Electronic Colloquium on Computational Complexity, Report No. 100 (2015)

Succinct Encodings of Graph Isomorphism

Bireswar Das*[1], Patrick Scharpfenecker†[2], and Jacobo Torán‡[2]

[1]Indian Institute of Technology Gandhinagar, India
[2]University of Ulm, Institute of Theoretical Computer Science, Ulm, Germany

June 16, 2015

Abstract

It is well known that problems encoded with circuits or formulas generally gain an exponential complexity blow-up compared to their original complexity.

We introduce a new way for encoding graph problems, based on CNF or DNF formulas. We show that contrary to the other existing succinct models, there are examples of problems whose complexity does not increase when encoded in the new form, or increases to an intermediate complexity class less powerful than the exponential blow up.

We also study the complexity of the succinct versions of the Graph Isomorphism problem. We show that all the versions are hard for PSPACE. Although the exact complexity of these problems is still unknown, we show that under most existing succinct models the different versions of the problem are equivalent. We also give an algorithm for the DNF encoded version of GI whose running time depends only on the size of the succinct representation.

1 Introduction

In many applications, like VLSI design or computer aided verification, graphs and other combinatorial structures present many regularities allowing to encode them in a compact way, much more succinctly than for example the usual adjacency matrices or lists. Galperin and Wigderson [9] studied for the first time the complexity of several graph decision problems when the adjacency relation is presented as a Boolean circuit. The hope was that the succinctly encoded instances contain enough structure to make the considered problem easier to solve. They observed however an exponential blow-up in the complexity of these problems, showing that the regularities that allow a succinct representation, do not really help in order to solve the problem. Several extensions to this work [12, 2, 16, 17] showed that the exponential complexity blow-up is the general

*bireswar@iitgn.ac.in
†patrick.scharpfenecker@uni-ulm.de, Supported by DFG grant TO 200/3-1.
‡jacobo.toran@uni-ulm.de

ISSN 1433-8092

behavior, by proving upgrading theorems for several reducibilities: if a problem is complete with respect to certain low level reducibility for a complexity class, then the succinct version of the problem with its instances encoded as a Boolean circuit is complete for the corresponding exponentially higher class, with respect to polynomial time reducibilities. Other extensions of the original work concentrated in more restricted encodings of the input instances, like Boolean formulas [16] or ordered binary decision diagrams (OBDDs) [5],[17]. Even if these representation models are more restricted than the Boolean circuits, in these works the same exponential blow-up in the complexity of the succinct version of the problem is shown.

Very recently, the class of NC^0 circuits was considered as a succinct model for encoding problem instances. In [10] the authors prove a blow-up result for the complexity of the Satisfiability problem encoded this way.

We introduce here the use of Boolean formulas in conjunctive (CNF) or disjunctive normal form (DNF) in order to encode graphs. This is a considerable restriction with respect to the model based on general Boolean formulas from Veith [17]. The size of these representations is closely related to the bi-clique decomposition of the graphs [6]. We show that in this limited model an upgrade theorem is not possible since there are examples of problems whose complexity does not increase when the input graphs are given in the form of a DNF formula, while it presents an exponential blow-up when the dual representation is considered. In other cases, the complexity of the succinct version of the problem is neither that of the original problem nor exponentially higher. For example the Dominating Set problem becomes PP-complete when the input graphs are encoded as a DNF formula and it becomes complete for NEXP when a CNF formula is considered, while the connectivity problem for directed graphs remains NL complete with a DNF encoding and becomes complete for PSPACE with a CNF encoding. We present several other examples in Section 3. This is a new phenomenon in the area of succinct representations, since in all the existing models and examples the complexity of the problem blows up exponentially in the succinct version.

The graph isomorphism problem, GI, asks whether there is a bijection between the nodes of two given graphs preserving the adjacency relationship. The problem has been extensively studied (see e.g [11]) because of its graph theoretic importance, but also because it is one of the few problems in NP whose exact complexity is unknown: the problem is not known to be solvable in polynomial time but also it is not expected to be NP-complete. We study here the complexity of the succinct version of this problem considering several input representations. The motivation for this is twofold. On the one hand the complexity of the succinct version of GI might shed some light to the complexity of the standard version of the problem. On the other hand, algorithms for succinct GI that go beyond the trivial decoding of the graph and then the application of an algorithm for the standard problem to it, would be very useful in areas like computer aided verification. Concerning the first goal, since in all the considered models the graph encoded in the input is at most exponentially larger that the input itself, the succinct version of GI is in NEXP. We obtain in Section 6 for all the succinct encoding methods discussed here a hardness result for succinct-GI for the class PSPACE. This is done with the help of the existing upgrade theorems and the hardness results for the standard version of GI.

Although the exact complexity of the succinct versions of GI is still unknown,

we prove that in most of the encoding models there is no difference in the complexity of the succinct problem. We show that cir(GI), cnf(GI) and dnf(GI) are equally powerful by giving polynomial time reductions between all these problems. Additionally, we reduce obdd(GI) to dnf(GI). This contrasts with the computational power of the models, since it is well known that OBDDs and formulas can be exponentially larger than Boolean circuits computing certain functions, and that CNF formulas and OBDD's cannot simulate each other without superpolynomially increasing their size.

Concerning the second point in the motivation, we give an algorithm for cnf(GI) (or for dnf(GI)) whose complexity depends solely on the size of the encoding formulas. Based on kernelization methods from the area of parametrized algorithms [6], we present an algorithm that on input two CNF formulas F_1 and F_2 with $2n$ variables each and at most s clauses, encoding graphs of size 2^n, decides whether the encoded graphs are isomorphic in time $O(2^{\sqrt{s}2^{O(s)}})$. This presents an improvement over the straightforward method decoding the graphs and then applying the fastest known isomorphism algorithm to them, in the cases in which s is smaller than n.

The rest of this paper is organized as follows: after a preliminary section explaining our notation and basic definitions, we introduce in Section 3 the CNF and DNF succinct encodings and show some examples on how the complexity of the succinct versions can vary. Section 4 shows the equivalence for encodings in case of the GI problem. Based on these results, we present in Section 5 an algorithm for graphs encoded with DNF formulas. Finally, we show in Section 6, that all the considered succinct versions of GI, obdd(GI), cnf(GI), dnf(GI) and cir(GI) are hard for PSPACE.

2 Preliminaries

We refer the reader to standard textbooks for basic definitions and notation in complexity theory including complexity classes, reductions and graphs. Building on that, we present some further definitions.

$\leq_m^p$ denotes the polynomial time many-one reduction. With $\leq_m^{\mathrm{LT}}$, we denote a many-one reduction defined by a function f so that computing the i-th bit of $f(x)$ (denoted by $f(x)_i$) can be done in logarithmic time in $|x|$. For this we consider a machine model with direct access to the input. This kind of machine has a special query tape, which on input a position, outputs the bit written in the corresponding input position. This is a very weak type of reduction since it has very limited access to the input x [2]. $\leq^{\mathrm{qfr}}$ denotes the quantifier free reduction. We include the basic definitions in descriptive complexity and Finite Model Theory (see for example [17] and [3]) needed to understand this reduction.

Let $\tau = (R_1, \ldots, R_s, c_1, \ldots, c_t)$ be a vocabulary representing for all i, j, R_i a_i-ary relations and constants c_j. Then A with a universe U of elements is a finite structure of τ if it consists of instantiations $R_i^A \subseteq U^{a_i}$ and $c_j \in U$ for all R_i and c_j in τ. Let $struct(\tau)$ be the set of all finite τ-structures. We call $C \subseteq struct(\tau)$ a class and associate with it the decision problem: given x, is $x \in C$?

For example, if $\tau = (E, s, t)$ with arity 2 for E, constants s and t as well as $U = [n]$, then $struct(\tau)$ is the set of all n-node graphs with two distinguished

nodes s and t. Then STCONN is the class of all graphs G with nodes s and t for which there is a path in G from s to t.

Now an interpretation I of $struct(\tau)$ into $struct(\sigma)$ for given vocabularies $\tau = (R_1, \ldots, R_s, c_1, \ldots, c_t)$ and $\sigma = (S_1, \ldots, S_u, d_1, \ldots, d_v)$ is a tuple of first-order formulas $I = (\phi, \phi_1, \ldots \phi_u, \chi_1, \ldots, \chi_v)$ with dimension $k \in \mathbb{N}$ such that given $A \in struct(\tau)$ and $B \in struct(\sigma)$, $I(A) \in struct(\sigma)$ with $U^B = \{(a_1, \ldots, a_k) \in (U^A)^k | (a_1, \ldots, a_k) \models \phi\}$ and for all i, j $S_i = \{(a_1, \ldots, a_k) \in (U^A)^k | (a_1, \ldots, a_k) \models \phi_i\} \cap U^B$ and $d_j = (a_1, \ldots, a_k) \in U^B$ such that $d_j = (a_1, \ldots, a_k) \models \chi_j$. The first-order formulas in I are allowed to use quantifiers $\exists$ and $\forall$, connectives $\wedge, \vee, \neg$, relations and constants of A as well as 1, max and $succ(\_, \_)$ according to U.

A logical reduction from $A \subseteq struct(\tau)$ to $B \subseteq struct(\sigma)$ is a k-dimensional interpretation I from A into B such that for all $x \in struct(\tau)$, $I(x) \in A$ iff $x \in B$. A logical reduction is called a quantifier-free reduction (denoted by $\leq^{\text{qfr}}$) if its defining formulas in I are quantifier-free. Note that these reductions are even weaker than logarithmic-time reductions and are closed under composition.

2.1 Succinct Encodings

A Boolean circuit or formula $C(x, y)$ (where x and y are variable vectors) can be interpreted as a succinct encoding of a graph $G = (V, E)$ (or any other structure). If $|x| = |y| = n$, we consider that $C(x, y)$ represents a directed graph on the set of vertices $V = [2^n]$ and edges defined by $C(x, y) = 1$ iff $(x, y) \in E$. Note that the encoded graph has exponential size in n. We say that C encodes the graph G_C. If we talk about undirected graphs, we define $\{x, y\}) \in E$ iff $C(x, y) = 1$ or $C(y, x) = 1$ Also formulas in CNF and DNF can be encoded in a similar way and we consider F_C, to be the CNF formula with $2^{|x|}$ literals and $\leq 2^{|y|}$ clauses with $C(x, y) = 1$ iff literal x is in clause y (similarly for formulas in DNF). We only consider non-empty clauses.

We will consider the input model in which C is a CNF or DNF formula. Note that every graph can be encoded with a polynomially sized DNF formula with each edge encoded as a single implicant. A CNF encoding is similar with each non-edge encoded as a single clause. This trivial observation shows that the problems encoded as CNF or DNF formulas cannot be easier than the original problems, for example, the Dominating Set problem for graphs encoded as CNF formulas is hard for NP.

Looking at graphs encoded as DNF or CNF formulas, we note that in the first case, each implicant or term adds a directed biclique to the total graph (a biclique is a complete bipartite graph). The source side of the biclique is defined by the set of vertices whose labels satisfy the x part of the implicant, while the target side of the biclique is defined by the vertices satisfying the part of the y variables. Therefore a graph encoded by a DNF with m implicants can be decomposed as the union of m bicliques.

In the case of a CNF formula, each clause subtracts a directed biclique from the complete graph K_{2^n}. Every clause removes all edges between the set of x and y nodes falsifying the clause.

Another class of succinct encodings considered in the literature [5, 17] is that defined by ordered binary decision diagrams, OBDDs. A binary decision diagram is a directed, acyclic, rooted graph. Vertices are labeled with input variables x_i and edges are labeled with 0 or 1. Every vertex (except the two

sinks) has two outgoing edges and they have different labels. There are two special vertices without successors, denoted 0 and 1, denoting the Boolean values true and false. A BDD O describes a Boolean function f over a set of variables $x_1, \ldots, x_n$ as follows: beginning at the root, in each vertex labeled with a variable the outgoing edge labeled with the value of this variable is followed, until the 0 or the 1 vertex is reached. This represents the value of the function. O is an OBDD if there is a permutation π on the set of variables such that in all the paths from the root to one of the special vertices the variables are consistent with the order defined by π.

Two graphs G, H are isomorphic, (represented by $G \cong H$) iff there is a bijection $\pi : V_G \to V_H$ such that for all pairs of vertices $u, v \in V_G$, $(u, v) \in E_G$ iff $((\pi(u), \pi(v)) \in E_H$ holds. The decision problem, given two graphs, determine if they are isomorphic, is denoted as GI.

cir(GI), cnf(GI), dnf(GI) and obdd(GI) are the succinct versions of the isomorphism problem when the graphs are encoded respectively by circuits, CNF, DNF formulas or OBDDs.

3 Succinct CNF and DNF Encodings

We present in this section CNF and DNF encodings of several well known problems, showing that in some cases the inputs given in this way change the complexity of the problems while in others the complexity of the original problem is preserved. Figure 1 summarizes our results. We give formal definitions for the decision problems used. It is well known that all these problems are NP-complete, except for STCONN, which is complete for NL.

Dominating Set: Given an undirected graph $G = (V, E)$ and a $k \in \mathbb{N}$. Is there a set $S \subseteq V$ with $|S| \leq k$ such that for all $u \in V$ there is a $v \in S$ with $(u, v) \in E$?

CNF-SAT: Given a boolean formula $F(x_1, \ldots, x_n)$ in conjunctive normal-form. Is there an assignment $s : \{x_1, \ldots, x_n\} \to \{0, 1\}$ such that $F(s(x_1),$ $\ldots, s(x_n)) = 1$?

STCONN: Given a directed graph $G = (V, E)$ and $s, t \in V$. Is there a path from s to t in G?

k-COL: Given an undirected graph $G = (V, E)$ and a $k \in \mathbb{N}$. Is there a mapping $c : V \to \{1, \ldots, k\}$ such that for all $e = (i, j) \in E$, $c(i) \neq c(j)$?

Longest Induced Path (LIP): Given an undirected graph $G = (V, E)$ and a $k \in \mathbb{N}$. Is there an induced path of size at least k?

Thereby an induced path is an ordered subset S of l nodes with exactly the edge set $E = \{(s_i, s_{i+1}) \mid 0 < i < l\}$. On hypercube graphs this problem is better known as Snake-In-The-Box.

Theorem 1. *dnf(Dominating Set) is PP-complete.*

Proof. We show that dnf(Dominating Set) is in the class PP. For this we sketch an algorithm that on input a DNF formula F and a number k decides if the graph encoded by the formula contains a dominating set of size $\leq k$. The algorithm works in polynomial time with the help of non-adaptive queries to PP. Since PP is closed under truth-table reductions [7], this proves the result. A dominating set consists of all isolated vertices (vertices with degree 0) together with a subset of the vertices dominating all vertices with degree greater than 0.

Problem	**CNF encoding**	**DNF encoding**
Dominating Set	NEXP-complete	PP-complete
CNF-SAT	NEXP-complete	NP-complete
STCONN	PSPACE-complete	NL-complete
k-COL	NEXP	Σ_2
Longest Induced Path	NEXP-complete	NP-complete

Figure 1: Complexity differences between DNF and CNF encodings

The key observation is that this second subset cannot be larger than $2m$, where m is the number of implicants in F. Recall that an implicant contains an x and y part where the vertices consistent with the x literals share an edge with all the vertices consistent with the y literals. Therefore taking at most one vertex satisfying the x part and one satisfying the y part in each of the m implicants is enough to dominate all vertices with degree greater than 0. We consider the list of all possible pairs (i, j) with $0 \leq i \leq 2m$ and $j = k - i$ and ask two queries for each pair: 1) is the number of vertices with degree 0 at most j? and 2) is there a subset of vertices of degree at least 1, of size $\leq i$ and such that all the edges defined by an implicant has an endpoint in the subset. It is not hard to see that both are PP queries. We can conclude that there is a dominating set of size at most k if and only if, for at least one of the $2m$ pairs both queries are answered positively.

Observe that in the definition of the problem we only require that the vertices of degree at least 1 are dominated (not considering the isolated vertices), then the above argument shows that this version of the problem is in NP.

For the hardness proof, we reduce the following problem, known to be PP-complete to Dominating Set. Given a Boolean formula F in DNF and a number k, does F have at least k satisfying assignments?. Let $x_1, \ldots, x_n$ be the set of variables in F. We construct a graph on 2^{n+1} vertices (two copies of the possible assignments for F with a extra initial bit in the variable x_0). We add to every implicant in F the literals $\overline{x_0}$ and $y_0, y_1, \ldots y_n$. The set of edges encoded by the new formula are those connecting a satisfying assignment of F (with an additional initial 0, with the vertex 1^{n+1}. The number of isolated vertices in the encoded graph is exactly $2^n - 1$ plus the number of non-satisfying assignments of F. The vertex 1^{n+1} dominates all other vertices. Therefore the original formula has at least k satisfying assignments if and only if the graph has a dominating set of size at most $2^n + 2^n - k$. This defines a polynomial time reduction. □

Using the next result for cnf(CNF-SAT) and the standard reduction from SAT to Dominating Set, it is not hard to see that cnf(Dominating Set) is also NEXP-complete.

Theorem 2. *cnf(CNF-SAT) is NEXP-complete and dnf(CNF-SAT) is NP-complete*

Proof. It is clear that cnf(CNF-SAT) is in NEXP. For showing the hardness we use a recent result from Jahanjou, Miles and Viola [10]. There it is shown that the satisfiability problem for formulas in 3-CNF, when encoded with polynomially many NC^0 functions is NEXP-complete. In their setting the formula

F being tested for satisfiability is encoded in a way in which an NC^0 function f_k gets as input a clause index from F and computes the k-th bit of the three literals contained in that clause. We reduce this problem to cnf(CNF-SAT). For this we make a transformation between both types of encodings. We use a single formula encoding the literal-clause relation of F instead of a polynomial amount of NC^0 functions computing the bit encoding of a clause. For this we first transform each of the circuits computing the f_k functions into three NC^0 circuits, with one output bit each, computing $f_{i,j}$ for the j-th bit of the i-th literal ($1 \leq i \leq 3$). Each such function depends only on constant many input bits and can therefore be computed by a constant size CNF. The following circuit computes the literal-clause relation for a clause encoded in the x variables and a literal encoded in the y variables.

$$\bigvee_{i \in \{1,2,3\}} \bigwedge_{j \leq |y|} f_{i,j}(x) = y_j$$

Since $f_{i,j}$ can be represented by a constant size formula, $f_{i,j} \leftrightarrow y_j$ can also be expressed by a constant size CNF formula. The conjunction of these $|y|$ formulas (one for each j) is still a CNF. Finally transforming the disjunction of the 3 resulting formulas (one for each i) into CNF has size $O(|y|^3) = O(poly(n))$. This completes the reduction.

To show that dnf(CNF-SAT) is in NP and therefore NP-complete we use the same idea as in the algorithm in Theorem 1. A clause is satisfied if we set at least one of its literals to 1. As we again have the biclique structure, a satisfying assignment is a (directed) Dominating Set for the clause-nodes which does not contain a negated and non-negated variable. Besides isolated vertices, a Dominating Set is of size at most polynomial in n. By convention, there are no isolated vertices (empty clauses). So we just guess $|F|$ many literals and check whether they satisfy all clauses by checking that the clause-sides of every biclique are covered and there are no two dual literals. □

Theorem 3. *cnf(STCONN) is PSPACE-complete and dnf(STCONN) is NL-complete.*

Proof. For the first part, we note that cir(STCONN) is complete for PSPACE (see [2]) and that the reduction between graphs we will use in the proof of Theorem 6 for showing cir(GI) $\leq_m^p$ cnf(GI) is connectivity preserving.

We now give an NL-algorithm for dnf(STCONN). Given a DNF-formula F and two nodes s, t, we first guess a term D_i by its index i in F and check if s is in the source-set of this biclique. We now "travel" this biclique to the target set. We repeatedly guess new bicliques and check if the new bicliques source-set is compatible with the old target-set. Then the intersection of these two sets is non-empty and it is valid to use both bicliques successively. We continue this walk until we hit a target-set containing t. As we allways need to remember only two bicliques, $O(\log n)$ space is sufficient. □

Theorem 4. *dnf(k-COL) is in* Σ_2.

We first need the following Lemma:

Lemma 1. *Every biclique defined by a term in an DNF-formula on* $2n$ *variables having intersecting source- and target-sets can be replaced by at most* $poly(n)$ *new non-intersecting bicliques represented by the same number of terms.*

Proof. Given a DNF formula F on variables $x_1, \ldots, x_n, y_1, \ldots, y_n$, let $I, I' \subseteq \{1, \ldots, n\}$ be arbitrary with $I \cap I' \neq \emptyset$. Let our biclique be the term $D = \{x_i | i \in I\} \cup \{y_i | i \in I'\}$,(negated variables work similarly). We illustrate the construction of the new bicliques with an example, see figure 2. First of all, note that the intersection itself represents a complete clique. Such a complete graph on n nodes can be represented with $\log n$ bicliques because all these nodes can be represented with a $\log n$ bit string (they are all in the intersection of the source and target parts of a term) and then, for every position j, we can a biclique between all nodes with the bit in this position set to 0 and those with j-th bit set to 1. The number of blicliques is therefore logarithmic in the clique size.

In a second step, we create a set S_i for every variable x_i, $i \in I' \setminus I$ not appearing in the source-set but in the target-set and negating it. Similarly, there is a set T_j for each $y_j \in I \setminus I'$. We now add all bicliques $S_i \times T_j$ and bicliques between all S_i to the intersection set $I \cap I'$ and from this set to all T_j. The new graph contains all the edges from the original one (except the self loops). Figure 2 shows the construction for such an intersecting biclique. As the number of variables is polynomial, we only add a polynomial number of bicliques. □

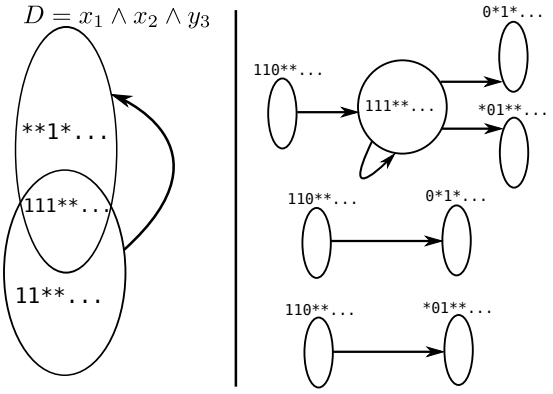

Figure 2: Replacing intersecting by non-intersecting bicliques.

We can now prove the Theorem:

Proof. We first apply the transformation explained in Lemma 1 to the formula describing the graph and get an equivalent DNF-formula where all bicliques source- and target-sets do not intersect. We further observe that each side of a biclique can use at most $k-1$ of the k colors. Else the other side cannot be colored. So if the graph is k colorable, there is a coloring in which every side of a biclique uses at most $k-1$ colors. A Σ_2 algorithm for k-colorability works in the following way: The existential part first guesses for every biclique side which subset of colors it uses. We call this an inconcrete coloring. Although the concrete coloring is not defined (only the set of colors of a set). We can check with the universal part of the algorithm whether a k-coloring exists. We check for all (exponentially many) nodes that 1) the intersection of colors of all

biclique-sides it belongs to is not empty and 2) at least one of these colors is not used by any of the neighbouring biclique-sides.

Such an inconcrete coloring exists of course if the graph is k-colorable. On the other hand, if there is an inconcrete coloring satisfying the above two properties, it is safe to assign such a color to every node and get a correct k-coloring. □

Note that the above algorithm even works for the general colorability problem where k is part as the input as long as $k \in poly(n)$. But it fails if the number of colors is more than a polynomial, since the size of a witness then is not polynomially bounded. It would be interesting to bypass this restriction.

Theorem 5. *cnf(LIP) is NEXP-complete and dnf(LIP) is NP-complete.*

Proof. For the first part, we just note that the reduction in [18], Theorem 1 case 2 can be performed in logarithmic time for every edge when using an apropriate encoding for the nodes. Therefore, $CNF-SAT \leq_m^{\mathrm{LT}} LIP$ and, with the Conversion Lemma (see Lemma 2) $cir(CNF-SAT) \leq_m^p cir(LIP)$. As stated before for the cir(STCONN) problem, the reduction we will introduce in Theorem 6 is connectivity preserving and, in addition, replaces every edge by two consecutive edges with a new node in the middle. This new node has only one additional edge to another new node with no other edges. So basically every induced path in a circuit encoded graph is an induced path of double size in the CNF encoded graph. In summary, we reduce the cir(LIP) problem, given a circuit C and a k to a CNF formula F and $2k$. This shows that cnf(LIP) is NEXP-complete.

We give an NP algorithm for the second part of the result. So for a given formula F and a number k, we observe that an induced path can use at most 2 edges (i,j) and (u,v) of every biclique. These two edges have to occure successive with $j=u$ or else there would either be an additional edge (i,u) or (i,v) (depending on the direction the second edge), contradicting the property that the induced graph is a path. Without loss of generality, let $j=u$.

Suppose there is a third edge (x,y). Again, this edge has to occur directly before or after i,j,v. Suppose $v=x$ and the three edges are (i,j), (j,v) and (v,y). Then there is an edge (i,y), contradicting our assumption.

Since every biclique can only contribute at most 3 successive nodes and we have only $poly(n)$ bicliques, the longest induced path is polynomially bounded. If, given F and k, $k > 3 \cdot |F|$, we reject. Else we guess k nodes and check if this is an induced path. □

4 Succinct Encodings of GI

In this and the following sections we concentrate on the complexity of the different succinct versions of GI. We start by investigating the relation between DNF, CNF, circuit- and OBDD encodings for this problem. Theorem 6 states our results. As described in the preliminaries, we have chosen to use encodings for directed graphs. This is not a restriction for studying the complexity of GI since both versions of the problem, for directed or undirected graphs, are equivalent.

Theorem 6. *dnf(GI)* $\equiv_m^p$ *cnf(GI)* $\equiv_m^p$ *cir(GI) and obdd(GI)* $\leq_m^p$ *cnf(GI).*

Proof. Obviously, cnf(GI) $\leq_m^p$ cir(GI), since a Boolean formula is a restricted version of a circuit. The equivalence between dnf(GI) and cnf(GI) follows from the observation, that two graphs are isomorphic iff their complementary graphs are isomorphic. Given a CNF (DNF) formula encoding a graph, the negation of the formula can be easily written as a DNF (CNF) formula and encodes the complementary graph. More interesting is the proof of cir(GI) $\leq_m^p$ cnf(GI).

Given two circuits $C(x, y)$ and $C'(x, y)$ with $|x| = |y| = n$, encoding two graphs $G = G_C$ and $H = G_{C'}$ on the vertex set $V = [2^n]$, we create two formulas F and F' encoding two new graphs $G' = G_F$ and $H' = G_{F'}$ such that $G \cong H \Leftrightarrow G' \cong H'$. We give first an intuitive description of our construction of F. We use the so called Tseitin transformation from circuits to satisfiability equivalent CNF formulas (see for example [13]). In this transformation a Boolean circuit C is transformed into a satifiability equivalent formula F_C by introducing for each each gate g in C a new variable z_g and a small set of clauses expressing the value of the gate for a certain input. For example if gate g is an OR gate with the input gates e and f we add clauses expressing the subformula $z_g \leftrightarrow (z_e \vee z_f)$. The transformation also adds the single variable clause z_{output} for the output gate of the circuit. It should be clear that C is satisfiable if an only if F_C is satisfiable.

Going back to our construction, we can get from $C(x, y)$ a formula $F_1(x, y, z)$ which encodes the structure and evaluation of the gates of C on input x and y. Variable vector z contains a bit for each gate in C. The satisfying assignments for $F_1(x, y, z)$ consists of values for the x and y variables satisfying $C(x, y)$ plus further assignment values for the z variables with the value of the corresponding gate in C on input x, y. z contains $2n$ additional bits (at the beginning) containing a copy of x and y. This implies that for each satisfying assignment x, y for C, there is a unique z such that $F(x, y, z) = 1$. Moreover, no other satisfying assignment x', y' shares this particular z. If C has s gates, the new formula has linear size in s, is in 3-CNF, and can be interpreted as an encoding for a hypergraph G_1 on the set of vertices $[2^{2n+s}]$ that contains only hyperedges of degree 3. In a second step, these hypergraphs are transformed into standard graphs

We now give a more detailed construction of F. F includes a constant number of clauses for each gate i in C (represented as D_i in the following formula, encoding the evaluation of this gate. We also separate the original vertices encoded in the x and y assignments from the new z vertices by forcing the x, y vertices to begin with 0 and the z vertices with 1. This can be done with one additional variable. The succinct formula model assumes that all vertices are encoded with the same number of bits. To achieve this, we pad the x and y vectors with $(s + 2n + 1) - (n + 1) = s + n$ zero bits. The last line of the following formula enforces that in each satisfying assignment, the first $2n$ bits of z contain the values of x and y at the beginning, making it unique.

$$\begin{aligned} F(x, y, z) = \quad & D_1 \wedge \ldots \wedge D_s \\ & \wedge (x_0 = 0) \wedge (y_0 = 0) \wedge (z_0 = 1) \\ & \wedge \textstyle\bigwedge_{i=n+1}^{s+2n+1} (x_i = y_i = 0) \\ & \wedge \textstyle\bigwedge_{i=1}^{n} (x_i = z_i) \wedge \bigwedge_{i=n+1}^{2n} (y_{i-n} = z_i) \end{aligned}$$

Let G_1 be the hypergraph encoded by $F(x, y, z)$ (analogously for H_1). We claim:

Claim 1. $G \cong H \Leftrightarrow G_1 \cong H_1$

Proof: Clearly, if $G \cong H$, then $G_1 \cong H_1$ since an isomorphism between G and H can be extended to map the z vertices according to the unique hyperdedge they belong to.

Conversely, suppose $G_1 \cong H_1$ via an isomorphism ρ. Observe that all (non isolated) z vertices belong to exactly one hyperedge of 3 nodes. If ρ maps z vertices to other z vertices then ρ defines an isomorphism between G and H. Suppose that there is a z vertex being mapped to one of the x vertices. Then x belongs to a unique hyperdedge. We look at two cases:

- Both neighbors of x belong to a unique hyperedge. Then these three vertices define an isolated hyperedge. ρ can be easily modified to another isomorphism respecting the z vertices, by mapping z to the z neighbor of x. This also defines an isomorphism between G and H.
- Only one of the neighbors of x belongs to a unique hyperedge. Then x is connected to z and y of degree > 2. In the original graphs, x only had one neighbor (y). Again, ρ defines an isomorphism between G and H by swapping the roles of x and z.

■

In a second step we create two standard graphs G_2 and H_2 such that $G_1 \cong H_1 \Leftrightarrow G_2 \cong H_2$. The vertex set of these graphs is the set of assignments for the variable vectors u, x, y, z with $|u| = 1$. Each such vertex encodes a hyperedge (x, y, z) in the previous construction and an additional bit. The formula $F^*(x^*, y^*) = F'(x^*, y^*) \vee F''(x^*, y^*)$, made of the following two subformulas, implements this transformation.

$$\begin{aligned} F'(uxyz, vx'y'z') = \quad & F(x,y,z) \wedge (u=0) \wedge (v=1) \\ & \wedge \textstyle\bigwedge_{i=0}^{|x'|-1} x'_i = y'_i = 0 \\ & \wedge (z' = y \vee z' = z) \end{aligned}$$

$$\begin{aligned} F''(vx'y'z', uxyz) = \quad & F(x,y,z) \wedge (u=0) \wedge (v=1) \\ & \wedge \textstyle\bigwedge_{i=0}^{|x'|-1} x'_i = y'_i = 0 \\ & \wedge (z' = x \vee z' = z) \end{aligned}$$

Note that u and v are single bit variables. F^* encodes a graph that is the union of two vertex sets. The first one, encoded with a leading $u = 0$ encodes the set of all degree 3 hyperedges. The second one is the set of all vertices in G_F. These are forced to begin with 1 and are padded with zeroes.

F' adds edges from a hyperedge vertex $\{x, y, z\}$ to the vertices y and z. Similarly F'' adds edges from x and z to the hyperedge $\{x, y, z\}$. This encodes the directed edge (x, y) in G_F.

F^* is a CNF formula. The whole construction from C to F^* only needs polynomial time. Applying this transformation to C and C' (using some additional dummy gates to get equal circuit sizes) gives us the formulas F and F' satisfying for $G' = G_F$ and $H' = G_{F'}$

$$G \cong H \Leftrightarrow G' \cong H'$$

The statement obdd(GI) $\leq_m^p$ cnf(GI) follows from the fact that an OBDD for a Boolean function can be transformed into a polynomial size Boolean circuit for the function, with the above reductions from cir(GI) to cnf(GI). □

5 An Algorithm for dnf(GI)

Using the characterization for DNF encoded graphs as the union of bicliques, we give an algorithm for the succinct version of GI when the input graphs are encoded as DNF formulas. The running time of the algorithm depends on the number of implicants in the input.

To achieve this, we extend a kernelization technique explained in [6]. Kernelization is a common method in fixed parametrized algorithms. It basically consists in reducing problem instances to a small part of it from which the complexity of the input can still be recovered. This is called a kernel. In some cases algorithms can be designed having polynomial running time for performing the kernelization plus an exponential running time on the size of the kernel.

Looking at graph isomorphism, our kernelization first consists in merging together all vertices that are in exactly the same set of biclique-sides. Such nodes lie in the same intersection and can be considered equivalent. If the intersection is due to two intersecting biclique-sides it is a clique and the corresponding kernel-node gets a self-loop. Afterwards, we further reduce the kernel by applying a modified kernelization-step. We merge nodes which have the same set of neighbours, including self-loops. So non-adjacent nodes get merged if they have the same set of neighbours and therefore have no self-loops. Similar, adjacent-nodes get merged when they both have self-loops. Doing this in both graphs has to be done with some care, because some information might be lost in this way, namely the number of merged nodes. For example, if one graph contains one vertex connected to vertices a, b and c and the second graph contains two such vertices, the graphs may seem isomorphic after merging these nodes. To avoid this problem, we add some coloring or labeling encoding the number of vertices that are merged. In the previous example, we would give the vertex in the kernel in the first graph a certain color and in the second graph a different one. These colors can be replaced with gadgets. Such a kernelization preserves isomorphism and can be constructed in time polynomial in the size of the given graphs.

If our graphs are unions of s bicliques, we assign to every vertex v a vector $b(v)$ in $\{0,1,2,3\}^s$ where $b(v)_i$ is 1 if v is in the source side of the i-th biclique, meaning that v has outgoing edges to all vertices in the target side of the i-th DNF term. Similar, $b(v)_i = 2$ if v is in the target side of i, $b(v)_i = 3$ if it is both sides and $b(v)_i = 0$ if it is in none of them. Note that, in contrast to [6], we have four cases since we allow a node to be in the source and target side. There are at most 4^s possible vectors. It should be clear that after this first kernelization step the vertices with the same vector are equivalent. There may still be however equivalent vertices left with different vectors, which can create problems when dealing with the isomorphism of two graphs. Consider for example the case in which two different terms of a DNF formula define two different sets of source vertices but with the same target set. The source sets would have different vectors although they might be equivalent. But this and similar problems are easy to detect. For all pairs of different vertices remaining

after the first part of the kernelization, we check if their in- and out-going edges are the same. This is done comparing the union of all relevant source sides and the union of the relevant target sides. If they are the same, we merge these two vectors. This comes at a cost of $(4^s)^2 = 4^{O(s)}$ steps but only decreases the size of the kernel. At last, the kernelization merges all vertices with the same vector and colors this class of nodes with its size.

The kernelization together with an algorithm for GI can be used to compute dnf(GI).

Theorem 7. *Given two DNF formulas F_1 and F_2 with $2n$ inputs each and with at most s implicants, encoding graphs G_1 and G_2 on 2^n vertices, isomorphism for these graphs can be tested in time $2^{\sqrt{s2^{O(s)}}} + 2^{O(n)}$.*

Proof. We describe the algorithm and prove its running time and correctness. On input two DNF formulas we apply the explained kernelization to them obtaining two graphs of size at most $4^s = 2^{O(s)}$ in time $2^{O(s)}$. Using the isomorphism test of Babai and Luks (see [1]), which runs in time $2^{\sqrt{n \log n}}$ on graphs with n vertices, it can be tested whether the kernels are isomorphic. This clearly gives a correct result. The running time is $2^{O(s)}$ for the kernelization as well as $2^{\sqrt{s2^{O(s)}}}$ for the isomorphism test. □

The same algorithm works for CNF formulas by first negating the formula and then applying this algorithm to the resulting DNF formula. If $s \in o(n)$, this provides a better upper bound than obtaining an explicit representation of the graphs (in time $2^n poly(n)$) and applying then the algorithm from Babai and Luks.

Note that this algorithm can be also transformed into an algorithm for cir(GI) by first using our transformation from cir(GI) to dnf(GI) and then applying the kernelization and the isomorphism test. But since the size of the computed DNF formula is $O(c^2)$ where $c \geq n$ is the circuit size, the algorithm would have a worse running time than decoding the graphs from their succinct representations and applying then an isomorphism algorithm to them.

6 Hardness Results for GI

We show in this section that the circuit, DNF, CNF and OBDD succinct versions of GI are hard for PSPACE under polynomial time many-one reducibilities. For this we use the following Conversion Lemma relating the complexity of standard and succinct encodings of the same problem. The Lemma for circuits appeared in [2] and was improved in [4]. The version for OBDD's is from [17].

Lemma 2. *Let $A, B \subseteq \{0,1\}^*$. If $A \leq_m^{LT} B$, then $cir(A) \leq_m^p cir(B)$. If $A \leq^{qfr} B$ then $obdd(A) \leq_m^p obdd(B)$.*

Consider cir(USTCONN), the succinct version of the undirected reachability problem. It is known that this problem is PSPACE-complete under polynomial time many-one reducibilities [2]. Moreover, USTCONN is AC^0 reducible to the complement of GI [14, 15]. We show here that in fact, this reduction can be done in logarithmic time.

Theorem 8. *$USTCONN \leq_m^{LT} \overline{GI}$.*

Proof. Let $G = (V, E)$ be an undirected graph with $|V| = n$ and two designated vertices $s, t \in V$. Consider a graph $G' = G_1 \cup G_2$ where G_1 and G_2 are two copies of G, and for a vertex $v \in V$ let us call v_1 and v_2 the copies of v in G_1 and G_2 respectively. Furthermore, G' is defined to have vertex t_1 labeled with color 1, (the rest of the vertices have color 0). It is not hard to see that there are not any paths from s to t in G if and only if there is an automorphism φ in G' mapping s_1 to s_2. The question of whether there is an automorphism in G' with the mentioned properties, can in turn be reduced to GI by considering the pair of graphs $(\hat{G}, \hat{H})$ where $\hat{G}$ are $\hat{H}$ are copies of G' but with s_1 marked with a new color 2 in $\hat{G}$ and s_2 marked with the same color in $\hat{H}$. It holds that $G \in$ USTCONN iff $(\hat{G}, \hat{H}) \in \overline{\text{GI}}$. Figure 3 illustrates this construction.
It is only left to show that the constructions of the graphs $\hat{G}$ and $\hat{H}$ as well as the color labels in the reduction can be done in logarithmic time. A way to do this, is to consider that graph $\hat{G}$ has $4n$ vertices (the construction of $\hat{H}$ is completely analogous). For each vertex v in V we consider the four vertices abv with $a, b \in \{0, 1\}$. The vertices $00v$ and $11v$ define two exact copies of G, while the vertices $01v$ and $10v$ are used for the color labels of s_1 and t_1. For this we can add edges connecting all the $01v$ vertices between themselves forming a clique and to $00s_1$ and connecting all the $10v$ vertices to $00t_1$ (and not between themselves to distinguish them from the 01 vertices). With this construction, one bit of the adjacency matrix on $\hat{G}$ can be computed in logarithmic time (on input G and the position in the matrix) since at most one position in the adjacency matrix of G is needed for this. □

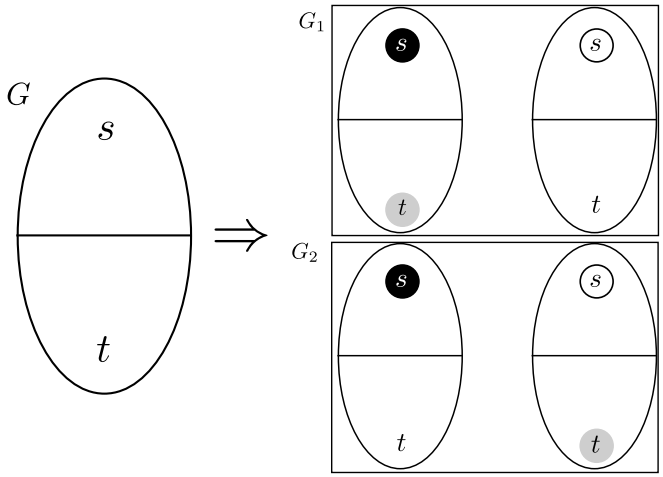

Figure 3: Reducing Reachability to GI

This result, together with Lemma 2 imply that cir(GI) is PSPACE-hard with respect to the polynomial time many-one reducibility. Theorem 6 implies that even DNF encoded GI is hard for PSPACE.

Corollary 1. *dnf(GI) is hard for PSPACE.*

Theorem 8 can in fact be strengthened for the case of a quantifier free reduction [17]. Using the second part of Lemma 2 this proves that also the OBDD version of the problem is hard for PSPACE.

Theorem 9. *USTCONN* $\leq^{qfr} \overline{GI}$.

Proof. We show that $\overline{\text{USTCONN}} \leq^{\text{qfr}} \text{GI}$ with steps $\overline{\text{USTCONN}} \leq^{\text{qfr}} \text{cGI}$ and $\text{cGI} \leq^{\text{qfr}} \text{GI}$. Let $\overline{\text{USTCONN}}$ be the set of all tuples of the form (E^*, s, t) such that $E^* \subseteq U = [n]$ is a set of undirected edges and there is no path from s to t. cGI is the set of all 3-colored graph-pairs $(E, E', c_0, c_1, c_2, c'_0, c'_1, c'_2)$ with a color-preserving isomorphism. Let $I = (\phi, \phi_E, \phi_{E'}, \phi_{c_0}, \phi_{c_1}, \phi_{c_2}, \phi_{c'_0}, \phi_{c'_1}, \phi_{c'_2})$ be a $k = 2$-dimensional interpretation such that

$$\begin{aligned}
\phi(u,v) &= (u=1) \vee (u=2) \\
\phi_E(u,v,u',v') &= E^*(v,v') \\
\phi_{E'}(u,v,u',v') &= E^*(v,v') \\
\phi_{c_0}(u,v) &= \neg(c_1 \vee c_2) \\
\phi_{c_1}(u,v) &= (u=1) \wedge (v=s) \\
\phi_{c_2}(u,v) &= (u=1) \wedge (v=t) \\
\phi_{c'_0}(u',v') &= \neg(c'_1 \vee c'_2) \\
\phi_{c'_1}(u',v') &= (u=2) \wedge (v=s) \\
\phi_{c'_2}(u',v') &= (u=1) \wedge (v=t)
\end{aligned}$$

The reader may verify that this interpretation implements the reduction given in the proof of Theorem 8. We use the first dimension to create two copies of the original graph (denoted with $u = 1$ or $u = 2$) such that the second dimension v, v' is checked against $(v, v') \in E^*$. There are no edges between $u = 1$ and $u = 2$. The coloring is done as described in Theorem 8.

Adding a reduction to GI is easy. A 2-dimensional interpretation adds enough nodes for two gadgets, for example K_{2n} and K_{3n} using the first dimension from $u = 2$ to $u = 6$ as the second dimension contains n nodes. So for $u \in 2, \ldots, 5$ the second dimension contains K_{2n} and K_{3n}. For $u = 1$ the second dimension contains the originaly colored graph. We then assign the color c_1 to the first node with the first dimension of $u = 2$, $(2, 1)$ and c_2 to the first node with $u = 4$, $(4, 1)$. Then a node (u, v) is connected to $(2, 1)$ if $u = 1$ or $u = 2$ and $v \models c_1$. Similar (u, v) is connected to $(4, 1)$ if $u = 1$ or $u = 2$ and $v \models c_2$. The same can be done for the second graph. □

Corollary 2. *obdd(GI) is hard for PSPACE.*

There are stronger hardness results for GI than the one used here [14]. However, applying the Conversion Lemma to these we do not obtain better hardness results for cir(GI). The translation of these results would imply hardness for the class #PSPACE, but this class is known to coincide with FPSPACE [8].

7 Conclusions and Open Problems

We introduced the new CNF and DNF models for encoding problems succinctly. We showed that contrary to the other existing models, there are examples for which the complexity of succinct version of the problem does not blow up exponentially. The size of the graph encoding in the new models are related to certain graph decompositions. It would be interesting to study further examples of graph problems encoded in these models, trying to obtain algorithms acting directly on the succinct versions as we did for the case of GI.

We also studied the complexity of succinct-GI in the different models and proved that although the complexity of this problem is not well understood yet, the versions for GI under DNF, CNF and even circuit encodings are all equivalent. A question that remains open is whether the OBDD version of GI is also equivalent to the other versions or easier.

References

[1] László Babai and Eugene M. Luks. Canonical labeling of graphs. In *Proceedings of the fifteenth annual ACM symposium on Theory of computing - STOC '83*, pages 171–183, New York, New York, USA, 1983. ACM Press.

[2] José L. Balcázar, Antoni Lozano, and Jacobo Torán. The Complexity of Algorithmic Problems on Succinct Instances. *Computer Science, Research and Applications, Springer US*, 1992.

[3] H.D. Ebbinghaus and J. Flum. *Finite Model Theory*. Perspectives in mathematical logic. Springer, 2005.

[4] Thomas Eiter, Georg Gottlob, and Heikki Mannila. Adding disjunction to datalog (extended abstract). In *Proceedings of the thirteenth ACM SIGACT-SIGMOD-SIGART symposium on Principles of database systems - PODS '94*, pages 267–278, New York, New York, USA, May 1994. ACM Press.

[5] Joan Feigenbaum, Sampath Kannan, Moshe Y. Vardi, and Mahesh Viswanathan. Complexity of Problems on Graphs Represented as OBDDs. *STACS 98*, 1998.

[6] Herbert Fleischner, Egbert Mujuni, Daniel Paulusma, and Stefan Szeider. Covering Graphs with Few Complete Bipartite Subgraphs. *FSTTCS 2007: Foundations of Software Technology and Theoretical Computer Science*, 4855, 2007.

[7] Lance Fortnow and Nick Reingold. PP Is Closed under Truth-Table Reductions. *Information and Computation*, 124(1):1–6, 1996.

[8] Matthias Galota and Heribert Vollmer. Functions computable in polynomial space. *Information and Computation*, 198(1):56–70, April 2005.

[9] Hana Galperin and Avi Wigderson. Succinct representations of graphs. *Information and Control*, 56(3):183–198, March 1983.

[10] Hamidreza Jahanjou, Eric Miles, and Emanuele Viola. Local reductions, 2013.

[11] Johannes Köbler, Uwe Schöning, and Jacobo Torán. *The graph isomorphism problem: its structural complexity*. Birkhauser, August 1994.

[12] Christos H. Papadimitriou and Mihalis Yannakakis. A note on succinct representations of graphs. *Information and Control*, 71(3):181–185, December 1986.

[13] Uwe Schöning and Jacobo Torán. *The Satisfiability Problem: Algorithms and Analyses*. Lehmanns Media, 2013.

[14] Jacobo Toran. On the hardness of graph isomorphism. In *Foundations of Computer Science, 2000. Proceedings. 41st Annual Symposium on*, pages 180–186, 2000.

[15] Jacobo Torán. Reductions to Graph Isomorphism. *Theory of Computing Systems*, 47(1):288–299, December 2008.

[16] Helmut Veith. Languages represented by Boolean formulas. *Information Processing Letters*, 63(5):251–256, September 1997.

[17] Helmut Veith. How to encode a logical structure by an OBDD. In *Proceedings. 13th IEEE Conference on Computational Complexity*, pages 122–131. IEEE Comput. Soc, 1998.

[18] Mihalis Yannakakis. The effect of a connectivity requirement on the complexity of maximum subgraph problems. *J. ACM*, 26(4):618–630, October 1979.

ECCC ISSN 1433-8092

http://eccc.hpi-web.de