From Clauses to Klauses

Joseph E. Reeves, Marijn J. H. Heule, and Randal E. Bryant

Carnegie Mellon University

CAV 2024

From Clauses to Klauses

Conjunctive Normal Form (CNF): conjunction of clauses

$$(x_1 \vee x_2 \vee x_3) \wedge \ldots$$

- Standard input to satisfiability (SAT) solvers for $30+$ years

From Clauses to Klauses

Conjunctive Normal Form (CNF): conjunction of clauses

$$(x_1 \lor x_2 \lor x_3) \land \dots$$

- Standard input to satisfiability (SAT) solvers for $30+$ years

Cardinality Conjunctive Normal Form (KNF): conjunction of cardinality constraints (klauses)

$$(x_1 + x_2 + x_3 + x_4 + x_5 + x_6 \geq 3) \land \dots$$

- Extend input for more flexibility in solving
- Incremental change without sacrificing general usage

Cardinality Constraints Extend Clauses

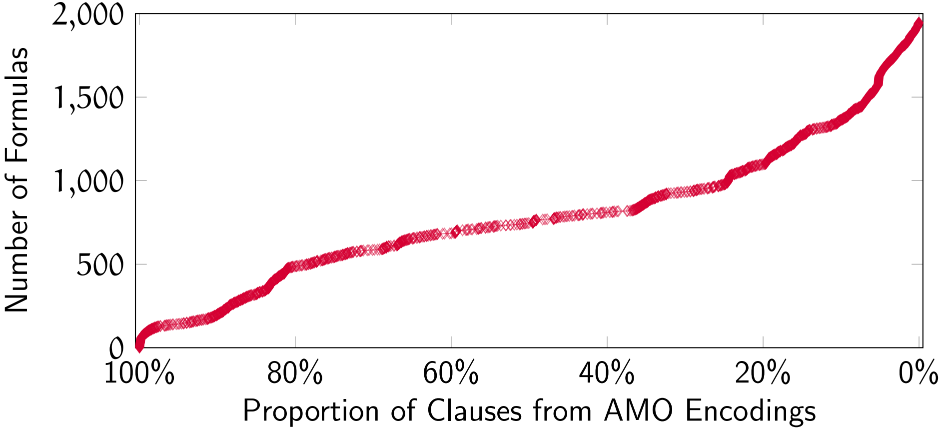
- A clause can be represented as a cardinality constraint

$$(x_1 \vee x_2 \vee x_3) = (x_1 + x_2 + x_3 \geq 1)$$

- Comparison operators $(=, >, \geq, <, \leq)$ can be represented with one or two $\geq$ constraints
- At-Most-One (AMO): at most one literal is true

$$AMO(x_1, x_2, x_3) = (x_1 + x_2 + x_3 \leq 1) = (\overline{x}_1 + \overline{x}_2 + \overline{x}_3 \geq 2)$$

AMO Cardinality Constraints in Competition Formulas



- Anniversary Track: two decades of SAT competition
- 36% of 5,300 formulas with AMO of size 5 or larger

Historic Motivations for Cardinality Input

- Easier for users to create problems
  - Complex encoding types (e.g. modulus k-totalizer) error-prone
- Can use stronger reasoning techniques
  - e.g., PHP from exponential to linear solving
- Smaller formulas
  - e.g., Magic Squares 6×6: 4k constraints $\rightarrow$ 600k clauses
- Faster constraint propagation
  - e.g., AMO can propagate everything in single step

Historic Motivations for Cardinality Input

- Easier for users to create problems
  - Complex encoding types (e.g. modulus k-totalizer) error-prone
- Can use stronger reasoning techniques
  - e.g., PHP from exponential to linear solving
- Smaller formulas
  - e.g., Magic Squares 6×6: 4k constraints $\rightarrow$ 600k clauses
- Faster constraint propagation
  - e.g., AMO can propagate everything in single step

Failed to replace CDCL-based CNF solvers for general use

- Clausal reasoning: optimizations, heuristics, inprocessing
- Clausal encodings: better clause learning, shorter proofs

Cardinality Input for CDCL: an Incremental Change

CNF: CDCL

- Auxiliary variables to encode high-level constraints

$$\mathrm{AMO}(x_1, x_2, x_3, x_4, x_5) \rightarrow \mathrm{AMO}(x_1, x_2, x_3, y) \wedge \mathrm{AMO}(\overline{y}, x_4, x_5)$$

Cardinality Input for CDCL: an Incremental Change

CNF: CDCL

- Auxiliary variables to encode high-level constraints

$$\mathrm{AMO}(x_1, x_2, x_3, x_4, x_5) \rightarrow \mathrm{AMO}(x_1, x_2, x_3, y) \wedge \mathrm{AMO}(\overline{y}, x_4, x_5)$$

KNF: Cardinality-CDCL

- Lift inprocessing, proof checking, constraint propagation
- Leverage auxiliary variables with clausal encodings
- No stronger reasoning, no separate propagation engine

Contributions

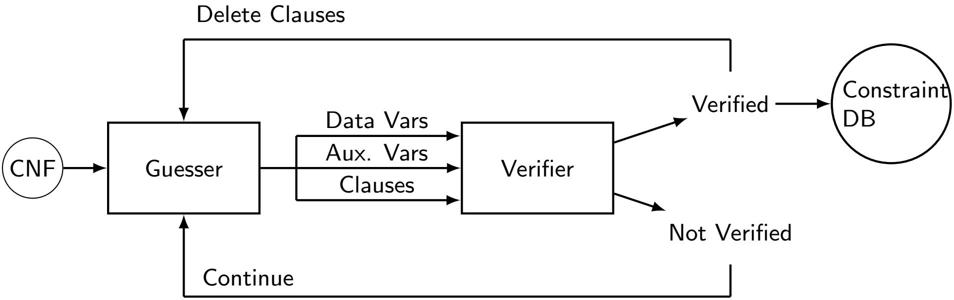
Long Term Goal: User generated KNF formulas

- Problem specific encoding optimizations
- Efficient KNF proof checking
- More paradigms: local search, MaxSAT, parallel solving

Short Term Goal: Backwards compatibility with CNF

- Cardinality constraint extraction producing KNF
- Multiple configurations for KNF solving
- End-to-end proof checking for KNF extraction and solving

Cardinality Constraint Extraction Framework



- Guess a candidate cardinality constraint
- Verify the constraint structure by constructing a BDD and filter out non-constraints

Extractor Comparison on PySAT AMO Encodings

Table: Size 10 AMO on 8 PySAT encodings. ✓ if complete AMO is extracted.

Tool	Pair	SCnt	CNet	SNet	Tot	mTot	mkTot	Lad
Guess-and-Verify	✓	✓	✓	✓	✓	✓	✓	X
Lingeling	✓	X	X	X	X	X	X	X
Riss	✓	X	X	X	X	X	X	X

- Lingeling and Riss only find smaller sub constraints
- BDD verifier works for general cardinality constraints
- Need more sophisticated heuristics for guesser

Cardinality Constraint Solving Options

NATIVE:

- Propagate natively on cardinality constraints (CCDCL)
  - Extends CDCL watch-pointers and conflict analysis
  - Faster propagation, no aux. variables generally better on SAT

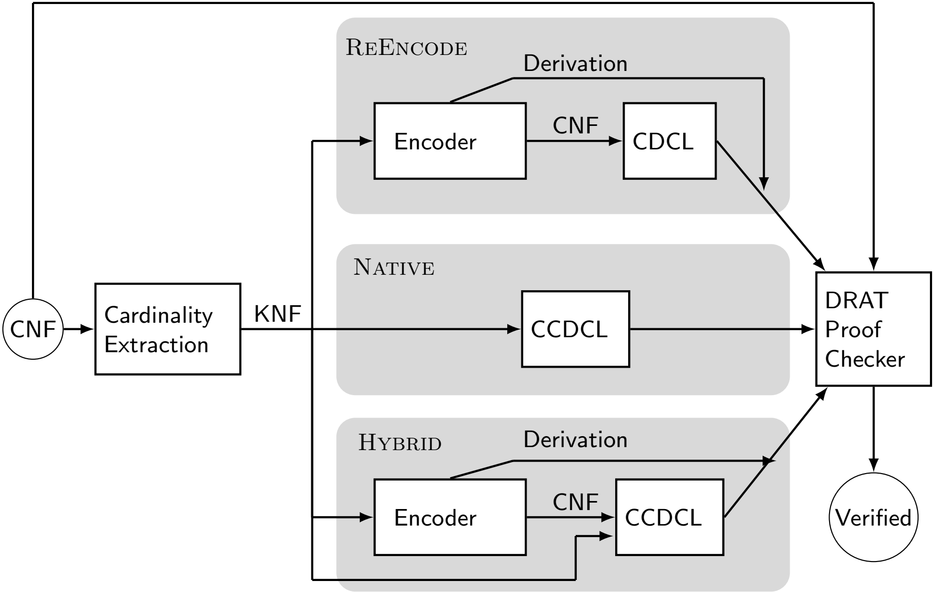
REENCODE:

- Encode cardinality constraints into clauses
  - Encoded constraints generally better on UNSAT

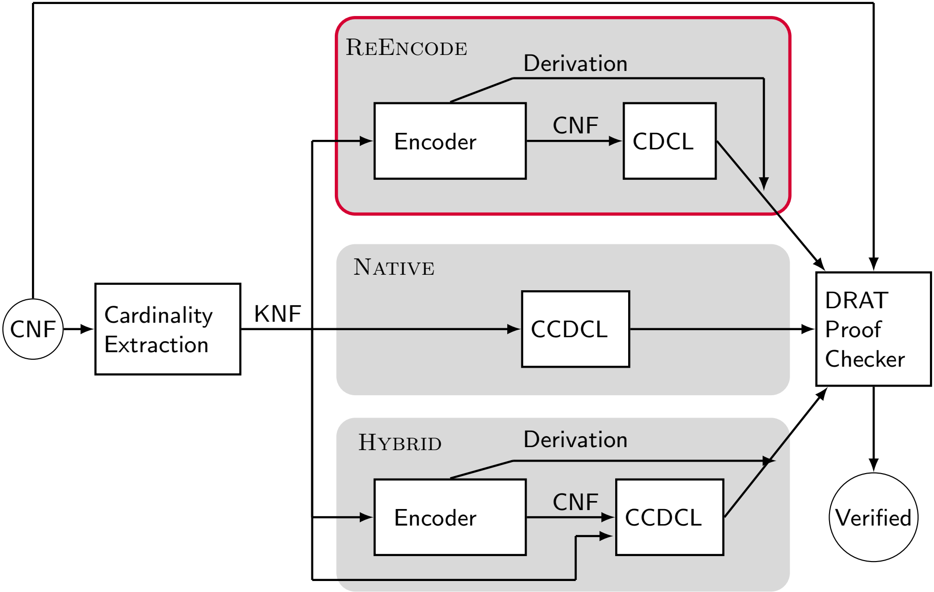
HYBRID:

- Combination of NATIVE and REENCODE
  - Reencoded clauses kept throughout solving
  - Native propagation enabled half of the time
  - Good for both SAT and UNSAT instances

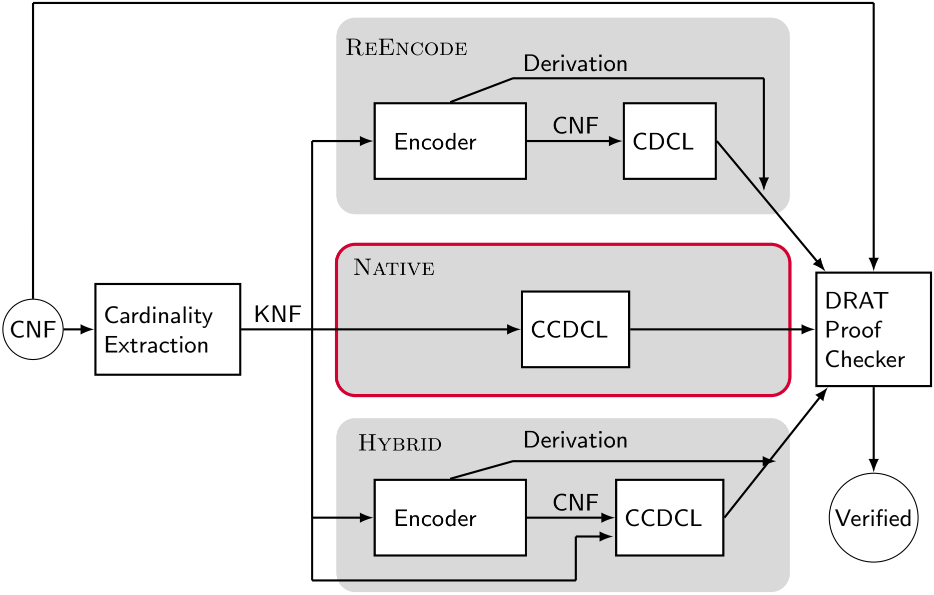
Cardinality Solving and Proof Checking for CNF



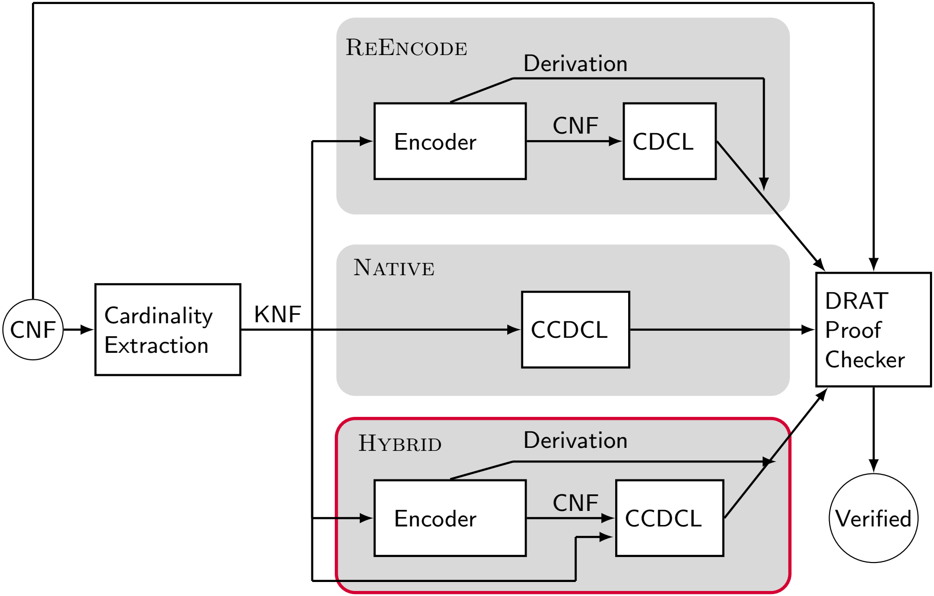
Cardinality Solving and Proof Checking for CNF



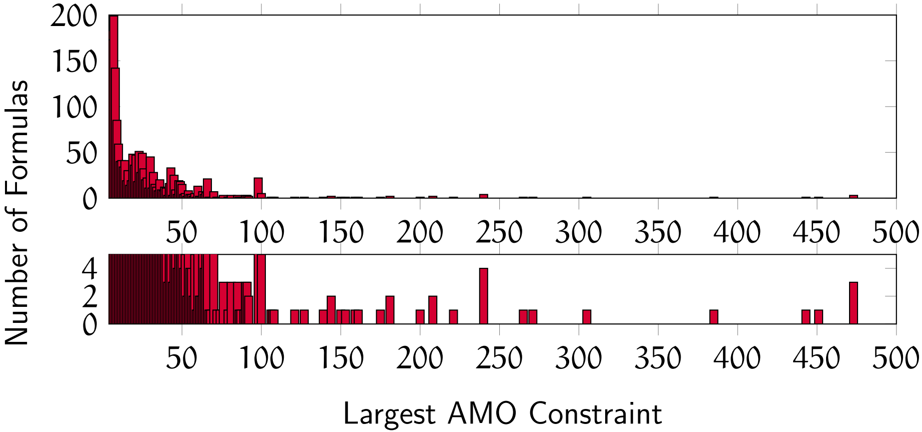
Cardinality Solving and Proof Checking for CNF



Cardinality Solving and Proof Checking for CNF

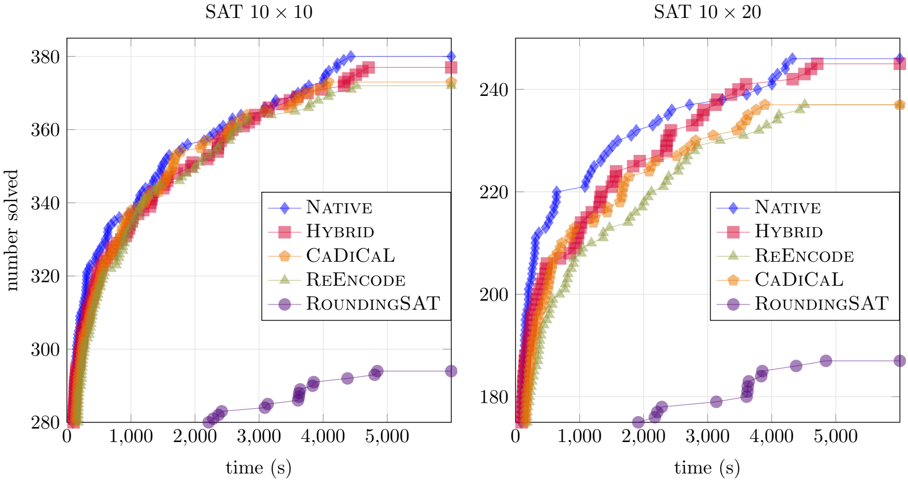


Extracting AMO Constraints on Competition Formulas



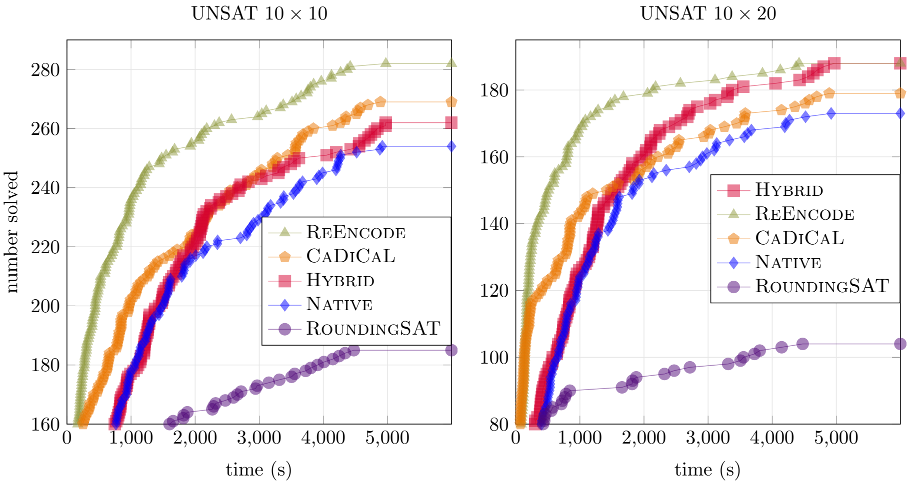
- planning (473), petrinet (451), edge-matching (140)
- Runtime average 69s, 78% below 15s

Solving Extracted Satisfiable Formulas



- 10×10: 933 formulas with at least 10 AMOs of size ≥ 10
- 10×20: 587 formulas with at least 10 AMOs of size ≥ 20

Solving Extracted Unsatisfiable Formulas



- ReEncode displays strength of auxiliary variables
- Hybrid suffers overhead from mode switching

Conclusion

- KNF input requires moderate changes to a SAT solver
- Combining clausal encodings and native propagation good for SAT and UNSAT problems
- Apply solver to KNF problems too large for CNF encoding
  - Computational geometry, e.g., point-discrepancy problem

Conclusion

- KNF input requires moderate changes to a SAT solver
- Combining clausal encodings and native propagation good for SAT and UNSAT problems
- Apply solver to KNF problems too large for CNF encoding
  - Computational geometry, e.g., point-discrepancy problem

We'd like to hear about cardinality constraint problems you'd like to solve!