

VERA: Approximate Verification

Final Report

October 30th 2006

LRI, University Paris-South
Equipe de Logique, University Paris VII

Abstract. The VERA (Approximate Verification) research action (2003-2006) proposed approximate versions of verification. Two complementary approaches have been defined, one for the approximate verification of a qualitative property, the other for the approximate verification of a quantitative property of a probabilistic system. The approximate verification of a non-deterministic system follows the *Property testing* approach. We wish to distinguish between a correct system and a system which is far from satisfying a property. The approximate verification of a quantitative property approximates the probability that a system satisfies the property using a Monte-Carlo method.

Résumé. Le projet VERA (Verification Approchée 2003-2006) a proposé des notions d'approximation pour faciliter la vérification. Deux approches complémentaires ont été proposées, l'une pour la vérification qualitative approchée d'une propriété, l'autre pour la vérification quantitative approchée d'une propriété d'un système probabiliste.

La vérification approchée d'un système non déterministe est inspirée du *test de propriété*. On distingue un système correct d'un système loin d'être correct, avec grande probabilité, après avoir défini une mesure sur les entrées du système. La vérification quantitative approchée d'un système probabiliste approxime la probabilité qu'une propriété soit satisfaite, selon une méthode de Monte-Carlo.

1 Introduction

The Trust in Computer Systems requires verification techniques which adapt to partially known environments where the systems are used, in particular distributed networks. These systems need to satisfy various properties but need to be also robust, i.e. react in a predictive way, even if when noisy and corrupted data are exchanged.

In recent years, the use of model checking methods has considerably increased for program and protocol verification. These methods still raise many complexity problems related to the state explosion phenomenon. Classical approaches to overcome these problems are symbolic representation (OBDD), abstraction methods, and SAT-based bounded model checking. In many case studies, the resulting representation, for example OBDDs, still remains of exponential size. If the system is given as a black-box, the verification is even more difficult.

Approximation techniques have been traditionally applied to optimization problems. Property Checking introduced in [19, 7] have generalized this approach to decision problems. In the VERA project (<http://www.lri.fr/~mdr/vera/>), we defined approximate verification based on property testing. The following examples illustrate how approximate verification can be useful:

- Consider a system which tries to decide if a DNA sequence is close to a regular property. The system accesses noisy data, because the DNA sequence comes from measure instruments, and should decide correctly if a noisy input, i.e. close to an unknown perfect input, is close to a property.
- Consider a system which tries to decide if an XML file is close to a DTD. Web data, given the various natural languages used and the physical imperfections are noisy and programs which manipulate them should be proved robust, i.e. decide correctly a property for close inputs.
- Consider a system viewed as an automaton on infinite sequences, and an LTL property. Classical verification transforms the LTL property into another automaton and decides the containment property of the corresponding languages, i.e. $L_1 \subseteq L_2$. How can we verify that the languages of the automata are approximately contained, i.e. an input $x \in L_1$ is always close to an $x' \in L_2$.

Testers of property are *robust* as they apply to partially known environments. We showed how these three situations correspond to classical approximate decision problems which have efficient solutions.

- We proved the existence of Testers for regular properties of trees [15] (ICALP 2004, to appear in *Algorithmica*) with the *Tree Edit Distance with Moves*, which lead to efficient (constant time) approximate membership decisions for regular properties.
- We proved the existence of Testers for a group property of matrices [6] (FOCS 2005) with an Edit Distance on Matrices.
- We gave efficient Equivalence Testers for regular properties [5] (LICS 2006).

For the two previous examples, the randomized techniques verify a regular property of words and trees in constant time and only sample the inputs. For the third example on approximate equivalence, the proposed technique operates in polynomial time, whereas the exact equivalence is exponential.

Model checking techniques can also be used to verify quantitative properties of probabilistic systems, such as performance and reliability in communication or security protocol. General techniques of PRISM [11] rely on solving large linear systems, and Monte Carlo techniques to estimate the probabilities have also been proposed [10] in this context.

2 Members

The project was common to two groups which met every two or three weeks either at LRI (University Paris-South) or at Chevaleret (University Paris VII).

2.1 LRI, Algorithms and Complexity

1. M-C. Gaudel. Professor University Paris-South. Software Engineering, Testing.
2. F. Magniez, CR CNRS. Quantum Algorithms and Testing.
3. M. de Rougemont, Professor University Paris II, Logic, Complexity, Uncertainty.
4. M. Santha, DR CNRS, Quantum algorithms, Testing, Theoretical Computer Science.
5. Claudia Hess, Ph.D. Student.
6. Adrien Viellerivière, Ph.D. Student.

Guests:

7. E. Fischer, Professor Technion (Israel),
8. P. Singh, IIT Kanpur, India,
9. Abishek Kumar Mall, IIT Kanpur, India.

2.2 Equipe de Logique, University Paris VII

1. Richard Lassaigne, Assistant Professor University Paris 7, Logic, Complexity, Probabilistic Model Checking.
2. S. Peyronnet, Assistant Professor EPITA, Probabilistic Model Checking.

3 Approximate Verification of Properties

Interactive proofs, Probabilistic Checkable proofs, and Testers allow for randomized techniques which efficiently verify properties. In the case of testers, we want to verify if $Prob_{\Omega}[\psi] = 1$ or if the system is far from satisfying $Prob_{\Omega}[\psi] > 2/3$ and Ω is the probabilistic space of the algorithm, i.e. the samples in the case of a tester.

Let \mathbf{K} be a class of finite structures U , such as words or ranked ordered trees. A property P is a subset of \mathbf{K} . A formula F in the language of \mathbf{K} is defined in some Logic as First-Order Logic or Monadic Second-Order Logic, and we use the equivalence between regular properties of words (resp. trees) and Monadic Second Order Logic. We want to extend the *satisfiability relation*, i.e. $U \in \mathbf{K}$ satisfies P , or $U \models P$, to $U \models_{\varepsilon} P$ and extend the equivalence relation between two formulas $F_1 \equiv F_2$ to $F_1 \equiv_{\varepsilon} F_2$, using a distance between structures

An *elementary operation* on a word w is either an insertion, a deletion or a modification of a letter, or the *move* of a subword of w into another position. The *edit distance with moves* $\text{dist}(w, w')$ between w and w' is the minimal number of elementary operations on w to obtain w' . The distance between w and a language L , noted $\text{dist}(w, L)$ is the minimum distance $\text{dist}(w, w')$ for $w' \in L$. The above distance is extended to trees by

generalizing the elementary operations. An *elementary operation* (see Figure 1) on an unranked ordered tree T is either an insertion or a deletion of a node, the substitution of a label, or the move of an entire subtree [15]. More precisely, a *move* (u, v, i) moves in one step u (and the corresponding subtree rooted at u) to be the i -th successor of v , shifting all the j successors of v for $j \geq i$ by one. As a consequence, the new parent of u is now v . When trees are specified to be r -ranked, we will restrict ourselves only to deletions and moves that gives a r -ranked tree.

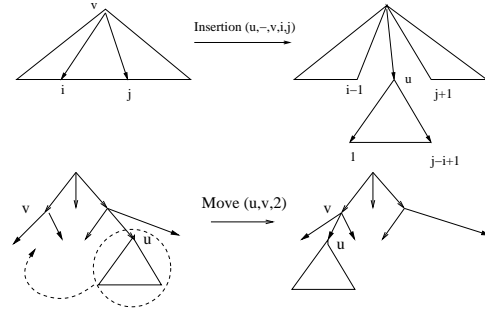


Figure 1. Elementary operations on trees.

3.1 Approximate Satisfiability and Equivalence on Words

We define the notion of approximate satisfiability as in property testing [19, 7], whose background goes back to self-testers [2]. Let \mathbf{K} be a class of finite structures U with a distance dist between structures. Since property testing is an approximate notion of verification for dense instances, or equivalently for normalized distances, we first define a suitable notion of closeness for any distance dist . We say that $U, U' \in \mathbf{K}$ are ε -close if their distance is at most $\varepsilon \times M$, where M is a normalization factor, that is the maximum of $\text{dist}(V, V')$ when V and V' range over \mathbf{K} and have respectively same sizes than U and U' . They are ε -far if they are not ε -close. For words and trees, M is set to the maximal size of the respective structures, since this is always the order of M . (For dense graphs, M is the square of the maximal size of the respective structures.)

Definition 3.1. Let P be a property on \mathbf{K} . A structure $U \in \mathbf{K}$ ε -satisfies P , or $U \models_{\varepsilon} P$ for short, if U is ε -close to some $V \in P$.

Note that $U \not\models_{\varepsilon} P$ means that U is ε -far to every V such that $V \models_{\varepsilon} P$.

Definition 3.2 (Tester [7]). Let $\varepsilon > 0$. An ε -tester for a property $P \subseteq \mathbf{K}$ is a randomized algorithm A such that, for any structure $U \in \mathbf{K}$ as input:

- (1) If $U \models P$, then A accepts with probability at least $2/3$;
- (2) If $U \not\models_{\varepsilon} P$, then A rejects with probability at least $2/3$.

If in addition the algorithm is guaranteed to always accept if $U \in P$, then we call it a *one-sided error* ε -tester. When (1) is relaxed for some $0 < \varepsilon_0 < \varepsilon$: (1') If $U \models_{\varepsilon_0} P$, then A accepts with prob. at least $2/3$; we say that the tester is a (*tolerant*) $(\varepsilon_0, \varepsilon)$ -tester [16]. Approximation algorithms are related to tolerant testers [16].

A *query* to a structure U depends on the model for accessing the structure. For a word w , a query is asking for the value of $w[i]$, for some i . For a tree T , a query is asking for the value of the label of i , for some i , and potentially for the index of its ancestor and its j -th successor, for some j . The *query complexity* is the number of queries made to the structure. The *time complexity* is the usual definition, where we assume that the following operations are performed in constant time: arithmetic operations, a uniform random choice of an integer from any finite range not larger than the input size, and a query to the input.

Definition 3.3. A property $P \subseteq \mathbf{K}$ is testable, if there exists a randomized algorithm A such that, for every real $\varepsilon > 0$ as input, $A(\varepsilon)$ is an ε -tester of P and the query and time complexities of the algorithm A only depend on ε .

It has been observed [1] that regular properties of words are testable for the Hamming distance, and in [15] that regular properties of trees are testable for the Tree Edit distance with Moves. We extend those previous definitions to any formula F that defines P . The notion of equivalence testing for two properties P_1 and P_2 and in particular when the properties are definable by two formulas F_1 and F_2 over a logic \mathcal{L} is defined as follows.

Definition 3.4. Let $\varepsilon > 0$. Let P_1, P_2 be two properties.

- (1) P_1 is ε -contained in P_2 , if all but finitely many structures which satisfy P_1 are ε -close to P_2 ;
- (2) P_1 is ε -equal to P_2 , if both P_1 is ε -contained in P_2 and P_2 is ε -contained in P_1 ;
- (3) Let F_1 and F_2 be two formulas which define the properties P_1 and P_2 respectively. F_1 is ε -equivalent to F_2 , or $F_1 \equiv_\varepsilon F_2$ for short, if P_1 is ε -equal to P_2 .

Definition 3.5 (Equivalence tester [5]). Let $\varepsilon > 0$. A (deterministic) ε -equivalence tester for \mathcal{L} is a (deterministic) algorithm $A(\varepsilon)$ such that, given as input $F_1, F_2 \in \mathcal{L}$ and ε :

- (1) If F_1 and F_2 define the same property, then A accepts;
- (2) If F_1 and F_2 are not ε -equivalent, then A rejects.

3.2 Application to Trees

Consider only 2-ranked labeled ordered trees but our results can be extended to any ranked trees. Let Σ be the finite label alphabet. The *size of a tree* is the number of its nodes, which we will denote by n . The *degree of a node* is the number of its successors. Let k be an integer and $\varepsilon = 1/k$.

We define the k -compression of a tree T , which basically consists of removing every node whose subtree has size $\leq k$, and encoding the removed subtrees into the labels of their ancestor nodes. This compression leads naturally to a word $w(T)$ that encodes T such that $\text{u-stat}(w(T))$ can be approximately sampled from samples on T . Then some of our previous results on words can be extended to trees.

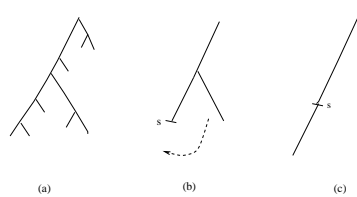


Figure 2. k -compression and word embedding.

Compression Initial labels are named *simple labels*. We introduce new *tree labels* for leaves l whose purpose is to encode a subtree from l . We also interpret a simple label on a leaf as a tree label. A *mixed label* is an ordered pair of a simple label and a tree label, whose tree label encodes the subtree of the corresponding successor. Notice that an internal node might have either a simple label or a mixed label. Such a labeled tree *encodes* T when expanding the tree labels (from leaves and from the tree components of mixed labels) leads to the initial tree T .

The *size of a simple label* is 1. The *size of a tree label* is the size of the tree that it encodes. The *size of a mixed label* is the sum of the sizes of its labels, that is 1 plus the size of its tree label part.

Definition 3.6. Let T be a tree and $k \geq 1$ be an integer. The k -compression T_k of T is the tree encoding of T such that each tree label has the minimal possible value in $[k, 2k - 1]$.

The k -tree alphabet (denoted $\Sigma^{(k)}$), is the set of any possible labels that come from a k -compression, that is when tree labels encode trees of size in $[k, 2k - 1]$. Therefore $|\Sigma^{(k)}| = |\Sigma|^{O(k)}$. The new label of a node v can be computed using $O(k)$ queries to T by a procedure **Encode**(T, v, k) that either computes its label on T_k , or rejects if v is not anymore in T_k .

In fact the k -compression of a tree T is almost a word, as the number of remaining 2-degree nodes in T_k are small. T_k has at most εn 2-degree nodes. The u-stat embedding generalizes from words to trees and allow for similar efficient testers.

4 Main results

One of the conclusions of VERA is that one can associate with a word x of length n a statistical embedding $\text{stat}(x)$ which only depends on an ε parameter. The *block statistics* $\mathbf{b}\text{-stat}(w)$ is the statistics vector of the block letters of w , that is, for every $u \in \Sigma^k$, the value $\mathbf{b}\text{-stat}(w)[u]$ is equal to the probability that for a uniformly random choice of $j \in \{1, \dots, n/k\}$ we get $w[j]_b = u$. The *uniform distribution* $\mathbf{u}\text{-stat}(w)$ corresponds to a uniform and random choice of a subword of size k of w . For example, for binary words, if $k = 2$ (and so $\varepsilon = 1/k = 0.5$), there are 4 possible subwords of length 2, which we take in lexicographic order. For the binary word $w = 000111$, $\mathbf{b}\text{-stat}(w) = (1/3, 1/3, 0, 1/3)$, whereas $\mathbf{u}\text{-stat}(w) = (2/5, 1/5, 0, 2/5)$, as there are 2 blocks 00, 1 block 01, no block 10 and 2 blocks 11 among the possible 5 blocks.

The set of $\text{stat}(x)$ of words x which are ε -close to a non deterministic automaton A , is a union of polytopes H with the following properties.

- if x is accepted by A then $\text{stat}(x)$ is ε -close to H .
- if x is ε -far from A , then $\text{stat}(x)$ is ε -far from H .
- if x is ε -far from A , the geometrical distance is close to the structural distance.

The construction of H is polynomial in the size A and we can visualize this embedding with the figure below.

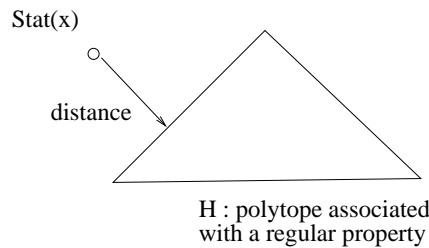


Figure 3. The polytope H associated with an automaton A and a $\text{stat}(x)$ point.

In order to approximately decide if $x \in L$, we estimate $\mathbf{u}\text{-stat}(x)$ with N samples and N only depends on ε . We compare the point $\mathbf{u}\text{-stat}(x)$ with H , associated with L , i.e. whether $\mathbf{u}\text{-stat}(x) \in H$ or $\mathbf{u}\text{-stat}(x)$ is ε -far from H , and decide if $x \in L$ or x is ε -far from L . If we want to decide if $L_1 \subseteq L_2$, we compute H_1 and H_2 in polynomial time as a function of the length of the regular expressions, and verify that $H_1 \subseteq_\varepsilon H_2$.

This principle generalizes to trees, through the Tree embedding. This principle is the core of the approximate verification methods [5] which allow for:

- Approximate verification of a property in time $O(1)$, which only depends on ε .

- Approximate equivalence of two properties in polynomial time.

This Approximate equivalence is the basis for an efficient Approximate Model Checking method, polynomial in the size of the representation of the properties.

5 Approximate Quantitative Verification

Model checking has been successfully used to verify if the behavior of concurrent and reactive systems satisfy some correctness properties, which are usually expressed in temporal logic. While many important system properties can be studied in this framework, others such as reliability and performance, require instead a probabilistic characterization of the system.

For probabilistic verification, time complexity is in general polynomial in the size of the model and polynomial or exponential in the size of the formula. As well as for classical model checking, symbolic representation methods have been generalized in the probabilistic framework. However, in spite of using these techniques, for many examples the verification is limited to small values of the main parameters of the model, due to the space complexity of the representation. Thus it seems natural to ask the question: can probabilistic verification be efficiently approximated?

5.1 Randomized Approximation Schemes

In [10], we defined some efficient algorithms to approximate satisfaction probabilities of monotone *LTL* formulas against probabilistic transition systems. For example, reachability properties are monotone and safety properties can be expressed as negation of monotone properties. These randomized approximations, with absolute error, allow to obtain two types of results, of independent interest for practical verification:

- polynomial time randomized approximation of bounded-time satisfaction probabilities,
- randomized approximation of non bounded satisfaction probabilities.

Although in the last case we can not assure polynomial time in the size of a succinct representation of the model, the main advantage is to eliminate space complexity by using a Monte-Carlo method and efficiently bounding sample size.

5.2 General non approximability result

There are serious complexity reasons to think that probabilistic verification can not be efficiently approximated in general. In [12], we proved that there is no fully polynomial randomized approximation scheme, with relative error, for the problem of computing satisfaction probability of a general *LTL* formula, unless some unlikely complexity hypothesis is satisfied. We obtained this result by a reduction of the problem of counting satisfying assignments for a SAT formula to the problem of counting finite execution paths, of given length, whose extensions satisfy an associated *LTL* formula.

5.3 Approximate Probabilistic Model Checker

In 2003, we started the design of a tool that implements the randomized approximation scheme with additive error described above. This tool [10], called APMC, is freely available under GPL. APMC is now a probabilistic distributed model checker that uses a distributed computation model in order to speed up the verification process by distributing the **Random Path** function on a cluster of workstations [8]. Extensive tests with hundreds of

machines were done. We have also collaborated with the Kwiatkowska's team in Birmingham to achieve the integration of APMC with PRISM, that is the main model checking tool for the verification of quantitative properties against probabilistic systems.

Numerous case studies have been done, such as the verification of various probabilistic distributed algorithms (mutual exclusion, dining philosophers, leader election...) [10], of the IEEE 802.3 CSMA/CD protocol [14], that is part of the ethernet protocol, and of very large sensor networks [4]. Recently, we have extended the random path generator of APMC to the verification of continuous time Markov chains [9].

6 Extension to distributed systems

In a first step, we can view Protocols as products of Automata or Reactive systems. We tried to extend the approximation to traces, so that close traces assure close behaviors. We can view a run as a two dimensional word, growing horizontally with the time and vertically with the number of agents. Unfortunately, we were not able to find meaningful distances on these objects.

We can also view Protocols as an N-players game, where utilities are associated with the players. Nash Equilibria are hard to compute in general but can often be approximated by strategies of small support. Properties of the Nash equilibria are mostly NP-hard but can be approximated in some cases, and we currently investigate testers in this context.

Protocols can also be considered as Evolutionary games, i.e. dynamical systems which may converge to a stable situation. It is a fundamental problem to predict the stability and the properties that these stable equilibria may satisfy but the *right approximation* is elusive at this point. We need to restrict the Protocols, as Mechanisms do in defining utilities associated with adversary behaviors.

7 Software developments of the VERA project

There have been two independent software developments:

- A Corrector for regular tree properties [3] has been implemented following the theoretical approach. This has led to the practical XML Corrector:

<http://www.lri.fr/~mdr/xml/>

This tool is being used in new research projects, for Data Exchange [17], Query Answering [18] and Structural Ranking.

- APMC [9], which estimates probabilities associated with properties on a probabilistic system with Monte-Carlo methods.

There is a need for a tool which would combine the two approaches, i.e. would decide if an input x is close to an input x' such that the probability that x' satisfies a property is greater than a .

8 Future research

The classical Yannakakis verification schema [21] of figure 4, distinguishes the various possible tasks: Model Checking, Black Box Checking and Conformance Testing. In the case of Black Box Checking, there are close connections with learning methods, as a first approach is to learn the model of the Black Box from samples. The

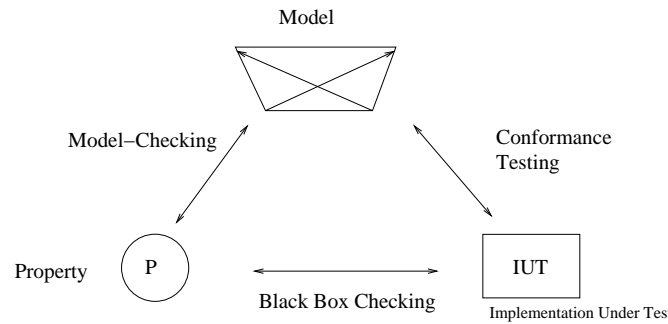


Figure 4. Checking and Testing a Model and a Black Box.

close connection between Property Testing and Learning allows us to propose a general approach to approximate Black Box Checking and Conformance Testing.

We concentrate on four specific tasks:

- Approximate Black Box Testing of Automata: when an automaton is given as a Black Box, how do we Check if it approximately satisfies a property?
- Approximate Black Box Testing of probabilistic Automata: when a probabilistic automaton is given as a Black Box, how do we Check if it approximately satisfies a quantitative property?
- Comparison with other measures: many recent approaches aim at extending bisimulation to probabilistic systems and we will compare our approximate equivalence with these notions.
- Generalizations to Transition Systems: extend approximate verification to models which are more general than automata.

References

- [1] N. Alon, M. Krivelevich, I. Newman, and M. Szegedy. Regular languages are testable with a constant number of queries. *SIAM J. Comp.*, 30(6):1842–1862, 2000.
- [2] M. Blum and S. Kannan. Designing programs that check their work. *J. ACM*, 42(1):269–291, 1995.
- [3] U. Boobna and M. de Rougemont, Correctors for XML Data, *XSym*, LNCS 3186: 97-11, 2004.
- [4] M. Cadilhac, T. Herault, R. Lassaigne, S. Peyronnet, and Sebastien Tixeuil. Evaluating complex MAC protocols for sensor networks with APMC. *Proc. 6th Int. Workshop on Automated Verification of Critical Systems*, 2006.
- [5] E. Fischer, F. Magniez and M. de Rougemont. Approximate Satisfiability and Equivalence. *IEEE Logic in Computer Science*, 2006.
- [6] K. Friedl, G. Ivanyos, M. Santha, Efficient Testing of groups *ACM STOC 2005*
- [7] O. Goldreich, S. Goldwasser, and D. Ron. Property testing and its connection to learning and approximation. *J. ACM*, 45(4):653–750, 1998.
- [8] G. Guirado, T. Herault, R. Lassaigne and S. Peyronnet. Distribution, approximation and probabilistic model checking. *Proc. Int. Workshop on Parallel and Distributed Methods in verification*, ENTCS 135(2): 19-30, 2006.

- [9] T. Herault, R. Lassaigne, and S. Peyronnet. APMC 3.0: Approximate verification of Discrete and Continuous Time Markov Chains. *Proc. 3rd Int. Conf. on Quantitative Evaluation of Systems*, 2006.
- [10] T. Herault, R. Lassaigne, F. Magniette and S. Peyronnet. Approximate Probabilistic Model Checking. *Proc. 5th VMCAI*, LNCS n2937, 73–84, 2004.
- [11] M. Kwiatkowska, G. Norman and D. Parker. Probabilistic Symbolic Model Checking with PRISM: A Hybrid Approach. *Proc. TACAS'02*, LNCS n2280 , 52–66, 2002.
- [12] R. Lassaigne, and S. Peyronnet. Probabilistic verification and approximation. *Proc. 12th Workshop on Logique, Language, Information and Computation*, ENTCS 143: 101-114, 2006.
- [13] D. Lee and M. Yannakakis, Principles and methods of Testing Finite State Machines: a survey, *The Proceedings of IEEE*, 84 (8): 1089-1123, 1996.
- [14] M. Duflot, L. Fribourg, T. Herault, R. Lassaigne, F. Magniette, S. Messika, S. Peyronnet, and C. Picaronny. Probabilistic model checking of the CSMA/CD protocol using PRISM and APMC. In *Proc. 4th Int. Workshop on Automated Verification of Critical Systems*, ENTCS 128(6): 195-214, 2005.
- [15] F. Magniez and M. de Rougemont. Property testing of regular tree languages. In *Proc. ICALP*, pp. 932–944, 2004. To appear in *Algorithmica*.
- [16] M. Parnas, D. Ron, and R. Rubinfeld. Tolerant property testing and distance approximation. TR04-010, ECCC, 2004.
- [17] M. de Rougemont and A. Vielleribière, Approximate Data Exchange, *ICDT*, 2007.
- [18] M. de Rougemont and A. Vielleribière, Approximate Schemas and Query Answering, *ISIP*, 2005.
- [19] R. Rubinfeld and M. Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM J. Comp.*, 25(2):23–32, 1996.
- [20] M.Y. Vardi. Automatic verification of probabilistic concurrent finite-state programs. *Proc. 26th Annual Symposium on Foundations of Computer Science*, pages 327–338, 1985.
- [21] M. Yannakakis. Testing, Optimization and Games. *Proc. 31st ICALP*, pages 28-45, 2004.