

Haroldo G. Benatti*
Ruy J.G.B. de Queiroz†

ON THE DESCRIPTIVE COMPLEXITY OF TWO DISJOINT PATHS PROBLEM OVER UNDIRECTED GRAPHS

Abstract

We are concerned with the problem of determining whether an undirected graph (V, E) with distinguished vertices s_1, s_2, t_1, t_2 has node-disjoint paths from s_1 to t_1 and s_2 to t_2 . We show that it is definable in least fixed point logic, meaning that it can be answered in polynomial time the question whether (G, s_1, t_1, s_2, t_2) is a yes instance of the problem by iteratively evaluating a first-order formula on the graph until a fixed-point is reached. We also obtain partial results w.r.t. to the undefinability of the same problem in $\exists L_{\infty\omega}^\omega$.¹

Keywords: descriptive complexity, disjoint paths problem, infinitary logic, Ehrenfeucht games

*Supported by a grant from CAPES (www.capes.gov.br).

†Research partially supported by a Senior Research Fellow grant n. 304365/2003-3 from CNPq (www.cnpq.br). The final version was prepared while in the chair of Edward Larocque Tinker Visiting Professor, Dept Philosophy, Stanford Univ, USA. Thanks are due to Profs. Solomon Feferman and Grigori Mints for their kind nomination to the Tinker Visiting Chair and their great hospitality. Thanks also due to Prof. Herbert Klein, Molly Vitorte, Megan Gorman and Omar Ochoa at the Center for Latin American Studies, Stanford Univ, for making the Visiting Professorship possible, and run so smoothly. A postdoctoral grant n. 2857-05-7 from CAPES (www.capes.gov.br) is also gratefully acknowledged.

¹The results reported here were obtained as part of the first author's doctorate thesis [4].

1. Introduction

Barland [3] has shown that the *Triangle Problem*, which consists of deciding whether there exists a circuit visiting three given vertices in an undirected graph, is definable in $\exists L_{\infty\omega}^{\omega}$. Two natural questions arise:

- Is the *Triangle Problem* definable in *Datalog*(\neq)?²
The most important tool to establish a negative answer to questions of this sort is the k -pebbles existential game. However, in this particular case, it is not applicable since Barland has also shown that the problem is definable in $\exists L_{\infty\omega}^{\omega}$.
- What happens with other homeomorphism problems in undirected graphs?
Here we shall analyse the 2-disjoint paths problem over undirected graphs, which we shall denote by *2DP*.

By way of motivation, let us recall that purely logical algorithms are desirable in situations where one has no rights to manipulate the encoding of the input (e.g., in relational databases). This has had certain impact in the past for other problems and is still an interesting, even if specialized, topic. Moreover, there is a connection between this question and some interesting open questions in finite model theory, which we will point out at the end of the paper.

To start with, we shall present some queries defined in *Stratified Datalog*(\neq), which is the language of negation-free, function-free Horn clauses with inequalities allowed in the bodies of rules (for more details, see [5]). Such queries will be used to write a program in *Stratified Datalog*(\neq) to solve the two disjoint paths problems over undirected graphs in case there exists an expression in *Stratified Datalog*(\neq) for this problem restricted to 3-connected graphs. We will use such an expression to show that the two disjoint paths problem over undirected graphs is definable in $FO + LFP$, the logic obtained from closing first-order logic under least fixed-points of operations definable by positive formulas, as described by Ebbinghaus in [5]. In order to establish this result we will make use of the polynomial

²According to Ebbinghaus & Flum [5], a *Datalog* program is a general logic program, in which no intentional symbol occurs negated in the body of any clause. *Datalog*(\neq) is then the language of negation-free, function-free Horn clauses with inequalities allowed in the bodies of rules.

algorithm presented by Shiloach [12] for this problem, but the result we get tells us something more about the problem, since $FO + LFP \subset \mathcal{P}$.

Later on we will present a partial result for the expressiveness of the two disjoint paths problem for undirected graphs in $\exists L_{\infty\omega}^{\omega}$, the negation-free existential fragment of infinitary logic with a finite number of variables. This result tells us which are the classes of graphs that we must use to prove $2DP \notin \exists L_{\infty\omega}^{\omega}$. Last, but not least, we shall discuss some connections between this problem and preservation problems in Finite Model Theory.

2. Definability of $2DP$ in $FO + LFP$

In this section we demonstrate that the $2DP$ can be expressed in $FO + LFP$.

THEOREM 2.1. $2DP \in FO + LFP$.

PROOF IDEA. To get this result first we show, by a series of reductions, that the existence of an expression in *Stratified Datalog*(\neq) for this problem over undirected graphs follows from the existence of one expression in this language to the version of the problem over 3-connected undirected graphs. Next we observe the existence of a polynomial algorithm to the $2DP$ over 3-connected undirected planar graphs and also a result of Shiloach [12] which asserts the existence of the desired paths to non-planar graphs with an additional property (explained in case 4 below) to conclude that $2DP \in FO + LFP$.

PROOF. In the following we present all the queries that we use to reduce the expressibility of $2DP$ over undirected graphs on *Stratified Datalog*(\neq) to the class of 3-connected undirect graphs.

In the auxiliary queries which we will define below we will make use of a parameter F , instead of a set of edges of the given graph, since in the program we will write we use different sets of edges. Next we will explain each one of the auxiliary queries and present an expression in *Stratified Datalog*(\neq) which defines the corresponding query.

$PathAv1(F; x, y, v)$ is valid if there exists a path (in the graph whose set of edges is F) from x to y which avoids vertex v :

$$\begin{aligned} PathAv1(F; x, y, v) &\leftarrow F(x, y) \wedge x \neq v \wedge y \neq v \\ PathAv1(F; x, y, v) &\leftarrow F(x, y) \wedge PathAv1(F; z, y, v) \\ &\quad \wedge x \neq v \end{aligned}$$

Similarly, $PathAv2(F; x, y, v, w)$ is valid if there exists a path from x to y which avoids v and w :

$$\begin{aligned} PathAv2(F; x, y, v, w) &\leftarrow F(x, y) \wedge x \neq v \wedge x \neq w \\ &\quad \wedge y \neq v \wedge y \neq w \\ PathAv2(F; x, y, v, w) &\leftarrow F(x, y) \\ &\quad \wedge PathAv2(F; z, y, v, w) \\ &\quad \wedge x \neq v \wedge x \neq w \end{aligned}$$

The query $2Star(F; s, u, v)$ which we define next states that there exist two paths, one from s to u and another one from s to v , disjoint with respect to the vertices, except for vertex s . Kolaitis & Vardi [7] noticed that this query is equivalent to the following query $Q(F; s, u, v)$: “There exists, in the graph whose set of edges is F , a path P from s to v , such that for all vertices x (except for $x = s$) in this path there exists a path from s to u which avoids x ”. We make use of such equivalence to define the query $2Star(F; s, u, v)$:

$$\begin{aligned} 2Star(F; s, u, v) &\leftarrow F(s, v) \wedge PathAv1(F; s, u, v) \\ 2Star(F; s, u, v) &\leftarrow 2Star(F, s, u, w) \wedge F(w, v) \\ &\quad \wedge PathAv1(F; s, u, v) \end{aligned}$$

Note that the query $2Star(F; s, u, v)$ is not valid when $u = v$. The following query is a variant of this query which admits this equality:

$$2StarMerge(F; x, y, z) \leftarrow 2Star(F; x, y, w) \wedge F(w, z)$$

Clearly, in a similar way we can define the query $3StarMerge(F; x, y, w, z)$, which tells us that there exist three disjoint paths from x to y , w and z and allows for $y = w = z$.

We say that a graph \mathcal{G} is k -connected if $|\mathcal{G}| > k$ and $\mathcal{G} - X$ is connected for every set of vertices $X \subseteq \mathcal{G}$, with $|X| < k$. A k -connected component of a graph \mathcal{G} is a maximal k -connected subgraph of \mathcal{G} . The relation “two vertices s and t are in the same k -connected component” defines an equivalence relation. So, two vertices s and t are in the same biconnected component if there exist two disjoint paths from s to t , or, equivalently, if there exists a circuit which passes through s and t . We

define the query $Bic(F; s, t)$, which says that two vertices s and t are in the same biconnected component, using the query $2StarMerge$:

$$Bic(F; s, t) \leftarrow 2StarMerge(F; s, t, t)$$

A *block* of a graph \mathcal{G} is maximal connected subgraph of \mathcal{G} which does not have a cut vertex. Thus, a block is a biconnected component, or a bridge, or an isolated vertex. It follows that two vertices are in the same block if there exists a circuit which passes through them or there exists an edge connecting them. However, three vertices are in the same block if all of them are in the same biconnected component.

$$\begin{aligned} SameBl(F; x, y, z) \leftarrow & Bic(F; y, z) \wedge Bic(F; x, y) \\ & \wedge Bic(F; x, z) \end{aligned}$$

Let $S = \{u, v\}$ be a cut set of \mathcal{G} . The graph $\mathcal{G}' = (G', E')$ is a *weak component modulo* S of \mathcal{G} if it is a connected component of $\mathcal{G} - S$. The graph $\mathcal{G}'' = (G' \cup S, E' \cup E'')$, where E'' is the set of edges in \mathcal{G} linking u or v to some vertex of \mathcal{G}' , is called a *strong component of* \mathcal{G} *mod* S . The following query is valid when two vertices s and t are in the same strong component modulo $\{x, y\}$:

$$\begin{aligned} SameStr(F; s, t, x, y) \leftarrow & F(s, t) \wedge s \neq x \wedge s \neq y \wedge s \neq t \\ SameStr(F; s, t, x, y) \leftarrow & SameStr(F, s, w, x, y) \\ & \wedge F(w, t) \wedge t \neq x \wedge t \neq y \end{aligned}$$

Two vertices x and y form a cut set in the graph \mathcal{G} if there exist vertices u and v in this graph such that any path connecting these two vertices passes through at least one of the vertices x or y :

$$CutSet2(F; x, y) \leftarrow \neg PathAv2(F; u, v, x, y)$$

Let $\mathcal{G} = (G, E'', s_1, s_2, t_1, t_2)$ be an undirected graph given as input to $2DP$. Shiloach's algorithm [12] decides if there exist disjoint paths from s_1 to t_1 and s_2 to t_2 by making a series of reductions in this input graph. We shall follow those reductions even if not too closely. But before we proceed, we will add to the input graph the edges (s_1, s_2) and (t_1, t_2) . Clearly, adding this information (i.e., the edges) will not modify the problem, since the

edges will not contribute to form the desired paths. We may consider only the biconnected component of the graph resulting from this operation which contains s_1, s_2, t_1, t_2 . If these vertices are not in the same biconnected component then the input graph \mathcal{G} does not have the desired disjoint paths. Otherwise, we continue with $2DP(s_1, s_2, t_1, t_2)$:

$$\begin{aligned} E'(x, y) &\leftarrow E''(x, y) \\ E'(x, y) &\leftarrow x = s_1 \wedge y = s_2 \\ E'(x, y) &\leftarrow x = t_1 \wedge y = t_2 \end{aligned}$$

$$\begin{aligned} E(x, y) \leftarrow E'(x, y) &\wedge \text{SameBl}(E'; x, s_1, t_1) \\ &\wedge \text{SameBl}(E'; y, s_1, t_1) \\ &\wedge \text{SameBl}(E'; x, s_2, t_2) \\ &\wedge \text{SameBl}(E'; y, s_2, t_2) \end{aligned}$$

$$\begin{aligned} 2DPa(s_1, s_2, t_1, t_2) &\leftarrow \text{Bic}(E'; s_1, t_1) \\ &\wedge \text{Bic}(E'; s_2, t_2) \\ &\wedge 2DP(s_1, s_2, t_1, t_2) \end{aligned}$$

From now on we will use E as the set of edges and we shall omit the parameter for the set of edges which is used, context permitting. In the first part of the algorithm we analyse cut sets of size two which allow us to make a reduction of the problem to an input graph with different distinguished vertices. Intuitively, these vertices are getting closer to each other.

CASE 1.1: $\{s_1, v\}$ is a cut set which separates s_2 from t_1 and t_2 :

In this case we consider the equivalent query $2DP(s_1, v, t_1, t_2)$, where v appears in the place of s_2 in the original query. Clearly, there exist disjoint paths from s_1 to t_1 and from s_2 to t_2 only if there exist two disjoint paths from s_1 to t_1 and from v to t_2 .

$$\begin{aligned} 2DP(s_1, s_2, t_1, t_2) &\leftarrow \text{CutSet2}(s_1, v) \wedge v \neq s_2 \wedge v \neq t_1 \wedge v \neq t_2 \\ &\wedge \neg \text{SameStr}(s_2, t_1, s_1, v) \\ &\wedge \neg \text{SameStr}(s_2, t_2, s_1, v) \\ &\wedge 2DP(s_1, v, t_1, t_2) \end{aligned}$$

Similar cases: $\{s_1, v\}$ is a cut set which separates t_2 from t_1 and s_2 ; $\{s_2, v\}$ is a cut set which separates s_1 from t_1 and t_2 ; $\{s_2, v\}$ is a cut set which separates t_1 from s_1 and t_2 ; $\{t_1, v\}$ is a cut set which separates s_2 from t_2 and s_1 ; $\{t_1, v\}$ is a cut set which separates t_2 from s_2 and s_1 ; $\{t_2, v\}$ is a cut set which separates s_1 from t_1 and s_2 ; $\{t_2, v\}$ is a cut set which separates t_1 from s_1 and s_2 .

CASE 1.2: $\{s_1, t_1\}$ is a cut set, s_2 and t_2 are in the same strong component modulo $\{s_1, t_1\}$.

There exist paths from s_1 to t_1 and from s_2 to t_2 in different strong components modulo $\{s_1, t_1\}$. Hence, these paths are disjoint.

$$2DP(s_1, s_2, t_1, t_2) \leftarrow CutSet2(s_1, t_1) \wedge SameStr(s_2, t_2, s_1, t_1)$$

Similar case: $\{s_2, t_2\}$ is a cut set, s_1 and t_1 are in the same strong component modulo $\{s_2, t_2\}$.

CASE 1.3: $\{s_1, s_2\}$ is a cut set, t_1 and t_2 are in the same strong component modulo $\{s_1, s_2\}$.

Again there exist paths from s_1 to t_1 and from s_2 to t_2 in different strong components modulo $\{s_1, s_2\}$.

$$2DP(s_1, s_2, t_1, t_2) \leftarrow CutSet2(s_1, s_2) \wedge \neg SameStr(t_1, t_2, s_1, s_2)$$

Similar cases: $\{s_1, t_2\}$ is a cut set, t_1 and s_2 are not in the same strong component modulo $\{s_1, t_2\}$; $\{t_1, s_2\}$ is a cut set, s_1 and t_2 are not in the same strong component modulo $\{t_1, s_2\}$; $\{t_1, t_2\}$ is a cut set, s_1 and s_2 are not in the same strong component modulo $\{t_1, t_2\}$.

CASE 1.4: $\{u, v\}$ is a cut set and separates s_1 and s_2 from t_1 and t_2 .

If the input graph has two disjoint paths between s_1, s_2 and t_1, t_2 then such paths must pass through u and v . Then, we proceed to subproblems $2DP(s_1, s_2, u, v)$ and $2DP(u, v, t_1, t_2)$.

$$\begin{aligned} 2DP(s_1, s_2, t_1, t_2) &\leftarrow CutSet2(u, v) \\ &\wedge u \neq s_1 \wedge u \neq s_2 \wedge v \neq s_1 \wedge v \neq s_2 \\ &\wedge u \neq t_1 \wedge u \neq t_2 \wedge v \neq t_1 \wedge v \neq t_2 \\ &\wedge \neg SameStr(s_1, t_1, u, v) \\ &\wedge \neg SameStr(s_1, t_2, u, v) \end{aligned}$$

$$\begin{aligned}
&\wedge \neg \text{SameStr}(s_2, t_1, u, v) \\
&\wedge \neg \text{SameStr}(s_2, t_2, u, v) \\
&\wedge 2DP(s_1, s_2, u, v) \\
&\wedge 2DP(u, v, t_1, t_2)
\end{aligned}$$

Similar case: $\{u, v\}$ is a cut set and separates s_1 and t_2 from t_1 and s_2 .

CASE 1.5: $\{u, v\}$ is a cut set which separates s_1 and t_1 from s_2 and t_2 .

Clearly, there exist disjoint paths from s_1 to t_1 and from s_2 to t_2 .

$$\begin{aligned}
2DP(s_1, s_2, t_1, t_2) &\leftarrow \text{CutSet2}(u, v) \\
&\wedge u \neq s_1 \wedge u \neq s_2 \wedge v \neq s_1 \wedge v \neq s_2 \\
&\wedge u \neq t_1 \wedge u \neq t_2 \wedge v \neq t_1 \wedge v \neq t_2 \\
&\wedge \neg \text{SameStr}(s_1, s_2, u, v) \\
&\wedge \neg \text{SameStr}(s_1, t_2, u, v) \\
&\wedge \neg \text{SameStr}(t_1, s_2, u, v) \\
&\wedge \neg \text{SameStr}(t_1, t_2, u, v)
\end{aligned}$$

CASE 1.6: $\{u, v\}$ is a cut set and separates s_1 from s_2, t_1, t_2 .

If the input graph has disjoint paths from s_1 to t_1 and from s_2 to t_2 then such paths must pass through at least one of the vertices of the cut set $\{u, v\}$. Thus we can replace s_1 with these vertices.

$$\begin{aligned}
2DP(s_1, s_2, t_1, t_2) &\leftarrow \text{CutSet2}(u, v) \\
&\wedge u \neq s_1 \wedge u \neq s_2 \wedge v \neq s_1 \wedge v \neq s_2 \\
&\wedge u \neq t_1 \wedge u \neq t_2 \wedge v \neq t_1 \wedge v \neq t_2 \\
&\wedge \neg \text{SameStr}(s_1, s_2, u, v) \\
&\wedge \neg \text{SameStr}(s_1, t_1, u, v) \\
&\wedge \neg \text{SameStr}(s_1, t_2, u, v) \\
&\wedge 2DP(u, s_2, t_1, t_2)
\end{aligned}$$

Similar cases: $\{u, v\}$ is a cut set and separates s_2 from s_1, t_1, t_2 ; $\{u, v\}$ is a cut set and separates t_1 from s_1, s_2, t_2 ; $\{u, v\}$ is a cut set and separates t_2 from s_1, s_2, t_1 .

CASE 1.7: $\{s_1, v\}$ is a cut set and separates t_1 from s_2 and t_2 .

There exist two desired disjoint paths in different strong components. The path from s_2 to t_2 may eventually use vertex v .

$$\begin{aligned} 2DP(s_1, s_2, t_1, t_2) &\leftarrow CutSet2(s_1, v) \\ &\wedge v \neq s_2 \wedge v \neq t_1 \wedge v \neq t_2 \\ &\wedge \neg SameStr(t_1, s_2, s_1, v) \\ &\wedge \neg SameStr(t_1, t_2, s_1, v) \end{aligned}$$

Similar cases: $\{t_1, v\}$ is a cut set and separates s_1 from s_2 and t_2 ; $\{s_2, v\}$ is a cut set and separates t_2 from s_1 and t_1 ; $\{t_2, v\}$ is a cut set and separates s_2 from s_1 and t_1 .

CASE 2: There exists no cut set of size two which separates two vertices among s_1, s_2, t_1, t_2 .

In this case, for each cut set of size two we will replace the weak components modulo $\{u, v\}$ which do not contain s_1, s_2, t_1, t_2 with the edge (u, v) . The idea is that that edge will replace part of a path from s_1 to t_1 or part of a path from s_2 to t_2 .

$$\begin{aligned} 2DP(s_1, s_2, t_1, t_2) &\leftarrow 3StarMerge(s_1, s_2, s_2, s_2) \\ &\wedge 3StarMerge(s_1, t_1, t_1, t_1) \\ &\wedge 3StarMerge(s_1, t_2, t_2, t_2) \\ &\wedge 2DPt(s_1, s_2, t_1, t_2) \end{aligned}$$

$$\begin{aligned} ERASED(x, y) &\leftarrow CutSet2(u, v) \\ &\wedge [\neg SameStr(x, s_1, u, v) \vee \neg SameStr(x, s_2, u, v) \\ &\vee \neg SameStr(x, t_1, u, v) \vee \neg SameStr(x, t_2, u, v)] \end{aligned}$$

$$ADDED(x, y) \leftarrow CutSet2(x, y) \vee E(x, y)$$

$$E^t(x, y) \leftarrow ADDED(x, y) \wedge \neg ERASED(x, y)$$

(NB.: $2DPt(s_1, s_2, t_1, t_2) \leftarrow \mathbf{S-Datalog}(\neq)$ program for the 2DP on 3-connected graphs.)

It follows from cases 1 and 2 that

PROPOSITION 2.2. *If there exists an expression in Stratified Datalog(\neq) to define the problem of the two disjoint paths over 3-connected undirected graphs then there exists an expression in Stratified Datalog(\neq) for this problem over undirected graphs.*

We shall proceed with our analysis, showing that $2DP$ over undirected graphs is definable in $FO + LFP$. In order to do that we will use the well-known fact that $Stratified\ Datalog(\neq) \subseteq FO + LFP$.

CASE 3: \mathcal{G} is planar. But then:

- (i) The class of planar graphs is definable in $FO + LFP$ [6].
- (ii) $FO + LFP$ captures the complexity class \mathcal{P} over the 3-connected planar graphs [6].
- (iii) There exists a polynomial-time algorithm for the problem of the two disjoint paths over the class of 3-connected undirected planar graphs [8].

CASE 4: Suppose that there exist disjoint paths connecting s_1, t_1, s_2, t_2 to any other path of four vertices or less.

In this case, for each cut set of size three, say $\{u, v, z\}$, we replace the strong components modulo $\{u, v, z\}$ which do not contain s_1, s_2, t_1, t_2 with the edges $\{(u, v), (u, z), (v, z)\}$. Shiloach [12] has proved that a solution to the first graph may be obtained from a solution for the second graph. Moreover, if a graph satisfies Case 4 and is not planar then a solution for the $2DP$ is positive.

The query $SameStr'(x, y, u, v, z)$, which is valid if the vertices x and y are in the same strong component modulo $\{u, v, z\}$, and $CutSet3(x, y, z)$, which tells us that $\{x, y, z\}$ is a cut set of size three, are generalizations relatively easy of queries $SameStr\{x, y, z\}$ and $CutSet2$.

The aforementioned reduction is in fact definable in *Stratified Datalog* (\neq):

$$\begin{aligned}
 ERASED_{FC}(x, y) &\leftarrow CutSet3(u, v, z) \\
 &\wedge \neg SameStr'(x, s_1, u, v, z) \\
 &\wedge \neg SameStr'(x, s_2, u, v, z) \\
 &\wedge \neg SameStr'(x, t_1, u, v, z) \\
 &\wedge \neg SameStr'(x, t_2, u, v, z)
 \end{aligned}$$

$$\begin{aligned}
 ADDED_{FC}(x, y) &\leftarrow CutSet3(x, y, z) \\
 &\wedge \neg SameStr'(s_1, w, x, y, z) \\
 &\wedge \neg SameStr'(t_1, w, x, y, z) \\
 &\wedge \neg SameStr'(s_2, w, x, y, z) \\
 &\wedge \neg SameStr'(t_2, w, x, y, z)
 \end{aligned}$$

$$ADDED_{FC}(x, y) \leftarrow E^t(x, y)$$

$$E^{FC}(x, y) \leftarrow ADDED_{FC}(x, y) \wedge \neg ERASED_{FC}(x, y)$$

By noticing that *Stratified Datalog*(\neq) is contained in $FO + LFP$, the theorem follows from Proposition 2.2 and Cases 3 and 4. \square

3. Partial result about definability of 2DP in $\exists L_{\infty\omega}^\omega$

What comes next is a partial result concerning the definability of 2DP in the negation-free existential fragment of infinitary logic with a finite number of variables. The result which we obtain tells us what graphs may be used in the existential game with k pebbles to prove that 2DP is not definable in $\exists L_{\infty\omega}^\omega$, if this is the case. Though partial, this result is rather interesting since we manage to bound the classes of graphs which interest us for the construction of a possible proof that $2DP \notin \exists L_{\infty\omega}^\omega$ using games. Let us recall that the choice of adequate structures is one of the difficulties in the use of such tool.

If we manage to make a step forward in this respect, by proving that 2DP indeed is not definable in $\exists L_{\infty\omega}^\omega$, then we will answer negatively to a preservation problem in Finite Model Theory. We shall discuss this aspect at the end of the paper.

Let \mathcal{D} be the class of graphs $\mathcal{G} = (V_{\mathcal{G}}, E_{\mathcal{G}}, s_1, s_2, t_1, t_2)$, where \mathcal{G} consists of disjoint paths from s_1 to t_1 and from s_2 to t_2 , and assume that \mathcal{T} is the class of all graphs $\mathcal{H} = (V_{\mathcal{H}}, E_{\mathcal{H}}, s_1, s_2, t_1, t_2)$ such that \mathcal{H} is a 3-connected planar graph, and the vertices s_1, s_2, t_1 and t_2 appear in some boundary in this order.

THEOREM 3.1. *If $2DP$ is not definable in $\exists L_{\infty\omega}^{\omega}$ then the Duplicator has a strategy for the existential game with k pebbles over graphs \mathcal{G} and \mathcal{H} , where $\mathcal{G} \in \mathcal{D}$ and $\mathcal{H} \in \mathcal{T}$.*

PROOF. Suppose that the Duplicator has a strategy to win the existential game with k pebbles over the graphs $\mathcal{G}' = (V_{\mathcal{G}'}, E_{\mathcal{G}'}, s_1^{\mathcal{G}'}, s_2^{\mathcal{G}'}, t_1^{\mathcal{G}'}, t_2^{\mathcal{G}'})$ and $\mathcal{H}' = (V_{\mathcal{H}'}, E_{\mathcal{H}'}, s_1^{\mathcal{H}'}, s_2^{\mathcal{H}'}, t_1^{\mathcal{H}'}, t_2^{\mathcal{H}'})$ such that $\mathcal{G}' \in 2DP$ and $\mathcal{H}' \notin 2DP$. Assume that the graph \mathcal{G}' is in \mathcal{D} . We will show that the graph \mathcal{H} may be taken from \mathcal{T} .

Let \mathcal{G} be the graph obtained from \mathcal{G}' by adding the edges $\{s_1^{\mathcal{G}'}, s_2^{\mathcal{G}'}\}$ and $\{t_1^{\mathcal{G}'}, t_2^{\mathcal{G}'}\}$ and assume that \mathcal{H} is the graph obtained from \mathcal{H}' by adding the edges $\{s_1^{\mathcal{H}'}, s_2^{\mathcal{H}'}\}$ and $\{t_1^{\mathcal{H}'}, t_2^{\mathcal{H}'}\}$. Since the Duplicator has a strategy for the existential game with k pebbles over the graphs \mathcal{G}' and \mathcal{H}' then the Duplicator also has a strategy to win the existential game with k pebbles over the graphs \mathcal{G} and \mathcal{H} .

FACT 3.2. The vertices $s_1^{\mathcal{H}}, s_2^{\mathcal{H}}, t_1^{\mathcal{H}}$ and $t_2^{\mathcal{H}}$ are in the same block from \mathcal{H} .

PROOF. Let us suppose that this statement is not true and show that in this case the Spoiler has a strategy to win the game.

Suppose that $s_1^{\mathcal{H}}$ and $t_1^{\mathcal{H}}$ are in different blocks. Then these vertices may be in distinct components of \mathcal{H} , or there may exist an edge touching on both, or there exists a path between these vertices which goes through a cut vertex:

1. If the vertices $s_1^{\mathcal{H}}$ and $t_1^{\mathcal{H}}$ are in distinct components of \mathcal{H} , then obviously the Spoiler can win the game by walking from $s_1^{\mathcal{G}}$ to $t_1^{\mathcal{G}}$ in \mathcal{G} , since it is not possible for the Duplicator to describe a path from $s_1^{\mathcal{H}}$ to $t_1^{\mathcal{H}}$.
2. If $s_1^{\mathcal{H}}$ and $t_1^{\mathcal{H}}$ have an edge (but not another path) between them, then the Spoiler can win by walking from $s_1^{\mathcal{G}}$ to $t_2^{\mathcal{G}}$, and from $t_2^{\mathcal{G}}$ to $t_1^{\mathcal{G}}$ in the graph \mathcal{G} , visiting vertex $s_1^{\mathcal{G}}$ only once. For the Duplicator to keep a homomorphism, he must walk from $s_1^{\mathcal{H}}$ to $t_2^{\mathcal{H}}$ and then to $t_1^{\mathcal{H}}$ in \mathcal{H} , but then he would have to use again vertex $s_1^{\mathcal{H}}$ (since otherwise vertices $s_1^{\mathcal{H}}$ and $t_1^{\mathcal{H}}$ would be in the same block). Hence, the homomorphism is not one-to-one and the Spoiler wins the game.
3. If there exists a path from $s_1^{\mathcal{H}}$ to $t_1^{\mathcal{H}}$ which goes through a cut vertex, say x_H , then the Spoiler can win the game by proceeding in the following way: the Spoiler starts to make a walk from $s_1^{\mathcal{G}}$ to $t_1^{\mathcal{G}}$ in the

graph \mathcal{G} . The Duplicator is then forced to do the same in the graph \mathcal{H} . By doing this the Duplicator eventually puts a pebble over vertex x_H . At this moment, the Spoiler leaves the corresponding pebble over vertex x_G and starts to walk in a path from s_1^G to t_1^G which avoids x_G . The Duplicator needs to walk on a path from s_1^H to t_1^H , but in doing so he is forced to put a pebble over x_H , and the mapping will not be one-to-one. The Spoiler wins the game.

It follows that s_1^H and t_1^H are in the same block. Similar arguments may be given for the other pairs of $s_1^H, s_2^H, t_1^H, t_2^H$. Thus, the vertices s_1^H, s_2^H, t_1^H and t_2^H are in the same block. \square

FACT 3.3. The Spoiler can force the Duplicator to play inside the block of \mathcal{H} which contains vertices s_1^H, s_2^H, t_1^H , and t_2^H , i.e. any winning strategy of the Duplicator must have all of its moves in the particular component.

PROOF. Suppose the Duplicator, in response to a move by the Spoiler which has put a pebble over vertex y_G , put a pebble over a vertex y_H , which is separated from s_1^H, s_2^H, t_1^H and t_2^H by a cut vertex x_H . The Spoiler may then win the game proceeding in the manner described as follows. The Spoiler leaves the pebble over y_G and starts to go along a path from y_G to s_1^G . The Duplicator is forced to do the same and will eventually put a pebble over cut vertex x_H . At this moment, the Spoiler leaves the corresponding pebble over vertex x_G and starts a path from y_G to s_1^G which avoids vertex x_G . But it is not possible for the Duplicator to go along a path from y_H to s_1^H without going through x_H and, hence, the Spoiler wins the game. \square

It follows from 3.3 that we may assume that the graph \mathcal{H} is biconnected.

FACT 3.4. If the set $\{u, v\}$ is a cut set for \mathcal{H} and $\{s_1^H, s_2^H, t_1^H, t_2^H\} \cap \{u, v\} \neq \emptyset$ then the Spoiler may force the Duplicator to play in some strong component of \mathcal{H} modulo $\{u, v\}$, possibly extended with the edge $\{u, v\}$.

PROOF. We shall analyse a series of cases which depend upon the cut set $\{u, v\}$.

CASE 1: $|\{u, v\} \cap \{s_1^H, s_2^H, t_1^H, t_2^H\}| = 1$.

CASE 1.1: $\{s_1^H, v_H\}$ is a cut set which separates s_2^H from t_1^H and t_2^H .

Suppose the Spoiler starts walking along a path from s_2^G to t_1^G which avoids s_1^G . The Duplicator must do the same in the graph \mathcal{H} . But by doing so, he is forced to put a pebble over v_H . At this moment, the Spoiler leaves the corresponding pebble over vertex v_G . Since the Duplicator has a strategy for \mathcal{G} and \mathcal{H} then he must have a strategy for $(\mathcal{G}'', s_1^G, v_G, t_1^G, t_2^G)$ and $(\mathcal{H}'', s_1^H, v_H, t_1^H, t_2^H)$, where \mathcal{G}'' is the graph obtained from \mathcal{G} contracting the path from s_2^G to v_G and adding the edge $\{s_1^G, v_G\}$, and \mathcal{H}'' is the strong component of \mathcal{H} modulo $\{s_1^H, v_H\}$ which contains t_1^H , augmented with the edge $\{s_1^H, v_H\}$.

Similar Cases: $\{s_2^H, v_H\}$ is a cut set which separates s_1^H from t_1^H and t_2^H ; $\{t_1^H, v_H\}$ is a cut set which separates t_2^H from s_1^H and s_2^H ; $\{t_2^H, v_H\}$ is a cut set which separates t_1^H from s_1^H and s_2^H ;

CASE 1.2: $\{s_1^H, v_H\}$ is a cut set and s_2^H, t_1^H, t_2^H are in the same strong component of \mathcal{H} modulo $\{s_1^H, v_H\}$.

Suppose the Duplicator has put a pebble over vertex x_H , in response to a move by the Spoiler which has put the corresponding pebble over vertex x_G , which is in a strong component modulo $\{s_1, v\}$ of \mathcal{H} that does not contain s_2^H . If x_G were between s_2^G and t_2^G then \mathcal{H} would not satisfy the query $2StarAvoid(x_H, s_2^H, t_2^H, s_1^H)$. But \mathcal{G} satisfies $2StarAvoid(x_G, s_2^G, t_2^G, s_1^G)$. Since the query $2StarAvoid$ is expressible in $Datalog(\neq)$, then the Spoiler would win the game, contradicting the hypothesis of the existence of a strategy for the Duplicator. It follows that x_G must be between s_1^G and t_1^G . The Spoiler then starts walking along a path from x_G to t_1^G which avoids s_1^G . The Duplicator is forced to do the same in the graph \mathcal{H} and will, eventually, put a pebble over vertex v_H . The Spoiler then leaves the corresponding pebble over vertex v_G between s_1^G and t_1^G . Since the Duplicator has a strategy for \mathcal{G} and \mathcal{H} then he has a strategy for $(\mathcal{G}'', s_1^G, s_2^G, t_1^G, t_2^G)$ and $(\mathcal{H}'', s_1^H, s_2^H, t_1^H, t_2^H)$, where \mathcal{G}'' is the graph obtained from \mathcal{G} by replacing the path from s_1^G to v_G with the edge $\{s_1^G, v_G\}$ and \mathcal{H}'' is the strong component modulo $\{s_1, v\}$ of \mathcal{H} which contains s_2 augmented with the edge $\{s_1^H, v_H\}$.

CASE 1.3: The following cases are not possible: $\{s_1^H, v_H\}$ is a cut set which separates t_2^H from t_1^H and s_2^H ; $\{s_2^H, v_H\}$ is a cut set which separates t_1^H from s_1^H and t_2^H ; $\{t_1^H, v_H\}$ is a cut set which separates s_2^H from t_2^H and s_1^H ; $\{t_2^H, v_H\}$ is a cut set which separates t_1^H from s_1^H and s_2^H ; $\{s_1^H, v_H\}$ is a cut set which separates t_1^H from s_2^H and t_2^H ; $\{t_1^H, v_H\}$ is a cut set which

separates s_1^H from s_2^H and t_2^H ; $\{s_2^H, v_H\}$ is a cut set which separates t_2^H from s_1^H and t_1^H ; $\{t_2^H, v_H\}$ is a cut set which separates s_2^H from s_1^H and t_1^H .

since $\{s_1^H, s_2^H\}$ and $\{t_1^H, t_2^H\}$ are edges of \mathcal{H} .

CASE 2: $|\{u, v\} \cap \{s_1^H, s_2^H, t_1^H, t_2^H\}| = 2$.

CASE 2.1: $\{s_1^H, t_1^H\}$ is a cut set which separates s_2^H from t_2^H .

In this case, the Spoiler would win the game by going along a path from s_2^G to t_2^G which avoids s_1^G and t_1^G , since for the Duplicator to do the same he would have to put a pebble over s_1^H or t_1^H . But, by hypothesis, the Duplicator has a strategy to win the game.

CASE 2.2: $\{s_1^H, t_1^H\}$ is a cut set, s_2^H and t_2^H are in the same strong component modulo $\{s_1^H, t_1^H\}$.

This is not possible either since \mathcal{H} is not in $2DP$.

Similar Cases: $\{s_2^H, t_2^H\}$ is a cut set, s_1^H and t_1^H are in the same strong component modulo $\{s_2^H, t_2^H\}$; $\{s_1^H, t_2^H\}$ is a cut set which separates t_1^H from s_2^H ; $\{s_2^H, t_1^H\}$ is a cut set which separates s_1^H from t_2^H .

CASE 2.3: $\{s_1^H, s_2^H\}$ is a cut set which separates t_1^H from t_2^H .

Not possible either since $\{t_1^H, t_2^H\}$ is an edge of \mathcal{H} .

Similar Case: $\{t_1^H, t_2^H\}$ is a cut set which separates s_1^H from s_2^H .

CASE 2.4: $\{s_1^H, s_2^H\}$ is a cut set, t_1^H and t_2^H are in the same strong component modulo $\{s_1^H, s_2^H\}$.

Suppose that the Duplicator, in response to a move by the Spoiler which has put a pebble on x_G , puts a pebble over vertex x_H which is in some strong component modulo $\{s_1^H, s_2^H\}$ of \mathcal{H} that does not contain t_1^H and t_2^H . The graph \mathcal{H} does not satisfy the query $PathAv2(x_H, t_1^H, s_1^H, s_2^H) \vee PathAv2(x_H, t_2^H, s_1^H, s_2^H)$. But the graph \mathcal{G} satisfies the query $PathAv2(x_G, t_1^G, s_1^G, s_2^G) \vee PathAv2(x_G, t_2^G, s_1^G, s_2^G)$, contradicting the hypothesis that the Duplicator has a strategy for the graphs \mathcal{G} and \mathcal{H} . Thus, from the hypothesis that the Duplicator has a strategy for \mathcal{G} and \mathcal{H} it follows that he must have a strategy for \mathcal{G} and the strong component modulo $\{s_1^H, s_2^H\}$ of \mathcal{H} which contains t_1^H and t_2^H .

Similar Cases: $\{t_1^H, t_2^H\}$ is a cut set, s_1^H and s_2^H are in the same strong component modulo $\{t_1^H, t_2^H\}$; $\{s_1^H, t_2^H\}$ is a cut set, t_1^H and s_2^H are in the same strong component modulo $\{s_1^H, t_2^H\}$; $\{s_2^H, t_1^H\}$ is a cut set, s_1^H and t_2^H are in the same strong component modulo $\{s_2^H, t_1^H\}$. \square

FACT 3.5. Let $\mathcal{H}' = (V', E')$ be a weak component modulo $\{u_H, v_H\}$ of \mathcal{H} and $V' \cap \{s_1^H, s_2^H, t_1^H, t_2^H\} = \emptyset$. If the Duplicator has a strategy on the pair \mathcal{G}, \mathcal{H} then he also has one on the pair $\mathcal{G}, \mathcal{H}'$.

PROOF. Suppose that $V' \cap \{s_1, s_2, t_1, t_2\} = \emptyset$ and that the Duplicator has put a pebble over a vertex x_H in \mathcal{H}' . Then the Spoiler can force the Duplicator to put pebbles over the vertices u_H and v_H : if x_G , the vertex which corresponds to x_H is between s_1^G and t_1^G then he starts to go along a path from x_G to s_1^G which does not go through t_1^G . The Duplicator must proceed similarly, but in doing so he will eventually put a pebble over u_H or v_H . At this moment, the Spoiler stops, and the pebbles are left over u_H and u_G (v_H and v_G). Assume that the pebble was put over u_G . In this case, the Spoiler starts to walk from x_G to t_1^G without going through s_1^G or u_G . The Duplicator, in doing the same, will eventually put a pebble over v_H . At this moment, the Spoiler leaves the corresponding pebble over v_G . Since the Duplicator has a strategy for \mathcal{G} and \mathcal{H} then he has a strategy for $(\mathcal{G}'', s_1^G, s_2^G, t_1^G, t_2^G)$ and $(\mathcal{H}'', s_1^H, s_2^H, t_1^H, t_2^H)$, where \mathcal{G}'' is the graph obtained from \mathcal{G} by replacing the path from u_G to v_G (which goes through x_G) with the edge $\{u_G, v_G\}$ and \mathcal{H}'' is the subgraph obtained from \mathcal{G} by removing subgraph \mathcal{H}' and adding the edge $\{u_H, v_H\}$. \square

FACT 3.6. There exist no cut set $\{u, v\}$ which separates s_1^H from s_2^H or that separates t_1^H from t_2^H .

PROOF. $\{s_1^H, s_2^H\}$ and $\{t_1^H, t_2^H\}$ are edges of the graph \mathcal{H} . \square

FACT 3.7. If $\{u_H, v_H\}$ is a cut set of \mathcal{H} which separates s_1^H and s_2^H from t_1^H and t_2^H then the Spoiler can force the Duplicator to play over pairs of strong components modulo $\{u_H, v_H\}$ of \mathcal{G} and \mathcal{H} .

PROOF. Suppose there exists a cut set $\{u_H, v_H\}$ which separates s_1^H and s_2^H from t_1^H and t_2^H . The Spoiler starts walking along a path from s_1^G to t_1^G . In doing the same the Duplicator will eventually put a pebble over vertex u_H or v_H . Assume that a pebble was put over vertex u_H . The Spoiler then leaves the corresponding pebble over the vertex u_G and starts to go along a path from s_2^G to t_2^G which avoids u_G . The Duplicator is forced to do the same in \mathcal{H} and will eventually be forced to put a pebble over vertex v_H . At this moment, the Spoiler stops, and this pebble remains over v_H . Since the Duplicator has a strategy to win

the existential game with k pebbles over \mathcal{G} and \mathcal{H} then he must have a strategy for the game over $(\mathcal{G}', s_1^G, s_2^G, u_G, v_G)$ and $(\mathcal{H}', s_1^H, s_2^H, u_H, v_H)$ and over $(\mathcal{G}'', u_G, v_G, t_1^G, t_2^G)$ and $(\mathcal{H}'', u_H, v_H, t_1^H, t_2^H)$ or else for the game over $(\mathcal{G}', s_1^G, s_2^G, v_G, u_G)$ and $(\mathcal{H}', s_1^H, s_2^H, v_H, u_H)$ and over $(\mathcal{G}'', v_G, u_G, t_1^G, t_2^G)$ and $(\mathcal{H}'', v_H, u_H, t_1^H, t_2^H)$, where \mathcal{G}' , \mathcal{G}'' (\mathcal{H}' , \mathcal{H}'') are strong components modulo $\{u_G, v_G\}$ of \mathcal{G} (strong components modulo $\{u_H, v_H\}$ of \mathcal{H}) which contain s_1^G , s_2^G and t_1^G , t_2^G (s_1^H , s_2^H and t_1^H , t_2^H), respectively, augmented with the edge $\{u_G, v_G\}$ ($\{u_H, v_H\}$).

It follows from facts 3.4, 3.5, 3.6, 3.7 that we may assume that \mathcal{H} is 3-connected. \square

FACT 3.8. If \mathcal{H} is not planar then we can modify the graph \mathcal{H} in such a way as to obtain a planar graph or a 3-connected graph where there exist four disjoint paths connecting $s_1^H, s_2^H, t_1^H, t_2^H$ to any set of four vertices or less such that there exist disjoint paths from s_1^H to t_1^H and from s_2^H to t_2^H if, and only if, there exist such disjoint paths in the graph obtained.

PROOF. This reduction is presented in [12]. But in a non-planar 3-connected graph, where there exist four disjoint paths connecting $s_1^H, s_2^H, t_1^H, t_2^H$ to any set of four vertices or less there exist disjoint paths from s_1^H to t_1^H and from s_2^H to t_2^H . Since \mathcal{H} does not have disjoint paths from s_1^H to t_1^H and from s_2^H to t_2^H then the graph obtained is planar. \square

FACT 3.9. If \mathcal{H} is planar then the vertices s_1^H, s_2^H, t_1^H and t_2^H appear in a facial cycle in this order.

PROOF. Perl & Shiloach [8] have shown that a 3-connected planar graph has disjoint paths from s_1 to t_1 and from s_2 to t_2 if, and only if, these vertices do not appear in a facial cycle in the order s_1, s_2, t_1, t_2 . \square

Thus, the graph \mathcal{H} may be taken to be in \mathcal{T} . \square

4. Conclusion

In this paper we have proven:

1. $2DP$ is definable in $FO + LFP$; and
2. a partial result about the definability of $2DP$ in $\exists L_{\infty\omega}^\omega$ which tells us what graphs may be used in existential game with k pebbles to assert it negatively.

With respect to the latter, we would like to make some additional considerations. In order to show that $2DP \in \exists L_{\infty\omega}^\omega$ we have to show that the logic $\exists L_{\infty\omega}^\omega$ is sufficiently expressive to separate graphs which are in \mathcal{D} from graphs which are in \mathcal{T} , that is, that there exists a sentence φ in this logic which is satisfied by some graph of \mathcal{D} and is not satisfied by some graph of \mathcal{T} . Intuitively, a possible sentence φ would be a sentence which were sensitive to the order in which the vertices appear in a facial cycle. One can also exhibit a sentence $\psi(x, y, z, v)$ such that for each planar graph \mathcal{G} and $a, b, c, e \in G$ one has $\mathcal{G} \models \psi(a, b, c, e)$ if, and only if, abc is an angle of \mathcal{G} and e is contained in the facial cycle determined by abc , and this would in fact be sufficient to establish a strategy for the Spoiler to win the existential game with k pebbles over graphs \mathcal{G} and \mathcal{H} , with $\mathcal{G} \in \mathcal{D}$ and $\mathcal{H} \in \mathcal{T}$. M. Grohe [6] has shown that there exists such a sentence in the logic $FO + LFP$. Nevertheless, everything leads us to believe that this sentence is not equivalent to any sentence in $\exists L_{\infty\omega}^\omega$.

The negative result for the question $2DP \in \exists L_{\infty\omega}^\omega$ is particularly interesting since with this we would obtain a counterexample for the following question:

$$(FO + LFP) \cap HOM \subseteq \exists L_{\infty\omega}^\omega,$$

where HOM is the class consisting of all sets of finite structures which are closed under homomorphism.

It is interesting to note that Rosen & Weinstein ([9], [10]) have answered various questions concerning preservation over finite structures and raised some questions rather close to the one we mentioned in the previous paragraph, such as for example:

1. $FO \cap HOM \subseteq Datalog$?
2. $FO \cap HOM \subseteq \exists L_{\infty\omega}^\omega(+)$?

where $\exists L_{\infty\omega}^\omega(+)$ denotes the positive existential fragment of the logic $\exists L_{\infty\omega}^\omega$.

Question (1) is of particular interest since Ajtai & Gurevich [1] have shown that $FO \cap Datalog = \exists FO(+)$, where $\exists FO(+)$ denotes the positive existential fragment of first order logic. Thus, a positive answer to the

first question would imply the validity of the Preservation Theorem under homomorphism when only finite structures are considered, i.e.,³

$$FO \cap HOM = \exists FO(+).$$

References

- [1] M. Ajtai and Y. Gurevich, *Monotone versus positive*, **Journal of the ACM** 34 (1987), pp. 1004–1015.
- [2] A. Atserias, A. Dawar and Ph. G. Kolaitis, *On Preservation under Homomorphisms and Unions of Conjunctive Queries*, [in:] **Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of Database Systems (PODS 2004)**, June 14–16, 2004, Paris, France, pp. 319–329.
- [3] I. Barland, **Expressing Optimization Problems as Integer Programs, and Undirected Path Problems: A Descriptive Complexity Approach**, PhD thesis, Rice University, 1999.
- [4] H. G. Benatti, **Complexidade Descritiva de Problemas em Grafos**, (“*Descriptive Complexity of Problems on Graphs*”), PhD Thesis, Informática, Universidade Federal de Pernambuco, Recife, Brazil, March 2000. (In Portuguese.)
- [5] H.-D. Ebbinghaus and J. Flum, **Finite Model Theory**, Springer-Verlag 1995.
- [6] M. Grohe, *Fixed-point logics on planar graphs*, [in:] **Proceedings of the 13th Annual IEEE Symposium on Logic in Computer Science**, pp. 6–15, 1998.
- [7] Ph. G. Kolaitis and M. Y. Vardi, *On the expressive power of Datalog: tools and a case study*, **Journal of Computer and System Sciences** 51(1) August (1985), pp. 110–134. Special Issue: Selections from

³At the time we obtained our results we were not aware of the works recently published, in particular, V. Vianu (personal communication) has informed us that: (1) a recent paper by Albert Atserias, Anuj Dawar and Phokion Kolaitis, i.e. [2], discusses various relevant issues concerning preservation under homomorphisms and unions of conjunctive queries; (2) a proof of the preservation theorem for FO has been proved by Benjamin Rossman in [11].

Ninth Annual ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems (PODS), Nashville, TN, USA, 2-4 April 1990.

[8] Y. Perl and Y. Shiloach, *Finding two disjoint paths between two pairs of vertices in a graph*, **Journal of the ACM** 25 (1978), pp. 1–9.

[9] E. Rosen, **Finite model theory and finite variable logic**, PhD thesis, University of Pennsylvania, 1995.

[10] E. Rosen and S. Weinstein, *Preservation theorems in finite model theory*, [in:] **Logic and Computational Complexity**, volume 960 of Lecture Notes in Computer Science, pp. 480–502, 1995.

[11] B. Rossman, *Existential Positive Types and Preservation under Homomorphisms*, **Proc. 20th IEEE Symp. on Logic in Computer Science**, pp. 467–476, 2005.

[12] Y. Shiloach, *A polynomial solution to the undirected two paths problem*, **Journal of the ACM** 27 (1980), pp. 445–456.

[13] M. Yannakakis, F. N. Afrati, S. S. Cosmadakis, *On Datalog vs. polynomial time*, **J. Comput. Syst. Sci.** 51 (1995), pp. 177–196.

Dept. of Mathematics
Univ. Estadual de Feira de Santana, Bahia, Brazil
e-mail: benatti@uefs.br

Centro de Informática
Univ. Federal de Pernambuco, Recife, Brazil
e-mail: ruy@cin.ufpe.br.