Underapproximation for Model-Checking Based on Random Cryptographic Constructions

Arie Matsliah[1,2] and Ofer Strichman[1,3]

[1] IBM Haifa Research Laboratory, Haifa, Israel
[2] Faculty of Computer Science, Technion, Haifa, Israel
ariem@cs.technion.ac.il
[3] Information Systems Engineering, IE, Technion, Haifa, Israel
ofers@ie.technion.ac.il

Abstract. For two naturals m, n such that $m < n$, we show how to construct a circuit C with m inputs and n outputs, that has the following property: for some $0 \leq k \leq m$, the circuit defines a k-universal function. This means, informally, that for every subset K of k outputs, every possible valuation of the variables in K is reachable (we prove that k is very close to m with an arbitrarily high probability). Now consider a circuit M with n inputs that we wish to model-check. Connecting the inputs of M to the outputs of C gives us a new circuit M' with m inputs, that its original inputs have freedom defined by k. This is a very attractive feature for underapproximation in model-checking: on one hand the combined circuit has a smaller number of inputs, and on the other hand it is expected to find an error state fast if there is one.

We report initial experimental results with bounded model checking of industrial designs (the method is equally applicable to unbounded model checking and to simulation), which shows mixed results. An interesting observation, however, is that in 13 out of 17 designs, setting m to be $n/5$ is sufficient to detect the bug. This is in contrast to other underapproximation that are based on reducing the number of inputs, which in most cases cannot detect the bug even with $m = n/2$.

1 Introduction

Experience with model-checking of industrial hardware designs shows that when the model violates a specification, it is frequently the case that the values of only some of the inputs is important for triggering an erroneous behavior (as the saying goes: “when it rains - it pours!”). Based on this observation it is appealing to underapproximate the model, attempting to make it easier to check, yet not eliminating the problematic behavior altogether. In other words, the challenge is to underapproximate by finding those restrictions that do not prevent all error states from being reached. Designing a fully automatic model-checking algorithm based on underapproximation that is still sound and complete requires an iterative process of underapproximation and refinement.

Automatic underapproximation/refinement for model-checking is not nearly as popular as its dual, automated overapproximation/refinement. An

W. Damm and H. Hermanns (Eds.): CAV 2007, LNCS 4590, pp. 339–351, 2007.

overapproximating abstraction may result in a false negative, accompanied by a spurious (abstract) counterexample. This counterexample can then be used to guide the refinement process, as in the CEGAR [8,4,5,3] and proof-based [1] frameworks (in the latter only the length of the counterexample is used). All of these works are based on overapproximation.

An underapproximation, on the other hand, may result in a false positive: here, good refinements are harder to achieve, as there is no equivalent to the counterexample that can guide it. An exception to this rule is in SAT-based Bounded Model-Checking (BMC), where the unsatisfiable core can guide the refinement: Grumberg et al. [6] used this fact in their work on underapproximation-refinement for bounded model checking of multi-process systems. We are only aware of few works on underapproximations with BDDs (e.g., [10,11,2]), all of which are based on the size of the BDD (e.g., restricting the growth of the reachable state-space when the BDD size becomes too large), but none of them are fully automatic and complete.

In this paper we focus on underapproximations that are based on reducing the number of inputs to the model. In theory this should make the model easier to solve, at least in the worst-case, since the number of computation paths has exponential dependency on the number of inputs[1]. The most basic technique is to restrict some of the inputs to constants. Such naive underapproximation, combined with a gradual lifting of these restrictions (typically in a manual manner) is a common practice in the industry probably from the very first days of industrial model-checking. If no user-guidance is provided, however, an automated refinement based on some arbitrary order of lifting the restrictions has a small chance to succeed, unless the bug is ubiquitous enough to be very simple to find. It is enough for one of the inputs necessary for exposing the error-trace to be falsely restricted, to potentially make the model too big for model-checking by the time this input is released. Another option is to combine inputs (arbitrarily) and refining by splitting the combined sets. In Section 2.2 we analyze these options in more depth.

What is this article about? The current work suggests an underapproximation which reduces the number of inputs as well, but it is based on adding circuitry to the model, while maintaining a measurable and uniform degree of freedom to the original inputs. This technique is automatic, easy to combine in an underapproximation-refinement method, and is applicable to any form of model-checking or simulation, whether it is SAT-based or BDD-based. The technique is inspired by theoretical constructions of cryptographic circuits, the Pseudo Random Generators (PRGs). These PRGs can expand a short truly random Boolean sequence into a longer one, which is almost random (more details are given in Section 2). Based on constructions of these PRGs, we build simple Boolean circuits and prove that they have the universality property as defined below.

[1] In the context of SAT this is less obvious because SAT does not distinguish between inputs and other variables. But the reduction in the number of inputs implies that it has a smaller upper-bound on the size of the smallest back-door set [13], namely the inputs, which suggest a better upper-bound on the run-time.

Consider a model M with n inputs that we wish to model-check. We build a Boolean circuit with m inputs and n outputs, $0 < m < n$, which is *k-universal*. Informally, this means that the circuit implements a function such that any valuation of at most k outputs can be reached under some assignment to the inputs. We then connect the outputs of C to the inputs of M (see Figure 1). The composed model M' has less inputs and underapproximates the original model M. One of the challenges in such a construction is to guarantee high values of k for a given value of m. We discuss this question in detail in Section 3.1.

Universality was also used in [7], in the context of simulation. The authors constructed vectors that have a certain degree of universality and showed that this indeed has a better chance to expose problems in comparison to alterative vector sets of the same size.

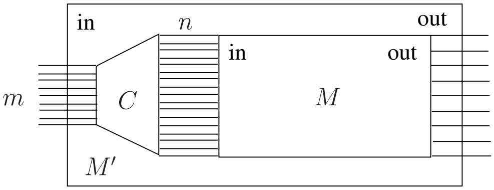

Fig. 1. Since the attached Boolean circuit is k-universal, *any* assignment on *any* k out of the n inputs of the original model M, can be achieved under some assignment on the inputs of M'

The main contribution of this paper is theoretical: we show how to construct M' and derive lower-bounds on the value of k as a function of m. Since the construction is based on a random function, the results are probabilistic. We also define a weaker version of universality, called (k, ϵ)-universality, in which for only a $1 - \epsilon$ fraction of the subsets of size k, any assignment is possible (k-universality corresponds to $\epsilon = 0$). With this relaxation we prove that for $k = \max(0, m - \log \frac{1}{\epsilon \cdot \delta})$, where δ is the confidence level, the circuit C is (k, ϵ)-universal with probability at least $1 - \delta$. For example, with probability 0.99, for 99% of the subsets of size $k = \max(0, m - 14)$, any assignment can be achieved.

In Section 4 we describe our experiments, which attempt to check whether k-universality can be useful in the context of model-checking. In other words, whether the freedom on the original inputs as guaranteed by this method is indeed helpful in detecting bugs in real designs, in comparison to other forms of underapproximation that have the same search-space. The answer is conclusive: it is able to find bugs with far less inputs. The results are less conclusive, but still positive, when it comes to comparing to a run without underapproximation at all. This is probably due to the fact that our construction is based on a XOR function, which is notoriously hard for SAT solvers. We conclude in Subsection 4.1 by pointing to several practical issues in applying this method that are still open.

2 Local Universality

2.1 k-Universal Circuits and Upper-Bound on k

Let C be a Boolean circuit with m inputs and n outputs, $m \leq n$, implementing a corresponding function $C : \{0,1\}^m \rightarrow \{0,1\}^n$.

Definition 1 (k-universal functions). *The function C is* k-universal *if for every subset $K \subset \{1,\ldots,n\}$ of k outputs and every partial assignment $\alpha_K \in \{0,1\}^k$ on K, there is a full assignment $\alpha \in \{0,1\}^m$ on the inputs of C such that $C(\alpha)|_K = \alpha_K$.* □

In other words, any subset of k output bits can take all 2^k possible assignments in a k-universal function C.

Example 1. The following function $C : \{0,1\}^2 \rightarrow \{0,1\}^3$ is 2-universal, since every two output coordinates have all four values:

$$\begin{array}{l} C(00) = 000 \\ C(01) = 011 \\ C(10) = 101 \\ C(11) = 110 \end{array} \tag{1}$$

□

In Section 3 we present a method for constructing k-universal circuits.

2.2 Universality of Some Known Underapproximations

Underapproximations based on restricting the inputs can be seen as functions mapping inputs of the restricted model to inputs of the original model. It is worthwhile to check how universal these functions are. Recall that if the model is unrestricted, it is n-universal, where n is the number of inputs.

- *Underapproximation by restricting a subset of the inputs to constant values.* Regardless of the method for choosing these inputs and their values, or whether it is part of a refinement process or not, it is clear that the underlying set of possible assignment vectors to the restricted model is not even 1-universal, since there are inputs that cannot have both values.
- *Underapproximation by combining inputs.* In this method the set of inputs is partitioned, and all inputs in the same partition class are forced to agree on their value. Regardless of the partitioning method, this method guarantees 1-universality, but not 2-universality, because two inputs in the same partition class cannot have all 4 valuations.

3 The PRG-Like Construction

The structure of our k-universal circuits, as mentioned earlier, were inspired by constructions of Pseudo Random Generators. PRG is a circuit that, given a short sequence of truly random bits, outputs a longer sequence of pseudo random bits. More formally:

Definition 2 (PRG). Pseudo Random Generator (PRG) *is a deterministic polynomial time function* $G : \{0,1\}^m \rightarrow \{0,1\}^n$*, where* $n > m$*, such that the following distributions are not distinguishable by circuits of size* n*:*

- *Distribution* G_n *defined as the output of function* G *on a uniformly selected input in* $\{0,1\}^m$*.*
- *Distribution* U_n *defined as the uniform distribution on* $\{0,1\}^n$*.* □

The original motivation for constructing PRG's was derandomizing probabilistic algorithms[2].

In this section we sketch briefly how the original PRG of [9] is constructed, and introduce a slightly different (random) construction that, as we prove later, provides with arbitrarily high probability, k-universal circuits. The parameter k here is almost linear in m, with practically small coefficients. Without going into the details, based on a result in [12] it can be shown that ($2^k \log n \leq 2^m$), which means that an upper bound on k is $m - \log\log n$. Hence, the circuit we construct has nearly optimal parameters.

Definition 3 (System[3]). *A family* $S = (S_1, S_2, \ldots, S_n)$ *of equally-sized subsets* $S_i \subset \{1, 2, \ldots, m\}$ *is a* (l, ρ, m, n)-system *if*

- $\forall i, \quad |S_i| = l$
- $\forall i, j \quad |S_i \cap S_j| \leq \rho$ □

Given a Boolean function $f : \{0,1\}^l \rightarrow \{0,1\}$ and a system $S = (S_1, S_2, \ldots, S_n)$, we construct the circuit $C = C(S, f)$ as follows:

- $I_C = \{i_1, \ldots, i_m\}$ are the inputs of C.
- $O_C = \{o_1, \ldots, o_n\}$ are the outputs of C.
- For $j \in \{1, \ldots, n\}$,
  - Let $I(o_j) = \{i_h : h \in S_j\}$ be a set of l inputs chosen according to the system S.
  - Set $o_j = f(I(o_j))$.

In the original paper [9] the existence of systems with "good" parameters is proved, and the PRG's are constructed based on these "good" systems using functions f that have some specific cryptographic properties. Further details are given in the above reference.

Now we define our random systems, based on which we will build k-universal circuits.

[2] For instance, a "perfect" PRG would be a function $G : \{0,1\}^{\log n} \rightarrow \{0,1\}^n$. If we have such a PRG, then we can deterministically simulate any probabilistic algorithm by going over all $2^{\log n} = n$ possible seeds for G, running the probabilistic algorithm and taking the majority vote.

[3] In the original terminology this set system is called a *Design*. We avoid this term to prevent ambiguity.

Definition 4 (Random System). *Let n, m be naturals such that $1 \leq m \leq n$. An (m,n)-*Random System *is a family $RS = (S_1, S_2, \dots, S_n)$ of n uniformly chosen random subsets $S_i \subset \{1, 2, \dots, m\}$. Namely, for every $1 \leq i \leq n$ (independently of each other), the set S_i is chosen uniformly at random out of all 2^m possible subsets of $\{1, 2, \dots, m\}$.* □

Similarly to the previous construction, we build the circuit $C = C(RS, f)$ where we set f to be the XOR function ($\oplus$). Formally,

- $I_C = \{i_1, \dots, i_m\}$ are the inputs of C.
- $O_C = \{o_1, \dots, o_n\}$ are the outputs of C.
- For $j \in \{1, \dots, n\}$,
  - Let $I(o_j) = \{i_h : h \in S_j\}$ be the randomly chosen set of inputs from RS.
  - Set $o_j = \oplus(I(o_j))$.

In the following section we prove that with arbitrary high probability these circuits are k-universal for relatively high k.

3.1 Lower Bounds on *k*

First we prove that if the family RS has certain algebraic properties, then the circuit C that is built from RS is k-universal.

Lemma 1. *Let A be an $n \times m$ Boolean matrix defined by the family RS. Formally, the entry $a_{ij} \in A$ is 1 if $j \in S_i$ and 0 otherwise. Then if every k rows of A are linearly independent*[4]*, the circuit $C = C(RS, \oplus)$ as above is k-universal.*

Proof (of Lemma 1). First notice that the i'th output of C implements a XOR function on the inputs that correspond to the '1' entries of the i'th row in the matrix A. So we can think of C as a linear transformation in field $GF(2)$ (Galois Field), induced by multiplying the matrix A with the input vector (recall that addition in $GF(2)$ is equivalent to the XOR operator). In other words, for every $\alpha_1\alpha_2\cdots\alpha_m \in \{0,1\}^m$ and $\beta_1\beta_2\cdots\beta_n \in \{0,1\}^n$, $C(\alpha_1\alpha_2\cdots\alpha_m) = \beta_1\beta_2\cdots\beta_n$ if and only if the following holds:

$$\begin{pmatrix} a_{11} & a_{12} & \dots & a_{1m} \\ a_{21} & a_{22} & \dots & a_{2m} \\ . & & & . \\ . & & & . \\ . & & & . \\ a_{n1} & a_{n2} & \dots & a_{nm} \end{pmatrix} \times \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_m \end{pmatrix} = \begin{pmatrix} \beta_1 \\ \beta_2 \\ . \\ . \\ . \\ \beta_n \end{pmatrix} . \tag{2}$$

Let $K = \{o_1, o_2, \dots, o_k\} \subset \{1, 2, \dots, n\}$ be arbitrary set of k outputs, and let $\beta_{o_1}\beta_{o_2}\cdots\beta_{o_k}$ be any partial assignment on K. Notice that for any $\alpha_1\alpha_2\cdots\alpha_m$ the value $C(\alpha_1\alpha_2\cdots\alpha_m)$ restricted to K equals $\beta_{o_1}\beta_{o_2}\cdots\beta_{o_k}$ if and only if

[4] Equivalently, every k rows of A form a full rank matrix.

$$\begin{pmatrix} a_{o_1}1 & a_{o_1}2 & \dots & a_{o_1}m \\ a_{o_2}1 & a_{o_2}2 & \dots & a_{o_2}m \\ . & & & . \\ . & & & . \\ . & & & . \\ a_{o_k}1 & a_{o_k}2 & \dots & a_{o_k}m \end{pmatrix} \times \begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_m \end{pmatrix} = \begin{pmatrix} \beta_{o_1} \\ \beta_{o_2} \\ . \\ . \\ . \\ \beta_{o_k} \end{pmatrix} . \tag{3}$$

We denote this restricted $k \times m$ matrix by B. Recall that our purpose is to prove that such an assignment $\alpha_1\alpha_2\cdots\alpha_m$ indeed exists. Here we use the fact that every k rows in A are linearly independent, and thus the matrix B is invertible. Therefore such an assignment exists, and it can be computed by:

$$\begin{pmatrix} \alpha_1 \\ \alpha_2 \\ \vdots \\ \alpha_m \end{pmatrix} = \begin{pmatrix} a_{o_1}1 & a_{o_1}2 & \dots & a_{o_1}m \\ a_{o_2}1 & a_{o_2}2 & \dots & a_{o_2}m \\ . & & & . \\ . & & & . \\ . & & & . \\ a_{o_k}1 & a_{o_k}2 & \dots & a_{o_k}m \end{pmatrix}^{-1} \times \begin{pmatrix} \beta_{o_1} \\ \beta_{o_2} \\ . \\ . \\ . \\ \beta_{o_k} \end{pmatrix} . \tag{4}$$

□

The next lemma states that with probability $1 - \delta$ (wher $\delta > 0$ is an arbitrary confidence parameter), in the matrix A defined by the family RS, every k rows are linearly independent.

Lemma 2. *Let $k > 1$, $a > 1$, $b > 1$ be natural numbers and let $\delta > 0$ be a fixed confidence parameter. Set $b = m/k$ and $a = n/m$. Let RS be a family of subsets in (m, n)-Random System and let A be the underlying matrix as above. If $b > \log(e \cdot ab(1/\delta)^{1/k}) + 1$ then with probability at least $1 - \delta$ every k rows in A are linearly independent*[5].

Before proving the lemma, we list some known useful inequalities:

(i) Let $x_1, x_2, \dots, x_n$ be non negative reals. Then $\prod_{i=1}^{n}\Big(1 - x_i\Big) > 1 - \sum_{i=1}^{n} x_i$.

(ii) $\binom{n}{k} < (\frac{en}{k})^k$.

(iii) Let m, k be naturals such that $m > k$. Then $\sum_{i=1}^{k} 2^{i-m} \leq 2 \cdot 2^{k-m}$.

Proof (of Lemma 2). According to the construction of random systems, every row in A is a random Boolean vector of length m. Let $K = \{o_1, o_2, \dots, o_k\} \subset \{1, 2, \dots, n\}$ be any sequence of k rows in A. Now we define a sequence of "bad" event indicators: $I_j = 1$ if and only if the j'th row $o_j \in K$ is a linear combination of the rows $o_1, \dots, o_{j-1}$. Obviously if $(\sum_{j=1}^{k} I_j) = 0$ then the rows in K are linearly independent. Note that in every step j, the $j - 1$ preceding vectors span

[5] $e = 2.718...$ is the Euler constant.

a linear space of size at most 2^{j-1}. Since the rows of A are chosen uniformly at random (independently of each other), we have $\Pr[I_j = 0] \geq \frac{2^m - 2^{j-1}}{2^m}$. Therefore,

$$\Pr\Big[\Big(\sum_{j=1}^{k} I_j\Big) = 0\Big] = \prod_{j=1}^{k} \frac{2^m - 2^{j-1}}{2^m} = \tag{5}$$

$$= \prod_{j=1}^{k}(1 - 2^{j-1-m}) \geq 1 - \sum_{j=1}^{k} 2^{j-1-m} \geq 1 - 2^{k-m} \ . \tag{6}$$

The last two inequalities follow from (i) and (iii). We can now conclude that

$$\Pr\Big[\Big(\sum_{j=1}^{k} I_j\Big) > 0\Big] \leq 2^{k-m} \ . \tag{7}$$

There are $\binom{n}{k} \leq (\frac{en}{k})^k$ possible sets of k rows, and by the Union Bound[6] the probability that some set of k rows is not linearly independent is at most

$$(\frac{en}{k})^k \cdot 2^{k-m} = (eab)^k \cdot 2^{(1-b)k} \leq (eab)^k \cdot 2^{-\log(eab(1/\delta)^{1/k})\cdot k} = \delta \ . \tag{8}$$

□

Sample Values of Universality. It is worthwhile to see some values of k given n, m and δ. For instance, for $n = 140$, $m = 70$ and $\delta = 0.02$ we can get $k = 10$-universality with probability at least 0.98. This means that we can reduce the number of inputs to the model by half, and still get 10-universality in a very high probability.

In general δ has negligible effect on k, hence the probability of success can be made very close to 1. The chart in Figure 2 refers to a fixed value $\delta = 0.02$. The chart shows the value of k for $n = 100, 200, \ldots, 500$, where m is sampled 9 times for each value of n, in the range $n/10 \ldots 9n/10$. It is clear from the graph that k is close to linear in m, and that it has a constant factor of about 5. In fact, the equation $b = \log(e \cdot ab(1/\delta)^{1/k}) + 1$ from Lemma 2 implies that $k \sim \frac{m}{\log(n/k)}$, which means that k is linear in m for all practical n.

Corollary 1. *Let $k > 1$, $a > 1$, $b > 1$ be natural numbers and let $\delta > 0$ be a fixed confidence parameter, such that $b > \log(e \cdot ab(1/\delta)^{1/k}) + 1$. Set $b = m/k$ and $a = n/m$. Then with probability at least $1 - \delta$, a circuit C based on the family RS of a random system as described above (with parameters m, n) is k-universal.*

Proof. By Lemma 2 we know that with these parameters, in the underlying matrix A every k rows are linearly independent with probability $1 - \delta$ or higher. On the other hand, by Lemma 1 we know that if every k rows in A are linearly independent, then the circuit $C = C(RS, \oplus)$ is k-universal. □

[6] Union Bound: For a countable set $A_1, A_2, A_3, \ldots$ of events, $\Pr\Big[\bigcup_i A_i\Big] \leq \sum_i \Pr\Big[A_i\Big]$.

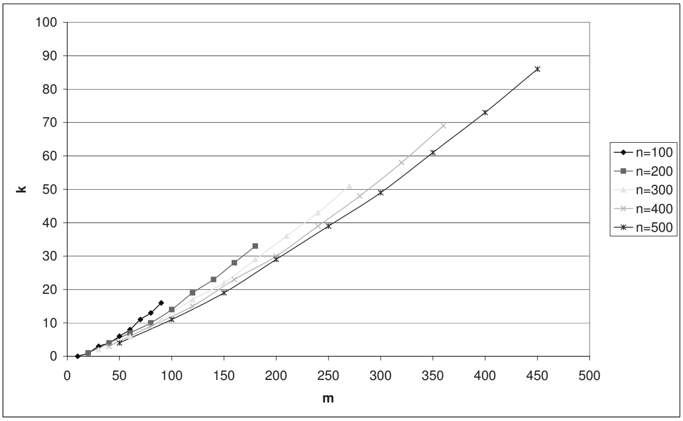

Fig. 2. The value of k for different values of m and n, and a fixed value of δ (0.02)

Based on Corollary 1, it is left to show how we construct the underapproximating model M'. The construction is as follows:

– Let $\{i_1, \dots, i_n\}$ be the primary inputs of M. Construct the k-universal circuit C based on a random system $RS = (S_1, \dots, S_n)$.
– For each $j \in \{1, \dots, n\}$, connect the j'th input of M to the j'th output of C.

The inputs of the underapproximating model M' are the m inputs of C.

3.2 A Better Lower Bounds on *k* for "Almost" *k*-Universality

In practice, given n and m the parameter of universality (k) is expected to be significantly higher than what our analytic lower bound provides. But it is quite challenging to estimate the gap between the lower bound and the actual values of k, since checking k universality of a circuit $C : \{0,1\}^m \rightarrow \{0,1\}^n$ is hard for reasonably large n, m and k. But if we slightly relax our notion of universality we can get much better bounds on k. Formally, let m, n, k and $C = C(RS, \oplus)$ be as above. Given a subset $K \subset \{1, \dots, n\}$ of k outputs, we say that the subset K is *covered* by C if for every partial assignment $\alpha_K \in \{0,1\}^k$ on K, there is a full assignment $\alpha \in \{0,1\}^m$ on the inputs of C such that $C(\alpha) \mid_K = \alpha_K$.

Definition 5 ((k, ϵ)-universality). *A circuit C is (k, ϵ)-universal if C covers at least $(1-\epsilon)\binom{n}{k}$ subsets $K \subset \{1, \dots, n\}$ of k outputs.* □

Recall that our previous bounds on k were valid for circuits that cover *all* $\binom{n}{k}$ subsets K, i.e. $(k, 0)$-universal circuits. The following result is another lower-bound, which is better than the previous one as long as ϵ is not too small.

Lemma 3. *Let $m < n$ be naturals and let $C = C(RS, \oplus)$ be a circuit as defined above. Fix $0 < \delta$, $0 < \epsilon < 1$ and set $k = \max(0, m - \log \frac{1}{\epsilon \cdot \delta})$. The circuit C is (k, ϵ)-universal with probability at least $1 - \delta$.*

Observe the implication of this result: since m is an absolute upper bound on k, it means that with a small sacrifice of universality and confidence we obtain a value close to this theoretical limit. For example, for $\delta = \epsilon = 0.1$ (and $m \geq 7$), we get $k = m - 7$, i.e., with probability at least 0.9, the circuit C is $(\max(0, m-7), 0.1)$-universal. Now consider a negligible sacrifice and failure probability, such as $\delta = \epsilon = 0.01$. In this case we get $(k, 0.01)$-universality for $k = \max(0, m - 14)$.

Proof (of Lemma 3). The proof is a simple application of Markov's inequality[7] on one of the consequences from the proof of Lemma 2. For every subset $K \subset \{1, 2, \ldots, n\}$ of size k, we define X_K as a random $0, 1$ variable, such that $X_K = 1$ if and only if the subset K is *not* covered by C. Referring to the proof of Lemma 1, the set K is covered by C if and only if the sub-matrix B that corresponds to K has full rank (otherwise the linear transformation is not injective). Then from the proof of Lemma 2 we have $\Pr[X_K = 1] \leq 2^{k-m}$. Now let

$$X = \sum_{K \subset \{1,\ldots,n\}, |K|=k} X_K$$

be the sum of these variables. By linearity of expectation[8],

$$E[X] = \sum_K E[X_K] \leq \binom{n}{k} \cdot 2^{k-m} \; , \tag{9}$$

and by Markov's inequality,

$$\Pr\Big[X \geq \epsilon \cdot \binom{n}{k}\Big] = \Pr\Big[X \geq \epsilon \cdot 2^{m-k} \cdot \binom{n}{k} \cdot 2^{k-m}\Big] \leq \frac{1}{\epsilon \cdot 2^{m-k}} = \delta \; . \tag{10}$$

From (10) we derive $k \geq m - \log \frac{1}{\epsilon \cdot \delta}$. □

4 Experimental Results

We interfaced our tool with IBM's model-checker RuleBase. We experimented with bounded model-checking of 17 different real designs (after Rulebase has applied numerous optimizations on them in the front-end, hence the relatively small number of inputs) that had previously known bugs. The tables show our results *without* an automatic refinement procedure. The reason we are giving the tables in this form is that we want to show the influence of m on run-time and chances to find the bug with each underapproximation technique. The tables show run-times in seconds until detecting the bug, for different values of m, where m in all techniques represent the number of inputs to the underapproximated

[7] Markov inequality: Let X be a random variable assuming only non-negative values. Then for all $c > 0$, $\Pr\Big[X \geq c \cdot E[X]\Big] \leq \frac{1}{c}$.

[8] Linearity of Expectation: For any n random variables $X_1, \ldots, X_n$ the following holds: $\mathrm{E}\Big[\sum_{i=1}^n X_i\Big] = \sum_{i=1}^n \mathrm{E}[X_i]$.

model. A sign '-' denotes that the bug was not found up to a bound of 100. 'TO' denotes a timeout of 6 hours.

The table in Figure 3 summarizes results with our construction, hence m is the number of inputs to the circuit. The column S denotes run-time with no underapproximation. It is clear from this table that while $m = n/10$ is too low, $m = n/5$ is high enough to find the bug in 13 out of 17 cases, and typically in less time comparing to the S column, despite the complexity of the XOR function in the PRG-like circuit. Thus, our refinement procedure is set to begin with this value. The last three designs indicate that there are cases in which underapproximation does not work (in all three methods – see Figure 4 as well). Since Rulebase activates various engines in parallel, this is not a serious issue: the contribution of a tool is mainly measured by the number of wins rather than by the average run-time. This is also the reason it is acceptable that such a method has no value if the design satisfies the property.

			(PRG) $m = ...$			
Design	inputs (n)	S	$n/2$	$n/3$	$n/5$	$n/10$
IBM#1	45	96	66	63	66	63
IBM#2	76	173	149	76	72	68
IBM#3	76	191	127	77	79	-
IBM#4	85	211	170	121	105	140
IBM#5	68	61	65	20	592	-
IBM#6	68	73	59	14	661	-
IBM#7	68	482	308	46	52	-
IBM#8	68	122	152	16	90	-
IBM#9	64	2101	1915	1966	1654	1208
IBM#10	80	1270	1392	1830	1137	-
IBM#11	83	2640	2364	2254	1845	-
IBM#12	6	8201	7191	-	-	-
IBM#13	60	942	453	432	351	-
IBM#14	218	965	735	778	510	396
IBM#15	52	1206	-	-	-	-
IBM#16	157	953	-	-	-	-
IBM#17	68	21503	TO	TO	TO	TO

Fig. 3. Run-times with the PRG construction. The second column indicates the number of inputs in the design, i.e., n. The column 'S' stands for run-times without any underapproximation.

In Figure 4 we show results for the two alternative underapproximations described in Subsection 2.2. It is clear from these tables that universality matters: both of these underapproximations need far more inputs than the PRG construction in order to find the bug. Somewhat surprisingly even in the cases they are able to find the bug, they do so in time comparable or longer than without underapproximation at all. The reason seems to be that the underapproximation

			(FIX) $m = ...$				(Group) $m = ...$			
Design	inputs (n)	S	$n/2$	$n/3$	$n/5$	$n/10$	$n/2$	$n/3$	$n/5$	$n/10$
IBM#1	45	96	246	-	-	-	223	229	227	231
IBM#2	76	173	-	-	-	-	361	446	-	-
IBM#3	76	191	373	-	-	-	168	317	-	-
IBM#4	85	211	191	317	-	-	306	289	405	-
IBM#5	68	61	-	-	-	-	410	-	-	-
IBM#6	68	73	-	-	-	-	-	-	-	-
IBM#7	68	482	-	-	-	-	561	491	-	-
IBM#8	68	122	-	-	-	-	113	-	-	-
IBM#9	64	2101	1693	-	-	-	2150	-	-	-
IBM#10	80	1270	-	-	-	-	-	-	-	-
IBM#11	83	2640	-	-	-	-	-	-	-	-
IBM#12	6	8201	-	-	-	-	-	-	-	-
IBM#13	60	942	1206	-	-	-	413	407	-	-
IBM#14	218	965	-	-	-	-	969	1102	-	-
IBM#15	52	1206	-	-	-	-	-	-	-	-
IBM#16	157	953	-	-	-	-	-	-	-	-
IBM#17	68	21503	-	-	-	-	TO	-	-	-

Fig. 4. Run-times when (left) fixing $n - m$ inputs to an arbitrary value and (right) grouping the inputs into m sets, and forcing inputs in the same set to be equal. See Section 2.2 for more details on these underapproximations. The column 'S' stands for run-times without any underapproximation.

delays the finding of the bug to deeper cycles, which in general affects negatively the run time of SAT.

4.1 Further Directions

There are various directions in which this research can progress. First, it has to be evaluated with unbounded model-checking and simulation. Simulation is insensitive to the XOR circuit, which indicates that it might show a stronger influence on the results. Second, our current implementation of refinement is very naive, as it simply increases m. There are probably better alternatives for refinement, and we leave it for future work to find them. In the case of SAT-based model checking, for example, the unsatisfiable core can guide the refinement.

Finally, the fact that in Bounded Model Checking the inputs of each time-frame are represented by different variables can be exploited for reducing m further. The PRG construction can be attached to the *unrolled* circuit. This construction will now have m inputs for $0 < m < n \cdot \mathcal{K}$, where $\mathcal{K}$ is the unrolling bound. It is very likely that errors can be found this way with a smaller set of inputs per cycle.

Acknowledgements. We thank E. Ben-Sasson, M. Shamis and K. Yorav for useful discussions.

References

1. Amla, N., McMillan, K.: Automatic abstraction without counterexamples. In: Garavel, H., Hatcliff, J. (eds.) ETAPS 2003 and TACAS 2003. LNCS, vol. 2619, Springer, Heidelberg (2003)
2. Barner, S., Grumberg, O.: Combining symmetry reduction and upper-approximation for symbolic model checking. In: Brinksma, E., Larsen, K.G. (eds.) CAV 2002. LNCS, vol. 2404, Springer, Heidelberg (2002)
3. Clarke, E., Grumberg, O., Jha, S., Lu, Y., Veith, H.: Counterexample-guided abstraction refinement. J. ACM 50(5), 752–794 (2003)
4. Clarke, E., Gupta, A., Strichman, O.: SAT based counterexample-guided abstraction-refinement. Transactions on Computer Aided Design (TCAD) 23(7), 1113–1123 (2004)
5. Glusman, M., Kamhi, G., Mador-Haim, S., Fraer, R., Vardi, M.Y.: Multiple-counterexample guided iterative abstraction refinement: An industrial evaluation. In: Garavel, H., Hatcliff, J. (eds.) ETAPS 2003 and TACAS 2003. LNCS, vol. 2619, pp. 176–191. Springer, Heidelberg (2003)
6. Grumberg, O., Lerda, F., Strichman, O., Theobald, M.: Proof-guided underapproximation-widening for multi-process systems. In: POPL '05: Proceedings of the 32nd ACM SIGPLAN-SIGACT sysposium on Principles of programming languages, pp. 122–131. ACM Press, New York (2005)
7. Hartman, A., Raskin, L.: Problems and algorithms for covering arrays. Discrete Math 284, 149–156 (2004)
8. Kurshan, R.: Computer aided verification of coordinating processes. Princeton University Press, Princeton, NJ (1994)
9. Nisan, N., Wigderson, A.: Hardness vs randomness. Journal of Computer and System Sciences 49, 146–167 (1994)
10. Ravi, K., Somenzi, F.: High-density reachability analysis. In: Proc. Intl. Conf. on Computer-Aided Design, pp. 154–158 (November 1995)
11. Ravi, K., Somenzi, F.: Hints to accelerate symbolic traversal. In: Pierre, L., Kropf, T. (eds.) CHARME 1999. LNCS, vol. 1703, pp. 250–264. Springer, Heidelberg (1999)
12. Seroussi, G., Bshouty, N.: Vector sets for exhaustive testing of logic circuits. IEEE Transactions on Information Theory, 34 (1988)
13. Williams, R., Gomes, C.P., Selman, B.: Backdoors to typical case complexity. In: IJCAI, pp. 1173–1178 (2003)