Encoding First Order Proofs in SMT

Jeremy Bongio, Cyrus Katrak, Hai Lin, Christopher Lynch, Eric McGregor and Yuefeng Tang

Contents

- First Order Tableau
- Encoding Tableau with SAT
- Encoding Tableau with SMT
- Experimental Results
- Related and FutureWork

Rigid Unsatisfiability

- Given set of clauses S, is there substitution σ such that $S\sigma$ is unsat?

- Example 1:

  $S = \{P(a), \quad \sim P(x) \vee Q(f(x)), \quad \sim Q(y)\}$

- Unsat: Let $\sigma = [x \mapsto a, y \mapsto f(a)]$

Examples of Rigid SAT

- Example 2:

  $S = \{P(x), \quad \sim P(a) \wedge \sim P(b)\}$

- Example 3:

  $S = \{P(a), \quad \sim P(x) \vee P(f(x)), \quad \sim P(f(f(a)))\}$

- Both of these are Unsat but not Rigid Unsat

Why Rigid Unsat

- Could be used to solve Unsat
  - Keep adding new variants of clauses
- Also useful for Protocol Verification
  - Rigid Variables used to bound number of sessions

Complexity of Rigid Unsat

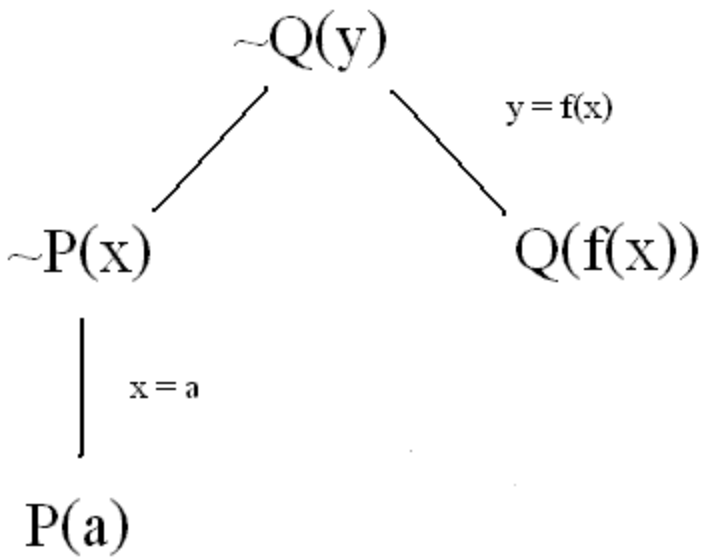
- Rigid Unsat is Σ_2^p-complete
- Rigid Horn Unsat is NP-complete

- Suggests that Rigid Horn Unsat can be encoded by SAT, but Rigid Unsat cannot be encoded directly
  - So first we consider encoding of Horn

How to encode Rigid Unsat

- We could encode existence of resolution proof [PL encode Prop Unsat as sequence of n clauses]
- Instead we encode existence of tableau
  - Encoding is more direct
  - Fewer permutations of tableau
  - Don’t know if [PL] can be extended to FO

Tableau for Rigid Horn UNSAT



Rules for Horn Tableau

- Start with negative clause
- Each negated literal extended with some clause containing complementary literal
- L is extended with C by adding edge from L to every literal in C

Rules for Horn Tableau

- Start with negative clause
- Each negated literal extended with some clause containing complementary literal
- L is extended with C by adding edge from L to every literal in C
- Unifications must be consistent

Rules for Horn Tableau

- Start with negative clause
- Each negated literal extended with some clause containing complementary literal
- L is extended with C by adding edge from L to every literal in C
- Unifications must be consistent
- Tableau must be finite – no cycles

Preparing for SAT encoding

- Enumerate clauses:

  $C_1,\cdots,C_n$

- Enumerate negative literals of C_i:

  $L_{i1},\cdots,L_{im}$

Prop. Variables used in encoding

- c_i: C_i in tableau
- l_{ij}: L_{ij} in tableau
- e_{ijk}: L_{ij} extended with clause C_k
- $u_{x=t}$: x=t is implied by unifiers
- p_{ik}: there is a path from C_i to C_k

Clauses in SAT encoding

- $\vee\{c_i \mid C_i$ is negative clause$\}$
- $c_i \rightarrow l_{ij}$
- $l_{ij} \rightarrow \vee\{e_{ijk} \mid L_{ij}$ is complementary to positive literal in $C_k\}$
- $e_{ijk} \rightarrow c_k$
- $e_{ijk} \rightarrow u_{x=t}$ if x=t used in unifier of that ext.
- $e_{ijk} \rightarrow p_{ik}$

More clauses in SAT encoding

- So far encoding is quadratic in number of literals
- But we have not yet encoded finiteness of tableau or consistency of unifers
  - They will be cubic

Finiteness of Tableau

- $p_{ij} \wedge p_{jk} \rightarrow p_{ik}$
- $\neg p_{ii}$

- Transitivity is cubic in number of clauses

Consistency of Unifiers

- Propagate assignments and detect $\perp$
- $u_{x=f(y)} \wedge u_{x=f(a)} \rightarrow u_{y=a}$
- $\neg u_{x=a} \vee \neg u_{x=b}$

- Cubic and also approximate
  - No occurs check

Problem

- Our implementation (based on Minisat) worked well sometimes but not always
- Basic encoding is quadratic
- But encoding of finiteness and unification is cubic
- Solution: Use SMT
  - Theories encode finiteness and unification

Idea

- Quadratic time is building tableau and making choices
- Cubic time is deterministic verification of global properties
- SMT solves NP-complete problems
  - Construct witness using local properties (SAT)
  - Verify correctness of global properties (theory)

Finiteness of Tableau

- x_i is rational number
- $e_{ijk} \rightarrow x_i < x_j$
- If C_i appears above C_j then $x_i < x_j$
- Don't need to encode transitivity
- Yices decides it using linear rational arith.
  - Actually, simple difference equations

Representing Unification

- Yices allows inductive datatypes
- Terms represented by inductive datatypes
- Function, predicate symbol is constructor
- FO variables are instances of terms
- Unification is equality of two terms
- Finiteness and Unification no longer encoded by SAT

Implementation

- SAT encoding
  - Solved by Minisat
  - Solved by Yices
- SMT encoding
  - Solved by Yices

ChewTPTP

- We tested our implementation on TPTP (Thousands of Problems for Theorem Provers)
- Our implementation is called ChewTPTP (Chris Hai Eric Wenjin Todd and Patty Theorem Prover)

Experimental Times

Problem	SAT-M	SAT-Y	SMT-Y
NLP108-1	2.94	1.94	0.07
RNG001-5	6.93	5.32	0.84
SWV017-1	10.82	11.27	0.1

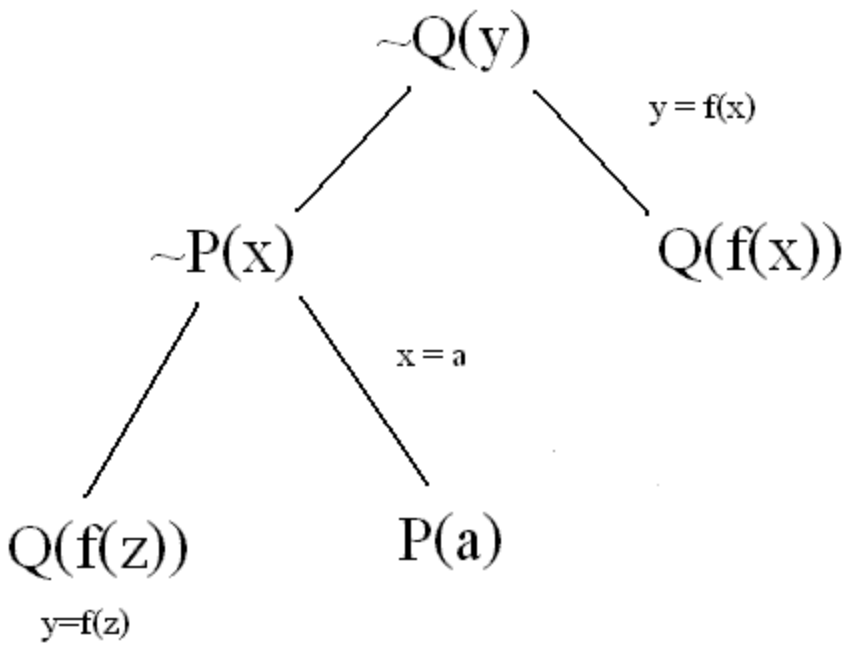
Number of Clauses

Problem	SAT	SMT
NLP108-1	134,667	287
RNG001-5	258,888	1527
SWV017-1	625,119	1137

Analysis

- Number of Clauses/Variables reduced by factor of more than thousand
- Runs much faster (mostly due to time generating clauses)
- Yices slightly faster than Minisat

NonHorn clause tableau



Encoding of nonHorn tableau

- nonHorn encoding like Horn encoding
- Except that we must encode existence of complementary ancestor literal that is not parent node
- This is similar to earlier encoding of acyclicity, so cubic in size
- So SMT encoding remains cubic

Another encoding difference

- In tableau, literal may appear twice but closed with different literal each time
- This requires successive encodings with more copies of each clause
- Not surprising since Rigid Unsatisfiability is Σ_2^P-complete

Experimental Times

Problem	SAT-M	SAT-Y	SMT-Y
ALG002-1	43.51	64.92	75.33
ANA004-5	47.25	21.5	83.54
COM003-2	88.72	84.54	168.1

Number of Clauses

Problem	SAT	SMT
ALG002-1	54,559	32,731
ANA004-5	101,166	44,953
COM003-2	2,920,669	2,365,922

Analysis

- Number of Clauses/Variables reduced by factor of less than one half
- Runs slower

Conclusions

- We encoded first order theorem proving first in SAT and then in SMT
- SAT part builds constructs (tableau) according to local properties
- Theory part verifies global properties
- Horn case: SMT is faster due to reduction in size from $O(n^3)$ to $O(n^2)$
- NonHorn case: worse, remains $O(n^3)$

Future Work

- Add theory to make nonHorn case $O(n^2)$
- Determine nonRigid satisfisiability
- Can our method be used to add quantifiers to SMT
- Better understand when to use SMT encoding