Equivalence in finite-variable logics is complete for polynomial time

Martin Grohe*
Institut für mathematische Logik
Albert-Ludwigs-Universität Freiburg
Eckerstr. 1, 79104 Freiburg, Germany
grohe@sun1.mathematik.uni-freiburg.de

Abstract

How difficult is it to decide whether two finite structures can be distinguished in a given logic? For first order logic, this question is equivalent to the graph isomorphism problem with its well-known complexity theoretic difficulties. Somewhat surprisingly, the situation is much clearer when considering the fragments L^k *of first-order logic whose formulae contain at most k (free or bound) variables (for some* $k \geq 1$*). We show that for each* $k \geq 2$*, equivalence in the k-variable logic* L^k *is complete for polynomial time under quantifier-free reductions (a weak form of* NC_0 *reductions). Moreover, we show that the same completeness result holds for the powerful extension* C^k *of* L^k *with counting quantifiers (for every* $k \geq 2$*).*

1. Introduction

Two structures are called *equivalent* in a logic L if they satisfy the same sentences of L. A basic observation is that two finite structures are equivalent in first-order logic if, and only if, they are isomorphic. We consider equivalence in the fragments L^k of first-order logic whose formulae contain at most k free or bound variables (for some $k \geq 1$). Our main result asserts that, for every $k \geq 2$, deciding equivalence in L^k is complete for polynomial time under quantifier-free reductions (which constitute a weak form of NC_0 reductions). The result can be generalized to the extensions C^k of L^k by counting quantifiers.

Since graph isomorphism corresponds to equivalence in first-order logic, the parametrization of first-order formulae by the number of variables can be seen as an approximation of isomorphism. Based on this idea, Immerman and Lander [12] suggested a "first-order approach to graph canonization". They found that on many important classes of graphs canonization can be described in L^k or C^k, and thus that on these graphs equivalence in the logics coincides with isomorphism. Moreover, for each of the logics L^k and C^k they gave PTIME-algorithms deciding equivalence and producing a canonical labelling for graphs that can be characterized in the logic. Our results give a lower bound for such algorithms. Furthermore, our studies can be seen as a logical approach to the open question of whether graph isomorphism is hard for polynomial time.

Let us say that two elements of a structure have the same L^k*-type* if they satisfy the same L^k-formulae with one free variable. We also show that the question of whether two elements of a finite structure have the same L^k-type is complete for polynomial time under quantifier-free reductions (for all $k \geq 2$). There is a striking formal similarity between the question of whether two vertices of a graph have the same L^2-type and the question of whether they are *strongly bisimilar*. And indeed, we are going to see that *on undirected graphs* the L^2-type and the bisimilarity question are equivalent with respect to quantifier-free reductions. Thus, as a corollary, we obtain that bisimilarity is PTIME-complete even on undirected graphs (for directed graphs this is due to Balcázar, Gabarró, and Sántha [2]).

Logics with finitely many variables have always been playing an important role in finite model theory (see for example [5, 8, 11, 14, 15, 16]). Our result underlines once more the deep connection between these logics and computational complexity. One of the most important results in finite model theory is the Abiteboul-Vianu Theorem [1]. It states that the question of whether PTIME equals PSPACE is equivalent to the purely logical question of whether *least fixed point logic* and *partial fixed-point logic* have the same expressive power. The proof is based on the fact that equivalence in L^k can be defined in least fixed-point logic. An easy corollary of our theorem is that any logic able to de-

*This research was done while the author visited the University of California in Santa Cruz; supported by a NATO Scientific Council grant administrated by DAAD.

0272-5428/96 $05.00 © 1996 IEEE

fine equivalence in L^k can already define all PTIME-queries on ordered structures. This offers an explanation why all attempts to get results analogous to the Abiteboul-Vianu Theorem for complexity classes below PTIME with similar methods failed.

2. Preliminaries

A *vocabulary* is a set σ containing finitely many relation and constant symbols. A *σ-structure* $\mathfrak{A}$ consists of a set A, called the *universe* of $\mathfrak{A}$, an interpretation $R^{\mathfrak{A}} \subseteq A^r$ for each r-ary relation symbol $R \in \sigma$, and an interpretation $c^{\mathfrak{A}} \in A$ of each constant symbol $c \in \sigma$. We restrict our attention to structures whose universe is finite. A *Boolean query* is a class of structures of the same vocabulary that is closed under isomorphism. For example, graphs are structures $\mathfrak{A} = (A, E^{\mathfrak{A}})$ where E is a binary relation symbol, and coloured graphs are structures $\mathfrak{A} = (A, E^{\mathfrak{A}}, C_1^{\mathfrak{A}}, \dots, C_m^{\mathfrak{A}})$ where E is a binary and $C_1, \dots, C_m$ are unary relation symbols. A typical Boolean query is the class of all connected graphs.

An important class of structures is the class of *ordered* structures. Possibly among other symbols, the vocabulary of an ordered structure contains the binary relation symbols $<$, S and the constant symbols $\underline{0}$, $\underline{e}$. These symbols are interpreted as a linear order of the universe, the corresponding successor relation, the first, and the last element, respectively. As usually in descriptive complexity theory, computational problems are considered as Boolean queries on ordered structures (that is, classes of ordered structures of the same vocabulary that are closed under isomorphism). This is possible because there is a canonical way to associate a language $L_{\mathcal{C}} \subseteq \{0,1\}^*$ with each Boolean query $\mathcal{C}$ on ordered structures. Thus, for example, PTIME is the collection of all Boolean queries $\mathcal{C}$ on ordered structures such that $L_{\mathcal{C}} \in$ PTIME in the usual sense. There is no such canonical translation into languages for Boolean queries $\mathcal{C}$ on arbitrary structures, but with each Boolean query $\mathcal{C}$ on arbitrary structures, we can associate the query $\mathcal{C}^<$ consisting of all ordered $\sigma \cup \{<, S, \underline{0}, \underline{e}\}$-structures whose σ-reduct (that is, the result of forgetting the interpretations of symbols not in σ) is in $\mathcal{C}$. Then we say that $\mathcal{C}$ is *computable in* PTIME if $\mathcal{C}^< \in$ PTIME.

Atomic formulae are of the form $Rt_1 \dots t_r$ or $t_1 = t_2$, where R is an r-ary relation symbol and the t_i are *terms*, that is, constant symbols or *variables*. *Quantifier-free* formulae are Boolean combinations of atomic formulae. The class of *first-order formulae* is the result of closing the atomic formulae under Boolean combinations and existential and universal quantification. The semantics of *first-order logic* FO is defined in the usual way. For example, the sentence $\forall x \forall y \forall z \neg(Exy \wedge Exz \wedge Eyz)$ says that a graph is triangle free.

Quantifier-free reductions

All our results are stated in terms of the very low level quantifier-free reductions. As a matter of fact, quantifier-free reductions are NC_0-reductions with strong uniformity conditions. Thus the reader feeling uncomfortable with the concept should always have uniform NC_0-reductions in mind.

Let $\mathfrak{A}$ be a σ-structure. Observe that a first-order formula $\varphi(\bar{x}_1, \dots, \bar{x}_r)$, where $\bar{x}_1, \dots, \bar{x}_r$ are k-tuples of free variables, defines an r-ary relation on the kth power A^k of the universe of $\mathfrak{A}$. Thus, for any vocabulary τ we can associate a τ-structure with $\mathfrak{A}$, taking several such formulae φ (one defining the universe and one for each relation and constant symbol in τ, with the additional requirement that the formulae defining the constants are satisfied by a unique tuple of the universe). Such a mapping Φ from σ to τ-structures is called a *first-order interpretation of τ in σ*. It is called a *first-order reduction* from a Boolean query $\mathcal{C}$ of vocabulary σ to a Boolean query $\mathcal{D}$ of vocabulary τ if for each σ-structure $\mathfrak{A}$ we have: $\mathfrak{A}$ is in $\mathcal{C}$ if, and only if, its image $\Phi(\mathfrak{A})$ is in $\mathcal{D}$. A *quantifier-free reduction* is a first-order reduction whose defining formulae are quantifier-free. We write $\mathcal{C} \leq_{\text{qf}} \mathcal{D}$ if there exists a quantifier-free reduction from $\mathcal{C}$ to $\mathcal{D}$ and $\mathcal{C} \approx_{\text{qf}} \mathcal{D}$ if $\mathcal{C} \leq_{\text{qf}} \mathcal{D}$ and $\mathcal{D} \leq_{\text{qf}} \mathcal{C}$. A well-known fact that we frequently use, without explicitly mentioning it, is that the relation $\leq_{\text{qf}}$ is transitive.

We can now define *hardness* and *completeness* under quantifier-free reductions in the usual way (see, for example, [10]). In particular, a Boolean query $\mathcal{C}$ is *hard for* PTIME *under quantifier-free reductions* if for every Boolean query $\mathcal{D} \in$ PTIME (on ordered structures, recall that we consider PTIME as a collection of queries on ordered structures) we have $\mathcal{D} \leq_{\text{qf}} \mathcal{C}$. $\mathcal{C}$ is *complete for* PTIME *under quantifier-free reductions* if in addition it is computable in PTIME.

The monotone circuit value problem

The "canonical" PTIME-complete problem is the *monotone circuit value problem*, asking for the value a monotone circuit computes on a given input. As an immediate consequence of the PTIME-completeness of AGAP under quantifier-free reductions [10], one gets that it is complete for PTIME under quantifier-free reductions [18].

It is also known that the circuit value problem remains PTIME-complete (under, say, LOGSPACE reductions) when restricted to circuits of fan-in 2. As an easy exercise in quantifier-free reductions one can proof that this already holds with respect to quantifier-free reductions. Since our proofs are based on this fact, let us state it formally: We encode *monotone circuits of fan-in 2* as $\{E_M, E_F, \sqcap, \sqcup, O, Z, c\}$-structures $\mathfrak{C}$ where $(C, E_M^{\mathfrak{C}} \cup E_F^{\mathfrak{C}})$ is a directed acyclic graph in which each node has either exactly one E_M and one E_F-predecessor or no predecessor at all.

Furthermore, $\sqcup^{\mathfrak{C}}$ and $\sqcap^{\mathfrak{C}}$ are unary relations on the nodes of fan-in 2 representing OR and AND gates, respectively. $O^{\mathfrak{C}}$ and $Z^{\mathfrak{C}}$ are unary relations on the nodes of fan-in 0 representing input nodes with value 1 and 0, respectively, and $c^{\mathfrak{A}}$ represents the output node. The value computed by the circuit at some node $b \in C$ is denoted by $\text{VAL}_{\mathfrak{C}}(b) \in \{0,1\}$. We let MC_2 denote the class of all circuits $\mathfrak{C}$ of fan-in 2 with $\text{VAL}_{\mathfrak{C}}(c^{\mathfrak{C}}) = 1$.

Lemma 2.1 MC_2 *is complete for polynomial time under quantifier-free reductions.*

3. Finite variable logics

Recall that L^k denotes the fragment of first-order logic whose formulae contain at most k (free or bound) variables. We write $\mathfrak{A} \equiv_{\text{L}^k} \mathfrak{B}$ if two structures $\mathfrak{A}$ and $\mathfrak{B}$ of the same vocabulary are equivalent in L^k and $a \equiv^{\mathfrak{A}}_{\text{L}^k} b$ if two elements $a, b \in A$ have the same L^k*-type*. We are interested in the following two queries: the L^k*-equivalence query* $\text{EQUIVALENCE}_k = \{(\mathfrak{A}, \mathfrak{B}) \mid \mathfrak{A}, \mathfrak{B}$ directed graphs, $\mathfrak{A} \equiv_{\text{L}^k} \mathfrak{B}\}$ and the L^k*-type query* $\text{TYPE}_k = \{(\mathfrak{A}, a, b) \mid \mathfrak{A}$ directed graph, $a \equiv^{\mathfrak{A}}_{\text{L}^k} b\}$. To match our formal definitions, we have to encode the pairs of structures in EQUIVALENCE_k into a single structure, which can easily be done by well-known techniques.

The choice of directed graphs as our basic structures is somewhat arbitrary. We could use any sufficiently rich class of structures instead. For example, it is not hard to see that for any vocabulary σ containing at least one binary relation symbol the L^k-equivalence and L^k-type queries on the class of all σ-structures have the same complexity as the respective queries on directed graphs. (The easiest way to see this is to note that the queries are all in PTIME (due to [12], cf. Fact 3.4) and then apply Theorem 3.5.) For $k \geq 3$, undirected loop-free graphs also suffice. Only for $k = 2$ one has to be careful since there are only eight L^2-equivalence classes of undirected loop-free graphs which are quite easy to determine. If we allow loops or add one colour, undirected graphs are also sufficient for $k = 2$.

Our queries have a nice combinatorial characterization in terms of the following game:

Definition 3.1 Let $k \geq 1$. The *k-pebble game* is played by a *spoiler* and a *duplicator* on a pair $\mathfrak{A}, \mathfrak{B}$ of structures of the same vocabulary σ.

A *position* of the game is a subset of $A \times B$ of size at most k. The game consists of a sequence of positions, starting with an *initial position*. In each position P the spoiler may either remove a pair from P or, if $|P| \leq k-1$, choose one of the following moves: In an $\exists$*-move* he selects an $a \in A$. The duplicator answers by selecting a $b \in B$, and the new position is $P \cup \{ab\}$. In a $\forall$*-move* he selects a $b \in B$, the duplicator answers by selecting an $a \in A$, and again the new position is $P \cup \{ab\}$.

The duplicator *wins* the game if for any position P that occurs, $P \cup \{c^{\mathfrak{A}}c^{\mathfrak{B}} \mid c \in \sigma \text{ constant symbol}\}$ is a partial isomorphism between $\mathfrak{A}$ and $\mathfrak{B}$ (that is, an isomorphism whose domain is the substructure of $\mathfrak{A}$ with universe $\{a \in A \mid \exists b \in B : ab \in P\} \cup \{c^{\mathfrak{A}} \mid c \in \sigma\}$).

Fact 3.2 *[3, 8] Let $k \geq 1$. Moreover, let $\mathfrak{A}$ and $\mathfrak{B}$ be structures of the same vocabulary and $a, b \in A$. Then*
(i) $\mathfrak{A} \equiv_{\text{L}^k} \mathfrak{B}$ if, and only if, the duplicator has a winning strategy in the k-pebble game on $\mathfrak{A}, \mathfrak{B}$ with initial position $\emptyset$.
(ii) $a \equiv^{\mathfrak{A}}_{\text{L}^k} b$ if, and only if, the duplicator has a winning strategy in the k-pebble game on $\mathfrak{A}, \mathfrak{A}$ with initial position $\{ab\}$.

Remark 3.3 The game actually describes equivalence in the infinitary logic $\text{L}^k_{\infty\omega}$ which is defined in the same way as L^k, but allowing infinitary disjunctions an conjunctions. However, on finite structures equivalence in the two logics coincides.

Given the game characterization of L^k-equivalence and L^k-types, the upper bound in our completeness result, due to Immerman and Lander [12], is not too hard to get:

Fact 3.4 *[12] For all $k \geq 1$, TYPE_k and EQUIVALENCE_k are computable in* PTIME.

Our following main theorem is proved in Section 5.

Theorem 3.5 *For all $k \geq 2$, we have*

$$\text{MC}_2 \leq_{\text{qf}} \text{TYPE}_k \quad \textit{and} \quad \text{MC}_2 \leq_{\text{qf}} \text{EQUIVALENCE}_k.$$

Hence TYPE_k *and* EQUIVALENCE_k *are complete for polynomial time under quantifier-free reductions.*

4. Bisimilarity

In our context, the easiest way to introduce bisimilarity is via a game. A *labelled transition system* is a structure whose vocabulary only contains binary relation symbols. The *bisimilarity game* is played by a spoiler and a duplicator on a pair $\mathfrak{A}, \mathfrak{B}$ of labelled transition systems in the same way as the 2-pebble game with the following restricted versions of $\exists$ and $\forall$ moves: Let $P = \{ab\}$ be the current position. In a *forward* $\exists$*-move* the spoiler picks an $a' \in A$ such that Eaa' for some E in the vocabulary of $\mathfrak{A}, \mathfrak{B}$. The duplicator answers with some $b' \in B$, and the new position is $\{ab, a'b'\}$. A *forward* $\forall$*-move* is defined analogously. The winning condition for the duplicator remains the same as in the pebble game.

We say that $a \in A$, $b \in B$ are *(strongly) bismilar* ($a \equiv_\beta b$) if the duplicator wins the bisimilarity game on $\mathfrak{A}, \mathfrak{B}$ with initial position $\{ab\}$. It is not hard to see that this coincides with the usual definition of strong bisimilarity (see, for example, [2]).

As for the L^k-types, we define the *bisimilarity query* to be BISIMILARITY $= \{(\mathfrak{A}, a, b) \mid \mathfrak{A}$ directed graph, $a \equiv_\beta b\}$. Balcázar, Gabarró, and Sántha [2] showed that this query is PTIME-complete under NC-reductions. Below we give a simple proof that actually refines this result by showing completeness under quantifier-free reductions. Before we do so, let us consider a consequence of our theorem for bisimilarity. Let U-BISIMILARITY $= \{(\mathfrak{A}, a, b) \mid \mathfrak{A}$ undirected graph (that may contain loops), $a \equiv_\beta b\}$. (Instead of undirected graphs with loops we could have equivalently taken coloured undirected graphs, or transition systems with two undirected binary relations). In the full paper we show the following; this is actually not hard:

Proposition 4.1 TYPE$_2$ $\approx_{\mathrm{qf}}$ U-BISIMILARITY.

Corollary 4.2 *Bisimilarity on undirected transition systems is complete for polynomial time under quantifier-free reductions.*

The following proof of Proposition 4.5 should also be seen as a first, easy step towards the proof of our main theorem. Let us start by defining a simple concept, which is quite important in our proofs:

Definition 4.3 Let $\mathfrak{A}, \mathfrak{B}$ be structures of the same vocabulary and let P, Q be two positions of the k-pebble game or the bisimilarity game on $\mathfrak{A}, \mathfrak{B}$. We say that the spoiler can *reach* Q from P if he has a strategy for the game with initial position P such that he either wins the game or position Q eventually occurs.

Conversely, we say that duplicator can *avoid* Q from P if she has a winning strategy for the game with initial position P in which Q never occurs.

Rather than just working with directed graphs in our reductions, we colour them. It is easy to get rid of the colours afterwards by reducing the coloured to the colour-free version. We use two little gadgets $\mathfrak{H}$ and $\mathfrak{I}$ (see Figure 4.4) to simulate AND and OR gates, respectively, of a circuit. ($\mathfrak{I}$ has been introduced by Immerman [7] to prove non-expressibility results.) $\mathfrak{I}$ has 3 nontrivial automorphisms denoted by $\mathrm{fix}_h, \mathrm{fix}_i, \mathrm{fix}_j$. Each of them fixes one of the pairs hh', ii', jj' (fix_i fixes i and i') and switches the other two (*switching* ii' means mapping i to i' and vice versa). $\mathfrak{H}$ only has one nontrivial automorphism, denoted by swi, that switches all 3 pairs. On the other hand, in the bisimilarity game on $\mathfrak{I}$ the duplicator cannot avoid both $\{ii'\}$ and $\{jj'\}$ from $\{hh'\}$, and in the game on $\mathfrak{H}$, the spoiler can reach both $\{ii'\}$ and $\{jj'\}$ from $\{hh'\}$.

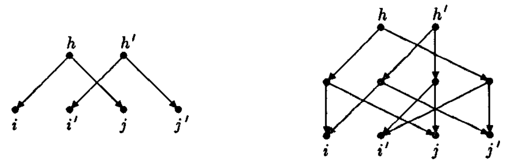

Figure 4.4. *The structures $\mathfrak{H}$ and $\mathfrak{I}$. In both structures, h, h' are coloured green, i, i' are coloured red, and j, j' are coloured blue.*

Proposition 4.5 MC$_2$ $\leq_{\mathrm{qf}}$ BISIMILARITY.

PROOF (sketch). Given a circuit $\mathfrak{C}$ of fan-in 2, we construct a coloured directed graph $\mathfrak{D}$ that contains, for all $a \in \mathfrak{C}$, two vertices a, a' such that $\mathrm{Val}_{\mathfrak{C}}(a) = 1$ if, and only if, $a \equiv_\beta a'$ $(*)$.

It will be obvious that the construction and also the mapping associating a with a, a' is definable by quantifier-free formulae. So let $\mathfrak{C}$ be a circuit of fan-in 2. We say that we *connect* a pair aa' with a pair bb' if we add an edge from a to b and an edge from a' to b'. We start building our directed graph $\mathfrak{D}$ by taking two vertices a, a' for each $a \in C$. For each OR-node $a \in \sqcup^{\mathfrak{C}}$ with parents b, c we add a copy of $\mathfrak{I}$ and connect aa' with hh', connect ii' with bb' and jj' with cc'. For each AND-node $a \in \sqcap^{\mathfrak{C}}$ with parents b, c, we add a copy of $\mathfrak{H}$ and connect aa' with hh', connect ii' with bb', and jj' with cc'. In addition to the colours already used in the gadgets $\mathfrak{I}$ and $\mathfrak{H}$, we colour the node a (but not a') yellow if $a \in Z^{\mathfrak{C}}$.

Observe that, by the last clause of the definition, for input nodes $a \in C$ the mapping $a \mapsto a'$ is a partial isomorphism of $\mathfrak{D}$ if, and only if, the value of a in $\mathfrak{C}$ is 1 (that is, $a \notin Z^{\mathfrak{C}}$). This implies $(*)$ for input nodes. Now an easy induction on the height of $a \in C$ shows that actually $(*)$ holds for all nodes.

The reduction is quantifier-free because it simply replaces each node a of the circuit by a graph that only depends on the quantifier-free type of this node. □

5. Proof of the main theorem

There are two difficulties in trying to prove the PTIME-completeness of the L^k-type query similarly to the completeness of bisimilarity: The first is that positions of the k-pebble game may have size k, rather than just 2 as in the bisimilarity game. The second problem is that in the pebble game the spoiler may choose to play "backwards" (that is, he is no longer restricted to forward-$\exists$ or forward-$\forall$ moves). The proof of Theorem 3.5 proceeds in 4 steps. The first two of them take care of the respective problems mentioned above. In the third we reduce MC_2 to TYPE_k in a similar way as we reduced it to BISIMILARITY in the proof of Proposition 4.5, using the gadgets we have constructed in the first two steps. In the last step, we modify our reduction to reduce MC_2 to $\mathrm{EQUIVALENCE}_k$.

STEP 1. We construct a coloured directed graph, called a *k-threshold gadget*, serving as the basic building block of the structures to be constructed in the following steps. Its "interface" (to the rest of a bigger structure in which it is used) consists of two pairs tt' and uu' of vertices. Its basic property is that the spoiler can reach $\{uu'\}$ from $\{tt'\}$ in the k-pebble game, but not in the $(k-1)$-pebble game. Thus if the spoiler wants to make any progress on this gadget during the game, he needs to concentrate all his resources to the gadget, and in some sense he loses the advantage of playing the k-pebble rather than the 2-pebble game.

The following Lemma stating the existence of threshold gadgets looks quite complicated, but unfortunately it is needed in this form to prove Lemma 5.4 below. The essential statements are (i)-(iii); (iv) and (v) are rather technical and do not contribute much to the understanding of the Lemma.

Lemma 5.1 *For each $k \geq 2$ there exists a coloured graph $\mathfrak{T}^k$ (a k-*threshold gadget*) with two pairs tt' and uu' of vertices such that the following holds:*

(i) In the k-pebble game on $\mathfrak{T}^k$, the spoiler can reach position $\{uu'\}$ from $\{tt'\}$.

(ii) There is an automorphism s of $\mathfrak{T}^k$ with $s^{-1} = s$ that switches tt' and uu' (that is, maps t to t', t' to t, u to u', and u' to u).

The following statements talk about a class of winning strategies for the duplicator in the k-pebble game on $\mathfrak{T}^k$ that we call safe *strategies. A position of the game is* safe *if it occurs in a safe strategy.*

(iii) Positions $\{tt, uu\}$, $\{tt', uu\}$, $\{tt, uu'\}$, and $\{tt', uu'\}$ are safe.

(iv) If P is safe, then there always exist $\bar{t} \in \{tt, tt'\}$ and $\bar{u} \in \{uu, uu'\}$ such that $P \cup \{\bar{t}\}$ and $P \cup \{\bar{u}\}$ are safe. Furthermore, if $P \cup \{\bar{t}\}$ and $P \cup \{\bar{u}\}$ are safe for some $\bar{t} \in \{tt, tt'\}$ and $\bar{u} \in \{uu, uu'\}$ then $P \cup \{\bar{t}, \bar{u}\}$ is also safe.

(v) If a position $\{ab\}$ is safe then either both $\{ab, tt\}$ and $\{ab, tt'\}$ or both $\{ab, uu\}$ and $\{ab, uu'\}$ are also safe. (Intuitively, this says that a position of size 1 does not determine what happens on both sides of the structure).

Observe that the spoiler cannot reach $\{uu'\}$ from $\{tt'\}$ in the $(k-1)$-pebble game on a k-threshold gadget. (Suppose for contradiction that he could. Then he could reach $\{uu', uu\}$ from $\{tt', uu\}$ in the k-pebble game. Now $\{uu', uu\}$ is clearly not a partial isomorphism, and hence the duplicator loses, in contradiction to the statement that $\{tt', uu\}$ is a winning position).

The proof is based on the following Lemma. It is a slight strengthening of Theorem 2.17 in [6]. The proof given there can easily be modified to give the stronger result.

$\mathrm{EDGE}_l(\overset{l}{x}, \overset{l}{y})$ is a formula saying that $\overset{l}{x}\overset{l}{y}$ form a $2l$–clique. $\mathrm{PATH}_l(\overset{l}{x}, \overset{l}{y})$ defines the transitive closure of $\mathrm{EDGE}_l(\overset{l}{x}, \overset{l}{y})$. (Here $\overset{l}{x}$ denotes the l-tuple $x_1 \dots x_l$.)

Lemma 5.2 *For all $n, l \geq 2$ there exists a finite graph $\mathfrak{G}^{l,n}$ with the following properties:*

(i) $G^{l,n}$ can be partitioned into n (disjoint) rows *in a way that there are only edges between elements of the same row or succeeding rows.*

(ii) There exists an automorphism s of $\mathfrak{G}^{l,n}$ preserving the rows such that $s^{-1} = s$.

(iii) There exist a tuple $\overset{l}{c}$ in the first and a tuple $\overset{l}{d}$ in the last row such that

$$\mathfrak{G}^{l,n} \models \mathrm{PATH}_l(\overset{l}{c}, \overset{l}{d}), \quad \textit{but} \quad \mathfrak{G}^{l,n} \not\models \mathrm{PATH}_l(\overset{l}{c}, s(\overset{l}{d})).$$

(iv) For each set $A \subseteq G^{l,n}$ that contains at most $(l-1)$ elements of each row (that may, however, contain elements of several rows) there exists an automorphism f of $\mathfrak{G}^{l,n}$ such that

- *$f^{-1} = f$*
- *f preserves the rows*
- *$\forall a \in A: f(a) = s(a)$*
- *For all $b \in G^{l,n}$ of distance greater than one to A (that is, $\forall a \in A: |\mathrm{row}(b) - \mathrm{row}(a)| > 1$) we have $f(b) = b$.*

PROOF (of Lemma 5.1). Observe that in the case $k = 2$ the lemma is trivial, we just let $\mathfrak{T}^2$ consist of two paths of length 3, one from t to u and one from t' to u'. In general, we have to distinguish between the cases of an even and an odd k.

Assume first that $k > 2$ is even. Let $l = \frac{k}{2}$ and $n = 2k + 2$. $\mathfrak{T}^k$ consists of a copy $\mathfrak{G}$ of $\mathfrak{G}^{l,n}$ (satisfying Lemma 5.2) and the four additional vertices t, t', u, u'. The vertices t and t' are coloured orange and u, u' are coloured purple. Furthermore there are edges between t and every node c_i, between t' and every node $s(c_i)$, between u and every node d_i, and between u' and every node $s(d_i)$ (for $1 \leq i \leq l$), where s and $\overset{l}{c}, \overset{l}{d}$ are chosen according to Lemma 5.2.

We extend the automorphism s from $\mathfrak{G}$ to $\mathfrak{T}^k$ by letting $s(t) = t'$, $s(t') = t$, $s(u) = u'$, $s(u') = u$. This already proves (ii). Statement (i) follows from Lemma 5.2(iii).

We extend the mapping row from $\mathfrak{G}$ to $\mathfrak{T}^k$ by adding t and t' to the first row and u and u' to the last row. In her safe strategies, the duplicator always plays row preserving (that is, if the spoiler selects an element in an $\exists$ or $\forall$ move the duplicator answers by selecting an element of the same row). In a position P of the game we call a row r *empty* if P contains no elements of row r. We call row r *critical* if

- $r \in \{1,2\}$ and P contains at least 1 pair of elements of row 1 or row 2, or
- $3 \leq r \leq n-2$ and P contains at least l pairs of elements of row r, or
- $r \in \{n-1, n\}$ and P contains at least 1 pair of elements of row $n-1$ or row n.

We call a strategy for the duplicator *safe* if in each position P occuring in this strategy there exists an automorphism h such that:

(1) $h^{-1} = h$

(2) h preserves the rows

(3) h is the identity on the first and last row

(4) For each pair $ab \in P$ we either have $h(a) = b$ or $s(h(a)) = b$. In the first case we call the row of a and b *fixed*, in the second case *switched*.

(5) No row is fixed and switched, and neighboured rows both containing pebbled elements are either both fixed or both switched.

(6) The distance between two critical rows one of which is fixed and the other switched is $\geq k$.

Observe that safe strategies are indeed winning strategies for the duplicator, since each position satisfying (2)–(5) gives rise to a partial isomorphism of $\mathfrak{T}^k$. Once we have proved that safe strategies exist, (iii)–(v) are immediate.

So suppose P is a position satisfying (1)–(6) via the automorphism h. We have to show that the duplicator can play in a way that (1)–(6) still hold in the next position. Clearly this is the case if the spoiler removes a pair from P. So let $P = \{a_1b_1, \ldots, a_{k-1}b_{k-1}\}$ (assuming without loss of generality that $|P| = k-1$). Say, the spoiler decides to make an $\exists$-move (the case of a $\forall$-move can be treated symmetrically) and selects $a \in T^k$. We shall define the duplicator's answer to this move and a new automorphism h' such that the new position satisfies (1)–(6) via h'. Let r be the row of a.

CASE 1: $r \in \{1,2\}$

(a) Rows 1 and 2 are both empty.
Let p be the smallest non-empty row, and q the smallest critical row (if no critical row exists, let $q = n+1$). If $q \geq k+2$, we let row r be switched if, and only if, row p is switched and $h' = h$. That is, the duplicator selects $b = h(a)$ if row p is fixed and $b = s(h(a))$ if row p is switched. Clearly, this choice guarantees (1)–(6) in the new position. If $q \leq k+1$, we want to let row r be switched if, and only if, row q is switched (to preserve (6)). Assume, without loss of generality, that q is switched. If p is also switched we have no problems. So assume further that p is fixed. By (5), there must be an empty row p' such that $p < p' < q$ and all rows between $p-1$ and p' are fixed. Let $\{a_{i_1}b_{i_1}, \ldots, a_{i_m}b_{i_m}\} \subseteq P$ be the set of pairs in P which are contained in a row between $p-1$ and p' and $A = \{b_{i_1}, \ldots, b_{i_m}\}$. Then A contains at most $(l-1)$ elements per row (since q is the smallest critical row). By Lemma 5.2 (iv) there exists an automorphism f of $\mathfrak{G}$ such that for all $x \in A$ we have $f(x) = s(x)$ whereas f is the identity on all rows below $p-1$ or above p'. We let $h' = f \circ h$, and the duplicator selects $b = s(h'(a))$. By the choice of f, (1)–(2) remain valid. (3) follows from the fact that $p \geq 3$ and $p' \leq n-1$. For all elements x in a row below $p-1$ or above p' we have $h'(x) = h(x)$ thus (4) still holds in these rows. If $a_ib_i \in P$ is in a row between $p-1$ and p' we have $h(a_i) = b_i$ (by our assumption that these rows are fixed in position P'), hence

$$s(h'(a_i)) = s(f(b_i)) = s(s(b_i)) = b_i.$$

Thus these rows are switched in the new position. By the definition of b row r is switched. Hence (5) holds in the new position. (6) holds since both r and q are switched.

(b) There is an $r' \in \{1,2\}$ such that row r' is not empty. The duplicator wants row r to be switched if and only if row r' is switched. This might cause a problem if $r' = 1, r = 2$, row 3 is not empty, row 1 is switched and row 3 is fixed (or vice versa, but let us assume without loss of generality that 1 is switched and 3 is fixed). Let q be the smallest critical row ≥ 3 (or $n+1$ if no critical row ≥ 3 exists). As in (a), there is a smallest empty row p' such that $3 < p' < q$. There exists an automorphism f that switches the elements of P between row 2 and p' and leaves row 1 and all rows above p' fixed. We let $h' = f \circ h$, and the duplicator selects $b = s(h'(a))$. Again, it is easy to see that (1)–(6) still hold in the new position.

CASE 2: $3 \leq r \leq n-2$

(a) Row r contains precisely $l-1$ elements of P.
Observe, and this is crucial, that there is at most one critical row q such that $|r-q| < k$. (Suppose for contradiction that there are two critical rows q and q' with $|q-r| < k$ and $|q'-r| < k$. Then $|q-q'| \leq (2k-2)$. Hence we do not have both $q \leq 2$ and $q' \geq n-1$ (or vice versa), and since they are both critical either q or q' contains at least l elements of P. The other (or its neighbour) contains at least one, and row r contains $l-1$ elements, but P has size $k-1 = 2l-1$, giving us the desired contradiction). If there is no critical

row at all we just let $h' = h$, and the duplicator selects $b = h(a)$ or $s(h(a))$, depending on whether row r is fixed or switched. Otherwise, let q be the nearest critical row (to r). If both row q and r are fixed or both are switched we let $h' = h$ and the duplicator selects $b = h(a)$ or $b = s(h(a))$, respectively. So let us assume without loss of generality that row q is fixed and row r is switched. Let $p < r$ and $p' > r$ be the nearest empty rows. By (5), we either have $q < p$ or $q > p'$. Furthermore, each row between p and p' contains at most $l - 1$ elements of P, so again we can adjust things by a suitable automorphism.

(b) Row r contains at most $(l - 2)$ or at least l elements of P.

If row r is not empty we let $h' = h$ and the duplicator places her pebble on $h(a)$ or $s(h(a))$, depending on whether row r is fixed or switched. So assume that row r is empty. If both row $(r - 1)$ and row $(r + 1)$ are fixed or both are switched we also let $h' = h$ and the duplicator places her pebble on $h(a)$ or $s(h(a))$, respectively. If one of the neighboured rows is empty we again let $h' = h$ and the duplicator decides where to place her pebble depending on whether the other neighboured row is fixed or switched. If it is also empty the duplicator decides to fix row r, that is, she places her pebble on $h(a)$. Otherwise, we assume without loss of generality that row $(r-1)$ is switched and row $(r+1)$ is fixed. Let $p < r$ be the nearest empty row below r (if no such row exists let $p = 0$). Analogously let $p' > r$ be the nearest empty row above r (or $p' = n + 1$ if no such row exists). Since the size of P is $(k-1)$ we have $p' - p \leq k+1$ (by the pigeonhole principle). Note that there cannot exist two critical rows q and q' such that $p < q < r < q' < p'$ (by (6)) and assume without loss of generality that there exists no critical row between p and r. As in the previous cases, we can adjust things by a suitable automorphism switching just the elements of P between rows p and r.

CASE 3: $r \in \{n - 1, n\}$.
Can be treated completely analogously to the first case.

To complete the proof we still have to consider the case that $k \geq 3$ is odd. Here $\mathfrak{T}^k$ consists of a copy of $\mathfrak{T}^{2k}$, together with a new node x_{ab} for every unordered pair $\{a, b\}$ such that a and b are either in the same or in neighboured rows. Each node x_{ab} has an edge to both a and b and is coloured brown. Obviously, the winning strategy for the duplicator for the $2k$-pebble on $\mathfrak{T}^{2k}$ we have described above can easily be translated to a winning strategy for the k-pebble game on $\mathfrak{T}^k$ (simply by representing the selection of x_{ab} in an $\exists$ or $\forall$ move of the k-pebble game on $\mathfrak{T}^k$ by two moves of the $2k$ pebble game on $\mathfrak{T}^{2k}$ in which a and b are selected). Clearly the translations of safe strategies still preserve (iii)–(v).

However, we cannot necessarily translate every strategy for the spoiler for the $2k$-pebble game on $\mathfrak{T}^{2k}$ to a strategy for the k-pebble game on $\mathfrak{T}^k$. Thus (i) still has to be checked. The spoiler's strategy to reach uu' from tt' basically remains the same, namely go along an EDGE$_k$-path from the first row to the last and use that there exist such paths from $\overset{k}{c}$ to $\overset{k}{d}$ and from $s(\overset{k}{c})$ to $s(\overset{k}{d})$, but not from $s(\overset{k}{c})$ to $\overset{k}{d}$. Let $\overset{k}{a}_1 = \overset{k}{c}, \ldots, \overset{k}{a}_n = \overset{k}{d}$ be an EDGE$_k$-path. Starting in position $\{tt'\}$, the spoiler selects the following $(k-1)$ elements: $x_{a_{11}a_{12}}, x_{a_{13}a_{14}}, \ldots, x_{a_{1k}a_{21}}, x_{a_{22}a_{23}}, \ldots, x_{a_{2(k-3)}a_{2(k-2)}}$. If she does not want to lose right away, the duplicator has to answer by selecting: $x_{s(a_{11})s(a_{12})}, x_{s(a_{13})s(a_{14})}, \ldots, x_{s(a_{1k})s(a_{21})}, x_{s(a_{22})s(a_{23})}, \ldots, x_{s(a_{2(k-3)})s(a_{2(k-2)})}$. Then the spoiler removes the pair tt' and selects $x_{a_{2(k-1)}a_{2k}}$ in an $\exists$-move; after that he removes the pair $x_{a_{11}a_{12}} x_{s(a)_{11}s(a)_{12}}$ and selects $x_{a_{21}a_{22}}$, et cetera. □

Remark 5.3 Our k-threshold gadgets are exponentially large in k. It would be interesting to construct k-threshold gadgets of polynomial size. This might be possible since in some sense Lemma 5.2, on which our construction is based, is far too strong (by stating the existence of automorphisms between certain $(k-1)$-tuples where we only need the tuples to have the same L^k-type).

As an alternative to the proof given here, it seems to be possible to prove Lemma 5.1 using methods of Cai, Fürer, and Immerman [4]. However, this approach also gives exponentially large gadgets, and it does not appear to be easier.

STEP 2. We need another gadget that takes care of the "backwards" moves by having the property that the spoiler can only transport information in one direction.

Lemma 5.4 *For each $k \geq 2$, there exists a k-*one-way gad*get with two pairs xx' and yy' of vertices such that in the k-pebble game on this gadget the following holds:*
(i) The spoiler can reach $\{yy'\}$ from $\{xx'\}$.
(ii) The duplicator can avoid $\{xx\}$ from $\{xx', yy\}$ and $\{xx', yy'\}$.
(iii) The duplicator can avoid $\{xx'\}$ from $\{xx, yy'\}$. In particular, the spoiler cannot reach $\{xx'\}$ from $\{yy'\}$.

PROOF *(sketch).* The k-one-way gadget is built as in Figure 5.5 from the structures $\mathfrak{I}$ and $\mathfrak{H}$ (cf. Figure 4.4) and two copies $\mathfrak{T}_{\text{red}}$ and $\mathfrak{T}_{\text{blue}}$ of the k-threshold gadget. To reach

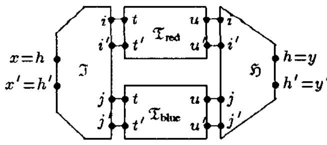

Figure 5.5 *A one-way gadget.*

$\{yy'\}$ from $\{xx'\}$, the spoiler uses the fact that the duplicator cannot avoid both $\{ii'\}$ and $\{jj'\}$ from $\{hh'\}$ $(= \{xx'\})$ in the game on $\mathfrak{I}$. From both of these positions, he can reach the corresponding position in $\mathfrak{H}$ via the threshold gadget, and from there he can reach $\{yy'\}$.

The strategy for the duplicator in (iii) is very simple. On $\mathfrak{I}$ she plays according to the automorphism fix_h fixing hh' $(= xx')$ and switching ii' and jj', on $\mathfrak{H}$ according to the automorphism swi switching everything, and on the threshold gadgets inbetween according to the automorphism s switching both tt' and uu'.

The strategies required in (ii) are more complicated. The basic idea is that, in order to explore the mismatch between $\mathfrak{I}$ on the left and $\mathfrak{H}$ on the right side, the spoiler would have to reach $\{uu'\}$ from $\{tt'\}$ in both $\mathfrak{T}_{\text{red}}$ and $\mathfrak{T}_{\text{blue}}$ in parallel. However, this is not possible with only k pebbles because of the "threshold" properties of the threshold gadgets.

STEP 3. As in the proof of Proposition 4.5, given a circuit $\mathfrak{C}$ of fan-in 2, we construct a coloured directed graph $\mathfrak{D}$ which contains, for all $a \in \mathfrak{C}$, two vertices a, a' such that $\mathrm{Val}_{\mathfrak{C}}(a) = 1$ if, and only if, $a \equiv^{\mathfrak{D}}_{L^k} a'$. Again, the construction is definable by quantifier-free formulae.

Given $\mathfrak{C}$, the graph $\mathfrak{D}$ is defined as follows (see Figure 5.6): For each $a \in C$ we take 2 vertices a, a'. For each OR-node $a \in \sqcup^{\mathfrak{C}}$ with parents b, c we add a copy $\mathfrak{I}_a$ of $\mathfrak{I}$ and two copies $\mathfrak{O}^{\text{black}}_a$ and $\mathfrak{O}^{\text{white}}_a$ of the k-one-way gadget. We connect aa' with hh' of $\mathfrak{I}_a$, connect ii' and jj' of $\mathfrak{I}_a$ with xx' of $\mathfrak{O}^{\text{black}}_a$ and $\mathfrak{O}^{\text{white}}_a$, respectively, and yy' of $\mathfrak{O}^{\text{black}}_a$ and $\mathfrak{O}^{\text{white}}_a$ with bb' and cc', respectively. Analogously, for each AND-node $a \in \sqcap^{\mathfrak{C}}$ with parents b, c we add a copy $\mathfrak{H}_a$ of $\mathfrak{H}$ and two copies $\mathfrak{O}^{\text{black}}_a$ and $\mathfrak{O}^{\text{white}}_a$ of the k-one-way gadget. We connect aa' with hh' of $\mathfrak{H}_a$, connect ii' and jj' of $\mathfrak{H}_a$ with xx' of $\mathfrak{O}^{\text{black}}_a$ and $\mathfrak{O}^{\text{white}}_a$, respectively, and yy' of $\mathfrak{O}^{\text{black}}_a$ and $\mathfrak{O}^{\text{white}}_a$ with bb' and cc', respectively. In addition to the colours already used we colour node a (but not a') yellow if $a \in Z^{\mathfrak{C}}$.

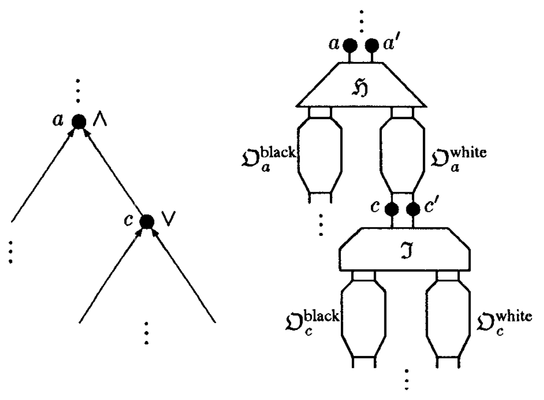
a
$\wedge$
c
$\vee$
a
a'
$\mathfrak{H}$
$\mathfrak{O}^{\text{black}}_a$
$\mathfrak{O}^{\text{white}}_a$
c
c'
$\mathfrak{I}$
$\mathfrak{O}^{\text{black}}_c$
$\mathfrak{O}^{\text{white}}_c$

Figure 5.6 *A part of a circuit $\mathfrak{C}$ and its translation to $\mathfrak{D}$.*

Now the proof that the spoiler wins the k-pebble game on $\mathfrak{D}$ with initial-position $\{aa'\}$ if $\mathrm{Val}_{\mathfrak{C}}(a) = 0$ is obtained via an easy induction on the height of a in $\mathfrak{C}$. Conversely, we have to prove that if $\mathrm{Val}_{\mathfrak{C}}(a) = 1$ then the duplicator has a winning strategy for the k-pebble game on $\mathfrak{D}$ with initial position $\{aa'\}$.

As usually, the duplicator plays in a way that she preserves the building blocks of the structure. For any position P of the game and $a \in \sqcup^{\mathfrak{C}}$ with parents b, c we let

$$P_{\mathfrak{I}_a} = \begin{cases} (P \cap (I_a)^2) \cup \{hh\} & \text{if } aa \in P \text{ or } a'a' \in P \\ (P \cap (I_a)^2) \cup \{hh'\} & \text{if } aa' \in P \text{ or } a'a \in P \\ P \cap (I_a)^2 & \text{otherwise,} \end{cases}$$

$$P_{\mathfrak{O}^{\text{black}}_a} = \begin{cases} (P \cap (O^{\text{black}}_a)^2) \cup \{yy\} & \text{if } bb \in P \text{ or } b'b' \in P \\ (P \cap (O^{\text{black}}_a)^2) \cup \{yy'\} & \text{if } bb' \in P \text{ or } b'b \in P \\ P \cap (O^{\text{black}}_a)^2 & \text{otherwise.} \end{cases}$$

Similarly, we define $P_{\mathfrak{O}^{\text{white}}_a}$ and $P_{\mathfrak{H}_a}, P_{\mathfrak{O}^{\text{black}}_a}, P_{\mathfrak{O}^{\text{white}}_a}$ for $a \in \sqcap^{\mathfrak{C}}$. We say that a one-way gadget $\mathfrak{O}$ *is sitting on aa'* if it is a substructure $\mathfrak{O}^{\text{black}}_b$ or $\mathfrak{O}^{\text{white}}_b$ of $\mathfrak{D}$ such that yy' of $\mathfrak{O}$ is connected to aa'.

We show that the duplicator has a strategy such that in every position P of the game the following holds for all $a \in C$:

CASE 0: $\mathrm{Val}_{\mathfrak{C}}(a) = 0$. Then for any $\mathfrak{O}$ sitting on aa', $P_{\mathfrak{O}}$ extends to the identity. Furthermore, if a is not an input node, $P_{\mathfrak{I}_a}$ or $P_{\mathfrak{H}_a}$ (depending on what kind of node a is) extends to the identity and the duplicator has winning strategies for the k-pebble game on $\mathfrak{O}^k$ with initial positions $P_{\mathfrak{O}^{\text{black}}_a}$ and $P_{\mathfrak{O}^{\text{white}}_a}$ that avoid $\{xx'\}$.

CASE 1: $\mathrm{Val}_{\mathfrak{C}}(a) = 1$. There are two possibilities:

(1) The first is that for every $\mathfrak{O}$ sitting on aa', position $P_{\mathfrak{O}} \cup \{yy\}$ is a winning position for the duplicator in the k-pebble game on $\mathfrak{O}^k$. Moreover, if a is not an input node then $P_{\mathfrak{I}_a}$ or $P_{\mathfrak{H}_a}$ extends to the identity and the duplicator has winning strategies for the k-pebble game on $\mathfrak{O}^k$ with initial positions $P_{\mathfrak{O}^{\text{black}}_a}$ and $P_{\mathfrak{O}^{\text{white}}_a}$ that avoid $\{xx'\}$.

(2) The second possibility is that for every $\mathfrak{O}$ sitting on aa', $P_{\mathfrak{O}} \cup \{yy'\}$ is a winning position for the duplicator in the k-pebble game on $\mathfrak{O}^k$. Furthermore, if a is an AND-node then $P_{\mathfrak{H}_a}$ extends to the automorphism swi of $\mathfrak{H}$ and the duplicator has winning strategies for the k-pebble game on $\mathfrak{O}^k$ with initial positions $P_{\mathfrak{O}^{\text{black}}_a}$ and $P_{\mathfrak{O}^{\text{white}}_a}$ that avoid $\{xx\}$. If a is an OR-node then $P_{\mathfrak{I}_a}$ extends to the automorphism fix_i (or fix_j) of $\mathfrak{I}$ and the duplicator has a winning strategy for the k-pebble game on $\mathfrak{O}^k$ with initial position $P_{\mathfrak{O}^{\text{black}}_a}$ ($P_{\mathfrak{O}^{\text{white}}_a}$, respectivly) that avoids $\{xx'\}$ and a winning strategy for the game with initial position $P_{\mathfrak{O}^{\text{white}}_a}$ ($P_{\mathfrak{O}^{\text{black}}_a}$, respectively) that avoids $\{xx\}$.

Actually we need a slightly stronger statement; the different requirements for a position $P_{\mathfrak{O}}$ associated with a one way gadget $\mathfrak{O} = \mathfrak{O}^{\text{black}}_b$ or $\mathfrak{O}^{\text{white}}_b$ should be consistent in the sense that, for example, if the duplicator is required to have a winning strategy for the game on $\mathfrak{D}^k$ with initial position $P_{\mathfrak{D}}$ that avoids $\{xx'\}$, and also $P_{\mathfrak{D}} \cup \{yy\}$ needs to be a winning position for the duplicator, then actually we want the duplicator to have a winning strategy for the k-pebble game on $\mathfrak{D}^k$ with initial position $P_{\mathfrak{D}} \cup \{yy\}$ that avoids $\{xx'\}$. It will be clear, however, that the strategy given below actually preserves this stronger condition.

Observe that Case 0 and Case 1 are consistent with each other just because we installed $\mathfrak{I}$'s for the OR-nodes and $\mathfrak{H}$'s for the AND-nodes.

What the conditions above really do is give a "local strategy" for the duplicator on all the building blocks of $\mathfrak{D}$ and say that the local strategies are consistent on the interfaces between the building blocks. The duplicator's strategy is to play according to these local strategies everywhere and maintain the consistency between them.

Note that for any $a \in C$ with $\text{Val}_{\mathfrak{C}}(a) = 1$ position $\{aa'\}$ satisfies the conditions above. Since obviously removing pebbles creates no problems, let us assume we are in a position P of size $(k-1)$ satisfying the conditions above, and the spoiler selects $d \in D$ in an $\exists$-move (as usual, a $\forall$-move is treated analogously).

CASE A. $d \in I_a$ (or $d \in H_a$) for some $a \in C$.
The duplicator just chooses d' according to the automorphism which determines the local strategy on $\mathfrak{I}_a$ ($\mathfrak{H}_a$, respectively).

CASE B. $d \in \{a, a'\}$ for some $a \in C$.
The duplicator selects $d' = a$ or $d' = a'$, whatever is consistent with her local strategy on the adjacent $\mathfrak{I}_a$ or $\mathfrak{H}_a$, and the $\mathfrak{O}$ sitting on aa'.

CASE C. $d \in \mathfrak{O}_a^{\text{black}}$ for some $a \in C$
The local strategy on $\mathfrak{O}_a^{\text{black}}$ is some winning strategy for the duplicator from initial position $P_{\mathfrak{O}_a^{\text{black}}} \cup \{yy\}$ or $P_{\mathfrak{O}_a^{\text{black}}} \cup \{yy'\}$ avoiding $\{xx\}$ or $\{xx'\}$. The strategy might just be to play identically, and actually it must be if $\mathfrak{O}_a^{\text{black}}$ is sitting on some bb' where $\text{Val}_{\mathfrak{C}}(b) = 0$ (by Case 0 above).

If the strategy is to play identically the duplicator selects $d' = d$ and is fine. If not, $\mathfrak{O}_a^{\text{black}}$ is sitting on a bb' where $\text{Val}_{\mathfrak{C}}(b) = 1$. We have to distinguish two subcases now.

SUBCASE C1. $|P_{\mathfrak{O}_a^{\text{black}}}| \leq k-2$.
The duplicator answers according to her local strategy, but actually pretending that she was in position $P_{\mathfrak{O}_a^{\text{black}}} \cup \{yy'\}$ or $P_{\mathfrak{O}_a^{\text{black}}} \cup \{yy'\}$ rather than just $P_{\mathfrak{O}_a^{\text{black}}}$. This guarantees that nothing happens at the "interface" yy' that might collide with the positions on other building blocks of $\mathfrak{D}$.

SUBCASE C2. $|P_{\mathfrak{O}_a^{\text{black}}}| = k-1$.
Again the duplicator chooses d' according to her local strategy on $\mathfrak{O}_a^{\text{black}}$, but now she has to start from the actual position $P_{\mathfrak{O}_a^{\text{black}}}$. Let $P' = P \cup \{dd'\}$ be the new position and consider $P'_{\mathfrak{O}_a^{\text{black}}}$. It might happen that $P'_{\mathfrak{O}_a^{\text{black}}} \cup \{yy'\}$ is no longer a winning position for the duplicator although $P_{\mathfrak{O}_a^{\text{black}}} \cup \{yy'\}$ was, and this changes the behaviour of $\mathfrak{O}_a^{\text{black}}$ at its interface. But it cannot interfere with anything on any other building block since we have $P' = P'_{\mathfrak{O}_a^{\text{black}}}$, that is, no elements from any other part of the structure are in the current position. And since $\mathfrak{O}_a^{\text{black}}$ is sitting on a bb' where $\text{Val}_{\mathfrak{C}}(b) = 1$ we have no problems there.

Note that such a problem cannot occur on the other side, that is, with xx', since the local strategies avoid either $\{xx'\}$ or $\{xx\}$.

CASE D. $d \in \mathfrak{O}_a^{\text{white}}$ for some $a \in C$.
Can be treated analogously.

STEP 4. We have reduced a given circuit $\mathfrak{C}$ to a (coloured) directed graph $\mathfrak{D}$ with a pair cc' corresponding to the output node $c^{\mathfrak{C}}$ of $\mathfrak{C}$ in a way that $\text{Val}_{\mathfrak{C}}(c^{\mathfrak{C}}) = 1$ if, and only if, $c \equiv_{\text{L}^k}^{\mathfrak{D}} c'$. We further reduce $\mathfrak{D}$ to a pair $\mathfrak{A}, \mathfrak{B}$ of structures such that $\mathfrak{A} \equiv_{\text{L}^k} \mathfrak{B}$ if, and only if, $c \equiv_{\text{L}^k}^{\mathfrak{D}} c'$. Both $\mathfrak{A}$ and $\mathfrak{B}$ consist of a copy of $\mathfrak{D}$ and a k-one-way gadget. The pair yy' of the one-way gadget is connected to cc'. Furthermore, in structure $\mathfrak{A}$ the vertex x of the one-way gadget is coloured pink, whereas in $\mathfrak{B}$ vertex x' is coloured pink. Then obviously the spoiler wins the k-pebble game on $\mathfrak{A}, \mathfrak{B}$ if he wins the game on $\mathfrak{D}$ with initial position cc'. (In the first move he selects x, and the duplicator has to answer by selecting x' as the only pink vertex of $\mathfrak{B}$. From $\{xx'\}$ he can reach $\{yy'\}$ and then $\{cc'\}$, and he wins.) On the other hand, if the duplicator has a winning strategy for the game on $\mathfrak{D}$ with initial position $\{cc'\}$ she can easily extend it to a strategy for the game on $\mathfrak{A}, \mathfrak{B}$, by playing a strategy avoiding $\{xx\}$ on the one-way gadget. □

6. Towards graph isomorphism

Very little is known about the precise complexity of graph isomorphism. In particular, it is an open question whether this problem is hard for PTIME. So, it is tempting to try to generalize our proof. Unfortunately, there is no (obvious) way to extend it to isomorphism rather than L^k-equivalence. It is heavily based on the existence of one-way gadgets, and clearly for isomorphism no such gadgets can exist. The reason is a fundamental algebraic difference between the classes of elements of the same L^k-type and the orbits of the automorphism group of a structure. The former are just classes of an equivalence relation that a priori does not satisfy any algebraic conditions. It is much easier to find structures in which this equivalence relation has certain nice properties, such as being "directed" as in the one-way gadgets. But there are no good reasons why graph isomorphism should be hard for PTIME anyway.

Still we can pursue our logical approach, replacing the L^k by more expressive logics whose equivalences give better approximations of isomorphism.

Adding counting quantifiers

Immerman and Lander [12] observed that all obvious differences between L^k-equivalence and isomorphism are based on a similar principle, namely the fact that when restricted to k variables first-order logic loses its ability to count. Consider, for example, two complete graphs of sizes k and $k+1$, respectively. They are obviously not isomorphic, but an easy pebble-game argument shows that they are L^k-equivalent. To overcome such difficulties, Immerman and Lander suggested to add *counting quantifiers* $\exists^{\geq m}$ (for each $m \geq 1$) to the logic. The formula $\exists^{\geq m} x\varphi(x)$ has the obvious meaning "there exist at least m distinct elements x that satisfy $\varphi(x)$". We let C^k denote the logic built with only k-variables from the Boolean connectives $\vee$, $\neg$ and the quantifiers $\exists^{\geq m}$ for $m \geq 1$. Note that the quantifiers $\exists^{\geq m}$ are definable in first-order logic, but require m variables. Thus $L^k \subset C^k \subset FO$ for all $k \geq 1$.

Equivalence in C^k is apparently a much better approximation to graph isomorphism. Essentially, it captures the colour refinement method of graph canonization (see [4]). Although Cai, Fürer and Immerman [4] have shown that there are non-isomorphic graphs (even of bounded degree) that are not C^k-equivalent, Immerman and Lander [12] demonstrated that on many important classes of graphs C^k-equivalence corresponds to isomorphism. On the other hand, C^k-equivalence shares the convenient properties of L^k-equivalence: There is a nice game characterization and a PTIME-algorithm that decides C^k-equivalence. In the full paper we show that our hardness proof for equivalence can be extended to the logics C^k (for $k \geq 2$) without much further effort. Hence we have:

Theorem 6.1 *For each $k \geq 2$, C^k-equivalence is complete for polynomial time under quantifier-free reductions.*

Analysing the proof carefully, we actually find that the hardness result holds for all logics whose expressive power is between that of L^k and C^k.

7. Further research

Our work suggest two lines of further research: On the logical side, the question occurs whether our result extends to least fixed-point logic on arbitrary finite structures, that is, whether L^k-equivalence is complete for least fixed-point logic under quantifier-free reductions. This question is motivated by the well-known fact that PTIME is precisely the collection of classes of ordered finite structures that are definable in least-fixed-point logic [9, 17], and by the observation that L^k-equivalence is definable in least-fixed-point logic [13]. On the complexity theoretical side, it would be interesting to find more expressive logics (whose equivalence approximates isomorphism even better) for which equivalence remains PTIME-hard.

Acknowledgement

I would like to thank Phokion Kolaitis, for motivating me to work on these questions, and for many helpful discussions at any stage of this work.

References

[1] S. Abiteboul and V. Vianu. Generic computation and its complexity. In *Proceedings of the 23rd ACM Symposium on Theory of Computing*, pages 209–219, 1991.

[2] J. Balcázar, J. Gabarró, and M. Sántha. Deciding bisimilarity is P-complete. *Formal Aspects of Computing*, 4:638–648, 1992.

[3] J. Barwise. On Moschovakis closure ordinals. *Journal of Symbolic Logic*, 42:292–296, 1977.

[4] J. Cai, M. Fürer, and N. Immerman. An optimal lower bound on the number of variables for graph identification. *Combinatorica*, 12:389–410, 1992.

[5] A. Dawar, S. Lindell, and S. Weinstein. Infinitary logic and inductive definability over finite structures. *Information and Computation*, 119:160–175, 1995.

[6] M. Grohe. Arity hierarchies, 1995. To appear in *Annals of Pure and Applied Logic*.

[7] N. Immerman. Number of quantifiers is better than number of tape cells. *Journal of Computer and System Sciences*, 22:384–406, 1981.

[8] N. Immerman. Upper and lower bounds for first-order expressibility. *Journal of Computer and System Sciences*, 25:76–98, 1982.

[9] N. Immerman. Relational queries computable in polynomial time. *Information and Control*, 68:86–104, 1986.

[10] N. Immerman. Languages that capture complexity classes. *SIAM Journal of Computing*, 16:760–778, 1987.

[11] N. Immerman and D. Kozen. Definability with a bounded number of variables. *Information and Computation*, 83:121–139, 1989.

[12] N. Immerman and E. Lander. Describing graphs: A first-order approach to graph canonization. In A. Selman, editor, *Complexity theory retrospective*, pages 59–81. Springer-Verlag, 1990.

[13] P. G. Kolaitis and M. Y. Vardi. Fixpoint logic vs. infinitary logic in finite-model theory. In *Proceedings of the 7th IEEE Symposium on Logic in Computer Science*, 1992.

[14] P. G. Kolaitis and M. Y. Vardi. On the expressive power of variable-confined logics. In *Proceedings of the 11th IEEE Symposium on Logic in Computer Science*, 1996.

[15] M. Otto. Bounded variable logics and counting — a study in finite model theory, 1995. Habilitationsschrift at the RWTH-Aachen.

[16] B. Poizat. Deux ou trois choses que je sais de L_n. *Journal of Symbolic Logic*, 47:641–658, 1982.

[17] M. Y. Vardi. The complexity of relational query languages. In *Proceedings of the 14th ACM Symposium on Theory of Computing*, pages 137–146, 1982.

[18] H. Veith. Succinct representation, leaf languages and projection reductions. In *Proceedings of the 11th IEEE Conference on Computational Complexity*, 1996.