An Efficient SAT Encoding of Circuit Codes

Yury Chebiryak[1] Daniel Kroening[2]

[1]Computer Systems Institute, ETH Zurich, Switzerland

[2]Computing Laboratory, Oxford University, UK

IEEE International Symposium on Information Theory and its Applications, Auckland, New Zealand, 2008.

This research was supported in part by an award from IBM Research
and by ETH Research Grant TH-19 06-3.

References

Knuth05 Knuth, D. E. *The Art of Computer Programming, vol. 4, fascicle 2: Generating All Tuples and Permutations*, Addison-Wesley Professional, 2005

Chebiryak+08 Chebiryak, Y. and Kroening, D. *Towards a classification of Hamiltonian cycles in the 6-cube*, Journal on Satisfiability, Boolean Modeling and Computation, vol. 4, pp. 57-74, 2008

Suparta06 Suparta, I. N. *Counting sequences, Gray codes and lexicodes*, Ph.D. thesis, Delft University of Technology, 2006

Zinovik+07 Zinovik, I.; Kroening, D. & Chebiryak, Y. *An Algebraic Algorithm for the Identification of Glass Networks with Periodic Orbits Along Cyclic Attractors*, Procs. Algebraic Biology, pp. 140-154, 2007

Zinovik+08 Zinovik, I.; Kroening, D. & Chebiryak, Y. *Computing Binary Combinatorial Gray Codes via Exhaustive Search with SAT-solvers*, IEEE Transactions on Information Theory, vol. 54, pp. 1819-1823, 2008

Chinese Ring Puzzle

aka. The Devils Needle Puzzle aka. Cardan's rings

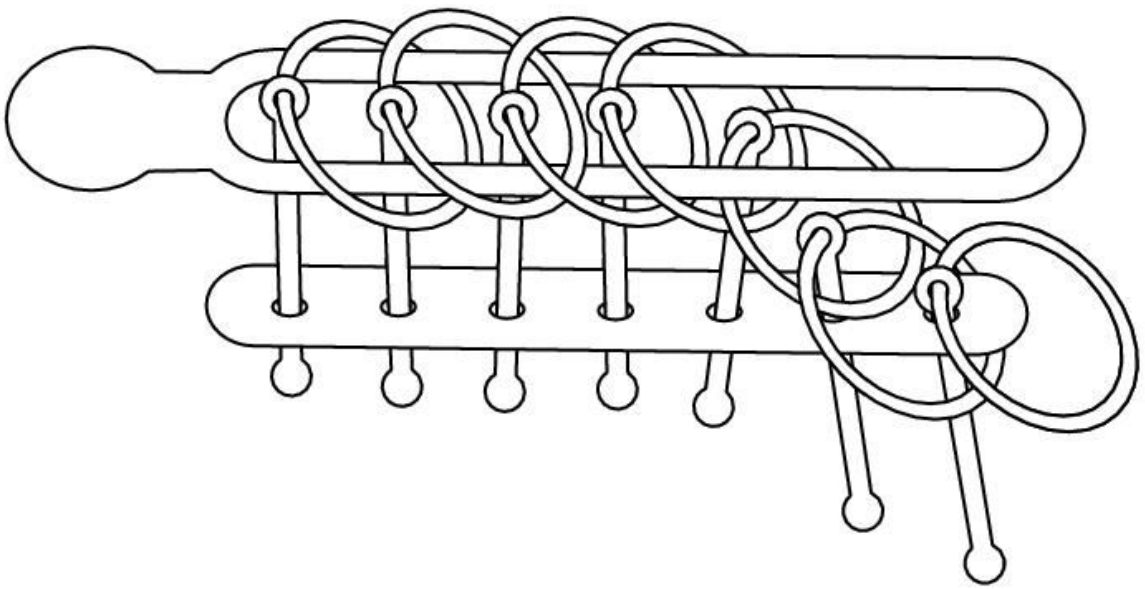


Figure adopted from [Knuth'05]

Empty the bar by

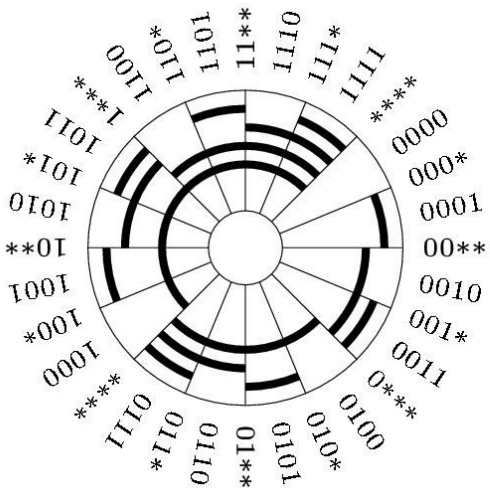
1. removing or replacing rightmost ring or
2. moving any ring if right ring is on bar and rest are off

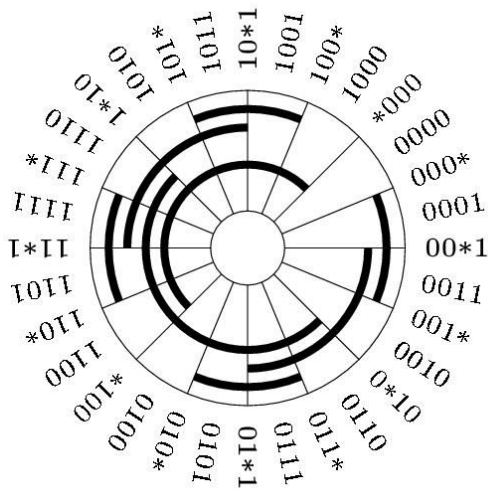
Binary Reflected Gray Code

Gray codes application:

- analog to digital information conversion
- error correction
- circuit testing
- signal encoding
- data compression
- diagnosis of multiprocessors
- computational biology (e.g. [Zinovik+07])

Figure adopted from [Knuth'05]





Combinatorial Gray Codes

- Binary Reflected Gray Code = Hamiltonian cycle
- Snake-in-the-box = induced path = chord-free path
- distance-preserving, **circuit codes** = generalized snakes
- single-track codes
- binary necklaces

Long codes are desired

Combinatorial explosion

New codes and their classifications reported in [Chebiryak+08], [Zinovik+08], [Zinovik+07]

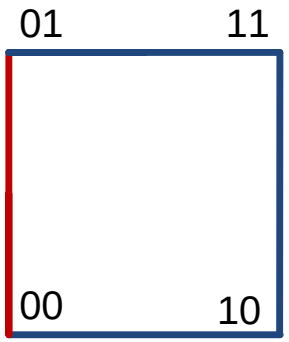
All (combinations of) these codes can be found using CODESAT!

Hypercube

$$Q_1 = K_2, \quad Q_n = K_2 \times Q_{n-1}$$

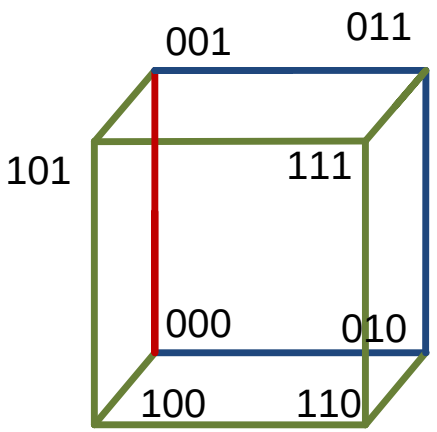
Hypercube

$$Q_1 = K_2, \quad Q_n = K_2 \times Q_{n-1}$$



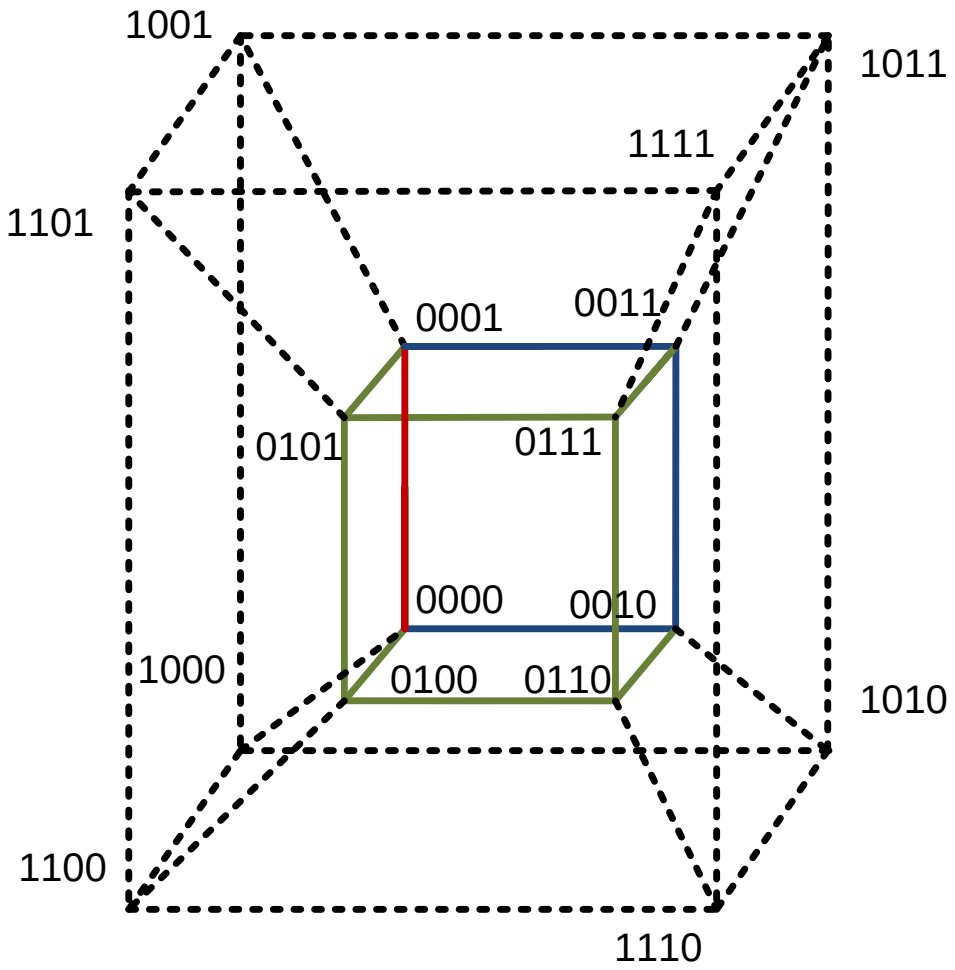
Hypercube

$$Q_1 = K_2, \quad Q_n = K_2 \times Q_{n-1}$$



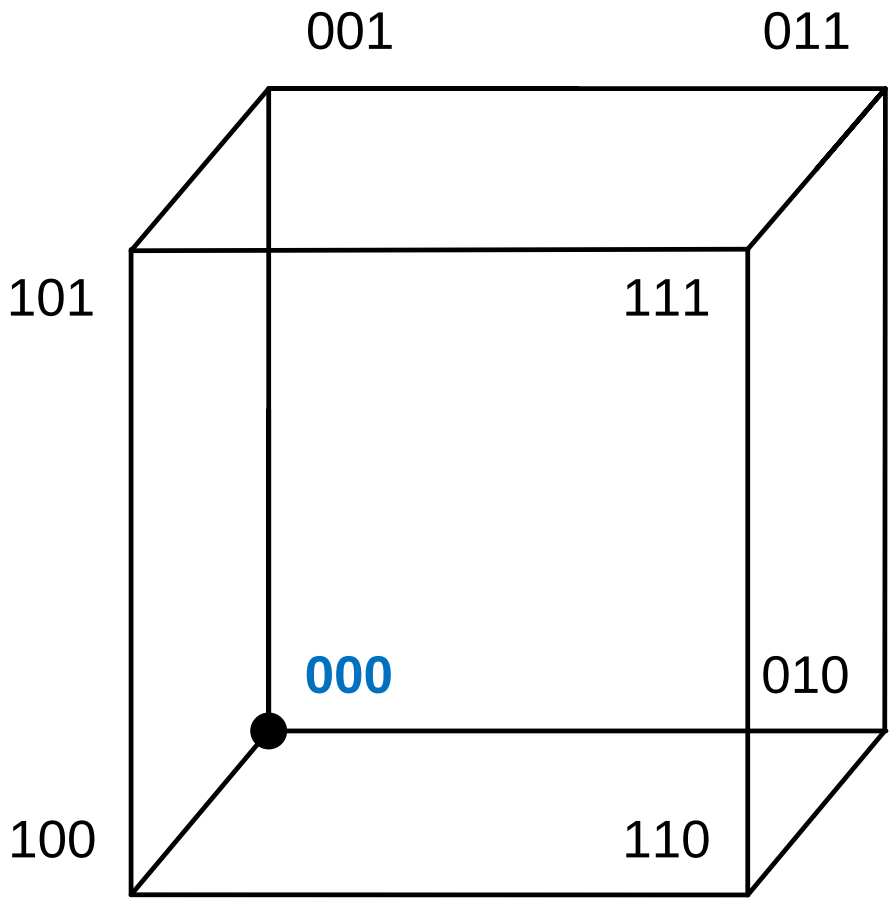
Hypercube

$$Q_1 = K_2, \quad Q_n = K_2 \times Q_{n-1}$$



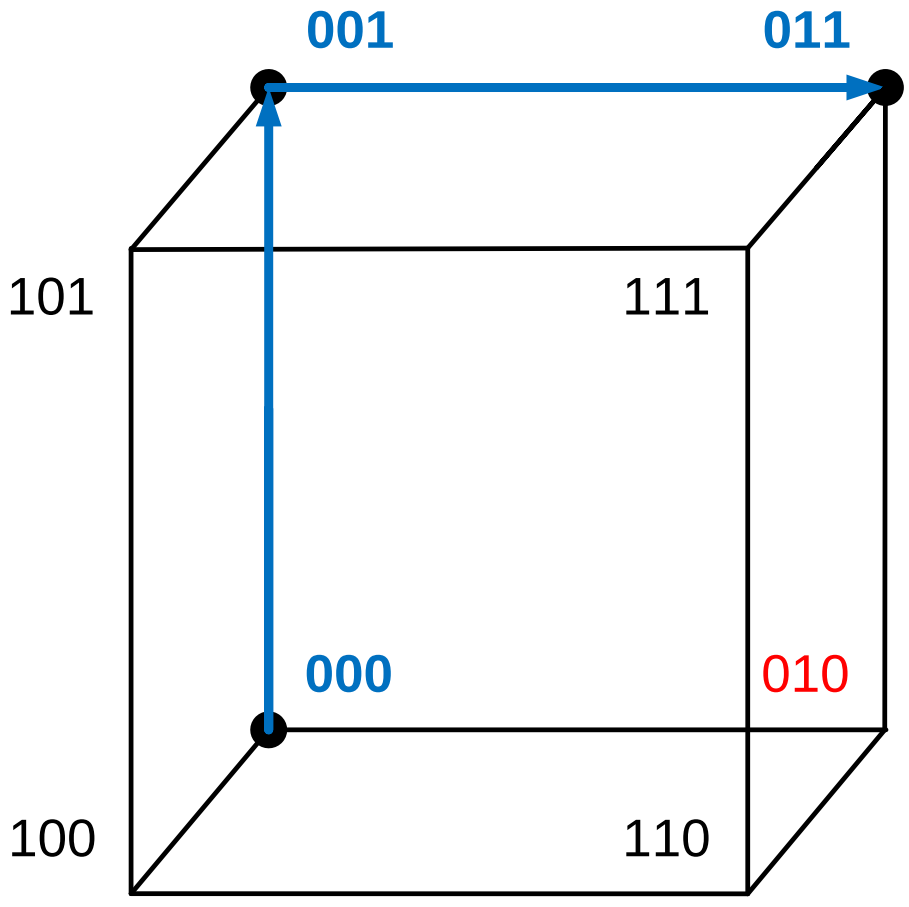
Coil-in-the-box: Example

Initial node: 000,



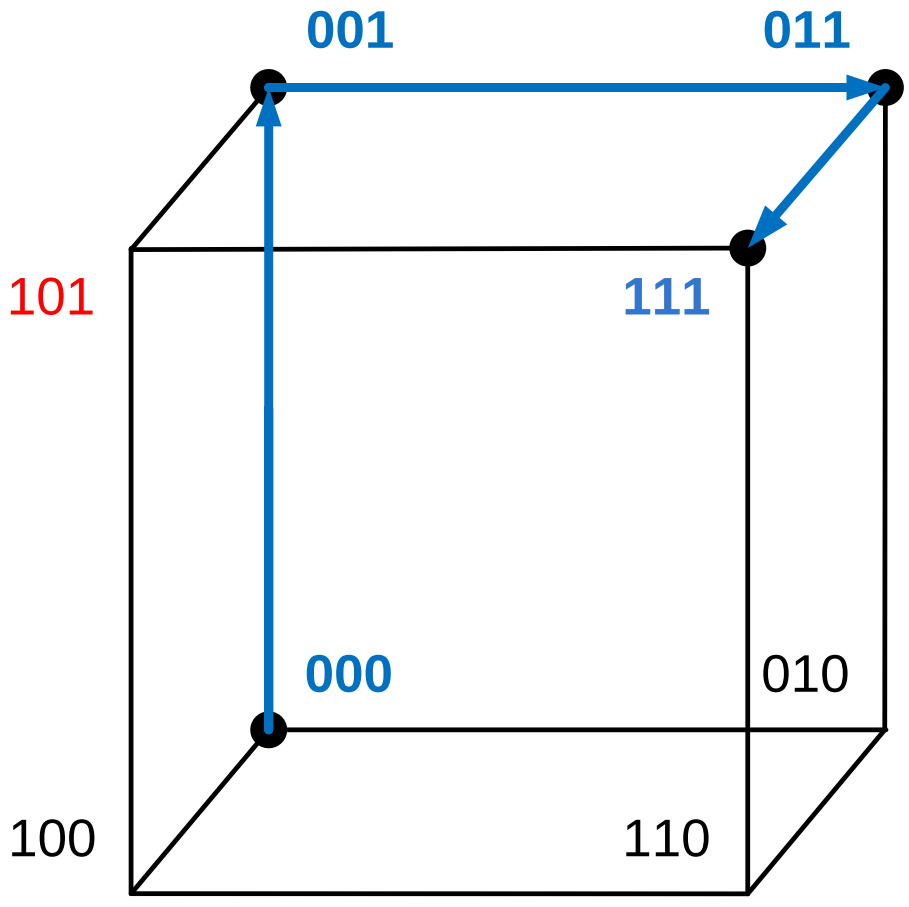
Coil-in-the-box: Example

Initial node: 000, Transition sequence: 1, 2



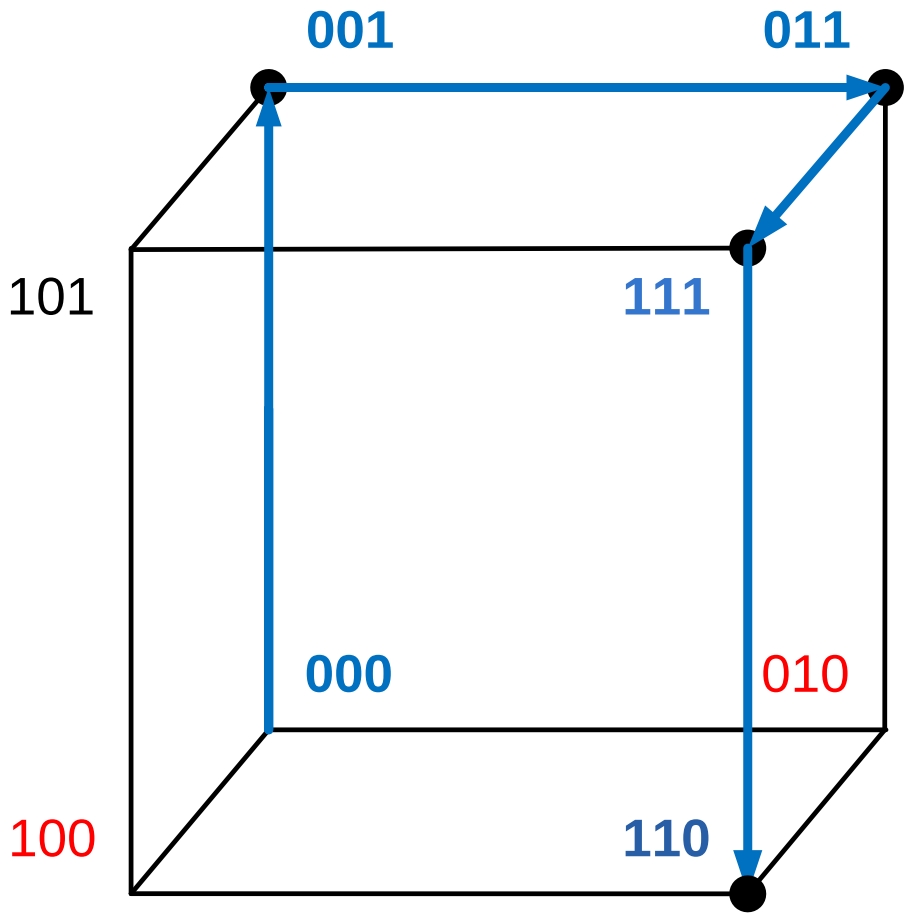
Coil-in-the-box: Example

Initial node: 000, Transition sequence: 1, 2, 3



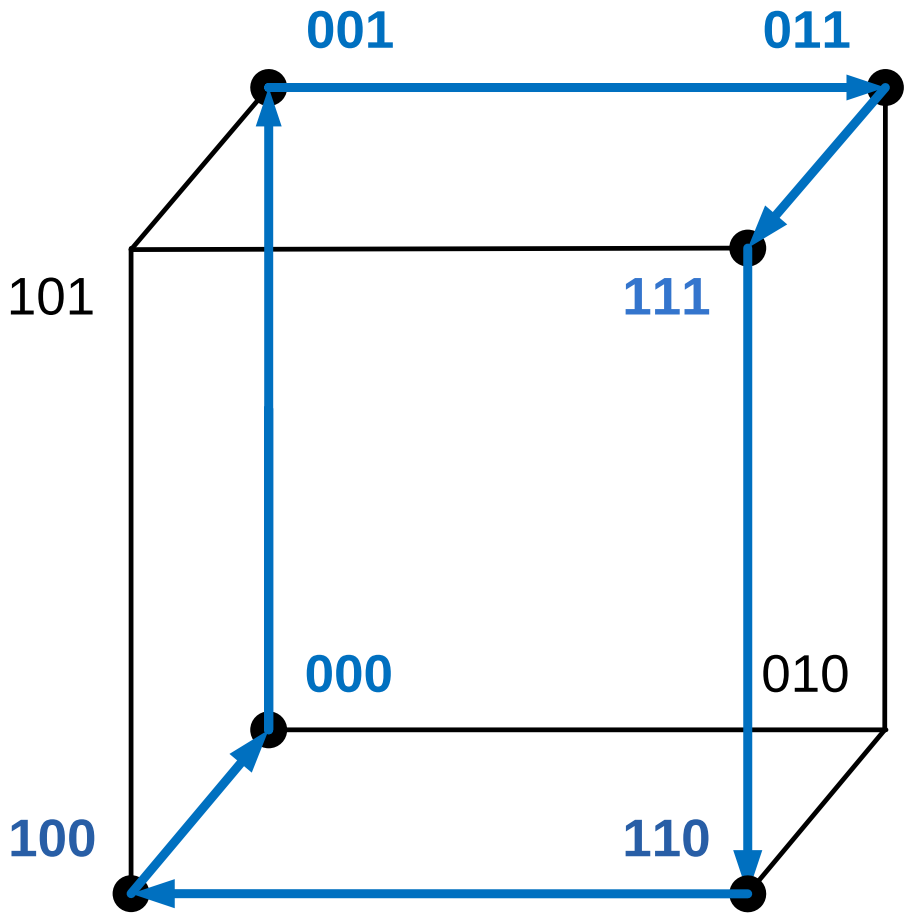
Coil-in-the-box: Example

Initial node: 000, Transition sequence: 1, 2, 3, 1



Coil-in-the-box: Example

Initial node: 000, Transition sequence: 1, 2, 3, 1, 2, 3



Hamming Distance, Paths on Hypercube

For two nodes W_k and W_l of n-cube Hamming distance

$$d_H(W_k, W_l) = \left|\{\, i \mid W_k[i] \neq W_l[i] \}\right| ,$$

Path $P = W_0, W_1, \ldots, W_{L-1}$ of length L,

$$\forall j \in \{0, 1, \ldots, L-2\}.\ \ d_H(W_j, W_{j+1}) = 1 .$$

Cycle

$$\forall j \in \{0, 1, \ldots, L-1\}.\ \ d_H(W_j, W_{j+1 \bmod L}) = 1 .$$

Cyclic distance

Definition

The *cyclic distance* $d_C(W_j, W_k)$ of two nodes W_j and W_k of a cycle of length L in an n-cube is

$$d_C(W_j, W_k) = \min\{|k-j|,\ L-|j-k|\}\ .$$

[Suparta06]

Coil-in-the-box vs Circuit code

Definition

An *induced cycle* $I_0, I_1, \ldots, I_{L-1}$ in an n-cube is a cycle such that any two nodes on the cycle are adjacent in the n-cube only if they are neighbours on the cycle:

$$\forall j, k \in \{0, 1, \ldots, L-1\}.$$
$$d_H(I_j, I_k) < 2 \Longrightarrow d_C(I_j, I_k) < 2 \ .$$

Definition

A *spread* δ *cyclic circuit* code $I_0, I_1, \ldots, I_{L-1}$:

$$\forall j, k \in \{0, 1, \ldots, L-1\}.$$
$$d_H(I_j, I_k) < \delta \Longrightarrow d_C(I_j, I_k) < \delta \ .$$

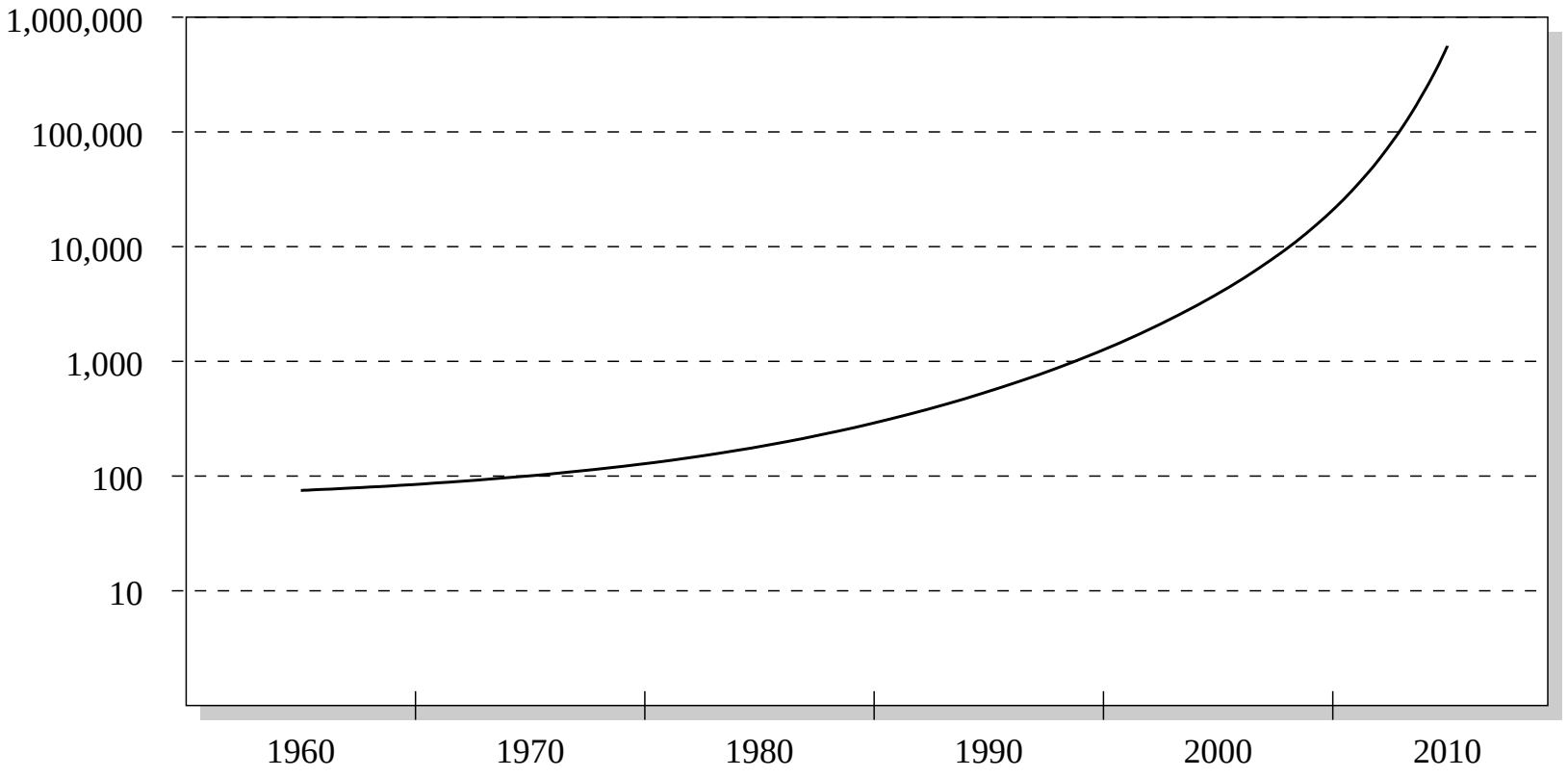
Boolean Satisfiability

- Formula over Boolean variables
- Determine if variables can be assigned to make it **true**
- if SAT provide a sample assignment
- otherwise return a proof of UNSAT
- NP-complete

Propositional SAT: Advantages

- definite answer
- provably correct (satisfying assignment or proof)
- easy to modify encoding
- easy to add new constraints
- can benefit from theoretical results
- enumeration/classification (ALL-SAT)

Propositional SAT: Solvers Progress



Encoding circuit cycles

Input	dimension n, length L
Coordinates	$(n \cdot L)$ boolean variables $I_j[k]$, where $0 \leq j < L$ and $0 \leq k < n$
Transition seq	$(n \cdot L)$ XOR gates $xor^{k,k+1}[l]$
Cycle	$\bigwedge_{j=0}^{L-1} \quad d_H(I_j, I_{j+1 \bmod L}) = 1$
Chordless	$\bigwedge_{j=0}^{L-3} \bigwedge_{k=j+\delta}^{L-1} \quad d_H(I_j, I_k) > \delta$

How to encode d_H efficiently?

if $d_H = 1$	use *once* and *twice* predicates
else	sorting networks for *unary sum* of different bits

Encoding circuit cycles: Once-twice predicates

- $once^{A,B}$ - at least one of $xor^{A,B}[i]$ is enabled
- $twice^{A,B}$ - at least two ...
- $d_H(A,B) = 1$ is encoded as

$$once^{A,B} \wedge \neg twice^{A,B}$$

- $d_H(A,B) > 1$ as

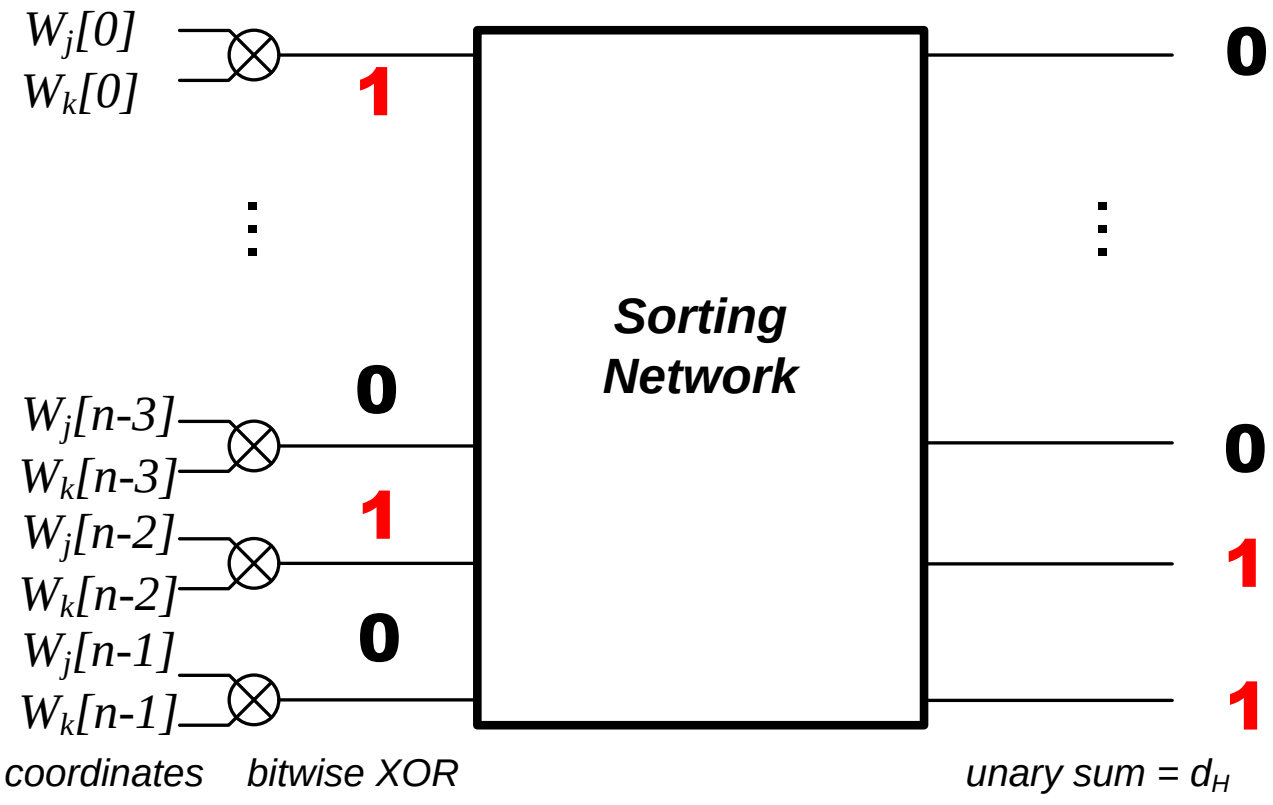
$$once^{A,B} \wedge twice^{A,B}$$

- and $d_H(A,B) \geq 1$ as

$$once^{A,B}$$

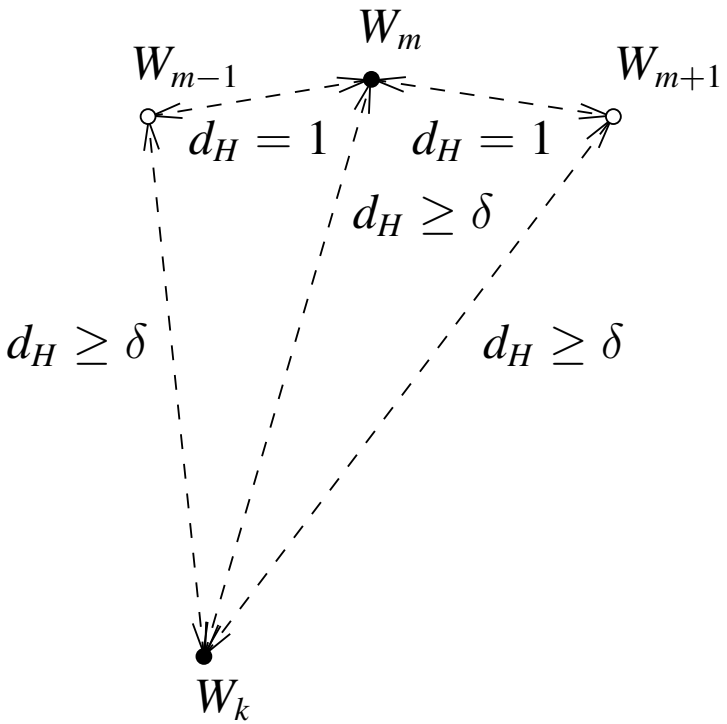
variety of encodings: OR-tree, long clause, etc.

Sorting Networks: Hamming Distance



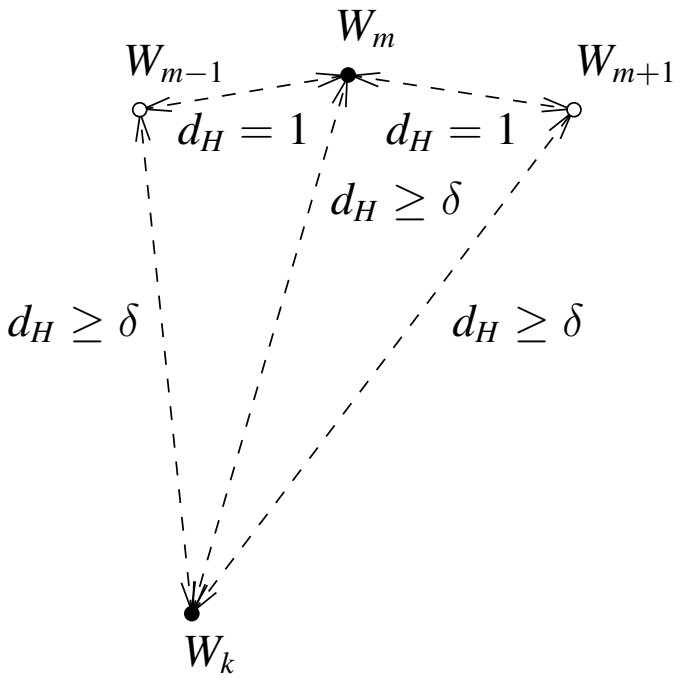
Observation

k and m have same parity
$\Longrightarrow d_C$ is even
$\Longrightarrow d_H$ is also even

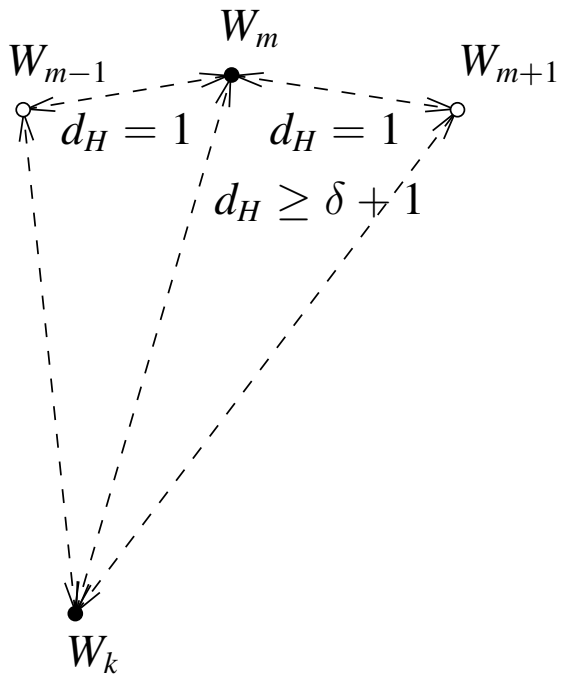


Observation

k and m have same parity
$\Longrightarrow d_C$ is even
$\Longrightarrow d_H$ is also even
if d_H is odd



$\Longrightarrow$



New Circuit Codes

Dimension \ Spread	2	3	4	5	6	7	8	9	10
4	**8**	**8**	**8**						
5	**14**	**10**	**10**	**10**					
6	**26**	**16**	**12**	**12**	**12**				
7	**48**	**24**	**14**	**14**	**14**	**14**			
8	96	**36**	**22**	**16**	**16**	**16**	**16**		
9	180	56	**30**	**24**	**18**	**18**	**18**	**18**	
10	340	86 (84)	46	**28**	**20**	**20**	**20**	**20**	**20**
11	620	154	68 (66)	40	**30**	**22**	**22**	**22**	**22**
12	1238	288	98 (96)	56 (54)	**36**	**32**	**24**	**24**	**24**
13	2468	442	182	78	48	**36**	**26**	**26**	**26**
14	4934	700	280	102	66 (64)	48	**38**	**28**	**28**
15	9868	1290	450	210	82 (80)	58	42	**40**	**30**
16	19740	2176	672	288	106 (104)	72 (70)	52	**44**	**32**
17	39840	3842	1088	476	204	90 (88)	62	50	**46**

Zinovik, I., Kroening, D. and Chebiryak, Y., "Computing Binary Combinatorial Gray Code via Exhaustive Search with SAT solvers", IEEE Trans. Inf. Theory, vol. 54 (2008), pp. 1819-1823.

CODESAT: variety of codes

- Gray codes
- snake-in-the-box
- distance-preserving
- circuit codes
- single-track codes
- binary necklaces
- with balanced distribution of weights
- classification
- incremental search
- external SAT solver of your choice

http://yury.chebiryak.name/codesat

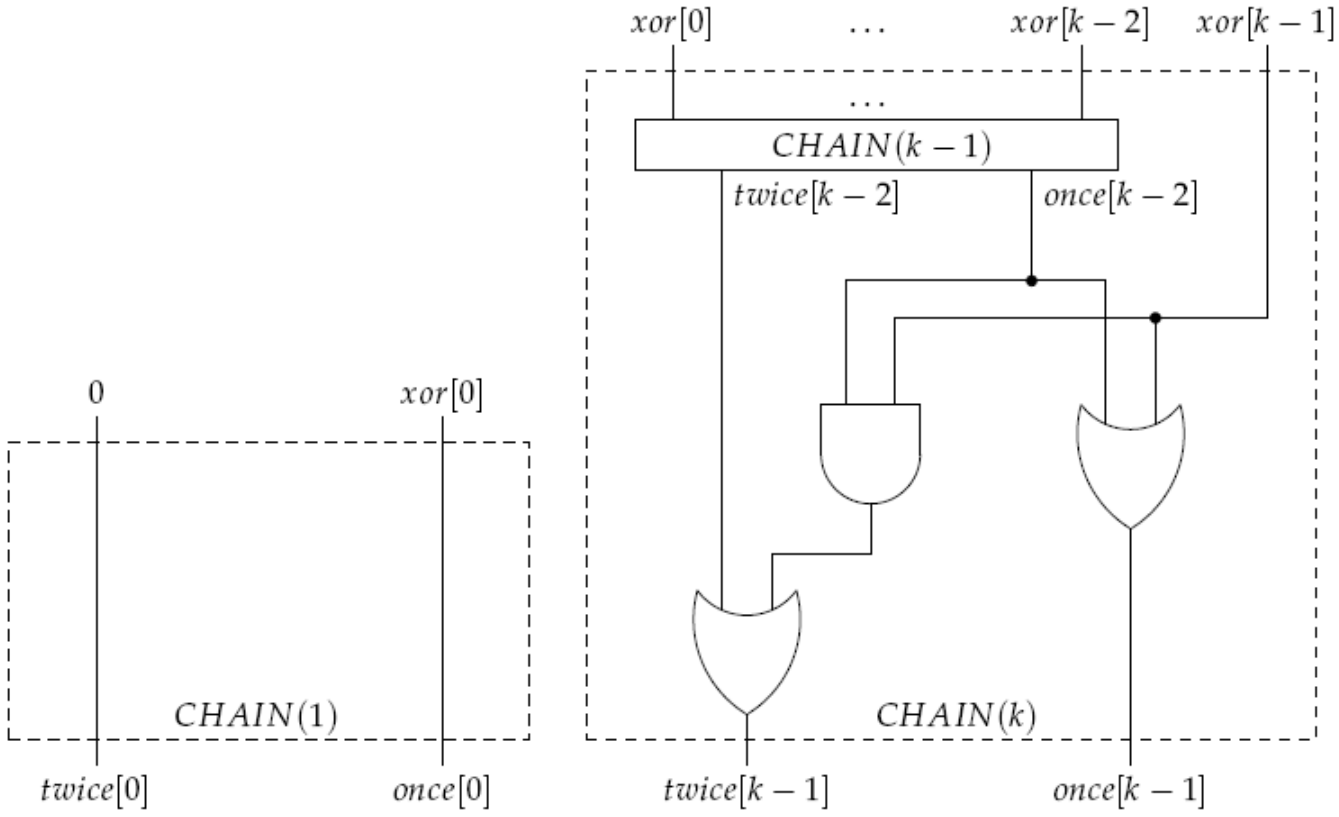
CODESAT: Simple Syntax

```
CYCLE
DIM 6
SPREAD 3
LENGTH 16
ENCODE_CIRCUIT_CODE 16
SOLVE (or DUMP_CNF)
DUMP_CODE
DUMP_COORDSEQ
END
```

Encoding circuit codes: Once-twice chains

$$
\begin{aligned}
&\left(once^{A,B}[0] := xor^{A,B}[0]\right) , \\
&\bigwedge_{j=1}^{n-1} \left(once^{A,B}[j] \longleftrightarrow once^{A,B}[j-1] \vee xor^{A,B}[j]\right) , \\
&\left(twice^{A,B}[0] := \mathsf{false}\right) , \\
&\bigwedge_{j=1}^{n-1} \left(twice^{A,B}[j] \longleftrightarrow twice^{A,B}[j-1] \vee \right. \\
&\qquad \left. once^{A,B}[j-1] \wedge xor^{A,B}[j]\right) .
\end{aligned}
$$

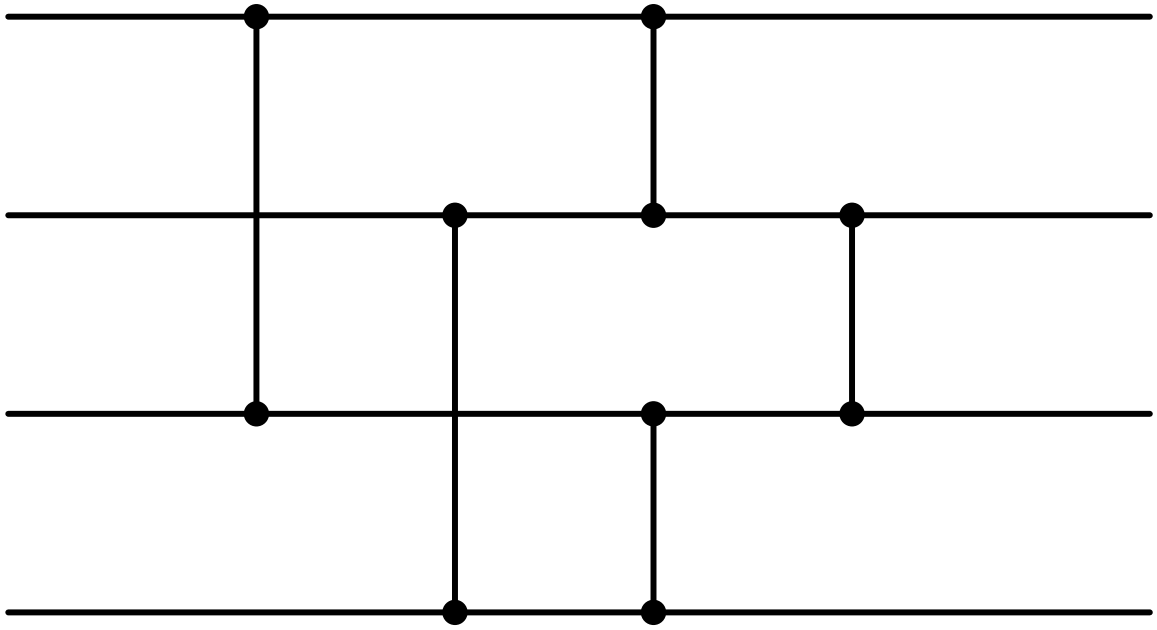
Encoding circuit codes: Once-twice chains



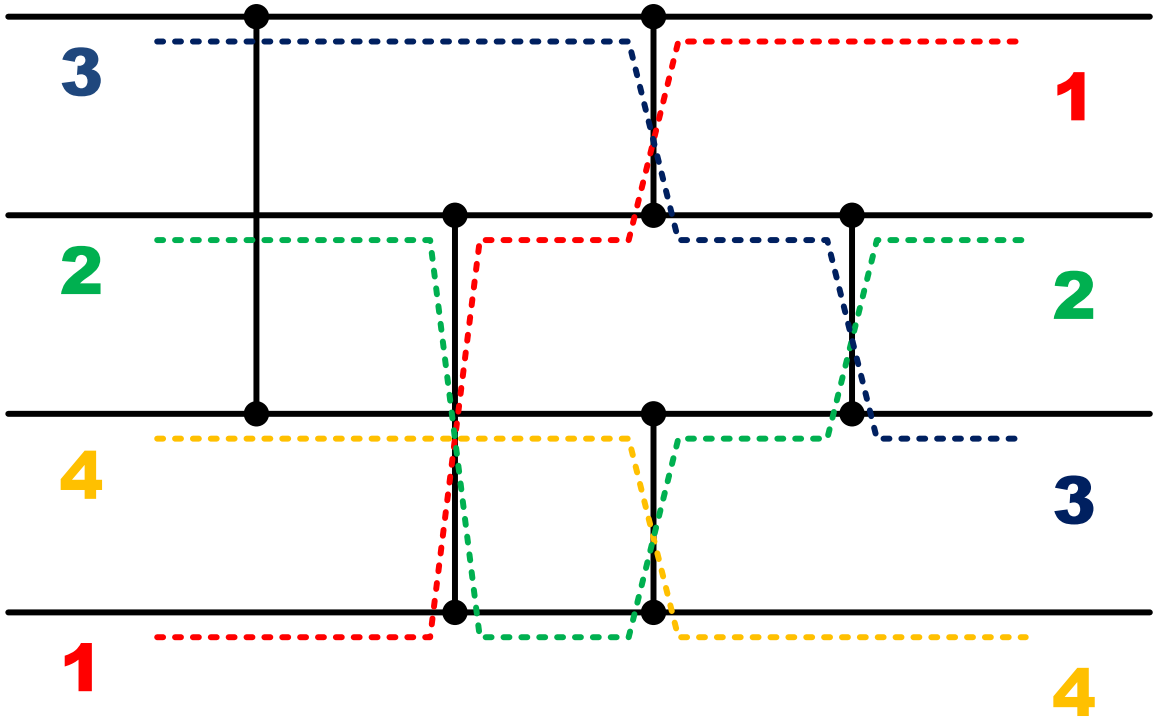
(a) base case (b) inductive definition

[Chebiryak&Kroening, JSAT'2008]

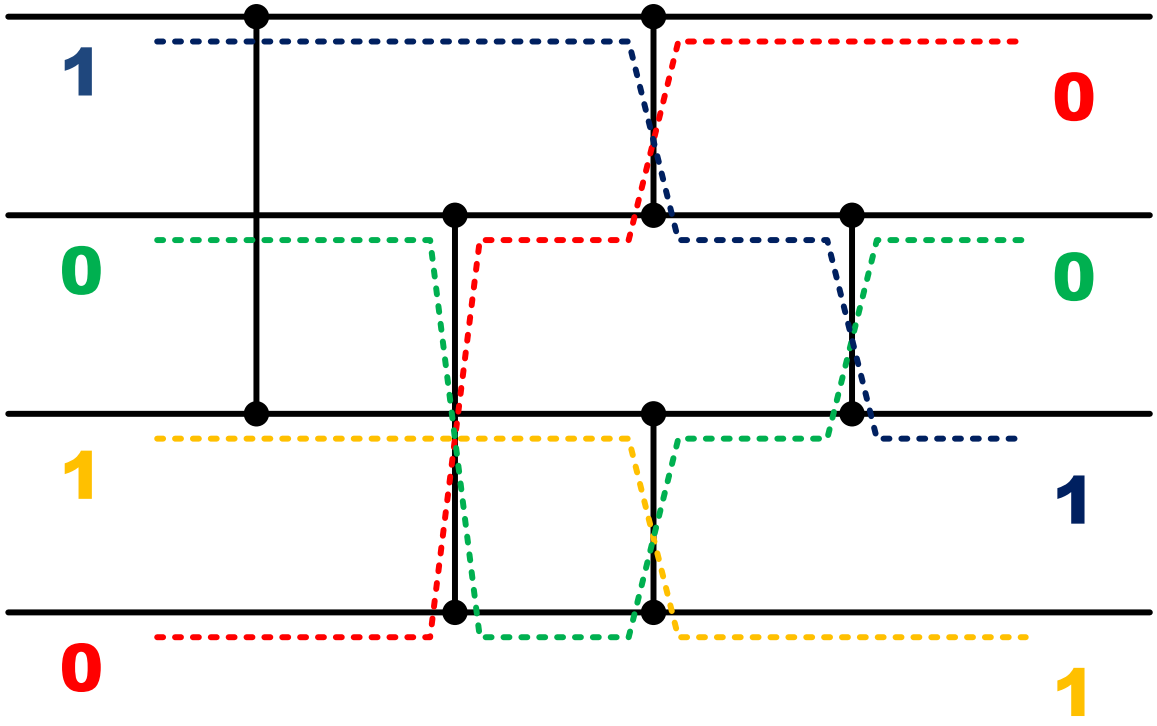
Sorting Networks: Simple Example



Sorting Networks: Sorted Sequence



Sorting Networks: 0-1 Principle



Sorting Networks: Bubble Sort