Decision Diagrams – Concepts and Applications

24.01. - 29.01.1999

organized by

B. Becker (Freiburg)
C. Meinel (Trier)
S.-I. Minato (Japan)
F. Somenzi (Boulder)

The fifth workshop **Decision Diagrams – Concepts and Applications** in the series *Computer Aided Design and Test* at the IBFI Schloß Dagstuhl was organized by Bernd Becker (Univ. Freiburg), Christoph Meinel (Univ. Trier), Shin-Ichi Minato (NTT Optical Network, Japan), and Fabio Somenzi (Univ. of Colorado). It was attended by 31 scientists.

Decision Diagrams (DDs) have found widespread use in computer-aided design of digital circuits. They form the heart of many tools for formal verification and are also commonly used in logic synthesis, circuit testing and in the verification of communication protocols. With increasing number of applications, also in non–CAD areas, classical methods to handle DDs are being improved and new questions and problems evolve and have to be solved.

The organizers took the opportunity to bring together researchers from different areas in computer science, electrical engineering and industry. The common aim of all researchers is to deepen the understanding of DDs as a data structure, to improve existing techniques and to explore new fields of application. At the workshop, 23 lectures were presented covering different topics of DD research among them being:

- Potential and limitations of DDs, complexity of algorithms for (Boolean) function manipulation
- Minimization and approximation of Binary DDs (BDDs)
- Formal verification of sequential circuits with BDD based methods
- Extensions beyond Boolean functions to represent and manipulate word-level circuit functions
- Applications in synthesis, design and test of real-time systems, state/event systems

There were many discussions concerning challenging open questions — at universities and in industry as well — and future directions of research in the DD area.

As always, Schloß Dagstuhl and its staff provided a very convenient and stimulating environment for the workshop. The organizers wish to thank all those who helped in establishing this excellent research atmosphere.

More detailed information including some full papers can be found on the WWW-pages with the URL:

`http://www.informatik.uni-freiburg.de/ira/events/design_and_test_99/`

Contents

1 Teaching BDDs — Some Instructive Examples

Ingo Wegener, Universität Dortmund

It is well accepted that BDDs have a lot of applications. Therefore, BDDs have become a main topic of lectures. For this reason, instructive examples are necessary to explain the structural behavior of algorithms and models. In this talk examples as simple and clearly structured are presented for the following topics:

- worst case examples for apply and compose, even if the result is a constant function,
- worst and best case examples for sifting,
- an example where sifting increases the size before decreasing it,
- examples with many bad orderings and some good ones,
- examples where one BDD model has polynomial size while others have exponential size.

2 A New Framework for Dynamic Variable Reordering in Reachability Analysis

Hiroyuki Higuchi, Fujitsu Laboratories LTD

Sifting-based dynamic variable reordering drastically reduces BDD sizes during reachability analysis, mainly because each image computation handles different state sets in general and requires different variable orderings. However, it requires much computation time to reorder the variables. This is a bottleneck to apply symbolic traversal to practical examples.

In this talk a new dynamic variable reordering framework for reachability analysis is presented. In this framework a next state variable is reordered once in each image computation just before the variable is put into the partial products. Reordering of a next state variable is done efficiently by finding a good position for the variable using structural information of BDDs of the partial product and partitioned transition relations.

Several experimental results are given to demonstrate the effectiveness of the proposed framework. Methods to find a good position for a next state variable are also discussed in details.

3 Accelerating Variable Reordering

Christian Stangier, Universität Trier

Model Checking and reachability analysis are widely used techniques in verification of sequential circuits, reactive systems, protocols, etc. OBDDs allow a symbolic representation of the system that may bypass the state space explosion problem of larger systems. As the size of OBDDs and also the computation time depends on the order of the input variables, the verification may only succeed if a well suited variable order is chosen. Since the characteristics of the represented functions are changing, the variable order has to be adapted dynamically. Unfortunately, dynamic variable reordering strategies are often very time consuming and sometimes do not provide any improvement of the OBDD representation.

We present two techniques — "Block Restricted Sifting" and "Sample Sifting" — that substantially accelerate the reordering process by using either easily computable structural or implicit semantical information about the represented functions.

4 Nonapproximability Results for OBDD- and FBDD-Minimization

Detlef Sieling, Universität Dortmund

The variable ordering problem for OBDDs is the problem to compute a variable ordering minimizing the OBDD size of a function given by an OBDD. Its complexity is of theoretical and practical importance because the choice of the variable ordering can decide between polynomial or exponential OBDD size and between success or failure of an application. The known NP-hardness results do not exclude polynomial time approximation algorithms for the variable ordering problem, i.e. algorithms which guarantee to obtain variable orderings for which the OBDD size is larger than the optimum by at most some constant factor. The main result is that the existence of such an algorithm implies P=NP. Hence, we get a justification to use heuristics and to give up the search for approximation algorithms for the variable ordering problem.

Besides OBDDs, Free BDDs (FBDDs) can be used as a data structure for Boolean functions. The manipulation of FBDDs is efficient if only FBDDs respecting a fixed graph ordering are used. Graph orderings are a generalization of variable orderings. Hence, we get the problem of minimizing FBDDs and of optimizing graph orderings for a function given by an FBDD. The other main result is that polynomial time approximation schemes

for these problems imply P=NP or ZPP=NP, respectively. Approximation schemes are algorithms whose result is larger than the optimum by a factor of at most $1+\epsilon$ where $\epsilon > 0$ is part of the input. Again, we get a justification to use heuristics.

5 Minimization of Free BDDs

Wolfgang Günther, Universität Freiburg

Free BDDs (FBDDs) are a generalization of OBDDs that allow different orders of the variables on each path. They allow a more compact representation for Boolean functions, in some cases even an exponential reduction compared to OBDDs can be observed. Furthermore, many properties of OBDDs also hold for FBDDs, like e.g. the realization in PTL or the testability of the circuits (if they are directly mapped).

First, we presented an exact minimization algorithm for FBDDs, i.e. an algorithm that finds the minimal FBDD (in terms of the number of nodes) among all FBDDs representing the given function. But in spite of an effective pruning technique and the use of symmetries, the algorithm is only applicable to small functions. For those functions it turned out that sizes could not be reduced in most cases.

Furthermore, we presented a simple heuristic which can be seen as an extension of the exact method. Experimental results showed that in many cases, the best known OBDD sizes could be reduced by about 10%.

6 On a Method to Accelerate Functional Decompositions

Tsutomu Sasao, Kyushu Institute of Technology, Japan

A function $f(X)$ has a simple disjoint decomposition if f is represented as $f(X_1, X_2) = g(h(X_1), X_2)$. The problem of functional decomposition is to find a bipartition (X_1, X_2) of X such that $f(X_1, X_2) = g(h(X_1), X_2)$; In this talk, I showed an efficient method to detect bipartitions (X_1, X_2) that do not produce decompositions for $f(X)$. It quickly reduces the search space for the decompositions by using a look-up table for undecomposable functions. A systematic method to find simple disjoint decompositions by BDD was presented.

7 Synthesis of controllers from Interval Temporal Logic specification and application to C program validation

Masahiro Fujita, Fujitsu Laboratories of America

Interval Temporal Logic (ITL) gives much more flexibility in describing both serial and parallel behaviors of digital systems. In order to verify systems with respect to it or synthesize circuits from it, ITL formulae must be expanded into automata first. This process is very complicated and mostly very time consuming. Here a restricted ITL is defined, whose intervals are always finite. Then FSM models can be relatively easily generated from such ITL formulae. Moreover, by using decision diagrams to represent generated sub-formulae when expanding given ITL formulae, the expansion process can be efficiently implemented. The experimental results demonstrated the usefulness of the proposed approach.

As an application of the proposed approach, ways to validate C programs with respect to ITL specification are also discussed.

8 Verification of Hierarchical State/Event Systems using Reusability and Compositionality

Gerd Behrmann, Aalborg University, Denmark

We propose two new techniques for the verification of state/event systems (a parallel extension of Mealy Machines) and a hierarchical extension. Compositional Backwards Reachability (CBR) uses a dependency analysis between the components of the system to determine which components to include in the reachability analysis. In this way CBR starts with a minimal set of components, performing backward steps that are possible regardless of the behavior of the remaining components. If necessary the set of components considered is extended according to the dependency analysis, reusing the work done with the smaller set. Using CBR we were able to verify a large number of reachability properties for a system containing 1492 parallel components (declared statespace of 10^{476}).

The Reusable Reachability Check (RRC) improves the verification of reachability properties for hierarchical systems. The technique chooses superstates as landmarks that guide the verification. Ones reachability of a superstate has been established, that information is reused for establishing reachability of subsets of the state space of the super state. When

combined with CBR, RRC has the additional benefit of dividing a reachability analysis into several simpler checks, each only requiring a small set of components in the CBR analysis. Experimental results show that CBR alone would suffer from an increasing hierarchical depth. With the addition of RRC verification of hierarchical systems becomes independent of the depth of the system.

9 Guided Search for LTL model checking

Roderick Bloem, University of Colorado at Boulder

In order to avoid BDD size explosion during fixpoint computations, we can use a series of underapproximations to the transition relation. Using these underapproximations, we replace a fixpoint computation by a series of fixpoints that is often much easier to compute.

The underapproximations are provided by the user in the form of constraints on the circuit (hints). They are chosen so as to allow a "regular" exploration of the state space, avoiding a BDD blowup.

The application of hints to reachability analysis is known. Here we extend this approach to ω-regular model checking in general, and LTL model checking in particular. We make a structural distinction between three types of Büchi automata: general, weak, and terminal. The latter two types occur frequently in LTL model checking, and can be checked more efficiently. Hints can be used for either of these types. Our preliminary experimental results show that this allows for efficient LTL model checking in many cases.

10 Using Lower Bounds during BDD Minimization

Rolf Drechsler, Universität Freiburg

Ordered Binary Decision Diagrams (OBDDs) are a data structure for representation and manipulation of Boolean functions often applied in VLSI CAD. The choice of the variable ordering largely influences the size of the representation, i.e. it may vary from linear to exponential.

Efficient methods for exact and heuristic minimization of OBDDs are presented that make intensive use of lower bound computations during the optimization. By this large parts of the search space can be pruned resulting in very fast algorithms.

Using the exact algorithm it is for the first time possible to exactly minimize larger functions, e.g. adder functions with up to 64 variables. The heuristic approach achieves speed-ups of 70% compared to the classical approach that does not consider lower bounds.

11 OHO — OBDD Heuristics Online

Arno Wagner, Universität Trier
(http://www.informatik.uni-trier.de/TI/OHO.html)

One main problem with new published OBDD heuristics is that the information available to evaluate the characteristics of the heuristic is limited to the performance on some limited set of benchmarks. While this is sufficient for a first impression, if there is need for more information, like the performance on a special class of circuits, presently the only option is to reimplement the heuristic or to obtain the code. To really work with a heuristic one of these two things has eventually to be done, but as this is a lot of effort it is unsuitable for early evaluation purposes.

To address this problem we are constructing a system, called OHO, that using the WWW offers an easy way to use OBDD-tools online. The user is presented with a web-form that allows to choose parameters and submit input data for the tool that contains the heuristic. Presently available are Block Restricted Sifting and (soon) Sample Sifting in SMV as well as Block Restricted Sifting, Sample Sifting and Linear Sifting within CUDD via nanotrav.

We describe architectural and implementational aspects such as scalability, dependability and flexibility. With our system it is quite easy to add additional user-interfaces and OBDD-based tools. Adding more machines to do the actual computations is easy as well. An additional benefit is that these machines doing the computations need not to be at our site, but even remote machines connected to the internet can be used to do computations with tools that are only available on these remote machines. Such remote computations would not be visible to the user.

12 More Bad BDD Ideas

Alan J. Hu, University of British Columbia

Following my pattern from the seminar two years ago, I decided once again to present ideas that did not work — as a basis for discussion and feedback. I presented three different ideas:

1. For the problem of computing the existential quantification of the conjunction of a set of BDDs, the idea is as follows:
   Given $\exists x\ [c_1 \wedge \ldots \wedge c_n]$.
   Let $c_i' = \exists x\ c_i$.

Let $\hat{c}_i = c_i \downarrow c'_i$, where $\downarrow$ denotes any simplification operation.
Then the desired result is

$$c'_1 \wedge \ldots \wedge c'_n \wedge \underbrace{\exists x(\bigvee_{i=1}^{n} \hat{c}_i)}_{\text{This gets built as a single BDD.}}.$$

Unfortunately, the method usually fails because the conjunction of the $\hat{c}_i$ blows up.

2. In BDD result caching, the observation was made that results based on variables earlier in the variable ordering represent more work. Hence, it might make sense to give priority to higher-level calls. We were able to improve the hit rate, but not to gain speed up. A major difficulty is that the cache gets called so frequently that we cannot afford much computation.

3. I continue to feel that there must be some way to get "global" functional information for variable ordering, rather than relying on the local search of e.g. sifting. For this failed idea, we started out trying to use sparse matrix profile reduction heuristics combined with ad hoc measures of affinity between variables. Eventually, we settled on a simple heuristic:

   (a) For variables i and j, $i < j$, define

   $$a_{ij} = \min\left(\frac{t_{ij}}{n_j}, \frac{f_{ij}}{n_j}\right),$$

   where n_j = number of nodes on level j
   t_{ij} = number of nodes on level j assuming x_i is true
   f_{ij} = number of nodes on level j assuming x_i is false

   It is easy to get these numbers without building the cofactors.

   (b) Make the matrix symmetric. $a_{ii} = 1$.

   (c) Set the lower 90% of matrix entries to 0 to get a sparse matrix.

   (d) Compute the order that pushes the zeros to the lower right.

   This algorithm on the ISCAS 89 benchmarks produced results that were better than window 4 to convergence and worse than sifting. It also ran fast, but not fast enough to be interesting.

I closed my talk by emphasizing the need for good experimental methodology, especially the need for good examples for benchmarks that are public, easy to process, but also large and real.

The ideas presented included work by Felix Chary, David Currie, Paul Kundarewich, and Kim Milvary-Jenson. They should share in the credit for any good ideas here; I assume full responsibility for any stupidity.

13 An Implicit Method of Boolean Resubstitution in Multi-Level Logic Synthesis

Shin-ichi Minato, NTT Optical Network Systems Labs., Japan

In recent a few years, we have been developing a multi-level logic synthesis program based on implicit cube set representation with zero-suppressed BDDs (ZBDDs). Our program "SDDOPT" is much faster than SIS, which is based on explicit cube set representation, and in many cases SDDOPT produces as good results as SIS in terms of optimization quality. However, there are some exceptional cases where the result of SDDOPT is much worse (more than 400%) than SIS's result. In such cases, SIS uses Boolean resubstitution techniques by using the correlationship of the multiple primary output functions. This is sometimes quite effective, so we introduced this resubstitution techniques into SDDOPT.

SIS performs Boolean resubstitution using ESPRESSO based on explicit cube set representation, but we developed a new method of Boolean resubstitution using ISOP (Minato-Morreale) algorithm based on implicit cube set representation. In this talk, we show a trivial but effective heuristic method to perform Boolean resubstitution using M-M algorithm. Experimental results show that our method is very effective to the instances where SDDOPT was much worse than SIS, and thus our method fills the gap between SDDOPT and SIS in terms of optimization power. Computation time is a little increased, but still much faster than SIS. Lastly, we can conclude that there are some typical cases where the optimization result is very sensitive to the Boolean resubstitution, and our method is useful to such cases.

14 In-Place Power Optimization for LUT Based FPGAs

Balakrishna Kumthekar, University of Colorado at Boulder

We consider a technique to perform power-oriented re-configuration of a system implemented using LUT-based FPGAs. The main features of our approach are: Aggressive exploitation of degrees of freedom, concurrent optimization of multiple LUTs based on Boolean relation, and in-place reprogramming without re-placement and re-routing. Our technique optimizes the combinational component of the CLBs after layout and does not require any re-routing or re-wiring. Hence, delay and CLB usage are left unchanged, while power is minimized. The algorithm operates locally on the various LUT clusters of the network. It is applicable and best performs on large examples. An average power reduction of 20% has been obtained on standard benchmarks.

15 Symbolic BDD techniques for Exact Scheduling

Forrest Brewer, ECE/UCSB Santa Barbara CA

We describe a technique for representation of CDFG resource scheduling on a BDD-based automata model. This model is very efficient when compared to previous work and several examples of exact solution of large problems are reported for the first time. Key to this technique is the representation of our operation node on a single bit NFA automata. In this way, all scheduling constraints are easily added to the automata transition relation. Scheduling then proceeds via symbolic state enumeration until a terminal state is reached. A reverse traversal then restricts solutions to only the shortest paths. In the case of control dependent scheduling, added bits guard the switching behavior and implicitly allow all forms of code hoisting and CDFG optimization. However, the set of minimum time schedules must be validated to ensure causal scheduling. This is done using an iterative fixed point procedure. We also introduced the problem of protocol based scheduling based on the idea of automata co-execution in a generalization of this model.

16 MORE good BDD ideas

Andreas Hett, Universität Freiburg

The MORE-approach idea performing synthesis by introducing a set of operator nodes atop of the operand BDDs, shifting them by specialized level exchanges and regaining canonicity by means of a reduction run, proved to be a good alternative for combinational problems. However for sequential problems (e.g. model checking) synthesis behavior differs a lot. Especially due to the lack of parallelity (e.g. pipelining) a CT (computed table) has a much higher influence on the calculation process and this was added in our new approach. In order to allow transformations on operator nodes (giving the opportunity to eliminate operator nodes before actually carrying out the operator), an online-reduction (for eliminating superfluous calculation paths as soon as possible) and the opportunity to synthesize on selected "promising frontiers" we introduced a new implementation of MORE. This last feature allows us to not only handle 1-path (DFS) or all nodes of a level (BFS) but handling the most "promising paths" first. The best solution to find these paths would be an oracle telling us in no time which sub-problem to solve first in order to get the most reductions eliminating neighbored branches which leads to a better performance. However this oracle doesn't exist and we have to rely on an estimation of our current stage (i.e. analysis of the BDD operands that are to be combined) to get this data. The better, faster and more precise this estimation works, the more profit we can make, improving on runtime and memory usage as well.

17 Improving Reachability Analysis by means of Activity Profiles

Gianpiero Cabodi, Politecnico di Torino

Symbolic techniques are still limited by the size of the BDDs involved in computations. Extending their applicability to larger and real circuits is still a key issue.

Within this framework, we introduce "activity profiles" as a novel technique to characterize transition relations. We use an inexpensive reachability analysis phase as a "learning" methodology to collect activity measures for each BDD node of the transition relation. We operate within inner steps of image computations based on the relational product operator. For each node of the transition relation we record countings of recursions and operation cache hits, as well as a measure of newly generated nodes.

The above informations can be used for several purposes. In particular, we present an application of activity profiles in the field of reachability analysis. We propose transition relation subsetting and partial traversals of the state transition graph. We show that a sequence of partial traversals is able to complete a reachability analysis problem with improved memory and time performance.

18 Approximate Reachability Don't Cares in Model Checking

In-Ho Moon, University of Colorado at Boulder

RDCs (Reachability Don't Cares) can have a dramatic impact on the cost of CTL Model checking. Unfortunately, RDCs are often much more difficult to compute than the satisfying set of typical CTL formulas. We address this problem through the use of Approximate Reachability Don't Cares (ARPCs), computed with the algorithms developed for the VERITAS sequential synthesis package. ARDCs represent an upper bound on the set of true reachable (Don't Cares) states. ARDCs can be 10X to 100X (or much more for very large circuits) cheaper to compute than RDCs, and in some cases have the same dramatic effect on CTL model checking as the real RDCs. We also discuss the application of ARDCs to the problem of exactly computing the RDCs themselves. Experiments on industrial benchmarks show that order of magnitude speedups are possible, and occurs frequently. The experimental results presented strongly support our claim that ARDCs play a safe and important way out of a serious dilemma: RDCs are necessary for tractable model checking of many large circuits, but the computation of the RDCs themselves in

often intractable. We include significant extensions of the VERITAS algorithms, and show that they can be up to an order of magnitude faster, while computing a virtually identical upper bound.

19 Decision Diagrams and Division

Christoph Scholl, Universität Freiburg

Several types of Decision Diagrams (DDs) have been proposed for the verification of Integrated Circuits. Recently, word-level DDs like BMDs, ∗BMDs, HDDs, K∗BMDs and ∗PHDDs have been attracting more and more interest, e.g., by using ∗BMDs and ∗PHDDs it was for the first time possible to formally verify integer multipliers and floating point multipliers of "significant" bitlengths, respectively.

On the other hand, it has been unknown, whether division, the operation inverse to multiplication, can be efficiently represented by some type of word-level DDs. In this paper we show that the representational power of any word-level DD is too weak to efficiently represent integer division. Thus, neither a clever choice of the variable ordering, the decomposition type or the edge weights, can lead to a polynomial DD size for division.

For the proof we introduce Word-Level Linear Combination Diagrams (WLCDs), a DD which may be viewed as a "generic" word-level DD. We derive an exponential lower bound on the WLCD representation size for integer dividers and show how this bound transfers to all other word-level DDs.

Moreover we presented an idea how to verify dividers using word-level DDs all the same (by means of a transformation approach).

20 A Word-Level graph manipulation Package

Stefan Höreth, Siemens AG

In the talk I presented a decision diagram package for Word-level graph manipulation. The package is a framework that supports many well-known, ordered types of decision diagrams, like OBDDs, ∗BMDs and K∗BMDs.

Major features of the package, currently not found in other implementations are

- its support for hybrid operations involving multiple graph types
- its support for dynamic reordering techniques for graph types like ∗BMDs or K∗BMDs

- a complete set of data-path operations.

I like to invite everyone to check-out the web-page

http://www.rs.e-technik.tu-darmstadt.de/∼sth

for background information and demos.

21 Automatic Test Derivation for Real-Time Systems

Brian Nielsen, Aalborg University, Denmark

For real-time systems the timelyness of a response is equally important as the correct type of response. An actual implementation of such a system must therefore be checked by means of testing with respect to both logical and timing errors.

We propose an algorithm for automatically deriving test-cases from specifications given as Event Recording Automata, a subclass of timed automata. To aid testselection we partition the statespace of the specification into coarse equivalence classes and cover each with at least one test. The reachable parts of the equivalent classes can be computed by solving linear inequalities.

Our technique is implemented in a prototype tool which uses difference bound matrixes as the underlying data-structure.

22 Satisfiability on a netlist of BDDs

Vigyan Singhal, Cadence Berkeley Labs.

We are interested in solving a SAT problem, expressed as a set of BDDs: the root BDD, $f(\bar{X}, \bar{Y})$ and the cutpoint BDDs, $y_i \equiv f_i(\bar{X}, \bar{Y})$. Here $\bar{X}$ and $\bar{Y}$ represent the sets of input and cutpoint variables. These problems arise from solving analysis problems on Boolean circuits, where the circuits are so large that building *one* BDD is impossible, and cutpoints (denoted by $\bar{Y}$ variables) have to be introduced. Our applications for these problems include timing analysis, combinational verification, LTL model checking and directed simulation.

The basic algorithm to solve this problem is to successively substitute the cutpoint functions until either a) the root BDD is 0, or b) there exists a satisfiable path in the root BDD consisting of only $\bar{X}$ variables. Clearly the order of composition is important,

and we have some heuristics. In our experience, using this compose algorithm alone does not yield a robust method. It is best to balance this algorithm with other search-based methods which have complementary strengths. As we show, it is important to integrate the networks tightly so that each method yields a successively simpler problem for the next method.

I also present an open problem: To determine the complexity of: "Given two sequential circuits with same number of latches, does there exist a one-to-one mapping of latches such that the associated combinational function for the next state functions of each latch pair are equivalent (note that the input values are determined by the one-to-one mapping). My guess is that the problem is Σ^2-complete.

23 Minimization and Approximations of Binary Decision Diagrams

Fabio Somenzi, University of Colorado at Boulder

We consider the problem of deriving a small BDD from a given function interval. We assume a fixed variable order and suppose that the function interval is given as either a lower bound and upper bound, or an on-set and a care-set. We review several algorithms that have been proposed for this problem that we call the minimization problem.

The first and most influential algorithm for the minimization problem is due to Coudert, Berthet, and Madre. It is known by the name of "constrain". We examine its properties and contrast it to more recent algorithms like "restrict", "compaction", and "squeeze". The last one is a new algorithm based on one-sided matching and working on function intervals specified as pairs of bounds.

The approximation problem is the problem of finding a function at a small Hamming distance from the input function and a small BDD. If the new function is contained in the old one, the problem is called underapproximation. We review several algorithms that solve the underapproximation problem by redirecting edges inside the BDD.

Early algorithms like "heavy-branch subsetting" and "short-path subsetting" run in time linear in the size of the input BDD. More recent algorithms trade off increased runtime for higher quality of the solution. In particular, we discuss a new algorithm called "remapping underapproximation", that combines the remapping idea of "constrain" with a scheme for the estimation of the impact of redirecting one edge. The algorithm is "safe" in that it does not decrease the density of a BDD (the ratio of minterms to nodes).