SAT Oracles, for NP-Complete Problems and Beyond

Combinatorial problem solving using SAT solvers

Daniel Le Berre

CNRS - Université d'Artois

ECSA/SPLC 19, Paris, September 11-13, 2019

UNIVERSITÉ D'ARTOIS

Purpose of this talk

- Using SAT solvers are black boxes
- Importance of the interaction with the solver
- Importance of encodings
- When encodings are too large

Outline

SAT, SAT Oracle, SAT Solver

Importance of the interaction with the solver

Importance of the encodings

When encodings are too large

The SAT problem: textbook definition

Definition

Input : A set of clauses C built from a propositional language with n variables.

Output : Is there an assignment of the n variables that satisfies all those clauses?

The SAT problem: textbook definition

Definition

Input : A set of clauses C built from a propositional language with n variables.
Output : Is there an assignment of the n variables that satisfies all those clauses?

Example

$$C_1 = \{\neg a \vee b, \neg b \vee c\} = (\neg a \vee b) \wedge (\neg b \vee c) = (a' + b).(b' + c)$$

$$C_2 = C_1 \cup \{a, \neg c\} = C_1 \wedge a \wedge \neg c$$

For C_1, the answer is yes, for C_2 the answer is no

$$C_1 \models \neg(a \wedge \neg c) = \neg a \vee c$$

The SAT ~~problem~~ solver: practical point of view 1/3

Definition

Input : A set of clauses C built from a propositional language with n variables.

Output : If there is an assignment of the n variables that satisfies all those clauses, provide such assignment, else answer UNSAT.

The SAT ~~problem~~ solver: practical point of view 1/3

Definition

Input : A set of clauses C built from a propositional language with n variables.

Output : If there is an assignment of the n variables that satisfies all those clauses, provide such assignment, else answer UNSAT.

Example

$$C_1 = \{\neg a \vee b, \neg b \vee c\} = (\neg a \vee b) \wedge (\neg b \vee c) = (a' + b).(b' + c)$$

$$C_2 = C_1 \cup \{a, \neg c\} = C_1 \wedge a \wedge \neg c$$

For C_1, one answer is $\{a, b, c\}$, for C_2 the answer is UNSAT.

The SAT ~~problem~~ solver: practical point of view 1/3

Definition

Input : A set of clauses C built from a propositional language with n variables.

Output : If there is an assignment of the n variables that satisfies all those clauses, provide such assignment, else answer UNSAT.

Example

$$C_1 = \{\neg a \vee b, \neg b \vee c\} = (\neg a \vee b) \wedge (\neg b \vee c) = (a' + b).(b' + c)$$

$$C_2 = C_1 \cup \{a, \neg c\} = C_1 \wedge a \wedge \neg c$$

For C_1, one answer is $\{a, b, c\}$, for C_2 the answer is UNSAT.

SAT answers can be checked: trusted model oracle

The SAT ~~problem~~ solver: practical point of view 2/3

Definition

Input : A set of clauses C built from a propositional language with n variables.

Output : If there is an assignment of the n variables that satisfies all those clauses, provide such assignment, else provide a subset of C which cannot be satisfied.

The SAT ~~problem~~ solver: practical point of view 2/3

Definition

Input : A set of clauses C built from a propositional language with n variables.

Output : If there is an assignment of the n variables that satisfies all those clauses, provide such assignment, else provide a subset of C which cannot be satisfied.

Example

$$C_1 = \{\neg a \vee b, \neg b \vee c\} = (\neg a \vee b) \wedge (\neg b \vee c) = (a' + b).(b' + c)$$

$$C_2 = C_1 \cup \{a, \neg c\} = C_1 \wedge a \wedge \neg c$$

For C_1, one answer is $\{a, b, c\}$, for C_2 the answer is C_2

The SAT ~~problem~~ solver: practical point of view 2/3

Definition

Input : A set of clauses C built from a propositional language with n variables.

Output : If there is an assignment of the n variables that satisfies all those clauses, provide such assignment, else provide a subset of C which cannot be satisfied.

Example

$$C_1 = \{\neg a \vee b, \neg b \vee c\} = (\neg a \vee b) \wedge (\neg b \vee c) = (a' + b).(b' + c)$$

$$C_2 = C_1 \cup \{a, \neg c\} = C_1 \wedge a \wedge \neg c$$

For C_1, one answer is $\{a, b, c\}$, for C_2 the answer is C_2

UNSAT core may explain inconsistency if much smaller than C: informative UNSAT oracle

The SAT ~~problem~~ solver: practical point of view 3/3

Definition

Allow the solver to decide the satisfiability of a formula with:

- increasing number of constraints
- provided some "assumptions" are satisfied

The SAT ~~problem~~ solver: practical point of view 3/3

Definition

Allow the solver to decide the satisfiability of a formula with:

- increasing number of constraints
- provided some "assumptions" are satisfied

Example

$$C = \{s_1 \vee \neg a \vee b, s_1 \vee \neg b \vee c, s_2 \vee a, s_2 \vee \neg c\}$$

$$C_1 \equiv C \wedge \neg s_1 \wedge s_2$$

$$C_2 \equiv C \wedge \neg s_1 \wedge \neg s_2$$

The SAT ~~problem~~ solver: practical point of view 3/3

Definition

Allow the solver to decide the satisfiability of a formula with:

- increasing number of constraints
- provided some "assumptions" are satisfied

Example

$$C = \{s_1 \vee \neg a \vee b, s_1 \vee \neg b \vee c, s_2 \vee a, s_2 \vee \neg c\}$$

$$C_1 \equiv C \wedge \neg s_1 \wedge s_2$$

$$C_2 \equiv C \wedge \neg s_1 \wedge \neg s_2$$

The solver is considered as a stateful system: as long as the constraints are satisfiable, learn clauses can be kept: incremental SAT oracle

A short history of SAT in one slide

- 60's First algorithms [DP60,DLL62,Robinson65]
  DP + DLL = DPLL
- 70's SAT is NP-complete [Cook71]
  SAT is one of the simplest hard problems in CS
- 90's Applications, Solvers, Competitions
  Planning as Satisfiability, Alloy, Bounded Model Checking
  Solvers available in source (GRASP, SATO, RELSAT, WALKSAT, and many more)
  Padderborn (92), DIMACS@Rutgers (93) and Beijing (96)
- 00's Revolution, Competitions, Adoption
  Chaff (2001) and Minisat (2003)
  Yearly competition or race
  SAT increasingly used both in academia and industry
- 10's NP and Beyond NP
  MAXSAT, QBF
  Largest mathematical proof (Pythagorean triples, 200TB)

FUN FACT: comparing computer vs human execution time

In the present paper, a uniform proof procedure for quantification theory is given which is feasible for use with some rather complicated formulas and which does not ordinarily lead to exponentiation. The superiority of the present procedure over those previously available is indicated in part by the fact that a formula on which Gilmore's routine for the IBM 704 causes the machine to compute for 21 minutes without obtaining a result was worked successfully by hand computation using the present method in 30 minutes [Davis and Putnam, 1960].

The well-formed formula (...) which was beyond the scope of Gilmore's program was proved in under two minutes with the present program [Davis et al., 1962]

Outline

SAT, SAT Oracle, SAT Solver

Importance of the interaction with the solver

Importance of the encodings

When encodings are too large

How to solve ~~MaxSat~~ MinUnsat with SAT?

- Associate to each clause a weight (penalty) w_i taken into account if the clause is violated: Soft clauses S.
- Special weight (∞) for clauses that cannot be violated: hard clauses H

Definition (Partial Weighted MaxSat)

Find a model M of H that minimizes $weight(M, S)$ such that:

- $weight(M, (c_i, w_i)) = 0$ if M satisfies c_i, else w_i.
- $weight(M, S) = \sum_{wc \in S} weight(M, wc)$

Simply called MaxSAT if $k = 1$ and $H = \emptyset$

How to solve ~~MaxSat~~ MinUnsat with SAT?

- Associate to each clause a weight (penalty) w_i taken into account if the clause is violated: Soft clauses S.
  $(\neg a \vee b, 6) \wedge (\neg b \vee c, 8)$
- Special weight (∞) for clauses that cannot be violated: hard clauses H

Definition (Partial Weighted MaxSat)

Find a model M of H that minimizes $weight(M, S)$ such that:

- $weight(M, (c_i, w_i)) = 0$ if M satisfies c_i, else w_i.
- $weight(M, S) = \sum_{wc \in S} weight(M, wc)$

Simply called MaxSAT if $k = 1$ and $H = \emptyset$

How to solve ~~MaxSat~~ MinUnsat with SAT?

- Associate to each clause a weight (penalty) w_i taken into account if the clause is violated: Soft clauses S.
  $(\neg a \vee b, 6) \wedge (\neg b \vee c, 8)$
- Special weight (∞) for clauses that cannot be violated: hard clauses H
  $(a, \infty) \wedge (\neg c, \infty)$

Definition (Partial Weighted MaxSat)

Find a model M of H that minimizes $weight(M, S)$ such that:

- $weight(M, (c_i, w_i)) = 0$ if M satisfies c_i, else w_i.
- $weight(M, S) = \sum_{wc \in S} weight(M, wc)$

Simply called MaxSAT if $k = 1$ and $H = \emptyset$

How to solve ~~MaxSat~~ MinUnsat with SAT?

- Associate to each clause a weight (penalty) w_i taken into account if the clause is violated: Soft clauses S.
  $(\neg a \vee b, 6) \wedge (\neg b \vee c, 8)$
- Special weight (∞) for clauses that cannot be violated: hard clauses H $(a, \infty) \wedge (\neg c, \infty)$

Definition (Partial Weighted MaxSat)

Find a model M of H that minimizes $weight(M, S)$ such that:

- $weight(M, (c_i, w_i)) = 0$ if M satisfies c_i, else w_i.
- $weight(M, S) = \sum_{wc \in S} weight(M, wc)$ weight of $\{a, \neg b, \neg c\}$ is 6

Simply called MaxSAT if $k = 1$ and $H = \emptyset$

Linear Search for solving MaxSAT

x_6, x_2 $\quad \neg x_6, x_2$ $\quad \neg x_2, x_1$ $\quad \neg x_1$

$\neg x_6, x_8$ $\quad x_6, \neg x_8$ $\quad x_2, x_4$ $\quad \neg x_4, x_5$

x_7, x_5 $\quad \neg x_7, x_5$ $\quad \neg x_5, x_3$ $\quad \neg x_3$

Example CNF formula ($k = 1$ for each clause, not displayed)

Linear Search for solving MaxSAT

x_6, x_2, b_7 $\quad \neg x_6, x_2, b_8$ $\quad \neg x_2, x_1, b_1$ $\quad \neg x_1, b_2$

$\neg x_6, x_8, b_9$ $\quad x_6, \neg x_8, b_{10}$ $\quad x_2, x_4, b_3$ $\quad \neg x_4, x_5, b_4$

x_7, x_5, b_{11} $\quad \neg x_7, x_5, b_{12}$ $\quad \neg x_5, x_3, b_5$ $\quad \neg x_3, b_6$

Add selector or blocking variables b_i

Linear Search for solving MaxSAT

x_6, x_2, b_7 $\quad \neg x_6, x_2, b_8$ $\quad \neg x_2, x_1, b_1$ $\quad \neg x_1, b_2$

$\neg x_6, x_8, b_9$ $\quad x_6, \neg x_8, b_{10}$ $\quad x_2, x_4, b_3$ $\quad \neg x_4, x_5, b_4$

x_7, x_5, b_{11} $\quad \neg x_7, x_5, b_{12}$ $\quad \neg x_5, x_3, b_5$ $\quad \neg x_3, b_6$

Formula is SAT; eg model M contains

$b_1, \neg b_2, b_3, \neg b_4, b_5, \neg b_7, \neg b_8, \neg b_9, b_{10}, \neg b_{11}, b_{12}$

Linear Search for solving MaxSAT

x_6, x_2, b_7 $\quad \neg x_6, x_2, b_8$ $\quad \neg x_2, x_1, b_1$ $\quad \neg x_1, b_2$

$\neg x_6, x_8, b_9$ $\quad x_6, \neg x_8, b_{10}$ $\quad x_2, x_4, b_3$ $\quad \neg x_4, x_5, b_4$

x_7, x_5, b_{11} $\quad \neg x_7, x_5, b_{12}$ $\quad \neg x_5, x_3, b_5$ $\quad \neg x_3, b_6$

$\sum_{i=1}^{12} b_i < 5$

Bound the number of constraints to be relaxed: $|M \cap B| = 5$

Linear Search for solving MaxSAT

x_6, x_2, b_7 $\quad \neg x_6, x_2, b_8$ $\quad \neg x_2, x_1, b_1$ $\quad \neg x_1, b_2$

$\neg x_6, x_8, b_9$ $\quad x_6, \neg x_8, b_{10}$ $\quad x_2, x_4, b_3$ $\quad \neg x_4, x_5, b_4$

x_7, x_5, b_{11} $\quad \neg x_7, x_5, b_{12}$ $\quad \neg x_5, x_3, b_5$ $\quad \neg x_3, b_6$

$\sum_{i=1}^{12} b_i < 5$

Formula is (again) SAT; eg model contains

$b_1, \neg b_2, \neg b_3, \neg b_4, \neg b_5, \neg b_7, \neg b_8, \neg b_9, \neg b_{10}, \neg b_{11}, b_{12}$

Linear Search for solving MaxSAT

x_6, x_2, b_7 $\quad \neg x_6, x_2, b_8$ $\quad \neg x_2, x_1, b_1$ $\quad \neg x_1, b_2$

$\neg x_6, x_8, b_9$ $\quad x_6, \neg x_8, b_{10}$ $\quad x_2, x_4, b_3$ $\quad \neg x_4, x_5, b_4$

x_7, x_5, b_{11} $\quad \neg x_7, x_5, b_{12}$ $\quad \neg x_5, x_3, b_5$ $\quad \neg x_3, b_6$

$\sum_{i=1}^{12} b_i < 2$

Bound the number of constraints to be relaxed $|M \cap B| = 2$

Linear Search for solving MaxSAT

x_6, x_2, b_7	$\neg x_6, x_2, b_8$	$\neg x_2, x_1, b_1$	$\neg x_1, b_2$
$\neg x_6, x_8, b_9$	$x_6, \neg x_8, b_{10}$	x_2, x_4, b_3	$\neg x_4, x_5, b_4$
x_7, x_5, b_{11}	$\neg x_7, x_5, b_{12}$	$\neg x_5, x_3, b_5$	$\neg x_3, b_6$
$\sum_{i=1}^{12} b_i < 2$			

Instance is now UNSAT

Linear Search for solving MaxSAT

x_6, x_2, b_7	$\neg x_6, x_2, b_8$	$\neg x_2, x_1, b_1$	$\neg x_1, b_2$
$\neg x_6, x_8, b_9$	$x_6, \neg x_8, b_{10}$	x_2, x_4, b_3	$\neg x_4, x_5, b_4$
x_7, x_5, b_{11}	$\neg x_7, x_5, b_{12}$	$\neg x_5, x_3, b_5$	$\neg x_3, b_6$
$\sum_{i=1}^{12} b_i < 2$			

MaxSAT solution is $|\varphi| - |M \cap B| = 12 - 2 = 10$

Note that ...

- No initial upper or lower bounds: the first model provides a first upper bound.
- In practice, the objective function can be used to guide the search
- The procedure follows a SAT, SAT, SAT, SAT, ..., UNSAT pattern with linear search
- Binary search is possible but:
  - SAT answer is usually faster than UNSAT
  - the solver must be reset in case on unsatisfiability
- In lucky case, two calls to the SAT solver are sufficient (one SAT + one UNSAT).
- Used in Sat4j since 2006, was state-of-the-art in 2009
- Main issue: how to represent the bound constraint?

From Unsat Core computation to MaxSat: MSU

Z. Fu and S. Malik, On solving the partial MAX-SAT problem, in International Conference on Theory and Applications of Satisfiability Testing, August 2006, pp. 252-265.

Other SAT-based approaches in practical Max Sat solving rely on unsat core computation [Fu and Malik 2006]:

- Compute one unsat core C' of the formula C
- Relax it by replacing C' by $\{ r_i \vee C_i | C_i \in C' \}$
- Add the constraint $\sum r_i \leq 1$ to C
- Repeat until the formula is satisfiable
- If $MinUnsat(C) = k$, requires $k + 1$ loops.

Many improvement since then (PM1, PM2, MsUncore, etc): works for Weighted Max Sat, reduction of the number of relaxation variables, etc.

Fu&Malik's Algorithm: msu1.0

x_6, x_2 $\quad \neg x_6, x_2$ $\quad \neg x_2, x_1$ $\quad \neg x_1$

$\neg x_6, x_8$ $\quad x_6, \neg x_8$ $\quad x_2, x_4$ $\quad \neg x_4, x_5$

x_7, x_5 $\quad \neg x_7, x_5$ $\quad \neg x_5, x_3$ $\quad \neg x_3$

Example CNF formula

Fu&Malik's Algorithm: msu1.0

x_6, x_2	$\neg x_6, x_2$	$\neg x_2, x_1$	$\neg x_1$
$\neg x_6, x_8$	$x_6, \neg x_8$	x_2, x_4	$\neg x_4, x_5$
x_7, x_5	$\neg x_7, x_5$	$\neg x_5, x_3$	$\neg x_3$

Formula is UNSAT; Get unsat core

Fu&Malik's Algorithm: msu1.0

x_6, x_2 $\quad \neg x_6, x_2$ $\quad \neg x_2, x_1, b_1$ $\quad \neg x_1, b_2$

$\neg x_6, x_8$ $\quad x_6, \neg x_8$ $\quad x_2, x_4, b_3$ $\quad \neg x_4, x_5, b_4$

x_7, x_5 $\quad \neg x_7, x_5$ $\quad \neg x_5, x_3, b_5$ $\quad \neg x_3, b_6$

$\sum_{i=1}^{6} b_i \leq 1$

Add blocking variables and AtMost1 constraint

Fu&Malik's Algorithm: msu1.0

x_6, x_2	$\neg x_6, x_2$	$\neg x_2, x_1, b_1$	$\neg x_1, b_2$
$\neg x_6, x_8$	$x_6, \neg x_8$	x_2, x_4, b_3	$\neg x_4, x_5, b_4$
x_7, x_5	$\neg x_7, x_5$	$\neg x_5, x_3, b_5$	$\neg x_3, b_6$
$\sum_{i=1}^{6} b_i \leq 1$			

Formula is (again) UNSAT; Get unsat core

Fu&Malik's Algorithm: msu1.0

x_6, x_2, b_7 $\quad \neg x_6, x_2, b_8$ $\quad \neg x_2, x_1, b_1, b_9$ $\quad \neg x_1, b_2, b_{10}$

$\neg x_6, x_8$ $\quad x_6, \neg x_8$ $\quad x_2, x_4, b_3$ $\quad \neg x_4, x_5, b_4$

x_7, x_5, b_{11} $\quad \neg x_7, x_5, b_{12}$ $\quad \neg x_5, x_3, b_5, b_{13}$ $\quad \neg x_3, b_6, b_{14}$

$\sum_{i=1}^{6} b_i \leq 1$ $\quad \sum_{i=7}^{14} b_i \leq 1$

Add new blocking variables and AtMost1 constraint

Fu&Malik's Algorithm: msu1.0

x_6, x_2, b_7 $\quad \neg x_6, x_2, b_8$ $\quad \neg x_2, x_1, b_1, b_9$ $\quad \neg x_1, b_2, b_{10}$

$\neg x_6, x_8$ $\quad x_6, \neg x_8$ $\quad x_2, x_4, b_3$ $\quad \neg x_4, x_5, b_4$

x_7, x_5, b_{11} $\quad \neg x_7, x_5, b_{12}$ $\quad \neg x_5, x_3, b_5, b_{13}$ $\quad \neg x_3, b_6, b_{14}$

$\sum_{i=1}^{6} b_i \leq 1$ $\quad \sum_{i=7}^{14} b_i \leq 1$

Instance is now SAT

Fu&Malik's Algorithm: msu1.0

x_6, x_2, b_7 $\quad \neg x_6, x_2, b_8$ $\quad \neg x_2, x_1, b_1, b_9$ $\quad \neg x_1, b_2, b_{10}$

$\neg x_6, x_8$ $\quad x_6, \neg x_8$ $\quad x_2, x_4, b_3$ $\quad \neg x_4, x_5, b_4$

x_7, x_5, b_{11} $\quad \neg x_7, x_5, b_{12}$ $\quad \neg x_5, x_3, b_5, b_{13}$ $\quad \neg x_3, b_6, b_{14}$

$\sum_{i=1}^{6} b_i \leq 1$ $\quad \sum_{i=7}^{14} b_i \leq 1$

MaxSAT solution is $|\varphi| - \mathcal{I} = 12 - 2 = 10$

Note that ...

- Unsat core may not be minimal
- Nice property: if k constraints must be relaxed, then the procedure requires exactly $k + 1$ calls to the SAT solver.
- How to represent the cardinality constraints?

MaxHS: SAT and MIP solver interplay

Jessica Davies, Fahiem Bacchus: Solving MAXSAT by Solving a Sequence of Simpler SAT Instances. CP 2011: 225-239

- Core guided MAXSAT solver can be seen as a two step procedure:
  - Discover UNSAT cores of the formula
  - Stop as soon as one minimal Hitting Set of the cores satisfies the formula
- The size of the HS provides the number of constraints to relax
- May require to enumerate all MUS of a formula
- Or less if lucky

MaxHS principle

x_6, x_2, b_7 $\quad \neg x_6, x_2, b_8$ $\quad \neg x_2, x_1, b_1$ $\quad \neg x_1, b_2$

$\neg x_6, x_8, b_9$ $\quad x_6, \neg x_8, b_{10}$ $\quad x_2, x_4, b_3$ $\quad \neg x_4, x_5, b_4$

x_7, x_5, b_{11} $\quad \neg x_7, x_5, b_{12}$ $\quad \neg x_5, x_3, b_5$ $\quad \neg x_3, b_6$

$Cores = \{\}$ $\qquad$ $HS = \emptyset$

MaxHS principle

x_6, x_2, b_7 $\quad \neg x_6, x_2, b_8$ $\quad \neg x_2, x_1, b_1$ $\quad \neg x_1, b_2$

$\neg x_6, x_8, b_9$ $\quad x_6, \neg x_8, b_{10}$ $\quad x_2, x_4, b_3$ $\quad \neg x_4, x_5, b_4$

x_7, x_5, b_{11} $\quad \neg x_7, x_5, b_{12}$ $\quad \neg x_5, x_3, b_5$ $\quad \neg x_3, b_6$

$\{\{b_1, b_2, b_3, b_4, b_5, b_6\}\}$

$HS = \{b_4\}$

MaxHS principle

x_6, x_2, b_7	$\neg x_6, x_2, b_8$	$\neg x_2, x_1, b_1$	$\neg x_1, b_2$
$\neg x_6, x_8, b_9$	$x_6, \neg x_8, b_{10}$	x_2, x_4, b_3	$\neg x_4, x_5, b_4$
x_7, x_5, b_{11}	$\neg x_7, x_5, b_{12}$	$\neg x_5, x_3, b_5$	$\neg x_3, b_6$

$\{\{b_1, b_2, b_3, b_4, b_5, b_6\}, \{b_1, b_2, b_7, b_8\}\}$

$HS = \{b_1\}$

MaxHS principle

x_6, x_2, b_7	$\neg x_6, x_2, b_8$	$\neg x_2, x_1, b_1$	$\neg x_1, b_2$
$\neg x_6, x_8, b_9$	$x_6, \neg x_8, b_{10}$	x_2, x_4, b_3	$\neg x_4, x_5, b_4$
x_7, x_5, b_{11}	$\neg x_7, x_5, b_{12}$	$\neg x_5, x_3, b_5$	$\neg x_3, b_6$

$\{\{b_1, b_2, b_3, b_4, b_5, b_6\}, \{b_1, b_2, b_7, b_8\}, \{b_{11}, b_{12}, b_5, b_6\}\}$
$HS = \{b_2, b_5\}$

MaxHS principle

x_6, x_2, b_7	$\neg x_6, x_2, b_8$	$\neg x_2, x_1, b_1$	$\neg x_1, b_2$
$\neg x_6, x_8, b_9$	$x_6, \neg x_8, b_{10}$	x_2, x_4, b_3	$\neg x_4, x_5, b_4$
x_7, x_5, b_{11}	$\neg x_7, x_5, b_{12}$	$\neg x_5, x_3, b_5$	$\neg x_3, b_6$

Instance is SAT. MaxSAT solution is $12 - |\{b_2, b_5\}| = 10$

3 ways to solve the same [optimization] problem

- Take advantage of SAT solvers feedback: model or core
- No single approach outperforms the others
- Core-guided and MaxHS work best currently on "application" benchmarks (not crafted ones)

Linear Search or Core-Guided approaches require encoding cardinality constraints in CNF (or use native support for such constraints as found in Sat4j)

Outline

SAT, SAT Oracle, SAT Solver

Importance of the interaction with the solver

Importance of the encodings

When encodings are too large

Quick question for the audience

How would you encode

$$x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} \leq 1$$

as a CNF?

Quick question for the audience

How would you encode

$$x_1 + x_2 + x_3 + x_4 + x_5 + x_6 + x_7 + x_8 + x_9 + x_{10} \leq 1$$

as a CNF?

$\neg x_1 \vee \neg x_2,\ \neg x_1 \vee \neg x_3,\ \neg x_1 \vee \neg x_4,\ \neg x_1 \vee \neg x_5, \neg x_1 \vee \neg x_6,$
$\neg x_1 \vee \neg x_7,\ \neg x_1 \vee \neg x_8,\ \neg x_1 \vee \neg x_9,\ \neg x_1 \vee \neg x_{10},$
$\neg x_2 \vee \neg x_3,\ \neg x_2 \vee \neg x_4,\ \neg x_2 \vee \neg x_5, \neg x_2 \vee \neg x_6,$
$\neg x_2 \vee \neg x_7,\ \neg x_2 \vee \neg x_8,\ \neg x_2 \vee \neg x_9,\ \neg x_2 \vee \neg x_{10},$
$\neg x_3 \vee \neg x_4,\ \neg x_3 \vee \neg x_5, \neg x_3 \vee \neg x_6,\ \neg x_3 \vee \neg x_7,\ \neg x_3 \vee \neg x_8,$
$\neg x_3 \vee \neg x_9,\ \neg x_3 \vee \neg x_{10},$
$\neg x_4 \vee \neg x_5,\ \neg x_4 \vee \neg x_6,\ \neg x_4 \vee \neg x_7,\ \neg x_4 \vee \neg x_8,\ \neg x_4 \vee \neg x_9,\ \neg x_4 \vee \neg x_{10}$
$\neg x_5 \vee \neg x_6,\ \neg x_5 \vee \neg x_7,\ \neg x_5 \vee \neg x_8,\ \neg x_5 \vee \neg x_9,\ \neg x_5 \vee \neg x_{10},$
$\neg x_6 \vee \neg x_7,\ \neg x_6 \vee \neg x_8,\ \neg x_6 \vee \neg x_9,\ \neg x_6 \vee \neg x_{10},$
$\neg x_7 \vee \neg x_8,\ \neg x_7 \vee \neg x_9,\ \neg x_7 \vee \neg x_{10},$
$\neg x_8 \vee \neg x_9,\ \neg x_8 \vee \neg x_{10},\ \neg x_8 \vee \neg x_{10}.$

Pairwise encoding, 45 binary clauses

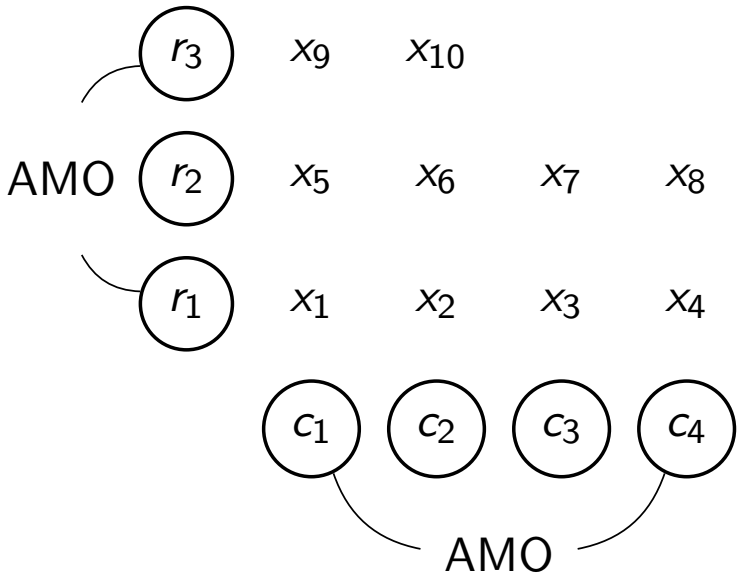
Some known encodings for cardinality constraints

Short list of known encodings :

- Pairwise encoding [Cook et al., 1987]
- Nested encoding
- Two product encoding [Chen, 2010]
- Sequential encoding [Sinz, 2005]
- Commander encoding [Frisch and Giannaros, 2010]
- Ladder encoding [Gent and Nightingale, 2004]
- Adder encoding [Eén and Sörensson, 2006]
- Cardinality Networks [Asín et al., 2009]
- ...

Two product encoding

Chen, J.-C.: A new sat encoding of the at-most-one constraint. In Proc. of the Tenth Int. Workshop of Constraint Modelling and Reformulation, 2010



Selection clauses

$$(\neg x_1 \vee r_1) \quad (\neg x_1 \vee c_1) \quad (\neg x_2 \vee r_1)$$
$$(\neg x_2 \vee c_2) \quad (\neg x_3 \vee r_1) \quad (\neg x_3 \vee c_3)$$
$$(\neg x_4 \vee r_1) \quad (\neg x_4 \vee c_4) \quad (\neg x_5 \vee r_2)$$
$$(\neg x_5 \vee c_1) \quad (\neg x_6 \vee r_2) \quad (\neg x_6 \vee c_2)$$
$$(\neg x_7 \vee r_2) \quad (\neg x_7 \vee c_3) \quad (\neg x_8 \vee r_2)$$
$$(\neg x_8 \vee c_4) \quad (\neg x_9 \vee r_3) \quad (\neg x_9 \vee c_1)$$
$$(\neg x_{10} \vee r_3) \quad (\neg x_{10} \vee c_2)$$

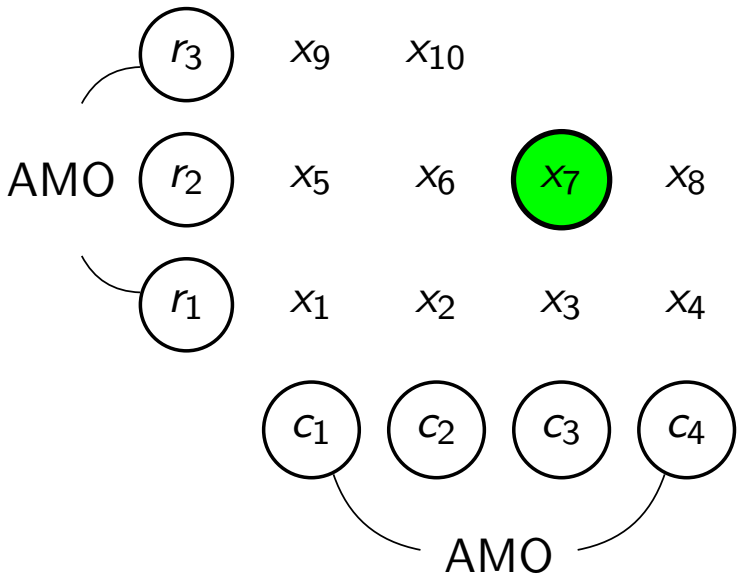
final AtMost-1

$$(\neg r_1 \vee \neg r_2) \quad (\neg r_1 \vee \neg r_3) \quad (\neg r_2 \vee \neg r_3)$$
$$(\neg c_1 \vee \neg c_2) \quad (\neg c_1 \vee \neg c_3) \quad (\neg c_1 \vee \neg c_4)$$
$$(\neg c_2 \vee \neg c_3) \quad (\neg c_2 \vee \neg c_4) \quad (\neg c_3 \vee \neg c_4)$$

encoding $\sum_{i=1}^{10} x_i \leq 1$

Two product encoding

Chen, J.-C.: A new sat encoding of the at-most-one constraint. In Proc. of the Tenth Int. Workshop of Constraint Modelling and Reformulation, 2010



Selection clauses

$$(\neg x_1 \vee r_1) \quad (\neg x_1 \vee c_1) \quad (\neg x_2 \vee r_1)$$
$$(\neg x_2 \vee c_2) \quad (\neg x_3 \vee r_1) \quad (\neg x_3 \vee c_3)$$
$$(\neg x_4 \vee r_1) \quad (\neg x_4 \vee c_4) \quad (\neg x_5 \vee r_2)$$
$$(\neg x_5 \vee c_1) \quad (\neg x_6 \vee r_2) \quad (\neg x_6 \vee c_2)$$
$$(\neg x_7 \vee r_2) \quad (\neg x_7 \vee c_3) \quad (\neg x_8 \vee r_2)$$
$$(\neg x_8 \vee c_4) \quad (\neg x_9 \vee r_3) \quad (\neg x_9 \vee c_1)$$
$$(\neg x_{10} \vee r_3) \quad (\neg x_{10} \vee c_2)$$

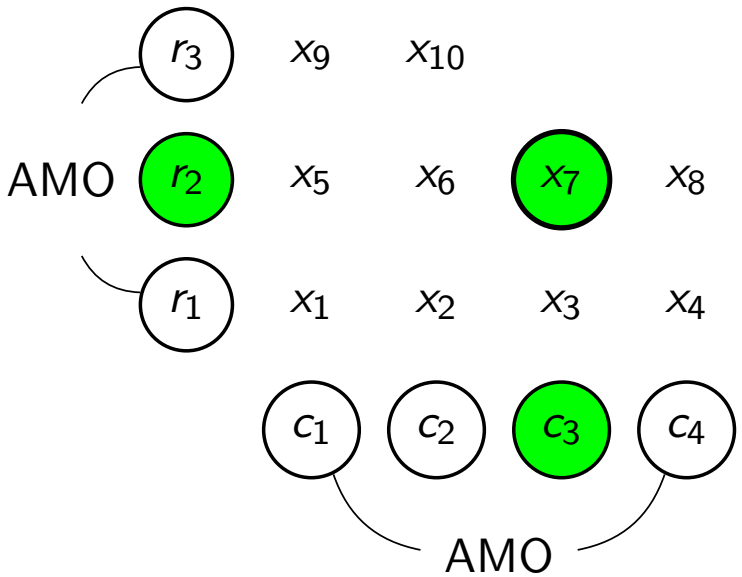
final AtMost-1

$$(\neg r_1 \vee \neg r_2) \quad (\neg r_1 \vee \neg r_3) \quad (\neg r_2 \vee \neg r_3)$$
$$(\neg c_1 \vee \neg c_2) \quad (\neg c_1 \vee \neg c_3) \quad (\neg c_1 \vee \neg c_4)$$
$$(\neg c_2 \vee \neg c_3) \quad (\neg c_2 \vee \neg c_4) \quad (\neg c_3 \vee \neg c_4)$$

encoding $\sum_{i=1}^{10} x_i \leq 1$

Two product encoding

Chen, J.-C.: A new sat encoding of the at-most-one constraint. In Proc. of the Tenth Int. Workshop of Constraint Modelling and Reformulation, 2010



Selection clauses

$(\neg x_1 \vee r_1)$ $(\neg x_1 \vee c_1)$ $(\neg x_2 \vee r_1)$
$(\neg x_2 \vee c_2)$ $(\neg x_3 \vee r_1)$ $(\neg x_3 \vee c_3)$
$(\neg x_4 \vee r_1)$ $(\neg x_4 \vee c_4)$ $(\neg x_5 \vee r_2)$
$(\neg x_5 \vee c_1)$ $(\neg x_6 \vee r_2)$ $(\neg x_6 \vee c_2)$
$(\neg x_7 \vee r_2)$ $(\neg x_7 \vee c_3)$ $(\neg x_8 \vee r_2)$
$(\neg x_8 \vee c_4)$ $(\neg x_9 \vee r_3)$ $(\neg x_9 \vee c_1)$
$(\neg x_{10} \vee r_3)$ $(\neg x_{10} \vee c_2)$

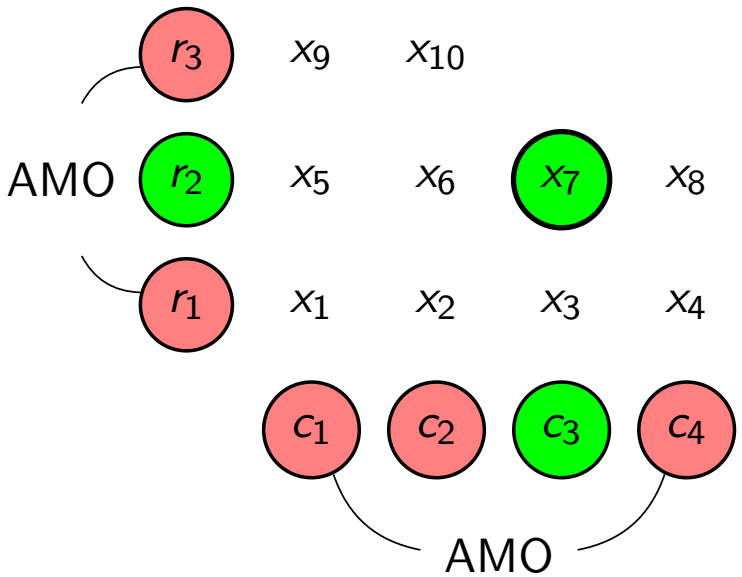
final AtMost-1

$(\neg r_1 \vee \neg r_2)$ $(\neg r_1 \vee \neg r_3)$ $(\neg r_2 \vee \neg r_3)$
$(\neg c_1 \vee \neg c_2)$ $(\neg c_1 \vee \neg c_3)$ $(\neg c_1 \vee \neg c_4)$
$(\neg c_2 \vee \neg c_3)$ $(\neg c_2 \vee \neg c_4)$ $(\neg c_3 \vee \neg c_4)$

encoding $\sum_{i=1}^{10} x_i \leq 1$

Two product encoding

Chen, J.-C.: A new sat encoding of the at-most-one constraint. In Proc. of the Tenth Int. Workshop of Constraint Modelling and Reformulation, 2010



Selection clauses

$(\neg x_1 \vee r_1)$ $(\neg x_1 \vee c_1)$ $(\neg x_2 \vee r_1)$
$(\neg x_2 \vee c_2)$ $(\neg x_3 \vee r_1)$ $(\neg x_3 \vee c_3)$
$(\neg x_4 \vee r_1)$ $(\neg x_4 \vee c_4)$ $(\neg x_5 \vee r_2)$
$(\neg x_5 \vee c_1)$ $(\neg x_6 \vee r_2)$ $(\neg x_6 \vee c_2)$
$(\neg x_7 \vee r_2)$ $(\neg x_7 \vee c_3)$ $(\neg x_8 \vee r_2)$
$(\neg x_8 \vee c_4)$ $(\neg x_9 \vee r_3)$ $(\neg x_9 \vee c_1)$
$(\neg x_{10} \vee r_3)$ $(\neg x_{10} \vee c_2)$

final AtMost-1

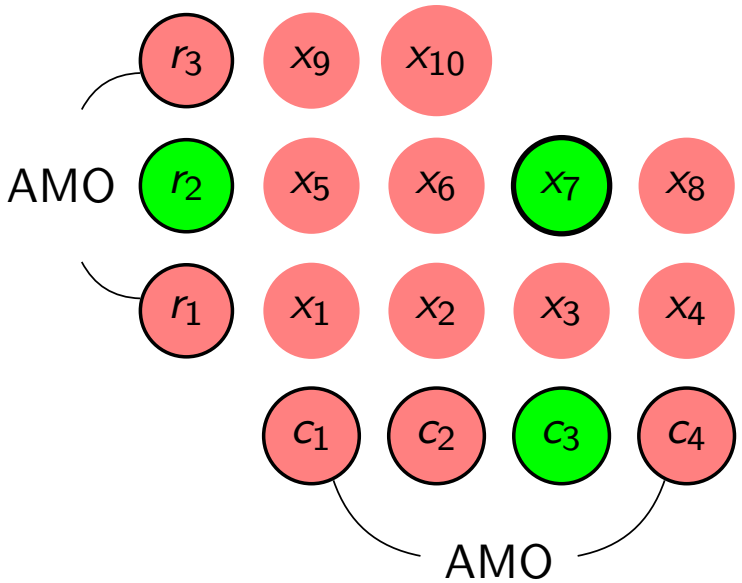
$(\neg r_1 \vee \neg r_2)$ $(\neg r_1 \vee \neg r_3)$ $(\neg r_2 \vee \neg r_3)$
$(\neg c_1 \vee \neg c_2)$ $(\neg c_1 \vee \neg c_3)$ $(\neg c_1 \vee \neg c_4)$
$(\neg c_2 \vee \neg c_3)$ $(\neg c_2 \vee \neg c_4)$ $(\neg c_3 \vee \neg c_4)$

encoding $\sum_{i=1}^{10} x_i \leq 1$

UNIVERSITÉ D'ARTOIS

Two product encoding

Chen, J.-C.: A new sat encoding of the at-most-one constraint. In Proc. of the Tenth Int. Workshop of Constraint Modelling and Reformulation, 2010



Selection clauses

$(\neg x_1 \vee r_1)$ $(\neg x_1 \vee c_1)$ $(\neg x_2 \vee r_1)$
$(\neg x_2 \vee c_2)$ $(\neg x_3 \vee r_1)$ $(\neg x_3 \vee c_3)$
$(\neg x_4 \vee r_1)$ $(\neg x_4 \vee c_4)$ $(\neg x_5 \vee r_2)$
$(\neg x_5 \vee c_1)$ $(\neg x_6 \vee r_2)$ $(\neg x_6 \vee c_2)$
$(\neg x_7 \vee r_2)$ $(\neg x_7 \vee c_3)$ $(\neg x_8 \vee r_2)$
$(\neg x_8 \vee c_4)$ $(\neg x_9 \vee r_3)$ $(\neg x_9 \vee c_1)$
$(\neg x_{10} \vee r_3)$ $(\neg x_{10} \vee c_2)$

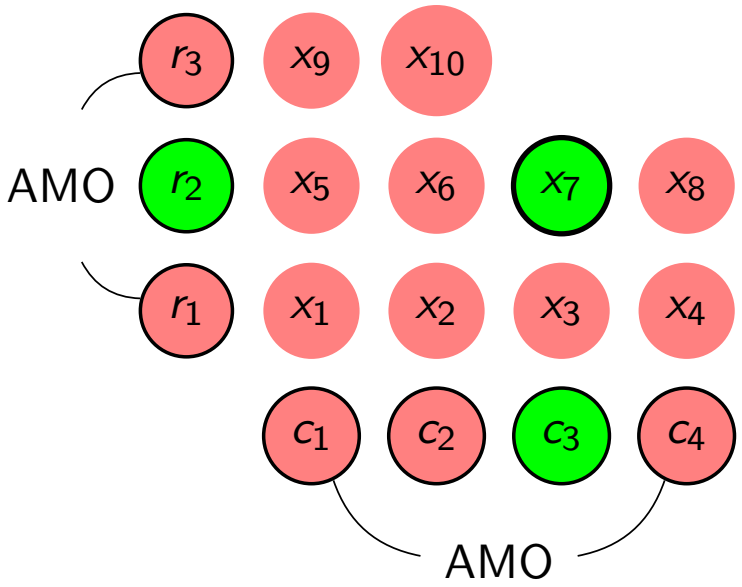
final AtMost-1

$(\neg r_1 \vee \neg r_2)$ $(\neg r_1 \vee \neg r_3)$ $(\neg r_2 \vee \neg r_3)$
$(\neg c_1 \vee \neg c_2)$ $(\neg c_1 \vee \neg c_3)$ $(\neg c_1 \vee \neg c_4)$
$(\neg c_2 \vee \neg c_3)$ $(\neg c_2 \vee \neg c_4)$ $(\neg c_3 \vee \neg c_4)$

encoding $\sum_{i=1}^{10} x_i \leq 1$

Two product encoding

Chen, J.-C.: A new sat encoding of the at-most-one constraint. In Proc. of the Tenth Int. Workshop of Constraint Modelling and Reformulation, 2010



Selection clauses

$(\neg x_1 \vee r_1)$ $(\neg x_1 \vee c_1)$ $(\neg x_2 \vee r_1)$
$(\neg x_2 \vee c_2)$ $(\neg x_3 \vee r_1)$ $(\neg x_3 \vee c_3)$
$(\neg x_4 \vee r_1)$ $(\neg x_4 \vee c_4)$ $(\neg x_5 \vee r_2)$
$(\neg x_5 \vee c_1)$ $(\neg x_6 \vee r_2)$ $(\neg x_6 \vee c_2)$
$(\neg x_7 \vee r_2)$ $(\neg x_7 \vee c_3)$ $(\neg x_8 \vee r_2)$
$(\neg x_8 \vee c_4)$ $(\neg x_9 \vee r_3)$ $(\neg x_9 \vee c_1)$
$(\neg x_{10} \vee r_3)$ $(\neg x_{10} \vee c_2)$

final AtMost-1

$(\neg r_1 \vee \neg r_2)$ $(\neg r_1 \vee \neg r_3)$ $(\neg r_2 \vee \neg r_3)$
$(\neg c_1 \vee \neg c_2)$ $(\neg c_1 \vee \neg c_3)$ $(\neg c_1 \vee \neg c_4)$
$(\neg c_2 \vee \neg c_3)$ $(\neg c_2 \vee \neg c_4)$ $(\neg c_3 \vee \neg c_4)$

encoding $\sum_{i=1}^{10} x_i \leq 1$

29 binary clauses

UNIVERSITÉ D'ARTOIS

Cardinality/Pseudo-Boolean constraints in CNF

- Translation in CNF without adding new variables often not an option
- Various encodings available, with different properties (number of additional variables, number of generated clauses, size of generated clauses, preserve or not arc consistency, etc.)
- Different solvers may behave differently on different encodings (e.g. because of specific management of binary clauses).
- For a survey of the effect of various encodings for MaxSat, see [Martins et al., 2012].

Cardinality/Pseudo-Boolean constraints ad hoc

Other option: do not encode! (our approach in Sat4j)

- Space efficient
- Can use extended reasoning: e.g. Generalized Resolution [Hooker, 1988]
- Cannot reuse off-the-shelf solver
- Requires to maintain the constraints in the solver

When the input is in CNF, retrieve cardinality constraints [Biere et al., 2014].

Outline

SAT, SAT Oracle, SAT Solver

Importance of the interaction with the solver

Importance of the encodings

When encodings are too large

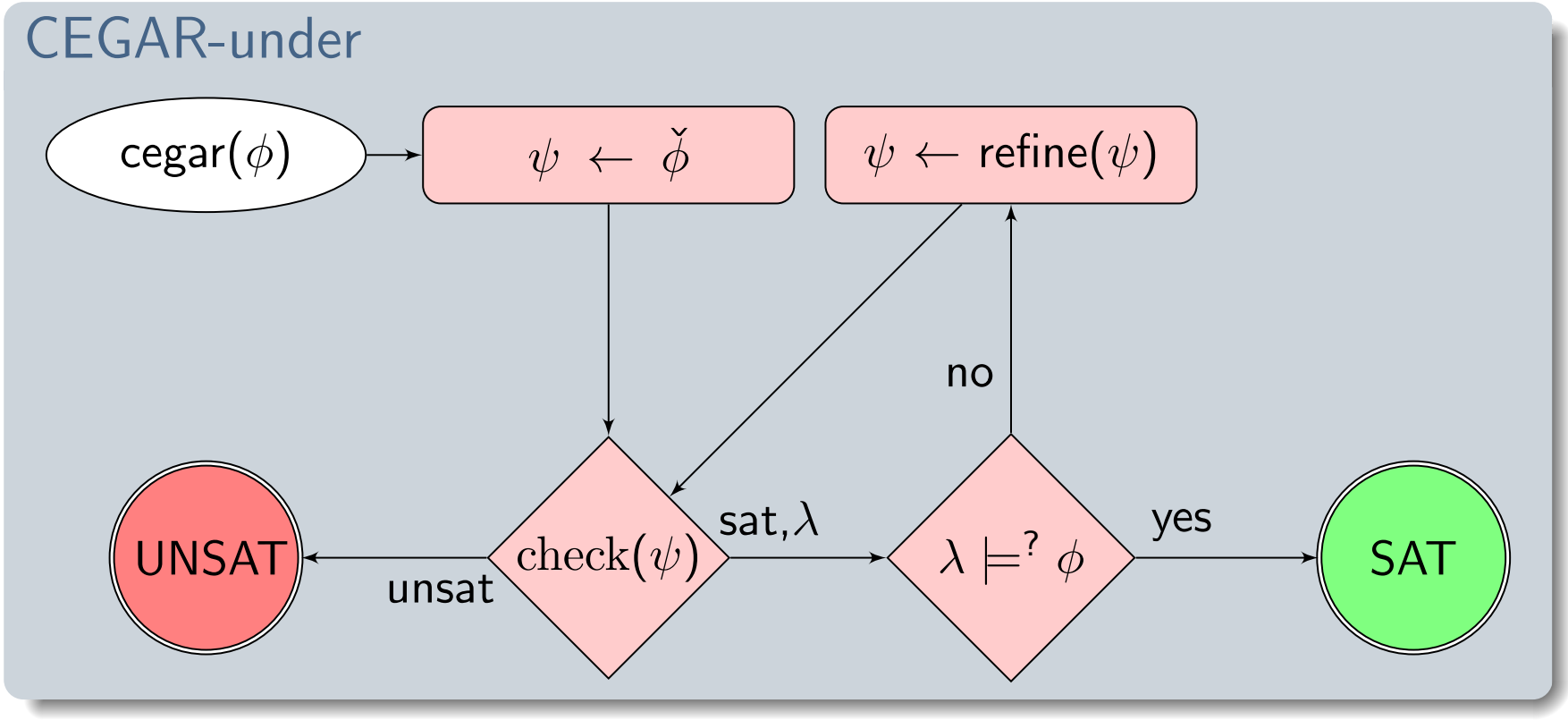
Sometimes the CNF encoding is just too large

- It is often the case that CNF encodings reach GB of space
- A popular technique in that case is to provide only parts of the constraints to the solver
- If the set of constraints is UNSAT, the original problem is UNSAT
- If the set of constraints is SAT, the model is checked against the original problem
- If the model is a solution of the original problem, the problem is solved (Lucky Outcome
- Else new constraints (clauses) are added to prevent such kind of spurious solution (Refinement)

Counter Example Guided Abstraction Refinement

CEGAR using under-abstractions

Edmund M. Clarke, Orna Grumberg, Somesh Jha, Yuan Lu, Helmut Veith: Counterexample-Guided Abstraction Refinement. CAV 2000: 154-169

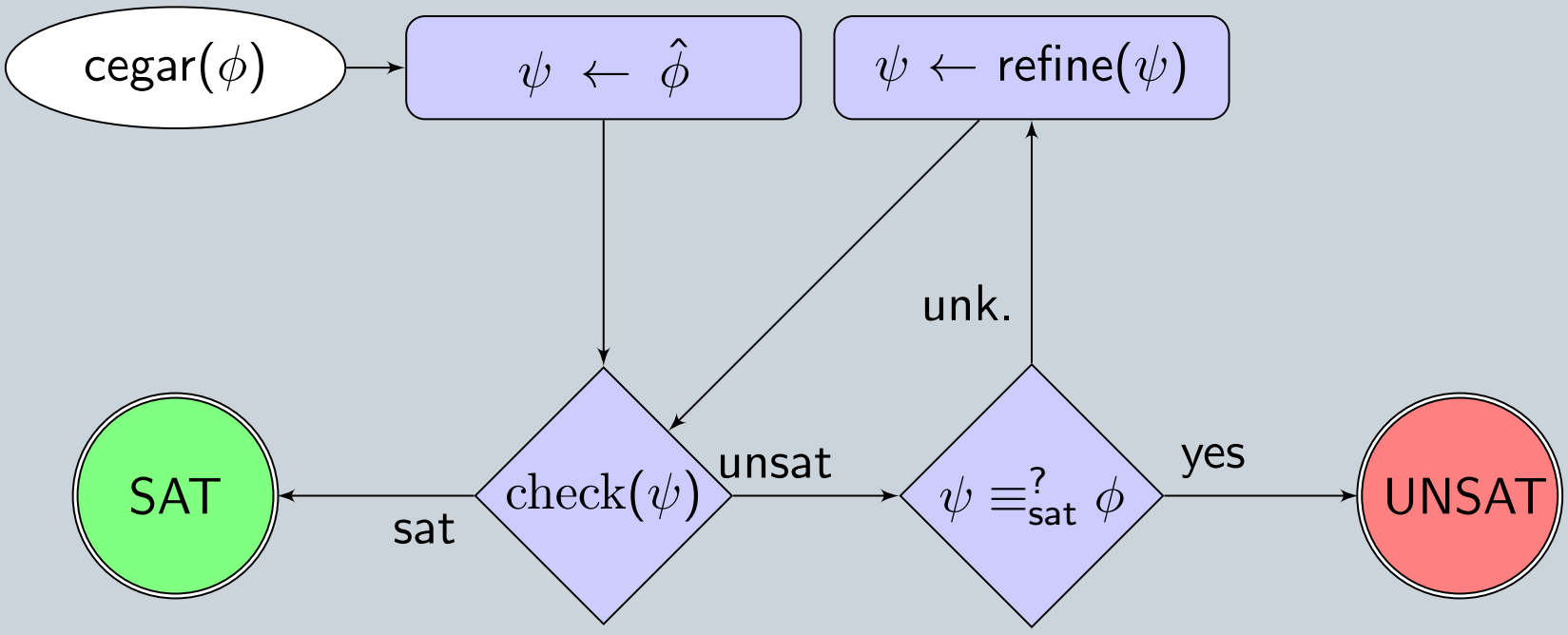


Example

Hamiltonian cycle problem

CEGAR using over-abstractions

CEGAR-over



Example

Planning problem, by increasing step by step the horizon; Bounded Model Checking

UNIVERSITÉ D'ARTOIS

CounterExample Guided Abstraction Refinement

Advantages

- If problem mainly satisfiable: CEGAR-over
- If problem mainly unsatisfiable: CEGAR-under
- When check improves, CEGAR improves
- Many applications already use CEGAR

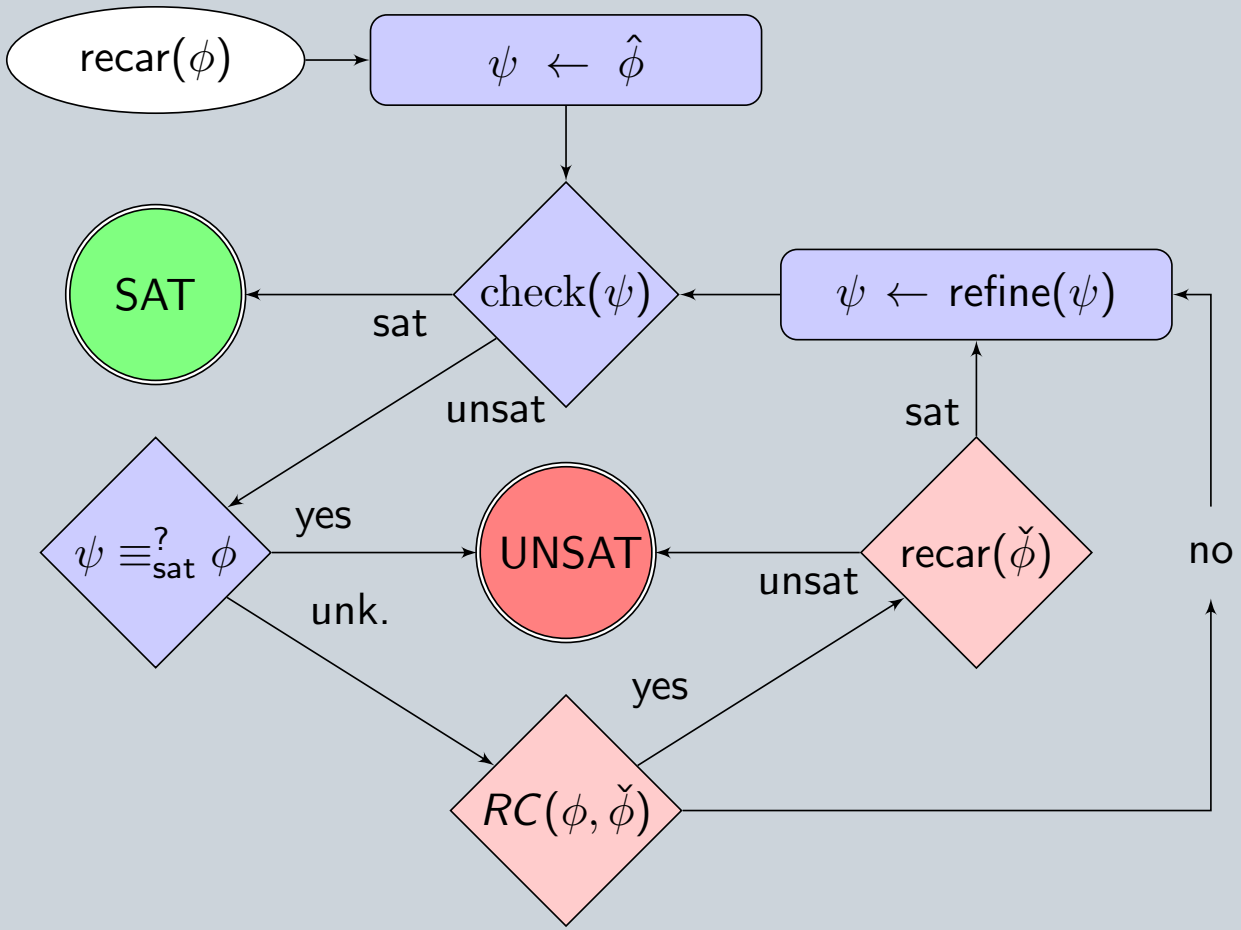
Drawbacks

- Not efficient when 50/50 chances of being SAT/UNSAT
- Not efficient when we need many refinement steps

Recursive Explore and Check Abstraction Refinement

Jean-Marie Lagniez, Daniel Le Berre, Tiago de Lima, Valentin Montmirail: A Recursive Shortcut for CEGAR: Application To The Modal Logic K Satisfiability Problem. IJCAI 2017: 674-680

RECAR



Recursive Explore and Check Abstraction Refinement

RECAR[Lagniez et al., 2017]

- 2 levels of abstractions
  - One at the Oracle level ($\mathrm{check}(\psi)$)
  - One at the Domain level (recursive call)
- Efficient even when 50/50 chance of being SAT/UNSAT
- When check improves, RECAR improves
- The return of the recursive call can reduce the number of refinements
- SAT and UNSAT shortcuts can be inverted if needed
- Totally generic, can change SAT solver by QBF/SMT/FO solver

Application to Modal Logic K

RECAR for Modal Logic K

- Modal Logic K is **PSPACE**-complete [Ladner, 1977, Halpern, 1995]
- What is Modal Logic K?
- How we over-approximate a formula ϕ?
- How we under-approximate a formula ϕ?
- Is it competitive against a CEGAR approach?
- Is it competitive against the state-of-the-art approaches?

Modal Logic

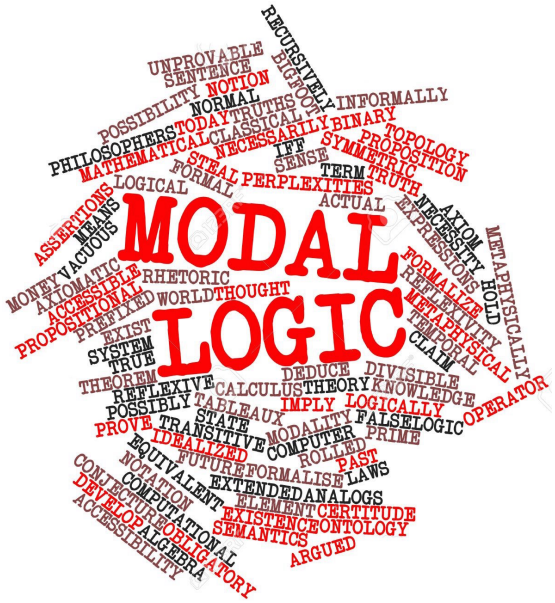
Modal Logic = Propositional Logic + $\Box$ and $\Diamond$

Modal Logic

- $\Box\phi$ means ϕ is necessarily true
- $\Diamond\phi$ means ϕ is possibly true

$$\Diamond\phi \leftrightarrow \neg\Box\neg\phi$$

$$\Box\phi \leftrightarrow \neg\Diamond\neg\phi$$



Satisfiability of Modal Logic formulas

✓ $\phi_1 = \Box(\bullet)$

✗ $\phi_2 = \Box\Diamond(\bullet)$

✓ $\phi_3 = \Diamond(\bullet \wedge \Diamond\neg\bullet)$

✓ $\phi_4 = (\bullet \vee \bullet \vee \bullet)$

✗ $\phi_5 = \Diamond\Diamond(\bullet \wedge \Box\neg\bullet)$

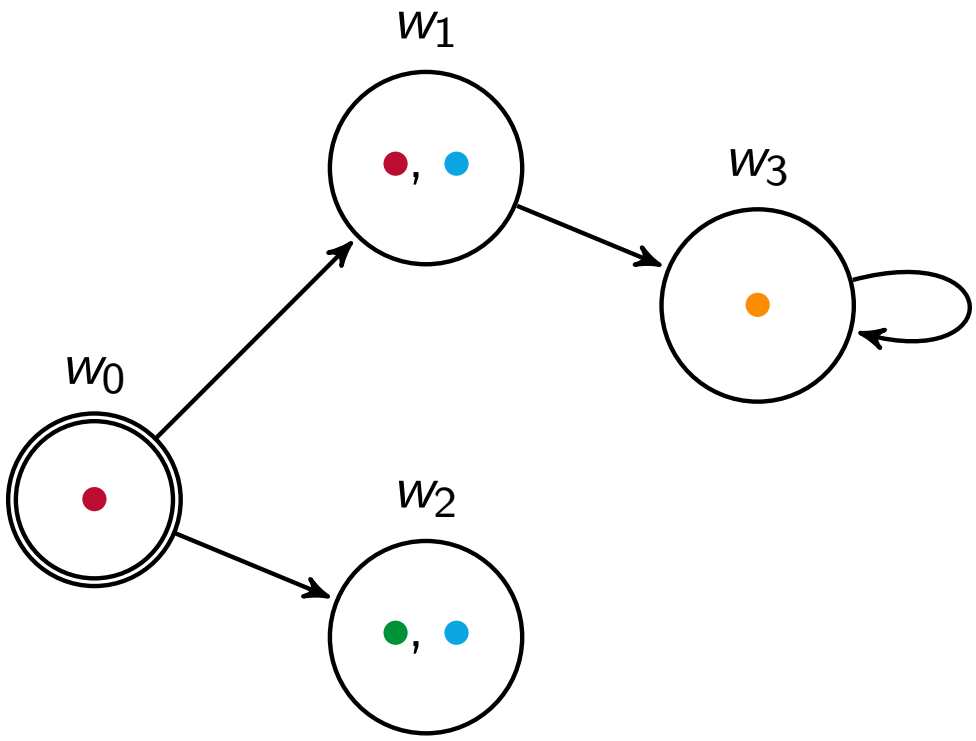


Figure: Example $\mathcal{K}$

Satisfiability of Modal Logic formulas

✓ $\phi_1 = \Box(\bullet)$

✗ $\phi_2 = \Box\Diamond(\bullet)$

✓ $\phi_3 = \Diamond(\bullet \wedge \Diamond\neg\bullet)$

✓ $\phi_4 = (\bullet \vee \bullet \vee \bullet)$

✓ $\phi_5 = \Diamond\Diamond(\bullet \wedge \Box\neg\bullet)$

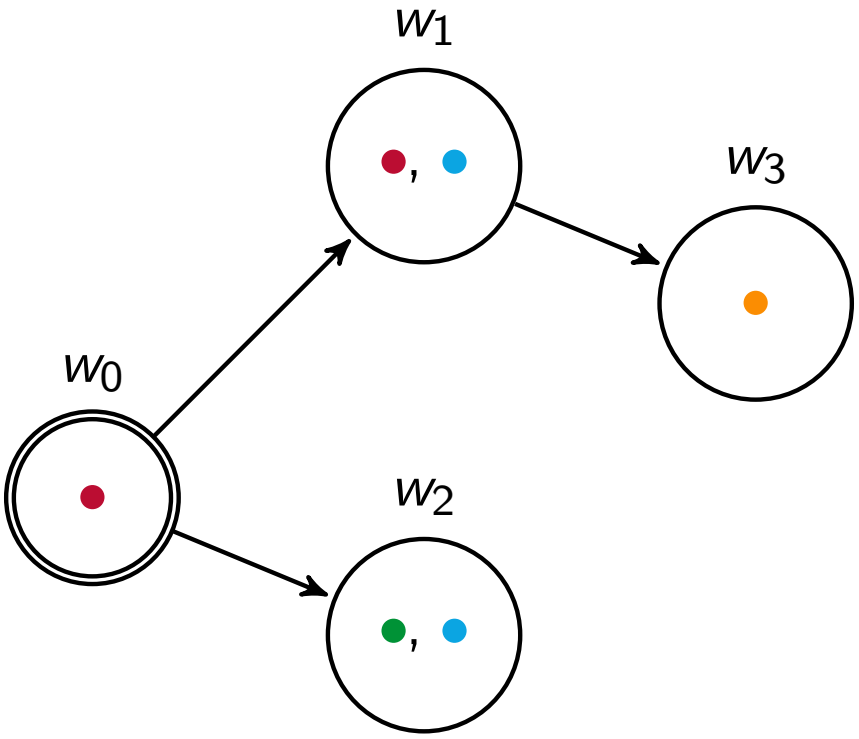
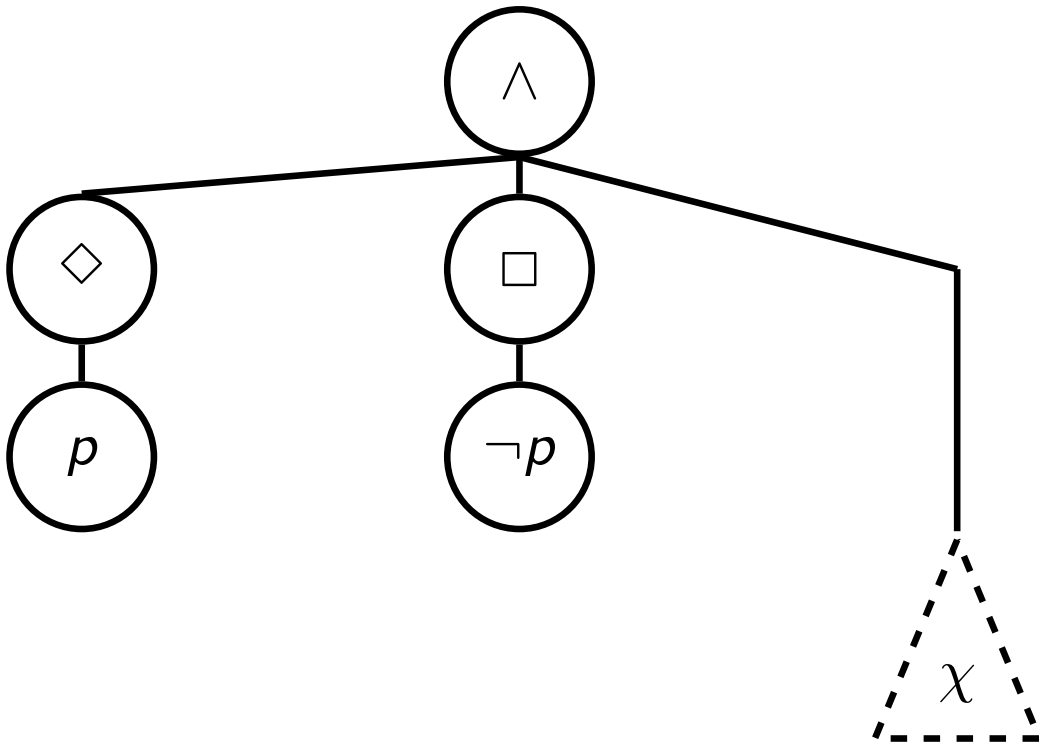


Figure: Example $\mathcal{K}$

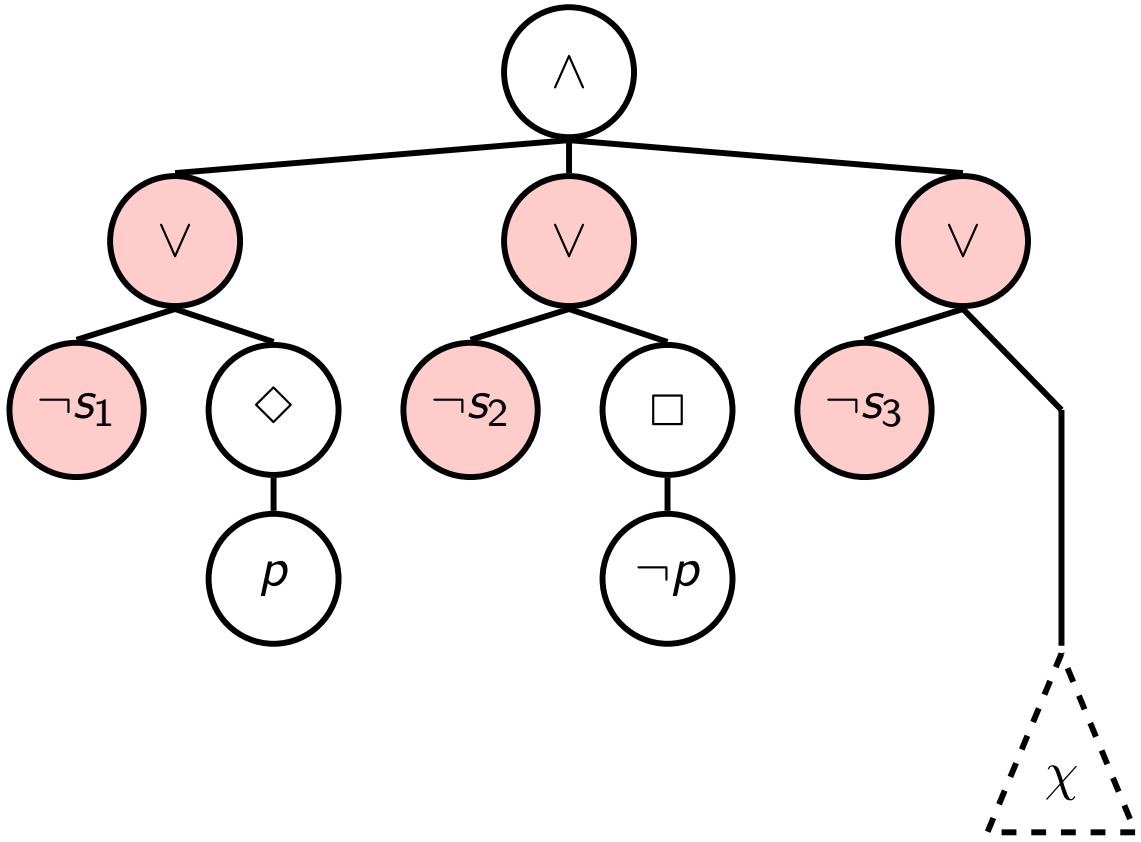
MoSaiC: Under-Approximation (modal logic level)

Suppose we want to solve the formula below, with χ huge but satisfiable...



Worst case for CEGAR using an over approximation, i.e. unrolling the Kripke structure

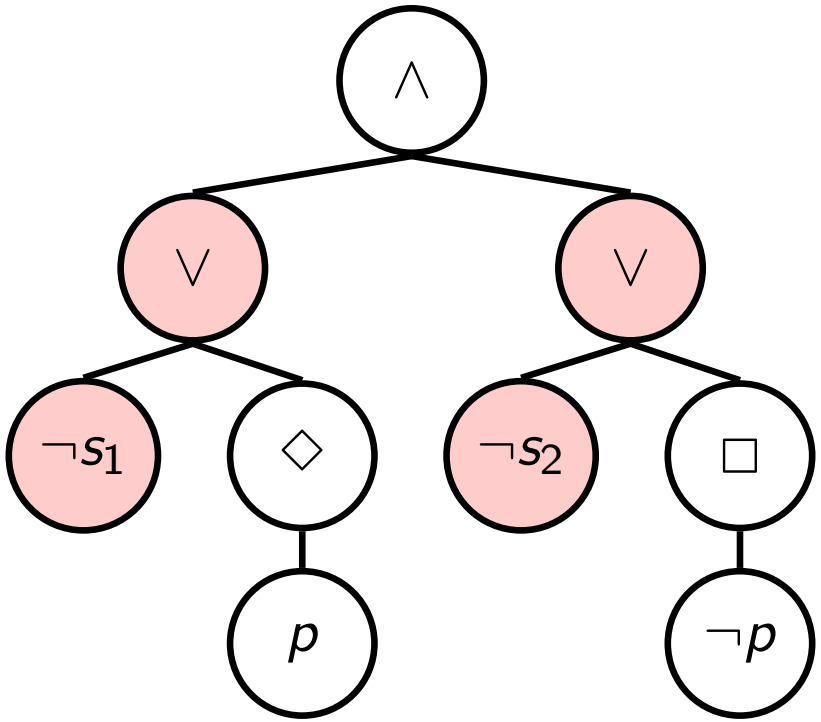
MoSaiC: Under-Approximation (modal logic level)



Modern SAT solvers returns 'the reason' why a formula with n worlds is unsatisfiable ($core = \{s_1, s_2\}$)

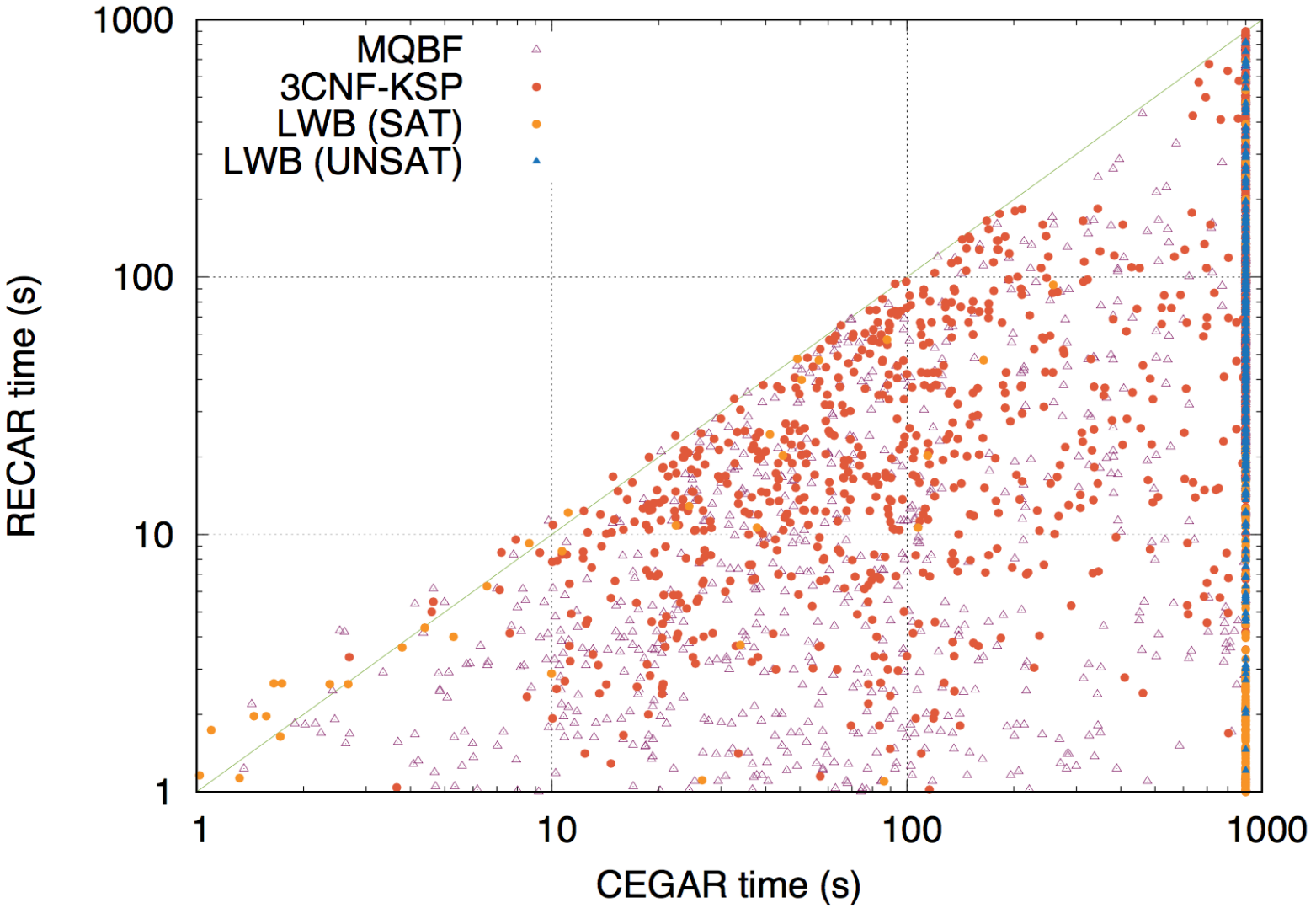
MoSaiC: Under-Approximation (modal logic level)

We want to cut what is not part of the 'unsatisfiability' ($s_i \notin core$)



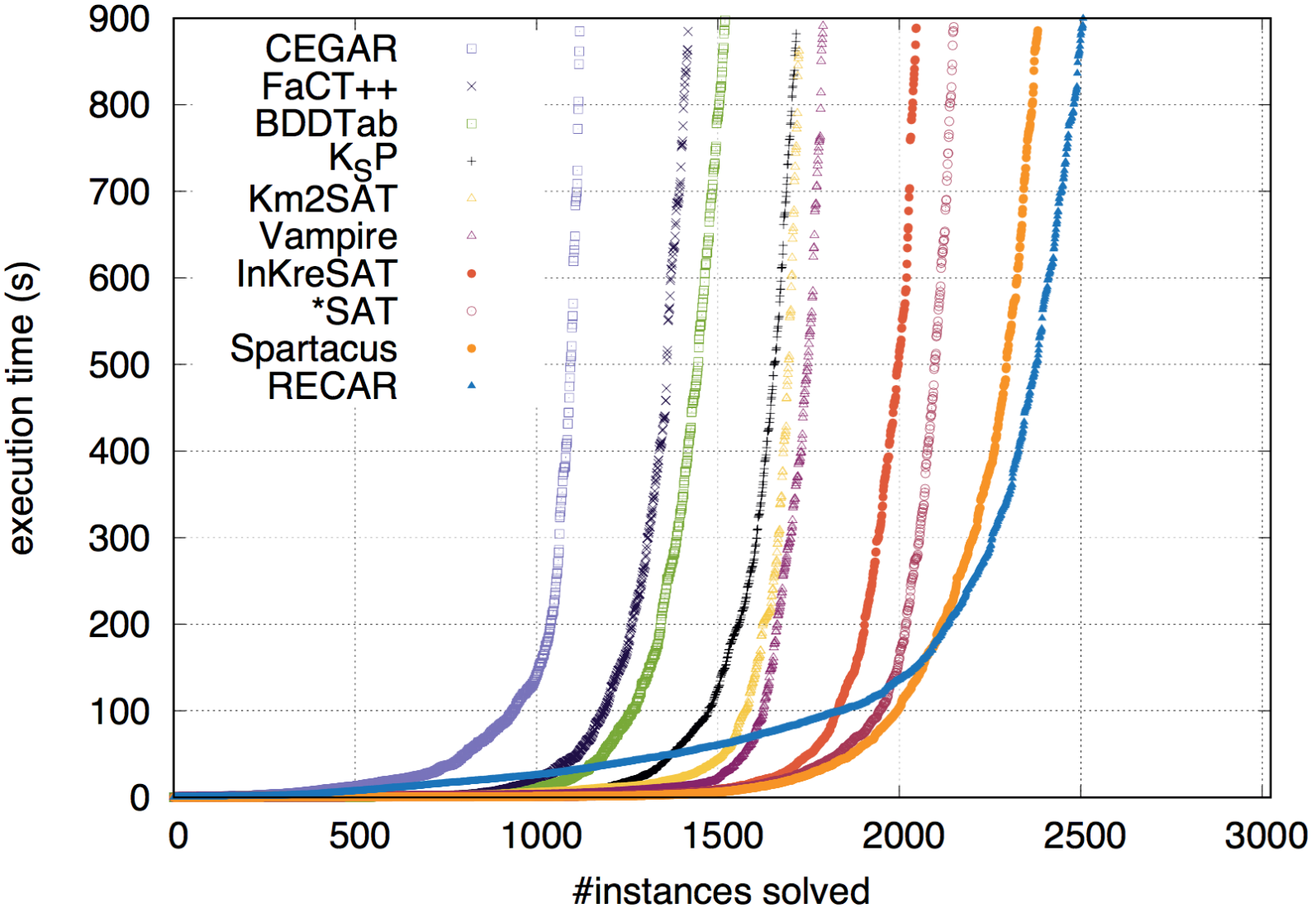
We just create $\check{\phi}$ smaller than ϕ and easier to solve.
The function *RC* from RECAR just says here: did we cut something ?

MoSaiC: RECAR for Modal Logic K



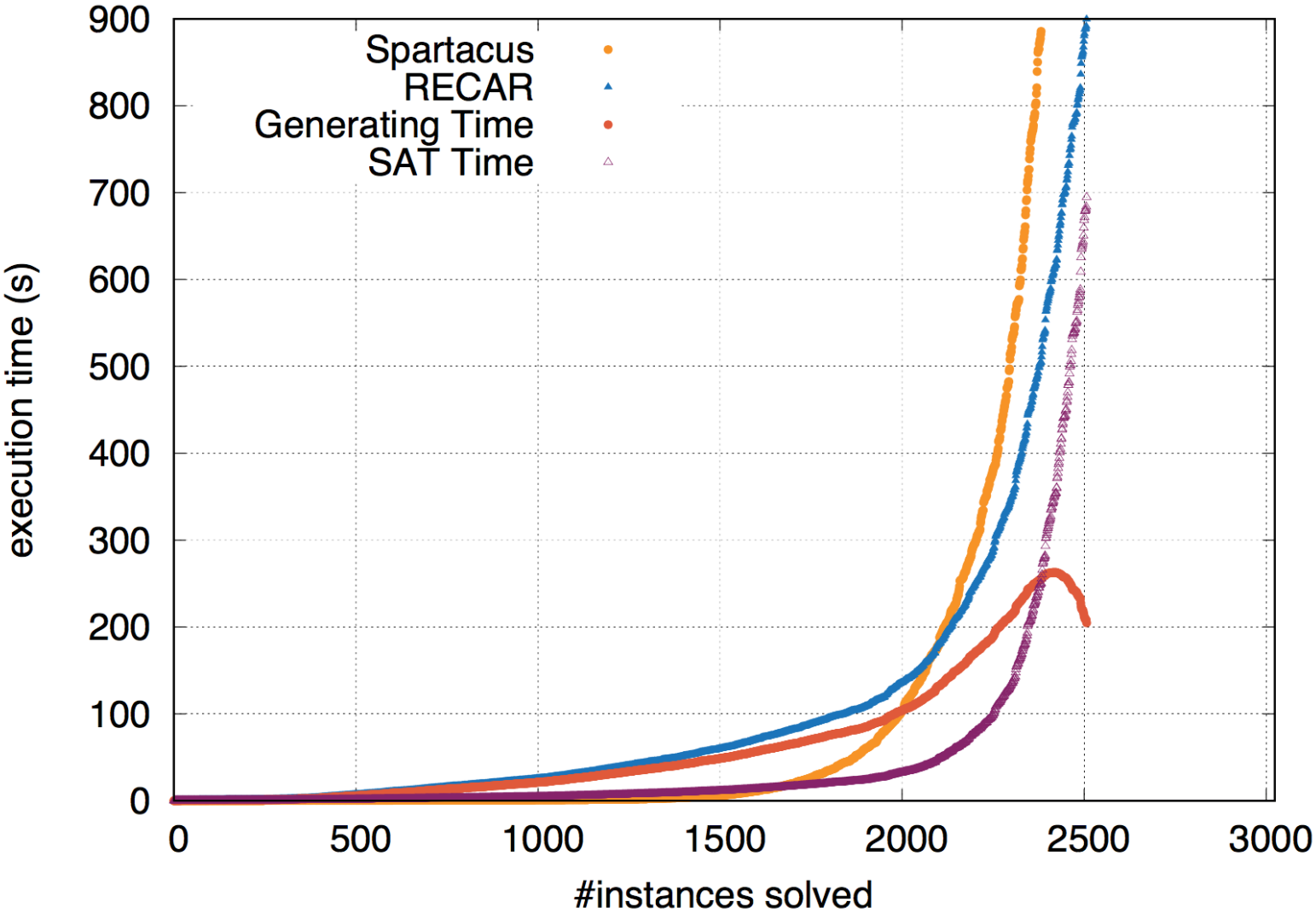
UNIVERSITÉ D'ARTOIS

MoSaiC: RECAR for Modal Logic K

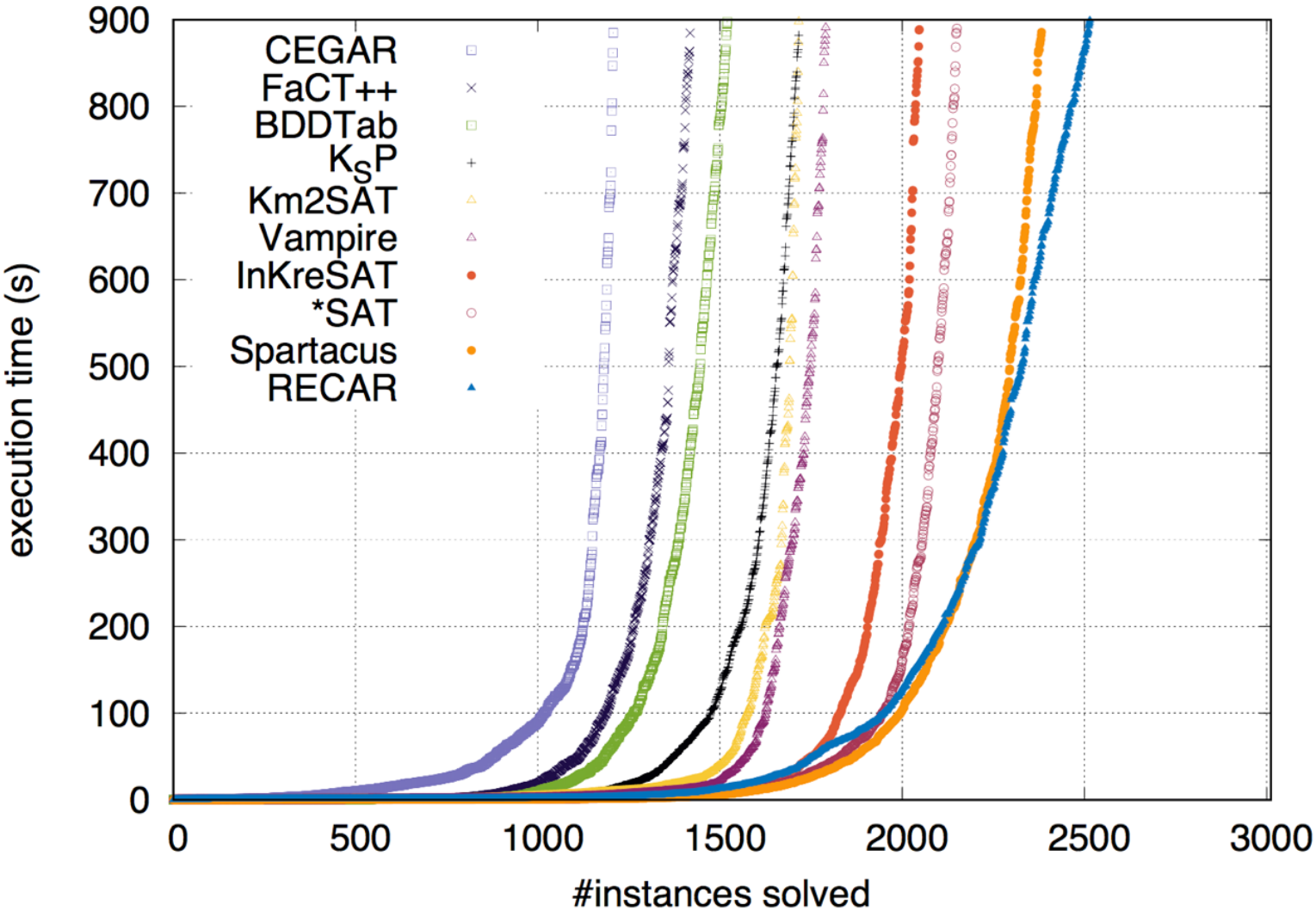


UNIVERSITÉ D'ARTOIS

Explanation of the Cactus-Plot



Some tweaks improve the results



UNIVERSITÉ D'ARTOIS

Conclusion

- SAT-based problem solving similar to assembly language programming
  - limited expressiveness
  - highly efficient
  - not for casual programmers

Conclusion

- SAT-based problem solving similar to assembly language programming
  - limited expressiveness
  - highly efficient
  - not for casual programmers
- Practical SAT solving is about
  - Encoding efficiently problems into CNF
  - Designing innovative SAT-based algorithms
  - Improving SAT solvers
  - Trusting solvers as efficient search space explorators
  - Being optimistic (versus worst case complexity)

Conclusion

- SAT-based problem solving similar to assembly language programming
  - limited expressiveness
  - highly efficient
  - not for casual programmers
- Practical SAT solving is about
  - Encoding efficiently problems into CNF
  - Designing innovative SAT-based algorithms
  - Improving SAT solvers
  - Trusting solvers as efficient search space explorators
  - Being optimistic (versus worst case complexity)

Definition (SAT-based problem solving)

- if proposal works, done
- else, try again, changing something
  (approach, encoding, solver, computers)
  driven by cause of failure

Thanks for your attention

Questions?

Bibliography I

Asín, R., Nieuwenhuis, R., Oliveras, A., and Rodríguez-Carbonell, E. (2009).
Cardinality networks and their applications.
In Kullmann, O., editor, *SAT*, volume 5584 of *Lecture Notes in Computer Science*, pages 167–180. Springer.

Biere, A., Berre, D. L., Lonca, E., and Manthey, N. (2014).
Detecting cardinality constraints in CNF.
In Sinz, C. and Egly, U., editors, *Theory and Applications of Satisfiability Testing - SAT 2014 - 17th International Conference, Held as Part of the Vienna Summer of Logic, VSL 2014, Vienna, Austria, July 14-17, 2014. Proceedings*, volume 8561 of *Lecture Notes in Computer Science*, pages 285–301. Springer.

Bibliography II


Chen, J.-C. (2010).
A new sat encoding of the at-most-one constraint.
In *In Proc. of the Tenth Int. Workshop of Constraint Modelling and Reformulation*.


Cook, W., Coullard, C., and Turán, G. (1987).
On the complexity of cutting-plane proofs.
Discrete Applied Mathematics, 18(1):25 – 38.


Eén, N. and Sörensson, N. (2006).
Translating pseudo-boolean constraints into sat.
JSAT, 2(1-4):1–26.

Bibliography III


Frisch, A. and Giannaros, P. (2010).
Sat encodings of the at-most-k constraint: Some old, some new, some fast, some slow.
In *Proceedings of the The 9th International Workshop on Constraint Modelling and Reformulation (ModRef 2010)*.


Gent, I. P. and Nightingale, P. (2004).
A new encoding of alldifferent into sat.
Proc. 3rd International Workshop on Modelling and Reformulating Constraint Satisfaction Problems, pages 95–110.

Bibliography IV


Halpern, J. Y. (1995).
The Effect of Bounding the Number of Primitive Propositions and the Depth of Nesting on the Complexity of Modal Logic.
Artificial Intelligence, 75(2):361–372.


Hooker, J. N. (1988).
Generalized resolution and cutting planes.
Ann. Oper. Res., 12(1-4):217–239.


Ladner, R. E. (1977).
The Computational Complexity of Provability in Systems of Modal Propositional Logic.
SIAM J. Comput., 6(3):467–480.

Bibliography V


Lagniez, J.-M., Le Berre, D., de Lima, T., and Montmirail, V. (2017).
A Recursive Shortcut for CEGAR: Application To The Modal Logic K Satisfiability Problem.
In *Proc. of IJCAI'17*.


Martins, R., Manquinho, V. M., and Lynce, I. (2012).
Parallel search for maximum satisfiability.
AI Commun., 25(2):75–95.


Sinz, C. (2005).
Towards an optimal cnf encoding of boolean cardinality constraints.
In van Beek, P., editor, *CP*, volume 3709 of *Lecture Notes in Computer Science*, pages 827–831. Springer.

UNIVERSITÉ D'ARTOIS

Bibliography VI