# Logic, Courcelle's theorem and Applications

M. Praveen

The Institute of Mathematical Sciences, India

IMPECS School on Parameterized and Exact Computation
December 2010

# Outline

# Using logic to define problems

A graph is three colorable iff the set of vertices can be partitioned into three parts such that, there is no edge between vertices in the same partition.

# Using logic to define problems

A graph is three colorable iff the set of vertices can be partitioned into three parts such that, there is no edge between vertices in the same partition.

$$
\begin{aligned}
&\exists X_1, X_2, X_3 \quad \forall x \\
&\{ \\
&\qquad\qquad [x \in X_1 \vee x \in X_2 \vee x \in X_3] \\
&\qquad\qquad \wedge \neg[(x \in X_1 \wedge x \in X_2) \vee (x \in X_2 \wedge x \in X_3) \\
&\qquad\qquad\qquad \vee (x \in X_1 \wedge x \in X_3)] \\
&\} \\
&\wedge \forall y, z \\
&\{ \\
&\qquad\qquad E(y,z) \Rightarrow \neg(y \in X_1 \wedge z \in X_1) \\
&\qquad\qquad \wedge E(y,z) \Rightarrow \neg(y \in X_2 \wedge z \in X_2) \\
&\qquad\qquad \wedge E(y,z) \Rightarrow \neg(y \in X_3 \wedge z \in X_3) \\
&\}
\end{aligned} \tag{1}
$$

# Using logic to define problems

A graph is three colorable iff the set of vertices can be partitioned into three parts such that, there is no edge between vertices in the same partition.

$$
\begin{aligned}
&\exists X_1, X_2, X_3 \quad \forall x \\
&\{ \\
&\qquad\qquad [x \in X_1 \vee x \in X_2 \vee x \in X_3] \\
&\qquad\qquad \wedge \neg[(x \in X_1 \wedge x \in X_2) \vee (x \in X_2 \wedge x \in X_3) \\
&\qquad\qquad\qquad \vee (x \in X_1 \wedge x \in X_3)] \\
&\} \\
&\wedge \forall y, z \\
&\{ \\
&\qquad\qquad E(y,z) \Rightarrow \neg(y \in X_1 \wedge z \in X_1) \\
&\qquad\qquad \wedge E(y,z) \Rightarrow \neg(y \in X_2 \wedge z \in X_2) \\
&\qquad\qquad \wedge E(y,z) \Rightarrow \neg(y \in X_3 \wedge z \in X_3) \\
&\}
\end{aligned} \tag{1}
$$

# Using logic to define problems

A graph is three colorable iff the set of vertices can be partitioned into three parts such that, there is no edge between vertices in the same partition.

$$
\begin{aligned}
&\exists X_1, X_2, X_3 \quad \forall x \\
&\{ \\
&\qquad\qquad [x \in X_1 \vee x \in X_2 \vee x \in X_3] \\
&\qquad\qquad \wedge \neg[(x \in X_1 \wedge x \in X_2) \vee (x \in X_2 \wedge x \in X_3) \\
&\qquad\qquad\qquad \vee (x \in X_1 \wedge x \in X_3)] \\
&\} \\
&\wedge \forall y, z \\
&\{ \\
&\qquad\qquad E(y,z) \Rightarrow \neg(y \in X_1 \wedge z \in X_1) \\
&\qquad\qquad \wedge E(y,z) \Rightarrow \neg(y \in X_2 \wedge z \in X_2) \\
&\qquad\qquad \wedge E(y,z) \Rightarrow \neg(y \in X_3 \wedge z \in X_3) \\
&\}
\end{aligned}
\tag{1}
$$

# Using logic to define problems

A graph is three colorable iff the set of vertices can be partitioned into three parts such that, there is no edge between vertices in the same partition.

$$
\begin{aligned}
&\exists X_1, X_2, X_3 \quad \forall x \\
&\{ \\
&\qquad\qquad [x \in X_1 \vee x \in X_2 \vee x \in X_3] \\
&\qquad\qquad \wedge \neg[(x \in X_1 \wedge x \in X_2) \vee (x \in X_2 \wedge x \in X_3) \\
&\qquad\qquad\qquad \vee (x \in X_1 \wedge x \in X_3)] \\
&\} \\
&\wedge \forall y, z \\
&\{ \\
&\qquad\qquad E(y,z) \Rightarrow \neg(y \in X_1 \wedge z \in X_1) \\
&\qquad\qquad \wedge E(y,z) \Rightarrow \neg(y \in X_2 \wedge z \in X_2) \\
&\qquad\qquad \wedge E(y,z) \Rightarrow \neg(y \in X_3 \wedge z \in X_3) \\
&\}
\end{aligned} \tag{1}
$$

# Monadic Second Order (MSO) logic of graphs

- Let $x, y, x_1, x_2, x_3, \ldots$ be variables that denote vertices.
- Let $X, Y, X_1, X_2, X_3, \ldots$ be variables that denote subsets of vertices.
- Let $E(x_1, x_2)$ denote the fact that there is an edge between $x_1$ and $x_2$.

# Monadic Second Order (MSO) logic of graphs

- Let $x, y, x_1, x_2, x_3, \ldots$ be variables that denote vertices.
- Let $X, Y, X_1, X_2, X_3, \ldots$ be variables that denote subsets of vertices.
- Let $E(x_1, x_2)$ denote the fact that there is an edge between $x_1$ and $x_2$.
- Monadic Second Order logic formulas (denoted as $\phi, \phi_1, \phi_2$ etc.) are those that can be constructed using the following:
  - $x \in Y$
  - $x_1 = x_2$
  - $E(x_1, x_2)$
  - $\phi_1 \wedge \phi_2$, $\phi_1 \vee \phi_2$, $\neg\phi_1$
  - $\exists x \phi, \forall x \phi$
  - $\exists X \phi, \forall X \phi$

# Monadic Second Order (MSO) logic of graphs

- Let $x, y, x_1, x_2, x_3, \ldots$ be variables that denote vertices.
- Let $X, Y, X_1, X_2, X_3, \ldots$ be variables that denote subsets of vertices.
- Let $E(x_1, x_2)$ denote the fact that there is an edge between $x_1$ and $x_2$.
- Monadic Second Order logic formulas (denoted as $\phi, \phi_1, \phi_2$ etc.) are those that can be constructed using the following:
  - $x \in Y$
  - $x_1 = x_2$
  - $E(x_1, x_2)$
  - $\phi_1 \wedge \phi_2$, $\phi_1 \vee \phi_2$, $\neg\phi_1$
  - $\exists x \phi, \forall x \phi$
  - $\exists X \phi, \forall X \phi$
  - For Counting MSO, add $|X| \equiv q \mod p$, $p, q \in \mathbb{N}$.

# MSO logic of graphs contd…

- In $\exists x \phi$, all occurrences of $x$ inside $\phi$ are said to be bound by the quantifier $\exists$ occurring in front of $\phi$. Similarly for $\forall x \phi$, $\exists X \phi$ and $\forall X \phi$.

# MSO logic of graphs contd...

- In $\exists x \phi$, all occurrences of $x$ inside $\phi$ are said to be bound by the quantifier $\exists$ occurring in front of $\phi$. Similarly for $\forall x \phi$, $\exists X \phi$ and $\forall X \phi$.
- Variables not bounded by any quantifier are said be free. Ex: $\exists x_1 E(x_2, x_1)$.

# Courcelle's theorem

- Let $G$ be a graph.
- Let $\phi$ be a MSO sentence (a MSO formula without free variables).
- Let $\mathit{treewidth}(G) + \mathit{size}(\phi)$ be the parameter.

# Courcelle's theorem

- Let $G$ be a graph.
- Let $\phi$ be a MSO sentence (a MSO formula without free variables).
- Let $\mathit{treewidth}(G) + \mathit{size}(\phi)$ be the parameter.
- [Courcelle's theorem]: Checking whether $G$ satisfies $\phi$ is Fixed Parameter Tractable.

# Courcelle's theorem

- Let $G$ be a graph.
- Let $\phi$ be a MSO sentence (a MSO formula without free variables).
- Let $\mathit{treewidth}(G) + \mathit{size}(\phi)$ be the parameter.
- [Courcelle's theorem]: Checking whether $G$ satisfies $\phi$ is Fixed Parameter Tractable. There is an algorithm with running time $f(\mathit{treewidth}(G), \mathit{size}(\phi))n$.

# Example: CNF SAT

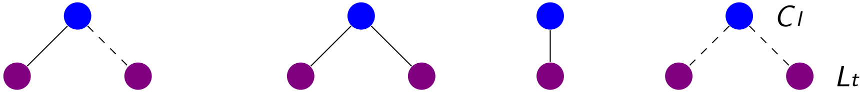
$$\underset{c\ell_1}{(x_1 \vee \neg x_2)} \quad \wedge \quad \underset{c\ell_2}{(x_3 \vee x_4)} \quad \wedge \quad \underset{c\ell_3}{x_5} \quad \wedge \quad \underset{c\ell_4}{(\neg x_6 \vee \neg x_7)}$$

CNF Satisfiability: There is a subset of variables such that, for every clause

Either there is a variable in the subset occurring positively

Or there is a variable not in the subset occurring negatively.

# Example: CNF SAT

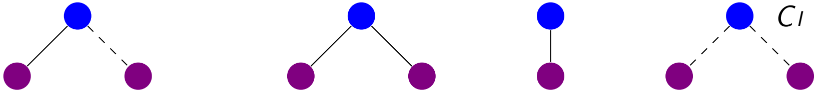



CNF Satisfiability: There is a subset of variables such that, for every clause

Either there is a variable in the subset occurring positively

Or there is a variable not in the subset occurring negatively.

# Example: CNF SAT



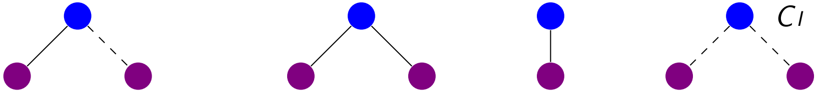

CNF Satisfiability: There is a subset of variables such that, for every clause

**Either** there is a variable in the subset occurring positively

**Or** there is a variable not in the subset occurring negatively.

$$\exists T_r \subseteq L_t : \forall c_\ell \in C_\ell : \\ \left[ \left(\exists l_t \in T_r : E(c_\ell, l_t)\right) \vee \left(\exists l_t \in L_t \setminus T_r : \overline{E}(c_\ell, l_t)\right)\right] \tag{2}$$

# Example: CNF SAT



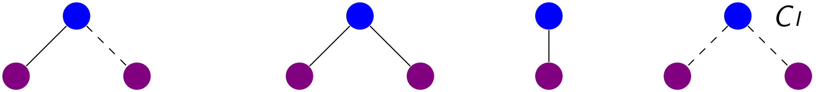

CNF Satisfiability: There is a subset of variables such that, for every clause

Either there is a variable in the subset occurring positively

Or there is a variable not in the subset occurring negatively.

$$\exists T_r \subseteq L_t : \forall c_\ell \in C_\ell : \\ \left[ (\exists l_t \in T_r : E(c_\ell, l_t)) \vee \left(\exists l_t \in L_t \setminus T_r : \overline{E}(c_\ell, l_t)\right)\right] \tag{2}$$

# Example: CNF SAT



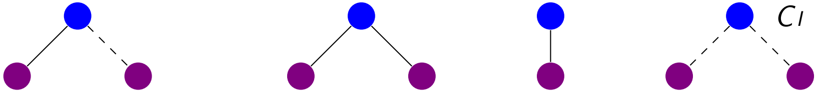

CNF Satisfiability: There is a subset of variables such that, for every clause

Either there is a variable in the subset occurring positively

Or there is a variable not in the subset occurring negatively.

$$\exists T_r \subseteq L_t : \forall c_\ell \in C_\ell : \\ \left[ \left(\exists l_t \in T_r : E(c_\ell, l_t)\right) \vee \left(\exists l_t \in L_t \setminus T_r : \overline{E}(c_\ell, l_t)\right) \right] \tag{2}$$

# Example: CNF SAT



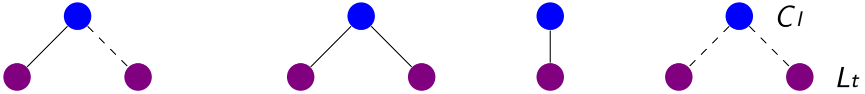

CNF Satisfiability: There is a subset of variables such that, for every clause

**Either** there is a variable in the subset occurring positively

**Or** there is a variable not in the subset occurring negatively.

$$\exists T_r \subseteq L_t : \forall c_\ell \in C_\ell : \\ \left[ (\exists l_t \in T_r : E(c_\ell, l_t)) \vee \left(\exists l_t \in L_t \setminus T_r : \overline{E}(c_\ell, l_t)\right)\right] \tag{2}$$

# Example: CNF SAT




CNF Satisfiability: There is a subset of variables such that, for every clause

Either there is a variable in the subset occurring positively

Or there is a variable not in the subset occurring negatively.

$$\exists T_r \subseteq L_t : \forall c_\ell \in C_\ell : \\ \left[ \left(\exists l_t \in T_r : E(c_\ell, l_t)\right) \vee \left(\exists l_t \in L_t \setminus T_r : \overline{E}(c_\ell, l_t)\right) \right] \tag{2}$$

# Dominating set of size three

A graph has a dominating set of size three iff there is a set of three vertices such that all other vertices are adjacent to some vertex in the set.

# Dominating set of size three

A graph has a dominating set of size three iff there is a set of three vertices such that all other vertices are adjacent to some vertex in the set.

$$\exists x_1, x_2, x_3 \quad \forall x$$
$$\{$$
$$(x = x_1 \vee x = x_2 \vee x = x_3) \tag{3}$$
$$\vee \exists y \quad E(x, y) \wedge (y = x_1 \vee y = x_2 \vee y = x_3)$$
$$\}$$

# Dominating set of size three

A graph has a dominating set of size three iff there is a set of three vertices such that all other vertices are adjacent to some vertex in the set.

$$
\begin{aligned}
&\exists x_1, x_2, x_3 \quad \forall x \\
&\{ \\
&\qquad\qquad (x = x_1 \vee x = x_2 \vee x = x_3) \\
&\qquad\qquad \vee \exists y \quad E(x, y) \wedge (y = x_1 \vee y = x_2 \vee y = x_3) \\
&\}
\end{aligned} \tag{3}
$$

# Dominating set of size three

A graph has a dominating set of size three iff there is a set of three vertices such that all other vertices are adjacent to some vertex in the set.

$$\exists x_1, x_2, x_3 \quad \forall x$$
$$\{$$
$$(x = x_1 \vee x = x_2 \vee x = x_3)$$
$$\vee \exists y \quad E(x, y) \wedge (y = x_1 \vee y = x_2 \vee y = x_3) \tag{3}$$
$$\}$$

# Dominating set of size three

A graph has a dominating set of size three iff there is a set of three vertices such that all other vertices are adjacent to some vertex in the set.

$$
\begin{aligned}
&\exists x_1, x_2, x_3 \quad \forall x \\
&\{ \\
&\qquad (x = x_1 \vee x = x_2 \vee x = x_3) \\
&\qquad \vee \exists y \quad E(x, y) \wedge (y = x_1 \vee y = x_2 \vee y = x_3) \\
&\}
\end{aligned} \tag{3}
$$

For each dominating set of size $k$, a formula can be written.

# Extensions - Motivation

In a graph $G$, a subset $X$ of vertices is a dominating set iff all other vertices are adjacent to some vertex in $X$.

# Extensions - Motivation

In a graph $G$, a subset $X$ of vertices is a dominating set iff all other vertices are adjacent to some vertex in $X$.

$$
\begin{aligned}
& ds(X) = \forall x \\
& \{ \\
& \qquad\qquad x \in X \\
& \qquad\qquad \vee \exists y \quad E(x, y) \wedge y \in X \\
& \}
\end{aligned}
\tag{4}
$$

# Extensions - Motivation

In a graph $G$, a subset $X$ of vertices is a dominating set iff all other vertices are adjacent to some vertex in $X$.

$$ds(X) = \forall x \left\{ \begin{array}{l} x \in X \\ \vee \exists y \quad E(x,y) \wedge y \in X \end{array} \right\} \tag{4}$$

# Extensions - Motivation

In a graph $G$, a subset $X$ of vertices is a dominating set iff all other vertices are adjacent to some vertex in $X$.

$$ds(X) = \forall x \left\{ \begin{array}{l} x \in X \\ \vee \exists y \quad E(x, y) \wedge y \in X \end{array} \right\} \tag{4}$$

Smallest dominating set: what is the size of a smallest subset $X$ of vertices such that $G$ satisfies $ds(X)$?

# Extended MSO

- Let $\phi(X_1, \cdots, X_l)$ be a MSO formula with free variables $X_1, \cdots, X_l$.

# Extended MSO

- Let $\phi(X_1, \cdots, X_l)$ be a MSO formula with free variables $X_1, \cdots, X_l$.
- [Arnborg, Lagergren, Seese]: The following problem is Fixed Parameter Tractable: Maximising/minimizing any linear combination of $|X_1|, \cdots, |X_l|$.

# Extended MSO

- Let $\phi(X_1, \cdots, X_l)$ be a MSO formula with free variables $X_1, \cdots, X_l$.
- [Arnborg, Lagergren, Seese]: The following problem is Fixed Parameter Tractable: Maximising/minimizing any linear combination of $|X_1|, \cdots, |X_l|$.
- Many other extensions are also proved: adding conditions like $|X_1| > |X_2|$, $|X_1| + |X_2| \leq |X_3|$ and so on. However, the degree of the polynomial in the running time depends on the number of free variables.

# Proof Idea - Path graphs

- A path graph: 

# Proof Idea - Path graphs

- A path graph: 
- Presenting the above graph as input to an algorithm:
  $a \quad a \quad a \quad a \quad a.$

# Proof Idea - Path graphs

- A path graph: 
- Presenting the above graph as input to an algorithm:
  $a \quad a \quad a \quad a \quad a$.
- The second vertex is $x$: $\binom{a}{0}\binom{a}{1}\binom{a}{0}\binom{a}{0}\binom{a}{0}$.

# Proof Idea - Path graphs

- A path graph: 
- Presenting the above graph as input to an algorithm: $a \quad a \quad a \quad a \quad a$.
- The second vertex is $x$: $\binom{a}{0}\binom{a}{1}\binom{a}{0}\binom{a}{0}\binom{a}{0}$.
- The first, third and fourth vertices form the set $X$: $\begin{pmatrix} a \\ 0 \\ 1 \end{pmatrix}\begin{pmatrix} a \\ 1 \\ 0 \end{pmatrix}\begin{pmatrix} a \\ 0 \\ 1 \end{pmatrix}\begin{pmatrix} a \\ 0 \\ 1 \end{pmatrix}\begin{pmatrix} a \\ 0 \\ 0 \end{pmatrix}$.

# A question about path graphs

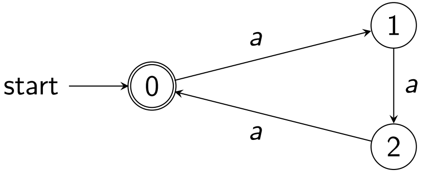
- Question: Is there an algorithm to check that the length of a path graph is $0 \mod 3$?

# A question about path graphs

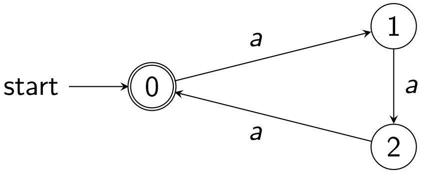
- Question: Is there an algorithm to check that the length of a path graph is $0 \mod 3$?
- Answer: Read from left to right, keeping track of the current length modulo 3.

# A question about path graphs

- Question: Is there an algorithm to check that the length of a path graph is $0 \mod 3$?
- Answer: Read from left to right, keeping track of the current length modulo 3.
-



# A question about path graphs

- Question: Is there an algorithm to check that the length of a path graph is $0 \mod 3$?
- Answer: Read from left to right, keeping track of the current length modulo 3.
- 


- Question: For what kind of questions can we construct finite state automata?

# Answer

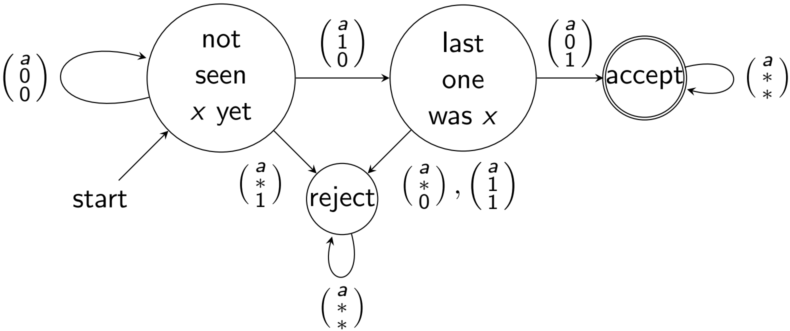
- Answer [Büchi, Elgot, Trakhtenbrot theorem]: For precisely those questions that can be stated in the MSO logic of path graphs.

# Answer

- Answer [Büchi, Elgot, Trakhtenbrot theorem]: For precisely those questions that can be stated in the MSO logic of path graphs.
- MSO logic of path graphs: In MSO logic of graphs, replace $E(x, y)$ by $y = x + 1$.

# Answer

- Answer [Büchi, Elgot, Trakhtenbrot theorem]: For precisely those questions that can be stated in the MSO logic of path graphs.
- MSO logic of path graphs: In MSO logic of graphs, replace $E(x, y)$ by $y = x + 1$.
- Automaton for checking $y = x + 1$: $\begin{pmatrix} a \\ x \\ y \end{pmatrix}$

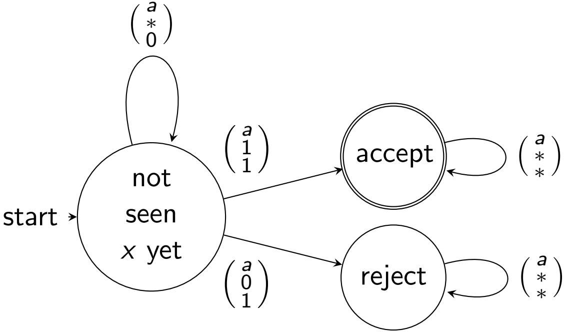


# Constructing automaton for MSO formulas

Automaton for checking $y = x$: Exercise.

# Constructing automaton for MSO formulas

Automaton for checking $y = x$: Exercise.
Automaton for checking $x \in X$: $\begin{pmatrix} a \\ X \\ x \end{pmatrix}$



# Automaton for MSO formulas contd…

- Automaton for $\neg\phi$:

# Automaton for MSO formulas contd...

- Automaton for $\neg\phi$: Complement the automaton $A_\phi$ of $\phi$.

# Automaton for MSO formulas contd. . .

- Automaton for $\neg\phi$: Complement the automaton $A_\phi$ of $\phi$.
- $\phi_1 \wedge \phi_2$:

# Automaton for MSO formulas contd. . .

- Automaton for $\neg\phi$: Complement the automaton $A_\phi$ of $\phi$.
- $\phi_1 \wedge \phi_2$: Path graphs accepted by both $A_{\phi_1}$ and $A_{\phi_2}$.

# Automaton for MSO formulas contd. . .

- Automaton for $\neg\phi$: Complement the automaton $A_\phi$ of $\phi$.
- $\phi_1 \wedge \phi_2$: Path graphs accepted by both $A_{\phi_1}$ and $A_{\phi_2}$. Take $A_{\phi_1} \cap A_{\phi_2}$.

# Automaton for MSO formulas contd...

- Automaton for $\neg\phi$: Complement the automaton $A_\phi$ of $\phi$.
- $\phi_1 \wedge \phi_2$: Path graphs accepted by both $A_{\phi_1}$ and $A_{\phi_2}$. Take $A_{\phi_1} \cap A_{\phi_2}$.
- $\phi_1 \vee \phi_2$:

# Automaton for MSO formulas contd...

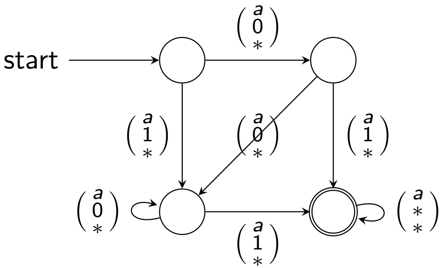
- Automaton for $\neg\phi$: Complement the automaton $A_\phi$ of $\phi$.
- $\phi_1 \wedge \phi_2$: Path graphs accepted by both $A_{\phi_1}$ and $A_{\phi_2}$. Take $A_{\phi_1} \cap A_{\phi_2}$.
- $\phi_1 \vee \phi_2$: $A_{\phi_1} \cup A_{\phi_2}$.

# Automaton for MSO formulas contd…

- Automaton for $\neg\phi$: Complement the automaton $A_\phi$ of $\phi$.
- $\phi_1 \wedge \phi_2$: Path graphs accepted by both $A_{\phi_1}$ and $A_{\phi_2}$. Take $A_{\phi_1} \cap A_{\phi_2}$.
- $\phi_1 \vee \phi_2$: $A_{\phi_1} \cup A_{\phi_2}$.
- $\exists x \phi(x, y)$:

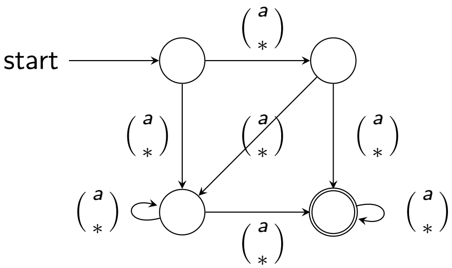
# Automaton for MSO formulas contd...

- Automaton for $\neg\phi$: Complement the automaton $A_\phi$ of $\phi$.
- $\phi_1 \wedge \phi_2$: Path graphs accepted by both $A_{\phi_1}$ and $A_{\phi_2}$. Take $A_{\phi_1} \cap A_{\phi_2}$.
- $\phi_1 \vee \phi_2$: $A_{\phi_1} \cup A_{\phi_2}$.
- $\exists x \phi(x, y)$: Suppose $A_{\phi(x,y)}$ is already constructed.



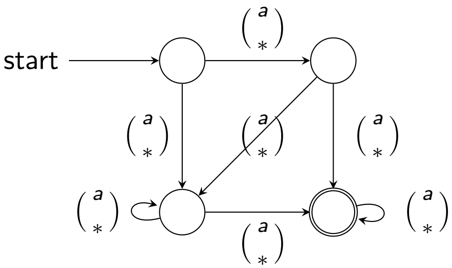
# Automaton for MSO formulas contd. . .

- Automaton for $\neg\phi$: Complement the automaton $A_\phi$ of $\phi$.
- $\phi_1 \wedge \phi_2$: Path graphs accepted by both $A_{\phi_1}$ and $A_{\phi_2}$. Take $A_{\phi_1} \cap A_{\phi_2}$.
- $\phi_1 \vee \phi_2$: $A_{\phi_1} \cup A_{\phi_2}$.
- $\exists x \phi(x, y)$: Suppose $A_{\phi(x,y)}$ is already constructed.



# Automaton for MSO formulas contd...

- Automaton for $\neg\phi$: Complement the automaton $A_\phi$ of $\phi$.
- $\phi_1 \wedge \phi_2$: Path graphs accepted by both $A_{\phi_1}$ and $A_{\phi_2}$. Take $A_{\phi_1} \cap A_{\phi_2}$.
- $\phi_1 \vee \phi_2$: $A_{\phi_1} \cup A_{\phi_2}$.
- $\exists x \phi(x, y)$: Suppose $A_{\phi(x,y)}$ is already constructed.




- That's all! A deterministic automaton may become non-deterministic.

# Automaton for MSO formulas contd...

- $\forall x \phi(x)$ is same as $\neg \exists x \neg \phi(x)$.

# Automaton for MSO formulas contd...

- $\forall x \phi(x)$ is same as $\neg \exists x \neg \phi(x)$.
- Construct a non-deterministic automaton for $\exists x \neg \phi(x)$ and complement it. This needs determinization and involves an exponential blow-up.

# Automaton for MSO formulas contd...

- $\forall x \phi(x)$ is same as $\neg \exists x \neg \phi(x)$.
- Construct a non-deterministic automaton for $\exists x \neg \phi(x)$ and complement it. This needs determinization and involves an exponential blow-up.
- Similarly handle $\exists X \phi(X)$ and $\forall X \phi(X)$.

# Automaton for MSO formulas contd...

- $\forall x \phi(x)$ is same as $\neg \exists x \neg \phi(x)$.
- Construct a non-deterministic automaton for $\exists x \neg \phi(x)$ and complement it. This needs determinization and involves an exponential blow-up.
- Similarly handle $\exists X \phi(X)$ and $\forall X \phi(X)$.
- Size of the automaton depends on the size of the formula. Let this size be $f(|\phi|)$.

# Automaton for MSO formulas contd...

- $\forall x \phi(x)$ is same as $\neg \exists x \neg \phi(x)$.
- Construct a non-deterministic automaton for $\exists x \neg \phi(x)$ and complement it. This needs determinization and involves an exponential blow-up.
- Similarly handle $\exists X \phi(X)$ and $\forall X \phi(X)$.
- Size of the automaton depends on the size of the formula. Let this size be $f(|\phi|)$.
- To check if a path graph of $n$ vertices satisfies $\phi$, just check if $A_\phi$ accepts the (sequence representing the) path graph.

# Automaton for MSO formulas contd...

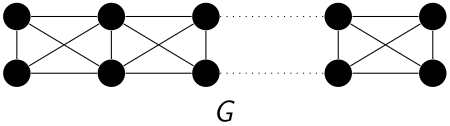
- $\forall x \phi(x)$ is same as $\neg \exists x \neg \phi(x)$.
- Construct a non-deterministic automaton for $\exists x \neg \phi(x)$ and complement it. This needs determinization and involves an exponential blow-up.
- Similarly handle $\exists X \phi(X)$ and $\forall X \phi(X)$.
- Size of the automaton depends on the size of the formula. Let this size be $f(|\phi|)$.
- To check if a path graph of $n$ vertices satisfies $\phi$, just check if $A_\phi$ accepts the (sequence representing the) path graph.
- This can be done in time $f(|\phi|)n$.

# Automaton for MSO formulas contd...

- $\forall x \phi(x)$ is same as $\neg \exists x \neg \phi(x)$.
- Construct a non-deterministic automaton for $\exists x \neg \phi(x)$ and complement it. This needs determinization and involves an exponential blow-up.
- Similarly handle $\exists X \phi(X)$ and $\forall X \phi(X)$.
- Size of the automaton depends on the size of the formula. Let this size be $f(|\phi|)$.
- To check if a path graph of $n$ vertices satisfies $\phi$, just check if $A_\phi$ accepts the (sequence representing the) path graph.
- This can be done in time $f(|\phi|)n$.
- Fixed Parameter Tractable when $|\phi|$ is a parameter.

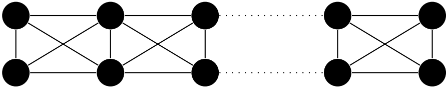
# Extension to graphs that are very near to being paths

# Extension to graphs that are very near to being paths




$\phi$

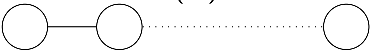
# Extension to graphs that are very near to being paths



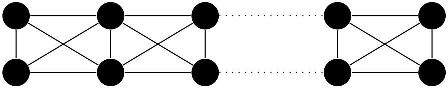
$G$

$\phi$

$P(G)$



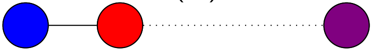
# Extension to graphs that are very near to being paths



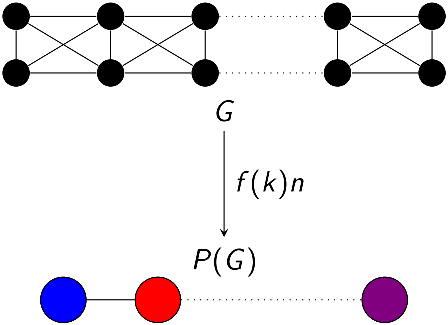
$G$

$\phi$

$P(G)$

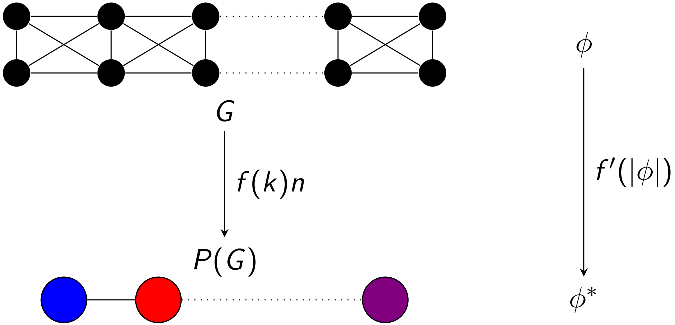


# Extension to graphs that are very near to being paths

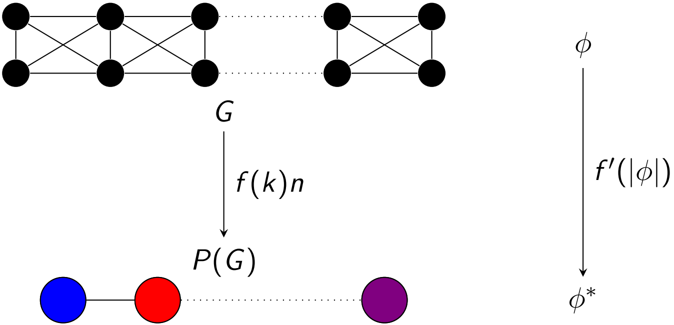



$\phi$

# Extension to graphs that are very near to being paths

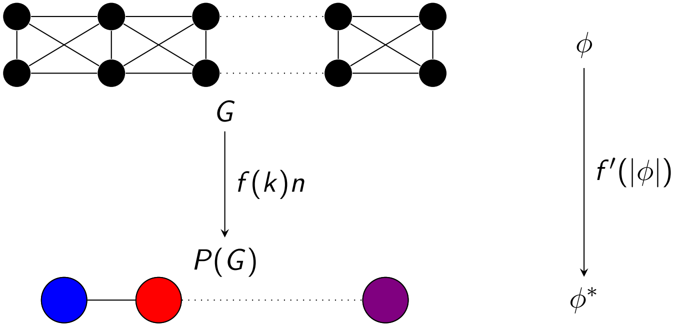


# Extension to graphs that are very near to being paths




$G$ satisfies $\phi$ iff $P(G)$ satisfies $\phi^*$.

# Extension to graphs that are very near to being paths




$G$ satisfies $\phi$ iff $P(G)$ satisfies $\phi^*$.
Check if $A_{\phi^*}$ accepts $P(G)$.

# Extension to treewidth

- [Doner, Thatcher, Wright]: Analogue of BET theorem for trees.

# Extension to treewidth

- [Doner, Thatcher, Wright]: Analogue of BET theorem for trees.
- For checking MSO properties of graphs with bounded treewidth, use tree decomposition instead of path decomposition. Use tree automata instead of the usual string automata.

# Lower bounds

- Consider the sentence $\psi = \forall x_1 \exists x_2 \forall x_3 \exists x_4 \cdots \forall x_9 \exists x_{10} \phi(x_1, \ldots, x_{10})$.
- If $A_{\phi(x_1,\ldots,x_{10})}$ has $m$ states, how many will $A_\psi$ have?

# Lower bounds

- Consider the sentence
  $\psi = \forall x_1 \exists x_2 \forall x_3 \exists x_4 \cdots \forall x_9 \exists x_{10} \phi(x_1, \ldots, x_{10})$.
- If $A_{\phi(x_1,\ldots,x_{10})}$ has $m$ states, how many will $A_\psi$ have?
- For every alternation in the quantifier sequence, a determinization and complementation is performed, incurring an exponential blowup.

# Lower bounds

- Consider the sentence $\psi = \forall x_1 \exists x_2 \forall x_3 \exists x_4 \cdots \forall x_9 \exists x_{10} \phi(x_1, \ldots, x_{10})$.
- If $A_{\phi(x_1,\ldots,x_{10})}$ has $m$ states, how many will $A_\psi$ have?
- For every alternation in the quantifier sequence, a determinization and complementation is performed, incurring an exponential blowup.
- The number of states will be $2^{2^{\cdot^{\cdot^{\cdot^{m}}}}}$.

# Lower bounds

- Consider the sentence $\psi = \forall x_1 \exists x_2 \forall x_3 \exists x_4 \cdots \forall x_9 \exists x_{10} \phi(x_1, \ldots, x_{10})$.
- If $A_{\phi(x_1,\ldots,x_{10})}$ has $m$ states, how many will $A_\psi$ have?
- For every alternation in the quantifier sequence, a determinization and complementation is performed, incurring an exponential blowup.
- The number of states will be $2^{2^{\cdot^{\cdot^{\cdot^{m}}}}}$.
- Question: can we do better?

# Lower bounds

- Consider the sentence $\psi = \forall x_1 \exists x_2 \forall x_3 \exists x_4 \cdots \forall x_9 \exists x_{10} \phi(x_1, \ldots, x_{10})$.
- If $A_{\phi(x_1,\ldots,x_{10})}$ has $m$ states, how many will $A_\psi$ have?
- For every alternation in the quantifier sequence, a determinization and complementation is performed, incurring an exponential blowup.
- The number of states will be $2^{2^{\cdot^{\cdot^{\cdot^{m}}}}}$.
- Question: can we do better?
- [Frick, Grohe]: No, unless $\mathrm{P}=\mathrm{NP}$.

# Generalizations and specializations

- Courcelle's theorem: MSO formulas and class of graphs with bounded treewidth.

# Generalizations and specializations

- Courcelle's theorem: MSO formulas and class of graphs with bounded treewidth.
- Take a weaker logic and a bigger class of graphs.

# Generalizations and specializations

- Courcelle's theorem: MSO formulas and class of graphs with bounded treewidth.
- Take a weaker logic and a bigger class of graphs.
- Weaker logic: Remove $\exists X$ and $\forall X$ from MSO (First Order logic, FO).

# Generalizations and specializations

- Courcelle's theorem: MSO formulas and class of graphs with bounded treewidth.
- Take a weaker logic and a bigger class of graphs.
- Weaker logic: Remove $\exists X$ and $\forall X$ from MSO (First Order logic, FO).
- Bigger class of graphs: graphs with bounded local treewidth.

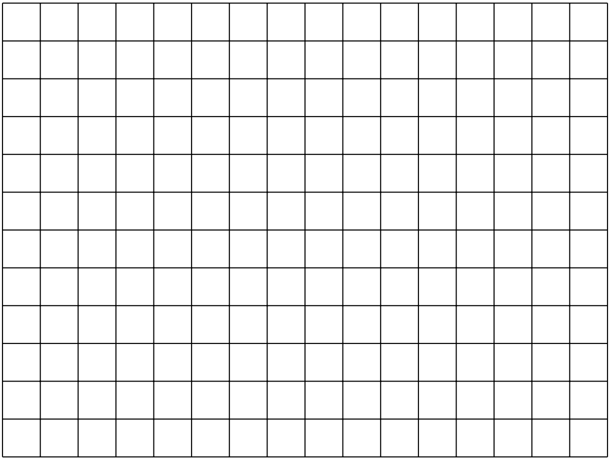
# Generalizations and specializations

- Courcelle's theorem: MSO formulas and class of graphs with bounded treewidth.
- Take a weaker logic and a bigger class of graphs.
- Weaker logic: Remove $\exists X$ and $\forall X$ from MSO (First Order logic, FO).
- Bigger class of graphs: graphs with bounded local treewidth.
- Bounded local treewidth: there is a function $f : \mathbb{N} \to \mathbb{N}$ such that any sphere of radius $r$ has treewidth at most $f(r)$.

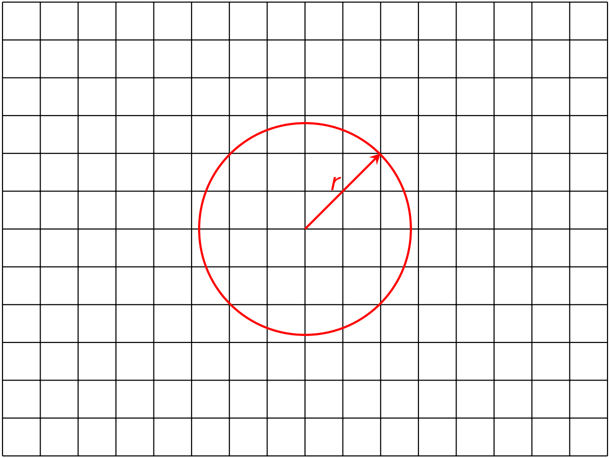
# Generalizations and specializations

- Courcelle's theorem: MSO formulas and class of graphs with bounded treewidth.
- Take a weaker logic and a bigger class of graphs.
- Weaker logic: Remove $\exists X$ and $\forall X$ from MSO (First Order logic, FO).
- Bigger class of graphs: graphs with bounded local treewidth.
- Bounded local treewidth: there is a function $f : \mathbb{N} \to \mathbb{N}$ such that any sphere of radius $r$ has treewidth at most $f(r)$.
- Example: For planar graphs, $f(r) = 3r$.

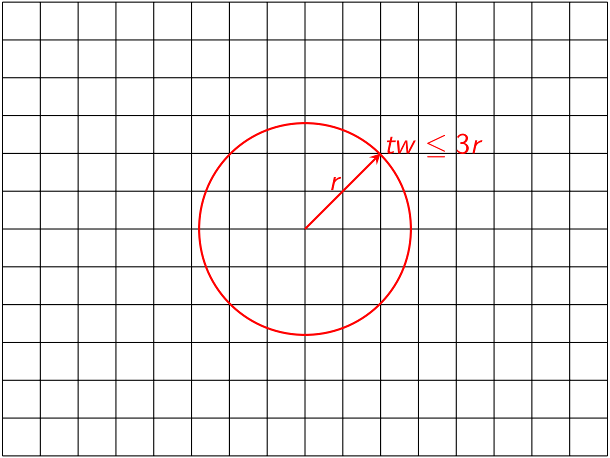
# Bounded local treewidth



# Bounded local treewidth



# Bounded local treewidth



# Bounded local treewidth contd...

- The treewidth of the whole graph may be very large, so Courcelle's theorem cannot be applied directly.

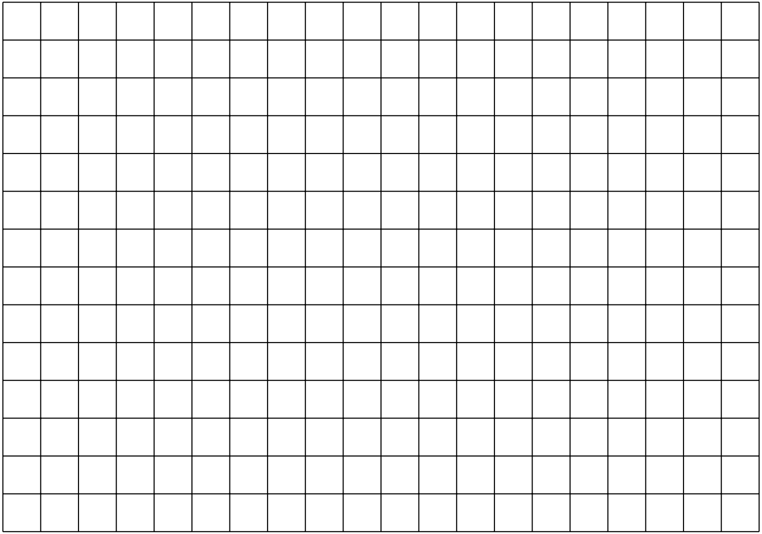
# Bounded local treewidth contd. . .

- The treewidth of the whole graph may be very large, so Courcelle's theorem cannot be applied directly.
- [Frick, Grohe]: If a class of graphs has effectively bounded local treewidth, then checking FO sentences on graphs from that class is Fixed Parameter Tractable, where the length of the FO sentence is the parameter.

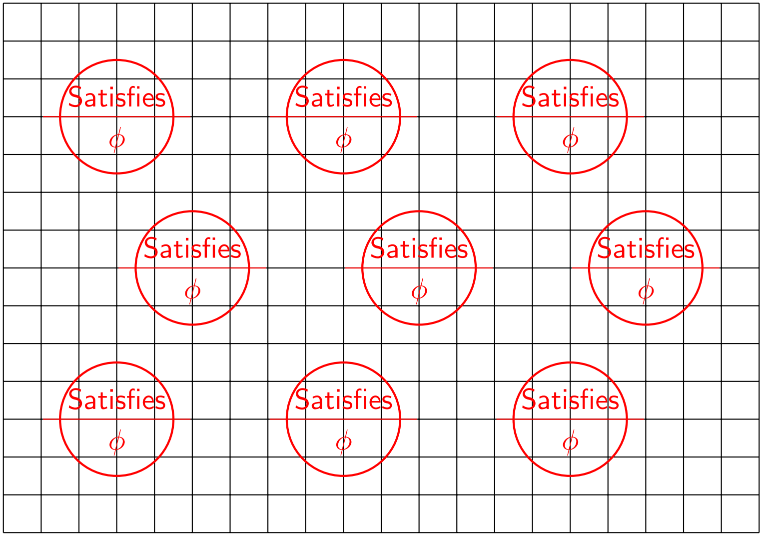
# Bounded local treewidth contd…

- The treewidth of the whole graph may be very large, so Courcelle's theorem cannot be applied directly.
- [Frick, Grohe]: If a class of graphs has effectively bounded local treewidth, then checking FO sentences on graphs from that class is Fixed Parameter Tractable, where the length of the FO sentence is the parameter.
- Proof relies on Gaifman's locality theorem: A given FO sentence can only reason about a fixed number of pairwise disjoint spheres that satisfy some FO property.

# Bounded local treewidth contd...



# Bounded local treewidth contd. . .



# Extending the generalization

- [Flum, Grohe]: For any class of graphs that excludes a minor, checking FO sentences is Fixed Parameter Tractable, where the length of the FO sentence is the parameter.

# Extending the generalization

- [Flum, Grohe]: For any class of graphs that excludes a minor, checking FO sentences is Fixed Parameter Tractable, where the length of the FO sentence is the parameter.
- [Dawar, Grohe, Kreutzer]: For any class of graphs that locally excludes a minor, checking FO sentences is Fixed Parameter Tractable, where the length of the FO sentence is the parameter.

# Myhill-Nerode classes

- Consider the example of modulo 3 counting on path graphs again.

# Myhill-Nerode classes

- Consider the example of modulo 3 counting on path graphs again.
- $G_1$: length 5, $G_2$: length 8, $G_3$: arbitrary.
- Suppose $G_1 \cdot G_3$ has length 0 modulo 3. What about $G_2 \cdot G_3$?

# Myhill-Nerode classes

- Consider the example of modulo 3 counting on path graphs again.
- $G_1$: length 5, $G_2$: length 8, $G_3$: arbitrary.
- Suppose $G_1 \cdot G_3$ has length 0 modulo 3. What about $G_2 \cdot G_3$?
- $|G_2 \cdot G_3| \equiv |G_1 \cdot G_3| \mod 3$. $G_1$ and $G_2$ are "equivalent".

# Myhill-Nerode classes

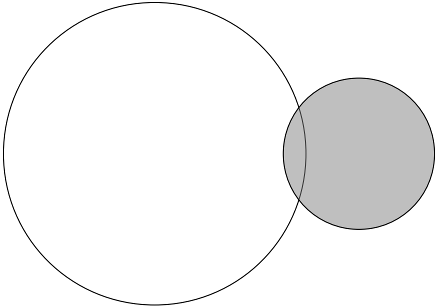
- Consider the example of modulo 3 counting on path graphs again.
- $G_1$: length 5, $G_2$: length 8, $G_3$: arbitrary.
- Suppose $G_1 \cdot G_3$ has length 0 modulo 3. What about $G_2 \cdot G_3$?
- $|G_2 \cdot G_3| \equiv |G_1 \cdot G_3| \mod 3$. $G_1$ and $G_2$ are "equivalent".
- There are 3 equivalence classes for this particular problem. They are called Myhill-Nerode classes.

# Application to Kernelization

[Bodlaender, Fomin, Lokshtanov, Penninkx, Saurabh, Thilikos]: Certain class of problems expressible in Counting MSO have polynomial kernels on graphs of bounded genus.

# Application to Kernelization

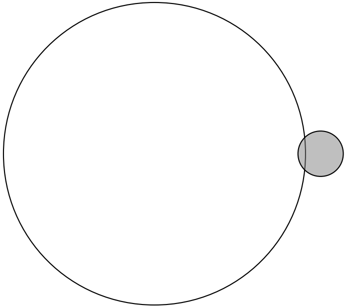
[Bodlaender, Fomin, Lokshtanov, Penninkx, Saurabh, Thilikos]: Certain class of problems expressible in Counting MSO have polynomial kernels on graphs of bounded genus.



In a big enough graph, there will always be a Protrusion.

# Application to Kernelization

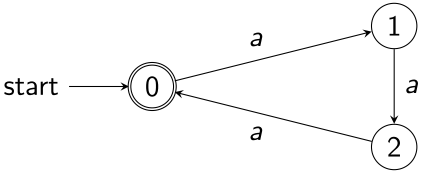
[Bodlaender, Fomin, Lokshtanov, Penninkx, Saurabh, Thilikos]: Certain class of problems expressible in Counting MSO have polynomial kernels on graphs of bounded genus.



In a big enough graph, there will always be a Protrusion.
Replace by a smallest one in the same Myhill-Nerode class.

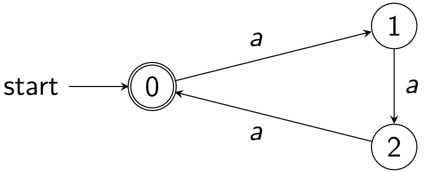
# Designing dynamic programming algorithms

- Myhill-Nerode classes have close relationship with states of a finite automaton. Example:



# Designing dynamic programming algorithms

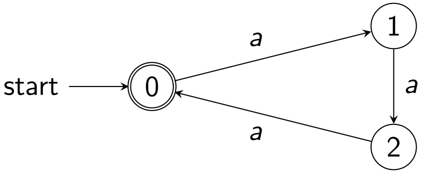
- Myhill-Nerode classes have close relationship with states of a finite automaton. Example:




- Studying the equivalence classes for individual problems can lead to good dynamic programming algorithms [Abrahamson, Fellows], [Ganian, Hliněný].

# Designing dynamic programming algorithms

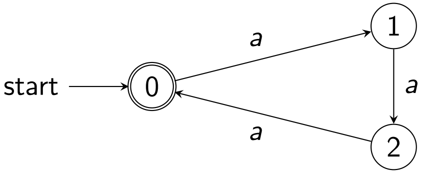
- Myhill-Nerode classes have close relationship with states of a finite automaton. Example:




- Studying the equivalence classes for individual problems can lead to good dynamic programming algorithms [Abrahamson, Fellows], [Ganian, Hliněný].
- [Courcelle, Durand]: Work around huge intermediate automata and compute transitions when required.

# Designing dynamic programming algorithms

- Myhill-Nerode classes have close relationship with states of a finite automaton. Example:




- Studying the equivalence classes for individual problems can lead to good dynamic programming algorithms [Abrahamson, Fellows], [Ganian, Hliněný].
- [Courcelle, Durand]: Work around huge intermediate automata and compute transitions when required.
- [Gottlob, Pichler, Wei]: Fragment of datalog that do not need further translations.

# Conclusion

- Courcelle's theorem is a powerful tool for proving Fixed Parameter Tractability results.
- Leads to many interesting questions.
- Overcoming problems in practical implementation: ongoing area of research.

# Conclusion

- Courcelle's theorem is a powerful tool for proving Fixed Parameter Tractability results.
- Leads to many interesting questions.
- Overcoming problems in practical implementation: ongoing area of research.

Thank you. Questions?

# References I


Karl R. Abrahamson and Michael R. Fellows.
Finite automata, bounded treewidth, and well-quasiordering.
In Neil Robertson and Paul D. Seymour, editors, *Graph Structure Theory*, pages 539–564. American Mathematical Society, 1991.

Stefan Arnborg, Jens Lagergren, and Detlef Seese.
Easy problems for tree-decomposable graphs.
*J. Algorithms*, 12:308–340, April 1991.

Hans L. Bodlaender, Fedor V. Fomin, Daniel Lokshtanov, Eelko Penninkx, Saket Saurabh, and Dimitrios M. Thilikos.
(Meta) kernelization.
In *FOCS*, pages 629–638, 2009.

# References II


Bruno Courcelle.
The monadic second-order logic of graphs I: Recognizable sets of finite graphs.
*Information and Computation*, 85:12–75, 1990.

Bruno Courcelle and Iréne Anne Durand.
Verifying monadic second-order graph properties with tree automata.
In *European Lisp Symposium*, 2010.

Anuj Dawar, Martin Grohe, and Stephan Kreutzer.
Locally excluding a minor.
In *LICS*, pages 270–279, 2007.

Jörg Flum and Martin Grohe.
Fixed-parameter tractability, definability, and model-checking.
*SIAM J. Comput.*, 31(1):113–145, 2001.

# References III


Jörg Flum and Martin Grohe.
*Parameterized Complexity Theory*.
Springer, 2006.
Chapters 10, 11 and 12.

Markus Frick and Martin Grohe.
Deciding first-order properties of locally tree-decomposable graphs.
In *ICALP*, pages 331–340, 1999.

Markus Frick and Martin Grohe.
The complexity of first-order and monadic second-order logic revisited.
In *LICS*, pages 215–224, 2002.

# References IV


Robert Ganian and Petr Hliněný.
On parse trees and myhill-nerode-type tools for handling graphs of bounded rank-width.
*Discrete Applied Mathematics*, 158(7):851–867, 2010.

Georg Gottlob, Reinhard Pichler, and Fang Wei.
Monadic datalog over finite structures of bounded treewidth.
*ACM Trans. Comput. Logic*, 12:3:1–3:48, November 2010.

Martin Grohe.
Logic, graphs and algorithms.
In Jörg Flum, Erich Grädel, and Thomas Wilke, editors, *Logic and Automata — History and Perspectives*. Amsterdam University Press, 2007.

# References V


Stephan Kreutzer.
Algorithmic meta-theorems.
http://web.comlab.ox.ac.uk/people/stephan.kreutzer/Publications/ar
survey.pdf.

Kamal Lodaya.
Monadic second-order logic of graphs defined by operations.
http://www.imsc.res.in/%7Ekamal/tut/msot.ps.gz.