Outline

• Previous work on cvc5’s proof architecture

• Introducing: AletheLF (ALF) proof format
  • Standard format for proofs from cvc5

• alfc: a proof checker for ALF

• Preliminary results and future work
cvc5’s internal proof architecture
Architecture of cvc5 + Proofs

- Input F
  - Preprocessor
    - CNF+Preprocess Proof
      - Proof Sketch (resolution+t-lemmas)
        - Proof (internal)
          - Proof Converter X
          - Proof Printer X
            - Proof (format X)
  - Theory Solvers
  - SAT Solver

- SAT Solver
  - Preprocessor
    - CNF+Preprocess Proof
      - Proof Sketch (resolution+t-lemmas)
        - Proof (internal)
          - Proof Converter X
          - Proof Printer X
            - Proof (format X)

- cvc5
  - SAT Solver
  - Theory Solvers
  - Preprocessor
  - CNF+Preprocess Proof
  - Proof Sketch (resolution+t-lemmas)
  - Proof (internal)
  - Proof Converter X
  - Proof Printer X
  - Proof (format X)

- Proof Checker X
cvc5 + Proofs

Flexible Proof Production in an Industrial-Strength SMT Solver

Haniel Barbosa¹, Andrew Reynolds², Gereon Kremer³, Hanna Lachnitt³, Aina Niemetz³, Andres Nötzli³, Alex Ozdemir³, Mathias Preiner³, Arjun Viswanathan², Scott Viteri³, Yoni Zohar⁴, Cesare Tinelli², Clark Barrett³

¹ Universidade Federal de Minas Gerais, Belo Horizonte, Brasil
² The University of Iowa, Iowa City, USA
³ Stanford University, Stanford, USA
⁴ Bar-Ilan University, Ramat Gan, Israel

- Internal proof calculus
  - Roughly 142 rules (132 core + 10 macro)
  - Native proof checker in cvc5’s core
  - Original focus was on theory of strings

- Evaluated on many SMT-LIB theories Barbosa et al IJCAR22
- For more details, see Barbosa et al CACM23
cvc5 + Proofs

• Instrumented *many parts of system*:
  • Preprocessing
    • Boolean circuit propagation
  • Rewriting
  • SAT solver
  • CNF conversion
  • Quantifier instantiation
    • Used for reasoning about extended string functions
  • UF Theory Solver / congruence closure
  • Linear Arithmetic Solver
  • Theory Combination
  • Strings Theory Solver
    • Regular expression unfolding
    • Extended function reductions
    • Core calculus (Liang et al CAV 2014)
Rules (51) used in 4.5M Zelkova Problems

<table>
<thead>
<tr>
<th>Rule Name</th>
<th>Frequency</th>
<th>Other Rule Name</th>
<th>Frequency</th>
<th>Rule Name</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>AND_ELIM</td>
<td>83866713</td>
<td>CONG</td>
<td>174513997</td>
<td>NOT_OR_ELIM</td>
<td>265133</td>
</tr>
<tr>
<td>AND_INTRO</td>
<td>32521977</td>
<td>CONTRA</td>
<td>198795</td>
<td>RE_INTER</td>
<td>62</td>
</tr>
<tr>
<td>ARITH_MULT_NEG</td>
<td>88</td>
<td>EQ_RESOLVE</td>
<td>50749239</td>
<td>RE_UNFOLD_POS</td>
<td>886262</td>
</tr>
<tr>
<td>ARITH_MULT_POS</td>
<td>88</td>
<td>EQUIV_ELIM1</td>
<td>1866156</td>
<td>REFL</td>
<td>116156677</td>
</tr>
<tr>
<td>ARITH_POLY_NORM</td>
<td>517</td>
<td>EQUIV_ELIM2</td>
<td>199786</td>
<td>REMOVE_TERM_FORMULA</td>
<td>44</td>
</tr>
<tr>
<td>ARITH_SUM_UB</td>
<td>88</td>
<td>EVALUATE</td>
<td>28201002</td>
<td>REORDERING</td>
<td>2811609</td>
</tr>
<tr>
<td>ASSUME</td>
<td>83889221</td>
<td>FACTORING</td>
<td>245432</td>
<td>RESOLUTION</td>
<td>2830376</td>
</tr>
<tr>
<td>BV_BITBLAST_STEP</td>
<td>660</td>
<td>FALSE_ELIM</td>
<td>1041313</td>
<td>SCOPE</td>
<td>3031886</td>
</tr>
<tr>
<td>CHAIN_RESOLUTION</td>
<td>3973499</td>
<td>FALSE_INTRO</td>
<td>208289</td>
<td>SKOLEM_INTRO</td>
<td>265</td>
</tr>
<tr>
<td>CNF_AND_NEG</td>
<td>113770</td>
<td>IMPLIES_ELIM</td>
<td>1052199</td>
<td>STRING_EAGER_REDUCTION</td>
<td>6</td>
</tr>
<tr>
<td>CNF_AND_POS</td>
<td>3105408</td>
<td>ITE_ELIM1</td>
<td>44</td>
<td>SYMM</td>
<td>35719138</td>
</tr>
<tr>
<td>CNF_EQUIV_POS1</td>
<td>45034</td>
<td>ITE_ELIM2</td>
<td>44</td>
<td>TRANS</td>
<td>71642174</td>
</tr>
<tr>
<td>CNF_EQUIV_POS2</td>
<td>32532</td>
<td>MODUS_PONENS</td>
<td>1041495</td>
<td>TRUE_ELIM</td>
<td>1312434</td>
</tr>
<tr>
<td>CNF_IMPONES_NEG1</td>
<td>6316</td>
<td>NOT_AND</td>
<td>861574</td>
<td>TRUE_INTRO</td>
<td>37</td>
</tr>
<tr>
<td>CNF_IMPONES_NEG2</td>
<td>6507</td>
<td>NOT_IMPONES_ELIM1</td>
<td>47603</td>
<td>TRUST_PREPROCESS</td>
<td>545</td>
</tr>
<tr>
<td>CNF_OR_NEG</td>
<td>121563</td>
<td>NOT_IMPONES_ELIM2</td>
<td>55869</td>
<td>TRUST_THEORY_INERENCE</td>
<td>655</td>
</tr>
<tr>
<td>CNF_OR_POS</td>
<td>135584</td>
<td>NOT_NOT_ELIM</td>
<td>107846</td>
<td>TRUST_THEORY_REWRITE</td>
<td>86215359</td>
</tr>
</tbody>
</table>
Rules (51) used in 4.5M Zelkova Problems

<table>
<thead>
<tr>
<th>Rule Name</th>
<th>Count</th>
<th>Rule Name</th>
<th>Count</th>
<th>Rule Name</th>
<th>Count</th>
</tr>
</thead>
<tbody>
<tr>
<td>AND_ELIM</td>
<td>83866713</td>
<td>CONG</td>
<td>174513997</td>
<td>NOT_OR_ELIM</td>
<td>265133</td>
</tr>
<tr>
<td>AND_INTRO</td>
<td>32521977</td>
<td>CONTRA</td>
<td>198795</td>
<td>RE_INTER</td>
<td>62</td>
</tr>
<tr>
<td>ARITH_MULT_NEG</td>
<td>88</td>
<td>EQ_RESOLVE</td>
<td>50749239</td>
<td>RE_UNFOLD_POS</td>
<td>886262</td>
</tr>
<tr>
<td>ARITH_MULT_POS</td>
<td>88</td>
<td>EQUIV_ELIM1</td>
<td>1866156</td>
<td>REFL</td>
<td>116156677</td>
</tr>
<tr>
<td>ARITH_POLY_NORM</td>
<td>517</td>
<td>EQUIV_ELIM2</td>
<td>199786</td>
<td>REMOVE_TERM_FORMULA</td>
<td>44</td>
</tr>
<tr>
<td>ARITH_SUM_UB</td>
<td>88</td>
<td>EVALUATE</td>
<td>28201002</td>
<td>REORDERING</td>
<td>2811609</td>
</tr>
<tr>
<td>ASSUME</td>
<td>83889221</td>
<td>FACTORING</td>
<td>245432</td>
<td>RESOLUTION</td>
<td>2830376</td>
</tr>
<tr>
<td>BV_BITBLAST_STEP</td>
<td>660</td>
<td>FALSE_ELIM</td>
<td>1041313</td>
<td>SCOPE</td>
<td>3031886</td>
</tr>
<tr>
<td>CHAIN_RESOLUTION</td>
<td>3973499</td>
<td>FALSE_INTRO</td>
<td>208289</td>
<td>SKOLEM_INTRO</td>
<td>265</td>
</tr>
<tr>
<td>CNF_AND_NEG</td>
<td>113770</td>
<td>IMPLIES_ELIM</td>
<td>1052199</td>
<td>STRING_EAGER_REDUCTION</td>
<td>6</td>
</tr>
<tr>
<td>CNF_AND_POS</td>
<td>3105408</td>
<td>ITE_ELIM1</td>
<td>44</td>
<td>SYMM</td>
<td>35719138</td>
</tr>
<tr>
<td>CNF_EQUIV_POS1</td>
<td>45034</td>
<td>ITE_ELIM2</td>
<td>44</td>
<td>TRANS</td>
<td>71642174</td>
</tr>
<tr>
<td>CNF_EQUIV_POS2</td>
<td>32532</td>
<td>MODUS_PONENS</td>
<td>1041495</td>
<td>TRUE_ELIM</td>
<td>1312434</td>
</tr>
<tr>
<td>CNF_IMPLIES_NEG1</td>
<td>6316</td>
<td>NOT_AND</td>
<td>861574</td>
<td>TRUE_INTRO</td>
<td>37</td>
</tr>
<tr>
<td>CNF_IMPLIES_NEG2</td>
<td>6507</td>
<td>NOT_IMPLIES_ELIM1</td>
<td>47603</td>
<td>TRUST_PRESPROCESS</td>
<td>545</td>
</tr>
<tr>
<td>CNF_OR_NEG</td>
<td>121563</td>
<td>NOT_IMPLIES_ELIM2</td>
<td>55869</td>
<td>TRUST_THEORY_INFERENC</td>
<td>655</td>
</tr>
<tr>
<td>CNF_OR_POS</td>
<td>135584</td>
<td>NOT_NOT_ELIM</td>
<td>107846</td>
<td>TRUST_THEORY_REWRITE</td>
<td>86215359</td>
</tr>
</tbody>
</table>

...can elaborate via rewrite DSL
Proofs for Rewrites in cvc5
RARE: a DSL for Proofs of Rewrites

- Introduced DSL (RARE) for smt2 rewrites
- Automatic proof elaboration from user specification
- Developed rewrites, focusing on strings
  Noetzli et al FMCAD22
- Under submission to TACAS 24:
  Lachnitt et al, “Automatic Verification of SMT Rewrites in Isabelle/HOL”
cvc5 + Proofs + RARE

Input F
Preprocessor
Preprocessing Proof
Proof Sketch
Postprocessor
Proof (internal)
Proof Converter X
Proof Printer X
Rewrites (format X)
Proof (format X)
Proof Checker X

Input F'
Theory Solvers
SAT Solver
Rewrites (RARE)
IsaRare

Rewrites (RARE)
Rules (51) used in 4.5M Zelkova Problems

<table>
<thead>
<tr>
<th>Rule Name</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>AND_ELIM</td>
<td>83866713</td>
</tr>
<tr>
<td>AND_INTRO</td>
<td>32521977</td>
</tr>
<tr>
<td>ARITH_MULT_NEG</td>
<td>88</td>
</tr>
<tr>
<td>ARITH_MULT_POS</td>
<td>88</td>
</tr>
<tr>
<td>ARITH_POLY_NORM</td>
<td>517</td>
</tr>
<tr>
<td>ARITH_SUM_UB</td>
<td>88</td>
</tr>
<tr>
<td>ASSUME</td>
<td>83889221</td>
</tr>
<tr>
<td>BV_BITBLAST_STEP</td>
<td>660</td>
</tr>
<tr>
<td>CHAIN_RESOLUTION</td>
<td>3973499</td>
</tr>
<tr>
<td>CNF_AND_NEG</td>
<td>113770</td>
</tr>
<tr>
<td>CNF_AND_POS</td>
<td>3105408</td>
</tr>
<tr>
<td>CNF_EQUIV_POS1</td>
<td>45034</td>
</tr>
<tr>
<td>CNF_EQUIV_POS2</td>
<td>32532</td>
</tr>
<tr>
<td>CNF_IMPLIES_NEG1</td>
<td>6316</td>
</tr>
<tr>
<td>CNF_IMPLIES_NEG2</td>
<td>6507</td>
</tr>
<tr>
<td>CNF_OR_NEG</td>
<td>121563</td>
</tr>
<tr>
<td>CNF_OR_POS</td>
<td>135584</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rule Name</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>CONG</td>
<td>174513997</td>
</tr>
<tr>
<td>CONTRA</td>
<td>198795</td>
</tr>
<tr>
<td>EQ_RESOLVE</td>
<td>50749239</td>
</tr>
<tr>
<td>EQUIV_ELIM1</td>
<td>1866156</td>
</tr>
<tr>
<td>EQUIV_ELIM2</td>
<td>199786</td>
</tr>
<tr>
<td>EVALUATE</td>
<td>28201002</td>
</tr>
<tr>
<td>FACTORING</td>
<td>245432</td>
</tr>
<tr>
<td>FALSE_ELIM</td>
<td>1041313</td>
</tr>
<tr>
<td>FALSE_INTRO</td>
<td>208289</td>
</tr>
<tr>
<td>FACTORIZING</td>
<td>245432</td>
</tr>
<tr>
<td>SCOPE</td>
<td>3031886</td>
</tr>
<tr>
<td>SKOLEM_INTRO</td>
<td>265</td>
</tr>
<tr>
<td>STRING_EAGER_REDUCTION</td>
<td>6</td>
</tr>
<tr>
<td>SYMM</td>
<td>35719138</td>
</tr>
<tr>
<td>SYM</td>
<td>35719183</td>
</tr>
<tr>
<td>TRANS</td>
<td>71642174</td>
</tr>
<tr>
<td>TRUE_ELIM</td>
<td>1312434</td>
</tr>
<tr>
<td>TRUE_INTRO</td>
<td>37</td>
</tr>
<tr>
<td>TRUST_PREPROCESS</td>
<td>545</td>
</tr>
<tr>
<td>TRUST_THEORY_INFERENC</td>
<td>655</td>
</tr>
<tr>
<td>TRUST_THEORY_REWRITE</td>
<td>5,085,132</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Rule Name</th>
<th>Frequency</th>
</tr>
</thead>
<tbody>
<tr>
<td>RE_UNFOLD_POS</td>
<td>886262</td>
</tr>
<tr>
<td>RE_INTER</td>
<td>62</td>
</tr>
<tr>
<td>RE_ORDERING</td>
<td>2830376</td>
</tr>
<tr>
<td>SCOPE</td>
<td>3031886</td>
</tr>
<tr>
<td>SCOPE</td>
<td>3031886</td>
</tr>
<tr>
<td>SCOPE</td>
<td>3031886</td>
</tr>
<tr>
<td>SCOPE</td>
<td>3031886</td>
</tr>
<tr>
<td>SCOPE</td>
<td>3031886</td>
</tr>
<tr>
<td>SCOPE</td>
<td>3031886</td>
</tr>
</tbody>
</table>

94.1% rewrites elaborated ... expands to 34 DSL rewrite rules (as of March 2023)
Next Step: Exporting cvc5 Proofs
Key question: ...which proof format X to Export?

Input F

Preprocessor

Input F'

Theory Solvers

SAT Solver

Preprocessing Proof

Proof Sketch

Postprocessor

Proof (internal)

Proof Converter X

Proof Printer X

Rewrites (format X)

Proof (format X)

Rewrites (RARE)

cvc5

Proof Checker X
Previous Work: **LFSC** for SMT proofs

- Based on Edinburgh Logical Framework (LF)
  - Extended with side conditions
- User defined proof systems
  - Oe et al SMT09, Stump et al FMSD13, Hadarean et al LPAR15, Katz et al FMCAD16

- Why this was a suboptimal solution:
  - Performance
  - Proof rules must encoded at a low level
  - Syntax for terms does not match SMTLIB
    - No native for SMTLIB values
  - Limited tooling support
Previous Work: **Alethe** for SMT proofs

- Proof format based on SMTLIB syntax
  - Proofs are lists of extended smt2 commands
- Also produced by SMT solver veriT
- Efficient checker (Carcara)
  
  Schurr et al PxTP 2021, Andreotti et al TACAS 2023

- Why this was a suboptimal solution:
  - Proof rules are hardcoded in pen-and-paper standard
    - Available only for limited theories
  - Cannot capture cvc5’s type system, proof calculus
  - Time consuming and error prone to extend to new rules

**Alethe: Towards a Generic SMT Proof Format**
(extended abstract)

Hans-Jörg Schurr
University of Lorraine, CNRS, Inria, and LORIA, Nancy, France
hans-jorg.schurr@inria.fr

Mathias Fleury
Johannes Kepler University Linz, Austria
mathias.fleury@jku.at

Haniel Barbosa
Universidade Federal de Minas Gerais, Belo Horizonte, Brazil
hbarbosa@dcc.ufmg.br

Pascal Fontaine
University of Liège, Belgium
pascal.fontaine@uliege.be

**Carcara: An efficient proof checker and elaborator for SMT proofs in the Alethe format**

Bruno Andreotti¹, Hanna Lachnitt², Haniel Barbosa¹

¹ Universidade Federal de Minas Gerais, Belo Horizonte, Brazil
² Stanford University, Stanford, USA
Other Related Proof Efforts

• DRAT/LRAT formats used in SAT community
  - Handles only the SAT portion of the proof

• Extensions to SMT theories, e.g. eDRAT
  - Does not handle proofs of preprocessing or rewriting

• Verified proof checking in Lean
  - Requires substantially more effort, expected to be slower
Introducing: The AletheLF Proof Format
Introducing: *AletheLF*

- **Flexibility** of LFSC
- **Syntax** of Alethe and SMTLIB v3.0
- **Granularity** of RARE for rewrite rules
- **Speed** of checkers for other formats e.g. DRAT via *oracles*

**Goals:**
- Clear definition of cvc5’s proof calculus
- Baseline proof checker (alfc)
- Intermediate step towards an efficient fully *verified* checking for SMT
What is AletheLF (ALF)?

- Proof format based on **SMTLIB version 3.0**
  - Commands for defining **theory signatures**
  - Commands for defining **proof rules**
  - Commands for writing **proofs**

  - Features: side conditions, builtin evaluation operators, oracles

  (declare-const and (→ Bool Bool Bool) :right-assoc-nil true)

  (declare-rule and_elim ((Fs Bool) (i Int))
   :premises (Fs)
   :args (i)
   :conclusion (alf.extract and Fs i))

  (assume p0 (and A B C))
  (step p1 B :rule and_elim :premises (p0) :args (1))
How have we used ALF so far?

• Translation of cvc5’s internal calculus to ALF format:
  • ~2400 LOC (*.smt3)
  • ~90 proof rules
    • Most common rules of cvc5’s internal calculus have been translated

• Includes:
  • Complete definition of cvc5’s theory symbols and its type system
  • Formalization of most internal symbols (skolems) introduced by cvc5
  • Proof rules covering:
    • CNF, resolution, equality, strings, arithmetic, arrays, quantifiers
    • Alternative: DRAT for SAT proofs
alfc: A Proof Checker for AletheELF
alfc: A Proof Checker for AletheLF

• Efficient proof checker for ALF
  • ~9300 LOC (C++)

• Incorporates:
  • A parser and lexer for smt3
  • Utilities for evaluation over SMTLIB values (integers, rationals, bitvectors, strings)
  • Implementation of core logic for type checking and evaluation

• Support for several key features:
  • Side conditions
  • Reference validation (reference “original.smt2”)
  • Oracle functions (declare-oracle-fun f () Int ./binary)
  • Signature compilation to C++

• Available at https://github.com/cvc5/alfc
  • User manual available

⇒ Also serves as a reference parser for SMTLIB version 3
Key Features of alfc
How do I know an ALF proof is for the right input?

- alfc accepts a reference command
- Checks that assume match assert in reference file

* cvc5 prints reference command if option --alf-print-reference is enabled
How can we leverage DRAT?

• cvc5 now uses configurable SAT solver
  • Must conform to standard interface (IPASIR-UP)
    Fazekas et al SAT 2023
  • Notably, the main branch of cvc5 supports CaDiCaL
    • Via option --sat-solver=cadical
  • Choice of SAT solver can dramatically improve solving times

⇒ ALF proofs can incorporate black-box (DRAT) proof output from a SAT solver
Incorporating ALF+DRAT

Input F

Preprocessor

Input F'

Theory Solvers

SAT Solver (Minsat)

Postprocessor

Proof Sketch
(resolution)

Proof (internal)

Proof Converter (ALF)

Proof Printer (ALF)

Rewrites (RARE)

Theory Sig (ALF)

Rewrites (ALF)

Proof (ALF)

cvc5

alfc
Incorporating ALF+DRAT (option 1)

**Input F**
- Preprocessor
- Theory Solvers
  - SAT Solver (CaDiCaL)

**Input F’**
- Preprocessor
- Theory Solvers
  - Theory Sig (ALF)

**Rewrites (RARE)**
- Proof Sketch
  - Preprocessing Proof

**Postprocessor**
- Proof (internal)
  - Proof Converter (ALF)
  - Proof Printer (ALF)

**Proof (ALF)**

**cvc5**
- Rewrites (ALF)
- Proof (ALF)

**DIMACS**
- Proof (DRAT)

**..via oracle**

**drat-trim**

**alfc**

**Proof Converter (ALF)**

**Proof Printer (ALF)**
alfc + Oracles

• alfc supports *oracle functions*
  • Syntax and semantics introduced in: Polgreen et al VMCAI 2022
• Command `declare-oracle-fun`

• Oracle funs are like `declare-fun` but have semantics implemented by external binary
  • Communication via smt2 (or smt3) text
Oracles for DRAT

1. A proof checker for DRAT (`drat-trim`)
2. A input verifier DIMACS↔smt2 (`drat-verify`, ~150 LOC C++)
Incorporating ALF+DRAT (option 2)

- **Preprocessor**
- **Theory Solvers**
- **SAT Solver**
- **Input F**
- **Input F’**
- **Proof Sketch** + reference to DIMACS
- **Preprocessing Proof**
- **Rewrites (RARE)**
- **Rewrites (ALF)**
- **Theory Sig (ALF)**
- **DIMACS**
- **Proof (internal)**
- **Proof Converter (ALF)**
- **Proof Printer (ALF)**
- **Proof (ALF)**
- **cvc5**
- **CaDiCaL+drat-trim**
- **DRAT**
- **...via oracle**
- **..input (minimized via unsat core) + theory lemmas**
Preliminary Evaluation
Results: alfc vs lfsc (on resolution proofs)

- 97348 benchmarks
- 60 second timeout
- All quantifier-free SMTLIB logics with
  - strings (S)
  - linear arithmetic (LIRA)
  - uninterpreted functions (UF)

⇒ alfc 1.56x faster proof checking than lfsc
  - Due to flexibility in alfc (e.g. uses chain resolution)
Results: alfc+DRAT vs alfc+resolution

⇒ DRAT scales better than res on harder examples
  • 1.34x faster for benchmarks >5 seconds
Results: LFSC vs alfc+DRAT, checking overhead

cvc5+minisat+proofs
(6.0x overhead)

cvc5+cadical+proofs
(2.3x overhead)
Results: alfc+DRAT vs Solving

⇒ 7.9x overhead solving+proof checking (UNSAT)
  • 1.4x for SAT problems
Future Work
Future Work

Input F
Preprocessor

Input F’
Preprocessor

Theory Solvers

SAT Solver

Rewrites (RARE)

Postprocessor

Proof (internal)

Proof Converter (ALF)

Proof Printer (ALF)

Proof (ALF)

Theory Sig (ALF)

User API

checker $X_1$

checker $X_n$

drat-trim

alfc

X

cvc5
Future Work

• Support current uses of Alethe in SMTCoq, Isabelle
• Formalize core logic of ALF checker in Agda
• Improvements to cvc5 performance
  • In particular, preprocessing and minimizing theory lemmas in DRAT
• Support further integrations with formats (LRAT), other checkers
Conclusion

• **cvc5** at [https://cvc5.github.io/](https://cvc5.github.io/)
  - Proof output now available on main via `--dump-proofs --proof-format=alf`
  - DRAT and RARE integrations available on dev branch, will be in v1.1

• ALF checker **alfc** available at [https://github.com/cvc5/alfc](https://github.com/cvc5/alfc)
  - Available via cvc5 repo `.contrib/get-alf-checker`