Session I: Monday, August 14
Track B: Descriptive Complexity
Speaker: N. Immerman

1 Introduction to Descriptive Complexity

Complexity Theory is the study of how much time and space resources are needed to answer a Boolean query, as a function of the size of the input.

Descriptive Complexity Theory studies how rich a language, how descriptive a syntactical resource (e.g. quantifier rank, alternation depth, number of variables) is needed to define a query.

The Complexity Classes.

- $P = \bigcup_{k=1}^{\infty} DTIME[n^k]$
  P includes all "feasible problems": connectivity, network flow, linear algebra, evaluating circuits. In general, anything that we can solve exactly for all "reasonable size instances".
- $NP = \bigcup_{k=1}^{\infty} NTIME[n^k]$
  NP includes many important optimization problems: 3 colorability, SAT, etc. The best known algorithms for these problems run in exponential time.
- Other classes: $LOGSPACE$ (abbreviated L), $NLOGSPACE$ (abbreviated NL).

A classical example of a problem (set of structures) in NL is

$$GAP = \{G = (V, E, s, t) \mid \text{ there is a path from } s \text{ to } t\ \}$$

To go down to L we kill the possibilities of having choices for different paths:

$$1GAP = \{G \in GAP \mid outdegree(G) = 1\}$$

The General Picture.

$$L \subseteq NL \subseteq P \subseteq NP \subseteq PSPACE \subseteq EXPTIME$$

General Lower Bound Methods.

- Diagonalization

  List all Turing machines, $M_1, M_2, M_3, \ldots$, and list all possible inputs, x_1, x_2, $x_3, \ldots$ Define a new machine M so that $M(x_i) \neq M_i(x_i)$. Then M is not on the list of machines.

  Using the Diagonalization method, we can prove hierarchy theorems that say "more of some resource gives us more", e.g.

  $$DTIME[n] \subset DTIME[n^2],\ DSPACE[n] \subset DSPACE[n^2]$$

  $$NTIME[n] \subset NTIME[n^2],\ NSPACE[n] \subset NSPACE[n^2]$$

  However, with Diagonalization, all we can say about the general picture is that $NLOGSPACE \neq PSPACE$ and that $P \neq EXPTIME$.

Relation with Logic.

Theorem 1

$$FO \subseteq LOGSPACE$$

i.e., the set of first order definable Boolean queries is contained in $LOGSPACE$.

Proof: Let $\varphi \in FO$. We must construct a $LOGSPACE$ T.M. M_φ that takes as input a structure $\mathcal{A}$, and checks if $\mathcal{A} \models \varphi$. Let $\varphi = \exists x_1 \forall x_2 \ldots Q_k x_k \psi(x_1, \ldots, x_k)$. We show by induction on the structure of φ that φ is checkable in L. Assume inductively that ψ is checkable by a logspace machine M, where $\varphi = \exists x_1 \psi$. Using an extra $\log n$ bits of memory, we construct a machine M' which tests $\psi(i)$ for $i = 0, \ldots, n-1$. If M ever answers yes then M' accepts, otherwise it rejects. QED

We shall later see that the containment in the above theorem is proper.

The origins of Descriptive Complexity Theory can be traced back to a theorem by Ron Fagin from 1974.

Theorem 2 (Fagin,1974)

$$SO\exists = NP$$

Proof: ($\subseteq$): Let $\Phi = (\exists R_1) \ldots (\exists R_k)\, \psi(\vec{R})$, where R_i has arity a_i for $i = 1, \ldots, k$, and $\psi \in FO$. Note that it takes n^{a_i} bits to completely describe a relation of arity a_i on a universe of size n. An NP machine can thus guess $R_1, \ldots, R_k$ using polynomial nondeterministic time. Now, the problem reduces to checking if the expanded structure $(\mathcal{A}, R_1, \ldots, R_k) \models \Phi$. We have just seen that this is doable in L, and thus certainly in NP.
($\supseteq$): The proof is just a uniform version of Cook's 1971 theorem that SAT is NP-complete.

Let N be an arbitrary $NTIME[n^k]$ T.M. We want to build a $SO\exists$ sentence Φ such that $L(N) = MOD(\Phi)$. That is, for all $\mathcal{A}$,

$$N \text{ accepts } \mathcal{A} \Leftrightarrow \mathcal{A} \models \Phi$$

Φ will assert the existence of a computational tableau for N, which is an $n^k \times n^k$ array for inputs (a finite structure $\mathcal{A}$) of size n. Each cell contains a symbol from a finite alphabet $\Gamma = \Sigma \cup (Q \times \Sigma)$. Here $\sigma \in \Sigma$ represents a tape symbol and the pair (q, σ) represents that N is in state q and its head is reading symbol σ.

Let $\Gamma = \{\gamma_1, ..., \gamma_r\}$. We will use $2k$-ary relations $C_i(t_1, ..., t_k, S_1, ..., S_k)$, $i = 1, ..., r$, to represent $N's$ computation. Here the intuitive meaning of $C_i(t_1, ..., t_k, S_1, ..., S_k)$ is that at time $t_1, ..., t_k$, cell $S_1, ..., S_k$ contains symbol γ_i. (Note that we are using a k-tuple of variables, e.g. $t_1, ..., t_k$, to represent a number between 0 and $n^k - 1$. To do this we need a total ordering on the universe. If this is not present then Φ will first say $(\exists R)\,(R$ is a total ordering of the universe $\ldots$).)

Now, $\Phi = (\exists C_1 ... C_r)\,\omega(\vec{C})$ where $\omega(\vec{C})$ means "$\vec{C}$ is a valid accepting computation of N on the input $\mathcal{A}$".

We write ω as a conjunction

$$\omega = \alpha \wedge \beta \wedge \gamma \wedge \delta$$

where

$\alpha =$ "the 0th line of the computation is correct"

$\beta =$ "the first cell at the last step contains the accept symbol" (This may be coded as $C_r(\vec{m}, \vec{0})$, where m is the last element in the ordering, 0 is the first, and γ_r is the accept state looking at a block symbol).

$\gamma =$ "each cell contains exactly one symbol"

$\delta =$ "each line of the computation follows from the previous line via the rules of N"

All of the above are straight forward except for δ. To express δ, we must look at all rectangles of the form

$$\begin{array}{cccc} & \vec{s}-1 & \vec{s} & \vec{s}+1 \\ \vec{t} & a_{-1} & a_0 & a_1 \\ \vec{t}+1 & b_{-1} & b_0 & b_1 \end{array}$$

and assert that the bottom triple follows from the top triple via a move of N. This is simply a disjunction over all such 6-tuples and may be read from the state table of N

$$\delta \equiv (\forall \vec{s}\vec{t}) \bigvee_{a_{-1}a_0a_1 \to b_{-1}b_0b_1} (C_{a_{-1}}(\vec{s}-1, \vec{t}) \wedge C_{a_0}(\vec{s}, \vec{t}) \wedge C_{a_1}(\vec{s}+1, \vec{t})$$

$$\wedge\, C_{b_{-1}}(\vec{s}-1, \vec{t}+1) \wedge C_{b_0}(\vec{s}, \vec{t}+1) \wedge C_{b_1}(\vec{s}+1, \vec{t}+1))$$

While some details remain, this gives the main idea of the proof. QED

notes taken by A. Arratia, arratia@cs.wisc.edu

Session I: Tuesday, August 15
Track B: Descriptive Complexity
Speaker: N. Immerman

2 Complexity Classes and Logics

We are assuming finite structures whose universes are ordered. With that assumption, we characterize various complexity classes as we go along. The complexity classes we want to characterize in terms of definability of queries in various languages are as follows:

- **PSPACE**
- **PH**(The polynomial hierarchy)
- **NP**
- **P**
- **NL**
- **L**
- AC^0

We already saw that by Fagin's theorem $NP = SO\exists$. We also know that $PH = SO$ where $SO\exists$ means the sentences in the existential fragment of second order logic and SO means the sentences in second order logic.
At the end of this class we will see characterisations of P and NL and AC^0.

Elaboration on the assumption of Ordering in structures

We give examples from the class of ordered structures :

- Consider the class of finite binary strings. A binary string is an ordered structure. If $w = w_1 w_2 ... w_n$ is a binary string the corresponding ordered structure representation is given as $\mathcal{A}_w = \{\{1, ..., n\}, M, \leq\}$ where M is a unary relation that tells which of the bit positions are ones. $\leq$ is the ordering relation.
- Class of finite Graphs: As an ordered structure an n node graph is represented as $\mathcal{G} = \{\{0, .., n-1\}, \leq, E$ where E is the binary relation denoting the edges and $\leq$ is a total ordering on the vertices.

Note that a non-ordered string doesn't make sense. However, a graph with no ordering on the vertices makes perfect sense. For ordered structures, we will assume for convenience that we also have numeric constants 0 and m representing the minimum and maximum elements in the ordering. Now we shall see the characterisation of P on ordered structures in the following theorem (Immerman-Vardi theorem)

Theorem 3 *On ordered finite structures* $P = FO + LFP$ [1]

An immediate corollary follows:

Corollary 4 *Over ordered finite structures the following are equivalent:*

1. $P = NP$
2. $FO + LFP = SO$
3. $FO + LFP = SO\exists$

The proof of the corollary can be obtained by recalling the following fact. If $P = NP$ then $NP = co - NP$ and hence PH collapses to NP and if $P = PH$ then obviously $NP = P$.

Before we prove the Immerman-Vardi theorem, we give some more examples of **inductive definitions.**

- AGAP (Circuit value Problem) Given a graph with a relation A which lists the AND nodes in the graph (the nodes not in the relation are OR nodes) we want to know if there is a alternating path between two given vertices x and y. In relational terms we want to construct a binary relation ATC such that there are alternating paths between the nodes in each pair in that binary relation. Now $ATC(x, y) \equiv (x = y) \vee [(\exists z)(E(x, z) \wedge ATC(z, y)) \wedge (A(x) \rightarrow (\forall z)(E(x, z) \rightarrow ATC(z, y)))]$

  So we can give an inductive definition of ATC with the following first order formula:
  $\varphi(x, y, ATC) \equiv (x = y) \vee$
  $[(\exists z)(E(x, z) \wedge ATC(z, y)) \wedge (A(x) \rightarrow (\forall z)(E(x, z) \rightarrow ATC(z, y)))]$

  We get a corresponding operator Φ . A fixpoint for Φ exists by the *Tarski-Knaster* theorem, so $\Phi^{\infty} = ATC$.
  For an n node graph $G(V, E)$ $\Phi_G(\emptyset) = \{(x, x) \mid x \in V\}$.
  $\Phi^n_G(\emptyset) = ATC$. We say that LFP of Φ closes in $n = |V|$ steps. We say

[1] Recall that (FO + LFP) is first-order logic with the addition of a least fixed point operator (LFP) which may be applied only to positive formulas.

that n is the *closure ordinal* of φ.

$$AGAP = \{\mathcal{A} = (V, E, A, s, t) \mid \mathcal{A} \models ATC(s, t)\}$$

- Transitive closure.
  $TC(x, y) \equiv E(x, y) \vee (\exists z)(TC(x, z) \wedge TC(z, y))$
  So we define a fixpoint formula:
  $\alpha(x, y, R) \equiv E(x, y) \vee (\exists z)(R(x, z) \wedge R(z, y))$.
  This is the fast-transitive closure and the fix point operator closes in $\log n$ steps. $TC = \alpha^{\log n}$.

Now we prove the Immerman-Vardi theorem.

Proof: We first prove the easy part. We want to prove that on all ordered finite structures $FO + LFP \subseteq P$. Fix a given structure $\mathcal{A}$ let n be its size. Consider a formula $\varphi^\infty = LFP(\varphi(R))$. We have seen that $\varphi^\infty = \varphi^{n^k}$ where k is the arity of R. Thus to test if $\mathcal{A} \models \varphi^\infty$ we must evaluate the query φ polynomially many times. We have seen that each such evaluation is computable in logspace. Thus the whole evaluation can be done in polynomial time.

To prove the other we are given a $DTIME[n^k]$ Turing machine, M. We will write a formula $\Psi \in (FO + LFP)$ such that for any input structure $\mathcal{A}$, we have that

$$(\mathcal{A} \models \Psi) \quad \Leftrightarrow \quad (\mathcal{A} \in L(M))$$

This is very similar to yesterday's proof of Fagin's Theorem. Recall there that we wrote a formula

$$\Phi \;\equiv\; (\exists C_1 \ldots C_r)(\alpha \wedge \beta \wedge \gamma \wedge \delta)$$

where Φ asserts the existence of a valid accepting computation of a nondeterministic TM, N. Now, instead of asserting the existence of some $\bar{C}$, we define the unique C by induction. Consider the following system of inductive definitions:

$$\begin{aligned}
\psi_1(C_1, \ldots C_r, t_1, \ldots t_k, x_1, \ldots x_k) \quad &\equiv \quad (\alpha_1 \wedge \beta_1 \wedge \delta_1) \\
\psi_2(C_1, \ldots C_r, t_1, \ldots t_k, x_1, \ldots x_k) \quad &\equiv \quad (\alpha_2 \wedge \beta_2 \wedge \delta_2) \\
\vdots \qquad\qquad\qquad &\equiv \qquad\quad \vdots \\
\psi_r(C_1, \ldots C_r, t_1, \ldots t_k, x_1, \ldots x_k) \quad &\equiv \quad (\alpha_r \wedge \beta_r \wedge \delta_r)
\end{aligned}$$

This system inductively defines the computation of M on its input. Just as in Fagin's theorem $\bar{\alpha}$ asserts that the first line of the computation is correct, i.e., it corresponds to its input. This the the base of the induction. Next, $\bar{\beta}$ asserts that the last state is the accept state. Finally the inductive step consists of the formulas $\bar{\delta}$. In particular, $\delta_i(\bar{C}, \bar{t}, \bar{x})$ says that the symbols in the three cells at positions $\bar{x}-1, \bar{x}, \bar{x}+1$ lead to the symbol γ_i in one computational step of M. This is a disjunction over all triples of symbols that would lead to γ_i:

$$\delta_i(\bar{C}, \bar{t}, \bar{x}) \equiv \bigvee_{\gamma_a, \gamma_b, \gamma_c \rightarrow \gamma_i} (C_a(\bar{t}-1, \bar{x}-1) \wedge C_b(\bar{t}-1, \bar{x}) \wedge C_c(\bar{t}-1, \bar{x}+1))$$

$\square$

Although we are assuming the finite structures to be ordered, having the S (successor) relation is sufficient. So it might be a good exercise to see that S with $FO + LFP$ can give the order.

Exercise 5 *Define $\leq$ in $FO + S + LFP$.*

Next, we define a transitive closure operator, TC. Let $\varphi(\overline{x}, \overline{x'})$ be a formula in which the $2k$ free variables $x_1, \ldots, x_k, x'_1, \ldots x'_k$ may occur. φ may be thought of as a binary predicate on $k-tuples$. Let $TC_{\overline{x}, \overline{x'}}(\varphi)$ be a new formula denoting the *reflexive* and *transitive* closure of $\varphi(\overline{x}, \overline{x'})$. Let (FO + TC) be the closure of first-order logic using the TC operator. Let (FO + pos TC) be the subset of (FO + TC) in which only positive occurrences of TC are allowed, i.e. no negation symbols may be applied to formulas with TC occurring in them. Later we will see that over ordered structures, (FO + pos TC) = (FO + TC). For example, the formula $TC(E)(s, t)$ which expresses the GAP query is an element of (FO + TC), and in fact of (FO + pos TC).

Theorem 6 *Over ordered structures $NL = (FO + TC)$.*

Before proving this theorem we look at an immediate corollary of this.

Corollary 7 *Over ordered finite structures the following are equivalent.*

1. $NL = NP$
2. $(FO + TC) = SO$
3. $(FO + TC) = SO\exists$

The proof of this corollary is similar to that of Corollary 4.

Now we get into the proof of theorem 6. We will in fact prove that NL = (FO + pos TC), and then show later that over ordered structures this is equal to (FO + TC).

Proof: One direction is easy: $(FO + pos\, TC) \subseteq NL$. This is shown inductively: Let $\psi \equiv TC(\varphi)(\bar{s}, \bar{t})$ and assume inductively that $\varphi(\bar{x}, \bar{y})$ can be checked in NL. Note that in the base case, φ is first-order and thus in logspace. Then to test if ψ holds, we nondeterministically choose $\bar{a}_1$ and check that $\varphi(\bar{s}, \bar{a}_1)$ holds. Next we choose $\bar{a}_2$ and check that $\varphi(\bar{a}_1, \bar{a}_2)$ holds, and so on. If we ever reach $\bar{t}$ then we accept.

Now for the other direction we have to prove $NL \subseteq (FO + posTC)$ for ordered finite structures. Let N be an $NSPACE[k\log n]$ Turing machine. Let w be an input string. The input to our formula will thus be the structure $\mathcal{A}_w$ whose universe has size $n = |w|$. Using $k+2$ variables we can encode the configurations of N on input w. k variables each representing $\log n$ bits of the work tape, r a variable represnting the position of the read head and h another variable denoting the state, and the position of the work head.

So ID is typically of the form $ID = a_1, a_2, ..., a_k, r, h$. We next show how to write the formula $Next_N(ID_1, ID_2)$ whose meaning is that ID_2 follows from ID_1 via a move of N.

Then it follows that if $\bar{0}$ codes the initial ID, and $\bar{m}$ codes the accept ID, then

$$(w \in L(N) \quad \Leftrightarrow \quad (\mathcal{A}_w \models TC(Next_N)(\bar{0}, \bar{m}))$$

To write $Next_N$ we must make sure that our formulas have access to all the bits that N has. Note that $\mathcal{A}_w \models M(r)$ iff N's read head is looking at a one. To express which bit the work head is looking at, be need to express the predicate $\mathrm{BIT}(x,i)$ meaning the the i^{th} bit in the binary expansion of x is a one. Once this is done, the predicate $Next_N$ is just a listing out of the bounded number of possible moves from N's state table.

Exercise 8 *Prove that* $\mathrm{BIT}(x,i)$ *– meaning that the* i^{th} *bit of* x *is one – is expressible in* $(FO + posTC)$*, in the presence of the successor relation* S*. Note that* S *is first-order definable from* $\leq$*.*

To help you do the above, here is a first-order definition of addition. In a similar way, you can define multiplication and exponentiation, and thus BIT. Consider the following formula

$$\alpha(x_1, x_2, x_1', x_2') \equiv S(x_1, x_1') \wedge S(x_2', x_2)$$

Observe that $\mathrm{PLUS}(a,b,c) \equiv TC(\alpha)(a,b,c,0)$. $\square$

Next, we want to characterise the complexity class L via the **deterministic transitive closure.**

Definition 9 *Let* $\varphi(\overline{x}, \overline{x'})$ *be a formula with* $2k$ *free variables. Define the formula* φ_d *– the deterministic reduct of* φ *as follows. Note that* φ_d *cuts all multiple edges coming out of a vertex. Thus* φ_d *represents a binary edge relation on* k*-tuples that has out-degree one.*

$$\varphi_d(\overline{w}, \overline{w'}) \equiv \varphi(\overline{w}, \overline{w'}) \wedge (\forall \overline{z})(\varphi(\overline{w}, \overline{w'}) \rightarrow \overline{w'} = \overline{z}).$$

Now define the deterministic transitive closure as follows:

$$DTC(\varphi) \quad \equiv \quad TC(\varphi_d)$$

Theorem 10 *Over finite, ordered structures $L = (FO + DTC)$.*

The proof of Theorem 10 is very similar to the proof of Theorem 6. Note that for a deterministic Turing machine M, the relation $Next_M$ has outdegree one. Furthermore, the defintion of BIT also only uses transitive closure on relations of out-degree one.

For languages weaker than (FO + DTC) we often add the relation BIT to our first-order languages. Lindell has shown that BIT is first-order definable from plus and times and conversely, so this is a very natural numeric predicate. For structures with numeric predicates $\leq$ and BIT, first-order logic is a robust definition for a uniform version of the complexity class AC^0.

The following is an immediate corollary of Theorem 10:

Corollary 11 *Over finite ordered structures the following are equivalent:*

1. $L = NP$
2. $FO + DTC = SO$
3. $FO + DTC = SO\exists$.

(For further detail, most of the results of today's lecture are contained in the paper, "Language that capture complexity classes", SIAM Journal of Computing 16:4 (1987), 760-778.)

Now we arrive at some Normal form theorems :

Theorem 12 *Over finite structures with successor the complexity classes L, NL, P have normal forms as follows:*

1. $L = FO + DTC = DTC(\alpha)(\bar{0}, \bar{m})$.
2. $NL = FO + pos\, TC = TC(\alpha)(\bar{0}, \bar{m})$
3. $P = FO + pos\, ATC = ATC(\alpha)(\bar{0}, \bar{m})$

where α is a quantifier-free formula in a special form called a **quantifier free projection(qfp)**[2].

So we find that all queries in the corresponding complexity classes are reducible via qfps to 1GAP, GAP and AGAP problems. This leads to the following corollary to the above theorem.

Corollary 13 *1GAP, GAP and AGAP are* **complete** *for the classes L, NL and P respectively via quantifier free reductions.*

[2]Projections are a very weak form of complexity theoretic reduction due to Valiant. qfp's are quantifier-free reductions that are also projections. These will be defined tomorrow.

Another corollary follows immediately:

Corollary 14 *The following are equivalent:*

1. $L = NP$
2. *3 colorability is expressible as* $DTC(\alpha)(\bar{0}, \bar{m})$ *where* α *is a quantifier free projection.*

We finish this lecture by mentioning the **space complementation** theorem of Immerman and Szelepcsényi.

Theorem 15 *For all* $S(n) \geq \log n$, $NSPACE[S(n)] = co - NSPACE[S(n)]$.

Note that it follows from Theorem 12 that NL is closed under complementation iff $\overline{GAP}$ is expressible in the form $TC(\alpha)(\bar{0}, \bar{m})$, with α is a quantifier-free formula. In fact, my original proof was a qfp on eight-tuples, where, for example, the tuple $(1, d, n_d, 0, 0, 0, 0, 0)$ is reachable from $\bar{0}$ iff there are exactly n_d vertices reachable from vertex 0 in the original graph.

The corollary of Theorem 15 is interesting because this shows that the context-sensitive languages are closed under complementation. This was an open problem since the 1960's.

Corollary 16 $NL = co - NL$
$CSL = co - CSL = NSPACE[n]$

notes taken by Sandeep K. Shukla, sandeep@cs.albany.edu

Session III: Wednesday, August 16

Track B: Descriptive Complexity

Speaker: N. Immerman

3 The Normal Form Theorem

As usual, we'll always assume our structures are ordered or, better yet, come equipped with a successor function.

Theorem 17 (Immerman) *The following hold:*

$$\begin{array}{rcccl} L & = & (FO + DTC) & = & DTC(\alpha)(\bar{c}) \\ NL & = & (FO + TC) & = & TC(\alpha)(\bar{c}) \\ P & = & (FO + ATC) & = & ATC(\alpha)(\bar{c}) \end{array}$$

where in each case α is a quantifier-free projection.

Here "quantifier-free projection" (qfp) means that α is of the form

$$\gamma_0 \vee (\gamma_1 \wedge \delta_1) \vee \ldots \vee (\gamma_n \wedge \delta_n)$$

where each γ_i is quantifier-free *numeric* (that is, built up from equality and successor) and each δ_i is a *literal* (that is, an input predicate or its negation). If the γ_i are instead first-order numeric, call this a "first-order projection" (fop) instead.

Corollary 18 *$1GAP$, GAP, and $AGAP$ are complete for L, NL, and P via qfp reductions (to be defined below).*

4 Reductions in Complexity Theory

Since the 70's successively stronger reductions have been considered. The following is a brief overview of their history:

1. (Cook, 71) proved that SAT is NP-complete via polynomial time *Turing* reductions.
2. (Karp, 72) showed that a number of important optimization problems are complete via polynomial time many-one reductions.
3. (Jones, 74) showed that SAT is NP-complete via logspace reductions.
4. (Hartmanis, Immerman, Mahaney, 79) showed that most of the standard examples are in fact NP-complete via 1-way logspace reductions.

5. (Lovász and Gács, 77) showed that SAT is NP-complete via fop's.

6. (Dahlhaus, 84) showed that SAT is NP-complete via qfp's.

One motivation for getting the reductions as tight as possible is that it may help in proving lower bounds. For example, if $L = NP$ then 3-colorability is qfp-reducible to $1GAP$, which seems very unlikely.

So far, we've defined qfp and fop formulas. We now define the corresponding reductions.

Definition 19 *Let σ and τ be signatures,*

$$\tau = (R_1^{n_1}, \dots, R_s^{n_s}).$$

Let $\mathcal{S}$ be a class of σ-structures and $\mathcal{T}$ be a class of τ-structures. Then a first-order reduction *I from $\mathcal{S}$ to $\mathcal{T}$ is a map from the set of σ-structures to the set of τ-structures given by first-order formulas*

$$\langle \varphi_1(x_{11}, \dots, x_{n_1 k}), \dots, \varphi_s(x_{11}, \dots, x_{n_s k}) \rangle$$

such that

$$\mathcal{A} \in \mathcal{S} \Leftrightarrow I(\mathcal{A}) \in \mathcal{T}.$$

Here $I(\mathcal{A})$ is defined as follows: if $\mathcal{A}$ is the structure

$$(\{0, \dots, n-1\}, Q_1, \dots, Q_t)$$

then $I(\mathcal{A})$ denotes the structure

$$(|A|^k = \{0, \dots, n^{k-1}\}, R_1^{I(\mathcal{A})}, \dots, R_s^{I(\mathcal{A})})$$

where

$$R_i^{I(\mathcal{A})} = \{(\bar{a}_1, \dots, \bar{a}_{n_i} \mid \mathcal{A} \models \varphi_i(\bar{a}_1, \dots, \bar{a}_{n_i})\}.$$

Note that elements of A correspond to k-tuples of $I(A)$, but otherwise I corresponds to the usual logical notion of an *interpretation*. For example, the following gives a first-order reduction of connectivity to GAP:

$$\varphi(x, y, x', y') \equiv (x = x' \wedge E(y, y')) \vee (suc(x, x') \wedge y = x' = y').$$

For example, if $\mathcal{G}$ is the undirected graph in Figure 1, $I(\mathcal{G})$ is the directed graph in Figure 2.
We'll leave it to the reader to verify that in general $\mathcal{G}$ is connected iff there is a path from the first to last node of $I(\mathcal{G})$.

If the formulas φ_i are given by fop's (resp. qfp's), call the reduction an fop (qfp) reduction. From a circuit point of view, such a reduction tells you how to build a circuit for $\mathcal{S}$ on inputs of size n given a circuit for $\mathcal{T}$ for inputs of size n^k with inputs numbered $\bar{x} = (x_1, \dots, x_k)$; the clause $\gamma_i(\bar{x}) \wedge \delta_i(\bar{x})$ says "if $\gamma_i(\bar{x})$ is true, then connect the input number $\bar{x}$ of the circuit for $\mathcal{T}$ and attach it to the input corresponding to the literal $\delta_i(\bar{x})$."

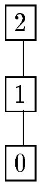

Figure 1: Undirected Graph

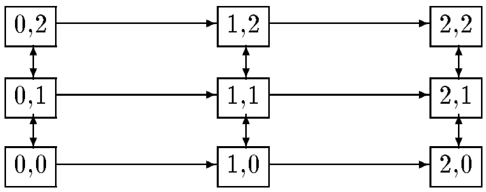

Figure 2: Directed Graph Constructed by the Reduction

5 Defining the Classes $FO[t(n)]$, $IND[t(n)]$, and $CRAM[t(n)]$

Towards defining $FO[t(n)]$ we'll begin with an example. Consider the predicate

$$Path(x,y) \equiv E(x,y) \vee \exists z(Path(x,z) \wedge Path(z,y))$$

expressing the existence of a path from x to y. Using the notation

$$(\exists x.\varphi)\alpha \equiv \exists x(\varphi \wedge \alpha)$$

and

$$(\forall x.\varphi)\alpha \equiv \forall x(\varphi \Rightarrow \alpha)$$

we can write

$$\begin{aligned} Path(x,y) &\equiv (\forall z.\neg E(x,y))(\exists z)(Path(x,z) \wedge Path(z,y)) \\ &\equiv (\forall z.\neg E(x,y))(\exists z)(\forall u,v.(u=x \wedge v=z) \vee (u=z \wedge v=y)) \\ &\qquad (Path(u,v)) \\ &\equiv (\forall z.\neg E(x,y))(\exists z)(\forall u,v.(u=x \wedge v=z) \vee (u=z \wedge v=y)) \\ &\qquad (\exists x,y.x=u \wedge y=v)(Path(x,y)). \end{aligned}$$

In other words, we've reworked the formula so that it consists of a quantifier block followed by the single relation $Path(x,y)$. That this can be done in general amounts to the following

Lemma 20 *If $\varphi(R, x_1, \ldots, x_k)$ is FO and R-positive, it can be written in the form*

$$\varphi \equiv [(Q_1 z_1.M_1)\ldots(Q_l z_k.M_k)]R(x_1, \ldots, x_k)$$

where the formulas M_i are quantifier-free and R-free.

Definition 21 *We say that $\mathcal{S} \in FO[t(n)]$ if there is a quantifier block $[QB]$ and a quantifier-free formula M_0 such that*

$$\mathcal{S} = \{\mathcal{A} \mid \mathcal{A} \models [QB]^{t(|\mathcal{A}|)} M_0\}.$$

So, for example $GAP \in FO[\log n]$ since

$$GAP = \{\mathcal{G} \mid \mathcal{G} \models [QB_T]^{\log(|\mathcal{G}|)}(0 \neq 0)\}$$

where QB_T is as in the example above.

Corollary 22 *If $\varphi(R, x_1, \ldots, x_k)$ is FO and R-positive, then*

$$\varphi^{\infty} = [QB]^{n^k}(x_1, \ldots, x_n).$$

Definition 23 *$IND[t(n)]$ is the restriction of $(FO + LFP)$ to induction of depth $O(t(n))$ (i.e. worst case depth on all structures).*

Definition 24 *$CRAM[t(n)]$ is the set of queries that are computable by CRCW PRAM using time $O(t(n))$ and $n^{O(1)}$ hardware.*

So in the $CRAM$ model there are polynomially many processors P_i, each with an ID number, basic instructions (such as $+$, $=$, and shift left), and some finite registers. The processors share a common memory and run synchronously in concurrent read, concurrent write mode. (With concurrent write, we need to choose how to resolve conflicts. "Priority" means that the lowest processor number wins; "common" means that all the processors promise to write the same value; "arbitrary" means that the winner is unspecified. In the settings we consider, all modes yield the same computational strength.)

If we want to specify something other than polynomial hardware (processors+memory), we'll write

$$CRAM[t(n)]\text{-}HARD[s(n)].$$

In the next lecture we'll prove that if $t(n)$ is polynomially bounded and space constructible, then

$$IND[t(n)] = FO[t(n)] = CRAM[t(n)].$$

(To make the above true for $t(n) < \log n$ we need to add the numeric predicates addition and multiplication to first-order logic.)

In fact,

$$CRAM[t(n)]\text{-}HARD[n^v] \approx FO[t(n)]\text{-}VAR[v]$$

so that parallel time corresponds to iteration depth and hardware corresponds to number of variables.

notes taken by J. Avigad, avigad@math.lsa.umich.edu

Session IV: Thursday, August 17

Track B: Descriptive Complexity

Speaker: N. Immerman

6 Historical Background

Definition 25 *$FOSIZE(s(n))$ is the set of queries definable by a uniform sequence of FO formulas $\varphi_1, \varphi_2, \ldots$ of size $s(n)$. In other words, if*

$$\mathcal{S} \in FOSIZE(s(n))$$

then for some such sequence

$$\mathcal{S} = \{\mathcal{A} \mid \mathcal{A} \models \varphi_{|\mathcal{A}|}\}.$$

Theorem 26 (Immerman 79) *For $s(n) \geq \log n$,*

$$NSPACE[s(n)] \subseteq FOSIZE[\frac{s(n)^2}{\log n}] \subseteq DSPACE[s(n)^2].$$

Theorem 27 (Immerman, FOCS 80) *The following hold:*

$$\begin{array}{rcccl} NL \subseteq AC^1 & = & FOSIZE[\log n]\text{-}VAR[O(1)] & = & FO[\log n] \\ P & = & FOSIZE[n^{O(1)}]\text{-}VAR[O(1)] & = & FO[n^{0(1)}] \\ PSPACE & = & FOSIZE[2^{n^{O(1)}}]\text{-}VAR[O(1)] & = & FO[2^{n^{0(1)}}] \end{array}$$

A fundamental question in the field of descriptive complexity is this: what is the tradeoff between the number of variables and quantifier rank (i.e. the depth of the quantifier block iteration)? The number of variables generally corresponds to hardware (the number of gates in a circuit, or the number of processors in a $CRAM$) and the quantifier rank corresponds to parallel time.

7 The Relationship to Infinitary Logic

For structures with numbers, $FO[t(n)] \subseteq L^{\omega}_{\infty,\omega}$ since any $\varphi \in FO[t(n)]$ can be written

$$\varphi = \bigvee_{n=1}^{\infty} (\text{“the universe has size } n\text{”} \wedge [QB]^{t(n)} M_0)$$

where the quantifiers in $[QB]$ become finite conjunctions and disjunctions. (Without numbers, evenness is easily $FO[t(n)]$ expressible, but not $L^{\omega}_{\infty,\omega}$ expressible.) Over *ordered* structures $L^{\omega}_{\infty,\omega}$ is too strong; in fact, any graph query can be expressed in $L^{2}_{\infty,\omega}$.

Proposition 28 *Let Q be any query over finite, ordered structures of vocabulary τ. Let k be the maximum of 2 and the arity of any relation symbols in τ. Then $Q \in L^k_{\infty,\omega}$.*

To prove this, first prove the following:

Lemma 29 *Over ordered structures, for each i there is a formula*

$$N_i(x) \in L^2 (= L^2_{\omega,\omega})$$

expressing the notion "x is the ith element in the ordering"

In fact, $N_i(x)$ can be written with quantifier rank i. With one more variable (i.e. in L^3) it can be written with quantifier rank $\log i$.

Using the lemma, for every structure $\mathcal{A}$ we can write down a formula $CD(\mathcal{A})$ giving a "complete description" of $\mathcal{A}$, and then express any query Q as

$$\bigvee_{\mathcal{A} \in Q} CD(\mathcal{A}).$$

8 Relating the Main Classes

The theorems in this section relate some of the main complexity classes introduced so far, including $IND[t(n)]$, $FO[t(n)]$, and $CRAM[t(n)]$.

Theorem 30 (Chandra, Stockmeyer, Vishkin 84) *The following holds:*

$$\textit{non-uniform-}CRCW\ PRAM[t(n)] = \textit{non-uniform-}AC[t(n)].$$

Non-uniform PRAMS mean that we may have a different polynomial-size program for each size input. Similarly non-uniform circuits come with no assumption about how (intellectually) hard it is to build the nth circuit. Note that $CRAM[t(n)]$ is a particular uniform version of $CRCW\ PRAM[t(n)]$.

Theorem 31 (Immerman, SIAM J. Computing 89) *Let $t(n)$ be space constructible. Over ordered, finite structures we have:*

1. *If $t(n)$ is polynomially bounded, then*

$$IND[t(n)] = FO[t(n)] = CRAM[t(n)].$$

2. *In general,*

$$ITER[t(n)] = FO[t(n)] = CRAM[t(n)].$$

Recall that $CRAM[t(n)]$ implicitly assumes polynomial hardware. We haven't defined $ITER[t(n)]$ yet; but it means an unbounded iteration operator, iterated $t(n)$ times. IND uses the LFP operator and thus is syntactically forced to be polynomially bounded. ITER uses the PFP operator and is thus not so restricted.

Corollary 32 *The following hold:*

$$PH = CRAM[O(1)]\text{-}HARD[2^{n^{O(1)}}]$$

and more generally

$$SO[t(n)] = CRAM[t(n)]\text{-}HARD[2^{n^{O(1)}}].$$

The corollary follows since with 2^{n^k}-hardware each processor has an identification number that is a string of length n^k. Given some second-order predicates, each processor (or, really, a clump of polynomially many) can check the first-order part of the query on the predicates corresponding to its ID in constant time.

Note, incidentally, that $SO[n^{O(1)}] = PSPACE$.

Using similar methods one can show that AC^k (the class of languages accepted by uniform $O((\log n)^k)$-depth unbounded fanin circuits) is the same as the class $FO[(\log n)^k]$.

9 The Proof of Theorem 31

The inclusion

$$IND[t(n)] \subseteq FO[t(n)].$$

follows in a straightforward way from the definition. To show that

$$FO[t(n)] \subseteq CRAM[t(n)],$$

let

$$QB \equiv (Q_k x_k.M_k)\ldots(Q_1 x_1.M_1)$$

be a quantifier-block; we need to describe a $CRAM$ program to evaluate

$$\varphi \equiv [QB]^{t(n)}.M_0$$

in $O(t(n))$ steps.

Build a $CRAM$ with n^k processors, each one labelled with an ID number $a_1a_2\ldots a_k$ $(0 \leq a_i \leq n-1)$, and n^{k-1} bits of global memory M. Let

$$a_1a_2\ldots\hat{a}_m\ldots a_k$$

denote the string of the a_i with a_m left out; we will use these to address memory locations.

We will design our program to run in $kt(n)$ stages, so that at stage t, letting $f(t)$ denote the value of $t(\mathrm{mod}\, k)$:

$$M(a_1\ldots\hat{a}_{f(t)}\ldots a_k) = 1 \Leftrightarrow \mathcal{A} \models \varphi_t(x_1/a_1,\ldots x_k/a_k)$$

where
$$\varphi_t = (Q_{f(t)}x_{f(t)}.M_{f(t)})\dots(Q_{f(1)}x_{f(1)}.M_{f(1)})M_0$$
(in other words, the innermost t-many quantifiers of φ).

Inductively, we have

$$\varphi_{t+1} = (Q_{f(t+1)}x_{f(t+1)}.M_{f(t+1)})\varphi_t.$$

At stage $t+1$, each processor $a_1a_2\dots a_k$ does the following (assuming $Q_{f(t+1)} = \exists$; the $\forall$ case is similar):

1. $b \leftarrow M(a_1 \dots \hat{a}_{f(t)} \dots a_k)$
2. $M(a_1 \dots \hat{a}_{f(t+1)} \dots a_k) \leftarrow 0$
3. If $b \wedge M_{f(t)}(a_1, \dots, a_k)$ then $M(a_1 \dots \hat{a}_{f(t+1)} \dots a_k) \leftarrow 1$

The idea is that each processor checks the formula

$$(\exists x_{f(t+1)}.M_{f(t+1)}).\varphi_t(a_1, \dots, a_k)$$

on its ID number, where $x_{f(t+1)}$ is set to $a_{f(t+1)}$, and reports success if

$$M_{f(t+1)} \wedge \varphi_t$$

holds there.

Finally, we need to show that

$$CRAM[t(n)] \subseteq IND[t(n)].$$

That means that we have to show how to simulate a $CRAM$ computation with an $IND[t(n)]$ formula, as in Fagin's simulation of NP computations with Σ^1_1 formulas. In other words, we have to give an inductive definition of the computation process that closes after $t(n)$ iterations.

In the $CRAM$ model there are polynomially many processors and memory bits to keep track of, and, as usual, they can be numbered by tuples. Letting the tuple $\bar{x}$ label a bit in a processor register or common memory, we want to describe the relation
$$contents_t(\bar{x}, b)$$
stating that "the bit labelled by $\bar{x}$ is b" (allowing Boolean variables for convenience). The main chore is to define

$$contents_{t+1}(\bar{x}, b) = \varphi(contents_t(\bar{x}, b))$$

where $\varphi \in FO$, in which case the fixed point φ^∞ will describe the final configuration. The details are tedious but routine.

notes taken by J. Avigad, avigad@math.lsa.umich.edu

Session I: Friday, August 18
Track B: Descriptive Complexity
Speaker: N. Immerman

10 A Lower Bound

We have seen from Phokion's lectures where all the descriptive classes are, and from mine where they are all placed in the presence of *ordering*. We have also seen how, starting with FO and iterating quantifier blocks, we may obtain the classes $\mathrm{AC}^0, \mathrm{NC}, \mathrm{P}$ and PSPACE. We know how to prove lower bounds in the absence of ordering, but they do not tell us much about computation. For example, in the absence of ordering, here is a theorem that I will state without proof, that proves a lower bound on the formula size for a natural computational problem:

Theorem 33 (Immerman, FOCS 1979) *The alternating graph accessibility problem, AGAP, does not have a formula of size* $2^{\sqrt{\log n}}$.

Recall that $(\log n)^{O(1)} < 2^{\sqrt{\log n}} < n^\epsilon$ for any $\epsilon > 0$. Recall also that in this model, we allow the formula size to grow with the input size n. AGAP has a formula of size $O(n)$, in fact, it is in $\mathrm{FO}[n]$. Theorem 33 was proved by exhibiting graphs A_n, B_n such that $A_n \in AGAP$, $B_n \notin AGAP$, yet they agree on all formulas of size $2^{\sqrt{\log n}}$. This is quite a strong lower bound, and suggested the following: if we could show $AGAP \notin \mathrm{FO}[2^{\sqrt{\log n}}]_\leq$, that is, *in the presence of ordering*, then we would have separated L, NL, and NC from P.

Of course, why do we need ordering to express order-independent properties? This brought me to the question asked by Chandra and Harel and by Gurevich: is there a logic for order-independent polynomial-time properties? The example of even cardinality, which is an order-independent polynomial-time property that is not in FO + LFP, led me to the following very natural conjecture ("COUNT" below stands for allowing a counting quantifier, which will be explained shortly).

Conjecture 34 (Immerman, STOC 1982) FO + LFP + COUNT = *order-independent* P.

As Phokion mentioned, we don't even know how to enumerate the latter class, while, on the other hand, there does exist a recursive enumeration of all polynomial-time properties of ordered graphs: simply list all polynomial-time Turing machines – TM's M_i with explicit $n^{f(i)}$ clocks!
NB: For the rest of this lecture, ordering will be absent (unless explicitly mentioned).

We will now introduce a very important problem in connection with order-independent properties—the Graph Canonization problem (GCAN). The basic idea is to come up with a function f that produces a canonical form for every graph such that two graphs are isomorphic iff their canonical versions are the same: $\widehat{G} = f(G)$ such that $\widehat{G} \cong G$ and if $G_1 \cong G_2$ then $f(G_1) = f(G_2)$.

Proposition 35 *If graph canonization can be done in polynomial time, then we have a logic for order-independent polynomial-time properties.*

Proof. It suffices to prove that there is a recursive enumeration of all polynomial-time Turing machines that accept order-independent polynomial-time properties of graphs. Simply attach a "canonization module" to each polynomial time TM: on input $G = (V, E, \leq)$, compute $\widehat{G} \doteq (\widehat{V}, \widehat{E}, \widehat{\leq}) = f(G)$ and then run the Turing machine on $(\widehat{V}, \widehat{E}, \widehat{\leq})$. $\square$

Corollary 36 *If there is no recursive enumeration of all polynomial-time Turing machines that accept order-independent graph properties, then* $\mathrm{P} \neq \mathrm{NP}$.

Proof. The corollary follows from the fact that Graph Canonization can be done in the polynomial-time hierarchy, in fact in $\mathrm{FP}^{\mathrm{NP}}$. It follows that if P were equal to NP, then PH would also be equal to NP, and thus graph canonization would be doable in P. The idea: given a graph G on n vertices as an n^2-bit adjacency matrix encoding, output the lexicographically smallest n^2-bit adjacency matrix that represents a graph isomorphic to G. This can be done by a binary search over all n^2-bit strings, with oracle access to the following problem from NP: Let $\mathcal{O}$ be the set of pairs (A, G) such that there exists an adjacency matrix isomorphic to G that is lexicographically less than A:

$$\mathcal{O} \;=\; \big\{(A, G) \;\big|\; (\exists A')(A' < A \text{ and } A' \cong G)\big\}$$

$\square$

The preceding proposition and corollary intimately relate logics for order-independent properties of graphs with the P vs. NP problem and the complexity of the Graph Isomorphism problem.

Let us get back to the conjecture that asserts $(\mathrm{FO} + \mathrm{LFP} + \mathrm{COUNT}) =$ order-independent P. We will first introduce the counting quantifier formally and introduce the language $\mathcal{C}^k$, which is $\mathcal{L}^k$ with the counting quantifier. (Recall that $\mathcal{L}^k$ is merely first-order logic restricted to k variables.) We will assume that graphs are presented as two sorted structures—"vertices" $\{v_0, v_1, \ldots, v_{n-1}\}$ and "numbers" $\{0, \ldots, n-1\}$,—with the usual edge relation E on the vertices and the relation $\leq$ on the numbers. Informally, this can be thought of as an intermediate notion between unordered graphs and ordered graphs. It is important to note that the numbers are not associated with the vertices.

Using the fact that PLUS can be written in $\mathrm{FO} + \mathrm{LFP}$, it is clear, even without new quantifiers, that we can now write a first-order sentence that says "the size of the universe is even," something we couldn't do with FO + LFP. However, we can't (yet) say "the size of the edge set is even." To enhance the expressive power, we will introduce the counting quantifier "$(\exists ix)$," where i ranges over the numbers and x over the vertices. The meaning of "$(\exists\ i\ x)\varphi(x)$" is that there are at least i x's such that $\varphi(x)$ holds. In what follows, we will use the letters i, j, k to range over the numbers, and x, y, z to range over the vertices.

It is a useful to note that since FO+LFP = P, any polynomial-time predicate on the numbers is now freely available for use. In particular, we can express addition and multiplication of numbers. For example, the property "there are an even number of vertices with self-loops" can be written as $(\exists i)(\exists j)(i + i = j)[(\exists\ j\ x)E(x, x) \wedge \neg(\exists\ j + 1\ x)E(x, x)]$. Finally, the language $\mathcal{C}^k$ is defined to be FO + COUNT with k variables (which, of course, may be reused), and, for convenience, the constants $0, \ldots, n - 1$.

The basis of the conjecture was that we have added precisely what was (or what was noted to be) missing, namely counting.

The Ehrenfeucht–Fraïssé game for $\mathcal{C}^k$ is an extension of the $\mathcal{L}^k$ game. The *Spoiler* has k red pebbles $r_1, \ldots, r_k$, and the *Duplicator* has k green pebbles $g_1, \ldots, g_k$. The game is the same as the r-move k-pebble game on graphs G and H, except for the extra *counting move*, which has two phases. At any move i, the Spoiler may do the following (the first phase): pick up pebble r_j (corresponding to the j-th variable), and choose a set $S \subseteq |G|$ or $S \subseteq |H|$. The Duplicator then has to pick up pebble g_j and choose some set S' of the other structure ($|H|$ or $|G|$) such that $|S'| = |S|$. The intuitive idea is that the Spoiler maintains that the graph he chooses has a set S with certain properties, and the Duplicator must demonstrate that the other graph has a set S' of the same cardinality satisfying the same properties. In the second phase of the move, the Spoiler places r_j on some element of S' (he challenges the Duplicator's choice by pointing out a difference), and the Duplicator responds by placing g_j on some element of S (she exhibits a correspondence between S' and S). The definition of winning and strategies are just as before. The following theorem, whose proof mirrors the proof for $\mathcal{L}^k$, brings out the importance of the EF game for $\mathcal{C}^k$.

Theorem 37 (Immerman, Lander, 1990) *The Duplicator wins the r-move $\mathcal{C}^k$ game on G and H if and only if G and H satisfy the same* (FO + COUNT)*-sentences with k variables and quantifier rank r, that is, $G \equiv_r^k H$ in $\mathcal{C}^k$.*

As an example, let us compute the $\mathcal{C}^2$ types of the vertices of the undirected graph in Figure 3. The $\mathcal{C}_0^2$ types are just all the eight vertices, since they are all indistinguishable. The $\mathcal{C}_1^2$ types correspond to vertices with the same degree: $\{4, 7\}, \{1, 2, 5, 8\}$, and $\{3, 6\}$. The $\mathcal{C}_2^2$ types count, for each vertex, the number of neighbors of each color that the vertex has. Here they are $\{4, 7\}, \{1, 2, 5, 8\}$, and $\{3, 6\}$. Noting that $\mathcal{C}_2^2 = \mathcal{C}_1^2$, we declare the $\mathcal{C}^2$ type of the graph to be "stable."

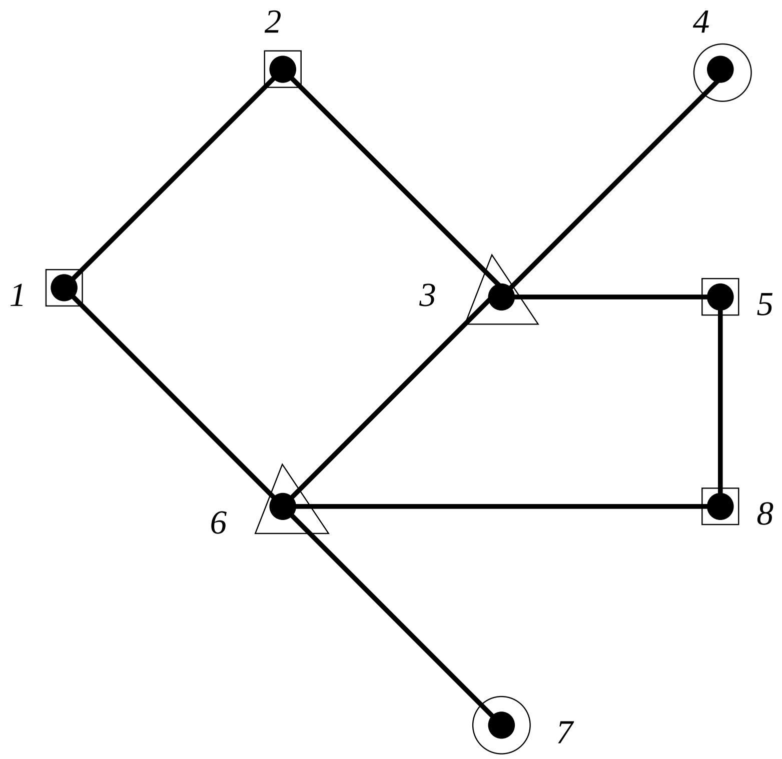

Figure 3: Illustration of $\mathcal{C}^2$ types

It turns out that this notion connects to a lot of independent combinatorial work on the Graph Isomorphism problem. In particular, the identification of the $\mathcal{C}^2$ types corresponds to the so-called "vertex-refinement" algorithm, which is a special case of the k-dimensional Weisfeiler–Lehman method (with $k = 1$). (Scribe note: For an interesting account of these connections—in particular, a proof that $\mathcal{C}^{k+1}$ equivalence of two graphs is precisely the same as equivalence in the k-dimensional Weisfeiler–Lehman method—see the paper by Cai, Fürer, and Immerman. This paper first appeared in FOCS 1989, and then an expanded journal form appeared in *Combinatorica*, **12**(4), 1992, pages 389–410. The latter version is also available from Prof. Immerman's homepage.) Below we mention some of these interesting results.

Theorem 38 (Babai and Kučera, FOCS 1979) *With probability $1-\alpha^n$ for some small $\alpha < 1$, a random graph on n vertices has a unique $\mathcal{C}^2_4$ description of each vertex.*

Corollary 39 *Almost all graphs are rigid.*

Corollary 40 *For almost all graphs, we can solve Graph Isomorphism in $O(n^3)$ time, or in 4 steps on an ultracomputer (in $\mathcal{C}^2$, or in $\mathrm{FO} + \mathrm{LFP} + \mathrm{COUNT}$ with 2 variables).*

The following proposition shows the time complexity of identifying the $\mathcal{C}^2$-types of the vertices of a graph. It uses the "partition" algorithm, which was originally motivated by DFA state minimization.

Proposition 41 (Hopcroft, see Aho, Hopcroft, and Ullman, §4.13) *Vertex refinement (identification of $\mathcal{C}^2$-types) can be done on a RAM in $O(n^2 \log n)$ time.*

In general, refinement of $k-1$ tuples of vertices, that is, identifying the $\mathcal{C}^k$-types, can be done in $O(n^k \log n)$ time, hence in polynomial time for any fixed k. An important open question in the combinatorial literature, in particular, the Russian school of Graph Isomorphism, couched in our language, is whether there is a fixed k such that $\mathcal{C}^k$-equivalence precisely captures graph isomorphism.

Conjecture 42 (Weisfeiler) *There exists a k such that for all G, H, $G \equiv^{\mathcal{C}^k} H$ if and only if $G \cong H$.*

Corollary 43 (of Weisfeiler's conjecture) *Graph Isomorphism can be solved in $O(n^k \log n)$ for some fixed k, and hence in polynomial time.*

The theorem of Babai and Kučera implies that for *almost all* graphs G and H, graph isomorphism can be solved in $O(n^2 \log n)$ time! That is, for almost all G, H, $G \equiv^{\mathcal{C}^k} H \Rightarrow G \cong H$. Also, Babai and Luks (STOC 1983) showed that Graph Isomorphism can be solved for graphs of fixed color class size in polynomial time. Moreover, for graphs G, H of color class size at most 3, $G \cong H$ if and only if $G \equiv^{\mathcal{C}^3} H$. However, little was known about the correct answer to Weisfeiler's conjecture—it was even open whether $\mathcal{C}^4$ would suffice. Much work exists along this line (see Cai, Fürer, and Immerman, 1992), including a conjecture of Immerman and Lander that $\mathcal{C}^{\log n}$ suffices, and hence Graph Isomorphism can be solved in $O(n^{\log n})$ time. All these conjectures were finally refuted by the work of Cai, Fürer, and Immerman:

Theorem 44 (Cai, Fürer, and Immerman, FOCS 1989, Combinatorica 1992) *There is an order-independent property S of graphs of color class size 4, and an $\epsilon > 0$, such that S can be decided in logspace, but S cannot be expressed in $\mathcal{C}^{\epsilon n}$.*

That is, we need $\Omega(n)$ variables to express S in first-order with counting, and thus in particular we have as a corollary that $S \in \mathrm{L} - \mathcal{C}^{\omega}_{\infty\omega}$, since two structures are $\mathcal{C}^k$-equivalent if and only if they satisfy the same $\mathcal{C}^{\omega}_{\infty\omega}$ formulas. (Both of these conditions are equivalent to the Duplicator winning the r-move $\mathcal{C}^k$ game for every r.) Since $(\mathrm{FO} + \mathrm{LFP} + \mathrm{COUNT}) \subseteq \mathcal{C}^{\omega}_{\infty\omega}$, we have:

Corollary 45 *$S \in \mathrm{L} - (\mathrm{FO} + \mathrm{LFP} + \mathrm{COUNT})$.*

Hella (LICS, 1992) has strengthened Theorem 44 and shown that, in fact, FO+LFP+COUNT with all "generalized quantifiers" of arity 1 cannot express order-independent Logspace properties.

Before we proceed to sketch the proof of Theorem 44, we will comment about another aspect of $\mathcal{C}^k$. Let φ^∞ be an expression in $(\mathrm{FO}+\mathrm{LFP}+\mathrm{COUNT})_k$ (that is, it is expressible in this logic with k variables). Then for each n, there is a *$k - variable$* formula that expresses this query for structures of size at most n, and hence $\varphi^\infty = \bigvee_{n=1}^{\infty} \varphi^n$, where each of the $\varphi^n \in (\mathrm{FO} + \mathrm{COUNT})_k$ is the restriction to structures of size $\leq n$. That makes the following proposition obvious:

Proposition 46 *If $A \equiv^{\mathcal{C}^k} B$, then $A \equiv^{(\mathrm{FO}+\mathrm{LFP}+\mathrm{COUNT})_k} B$, that is, the $\mathcal{C}^k$ logic is stronger than $(\mathrm{FO} + \mathrm{LFP} + \mathrm{COUNT})_k$.*

The rest of this lecture is devoted to sketching a proof of Theorem 44. Fix k. For every n, we will produce graphs G_n and H_n such that $G_n \in S$, $H_n \notin S$, hence $G_n \not\cong H_n$, $|G_n| = |H_n| = O(n)$, yet $G_n \equiv^{\mathcal{C}^n} H_n$. These graphs will have color class size 4, and will be "almost ordered." The crucial part of the construction is the following gadget X_3, shown in Figure 4. The vertex-pairs $(a_1, b_1), (a_2, b_2)$, and (a_3, b_3) are each assigned a distinct color, and the middle four vertices are assigned another, different, color. Coloring is only a conceptual tool to enforce that automorphisms/isomorphisms have a simple structure (since they must preserve the color). The coloring can be replaced by simple gadgets. The important property of X_3 is that the only automorphisms of X_3 are obtained by exchanging exactly zero or two of the a_i's with their corresponding b_i's. (This

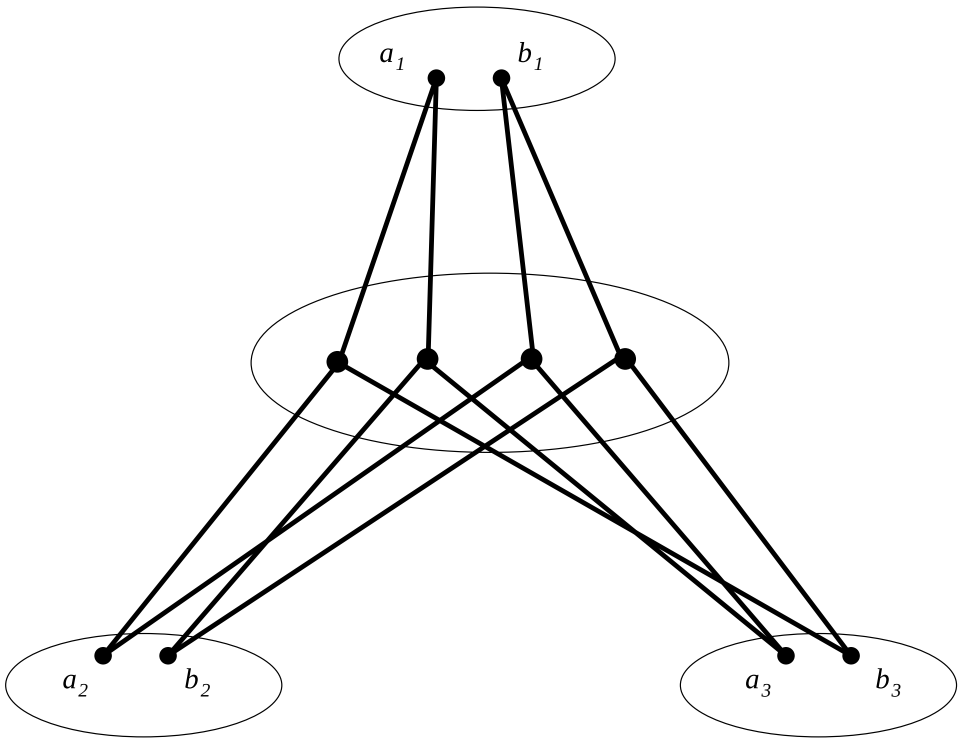

Figure 4: The Gadget X_3

can be seen by noticing that each of the middle vertices represents a subset of $\{1, 2, 3\}$ of even cardinality—edges to the a_i's represent the elements in the subset, and the edges to the b_i's represent the elements not in the subset.)

We next define the family $\{A_n\}$ of graphs: A_n is a 3-regular graph on $O(n)$ vertices with a separator of size $n+1$. (Separator size $n+1$ means that if one removes fewer than $n+1$ vertices from the graph, it remains connected.) The existence of such graphs can easily be proved probabilistically. The graph G_n is built as follows. First, replace each vertex u of A_n by a copy $X(u)$ of X_3. For each neighbor v of u, designate one of the (a, b) pairs of $X(u)$ as the "link" to v. Call this pair $(a(u, v), b(u, v))$. Add an edge connecting $a(u, v)$ to $a(v, u)$, and an edge connecting $b(u, v)$ to $b(v, u)$. H_n is the same as G_n, except that one of the (a, b) interconnections is reversed—$a(u, v)$ is connected to $b(v, u)$ and $b(u, v)$ to $a(v, u)$. See Figure 5. It is shown in the paper by Cai, Fürer, and Immerman that $G_n \not\cong H_n$, yet $G_n \equiv^{\mathcal{C}^n} H_n$. The intuitive idea is that since the separator is large, the Spoiler cannot "expose the twist" with just n pebbles. Counting doesn't help at all since the graphs have color classes of size only 4, so we will never have to count ≥ 4.

The property S that G_n and H_n differ on is "there are an even number of (a, b) flips." This can easily be seen to be checkable in AC^0 with parity gates, or expressible in $(\mathrm{FO} + \oplus_2)_{\leq,+,\times}$. $\square$

In conclusion:

Hope:

(1) There are languages for order-independent P, NL, L, ...

(2) It would be very useful to have more tools and games for the logics, especially logics with weaker versions of ordering. There is scope for lot of research in this direction.

notes taken by D. Sivakumar, sivak-d@cs.buffalo.edu

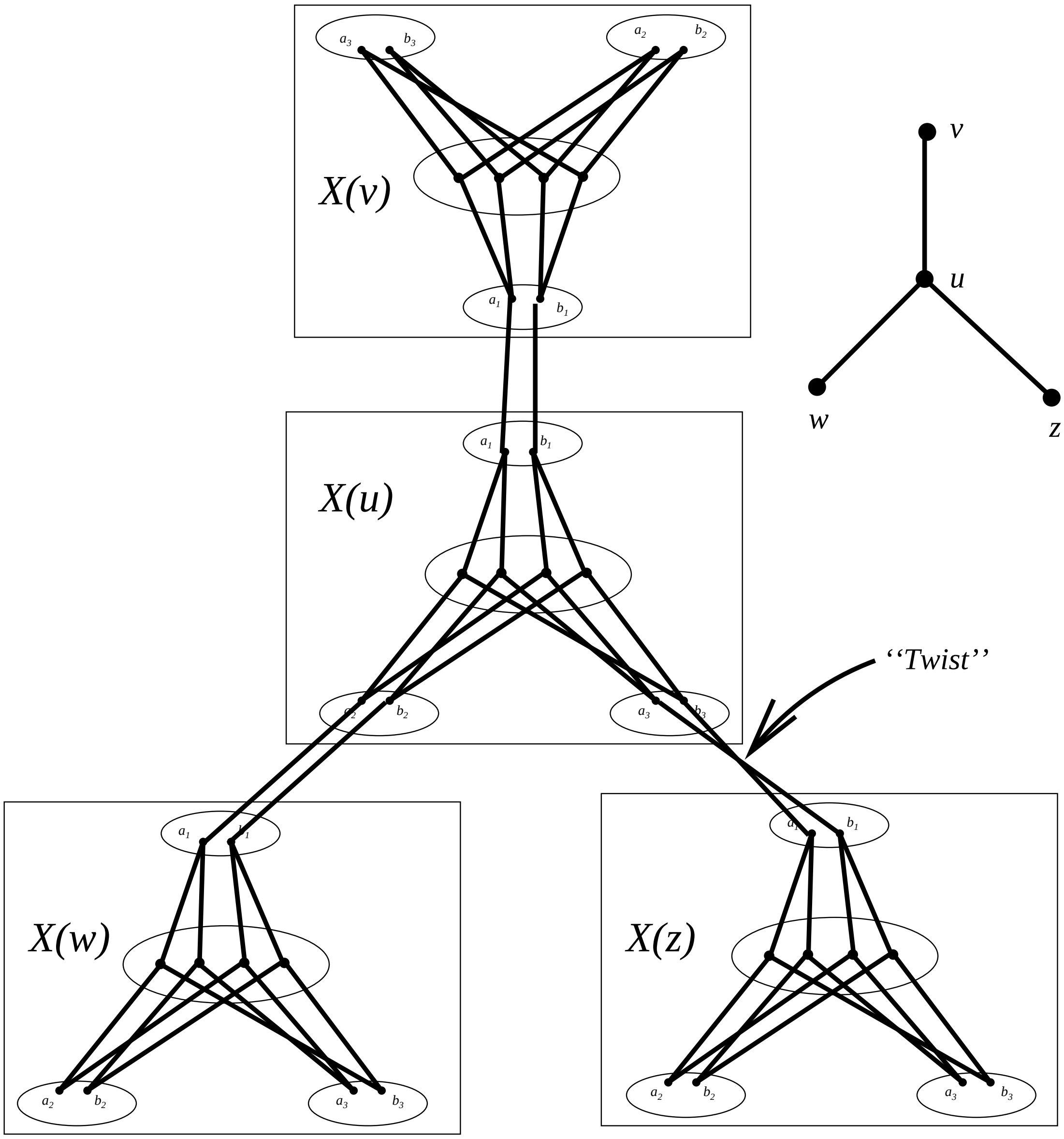

Figure 5: Defining the graphs G_n using X_3