MATHEMATICAL LOGIC FOR COMPUTER SCIENCE
(A.Y. 20/21)

HANDOUT N. 10

ABSTRACT. We prove Fagin's Theorem: the **NP** properties of finite structures are exactly those definable by a formula with existential quantifiers on relations.

1. Logical characterization of Complexity classes

We used the method of describing computation by formulas to obtain undecidability results (for finite and general validity). The same method can be used to obtain a logical characterization of known complexity classes.

What is meant by "logical characterization"? The idea is to show a strict correspondence between the fact that some problem S is in some complexity class C (*algorithmic complexity*) and the fact that S can be described/expressed/defined by a formula F in some logical language L (*descriptive complexity*).

FIGURE 1. Ronald Fagin (1945-)

In particular, we can obtain a logical characterization of the class **NP** of non-deterministically polynomial time decidable problems. The result is due to Ronald Fagin and is a milestone of so-called Descriptive Complexity Theory.

The proof is very similar to the proof of Church's and Trakhtenbrot's Theorem yet we have to use a different logic rather than first-order logic. The type of logic that we need will come out of the proof itself. Before going into the technical details it is useful to give a rough intuitive description.

Recall that a language/problem is in **NP** if it is accepted by a polynomial time non-deterministic Turing machine (i.e., a Turing machine in which the transition δ is a relation rather than a function).

We can suitably identify **problems** dealt with in Complexity Theory as **properties of finite structures**. For example, the 3-Colorability problem on graphs can be identified with the subclass of the class of finite graphs that happen to be 3-colorable. We can then deal with algorithms for deciding properties of finite relational structures.

But how can we give a finite structure as input to a Turing Machine/program? We need to set up a **coding of finite relational structures**. This coding naturally requires the use of **a linear ordering of the domain**, since the domain of a structure is, in general, just a set, and we need an ordering to identify and access the elements. We will thus associate a symbolic or numerical code to any pair $(\mathfrak{A}, <^\bullet)$ where $\mathfrak{A}$ is a finite structure for some language and $<^\bullet$ is a linear ordering of the domain of $\mathfrak{A}$.

Notes by Lorenzo Carlucci, carlucci@di.uniroma1.it. Most of the material of this handout is taken from E. Grädel's lecture notes on *Algorithmic Model Theory* and chapter *Finite Model Theory and Descriptive Complexity* in Grädel et al., *Finite Model Theory and its Applications*, Springer 2007.

Naturally we want our decision algorithms to be *independent* of the ordering: we don't want our decision procedure to say that a graph G is 3-colorable if presented with some ordering $<_1$ but not 3-colorable if presented with some other ordering $<_2$. Therefore we will say that an algorithm/machine M **decides** a problem/class of finite structures if it decides the set of all codes of structures in the class, *relative to any linear ordering of their domain.*

Given a non-deterministic polytime TM M with time bound $T(x) = x^k$ we wish to write a **first-order formula** $G_{M,k}$ such that when $G_{M,k}$ is evaluated in a structure $\mathfrak{A}$ (with linear ordering $<^\bullet$) it expresses the existence of an accepting computation by M on input $(\mathfrak{A}, <^\bullet)$ in time n^k with $n = |A|$.

We know in part from previous proofs how to write such a formula (but we have to deal with relating the structure on which we check whether the fomula is satisfiable to the input on which the formula "mimicks" the computation of M).

The formula $G_{M,k}$ will contain, among other, some predicate symbols used to describe the computation, and the symbol $<$. Let the relation symbols used in $G_{M,k}$ to describe the computation be $R_1, \ldots, R_m$.

We wish to obtain roughly the following behaviour:

(1) If **there exist relations** $R_1, \ldots, R_m$ and $<^\bullet$ such that the structure $(\mathfrak{A}, <^\bullet, R_1^\bullet, \ldots, R_m^\bullet)$ satisfies $G_{M,k}$ then the relations $R_i^\bullet$ contain a representation of an accepting computation of M on input $(\mathfrak{A}, <^\bullet)$; and viceversa,

(2) If **there exist relations** $R_1^\bullet, \ldots, R_m^\bullet$ contain a representation of an accepting computation of M on input $(\mathfrak{A}, <^\bullet)$ then the structure $(\mathfrak{A}, <^\bullet, R_1^\bullet, \ldots, R_m^\bullet)$ satisfies the formula $G_{M,k}$.

Obviously the meaning of "representing a computation" needs to be made precise. Since we are dealing with non-deterministic computation, machine M can have many different computation paths on the same input $\mathfrak{A}$. This means that we want to deal with different possible interpretations not only of the usual first-order variables but also of the predicate symbols: each computation on the input structure $\mathfrak{A}$ determines a different interpretation $R_1^\bullet, \ldots, R_m^\bullet$ of the symbols $R_1, \ldots, R_m$ describing the computation.

We are "some existentials away" from the desired result. Thus, all we need is to allow the use of *existential quantifiers on relation symbols.* We could then consider the sentence F_M: $\exists R_1, \ldots, \exists R_m \exists < G_{M,k}$, with the intuitive interpretation. This sentence will satisfy:

M accepts with input $(\mathfrak{A}, <)$ if and only if $\mathfrak{A}$ satisfies F_M, where "satisfies" is not the usual first-order satisfaction relation but its natural extension to formulas with existential quantifiers on relations.

We are thus lead to a logic whose syntax and semantics exactly allows to express the above: what we need is (existential) quantification on relation symbols.

2. (Existential) Second-Order Logic

The logic suggested by the above discussion is called *Existential Second-Order Logic* (the expression "second order logic" was introduced by the logician and philosopher C.S. Peirce. Systems of Second Order Logic were used early in the history of modern logic by Gottlob Frege, among others).

The basic idea is to allows existential quantifiers on relational symbols of finite arity in addition to usual first-order quantifiers. A quantifier $\exists R$ where R is an n-ary relational symbol is intended to range, given a structure $\mathfrak{A}$, over subsets of A^n. The semantics is the intuitive one. We say that $\mathfrak{A} \models \exists R_1 \ldots \exists R_k F$ if and only if there exist relations $R_1^\bullet \subseteq A^{d_1}$ ..., $R_k^\bullet \subseteq A^{d_k}$, where the d_is are the arities of the symbols R_is, such that $(\mathfrak{A}, R_1^\bullet, \ldots, R_k^\bullet) \models F$. In the latter expression we are denoting by $(\mathfrak{A}, R_1^\bullet, \ldots, R_k^\bullet)$ the structure that is identical to $\mathfrak{A}$ and interprets the relational symbol R_i as $R_i^{\mathfrak{A}}$.

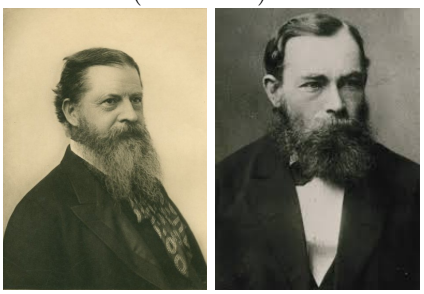
This fragment of second-order logic is called Existential Second-Order logic. Existential Second Order formulas (also called Σ_1^1 formulas in the literature) are formulas of the form $\exists R_1 \ldots \exists R_k F$, where $R_1, \ldots, R_k$ are relation symbols (second-order variables) and F is first-order.

It is easy to observe that this logic is more expressive than usual first-order logic. In particular it allows to define some well-known **NP** problems.

The following sentence expresses k-colorability:

$$\exists C_1 \ldots \exists C_k (\forall x (\bigvee C_i(x)) \wedge \forall x (\bigwedge \neg (C_i(x) \wedge C_j(x)) \wedge \forall x \forall y (E(x,y) \rightarrow \bigwedge \neg (C_i(x) \wedge C_i(y))))$$

FIGURE 2. Charles S. Peirce (1839-1914) and Gottlob Frege (1848-1925)



Let $\exists C_1 \dots \exists C_k F$ denote the above formula with F its first-order part. For any graph $\mathcal{G} = (V, E)$ we have that $\mathcal{G} \models \exists C_1 \dots \exists C_k F$ iff there exists color classes $C_1^{\mathcal{G}}, \dots, C_k^{\mathcal{G}} \subseteq V$ such that

$$(\mathcal{G}, C_1^{\mathcal{G}}, \dots, C_k^{\mathcal{G}}) \models F$$

iff $\mathcal{G}$ is k-colorable.

We will prove that if a property of finite structures is decidable by an **NP** algorithm, then it can be defined by a formula of Existential Second Order Logic. We will in fact also show that the viceversa holds. In this sense we can say that Existential Second Order Logic completely *captures* the complexity class **NP**.

Before proving the result we need to set up a coding that enables us to present finite relational structures as inputs to algorithms.

3. Decision problems on finite structures

We first sketch how to encode finite relational structures in order to give them as inputs to Turing machines. To encode a finite structure we exploit the opportunity to define a linear order on its domain.

An example of a coding with good properties is the following. Let $\mathfrak{A}$ be a structure for the language $\{R_1, \dots, R_t\}$ with domain $A = \{a_1, \dots, a_n\}$. Let $<^\bullet$ be a linear order on A. For the discussion let's fix $a_1 <^\bullet a_2 <^\bullet \cdots <^\bullet a_n$.

Let R_i be of arity d. We can encode it as a binary string of length n^d as follows: consider the set of d-tuples in A ordered by the lexicographic ordering inherited by $<^\bullet$. We can code R_i by the binary string of length n^d that has a 1 in position j if and only if the jth tuple in the above enumeration of A^k is in R_i.

We can code the structure $\mathfrak{A}$, relative to the ordering $<^\bullet$, by concatenating the codes of $R_1, \dots, R_t$.

Since in some cases it is useful that the code contains further information on the structure, we can add the prefix $1^n 0^{n^m - n}$ where m is the maximum of the arities of $R_1, \dots, R_t$.

We can furthermore uniformize the length of the codes of the relations by adding the suffix $0^{n^m - n^d}$ to the code of R_i where d is the arity of R_i.

From the code we can read-off the cardinality of the domain and the maximum arity of relations. The code thus defined has other good properties, in particular: the coding map is first-order definable (see below for detail), the size of the code is polynomial in the dimension of the domain of the structure and the encoding allows to calculate efficiently the truth-values of the atomic formulas in the structure.

Now we can define what it means that an algorithm/machine decides a property of finite structures. As usual we are interested in classes that are closed under isomorphisms.

The inputs of the algorithms we consider are encodings of a pair $(\mathfrak{A}, <^\bullet)$, where $\mathfrak{A}$ is a finite structure for language $\mathcal{L}$ and $<^\bullet$ is a linear order on A. The algorithms that we consider only depend on the structure and not on the linear order chosen for coding, i.e. for each $<^\bullet$ and $<^\star$ linear orders over A, the output of the algorithm is the same on $(\mathfrak{A}, <^\bullet)$ and on $(\mathfrak{A}, <^\star)$.

Definition 3.1. An algorithm *decides* a class $\mathcal{C}$ of finite structures (for a language $\mathcal{L}$) if it decides the set of codes of the pairs $(\mathfrak{A}, <^\bullet)$ with $\mathfrak{A} \in \mathcal{C}$ and $<^\bullet$ any linear order on the domain A of $\mathfrak{A}$.

4. Fagin's Theorem

We always consider properties (i.e. classes) of structures that are invariant (i.e. closed) under isomorphism.

Theorem 4.1 (Fagin's Theorem)**.** *Let $\mathcal{L}$ be finite relational language.*

If F is an Existential Second-Order formula in $\mathcal{L}$ then the class of finite structures for $\mathcal{L}$ that satisfy F is in **NP***.*

If $\mathcal{C}$ is a property of finite structures for $\mathcal{L}$ and is in **NP***, then it is expressible/axiomatizable by an Existential Second-Order formula in $\mathcal{L}$.*

Proof. Let $F = \exists R_1 \dots \exists R_m G$ be an Existential Second Order formula. We want to design a non-deterministic algorithm that, on input $\mathfrak{A}$ (a finite relational structure for the language of G), decides in polytime whether it satisfies F or not.

Recall that if we are given a structure $\mathfrak{A}^\bullet = (A, R_1^\bullet, \dots, R_m^\bullet)$ we can check in polynomial time if $\mathfrak{A}^\bullet \models G$. Let $\mathfrak{A}$ be the input. Assume that $A = [0, n-1]$. Choose non-deterministically m relations $R_1^\bullet, \dots, R_m^\bullet$ on A of the appropriate size. In other words we determine non-deterministically m sets of binary strings on A of appropriate length. In **LOGSPACE** you can decide if $(\mathfrak{A}, R_1^\bullet, \dots, R_m^\bullet) \models G$.

We prove now the other implication. Let $\mathcal{C}$ be a class (property) of structures for the language $\mathcal{L}$. We always assume that the class/property is closed under isomorphism.

Let M be a non-deterministic Turing machine that decides if $\mathfrak{A} \in \mathcal{C}$ in polynomial time. We define an Existential Second Order formula F_M whose finite models are the structures in $\mathcal{C}$. I.e. F_M is such that for each finite structure $\mathfrak{A}$ for $\mathcal{L}$, for any linear order $<^\bullet$ on A,

$$\mathfrak{A} \models F_M \quad \Leftrightarrow \quad M \text{ accepts } (\mathfrak{A}, <^\bullet).$$

Recall we are assuming that the answer of M does not depend on the choice of the linear order $<^\bullet$, i.e. that M accepts $(\mathfrak{A}, <^\bullet)$ for some linear order $<^\bullet$ on A if and only if M accepts $(\mathfrak{A}, <^\bullet)$ for every linear order $<^\bullet$ on A.

Let $M = (Q, \Sigma, q_0, F^a, F^r, \delta)$, $\delta : (Q \times \Sigma) \to \mathcal{P}(Q \times \Sigma \times \{-1, 0, 1\})$ a non-deterministic Turing machine such that for each finite structure $\mathfrak{A}$ for $\mathcal{L}$ and for every linear order $<^\bullet$ on A, M on input $(\mathfrak{A}, <^\bullet)$ decides in polynomial time if $\mathfrak{A} \in \mathcal{C}$. Let $Q = \{q_0, \dots, q_u\}$ and $\Sigma = \{\sigma_0, \dots, \sigma_v\}$.

Let k be such that each computation of M on $\mathfrak{A}$ reaches an accepting or rejecting state within $|A|^k - 1$ steps of computation. We can assume, for technical convenience, that M always performs exactly $|A|^k - 1$ steps.

We use a tuple of relations $\vec{X}$ to represent a computation of M and define a first-order formula G_M in the language $\mathcal{L} \cup \{<\} \cup \{X_1, \dots, X_N\}$ such that

$$(\mathfrak{A}, <^\bullet, X_1^\bullet, \dots, X_N^\bullet) \models G_M \quad \Leftrightarrow \quad (X_1^\bullet, \dots, X_N^\bullet) \text{ represents an accepting computation of } M \text{ on } (\mathfrak{A}, <).$$

Let $LO_<$ be a statement that expresses in first-order that $<$ is a linear order. Then

$$F_M := \exists < \exists X_1 \dots \exists X_N (LO_< \wedge G_M)$$

satisfies the theorem.

We use the following language to represent the computation.

- Represent the numbers in $[0, n^k - 1]$ as tuples in A^k ordered according to the lexicographic ordering inherited by $<^\bullet$. This allows to represent all the parameters of time and space of the computation. Accordingly, we will use vectors of length k as numbers to refer to time instants and cell positions during the computation. On the formula side this will correspond to using vectors of k variables $(x_1, \dots, x_k)$ (also denoted $\vec{x}$). For a k-ary vector $\vec{a} \in A^k$ we say "the $\vec{a}$th number" to mean the jth number where j is the position of $\vec{a}$ in the lexicographic ordering on A inherited by $<^\bullet$.

- Represent a state $q \in Q$ by a k-ary predicate $S_q(\cdot)$, whose intended interpretation is the set of time parameters $\vec{t}$ (a k-ary vector in A) such that at the $\vec{t}$th-time instant M is in state q.
- Represent a symbol $\sigma \in \Sigma$ with a $2k$-ary predicate $T_\sigma(\cdot,\cdot)$, whose intended interpretation is the set of pairs of time and cell parameters $(\vec{t},\vec{a})$ such that at time $\vec{t}$ in cell number $\vec{a}$ is the symbol σ.
- Represent the position of the head of M with a $2k$-ary predicate $H(\cdot,\cdot)$ whose intended interpretation is the set of pairs of time and cell parameters $(\vec{t},\vec{a})$ such that at time $\vec{t}$ the head of M is on the cell number $\vec{a}$.

(The predicates S_q, T_σ, H are the relations symbols $X_1,\ldots,X_N$ used in the proof sketch above).

Remark 4.2. The minimum, maximum and the successor relation induced by the linear order are first-order definable. Also the successor relation induced on k-ples, (denoted by $\vec{y} = \vec{x}+1$) that holds if $\vec{y}$ is the k-ple in A representing the number $n+1$ where n is the number represented by the k-ple $\vec{x}$ in A) is definable by an open formula in a language that is equipped with a binary relation S and constants 0, max for the first and the last element of the domain. For each fixed m, the relation that holds of $(\vec{x},\vec{y})$ if $\vec{y}$ denotes the m-the successor of the number denoted by $\vec{x}$ is similarly definable. We denote this relation by writing $\vec{y} = \vec{x}+m$.

Remark 4.3. We note that the encoding is such that for every k, for every $\sigma \in \Sigma$, there is a first-order formula $\beta_\sigma(x_1,\ldots,x_k)$ in the language $\mathcal{L} \cup \{<\}$ such that for each structure $\mathfrak{A}$ and every linear order $<^\bullet$ on $\mathfrak{A}$, and for every k-ple $\vec{a}$ of elements of A,

$$(\mathfrak{A},<^\bullet) \models \beta_\sigma(x_1,\ldots,x_k)[a_1,\ldots,a_k] \Leftrightarrow \text{ symbol number } \vec{a} \text{ in the code of } (\mathfrak{A},<^\bullet) \text{ is } \sigma.$$

We define the **first-order** formula $G_{M,k}$ as the universal closure of $START \wedge STEP \wedge END$.

$$START := S_{q_0}(\vec{0}) \wedge \bigwedge_{q_i \neq q_0} \neg S_{q_i}(\vec{0}) \wedge H(\vec{0},\vec{0}) \wedge H(\vec{0},\vec{s}) \rightarrow s = \vec{0} \wedge \bigwedge_{\sigma\in\Sigma} (\beta_\sigma(x_1,\ldots,x_k) \rightarrow T_\sigma(\vec{0},x_1,\ldots,x_k)),$$

$$STEP := NOCHANGE \wedge CHANGE$$

$$NOCHANGE := \bigwedge_{\sigma\in\Sigma} (T_\sigma(\vec{t},\vec{x}) \wedge H(\vec{t},\vec{y}) \wedge \vec{y} \neq \vec{x} \wedge \vec{t'} = \vec{t}+1 \rightarrow T_\sigma(\vec{t'},\vec{x}))$$

$$CHANGE := \bigwedge_{q\in Q,\sigma\in\Sigma} (PRE_{(q,\sigma)} \rightarrow \bigvee_{(q',\sigma',m)\in\delta(q,\sigma)} POST_{(q',\sigma',m)})$$

$$PRE_{(q,\sigma)} := S_q(\vec{t}) \wedge H(\vec{t},\vec{x}) \wedge T_\sigma(\vec{t},\vec{x})$$

$$POST_{(q',\sigma',m)} := S_{q'}(\vec{t'}) \wedge T_{\sigma'}(\vec{t'},\vec{x}) \wedge \exists\vec{y}(\vec{x}+m = \vec{y} \wedge H(\vec{t'},\vec{y}))$$

$$END := \bigwedge_{q\in F^r} \neg S_q(\vec{t})$$

The prefix of quantifiers is $\forall\vec{s}\forall\vec{x}\forall\vec{y}\forall\vec{t}\forall\vec{t'}$.

We verify in the next two Lemmas that the formula has the desired properties.

Lemma 4.4. *If M accepts $(\mathfrak{A},<^\bullet)$, then $(\mathfrak{A},<^\bullet) \models \exists S_{q_0}\ldots\exists S_{q_u}\exists T_{\sigma_0}\ldots\exists T_{\sigma_v}\exists H G_M$.*

Proof. By construction: from an accepting computation of M on $(\mathfrak{A},<^\bullet)$ define appropriate interpretations for the relation symbols $S_{q_0},\ldots,S_{q_u},T_{\sigma_0},\ldots,T_{\sigma_v},H$. □

Lemma 4.5. *If $(\mathfrak{A},<^\bullet,S^\bullet_{q_0},\ldots,S^\bullet_{q_u},T^\bullet_{\sigma_0},\ldots,T^\bullet_{\sigma_v},H^\bullet) \models G_M$, then M accepts $(\mathfrak{A},<^\bullet)$.*

Proof. This is the usual forcing argument arguing from a model of the formula to properties of the real computation.

We need as usual a formula to express that a configuration C holds at time j:

$$F_{C,j} := S_q(j) \wedge H(j,p) \wedge \bigwedge_{i=0}^{n^k-1} T_{b_i}(j,i),$$

where $C = (q, p, b_0 \dots b_{n^k-1})$ is a configuration (the b_is are alphabet symbols in Σ).

We have the following points.

- If $(\mathfrak{A}, <^\bullet, S^\bullet_{q_0}, \dots, S^\bullet_{q_u}, T^\bullet_{\sigma_0}, \dots, T^\bullet_{\sigma_v}, H^\bullet) \models START$ then $(\mathfrak{A}, <, S^\bullet_{q_0}, \dots, S^\bullet_{q_u}, T^\bullet_{\sigma_0}, \dots, T^\bullet_{\sigma_v}, H^\bullet)$ models the formula describing the initial configuration of M with input $(\mathfrak{A}, <^\bullet)$. Obviously this is also the situation at the beginning of the real computation.
- If $(\mathfrak{A}, <^\bullet, S^\bullet_{q_0}, \dots, S^\bullet_{q_u}, T^\bullet_{\sigma_0}, \dots, T^\bullet_{\sigma_v}, H^\bullet) \models STEP$ then for every non-final configuration C , if
$$(\mathfrak{A}, <^\bullet, S^\bullet_{q_0}, \dots, S^\bullet_{q_u}, T^\bullet_{\sigma_0}, \dots, T^\bullet_{\sigma_v}, H^\bullet) \models F_{C,t}$$
with $t < n^k - 1$, then there exists a configuration C' that follows C according to M such that
$$(\mathfrak{A}, <^\bullet, S^\bullet_{q_0}, \dots, S^\bullet_{q_u}, T^\bullet_{\sigma_0}, \dots, T^\bullet_{\sigma_v}, H^\bullet) \models F_{C',t+1}.$$
In other words
$$G_M, F_{C,t} \models \bigvee_{C' \in Next(C)} F_{C',t+1}.$$
Therefore there exists a computation of M on $(\mathfrak{A}, <)$, $C_0, C_1, \dots, C_{n^k-1}$ such that for every $t < n^k$,
$$(\mathfrak{A}, <^\bullet, S^\bullet_{q_0}, \dots, S^\bullet_{q_u}, T^\bullet_{\sigma_0}, \dots, T^\bullet_{\sigma_v}, H^\bullet) \models F_{C_t,t}.$$
- If $(\mathfrak{A}, <^\bullet, S^\bullet_{q_0}, \dots, S^\bullet_{q_u}, T^\bullet_{\sigma_0}, \dots, T^\bullet_{\sigma_v}, H^\bullet) \models END$ then C_{n^k-1} is not rejecting.

Therefore, if $(\mathfrak{A}, <^\bullet, S^\bullet_{q_0}, \dots, S^\bullet_{q_u}, T^\bullet_{\sigma_0}, \dots, T^\bullet_{\sigma_v}, H^\bullet) \models G_M$, then there exists an accepting computation of M on input $(\mathfrak{A}, <^\bullet)$ and the lemma is proved. □

It remains to be observed that the property of being a linear order is finitely axiomatizable in first-order logic. Let $LO_<$ be a statement that expresses in first-order that $<$ is a linear order. Then we have that
$$\mathfrak{A} \in \mathcal{C} \Leftrightarrow \mathfrak{A} \models \exists < \exists S_{q_0} \dots \exists S_{q_u} \exists T_{\sigma_0} \dots \exists T_{\sigma_v} \exists H (LO_< \wedge G_M).$$

□

FIGURE 3. Stephen A. Cook (1939-)

From Fagin's Theorem we can obtain elegant **NP**-completeness proofs. For example we can give a short proof of the famous result by Stephen Cook on the **NP**-completeness of propositional satisfiability.

Corollary 4.6. *The set of satisfiable propositional formulae is* **NP***-complete.*

Proof. It is sufficient to show that every problem in **NP** is reducible to SAT (by a polynomial reduction).

We can assume that all problems in question are expressed in a fixed finite relational language $\mathcal{L}$, so they can be identified with a class $\mathcal{C}$ of finite structures.

By Fagin's Theorem we know that this class is completely described by an Existential Second Order sentence, in the sense that $\mathfrak{A} \in \mathcal{C}$ if and only if $\mathfrak{A} \models \exists R_1 \dots \exists R_n F$.

For a finite structure $\mathfrak{A} = (\{a_1, \dots, a_n\}, S^{\mathfrak{A}}_1, \dots, S^{\mathfrak{A}}_t)$, consider the propositional formula $F_{\mathfrak{A}}$ defined as follows: replace in F all subformulas of the form $\exists x G$ by the disjunction $G[x/a_1] \vee \dots \vee G[x/a_n]$; all formulas $\forall x G$ by the conjunction $G[x/a_1] \wedge \dots \wedge G[x/a_n]$ (the a_is are constant symbols corresponding to the elements of the domain).

Once these substitutions are done we can have atomic formulas of two types.

Substitute all atomic formulas $R(a_1, \dots, a_s)$ by propositional variables $p_{R(a_1,\dots,a_s)}$.

Substitute all atoms of the form $S(a_1, \ldots, a_m)$ for each relation symbol S in the language of $\mathfrak{A}$ by their truth value in $\mathfrak{A}$. Note that by the value of these atomic formulas can be efficiently computed from the code of the structure, if the coding is good.

The propositional formula obtained in this way is such that $\mathfrak{A} \in \mathcal{C}$ iff $F_{\mathfrak{A}}$ is satisfiable.

Also note that the reduction is computationally efficient (in particular logspace).

□

A characterization as in Fagin's Theorem but for other complexity classes is known. A major open problem is to get such a characterization for Polynomial Time. A Theorem by Immerman and Vardi (resp. Grädel) gives such a characterization in terms of fix-point logics (resp. SO-HORN logics) but *only* for ordered structures.