

UNIVERSITY OF CALIFORNIA
SANTA CRUZ

**Computational aspects of first-order logic
on finite structures**

A thesis submitted in partial satisfaction of the
requirements for the degree of

MASTER OF SCIENCE

in

COMPUTER SCIENCE

by

Albert Atserias

March 1999

The dissertation of Albert Atserias is approved:

Professor Phokion G. Kolaitis, Chair

Professor David P. Helmbold

Professor Manfred K. Warmuth

Dean of Graduate Studies

Copyright © by

Albert Atserias

1999

Contents

1	Introduction	1
1.1	Background	1
1.2	Outline of this thesis and summary of results	7
2	Preliminaries	12
2.1	Basic definitions	12
2.2	Logics	14
2.2.1	First-order logic	14
2.2.2	Second-order logic	17
2.2.3	Fixed-point logics	20
2.2.4	Definability of relations and queries	23
2.3	Models of computation	24
2.3.1	Turing machines	24
2.3.2	Complexity classes	27
2.3.3	Uniform circuits	28

3	Descriptive Complexity	30
3.1	Encodings and computability of queries	30
3.2	The complexity of logical definitions	31
3.2.1	First-order logic in LH	32
3.2.2	Least fixed-point logic in P	34
3.2.3	Existential second-order logic in NP	36
3.3	Logic captures computational complexity	37
3.3.1	Existential second-order logic captures NP	37
3.3.2	Least fixed-point logic captures P	39
3.3.3	First-order logic captures LH	41
4	Background on the Ordered Conjecture	46
4.1	Restricted classes of finite structures	46
4.2	Statement of the ordered conjecture	48
4.3	Restrictions and relaxations that hold true	49
4.3.1	Linear orders	49
4.3.2	Unary vocabularies	54
4.3.3	Partial fixed-point logic	56
5	The conjecture on finite arithmetical structures	60
5.1	Motivation	60
5.2	Arithmetical versus pure arithmetical structures	61
5.3	Definability on the standard model of arithmetic	68

5.3.1	Some definitions and easy facts	68
5.3.2	Absoluteness properties of Δ_0 -formulas and inductions	71
5.3.3	Characterization of absolute definable queries	73
5.3.4	Characterization of the absolute collapse	76
5.3.5	Characterization of the collapse	78
5.4	Evidence for the conjecture	79
6	Conclusions	80
6.1	Overview	80
6.2	Directions for future work	81
A	Alternating Turing machines with random access	84
A.1	A useful trick	84
A.2	The easy direction	86
A.3	Time-constructible sub-linear functions	88
A.4	The difficult direction	90
A.5	The bottom line	95
B	Arithmetic predicates	97
B.1	Computability of arithmetic predicates	97
B.2	Finite variants	97
B.3	Definability of arithmetic predicates	99

Computational aspects of first-order logic on finite structures

Albert Atserias

Abstract

Descriptive complexity aims to classify properties of finite structures according to the logical resources that are needed to express them. The Immerman-Vardi Theorem states that a property of finite ordered structures is computable in polynomial-time if and only if it can be expressed in least fixed-point logic. Here, least fixed-point logic is the extension of first-order logic with the ability to build-in inductive definitions. It is known that first-order logic is a fairly weak expressive language in the context of descriptive complexity. For example, least fixed-point logic is strictly more expressive than first-order logic on the class of all finite ordered structures.

The ordered conjecture, formulated by Kolaitis and Vardi in 1992, asks whether this is also the case for any infinite class of finite ordered structures. Although some significant progress has been made since its formulation, the ordered conjecture remains open. In fact, it has been shown that any way of resolving it would have important consequences in computational complexity theory. For if it fails, then P is not equal to $PSPACE$, and if it holds, then $LINH$ is not equal to E ; here $LINH$ is the linear-time hierarchy introduced by Wrathall, and E is the class of problems solvable in linear exponential-time.

The goal of this thesis is two-fold. First, we overview the progress made towards the ordered conjecture in recent years, and second, we contribute to the understanding of

problem with some new results. Before we get into the ordered conjecture itself, we survey some classical results in descriptive complexity. We state and prove Fagin's Theorem, the Immerman-Vardi Theorem, and a result due to Barrington, Immerman and Straubing asserting that first-order logic captures the logarithmic-time hierarchy on any class of finite arithmetical structures. Here, finite arithmetical structures are finite structures equipped with the standard arithmetic operations of addition and multiplication.

After this, we concentrate on the ordered conjecture. We begin with a survey of the literature. First, we show the well-known fact that least fixed-point logic is strictly more expressive than first-order logic on the class of finite linear orders. We also give a new proof of the result of Dawar, Lindell, and Weinstein that the ordered conjecture holds when restricted to unary vocabularies, and a simplified proof of the result of Dawar and Hella that partial fixed-point logic is strictly more expressive than first-order logic on any infinite class of finite ordered structures.

After this survey, we study the ordered conjecture when restricted to finite arithmetical structures. We show that the conjecture holds when the vocabulary is non-empty. Moreover, we show that when the vocabulary is empty, the problem is literally equivalent to the separation of LINH from E, and to an open problem in uniform circuit complexity pointed out by Gurevich, Immerman, and Shelah. We also observe that the conjecture on finite arithmetical structures is literally equivalent to the conjecture on finite structures equipped with the $\text{BIT}(n,m)$ predicate which is true if and only if the n -th bit of the binary representation of m is one. We demonstrate that the ordered conjecture on finite

arithmetical structures is equivalent to a question of definability on the standard model of arithmetic. Finally, we show that the conjecture holds on finite arithmetical structures, unless the polynomial-time hierarchy collapses.

Acknowledgment

I wish to thank my advisor Phokion Kolaitis for his permanent support and encouragement, and for teaching me everything I know in Finite Model Theory.

Chapter 1

Introduction

1.1 Background

Computability theory and mathematical logic have been interacting disciplines since their initial development in the early years of the 20th century. Seminal works of Gödel [Gö31], Church [Chu36], Turing [Tur36], and Kleene [Kle36] have contributed to formalize the intuitive notion of computation into a precise mathematical framework.

With the advance of computers later in the century, computability theory led to the field of complexity theory in which not only effective computations are considered, but their efficiency is also taken into account. Needless to say, mathematical logic has also been here a recurrent source of examples and motivation. Probably the most well-known instance of this fact is the seminal work of Cook [Coo71], showing that the satisfiability problem for propositional logic is **NP**-complete, and therefore, it plays a decisive role in the, by now, most important problem in complexity theory: the **P** versus **NP** question. On the other

hand, a surprising different connection between logic and complexity is found in the field of descriptive complexity. In his well-known paper, Fagin characterized the complexity class **NP** in terms of definability in existential second-order logic on finite models [Fag74]. This result linked finite model theory to complexity, and started a succession of results that turned the discipline into a promising approach to address the numerous open problems of this field.

This thesis is about descriptive complexity. The general framework of the field is as follows. Our objects of study are finite structures (also called models) for an arbitrary relational vocabulary; canonical examples are finite directed graphs $G = (V, E)$, which consist of a universe V , the set of nodes, and a binary relation on the universe $E \subseteq V^2$, the set of edges. Finite structures may be encoded under some standard representation, so that computation can be performed on them. The general objective of the field of descriptive complexity is to classify properties of finite structures according to the difficulty to express them in some logical language. As mentioned above, Fagin's paper started the field with the result that the properties that are expressible in existential second-order logic coincide exactly with the properties that are computable in polynomial-time by non-deterministic Turing machines. Here, existential second-order logic, denoted Σ_1^1 , is the logical language that enhances first-order logic with the ability to existentially quantify over subsets of the universe, or in general, over relations on the universe. Thus, properties that are computable in **NP**, such as “ G is a Hamiltonian graph”, admit definitions in this formalism; in this particular example, the existential second-order formula would say

that “there exists a simple path (existential second-order) that contains every node (first-order)”. This result is particularly remarkable because it provides a syntactical, machine independent characterization of **NP**. We say that Σ_1^1 captures **NP** on classes of finite structures.

In the early 1980’s, Immerman [Imm86], and independently, Vardi [Var82], obtained a logical characterization of the complexity class **P** too. However, their result is only about classes of finite structures that are equipped with a linear order on their universe, or, as we say, finite ordered structures. The result says that on all those classes, the properties that are expressible in least fixed-point logic are exactly those that are computable in polynomial-time by a deterministic Turing machine. Here, least fixed-point logic, denoted LFP, is the language that extends first-order logic with the ability to build-in inductive definitions. Thus, properties that are computable in **P**, such as “there is a path from a to b in graph G ”, admit definitions in this formalism; in this particular example, the inductive definition would say that “either a and b are connected by an edge (base case), or there exists an intermediate node that is connected by an edge to a , and from which there is a path to b in G (inductive case)”. We say that LFP captures **P** on classes of ordered finite structures. The need to restrict one-self to finite ordered structures is not apparent in the example above, but can be proved to be necessary for least fixed-point logic to express all properties computable in **P**; for example, very simple properties such as “the cardinality of graph G is even” are not expressible in least fixed-point logic on the class of all (non-ordered) finite graphs (see Chandra and Harel [CH82]). In fact, one of the

main open problems in descriptive complexity is to determine if there exists a logic (for an appropriate definition of this term) that captures \mathbf{P} on the class of all finite structures (see Gurevich [Gur88]). Note that by Fagin's theorem above, if no such logic exists, then $\mathbf{P} \neq \mathbf{NP}$.

Of course, before getting into fixed-point or second-order logics, one would like to know as much as possible about first-order logic, denoted FO. It was a well-known fact that every property of finite structures that is expressible in first-order logic is computable in logarithmic space, and thus, it is a fairly weak language. Moreover, simple properties such as “the cardinality of graph G is even” above, are not expressible in first-order logic on the class of all finite graphs either. One may ask if this property is expressible when graphs are equipped with a linear order, as it is the case of least fixed-point logic. It turns out that even on finite ordered structures, first-order logic is not able to count. But one can keep equipping structures with more and more powerful predicates, or as we say, built-in's. In this sense, if we consider classes of finite ordered structures equipped with the usual arithmetic operations $+$ and \times , then the properties that are expressible in first-order logic are exactly those that are computable in the Logarithmic-time Hierarchy \mathbf{LH} , introduced by Sipser (see [Bus87, BIS90] for more information). Here, the Logarithmic-time Hierarchy is the class of languages accepted by alternating Turing machines running in logarithmic-time and a constant number of alternations. This result is essentially due to Barrington, Immerman, and Straubing [BIS90]. We say that FO captures \mathbf{LH} on classes of finite arithmetical structures.

In the discussion above, we have focussed on properties of finite structures as Boolean queries; that is, a property that may or may not hold on a particular finite structure. But one may want to consider the more general concept of query of arbitrary arity, or uniform relation. For example, let C be a class of finite graphs, and let Q be the mapping that assigns to each graph $G \in C$, the set of pairs of nodes (a, b) of G that are connected by a path. We say that Q is a binary query on the class of graphs C . In general, an r -ary query on a class of finite structures C is essentially a mapping that assigns an r -ary relation to each finite structure in C . We may still consider expressibility of queries in particular logical languages, as well as their computability. It turns out that the three results mentioned above carry over to queries; that is, queries expressible in Σ_1^1 are exactly those computable in **NP** on arbitrary finite structures, queries expressible in LFP are exactly those computable in **P** on finite ordered structures, and queries expressible in first-order logic are exactly those computable in **LH** on arithmetical finite structures. From now on, the term “capture” refers to queries and not only to properties (Boolean queries).

We have seen that increasingly powerful built-in predicates allow logics to capture natural complexity classes. One may ask the following question: given a logic \mathcal{L} and a complexity \mathcal{C} , on which classes of finite structures does \mathcal{L} capture \mathcal{C} ? Fagin’s theorem answers the question when \mathcal{L} is Σ_1^1 and \mathcal{C} is **NP**; namely, it says that Σ_1^1 captures **NP** on *any* class of finite structures. For LFP and **P**, or FO and **LH**, the answers are not clear yet, and a satisfactory answer would be of theoretical and practical interest. We will

return to this question later. A seemingly more tractable question at this point is whether there exists any (non-trivial) class of finite structures on which first-order logic captures \mathbf{P} . In view of the Immerman-Vardi Theorem, the answer to this question is negative only if the so-called Ordered Conjecture, formulated by Kolaitis and Vardi in 1992, holds:

Conjecture 1 [KV92] *Let C be an infinite class of finite ordered structures over an arbitrary relational vocabulary. There exists a query on C that is definable in least fixed-point logic, but is not definable in first-order logic.*

It has turned out that any way of resolving the question would have important consequences in complexity theory. Dawar and Hella [DH95] have shown that if the conjecture fails, then $\mathbf{P} \neq \mathbf{PSPACE}$. In addition, Dawar, Lindell and Weinstein [DLW96] pointed out that if the conjecture holds, then $\mathbf{LINH} \neq \mathbf{E}$. Here, \mathbf{LINH} is the Linear-time Hierarchy introduced by Wrathall [Wra78], and \mathbf{E} is the class of languages accepted by deterministic Turing machines running in linear exponential-time, that is, time bounded by 2^{cn} where c is a constant, and n is the length of the input. The separation of \mathbf{LINH} from \mathbf{E} is the linear analogue of the separation of \mathbf{PH} from \mathbf{EXP} . Here \mathbf{PH} is the Polynomial-time Hierarchy introduced by Stockmeyer [Sto77], and \mathbf{EXP} is the class of languages accepted by deterministic Turing machines running in polynomial exponential-time, that is, time bounded by 2^{n^c} where c is a constant, and n is the length of the input. On the other hand, one can see that \mathbf{PH} and \mathbf{EXP} are the closures under polynomial-time many-one reductions of \mathbf{LINH} and \mathbf{E} respectively. Although both these separations are widely believed to hold, neither has been established so far.

1.2 Outline of this thesis and summary of results

This thesis is on the Ordered Conjecture. The goal is two-fold. First, we survey the progress made on the Ordered Conjecture since its formulation in 1992; and second, we contribute to the problem with some new results.

After a necessary chapter on technical preliminaries, we address the question of the descriptive complexity of the classes **NP**, **P**, and **LH** in Chapter 3. As mentioned in the previous section, built-in predicates allow logics to capture natural complexity classes. We re-prove the three results mentioned above. Namely, we prove Fagin's Theorem that Σ_1^1 captures **NP** on any class of finite structures; we prove the Immerman-Vardi Theorem that LFP captures **P** on any class of finite ordered structures; and we also prove that FO captures **LH** on any class of finite arithmetical structures. Our proof of the latter result essentially follows the one by Barrington, Immerman, and Straubing [BIS90]. However, two important differences need to be pointed out. First, our result is slightly more general in the sense that there are hardly any restrictions in our model of alternating Turing machine. Barrington, Immerman, and Straubing assume that their alternating Turing machines only query the input once in each path. Although they prove that alternating Turing machines that query the input at each step can be simulated by alternating machines that only query the input once in each path by only doubling the running time, they do not show that the number of alternations can be kept constant. In fact, their transformation introduces an alternation for each query that the simulated machine may ask. Therefore, their assumption on alternating Turing machines is essential for their

proof. However, using techniques introduced by Buss [Bus87], one can prove that such a simulation can be made to keep time and alternation bounds within a constant loss in efficiency. A careful proof of this fact is provided in Appendix A. The second difference comes from the fact that in their result, the predicate $\text{BIT}(x, y)$ that is true if the x -th bit of y is one, is available. In order to replace $\text{BIT}(x, y)$ by $+$ and \times , we need to make use of a difficult old result of Bennet [Ben62] saying that the graph of exponentiation is *constructive arithmetic*, together with the observation of Lindell [Lin92] that $+$, \times , and exponentiation are enough to define BIT. Again, since the main objective of this thesis is the Ordered Conjecture, the technical details that deviate from the original proof of Barrington, Immerman, and Straubing are delegated to Appendix B.

In Chapter 4 we formulate the statement of the Ordered Conjecture. Before that, we address the question we left open in the previous section: on which classes of structures does least fixed-point logic capture \mathbf{P} ? Our small contribution to this open question is that any such class has to be of asymptotic uniform probability zero. Our proof uses a result of Babai, Erdős, and Selkow [BES80] on random graphs, together with a result of Hella, Kolaitis, and Luosto [HKL94] that characterizes the classes of finite rigid structures on which a linear order is definable in least fixed-point logic. Then we turn to the question: on which classes of structures does first-order logic capture \mathbf{P} ? We observe that there is no such (non-trivial) class only if the Ordered Conjecture is true. This leads us to the study of the conjecture itself. It is known that several restrictions and relaxations of the Ordered Conjecture are true. Namely, it is known that on the class of all linear orders,

fixed-point logic is more powerful than first-order logic. We give a complete proof of this result, and its easy extension to the class of *all* ordered structures for any particular vocabulary. Then we focus on a different restriction: unary vocabularies. Dawar, Lindell, and Weinstein [DLW96] showed that the Ordered Conjecture holds when restricted to unary vocabularies, that is, vocabularies that only contain unary relation symbols. More precisely, for every infinite class of finite ordered structures for a unary vocabulary, least fixed-point logic is more expressive than first-order logic. Their proof uses a result of Poizat [Poi82], and independently Immerman and Kozen [IK89], together with powerful techniques from bounded-variable logics. We give an alternative proof here that relies on Poizat's result, together with the fact that the combined complexity of first-order logic with a bounded number of variables is in \mathbf{P} (see Vardi [Var96]). Then we consider a relaxation of the Ordered Conjecture. We give the proof of a result of Dawar and Hella that partial fixed-point logic is more expressive than first-order logic on any infinite class of finite ordered structures. Here, partial fixed-point logic, PFP for short, is the extension of least fixed-point logic to make inductive definitions that are not necessarily positive. The proof is a simple diagonalization using the fact that PFP captures \mathbf{PSPACE} on classes of finite ordered structures (see Abiteboul and Vianu [AV95]). As a consequence, Dawar and Hella point out that if the Ordered Conjecture fails, then $\mathbf{P} \neq \mathbf{PSPACE}$.

The last result of Chapter 4 suggests that a counter-example to the Ordered Conjecture, if any, would be difficult to find. In Chapter 5 we seek complexity-theoretic consequences of the Ordered Conjecture holding true. Since arithmetical built-in predicates

provided a computational characterization of first-order logic, it is natural to compare LFP with FO on those structures. We first show that the Ordered Conjecture holds on the class of all finite arithmetical structures for a vocabulary that contains at least one relation symbol. The proof is an easy consequence of the result of Furst, Saxe, and Sipser [FSS84], and independently, Ajtai [Ajt83], that the parity function does not have bounded-depth, unbounded fan-in, polynomial-size circuits. Surprisingly as it seems, the cumbersome hypothesis that the vocabulary contains at least one relation symbol is not easy to remove. Namely, we show that if the Ordered Conjecture holds on the class of finite pure arithmetical structures, that is, finite arithmetical structures for the empty vocabulary, then $\mathbf{LINH} \neq \mathbf{E}$. This was already observed by Dawar, Lindell, and Weinstein [DLW96]. In fact, we show that the two questions are literally equivalent, and in turn, they are equivalent to an important question on circuit uniformity that we discuss next. Gurevich, Immerman, and Shelah [GIS94] pointed out without proof that LFP is more expressive than FO on the class of finite ordered structures with only the BIT predicate if and only if $\mathbf{DLOGTIME}$ -uniform $\mathbf{AC}^0 = \mathbf{P}$ -uniform \mathbf{AC}^0 . Here \mathcal{C} -uniform \mathbf{AC}^0 is the class of languages that are accepted by constant depth, unbounded fan-in, polynomial-size circuits that are \mathcal{C} -uniform in the sense that the description of each circuit is computable in the complexity class \mathcal{C} . In Chapter 2 we give precise definitions of these classes. Here we prove the claim of Gurevich, Immerman, and Shelah, and prove its equivalence to the Ordered Conjecture on finite pure arithmetical structures.

In the next section of Chapter 5 we turn to a different question. We observe a connec-

tion between the Ordered Conjecture on finite pure arithmetical structures and definability on the standard model of arithmetic $\mathbb{N} = (\omega, +, \times, \leq)$ where $\omega = \{0, 1, \dots\}$ is the set of natural numbers. Consider first-order formulas of arithmetic where all the quantifiers are of the form $(\forall x \leq y)$ or $(\exists x \leq y)$. Let us call quantifiers of this type bounded, and let us call the class of formulas of this type Δ_0 . Our result is that the Ordered Conjecture holds on finite pure arithmetical structures if and only if every least fixed-point of a positive Δ_0 -formula is equivalent to a Δ_0 -formula on the standard model of arithmetic. This is not too surprising after all, because Δ_0 -formulas define exactly the constructive arithmetic predicates, which in turn, are exactly the rudimentary predicates as shown by Bennet [Ben62], which in turn, are exactly the predicates computable in **LINH**, as shown by Wrathall [Wra78]. Our proof follows a completely different approach. We characterize the queries on finite pure arithmetical structures that are expressible by Δ_0 -formulas, and least fixed-points of positive Δ_0 -formulas, as those queries that are *absolute* for a precise definition of this term. This characterization provides a *reflection* result to the standard model of arithmetic using standard absoluteness arguments.

We conclude Chapter 5 with a result that gives evidence that the Ordered Conjecture is true on finite pure arithmetical structures. We show that its failure would yield the collapse of the Polynomial-time Hierarchy. The proof uses the well-known fact that **PH** does not have complete problems unless it collapses (see [BDG96]).

Chapter 2

Preliminaries

2.1 Basic definitions

In this section we define the basic concepts and notation that will be used throughout the text. An *alphabet* is a finite set of *symbols*. In the following, let Σ denote a non-empty alphabet. A *word over Σ* is a finite sequence of symbols from Σ . The *empty word* is denoted λ , and corresponds to the empty sequence. The *length* of a word w is denoted $|w|$, and is the length of the sequence. The set of all words over Σ of length n is denoted Σ^n . The set of all words is $\Sigma^* = \bigcup_{n \geq 0} \Sigma^n$. The set of non-empty words is $\Sigma^+ = \bigcup_{n > 0} \Sigma^n$. A *language* is a subset of Σ^* .

Example 1 Let Σ be the finite alphabet that consists of the two symbols 0 and 1. A word over Σ is called a binary word. We should denote words as tuples like in $(0, 1, 0, 0, 0, 1, 0)$. We will rather use the notation 0100010 to denote the same word. For a symbol a , the

notation a^n will be a shortcut for the word $aa \cdots a$ of length n . For a word w , the i -th symbol of w is denoted w_i . An example of a language over Σ is the set $L = \{0^n 1^n : n \geq 0\}$ of binary words consisting of a row of 0's followed by a row of 1's of the same length. Note that $\lambda \in L$.

For us, natural numbers are finite ordinals, and ω is the set of all natural numbers. This means that $\omega = \{0, 1, \dots\}$ where $n = \{0, 1, \dots, n-1\}$; in particular, $0 = \emptyset$. The standard linear order of an ordinal α is the binary relation $\{(\gamma, \delta) \in \alpha^2 : \gamma \in \delta\}$; thus, the standard linear order of a natural number n is just $\{(p, q) \in \{0, \dots, n-1\} : p < q\}$. For a set A , the notation $\mathcal{P}(A)$ stands for the set of its subsets. It is common in the theory of computation to use the notation $\log n$ for the length of the shortest binary representation for the natural number n . If we wish to use the true logarithm to the base two, we use the notation $\log_2(n)$. Thus, $\log n = \lfloor \log_2(n) \rfloor + 1$. For natural numbers $m \leq n$, we let $b_n(m)$ be the unique word from $\{0, 1\}^{\log n}$ that represents m in binary.

A relational *vocabulary* (or *signature*, *similarity type*, *logical language*) is a finite sequence of *relation symbols* (we will not consider functions, constants, or infinite vocabularies in this text). To each relation symbol R of a vocabulary will be associated a positive natural number $\text{ar}(R)$ called its *arity* (we may not consider 0-ary, or propositional, relations). In the following, let $\sigma = (R_1, \dots, R_s)$ denote a relational vocabulary where the R_i 's are distinct relation symbols. A *structure* (or *model*) for σ is a sequence $\mathfrak{A} = (A, R_1^{\mathfrak{A}}, \dots, R_s^{\mathfrak{A}})$ where A is a non-empty set called the *universe*, and $R_i^{\mathfrak{A}}$ is the *interpretation* of the symbol R_i ; here, $R_i^{\mathfrak{A}}$ is an $\text{ar}(R_i)$ -ary relation on A . It will be

convenient to assume that universes are always ordinals, and so we do. Two structures $\mathfrak{A} = (A, R_1^{\mathfrak{A}}, \dots, R_s^{\mathfrak{A}})$ and $\mathfrak{B} = (B, R_1^{\mathfrak{B}}, \dots, R_s^{\mathfrak{B}})$ for σ are *isomorphic*, denoted $\mathfrak{A} \cong \mathfrak{B}$, if and only if there exists a bijective function $g : A \rightarrow B$ such that for every relation symbol R_i of arity $r = \text{ar}(R_i)$ and every $(a_1, \dots, a_r) \in A^r$ it holds that $(a_1, \dots, a_r) \in R_i^{\mathfrak{A}}$ if and only if $(g(a_1), \dots, g(a_r)) \in R_i^{\mathfrak{B}}$. The class of all structures for σ is denoted $\text{Mod}(\sigma)$. A class of structures for σ is just a subset of $\text{Mod}(\sigma)$. Letters \mathfrak{A} and \mathfrak{B} will denote structures, and letters A and B will denote their universes.

Example 2 Consider the vocabulary $\sigma = (E)$ consisting of a single relation symbol E of arity two. This vocabulary can be used to represent graphs. In effect, a structure for σ , say $\mathfrak{A} = (A, E^{\mathfrak{A}})$, represents the graph whose set of vertices is A and whose edge relation is $E^{\mathfrak{A}}$. In fact, for \mathfrak{A} to represent a valid graph we need $E^{\mathfrak{A}}$ to be an irreflexive and symmetric relation. An example of a set of structures for σ is the set of all structures for σ that represent a connected graph.

2.2 Logics

In this section we introduce three logical formalisms that allow us to talk about properties of structures for a given vocabulary.

2.2.1 First-order logic

Let $\Xi_0 = \{\wedge, \neg, \forall, (,), , =, \mathbf{v}, 0, 1\}$ be an alphabet. The set of *first-order variables*, denoted \mathcal{V}_0 , is defined as the set of all words from Ξ^* that start with \mathbf{v} and are followed by a non-

empty word of zeros and ones; formally, $\mathcal{V}_0 = \{vw \in \Xi^* : w \in \{0, 1\}^+\}$. For example, $v01100$ and $v101$ are first-order variables. Let $\sigma = (R_1, \dots, R_s)$ be a relational vocabulary, and let r_i be the arity of R_i . We say that a word φ from $(\Xi \cup \{R_1, \dots, R_s\})^*$ is a *first-order formula for σ* , denoted $\varphi \in \text{FO}(\sigma)$, if and only if one of the following holds:

- (i) φ is $x = y$ for some $x, y \in \mathcal{V}_0$,
- (ii) φ is $R_i(x_1, \dots, x_{r_i})$ for some $x_1, \dots, x_{r_i} \in \mathcal{V}_0$,
- (iii) φ is $\neg\psi$ for some $\psi \in \text{FO}(\sigma)$,
- (iv) φ is $(\psi \wedge \theta)$ for some $\psi, \theta \in \text{FO}(\sigma)$,
- (v) φ is $(\forall x)(\psi)$ for some $\psi \in \text{FO}(\sigma)$ and $x \in \mathcal{V}_0$,

Given a first-order formula φ , we say that $x \in \mathcal{V}_0$ is a *free variable* in φ if it does not appear within the scope of a quantifier \forall . Formally, the *set of first-order free variables of φ* , denoted $F_0(\varphi)$, is defined inductively on the construction of φ . Thus, $F_0(x = y) = \{x, y\}$, $F_0(R_i(x_1, \dots, x_{r_i})) = \{x_1, \dots, x_{r_i}\}$, $F_0(\neg\psi) = F_0(\psi)$, $F_0((\psi \wedge \theta)) = F_0(\psi) \cup F_0(\theta)$, and $F_0((\forall x)(\psi)) = F_0(\psi) - \{x\}$. A *sentence* is a formula without free variables. We may use three more symbols as abbreviations. Namely, $(\psi \vee \theta)$ may be used as an abbreviation for $\neg(\neg\psi \wedge \neg\theta)$; similarly, $(\psi \rightarrow \theta)$ may be used as an abbreviation for $\neg(\psi \wedge \neg\theta)$, and $(\exists x)(\psi)$ may be used as an abbreviation for $\neg(\forall x)(\neg\psi)$. We may also drop some unnecessary parenthesis by interpreting \wedge and \vee as associative operators on formulas, and giving them less priority than \neg , and more priority than \rightarrow . An example may help.

Example 3 Let $\sigma = (E)$ be the vocabulary of directed graphs and let $x, y \in \mathcal{V}_0$ be two first-order variables. The word $(\exists y)(\forall x)(\neg y = x \rightarrow E(x, y) \wedge \neg E(y, x))$ is an abbreviation for

the following first-order formula for σ : $\neg(\forall y)(\neg(\forall x)(\neg(\neg y = x \wedge \neg(E(x, y) \wedge \neg E(y, x))))))$.

Observe that \neg has taken maximum priority, followed by \wedge , and finally \rightarrow .

We have defined the syntax of first-order logic, and we are ready to turn to its semantics. As expected, first-order sentences will be assigned a truth value in each structure for its vocabulary. Given a structure \mathfrak{A} for σ , a *valuation function for \mathfrak{A}* is a mapping from the set of first-order variables to the universe of A . Given a valuation function $f : \mathcal{V}_0 \rightarrow A$, a variable $x \in \mathcal{V}_0$, and an element $a \in A$, the notation $f|_a^x$ denotes the valuation function such that $f|_a^x(y) = f(y)$ if $y \neq x$, and $f|_a^x(y) = a$ if $y = x$. We say that \mathfrak{A} and f *satisfy* the first-order formula φ for σ , denoted $\mathfrak{A} \models \varphi[f]$, if and only if one of the following holds:

- (i) φ is $x = y$ and $f(x) = f(y)$,
- (ii) φ is $R_i(x_1, \dots, x_{r_i})$ and $(f(x_1), \dots, f(x_{r_i})) \in R_i^{\mathfrak{A}}$,
- (iii) φ is $\neg\psi$ and not $\mathfrak{A} \models \psi[f]$,
- (iv) φ is $(\psi \wedge \theta)$ with $\mathfrak{A} \models \psi[f]$ and $\mathfrak{A} \models \theta[f]$,
- (v) φ is $(\forall x)(\psi)$ and for every $a \in A$ we have $\mathfrak{A} \models \psi[f|_a^x]$.

If φ is a formula with free variables among x_1, \dots, x_k appearing syntactically in this order, the notation $\mathfrak{A} \models \varphi[a_1, \dots, a_k]$ for arbitrary elements $a_1, \dots, a_k \in A$, means that $\mathfrak{A} \models \varphi[f]$ for every valuation function such that $f(x_i) = a_i$, $i = 1, \dots, k$.

The notion of satisfaction associates a truth value in a structure to each sentence. Thus, we say that a sentence φ is *true in \mathfrak{A}* if and only if $\mathfrak{A} \models \varphi$. Similarly, satisfaction associates a relation of the universe of a structure to each formula with free variables.

Thus, if φ is formula with free variables x_1, \dots, x_k appearing syntactically in this order, the k -ary relation defined by φ in \mathfrak{A} , denoted $\varphi^{\mathfrak{A}}$, is the set

$$\{(a_1, \dots, a_k) \in A^k : \mathfrak{A} \models \varphi[a_1, \dots, a_k]\}$$

We say that φ *defines* a relation $R \subseteq A^k$ in \mathfrak{A} if and only if $\varphi^{\mathfrak{A}} = R$.

Example 4 Let φ be the first-order sentence of Example 2.2.1. Then, φ is true in every directed graph that contains a *sink*; that is, a node that receives an edge from every other node except, maybe, itself, and that does not send an edge to any other node. If we drop the first existential quantifier in φ , we obtain the first-order formula $(\forall y)(\neg y = x \rightarrow E(y, x) \wedge \neg E(x, y))$ with one free variable x . Call ψ to this formula. For a directed graph \mathfrak{A} , the set $\psi^{\mathfrak{A}}$ is the set of sink nodes of \mathfrak{A} .

2.2.2 Second-order logic

We wish to enhance first-order logic with the ability to quantify over sets and relations of individuals. Let Ξ_1 be the alphabet $\Xi \cup \{\mathbf{V}\}$; that is, Ξ_1 is the extension of Ξ with a new symbol \mathbf{V} . The set of *second-order variables*, denoted \mathcal{V}_1 , is defined as the set of all words from Ξ_1^* that start with \mathbf{V} and are followed by a non-empty word of zeros and ones; formally, $\mathcal{V}_1 = \{\mathbf{V}w \in \Xi_1^* : w \in \{0, 1\}^+\}$. For example, $\mathbf{V}01100$ and $\mathbf{V}101$ are second-order variables. To each second-order variable $X \in \mathcal{V}_1$, we associate an arity, denoted $\text{ar}(X)$, with the property that there are infinitely many second-order variables of each arity. Let $\sigma = (R_1, \dots, R_s)$ be a relational vocabulary, and let r_i be the arity of R_i . We say that a

word Φ from $(\Xi_1 \cup \{R_1, \dots, R_s\})^*$ is a *second-order formula* for σ , denoted $\Phi \in \text{SO}(\sigma)$, if and only if one of the following holds:

- (i) Φ is in $\text{FO}((\sigma, X_1, \dots, X_{s'}))$ for some $s' \in \omega$ and $X_1, \dots, X_{s'} \in \mathcal{V}_1$,
- (ii) Φ is $\neg\Psi$ for some $\Psi \in \text{SO}(\sigma)$,
- (iii) Φ is $(\Psi \wedge \Theta)$ for some $\Psi, \Theta \in \text{SO}(\sigma)$,
- (iv) Φ is $(\forall x)(\Psi)$ for some $\Psi \in \text{SO}(\sigma)$ and $x \in \mathcal{V}_0$,
- (v) Φ is $(\forall X)(\Psi)$ for some $\Psi \in \text{SO}(\sigma)$ and $X \in \mathcal{V}_1$,

We may also use symbols \forall , \rightarrow , and \exists as abbreviations like in the case of first-order formulas. Analogously, we may drop some unnecessary parenthesis. As with first-order formulas, we define the set of *free second-order variables* in the usual way. We say that a second-order formula Φ for σ is *second-order quantifier-free*, denoted $\Phi \in \Delta_0^1(\sigma)$, if and only if Φ satisfies clause (i) above. We say that it is *existential second-order*, denoted $\Phi \in \Sigma_1^1(\sigma)$, if and only if it is $(\exists X_1) \cdots (\exists X_{s'}) (\Psi)$ for some $s' \in \omega$, $X_1, \dots, X_{s'} \in \mathcal{V}_1$, and $\Psi \in \Delta_0^1(\sigma)$. Finally, it is *universal second-order*, denoted $\Phi \in \Pi_1^1(\sigma)$, if and only if it is $(\forall X_1) \cdots (\forall X_{s'}) (\Psi)$ for some $s' \in \omega$, $X_1, \dots, X_{s'} \in \mathcal{V}_1$, and $\Psi \in \Delta_0^1(\sigma)$.

Example 5 Let $\sigma = (E)$ be the vocabulary of directed graphs, and let $x, y \in \mathcal{V}_0$ and $X_i \in \mathcal{V}_1$ with $\text{ar}(X_i) = 1$. Then, $(\exists X_1)(\exists X_2)(\exists X_3)((\forall x)(\bigvee_{i=1}^3 X_i(x)) \wedge (\forall x)(\forall y)(E(x, y) \rightarrow \bigwedge_{i=1}^3 (\neg X_i(x) \vee \neg X_i(y))))$ is an abbreviation for a second-order sentence. In fact, a $\Sigma_1^1(\sigma)$ sentence.

The semantics are defined as for first-order logic, except that second-order variables need to be interpreted too. Given a structure \mathfrak{A} for σ , a *second-order valuation function* is a mapping from the set of second-order variables, to the set of relations on the universe of \mathfrak{A} . We require that r -ary second-order variables be mapped to r -ary relations. Given a valuation function $F : \mathcal{V}_1 \rightarrow \bigcup_{r \geq 0} A^r$, an r -ary second order variable $X \in \mathcal{V}_1$, and a relation $B \subseteq A^r$, the notation $F|_B^X$ denotes the valuation function such that $F|_B^X(Y) = F(Y)$ is $Y \neq X$, and $F|_B^X(Y) = B$ if $Y = X$. Let f and F be first-order and second-order valuation functions respectively. We say that \mathfrak{A} , f and F *satisfy* the second-order formula Φ for σ , denoted $\mathfrak{A} \models \Phi[f, F]$, if and only if one of the following holds:

- (i) Φ is in $\text{FO}((\sigma, X_1, \dots, X_s))$ and $(\mathfrak{A}, F(X_1), \dots, F(X_s)) \models \Phi[f]$,
- (ii) Φ is $\neg\Psi$ and not $\mathfrak{A} \models \Psi[f, F]$,
- (iii) Φ is $(\Psi \wedge \Theta)$ and $\mathfrak{A} \models \Psi[f, F]$ and $\mathfrak{A} \models \Theta[f, F]$,
- (iv) Φ is $(\forall x)(\Psi)$ and for every $a \in A$ we have $\mathfrak{A} \models \Psi[f|_a^x, F]$,
- (v) Φ is $(\forall X)(\Psi)$ and for every $B \subseteq A^{\text{ar}(X)}$ we have $\mathfrak{A} \models \Psi[f, F|_B^X]$.

If the free first-order and second-order variables of Φ are x_1, \dots, x_k , and $X_1, \dots, X_{k'}$ respectively, syntactically in this order, then the notation $\mathfrak{A} \models \Phi[a_1, \dots, a_k, B_1, \dots, B_{k'}]$ means that $\mathfrak{A} \models \Phi[f, F]$ for every f and F such that $f(x_i) = a_i$ and $F(X_i) = B_i$.

Just as first-order formulas with free variables define relations on a structure, second-order formulas with free first-order and second-order variables define operators on relations on a structure. Thus, if Φ is a second-order formula with free variables x_1, \dots, x_k and

$X_1, \dots, X_{k'}$, the *operator* defined by Φ in \mathfrak{A} , denoted $\Phi^{\mathfrak{A}}$, is the mapping

$$(B_1, \dots, B_{k'}) \mapsto \{(a_1, \dots, a_k) \in A^k : \mathfrak{A} \models \Phi[a_1, \dots, a_k, B_1, \dots, B_{k'}]\}$$

where $B_i \subseteq A^{\text{ar}(X_i)}$. We may write $\Phi^{\mathfrak{A}}[B_1, \dots, B_{k'}]$ to denote $\Phi^{\mathfrak{A}}(B_1, \dots, B_{k'})$. If Φ contains no free second-order variable, then $\Phi^{\mathfrak{A}}$ is just a relation on \mathfrak{A} (a constant operator).

Example 6 Let Φ be the second-order sentence of Example 2.2.2. Then Φ is true in every directed graph that is 3-colorable. If we drop the first existential quantifier in Φ , we obtain a second-order formula with one free variable X_1 . This formula defines the operator on directed graphs that maps every subset B of its set of nodes to $\{()\}$ or \emptyset , according to whether it can be extended to a 3-coloring or not respectively. We say that the formula defines a Boolean operator.

2.2.3 Fixed-point logics

We wish to extend first-order logic with the ability to build-in inductive definition. Recall the definition of second-order variables in the previous section. Let Ξ_{FP} be the alphabet $\Xi_1 \cup \{\text{FP}\}$; that is, Ξ_{FP} is the extension of Ξ_1 with a new symbol FP.

Let $\sigma = (R_1, \dots, R_s)$ be a relational vocabulary, and let r_i be the arity of R_i . We say that a word ϕ from $(\Xi_{\text{FP}} \cup \{R_1, \dots, R_s\})^*$ is a *fixed-point formula for σ* , denoted $\phi \in \text{FP}(\sigma)$, if and only if one of the following holds:

- (i) ϕ is FO(σ),
- (ii) ϕ is $\neg\psi$ for some $\psi \in \text{FP}(\sigma)$,

- (iii) ϕ is $(\psi \wedge \theta)$ for some $\psi, \theta \in \text{FP}(\sigma)$,
- (iv) ϕ is $(\forall x)(\psi)$ for some $x \in \mathcal{V}_0$ and $\psi \in \text{FP}(\sigma)$,
- (v) ϕ is $(\text{FP}_{x_1, \dots, x_k, X}\psi)$ for some $x = x_1, \dots, x_k \in \mathcal{V}_0$, $X \in \mathcal{V}_1$ with $\text{ar}(X) = k$, and $\psi \in \text{FP}((\sigma, X))$.

We may use abbreviations as usual. The free variables of fixed-point formulas are defined as usual with the additional fact that in $(\text{FP}_{x_1, \dots, x_k, X}\phi)$, variable X is not free. Let ϕ be a $\text{FP}(\sigma)$ formula for the vocabulary (σ, X) where $X \in \mathcal{V}_1$ does not belong to σ . We say that X *occurs positively* in ϕ if and only if X occurs free in ϕ within the scope of an even number of negations. We say that a word ϕ from $(\Xi_{\text{FP}} \cup \{R_1, \dots, R_s\})^*$ is a *least fixed-point formula* for σ , denoted $\text{LFP}(\sigma)$, if and only if one of the following holds:

- (i) ϕ is $\text{FO}(\sigma)$,
- (ii) ϕ is $\neg\psi$ for some $\psi \in \text{LFP}(\sigma)$,
- (iii) ϕ is $(\psi \wedge \theta)$ for some $\psi, \theta \in \text{LFP}(\sigma)$,
- (iv) ϕ is $(\forall x)(\psi)$ for some $x \in \mathcal{V}_0$ and $\psi \in \text{LFP}(\sigma)$,
- (v) ϕ is $(\text{FP}_{x_1, \dots, x_k, X}\psi)$ for some $x = x_1, \dots, x_k \in \mathcal{V}_0$, $X \in \mathcal{V}_1$ with $\text{ar}(X) = k$, and $\psi \in \text{LFP}((\sigma, X))$ in which X occurs positively or does not occur.

Notice that $\text{LFP}(\sigma) \subseteq \text{FP}(\sigma)$. We may sometimes write $(\text{LFP}_{\bar{x}, X}\phi)$ or $(\text{PFP}_{\bar{x}, X}\phi)$ to emphasize the fact that $(\text{FP}_{\bar{x}, X}\phi)$ is a least fixed-point, or general (partial) fixed-point formula respectively.

Example 7 Let $\sigma = (E)$ be the vocabulary of directed graphs, and let $x, y \in \mathcal{V}_0$, and

$X \in \mathcal{V}_1$ with $\text{ar}(X) = 2$. The word $(\text{LFP}_{x,y,X} x = y \vee (\exists z)(X(x,z) \wedge E(z,y)))$ is an abbreviation for a least fixed-point formula for σ . Observe that x and y occur free.

We turn to the semantics. Let $G : \mathcal{P}(A) \rightarrow \mathcal{P}(A)$ be an operator. We say that G is *monotone* if and only if for every $B, C \subseteq A$ with $B \subseteq C$ we have that $G(B) \subseteq G(C)$. Let $\alpha > 0$ be a successor ordinal. We define the α -*stage of G* , denoted I_G^α , as $G(I_G^{\alpha-1})$. If α is a limit ordinal, then $I_G^\alpha = \bigcup_{\beta < \alpha} I_G^\beta$. If there exists an α such that $I_G^\alpha = I_G^{\alpha+1}$, we let $I_G = I_G^\alpha$; otherwise, we let $I_G = \emptyset$. One can see that if G is monotone, then $I_G^\alpha = I_G^{\alpha+1}$ for some α . Moreover, in this case, I_G is the least fixed-point of the operator G . Let \mathfrak{A} be a structure for σ , let f be a valuation function, and let ϕ be a fixed-point formula for σ . We say that \mathfrak{A} and f satisfy ϕ if and only if one of the following holds:

- (i) ϕ is in $\text{FO}(\sigma)$ and $\mathfrak{A} \models \phi[f]$,
- (ii) ϕ is $\neg\psi$ and not $\mathfrak{A} \models \psi[f]$,
- (iii) ϕ is $(\psi \wedge \theta)$ and $\mathfrak{A} \models \psi[f]$ and $\mathfrak{A} \models \theta[f]$,
- (iv) ϕ is $(\forall x)(\psi)$ and for all $a \in A$ we have that $\mathfrak{A} \models \psi[f|_a^x]$,
- (v) ϕ is $(\text{FP}_{x_1, \dots, x_k, X} \psi)$ and $(f(x_1), \dots, f(x_k)) \in I_G$ where $G : \mathcal{P}(A^k) \rightarrow \mathcal{P}(A^k)$ is the operator defined as $G(B) = \{(a_1, \dots, a_k) \in A^k : \mathfrak{A} \models \psi[f|_{a_1, \dots, a_k}^{x_1, \dots, x_k}, B]\}$.

One can see that if ϕ is a fixed-point formula positive in X , then the operator defined as $G(B)$ in the definition is monotone. It follows that for least fixed-point formulas, fixed-points always exist. As with first-order logic, fixed-point formulas define relations on structures. We denote by $\phi^{\mathfrak{A}}$ the first-order relation defined by ϕ on \mathfrak{A} .

Example 8 Consider the least fixed-point formula ϕ in Example 2.2.3. Then, ϕ defines on graphs the set of pairs of nodes that are connected by a path. On the other hand, the sentence $(\forall x)(\forall y)(\phi)$ is true on a graph if and only if it is connected.

2.2.4 Definability of relations and queries

We have already seen that first-order formulas define relations on structures. Similarly, second-order formulas without free second-order variables define relations too, and so do fixed-point formulas. We will need a more general concept of definability. Consider the following definition:

Definition 1 Let σ be a relational vocabulary, let C be a class of structures for σ , and for every $\mathfrak{A} \in C$ let $Q(\mathfrak{A})$ be a subset of A^r for some $r \in \omega$. We say that $Q = \{(\mathfrak{A}, Q(\mathfrak{A})) : \mathfrak{A} \in C\}$ is an r -ary query on C if and only if for every $\mathfrak{A}, \mathfrak{B} \in C$ such that $\mathfrak{A} \cong \mathfrak{B}$ we have that $(\mathfrak{A}, Q(\mathfrak{A})) \cong (\mathfrak{B}, Q(\mathfrak{B}))$. If $r = 0$, we say that Q is a Boolean query.

Boolean queries are identified with subsets of C in the obvious way; namely, if Q is a Boolean query on C , we write $\mathfrak{A} \in Q$ or $\mathfrak{A} \notin Q$ to mean that $Q(\mathfrak{A}) = \{()\}$ or $Q(\mathfrak{A}) = \emptyset$ respectively. Sometimes we say that a query is a *uniform relation*. Essentially, an r -ary query is a map from structures to r -ary relations on those structures, that is closed under isomorphisms. This closure property is required for queries to be definable by logics in the following sense:

Definition 2 Let σ be a relational vocabulary, let C be a class of structures for σ , and let Q be an r -ary query on C . We say that Q is first-order definable on C if and only if

there exists a first-order formula for σ , say φ , such that for every $\mathfrak{A} \in C$ we have that $Q(\mathfrak{A}) = \varphi^{\mathfrak{A}}$.

We may extend the definition of definability of queries to second-order logic, and fixed-point logics in the obvious way. For a class of structures C , we define the notations $\text{FO}[C]$, $\text{SO}[C]$, $\Sigma_1^1[C]$, $\text{LFP}[C]$, and $\text{PFP}[C]$, to denote the classes of queries that are definable on C in first-order logic, second-order logic, existential second-order logic, least fixed-point logic, and partial fixed-point logic respectively. If a query is definable, we may say that it is uniformly definable to emphasize the fact that queries are collections of relations.

Example 9 Let $\sigma = (E)$ be the vocabulary of graphs, and let C be the class of all directed graphs. The least fixed-point sentence ϕ of Example 2.2.3, defines on C the binary query that maps each directed graph to the set of pairs of nodes that are connected by a path. Similarly, the existential second-order formula Φ of Example 2.2.2, defines on C the Boolean query (zero-ary query) that consists of the class of all 3-colorable directed graphs. We say that the class of 3-colorable directed graphs is definable on the class of all directed graphs by an existential second-order sentence.

2.3 Models of computation

2.3.1 Turing machines

Our model of computation is the *multi-tape Turing machine*. We will adopt the more general model of *alternating Turing machine* as defined by Chandra, Kozen and Stockmeyer

[CKS81] with the additional feature of *random access to the input* as discussed by Buss [Bus87], Ruzzo [Ruz88], or Barrington, Immerman and Straubing [BIS90]. The model is allowed to have direct access to the input by means of an additional *address tape*. This will allow us to restrict our Turing machines to run within sub-linear time which would not make sense for standard Turing machines because they would not even have time to scan the whole input.

Definition 3 *A k -tape alternating Turing machine with random access input is a seven-tuple $M = (Q, \Sigma, \Gamma, \delta, q_0, g, h)$ where $(Q, \Sigma, \Gamma, \delta, q_0, g)$ is a standard $(k+1)$ -tape alternating Turing machine, and $h \subseteq Q$ is called the set of query states. The extra tape is called the address tape. On this tape, only symbols from $\{0, 1\}$ and the blank symbol are allowed.*

We will choose to call standard alternating Turing machines as *alternating Turing machines with sequential input*. Define a *configuration* as Chandra, Kozen, and Stockmeyer [CKS81]; a configuration is called a *query configuration* if the internal state is a query state. The operation of random access alternating Turing machines is very similar to the ones with sequential input except for two details: (i) the successor relation between configurations, $\alpha \vdash \beta$, where α is not a query configuration, is independent of the cell scanned by the input tape; and (ii) the successor relation between configurations, $\alpha \vdash \beta$, where α is a query configuration, assumes that the head of the input tape is placed at the cell pointed by the number written in binary in the address tape. In other words, query states are allowed to look at the input through their address tape as opposed to non-query states whose transitions are independent of the input. Buss [Bus87] defines the model

so that each state is a query state; the two variants are clearly equivalent. Alternating Turing machines without universal states are just *non-deterministic Turing machines*. If the transition relation is a function, the machine is a *deterministic Turing machine*.

Example 10 This example is not only useful to illustrate the operation of random access machines, but it will also be very operative in the rest of the text. This result is attributed to Dowd by Buss [Bus87].

Lemma 1 *There exists a deterministic Turing machine with random access input, that on input w of length n , halts in exactly $4 \log n$ steps with the binary representation of n written on the address tape.*

Proof: First the general idea. The machine determines the last position occupied by w . To do that, it queries the symbols at positions $2^0, 2^1, \dots, 2^i$ until the blank symbol is found. Then it performs a binary search between positions 2^{i-1} and $2^i - 1$. The technicalities next. Observe that addresses $2^0, 2^1, \dots, 2^i$ can be written on the address tape by adding 0's to an initial 1. Observe too that the binary search between 2^{i-1} and $2^i - 1$ consists in filling the zeros in 10^{i-2} with zeros or ones, from left to right, and according to a simple rule: zero if the symbol at the current address with a one is blank, and one otherwise. Thus, the running time is $\log n$ steps for the first phase, $\log n$ steps to let the head return to the first zero in 10^{i-2} , and $2 \log n$ steps to complete the binary search; overall, $4 \log n$ steps. \square

In the literature, random access machines are used when the time bounds are sub-linear, such as $\log n$ or $\log^2 n$. Observe that if the time bounds are under $\log n$, random access machines are not useful just as sequential input machines are not useful for sub-linear time bounds. In contrast, sequential input machines are used when the time bounds are at least linear. In fact, it will not make any difference to use random access machines in this case too, provided the bounds are well-behaved enough. In Appendix A, we show that the two models of alternating Turing machines can simulate each other while preserving time bounds and number of alternations up to multiplicative constants. Similar techniques are introduced by Buss [Bus87] and Barrington, Immerman, and Straubing [BIS90]. Nevertheless, previous results are either restricted to particular time and alternation bounds, or show equivalence with respect to time but not alternations.

2.3.2 Complexity classes

Our complexity classes are defined as follows. Let $t(n)$, $s(n)$ and $a(n)$ be functions from ω to ω . We say that a Turing machine *runs* in time $t(n)$ if on inputs of length n it halts within $t(n)$ steps of computation. A similar definition is made for space and alternating bounds. We denote by $\mathbf{DTIME}(t(n))$ to the class of languages over finite alphabets accepted by deterministic Turing machines running in time $t(n)$; we denote by $\mathbf{NTIME}(t(n))$ to the class of languages over finite alphabets accepted by non-deterministic Turing machines running in time $t(n)$; we denote by $\mathbf{DSPACE}(s(n))$ to the class of languages over finite alphabets accepted by deterministic Turing machines running in space $s(n)$; we denote by $\mathbf{ATIME}(t(n), a(n))$ to the class of lan-

guages over finite alphabets accepted by alternating Turing machines running in time $t(n)$ and $a(n)$ alternations. We define $\mathbf{DLOGTIME} = \bigcup_{c>0} \mathbf{DTIME}(c \log n)$, $\mathbf{P} = \bigcup_{c>0} \mathbf{DTIME}(n^c)$, $\mathbf{E} = \bigcup_{c>0} \mathbf{DTIME}(2^{cn})$, $\mathbf{EXP} = \bigcup_{c>0} \mathbf{DTIME}(2^{n^c})$, $\mathbf{PSPACE} = \bigcup_{c>0} \mathbf{DSpace}(n^c)$, $\mathbf{NP} = \bigcup_{c>0} \mathbf{NTIME}(n^c)$, $\mathbf{LH} = \bigcup_{c>0} \mathbf{ATIME}(c \log n, c)$, $\mathbf{LINH} = \bigcup_{c>0} \mathbf{ATIME}(cn, c)$, $\mathbf{PH} = \bigcup_{c>0} \mathbf{ATIME}(n^c, c)$.

2.3.3 Uniform circuits

Formally, an *unbounded fan-in Boolean circuit* is a 6-tuple $C = (V, E, I, N, A, O)$ where (V, E) is a finite directed acyclic graph and the 4-tuple (I, N, A, O) is a partition of V ; that is, I, N, A , and O are disjoint subsets of V such that $V = I \cup N \cup A \cup O$. The elements of V are called *gates*. The elements of I are called *input gates*, the elements of N are called *negation gates*, the elements of A are called *AND gates*, and the elements of O are called *OR gates*. All gates in I have in-degree zero, all gates in N have in-degree one, exactly one gate, the *output*, has out-degree zero, and any other degree is unrestricted. The *depth* of a circuit is the length of the longest path of the graph (V, E) . The *size* of a circuit is the number of gates. The *dimension* of a circuit is the number of input gates. A Boolean circuit C of dimension n defines a Boolean function $f_C : \{0, 1\}^n \rightarrow \{0, 1\}$ in the obvious way. Let $F = (C_1, C_2, \dots)$ be a family of circuits of dimensions $1, 2, \dots$. We say that F *accepts* the language $L \subseteq \{0, 1\}^*$ if and only if $L \cap \{0, 1\}^n = \{w \in \{0, 1\}^n : f_{C_n}(w) = 1\}$ for every $n \geq 1$.

We define the complexity class \mathbf{AC}^0 as follows: a language $L \subseteq \{0, 1\}^*$ is in \mathbf{AC}^0 if and only if there exist a constant c and a family of circuits $F = (C_1, C_2, \dots)$ that accepts

L , and such that the depth of C_n is bounded by c , and the size of C_n is bounded by n^c for every $n \geq 1$. It is easy to show that \mathbf{AC}^0 contains non-recursive languages. In order to compare \mathbf{AC}^0 with standard complexity classes, we impose a notion of uniformity on the circuit families.

Let $C = (V, E, I, N, A, O)$ be a Boolean circuit of size m and dimension n . As with finite structures, we may assume that $V = \{0, \dots, m-1\}$ (in fact, a Boolean circuit is a finite structure for the vocabulary $\sigma = (E, I, N, A, O)$ where E is a binary relation symbol, and I, N, A, O are unary relation symbols). The following definitions are borrowed from Barrington, Immerman, and Straubing [BIS90]. Recall the definition of $b_m(n)$ from Section 2.1. The *direct connection language of C* is a language over the 7-symbol alphabet $\{0, 1, \mathbf{I}, \mathbf{N}, \mathbf{A}, \mathbf{O}, \#\}$, denoted $\text{DCL}(C)$, and defined as follows: $\text{DCL}(C)$ is the set of words of the form $G\#b_m(a)\#b_m(b)\#0^n$ where $(a, b) \in E$, $G \in \{\mathbf{I}, \mathbf{N}, \mathbf{A}, \mathbf{O}\}$, and $a \in G$; here, $a \in \mathbf{A}$ is to be interpreted as $a \in A$ and similarly for I, N, O . In other words, the direct connection language of C is its piecewise description. Given a family of circuits $F = (C_1, C_2, \dots)$, the direct connection language of F , denoted $\text{DCL}(F)$, is just $\bigcup_{n \geq 1} \text{DCL}(C_i)$.

Let \mathcal{C} be a complexity class. We define the complexity class \mathcal{C} -uniform \mathbf{AC}^0 as the class of languages $L \subseteq \{0, 1\}^*$ that belong to \mathbf{AC}^0 via a family of circuits F whose direct connection language is in \mathcal{C} ; that is, $\text{DCL}(F) \in \mathcal{C}$. This determines the complexity classes that will be of our interest: **DLOGTIME**-uniform \mathbf{AC}^0 and **P**-uniform \mathbf{AC}^0 .

Chapter 3

Descriptive Complexity

3.1 Encodings and computability of queries

Finite structures for a finite relational vocabulary $\sigma = (R_1, \dots, R_s)$ are encoded as words over the alphabet $\{0, 1, \#\}$ according to the following convention. Let $\mathfrak{A} = (A, R_1^{\mathfrak{A}}, \dots, R_s^{\mathfrak{A}})$ be a finite structure for σ . Recall that we are assuming that universes are ordinals; so let $A = n \in \omega$. To each relation $R_i^{\mathfrak{A}}$ of arity $r = \text{ar}(R_i)$, we associate a word $\chi(R_i^{\mathfrak{A}}) \in \{0, 1\}^{n^r}$ that is called its *characteristic sequence*, and that is defined as follows. We set $\chi(R_i^{\mathfrak{A}}) = a_0 a_1 \dots a_{n^r-1}$ where $a_m \in \{0, 1\}$, and $a_m = 1$ if and only if $(m_{r-1}, \dots, m_0) \in R_i^{\mathfrak{A}}$ where (m_{r-1}, \dots, m_0) is the n -ary representation of m ; that is, the unique sequence of r natural numbers smaller than n such that $m = \sum_{j=0}^{r-1} m_j \cdot n^j$. Then, the encoding of \mathfrak{A} is just

$$\langle \mathfrak{A} \rangle = 1^n \# \chi(R_1^{\mathfrak{A}}) \# \chi(R_2^{\mathfrak{A}}) \# \dots \# \chi(R_s^{\mathfrak{A}})$$

We extend the encoding of structures to include *individuals* as follows. Let $(a_1, \dots, a_k) \in A^k$ be a sequence of individuals. We define the encoding of $(\mathfrak{A}, a_1, \dots, a_k)$ as

$$\langle \mathfrak{A}, a_1, \dots, a_k \rangle = \langle \mathfrak{A} \rangle \# b_n(a_1) \# b_n(a_2) \# \dots \# b_n(a_k)$$

where $b_n(m) \in \{0, 1\}^*$, with $n \geq m$, is the unique word of length $\log n$ that encodes m in binary with leading zeros if necessary; see section 2.1. This encoding of finite structures, essentially abstract entities, into words of a finite alphabet, links logic to computability in a precise way. The definition of computability of a query will complete this link.

Definition 4 *Let C be a class of finite structures, let Q be an r -ary query on C , and let $t: \omega \rightarrow \omega$ be a function. We say that Q is computable on C in deterministic time $t(n)$ if and only if there exists a deterministic Turing machine M such that, for every $\mathfrak{A} \in C$ and every $(a_1, \dots, a_r) \in A^r$, whenever M is presented with input $\langle \mathfrak{A}, a_1, \dots, a_r \rangle$, it halts within $t(|\langle \mathfrak{A}, a_1, \dots, a_r \rangle|)$ steps of computation and accepts if and only if $(a_1, \dots, a_r) \in Q(\mathfrak{A})$.*

For Boolean queries, the machine is presented with the input \mathfrak{A} , only. Obviously, the same definition can be made for non-deterministic and alternating computation.

3.2 The complexity of logical definitions

We have agreed to link definability and computability through appropriate encodings of abstract structures. The question remains whether this link is natural enough to match familiar concepts from computability. The purpose of this section and the next is to show that, indeed, this is the case.

3.2.1 First-order logic in LH

It is known that first-order logic is particularly weak for uniform definability on classes of finite structures. In fact, every FO-definable query is computable in logarithmic space. The next theorem shows that they are even easier to compute; they are computable in the Logarithmic-Time Hierarchy introduced by Buss [Bus87].

Theorem 1 *Let σ be a relational vocabulary, let C be a class of finite structures for σ , and let Q be a query on C . If Q is FO-definable on C , then Q is computable on C in alternating logarithmic-time with a constant number of alternations.*

Proof: Assume that $\sigma = (R_1, \dots, R_s)$. Suppose that Q is defined by the first-order formula $\varphi(x_{k-1}, \dots, x_0)$. We show, by induction on the construction of φ , that Q is computable on C in time $c \log n$ and c alternations for some constant c . Suppose that φ is atomic of the form $R_i(x_{i_{k-1}}, \dots, x_{i_0})$ for some $i \in \{1, \dots, s\}$. The alternating Turing machine on input $\langle \mathfrak{A}, a_{k-1}, \dots, a_0 \rangle$ acts as follows. Essentially, the machine computes $b = \sum_{j=0}^{k-1} a_{i_j} \cdot n^j$, and determines if the b -th symbol of $\chi(R_i^{\mathfrak{A}})$ from the left is one. The machine first determines the length of its input according to Lemma 1, say n . Then it determines the cardinality of \mathfrak{A} , say m , by existentially guessing the position of the first occurrence of $\#$, and then universally branching over all smaller positions to check that they do not hold a $\#$. After that, it starts a loop in which a counter j is initially set to 0 and is incremented by one after each iteration, until k is reached. A variable b is initially set to 0. In each iteration of the loop, a new loop is started on which a new variable j' , initially set to 0, is incremented by one after each iteration, until k is reached too. For this second loop, a variable q is

initially set to n ; variable q is intended to hold the position of the j' -th occurrence $\#$ from the right at the end of each iteration. In each iteration of the second loop, the machine existentially guesses the position of the $(j' + 1)$ -th occurrence of $\#$ from the right of the input, say p , and universally branches over all positions between p and q to check that they do not contain a $\#$. After the position is guessed and verified, it checks if $j' = i_j$. If so, it updates b to $b \cdot n + a_{j'}$. Variable q is updated to p unconditionally. At the end of the two loops, b contains $\sum_{j=0}^{k-1} a_{i_j} \cdot n^j$. It remains to see if $(a_{i_{k-1}}, \dots, a_{i_0}) \in R_i^{\mathfrak{A}}$. To do that, the machine finds the position of the i -th occurrence of $\#$ from the left, say p , computes $p + b$, and accepts if and only if the $(p + b)$ -th symbol of the input is 1. For the running time reason as follows. To compute n and m , the machine spends $4 \log n$ steps and $2 \log n$ steps respectively. The number of alternations is 2 by now. The body of the inner loop takes at most $d \log n$ time and d alternations for some constant d , according to Section B.1 (a comparison, a multiplication, and an addition of logarithmic-size numbers); since this is repeated k^2 times, the time spent in them is at most $dk^2 \log n$ and the number of alternations is at most dk^2 . The last phase of the computation takes about $(2k + d) \log n$ time and $2k + d$ alternations.

Suppose next that the claim holds for ψ and that $\varphi = \neg\psi$. The alternating Turing machine for φ is obtained from that of ψ by exchanging universal and existential states, as well as accepting and rejecting states. For $\varphi = \psi \wedge \theta$, the machine is obtained from the machines for ψ and θ by running them in parallel after a universal branch. More precisely, the machine for φ universally starts two branches; on the left branch, the machine for

ψ is simulated, and on the right branch, the machine for θ is simulated. Suppose next that $\varphi = (\exists x)\psi(x_{k-1}, \dots, x_0, x)$. Let Q' be the $(k+1)$ -ary query on C defined by ψ and assume that the claim holds for it. The machine for φ on input $\langle \mathfrak{A}, a_{k-1}, \dots, a_0 \rangle$ first determines the length of the input, say n , then determines the cardinality of \mathfrak{A} , say m , and then existentially guesses $\log m$ bits, say $v \in \{0, 1\}^{\log m}$. The machine is ready to start the machine for ψ and whenever a position beyond n is queried, the machine answers pretending that the input is $\langle \mathfrak{A}, a_{k-1}, \dots, a_0 \rangle \# v$. It should be clear that the machine defines Q . According to the results in Appendix A, we may assume that the machine for ψ only queries the input once in each path; it then follows that the simulation can be done in $c \log n$ time and c alternations for some constant c , using the trick in Lemma 10 to access v sequentially when needed. This completes the induction and the proof of the theorem. \square

3.2.2 Least fixed-point logic in P

As LFP-definable queries on classes of finite structures concerns, the next theorem shows that they are computable in deterministic polynomial-time.

Theorem 2 *Let σ be a relational vocabulary, let C be a class of finite structures for σ , and let Q be a query on C . If Q is LFP-definable on C , then Q is computable on C in deterministic polynomial-time.*

Proof: Suppose that $\sigma = (R_1, \dots, R_s)$ and let $\varphi(x_{k-1}, \dots, x_0)$ be a LFP-formula that defines Q on C . The proof is by induction on the construction of φ . If φ is a first-

order formula, the claim follows immediately from the previous theorem; this is because an alternating Turing machine running in logarithmic-time and a constant number of alternations can be simulated in deterministic polynomial-time. So let us assume that $\varphi = (\text{LFP}_{\bar{z}, X} \psi(\bar{z}, \bar{x}, X))(x_{k-1}, \dots, x_0)$ where ψ is a LFP-formula of the vocabulary $\sigma' = (\sigma, X)$ that is positive in X ; here $\bar{z} = (z_{k-1}, \dots, z_0)$, $\bar{x} = (x_{k-1}, \dots, x_0)$, and X is a k -ary relation variable. Let us assume by induction hypothesis that the claim holds for ψ on the class of finite structures for σ' of the form $\mathfrak{A}' = (\mathfrak{A}, X^{\mathfrak{A}'})$ where $\mathfrak{A} \in \mathcal{C}$ and $X^{\mathfrak{A}'} \subseteq A^k$. The Turing machine for φ acts as follows. On input $\langle \mathfrak{A}, a_{k-1}, \dots, a_0 \rangle$, the machine first determines the length of its input, say n , then it determines the cardinality of \mathfrak{A} , say m , and then it writes the word 0^{m^k} on a separate tape. Observe that 0^{m^k} is the characteristic sequence for the empty relation \emptyset ; the tape will always contain the characteristic sequence of a relation $X \subseteq A^k$ that will be updated to compute the least fixed-point of ψ on \mathfrak{A} . The machine starts k nested loops to cycle through all tuples (b_{k-1}, \dots, b_0) where $b_i \in \{0, \dots, m-1\}$. For each such tuple, the machine starts a simulation of the polynomial-time machine for ψ on input

$$\langle \mathfrak{A}, X, b_{k-1}, \dots, b_0, a_{k-1}, \dots, a_0 \rangle$$

If the machine accepts, it updates the tape that contains the characteristic sequence of X , by switching to one its bit at position $\sum_{j=0}^{k-1} b_j \cdot n^j$. Whenever all k nested loops have been completed, the machine accepts or rejects according to whether the bit at position $\sum_{j=0}^{k-1} a_j \cdot n^j$ of the characteristic sequence of X is one or zero respectively. It should be clear that the machine computes Q . For the running time, observe that each loop spends

time $(n + m^k + 1 + k \log m + k)^c$ for some constant c by induction hypothesis, and this is repeated m^k times; overall, the running time is polynomial. When $\varphi = \neg\psi$, $\varphi = \psi \wedge \theta$ or $\varphi = (\exists x)(\psi)$, use the induction hypothesis and techniques as in Theorem 1. \square

3.2.3 Existential second-order logic in NP

We close this section showing that queries defined by existential second-order logic on classes of finite structures are computable in non-deterministic polynomial-time.

Theorem 3 *Let σ be a relational vocabulary, let C be a class of finite structures for σ , and let Q be a query on C . If Q is Σ_1^1 -definable on C , then Q is computable on C in non-deterministic polynomial-time.*

Proof: Suppose that $\sigma = (R_1, \dots, R_s)$ and let $\varphi(x_{k-1}, \dots, x_0)$ be a Σ_1^1 -formula that defines Q on C . Assume that $\varphi = (\exists X_1) \cdots (\exists X_t) \psi(X_1, \dots, X_t, x_{k-1}, \dots, x_0)$ where ψ is a first-order formula of the vocabulary $(R_1, \dots, R_s, X_1, \dots, X_t)$. Let r_i be the arity of X_i . Let Q' be the query defined by ψ on the class of structures of the form $\mathfrak{A}' = (\mathfrak{A}, X_1^{\mathfrak{A}'}, \dots, X_t^{\mathfrak{A}'})$ with $\mathfrak{A} \in C$ and $X_i^{\mathfrak{A}'} \subseteq A^{r_i}$. Clearly, by Theorem 2, query Q' is polynomial-time computable on this class. The non-deterministic Turing machine on input $\langle \mathfrak{A}, a_{k-1}, \dots, a_0 \rangle$, first determines the cardinality of \mathfrak{A} , say m , and then guesses a word $w_i \in \{0, 1\}^{m^{r_i}}$ for each $i \in \{1, \dots, t\}$. Each word w_i is the characteristic sequence of a relation $X_i \subseteq A^{r_i}$. The machine then starts a deterministic simulation of the machine witnessing that Q' is polynomial-time computable on input $\langle \mathfrak{A}, X_1, \dots, X_t, a_{k-1}, \dots, a_0 \rangle$. It should be clear that the machine computes Q and that it satisfies the required computational

limitations. For the running time, observe that $|\langle \mathfrak{A}, X_1, \dots, X_t, a_{k-1}, \dots, a_0 \rangle|$ is bounded by a polynomial in $|\langle \mathfrak{A}, a_{k-1}, \dots, a_0 \rangle|$. \square

3.3 Logic captures computational complexity

In this section we are interested in the following question: to what extent do the logics considered in the previous section match the complexity class they are included in?

3.3.1 Existential second-order logic captures NP

The following result is known as Fagin's Theorem. Among other things, it says that *logic exactly captures natural complexity classes*. It also provides with a neat characterization of an important open problem in Complexity Theory: whether **NP** is closed under complementation.

Theorem 4 *Let σ be a relational vocabulary, let C be a class of finite structures for σ , and let Q be a query on C . If Q is computable on C in non-deterministic polynomial-time, then Q is Σ_1^1 -definable on C .*

Proof: Suppose that $\sigma = (R_1, \dots, R_s)$ and assume that Q is computable on C by a non-deterministic Turing machine, say M , running in polynomial-time.

Assume that M satisfies the following properties: (i) on input $\langle \mathfrak{A}, a_{r-1}, \dots, a_0 \rangle$, the machine halts after exactly $|A|^c$ steps for some constant c , and (ii) each non-halting configuration has exactly two successors. It should be clear that M can be standardized to satisfy these properties with only a polynomial blow-up in the running time. Let

$\Gamma = \{s_1, \dots, s_{|\Gamma|}\}$ be the tape alphabet of M , and let $S = \{q_1, \dots, q_{|S|}\}$ be the set of internal states of M . The Σ_1^1 -sentence will take the form

$$(\exists N)(\exists E)(\exists \leq)(\exists +)(\exists \times)\psi(\leq, +, \times, E, N)$$

where \leq is a binary relation symbol, $+$ and \times are ternary relation symbols, E is a $(3c+3)$ -ary relation symbol, and N is a c -ary relation symbol. Symbols \leq , $+$, and \times will be forced to be a linear order on the universe and the corresponding arithmetical predicates; this is done by including their axioms in the first-order part ψ . See Lemma 16. The linear order on the universe induces a linear order on the set of c -tuples of the universe. For clarity of exposition, we identify a c -tuple $\bar{t} = (t_1, \dots, t_c)$ with its position in this linear order. The relation $E(\bar{t}, \bar{p}, \bar{p}', i, j, h)$ will be interpreted as: “at time \bar{t} , the j -th tape contains symbol s_i at position \bar{p} , its head pointing to position \bar{p}' , and the internal state of the machine being q_h .” For this predicate to be well-defined, we will assume that our structures have at least $\max\{|S|, |\Gamma|, k\}$ elements; according to Section B.2, this is no loss of generality. We will also have a c -ary predicate $N(\bar{t})$ meaning “at time \bar{t} , the left non-deterministic branch is taken”. It should be clear how to force the interpretations of E and N to represent a valid computation of M on input $\langle \mathfrak{A}, a_{r-1}, \dots, a_0 \rangle$. For the sake of example, the following clause would force steps to write proper symbols under the heads of the working tapes: suppose that on state q_h and with the head of tape j pointing to symbol s_{i_j} , the right non-deterministic choice of M leads to state $q_{h'}$, writes symbol $s_{i'_j}$ under the j -th head,

and moves it in direction d_j ; the clause is as follows:

$$(\forall \bar{t})(\forall \bar{p}_1) \cdots (\forall \bar{p}_k) \left(\bigwedge_{j=1}^k E(\bar{t}, \bar{p}_j, \bar{p}_j, i_j, j, h) \wedge \neg N(\bar{t}) \rightarrow \bigwedge_{j=1}^k E(\bar{t} + 1, \bar{p}_j, \bar{p}_j + d_j, i'_j, j, h') \right)$$

Here, the term $\bar{t} + 1$ is a short-hand for an existentially quantified variable forced to hold the successor of \bar{t} in the linear order over c -tuples induced by \leq . Terms $\bar{p} + d_j$ are to be read in a similar way. The arithmetical predicates $+$ and \times will be used to decode the input $\langle \mathfrak{A}, a_{r-1}, \dots, a_0 \rangle$. In particular, individual bits of a_i are decoded with the help of the predicate $\text{BIT}(x, y)$ saying that “the x -th bit of y is one”; this predicate is first-order definable from $+$ and \times according to Lemma 17. Finally, the first-order formula φ includes a clause saying that the computation path encoded this way eventually reaches an accepting configuration. \square

3.3.2 Least fixed-point logic captures P

The obvious question at this point is whether least fixed-point logic captures all polynomial-time computable queries. Although we can answer the question in the negative for arbitrary classes of finite structures (see [CH82]), it turns out that the answer is positive for classes of finite *ordered* structures. In the following, let \leq be a binary relation symbol.

Definition 5 *Let σ be a relational vocabulary that does not contain \leq , and let C be a class of structures for (σ, \leq) . We say that C is a class of ordered structures for (σ, \leq) if and only if for every $\mathfrak{A} \in C$ we have that $\leq^{\mathfrak{A}}$ is the standard linear order on the universe of \mathfrak{A} (an ordinal).*

The following result is known as the Immerman-Vardi Theorem. It also provides a link between logic and the question on whether \mathbf{P} equals \mathbf{NP} .

Theorem 5 *Let σ be a relational vocabulary, let C be a class of finite ordered structures for (σ, \leq) , and let Q be a query on C . If Q is computable on C in deterministic polynomial-time, then Q is LFP-definable on C .*

Proof: Suppose that Q is computed by a k -tape deterministic Turing machine, say M , running in polynomial-time. As in the proof of Fagin's Theorem, assume that on input $\langle \mathfrak{A}, a_{r-1}, \dots, a_0 \rangle$, the machine halts after exactly $|A|^c$ steps for some constant c . Let $\Gamma = \{s_1, \dots, s_{|\Gamma|}\}$ be the tape alphabet of M , and let $S = \{q_1, \dots, q_{|S|}\}$ be the set of internal states of M . A least fixed-point formula $\text{LFP}_{\bar{z}, X} \phi(\bar{z}, X)$ will define a $(3c + 3)$ -ary predicate $E(\bar{t}, \bar{p}, \bar{p}', i, j, h)$ whose interpretation will be that "at time \bar{t} , the j -th tape contains symbol s_i at position \bar{p} whereas its head is at position \bar{p}' and the internal state is q_h ". Observe that E is exactly the same predicate as in Fagin's theorem. In this case, predicate N will not be needed because the machine is deterministic. For predicate E to be well-defined, we will need to assume that all structures have at least $\max\{|S|, |\Gamma|, k\}$ elements; this is again no loss of generality according to Section B.2. Our least fixed-point formula will define E inductively on its first argument \bar{t} ; since the machine is deterministic, each configuration has a unique predecessor so that such a definition is possible. As an example, suppose that on state q_h and with the head of the j -th tape pointing to symbol s_{i_j} , the machine moves to state $q_{h'}$, writes symbol $s_{i'}$ under the head of tape j' , and moves it in direction d . Then, the inductive definition of $E(\bar{t}, \bar{p}, \bar{p}', i', j', h')$ would consist of a

disjunction of cases of the form

$$\dots \vee (\exists \bar{p}_1) \dots (\exists \bar{p}_k) (\bar{p}_{j'} = \bar{p} \wedge \bar{p}' = \bar{p} + d \wedge \bigwedge_{j=1}^k E(\bar{t} - 1, \bar{p}_j, \bar{p}_j, i_j, j, h)) \vee \dots$$

Here, constants and terms would be existentially quantified and forced to their appropriate values with the help of \leq . As in the proof of Fagin's theorem, the input $\langle \mathfrak{A}, a_{r-1}, \dots, a_0 \rangle$ would be decoded with the help of arithmetical predicates $+$ and \times which are LFP-definable in the presence of the order, as shown in Lemma 15. Finally, the LFP-sentence will say that the computation path defined this way, eventually reaches a configuration with an accepting state. \square

3.3.3 First-order logic captures LH

We turn next to first-order logic. Just as least fixed-point logic cannot capture all polynomial-time queries on arbitrary classes of finite structures, first-order logic is weak to capture all queries computable in alternating logarithmic-time and a constant number of alternations. More strongly, even on classes of finite ordered structures, first-order logic is not expressive enough to capture all those queries. For let \mathcal{O} be the class of all finite standard linear orders; that is, \mathcal{O} is the class of all finite structures of the form $\mathfrak{A} = (A, \leq^{\mathfrak{A}})$ where $\leq^{\mathfrak{A}}$ is the standard linear order on A , and it is its unique relation. It is a well-known result that the Boolean query Q on \mathcal{O} such that $\mathfrak{A} \in Q$ if and only if the cardinality of the universe of \mathfrak{A} is even, is not FO-definable on \mathcal{O} (a precise proof of this fact is provided in the next chapter). In contrast, this query is computable in alternating logarithmic-time since a machine can easily determine the length of its input using Lemma

1, and check if the last bit of its binary representation is one (observe that $\mathfrak{A} \in Q$ if and only if $|\langle \mathfrak{A} \rangle| = |A| + 1 + |A|^2$ is odd).

One can ask whether *built-in* predicates more powerful than the standard linear order can serve first-order logic to capture all queries that are computable in alternating logarithmic-time and a constant number of alternations. From the proofs of the previous two theorems, one may guess that arithmetical predicates could do this job. It turns out that this intuition is correct.

Definition 6 *Let σ be a relational vocabulary that does not contain \leq , $+$, and \times , and let C be a class of structures for $(\sigma, \leq, +, \times)$. We say that C is a class of arithmetical structures for $(\sigma, \leq, +, \times)$ if and only if for every $\mathfrak{A} \in C$ we have that $\leq^{\mathfrak{A}}$, $+^{\mathfrak{A}}$, and $\times^{\mathfrak{A}}$ are the standard arithmetical predicates on the universe of \mathfrak{A} (an ordinal).*

The following result is due to Barrington, Immerman, and Straubing [BIS90]. See the introduction for a discussion of the differences.

Theorem 6 *Let σ be a relational vocabulary, let C be a class of finite arithmetical structures for $(\sigma, \leq, +, \times)$, and let Q be a query on C . If Q is computable on C in alternating logarithmic-time and a constant number of alternations, then Q is FO-definable on C .*

Proof: Suppose that Q is computed by a k -tape alternating Turing machine with random access, say M , running it time $c \log n$ and d alternations for some constants c and d . We can assume, without loss of generality, that the machine satisfies the following properties: (i) each computation path of M halts after exactly $c \log n$ steps of computation, (ii) each

non-halting configuration has exactly two successors, (iii) each tape head moves at each step, and (iv) alternations are grouped into blocks of $b(n) = (c \log n)/d$ steps starting with an existential block; the constants c and d will be chosen so that d exactly divides c . The machine on input of length n executes $b(n)$ existential steps, followed by $b(n)$ universal steps, followed by $b(n)$ existential steps, and so on. For the first assumption, let the machine determine the length of its input n in $4 \log n$ steps according to Lemma 1, and then clock the machine to let it shut off its computation after exactly $c - 4$ scans of n had been performed.

As in the proof of Fagin's theorem, we will use predicates to define a computation of the machine. However, the encoding of these predicates will need to fit in $e \log n$ bits for some constant e , so that they can be quantified in a first-order way. Let $\Gamma = \{s_1, \dots, s_{|\Gamma|}\}$ be the tape alphabet of M , and let $S = \{q_1, \dots, q_{|S|}\}$ be the set of internal states of M . We will have a 5-ary predicate $E(t, i, j, d, h)$ meaning that "at time t , symbol s_i has been written on tape j , its head has moved following direction d , and the machine has entered state q_h ." Here, j is restricted to belong to $\{2, \dots, k + 1\}$ so that the contents of the input tape is not specified; it is accessed through the address tape numbered $k + 1$. On the other hand, $d = 0$ is interpreted as moving the head to the left, and $d = 1$ as moving it to the right. For this predicate to be well-defined, we will assume that our structures have at least $\max\{|S|, |\Gamma|, k + 1\}$ elements, and that $c \log n \leq n$; by Section B.2, this is no loss of generality. We will also have a unary predicate $N(t)$ meaning "at time t , the left (existential or universal) branch is taken". Observe that since M is time-bounded

by $c \log n$, the argument t need not go beyond this number. On the other hand, the arguments i, j, d, h are all bounded by the constant $m = \max\{|\Gamma|, k + 1, |S|\}$. It follows that the relation E can be encoded with $(c+1) \log n$ bits which fit in a first-order variables for some constant $a \geq c+1$. We choose a to be a multiple of d . Similarly, N can be encoded with a first-order variables. Our first-order formula will take the form:

$$(\exists n_{1,1}) \cdots (\exists n_{1,a/d}) (\forall n_{2,1}) \cdots (\forall n_{2,a/d}) \cdots (Q n_{d,1}) \cdots (Q n_{d,a/d}) (\exists e_{a-1}) \cdots (\exists e_0) \psi$$

Here Q is \exists or \forall depending on whether c is odd or even respectively. The sequence of variables $n_{i,j}$ encodes predicate N , while the sequence of variables e_i encodes predicate E . The first-order formula ψ will be similar to the first-order formula in the proof of Fagin's theorem. However, a few difficulties need to be fixed. First, each occurrence of $E(t, i, j, d, h)$ is replaced by the sub-formula

$$\text{BIT}_a(h + dm + jm^2 + im^3 + tm^4, e_{a-1}, \dots, e_0)$$

Here, $\text{BIT}_k(x, y_{k-1}, \dots, y_0)$ is the predicate expressing that the x -th bit of $\sum_{j=0}^{k-1} y_j n^j$ is one, and it is first-order definable from $+$ and \times as shown in Lemma 17. As usual, terms are intended to be existentially quantified variables defined to their appropriate values. A second difficulty comes from the fact that the contents of the work tapes are not available directly. Instead, we need a first-order formula $\theta(t, p, i, j)$ saying "at time t , the j -th tape contains symbol s_i in position p ". This formula can be constructed from $E(t, i, j, d, h)$: namely, $\theta(t, p, i, j)$ if and only if s_i is the symbol that was written last time the head scanned position p of tape j . It suffices then to define a first-order formula $\chi(t, p, j)$

saying “at time t , the head of the j -th tape is at position p .” To do that, one needs to check if $p = \sum_{t'=0}^t (2d_{t'} - 1)$ where $E(t', i_{t'}, j, d_{t'}, h_{t'})$ for some $i_{t'}, d_{t'}, h_{t'}$, and every $t' \leq t$. This can be done by extracting the $d_{t'}$'s from the e_i 's, putting them into a existentially quantified variables z_{a-1}, \dots, z_0 , and checking the aforementioned relation between p and the bits of the z_i 's; Lemma 17 assures that all this can be done in first-order logic from $+$ and \times . It remains to see how to access the input. Whenever a query state is reached, the contents of the address tape is existentially quantified, checked using $\theta(t, p, i, k + 1)$, and used to access the relations in $\langle \mathfrak{A}, a_{r-1}, \dots, a_0 \rangle$. Again, BIT, definable from $+$ and \times , serves us to decode the input. \square

Chapter 4

Background on the Ordered Conjecture

4.1 Restricted classes of finite structures

We have seen in Chapter 3 that logics capture natural complexity classes in a precise way. However, for complexity classes below **NP**, there is the technical need to restrict oneself to particular classes of finite structures such as the ordered or arithmetical structures. This raises the following question: given a logic \mathcal{L} and a complexity class \mathcal{C} , on which classes of finite structures does logic \mathcal{L} capture complexity class \mathcal{C} ?

Fagin's theorem answers the question when \mathcal{L} is existential second-order logic and \mathcal{C} is **NP**; namely, it says that Σ_1^1 captures **NP** on *every* class of finite structures. On the other hand, when \mathcal{L} is least fixed-point logic and \mathcal{C} is **P**, the answer to the above question

is not clear yet. In particular, a precise characterization of the classes of finite structures on which least fixed-point logic captures \mathbf{P} would be of theoretical and practical interest. Our small contribution to this question is that any such class has to be of asymptotic uniform probability zero. The proof of this result uses some concepts and methods that are not fully defined in this text. The reader is referred to the paper by Hella, Kolaitis, and Luosto [HKL94] for an account of all relevant material.

Theorem 7 *Let σ be a relational vocabulary and let C be a class of finite structures for σ . If every polynomial-time computable query on C is LFP-definable, then C has zero asymptotic uniform probability.*

Proof: Let $\mu(\cdot)$ denote the asymptotic uniform probability on the class of all finite structures for σ . We will make intensive use of the fact that if $\mu(A) \neq 0$ and $\mu(B) = 1$, then $\mu(A \cap B) \neq 0$. Here, $\mu(S) \neq 0$ means that either $\mu(S)$ does not exist, or it exists but it is greater than 0. Suppose that $\mu(C) \neq 0$. Then, the subclass of rigid structures of C has non-zero probability because $\mu(\text{RIG}) = 1$ (see [EF95, Proposition 3.3.11]); here RIG stands for the class of all finite rigid structures for σ . Let $D = C \cap \text{RIG}$, so that $\mu(D) \neq 0$. Consider the following query Q defined on C : for every $\mathfrak{A} \in C$, we let $Q(\mathfrak{A})$ be the set of all pairs $(a, b) \in A^2$ such that $a \prec b$ according to the partial order produced by the algorithm of Babai, Erdős, and Selkow [BES80, HKL94] to canonically label structures; the query is well defined because the result of the algorithm only depends on the isomorphism type of the input structure. Let F be the class of all finite structures \mathfrak{A} for σ such that $Q(\mathfrak{A})$ is a linear order on A . The principal property of that algorithm is that $\mu(F) = 1$. We show

that Q is not LFP-definable on C .

For suppose it were and let $E \subseteq D$ be the class of all structures $\mathfrak{A} \in D$ such that $Q(\mathfrak{A})$ is a linear order on A ; that is, $E = D \cap F$. It follows from the above that $\mu(E) \neq 0$ because $\mu(D) \neq 0$ and $\mu(F) = 1$. Since a linear order is LFP-definable on E , and all structures in E are rigid, it follows that E is a class of bounded rigidity (see [HKL94, Theorem 3.5]), and so $\mu(E) = 0$ (see [HKL94, just after Definition 3.4]); a contradiction. \square

It is clear that the converse of the previous theorem does not hold. Namely, there is a vocabulary σ and a class of finite structures C for σ , such that C has probability zero on the class of all finite structures for σ , but yet least fixed-point logic does not capture \mathbf{P} on it. For example, take $\sigma = (E)$ to be the vocabulary of a single binary relation, and let C be the class of all finite complete graphs; obviously, C has probability zero on the class of all finite structures for σ . On the other hand, least fixed-point logic inherits the 0-1 law on C from the 0-1 law on the class of all finite pure sets (structures for the empty vocabulary) (see [KV90]), and therefore, LFP cannot capture \mathbf{P} on C .

4.2 Statement of the ordered conjecture

In the previous section we asked about which classes of finite structures make least fixed-point logic capture \mathbf{P} . We could also pose the same question for first-order logic; namely, is there an infinite class of finite structures on which first-order logic captures \mathbf{P} ? According to the Immerman-Vardi Theorem, the answer to this question is negative only if the following conjecture stated by Kolaitis and Vardi in 1992 [KV92] holds.

Conjecture 2 *Let σ be a relational vocabulary. If C is an infinite class of finite ordered structures for (σ, \leq) , then $\text{FO}[C] < \text{LFP}[C]$.*

The Ordered Conjecture has received particular attention in recent years. As researchers have shown, any way of resolving the problem would have important consequences in complexity theory. Moreover, some special cases of it have been established true, while some others have been shown to be literally equivalent to long-standing open problems in complexity theory. The next sections of this chapter are devoted to overview those variations that hold true. We will consider the other cases mentioned above in the next chapter.

4.3 Restrictions and relaxations that hold true

4.3.1 Linear orders

Let us consider first the simplest case of the Ordered Conjecture: linear orders. Let $\sigma = ()$ be the empty vocabulary, and let \mathcal{O} be the class of all finite ordered structures for $(\sigma, \leq) = (\leq)$. Observe that any $\mathfrak{A} \in \mathcal{O}$ is of the form $\mathfrak{A} = (A, \leq^{\mathfrak{A}})$, where A is a finite ordinal, and $\leq^{\mathfrak{A}}$ is its standard linear order. We wish to show that the Ordered Conjecture holds on \mathcal{O} ; in other words, that there exists a query on \mathcal{O} that is least fixed-point definable, but not first-order definable. We will need some technical machinery.

Let $\sigma = (R_1, \dots, R_s)$ be a relational vocabulary, and let $\mathfrak{A}, \mathfrak{B}$ be two structures for σ . Let p be a partial function from A to B ; that is, p is a function from S to B for some $S \subseteq A$. We let $\text{dom}(p)$ and $\text{ran}(p)$ be the domain and the range of p ; that is, $\text{dom}(p) = S$,

and $\text{ran}(p) = \{p(a) : a \in S\}$. A *partial isomorphism* between \mathfrak{A} and \mathfrak{B} is a partial function from A to B that is injective in its domain, and such that for every $(a_1, \dots, a_k) \in \text{dom}(p)^k$ we have that $(a_1, \dots, a_k) \in R_i^{\mathfrak{A}}$ if and only if $(p(a_1), \dots, p(a_k)) \in R_i^{\mathfrak{B}}$. In the following, we identify a partial function p with its graph $\{(a, p(a)) : a \in \text{dom}(p)\}$.

Definition 7 Let σ be a relational vocabulary, and let \mathfrak{A} and \mathfrak{B} be two structures for σ . Let $a_1, \dots, a_k \in A$, let $b_1, \dots, b_k \in B$, and let $m \geq 0$. We say that $(\mathfrak{A}, a_1, \dots, a_k)$ and $(\mathfrak{B}, b_1, \dots, b_k)$ are *m-partially isomorphic*, denoted $(\mathfrak{A}, a_1, \dots, a_k) \cong_m (\mathfrak{B}, b_1, \dots, b_k)$, if and only if there exists a sequence of $m + 1$ sets of partial isomorphisms $I = (I_0, \dots, I_m)$ between \mathfrak{A} and \mathfrak{B} such that $\{(a_i, b_i) : i = 1, \dots, k\} \in I_0$, and for every $i \in \{0, \dots, m - 1\}$ and $p \in I_i$ it holds that

- (i) for every $a \in A$, there exists some $q \in I_{i+1}$ and $b \in B$ such that $\text{dom}(p) \subseteq \text{dom}(q)$,
 $a \in \text{dom}(q)$, and $q(a) = b$ (back property),
- (ii) for every $b \in B$, there exists some $q \in I_{i+1}$ and $a \in A$ such that $\text{dom}(p) \subseteq \text{dom}(q)$,
 $a \in \text{dom}(q)$, and $q(a) = b$ (forth property),

Let φ be a first-order formula for σ . The *quantifier rank* of φ , denoted $\text{qr}(\varphi)$, is defined inductively as follows: if φ is $x = y$, then $\text{qr}(\varphi) = 0$; if φ is $R_i(x_1, \dots, x_{r_i})$, then $\text{qr}(\varphi) = 0$; if φ is $\neg\psi$, then $\text{qr}(\varphi) = \text{qr}(\psi)$; if φ is $\psi \wedge \theta$, then $\text{qr}(\varphi) = \max\{\text{qr}(\psi), \text{qr}(\theta)\}$; if φ is $(\forall x)(\psi)$, then $\text{qr}(\varphi) = 1 + \text{qr}(\psi)$. The fundamental property about m -partial isomorphisms is Fraissé's Theorem.

Theorem 8 Let σ be a relational vocabulary, let $m \geq 0$, let $\mathfrak{A}, \mathfrak{B}$ be two structures for σ , let $a_1, \dots, a_k \in A$, and let $b_1, \dots, b_k \in B$. It holds that $(\mathfrak{A}, a_1, \dots, a_k) \cong_m (\mathfrak{B}, b_1, \dots, b_k)$ if

and only if for every first-order formula φ for σ of quantifier rank at most m and with free variables among x_1, \dots, x_k , we have that $\mathfrak{A} \models \varphi[a_1, \dots, a_k]$ if and only if $\mathfrak{B} \models \varphi[b_1, \dots, b_k]$.

Proof (half): We just show the direction of the theorem that is relevant to us; for the other direction, the reader is referred to Ebbinghaus and Flum [EF95]. Suppose that $(\mathfrak{A}, a_1, \dots, a_k) \cong_m (\mathfrak{B}, b_1, \dots, b_k)$; we show that a_1, \dots, a_k and b_1, \dots, b_k satisfy the same formulas of quantifier rank at most m in \mathfrak{A} and \mathfrak{B} respectively. The proof is by induction on m . For $m = 0$, the claim is clear from the definitions. Suppose that $m > 0$, and that the claim holds for every $m' < m$. Let $I = (I_0, \dots, I_m)$ be the sequence of sets of partial isomorphisms witnessing that $(\mathfrak{A}, a_1, \dots, a_k) \cong_m (\mathfrak{B}, b_1, \dots, b_k)$, and let φ be any formula of quantifier rank at most m and free variables among x_1, \dots, x_k . Suppose that $\mathfrak{A} \models \varphi[a_1, \dots, a_k]$. We show that $\mathfrak{B} \models \varphi[b_1, \dots, b_k]$ by induction on the construction of φ . Suppose first that φ is atomic of the form $x_i = x_j$. Then $a_i = a_j$ and so $b_i = b_j$ because $\{(a_i, b_i) : i = 1, \dots, k\} \in I_0$ is a well-defined partial isomorphism by assumption. If φ is of the form $R_i(x_{i_1}, \dots, x_{i_{r_i}})$ reason similarly. Suppose next that φ is $\neg\psi$. Then $\mathfrak{A} \not\models \psi[a_1, \dots, a_k]$, and by induction hypothesis on ψ we have $\mathfrak{B} \not\models \psi[b_1, \dots, b_k]$, and so $\mathfrak{B} \models \varphi[b_1, \dots, b_k]$. If φ is $\psi \wedge \theta$, reason similarly using the induction hypothesis on ψ and θ . Suppose finally that φ is $(\forall x_{k+1})(\psi)$ and assume for contradiction that $\mathfrak{B} \not\models \varphi[b_1, \dots, b_k]$. Then, there exists some $b \in B$ such that $\mathfrak{B} \models \neg\psi[b_1, \dots, b_k, b]$. Let $q \in I_1$ and $a \in A$ be the partial isomorphism, and the element, guaranteed to exist by the forth property such that $\text{dom}(\{(a_i, b_i) : i = 1, \dots, k\}) \subseteq \text{dom}(q)$, $a \in \text{dom}(q)$, and $q(a) = b$. Then, $(\mathfrak{A}, a_1, \dots, a_k, a) \cong_{m-1} (\mathfrak{B}, b_1, \dots, b_k, b)$ as witnessed by the sequence

(I_1, \dots, I_m) . Thus, by induction hypothesis on $m - 1$ we have that $\mathfrak{A} \models \neg\psi[a_1, \dots, a_k, a]$, so that $\mathfrak{A} \not\models \varphi[a_1, \dots, a_k]$ contradicting our assumption. To show that if $\mathfrak{B} \models \varphi[b_1, \dots, b_k]$, then $\mathfrak{A} \models \varphi[a_1, \dots, a_k]$, use the same argument applying the back property and the fact that partial isomorphisms are injective. \square

We are ready to prove the Ordered Conjecture on the class \mathcal{O} of linear orders, or in general, on the class of all finite ordered structures for any relational vocabulary σ . The following well-known technical lemma will be crucial.

Lemma 2 *Let $r \geq 0$, and let $\mathfrak{A}, \mathfrak{B} \in \mathcal{O}$ be finite linear orders of cardinalities n and m respectively. If $n > 2^r$ and $m > 2^r$, then $(\mathfrak{A}, 0, n - 1) \cong_r (\mathfrak{B}, 0, m - 1)$.*

Proof: We define a sequence of sets of partial isomorphisms as follows: $I_0 = \{\{(0, 0), (n - 1, m - 1)\}\}$, and for $i \in \{1, \dots, r\}$, we define

$$I_i = I_{i-1} \cup \{p \cup \{(a, b(a, p))\} : p \in I_{i-1}, a \in A\} \cup \{p \cup \{(a(b, p), b)\} : p \in I_{i-1}, b \in B\}$$

where $b(a, p)$ and $a(b, p)$ are defined according to the following rules: if there exists some $a' \in \text{dom}(p)$ such that $|a - a'| \leq 2^{r-i}$, then $b(a, p) = p(a') + a - a'$; otherwise, $b(a, p) = \lceil (p(c) + p(d))/2 \rceil$ where $c = \min\{a' \in \text{dom}(p) : a' > a\}$ and $d = \max\{a' \in \text{dom}(p) : a' < a\}$. The definition of $a(b, p)$ is analogous using $\text{ran}(p)$, p^{-1} , and b instead of $\text{dom}(p)$, p , and a . We only need to show that each element of I_i is a partial isomorphism; the back and forth properties are obvious by construction. We proceed by induction on $i \in \{0, \dots, r\}$. We also show that for every $q \in I_i$, and $a, a' \in \text{dom}(q)$ it holds that $a - a' = q(a) - q(a')$ if $|a - a'| \leq 2^{r-i-1}$ or $|q(a) - q(a')| \leq 2^{r-i-1}$. Clearly, I_0 satisfies the requirements.

Suppose next that $i > 0$ and that the claim holds for every $i' < i$. We need to show that given $p \in I_{i-1}$ and $a \in A$, the mapping $q = p \cup \{(a, b(a, p))\}$ is a partial isomorphism with the property above; the case $b \in A$ and $q = p \cup \{(a(b, p), b)\}$ is handled in a similar way. By induction hypothesis, p is a partial isomorphism on its domain with the required property; thus, it suffices to consider the pair $(a, b(a, p)) \in q$. Consider two cases: (i) for some $a' \in \text{dom}(p)$ we have that $|a - a'| \leq 2^{r-i}$, and (ii) for every $a' \in \text{dom}(p)$ we have that $|a - a'| > 2^{r-i}$. For case (i), by construction of $b(a, p)$ we have that $a - a' = b(a, p) - p(a')$, and $a \leq a'$ if and only if $b(a, p) \leq p(a')$. We also need to check that $b(a, p) \leq m - 1$. Suppose for contradiction that $b(a, p) > m - 1$; then $(m - 1) - p(a') < a - a' \leq 2^{r-i}$ and so, by induction hypothesis on $p \in I_{i-1}$, we have $(n - 1) - a' = (m - 1) - p(a') < a - a'$ because $(n - 1) \in \text{dom}(p)$. It follows that $a > n - 1$, which is ridiculous. Suppose on the other hand, that we are in case (ii). Let $c, d \in \text{dom}(p)$ be defined as in the construction of $b(a, p)$. It is clear this time that $b(a, p) \leq m - 1$. Moreover, we have that $|c - d| > 2 \cdot 2^{r-i} = 2^{r-i+1}$. It follows that $|p(c) - p(d)| > 2^{r-i}$ for otherwise, by induction hypothesis on $p \in I_{i-1}$, we would have $|c - d| = |p(c) - p(d)| \leq 2^{r-i} < 2^{r-i+1}$. Hence, from the choice of $b(a, p)$ we have $|b(a, p) - p(c)| > 2^{r-i-1}$ and $|b(a, p) - p(d)| > 2^{r-i-1}$, from which it follows that the required property is satisfied, and $a \leq a'$ if and only if $b(a, p) \leq p(a')$ for every $a' \in \text{dom}(p)$. Only one detail is left: that q is well-defined and injective. But this properties follows immediately from the above because $a = a'$ if and only if $a \leq a'$ and $a' \leq a$. This completes the induction step, and the proof. \square

Theorem 9 *Let σ be a relational vocabulary. If C is the class of all finite ordered struc-*

tures for (σ, \leq) , then $\text{FO}[C] < \text{LFP}[C]$.

Proof: Consider the following simple Boolean query Q on C . For every $\mathfrak{A} \in C$, we set $\mathfrak{A} \in Q$ if and only if the universe of \mathfrak{A} has even cardinality. It is obvious that Q is computable in polynomial-time, and therefore, by the Immerman-Vardi theorem, it is least fixed-point definable on C . We claim that Q is not first-order definable on C . For suppose φ is a first-order sentence for (σ, \leq) that defines it. Let $\psi(x, y)$ be the first-order formula for the vocabulary (\leq) that is obtained from φ as follows: first, let φ' be the result of syntactically replacing each occurrence of $R_i(x_1, \dots, x_k)$ in φ by $x \neq x$; then $\psi(x, y) = (\forall z)(x \leq z \vee z \leq y) \wedge \varphi'$. Let r be the quantifier rank of ψ , and let $\mathfrak{A}, \mathfrak{B} \in C$ be two structures of the form $(A, \emptyset, \dots, \emptyset, \leq^{\mathfrak{A}})$ and $(B, \emptyset, \dots, \emptyset, \leq^{\mathfrak{B}})$ of cardinalities $2^r + 1$ and $2^r + 2$ respectively. Such structures exist because C is the class of all finite ordered structures for (σ, \leq) . Observe that $\mathfrak{A} \not\models \varphi$ and $\mathfrak{B} \models \varphi$ by the assumption that φ defines Q on C . Moreover, it is not difficult to show that $(A, \leq^{\mathfrak{A}}) \not\models \psi[0, 2^r]$ and $(B, \leq^{\mathfrak{B}}) \models \psi[0, 2^r + 1]$. Since the quantifier rank of ψ is r , it follows from the contrapositive of Theorem 8, that $(A, \leq^{\mathfrak{A}}, 0, 2^r) \not\cong_r (B, \leq^{\mathfrak{B}}, 0, 2^r + 1)$. Since this contradicts Lemma 2, the proof is complete. \square

4.3.2 Unary vocabularies

A unary vocabulary is a relational vocabulary in which every relation symbol has arity one. Poizat [Poi82], and independently, Immerman and Kozen [IK89], have shown the following result:

Theorem 10 *Let σ be a unary vocabulary, let C be a class of finite ordered structures for (σ, \leq) , and let φ be a first-order formula for (σ, \leq) with at most three free variables. There exists a first-order formula ψ with at most three variables (free or not) such that φ and ψ define the same query on C .*

The proof of this theorem uses k -pebble games, a game-theoretic characterization of first-order logic with k variables (see [KV90]). Using techniques from finite model theory, Dawar, Lindell and Weinstein [DLW96] have shown that Poizat's result implies that the Ordered Conjecture holds when restricted to unary vocabularies. In the terminology used in that paper, they show that every class of ordered structures for a unary vocabulary is not 3-compact, and that on each such class, there exists a LFP-definable query that is not definable in first-order logic with three variables. Here we give a more direct proof of their result using a simple diagonalization argument that exploits the fact that first-order formulas with at most k variables can be evaluated in polynomial-time (for a polynomial that depends on k).

Theorem 11 *Let σ be a unary relational vocabulary. If C is an infinite class of finite ordered structures for (σ, \leq) , then $\text{FO}[C] < \text{LFP}[C]$.*

Proof: Let $\sigma = (R_1, \dots, R_s)$ where each R_i is a unary relation symbol. The proof is a simple diagonalization. Let $\varphi_1, \varphi_2, \dots$ be a complete enumeration of all first-order formulas for (σ, \leq) with a single free variable and at most three variables overall (free or not). We may assume that the encoding of φ_n , denoted $[\varphi_n]$, is a word over the alphabet $\Xi \cup \{R_1, \dots, R_s, \leq\}$ that is computable from input 1^n by a deterministic Turing machine

running within time $p(n)$ for some fixed polynomial p . Observe that the length of $[\varphi_n]$ is necessarily bounded by $p(n)$. Next define the following unary query Q on C . Let $\mathfrak{A} \in C$; recall the universe of \mathfrak{A} is an ordinal, say $A = \{0, \dots, n-1\}$. Then, $Q(\mathfrak{A}) = \{a \in A : \mathfrak{A} \models \neg\varphi_a[a]\}$. We first claim that Q is computable in polynomial-time on C . The machine on input $\langle \mathfrak{A}, a \rangle$, writes down the word 1^a , and runs the polynomial-time machine that generates $[\varphi_a]$. The machine then determines if $\mathfrak{A} \models \neg\varphi_a[a]$. Since φ_a has only three variables (free or not), this computation can be made in polynomial-time in the length of $\langle \mathfrak{A}, a \rangle \# [\varphi_a]$. We refer the reader to Vardi [Var96, Proposition 3.1] for this; the problem is called there *combined complexity of FO^s*. It follows that Q is computable on C in polynomial-time, and by the Immerman-Vardi Theorem, it is LFP-definable on C . The next claim is that Q is not first-order definable on C . For suppose it were; then, by Theorem 10 it would also be definable by a first-order formula φ with at most three variables (free or not). Let $n \in \omega$ be such that $\varphi = \varphi_n$, and let $\mathfrak{A} \in C$ be a structure of cardinality at least $n+1$. Such a structure exists because C is infinite. From the definition of Q we have that $n \in Q(\mathfrak{A})$ if and only if $\mathfrak{A} \models \neg\varphi_n[n]$, if and only if $\mathfrak{A} \models \neg\varphi[n]$ because $\varphi = \varphi_n$, if and only if $n \notin Q(\mathfrak{A})$ because φ defines Q . This absurdity proves the claim, and the theorem. \square

4.3.3 Partial fixed-point logic

Recall the definition of partial fixed-point logic in Section 2.2. Although we have characterized the computational power of least fixed-point logic on classes of finite ordered structures, we have not focussed yet on partial fixed-point logic. In fact, a result anal-

ogous to the Immerman-Vardi Theorem can be proved, namely, that partial fixed-point logic captures **PSPACE** on classes of finite ordered structures. More precisely, if σ is a relational vocabulary, C is a class of finite ordered structures for (σ, \leq) , and Q is a query on C , then Q is PFP-definable on C if and only if Q is computable on C in polynomial-space. We refer the reader to the book of Ebbinghaus and Flum [EF95, Chapter 6] for a proof of this fact; the proof is similar to the one for Theorem 5. The following result was proved by Dawar and Hella [DH95].

Theorem 12 *Let σ be a relational vocabulary. If C is an infinite class of finite ordered structures for (σ, \leq) , then $\text{FO}[C] < \text{PFP}[C]$.*

Proof: Let $\sigma = (R_1, \dots, R_s)$. The proof is almost identical to the proof of Theorem 11. Let $\varphi_1, \varphi_2, \dots$ be a complete enumeration of all first-order formulas for (σ, \leq) with a single free variable. We may assume that the encoding of φ_n , denoted $[\varphi_n]$, is a word over the alphabet $\Xi \cup \{R_1, \dots, R_s, \leq\}$ that is computable from input 1^n by a deterministic Turing machine running within time $p(n)$ for some fixed polynomial p . Observe that the length of $[\varphi_n]$ is necessarily bounded by $p(n)$. Next define the following unary query Q on C . Let $\mathfrak{A} \in C$; recall the universe of \mathfrak{A} is an ordinal, say $A = \{0, \dots, n-1\}$. Then, $Q(\mathfrak{A}) = \{m \in A : \mathfrak{A} \models \neg\varphi_m[m]\}$. Since to decide $\mathfrak{A} \models \neg\varphi_m[m]$ can be done in polynomial-space given $\langle \mathfrak{A}, m \rangle \# [\neg\varphi]$ (see [Var96, Table 1] under the name *combined complexity of FO*), the above query is computable in polynomial-space. It follows that Q is PFP-definable on C by the remarks preceding the theorem. We show that Q is not FO-definable on C . For suppose that it is definable via the first-order formula φ . Let $n \in \omega$

be such that $\varphi = \varphi_n$, and let $\mathfrak{A} \in C$ be a structure of cardinality at least $n + 1$; since C is an infinite class of finite structures, such a structure exists. Observe that $n \in Q(\mathfrak{A})$ if and only if $\mathfrak{A} \models \neg\varphi_n[n]$ from the definition of Q , if and only if $\mathfrak{A} \models \neg\varphi[n]$ because $\varphi = \varphi_n$, if and only if $n \notin Q(\mathfrak{A})$ because φ defines Q . Since this is absurd, the proof is complete. \square

As consequence to Theorem 12 Dawar and Hella obtain the following result linking the Ordered Conjecture to complexity theory.

Corollary 1 *If the Ordered Conjecture fails, then $\mathbf{P} \neq \mathbf{PSPACE}$.*

Proof: Suppose that the conjecture fails. Then, there exists a relational vocabulary σ , and an infinite class of finite ordered structures C for (σ, \leq) such that $\text{FO}[C] = \text{LFP}[C]$. According to Theorem 12 we have that $\text{FO}[C] < \text{PFP}[C]$. Let Q be a query witnessing this fact, and let r be the arity of Q . Since Q is PFP-definable on C , by the remarks preceding Theorem 12, there exists a deterministic Turing machine M and a polynomial p such that, for every $\mathfrak{A} \in C$ and every $(a_1, \dots, a_r) \in A^r$, whenever M is presented with input $\langle \mathfrak{A}, a_1, \dots, a_r \rangle$, it runs within $p(|\langle \mathfrak{A}, a_1, \dots, a_r \rangle|)$ space, eventually halts, and accepts if and only if $(a_1, \dots, a_r) \in Q(\mathfrak{A})$. Set M to shut off its computation if more than $p(n)$ space is used on *any* input of length n ; use the space constructibility of polynomials for this (see [BDG96]). Let L be the language over the alphabet $\{0, 1, \#\}$ decided by M . Clearly, $L \in \mathbf{PSPACE}$. We claim that $L \notin \mathbf{P}$. For suppose the contrary, and let N be a deterministic Turing machine running in polynomial-time that decides L . By definition, N computes Q on C , and by the Immerman-Vardi Theorem, Q is LFP-definable on C . But

$\text{LFP}[C] = \text{FO}[C]$, and therefore, Q is also FO-definable on C ; this contradiction proves the corollary. \square

Corollary 1 suggests that a refutation to the Ordered Conjecture, if any, would be difficult to prove. In Chapter 5 we will show that, in fact, a proof of the Ordered Conjecture would also have important consequences in complexity theory.

Chapter 5

The conjecture on finite arithmetical structures

5.1 Motivation

We have seen in Chapter 4 that some restrictions and relaxations of the Ordered Conjecture are true. Moreover, Corollary 1 shows that a negative solution to the Ordered Conjecture would imply a long-standing open problem in complexity theory. In this chapter, we are interested in the complementary question; namely, we seek for complexity theoretic consequences of a positive solution to the Ordered Conjecture. Ultimately, one would like to find a complete characterization of the problem.

The Ordered Conjecture is about all classes of finite ordered structures, and in particular, about classes of finite structures including any particular built-in's. Recall from

Chapter 3 that built-in's of increasing power, like the linear order and the arithmetical predicates, led to computational characterizations of least fixed-point logic and first-order logic respectively. Thus, in our quest for complexity theoretic consequences of the Ordered Conjecture, one may consider such classes. As a matter of fact, if the linear order is the only built-in, the conjecture is true as shown in Theorem 9. Thus, we focus our attention to finite structures with arithmetical built-in predicates.

5.2 Arithmetical versus pure arithmetical structures

Recall the definition of arithmetical structures in Section 3.3.2. We are interested in proving a result analogous to Theorem 9 for the class of all finite arithmetical structures for a relational vocabulary σ . Surprisingly as it seems at first sight, we can prove the result only if σ is a non-empty vocabulary. We elaborate on an easy consequence of the result of Furst, Saxe, and Sipser [FSS84], and independently, Ajtai [Ajt83], that the parity function requires super-polynomial size circuits of constant depth and unbounded fan-in. Namely, we use the fact that the language defined as $\text{PARITY} = \{w \in \{0,1\}^* : \text{the number of ones in } w \text{ is even}\}$ is not in **LH**; see Boppana and Sipser [BS90, page 777].

Theorem 13 *Let σ be a non-empty relational vocabulary. If C is the class of all finite arithmetical structures for $(\sigma, \leq, +, \times)$, then $\text{FO}[C] < \text{LFP}[C]$.*

Proof: Suppose that $\sigma = (R_1, \dots, R_s)$, and let $r_i = \text{ar}(R_i)$; since σ is non-empty, these numbers are well-defined positive numbers (recall that we are not considering propositional relations). Let Q be the following Boolean query on C : for every $\mathfrak{A} \in C$, we set $\mathfrak{A} \in Q$

if and only if the cardinality of the set $\{a \in A : (0, \dots, 0, a) \in R_1^{\mathfrak{A}}\}$ is even. It is not difficult to see that $\mathfrak{A} \in Q(\mathfrak{A})$ if and only if the word $\langle \mathfrak{A}, R_1^{\mathfrak{A}}, \dots, R_s^{\mathfrak{A}}, \leq^{\mathfrak{A}}, +^{\mathfrak{A}}, \times^{\mathfrak{A}} \rangle$ over the alphabet $\{0, 1, \#\}^*$, contains an even number of ones between the positions $n + 2$ and $2n + 1$ included. Clearly, Q is computable on C in polynomial-time, and therefore, by the Immerman-Vardi Theorem, it is LFP-definable on C . We claim that Q is not FO-definable on C ; this will prove the theorem. Suppose for contradiction that it is. By Theorem 6 we have that Q is computable by an alternating Turing machine running in logarithmic-time and a constant number of alternations. Let M be such a machine. We build an alternating machine N that decides PARITY in logarithmic-time and a constant number of alternations. This contradiction will prove the claim.

Let us assume that M only queries its input once in each path; according to Theorem 13, this is no loss of generality. We specify the behavior of N on input w , of length n . The machine first determines the length of its input in $4 \log n$ steps following the procedure of Lemma 1. Then it starts a simulation of M until a query is performed. Then, the machine resolves the query pretending that the input is

$$1^n \# w 0^{n^{r_1} - n} \# 0^{n^{r_2}} \# \dots \# 0^{n^{r_s}} \# \chi(\leq^n) \# \chi(+^n) \# \chi(\times^n). \quad (5.1)$$

In order to do so, suppose that p is the position queried by M . Then, if $p \leq n$, the machine pretends that the answer is 1; if $p = (n + 1)$, the machine pretends that the answer is $\#$; if $n + 2 \leq p \leq 2n - 1$, the machine looks up the symbol at position $p - n - 1$ of the actual input w , and answers accordingly; if $2n \leq p \leq n + 1 + n^{r_1}$, the machine pretends that the answer is a 0; if $p = n + 2 + n^{r_1}$, the machine pretends that the answer

is $\#$; and so on for the rest of positions. All the necessary arithmetic operations can be done in logarithmic-time and a constant number of alternations according to Section B.1. Once the unique query of the path being simulated is resolved, the machine proceeds to complete the simulation of M . Observe that the machine runs in logarithmic-time in the length of the word 5.1, which is also logarithmic in n . The number of alternations remains constant too (recall that the machine makes a unique query in each path). It remains to see that N decides PARITY. For suppose that $w \in \{0, 1\}^*$. Then, by the construction of Q we have that $w \in \text{PARITY}$ if and only if $\mathfrak{A} \in Q$ where \mathfrak{A} is the unique structure in C such that $\langle \mathfrak{A} \rangle$ is the word 5.1 above. Therefore, $w \in \text{PARITY}$ if and only if M on input $\langle \mathfrak{A} \rangle$ accepts, and by construction, if and only if N on input w accepts. This completes the proof of the claim, and the theorem. \square

The proof of Theorem 13 reveals that the hypothesis of σ being non-empty is crucial. In the following, we investigate the difficulty to remove this hypothesis. Finite arithmetical structures for the empty vocabulary are called *finite pure arithmetical structures*; the class of all them is denoted \mathcal{A} . The next theorem is about \mathcal{A} but it is formulated as it is to emphasize its complementarity with Theorem 13.

Theorem 14 *Let $\sigma = ()$ be the empty vocabulary, and let C be the class of all finite arithmetical structures for $(\sigma, \leq, +, \times)$. The following are equivalent:*

- (i) $\text{FO}[C] = \text{LFP}[C]$
- (ii) $\text{LINH} = \mathbf{E}$
- (iii) $\text{DLOGTIME-uniform } \text{AC}^0 = \text{P-uniform } \text{AC}^0$

Proof: We show that the statements close a cycle of implications.

((i) \Rightarrow (ii)) Assume that every LFP-definable query on C is already FO-definable. Clearly $\mathbf{LINH} \subseteq \mathbf{E}$. Fix any language $L \in \mathbf{E}$ over the alphabet $\{0, 1\}$; the case of larger alphabets can be reduced to this using standard techniques (see [BDG96]). We show that $L \in \mathbf{LINH}$. It suffices to show that the language $1L = \{1w : w \in L\} \in \mathbf{LINH}$ because to recognize L we can build an alternating Turing machine that prepends a 1 to the input and simulates the machine witnessing that $1L \in \mathbf{LINH}$. The leading one assures that different words in $1L$ encode different natural numbers in binary. Define a Boolean query Q on C as follows: let $\mathfrak{A}_n \in C$ be the pure arithmetical structure of universe $\{0, \dots, n-1\}$; then $\mathfrak{A}_n \in Q$ if and only if $b_n(n) \in 1L$; recall that $b_n(m)$ with $n \geq m$ is the unique binary representation of m of length $\log n$ with leading zeros if necessary. Observe that if \mathfrak{A}_n is a structure as above, then

$$|\langle \mathfrak{A}_n \rangle| = |1^n \# \chi(\leq^n) \# \chi(+^n) \# \chi(\times^n)| = 2n^3 + n^2 + n + 3 \leq 2^{c \log_2(n)} \leq 2^{c|b_n(n)|}$$

for some constant c . Thus, since $1L \in \mathbf{E}$, it is easy to see that Q is computable on C in polynomial-time; the machine on input $\langle \mathfrak{A}_n \rangle$ first computes $b_n(n)$, and then simulates the exponential-time machine for $1L$; its running time is $2^{d|b_n(n)|} \leq |\langle \mathfrak{A}_n \rangle|^e$ for some constants d and e . Therefore, since finite arithmetical structures are ordered, by the Immerman-Vardi Theorem, Q is LFP-definable on C . Now, by hypothesis, Q is also FO-definable, and therefore, by Theorem 1, it is computable on C in alternating logarithmic-time with a constant number of alternations. Using again the fact that $|\langle \mathfrak{A} \rangle| \leq 2^{c|b_n(n)|}$, we obtain that $1L$ can be decided in alternating linear-time and a constant number of alternations;

that is, $1L \in \mathbf{LINH}$.

((ii) \Rightarrow (iii)) Assume that $\mathbf{LINH} = \mathbf{E}$ and fix a language $L \in \mathbf{P}$ -uniform \mathbf{AC}^0 . Let $D \in \mathbf{P}$ be the direct connection language of a circuit family witnessing this fact. Define the language $F = \{w : w\#0^n \in D\}$. Since $D \in \mathbf{P}$ and for every word $w\#0^n \in D$ we have that $|w| = c \log n$ for some constant c (recall the definition of direct connection language in Section 2.3.3), we have that $F \in \mathbf{E}$. Now, by hypothesis, $F \in \mathbf{LINH}$. Using again the fact every word $w\#0^n \in D$ satisfies $|w| = c \log n$ we obtain that $D \in \mathbf{LH}$. Therefore, $L \in \mathbf{LH}$ -uniform \mathbf{AC}^0 . Now, from the results of Barrington, Immerman and Straubing [BIS90] we know that \mathbf{LH} -uniform $\mathbf{AC}^0 = \mathbf{DLOGTIME}$ -uniform \mathbf{AC}^0 and the claim follows.

((iii) \Rightarrow (i)) We show the contrapositive. Assume that Q is a LFP-definable query on C witnessing that $\text{FO}[C] < \text{LFP}[C]$. Let k be the arity of Q ; we can assume that $k > 0$ by adding dummy variables. Fix a number $n > 0$ and define L_n as the set of words of length n^k of the form $0^{m-1}10^{n^k-m}$, where the n -ary encoding of m , say (m_{k-1}, \dots, m_0) , is a tuple that belongs to $Q(\mathfrak{A})$. For the rest of this proof we identify numbers and their n -ary encodings when n is clear from the context. Define $L = \bigcup_{n \geq 1} L_n$. It is fairly easy to show that L has a \mathbf{P} -uniform \mathbf{AC}^0 family of circuits. Namely, the circuit for inputs of length n^k is described by the following formula

$$\bigvee_{\bar{m} \in Q(\mathfrak{A}_n)} \left(x_{\bar{m}} \wedge \bigwedge_{j=0, j \neq \bar{m}}^{n^k-1} \neg x_j \right).$$

Here x_0, \dots, x_{n^k-1} are the Boolean inputs of the circuit listed from left to right. For lengths that are not of the form n^k , the circuit is the trivial constantly false circuit.

The **P**-uniformity is also clear because Q is LFP-definable and therefore computable in polynomial-time.

We show that L is not in **DLOGTIME**-uniform **AC**⁰. Assume for contradiction that it is. Then, by Barrington, Immerman and Straubing [BIS90], it is computable in alternating logarithmic-time and a constant number of alternations. Let $L' = \{1^n \# w \# \chi(\leq^n) \# \chi(+^n) \# \chi(\times^n) : w \in L_n, n \geq 1\}$. Since each word in L_n has length n^k , it is clear that L' is a set of words of the form $\langle \mathfrak{A} \rangle$ where \mathfrak{A} is a finite arithmetical structure for the vocabulary $(R, \leq, +, \times)$ where R is a k -ary relation symbol. Let C be the class of all finite arithmetical structures for $(R, \leq, +, \times)$. Let Q' be the Boolean query on C defined as follows: for every $\mathfrak{A} \in C$, we set $\mathfrak{A} \in Q'$ if and only if $\langle \mathfrak{A} \rangle \in L'$. Since L' is computable in alternating logarithmic-time with a constant number of alternations, by Theorem 6 we have that Q' is FO-definable on C . Let φ be a first-order sentence for $(R, \leq, +, \times)$ that defines it. We let $\psi(x_1, \dots, x_k)$ be the formula obtained from φ by replacing each occurrence of $R(y_1, \dots, y_k)$ by $\bigwedge_{i=1}^k x_i = y_i$. By construction, ψ is a first-order formula for the vocabulary $(\leq, +, \times)$ that defines Q on C . Since this is a contradiction, the proof is complete. \square

Separating **LINH** from **E** is an open problem analogous to separating **PH** from **EXP**. In its equivalent setting in terms of circuit uniformity, the problem is particularly interesting. There are a number of problems that are known to have **P**-uniform circuits but not **DLOGTIME**-uniform circuits. For example, it is an open question whether integer division can be done by a **DLOGTIME**-uniform family of **NC**¹ circuits, while it is known

that a \mathbf{P} -uniform such family exists (see [BCH86]). This connection with uniform circuits was pointed out without proof by Gurevich, Immerman, and Shelah [GIS94]. They isolated the class of finite structures of the form $\mathfrak{B}_n = (\{0, \dots, n-1\}, \leq^{\mathfrak{B}_n}, \text{BIT}^{\mathfrak{B}_n})$ where $\leq^{\mathfrak{B}_n}$ is the standard linear order, and $\text{BIT}^{\mathfrak{B}_n}$ is the binary relation that contains all pairs $(a, b) \in \{0, \dots, n-1\}^2$ such that the a -th bit of the binary representation of b is one. Let $\mathcal{B} = \{\mathfrak{B}_n : n \in \omega\}$. We can prove the following:

Theorem 15 $\text{FO}[\mathcal{A}] < \text{LFP}[\mathcal{A}]$ if and only if $\text{FO}[\mathcal{B}] < \text{LFP}[\mathcal{B}]$.

Proof: The result follows immediately from the facts that $+$ and \times are definable from BIT and \leq , and that BIT is definable from $+$ and \times . See Appendix B for these results. \square

Recently, Dawar, Doets, Lindell, and Weinstein [DDLW98] have shown that \leq is definable from BIT . It follows that $\text{FO}[\mathcal{B}] < \text{LFP}[\mathcal{B}]$ if and only if $\text{FO}[\mathcal{C}] < \text{LFP}[\mathcal{C}]$, where \mathcal{C} is the class of finite structures of the form $\mathfrak{C}_n = (\{0, \dots, n-1\}, \text{BIT}^{\mathfrak{B}_n})$. The author and Kolaitis [AK99] have linked this question to finite set theory as a continuation to the work started by Dawar, Doets, Lindell, and Weinstein. Before we move to our next section, we are ready to state our first complexity consequence of the Ordered Conjecture being true:

Corollary 2 *If the Ordered Conjecture holds, then $\mathbf{LINH} \neq \mathbf{E}$.*

Proof: If the Ordered Conjecture holds, then $\text{FO}[C] < \text{LFP}[C]$ where C is the class of finite pure arithmetical structures. The result follows from Theorem 14. \square

5.3 Definability on the standard model of arithmetic

Theorems 13 and 14 strongly motivate the study of first-order and least fixed-point definability on finite pure arithmetical structures. There is an obvious connection between these structures and the standard model of arithmetic $(\omega, \leq, +, \times)$. The purpose of this section is to make this analogy precise. We show that our problem can be restated as a problem of definability on $(\omega, \leq, +, \times)$.

5.3.1 Some definitions and easy facts

Let $\mathfrak{A}_\omega = (\omega, \leq, +^{\mathfrak{A}_\omega}, \times^{\mathfrak{A}_\omega})$ be the standard model of arithmetic where $+^{\mathfrak{A}_\omega}$ and $\times^{\mathfrak{A}_\omega}$ are the graphs of the addition and multiplication functions on ω . We may write simply $(\omega, \leq, +, \times)$ to reduce notation. For each $n \in \omega$, let $\mathfrak{A}_n = (n, \leq^{\mathfrak{A}_n}, +^{\mathfrak{A}_n}, \times^{\mathfrak{A}_n})$ be the *finite pure arithmetical structure* with universe $n = \{0, \dots, n-1\}$. Let $\mathcal{A} = \{\mathfrak{A}_n : n \in \omega\}$ be the class of finite pure arithmetical structures. The following concept will be of fundamental importance to us.

Definition 8 *Let Q be a k -ary query on \mathcal{A} . We say that Q is absolute if and only if for every $n \in \omega$ we have that $Q(\mathfrak{A}_n) = (\bigcup_{i \in \omega} Q(\mathfrak{A}_i)) \cap n^k$.*

The following are some easy facts about absolute queries.

Fact 1 *Let Q be an absolute k -ary query on \mathcal{A} , let $\overline{Q}(\mathfrak{A}_n) = n^k - Q(\mathfrak{A}_n)$, let $R \subseteq \omega^k$, and let $Q'(\mathfrak{A}_n) = R \cap n^k$. It holds that*

$$(i) \bigcup_{i \in \omega} Q'(\mathfrak{A}_i) = R \text{ and } \bigcup_{i \in \omega} \overline{Q}(\mathfrak{A}_i) = \omega^k - \bigcup_{i \in \omega} Q(\mathfrak{A}_i),$$

(ii) Q' and \overline{Q} are absolute queries,

(iii) for every $m \leq n \leq \omega$ we have that $Q(\mathfrak{A}_m) = Q(\mathfrak{A}_n) \cap m^k$.

Proof: Clearly, $\bigcup_{i \in \omega} Q'(\mathfrak{A}_i) = R$. On the other hand, fix $\bar{a} \in \omega^k$. Then, $\bar{a} \in \bigcup_{i \in \omega} \overline{Q}(\mathfrak{A}_i)$ if and only if for some $i \in \omega$ we have $\bar{a} \in i^k - Q(\mathfrak{A}_i)$. By absoluteness of Q , if and only if for some $i \in \omega$ we have $\bar{a} \in i^k - (\bigcup_{j \in \omega} Q(\mathfrak{A}_j)) \cap i^k$, if and only if for some $i \in \omega$ we have $\bar{a} \in i^k - \bigcup_{j \in \omega} Q(\mathfrak{A}_j)$, if and only if $\bar{a} \in \omega^k - \bigcup_{j \in \omega} Q(\mathfrak{A}_j)$. This completes property (i). The absoluteness of Q' is trivial by definition; the absoluteness of \overline{Q} follows from the previous three claims; this is property (ii). For property (iii) reason as follows. Suppose that $\bar{m} \in m^k$ with $m \leq n \leq \omega$. Then, $\bar{m} \in Q(\mathfrak{A}_m)$ if and only if $\bar{m} \in (\bigcup_{i \in \omega} Q(\mathfrak{A}_i)) \cap m^k$ because Q is absolute, if and only if $\bar{m} \in (\bigcup_{i \in \omega} Q(\mathfrak{A}_i)) \cap n^k$ because $m \leq n$, if and only if $\bar{m} \in Q(\mathfrak{A}_n)$ because Q is absolute again. \square

We introduce a syntactical class of first-order formulas. Recall the alphabet Ξ from Section 2.2. Let $\sigma = (R_1, \dots, R_s)$ be a relational vocabulary, and let r_i be the arity of R_i . We say that a word φ from $(\Xi \cup \{R_1, \dots, R_s, \leq\})^*$ is a Δ_0 -formula for σ , denoted $\varphi \in \Delta_0(\sigma)$, if and only if one of the following holds:

- (i) φ is $x = y$ for some $x, y \in \mathcal{V}_0$,
- (ii) φ is $x \leq y$ for some $x, y \in \mathcal{V}_0$,
- (iii) φ is $R_i(x_1, \dots, x_{r_i})$ for some $x_1, \dots, x_{r_i} \in \mathcal{V}_0$,
- (iv) φ is $\neg\psi$ for some $\psi \in \Delta_0(\sigma)$,
- (v) φ is $\psi \wedge \theta$ for some $\psi, \theta \in \Delta_0(\sigma)$,

(vi) φ is $(\forall x \leq y)(\psi)$ for some $\psi \in \Delta_0(\sigma)$ and $x, y \in \mathcal{V}_0$.

We may use abbreviations for \forall , \rightarrow , and $(\exists x \leq y)$ in the usual way. The semantics of $(\forall x \leq y)(\psi)$ is defined as $(\forall x)(x \leq y \rightarrow \psi)$. Given a first-order formula $\varphi \in \text{FO}(\sigma)$ and a variable $x \in \mathcal{V}_0$, the *relativization of φ to x* , denoted $\varphi^{\leq x}$, is the Δ_0 -formula that results from φ if we replace each occurrence $(\forall y)$ by $(\forall y \leq x)$.

We also define a syntactical class of least fixed-point formulas. Namely, a word ϕ from $(\Xi_{\text{FP}} \cup \{R_1, \dots, R_s, \leq\})^*$ is a *least fixed-point Δ_0 -formula for σ* , denoted $\phi \in \text{LFP}(\Delta_0)(\sigma)$, if and only if one of the following holds:

- (i) ϕ is $\Delta_0(\sigma)$,
- (ii) ϕ is $\neg\psi$ for some $\psi \in \text{LFP}(\Delta_0)(\sigma)$,
- (iii) ϕ is $(\psi \wedge \theta)$ for some $\psi, \theta \in \text{LFP}(\Delta_0)(\sigma)$,
- (iv) ϕ is $(\forall x \leq y)(\psi)$ for some $\psi \in \text{LFP}(\Delta_0)(\sigma)$ and $x, y \in \mathcal{V}_0$,
- (v) ϕ is $(\text{FP}_{x_1, \dots, x_k, X}\psi)$ for some $x = x_1, \dots, x_k \in \mathcal{V}_0$, $X \in \mathcal{V}_1$ with $\text{ar}(X) = k$, and $\psi \in \text{LFP}(\Delta_0)((\sigma, X))$ in which X occurs positively or does not occur.

The semantics of $\text{LFP}(\Delta_0)$ formulas are defined according to the semantics of Δ_0 -formulas. We will need the following technical definitions. Let $\sigma = (R_1, \dots, R_s)$ be a relational vocabulary and let $\mathfrak{A}, \mathfrak{B}$ be two structures for σ . We say that \mathfrak{A} is a *substructure of \mathfrak{B}* , denoted $\mathfrak{A} \subseteq \mathfrak{B}$, if and only if $A \subseteq B$ and for every $\bar{a} \in A^{r_i}$ we have that $\bar{a} \in R_i^{\mathfrak{A}}$ if and only if $\bar{a} \in R_i^{\mathfrak{B}}$. Suppose that $\mathfrak{A}, \mathfrak{B}$ are ordered structures for (σ, \leq) . We say that \mathfrak{B} is a *end-restriction of \mathfrak{A}* , denoted $\mathfrak{A} \subseteq_{\text{end}} \mathfrak{B}$, if and only if $\mathfrak{A} \subseteq \mathfrak{B}$ and for every $a \in A$ we

have that $\{b \in A : (b, a) \in \leq^{\mathfrak{A}}\} = \{b \in B : (b, a) \in \leq^{\mathfrak{B}}\}$. Observe that $\mathfrak{A}_m \subseteq \mathfrak{A}_n$ for every $m \leq n \leq \omega$; more strongly, $\mathfrak{A}_m \subseteq_{\text{end}} \mathfrak{A}_n$ for every $m \leq n \leq \omega$.

5.3.2 Absoluteness properties of Δ_0 -formulas and inductions

We first show some nice *absoluteness properties* of Δ_0 -formulas and inductions. In the following, we assume the vocabulary $(\leq, +, \times)$ when omitted. Thus, Δ_0 means $\Delta_0(\leq, +, \times)$.

Lemma 3 (Absoluteness of Δ_0 formulas) *Let σ be a relational vocabulary, and let \mathfrak{A} and \mathfrak{B} be two structures for (σ, \leq) such that $\mathfrak{A} \subseteq_{\text{end}} \mathfrak{B}$. For every $\varphi \in \Delta_0(\sigma)$ and every $(a_1, \dots, a_k) \in A^k$ we have that $\mathfrak{A} \models \varphi[a_1, \dots, a_k]$ if and only if $\mathfrak{B} \models \varphi[a_1, \dots, a_k]$.*

Proof: The proof is by induction on the structure of φ . Use the hypothesis that $\mathfrak{A} \subseteq \mathfrak{B}$ when φ is atomic. The only interesting case is when φ is $(\forall x \leq x_i)(\psi)$ for some $\psi \in \Delta_0(\sigma)$ and $x \in \mathcal{V}_0$. Then, $\mathfrak{A} \models \varphi[a_1, \dots, a_k]$ if and only if for every $a \leq a_i$ we have $\mathfrak{A} \models \psi[a_1, \dots, a_k, a]$, if and only if for every $a \leq a_i$ we have $\mathfrak{B} \models \psi[a_1, \dots, a_k, a]$ by induction hypothesis, if and only if $\mathfrak{B} \models \varphi[a_1, \dots, a_k]$. Here we use the hypothesis that $\mathfrak{A} \subseteq_{\text{end}} \mathfrak{B}$.
qed

Lemma 4 *Let Q be an absolute k -ary query on \mathcal{A} . If $\varphi \in \Delta_0$, then φ defines Q on \mathcal{A} if and only if φ defines the relation $\bigcup_{i \in \omega} Q(\mathfrak{A}_i)$ on $(\omega, \leq, +, \times)$.*

Proof: Let $R = \bigcup_{i \in \omega} Q(\mathfrak{A}_i)$. Assume first that φ defines Q on \mathcal{A} . Fix $n \in \omega$ and $(a_1, \dots, a_k) \in n^k$. Write \bar{a} for (a_1, \dots, a_k) . By absoluteness of Δ_0 -formulas we have that

$\mathfrak{A}_\omega \models \varphi[\bar{a}]$ if and only if $\mathfrak{A}_n \models \varphi[\bar{a}]$, if and only if $\bar{a} \in Q(\mathfrak{A}_n)$. By absoluteness of Q we have that $\bar{a} \in Q(\mathfrak{A}_n)$ if and only if $\bar{a} \in R \cap n^k$, if and only if $\bar{a} \in R$ and so φ defines R on \mathfrak{A}_ω . Suppose next that φ defines R on \mathfrak{A}_ω , and fix $n \in \omega$ and $\bar{a} \in n^k$. Then, by absoluteness, $\mathfrak{A}_n \models \varphi[\bar{a}]$ if and only if $\mathfrak{A}_\omega \models \varphi[\bar{a}]$, if and only if $\bar{a} \in R$, if and only if $\bar{a} \in R \cap n^k$. Now by absoluteness of Q we have that $R \cap n^k = Q(\mathfrak{A}_n)$ and so φ defines Q on \mathcal{A} . \square

We turn next to least fixed-point formulas. In the following, given a monotone operator G , we will use the alternative definition $I_G^\gamma = G(\bigcup_{\delta < \gamma} I_G^\delta)$ for every ordinal γ as Moschovakis [Mos74]. The difference with the definition in Chapter 2 is inessential. Let \mathfrak{A} be a structure and let $f : \mathcal{V}_0 \rightarrow A$ be a valuation function for \mathfrak{A} . If φ is a formula that is positive in the second-order r -ary variable X , we use the notation $I_\varphi^\gamma(\mathfrak{A}, f)$ to denote I_G^γ , where $G : A^r \rightarrow A^r$ is the operator defined as $G(B) = \{(a_1, \dots, a_r) \in A^r : \mathfrak{A} \models \varphi[f_{a_1, \dots, a_r}^{x_1, \dots, x_k}, B]\}$.

Lemma 5 (Absoluteness of Δ_0 -stages) *Let $\varphi(z_1, \dots, z_r, Z)$ be a Δ_0 -formula that is positive in the r -ary relation symbol Z . Let $\mathfrak{A} \subseteq_{\text{end}} \mathfrak{B}$ and let $f : \mathcal{V}_0 \rightarrow A$ be any valuation function. For every $\gamma \geq 0$ we have that $I_\varphi^\gamma(\mathfrak{A}, f) = I_\varphi^\gamma(\mathfrak{B}, f) \cap A^k$.*

Proof: By induction on $\gamma \geq 0$. Let $\bar{a} = (a_1, \dots, a_r) \in A^r$. Write \bar{z} for (z_1, \dots, z_r) . We have that $\bar{a} \in I_\varphi^\gamma(\mathfrak{A}, f)$ if and only if $\mathfrak{A} \models \varphi[f_{\bar{a}}^{\bar{z}}, \bigcup_{\delta < \gamma} I_\varphi^\delta(\mathfrak{A}, f)]$, and by induction hypothesis, if and only if $\mathfrak{A} \models \varphi[f_{\bar{a}}^{\bar{z}}, \bigcup_{\delta < \gamma} (I_\varphi^\delta(\mathfrak{B}, f) \cap A^k)]$. By absoluteness we get, if and only if $\mathfrak{B} \models \varphi[f_{\bar{a}}^{\bar{z}}, \bigcup_{\delta < \gamma} (I_\varphi^\delta(\mathfrak{B}, f) \cap A^k)]$. Factoring out A^k and by monotonicity, if and only if $\mathfrak{B} \models \varphi[f_{\bar{a}}^{\bar{z}}, \bigcup_{\delta < \gamma} I_\varphi^\delta(\mathfrak{B}, f)]$. Finally, by the definition of I_φ^γ we get, if and only if $\bar{a} \in I_\varphi^\gamma(\mathfrak{B}, f)$. The lemma is proved. \square

Lemma 6 (Absoluteness of least fixed-point Δ_0 -formulas) *Let σ be a relational vocabulary, and let \mathfrak{A} and \mathfrak{B} be two structures for (σ, \leq) such that $\mathfrak{A} \subseteq_{\text{end}} \mathfrak{B}$. For every $\varphi \in \text{LFP}(\text{Delta}_0)(\sigma)$ and every $(a_1, \dots, a_k) \in A^k$ we have that $\mathfrak{A} \models \varphi[a_1, \dots, a_k]$ if and only if $\mathfrak{B} \models \varphi[a_1, \dots, a_k]$.*

Proof: The proof by induction on the construction of φ . The only interesting case is when φ is $\text{FP}_{\bar{z}, Z}\psi$ for some $\psi \in \text{LFP}(\Delta_0)(\sigma)$ and the result follows immediately from Lemma 5. \square

Lemma 7 *Let Q be an absolute k -ary query on \mathcal{A} . If $\varphi \in \text{LFP}(\Delta_0)$, then φ defines Q on \mathcal{A} if and only if φ defines the relation $\bigcup_{i \in \omega} Q(\mathfrak{A}_i)$ on $(\omega, \leq, +, \times)$.*

Proof: Exactly as in Lemma 4. \square

5.3.3 Characterization of absolute definable queries

We show that FO-definable absolute queries on \mathcal{A} are precisely those that are defined by Δ_0 -formulas.

Lemma 8 *A first-order definable query on \mathcal{A} is absolute if and only if it is definable by a Δ_0 -formula.*

Proof: The implication from right to left is very easy. For the other direction, suppose that φ is a first-order formula defining Q on \mathcal{A} . Consider the following formula ψ :

$$\bigvee_{i=1}^k \left(\bigwedge_{j=1}^k x_j \leq x_i \wedge \varphi^{\leq x_i} \right) \quad (5.2)$$

Observe that ψ is Δ_0 . We show that it also defines Q . Fix $n \in \omega$ and $\bar{a} = (a_1, \dots, a_k) \in n^k$. Let $m = \max\{a_1, \dots, a_k\}$ and let $i \in \{1, \dots, k\}$ be such that $m = a_i$; clearly $m < n$. We have that $\bar{a} \in Q(\mathfrak{A}_n)$ if and only if $\bar{a} \in Q(\mathfrak{A}_n) \cap (m+1)^k$. By absoluteness of Q we have that $\bar{a} \in Q(\mathfrak{A}_n) \cap (m+1)^k$ if and only if $\bar{a} \in Q(\mathfrak{A}_{m+1})$. By hypothesis on φ , if and only if $\mathfrak{A}_{m+1} \models \varphi[\bar{a}]$. Since $m = a_i$ is the maximum in \mathfrak{A}_{m+1} , if and only if $\mathfrak{A}_{m+1} \models \varphi^{\leq x_i}[\bar{a}]$ (recall the choice of i). Recalling the definition of ψ we have that $\mathfrak{A}_{m+1} \models \varphi^{\leq x_i}[\bar{a}]$ if and only if $\mathfrak{A}_{m+1} \models \psi[\bar{a}]$. It follows by the absoluteness of Δ_0 -formulas that $\bar{a} \in Q(\mathfrak{A}_n)$ if and only if $\mathfrak{A}_n \models \psi[\bar{a}]$ as was to be proved. \square

We can actually prove the same result for $\text{LFP}(\Delta_0)$ formulas. Namely, LFP -definable absolute queries on \mathcal{A} are exactly those queries that are definable by $\text{LFP}(\Delta_0)$ formulas. For the following proof, we use the notation $S(\cdot)$ where S is a second-order variable, to mean that the variables that instantiate S are mute. The exact meaning will be clear from the context. We also use the notation $\varphi(x_1, \dots, x_k)$ to indicate that the free variables of φ are among x_1, \dots, x_k .

Lemma 9 *A least fixed-point definable query on \mathcal{A} is absolute if and only if it is definable by a least fixed-point Δ_0 -formula.*

Proof: As before, the implication from right to left is very easy. For the other direction, suppose that $\varphi(x_1, \dots, x_k)$ is a least fixed-point formula defining an absolute k -ary query Q on \mathcal{A} . From the normal form theorems for LFP on finite models we can assume that φ is of the form $(\exists u)(\text{LFP}_{\bar{z}, Z}\psi(x_1, \dots, x_k, z_1, \dots, z_r, Z)(x_1, \dots, x_k, u, \dots, u))$ for some first-order formula $\psi(x_1, \dots, x_k, z_1, \dots, z_r, Z)$ that is positive in the r -ary relation symbol Z

and does not involve u (see the book of Ebbinghaus and Flum [EF95, Lemma 7.2.8]). We use the notations \bar{x} and \bar{z} to abbreviate x_1, \dots, x_k and z_1, \dots, z_r respectively. Let y be a new variable, and let S be a new $(r+1)$ -ary second-order variable. Consider the following formula χ :

$$\bigvee_{i=1}^k \left(\bigwedge_{j=1}^k x_j \leq x_i \wedge (\exists u \leq x_i) (\text{LFP}_{y, \bar{z}, S} \psi^{\leq y}(\bar{x}, y, z_1, \dots, z_r, S(y, \cdot))(\bar{x}, x_i, u, \dots, u)) \right)$$

The claim is that χ also defines Q on \mathcal{A} . Again, fix $n \in \omega$ and $\bar{a} = (a_1, \dots, a_k) \in n^k$. As before we have that $\bar{a} \in Q(\mathfrak{A}_n)$ if and only if $\bar{a} \in Q(\mathfrak{A}_{m+1})$ where $m = \max\{a_1, \dots, a_k\}$. Let $i \in \{1, \dots, k\}$ be such that $m = a_i$. By hypothesis on φ we have that $\bar{a} \in Q(\mathfrak{A}_{m+1})$ if and only if $\mathfrak{A}_{m+1} \models \varphi[\bar{a}]$. Let $\rho(\bar{x}, \bar{z})$ be the formula $\psi^{(m+1)^r}$, the $(m+1)^r$ -th iterate of the formula ψ . That is, $\rho(\bar{x}, \bar{z})$ is obtained from ψ by replacing occurrences of S by ψ up to $(m+1)^r$ times; we may rename conflicting variables, and the last occurrence of S is replaced by $x \neq x$. Since $I_\psi = I_\rho^{(m+1)^r}$ on \mathfrak{A}_{m+1} , we conclude that $\mathfrak{A}_{m+1} \models \varphi[\bar{a}]$ if and only if $\mathfrak{A}_{m+1} \models (\exists u) \rho(\bar{x}, u, \dots, u)[\bar{a}]$. Since $m = a_i$ is the maximum of \mathfrak{A}_{m+1} we have that $\mathfrak{A}_{m+1} \models (\exists u) \rho(\bar{x}, u, \dots, u)[\bar{a}]$ if and only if $\mathfrak{A}_{m+1} \models (\exists u \leq y) \rho^{\leq y}(\bar{x}, y, u, \dots, u)[\bar{a}, a_i]$, and by absoluteness, if and only if $\mathfrak{A}_n \models (\exists u \leq y) \rho^{\leq y}(\bar{x}, y, u, \dots, u)[\bar{a}, a_i]$. Finally observe that $\rho^{\leq y}(\bar{x}, y, \bar{z})$ is syntactically the same formula as the $(m+1)^r$ -th iterate of $\psi^{\leq y}(\bar{x}, y, \bar{z}, S(y, \cdot))$. By monotonicity, we get that $\mathfrak{A}_n \models (\exists u \leq y) \rho^{\leq y}(\bar{x}, y, u, \dots, u)[\bar{a}, a_i]$ if and only if $\mathfrak{A}_n \models (\exists u \leq y) (\text{LFP}_{y, \bar{z}, S} \varphi^{\leq y}(\bar{x}, y, \bar{z}, S(y, \cdot))(\bar{x}, x_i, u, \dots, u))[\bar{a}]$; if and only if $\mathfrak{A}_n \models \chi[\bar{a}]$ as was to be proved. \square

5.3.4 Characterization of the absolute collapse

In this section we show that the restriction to absolute queries is not essential to our problem. That is to say, a necessary and sufficient condition for least fixed-point logic to collapse to first-order logic on \mathcal{A} , is that they collapse on the absolute queries.

Theorem 16 *The following are equivalent:*

- (i) *every least fixed-point definable query is first-order definable on \mathcal{A} ,*
- (ii) *every least fixed-point definable absolute query is first-order definable on \mathcal{A} .*

Proof: That the first statement implies the second is absolutely obvious. For the other direction we show the contrapositive. Suppose that Q is a k -ary least fixed-point definable query that is not first-order definable on \mathcal{A} . Consider the query

$$Q'(\mathfrak{A}_n) := \{(m_1, \dots, m_k, m) \in n^{k+1} : (m_1, \dots, m_k) \in Q(\mathfrak{A}_{m+1})\}$$

We first show that Q' is absolute. Fix $n \in \omega$ and $(m_1, \dots, m_k, m) \in n^{k+1}$; observe that $m < n$. We need to show that $(m_1, \dots, m_k, m) \in Q'(\mathfrak{A}_n)$ if and only if $(m_1, \dots, m_k, m) \in (\bigcup_{i \in \omega} Q'(\mathfrak{A}_i)) \cap n^{k+1}$. From left to right it is plain. From right to left reason as follows. Suppose that $(m_1, \dots, m_k, m) \in Q'(\mathfrak{A}_i)$ for some $i \in \omega$. From the definition of Q' we have that $(m_1, \dots, m_k) \in Q(\mathfrak{A}_{m+1})$. Since $m_j \leq m < n$ for every $j \leq \{1, \dots, k\}$, it follows from the definition of Q' again that $(m_1, \dots, m_k, m) \in Q'(\mathfrak{A}_n)$, and that is what we wanted.

The next claim is that Q' is least fixed-point definable on \mathcal{A} . Let $\varphi(x_1, \dots, x_k)$ be the formula witnessing that Q is least fixed-point definable on \mathcal{A} . Use \bar{x} and \bar{z} as abbreviations for x_1, \dots, x_k and z_1, \dots, z_r respectively. Assume without loss of generality that $\varphi(\bar{x}) =$

$(\exists u)(\text{LFP}_{\bar{z}, Z}(\bar{x}, \bar{z}, Z)(\bar{x}, u, \dots, u))$ where $\psi(\bar{x}, \bar{z}, Z)$ is a first-order formula positive in the r -ary relation symbol Z that does not involve u . Let y be a new variable and let S by an $(r + 1)$ -ary relation symbol. Consider the following formula $\varphi'(\bar{x}, y)$:

$$(\exists u \leq y)(\text{LFP}_{y, \bar{z}, S} \psi^{\leq y}(\bar{x}, y, \bar{z}, S(y, \cdot))(\bar{x}, y, u, \dots, u))$$

We see that $\varphi'(\bar{x}, y)$ defines Q' on \mathcal{A} . Fix $n \in \omega$ and $(m_1, \dots, m_k, m) \in n^{k+1}$. Write (\bar{m}, m) for (m_1, \dots, m_k, m) . We have that $(\bar{m}, m) \in Q'(\mathfrak{A}_n)$ if and only if $\bar{m} \in Q(\mathfrak{A}_{m+1})$, if and only if $\mathfrak{A}_{m+1} \models \varphi[\bar{m}]$. Let $\rho(\bar{x}, \bar{z})$ be the formula $\psi^{(m+1)^r}$, the $(m + 1)^r$ -th iterate of ψ . Since $I_\psi(\mathfrak{A}_{m+1}) = I_\psi^{(m+1)^r}(\mathfrak{A}_{m+1})$, we conclude that $\mathfrak{A}_{m+1} \models \varphi[\bar{m}]$ if and only if $\mathfrak{A}_{m+1} \models (\exists u)\rho(\bar{x}, u, \dots, u)[\bar{m}]$. Since m is the maximum in \mathfrak{A}_{m+1} we have that $\mathfrak{A}_{m+1} \models (\exists u)\rho(\bar{x}, u, \dots, u)[\bar{m}]$ if and only if $\mathfrak{A}_{m+1} \models (\exists u \leq y)\rho^{\leq y}(\bar{x}, y, u, \dots, u)[\bar{m}, m]$, and by absoluteness, if and only if $\mathfrak{A}_n \models (\exists u \leq y)\rho^{\leq y}(\bar{x}, y, u, \dots, u)[\bar{m}, m]$. Finally observe that $\rho^{\leq y}(\bar{x}, y, \bar{z})$ is syntactically the same formula as the $(m + 1)^r$ -th iterate of $\psi^{\leq y}(\bar{x}, y, \bar{z}, S(y, \cdot))$. By monotonicity, we get that $\mathfrak{A}_n \models (\exists u \leq y)\rho^{\leq y}(\bar{x}, y, u, \dots, u)[\bar{m}, m]$ if and only if $\mathfrak{A}_n \models (\exists u \leq y)(\text{LFP}_{y, \bar{z}, S} \psi^{\leq y}(\bar{x}, y, \bar{z}, S(y, \cdot))(\bar{x}, y, u, \dots, u))[\bar{m}, m]$ which is exactly what we wanted.

It remains to see that Q' is not first-order definable. Suppose it were via the formula $\varphi'(\bar{x}, y)$. Consider the first-order formula $\varphi(\bar{x})$:

$$(\exists y)((\forall z)(z \leq y) \wedge \varphi'(\bar{x}, y))$$

Fix $n \in \omega$ and $\bar{m} = (m_1, \dots, m_k) \in n^{k+1}$. Since $n - 1$ is the maximum in \mathfrak{A}_n , we have that $\mathfrak{A}_n \models \varphi[\bar{m}]$ if and only if $\mathfrak{A}_n \models \varphi'[\bar{m}, n - 1]$. But $\varphi'(\bar{x}, y)$ defines Q' by assumption

so that $\mathfrak{A}_n \models \varphi'[\bar{m}, n-1]$ if and only if $\bar{m} \in Q(\mathfrak{A}_{n-1+1}) = Q(\mathfrak{A}_n)$. It follows that $\varphi(\bar{x})$ is a first-order formula defining Q on \mathcal{A} which is a contradiction to our assumption. \square

5.3.5 Characterization of the collapse

As a consequence of the previous results, we obtain a characterization of the Ordered Conjecture on the class of finite pure arithmetical structures in terms of a definability question in the standard model of arithmetic.

Corollary 3 *Let $\sigma = ()$ be the empty vocabulary, and let C be the class of all finite arithmetical structures for $(\sigma, \leq, +, \times)$. Then, the following are equivalent:*

- (i) $\text{FO}[C] = \text{LFP}[C]$,
- (ii) every LFP-definable query on C is FO-definable on C ,
- (iii) every LFP-definable absolute query on C is FO-definable on C ,
- (iv) every LFP(Δ_0)-definable absolute query on C is Δ_0 -definable on C ,
- (v) every LFP(Δ_0)-definable relation on $(\omega, \leq, +, \times)$ is Δ_0 -definable on $(\omega, +, \times, \leq)$.

Proof: Observe that C is our \mathcal{A} . We show pairwise equivalence: (i) and (ii) are equivalent by definition; (ii) and (iii) are equivalent by Theorem 16; (iii) and (iv) are equivalent by Lemmas 8 and 9; (iv) and (v) are equivalent by Fact 1, and Lemmas 4 and 7. Thus, they all are equivalent. \square

5.4 Evidence for the conjecture

The Ordered Conjecture on finite pure arithmetical structures has grown in interest after the results of Theorem 13, Theorem 14, and Corollary 3. We may ask how much of the separation can actually be proved. Our next result gives evidence that the separation is true.

Theorem 17 *Let $\sigma = ()$ be the empty vocabulary, and let C be the class of all finite arithmetical structures for $(\sigma, \leq, +, \times)$. If $\text{FO}[C] = \text{LFP}[C]$, then **PH** collapses.*

Proof: By Theorem 14 we know that if $\text{FO}[C] = \text{LFP}[C]$, then **LINH** = **E**. On the other hand, **E** contains complete problems for **EXP** under polynomial-time many-one reductions (see [SC79]). Since **E** = **LINH** \subseteq **PH** \subseteq **EXP**, it follows that **PH** has complete problems under polynomial-time many-one reductions. The collapse of **PH** is immediate (see [BDG96]); the proof is as follows. Let $L \in \mathbf{PH}$ be such a complete problem. Then, for some $k \geq 0$ we have that $L \in \Sigma_k^{\mathbf{P}}$. Now, for every $L' \in \mathbf{PH}$ we have that $L' \leq_m^{\mathbf{P}} L$ because L is complete for **PH** under $\leq_m^{\mathbf{P}}$ -reductions. Since $\Sigma_k^{\mathbf{P}}$ is closed under $\leq_m^{\mathbf{P}}$ -reductions, it follows that $L' \subseteq \Sigma_k^{\mathbf{P}}$; that is, **PH** collapses. \square

Chapter 6

Conclusions

6.1 Overview

Our work has focussed on the Ordered Conjecture of Kolaitis and Vardi. The conjecture remains open. In Chapter 4, we have surveyed known results, including the following: (i) the ordered conjecture is true on the class of all finite ordered structures for any particular vocabulary; (ii) the result of Dawar, Lindell, and Weinstein that the ordered conjecture is true when restricted to unary vocabularies; and (iii) the result of Dawar and Hella that partial fixed-point logic is more expressive than first-order logic on any class of finite ordered structures.

Chapter 5 was dedicated to the important special case that arises if we restrict attention to the class of finite arithmetical structures. We showed that for non-empty vocabularies the conjecture is true, and that for the empty vocabulary, the problem is literally equivalent to an important open problem in complexity theory: the problem of separating **LINH** from

\mathbf{E} , or equivalently, the problem of separating $\mathbf{DLOGTIME}$ -uniform \mathbf{AC}^0 from \mathbf{P} -uniform \mathbf{AC}^0 . Therefore, in a precise technical sense, the Ordered Conjecture is difficult because even restrictions of it are already equivalent to complexity-theoretic open problems. In the same chapter, we presented a different perspective to the problem by showing it to be equivalent to a definability question on the standard model of arithmetic $(\omega, +, \times, \leq)$. Specifically, the problem is equivalent to whether Δ_0 -formulas of arithmetic have the same expressive power as the least fixed-points of positive Δ_0 -formulas. This question links finite model theory to bounded arithmetic [Kra95].

6.2 Directions for future work

The study of the Ordered Conjecture on finite arithmetical structures is interesting in its own right. We have observed in Chapter 5 that the problem is equivalent to the Ordered Conjecture on finite BIT structures, which recently has been connected to *finite set theory* [DDLW98, AK99]. The resolution of this particular case of the conjecture would have complexity-theoretic consequences that we discuss next. As stated earlier, its resolution would imply $\mathbf{LINH} \neq \mathbf{E}$. As proved by Wrathall [Wra78] however, \mathbf{LINH} coincides with the class of rudimentary predicates. Now, let \mathbf{NE} be the non-deterministic counterpart of \mathbf{E} ; that is, $\mathbf{NE} = \bigcup_{c>0} \mathbf{NTIME}(2^{cn})$. It is known that \mathbf{NE} coincides with the class of first-order spectra; that is, the class of sets of cardinalities of finite models of a first-order sentence [JS74]. Therefore, the resolution of the Ordered Conjecture on finite arithmetical structures would imply the separation of the class of the rudimentary predicates, and the

class of spectra, a problem raised by Bennet in 1962. In complexity-theoretic terms,

Problem 1 *Prove that $\mathbf{LINH} \neq \mathbf{NE}$.*

Although the above problem still remains open, a number of related results have been established in the mean time. Paul, Pippenger, Szemerédi, and Trotter [PPST83] proved that the first level of the Linear-time Hierarchy is proper; that is, $\mathbf{LIN} \neq \mathbf{NLIN}$ where $\mathbf{LIN} = \bigcup_{c>0} \mathbf{DTIME}(cn)$ is the class of languages accepted in deterministic linear-time, and $\mathbf{NLIN} = \bigcup_{c>0} \mathbf{NTIME}(cn)$ is the class of languages accepted in non-deterministic linear-time. Wrathall showed that \mathbf{LIN} and \mathbf{NLIN} play a role in the Linear-time Hierarchy that is the linear analogue to the role that play \mathbf{P} and \mathbf{NP} in the Polynomial-time Hierarchy. It is also known that $\mathbf{LINH} \neq \mathbf{EH}$ where $\mathbf{EH} = \bigcup_{c>0} \mathbf{ATIME}(2^{cn}, c)$ is the Linear Exponential-time Hierarchy (this result is implicit in Bennet [Ben62]). Note that the first level of \mathbf{EH} is \mathbf{E} . To mention a few more results, it is known that $\mathbf{NL} \subseteq \mathbf{LINH}$ [Nep70] (see also [Wra78]), and that $\mathbf{LINH} \subseteq \mathbf{Linspace}$ [Myh60] (see also [WW86, Theorem 11.16]), where $\mathbf{NL} = \bigcup_{c>0} \mathbf{NSPACE}(c \log n)$ is the class of languages accepted in non-deterministic logarithmic space, and $\mathbf{Linspace} = \bigcup_{c>0} \mathbf{DSpace}(cn)$ is the class of languages accepted in deterministic linear-time. It is interesting to note that it is known that $\mathbf{LINH} \neq \mathbf{EH}$, but it is open whether $\mathbf{LINH} \neq \mathbf{PH}$. One may speculate that the second separation holds because \mathbf{LINH} may not be closed under polynomial-time many-one reductions (just as $\mathbf{Linspace}$ is not closed under such reductions and so $\mathbf{Linspace} \neq \mathbf{PH}$ [BDG96]). Nevertheless, the separation $\mathbf{LINH} \neq \mathbf{PH}$ would imply that $\mathbf{NL} \neq \mathbf{PH}$ because $\mathbf{NL} \subseteq \mathbf{LINH} \subseteq \mathbf{PH}$. In turn, this would imply that $\mathbf{NL} \neq \mathbf{NP}$

because $\mathbf{NL} \subseteq \mathbf{P} \subseteq \mathbf{NP}$ and $\mathbf{P} = \mathbf{NP}$ implies $\mathbf{P} = \mathbf{PH}$. Note that L. Fortnow [For97] refers to the problem $\mathbf{NL} \neq \mathbf{NP}$ as follows: "... separating these classes might not be nearly as difficult as previously believed, perhaps considerably easier than separating \mathbf{P} from \mathbf{NP} ".

The descriptive complexity perspective brings new insights to these problems. We conclude with the hope that the Ordered Conjecture on arithmetical finite structures, and its equivalent formulation in terms of a definability question in the standard model of arithmetic will help us make progress on these outstanding problems in Computational Complexity Theory.

Appendix A

Alternating Turing machines with random access

A.1 A useful trick

For every $w \in \{0,1\}^*$, we define $\sigma(w)$ as the natural number whose shortest binary representation is $1w$ (a one followed by w). It is clear that σ is a bijection between $\{0,1\}^*$ and $\omega - \{0\}$. The leading one in $1w$ deals with the technical case of leading zeros. Recall that the notation $\log n$ denotes the minimum number of bits needed to represent n . Observe that $|\sigma^{-1}(n)| = \log n$. The following result is well-known, and a proof can be found in Paul [Pau78], for example, although in German.

Lemma 10 *There exists a deterministic Turing machine that when started with two words $w, v \in \{0,1\}^*$ written on separate tapes, halts within $8|w| + |v|$ steps, and accepts if and*

only if the $\sigma(v)$ -th symbol of w from the left is 1.

Proof: The proof is an easy exercise in amortized analysis that we solve as follows. Let T1 and T2 be the tapes on which w and v sit, respectively. Consider the obvious algorithm that consists in keep decreasing by one the counter in T2 while moving the head of T1 to the right. Whenever the counter reaches 0 or the whole word w is scanned, the machine halts. It will accept if and only if the cell scanned by T1 contains a 1.

For the analysis reason as follows. Let $n = |w|$ and $m = \log n$. There are exactly 2^{m-i} words of length m with suffix 10^{i-1} . For each such word, subtracting one requires i steps to find the first one from the right, which will be set to zero, and i steps to return to the right-end of the word while setting all the scanned bits to one. Since this criteria determines a partition of the set $\{0, 1\}^m$, the total number of steps to subtract one to n until zero is reached is bounded by

$$\begin{aligned} T &= 2m \cdot 2^0 + (2m - 1) \cdot 2^1 + \dots + 2 \cdot 2^{m-1} = \sum_{i=0}^m (2i) \cdot 2^{m-i} = \\ &= 2^{m+1} \cdot \sum_{i=0}^m i \cdot 2^{-i} \leq 2^{m+1} \cdot \sum_{i=0}^{\infty} i \cdot 2^{-i} = \\ &= 2^{m+1} \cdot \frac{1/2}{(1 - 1/2)^2} = 2^{m+2} \leq 8n \end{aligned}$$

Since $|v|$ additional steps are needed to put the head of T2 to the right-end of v , the total number of steps is at most $8n + |v| = 8|w| + |v|$ as was to be proved. \square

The same kind of analysis would show that there is a deterministic Turing machine that, when started with $\sigma^{-1}(n)$, the binary representation of n , written on a tape, it halts within $O(n)$ time and with the word 0^n written on a separate tape. Similarly, a

machine exists that, when started with the head of a tape at position n from its left-end, halts within $O(n)$ time with the binary representation of n written on a separate tape. To implement the latter case, the machine keeps incrementing a counter initially set to 0 while moving the head of the tape to the left. Similarly, we can sequentially determine the binary representation of the length of a word of length n in $O(n)$ steps. The analysis are the same as in the proof of Lemma 10. We will make free use of these observations.

A.2 The easy direction

We start with the easy direction; namely, that sequential access machines can be simulated by random access machines.

Lemma 11 *Let t and s be functions. Every alternating Turing machine with sequential input running within time t and s alternations is equivalent to an alternating Turing machine with random access input running within time $O(\max\{t, n\})$ and $O(s)$ alternations. In particular, if $t(n) \geq n$, the time bound is $O(t)$.*

Proof: The simulating machine first determines the length of the input using Lemma 1. Then it writes the word 0^n on a separate tape that will be used as a guide; it can do this using the observation after Lemma 10. The machine existentially guesses n *virtual input* bits and put them in a separate tape that we name VI. Then it universally guesses n *challenge input* bits and put them in a further separate tape named CI. The length of the guesses are guided by the tape containing 0^n . The machine is ready to start a hardware simulation of the original machine using VI instead of the actual input tape. The heads of

tapes VI, CI and the input tape will execute the same moves at all times. The transition function will never depend on the contents of the cell scanned in the input tape. When the cell scanned in CI contains a 1, the machine determines the binary representation of the position of the head in CI using the observation after Lemma 10, and queries the input at that position. If the cell scanned by the virtual input tape coincides with the answer, the machine accepts, and otherwise, it rejects.

We analyze the running time. First, $O(\log n)$ time is spent to determine the length of the input, and $O(n)$ time is spent to write 0^n . Guessing $2n$ virtual and challenge bits requires $O(n)$ steps. After that, $t(n)$ steps are spent to simulate the machine. It remains to estimate the time spent to determine the binary representation of the position scanned by CI when the simulation halts. Since the simulation takes $t(n)$ steps, the head cannot be any further than $t(n)$ cells from the left-end. It follows from the discussion after Lemma 10 that its position can be determined in $O(t)$ time. Overall, the running time is $O(\max\{t, n\})$. The number of alternations is clearly bounded by $s(n) + 2$ which is $O(s)$.

To see that the Turing machines are equivalent, observe that the only virtual inputs that survive are the ones that succeed all the challenges. Every input bit scanned by the simulation will be challenged by at least one challenge sequence. If all of the challenges succeed (are consistent with the input) for a particular virtual input, the simulation will eventually be finished and the simulating machine will accept if and only if the simulated machine does. Since the input is consistent, the lemma follows. \square

Observe that the proof of Lemma 11 shows a stronger result; namely, that the simulating machine can be chosen to have a particular normal form. Its computation consists of two phases: in the first phase, the machine deterministically computes the length of the input in $O(\log n)$ steps using random access; in the second phase, at most one more query is made in each computation path. However, the price we pay for this is that the simulating machine runs in $O(t)$ time as opposed to $t(n)$. We encapsulate this observation in the following:

Lemma 12 *Let t and s be functions such that $t(n) \geq n$ and $s(n) \geq 1$. Every alternating Turing machine with sequential input running within time t and s alternations is equivalent to an alternating Turing machine with random access input running within time $O(t)$ and $O(s)$ alternations. Moreover, the machine can be chosen so that it first determines the length of its input, and then only queries the input once in each path.*

We show next that a similar result holds for random access alternating machines. This partial result will be useful to prove the converse of Lemma 12. Unfortunately not any time-bound is suitable. We need a concept of time-constructibility for sub-linear functions.

A.3 Time-constructible sub-linear functions

To begin with, time-constructibility should be defined in terms of Turing machines with random access. On the other hand, among the several alternative definitions of time-constructibility, we will need to choose the weakest as discussed after the definition.

Definition 9 *Let t be a function such that $t(n) \geq \log n$. We say that t is time-constructible if there exists a deterministic Turing machine with random access input that given a word of length n , halts within $O(t)$ steps with the word $0^{t(n)}$ written on a separate tape.*

The other classical definition of time-constructibility of t is that there exists a Turing machine that halts in exactly $t(n)$ steps on inputs of length n . In this case we say that t is time-constructible in the strong sense. Although the two definitions are provably equivalent for time bounds that are lower bounded by n [Kob85, BDG96], this is not the case for sub-linear time bounds. For example, consider $t(n) = \log n$. An easy information-theoretic argument shows that at least one query of length $1 + \log n$ is necessary to determine the length of the input; this shows that $\log n$ cannot be time-constructible in the strong sense. In contrast, $\log n$ is time-constructible in the sense of Definition 9 because by Lemma 1, we can find the binary representation of the length of the input in $O(\log n)$ steps, and so the word $0^{\log n}$ can be constructed by scanning this binary representation while writing 0's on a separate tape.

Most natural functions are time-constructible in the sense of Definition 9. We already showed that $\log n$ is constructible. Another important example is, of course, $t(n) = n$. It is easily seen to be constructible as follows: the machine first determines the length of the input in $O(\log n)$ steps, and then generates the word 0^n as discussed after Lemma 10 in $O(n)$ time. Overall, the time taken is $O(n)$. We exhibit one more example. Let $t(n) = \log^2 n$. In this case the machine first generates the word $0^{\log n}$ as discussed before, makes a copy of it, and starts scanning one of the copies back and forth while advancing

the head of the other copy after each scan. When the second copy had been scanned completely, the machine would have performed $\log^2 n$ steps; if at each such step, the machine writes a 0 on a separate tape, the construction will be complete.

The composition of time-constructible functions may result in time-constructible functions. This is certainly the case if the outer function is lower bounded by n . For example, if $f(n) \geq n$, we can construct $f(g(n))$ by constructing first $0^{g(n)}$, and then constructing $0^{f(g(n))}$ from it. The running time is $O(\alpha \cdot g(n) + f(g(n)))$ for some constant α , which is $O(f(g(n)))$ because $f(n) \geq n$. We will use this fact later in the text. We have not investigated the time-constructibility of the composition of sub-linear functions. Finally, observe that if t and s are time constructible functions, then $\min\{t(n), s(n)\}$ is also time constructible. To see this, concurrently run the two machines witnessing the time-constructibility of each function and stop as soon as the first one of them stops; the running time is clearly $O(\min\{t(n), s(n)\})$. We will use this observation later.

A.4 The difficult direction

Once we have discussed the issue of time-constructibility, we can proceed to prove the second direction of the equivalence between random access alternating Turing machines and sequential access ones. The following intermediate construction follows ideas due to Buss [Bus87]. Although the techniques are similar, the results of Buss do not deal with non-constant number of alternations and time bounds beyond $O(\log n)$. On the other hand, the construction suggested by Barrington, Immerman, and Straubing preserves

arbitrary time bounds but is not guaranteed to preserve the number of alternations.

Lemma 13 *Let t and s be time-constructible functions such that $t(n) \geq \log n$ and $s(n) \geq 1$. Every alternating Turing machine with random access input running within time t and s alternations is equivalent to an alternating Turing machine with random access input running within time $O(t)$ and $O(s)$ alternations that first determines the length of its input, and then only queries the input once in each path.*

Proof: Let us describe the intuition first, and the details next. The simulating machine executes four distinct phases. In the first phase, it determines the places in which the alternations of the simulation will take place. In the second phase, it determines a computation tree of the simulation according to these alternations. The third phase guesses virtual and challenge bits as in the proof of Lemma 11. The final fourth phase is devoted to a deterministic simulation of the machine through the computation path determined in phase II, and with the virtual input determined in phase III. This phase will also verify whether the challenge is won and whether the alternation sequence is correct. The very first thing the machine is going to do, is to determine the length of its input, and then write down on separate tapes the words $0^{t(n)}$ and $0^{\min\{t(n),s(n)\}}$ where n is the length of the input. To do this, the machine will use the time-constructibility hypothesis and the remarks on the constructibility of $\min\{t(n),s(n)\}$ after Definition 9.

Phase I: Let ALT be the name for a separate work tape. The machine guesses a word from the alphabet $\{\wedge, \vee\}$ of length $t(n)$ indicating at which points the alternations are to be produced. Write this word in ALT. To make sure that no more than $s(n)$ alternations

are made, the machine first writes a sequence of 0's of length $\min\{s(n), t(n)\}$ in a tape named S. Then, during the guessing procedure, the machine keeps moving the head of S to the left whenever an alternation between symbols is produced in ALT. When the left-end of S is reached, the machine stops guessing and deterministically copies the last guessed symbol until the word has length $t(n)$. Once this process is complete, ALT contains a sequence of \wedge 's and \vee 's that contains at most $s(n)$ alternations. Observe that any such sequence is a possible outcome. For this phase, the length of the guesses will be guided by the words $0^{t(n)}$ and $0^{\min\{t(n), s(n)\}}$ previously recorded in separate tapes.

Phase II: The machine is ready to guess a computation path of the simulated machine. To this end, it will start scanning ALT from left to right. At each point, if the symbol scanned in ALT is \wedge , the machine universally guesses a symbol from $\{\swarrow, \searrow\}$; if the symbol scanned in ALT is \vee , the machine existentially guesses a symbol also from $\{\swarrow, \searrow\}$. Write the sequence of guesses into a separate tape named PATH. Intuitively, if the i -th guessed symbol is \swarrow , the simulation will take its left option in its i -th step and conversely for \searrow . This determines a path in the computation tree; we can turn to phase III.

Phase III: The rest of the construction is similar to the previous proof. The machine existentially guesses $t(n)$ *virtual input* bits, and universally guesses $t(n)$ *challenge input* bits. Write them in tapes named VI and CI respectively, and move the heads to their left-ends of the tapes. During this phase, the heads of ALT and PATH are independently moved to their left-end too.

Phase IV: At this point, the machine starts a deterministic hardware simulation of the original machine through the computation path determined by the sequence in PATH, and using the virtual input bits as answers to the queries. The i -th virtual input bit from the left will be used to answer the i -th query (there will be no random look-up; just keep moving the head in VI to the right). Moreover, whenever a query is performed, the machine looks up if the corresponding challenge input bit is set to 1. In that case, the simulation halts and makes a query to check that the virtual input bit coincides with the actual input bit required by the simulation. If so, the machine accepts, and otherwise, it rejects. During the simulation, the machine will scan tape ALT from left to right and check that a state-alternation is produced in the simulation if and only if a symbol-alternation is produced in the tape. If this is not the case or the type of alternation is not correct, the machine rejects. This concludes the construction of the machine.

For the running time reason as follows. From the time-constructibility hypothesis, we know that writing the words $0^{t(n)}$ and $0^{\min\{t(n),s(n)\}}$ requires time $O(t)$. Determining the length of the input is $O(\log n)$ time. In phase I, $O(t)$ time is spent to guess the \vee 's and \wedge 's. In phase II we need to guess a computation path which takes $O(t)$ steps. For phase III we need to guess $t(n)$ virtual bits, $t(n)$ challenge bits, and $t(n)$ more steps are required to move the heads. This is $O(t)$ steps overall. For the last phase, the simulation might take $t(n)$ time and the verification of the challenge does not take any time because the query is already written down, and the head of CI can be moved together with the head of VI. Overall, the running time is $O(t)$ because $t(n) \geq \log n$. For the number of

alternations, the first phase only introduces one alternation, the second phase introduces at most $s(n)$ alternations, the third phase introduces two alternations, and the fourth phase is deterministic. Overall, these are $s(n) + 3$ alternations.

To see that the machines are equivalent, observe as in the proof of Lemma 11, that the only virtual inputs that will survive are the ones that succeed all challenges. In those cases, the virtual input and the actual input are consistent with respect to the simulation, and the machines must be equivalent. However, this will only be so if the sequence of alternations guessed in phase I is consistent with the computation path determined by PATH. The machine takes care of this in phase IV. The proof is complete. \square

The lemma we just proved will be useful to show a converse to Lemma 11. We wish to show that alternating Turing machines with random access are not more powerful than alternating machines with sequential input. Of course, for this to be meaningful we need our running time to be lower-bounded by n .

Lemma 14 *Let t and s be time-constructible functions such that $t(n) \geq n$ and $s(n) \geq 1$. Every alternating Turing machine with random access input running within time t and s alternations is equivalent to an alternating Turing machine with sequential input running within time $O(t)$ and $O(s)$ alternations.*

Proof: Apply Lemma 12 to get an equivalent machine that first determines the length of its input, and then only queries the input once in each path. The simulating machine starts a hardware simulation ignoring the phase to determine the length of the input; instead, it will determine the length by its own, using a counter as in the observations

after Lemma 10. After that, the machine continues the simulation until the only query is performed. Let h be the position queried. Then the machine looks up the h -th symbol of the input tape using the procedure of Lemma 10, and resumes the simulation taking the scanned symbol in the input tape as answer to the query. Before that, we will need to restore h a copy of which will be stored first. We know that no more queries will be performed and the simulation can be completed.

For the running time reason as follows. Determining the length of the input takes $O(n)$ time in a sequential input machine. The simulation, ignoring the treatment of the query, takes $t(n)$ steps. For the query, observe that the binary representation of h has length at most $t(n)$. Therefore, the time to look up the symbol is $O(t(n) + n)$ after Lemma 10. We will require $O(t)$ more steps to save and restore a copy of h . Overall, the running time is $O(t)$ because $t(n) \geq n$. The number of alternations is clearly bounded by $s \in O(s)$. The proof is complete. \square

A.5 The bottom line

These lemmas show that for all purposes, random access machines are an appropriate model when time and alternation bounds are time-constructible and lower bounded by $\log n$ and 1 respectively. We can even assume, as Barrington, Immerman, and Straubing [BIS90] do for example, that the machine first determines the length of the input, and then the input is queried at most once in each path. It is safe to make the following proviso that we will choose to make explicit or not depending on the context.

Proviso 1 Every time bound is assumed to be time-constructible and lower bounded by $\log n$. Every alternation bound is assumed to be time-constructible and lower bounded by 1. In addition, every alternating Turing machine is assumed to be an alternating Turing machine with random access input. We will make free use of the fact that any such machine is equivalent to a machine that first determines the length of its input, and then only queries the input once in each path.

Appendix B

Arithmetic predicates

B.1 Computability of arithmetic predicates

We need the fact that the graphs of addition and multiplication are computable in linear-time and a constant number of alternations when numbers are represented in binary. For addition, the algorithm is straightforward and in fact, deterministic. For multiplication, the reader is referred to Lipton [Lip78, page 198] which makes use of the Chinese Remainder Theorem (see also Wilkie [Wil79, Lemma 3.1]). Lemma 1 also shows that we can decide in linear-time if the m -th bit of n is one when m and n are represented in binary.

B.2 Finite variants

We need a technical lemma. We need to see that the class of queries on finite structures that are definable in first-order logic is closed under finite variants. More precisely, let

C be a class of finite structures for a relational vocabulary $\sigma = (R_1, \dots, R_s)$, let Q_1 and Q_2 be two queries on C such that Q_1 is first-order definable on C , and $Q_2(\mathfrak{A}) = Q_1(\mathfrak{A})$ for all but finitely many $\mathfrak{A} \in C$. We want Q_2 to be first-order definable too. This fact is only true for classes of finite structures, and relies on the fact that every finite structure is definable *up to isomorphism* by a first-order sentence. Essentially, a first-order sentence can define a finite structure $\mathfrak{A} = (A, R_1^{\mathfrak{A}}, \dots, R_s^{\mathfrak{A}})$ by existentially quantifying as many elements as the universe of A , and forcing the relations between these elements to agree with $R_1^{\mathfrak{A}}, \dots, R_s^{\mathfrak{A}}$. Formally, the sentence $\phi_{\mathfrak{A}}$ that defines \mathfrak{A} would be

$$(\exists x_0) \cdots (\exists x_{|A|-1}) \left(\bigwedge_{i \neq j} x_i \neq x_j \wedge \bigwedge_{i=1}^s \bigwedge_{\bar{a} \in A^{r_i}} R_i(x_{a_1}, \dots, x_{a_{r_i}}) \wedge \bigwedge_{\bar{a} \notin A^{r_i}} \neg R_i(x_{a_1}, \dots, x_{a_{r_i}}) \right).$$

It is easy to see that for every finite structure \mathfrak{B} for σ we have that $\mathfrak{B} \models \phi_{\mathfrak{A}}$ if and only if $\mathfrak{B} \cong \mathfrak{A}$. Now Q_2 can be defined easily as follows. We focus on Boolean queries only for simplicity; the generalization to arbitrary arities is straightforward. Let φ be the first-order formula that defines Q_1 by assumption. Let $n_0 \in \omega$ be such that for structures $\mathfrak{A} \in C$ of cardinality at least n_0 we have that $Q_1(\mathfrak{A}) = Q_2(\mathfrak{A})$. Let Q'_2 be the finite set of all structures $\mathfrak{A} \in Q_2$ of cardinality at most n_0 . Then, the following sentence ψ defines Q_2 on C :

$$\bigvee_{\mathfrak{A} \in Q'_2} \phi_{\mathfrak{A}} \vee ((\exists x_1) \dots (\exists x_{n_0}) \left(\bigwedge_{i \neq j} x_i \neq x_j \right) \wedge \varphi)$$

From these observations, we may make free use of the fact that queries on finite structures that are definable in a logic that subsumes FO are closed under finite variants.

B.3 Definability of arithmetic predicates

We first show that the graphs of the addition and multiplication functions are definable in least fixed-point logic from the linear order. Recall that we are assuming that universes are ordinals.

Lemma 15 *Let σ be a relational vocabulary and let C be a class of finite ordered structures for the vocabulary (σ, \leq) . The following two queries are LFP-definable on C : for every $\mathfrak{A} \in C$,*

$$(i) \quad Q_+(\mathfrak{A}) = \{(a, b, c) \in A^3 : a + b = c\},$$

$$(ii) \quad Q_\times(\mathfrak{A}) = \{(a, b, c) \in A^3 : a \cdot b = c\}.$$

Proof: Let $\psi_Z(x)$ be the first-order formula $(\forall x')(x \leq x')$ that defines zero, and let $\psi_S(x', x)$ be the first-order formula $x' \neq x \wedge x' \leq x \wedge (\forall x'')(x'' \leq x' \vee x \leq x'')$ that defines the successor relation in the discrete linear order $\leq^{\mathfrak{A}}$. Let $\varphi_+(x, y, z, X)$ be the following first-order formula for the vocabulary (σ, \leq, X) where X is a ternary relation symbol:

$$(\psi_Z(x) \wedge y = z) \vee (\exists x')(\exists z')(\psi_S(x', x) \wedge \psi_S(z', z) \wedge X(x', y, z'))$$

The least fixed-point formula $\psi_+(x, y, z) = (\text{LFP}_{x,y,z,X}\varphi_+)$ defines the query Q_+ on C .

Let $\varphi_\times(x, y, z, X)$ be the following least fixed-point formula for the vocabulary (σ, \leq, X) where X is a ternary relation symbol:

$$\begin{aligned} & (\psi_Z(x) \wedge z = 0) \vee (\psi_Z(y) \wedge z = 0) \vee (\exists x')(\exists y')(\exists z')(\psi_S(x', x) \wedge \\ & \psi_S(y', y) \wedge X(x', y, z') \wedge \psi_+(y, z', z)) \end{aligned}$$

Again, the least fixed-point formula $\psi_{\times}(x, y, z) = (\text{LFP}_{x,y,z,X}\varphi_{\times})$ defines the query Q_{\times} on C . \square

It is not difficult to see that the positive formulas of the theorem above have a unique fixed-point in finite ordered structures. Essentially, the reason for this is that the structures being finite, their linear order is well-founded, and therefore, one can prove by induction on the maximum \leq -rank of the tuples that the greatest fixed-point collapses to the least fixed-point. As a consequence to this fact we obtain that the graphs of addition and multiplication are uniformly axiomatizable in first-order logic on finite ordered structures. More precisely,

Lemma 16 *Let σ be a relational vocabulary, let S, M be two ternary relation symbols, let C be a class of finite ordered structures for (σ, S, M, \leq) , and let $D = \{\mathfrak{A} \in C : S^{\mathfrak{A}} = Q_+(\mathfrak{A}) \wedge M^{\mathfrak{A}} = Q_{\times}(\mathfrak{A})\}$. Then D is FO-definable on C .*

Proof: Let ϕ be the universal closure of the conjunction of the axioms for the graphs of addition and multiplications; that is, $\phi = (\forall x)(\forall y)(\forall z)(\psi(x, y, z))$ where $\psi(x, y, z)$ is the conjunction of the two formulas below:

$$\begin{aligned} S(x, y, z) &\leftrightarrow ((\psi_Z(x) \wedge y = z) \vee (\exists x')(\exists z')(\psi_S(x', x) \wedge \psi_S(z', z) \wedge S(x', y, z'))) \\ M(x, y, z) &\leftrightarrow ((\psi_Z(x) \wedge z = 0) \vee (\psi_Z(y) \wedge z = 0) \vee (\exists x')(\exists y')(\exists z')(\psi_S(x', x) \wedge \\ &\quad \wedge \psi_S(y', y) \wedge M(x', y, z') \wedge S(y, z', z))) \end{aligned}$$

That is, ϕ says that S and M are fixed-points of the positive formulas φ_+ and φ_{\times} of the previous proof. Since such formulas have a unique fixed-point on finite ordered structures,

the claim follows. \square

We turn next to the question of first-order definability on finite arithmetical structures. We wish to show that a number of natural queries are first-order definable. In particular, the predicate $\text{BIT}(m, n)$ that is true if and only if the m -th bit of the binary representation of n is one, is first-order definable from $+$ and \times . Here, the 0-th bit is meant to be the least significant bit. To prove that, we will need the following technical old result of Bennet [Ben62]. See Section 5.3 for the definition of Δ_0 formulas of arithmetic.

Theorem 18 *The graph of exponentiation is constructive arithmetic; that is, there exists a Δ_0 formula φ such that for every $m, n, p \in \omega$ we have that $(\omega, \leq, +, \times) \models \varphi[p, n, m]$ if and only if $m = n^p$.*

Technically, constructive arithmetic predicates allow constants in their definition. It is not difficult to see that constants can be removed using standard techniques (in the proof of the next result we see how).

Lemma 17 *Let σ be a relational vocabulary and let C be a class of finite arithmetical structures for the vocabulary $(\sigma, \leq, +, \times)$. The following queries are FO-definable on C : for every constant $k \geq 0$, and for every $\mathfrak{A} \in C$ of cardinality n ,*

$$(i) \quad Q_k(\mathfrak{A}) = \{a \in A : a = k\},$$

$$(ii) \quad Q_{\max}(\mathfrak{A}) = \{a \in A : a = n - 1\},$$

$$(iii) \quad Q_{\exp}(\mathfrak{A}) = \{(a, b) \in A^2 : b = 2^a\},$$

$$(iv) \quad Q_{\log n}(\mathfrak{A}) = \{a \in A : a = \log n\},$$

$$(v) Q_{\text{BIT}}(\mathfrak{A}) = \{(a, b) \in A^2 : \text{BIT}(a, b)\},$$

$$(vi) Q_{\text{BIT}_k}(\mathfrak{A}) = \{(a, b_{k-1}, \dots, b_0) \in A^{k+1} : \text{BIT}(a, \sum_{j=0}^{k-1} b_j n^j)\},$$

$$(vii) Q_{\text{BSUM}}(\mathfrak{A}) = \{(a, b_{k-1}, \dots, b_0) \in A^{k+1} : a = \sum_{t=0}^{k-1} \sum_{j=0}^{\log n - 1} (2d_j(b_t) - 1)\},$$

where $d_j(b) \in \{0, 1\}$ and $d_j(b) = 1$ if and only if $\text{BIT}(j, b)$.

Proof: To define $Q_0(\mathfrak{A})$ use the formula $\varphi_0(a) \equiv a + a = a$, to define $Q_1(\mathfrak{A})$ use the formula $\varphi_1(a) \equiv a \times a = a \wedge a + a \neq a$, to define $Q_k(\mathfrak{A})$ use $\varphi_k(a) \equiv (\exists x_1) \cdots (\exists x_{k-1}) (\varphi_1(x_1) \wedge \bigwedge_{i=2}^{k-1} x_{i-1} + x_1 = x_i \wedge x_{k-1} + x_1 = a)$. In the following, we may use constants in our formulas; for example, the formula $x + 2 = y$ means $(\exists x_2) (\varphi_2(x_2) \wedge x + x_2 = y)$. We can do that according to the closure under finite variants. We may also use $+$ and \times as terms when what we really mean is that terms are existentially quantified and forced to their appropriate value: thus, $(x+2)^2 + 1 = y$ means $(\exists z) (\exists t) (x+2 = z \wedge z \times z = t \wedge t+1 = y)$. To define Q_{max} use the formula $\varphi_{\text{max}}(a) \equiv (\forall x) (x \leq a)$. We may also use max as a constant. To define Q_{exp} use the Δ_0 formula of Theorem 18. Namely, if $\varphi(p, n, m)$ is that formula, our formula will be $\varphi(a, 2, b)$. Using the absoluteness of Δ_0 formulas (see Lemma 4), and the fact that Q_{exp} is an absolute query according to Definition 8, the claim follows. To define $\log n$ use the following formula $\varphi_{\log n}(a) \equiv (\exists p) (\exists s) (2^p = s \wedge (\forall q) (p + p \neq q) \wedge ((s + (s - 1) = \text{max} \wedge p + 2 = a) \vee (s + (s - 1) \neq \text{max} \wedge p + 1 = a))$. Recall that $\log n = \lfloor \log_2(n) \rfloor + 1$ to understand this definition. We may use $\log n$ as a constant too. To define Q_{BIT} use the following formula $\varphi(a, b)$:

$$(\exists q) (\exists r) (\exists p) (\exists k) (2^a = p \wedge q \times p + r = b \wedge r < p \wedge 2k + 1 = q),$$

whose meaning is that $b \equiv 1 \pmod{2^a}$. To define Q_{BIT_k} use techniques as in Lindell [Lin92] to uniformly extend the BIT predicate. Namely, Q_{BIT_2} is defined by the formula $\varphi(a, b_1, b_0)$:

$$(a < \log n \wedge \text{BIT}(a, b_0)) \vee (a \geq \log n \wedge (\exists y)(y + \log n = a \wedge \text{BIT}(y, b_1))).$$

The generalization to $k > 2$ is straightforward. For Q_{BSUM} reason as follows. Barrington, Immerman, and Straubing [BIS90, Lemma 8.2] have shown how to define the binary query $a = \sum_{j=0}^{\log n - 1} d_j(b)$ using BIT and \leq (see also [Imm99, Lemma 1.18] and [Lee97, Remark 2.18] for a correction in that proof). Using (v), we can replace BIT by $+$ and \times . Then, our query is defined through the formula $\varphi_{\text{BSUM}}(a, b_{k-1}, \dots, b_0)$:

$$(\exists x_0) \cdots (\exists x_{k-1})(\exists y) \left(\bigwedge_{i=0}^{k-1} x_i = \sum_{j=0}^{\log n - 1} d_j(b_i) \wedge a + k \log n = y \wedge 2(x_0 + \cdots + x_{k-1}) = y \right).$$

The reader will notice that for sufficiently large cardinalities, the witnesses to these existentially quantified variables will exist because $2k \log n \leq n$ for sufficiently large n . \square

For the last result of this appendix, we refer the reader to the book by Immerman [Imm99, Theorem 1.17]. By a finite BIT structure for a vocabulary $(\sigma, \leq, \text{BIT})$ we mean a finite structure in which \leq and BIT are interpreted as the restrictions of the standard linear order and the BIT predicate to its universe (an ordinal).

Lemma 18 *Let σ be a relational vocabulary and let C be a class of finite BIT structures for the vocabulary $(\sigma, \leq, \text{BIT})$. The following queries are FO-definable on C : for every $\mathfrak{A} \in C$ of cardinality n ,*

$$(i) Q_+(\mathfrak{A}) = \{(a, b, c) \in A^3 : a + b = c\},$$

$$(ii) Q_\times(\mathfrak{A}) = \{(a, b, c) \in A^3 : a \cdot b = c\}.$$

The results of this appendix suggest that for all practical purposes, finite arithmetical structures or finite BIT structures are essentially the same classes of structures in what first-order definability concerns.

Bibliography

- [Ajt83] M. Ajtai. Σ_1^1 formulae on finite structures. *Annals of Pure and Applied Logic*, 24:1–48, 1983.
- [AK99] A. Atserias and Ph. G. Kolaitis. First-order logic vs. fixed-point logic in finite set theory. To appear in LICS'99, 1999.
- [AV95] S. Abiteboul and V. Vianu. Computing with first-order logic. *Journal of Computer and System Sciences*, 50(2):309–335, 1995.
- [BCH86] P. W. Beame, S. A. Cook, and H. J. Hoover. Log depth circuits for division and related problems. *SIAM Journal of Computing*, 15(4):994–1003, 1986.
- [BDG96] J. L. Balcázar, J. Díaz, and J. Gabarró. *Structural Complexity I*. Springer-Verlag, second edition, 1996.
- [Ben62] J. H. Bennett. *On Spectra*. PhD thesis, Princeton University, 1962.
- [BES80] L. Babai, P. Erdős, and S. M. Selkow. Random graph isomorphism. *SIAM Journal of Computing*, 9:628–635, 1980.

- [BIS90] D.M. Barrington, N. Immerman, and H. Straubing. On uniformity within NC^1 . *Journal of Computer and System Sciences*, 41(3):274–306, 1990.
- [BS90] R. Boppana and M. Sipser. The complexity of finite functions. In Jan van Leeuwen, editor, *Handbook of Theoretical Computer Science*, volume A. Elsevier, 1990.
- [Bus87] S. R. Buss. The boolean function value problem is in ALOGTIME. In *28th Annual IEEE Symposium on Foundations of Computer Science*, pages 123–131, 1987.
- [CH82] A. Chandra and D. Harel. Structure and complexity of relational queries. *Journal of Computer and System Sciences*, 25:99–128, 1982.
- [Chu36] A. Church. An unsolvable problem of elementary number theory. *American Journal of Mathematics*, 58:354–363, 1936.
- [CKS81] A. K. Chandra, D. C. Kozen, and L. J. Stockmeyer. Alternation. *Journal of the ACM*, 28:114–133, 1981.
- [Coo71] S. Cook. The complexity of theorem proving procedures. In *3rd Annual ACM Symposium on the Theory of Computing*, pages 151–158, 1971.
- [DDLW98] A. Dawar, K. Doets, S. Lindell, and S. Weinstein. Elementary properties of finite ranks. *Mathematical Logic Quarterly*, 44:349–353, 1998.

- [DH95] A. Dawar and L. Hella. The expressive power of finitely many generalized quantifiers. *Information and Computation*, 123:172–184, 1995.
- [DLW96] A. Dawar, S. Lindell, and S. Weinstein. First order logic, fixed point logic and linear order. In *Computer Science Logic '95*, volume 1092 of *Lecture Notes in Computer Science*, pages 161–177. Springer-Verlag, 1996.
- [EF95] H. Ebbinghaus and J. Flum. *Finite Model Theory*. Springer-Verlag, 1995.
- [Fag74] R. Fagin. Generalized first-order spectra and polynomial-time recognizable sets. In R. Karp, editor, *Complexity of Computation*, pages 27–41. SIAM-AMS Proceedings 7, 1974.
- [For97] L. Fortnow. Time-space tradeoffs for satisfiability. In *12th IEEE Conference in Computational Complexity*, pages 52–60, 1997.
- [FSS84] M. Furst, J. Saxe, and M. Sipser. Parity, circuits and the polynomial time hierarchy. *Mathematical Systems Theory*, 17:13–27, 1984.
- [G31] K. Gödel. Über formal unentscheidbare Sätze der Principia Mathematica und verwandter Systeme I. *Monatshefte für Mathematik und Physik*, 38:173–198, 1931.
- [GIS94] Y. Gurevich, N. Immerman, and S. Shelah. McColm's conjecture. In *9th IEEE Symposium on Logic in Computer Science*, pages 10–19, 1994.

- [Gur88] Y. Gurevich. Logic and the challenge of computer science. In E. Börger, editor, *Current Trends in Theoretical Computer Science*, pages 1–57. Computer Science Press, 1988.
- [HKL94] L. Hella, Ph.G. Kolaitis, and K. Luosto. How to define a linear order on finite models. In *9th IEEE Symposium on Logic in Computer Science*, pages 40–49, 1994.
- [IK89] N. Immerman and D. Kozen. Definability with bounded number of bound variables. *Information and Computation*, 83:121–139, 1989.
- [Imm86] N. Immerman. Relational queries computable in polynomial time. *Information and Computation*, 68:86–104, 1986.
- [Imm99] N. Immerman. *Descriptive Complexity*. Springer-Verlag, 1999.
- [JS74] N. D. Jones and A. L. Selman. Turing machines and the spectra of first-order formulas. *Journal of Symbolic Logic*, 39(1):139–150, 1974.
- [Kle36] S. C. Kleene. General recursive functions of natural numbers. *Mathematical Annals*, 112:727–742, 1936.
- [Kob85] K. Kobayashi. On proving time constructibility of functions. *Theoretical Computer Science*, 35:215–225, 1985.
- [Kra95] J. Krajicek. *Bounded arithmetic, propositional logic, and complexity theory*. Cambridge University Press, 1995.

- [KV90] Ph. G. Kolaitis and M. Y. Vardi. 0-1 laws for infinitary logics. In *5th IEEE Symposium on Logic in Computer Science*, pages 156–167, 1990.
- [KV92] Ph. G. Kolaitis and M. Y. Vardi. Fixpoint logic vs. infinitary logic in finite-model theory. In *7th IEEE Symposium on Logic in Computer Science*, pages 46–57, 1992.
- [Lee97] J. Lee. Counting in uniform TC^0 . Technical Report TR97-34, Electronic Colloquium in Computational Complexity, 1997.
- [Lin92] S. Lindell. A purely logical characterization of circuit uniformity. In *7th IEEE Structure in Complexity Theory*, pages 185–192, 1992.
- [Lip78] R. J. Lipton. Model theoretic aspects of computational complexity. In *19th Annual IEEE Symposium on Foundations of Computer Science*, pages 193–200, 1978.
- [Mos74] Y. N. Moschovakis. *Elementary Induction on Abstract Structures*. North-Holland, 1974.
- [Myh60] J. Myhill. Linear bounded automata. Technical Report 60-165, WADO, 1960.
- [Nep70] V. A. Nepomnjascii. Rudimentary interpretations of two-tape turing machines. *Kybernetika*, 2:29–35, 1970. (in Russian).
- [Pau78] W. Paul. *Komplexitätstheorie*. Teubner, 1978.

- [Poi82] B. Poizat. Deux ou trois choses que je sais de L_n . *Journal of Symbolic Logic*, 47:641–658, 1982.
- [PPST83] W. J. Paul, N. Pippenger, E. Szemerédi, and W. T. Trotter. On determinism versus non-determinism and related problems. In *24th Annual IEEE Symposium on Foundations of Computer Science*, pages 429–438, 1983.
- [Ruz88] W. L. Ruzzo. On uniform circuit complexity. *Journal of Computer and System Sciences*, 22:365–383, 1988.
- [SC79] L. J. Stockmeyer and A. K. Chandra. Provably difficult combinatorial games. *SIAM Journal of Computing*, 8(2):151–174, 1979.
- [Sto77] L. Stockmeyer. The polynomial-time hierarchy. *Theoretical Computer Science*, 3:1–22, 1977.
- [Tur36] A. M. Turing. On computable numbers, with an application to the Entscheidungsproblem. *Proceedings of the London Mathematical Society*, 42:230–265, 1936.
- [Var82] M. Vardi. Complexity of relational query languages. In *14th Annual ACM Symposium on the Theory of Computing*, pages 137–146, 1982.
- [Var96] M. Y. Vardi. On the complexity of bounded-variable queries. In *14th ACM Symposium on Principles of Database Systems*, pages 266–276, 1996.

- [Wil79] A. J. Wilkie. Applications of complexity theory to Σ_0 -definability problems in arithmetic. In *Model Theory of Algebra and Arithmetic*, volume 834 of *Lecture Notes in Mathematics*, pages 363–369. Springer-Verlag, 1979.
- [Wra78] C. Wrathall. Rudimentary predicates and relative computation. *SIAM Journal of Computing*, 7(2):194–209, 1978.
- [WW86] K. Wagner and G. Wechsung. *Computational Complexity*. D. Reidel Publishing Company, 1986.