Math. Struct. in Comp. Science (1996), *vol.* 6, *pp.* 505–526

Linear time computable problems and first-order descriptions

DETLEF SEESE

University Karlsruhe (TH), Germany
Email: seese@aifb.uni-karlsruhe.de

Received November 1995; revised 15 May 1996

It is well known that every algorithmic problem definable by a formula of first-order logic can be solved in polynomial time, since all these problems are in **L** (see Aho and Ullman (1979) and Immerman (1987)). Using an old technique of Hanf (Hanf 1965) and other techniques developed to prove the decidability of formal theories in mathematical logic, it is shown that an arbitrary **FO**-problem over relational structures of bounded degree can be solved in linear time.

1. Introduction

The investigation of connections between complexity classes on the one hand and descriptions in (logical) languages on the other started with Ronald Fagin's seminal paper Fagin (1974), he proved that **NP** coincides with the class of problems expressible in existential second-order logic (Σ_1^1). Immerman (Immerman 1986) proved corresponding results for **P**, **NL** and several other complexity classes using extensions of the classical first-order calculus by various operators, and thus prompted the development of descriptive complexity as its own branch of complexity theory. Many researchers followed this fruitful path and thus showed the usefulness of this connection between logic and complexity theory (see, for example, Ajtai and Fagin (1990), Fagin *et al.* (1993), Immerman (1987), Gurevich (1984), de Rougemont (1987) and Schwentick (1994)). Several of the articles in this area have concentrated on the famous **P-NP**–Problem (or the **NP-Co-NP**–Problem)† and investigated languages whose expressive power is higher than that of first-order logic. In particular, the very strong monadic second-order logic (MSO) got much attention (see Ajtai and Fagin (1990), Fagin *et al.* (1993), Schwentick (1994), Arnborg *et al.* (1991), Courcelle and Mosbah (1993) and Courcelle (1994)). This logic was not only of interest because it gave hope of shedding some light into the separability problem for **NP** and **Co-NP**, but also because all algorithmic problems expressible in monadic second-order logic become solvable in linear time for classes of structures of uniformly bounded tree-width.

† **NP** (or **P**, respectively) is the class of problems decidable by a non-deterministic (or deterministic, respectively) algorithm requiring polynomial time, and **Co-NP**$= \{\bar{L} : L \in \mathbf{NP}\}$, where $\bar{L}$ denotes the complement of L). The **P-NP**–Problem is the question of whether **P**$=$ **NP**, while, similarly, the **NP-Co-NP**–Problem asks whether **NP**$=$ **Co-NP**.

As long as the **P-NP**–Problem is undecided, there is not much hope of solving every MSO-problem for arbitrary structures in linear time, since many **NP**–complete problems are expressible in this logic. Another interesting branch of research was the investigation of extensions of first-order logic by certain operators, as, for example, a fixpoint operator (LFP), for which it was proved that $\mathbf{P} = (\mathbf{FO} + \mathrm{LFP})$ if structures with an ordered universe are considered (see Immerman (1986)). Pure first-order logic did not get as much attention, since its expressive power is relatively weak and it was one of the early results of this area that **FO** is strictly contained in **L**, and hence in **P**, where **L** denotes the class of deterministic logspace computable problems (see Aho and Ullman (1979) and Immerman (1982)).

Nevertheless, first-order logic is not so poor that interesting problems cannot be stated in it. It is an interesting question in graph theory to investigate the structure of graphs that do not have a given finite graph H as an induced subgraph. For most graphs H the problem of characterizing the structure of those graphs without an induced isomorphic copy of H is open, even for very simple ones (see Giakoumakis (1995)). However, for each fixed finite graph H the property

$$P_H(G) :\Leftrightarrow \text{'the graph } H \text{ is not an induced subgraph of the given graph } G\text{'}$$

is a first-order property, and hence it has a polynomial time solution. But in most cases (*e.g.*, if H is a path of length 5) a characterization of the structure of G is not known, so there is no really efficient algorithm to solve the problem P_H.

Stolboushkin and Taitslin (1994) addressed the question of whether **FO** is contained in an initial segment of **PTIME**, *i.e.*, in a class $\mathrm{DTime}(n^k)$ for a certain k. It was proved that if there is no such k, then **P≠PSPACE**. Moreover, it was shown there that if the least fixed point hierarchy collapses by dimension on ordered models, there is no such k. Hence, in this case it is impossible to prove that each **FO** problem can be solved in linear time. However, arbitrary signatures are considered here.

In cases where the signature contains unary relation symbols only, one can use quantifier elimination to show that any formula is equivalent to one of quantifier depth 1 (see Behmann (1922) and Maltsev (1959)). Hence, all **FO** problems over such signatures can be solved in linear time. However, in general, the quantifier hierarchy of first-order formulas does not collapse (see Chandra and Harel (1980)). Thus, in general, one cannot use the elimination of quantifiers to find efficient algorithms.

The main result of this article is a proof that each **FO** problem can be solved in linear time if only relational structures of bounded degree are considered. The basic idea of the proof is a localization technique based on a method that was originally developed by Hanf (Hanf 1965) to show that the elementary theories of two structures are equal under certain conditions, *i.e.*, that two structures agree on all first-order sentences. Fagin, Stockmeyer and Vardi (Fagin *et al.* 1993) developed a variant of this technique, which is applicable in descriptive complexity theory to classes of finite relational structures of uniformly bounded degree. Variants of this result can also be found in Gaifman (1982) (see also Thomas (1991)). The essential content of this result, which is also called the Hanf-Sphere Lemma, is that two relational structures of bounded degree satisfy the same

first-order sentences of a certain quantifier-rank if both contain, up to a certain number m, the same number of isomorphism types of substructures of a bounded radius r.

In addition, a technique of model interpretability from Rabin (1965) (see also Arnborg *et al.* (1991), Seese (1992), Compton and Henson (1987) and Baudisch *et al.* (1982)) is adapted to descriptive complexity classes, and proved to be useful for reducing the case of an arbitrary class of relational structures to a class of structures consisting only of the domain and one binary irreflexive and symmetric relation, *i.e.*, the class of simple graphs. It is shown that the class of simple graphs is lintime-universal with respect to first-order logic, which shows that many problems on descriptive complexity classes, described in languages extending first-order logic for arbitrary structures, can be reduced to problems on simple graphs.

This paper is organized as follows: Section 2 introduces the basic terminology; in Section 3, linear time interpretability is introduced as a tool for reducing difficult structures to simpler ones; Section 4 introduces local r-types and handles the case of structures of bounded degree by reducing the general problem to a local investigation of r-types; finally, some open problems and remarks conclude the paper in Section 5.

Martin Kreidler prepared the LaTeX-version of this article. I wish to thank him for his efficient and diligent work. Furthermore, I thank him for his reading of several versions of the 'final' text. Moreover, I must express my gratitude to Wolfgang Thomas and his colleagues in Kiel, where I presented and discussed the first version of the main result. Finally, I have to thank C. Lautemann, S. Lindell, C. Schlepphorst, K. Wagner and the anonymous referees for their interesting remarks and comments, and especially for pointing out some mistakes of the preprint version of this article.

2. Definitions and conventions

This section is devoted to a brief introduction to the basic terminology, especially to the notion of first-order problems, and a useful model of linear-time computability. Notions from logic or complexity theory not introduced in this or the following sections are standard, and the reader is referred, *e.g.*, to Ebbinghaus *et al.* (1993) or Papadimitriou (1994).

A *finite signature* S for relational structures consists of a finite set of relation symbols $R_1, \ldots, R_s$, each with a fixed arity $r_i > 0$, and constant symbols $c_1, \ldots, c_t$, but without function symbols. An S-structure $\mathcal{G} = (A^{\mathcal{G}}, R_1^{\mathcal{G}}, \ldots, R_s^{\mathcal{G}}, c_1^{\mathcal{G}}, \ldots, c_t^{\mathcal{G}})$ consists of a nonempty set $A^{\mathcal{G}}$, the domain or universe, relations $R_i^{\mathcal{G}}$ over $A^{\mathcal{G}}$ of arity r_i (for each $i : 1 \leq i \leq s$) and elements $c_j^{\mathcal{G}}$ of $A^{\mathcal{G}}$ (for each $j : 1 \leq j \leq t$). $R_i^{\mathcal{G}}$ and $c_j^{\mathcal{G}}$ are also called an *interpretation* of R_i and c_j in $\mathcal{G}$. The elements of the domain of a structure are sometimes called *points*, or, in analogy with graphs, *vertices*. Moreover, we assume that $A^{\mathcal{G}}$ is the set $\{1, \ldots, n\}$ for a natural number n. This n will be denoted by the size $| A^{\mathcal{G}} |$ of $A^{\mathcal{G}}$. The set $\{1, \ldots, n\}$ will be denoted by $[n]$. For a structure $\mathcal{G}$, we will also denote its domain by $| \mathcal{G} |$. Furthermore, $| B |$ will also denote the number of elements of an arbitrary set B. It will be easy to distinguish this different use of the notation $| \ldots |$, since different types of letters for structures and sets will be used. An S-structure $\mathcal{G}$ is called finite when its domain is. Unless otherwise stated, throughout the rest of this article we make the

assumption that all structures under consideration will be finite. If S is a signature, let $STRUCT(S) = \{\mathcal{G} : \mathcal{G} \text{ is a finite } S\text{-structure}\}$.

For a subset B of $A^{\mathcal{G}}$ that contains all $c_j^{\mathcal{G}}$ (with $1 \leq j \leq t$), the *induced substructure* $\mathcal{G} \downarrow B$ is the structure $(B, R_1^{\mathcal{G}} \cap B^{r_1}, ..., R_s^{\mathcal{G}} \cap B^{r_s}, c_1^{\mathcal{G}}, \ldots, c_t^{\mathcal{G}})$. When B does not contain all $c_j^{\mathcal{G}}$ (with $1 \leq j \leq t$), the *induced substructure* $\mathcal{G} \downarrow B$ is defined as the structure $\mathcal{G} \downarrow B'$, with $B' := B \cup \{c_1^{\mathcal{G}}, \ldots, c_t^{\mathcal{G}}\}$. Individuals a and b from $A^{\mathcal{G}}$ are said to be *adjacent* by $R_i^{\mathcal{G}}$, if there are $x_1, \ldots, x_{r_i}$, such that $\left[(x_1, \ldots, x_{r_i}) \in R_i^{\mathcal{G}} \text{ and } a = x_j, b = x_k \text{ for some } j, k \leq r_i\right]$. We will sometimes call $(x_1, \ldots, x_{r_i})$ the $R_i^{\mathcal{G}}$*-edge* connecting a and b. a and b are said to be *adjacent in* $\mathcal{G}$ if there is a relation $R_i^{\mathcal{G}}$ such that a and b are adjacent by $R_i^{\mathcal{G}}$. In this case a is said to be *incident* with the corresponding edge $(x_1, ..., x_{r_i})$. The *degree* of an individual a is the cardinality of the set of individuals adjacent to a but not equal to a. A vertex of degree 0 is said to be *isolated*. The *degree of a structure* is the maximum of the degrees of its individuals. A sequence $x_0, \ldots, x_m$ is called a $\mathcal{G}$*-path*, (or simply a path if it is clear which structure is used), if for every $j < m$, x_j and x_{j+1} are adjacent. m is called the *length* of this path. The *distance* $d^{\mathcal{G}}(a, b)$ between a and b in $\mathcal{G}$ is the length of a shortest path from a to b in $\mathcal{G}$. For $r \geq 0$, the *r-neighbourhood* of a in $\mathcal{G}$, $N_r^{\mathcal{G}}(a)$, consists of all $b \in A^{\mathcal{G}}$ with $d^{\mathcal{G}}(a, b) \leq r$. The superscript $\mathcal{G}$ will be omitted whenever possible.

The language $L(S)$ of first-order logic for the signature S contains the relations and constant symbols from S: '=' denoting the equality relation; $\wedge$, $\vee$, $\neg$ denoting the logical connectives 'and', 'or' and 'not'; x, y, z, x_1, x_2, x_3,... denoting individual variables, running over the elements of the domain; and $\forall$, $\exists$ denoting symbols for the quantifiers 'for all' and 'there exists'. The first-order formulas for this language $L(S)$ are built as usual.

The *quantifier-rank* of a formula φ, denoted by $qr(\varphi)$, is defined by induction on the structure of formulas:

$$\begin{aligned} qr(\varphi) &:= 0 \quad \text{if } \varphi \text{ is atomic,} \\ qr(\varphi \wedge \psi) &:= max(qr(\varphi), qr(\psi)), \\ qr(\varphi \vee \psi) &:= max(qr(\varphi), qr(\psi)), \\ qr(\neg\varphi) &:= qr(\varphi), \\ qr(\forall x\, \varphi) &:= qr(\varphi) + 1, \\ qr(\exists x\, \varphi) &:= qr(\varphi) + 1. \end{aligned}$$

Let $\mathcal{G}_1$ and $\mathcal{G}_2$ be two S-structures and assume $n \in \mathbb{N}$ (throughout the paper $\mathbb{N}$ will denote the set of the natural numbers). We define the relation $\equiv_n$ between structures of the same signature by

$\mathcal{G}_1 \equiv_n \mathcal{G}_2$ if and only if for each $L(S)$-formula φ with $qr(\varphi) \leq n$,

$$\mathcal{G}_1 \models \varphi \text{ if and only if } \mathcal{G}_2 \models \varphi,$$

that is, in $\mathcal{G}_1$ and $\mathcal{G}_2$ the same formulas of quantifier-rank $\leq n$ hold. We will say that $\mathcal{G}_1$ and $\mathcal{G}_2$ are *n-equivalent* when $\mathcal{G}_1 \equiv_n \mathcal{G}_2$ holds. A proof of the following lemma can be found in Ebbinghaus *et al.* (1993).

Lemma 2.1. Let S be a finite signature for relational structures. Then $\equiv_n$ is an equivalence relation with finitely many equivalence classes, and for each equivalence class Γ there is an $L(S)$-formula η_Γ of quantifier-rank n such that

$$\Gamma = \{\mathcal{G} : \mathcal{G} \text{ is an } S\text{-structure and } \mathcal{G} \models \eta_\Gamma\}.$$

We will think of a *problem* or a *property* P as a set of S-structures for a signature S. P is *expressible* or *definable* by a formula φ from L(S) (L(S)-definable for short) if $P = \{\mathcal{G} : \mathcal{G}$ is a finite S-structure and $\mathcal{G} \models \varphi\}$. The latter set is also denoted by $MOD(\varphi)$. Let **FO** be the set of all problems definable by a first-order formula, that is, **FO** := $\{P$: there is a finite signature S and a first-order formula $\varphi \in L(S)$ with $P = MOD(\varphi)\}$. Similarly, we define **FO**(S) := $\{P$: there is a first-order formula $\varphi \in L$(S) with $P = MOD(\varphi)\}$. Sometimes we will not distinguish a problem and a formula defining it.

Aho, Ullman and Immerman proved the following theorem.

Theorem 2.1. (Aho and Ullman 1979; Immerman 1987) **FO** $\subset$ **L**.

Here, **L** is the set of problems decidable by a deterministic Turing machine in logarithmic space. Hence, each **FO**-problem can be solved in polynomial time.

It is the objective of this article to show that each **FO**-problem can be decided in linear time for structures of bounded degree. However, linear time is not such a robust notion as polynomial time, and there is no canonical definition of linear time in the literature, since it depends heavily on the model of computation used (see Grandjean (1993) and Grädel (1990)). For this reason, we need to define what is meant by linear time computable problems in this article. First, we have to define the size of a relational structure $\mathcal{G}$. For an S-structure $\mathcal{G} = (A^{\mathcal{G}}, R_1^{\mathcal{G}}, \ldots, R_s^{\mathcal{G}}, c_1^{\mathcal{G}}, \ldots, c_t^{\mathcal{G}})$ define $size(\mathcal{G}) := | A^{\mathcal{G}} | + \sum_{i=1}^{s} | R_i^{\mathcal{G}} | + t$. A *linear time algorithm* could then be characterized as an algorithm that needs for each input $\mathcal{G}$ at most $O(size(\mathcal{G}))$ elementary operations. A class K of graphs is said to be *linear time computable* if its membership problem can be solved by a linear time algorithm. If the degree of K is uniformly bounded, such an algorithm needs only $O(| A^{\mathcal{G}} |)$ time, *i.e.*, it is linear in the number of vertices.

To define elementary operations, we have to fix the data structure that will be used. For this purpose, it is not difficult to generalize the adjacency list representation (see Figure 4), which is one of the standard representations for graphs. Here, we have an entry for each individual of the domain with s pointers to the lists of the r_j-tuples with which the corresponding individual is incident. The r_j-tuples are represented as lists with pointers to the corresponding vertices, while the constants are marked by t additional entries with pointers to and from the corresponding individuals. We will also call this generalized adjacency list representation the *gal representation*. Later it will be shown that it is sufficient to restrict attention to graphs, *i.e.*, to structures with exactly one binary relation, hence the ordinary adjacency list representation (see, *e.g.*, Cormen *et al.* (1990)) will do, and we shall omit the details of this generalized adjacency list representation.

The number of registers needed to store this gal representation of a structure $\mathcal{G}$ is obviously $O(size(\mathcal{G}))$. Here, we assume that the name of each of the individuals fits into one register, *i.e.*, we follow the idea of the uniform cost measure (see Aho *et al.* (1974)).

Elementary operations are now all operations that can be performed in one step of time on this data structure (independent of the size of the structure under consideration). This can be made more specific by introducing a very special kind of RAM, the deterministic successor RAM with output (denoted as DSRAM, or simply SRAM, since only deterministic RAMs are considered in the following), which was essentially introduced

in Minsky (1967) under the name of program machines. It allows as arithmetic operations only the addition and subtraction of 1. Here we use the notation of Wagner and Wechsung (1986), which allows, in addition to these arithmetic operations, load and store commands using direct and indirect addressing and jumps depending on the outcome of a comparison of the content of two registers (using $=, \leq, <, \geq, >$). Furthermore, we extend it by a one-way output tape, to allow the calculation of functions as well as the ordinary accepting behaviour.

With this SRAM we can solve decision problems as well as calculate functions. Let **SLIN** be the set of all problems on relational structures that can be decided by an SRAM in time $O(size(\mathcal{G}))$, starting with the above gal representation of $\mathcal{G}$ in the first cells of the input register. Moreover, let **SLINFU**(S) be the set of all functions from $STRUCT(S)$ to $STRUCT(S)$ that are computable by an SRAM in time linear in the size of the input structure $\mathcal{G}$ in such a way that the gal representation of $\mathcal{G}$ is written in the first $size(\mathcal{G})$ cells of the input register, and the output structure $\mathcal{H}$ is also written in gal representation in the first cells of the output register.

It is not the goal of this article to investigate the relation of this model of linear time computation to other models: see Wagner and Wechsung (1986) for details (*e.g.*, Theorems 5.1, 5.2 and 20.11). Here, it serves only as an aid to give a precise specification of the notion of a linear time algorithm.

We conclude this section with some observations on these notions. It is obvious that the ordinary **LINTIME**, *i.e.*, the set of languages decidable by a deterministic Turing machine in linear time, is contained (up to a certain coding) in **SLIN**. Moreover, it is not difficult to see that the model is not too powerful. For that reason, we interpret the content of the registers as natural numbers. Let k be the largest number of an input register and assume that the input length, the number of nonempty input cells, is n. Then the largest number represented by an arbitrary register cell after the performance of $c * n$ steps is at most $c * n + k$. On the other hand, it is suitably designed for handling problems for relational structures, and especially graphs, since some of the well-known linear time computable graph functions can be computed on SRAMs in linear time.

Lemma 2.2. Let S be the signature for graphs. Assume that DFS or BFS are the functions computing for a graph one of its depth or breadth first search trees. Then DFS and BFS are in **SLINFU**(S). The problem to decide for a graph its connectedness is in **SLIN**.

To prove this lemma, one simply follows the standard linear time algorithms (see, *e.g.*, Cormen *et al.* (1990) and Mehlhorn (1984)), and observes that they can be performed on SRAMs in linear time. The details are left to the reader. Similarly, one can calculate strongly connected components, decide planarity and perform topological sort. It is useful to observe that functions computable in linear time by SRAMs are closed under composition.

Lemma 2.3. Let S be an arbitrary signature for relational structures. Then $f, g \in$ **SLINFU**(S) implies $f \circ g \in$ **SLINFU**(S).

This is obvious, since one can easily combine two SRAMs using the following idea: the main problem is the output of the first machine. But one can easily get round this problem by transforming the first SRAM in such a way that it uses each second register

instead of the cells of the output tape. This registers can then be used by the second SRAM transformed similarly as the 'input tape'. The details are left to the reader.

3. Interpretability

It is often easier to consider simple structures rather than complicated ones. For that reason, we present here a very simple reduction technique with the help of which it is possible to reduce the general case of arbitrary relational structures to simple graphs, *i.e.*, in this case to structures with exactly one binary irreflexive and symmetric relation. The proof could also be performed directly for arbitrary structures. But this technique seems to be of interest in its own right, since it is an adaptation of the method of interpretability, which has already found many applications in mathematical logic and complexity theory. The form of semantic interpretations goes back to Rabin (1965) (see also Rabin (1977) and Baudisch *et al.* (1982)), where it was introduced as a tool for deducing the decidability or undecidability of one formal theory from another formal theory. The method found applications in proofs of lower bounds for the complexity of theories (see Compton and Henson (1987)), it has been used as a tool to investigate complexity classes (see, *e.g.*, Dahlhaus (1983) and Creignou (1993)) and it has been applied to deducing polynomial and linear time algorithms for large classes of graph problems (see Arnborg *et al.* (1991) and Seese (1992)). The variant we will use here can be viewed as an extension of Rabin's semantic interpretations.

Let S_1, S_2 be two relational signatures, and let K_1, K_2 be sets (or classes) of S_1-, S_2-structures. Assume that there are $\mathrm{L}(S_2)$-formulas $\varepsilon(x_1, x_2), \alpha(x), \beta_i(x_1 \ldots, x_{r_i})$ (for each relational symbol R_i of S_1 of arity r_i, where $R_1, \ldots, R_s$ are assumed to be the relational symbols of S_1), and $\gamma_j(x)$ (for each constant c_j from S_1, where $c_1, \ldots, c_t$ are assumed to be the constants from S_1), which have only the indicated variables as free variables. For each structure $\mathcal{G} \in K_2$ and each $\mathrm{L}(S_2)$-formula $\varphi(x_1, \ldots, x_k)$, define $\mathcal{G}(\varphi)$ to be the k-ary relation that is defined by φ in $\mathcal{G}$, that is, the set of tuples $\{(a_1, \ldots, a_k) \ : \ a_i$ is in the domain of $\mathcal{G}$ for all $i : 1 \leq i \leq k$ and $\mathcal{G} \models \varphi[a_1, \ldots, a_k]\}$. We also require that $\mathcal{G}(\varepsilon)$ is an equivalence relation for every $\mathcal{G} \in K_2$. Moreover, assume that the relations $\mathcal{G}(\gamma_j)$ contain exactly one element $d_j^{\mathcal{G}}$, for which $\alpha(d_j^{\mathcal{G}})$ is true in $\mathcal{G}$, for every structure $\mathcal{G} \in K_2$.

We will denote such a sequence $I := (\alpha, \beta_1, \ldots, \beta_s, \gamma_1, \ldots, \gamma_t, \varepsilon)$ of $\mathrm{L}(S_2)$-formulas as an *interpretation*. For each structure $\mathcal{G} \in K_2$, define $\mathcal{G}^I$ to be the following structure: $(\mathcal{G}(\alpha), \mathcal{G}(\beta_1), \ldots, \mathcal{G}(\beta_s), d_1^{\mathcal{G}}, \ldots, d_t^{\mathcal{G}})/\mathcal{G}(\varepsilon)$, where $/\mathcal{G}(\varepsilon)$ denotes the factorization by the equivalence relation $\mathcal{G}(\varepsilon)$.

$\mathcal{G}^I$ is obviously a structure for $\mathrm{L}(S_1)$, but not necessarily a structure in K_1. Examples of such interpretations can be found in Rabin (1965) or Arnborg *et al.* (1991). Let $f : \mathbb{N} \mapsto \mathbb{N}$ be a function over the natural numbers.

Definition 3.1. K_1 is said to be *f-interpretable* into K_2 with respect to $\mathrm{L}(S_1)$, $\mathrm{L}(S_2)$ if there is an interpretation I as above and an algorithm $\mathcal{A}$ that constructs for each $\mathcal{G} \in K_1$ (that is, the corresponding gal) a structure $\mathcal{A}(\mathcal{G}) \in \mathcal{K}_2$ (that is, the corresponding gal) with time complexity $\in \mathrm{O}(f)$ such that $\mathcal{G} \cong (\mathcal{A}(\mathcal{G}))^I$. When f is a linear (or, respectively, polynomial) function K_1 is said to be lintime (respectively, polytime) interpretable into K_2, respectively.

The following technical lemma is useful in applications of this technique.

Lemma 3.1. Lintime and polytime interpretability are transitive.

The proof is a straightforward application of the definitions, since the required interpretation results simply from substituting the atomic formulas of the defining formulas for the first interpretation by the corresponding defining formulas of the second interpretation. The algorithmic part follows from Lemma 2.3.

With the help of interpretations it is not only possible to transform S_2-structures into S_1-structures, moreover, there is a canonical way to translate $L(S_1)$-formulas φ into $L(S_2)$-formulas φ^I. This translation is defined by induction on the structure of the formulas. We first define a translation $()^*$, which translates formulas without paying attention to the constants from S_1. Then they are eliminated in the last step.

$$
\begin{aligned}
(x_1 = x_2)^* &:= \varepsilon(x_1, x_2) \\
\left(R_i(x_1, \ldots, x_{r_i})\right)^* &:= \exists y_1 \ldots \exists y_{r_i} \left(\bigwedge_{1 \le i \le r_i} \varepsilon(x_i, y_i) \wedge \beta_i(y_1, \ldots, y_{r_i}) \right) \\
&\quad \text{for } 1, \ldots, s, \text{ that is, for each relational symbol from } S_1 \\
(\neg\varphi)^* &:= \neg(\varphi^*) \\
(\varphi \wedge \psi)^* &:= \varphi^* \wedge \psi^* \\
(\exists x \varphi)^* &:= \exists x(\alpha(x) \wedge \varphi^*)
\end{aligned}
$$

Then define $\varphi^I := \exists y_1 \ldots \exists y_t \left(\varphi^*[y_1/c_1, \ldots, y_t/c_t] \wedge \gamma_1(y_1) \wedge \ldots \wedge \gamma_t(y_t) \right)$, where $\varphi^*[y_1/c_1, \ldots, y_t/c_t]$ results from substituting the constants c_j by y_j, while the variables y_j are assumed to be different from all the other variables in φ^*.

The following result was essentially proved by Rabin in Rabin (1965). The only new part is that we now also consider the complexity of the transformation of the structures.

Theorem 3.1. Assume K_1 in $\mathbf{FO}(S_1)$ and K_2 in $\mathbf{FO}(S_2)$ for relational signatures S_1 and S_2. If K_1 is lintime (polytime) interpretable into K_2 by an interpretation I over $L(S_1)$, $L(S_2)$, then each $L(S_1)$-problem φ can be solved over K_1 in linear (polynomial) time if the $L(S_2)$-problem φ^I can be solved over K_2 in linear (polynomial) time, respectively. Hence, all **FO** problems over K_1 can be solved in linear (polynomial) time if all **FO** problems over K_2 can be solved in linear (polynomial) time, respectively.

The last statement also holds for all extensions of first-order logic. The proof of this result is exactly the same as in Rabin (1965), with the additional observation that the time to perform the transformations is linear or polynomial, respectively. Its main idea comes from the following lemma from Rabin.

Lemma 3.2. (Rabin 1965) Under the above assumptions, for each $\mathcal{G} \in K_2$ and an arbitrary $L(S_1)$-formula φ : $\mathcal{G} \models \varphi^I \Leftrightarrow \mathcal{G}^I \models \varphi$.

The theorem follows by a combination of this lemma and the definition of f-interpretability. The details are left to the reader, since the proof is literally the same as in Rabin (1965).

Corollary 3.1. Assume that S_1 and S_2 are signatures for relational structures. Let $K_1 \in \mathbf{FO}(S_1)$ and $K_2 \in \mathbf{FO}(S_2)$ be sets of structures such that K_1 is lintime interpretable into K_2 with respect to $L(S_1)$, $L(S_2)$. Then $K_2 \in \mathbf{SLIN}$ implies $K_1 \in \mathbf{SLIN}$.

Following an idea of Hauschild and Rautenberg (1971) for the interpretrability of theories we use the following definition.

Definition 3.2. Assume that K is an arbitrary class of structures, not necessarily belonging to a fixed signature. Let S_1 be a signature and K_1 a set of S_1-structures. K_1 is said to be *lintime- (polytime-) universal* with respect to K if for each signature S_2, each set of S_2-structures K_2, with $K_2 \subseteq K$ is lintime- (polytime-) interpretable into K_1. When K is the class of all structures, we use lintime- (polytime-) universal instead of lintime- (polytime-) universal with respect to K.

If K_1 is lintime-, polytime-universal with respect to K, each first-order problem over a set $K_2 \subset K$ of structures can be reduced to an equivalent problem over K_1.

Definition 3.3. Let SG be the class of finite simple graphs (considered as relational structures), *i.e.*, the class of relational structures with exactly one binary relation, which is assumed to be irreflexive and symmetric, and that do not contain other relations or constants. For an arbitrary natural number $d \geq 1$, let SG_d be the class of all members of SG of degree $\leq d$. Moreover, let BD_d be the class of arbitrary relational structures of degree $\leq d$.

Theorem 3.2. The class SG is lintime-universal.

The proof follows the same line as the proof of the universality of the theory of irreflexive-symmetric graphs from Hauschild and Rautenberg (1971) (see Rabin (1965) and Church and Quine (1953) also). One has to observe only that the corresponding transformations can be performed in linear time. The first step is the transformation of arbitrary structures to relational structures. Here, one simply transforms the functions into the corresponding relations. The problem of coding an arbitrary structure as input to our SRAM will be left to the reader, since the main focus of this paper is relational structures.

To illustrate the basic idea for relational structures, we consider the example of a relational structure $\mathcal{G} = (A, R)$ with exactly one ternary relation R and no isolated vertices. In this case the structure $\mathcal{H} := \mathcal{A}(\mathcal{G})$ is constructed in the following way: its vertices are the vertices of A together with some additional vertices; for each vertex a a new vertex a' is added, and is connected to a by an edge of $\mathcal{H}$; each entry $(a, b, c) \in R$ is transformed into the finite structure shown in Figure 1.

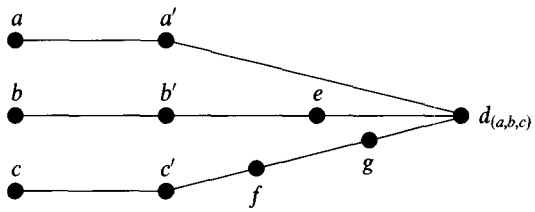

Fig. 1. A structure representing $(a, b, c) \in R$. The index (a, b, c) is omitted from e, f, g.

Here, the vertices $e_{(a,b,c)}, f_{(a,b,c)}, g_{(a,b,c)}$ and $d_{(a,bc)}$ are new vertices. The idea is demonstrated in Figure 2, where the graph $\mathcal{H} = \mathcal{A}(\mathcal{G})$ for the structure

$$\mathcal{G} = (A = \{a, b, c, d\}, R = \{(a, b, c), (b, c, d)\})$$

is presented.

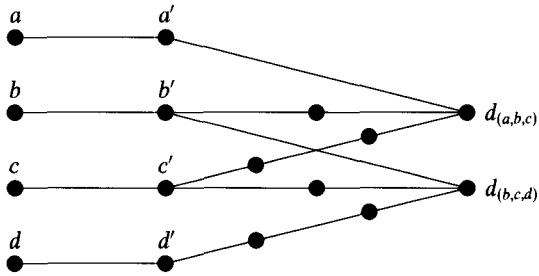

Fig. 2. A structure representing $(\{a,b,c,d\},\{(a,b,c),(b,c,d)\})$

The resulting transformation algorithm $\mathscr{A}$ can be performed in linear time, since each entry of the relation R is transformed once and only once to a finite part of the new structure, and all these parts are isomorphic.

The formula $\alpha(x)$ selecting the original vertices is a formula expressing 'the degree of x is 1'. Before we can define the formula $\beta(x,y,z)$, which defines the ternary relation, we have to find a formula $\sigma(x)$, which defines the set of all vertices $\{d_{(a,b,c)} \mid (a,b,c) \in R\}$. We choose a formula expressing: 'x has degree 3 and there are three vertices a,b,c of degree 1 such that there is a path from x to a of length 2, there is a path from x to b of length 3 and there is a path from x to c of length 4'. This is obviously expressible in first-order logic. Now $\beta(x,y,z)$ can be defined as a formula expressing: 'the vertices x,y,z have degree 1 and there exists a vertex u with $\sigma(u)$'. Choose as $\varepsilon(x,y)$ the formula $x=y$. The proof of $\mathscr{G} \cong (\mathscr{A}(\mathscr{G}))^I$ is easy. The existence of the vertex $d_{(a,b,c)}$ together with the structure of Figure 1 corresponds exactly to $(a,b,c) \in R$, for all vertices a,b,c of $\mathscr{G}$.

The case of structures $\mathscr{G}$ with isolated vertices can be treated as follows. One does not change the isolated vertices and changes all other parts as explained above. Define the new $\alpha(x)$ as a formula expressing 'x has degree 0 or 1'. All other formulas are the same as above.

Theorem 3.3. The class SG_3 is lintime-universal with respect to BD_d.

The first step in the proof of this result is the same construction as above. Here, one can observe that the degree of the structure $\mathscr{A}(\mathscr{G})$ is bounded if the degree of $\mathscr{G}$ is bounded. So by Lemma 3.1 it is sufficient to regard structures with one binary relation of bounded degree. But it is not difficult to find an interpretation of the class of these structures into the class of graphs of degree 3. We give an illustration in Figure 3.

The idea is to substitute each vertex a of degree d by a cycle of $d+2$ vertices, with one vertex a' of degree 1 attached to it. Isolated vertices are excluded from this construction. They are treated in the same way as above. d vertices of this cycle are used for (at most d) connections to other cycles, representing other vertices. The attached vertex a' is used to represent the original vertex a. Edges are represented as paths of length $d+1$ between cycles. The new vertices of degree 1 can be recognized, since the cycles to which they are attached can also be recognized. These cycles can be recognized, since their length is $d+2$, and there are no other cycles of this length.

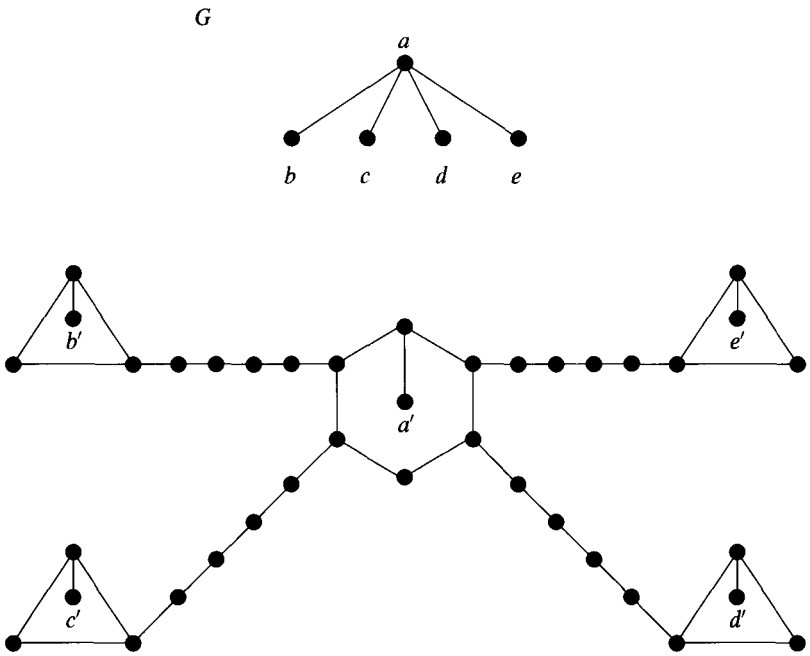

Fig. 3. A graph $\mathcal{G}$ and its representation by a graph of degree ≤ 4

Remark 3.1. It is not possible to prove this result for SG_2, since the structures in SG_2 are disjoint unions of paths and cycles, and it is not possible to define arbitrary linear orderings in a uniform way (given by the desired interpretations).

To illustrate the idea, assume that I is an interpretation of the class of linear orderings into SG_2. Let $\mathcal{A}$ be the algorithm corresponding to I, and assume that $\mathcal{G}$ is a sufficiently large linear ordering. Assume that f is an isomorphism corresponding to $\mathcal{G} \cong (\mathcal{A}(\mathcal{G}))^I$. The connected components of $\mathcal{A}(\mathcal{G})$ are paths or cycles. Let B be the set $\{a \mid \mathcal{A}(\mathcal{G}) \models \alpha[a]\}$. One has to distinguish two cases. Assume in the first case that there are two vertices a and b in B that lie in the same component of $\mathcal{A}(\mathcal{G})$ such that $f^{-1}(a) < f^{-1}(b)$. Since a and b are in the same component, there is an automorphism of $\mathcal{A}(\mathcal{G})$ mapping a to b and leaving the vertices of all other components unchanged. But then the formulas of the interpretation are unable to distinguish between a and b. Hence one also gets $f^{-1}(a) > f^{-1}(b)$, contradicting the above assumption.

In the other case, assume that m' is the quantifier rank of the formula defining $<$. Here, one shows the existence of two vertices a and b in different components of $\mathcal{A}(\mathcal{G})$ such that the extension of the structure $\mathcal{A}(\mathcal{G})$ by the vertices a and b is m'-equivalent to the structure $\mathcal{A}(\mathcal{G})$ extended by b and a (the order of a and b is essential here). Here, Ehrenfeucht–Fraïssé Games (see Ebbinghaus *et al.* (1993)) are used. The idea is based on the fact that two cycles or two paths are m'-equivalent if they are large enough.

4. Bounded degree and local properties

For an arbitrary class K of relational structures, define $\mathbf{FO}\downarrow K := \{P \cap K : P \in \mathbf{FO}\}$. Now the main result can be formulated.

Theorem 4.1. For each natural number $d \geq 1$,

$$\mathbf{FO} \downarrow BD_d \subseteq \mathbf{SLIN}.$$

For $d = 1$ the result is almost trivial, and for $d = 2$ it is easy for the simplicity of the structures in BD_1 and BD_2, since in this case even monadic second-order problems can be solved in linear time. But it is not necessary to single out these cases, since it is easy to observe that if the statement holds for an arbitrary d, it also holds for all $d' \leq d$, since $BD_{d'} \subseteq BD_d$. Hence, by Theorem 3.3 and Corollary 3.1, it is sufficient to prove $\mathbf{FO}\downarrow SG_3 \subseteq \mathbf{SLIN}$. The idea used to prove this could also be used to prove Theorem 4.1 directly, but the technical details are easier in the special case of simple graphs. One of the ingredients in the proof is a technique of Hanf (Hanf 1965), which was developed to prove the elementary equivalence of infinite structures, which was adapted to finite structures and brought to the attention of the computer science audience by Fagin, Stockmeyer and Vardi (Fagin *et al.* 1993) (see also Thomas (1991) and Giammarresi *et al.* (1993) for other applications).

Let $\mathcal{G}$ be a relational structure, a be an element in the domain of $\mathcal{G}$, and r be an arbitrary natural number. Hanf defined the r-type of a in $\mathcal{G}$ to be the isomorphism type of $(\mathcal{G} \downarrow N_r^{\mathcal{G}}(a), a)$, where $(\mathcal{G} \downarrow N_r^{\mathcal{G}}(a), a)$ is the restriction of $\mathcal{G}$ to the r-neighbourhood of a in $\mathcal{G}$, where a is designated as the value of a new constant, which is called the *centre*. More precisely, individuals a_1 and a_2 in the domain of two structures $\mathcal{G}_1$ and $\mathcal{G}_2$, respectively, have the same *r-type* if $\mathcal{G}_1 \downarrow \mathcal{N}_r^{\mathcal{G}_1}(a_1) \cong \mathcal{G}_2 \downarrow \mathcal{N}_r^{\mathcal{G}}(a_2)$ under an isomorphism mapping a_1 to a_2. r will be called the *radius* of the r-type.

A structure representing such a type is called an *S-structure of radius r*. Let $r \in \mathbb{N}$ and $m \geq 1$ be given, and assume that S is a signature for relational structures. Following Fagin, Stockmeyer and Vardi (Fagin *et al.* 1993) we define: S-structures $\mathcal{G}_1$ and $\mathcal{G}_2$ are said to be (r, m)-*equivalent* if and only if for every r-type τ, either $\mathcal{G}_1$ and $\mathcal{G}_2$ have the same number of individuals with r-type τ, or both have at least m individuals with r-type τ. Hence, $\mathcal{G}_1$ and $\mathcal{G}_2$ are (r, m)-equivalent if they have both the same number of individuals with r-type τ, where we can count only as high as m. The following result is due to Fagin, Stockmeyer and Vardi.

Theorem 4.2. (Fagin *et al.* 1993) Let n and d be positive integers. There are positive integers r, m, where r depends only on n, such that whenever $\mathcal{G}_1$ and $\mathcal{G}_2$ are (r, m)-equivalent structures of degree at most d, then $\mathcal{G}_1 \equiv_n \mathcal{G}_2$.

Normally, an arbitrary **FO**-problem has to be considered as a global problem, *i.e.*, a problem that can only be decided by simultaneously examining different locations of the structure under consideration, which are usually far away from each other. Theorem 4.2 is one of the essential ingredients of the proof of our Theorem 4.1, since it enables us to reduce an arbitrary **FO**-problem to a *local* problem, *i.e.*, a problem that can be decided by *visiting once* each vertex of the structure and looking only at its neighbourhood to a certain *fixed radius*.

Proof of Theorem 4.1: We split the proof into two parts. First, we collect all the necessary conditions to formulate the algorithm and then we prove that the algorithm has an SRAM implementation working in linear time.

Assume that φ is an L(S)-formula for a signature S consisting of only one binary relation, *i.e.*, a signature for SG. Let n be the quantifier-rank of φ and let d be a natural number. d will be an upper bound to the degree of the structures under consideration. Let Γ be an equivalence class of $\equiv_n$ (see Lemma 2.1). If φ holds in one of the structures $\mathcal{G}$ from Γ, that is, $\mathcal{G} \models \varphi$, then it holds in them all, since $qr(\varphi) \leq n$. In this case we will say that φ holds in Γ. Let $\Gamma_1, \ldots, \Gamma_{k_1}$ be an enumeration of all equivalence classes for $\equiv_n$ in which φ holds, and let $\Lambda_1, \ldots, \Lambda_{k_2}$ be an enumeration of all equivalence classes for $\equiv_n$ in which φ does not hold. Then $\Gamma_1, \ldots, \Gamma_{k_1}, \Lambda_1, \ldots, \Lambda_{k_2}$ is a complete enumeration of all equivalence classes of $\equiv_n$.

Now, choose r and m as in Theorem 4.2 for the above n and d. There is only a finite number of r-types for S-structures of degree $\leq d$, since there is a uniform bound for the number of vertices. Let $\tau_1, \ldots, \tau_p$ be a complete enumeration of all possible r-types for S-structures. Each of these r-types τ_i can be represented by a structure $\mathcal{H}_i$. One should notice that these structures are almost S-structures, since they have one constant additional to the signature, the centre. Hence, the enumeration $\tau_1, \ldots, \tau_p$ can be effectively determined by investigating all possible S-structures of radius $\leq r$. Now, two S-structures $\mathcal{G}_1$ and $\mathcal{G}_2$ are (r,m)-equivalent if and only if for all i, with $1 \leq i \leq p$ the following holds:

$$\Big(\; |\{a \;:\; a \in |\, \mathcal{G}_1 \,|\text{ and } (N_r^{\mathcal{G}_1}(a), a) \text{ is of } r\text{-type } \tau_i\}| \;=\; |\{b \;:\; b \in |\, \mathcal{G}_2 \,|\text{ and } (N_r^{\mathcal{G}_2}(b), b) \text{ is of } r\text{-type } \tau_i\}| \; \Big)$$

or

$$\Big(\; |\{a \;:\; a \in |\, \mathcal{G}_1 \,|\text{ and } (N_r^{\mathcal{G}_1}(a), a) \text{ is of } r\text{-type } \tau_i\}| \geq m \text{ and } |\{b \;:\; b \in |\, \mathcal{G}_2 \,|\text{ and } (N_r^{\mathcal{G}_2}(b), b) \text{ is of } r\text{-type } \tau_i\}| \geq m \; \Big)$$

For an arbitrary S-structure $\mathcal{G}$ define $\varrho(\mathcal{G})$ to be the p-tuple $(\varrho_1, \ldots, \varrho_p)$ by

$$\varrho_i := \begin{cases} |\{a \;:\; a \in |\, \mathcal{G} \,|\text{ and } (N_r^{\mathcal{G}}(a), a) \text{ is of } r\text{-type } \tau_i\}| & \text{if this number is } < m \\ m & \text{otherwise} \end{cases}$$

Using this idea, we get that two S-structures $\mathcal{G}_1$ and $\mathcal{G}_2$ are (r,m)-equivalent if and only if $\varrho(\mathcal{G}_1) = \varrho(\mathcal{G}_2)$. Obviously, (r,m)-equivalence is an equivalence relation on the set of S-structures. But the number of such p-tuples $\tau(\mathcal{G})$ is bounded by $(m+1)^p$, since only $0, 1, \ldots, m$ appear as components. Hence, there are at most $(m+1)^p$ equivalence classes of S-structures for the (r,m)-equivalence relation. Each such (r,m)-equivalence class can be represented by a p-tuple $\tau(\mathcal{G})$ for a suitable S-structure $\mathcal{G}$. Note that not all such p-tuples can represent a real S-structure, since some may not be realizable. By Theorem 4.2, the equivalence relation for (r,m)-equivalence is a subdivision of the equivalence relation for $\equiv_n$, that is, (r,m)-equivalence is a refinement of $\equiv_n$. Let $\mathcal{U}_1, \ldots, \mathcal{U}_q$ with $q \leq (m+1)^p$ be a maximal system of pairwise not (r,m)-equivalent S-structures, that is, a complete system of representatives for each (r,m)-equivalence class. Now select those of these

representatives $\mathcal{U}_i$ in which φ holds. Assume that $\mathcal{V}_1, \ldots, \mathcal{V}_{q_1}$ is a complete system of representatives for these. The structures $\mathcal{V}_1, \ldots, \mathcal{V}_{q_1}$ can be chosen as representatives for the $(\equiv_n)$—equivalence classes $\Gamma_1, \ldots, \Gamma_{k_1}$, where some of these classes have many representatives among the structures $\mathcal{V}_1, \ldots, \mathcal{V}_{q_1}$, since (r,m)-equivalence refines $(\equiv_n)$. Such a system can be effectively determined if $\mathcal{U}_1, \ldots, \mathcal{U}_q$ is given. The result will then follow from statements (*) and ($\diamond$) below.

(*) For an arbitrary S-structure $\mathcal{G}$, we have $\mathcal{G} \models \varphi$ if and only if there is an i with $1 \leq i \leq q_1$ such that $\mathcal{G}$ and $\mathcal{V}_i$ are (r,m)-equivalent.

To prove (*), assume first that $\mathcal{G} \models \varphi$. There exists a j with $1 \leq j \leq q$ such that $\mathcal{G}$ is (r,m)-equivalent to $\mathcal{U}_j$, since $\mathcal{U}_1, \ldots, \mathcal{U}_q$ is a complete system of representatives for (r,m)-equivalence. By Theorem 4.2, we have that $\mathcal{U}_j \models \varphi$. But then $\mathcal{U}_j$ is one of the $\mathcal{V}_i$ for a certain i with $1 \leq i \leq q_1$. For the other direction, assume that i with $1 \leq i \leq q_1$ is chosen such that $\mathcal{G}$ and $\mathcal{V}_i$ are (r,m)-equivalent. By Theorem 4.2, we have $\mathcal{V}_i \equiv_n \mathcal{G}$ and hence $\mathcal{G} \models \varphi$, since $qr(\varphi) \leq n$ and $\mathcal{V}_i \models \varphi$, which proves (*).

($\diamond$) For an arbitrary S-structure G, we have $\mathcal{G} \models \varphi$ if and only if there is an i with $1 \leq i \leq q_1$ such that $\varrho(\mathcal{G}) = \varrho(\mathcal{V}_i)$.

This is a simple combination of the above statement on ϱ and (*).

Now we are able to formulate the algorithm. Its input is an S-structure $\mathcal{G}$ of degree $\leq d$. The idea is to visit once each vertex of $\mathcal{G}$, and check the r-type of $N_r^{\mathcal{G}}(a)$. The number of r-types occurring is counted up to m. For this reason, we use a vector $\mu := (\mu_i)_{1 \leq i \leq p}$, where p is the number of all possible r-types and μ_i is the number of all vertices a for which $N_r^{\mathcal{G}}(a)$ has the r-type τ_i. At the start μ is initialized by setting all $\mu_i := 0$. After all vertices of $\mathcal{G}$ are visited, we check whether μ is an element of $\{\varrho(\mathcal{V}_1), \ldots, \varrho(\mathcal{V}_{q_1})\}$. If this is the case, φ is satisfied in $\mathcal{G}$. In the other case it is not satisfied.

FOR $i := 1$ **TO** p **DO** $\mu_i := 0$;
$\mu := (\mu_i)_{1 \leq i \leq p}$
FOR each vertex $a \in | \mathcal{G} |$ **DO**
  determine $N_r^{\mathcal{G}}(a)$;
  FOR $i := 1$ **TO** p **DO**
    IF [$N_r^{\mathcal{G}}(a)$ has r-type τ_i and $\mu_i \leq m$] **THEN** $\mu_i := \mu_i + 1$
  END (* FOR *)
  IF $\mu \in \{\varrho(\mathcal{V}_1), \ldots, \varrho(\mathcal{V}_{q_1})\}$ **THEN** print "$\mathcal{G} \models \varphi$" **ELSE** print "$\mathcal{G} \models \neg\varphi$"
END (* FOR *).

The rest of the proof of Theorem 4.1 follows from the next Lemma.

Lemma 4.1. The above algorithm is correct and has a linear time implementation on an SRAM.

The correctness of this algorithm follows immediately from ($\diamond$). It remains to prove that it can be performed on an SRAM in time linear in $size(\mathcal{G})$. First we show that there is an ordinary linear time algorithm, *i.e.*, an algorithm using a linear number of elementary operations. For that reason, assume that $\mathcal{G}$ is given in gal representation. First, note that the size of an arbitrary S-structure of radius $\leq r$ and degree $\leq d$ is uniformly bounded for fixed d and r. Hence, for each vertex a of $\mathcal{G}$ the size of $N_r^{\mathcal{G}}(a)$ is uniformly bounded (*i.e.*, independent of the input $\mathcal{G}$) for S-structures $\mathcal{G}$ of degree $\leq d$. In addition to this, the size of each of the structures $\mathcal{V}_1, \ldots, \mathcal{V}_{q_1}$ is uniformly bounded. Moreover, p and q_1 is fixed and independent of $\mathcal{G}$.

A typical step of the algorithm is to choose a vertex a from the list of all vertices of the gal representation of $\mathcal{G}$ to calculate $N_r^{\mathcal{G}}(a)$ and to determine which r-type τ_i it has. But even this can be done in constant time (independent of $\mathcal{G}$, a and i), since all those structures under consideration have a uniformly bounded size. If τ_i is known, one has to check whether $\mu_i \leq m$ holds. In the positive case one sets $\mu_i := \mu_i + 1$. But both can be done in constant time. Therefore, the time for each step (for each new vertex a) of the main loop of the algorithm can be bounded by a uniformly chosen constant. But each vertex a (that is, together with its neighbourhood) is considered exactly once, hence the main part of the algorithm works in linear time.

If each vertex a of $\mathcal{G}$ is handled as above, it remains only to test the vector μ for membership in the set $\{\varrho(\mathcal{V}_1), \ldots, \varrho(\mathcal{V}_{q_1})\}$. But this test can also be performed in constant time, since each component of the vector $\varrho(\mathcal{V}_i)$ is $\leq m$. Hence, the time needed for the complete algorithm is linear in $size(\mathcal{G})$.

Now, some ideas have to be given concerning the transformation of the algorithm to real SRAM code. At first, it is necessary to code finite graphs of bounded degree by the content of the registers of our SRAM in such a way that the operations needed by our algorithm can be performed by the operations of the SRAM.

For this purpose we store the gal representation of a graph G as a block of consecutive registers. Each register contains exactly one vertex, numbered $1, \ldots, n$, the address of another register or 0. The idea is illustrated by Figures 4 and 5.

Here we give the gal representation of a simple path of two vertices and the content of the first twelve registers storing it. The representation of the gal representation in the registers is organized as follows. It consists of subblocks representing a 'line' of the gal representation, denoted as *line blocks*. Each subblock consists of smaller subblocks, the *vertex blocks*, built from three consecutive registers with numbers $3i + 1, 3i + 2, 3i + 3$. Each line-bock starts with a vertex block representing a vertex j, and all following vertex blocks represent vertices adjacent to j. The first vertex block of a line block represents in its first register the number of its vertex j. Its second register stores the address of the first register of the next line block, *i.e.*, the first register of the line block for vertex $j + 1$ and its adjacent vertices. Its third register stores the address of the first register of the next vertex block of the same line block, *i.e.*, the vertex block representing the first vertex (in the order given by the corresponding 'line' of the gal representation) adjacent to j.

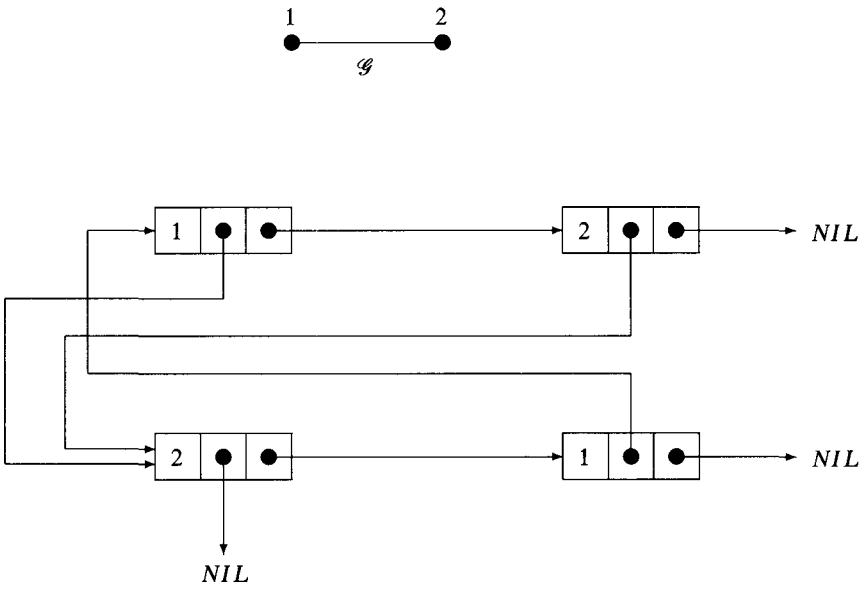

Fig. 4. A graph $\mathcal{G}$ and its gal representation

1	2	3	4	5	6	7	8	9	10	11	12
1	7	4	2	7	0	2	0	10	1	1	0

Fig. 5. The register code of the gal representation of $\mathcal{G}$

The vertex block of an adjacent vertex, *i.e.*, one, that is not in the first position of a line block, stores in its first position the number of the corresponding adjacent vertex, say j_1, and in its second position, the address of the first register of the line block related to j_1. The third register contains the address of the first register of the vertex block of the next adjacent vertex of the line block corresponding to j. The pointer to NIL is indicated by the entry 0. Let us call this representation the *register code* of this gal representation of the graph under consideration.

The algorithm has only two essential parts, the test of the r-types of $N_r^{\mathcal{G}}(a)$ and the test of whether μ is an element of $\{\varrho(\mathcal{V}_1),\ldots,\varrho(\mathcal{V}_{q_1})\}$. The first test can be made quite easy by appropriate preprocessing. Here, the first problem is how to store the r-types τ_i. We know that the size of an arbitrary representative of τ_i is uniformly bounded. Hence, we can assume that there is a number k' such that all possible representatives have at most k' vertices. So it can be assumed that the set of their vertices is a subset of $\{1,\ldots,k'\}$. But there is only a bounded number of such structures, even if we allow arbitrary permutations of the names of the vertices. For each such representative of an r-type we now choose all possible gal representations. But their number is uniformly bounded as well. Let $\xi_1,\ldots,\xi_{q_2}$ for a natural number q_2 be a complete enumeration of

all possible gal representations of all possible representatives of all r-types $\tau_1, \ldots, \tau_p$ with vertices from $\{1, \ldots, k'\}$. This enumeration depends only on r and d, and not on the input $\mathcal{G}$. But r and d are fixed. Hence, we can instruct our program to write at the beginning of the calculation the register code of each ξ_i into a large block of registers. Assume that two different register codes, ξ_i and ξ_{i+1}, are separated by three registers with entry 0 followed by a register storing the number j, if the r-type of the left register code ξ_i is τ_j, which is followed again by three registers with entry 0. This kind of separation is used to recognize the r-type represented by ξ_i.

Assume now that we want to determine the r-type of $N_r^{\mathcal{G}}(a)$ for a certain vertex a. To begin with, we have to determine the register representation of a gal representation of $N_r^{\mathcal{G}}(a)$. We obtain this by gathering the corresponding information in the register representation of the gal representation of $\mathcal{G}$. This needs not more than $c * d^r$ steps for a constant c, if the address of the vertex a in the input register is known. But this is the case, since we regard each vertex successively. We store this representation in another block of registers. If the numbers of the vertices of this representation are chosen from $\{1, \ldots, k'\}$, our SRAM could simply compare this block with the blocks representing all the ξ_i. To do this, one only has to add a constant number (the absolute difference of the number of the first registers from both blocks) to the content of all those registers storing addresses of registers. If the changed block equals the block representing ξ_i, and ξ_i is a representative of the r-type τ_j, then $N_r^{\mathcal{G}}(a)$ has r-type τ_j. Obviously, the r-type does not change if the numbers of the vertices are renamed. But it is easy for our SRAM to carry out such a renaming. For this reason, one has only to start a list containing on one side all numbers of vertices appearing in the register representation, and on the other, some of the numbers $1, \ldots, k'$ (as many as needed). But this can be done in constant time, since the size of all possible such blocks is uniformly bounded. Hence, the number of the r-type of $N_r^{\mathcal{G}}(a)$ can be determined for each vertex a in constant time by an SRAM. But the adaptation of the vector μ can also be done in constant time. Hence, the first part of the algorithm has an implementation on an SRAM working in linear time.

It remains to consider the test of whether μ is an element of $\{\varrho(\mathscr{V}_1), \ldots, \varrho(\mathscr{V}_{q_1})\}$. But all the vectors $\varrho(\mathscr{V}_1), \ldots, \varrho(\mathscr{V}_{q_1})$ and q_1 depend only on φ, d, m, r, but not on $\mathcal{G}$. Each vector can be represented by a finite block of registers of uniformly bounded size. Hence, this part of the algorithm has an implementation on an SRAM working in constant time.

This finishes the proof of Lemma 4.1, and thus also the proof of Theorem 4.1.

Remark 4.1. It is not difficult to see that a similar result can be proved if instead of the uniform cost measure the logarithmic cost measure is considered. But then one has to measure the time complexity of the algorithm in the logarithmic size of the input structure $\mathcal{G}$. Here, the time needed for the calculations becomes longer by a factor of $\log n$. But this does not matter, since the size of $\mathcal{G}$ also becomes larger by a multiplicative factor $\log n$, where n is the number of individuals of $\mathcal{G}$.

The result is surprising, since at first sight one would expect a polynomial time algorithm to decide for a given (arbitrary) structure the truth of a given (but fixed) formula with a large block of nested quantifiers, since it seems natural that one has to run the search for every single quantifier in the formula under consideration.

The result shows the usefulness of Theorem 4.2, which can be regarded as a general principle for reducing global (first-order) problems to local problems (on neighbourhoods).

5. Concluding remarks

Obviously, there are linear time computable problems that are not expressible in first-order logic. One of those problems is for instance EVEN, the problem of determining whether the domain of a given structure has even cardinality, or the problem of the connectedness of a graph.

The algorithm presented in this article works in linear time, but it is not practical, since the 'constant factors' grow exponentially. But as long as the **P≠NP**–problem is undecided, there is no hope of avoiding this disadvantage.

Theorem 5.1. If **P≠NP**, there is no polynomial time algorithm to solve the above problem whose hidden constants are polynomially bounded.

This can easily be seen by coding the satisfiability problem of an arbitrarily given propositional formula with n propositional variables into a first-order formula with n existential quantifiers in the signature with only one unary relation, denoted, for example, by Q. Each propositional variable p then corresponds to $Q(x_p)$, and each negated propositional variable $\neg q$ corresponds to $\neg Q(x_q)$, where x_p and x_q are new individual variables. The propositional connectives $\neg$, $\vee$ and $\wedge$ are translated by themselves. The newly introduced individual variables x_p (for each propositional variable of the original proposition) are quantified at the beginning of the transformed formula by existential quantifiers ($\exists x_q$). The original propositional formula is satisfiable if the transformed first-order formula is true in a very simple special model, which has $2n$ isolated vertices, from which n are in the unary relation and n are not. The truth of the transformed formula in the special model could be decided in polynomial time when the hidden constants in the algorithm are polynomially bounded in the size of the input structure, hence polynomially bounded by n in this case. But then we have **P=NP**, which contradicts our assumption.

Even if one accepts these large constants as inevitable, the method is unsatisfactory, since the presented algorithm depends effectively on the key structures $\mathscr{V}_1, \ldots, \mathscr{V}_{p_1}$, which are very difficult to find. Here is an analogy to the results of Robertson and Seymour's polynomial time algorithms for minor closed classes of graphs (Robertson and Seymour 1985). But in contrast to these results, here one can show that the key structures cannot be found algorithmically, since then the satisfiability problem of predicate logic could be solved. More precisely, we have the following lemma.

Lemma 5.1. There is no algorithm that takes as input an arbitrary first-order formula φ and calculates the key structures $\mathscr{V}_1, \ldots, \mathscr{V}_{p_1}$ in the above algorithm.

To see this, one has simply to observe that if one could compute these structures for an arbitrary first-order formula φ, one could solve the satisfiability problem for finite structures, which is impossible by Trahtenbrot's Theorem (see Ebbinghaus *et al.* (1993)). It is not difficult to observe that this result holds also for relational structures of *bounded* degree. But to decide finite satisfiability it is only necessary to know whether $p_1 = 0$ or not.

Hence, there is no algorithm that computes the key structures for an arbitrary first-order formula.

But this does not mean that there is no method capable of constructing from an arbitrarily given first-order formula a linear-time algorithm to solve the algorithmic problem defined by the formula for structures of bounded degree. In fact, it is even possible to use a simple variant of the above method. We do not really need the structures $\mathscr{V}_1, \ldots, \mathscr{V}_{p_1}$ to determine the truth $\mathscr{G} \models \varphi$. Let $\mu(\mathscr{G})$ be the vector μ calculated in our algorithm for the input $\mathscr{G}$. All that is really needed is to find for each formula φ a finite set $\Delta(\varphi)$ of vectors such that for an arbitrary S-structure G we have $\mathscr{G} \models \varphi$ if and only if $\mu(\mathscr{G}) \in \Delta(\varphi)$.

By Theorem 4.2 and the proof of Theorem 4.1, we know that there is such a set $\Delta(\varphi)$. Hence it is not difficult to deduce one using the Completeness Theorem of first-order logic (see Ebbinghaus *et al.* (1993)). The basic idea is to transform $\Delta(\varphi)$ into a formula ψ such that the formula $(\varphi \leftrightarrow \psi)$ is a valid sentence. But if ψ has the 'right form', one can deduce a possible set of vectors $\Delta(\varphi)$ from ψ (see also Hanf (1965) and Gaifman (1982)). Hence, it is sufficient to enumerate all possible valid sentences and look for a formula $(\varphi \leftrightarrow \psi)$, where ψ is of the right form. But this is possible using the Completeness Theorem.

It seems to be open, whether the following holds:

$$\mathbf{FO} \subset \mathbf{SLIN} \ ?$$

But with respect to the results of Stolboushkin and Taitslin (1994), this seems to be difficult to prove. An essential step here is the problem of handling arbitrary binary relations of unbounded degree. In this unbounded case one is unable to investigate isomorphism types of local neighbourhoods, since they can become very large; in the worst case they can contain the whole structure. A weaker problem could be to investigate first-order problems for structures of bounded degree but with one additional linear ordering on the domain as exception to the degree bound. Remembering $\mathbf{AC}^0 = \mathrm{FO} + <$ (for an arbitrary linear ordering $<$ of the domain), it seems of interest to investigate for which class K of structures $\mathbf{AC}^0$ restricted to K is contained in **LINTIME** or **SLIN**. Moreover, it could be interesting to know which extension one has to add to first-order logic to capture linear time, when one restricts both classes to structures of bounded degree (BD_d for fixed d)?

Another possible branch of research could be to investigate 'linear or polynomial time solvable problems over infinite structures'. This could make sense in the following way. Assume that we are given an arbitrary but fixed infinite structure $\mathscr{G}$, for example, arithmetic of natural numbers augmented by some additional relations and possibly an additional domain. Moreover, assume that there is an arbitrary but fixed first-order formula $\varphi(x, \ldots, y, z, \ldots, u)$ from the corresponding language, where $x, \ldots, y, z, \ldots, u$ are individual variables. Let $a, \ldots, b$ be arbitrary individuals from $| \mathscr{G} |$. Assume that with each individual is connected its size. The general problem can then be stated as follows:

Problem 5.1. Characterize those formulas $\varphi(x, \ldots, y, z, \ldots, u)$ for which it is possible to find individuals $c, \ldots, d$ from $| \mathscr{G} |$, for arbitrarily given individuals $a, \ldots, b$ from $| \mathscr{G} |$ in time linear or polynomial in the size of $a, \ldots, b$, such that $\mathscr{G} \models \varphi[a, \ldots, b, c, \ldots, d]$.

This problem can be considered for several infinite structures $\mathcal{G}$, and it has many variants, *e.g.*, one can also measure the size of the individuals $c, \ldots, d$, and one could also consider some additional finite relations as part of the input. An interesting attempt in this direction is provided by the investigations of Eaves and Rothblum (Eaves and Rothblum 1994), which also show the usefulness of transferring ideas and techniques developed in mathematical logic for proving the decidability of formal theories to investigations in complexity theory, which was one of the minor objectives of this paper.

For example, the method of interpretability is neither restricted to relational structures nor to first-order problems. But it is easy to generalize it to arbitrary classes of structures and to other languages. So it is possible to show that the set of trees is lintime-universal with respect to the class of graphs of tree-width $\leq k$ (for an arbitrary but fixed $k \in \mathbb{N}$) for the CMSO-logic and some of its extensions, and hence we can get linear-time computability results for many algorithmic problems on such classes of graphs (using ideas from Arnborg *et al.* (1991) and Seese (1992)).

References

Abitoul, S., Vardi, M. and Vianu, V. (1992) *Fixpoint logics, relational machines and computational complexity*, Proceedings of the 33rd Annual Symposium on Foundations of Computer Science 156–192.

Aho, A., Hopcroft, J. and Ullman, J. D. (1974) *The design and analysis of computer algorithms*, Addison-Wesley.

Aho, A. and Ullman, J. D. (1979) *Universality of Data Retrieval Languages*, 6th Symposium on Principles of Programming Languages 110–117.

Ajtai, M. and Fagin, R. (1990) Reachability is harder for directed than for undirected finite graphs. *Journal of Symbolic Logic* **55** (1) 113–150.

Arnborg, S., Courcelle, B., Proskurowski, A. and Seese, D. (1993) An algebraic theory of graph reduction. *JACM* **40** (5) 1134–1164.

Arnborg, S., Lagergren, J. and Seese, D. (1991) Easy problems for tree-decomposable graphs. *Journal of Algorithms* **12** 308–340.

Baudisch, A., Seese, D., Tuschik, P. and Weese, M. (1982) Decidability and Quantifier-Elimination. In: Barwise, J. and Feferman, S. (eds.) *Model-Theoretic Logics*, Springer-Verlag 235–268.

Behmann, H. (1922) Beiträge zur Algebra der Logik, insbesondere zum Entscheidungsproblem. *Math. Ann.* **86** 163–229.

Chandra, A. K. and Harel, D. (1980) Stucture and complexity of relational queries. *J. Comput. Syst. Sci.* **25** 156–178.

Church, A. and Quine, W. (1953) Some theorems on definability and decidability. *Journ. Symb. Logic* **17** 179–187.

Compton, K. and Henson, C. (1987) A uniform method for proving lower bounds on the computational complexity of logical theories, The University of Michigan, Computing Research Laboratory, CRL-TR-04-87.

Cormen, T., Leiserson, C. and Rivest, R. (1990) *Introduction to Algorithms*, The MIT Press.

Courcelle, B. (1994) Monadic second-order definable transductions: a survey. *Theoretical Computer Science* **126** 53–75.

Courcelle, B. and Mosbah, M. (1993) Monadic second-order evaluations on tree-decomposable graphs. *Theoretical Computer Science* **109** 49–82.

Creignou, N. (1993) The class of problems that are linearly equivalent to Satisfiability or a uniform method for proving **NP**–completeness. In: Computer Science Logic 1992, San Miniato. *Springer-Verlag Lecture Notes in Computer Science* **702** 115–133.

Dahlhaus, E. (1983) Reduction to **NP**–complete problems by interpretations. In: Boerger, Hasenjaeger and Roedding (eds.) Logic and Machines, Decision problems and Complexity. *Springer-Verlag Lecture Notes in Computer Science* **171** 357–365.

Eaves, B. and Rothblum, U. (1994) Linear Problems and linear algorithms, Rutcor Research Report, RRR28-94.

Ebbinghaus, H.-D., Flum, J. and Thomas, W. (1993) *Mathematical Logic*, Second Edition, Springer-Verlag.

Fagin, R. (1974) Generalized First-Order spectra and polynomial-time recognizable sets. In: Karp, R. (ed.) Complexity of Computation. *SIAM-AMS Proc.* **7** 27–41.

Fagin, R., Stockmeyer, L. and Vardi, M. (1993) On monadic NP vs. monadic co-NP. In: The Proceedings of the 8th Annual IEEE Conference on Structure in Complexity Theory 19–30. (Full paper in Information and Computation (1995) **120** (1) 78–92.)

Gaifman, H. (1982) On local and nonlocal properties. In: Stern, J. (ed.) *Logic Colloquium '81*, North-Holland 105–135.

Giakoumakis, V. (1995) On imperfect P_4 - sparse graphs. In: Faigle, U. and Hoede, C. (eds.) *Proceedings 4th Twente workshop on graphs and combinatorial optimization* 110–113.

Giammarresi, D., Restivo, A., Seibert, S. and Thomas, W. (1993) Monadic second order-logic over rectangular pictures and recognizability by tiling systems, Report No. 9318, Christian-Albrechts-Universität Kiel (also to appear in *Information and Computation*).

Grädel, E. (1990) On the notion of linear time computability. *Internat. J. Foundations Comput. Sci.* **1** 295–307.

Grandjaen, E. (1993) Linear time algorithms and **NP**–complete problems. In: Proceedings Computer Science Logic 1992, San Miniato. *Springer-Verlag Lecture Notes in Computer Science* **702** 248–273 (also to appear in *SIAM J. on Computing*).

Gurevich, V (1984) Toward logic tailored for computational complexity. In: Börger, E. *et al.* (eds.) Computation and Proof Theory. *Springer-Verlag Lecture Notes in Mathematics* **1104** 175–216.

Gurevich, Y. (1985) Monadic second-order theories. In: Barwise, J. and Feferman, S. (eds.) *Model-Theoretic Logics* Chap. XIII, Springer-Verlag 479–506.

Hanf, W. (1965) Model-theoretic methods in the study of elementary logic. In: Addison, J., Henkin, L. and Tarski, A. (eds.) *The Theory of Models*, North-Holland 132–145.

Hauschild, K. and Rautenberg, W. (1971) Interpretierbarkeit in der Gruppentheorie. *Algebra universalis* **1** fasc. 2 136–151.

Immerman, N. (1982) Upper and Lower Bounds for First Order Expressability. *JCSS* **25** (1).

Immerman, N. (1986) Relational queries computable in polynomial time. *Information and Control* **68** 86–104.

Immerman, N. (1987) Languages that capture complexity classes. *SIAM J. Comput.* **16** (4) 760–778.

A. I. Maltsev (1959) *Regular products of models*, Izv. Acad. Nauk. SSSR, Ser. Mat., 23 489–502.

Mehlhorn, K. (1984) *Graph Algorithms and NP-Completeness*, Springer-Verlag.

Minsky, M. L. (1967) *Computation: Finite and Infinite Machines*, Prentice-Hall.

Papadimitriou, C. (1994) *Computational Complexity*, Addison-Wesley.

Rabin, M. (1965) A simple method for undecidability proofs and some applications. In: Bar-Hillel (ed.) *Logic, Methodology and Philosophy of Science* II, North-Holland 58–68.

Rabin, M. (1977) Decidable Theories. In: Barwise, J. (ed.) *Handbook of Mathematical Logic*, North-Holland 595–629.

Robertson, N. and Seymour, P. (1985) Graph minors — A survey. In: Anderson, I. (ed.) *Surveys in Combinatorics*, Cambridge University Press 153–171.

de Rougemont, M. (1987) Second-order and inductive definability on finite structures. *Zeitschrift f. math. Logik und Grundlagen d. Math.* **33** 47–63.

Schwentick, T. (1994) Graph connectivity and monadic NP, Informatik-Bericht Nr. 2/94, Johannes Gutenberg - Universität Mainz, Institut für Informatik, FB 17 1–33.

Seese, D. (1992) Interpretability and tree automata: a simple way to solve algorithmic problems on graphs closely related to trees. In: Nivat, M. and Podelski, V (eds.) *Tree Automata and Languages*, Elsevier Science Publishers 83–114.

Stolboushkin, A. P. and Taitslin, M. A. (1994) Is First Order Contained in an Initial Segment of PTIME? In: Pacholski, L. and Tiuryn, J. (eds.) Proceedings of CSL'94. *Springer-Verlag Lecture Notes in Computer Science* **933** 242–248.

Thomas, W. (1991) On logics, tilings and automata. In Proc. 18th ICALP. *Springer-Verlag Lecture Notes in Computer Science* **510** 441–454.

Wagner, K. and Wechsung, G. (1986) *Computational Complexity*, VEB Deutscher Verlag der Wissenschaften, Berlin.