CaDiCaL($\mathcal{T}$): CaDiCaL as CDCL($\mathcal{T}$) Engine in cvc5

Aina Niemetz and **Mathias Preiner**

Shonan Meeting 180, October 2–5, 2023

The cvc5 SMT Solver

- **state-of-the-art** SMT solver
- most recent incarnation of the CVC tools
- **joint project** led by **Stanford University** and **University of Iowa** in collaboration with
  - Universidade Federal de Minas Gerais (Brazil)
  - Bar Ilan University (Israel)
- based on **CDCL($\mathcal{T}$)** framework
- supports **wide range of theories** in combination with **quantifiers**
  - all SMT-LIB theories + **non-standard theories** and **theory extensions**
- support for **proofs** (incl. preprocessing, rewriting)
- capabilities **beyond standard SMT**
  - SyGuS, abduction, interpolation, quantifier elimination, optimization (WIP)

The CDCL($\mathcal{T}$) Lazy SMT Framework

- **propositional abstraction** of the input formula
- **iteratively refined** until abstraction is $\mathcal{T}$-consistent or unsat
- theory layer **guides** the search of the SAT solver
- **online, tight integration of SAT solver**
  - **theory layer** interacts with SAT solver **during the search**
  - backward communication channel to **notify theory layer** about variable assignments, decisions, backtracks
  - theory layer **derives** conflicts, **propagates** theory literals, **suggests** decisions based on theory-guided heuristics

CDCL($\mathcal{T}$) SAT Solver: Current State-of-the-Art

- **no standardized SAT solver interface**
  for **interactive** incremental SAT solving

- **solver-specific** workarounds and modifications to the SAT solver
- **error prone**, high potential for unintentional **performance** hits
- **difficult to replace**
- **missed** opportunities to take advantage of **improvements** in SAT

CDCL($\mathcal{T}$) SAT Solver: Current State-of-the-Art

- **no standardized SAT solver interface**
  for **interactive** incremental SAT solving

- **solver-specific** workarounds and modifications to the SAT solver
- **error prone**, high potential for unintentional **performance** hits
- **difficult to replace**
- **missed** opportunities to take advantage of **improvements** in SAT

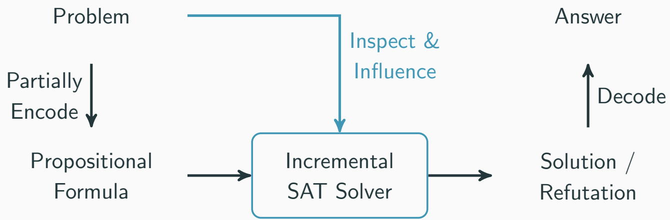
Situation in cvc5 (until recently)

- integrates **highly customized version of MiniSat**
  - produces **resolution proofs**
  - **push/pop** for adding/deleting clauses and variables
  - custom **theory-guided decision heuristics**

IPASIR-UP in a Nutshell

IPASIR-UP = **IPASIR** + **U**ser **P**ropagators (Fazekas et al. 2023)

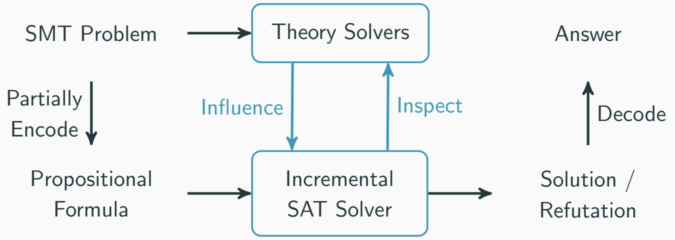
- presented at SAT 2023
- a SAT solver **interface** for
- **interactive** incremental SAT solving



IPASIR-UP in a Nutshell

IPASIR-UP = **IPASIR** + **U**ser **P**ropagators (Fazekas et al. 2023)

- presented at SAT 2023
- a SAT solver **interface** for
- **interactive** incremental SAT solving



- **our focus:** Integration as CDCL($\mathcal{T}$) SAT solver

IPASIR-UP in cvc5

CaDiCaL($\mathcal{T}$) Integration (via IPASIR-UP)

- **Full utilization** of interface
- ∼**500 LOC in C++** for implementing interface (∼800 including comments)
- "easily" replaced with any SAT solver implementing IPASIR-UP
- **supports all* cvc5 features**
- *proof support **work-in-progress**
- **changes** compared to MiniSat
  - resolution proofs → DRAT/LRAT proofs (WIP)
  - native push/pop → activation literals

IPASIR-UP Interface

Notifications (for Inspecting CDCL Search)

- notify_assignment
- notify_new_decision_level
- notify_backtrack

Callbacks (for Influencing CDCL Search)

- cb_propagate
- cb_add_reason_clause_lit
- cb_decide
- cb_add_external_clause_lit
- cb_check_found_model

Assignment Notifications

notify_assignment

- sends assignment notification for observed variables

- **track assignment** for theory literals
  - constructs (partial) assignment of propositional abstraction
- track whether assignment is
  - decision
  - fixed
- **notify** theory solvers about assigned theory literal,
  e.g., if (observed) variable corresponding to theory literal $a < 42$ is assigned to
  - true: send $a < 42$ to arithmetic solver
  - false: send $a \geq 42$ to arithmetic solver

Decision Notifications

- Used to **manage the incremental state** of cvc5
- **backtrackable data structures** (context-dependent), associated with a context
  - **SAT context**, backtracks when SAT solver backtracks (decision-level push/pop)
  - user context (SMT-level push/pop)

notify_new_decision_level

- **push SAT context**
- track decision + level

notify_backtrack (L)

- **pop SAT context** to L
- **undo assignments** at level $> L$
- **resend** "popped" fixed theory literals
  - theory literals **fixed** at levels $> L$ are **popped**
  - fixed assignments only **notified once**
  - **resend fixed** theory literals at level L

Check Full Assignment for $\mathcal{T}$-Consistency

cb_check_found_model

- called when SAT solver found **satisfying assignment**
  - ▷ returns **true** if assignment $\mathcal{T}$-consistent, **false** otherwise
- checks if assignment is $\mathcal{T}$-**consistent** (**full effort** check)
  - theory solvers check $\mathcal{T}$-consistency of assigned theory literals
    - send **conflict** clause
    - send **lemmas**
    - send **theory propagations**
      - ▷ adds **eager explanations** at this point to force SAT solver to propagate
  - $\mathcal{T}$-**consistent if**:
    - theory solvers **performed all checks** and
    - **no new variables** were added and
    - **no new lemmas or conflicts** sent, i.e., no new clauses added

Injecting Decisions

cb_decide

- called before SAT solver makes decision
- used to **inject theory-guided decisions**
  - **theory decisions** (required)
    - ▷ decision strategies used by theory solvers
  - **decision requests** (optional)
    - ▷ custom decision heuristics
    - ▷ e.g.: justification heuristic, chooses next decision based on structure of formula
- may discover **partial satisfying assignment**
  - triggers full effort check, i.e., calls **cb_check_found_model**
  - stops search if current assignment is $\mathcal{T}$-consistent

Adding Clauses (during Search)

cb_add_external_clause_lit

- clauses added **during search are buffered**
  - theory lemmas
  - theory conflicts
- buffered clauses are only **added during callback**

cb_has_external_clause

- checks whether new clauses are pending

Propagations

cb_propagate

- called after SAT solver is done with propagation
- performs **lightweight checks** in theory solvers (**standard effort** check)
- **theory propagations**

cb_add_reason_clause_lit (prop_lit)

- called when theory propagation **prop_lit is involved in conflict**
- **explain** theory propagation
- adds explanation (reason clause)

SMT push/pop via Activation Literals

- **happens between** SAT solver calls, **not during search**
- **push** assertion level
  - create fresh activation literal l_n for pushed level n
  - add l_n to each clause added in level n
  - prior to solving, assume $\neg l_i$ for $i \in \{1, n\}$
- **pop** assertion level
  - add unit clause $\{l_n\}$ for popped level n
    ▷ garbage collects all clauses added at level n
  - **unobserve** and **fix value** of variables introduced in n (**important** for performance)
- **renotify fixed literals** with fixed level $>$ intro level
  - **requires** keeping track of assertion levels when
    - variable was introduced
    - variable assignment was fixed

```
(set-logic ...)
...
(assert ...)   ; A1
(assert ...)   ; A2
(push 1)
 (assert ...)  ; A3
 (check-sat)
(pop 1)
(check-sat)
...
```

Evaluation

- all **incremental** and **non-incremental** benchmarks of SMT-LIB 2023
  - 434, 212 non-incremental benchmarks
  - 43, 287 incremental benchmarks
- 300s time limit, 8GB memory limit
- comparison of cvc5-1.0.8-dev with
  - **MiniSat** (custom, based on 2.2.0)
  - **CaDiCaL** (IPASIR-UP, version 1.7.4)

Evaluation: SMT-COMP Non-Incremental Divisions

Division	cvc5+MiniSat solved	cvc5+MiniSat time [s]	cvc5+CaDiCaL solved	cvc5+CaDiCaL time [s]
Arith (6,925)	**6,341**	181,329	6,332	183,417
BitVec (6,185)	**5,645**	168,844	5,625	175,110
Equality (12,159)	5,331	2,060,608	**5,337**	2,059,279
Equality+LinearArith (56,562)	**45,970**	3,196,706	45,966	3,198,129
Equality+MachineArith (10,911)	1,073	2,958,372	**1,075**	2,958,746
Equality+NonLinearArith (21,162)	**13,333**	2,425,551	13,123	2,474,917
FPArith (3,979)	3,133	275,579	**3,138**	272,751
QF_Bitvec (46,191)	**43,735**	1,092,892	43,713	1,092,907
QF_Datatypes (8,403)	8,083	109,941	**8,158**	84,593
QF_Equality (8,054)	8,043	9,338	**8,047**	6,968
QF_Equality+Bitvec (16,801)	15,922	355,232	**16,132**	263,786
QF_Equality+LinearArith (3,644)	3,464	65,242	**3,497**	52,176
QF_Equality+NonLinearArith (906)	**721**	61,692	711	64,217
QF_FPArith (76,252)	76,072	93,150	**76,087**	77,682
QF_LinearIntArith (16,389)	11,530	1,604,847	**12,017**	1,489,186
QF_LinearRealArith (2,008)	1,686	142,921	**1,784**	107,522
QF_NonLinearIntArith (25,446)	13,076	4,080,649	**14,058**	3,696,580
QF_NonLinearRealArith (12,134)	11,155	336,630	**11,247**	309,251
QF_Strings (100,101)	98,407	619,928	**98,870**	483,260
Total (434,212)	372,720	19,839,459	**374,917**	19,050,487

- +2197 solved instances
- ∼ 25% **faster** on commonly solved instances
- 2–4× **faster** in several logics
- **13 of 19** divisions **improved**
  - quantifier-free better overall
  - quantified logics a bit behind
- **promising performance** without much tuning or optimizations
- **solid baseline** for future tuning with IPASIR-UP interface

Evaluation: SMT-COMP Incremental Divisions

Division	cvc5+MiniSat solved	cvc5+MiniSat time [s]	cvc5+CaDiCaL solved	cvc5+CaDiCaL time [s]
Arith (11)	41,362	233	41,362	240
BitVec (18)	36,114	2,992	**36,117**	3,031
Equality (4,067)	**46,256**	620,984	46,216	623,400
Equality+LinearArith (1,894)	**431,172**	57,390	430,552	59,637
Equality+MachineArith (4)	818	310	818	309
Equality+NonLinearArith (4,374)	82,721	651,804	**83,801**	644,742
FPArith (10)	**3,422**	1,849	3,421	1,849
QF_Bitvec (2,590)	**51,334**	63,165	51,260	62,036
QF_Equality (1,778)	29,981	4,616	**29,982**	4,588
QF_Equality+Bitvec (3,633)	**7,677**	148,084	7,620	153,446
QF_Equality+Bitvec+Arith (664)	959	51,776	**985**	44,466
QF_Equality+LinearArith (3,947)	**2,266,894**	130,331	1,893,335	133,167
QF_Equality+NonLinearArith (1,018)	**96,917**	24,307	92,813	23,932
QF_FPArith (19,188)	538,936	955,264	**560,379**	745,166
QF_LinearIntArith (69)	**1,332,173**	17,582	1,089,226	17,109
QF_LinearRealArith (10)	482	3,004	**571**	2,918
QF_NonLinearIntArith (12)	**349,862**	3,603	326,463	3,603
Total (43,287)	**5,317,080**	2,737,301	4,694,921	2,523,646

- improvements in **some logics**
- overall performance not there yet

- poor performance on benchmarks with **many check-sat calls**
- overhead of **activation literals?**

Discussion on Incremental Performance

Observation

Performance poor on benchmarks with **large number of check-sat calls**

Example: `kundu_true-*.smt2` (QF_LIA)

- 900k+ check-sat calls
- solved queries within 300 seconds
  - **MiniSat:** 148,997
  - **CaDiCaL:** 103,843

Discussion on Incremental Performance

Observation

Performance poor on benchmarks with
large number of check-sat calls

Example: `kundu_true-*.smt2` (QF_LIA)

- 900k+ check-sat calls
- solved queries within 300 seconds
  - **MiniSat:** 148, 997
  - **CaDiCaL:** 103, 843

Activation Literal Overhead Experiment

`(push 1)`	...	fresh literal l_n
`(assert true)`	...	add clause $(l_n \vee \top)$
`(check-sat)`	...	assume $\neg l_n$
`(pop 1)`	...	add clause (l_n)

- Repeated N times in one benchmark

Discussion on Incremental Performance

Observation

Performance poor on benchmarks with **large number of check-sat calls**

Example: `kundu_true-*.smt2` (QF_LIA)

- 900k+ check-sat calls
- solved queries within 300 seconds
  - **MiniSat:** 148,997
  - **CaDiCaL:** 103,843

Activation Literal Overhead Experiment

`(push 1)`	...	fresh literal l_n
`(assert true)`	...	add clause $(l_n \vee \top)$
`(check-sat)`	...	assume $\neg l_n$
`(pop 1)`	...	add clause (l_n)

- Repeated N times in one benchmark

N	MiniSat	CaDiCaL	Slowdown
10k	265ms	462ms	1.7×
25k	625ms	1.8s	2.8×
50k	1.2s	5.8s	4.8×
75k	1.8s	11.9s	6.6×
100k	2.5s	20.3s	8.1×

Conclusion

Summary

- **non-incremental** performance solid
- incremental performance still lagging behind
- IPASIR-UP integration
  - **simple** and **flexible**
  - **captures all** functionality required by cvc5

What's Next?

- **DRAT/LRAT** proof integration (WIP)
- SAT solver **tuning** (currently default options)
- improve performance on **quantified problems**
- improve **incremental performance**
- IPASIR-UP: reduce callbacks, notifications

CVC5

`https://cvc5.github.io`

Scatter Plots QF_S*, QF_A*

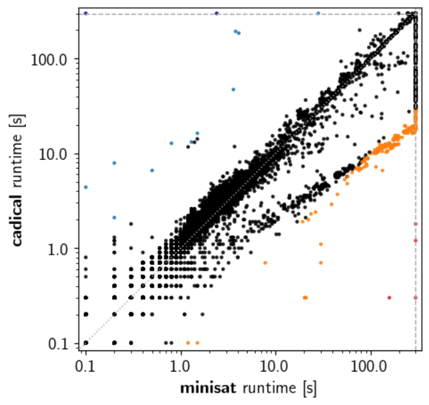


Figure 1: QF_S* (Logics with Strings)

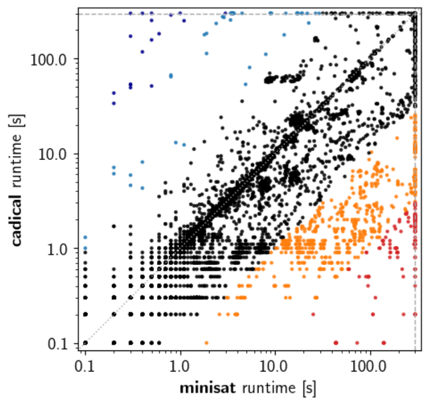


Figure 2: QF_A* (Logics with Arrays)

Quanitfier-free and Quanitifed Logics

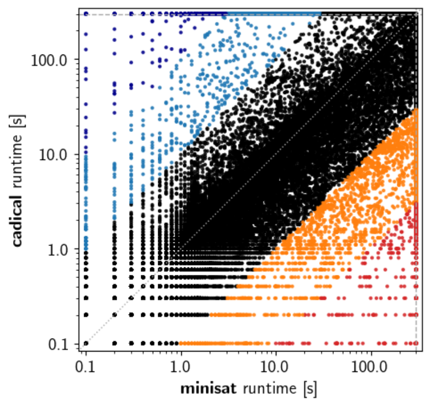


Figure 3: Quantifier-free Logics

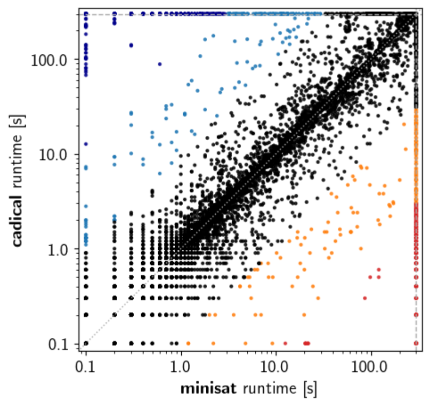


Figure 4: Quantified Logics

References

Fazekas, Katalin et al. (2023). **“IPASIR-UP: User Propagators for CDCL”.** In: *26th International Conference on Theory and Applications of Satisfiability Testing, SAT 2023, July 4-8, 2023, Alghero, Italy*. Ed. by Meena Mahajan and Friedrich Slivovsky. Vol. 271. LIPIcs. Schloss Dagstuhl - Leibniz-Zentrum für Informatik, 8:1–8:13. DOI: 10.4230/LIPIcs.SAT.2023.8. URL: https://doi.org/10.4230/LIPIcs.SAT.2023.8.