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

HANDOUT N. 7

Abstract. Definability, Axiomatizability, Expressibility. Queries over finite structures. Bounded elementary equivalence. Fraïssé's characterization by Back and Forth conditions.

1. Axiomatizability of Properties, Expressibility of Queries

A property P of a class of structures $\mathcal{C}$ is **expressible** (or **axiomatizable**) in a language $\mathcal{L}$ is there exists a set of sentences T (a theory) in $\mathcal{L}$ such that, for any structure $\mathfrak{A}$ in $\mathcal{C}$,

$$\mathfrak{A} \text{ has property } P \text{ if and only if } \mathfrak{A} \models T.$$

If T is finite we speak of **finite expressibility** (or **finite axiomatizability**).

We will focus first on (non)-expressibility results *over the class of all finite relational structures.*

We have seen two isomorphic structures satisfy the same sentences, i.e., are elementary equivalent.

Often, elementary equivalence is enough for proving many properties of theories. It is therefore interesting to isolate conditions ensuring elementary equivalence.

Elementary equivalence can be used to prove non-expressibility results.

Method 1 for non-expressibility proofs:

If there exist two structures $\mathfrak{A}$ and $\mathfrak{B}$ such that $\mathfrak{A}$ is elementary equivalent to $\mathfrak{B}$ but $\mathfrak{A}$ has property P and $\mathfrak{B}$ does not have property P then property P is not axiomatizable in first-order logic.

This method fails over finite models, because it is easy to see that if two finite models for a relational language satisfy the same sentences (i.e. are elementary equivalent) then they are isomorphic. So the two models cannot differ by any sensible property P (as long as P is preserved under isomorphisms, a standard assumption in both Maths and CS).

As we will see, a refinement of the notion of elementary equivalence will work for finite structures.

A finite relational structure is essentially a relational database and the study of finite axiomatizability is connected to the study of expressibility of queries over databases. We first formalize these ideas as follows.

Definition 1.1. A **class of structures** is a collection $\mathcal{C}$ of structures that is closed under isomorphism (we do not distinguish between isomorphic structures).

A k**-ary query** on a class $\mathcal{C}$ of structures is a map Q such that, for each $\mathfrak{A} \in \mathcal{C}$, $Q(\mathfrak{A})$ is a k-ary relation over the domain A of $\mathfrak{A}$.

A **boolean query** on a class $\mathcal{C}$ of structures is a map Q such that for each $\mathfrak{A} \in \mathcal{C}$, $Q(A)$ is either 0 or 1.

Consider for example the class $\mathcal{G}$ of graphs. Any meaningful property of graphs is a boolean query on $\mathcal{G}$: e.g., being connected, being eulerian, being a tree, etc. An important 2-ary query that is meaningful on any class of structures containing a binary relation R is the transitive closure property:

$$TC(\mathfrak{A}) = \{(a, b) : a, b \in A \text{ and in the transitive closure of the relation } R\}.$$

For the case of graphs, this is the same as

$$TC(\mathfrak{G}) = \{(v, w) : v, w \in V \text{ such that there is a path from } a \text{ to } b\}.$$

Notes by Lorenzo Carlucci, carlucci@di.uniroma1.it.

We now formalize what it means that a query is expressible in a given predicate language $\mathcal{L}$.

Definition 1.2. A k-ary query Q is **expressible** in a predicate language $\mathcal{L}$ over a class $\mathcal{C}$ if for some formula $F(x_1, \dots, x_k)$ in $\mathcal{L}$ with k free variables, for any structure $\mathfrak{A}$ in $\mathcal{C}$, for any k-tuple $(a_1, \dots, a_k)$ in A^k,

$$(a_1, \dots, a_k) \in Q(\mathfrak{A}) \text{ if and only if } \mathfrak{A} \models F(x_1, \dots, x_k)[a_1, \dots, a_k].$$

A boolean query Q is **expressible** in a predicate language $\mathcal{L}$ over a class $\mathcal{C}$ if for some sentence E in $\mathcal{L}$, for any structure $\mathfrak{A}$ in $\mathcal{C}$,

$$Q(\mathfrak{A}) = 1 \text{ if and only if } \mathfrak{A} \models E.$$

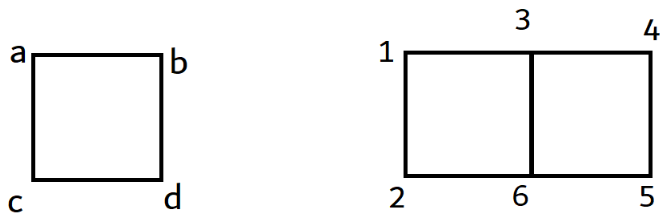
The main cases of interests are when $\mathcal{C}$ is the class of all structures (uniform definability, classical Model Theory), a single structure of particular interest (local definability), or the class of all finite structures (expressibility of queries, Finite Model Theory).

It is easy to see, for example, that: the property (query) of being a group is expressible in any language with a binary relation symbol over all structures, the property (query) of being a graph is expressible in any language with a binary relation symbol over all structures. Also, the boolean query of having an isolated node is expressible in the language of graphs over graphs, the unary query "v has at least two distinct neighbours" is expressible in the language of graphs over the class of graphs, the binary query "there exists a path of length n from v to w" is expressible, for any fixed n, in the language of graphs over graphs, and so on. (Exercise: write the corresponding sentences or formulas expressing the queries).

2. Bounded elementary equivalence

We will consider a *graded* version of elementary equivalence. The intuition is the following: elementary equivalence expresses the idea that two structures cannot be distinguished by any sentence, in the sense that there is no sentence that holds in on structure but not in the other. Bounded elementary equivalence expresses the idea that two structures cannot be distinguished by predicate sentences *of a certain syntactic complexity.*

Example 2.1. Consider the following two graphs $\mathfrak{A}$ (left) and $\mathfrak{B}$ (right).



In class we wrote sentences distinguishing the two structures. For example

$$S_1 = \exists x_1 \exists x_2 \exists x_3 \exists x_4 (\bigwedge \neg(x_i = x_j) \wedge E(x_1, x_2) \wedge E(x_2, x_3) \wedge E(x_2, x_4)).$$

The sentence S_1 is true in $\mathfrak{B}$ and false in $\mathfrak{A}$. It has 4 nested quantifiers.

Trying to use less quantifiers we came up with other sentences, such as:

$$S_2 = \exists x_1 \exists x_2 \exists x_3 (\bigwedge \neg(x_i = x_j) \wedge \bigwedge \neg E(x_i, x_j)).$$

or

$$S_3 = \exists x_1 \exists x_2 (\neg E(x_1, x_2) \wedge \forall x_3 (\neg(x_1, x_3) \wedge \neg(x_2, x_3) \to E(x_1, x_3))),$$

expressing that there is a vertex with exactly one non-connected vertex, which is true in $\mathfrak{A}$ but false in $\mathfrak{B}$.

We could not find a sentence with less that 3 nested quantifiers that distinguishes between the two structures.

We define a notion of syntactic complexity for predicate formulas, the *degree* (or quantifier-rank), based on the nesting of quantifiers.

Definition 2.2 (Degree/Quantifier-Rank of a Formula)**.** An atomic formula has degree 0, the degree of $G \vee H$ and $G \wedge H$ is the max of the degree of G and the degree of H, the degree of $\neg G$ is the degree of G, the degree of $\forall x G$ and of $\exists x G$ is the degree of G plus 1.

Next we formalize the the notion of two structures being indistinguishable by sentences of a certain complexity.

Definition 2.3 (k-elementary equivalence, $\equiv_k$)**.** Two structures are k-elementary equivalent if they satisfy the same sentences of degree $\leq k$. In that case we write $\mathfrak{A} \equiv_k \mathfrak{B}$.

Two structures are *elementary equivalent* (written $\mathfrak{A} \equiv \mathfrak{B}$) if they are k-elementary equivalent for all k.

The notion of k-elementary equivalence gives rise to a method for proving non-expressibility of queries over finite models. The **basic idea** is the following. If a property P is *finitely* axiomatizable, then the axioms have some maximal syntactic complexity, say k. Then any two structures that satisfy the same sentences of that complexity either both have property P or both don't.

If we are able to prove that for every k there exist two structures that can be distinguished with respect to property P but cannot be distinguished by formulas of complexity up to k, then property P is not finitely axiomatizable.

If P is finitely axiomatizable on finite structures, then $\mathfrak{A}$, $\mathfrak{B}$ as above cannot exist with *finite* cardinality.

We now proceed to formalize this idea, as well as to give a method for finding $\mathfrak{A}$ and $\mathfrak{B}$, if they exist.

First let's formalize the notion of syntactical complexity of a sentence:

Now are basic idea can be expressed more formally.

Method 2 for non-expressibility proofs:

Let $\mathcal{C}$ be a class of structures and P a property of structures. If for all k there exist $\mathfrak{A}$ and $\mathfrak{B}$ in $\mathcal{C}$ such that

- $\mathfrak{A}$ has P and $\mathfrak{B}$ doesn't have P, but
- $\mathfrak{A} \equiv_k \mathfrak{B}$

then P is not finitely axiomatizable over $\mathcal{C}$.

If $\mathfrak{A}$ and $\mathfrak{B}$ are finite, then P is not finitely axiomatizable in the finite.

It is therefore of interest to isolate methods for proving the $\mathfrak{A} \equiv \mathfrak{B}$ or that $\mathfrak{A} \equiv_k \mathfrak{B}$.

As we we will see we can obtain a game-theoretic characterization of $\equiv$ and $\equiv_k$. We first start with a somewhat combinatorial characterization.

3. Back-and-Forth Relations

We give a characterization of graded elementary equivalence in terms of an extendibility property. The idea is to generalize the back-and-forth method.

It is very useful for the forthcoming definitions and proofs to introduce the concept of *expansion* of a structure.

Definition 3.1 (Expansion of a Structure)**.** Let $\mathfrak{A}$ be a structure for language $\mathcal{L}$ and let $c_1, \ldots, c_n$ be n new constant symbols. Let $a_1, \ldots, a_n$ be a sequence in A. We denote by $(\mathfrak{A}, a_1, \ldots, a_n)$ the structure for the language $\mathcal{L}^c = \mathcal{L} \cup \{c_1, \ldots, c_n\}$ that coincides with $\mathfrak{A}$ on $\mathcal{L}$ and interprets c_i as a_i. If $\varphi(x_1, \ldots, x_n)$ is a formula in $\mathcal{L}$ with n free variables we denote by φ^c the **sentence** in $\mathcal{L}^c$ obtained by substituting each occurrence of x_i by c_i.

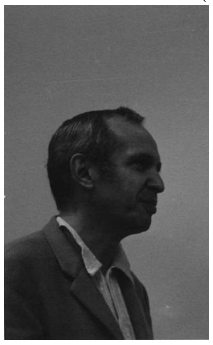
NB: The language $\mathcal{L}^c$ for structure $(\mathfrak{A}, a_1, \ldots, a_n)$ contains more sentences (in particular more atomic sentences) than language $\mathcal{L}$.

Is is easy to observe that that

$$(\mathfrak{A}, a_1, \ldots, a_n) \models \varphi^c \Longleftrightarrow \mathfrak{A} \models \varphi(x_1, \ldots, x_n)[a_1, \ldots, a_n].$$

We now show that $\equiv_{k+1}$ satisfies an inductive description. This is due to R. Fraïssé.

FIGURE 1. Roland Fraïssé (1920-2008)



Theorem 3.2 (R. Fraïssé). *For all $k \geq 0$, $\mathfrak{A} \equiv_{k+1} \mathfrak{B}$ if and only if the following two conditions hold:*

(1) *(Forth) For all $a \in A$ there exists $b \in B$ such that $(\mathfrak{A}, a) \equiv_k (\mathfrak{B}, b)$.*
(2) *(Back) For all $b \in B$ there exists $a \in A$ such that $(\mathfrak{A}, a) \equiv_k (\mathfrak{B}, b)$.*

Proof. We consider the general case $k + 1$.

First consider the $\Rightarrow$ direction. Suppose $\mathfrak{A} \equiv_{k+1} \mathfrak{B}$. We have to show that the Forth and the Back conditions are satisfied. Consider the Forth condition.

Let $a \in A$. We have to show that for some $b \in B$ we have $(\mathfrak{A}, a) \equiv_k (\mathfrak{B}, b)$. This means that we have to find a $b \in B$ such that the expansion of $\mathfrak{B}$ by b satisfies the same sentences of degree $\leq k$ as the expansion of $\mathfrak{A}$ by a. Recall that the expansions are structures for the language $\mathcal{L} \cup \{c\}$ where $\mathcal{L}$ is the language of $\mathfrak{A}$ and $\mathfrak{B}$ and c is a new constant. Notice that the senteces of degree $\leq k$ in the language $\mathcal{L} \cup \{c\}$ are either sentences of degree $\leq k$ in $\mathcal{L}$ (this is the case if they do not contain the new constant c) or else they are of the form F^c for some F in $\mathcal{L}$ of degree $\leq k$. In the first case we are done since a sentence not containing c is satisfied in $\mathfrak{A}$ if and only if it is satisfied in $(\mathfrak{A}, a)$, and we are under the hypothesis that $\mathfrak{A} \equiv_{k+1} \mathfrak{B}$.

So consider the case in which we are dealing with F^c.

If $(\mathfrak{A}, a) \models F^c$ then $\mathfrak{A} \models F(x)[\binom{x}{a}]$. Then $\mathfrak{A} \models \exists x F(x)$. The latter is a sentence of complexity $\leq k + 1$ and therefore $\mathfrak{B} \models \exists x F(x)$. By definition of satisfaction there exists $b \in B$ such that $\mathfrak{B} \models F(x)[\binom{x}{b}]$. Then $(\mathfrak{B}, b) \models F^c$, and we are done. ... Well, not quite!!! We have just shown that there is a $b \in B$ such that the expansion $(\mathfrak{B}, b)$ satisfies the sentence F^c. Instead, we need to show that there is a $b \in B$ such that the expansion $(\mathfrak{B}, b)$ satisfies all the sentences of complexity $\leq k$ satisfied by the expansion $(\mathfrak{A}, a)$.

Let's try as follows: list all sentences of complexity $\leq k$ in $\mathcal{L}$ satisfied by a in $\mathfrak{A}$. This list might be countably infinite, let it be $F_1(x), F_2(x), \ldots,$. For each $F_i(x)$ we have that $(\mathfrak{A}, a) \models F_i^c$ and we can find as above a $b \in B$ such that $(\mathfrak{B}, b) \models F_i^c$. But, again, we don't get a single b!!

We overcome this problem as follows: the idea is just that – if the language is finite and relational – we can find a single formula $H(x)$ of complexity $\leq k$ with one free variable that completely describes the $F_i(x)'s$ that are satisfied by an element $a \in A$. Then, applying the above reasoning to that particular formula H is enough to get a b that works for all the F_is.

Given $a \in A$ we can consider, for each fixed k, the set of formulas with one variable and of quantifier rank $\leq k$ that are satisfied in $\mathfrak{A}$ by a. Let's call the following set the **k-type of a in $\mathfrak{A}$**:

$$\{F(x) : F(x) \text{ of rank } \leq k \text{ with one free variable } x \text{ s.t. } \mathfrak{A} \models F(x)[\binom{x}{a}]\}.$$

The first observation is that the set of formulas of complexity $\leq k$ with one free variable is finite modulo logical equivalence, if the language is a finite relational language (Exercise: prove this fact). Therefore any k-type can be expressed by a combination of finitely many formulas.

Let

$$X = \{G_1(x), \ldots, G_m(x)\}$$

be a set of representatives of formulas of degree $\leq k$ in 1 free variable modulo logical equivalence. Then the k-type of an element $a \in A$ is uniquely determined (modulo logical equivalence) by a subset of X, i.e. by a set

$$\{G_{i_1}(x), \dots, G_{i_s}(x)\}$$

for some $s, i_1, \dots, i_s \leq m$. Let $S = \{i_1, \dots, i_s\}$.

Since S and $\{1, \dots, m\} - S$ are finite we can express by a single formula the fact that x satisfies all and only the G_i with $i \in S$. For any $S \subseteq \{1, \dots, m\}$ we define

$$H_S(x) = \bigwedge_{i \in S} G_i \wedge \bigwedge_{j \notin S} \neg G_j.$$

The formula $H_S(x)$ has degree $\leq k$ and one free variable x.

It is easy to see that if $S' \neq S$, then $H_S(x)$ are $H_{S'}(x)$ are mutually exclusive ($\mathfrak{A} \models H_S(a)$ implies $\mathfrak{A} \models \neg H_{S'}(a)$).

Furthermore, each formula of degree $\leq k$ is equivalent to a disjunction of formulas of type H_S. If F has degree $\leq k$, then F is equivalent to G_i for some $1 \leq i \leq m$. But G_i is equivalent to $\bigvee_{S \text{ s.t. } i \in S} H_S$:

$$G_i(x) \equiv \bigvee_{S \subseteq \{1, \dots, m\}, i \in S} H_S(x).$$

Now the argument goes through as follows: let $a \in A$ be arbitrary. Consider the formula $H_S(x)$ describing the k-type of a in $\mathfrak{A}$. Since $\mathfrak{A} \models H_S(x)[\binom{x}{a}]$, we have $\mathfrak{A} \models \exists x H_S(x)$. By hypothesis then $\mathfrak{B} \models \exists x H_S(x)$. Then for some $b \in B$ we have $\mathfrak{B} \models H_S(x)[\binom{x}{b}]$. Then, for any sentence F^c in $\mathcal{L}^c$ of complexity $\leq k$, we have that $(\mathfrak{A}, a) \models F^c$ iff $(\mathfrak{B}, b) \models F^c$. Indeed suppose $(\mathfrak{A}, a) \models F^c$. Then $F(x)$ belongs to the k-type of a in $\mathfrak{A}$, hence also to the k-type of b in $\mathfrak{B}$ and thus $(\mathfrak{B}, b) \models F^c$. Viceversa, if $(\mathfrak{B}, b) \models F^c$ then $F(x)$ belongs to the k-type of b in $\mathfrak{B}$, i.e., satisfies $H_S(x)$ in $\mathfrak{B}$. Then it also belongs to the k-type of a in $\mathfrak{A}$.

Now consider the $\Leftarrow$ direction. We assume that the Forth and Back conditions hold and we want to prove that $\mathfrak{A} \equiv_{k+1} \mathfrak{B}$. For this it is enough to show that $\mathfrak{A}$ and $\mathfrak{B}$ satisfy the same sentences of the form $\exists x F(x)$ where $F(x)$ is a formula of complexity $\leq k$ with one free variable x. So suppose $\mathfrak{A} \models \exists x F(x)$. Then for some $a \in A$ we have $\mathfrak{A} \models F(x)[\binom{x}{a}]$. Then also $(\mathfrak{A}, a) \models F^c$. Then for some $b \in B$ we have $(\mathfrak{B}, b) \models F^c$ and therefore $\mathfrak{B} \models F(x)[\binom{x}{b}]$ and finally $\mathfrak{B} \models \exists x F(x)$.

$\square$