarrow M_1 = M_2)$$

$$(\forall T : \mathit{tree})(\exists M_1, M_2, M_3 : \mathit{monkey})((\bigwedge_{i=1}^{3} \mathrm{sits}(M_i) = T) \wedge \mathrm{dist}(M_1, M_2, M_3))$$

$$(\forall M_1, M_2, M_3, M_4 : \mathit{monkey})(\forall T : \mathit{tree})((\bigwedge_{i=1}^{4} \mathrm{sits}(M_i) = T) \Rightarrow \neg \mathrm{dist}(M_1, M_2, M_3,$$

$$(\forall M : \mathit{monkey})(\mathrm{partner}(M) \neq M \wedge \mathrm{partner}(\mathrm{partner}(M)) = M)$$

Organized Monkey Village example

Example with three sorts: *monkey*, *banana*, *tree*

$$(\forall M : \mathit{monkey})(\mathrm{owns}(M, \mathrm{b}_1(M)) \wedge \mathrm{owns}(M, \mathrm{b}_2(M)) \wedge \mathrm{b}_1(M) \neq \mathrm{b}_2(M))$$

$$(\forall M_1, M_2 : \mathit{monkey})(\forall B : \mathit{banana})(\mathrm{owns}(M_1, B) \wedge \mathrm{owns}(M_2, B) \rightarrow M_1 = M_2)$$

$$(\forall T : \mathit{tree})(\exists M_1, M_2, M_3 : \mathit{monkey})((\bigwedge_{i=1}^{3} \mathrm{sits}(M_i) = T) \wedge \mathrm{dist}(M_1, M_2, M_3))$$

$$(\forall M_1, M_2, M_3, M_4 : \mathit{monkey})(\forall T : \mathit{tree})((\bigwedge_{i=1}^{4} \mathrm{sits}(M_i) = T) \Rightarrow \neg \mathrm{dist}(M_1, M_2, M_3,$$

$$(\forall M : \mathit{monkey})(\mathrm{partner}(M) \neq M \wedge \mathrm{partner}(\mathrm{partner}(M)) = M)$$

Fair, but blind enumeration: $(m = 1, b = 1, t = 1)$

Organized Monkey Village example

Example with three sorts: *monkey*, *banana*, *tree*

$$(\forall M : \mathit{monkey})(\mathrm{owns}(M, \mathrm{b}_1(M)) \wedge \mathrm{owns}(M, \mathrm{b}_2(M)) \wedge \mathrm{b}_1(M) \neq \mathrm{b}_2(M))$$

$$(\forall M_1, M_2 : \mathit{monkey})(\forall B : \mathit{banana})(\mathrm{owns}(M_1, B) \wedge \mathrm{owns}(M_2, B) \to M_1 = M_2)$$

$$(\forall T : \mathit{tree})(\exists M_1, M_2, M_3 : \mathit{monkey})((\bigwedge_{i=1}^{3} \mathrm{sits}(M_i) = T) \wedge \mathrm{dist}(M_1, M_2, M_3))$$

$$(\forall M_1, M_2, M_3, M_4 : \mathit{monkey})(\forall T : \mathit{tree})((\bigwedge_{i=1}^{4} \mathrm{sits}(M_i) = T) \Rightarrow \neg \mathrm{dist}(M_1, M_2, M_3,$$

$$(\forall M : \mathit{monkey})(\mathrm{partner}(M) \neq M \wedge \mathrm{partner}(\mathrm{partner}(M)) = M)$$

Fair, but blind enumeration: $(m = 1, b = 1, t = 1) \to^1 (2, 1, 1) \to^2 (1, 2, 1) \to^3 (1, 1, 2) \to^4 (3, 1, 1) \to^5 (2, 2, 1) \to^6 (2, 1, 2) \to^7 (1, 3, 1) \to^8 (1, 2, 2) \to^9 (1, 1, 3) \to^{10} \ldots$

Organized Monkey Village example

Example with three sorts: *monkey*, *banana*, *tree*

$$(\forall M : \mathit{monkey})(\mathrm{owns}(M, \mathrm{b}_1(M)) \wedge \mathrm{owns}(M, \mathrm{b}_2(M)) \wedge \mathrm{b}_1(M) \neq \mathrm{b}_2(M))$$

$$(\forall M_1, M_2 : \mathit{monkey})(\forall B : \mathit{banana})(\mathrm{owns}(M_1, B) \wedge \mathrm{owns}(M_2, B) \rightarrow M_1 = M_2)$$

$$(\forall T : \mathit{tree})(\exists M_1, M_2, M_3 : \mathit{monkey})((\bigwedge_{i=1}^{3} \mathrm{sits}(M_i) = T) \wedge \mathrm{dist}(M_1, M_2, M_3))$$

$$(\forall M_1, M_2, M_3, M_4 : \mathit{monkey})(\forall T : \mathit{tree})((\bigwedge_{i=1}^{4} \mathrm{sits}(M_i) = T) \Rightarrow \neg \mathrm{dist}(M_1, M_2, M_3,$$

$$(\forall M : \mathit{monkey})(\mathrm{partner}(M) \neq M \wedge \mathrm{partner}(\mathrm{partner}(M)) = M)$$

Fair, but blind enumeration: $(m = 1, b = 1, t = 1) \rightarrow^1 (2, 1, 1) \rightarrow^2$
$(1, 2, 1) \rightarrow^3 (1, 1, 2) \rightarrow^4 (3, 1, 1) \rightarrow^5 (2, 2, 1) \rightarrow^6 (2, 1, 2) \rightarrow^7$
$(1, 3, 1) \rightarrow^8 (1, 2, 2) \rightarrow^9 (1, 1, 3) \rightarrow^{10} \ldots$
$\ldots \rightarrow^{2661} (m = 6, b = 12, t = 2)$

Our Solution

Intuitively:

- When we fail to find a model look to see which constraints blocked us
- Relax those constraints next time

Technically (abstractly):

- enrich the encoding with special <u>marking literals</u>
- SAT-solving under assumptions
- from an UNS answer we learn a <u>constraint</u>
- constraints prune the space of domain size assignments to try

Demonstrate Monkey Village Example

Run this on the command line to see what happens.