INFORMATION AND COMPUTATION 90, 50–66 (1991)

The Expressive Power of Stratified Logic Programs

PHOKION G. KOLAITIS*

Computer and Information Sciences, University of California, Santa Cruz, Santa Cruz, California 95064

1. INTRODUCTION

The study of negation in logic programming has been the topic of substantial research activity during the past several years, starting with the negation as failure semantics in Clark (1978), and Apt and van Emden (1982). More recently, a major direction of research has focused on the class of *stratified logic programs*, in which no predicate is defined recursively in terms of its own negation and which can be given natural semantics in terms of iterated fixpoints. Stratified logic programs were introduced and studied first by Chandra and Harel (1985), but soon attracted the interest of researchers from both database theory and artificial intelligence. Recent research work on stratified logic programs and their generalizations includes the papers by Apt, Blair, and Walker (1988), Van Gelder (1986), Lifschitz (1988), Przymusinski (1988), Apt and Pugin (1987), and others. At the same time, stratified logic programs became the choice for the treatment of negation in the NAIL! system developed at Stanford University by Ullman and his co-workers (cf. Morris *et al.*, 1986, 1987).

A fundamental question about any query language is to determine its exact expressive power. Issues of expressiveness have been the driving force behind the introduction of progressively more powerful logical query languages. Aho and Ullman (1979), for example, pointed out the poverty of first-order query languages on finite structures and suggested augmenting first-order logic with operators capable of capturing natural properties, such as connectivity, that are not first-order definable. DATALOG (the language of Horn-clause programs without function symbols and without negation) can be viewed as a natural development of this suggestion.

It is well known that stratified logic programs have strictly higher expressive power than DATALOG programs. Indeed, stratified logic

* During the preparation of this paper the author was partially supported by NSF Grant DMS-8896255.

0890-5401/91 $3.00

programs have the property that they can be divided into layers, so that at any given layer all predicates occurring negatively have been defined at a lower layer (the first layer, in particular, is negation-free). It follows that every DATALOG program can be viewed as a stratified program with a single layer. Moreover, the queries computable by stratified logic programs contain properly those computable by DATALOG programs, since the former are closed under negation, while the latter are not. As a typical example of this situation, consider the following stratified program π in which the predicate P computes the complement of the transitive closure of a given binary database relation E:

$$P(x, y) \leftarrow \neg S(x, y),$$
$$S(x, y) \leftarrow E(x, y),$$
$$S(x, y) \leftarrow E(x, z), S(z, y).$$

What is the exact expressive power of stratified logic programs? Chandra and Harel (1985) addressed this question and stated a theorem (their Theorem 5.4) to the effect that on finite structures stratified logic programs have the same expressive power as *fixpoint logic*, which is obtained from first-order logic by adding the least fixed point operator for positive first-order formulas. Fixpoint logic is one of the most well studied and understood extensions of first-order logic on both infinite structures (Moschovakis, 1974) and finite structures (Chandra and Harel, 1982; Vardi, 1982; Immerman, 1983, 1986; Gurevich and Shelah, 1986; and others). Thus, it appeared for awhile that the question concerning the expressive power of stratified logic programs was settled. Later on, however, P. Kanellakis (1987, personal communication) pointed out certain difficulties in carrying out the sketch of proof of Theorem 5.4 given in (Chandra and Harel, 1985) and raised the problem of producing a complete proof of the claimed result. It turns out that this result is *not* correct.

In a recent paper Dahlhaus (1987) obtained results about normal forms in fixpoint logic. In particular, he proved that fixpoint logic on finite structures is strictly more expressive than its *existential fragment* EFP, in which only existential formulas are iterated. It is not hard to show that the queries computable by stratified logic programs are also expressible by formulas in the existential fragment EFP of fixpoint logic. As a result, fixpoint logic on finite structures has *strictly higher* expressive power than stratified logic programs.

Dahlhaus' (1987) key idea was to study EFP and fixpoint logic on certain "game tree" structures and to exhibit a natural game theoretic query that turns out to be expressible in fixpoint logic, but not definable by *any* EFP formula (and, a fortiori, not definable by *any* stratified logic program)

on finite "game trees." These "game trees" were first introduced in an earlier paper by Chandra and Harel (1982) in order to establish that on finite structures first-order formulas of distinct quantifier nesting possess distinct expressive power.

In (Dahlhaus, 1987) there is no mention of stratified logic programs and the proof for the separation between EFP and fixpoint logic is rather terse. We present here the relation between stratified logic programs and EFP, and give a detailed proof of this separation result, which we hope will be of interest and use to researchers in database theory or artificial intelligence. We include also some additional properties of fixpoint logic on finite structures (cf. Lemma 4 and Remark 1 in Section 3), which we had to establish in order to produce a complete proof of the separation. These in turn required use of more sophisticated machinery from fixpoint logic, namely the induction completeness theorem from Moschovakis (1974) and Immerman's (1986) theorem about the collapse of the fixpoint hierarchy at the first level.

The above result establishes that fixpoint logic has strictly higher expressive power than stratified logic programs on finite structures over a vocabulary σ with at least one binary and one unary relational symbol. With minor modifications in the definition of the "game trees" we can also separate stratified logic programs from fixpoint logic on finite structures over a vocabulary consisting of a single binary relation symbol. This result turns out to be optimal, because, in terms of expressive power both fixpoint logic and stratified logic programs reduce to first-order logic over vocabularies consisting of unary relational symbols only.

Since the techniques used to separate stratified logic programs from fixpoint logic are special to finite structures, it is natural to ask if this separation still holds when we consider infinite structures. Using standard results from hyperarithmetic theory, we observe first that on the structure $\mathbf{N} = (N, +, \cdot)$ of the integers fixpoint logic has strictly higher expressive power than stratified logic programs. We then show, using more advanced methods from fixpoint logic, that this separation holds on any infinite structure with a "long" fixpoint program. We can conclude, therefore, that fixpoint logic is more expressive than stratified logic programs on many infinite structures of mathematical interest.

2. Stratified Logic Programs and Fixpoint Logic

This section contains the definitions of the main concepts and a minimum amount of necessary background material. We assume that we have a fixed underlying vocabulary σ consisting of relational symbols $R_1, \ldots, R_s$ and we consider relational structures over this vocabulary.

2.1. *Stratified Logic Programs*

A *general logic program* is a set of *rules* of the form

$$t_0 \leftarrow t_1, t_2, ..., t_k,$$

where the t_i's are *literals*. The literal t_0 is the *head* of the rule, while the others constitute the *body* of the rule. Each literal in the body is an atomic formula $Q(x_1, ..., x_n)$ or a negated atomic formula $\neg Q(x_1, ..., x_n)$, where Q is one of the relational symbols R_i in the vocabulary σ or some other relational variable S which is not in σ. The head of the rule is an atomic formula $Q(x_1, ..., x_n)$, where Q is a relational variable. One of the relational variables of the program is a distinguished *goal* predicate.

A DATALOG program is a general logic program such that no literal in the body is a negated atomic formula involving a relational variable. The following is a typical example of a DATALOG program:

$$S(x, y) \leftarrow E(x, y),$$

$$S(x, y) \leftarrow S(x, z), S(z, y)$$

(here E is a relational symbol in the vocabulary σ).

Let $D = (A, R_1, ..., R_s)$ be a relational structure over the vocabulary σ and let π be a DATALOG program with $S_1, ..., S_m$ as its relational variables; moreover, let us assume that $S_1, ..., S_m$ have arities $n_1, ..., n_m$, respectively. The program π gives rise to an operator Θ_D, or simply Θ, assigning values $\Theta(\bar{S})$ to every sequence $\bar{S} = (S_1, ..., S_m)$ of relations on the universe A of D whose arities are $n_1, ..., n_m$. $\Theta(\bar{S})$ is itself a sequence of relations on A of arities $n_1, ..., n_m$; these relations are obtained by applying all the rules of π to the relations $R_1, ..., R_l$ of D and the relations $S_1, ..., S_m$.

If the relational variable S_j is the designated goal predicate, then the *semantics* of the program π on D is the coordinate S_j^∞ of the *least fixed point* $\Theta^\infty = (S_1^\infty, ..., S_m^\infty)$ of the operator Θ on D. The *semantics* of the program π on D can alternatively be obtained by iterating the operator Θ until its least fixed point is reached (where the iteration starts by assigning the empty set to the relational variables of Θ). More formally, if

$$\Theta^1 = \Theta(\bar{\varnothing}) \qquad \text{and} \qquad \Theta^{n+1} = \Theta(\Theta^n),$$

then the least fixed point of Θ is

$$\Theta^\infty = \bigcup_{n=1}^{\infty} \Theta^n,$$

where the union is taken coordinatewise.

In the case of the previous example, on any structure of the form $D = (A, E)$ the program gives rise to the operator

$$\Theta(S) = \{(x, y): E(x, y) \vee \exists z(S(x, z) \wedge S(z, y))\}.$$

In this case for each $n \geqslant 1$ we have that Θ^n is the binary relation on A consisting of all pairs (a, b) connected by a path along E of length at most 2^{n-1}. Thus, the least fixed point $\Theta^\infty = \bigcup_{n=1}^{\infty} \Theta^n$ is the transitive closure of the relation E.

A general logic program is *stratified* if, intuitively, recursion is not allowed "through" negation. More formally, a general logic program π having $S_1, ..., S_m$ as its relational variables is *stratified* if there is a partition $P = \bigcup_{i \leqslant l} P_i$ of the clauses of π such that:

- If a relational variable S_t occurs positively in the body of a rule in some P_i, then every rule having S_t in its head is contained in $\bigcup_{j \leqslant i} P_j$.
- If a relational variable S_t occurs negatively in the body of a rule in some P_i, then every rule having S_t in its head is contained in $\bigcup_{j < i} P_j$ (in particular no relational variable occurs negatively in the body of a rule in the first stratum P_1).

Stratified logic programs can be given natural semantics, using the least fixed point semantics for DATALOG programs repeatedly (iterated fixed point semantics; Chandra and Harel, 1985; Van Gelder, 1986; Apt *et al.*, 1988). Given a structure D, we start by computing the least fixed point of the first stratum (which is itself a DATALOG program over D) and then proceed inductively through the other strata, viewing each stratum P_i as a DATALOG program over the expansion of D obtained by augmenting it with the values of the relational variables computed in lower strata. It is known (Apt *et al.*, 1988) that the semantics of stratified logic programs is independent of the stratification used. The program π given in the introduction is an example of a stratified program with two strata; it computes the complement of the transitive closure of E. On the other hand, the program

$$T(x) \leftarrow E(y, x), \neg T(y)$$

is clearly not stratified.

2.2. *Fixpoint Logic*

The syntax of *fixpoint logic* FP is obtained by augmenting the syntax of first-order logic with the

- *Least Fixpoint Formation Rule*: If $\varphi(x_1, ..., x_n, S)$ is a formula over the vocabulary $\sigma \cup \{S\}$ in which S is a n-ary relational variable symbol

with *positive* occurrences only (i.e., within an even numbers of negations), then $\varphi^{\infty}(x_1, ..., x_n)$ is also a formula, called the *least fixpoint* of φ.

If D is a structure over the vocabulary σ, then $\varphi^{\infty}(x_1, ..., x_n)$ on D is interpreted to be the smallest n-ary relation S on the universe of D such that

$$D \models (\forall \bar{x})(S(\bar{x}) \leftrightarrow \varphi(\bar{x}, S)),$$

where $\bar{x} = (x_1, ..., x_n)$. In other words, φ^{∞} is the least fixed point of the operator

$$\Theta(S) = \{\bar{x}: D \models \varphi(\bar{x}, S)\}.$$

Note that this least fixed point always exists, because the positivity condition on φ guarantees that Θ is monotone. The least fixpoint φ^{∞} can also be defined "from below" by induction on the ordinals as

$$\varphi^{\infty} = \bigcup_{\xi} \varphi^{\xi},$$

where the *stages* φ^{ξ} are given by

$$\varphi^{\xi}(\bar{x}) \Leftrightarrow \varphi\left(\bar{x}, \bigcup_{\eta < \xi} \varphi^{\eta}\right).$$

On any structure D (finite or infinite) there is a least ordinal ξ_0 such that $\varphi^{\infty} = \varphi^{\xi_0} = \varphi^{\xi}$ for every $\xi \geqslant \xi_0$. This is called the *closure ordinal* of φ on D and is denoted by $cl(\varphi, D)$, or $|\varphi|$ when D is understood. If D is a finite structure with r elements, then the closure ordinal of φ is a number $\leqslant r^n$ (where n is the number of free variables in φ). On infinite structures, however, $|\varphi|$ may be a finite or an infinite ordinal $\geqslant \omega$.

Some examples are in order. Let $\varphi(x, y, S)$ be the first-order formula

$$E(x, y) \vee (\exists z)(S(x, z) \wedge S(z, y)),$$

where E is a binary relation symbol in the vocabular σ. We have that

$$\varphi^{\infty}(x, y) \Leftrightarrow \text{there is a path from } x \text{ to } y;$$

i.e., the least fixpoint of φ defines the *transitive closure* query. We also have that $|\varphi| \leqslant \omega$ on any structure and that $|\varphi| = \omega$ on exactly the structures of infinite diameter. For a different example, let $\psi(x, S)$ be the first-order formula

$$(\forall y)(E(y, x) \rightarrow S(y)).$$

It is easy to verify that

$$\psi^{\infty}(x) \Leftrightarrow (\textit{there is no infinite sequence } x_1, ..., x_n, ... \textit{ such that } x_1 = x \textit{ and } E(x_{n+1}, x_n) \textit{ for all } n \geqslant 1).$$

On finite structures ψ^{∞} consists of all vertices for which there is no path to them from a cycle. On the other hand, if E is an infinite well-founded relation of rank an ordinal ξ, then $|\psi| = \xi$ and thus on infinite structures the closure ordinal of ψ may be an arbitrarily large ordinal.

Note that fixpoint logic allows for the interleaving of the least fixed operator with the propositional connectives and the first-order quantifiers. On infinite structures this gives rise in general to a hierarchy, where different levels of the syntax may have different expressive power. Let FP_1 be the fragment of FP obtained by applying the first-order quantifiers, conjunction, and disjunction (but no negation) to the least fixpoints of all positive first-order formulas. It is well known (cf. Moschovakis, 1974) that on the structure $\mathbf{N} = (N, +, \cdot)$ of the integers the relations definable by formulas in FP_1 coincide with the Π^1_1 predicates, i.e., the relations definable by universal second-order formulas. Since the Π^1_1 relations on the integers are not closed under complement, it follows that FP_1 has strictly lower expressive power than the full fixpoint logic FP on $\mathbf{N}$. The situation, however, changes dramatically when we consider fixpoint logic uniformly on finite structures. Indeed, Immerman (1986) proved that on finite structures the fixpoint hierarchy collapses down to FP_1; i.e., every query in FP is expressible by a formula of FP_1. In other words, on the class of all finite structures a *single* application of the least fixpoint operator to positive first-order formulas suffices to generate *all* queries in fixpoint logic.

2.3. *Stratified Logic Programs vs Fixpoint Logic*

What is the relation between stratified logic programs and fixpoint logic? It is not hard to verify that the queries computable by stratified logic programs are also expressible in fixpoint logic. Actually they are expressible in the *existential fragment* EFP of fixpoint logic, which is obtained by interleaving the *least fixpoint operator* with the propositional connectives and *existential quantification.* More formally, let E be the collection of all formulas of the form $(\exists \bar{y})\chi(\bar{x}, \bar{y}, \bar{R}, \bar{S})$, where χ is quantifier-free, $\bar{R}$ are the relations in the vocabulary σ, and $\bar{S}$ are relational variables which may or may not occur positively in χ. We define by induction on l a sequence E_l of classes of formulas; the existential fragment EFP will be the union of these classes.

- To define the first level EFP_1 of EFP, we start by taking least fixpoints of formulas $(\exists \bar{y})\chi(x_1, ..., x_n, \bar{y}, \bar{R}, S)$ in E such that S is an n-ary relational variable occurring only positively in χ. We define EFP_1 to be the

smallest class of formulas obtained from these least fixpoints by applying disjunction, conjunction, and existential quantification.

- We define EFP_{l+1} from the class EFP_l as follows: form first least fixpoints (with respect to S) of formulas $(\exists \bar{y})\chi(x_1, ..., x_n, \bar{y}, \bar{R}, S, S_1, ..., S_k)$, where χ is quantifier-free, S is a n-ary relational variable occurring only positively in χ, and for the relational variables $S_1, ..., S_k$ (which need not occur positively in χ) we have substituted formulas from $\bigcup_{i \leqslant l} \mathrm{EFP}_i$. Now let EFP_{l+1} be the smallest class of formulas obtained from these fixpoints by applying disjunction, conjunctions, and existential quantification.

- Finally, set

$$\mathrm{EFP} = \bigcup_{l=1}^{\infty} \mathrm{EFP}_l.$$

The following theorem provides the connection between stratified logic programs and EFP. It can be proved in a straightforward manner along the lines of Theorems 4.1 and 5.1 in Chandra and Harel (1985).

THEOREM 1. *If a query is computable by a stratified program having a stratification* $P = \bigcup_{i \leqslant l} P_i$, *then it is expressible by a formula in* EFP_l. *As a result, fixpoint logic* FP *contains all queries expressible by stratified logic programs.*

Theorem 5.4 in (Chandra and Harel, 1985) states that fixpoint logic and stratified logic programs have identical expressive power. The suggested method of proof in (Chandra and Harel, 1985) is to use induction on the construction of fixpoint formulas to show they can be simulated by stratified logic programs. It turns out, however, that this is *not* possible. In the next section we prove that on the class of all finite structures over a vocabulary with one binary and one unary relational symbol fixpoint logic has *strictly higher* expressive power than stratified programs.

3. STRATIFIED LOGIC PROGRAMS ON FINITE STRUCTURES

We define here the "game tree" structures that were introduced in Chandra and Harel (1982) and compare the expressive power of stratified logic programs vs fixpoint logic on these structures. Let σ be the vocabulary consisting of a binary symbol *Move* and a unary symbol *Black*. For every $k \geqslant 1$ we define by induction on $i \geqslant 0$ simultaneously two sequences of structures $B_{i,k}$ and $B'_{i,k}$ over σ as follows:

• $B_{0,k}$ and $B'_{0,k}$ have a singleton as their universe and the predicate *Move* is empty in both. The predicate *Black* is non-empty in $B_{0,k}$ and empty in $B'_{0,k}$. Thus, they are of the form:

$$B_{0,k} = (\{d_0\}, \varnothing, \{d_0\}) \qquad \text{and} \qquad B'_{0,k} = (\{d'_0\}, \varnothing, \varnothing).$$

• $B_{i+1,k}$ consists of a copy of $B'_{i,k}$, k disjoint copies of $B_{i,k}$, and a separate *root* d_{i+1}. The predicate *Move* is the union of the *Move* predicates in the $B_{i,k}$'s and $B'_{i,k}$ together with new "moves" from the root d_{i+1} to the roots of the copy of $B'_{i,k}$ and to the copies of $B_{i,k}$. The predicate *Black* is the union of the *Black* predicates in the $B_{i,k}$'s and $B'_{i,k}$.

• $B'_{i+1,k}$ consists of $k+1$ disjoint copies of $B_{i,k}$ and a separate *root* d'_{i+1}. The predicate *Move* is the union of the *Move* predicates in the $B_{i,k}$'s together with new "moves" from the root d'_{i+1} to the roots of the $B_{i,k}$'s. The predicate *Black* is the union of the *Black* predicates in the $B_{i,k}$'s.

In the sequel we use the term *game tree* to refer to one of the structures $B_{i,k}$ or $B'_{i,k}$. With each game tree and each interior node x of it, we associate a *Game* $G(x)$ played between *Player I* and *Player II* as follows: a round of the game starts with Player I picking a node y such that *Move*(x, y) holds. After this, the two players take turns picking every time a node z that is a "next move" of the node y (i.e., *Move*(y, z)) played in the previous step, until they reach the "leaves" of the tree. Player I wins the round if the last node played by Player II is in *Black*. We say that *Player I wins the game* $G(x)$ if he has a winning strategy that allows him to win every round of the game. The game trees $B_{4,1}$ and $B'_{4,1}$ are illustrated in Fig. 1.

LEMMA 2. *There is a first-order formula* $\varphi(x, S)$ *over the vocabulary* σ *such that* S *has positive occurrences only and*

$$\varphi^{\infty}(x) \Leftrightarrow \textit{Player I wins the game } G(x)$$

on every game tree.

Proof. Let $\varphi(x, S)$ be the formula

$$(\exists y)[\textit{Move}(x, y) \wedge (\forall z)(\textit{Move}(y, z) \rightarrow \textit{Black}(z))]$$
$$\vee\ (\exists y)[\textit{Move}(x, y) \wedge (\forall z)(\textit{Move}(y, z) \rightarrow S(z))].$$

An induction on the stages φ^{ξ} of φ establishes that it defines the query "Player I wins the game $G(x)$" on game tress. ∎

Note that the root of each game tree is definable on every game tree of height at least 2 by the first-order formula $\rho(x)$ asserting that

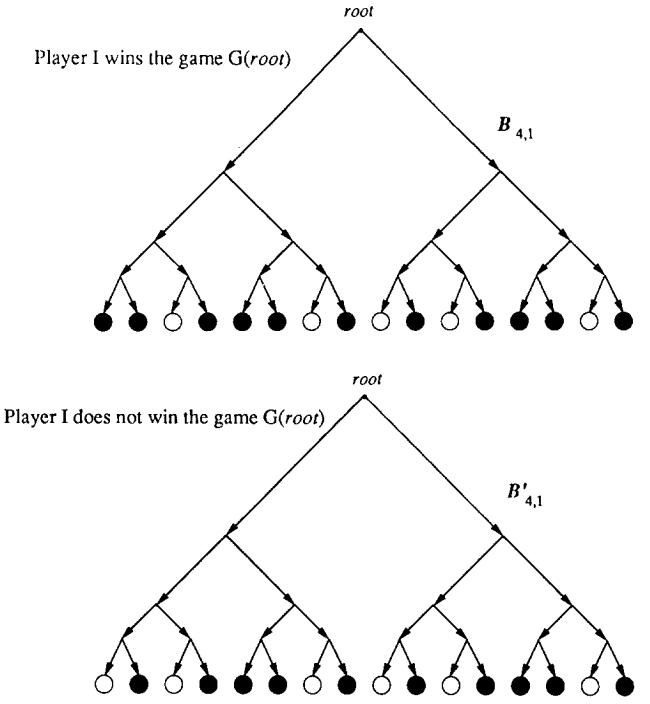

FIGURE 1

$(\exists y)\, Move(x, y) \wedge (\forall z)(\neg\, Move(z, x))$. It follows from Lemma 2 that the sentence ψ

$$(\exists x)(\rho(x) \wedge \varphi^{\infty}(x))$$

is in fixpoint logic and defines exactly those game trees for which Player I wins the game $G(root)$ (i.e., Player I starts at the root). It is now easy to verify that for any $i \geqslant 1$ and any $k \geqslant 1$

$$B_{2i,k} \models \psi \text{ while } B'_{2i,k} \models \neg\, \psi.$$

Indeed, the strategy of Player I on the game trees $B_{2i,k}$ is to always move along the leftmost branch of the tree (compare also the game trees $B_{4,1}$ and $B'_{4,1}$ in Fig. 1).

We now show that the property "Player I wins the game $G(root)$" is *not* expressible by any stratified program on game trees. We actually prove that no sentence in the existential fragment EFP of fixpoint logic is equivalent to the sentence ψ above and then appeal to Theorem 1 to derive the separation between stratified logic programs and fixpoint logic. The proof

is carried out in a sequence of lemmas to the effect that every sentence in some level EFP_l of EFP is equivalent to a fixed first-order sentence with l alternations of quantifiers on all game trees $B_{l+2,k}$ and $B'_{l+2,k}$ for any $k \geqslant 1$.

DEFINITION. For any $k \geqslant 1$ and any $l \geqslant 1$, let $\Sigma_{l,k}$ be the collection of all first-order formulas over σ in prenex form with l alternations of quantifiers starting with an existential quantifier and having at most k quantifiers in every quantifier block of the same type. Thus, every formula in $\Sigma_{l,k}$ is of the form

$$(\exists \bar{x}_1)(\forall \bar{x}_2) \cdots (Q\bar{x}_l)\theta(\bar{x}_1, \bar{x}_2, ..., \bar{x}_l),$$

where θ is quantifier-free and each $\bar{x}_i$, $1 \leqslant i \leqslant l$, is a sequence of variables of length at most k.

Recall from Section 2 that the closure ordinal $cl(\varphi, D)$ of a formula $\varphi(x_1, ..., x_n, S)$ of fixpoint logic on some structure D is the smallest ordinal ξ_0 such that $D \models \varphi^{\xi} \leftrightarrow \varphi^{\xi_0}$ for all $\xi \geqslant \xi_0$. If, in addition, the structure D is finite, then $cl(\varphi, D)$ is a finite number. The next two lemmas reveal some crucial properties of formulas in EFP and their closure ordinals on arbitrary finite structures.

LEMMA 3. *If D is a finite structure and χ is a formula of* EFP_l *for some $l \geqslant 1$, then there is a number $k \geqslant 1$ and a first-order formula χ^* in $\Sigma_{l,k}$ such that*

$$D \models \chi \leftrightarrow \chi^*.$$

Proof (Outline). If $\varphi(x_1, ..., x_n, S, S_1, ..., S_t)$ is an existential formula in which S is an n-ary relational variable symbol occurring only positively and $S_1, ..., S_t$ are additional predicates, then on any structure the finite stages φ_m, $1 \leqslant m < \omega$, of φ are definable by existential formulas over the vocabulary $\sigma \cup \{S_1, ..., S_t\}$. Thus, on any finite structure D the least fixpoint φ^{∞} of φ is definable by an existential formula over $\sigma \cup \{S_1, ..., S_t\}$. The lemma now follows from this observation and the definition of EFP, using induction on the levels EFP_l, $l \geqslant 1$, of EFP. ∎

LEMMA 4. *Let $\varphi(x_1, ..., x_n, S)$ be a formula of fixpoint logic in which S is a n-ary relational variable symbol with positive occurrences only. There is a first-order formula*

$$\chi(y_1, ..., y_m, x_1, ..., x_n, T)$$

such that T is a $(n+m)$-ary relational variable symbol occurring only positively and such that on any finite structure D

$$D \models \varphi^{\infty}(\bar{x}) \leftrightarrow (\exists \bar{y})\chi^{\infty}(\bar{y}, \bar{x})$$

and

$$cl(\varphi, D) \leqslant cl(\chi, D).$$

Proof (Outline). This lemma is a refinement of Theorem 2 in Immerman (1986), asserting that on finite structures fixpoint logic FP collapses to its first level FP_1 (obtained by applying first-order operations other than negation to least fixpoints of positive first-order formulas). Immerman (1986) shows how to replace negations of least fixed points by predicates in FP_1 uniformly on finite structures. The lemma follows then by combining that proof with Theorem 6B.4 (Positive Induction Completeness Theorem) in Moschovakis (1974), which states that if $\varphi(x_1, ..., x_n, S, S_1, ..., S_t)$ is a first-order formula in which we substitute predicates from FP_1 for $S_1, ..., S_t$, then the least fixpoint φ^∞ of φ can be obtained by existential quantification from the least fixpoint χ^∞ of a first-order formula χ such that on any finite structure D and any additional predicates $S_1, ..., S_t$ in FP_1 we have $cl(\varphi, D') \leqslant cl(\chi, D)$, where D' is D expanded with $S_1, ..., S_t$. ∎

The next two lemmas (Lemmas 5 and 6) give special properties of the game trees. The crucial fact behind these properties is that the number of n-types on the game trees $B_{i,k}$ and $B'_{i,k}$ is bounded by a function that depends only on i (and not on k).

LEMMA 5. *Let $\varphi(x_1, ..., x_n, S)$ be a first-order formula in which S is a n-ary relational variable symbol with positive occurrences only. There are functions $f(\varphi, i)$ and $f'(\varphi, i)$ such that for any $i \geqslant 1$ and any $k \geqslant 1$*

$$cl(\varphi, B_{i,k}) \leqslant f(\varphi, i) \qquad \text{and} \qquad cl(\varphi, B'_{i,k}) \leqslant f'(\varphi, i).$$

Proof (Outline). We say that two sequences $(a_1, ..., a_n)$ and $(b_1, ..., b_n)$ of elements from the universe of a structure D have the same *n-type* if they satisfy the same first-order formulas on D. If two sequences have the same n-type, then either they enter the least fixpoint φ^∞ of $\varphi(x_1, ..., x_n, S)$ of φ at the same stage or they are not in φ^∞. It follows that $cl(\varphi, D)$ is bounded by the number of distinct n-types on D. Thus, to establish the lemma suffices to prove that the number of n-types on $B_{i,k}$ and $B'_{i,k}$ is given by functions $f_n(i)$ and $f'_n(i)$ that depend only on i. This in turn is proved by induction on n and i using the recursive construction of the game trees. We illustrate here the argument for $n = 1$ and $n = 2$.

For $n = 1$ it is clear that the required functions satisfy the equations: $f_1(1) = 3$, $f'_1(1) = 2$, $f_1(i+1) = f'_1(i) + f_1(i) + 1$, $f'_1(i+1) = f_1(i) + 1$. For $n = 2$ and for $k > 2$ we have first that $f_2(1) = 4$, $f'_2(1) = 2$. The number of unordered distinct 2-types on $B_{i+1,k}$ is bounded by the expression

$$f_1(i) + f'_1(i) + f_1(i)\, f'_1(i) + f'^2_1(i) + f_2(i) + f'_2(i).$$

The first term accounts for the 2-types between the root and nodes in the copies of $B_{i,k}$, the second for the 2-types between the root and nodes in the copy of $B'_{i,k}$, the third for the 2-types between a node from a copy of $B_{i,k}$ and a node from the copy of $B'_{i,k}$, the third for the 2-types between a node from a copy of $B_{i,k}$ and a node from the copy of $B'_{i,k}$, the fourth for the 2-types between nodes from two different copies of $B_{i,k}$, and finally the last two terms account for the 2-types between two nodes from the copy of $B_{i,k}$ or from the same copy of $B'_{i,k}$. A similar reasoning shows that the number of unordered distinct 2-types on $B'_{i,k}$ is bounded by the expression

$$f_1(i) + f_1^2(i) + f_2(i). \quad \blacksquare$$

LEMMA 6. *Let χ be a formula of* EFP_l *for some $l \geqslant 1$. For every $i \geqslant 1$ there is a number k_0 and a formula χ^* of Σ_{l,k_0} such that χ is equivalent to χ^* on the game trees $B_{i,k}$ and $B'_{i,k}$ for any $k \geqslant 1$.*

Proof (Outline). If χ is in EFP_l, then for any finite structure D there is a k and a formula χ^* in $\Sigma_{l,k}$ equivalent to χ on D. The formula χ^* depends only on χ and the closure ordinals on D of the EFP formulas whose least fixpoints are subformulas of χ. We can now apply Lemma 4 repeatedly to show that there is a first-order formula φ whose closure ordinal dominates the closure ordinals of the subformulas of χ. But, by Lemma 5, for any fixed i the closure ordinal of this φ on $B_{i,k}$ and $B'_{i,k}$ is independent of k, and therefore we have a k_0 and a fixed formula χ^* in Σ_{l,k_0} equivalent to χ on $B_{i,k}$ and $B'_{i,k}$ for any $k \geqslant 1$. $\blacksquare$

We now have all the machinery needed to state and prove the following results:

THEOREM 7. *The property "Player I wins the game $G(root)$" is not expressible by any* EFP *formula.*

Proof. Let ψ be the sentence of fixpoint logic that expresses the property "Player I wins the game $G(root)$" (cf. Lemma 2). Assume that there is an integer $l \geqslant 1$ and a sentence χ of EFP_l that is equivalent to ψ on games trees. Without loss of generality, we may assume that l is an even number. Now apply Lemma 6 to χ and to $i = l+2$ to obtain a number k_0 and a fixed first-order sentence χ^* in Σ_{l,k_0} such that χ is equivalent to χ^* on the game trees $B_{l+2,k}$ and $B'_{l+2,k}$ for any $k \geqslant 1$. Since $l+2$ is even, the comments following Lemma 2 imply in particular that

$$B_{l+2,k_0} \models \chi^* \text{ while } B'_{l+2,k_0} \models \neg\, \chi^*.$$

This, however, is a contradiction, because χ^* is a first-order sentence in Σ_{l,k_0} and, by Lemma 3.9 in Chandra and Harel (1982), the game trees B_{l+2,k_0} and B'_{l+2,k_0} satisfy the same sentences in Σ_{l,k_0}. $\blacksquare$

THEOREM 8. *The property "Player I wins the game G(root)" is not computable by any stratified logic program. As a result, fixpoint logic has strictly higher expressive power than stratified logic programs on finite structures over a vocabulary consisting of one binary and one unary relational symbols.*

Proof. Immediate from Theorem 1 and Theorem 7. ▮

Remark 1. As mentioned in the Introduction, Theorem 7 was obtained in Dahlhaus (1987), where a brief proof of it is given. In particular, the crucial Lemma 5 is derived there and then it is stated that Lemma 6 follows. We did not see how to prove Lemma 6 from Lemma 5 directly, without first establishing the properties of fixpoint logic in Lemma 4. The difficulty arises from the fact that, since EFP_l formulas are built using predicates definable at "lower" levels $\mathrm{EFP}_{l'}$, $l' < l$, of EFP, one has to obtain an analog of Lemma 5 for structures that are game trees expanded with predicates expressible in EFP. Lemma 4 allows us to circumvent this difficulty.

Remark 2. With a little extra work we can show that fixpoint logic has strictly higher expressive power than stratified logic programs on finite structures over a vocabulary consisting of a binary relational symbol only. This is done by dropping the unary predicate *Black* and allowing the *Move* predicate to contain self-loops *Move* (x, x) for the nodes in *Black* (in particular $B_{0,k}$ is modified so that its *Move* predicate is $\{(d_0, d_0)\}$). Observe that the root is again uniformly first-order definable on game trees, since it is the only node x not having some node $y \neq x$ such that $Move(y, x)$. Also, the leaves of the game tress are distinguished from the interior nodes by a formula $L(x)$ saying that there is no node $y \neq x$ such that $Move(x, y)$. This in turn allows for the simulation of the *Black* predicate. The rest of the argument remains essentially unchanged. Since binary relational symbols can be simulated by relational symbols of higher arity, we have established the following

THEOREM 9. *Fixpoint logic has higher expressive power than stratified logic programs on finite structures over any vocabulary that has a relational symbol of arity at least two.*

This separation of fixpoint logic from stratified logic programs is tight, because of the following well known

THEOREM 10. *Let σ be a vocabulary having unary relational symbols only. Then every formula of fixpoint logic is equivalent to a first-order formula uniformly on finite structures. As a result, fixpoint logic and stratified logic programs have the same expressive power over the vocabulary σ.*

Proof (Outline). If σ consists of m distinct unary relation symbols, then for every $n \geqslant 1$ there are at most 2^{mn} distinct n-ary types on every structure. It follows that every least fixpoint φ^{∞} of a first-order formula $\varphi(x_1, ..., x_n, S)$ is equivalent to the stage $\varphi^{2^{mn}}$, which is in turn definable by a first-order formula. ∎

An Open Problem. The techniques presented here for separating fixpoint logic from stratified logic programs apply to queries that require essentially an "unbounded" alternation of quantifiers.

Consider now the *distance query* $D(x, y, x', y')$ on finite graphs $D = (A, E)$:

$D(x, y, x', y')$ holds if and only if the *distance* (= the length of the shortest path) from x to y is less than the distance from x' to y' (this also includes the case where $TC(x, y) \wedge \neg\, TC(x', y')$ holds; i.e., there is a path from x to y, but no path from x' to y'). Using the stage comparison theorem (cf. Moschovakis, 1974) it is easy to see that this query is expressible in fixpoint logic; indeed, note that the distance query is the stage comparison relation of the logic program that computes the transitive closure query.

We conjecture that the distance query D is *not* expressible by any stratified logic programs on finite structures. It should be pointed out, however, that the methods used here do not seem powerful enough to settle this conjecture. The reason for this is that on every finite structure the distance query can be expressed by a first-order formula in $\Sigma_{2,k}$ for some $k \geqslant 1$ that depends on the structure, and thus this query does not require an "unbounded" alternation of quantifiers.

4. Stratified Logic Programs on Infinite Structures

In view of the results in the previous section, it is natural to ask how stratified logic programs and fixpoint logic compare on infinite structures. Note that if fixpoint logic collapses to first-order logic on an infinite structure, then stratified logic programs and fixpoint logic have identical expressive power. This is, for example, the case on any infinite structure D whose first-order theory is ω-categorical; i.e., it has a unique countable model up to isomorphism (cf. Gurevich, 1984).

We show here that if fixpoint logic is "non-trivial" on an infinite structure, then it has strictly higher expressive power than stratified logic programs. We begin by discussing the situation on the structure $\mathbf{N} = (N, +, \cdot)$ of the integers.

The Π^1_1 relations on the integers are the relations definable by universal second-order formulas, while the Σ^1_1 relations are those definable by existential second-order formulas. A relation is Δ^1_1 if it is both Σ^1_1 and Π^1_1.

A classical theorem of Kleene and Spector (cf. Moschovakis, 1974) states that a relation is Π_1^1 on $\mathbf{N}$ if and only if it is at the first level FP_1 of fixpoint logic (cf. Section 2). Moreover, $\Sigma_1^1 \neq \Pi_1^1$ and, as a result, Δ_1^1 is a proper subset of $\Sigma_1^1 \cap \Pi_1^1$.

It is not hard to show that if $\varphi(x_1, ..., x_n, S, S_1, ..., S_t)$ is an existential formula in which S occurs only positively and $S_1, ..., S_t$ are replaced by Δ_1^1 predicates, then the least fixpoint φ^∞ of φ is also Δ_1^1. Combining this fact repeatedly with Proposition 1 we obtain

THEOREM 11. *Every stratified program computes a Δ_1^1 relation on $\mathbf{N} = (N, +, \cdot)$. As a result, fixpoint logic has strictly higher expressive power than stratified logic programs on the integers.*

If D is an infinite structure, then the *closure ordinal* κ^D of D is defined to be the supremum of the closure ordinals $cl(D, \varphi)$ of first-order formulas $\varphi(x_1, ..., x_n, S)$ on D. For the integers, for example, it is well known that $\kappa^{\mathbf{N}} = \omega_1^{ck}$, the smallest non-recursive ordinal. We say that an infinite structure D is *reachable* if $\kappa^D > \omega$ and there is a first-order formula $\varphi(x_1, ..., x_n, S)$ such that $\kappa^D = cl(\varphi, D)$. More intuitively, a structure is reachable if there is a "long" fixpoint formula. There are many reachable structures of mathematical interest in addition to the integers, including the reals (viewed as infinite sequences of integers), the field of rational numbers, every infinite ordinal (λ, ε), initial segments (V_k, ε) of the universe of sets, etc.

THEOREM 12. *Fixpoint logic has higher expressive power than stratified logic programs on every reachable structure.*

Proof (Hint). Combine the stage comparison theorem and the positive induction completeness theorem in (Moschovakis, 1974) to show that the class of relations computable by stratified logic programs is properly contained in the first level FP_1 of fixpoint logic on every reachable structure. ∎

RECEIVED September 3, 1988; FINAL MANUSCRIPT RECEIVED July 31, 1989

REFERENCES

AHO, V. H., AND ULLMAN, J. D. (1979), Universality of data retrieval languages, *in* "Proceedings, 6th ACM Symposium on Principles of Programming Languages, 1979," pp. 110–117.

APT, K., BLAIR, H., AND WALKER, A. (1988), Towards a theory of declarative knowledge, *in* "Foundations of Deductive Databases and Logic Programming" (J. Minker, Ed.), pp. 89–148, Morgan Kaufmann, Los Altos, CA.

APT, K., AND PUGIN, J. M. (1987), Maintenance of stratified databases viewed as a belief revision system, *in* "Proceedings, Sixth ACM SIGACT-SIGMOD-SIGART Symposium on Principles of Database Systems, San Diego, 1987," pp. 136–145.

APT, K., AND VAN EMDEN, M. H. (1982), Contributions to the theory of logic programming, *J. Assoc. Comput. Mach.* **29**, 80–88.

CHANDRA, A., AND HAREL, D. (1982), Structure and complexity of relational queries, *J. Comput. System Sci.* **25**, 99–128.

CHANDRA, A., AND HAREL, D. (1985), Horn clause queries and generalizations, *J. Logic Programming* **1**, 1–15.

CLARK, K. L. (1978), Negation as failure, *in* "Logic and Data Bases" (H. Gallaire and J. Minker, Eds.), pp. 293–322, Plenum, New York.

DAHLHAUS, E. (1987), Skolem normal forms concerning the least fixpoint, *in* "Computation Theory and Logic" (E. Börger, ed.), Lecture Notes in Computer Science, Vol. 270, pp. 101–106, Springer-Verlag, Berlin/New York.

GUREVICH, Y. (1984), Towards logic tailored for computational complexity, *in* "Computation and Proof Theory" (M. M. Richter *et al.*, Eds.), Lecture Notes in Mathematics, Vol. 1104, pp. 175–216, Springer-Verlag, Berlin/New York.

GUREVICH, Y., AND SHELAH, S. (1986), Fixed points extensions of first-order logic, *Ann. Pure Appl. Logic* **32**, 265–280.

IMMERMAN, N. (1983), Languages which capture complexity classes, *in* "Proceedings, 15th ACM Symposium on the Theory of Computing, 1983," pp. 347–354.

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

KANELLAKIS, P. (1987), Personal communication.

LIFSCHITZ, V. (1988), On the declarative semantics of logic programs with negation, *in* "Foundations of Deductive Databases and Logic Programming" (J. Minker, Ed.), pp. 177–192, Morgan Kaufmann, Los Altos, CA.

MORRIS, K., ULLMAN, J. D., AND VAN GELDER, A. (1986), Design overview of the NAIL! system, *in* "Third International Conference on Logic Programming" (G. Goos and J. Hartmanis, Eds.), Lecture Notes in Computer Science, Vol. 225, pp. 554–568, Springer-Verlag, Berlin/New York.

MORRIS, K., NAUGHTON, J. F., SARAIYA, Y., ULLMAN, J. D., AND VAN GELDER, A. (1987), YAWN! (Yet Another Window on NAIL!), preprint, Stanford University.

MOSCHOVAKIS, Y. N. (1974), "Elementary Induction on Abstract Structures," North-Holland, Amsterdam.

PRZYMUSINSKI, T. (1988). On the declarative semantics of deductive databases and logic programs, *in* "Foundations of Deductive Databases and Logic Programming" (J. Minker, Ed.), pp. 193–216, Morgan Kaufmann, Los Altos, CA.

VAN GELDER, A. (1986), Negation as failure using tight derivations for general logic programs, *in* "Proceedings, Third IEEE Symposium on Logic Programming," pp. 137–146.

VARDI, M. Y. (1982), The complexity of relational query languages, *in* "Proceedings, 14th ACM Symposium on Theory of Computing, San Francisco, 1982," pp. 137–146.