# Introduction to Descriptive Complexity

**George Voutsadakis**[1]

[1]Mathematics and Computer Science
Lake Superior State University

LSSU Math 600

1. First-Order Reductions
   - FO $\subseteq$ L
   - Dual of a First-Order Query
   - Complete Problems for L and NL
   - Complete Problems for P

Subsection 1

## FO $\subseteq$ L

# FO ⊆ L

- Recall that FO is the set of first-order definable boolean queries.

**Theorem**

The set of first-order boolean queries is contained in the set of boolean queries computable in deterministic logspace: FO ⊆ L.

- Let $\sigma = \langle R_1^{a_1}, \ldots, R_r^{a_r}, c_1, \ldots, c_s \rangle$ be a vocabulary.
  Consider a first-order boolean query $I_\varphi : \text{STRUC}[\sigma] \to \{0, 1\}$, determined by
  $$\varphi \equiv (\exists x_1)(\forall x_2)\cdots(Q_k x_k)\alpha(\overline{x}) \in \mathcal{L}(\sigma),$$
  where $\alpha$ is quantifier-free.
  We must construct a logspace Turing machine $M$, such that, for all $\mathcal{A} \in \text{STRUC}[\sigma]$, $\mathcal{A}$ satisfies $\varphi$ iff $M$ accepts the binary encoding of $\mathcal{A}$. In symbols,
  $$\mathcal{A} \vDash \varphi \;\Leftrightarrow\; M(\text{bin}(\mathcal{A})) \downarrow.$$

# FO $\subseteq$ L (Cont'd)

- We construct the logspace Turing machine $M$ inductively on $k$, the number of quantifiers occurring in $\varphi$.

  If $k = 0$, then $\varphi = \alpha$ is a quantifier-free sentence.

  Thus, $\alpha$ is a fixed, finite boolean combination of atomic formulas.

  The atomic formulas have one of the following types:

  - Input relations $R_i(p_1, \ldots, p_{a_i})$;
  - Numeric relations $p_1 = p_2$, $p_1 \leq p_2$ or $\text{BIT}(p_1, p_2)$.

  The $p_i$'s are members of $\{c_1, \ldots, c_s, 0, 1, \max\}$.

  Suppose we know that $M$ can determine, on input $\mathcal{A}$, whether $\mathcal{A}$ satisfies each of these atomic formulas.

  $M$ can then determine whether $\mathcal{A} \models \alpha$, by performing the fixed, finite boolean combination using its finite control.

# FO ⊆ L (Cont'd)

- So we must convince ourselves that a logspace machine that knows its input is of the form $\text{bin}(\mathcal{A})$, for some $\mathcal{A} \in \text{STRUC}[\sigma]$, can calculate the values $n$ and $\lceil \log n \rceil$.

  Then, by counting, the machine can go to the appropriate constants and copy the $p_i$'s that it needs onto its worktape.

  To calculate one of the input predicates, $M$ can just look up the appropriate bit of its input.

# FO ⊆ L (Cont'd)

- Suppose, e.g., $M$ wants to calculate $R_3(c_2, \max, c_1)$.

  $M$ first copies the values $c_2$, $n-1$, $c_1$ to its worktape.

  Next it moves its read head to location $n^{a_1} + n^{a_2} + 1$, which is the beginning of the array encoding $R_3$.

  Finally, it moves its read head $n^2 \cdot c_2 + n(n-1) + c_1$ spaces to the right.

  The bit now being read is "1" iff $\mathcal{A} \vDash R_3(c_2, \max, c_1)$.

  It is easy to see that a logspace Turing machine may test the numeric predicates.

  This completes the construction of $M$ in the base case.

# FO ⊆ L (Cont'd)

- Inductively, assume that all first-order queries with $k-1$ quantifiers are logspace computable.

  Suppose

  $$\psi(x_1) = (\forall x_2)\cdots(Q_k x_k)\alpha(\overline{x}).$$

  Let $M_0$ be the logspace Turing machine that computes $\psi(c)$.

  $c$ is a new constant symbol substituted for the free variable $x_1$.

  To compute the query $\varphi \equiv (\exists x_1)(\psi(x_1))$ we build the logspace machine that:

  - Cycles through all possible values of $x_1$;
  - Substitutes each of these for $c$;
  - Runs $M_0$.
  - If any of these lead $M_0$ to accept, then we accept, else we reject.

  Note that the extra space needed is just $\log n$ bits to store the possible values of $x_1$.

  Simulating a universal quantifier is similar.

Subsection 2

## Dual of a First-Order Query

# Introduction

- A first-order query $I$ from STRUC$[\sigma]$ to STRUC$[\tau]$ maps any $\mathcal{A} \in$ STRUC$[\sigma]$ to $I(\mathcal{A}) \in$ STRUC$[\tau]$.
- It does this by defining the relations of $I(\mathcal{A})$ via first-order formulas.
- In a similar way, $I$ has a natural dual $\widehat{I}$, which translates any formula in $\mathcal{L}(\tau)$ to a formula in $\mathcal{L}(\sigma)$.
- The dual is useful in showing that relevant languages and complexity classes are closed under first-order reductions.
- Let $I$ be a $k$-ary first-order query.
- Consider a formula $\varphi \in \mathcal{L}(\tau)$.
- The formula $\widehat{I}(\varphi) \in \mathcal{L}(\sigma)$ is constructed as follows:
  - Replace each variable by a $k$-tuple of variables;
  - Replace each symbol of $\tau$ by its definition in $I$.
- It follows that the length of $\widehat{I}(\varphi)$ is linear in the length of $\varphi$.

# Definition of the Dual of $I$

- Consider a $k$-ary first-order query from STRUC$[\sigma]$ to STRUC$[\tau]$

$$I = \lambda_{x_1 \ldots x_d} \langle \varphi_0, \varphi_1, \ldots, \varphi_r, \psi_1, \ldots, \psi_s \rangle.$$

- Then $I$ defines a **dual**

$$\widehat{I} : \mathcal{L}(\tau) \to \mathcal{L}(\sigma).$$

- Suppose

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

- For $\varphi \in \mathcal{L}(\tau)$, $\widehat{I}(\varphi)$ is the result of replacing all relation and constant symbols in $\varphi$ by the corresponding formulas in $I$.
- To accomplish the replacement, we use a map $f_I$ defined as follows:
    - Each variable $v$ is mapped to a $k$-tuple of variables,

$$f_I(v) = v^1, \ldots, v^k;$$

# Definition of the Dual of $I$ (Cont'd)

- We continue the definition of the map $f_I$:
  - Input relations are replaced by their corresponding formulas,

$$f_I(R_i(v_1,\ldots,v_{a_i})) = \varphi_i(f_I(v_1),\ldots,f_I(v_{a_i}));$$

  - Constant $c_i$ is replaced by a special $k$-tuple of variables,

$$f_I(c_i) = z_i^1,\ldots,z_i^k;$$

  - Quantifiers are replaced by restricted quantifiers,

$$f_I(\exists v) = (\exists f_I(v).\varphi_0(f_I(v)));$$

  - The equality relation and the other numeric relations are replaced by their appropriate formulas;
  - Second-order quantifiers have the arities of the relations being quantified, multiplied by $k$,

$$f_I(\exists R^a) = (\exists R^{ka});$$

  - On boolean connectives, $f_I$ is the identity.

# Definition of the Dual of $I$ (Cont'd)

- The only thing to add is that the variables $z_i^1, \dots, z_i^k$, corresponding to the constant symbol $c_i$, must be quantified before they are used.
- It does not matter where these quantifiers go because the values are uniquely defined.
- Typically, we can place these quantifiers at the beginning of a first-order formula.
- For a second-order formula, they would be placed just after the second-order quantifiers.
- Thus, the mapping $\widehat{I}$ is defined as follows, for $\theta \in \mathcal{L}(\tau)$,

$$\widehat{I}(\theta) = (\exists z_1^1 \dots z_1^k . \psi_1(z_1^1 \dots z_1^k)) \dots (\exists z_s^1 \dots z_s^k . \psi_s(z_s^1 \dots z_s^k))(f_I(\theta)).$$

# Example

- Consider the query $I_{PM} : \text{STRUC}[\tau_s] \to \text{STRUC}[\tau_{abcd}]$, given by

$$\begin{array}{rcl} \varphi_A(x,y) & \equiv & y = \max \wedge S(x); \\ \varphi_B(x,y) & \equiv & y = \max; \\ \psi_c(x,y) & \equiv & \langle 0, \max \rangle; \\ \psi_d(x,y) & \equiv & \langle 0, 1 \rangle; \\ I_{PM} & \equiv & \lambda_{xy} \langle \mathbf{true}, \varphi_A, \varphi_B, \psi_c, \psi_d \rangle. \end{array}$$

  One sample value of the map $\widehat{I_{PM}}$ is

$$\begin{array}{rcl} \widehat{I_{PM}}(A(c)) & \equiv & (\exists z_1 z_2 . z_1 = 0 \wedge z_2 = \max)(z_2 = \max \wedge S(z_1)) \\ & \equiv & S(0). \end{array}$$

  We may similarly compute the value of $\widehat{I_{PM}}$ on the following:
  1. $(\forall v)(A(v) \leftrightarrow B(v))$;
  2. $A(\max)$;
  3. $A(0)$.

# The Satisfaction Relation Between $I$ and $\widehat{I}$

## Proposition

Let $\sigma$, $\tau$, and $I$ be as in the previous definitions. Then, for all sentences $\theta \in \mathcal{L}(\tau)$ and all structures $\mathcal{A} \in \text{STRUC}[\sigma]$,

$$\mathcal{A} \vDash \widehat{I}(\theta) \quad \text{iff} \quad I(\mathcal{A}) \vDash \theta.$$

- The result goes through for formulas with free variables as well.
- Then $I$ must behave appropriately on interpretations of variables.
- That is, $I(\mathcal{A}, i) = (I(\mathcal{A}), i')$, where:
  - $i'(x)$ is defined iff all of $i(x^1), \ldots, i(x^k)$ are defined;
  - In this case, $i'(x) = \langle i(x^1), \ldots, i(x^k) \rangle$.

# Everything is a Graph

- Let $\sigma$ be any vocabulary.
- Let $\tau_e = \langle E^2 \rangle$ be the vocabulary with one binary relation symbol.
- $\tau_e$ is the vocabulary of graphs with no specified points.
- We can show that every structure may be thought of as a graph.
- More precisely, we may construct first-order queries

$$I_\sigma : \text{STRUC}[\sigma] \to \text{STRUC}[\tau_e];$$
$$I_\sigma^{-1} : \text{STRUC}[\tau_e] \to \text{STRUC}[\sigma],$$

  such that, for all $\mathcal{A} \in \text{STRUC}[\sigma]$,

$$I_\sigma^{-1}(I_\sigma(\mathcal{A})) \cong \mathcal{A}.$$

- To build the graph $I_\sigma(\mathcal{A})$, one can construct "gadgets", i.e., small recognizable graphs, to label different sorts of vertices.
- E.g., we may have gadgets corresponding to:
  - Elements of $|\mathcal{A}|$;
  - Tuples from each relation $R_i^{\mathcal{A}}$, etc.

# Closure Under First-Order Reductions

- Suppose $A$ and $B$ are boolean queries.
- If $A$ is first-order reducible to $B$ ($A \leq_{\text{fo}} B$), then intuitively the complexity of $A$ is not greater than the complexity of $B$.

**Definition (Closure Under First-Order Reductions)**

A set of boolean queries $\mathcal{S}$ is **closed under first-order reductions** if, for all boolean queries $A$ and $B$,

$$B \in \mathcal{S} \quad \text{and} \quad A \leq_{\text{fo}} B \quad \text{imply} \quad A \in \mathcal{S}.$$

We say that a language $\mathcal{L}$ is **closed under first-order reductions** if the set of boolean queries definable in $\mathcal{L}$ is closed under first-order reductions.

# LogSpace and First-Order Reductions

- We have seen that the set of first-order boolean queries is contained in the set of boolean queries computable in deterministic logspace,

$$\mathrm{FO} \subseteq \mathrm{L}.$$

- This immediately yields

**Proposition**

Let $\mathcal{S}$ be any set of boolean queries that is closed under logspace reductions. Then $\mathcal{S}$ is also closed under first-order reductions.

- Suppose $A \in \mathcal{S}$ and $B \leq_{\mathrm{fo}} A$.

  By hypothesis, $A \in \mathcal{S}$.

  Since $\mathrm{FO} \subseteq \mathrm{L}$ and, by hypothesis, $B \leq_{\mathrm{fo}} A$, we get $B \leq_{\mathrm{L}} A$.

  Since $\mathcal{S}$ is closed under $\leq_{\mathrm{L}}$, $B \in \mathcal{S}$.

  Hence, $\mathcal{S}$ is also closed under $\leq_{\mathrm{fo}}$.

# Complexity Classes, Languages and Reductions

## Meta-Proposition

- All the complexity classes $\mathcal{C}$ that we discuss in these notes are closed under first-order reductions.
- All the languages $\mathcal{L}$ that we discuss in these notes are closed under first-order reductions.

- There is a general method for proving this proposition whenever a new complexity class or logical language is encountered.

  For complexity classes we can usually use the preceding proposition.

  This is because most complexity classes are closed under logspace reductions.

# Complexity Classes, Languages and Reductions (Cont'd)

- Now we look at the case of languages.
  Let $A$, $B$ be two boolean queries.
  Suppose $B$ is expressible as the formula $\varphi_B$ in language $\mathcal{L}$.
  Suppose, also, that $A \leq_{\text{fo}} B$.
  Let $I_{AB}$ be the first-order reduction from $A$ to $B$.
  We know that for all structures $\mathcal{S}$,
  $$\mathcal{S} \in A \quad \text{iff} \quad I_{AB}(\mathcal{S}) \in B.$$
  It follows from the preceding proposition that
  $$\mathcal{S} \in A \quad \text{iff} \quad \mathcal{S} \vDash \widehat{I_{AB}}(\varphi_B).$$
  So, if $\widehat{I_{AB}}(\varphi_B)$ is in $\mathcal{L}$, then the proof is complete.
  By definition, $\widehat{I}(\varphi)$ is a simple substitution that does not change the structure of $\varphi$ very much.
  So, for the languages we consider, $\widehat{I_{AB}}(\varphi_B)$ will be in $\mathcal{L}$.

# Completeness and Expressibility

- Suppose that we know that a boolean query $A$ is complete via first-order reductions for a complexity class $\mathcal{C}$.
- Suppose, further, that $A$ is expressible in a language $\mathcal{L}$ which is closed under first-order reductions.
- It follows that $\mathcal{L}$ expresses everything in $\mathcal{C}$.
  - Let $B \in \mathcal{C}$.
  - By hypothesis, $B \leq_{\text{fo}} A$.
  - Also by hypothesis, $A$ is expressible by $\varphi_A \in \mathcal{L}$.
  - As $\mathcal{L}$ is closed under $\leq_{\text{fo}}$, $B$ is also expressible in $\mathcal{L}$.

# Expressibility and Complexity

- Suppose that $\mathcal{L}$ is a set of boolean queries describable in some language.
- Suppose that $\mathcal{C}$ is a complexity class, that is, a set of boolean queries computable in some complexity bound.
- Showing that $\mathcal{L} = \mathcal{C}$ usually involves four steps.
  1. Show that $\mathcal{L} \subseteq \mathcal{C}$ by producing, for each formula $\varphi \in \mathcal{L}$, an algorithm in $\mathcal{C}$ that computes the boolean query
  $$\text{MOD}[\varphi] = \{\mathcal{A} : \mathcal{A} \vDash \varphi\}.$$
  2. Produce a boolean query $T$ that is complete for $\mathcal{C}$ via first-order reductions.
  3. Show that $\mathcal{L}$ is closed under first-order reductions.
  4. Show that $T \in \mathcal{L}$ by expressing $T$ in the language.

# Example

- We will show later that NP = SO∃.
- To accomplish this, we can show:
  (1) Each SO∃ formula can be checked by an NP machine;
  (2) The problem SAT is complete for NP via $\leq_{fo}$;
  (3) SO∃ is closed under first-order reductions;
  (4) SAT is expressible in SO∃.

Subsection 3

## Complete Problems for L and NL

# Reachability

## Definition

Define REACH to be the set of directed graphs $G$, such that there is a path in $G$ from $s$ to $t$.

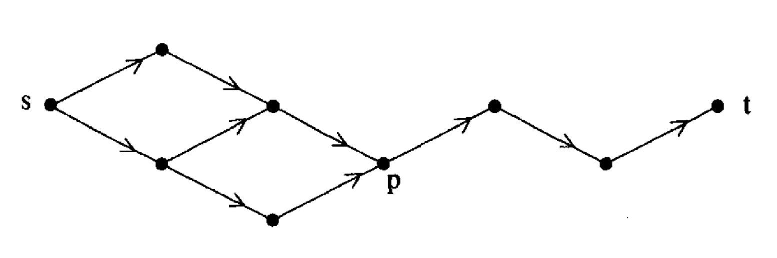
Define $\text{REACH}_d$ to be the subset of REACH, such that the path from $s$ to $t$ is *deterministic*. This means that for each edge $(u, x)$ on the path, this is the unique edge in $G$ leaving $u$.

Define $\text{REACH}_u$, the undirected graph reachability problem, to be the restriction of REACH to undirected graphs,

$$\text{REACH}_u = \{G \in \text{REACH} : G \vDash (\forall xy)(E(x,y) \rightarrow E(y,x))\}.$$

# Example

- Consider the directed graph in the figure.




- It is in REACH.
- However, it is not in $\text{REACH}_d$.
- Note that there is a directed path from $p$ to $t$.

# Algorithm for REACH

- The following NSPACE[$\log n$] algorithm recognizes REACH.

**Algorithm (Recognizing REACH in NL)**

$b := s$<br>
while $(b \neq t)$ do $\{$<br>
    $a := b$<br>
    nondeterministically choose new $b$<br>
    if $(\neg E(a, b))$ then reject$\}$<br>
accept

- Note that the space used is just the $O(\log n)$ bits needed to name the two vertices $a$ and $b$.

# Completeness of REACH for NL

**Theorem**

REACH is complete for NL via first-order reductions.

- Let $S \subseteq \text{STRUC}[\sigma]$ be a boolean query in NL.
  Let $N$ be the nondeterministic logspace Turing machine accepting $S$.
  We construct a first-order reduction $I : \text{STRUC}[\sigma] \to \text{STRUC}[\tau_g]$, such that, for all $\mathcal{A} \in \text{STRUC}[\sigma]$,
  $$N(\text{bin}(\mathcal{A}))\downarrow \quad \text{iff} \quad I(\mathcal{A}) \in \text{REACH}.$$
  Let $c$ be such that $N$ uses, on inputs $\text{bin}(\mathcal{A})$, with $n = \|\mathcal{A}\|$, at most $c \log n$ bits of worktape.
  Let $\sigma = \langle R_1^{a_1}, \ldots, R_r^{a_r}, c_1, \ldots, c_s \rangle$.
  Let $a = \max\{a_i : 1 \leq i \leq r\}$.
  Let $k = 1 + a + c$.
  Consider a run of $N$ on input $\text{bin}(\mathcal{A})$.

# Completeness of REACH for NL (IDs)

- We code an instantaneous description (ID) of $N$'s computation as a $k$-tuple of variables,

$$\text{ID} = (p, r_1, \ldots, r_a, w_1, \ldots, w_c).$$

  Variables $r_1, \ldots, r_a$ encode where in one of the input relations the read head of $N$ is looking.

  Suppose, for example it is looking at relation $R_i$.

  Then $N$'s read head is looking at a "1" iff $\mathcal{A} \vDash R_i(r_1, \ldots, r_{a_i})$.

  Variables $w_1, \ldots, w_c$ encode the contents of $N$'s work tape.

  Each variable represents an element of $\mathcal{A}$'s $n$-element universe.

  So it corresponds to a $\log n$-bit number.

  We are assuming the presence of the numerical relations $\leq$ and BIT.

  Of these, $\leq$ is necessary, but BIT is merely convenient.

# Completeness of REACH for NL (IDs Cont'd)

- Finally, we need $O(\log\log n)$ bits of further information to encode:
  (1) The state of $N$;
  (2) Which input relation or constant symbol the read head is currently scanning;
  (3) The position of the work head.

  We assume that $n$ is sufficiently large that all of this information can be encoded into a single variable, $p$.

  Next, we build the desired $k$-ary first-order query $I$.

  Moreover, we show that it satisfies

$$(\text{bin}(\mathcal{A}))\downarrow \quad \text{iff} \quad I(\mathcal{A}) \in \text{REACH}.$$

# Completeness of REACH for NL (Query)

- $I$ is constructed as

$$I = \lambda_{\mathrm{ID},\mathrm{ID}'}\langle \mathbf{true}, \varphi_N, \alpha, \omega \rangle,$$

where:

1. The universe relation being "**true**" indicates that for any $\mathcal{A} \in \mathrm{STRUC}[\sigma]$, the universe of $I(\mathcal{A})$ consists of *all* $k$-tuples from the universe of $\mathcal{A}$, $|I(\mathcal{A})| = |\mathcal{A}|^k$;
2. $\mathcal{A} \vDash \varphi_N(\mathrm{ID}, \mathrm{ID}')$ iff $(\mathrm{ID}, \mathrm{ID}')$ is a valid move of $N$ on input bin(A);
3. $\mathcal{A} \vDash \alpha(\mathrm{ID}_i)$ iff $\mathrm{ID}_i$ is the unique initial ID of $N$, for inputs of size $\|\mathcal{A}\|$;
4. $\mathcal{A} \vDash \omega(\mathrm{ID}_f)$ iff $\mathrm{ID}_f$ is the unique accept ID of $N$ for inputs of size $\|\mathcal{A}\|$.

Formulas $\alpha$ and $\omega$ are the following,

$$\begin{array}{rcl} \alpha(x_1, \ldots, x_k) & \equiv & x_1 = x_2 = \cdots = x_k = 0; \\ \omega(x_1, \ldots, x_k) & \equiv & x_1 = x_2 = \cdots = x_k = \max . \end{array}$$

# Completeness of REACH for NL (Query Cont'd)

- Formula $\varphi_N$ is not hard, but it is more tedious.

  It is essentially a disjunction over $N$'s finite transition table.

  A typical entry in the transition table is

  $$(\langle q, b, w\rangle, \langle q', i_d, w', w_d\rangle).$$

  This says that:
  - In state $q$,
  - Looking at bit $b$ with the input head;
  - Looking at bit $w$ with the work head;

  $N$ may:
  - Go to state $q'$;
  - Move its input head one step in direction $i_d$;
  - Write bit $w'$ on its work tape;
  - Move its work head one step in direction $w_d$.

# Completeness of REACH for NL (Query Cont'd)

- The corresponding disjunct in $\varphi_N$ must decode from variable $p$:
  - The old state;
  - Which input relation is being read, say $R$.

  Then the bit $b$ is "1" iff $R_i(r_1, \ldots, r_{a_i})$ holds.

  Similarly, we must extract from $p$:
  - The segment $j$ of the work tape that is currently being scanned;
  - The position $s$ on that worktape.

  Thus, bit $w$ is "1" iff $\mathrm{BIT}(w_j, s)$ holds.

  By construction, for any $\mathcal{A} \in \mathrm{STRUC}[\sigma]$, $I(\mathcal{A})$ is the computation graph of $N$ on input $\mathrm{bin}(\mathcal{A})$.

  So $N$ accepts $\mathrm{bin}(\mathcal{A})$ iff there is a path in $I(\mathcal{A})$ from $s$ to $t$.

# Gaps in the Proof Sketch

- There are several gaps left in the preceding proof.
  1. Using numeric relation BIT, we may write first-order formulas to uniquely identify elements $\ell_1 = \lceil \log n \rceil$ and $\ell_2 = \lceil \log \log n \rceil$ of the universe.
  2. Since the coding is somewhat arbitrary, it is possible to use the given equations as our definitions of $\alpha$ and $\omega$.
  3. Assuming that the first $\ell_2$ bits of $p$ encode the work head's position $s$, we may write a formula to uniquely identify element $s$.
  4. Assuming that the bits of $s$ are encoded in the last $\ell_2$ bits of $p$, we may write a formula to uniquely identify element $s$.

     To do this we need addition, which is available by a previous theorem.

# Completeness of $\text{REACH}_d$ for L

- We show $\text{REACH}_d$ is complete for L via first-order reductions.
- We first show that $\text{REACH}_d$ is in L.
- We modify the algorithm for REACH.
- A deterministic path has at most one edge leaving each vertex.
- So nondeterminism is no longer needed.
- We add a counter to detect cycles.

**Algorithm (Recognizing $\text{REACH}_d$ in L)**

$b := s; i := 0; n := \|G\|$

while $b \neq t \wedge i < n \wedge (\exists! a)(E(b,a))$ do $\{$

  $b :=$ the unique $a$ for which $E(b,a)$

  $i := i + 1\}$

if $b = t$ then accept else reject

# Completeness of $\mathrm{REACH}_d$ for L (Cont'd)

- The definition of $\mathrm{REACH}_d$ was made just so that the following theorem would be true.

**Theorem**

$\mathrm{REACH}_d$ is complete for L via first-order reductions.

- This proof is similar to the corresponding one for REACH.

  We copy the whole construction with $S \subseteq \mathrm{STRUC}[\sigma]$ an arbitrary boolean query from L.

  The only difference is that now $N$ is a deterministic logspace Turing machine that computes $S$.

  Since $N$ is deterministic, for any $\mathcal{A} \in \mathrm{STRUC}[\sigma]$, the graph $I(\mathcal{A})$ has at most one edge leaving any vertex.

  It follows that $I(\mathcal{A})$ is in REACH iff it is in $\mathrm{REACH}_d$.

  Thus, $N(\mathrm{bin}(\mathcal{A}))\downarrow$ iff $I(\mathcal{A}) \in \mathrm{REACH}_d$.

Subsection 4

# Complete Problems for P

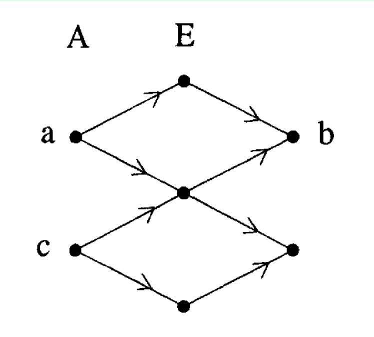
# Alternating Graphs

- Let an **alternating graph** $G = (V, E, A, s, t)$ be a directed graph whose vertices are labeled universal or existential.
- $A \subseteq V$ is the set of universal vertices.
- Let $\tau_{ag} = \langle E^2, A^1, s, t\rangle$ be the vocabulary of alternating graphs.
- Alternating graphs have a different notion of **accessibility**.
- Let $P_a^G(x, y)$ be the smallest relation on vertices of $G$ such that:
  1. $P_a^G(x, x)$;
  2. If $x$ is existential and $P_a^G(z, y)$ holds for some edge $(x, z)$, then $P_a^G(x, y)$.
  3. If $x$ is universal, there is at least one edge leaving $x$, and $P_a^G(z, y)$ holds for all edges $(x, z)$, then $P_a^G(x, y)$.
- Let

$$\text{REACH}_a = \left\{G : P_a^G(s, t)\right\}.$$

# Example

- The figure shows an alternating graph.




- In the graph $P_a^G(a, b)$ holds.
- On the other hand, $P_a^G(c, b)$ does not hold.

# Recognizing $\text{REACH}_a$ in Linear Time on a RAM

- The following marking algorithm computes $\text{REACH}_a$ in linear time.

**Algorithm (Recognizing $\text{REACH}_a$ in Linear Time on a RAM)**

make QUEUE empty; mark($t$); insert $t$ into QUEUE<br>
while QUEUE not empty do $\{$<br>
    remove first element, $x$, from QUEUE<br>
    for each unmarked vertex $y$ such that $E(y, x)$ do $\{$<br>
        delete edge $(y, x)$<br>
        if $y$ is existential or $y$ has no outgoing edges<br>
         then mark($y$); insert $y$ into QUEUE$\}$ $\}$<br>
if $s$ is marked then accept else reject

# Completeness of $\text{REACH}_a$ for P

**Theorem**

$\text{REACH}_a$ is complete for P via first-order reductions.

- Let $S \subseteq \text{STRUC}[\sigma]$ be an arbitrary boolean query.
  Assume that $S \in \text{P}$.
  Let $T$ be the alternating, logspace Turing machine that computes $S$.
  We construct a first-order reduction

$$I_a : \text{STRUC}[\sigma] \to \text{STRUC}[\tau_{ag}]$$

such that, for all $\mathcal{A} \in \text{STRUC}[\sigma]$,

$$T(\text{bin}(\mathcal{A}))\downarrow \quad \text{iff} \quad I_a(\mathcal{A}) \in \text{REACH}_a.$$

The only difference between $I$, the query from the proof of REACH, and $I_a$ is that $I_a$ must also describe the relation $A$ that identifies the universal states of $T$.

# Completeness of $\text{REACH}_a$ for P (Cont'd)

- Assume for simplicity that the universal states are exactly the odd-numbered states.

  Assume, further, that the variable $p$ in an ID encodes its state in its low-order bits.

  Thus, the state of an ID is universal iff the corresponding $p$ is odd.

  This occurs iff $\text{BIT}(p, 0)$ holds.

  Thus, we let

$$I = \lambda_{\text{ID},\text{ID}'} \langle \textbf{true}, \varphi_T, \psi_A, \alpha, \omega \rangle,$$

  where:

  - $\psi_A = \text{BIT}(p, 0)$;
  - $\varphi_T$, $\alpha$, $\omega$ are defined exactly as in the previous proof.

  We have $T(\text{bin}(\mathcal{A}))\downarrow$ iff $I_a(\mathcal{A}) \in \text{REACH}_a$.