Introduction to Descriptive Complexity

George Voutsadakis[1]

[1]Mathematics and Computer Science
Lake Superior State University

LSSU Math 600

1. Background in Logic
   - Introduction and Preliminary Definitions
   - Ordering and Arithmetic
   - FO(BIT) = FO(PLUS, TIMES)
   - Isomorphism
   - First-Order Queries

Subsection 1

Introduction and Preliminary Definitions

Vocabulary

- A **vocabulary**

$$\tau = \langle R_1^{a_1}, \ldots, R_r^{a_r}, c_1, \ldots, c_s, f_1^{r_1}, \ldots, f_t^{r_t} \rangle$$

is a tuple consisting of relation symbols, constant symbols and function symbols.
  - R_i is a relation symbol of arity a_i;
  - c_j is a constant symbol;
  - f_k is a function symbol of arity r_k.

Examples:
  - $\tau_g = \langle E^2, s, t \rangle$, the vocabulary of graphs with specified source and terminal nodes;
  - $\tau_s = \langle \leq^2, S^1 \rangle$, the vocabulary of binary strings.

Structures

- A **structure** with vocabulary τ is a tuple

$$\mathcal{A} = \langle |\mathcal{A}|, R_1^{\mathcal{A}}, \ldots, R_r^{\mathcal{A}}, c_1^{\mathcal{A}}, \ldots, c_s^{\mathcal{A}}, f_1^{\mathcal{A}}, \ldots, f_t^{\mathcal{A}} \rangle$$

  whose universe is the nonempty set $|\mathcal{A}|$.
  - For each relation symbol R_i of arity a_i in τ, $\mathcal{A}$ has a relation $R_i^{\mathcal{A}}$ of arity a_i defined on $|\mathcal{A}|$, i.e., $R_i^{\mathcal{A}} \subseteq |\mathcal{A}|^{a_i}$.
  - For each constant symbol $c_j \in \tau$, $\mathcal{A}$ has a specified element of its universe $c_j^{\mathcal{A}} \in |\mathcal{A}|$.
  - For each function symbol $f_k \in \tau$, $f_k^{\mathcal{A}}$ is a total function from $|\mathcal{A}|^{r_k}$ to $|\mathcal{A}|$.
- A vocabulary without function symbols is called a **relational vocabulary**.
- In this notes, unless otherwise stated, all vocabularies are relational.
- The notation $\|\mathcal{A}\|$ denotes the cardinality of the universe of $\mathcal{A}$.

Finite Structures

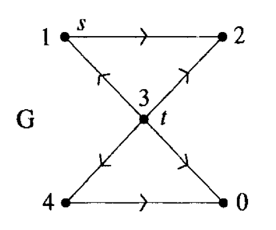
- Define STRUC$[\tau]$ to be the set of finite structures of vocabulary τ.

  Example: Consider the graph $G = \langle V^G, E^G, 1, 3 \rangle$ defined by

$$\begin{array}{rcl} V^G & = & \{0,1,2,3,4\}, \\ E^G & = & \{(1,2),(3,0),(3,1),(3,2),(3,4),(4,0)\}. \end{array}$$

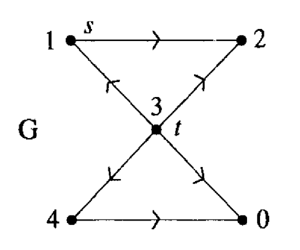
  It is a structure of vocabulary τ_g.

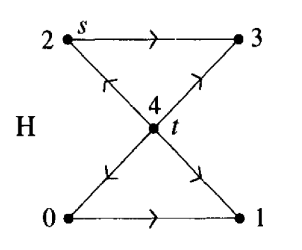
  It consists of a directed graph with two specified vertices s and t.



Example

- Consider the τ_g-structures G and H depicted below.





- G has five vertices and six edges.
- The graph H on the right is isomorphic but not equal to G.

Another Example

- Consider the binary string

$$w = \text{“01101”}.$$

- We can code w as the structure

$$\mathcal{A}_w = \langle \{0, 1, \dots, 4\}, \leq, \{1, 2, 4\} \rangle$$

  of vocabulary τ_s.
  - $\leq$ represents the usual ordering on $0, 1, \dots, 4$.
  - Relation $S^w = \{1, 2, 4\}$ represents the positions where w is one.
- Relation symbols of arity one, such as S^w, are sometimes called **monadic** or **unary**.

Relational Databases

- A **relational database** is exactly a finite relational structure.

  Running Example: Consider a genealogical database

$$\mathcal{B}_0 = \langle U_0, F_0, P_0, S_0 \rangle,$$

  where:
  - U_0 is a finite set of people, $U_0 = \{\text{Abraham}, \text{Isaac}, \text{Rebekah}, \text{Sarah}, \ldots\}$;
  - F_0 is a monadic relation that is true of the female elements of U_0, $F_0 = \{\text{Sarah}, \text{Rebekah}, \ldots\}$;
  - P_0 is the binary relation for parent $P_0 = \{(\text{Abraham}, \text{Isaac}), (\text{Sarah}, \text{Isaac}), \ldots\}$;
  - S_0 is the binary relation for spouse $S_0 = \{(\text{Abraham}, \text{Sarah}), (\text{Isaac}, \text{Rebekah}), \ldots\}$.

  Thus, $\mathcal{B}_0$ is a structure of vocabulary $\langle F^1, P^2, S^2 \rangle$.

First Order Languages

- For any vocabulary τ, define the **first-order language** $\mathcal{L}(\tau)$ to be the set of formulas built up from:
  - The relation and constant symbols of τ;
  - The logical relation symbol $=$;
  - The boolean connectives $\wedge, \neg$;
  - Variables: $\mathrm{VAR} = \{x, y, z, \ldots\}$;
  - The quantifier $\exists$.
- Other connectives (e.g., $\vee, \rightarrow, \ldots$) and the quantifier $\forall$, when they appear, will be taken to be abbreviations.

Bound and Free Variables

- We say that an occurrence of a variable v in φ is **bound** if it lies within the scope of a quantifier $(\exists v)$ or $(\forall v)$.
- Otherwise, the occurrence of v is **free**.
- Variable v is **free** in φ iff it has a free occurrence in φ.

  Example: The free variables in the following formula are x and y:

$$\alpha \equiv [(\exists y)(y + 1 = x)] \wedge x < y.$$

Metalogical Symbols

- We use the symbol "≡" to define or denote equivalence of formulas.
- In a similar way we sometimes use "⇔" to indicate that two previously defined formulas or conditions are equivalent.
- Bound variables are "dummy" variables and may be renamed to avoid confusion.

  Example: Consider the formula

  $$\alpha \equiv [(\exists y)(y + 1 = x)] \wedge x < y.$$

  It is equivalent to the formula

  $$\alpha' \equiv [(\exists z)(z + 1 = x)] \wedge x < y.$$

  α' also has free variables x and y.

Interpretations

- We write $\mathcal{A} \vDash \varphi$ to mean that $\mathcal{A}$ *satisfies* φ, i.e., that φ *is true in* $\mathcal{A}$.
- If φ contains free variables, they need to be interpreted.
- An **interpretation** into $\mathcal{A}$ is a map

$$i : V \to |\mathcal{A}|,$$

  where V is some finite subset of VAR.
- For convenience, for every constant symbol $c \in \tau$ and any interpretation i for $\mathcal{A}$, we let $i(c) = c^{\mathcal{A}}$.
- If τ has function symbols, then the definition of i extends to all terms via the recurrence

$$i(f_k(t_1, \ldots, t_{r_k})) = f_k^{\mathcal{A}}(i(t_1), \ldots, i(t_{r_k})).$$

Definition of Truth

- Let $\mathcal{A} \in \text{STRUC}[\tau]$ be a structure.
- Let i be an interpretation into $\mathcal{A}$ whose domain includes all the relevant free variables.
- We inductively define whether a formula $\varphi \in \mathcal{L}(\tau)$ **is true in** $(\mathcal{A}, i)$.

$$\begin{array}{rcl}
(\mathcal{A}, i) \vDash t_1 = t_2 & \Leftrightarrow & i(t_1) = i(t_2); \\
(\mathcal{A}, i) \vDash R_j(t_1, \ldots, t_{a_j}) & \Leftrightarrow & \langle i(t_1), \ldots, i(t_{a_j}) \rangle \in R_j^{\mathcal{A}}; \\
(\mathcal{A}, i) \vDash \neg\varphi & \Leftrightarrow & \text{it is not the case that } (\mathcal{A}, i) \vDash \varphi; \\
(\mathcal{A}, i) \vDash \varphi \wedge \psi & \Leftrightarrow & (\mathcal{A}, i) \vDash \varphi \text{ and } (\mathcal{A}, i) \vDash \psi; \\
(\mathcal{A}, i) \vDash (\exists x)\varphi & \Leftrightarrow & (\text{there exists } a \in |\mathcal{A}|)(\mathcal{A}, i, a/x) \vDash \varphi, \\
 & & \text{where } (i, a/x)(y) = \begin{cases} i(y), & \text{if } y \neq x \\ a, & \text{if } y = x \end{cases}
\end{array}$$

- Write $\mathcal{A} \vDash \varphi$ to mean that $(\mathcal{A}, \varnothing) \vDash \varphi$.

Abbreviations

- We define the "for all" quantifier, $\forall$, as the dual of $\exists$ and the boolean "or", $\vee$, as the dual of $\wedge$,

$$(\forall x)\varphi \equiv \neg(\exists x)\neg\varphi; \qquad \alpha \vee \beta \equiv \neg(\neg\alpha \wedge \neg\beta).$$

- It is convenient to introduce other abbreviations into our formulas.
  - "$y \neq z$" is an abbreviation for "$\neg y = z$";
  - "$\alpha \to \beta$" is an abbreviation for "$\neg\alpha \vee \beta$";
  - "$\alpha \leftrightarrow \beta$" is an abbreviation for "$\alpha \to \beta \wedge \beta \to \alpha$".
- Abbreviations are directly translatable into the real language.
- They help critically in making formulas more readable.
- Without abbreviations and the breaking of formulas into modular descriptions, it would be impossible to communicate complicated ideas in first-order logic.

Priority of Operations and Paremtheses

- We use spacing and parentheses to make the order of operations clear.
- Our convention for operator precedence is:
  - "$\neg$" has highest precedence;
  - "$\wedge$" and "$\vee$" come next;
  - "$\rightarrow$" and "$\leftrightarrow$" are last;
  - Operators of equal precedence are evaluated left to right.

Example: The following two formulas are equivalent,

$$\neg R(a) \rightarrow R(b) \wedge R(c) \leftrightarrow R(d),$$
$$((\neg R(a)) \rightarrow (R(b) \wedge R(c))) \leftrightarrow R(d).$$

Sentences

- A **sentence** is a formula with no free variables.
- Every sentence $\varphi \in \mathcal{L}(\tau)$ is either true or false in any structure $\mathcal{A} \in \text{STRUCT}[\tau]$.

  Example: Consider the following formula in the language of graphs,

$$\varphi_{\text{undir}} \equiv (\forall x)(\forall y)(\neg E(x,x) \wedge (E(x,y) \to E(y,x))).$$

  It says that the graph in question is undirected and has no loops.

Examples in the Language of Graphs

- Consider the formula

$$\begin{aligned}\varphi_{\text{out2}} \quad \equiv \quad & (\forall x)(\exists yz)(y \neq z \wedge E(x,y) \wedge E(x,z) \wedge \\ & (\forall w)(E(x,w) \to (w = y \vee w = z))).\end{aligned}$$

  It says that every vertex has exactly two edges leaving it.

- Consider now the formula

$$\varphi_{\text{deg2}} \equiv \varphi_{\text{undir}} \wedge \varphi_{\text{out2}}.$$

  It says that the graph in question is undirected, has no loops and is *regular of degree two*, i.e., every vertex has exactly two neighbors.

Examples in the Language of Graphs (Cont'd)

- Consider the following formulas.

$$
\begin{aligned}
\varphi_{\text{dist1}} &\equiv x = y \vee E(x,y) \\
\varphi_{\text{dist2}} &\equiv (\exists z)(\varphi_{\text{dist1}}(x,z) \wedge \varphi_{\text{dist1}}(z,y)); \\
\varphi_{\text{dist4}} &\equiv (\exists z)(\varphi_{\text{dist2}}(x,z) \wedge \varphi_{\text{dist2}}(z,y)); \\
\varphi_{\text{dist8}} &\equiv (\exists z)(\varphi_{\text{dist4}}(x,z) \wedge \varphi_{\text{dist4}}(z,y)); \\
&\vdots
\end{aligned}
$$

- They say that there is a path from x to y of length at most 1, 2, 4, 8, ..., respectively.
- Note that these formulas have free variables x and y.

Free Variables and Substitutions

- Formulas express properties about their free variables.

  Example: Consider a pair of vertices a, b in the universe of a graph G. Then the meaning of

  $$(G, a/x, b/y) \vDash \varphi_{\mathsf{dist8}}$$

  is that the distance from a to b in G is at most 8.
- Sometimes we will make the free variables in a formula explicit.
- E.g., we may write $\varphi_{\mathsf{dist8}}(x, y)$ instead of just φ_{dist8}.
- This offers the advantage of making substitutions more readable.
- We can write $\varphi_{\mathsf{dist8}}(a, b)$ instead of $\varphi_{\mathsf{dist8}}(a/x, b/y)$.

Examples in the Language of Arithmetic

- Consider the language of arithmetic

$$\tau_a = \langle \mathrm{PLUS}^3, \mathrm{TIMES}^3, 0, 1, \max \rangle.$$

- For $n \in \mathbb{N}$, consider the structure

$$\mathcal{A}_n = \langle \{0, 1, \ldots, n-1\}, \mathrm{PLUS}^{\mathcal{A}_n}, \mathrm{TIMES}^{\mathcal{A}_n}, 0, 1, n-1 \rangle,$$

  where PLUS and TIMES are the arithmetic relations.
- That is, for all $i, j, k < n$:
  - $\mathcal{A}_n \vDash \mathrm{PLUS}(i, j, k)$ iff $i + j = k$;
  - $\mathcal{A}_n \vDash \mathrm{TIMES}(i, j, k)$ iff $i \cdot j = k$.

Examples in the Language of Arithmetic (Cont'd)

- In $\mathcal{L}(\tau_a)$ we may write a formula DIVIDES(x, y) that says "x divides y" or, equivalently, "y is a multiple of x".

$$\text{DIVIDES}(x, y) \equiv (\exists z)\text{TIMES}(x, z, y).$$

- Similarly, we may write a formula PRIME(x) that says that "x is a prime number".

$$\text{PRIME}(x) \equiv (x \neq 1) \wedge (\forall y)(\text{DIVIDES}(y, x) \rightarrow y = 1 \vee y = x).$$

- Finally, via a formula $p_2(x)$, we may express that "x is a power of 2".

$$p_2(x) \equiv (\forall y)(\text{DIVIDES}(y, x) \wedge \text{PRIME}(y) \rightarrow y = 2).$$

- Note that $p_2(x)$ exploits the fact that x is a power of 2 if and only if 2 is the only prime divisor of x.

Examples in the Language of Strings

- Recall the language of strings

$$\tau_s = \langle \leq, S^1 \rangle.$$

- S is a unary relation indicating the positions of "1"s.
- The following formula in the language of strings uses the abbreviation "$x < y$" to mean "$x \leq y \wedge x \neq y$".

$$\varphi_{\text{no11}} \equiv (\forall x)(\forall y)(\exists z)((S(x) \wedge S(y) \wedge x < y) \rightarrow (x < z < y \wedge \neg S(z))).$$

- It describes the set of strings that have no consecutive "1"s.

Examples in the Language of Strings (Cont'd)

- Introduce the abbreviation "distinct".

$$\text{distinct}(x_1, \ldots, x_k) \equiv (x_1 \neq x_2 \wedge \cdots \wedge x_1 \neq x_k \wedge \cdots \wedge x_{k-1} \neq x_k).$$

- The following formula uses the abbreviation "distinct".

$$\begin{aligned} \varphi_{\text{five1}} \quad \equiv \quad & (\exists uvwxy)(\text{distinct}(u, v, w, x, y) \wedge S(u) \wedge S(v) \\ & \wedge S(w) \wedge S(x) \wedge S(y)). \end{aligned}$$

- It says that the given string contains at least five "1"s.
- Note that φ_{five1} uses five variables to say that there are five "1"s.

Examples in the Language of Strings (Cont'd)

- Using the ordering relation, we can reduce the number of variables.
- The following formula is equivalent to φ_{five1}, but uses only two variables:

$$(\exists x)(S(x) \wedge (\exists y)(x < y \wedge S(y) \wedge (\exists x)(y < x \wedge S(x) \wedge \\ (\exists y)(x < y \wedge S(y) \wedge (\exists x) y < x \wedge S(x))))).$$

- A good way to think of this sentence is that we have two fingers and are trying to count the number of "1"s in a string.
  - We put finger x down on the first "1";
  - Then we put finger y down on the next "1" to the right;
  - Now we don't need x anymore;
    So we can move it to the next "1" to the right of y;

    $\vdots$

Example: Two Binary Strings

- Let $\tau_{ab} = \langle \leq^2, A^1, B^1 \rangle$ consist of:
  - An ordering relation;
  - Two monadic relation symbols A and B, each serving the same role as the symbol S in τ_s.
- Let $\mathcal{A} \in \text{STRUC}[\tau_{ab}]$, and let $n = \|\mathcal{A}\|$.
- Then $\mathcal{A}$ is a pair of binary strings A, B, each of length n.
- These binary strings represent natural numbers, where we think of:
  - Bit zero as most significant;
  - Bit $n-1$ as least significant.

Example: Two Binary Strings (Cont'd)

- $A(i)$ is true iff bit i of A is "1".
- The following sentence expresses the ordering relation on such natural numbers represented in binary.

$$\text{LESS}(A,B) \equiv (\exists x)(B(x) \wedge \neg A(x) \wedge (\forall y.y < x)(A(y) \to B(y))).$$

- The **restricted quantifiers** are abbreviations.

$$\begin{array}{rcl} (\forall x.\alpha)\varphi & \equiv & (\forall x)(\alpha \to \varphi); \\ (\exists x.\alpha)\varphi & \equiv & (\exists x)(\alpha \wedge \varphi). \end{array}$$

Expressibility of Addition

Proposition

Addition of natural numbers, represented in binary, is first-order expressible.

- We use the well-known "carry-look-ahead" algorithm.

  In order to express addition, we first express the carry bit,

$$\varphi_{\text{carry}}(x) \equiv (\exists y. x < y)[A(y) \wedge B(y) \wedge (\forall z. x < z < y)(A(z) \vee B(z))].$$

  The formula $\varphi_{\text{carry}}(x)$ holds if:
  - There is a position y to the right of x where $A(y)$ and $B(y)$ are both one (i.e., the carry is generated);
  - For all intervening positions z, at least one of $A(z)$ and $B(z)$ holds (that is, the carry is propagated).

Expressibility of Addition (Cont'd)

- Let $\oplus$ be an abbreviation for the commutative and associative "exclusive or" operation.
- We can express φ_{add} as follows.

$$\begin{aligned} \alpha \oplus \beta &\equiv \alpha \leftrightarrow \neg\beta; \\ \varphi_{\text{add}}(x) &\equiv A(x) \oplus B(x) \oplus \varphi_{\text{carry}}(x). \end{aligned}$$

- Note that the formula $\varphi_{\text{add}}(x)$ has the free variable x.
- Thus, φ_{add} is a description of n bits, one for each possible value of x.

Substructures

- An important relation between two structures of the same type is that one may be a substructure of the other.
- $\mathcal{A}$ is a substructure of $\mathcal{B}$ if the universe of $\mathcal{A}$ is a subset of the universe of $\mathcal{B}$ and the relations and constants on $\mathcal{A}$ are inherited from $\mathcal{B}$.

Definition (Substructure)

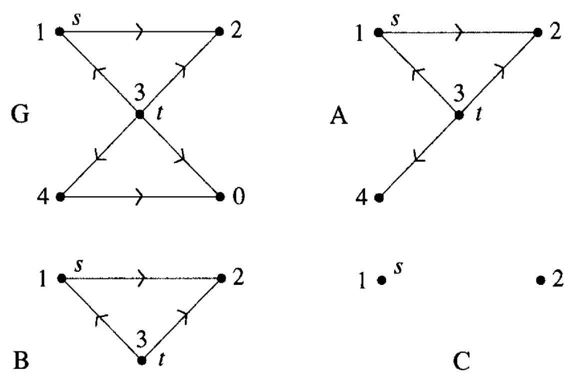
Let $\mathcal{A}$ and $\mathcal{B}$ be structures of the same vocabulary

$$\tau = \langle R_1^{a_1}, \ldots, R_r^{a_r}, c_1, \ldots, c_s \rangle.$$

We say that $\mathcal{A}$ is a **substructure of** $\mathcal{B}$, written $\mathcal{A} \leq \mathcal{B}$, iff the following conditions hold:

1. $|\mathcal{A}| \subseteq |\mathcal{B}|$;
2. For $i = 1, 2, \ldots, r$, $R_i^{\mathcal{A}} = R_i^{\mathcal{B}} \cap |\mathcal{A}|^{a_i}$;
3. For $j = 1, 2, \ldots, s$, $c_j^{\mathcal{A}} = c_j^{\mathcal{B}}$.

Example



- $\mathcal{A}$ and $\mathcal{B}$ are substructures of $\mathcal{G}$.
- $\mathcal{C}$ is not a substructure of $\mathcal{G}$ for two reasons.
  - It does not contain the constant t;
  - The induced edge from vertex 1 to vertex 2 is missing.

Substructures and Restricted Quantification

Proposition

Let $\mathcal{A} \in \mathrm{STRUC}[\tau]$ be a structure. Let $\alpha(x)$ be a formula, such that $\mathcal{A} \vDash (\exists x)\alpha(x)$. Assume, also, that for every constant symbol c in τ, $\mathcal{A} \vDash \alpha(c)$. Let $\mathcal{B}$ be the substructure of $\mathcal{A}$ with universe

$$|\mathcal{B}| = \{a \in |\mathcal{A}| : \mathcal{A} \vDash \alpha(a)\}.$$

Let φ be a sentence in $\mathcal{L}(\tau)$. Define the restriction of φ to α to be the sentence φ^α, the result of changing every quantifier $(\forall y)$ or $(\exists y)$ in φ to the restricted quantifier $(\forall y.\alpha(y))$ or $(\exists y.\alpha(y))$, respectively. Then

$$\mathcal{A} \vDash \varphi^\alpha \quad \text{iff} \quad \mathcal{B} \vDash \varphi.$$

- By induction, for all formulas $\varphi(\overline{x})$ and all $\overline{b} \in |\mathcal{B}|$,

$$(\mathcal{B}, \overline{b}) \vDash \varphi \quad \text{iff} \quad (\mathcal{A}, \overline{b}) \vDash \varphi^\alpha \text{ and } \mathcal{A} \vDash \alpha(b_i), \text{ for all } i.$$

Universal and Existential Formulas

- We say that φ is **universal** iff it can be written in prenex form, i.e., with all quantifiers at the beginning, using only universal quantifiers.
- Similarly, we say that φ is **existential** iff it can be written in prenex form with only existential quantifiers.
- The following "preservation theorems" provide a good way of proving that a formula is existential or universal.

Proposition

Let $\mathcal{A} \leq \mathcal{B}$ be structures and φ a first-order sentence.

1. Suppose φ is existential. If $\mathcal{A} \vDash \varphi$, then $\mathcal{B} \vDash \varphi$.
2. Suppose φ is universal. If $\mathcal{B} \vDash \varphi$, then $\mathcal{A} \vDash \varphi$.

- By induction on the structure of φ.

Subsection 2

Ordering and Arithmetic

Structures to Numbers to Words

- Let $\mathcal{A} \in \text{STRUC}[\tau]$ be an ordered structure.
- Let $n = \|\mathcal{A}\|$.
- Suppose the elements of $|\mathcal{A}|$ in increasing order are $a_0, a_1, \ldots, a_{n-1}$.
- Then there is a 1:1 correspondence $i \mapsto a_i$, $i = 0, 1, \ldots, n-1$.
- We usually identify the elements of the universe with the set of natural numbers less than n.
- In a computer these would be represented as $\lceil \log n \rceil$-bit words.
- Moreover, the operations plus, times, and even picking out bit j of such a word, would all be wired in.

Numeric Relations

- The following numeric relations are useful.
  1. PLUS(i, j, k), meaning $i + j = k$;
  2. TIMES(i, j, k), meaning i$\cdot j = k$;
  3. BIT(i, j), meaning bit j in the binary representation of i is 1.
- In the definition of BIT we will take bit 0 to be the low order bit.
- So we have BIT$(i, 0)$ holds iff i is odd.

Numeric Relations and Constants

- We may use the successor relation SUC in lieu of, or in addition to, $\leq$.
- SUC is first-order definable from $\leq$,

$$\text{SUC}(x, y) \equiv (x < y) \wedge (\forall z)(\neg(x < z \wedge z < y)).$$

- The symbols $\leq$, PLUS, TIMES, BIT, SUC, 0, 1, max are called **numeric relation** and **constant symbols**.
- They depend only on the size of the universe.
- The remainder of τ are the **input relation** and **constant symbols**.
- The numeric relations and constants are not explicitly given in the input, since they are easily computable as functions of the size of the input.
- Whenever any of the numeric relation or constant symbols occur, they are required to have their standard meanings.

Ordering Proviso

- From now on, unless stated otherwise, we assume that the numeric relations and constants:

$$\leq, \mathrm{PLUS}, \mathrm{TIMES}, \mathrm{BIT}, \mathrm{SUC}, 0, 1, \max$$

  are present in all vocabularies.
- When we define vocabularies, we do not explicitly mention or show these symbols, unless they are not present.
- We use the notation $\mathcal{L}(\mathrm{wo}\leq)$ to indicate language $\mathcal{L}$ without any of the numeric relations.
- We will write $\mathcal{L}(\mathrm{woBIT})$ to indicate language $\mathcal{L}$, including ordering, but not arithmetic, i.e., only the numeric relations $\leq$ and SUC and the constants 0, 1, max are included.

Boolean Constants Proviso

- The following proviso eliminates the trivial, and sometimes annoying, case of the structure with only one element.
- This structure satisfies the equation $0 = 1$.

  Boolean Constants Proviso: From now on, we assume that all structures have at least two elements.
- In particular, we will assume that we have two unequal constants denoted by 0 and 1.

Boolean Variables

- Next, we define what it means to have a boolean variable in a first-order formula.
- When we measure the number of first-order variables needed, we discount the (bounded) number of boolean variables.

Definition

A **boolean variable** in a first-order formula is a variable that is restricted to being either 0 or 1. Here 0 is identified with **false** and 1 is identified with **true**. We typically use the letters b, c, d, e for boolean variables.
We use the following abbreviations:

- $\text{bool}(b) \equiv b \leq 1$;
- $(\exists b) \equiv (\exists b.\text{bool}(b))$;
- $(\forall b) \equiv (\forall b.\text{bool}(b))$

Subsection 3

FO(BIT) = FO(PLUS, TIMES)

Interdefinability of BIT and PLUS, TIMES

- We prove that adding BIT to first-order logic is equivalent to adding PLUS and TIMES.
- We use the Bit Sum Lemma, which is interesting in its own right.

Theorem

Let τ be a vocabulary that includes ordering. Then:

1. If BIT $\in \tau$, then PLUS and TIMES are first-order definable;
2. If PLUS, TIMES $\in \tau$, then BIT is first-order definable.

1. We have seen that PLUS is expressible using BIT.

   To prove that TIMES is expressible, we need the Bit Sum Lemma.

The Bit Sum Lemma

Lemma (Bit Sum Lemma)

Let $\mathrm{BSUM}(x, y)$ be true iff y is equal to the number of ones in the binary representation of x. BSUM is first-order expressible using ordering and BIT.

- The bit-sum problem is to add a column of $\log n$ 0's and 1's.

  The idea is to keep a running sum.

  The sum of $\log n$ 1's requires at most $\log \log n$ bits to record.

  So we maintain running sums of $\log \log n$ bits each.

  With one existentially quantified variable, we can guess $\frac{\log n}{\log \log n}$ of these.

  Thus, to express $\mathrm{BSUM}(x, y)$ we existentially quantify s, the $\log \log n \cdot \frac{\log n}{\log \log n}$ bits of running sums.

The Bit Sum Lemma (Cont'd)

- In the following example, $n = 2^{16}$.

  So x and y each have 16 bits.

  We wish to assert $\mathrm{BSUM}(0110110110101101, 1010)$.

  We would guess $s = 0010010101111010$ as our partial sum bit string.

  Next we say that for all i, $i \leq \frac{\log n}{\log\log n}$, running sum i, plus the number of 1's in segment $(i+1)$ is equal to the running sum $(i+1)$.

  Thus, it suffices to express the bit sum of a segment of length $\log\log n$.

  We do this by keeping a running sum at every position.

  This requires only $\log\log\log n \cdot \log\log n$ bits.

  Note this is less than $\log n$ for sufficiently large n.

0	
1	
1	
0	0010
1	
1	
0	
1	0101
1	
0	
1	
0	0111
1	
1	
0	
1	1010

Interdefinability (Part 1 Cont'd)

- We next show that TIMES is first-order expressible using BIT.

  TIMES is equivalent to the addition of $\log n$ $\log n$-bit numbers

  $$A = A_1 + A_2 + \cdots + A_{\log n}.$$

  We split each A_i, into a sum of two numbers $A_i = B_i + C_i$, so that B_i and C_i have blocks of $\log\log n$ bits separated by $\log\log n$ 0's.

  $$\begin{array}{rcccccccccccc} B_i & = & a_{i,1} & \cdots & a_{i,\ell} & 0 & \cdots & 0 & \cdots & a_{i,\log n+1-\ell} & \cdots & a_{i,\log n} \\ +\,C_i & = & 0 & \cdots & 0 & a_{i,\ell+1} & \cdots & a_{i,2\ell} & \cdots & 0 & \cdots & 0 \\ \hline A_i & = & a_{i,1} & \cdots & a_{i,\ell} & a_{i,\ell+1} & \cdots & a_{i,2\ell} & \cdots & a_{i,\log n+1-\ell} & \cdots & a_{i,\log n} \end{array}$$

  We compute the sum of the B_i's and of the C_i's.

  In this way, we insure that no carries extend more than $\log\log n$ bits.

  Finally, we add the two sums with a single use of PLUS.

  In the following, let $\ell = \lceil \log\log n \rceil$.

Interdefinability (Part 1 Cont'd)

- In this way, we have reduced the problem of adding $\log n$ $\log n$-bit numbers to that of adding $\log n$ $\log\log n$-bit numbers.

  We can simultaneously guess the sums of each of the $\log\log n$ columns in a single variable c.

  Using BSUM and a universal quantifier, we can verify that each section of c is correct.

  Finally, we can add the $\log\log n$ numbers in c.

  We can do this by maintaining all the running sums, as in the last paragraph of the proof of the Bit Sum Lemma.

Interdefinability (Part 2)

2. We show BIT is first-order expressible using PLUS and TIMES.
   We do this with a series of definitions.
   First, recall $p_2(y)$, meaning that y is a power of 2.
   Next, define $\text{BIT}'(x, y)$ to mean, for some i, $y = 2^i$ and $\text{BIT}(x, i)$,
   $$\text{BIT}'(x, y) \equiv p_2(y) \wedge (\exists uv)(x = 2uy + y + v \wedge v < y).$$
   Using BIT' we can copy a sequence of bits.
   For example, the following formula says that if $y = 2^i$ and $z = 2^j$, then bits $i + j, \ldots, i$ of x are the same as bits $j, \ldots, 0$ of c.
   $$\text{COPY}(x, y, z, c) \equiv (\forall u.p_2(u) \wedge u < z)(\text{BIT}'(x, yu) \leftrightarrow \text{BIT}'(c, u)).$$
   Finally, to express BIT, we would like to express the relation $2^i = y$.
   We express this using the following recurrence,
   $$2^i = y \Leftrightarrow (\exists j)(\exists z.2^j = z)((i = 2j + 1 \wedge y = 2z^2) \vee (i = 2j \wedge y = z^2)).$$

Interdefinability (Part 2 Cont'd)

- We can guess two variables, Y, I, that simultaneously include all but a bounded number of the $\log i$ computations indicated by the recurrence.

  Namely all those such that $i > 2 \log i$.

  This is done as follows.

  Place a "1" in positions i, j, etc., of Y.

  Place the binary encoding of i starting at position i of I.

  Place the binary encoding of j starting at position j of I and so on.

  Using a universal quantifier we say that the variables Y and I encode all the relevant and sufficiently large computations of the recurrence.

Interdefinability (Part 2 Cont'd)

- The following table shows the encodings Y and I for the proposition

$$2^{15} = 32,768.$$

Position	16	15	14	13	12	11	10	9	8	7	6	5	4	3	2	1	0
Y	0	1	0	0	0	0	0	0	0	1	0	0	0	1	0	1	0
I	0	1	1	1	1	0	0	0	0	1	1	1	0	1	1	0	0

Encoding of $2^{15} = 32,768$.

Note that I records:
  - The exponent 15, which is 1111 in binary, starting at position 15;
  - The exponent 7 which is 111 in binary, starting at position 7;
  - The exponent 3 which is 11 in binary, starting at position 3.

We skip the details of actually writing the relevant first-order formula.

Subsection 4

Isomorphism

Isomorphism

- Two structures are isomorphic iff they are identical except perhaps for the names of the elements of their universes.

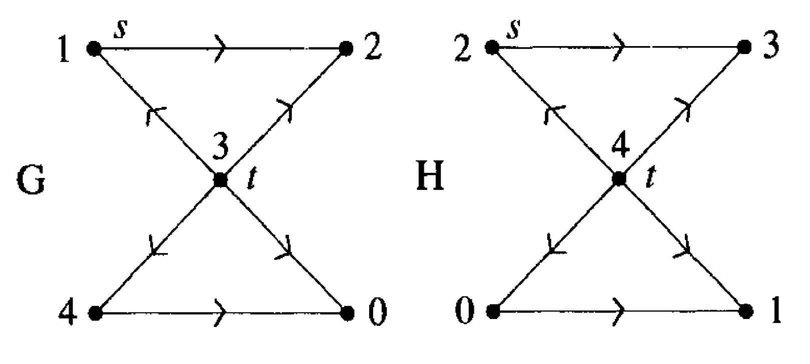
Definition (Isomorphism of Unordered Structures)

Let $\mathcal{A}$ and $\mathcal{B}$ be structures of vocabulary $\tau = \langle R_1^{a_1}, \ldots, R_r^{a_r}, c_1, \ldots, c_s \rangle$. We say that $\mathcal{A}$ is **isomorphic** to $\mathcal{B}$, written, $\mathcal{A} \cong \mathcal{B}$, iff there is a map $f : |\mathcal{A}| \to |\mathcal{B}|$ with the following properties:

1. f is 1-1 and onto;
2. For every input relation symbol R_i and for every a_i-tuple of elements of $|\mathcal{A}|$, $e_1, \ldots, e_{a_i}$,
$$\langle e_1, \ldots, e_{a_i} \rangle \in R_i^{\mathcal{A}} \quad \Leftrightarrow \quad \langle f(e_1), \ldots, f(e_{a_i}) \rangle \in R_i^{\mathcal{B}};$$
3. For every input constant symbol c_i, $f(c_i^{\mathcal{A}}) = c_i^{\mathcal{B}}$.

The map f is called an **isomorphism**.

Example



- Graphs $\mathcal{G}$ and $\mathcal{H}$ are isomorphic using the map that adds one mod 5 to the numbers of the vertices of $\mathcal{G}$.

Remarks on Isomorphisms

- Note that we have defined isomorphisms so that they need only preserve the input symbols, not the ordering and other numeric relations.
- If we included the ordering relation, then

$$\mathcal{A} \cong \mathcal{B} \quad \text{iff} \quad \mathcal{A} = \mathcal{B}.$$

- To be completely precise, we should call the mapping f defined above an "**isomorphism of unordered structures**" and say that $\mathcal{A}$ and $\mathcal{B}$ are "**isomorphic as unordered structures**".

Remarks on Isomorphisms (Cont'd)

- Note also that, since "unordered string" does not make sense, neither does the concept of isomorphism for strings.
- By the definition, two strings would be isomorphic as unordered structures iff they had the same number of each symbol.

Proposition

Suppose $\mathcal{A}$ and $\mathcal{B}$ are isomorphic. Then, for all sentences $\varphi \in \mathcal{L}(\tau - \{\leq\})$,

$$\mathcal{A} \vDash \varphi \quad \text{iff} \quad \mathcal{B} \vDash \varphi.$$

- One uses induction on the structure of a $\tau(\text{wo}\leq)$-formula $\varphi(\overline{x})$. More specifically, one shows that, for any assignment $\overline{a}$,

$$(\mathcal{A}, \overline{a}) \vDash \varphi \quad \text{iff} \quad (\mathcal{B}, f(\overline{a})) \vDash \varphi,$$

where $f : \mathcal{A} \to \mathcal{B}$ is an isomorphism.

Subsection 5

First-Order Queries

Queries

Definition

A **query** is any mapping

$$I : \mathrm{STRUC}[\sigma] \to \mathrm{STRUC}[\tau]$$

from structures of one vocabulary to structures of another vocabulary, that is polynomially bounded. That is, such that, there is a polynomial p, such that, for all $\mathcal{A} \in \mathrm{STRUC}[\sigma]$,

$$\|I(\mathcal{A})\| \le p(\|\mathcal{A}\|).$$

A **boolean query** is a map

$$I_b : \mathrm{STRUC}[\sigma] \to \{0, 1\}.$$

A boolean query may also be thought of as a subset of $\mathrm{STRUC}[\sigma]$ - the set of structures $\mathcal{A}$ for which $I(\mathcal{A}) = 1$.

Order-Independent or Generic Queries

- An important subclass of queries are the order-independent queries.
- These are called "generic" in database theory.

Definition

Let I be a query defined on STRUC$[\sigma]$.
Then I is **order-independent** iff, for all structures $\mathcal{A}, \mathcal{B} \in$ STRUC$[\sigma]$,

$$\mathcal{A} \cong \mathcal{B} \quad \text{implies} \quad I(\mathcal{A}) \cong I(\mathcal{B}).$$

For boolean queries, $I(\mathcal{A}) \cong I(\mathcal{B})$ translates to $I(\mathcal{A}) = I(\mathcal{B})$.

Introducing First-Order Queries

- The simplest kind of query is a **first-order query**.
- Any first-order sentence $\varphi \in \mathcal{L}(\tau)$ defines a boolean query I_φ on $\text{STRUC}[\tau]$, where
$$I_\varphi(\mathcal{A}) = 1 \quad \text{iff} \quad \mathcal{A} \vDash \varphi.$$

  Example: Let DIAM[8] be the query on graphs that is true of a graph iff its diameter is at most eight.

  Recall the formula φ_{dist8}, with free variables x, y, expressing that there is a path from x to y of length at most eight.

  Then the query DIAM[8] is a first-order query given by
$$\text{DIAM}[8] \equiv (\forall xy)\varphi_{\text{dist8}}.$$

Example

- Consider the query I_{add}, which, given a pair of natural numbers represented in binary, returns their sum.

  This query is defined by the first order formula φ_{add} encountered previously.

  More explicitly, let

  $$\mathcal{A} = \langle |\mathcal{A}|, \leq, A, B \rangle$$

  be any structure in $\text{STRUC}[\tau_{ab}]$.

  $\mathcal{A}$ is a pair of natural numbers, each of length $n = \|\mathcal{A}\|$ bits.

  Their sum is given by $I_{\text{add}}(\mathcal{A}) = \langle |\mathcal{A}|, S \rangle$, where

  $$S = \{a \in |\mathcal{A}| : (\mathcal{A}, a/x) \vDash \varphi_{\text{add}}\}.$$

  The first-order query $I_{\text{add}} : \text{STRUC}[\tau_{ab}] \to \text{STRUC}[\tau_s]$ maps structure $\mathcal{A}$ to another structure with the same universe, i.e., $|\mathcal{A}| = |I_{\text{add}}(\mathcal{A})|$.

First-Order Queries

- Let σ and τ be any two vocabularies where

$$\tau = \langle R_1^{a_1}, \ldots, R_r^{a_r}, c_1, \ldots, c_s \rangle.$$

- Let k be a fixed natural number.
- We want to define the notion of a first-order query,

$$I : \text{STRUC}[\sigma] \to \text{STRUC}[\tau].$$

- I is given by an $(r+s+1)$-tuple of formulas from $\mathcal{L}(\sigma)$,

$$\varphi_0, \varphi_1, \ldots, \varphi_r, \psi_1, \ldots, \psi_s.$$

- For each structure $\mathcal{A} \in \text{STRUC}[\sigma]$, these formulas describe a structure $I(\mathcal{A}) \in \text{STRUC}[\tau]$, defined as follows.

First-Order Queries (Cont'd)

- We have

$$I(\mathcal{A}) = \langle |I(\mathcal{A})|, R_1^{I(\mathcal{A})}, \ldots, R_r^{I(\mathcal{A})}, c_1^{I(\mathcal{A})}, \ldots, c_s^{I(\mathcal{A})} \rangle,$$

where:

  - The universe of $I(\mathcal{A})$ is a first-order definable subset of $|\mathcal{A}|^k$,

$$|I(\mathcal{A})| = \{\langle b^1, \ldots, b^k \rangle : \mathcal{A} \vDash \varphi_0(b^1, \ldots, b^k)\};$$

  - Each relation $R_i^{I(\mathcal{A})}$ is a first-order definable subset of $|I(\mathcal{A})|^{a_i}$,

$$R_i^{I(\mathcal{A})} = \{(\langle b_1^1, \ldots, b_1^k \rangle, \ldots, \langle b_{a_i}^1, \ldots, b_{a_i}^k \rangle) \in |I(\mathcal{A})|^{a_i} : \mathcal{A} \vDash \varphi_i(b_1^1, \ldots, b_{a_i}^k)\};$$

  - Each constant symbol $c_j^{I(\mathcal{A})}$ is a first-order definable element of $|I(\mathcal{A})|$,

$$c_j^{I(\mathcal{A})} = \text{the unique } \langle b^1, \ldots, b^k \rangle \in |I(\mathcal{A})| \text{ such that } \mathcal{A} \vDash \psi_j(b^1, \ldots, b^k).$$

First-Order Queries (Cont'd)

- When we need to be formal, we use the following conventions.
- We let

$$a = \max\{a_i : 1 \le i \le r\}$$

be the maximum among the arities of the relation symbols.
  - The free variables of φ_0 are $x_1^1, \ldots, x_1^k$.
  - The free variables of φ_i be $x_1^1, \ldots, x_1^k, \ldots, x_{a_i}^1, \ldots, x_{a_i}^k$.
  - The free variables of ψ_j are $x_1^1, \ldots, x_1^k$.
- If the formulas ψ_j have the property that for all $\mathcal{A} \in \mathrm{STRUC}[\sigma]$,

$$|\{\langle b^1, \ldots, b^k \rangle \in |\mathcal{A}|^k : (\mathcal{A}, b^1/x_1^1, \ldots, b^k/x_1^k) \vDash \varphi_0 \wedge \psi_j\}| = 1,$$

then we write

$$I = \lambda_{x_1^1 \ldots x_a^k} \langle \varphi_0, \ldots, \psi_s \rangle$$

and say that I is a k-**ary first-order query** from $\mathrm{STRUC}[\sigma]$ to $\mathrm{STRUC}[\tau]$.

Examples

- It is often possible to name constant $c_j^{I(\mathcal{A})}$ explicitly as a k-tuple of constants $\langle t^1, \ldots, t^k \rangle$.
- In this case, we may simply write this tuple in place of its corresponding defining formula,

$$\psi_j \equiv x_1^1 = t^1 \wedge \cdots \wedge x_1^k = t^k.$$

- As another example, in a 3-ary query I, the numerical constants 0, 1 and max will be mapped to the following:

$$0^{I(\mathcal{A})} = \langle 0,0,0 \rangle; \quad 1^{I(\mathcal{A})} = \langle 0,0,1 \rangle; \quad \max^{I(\mathcal{A})} = \langle \max, \max, \max \rangle.$$

Terminology and Notation

- A **first-order query** is one of the following types:
  - Boolean, and, thus, defined by a first-order sentence;
  - A k-ary first-order query, for some k.
- We denote by

$$\mathrm{FO}$$

  the set of first-order boolean queries.
- We denote by

$$\mathrm{Q(FO)}$$

  the set of all first-order queries.

Example

- Consider the genealogical database of a previous example.
- Consider the following pair of formulas.

$$\begin{array}{rcl} \varphi_{\text{sibling}}(x,y) & \equiv & (\exists fm)(x \neq y \wedge f \neq m \wedge P(f,x) \\ & & \wedge P(f,y) \wedge P(m,x) \wedge P(m,y)); \\ \varphi_{\text{aunt}}(x,y) & \equiv & (\exists ps(P(p,y) \wedge \varphi_{\text{sibling}}(p,s) \\ & & \wedge (s = x \vee S(x,s)))) \wedge F(x). \end{array}$$

- They define a unary query

$$I_{sa} = \lambda_{xy} \langle \mathbf{true}, \varphi_{\text{sibling}}, \varphi_{\text{aunt}} \rangle$$

  from genealogical databases to structures of vocabulary $\langle \text{SIBLING}^2, \text{AUNT}^2 \rangle$.
- We will see that many queries of interest are not first-order.
- One such example is the ancestor query on genealogical databases.

Example

- The first-order query

$$I_{\text{add}} : \text{STRUC}[\tau_{ab}] \to \text{STRUC}[\tau_s]$$

is a unary query, i.e., $k = 1$, given by

$$I_{\text{add}} = \lambda_{xy} \langle \textbf{true}, \varphi_{\text{add}} \rangle.$$

- In this case, $\varphi_0 = \textbf{true}$ means that the universe of $I_{\text{add}}(\mathcal{A})$ is equal to the universe of $\mathcal{A}$.

Example

- Consider the binary first-order query from graphs to graphs

$$I = \lambda_{x,y,x',y'}\langle \textbf{true}, \alpha, \langle 0,0\rangle, \langle \max,\max\rangle\rangle,$$

  where

$$\alpha(x,y,x',y') \equiv (x = x' \wedge E(y,y')) \vee (\mathrm{SUC}(x,y) \wedge x' = y' = y).$$

- Part of the meaning of this query is that, given a structure $\mathcal{A} \in \mathrm{STRUC}[\tau_g]$, with $n = \|\mathcal{A}\|$, we have:
  - $|I(\mathcal{A})| = \{\langle i,j\rangle : i,j \in |\mathcal{A}|\}$;
  - $s^{I(\mathcal{A})} = \langle 0,0\rangle$;
  - $t^{I(\mathcal{A})} = \langle n-1, n-1\rangle$.
- We can show that I satisfies the property that, for all undirected graphs G,

$$G \text{ is connected} \quad \text{iff} \quad t \text{ is reachable from } s \text{ in } I(G).$$

Closure of First Order Queries under Composition

- The set of first-order queries is closed under composition.

Proposition

Let

$$I_1 : \text{STRUC}[\sigma] \to \text{STRUC}[\tau]$$

be a k-ary first-order query. Let

$$I_2 : \text{STRUC}[\tau] \to \text{STRUC}[\nu]$$

be an m-ary first-order query. Then

$$I_2 \circ I_1 : \text{STRUC}[\sigma] \to \text{STRUC}[\nu]$$

is an mk-ary first-order query.

Remark

- If I is a first-order query on ordered structures, then it must include first-order definitions of the numeric relations and constants.
- Unless we state otherwise, the ordering on $I(\mathcal{A})$ will be the lexicographic ordering of k-tuples $\leq^k$ inherited from $\mathcal{A}$.
- This is defined inductively by

$$\leq^1 = \leq;$$

$$\begin{aligned}\langle x_1, \ldots, x_k \rangle \leq^k \langle y_1, \ldots, y_k \rangle \quad \equiv \quad & x_1 < y_1 \vee (x_1 = y_1 \wedge \\ & \langle x_2, \ldots, x_k \rangle \leq^{k-1} \langle y_2, \ldots, y_k \rangle).\end{aligned}$$

- For the first-order queries used here, we usually limit ourselves to the case that $\varphi_0 \equiv \mathbf{true}$.
- If this is not the case, we must express the new numeric relations explicitly.

Definitions of Numeric Relations and Constants

- Let I be a first-order query on ordered structures.
- The successor and bit relations must be defined.
  1. We must give the formulas defining 0, 1, and max, the minimum, second, and maximum elements, respectively, of the new universe under the lexicographical ordering.
     - If $\varphi_0 \equiv \textbf{true}$, then these are just k-tuples of constants:

       $$0^{I(\boldsymbol{A})} = \langle 0, \ldots, 0 \rangle; \quad 1^{I(\mathcal{A})} = \langle 0, \ldots, 0, 1 \rangle; \quad \max^{I(\mathcal{A})} = \langle \max, \ldots, \max \rangle.$$

     - In the more general case, we use quantifiers to say that the given element is the minimum, second, maximum in the lexicographical ordering.
  2. Assuming that $\varphi_0 \equiv \textbf{true}$, we can write a quantifier-free formula defining the new SUC relation.
  3. Assuming that $\varphi_0 \equiv \textbf{true}$, we can write the formula defining the new BIT relation.
- We have seen that BIT suffices to define PLUS and TIMES.

Remark

- Without the assumption that $\varphi_0 \equiv$ **true**, BIT need not be first-order definable in the image structures.

  Example: Suppose $\sigma = \tau_s$ and let

  $$\varphi_0(x) \equiv S(x).$$

  The parity of the universe of $I(\mathcal{A})$ is not first-order expressible in $\mathcal{A}$.

  If BIT were definable in $I(\mathcal{A})$, then so would the parity of its universe.
- For this reason, when we define first-order reductions, we restrict our attention to very simple formulas φ_0 that define the universe of the image structure.