Reflections Vol. 1 (2013) 1–101

Generalization of CNF and Consequences for DNF of Implicants under Distributive Expansion

Wolfgang Scherer Wolfgang.Scherer@gmx.de

Abstract

The conjunctive normal form CNF, is generalized to a conjunction of disjunctive normal form clauses CDF, by dropping the restrictions for syllogistic formulas. It is shown, that this leads to more desirable results for solving satisfiability and counting problems. The process of distributive expansion is further broken down into polynomial time and exponential time parts to develop a systematic and ordered algorithm, which cannot be effciently provided by other methods.

Contents

1. CNF, Syllogistic Formulas, BCF

This article is retrofitted onto the work of Blake, *Canonical expressions in Boolean algebra*[BLAKE] as outlined in *Boolean Reasoning: The Logic of Boolean Equations*[BROWN, chapter 4 and appendix A].

Obviously, the restrictions for a conjunctive normal form CNF — namely removal of duplicate literals and elimination of clauses with contradictory literals — have been loosened over time. However, this seems to have been a process of ad hoc reasoning[w3s].

Since Blake's proof depends on the notion of syllogistic formulas, the original definition of a CNF is kept fully intact, as it guarantees that a CNF is syllogistic[BROWN, chapter 4.6]. The restrictions are lifted separately by generalization.

Theorem 1. *Distributive expansion of a CNF formula P_c (multiplication of a product of sums, POS) results in a formula P_d in disjunctive normal form DNF (sum of products, SOP) defining the set of all prime implicants I_p for P_c. P_d is called a Blake canonical form BCF[bcf].*

The proof is given in [BROWN, theorem A.2.1].

2. From CNF to a Conjunction of DNF Clauses

As is pointed out in [BROWN, section 4.6.3], a syllogistic formula or a BCF are not necessarily always desired, since they may consist of a considerable amount of redundant prime implicates/implicants. However, no alternative method involving distributive expansion is given.

For an informal exploration by example, let

$$\begin{aligned}
m_0 &= (\ \ p \lor \ \ q \lor \ \ r), \\
m_1 &= (\neg p \land \ \ q), \\
m_2 &= (\neg p \land \neg q \land \ \ r), \\
m_3 &= (\ \ p \lor m_1 \lor m_2), \\
m_4 &= (\ \ p \land \ \ s), \\
m_5 &= (\ \ p \land \neg s \land \ \ t).
\end{aligned}$$

The following truth table shows that a disjunctive clause, $m_o = (\ \ p \lor \ \ q \lor \ \ r)$, of a CNF formula allows the maximum number of conjunctions during distributive expansion with another clause containing the literal p (see columns $(p \land q), (p \land r)$). It also shows, that a slight variation in the terms of a disjunctive clause, $m_3 = (p \lor (\neg p \land q) \lor (\neg p \land \neg q \land r))$, minimizes the number of possible conjunctions during distributive expansion (see columns $(p \land m_1), (p \land m_2)$).

p	q	r	m_0	p	m_1	m_2	m_3	$p \land q$	$p \land m_1$	$p \land r$	$p \land m_2$
0	0	0	0	0	0	0	0	0	0	0	0
0	0	1	1	0	0	1	1	0	0	0	0
0	1	0	1	0	1	0	1	0	0	0	0
0	1	1	1	0	1	0	1	0	0	0	0
1	0	0	1	1	0	0	1	0	0	0	0
1	0	1	1	1	0	0	1	0	0	1	0
1	1	0	1	1	0	0	1	1	0	0	0
1	1	1	1	1	0	0	1	1	0	1	0

While $m_0 = (p \lor q \lor r)$ and $m_3 = (p \lor (\neg p \land q) \lor (\neg p \land \neg q \land r))$ are logically equivalent, distributive expansion with another clause containing p should result in fewer conjunctions for m_3 than for m_0.

In example 1, normal distributive expansion of a CNF formula P produces the set of all prime implicants for P:

$$\begin{aligned}
P = &\ (\ \ p \lor \ \ q \lor \ \ r) \land (\neg p \lor \ \ s \lor \ \ t) \\
= &\ (\ \ p \land (\neg p \lor \ \ s \lor \ \ t)) \lor (\ \ q \land (\neg p \lor \ \ s \lor \ \ t)) \lor \\
&\ (\ \ r \land (\neg p \lor \ \ s \lor \ \ t)) \\
= &\ (\ \ p \land \neg p) \lor (\ \ p \land \ \ s) \lor (\ \ p \land \ \ t) \lor (\ \ q \land \neg p) \lor \\
&\ (\ \ q \land \ \ s) \lor (\ \ q \land \ \ t) \lor (\ \ r \land \neg p) \lor (\ \ r \land \ \ s) \lor \\
&\ (\ \ r \land \ \ t) \\
= &\ (\ \ p \land \ \ s) \lor (\ \ p \land \ \ t) \lor (\ \ q \land \neg p) \lor (\ \ q \land \ \ s) \lor \\
&\ (\ \ q \land \ \ t) \lor (\ \ r \land \neg p) \lor (\ \ r \land \ \ s) \lor (\ \ r \land \ \ t)
\end{aligned}$$

In example 2, distributive expansion of problem P_m (logically equivalent to P) produces the structurally equivalent CNF problem P by expansion and simplification of innermost clauses first:

$$\begin{aligned}
P_m = \; & (\ p \vee (\neg p \wedge \ q) \vee (\neg p \wedge \neg q \wedge \ r)) \wedge \\
& (\neg p \vee (\ p \wedge \ s) \vee (\ p \wedge \neg s \wedge \ t)) \\
= \; & ((\ p \vee \neg p) \wedge (\ p \vee \ q)) \vee (\neg p \wedge \neg q \wedge \ r)) \wedge \\
& ((\neg p \vee \ p) \wedge (\neg p \vee \ s)) \vee (\ p \wedge \neg s \wedge \ t)) \\
= \; & (\ p \vee \ q \vee (\neg p \wedge \neg q \wedge \ r)) \wedge \\
& (\neg p \vee \ s \vee (\ p \wedge \neg s \wedge \ t)) \\
= \; & (\ p \vee \ q \vee \neg p) \wedge (\ p \vee \ q \vee \neg q) \wedge (\ p \vee \ q \vee \ r) \wedge \\
& (\neg p \vee \ s \vee \ p) \wedge (\neg p \vee \ s \vee \neg s) \wedge (\neg p \vee \ s \vee \ t) \\
= \; & (\ p \vee \ q \vee \ r) \wedge (\neg p \vee \ s \vee \ t)
\end{aligned}$$

Further distributive expansion leads to the same result as example 1. This result is disappointing and probably the reason, why distributive expansion is intuitively categorized as purely exponential method. But the next example shows that there is a remedy.

In example 3, distributive expansion of P_m (logically equivalent to P), but with expansion and simplification of outermost clauses first, produces a set of implicants, which is structurally different from the set of prime implicants (some steps omitted for brevity):

$$\begin{aligned}
P_m = \; & (\ p \vee m_1 \vee m_2) \wedge (\neg p \vee m_4 \vee m_5) \\
= \; & (\ p \wedge (\neg p \vee m_4 \vee m_5)) \vee (m_1 \wedge (\neg p \vee m_4 \vee m_5)) \vee \\
& (m_2 \wedge (\neg p \vee m_4 \vee m_5)) \\
= \; & (\ p \wedge \neg p) \vee (\ p \wedge m_4) \vee (\ p \wedge m_5) \vee (m_1 \wedge \neg p) \vee \\
& (m_1 \wedge m_4) \vee (m_1 \wedge m_5) \vee (m_2 \wedge \neg p) \vee (m_2 \wedge m_4) \vee \\
& (m_2 \wedge m_5) \\
= \; & (\ p \wedge m_4) \vee (\ p \wedge m_5) \vee (m_1 \wedge \neg p) \vee (m_1 \wedge m_4) \vee && \mid m_1 = \neg p \wedge \ q \\
& (m_1 \wedge m_5) \vee (m_2 \wedge \neg p) \vee (m_2 \wedge m_4) \vee (m_2 \wedge m_5) \\
= \; & (\ p \wedge m_4) \vee (\ p \wedge m_5) \vee (\neg p \wedge \ q) \vee (\neg p \wedge \ q \wedge m_4) \vee && \mid m_4 = \ p \wedge \ s \\
& (\neg p \wedge \ q \wedge m_5) \vee (m_2 \wedge \neg p) \vee (m_2 \wedge m_4) \vee (m_2 \wedge m_5) \\
= \; & (\ p \wedge \ s) \vee (\ p \wedge m_5) \vee (\neg p \wedge \ q) \vee && \mid m_2 = \neg p \wedge \neg q \wedge \ r \\
& (\neg p \wedge \ q \wedge m_5) \vee (m_2 \wedge \neg p) \vee \\
& (m_2 \wedge \ p \wedge \ s) \vee (m_2 \wedge m_5) \\
= \; & (\ p \wedge \ s) \vee (\ p \wedge m_5) \vee (\neg p \wedge \ q) \vee (\neg p \wedge \ q \wedge m_5) \vee && \mid m_5 = \ p \wedge \neg s \wedge \ t \\
& (\neg p \wedge \neg q \wedge \ r) \vee (\neg p \wedge \neg q \wedge \ r \wedge m_5) \\
= \; & (\ p \wedge \ s) \vee (\ p \wedge \neg s \wedge \ t) \vee (\neg p \wedge \ q) \vee (\neg p \wedge \neg q \wedge \ r)
\end{aligned}$$

The results are repeated to make comparison easier:

$$
\begin{aligned}
& (\ p \vee \ q \vee \ r) \wedge (\neg p \vee \ s \vee \ t) \\
= \ & (\ p \wedge \ s) \vee (\ p \wedge \ t) \vee (\ q \wedge \neg p) \vee (\ q \wedge \ s) \vee \\
& (\ q \wedge \ t) \vee (\ r \wedge \neg p) \vee (\ r \wedge \ s) \vee (\ r \wedge \ t)
\end{aligned}
$$

$$
\begin{aligned}
& (\ p \vee (\neg p \wedge \ q) \vee (\neg p \wedge \neg q \wedge \ r)) \wedge \\
& (\neg p \vee (\ p \wedge \ s) \vee (\ p \wedge \neg s \wedge \ t)) \\
= \ & (\ p \vee m_1 \vee m_2) \wedge (\neg p \vee m_4 \vee m_5) \\
= \ & (\ p \wedge \ s) \vee (\ p \wedge \neg s \wedge \ t) \vee (\neg p \wedge \ q) \vee (\neg p \wedge \neg q \wedge \ r)
\end{aligned}
$$

The results are logically equivalent, since they were both obtained by algebraic transformation. However they differ structurally in the number of conjunctive clauses and the properties of the clauses in relation to each other.

The result from example 3 provides the incentive for a generalization of CNF formulas.

A generalized conjunction of DNF clauses is called CDF to avoid clashes with the abbreviation for a canoncial disjunctive normal form, CDNF. It also signifies, that there is no intention of constructing artificial "normal" or "canonical" forms, which are aesthetically nice, but quite impractical for, e.g., counting the number of satisfying total assignments.

While the definition of a DNF holds, the requirements for a CNF are lifted. This is important, when all conjunctions s of a DNF S consist of a single literal and therefore S degrades to a simple disjunction with plain literals of atomic variables. There is explicitely no requirement to eliminate duplicate literals in a degraded DNF. Variables can also appear both negated and unnegated in a degraded DNF.

Any CNF formula is therefore also a CDF formula, whereas not all CDF formulas are proper CNF formulas.

Theorem 2. *Any CDF formula P can be transformed to an equisatisfiable CNF formula P_s, which is syllogistic.*

Proof. Any CDF formula P can be trivially converted to a selection problem P_s by assigning a new variable v_{i_j} to each conjunction s_{i_j} of each DNF S_i of $P, i = (1, 2, \ldots, |P|), j = (1, 2, \ldots, |S_i|)$. A disjunction of all unnegated variables v_{i_j}, $(v_{i_1} \vee v_{i_2} \vee \ldots \vee v_{i_{|S_i|}})$, is added to P_s for each DNF S_i (at-least-one clauses). For each pair of variables $(v_{i_j}, v_{i_h}), j \neq h, h = (1, 2, \ldots, |S_i|)$, a disjunction $(\neg v_{i_j} \vee \neg v_{i_h})$ is added to P_s (at-most-one clauses). For each conflicting pair of conjunctions $(s_{i_j}, s_{f_g}), i \neq f, f = (1, 2, \ldots, |P|), g = (1, 2, \ldots, |S_f|)$ a disjunction $(\neg v_{i_j} \vee \neg v_{f_g})$ is added to P_s (conflict clauses)[HOS, chapter 2, direct encoding].

Obviously P_s is syllogistic, since there are either only negated or only unnegated variables in each clause[BROWN, chapter 4.6]. □

Note, that the weaker restrictions of a CDF formula P allow to extend P by adding all propositional variables p appearing in P, represented by disjunctions $(p \vee \neg p)$ for each variable. The corresponding selection problem P_s then carries its own translation map for the original set of variables. As a courtesy, SAT solvers will solve the selection problem for both the variables of the selection problem and the variables of the original CDF problem and a tedious mapping process is not necessary.

Since a selection problem P_s is again a CDF, a malicious encoder can boost the virtual hardness of any problem by applying theorem 2 any number of times.

3. Conflict Maximization

The term conflict exclusively refers to opposing literals in clauses and should not be confused with the term conflict from the context of conflict driven clause learning CDCL.

While a CDF is not restricted to special DNF clauses, the most interesting in the context of this article is the set E of DNF clauses which are logically equivalent to a disjunction of literals (aka. *regular* CNF clause).

The most prominent members of E are DNF clauses, where all conjunctions have a maximum of conflicting literals.

Theorem 3. *If we transform a disjunctive clause S_d with k literals to a disjunctive clause S_m of conjunctions by replacing each literal $l_i, i = (1, ..., k)$ with the conjunction $(\neg l_1 \wedge ... \wedge \neg l_{i-1} \wedge l_i)$, then S_m will be logically equivalent to S_d. The clause S_m is called a clause with maximized conflicts.*

Proof. Distributive expansion of S_m shows the logical equivalence to the unmaximized clause S_d. E.g.:

$$\begin{aligned} & ((p) \vee (\neg p \wedge q)) \\ = \; & (p \vee \neg p) \wedge (p \vee q) \\ = \; & (p \vee q) \end{aligned}$$

□

Theorem 4. *Distributive expansion of a CDF formula with maximized conflicts P_m, results in a DNF formula P_u defining a set of (not necessarily prime) implicants I_u for P_m, if expansion and simplification of outermost clauses is performed strictly before innermost clauses. The implicants from I_u cover all possible satisfying total assignments for P_m.*

Proof. Since the order of expansion and simplification does not change the logical function represented by the expansion, the result I_p from theorem 1 must be logically equivalent to I_u. Since I_p covers all satisfying total assignments, by extension I_u must also cover all satisfying total assignments. □

Note, that the order of expansion and simplification is significant. If innermost clauses are simplified first, P_m degrades to a regular CNF formula (as shown in example 2) and the result of distributive expansion is the set of all prime implicants.

Hypothesis 5. *The implicants I_u established by theorem 4 are unique, in that no implicant $M_x \in I_u$ covers the satisfying total assignments of any other implicant $M_y \in I_u, x \neq y; x, y \in \{1, 2, \dots, |I_u|\}$.*

The proof is omitted here, since it becomes much simpler in the generalized theory of the satoku matrix. However, it is still mentioned, since uniqueness of implicants is very convenient, if the number of satisfying total assignments must be counted. This is much easier with the set of implicants I_u than with the set of prime implicants I_p from theorem 1, where duplicate total assignments have to be accounted for.

4. Partial Distributive Expansion

Now that we have established that distributive expansion has more than just trivial aspects, a systematic analysis of its properties is warranted.

Since partial distributive expansion PDE is based on a computer algorithm, indices are shifted to programming language conventions to avoid translation errors between the program source code and the description:
Disjunctions S are labeled $S_i, i = (0, 1, \dots, m-1), m = |P|$.
Conjunctions s are labeled $s_{i_j}, j = (0, 1, \dots, |S_i| - 1)$.
Dependencies between all conjunctions of disjunctions S_i and $S_f, f = (0, 1, \dots, m-1)$ are denoted as $S_{i,f}$.
Dependencies between conjunctions s_{i_j} and $s_{f_g}, g = (0, 1, \dots, |S_f| - 1)$ are denoted as row, column pairs s_{i_j, f_g}.
Literals l_{i_j} of CNF clauses S_i relate to the conjunction dependencies s_{i_j, i_j} in the PDE matrix.

Partial distributive expansion PDE is designed as a systematic process to refine partial assignments incrementally without arbitrary decisions. Each conjunction s_{i_j} of a DNF S_i of a CDF P is interpreted as a partial assignment for P. This is motivated by the fact that distributive expansion produces a set of implicants I for P. Each implicant of I therefore necessarily refines one or more conjunctions s_{i_j}.

It turns out, that neither implication graphs nor adjacency matrices nor clause-variable matrices[HOS, section 11.2], let alone the principles of decision algorithms and CDLC, are sufficient to describe all mechanisms of PDE efficiently. The closest representation for PDE is an adjacency matrix. However, an adjacency matrix lacks the necessary properties to systematically examine the consequences of dependencies between disjunctions of conjunctions (clause constraints).

The PDE matrix is itself only a stepping stone to the generalized satoku matrix. Its sole purpose is to illustrate the relation between a CDF problem with clauses and variables and the fully abstracted satoku matrix, where the notion of a substantial difference between clauses and variables becomes meaningless. Therefore, only an informal presentation by example is included, postponing the precise formalization to the satoku matrix.

Starting with a propositional CNF formula P with m disjunctive clauses C_i of size k, $m = |P|, i = 0 \ldots (m-1), k = |C_i|$:

$$\begin{array}{l} (\neg a \vee \neg b) \wedge \\ (\neg a \vee \;\; d) \wedge \\ (\;\; a \vee \;\; b) \wedge \\ (\;\; a \vee \;\; c) \end{array}$$

Convert each disjunction C_i of literals l_{i_j} to a disjunction S_i of conjunctions s_{i_j}, $j = 0 \ldots (|C_i| - 1)$:

$$\begin{array}{l} ((\neg a) \vee (\neg b)) \wedge \\ ((\neg a) \vee (\;\; d)) \wedge \\ ((\;\; a) \vee (\;\; b)) \wedge \\ ((\;\; a) \vee (\;\; c)) \end{array}$$

As visual hint, spread each disjunction S_i over two lines:

$$\begin{array}{lccccccccl} ((& \neg a & & & & & & & &) \vee \\ \;(& & \neg b & & & & & & &)) \wedge \\ ((& & & \neg a & & & & & &) \vee \\ \;(& & & & d & & & & &)) \wedge \\ ((& & & & & a & & & &) \vee \\ \;(& & & & & & b & & &)) \wedge \\ ((& & & & & & & a & &) \vee \\ \;(& & & & & & & & c &)) \end{array}$$

Extend each conjunction s_{i_j} with truth values T (signifying independency) to a length of $\sum_{i=0}^{[P]-1} |S_i|$ in the following manner:

$$\begin{array}{lccccccccccccccccl} ((& \neg a & \wedge & T & \wedge & T & \wedge & T & \wedge & T & \wedge & T & \wedge & T & \wedge & T &) \vee \\ \;(& T & \wedge & \neg b & \wedge & T & \wedge & T & \wedge & T & \wedge & T & \wedge & T & \wedge & T &)) \wedge \\ ((& T & \wedge & T & \wedge & \neg a & \wedge & T & \wedge & T & \wedge & T & \wedge & T & \wedge & T &) \vee \\ \;(& T & \wedge & T & \wedge & T & \wedge & d & \wedge & T & \wedge & T & \wedge & T & \wedge & T &)) \wedge \\ ((& T & \wedge & T & \wedge & T & \wedge & T & \wedge & a & \wedge & T & \wedge & T & \wedge & T &) \vee \\ \;(& T & \wedge & T & \wedge & T & \wedge & T & \wedge & T & \wedge & b & \wedge & T & \wedge & T &)) \wedge \\ ((& T & \wedge & T & \wedge & T & \wedge & T & \wedge & T & \wedge & T & \wedge & a & \wedge & T &) \vee \\ \;(& T & \wedge & T & \wedge & T & \wedge & T & \wedge & T & \wedge & T & \wedge & T & \wedge & c &)) \end{array}$$

The PDE matrix is constructed to fully represent this structure with additional vertical lines to denote clause limits.

s_{0_0}	$\neg a$							
s_{0_1}		$\neg b$						
s_{1_0}			$\neg a$					
s_{1_1}				d				
s_{2_0}					a			
s_{2_1}						b		
s_{3_0}							a	
s_{3_1}								c

4.1 Requirement Identification

The PDE matrix can also be interpreted as a representation of the selection problem for P, where each row s_{i_j} shows the literals that must become true, when row s_{i_j} is selected from clause S_i. It is suitable to discuss requirement propagation in this context.

Following the consequences of the selection problem, observe, that if s_{0_0} is *eventually* selected, then s_{2_0} can no longer be selected, because literal $\neg a$ at position $s_{0_0,0_0}$ conflicts with literal a at position $s_{2_0,2_0}$ (x), therefore, s_{2_1} must *then* be selected to preserve satisfiability.

It makes no difference, whether this required selection is made later as a matter of circumstances, or if it is promised in advance by refining partial assignment s_{0_0} with literal b at position $s_{0_0,2_1}$:

s_{0_0}	$\neg a$				x	b		
s_{0_1}		$\neg b$						
s_{1_0}			$\neg a$					
s_{1_1}				d				
s_{2_0}					a			
s_{2_1}						b		
s_{3_0}							a	
s_{3_1}								c

By analogy, the same is true for selection s_{2_0} and the conflicting selection at s_{0_0} (x):

s_{0_0}	$\neg a$					b		
s_{0_1}		$\neg b$						
s_{1_0}			$\neg a$					
s_{1_1}				d				
s_{2_0}	x	$\neg b$			a			
s_{2_1}						b		
s_{3_0}							a	
s_{3_1}								c

Distributive expansion of clauses S_0, S_2 shows, that this is exactly what will happen:

$$\begin{aligned} & (\neg a \vee \neg b) \wedge (a \vee b) \\ = \; & ((\neg a \wedge (a \vee b)) \vee (\neg b \wedge (a \vee b))) \\ = \; & (\neg a \wedge a) \vee (\neg a \wedge b) \vee (\neg b \wedge a) \vee (\neg b \wedge b) \\ = \; & (\neg a \wedge b) \vee (\neg b \wedge a) \end{aligned}$$

However, the result was obtained without the exponential number of intermediate results of distributive expansion.

4.2 Requirement Propagation Round 1

Proceeding further, the PDE matrix is transformed to:

s_{0_0}	$\neg a$					b		c
s_{0_1}		$\neg b$			a			
s_{1_0}			$\neg a$			b		c
s_{1_1}				d				
s_{2_0}		$\neg b$		d	a			
s_{2_1}	$\neg a$					b		
s_{3_0}		$\neg b$		d			a	
s_{3_1}								c

Observe, that partial assignment s_{0_1}, which requires partial assignment s_{2_0} is no longer consistent, since s_{2_0} has been refined with d from $s_{1_1,1_1}$ during the first round of partial distributive expansion.

4.3 Requirement Propagation Round 2

It is therefore necessary to enter another round of requirement propagation and refine s_{0_1} with d at position $s_{0_1,1_1}$ also:

s_{0_0}	$\neg a$					b		c
s_{0_1}		$\neg b$		d	a	x		
s_{1_0}			$\neg a$			b		c
s_{1_1}				d				
s_{2_0}		$\neg b$		d	a			
s_{2_1}	$\neg a$					b		
s_{3_0}		$\neg b$		d			a	
s_{3_1}								c

Following all consequences, the PDE matrix is transformed to:

s_{0_0}	$\neg a$					b		c
s_{0_1}		$\neg b$		d	a			
s_{1_0}	$\neg a$		$\neg a$			b		c
s_{1_1}				d				
s_{2_0}		$\neg b$		d	a			
s_{2_1}	$\neg a$					b		c
s_{3_0}		$\neg b$		d	a		a	
s_{3_1}								c

It is obvious, that requirement propagation must terminate and has a strictly polynomial worst case running time. A more detailed rationale is postponed to the description of the satoku matrix.

4.4 Translate PDE to CDF

Translating the PDE matrix to a CDF formula P_t renders a conjunction of disjunctions S_i of partial assignments s_{i_j}:

$$
\begin{array}{l}
((\neg a \wedge b \wedge c) \vee (a \wedge \neg b \wedge d)) \wedge \\
((\neg a \wedge b \wedge c) \vee (d)) \wedge \\
((a \wedge \neg b \wedge d) \vee (\neg a \wedge b \wedge c)) \wedge \\
((a \wedge \neg b \wedge d) \vee (c)).
\end{array}
$$

After removal of redundant claues, P_t reduces further to:

$$
\begin{array}{l}
((\neg a \wedge b \wedge c) \vee (a \wedge \neg b \wedge d)) \wedge \\
((\neg a \wedge b \wedge c) \vee (d)) \wedge \\
((a \wedge \neg b \wedge d) \vee (c)).
\end{array}
$$

Induction over the full truth table shows that P and P_t are logically equivalent, which is necessarily so.

The partial assignments $(a \wedge \neg b \wedge d)$ and $(\neg a \wedge b \wedge c)$ are also implicants for P and satisfiability of P is asserted without the need to perform the final exponential step of distributive expansion.

4.5 PDE with Conflict Maximization

Applying theorem 3 to P produces the CDF P_m:

$$
\begin{array}{l}
((\neg a) \vee (\;\; a \wedge \neg b)) \wedge \\
((\neg a) \vee (\;\; a \wedge \;\; d)) \wedge \\
((\;\; a) \vee (\neg a \wedge \;\; b)) \wedge \\
((\;\; a) \vee (\neg a \wedge \;\; c))
\end{array}
$$

The corresponding PDE matrix presents as:

s_{0_0}	$\neg a$			
s_{0_1}	$a \wedge \neg b$			
s_{1_0}		$\neg a$		
s_{1_1}		$a \wedge d$		
s_{2_0}			a	
s_{2_1}			$\neg a \wedge b$	
s_{3_0}				a
s_{3_1}				$\neg a \wedge c$

Requirement propagation transforms the PDE matrix to:

s_{0_0}	$\neg a$	$\neg a$	$\neg a \wedge b$	$\neg a \wedge c$
s_{0_1}	$a \wedge \neg b$	$a \wedge d$	a	a
s_{1_0}	$\neg a$	$\neg a$	$\neg a \wedge b$	$\neg a \wedge c$
s_{1_1}	$a \wedge \neg b$	$a \wedge d$	a	a
s_{2_0}	$a \wedge \neg b$	$a \wedge d$	a	a
s_{2_1}	$\neg a$	$\neg a$	$\neg a \wedge b$	$\neg a \wedge c$
s_{3_0}	$a \wedge \neg b$	$a \wedge d$	a	a
s_{3_1}	$\neg a$	$\neg a$	$\neg a \wedge b$	$\neg a \wedge c$

Translation to a CDF P_u results in:

$$
\begin{array}{l}
((\neg a \wedge b \wedge c) \vee (a \wedge \neg b \wedge d)) \wedge \\
((\neg a \wedge b \wedge c) \vee (a \wedge \neg b \wedge d)) \wedge \\
((a \wedge \neg b \wedge d) \vee (\neg a \wedge b \wedge c)) \wedge \\
((a \wedge \neg b \wedge d) \vee (\neg a \wedge b \wedge c))
\end{array}
$$

After removal of redundant clauses, P_u presents a DNF of implicants:

$$(a \wedge \neg b \wedge d) \vee (\neg a \wedge b \wedge c)$$

This shows, that the PDE matrix is sufficient to process formulas according to theorem 4 by inherently avoiding early simplification of innermost clauses.

4.6 Rationale for PDE

It has been shown, that it is possible to efficiently transform the formula P:

$$\begin{array}{l}(\neg a \vee \neg b) \wedge (\neg a \vee \quad d) \wedge \\ (\quad a \vee \quad b) \wedge (\quad a \vee \quad c)\end{array}$$

to the logically equivalent formula P_u:

$$(a \wedge \neg b \wedge d) \vee (\neg a \wedge b \wedge c)$$

without any clause learning through exponential back-tracking and without any exponential distributive expansion.

5. Experiments

To show that PDE is useful for more than just trivial 2-clause problems, experiments with 100 randomly generated 3-CNF formulas having 40 variables and 171 clauses[ws-exp] were conducted. All problems, except one were solved trivially in polynomial time with a PDE based algorithm alone.

See appendix B for a detailed summary of the experiments.

Further experiments with different versions of transformations according to theorem 2 (direct encoding) show that DPLL/CDCL solvers suffer significantly from the virtual increase in hardness by the increased number of clauses and variables, when conflict maximization is not applied.

CNF			**Direct enc.**			**Cfl max.**			**Cfl red.**		
Var	Cls	Dec/a	Var	Cls/a	Dec/a	Var	Cls/a	Dec/a	Var/a	Cls/a	Dec/a
40	171	30	513	2309	236	513	97905	2	86	2934	2

Solver lingeling, Var = variables, Cls = clauses, Dec = Decisions, /a = average

A surprising result, which seems counter-intuitive at first is the fact, that conflict maximization with subsequent transformation to a selection problem significantly reduces the number of decisions needed by DPLL/CDCL solvers (even below the number of decisions needed for the original CNF encoding in more than 90 out of 100 cases).

The average number of clauses (97905/2934) for the selection problems with conflict maximization is much higher than for the original CNF encoding (171) and also higher than the average number of clauses for the selection problem without conflict maximization (2309).

The results imply that there is no correlation between the hardness of a problem and the raw number of variables and clauses at all.

For the lookahead solver march_rw, the results indicate a somewhat proportional increase in virtual hardness as expected. I.e., there is no gain through conflict maximization in relation to the original CNF encoding. However, the decrease in virtual hardness in relation to direct encoding without conflict maximization is also significantly high.

CNF			**Direct enc.**			**Cfl max.**			**Cfl red.**		
Var	Cls	LA/a	Var	Cls/a	LA/a	Var	Cls/a	LA/a	Var/a	Cls/a	LA/a
40	171	122	513	2309	10720	513	97905	365	86	2934	220

Solver march_rw, Var = variables, Cls = clauses, LA = LookAheadCount, /a = average

These results as well imply that there is no correlation between the hardness of a propositional problem and the raw number of variables and clauses.

A consistent test for a local search solver (WalkSat) could not be conducted, since many problems could no longer be solved in reasonable time after transformation to a selection problem.

6. Conclusion

It is obvious, that the PDE method has its limits and certainly does not generalize to an entirely polynomial algorithm for distributive expansion (sadly, there is no magic in logic).

However, the experiments with the effects of PDE on SAT solvers alone provide enough incentive to study the consequences of this method further.

The analysis shows, that the current models of logic, namely graph theory (lacking full clause constraints and dynamic clause operations) and the theory of decision problems (working on a sparse one-dimensional representation of a propositional formula) cannot efficiently produce the results of partial distributive expansion.

Theorem 2 shows vividly, that the notion of normal forms and fixed sets of variables is illusionary. There is not even a logically sound argument for generalized equisatisfiability implying any kind of nice (aka. algebraic) functional relation between one set of variables and another set of variables, although both correctly define the dependencies of a problem with the same core clause structure.

The fact that there are infinitely many versions of the exact same problem — where one version should not be any harder to solve than any other version no matter how many additional 2-literal clauses or variables (which are only special cases of 2-literal clauses) are added — leads to the conclusion that a correct model of logic must focus on the structural properties of clauses alone.

Ultimately, the number of 2-literal clauses must not have any substantial influence on the running time of a solver algorithm. Therefore, a correct model of logic for satisfiability problems must also solve 2-SAT problems in polynomial time (which DPLL/CDCL does not) and it must deliver consistent explanations for the hardness of a problem (which graph theory does not, not even for 2-SAT problems).

PDE and conflict maximization are only the most basic principles of an (extremely simple) generalized theory, derived directly from logic itself, which is capable of modeling the intrinsic polymorphy and self-referentiality of logic, offering both a method to implicitely solve 2-SAT problems efficiently and consistent explanations for the hardness of satisfiability problems.

However, the popular belief and common expectation that the description should somehow be meaningfully produced with the incomplete concepts of graph theory and decision problems seems quite outlandish.

Citing the *Handbook of Satisfiability*:

> "In short, CNF modelling [sic] is an art and we must often proceed by intuition and experimentation"[HOS, section 2.5],

the whole point of this article can be summarized as:

> CDF modeling should not be an art, and it does not have to be. And once it no longer is, propositional problems are modeled correctly.

Appendix A. Examples

Examples for theorems are given to illustrate the different consequences.

A.1. Example for Theorem 1

$$
\begin{array}{rl}
 & (a \vee b) \wedge (c \vee d) \wedge (\neg a \vee \neg c) \\
= & ((a \wedge c) \vee (a \wedge d) \vee (b \wedge c) \vee (b \wedge d)) \wedge (\neg a \vee \neg c) \\
= & (\neg a \wedge ((a \wedge c) \vee (a \wedge d) \vee (b \wedge c) \vee (b \wedge d))) \\
 & (\neg c \wedge ((a \wedge c) \vee (a \wedge d) \vee (b \wedge c) \vee (b \wedge d))) \\
= & (\neg a \wedge a \wedge c) \vee (\neg a \wedge a \wedge d) \vee (\neg a \wedge b \wedge c) \vee (\neg a \wedge b \wedge d) \vee \\
 & (\neg c \wedge a \wedge c) \vee (\neg c \wedge a \wedge d) \vee (\neg c \wedge b \wedge c) \vee (\neg c \wedge b \wedge d) \\
= & (a \wedge \neg c \wedge d) \vee (\neg a \wedge b \wedge c) \vee (\neg a \wedge b \wedge d) \vee (b \wedge \neg c \wedge d)
\end{array}
$$

A.2. Example for Theorem 3

$$
\begin{array}{rcl}
(a \vee b) & \mapsto & ((a) \vee (\neg a \wedge b)) \\
(a \vee b \vee c) & \mapsto & ((a) \vee (\neg a \wedge b) \vee (\neg a \wedge \neg b \wedge c))
\end{array}
$$

A.3. Example 1 for Theorem 4

Maximization principle: maximize conflicts for most frequent variables last.

$$
\begin{array}{rll}
 & (a \vee b) \wedge (c \vee d) \wedge (\neg a \vee \neg c) & \mid p \vee q = (p \vee (\neg p \wedge q)) \\
= & (a \vee (\neg a \wedge b)) \wedge (c \vee (\neg c \wedge d)) \wedge (\neg a \vee (a \wedge \neg c)) & \mid x_1 := (\neg a \wedge b), \\
 & & \mid x_2 := (\neg c \wedge d), \\
 & & \mid x_3 := (a \wedge \neg c) \\
= & (a \vee x_1) \wedge (c \vee x_2) \wedge (\neg a \vee x_3) & \\
= & (((a \vee x_1) \wedge c) \vee ((a \vee x_1) \wedge x_2)) \wedge (\neg a \vee x_3) & \\
= & ((a \wedge c) \vee (x_1 \wedge c) \vee (a \wedge x_2) \vee (x_1 \wedge x_2)) \wedge (\neg a \vee x_3) & \\
= & (((a \wedge c) \vee (x_1 \wedge c) \vee (a \wedge x_2) \vee (x_1 \wedge x_2)) \wedge \neg a) \vee & \\
 & (((a \wedge c) \vee (x_1 \wedge c) \vee (a \wedge x_2) \vee (x_1 \wedge x_2)) \wedge x_3) & \\
= & (a \wedge c \wedge \neg a) \vee (x_1 \wedge c \wedge \neg a) \vee (a \wedge x_2 \wedge \neg a) \vee & \mid p \wedge \neg p = F \\
 & (x_1 \wedge x_2 \wedge \neg a) \vee (a \wedge c \wedge x_3) \vee (x_1 \wedge c \wedge x_3) \vee & \\
 & (a \wedge x_2 \wedge x_3) \vee (x_1 \wedge x_2 \wedge x_3) & \\
= & (x_1 \wedge c \wedge \neg a) \vee (x_1 \wedge x_2 \wedge \neg a) \vee (a \wedge c \wedge x_3) \vee & \mid x_1 = (\neg a \wedge b) \\
 & (x_1 \wedge c \wedge x_3) \vee (a \wedge x_2 \wedge x_3) \vee (x_1 \wedge x_2 \wedge x_3) & \\
= & (\neg a \wedge b \wedge c \wedge \neg a) \vee (\neg a \wedge b \wedge x_2 \wedge \neg a) \vee (a \wedge c \wedge x_3) \vee & \mid p \wedge p = p, \\
 & (\neg a \wedge b \wedge c \wedge x_3) \vee (a \wedge x_2 \wedge x_3) \vee (\neg a \wedge b \wedge x_2 \wedge x_3) & \mid p \wedge \neg p = F \\
= & (\neg a \wedge b \wedge c) \vee (\neg a \wedge b \wedge x_2) \vee (a \wedge c \wedge x_3) \vee & \mid x_2 = (\neg c \wedge d) \\
 & (\neg a \wedge b \wedge c \wedge x_3) \vee (a \wedge x_2 \wedge x_3) \vee (\neg a \wedge b \wedge x_2 \wedge x_3) & \\
= & (\neg a \wedge b \wedge c) \vee (\neg a \wedge b \wedge \neg c \wedge d) \vee & \mid x_3 = (a \wedge \neg c) \\
 & (a \wedge c \wedge x_3) \vee (\neg a \wedge b \wedge c \wedge x_3) \vee & \\
 & (a \wedge \neg c \wedge d \wedge x_3) \vee (\neg a \wedge b \wedge \neg c \wedge d \wedge x_3) & \\
= & (\neg a \wedge b \wedge c) \vee (\neg a \wedge b \wedge \neg c \wedge d) \vee (a \wedge c \wedge a \wedge \neg c) \vee & \\
 & (\neg a \wedge b \wedge c \wedge a \wedge \neg c) \vee (a \wedge \neg c \wedge d \wedge a \wedge \neg c) \vee & \\
 & (\neg a \wedge b \wedge \neg c \wedge d \wedge a \wedge \neg c) & \\
= & (a \wedge \neg c \wedge d) \vee (\neg a \wedge b \wedge c) \vee (\neg a \wedge b \wedge \neg c \wedge d) &
\end{array}
$$

A.4. Example 2 for Theorem 4

Maximization principle: maximize conflicts for most frequent variables first.

$$
\begin{array}{rll}
 & (a \vee b) \wedge (c \vee d) \wedge (\neg a \vee \neg c) & \mid p \vee q = (p \vee (\neg p \wedge q)) \\
= & ((a \wedge \neg b) \vee b) \wedge ((c \wedge \neg d) \vee d) \wedge ((\neg a \wedge c) \vee \neg c) & \mid x_1 := (a \wedge \neg b), \\
 & & \mid x_2 := (c \wedge \neg d), \\
 & & \mid x_3 := (\neg a \wedge c) \\
= & (x_1 \vee b) \wedge (x_2 \vee d) \wedge (x_3 \vee \neg c) & \\
= & ((x_1 \wedge (x_2 \vee d)) \vee (b \wedge (x_2 \vee d))) \wedge (x_3 \vee \neg c) & \\
= & ((x_1 \wedge x_2) \vee (x_1 \wedge d) \vee (b \wedge x_2) \vee (b \wedge d)) \wedge (x_3 \vee \neg c) & \\
= & (x_1 \wedge x_2 \wedge (x_3 \vee \neg c)) \vee (x_1 \wedge d \wedge (x_3 \vee \neg c)) \vee & \\
 & (b \wedge x_2 \wedge (x_3 \vee \neg c)) \vee (b \wedge d \wedge (x_3 \vee \neg c)) & \\
= & (x_1 \wedge x_2 \wedge x_3) \vee (x_1 \wedge x_2 \wedge \neg c) \vee (x_1 \wedge d \wedge x_3) \vee & \mid x_1 = (a \wedge \neg b) \\
 & (x_1 \wedge d \wedge \neg c) \vee (b \wedge x_2 \wedge x_3) \vee (b \wedge x_2 \wedge \neg c) \vee & \\
 & (b \wedge d \wedge x_3) \vee (b \wedge d \wedge \neg c) & \\
= & (a \wedge \neg b \wedge x_2 \wedge x_3) \vee (a \wedge \neg b \wedge x_2 \wedge \neg c) \vee & \mid x_2 = (c \wedge \neg d) \\
 & (a \wedge \neg b \wedge d \wedge x_3) \vee (a \wedge \neg b \wedge d \wedge \neg c) \vee & \\
 & (b \wedge x_2 \wedge x_3) \vee (b \wedge x_2 \wedge \neg c) \vee & \\
 & (b \wedge d \wedge x_3) \vee (b \wedge d \wedge \neg c) & \\
= & (a \wedge \neg b \wedge c \wedge \neg d \wedge x_3) \vee (a \wedge \neg b \wedge c \wedge \neg d \wedge \neg c) \vee & \mid p \wedge \neg p = F \\
 & (a \wedge \neg b \wedge d \wedge x_3) \vee (a \wedge \neg b \wedge d \wedge \neg c) \vee & \\
 & (b \wedge c \wedge \neg d \wedge x_3) \vee (b \wedge c \wedge \neg d \wedge \neg c) \vee & \\
 & (b \wedge d \wedge x_3) \vee (b \wedge d \wedge \neg c) & \\
= & (a \wedge \neg b \wedge c \wedge \neg d \wedge x_3) \vee (a \wedge \neg b \wedge d \wedge x_3) \vee & \mid x_3 = (\neg a \wedge c) \\
 & (a \wedge \neg b \wedge d \wedge \neg c) \vee (b \wedge c \wedge \neg d \wedge x_3) \vee & \\
 & (b \wedge d \wedge x_3) \vee (b \wedge d \wedge \neg c) & \\
= & (a \wedge \neg b \wedge c \wedge \neg d \wedge \neg a \wedge c) \vee (a \wedge \neg b \wedge d \wedge \neg a \wedge c) \vee & \mid p \wedge p = p, \\
 & (a \wedge \neg b \wedge d \wedge \neg c) \vee (b \wedge c \wedge \neg d \wedge \neg a \wedge c) \vee & \mid p \wedge \neg p = F \\
 & (b \wedge d \wedge \neg a \wedge c) \vee (b \wedge d \wedge \neg c) & \\
= & (a \wedge \neg b \wedge \neg c \wedge d) \vee (\neg a \wedge b \wedge c \wedge d) \vee & \\
 & (\neg a \wedge b \wedge c \wedge \neg d) \vee (b \wedge \neg c \wedge d) &
\end{array}
$$

A.5. Summary for Verification of Examples

BCF from example 1:
$(a \land \neg c \land d) \lor (\neg a \land b \land c) \lor (\neg a \land b \land d) \lor (b \land \neg c \land d)$

DNF from example 1 for theorem 4:
$(a \land \neg c \land d) \lor (\neg a \land b \land c) \lor (\neg a \land b \land \neg c \land d)$

DNF from example 2 for theorem 4:
$(a \land \neg b \land \neg c \land d) \lor (\neg a \land b \land c \land d) \lor (\neg a \land b \land c \land \neg d) \lor (b \land \neg c \land d)$

Solutions for Examples:

$$
\begin{array}{l}
(\ \ a \land \ \ b \land \neg c \land \ \ d) \lor \\
(\ \ a \land \neg b \land \neg c \land \ \ d) \lor \\
(\neg a \land \ \ b \land \ \ c \land \ \ d) \lor \\
(\neg a \land \ \ b \land \ \ c \land \neg d) \lor \\
(\neg a \land \ \ b \land \neg c \land \ \ d)
\end{array}
$$

Appendix B. Detailed Summary of Experiments

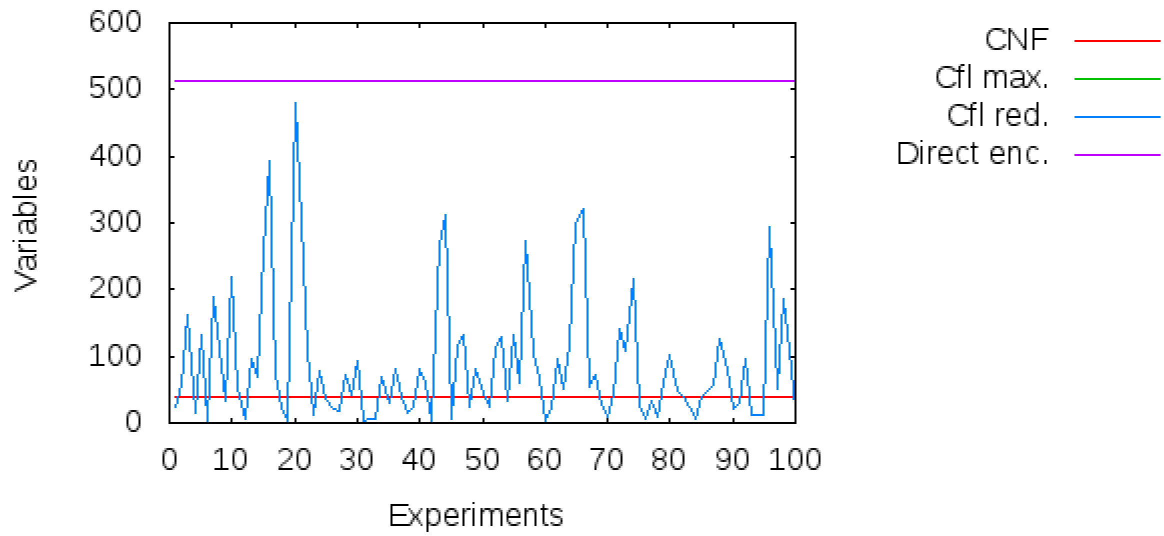
Experiments with 100 randomly generated 3-CNF formulas (genAlea, 2004) with 40 variables and 171 clauses[ws-exp] were conducted and all problems except one were solved trivially in polynomial time with a PDE based algorithm alone.

The original encoding is denoted as “CNF”. The problem was further re-encoded as selection problem in direct encoding without conflict maximization, denoted as “Direct enc.”.

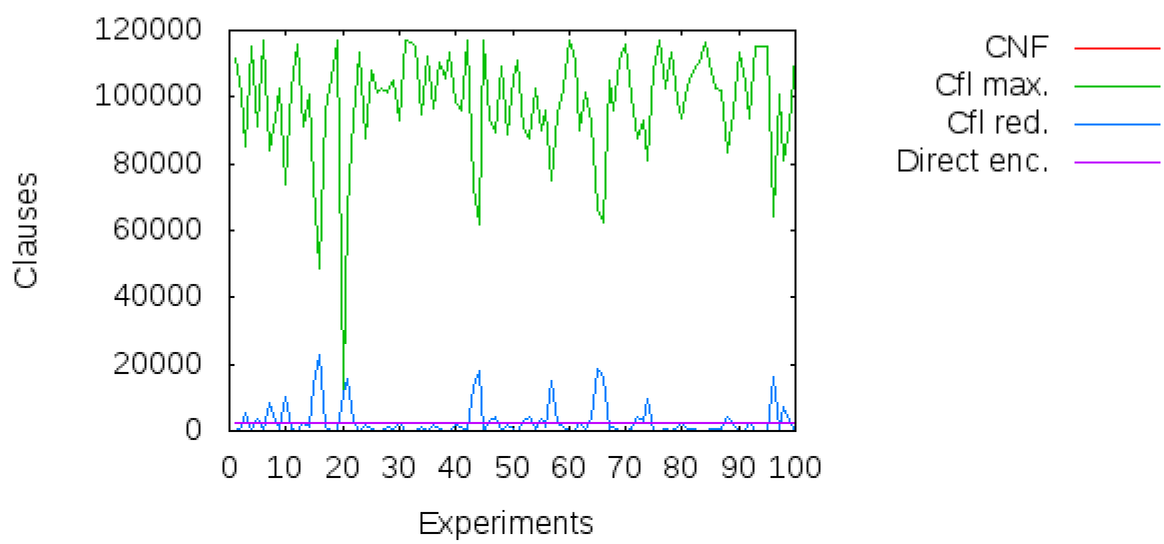
Two versions of conflict maximization were produced. The first, “Cfl max.”, was re-encoded with direct encoding and without redundancy removal. The second, “Cfl red.”, was re-encoded with direct encoding after redundancy removal.

The common parameters for all solvers are the number of variables and clauses of each experiment.

The number of variables is necessarily constant for “CNF” (40), “Direct enc.” (513), and “Cfl max.” (513). Since redundancy removal produces varying results, the number of variables necessarily varies for “Cfl red.” (avg. 86).



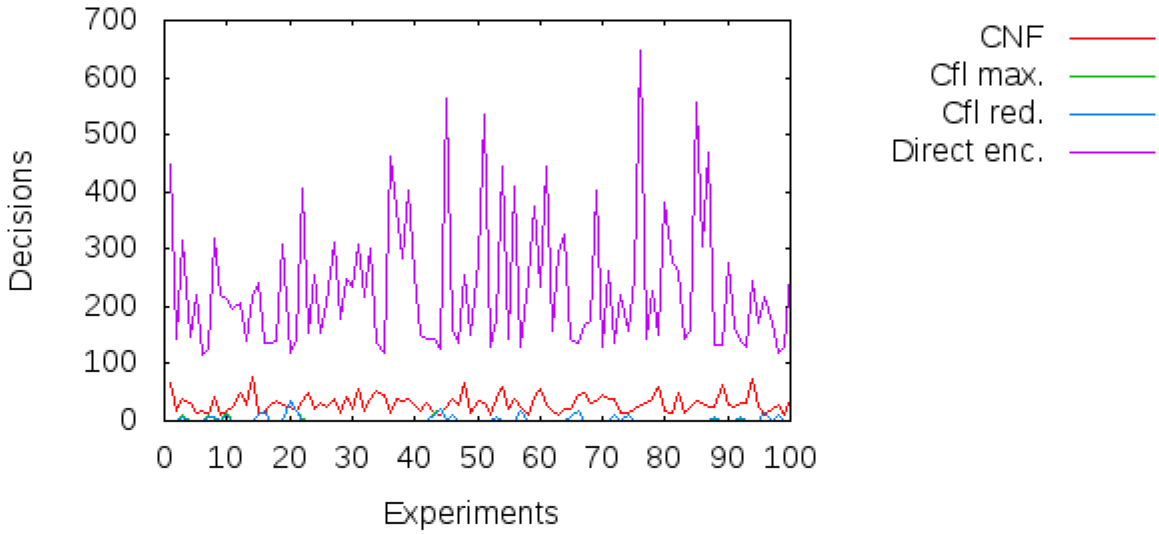
The number of clauses is constant for CNF (171). The number of clauses for the versions in direct encoding depends on the number of conflicting literals between clauses. Therefore it is necessarily higher than for CNF. The average number of clauses for “Cfl max.” is 97905, for “Cfl red.” 2934, and for “Direct enc.” 2309.



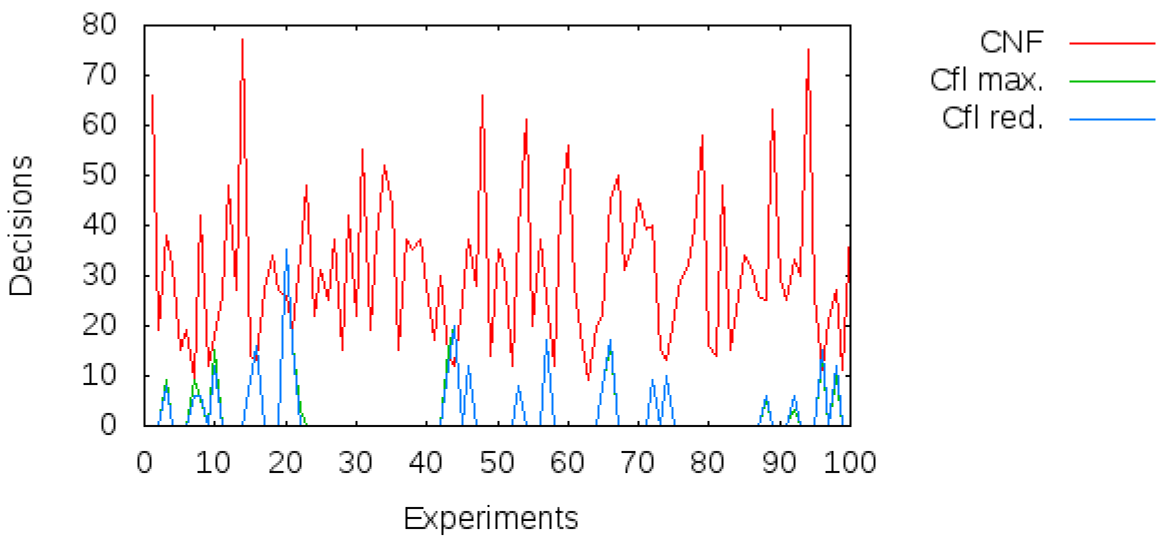
The raw number of decisions is used as a measure how hard the problem appears to a CDCL SAT-solver (lingeling).

CNF			Direct enc.			Cfl max.			Cfl red.		
Var	Cls	Dec/a	Var	Cls/a	Dec/a	Var	Cls/a	Dec/a	Var/a	Cls/a	Dec/a
40	171	30	513	2309	236	513	97905	2	86	2934	2

Solver lingeling, Var = variables, Cls = clauses, Dec = Decisions, /a = average



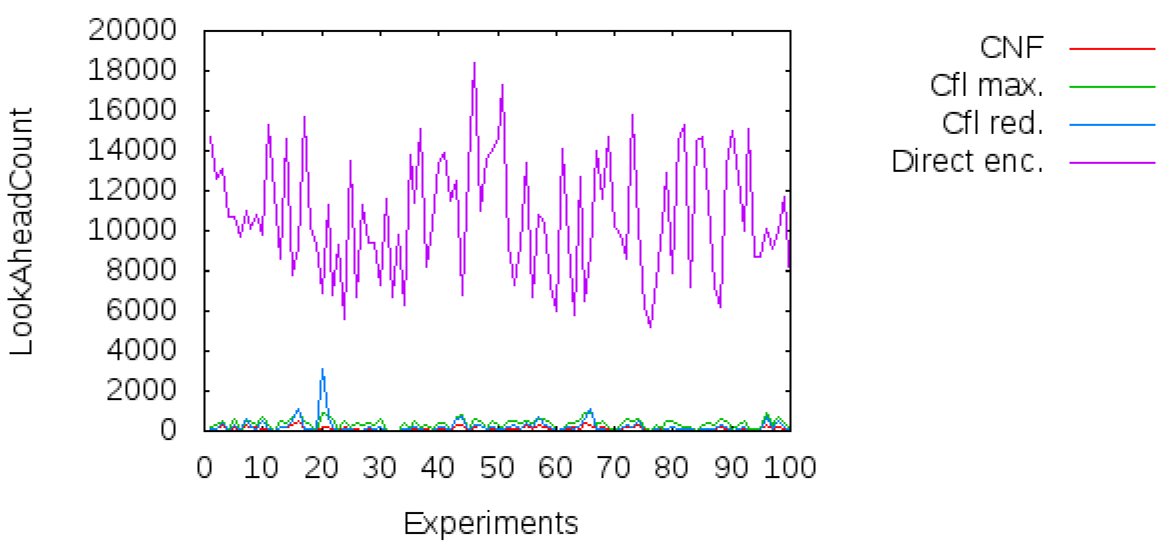
Leaving out "Direct enc." provides a more detailed view of the CNF and conflict maximization decisions.



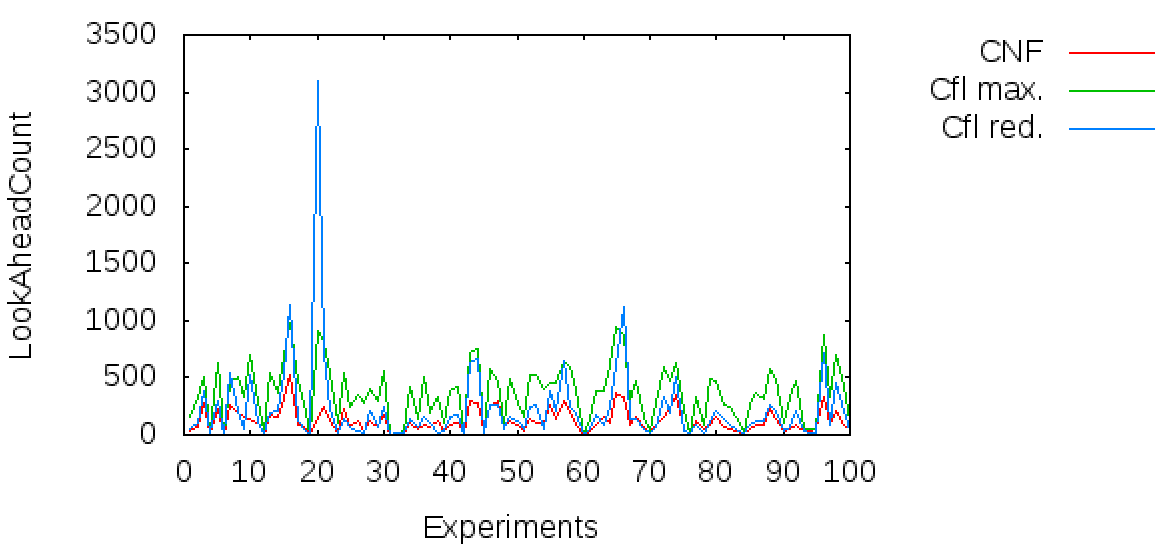
For the lookahead solver march_rw, the LookAheadCount is used as measure for the problem hardness.

CNF			Direct enc.			Cfl max.			Cfl red.		
Var	Cls	LA/a	Var	Cls/a	LA/a	Var	Cls/a	LA/a	Var/a	Cls/a	LA/a
40	171	122	513	2309	10720	513	97905	365	86	2934	220

Solver march_rw, Var = variables, Cls = clauses, LA = LookAheadCount, /a = average



More detailed view without “Direct enc.”:



References

[w3s] Various authors, Boolean satisfiability problem — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/wiki/Boolean_satisfiability_problem#3-satisfiability`, accessed 2013-12-03.

[wfc] Various authors, Problem of future contingents, — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/wiki/Future_contingent`, accessed 2013-12-07.

[bcf] Various authors, Blake canonical form — Wikipedia, the free encyclopedia. `https://en.wikipedia.org/wiki/Blake_canonical_form`, accessed 2013-12-02.

[BLAKE] Blake, Archie, Canonical expressions in Boolean algebra, University of Chicago, 1938.

[BROWN] Brown, Frank Markham, Boolean Reasoning: The Logic of Boolean Equations, Kluwer Academic Publishers, Boston, 1990. Second edition, Dover Publications, Mineola, 2003. zbMATH review.

[HOS] A. Biere, M. Heule, H. van Maaren, T. Walsh, Handbook of Satisfiability: Volume 185 Frontiers in Artificial Intelligence and Applications, IOS Press Amsterdam, The Netherlands, 2009, ISBN:1586039296 9781586039295

[MOUNT] David M. Mount, CMSC 451 Lecture Notes, Fall 2012, `http://www.cs.umd.edu/class/fall2012/cmsc451/Lects/cmsc451-lects.pdf`, accessed 2013-10-03.

[ws-exp] Wolfgang Scherer, CDCL and Direct Encoding. `http://sw-amt.ws/satoku/doc/doc-experiments/README.html`, accessed 2013-12-07.