An Update on the OpenSMT Solver

16th Alpine Verification Meeting (AVM'24)

Tomáš Kolárik Martin Blicha Konstantin Britikov Natasha Sharygina

University of Lugano, Switzerland

September 6, 2024

OpenSMT

- First open-source SMT solver developed since 2008

OpenSMT

- First open-source SMT solver developed since 2008
- Successful state-of-the-art tool with unique features

OpenSMT

- First open-source SMT solver developed since 2008
- Successful state-of-the-art tool with unique features
- Focus on **QF** logics: **LA**, **UF**, **AX**, and their combinations
  - Specialized solver for difference logics

OpenSMT

- First open-source SMT solver developed since 2008
- Successful state-of-the-art tool with unique features
- Focus on **QF** logics: **LA**, **UF**, **AX**, and their combinations
  - Specialized solver for difference logics
- Resolution and DRAT **proofs** of unsatisfiability

$\rightarrow$ Certification of the results

OpenSMT

- First open-source SMT solver developed since 2008
- Successful state-of-the-art tool with unique features
- Focus on **QF** logics: **LA**, **UF**, **AX**, and their combinations
  - Specialized solver for difference logics
- Resolution and DRAT **proofs** of unsatisfiability

$\rightarrow$ Certification of the results

- Support of **unsatisfiable cores**

$\rightarrow$ Based on resolution proofs

OpenSMT

- First open-source SMT solver developed since 2008
- Successful state-of-the-art tool with unique features
- Focus on **QF** logics: **LA**, **UF**, **AX**, and their combinations
  - Specialized solver for difference logics
- Resolution and DRAT **proofs** of unsatisfiability

$\rightarrow$ Certification of the results
- Support of **unsatisfiable cores**

$\rightarrow$ Based on resolution proofs
- Framework of **flexible interpolation** algorithms
  - Based on resolution proofs

OpenSMT

- First open-source SMT solver developed since 2008
- Successful state-of-the-art tool with unique features
- Focus on **QF** logics: **LA**, **UF**, **AX**, and their combinations
  - Specialized solver for difference logics
- Resolution and DRAT **proofs** of unsatisfiability

  $\rightarrow$ Certification of the results
- Support of **unsatisfiable cores**

  $\rightarrow$ Based on resolution proofs
- Framework of **flexible interpolation** algorithms
  - Based on resolution proofs
  - Strong and weak Craig interpolants for QF_LRA and QF_UF

OpenSMT

- First open-source SMT solver developed since 2008
- Successful state-of-the-art tool with unique features
- Focus on **QF** logics: **LA**, **UF**, **AX**, and their combinations
  - Specialized solver for difference logics
- Resolution and DRAT **proofs** of unsatisfiability

$\rightarrow$ Certification of the results
- Support of **unsatisfiable cores**

$\rightarrow$ Based on resolution proofs
- Framework of **flexible interpolation** algorithms
  - Based on resolution proofs
  - Strong and weak Craig interpolants for QF_LRA and QF_UF
  - Flexible propositional size of interpolants

OpenSMT

- First open-source SMT solver developed since 2008
- Successful state-of-the-art tool with unique features
- Focus on **QF** logics: **LA**, **UF**, **AX**, and their combinations
  - Specialized solver for difference logics
- Resolution and DRAT **proofs** of unsatisfiability

$\rightarrow$ Certification of the results
- Support of **unsatisfiable cores**

$\rightarrow$ Based on resolution proofs
- Framework of **flexible interpolation** algorithms
  - Based on resolution proofs
  - Strong and weak Craig interpolants for QF_LRA and QF_UF
  - Flexible propositional size of interpolants
  - Incrementality supported as well

OpenSMT

- First open-source SMT solver developed since 2008
- Successful state-of-the-art tool with unique features
- Focus on **QF** logics: **LA**, **UF**, **AX**, and their combinations
  - Specialized solver for difference logics
- Resolution and DRAT **proofs** of unsatisfiability

  $\rightarrow$ Certification of the results
- Support of **unsatisfiable cores**

  $\rightarrow$ Based on resolution proofs
- Framework of **flexible interpolation** algorithms
  - Based on resolution proofs
  - Strong and weak Craig interpolants for QF_LRA and QF_UF
  - Flexible propositional size of interpolants
  - Incrementality supported as well
- Experimental *look-ahead* core
  - Alternative to CDCL(T)
  - Possibility of combined heuristic of CDCL(T) and look-ahead

SMTS

- We also develop in-house **distributed SMT solver**: *SMTS*

SMTS

- We also develop in-house **distributed SMT solver**: *SMTS*
- Based on partition-tree algorithm *cube-and-conquer*

SMTS

- We also develop in-house **distributed SMT solver**: *SMTS*
- Based on partition-tree algorithm *cube-and-conquer*
- Local portfolio solving with lemma sharing

SMTS

- We also develop in-house **distributed SMT solver**: *SMTS*
- Based on partition-tree algorithm *cube-and-conquer*
- Local portfolio solving with lemma sharing

$\rightarrow$ **Combination** of portfolio solving and partitioning

SMTS

- We also develop in-house **distributed SMT solver**: *SMTS*
- Based on partition-tree algorithm *cube-and-conquer*
- Local portfolio solving with lemma sharing

→ **Combination** of portfolio solving and partitioning

- Scatter or Lookahead OpenSMT core

SMT-COMP

- We constantly achieve good results

SMT-COMP

- We constantly achieve good results
- SMT-COMP'24 (`smt-comp.github.io`): **all tracks** covered

SMT-COMP

- We constantly achieve good results
- SMT-COMP’24 (`smt-comp.github.io`): **all tracks** covered
  - Single query & incremental tracks

SMT-COMP

- We constantly achieve good results
- SMT-COMP'24 (`smt-comp.github.io`): **all tracks** covered
  - Single query & incremental tracks
  - Model validation & unsat core tracks

SMT-COMP

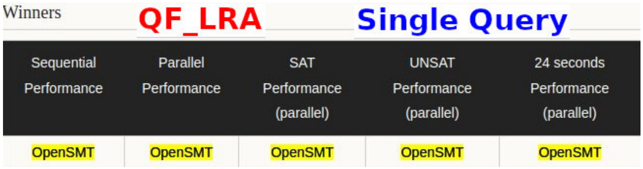
- We constantly achieve good results
- SMT-COMP’24 (`smt-comp.github.io`): **all tracks** covered
  - Single query & incremental tracks
  - Model validation & unsat core tracks
  - Parallel & cloud tracks (not evaluated yet)

SMT-COMP

- We constantly achieve good results
- SMT-COMP'24 (`smt-comp.github.io`): **all tracks** covered
    - Single query & incremental tracks
    - Model validation & unsat core tracks
    - Parallel & cloud tracks (not evaluated yet)
- Logics: LA, UF and combinations

SMT-COMP

- We constantly achieve good results
- SMT-COMP'24 (`smt-comp.github.io`): **all tracks** covered
  - Single query & incremental tracks
  - Model validation & unsat core tracks
  - Parallel & cloud tracks (not evaluated yet)
- Logics: LA, UF and combinations

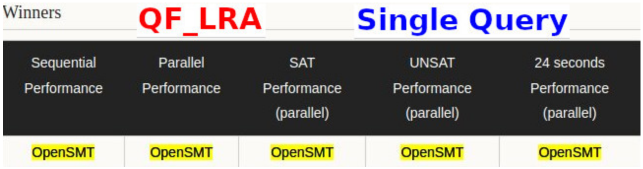


Winners QF_LRA Single Query

Sequential Performance	Parallel Performance	SAT Performance (parallel)	UNSAT Performance (parallel)	24 seconds Performance (parallel)
OpenSMT	OpenSMT	OpenSMT	OpenSMT	OpenSMT

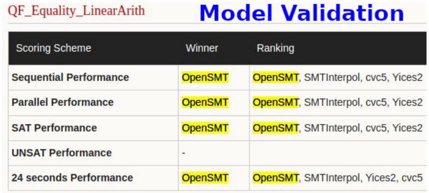
SMT-COMP

- We constantly achieve good results
- SMT-COMP'24 (`smt-comp.github.io`): **all tracks** covered
  - Single query & incremental tracks
  - Model validation & unsat core tracks
  - Parallel & cloud tracks (not evaluated yet)
- Logics: LA, UF and combinations



Winners QF_LRA Single Query

Sequential Performance	Parallel Performance	SAT Performance (parallel)	UNSAT Performance (parallel)	24 seconds Performance (parallel)
OpenSMT	OpenSMT	OpenSMT	OpenSMT	OpenSMT

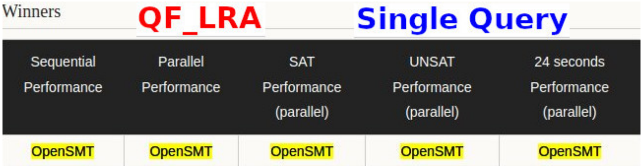


QF_Equality_LinearArith Model Validation

Scoring Scheme	Winner	Ranking
Sequential Performance	OpenSMT	OpenSMT, SMTInterpol, cvc5, Yices2
Parallel Performance	OpenSMT	OpenSMT, SMTInterpol, cvc5, Yices2
SAT Performance	OpenSMT	OpenSMT, SMTInterpol, cvc5, Yices2
UNSAT Performance	-	
24 seconds Performance	OpenSMT	OpenSMT, SMTInterpol, Yices2, cvc5

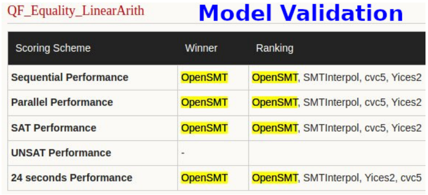
SMT-COMP

- We constantly achieve good results
- SMT-COMP'24 (`smt-comp.github.io`): **all tracks** covered
  - Single query & incremental tracks
  - Model validation & unsat core tracks
  - Parallel & cloud tracks (not evaluated yet)
- Logics: LA, UF and combinations



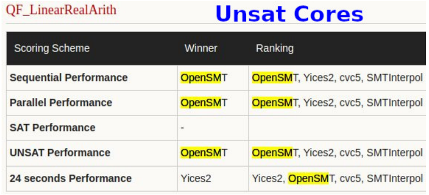
Winners **QF_LRA** **Single Query**

Sequential Performance	Parallel Performance	SAT Performance (parallel)	UNSAT Performance (parallel)	24 seconds Performance (parallel)
OpenSMT	OpenSMT	OpenSMT	OpenSMT	OpenSMT



QF_Equality_LinearArith **Model Validation**

Scoring Scheme	Winner	Ranking
Sequential Performance	OpenSMT	OpenSMT, SMTInterpol, cvc5, Yices2
Parallel Performance	OpenSMT	OpenSMT, SMTInterpol, cvc5, Yices2
SAT Performance	OpenSMT	OpenSMT, SMTInterpol, cvc5, Yices2
UNSAT Performance	-	
24 seconds Performance	OpenSMT	OpenSMT, SMTInterpol, Yices2, cvc5



QF_LinearRealArith **Unsat Cores**

Scoring Scheme	Winner	Ranking
Sequential Performance	OpenSMT	OpenSMT, Yices2, cvc5, SMTInterpol
Parallel Performance	OpenSMT	OpenSMT, Yices2, cvc5, SMTInterpol
SAT Performance	-	
UNSAT Performance	OpenSMT	OpenSMT, Yices2, cvc5, SMTInterpol
24 seconds Performance	Yices2	Yices2, OpenSMT, cvc5, SMTInterpol

Current Challenges

- Fast computation of subset-minimal **unsat cores**
  - Based on (DRAT) proofs?

Current Challenges

- Fast computation of subset-minimal **unsat cores**
  - Based on (DRAT) proofs?
- **Interpolation** and producing models for **arrays**

Current Challenges

- Fast computation of subset-minimal **unsat cores**
  - Based on (DRAT) proofs?
- **Interpolation** and producing models for **arrays**
- Theory of **floating-point** numbers

Current Challenges

- Fast computation of subset-minimal **unsat cores**
  - Based on (DRAT) proofs?
- **Interpolation** and producing models for **arrays**
- Theory of **floating-point** numbers
  - State-of-the-art methods may not scale to industrial problems

Current Challenges

- Fast computation of subset-minimal **unsat cores**
  - Based on (DRAT) proofs?
- **Interpolation** and producing models for **arrays**
- Theory of **floating-point** numbers
  - State-of-the-art methods may not scale to industrial problems
  - $\rightarrow$ Based on Simplex rather than bit-blasting?

Current Challenges

- Fast computation of subset-minimal **unsat cores**
  - Based on (DRAT) proofs?
- **Interpolation** and producing models for **arrays**
- Theory of **floating-point** numbers
  - State-of-the-art methods may not scale to industrial problems
  - $\rightarrow$ Based on Simplex rather than bit-blasting?
- Theory-specific algorithms for **distributed solving**

Usage of OpenSMT

- Features tuned and driven by meaningful verification tasks

Usage of OpenSMT

- Features tuned and driven by meaningful verification tasks
- In-house **model checking** software:
  - HiFrog, FunFrog, LoopFrog, eVolCheck, UpProver, Golem
  - Incremental verification of ANSI-C programs
  - . . .

Usage of OpenSMT

- Features tuned and driven by meaningful verification tasks
- In-house **model checking** software:
  - HiFrog, FunFrog, LoopFrog, eVolCheck, UpProver, Golem
  - Incremental verification of ANSI-C programs
  - . . .
- **Explanations** of classifications of neural networks

Usage of OpenSMT

- Features tuned and driven by meaningful verification tasks
- In-house **model checking** software:
  - HiFrog, FunFrog, LoopFrog, eVolCheck, UpProver, Golem
  - Incremental verification of ANSI-C programs
  - . . .
- **Explanations** of classifications of neural networks
- Multi-agent Path-finding with Continuous Time

$\rightarrow$ Planning tasks are useful too

Usage of OpenSMT

- Features tuned and driven by meaningful verification tasks
- In-house **model checking** software:
  - HiFrog, FunFrog, LoopFrog, eVolCheck, UpProver, Golem
  - Incremental verification of ANSI-C programs
  - ...
- **Explanations** of classifications of neural networks
- Multi-agent Path-finding with Continuous Time

$\rightarrow$ Planning tasks are useful too

Related Work

- **Automated test-generation** for Solidity programs
  - WIP: Use OpenSMT

Usage of OpenSMT

- Features tuned and driven by meaningful verification tasks
- In-house **model checking** software:
  - HiFrog, FunFrog, LoopFrog, eVolCheck, UpProver, Golem
  - Incremental verification of ANSI-C programs
  - . . .
- **Explanations** of classifications of neural networks
- Multi-agent Path-finding with Continuous Time

→ Planning tasks are useful too

Related Work

- **Automated test-generation** for Solidity programs
  - WIP: Use OpenSMT
- **Certification** production for **smart contracts** in Solidity

Promotion

We have open Ph.D. positions in Lugano!

Questions?

Websites:

verify.inf.usi.ch
github.com/usi-verification-and-security/opensmt

Contact:

tomas.kolarik@usi.ch
natasha.sharygina@usi.ch