Finding Single Track Gray Codes with SAT

Norbert Manthey
Knowledge Representation and Reasoning Group
TU Dresden, Germany

***Abstract*—Binary Codes are used in channel encodings, but also for encoding positions that can be decoded by sensors. These binary codes can be constructed, however, a code can also be found by searching a solution for a given specification. The paper presents how to specify codes for gray codes and single track gray codes, similar to [1].**

I. Introduction

Gray codes [2] are binary codes where consecutive words have bit representations that differ in exactly one bit. These codes are useful to measure positions. Single track gray codes [3], [4] are special gray codes that furthermore enforce that the pattern that is formed by the first bits of all words appears again for all other bit positions and all words, just shifted by a few bits. Then, the word that encodes the current position can be read by reading only the first bits from a few selected words. These reads have to be placed such that the distance matches the offsets of the bit patterns.

This documents describes CNF encodings for gray and single track gray codes. Similar formula constructions have been discussed in [1]. Additionally to [1], we present show how certain symmetries of the input problem can be broken by adding further constraints.

II. Gray Codes

To construct a gray code G, the number of words m is given, as well as the length of each word in bits n. Then, for each pair of words in the code we enforce that the two words are not equal. Furthermore, for each consecutive pair of words we enforce that the bit representation is different for exactly one bit.

The used Boolean variables are denoted as follows: $w_{i,p}$ represents the p-th bit of the i-th word of the code, where $1 \leq i \leq m$ and $1 \leq p \leq n$.

A. Avoiding Duplicate Words

First, we have to require that two words are inequal, i.e. they have to be different in at least one bit:

$$\bigwedge_{1\leq i\leq m} \bigwedge_{i+1\leq j\leq m} \bigvee_{1\leq p\leq n} w_{i,p} \leftrightarrow \overline{w_{j,p}}$$

As converting this equation into CNF would result in a blow up, due to the disjunction in the conjunction, auxiliary variables are introduced. For each pair of words w_i and w_j, we introduce a set of propositional variables $x_{i,h,p}$, where p denotes the bit. Then, $x_{i,j,p} = \top$, if the two words w_i and w_j are different in bit p, i.e. $x_{i,j,p} \leftrightarrow (w_{i,p} \leftrightarrow \overline{w_{j,p}})$. Hence, we add to the formula the definition of all auxiliary variables $x_{i,j,p}$. Due to the symmetry of the equation, we encode only the cases where the first word has the smaller index, i.e. $i < j$:

$$\bigwedge_{1\leq i\leq m} \bigwedge_{i+1\leq j\leq m} \bigwedge_{1\leq p\leq n} (x_{i,j,p} \leftrightarrow (w_{i,p} \leftrightarrow \overline{w_{j,p}})) \qquad (1)$$

The formula that ensures that a model corresponds to a valid code with no duplicates words, can be derived from equation 1:

$$\bigwedge_{1\leq i\leq m} \bigwedge_{i+1\leq j\leq m} \bigvee_{1\leq p\leq n} x_{i,j,p} \qquad (2)$$

By combining the two formulas in equation 2 and equation 1, a satisfying assignment will result in a code that does not contain duplicate words, as each pair of words has to have at least on different bit (equation 1), and this relation is encoded for all pairs of words (equation 2).

B. Hamming Distance

In a gray code the bit distance between two consecutive words is exactly one. Translated to the CNF representation, exactly one bit of two consecutive words w_i and w_j has to be set different. This difference is already represented by auxiliary variables $x_{i,j,p}$. To ensure that exactly one bit is set different, for a pair of words exactly one variable $x_{i,j,p}$ for $1 \leq p \leq n$ has to be satisfied. To encode this relation, we use an auxiliary constraints EXO, which takes a set of literals as argument and returns the corresponding CNF formula.

The EXO constraint of a set of literals X is encoded as conjunction of an at-least constraints (ALO) and an at-most-one constraint (AMO). While ALO is encoded with a single clause of the literals of X, the AMO constraint is encoded based on the nested AMO encoding, which is based on recursive partitioning and introducing auxiliary variables as discussed in [5].

For a hamming distance of one for two words w_i and w_j the resulting formula is then $EXO(x_{i,j,1}, \ldots, x_{i,j,n})$. To encode this relation for all words of the code, the following formula is necessary:

$$EXO(x_{n,1,1}, \ldots, x_{n,1,n}) \bigwedge_{1\leq i<m} EXO(x_{i,i+1,1}, \ldots, x_{i,i+1,n}) \qquad (3)$$

The CNF in equation 3 ensures for each pair of consecutive words in the code, w_i and w_{i+1}, that the two words differ in exactly one bit. Note, the special case for w_n and w_1 is also encoded (left part of equation 3).

C. Avoiding Redundancy

A formula that encodes a gray code for m words with n bits is constructed as the union of the two above-mentioned formulas, . This formula contains redundant clauses: equation 3 ensures that consecutive words w_i and w_{i+1} differ in exactly one bit. Furthermore, equation 2 ensures that two words are different. For consecutive words this relation is already ensured, so that the resulting clauses can be dropped, resulting in the reduced formula:

$$\left(\bigwedge_{(3\leq j\leq m-1)} \bigvee_{(1\leq p\leq n)} x_{1,j,p}\right)\left(\bigwedge_{2\leq i\leq m} \bigwedge_{i+2\leq j\leq m} \bigvee_{1\leq p\leq n} x_{i,j,p}\right) \tag{4}$$

The left part of equation 4 encodes the special case for the first word of the code, which does not need to encode the difference for the second word, and the last word of the code. For all other words $(i > 1)$ of the code, it is sufficient to enforce difference to all non-neighboring words $(j > i + 1)$.

The CNF formula to generate a gray code for m words and n bits is the conjunction of equation 4 and equation 3, together with the definition of the auxiliary variables in equation 1.

III. Single Track Gray Codes

A single track gray code is a special gray code, where the pattern of the first bit of all words is repeated for all other bits: For all bit positions $2 \leq p \leq n$ there exists a k $(1 \leq k \leq m)$, such that for all word pairs i and j of the code $w_{i,p} = w_{j,(p+k)}$ holds, where $p + k$ is ensured to be in the range 1 to n, by subtracting n whenever necessary. This is ensured by the function

$$\mathtt{W}(x)\begin{cases} x & \text{if } x \leq m \\ x - m & \text{otherwise} \end{cases}$$

which ensures that a given number stays in the number of possible words (within the context of this paper).

A. Encoding Bit Pattern Offset

To encode the offsets in the patterns for the bits, we furthermore add a set of auxiliary variables. If the bit pattern from the first bit to bit q is set to k, then the auxiliary variable is equal to $\top$. Hence, for each bit position $(1 \leq q \leq n)$ an auxiliary variable $y_{q,k}$ is introduced for each possible offset $(1 \leq k \leq m)$. With the help of these variables, the offset for each bit is encoded as a one-out-of-N representation. Then, for each bit only a single offset is allowed, such that another EXO constraint has to be encoded:

$$\bigvee_{1\leq q\leq n} EXO(y_{q,1},\ldots,y_{q,m}) \tag{5}$$

The above constraint (equation 5) ensures that for each bit exactly one offset between 1 and m is chosen. To ensure that each bit has a different offset, another difference constraint is added.[1] As above, another auxiliary variable is added to encode this property: $z_{k,p,q} \leftrightarrow (y_{p,k} \leftrightarrow \overline{y_{q,k}})$. The formula that defines all auxiliary variables is given below:

$$\bigwedge_{1\leq p\leq n} \bigwedge_{p+1\leq q\leq n} \bigwedge_{1\leq k\leq m} z_{k,p,q} \leftrightarrow (y_{p,k} \leftrightarrow \overline{y_{q,k}}) \tag{6}$$

For a given offset k and a pair of bits p and q, $z_{k,p,q}$ is satisfied of the two two position of the encoded offset are different. Two offsets are different, if at least one of their propositional variables is mapped to a different value. Hence, we add the following formula:

$$\bigwedge_{1\leq p\leq n} \bigwedge_{p+1\leq q\leq n} \bigvee_{1\leq k\leq m} z_{k,p,q} \tag{7}$$

B. Applying Bit Pattern Offsets

With the formulas to encode a bit pattern offset, we can now apply the picked offset to ensure that the bits in the code words are set accordingly.

$$\bigwedge_{1\leq k\leq m} \bigwedge_{1\leq i\leq m} \bigwedge_{2\leq p\leq n} y_{p,k} \rightarrow \left(w_{i,1} \leftrightarrow w_{\mathtt{W}(i+k),p}\right) \tag{8}$$

The above equation 8 applies the offset to the bits of the code. For a given offset k for a bit position p, for all pairs of words i $(1 \leq i \leq m)$ and $\mathtt{W}(i + k)$ we ensure that the first bit of the first word has the same value as the p-th bit of the latter word. Note, the formula must select first bit position and the latter word, as those can be calculated from the other values.

C. Generating a Single Track Gray Code

A single track gray code for m words and n bits is a gray code that has shifted bit patterns. Hence, we conjoin the formulas from equation 4, equation 3 and equation 1 to obtain a gray code. The single track property is added by furthermore adding the formulas from equation 5, equation 6, equation 7 and equation 8. A satisfying assignment for this formula contains the gray code in the variables $w_{i,p}$, and furthermore gives the offsets for the bits. If no such code exists, then there is no satisfying assignment for the formula. A trivial unsatisfiable formula can be obtained by picking an odd number of words in the code, because there is no gray code with an odd number of words. Hence, only formulas for even values of m have been submitted.

IV. Symmetry Breaking

There are multiple symmetries in a gray code that can be broken during generation already. For example, the first word in the code can be an arbitrary word. Furthermore, an existing code can be written in two ways (forward or backward). Additionally, the offsets of the bit patterns could be arbitrarily ordered. We add the following constraints, to avoid the mentioned types of symmetries.

[1] After submitting the formulas to the SAT race, we noted that this difference is not necessary, as together with the constraints for a gray code this difference is implied from the other parts of the formula already.

A. Defining the Direction of the Code

We enforce the direction of the code, and the start of the code, by enforcing the first word in the code to be the binary representation of the number 0. Furthermore, the second word in the code will be the binary representation of the number 1. The resulting formula is equisatisfiable to the initial formula, as all constraints are still met. We just removed a few possible satisfying assignments by giving an initialization.

B. Ordering Offsets

By sorting the offsets of consecutive bit patterns, many possible solutions can be cut off. As reordering the bits in all words of a gray code equally results in another valid gray code, adding the ordering results in an equisatisfiable formula as well. Formally, we encode the following constraints:

$$\bigwedge_{1 \leq p < n} \text{offset}_p > \text{offset}_{p+1}.$$

Due to transitivity of the greater-than operator, we obtain the largest offset for the second bit of all words, and then the offsets strictly decrease. The CNF below create the desired ordering:

$$\bigwedge_{1 \leq p < n} \bigwedge_{1 \leq i \leq m} \bigwedge_{i \leq j \leq m} y_{p,i} \rightarrow \overline{y_{p+1,j}}. \tag{9}$$

This formula ensures that the offset $y_{p,i}$ is set to the value i for some bit p, then for the consecutive bit $p+1$ the offset cannot be set to any higher number. The implication is turned into a binary clause by negating the first literal.

V. The Formulas

We implemented a generator that produces the presented CNF formula for a gray code based on the number of words in a code m, and the number of bits per word n. Furthermore, options can be specified to furthermore enable the single track property, as well as the initialization and the enforced pattern ordering. The submitted formulas were constructed from the tool, where all options haven been enabled.

For a few pairs of m and n, solutions for the single track gray code are known to exist. With a fixed n, the smallest solution is $2n$, and the largest solution is $2^n - 2n$ [4]. From preliminary experiments, a few other known solutions seem to follow the pattern $m = 2n$, $m = 6n$ and $m = 10n$. For most other values, there does not exist a solution.

References

[1] I. Zinovik, D. Kroening, and Y. Chebiryak, "Computing binary combinatorial gray codes via exhaustive search with SAT solvers," *IEEE Transactions on Information Theory*, vol. 54, no. 4, pp. 1819–1823, 2008. [Online]. Available: http://dx.doi.org/10.1109/TIT.2008.917695

[2] C. Savage, "A survey of combinatorial gray codes," *SIAM Rev.*, vol. 39, no. 4, pp. 605–629, Dec. 1997. [Online]. Available: http://dx.doi.org/10.1137/S0036144595295272

[3] B. Spedding, "A position encoder," Patent, 1994.

[4] A. P. Hiltgen and K. G. Paterson, "Single-track circuit codes," *IEEE Transactions on Information Theory*, pp. 2587–2595, 2001.

[5] N. Manthey, M. J. Heule, and A. Biere, "Automated reencoding of Boolean formulas," in *Hardware and Software: Verification and Testing*, ser. Lecture Notes in Computer Science, A. Biere, A. Nahir, and T. Vos, Eds., vol. 7857. Springer Berlin Heidelberg, 2013, pp. 102–117. [Online]. Available: http://dx.doi.org/10.1007/978-3-642-39611-3_14