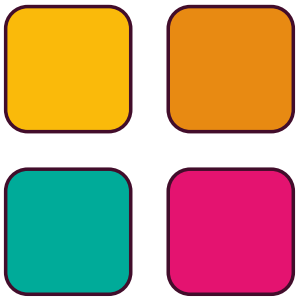
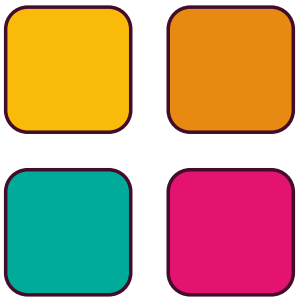


Incremental SAT and applications

Texte

Gilles Audemard

Montpellier - March 2014



Introduction and Motivations

SAT solving in practice

- SAT is a success story in computer science
- Many practical applications

Source J. Marques-Silva

Erdos discrepancy conjecture (1932)

- Infinite sequence of +1 and -1 : $< -1, 1, 1, -1 - 1, x_6, x_7, \ldots >$
- $\forall C \exists k, d$ *such that* $|\sum_{i=1}^{k} x_{i \times d}| \geq C$

- A new application
- The conjecture was proven for $C = 2$ using a SAT solver with an UNSAT certificate (Glucose...)
- UNSAT proof... 13 Gb

- `cgi.csc.liv.ac.uk/~konev/SAT14/`
- `www.newscientist.com/article/dn25068-wikipediasize-maths-proof-too-big-for-humans-to-check.html`

Seminal papers

- "*Planning as Satisfiability*". Kautz and Selman. ECAI 92
- "*Symbolic Model Checking Without BDD*". Biere, Cimatti, Clarke and Zhu. TACAS 1999.

- "*GRASP - a new search algorithm for satisfiability*".Sakallah and Marques-Silva. ICCAD 1996
- "*Chaff : Engineering an Efficient SAT Solver*". Moskewicz, Madigan, Zhao, Zhang and Malik. DAC 2001.
- "*An extensible SAT solver*". Een and Sorensson. SAT 2003
- Minisat source code

CDCL architecture

CDCL architecture

- Preprocessing
  - Encoding does not take into account connexions between some clauses and some variables
  - Simplify the initial formula
  - Very important

CDCL architecture

- Decision variable heuristic
  - Dynamic : Favor variables that are used recently in conflict analysis
  - Phase saving (do not forget the past)

CDCL architecture

- Restarts : dynamic or not

CDCL architecture

- Learnt clauses database cleaning
  - Avoid memory explosion and more

A short overview of CDCL solvers

Sequence of decision, propagations

$$
\begin{aligned}
c_1 &= x_1 \vee x_4 \\
c_2 &= x_1 \vee \overline{x_3} \vee \overline{x_8} \\
c_3 &= x_1 \vee x_8 \vee x_{12} \\
c_4 &= x_2 \vee x_{11} \\
c_5 &= \overline{x_3} \vee \overline{x_7} \vee x_{13} \\
c_6 &= \overline{x_3} \vee \overline{x_7} \vee \overline{x_{13}} \vee x_9 \\
c_7 &= x_8 \vee \overline{x_7} \vee \overline{x_9}
\end{aligned}
$$

A short overview of CDCL solvers

Sequence of decision, propagations

DL 1

$$
\begin{aligned}
c_1 &= x_1 \vee x_4 \\
c_2 &= x_1 \vee \overline{x_3} \vee \overline{x_8} \\
c_3 &= x_1 \vee x_8 \vee x_{12} \\
c_4 &= x_2 \vee x_{11} \\
c_5 &= \overline{x_3} \vee \overline{x_7} \vee x_{13} \\
c_6 &= \overline{x_3} \vee \overline{x_7} \vee \overline{x_{13}} \vee x_9 \\
c_7 &= x_8 \vee \overline{x_7} \vee \overline{x_9}
\end{aligned}
$$

A short overview of CDCL solvers

Sequence of decision, propagations

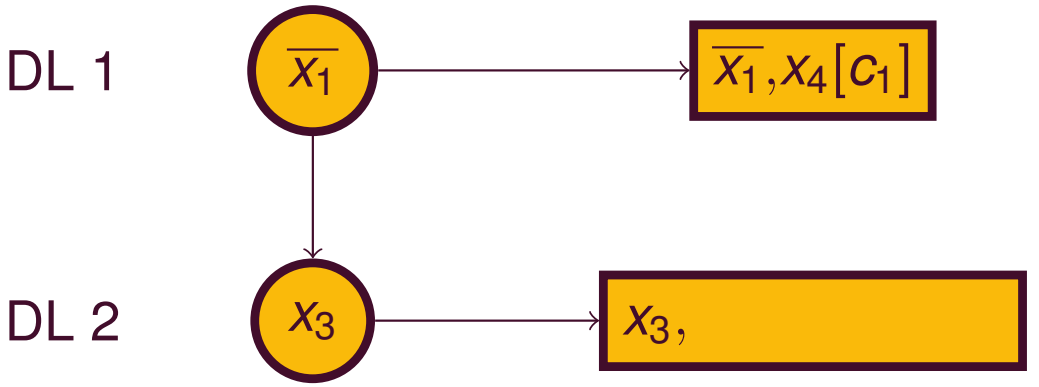
DL 1

$$c_1 = x_1 \vee x_4$$
$$c_2 = x_1 \vee \overline{x_3} \vee \overline{x_8}$$
$$c_3 = x_1 \vee x_8 \vee x_{12}$$
$$c_4 = x_2 \vee x_{11}$$
$$c_5 = \overline{x_3} \vee \overline{x_7} \vee x_{13}$$
$$c_6 = \overline{x_3} \vee \overline{x_7} \vee \overline{x_{13}} \vee x_9$$
$$c_7 = x_8 \vee \overline{x_7} \vee \overline{x_9}$$

A short overview of CDCL solvers

Sequence of decision, propagations

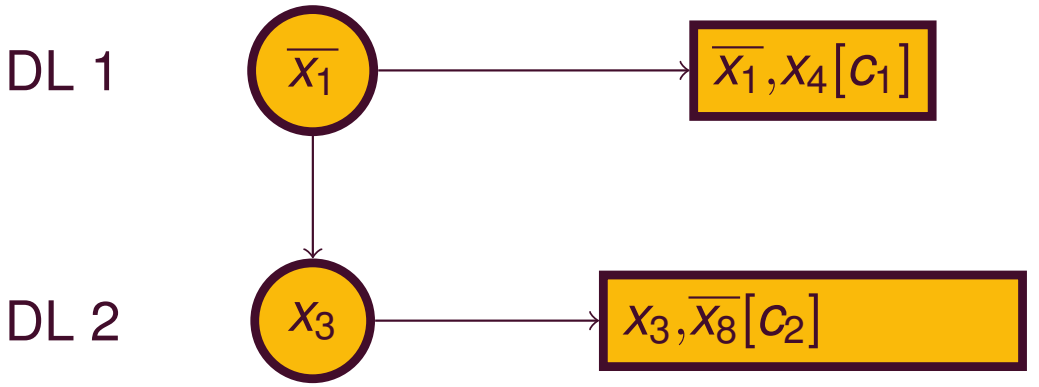
$c_1 = x_1 \vee x_4$
$c_2 = x_1 \vee \overline{x_3} \vee \overline{x_8}$
$c_3 = x_1 \vee x_8 \vee x_{12}$
$c_4 = x_2 \vee x_{11}$
$c_5 = \overline{x_3} \vee \overline{x_7} \vee x_{13}$
$c_6 = \overline{x_3} \vee \overline{x_7} \vee \overline{x_{13}} \vee x_9$
$c_7 = x_8 \vee \overline{x_7} \vee \overline{x_9}$



A short overview of CDCL solvers

Sequence of decision, propagations

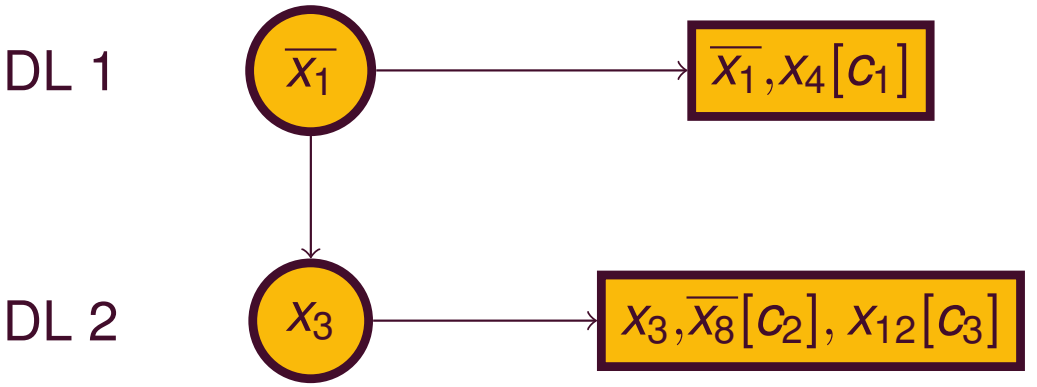
$c_1 = x_1 \vee x_4$
$c_2 = x_1 \vee \overline{x_3} \vee \overline{x_8}$
$c_3 = x_1 \vee x_8 \vee x_{12}$
$c_4 = x_2 \vee x_{11}$
$c_5 = \overline{x_3} \vee \overline{x_7} \vee x_{13}$
$c_6 = \overline{x_3} \vee \overline{x_7} \vee \overline{x_{13}} \vee x_9$
$c_7 = x_8 \vee \overline{x_7} \vee \overline{x_9}$



A short overview of CDCL solvers

Sequence of decision, propagations

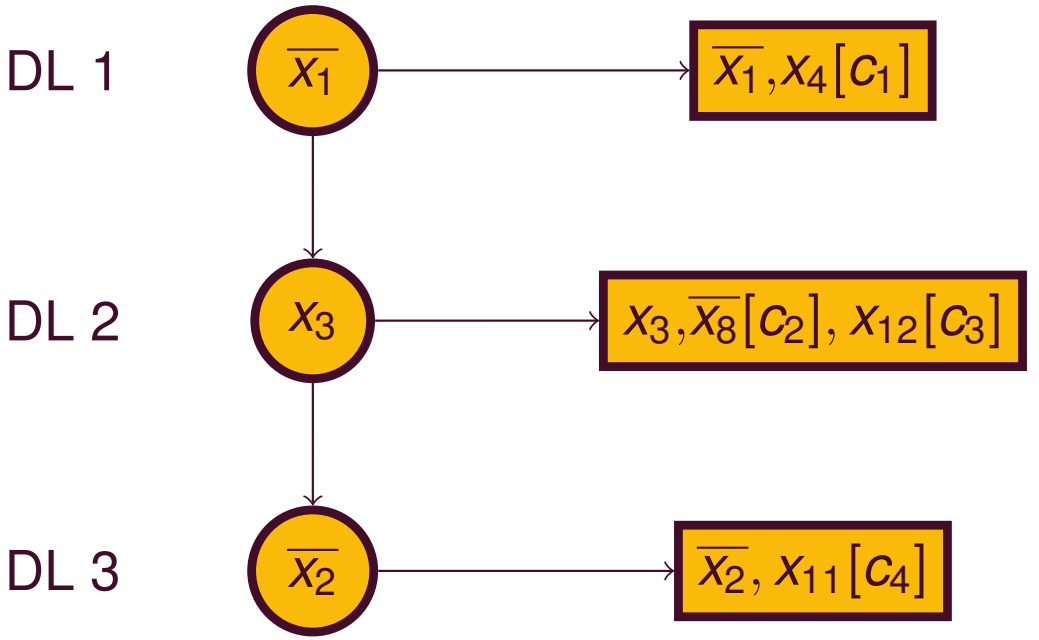
$c_1 = x_1 \vee x_4$
$c_2 = x_1 \vee \overline{x_3} \vee \overline{x_8}$
$c_3 = x_1 \vee x_8 \vee x_{12}$
$c_4 = x_2 \vee x_{11}$
$c_5 = \overline{x_3} \vee \overline{x_7} \vee x_{13}$
$c_6 = \overline{x_3} \vee \overline{x_7} \vee \overline{x_{13}} \vee x_9$
$c_7 = x_8 \vee \overline{x_7} \vee \overline{x_9}$



A short overview of CDCL solvers

Sequence of decision, propagations

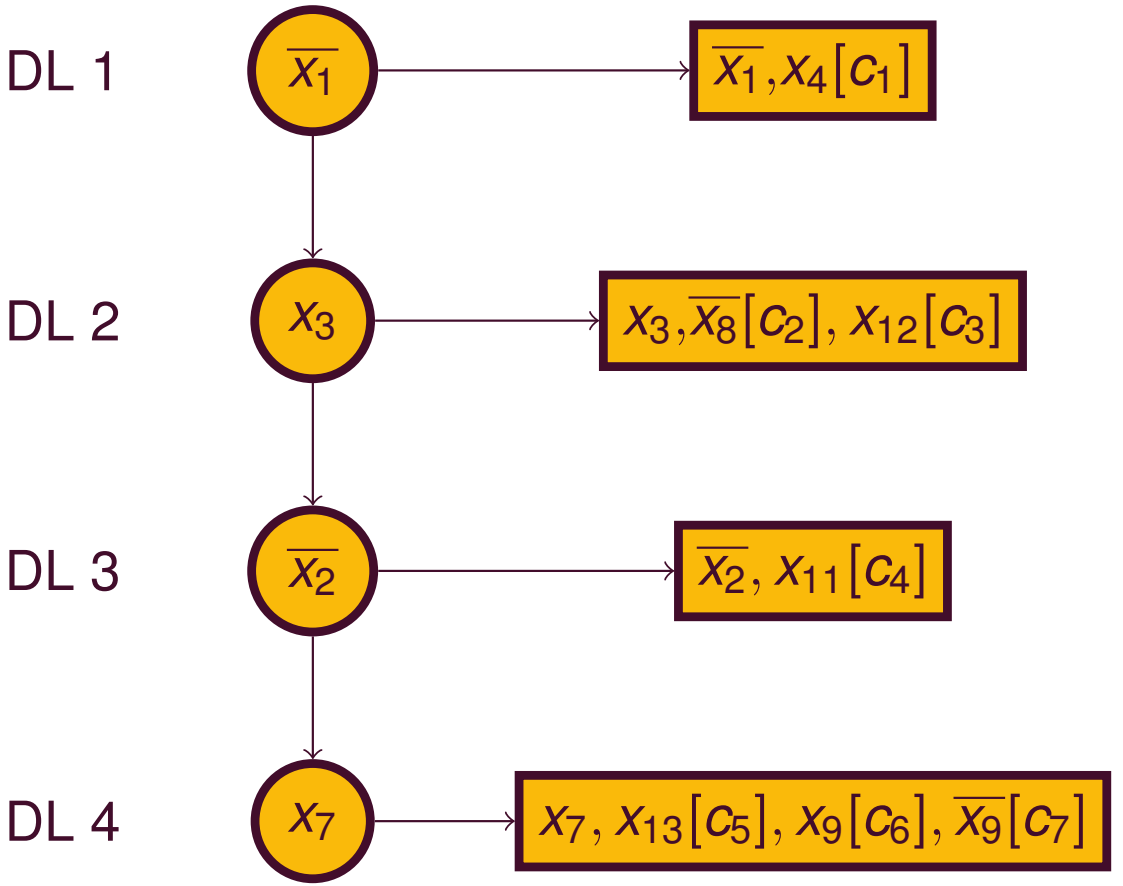
$c_1 = x_1 \vee x_4$
$c_2 = x_1 \vee \overline{x_3} \vee \overline{x_8}$
$c_3 = x_1 \vee x_8 \vee x_{12}$
$c_4 = x_2 \vee x_{11}$
$c_5 = \overline{x_3} \vee \overline{x_7} \vee x_{13}$
$c_6 = \overline{x_3} \vee \overline{x_7} \vee \overline{x_{13}} \vee x_9$
$c_7 = x_8 \vee \overline{x_7} \vee \overline{x_9}$



A short overview of CDCL solvers

Sequence of decision, propagations

$c_1 = x_1 \vee x_4$
$c_2 = x_1 \vee \overline{x_3} \vee \overline{x_8}$
$c_3 = x_1 \vee x_8 \vee x_{12}$
$c_4 = x_2 \vee x_{11}$
$c_5 = \overline{x_3} \vee \overline{x_7} \vee x_{13}$
$c_6 = \overline{x_3} \vee \overline{x_7} \vee \overline{x_{13}} \vee x_9$
$c_7 = x_8 \vee \overline{x_7} \vee \overline{x_9}$



A short overview of CDCL solvers

Conflict analysis

$c_1 = x_1 \vee x_4$
$c_2 = x_1 \vee \overline{x_3} \vee \overline{x_8}$
$c_3 = x_1 \vee x_8 \vee x_{12}$
$c_4 = x_2 \vee x_{11}$
$c_5 = \overline{x_3} \vee \overline{x_7} \vee x_{13}$
$c_6 = \overline{x_3} \vee \overline{x_7} \vee \overline{x_{13}} \vee x_9$
$c_7 = x_8 \vee \overline{x_7} \vee \overline{x_9}$

DL 4

A short overview of CDCL solvers

Conflict analysis

$c_1 = x_1 \vee x_4$
$c_2 = x_1 \vee \overline{x_3} \vee \overline{x_8}$
$c_3 = x_1 \vee x_8 \vee x_{12}$
$c_4 = x_2 \vee x_{11}$
$c_5 = \overline{x_3} \vee \overline{x_7} \vee x_{13}$
$c_6 = \overline{x_3} \vee \overline{x_7} \vee \overline{x_{13}} \vee x_9$
$c_7 = x_8 \vee \overline{x_7} \vee \overline{x_9}$

DL 4

$$d^* = c_7 \otimes_{x_9} c_6 = \overline{x_3} \vee x_8 \vee \overline{x_7} \vee \overline{x_{13}}$$

A short overview of CDCL solvers

Conflict analysis

$c_1 = x_1 \vee x_4$
$c_2 = x_1 \vee \overline{x_3} \vee \overline{x_8}$
$c_3 = x_1 \vee x_8 \vee x_{12}$
$c_4 = x_2 \vee x_{11}$
$c_5 = \overline{x_3} \vee \overline{x_7} \vee x_{13}$
$c_6 = \overline{x_3} \vee \overline{x_7} \vee \overline{x_{13}} \vee x_9$
$c_7 = x_8 \vee \overline{x_7} \vee \overline{x_9}$

DL 4

$d^* = c_7 \otimes_{x_9} c_6 = \overline{x_3} \vee x_8 \vee \overline{x_7} \vee \overline{x_{13}}$
$d_1 = d^* \otimes_{x_{13}} c_5 = \overline{x_3} \vee x_8 \vee \overline{x_7}$

A short overview of CDCL solvers

Conflict analysis

$c_1 = x_1 \vee x_4$
$c_2 = x_1 \vee \overline{x_3} \vee \overline{x_8}$
$c_3 = x_1 \vee x_8 \vee x_{12}$
$c_4 = x_2 \vee x_{11}$
$c_5 = \overline{x_3} \vee \overline{x_7} \vee x_{13}$
$c_6 = \overline{x_3} \vee \overline{x_7} \vee \overline{x_{13}} \vee x_9$
$c_7 = x_8 \vee \overline{x_7} \vee \overline{x_9}$

DL 4

$d^* = c_7 \otimes_{x_9} c_6 = \overline{x_3} \vee x_8 \vee \overline{x_7} \vee \overline{x_{13}}$
$d_1 = d^* \otimes_{x_{13}} c_5 = \overline{x_3} \vee x_8 \vee \overline{x_7}$

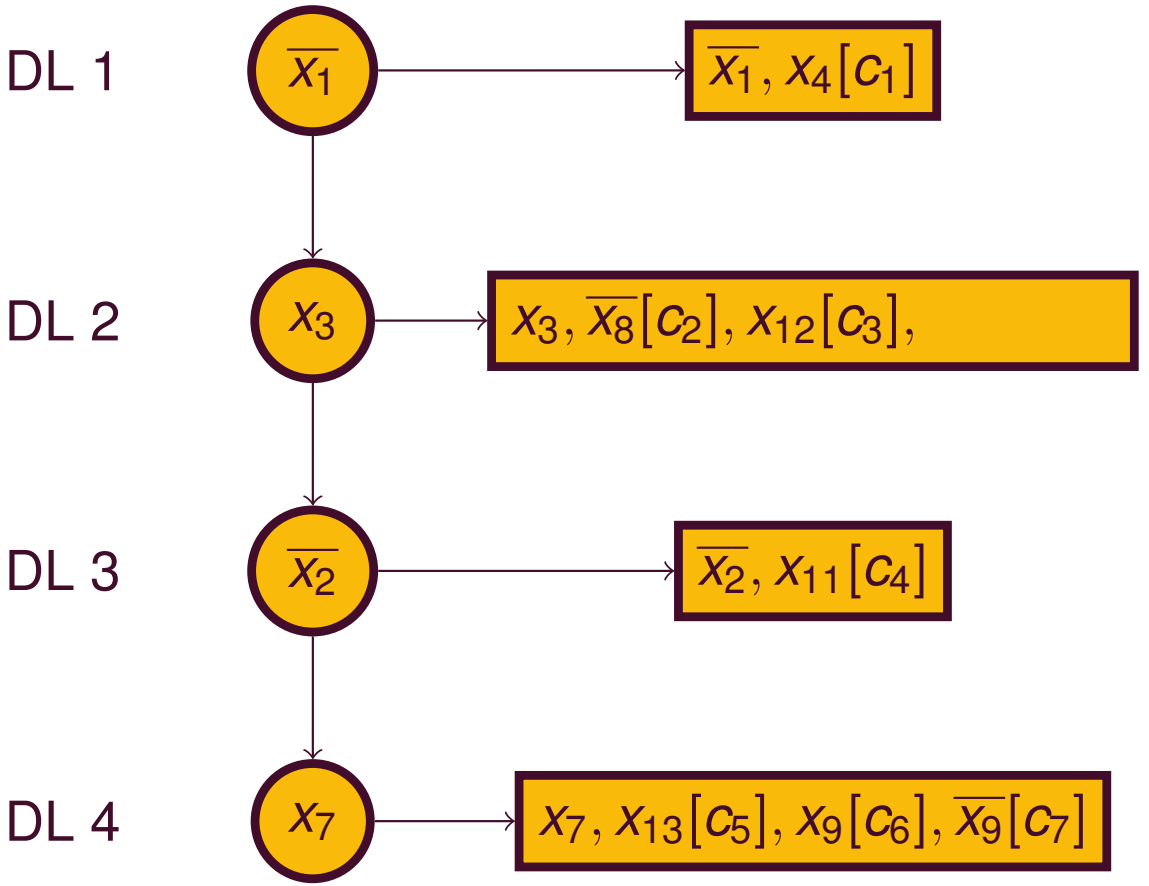
- First resolvant that contains only one literal of the last decision level
- First UIP learning scheme (related to implication graph)
- d_1 is added to the formula and...

A short overview of CDCL solvers

Backjumping

$c_1 = x_1 \vee x_4$
$c_2 = x_1 \vee \overline{x_3} \vee \overline{x_8}$
$c_3 = x_1 \vee x_8 \vee x_{12}$
$c_4 = x_2 \vee x_{11}$
$c_5 = \overline{x_3} \vee \overline{x_7} \vee x_{13}$
$c_6 = \overline{x_3} \vee \overline{x_7} \vee \overline{x_{13}} \vee x_9$
$c_7 = x_8 \vee \overline{x_7} \vee \overline{x_9}$

$d_1 = \overline{x_3} \vee x_8 \vee \overline{x_7}$

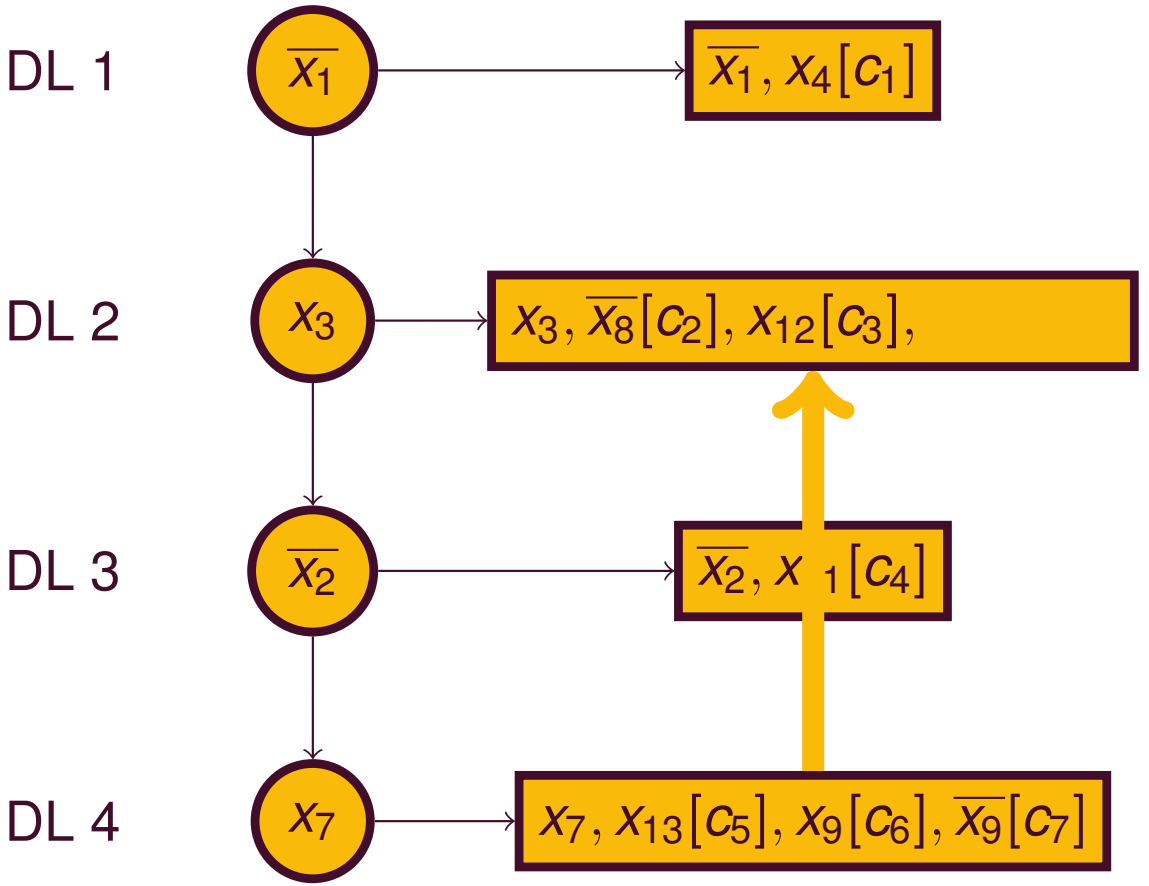


A short overview of CDCL solvers

Backjumping

$c_1 = x_1 \vee x_4$
$c_2 = x_1 \vee \overline{x_3} \vee \overline{x_8}$
$c_3 = x_1 \vee x_8 \vee x_{12}$
$c_4 = x_2 \vee x_{11}$
$c_5 = \overline{x_3} \vee \overline{x_7} \vee x_{13}$
$c_6 = \overline{x_3} \vee \overline{x_7} \vee \overline{x_{13}} \vee x_9$
$c_7 = x_8 \vee \overline{x_7} \vee \overline{x_9}$

$d_1 = \overline{x_3} \vee x_8 \vee \overline{x_7}$

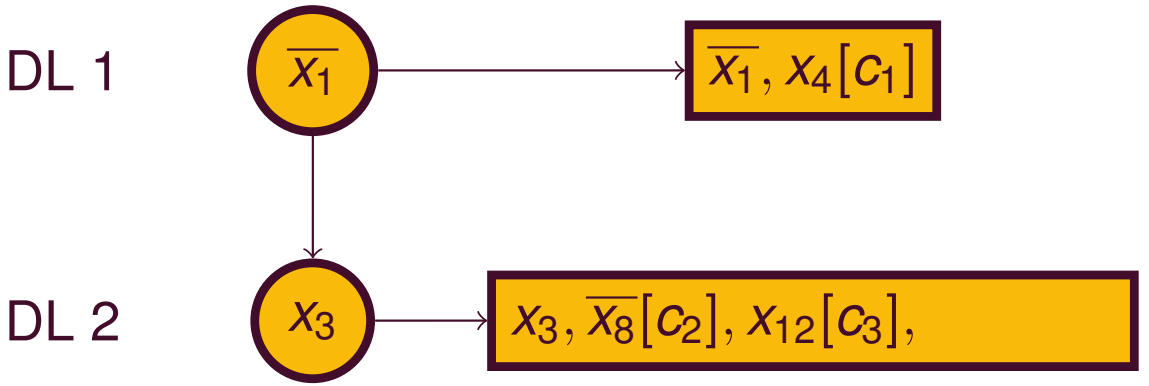


A short overview of CDCL solvers

Backjumping

$c_1 = x_1 \vee x_4$
$c_2 = x_1 \vee \overline{x_3} \vee \overline{x_8}$
$c_3 = x_1 \vee x_8 \vee x_{12}$
$c_4 = x_2 \vee x_{11}$
$c_5 = \overline{x_3} \vee \overline{x_7} \vee x_{13}$
$c_6 = \overline{x_3} \vee \overline{x_7} \vee \overline{x_{13}} \vee x_9$
$c_7 = x_8 \vee \overline{x_7} \vee \overline{x_9}$

$d_1 = \overline{x_3} \vee x_8 \vee \overline{x_7}$

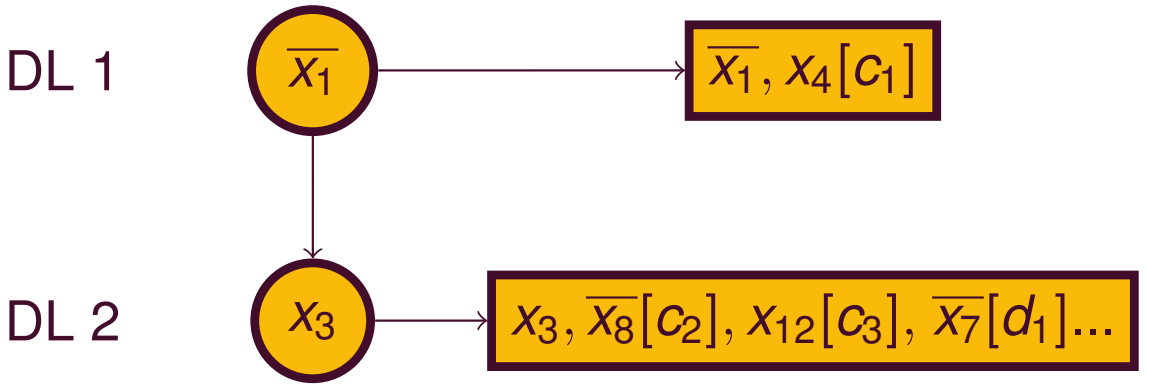


A short overview of CDCL solvers

Backjumping

$c_1 = x_1 \vee x_4$
$c_2 = x_1 \vee \overline{x_3} \vee \overline{x_8}$
$c_3 = x_1 \vee x_8 \vee x_{12}$
$c_4 = x_2 \vee x_{11}$
$c_5 = \overline{x_3} \vee \overline{x_7} \vee x_{13}$
$c_6 = \overline{x_3} \vee \overline{x_7} \vee \overline{x_{13}} \vee x_9$
$c_7 = x_8 \vee \overline{x_7} \vee \overline{x_9}$

$d_1 = \overline{x_3} \vee x_8 \vee \overline{x_7}$

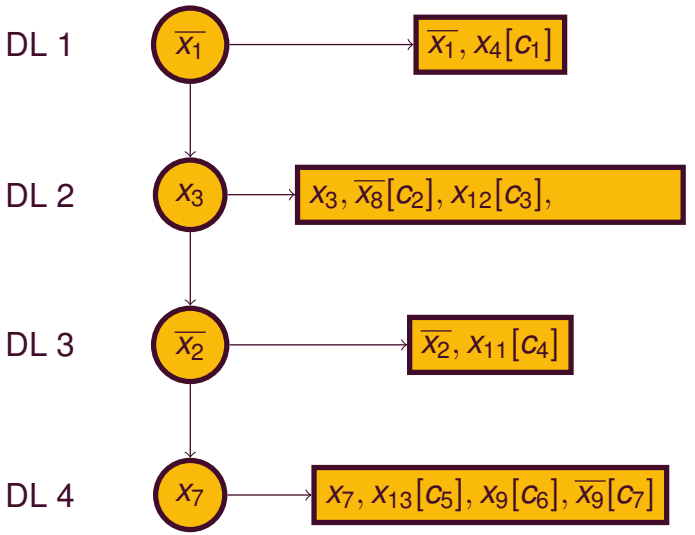


A word on Glucose

- SAT solver developed by Laurent Simon (LABRI) and myself
- Essentially based on a static measure identifying good learnt clauses (LBD)
  - Agressive cleaning strategy : bad clauses have big LBD value
  - Dynamic restarts based on LBD
  - Other features...
- Efficient on UNSAT problems

$c_1 = x_1 \vee x_4$
$c_2 = x_1 \vee \overline{x_3} \vee \overline{x_8}$
$c_3 = x_1 \vee x_8 \vee x_{12}$
$c_4 = x_2 \vee x_{11}$
$c_5 = \overline{x_3} \vee \overline{x_7} \vee x_{13}$
$c_6 = \overline{x_3} \vee \overline{x_7} \vee \overline{x_{13}} \vee x_9$
$c_7 = x_8 \vee \overline{x_7} \vee \overline{x_9}$

$d_1 = \overline{x_3} \vee x_8 \vee \overline{x_7}$



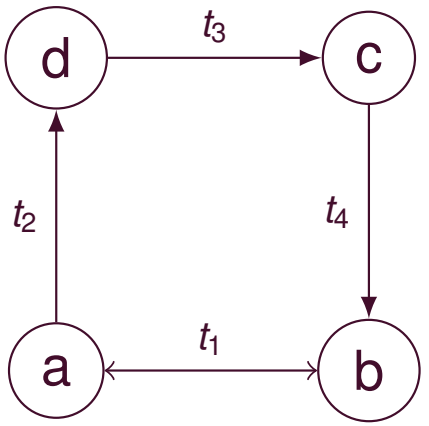
Incremental SAT solving : an introduction

- A surprising effect of solvers' efficiency : used as NP-Complete oracles
  - IC3 : thousands of calls of *simple* formulas [Bradley 2012]
  - MUS extraction [Belov etal. 2012]
  - ...
- Many calls on similar instances
- CDCL solvers learnt form the PAST ! !

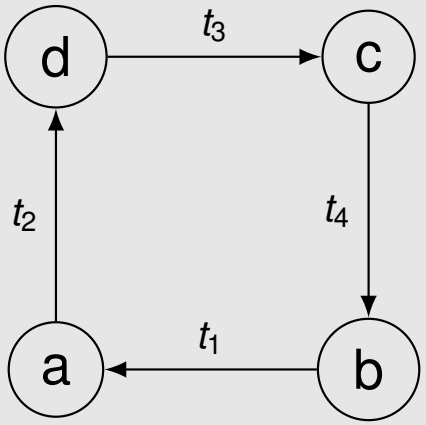
Keep the solver alive

Introduction to BMC

- Bounded Model Checking
- Verification of properties
- Given an automaton with transition states T, an initial state I and a given property P that must be true at each step
- Bound the number of steps (by k), check if P is falsified
- General form : $I(s_0) \wedge T(s_0, s_1) \wedge T(s_1, s_2) \wedge \ldots \wedge T(s_{k-1}, s_k) \wedge \overline{p(s_k)}$
- Instance SAT : a bug is found
- Instance UNSAT : We know .... nothing, we need to increase the bound k

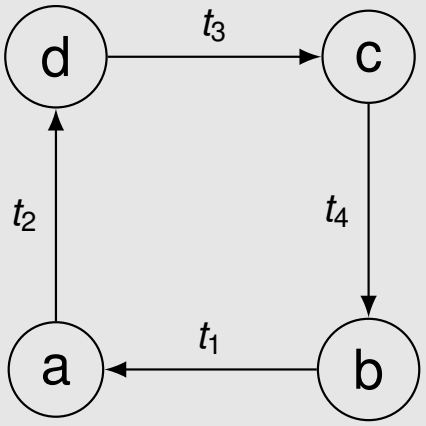


Example



- $a_0, a_1, \ldots$
- $b_0, \ldots, c_0, \ldots, d_0, \ldots$
- $t_1^0, t_1^1, t_1^2 \ldots$
- $t_1^0, t_1^1, t_1^2 \ldots$

Example

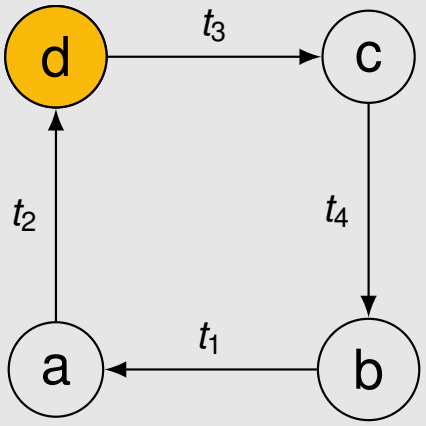


- $a_0, a_1, \ldots$
- $b_0, \ldots, c_0, \ldots, d_0, \ldots$
- $t_1^0, t_1^1, t_1^2 \ldots$
- $t_1^0, t_1^1, t_1^2 \ldots$

Step

a
b
e
d

Example

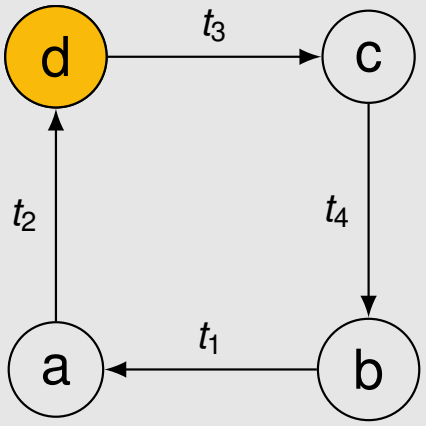


- $a_0, a_1, \ldots$
- $b_0, \ldots, c_0, \ldots, d_0, \ldots$
- $t_1^0, t_1^1, t_1^2 \ldots$
- $t_1^0, t_1^1, t_1^2 \ldots$

Step 0

a	F
b	F
e	F
d	T

Example

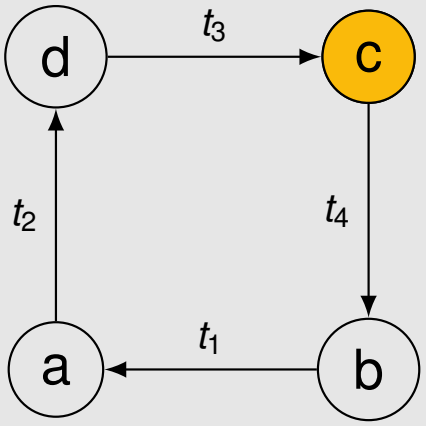


- $a_0, a_1, \ldots$
- $b_0, \ldots, c_0, \ldots, d_0, \ldots$
- $t_1^0, t_1^1, t_1^2 \ldots$
- $t_1^0, t_1^1, t_1^2 \ldots$

Step 0 t_3^0

a	F
b	F
e	F
d	T

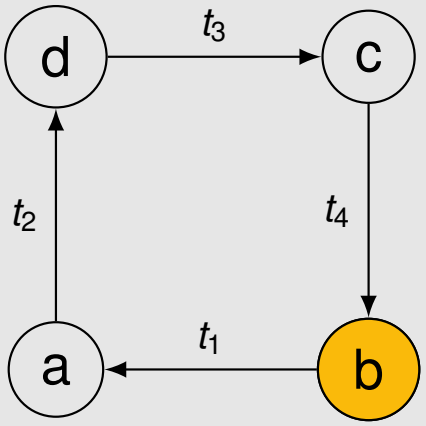
Example



- $a_0, a_1, \ldots$
- $b_0, \ldots, c_0, \ldots, d_0, \ldots$
- $t_1^0, t_1^1, t_1^2 \ldots$
- $t_1^0, t_1^1, t_1^2 \ldots$

Step	0	t_3^0	1
a	F		F
b	F		F
e	F		T
d	T		F

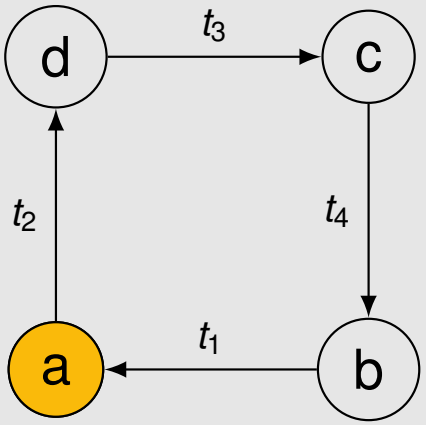
Example



- $a_0, a_1, \ldots$
- $b_0, \ldots, c_0, \ldots, d_0, \ldots$
- $t_1^0, t_1^1, t_1^2 \ldots$
- $t_1^0, t_1^1, t_1^2 \ldots$

Step	0	t_3^0	1	t_4^1	2
a	F		F		F
b	F		F		T
e	F		T		F
d	T		F		F

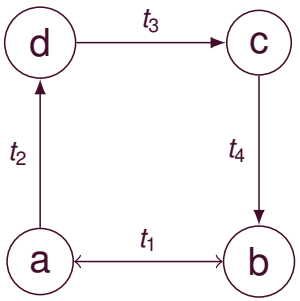
Example



- $a_0, a_1, \ldots$
- $b_0, \ldots, c_0, \ldots, d_0, \ldots$
- $t_1^0, t_1^1, t_1^2 \ldots$
- $t_1^0, t_1^1, t_1^2 \ldots$

Step	0	t_3^0	1	t_4^1	2	t_1^2	3
a	F		F		F		T
b	F		F		T		F
e	F		T		F		F
d	T		F		F		F

Example : Resulting clauses



Initial State :

d_0

Only one state at each step :

$(a_0 \vee b_0 \vee c_0 \vee d_0) \wedge (\neg a_0 \vee \neg b_0) \ldots (\neg c_0 \vee \neg d_0)$
$(a_1 \vee b_1 \vee c_1 \vee d_1) \wedge (\neg a_1 \vee \neg b_1) \ldots (\neg c_1 \vee \neg d_1)$
$(a_2 \vee b_2 \vee c_2 \vee d_2) \wedge (\neg a_2 \vee \neg b_2) \ldots (\neg c_2 \vee \neg d_2)$
$(a_3 \vee b_3 \vee c_3 \vee d_3) \wedge (\neg a_3 \vee \neg b_3) \ldots (\neg c_3 \vee \neg d_3)$

Only one transition at each state :

$(t_1^0 \vee t_2^0 \vee t_3^0 \vee t_4^0) \wedge (\neg t_1^0 \vee \neg t_2^0) \ldots$
$(t_1^1 \vee t_2^1 \vee t_3^1 \vee t_4^1) \wedge (\neg t_1^1 \vee \neg t_2^1) \ldots$
$(t_1^2 \vee t_2^2 \vee t_3^2 \vee t_4^2) \wedge (\neg t_1^2 \vee \neg t_2^2) \ldots$

Transition can occur only if one is on the departure state :

$(\neg d_0 \vee t_3^0) \wedge (\neg c_O \vee t_4^0) \ldots$
$(\neg d_1 \vee t_3^1) \wedge (\neg c_1 \vee t_4^1) \ldots$
$\ldots$

After a transition, one is on the arrival state :

$(\neg t_3^0 \vee c_1) \wedge (\neg t_4^0 \vee b_1) \ldots$
$(\neg t_3^1 \vee c_2) \wedge (\neg t_4^1 \vee b_2) \ldots$

Set of Formulas

- $I(s_0) \wedge T(s_0, s_1) \wedge (\neg p(s_1))$

Set of Formulas

- $I(s_0) \wedge T(s_0, s_1) \wedge (\neg p(s_1))$
- $I(s_0) \wedge T(s_0, s_1) \wedge T(s_1, s_2) \wedge (\neg p(s_1) \vee \neg p(s_2))$

Set of Formulas

- $I(s_0) \wedge T(s_0, s_1) \wedge (\neg p(s_1))$
- $I(s_0) \wedge T(s_0, s_1) \wedge T(s_1, s_2) \wedge (\neg p(s_1) \vee \neg p(s_2))$
- $I(s_0) \wedge T(s_0, s_1) \wedge T(s_1, s_2) \wedge T(s_2, s_3) \wedge (\neg p(s_1) \vee \neg p(s_2) \vee \neg p(s_3))$

Set of Formulas

- $I(s_0) \wedge T(s_0, s_1) \wedge (\neg p(s_1))$
- $I(s_0) \wedge T(s_0, s_1) \wedge T(s_1, s_2) \wedge (\neg p(s_1) \vee \neg p(s_2))$
- $I(s_0) \wedge T(s_0, s_1) \wedge T(s_1, s_2) \wedge T(s_2, s_3) \wedge (\neg p(s_1) \vee \neg p(s_2) \vee \neg p(s_3))$
- $I(s_0) \wedge T(s_0, s_1) \wedge T(s_1, s_2) \wedge T(s_2, s_3) \wedge T(s_3, s_4)$
  $\wedge (\neg p(s_1) \vee \neg p(s_2) \vee \neg p(s_3) \vee \neg p(s_4))$
- ...

Set of Formulas

- $I(s_0) \wedge T(s_0, s_1) \wedge (\neg p(s_1))$
- $I(s_0) \wedge T(s_0, s_1) \wedge T(s_1, s_2) \wedge (\neg p(s_1) \vee \neg p(s_2))$
- $I(s_0) \wedge T(s_0, s_1) \wedge T(s_1, s_2) \wedge T(s_2, s_3) \wedge (\neg p(s_1) \vee \neg p(s_2) \vee \neg p(s_3))$
- $I(s_0) \wedge T(s_0, s_1) \wedge T(s_1, s_2) \wedge T(s_2, s_3) \wedge T(s_3, s_4)$
  $\wedge (\neg p(s_1) \vee \neg p(s_2) \vee \neg p(s_3) \vee \neg p(s_4))$
- ...

- Iterate on the same formula
- We need to add variables (very easy)
- We need to add clauses (very easy)
- We need to remove clauses (harder, see later...)

Incremental SAT : another example

- Verification of properties on some circuit
- Too large too be fully encoded in SAT
- Need to select a sub-circuit and make BMC on it
- Use **assumptions**
- Mimic the environment of the larger circuit by imposing assumptions
- If the result is satisfiable
  - The environment is not set correctly, that is, assumptions are incorrect or missing
  - Or, there is a real bug

Working with Assumptions

- A formula F
- A set of assumptions, $\ell_1, \ell_2, \ldots, \ell_n$ with ℓ_i are literals
- Solve $F \wedge \ell_1 \wedge \ell_2 \ldots \wedge \ell_n$
- Incremental SAT solving : the process can be repeated with new assumptions

Working with Assumptions

- A formula F
- A set of assumptions, $\ell_1, \ell_2, \ldots, \ell_n$ with ℓ_i are literals
- Solve $F \wedge \ell_1 \wedge \ell_2 \ldots \wedge \ell_n$
- Incremental SAT solving : the process can be repeated with new assumptions

First solution

- Simplify : $F' = F \wedge \ell_1 \wedge \ell_2 \ldots \wedge \ell_n$
- Solve F'
- Learnt clauses can not be kept

Working with Assumptions

- A formula F
- A set of assumptions, $\ell_1, \ell_2, \ldots, \ell_n$ with ℓ_i are literals
- Solve $F \wedge \ell_1 \wedge \ell_2 \ldots \wedge \ell_n$
- Incremental SAT solving : the process can be repeated with new assumptions

Working with Assumptions

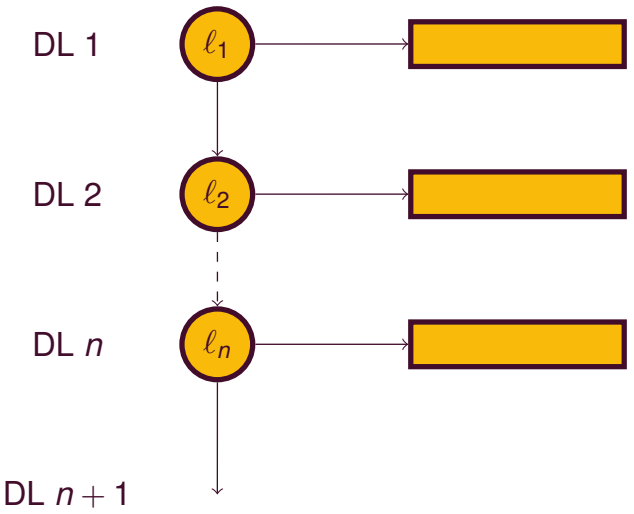
- A formula F
- A set of assumptions, $\ell_1, \ell_2, \ldots, \ell_n$ with ℓ_i are literals
- Solve $F \wedge \ell_1 \wedge \ell_2 \ldots \wedge \ell_n$
- Incremental SAT solving : the process can be repeated with new assumptions

Second Solution

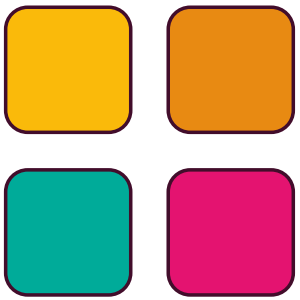
- First, selects all assumptions as decision variables :
  one level => one assumption
- Second, Run the SAT solver as usual
- All learnt clauses can be kept
- One can explain unsatisfiability wrt set of assumptions !

Working with Assumptions

- A formula F
- A set of assumptions, $\ell_1, \ell_2, \ldots, \ell_n$ with ℓ_i are literals



- If $\neg\ell_i$ must be assigned, UNSAT must be returned
- Analysis of the reason of $\neg\ell_i$ gives the subset of assumptions responsible of the conflict



Application to MUS extraction

Minimum Unsatisfiable Subformula

$x \vee y \vee z$

$\neg x \vee y \vee z$

$\neg x \vee \neg y$

$x \vee \neg y$

$x \vee w$

$\neg x \vee \neg z$

UNSAT

$x \vee \neg z$

$w \vee z \vee \neg y$

$w \vee \neg x \vee \neg z$

- The formula is inconsistant : Why ?
- Minimal unsatisfiable subset of clauses

Minimum Unsatisfiable Subformula

$x \vee y \vee z$	$x \vee \neg y$	$x \vee \neg z$
$\neg x \vee y \vee z$	$x \vee w$	$w \vee z \vee \neg y$
$\neg x \vee \neg y$	$\neg x \vee \neg z$	$w \vee \neg x \vee \neg z$

- The formula is inconsistant : Why ?
- Minimal unsatisfiable subset of clauses

Minimum Unsatisfiable Subformula

$x \vee y \vee z$

$\neg x \vee y \vee z$

$\neg x \vee \neg y$

$x \vee \neg y$

$x \vee w$

$\neg x \vee \neg z$

$x \vee \neg z$

$w \vee z \vee \neg y$

$w \vee \neg x \vee \neg z$

- The formula is inconsistant : Why ?
- Minimal unsatisfiable subset of clauses
- Different approaches
  - Local search [Piette et al, ECAI 2006]
  - Resolution based [Nadel, FMCAD 2010]
  - Constructive or destructive [Belov etal, AI Com 2012]. The tool MUSER

Minimum Unsatisfiable Subformula

$\neg x \vee y \vee z$

$\neg x \vee \neg y$

$x \vee \neg y$

$x \vee w$

$\neg x \vee \neg z$

SAT

$x \vee \neg z$

$w \vee z \vee \neg y$

$w \vee \neg x \vee \neg z$

- The formula is inconsistant : Why ?
- Minimal unsatisfiable subset of clauses
- Different approaches
  - Local search [Piette et al, ECAI 2006]
  - Resolution based [Nadel, FMCAD 2010]
  - Constructive or destructive [Belov etal, AI Com 2012]. The tool MUSER

Minimum Unsatisfiable Subformula

$x \vee y \vee z$	$x \vee \neg y$	$x \vee \neg z$
$\neg x \vee y \vee z$	$x \vee w$	$w \vee z \vee \neg y$
$\neg x \vee \neg y$	$\neg x \vee \neg z$	$w \vee \neg x \vee \neg z$

- The formula is inconsistant : Why ?
- Minimal unsatisfiable subset of clauses
- Different approaches
  - Local search [Piette et al, ECAI 2006]
  - Resolution based [Nadel, FMCAD 2010]
  - Constructive or destructive [Belov etal, AI Com 2012]. The tool MUSER

Minimum Unsatisfiable Subformula

$x \vee y \vee z$	$x \vee \neg y$	$x \vee \neg z$
$\neg x \vee y \vee z$	$x \vee w$	$w \vee z \vee \neg y$
$\neg x \vee \neg y$	$\neg x \vee \neg z$	$w \vee \neg x \vee \neg z$

- The formula is inconsistant : Why ?
- Minimal unsatisfiable subset of clauses
- Different approaches
  - Local search [Piette et al, ECAI 2006]
  - Resolution based [Nadel, FMCAD 2010]
  - Constructive or destructive [Belov etal, AI Com 2012]. The tool MUSER

Minimum Unsatisfiable Subformula

$x \vee y \vee z$	$x \vee \neg y$	$x \vee \neg z$
$\neg x \vee y \vee z$		$w \vee z \vee \neg y$
$\neg x \vee \neg y$	$\neg x \vee \neg z$	$w \vee \neg x \vee \neg z$
	UNSAT	

- The formula is inconsistant : Why ?
- Minimal unsatisfiable subset of clauses
- Different approaches
  - Local search [Piette et al, ECAI 2006]
  - Resolution based [Nadel, FMCAD 2010]
  - Constructive or destructive [Belov etal, AI Com 2012]. The tool MUSER

Minimum Unsatisfiable Subformula

$x \vee y \vee z$	$x \vee \neg y$	$x \vee \neg z$
$\neg x \vee y \vee z$		
$\neg x \vee \neg y$	$\neg x \vee \neg z$	$w \vee \neg x \vee \neg z$
	UNSAT	

- The formula is inconsistant : Why ?
- Minimal unsatisfiable subset of clauses
- Different approaches
  - Local search [Piette et al, ECAI 2006]
  - Resolution based [Nadel, FMCAD 2010]
  - Constructive or destructive [Belov etal, AI Com 2012]. The tool MUSER

Minimum Unsatisfiable Subformula

$x \vee y \vee z$ $x \vee \neg y$ $x \vee \neg z$

$\neg x \vee y \vee z$

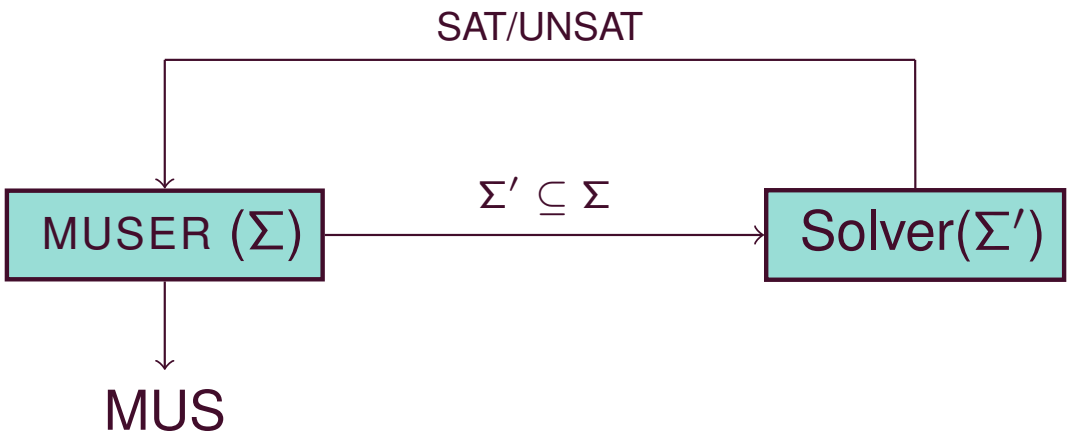
$\neg x \vee \neg y$ $\neg x \vee \neg z$

MUS !

- The formula is inconsistant : Why ?
- Minimal unsatisfiable subset of clauses
- Different approaches
  - Local search [Piette et al, ECAI 2006]
  - Resolution based [Nadel, FMCAD 2010]
  - Constructive or destructive [Belov etal, AI Com 2012]. The tool MUSER

Muser Architecture

Incremental SAT



- Successive calls to a SAT oracle
- Non independent calls
- Informations between two calls are preserved
  - Heuristics : VSIDS, phase saving, restarts...
  - **Learnt clauses**

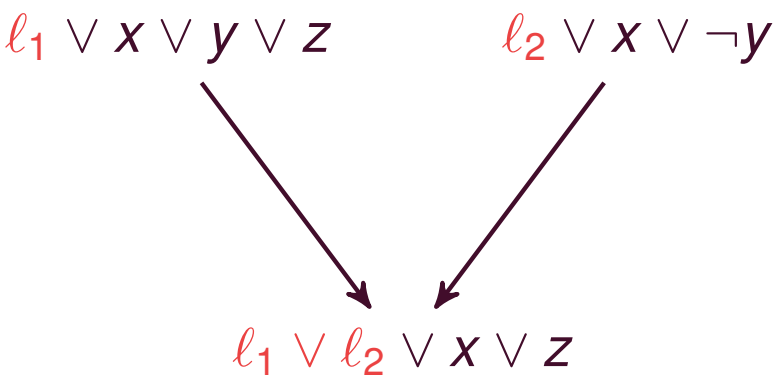
Forget some clauses

- Add one selector (fresh variable) ℓ_i per clause

$\ell_1 \vee x \vee y \vee z$
$\ell_4 \vee \neg x \vee y \vee z$
$\ell_7 \vee \neg x \vee \neg y$

$\ell_2 \vee x \vee \neg y$
$\ell_5 \vee x \vee w$
$\ell_8 \vee \neg x \vee \neg z$

$\ell_3 \vee x \vee \neg z$
$\ell_6 \vee w \vee z \vee \neg y$
$\ell_9 \vee w \vee \neg x \vee \neg z$



- Learnt clause contains selectors of all original clauses used to generate it

Forget some clauses

- Assign ℓ_i (as an assumption) to false to activate the clause i
- Assign ℓ_j (as an assumption) to true to disable the clause j
- All learnt clauses related to the clause j a disable clause are disabled too !

$\ell_1 \vee x \vee y \vee z$
$\ell_2 \vee x \vee \neg y$
$\ell_3 \vee x \vee \neg z$
$\ell_4 \vee \neg x \vee y \vee z$
$\ell_5 \vee x \vee w$
$\ell_6 \vee w \vee z \vee \neg y$
$\ell_7 \vee \neg x \vee \neg y$
$\ell_8 \vee \neg x \vee \neg z$
$\ell_9 \vee w \vee \neg x \vee \neg z$

$\ell_1 \vee \ell_2 \vee x \vee z$

Forget some clauses

- Assign ℓ_i (as an assumption) to false to activate the clause i
- Assign ℓ_j (as an assumption) to true to disable the clause j
- All learnt clauses related to the clause j a disable clause are disabled too !

$\ell_1 \vee x \vee y \vee z$
$\ell_2 \vee x \vee \neg y$
$\ell_3 \vee x \vee \neg z$
$\ell_4 \vee \neg x \vee y \vee z$
$\ell_5 \vee x \vee w$
$\ell_6 \vee w \vee z \vee \neg y$
$\ell_7 \vee \neg x \vee \neg y$
$\ell_8 \vee \neg x \vee \neg z$
$\ell_9 \vee w \vee \neg x \vee \neg z$

$\ell_1 \vee \ell_2 \vee x \vee z$

Forget some clauses

- Assign ℓ_i (as an assumption) to false to activate the clause i
- Assign ℓ_j (as an assumption) to true to disable the clause j
- All learnt clauses related to the clause j a disable clause are disabled too !

$$
\begin{aligned}
& x \vee y \vee z \\
\ell_2 \vee & x \vee \neg y \\
\ell_3 \vee & x \vee \neg z \\
\ell_4 \vee & \neg x \vee y \vee z \\
\ell_5 \vee & x \vee w \\
\ell_6 \vee & w \vee z \vee \neg y \\
\ell_7 \vee & \neg x \vee \neg y \\
\ell_8 \vee & \neg x \vee \neg z \\
\ell_9 \vee & w \vee \neg x \vee \neg z \\
\\
\ell_1 \vee \ell_2 \vee & x \vee z
\end{aligned}
$$

Forget some clauses

- Assign ℓ_i (as an assumption) to false to activate the clause i
- Assign ℓ_j (as an assumption) to true to disable the clause j
- All learnt clauses related to the clause j a disable clause are disabled too !

$$x \vee y \vee z$$
$$\ell_2 \vee x \vee \neg y$$
$$\ell_3 \vee x \vee \neg z$$
$$\ell_4 \vee \neg x \vee y \vee z$$
$$\ell_5 \vee x \vee w$$
$$\ell_6 \vee w \vee z \vee \neg y$$
$$\ell_7 \vee \neg x \vee \neg y$$
$$\ell_8 \vee \neg x \vee \neg z$$
$$\ell_9 \vee w \vee \neg x \vee \neg z$$

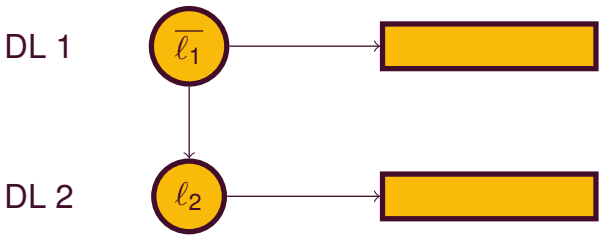
$$\ell_2 \vee x \vee z$$

Forget some clauses

- Assign ℓ_i (as an assumption) to false to activate the clause i
- Assign ℓ_j (as an assumption) to true to disable the clause j
- All learnt clauses related to the clause j a disable clause are disabled too !

$$x \vee y \vee z$$
$$\ell_2 \vee x \vee \neg y$$
$$\ell_3 \vee x \vee \neg z$$
$$\ell_4 \vee \neg x \vee y \vee z$$
$$\ell_5 \vee x \vee w$$
$$\ell_6 \vee w \vee z \vee \neg y$$
$$\ell_7 \vee \neg x \vee \neg y$$
$$\ell_8 \vee \neg x \vee \neg z$$
$$\ell_9 \vee w \vee \neg x \vee \neg z$$

$$\ell_2 \vee x \vee z$$



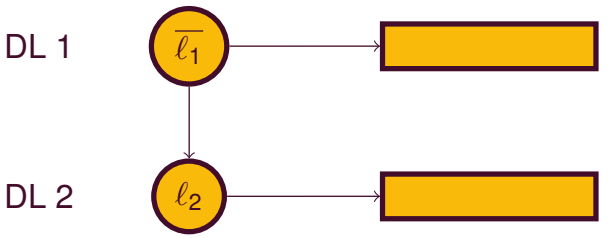
Forget some clauses

- Assign ℓ_i (as an assumption) to false to activate the clause i
- Assign ℓ_j (as an assumption) to true to disable the clause j
- All learnt clauses related to the clause j a disable clause are disabled too !

$$x \vee y \vee z$$

$$\ell_3 \vee x \vee \neg z$$
$$\ell_4 \vee \neg x \vee y \vee z$$
$$\ell_5 \vee x \vee w$$
$$\ell_6 \vee w \vee z \vee \neg y$$
$$\ell_7 \vee \neg x \vee \neg y$$
$$\ell_8 \vee \neg x \vee \neg z$$
$$\ell_9 \vee w \vee \neg x \vee \neg z$$

$$\ell_2 \vee x \vee z$$

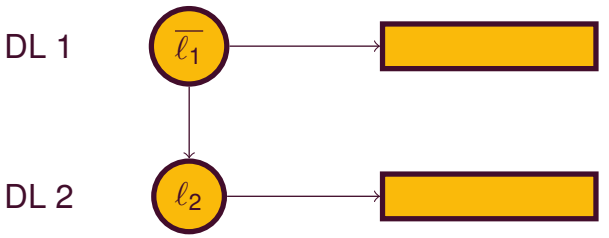


Forget some clauses

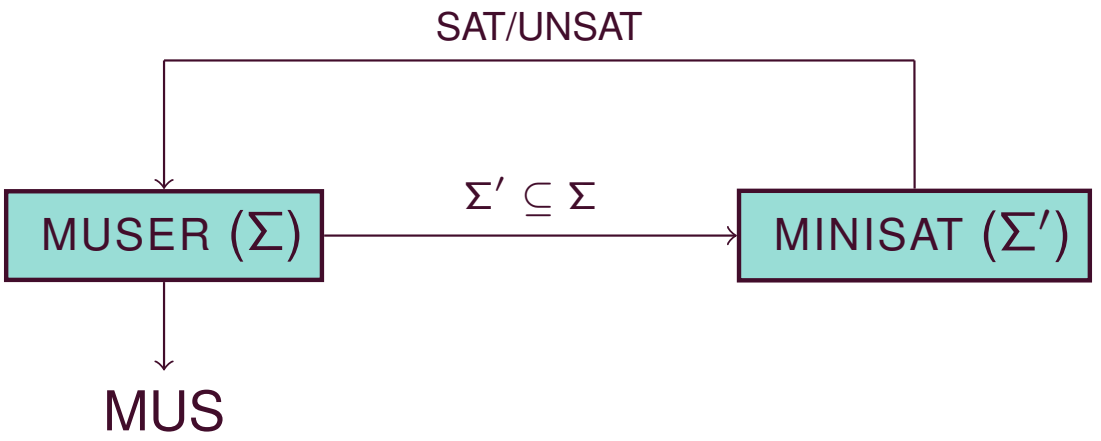
- Assign ℓ_i (as an assumption) to false to activate the clause i
- Assign ℓ_j (as an assumption) to true to disable the clause j
- All learnt clauses related to the clause j a disable clause are disabled too !

$$x \vee y \vee z$$

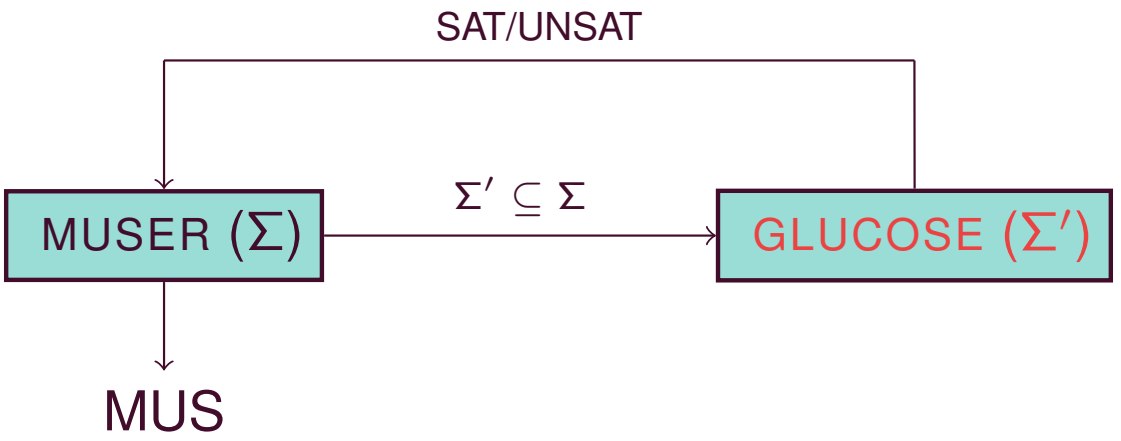
$$\ell_3 \vee x \vee \neg z$$
$$\ell_4 \vee \neg x \vee y \vee z$$
$$\ell_5 \vee x \vee w$$
$$\ell_6 \vee w \vee z \vee \neg y$$
$$\ell_7 \vee \neg x \vee \neg y$$
$$\ell_8 \vee \neg x \vee \neg z$$
$$\ell_9 \vee w \vee \neg x \vee \neg z$$



Glucose inside Muser



Glucose inside Muser



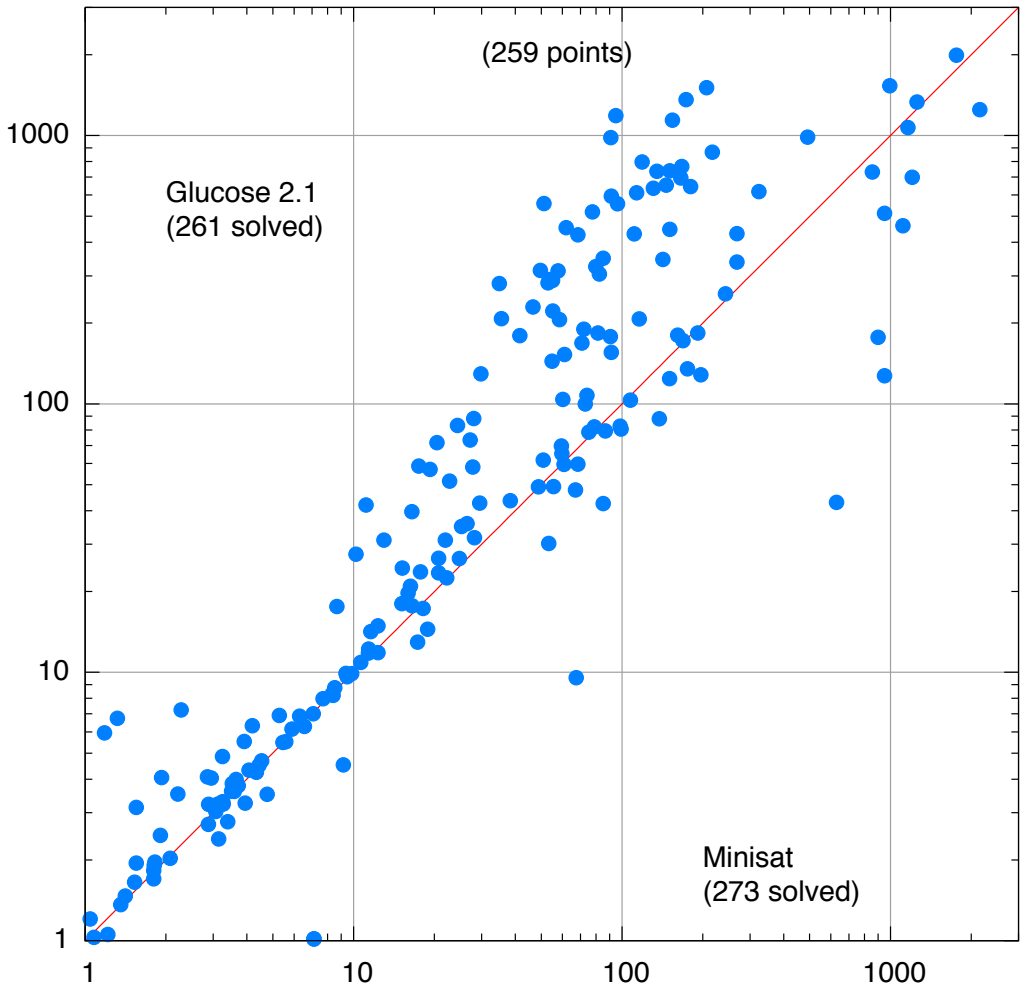
- Plug GLUCOSE in MUSER
- Adapt and modify GLUCOSE to improve MUSER performances

Improve SAT oracle in order to improve the MUSER tool

Test set

- 300 instances from the SAT competition 2011, MUS category
- timeout set to 2400 seconds
- MUSER is used with default options (destructive approach, model rotation)

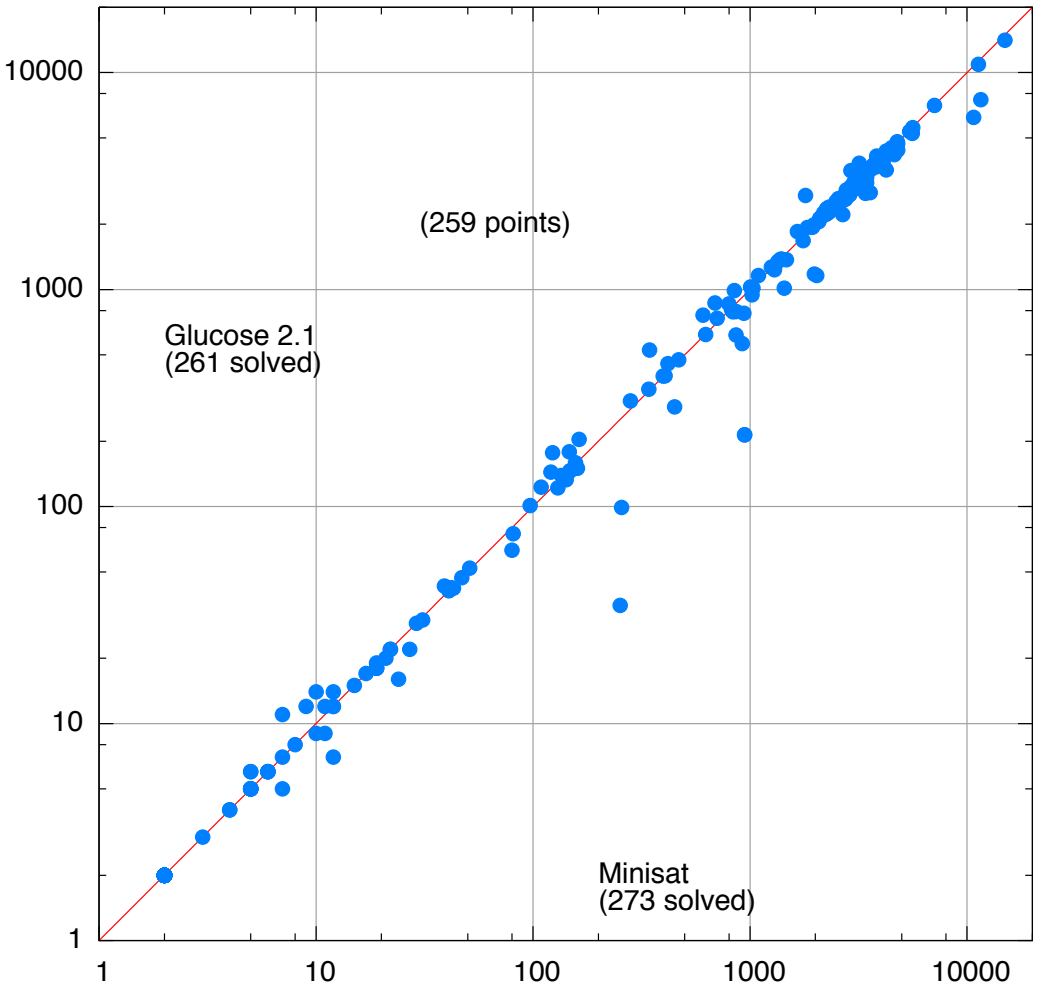
A first Attempt



Disappointing results

Trying to explain these bad results

Disappointing results



Disappointing results

Trying to explain these bad results

- Comparable number of oracle calls
- Easy SAT calls
- Difficult UNSAT ones
- GLUCOSE is supposed to be good on UNSAT formulas

Disappointing results

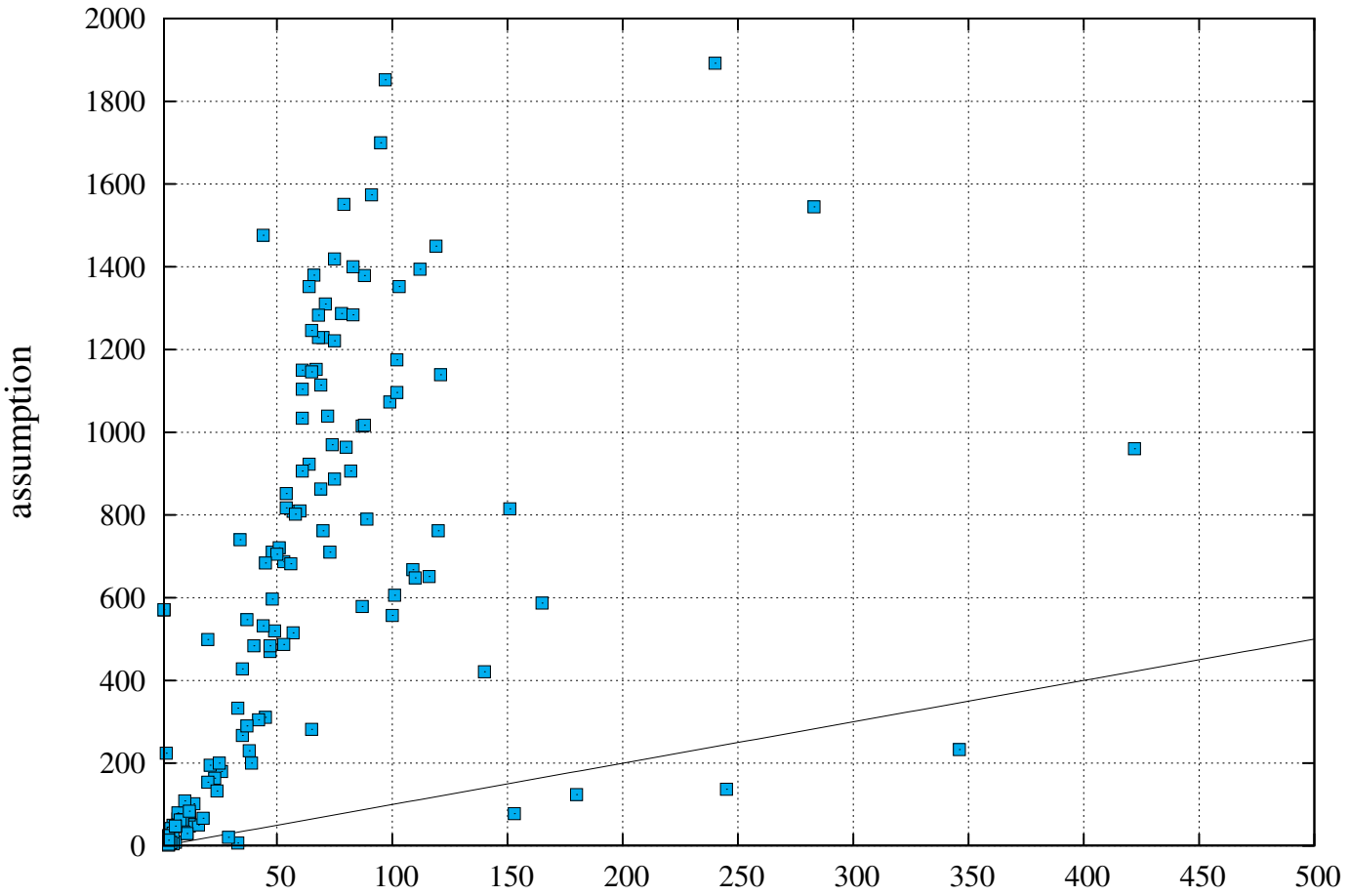
Trying to explain these bad results

- Comparable number of oracle calls
- Easy SAT calls
- Difficult UNSAT ones
- GLUCOSE is supposed to be good on UNSAT formulas

- GLUCOSE uses LBD for cleaning, restarts...
- Each assumption uses its own decision level

Disappointing results

- Each point represents an instance
- x-axis is the average number of initial variables in learnt clauses
- y-axis is the average number of selector variables in learnt clauses



Disappointing results

		LBD				
			size		LBD	
Instance	#C	time	avg	max	avg	max
fdmus_b21_96	8541	29	1145	5980	1095	5945
longmult6	8853	46	694	3104	672	3013
dump_vc950	360419	110	522	36309	498	35873
g7n	70492	190	1098	16338	1049	16268

- LBD looks like size
- Clauses are very long

Disappointing results

Trying to explain these bad results

- Comparable number of oracle calls
- Easy SAT calls
- Difficult UNSAT ones
- GLUCOSE is supposed to be good on UNSAT formulas

- GLUCOSE uses LBD for cleaning, restarts...
- Each assumption uses its own decision level
- **The LBD of a clause looks like its size !**

Disappointing results

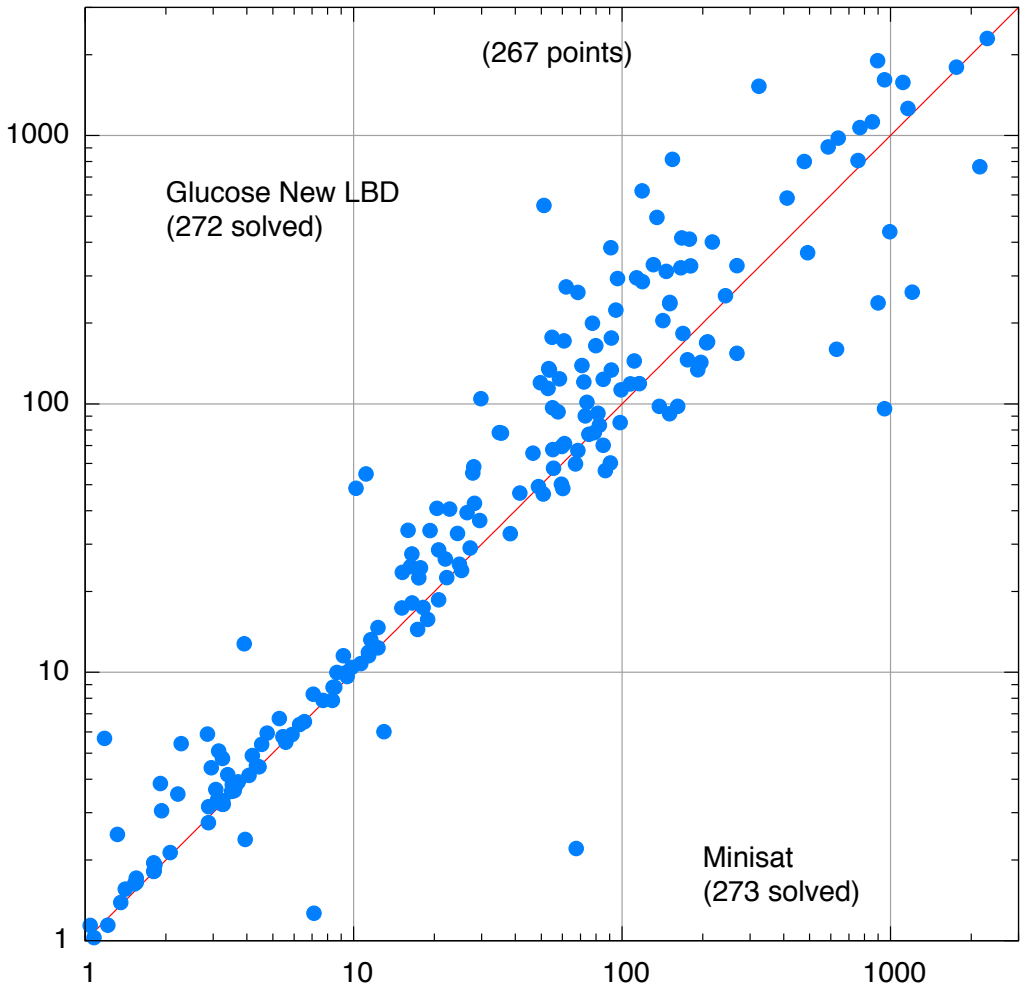
Trying to explain these bad results

- Comparable number of oracle calls
- Easy SAT calls
- Difficult UNSAT ones
- GLUCOSE is supposed to be good on UNSAT formulas

- GLUCOSE uses LBD for cleaning, restarts...
- Each assumption uses its own decision level
- **The LBD of a clause looks like its size !**

Refine LBD : Do not take into account selectors

A second attempt



New LBD

		LBD					New LBD				
			size		LBD			size		LBD	
Instance	#C	time	avg	max	avg	max	time	avg	max	avg	max
fdmus_b21_96	8541	29	1145	5980	1095	5945	**11**	972	6391	**8**	71
longmult6	8853	46	694	3104	672	3013	**14**	627	2997	**11**	61
dump_vc950	360419	110	522	36309	498	35873	**67**	1048	36491	**8**	307
g7n	70492	190	1098	16338	1049	16268	**75**	1729	17840	**27**	160

LBD matters

- However, results need to be improve yet

Clauses are too long

Many algorithms have to traverse clauses

- Dynamic computing of LBD (useful but costly)
- Conflict analysis
- Unit propagation
- Deleting satisfiable clauses

Clauses are too long

Many algorithms have to traverse clauses

- Dynamic computing of LBD (useful but costly)
  - → Store the number of selectors in the clause
  - → Stop when all initial literals have been tested
- Conflict analysis
- Unit propagation
- Deleting satisfiable clauses

Clauses are too long

Many algorithms have to traverse clauses

- Dynamic computing of LBD (useful but costly)
  - → Store the number of selectors in the clause
  - → Stop when all initial literals have been tested
- Conflict analysis
  - → Force initial literals to be placed at the beginning
- Unit propagation
- Deleting satisfiable clauses

Clauses are too long

Many algorithms have to traverse clauses

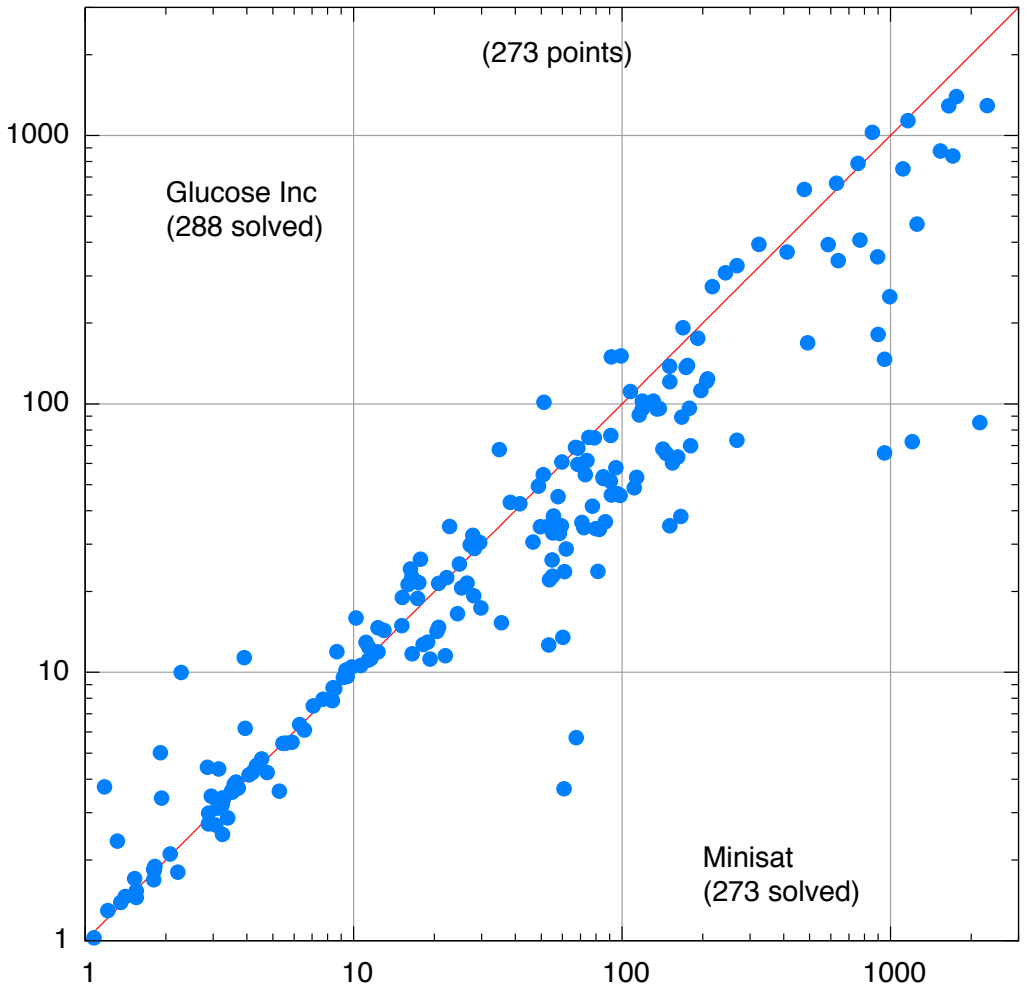
- Dynamic computing of LBD (useful but costly)
  - → Store the number of selectors in the clause
  - → Stop when all initial literals have been tested
- Conflict analysis
  - → Force initial literals to be placed at the beginning
- Unit propagation
  - → Look for a non selector literal or a satisfied one
  - → Push selectors at the end of the clause
- Deleting satisfiable clauses

Clauses are too long

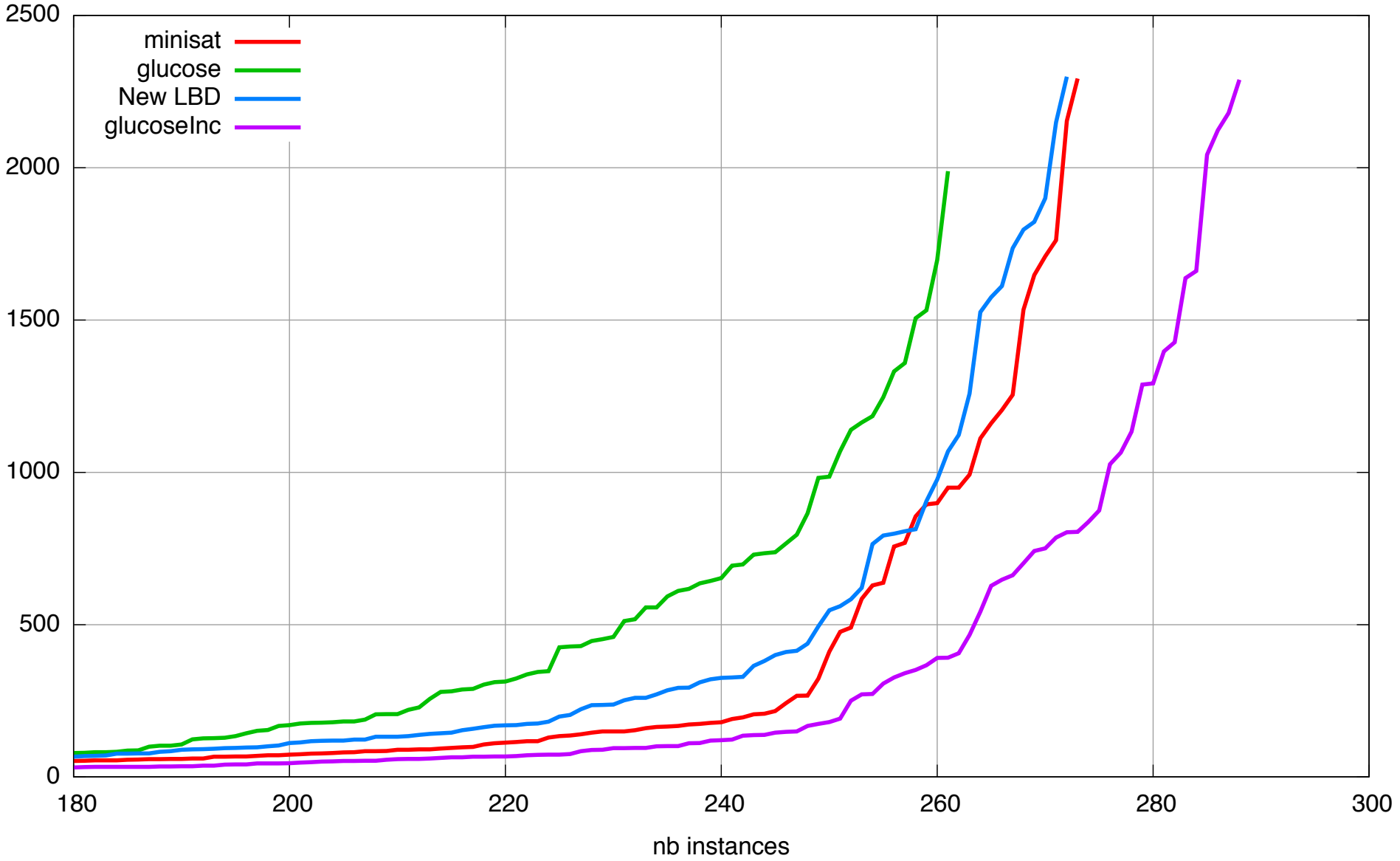
Many algorithms have to traverse clauses

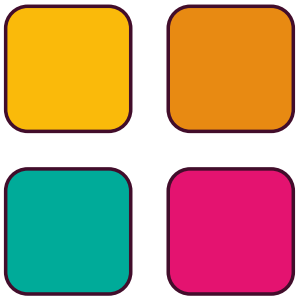
- Dynamic computing of LBD (useful but costly)
  - → Store the number of selectors in the clause
  - → Stop when all initial literals have been tested
- Conflict analysis
  - → Force initial literals to be placed at the beginning
- Unit propagation
  - → Look for a non selector literal or a satisfied one
  - → Push selectors at the end of the clause
- Deleting satisfiable clauses
  - → Take only watched literals into account

Third attempt



Final comparison





Application to knowledge compilation

Autonomous Systems

- Autonomous system : must be able to make decisions depending on the current situation
- Finding the decision policy
- Problem hard to solve

Explorer Robot

- Explore a zone
- Gather informations
- When enough informations have been gathered, go to another zone

Slides from Alexandre Niveau's PhD defense

Control of an Autonomous Systems

- Solving the problem online
- Limited by the embedded computational power
- Reactivity not ensured

- Solving the problem offline
- Anticipate all decisions for every possible situation
- Reactivity is ensured
- Limited by embedded memory ressources

Slides from Alexandre Niveau's PhD defense

Looking for a Compromise

- Need a tradeoff between reactivity and spatial compactness
- Maximizing reactivity under memory space constraints
- This is the object of knowledge compilation

Idea

- Transform the problem into a compiled form
- Make its resolution tractable
- Is as compact as possible

Slides from Alexandre Niveau's PhD defense

Two phases

Offline phase

- Theory is compiled into a target language
- Can be slow
- Need to be compact
- Some target languages : (RO)BDD, Tree of BD, DNNF, DNNF, SDD

Online phase

- The compiled target is used to efficiently answer a number of queries
- Need to be fast (polynomial ? ?)

Some Queries

- Consistency ($F \models \bot$?)
- Clause entailment ($F \models c$?)
- Implicant ($t \models F$?)
- Equivalence ($F \equiv G$?)
- Sentential Entailment ($F \models G$?)
- Model Counting
- Model Enumeration

Some Operators

- Conditioning : subset of variables is set to true/false
- Forgetting : Some variables are removed
- Conjunction : $F \wedge G$
- Disjunction $F \vee G$
- Negation $\neg F$

Knowledge Compilation Map

- "*A Knowledge Compilation Map*". Adnan Darwiche, Pierre Marquis. J. Artif. Intell. Res. (JAIR) 17 : 229-264 (2002)

L	CO	VA	CE	IM	EQ	SE	CT	ME
NNF	○	○	○	○	○	○	○	○
DNNF	√	○	√	○	○	○	○	√
d-NNF	○	○	○	○	○	○	○	○
s-NNF	○	○	○	○	○	○	○	○
f-NNF	○	○	○	○	○	○	○	○
d-DNNF	√	√	√	√	?	○	√	√
sd-DNNF	√	√	√	√	?	○	√	√
BDD	○	○	○	○	○	○	○	○
FBDD	√	√	√	√	?	○	√	√
OBDD	√	√	√	√	√	○	√	√
$\text{OBDD}_<$	√	√	√	√	√	√	√	√
DNF	√	○	√	○	○	○	○	√
CNF	○	√	○	√	○	○	○	○
PI	√	√	√	√	√	√	○	√
IP	√	√	√	√	√	√	○	√
MODS	√	√	√	√	√	√	√	√

√ : polynomial query

○ no polynomial unless $P = NP$

Knowledge Compilation Map

- "*A Knowledge Compilation Map*". Adnan Darwiche, Pierre Marquis. J. Artif. Intell. Res. (JAIR) 17 : 229-264 (2002)

L	CD	FO	SFO	∧C	∧BC	∨C	∨BC	¬C
NNF	√	○	√	√	√	√	√	√
DNNF	√	√	√	○	○	√	√	○
d-NNF	√	○	√	√	√	√	√	√
s-NNF	√	○	√	√	√	√	√	√
f-NNF	√	○	√	•	•	•	•	√
d-DNNF	√	○	○	○	○	○	○	?
sd-DNNF	√	○	○	○	○	○	○	?
BDD	√	○	√	√	√	√	√	√
FBDD	√	•	○	•	○	•	○	√
OBDD	√	•	√	•	○	•	○	√
$\text{OBDD}_<$	√	•	√	•	√	•	√	√
DNF	√	√	√	•	√	√	√	•
CNF	√	○	√	√	√	•	√	•
PI	√	√	√	•	•	•	√	•
IP	√	•	•	•	√	•	•	•
MODS	√	√	√	•	√	•	•	•

√ : polynomial

○ no polynomial unless $P = NP$

• no polynomial

Our Proposal

- Forget offline phase
- Use (incremental) SAT
- No theoretical guarantee about effectiveness
- Practical guarantee about effectiveness ?

Our Proposal

- Forget offline phase
- Use (incremental) SAT
- No theoretical guarantee about effectiveness
- Practical guarantee about effectiveness ?

- Clause entailment is easy : $F \wedge \neg c$, $\neg c$ is treated with assumptions
- Implicant (quite easy)
- Model counting, possible but. . .

Our Proposal

- Forget offline phase
- Use (incremental) SAT
- No theoretical guarantee about effectiveness
- Practical guarantee about effectiveness ?

- Clause entailment is easy : $F \wedge \neg c$, $\neg c$ is treated with assumptions
- Implicant (quite easy)
- Model counting, possible but. . .

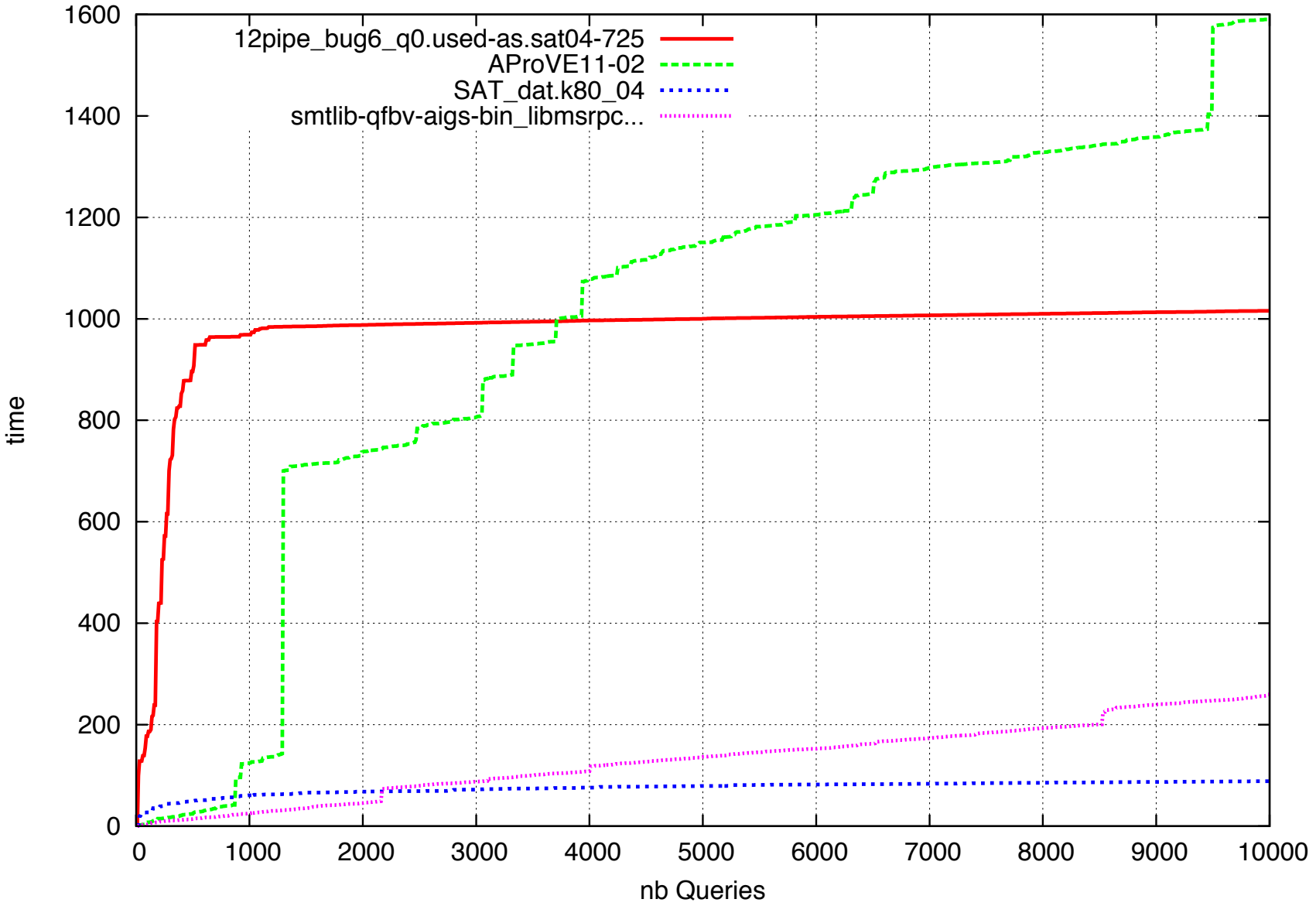
- Conditioning is easy (use assumptions), conjunction also
- Forgetting is more difficult to handle

Some Experiments

- First experiments with benchmarks from ToBDD paper : Too easy !
- Pick some SAT benchmarks from the SAT competition 2011
- Performing 10,000 random queries (Clause Entailment)

Instance		Incremental				No Incremental			ToBDD
#V	#C	time	avg	unit lits	conflicts	time	avg	conflicts	Time
106,815	552,819	2,088	0.2	4,652	1,169	TO		$>$ 5,000	–
17,914	147,360	14	0.001	13,945	85	403	0.4	4,000	10
81,184	4,547,858	1,046	0.1	8,987	4,371	TO		$>$15,000	–
42,375	262,415	101	0.01	12,744	3,288	TO		$>$13,000	–
42,608	212,741	482	0.04	0	2,805	730	0.7	1,042	–

Amortization



Conclusion

- Incremental SAT solving
- CDCL SAT solvers learn from the past
- Keep the solver alive ! !
- Possibility to add variables and clauses
- Possibility to remove clauses
- Many potential applications

Some references

"*Incremental SAT solving*". J. Hooker. J. Log. Program. 15 : 177-186 (1993).

"*Pruning Techniques for the SAT-Based Bounded Model Checking Problem*". O. Strichman. CHARME 2001 : 58-70.

"*SATIRE : A New Incremental Satisfiability Engine*". J. Whittemore, J. Kim, K. Sakallah. DAC 2001 : 542-545.

" *Temporal induction by incremental SAT solving*. N. Eén, N. Sörensson. Electr. Notes Theor. Comput. Sci. 89(4) : 543-560 (2003).

"*Efficient SAT Solving under Assumptions*". A. Nadel, V. Ryvchin. . SAT 2012 : 242-255.

"*Preprocessing in Incremental SAT*". A. Nadel, V. Ryvchin, O. Strichman. SAT 2012 : 256-269

"*Improving Glucose for Incremental SAT Solving with Assumptions : Application to MUS Extraction*". G. Audemard, JM. Lagniez, L. Simon. SAT 2013 : 309-317.

"*Just-In-Time Compilation of Knowledge Bases*". G. Audemard, JM. Lagniez, L. Simon. IJCAI 2013.