September 1991

# Investigating Logics for Feasible Computation

Anuj Dawar
*University of Pennsylvania*



University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-91-69.

# Investigating Logics for Feasible Computation


## Abstract

The most celebrated open problem in theoretical computer science is, undoubtedly, the problem of whether P = NP. This is actually one instance of the many unresolved questions in the area of computational complexity. Many different classes of decision problems have been defined in terms of the resources needed to recognize them on various models of computation, such as deterministic or non-deterministic Turing machines, parallel machines and randomized machines. Most of the non-trivial questions concerning the inter-relationship between these classes remain unresolved. On the other hand, these classes have proved to be robustly defined, not only in that they are closed under natural transformations, but many different characterizations have independently defined the same classes. One such alternative approach is that of descriptive complexity, which seeks to define the complexity, not of computing a problem, but of describing it in a language such as the Predicate Calculus. It is particularly interesting that this approach yields a surprisingly close correspondence to computational complexity classes. This provides a natural characterization of many complexity classes that is not tied to a particular machine model of computation.



## Comments

University of Pennsylvania Department of Computer and Information Science Technical Report No. MS-CIS-91-69.

# Investigating Logics For Feasible Computation

MS-CIS-91-69

Anuj Dawar

Department of Computer and Information Science
School of Engineering and Applied Science
University of Pennsylvania
Philadelphia, PA 19104-6389

September 1990

# Investigating Logics for Feasible Computation

Anuj Dawar[1]
Department of Computer and Information Science
Univesity of Pennsylvania
Philadelphia, PA 19104.

September 30, 1991

[1]This research supported in part by ONR grant N00014-89-J-1725.

# Contents

# Chapter 1

# Introduction and Background

The most celebrated open problem in theoretical computer science is, undoubtedly, the problem of whether P = NP. This is actually one instance of the many unresolved questions in the area of computational complexity. Many different classes of decision problems have been defined in terms of the resources needed to recognize them on various models of computation, such as deterministic or non-deterministic Turing machines, parallel machines and randomized machines. Most of the non-trivial questions concerning the inter-relationship between these classes remain unresolved. On the other hand, these classes have proved to be robustly defined, not only in that they are closed under natural transformations, but many different characterizations have independently defined the same classes. One such alternative approach is that of descriptive complexity, which seeks to define the complexity, not of computing a problem, but of desribing it in a language such as the Predicate Calculus. It is particularly interesting that this approach yields a surprisingly close correspondence to computational complexity classes. This provides a natural characterization of many complexity classes that is not tied to a particular machine model of computation.

This line of investigation began with Fagin who showed that the properties expressed by existential second-order sentences are exactly those that are in the class NP[Fag74]. Immerman [Imm86] and Vardi[Var82] showed that the polynomial-time computable properties of structures in which there is an built-in order are captured exactly by the extension of first-order logic with a least-fixed-point operation. Since then, several further logical characterizations of computational complexity classes have been studied (see, for instance, [Imm89]).

The early work of Chandra and Harel[CH82] and Vardi[Var82] linked this to the development of query languages for relational databases. A relational database is nothing but a finite relational structure and query languages for such databases have generally been based on the Predicate

Calculus. This has also spurred interest in the study of the model theory of finite structures. Descriptive complexity can be seen as the study of the logical properties of finite structures, and as we shall see, the tools of classical model theory prove inadequate when applied to only finite structures.

All the results that relate descriptive logics to complexity classes below NP have relied on the presence of an ordering on the elements of the structures being described. In particular, the least-fixed-point extension of first-order logic fails to capture the class P over all structures. It is an open question whether the order-independent P-time properties even form a recursively enumerable set. This is a question of great interest, because on the one hand the P-time computable properties are generally identified as those that can be *feasibly* computed, and on the other hand the order-independent properties are those that are *generic* in that they do not depend on the order in which the data are presented, but only on their logical properties.

The inductive logics that correspond to complexity classes in the presence of ordering, namely FO + LFP and FO + PFP can be viewed as fragments of an infinitary logic with a bounded number of variables - which we denote $L^{\omega}_{\infty\omega}$. A recently proposed model of generic computation – $\mathrm{GM}^{loose}$ in [AV91b] – can also be seen as a fragment of this logic. While $L^{\omega}_{\infty\omega}$ does not capture all generic properties, as we shall see, it provides a useful tool for studying the expressive power of the inductive logics and their inter-relationship. We propose to investigate the expressive power of this and related logics.

## 1.1 Definitions and Notation

A *signature* (also sometimes called a *language* or a *vocabulary*) $\sigma$ is a finite sequence of relation and constant symbols $\langle R_1, \ldots, R_m, c_1, \ldots, c_n \rangle$. Associated with each relation symbol, $R_i$ is an arity $a_i$. We will only be considering languages without function symbols. Where this makes a difference to the results, we will mention it explicitly.

A structure over the the signature $\sigma$, $\mathcal{A} = \langle A, R_1^{\mathcal{A}}, \ldots, R_m^{\mathcal{A}}, c_1^{\mathcal{A}}, \ldots, c_n^{\mathcal{A}} \rangle$ consists of a set $A$, the universe of the structure, relations $R_i^{\mathcal{A}} \subseteq A^{a_i}$ interpreting the relation symbols in $\sigma$ and distinguished elements $c_1^{\mathcal{A}}, \ldots, c_n^{\mathcal{A}}$ of $A$ interpreting the constant symbols. Unless otherwise mentioned, all structures we will be dealing with are assumed to have finite universe.

The collection of first-order formulas over a language $\sigma$ is defined in the usual way as the smallest collection containing the atomic formulas, $t_i = t_j$ and $R_i(t_1, \ldots, t_{a_i})$ where $R_i$ is a relation symbol in $\sigma$ and each $t_j$ is either a constant symbol or a variable from a countable collection $\{x, y, \ldots\}$, and closed under the operations of negation ($\neg$), conjunction ($\wedge$), disjunction ($\vee$) and universal and existential quantification ($\forall$ and $\exists$, respectively).

The notion of a structure $\mathcal{A}$ satisfying a sentence (*i.e.* a formula without free variables), written $\mathcal{A} \models \phi$ is defined in the usual way (see, for instance, [End72]). We will sometimes write $\phi(x_1, \ldots, x_m)$ to denote a formula $\phi$ with free variables among $x_1, \ldots, x_m$, and $\mathcal{A} \models \phi[a_1 \ldots a_m]$ to indicate that $\mathcal{A}$ satisfies $\phi$ with the assignment of the elements $a_1, \ldots, a_m \in A$ to the variables $x_1, \ldots, x_m$. A model $\mathcal{A}$ satisfies a set of sentences $\Sigma$, written $\mathcal{A} \models \Sigma$ if and only if it satisfies every sentence in $\Sigma$, and we say that $\Sigma$ is satisfiable just in case it is satisfied by some model. The collection of structures that satisfy a given sentence $\phi$ is denoted $\mathrm{Mod}(\phi)$

Given any two structures $\mathcal{A}$ and $\mathcal{B}$ over the same signature $\sigma$, a mapping $f : A \rightarrow B$ is an isomorphism if it is *one-to-one* and *onto* and for every constant symbol $c$ in $\sigma$, $f(c^{\mathcal{A}}) = c^{\mathcal{B}}$ and for every relation symbol $R$ in $\sigma$ of arity $m$ and every $m$-tuple, $a_1, \ldots, a_m$, of elements of $A$, $\mathcal{A} \models R^{\mathcal{A}}(a_1, \ldots, a_m)$ if and only if $\mathcal{B} \models R^{\mathcal{B}}(f(a_1), \ldots, (a_m))$. If there is an ismorphism from $\mathcal{A}$ to $\mathcal{B}$, we say the two structures are isomorphic, written $\mathcal{A} \cong \mathcal{B}$. $\mathcal{A}$ and $\mathcal{B}$ are said to be *elementarily equivalent*, written $\mathcal{A} \equiv \mathcal{B}$ if they satisfy exactly the same sentences of first-order logic. Clearly, any two structures that are isomorphic are elementarily equivalent. When considering only finite structures, the converse holds as well. In fact, every finite structure is defined up to isomorphism by a single sentence of first-order logic:

**Proposition 1.1** *For every finite structure $\mathcal{A}$, there is a first-order sentence $\phi$ such that for any structure $\mathcal{B}$, $\mathcal{B} \models \phi$ if and only if $\mathcal{A} \cong \mathcal{B}$.*

Let $K$ be a collection of structures that is closed inder isomorphism, that is to say, if $\mathcal{A} \in K$ and $\mathcal{A} \cong \mathcal{B}$ then $\mathcal{B} \in K$. We will call $K$ a *query*. In order to encode the query $K$, we will generally identifyit with its subcollection, $K'$, where for each $\mathcal{A} \in K'$, the universe of $\dashv$ is a propoer initial segment of the integers.

## 1.2 Finite Model Theory

Model Theory is the study of the logical properties of mathematical structures. Classical Model Theory has concerned itself with both finite and infinite structures. That is to say the mathematical structures considered are not constrained to be one or the other. This is what one would expect, given that the subject was developed to deal with foundational issues in mathematics where elucidating the notion of infinity was central. Moreover, the mathematical structures that most mathematicians deal with are infinite objects. The situation is dramatically different in the field of computer science. The structures that computers manipulate, whether strings or graphs or trees or databases, are always finite. Furthermore, all of these data structures can be viewed as relational structures and hence we can talk about their logical properties.

It turns out however that the model theory of finite structures is very different from the classical subject. Most of the important tools and techniques of classical model theory do not carry over to finite model theory. This is a consequence of the fact that fundamental results such as the Compactness Theorem and the Completeness Theorem do not hold when restricted to finite structures. Another difficulty arises from the fact that first-order logic, the most widely-studied logic, proves inadequate in dealing with finite models. On the one hand it is intractable, that the sentences of first-order logic that are valid on finite structures do not form a recursively enumerable set. On the other hand, first-order logic is extremely limited in its expressive power. Many classes of finite structures that are interesting from a computational point of view are not expressible by sentences of this logic.

### 1.2.1 Failure of classical model theory

Perhaps the most important result of classical model theory is the Compactness Theorem. Many other results rely, either directly or indirectly, on this result. The theorem states that if any set of sentences $\Sigma$ has the property that every finite subset $\Delta \subseteq \Sigma$ is satisfiable, then $\Sigma$ is satisfiable. If we define satisfiable to mean satisfied in a finite structure, then this theorem is clearly false. To see this, let $\Sigma = \{\phi_i | i \in \mathbf{N}\}$ where $\phi_i$ is the first-order formula stating that there are at least $i$ elements in the universe. Every finite subset $\Delta$ of $\Sigma$ is satisfied in a structure of size $m$ where $m = \max\{i | \phi_i \in \Delta\}$, but any model of $\Sigma$ must be infinite.

More remarkable is the failure of Gödel's Completeness Theorem. While this theorem as usually stated asserts the completeness of some logical calculus, in its abstract form it can be taken as saying that validities of first-order logic, *i.e.* those sentences that are satisfied by all structures, form a recursively enumerable set. This is remarkable considering that the set is defined by a universal quantification over a proper class. It is all the more surprising therefore that the set of finite validities, *i.e.* those sentences true in all finite structures, is not recursively enumerable. This was shown by Trakhtenbrot[Tra50]. We present a proof of a strengthening of this result in Section 4.1.

Other basic results of model theory that have been shown to fail in the case of finite structures include Craig's Interpolation Theorem, Beth's Definability Theorem and various preservation theorems. For a survey of these results, see [Gur84].

### 1.2.2 Ehrenfeucht-Fraissé games

One set of techniques that do survive the transition to finite model theory in the most natural way are the games of Ehrenfeucht and Fraissé[Ehr61]. We shall have many occasions to use these techniques (and various extensions of them) in what follows. We give below an exposition of the

basic result. The proof is deferred until we consider an extension in Section 3.2. We begin with some definitions:

**Definition 1.1** *The* quantifier rank *of a formula $\phi$, written $qr(\phi)$ is defined inductively as follows:*

1. *if $\phi$ is atomic then $qr(\phi) = 0$,*
2. *if $\phi = \neg\psi$ then $qr(\phi) = qr(\psi)$,*
3. *if $\phi = \psi_1 \vee \psi_2$ or $\phi = \psi_1 \wedge \psi_2$ then $qr(\phi) = \max(qr(\psi_1), qr(\psi_2))$, and*
4. *if $\phi = \exists x\psi$ or $\phi = \forall x\psi$ then $qr(\phi) = qr(\psi) + 1$*

Intuitively, the quantifer rank of a formula is the maximum level of nesting of quantifiers within the formula.

**Definition 1.2** *A function $f$ is a* partial isomorphism *from $\mathcal{A}$ to $\mathcal{B}$ if the domain of $f$ is a subset of $\mathcal{A}$ that includes the interpretations of all constants in the language of $\mathcal{A}$ and if $f$ is an isomorphic map over this domain,* i.e. *$f(c^{\mathcal{A}}) = c^{\mathcal{B}}$ for all constants $c$ and for all relation symbols $R$ and $a_1, \ldots, a_m$ in the domain of $f$, $\mathcal{A} \models R^{\mathcal{A}}(a_1, \ldots, a_m)$ if and only if $\mathcal{B} \models R^{\mathcal{B}}(f(a_1), \ldots, f(a_m))$.*

**Definition 1.3** *A sequence $I_0, \ldots, I_n$ of sets of partial isomorphisms from $\mathcal{A}$ to $\mathcal{B}$ has the* back and forth property through *$n$ if for $0 \leq i < n$,*

1. *$I_i$ is non-empty,*
2. *$I_i \subseteq I_{i+1}$,*
3. *for every $f \in I_i$ and $a \in A$ there is a $g \in I_{i+1}$ such that $f \subseteq g$ and $a \in dom(g)$, and*
4. *for every $f \in I_i$ and $b \in B$ there is a $g \in I_{i+1}$ such that $f \subseteq g$ and $b \in rng(g)$.*

The result can now be stated as:

**Theorem 1.1** *There is a sequence of sets of partial isomorphisms from $\mathcal{A}$ to $\mathcal{B}$ with the back and forth property through $n$, if and only if for any sentence, $\phi$, with $qr(\phi) \leq n$, $\mathcal{A} \models \phi$ if and only if $\mathcal{B} \models \phi$.*

One way to look at this result is in terms of the following two-player pebble game. We have a board consisting of one copy of each of the structures $\mathcal{A}$ and $\mathcal{B}$. There is also an infinite supply of pairs of pebbles $\{\langle a_i, b_i\rangle | i \in \mathbf{N}\}$. At move $i$ of the game, Player I picks up one of the pair of pebbles $\langle a_i, b_i\rangle$ and places it on an element of the corresponding structure (*i.e.* she places $a_i$

on an element of $A$ or $b_i$ on an element of $B$). Player II then responds by placing the unused pebble in the pair on an element of the other structure. Player II loses if the resulting map, $f$, from $\mathcal{A}$ to $\mathcal{B}$, given by $f(a_j) = b_j, j \leq i$, is not a partial isomorphism. Player II wins the $n$-move game if she has a strategy to avoid losing in the first $n$ moves, regardless of what moves are made by Player I. It should be clear that such a winning strategy corresponds to a sequence of sets of partial isomorphisms with the back and forth property through $n$. As a corollary, if Player II can play the game indefinitely without losing, then the two structures are elementarily equivalent.

## 1.3 Descriptive Complexity

In one sense first-order logic is too strong when restricted to finite models, as demonstrated by the failure of the Completeness theorem. In another sense, however, it is too weak. It fails to express many properties that are of interest. More precisely, there are classes of structures that are easily recognizable (*i.e.* with low computational complexity) that cannot be described in first-order logic. There has been much recent work done in matching the expressive power of various logics to known complexity classes, and this has focussed on logics that are stronger than first-order logic. We formalize these notions below.

We can speak of the computational complexity of a collection of structures $K$ in the following sense: we choose some fixed reasonable encoding of structures of signature $\sigma$ into strings over some alphabet (for simplicity, we will generally assume that they are binary strings). We then identify the complexity of recognising $K$ with the complexity of the decision problem for $S$, the set of strings that are encodings of structures in $K$. We will speak of the complexity measure as being a function of the size of the structure. Since the strings encoding structures are at most polynomialy larger than the structures (to be precise, their size is bounded by $n^k$, where $k$ is the maximum of the arities of the relation symbols in $\sigma$), this does not cause a problem.[1]

Another notion of complexity is that of the complexity of describing a collection of structures $K$. For instance, if there is a sentence of first-order logic $\phi$ such that $K = \text{Mod}(\phi)$, then $K$ is expressible in first-order logic. Hence, we can look at the class of decision problems on structures of a given signature $\sigma$ expressible in first-order logic as a complexity class, which we will denote FO. This raises the question of what the correspondence is between complexity classes defined in the usual way in terms of resource bounded computing devices and the classes defined by the expressive power of various logics.

It is fairly easy to show that FO is contained in DSPACE[log($n$)]. To see this, let $\phi$ be any

[1] In the special case where $\sigma = \{\}$, the string can be of size log($n$). We will mention this explicitly when we encounter it.

first-order sentence and let $k$ be an upper bound on the number of free variables in any sub-formula of $\phi$. Let $\mathcal{A}$ be any structure of size $n$. To check whether $\mathcal{A} \models \phi$, we proceed recursively through the sub-formulas of $\phi$. At any given stage, we need to check at most $n^k$ assignments to the free variables of the sub-formula. These can be encoded using $k \log(n)$ bits. Furthermore, as we shall see, there are some fairly elementary properties in DSPACE[$\log(n)$] that are not in FO, such as *parity* – the collection of structures of even size.

### 1.3.1 Fagin's theorem

The earliest result relating computational complexity to the expressive power of logic was that of Fagin[Fag74] who showed that the class NP is equivalent to the properties expressible in existential second-order logic.

Second-order logic is the extension of first-order logic with second-order quantifiers. That is, we have countably many additional relational variables $P_i$ with associated arities $a_i$ and, for instance, the formula $\exists P_i \phi$ is true in a structure a $\mathcal{A}$ just in case for every interpretation of $P_i$ as a relation in $A^{a_i}$, $\phi$ holds in $\mathcal{A}$. It is easy to see that any second-order formula can be transformed (possibly increasing the arity of the quantified relations) into an equivalent one in prenex form, with all the second-order quantifiers in front. A formula in existential second-order logic is a second-order formula with no universal second-order quantifiers in prenex form. Existential second-order sentences are also called $\Sigma_1^1$ sentences, and by extension, a class of structures is called $\Sigma_1^1$ if it is expressible by a $\Sigma_1^1$ sentence.

As an example, we can express the property of a graph being 3-colourable with a $\Sigma_1^1$ sentence in the language of graphs $\sigma = \{E\}$. The sentence is $\exists P_1 \exists P_2 \exists P_3 \phi$ where the $P_i$ are unary relations (colours) and $\phi$ is a first-order sentence in the language $\{E, P_1, P_2, P_3\}$ asserting that each element has a unique colour and no two elements that are connected by an edge have the same colour.

**Theorem 1.2 (Fagin)** $\Sigma_1^1 = NP$

**Proof:**

To see that any property defined by a $\Sigma_1^1$ sentence $\exists R_1 \ldots \exists R_m \phi$, where $\phi$ is first-order is in NP, note that a relation $R^{\mathcal{A}}$ of arity $k$ has cardinality at most $n^k$, where $n$ is the size of the structure $\mathcal{A}$. Thus, a non-deterministic machine, on input $\mathcal{A}$ can guess the interpretations of the relations $R_1, \ldots, R_m$ on $\mathcal{A}$ and each of these is an object whose is polynomial in the size of the input. Given these interpretations, the task of verifying that the first-order formula $\phi$ holds in the expanded structure can be done in DSPACE[$\log(n)$], as noted above.

In the other direction, let $N$ be a non-deterministic machine that accepts inputs $\mathcal{A}$ of size $n$

over the signature $\sigma$ in time $n^k$. We will asume that the input to the machine is in the form of a binary string encoding the structure. We can write a first order sentence $\phi$ over the signature $\sigma \cup \{O,T,S\}$ ($O$ and $T$ being relation symbols of arity $2k$ and $S$ of arity $3k$) to encode the machine $N$. $O$ is to be taken as a total ordering on tuples of length $k$ from the universe of the input structure. Using this ordering, these tuples can then be used to represent the integers $0,\ldots,n^k-1$. $T$ is used to encode the contents of the tape and $S$ the state of the machine. $T(n_1,n_2)$ is true just in case the content of tape cell $n_2$ at time $n_1$ is a 1. $S(n_1,n_2,n_3)$ is true just in case at time $n_1$, the machine is in state $n_2$ with the head reading cell $n_3$. The sentence $\phi$ says that these represent an accepting computation of the machine $N$. That is to say, it asserts the following:

1. $O$ is a total ordering,
2. $T(0,0),\ldots,T(0,n^k-1)$ represents an encoding of the input structure,
3. given $S(x,y,z)$ and $T(x,0),\ldots,T(x,n^k-1)$, there is a move in the transition function of $N$ yielding $S(x+1,y',z')$ and $T(x+1,0),\ldots,T(x+1,n^k-1)$, and
4. $S(n^k,y,z)$ is an accepting state.

Then, the $\Sigma^1_1$ sentence $\exists O\exists T\exists S\phi$ is true in exactly those structures accepted by $N$. ■

As a corollary to this theorem, we have the result that full second-order logic expresses exactly the queries in the polynomial hierarchy[Sto77], with the levels of the hierarchy corresponding exactly to the number of second-order quantifier alternations.

### 1.3.2 Inductive logic

Another way of extending first-order logic is by adding some kind of induction operation. For instance, consider the class of graphs that are connected. It is possible to show that this is a class that is not expressible in first-order logic by an application of the Ehrenfeucht-Fraissé game (see, for instance, [Gur84]). On the other hand, it is clear that connectedness of a graph can be expressed by the sentence $\forall x\forall y R(x,y)$ where $R$ is the reflexive and transitive closure of the edge relation. While such an $R$ cannot be defined in first-order logic, it seems it can be defined by an induction, because it is the smallest relation satisfying the equivalence:

$$R(x,y) \equiv x = y \vee \exists z(E(x,z) \wedge R(z,y)) \tag{1.1}$$

In general, let $\phi(R,x_1,\ldots,x_k)$ be a first-order formula over the signature $\sigma \cup \{R\}$ with free variables $x_1,\ldots,x_k$ where $k$ is the arity of $R$. For any structure $\mathcal{A}$ over the signature $\sigma$, $\phi$ defines a mapping, $\Phi$ on relations of arity $k$ in the following sense — given a relation $R^{\mathcal{A}} \subseteq |\mathcal{A}|^k$, let

$\langle \mathcal{A}, R^{\mathcal{A}} \rangle$ be the expansion of $\mathcal{A}$ interpreting $R$ as $R^{\mathcal{A}}$. Then, $\Phi(R^{\mathcal{A}}) = \{\langle a_1, \ldots a_k \rangle | \langle \mathcal{A}, R^{\mathcal{A}} \rangle \models \phi(R, x_1, \ldots, x_k)\}$

This map $\Phi$ is called monotone if for any relations $R$ and $S$ such that $R \subseteq S$, $\Phi(R) \subseteq \Phi(S)$. A map that is monotone has a least fixed point, *i.e.* a smallest relation $R$ such that $\Phi(R) = R$. Moreover, this least fixed point can be obtained by the following iterative construction: Let $\Phi^0 = \emptyset$ and $\Phi^{m+1} = \Phi(\Phi^m)$. Then for some $m$ (depending on the structure $\mathcal{A}$), $\Phi^{m+1} = \Phi^m =$ the least fixed point of $\Phi$. $m$ is called the *closure ordinal* of $\Phi$ on the structure $\mathcal{A}$. If $n$ is the size of $\mathcal{A}$, then there are $n^k$ $k$-tuples in $\mathcal{A}$ and since $\Phi$ is monotone, $m \leq n^k$.

A sufficient syntactic condition for the formula $\phi$ to define a monotone map on all structures is that $\phi$ be *positive* in $R$, that is to say that all occurrences of $R$ in $\phi$ be within the scope of an even number of negations. We can now define the logic FO + LFP over signature $\sigma$ as the smallest set of formulas satisfying:

- if $\phi$ is first-order formula over $\sigma$, then $\phi \in$ FO + LFP($\sigma$),
- if $\phi$ is formed from formulas in FO + LFP($\sigma$) by conjunction, disjunction and first-order quantification, then $\phi \in$ FO + LFP($\sigma$), and
- if $\phi \in$ FO + LFP($\sigma \cup \{R\}$), $\phi$ is positive in $R$ and $x_1, \ldots, x_k$ are distinct variables, where $k$ is the arity of $R$, then $\mathbf{lfp}(R, x_1 \ldots x_k)\phi(t_1 \ldots t_k) \in$ FO + LFP($\sigma$) for any terms $t_1, \ldots, t_k$.

The way to read the last clause above is that the operator **lfp** binds the second order variable $R$ and the first-order variables $x_1, \ldots x_k$ in $\phi$ to form a new predicate. This predicate is to be interpreted as the $k$-ary relation that is the least fixed point of the monotone operator defined by $\phi$. This predicate is then evaluated at the elements $t_1, \ldots, t_k$.

As an example, let $\phi(R, x, y)$ be $x = y \vee \exists z(E(x, z) \wedge R(z, y))$, the right side of 1.1 above. Then, $\mathbf{lfp}(R, x, y)\phi(x, y)$ is a formula in two free variables that expresses the transitive closure of the edge relation on any graph.

Immerman [Imm86] and Vardi [Var82] independently showed that when we include a total ordering on the domain as part of the logical vocabulary, the language FO + LFP expresses exactly the class of polynomial time computable queries.

**Theorem 1.3** *FO + LFP with ordering = P*

**Proof sketch:**

One direction is easy. As we observed above, the number of iterations needed to compute the least fixed point of a $k$-ary formula is at most $n^k$ in a structure of size $n$. Since each iteration involves the evaluation of a first-order formula, this can be done in DSPACE[log($n$)]. Thus, the total time taken is polynomial in $n$.

In the other direction, we are given a machine $M$ running in time $n^k$. Note that with the help of the ordering relation we can define an ordering on $k$-tuples which, as in the proof of Theorem 1.2 can then be used to count up to $n^k$. Also, as before, we can write a formula $\phi(C)$ in the second order variable $C$ such that if the relation $C$ encodes a computation of $M$ for $m$ steps, $\phi(C)$ is a relation encoding the computation for $m+1$ steps. Then, the least fixed point of $C$ encodes the entire computation of $M$. ■

The following normal form result was also established in [Imm86] for FO + LFP even in the absence of an ordering,

**Theorem 1.4** *In any vocabulary containing constant symbols, every formula in FO + LFP is equivalent to a formula* $\mathbf{lfp}(R, \bar{x})\phi(\bar{t})$, *where* $\phi$ *is first-order.*

We saw above how a formula with one free predicate variable defined an operator on relations. This, of course, is true even when the formula is not positive in the predicate variable and the operator, in turn may or may not be monotone. Moreover, the iterative stages of the operator can still be defined, though they are not guaranteed to converge to a fixed point in the case of non-monotone operators. Let $\phi(R, \bar{x})$ be a formula that defines a (possibly non-monotone) operator $\Phi$. Define the *partial fixed point* of $\phi$ to be $\Phi^m$ if there is an $m$ such that $\Phi^{m+1} = \Phi^m$, and empty otherwise. We can then define another extension of first-order logic called FO + PFP with a syntax similar to that of FO + LFP except that the **lfp** operation is replaced by **pfp**, which can operate on arbitrary formulas, not just positive ones. $\mathbf{pfp}(R, \bar{x})\phi$ denotes the partial fixed point of $\phi$.

It has been shown in [AV91a] that the language FO + PFP is equivalent to the query language *while* – an extension of first-order logic with an iterative operation. Putting this together with a result of Vardi [Var82], we get the following:

**Theorem 1.5** *FO + PFP with ordering = PSPACE*

# Chapter 2

# A logic for P

As we have seen, the logic FO + LFP expresses exactly the P-time properties of structures with a built-in ordering. However, in the absence of ordering, there are collections of structures that are recognizable in polynomial time that are not expressible in this logic. This raises the quesion whether there is a logic that expresses exactly the P-time properties of all structures. This is the question we examine in this chapter.

In showing that any polynomial time computable property of ordered structures can be expressed in FO + LFP, we crucially used the ordering relation in order to simulate the computation of the machine. This was not required as an additional relation in the proof of Theorem 1.2 because we can always non-deterministically choose an ordering in $\Sigma^1_1$. All known results in descriptive complexity that characterize complexity classes below NP in terms of a logical language require a total ordering as part of the logical vocabulary. This fact appears to be related to the fact that the problem of canonical labeling of graphs is known to be in NP (see, for instance, [Kuč87]), but not in any lower complexity class. We turn to these considerations next.

## 2.1 Unordered Structures

The question that we raised – is there a natural logic that expresses the P-time properties of all structures – is, of course, not precisely formulated, since we have not defined what a natural logic is. However, this can be seen as part of the broader question – are the P-time properties of all structures recursively enumerable? We can certainly enumerate all Turing machines that run in time polynomial in the size of the input. This can be acheived by including a clock in the definition of the machine. But, not all these machines compute properties of structures.

For concreteness, consider graphs, *i.e.* structures over the signature $\{E\}$. To serve as input to the Turing machines, graphs can be encoded as binary strings. For a graph of size $n$, number

the vertices 1 through $n$ and take the $n \times n$ adjacency matrix – where bit $\langle i, j \rangle$ is 1 iff there is an edge from vertex $i$ to vertex $j$ – and enumerate the matrix in row-major order, giving us an input size $n^2$. However, the same graph can be encoded in many different ways (up to $n!$) and some machines will accept some encodings of a given graph but not others. We say a Turing machine accepts a graph property if it does not break isomorphism classes of graphs, that is to say it either accepts or rejects all encodings of any given graph. We can now restate the question as follows: is there a recursive indexing of the Turing machines that run in polynomial time and accept graph properties?

We chose the language of graphs for a reason. Structures over any given signature $\sigma$ can be encoded as graphs while preserving isomorphism. Moreover this can be done in polynomial time. In fact, the translation of arbitrary structures into graphs is first-order definable. That is to say, for a given signature, $\sigma$, there are first order formulas $\phi_\forall$ and $\phi_E$ in the language $\sigma$ that define a graph $G(\mathcal{A})$ for any structure $\mathcal{A}$. $\phi_\forall$ defines the set of vertices and $\phi_E$ defines the edge relation and $G(\mathcal{A}) \cong G(\mathcal{B})$ *if and only if* $\mathcal{A} \cong \mathcal{B}$ (see, for instance, [Lin87]). Thus, the problem of the recursive enumerability of the P-time properties of arbitrary structures reduces to that for graphs.

We can look at the same problem in another way. The binary string encoding of graphs described above can be seen as encoding an ordered graph, with the ordering given by the natural ordering on the integers. In this way, each ordered graph gives a unique binary string and hence no Turing machine breaks isomorphism classes. Hence, we know that we can enumerate all the P-time properties of ordered graphs. We also obtain such an enumeration simply by enumerating all the sentences of FO + LFP over the signature $\{E, <\}$. The subset of this list in which the symbol $<$ does not occur, is the set of FO + LFP sentences in the language of graphs, and we know that this does not exhaust all the P-time properties of graphs. Therefore, there must be sentences in the enumeration that use the symbol $<$ but are order-invariant, that is their truth value is not affected by the choice of ordering to interpret this symbol. Thus, another way of formulating the problem we are considering is – is there a recursive enumeration of the order-invariant sentences of FO + LFP up to equivalence.

We can actually show that there is no recursive enumeration of *all* the order-invariant sentences of FO + LFP. This is accomplished by a simple reduction from the set of validities, which we know from Trakhtenbrot's result are not r.e. (see Section 4.1). Let $\psi$ be the first-order sentence in the language $\{P, <\}$, $P$ as unary relation symbol, that says that $<$ is an ordering and that the first element under this ordering is not in $P$ and the last one is. Then, for any sentence $\phi$ whose vocabulary is disjoint from that of $\psi$, $\phi$ is valid if and only if $\phi \vee \psi$ is order-invariant. This, of course, still leaves open the question of whether we can provide a recursive indexing, up

to equivalence, of the order-invariant sentences.

## 2.2 The Canonization Problem

One way to overcome the problem outlined above is to find a way of encoding graphs as binary strings in an isomorphism invariant way. Equivalently, we want to find a uniform way of ordering the vertices of any graph. Clearly, we cannot define such an ordering in FO + LFP, since that would imply that we could express all polynomial time properties in FO + LFP. It is still conceivable, though, that we could do so in polynomial time.

The problem of giving a canonical ordering on the vertices of a grph is known as the graph canonization problem. It is closely related to the graph isomorphism problem, *i.e.* given two graphs determine if they are isomorphic. Neither of these problems is known to have a polynomial time solution. The best known algorithms run in mildly exponential time. However, both the problems are in NP and neither has been shown to be NP-complete.

A polynomial-time algorithm for canonical labeling of graphs would establish that the P-time properties of graphs are indeed r.e. Consider an enumeration, $M_0, \ldots, M_i, \ldots$ of all polynomial time Turing machines, as before, and let $M$ be a machine running in polynomial time that takes its input graph (in any order) and produces as output the same graph in canonical order. Then, enumerate the machines $M \rightarrow M_i$ which run $M$ on the input and then run $M_i$ on the output produced by $M$. This enumeration contains only graph properties and it contains all P-time graph properties because the original listing does.

While we saw in Theorem 1.5 that the logic FO + PFP expresses the PSPACE properties of ordered structures, it fails, like FO + LFP to express some simple queries on unordered structures, including *parity*. However, by an argument like the one above, we know that the PSPACE properties are recursively enumerable because the canonization problem is in NP and hence in PSPACE. Having observed this, we obtain the following logic for PSPACE:

**Proposition 2.1** *The PSPACE properties of unordered structures are expressed exactly by formulas $\exists R_1 \ldots \exists R_m \phi$, where the $R_i$ are second-order variables and $\phi$ is in FO + PFP.*

To evaluate a formula as above in a given structure, we can non-deterministically guess the interpretations of the quantified predicate variables (each interpretation being of size polynomial in the size of the structure) and then evaluate $\phi$. Since the latter step we know to be in PSPACE, the entire operation is in NPSPACE and hence in PSPACE. In the other direction, since we can choose an ordering with an existential second-order quantifier and FO + PFP expresses all of PSPACE on ordered structures, we are done.

## 2.3 Limitations of the Logic

We mentioned earlier that the *parity* query, *i.e.* the set of structures with a universe of even size is not expressible in FO + LFP. We shall now prove this fact.

### 2.3.1 The $k$-pebble game

The technique we use to show the limitation on the expressive power of FO + LFP is a modification of the pebble games introduced in Section 1.2.2. As in the earlier version, two players play on two structure $\mathcal{A}$ and $\mathcal{B}$ by placing pebbles on elements of the structures, except now the supply of pebbles is limited to $k$ pairs $\{\langle a_1, b_1\rangle, \ldots, \langle a_k, b_k\rangle\}$. Once all $k$ pairs have been placed on the board, Player I can move by picking one of the pebbles already on the board and placing it on any element in the same structure. Player II responds by picking the matching pebble in the other structure and placing it on an element of that structure. As before, Player I wins at any stage if the map from $\mathcal{A}$ into $\mathcal{B}$ defined by the pebble pairs is not a partial isomorphism.

**Lemma 2.1** *If Player II has a winning strategy for $n$ moves of the $k$ pebble game on structures $\mathcal{A}$ and $\mathcal{B}$, then $\mathcal{A}$ and $\mathcal{B}$ agree on all first-order sentences of quantifier rank up to $n$ with at most $k$ distinct variables.*

This result is a corollary of Theorem 3.1 in Section 3.2.

It follows from the above that if Player II can play the $k$-pebble game indefinitely without losing, then the two structures agree on all first-order sentences with at most $k$ distinct variables. Call this fragment of first-order logic $L_k$. To use the above game to establish results about FO + LFP, we need the following lemma:

**Lemma 2.2** *If two structures agree on all sentences of $L_k$, then they agree on all sentences of FO + LFP with at most $k$ distinct variables.*

**Proof:**

If two structures differ on a sentence $\phi \equiv \mathbf{lfp}(R)\psi(R)$, for some first-order $\psi$, then they differ on some $\Phi^m$, *i.e.* some iterate of the operator defined by $\phi$. Each of the $\Phi^m$ can be written out as a first-order formula as follows:

$$\begin{aligned} \Phi^0 &\equiv \neg(x = x) \\ \Phi^{m+1} &\equiv \psi(\Phi^m)\text{obtained from } \psi \text{ by replacing every occurrence of } R \text{ by } \Phi^m \end{aligned}$$

Observe that these formulas have no more variables that $\phi$. Then, if the two structures differ on any formula $\theta$ in FO + LFP, we can replace all occurrences of **lfp** in $\theta$ to get an $L_k$ formula distinguidhing them. ■

### 2.3.2 The parity query

To show that parity is not definable in FO + LFP, we only need to show that for every $k$, there are two structures $\mathcal{A}_k$ and $\mathcal{B}_k$ such that the universe of $\mathcal{A}_k$ has even cardinality, the universe of $\mathcal{B}_k$ has odd cardinality and the two structures agree on all sentences of $L_k$.

Let $\mathcal{A}_k$ and $\mathcal{B}_k$ be structures over the pure identity language, *i.e.* the language with no non-logical vocabulary, with $\text{card}(\mathcal{A}_k) = 2k$ and $\text{card}(\mathcal{B}_k) = 2k + 1$. Then, since any map on at most $k$ points from $\mathcal{A}_k$ into $\mathcal{B}_k$ is clearly a partial isomorphism, Player II can, indeed,play the $k$-pebble game indefinitely on these two structures.

### 2.3.3 The problem of counting

In general, the kind of argument we exhibited above can be used to show that FO + LFP fails to express any notion of cardinality. For a long time, these were the only known examples of properties that are in P-time but are not expressible in FO + LFP. This led to the conjecture that the addition of some kind of counting construct to the language might capture all of P.

Lindell[Lin91] showed, however, that there are polynomial time computable properties of complete binary trees that are not expressible in FO + LFP[1]. This is significant because FO + LFP does express cardinality on complete binary trees in the following sense. There is a formula $\phi(x, S)$ of FO + LFP with the first-order variable $x$ and the unary predicate variable $S$ free, such that for any complete binary tree $T$, the set $\{v \in T | \langle T, S^T \rangle \models \phi[v]\}$ is uniquely determined by the cardinality of $S$.

## 2.4 Counting Quantifiers

One kind of construct to express counting is the counting quantifier described in [Imm87]. For each integer $i$ let $\exists i$ be a new quantifier in the language. This quantifier says that there are $i$ elements – for instance, $\exists i x \phi(x)$ says there are $i$ elements in the set defined by $\phi(x)$. Clearly, for any given $i$, this could also have been stated by a first-order formula. Therefore, adding these counting quantifiers to the language FO + LFP does not appear to increase the expressive power of the language. To see their value, we need to consider an alternative characterization of inductive logic.

Every formula $\phi$ of FO + LFP can be written out as a uniform infinite sequence of FO formulas $\psi_n$ such that: there is a constant $c$ such that the length of $\psi_n$ is less than $n^c$; for all structures of size $n$ or less, $\phi$ and $\psi_n$ agree; and there is a constant $k$ such that $\psi_n$ has no more than $k$ distinct variables. The condition on uniformity in the above can be formulated purely

[1] We return to this topic in some detail in Section 4.3.4, where a proof of this fact is also given.

syntactically (see [Imm87]). Call the class of queries definable by such sequences IND. In the presence of ordering, IND = FO + LFP [Imm86].

We now define IND(C) to be the properties definable by uniform sequences, $\phi_n$ as above, but where $\phi_n$ may contain counting quantifiers. Note that this is not the same as adding counting quantifiers to FO + LFP since we could have increasing quantifiers in $\phi_n$ as $n$ grows, and a translation of these quantifiers into first-order formulas would give a sequence with no bound on the number of variables. In fact, we can express the parity query in IND(C).

It was conjectured that IND(C) = P [Imm87]. If this were true, it would imply that every query $q \in P$ is closed under $C_k$ equivalence for some $k$, that is, for every structure $\mathcal{A} \in q$ and any structure $\mathcal{B}$ such that $\mathcal{A}$ and $\mathcal{B}$ agree on all first-order sentences with counting quantifiers and at most $k$ distinct variables, $\mathcal{B} \in q$. This conjecture was refuted by Cai, Fürer and Immerman in [CFI89]. They exhibited a set of graphs $S$ recognizable in polynomial time on which the isomorphism problem is in P, and a sequence of pairs $\langle G_n, H_n \rangle$ of graphs from $S$ such that $G_n$ and $H_n$ are $C_n$ equivalent but not isomorphic. This means that the isomorphism query on $S$ defined as the query $q$ over the language $\{E_1, E_2\}$ of two binary relations such that $q = \{\langle V, E_1, E_2 \rangle |$ the graphs $\langle V, E_1 \rangle$ and $\langle V, E_2 \rangle$ are in $S$ and are isomorphic$\}$ is in P but is not closed under $C_k$ equivalence for any $k$.

# Chapter 3

# Infinitary Logic with a Bounded Number of Variables

We noted in the previous chapter that formulas of the logic FO + LFP can be seen as uniform infinite sequences of first-order formulas with a bounded number of variables. This suggests another way of extending first-order logic that includes the extension by a least-fixed-point operator, and that is by including infinitary syntax. In this chapter, we look at the restriction of infinitary logic obtained by allowing only finitely many variables in any given sentence. This serves as a natural extension of inductive logics and provides a tool for studying their expressive power.

## 3.1 Infinitary Logic

We first define the syntax of full infinitary logic. This language is denoted $L_{\infty\omega}$, the first subscript indicating that conjunctions and disjunctions can be taken over arbitrary sets of formulas and the second subscript that only finite quantifier blocks are allowed[1]. In this notation, first-order logic would be $L_{\omega\omega}$. The formulas of $L_{\infty\omega}$ are defined as for first-order logic, except that conjunction and disjunction are no longer binary operations. Rather, for any set of infinitary formulas $\Phi$, $\bigvee\Phi$ and $\bigwedge\Phi$ are both formulas of $L_{\infty\omega}$.

$L_{\infty\omega}$ is complete in expressive power in the following sense. Consider any class of finite structures $\mathcal{C}$ such that $\mathcal{C}$ is closed under isomorphism. Since any finite structure $\mathcal{A}$ is completely characterized up to isomorphism by a first-order sentence, $\phi_{\mathcal{A}}$, $\mathcal{C}$ is expressed by the $L_{\infty\omega}$ sentence $\bigvee\{\phi_{\mathcal{A}} \; \mathcal{A} \in \mathcal{C}\}$. Clearly, this language is too strong. One restriction of this language that has

[1]The notation for $L_{\infty\omega}$, $L^k_{\infty\omega}$ and $L^\omega_{\infty\omega}$ is borrowed from [Bar77] where the latter two were first introduced.

been studied is obtained by allowing only finitely many variables in any single formula.

**Definition 3.1** *$L^k_{\infty\omega}$ is the collection of formulas of $L_{\infty\omega}$ that have at most k distinct variables (free or bound). $L^\omega_{\infty\omega}$ is the collection of formulas of $L_{\infty\omega}$ that have a finite number of distinct variables.*

$$L^\omega_{\infty\omega} = \bigcup_{k=1}^{\infty} L^k_{\infty\omega}$$

The language $L^\omega_{\infty\omega}$ is restricted in its expressive power when compared with $L_{\infty\omega}$, yet it is still powerful enough to express properties that are not recursive. Consider, for instance, the following recursive definition of the first order formulas $P_n$ in the language, $\{\leq, U\}$ of binary strings[2]:

$$P_0(x) \equiv \neg U(x) \wedge \forall y(y \leq x \rightarrow \neg U(y))$$

$$P_1(x) \equiv U(x) \wedge \forall y(y \leq x \rightarrow \neg U(y))$$

$$P_{2n}(x) \equiv \neg U(x) \wedge \exists y(y \leq x \wedge \forall z(z < x \rightarrow z \leq y) \wedge \exists x(x = y \wedge P_n(x)))$$

$$P_{2n+1}(x) \equiv U(x) \wedge \exists y(y \leq x \wedge \forall z(z < x \rightarrow z \leq y) \wedge \exists x(x = y \wedge P_n(x)))$$

Over a binary string $b_1 \ldots b_m$, $P_n(b_i)$ is true if and only if the string $b_1 \ldots b_i$ is the binary representation of $n$. Hence, $\exists x(\forall y(y \leq x) \wedge P_n(x))$ is true only in the binary string representing $n$. Note that only three distinct variables are used in any of the formulas. Thus, for any set of integers $S$, the formula $\bigvee_{n \in S} P_n(x)$ is in $L^3_{\infty\omega}$. Since $S$ could be non-recursive, $L^\omega_{\infty\omega}$ can express non-recursive properties. On the other hand, as we shall see shortly, parity is not expressible in this language.

## 3.2 Pebble Games

We showed above that unrestricted infinitary logic, $L_{\infty\omega}$, is complete in its expressive power over finite structures. We also showed that the restricted version of infinitary logic $L^\omega_{\infty\omega}$ can express non-recursive properties. To show that the restriction is real, we need to exhibit some property that cannot be expressed in the latter language. To this end, we now present a version of the Ehrenfeucht-Fraissé games.

We state and prove the following result in its full generality. In particular, the theorem, as stated, is true for *all* structures, not just finite ones. We will then consider the special cases that are of interest. For instance, Theorem 1.1 is a special case.

[2]It is assumed that the symbol $\leq$ represents a total ordering on the domain. This could be stated in a first-order formula with 3 variables and added as a clause to the formulas $P_n$

As before, $qr(\phi)$ denotes the ordinal that is the quantifier rank of the formula $\phi$. Note that for infinitary formulas, the quantifier rank may be a transfinite ordinal. For this, we need to modify the definition of quantifier rank (Definition 1.1). Specifically, the clause for conjunction and disjunction (clause 3) should read: if $\phi = \bigvee \Phi$ or $\phi = \bigwedge \Phi$ then $qr(\phi) = \sup\{qr(\psi) | \psi \in \Phi\}$. $dom(f)$ denotes the domain of the function $f$, $rng(f)$ its range and $|f|$ its cardinality. We also assume that any formula in $L^k_{\infty\omega}$ is written so as to use only the variables $x_0, \ldots, x_{k-1}$

**Theorem 3.1** *For any two structures, $\mathcal{A} = \langle A, \ldots \rangle$, $\mathcal{B} = \langle B, \ldots \rangle$ in a purely relational language, the following statements are equivalent:*

1. *For all sentences $\phi \in L^k_{\infty\omega}$ with $qr(\phi) \leq \alpha$,*

   $$\mathcal{A} \models \phi \text{ iff } \mathcal{B} \models \phi$$

2. *There is a collection $\{I_\beta \mid \beta \leq \alpha\}$ of non-empty sets of partial isomorphisms from $\mathcal{A}$ to $\mathcal{B}$ such that:*

   *(a)* $I_0 \supseteq I_1 \supseteq \ldots \supseteq I_\beta \supseteq \ldots \supseteq I_\alpha$,

   *(b) If $f \in I_\beta$ $(0 \leq \beta \leq \alpha)$ and $g \subseteq f$ then $g \in I_\beta$, and*

   *(c) For every $f \in I_{\beta+1}$ $(0 \leq \beta < \alpha)$ such that $|f| < k$ and every $a \in A$ (resp. $b \in B$), there is a $g \in I_\beta$ with $f \subseteq g$ and $a \in dom(g)$ (resp. $b \in rng(g)$).*

**Proof:**

**(2 $\Rightarrow$ 1)** We show by induction on $\beta$ that for formulas $\phi(y_0 \ldots y_m) \in L^k_{\infty\omega}$, with $qr(\phi) \leq \beta$, if $f \in I_\beta$ and $a_0, \ldots, a_m \in dom(f)$, then $\mathcal{A} \models \phi[a_0 \ldots a_m]$ iff $\mathcal{B} \models \phi[f(a_0) \ldots f(a_m)]$.

*Basis*:

If $qr(\phi) = 0$ then $\phi$ is a boolean combination of atomic formulas and since $f$ is a partial isomorphism, the result follows.

*Induction Step*:

We now proceed by induction on the structure of the formula $\phi$. The cases $\phi = \neg\psi$ and $\phi = \bigwedge_{j \in J} \psi_j$ are trivial. So, we only need to consider the case where $\phi = \exists y_0 \psi[y_0 \ldots y_m]$. Note that $qr(\phi) = \delta + 1$ where $qr(\psi) = \delta$.

Suppose $\mathcal{A} \models \phi[a_1 \ldots a_m]$ for some $a_1, \ldots, a_m \in A$ and that $a_1, \ldots, a_m \in dom(f)$ for some $f \in I_{\delta+1}$. Then, there is an $a_0 \in A$ such that $\mathcal{A} \models \psi[a_0 a_1 \ldots a_m]$. By *(b)* above, there is an $f' \in I_{\delta+1}$ with $dom(f') = \{a_1, \ldots, a_m\}$. Since $|f'| < k$, by *(c)* there is a $g \in I_\delta$ extending $f'$ such that $a_0 \in dom(g)$. But then, by the induction hypothesis, $\mathcal{B} \models \psi[g(a_0) g(a_1) \ldots g(a_m)]$,

*i.e.* $\mathcal{B} \models \phi[g(a_1) \ldots g(a_m)]$ and therefore $\mathcal{B} \models \phi[f(a_1) \ldots f(a_m)]$, since $f$ and $g$ agree on $a_1, \ldots, a_m$.

Similarly, if $\mathcal{B} \models \phi[b_1 \ldots b_m]$ and $b_1, \ldots, b_m \in rng(f)$, then $\mathcal{A} \models \phi[f^{-1}(b_1) \ldots f^{-1}(b_m)]$.

**(1 $\Rightarrow$ 2)** Define the $I_\beta$ as follows: $f \in I_\beta$ *if and only if* $f$ is a partial isomorphism from $\mathcal{A}$ to $\mathcal{B}$ and for all formulas $\phi \in L^k_{\infty\omega}$ with $qr(\phi) \leq \beta$ and all $a_1 \ldots a_m \in dom(f)$, $\mathcal{A} \models \phi[a_0 \ldots a_m]$ iff $\mathcal{B} \models \phi[f(a_0) \ldots f(a_m)]$.

By definition, $I_\delta \supseteq I_\beta$ for $\delta \leq \beta$. Also, since $\mathcal{A}$ and $\mathcal{B}$ agree on all sentences of quantifier rank up to $\alpha$, the empty partial isomorphism is in $I_\alpha$ and therefore all the $I_\beta$ are non-empty. It is also clear that if $f \in I_\beta$ and $g \subseteq f$ then $g \in I_\beta$. Thus, we only need to show property *(c)*.

For contradiction, suppose that there is an $f \in I_{\beta+1}$ with $|f| < k$ and an $a \in A$ such that for all $g \in I_\beta$ with $g \supseteq f$, $a \notin dom(g)$. Then, for every $b \in B$, there must be a formula $\psi_b[y_0 y_1 \ldots y_m]$ with $qr(\psi_b) \leq \beta$ such that $\mathcal{A} \models \psi_b[a a_1 \ldots a_m]$ and $\mathcal{B} \models \neg\psi_b[b f(a_1) \ldots f(a_m)]$ (where $a_1, \ldots, a_m \in dom(f)$). Let $\phi = \exists y \bigwedge_{b \in B} \psi_b[y y_1 \ldots y_m]$. But then, $qr(\phi) = \beta + 1$, $\mathcal{A} \models \phi[a_1 \ldots a_m]$ and $\mathcal{B} \models \neg\phi[f(a_1) \ldots f(a_m)]$ contradicting the assumption that $f \in I_{\beta+1}$. ■

Note that if in the above theorem, we take $\alpha$ to be a finite ordinal $n$ and we remove the restriction on the size of the function $f$ in clause *(c)*, we get a statement equivalent to Theorem 1.1. To establish this, we need to see that two structures agree on all sentences of $L_{\infty\omega}$ of quantifier rank less than $n$ if they agree on all first-order sentences of quantifier rank less than $n$. This can be shown by a simple induction argument on the structure of the infinitary formulas[3].

If the two structures $\mathcal{A}$ and $\mathcal{B}$ are finite, then any chain of sets of partial isomorphisms as above of length $\omega$ can be extended to any ordinal length. To see this, note that there are only finitely many maps from subsets of $A$ into $B$. Thus, one of the sets in the chain must be repeated, and hence, can be repeated indefinitely. This gives us the following corollary:

**Corollary 3.1** *For finite structures $\mathcal{A}$ and $\mathcal{B}$, the following are equivalent:*

- *For every sentence $\phi \in L^k_{\infty,\omega}$, $\mathcal{A} \models \phi$ iff $\mathcal{B} \models \phi$*
- *For every sentence $\phi \in L_k$, $\mathcal{A} \models \phi$ iff $\mathcal{B} \models \phi$*

We write $\mathcal{A} \equiv_k \mathcal{B}$ to denote that $\mathcal{A}$ and $\mathcal{B}$ satisfy the same sentences of $L_k$. When the sequence of sets of partial isomorphisms is finite, we can view it as a formalization of the $k$-pebble game introduced in Section 2.3.1. The above Corollary then provides a proof of Lemma 2.1.

[3]This works only in purely relational languages – it is not true when function symbols are present. This is because the use of function symbols involves a "hidden" increase in quantifier rank, as can be seen by the process of re-writing formulas with functions into equivalent relational formulas

## 3.3 Fragments of $L^{\omega}_{\infty\omega}$

The extensions of first-order logic with the least-fixed-point operation (FO + LFP) and with the partial-fixed-point operation (FO + PFP) (see Section 1.3.2) can be viewed as fragments of $L^{\omega}_{\infty\omega}$.

Consider any formula $\phi \equiv \mathbf{lfp}(S, x, \ldots, x_n)\psi(S)$, where $\psi$ is a first-order formula positive in $S$ and let $k$ be the number of distinct variables occurring in $\psi$. As we saw in the proof of Lemma 2.2, the $m^{\text{th}}$ iterative stage of $\phi$ can be represented by a first-order formula $\psi^m$ which has no more than $k$ distinct variables. Then, $\phi$ is equivalent to the formula $\bigvee_{m=0}^{\infty} \psi^m$ which is in $L^{\omega}_{\infty\omega}$. Similarly, $\mathbf{pfp}(S, x, \ldots, x_n)\psi(S)$ is equivalent to $\bigvee_{m=0}^{\infty}(\psi^m(x_1 \ldots x_n) \wedge \forall x_1 \ldots \forall x_n(\psi^m(x_1 \ldots x_n) \leftrightarrow \psi^{m+1}(x_1 \ldots x_n)))$.

$L^{\omega}_{\infty\omega}$ is also an extension of $\text{GM}^{loose}$ (loosely coupled generic machine), a model of generic computation introduced by Abiteboul and Vianu [AV91b]. Since every query computable by a $\text{GM}^{loose}$ is recursive, the containment is proper. On the other hand, both FO + LFP and FO + PFP are contained in $\text{GM}^{loose}$. This fact is used in [AV91b] to study the relationship of these two languages. Abiteboul and Vianu establish that FO + LFP = FO + PFP if and only if P = PSPACE. Since the two languages correspond exactly to P and PSPACE on ordered structures, if they coincide on all structures, the complexity classes collapse. The entailment in the other direction had, until recently been an open question. We present a proof of it in Section 4.3.3.

An even more remarkable result proved in [AV91b] is as follows. Define FO + PFP|P to be the sub-class of FO + PFP in which each iterative definition closes in a number of steps polynomial in the size of the structure. This class is actually contained within P and within $L^{\omega}_{\infty\omega}$ and contains FO + LFP. It was conjectured that in fact FO + LFP = FO + PFP|P = $L^{\omega}_{\infty\omega} \cap P$. Abiteboul and Vianu showed that the first equality in this is also equivalent to the problem P = PSPACE. We present a proof of this in Section 4.3.4, where we also prove that FO + LFP is properly contained in $L^{\omega}_{\infty\omega} \cap P$.

# Chapter 4

# Preliminary Results

In this chapter we look at the logics with a bounded number of variables and establish some preliminary results in order to study their expressive power. First we show that restricting first-order logic to a fixed number of variables does not restore the completeness theorem. Next, we determine every finite structure is characterized up to equivalence in this logic by a single sentence. This also gives us a way to define a type in this logic by a single formula. We show how these types can be inductively ordered and use this ordering to establish results about inductive logics.

## 4.1 Validities are not R.E.

We begin by showing that the sentence of $L_k$, for a fixed $k$ that is sufficiently large, valid over finite structures are not recursively enumerable. This is a strengthening of Trakhtenbrot's result obtained by a modification of the proof of that result as presented in [EFT84].

Let $M$ be a two-counter machine, *i.e.* $M$ is a machine with finite control, a read-only input tape and two storage tapes that serve as counters. The tape alphabet consists of a single symbol 1, so that all integers (including the input) are represented in unary. The only operations that are permitted on the storage tapes are writing a 1 at the end, and erasing a 1 from the end. Such machines are well known to be Turing equivalent, *i.e.* for any recursively enumerable set, there is a two-counter machine that accepts it. Indeed, an effective enumeration $M_1, M_2, \ldots$ of the two-counter machines can be established in such a way that the enumeration $\{C_i | i \in \mathbf{N}\}$ is an acceptable numbering of the recursively enumerable sets, where $C_i \subseteq \mathbf{N}$ is the set accepted by $M_i$. It follows from this that $K = \{i \in \mathbf{N} | i \in C_i\}$ is recursively enumerable and that $\bar{K} = \mathbf{N} - K$ is not recursively enumerable.

In order to establish that the set of finitely valid formulas of $L_k$ is not recursively enumerable,

we will show that $\bar{K}$ is $m$-reducible to it. First we show that for every two counter machine $M_i$ we can construct a formula $\phi_i(x)$ in $L_{11}$ such that, for any $n \in \mathbf{N}$, $n \in C_i$ if and only if $\phi_i(\overline{n})$ has a finite model. It follows that $\bar{K}$ is $m$-reducible to the finitely valid sentences of $L_{11}$, since $n \in \bar{K}$ iff $\neg\phi_n(\overline{n})$ is finitely valid.

A configuration of $M$ can be given by $qn_1n_2n_3n_4$, where $q$ is a state of $M$ and the $n_i$ are integers. It indicates that there are $n_1$ symbols on the input tape to the left of the read head, $n_2$ symbols on the input tape to the right of the read head and $n_3$ and $n_4$ symbols on the two counters, respectively.

Let $L$ be the language with a 6-ary relation $R$, a binary relation $<$ and a binary realtion $S$. Intuitively, a structure for $L$ would be an initial segment of the natural numbers encoding a computation by a two-counter machine with $<$ being the usual ordering and $S$ being the successor relation. $R$ encodes the configurations of the machine, so that $R(k, i, n_1, n_2, n_3, n_4)$ is true if and only if the machine is in configuration $q_in_1n_2n_3n_4$ after $k$ computation steps.

Let $\psi_0$ be a sentence that states that $<$ is a linear order, and that $S(x, y)$ if and only if $y$ is the succesor of $x$ in the ordering, unless $x$ is a maximal element, in which case $S(x, x)$. We also define the formulas $\sigma_i(x)$ inductively as follows:

$\sigma_0(x)$ is $\forall y(x < y \vee x = y)$,
$\sigma_{i+1}$ is $\exists z S(x, z) \wedge \sigma_i(z)$ if $i$ is even, *and*
$\sigma_{i+1}$ is $\exists y S(x, y) \wedge \sigma_i(y)$ if $i$ is odd.

Note that the number of variables in any $\sigma_i$ is at most three. In the following, we use the notation $\phi(\overline{n})$ to denote the formula $\exists x \sigma_n(x) \wedge \phi(x)$.

Let the states of $M$ be $q_1, \ldots, q_s$, with $q_s$ being the unique accepting state.

For each state $q_i$, other than the accepting state, there are two transitions defined, depending on whether the input symbol under the read head is a 1 or a blank ($B$). On each of the transitions, either of the counters may have a symbol appended to it, erased from it or it may be left alone. All of this can be encoded in a sentence $\psi_i$. For instance, if from a given state $q_i$, the transition on input 1 is to state $q_j$, moving the head to the left, adding a 1 to the first counter and erasing one from the second, while on input $B$, the head remains where it was, the machine goes into state $q_k$ and both counters are unchanged, then the corresponding $\psi_i$ is:

$$\begin{aligned}
\forall vwxyz \quad & (R(w, \overline{i}, x, v, y, z) \wedge \neg\sigma_0(v)) \rightarrow \\
& \exists v'w'x'y'z'(S(w, w') \wedge w \neq w' \wedge S(v, v') \wedge S(y, y') \wedge S(x', x) \wedge S(z', z) \\
& \quad \wedge R(w', \overline{j}, x', v', y', z')) \\
\wedge \quad & (R(w, \overline{i}, x, v, y, z) \wedge \sigma_0(v)) \rightarrow \\
& (\exists w' S(w, w') \wedge w \neq w' \wedge R(w', \overline{k}, x, v, y, z))
\end{aligned}$$

$\psi_s$ is the sentence $\forall vwxyz R(w, \overline{s}, x, v, y, z) \rightarrow R(w, \overline{s+1}, x, v, y, z)$

Finally, we define the sentence $\phi_i(x)$ as $\bigwedge_{0\leq j\leq s} \psi_j \wedge R(0,\overline{1},x,0,0,0)$. It is clear that a model of $\phi_j(\overline{n})$ encodes the computation of $M_j$ on input $n$, and in particular, $\phi_j(\overline{n})$ has a finite model if and only if $M_j$ halts on input $n$ and our construction is complete. We only need to note that as written above, each of the $\psi_i$ has at most 10 variables. The $\overline{i}$'s can be expanded with the introduction of one additional variable. Hence, each of the $\phi_j$ is in $L_{11}$.

## 4.2 Characterizing Structures up to $L_k$-equivalence

As we have seen earlier (Section 1.1), for every finite structure $\mathcal{A}$, we can write a first-order sentence $\phi_{\mathcal{A}}$ such that any structure that satisfies $\phi_{\mathcal{A}}$ is isomorphic to $\mathcal{A}$. However, obviously, not all such sentences are in $L_k$ for any given $k$. This raises the question of whether there is a sentence $\phi^k_{\mathcal{A}}$ of $L_k$ associated with $\mathcal{A}$ such that any structure satisfying it is $L_k$-equivalent to $\mathcal{A}$. In this section, we answer this question in the affirmative. The proof is adapted from the proof of Scott's theorem in [Bar73]. For the purpose of this section, we will assume that there are no constants in the language being considered. The results can be easily generalised to the case where constants are present.

Let $A$ be the universe of $\mathcal{A}$ and let $S = A^{\leq k}$ be the set of sequences of element of $A$ of length less than or equal to $k$. For $s \in S$ and $a \in A$, let $s \cdot \langle a \rangle$ denote the sequence obtained by extending $s$ by the single element $a$.

We define a formula $\phi^m_s$ for each $s \in S$ and each $m \in \mathbf{N}$. The formula has free variables $x_1, \ldots, x_l$, where $l$ is the length of $s$. These formula are defined by induction as follows:

for all $s = \langle a_1 \ldots a_l \rangle$,

$\phi^0_s(x_1 \ldots x_l)$ is the conjunction of all atomic and negated atomic formulas $\theta(x_1 \ldots x_l)$ such that $\mathcal{A} \models \theta[a_1 \ldots a_l]$

if length$(s) < k$ then,

$$\phi^{m+1}_s(x_1 \ldots x_l) = \phi^m_s(x_1 \ldots x_l) \wedge \tag{4.1}$$

$$\bigwedge_{a\in A} \exists x_{l+1} \phi^m_{s\cdot\langle a\rangle}(x_1 \ldots x_{l+1}) \wedge \tag{4.2}$$

$$\forall x_{l+1} \bigvee_{a\in A} \phi^m_{s\cdot\langle a\rangle}(x_1 \ldots x_{l+1}) \tag{4.3}$$

if length$(s) = k$ then,

$$\phi^{m+1}_s(x_1 \ldots x_l) = \bigwedge_{i=1\ldots k} \phi^{m+1}_{s_i}(x_1 \ldots x_{i-1} x_{i+1} \ldots x_l) \tag{4.4}$$

where $s_i$ is the sequence obtained from $s$ by deleting the $i^{\text{th}}$ element

**Lemma 4.1** *Let $s = \langle a_1 \dots a_l \rangle \in S$ be a sequence of elements from $\mathcal{A}$ with $l \leq k$. For any finite structure $\mathcal{B} = (B, \dots)$ and $b_1, \dots, b_l \in B$, $\mathcal{B} \models \phi_s^m[b_1 \dots b_l]$ if and only if there is a sequence of sets of partial isomorphisms $I_0 \supseteq \dots \supseteq I_m$ with the $k$ back and forth property and $f = \{\langle a_1, b_1 \rangle \dots \langle a_l, b_l \rangle\} \in I_m$*

**Proof:**

$\Leftarrow$ This follows immediately from the proof of Theorem 3.1 since the existence of such a sequence implies that for any $\phi$ of quantifier rank $m$, $\mathcal{B} \models \phi[b_1 \dots b_l]$ if $\mathcal{A} \models \phi[a_1 \dots a_l]$. Clearly, $qr(\phi_s^m) = m$ and $\mathcal{A} \models \phi_s^m[s]$.

$\Rightarrow$ The proof is by induction on $m$.

**Basis** Let $I_0 = \{g | g \subseteq f\}$. Even if $s = \langle\rangle$ and $f$ is the empty map, $I_0$ is non-empty.

**Induction Step** There are two cases to be considered:

**Case:** $l < k$

Let $I_{m+1} = \{g | g \subseteq f\}$.

By induction hypothesis and 4.1, there is a sequence $I_0^s \dots I_m^s$ with the $k$ back and forth property and $f \in I_m^s$.

Furthermore, by 4.2 and the induction hypothesis, for every $a \in A$, there is a $b \in B$ and a sequence $I_0^{s \cdot \langle a \rangle} \dots I_m^{s \cdot \langle a \rangle}$ with the $k$ back and forth property such that $\{\langle a_1, b_1 \rangle \dots \langle a_l, b_l \rangle, \langle a, b \rangle\} \in I_m^{s \cdot \langle a \rangle}$.

Let $I_j = I_j^s \cup \bigcup_{a \in A} I_j^{s \cdot \langle a \rangle}$ (for $0 \leq j \leq m$). Note that, in general, the $k$ back and forth property *is* preserved under this kind of element-wise union. Thus, we need to verify that $I_{m+1} \subseteq I_m$ – this follows from the fact that $I_{m+1} \subseteq I_m^s$ – and that every element of $I_{m+1}$ is extensible in $I_m$ to arbitrary elements of $A$ and $B$. This follows from 4.2 and 4.3 respectively.

**Case:** $l = k$

By the argument for the case above, there are sequences $I_0^i \dots I_{m+1}^i$ corresponding to each of the partial isomorphisms, $f_i$, obtained by dropping the pair $\langle a_i, b_i \rangle$ from $f$.

Let $I_j = \{f\} \cup \bigcup_{i=1 \dots k} I_j^i$ for $0 \leq j \leq m+1$. Each of the $I_j$ is still closed under restrictions, because if $g \subseteq f$, then either $g = f$ or $g \subseteq f_i$ for some $i$. Since $|f| = k$, extensibility of $f$ is not required, and we are done. ■

For a given sequence $s$ of length $l$, let $X_s^m = \{s' \in S | \mathcal{A} \models \phi_s^m[s']\}$. Each $X_s^m$ is a set of $l$-tuples of $A$ and $X_s^m \supseteq X_s^{m+1}$. Since $A$ is finite, there must be an $m_s$ such that $X_s^{m_s} = X_s^m$ for all $m > m_s$. Let $m^\star = \max(m_s | s \in S)$. Now, define the sentence $\phi$ as follows:

$$\phi \equiv \phi^{m^\star}_{\langle\rangle} \wedge \bigwedge_{s \in S} \forall x_1 \ldots \forall x_k (\phi^{m^\star}_s \rightarrow \phi^{m^\star+1}_s)$$

Note that $\phi \in L_k$ and that $\mathcal{A} \models \phi$ We now show that this sentence characterizes the structure $\mathcal{A}$ up to $L_k$ equivalence.

**Theorem 4.1** *For every finite structure $\mathcal{A}$ and any $k$, there is a sentence, $\phi$ of $L_k$ such that $\mathcal{A} \models \phi$ and for any structure $\mathcal{B}$, $\mathcal{B} \models \phi$ if and only if $\mathcal{A} \equiv_k \mathcal{B}$.*

**Proof:**

Let $\phi$ be as defined above. We only need to show that if $\mathcal{B} \models \phi$, then $\mathcal{A} \equiv_k \mathcal{B}$. Let $F$ be the set of maps $\{\langle a_1, b_1\rangle, \ldots, \langle a_l, b_l\rangle\}$ such that $\mathcal{B} \models \phi^{m^\star+1}_{\langle a_1 \ldots a_l\rangle}[b_1 \ldots b_l]$. The set $F$ is non-empty since $\mathcal{B} \models \phi^{m^\star+1}_{\langle\rangle}$. By Lemma 4.1, for each $f \in F$, there is a sequence $I^f_0 \ldots I^f_{m^\star+1}$ with the $k$ back and forth property. Let $I_i = \bigcup_{f \in F} I^f_i$ and let $I_m = I_{m^\star+1}$ for all $m > m^\star + 1$. We claim the infinite sequence $I_0 \supseteq \ldots \supseteq I_m \ldots$ has the $k$ back and forth property. We will establish the extensibility of every element of $I_{m^\star+2}$. The rest then follows.

Consider any $f \in I_{m^\star+2}$ with $|f| < k$ and any $a \in A$. By definition, $f \in I_{m^\star+1}$. Since we know that the sequence through $I_{m^\star+1}$ has the $k$ back and forth property, there is a $g \in I_{m^\star}$ such that $f \subseteq g$ and $a \in dom(g)$. But then, by the other direction of Lemma 4.1 $\mathcal{B} \models \phi^{m^\star}_{\langle dom(g)\rangle}[\langle rng(g)\rangle]$ and therefore, by the implication in $\phi$, $g \in I_{m^\star+1}$ and we are done. ■

There are some points about the above construction that are noteworthy. First of all, we could have, alternatively, defined $m^\star$ as the smallest $m$ such that $X^m_s = X^{m+1}_s$ for all $s$. To see this, just observe that this is the only property of $m^\star$ used in the above proof. Given that $k$ is the maximum length of any sequence in $s$, and that there are $n^k$ $k$-tuples in a structure of size $n$, we can derive the bound $m^\star \leq n^k$. This gives us the following:

**Corollary 4.1** *If $\mathcal{A}$ is a structure of size $n$ and $\mathcal{B}$ a structure such that $\mathcal{A}$ and $\mathcal{B}$ agree on all sentences of $L_k$ of quantifier rank up to $n^k + k + 1$, then $\mathcal{A} \equiv_k \mathcal{B}$.*

The following corollary is also immediate:

**Corollary 4.2** *If $K$ is a query closed under $L_k$ equivalence (that is, if $\mathcal{A} \in K$ and $\mathcal{A} \equiv_k \mathcal{B}$ then $\mathcal{B} \in K$), then $K$ is definable in $L^k_{\infty\omega}$.*

**Proof:**

If we write $\phi_{\mathcal{A}}$ for the sentence of $L_k$ that chracterizes a structure $\mathcal{A}$ up to $L_k$ equivalence, then $K$ is defined by the sentence $\bigvee\{\phi_{\mathcal{A}} | \mathcal{A} \in K\}$. ■

Finally, it is not only structures that are characterised up to $L_k$ equivalence in the above proof. For any sequence $s = \langle a_1 \ldots a_l\rangle$ of elements in a structure $\mathcal{A}$, define the *$L_k$-type* of $s$ to be the set of formulas, $\phi$, in $l$ free variables, such that $\mathcal{A} \models \phi[a_1 \ldots a_l]$. Then, we get the following:

**Corollary 4.3** *For every structure $\mathcal{A}$ and sequence $a_1, \ldots, a_l (l \leq k)$ of elements from $\mathcal{A}$, there is a formula, $\phi$, in $L_k$ with $l$ free variables such that if $\mathcal{B} \models \phi[b_1 \ldots b_l]$ in any structure $\mathcal{B}$, then $a_1 \ldots a_l$ and $b_1 \ldots b_l$ have the same $L_k$-type.*

## 4.3 Ordering the Types

Having seen how, for a particular $L_k$ type, we can write a formula that characterizes it, we now turn to writing a formula that will define a total ordering of these types. We will show that this can be done uniformly in FO + LFP, *i.e.* a single formula will define the ordering on all structures. From this result we will derive Abiteboul and Vianu's result [AV91b] that PSPACE = P if and only if FO + LFP = FO + PFP.

### 4.3.1 Inductive definition of the ordering

Looking again at the definitions of the $\phi_s^m$ in the last section, we can see that these formulas were defined by a simulatneous induction on first-order formulas – simultaneous in all the sequences $s$. This, along with the observation that the basis of this induction is finite, in the sense that there are, up to equivalence, only finitely many quantifier free formulas in a finite relational language, suggests that we could accomplish the entire process with a single formula of FO + LFP. We formalize this intuition below.

We first construct a formula of FO + LFP which defines, on any structure $\mathcal{A}$, an equivalence relation on $k$-tuples of elements such that two tuples are equivalent if and only if they have the same $L_k$-type. In the following, for ease of reading, we will use the notation $x_1 \ldots x \ldots x_k$ to indicate a sequence of variables in which $x$ has been substituted for $x_i$, when the particular $i$ is clear from the context.

Let $\alpha_1(x_1 \ldots x_k), \ldots, \alpha_q(x_1 \ldots x_k)$ be an enumeration, up to equivalence, of all quantifier free formulas of $L_k$ with $k$ free variables on the finite signature $\sigma$. Then, define $\phi_0$ as follows:

$$\phi_0(x_1 \ldots x_k y_1 \ldots y_k) \equiv \bigvee_{1 \leq i \neq j \leq q} (\alpha_i(\bar{x}) \wedge \alpha_j(\bar{y}))$$

where $\alpha_i(\bar{y})$ is obtained from $\alpha_i(\bar{x})$ by replacing every $x_j$ by $y_j$.

Now, define $\phi$ and $\psi$ as follows:

$$\phi(R, x_1 \ldots x_k y_1 \ldots y_k) \equiv \phi_0(\bar{x}\bar{y}) \vee \bigvee_{1 \leq i \leq k} \exists x \forall y R(x_1 \ldots x \ldots x_k y_1 \ldots y \ldots y_k) \vee \bigvee_{1 \leq i \leq k} \exists y \forall x R(x_1 \ldots x \ldots x_k y_1 \ldots y \ldots y_k)$$

$$\psi(z_1 \ldots z_{2k}) \equiv \neg \mathbf{lfp}(R, \bar{x}, \bar{y}) \phi(z_1 \ldots z_{2k})$$

**Claim 4.1** *For any structure $\mathcal{A}$ on signature $\sigma$, $\mathcal{A} \models \psi[a_1 \ldots a_k a'_1 \ldots a'_k]$ if and only if $\bar{a}$ and $\bar{a'}$ have the same $L_k$-type.*

**Proof:**

To establish this claim, we need to show that $\mathbf{lfp}(R, \bar{x}, \bar{y})\phi[\bar{a}\bar{a'}]$ expresses the *inequivalence* of the two $k$-tuples. Picture the $k$-pebble game being played on two isomorphic copies of $\mathcal{A}$, and at some stage the pebbles are placed on $\langle a_1, a'_1\rangle, \ldots \langle a_k, a'_k\rangle$. By Lemma 4.1, if the two tuples have the same $L_k$-type, then Player II can play indefinitely from this point on without losing. We claim that if $\Phi^r[\bar{a}\bar{a'}]$, then Player I can win in $r$ moves or less. Clearly, if $r = 0$, by the definition of $\phi_0$, $\bar{a}$ and $\bar{a'}$ differ on a quantifier free formula and hence the map from one to the other is not a partial isomorphism. If $r = m + 1$, then the definition of $\phi$ tells us that we can, in one move, get to a configuration that is in $\Phi^m$. ■

We will henceforth use the symbol $\sim_k$ in infix notation to denote the relation defined by $\psi$. We will know give an inductive definition of an ordering relation on the equivalence classes defined by this relation. In other words, we will define a $2k$-ary relation that is a pre-order on $k$-tuples such that two tuples are not ordered by this pre-order just in case they have the same $L_k$-type.

Define the following formulas for each $1 \leq i \leq k$:

$$\beta_i \equiv \forall x \exists y((x_1 \ldots x \ldots x_k) \sim_k (y_1 \ldots y \ldots y_k))$$

$$\delta_i \equiv \bigwedge_{j<i} \beta_j \wedge \neg\beta_i$$

Also, let

$$\phi_0(x_1 \ldots x_k y_1 \ldots y_k) \equiv \bigvee_{1 \leq i < j \leq q} (\alpha_i(\bar{x}) \wedge \alpha_j(\bar{y}))$$

We now define

$$\phi(\bar{x}\bar{y}) \equiv \phi_0(\bar{x}\bar{y}) \vee \bigvee_{1 \leq i \leq k} (\delta_i \wedge \exists x \forall y(\exists z( \quad (x_1 \ldots z \ldots x_k) \not\sim_k (y_1 \ldots y \ldots y_k) \rightarrow R(x_1 \ldots x \ldots x_k y_1 \ldots y \ldots y_k))))$$

$$\psi(z_1 \ldots z_{2k}) \equiv \mathbf{lfp}(R, \bar{x}\bar{y})\phi(z_1 \ldots z_{2k})$$

To interpret the above formula, define $move_i(\langle a_1 \ldots a_k\rangle) = \{\langle a_1 \ldots a \ldots a_k\rangle | a \in A\}$, *i.e.* the set of tuples obtainable from $\bar{a}$ by replacing $a_i$. Then, what $\phi$ says is that tuples $\bar{a}$ and $\bar{a'}$ are ordered ($\bar{a} < \bar{a'}$) if, for the smallest $i$ such that $move_i(\bar{a})$ is different from $move_i(\bar{a'})$, the smallest element in their set difference is in $move_i(\bar{a})$.

$\phi_0$ defines a pre-order on tuples, distinguishing them based on their quantifier free $L_k$-type. The inductive stages then refine this pre-order according to the rule given above. This gives us the following:

**Claim 4.2**

1. *On any structure, $\mathcal{A}$, $\psi$ defines a pre-order on $k$-tuples. We will write $\bar{x} <_k \bar{y}$ for $\psi(\bar{x}\bar{y})$.*

2. *If $\bar{a}$ and $\bar{a'}$ have the same $L_k$-type, then neither $\bar{a} <_k \bar{a'}$ nor $\bar{a} <_k \bar{a'}$.*

Using the formulas just defined, it is fairly easy to define the $L_k$ equivalence and the corresponding pre-order relation on tuples shorter than $k$.

### 4.3.2 Rigid Structures

Consider the pre-order $<_k^1$ on single elements, *i.e.* tuples of length 1. Clearly, if there is at most one element of any $L_k$-type in a structure, then $<_k^1$ defines a total ordering on the universe of the structure. Since this ordering is definable in FO + LFP, and since FO + LFP expresses all of P in the presence of ordering, this implies that FO + LFP expresses all of P on these structures. We formalize this below:

**Definition 4.1** *A structure $\mathcal{A}$ is called* rigid *if the only automorphism on $\mathcal{A}$ is the identity.*

**Definition 4.2** *Call a structure $\mathcal{A}$ $k$-rigid if no two elements of $\mathcal{A}$ have the same $L_k$-type.*

Clearly, every $k$-rigid structure is rigid. Conversely,

**Proposition 4.1** *Every rigid structure $\mathcal{A}$ is $k$-rigid for some $k$.*

**Proof:**
For contradiction, assume that $\mathcal{A}$ is a rigid structure that is not $k$-rigid for any $k$. Then there are distinct elements $a_1, a_2$ in $\mathcal{A}$ such that they have the same $L_k$-type for all $k$. This is because two elements that share an $L_k$-type share their $L_l$-type for all $l < k$. But then, $a_1$ and $a_2$ have the same first-order type. Now, expand the vocabulary by a constant symbol $c$, and consider the expanded structures $\langle \mathcal{A}, a_1 \rangle$ and $\langle \mathcal{A}, a_2 \rangle$. These structures are elementarily equivalent. To see this, note that any first-order sentence $\phi$ not involving the constant $c$ is true in one of the structures just in case it is true in $\mathcal{A}$. On the other hand, if $\phi$ involves $c$ and is true in $\langle \mathcal{A}, a_1 \rangle$, then $\phi(x)$, obtained by replacing all occurrences of $c$ by the new variable $x$ is in the first-order type of $a_1$ in $\mathcal{A}$ and hence true at $a_2$ as well. Because any two finite structures that are elementarily equivalent are isomorphic, $\mathcal{A}$ is not rigid and we have a contradiction. ■

The argument we gave above on the expressiveness of FO + LFP can now be formally stated as:

**Proposition 4.2** *Let $K$ be a query computable in polynomial time such that there is a $k$ such that every structure in $K$ is $k$-rigid. Then $K$ is expressible by a sentence of FO + LFP.*

Observe that any structure with a linear ordering, $<$, is 2-rigid. There is a formula $\alpha_i(x) \in L_2$ which defines the $i^{\text{th}}$ element in the ordering uniquely. For instance,

$$\alpha_3(x) \equiv \exists y(y < x \wedge \exists x(x < y \wedge \forall y(\neg y < x)))$$

We can interpret the above proposition as telling us the crucial property of the ordering that allows us to express all of P in FO + LFP.

### 4.3.3 Translation to an Ordered Structure

In general, on a structure, $\mathcal{A}$, that is not rigid, $<_k$ defines a pre-order, or alternatively a total ordering on the $L_k$ equivalence classes. We can look at this as the basis for a translation of the structure $\mathcal{A}$ onto a totally ordered structure in which each of the equivalence classes is collapsed to a point. This translation is interesting from the following point of view – consider any formula $\phi \equiv \mathbf{lfp}(R, \bar{x})\psi$ (or $\mathbf{pfp}(R, \bar{x})\psi$) with only $k$ variables. Then, not only is the relation defined by $\phi$ on $\mathcal{A}$ closed under $L_k$ equivalence, but so is every iterative stage of $\phi$. This raises the possibility that we can describe $\phi$ as an induction on the $L_k$ equivalence classes of tuples.

More formally, for any structure $\mathcal{A} = \langle A, R_1, \ldots, R_m \rangle$, let

$$E_k(\mathcal{A}) = \langle A^{\leq k} / \sim_k, <_k, =', R'_1, \ldots, R'_m, U_1, \ldots, U_k, X_i, P^i_\pi \rangle$$

be the structure defined as follows:

- The universe of $E_k(\mathcal{A})$ is $A^{\leq k} / \sim_k$, *i.e.* the equivalence classes of tuples from $A$ of length $\leq k$ under the equivalence relation $\sim_k$. We will write $[\bar{a}]$ to denote the equivalence class that includes $\bar{a}$.

- The relations $U_1 \ldots U_k$ are unary relations such that $U_i([\bar{a}])$ holds if and only if $\bar{a}$ is of length $i$. Note that these relations are well-defined because tuples of different lengths cannot be equivalent.

- $<_k$ is a total ordering on the universe of $E_k(\mathcal{A})$ defined from the relations $<^i_k$ on tuples of length $i$ as follows: $[\bar{a}] <_k [\bar{a}']$ if and only if $\mathit{length}(\bar{a}) < \mathit{length}(\bar{a}')$ or if $\mathit{length}(\bar{a}) = \mathit{length}(\bar{a}') = i$ and $[\bar{a}] <^i_k [\bar{a}']$.

- $='$ is a unary relation such that $='([\bar{a}])$ holds if and only if $\bar{a} = \langle a, a \rangle$ for some $a \in A$.

- For each relation $R_i$ in $\mathcal{A}$ we have a relation $R'_i$ in $E_k(\mathcal{A})$ such that $R'_i([\bar{a}])$ holds if and only if $\mathcal{A} \models R_i[\bar{a}]$.[1]

[1] As defined, this works only if the arity, $m$, of $R_i$ is at most $k$. If this is not the case and $m > k$, we first replace $R_i$ by a collection of relations of arity $k$ by taking all the ways that we can form an $m$-tuple from at most $k$ elements. This does not affect the results, since we are only considering formulas with at most $k$ variables

- $X_i$ – the extension relation for tuples of length $i$– is a binary relation such that $X_i([\bar{a}],[\bar{a}'])$ holds if and only if $\mathit{length}(\bar{a}) = i$ and there is an $a \in A$ such that $\bar{a}' = \langle \bar{a}, a \rangle$. This relation is well-defined, because two tuples that differ on the types to which they can be extended cannot be of the same type.

- $P^i_\pi$ is a binary relation for every $i \leq i \leq k$ and every permutation, $\pi$, on $i$ elements. $P^i_\pi([\bar{a}],[\bar{a}'])$ holds if and only if $\pi(\bar{a}) = \bar{a}'$. This relation is well-defined since, if $\phi$ is a formula in the $L_k$-type of $\bar{a}$, then so is any formula obtained by permuting the free variables of $\phi$. Hence, if $\bar{a}_1$ and $\bar{a}_2$ have the same $L_k$-type, then so do $\pi(\bar{a}_1)$ and $\pi(\bar{a}_2)$

We will also write $E_k(\sigma)$ to denote the signature of $E_k(\mathcal{A})$, when $\sigma$ is the signature of $\mathcal{A}$.

**Lemma 4.2** *The translation $E_k$ is computable on all structures $\mathcal{A}$ in time polynomial in the size of the structure $\mathcal{A}$.*

**Proof:**

The number of tuples in $A^{\leq k}$ is $O(n^k)$ where $n = |A|$. The equivalence relation $\sim_k$ is defined by an FO + LFP formula and hence computable in polynomial time as is the ordering $<_k$. We can get, therefore, in polynomial time, a representation of the universe of $E_k(\mathcal{A})$. All the other relations are easily defined on $\mathcal{A}$ (in FO). ■

Let $\mathcal{R}$ be a unary relation on structures over $E_k(\sigma)$. We say that $\mathcal{R}$ respects length if whenever $\mathcal{R}([\bar{a}])$ and $\mathcal{R}[\bar{a}']$ then $\mathit{length}(\bar{a}) = \mathit{length}(\bar{a}')$.

**Lemma 4.3** *For every first-order formula $\phi$ with at most one free variable in the language $E_k(\sigma)$, if the relation defined by $\phi$ always respects length, then there is an FO + LFP formula $\phi'$ in the language $\sigma$ such that for any structure $\mathcal{A}$, $\mathcal{A} \models \phi'[\bar{a}]$ if and only if $E_k(\mathcal{A}) \models \phi[[\bar{a}]]$.*

**Proof sketch:**

We have already seen above that all the relations on $E_k(\mathcal{A})$, including equality, are definable in FO + LFP on $\mathcal{A}$. Moreover, these definitions are uniform, *i.e.* for each $R \in E_k(\sigma)$ there is a single FO + LFP formula defining it for all $\mathcal{A}$. So, we obtain $\phi'$ by substituting these definitions in $\phi$. ■

Let be $\phi(R)$ a first-order formula in the language $E_k(\sigma) \cup \{R\}$ where $R$ is a unary relation symbol. Suppose $\phi$ has the property that the relation it defines on a structure $\langle E_k(\mathcal{A}), \mathcal{R} \rangle$ respects length if $\mathcal{R}$ does. Let $R^{\mathcal{A}} = \{\bar{a} | \mathcal{R}([\bar{a}])\}$ and let $\mathcal{A}'$ be the structure $\langle \mathcal{A}, R^{\mathcal{A}} \rangle$ interpreting the language $\sigma \cup R'$. Clearly, $E_k(\mathcal{A}') = \langle E_k(\mathcal{A}), \mathcal{R} \rangle$. Therefore, by Lemma 4.3, there is an FO + LFP formula in $\sigma \cup R'$ such that $\mathcal{A}' \models \phi'[\bar{a}]$ if and only if $\langle E_k(\mathcal{A}), \mathcal{R} \rangle \models \phi[[\bar{a}]]$. This gives us the following result:

**Lemma 4.4** *For every FO + LFP (respectively FO + PFP) formula $\phi$ in the language $E_k(\sigma)$ that respects length, there is an FO + LFP (respectively FO + PFP) formula $\phi'$ in the language $\sigma$ such that for any structure $\mathcal{A}$, $\mathcal{A} \models \phi'[\bar{a}]$ if and only if $E_k(\mathcal{A}) \models \phi[[\bar{a}]]$.*

**Proof:**

We will only prove the case of FO + LFP, the other case being analogous.

Let $\phi \equiv \mathbf{lfp}(R, s)\psi(s)$ be a formula in the language $E_k(\sigma)$. We know by the above argument that there is a formula $\psi'(R')$ of FO + LFP in $\sigma \cup R'$ such that $\mathcal{A}' \models \psi'[\bar{a}]$ if and only if $\langle E_k(\mathcal{A}), \mathcal{R}\rangle \models \psi[[\bar{a}]]$, provided that $\mathcal{R}$ respects length. Since at every iteration of $\phi$, the relation defined respects length, let $\phi' \equiv \mathbf{lfp}(R', \bar{x})\psi'(\bar{x})$. ■

Note that every relation that is defined by a formula obtained in this way by translating back from the language $E_k(\sigma)$ is closed under the $L_k$ equivalence relation.

We now establish a translation of formulas in the other direction. Let $\phi$ be a formula of $L_k$ in the language $\sigma$. We will define, by induction on the structure of $\phi$ a first-order formula $\phi^\star$ in the language $E_k(\sigma)$. First, we define some notation. In what follows, we will write $\bar{x}$ for a sequence of variables, and $ln(\bar{x})$ for its length. In the translation we define, every sub-formula of $\phi$ with free variables $\bar{x}$ is translated into a sub-formula of $\phi^\star$ with exactly one free variable. For clarity, we will write $x^\star$ for this variable. We have, in the language $E_k(\sigma)$, the relation symbols $X_i$ which "extend" tuples of length $i$ by one element. From these, we can define relations $X_i^{i+j}$ which extend a tuple of length $i$ by $j$ elements. For instance, for $j = 2$:

$$X_i^{i+2}(x, y) \equiv \exists z(X_i(x, z) \wedge X_{i+1}(z, y))$$

The translation is defined as follows:

- If $\phi \equiv x_1 = x_2$, then $\phi^\star \equiv =_k(x^\star)$
- If $\phi \equiv R_j(\bar{x})$ and all the variables in $\bar{x}$ are distinct, then $\phi^\star \equiv R'_j(x^\star)$.

  If the variables are not all distinct, let $i = ln(\bar{x})$. For every pair $p = \langle p_1, p_2\rangle$ of positions in $\bar{x}$ that are occupied by the same variable, let $\pi_p$ be a permutation on $i$ elements that maps $\langle p_1, p_2\rangle \rightarrow \langle 1, 2\rangle$ and let $\theta_p \equiv \exists s_1 \exists s_2 X_2^i(s_1, s_2) \wedge P^i_{\pi_p}(x^\star, s_2) \wedge =_k(s_1)$. Then,

  $$\phi^\star \equiv R'(x^\star) \wedge \bigwedge_p \theta_p$$

- If $\phi \equiv \neg\psi(\bar{x})$ and $ln(\bar{x}) = i$, then

  $$\phi^\star \equiv U_i(x^\star) \wedge \neg\psi^\star(x^\star)$$

- Suppose $\phi(\bar{x}) \equiv \psi_1(\bar{x}_1) \wedge \psi_2(\bar{x}_2)$. Here, $\bar{x}$ is a sequence comprising all the variables in $\bar{x}_1$ and $\bar{x}_2$. Let $\bar{z}_1$ be a sequence extending $\bar{x}_1$ and $\pi_1$ a permutation mapping $\bar{z}_1$ to $\bar{x}$.

Similarly, let $\pi_2$ map an extension $\bar{z}_2$ of $\bar{x}_2$ to $\bar{x}$. Finally, let $ln(\bar{x}) = i$, $ln(\bar{x}_1) = i_1$ and $ln(\bar{x}_2) = i_2$. Then,

$$\phi^\star(x^\star) \equiv \exists x_1^\star \exists x_2^\star \exists z_1^\star \exists z_2^\star ( \; X^i_{i_1}(x_1^\star, z_1^\star) \wedge X^i_{i_2}(x_2^\star, z_2^\star) \wedge P^i_{\pi_1}(z_1^\star, x^\star) \wedge P^i_{\pi_2}(z_2^\star, x^\star) \\ \wedge \psi_1^\star(x_1^\star) \wedge \psi_2^\star(x_2^\star))$$

- If $\phi(\bar{x}) \equiv \exists z \psi(\bar{y})$ where $\bar{y} = \bar{x} \cdot z$ and $ln(\bar{x}) = i$ then:

$$\phi^\star(x^\star) \equiv \exists (y^\star X_i(x^\star, y^\star) \wedge \psi^\star(y^\star))$$

It should be clear from the construction of $\phi^\star$ that $\mathcal{A} \models \phi[\bar{a}]$ just in case $E_k(\mathcal{A}) \models \phi^\star[[\bar{a}]]$. Moreover, this is true even if $\phi$ is in an expanded language $\sigma \cup \{R\}$ as long as the interpretation of $R$ on $\mathcal{A}$ is closed under $L_k$ equivalence. This gives us the following result:

**Lemma 4.5** *For every FO + LFP (respectively FO + PFP) formula $\phi$ in the language $\sigma$, there is an FO + LFP (respectively FO + PFP) formula $\phi^\star$ in the language $E_k(\sigma)$ such that for any structure $\mathcal{A}$, $\mathcal{A} \models \phi[\bar{a}]$ if and only if $E_k(\mathcal{A}) \models \phi^\star[[\bar{a}]]$.*

We are now in a position to prove the following result from [AV91b]:

**Theorem 4.2** *FO + LFP = FO + PFP if and only if P = PSPACE.*

**Proof:**

$\Rightarrow$ This follows immediately from the fact that FO + LFP = P and FO + PFP = PSPACE on ordered structures. (Theorems 1.3 and 1.5 respectively).

$\Leftarrow$ Suppose P = PSPACE. Let $\phi$ be a sentence in FO + PFP over signature $\sigma$ and let the number of distinct variables in $\phi$ be $k$. Take $\phi^\star$ the corresponding sentence of FO + PFP in the language $E_k(\sigma)$. Since $\phi^\star$ is in FO + PFP, it is computable in PSPACE and hence in P. Since $E_k(\sigma)$ is a language with a total ordering, there is a sentence of FO + LFP, $\psi$ equivalent to $\phi^\star$. Moreover, because $\phi^\star$ is the translation of a relation defined over $\sigma$, it respects length and therefore so does $\psi$. Then, by Lemma 4.4 there is a $\psi'$ in FO + LFP over $\sigma$ that is equivalent to $\phi$ ■

### 4.3.4 Complete Binary Trees

As was stated in the proof of Lemma 4.2, the size of the structures $E_k(\mathcal{A})$ is bounded by a polynomial over all structures $\mathcal{A}$. Over some classes of structures, it can be considerably smaller. For instance, if we consider all structures over the language of identity, we find that all tuples of the same length are $L_k$ equivalent for any given $k$. It follows that there is a bound dependent

only on $k$ on the size of the structures $E_k(\mathcal{A})$. Another class of structures for which the size of $E_k(\mathcal{A})$ is much smaller than that of $\mathcal{A}$ is the class of complete binary trees. This yields some interesting results concerning logical expressibility.

Complete binary trees are graphs, *i.e.* structures $\langle V, E\rangle$ with one binary relation $E$ satisfying the following axioms:

1. $\forall x(\forall y(\neg Exy) \vee \exists y\exists z(y \neq z \wedge Exy \wedge Exz \wedge \forall w(Exw \rightarrow w = y \vee w = z)))$

   This says that every vertex has exactly 0 or 2 children.

2. $\forall x(\forall y(\neg Eyx) \vee \exists y(Eyx \wedge \forall z(Ezx \rightarrow z = y)))$

   This says that every vertex has exactly 0 or 1 parent.

3. $\exists x(\forall y(\neg Eyx) \wedge \forall z(\forall y(\neg Eyx) \rightarrow x = z))$

   This says that there is exactly one vertex (the root) that has no parent.

4. $\forall x\forall y\mathbf{lfp}(R, x, y)(x = y \vee \exists z(Rxz \wedge Ezy) \vee \exists z(Ryz \wedge Ezx))(x, y)$

   This says that the graph is connected, *i.e.* every pair of vertices is in the reflexive, transitive and symmetric closure of the edge relation.

5. $\forall x\neg(\mathbf{lfp}(R, x, y)(Exy \vee \exists z(Rxz \wedge Ezy))(x, x))$

   This says that there are no cycles.

6. $\forall x\forall y((\forall z(\neg Exz) \wedge \forall z(\neg Eyz)) \rightarrow \delta(x, y))$

   where,

   $\delta \equiv \mathbf{lfp}(R, x, y)((\forall z(\neg Ezx) \wedge \forall z(\neg Ezy)) \vee \exists w\exists z(Rwz \wedge Ewx \wedge Ezy))(x, y))$

   This says that all leaves are at the same distance from the root ($\psi$ defines an equivalence relation that relates vertices at the same depth).

If we let $CBT = \{G = \langle V, E\rangle | G \text{ is a binary tree}\}$, then by the above definition $CBT \in$ FO + LFP. Moreover, since we used only four distinct variables, $CBT \in L^4_{\infty\omega}$.

Define the formulas $\alpha_n$ recursively as follows:

$$\begin{aligned} \alpha_0(x) &\equiv \forall y\neg Eyx \\ \alpha_{n+1}(x) &\equiv \exists yEyx \wedge \exists x(x = y \wedge \alpha_n(x)) \end{aligned}$$

Then, for $T \in CBT$, $T \models \alpha_d[v]$ just in case $v$ is a vertex of depth $d$ in $T$. So, if $T_d$ is a complete binary tree, it has depth $d$ if and only if $T_d \models \exists x(\alpha_d) \wedge \neg\exists x(\alpha_{d+1})$. Note that each $\alpha_n$ contains only two distinct variables. Since any two complete binary trees of the same depth are isomorphic, we can conclude the following:

**Proposition 4.3** *If $T_1$ and $T_2$ are two complete binary trees such that $T_1 \equiv_2 T_2$, then $T_1 \cong T_2$.*

Combining this with the axiomatization above, we get the following result:

**Proposition 4.4** *If $q$ is any query in the language of graphs consisting only of complete binary trees, then $q$ is definable in $L^4_{\infty\omega}$.*

Define the class, $\mathcal{T}$, of labeled binary trees as the class of structure over the vocabulary $\{E, U\}$ which satisfy, in addition to the above six axioms, the following one:

7. $\forall x \forall y(\delta(x,y) \rightarrow ((Ux \wedge Uy) \vee (\neg Ux \wedge \neg Uy)))$

   That is, all vertices at the same level are either labeled or unlabeled.

Observe that the propositions shown above for complete binary trees apply equally well to labeled binary trees.

We also define the class, $\mathcal{B}$, of binary strings as structures over the same vocabulary $\{E, U\}$ that make true Axioms 2 through 5 above, as well as:

1'. $\forall x(\forall y(\neg Exy) \vee \exists y(Exy \wedge \forall z(Exz \rightarrow z = y)))$

   That is every vertex has exactly 0 or 1 children.

There is a natural correspondence between labeled binary trees and binary strings. In some sense, they encode the same information, with the $i^{\text{th}}$ bit of the binary string corrsponding to the $i^{\text{th}}$ level of the tree. While we give formal definitions below, it will be instructive to keep this intuitive picture in mind and we will make appeal to it to simplify the presentation.

**Definition 4.3** *If $B \in \mathcal{B}$ and $T \in \mathcal{T}$, then $B \triangleright T$ if and only if, for all $d$:*

- *$B \models \exists x \alpha_d$ if and only if $T \models \exists x \alpha_d$, and*
- *$B \models \forall x(\alpha_d \rightarrow Ux)$ if and only if $T \models \forall x(\alpha_d \rightarrow Ux)$.*

Note that if $B \triangleright T$ and the size of $B$ is $n$, then the size of $T$ is $2^n - 1$

**Definition 4.4** *For any queries $q_B \subseteq \mathcal{B}$ and $q_T \subseteq \mathcal{T}$, define:*

$$\begin{aligned} h(q_B) &= \{T | B \triangleright T \text{ for any } B \in q_B\} \\ h^{-1}(q_T) &= \{B | B \triangleright T \text{ for any } T \in q_T\} \end{aligned}$$

It should be clear that $h^{-1}(h(q_B)) = q_B$.

Lindell[Lin91] used this correspondence between binary strings and labeled binary trees to show that FO + LFP does not express all the polynomial-time queries on binary trees.

**Lemma 4.6** *If $q_B \in DTIME[2^{O(n)}]$ then $h(q_B) \in P$.*

**Proof:**

Given an input $T$, we can verify that it is a labeled binary tree in polynomial time, since $\mathcal{T} \in$ FO + LFP. We can also extract from it a $B$ such that $B \triangleright T$ in DSPACE[log($n$)]. We then pass $B$ as the input to the acceptor for $q_B$ which runs in time $2^{O(d)}$, where $d$ is the size of $B$, but this is only polynomial in the size of $T$. ■

**Lemma 4.7** *If $q_T \in FO + LFP$, then $h^{-1}(q_T) \in FO + LFP$.*

The proof of this lemma is based on a syntactic translation similar to the one given in Section 4.3.3. The key element of Lindell's construction is that $k$-tuples of vertices from the tree can be encoded as fixed length tuples in the corresponding binary string. This is because a complete set of invariants (up to automorphism) for a tuple on a complete binary tree is the sequence of depths of the least common ancestors of pairs of elements in the tuple. We refer to [Lin91] for details of the translation.

Given that there are queries on strings in DTIME[$2^{O(n)}$] that are not in P [HS65], we conclude the following:

**Theorem 4.3 ([Lin91])** *There is a $q_T \subseteq \mathcal{T}$ such that $q_T \in P$, but $q_T \notin FO + LFP$.*

Since we observed above that for every $q$ such that $q \subseteq \mathcal{T}$, $q \in L^4_{\infty\omega}$, we conclude that:

**Corollary 4.4** *$FO + LFP \subset L^\omega_{\infty\omega} \cap P$.*

Define the class FO + PFP|P of queries expressed by a formula of FO + PFP with the property that every occurrence of the **pfp** operation closes in polynomially many steps in any structure. Any query in FO + PFP|P is clearly computable in polynomial time. Also, since the operator **lfp** can be seen as an instance of **pfp** that always closes in polynomialy many steps, we get

$$\text{FO} + \text{LFP} \subseteq \text{FO} + \text{PFP|P} \subseteq L^\omega_{\infty\omega} \cap P$$

It had been conjectured that these three classes are, in fact, equal. We have shown above that the first and the third can be seperated. Abiteboul and Vianu[AV91b] have recently shown that the first and the second are equal if and only if P = PSPACE. This result is proved using a padding technique similar to the one above. We encoded binary strings of size $n$ as trees of size $2^n$. For the purpose of the next result, we will need to encode them into trees of size $2^{n^k}$. To this end, we introduce, for every $k$ the class of structures $\mathcal{T}_k$ over the signature $\{E, U, L\}$ which in addition to the Axioms 1 through 7, satisfy:

8. $\forall x \forall y(\delta(x, y) \rightarrow ((Lx \wedge Ly) \vee (\neg Lx \wedge \neg Ly)))$

   That is, all vertices at the same level are either in $L$ or not.

9. $\forall x \forall y((Lx \wedge Eyx) \rightarrow Ly)$

   If a vertex is in $L$, then so is its parent.

10. The depth of the tree is $n^k$, where $n$ is the number of levels labeled by $L$. This can be stated in FO + LFP by defining a $k$-ary induction on the levels in $L$ that is an ordering of length $n^k$ on $k$-tuples.

The binary string encoded by a tree in $\mathcal{T}_k$ of depth $n^k$ can be extracted by looking at the topmost $n$ levels (the levels labeled by $L$) and looking at the string defined by the relation $U$ on these levels. We can formalize this as before with a map $h_k$ from queries on binary strings to queries on $\mathcal{T}_k$.

**Lemma 4.8** *If $q_B \subseteq \mathcal{B}$ is a query in FO + LFP (respectively FO + PFP), then $h_k(q_B)$ is in FO + LFP (respectively FO + PFP).*

The translation is obtained simply by replacing the equality relation by the formula $\delta$ defined in Axiom 6 and by relatibvizing all quantifiers to $L$.

**Theorem 4.4 ([AV91b])** *FO + PFP| P = FO + LFP if and only if PSPACE = P.*

**Proof:**
One direction follows immediately from Theorem 4.2. In the other direction, suppose FO + PFP|P = FO + LFP. Let $S$ be a language in PSPACE and hence in DTIME[$2^{n^k}$] for some $k$. Let $q_B \subseteq \mathcal{B}$ be the collection of structures corresponding to strings in $S$. Since an ordering is easily (in FO + LFP) definable on structures in $\mathcal{B}$, $q_B \in$ FO + PFP. Hence $h_k(q_B) \in$ FO + PFP. Moreover, the translation in Lemma 4.8 does not increase the inductive depth of any occurrences of **pfp** in the formula defining $q_B$, which are therefore polynomial in the size of the trees in $\mathcal{T}_k$. Any new inductions introduced by the translation are expressible by **lfp** and hence also polynomial. Thus, $h_k(q_B) \in$ FO + PFP|P. By hypothesis, then, $h_k(q_B) \in$ FO + LFP and by Lemma 4.7, $q_B \in P$. ■

This result is remarkable in that it reduces the separation of P and PSPACE to the seperation two classes that are properly contained in P.

# Chapter 5

# Research Directions

In this chapter, we outline some proposals for continuing research in the area. While some of these are precisely formulated, in particular the first one below, the later ones are somewhat more tentative.

## 5.1 $L_k$ canonization

Define the problem of *$L_k$-canonization* as the problem of computing a function $F$ on structures $\mathcal{A}$ with the property that $F(\mathcal{A}) \equiv_k \mathcal{A}$ and if $\mathcal{A} \equiv_k \mathcal{B}$ then $F(\mathcal{A}) = F(\mathcal{B})$. We will also call $F(\mathcal{A})$ an $L_k$-canon of $\mathcal{A}$.

We noted earlier that the polynomial time properties of unordered graphs are recursively enumerable if the problem of canonical labeling of a graph has a polynomial time solution. We can use a similar argument to show that if we have, for every $k$, a polynomial time algorithm, $M_k$ that produces an *$L_k$-canon*, and moreover the set of such $M_k$ is recursively enumerable, then the class of queries in $L^{\omega}_{\infty\omega} \cap P$ is recursively indexable. This assumes that the $L_k$-canon produced is at most polynomial in the size of the input.

As of the time of this writing, the problem of an effective listing of polynomial time algorithms for $L_k$-canonization is still unresolved. However, the situation is different from the case of graph canonization. In the latter case, neither the isomorphism problem nor the canonical labeling problem are known to be in P though they are both known to be in NP. On the other hand, we know that we can test $L_k$ equivalence in polynomial time.

In [BG84], Blass and Gurevich defined four classes of problems associated with any equivalence relation $E$ over some domain $\mathcal{D}$:

**Equivalence Testing** Given two elements $x, y \in \mathcal{D}$, determine whether $xEy$.

**Invariant** Compute a function $F$ from $\mathcal{D}$ to some set $\mathcal{D}'$ such that $F(x) = F(y)$ just in case $xEy$.

**Normal Form** Compute an invariant function $F$ from $\mathcal{D}$ to $\mathcal{D}$ such that $F(x)Ex$.

**First Member** Given $x$ find the smallest $y$ (under some pre-determined ordering on $\mathcal{D}$) such that $xEy$.

It is clear that a polynomial time solution to any problem on this list yields a solution to all problems above it. Blass and Gurevich showed that the other direction does not necessarily hold. In fact, given a polynomial time equivalence test, the best that can be guaranteed about the first-member problem is that it is in $\Delta_2^p$, the second level of the polynomial hierarchy. Moreover, there are equivalence relations with a polynomial-time test for which the First Member problem is complete for $\Delta_2^p$.

We can now characterize the problem of $L_k$ canonization in the above terms as follows. The translation of a structure $\mathcal{A}$ to $E_k(\mathcal{A})$ is the computation of an invariant structure, in the sense that $E_k(\mathcal{A})$ and $E_k(\mathcal{B})$ isomorphic if $\mathcal{A} \equiv_k \mathcal{B}$ (we prove this result below). Since the translation produces ordered structures, two isomorphic structues are represented by identical binary strings. Thus, this gives us a polynomial time invariant function. To prove that $L^{\omega}_{\infty\omega} \cap P$ is recursively indexable, it would suffice to show that there is a polynomial time computable Normal Form function, since this would also guarantee that the $L_k$-canon was no more than polynomial in the size of the input.

We now give the proof that the translation $E_k$ does indeed compute an invariant structure.

**Theorem 5.1** *For any two structures $\mathcal{A}$ and $\mathcal{B}$, $\mathcal{A} \equiv_k \mathcal{B}$ if and only if $E_k(\mathcal{A}) \cong E_k(\mathcal{B})$*

**Proof:**

$\Rightarrow$ If $\mathcal{A} \equiv_k \mathcal{B}$ then every $L_k$-type that is realized in $\mathcal{A}$ is realized in $\mathcal{B}$ and vice versa. To see this, let $\bar{a}$ be an $l$-tuple from $\mathcal{A}$. Recall from Corollary 4.3 that there is a formula $\phi(x_1 \ldots x_l)$ in $L_k$ with $l$ free variables that expresses this type. But then, $\mathcal{A} \models \exists x_1 \ldots x_l \phi$ and therefore $\mathcal{B} \models \exists x_1 \ldots x_l \phi$. This tells us that the structures $E_k(\mathcal{A})$ and $E_k(\mathcal{B})$ have the same size.

Let $f$ be the order-preserving map from $E_k(\mathcal{A})$ to $E_k(\mathcal{B})$. If $f([\bar{a}]) = [\bar{b}]$, then $\bar{a}$ and $\bar{b}$ have the same $L_k$-type. This is because the definition of the ordering relation $<_k$ is uniform. As a result, the relations $U_i$ and $R_j$ are clearly preserved by $f$. Consider the case $X_i([\bar{a}], [\bar{a}']$. Let $\phi(x_1 \ldots x_{i+1})$ be the $L_k$ formula expressing the $L_k$-type of $\bar{a}'$. Then, $\exists x_{i+1}\phi$ is in the $L_k$-type of $\bar{a}$ and hence of any element of $f([\bar{a}])$. It follows that $X_i(f([\bar{a}]), f([\bar{a}'])$. Similarly, $f$ preserves the $P^i_\pi$ because the $L_k$-types are closed under permutations of the free variables. Thus, $f$ is an isomorphism.

$\Leftarrow$ Let $f$ be an isomorphism from $E_k(\mathcal{A})$ to $E_k(\mathcal{B})$. We show that Player II has a strategy for playing the $k$-pebble game on $\mathcal{A}$ and $\mathcal{B}$ indefinitely. Suppose that at some stage of the game, the pebbles are on the elements $\bar{a}$ and $\bar{b}$. Further suppose, without loss of generality, that Player I moves on $\mathcal{A}$ resulting in the configuration $\bar{a}'$. Player II finds a tuple $\bar{b}' \in f([\bar{a}'])$ such that $\bar{b}'$ is one move away from $\bar{b}$ and then plays that move. We need to show that such a $\bar{b}'$ can always be found. Note that we can assume, as an inductive hypothesis that $f([\bar{a}]) = [\bar{b}]$. There are two cases:

**Case 1:** $ln(\bar{a}) = i < k$ and Player I moves by placing an additional pebble on the board. Then, $X_i([\bar{a}], [\bar{a}']$ and therefore $X_i([\bar{b}], f([\bar{a}'])$ and there is a $\bar{b}' \in f([\bar{a}'])$ that is an extension of $\bar{b}$

**Case 2:** $ln(\bar{a}) = i + 1 \leq k$ and Player I moves by moving the pebble from $a_l$ to a new element. Then, there are $\bar{a}_1$ and $\bar{a}_2$ and a permutation $\pi$ on $i$ elements such that the following holds: $P^i_\pi([\bar{a}_1], [\bar{a}_2]) \wedge X_i([\bar{a}_1], [\bar{a}]) \wedge X_i([\bar{a}_2], [\bar{a}'])$. Once again, because $f$ is an isomorphism, $P^i_\pi(f([\bar{a}_1]), f([\bar{a}_2])) \wedge X_i(f([\bar{a}_1]), f([\bar{a}])) \wedge X_i(f([\bar{a}_2]), f([\bar{a}']))$ holds and we can get from $\bar{b}$ to $f([\bar{a}'])$ by moving $b_l$ ■

The most direct approach to constructuing an $L_k$-canon, given a polynomial time algorithm for the translation $E_k$, would be to try and invert $E_k$, *i.e.* given an input structure $\mathcal{K}$, to find an $\mathcal{A}$ such that $\mathcal{K} = E_k(\mathcal{A})$. However, this cannot be done in time polynomial in the size of $\mathcal{K}$. To see this, suppose for contradiction that we have a polynomial time computable $E_k^{-1}$ which acts as a translation from the range of $E_k$ into its domain. Since the range of $E_k$ consists of totally ordered structures, $E_k^{-1}$ is definable in FO + LFP. Composing this with the FO + LFP definition of $E_k$, we get an FO + LFP translation that yields an $L_k$ canon. and therefore that $L^\omega_{\infty,\omega} \cap \mathrm{P} \subseteq$ FO + LFP, which we know to be false. It is still conceivable that the computation of $E_k^{-1}$, while not polynomial in the size of the input $E_k(\mathcal{A})$ is polynomial in the size of $\mathcal{A}$, since the former could be much smaller. In fact, it is exactly the case where $E_k(\mathcal{A})$ is much smaller than $\mathcal{A}$ that demonstrated that FO + LFP$\neq L^\omega_{\infty,\omega} \cap \mathrm{P}$.

## 5.2 Relationship with GM$^{loose}$

Another line of investigation related to the one above is to establish the relationship between GM$^{loose}$, the model of generic computation defined in [AV91b] and $L^\omega_{\infty\omega}$. As we have already seen, the former corresponds to a fragment of the latter. We also know that all properties in GM$^{loose}$ are recursive. Does it define exactly the recursive fragment of $L^\omega_{\infty\omega}$? A related question is whether the restrictions of GM$^{loose}$ to various complexity classes correspond to similar

restrictions on $L^{\omega}_{\infty\omega}$. This might provide an alternative way of answering the question on the recursive enumerability of $L^{\omega}_{\infty,\omega} \cap \mathrm{P}$.

## 5.3 Adding Counting Quantifiers

We can define the language $C^{\omega}_{\infty,\omega}$ as $L^{\omega}_{\infty,\omega}$ augmented by the counting quantifiers described in Section 2.4. This immediately raises a number of questions regarding the extension of our results on $L^{\omega}_{\infty,\omega}$ to this language. Namely:

1. A polynomial time algorithm for testing $C_k$ equivalence is given in [IL90]. This yields an inductive ordering on the $C_k$-types in a manner analogous to the ordering on $L_k$ types presented in Section 4.3.1. Does this yield similar results seperating inductive $C_k$ logic from $C^{\omega}_{\infty,\omega} \cap \mathrm{P}$?

2. Can one obtain from this a $C_k$ canonization algorithm? In some sense, this problem seems easier than the one for $L_k$ because structures that are $C_k$ equivalent must have the same size, so we do not have a change in size between a structure and its canon.

3. The result in [CFI89] implies that $C^{\omega}_{\infty,\omega} \cap \mathrm{P}$ is properly contained in P. On the other hand, we know that $L^{\omega}_{\infty,\omega} \cap \mathrm{P}$ is properly contained in $C^{\omega}_{\infty,\omega} \cap \mathrm{P}$, since the latter can express parity. So, what is the expressive power of $C^{\omega}_{\infty,\omega} \cap \mathrm{P}$?

## 5.4 Other Inductive Operations

The language $L^{\omega}_{\infty,\omega}$ is interesting as a generalization of the inductive languages FO + LFP and FO + PFP, which in the presence of ordering capture natural complexity classes. Can we define other inductive operations within $L^{\omega}_{\infty,\omega}$ that correspond to complexity classes such as NP on ordering? In other words, can we define some kind of a non-deterministic inductive operation?

# Bibliography


[AV91a] S. Abiteboul and V. Vianu. Datalog extensions for database queries and updates. *Journal of Computer and System Sciences*, 43:62–124, 1991.

[AV91b] S. Abiteboul and V. Vianu. Generic computation and its complexity. In *Proceedings of the 23rd ACM Symposium on the Theory of Computing*, 1991.

[Bar73] J. Barwise. Back and forth through infinitary logic. In M. D. Morley, editor, *Studies in Model Theory*, Mathematical Association of America, 1973.

[Bar77] J. Barwise. On Moschovakis closure ordinals. *Journal of Symbolic Logic*, 42:292–296, 1977.

[BG84] A. Blass and Y. Gurevich. Equivalence relations, invariants, and normal forms. *SIAM Journal on Computing*, 13(4):682–689, 1984.

[CFI89] J-y. Cai, M. Fürer, and N. Immerman. An optimal lower bound on the number of variables for graph identification. In *Proceedings of the 30th IEEE Symposium on Foundations of Computer Science*, pages 612–617, 1989.

[CH82] A. Chandra and D. Harel. Structure and complexity of relational queries. *Journal of Computer and System Sciences*, 25:99–128, 1982.

[EFT84] H-D. Ebbinghaus, J. Flum, and W. Thomas. *Mathematical Logic.* Springer-Verlag, 1984.

[Ehr61] A. Ehrenfeucht. An application of games to the completeness problem for formalized theories. *Fund. Math.*, 49:129–141, 1961.

[End72] H. B. Enderton. *A Mathematical Introduction to Logic.* Academic Press, 1972.

[Fag74] R. Fagin. Generalized first-order spectra and polynomial-time recognizable sets. In R. M. Karp, editor, *Complexity of Computation, SIAM-AMS Proceedings, Vol 7*, pages 43–73, 1974.

[Gur84] Y. Gurevich. Toward logic tailored for computer science. In M. Richter et. al., editor, *Computation and Proof Theory*, pages 175–216, Springer Lecture Notes in Mathematics, 1984.

[HS65] J. Hartmanis and R. E. Stearns. On the computational complexity of algorithms. *Transactions of the AMS*, 117:285–306, 1965.

[IL90] N. Immerman and E. S. Lander. Describing graphs: a first-order approach to graph canonization. In A. Selman, editor, *Complexity Theory Retrospective*, Springer-Verlag, 1990.

[Imm86] N. Immerman. Relational queries computable in polynomial time. *Information and Control*, 68:86–104, 1986.

[Imm87] N. Immerman. Expressibility as a complexity measure: results and directions. In *Proceedings of the 2nd Conference on Structure in Complexity Theory*, pages 194–202, 1987.

[Imm89] N. Immerman. Descriptive and computational complexity. In J. Hartmanis, editor, *Computational Complexity Theory, Proc. of AMS Symposia in Appl. Math.*, pages 75–91, 1989.

[Kuč87] L. Kučera. Canonical labeling of regular graphs in linear average time. In *Proceedings of the 28th IEEE Symposium on Foundations of Computer Science*, pages 271–279, 1987.

[Lin87] S. Lindell. *The Logical Complexity of Queries on Unordered Graphs.* PhD thesis, University of California, Los Angeles, 1987.

[Lin91] S. Lindell. An analysis of fixed-point queries on binary trees. *Theoretical Computer Science*, 1991. To appear.

[Sto77] L. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3:1–22, 1977.

[Tra50] B. A. Trakhtenbrot. Impossibility of an algorithm for the decision problem in finite classes. *Dokdaly Akademii Nauk SSSR*, 70:569–572, 1950.

[Var82] M. Y. Vardi. The complexity of relational query languages. In *Proceedings of the 14th ACM Symposium on the Theory of Computing*, pages 137–146, 1982.