Logic and Complexity
(Term Paper for cs357, spring 2001.)

Vijay Ganesh. (vganesh@stanford.edu)

June 6, 2001

Abstract

Finite Model theory(FMT), a branch of Model Theory, is the study of the logical properties of finite mathematical structures. Complexity theory is concerned with defining models of computation based on resources such as polynomial time, nondeterministic time, space etc. and the class of languages recognized by these different models of computation. Superficially they do not seem to be linked but researchers have found surprising connections between them. In fact one of the earliest such result is due to Buchï(discussed in class) where he established that the class of languages(regular) accepted by finite automata on finite words is definable in monadic second order logic. We were motivated to explore this connection further and found recent and surprisingly rich literature on this topic. Unfortunately we could not find any one article which discussed all the finite model theoretic tools and techniques useful in characterizing complexity classes nor any comparisons between these tools. The primary goal of this article is to explain and contrast all such techniques, explain why some important tools from Model theory(over both finite and infinite structures) like G*ö*del's Completeness theorem and Compactness theorem for First Order Logic(FOL) do not work in FMT and elaborate on the usefulness of characterizing complexity classes in terms of logic. The techniques that we will discuss include Ehrenfeucht-Fraïssé games (EF games), methods based on spectra, and descriptive complexity.

1 Introduction

Traditionally researchers have studied the complexity of the satisfiability or the model checking problem of certain logics. But characterizing complexity classes in terms of logics has been rarer. One of the first such results was due to Buchï [1, 4] where he established that the class of regular languages is definable in WS1S. The models of WS1S are finite models and generally speaking Finite Model Theory(FMT) has proved to be the area of logic which has deep connection to complexity theory. A plethora of logics over finite structures have been shown to capture complexity classes. Unfortunately many of the important results which hold in (general)model theory do not hold in FMT. For instance, *Gödel's* completeness theorem for FOL does not hold for FOL sentences over finite structures. This follows from the following famous result by Trahtenbrot.

Theorem 1 (Trahtenbrot's Theorem) *We define a vocabulary as a finite non-empty set of relation symbols and constant symbols. Assume that a vocabulary L contains some relation symbol that is not unary. Then the set of first-order sentences over the vocabulary L valid over finite structures is co-r.e. but not r.e.*

A consequence of Theorem 1 is the failure of Completeness Theorem over finite structures. If the Completeness Theorem holds for a set of FOL sentences then the set is r.e. So clearly if the set of sentences under consideration is not r.e., as is the case with FOL sentences over finite structures(by Trahtenbrot's Theorem), then Completeness Theorem does not hold. Similarly the Compactness Theorem which is a very important tool in model theory does not hold in FMT.

But some tools like EF games from model theory do survive into FMT. The most important use of EF games(resp. descriptive complexity) which we shall discuss is to establish non-definability(resp. definability) of a *class of languages* in certain logics. The class of languages so defined by a logic(in fact the encodings of the models of some sentence in that logic) will then be shown to be exactly equal to the languages in some complexity class, thus establishing the connection between the logic and the complexity class under consideration. In spectra(set of cardinalities of the models of first order logic sentence i.e. a set of positive numbers) based techniques we establish that a certain computational model accepts the spectra of some sentence and every set of numbers accepted by that computation model is a spectra. We now give a formal definition of a language and definability of a language. In subsequent sections we will define and discuss in detail EF games, techniques based on spectra and descriptive complexity. Many details of the proofs have been suppressed in favor of intuitive explanations due to lack of space.

Definition 1 *Let* $\mathbf{A}$ *be a finite alphabet and* $\mathbf{A}^*$ *the set of strings over* $\mathbf{A}$*. Then the subsets of* $\mathbf{A}^*$ *are referred to as languages.*

Let $\mathbf{A}$ be the alphabet of the language under consideration. Let $\tau(\mathbf{A})$ be the vocabulary $\{<\} \cup \{P_a \mid a \in \mathbf{A}\}$ of the logic, where $<$ is binary and P_a are unary relations. P_a denotes the set of positions in which the letter $\mathbf{a}$ occurs in a string over the alphabet $\mathbf{A}$. We define finite word models as usual over this vocabulary.

Definition 2 *We now say, A language* $\mathbf{L} \subseteq \mathbf{A}^*$ *is definable in MSO logic, if there is a sentence* ϕ *in MSO logic over the vocabulary* $\tau(\mathbf{A})$ *such that models of* ϕ *are exactly the strings in* $\mathbf{L}$*. Definability in other logics is defined similarly.*

2 EF games

The Connection between FOL sentences and EF games can be intuitively understood as follows. Consider: $\forall x_1 \exists x_2 \forall x_3 \phi(x_1, x_2, x_3)$. View the sentence as a statement about a game with two players, Spoiler and Duplicator, who alternate in picking values for x_1, x_2, x_3. This sentence says that Duplicator can always force a choice of values that make ϕ true. No matter which values Spoiler chooses for x_1, Duplicator can pick an x_2 such that, no matter which x_3 is chosen next by Spoiler, ϕ is true. We can similarily define EF games for logics other than FOL too.

Consider FOL(with equality) over a vocabulary, Sig, with unary relation symbols $Q_1, ..., Q_k$ and the binary relation symbols $R_1, ..., R_l$. Now consider a finite structure over this vocabulary $\mathcal{S} = (\mathrm{S}, \mathrm{Q}_1^S, ..., Q_k^S, R_1^S, ..., R_l^S)$.*Let* $\bar{s} = (s_1, ..., s_n)$ is an n-tuple of elements from S and $\phi(\bar{x})$ is a formula where at most the variables of $\bar{x} = (x_1, ..., x_n)$ occur free, then $(\mathcal{S}, \bar{s}) \models \phi(\bar{x})$ indicates that ϕ holds in $\mathcal{S}$ when interpreting x_i by s_i for i = 1,...,n.

The *quantifier-depth* $qd(\phi)$ of formulas ϕ is the maximal number of nested quantifiers in ϕ. Given $m \geq 0$,

two structures $\mathcal{S}$, $\mathcal{T}$ with universes S, T and designated n-tuples $\bar{s}$, $\bar{t}$ of elements from S,T respectively, are called m-equivalent (written $(\mathcal{S},\bar{s}) \equiv_m (\mathcal{T},\bar{t})$) if $(\mathcal{S},\bar{s}) \models \phi(\bar{x}) \iff (\mathcal{T},\bar{t}) \models \phi(\bar{x})$ for all Sig-formulas $\phi(\bar{x})$ of quantifier depth $\leq$ m. The EF-game allows us to verify the claim $(\mathcal{S},\bar{s}) \equiv_m (\mathcal{T},\bar{t})$). We now introduce some definitions.

Definition 3 *Given Sig-structures $\mathcal{S}$ and $\mathcal{T}$ with universes S and T, we indicate a finite relation $\{(s_1,t_1),...(s_n,t_n)\} \subseteq S \times T$ by $\bar{s} \mapsto \bar{t}$. Such a relation is called a partial isomorphism if the assignment $s_i \mapsto t_i$ determines an injective partial function p from S to T, which moreover preserves all relations Q^S, R^S in the following sense. $s \in Q^S \iff p(s) \in Q^T$ for all symbols Q in Sig and all s in the domain of p. Similarily for R.*

The EF game is played between two players called Spoiler and Duplicator on the structures $(\mathcal{S},\bar{s})$ and $(\mathcal{T},\bar{t})$. There are m rounds(Observe that we are concerned with formulas of quantifier depth $\leq$ m). The initial configuration is the relation $s_i \mapsto t_i$. Given a configuration r, a round is composed of two moves: first Spoiler picks an element s from S or t from T, and then Duplicator reacts by choosing an element in the other structure. The new configuration is $r \cup \{(s,t)\}$. After m rounds, Duplicator has won if the final configuration is a partial isomorphism otherwise the Spoiler has won. We say that Duplicator wins the game $G_m((\mathcal{S},\bar{s}),(\mathcal{T},\bar{t}))$ if Duplicator has a strategy to win each play.

Example 1 *Let u = aabaacaa and v = aacaabaa and consider the game $G_2(\bar{u},\bar{v})$ over the word models for u,v. Assume the Spoiler picks the u-positions with the letters b and c, whence Duplicator can only respond by picking the positions with b and c respectively in v, since it has to preserve the relations Q_b and Q_c. So for $2 \in Q_b^S$ we have p(2) = 5 since $5 \in Q_b^T$. But this implies that the order relation is violated. We know that $(2,3) \in <^S$. But $(p(2),p(3)) \notin <^T$. hence Duplicator can never build a suitable partial isomorphism. Therefore Duplicator always loses.*

How can we in general ascertain that Duplicator wins the game $G_m((\mathcal{S},\bar{s}),(\mathcal{T},\bar{t}))$? One way to do that would be specify a set of partial isomorphisms which would allow the Duplicator to win k rounds ahead. Let $I_m,...,I_0$ be nonempty sets of partial isomorphisms, such that each of them extends $s_i \mapsto t_i$ and for all k=m,...,1 the following two properties hold:

(1) $(\forall p \in I_k)(\forall t \in T)(\exists s \in S)$ such that $p \cup \{(s,t)\} \in I_{k-1}$. **(2)** $(\forall p \in I_k)(\forall s \in S)(\exists t \in T)$ such that $p \cup \{(s,t)\} \in I_{k-1}$.

If a sequence $I_m,...,I_0$ with these properties exists, we write $(\mathcal{S},\bar{s}) \cong_m (\mathcal{T},\bar{t})$. By induction on m we can show that this condition holds iff Duplicator wins.

Theorem 2 (Ehrenfeucht-Fraïssé Theorem) *For $m \geq 0$, $(\mathcal{S},\bar{s}) \equiv_m (\mathcal{T},\bar{t})$ iff $(\mathcal{S},\bar{s}) \cong_m (\mathcal{T},\bar{t})$ iff Duplicator wins $G_m((\mathcal{S},\bar{s}),(\mathcal{T},\bar{t}))$.*

Since we are more interested in seeing this theorem in action we will skip the proof. Following is an example language whose non-definability can be established by using the above theorem.

Example 2 *The language $\{a^n \mid n \text{ is even}\}$ is not FO-definable.*

Proof by contradiction. Consider words defined over the alphabet A = {a}. Consider word models $\mathcal{S}$ = (S,$<^S$,Q_a^S) and $\mathcal{T}$ = (T,$<^T$,Q_a^T). For S=a^i,T=a^j we show that Duplicator wins the game $G_m(a^i,a^j)$ for any i,j $\geq 2^m$ - 1. Each move by the Spoiler decomposes a^i into a^{i1}aa^{i2}. Each corresponding move by the Duplicator decomposes a^j into a^{j1}aa^{j2}. The strategy for the Duplicator is to chose letter-positions such that if k rounds remain, the letter-blocks delimited by positions by the Duplicator are of length $\geq 2^k$-1 or are of same length as determined by Spoiler's choice. This will ensure that Duplicator has a letter to choose for every choice of Spoiler's choice. In this way one sees that Duplicator wins $G_m(a^i,a^j)$ for any $i,j \geq 2^m - 1$. So now choose i=2^m, j=$2^m + 1$. Consider a FOL sentence ϕ with quantifier depth $\leq$ m. Assume this sentence is satisfied in a^{2^m} only. But by the above argument and the Theorem 3 if a FOL sentence is satisfied by a^{2^m} then it has to be satisfied by a^{2^m+1} too. Thus our assumption that ϕ is satisfied only in a^{2^m} is contradicted.

3 Descriptive Complexity

Descriptive complexity theory analyzes the complexity of all *queiries* definable in a logic, the central question being

the following: Given a complexity class $\mathcal{C}$, is there a logic $\mathcal{L}$ such that the queries definable in $\mathcal{L}$ are precisely the queries in $\mathcal{C}$? in other words Given a complexity class $\mathcal{C}$, is there a logic that captures $\mathcal{C}$. A FO-definable query can simply be defined by a first order formula or corresponds to a first order formula. Unfortunately FOL by itself is too weak to capture many complexity classes. So logicians have studied extensions like inflationary fixed point logic FO(IFP), partial fixed-point logic FO(PFP), deterministic transitive closure logic FO(DTC), and transitive closure logic FO(TC). These logics are obtained from FOL by adding operations well-suited to describe iterative and recursive procedures. We will discuss FO(DTC) and see how it captures the complexity class LOGSPACE.

3.1 FO(DTC) logic

Let R be a binary relation on a set M, $R \subseteq M^2$. The deterministic transitive closure DTC(R) is defined by

> DTC(R) := {(a,b) $\in M^2$ | there exist $n > 0$ and $e_0, ..., e_n \in M$ such that $a = e_0, b = e_n$, and for all $i < n$, e_{i+1} is the unique e for which (e_i,e) $\in$ R}.

A FO(DTC) well formed formula(wff) over vocabulary, τ, is constructed in fashion similar to FOL except for the following extra operator .

> $[DTC_{\bar{x},\bar{y}}\phi]\bar{s}\bar{t}$ is a wff where the variables $\bar{x}\bar{y}$ are pairwise distinct and where the tuples $\bar{x}, \bar{y}, \bar{s}$ and $\bar{t}$ are all of the same length, $\bar{s}$ and $\bar{t}$ being tuples of terms.

The meaning of $[DTC_{\bar{x},\bar{y}}\phi]\bar{s}\bar{t}$ is $(\bar{s}, \bar{t}) \in DTC(\{(\bar{x}, \bar{y}) \mid \phi(\bar{x}, \bar{y}, \bar{u})\})$.

Definition 4 *Let K be a class of τ-structures and $\mathcal{L}$ a logic. K is axiomatizable in $\mathcal{L}$, if there is a sentence,ϕ, of $\mathcal{L}$ of vocabulary τ such that K=Models of (ϕ)(written as Mod(ϕ)).*

3.2 Turing Machines and Structures as inputs

We fix the conventions which allow us to regard finite structures as inputs to Turing Machines,TMs. We consider only ordered structures.

3.2.1 Orderings or Ordered Structures

Let $\tau = \{<\}$ be some vocabulary. A τ-structure $\mathcal{A} = (\mathbf{A}, <^{\mathbf{A}})$ is called an ordering if the following hold. for all a, b, c $\in \mathbf{A}$, the relation $<^{\mathbf{A}}$ is non-reflexive, transitive, anti-symmetric and respects trichotomy.

Suppose that τ_0 is a vocabulary with $\{<\} \subseteq \tau_0 \subseteq \{<, S, min, max\}$ and let τ be an arbitrary vocabulary with $\tau_0 \subseteq \tau$. Now a finite τ-structure $\mathcal{A}$ is said to be ordered if the reduct $\mathcal{A}|\tau_0$(i.e. the τ_0-structure obtained from $\mathcal{A}$ by forgetting the interpretations of the symbols in $\tau \setminus \tau_0$) is an ordering. O(τ) is the class of ordered τ-structures. If ψ is a sentence in the vocabulary τ, ordMod(ψ) denotes the class of ordered models of ψ.

3.2.2 Structures as inputs

Suppose $\tau = \tau_0 \cup \tau_1$, say $\tau_1 = \{R_1, ..., R_k, c_1, ..., c_l\}$ and τ_0 as in the preceding subsection. A TM, for τ-structures will have 1+k+l input tapes and m work tapes for some m $\geq$ 1. All tapes are bounded to the left and unbounded to the right and the TM has finite number of states, with a start state, s_0, an accepting state, s_+, and a rejecting state, s_-. With an ordered τ-structure $\mathcal{A}$ we associate the following input inscriptions on the 1+k+l input tapes. The 0-th tape, the "universe tape", contains a sequence of 1's of length n := $\|A\|$. For $1 \leq i \leq k$, the i-th input tape contains the information about R := R_i coded as follows: Suppose R is r-ary, that is, $R^A \subseteq \{0, ..., n-1\}^r$. For $j < n^r$, let $\mid j \mid_r$ be the j-th r-tuple in the lexicographic ordering of $\{0, ..., n-1\}^r$. Then the i-th input tape has a string $a_0 a_1 ... a_{n^r-1}$ where a_j=1 iff $R^A \mid j \mid_r$ and otherwise a_j=0.
For $1 \leq i \leq l$, the (k+i)-th input tape contains the binary representation of j := c_i^A without leading zeros.

3.2.3 TMs which accept structures

Consider the TM described in the previous subsection. values on the tape can be 0,1,leftmost-indicator or end-of-input. Instruction have the form

$$sb_0...b_{k+l}d_1...d_m \longrightarrow s' d_1'...d_m' h_0...h_{k+l+m}$$

which means, If you are in state s, your heads scan tuple of values $b_0...b_{k+l}$ on the input tapes and tuple of values $d_1, ..., d_m$ on the work tapes, replace $d_1, ..., d_m$ by $d_1^{'}, ..., d_m^{'}$, move the i-th head according to h_i and, finally, change to state $s^{'}$.

Definition 5 *Let K be a class of ordered τ-structures. M accepts K if M accepts exactly those ordered τ-structures that lie in K. We say K is in LOGSPACE(written $K \in LOGSPACE$) iff there is a deterministic machine M and $d \geq 1$ such that M accepts K and M is d-log space-bounded.*

Definition 6 *We loosely define a configuration, CONF, as the set of data on the current state, current inscriptions of the work tapes, and the current position of the heads on both the input and the work tapes. An accepting configuration is a configuration with the state s_+. A configuration $CONF^{'}$ is said to be a successor of the configuration CONF, if an instruction of the TM M allows M to pass from CONF to $CONF^{'}$ in one step.*

3.3 Logical Descriptions of Computations

Definition 7 *Let K be a class of ordered τ-structures. We write $K \in DTC$ if K is axiomatizable(refer definition 4) in FO(DTC). We use similar notions for other logics.*

What we have been trying to prove all along is K ∈ LOGSPACE iff K ∈ DTC. In this subsection we will prove the implication from left to right. In the next subsection we prove the other direction. Let C be some complexity class and L be the logic we want to associate with C. Assume K ∈ C, and let M be a TM witnessign that K ∈ C. We will describe the behaviour of M by a formula ϕ_M of L in such a way that for any ordered structure $\mathcal{A}$, $\mathcal{A} \models \phi_M$ iff M accepts $\mathcal{A}$. Recall the definition of K ∈ LOGSPACE. Now, using d FO(DTC) variables, a number independent of n, we can represent the relevant content of a work tape(where n = ‖ A ‖). Moreover, by restricting ourselves to sufficiently large structures $\mathcal{A}$, we can assume that d.logn < n. Hence each head position can be represented by a single number < n. Altogether, we can describe the data of a configuration by a sequence of natural numbers < n of length independent of n, where we agree to use the first number to reprsent the state. We state without proof the following lemma which fomalizes the above argument.

Lemma 1 *Let M be d.log space-bounded. Then there are formulas $\chi_{start}(\bar{x})$ of FOL and $\chi_{succ}(\bar{x}, \bar{x}^{'})$ of FO(DTC) such that for all sufficiently large $\mathcal{A}$ and $\bar{a}$ in A,* ***(a)*** *$\mathcal{A} \models \chi_{start}(\bar{a})$ iff $\bar{a}$ is the starting configuration.* ***(b)*** *For any (d.logn)-bounded configuration $\bar{a}$ and any $\bar{b}$, $\mathcal{A} \models \chi_{succ}(\bar{a}, \bar{b})$ iff $\bar{b}$ is a (d.logn)-bounded successor configuration of $\bar{a}$.*

We now state the theorem for the left to right direction.

Theorem 3 *Let K be a class of ordered structures. If $K \in LOGSPACE$ then K is axiomatizable in FO(DTC) or $K \in DTC$.*

Proof. Let M be a deterministic machine witnessing K ∈ LOGSPACE. Let χ_{start} and χ_{succ} be the formulas corresponding to M according to the preceding lemma. Then by (a) and (b) of the lemma, we have: M accepts $\mathcal{A}$ iff there is a sequence $\bar{a}_0, ..., \bar{a}_k$ of (d.n)-bounded configurations such that $\bar{a}_0$ is the starting configuration, $\bar{a}_{i+1}$ is the successor configuration of $\bar{a}_i$, and $\bar{a}_k$ is an accepting configuration. (observe how we need a deterministic transitive closure operator here to express the sequence of configuration ending with the accepting one(b'cos the sequence of configuration is a DTC operation on the initial configuration). we skip a more formal formulation of this english sentence)

3.4 The Reverse Direction

Theorem 4 *Let K be a class of ordered structures. If $K \in DTC$ then $K \in LOGSPACE$.*

Proof. By induction on the FO(DTC) formula, ϕ, whose models are K. The idea is to construct a TM for atomic FO(DTC) formulae such that models of these formulae are in LOGSPACE and then continue such constructions inductively on FO(DTC) formulae. Observe that both directions of the proof are generalizations(TMs instead of automata and different type of encodings of the models) of the Buchï Theorem discussed in class. For more instances of similar results(like fagin's characterization of NP as EMSO we refer reader to [2]).

Theorem 5 *$K \in LOGSPACE$ iff $K \in DTC$. Similarly $K \in NP$ iff $K \in \Sigma_1^1$(Fagin's Theorem). $K \in NLOGSPACE$ iff $K \in TC$.*

4 Spectra and Complexity

We now turn to results based on the Spectra of a sentence in a certain logic(here FOL with equality). The Spectra of sentence is the set of cardinalities of its models(here finite), a set of positive numbers. We define an automaton which accepts a set of positive numbers iff the set is a spectra. Then we define a time bounded TM which is shown to be equivalent to this automaton. Then we are done. Unfortunately the proofs are too involved and due to space constraints we cannot present them here. A *scene* is a k-dimensional cubical array of symbols from a fixed alphabet. Informally, A *spectrum automaton* is a multihead nonwriting deterministic finite state automaton which scans the vertices of a scene. In a single step, the automaton reads the scene symbols scanned by each of its heads, and uses this information and its control state to decide which way to move each of its heads and what the next control state is to be. Further, the machine is able to sense when it is on the perimeter of the scene, and change state accordingly. An integer, n, is accepted iff there is a scene with sides of length n which is accepted.

Theorem 6 *If M is a spectrum automaton, there is a formula ϕ_M of FOL with equality such that Spectra of ϕ_M = set of integers accepted by M.*

Theorem 7 *If a set $S \subseteq N^+$ is a spectrum, then there is a nondeterministic TM which accepts S in time $O(2^{cx})$, where x is the binary representation of some $n \in S$. (the size of input is the size of the elements of S, not size of S)*

Theorem 8 *If S is accepted by a nondeterministic TM in time $O(2^{cx})$, then S is accepted by a spectrum automaton.*

By putting Theorem 7,8 together we get our result: S is a spectrum of 'FOL with equality' sentence iff S is accepted by a Nondeterministic TM in time $O(2^{cx})$.

5 Conclusions

As we saw, EF-games of a certain logic can be used to charecterize string and graph properties and non-definability of certain complexity classes. They can be used to characterize pumping lemmas and lower bounds[4, 1]. On the other hand descriptive complexity tools are useful to classify a wide range of complexity classes from regular(Buchi) all the way upto NP(Fagin)[2] based on constructing machines which accept structures and suitably encoding the models of defining sentences. Techniques based on Spectra on the other hand are based on finding suitable sentences whose spectra (set of positive numbers) can be accepted by a machine. Spectra based techniques have been shown to have connections to context-sensitive languages and classes between NP and NE[3, 2]. There is a plethora of open problems in this field and we refer the reader to [3, 2]. Characterizing complexity classes using logic is valuable because it adds more tools and new perspectives in understanding and solving the open problems in complexity theory. For instance, recently Immerman et al. used descriptive complexity to show the surprising result that nondeterministic spaces are closed under complement.

References

[1] H.D. Ebbinghaus and J. Flum. *Finite Model Theory*. Perspectives in Mathematical Logic. Springer Verlag, 1st edition, May 1995.

[2] Ronald Fagin. Finite model theory– a personal perspective. *Theoretical Computer Science*, (113):3–31, 1993.

[3] N.D. Jones and A.L. Selman. Turing machines and the spectra of first-order formulas with equality. *Journal of Symbolic Logic*, pages 139–150, 1974.

[4] Wolfgang Thomas. Languages, automata, and logic. Technical Report 9607, Institut *für* Informatik und Praktische Mathematik, May 1996.