General theorem proving for satisfiability modulo theories: an overview

Maria Paola Bonacina

Dipartimento di Informatica
Università degli Studi di Verona
Verona, Italy, EU

Talk given at Microsoft Research, Redmond, Washington, USA

14 May 2008

Motivation

- Software is everywhere
- Needed: *Reliability*
- Difficult goal: Software may be
    - Artful
    - Complex
    - Huge
    - Varied
    - Old (and undocumented)
    - ... possibly reflecting some "natural" laws of computing?
    - ...
- Software/hardware border: blurred, evolving

Some approaches to software reliability

- Testing (test case generation ...)
- Programmer assistants
- Program analyzers
- Static analysis (types, extended static checking, abstract interpretations ...)
- Dynamic analysis (traces ...)
- Software model checkers (+ theorem proving, e.g., BMC, CEGAR-SMC)
- ...

Reasoning about software

Reasoning about software

- Help find and remove bugs
- Find and remove bugs
- Prove a program free of certain bugs
- Prove a program correct

Systems with reasoning about software

Typical architecture:

- *Front-end*: interface, problem modelling, compiling
- *Back-end*: problem solving by reasoning engine

Focus of this talk: **the reasoning engine**

Reasoning: theorem proving, model building

Problems for the back-end reasoner

- From programs to formulæ (via specifications, annotations ...)
- Formula: $H \supset \varphi$
- Problems: determine whether
  - $H \supset \varphi$ is *valid*, i.e.
    $H \models \varphi$, i.e.
    $H \cup \{\neg\varphi\}$ is *unsatisfiable*
    by giving refutation (proof): *theorem proving*;
  - or $H \cup \{\neg\varphi\}$ is *satisfiable*, i.e.
    $H \supset \varphi$ is *not valid*
    by giving model (counter-model): *model building*

Ingredients of formulæ

- Propositional logic (PL): $\vee$, $\neg$, $\wedge$
- Equality: $\simeq$, $\not\simeq$, $a, b, c, \ldots$, $f, g, h, \ldots$
- First-order theories, e.g.:
  - Theories of *data structures*, e.g.:
    - Lists
    - Recursive data structures (with constructors and selectors)
    - Arrays
    - Records
    - Bitvectors
  - Linear arithmetic: $\leq$, $+$, $-$, $\ldots -2, -1, 0, 1, 2, \ldots$
- First-order logic (FOL): $\forall$, $\exists$, $P, Q, R, \ldots$

Reasoning procedures

- Semi-decidable problem: *semi-decision procedure*
- Decidable problem: *decision procedure*

Quantifier-free fragment: ground formulæ

- $\mathcal{T}$-**decision procedure**: decide satisfiability of a ground formula (w.l.o.g. a set of ground clauses S) in a theory $\mathcal{T}$
- $\mathcal{T}$-**satisfiability procedure**: decide satisfiability of a conjunction of ground literals S in $\mathcal{T}$

Desiderata for reasoning procedures

- *Expressive*: handle all ingredients (e.g., all theories) in formula
- *Sound and complete*: no false negatives, no false positives
- *Efficient*: each formula only a sub-task
- *Scalable*: practical problems generate huge formulæ
- *Proof-producing*: check proof, manipulate proof (e.g., extract info for predicate abstraction)
- *Model-producing*: model as counter-example, bug finding

Some reasoning methods

- Davis-Putnam-Logemann-Loveland (DPLL) procedure: case analysis
- Congruence closure (CC) algorithm
- Theory solvers, e.g., Simplex method
- DPLL(T) (e.g., DPLL(EUF) with CC)
- Combination of theory solvers (on top of CC): DPLL-based SMT-solvers
  - Nelson-Oppen method
  - Delayed theory combination
  - Model-based theory combination

More reasoning methods

- Rewriting/Simplification: well-founded ordering, normal/canonical form, matching
- Resolution (unification): deduce clauses (synthetic)
- E-matching, E-unification
- Instance generation
- Tableaux: subgoal-reduction (analytic), model elimination
- Knuth-Bendix completion (Rewriting+Superposition): deduce equations
- Resolution+Rewriting+Superposition/Paramodulation: deduce clauses with equations

Which problems may they be especially good for

- DPLL: SAT-problems; large non-Horn clauses
- CC: ground equations
- Theory solvers: e.g., linear arithmetic, bitvectors
- DPLL-based SMT-solvers: ground SMT-problems
- Rewriting and KB completion: non-ground equations
  Non-ground: with (implicitly) *universally quantified variables*
- Resolution: non-ground FOL clauses, especially Horn
- Resolution+Paramodulation/Superposition+Rewriting: non-ground FOL+= clauses, especially Horn

General theorem proving

Inf: rewrite-based inference system for FOL+=
(e.g., Resolution+Paramodulation/Superposition+Rewriting)

TP strategy: inference system + search plan (e.g., *Inf*-strategy)

Refutationally complete inference system
+
Fair search plan
=
Complete strategy

A rewrite-based approach to SMT: main idea

If *Inf* is guaranteed to **terminate** on any $\mathcal{T}$-satisfiability problem, any complete *Inf*-strategy is a

decision procedure

for $\mathcal{T}$-satisfiability.

- Input: $\mathcal{T} \cup S$, where $\mathcal{T}$ is presentation of theory.
- $\mathcal{T}$ can be union of presentations of theories.
- Non-ground formulæ may migrate from S to $\mathcal{T}$.

Some advantages

- *Sound* and *complete* inference system, *complete* strategies
- *Expressivity*: FOL$+=$ (native quantifier reasoning)
- *Combination of theories*: give union of presentations as input
- *Flexibility* in drawing the line between theory and problem
- Use existing theorem provers "off the shelf"
- *Proof generation*: already there by default
- *Model generation*: final $\mathcal{T}$-satisfiable set as starting point

Our results

- *Termination*: $\mathcal{T}$-sat procedures for data structures theories, with cases of polynomial complexity
- *Combination of theories*: *modularity* of termination
- Some experimental evidence: *efficiency*, *scalability*
- Generalization from $\mathcal{T}$*-satisfiability* to $\mathcal{T}$*-decision* problems
- *Decomposition* approach: FOL+= prover | SMT-solver
  Choose
  what to pre-process by prover,
  what to pass on to solver (e.g., arithmetic, bitvectors)

Termination results

$\mathcal{SP}$: rewrite-based inference system for FOL+=

Complete simplification ordering (CSO) $\succ$ such that $t \succ c$ for all compound terms t and constants c

Complete $\mathcal{SP}_{\succ}$-strategy : $\mathcal{SP}_{\succ}$ + fair search plan

Theorem: A complete $\mathcal{SP}_{\succ}$-strategy is a $\mathcal{T}$-satisfiability procedure.

Covered theories

- Lists
  - *non-empty possibly cyclic* (*polynomial time*)
  - *possibly empty possibly cyclic*
- *Arrays* with or without extensionality
- *Records* with or without extensionality (*polynomial time*)
- Fragments of linear arithmetic:
  - *integer offsets* (*polynomial time*)
  - *integer offsets modulo* (*polynomial time*)
- *Recursive data structures* with one constructor and k selectors:
  - $k = 1$: integer offsets (*pred* and *succ*)
  - $k = 2$: non-empty acyclic lists (*cons*, *car* and *cdr*)

Integer offsets

$$\forall x. \quad \mathsf{s}(\mathsf{p}(x)) \simeq x$$
$$\forall x. \quad \mathsf{p}(\mathsf{s}(x)) \simeq x$$
$$\forall x. \quad \mathsf{s}^i(x) \not\simeq x \qquad \text{for } i > 0$$

s: successor p: predecessor

Infinitely many acyclicity axioms: Problem reduction.

Modularity of termination for combination of theories

Modularity of termination:
if $\mathcal{SP}_{\succ}$-strategy terminates on $\mathcal{T}_i$-sat problems then it terminates on $\mathcal{T}$-sat problems for $\mathcal{T} = \bigcup_{i=1}^{n} \mathcal{T}_i$.

Hypotheses:

- No shared function symbols (shared constants allowed): standard condition
- *Variable-inactive* theories: technical, but simple condition

The modularity theorem

Theorem: if

- $\mathcal{T}_i$, $1 \leq i \leq n$, do not share function symbols
- $\mathcal{T}_i$, $1 \leq i \leq n$, variable-inactive
- $\mathcal{SP}_{\succ}$-strategy is a $\mathcal{T}_i$-satisfiability procedure, $1 \leq i \leq n$,

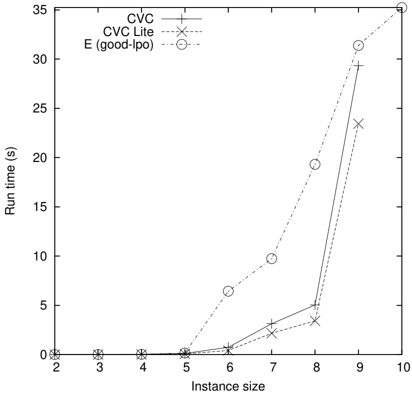
then it is a $\mathcal{T}$-satisfiability procedure.

All above mentioned theories satisfy these hypotheses.

Experiments

- Reasoners: E 0.82, CVC 1.0a, CVC Lite 1.1.0
- Six sets of synthetic parametric benchmarks to test *scalability*
- Both satisfiable and unsatisfiable instances
- Combinations of theories
- Large sets of literals from the UCLID suite

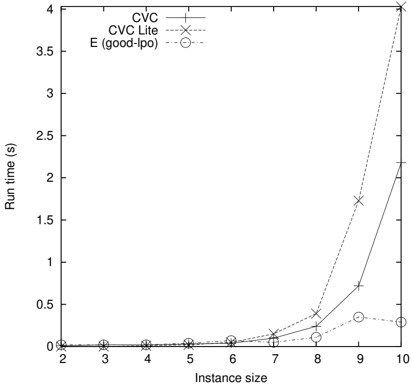
Benchmarks $\texttt{SWAP}(n)$: unsat instances



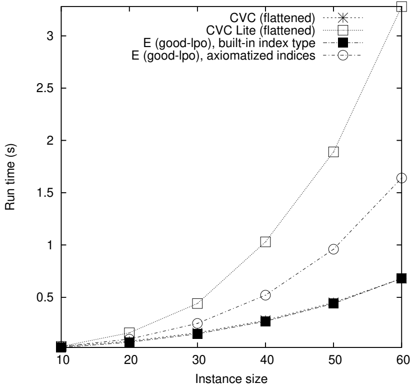
No system terminated for $n \geq 10$

Added lemma for E: additional flexibility

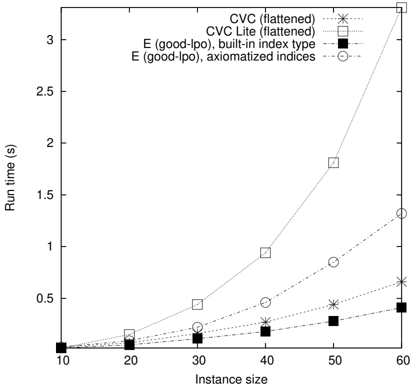
Benchmarks SWAP(n): sat instances



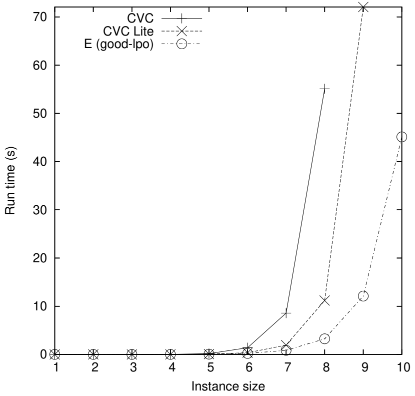
Benchmarks STORECOMM(n): unsat instances



Benchmarks STORECOMM(n): sat instances

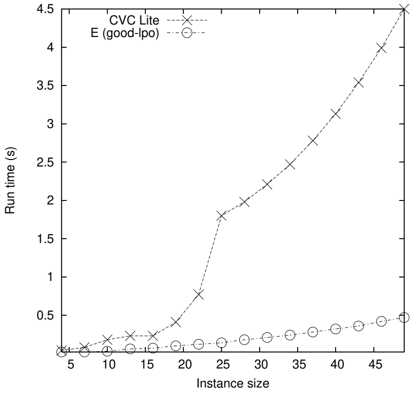


Benchmarks STOREINV(n): unsat instances



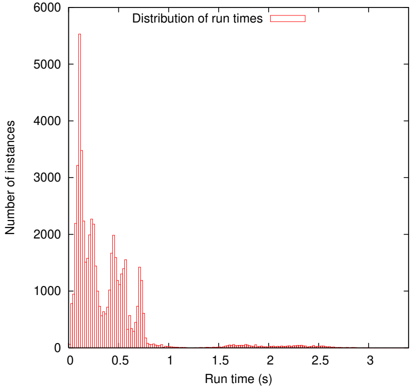
E(std-kbo) does it in *nearly constant time*

Benchmarks CIRCULAR_QUEUE(n, k) instances $k = 3$



CVC did not handle integers mod k

Run time distribution for *E(auto)* on UCLID set



Auto mode: prover chooses search plan by itself

From $\mathcal{T}$-satisfiability to $\mathcal{T}$-decision problems

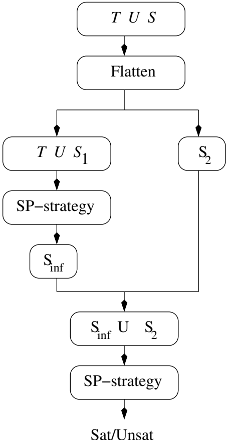
From S conjunction of ground unit clauses to S conjunction of ground clauses.

Theorem: if

- $\mathcal{T}$ is variable inactive
- $\mathcal{SP}_{\succ}$-strategy is $\mathcal{T}$-satisfiability procedure

then it is also $\mathcal{T}$-decision procedure.

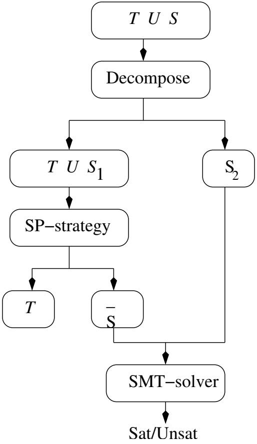
$\mathcal{T}$-decision scheme



Decomposition: unite FOL+= and SMT strengths

- *Decomposition*: definitional and operational part
- *Theory compilation*: apply FOL+= prover to "compile" the definitional part: theory reasoning, non-ground equational reasoning
- *Decision*: apply SMT-solver to subset of saturated set (without $\mathcal{T}$-axioms) + operational part
- Sufficient conditions to preserve satisfiability

$\mathcal{T}$-decision by stages



Summary

- Termination results: $\mathcal{T}$-sat procedures based on generic reasoning
- *Modularity theorem* for combination of theories
- Experiments on $\mathcal{T}$-sat problems with prover *taken off the shelf* and optimized for very different search problems
- *Generalization* to $\mathcal{T}$-decision procedures
- *Decision by stages*: pipeline of FOL$+=$ prover and SMT-solver

Some current and future work

- Experiments with $\mathcal{T}$-decision problems
- More termination results for more (powerful) decision procedures
- Search plans for $\mathcal{T}$-sat and $\mathcal{T}$-decision problems
- Integration with automated model building, especially in combinations of theories

References

- Alessandro Armando, Maria Paola Bonacina, Silvio Ranise and Stephan Schulz. *New results on rewrite-based satisfiability procedures*. *ACM Trans. on Computational Logic*, To appear. (Presented in part at *FroCoS 2005* and *PDPAR 2005*)
- Maria Paola Bonacina and Mnacho Echenim. *On variable-inactivity and polynomial T-satisfiability procedures*. *Journal of Logic and Computation*, 18(1): 77-96, Feb. 2008. (Presented in part at *PDPAR 2006*)
- Maria Paola Bonacina and Mnacho Echenim. *Theory decision by decomposition*. Submitted to journal, April 2008. (Presented in part at *CADE 2007*)

Thanks

Many thanks to Alessandro Armando, Mnacho Echenim, Silvio Ghilardi, Silvio Ranise, Michael Rusinowitch, Stephan Schulz ...

Looking for more

- friends to work with, including post-doc's, students,
- problems, applications, theories to try ...

Thank you!