Probabilistic Abstraction for Model Checking: an Approach Based on Property Testing*

Sophie Laplante† Richard Lassaigne‡ Frédéric Magniez§ Sylvain Peyronnet¶
Michel de Rougemont‖

Abstract

The goal of model checking is to verify the correctness of a given program, on all its inputs. The main obstacle, in many cases, is the intractably large size of the program's transition system. Property testing is a randomized method to verify whether some fixed property holds on individual inputs, by looking at a small random part of that input. We join the strengths of both approaches by introducing a new notion of probabilistic abstraction, and by extending the framework of model checking to include the use of these abstractions.

Our abstractions map transition systems associated with large graphs to small transition systems associated with small random subgraphs. This reduces the original transition system to a family of small, even constant-size, transition systems. We prove that with high probability, "sufficiently" incorrect programs will be rejected (ε-robustness). We also prove that under a certain condition (exactness), correct programs will never be rejected (soundness).

Our work applies to programs for graph properties such as bipartiteness, k-colorability, or any $\exists\forall$ first order graph properties. Our main contribution is to show how to apply the ideas of property testing to syntactic programs for such properties. We give a concrete example of an abstraction for a program for bipartiteness. Finally, we show that the relaxation of the test alone does not yield transition systems small enough to use the standard model checking method. More specifically, we prove, using methods from communication complexity, that the OBDD size remains exponential for approximate bipartiteness.

1 Introduction

The verification of programs is a fundamental problem in computer science, where logic, complexity and combinatorics have brought new ideas which have been influential in practical applications. We combine two general methods: model checking, where one formally proves that a program is correct for all its inputs, up to a given length, and property testing, where a randomized algorithm makes random local checks within a particular input to decide if this input has a given property. Our approach brings the notions of sampling and approximation from property testing to model checking.

Model checking is an algorithmic method for deciding if a program with bounded inputs, modeled as a transition system, satisfies a specification, expressed as a formula of a temporal logic such as CTL or CTL* [10]. This verification can be carried out, for CTL, in time linear in both the size of the transition system and of the specification. For CTL* it is still linear in the size of the transition system, but exponential in the size of the specification [8]. However, a program given in a classical programming language, like C, converted to a transition system, typically undergoes an exponential blowup in the size of the input. Symbolic model checking [18, 10] addresses this problem by using ordered binary decision diagrams [5, 7] (OBDDs,

*Preliminary version of this paper appeared in *Proceedings of 17th IEEE Symposium on Logic in Computer Science*, pages 30–39, 2002. Research partially supported by LRI, the EU 5th framework program RAND-APX IST-1999-14036, the Alliance Partnership Programme 05690ZH, and by CNRS/STIC 01N80/0607 grant.

†LRI, UMR 8623 CNRS, Université Paris–Sud, 91405 Orsay, France. Email: laplante@lri.fr

‡Equipe de Logique, UMR 7056 CNRS, Université Paris 7, France. Email: lassaign@logique.jussieu.fr

§CNRS–LRI, UMR 8623 Université Paris–Sud, 91405 Orsay, France. Email: magniez@lri.fr

¶Epita, France. Email: Sylvain.Peyronnet@lrde.epita.fr

‖Université Paris II and LRI, Université Paris–Sud, 91405 Orsay, France. Email: mdr@lri.fr

or equivalently read-once branching programs with an ordering restriction on the variables), which in many practical cases provide a compact representation of the transition system. Nevertheless, in some cases, such as programs for integer multiplication or bipartiteness, the OBDD size remains exponential.

The abstraction method [9] provides a solution in some cases when the OBDD size is intractable. By way of an abstraction, a large transition system is approximated by a smaller one, on which the specification can be efficiently verified. A classical example is multiplication, where modular arithmetic is the basis of the abstraction. Our goal is to extend the range of abstractions to programs for a large family of graphs properties using randomized methods.

In the late eighties the theory of *program checking* and *self-testing/correcting* was pioneered by the work of Blum and Kannan [3], and Blum, Luby and Rubinfeld [4]. This theory addresses the problem of program correctness by verifying a coherence property (such as linearity) between the outputs of the program on randomly selected inputs. Rubinfeld and Sudan [19] formulated the notion of *property testing* which arises in every such tester. One is interested in deciding whether an object has a global property ϕ by performing random local checks, or queries, on it. The goal is to distinguish with sufficient confidence between objects that satisfy ϕ and those that are ε-far from any objects that satisfy ϕ, for some confidence parameter $\varepsilon > 0$, and some distance measure. The surprising result is that when ε is fixed, this relaxation is sufficient to decide many properties with a sub-linear or even a constant number of queries.

Goldreich, Goldwasser, and Ron [12, 14, 13] investigated property testing for several graph properties such as k-colorability. Alon, Fischer, Krivelevich, and Szegedy [2] showed a general result for all first order graph properties of type $\exists\forall$.

We identify a notion which is implicit in many graph property testers: a graph property ϕ is *ε-reducible to ψ* if testing ψ on small random subgraphs suffices to distinguish between graphs which satisfy ϕ, and those that are ε-far from satisfying ϕ. Our goal will be to distinguish with sufficient confidence between programs that accept only graphs that satisfy ϕ and those which accept some graph that is ε-far from any graph that satisfies ϕ. We introduce *probabilistic abstractions* which associate to a program small random transition systems. We show that for probabilistic abstractions based on ε-reducibility this goal can be achieved.

In Section 2 we review basic notions of model checking and property testing, and define ε-reducibility (**Definition 2.6**). In Section 3, we introduce the notion of probabilistic abstraction (**Definition 3.1**). To be useful, an abstraction must preserve the behavior of the program. An abstraction is *sound* (**Definition 3.3**) if it preserves program correctness. It is *ε-robust* (**Definition 3.2**) if correctness on the abstraction implies that the original program does not accept any graph that is ε-far from any graph that satisfies ϕ. The latter is an extension to abstraction of robustness introduced in [19]. We show how to derive a probabilistic abstraction using ε-reducibility (Section 3.4). We give a generic proof of ε-robustness for a large class of specifications (**Theorem 3.5**). Moreover, we give a sufficient condition for soundness to hold (**Theorem 3.7**). We establish the applicability of our method by applying it to a program for bipartiteness. On the one hand, we show how to construct a robust and sound abstraction on a specific program for testing bipartiteness (**Corollary 3.9**) and other temporal properties. On the other hand, in Section 4 we show that abstraction is necessary, in the sense that the relaxation of the test alone does not yield OBDDs small enough to use the standard model checking method. More specifically, we prove, using methods from communication complexity [15], that the OBDD size remains exponential for approximate bipartiteness (**Theorem 4.2**). This lower bound may also be of independent interest.

2 Framework and preliminaries

2.1 Programs and transition systems

We will use transition systems to represent all possible executions of a given program.

Definition 2.1. *A* transition system *is a triple $M = \langle S, I, R\rangle$ where S is the* set of states, *$I \subseteq S$ is the* set of initial states *and $R \subseteq S \times S$ is the* transition relation.

To simplify the discussion we only consider programs which implement boolean functions. They will be written in a simple language that manipulates bit, bounded integer and finite array variables, using basic instructions: while statements, conditionals, assignments and a `get` instruction which allows the user to

interact with the program. The *input variables* correspond to the function's input. An implicit variable `ack` is set to *false* at the beginning of the program and is set to *true* at the end of the computation. An implicit variable `ret` is defined together with the instruction `RETURN` where `RETURN b` sets `ret` to `b` and `ack` to *true*. To verify properties on the behavior of a program, we must know values of the variables at certain points of the program, called *control points*. The control points are at the beginning of lines labeled by integers. An implicit variable `PC` contains the label value of the last visited control point.

Let P be such a program with a finite set of variables $\{v_1, \ldots, v_n\}$ including the implicit variables `PC`, `ack`, and `ret`. Each variable v_i ranges over a (finite) domain D_i. We define the *transition system of* P. A *state of* P is an n-tuple $s=(s_1, \ldots, s_n)$ corresponding to an assignment of variables $v_1, \ldots, v_n$ at a control point during a computation. The *set of states of* P is $S=D_1 \times \ldots \times D_n$. The *initial states of* P are all the possible states before any computation starts, and the *transition relation of* P is the set of all possible transitions of the program between two control points. When the program terminates, the transition system loops with an infinite sequence of transitions on the final state.

Model checking does not manipulate transition system directly; it manipulates a logical representation of the transition system, expressed as a set of relational expressions. A *relational expression* is a formula of first-order logic built up from the programming language's constants and basic operators (such as $+$, $-$, and $=$). We always assume that relational expressions are in negation normal form, that is, negations pushed down to the atomic level.

Definition 2.2. *Let $\mathcal{I}$ (resp. $\mathcal{R}$) be a relational expression on S (resp. $S \times S$). Then $(\mathcal{I}, \mathcal{R})$ is a* representation *of a transition system $\langle S, I, R\rangle$ if and only if $I = \{s \in S : \mathcal{I}(s)=true\}$ and $R = \{(s, s') \in S \times S : \mathcal{R}(s, s')=true\}$.*

Example. We give a sample toy program, with its transition system and a specification of its behavior, as we define in the following paragraphs.

```
FUNCTION GUESS
  INPUT a : BOOLEAN
  VAR b : BOOLEAN
1: get(b)
2: IF (a=b) RETURN true
   ELSE     RETURN false
```

The program variables are `a` and `b` with the implicit variables `PC`, `ack`, and `ret`. A state of the program is a 5-tuple $(\mathtt{PC}, \mathtt{ack}, \mathtt{ret}, \mathtt{a}, \mathtt{b})$. A transition of the program is a pair of states $\big((\mathtt{PC}, \mathtt{ack}, \mathtt{ret}, \mathtt{a}, \mathtt{b}), (\mathtt{PC}', \mathtt{ack}', \mathtt{ret}', \mathtt{a}', \mathtt{b}')\big)$. The relational expression for the initial states of the program is $(\mathtt{PC} = 1) \wedge (\mathtt{ack} = \mathit{false})$. The relational expression for the transition relation of the program is defined as the disjunction of the following three formulas:

$$(\mathtt{PC} = 1) \wedge (\mathtt{PC}' = 2) \wedge (\mathtt{ack}' = \mathtt{ack}) \wedge (\mathtt{ret}' = \mathtt{ret}) \wedge (\mathtt{a}' = a),$$
$$(\mathtt{PC} = 2) \wedge (\mathtt{PC}' = 2) \wedge (\mathtt{ack}' = \mathit{true}) \wedge (\mathtt{ret}' = \mathit{true}) \wedge (a = b) \wedge (\mathtt{a}' = \mathtt{a}) \wedge (\mathtt{b}' = \mathtt{b}),$$
$$(\mathtt{PC} = 2) \wedge (\mathtt{PC}' = 2) \wedge (\mathtt{ack}' = \mathit{true}) \wedge (\mathtt{ret}' = \mathit{false}) \wedge (a \neq b) \wedge (\mathtt{a}' = \mathtt{a}) \wedge (\mathtt{b}' = \mathtt{b}).$$

Due to user interaction, $\mathtt{b}'$ does not appear in the first formula, and the first transition is therefore nondeterministic. The following CTL* formula (see Section 2.2) is a specification of the behavior of the program `GUESS`:

$$\forall\Big(\neg\mathtt{ack}\,\mathbf{U}\,\mathtt{ack}\wedge\Big(\big(\mathtt{ret}\wedge(\mathtt{a}{=}\mathtt{b})\big)\vee\big(\neg\mathtt{ret}\wedge(\mathtt{a}{\neq}\mathtt{b})\big)\Big)\Big)$$

2.2 Temporal logic and model checking

To express the desired behavior of a transition system (associated with a program), we use a branching-time temporal logic. All our results will be stated for the temporal logic CTL*. We refer the reader to [10] for more details on CTL*, but briefly, formulas of CTL* are defined inductively from a set of atomic propositions and built up by boolean connectives ($\neg$, $\wedge$, $\vee$), path quantifiers $\forall$ ("for all paths") and $\exists$ ("for some path"), temporal operators $\mathbf{X}$ ("next") and $\mathbf{U}$ ("until"). The *Future* operator $\mathbf{F}$ is such that $\mathbf{F}\psi$ iff $(\mathrm{true}\mathbf{U}\psi)$.

In our framework, the *atomic propositions* are $(v_i = d)$ where v_i is a variable which corresponds to the ith coordinate of a state, and d is any constant.

Let M be a transition system with representation $(\mathcal{I}, \mathcal{R})$ and let Θ be a CTL* formula. Let us briefly review the symbolic model checking method [10] for $M \models \Theta$. Notice that in model checking, the notation $M \models \Theta$ is shorthand for $M, s \models \Theta$ for every initial state s. The verification proceeds in three steps. This process is fully algorithmic, in contrast with methods which require human assistance. First, OBDD representations [5, 7] are constructed for $\mathcal{R}$ and $\mathcal{I}$. Then, an OBDD $\mathrm{check}(\mathcal{R}, \Theta)$ is constructed, whose entries are states of S, such that for every $s \in S$, $(\mathrm{check}(\mathcal{R}, \Theta)(s) = \mathit{true}) \iff (M, s \models \Theta)$. Finally, the verification of $M \models \Theta$ is achieved by checking the validity of the OBDD of $\big(\neg\mathcal{I} \vee \mathrm{check}(\mathcal{R}, \Theta)\big)$.

When the resulting OBDD is polynomial in size, the verification can be carried out in polynomial time. A typical example where this is not the case is multiplication, since any OBDD for multiplication has exponential size [6]. In the next section, we will see how abstraction has been used to successfully overcome this problem in some cases, including for multiplication.

2.3 Abstractions

The use of an *abstraction* [9] helps in some cases to overcome the problem of intractably large OBDDs. The objective of abstractions is to replace the transition system with an abstract version which is smaller, but sufficient for verifying the specification on the original system. For each variable, a surjection is used to reduce the size of the domain, and transitions are made between the resulting equivalence classes, as we define below.

Definition 2.3. *([9]) Let $M = \langle S, I, R\rangle$ be a transition system, where $S = D_1 \times \ldots \times D_n$. An* abstraction for M *is a surjection $h : S \to \widehat{S}$, such that h can be decomposed into an n-tuple $h = (h_1, \ldots, h_n)$, where $h_i : D_i \to \widehat{D}_i$ is any surjection, and $\widehat{D}_i$ is any set. The* minimal transition system of M with respect to h *is the transition system $\widehat{M}_{\min} = \langle \widehat{S}, \widehat{I}_{\min}, \widehat{R}_{\min}\rangle$ such that $\widehat{S} = \widehat{D}_1 \times \ldots \times \widehat{D}_n$, $\widehat{I}_{\min} = h(I)$, and*

$$(\widehat{s}, \widehat{s}') \in \widehat{R}_{\min} \quad \iff \quad \exists (s, s') \in S^2,\ \big(h(s) = \widehat{s}\big) \wedge \big(h(s') = \widehat{s}'\big) \wedge \big((s, s') \in R\big).$$

Note that minimal transition systems and all the notions that follow are defined with respect to a fixed abstraction h. When it is not clear from the context, we will specify the abstraction h as a superscript. When h is applied to a variable, it is understood that the corresponding h_i is applied.

For every abstraction h, define the operator $[\cdot]$ so that for any first order formula ϕ:

$$[\phi](\widehat{v}_1, \ldots, \widehat{v}_k) \stackrel{\mathrm{def}}{=} \exists v_1 \ldots \exists v_k\ \Big(\bigwedge_{i=1}^{k} \widehat{v}_i = h(v_i)\Big) \wedge \phi(v_1, \ldots, v_k).$$

Then observe that $\widehat{R}_{\min} = [R]$. In general, it is very difficult to construct $\widehat{M}_{\min}$ because the full description of the transition system M is needed in order to carry out the construction. Nevertheless, one can produce an approximation directly from its representation. Let us first define the notion of approximation.

Definition 2.4. *([9]) Let $M = \langle S, I, R\rangle$ be a transition system, and let $h : S \to \widehat{S}$ be an abstraction for M. A transition system $\widehat{M} = \langle \widehat{S}, \widehat{I}, \widehat{R}\rangle$* approximates M with respect to h *($M \sqsubseteq_h \widehat{M}$ for short) if and only if $\widehat{I}_{\min} \subseteq \widehat{I}$ and $\widehat{R}_{\min} \subseteq \widehat{R}$.*

The approximation operator, which is denoted by $\mathcal{A}$, is inductively defined on formulas that are in negation normal form by applying $[\cdot]$ only at the atomic level (including negations). For every transition system $M = \langle S, I, R\rangle$ with the representation $(\mathcal{I}, \mathcal{R})$, we denote by $\mathcal{A}(M)$ the transition system with the set of states $\widehat{S} = h(S)$ and representation $(\mathcal{A}(\mathcal{I}), \mathcal{A}(\mathcal{R}))$. Clarke, Grumberg and Long [9] show that the approximation operator $\mathcal{A}$ gives an approximation for any M, h, that is, $M \sqsubseteq_h \mathcal{A}(M)$.

Let $\widehat{M}$ be an approximation of M. Suppose that $\widehat{M} \models \Theta$. What can we conclude on the concrete model M? To answer, let us first consider the following transformations $\mathcal{C}$ and $\mathcal{D}$ between CTL* formulas on M

and their approximation on $\widehat{M}$. These transformations preserve boolean connectives, path quantifiers, and temporal operators, and act on atomic propositions as follows:

$$\mathcal{C}(\widehat{v}_i = \widehat{d}_i) \stackrel{\text{def}}{=} \bigvee_{d_i : h_i(d_i) = \widehat{d}_i} (v_i = d_i), \qquad \mathcal{D}(v_i = d_i) \stackrel{\text{def}}{=} (\widehat{v}_i = h_i(d_i)).$$

Denote by $\forall$CTL* and $\exists$CTL* the universal fragment and the existential fragment of CTL*. The following theorem gives correspondences between concrete models and their approximations.

Theorem 2.5 ([9]). *Let $M = \langle S, I, R \rangle$ be a transition system. Let $h : S \to \widehat{S}$ be an abstraction for M, and let $\widehat{M}$ be such that $M \sqsubseteq_h \widehat{M}$. Let Θ be a $\forall$CTL* formula on $\widehat{M}$, and Θ' be a $\exists$CTL* formula on M. Then*

$$\widehat{M} \models \Theta \implies M \models \mathcal{C}(\Theta) \quad \textit{and} \quad M \models \Theta' \implies \widehat{M} \models \mathcal{D}(\Theta').$$

The second implication of the theorem is only implicit in [9]. Notice that the two statements are not reciprocals of one another. In both cases, reciprocals can be shown under certain conditions [9]. The first result validates the usefulness of abstractions in practical model checking. The second will be used in our proof of Theorem 3.5.

2.4 Property testing

We consider only undirected, simple graphs (no multiple edges or self-loops). For a graph G, we denote by V_G its vertex set, by E_G its edge set, and by n the cardinality $|V_G|$ of V_G. When there is no ambiguity, we will simply write V and E instead of V_G and E_G. In the remainder of the paper we will use the following distance measure: for any two graphs G and G' on the same n-vertex set, $\mathsf{Dist}(G, G')$ is the number of edges on which the graphs disagree, divided by n^2. Our theory and our main results (Theorems 3.5 and 3.7) hold for any distance measure.

Let ϕ be a graph property and $G \models \phi$ is the classical logical notation stating that G has the property ϕ. An *ε-test* for ϕ is a probabilistic algorithm that accepts every graph with property ϕ, and rejects with probability 2/3 every graph which has distance more than ε from any graph having the property.[1] Moreover, an ε-test can only access the input graph by querying whether or not any chosen pair of vertices are adjacent. The property ϕ is called *testable* if for every $\varepsilon > 0$, there exists an ε-test for ϕ whose total number of queries depends on ε, but does not depend on the size of the graph. In several cases, the proof of testability is based on a reduction between two properties. The notion of ε-reducibility highlights this idea. This notion is central to the design of our abstractions. For every graph property ϕ and every graph G, we denote by $G \models_\varepsilon \phi$ the assertion:

$$\exists H,\ V_H = V_G,\ \mathsf{Dist}(G, H) \leq \varepsilon,\ \text{and}\ H \models \phi.$$

We say that G is ε-close to ϕ if $G \models_\varepsilon \phi$ and G is ε-far to ϕ if $G \not\models_\varepsilon \phi$ i.e. $G \models_\varepsilon \phi$ is false.

For every graph G and integer $k \geq 1$, let Π denote the set of all $\pi \subseteq V_G$ such that $|\pi| = k$, where it is understood that Π depends on both k and V_G. For convenience, we will always assume that $|V_G| \geq k$. Denote by G_π the vertex-induced subgraph of G on the vertex set $\pi \subseteq V_G$.

Definition 2.6. *Let $\varepsilon > 0$ be a real, $k \geq 1$ an integer, and ϕ, ψ two graph properties. Then ϕ is (ε, k)-*reducible to ψ *if and only if for every graph G,*

$$\begin{aligned} G \models \phi \quad &\implies \quad \forall \pi \in \Pi,\ G_\pi \models \psi, \\ G \not\models_\varepsilon \phi \quad &\implies \quad \mathop{\mathbf{Pr}}_{\pi \in \Pi} [G_\pi \models \psi] \leq 1/3. \end{aligned}$$

We say that ϕ is *ε-reducible to ψ* if there exists a constant k such that ϕ is (ε, k)-reducible to ψ.

We can recast the testability of c-colorability and bipartiteness [12, 1] in terms of ε-reducibility.

Theorem 2.7 ([1]). *For all $c \geq 3$, $\varepsilon > 0$,*

1. *c-colorability is $(\varepsilon, O((c \ln c)/\varepsilon^2))$-reducible to c-colorability;*

[1] We may also consider two-sided error, and the choice of 2/3 as the success probability is of course arbitrary.

2. bipartiteness is $(\varepsilon, O((\ln^4(\frac{1}{\varepsilon})\ln\ln(\frac{1}{\varepsilon}))/\varepsilon))$*-reducible to bipartiteness.*

Recently, Alon, Fischer, Krivelevich, and Szegedy [2] showed that all first order graph properties of type $\exists\forall$ have an ε-tester. Their results can also be recast in terms of ε-reducibility, as follows. Note, however, that in this result, the function f is a tower of towers.

Theorem 2.8 ([2]). *There exists a function* $f : \mathbb{R}_+ \to \mathbb{R}_+$*, such that every first order graph property of type* $\exists\forall$ *with* t *bound variables is* $(\varepsilon, O(f(t+1/\varepsilon)))$*-reducible to some graph property.*

3 Verification of graph properties

3.1 Context and objectives

In order to extend the framework of model checking to include the use of probabilistic abstractions, we would like to prove an analogue of Theorem 2.5 for probabilistic abstractions based on ε-reducibility. We prove that with high probability, "sufficiently" incorrect programs (in a sense to be defined below) will be rejected (*ε-robustness*). We also prove a reciprocal, which states that under a certain condition (*exactness*), correct programs will never be rejected (*soundness*).

A second goal is to extend the framework of model checking to include the verification of programs purportedly deciding graph properties or properties of finite structures. The standard model checking method is not adapted to programs on inputs that are first-order structures such as graphs. We overcome this by dealing with the specification of the program, and the property of the graph, separately. The former is handled with standard tools of model checking. The latter will reduce, as a result of the ε-reduction, to verifying a property on constant size graphs, which can be carried out in constant time.

We give an example of a very simple program for bipartiteness, together with an abstraction, and show that the approximation operator $\mathcal{A}$ results in an exact approximation of the transition system. Hence, this abstraction can be used to verify the program. One might ask whether the relaxation brought about by the use of property testing is in itself enough to beat the exponential lower bounds on the original problem. We will show in Section 4 that this is not the case, by giving a lower bound on the OBDD complexity of the relaxed version of bipartiteness.

3.2 Handling first order structures

Consider the following example of a formula we would like to verify, where P is a program which is supposed to compute some boolean function on bounded size graphs, and ϕ is a graph property:

The program P accepts only graphs which satisfy ϕ.

Suppose that $\mathtt{G}$ is an input variable of P, such that $\mathtt{G}$ is interpreted as a graph G (with respect to some fixed encoding): this will be written as $\mathtt{G} = G$. A state s of the transition system $M = \langle S, I, R\rangle$ of P is a finite sequence of variables $(\ldots, \mathtt{G}, \ldots)$. For every graph G, we then define $I_G = \{s \in I : \mathtt{G} = G\}$. Formally, what we would like to check is the following:

$$\forall G\Big(\big(\forall s \in I_G \quad M, s \models \exists\,((\neg\mathtt{ack})\,\mathbf{U}(\mathtt{ack} \wedge \mathtt{ret}))\big) \quad \Longrightarrow \quad G \models \phi\Big).$$

Note that on the right hand side of the implication, ϕ is interpreted in a structure for G which does not include the transition system. This is because the standard model checking algorithms are not suited for programs with inputs that are first order structures. When there is no ambiguity, we will write $M, G \models \Theta$ instead of $\forall s \in I_G, \quad M, s \models \Theta$.

More generally, our framework applies to the following type of formulas:

$$\forall G \quad (M, G \models \Theta \quad \Longrightarrow \quad G \models \phi), \tag{1}$$

where the input includes the graph G and may also include auxiliary data, Θ is a CTL* formula, and ϕ is a graph property.

Henceforth, we always assume a graph G to be an input variable in the program. Since G is a bounded size graph and ϕ is a formula expressing a property on G, we can determine whether $G \models \phi$ using an OBDD. Let $\mathrm{sat}(\phi, G)$ be such an OBDD. Then verifying (1) can be achieved by checking the validity of $\Big((\neg \mathcal{I}_G \vee \mathrm{check}(\mathcal{R}, \Theta)) \implies \mathrm{sat}(\phi, G)\Big)$, where $\mathcal{I}_G = \mathcal{I}(G/\mathtt{G})$ (i.e., all occurrences of the variable `G` are substituted for G).

For the graph properties that we consider, such as bipartiteness, the OBDDs for $G \models \phi$ have exponential size. As we show in Section 4, the relaxation brought about by property testing is not sufficient to reduce the OBDD size of bipartiteness. We use ε-reducibility to construct probabilistic abstractions, yielding smaller, even constant-size, OBDDs. Using such OBDDs, we are able to guarantee that P approximately decides ϕ on all its inputs.

3.3 Probabilistic abstractions

Definition 3.1. *Let M be a transition system. A* probabilistic abstraction of M *is a triple $(\mathcal{H}, \mathcal{M}, \mu)$, where $\mathcal{H}$ is a set of abstractions for M, $\mathcal{M}$ is a functional which maps every $h \in \mathcal{H}$ to a transition system $\widehat{M}^h = \mathcal{M}(h)$ such that $M \sqsubseteq_h \widehat{M}^h$, and μ is a probability distribution over $\mathcal{H}$.*

Let Θ be a CTL* formula on M, and ψ be a graph property. Then any probabilistic abstraction of M induces the following probabilistic test, where we require that $\widehat{G}^h$ be interpreted as a graph, and the operator $\mathcal{D}$ (see Section 2.3) is applied with respect to the chosen abstraction h.

Generic Test$\big((\mathcal{H}, \mathcal{M}, \mu), \Theta, \psi\big)$

1. Choose an element $h \in \mathcal{H}$ according to μ.
2. Accept if (and only if)
   $\forall \widehat{G}^h \quad (\widehat{M}^h, \widehat{G}^h \models \mathcal{D}(\Theta) \quad \implies \quad \widehat{G}^h \models \psi)$.

The probability that the test rejects will be denoted by $\mathsf{Rej}\big((\mathcal{H}, \mathcal{M}, \mu), \Theta, \psi\big)$. The distribution μ will be omitted when it denotes the uniform probability distribution. To be useful in practice, a probabilistic abstraction should be both ε-robust (programs are rejected with probability $2/3$ if the relaxed specification is false for some input) and sound (no correct programs are rejected), in which case we say that it is an ε-abstraction. When this is the case, checking the correctness of a program can be easily done on the abstracted model with high confidence using **Generic Test**. Fix a confidence parameter $0<\gamma<1$, and iterate **Generic Test** $O(\ln 1/\gamma)$ times. If the program is correct, **Generic Test** always accepts; and if there is an instance on which the program is not correct with respect to the relaxed specification, **Generic Test** rejects at least once with probability at least $(1-\gamma)$.

Definition 3.2. *Let M be a transition system, $\varepsilon > 0$, Θ be a* CTL* *formula, and let ϕ, ψ be two graph properties. A probabilistic abstraction $(\mathcal{H}, \mathcal{M}, \mu)$ of M is* ε-robust with respect to (Θ, ϕ, ψ) *if*

$$\big(\exists G \quad (M, G \models \Theta \text{ and } G \not\models_\varepsilon \phi)\big) \qquad \implies \qquad \mathsf{Rej}\big((\mathcal{H}, \mathcal{M}, \mu), \Theta, \psi\big) \geq \frac{2}{3}.$$

Definition 3.3. *Let M be a transition system, Θ be a* CTL* *formula, and let ϕ, ψ be two graph properties. A probabilistic abstraction $(\mathcal{H}, \mathcal{M}, \mu)$ of M is* sound with respect to (Θ, ϕ, ψ) *if*

$$\big(\forall G \quad (M, G \models \Theta \implies G \models \phi)\big) \qquad \implies \qquad \mathsf{Rej}\big((\mathcal{H}, \mathcal{M}, \mu), \Theta, \psi\big) = 0.$$

Definition 3.4. *Let M be a transition system, $\varepsilon > 0$, Θ be a* CTL* *formula, and let ϕ, ψ be two graph properties. A probabilistic abstraction $(\mathcal{H}, \mathcal{M}, \mu)$ of M is an* ε-abstraction for (Θ, ϕ, ψ) *if it is both ε-robust and sound with respect to (Θ, ϕ, ψ).*

3.4 Constructing ε-abstractions

We now explain how to construct ε-abstractions based on ε-reducibility. Fix $\varepsilon > 0$, and assume that ϕ is (ε, k)-reducible to ψ, for some $k \geq 1$. We give a generic proof of robustness of our probabilistic abstraction, and we isolate a sufficient condition which implies soundness. Under this condition, we obtain an ε-abstraction. From Definition 2.4, for any fixed k and any fixed vertex set V, we let Π be the set of all subsets π of V with $|\pi| = k$, and for any graph G with $V_G = V$, the vertex-induced subgraph on the vertex set π is denoted by G_π.

Since we relax ϕ with respect to ε, we can decompose our initial specification (1) into the following family of reduced specifications:

$$\{\forall G \quad (M, G \models \Theta \quad \Longrightarrow \quad G_\pi \models \psi) : \pi \in \Pi\}.$$

For every π, the corresponding reduced specification can now be subject to an abstraction h_π. Every corresponding abstracted variable v and constant d will be denoted respectively by $\widehat{v}^\pi$ and $\widehat{d}^\pi$. We require that the abstraction of G be exactly G_π, that is, $\widehat{G}^\pi = G_\pi$. Let $\widehat{M}^\pi$ be such that $M \sqsubseteq_{h_\pi} \widehat{M}^\pi$. We define the (uniform) probabilistic abstraction $(\mathcal{H}, \mathcal{M})$ (also denoted by $(\Pi, \mathcal{M})$) as $\mathcal{H} = \{h_\pi : \pi \in \Pi\}$ and $\mathcal{M}(h_\pi) = \widehat{M}^\pi$, for every $\pi \in \Pi$. This leads to the following test, derived from **Generic Test** for this family of abstractions:

Graph Test$\big((\Pi, \mathcal{M}), \Theta, \psi\big)$

1. Randomly choose a subset of vertices $\pi \in \Pi$.
2. Accept if (and only if)
   $\forall \widehat{G}^\pi \quad (\widehat{M}^\pi, \widehat{G}^\pi \models \mathcal{D}(\Theta) \quad \Longrightarrow \quad \widehat{G}^\pi \models \psi)$.

We show that if Θ is an $\exists$CTL* formula and ϕ is ε-reducible to ψ, then our probabilistic abstraction is ε-robust with respect to (Θ, ϕ, ψ). This, together with its conditional reciprocal in Theorem 3.7, establishes the validity of the method.

Theorem 3.5. *Let Θ be a $\exists$CTL* formula. Let $\varepsilon > 0$ be a real, $k \geq 1$ an integer, and let ϕ be (ε, k)-reducible to ψ. Let $(\Pi, \mathcal{M})$ be a probabilistic abstraction such that $\widehat{G}^\pi = G_\pi$, for every $\pi \in \Pi$. Then $(\Pi, \mathcal{M})$ is ε-robust with respect to (Θ, ϕ, ψ).*

Proof. Let G be such that $M, G \models \Theta$ and $G \not\models \phi_\varepsilon$. By Theorem 2.5, $\widehat{M}^\pi, \widehat{G}^\pi \models \mathcal{D}(\Theta)$, for every $\pi \in \Pi$. Moreover, by definition of (ε, k)-reducibility we know that $\mathbf{Pr}_{\pi \in \Pi}\left[\widehat{G}^\pi \models \psi\right] \leq 1/3$. Therefore,

$$\mathop{\mathbf{Pr}}_{\pi \in \Pi}\left[\widehat{M}^\pi, \widehat{G}^\pi \models \mathcal{D}(\Theta) \quad \Longrightarrow \quad \widehat{G}^\pi \models \psi\right] \leq \frac{1}{3}.$$

We conclude by observing that the acceptance probability of **Graph Test** is bounded above by the term on the left hand side of the inequality. □

Having shown that the abstraction is ε-robust, we give a sufficient condition for soundness: exactness.

Definition 3.6. *Let M be a transition system, Θ be a* CTL* *formula, h be an abstraction, and let $\widehat{M}$ be such that $M \sqsubseteq_h \widehat{M}$. Then the approximation $\widehat{M}$ is* exact with respect to Θ *if and only if for every graph $\widehat{G}$:*

$$\widehat{M}, \widehat{G} \models \mathcal{D}(\Theta) \qquad \Longrightarrow \qquad \exists H, \quad \widehat{H} = \widehat{G} \text{ and } M, H \models \Theta.$$

The *exactness* property is a reciprocal to the theorem 2.5 stated for $\exists$CTL* formulas in our context.

Theorem 3.7. *Let Θ be a $\exists$CTL* formula. Let $\varepsilon > 0$ be a real, $k \geq 1$ an integer, and let ϕ be (ε, k)-reducible to ψ. Let $(\Pi, \mathcal{M})$ be a probabilistic abstraction such that $\widehat{G}^\pi = G_\pi$ and $\widehat{M}^\pi$ is an exact approximation with respect to Θ, for every $\pi \in \Pi$. Then $(\Pi, \mathcal{M})$ is sound with respect to (Θ, ϕ, ψ).*

Proof. Fix $\pi \in \Pi$. Let $\widehat{G}^\pi$ be a k-vertex graph such that $\widehat{M}^\pi, \widehat{G}^\pi \models \mathcal{D}(\Theta)$. From the exactness of $\widehat{M}^\pi$, there exists a graph H such that $\widehat{H}^\pi = \widehat{G}^\pi$ and $M, H \models \Theta$. Therefore, from the hypotheses we get $H \models \phi$. The (ε, k)-reducibility of ϕ to ψ implies that $H_\pi \models \psi$, that is, $\widehat{G}^\pi \models \psi$. Thus, for all $\pi \in \Pi$ and $\widehat{G}^\pi$:

$$\widehat{M}^\pi, \widehat{G}^\pi \models \mathcal{D}(\Theta) \quad \Longrightarrow \quad \widehat{G}^\pi \models \psi.$$

□

3.5 An ε-abstraction for bipartiteness

In this section, we give a short program for bipartiteness, and an ε-abstraction for this program. We consider a function which, given a graph G and a coloring *Color* (entered by the user), decides if *Color* is a bipartition for G. The graph G is represented in the program in the natural way by the upper triangular entries of a boolean matrix variable `G` and *Color* by a boolean array variable `Color`.

```
FUNCTION CHECK-PARTITION
  CONSTANT INTEGER n=10000
  INPUT G : ARRAY[n,n] of BOOLEAN
  VAR Color : ARRAY[n] of BOOLEAN
  VAR u,v : INTEGER 1..n+1
1: get(Color)
2: u=2
3: WHILE u<=n DO {
     v=1
4:   WHILE v<=u-1 DO {
5:    IF G[u,v]&&(Color[u]=Color[v]) RETURN false
      v=v+1          }
     u=u+1       }
6: RETURN true
```

3.5.1 Correctness property

We want to verify that, for every input `G`, if there exists an input value for `Color` for which the program accepts, then `G` represents a bipartite graph. More formally, we want to verify the following property:

$$\forall G \quad \Big(M, G \models \exists\big((\neg \mathtt{ack})\,\mathbf{U}(\mathtt{ack} \wedge \mathtt{ret})\big) \implies G \text{ is bipartite}\Big). \tag{2}$$

Note that $\exists$ ranges over all the possible initial values of `Color` which the user can enter with the instruction `get(Color)`. For each π we define the abstraction which maps G to the subgraph G_π, *Color* to the coloring on the subset of vertices induced by π, and $\mathtt{u}, \mathtt{v}$ refer to the vertices as follows: $\widehat{u}^\pi$ is u if $u \in \pi$, and is $\min\{w : \ \forall t\ (w \leq t \leq u \implies t \notin \pi)\}$ otherwise.

For this abstraction, we want to show robustness and soundness (Corollary 3.9). Robustness is directly obtained from Theorem 3.5, and using Theorem 3.7, we only need to prove the exactness of the abstraction.

Lemma 3.8. *For every $\pi \in \Pi$, $\mathcal{A}^\pi(M)$ is exact with respect to the $\exists$CTL* formula of Equation (2).*

Proof. For the sake of clarity, the proof is in two parts. First, we prove that the abstraction which only maps $G \mapsto G_\pi$, and preserves the other variables, is exact for every π. Then we show how it can be extended to the complete abstraction.

First, let us represent the transition system of `CHECK-PARTITION`. For convenience, we use a compact representation of relational expressions representing the transition relation. Each line corresponds to a transition between two control points. The transition relation is represented by the disjunction of these lines. We use '$i \mapsto j$:' as an abbreviation for $(\mathtt{PC} = i) \wedge (\mathtt{PC}' = j)$. On any given line, for any pair $(\mathtt{v}, \mathtt{v}')$ of program variables, if $\mathtt{v}'$ does not occur in the relational expression, then the atomic proposition $(\mathtt{v}' = \mathtt{v})$ is understood, but omitted from the compact form. Furthermore, the expression $(\mathtt{v}' = *)$ is used when the value of $\mathtt{v}'$ is unspecified, and it is the abbreviation for the formula stating that all other variables do not change. This typically occurs after a `get` instruction, and corresponds to a nondeterministic transition.

The initial states of the transition system of `CHECK-PARTITION` are $(\texttt{ack}{=}\mathit{false}) \wedge (\texttt{PC}{=}1)$. The relational expression of the transition system of `CHECK-PARTITION` is given in compact form by the disjunction of the following boolean formulas.

$$
\begin{array}{l}
1 \mapsto 2 : (\texttt{Color'}{=}*) \\
2 \mapsto 3 : (\texttt{u'}{=}2) \\
3 \mapsto 4 : (\texttt{u} \leq \texttt{n}) \wedge (\texttt{v'}{=}1) \\
3 \mapsto 6 : (\texttt{u}{=}\texttt{n}+1) \\
4 \mapsto 3 : (\texttt{v}{=}\texttt{u}) \wedge (\texttt{u'}{=}\texttt{u}+1) \\
4 \mapsto 5 : (\texttt{v} \leq \texttt{u}-1) \\
5 \mapsto 5 : \texttt{G[u,v]} \wedge (\texttt{Color[u]}{=}\texttt{Color[v]}) \wedge (\texttt{ack'}{=}\mathit{true}) \wedge (\texttt{ret'}{=}\mathit{false}) \\
5 \mapsto 4 : ((\neg \texttt{G[u,v]}) \vee (\texttt{Color[u]}{\neq}\texttt{Color[v]})) \wedge (\texttt{v'}{=}\texttt{v}+1) \\
6 \mapsto 6 : (\texttt{ack'}{=}\mathit{true}) \wedge (\texttt{ret'}{=}\mathit{true})
\end{array}
\tag{3}
$$

We first suppose that only `G` is abstracted. Let $\texttt{G}_\pi$ denote the corresponding abstraction of the variable `G`. We will show that there is a graph G_0 such that if there is an accepting path in the abstracted transition system when the graph input is set to G, then there is an accepting path in the concrete system when the graph input is set to G_0. Indeed, G_0 will be the restriction of $G = (V, E)$ to vertices in π, that is $G_0 = (V, E_0)$ is the n-vertex graph whose vertices are defined by

$$(u, v) \in E_0 \quad \Longleftrightarrow \quad (u, v) \in E \text{ and } u, v \in \pi.$$

The operator $\mathcal{A}$ transforms only the relational expression part of transitions $5 \mapsto 5$ and $5 \mapsto 4$ into:

$$
\begin{array}{l}
5 \overset{\pi}{\mapsto} 5 : \big(\exists H \ (H_\pi{=}\texttt{G}_\pi) \wedge H[\texttt{u}, \texttt{v}]\big) \wedge (\texttt{ack'}{=}\mathit{true}) \wedge (\texttt{Color[u]}{=}\texttt{Color[v]}) \wedge (\texttt{ret'}{=}\mathit{false}) \\
5 \overset{\pi}{\mapsto} 4 : \big(\exists H \ (H_\pi{=}\texttt{G}_\pi) \wedge ((\neg H[\texttt{u}, \texttt{v}]) \vee (\texttt{Color[u]}{\neq}\texttt{Color[v]}))\big) \wedge (\texttt{v'}{=}\texttt{v}+1)
\end{array}
\tag{4}
$$

For instance in the first line, the expression $\exists H \ (H_\pi{=}\texttt{G}_\pi) \wedge H[\texttt{u}, \texttt{v}]$ is the result of the approximation operator $\mathcal{A}$ on `G[u,v]`.

The following fact will enable us to conclude.

$$(4) \quad \Longrightarrow \quad (3)(G_0/\texttt{G}). \tag{5}$$

To prove this, observe that whenever the left hand side term of (5) is true for some H, then it is still true for any subgraph K of H such that $K_\pi = G_\pi$, and therefore for G_0.

With (5), we can conclude that there exists an accepting path in the concrete system with $\texttt{G} = G_0$. To an accepting path σ in the abstract system corresponds a unique coloring `Color`. Let τ be the path of the concrete system starting from the initial state, defined by $\texttt{G} = G_0$ and `Color`. The abstracted path $\widehat{\tau}^\pi$ of τ is in fact σ. To prove that τ is an accepting path, observe that it is enough to prove that whenever $\texttt{PC} = 5$, then the transition $5 \mapsto 4$ is done. Since the transition system is deterministic whenever $\texttt{PC} \neq 1$, this is equivalent to Formula (3) being true. By contradiction, assume that $\texttt{PC} = 5$ and (3) is false, then from (5) we get that the abstracted path is rejecting. This concludes the proof since we assumed that it is accepting.

In the general case, `G`, `Color`, `u` and `v` are abstracted. We will show that there is a graph G_0 and a coloring $Color_0$ such that if there is an accepting path in the abstracted transition system when the graph input is set to G and the coloring input to $Color$, then there is an accepting path in the concrete system when the graph input is set to G_0 and the coloring input to $Color_0$.

The operator $\mathcal{A}$ transforms the transition $5 \mapsto 4$ into:

$$
\begin{array}{l}
5 \overset{\pi}{\mapsto} 4 : \big(\exists H \ \exists COLOR \ \exists r \ \exists s \ \exists t \ (H_\pi{=}\texttt{G}_\pi \wedge \widehat{COLOR}^\pi{=}\widehat{\texttt{Color}}^\pi \wedge \widehat{r}^\pi{=}\widehat{\texttt{u}}^\pi \wedge \widehat{s}^\pi{=}\widehat{\texttt{v}}^\pi \wedge \widehat{t}^\pi{=}\widehat{s+1}^\pi) \\
\qquad\qquad \wedge ((\neg H[r, s]) \vee (COLOR[r]{\neq}COLOR[s])) \wedge (s'{=}t)\big)
\end{array}
\tag{6}
$$

We will first prove an analogue of (5), that is

$$(6) \quad \Longrightarrow \quad (3)(G_0/\texttt{G})(Color_0/\texttt{Color})(u_0/\texttt{u})(v_0/\texttt{v}), \tag{7}$$

where G_0 is defined as above, $Color_0$ is any coloring that coincides with the restriction of the coloring to vertices in π and u_0, v_0 are such $\widehat{\mathtt{u}}^\pi = \widehat{u_0}^\pi$ and $\widehat{\mathtt{v}}^\pi = \widehat{v_0}^\pi$. Since there is an accepting path in the abstracted transition system, there always exists a graph H such that $H_\pi = G_\pi$, a coloring $COLOR$ such that $\widehat{COLOR}^\pi = \widehat{Color}^\pi$, and r and s such that $\widehat{r}^\pi = \widehat{\mathtt{u}}^\pi$ and $\widehat{s}^\pi = \widehat{\mathtt{v}}^\pi$, so the system makes the transition $5 \overset{\pi}{\mapsto} 4$. Observe that the transition is also made when H is set to G_0. Notice also that only the indices of `Color` that appear in π are relevant for this transition. Moreover, the only relevant values of the vertex counters are those corresponding to vertices in π (these values are unchanged by the abstraction). Therefore (7) is true.

As in the first case, to an accepting path σ in the abstract system corresponds a coloring $\widehat{\mathtt{Color}}^\pi$. Define $Color_0$ as above. Let τ be the path of the concrete system starting from the initial state, defined by $\mathtt{G} = G_0$ and $\mathtt{Color} = Color_0$. The abstracted path $\widehat{\tau}^\pi$ of τ, where consecutive duplicates are removed, is in fact σ. Again, we can prove that τ is an accepting path using (7). □

Since the size of π is fixed, our abstraction induces a constant size OBDD. By Lemma 3.8, Theorems 2.7, 3.5, and 3.7, we know that our probabilistic abstraction is an ε-abstraction, so **Graph Test** can be used for checking the validity of (2).

Corollary 3.9. *Let $\varepsilon > 0$. Using the previous probabilistic abstraction,* **Graph Test** *on* `CHECK-PARTITION` *satisfies the following:*

1. *If* `CHECK-PARTITION` *satisfies specification* (2)*, then* **Graph Test** *always accepts;*
2. *If there exists a graph G which has distance more than ε from any bipartite graph, but which is accepted by* `CHECK-PARTITION` *for some coloring Color, then* **Graph Test** *rejects with probability at least* $2/3$*;*
3. *The time complexity of* **Graph Test** *is exponential in* $\mathrm{poly}(1/\varepsilon)$ *and does not depend on n, the input size.*

3.5.2 Other temporal properties

In the previous example, we used for Θ the basic *correctness* property, i.e. $\exists\big((\neg\mathtt{ack})\,\mathbf{U}\,(\mathtt{ack} \wedge \mathtt{ret})\big)$. We could use other temporal properties such as:

- $\exists\big((v < n)\ \mathbf{U}\ (v = n)\big)$
- $\exists\big(\mathbf{F}(v = n)\big)$
- $\exists\big(\mathbf{F}(u = n + 1)\big)$

All these temporal formulas are equivalent to the program's termination property, hence the previous abstraction satisfies both the robustness and soundness properties of Corollary 3.9.

On the other hand, a property such as

- $\exists\big(\mathbf{F}u \geq n\big)$

is not equivalent to the termination property as we may miss the last iteration for $u = n$ and $v = 1, \ldots n-1$. We may terminate without checking the corresponding last $O(n)$ edges of the graph. There are non bipartite graphs which are ε-bipartite that will pass the **Graph Test** for any ε. The method is not guaranteed to detect an error, because the distance to bipartiteness, $O(n)$ is too small, always less than εn^2.

Finally a property such as

- $\exists\big(\mathbf{F}u \geq \frac{9}{10} \cdot n\big)$

is not verified as we may miss the last iterations for $u = \lfloor \frac{9}{10} \cdot n \rfloor, \lfloor \frac{9}{10} \cdot n \rfloor + 1, \ldots n$. However, it will be detected with $\varepsilon = \frac{1}{10}$ because of the robustness property. The soundness property is not useful since we only want to find errors.

3.6 General application

The proposed method generalizes model checking to programs on first-order structures U of a class $\mathbf{K}$ such as finite graphs. We clearly distinguish the temporal property Θ applied to the transition system with the *correctness property* ϕ on the class of finite inputs U and we verify that:

$$\forall U \Big((\forall s \in I_U \quad M, s \models \Theta \quad \Longrightarrow \quad U \models \phi \Big)$$

If we have a tester for the property ϕ, i.e. if ϕ is (ε, k) reducible to some ψ, we can then look for an ε-abstraction for various Θs. We would then need to formally verify the relaxation of the property, in time depending only on ε.

4 Lower bound for approximate bipartiteness

In the previous sections we have shown that probabilistic abstractions can be used to reduce the transition system size, for the problem of ε-bipartiteness. Prior to this work, it was known that bipartiteness does not admit small OBDDs [15]; however, it was not known whether a small OBDD could be constructed for the relaxed version of the problem. Had this been possible, then standard model checking techniques would have sufficed for the verification of ε-bipartiteness. In this section we show that this is not the case.

4.1 Overview

The proof that any OBDD for bipartiteness has exponential size is based on communication complexity. In a communication complexity game, each player is given an input, say, player I is given x and player II is given y, and their goal is to compute some function $f(x, y)$. To compute f, the players follow a protocol, in which each player alternately sends a message to the other player, and at the end of the protocol, one of the players outputs the value of $f(x, y)$. The complexity of a function is the total size of the messages exchanged for the worst-case input of the best protocol. There is no limit on the computational power of the two players. When there is just one message from Player I to Player II, we use the term one-way communication complexity.

Giving a lower bound on the communication complexity suffices to give a lower bound on OBDD size. Assume we are given an minimal size OBDD for a function f, whose width is w, and which examines the variables in the order $x_1, \ldots, x_n$. If player I is given the first half of the variables, and the second player is given the second half, then there is a communication protocol that computes f using $\log(w)$ bits of communication: player I sends player II the state of the OBDD after the first half of the variables are scanned. Player II can then complete the evaluation of the OBDD from this state and output the value of f. By this argument, any lower bound on the one-way communication complexity of the function implies an exponentially-scaled lower bound on OBDD width, provided the lower bound still holds regardless of the way the variables are shared among Player I and Player II.

In the remainder of this section, we show how the communication complexity lower bound of Hajnal, Maass, and Turán [15] can be adapted to yield a lower bound on OBDD size for the relaxed version of bipartiteness. This establishes that in the case of bipartiteness, reducing the size of the OBDD cannot be achieved solely by relaxing the exactness of the result. The argument presented in the conference version of this paper was only partially correct for small $\varepsilon = O(1/n^2)$. We present a new graph construction which generalizes and simplifies the argument based on the graphs of [15].

The ε-bipartiteness problem is the following partial boolean function.

Definition 4.1 (ε-bipartiteness). *Let $\varepsilon > 0$ be a real. The* ε-bipartiteness problem in V *is a partial function f on graphs G on V:*

$$f(G) = \begin{cases} 1 & \text{if } G \text{ is bipartite,} \\ 0 & \text{if } G \text{ has distance more than } \varepsilon \text{ from any bipartite graph,} \\ \perp \text{ (undefined)} & \text{otherwise.} \end{cases}$$

We say that an OBDD *solves* a partial function f if its output agrees with f whenever f is defined. The rest of this section is devoted to proving the following theorem.

Theorem 4.2. *For every sufficiently small $\varepsilon > 0$, any OBDD that solves the ε-bipartiteness problem for graphs with n vertices has width $2^{\Omega(n)}$.*

The rest of the paper is devoted to the proof of this theorem. It follows directly from Propositions 4.3 and 4.5, and Lemmas 4.8 and 4.12 (see Section 4.2.3 for the outline of the proof).

4.2 Preliminaries and notation

4.2.1 Communication complexity

To establish the lower bound on OBDD size, we will use techniques from communication complexity. The disjoint union of sets A and B will be written $A \dot\cup B$. In a communication complexity protocol, two players each have part of an input string, and they wish to compute a function on this joined input. Let $f : \{0,1\}^N \to \{0,1\}$ be such a boolean function which two players wish to compute. Usually this function is total, that is, defined on every input. However in this paper we consider functions that may be partial, *i.e.* undefined (denoted $\perp$) on some of the inputs. Each player gets a part of the input according to some partition of the input bits as follows. Let $R \dot\cup Y$ be a partition of the N input variables into two equal parts. Player 1's input x corresponds to the variables of R, Player 2's input y corresponds to the variables of Y. When a partition of the input bits is fixed, we may write f as a 2-argument function $f^{R:Y}(x,y)$, where it is understood that the first argument is Player 1's input and the second argument is Player 2's input. Often when there is no ambiguity, the superscript $R:Y$ is dropped.

Note that for the bipartiteness problem, the inputs are graphs, so an input variable corresponds to a pair of vertices of the graph. This is why we will consider partition R, Y of the complete graph on V, for some set of vertices V. If an edge variable is set to 1, this means the edge is in the graph, and if it is set to 0, then the edge is not in the graph.

In a *one-way communication protocol* for f, Player 1 sends one message to Player 2, who outputs the value of $f^{R:Y}(x,y)$. The *communication* $\kappa_{\rightarrow}^{R:Y}(\mathcal{P}; x, y)$ incurred by a one way-communication protocol $\mathcal{P}$ on input x, y for the partition R, Y is the number of bits of the message sent by Player I. The *one-way communication complexity* of f for a partition R, Y is denoted $\kappa_{\rightarrow}^{R:Y}(f)$, and is the minimum, over all one-way communication protocols $\mathcal{P}$ for f, of $\max\{\kappa_{\rightarrow}^{R:Y}(\mathcal{P}; x, y)\}$, where the maximum is over all $x, y : |x| = |y| = N$ such that $f^{R:Y}(x,y) \neq \perp$. The *one-way communication complexity for the best-case partition of variables* is $\kappa_{\rightarrow}^{best}(f) = \min_{R \dot\cup Y}\{\kappa_{\rightarrow}^{R:Y}(f)\}$, where $R : Y$ ranges over all partitions with $|R| = |Y| \pm 1$.

Let f be a (possibly partial) boolean function whose variables are partitioned according to R, Y. In the usual setting, where f is a total function, the *communication matrix associated* of f is the matrix representation $M_f^{R:Y}$ of f, that is $(M_f^{R:Y})_{x,y} = f^{R:Y}(x,y)$. For partial functions, we let $(M_f^{R:Y})_{x,y} = \star$ whenever $f^{R:Y}(x,y) = \perp$. When R, Y is clear from the context, we drop the superscript $R : Y$.

The lower bound on the width of OBDDs that compute f is related to the communication matrix of f by the following proposition. We state the result for one-way communication complexity, which can be easily derived from [17], and which remains true for partial functions using the following notion. We say that two lines are *unambiguously distinct* if on some column, one line contains a 0 and the other contains a 1.

Proposition 4.3 (follows from [17, Page 144]).

1. *If there exists an OBDD of width at most w that solves f, then $\kappa_{\rightarrow}^{best}(f) \leq \log w$.*
2. *Let M_f be the communication matrix of f whose variables are partitioned according to R, Y. Then $\kappa_{\rightarrow}^{R:Y}(f) \geq \log(l)$, where l is the number of pairwise unambiguously distinct lines in M_f.*

4.2.2 Reductions in communication complexity

Informally, a communication problem reduces to another if the communication matrix of the first problem appears as a submatrix of the second problem's communication matrix. We give a formal definition below. In Section 4.4.2, we will show that the half-set disjointness problem, defined in Section 4.3, reduces to the ε-bipartiteness problem.

Definition 4.4. *Let $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ and $g : \{0,1\}^m \times \{0,1\}^m \to \{0,1\}$ be two boolean functions. Then g reduces to f (written $g \leq f$) if there exists two functions $h_x, h_y : \{0,1\}^m \to \{0,1\}^n$ such that*

$$g(x,y) \neq \perp \implies g(x,y) = f(h_x(x), h_y(y)).$$

As for standard reductions, the following proposition holds for one-way communication complexity.

Proposition 4.5. *Let $f : \{0,1\}^n \times \{0,1\}^n \to \{0,1\}$ and $g : \{0,1\}^m \times \{0,1\}^m \to \{0,1\}$ be two boolean functions. If $g \leq f$ then $\kappa_{\to}(g) \leq \kappa_{\to}(f)$.*

4.2.3 Outline of the proof

We now give the outline of the proof of Theorem 4.2. We find $l = 2^{\Omega(n)}$ such that

$$\log(l) \leq \kappa_{\to}(\text{half-set disjointness})$$

by giving a lower bound on the number of distinct lines in the communication matrix of the half-set disjointness problem (Lemma 4.8). When $\varepsilon > 0$ is sufficiently small, for any R, Y, the inequality

$$\kappa_{\to}(\text{half-set disjointness}) \leq \kappa_{\to}^{R:Y}(\varepsilon\text{-bipartiteness})$$

holds by virtue of the reduction half-set disjointness $\leq$ ε-bipartiteness$^{R:Y}$, for all coloring R, Y, (Lemma 4.12). By definition of $\kappa_{\to}^{best}$, this implies that

$$\log(l) \leq \kappa_{\to}^{best}(\varepsilon\text{-bipartiteness}).$$

By Proposition 4.3, if an OBDD for ε-bipartiteness has width w, then

$$\kappa_{\to}^{best}(\varepsilon\text{-bipartiteness}) \leq \log(w).$$

This allows us to conclude that $w \geq l = 2^{\Omega(n)}$.

4.3 Half-set disjointness

We introduce the half-set disjointness problem, an auxiliary problem which we will use to derive our lower bound. In Section 4.4 we will show that it reduces to the ε-bipartiteness communication matrix, for sufficiently small ε. Then, it will suffice to show a lower bound for the one-way communication complexity of the half-set disjointness problem.

Definition 4.6 (half-set disjointness problem). *Let S be a finite set. The* half-set disjointness problem *in S is a partial function g on subsets $T_1, T_2 \subset S$ of size $\lfloor |S|/4 \rfloor$:*

$$g(T_1, T_2) = \begin{cases} 1 & \text{if } T_1 \cap T_2 = \emptyset, \\ 0 & \text{if } |T_1 \cap T_2| \geq |T_1|/2, \\ \perp & \text{otherwise.} \end{cases}$$

For the half-set disjointness problem, we only consider one partition of the input variables: the input of Player 1 is (an encoding of) some subset T_1, and the input of Player 2 is (an encoding of) some subset T_2. The corresponding communication matrix is M, whose rows and columns are labeled by subsets $T_1, T_2 \subset S$ of size $|S|/4$.. The number of rows and columns is $\binom{|S|}{\lfloor |S|/4 \rfloor}$. We show that M has an exponential number of unambiguously distinct lines. Before we state a usefull technical result.

Proposition 4.7. *Let S be a finite set. There exist at least $\lfloor 2^{|S|/64} \rfloor$ distinct subsets $T_i \subset S$ of size $\lfloor |S|/4 \rfloor$ such that every pair $T_i \neq T_j$ satisfies $|T_i \cap T_j| \leq |T_i|/2$.*

Proof. We use a probabilistic argument to obtain a large family of sets that have the required property. Let $m = \lfloor |S|/4 \rfloor$. Assume that $|S| \geq 64$, *i.e.* $m \geq 16$, otherwise the proposition is trivial. First we bound above the probability $\mathbf{Pr}\left[|T_1 \cap T_2| > m/2\right]$ when $T_1, T_2 \subset S$ are chosen uniformly at random with $|T_1| = |T_2| = m$. Fix any $T_1 \subset S$ of size m. Then $\mathbf{Pr}_{x \in S}\left[x \in T_1\right] = |T_1|/|S| \leq 1/4$. When $T_2 \subset S$ is chosen uniformly at random among sets of size m, the size of $T_1 \cap T_2$ is a hypergeometric random variable on m trials. Therefore, we can apply a Chernoff-Hoeffding bound [11, Theorem 2.10, page 29][16] to obtain the following inequality

$$\mathop{\mathbf{Pr}}_{T_2 \subset S, |T_2| = m}\left[|T_1 \cap T_2| > m/2\right] \leq e^{-m/8}.$$

Since the resulting upper bound is established for any $T_1 \subset S$ of size m, it is still valid for a random T_1. Therefore,

$$\mathop{\mathbf{Pr}}_{T_1, T_2 \subset S, |T_1| = |T_2| = m}\left[|T_1 \cap T_2| > m/2\right] \leq e^{-m/8}.$$

Let $N \geq 2$ be any integer. By using the union bound we get that

$$\mathop{\mathbf{Pr}}_{T_i \subset S, |T_i| = m, i = 1, \ldots, N}\left[\exists i \neq j : |T_i \cap T_j| > m/2\right] \leq \binom{N}{2} e^{-m/8}.$$

Therefore if the right hand side term is less than 1, there necessarily exists a family $T_1, \ldots, T_N$ of subsets that have requiered property. To conclude, observe that if $N \leq 2^{|S|/64}$ this condition is indeed satisfied. □

Lemma 4.8. *Let S be a finite set. The communication matrix of the half-set disjointness problem in S has at least $\lfloor 2^{|S|/64} \rfloor$ unambiguously distinct lines.*

Proof. Let $N = \lfloor 2^{|S|/64} \rfloor$, $m = \lfloor |S|/4 \rfloor$, and $T_1, \ldots, T_N \subset S$ be the family of sets from Proposition 4.7. Fix $i \neq j$. Since $|T_i \cap T_j| \leq m/2$, there exists a subset $T \subset S$ of size m such that $T_i \cap T = \emptyset$ and $|T_j \cap T| \geq |T|/2$. By construction, the lines T_i and T_j of the communication matrix of the half-set disjointness problem in S, that is g, are unambiguously distinct because $g(T_i, T) = 1$ and $g(T_j, T) = 0$. □

4.4 Reduction to ε-bipartiteness

This section is devoted to reducing half-set disjointness to ε-bipartiteness, for any sufficiently small $\varepsilon > 0$.

Since the goal is to obtain a lower bound on the one-way communication complexity for the best-case partition of variables ($\kappa_{\rightarrow}^{best}$), we must prove the reduction for any partition of the input variables R, Y. For the ε-bipartiteness problem, the variables in the communication problem are pairs of vertices: the variable is 1 whenever the corresponding edge is in the graph, and 0 otherwise. Each player is given half of the edge variables according to some partition R, Y of the complete graph. The input of the communication protocol is the graph formed by the union of edges given to each player.

Hajnal, Maass and Turán [15] give a reduction from a property on partitions to connectivity and exact bipartiteness. They prove an $\Omega(n \log n)$ lower bound on the communication complexity of the property on partitions. We give a reduction from half-set disjointness to ε-bipartiteness, using some of their techniques. Our goal is to construct a class of graphs with many triangles, so that it is necessary to remove $\varepsilon \cdot n^2$ edges for it to become bipartite.

4.4.1 Multi-triangle graphs

When we refer to bipartite graphs H, we will use the notation $H = (A, B, E)$, where $A \dot{\cup} B$ denotes a partition of the vertex set of H for which the set of edges E satisfies $E \subseteq A \times B$.

Definition 4.9. *Let $H = (A, B, E)$ be a bipartite graph with $|A| = |B| = r$. Then H is* dense *if every vertex has degree at least $r/3$.*

If $G = (V, E)$ is a graph and $A \dot{\cup} B \subseteq V$, then $G|_{(A,B)}$ denotes the bipartite graph induced by G on $A \times B$. Using Szemerédi's regularity lemma, [15] prove a stronger version of the following. To get this result apply [15, Lemma 9] with parameters $0 < \varepsilon < 1/12$ and $\gamma = 1/2$.

Lemma 4.10. *There exists $0 < \alpha < 1$ and n_0 a positive integer such that for every partition R, Y of the edges of the complete graph $K = (V, R \dot\cup Y)$ on $n \geq n_0$ vertices V, the following holds: there exist subsets $V_0^1, V_0^2, V_1, V_2 \subseteq V$ each of size $r \geq \alpha n$ such that*

1. *V_0^1 and V_1, V_0^2 and V_2, V_1 and V_2 are respectively disjoint,*
2. *$r/3 \leq |V_0^1 \cap V_0^2| \leq r$,*
3. *$(V, R)|_{(V_0^1, V_1)}$ and $(V, Y)|_{(V_0^2, V_2)}$ are dense.*

In the remainder of the proof, we will set $A = V_1$, $B = V_0^1 \cap V_0^2$ and $C = V_2$. Notice that if T_1 and T_2 are two subsets of B, then there are many triangles connecting vertices of $T_1 \cap T_2$, vertices of A, and vertices C. We relate the size of the intersection of T_1 and T_2 to the number of triangles, and finally on the number of edges that must be removed for the graph to become bipartite. The construction of the following lemma is illustrated by Figure 1.

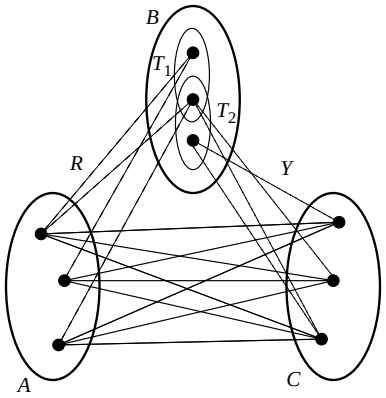

Figure 1: An instance of G^{T_1,T_2} with $|T_1 \cap T_2| \neq \emptyset$.

Lemma 4.11. *There exists $0 < \alpha < 1$ and n_0 a positive integer such that for every partition R, Y of the complete graph on $n \geq n_0$ vertices V, the following holds: there exist subsets $A \dot\cup B \dot\cup C \subseteq V$, and a graph $G = (V, E)$ such that:*

1. *$|A| = |C| \geq \alpha n$ and $|B| \geq \alpha n/3$,*
2. *$E \subseteq (A \times B) \dot\cup (B \times C) \dot\cup (C \times A)$ and $G|_{(A,C)}$ is a complete bipartite graph,*
3. *$G|_{(A,B)}$ is a subgraph of (V, R) and $G|_{(C,B)}$ is a subgraph of (V, Y),*
4. *for every subset $T_1, T_2 \subseteq B$, the graph $G^{T_1,T_2} = (V, E \cap ((T_1 \times A) \dot\cup (T_2 \times B) \dot\cup (A \times C)))$ is bipartite if $T_1 \cap T_2 = \emptyset$, and has distance at least $\alpha |T_1 \cap T_2|/(9n)$ from any bipartite graph*

Proof. Let α, n_0 be suitable for Lemma 4.10, and assume $n \geq n_0$. Apply Lemma 4.10 to the complete graph on V with partition R, Y to obtain sets V_0^1, V_0^2, V_1, V_2 of size $r = \alpha n$.

Let $A = V_1$, $C = V_2$ and $B = V_0^1 \cap V_0^2$. Note that $|B| \geq r/3 = \alpha n/3$ by Property (2) of Lemma 4.10. We now let G be the graph obtained by merging the complete bipartite graph from A and C, the edges in R between A and B, and the edges in Y between C and B. This graph satisfies Properties (1), (2) and (3) of the Theorem.

We now prove Property (4). Let $T = T_1 \cap T_2$. If $T = \emptyset$ then G^{T_1,T_2} is clearly bipartite, thus we only consider the case $T \neq \emptyset$. We first show that G^{T_1,T_2} contains at least $|T| \times r^2/9$ triangles. By Property (3) of Lemma 4.10, the vertices of T are connected to at least $r/3$ vertices of A. The same is true for C. Since $G^{T_1,T_2}|_{(A,C)}$ is a complete bipartite graph, we have (at least) the required number of triangles. Now, observe that removing an edge in G^{T_1,T_2} removes at most r triangles. Thus G^{T_1,T_2} is at distance at least $|T|r/(9n^2) \geq \alpha |T|/(9n)$ from the family of bipartite graphs. □

4.4.2 ε-bipartiteness and half-set disjointness

The goal of this section is to complete the proof of a lower bound on the best-case partition communication complexity for the relaxed bipartiteness problem, by giving the reduction from the half-set problem to ε-bipartition problem, whenever $\varepsilon > 0$ is sufficiently small. What remains to be shown is that for any partition R, Y of the edges of the complete graph on n vertices, any instance of the half-set disjointness can be mapped to an instance of ε-bipartiteness. The key ingredient in the proof is the family of multi-triangle graphs.

Lemma 4.12. *There exists $\varepsilon_0 > 0$ and n_0 a positive integer such that for every $0 < \varepsilon < \varepsilon_0$ and $n \geq n_0$ the following holds: for every partition R, Y of the complete graph on n vertices V, there exists an integer $m = \Theta(n)$ such that the half-disjointness problem in any set S of size m is reducible to the ε-bipartition problem in V for the partition R, Y.*

Proof. Applying Lemma 4.11 for a sufficiently large n, we obtain $0 < \alpha < 1$, subsets $A \dot{\cup} B \dot{\cup} C \subseteq V$, and a graph $G = (V, E)$ that satisfy Properties (1)–(4) of Lemma 4.11. We let $m = |B|$ and identify B with S. For every pair of subsets $T_1, T_2 \subseteq S$ of size $\lfloor |S| \rfloor /4$, let $T = T_1 \cap T_2$. Set $\varepsilon_0 = \alpha^2/216$. Then the following holds for evey $0 < \varepsilon < \varepsilon_0$:

1. the set of edges in R (respectively the edges in Y) in G_T can be computed from T_1 (respectively T_2).
2. if $T = \emptyset$ then G_T is bipartite,
3. if $|T| \geq |T_1|/2$ then G_T has distance at least $\varepsilon_0 > \varepsilon$ from any bipartite graph.

This concludes the reduction. □

4.5 Improving the lower bound

The lower bound presented in this section is not optimal for very small ε. In particular, when $\varepsilon = 0$, the lower bound of Hajnal, Maass and Turán [15] is $2^{\Omega(n \log(n))}$. It is possible to carry out an argument similar to the one in [15], based on partitions of S instead of two intersecting subsets. A lower bound for connectivity can also be proved using these techniques. A sketch of the proof is given in the conference version of this paper. The theorem in the proceedings claims a general lower bound of $2^{\Omega((1-2\sqrt{\varepsilon})n \log((1-2\sqrt{\varepsilon})n))}$, but the proof in the proceedings is incorrect. Nevertheless, for $\varepsilon = O(\frac{1}{n^2})$, it is still possible to prove a lower bound of $2^{\Omega(n \log(n))}$.

Acknowledgments

We would like to thank Wenceslas Fernandez de La Vega for the proof of Proposition 4.7, Lokam V. Satyanarayana for discussions in Section 4, and Miklos Santha for many comments.

References

[1] N. Alon and M. Krivelevich. Testing k-colorability. To appear in *SIAM Journal on Discrete Mathematics*.

[2] N. Alon, E. Fischer, M. Krivelevich, and M. Szegedy. Efficient testing of large graphs. *Combinatorica*, 20:451–476, 2000.

[3] M. Blum and S. Kannan. Designing programs that check their work. *Journal of the ACM*, 42(1):269–291, 1995.

[4] M. Blum, M. Luby, and R. Rubinfeld. Self-testing/correcting with applications to numerical problems. *Journal of Computer and System Sciences*, 47(3):549–595, 1993.

[5] R. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transactions on Computers*, 35(8):677–691, 1986.

[6] R. Bryant. On the complexity of VLSI implementations and graph representation of boolean functions with application to integer multiplication. *IEEE Transactions on Computers*, 40(2):205–213, 1991.

[7] R. Bryant. Symbolic Boolean Manipulation with Ordered Binary Decision Diagrams. *ACM Computing Surveys*, 24(3):293–318,1992.

[8] E. Clarke, E. Emerson, and A. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8(2):244–263, 1986.

[9] E. Clarke, O. Grumberg, and D. Long. Model checking and abstraction. *ACM Transactions on Programming Languages and Systems*, 16(5):1512–1542, 1994.

[10] E. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 1999.

[11] S. Janson, T. Łuczak, and A. Ruciński. *Random Graphs*. Wiley-Interscience Series in Descrete Mathematics and Optimization, 2000.

[12] O. Goldreich, S. Goldwasser, and D. Ron. Property testing and its connection to learning and approximation. *Journal of the ACM*, 45(4):653–750, 1998.

[13] O. Goldreich and D. Ron. A sublinear bipartiteness tester for bounded degree graphs. *Combinatorica* 19:335–373, 1999.

[14] O. Goldreich and D. Ron. Property testing in bounded degree graphs. *Algorithmica* 32(2):302–343, 2002.

[15] A. Hajnal, W. Maass, and G. Turán. On the communication complexity of graph properties. *Proceedings of the 20th ACM Symposium on the Theory of Computing*, pages 186–191, 1988.

[16] W. Hoeffding. Probability inequalities for sums of bounded random variables. *Journal of the American Statistical Association*, 58, pages 13–30, 1963.

[17] E. Kushilevitz and N. Nisan. *Communication Complexity*. Cambridge University Press, 1997.

[18] K. McMillan. *Symbolic Model Checking*. Kluwer Academic Publishers, 1993.

[19] R. Rubinfeld and M. Sudan. Robust characterizations of polynomials with applications to program testing. *SIAM Journal on Computing*, 25(2):23–32, 1996.

[20] E. Szemerédi. Regular partitions of graphs. In Éditions du CNRS, *Problèmes combinatoires et théorie des graphes*, pages 399–401, 1978.