Simulating Structual Reasoning on the CNF-Level

Symposium on Structure in Hard Combinatorial Problems

TU Vienna, Austria

Armin Biere

Institute for Formal Models and Verification
Johannes Kepler University, Linz, Austria

based on joined work with

Marijn Heule (UT Austin), Matti Järvisalo (Univ. Helsinki)

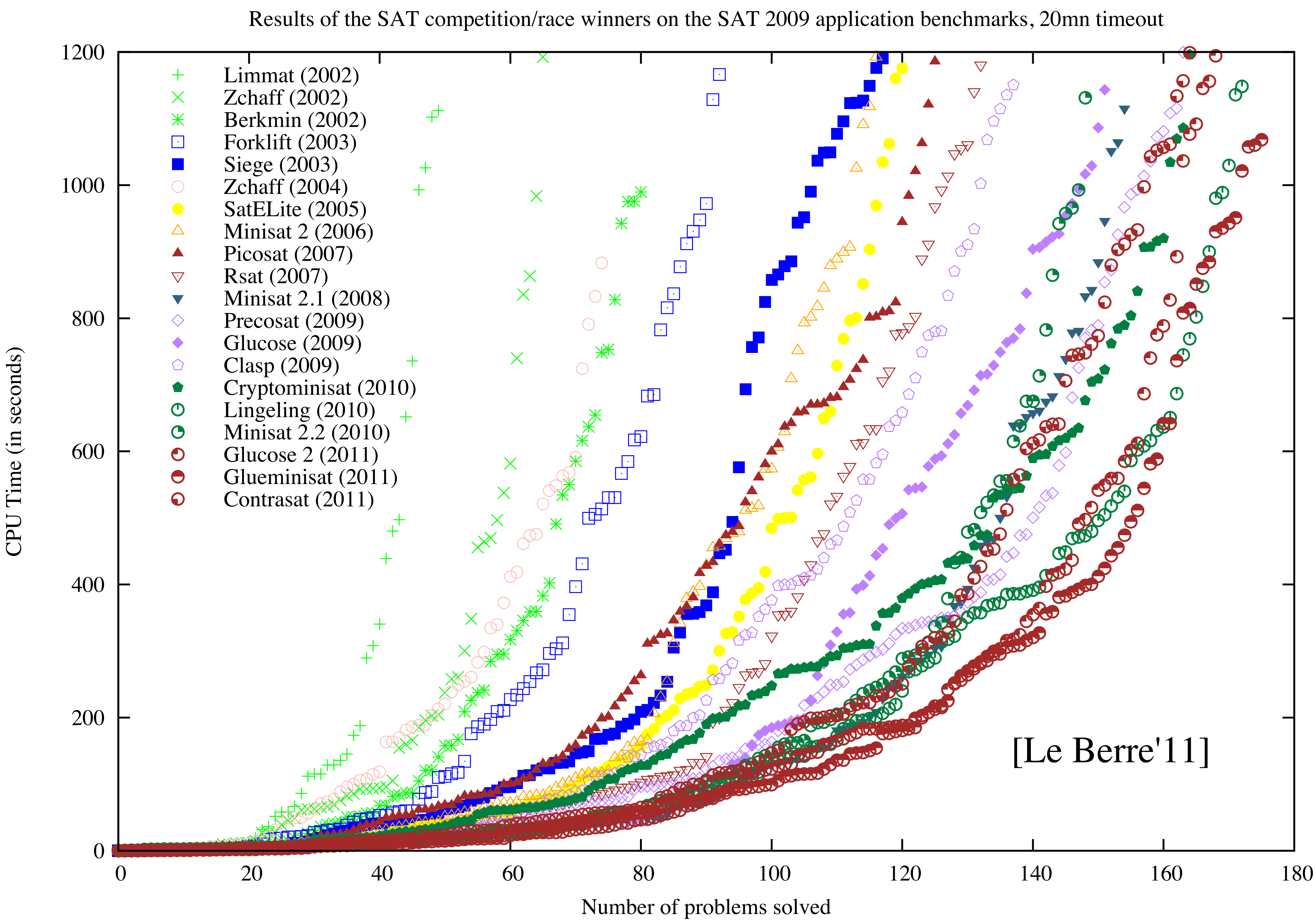
Gergely Kovásznai (JKU), Andreas Fröhlich (JKU), Norbert Manthey (TU Dresden)

May 18, 2013

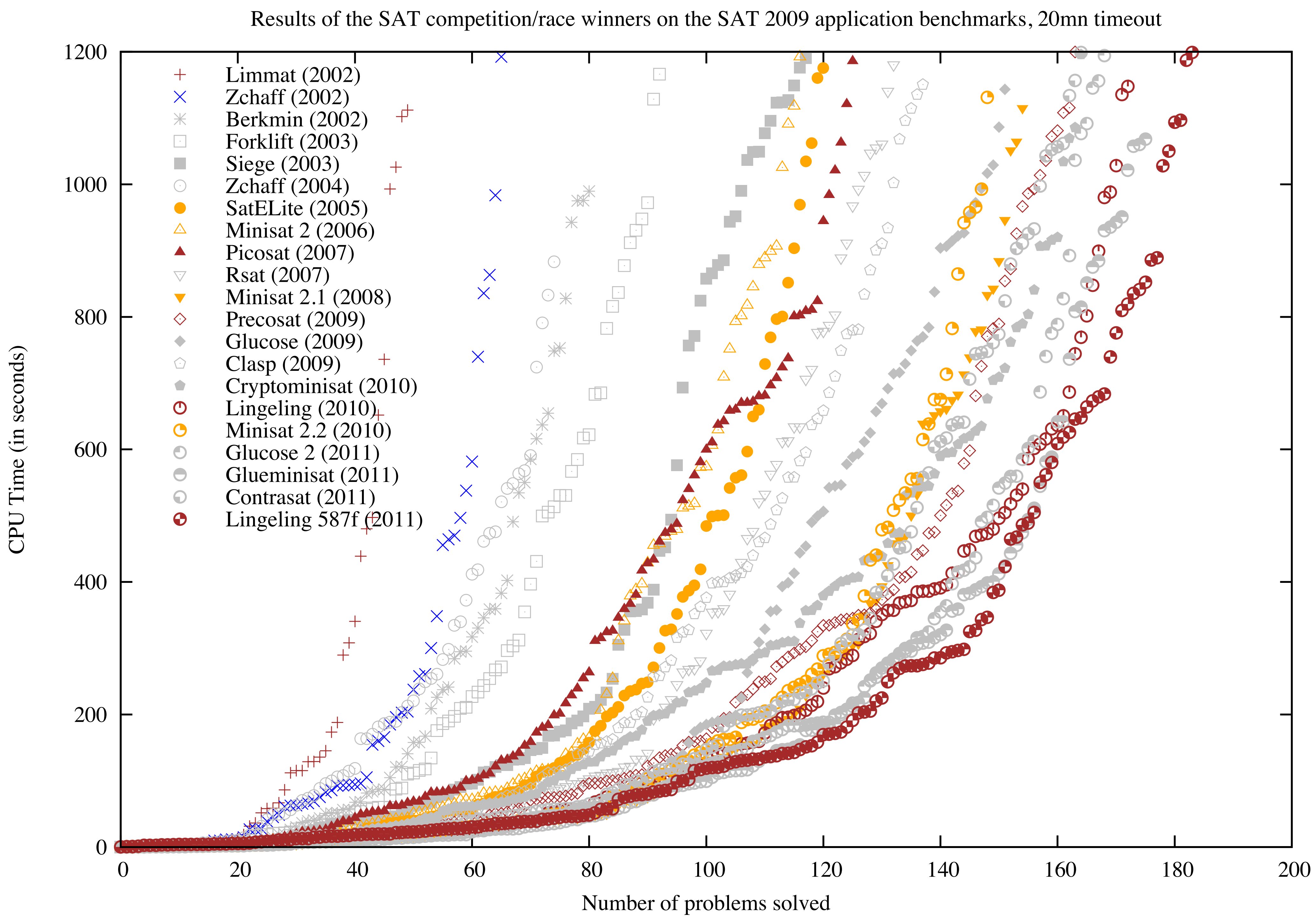
new version June 7, 2013

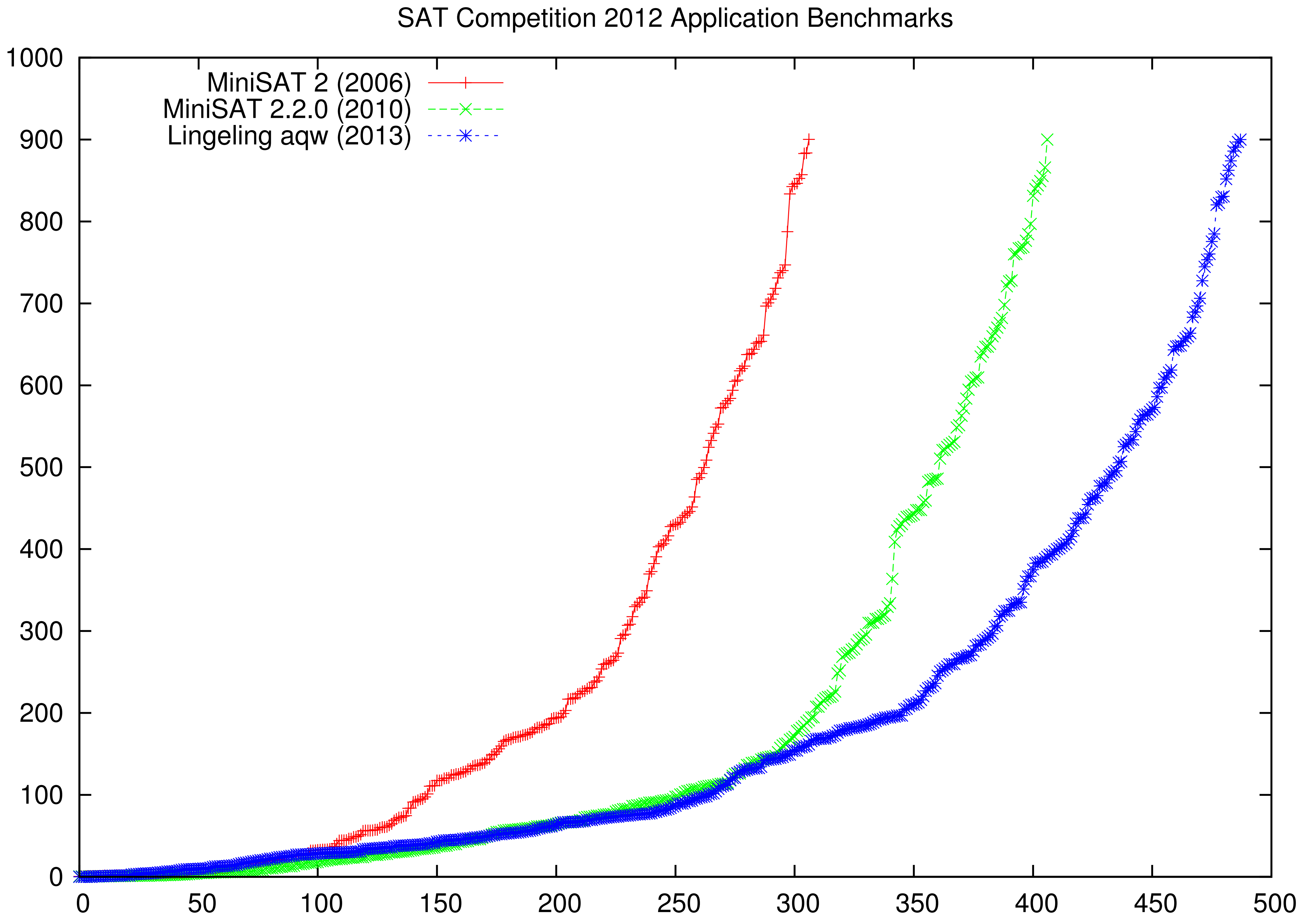
Overview

- What progress has been made since MiniSAT?
- Preprocessing and Inproccessing
- "Unhiding"
- Quantifier-free bit-vector logic (QF_BV)
  - demo: example of Pete Jeavons with QF_BV and solving with Boolector/Lingeling
  - complexity of decision problem for QF_BV, QF_BV$_{bw}$, QF_BV$_{\ll 1}$, QF_BV$_{\ll c}$, $\ldots$

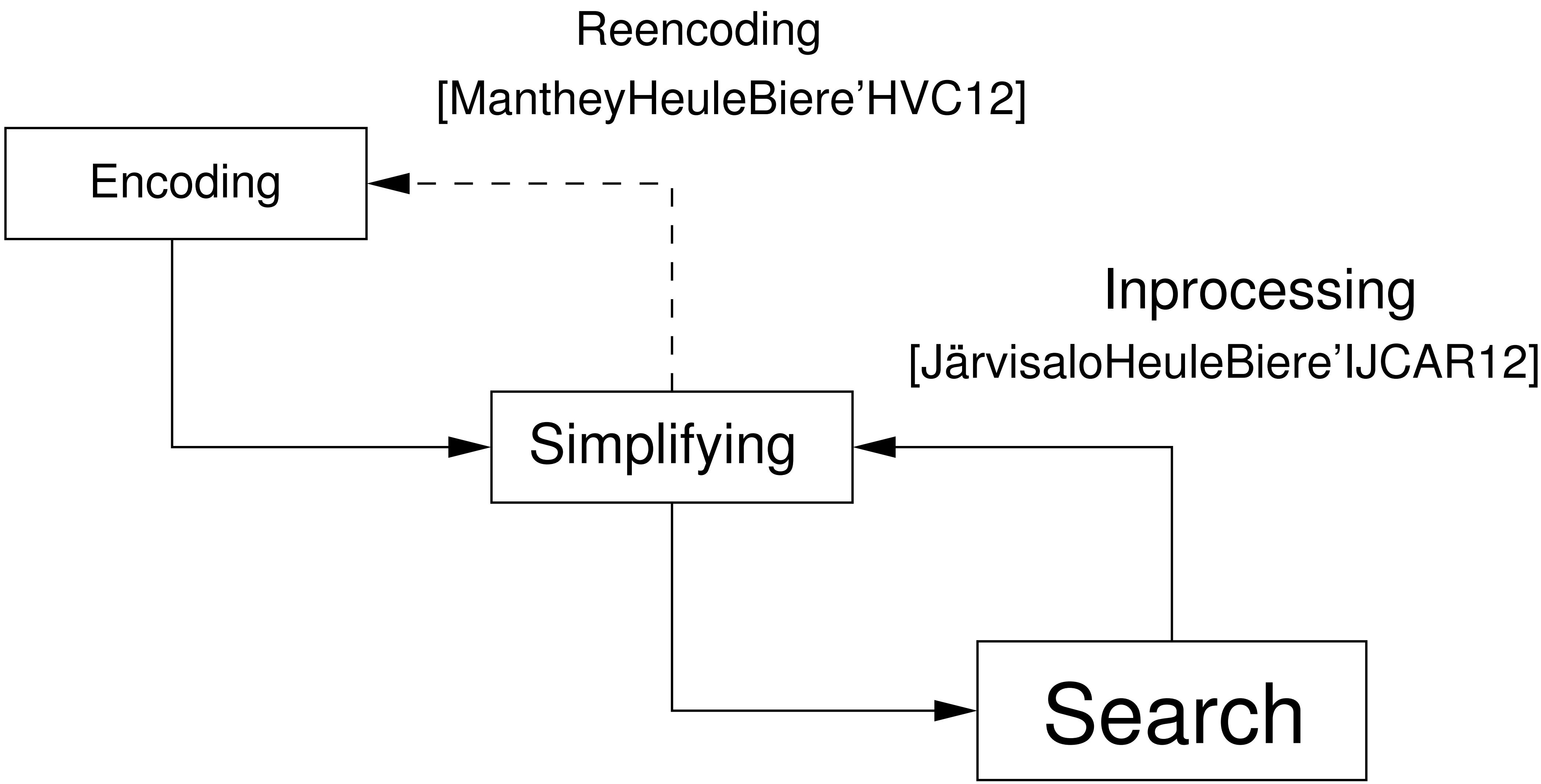


Results of the SAT competition/race winners on the SAT 2009 application benchmarks, 20mn timeout
CPU Time (in seconds)
Number of problems solved
Limmat (2002)
Zchaff (2002)
Berkmin (2002)
Forklift (2003)
Siege (2003)
Zchaff (2004)
SatELite (2005)
Minisat 2 (2006)
Picosat (2007)
Rsat (2007)
Minisat 2.1 (2008)
Precosat (2009)
Glucose (2009)
Clasp (2009)
Cryptominisat (2010)
Lingeling (2010)
Minisat 2.2 (2010)
Glucose 2 (2011)
Glueminisat (2011)
Contrasat (2011)
0
200
400
600
800
1000
1200
0
20
40
60
80
100
120
140
160
180
[Le Berre'11]


Results of the SAT competition/race winners on the SAT 2009 application benchmarks, 20mn timeout
Limmat (2002)
Zchaff (2002)
Berkmin (2002)
Forklift (2003)
Siege (2003)
Zchaff (2004)
SatELite (2005)
Minisat 2 (2006)
Picosat (2007)
Rsat (2007)
Minisat 2.1 (2008)
Precosat (2009)
Glucose (2009)
Clasp (2009)
Cryptominisat (2010)
Lingeling (2010)
Minisat 2.2 (2010)
Glucose 2 (2011)
Glueminisat (2011)
Contrasat (2011)
Lingeling 587f (2011)
CPU Time (in seconds)
Number of problems solved
0
200
400
600
800
1000
1200
0
20
40
60
80
100
120
140
160
180
200



SAT Competition 2012 Application Benchmarks
MiniSAT 2 (2006)
MiniSAT 2.2.0 (2010)
Lingeling aqw (2013)
1000
900
800
700
600
500
400
300
200
100
0
0
50
100
150
200
250
300
350
400
450
500


Reencoding
[MantheyHeuleBiere'HVC12]
Encoding
Inprocessing
[JärvisaloHeuleBiere'IJCAR12]
Simplifying
Search

[MantheyHeuleBiere'HVC12]

Non-working idea existing in the literature (for shrinking learned clauses):

$$(a \vee b \vee c \vee d) \quad (a \vee b \vee c \vee e) \quad (a \vee b \vee c \vee f)$$

by

$$(x \vee d) \quad (x \vee e) \quad (x \vee f) \quad (\bar{x} \vee a \vee b \vee c)$$

However, bounded variable elimination [DavisPutnam60][EénBiere05] eliminates x again.

No gain, so this does not work in practice.

Reverse of Davis & Putnam style variable elimination.

Replace

$$\begin{array}{cc} (a \vee d) & (a \vee e) \\ (b \vee d) & (b \vee e) \\ (c \vee d) & (c \vee e) \end{array}$$

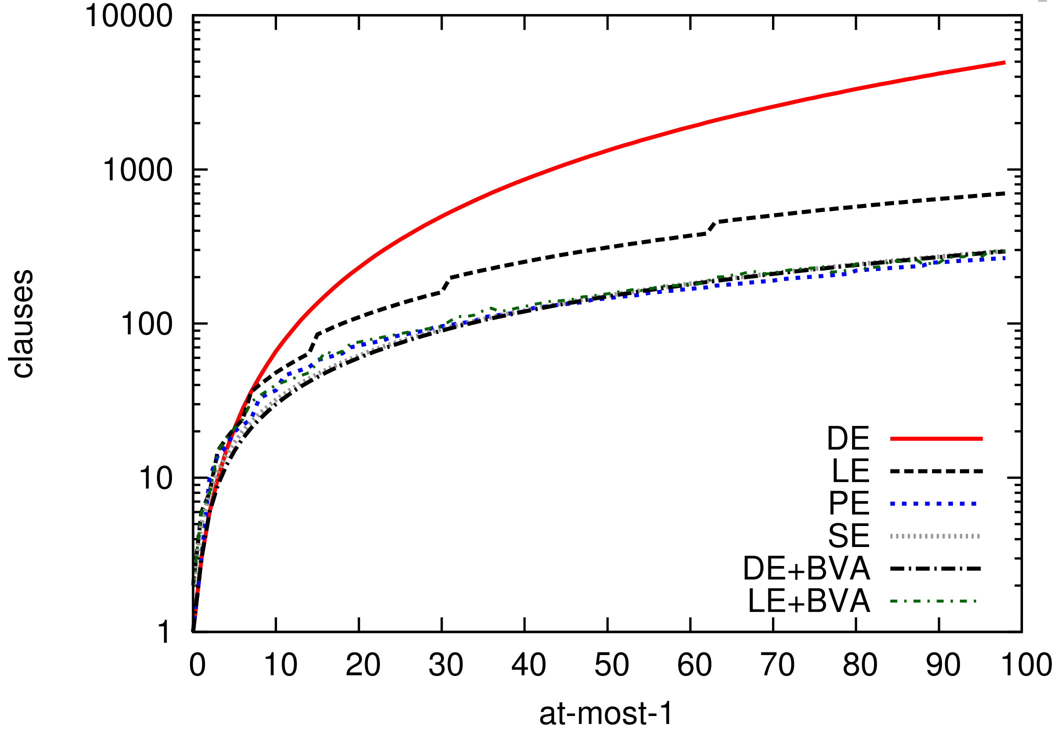
by

$$\begin{array}{ccc} (\bar{x} \vee a) & (\bar{x} \vee b) & (\bar{x} \vee c) \\ (x \vee d) & (x \vee e) & \end{array}$$

Paper gives an algorithm to find this kind of patterns fast.

At-Most-One Constraints and BVA

[MantheyHeuleBiere'HVC12]



Encoding	Clauses	Variables
DE	$\frac{n\cdot(n-1)}{2}$	n
LE	$n \cdot \lceil \log n \rceil$	$n + \log n$
PE	$2n + 4 \cdot \sqrt{n} + O(\sqrt[4]{n})$	$n + \sqrt{n} + O(\sqrt[4]{n})$
SE	$3n - 4$	$2n - 1$
DE+ BVA	$3n - 6$	$\sim 2n$
LE+ BVA	$\sim 3n$	$\sim 1.5n$

DE naïve quadratic encoding
PE [Chen'ModRef11]
LE [Prestwich'SAT07]
SE [Sinz'CP05]

Preprocessing/Inprocessing

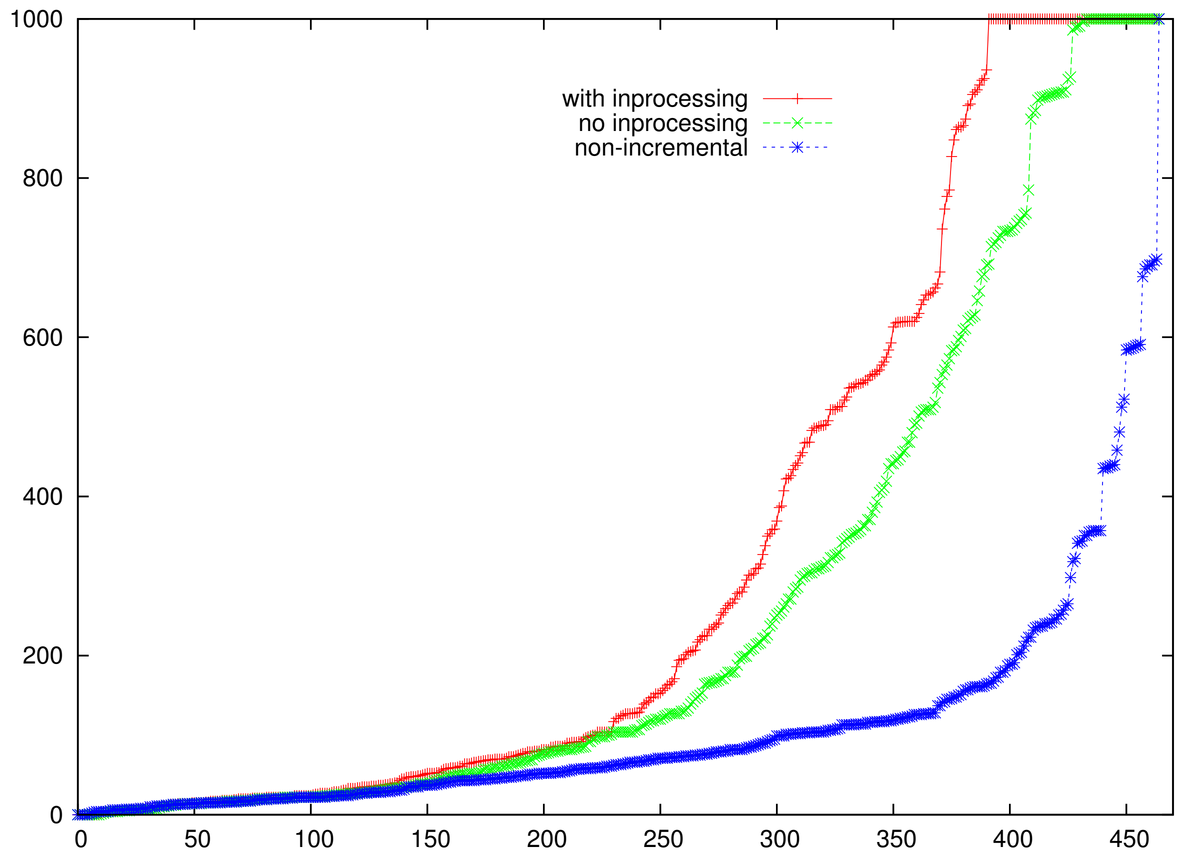
[JärvisaloHeuleBiere'IJCAR12]

Lingeling 587f on SC2009/2011 application instances:

Lingeling	2009 solved	2009 SAT	2009 UNSAT	2009 time	2011 solved	2011 SAT	2011 UNSAT	2011 time
original version 587f	196	79	117	114256	164	78	86	144932
only preprocessing	184	72	112	119161	159	77	82	145218
no pre- nor inprocessing	170	68	102	138940	156	78	78	153434

[JärvisaloHeuleBiere'IJCAR12]

Bounds reached with Blimc Bounded Model Checker:

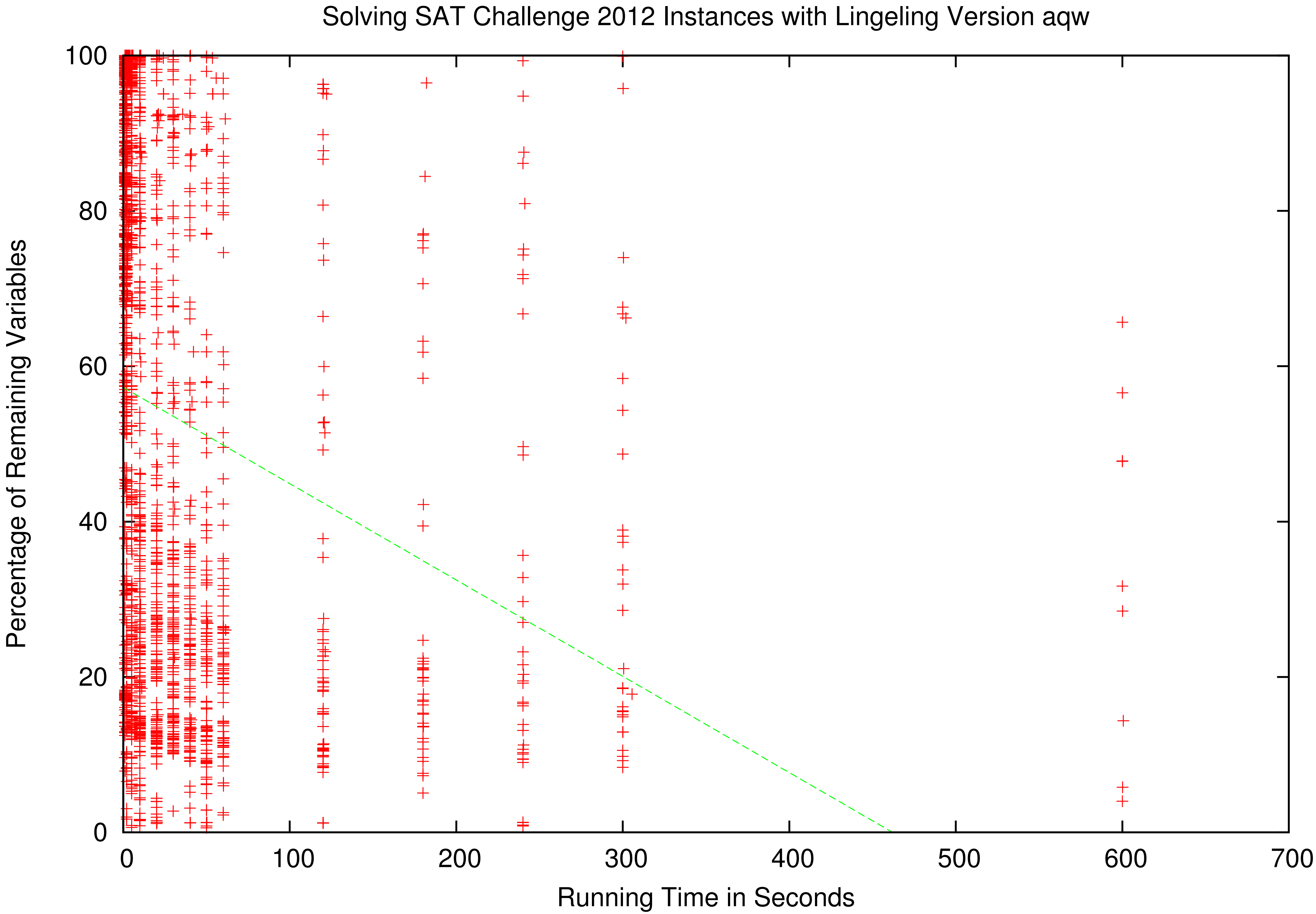


	*	bounds	time
with inpr.	158	153975	304158
no inpr.	115	125436	335393
non-incr.	67	49915	369104

* = #solved + #(bound 1000 reached)

bounds = $\sum$ reached bounds

Hardware Model Checking Competition 2011 Instances



Solving SAT Challenge 2012 Instances with Lingeling Version aqw
Percentage of Remaining Variables
Running Time in Seconds
0
20
40
60
80
100
0
100
200
300
400
500
600
700

Inprocessing Rules

[JärvisaloHeuleBiere'IJCAR12]

φ irredundant clauses ρ redundant clauses σ witness reconstruction stack

$$\frac{\varphi\,[\rho \wedge C]\,\sigma}{\varphi \wedge C\,[\rho]\,\sigma}$$

STRENGTHEN

$$\frac{\varphi\,[\rho \wedge C]\,\sigma}{\varphi\,[\rho]\,\sigma}$$

FORGET

$$\frac{\varphi\,[\rho]\,\sigma}{\varphi\,[\rho \wedge C]\,\sigma}\ \mathcal{L}$$

LEARN

$$\frac{\varphi \wedge C\,[\rho]\,\sigma}{\varphi\,[\rho \wedge C]\,\sigma, l{:}C}\ \mathcal{W}$$

WEAKEN

$\mathcal{L}$ is that $\boxed{\varphi \wedge \rho}$ and $\boxed{\varphi \wedge \rho \wedge C}$ are satisfiability-equivalent.

$\mathcal{W}$ is that $\boxed{\varphi}$ and $\boxed{\varphi \wedge C}$ are satisfiability-equivalent.

RAT – Resolution Asymmetric Tautology

[JärvisaloHeuleBiere'IJCAR12]

Idea: "Resolution Look-Ahead"

Clause C is an *Asymmetric Tautology* (AT) w.r.t. G *iff* $G \wedge \neg C$ refuted by BCP.

Given clause $C \in F$, $l \in C$.

Assume all resolvents R of C on l with clauses in F are AT w.r.t. $F \backslash \{C\}$.

Then C is called *Resolution Asymmetric Tautology* (RAT) w.r.t. F on l.

In this case F is satisfiability equivalent to $F \backslash \{C\}$.

Inprocessing Rules with RAT simulate all techniques in current SAT solvers

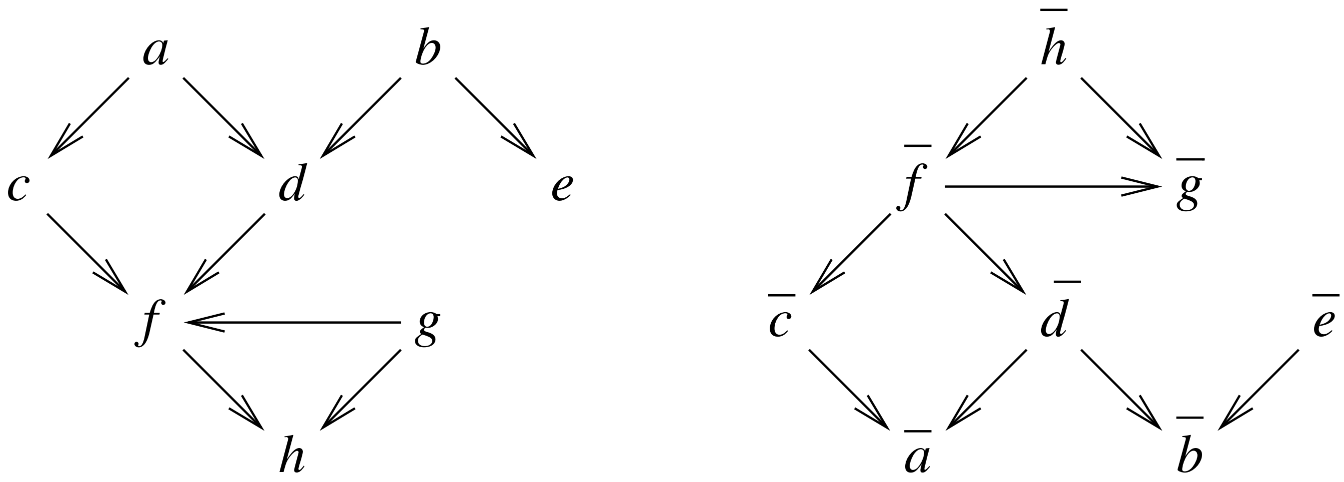
Motivation for Unhiding

[HeuleJärvisaloBiere SAT'11]

- SAT solvers applied to huge formulas
  - million of variables
  - fastests solvers use preprocessing/inprocessing
  - *need cheap and effective inprocessing techniques for millions of variables*
- this talk:
  - **unhiding** redundancy in large formulas
  - almost linear randomized algorithm
  - using the binary implication graph
  - fast enough to be applied to learned clauses
- see our SAT'11 paper for more details

Binary Implication Graph BIG

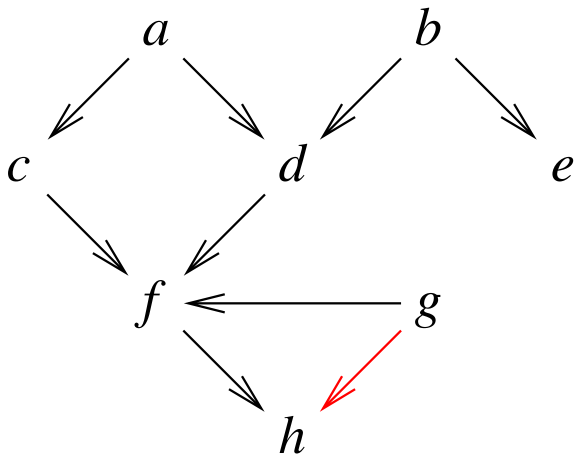
[HeuleJärvisaloBiere SAT'11]

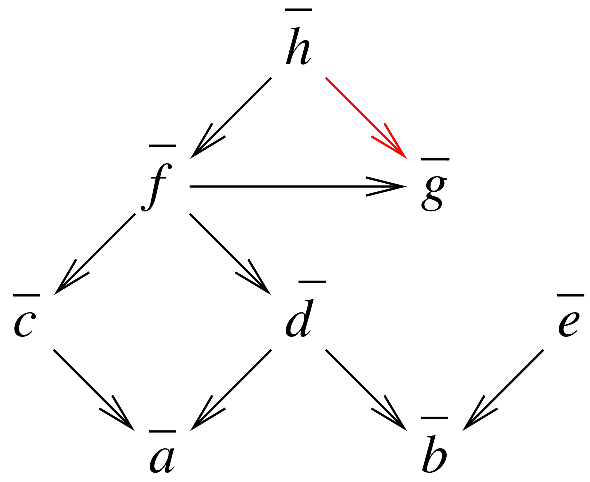


$$
\begin{aligned}
&(\bar{a} \vee c) \wedge (\bar{a} \vee d) \wedge (\bar{b} \vee d) \wedge (\bar{b} \vee e) \wedge \\
&(\bar{c} \vee f) \wedge (\bar{d} \vee f) \wedge (\bar{g} \vee f) \wedge (\bar{f} \vee h) \wedge \\
&(\bar{g} \vee h) \wedge \underbrace{(\bar{a} \vee \bar{e} \vee h) \wedge (\bar{b} \vee \bar{c} \vee h) \wedge (a \vee b \vee c \vee d \vee e \vee f \vee g \vee h)}_{\text{non binary clauses}}
\end{aligned}
$$

Transitive Reduction TRD

[HeuleJärvisaloBiere SAT’11]



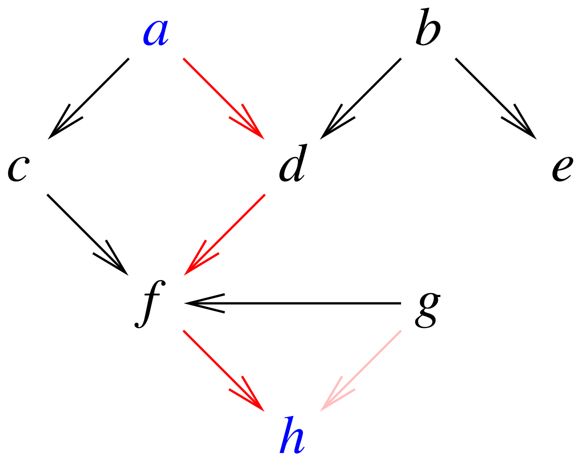


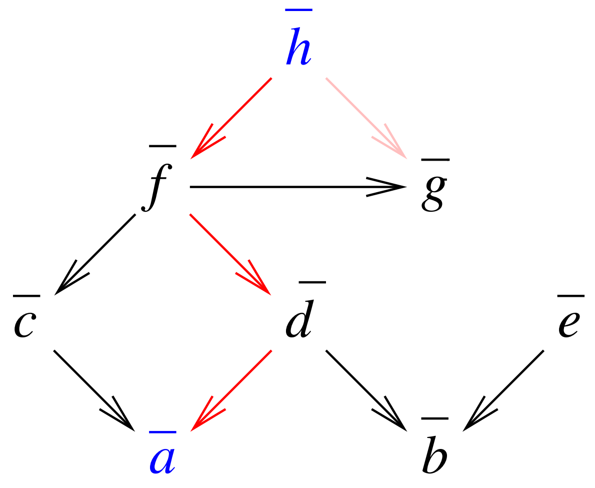
$$(\bar{a} \vee c) \wedge (\bar{a} \vee d) \wedge (\bar{b} \vee d) \wedge (\bar{b} \vee e) \wedge$$
$$(\bar{c} \vee f) \wedge (\bar{d} \vee f) \wedge (\bar{g} \vee f) \wedge (\bar{f} \vee h) \wedge$$
$$\cancel{(\bar{g} \vee h)} \wedge (\bar{a} \vee \bar{e} \vee h) \wedge (\bar{b} \vee \bar{c} \vee h) \wedge (a \vee b \vee c \vee d \vee e \vee f \vee g \vee h)$$

TRD

$g \rightarrow f \rightarrow h$

[HeuleJärvisaloBiere LPAR'2010]





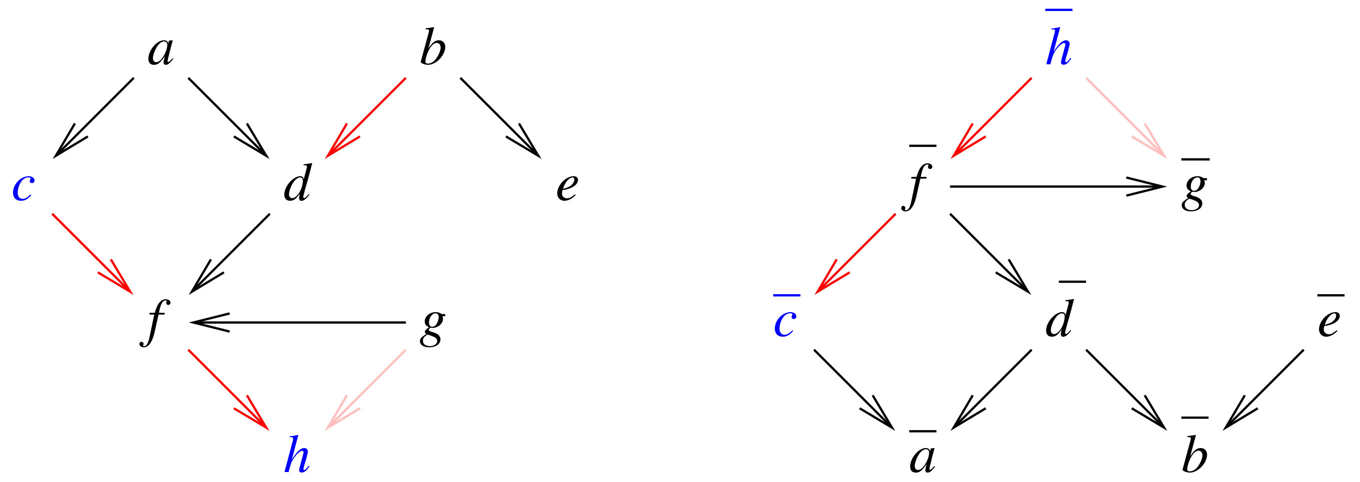
$$(\bar{a} \vee c) \wedge (\bar{a} \vee d) \wedge (\bar{b} \vee d) \wedge (\bar{b} \vee e) \wedge$$
$$(\bar{c} \vee f) \wedge (\bar{d} \vee f) \wedge (\bar{g} \vee f) \wedge (\bar{f} \vee h) \wedge$$
$$\cancel{(\bar{a} \vee \bar{e} \vee h)} \wedge (\bar{b} \vee \bar{c} \vee h) \wedge (a \vee b \vee c \vee d \vee e \vee f \vee g \vee h)$$

HTE

$a \to d \to f \to h$

Hidden Tautology Elimination HTE

[HeuleJärvisaloBiere LPAR'2010]



$$(\bar{a} \lor c) \land (\bar{a} \lor d) \land (\bar{b} \lor d) \land (\bar{b} \lor e) \land$$

$$(\bar{c} \lor f) \land (\bar{d} \lor f) \land (\bar{g} \lor f) \land (\bar{f} \lor h) \land$$

$$\cancel{(\bar{b} \lor \bar{c} \lor h)} \land (a \lor b \lor c \lor d \lor e \lor f \lor g \lor h)$$

HTE

$c \rightarrow f \rightarrow h$

Self-Subsuming Resolution SSR

[EénBiere-SAT'05]

$$\frac{C \vee l \qquad D \vee \bar{l}}{D} \quad C \subseteq D \qquad\qquad \frac{a \vee b \vee l \qquad a \vee b \vee c \vee \bar{l}}{a \vee b \vee c}$$

resolvent D subsumes second antecedent $D \vee \bar{l}$

assume given CNF contains both antecedents $\ldots(a \vee b \vee l)(a \vee b \vee c \vee \bar{l}) \cdots$

if D is added to CNF then $D \vee \bar{l}$ can be removed $\Downarrow$

which in essence *removes* $\bar{l}$ from $D \vee \bar{l}$ $\ldots(a \vee b \vee l)(a \vee b \vee c \quad)\ldots$

used in SATeLite preprocessor [EénBiere'SAT05]

now common in many SAT solvers

HTE = Reverse SSR + Tautology Elimination

[HeuleJärvisaloBiere LPAR'2010]

hidden literal addition (HLA) uses SSR in reverse order

$$\frac{C \vee l \qquad D \vee \bar{l}}{D} \quad C \subseteq D \qquad\qquad \frac{a \vee b \vee l \qquad a \vee b \vee c \vee \bar{l}}{a \vee b \vee c}$$

assume given CNF contains resolvent and first antecedent $\ldots(a \vee b \vee l)(a \vee b \vee c)\cdots$

we can replace D by $D \vee \bar{l}$ $\ldots(a \vee b \vee l)(a \vee b \vee c \vee \bar{l})\ldots$

which in essence *adds* $\bar{l}$ to D, repeat HLA until fix-point

keep remaining non-tautological clauses *after removing added literals again*

HTE = assume $C \vee l$ is a binary clauses more general versions in the paper

remove clauses with a literal implied by negation of another literal in the clause

HTE confluent and BCP preserving modulo equivalent variable renaming

Hidden Literal Elimination HLE

[HeuleJärvisaloBiere SAT'11]

better explained on binary implication graph

remove literal from a clause which implies another literal in the clause

$$\ldots(\bar{a} \vee b)(\bar{b} \vee c)(a \vee c \vee d)\ldots \quad \Rightarrow \quad \ldots(\bar{a} \vee b)(\bar{b} \vee c)(c \vee d)\ldots$$

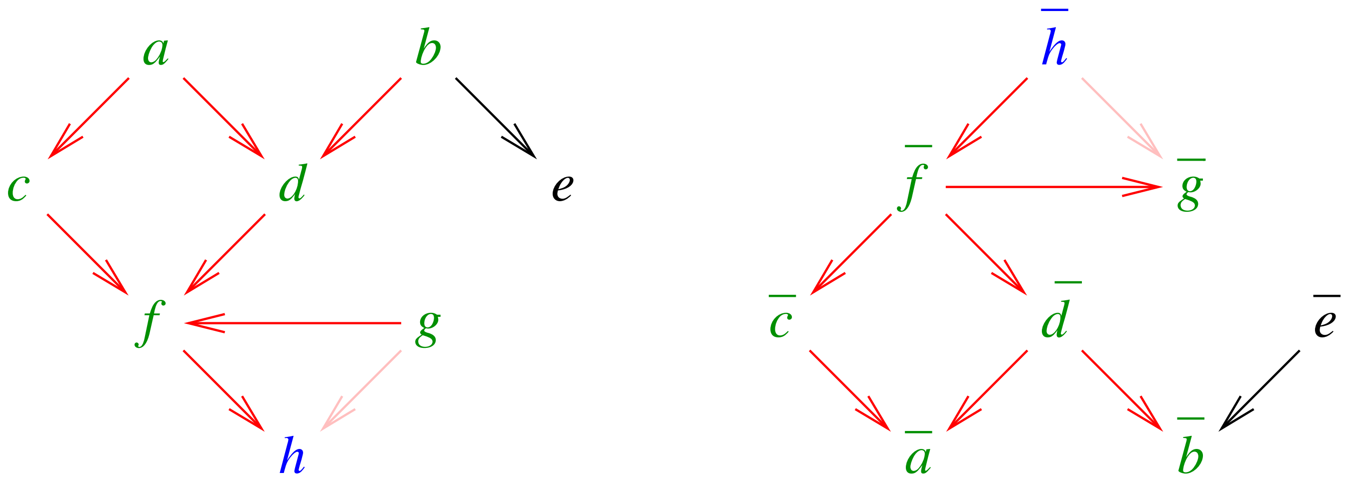
related work before all uses BCP:

- asymmetric branching — implemented in MiniSAT but switched off by default
- **distillation** — [JinSomenzi'05][HanSomenzi DAC'07]
- vivification — [PietteHamadiSais ECAI'08]
- caching technique in CryptoMiniSAT

HTE/HLE only uses the binary implication graph!

Hidden Literal Elimination HLE

[HeuleJärvisaloBiere SAT'11]



$$(\bar{a} \vee c) \wedge (\bar{a} \vee d) \wedge (\bar{b} \vee d) \wedge (\bar{b} \vee e) \wedge$$

$$(\bar{c} \vee f) \wedge (\bar{d} \vee f) \wedge (\bar{g} \vee f) \wedge (\bar{f} \vee h) \wedge$$

$$(\cancel{a \vee b \vee c \vee d \vee} e \vee \cancel{f \vee g \vee} h)$$

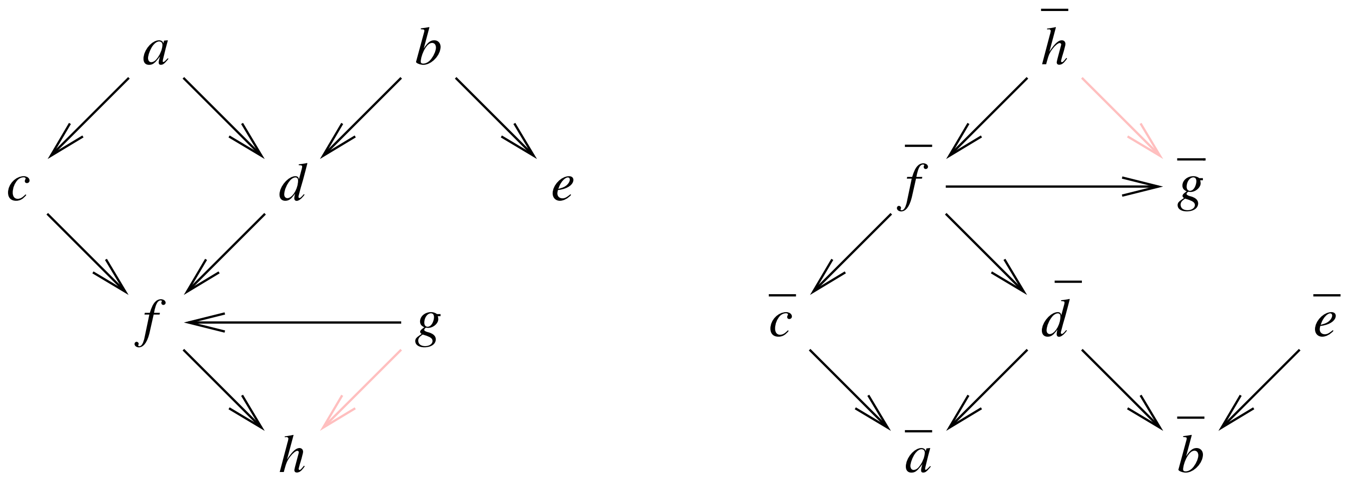
HLE

all but e imply h

also b implies e

Hidden Literal Elimination HLE

[HeuleJärvisaloBiere SAT'11]



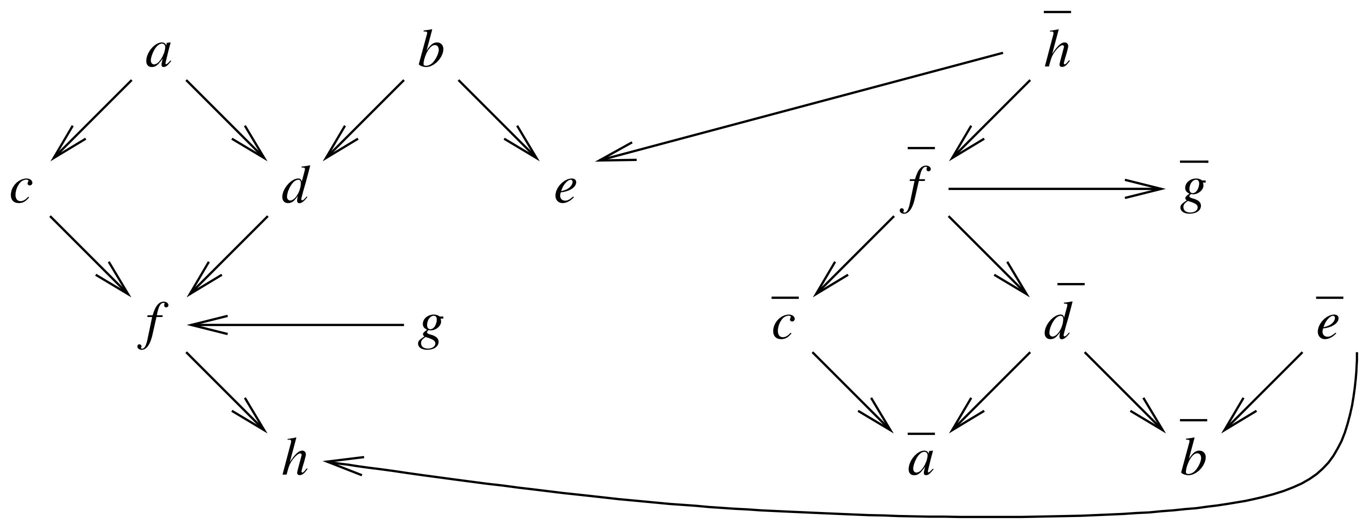
$$(\bar{a} \vee c) \wedge (\bar{a} \vee d) \wedge (\bar{b} \vee d) \wedge (\bar{b} \vee e) \wedge$$

$$(\bar{c} \vee f) \wedge (\bar{d} \vee f) \wedge (\bar{g} \vee f) \wedge (\bar{f} \vee h) \wedge$$

$$(\qquad e \vee \qquad h)$$

TRD + HTE + HLE

[HeuleJärvisaloBiere SAT’11]



$$(\bar{a} \vee c) \wedge (\bar{a} \vee d) \wedge (\bar{b} \vee d) \wedge (\bar{b} \vee e) \wedge$$
$$(\bar{c} \vee f) \wedge (\bar{d} \vee f) \wedge (\bar{g} \vee f) \wedge (\bar{f} \vee h) \wedge$$
$$(e \vee h)$$

Failed Literal Elimination FL

actually quite old technique . . . [Freeman PhdThesis'95] [LeBerre'01] . . .

assume literal l, BCP, if conflict, add unit $\bar{l}$

rather costly to run until completion in practice

one BCP is linear and also in practice can be quite expensive

need to do it for all variables and restart if new binary clause generated

useful in practice: lift common implied literals for assumption l and assumption $\bar{l}$

even on BIG (FL2) conjectured to be quadratic [VanGelder'05]

$\ldots(\bar{a} \vee b)(\bar{b} \vee c)(\bar{c} \vee d)(\bar{d} \vee \bar{a})\ldots \quad \Rightarrow$ add unit clause $\bar{a}$

subsumed by running one HLA until completion

Tree-based Look-Ahead saves some time [HeuleJärvisaloBiere'CPAIOR13]

Equivalent Literal Substitution ELS

[AspvallPlassTarjan'79] [Li'00] [Val'01] [Brafman'04] [VanGelder'05]

decompose BIG into strongly connect components (SCCs)

if there is an l with l and $\bar{l}$ in the same component $\Rightarrow$ *unsatisfiable*

otherwise replace all literals by a "representative"

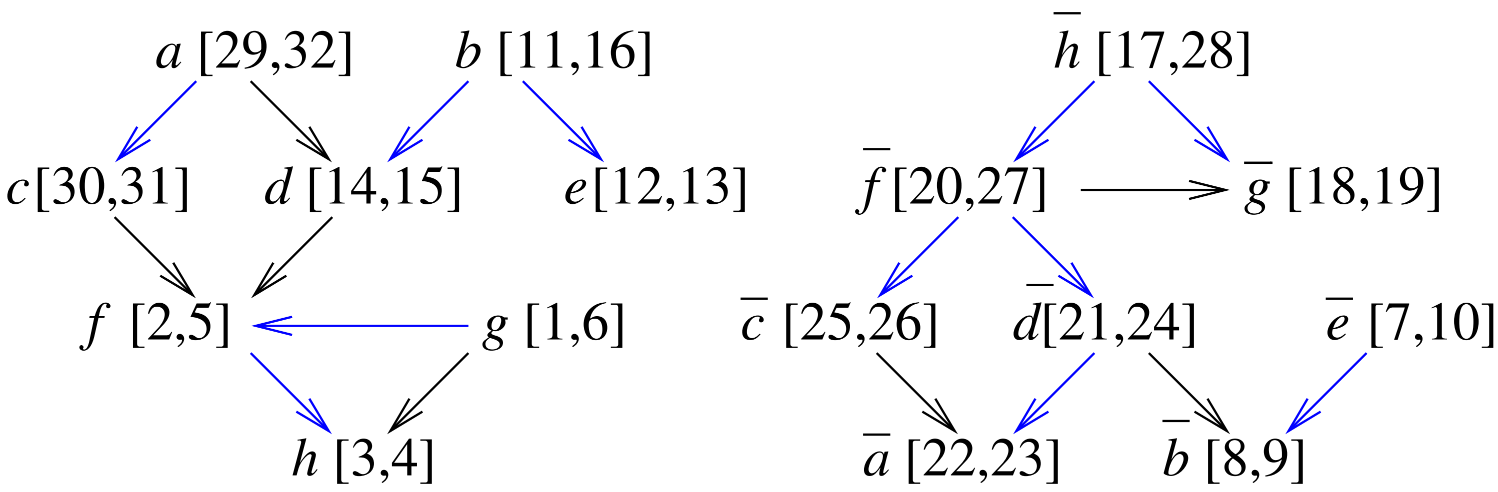
linear algorithm can be applied routinely during garbage collection

but as with failed literal preprocessing may generate new binary clauses

$$\ldots(\bar{a} \vee b)(\bar{b} \vee c)(\bar{c} \vee a)(a \vee b \vee c \vee d)\ldots \quad \Rightarrow \quad \ldots(a \vee d)\ldots$$

[HeuleJärvisaloBiere SAT'11]

DFS tree with discovered and finished times: $[\mathrm{dsc}(l), \mathrm{fin}(l)]$



tree edges

parenthesis theorem: l ancestor in DFS tree of k iff $[\mathrm{dsc}(k), \mathrm{fin}(k)] \subseteq [\mathrm{dsc}(l), \mathrm{fin}(l)]$

well known

ancestor relationship gives necessary conditions for (transitive) implication:

if $[\mathrm{dsc}(k), \mathrm{fin}(k)] \subseteq [\mathrm{dsc}(l), \mathrm{fin}(l)]$ then $l \rightarrow k$

if $[\mathrm{dsc}(\bar{l}), \mathrm{fin}(\bar{l})] \subseteq [\mathrm{dsc}(\bar{k}), \mathrm{fin}(\bar{k})]$ then $l \rightarrow k$

[HeuleJärvisaloBiere SAT'11]

- time stamping in previous example does not cover $b \to h$
  $[11,16] = [\mathrm{dsc}(b), \mathrm{fin}(b)] \not\subseteq [\mathrm{dsc}(h), \mathrm{fin}(h)] = [3,4]$
  $[17,28] = [\mathrm{dsc}(\bar{h}), \mathrm{fin}(\bar{h})] \not\subseteq [\mathrm{dsc}(\bar{b}), \mathrm{fin}(\bar{b})] = [8,9]$

- in example still both HTE "unhidden", HLE works too (since $b \to e$)

- "coverage" heavily depends on DFS order

- as solution we propose multiple **randomized DFS** rounds/phases

- so we approximate a quadratic problem (reachability) randomly by a linear algorithm

- if BIG is a tree *one* time stamping covers everything

Unhiding through Time Stamping

[HeuleJärvisaloBiere SAT'11]

Unhiding (formula F)

  $stamp := 0$
  foreach literal l in $\mathrm{BIG}(F)$ **do**
    $\mathrm{dsc}(l) := 0$; $\mathrm{fin}(l) := 0$
    $\mathrm{prt}(l) := l$; $\mathrm{root}(l) := l$
  foreach $r \in \mathrm{RTS}(F)$ **do**
    $stamp :=$ *Stamp*$(r, stamp)$
  foreach literal l in $\mathrm{BIG}(F)$ **do**
    if $\mathrm{dsc}(l) = 0$ **then**
      $stamp :=$ *Stamp*$(l, stamp)$
  return *Simplify*(F)

Stamp (literal l, integer $stamp$)

  $stamp := stamp + 1$
  $\mathrm{dsc}(l) := stamp$
  foreach $(\bar{l} \vee l') \in F_2$ **do**
    if $\mathrm{dsc}(l') = 0$ **then**
      $\mathrm{prt}(l') := l$
      $\mathrm{root}(l') := \mathrm{root}(l)$
      $stamp :=$ *Stamp*$(l', stamp)$
  $stamp := stamp + 1$
  $\mathrm{fin}(l) := stamp$
  return $stamp$

Simplify (formula F)

  foreach $C \in F$
    $F := F \setminus \{C\}$
    if *UHTE*(C) **then continue**
    $F := F \cup \{$*UHLE*$(C)\}$
  return F

Unhiding HTE

[HeuleJärvisaloBiere SAT'11]

UHTE (clause C)

$l_{\text{pos}} :=$ first element in $S^+(C)$
$l_{\text{neg}} :=$ first element in $S^-(C)$
while *true*
  if $\text{dsc}(l_{\text{neg}}) > \text{dsc}(l_{\text{pos}})$ **then**
    if l_{pos} is last element in $S^+(C)$ **then return** false
    $l_{\text{pos}} :=$ next element in $S^+(C)$
  else if $\text{fin}(l_{\text{neg}}) < \text{fin}(l_{\text{pos}})$ **or** $(|C| = 2$ **and** $(l_{\text{pos}} = \bar{l}_{\text{neg}}$ **or** $\text{prt}(l_{\text{pos}}) = l_{\text{neg}}))$ **then**
    if l_{neg} is last element in $S^-(C)$ **then return** false
    $l_{\text{neg}} :=$ next element in $S^-(C)$
  else return true

$S^+(C)$	sequence of literals in C ordered by $\text{dsc}()$
$S^-(C)$	sequence of negations of literals in C ordered by $\text{dsc}()$

$$O(|C| \log |C|)$$

Unhiding HLE

[HeuleJärvisaloBiere SAT'11]

UHLE (clause C)

$\textit{finished} :=$ finish time of first element in $S^{+}_{\mathrm{rev}}(C)$
foreach $l \in S^{+}_{\mathrm{rev}}(C)$ starting at second element
  if $\mathrm{fin}(l) > \textit{finished}$ **then** $C := C \setminus \{l\}$
  else $\textit{finished} := \mathrm{fin}(l)$
$\textit{finished} :=$ finish time of first element in $S^{-}(C)$
foreach $\bar{l} \in S^{-}(C)$ starting at second element
  if $\mathrm{fin}(\bar{l}) < \textit{finished}$ **then** $C := C \setminus \{l\}$
  else $\textit{finished} := \mathrm{fin}(\bar{l})$
return C

$S^{+}_{\mathrm{rev}}(C)$ reverse of $S^{+}(C)$

$$O(|C| \log |C|)$$

Advanced Time Stamping

[HeuleJärvisaloBiere SAT'11]

Stamp (literal l, integer $stamp$)

BSC $stamp := stamp + 1$

BSC $\mathrm{dsc}(l) := stamp$; $\mathrm{obs}(l) := stamp$

ELS $flag :=$ true   // l represents a SCC

ELS S.push(l)   // push l on SCC stack

BSC   **for each** $(\bar{l} \vee l') \in F_2$

TRD   **if** $\mathrm{dsc}(l) < \mathrm{obs}(l')$ **then** $F := F \setminus \{(\bar{l} \vee l')\}$; **continue**

FLE   **if** $\mathrm{dsc}(\mathrm{root}(l)) \leq \mathrm{obs}(\bar{l}')$ **then**

FLE $l_{\text{failed}} := l$

FLE   **while** $\mathrm{dsc}(l_{\text{failed}}) > \mathrm{obs}(\bar{l}')$ **do** $l_{\text{failed}} := \mathrm{prt}(l_{\text{failed}})$

FLE $F := F \cup \{(\bar{l}_{\text{failed}})\}$

FLE   **if** $\mathrm{dsc}(\bar{l}') \neq 0$ **and** $\mathrm{fin}(\bar{l}') = 0$ **then continue**

BSC   **if** $\mathrm{dsc}(l') = 0$ **then**

BSC $\mathrm{prt}(l') := l$

BSC $\mathrm{root}(l') := \mathrm{root}(l)$

BSC $stamp :=$ *Stamp*$(l', stamp)$

ELS   **if** $\mathrm{fin}(l') = 0$ **and** $\mathrm{dsc}(l') < \mathrm{dsc}(l)$ **then**

ELS $\mathrm{dsc}(l) := \mathrm{dsc}(l')$; $flag :=$ false   // l is equivalent to l'

OBS $\mathrm{obs}(l') := stamp$ // set last observed time attribute

ELS   **if** $flag =$ true **then**   // if l represents a SCC

BSC $stamp := stamp + 1$

ELS   **do**

ELS $l' := S$.pop()   // get equivalent literal

ELS $\mathrm{dsc}(l') := \mathrm{dsc}(l)$ // assign equal discovered time

BSC $\mathrm{fin}(l') := stamp$ // assign equal finished time

ELS   **while** $l' \neq l$

BSC   **return** $stamp$

[HeuleJärvisaloBiere SAT'11]

- implemented as one inprocessing phase in our SAT solver Lingeling
  beside variable elimination, distillation, blocked clause elimination, probing, ...

- bursts of randomized DFS rounds and sweeping over the whole formula

- fast enough to be applicable to large learned clauses as well
  unhiding is particullary effective for learned clauses

- beside UHTE and UHLE we also have added hyper binary resolution UHBR
  not useful in practice

Lingeling 571 on SAT'09 Competition Application Benchmarks

[HeuleJärvisaloBiere SAT'11]

configuration	sol	sat	uns	unhd	simp	elim
adv.stamp (no uhbr)	188	78	110	7.1%	33.0%	16.1%
adv.stamp (w/uhbr)	184	75	109	7.6%	32.8%	15.8%
basic stamp (no uhbr)	183	73	110	6.8%	32.3%	15.8%
basic stamp (w/uhbr)	183	73	110	7.4%	32.8%	15.8%
no unhiding	180	74	106	0.0%	28.6%	17.6%

configuration	hte	stamp	redundant	hle	redundant	units	stamp
adv.stamp (no uhbr)	22	64%	59%	291	77.6%	935	57%
adv.stamp (w/uhbr)	26	67%	70%	278	77.9%	941	58%
basic stamp (no uhbr)	6	0%	52%	296	78.0%	273	0%
basic stamp (w/uhbr)	7	0%	66%	288	76.7%	308	0%
no unhiding	0	0%	0%	0	0.0%	0	0%

similar results for crafted and SAT'10 Race instances

Demo: Solving Pete Jeavons CSP Example with Boolector

[HeuleJärvisaloBiere SAT'11]

Boolector:

`http://fmv.jku.at/boolector`

Generator for example in QF_BV logic of SMTLIB 2 format:

`http://fmv.jku.at/biere/talks/pjex.c`

[KovásznaiFröhlichBiere'SMT12]

Theorem Deciding QF_BV is NEXPTIME complete.

Proof (A) QF_BV $\in$ NEXPTIME: exponential (!) bit-blasting, then call SAT solver.

```
(set-logic QF_BV)
(declare-fun x () (_ BitVec 1000000))
(declare-fun y () (_ BitVec 1000000))
(assert (distinct (bvadd x y) (bvadd y x)))
```

(B) hardness: reduce DQBF to QF_BV

DQBF is NEXPTIME complete [AzharPetersonReif'01]

DQBF is QBF with explicit dependencies: $\forall x, y, z\, \exists a(x, y), b(y, z) \ldots.$

Idea: encode variables as bit-vectors of size 2^m, with m number universals (polynomial!)

encode independence from universal variables (using cofactors + binary magic numbers)

Complexity of Sub-classes of Bit-Vector Logics

[FröhlichKovásznaiBiere'CSR13]

QF_BV$_{bw}$ bit-wise operation, equality, and inequality

QF_BV$_{\ll 1}$ additionally left-shift by one

QF_BV$_{\ll c}$ additionally left-shift by arbitrary constant

Theorem Deciding QF_BV$_{bw}$ is NP-complete

Theorem Deciding QF_BV$_{\ll 1}$ is PSPACE-complete

Theorem Deciding QF_BV$_{\ll c}$ is NEXPTIME-complete

Proofs similar to or based on ideas/results by [Johannsen02][SpielmannKuncak'IJCAR12].

addition, comparison, etc. can be expressed polynomially in QF_BV$_{\ll 1}$