On Conversions from CNF to ANF

Jan Horáček   Martin Kreuzer

Faculty of Informatics and Mathematics
University of Passau, Germany

Jan.Horacek@uni-passau.de
Martin.Kreuzer@uni-passau.de

Background

ANF is "XOR of ANDs"

- indeterminates $x_1, \dots, x_n$
- $\mathbb{B}_n = \mathbb{F}_2[x_1, \dots, x_n]/F$, $F = \langle x_1^2 + x_1, \dots, x_n^2 + x_n \rangle$
- squarefree support

Set of $\mathbb{F}_2$-rational zeros

- $S = \{f_1, \dots, f_s\} \subseteq \mathbb{B}_n$
- $\mathcal{Z}(S) = \{a \in \mathbb{F}_2^n \mid f(a) = 0 \text{ for all } f \in S\}$

Algebraic solvers

- the Bool. Gröbner Basis Alg.
- the Bool. Border Basis Alg.
- the XL/XSL, ElimLin, ...

CNF is "AND of ORs"

- logical variables $X_1, \dots, X_n$
- $C = \{\{L_{1,1}, \dots, L_{1,n_1}\}, \dots, \{L_{k,1}, \dots, L_{k,n_k}\}\}$ corresponds to $\phi = (L_{1,1} \vee \dots \vee L_{1,n_1}) \wedge \dots \wedge (L_{k,1} \vee \dots \vee L_{k,n_k})$

Set of satisfying assignments

- `True` $\equiv 1$ and `False` $\equiv 0$
- $\mathrm{SAT}(C) = \{a \in \{0,1\}^n \mid C(a) \text{ evaluates to } 1\}$

SAT solvers

- DPLL
- CDCL, ...

Representations

Algebraic/logical representation

Let $S \subseteq \mathbb{B}_n$ be a set of Boolean polynomials and C a set of clauses in the logical variables $X_1, \ldots, X_n$. We say that C is a **logical representation** of S resp. S is an **algebraic representation** of C if and only if $\mathrm{SAT}(C) = \mathcal{Z}(S)$.

$$
\begin{tikzcd}
 & \boxed{\text{“Problem”}} \arrow[dl] \arrow[dr] & \\
\boxed{\text{BPSS encoding}} \arrow[rr, leftrightarrow] & & \boxed{\text{SAT encoding}}
\end{tikzcd}
$$

Standard CNF to ANF conversion

Algorithm 1 (Standard CNF to ANF Conversion)
Input: A set of clauses C in logical variables $X_1, \ldots, X_n$.
Output: A set $S \subseteq \mathbb{B}_n$ such that S is an algebraic representation of C.

$S := \emptyset$
foreach c **in** C **do**
    $f := 1$
    foreach L **in** c **do**
        if $L = X_i$ is positive **then**
            $f := f \cdot (x_i + 1)$
        else if $L = \bar{X}_i$ is negative **then**
            $f := f \cdot (x_i)$
        end if
    end foreach
    $S := S \cup \{f\}$
end foreach
return S

Standard CNF to ANF conversion

Example

$$\begin{aligned}
\{X_1, X_2\} &\rightarrow x_1x_2 + x_1 + x_2 + 1\\
\{\bar{X}_1, X_2, X_3\} &\rightarrow x_1x_2x_3 + x_1x_2 + x_1x_3 + x_1\\
\{X_4, X_5\} &\rightarrow x_4x_5 + x_4 + x_5 + 1\\
\{X_1, \bar{X}_2, X_3\} &\rightarrow x_1x_2x_3 + x_1x_2 + x_2x_3 + x_1\\
\{\bar{X}_1, \bar{X}_2, \bar{X}_3\} &\rightarrow x_1x_2x_3\\
\{X_4, \bar{X}_5\} &\rightarrow x_4x_5 + x_5
\end{aligned}$$

Too many polynomials ...
... of high degree!

Building m-Blocks

Definition

(a) The set of variables X_i such that X_i or $\bar{X}_i$ is contained in one of the clauses of C is denoted by $\mathrm{var}(C)$ and is called the **set of variables** of C.

(b) We say $c \in C$ has **positive** (resp. **negative**) **sign** if the number of negative literals is an even (resp. odd) number.

(c) We define the **length of a clause** $c \in C$ as the cardinality $\#c$.

(d) Let $c, c' \in C$. A number $m \geq 1$ such that $\#\big(\mathrm{var}(c) \cap \mathrm{var}(c')\big) \geq m$ is called an **overlapping number** of c and c'.

Building m-Blocks

Algorithm 2 (Building m-Blocks)
Input: A set of clauses C, an overlapping number $m \in \mathbb{N}$.
Output: A set of subsets $\mathcal{B}$ of C and a subset T of C such that for $B \in \mathcal{B}$ with $\#B \geq 2$ and for every $b \in B$, there exists an element $b' \in B \setminus \{b\}$ with the property that m is an overlapping number for b and b', and such that $\left(\bigcup_{B \in \mathcal{B}} B\right) \cup T = C$ and every clause in T contains less than m literals.

foreach c **in** C **do**
    $B_c := \left\{c' \in C \mid \#\left(\mathrm{var}(c) \cap \mathrm{var}(c')\right) \geq m\right\}$
end foreach
$\mathcal{B}' := \{B_c \mid c \in C,\ B_c \neq \emptyset\}$
Let $\mathcal{B}$ be the set of maximal elements of $\mathcal{B}'$ w.r.t. inclusion.
$T := C \setminus \bigcup_{c \in C} B_c$
return $(\mathcal{B}, T)$

Building m-Blocks

Example: $m = 2$

$$
\begin{array}{l}
\{X_1, X_2\} \\
\{\bar{X}_1, X_2, X_3\} \\
\{X_4, X_5\} \\
\{X_1, \bar{X}_2, X_3\} \\
\{\bar{X}_1, \bar{X}_2, \bar{X}_3\} \\
\{X_4, \bar{X}_5\}
\end{array}
\rightarrow
\left[\begin{array}{l}
\{X_1, X_2\} \\
\{\bar{X}_1, X_2, X_3\} \\
\{X_1, \bar{X}_2, X_3\} \\
\{\bar{X}_1, \bar{X}_2, \bar{X}_3\}
\end{array}\right],
\left[\begin{array}{l}
\{X_4, X_5\} \\
\{X_4, \bar{X}_5\}
\end{array}\right]
$$

Proposition

The output of Algorithm 2 is uniquely determined.

Blockwise CNF to ANF Conversion

Algorithm 3 (Blockwise CNF to ANF Conversion)
Input: A set of clauses C in logical variables $X_1, \dots, X_n$, a degree compatible term ordering σ, and an overlapping number $m \in \mathbb{N}$.
Output: A set $S_{\sigma,m} \subseteq \mathbb{B}_n$ such that $S_{\sigma,m}$ is an algebraic representation of C.
Requires: Algorithm 1 and 2, a reduced Boolean Gröbner basis algorithm.

```
S' := ∅
Using Algorithm2(C, m), compute a pair (𝓑, T).
𝓑 := 𝓑 ∪ ⋃_{t∈T} {t}
foreach B in 𝓑 do
    Q := Algorithm1(B)
    Let G be the reduced Boolean σ-Gröbner basis of the ideal ⟨Q⟩, i.e., the
    reduced Boolean Gröbner basis with respect to the term ordering σ.
    S' := S' ∪ G
end foreach
Let S_{σ,m} be an LT_σ-interreduced F_2-basis of ⟨S'⟩_{F_2} such that its coefficient
matrix w.r.t. σ is in reduced row echelon form.
return S_{σ,m}
```

Blockwise CNF to ANF Conversion

Example: $m = 2$, $\sigma = \texttt{degrevlex}$

$$\left.\begin{array}{ll} \{X_1, X_2\} & \to x_1x_2 + x_1 + x_2 + 1 \\ \{\bar{X}_1, X_2, X_3\} & \to x_1x_2x_3 + x_1x_2 + x_1x_3 + x_1 \\ \{X_1, \bar{X}_2, X_3\} & \to x_1x_2x_3 + x_1x_2 + x_2x_3 + x_1 \\ \{\bar{X}_1, \bar{X}_2, \bar{X}_3\} & \to x_1x_2x_3 \end{array}\right] \to \begin{array}{l} x_2x_3 + x_2 + x_3 + 1 \\ x_1 + x_2 + x_3 \end{array}$$

$$\left.\begin{array}{ll} \{X_4, X_5\} & \to x_4x_5 + x_4 + x_5 + 1 \\ \{X_4, \bar{X}_5\} & \to x_4x_5 + x_5 \end{array}\right] \to \; x_4 + 1$$

Proposition

The output of Algorithm 3 is an algebraic representation of C and is uniquely determined by σ and m.

Conversion to linear polynomials

Definition

A set of clauses B, all of which have the same length ℓ, which consists of all possible clauses with either only positive or only negative sign is called a **complete signed set** of clauses.

Example

Let $B = \{\{\bar{X}_1, X_2, X_3\}, \{X_1, \bar{X}_2, X_3\}, \{X_1, X_2, \bar{X}_3\}\{\bar{X}_1, \bar{X}_2, \bar{X}_3\}\}$.
B is logical representation of $x_1 + x_2 + x_3$.

Remark

A complete signed set of clauses B of length ℓ consists of $2^{\ell-1}$ clauses. The set B is a logical representation of a linear polynomial.

Conversion to Linear Polynomials

Proposition

Let ϕ, ψ be propositional logic formulas. Then we have $\phi \equiv (\phi \vee \psi) \wedge (\phi \vee \bar{\psi})$.

Example

Let $B = \{\{X_1, X_2\}, \{\bar{X}_1, X_2, X_3\}, \{X_1, \bar{X}_2, X_3\}, \{\bar{X}_1, \bar{X}_2, \bar{X}_3\}\}$. The first clause in B is equivalent to the two clauses $\{X_1, X_2, X_3\}$, $\{X_1, X_2, \bar{X}_3\}$. In view of this, we have covered all four possible combinations for negative signed clauses of length 3. Indeed, Algorithm 3 converts B into $x_1 + x_2 + x_3$ and $x_2 x_3 + x_2 + x_3 + 1$.

Notes

- Algorithm 3 produces at least the same number of linear polynomials as the brute-force extending of the input clauses.
- Algorithm 3 performs block-wise simple logic reasoning (DPLL rules).
- Conversion back and forth may solve the system.

Experiments

Instance	CNF		Algorithm 1			Algorithm 3		
	#vars	#clauses	#lin	#quad	#high	#lin	#quad	#high
AES-10-1-2-4	1081	3361	1	1792	1568	337	2194	0
AES-10-1-4-4	1862	5824	1	2986	2837	604	3692	0
AES-10-2-2-4	2441	7841	1	3584	4256	947	4407	0
AES-10-2-4-4	4289	13904	1	5986	7917	1785	7353	0
AES-10-4-1-4	3149	10065	1	4800	5264	1149	5915	0
AES-2-1-2-4	237	701	1	360	340	70	453	0
AES-2-1-4-4	412	1218	1	598	619	132	746	0
AES-2-2-2-4	526	1615	1	716	898	201	882	0
AES-2-2-4-4	935	2883	1	1196	1686	375	1491	0
AES-2-4-1-4	669	2065	1	960	1104	241	1191	0
AES-2-4-2-4	1157	3652	1	1434	2217	501	1778	0
AES-2-4-4-4	2077	6596	1	2394	4201	957	2978	0
fact-12601-18701	745	3853	2	616	3235	291	1365	2
fact-151-283	271	1333	2	250	1081	115	471	2
fact-1777-491	403	2029	2	354	1673	166	713	2
fact-2393-3371	466	2380	2	400	1978	181	855	2
fact-373-929	328	1640	2	294	1344	131	593	2
fact-583909-600203	1280	6784	2	1010	5772	471	2428	2
fact-59-1009	328	1640	2	294	1344	149	544	2
fact-59441-62201	826	4312	2	676	3634	318	1527	2
fact-81551-100057	947	4945	2	770	4173	359	1767	2
fact-9601-10067	638	3296	2	532	2762	243	1188	2

Table: Number of converted polynomials by degree.

Thank you!