Towards an Optimal CNF Encoding of Boolean Cardinality Constraints

Carsten Sinz

Institute for Formal Models and Verification
Johannes Kepler University Linz, A-4040 Linz, Austria
carsten.sinz@jku.at

Abstract. We consider the problem of encoding Boolean cardinality constraints in conjunctive normal form (CNF). Boolean cardinality constraints are formulae expressing that at most (resp. at least) k out of n propositional variables are true. We give two novel encodings that improve upon existing results, one which requires only $7n$ clauses and $2n$ auxiliary variables, and another one demanding $\mathcal{O}(n \cdot k)$ clauses, but with the advantage that inconsistencies can be detected in linear time by unit propagation alone. Moreover, we prove a linear lower bound on the number of required clauses for any such encoding.

1 Introduction

Cardinality constraints—expressing numerical bounds on discrete quantities—arise frequently out of the encoding of real-world problems, e.g. in product configuration, radio frequency assignment or in reconstructing images form computer tomographs [1–3]. With considerable progress made over the last years in solving propositional satisfiability (SAT) instances, interest increased in tackling problems that include cardinality constraints using SAT-solvers. This, however, requires an encoding of cardinality constraints in the language of purely propositional logic or, more specifically, in conjunctive normal form (CNF), the predominant input language of modern SAT-solvers.

Boolean cardinality constraints put numerical restrictions on the number of propositional variables that are allowed to be true at the same time. A typical construct like $\leq k\,(x_1, \ldots, x_n)$ expresses that not more than k of the n variables $x_1, \ldots, x_n$ are allowed to be true. The traditional way of converting a constraint like $\leq k\,(x_1, \ldots, x_n)$ to purely propositional logic is by explicitly excluding all possible combinations of $k+1$ variables being simultaneously true, thus obtaining

$$\bigwedge_{\substack{M \subseteq \{1,\ldots,n\} \\ |M|=k+1}} \bigvee_{i \in M} \neg x_i \ ,$$

which requires $\binom{n}{k+1}$ clauses of length $k+1$. In the worst case of $k = \lceil n/2 \rceil - 1$ this amounts to $O(2^n/\sqrt{n/2})$ clauses. Better encodings are known [3, 4] and will be further improved in this paper. The general idea of these improved encodings is to build a count-and-compare hardware circuit and then translate this circuit to CNF. Besides the constraint's variables $x_1, \ldots, x_n$, additional *encoding variables* $s_1, \ldots, s_m$ will be

allowed. Formally, we are looking for an optimal encoding (typically minimizing the number of clauses) according to the following definition.

Definition 1. *A clause set E over the variables $V = \{x_1, \ldots, x_n, s_1, \ldots, s_m\}$ is a* clausal encoding of $\leq k\,(x_1, \ldots, x_n)$ *if for all assignments $\alpha : \{x_1, \ldots, x_n\} \to \mathbb{B}$ the following holds: there is an extension of α to $\alpha^* : V \to \mathbb{B}$ that is a model of E if and only if α is a model of $\leq k\,(x_1, \ldots, x_n)$, i.e. if and only if at most k of the variables x_i are set to 1 by α.*

2 Encoding Using a Sequential Counter

We now give a CNF encoding for cardinality constraints of the form $\leq k\,(x_1, \ldots, x_n)$ that is based on a sequential counter circuit. The circuit is shown in Fig. 1 and computes partial sums $s_i = \sum_{j=1}^{i} x_j$ for increasing values of i up to the final $i = n$. The values of all s_i's are represented as unary numbers. The overflow bits v_i are set to true if the partial sum s_i is greater than k.

To convert this circuit to CNF, we first build defining equations for the partial sum bits $s_{i,j}$ and the overflow bits v_i. We then simplify these equations, noting that all overflow bits have to be zero. The resulting equations are then converted to CNF, further noting that one direction of the equations can be dropped due to polarity considerations (the basic technique was introduced by Tseitin [5], and later re-invented and extended by different authors, e.g. Jackson and Sheridan [6]). We thus arrive at a set of clauses, call it $\mathrm{LT}^{n,k}_{\mathrm{SEQ}}$, defining the cardinality constraint $\leq k\,(x_1, \ldots, x_n)$ based on the sequential counter (for $k > 0$ and $n > 1$):

$$
\begin{array}{l}
(\neg x_1 \vee s_{1,1}) \\
(\neg s_{1,j}) \qquad \text{for } 1 < j \leq k \\
(\neg x_i \vee s_{i,1}) \\
(\neg s_{i-1,1} \vee s_{i,1}) \\
(\neg x_i \vee \neg s_{i-1,j-1} \vee s_{i,j}) \quad \Big\} \quad \text{for } 1 < j \leq k \quad \Big\} \text{for } 1 < i < n \\
(\neg s_{i-1,j} \vee s_{i,j}) \\
(\neg x_i \vee \neg s_{i-1,k}) \\
(\neg x_n \vee \neg s_{n-1,k})
\end{array}
$$

$\mathrm{LT}^{n,k}_{\mathrm{SEQ}}$ consists of $2nk + n - 3k - 1$ clauses and requires $(n-1) \cdot k$ auxiliary variables for the encoding. Due to its practical importance, we explicitly give the clause set $\mathrm{LT}^{n,1}_{\mathrm{SEQ}}$ (for the case $k = 1$, as a formula):

$$
(\neg x_1 \vee s_{1,1}) \wedge (\neg x_n \vee \neg s_{n-1,1}) \wedge \bigwedge_{1<i<n} \Big((\neg x_i \vee s_{i,1}) \wedge (\neg s_{i-1,1} \vee s_{i,1}) \wedge (\neg x_i \vee \neg s_{i-1,1}) \Big)
$$

This clause set consists of $3n - 4$ clauses (and $n - 1$ additional encoding variables) and is thus—with regard to the number of clauses—superior to the naïve encoding for all $n > 5$. The following theorem summarizes our results.[1].

[1] Due to space limitations we do not give proofs here. They can be found, however, on the Web at http://www-sr.informatik.uni-tuebingen.de/~sinz/CardConstraints

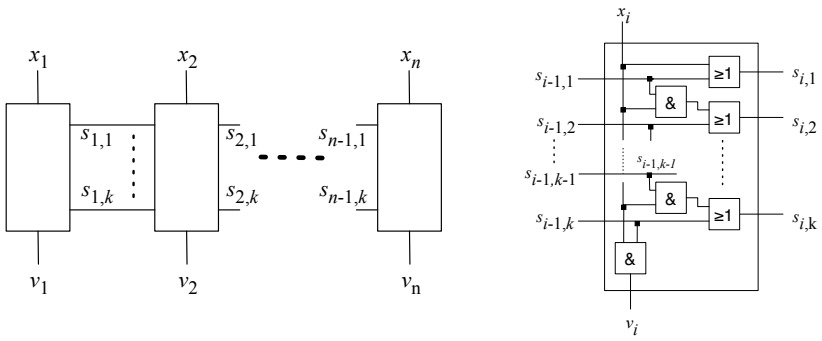

Fig. 1. Left: Circuit for computing $\leq k\,(x_1, \ldots, x_n)$. $s_{i,j}$ denotes the j-th digit of the i-th partial sum s_i in unary representation; variables v_i are overflow bits, indicating that the i-th partial sum is greater than k. **Right:** Sub-circuit for computing a partial sum s_i in unary representation.

Theorem 1. $\mathrm{LT}^{n,k}_{\mathrm{SEQ}}$ *is a clausal encoding of* $\leq k\,(x_1, \ldots, x_n)$ *requiring* $\mathcal{O}(n \cdot k)$ *clauses and* $\mathcal{O}(n \cdot k)$ *auxiliary variables.*

The encoding $\mathrm{LT}^{n,k}_{\mathrm{SEQ}}$ also fulfills the *efficiency condition* given by Bailleux and Boufkhad [3]. If more than k variables are set to true (which violates the cardinality constraint $\leq k\,(x_1, \ldots, x_n)$), this can be detected by unit propagation alone, i.e. by a linear time decision procedure. Moreover, for a partial assignment that sets k of the variables x_i to true, the value of all other x_i's can be derived by unit propagation.

3 Encoding Using a Parallel Counter

The second encoding we present is based on a parallel counter circuit designed by Muller and Preparata [7]. Their counter (shown in Fig. 2) recursively splits the input bits x_i into two halves, and counts the number of inputs that are set to true in each half. The results—represented as binary numbers—are then added using a standard m-bit binary adder. In order to obtain a circuit for cardinality constraints based on this counter, the output bits of the counter are handed on to a subsequent comparator which checks whether or not the counter value is less than k. (The comparator is not shown in Fig. 2.)

Parallel Counter Circuit. The parallel counter consists of $n - \lfloor \log n \rfloor - 1$ full-adders and at most $\lfloor \log n \rfloor$ half-adders, as was shown by Muller and Preparata ('log' denoting the *logarithmus dualis*). The encoding of each half-adder and full-adder is based on the well-known equations for these circuits. We finally obtain three clauses $\{(a \vee \neg b \vee s_{\text{out}}), (\neg a \vee \neg b \vee c_{\text{out}}), (\neg a \vee b \vee s_{\text{out}})\}$ for each half-adder (computing $a \oplus b$) and seven clauses

$$\begin{array}{lll} (a \vee b \vee \neg c \vee s_{\text{out}}) & (\neg a \vee b \vee c \vee s_{\text{out}}) & (\neg a \vee \neg b \vee c_{\text{out}}) \\ (a \vee \neg b \vee c \vee s_{\text{out}}) & (\neg a \vee \neg b \vee \neg c \vee s_{\text{out}}) & (\neg a \vee \neg c \vee c_{\text{out}}) \\ & & (\neg b \vee \neg c \vee c_{\text{out}}) \end{array}$$

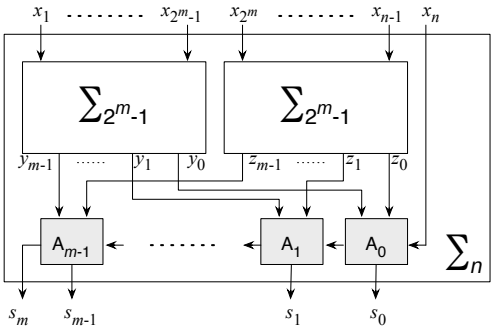

Fig. 2. Parallel counter according to Muller and Preparata [7] for recursively computing the binary number of inputs x_i that are set to true. The sub-circuits A_j are 1-bit-(full-)adders.

for each full-adder (computing $a \oplus b \oplus c$) , summing up to at most $7n - 4\lfloor \log n \rfloor - 7$ clauses. Here again, only implications, but not full equivalences, have to be encoded. This is due to the polarity-based simplification technique already used above and the observation that only ones have to be propagated to the outputs, but not zeroes. Auxiliary variables are needed for the sum and carry bits of the half- and full-adders, we therefore need at most $2 \cdot (n-1)$ additional encoding variables.

Comparator Circuit. The comparator circuit has to make sure that the result of the binary counter $(s_m s_{m-1} \dots s_0)$ is not greater than k. For building this binary comparator we assume that the constraint's limit k is given as an $(m+1)$-bit binary number, say $k = k_m \dots k_0$. We can then easily give recursive equations to generate the clauses for the $(m+1)$-bit comparator:

$$L(k_0) = \begin{cases} \{\{\neg s_0\}\} & \text{if } k_0 = 0 \\ \emptyset & \text{if } k_0 = 1 \end{cases}$$

$$L(k_i \dots k_0) = \begin{cases} \{\{\neg s_i\}\} \cup L(k_{i-1} \dots k_0) & \text{if } k_i = 0 \\ \{\{\neg s_i\}\} \otimes L(k_{i-1} \dots k_0) & \text{if } k_i = 1 \end{cases}$$

Here $\otimes$ denotes clause distribution, i.e. $A \otimes B = \{x \cup y \mid x \in A, y \in B\}$ for sets of clauses A, B. With this definition, $L(k_m \dots k_0)$ is the clause set that ensures that the counter's output is less than k. It contains at most $m+1$ clauses. Denoting the combined clause set (parallel counter and comparator) by $\mathrm{LT}^{n,k}_{\mathrm{PAR}}$, we obtain the following theorem (using $m = \lfloor \log n \rfloor$).

Theorem 2. $\mathrm{LT}^{n,k}_{\mathrm{PAR}}$ *is a clausal encoding of* $\leq k\,(x_1, \dots, x_n)$ *requiring at most* $7n - 3\lfloor \log n \rfloor - 6$ *clauses and* $2n - 2$ *auxiliary variables.*

4 Comparison, Lower Bound and Conclusion

Criteria for assessing the clausal encodings are (i) the number of clauses required; (ii) the number of additional propositional variables required; and (iii) the time needed to

decide the encoding. In our comparison we have included the naïve encoding (mentioned in the introduction) and the encodings of Bailleux&Boufkhad and Warners.

Table 1. Comparison of different encodings for $\leq k\,(x_1, \ldots, x_n)$.

Encoding	#clauses	#aux. vars	decided
Naïve	$\binom{n}{k+1}$	0	immediately
Sequential unary counter ($\mathrm{LT}^{n,k}_{\mathrm{SEQ}}$)	$\mathcal{O}(n \cdot k)$	$\mathcal{O}(n \cdot k)$	by unit prop.
Parallel binary counter ($\mathrm{LT}^{n,k}_{\mathrm{PAR}}$)	$7n - 3\lfloor \log n \rfloor - 6$	$2n - 2$	by search
Bailleux & Boufkhad [3]	$\mathcal{O}(n^2)$	$\mathcal{O}(n \cdot \log n)$	by unit prop.
Warners [4]	$8n$	$2n$	by search

With respect to the number of clauses required, our encoding $\mathrm{LT}^{n,k}_{\mathrm{PAR}}$ is the best, as can be seen from Table 1; however, it requires search to check whether the constraint is fulfilled or not. Among the encodings requiring no search is that of Bailleux&Boufkhad and our $\mathrm{LT}^{n,k}_{\mathrm{SEQ}}$ encoding. The latter performs better for small values of k, whereas the former is better for large bounds.

Considering optimality of clausal encodings for $\leq k\,(x_1, \ldots, x_n)$, we have shown elsewhere that for all $n \in \mathbb{N}$ and all k with $0 \leq k < n - 1$, each clausal (CNF) encoding of $\leq k\,(x_1, \ldots, x_n)$ requires at least n clauses. Such a proof touches the realm of Boolean function complexity [8]. It might be an interesting topic for future research to see in how far results from this field are transferrable to the area of minimal clausal encodings. We think that looking for improved lower bounds is worthwhile and still expect much room for improvement here. Moreover, an experimental evaluation of the different encodings should be of great practical value.

References

1. Küchlin, W., Sinz, C.: Proving consistency assertions for automotive product data management. J. Automated Reasoning **24** (2000) 145–163
2. Cabon, B., de Givry, S., Lobjois, L., Schiex, T., Warners, J.P.: Radio link frequency assignment. Constraints **4** (1999) 79–89
3. Bailleux, O., Boufkhad, Y.: Efficient CNF encoding of boolean cardinality constraints. In Rossi, F., ed.: 9th Intl. Conf. on Principles and Practice of Constraint Programming (CP 2003). Volume 2833 of Lecture Notes in Computer Science., Springer (2003) 108–122
4. Warners, J.P.: A linear-time transformation of linear inequalities into conjunctive normal form. Inf. Process. Lett. **68** (1998) 63–69
5. Tseitin, G.S.: On the complexity of derivation in propositional calculus. In Slisenko, A.O., ed.: Studies in Constructive Mathematics and Mathematical Logic. (1970) 115–125
6. Jackson, P., Sheridan, D.: The optimality of a fast CNF conversion and its use with SAT. Technical Report APES-82-2004, APES Research Group (2004) Available from http://www.dcs.st-and.ac.uk/ apes/apesreports.html.
7. Muller, D.E., Preparata, F.P.: Bounds to complexities of networks for sorting and for switching. J. ACM **22** (1975) 195–201
8. Wegener, I.: The Complexity of Boolean Functions. Wiley-Teubner (1987)