

# Structure-based simplification techniques of Boolean formulas

Lakhdar Saïs

CRIL-CNRS, Université Lille Nord de France  
Rue Jean Souvraz  
SP-18, F-62307, Lens Cedex 3, France

Tractability Workshop - Microsoft Research - Cambridge, July 5th, 2010

- 1 Introduction
- 2 Variable Elimination based Preprocessing
- 3 Simplification by reduction to a tractable fragment
- 4 Conclusion & perspectives

# Practical tractability ?

- Controversial : Tractability is a theoretical notion.
- 1992 : First french National Conference on **Practical solving of NP-Complete problems**

*Journées Nationales sur la résolution pratique de problèmes NP-Complets (JNPC)*

- push forward the limits of the currently-observed level of effectiveness of "solvers"
  - by widening the class of problems that can be solved in practice in a **reasonable** amount of time
- **A problem is tractable in practice** if most of the application instances can be solved in a reasonable amount of time
- How about SAT ?

# A zoom on the practical efficiency of SAT solvers

- Random category
  - Solving an unsatisfiable instance with 700 variables at the threshold will take 26 days [Dequen-Dubois 01]
  - While random satisfiable instances with millions of variables can be solved efficiently [Braunstein et al. 02]
- Application category (results from the SAT 2009 competition)
  - 70% of the 292 instances are solved in less than 15 mn (average).
  - 69 instances remain open ! (most of them are from crypto and bioinformatics family)
- Application instances with millions of variables and clauses are solved in seconds !!

# An example : post-cbmc-zfcp-2.8-u2.cnf

p cnf 11 483 525 (vars) 32 697 150 (clauses)

1 -3 0

$$x_3 = (x_1 \wedge x_2)$$

2 -3 0

-1 -2 3 0

...

Ten pages further

-11483518 -11483524 0

$$x_6 = \wedge(x_7, x_8, x_9, x_{10}, x_{11}, x_{12})$$

-11483519 -11483524 0

-11483520 -11483524 0

-11483521 -11483524 0

-11483522 -11483524 0

-11483523 -11483524 0

11483518 11483519 11483520 11483521 11483522 11483523 11483524 0

-8590303 -11483524 -11483525 0

$$x_{13} \Leftrightarrow x_{14} \Leftrightarrow x_{15}$$

8590303 11483524 -11483525 0

8590303 -11483524 11483525 0

-8590303 11483524 11483525 0

Solved in less than 1 minute !

How can we explain this gap between theory and practice ?

# Some hidden structures

Application instances present hidden structures such as

- **(Strong) backdoors** [Williams et al. 03]

*Subset of the "critical" variables such that once assigned a value the instance simplifies to a tractable class*

⇒ Real world problems have surprisingly small backdoor sets [Gomes et al. ]

- **Heavy tailed phenomena** [Gomes et al. 97]

*Certain problems, when solved by randomized backtracking, yield a runtime distribution that is heavy-tailed*

⇒ Strong variation in solution time, often from very short runs to very long

Explain how a solver can get "lucky" and solve very large instances

# Some apparent structures (encoding)

Application instances encoding include

- Boolean functions - circuit SAT

$$(y = f(x_1, x_2, \dots, x_n), f \in \{\wedge, \vee, \Leftrightarrow\})$$

- Dependencies (definability) (e.g.  $X \Rightarrow \phi$ )

- Large tractable sub-formulas (Horn, 2SAT, etc.)

Any boolean formula can be seen as a conjunction  
(network) of tractable classes

Explain the effectiveness of simplification techniques ?

# Simplifying boolean formulas ?

The objective is double

## 1 Simplify to reduce the solving time

(e.g. `NiVER` [Subbarayan & Pradhan, SAT'04], `SatElite` [Eén & Biere 05])

⇒ Eliminate redundancies

⇒ Identify a (sub-)formula capturing the instance hardness

## 2 Prove that the instance can be solved in polynomial time (e.g. `UHorn` [Fourdrinoy et al. 07])

⇒ e.g. tractable fragment memberships ?

Most of the simplification techniques belong to the first category

(e.g. `3-Resolution` [Li & Anbulagan 97], `2-SIMPLIFY` [Brafman 01], `HyPre`

[Bacchus & Winter 03], `ProbIt` [Lynce & Marques-Silva 03], `RemoveRed`

[Fourdrinoy-et al 07])

# Variable elimination based preprocessing

## Preliminary definitions

- CNF :  $\mathcal{F} = (\neg x_1 \vee \neg x_2 \vee x_3) \wedge (\neg x_1 \vee x_2) \wedge (\neg x_2 \vee \neg x_3) \wedge (\neg x_3)$
- Partial interpretation :  $\rho : X \subseteq \underline{\mathcal{V}(\mathcal{F})} \rightarrow \{\text{faux}, \text{vrai}\}$
- Simplification :  $\mathcal{F}|_\rho$  denotes the formula simplified by  $\rho$
- Formula  $\mathcal{F}$  closed by UP :  $\mathcal{F}^* = (\neg x_1 \vee \neg x_2) \wedge (\neg x_1 \vee x_2)$
- Resolution rule :  $\eta[x_2, (\neg x_1 \vee x_2), (\neg x_2 \vee \neg x_3)] = (\neg x_1 \vee \neg x_3)$
- Deduction % UP :  $\mathcal{F} \models^* c$ , iff  $(\mathcal{F}|_c)^* = \perp$
- Subsumption :  $c_1$  subsums  $c_2$  iff  $c_1 \subseteq c_2$ .
- Self Subsumption :  $\eta[x_2, (\neg x_1 \vee \neg x_2 \vee x_3), (\neg x_1 \vee x_2)] = (\neg x_1 \vee x_3)$
- Variable Elimination Resolution (VER) :  
 $\mathcal{F} \Leftrightarrow (\mathcal{F} - [\mathcal{F}_x \cup \mathcal{F}_{\neg x}]) \cup \eta[x, \mathcal{F}_x, \mathcal{F}_{\neg x}]$
- Hyper Binary Resolution :  $(a \vee b \vee c)(\neg a \vee d)(\neg b \vee d) \models (d \vee c)$
- Hyper Unary Resolution :  $(a \vee b \vee c)(\neg a \vee d)(\neg b \vee d)(\neg c \vee d) \models (d)$

# Variable elimination based preprocessing

## Non Increasing VER ( $NiVER$ ) [Sathiamoorthy et al. 04]

- $NiVER$  eliminates variables by VER only if there will be increase in space
- Most of the application instances have such variables, around 50% in many cases

### Example

$\mathcal{F}_c$	$\mathcal{F}_{-c}$
$(\neg a \vee c)$	$(a \vee b \vee \neg c)$
$(\neg b \vee c)$	$(\neg e \vee \neg f \vee \neg g \vee \neg c \neg d)$
$(\neg b \vee c)$	
Old number of literals : 14	Number of deleted clauses : 5
$(\neg a \vee \neg e \vee \neg f \vee \neg g \vee d)$	$(\neg b \vee a \vee b)$
$(\neg b \vee \neg e \vee \neg f \vee \neg g \vee d)$	$(\neg a \vee a \vee b)$
$(\neg d \vee a \vee b)$	$(\neg d \vee \neg e \vee \neg f \vee g \vee d)$
New number of literals : 13	Number of added clauses : 3

# Variable elimination based preprocessing

SatElite [Eén & Biere 05] includes

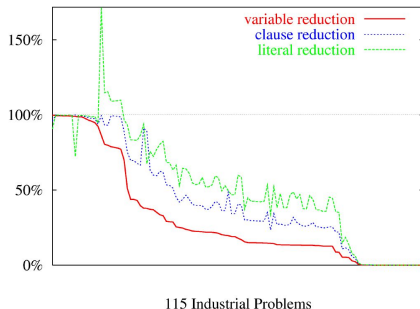
- Variable Elimination (when reduction in terms of the number of clauses is possible)
- Backward subsumption and Self subsumption

Details on Variable Elimination

- Try variable  $x$  where  $\#occur(x) \times \#occur(\neg x)$  is smallest first
- If both  $\#occur(x)$  and  $\#occur(\neg x)$  are  $> 10$ , skip.
- **Find definitions** [Ostrowski et al 04] :  $y = f(x_1, \dots, x_n)$ , where  $f \in \{\wedge, \vee\}$  **eliminate  $y$  by substitution**

# Size Reductions

Figure taken from [Eén & Biere 05]



New improvements,

- Vivification [Piette-Hamadi-Sais'08] (combines SatElite with sub-clause deduction)
- Blocked Clauses Elimination [Ostrowski et al. 04] [Järvisalo et al. 10]

# Simplification by reduction to a tractable fragment

Reducing SAT instances to polynomial ones [Fourdrinoy et al. 07]

**General idea** : Let  $\mathcal{C}$  be a tractable class (e.g. Horn), and  $\mathcal{F}$  a CNF formula

- 1 Partition  $\mathcal{F}$  into two sub-formula  $\mathcal{P} \in \mathcal{C}$  and  $\mathcal{R} = \mathcal{F} - \mathcal{P}$
- 2 Polynomially reduce  $\mathcal{R}$  to  $\mathcal{R}' \cup \mathcal{R}''$  where  $\mathcal{R}'' \in \mathcal{C}$
- 3 If  $\mathcal{R}'$  is empty,  $\mathcal{F} \equiv (\mathcal{P} \cup \mathcal{R}'') \in \mathcal{C}$

## Incarnation

- Consider Horn formula as the target tractable fragment  $\mathcal{C}$
- $\forall c \in \mathcal{R}$ , check if  $\mathcal{F} \models^* c'$  where  $c' \subset c$  a horn sub-clause ?

# Simplification by reduction to a tractable fragment

## Definition (U-Horn clause)

Let  $c = \{\neg n_1, \dots, \neg n_n, p_1, \dots, p_p\} \in \Sigma$ , with  $n \geq 0$  and  $p > 1$ .

$c$  is called U-Horn clause of  $\Sigma$  iff  $\exists c' = \{\neg n_1, \dots, \neg n_n, p_i\} \subset c$ , s.t.  $\Sigma \models^* c'$  or  $\Sigma \models^* \{\neg n_1, \dots, \neg n_n\}$ .

## Property

If  $c \in \Sigma$  is U-Horn and  $c' \subset c$  the Horn clause s.t.  $\Sigma \models^* c'$  then  $\Sigma$  is satisfiable iff  $(\Sigma \setminus \{c\}) \cup \{c'\}$  is satisfiable.

# Simplification by reduction to a tractable fragment

Definition (*U-redundancy*) (Fourdrinoy et al. 07)

A clause  $c \in \Sigma$  is *U-redundant* iff  $\Sigma \setminus \{c\} \models^* c$ .

Property

If  $c \in \Sigma$  is U-redundant then  $\Sigma$  is satisfiable iff  $\Sigma \setminus \{c\}$  is satisfiable.

Thus, U-redundant clauses can be safely deleted from  $\Sigma$ .

# Simplification by reduction to a tractable fragment

## Property

Let  $c = \{\neg n_1, \dots, \neg n_n, p_1, \dots, p_p\} \in \Sigma$ . If  $\Sigma \models^* \{\neg n_1, \dots, \neg n_n\}$  or  $\exists p_i \in c$  s.t.  $\Sigma \models^* \{\neg n_1, \dots, \neg n_n, p_i\}$  and  $\Sigma \models^* \{\neg n_1, \dots, \neg n_n, \neg p_i\}$  then  $\Sigma \models \{\neg n_1, \dots, \neg n_n\}$ .

## Definition (*U-NRes*)

When a clause  $c = \{\neg n_1, \dots, \neg n_n, p_1, \dots, p_p\} \in \Sigma$  satisfies the previous property,  $U-NRes(c) = \{\neg n_1, \dots, \neg n_n\}$ .

# Simplification by reduction to a tractable fragment

## Property

Let  $c = \{\neg n_1, \dots, \neg n_n, p_1, \dots, p_p\} \in \Sigma$ .

If  $\exists p_i \in c$  s.t.  $\Sigma \not\models^* \{\neg n_1, \dots, \neg n_n, p_i\}$  and  $\Sigma \models^* \{\neg n_1, \dots, \neg n_n, \neg p_i\}$ ,  
then  $\Sigma \models^* \{\neg n_1, \dots, \neg n_n, p_1, \dots, p_{i-1}, p_{i+1}, \dots, p_p\}$ .

## Definition

When a clause  $c = \{\neg n_1, \dots, \neg n_n, p_1, \dots, p_p\} \in \Sigma$  satisfies the previous property for the literals  $p_i$  to  $p_j$ ,

$U\text{-}PRes(c) = \{\neg n_1, \dots, \neg n_n, p_1, \dots, p_{i-1}, p_{j+1}, \dots, p_p\}$ .

# Algorithm 1: isU-Horn

**Input:** a SAT instance  $\Sigma$

**Output:** *true* if  $\Sigma$  is U-Horn; *false* otherwise

**begin**

$\Sigma' \leftarrow \{C \mid C \in \Sigma \text{ s.t. isHorn}(C)\};$

**forall**  $C \in \Sigma$  s.t.  $C = \{\neg n_1, \dots, \neg n_n, p_1, \dots, p_p\}, n \geq 0, p > 1$  **do**

**if**  $\Sigma \models^* \{\neg n_1, \dots, \neg n_n\}$  **then**  $\Sigma' \leftarrow \Sigma' \cup \{\{\neg n_1, \dots, \neg n_n\}\};$

**else**

$\Sigma'' \leftarrow \emptyset; C' \leftarrow C;$

**forall**  $p_i \in C$  **do**

**if**  $\Sigma \models^* \{\neg n_1, \dots, \neg n_n, p_i\}$  **then**

**if**  $\Sigma \models^* \{\neg n_1, \dots, \neg n_n, \neg p_i\}$  **then**

$\Sigma' \leftarrow \Sigma' \cup \{\{\neg n_1, \dots, \neg n_n\}\};$

**else**

$\Sigma'' \leftarrow \Sigma'' \cup \{\{\neg n_1, \dots, \neg n_n, p_i\}\};$

**else**

**if**  $\Sigma \models^* \{\neg n_1, \dots, \neg n_n, \neg p_i\}$  **then**  $C' \leftarrow C' \setminus \{p_i\};$

**if**  $\{\neg n_1, \dots, \neg n_n\} \not\subseteq \Sigma'$  **then**

**if**  $\Sigma'' = \emptyset$  **then**  $\Sigma' \leftarrow \Sigma' \cup \{C'\};$

**else**  $\Sigma' \leftarrow \Sigma' \cup \Sigma'';$

$\Sigma' \leftarrow \text{redundancyUP}(\Sigma');$

**return** isHorn( $\Sigma'$ );

**end**

# Some experiments

Instances	# var.	# cla.	#UP	time	CNF instances	# var.	# cla.	#UP	time
aim					IBM				
.100-2.0-1-2	100	200	456	0	.04..k15	15300	65598	397812	0.25
.100-6.0-1-1	100	600	2502	0	.05..k15	25128	134922	1708357	1.22
.100-6.0-1-2	100	600	2534	0	.15..k100	226970	893496	2432156	2.46
.100-6.0-1-3	100	600	777	0	.15..k15	30790	119911	184301	0.19
.100-6.0-1-4	100	600	568	0	.15..k20	42330	165416	252596	0.26
.200-6.0-1-2	200	1200	6113	0.01	.15..k25	53870	210921	329216	0.33
.200-6.0-1-4	200	1200	696	0	.15..k30	65410	256426	413391	0.42
.50-2.0-1-2	50	100	218	0	.15..k35	76950	301931	506031	0.5
.50-2.0-1-3	50	100	250	0	.15..k40	88490	347436	606086	0.6
.50-2.0-1-4	50	100	156	0	.15..k45	100030	392941	714746	0.71
.50-6.0-1-1	50	300	516	0	.15..k50	111570	438446	830681	0.83
.50-6.0-1-2	50	300	692	0	.15..k55	123110	483951	955361	0.99
.50-6.0-1-3	50	300	440	0	.15..k60	134650	529456	1087176	1.07
.50-6.0-1-4	50	300	1621	0	.15..k65	146190	574961	1227876	1.22
.-r1-b3-k1.2	660004	5281	56944	0.21	.15..k70	157730	620466	1375571	1.38
.-r1-b4-k1.1	397893	7089	105048	0.18	.15..k75	169270	665971	1532291	1.53
.-r1-b4-k1.2	922148	6818	60079	0.29	.15..k80	180810	711476	1695866	1.69
.-r2-b2-k1.2	406052	6064	54402	0.15	.15..k85	192350	756981	1868606	1.88
.-r2-b3-k1.2	668180	9169	100807	0.27	.15..k90	203890	802486	2048061	2.06
.-r2-b4-k1.1	406052	12784	178182	0.25	.15..k95	215430	847991	2236821	2.26
.-r2-b4-k1.2	930282	12464	175575	0.37	.22..k10	18919	77414	596987	0.4
jnh10	100	850	6737	0.02	.22..k15	29833	122814	1249118	0.96
jnh11	100	850	11187	0.02	.22..k20	40753	168249	1845706	1.48

TABLE: SAT 2007& SaTLib U-Horn instances : 127 instances belong to U-Horn class (8% of instances)

Instances	# var.	# cla.	#UP	time	CNF instances	# var.	# cla.	#UP	time
<b>jnh12</b>	100	850	5323	0.01	iso-brn005	1130	9866	13572	0.02
jnh13	100	850	4940	0.01	f19-b21-s0-0	746	3517	23805	0.03
jnh14	100	850	3362	0.01	f27-b10-s0-0	193	1113	8268	0.01
jnh15	100	850	7544	0.01	f27-b1-s0-0	193	1113	9401	0.01
jnh18	100	850	16943	0.03	f27-b2-s0-0	193	1113	5614	0.01
jnh19	100	850	10836	0.02	f27-b3-s0-0	193	1113	8716	0.01
jnh202	100	800	4641	0.01	f27-b4-s0-0	193	1113	5992	0.01
jnh203	100	800	18563	0.03	f27-b5-s0-0	193	1113	5626	0.01
jnh208	100	800	16108	0.03	f27-b8-s0-0	193	1113	7702	0.01
jnh20	100	850	8478	0.02	f27-b9-s0-0	193	1113	8684	0.01
jnh211	100	800	3030	0.01	f83-b11-s0-0	1000	43900	318968	0.74
jnh214	100	800	12131	0.02	f83-b14-s0-0	1000	43540	811348	1.61
jnh215	100	800	10558	0.02	f83-b17-s0-0	1000	43900	180456	0.37
jnh216	100	800	12821	0.02	par8-1-c	64	254	5613	0
jnh2	100	850	2201	0	<b>par8-1</b>	350	1149	9224	0
jnh302	100	900	246	0	<b>par8-2</b>	350	1157	7641	0
jnh303	100	900	13452	0.03	<b>par8-4-c</b>	67	266	6216	0
jnh304	100	900	1720	0	<b>par8-4</b>	350	1155	10248	0.01
jnh305	100	900	5348	0.01	<b>par8-5</b>	350	1171	7978	0
jnh307	100	900	2211	0	pitch.boehm	1192	6361	656	0.01
jnh308	100	900	15155	0.03	qg5-10.shuffled	1000	43900	318968	0.69
jnh309	100	900	2460	0.01	qg6-10.shuffled	1000	43540	811348	1.62
jnh310	100	900	3054	0.01	qg7-10.shuffled	1000	43900	180456	0.37
jnh4	100	850	5955	0.01	3col20_5_5.shuffled	40	176	774	0
jnh5	100	850	4151	0.01	3col20_5_6.shuffled	40	176	656	0
jnh8	100	850	4749	0.01	3col20_5_7.shuffled	40	176	903	0
jnh9	100	850	3099	0.01	3col20_5_9.shuffled	40	176	438	0

**TABLE:** SAT 2007& SatLib U-Horn instances : 127 instances belong to U-Horn class (8% of instances)

Instances	#var.	#cla.	cla	var	#lit.	#UP	time
een-tipb-sr06-par1	163647	484831	94%	95%	252004	68362283	38.98
ezfact16_10.shuffled	193	1113	26%	34%	335	5614	0.01
ezfact16_3.shuffled	193	1113	37%	44%	479	5992	0.01
IBM-02-04-rb-03-Sd.k30	29079	118925	44%	55%	31075	1393665	1.07
IBM-02-04-rb-05-Sd.k10	15399	81447	87%	93%	33203	1252239	0.76
IBM-02-04-rb-05-Sd.k20	34863	188452	74%	82%	72024	7798669	5.49
IBM-02-04-rb-05-Sd.k25	44598	241982	67%	75%	86760	18851484	16.38
IBM-02-04-rb-05-Sd.k30	54333	295512	60%	67%	99477	31503131	26.84
IBM-02-04-rb-06-Sd.k15	17501	75616	43%	49%	18130	1278040	1.07
IBM-02-04-rb-06-Sd.k20	23826	103226	71%	78%	41764	12961178	10.17
IBM-02-04-rb-10-Sd.k15	40278	159501	33%	35%	26022	8285670	6.89
IBM-02-04-rb-1-11-Sd.k10	28280	111519	47%	49%	25573	58410957	42.46
IBM-02-04-rb-18-Sd.k10	17141	69989	48%	55%	19878	13050828	8.7
IBM-02-04-rb-19-Sd.k10	21823	83902	24%	31%	13250	298260	0.26
IBM-02-04-rb-19-Sd.k15	34697	134023	17%	22%	14917	508638	0.47
IBM-02-04-rb-19-Sd.k20	47577	184178	17%	23%	23258	14607263	12.98
IBM-02-04-rb-20-Sd.k10	17567	72087	36%	41%	14004	5226452	3.63
IBM-02-04-rb-21-Sd.k10	15919	65180	35%	39%	11897	267966	0.21
IBM-02-04-rb-21-Sd.k15	25213	103881	25%	28%	13564	471438	0.39
IBM-02-04-rb-21-Sd.k20	34513	142616	26%	30%	21454	9624852	7.38
IBM-02-04-rb-22-Sd.k25	51673	213684	24%	27%	28739	30219471	22.32
IBM-02-04-rb-23-Sd.k10	18612	76086	41%	48%	16035	69713	0.09
IBM-02-04-rb-27-Sd.k10	6477	27070	62%	70%	10054	3826810	2.15
rip08.boehm	471	263	92%	59%	145	8728	0.01
x6dn.boehm	521	1255	86%	84%	1022	137818	0.07

TABLE: SAT 2007& SatLib U-Horn instances : 127 instances belong to U-Horn class (8% of instances)

# Conclusion

Simplifying Boolean formula is the best way to

- **Reduce** the formula to a core with less redundancies
- **Correct** the user encoding
- **Prove** that a given instance is tractable
- **Understand and explain** the efficiency of SAT solvers

# Perspectives

- 1 How to solve hard application (or random unsatisfiable) instances ?
  - Automatization of strong resolution proof systems  
**Extended resolution** [Tseitin 65] and **Symmetry resolution** [Krishnamurthy 85] proof systems
  - Move to Parallel SAT solving
- 2 Theoretical explanation of the efficiency of modern SAT solvers on application instances [French ANR project "TUPLES"] "Tractability for Understanding and Pushing forward the Limits of Efficient Solvers"

# French ANR project TUPLES

”Tractability for Understanding and Pushing forward the Limits of Efficient Solvers”

The aims and ambitions of the project can be summarized in three points :

- 1** theoretical and experimental research to better understand the reasons for both theoretical tractability and the practical effectiveness of solvers :
  - to better characterize what defines tractability,
  - to explain, using tractable classes, why solvers can solve instances of significant size,
- 2** research to develop new tractable classes that not only capture many instances or sub-instances of real-world problems but also allow effective and efficient implementation,
- 3** work to significantly expand the capabilities of current solvers, by designing, adapting or extending solvers to incorporate the exploitation of tractable classes.