Short Survey of Descriptive Complexity Plus . . .

Neil Immerman

February 26, 2010
Restrict attention to the complexity of computing individual bits of the output, i.e., decision problems.

How hard is it to check if input has property $S$?

How rich a language do we need to express property $S$?

There is a constructive isomorphism between these two approaches.
Restrict attention to the complexity of computing individual bits of the output, i.e., decision problems

How hard is it to check if input has property $S$?

How rich a language do we need to express property $S$?

There is a constructive isomorphism between these two approaches.
Restrict attention to the complexity of computing individual bits of the output, i.e., decision problems.

How hard is it to check if input has property $S$?

How rich a language do we need to express property $S$?

There is a constructive isomorphism between these two approaches.
Restrict attention to the complexity of computing individual bits of the output, i.e., **decision problems**

How hard is it to **check** if input has property $S$?

How rich a language do we need to **express** property $S$?

There is a **constructive isomorphism** between these two approaches.
Descriptive Complexity

Restrict attention to the complexity of computing individual bits of the output, i.e., decision problems

How hard is it to check if input has property $S$?

How rich a language do we need to express property $S$?

There is a constructive isomorphism between these two approaches.
Graph

\[ G = (\{v_1, \ldots, v_n\}, E, s, t) \]

Binary

\[ A_w = (\{p_1, \ldots, p_8\}, S) \]

String

\[ S = \{p_2, p_5, p_7, p_8\} \]

\[ w = 01001011 \]

Vocabularies:

\[ \tau_g = (E^2, s, t), \quad \tau_s = (S^1) \]
input symbols: from $\tau$
variables: $x, y, z, \ldots$
boolean connectives: $\land, \lor, \neg$
quantifiers: $\forall, \exists$
umeric symbols: $=, \leq, +, \times, \min, \max$

$\alpha \equiv \forall x \exists y (E(x, y)) \in \mathcal{L}(\tau_g)$

$\beta \equiv \exists x \forall y (x \leq y \land S(x)) \in \mathcal{L}(\tau_s)$

$\beta \equiv S(\min) \in \mathcal{L}(\tau_s)$
\[ \Phi_{3\text{-color}} \equiv \exists R^1 Y^1 B^1 \forall x y ((R(x) \lor Y(x) \lor B(x)) \land \\
(E(x, y) \rightarrow (\neg(R(x) \land R(y)) \land \neg(Y(x) \land Y(y)) \land \\
\neg(B(x) \land B(y))))\)]
Fagin’s Theorem: \[ \text{NP} = \text{SO} \exists \]

\[ \Phi_{3\text{-color}} \equiv \exists R^1 Y^1 B^1 \forall x y ((R(x) \lor Y(x) \lor B(x)) \land (E(x, y) \rightarrow (\neg (R(x) \land R(y)) \land \neg (Y(x) \land Y(y))) \land \neg (B(x) \land B(y)))) \]
Addition is First-Order

\[ Q_+ : \text{STRUC}[\tau_{AB}] \rightarrow \text{STRUC}[\tau_s] \]

\[
\begin{array}{cccccc}
A & a_1 & a_2 & \ldots & a_{n-1} & a_n \\
B & + & b_1 & b_2 & \ldots & b_{n-1} & b_n \\
S & & s_1 & s_2 & \ldots & s_{n-1} & s_n \\
\end{array}
\]
Addition is First-Order

$$Q_+ : \text{STRUCT}[\tau_{AB}] \rightarrow \text{STRUCT}[\tau_s]$$

\[
\begin{array}{ccccccc}
A & & a_1 & a_2 & \ldots & a_{n-1} & a_n \\
B & + & b_1 & b_2 & \ldots & b_{n-1} & b_n \\
S & & s_1 & s_2 & \ldots & s_{n-1} & s_n
\end{array}
\]

$$C(i) \equiv (\exists j > i)(A(j) \wedge B(j) \wedge$$
\[
(\forall k. j \geq k > i)(A(k) \vee B(k))
\]
Addition is First-Order

\[ Q_+ : \text{STRUC}[\tau_{AB}] \rightarrow \text{STRUC}[\tau_s] \]

\[
\begin{array}{cccccc}
A & a_1 & a_2 & \ldots & a_{n-1} & a_n \\
B & + & b_1 & b_2 & \ldots & b_{n-1} & b_n \\
S & & s_1 & s_2 & \ldots & s_{n-1} & s_n \\
\end{array}
\]

\[ C(i) \equiv (\exists j > i) \left( (A(j) \land B(j)) \land (\forall k. j > k > i)(A(k) \lor B(k)) \right) \]

\[ Q_+(i) \equiv A(i) \oplus B(i) \oplus C(i) \]
Parallel Machines:

\[ \text{CRAM}[t(n)] = \text{CRCW-PRAM-TIME}[t(n)]-\text{HARD}[n^{O(1)}] \]
Parallel Machines: Quantifiers are Parallel

\[ \text{CRAM}[t(n)] = \text{CRCW-PRAM-TIME}[t(n)]\text{-HARD}[n^{O(1)}] \]

Assume array \( A[x] : x = 1, \ldots, r \) in memory.
Parallel Machines: Quantifiers are Parallel

\[ \text{CRAM}[t(n)] = \text{CRCW-PRAM-TIME}[t(n)]-\text{HARD}[n^{O(1)}] \]

Assume array \( A[x] : x = 1, \ldots, r \) in memory.

\[ \forall x(A(x)) \equiv \text{write}(1); \text{proc } p_i : \text{if } (A[i] = 0) \text{ then } \text{write}(0) \]
Inductive Definitions

\[ E^*(x, y) \equiv x = y \lor E(x, y) \lor \exists z (E^*(x, z) \land E^*(z, y)) \]
Inductive Definitions

\[ E^*(x, y) \equiv x = y \lor E(x, y) \lor \exists z (E^*(x, z) \land E^*(z, y)) \]

\[ \varphi_{tc}(R, x, y) \equiv x = y \lor E(x, y) \lor \exists z (R(x, z) \land R(z, y)) \]
Inductive Definitions

\[ E^*(x, y) \equiv x = y \lor E(x, y) \lor \exists z(E^*(x, z) \land E^*(z, y)) \]

\[ \varphi_{tc}(R, x, y) \equiv x = y \lor E(x, y) \lor \exists z(R(x, z) \land R(z, y)) \]

\[ G \in \text{REACH} \iff G \models (\text{LFP}\varphi_{tc})(s, t) \]
### Inductive Definitions

\[ E^*(x, y) \equiv x = y \lor E(x, y) \lor \exists z(E^*(x, z) \land E^*(z, y)) \]

\[ \varphi_{tc}(R, x, y) \equiv x = y \lor E(x, y) \lor \exists z(R(x, z) \land R(z, y)) \]

\[ G \in \text{REACH} \iff G \models (\text{LFP}\varphi_{tc})(s, t) \]

Thus, \text{REACH} \in \text{IND}[\log n].
Inductive Definitions

\[ E^*(x, y) \equiv x = y \lor E(x, y) \lor \exists z (E^*(x, z) \land E^*(z, y)) \]

\[ \varphi_{tc}(R, x, y) \equiv x = y \lor E(x, y) \lor \exists z (R(x, z) \land R(z, y)) \]

\[ G \in \text{REACH} \iff G \models (\text{LFP} \varphi_{tc})(s, t) \]

Thus, \( \text{REACH} \in \text{IND}[\log n] \).

Next, we’ll show that \( \text{REACH} \in \text{FO}[\log n] \).
1. Dummy universal quantification for base case:

\[ \varphi_{tc}(R, x, y) \equiv \exists z (R(x, z) \land R(z, y)) \]

\[ M_1 \equiv \neg (x = y \lor E(x, y)) \]
$\varphi_{tc}(R, x, y) \equiv x = y \lor E(x, y) \lor \exists z (R(x, z) \land R(z, y))$

1. Dummy universal quantification for base case:

$$\varphi_{tc}(R, x, y) \equiv (\forall z. M_1)(\exists z)(R(x, z) \land R(z, y))$$

$$M_1 \equiv \neg(x = y \lor E(x, y))$$

2. Using $\forall$, replace two occurrences of $R$ with one:

$$\varphi_{tc}(R, x, y) \equiv (\forall z. M_1)(\exists z)(\forall uv. M_2)R(u, v)$$

$$M_2 \equiv (u = x \land v = z) \lor (u = z \land v = y)$$
\[ \varphi_{tc}(R, x, y) \equiv x = y \lor E(x, y) \lor \exists z (R(x, z) \land R(z, y)) \]

1. Dummy universal quantification for base case:

\[ \varphi_{tc}(R, x, y) \equiv (\forall z. M_1)(\exists z)(R(x, z) \land R(z, y)) \]
\[ M_1 \equiv \neg(x = y \lor E(x, y)) \]

2. Using \( \forall \), replace two occurrences of \( R \) with one:

\[ \varphi_{tc}(R, x, y) \equiv (\forall z. M_1)(\exists z)(\forall uv. M_2)R(u, v) \]
\[ M_2 \equiv (u = x \land v = z) \lor (u = z \land v = y) \]

3. Requantify \( x \) and \( y \).

\[ M_3 \equiv (x = u \land y = v) \]

\[ \varphi_{tc}(R, x, y) \equiv \left[ (\forall z. M_1)(\exists z)(\forall uv. M_2)(\exists xy. M_3) \right] R(x, y) \]

Every FO inductive definition is equivalent to a quantifier block.
\[ QB_{tc} \equiv \left( (\forall z. M_1)(\exists z)(\forall uv. M_2)(\forall xy. M_3) \right) \]

\[ \varphi_{tc}(R, x, y) \equiv \left( (\forall z. M_1)(\exists z)(\forall uv. M_2)(\exists xy. M_3) \right) R(x, y) \]
\[ \mathcal{QB}_{tc} \equiv [(\forall z. M_1)(\exists z)(\forall u v. M_2)(\forall x y. M_3)] \]

\[ \varphi_{tc}(R, x, y) \equiv [(\forall z. M_1)(\exists z)(\forall u v. M_2)(\exists x y. M_3)] R(x, y) \]

\[ \varphi_{tc}(R, x, y) \equiv [\mathcal{QB}_{tc}] R(x, y) \]
\[ QB_{tc} \equiv [(\forall z. M_1)(\exists z)(\forall uv. M_2)(\exists xy. M_3)] \]

\[ \varphi_{tc}(R, x, y) \equiv [(\forall z. M_1)(\exists z)(\forall uv. M_2)(\exists xy. M_3)] R(x, y) \]

\[ \varphi_{tc}(R, x, y) \equiv [QB_{tc}] R(x, y) \]

\[ \varphi^r_{tc}(\emptyset) \equiv [QB_{tc}]^r (\text{false}) \]
\(QB_{tc} \equiv [(\forall z. M_1)(\exists z)(\forall uv. M_2)(\exists xy. M_3)]\)

\[\varphi_{tc}(R, x, y) \equiv [(\forall z. M_1)(\exists z)(\forall uv. M_2)(\exists xy. M_3)]R(x, y)\]

\[\varphi_{tc}(R, x, y) \equiv [QB_{tc}]R(x, y)\]

\[\varphi_{tc}^r(\emptyset) \equiv [QB_{tc}]^r(\text{false})\]

Thus, for any structure \(\mathcal{A} \in \text{STRUC}[\tau_g]\),

\[\mathcal{A} \in \text{REACH} \iff \mathcal{A} \models (\text{LFP}\varphi_{tc})(s, t)\]

\[\iff \mathcal{A} \models ([QB_{tc}]^{1+\log \|\mathcal{A}\|} \text{false})(s, t)\]
CRAM\[t(n)\] = concurrent parallel random access machine; polynomial hardware, parallel time \(O(t(n))\)

IND\[t(n)\] = first-order, depth \(t(n)\) inductive definitions

FO\[t(n)\] = \(t(n)\) repetitions of a block of restricted quantifiers:

\[
QB = \left[ (Q_1x_1.M_1) \cdots (Q_kx_k.M_k) \right]; \quad M_i \text{ quantifier-free}
\]

\[
\varphi_n = \underbrace{[QB][QB] \cdots [QB]}_{t(n)} M_0
\]
Thm: For all constructible, polynomially bounded $t(n)$,

$$\text{CRAM}[t(n)] = \text{IND}[t(n)] = \text{FO}[t(n)]$$

Thm: For all $t(n)$, even beyond polynomial,

$$\text{CRAM}[t(n)] = \text{FO}[t(n)]$$
**Thm:** For \( v = 1, 2, \ldots \), \( \text{DSPACE}[n^v] = \text{VAR}[v + 1] \)

Number of variables corresponds to amount of hardware.

Since variables range over a universe of size \( n \), a constant number of variables can specify a polynomial number of gates:

A bounded number of variables corresponds to polynomially much hardware.
Natural complexity classes have natural descriptive characterizations

<table>
<thead>
<tr>
<th></th>
<th>Computational Complexity</th>
<th>Descriptive Complexity</th>
</tr>
</thead>
<tbody>
<tr>
<td>3-COLOR</td>
<td>NP</td>
<td>SO∃</td>
</tr>
<tr>
<td>REACH</td>
<td>NSPACE[log (n)]</td>
<td>FO-VAR[log (n), 4]</td>
</tr>
<tr>
<td>Addition</td>
<td>CRAM[1]</td>
<td>FO</td>
</tr>
</tbody>
</table>

\[
\begin{align*}
\text{FO}(\text{PFP}) &= \text{SO}(\text{TC}) = \text{PSPACE} \\
\text{SO∃} &= \text{NP} \\
\text{FO}(\text{LFP}) &= \text{P} \\
\text{FO}(\text{TC}) &= \text{NSPACE}[\text{log } n] \\
\text{FO}(\text{MAJ}) &= \text{ThC}^0
\end{align*}
\]
Fact: For constructible \( t(n) \), \( \text{FO}[t(n)] = \text{CRAM}[t(n)] \)

Fact: For \( k = 1, 2, \ldots \), \( \text{VAR}[k + 1] = \text{DSPACE}[n^k] \)

The complexity of computing a query is closely tied to the complexity of describing the query.

\[
\begin{align*}
P &= \text{NP} \quad \iff \quad \text{FO}(\text{LFP}) = \text{SO} \\
\text{ThC}^0 &= \text{NP} \quad \iff \quad \text{FO}(\text{MAJ}) = \text{SO} \\
P &= \text{PSPACE} \quad \iff \quad \text{FO}(\text{LFP}) = \text{SO}(\text{TC})
\end{align*}
\]
<table>
<thead>
<tr>
<th>Arithmetic Hierarchy</th>
<th>Recursive</th>
<th>Primitive Recursive</th>
</tr>
</thead>
<tbody>
<tr>
<td>co-r.e. complete</td>
<td>FO(N)</td>
<td>co-r.e.</td>
</tr>
<tr>
<td>FO ∨ (N)</td>
<td>r.e.</td>
<td>r.e. complete</td>
</tr>
<tr>
<td>FO ∃(N)</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Polynomial–Time Hierarchy</th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>co–NP complete</td>
<td>FO[2^{O(1)} n]</td>
<td>SO[2^{n^{O(1)}}]</td>
</tr>
<tr>
<td>co–NP</td>
<td>FO[n^{O(1)}]</td>
<td>SO[n^{O(1)}]</td>
</tr>
<tr>
<td>NP</td>
<td>PSPACE</td>
<td>FO(PFP)</td>
</tr>
<tr>
<td>NP ∩ co–NP</td>
<td></td>
<td>SO(LFP)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>P</td>
<td>SO–Horn</td>
<td>“truly feasible”</td>
</tr>
<tr>
<td>FO[n^{O(1)}]</td>
<td>NC</td>
<td></td>
</tr>
<tr>
<td>FO(LFP)</td>
<td>NC^2</td>
<td></td>
</tr>
<tr>
<td>FO[(log n)^{O(1)}]</td>
<td>NC^1</td>
<td></td>
</tr>
<tr>
<td>FO(CFL)</td>
<td>sAC^1</td>
<td></td>
</tr>
<tr>
<td>FO(TC)</td>
<td>NSPACE[log n]</td>
<td>SO–Krom</td>
</tr>
<tr>
<td>FO(DTC)</td>
<td>DSPACE[log n]</td>
<td></td>
</tr>
<tr>
<td>FO(REGULAR)</td>
<td>NC^1</td>
<td></td>
</tr>
<tr>
<td>FO(M)</td>
<td>ThC^0</td>
<td></td>
</tr>
<tr>
<td>FO</td>
<td>Logarithmic–Time Hierarchy</td>
<td>AC^0</td>
</tr>
</tbody>
</table>

Neil Immerman  
Short Survey of Descriptive Complexity Plus . . .
David: hide any differences between the two structures

Two-person combinatorial game for characterizing what is expressible in a given quantifier depth.

**Fundamental Thm:** D has a winning strategy on the $m$-move, $k$-pebble game on $A$, $B$ iff $A$ and $B$ agree on all formulas using $k$ variables and quantifier depth $m$.

$$A \sim^k_m B \iff A \equiv^k_m B$$

Samson: show a difference
We Need an Ordering on the Universe

\[ G = (V, E); \quad V = \{0, 1, \ldots, n - 1\}; \quad 0 < 1 < \cdots < n - 1 \]

An unordered graph makes sense mathematically, but you can’t store such an object in a computer as far as I know.

If you remove the ordering then the first-order descriptive characterizations fail:

\textbf{EVEN} requires \( \Omega(n) \) variables without ordering.

Thus, \textbf{EVEN} \( \not\in \text{FO}(\text{wo}\leq)[2^{n^{O(1)}}]; \quad (\text{FO}[2^{n^{O(1)}}] = \text{PSPACE}) \)

Fagin (\( \text{SO}\exists = \text{NP} \)) didn’t run into this problem because in \( \text{SO}\exists \) you can guess an ordering (unless we are dealing with \( \text{SO}\exists(\text{monadic}) \)).
**Theorem** [Ben Rossman] Any first-order formula with any numeric relations ($\leq, +, \times, \ldots$) that means “I have a clique of size $k$” must have at least $k/4$ variables.

Creative new proof idea using Håstad’s Switching Lemma gives the essentially optimal bound.

This lower bound is for a fixed formula, if it were for a sequence of polynomially-sized formulas, i.e., a fixed-point formula, it would follow that $\text{CLIQUE} \not\in \text{P}$ and thus $\text{P} \neq \text{NP}$.

**Best previous bounds:**

- $k$ variables necessary and sufficient without ordering or other numeric relations [I 1980].

- Nothing was known with ordering except for the trivial fact that 2 variables are not enough.
**Theorem** [Martin Grohe] Fixed-Point Logic with Counting captures Polynomial Time on all classes of graphs with excluded minors.

Grohe proves that for every class of graphs with excluded minors, there is a constant $k$ such that two graphs of the class are isomorphic iff they agree on all $k$-variable formulas in fixed-point logic with counting.

Using Ehrenfeucht-Fraïssé games, this can be checked in polynomial time, $(O(n^k(\log n)))$. In the same time we can give a canonical description of the isomorphism type of any graph in the class. Thus every class of graphs with excluded minors admits the same general polynomial time canonization algorithm: we’re isomorphic iff we agree on all formulas in $C_k$ and in particular, you are isomorphic to me iff your $C_k$ canonical description is equal to mine.
Related previous work

▶ **FO(LFP, COUNT)** introduced and shown to give linear time isomorphism algorithms for trees and almost all random graphs [I & Lander 1990].

Question asked: is **FO(LFP, COUNT) = order invariant P**?

▶ **FO(LFP, COUNT) ≠ order invariant P**, and in fact **Ω(n)** variables are needed to express a simple PTIME problem [Cai, Fürer & I 1989].
Express problem to be solved in convenient high level language, e.g. variant of SO.

Automatically translate if possible to an equivalent expression in a goal language.

Every sentence in the goal language can be automatically converted to correct and efficient executable code.

Think of SQL as an analogy.

Joint work in progress with Sumit Gulwani, Shachar Itzhaky, and Mooly Sagiv.
Example: Testing If a Graph is Bipartite

$$\Phi_{bp} \equiv \exists S^1 \forall xy (E(x, y) \rightarrow (S(x) \leftrightarrow \neg S(y)))$$
Example: Testing If a Graph is Bipartite

\[ \Phi_{bp} \equiv \exists S^1 \forall xy (E(x, y) \rightarrow (S(x) \leftrightarrow \neg S(y))) \]

Maintain Invariant: \( \beta \equiv \forall xy (E(x, y) \rightarrow (S(x) \leftrightarrow \neg S(y))) \)
incrementally as we add edges to an initially empty graph.
Example: Testing If a Graph is Bipartite

Φ_{bp} \equiv \exists S^1 \forall xy (E(x, y) \rightarrow (S(x) \leftrightarrow \neg S(y)))

Maintain Invariant: \( \beta \equiv \forall xy (E(x, y) \rightarrow (S(x) \leftrightarrow \neg S(y))) \)
incrementally as we add edges to an initially empty graph.

Base Case: \( G_0 = (V, \emptyset, \emptyset) \models \beta \)
Example: Testing If a Graph is Bipartite

\[ \Phi_{bp} \equiv \exists S^1 \forall xy (E(x, y) \rightarrow (S(x) \leftrightarrow \neg S(y))) \]

Maintain Invariant: \( \beta \equiv \forall xy (E(x, y) \rightarrow (S(x) \leftrightarrow \neg S(y))) \)
incrementally as we add edges to an initially empty graph.

Base Case: \( G_0 = (V, \emptyset, \emptyset) \models \beta \)

Inductively Assume: \( G = (V, E, S) \models \beta \) and add add an edge \( (a, b) \): \( E' := E \cup \{(a, b)\} \)
Example: Testing If a Graph is Bipartite

\[ \Phi_{bp} \equiv \exists S^1 \forall xy (E(x, y) \rightarrow (S(x) \leftrightarrow \neg S(y))) \]

Maintain Invariant: \( \beta \equiv \forall xy (E(x, y) \rightarrow (S(x) \leftrightarrow \neg S(y))) \)

incrementally as we add edges to an initially empty graph.

Base Case: \( G_0 = (V, \emptyset, \emptyset) \models \beta \)

Inductively Assume: \( G = (V, E, S) \models \beta \) and add add an edge \( (a, b) \): \( E' := E \cup \{(a, b)\} \)

Case 1: \( (V, E', S) \models \beta \) so we’re fine.
Case 2: \((V, E', S) \models \neg \beta\)

\((G, a/x, b/y) \models (S(x) \leftrightarrow S(y))\)
Case 2: \((V, E', S) \models \neg \beta\)

\[(G, a/x, b/y) \models (S(x) \leftrightarrow S(y))\]

WLOG to reestablish \(\beta\) we must change the value of \(S(a)\).
Case 2: \((V, E', S) \models \neg \beta\)

\[(G, a/x, b/y) \models (S(x) \leftrightarrow S(y))\]

WLOG to reestablish \(\beta\) we must change the value of \(S(a)\).

**Naive Incremental Algorithm:** If \(b\) is in this connected component, report failure
Else: change the value of \(S(c)\) for all \(c\) in the connected component of \(a\).

Naive Algorithm takes time \(O(nm)\).
Better Incremental Algorithm

Keep track of connected component of $a$ in two disjoint parts:

$$S(a) = C(a) \cap S$$
$$\bar{S}(a) = C(a) \cap \bar{S}$$
Better Incremental Algorithm

Keep track of connected component of $a$ in two disjoint parts:

$$S(a) = C(a) \cap S$$
$$\overline{S}(a) = C(a) \cap \overline{S}$$

1. \textbf{if} ($S(a) = \overline{S}(b)$): \textbf{return} (“not bipartite”)
2. $S(b) := S(b) \cup \overline{S}(a)$; \hspace{1em} $\overline{S}(b) := \overline{S}(b) \cup S(a)$
Better Incremental Algorithm

Keep track of connected component of $a$ in two disjoint parts:

$$S(a) = C(a) \cap S \quad \bar{S}(a) = C(a) \cap \bar{S}$$

1. if ($S(a) = \bar{S}(b)$): return(“not bipartite”)
2. $S(b) := S(b) \cup \bar{S}(a); \quad \bar{S}(b) := \bar{S}(b) \cup S(a)$

Finally, revisit case 1 and maintain the data structure:

$$S(b) := S(b) \cup S(a); \quad \bar{S}(b) := \bar{S}(b) \cup \bar{S}(a)$$
Better Incremental Algorithm

Keep track of connected component of $a$ in two disjoint parts:

$$S(a) = C(a) \cap S \quad \bar{S}(a) = C(a) \cap \bar{S}$$

1. if ($S(a) = \bar{S}(b)$): return ("not bipartite")
2. $S(b) := S(b) \cup \bar{S}(a); \quad \bar{S}(b) := \bar{S}(b) \cup S(a)$

Finally, revisit case 1 and maintain the data structure:

$$S(b) := S(b) \cup S(a); \quad \bar{S}(b) := \bar{S}(b) \cup \bar{S}(a)$$

With the sets $S, \bar{S}$ instantiated using Union/Find, the complexity of this incremental algorithm is then essentially linear, i.e, $O(m)$. 
## Arithmetic Hierarchy

<table>
<thead>
<tr>
<th>FO(N)</th>
<th>FO E(N)</th>
<th>FO A(N)</th>
</tr>
</thead>
<tbody>
<tr>
<td>co-r.e. complete</td>
<td>r.e.</td>
<td>co-r.e. complete</td>
</tr>
</tbody>
</table>

## Recursive

<table>
<thead>
<tr>
<th>FO V(N)</th>
<th>FO E(N)</th>
</tr>
</thead>
<tbody>
<tr>
<td>co-r.e. complete</td>
<td>r.e. complete</td>
</tr>
</tbody>
</table>

## Primitive Recursive

<table>
<thead>
<tr>
<th>SO[2 (n^{O(1)})]</th>
<th>EXPTIME</th>
<th>SO(LFP)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FO[2 (n^{O(1)})]</td>
<td>SO[n^{O(1)}]</td>
<td>PSPACE</td>
</tr>
<tr>
<td>SO</td>
<td>FO(PFP)</td>
<td>SO(TC)</td>
</tr>
</tbody>
</table>

## Polynomial–Time Hierarchy

<table>
<thead>
<tr>
<th>FO[n^{O(1)}]</th>
<th>SO (\cap)</th>
<th>co–NP complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>FO</td>
<td>NP</td>
<td>co–NP</td>
</tr>
<tr>
<td>SO V</td>
<td>NP</td>
<td>SO E</td>
</tr>
</tbody>
</table>

## Logarithmic–Time Hierarchy

<table>
<thead>
<tr>
<th>FO[(log n)^{O(1)}]</th>
<th>NC</th>
</tr>
</thead>
<tbody>
<tr>
<td>FO[log n]</td>
<td>NC^2</td>
</tr>
<tr>
<td>FO(CFL)</td>
<td>sAC^1</td>
</tr>
<tr>
<td>FO(TC)</td>
<td>NSPACE[log n]</td>
</tr>
<tr>
<td>FO(DTC)</td>
<td>DSPACE[log n]</td>
</tr>
<tr>
<td>FO(REGULAR)</td>
<td>NC^1</td>
</tr>
<tr>
<td>FO(M)</td>
<td>ThC^0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>FO</th>
<th>AC^0</th>
</tr>
</thead>
</table>

Neil Immerman  
Short Survey of Descriptive Complexity Plus …