0. Introduction

0.1 Finite Models, Logic and Complexity

Finite model theory deals with the model theory of finite structures. As a branch of model theory it is concerned with the analysis of structural properties in terms of logics. The attention to finite structures is not so much a restriction in scope as a shift in perspective. The main parts of classical model theory (the model theory related to first-order logic) as well as of abstract model theory (the comparative model theory of other logics) almost exclusively concern infinite structures; finite models are disregarded as trivial in some respects and as intractable in others. In fact, the most successful tools of classical model theory fail badly in restriction to finite structures. The compactness theorem in particular, which is one of the corner stones of classical model theory, does not hold in the realm of finite structures. Several examples of other important theorems from classical model theory that are no longer true in the finite case are discussed in [Gur84].

There are on the other hand specific new issues to be considered in the finite. These issues mainly account for the growing interest in finite model theory and promote its development into a theory in its own right. One of the specific issues in a model theory of finite structures is *complexity*. Properties and transformations of finite structures can be considered under *algebraic* and *combinatorial aspects*, under the aspect of *logical definability*, and also under the aspect of *computational complexity*. Issues of computational complexity form one of the main links also between finite model theory and theoretical computer science.

In this introduction I merely intend to indicate selectively some main ideas and lines of research that motivate the present investigations. There are a number of surveys that also cover various other aspects of finite model theory — see for instance [Fag90, Gur84, Gur88, Imm87a, Imm89]. A general reference is the new textbook on finite model theory by Ebbinghaus and Flum [EF95].

0.1.1 Logics for Complexity Classes

The study of the relationship between logical definability and computational complexity of structural properties is an essential branch of finite model the-

ory. This topic is most pronounced in the search for semantic matches between levels of computational complexity and logics. The search for *logics for complexity classes*, also suggestively described as *capturing complexity classes* [Imm87b] or *tailoring logic for complexity* [Gur84], has lead to an active research programme in finite model theory.

Consider any of the standard classes in computational complexity as a class of problems for finite structures, say for finite graphs. The class of all properties of finite graphs that can be recognized by PTIME algorithms is a typical example. A logical characterization of PTIME on finite graphs would have to provide a *logic for* PTIME on graphs in the sense that exactly all PTIME properties of finite graphs are definable by sentences of this logic. For the present purposes we work with a slightly informal notion of a logic for PTIME. The exact definition underlying our treatment is due to Gurevich [Gur88]. A more detailed discussion will be provided with Definition 1.7 in the next chapter. It is also shown in [Gur88] that the restriction to graphs rather than finite structures of arbitrary type is inessential for the present issue.

Definition 0.1 (Sketch). *A logic $\mathcal{L}$ is* a logic for PTIME *if exactly those properties of finite graphs that are* PTIME *recognizable are $\mathcal{L}$-definable.*
Minimal requirements on candidate logics to be imposed are the following: $\mathcal{L}$ has recursive syntax and recursive semantics that associates with each $\mathcal{L}$-sentence a PTIME *algorithm for checking its truth in finite models.*

It is a central open problem in the field whether there is a logic for PTIME.

Relationships between computational complexity classes and definability in logical systems are interesting for a number of reasons:

(a) The potential for theoretical transfer between different fields. Techniques from complexity theory may be brought to bear on logical and model theoretic issues and vice versa. To give an example, several of the outstanding open problems of complexity theory like the PTIME = NPTIME? or PTIME = PSPACE? questions have found appealing non-trivial model theoretic reformulations in terms of semantic equivalences of particular logical systems over finite structures (compare [AV91, DLW95, Daw95b]). Short of solving the original problems this offers new perspectives, and investigations of related logical issues may at least lead to a better understanding of these problems.

(b) Logical analysis of the required kind may yield deeper insights into the fundamental notion of complexity. Definability in logical systems can be viewed as a kind of complexity in itself. Whereas computational complexity controls the computational resources required in the solution of a problem, definability considerations control the logical or descriptive resources required in the specification of the problem: hence the term *descriptive complexity* as used in [Imm89]. The relationship between the

structure imposed by these completely different resources thus becomes part of a broader view of complexity theory.

(c) Exact matches between computational complexity classes and logics provide an appealing notion of *semantic completeness* for model theoretic considerations. A logic for PTIME say would be a logic that is complete for the world of PTIME computability over structures — or for computationally feasible problems, if PTIME computability is identified with efficient solvability or feasibility. The classical classes of computational complexity have emerged as natural levels of computational power, certified by robustness criteria and the existence of natural complete problems. Matching logics constitute naturally distinguished levels of expressiveness.

(d) In this connection there is also a strong theoretical interest from computer science. Problems related to structures like graphs (and more generally arbitrary relational structures corresponding to instantiations of relational databases) are ubiquitous in computer science applications and in particular in the theory of databases. A natural logic for PTIME would be a theoretically ideal database language for exactly all feasible queries: anything that can be specified in this language is guaranteed to have an efficient algorithmic solution; by semantic completeness for PTIME such a logic constitutes a universal language for all efficient tasks. And indeed this context is one of the original sources for the problem of capturing PTIME, as formulated by Chandra and Harel in [CH82].

The following are some of the well known results concerning complete matches between distinguished levels of computational complexity and logical systems.

Regular languages and monadic second-order logic: words over any finite alphabet can in a canonical way be identified with linearly ordered structures over an otherwise monadic vocabulary (one unary predicate for each letter to mark its occurrences in the word). Monadic second-order logic $\mathcal{L}^{II}_{\text{mon}}$ for the resulting *word models* defines exactly the regular languages, i.e. those languages that are recognized by finite automata. This is a classical result of Büchi [Büc60], Elgot [Elg61] and Trakhtenbrot [Tra61] that fits into the present framework as a precursor to the recent development of finite model theory (compare the treatment in [EF95]).

NPTIME and existential second-order logic: Fagin's theorem [Fag74] is the first result of this branch of finite model theory proper. It equates nondeterministic polynomial time recognizability with definability in existential second-order logic Σ^1_1.

PTIME and fixed-point logic with order: in restriction to linearly ordered finite structures PTIME has been characterized logically by Immerman [Imm86] and Vardi [Var82] through the very natural extension of first-order logic to fixed-point logic FP by means of an operator for monotone relational induction.

So there are the following semantic equivalences:

$$\begin{aligned} \text{Finite Automata} &\equiv \mathcal{L}^{II}_{\text{mon}} && \text{(for word models)} \\ \text{NPTIME} &\equiv \Sigma^1_1 \\ \text{PTIME} &\equiv \text{FP} && \text{(in the presence of linear order)} \end{aligned}$$

It is remarkable that all major complexity classes, in particular LOGSPACE, NLOGSPACE, PTIME and PSPACE, are captured by very natural extensions of first-order logic in the presence of order. The fundamental question whether similar matches can be found in the general case of not necessarily ordered structures is open. In particular the question whether there is a logic for PTIME, as raised by Chandra and Harel in [CH82], is a notorious open problem in finite model theory. In fact there is no capturing result at all for any standard complexity class below NPTIME that applies to the general case. Fagin's theorem NPTIME $\equiv \Sigma^1_1$ essentially remains the only general result on a strict match between a complexity class and a logic on finite structures. This phenomenon will be further discussed below.

0.1.2 Semantically Defined Classes

Consider the class of all PTIME recognizable graph properties — for the moment denote it graph-PTIME. It serves as a typical example of a *complexity class on finite structures.*

Why is it difficult to find a logic for graph-PTIME?

Recall that ordinary PTIME is the class of all problems that can be solved by polynomially time bounded Turing machines. A priori Turing machines work with words or strings as inputs. As far as recognition (i.e. decision) problems are concerned a *problem* is a *set of words* over some alphabet. Words over this alphabet are rejected or accepted, according to membership in the set, in time polynomial in their length.

In particular a Turing machine does not work with abstract graphs as inputs but rather with *encodings* of these. The standard encoding scheme for finite graphs uses adjacency matrices for the input representation. The adjacency matrix of a graph whose vertices are labelled $v_1, \ldots, v_n$ is the $n \times n$ boolean matrix with entries $a_{ij} = 0$ or 1 according to whether (v_i, v_j) is an edge. But obviously different adjacency matrices may encode the same, more precisely isomorphic, graphs. Any rearrangement of the vertices in a different order induces an equivalent representation that is different from the given one unless the rearrangement happens to be an automorphism of the abstract graph. Any *graph algorithm*, i.e. any algorithm that recognizes a graph property, must therefore satisfy a non-trivial *semantic invariance condition*: a graph algorithm must produce the same result on any two inputs that represent isomorphic abstract graphs. In other words it may not reject one graph and accept an isomorphic copy of that same graph. We adopt the terminology of complexity theory as presented in [Pap94] to distinguish

semantic presentations and *syntactic presentations* of complexity classes, or semantic and syntactic classes according to their presentation. Semantic presentations are given in terms of semantic constraints on algorithms. Owing to the invariance condition, graph-PTIME is clearly a semantically presented class. A syntactic presentation of a complexity class in contrast consists of a recursive or at least recursively enumerable set of algorithms of the required complexity, that contains at least one realization for every problem in the given class.[1] We shall mostly speak of *recursive presentations* in this sense.

For an example of a class that is not a priori syntactically defined but nevertheless admits a simple recursive presentation consider PTIME in the ordinary sense as a class of problems for words over finite alphabets. It is clearly presentable by the set of algorithms that limit their computation time by means of a step counter that is initialized in each computation to a polynomial in the input size. This presentation is suggestively referred to as through *polynomially clocked machines*.

Semantic invariance conditions like the one for graph-PTIME are non-recursive conditions on algorithms. In fact the set of *all* (syntactic descriptions of) graph algorithms is not even recursively enumerable (as an index set). It can furthermore be shown that the same applies to any of its intersections with standard complexity classes. In particular the ad-hoc presentation of graph-PTIME through the set of all PTIME graph algorithms does not provide a recursively enumerable presentation.

A logic $\mathcal{L}$ for PTIME in the sense of Definition 0.1 above, however, would induce the following recursive presentation for graph-PTIME. Let S be the recursive semantic mapping that associates a PTIME algorithm with each sentence of $\mathcal{L}$ (in the language of graphs). Obviously image(S) consists of PTIME graph algorithms. By semantic completeness of $\mathcal{L}$ for PTIME on graphs, any PTIME graph property is realized by some member of image(S). The recursively enumerable subset image(S) $\subseteq \{\mathcal{A} \mid \mathcal{A}$ a PTIME graph algorithm $\}$ therefore provides a *recursive presentation* for graph-PTIME. In the terminology of complexity theory, image(S) is a *syntactic presentation* of the *semantically defined* class graph-PTIME. In fact it can be shown that there is a logic for PTIME (in the sufficiently general sense of Definition 0.1) if and only if graph-PTIME admits a syntactic, i.e. recursive or recursively enumerable, presentation.

For properties of linearly ordered structures — properties of linearly ordered graphs say — these problems do not arise because there are canonical encodings for ordered structures. For ordered graphs we may use the adjacency matrix based on the natural labelling of the vertices as $v_1, \ldots, v_n$ in increasing order. This observation is easily turned into a recursive presentation for the class of all PTIME properties of ordered finite graphs.

[1] The difference between recursive and recursively enumerable syntax is not important in this kind of question. If $\mathcal{A}_1, \mathcal{A}_2, \ldots$ is a recursive enumeration of syntactic descriptions of algorithms, then the syntax $(\mathcal{A}_1, 1), (\mathcal{A}_2, 2), \ldots$ is recursive.

This crucial difference between the ordered and the unordered case is at the root of the apparent mismatch with respect to capturing complexity classes in the case of ordered structures or in the general case of not necessarily ordered structures. For the standard complexity classes it is almost trivial to see that the induced classes over ordered structures are presentable as syntactic classes and therefore can be captured by logics. The point of the corresponding capturing results indeed rather is that moreover they are captured by very natural logical systems.

As mentioned above, no complexity class below NPTIME has been captured or shown to be recursively presentable in the general case. NPTIME here marks a threshold because in NPTIME and above, the invariance problem can be side-stepped as follows. Consider the class graph-NPTIME of all NPTIME graph properties. From a graph property $Q \in$ graph-NPTIME we may pass to its ordered version $Q_<$, the class of all ordered graphs that possess the given property:

$$Q_< = \{(G,<) \mid G \in Q, < \text{ a linear ordering of the vertices } \}.$$

$Q_<$ is NPTIME recognizable, essentially through the algorithm for Q itself. But a plain graph G belongs to Q if and only if any expansion $(G,<)$ by a linear ordering of its vertices belongs to $Q_<$ (and also if and only if all such expansions belong to $Q_<$). It follows that graph-NPTIME is presentable through the class of all NPTIME algorithms that first guess a linear ordering and then evaluate an NPTIME property of ordered graphs on the result. From this observation we obtain a recursive presentation for graph-NPTIME in a standard manner. It is worth noting that this trick is also directly used in Fagin's proof that graph-NPTIME coincides with the class of all graph properties that are definable in existential second-order logic. The existential quantification over linear orderings $<$ that is implicit in the passage from $Q_<$ to Q is explicitly available in existential second-order logic.

Note that capturing results for complexity classes in the general case of not necessarily ordered structures are not merely of theoretical interest. The challenge is well motivated by the potential applications in database theory. Natural abstract databases often are not ordered. Their realizations at the machine level may involve an implicit linear ordering for representational purposes (naively: a numbering of memory cells). Even though an ordering is present then, it is not considered part of the intended data. A sound database query in this case corresponds to a property of unordered relational structures. In the query specification it is desirable to hide this ordering. Logically, one would have to have a query language corresponding to a logic for PTIME on unordered structures in order to achieve semantic completeness within PTIME and simultaneously to guarantee soundness — soundness in the sense of independence of a linear ordering that is an artifact of the realization.

0.1.3 Which Logics Are Natural?

Consider possible solutions, positive or negative, to the problem whether there is logic for PTIME. The above definition with its very liberal conditions on candidate logics is theoretically appealing because of its connection with recursive presentability. A negative solution in the sense of this definition would be a strong result to the effect that no reasonable logic at all can possibly capture PTIME. A positive result, however, might still leave much to be desired owing to the liberal notion of a logic. In other words, a recursive presentation of graph-PTIME might intuitively be far from constituting a natural logical system. As with the known positive results in the ordered case or for NPTIME much may depend on the style of the logic obtained.

The logics to be considered are extensions of first-order logic, as first-order sentences can be evaluated in LOGSPACE. The systematic study of extensions of first-order logic belongs to the domain of abstract model theory. It is worth to pursue this systematic study with particular focus on logics for finite structures. A systematic study of this kind is a possible approach to problems like that concerning the existence of a logic for PTIME. In particular if one conjectures that the problem of a logic for PTIME has a negative solution, then results that state the impossibility of capturing PTIME by logics that satisfy certain stronger criteria can be interesting approximations.

To some extent the formal framework available in abstract model theory is not necessarily well adapted to the finite case. Complexity considerations and considerations that concern logics under a procedural aspect are not a priori accommodated. The standard formalism in abstract model theory is that of *Lindström extensions* or of extensions by generalized quantifiers (Lindström quantifiers); compare the overview in [Ebb85]. Roughly, each such quantifier incorporates one single new structural property and the resulting extension is a minimal one to make this new property available under some natural closure conditions. While this formalism is universally applicable for many purposes — any extension of first-order logic that satisfies some corresponding closure properties is equivalent with a Lindström extension — it may be argued that it is not always optimally adapted to the demands of finite model theory. It seems that a framework for extensions of logics for finite structures that is sufficiently fine grained to reflect algorithmic constraints is still lacking. This issue is connected with the above-mentioned lack of criteria for the 'naturalness' of a logic for finite structures.

0.2 Natural Levels of Expressiveness

First-order logic is not well adapted to the programme of logics for complexity classes. While any individual finite structure is characterized up to isomorphism by a single sentence of first-order logic, natural properties that are of very low complexity are not first-order definable. For instance neither

connectedness nor regularity are first-order properties of finite graphs. In fact these examples are typical of the two most apparent defects in the expressive power of first-order logic: first-order logic does not provide expressive means to capture any relational process that requires true recursion (like the generation of the transitive closure of the edge predicate required for connectedness), and first-order logic has no means to express non-trivial cardinality properties (like the equality of the numbers of direct neighbours required for regularity). In short, first-order logic lacks *recursion* and *counting*.

0.2.1 Fixed-Point Logics and Their Counting Extensions

The first defect is taken care of in the extension to fixed-point logics. The adjunction of fixed-point operators leads to logics that capture certain levels of relational recursion. Least or inductive fixed-point logic FP in particular is a very natural logic that has been studied extensively. *Inductively defined* and *increasing* relational processes are captured by FP. The generation of the transitive closure is a simple but typical example for the expressive power of FP above that of first-order logic. An important point is that the increasing nature of these relational processes guarantees termination in a stationary value within polynomially many steps. A further extension in terms of relational recursion for arbitrary rather than increasing processes (that therefore may or may not terminate in a stationary value) is partial fixed-point logic PFP. The interest in FP is justified because by the theorem of Immerman and Vardi it captures PTIME for ordered structures. Similarly, PFP captures PSPACE in the presence of order [Var82, AV89]. In particular, in the presence of order, FP and PFP automatically remedy the second shortcoming of first-order logic: on ordered structures FP and PFP also capture all counting and PTIME, respectively PSPACE, cardinality properties of definable predicates. In the absence of order, however, this is not at all true. In the extreme case of pure sets (graphs without edges) it is easy to see that relational recursion and all of FP and PFP collapse to first-order. Simple cardinality properties of the size of sets like evenness of the number of vertices are not definable in FP or PFP. Moreover, all the simple examples of properties that are not FP-definable but may be recognized in PTIME involve such cardinality properties.

One of the themes underlying our present investigations is the attempt to treat these two most apparent shortcomings of first-order logic over finite structures — recursion and counting — on an equal footing and to consider the extensions FP and PFP to a framework that incorporates counting.

We thus obtain *fixed-point logic with counting* FP+C and *partial fixed-point logic with counting* PFP+C. Roughly speaking we deal with two-sorted variants of the given finite structures, augmented by a second ordered arithmetical sort. A link between the sorts is induced by counting terms that associate cardinality values with formulae that define sets. The usual fixed-

point operations can now be applied in this framework to combine relational recursion with the processing of cardinalities.

The conception of fixed-point logic with counting is due to Immerman [Imm87a]. It has not been studied in its own right or even rigorously formalized in the work of Immerman though. The fact that counting is the most obvious defect in FP as compared with PTIME had led Immerman to conjecture that an appropriate extension of FP to FP+C should even be a logic for PTIME in the general case. This conjecture was disproved in a strong sense by Cai, Fürer and Immerman [CFI89]. The sophisticated nature of their example for the separation of FP+C from PTIME indicates on the other hand that FP+C may still be regarded as an interesting level of expressiveness within PTIME that captures many PTIME properties that naturally arise for instance in graph theory. This view has since been corroborated by model theoretic as well as complexity oriented results in [GO93, Ott96a] and we shall see much of this in the sequel. The claim for the naturalness of FP+C mainly rests on the following:

- The expressive power of FP+C can be understood very well in terms of certain FP+C-definable *structural invariants*. An analogous phenomenon was first discovered and exploited in the analysis of FP itself in the work of Abiteboul and Vianu [AV91] and lead to their beautiful result that FP collapses to PFP if and only if PSPACE = PTIME. This approach could successfully be extended to FP+C and PFP+C. In some respects the link between the expressive power of FP+C and PFP+C and the associated invariants is even neater than for FP and PFP themselves. The resulting characterization of the expressive power of FP+C and its relation to PFP+C show that even though FP+C falls short of PTIME it extends to the general case some of the computational and model theoretic features that apply to FP only in the ordered case.
- FP+C and PFP+C are very robust with respect to the actual formalization of the counting extension. There are a number of equivalent characterizations of the expressive power of FP+C, both in terms of logical systems that turn out to be equivalent with FP+C and in computational terms.

Intuitively these show that FP+C constitutes a natural level of expressiveness that at the same time corresponds to some natural level in complexity — even though it is clear that this level is strictly contained in standard PTIME.

0.2.2 The Framework of Infinitary Logic

A different but related approach to the investigation of logics with respect to complexity classes focuses on the a priori logical framework given by certain fragments of infinitary logic. Consider firstly full-fledged infinitary logic $L_{\infty\omega}$, the logic generated by the usual first-order rules for the formation of formulae together with infinite disjunctions and conjunctions over arbitrary sets of

formulae. Now any finite graph is characterized up to isomorphism by a first-order sentence. It follows that *every* property of finite graphs is definable in $L_{\infty\omega}$ by a countable disjunction over first-order sentences that characterize all positive instances of this property. $L_{\infty\omega}$ is a universal logic for finite structures — and overshoots all sensible levels of expressiveness.

Particular fragments of infinitary logic, however, have emerged as very useful tools in finite model theory. These are defined in terms of restrictions on the number of variables that may occur (bound or free). These restrictions are well adapted to the study of relational recursion since the processes considered in relational recursion always involve a fixed bound on the maximal arity of auxiliary relations. Thus pure relational recursion is fully contained within $L^{\omega}_{\infty\omega}$, the fragment of $L_{\infty\omega}$ that consists of all formulae that use a finite number of variable symbols each. In particular fixed-point logic FP and partial fixed-point logic PFP are properly embedded in $L^{\omega}_{\infty\omega}$. It is important to note that the completely non-uniform constructors of infinite disjunctions and conjunctions allow to define non-recursive properties of finite structures as well. $L^{\omega}_{\infty\omega}$, too, is completely at odds with complexity on finite structures. Here this may be seen as an advantage. If we consider problems related to logics for complexity in restriction to the framework of $L^{\omega}_{\infty\omega}$ then it is important that this restriction in itself does not trivialize the issues. Since $L^{\omega}_{\infty\omega}$ allows to define properties of arbitrary complexity, the class of all those PTIME properties of finite graphs, that at the same time are $L^{\omega}_{\infty\omega}$-definable, is a non-trivial subclass of graph-PTIME for our purposes. Furthermore the restriction to bounded arity auxiliary relations can be considered as a natural restriction also in terms of the computational complexity of relational problems.

The points made about the inclusion of counting in connection with FP versus FP+C also apply to the framework of $L^{\omega}_{\infty\omega}$. Evenness of the number of vertices or regularity of graphs are not $L^{\omega}_{\infty\omega}$-definable. The reason is that although each individual expression of the from $\exists^{=m}x\varphi(x)$ asserting the existence of exactly m elements that satisfy φ is in first-order logic, the number of variables required in its formalization grows unboundedly with m. $L^{\omega}_{\infty\omega}$ compensates completely all defects of first-order that concern relational recursion but fails for the defects related to counting. It is natural therefore to study also the fragment $C^{\omega}_{\infty\omega}$ of infinitary logic with only finitely many variables in each formula but allowing all *counting quantifiers* $\exists^{=m}$ instead of the usual existential quantifier. These were also first considered in the work of Immerman on the counting extension of FP. FP+C and PFP+C are comprised in $C^{\omega}_{\infty\omega}$ just as FP and PFP are in $L^{\omega}_{\infty\omega}$. We denote the constituent sublogics with a fixed finite bound k on the number of variables $C^{k}_{\infty\omega}$ and $L^{k}_{\infty\omega}$ so that $C^{\omega}_{\infty\omega} = \bigcup_k C^{k}_{\infty\omega}$ and $L^{\omega}_{\infty\omega} = \bigcup_k L^{k}_{\infty\omega}$. The infinitary logics $C^{\omega}_{\infty\omega}$ and $L^{\omega}_{\infty\omega}$ and their constituents $C^{k}_{\infty\omega}$ and $L^{k}_{\infty\omega}$ will be used extensively as frameworks in our exposition. On the one hand they are used in the analysis of mostly still open restricted problems on capturing complexity classes. On