First-order theorem (dis)proving for reachability problems in verification and experimental mathematics

Alexei Lisitsa

University of Liverpool,

FoMM/Lean together 2020, CMU, January 8, 2020

Overview

- Preamble: MIU system and MU puzzle
- Reachability as deducibility
- Part I: Verification via disprovng by countermodel finding
  - Cache Coherence Protocols
  - **Linear Systems of Automata and Monotonic Abstraction**
  - Regular Model Checking
  - Regular Tree Model Checking
  - Lossy Channel Systems
  - Safety for general TRS and Tree Automata Completion
  - **Limitations and Challenges**
- Part II: Applications to Mathematics
  - Exploration of the Andrews-Curtis Conjecture via FO (dis)proving

MIU system

Alphabet: M, I and U
Axiom: MI
Derivation rules:

I. If xI is a theorem, so is xIU.

II. If Mx is theorem, so is Mxx.

III. In any theorem III can be replaced by U.

IV. UU can be dropped from any theorem.

MU puzzle

Is MU a theorem of MIU system?

Douglas Hofstadter, Goedel, Escher, Bach: An eternal Golden Braid, 1979

MU puzzle

- **Answer:** Negative, that is $MU \notin L_{MIU}$
- **Condition, I (GEB,79):** "the number of I symbols in any string in L_{MIU} cannot be multiple of three"
- **Condition, 2 (Swanson, McEliece, 1988):** "any MIU theorem should start with M followed by an arbitrary word in I's and U's"

MU puzzle (cont.)

- **Question:** How to solve it automatically?

MU puzzle (cont.)

- **Question:** How to solve it automatically?
- **Answer:** Let's apply classical FO logic ...

MU puzzle (cont.)

- **Question:** How to solve it automatically?
- **Answer:** Let's apply classical FO logic ...
  - Fully automated solution of the puzzle
  - Puzzle is considered as infinite state safety verification problem
  - Generic Finite Countermodels Method (FCM) is used

Back to MU puzzle: Logic encoding

FO theory MIU:

1. $(x * y) * z = x * (y * z)$ (associativity of concatenation);
2. $e * x = x$;
3. $x * e = x$;
4. $T(M * I)$ (MI is a theorem of MIU);
5. $T(x * I) \to T(x * I * U)$ (rule I of MIU);
6. $T(M * x) \to T(M * x * x)$ (rule II of MIU);
7. $T(x * I * I * I * y) \to T(x * U * y)$ (rule III of MIU)
8. $T(x * U * U * y) \to T(x * y)$ (rule IV of MIU)

Back to MU puzzle: Logic encoding (cont.)

Proposition

If $w \in L_{MIU}$ then $MIU \vdash T(t_w)$

Corollary

- *If $T(t_S)$ is not FO provable from T_{MIU}, that is $T_{MIU} \not\vdash_{FO} T(t_S)$ then $S \notin L_{MIU}$;*
- *For any non-ground term $t(\bar{x})$ in vocabulary $\{*, M, I, U\}$ over the set of variables X, if $T_{MIU} \not\vdash_{FO} \exists \bar{x} T(t(\bar{x}))$ then none of S such that t_S is a ground instance of $t(\bar{x})$ belongs to L_{MIU}.*

Finite countermodels

Now to show $MIU \not\vdash T(M * U)$ we are looking for

- Finite countermodels for $MIU \rightarrow T(M * U)$, or equivalently, for
- Finite models for $MIU \wedge \neg T(M * U)$

To find a model we apply generic finite model finding procedure, e.g. implemented in Mace4 finite model finder by W.McCune
(see demonstration)

- A model of size 3 is found in less than 0.01s. The property is proven!

Finite countermodels

Now to show $MIU \not\vdash T(M * U)$ we are looking for

- Finite countermodels for $MIU \rightarrow T(M * U)$, or equivalently, for
- Finite models for $MIU \wedge \neg T(M * U)$

To find a model we apply generic finite model finding procedure, e.g. implemented in Mace4 finite model finder by W.McCune
(see demonstration)

- A model of size 3 is found in less than 0.01s. The property is proven!

CounterModel as Invariant

The domain D of the model is a three element set $\{0, 1, 2\}$. Interpretations of constants: $[I] = [M] = 0$, $[U] = 1$. Interpretation of the predicate T: $[T] = \{1, 2\}$.
The interpretation of the binary function $*$ is given by the following table

	0	1	2
0	2	0	1
1	0	1	2
2	1	2	0

Invariant property which holds for any MIU theorem w:

$$[t_w] \in [T] = \{1, 2\}$$

Notice that $[t_{MU}] = 0 * 1 = 0 \notin [T]$

In summary

- The interpretation $[*]$ above defines the set of strings $L_{\mathcal{M}} = \{s \mid [t_s]_{\mathcal{M}} \in \{1,2\}\}$ for which
  - $L_{MIU} \subseteq L_{\mathcal{M}}$
  - $MU \notin L_{\mathcal{M}}$
- Thus, $L_{\mathcal{M}}$ is an invariant separating the theorems of MIU system and the string in question, MU

CounterModel as Invariant (cont.)

In summary

- The interpretation $[*]$ above defines the set of strings $L_{\mathcal{M}} = \{s \mid [t_s]_{\mathcal{M}} \in \{1, 2\}\}$ for which
  - $L_{MIU} \subseteq L_{\mathcal{M}}$
  - $MU \notin L_{\mathcal{M}}$
- Thus, $L_{\mathcal{M}}$ is an invariant separating the theorems of MIU system and the string in question, MU
- It is easy to see also that the invariant is a *regular* language

CounterModel as Invariant (cont.)

In summary

- The interpretation $[*]$ above defines the set of strings $L_{\mathcal{M}} = \{s \mid [t_s]_{\mathcal{M}} \in \{1,2\}\}$ for which
  - $L_{MIU} \subseteq L_{\mathcal{M}}$
  - $MU \notin L_{\mathcal{M}}$
- Thus, $L_{\mathcal{M}}$ is an invariant separating the theorems of MIU system and the string in question, MU
- It is easy to see also that the invariant is a *regular* language
- Interestingly, $L_{\mathcal{M}} \neq L_{MIU}$ as, for example, $[M * M] = 2 \in [T]$ hence $MM \in L_{\mathcal{M}}$ but $MM \notin L_{MIU}$.

$MM \notin L_{MIU}$

Let us search for countermodels for $MIU \to T(M * M)$.
Mace4 finds a countermodel $\mathcal{M}'$ of size 2, with the domain $\{0, 1\}$, the interpretations of constants M, I and U as $1, 0$ and 0, respectively; the interpretation $[T]$ of $T = \{1\}$. the interpretation of * is given by the table

```
[*]     0   1
          ----
      0  |0,1
      1  |1,0
```

The corresponding invariant $\{s \mid [t_s]_{\mathcal{M}'} = 1\}$ captures the "oddness" of M count in strings, which is sufficient to separate MM from L_{MIU}.

Subsets of configurations in FCM proofs

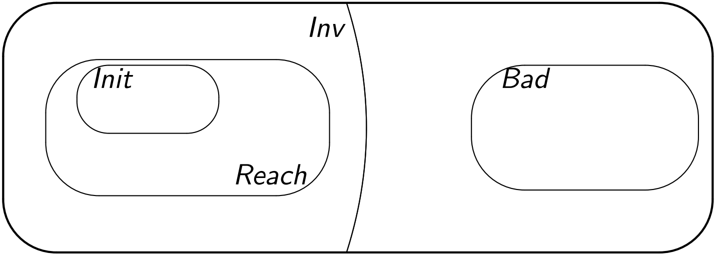

Figure: Subsets of configurations in general position

MU puzzle via formal verification

- MU puzzle was considered as an example in
  E. M. Clarke, A. Fehnker, Z. Han, B. Krogh, J. Ouakine, Abtsraction and Counterexample-Guided Refinement in Model Checking of Hybrid System, 2002
- It has been formally verified that MU is not a theorem of MIU, but the proof was not fully automated and required "a good deal of insight'

MU puzzle via formal verification

- MU puzzle was considered as an example in
  E. M. Clarke, A. Fehnker, Z. Han, B. Krogh, J. Ouakine, Abtsraction and Counterexample-Guided Refinement in Model Checking of Hybrid System, 2002
- It has been formally verified that MU is not a theorem of MIU, but the proof was not fully automated and required “a good deal of insight’
- Our FCM based verification was fully automated and did not require any insight! Only natural formalization (encoding) in FO is required.

What about MIU reachable words?

- $MIUI \in L_{MIU}$

What about MIU reachable words?

- $MIUI \in L_{MIU}$
- Can we show this automatically?

What about MIU reachable words?

- $MIUI \in L_{MIU}$
- Can we show this automatically?
- Yes, we can, by the first-order *proving*. Let us see the demonstration.

Reachability as deducibility

- Many problems in verification can be naturally formulated in terms of *reachability* within transition systems;
- We propose to use deducibility (or derivability) in first-order predicate logic to model reachability in transition systems of interest;
- Then verification can be treated as theorem (dis)proving in classical predicate logic;
- Many automated tools (provers and model finders) are readily available.

Reachability as deducibility

- Let $\mathcal{S} = \langle S, \rightarrow \rangle$ be a transition system with the set of states S and transition relation $\rightarrow$
- Let $e : s \mapsto \varphi_s$ be encoding of states of $\mathcal{S}$ by formulae of first-order predicate logic, such that
  - the state s' is reachable from s, i.e. $s \rightarrow^* s'$ if and only if $\varphi_{s'}$ is the logical consequence of φ_s, that is $\varphi_s \models \varphi_{s'}$ and $\varphi_s \vdash \varphi_{s'}$.
- Under such assumptions:
  - Establishing reachability $\equiv$ theorem proving
  - Establishing non-reachability $\equiv$ theorem disproving

Disproving: Verification of safety

- Safety $\equiv$ non-reachability of "bad" states
- Verification of safety properties $\equiv$ theorem disproving
- To disprove $\varphi \models \psi$ it is sufficient to a find a countermodel for $\varphi \rightarrow \psi$, or which is the same a model for $\varphi \wedge \neg\psi$
- In general, such a model can be inevitably infinite and the set of satisfiable first-order formulae is not r.e.
- One can not hope for full automation here
- Our proposal: use automated finite model finders/builders

Remarks

- For the verification of safety the weaker assumption on the encoding is sufficient:
  - $s \rightarrow^* s' \Rightarrow \varphi_s \vdash \varphi_{s'}$
- For the verification of parameterized systems general idea of reachability as deducibility should be suitably adjusted
  - depends on particular classes of systems
  - unary or binary predicates modeling reachabiity can be used

Origins

- The idea of using finite model finders for verification is not new (thanks to anonymous referees of FMCAD 2010 conference!)
- It was proposed and developed in the area of verification of security protocols in the following papers (at least):
  - C. Weidenbach Towards an Automatic Analysis of Security Protocols in First-Order Logic, in H. Ganzinger (Ed.): CADE-16, LNAI 1632, pp. 314–328, 1999.
  - Selinger, P.: Models for an adversary-centric protocol logic. Electr. Notes Theor. Comput. Sci. 55(1) (2001);
  - Goubault-Larrecq, J.: Towards producing formally checkable security proofs, automatically. In: Computer Security Foundations (CSF), pp. 224[U+FFFD]238 (2008)
  - Jan Jurjens and Tjark Weber, Finite Models in FOL-Based Crypto-Protocol Verification. Foundations and Applications of Security Analysis, LNCS 5511, 2009.

Further developements

AL (2009-. . .)

- Countermodel finding based verification methods are practically efficient for the verification of various classes of infinite state and parameterized systems:
  - lossy channel systems
  - cache coherence protocols
  - parameterized linear arrays of finite state automata
  - general term rewriting systems
  - etc.
- Completeness (for lossy channel systems verification)
- Relative completeness wrt to regular model checking (RMC); regular tree model checking (RTMC); tree automata completion techniques
- Generic MACE4 finite model finder by W.McCune has been successfully used to verify above systems

Case Study I: Parameterized mutual exclusion protocol

- Taken from the paper Parosh Aziz Abdulla, Giorgio Delzanno, Noomene Ben Henda, Ahmed Rezine. Monotonic Abstraction: on Efficient Verification of Parameterized Systems. *Int. J. Found. Comput. Sci.* 20(5): 779-801 (2009)
- Operates on the parameterized linear array of finite state automata

Protocol specification

The protocol is specified as a parameterized system $\mathcal{ME} = (Q, T)$, where $Q = \{green, black, blue, red\}$ is the set of local states of finite automata, and T consists of the following transitions:

- $\forall_{LR}\{green, black\} : green \to black$
- $black \to blue$
- $\exists_L\{black, blue, red\} : blue \to blue$
- $\forall_L\{green\} : blue \to red$
- $red \to black$
- $black \to green$

The correctness condition: if the protocol starts with all states being *green* it will never get to a state where there are two or more automata in the *red* state

Translation to the first-order logic,I

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

  ($*$ is a monoid operation and e is a unit of a monoid)

- $G(e)$
- $G(x) \rightarrow G(x * green)$

  (specification of configurations with all green states)

- $GB(e)$
- $GB(x) \rightarrow GB(x * green)$
- $GB(x) \rightarrow GB(x * black)$

  (specification of configurations with all states being green or black)

Translation to the first-order logic,II

- $G(x) \to R(x)$

  (initial states assumption: "allgreen" configurations are reachable)

- $(R((x * green) * y)\ \&\ GB(x)\ \&\ GB(y)) \to R((x * black) * y)$
- $R((x * black) * y) \to R((x * blue) * y)$
- $R((x * blue) * y)\ \&\ (x = (z * black) * w) \to R((x * blue) * y)$
- $R((x * blue) * y)\ \&\ (x = (z * blue) * w) \to R((x * blue) * y)$
- $R((x * blue) * y)\ \&\ (x = (z * red) * w) \to R((x * blue) * y)$
- $R((x * blue) * y)\ \&\ G(x) \to R((x * red) * y)$
- $R((x * red) * y) \to R((x * black) * y)$
- $R((x * black) * y) \to R((x * green) * y)$

  (specification of reachability by one step transitions from T; one formula per transition, except the case with existential condition, where three formulae are used)

Adequacy of encoding and Verification

- If a configuration $\bar{c}$ is reachable in $\mathcal{ME}$ then $\Phi_{\mathcal{P}} \vdash R(t_{\bar{c}})$
- To establish safety property of the protocol (mutual exclusion) it does suffice to show that $\Phi_{\mathcal{P}} \not\vdash \exists x \exists y \exists z R((((x * red) * y) * red) * z)$.
- Delegate the latter task to the finite model finder MACE4 (see demonstration)

Adequacy of encoding and Verification

- If a configuration $\bar{c}$ is reachable in $\mathcal{ME}$ then $\Phi_{\mathcal{P}} \vdash R(t_{\bar{c}})$
- To establish safety property of the protocol (mutual exclusion) it does suffice to show that $\Phi_{\mathcal{P}} \not\vdash \exists x \exists y \exists z R((((x * red) * y) * red) * z)$.
- Delegate the latter task to the finite model finder MACE4 (see demonstration)
- It takes approx. 0.01s to find a countermodel and verify the safety property!

Countermodel as Invariant

- Take a configuration $\bar{c}$ of the protocol, consider its term representation $t_{\bar{c}}$
- The following property is an invariant of the system:

$$[t_{\bar{c}}] \in [R]$$

Here $[\ldots]$ denote the interpretation in the (counter)model.

Model and Invariant

The domain D of the model is a four element set $\{0, 1, 2, 3\}$.
Interpretations of constants: $[black] = [blue] = 0$, $[e] = [green] = 1$, $[red] = 2$. Interpretations of unary predicates: $[G] = \{1\}$; $[GB] = \{0, 1\}$; $[R] = \{0, 1, 2\}$.
The interpretation of the binary function $*$ is given by the following table

	0	1	2	3
0	0	0	2	3
1	0	1	2	3
2	2	2	3	3
3	3	3	3	3

Invariant property which holds for any reachable configuration $\bar{c}$:

$$[t_{\bar{c}}] \in [R] = \{0, 1, 2\}$$

Relative completeness

Theorem (2010)

If the safety of parameterized linear system of automata can be demonstrated by monotonic abstraction method then it can be demonstrated by FCM too.

FCM is stronger than monotonic abstraction

The parameterized system (Q, T) where $Q = \{q_0, q_1, q_2, q_3, q_4\}$ and where T includes the following transition rules

1. $\forall_{LR}\{q_0, q_1, q_4\} : q_0 \to q_1$
2. $q_1 \to q_2$
3. $\forall_{L}\{q_0\} : q_2 \to q_3$
4. $q_3 \to q_0$
5. $\exists_{LR}\{q_2\} : q_3 \to q_4$
6. $q_4 \to q_0$

satisfies mutual exclusion for state q_4, but this fact *can not* be established by the monotonic abstraction method.
Using FCM we have verified mutual exclusion for this system, demonstrating that FCM method is stronger than monotone abstraction.
Mace4 has found a finite countermodel of the size 6 in 341s.

Further relative completeness results

Theorem (2010)

If the safety of a linear parameterized system can be demonstrated by regular model checking method then it can be demonstrated by FCM too.

Further relative completeness results

Theorem (2010)

If the safety of a linear parameterized system can be demonstrated by regular model checking method then it can be demonstrated by FCM too.

Theorem (2011)

If the safety of a tree-shape parameterized system can be demonstrated by regular tree model checking method then it can be demonstrated by FCM too.

Further relative completeness results

Theorem (2010)

If the safety of a linear parameterized system can be demonstrated by regular model checking method then it can be demonstrated by FCM too.

Theorem (2011)

If the safety of a tree-shape parameterized system can be demonstrated by regular tree model checking method then it can be demonstrated by FCM too.

Theorem (2011, RTA 2012)

If the safety of a term rewriting system can be demonstrated by tree automata completion technique then it can be demonstrated by FCM too.

Why does it work?

In all cases the proofs of relative completeness results rely upon existence of regular invariants, that is regular sets (of words or trees) subsuming all reachable states and disjoint with all unsafe states.

Beyond FCM: limitations of the method

- Can we always apply FCM to establish safety?

Beyond FCM: limitations of the method

- Can we always apply FCM to establish safety?
- No. Here is an example: consider TRS (term rewriting system):
  - $f(x, y) \leftrightarrow f(g(x), g(y))$
  - $f(a, g(x)) \to a$
  - $f(g(x), a) \to a$
- Is it true that $f(a, a) \not\to^* a$?

Beyond FCM: limitations of the method

- Can we always apply FCM to establish safety?
- No. Here is an example: consider TRS (term rewriting system):
  - $f(x, y) \leftrightarrow f(g(x), g(y))$
  - $f(a, g(x)) \rightarrow a$
  - $f(g(x), a) \rightarrow a$
- Is it true that $f(a, a) \not\rightarrow^{*} a$? Yes! But this can not be established by FCM, for there is no a regular invariant here separating reachable terms and a!

Beyond FCM: limitations of the method

- Can we always apply FCM to establish safety?
- No. Here is an example: consider TRS (term rewriting system):
  - $f(x, y) \leftrightarrow f(g(x), g(y))$
  - $f(a, g(x)) \rightarrow a$
  - $f(g(x), a) \rightarrow a$
- Is it true that $f(a, a) \not\rightarrow^{*} a$? Yes! But this can not be established by FCM, for there is no a regular invariant here separating reachable terms and a!
- **Challenge:** Extend the method to infinite countermodels! FCM $\rightarrow$ ICM

Observations on FCM

- We presented FCM method for safety verification of infinite state and parameterized systems

Observations on FCM

- We presented FCM method for safety verification of infinite state and parameterized systems
- FCM is simple

Observations on FCM

- We presented FCM method for safety verification of infinite state and parameterized systems
- FCM is simple
- FCM is at least as powerful as methods based on monotonic abstraction, RMC, RTMC, tree automata completion techniques in establishing safety

Observations on FCM

- We presented FCM method for safety verification of infinite state and parameterized systems
- FCM is simple
- FCM is at least as powerful as methods based on monotonic abstraction, RMC, RTMC, tree automata completion techniques in establishing safety
- FCM is efficient in practice (in many cases)

Observations on FCM

- We presented FCM method for safety verification of infinite state and parameterized systems
- FCM is simple
- FCM is at least as powerful as methods based on monotonic abstraction, RMC, RTMC, tree automata completion techniques in establishing safety
- FCM is efficient in practice (in many cases)

Part 2: Applications to Mathematics

Groups and their presentations

- Groups are algebraic structures which satisfy the following axioms
  - $(x \cdot y) \cdot z = x \cdot (y \cdot z)$
  - $x \cdot e = x$
  - $e \cdot x = x$
  - $x \cdot x' = e$

Groups and their presentations

- Groups are algebraic structures which satisfy the following axioms
  - $(x \cdot y) \cdot z = x \cdot (y \cdot z)$
  - $x \cdot e = x$
  - $e \cdot x = x$
  - $x \cdot x' = e$
- Groups can be defined in different ways, including by **presentations** $\langle x_1, \ldots, x_n; r_1, \ldots, r_m \rangle$, where $x_1, \ldots, x_n$ are *generators* and $r_1, \ldots r_m$ are *relators*

Groups and their presentations

- Groups are algebraic structures which satisfy the following axioms
  - $(x \cdot y) \cdot z = x \cdot (y \cdot z)$
  - $x \cdot e = x$
  - $e \cdot x = x$
  - $x \cdot x' = e$
- Groups can be defined in different ways, including by **presentations** $\langle x_1, \ldots, x_n; r_1, \ldots, r_m \rangle$, where $x_1, \ldots, x_n$ are *generators* and $r_1, \ldots r_m$ are *relators*
- Intuitively, the presentation above defines a group the elements of which are words in the alphabet $x_1, \ldots, x_n, x'_1, \ldots, x'_n$ taken up to the equivalence defined by $r_1 = e, \ldots, r_m = e$

Trivial group presentations

- $\langle a, b \mid ab, b \rangle$ (trivial example of the trivial group presentation)

Trivial group presentations

- $\langle a, b \mid ab, b \rangle$ (trivial example of the trivial group presentation)
- $\langle a, b \mid abab'a'b', aaab'b'b'b'$ (not so trivial example of the trivial groups presentation)

For a group presentation $\langle x_1, \ldots, x_n; r_1, \ldots r_m \rangle$ with generators x_i, and relators r_j, consider the following transformations.

- AC1 Replace some r_i by r_i^{-1}.
- AC2 Replace some r_i by $r_i \cdot r_j$, $j \neq i$.
- AC3 Replace some r_i by $w \cdot r_i \cdot w^{-1}$ where w is any word in the generators.

- Two presentations g and g' are called *Andrews-Curtis equivalent (AC-equivalent)* if one of them can be obtained from the other by applying a finite sequence of transformations of the types (AC1) - (AC3).
- A group presentation $g = \langle x_1, \ldots, x_n; r_1, \ldots r_m \rangle$ is called *balanced* if $n = m$, that is a number of generators is the same as a number of relators. Such n we call a *dimension* of g and denote by $Dim(g)$.

Conjecture (1965)

if $\langle x_1, \ldots, x_n; r_1, \ldots r_n \rangle$ is a balanced presentation of the trivial group it is AC-equivalent to the trivial presentation $\langle x_1, \ldots, x_n; x_1, \ldots x_n \rangle$.

Trivial Example

- $\langle a, b \mid ab, b \rangle \to \langle a, b \mid ab, b^{-1} \rangle \to \langle a, b \mid a, b^{-1} \rangle \to \langle a, b \mid a, b \rangle$

AC-conjecture: short profile

- AC-conjecture is open

AC-conjecture: short profile

- AC-conjecture is open
- AC-conjecture may well be false (prevalent opinion of experts?)

AC-conjecture: short profile

- AC-conjecture is open
- AC-conjecture may well be false (prevalent opinion of experts?)
- Series of potential counterexamples; smallest for which simplification is unknown is AK-3: $\langle x, y | xyxy^{-1}x^{-1}y^{-1}, x^3y^{-4} \rangle$

AC-conjecture: short profile

- AC-conjecture is open
- AC-conjecture may well be false (prevalent opinion of experts?)
- Series of potential counterexamples; smallest for which simplification is unknown is AK-3: $\langle x, y | xyxy^{-1}x^{-1}y^{-1}, x^3y^{-4} \rangle$
- How to find simplifications, algorithmically?

AC-conjecture: short profile

- AC-conjecture is open
- AC-conjecture may well be false (prevalent opinion of experts?)
- Series of potential counterexamples; smallest for which simplification is unknown is AK-3: $\langle x, y | xyxy^{-1}x^{-1}y^{-1}, x^3y^{-4} \rangle$
- How to find simplifications, algorithmically?
- If a simplification exists, it could be found by the exhaustive search/total enumeration (iterative deepening)
- The issue: simplifications could be very long (Bridson 2015; Lishak 2015)

Search of trivializations and elimination of counterexamples

- Genetic search algorithms (Miasnikov 1999; Swan et al. 2012)
- Breadth-First search (Havas-Ramsay, 2003; McCaul-Bowman, 2006)
- Todd-Coxeter coset enumeration algorithm (Havas-Ramsay,2001)
- Generalized moves and strong equivalence relations (Panteleev-Ushakov, 2016)
- ...

Search of trivializations and elimination of counterexamples

- Genetic search algorithms (Miasnikov 1999; Swan et al. 2012)
- Breadth-First search (Havas-Ramsay, 2003; McCaul-Bowman, 2006)
- Todd-Coxeter coset enumeration algorithm (Havas-Ramsay,2001)
- Generalized moves and strong equivalence relations (Panteleev-Ushakov, 2016)
- . . .

Our approach: apply generic automated FO reasoning instead of specialized algorithms
Our Claim: generic automated reasoning is (very) competitive

ACT rewriting system, dim =2

Equational theory of groups T_G:

- $(x \cdot y) \cdot z = x \cdot (y \cdot z)$
- $x \cdot e = x$
- $e \cdot x = x$
- $x \cdot r(x) = e$

For each $n \geq 2$ we formulate a term rewriting system modulo T_G, which captures AC-transformations of presentations of dimension n.
For an alphabet $A = \{a_1, a_2\}$ a term rewriting system ACT_2 consists the following rules:

$$\begin{array}{rl} \text{R1L} & f(x, y) \to f(r(x), y)) \\ \text{R1R} & f(x, y) \to f(x, r(y)) \\ \text{R2L} & f(x, y) \to f(x \cdot y, y) \\ \text{R2R} & f(x, y) \to f(x, y \cdot x) \\ \text{R3L}_i & f(x, y) \to f((a_i \cdot x) \cdot r(a_i), y) \text{ for } a_i \in A, i = 1, 2 \\ \text{R3R}_i & f(x, y) \to f(x, (a_i \cdot y) \cdot r(a_i)) \text{ for } a_i \in A, i = 1, 2 \end{array}$$

AC-transformations as rewriting modulo group theory

The rewrite relation $\rightarrow_{ACT/G}$ for ACT modulo theory T_G:
$t \rightarrow_{ACT/G} s$ iff there exist $t' \in [t]_G$ and $s' \in [s]_G$ such that $t' \rightarrow_{ACT} s'$.

Reduced ACT_2

Reduced term rewriting system $rACT_2$ consists of the following rules:

R1L $f(x, y) \to f(r(x), y))$

R2L $f(x, y) \to f(x \cdot y, y)$

R2R $f(x, y) \to f(x, y \cdot x)$

R3L$_i$ $f(x, y) \to f((a_i \cdot x) \cdot r(a_i), y)$ for $a_i \in A, i = 1, 2$

Proposition

Term rewriting systems ACT_2 and $rACT_2$ considered modulo T_G are equivalent, that is $\to^*_{ACT_2/G}$ and $\to^*_{rACT_2/G}$ coincide.

Proposition

For ground t_1 and t_2 we have $t_1 \to^*_{ACT_2/G} t_2 \Leftrightarrow t_2 \to^*_{ACT_2/G} t_1$, that is $\to^*_{ACT_2/G}$ is symmetric.

Equational Translation

Denote by E_{ACT_2} an equational theory $T_G \cup rACT^=$ where $rACT^=$ includes the following axioms (equality variants of the above rewriting rules):

E-R1L $f(x, y) = f(r(x), y))$

E-R2L $f(x, y) = f(x \cdot y, y)$

E-R2R $f(x, y) = f(x, y \cdot x)$

E-R3L$_i$ $f(x, y) = f((a_i \cdot x) \cdot r(a_i), y)$ for $a_i \in A, i = 1, 2$

Proposition

For ground terms t_1 and t_2 $t_1 \rightarrow^*_{ACT_2/G} t_2$ iff $E_{ACT_2} \vdash t_1 = t_2$

A variant of the equational translation: replace the axioms $\mathbf{E-R3L_i}$ by "non-ground" axiom $\mathbf{E-RLZ}$: $f(x, y) = f((z \cdot x) \cdot r(z), y)$

Implicational Translation

Denote by I_{ACT_2} the first-order theory $T_G \cup rACT_2^{\rightarrow}$ where $rACT_2^{\rightarrow}$ includes the following axioms:

I-R1L $R(f(x,y)) \rightarrow R(f(r(x),y)))$

I-R2L $R(f(x,y)) \rightarrow R(f(x \cdot y, y))$

I-R2R $R(f(x,y)) \rightarrow R(f(x, y \cdot x))$

I-R3L$_i$ $R(f(x,y)) \rightarrow R(f((a_i \cdot x) \cdot r(a_i), y))$ for $a_i \in A, i = 1, 2$

Proposition

For ground terms t_1 *and* t_2 $t_1 \rightarrow^*_{ACT_2/G} t_2$ *iff* $I_{ACT_2} \vdash R(t_1) \rightarrow R(t_2)$

Higher Dimensions

- An equational translation for $n = 3$ ("non-ground" variant):

$$f(x, y, z) = f(r(x), y, z) \quad f(x, y, z) = f(x, r(y), z)$$
$$f(x, y, z) = f(x, y, r(z)) \quad f(x, y, z) = f(x \cdot y, y, z)$$
$$f(x, y, z) = f(x \cdot z, y, z) \quad f(x, y, z) = f(x, y \cdot x, z)$$
$$f(x, y, z) = f(x, y \cdot z, z) \quad f(x, y, z) = f(x, y, z \cdot x)$$
$$f(x, y, z) = f(x, y, z \cdot y) \quad f(x, y, z) = f((v \cdot x) \cdot r(v), y, z)$$
$$f(x, y, z) = f(x, (v \cdot y) \cdot r(v), z) \quad f(x, y, z) = f(x, y, (v \cdot z) \cdot r(v)).$$

Automated Reasoning for AC conjecture exploration

For any pair of presentations p_1 and p_2,
to establish whether they are AC-equivalent one can formulate and try to solve first-order theorem proving problems

- $E_{ACT_n} \vdash t_{p_1} = t_{p_2}$, or
- $I_{ACT_n} \vdash R(t_{p_1}) \rightarrow R(t_{p_2})$

OR, theorem disproving problems

- $E_{ACT_n} \not\vdash t_{p_1} = t_{p_2}$, or
- $I_{ACT_n} \not\vdash R(t_{p_1}) \rightarrow R(t_{p_2})$

Automated Reasoning for AC conjecture exploration

For any pair of presentations p_1 and p_2,
to establish whether they are AC-equivalent one can formulate and try to solve first-order theorem proving problems

- $E_{ACT_n} \vdash t_{p_1} = t_{p_2}$, or
- $I_{ACT_n} \vdash R(t_{p_1}) \rightarrow R(t_{p_2})$

OR, theorem disproving problems

- $E_{ACT_n} \not\vdash t_{p_1} = t_{p_2}$, or
- $I_{ACT_n} \not\vdash R(t_{p_1}) \rightarrow R(t_{p_2})$

Our proposal: apply automated reasoning: ATP and finite model building.

Elimination of potential counterexamples

- **Known cases:** We have applied automated theorem proving using Prover9 prover to confirm that all cases eliminated as potential counterexamples in all known literature can be eliminated by our method too.

Theorem Proving for AC-Simplifications (cont.)

New cases (from Edjvet-Swan, 2005-2010):

T14 $\langle a, b \mid ababABB, babaBAA \rangle$
T28 $\langle a, b \mid aabbbbABBBB, bbaaaaBAAAA \rangle$
T36 $\langle a, b \mid aababAABB, bbabaBBAA \rangle$
T62 $\langle a, b \mid aaabbAbABBB, bbbaaBaBAAA \rangle$
T74 $\langle a, b \mid aabaabAAABB, bbabbaBBBAA \rangle$

T16 $\langle a, b, c \mid ABCacbb, BCAbacc, CABcbaa \rangle$
T21 $\langle a, b, c \mid ABCabac, BCAbcba, CABcacb \rangle$
T48 $\langle a, b, c \mid aacbcABCC, bbacaBCAA, ccbabCABB \rangle$
T88 $\langle a, b, c \mid aacbAbCAB, bbacBcABC, ccbaCaBCA \rangle$
T89 $\langle a, b, c \mid aacbcACAB, bbacBABC, ccbaCBCA \rangle$

T96 $\langle a, b, c, d \mid adCADbc, baDBAcd, cbACBda, dcBDCab \rangle$
T97 $\langle a, b, c, d \mid adCAbDc, baDBcAd, cbACdBa, dcBDaCb \rangle$ [ICMS 2018]

AC-trivialization for **T16**

$\langle ABCacbb, BCAbacc, CABcbaa \rangle$

$\xrightarrow{x,y,z \to x,y,azA} \langle ABCacbb, BCAbacc, aCABcba \rangle$

$\xrightarrow{x,y,z \to x,y,zx} \langle ABCacbb, BCAbacc, aCABacbb \rangle$

$\xrightarrow{x,y,z \to x,y,bzB} \langle ABCacbb, BCAbacc, baCABacb \rangle$

$\xrightarrow{x,y,z \to x,y,zy} \langle ABCacbb, BCAbacc, bac \rangle$

$\xrightarrow{x,y,z \to x,y,czC} \langle ABCacbb, BCAbacc, cba \rangle$

$\xrightarrow{x,y,z \to x',y,z} \langle BBCAcba, BCAbacc, cba \rangle$

$\xrightarrow{x,y,z \to x,y,z'} \langle BBCAcba, BCAbacc, ABC \rangle$

$\xrightarrow{x,y,z \to xz,y,z} \langle BBCA, BCAbacc, ABC \rangle$

$\xrightarrow{x,y,z \to x',y,z} \langle acbb, BCAbacc, ABC \rangle \xrightarrow{x,y,z \to x,y,z'} \langle acbb, BCAbacc, cba \rangle$

$\xrightarrow{x,y,z \to x,y,azA} \langle acbb, BCAbacc, acb \rangle \xrightarrow{x,y,z \to x,y,z'} \langle acbb, BCAbacc, BCA \rangle$

$\xrightarrow{x,y,z \to x,y,zx} \langle acbb, BCAbacc, b \rangle \xrightarrow{x,y,z \to x,y,z'} \langle acbb, BCAbacc, B \rangle$

$\xrightarrow{x,y,z \to xz,y,z} \langle acb, BCAbacc, B \rangle \xrightarrow{x,y,z \to xz,y,z} \langle ac, BCAbacc, B \rangle$

AC-trivialization for **T16** (cont.)

$$\xrightarrow{x,y,z \to x,y',z} \langle ac, CCABacb, B\rangle \xrightarrow{x,y,z \to x,yz,z} \langle ac, CCABac, B\rangle$$

$$\xrightarrow{x,y,z \to x,y',z} \langle ac, CAbacc, B\rangle \xrightarrow{x,y,z \to x,y,z'} \langle ac, CAbacc, b\rangle$$

$$\xrightarrow{x,y,z \to x',y,z} \langle CA, CAbacc, b\rangle$$

$$\xrightarrow{x,y,z \to x,yx,z} \langle CA, CAbacA, b\rangle \xrightarrow{x,y,z \to x,y',z} \langle CA, aCABac, b\rangle$$

$$\xrightarrow{x,y,z \to x,yx,z} \langle CA, aCAB, b\rangle \xrightarrow{x,y,z \to x,yz,z} \langle CA, aCA, b\rangle$$

$$\xrightarrow{x,y,z \to x',y,z} \langle ac, aCA, b\rangle \xrightarrow{x,y,z \to x,yx,z} \langle ac, a, b\rangle$$

$$\xrightarrow{x,y,z \to x,y',z} \langle ac, A, b\rangle \xrightarrow{x,y,z \to x,yx,z} \langle ac, c, b\rangle$$

$$\xrightarrow{x,y,z \to x,y',z} \langle ac, C, b\rangle \xrightarrow{x,y,z \to xy,y,z} \langle a, C, b\rangle$$

$$\xrightarrow{x,y,z \to x,yz,z} \langle a, Cb, b\rangle \xrightarrow{x,y,z \to x,y',z} \langle a, Bc, b\rangle$$

$$\xrightarrow{x,y,z \to x,y,zy} \langle a, Bc, c\rangle \xrightarrow{x,y,z \to x,y,z'} \langle a, Bc, C\rangle$$

$$\xrightarrow{x,y,z \to x,yz,z} \langle a, B, C\rangle \xrightarrow{x,y,z \to x,y,z'} \langle a, B, c\rangle$$

$$\xrightarrow{x,y,z \to x,y',z} \langle a, b, c\rangle$$

What about automated disproving?

Proposition

To simplify AK-3 (if at all it is possible) one really needs conjugation with both generators a and b.

Mace4 finite model builder finds countermodels of sizes 12 and 6 for the cases where either of the conjugation rules is omitted.

What about automated disproving?

Proposition

To simplify AK-3 (if at all it is possible) one really needs conjugation with both generators a and b.

Mace4 finite model builder finds countermodels of sizes 12 and 6 for the cases where either of the conjugation rules is omitted.

Can we disprove AC-conjecture by building a finite countermodel witnessing non-trivialization for one of the open cases (e.g. AK-3)?

What about automated disproving?

Proposition

To simplify AK-3 (if at all it is possible) one really needs conjugation with both generators a and b.

Mace4 finite model builder finds countermodels of sizes 12 and 6 for the cases where either of the conjugation rules is omitted.

Can we disprove AC-conjecture by building a finite countermodel witnessing non-trivialization for one of the open cases (e.g. AK-3)?

No, unfortunately (Borovik et al, *The Finitary Andrews-Curtis Conjecture*, 2005)

What about automated disproving?

Proposition

To simplify AK-3 (if at all it is possible) one really needs conjugation with both generators a and b.

Mace4 finite model builder finds countermodels of sizes 12 and 6 for the cases where either of the conjugation rules is omitted.

Can we disprove AC-conjecture by building a finite countermodel witnessing non-trivialization for one of the open cases (e.g. AK-3)?

No, unfortunately (Borovik et al, *The Finitary Andrews-Curtis Conjecture*, 2005)
We need to search for infinite countermodels to disprove AC-conjecture!

Observations on ATP for AC-conjecture

- Automated Proving and Disproving is an interesting and powerful approach to AC-conjecture exploration;
- AC-conjecture is a source of interesting challenging problems for ATP/ATD;

Time to prove simplifications

	T14	T28	T36	T62	T74	T16	T21	T48	T88	T89	T96	97
Dim	2	2	2	2	2	3	3	3	3	3	4	4
Equational	6.02s	6.50s	7.18s	24.34s	57.17s	12.87s	11.98s	34.63s	57.69s	17.50s	114.05s	115.10s
Implicational	1.57s	2.46s	1.34s	22.50s	6.29s	1.61s	1.45s	2.17s	1.97s	2.14s	102.34s	89.65s
Implicational GC	t/o	t/o	t/o	t/o	t/o	3.76s	1.61s	t/o	0.86s	0.75s	t/o	t/o

"t/o" stands for timeout in 200s; "GC" means encoding with ground conjugation rules; all other encodings are with non-ground conjugation rules.

Conclusions

- Formalization of reachability using FO is simple and powerful method
  - FO disproving can used to establish safety (non-reachability) properties
  - FO proving can be used to search for paths in complex domains

Conclusions

- Formalization of reachability using FO is simple and powerful method
  - FO disproving can used to establish safety (non-reachability) properties
  - FO proving can be used to search for paths in complex domains
- Formalization in FO + Automated Reasoning can be competitive wrt specialised algorithms

Conclusions

- Formalization of reachability using FO is simple and powerful method
  - FO disproving can used to establish safety (non-reachability) properties
  - FO proving can be used to search for paths in complex domains
- Formalization in FO + Automated Reasoning can be competitive wrt specialised algorithms

Thank you!