Towards Smarter MACE-style Model Finders

Mikoláš Janota\textsuperscript{1,2} Martin Suda\textsuperscript{2}

\textsuperscript{1} IST/INESC-ID, University of Lisbon, Portugal
\textsuperscript{2} Czech Technical University in Prague

LPAR, 2018, Ethiopia
Does a FOL formula have a finite model?
Background and Motivation

Does a FOL formula have a finite model?

- Finite models useful:
  - “debugging” of wrong theorems
  - “debugging” of wrong programs
  - information for lemma selection learning
Does a FOL formula have a finite model?

- Finite models useful:
  - “debugging” of wrong theorems
  - “debugging” of wrong programs
  - Information for lemma selection learning

- Advantage:
  - If a finite model exists, it is found in finite time.
  - Complete for some theories (Bernays-Schönfinkel, a.k.a. EPR)
MACE-style Framework

\[ k := 1 \]

\[ k < \text{limit?} \]
MACE-style Framework

$k := 1$

$k < \text{limit?}$

yes

has model of size $k$?
MACE-style Framework

$k := 1$

$k < \text{limit}$?

yes

has model of size $k$?

yes

SAT
MACE-style Framework

1. $k := 1$
2. $k < \text{limit}?$
   - yes (has model of size $k$?)
     - yes (SAT)
     - no ($k := k + 1$)
   - no
MACE-style Framework

\[ k := 1 \]

\[ k < \text{limit?} \]

- **no**
  - UNSAT
- **yes**
  - has model of size \( k \)?
    - **no**
      - \( k := k + 1 \)
    - **yes**
      - SAT
**Issue:**
Encoding directly to SAT is exponential, eventually blows up
Avoiding Space Explosion

**Issue:**
Encoding directly to SAT is exponential, eventually blows up

**Remedy:**
Encode into SAT *lazily* by Count-Example Abstraction Guided Refinement (CEGAR)
CEGAR for Fixed Size $k$

$$(\exists \vec{p}) (\forall \vec{x}) \phi$$

$\vec{p}$ predicates, $\vec{f}$ functions, $\vec{x}$ FOL variables

Algorithm sketch:

1. $\alpha := true$
CEGAR for Fixed Size $k$

\[ (\exists \vec{p} \vec{f})(\forall \vec{x}) \phi \]
\[ \vec{p} \text{ predicates, } \vec{f} \text{ functions, } \vec{x} \text{ FOL variables} \]

Algorithm sketch:

1. $\alpha := \text{true}$
2. Find model $\mathcal{I}$ for $\alpha$
CEGAR for Fixed Size $k$

$$(\exists \vec{p} \vec{f})(\forall \vec{x}) \phi$$

$\vec{p}$ predicates, $\vec{f}$ functions, $\vec{x}$ FOL variables

Algorithm sketch:

1. $\alpha := true$
2. Find model $I$ for $\alpha$
3. If $\alpha$ UNSAT, RETURN false
(∃\vec{p} \vec{f})(\forall \vec{x}) \phi

\vec{p} \text{ predicates, } \vec{f} \text{ functions, } \vec{x} \text{ FOL variables}

Algorithm sketch:

1. \( \alpha := \text{true} \)
2. Find model \( \mathcal{I} \) for \( \alpha \)
3. If \( \alpha \) UNSAT, RETURN \text{false}
4. Find counterexample \( \mu \) to \( \mathcal{I} \) in original formula
CEGAR for Fixed Size $k$

$$(\exists \vec{p} \vec{f})(\forall \vec{x}) \phi$$

$\vec{p}$ predicates, $\vec{f}$ functions, $\vec{x}$ FOL variables

Algorithm sketch:

1. $\alpha := true$
2. Find model $\mathcal{I}$ for $\alpha$
3. If $\alpha$ UNSAT, RETURN false
4. Find counterexample $\mu$ to $\mathcal{I}$ in original formula
5. If no counterexample, RETURN true
(∃\vec{p} \vec{f})(∀\vec{x}) \phi
\vec{p} \text{ predicates, } \vec{f} \text{ functions, } \vec{x} \text{ FOL variables}

Algorithm sketch:

1. \( \alpha := \text{true} \)
2. Find model \( \mathcal{I} \) for \( \alpha \)
3. If \( \alpha \) UNSAT, \text{RETURN} \ false
4. Find counterexample \( \mu \) to \( \mathcal{I} \) in original formula
5. If no counterexample, \text{RETURN} \ true
6. Strengthen abstraction: \( \alpha := \alpha \land \phi[\mu/\vec{x}] \)
CEGAR for Fixed Size $k$

$\left( \exists \vec{p} \vec{f} \right) \left( \forall \vec{x} \right) \phi$

$\vec{p}$ predicates, $\vec{f}$ functions, $\vec{x}$ FOL variables

**Algorithm sketch:**

1. $\alpha := true$
2. Find model $\mathcal{I}$ for $\alpha$
3. If $\alpha$ UNSAT, RETURN false
4. Find counterexample $\mu$ to $\mathcal{I}$ in original formula
5. If no counterexample, RETURN true
6. Strengthen abstraction: $\alpha := \alpha \land \phi[\mu/\vec{x}]$
7. GOTO 2
CEGAR for Fixed Size $k$

$\alpha := true$

$\alpha$ SAT?

Is model? yes

Strengthen no

UNSAT no
CEGAR for Fixed Size $k$

$\alpha := true$

$\alpha$ SAT?

yes

complete model
CEGAR for Fixed Size $k$

$\alpha := \text{true}$

$\alpha$ SAT?

- yes

- complete model

- Is model?
CEGAR for Fixed Size $k$

$\alpha := true$

$\alpha$ SAT?

- yes: complete model

Is model?

- yes: SAT

- no: UNSAT

Strengthen $\alpha$ no
CEGAR for Fixed Size $k$

$\alpha := true$

$\alpha$ SAT?

- yes
  - complete model
    - Is model?
      - yes
        - SAT
      - no
        - Strengthen $\alpha$

- no
  - Strengthen $\alpha$
CEGAR for Fixed Size $k$

- $\alpha := true$
- $\alpha$ SAT?
  - yes: complete model
  - no:
    - UNSAT
    - Is model?
      - yes: SAT
      - no:
        - Strengthen $\alpha$
          - no
We have model of ground $\alpha$.

**Example:**

$$p(0), \neg q(0) \models p(0) \lor q(0)$$
Completing Models

We have model of ground $\alpha$,

**Example:**

\[ p(0), \neg q(0) \models p(0) \lor q(0) \]

We need to complete into interpretation of original Examples for \( (\forall x)(p(x) \lor \neg q(y)) \)

\[ p \triangleq \{0\}, q \triangleq \{\} \]

\[ p \triangleq \{0\}, q \triangleq \{1\} \]

\[ p \triangleq 2^{1..k}, q \triangleq \{\} \]
Completing Models Contd.

Natural approach: set undefined to false/true

**Examples**

\[ \{ p(0), \neg p(1), \neg p(2) \} \ldots p \triangleq 2^{1\ldots k} \setminus \{1, 2\} \]

\[ \{ p(0), \neg p(1), \neg p(2) \} \ldots p \triangleq \{p(0)\} \]
Completing Models Contd.

Natural approach: set undefined to false/true

**Examples**

\[ \{p(0), \neg p(1), \neg p(2)\} \ldots p \triangleq 2^{1\ldots k} \setminus \{1, 2\} \]

\[ \{p(0), \neg p(1), \neg p(2)\} \ldots p \triangleq \{p(0)\} \]

**Issue:** completion uninformed
Completing Models Contd.

Natural approach: set undefined to false/true

Examples

\[ \{ p(0), \neg p(1), \neg p(2) \} \ldots p \triangleq 2^{1..k} \setminus \{1, 2\} \]

\[ \{ p(0), \neg p(1), \neg p(2) \} \ldots p \triangleq \{ p(0) \} \]

**Issue:** completion uninformed

**Remedy:**
Learn the completion with Machine Learning techniques.
1. \((\forall \vec{x}) \ p(x_1, \ldots, x_n) \iff (x_1 = t)\)
\[1 \quad (\forall \vec{x}) \ p(x_1, \ldots, x_n) \leftrightarrow (x_1 = t)\]

\[2 \quad \text{Ground by } \{x_1 \mapsto 0, x_2 \mapsto 0, \ldots, x_n \mapsto 0\}: \ p(0, \ldots, 0) \leftrightarrow 0 = t\]
1. $(\forall \vec{x}) \, p(x_1, \ldots, x_n) \leftrightarrow (x_1 = t)$

2. Ground by $\{x_1 \mapsto 0, x_2 \mapsto 0, \ldots, x_n \mapsto 0\} : p(0, \ldots, 0) \leftrightarrow 0 = t$

3. Ground by $\{x_1 \mapsto 1, x_2 \mapsto 0, \ldots, x_n \mapsto 0\} : p(1, \ldots, 0) \leftrightarrow 1 = t$
1. $(\forall \vec{x})\; p(x_1,\ldots,x_n) \leftrightarrow (x_1 = t)$

2. Ground by $\{x_1 \mapsto 0, x_2 \mapsto 0, \ldots, x_n \mapsto 0\}$: $p(0,\ldots,0) \leftrightarrow 0 = t$

3. Ground by $\{x_1 \mapsto 1, x_2 \mapsto 0, \ldots, x_n \mapsto 0\}$: $p(1,\ldots,0) \leftrightarrow 1 = t$

4. $\alpha = (p(0,\ldots,0) \leftrightarrow 0 = t) \land (p(1,\ldots,0) \leftrightarrow 1 = t)$
(1) \( (\forall \vec{x}) \ p(\vec{x}) \leftrightarrow (x_1 = t) \)

2. Ground by \( \{x_1 \mapsto 0, x_2 \mapsto 0, \ldots, x_n \mapsto 0\} \): \( p(0, \ldots, 0) \leftrightarrow 0 = t \)

3. Ground by \( \{x_1 \mapsto 1, x_2 \mapsto 0, \ldots, x_n \mapsto 0\} \): \( p(1, \ldots, 0) \leftrightarrow 1 = t \)

4. \( \alpha = (p(0, \ldots, 0) \leftrightarrow 0 = t) \land (p(1, \ldots, 0) \leftrightarrow 1 = t) \)

5. Model of \( \alpha \):
   - \( t \triangleq 1 \)
   - \( p(0, \ldots, 0) \triangleq \text{False} \)
   - \( p(1, \ldots, 0) \triangleq \text{True} \)
1. \((\forall \vec{x}) \; p(x_1, \ldots, x_n) \iff (x_1 = t)\)

2. Ground by \(\{x_1 \mapsto 0, x_2 \mapsto 0, \ldots, x_n \mapsto 0\}\): \(p(0, \ldots, 0) \iff 0 = t\)

3. Ground by \(\{x_1 \mapsto 1, x_2 \mapsto 0, \ldots, x_n \mapsto 0\}\): \(p(1, \ldots, 0) \iff 1 = t\)

4. \(\alpha = (p(0, \ldots, 0) \iff 0 = t) \land (p(1, \ldots, 0) \iff 1 = t)\)

5. Model of \(\alpha\):
   \(t \triangleq 1\)
   \(p(0, \ldots, 0) \triangleq \text{False}\)
   \(p(1, \ldots, 0) \triangleq \text{True}\)

6. **Learn:**
   \(t \triangleq 1\)
   \(p(x_1, \ldots, x_n) \triangleq (x_1 = 1)\)
Learning by decision trees
Learning by **decision trees**

- Function and predicates eliminated by **Ackermann reduction**
Learning by decision trees
Function and predicates eliminated by Ackermann reduction
Finite domains encoded to SAT by unary encoding
Learning by **decision trees**
- Function and predicates eliminated by **Ackermann reduction**
- Finite domains encoded to SAT by **unary encoding**
- Incremental SAT (**minisat**)
- Learning by **decision trees**
- Function and predicates eliminated by **Ackermann reduction**
- Finite domains encoded to SAT by **unary encoding**
- Incremental SAT (**minisat**)  
- Support for non-prenex
Learning by **decision trees**

- Function and predicates eliminated by **Ackermann reduction**
- Finite domains encoded to SAT by **unary encoding**
- Incremental SAT (**minisat**)  
- Support for non-prenex  
- Symmetry breaking, e.g. \( c_1 \triangleq 0 \)
Results EPR

- cegar+learn
- vam-fm
- cvc4
- cvc4-epr
- z3
- iprover

CPU time (s) vs instances for different solvers.
Results EPR: QFM

CPU time (s)
instances
cegar+learn
cegar
expand
Results EPR: Learning Method

The graph shows the CPU time (in seconds) against the number of instances for two different methods: cegar+learn-DT-LEQ (red line) and cegar+learn-DT-EQ (blue line). The x-axis represents the number of instances, while the y-axis shows the CPU time. The data indicates a steady increase in CPU time as the number of instances grows.
Results EPR: Learning Method (SAT)

- cegar+learn-DT-LEQ
- cegar+learn-DT-EQ

CPU time (s) vs instances graph.
Results SAT NON-EPR

CPU time (s)

instances

vam-fm
cegar
iprover
cvc4
expand
cegar+learn
z3
Summary and Future

- CEGAR for lazy SAT-based model finite model finding
CEGAR for lazy SAT-based model finite model finding
Observing a formula while solving, learn from that
Summary and Future

- CEGAR for lazy SAT-based model finite model finding
- Observing a formula while solving, learn from that
- Better learning methods?
Summary and Future

- CEGAR for lazy SAT-based model finite model finding
- Observing a formula while solving, learn from that
- Better learning methods?
- Learning in the presence of theories?
CEGAR for lazy SAT-based model finite model finding
- Observing a formula while solving, learn from that
- Better learning methods?
- Learning in the presence of theories?
- Infinite domains?
http://sat2019.tecnico.ulisboa.pt