On a rewriting approach to satisfiability procedures: extension, combination of theories and an experimental appraisal

Maria Paola Bonacina[1]

Dipartimento di Informatica
Università degli Studi di Verona

19th of September, 2005

[1] Joint work with Alessandro Armando (Università degli Studi di Genova), Silvio Ranise (INRIA Lorraine) and Stephan Schulz (Università degli Studi di Verona)

Decision procedures

- **Objective**: Decision procedures for application of automated reasoning to verification
- **Desiderata**: Fast, expressive, easy to use, extend, integrate, prove sound and complete
- **Issues**:
  - Combination of theories:
    usually done by combining procedures: complicated? *ad hoc*?
  - Soundness and completeness proof: usually *ad hoc*
  - Implementation: usually from scratch: correctness? integration in different environments? duplicated work?

"Little" engines and "big" engines of proof

- "Little" engines, e.g., validity checkers for specific theories
  Built-in theory, quantifier-free conjecture, decidable
- "Big" engines, e.g., general first-order theorem provers
  Any first-order theory, any conjecture, semi-decidable
- Not an issue of size (e.g., lines of code) of systems!
- Continuity: e.g., "big" engines may have theories built-in
- **Challenge**: can we get something good for decision procedures from big engines?

From a big-engine perspective

- *Combination of theories*: give union of presentations as input to prover
- *Soundness and completeness proof*: already given for first-order inference system
- *Implementation*: take first-order provers off the shelf
- *Proof generation*: already there by default
- *Model generation*: final T-sat set (starting point)
- **How to make it possible?**

Motivation

Rewrite-based satisfiability: new results

- A rewrite-based methodology for T-satisfiability
- Theories of data structures
- A modularity theorem for combination of theories

Experimental appraisal

- Comparison of E with CVC and CVC Lite
- Synthetic benchmarks (valid and invalid): evaluate scalability
- "Real-world" problems

Summary

What kind of theorem prover?

First-order logic with equality

$\mathcal{SP}$ inference system: rewrite-based

- *Simplification by equations*: normalize clauses
- *Superposition*: generate clauses

Complete simplification ordering (CSO) $\succ$ on terms, literals and clauses: $\mathcal{SP}_{\succ}$

(Fair) $\mathcal{SP}_{\succ}$*-strategy* : $\mathcal{SP}_{\succ}$ + (fair) search plan

Rewrite-based methodology for T-satisfiability

- *T-satisfiability*: decide satisfiability of set S of ground literals in theory (or combination) T
- *Methodology*:
  - *T-reduction*: apply inferences (e.g., to remove certain literals or symbols) to get equisatisfiable *T-reduced* problem
  - *Flattening*: flatten all ground literals (by introducing new constants) to get equisatisfiable T-reduced *flat* problem
  - *Ordering selection and termination*: prove that any fair $\mathcal{SP}_{\succ}$-strategy terminates when applied to a $\mathcal{T}$-reduced flat problem, provided $\succ$ is *$\mathcal{T}$-good*
- Everything *fully automated* except for termination proof

Covered theories

- *EUF*, *lists*, *arrays* with and without extensionality, *sets* with extensionality [Armando, Ranise, Rusinowitch 2003]
- *Records* with and without extensionality, *integer offsets*, *integer offsets modulo* [Armando, Bonacina, Ranise, Schulz 2005]

In experiments: arrays, records, integer offsets, integer offsets modulo, EUF and combinations

Records: presentation

Sort $\mathrm{REC}(id_1 : \mathrm{T}_1, \ldots, id_n : \mathrm{T}_n)$

$$
\begin{array}{lcl}
\forall x, v. & \mathsf{rselect}_i(\mathsf{rstore}_i(x, v)) \simeq v & 1 \leq i \leq n \\
\forall x, v. & \mathsf{rselect}_j(\mathsf{rstore}_i(x, v)) \simeq \mathsf{rselect}_j(x) & 1 \leq i \neq j \leq n \\
\forall x, y. & (\bigwedge_{i=1}^{n} \mathsf{rselect}_i(x) \simeq \mathsf{rselect}_i(y) \supset x \simeq y) &
\end{array}
$$

where x, y have sort REC and v has sort T_i.
Extensionality is the third axiom.

Records: termination of $\mathcal{SP}$

$\mathcal{R}$-reduction: eliminate disequalities between records by resolution with extensionality + splitting.

$\mathcal{R}$-good: $t \succ c$ for all ground compound terms t and constants c.

Termination: case analysis of generated clauses (CSO plays key role).

Theorem: A fair $\mathcal{R}$-good $\mathcal{SP}_{\succ}$-strategy is a satisfiability procedure for the theories of records and records with extensionality.

Integer offsets: presentation

A fragment of the theory of the integers:

s: successor

p: predecessor

$$
\begin{array}{lll}
\forall x. & \mathrm{s}(\mathrm{p}(x)) \simeq x & \\
\forall x. & \mathrm{p}(\mathrm{s}(x)) \simeq x & \\
\forall x. & \mathrm{s}^i(x) \not\simeq x & \text{for } i > 0
\end{array}
$$

Infinitely many *acyclicity axioms*!

Integer offsets: termination of $\mathcal{SP}$

$\mathcal{I}$-reduction: eliminate p by replacing $\mathrm{p}(c) \simeq d$ with $c \simeq \mathrm{s}(d)$:
first two axioms no longer needed.
Bound the number of acyclicity axioms:
$\forall x.\, \mathrm{s}^i(x) \not\simeq x$ for $0 < i \leq n+1$
if there are n occurrences of s.

$\mathcal{I}$-good: any CSO.
Termination: case analysis of generated clauses.

Theorem: A fair $\mathcal{SP}_{\succ}$-strategy is a satisfiability procedure for the theory of integer offsets.

Integer offsets modulo: presentation

To reason with indices ranging over the integers mod k ($k > 0$):

$$
\begin{array}{ll}
\forall x. & s(p(x)) \simeq x \\
\forall x. & p(s(x)) \simeq x \\
\forall x. & s^i(x) \not\simeq x \quad 1 \leq i \leq k-1 \\
\forall x. & s^k(x) \simeq x
\end{array}
$$

Finitely many axioms.

Integer offsets modulo: termination of $\mathcal{SP}$

$\mathcal{I}$-reduction: same as above.

$\mathcal{I}$-good: any CSO.

Termination: case analysis of generated clauses.

Theorem: A fair $\mathcal{SP}_{\succ}$-strategy is a satisfiability procedure for the theory of integer offsets modulo.

Termination also without $\mathcal{I}$-reduction.

A modularity theorem for combination of theories

- *Modularity*: if $\mathcal{SP}_{\succ}$-strategy decides $\mathcal{T}_i$-sat problems then it decides $\mathcal{T}$-sat problems, $\mathcal{T} = \bigcup_{i=1}^{n} \mathcal{T}_i$
- $\mathcal{T}_i$-reduction and flattening apply as for each theory
- Termination?

Three simple conditions

- $\succ$ $\mathcal{T}$-*good*, if $\mathcal{T}_i$-*good* for all i, $1 \leq i \leq n$
- The $\mathcal{T}_i$ do not share function symbols
  (*Intuition*: no superposition from compound terms across theories)
- Each $\mathcal{T}_i$ is *variable-inactive*:
  no maximal literal in a ground instance of a clause is instance of an equation $t \simeq x$ where $x \notin Var(t)$
  (*Intuition*: no superposition from variables across theories, since for $t \simeq x$ where $x \in Var(t)$, $t \succ x$)

A modularity theorem

Theorem: if

- No shared function symbol (shared constants allowed),
- Variable-inactive presentations $\mathcal{T}_i$, $1 \leq i \leq n$,
- Fair $\mathcal{T}_i$-good $\mathcal{SP}_{\succ}$-strategy is satisfiability procedure for $\mathcal{T}_i$,

then
a fair $\mathcal{T}$-good $\mathcal{SP}_{\succ}$-strategy is a satisfiability procedure for $\mathcal{T}$.

EUF, *arrays* (with or without extensionality), *records* (with or without extensionality), *integer offsets* and *integer offsets modulo*, all satisfy these hypotheses.

Two remarks on generality

- *Purely equational theories*:
  no trivial models $\Rightarrow$ variable-inactive
- *First-order theories*: variable-inactive excludes, e.g.,
  $a_1 \simeq x \vee \ldots \vee a_n \simeq x$, a_i constants (*)
  Such a clause means *not stably-infinite*, hence *not convex* under the no trivial models hypothesis:
  if $\mathcal{T}_i$ not variable-inactive for (*), Nelson-Oppen does not apply either.

Experimental setting

- Three systems:
  - The E theorem prover: E 0.82 [Schulz 2002]
  - CVC 1.0a [Stump, Barrett and Dill 2002]
  - CVC Lite 1.1.0 [Barrett and Berezin 2004]
- Generator of pseudo-random instances of synthetic benchmarks
- 3.00GHz 512MB RAM Pentium 4 PC: max 150 sec and 256 MB per run
- **Folklore**: systems with built-in theories are out of reach for prover with presentation as input ...

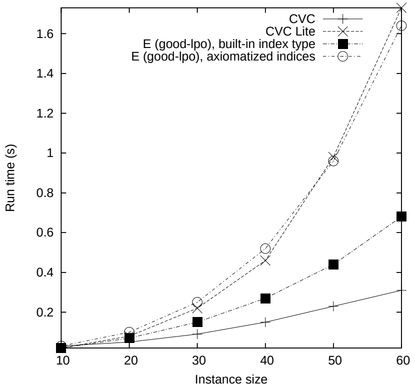
Synthetic benchmarks

- $\mathtt{STORECOMM}(n)$, $\mathtt{SWAP}(n)$, $\mathtt{STOREINV}(n)$: arrays with extensionality
- $\mathtt{IOS}(n)$: arrays and integer offsets
- $\mathtt{QUEUE}(n)$: records, arrays, integer offsets
- $\mathtt{CIRCULAR\_QUEUE}(n,k)$: records, arrays, integer offsets mod k

$\mathtt{STORECOMM}(n)$, $\mathtt{SWAP}(n)$, $\mathtt{STOREINV}(n)$: both valid and invalid instances

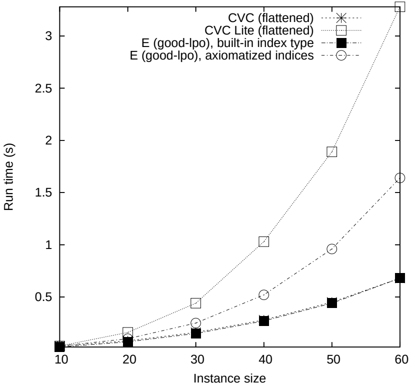
Parameter n: test *scalability*

Performances on valid STORECOMM(n) instances



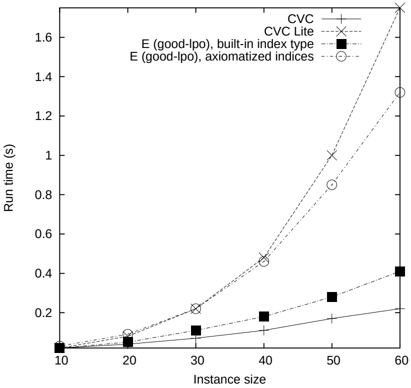
Native input: CVC wins but E better than CVC Lite

Performances on valid STORECOMM(n) instances



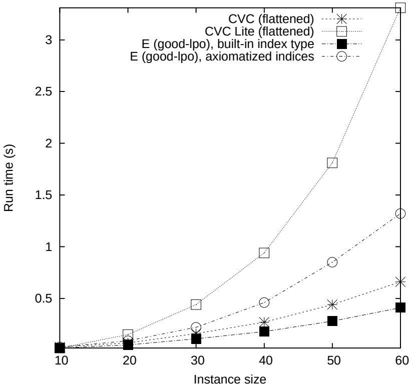
Flat input: E matches CVC

Performances on invalid STORECOMM(n) instances



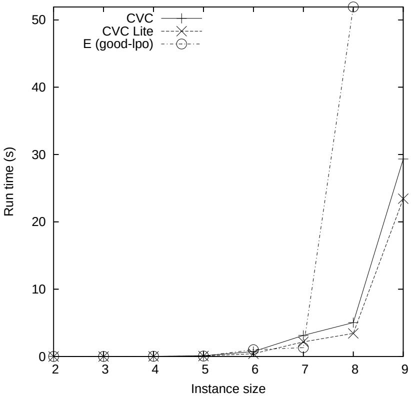
Native input: prover conceived for unsat handles sat even better

Performances on invalid $\mathtt{STORECOMM}(n)$ instances



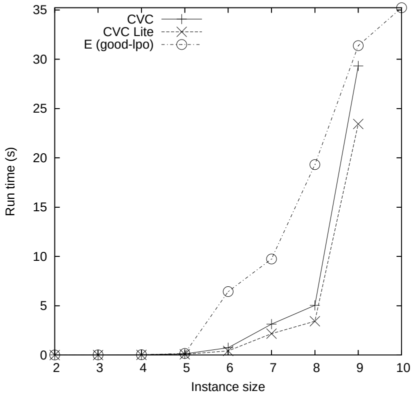
Flat input: E surpasses CVC

Performances on valid $\mathtt{SWAP}(n)$ instances



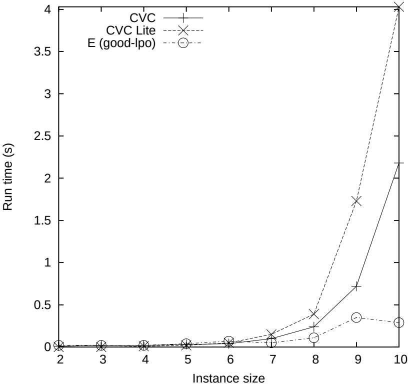
Harder problem: no system terminates for $n \geq 10$

Performances on valid SWAP(n) instances



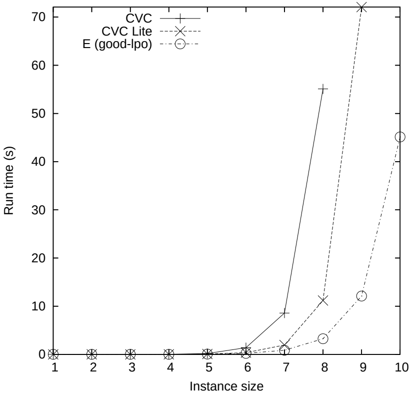
Added lemma for E: additional flexibility for the prover

Performances on invalid SWAP(n) instances



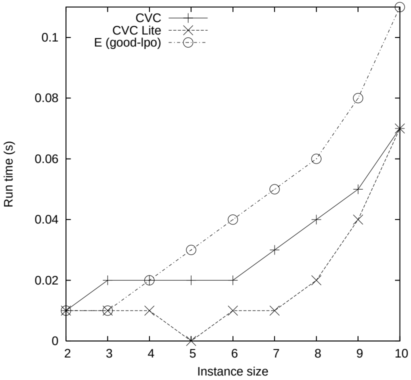
Easier problem, but E clearly ahead

Performances on valid $\texttt{STOREINV}(n)$ instances



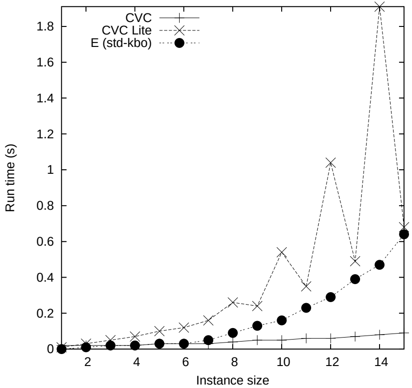
E(std-kbo) does it in *nearly constant time*!

Performances on invalid STOREINV(n) instances



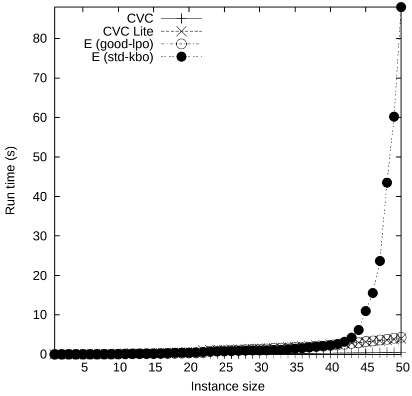
Not as good for E but run times are minimal

Performances on IOS instances



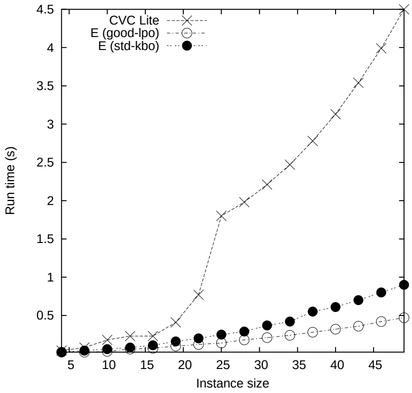
CVC and CVC Lite have built-in $\mathcal{LA}(\mathcal{R})$ and $\mathcal{LA}(\mathcal{I})$ respectively!

Performances on QUEUE instances (plain queues)



CVC wins (built-in arithmetic!) but E matches CVC Lite

Performances on $\mathtt{CIRCULAR\_QUEUE}(n, k)$ instances $k = 3$



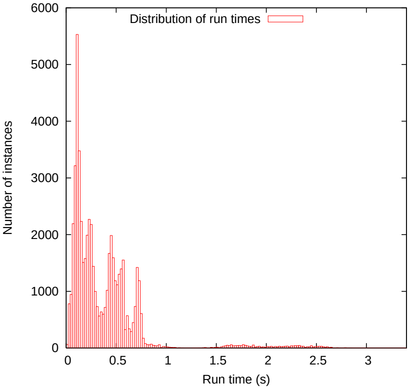
CVC does not handle integers mod k, E clearly wins

"Real-world" problems

- UCLID [Bryant, Lahiri, Seshia 2002]: suite of problems
- haRVey [Déharbe and Ranise 2003]: extract T-sat problems
- over 55,000 proof tasks: integer offsets and equality
- all valid

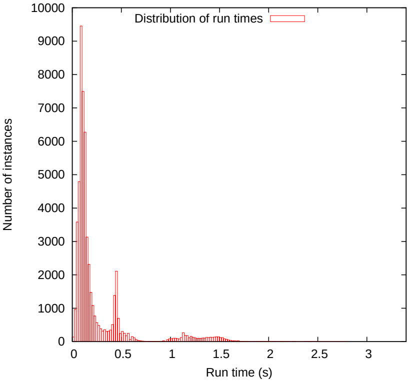
Test performance on huge sets of literals.

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



Auto mode: prover chooses search plan by itself

Better run time distribution for E on UCLID set



Optimized strategy: found by testing on random sample of 500 problems (less than 1%)

Summary

- *General methodology* for rewrite-based T-sat procedures and its application to several theories of data structures
- *Modularity theorem* for combination of theories
- Experiments: first-order prover
  - *taken off the shelf* and
  - conceived for very different search problems

  compares amazingly well with state-of-the-art verification tools

Directions for further research

- Prover's search plans for T-sat problems
- More or stronger termination results
- Precise relationship between variable-inactive and stably-infinite, convex
- Integration with approaches for full $\mathcal{LA}$ or bit-vectors
- $\mathcal{T}$-decision procedures (arbitrary quantifier-free formulæ): integration with SAT-solver? Other approaches?
- Combination with automated model building
- In general: explore "big" engines technology for decision procedures

Big picture

Reasoning environments for verification (and more):

- SAT-solvers
- "Little" engines
- "Big" engines
- Good interfaces
- ... ...