An Anytime Symmetry Detection Algorithm for ROBDDs

Neil Kettle and Andy King

University of Kent, Canterbury, CT2 7NF, United Kingdom

January 2006

Symmetry Applications

- Symmetries have applications in :-
  - Logic Synthesis
  - Technology Mapping
  - ROBDD Minimization
  - and detecting equivalence of Boolean functions for which input correspondence is unknown

Preliminaries

- Binary Decision Diagrams (BDDs) are an efficient tree-based data structure for the representation of Boolean functions
- A BDD for a Boolean functions f is a directed acyclic graph (DAG), internal vertices represent variables over which the function f_i is defined
- Each of these vertices is labeled with a variable x_i, and possesses two leaves, one leaf represents the **true** co-factor and the other represents the **false** co-factor
- Let $|G|$ denote the size of a BDD (i.e the number of vertices or nodes) over n variables

Preliminaries

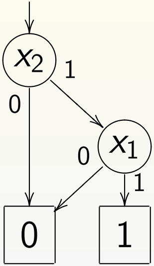

Figure: BDD for $x_1 \wedge x_2$

Preliminaries

- A co-factor of a Boolean function $f(x_1, \ldots, x_n)$ denoted $f|_{x_i \leftarrow b}$ defines a Boolean function $f(x_1, \ldots, x_{i-1}, b, x_{i+1}, \ldots, x_n)$ where $b \in \{0, 1\}$
- It is most common to specify multiple (2) variable co-factors, denoted $f|_{x_i \leftarrow b_1, x_j \leftarrow b_2}$ where $i < j$ defines a Boolean function $f(x_1, \ldots, x_{i-1}, b_1, x_{i+1}, \ldots, x_{j-1}, b_2, x_{j+1}, \ldots, x_n)$

Symmetries

- A Boolean function $f(x_1, \ldots, x_n)$ is "classically symmetric" in (x_i, x_j) iff it is equivalent to the function remaining unchanged should x_i and x_j be switched in f
- It can be shown that this is equivalent to,

$$f|_{x_i \leftarrow 1, x_j \leftarrow 0} = f|_{x_i \leftarrow 0, x_j \leftarrow 1}$$

Early Work

- Very early work on symmetries involved computing all $n^2 - n$ co-factor pairs where n is the number of variables
- Computing these co-factors requires $O(n^2(|G| \log |G|))$ time
- However, this does not include the creation of order $> n^2|G|$ nodes (not to mention *reduction*)

Mishchenko's [1] Algorithm

- Does not require co-factor computation
- $\approx O(|G|^3)$ complexity
- Presented results are indicative, experimental observation indicates the algorithm is "intractable" for $|G| > 180,000$
- Employs ZDD's for set operations,
    - paper argues creating ZDD's is irrelevant, however, in practice this is inhibiting for very large functions
- Algorithm is "monolithic"

[1] A. Mishchenko. 'Fast Computation of Symmetries in Boolean Functions'. IEEE Transactions on Computer-Aided Design, 22(11):1588-1593, 2003

Outline

- The algorithm presented here can be decomposed into the following steps,
    - we employ techniques developed in early work to sieve, that is quickly detect asymmetries
    - finally, remainder symmetries are computed using a further procedure
- Algorithm is "anytime"
- An anytime algorithm returns the best answer possible even if it is not allowed to run to completion, and may improve on the answer if it is allowed to run longer.

Outline

$A \leftarrow \texttt{ComputeAsymmetry}(f)$
$S \leftarrow \emptyset$
for $i = 1$ **to** $n-1$ **do**
  $C \leftarrow \{j \mid (i,j) \notin (S \cup A) \wedge i < j\}$
  $D \leftarrow \texttt{RemoveAsymmetry}(f, i, C)$
  $S \leftarrow S \cup \{(i,k),(k,i) \mid k \in D\}$
  $A \leftarrow A \cup \{(i,l),\ (l,i) \mid l \in C \setminus D\}$
return S

- The set C contains variables for which symmetry/asymmetry is unknown with variable i
- The set D contains those variables of C that are asymmetric with variable i
- The sets S and A contain pairs of symmetric and asymmetric variables respectively

Computing Asymmetries

- The following Lemmas describe the two asymmetry sieves found in [2]

Lemma

If an ROBDD f over a set of variables $X = \{x_1, \ldots, x_n\}$ is symmetric in the pair (x_i, x_j) and $i < j$, then every ROBDD rooted at a node labeled x_i must contain a node labeled x_j.

[2] D. Moller, J. Mohnke, and M. Weber. 'Detection of Symmetry of Boolean functions Represented by ROBDDs', International Conference on Computer-Aided Design, 680–684, 1993

Computing Asymmetries

Lemma

If an ROBDD f over a set of variables $X = \{x_1, \ldots, x_n\}$ is symmetric in the pair (x_i, x_j) and $i < j$, then every path from the root of f to a node labeled x_j must visit a node labeled x_i.

- Proofs for both Lemmas found in [2]
- All asymmetries can be found in time $O(n|G|)$ using one top-down and one bottom-up traversal

[2] D. Moller, J. Mohnke, and M. Weber. 'Detection of Symmetry of Boolean functions Represented by ROBDDs', International Conference on Computer-Aided Design, 680–684, 1993

Computing The Rest...

- We now attempt to resolve the remaining unknown variable pairs without co-factor computation

Corollary

An ROBDD f over a set of variables $X = \{x_1, \ldots, x_n\}$ is symmetric in the pair (x_i, x_j) and $i < j$ iff

- every ROBDD rooted at a node labeled x_i is symmetric in (x_i, x_j) and,
- every path from the root to a node labeled x_j passes through a node labeled x_i.

Computing The Rest...

- Observe, we have already filtered all pairs satisfying the second property
  - since, any such node x_j can reach the root without visiting a node labeled x_i
  - therefore, we don't consider them
- The procedure must therefore compare all co-factors of x_i with respect to x_j (the first property)

Optimized Algorithm

- Optimizing our algorithm is possible because of its decomposed structure (this is not possible with previous approaches)
- Further linear time asymmetry sieves can be incorporated to increase the size of the set A
- We can also exploit the transitivity of the symmetry relation

Optimized Algorithm

$A \leftarrow$ `ComputeAsymmetry`(f)
$M \leftarrow$ `ComputeSatisfyCounts`(f)
for $i = 1$ **to** n **do**
    for $j = i + 1$ **to** n **do**
        if $M(i) \neq M(j)$ **then**
            $A \leftarrow A \cup \{(i,j),(j,i)\}$
$S \leftarrow$ `ComputeAdjSymmetry`(f)
for $i = 1$ **to** $n - 2$ **do**
    $(A,S) \leftarrow$ `SymmetryClosure`(A,S)
    $C \leftarrow \{j \mid (i,j) \notin (S \cup A) \wedge i + 1 < j\}$
    $D \leftarrow$ `RemoveAsymmetry`(f,i,C)
    $S \leftarrow S \cup \{(i,k),(k,i) \mid k \in D\}$
    $A \leftarrow A \cup \{(i,l),\ (l,i) \mid l \in C \setminus D\}$
return S

Table: Experimental Results

Circuit	# In	# Out	Σ|G|	|S|	naïve	Mishchenko	S	O
pair	173	137	118066	1910	132.46	6.62	2.37	2.08
s4863	153	104	126988	547	20.60	5.30	1.41	0.82
s9234.1	**247**	**250**	**4434504**	**3454**	**>7200**	**1407.20**	**183.84**	**141.26**
s38584.1	1464	1730	150554	15629	337.59	16.70	3.12	2.80
C880	60	26	600998	262	704.54	13.90	7.75	5.20
C3540	50	22	4618194	81	>7200	132.72	71.64	65.04
urquhart2_25	48	1	722657	5	>7200	70.50	26.22	17.95
urquhart3_25	62	1	1771025	24	>7200	>7200	82.98	72.80
urquhart4_25	68	1	1736705	27	>7200	>7200	83.44	72.02
rope_0002	54	1	634914	3	>7200	192.77	22.48	18.50
rope_0004	62	1	1052214	10	>7200	487.26	41.71	37.82
rope_0006	61	1	759039	13	>7200	657.74	35.78	30.68
ferry8	111	1	290127	30	>7200	95.15	30.10	22.99
ferry10	**116**	**1**	**539419**	**38**	**>7200**	**1866.62**	**70.34**	**53.42**
ferry12	123	1	277291	36	>7200	142.10	37.63	30.95
gripper10	125	1	393485	28	>7200	261.32	52.97	44.74
gripper12	129	1	667877	43	>7200	368.50	106.32	84.90
gripper14	**118**	**1**	**767735**	**40**	**>7200**	**415.57**	**111.49**	**71.34**

Conclusions

- Our algorithm allows some symmetries to be detected **without** computing all symmetries
  - thus algorithm runtime can be tuned (anytime)
- Our runtimes are very favourably compared to those of Mishchenko
- If further linear time sieves can be devised, then our algorithm's runtime will improve with little or no extra cost
- The algorithm presented uses more intelligence, filtering many pairs before computing symmetries with `RemoveAsymmetry`(f, i, C).

Questions

Thank You.