Boolean Abstraction for Temporal Logic Satisfiability

A. Cimatti[1], M. Roveri[1], *V. Schuppan*[1], S. Tonetta[2]

[1]FBK-irst, Trento, Italy

[2]University of Lugano, Faculty of Informatics, Lugano, Switzerland

CAV'07, July 3–7, 2007, Berlin, Germany

Motivation

$\Rightarrow$ Property-based system design (PROSYD): work at the level of requirements.

$\Rightarrow$ In model checking, focus is on dealing with complexity in the model.

$\Rightarrow$ Satisfiability of large temporal formulas can be hard. (e.g., [Rozier, Vardi (SPIN'07)])

© 2007 V. Schuppan

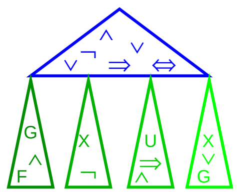
Contents

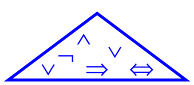
© 2007 V. Schuppan

Boolean Abstraction

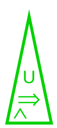
(well-known in SMT community)

temporal formula





Boolean combination

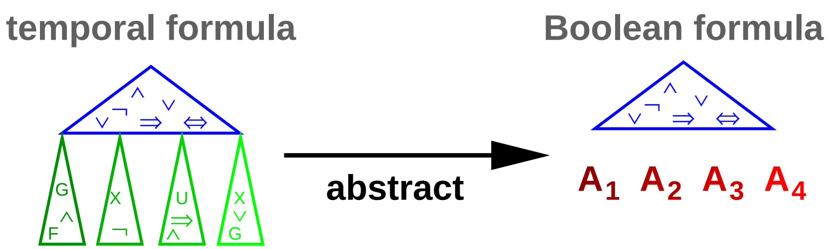


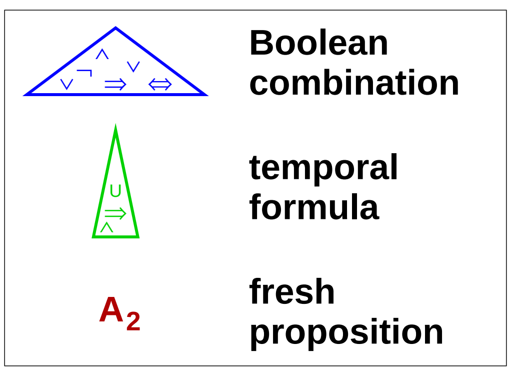
temporal formula

© 2007 V. Schuppan

Boolean Abstraction

(well-known in SMT community)

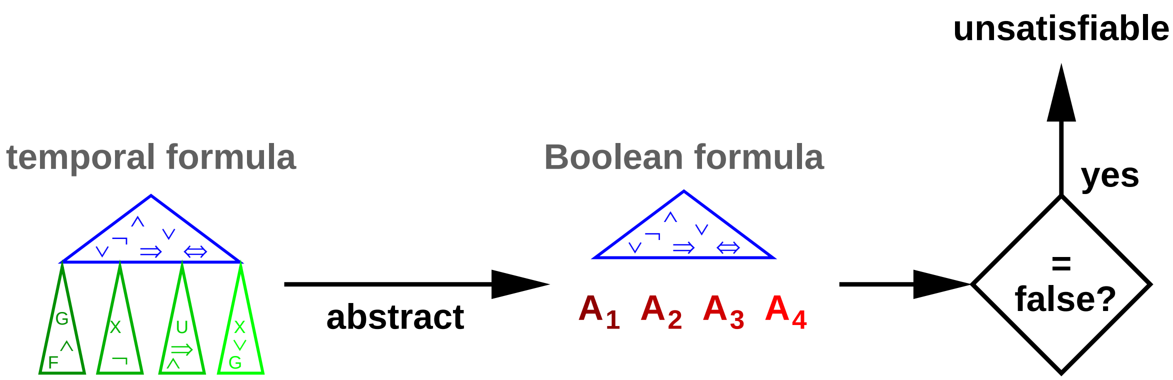


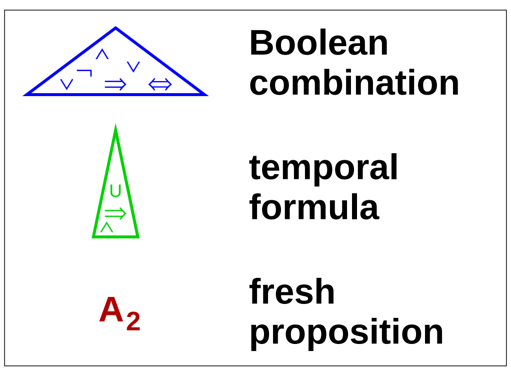


© 2007 V. Schuppan

Boolean Abstraction

(well-known in SMT community)

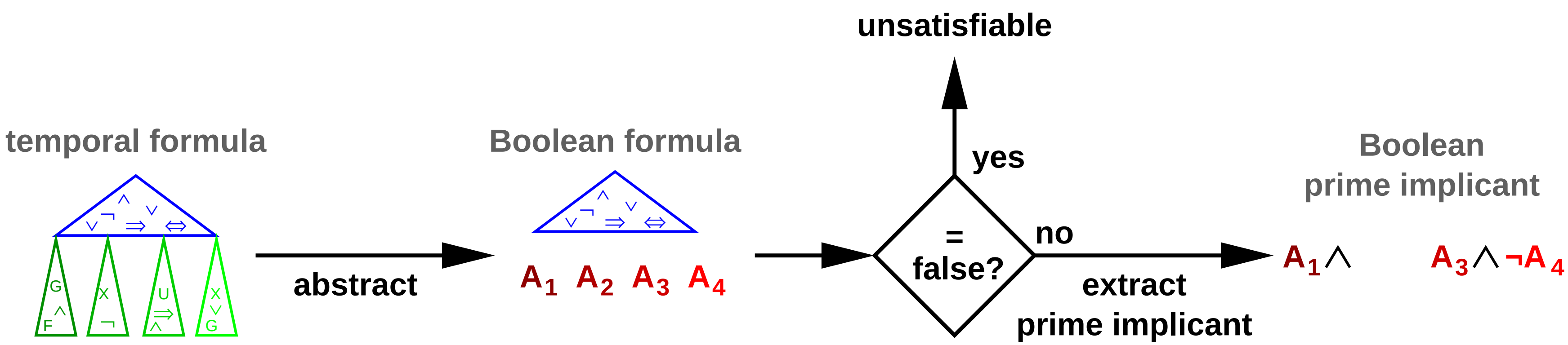


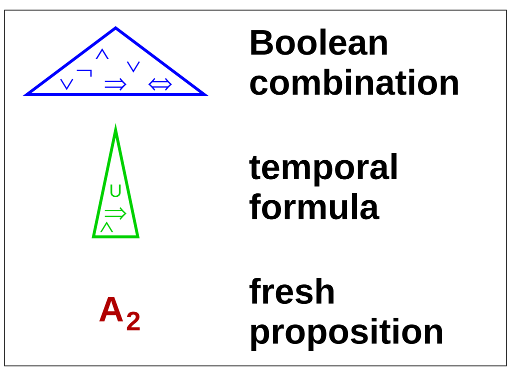


© 2007 V. Schuppan

Boolean Abstraction

(well-known in SMT community)

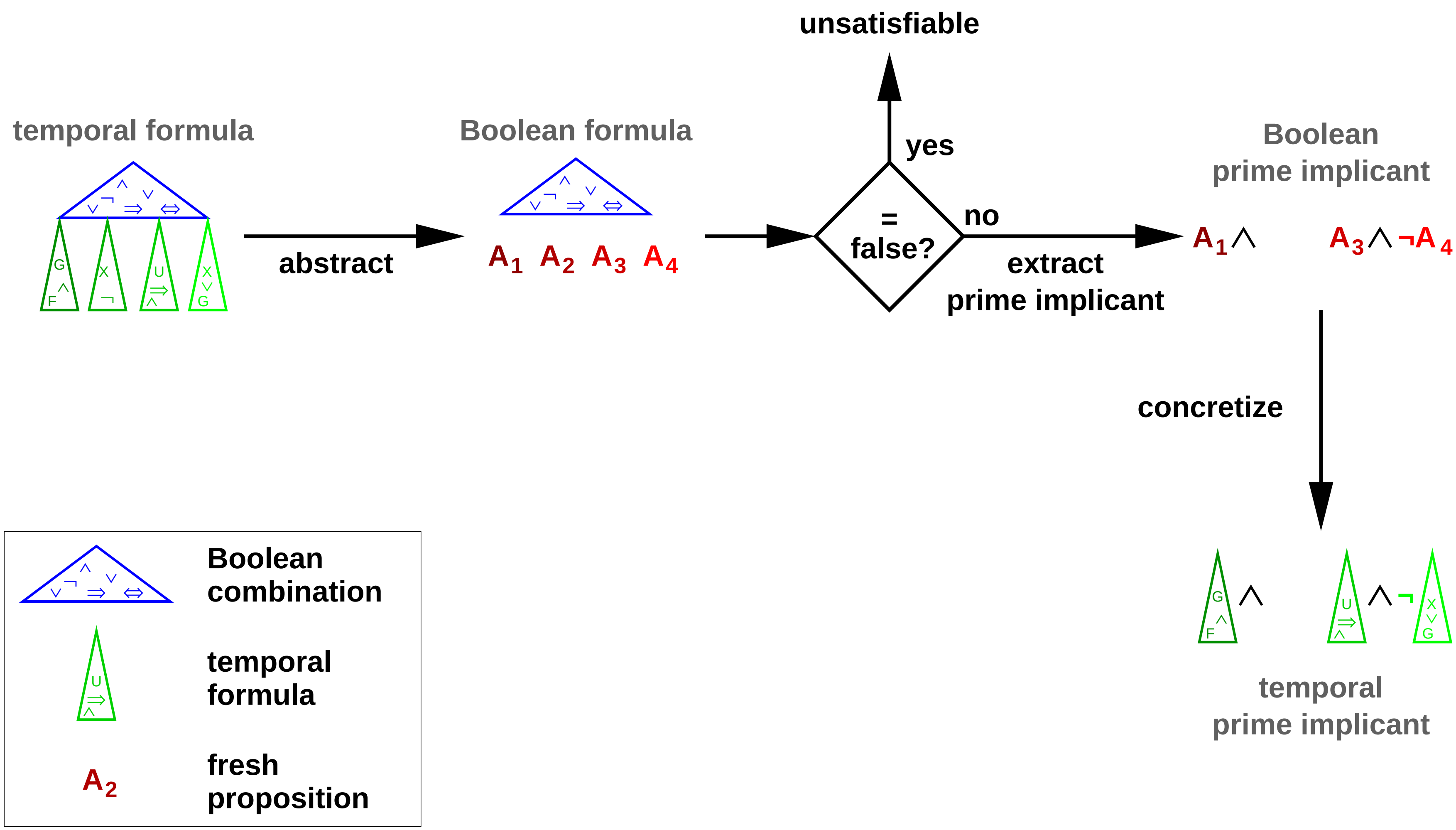




© 2007 V. Schuppan

Boolean Abstraction

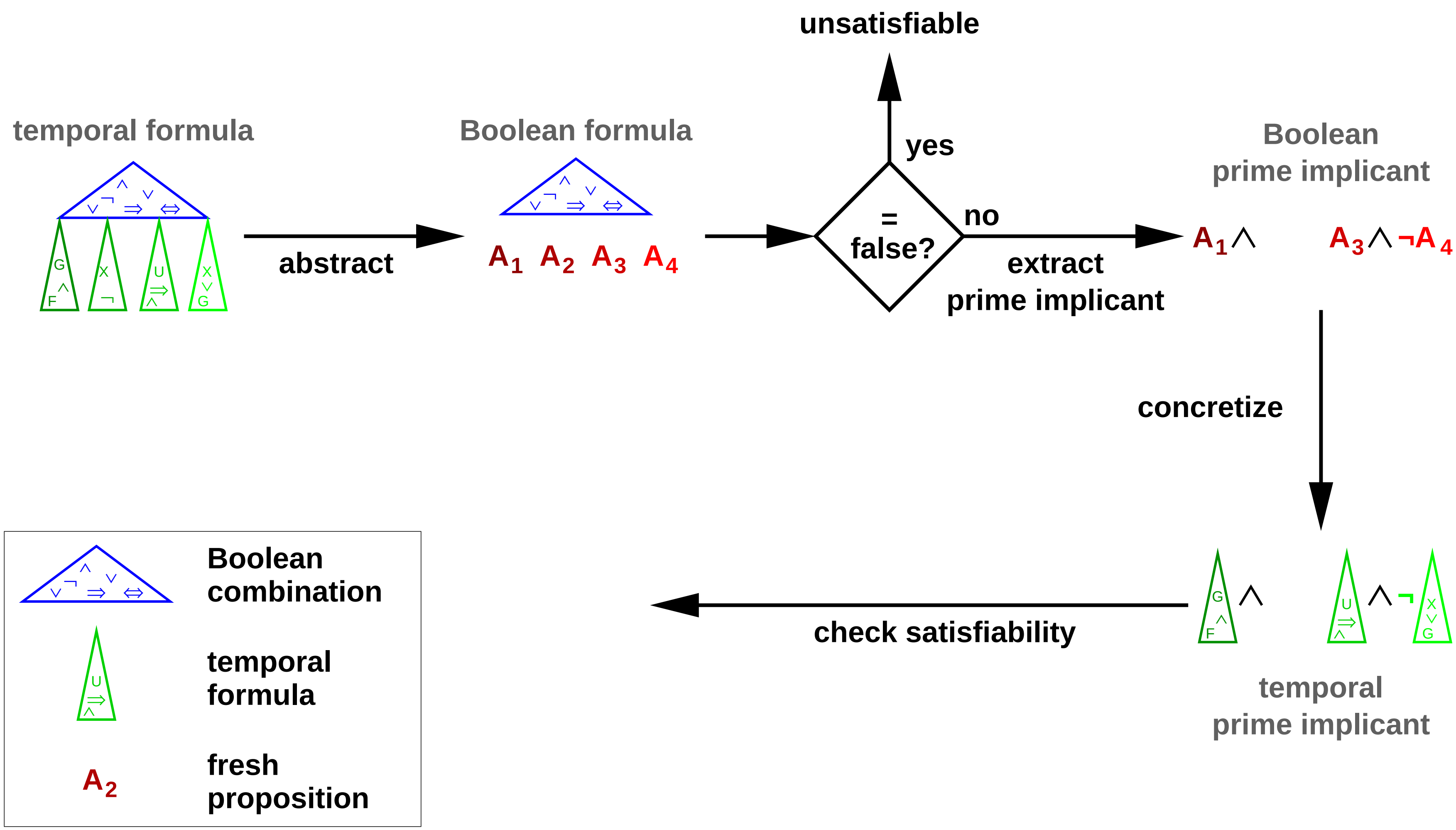
(well-known in SMT community)



© 2007 V. Schuppan

Boolean Abstraction

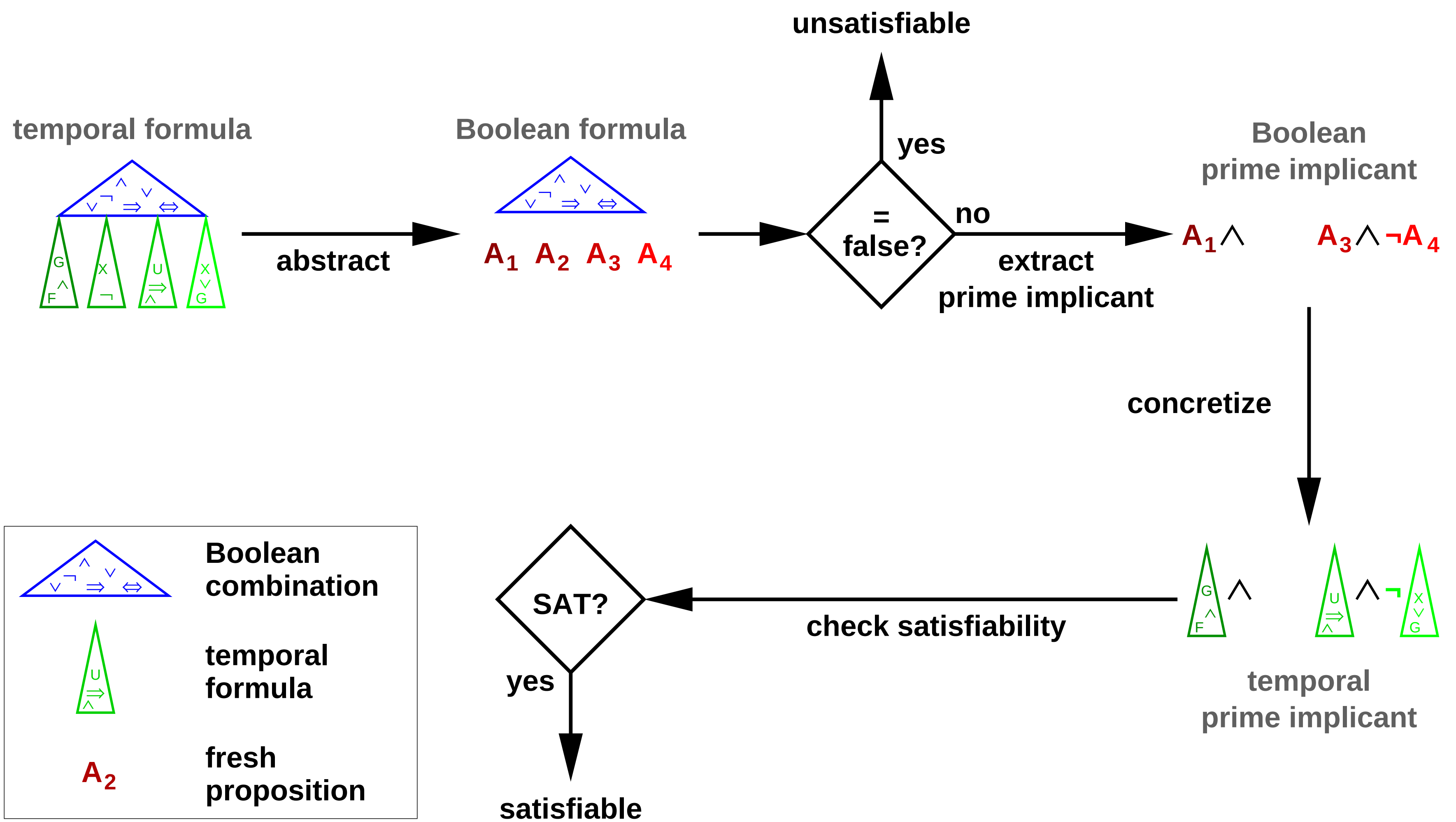
(well-known in SMT community)



© 2007 V. Schuppan

Boolean Abstraction

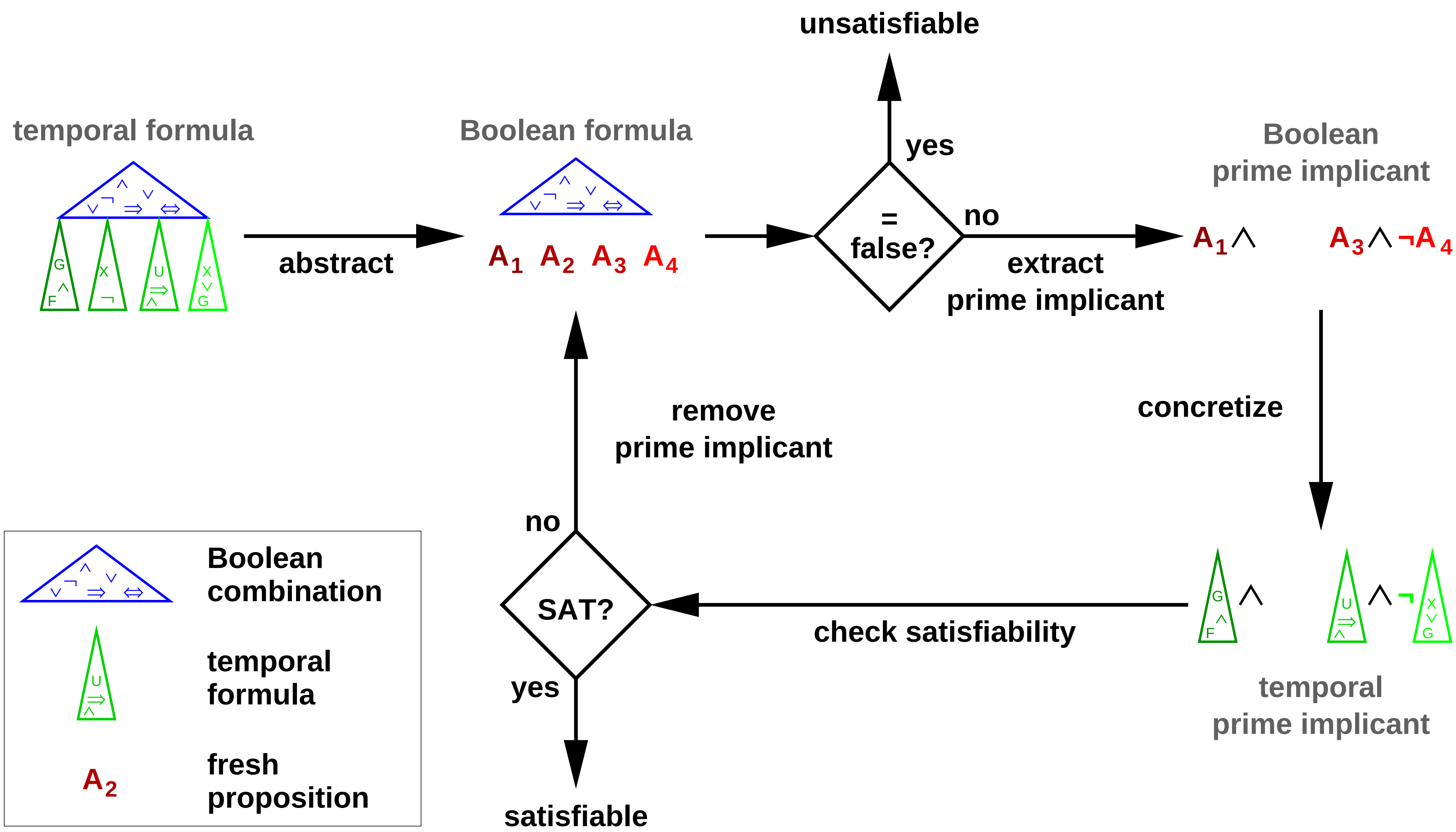
(well-known in SMT community)



© 2007 V. Schuppan

Boolean Abstraction

(well-known in SMT community)



© 2007 V. Schuppan

Contents

ⓒ 2007 V. Schuppan

Pure Literal Simplification — Propositional Logic

[Davis, Putnam (1960); Dunham, Fridshal, Sward (1959)]

Assume a propositional formula ϕ in CNF:

$$\underbrace{(l_{1,1} \vee \ldots \vee l_{1,n_1} \vee p) \wedge \ldots \wedge (l_{k,1} \vee \ldots \vee l_{k,n_k} \vee p)}_{\phi_1\,:\; p \text{ occurs only positively}} \quad \wedge \quad \underbrace{\phi_2}_{\text{no occurrence of } p}$$

Then: ϕ is satisfiable iff $p \wedge \phi$ is satisfiable.

(And similarly if p occurs only negatively in ϕ_1.)

© 2007 V. Schuppan

Pure Literal Simplification — PSL

Extend notion of pure literal to PSL (see paper).

Let ϕ be a PSL formula such that p is pure positive in ϕ.

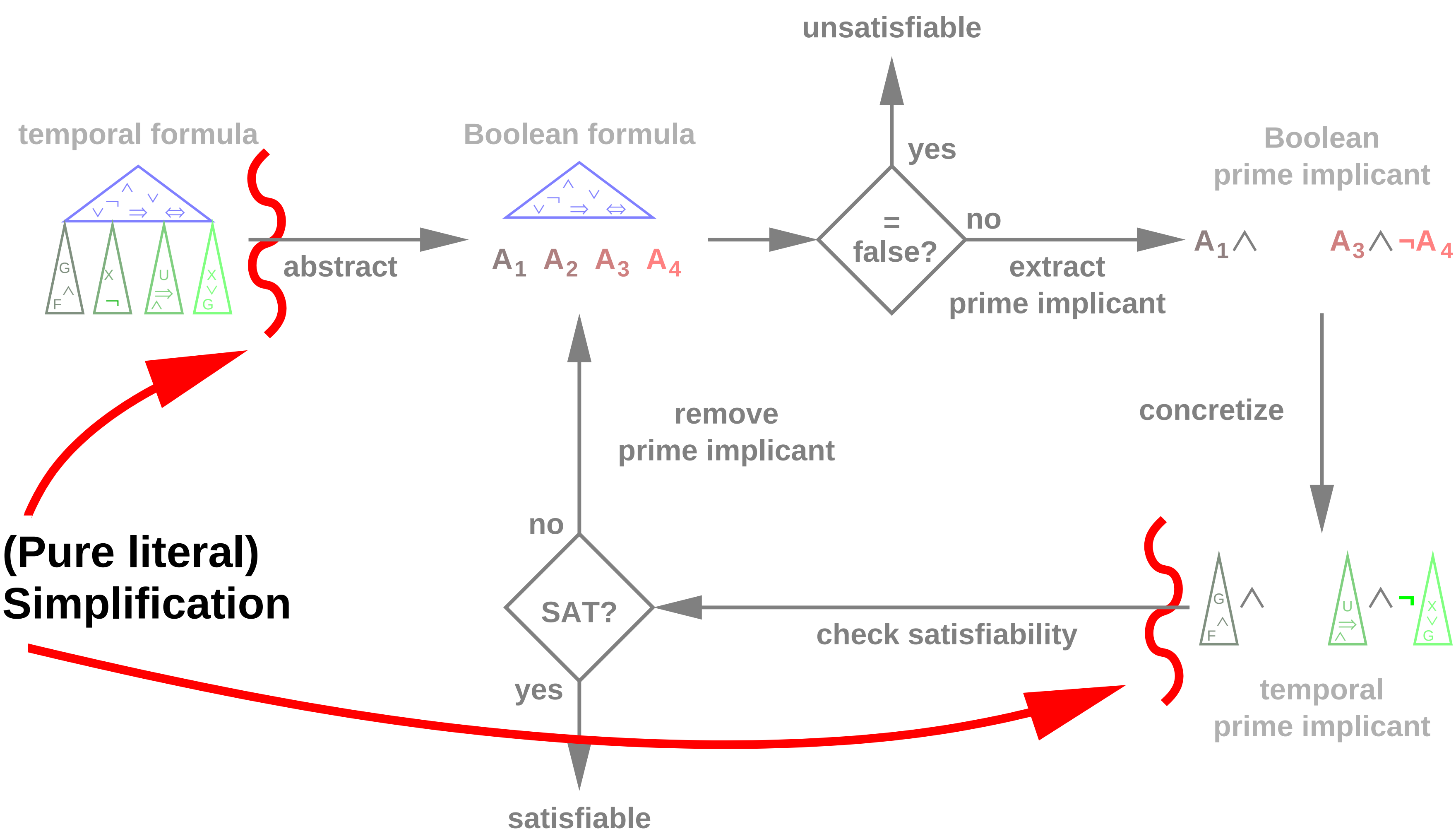
Then: ϕ is satisfiable iff $(\mathbf{G}p) \wedge \phi$ is satisfiable.

(And similarly if p is pure negative in ϕ.)

(Modal logic $\mathcal{K}$: [Pan, Sattler, Vardi (J. Applied Non-Classical Logics 2006)])

© 2007 V. Schuppan

Boolean Abstraction and Pure Literal Simplification



(Pure literal)
Simplification

© 2007 V. Schuppan

Contents

© 2007 V. Schuppan

Unsatisfiable Cores

Assume

$$\phi \equiv (\mathbf{G}p) \wedge (\mathbf{F}\neg p) \wedge ((\mathbf{X}p) \vee (\mathbf{XX}p))$$

Prime implicants:

$$(\mathbf{G}p) \wedge (\mathbf{F}\neg p) \wedge (\mathbf{X}p)$$

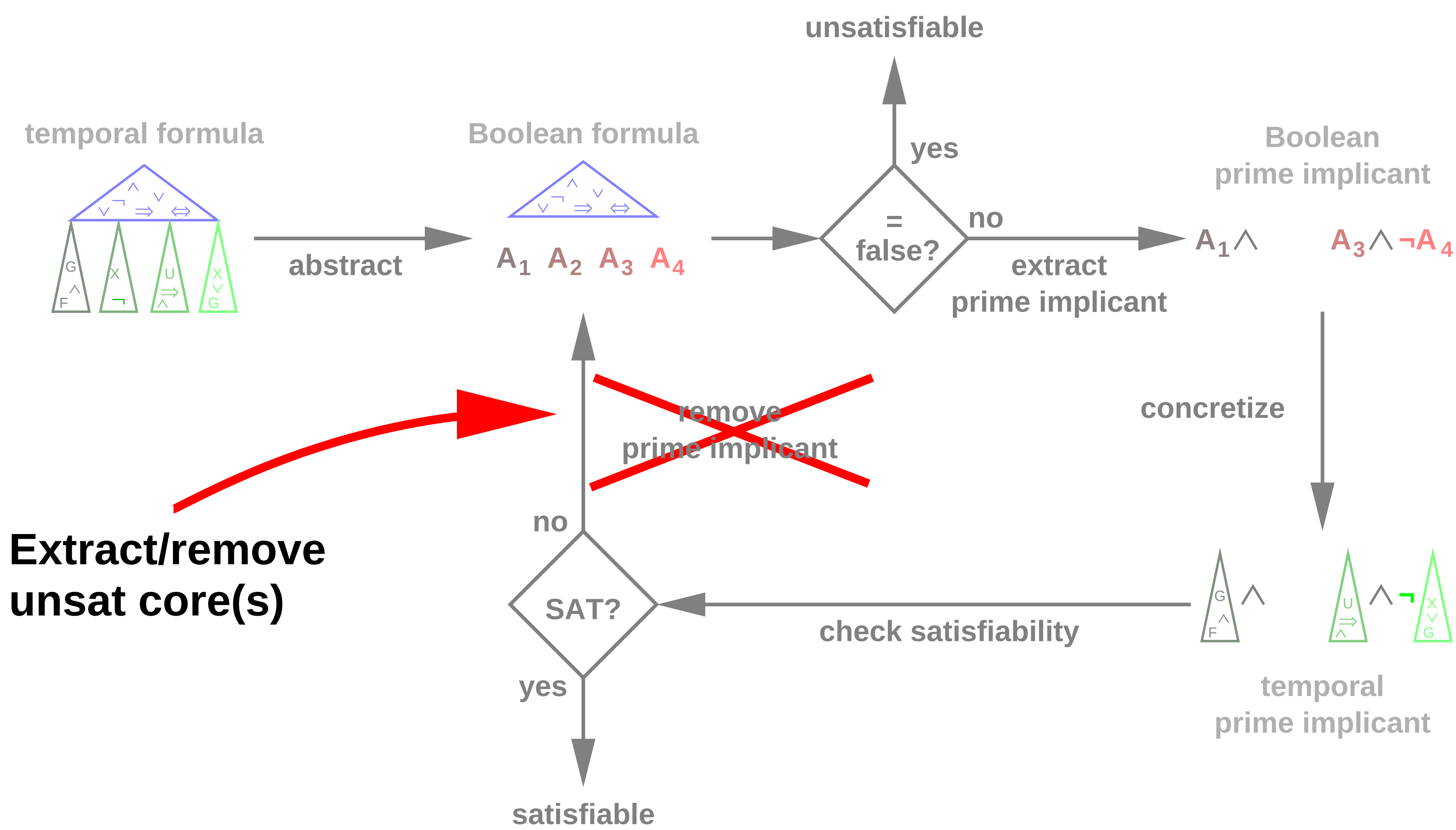
$$(\mathbf{G}p) \wedge (\mathbf{F}\neg p) \wedge (\mathbf{XX}p)$$

They share unsatisfiable part $\Rightarrow$ no need to check both!

Given $\{\phi_i \mid i \in I\}$ with $\bigwedge_{i \in I} \phi_i$ unsatisfiable,
any $\{\phi_j \mid j \in J \subseteq I\}$ with $\bigwedge_{j \in J} \phi_j$ unsatisfiable is an unsatisfiable core.

© 2007 V. Schuppan

Boolean Abstraction and Unsat Core Extraction



© 2007 V. Schuppan

Activation Variables

Propositional case: [Lynce, Marques-Silva (SAT'04)]

1. Assume prime implicant $\bigwedge_{i \in I} \phi_i$.

2. Introduce one fresh, Boolean activation variable A_i per ϕ_i.

3. Build Büchi automaton B for

$$\bigwedge_{i \in I} (A_i \rightarrow \phi_i)$$

Let $J \subseteq I$. B has fair path from some initial state with $\{A_j \mid j \in J\}$ true

iff

$\bigwedge_{j \in J} \phi_j$ is satisfiable.

Independent of how Büchi automaton is constructed!

© 2007 V. Schuppan

Extracting Unsatisfiable Cores with BDD-based Solvers

Let B be a Büchi automaton for $\bigwedge_{i \in I}(A_i \rightarrow \phi_i)$.

1. Let S be the set of states in B that are the start of a fair path (e.g., Emerson-Lei).
2. Restrict S to initial states in B.
3. Project S onto $\{A_i \mid i \in I\}$.
4. Complement S.

Now S contains the set of unsatisfiable cores of $\bigwedge_{i \in I} \phi_i$.

(We obtain all unsatisfiable cores.)

© 2007 V. Schuppan

Extracting Unsatisfiable Cores with SAT-based Solvers

Let B be a Büchi automaton for $\bigwedge_{i \in I}(A_i \rightarrow \phi_i)$.

1. Let $k \leftarrow 0$.
2. Encode feasibility of loop-free path of length k in B.
3. Check satisfiability assuming $\{A_i \mid i \in I\}$ is true at time 0.
4. If unsat, obtain conflict in terms of assumptions $\{A_j \mid j \in J \subseteq I\}$ at time 0.
5. Otherwise, increase k and repeat.

Now $\{\phi_j \mid j \in J\}$ contains an unsatisfiable core of $\bigwedge_{i \in I} \phi_i$.

(We obtain one unsatisfiable core.)

© 2007 V. Schuppan

Contents

© 2007 V. Schuppan

Benchmarks on PSL satisfiability

(Used in [Cimatti, Roveri, Semprini, Tonetta (FMCAD'06); Cimatti, Roveri, Tonetta (TACAS'07)])

1. Fill typical patterns extracted from industrial specifications [Ben-David, Orni (2005)] with random regular expressions.

2. Generate benchmarks by aggregating patterns from step 1 into the following shapes:
   - large conjunction,
   - (large conjunction) implies (large conjunction),
   - (large conjunction) iff (large conjunction),
   - random Boolean combination.

We'd love to have challenging realistic benchmarks from industry.

© 2007 V. Schuppan

Experiments

Implementation

- Basis: NuSMV
- Translation from PSL to automata: [Cimatti, Roveri, Tonetta (TACAS'07)]
- BDD-based solver: backward Emerson-Lei, dynamic reordering
  baseline for BDD-based approaches
- SAT-based solver: incremental and complete SBMC with MiniSat
  [Heljanko, Junttila, Latvala (CAV'05)]
  baseline for SAT-based approaches

Resources

- Time out: 120 seconds
- Memory out: 768 MB

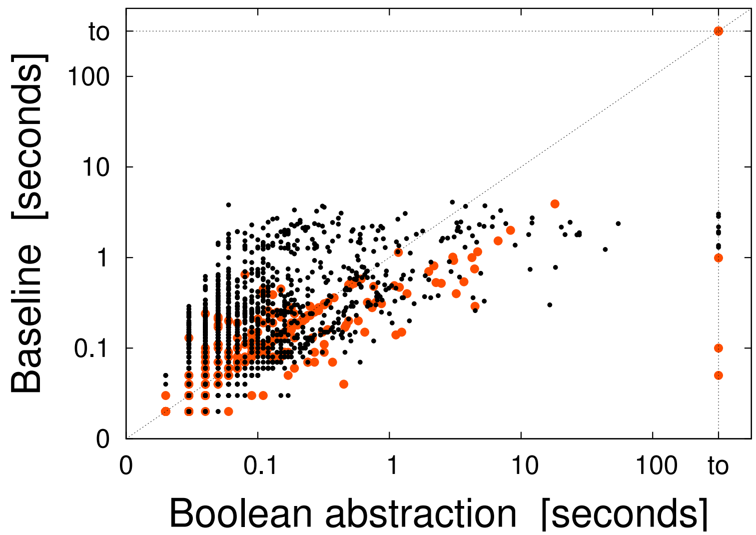
Download

`http://sra.itc.it/people/roveri/cav07-bapsl/`

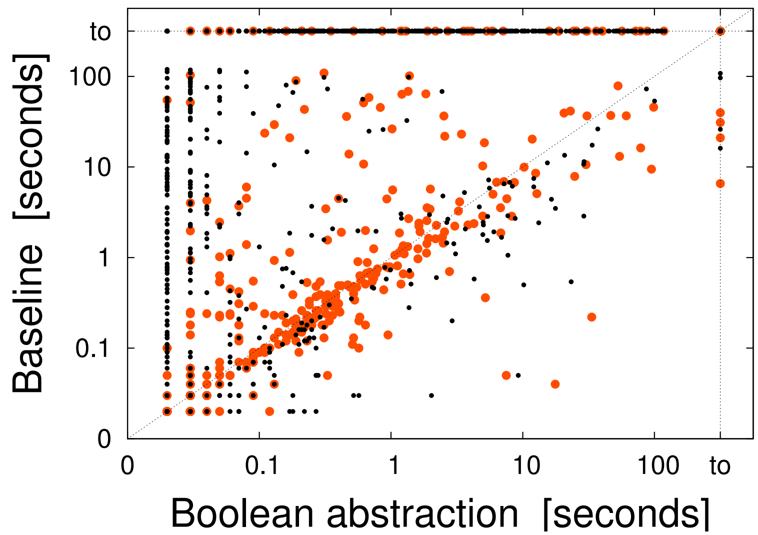
© 2007 V. Schuppan

Boolean abstraction vs. not

SAT



BDD



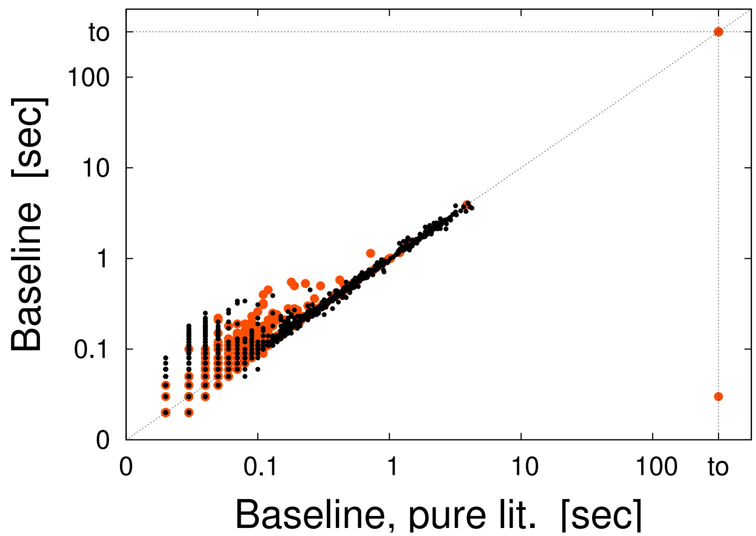
● unsat

• sat

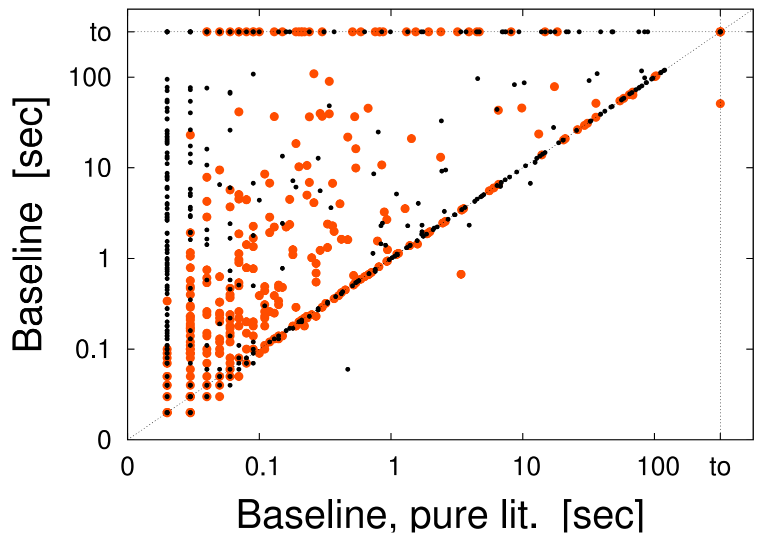
© 2007 V. Schuppan

Pure literal simplification vs. not (without Boolean abstraction)

SAT



BDD

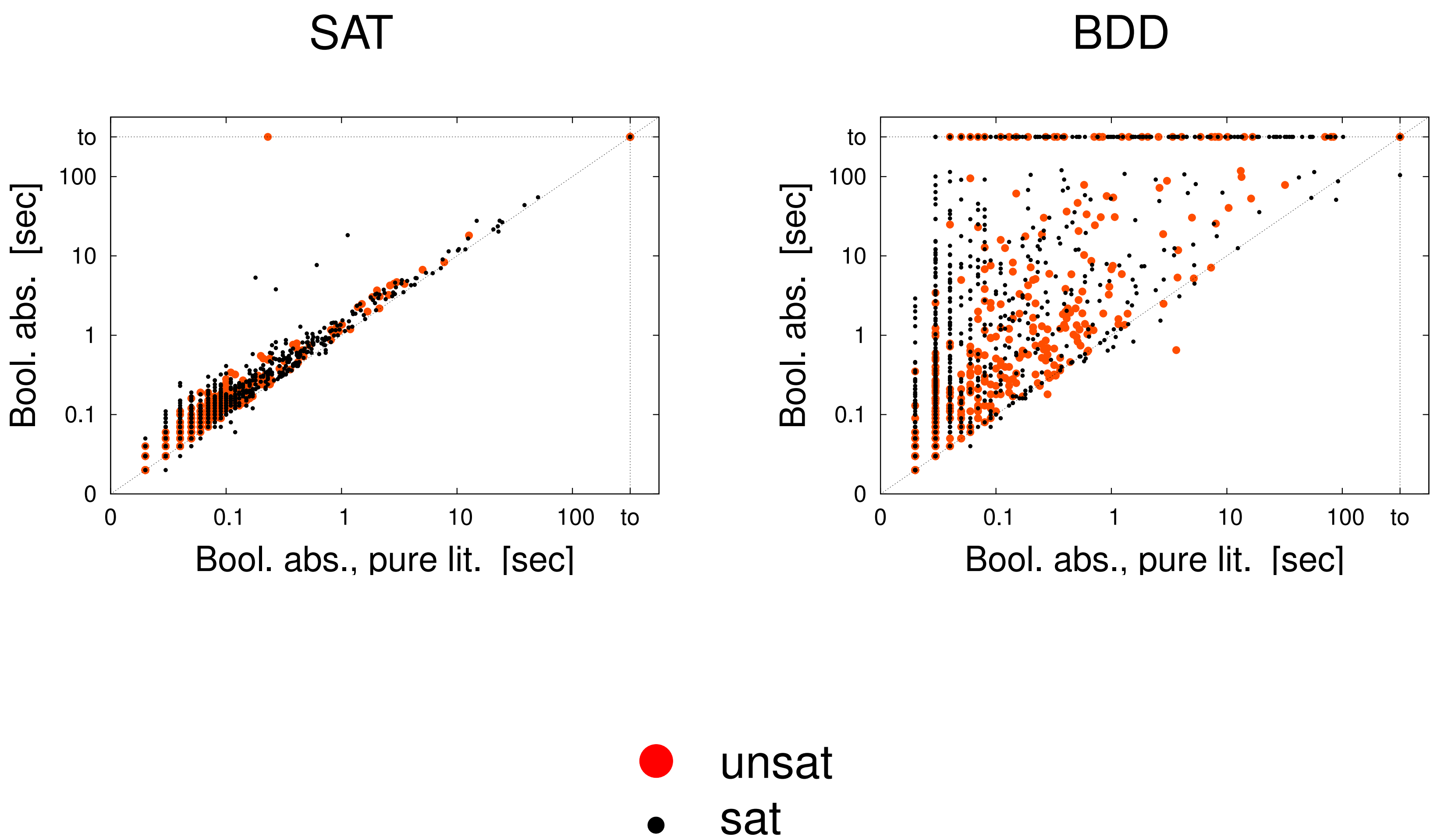


● unsat

• sat

© 2007 V. Schuppan

Pure literal rule vs. not (with Boolean abstraction)

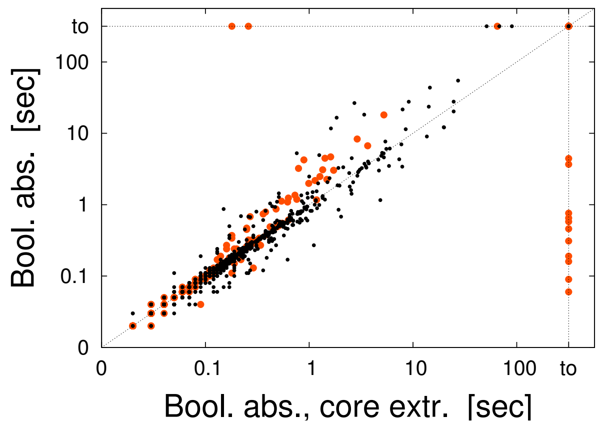


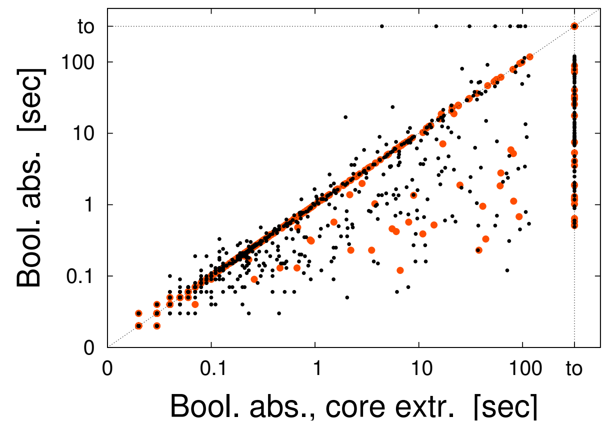
© 2007 V. Schuppan

Unsat core extraction vs. not

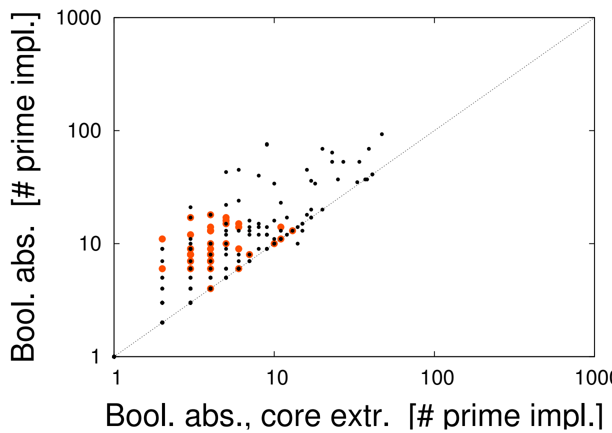
SAT BDD

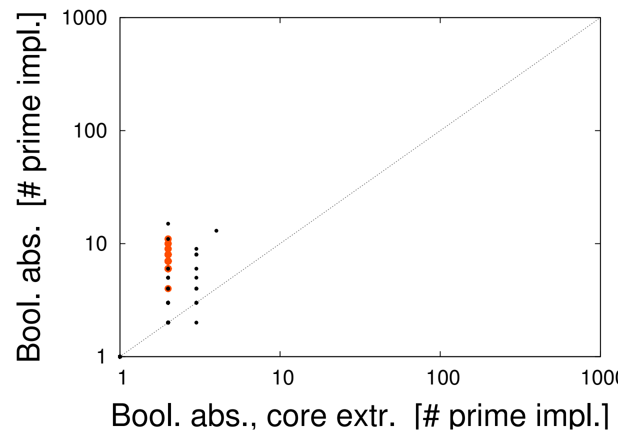
run time





search space





© 2007 V. Schuppan

The End

Introduce Boolean abstraction for PSL.
$\Rightarrow$ Very helpful with BDD-based, unclear with SAT-based solvers.
$\Rightarrow$ SAT- and BDD-based approaches complementary.

Extend pure literal simplification to PSL.
$\Rightarrow$ Very helpful, more so when applied to prime implicants.

Extract unsatisfiable cores from solvers.
$\Rightarrow$ Reduces search space, though at the cost of run time.

Much room for improvement:

- Optimize extraction of unsatisfiable cores.
- Reuse partial results between prime implicants.
- Improve prioritization of prime implicants.
- ... (and some more) ...

Thanks!

© 2007 V. Schuppan

Keep out!

Backup slides

© 2007 V. Schuppan

1. p ($\neg p$) is a positive (negative) occurrence of p.

2. A positive occurrence of p in ϕ, r is a positive (negative) occurrence of p in

$$\mathbf{X}\phi$$

$$\begin{array}{ll}
\phi \vee \psi & \psi \vee \phi \\
\phi \wedge \psi & \psi \wedge \phi \\
\phi \,\mathbf{U}\, \psi & \psi \,\mathbf{U}\, \phi \\
\phi \,\mathbf{R}\, \psi & \psi \,\mathbf{R}\, \phi \\
r \diamond\!\!\longrightarrow \psi & s \diamond\!\!\longrightarrow \phi \\
r \;|\!\!\!\rightarrow \psi & s \;|\!\!\!\rightarrow \phi
\end{array}$$

(and analogously for a negative occurrence of p).

p is pure positive (negative) in ϕ iff all occurrences of p in ϕ are positive (negative).

© 2007 V. Schuppan

Complete Simple Bounded Model Checking

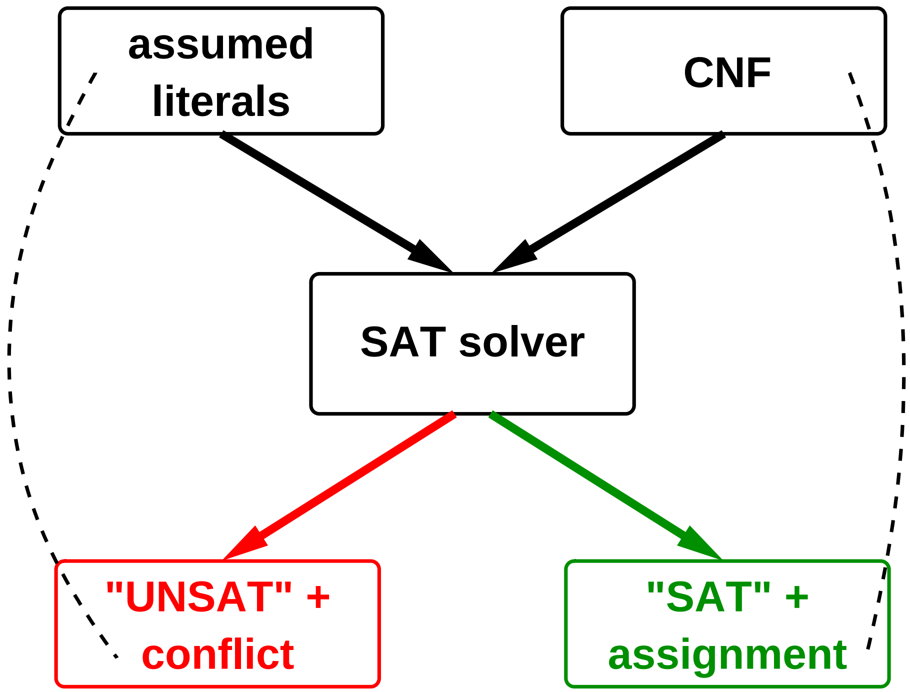
[Heljanko, Junttila, Latvala (CAV’05)]

```
1   k ← 0;
2   while true do
3       check for contradiction at length k;
4       if contradiction then return no fair path exists fi
5       check for non-redundant path of length k;
6       if no non-redundant path then return no fair path exists fi
7       check for fair lasso-shaped path of length k;
8       if fair lasso-shaped path then return fair path exists fi
9       k++;
10  od
```

Note: all constraints added in lines 3, 5 for k are present at lines 3, 5, 7 for $k' > k$.

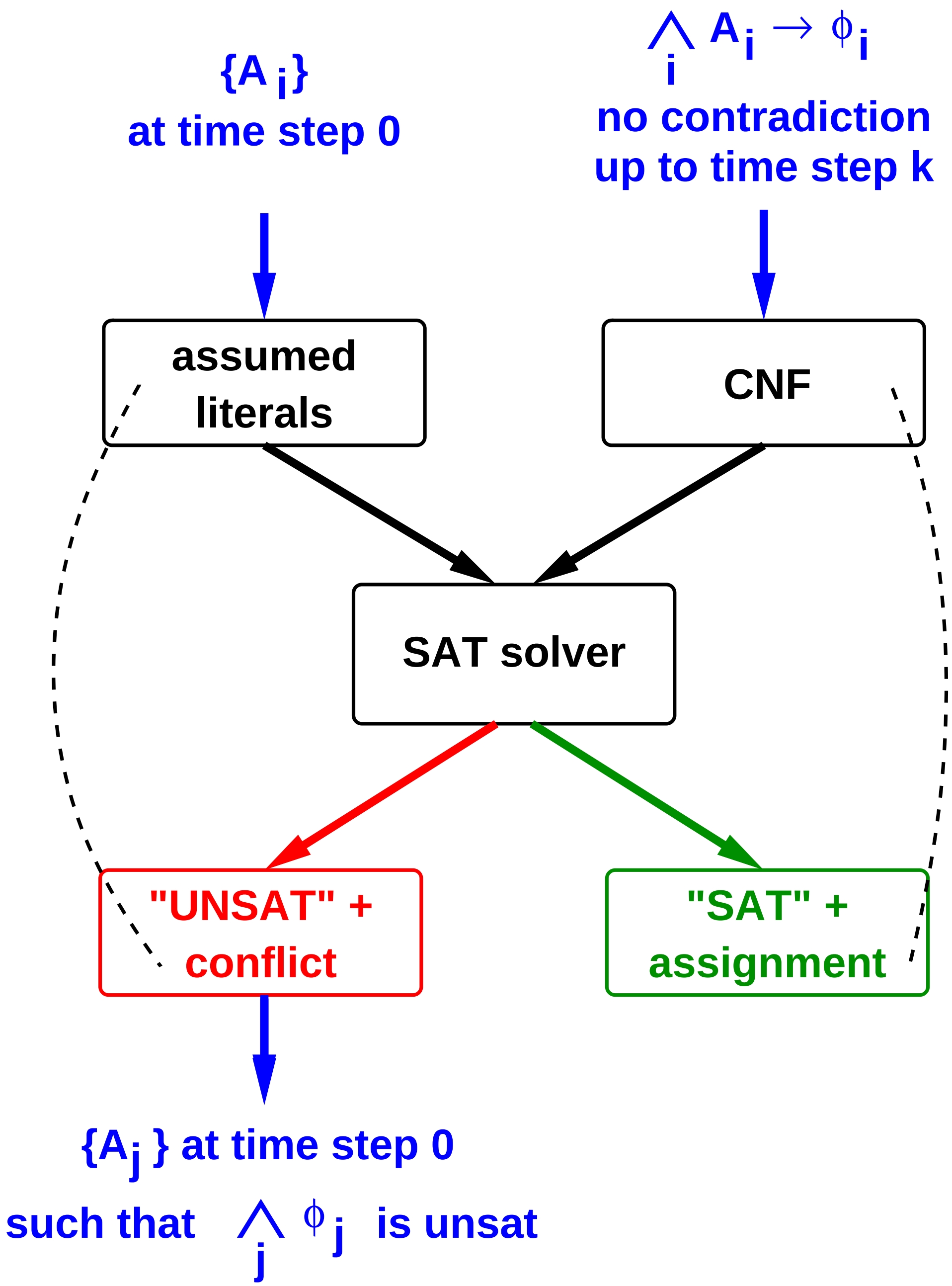
© 2007 V. Schuppan

Extracting Unsatisfiable Cores with SAT-based Solvers



© 2007 V. Schuppan

Extracting Unsatisfiable Cores with SAT-based Solvers

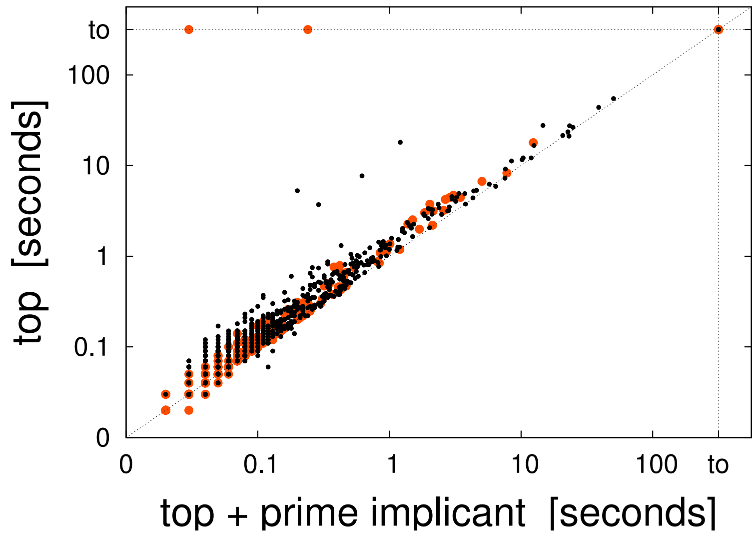


(We obtain one unsatisfiable core.)

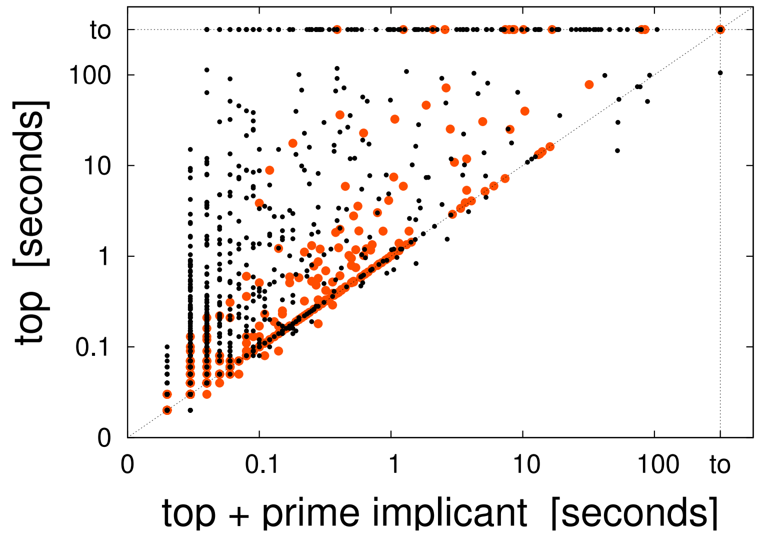
© 2007 V. Schuppan

Pure literal simplification at top + prime implicant levels vs. only at top

SAT



BDD



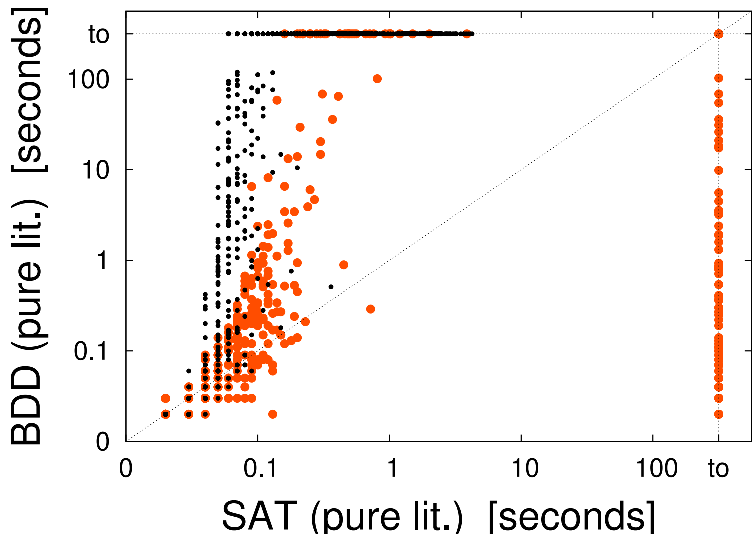
● unsat
• sat

© 2007 V. Schuppan

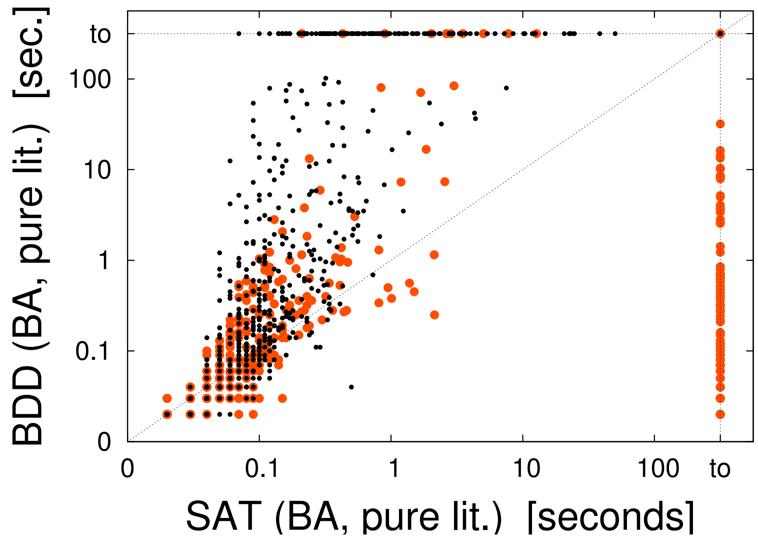
Results

SAT vs. BDD

without Boolean abstraction



with Boolean abstraction



● unsat

• sat

© 2007 V. Schuppan