Finite Model Theory
Lecture 10: Second Order Logic

Spring 2025

Announcements

- Homework 3 is posted and due on May 9
- Lecture topics on the Website are finally in a stable state
- Today: finish the discussion of SO
- Next week: recursion (datalog!), infinitary logics, pebble games.

Second Order Logic

Quick Review

Definition

Second Order Logic, SO, extends FO with *2nd order variables*, which range over relations.

- Add second order quantifiers: $\forall X, \exists Y$
- Add atoms of the form $X(u, v, w)$

Examples

Connectivity:

$$\forall U\,(\exists x \exists y (U(x) \land \neg U(y)) \rightarrow \exists u \exists v (E(u,v) \land U(u) \land \neg U(v)))$$

3-Colorability:

$$\begin{aligned}
\exists R \exists B \exists G \forall x (R(x) \lor B(x) \lor G(x))& \\
\land \forall x \forall y (E(x,y) \rightarrow \neg (R(x) \land R(y)))& \\
\land \forall x \forall y (E(x,y) \rightarrow \neg (G(x) \land G(y)))& \\
\land \forall x \forall y (E(x,y) \rightarrow \neg (B(x) \land B(y)))&
\end{aligned}$$

Review: Fragments of SO

Existential SO: ESO or ∃SO. Recall: captures NP

Monadic Second Order Logic, MSO

Existential Monadic SO, ∃MSO

Review: MSO on Words

Fix an alphabet Σ, e.g. $\Sigma = \{a, b, c\}$.
A word $w \in \Sigma^*$ encoded as a structure over $\sigma = (<, P_a, P_b, P_c)$.

Review: MSO on Words

Fix an alphabet Σ, e.g. $\Sigma = \{a, b, c\}$.
A word $w \in \Sigma^*$ encoded as a structure over $\sigma = (<, P_a, P_b, P_c)$.

Example: *aabaca* represented by $([6], <, P_a, P_b, P_c)$, $P_a = \{1, 2, 4, 6\}$, etc.

Review: MSO on Words

Fix an alphabet Σ, e.g. $\Sigma = \{a, b, c\}$.
A word $w \in \Sigma^*$ encoded as a structure over $\sigma = (<, P_a, P_b, P_c)$.

Example: *aabaca* represented by $([6], <, P_a, P_b, P_c)$, $P_a = \{1, 2, 4, 6\}$, etc.

Assume $\Sigma = \{a, b\}$: $\exists x P_a(x)$ $(a|b)^*.a.(a|b)^*$

Review: MSO on Words

Fix an alphabet Σ, e.g. $\Sigma = \{a, b, c\}$.
A word $w \in \Sigma^*$ encoded as a structure over $\sigma = (<, P_a, P_b, P_c)$.

Example: *aabaca* represented by $([6], <, P_a, P_b, P_c)$, $P_a = \{1, 2, 4, 6\}$, etc.

Assume $\Sigma = \{a, b\}$: $\exists x P_a(x)$ $(a|b)^*.a.(a|b)^*$

$\forall x \forall y((x < y \wedge P_a(y)) \Rightarrow P_a(x))$ $a^* b^*$

Review: MSO on Words

Fix an alphabet Σ, e.g. $\Sigma = \{a, b, c\}$.
A word $w \in \Sigma^*$ encoded as a structure over $\sigma = (<, P_a, P_b, P_c)$.

Example: *aabaca* represented by $([6], <, P_a, P_b, P_c)$, $P_a = \{1, 2, 4, 6\}$, etc.

Assume $\Sigma = \{a, b\}$: $\exists x P_a(x)$ $(a|b)^*.a.(a|b)^*$

$\forall x \forall y((x < y \wedge P_a(y)) \Rightarrow P_a(x))$ $a^* b^*$

$\forall x \forall y(x < y \wedge P_a(x) \wedge P_a(y) \Rightarrow \exists z(x < z \wedge z < y \wedge P_b(z)))$
$b^*.(a.b^+)^*(a|\varepsilon)$

Review: MSO on Words

Fix an alphabet Σ, e.g. $\Sigma = \{a, b, c\}$.
A word $w \in \Sigma^*$ encoded as a structure over $\sigma = (<, P_a, P_b, P_c)$.

Example: *aabaca* represented by $([6], <, P_a, P_b, P_c)$, $P_a = \{1, 2, 4, 6\}$, etc.

Assume $\Sigma = \{a, b\}$: $\exists x P_a(x)$ $(a|b)^*.a.(a|b)^*$

$\forall x \forall y((x < y \wedge P_a(y)) \Rightarrow P_a(x))$ $a^* b^*$

$\forall x \forall y(x < y \wedge P_a(x) \wedge P_a(y) \Rightarrow \exists z(x < z \wedge z < y \wedge P_b(z)))$
$b^*.(a.b^+)^*(a|\varepsilon)$

$$\forall x P_a(x) \wedge \exists X(X(\overline{\min}) \wedge \neg X(\overline{\max}) \wedge \\ \forall u \forall v(\mathrm{succ}(u, v) \Rightarrow (X(u) \wedge \neg X(v)) \vee (\neg X(u) \wedge X(v))))$$
$(a.a)^*$

Review: Büchi's Theorem

Theorem

MSO on strings captures regular languages

Proof

Part 1: Automaton with n states ⇒ MSO sentence $\varphi = \exists S_1 \cdots \exists S_n (\cdots)$

Part 2: Sub-formulas of φ to automaton over extended vocabulary.

Discussion

- $\exists$SO captures NP; $\exists SO \neq \forall SO$ iff $NP \neq coNP$
- MSO over words: linear time; expression complexity: non-elementary[1]
- Over words: $\exists MSO = MSO = \forall MSO$
- Courcelle's theorem: MSO over structures of bounded treewidth is in linear time.
- $\exists MSO \neq \forall MSO$ (today)

[1] Tower of exponentials of unbounded height.

FO on Words

FO on Words

FO cannot express $(a.a)^*$ WHY??

Will prove that FO captures precisely the star-free languages

Star-Free Languages

Fix an alphabet Σ. Regular expressions are:

$$E ::= \emptyset \mid \varepsilon \mid a \in \Sigma \mid E \cup E \mid E.E \mid C(E) \mid E^*$$

where $C(E)$ means "complement".

Star-Free Languages

Fix an alphabet Σ. Regular expressions are:

$$E ::= \varnothing \mid \varepsilon \mid a \in \Sigma \mid E \cup E \mid E.E \mid C(E) \mid E^*$$

where $C(E)$ means "complement".

E is called *star-free* if it is equivalent to an expression without $*$.

Star-Free Languages

Fix an alphabet Σ. Regular expressions are:

$$E ::= \varnothing \mid \varepsilon \mid a \in \Sigma \mid E \cup E \mid E.E \mid C(E) \mid E^*$$

where $C(E)$ means "complement".

E is called *star-free* if it is equivalent to an expression without $*$.
$\Sigma = \{a, b\}$, which of the expressions below are star-free?

- Σ^*
- b^*
- $(a.b)^*$
- $(a.a)^*$

Star-Free Languages

Fix an alphabet Σ. Regular expressions are:

$$E ::= \varnothing \mid \varepsilon \mid a \in \Sigma \mid E \cup E \mid E.E \mid C(E) \mid E^*$$

where $C(E)$ means "complement".

E is called *star-free* if it is equivalent to an expression without $*$.
$\Sigma = \{a, b\}$, which of the expressions below are star-free?

- Σ^* $\qquad C(\varnothing)$
- b^*
- $(a.b)^*$
- $(a.a)^*$

Star-Free Languages

Fix an alphabet Σ. Regular expressions are:

$$E ::= \varnothing \mid \varepsilon \mid a \in \Sigma \mid E \cup E \mid E.E \mid C(E) \mid E^*$$

where $C(E)$ means "complement".

E is called *star-free* if it is equivalent to an expression without $*$.
$\Sigma = \{a, b\}$, which of the expressions below are star-free?

- Σ^* $\qquad C(\varnothing)$
- b^* $\qquad C(\Sigma^*.a.\Sigma^*)$
- $(a.b)^*$
- $(a.a)^*$

Star-Free Languages

Fix an alphabet Σ. Regular expressions are:

$$E ::= \emptyset \mid \varepsilon \mid a \in \Sigma \mid E \cup E \mid E.E \mid C(E) \mid E^*$$

where $C(E)$ means "complement".

E is called *star-free* if it is equivalent to an expression without $*$.
$\Sigma = \{a, b\}$, which of the expressions below are star-free?

- Σ^* — $C(\emptyset)$
- b^* — $C(\Sigma^*.a.\Sigma^*)$
- $(a.b)^*$ — $C(\Sigma^*.a.a.\Sigma^* \cup \Sigma^*.b.b.\Sigma^* \cup b.\Sigma^* \cup \Sigma^*.a)$
- $(a.a)^*$

Star-Free Languages

Fix an alphabet Σ. Regular expressions are:

$$E ::= \varnothing \mid \varepsilon \mid a \in \Sigma \mid E \cup E \mid E.E \mid C(E) \mid E^*$$

where $C(E)$ means "complement".

E is called *star-free* if it is equivalent to an expression without $*$.
$\Sigma = \{a, b\}$, which of the expressions below are star-free?

- Σ^* — $C(\varnothing)$
- b^* — $C(\Sigma^*.a.\Sigma^*)$
- $(a.b)^*$ — $C(\Sigma^*.a.a.\Sigma^* \cup \Sigma^*.b.b.\Sigma^* \cup b.\Sigma^* \cup \Sigma^*.a)$
- $(a.a)^*$ — NOT star free! Let's prove it.

FO on Words

Theorem

FO over strings captures precisely the star-free regular languages.

Consequence: $(a.a)^*$ is not star-free.

Otherwise: express it in FO, use it for EVEN of $(L_n, <)$, contradiction.

Proof Part 1: Star Free Regular Exprssions to FO

First, convert E to an FO formula $\varphi_E(x, y)$ stating

"the substring $w[x : y)$ is in $L(E)$"

Proof Part 1: Star Free Regular Exprssions to FO

First, convert E to an FO formula $\varphi_E(x, y)$ stating

"the substring $w[x : y)$ is in $L(E)$"

- $\varnothing$ becomes $\mathtt{FALSE}$
- ε becomes $x = y$
- a becomes $P_a(x) \wedge \mathtt{succ}(x, y)$

Proof Part 1: Star Free Regular Exprssions to FO

First, convert E to an FO formula $\varphi_E(x, y)$ stating

"the substring $w[x : y)$ is in $L(E)$"

- $\varnothing$ becomes FALSE
- ε becomes $x = y$
- a becomes $P_a(x) \wedge \texttt{succ}(x, y)$
- $E_1 \cup E_2$ becomes $\varphi_{E_1}(x, y) \vee \varphi_{E_2}(x, y)$
- $E_1.E_2$ becomes $\exists z(\varphi_{E_1}(x, z) \wedge \varphi_{E_2}(z, y))$.

Proof Part 1: Star Free Regular Exprssions to FO

First, convert E to an FO formula $\varphi_E(x, y)$ stating

"the substring $w[x : y)$ is in $L(E)$"

- $\varnothing$ becomes $\texttt{FALSE}$
- ε becomes $x = y$
- a becomes $P_a(x) \wedge \texttt{succ}(x, y)$
- $E_1 \cup E_2$ becomes $\varphi_{E_1}(x, y) \vee \varphi_{E_2}(x, y)$
- $E_1.E_2$ becomes $\exists z(\varphi_{E_1}(x, z) \wedge \varphi_{E_2}(z, y))$.
- $C(E)$ becomes $\neg\varphi_E(x, y)$

Proof Part 1: Star Free Regular Exprssions to FO

First, convert E to an FO formula $\varphi_E(x, y)$ stating

"the substring $w[x : y)$ is in $L(E)$"

- $\emptyset$ becomes FALSE
- ε becomes $x = y$
- a becomes $P_a(x) \wedge \texttt{succ}(x, y)$
- $E_1 \cup E_2$ becomes $\varphi_{E_1}(x, y) \vee \varphi_{E_2}(x, y)$
- $E_1.E_2$ becomes $\exists z(\varphi_{E_1}(x, z) \wedge \varphi_{E_2}(z, y))$.
- $C(E)$ becomes $\neg\varphi_E(x, y)$

Finally, complete the translation with:

$$\boxed{\exists x \exists y(\text{isMin}(x) \wedge \text{isMax}(y) \wedge E(x, y))}$$

Proof: Part 2

For each sentence φ, construct regular expression E_φ s.t. $w \vDash \varphi$ iff $w \in L(E_\varphi)$

Proof: Part 2

For each sentence φ, construct regular expression E_φ s.t. $w \vDash \varphi$ iff $w \in L(E_\varphi)$

We showed to translate an MSO formula φ to an automaton over an extended alphabet $\overline{\Sigma}$

Can we use the same proof but instead of automaton construct a regular expression?

Proof: Part 2

For each sentence φ, construct regular expression E_φ s.t. $w \vDash \varphi$ iff $w \in L(E_\varphi)$

We showed to translate an MSO formula φ to an automaton over an extended alphabet $\overline{\Sigma}$

Can we use the same proof but instead of automaton construct a regular expression?

Let's take a closer look at that proof and see what exactly fails.

Review: MSO to Automata

$\Sigma = \{a, b, c\}$ $\varphi = \exists x P_a(x) \wedge \forall y (x < y \rightarrow P_b(y))$ Meaning???

Review: MSO to Automata

$\Sigma = \{a, b, c\}$ $\varphi = \exists x P_a(x) \wedge \forall y (x < y \rightarrow P_b(y))$ $(a|b|c)^*.a.b^*$

Review: MSO to Automata

$\Sigma = \{a, b, c\}$ $\qquad \varphi = \exists x P_a(x) \wedge \forall y (x < y \rightarrow P_b(y))$ $\qquad (a|b|c)^*.a.b^*$

$$\boxed{\exists x\, (P_a(x) \wedge \neg \exists y (x < y \wedge \neg P_b(y)))}$$

Review: MSO to Automata

$\Sigma = \{a, b, c\}$ $\varphi = \exists x P_a(x) \wedge \forall y (x < y \rightarrow P_b(y))$ $(a|b|c)^*.a.b^*$

$$\boxed{\exists x \left(P_a(x) \wedge \neg \exists y (x < y \wedge \neg P_b(y))\right)}$$

Extended alphabets: $\overline{\Sigma} \stackrel{\text{def}}{=} \{a, a^x, b, b^x, c, c^x\}$ $\overline{\overline{\Sigma}} = \{a, a^x, a^y, a^{xy}, \ldots\}$

Review: MSO to Automata

$\Sigma = \{a, b, c\}$ $\quad \varphi = \exists x P_a(x) \wedge \forall y(x < y \rightarrow P_b(y))$ $\quad (a|b|c)^*.a.b^*$

$$\boxed{\exists x\,(P_a(x) \wedge \neg\exists y(x < y \wedge \neg P_b(y)))}$$

At least one a^x: $\overline{\Sigma}^*.a^x.\overline{\Sigma}^*$

Review: MSO to Automata

$\Sigma = \{a, b, c\}$ $\varphi = \exists x P_a(x) \wedge \forall y(x < y \to P_b(y))$ $(a|b|c)^*.a.b^*$

$$\boxed{\exists x \, (P_a(x) \wedge \neg \exists y(x < y \wedge \neg P_b(y)))}$$

At least one a^x: $\overline{\Sigma}^*.a^x.\overline{\Sigma}^*$

x followed y: $\overline{\overline{\Sigma}}^*.(a^x|a^{xy}|b^x|\cdots).\overline{\overline{\Sigma}}^*.(a^y|a^{xy}|\cdots).\overline{\overline{\Sigma}}^*$

Review: MSO to Automata

$\Sigma = \{a, b, c\}$ $\varphi = \exists x P_a(x) \wedge \forall y(x < y \to P_b(y))$ $(a|b|c)^*.a.b^*$

$\boxed{\exists x\,(P_a(x) \wedge \neg\exists y(x < y \wedge \neg P_b(y)))}$

At least one a^x: $\overline{\Sigma}^*.a^x.\overline{\Sigma}^*$

x followed y: $\overline{\overline{\Sigma}}^*.(a^x|a^{xy}|b^x|\cdots).\overline{\overline{\Sigma}}^*.(a^y|a^{xy}|\cdots).\overline{\overline{\Sigma}}^*$

At least one y that is $\neq b$: $\overline{\overline{\Sigma}}^*.(a^y|a^{xy}|\ldots|c^{xy}).\overline{\overline{\Sigma}}^*$

Review: MSO to Automata

$\Sigma = \{a, b, c\}$ $\varphi = \exists x P_a(x) \wedge \forall y (x < y \rightarrow P_b(y))$ $(a|b|c)^*.a.b^*$

$$\boxed{\exists x \, (P_a(x) \wedge \neg \exists y (x < y \wedge \neg P_b(y)))}$$

At least one a^x: $\overline{\Sigma}^*.a^x.\overline{\Sigma}^*$

x followed y: $\overline{\overline{\Sigma}}^*.(a^x|a^{xy}|b^x|\cdots).\overline{\overline{\Sigma}}^*.(a^y|a^{xy}|\cdots).\overline{\overline{\Sigma}}^*$

At least one y that is $\neq b$: $\overline{\overline{\Sigma}}^*.(a^y|a^{xy}|\ldots|c^{xy}).\overline{\overline{\Sigma}}^*$

Review: MSO to Automata

$\Sigma = \{a, b, c\}$ $\varphi = \exists x P_a(x) \wedge \forall y(x < y \rightarrow P_b(y))$ $(a|b|c)^*.a.b^*$

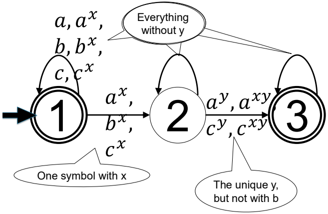
$\boxed{\exists x\,(P_a(x) \wedge \neg\exists y(x < y \wedge \neg P_b(y)))}$

At least one a^x: $\overline{\Sigma}^*.a^x.\overline{\Sigma}^*$

x followed y: $\overline{\overline{\Sigma}}^*.(a^x|a^{xy}|b^x|\cdots).\overline{\overline{\Sigma}}^*.(a^y|a^{xy}|\cdots).\overline{\overline{\Sigma}}^*$

At least one y that is $\neq b$: $\overline{\overline{\Sigma}}^*.(a^y|a^{xy}|\ldots|c^{xy}).\overline{\overline{\Sigma}}^*$

Intersect, enforce unique y:



Review: MSO to Automata

$\Sigma = \{a, b, c\}$ $\varphi = \exists x P_a(x) \wedge \forall y(x < y \to P_b(y))$ $(a|b|c)^*.a.b^*$

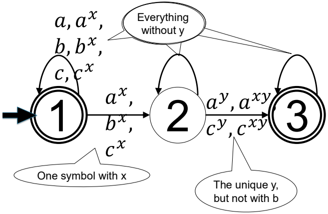
$$\exists x\,(P_a(x) \wedge \neg\exists y(x < y \wedge \neg P_b(y)))$$

At least one a^x: $\overline{\Sigma}^*.a^x.\overline{\Sigma}^*$

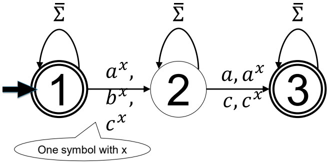
x followed y: $\overline{\overline{\Sigma}}^*.(a^x|a^{xy}|b^x|\cdots).\overline{\overline{\Sigma}}^*.(a^y|a^{xy}|\cdots).\overline{\overline{\Sigma}}^*$

At least one y that is $\neq b$: $\overline{\overline{\Sigma}}^*.(a^y|a^{xy}|\ldots|c^{xy}).\overline{\overline{\Sigma}}^*$

Intersect, enforce unique y:



Remove labels y: $\overline{\overline{\Sigma}} \to \overline{\Sigma}$



Some x followed by a|c: $\overline{\Sigma}^*.(a^x|b^x|c^x).\overline{\Sigma}^*.(a|a^x|c|c^x).\overline{\Sigma}^*$

Review: MSO to Automata

$\Sigma = \{a, b, c\}$ $\quad \varphi = \exists x P_a(x) \wedge \forall y(x < y \rightarrow P_b(y))$ $\quad (a|b|c)^*.a.b^*$

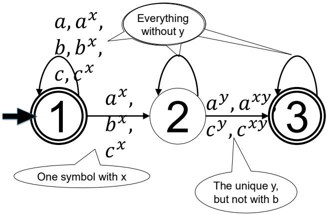
$$\exists x\,(P_a(x) \wedge \neg\exists y(x < y \wedge \neg P_b(y)))$$

At least one a^x: $\overline{\Sigma}^*.a^x.\overline{\Sigma}^*$

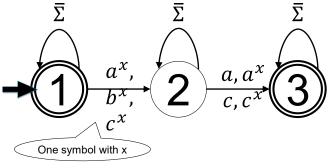
x followed y: $\overline{\overline{\Sigma}}^*.(a^x|a^{xy}|b^x|\cdots).\overline{\overline{\Sigma}}^*.(a^y|a^{xy}|\cdots).\overline{\overline{\Sigma}}^*$

At least one y that is $\neq b$: $\overline{\overline{\Sigma}}^*.(a^y|a^{xy}|\ldots|c^{xy}).\overline{\overline{\Sigma}}^*$

Intersect, enforce unique y:



Remove labels y: $\overline{\overline{\Sigma}} \rightarrow \overline{\Sigma}$



Some x followed by a|c: $\overline{\Sigma}^*.(a^x|b^x|c^x).\overline{\Sigma}^*.(a|a^x|c|c^x).\overline{\Sigma}^*$

Negate: after every x is b^*.

Review: MSO to Automata

$\Sigma = \{a, b, c\}$ $\qquad \varphi = \exists x P_a(x) \wedge \forall y (x < y \rightarrow P_b(y))$ $\qquad (a|b|c)^*.a.b^*$

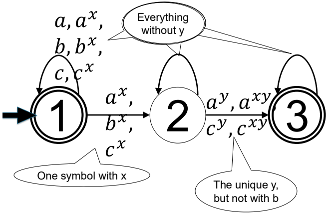
$$\boxed{\exists x \,(P_a(x) \wedge \neg \exists y (x < y \wedge \neg P_b(y)))}$$

At least one a^x: $\overline{\Sigma}^*.a^x.\overline{\Sigma}^*$

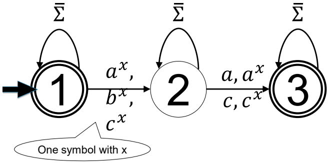
x followed y: $\overline{\overline{\Sigma}}^*.(a^x|a^{xy}|b^x|\cdots).\overline{\overline{\Sigma}}^*.(a^y|a^{xy}|\cdots).\overline{\overline{\Sigma}}^*$

At least one y that is $\neq b$: $\overline{\overline{\Sigma}}^*.(a^y|a^{xy}|\ldots|c^{xy}).\overline{\overline{\Sigma}}^*$

Intersect, enforce unique y:



Remove labels y: $\overline{\overline{\Sigma}} \rightarrow \overline{\Sigma}$



Some x followed by a|c: $\overline{\Sigma}^*.(a^x|b^x|c^x).\overline{\Sigma}^*.(a|a^x|c|c^x).\overline{\Sigma}^*$

Negate: after every x is b^*. $\qquad$ Intersect $P_a(x)$. $\qquad$ Drop x.

Important Takeaway

Fix two alphabets and a function $f : \overline{\Sigma} \to \Sigma$.

If A is an automaton, then:

$$\boxed{w \in L(f(A)) \text{ iff } \exists u(u \in L(A) \wedge f(u) = w)} \text{ i.e. } L(f(A)) = f(L(A))$$

Important Takeaway

Fix two alphabets and a function $f : \overline{\Sigma} \to \Sigma$.

If A is an automaton, then:

$\boxed{w \in L(f(A)) \text{ iff } \exists u(u \in L(A) \wedge f(u) = w)}$ i.e. $L(f(A)) = f(L(A))$

$\overline{\Sigma} = \{a, b\}$
$\Sigma = \{c\}$
$f(a) = f(b) = c.$

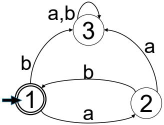
Important Takeaway

Fix two alphabets and a function $f : \overline{\Sigma} \to \Sigma$.

If A is an automaton, then:

$w \in L(f(A))$ iff $\exists u(u \in L(A) \wedge f(u) = w)$ i.e. $L(f(A)) = f(L(A))$

$(a.b)^*$:



$\overline{\Sigma} = \{a, b\}$
$\Sigma = \{c\}$
$f(a) = f(b) = c$.

Important Takeaway

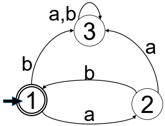
Fix two alphabets and a function $f : \overline{\Sigma} \to \Sigma$.

If A is an automaton, then:

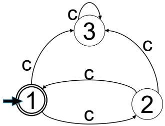
$\boxed{w \in L(f(A)) \text{ iff } \exists u(u \in L(A) \wedge f(u) = w)}$ i.e. $L(f(A)) = f(L(A))$

$\overline{\Sigma} = \{a, b\}$
$\Sigma = \{c\}$
$f(a) = f(b) = c$.

$(a.b)^*$:



$(c.c)^*$:



Important Takeaway

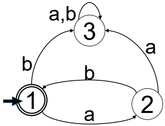
Fix two alphabets and a function $f : \overline{\Sigma} \to \Sigma$.

If A is an automaton, then:

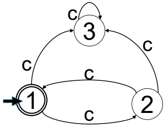
$$\boxed{w \in L(f(A)) \text{ iff } \exists u (u \in L(A) \wedge f(u) = w)} \text{ i.e. } L(f(A)) = f(L(A))$$

$\overline{\Sigma} = \{a, b\}$
$\Sigma = \{c\}$
$f(a) = f(b) = c.$

$(a.b)^*$:



$(c.c)^*$:



This fails for regular expressions with complement: $L(f(E)) \neq f(L(E))$

Important Takeaway

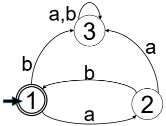
Fix two alphabets and a function $f : \overline{\Sigma} \to \Sigma$.

If A is an automaton, then:

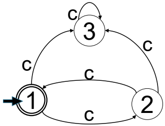
$w \in L(f(A))$ iff $\exists u(u \in L(A) \wedge f(u) = w)$ i.e. $L(f(A)) = f(L(A))$

$\overline{\Sigma} = \{a, b\}$
$\Sigma = \{c\}$
$f(a) = f(b) = c$.

$(a.b)^*$:



$(c.c)^*$:



This fails for regular expressions with complement: $L(f(E)) \neq f(L(E))$

$E = C((a|b)^*.a.a.(a|b)^* \cup (a|b)^*.b.b.(a|b)^* \cup b.(a|b)^* \cup (a|b)^*.a)$

$L(E) = (a.a)^*$

Important Takeaway

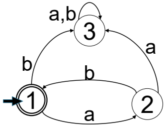
Fix two alphabets and a function $f : \overline{\Sigma} \to \Sigma$.

If A is an automaton, then:

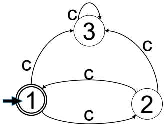
$$\boxed{w \in L(f(A)) \text{ iff } \exists u (u \in L(A) \wedge f(u) = w)} \text{ i.e. } L(f(A)) = f(L(A))$$

$\overline{\Sigma} = \{a, b\}$
$\Sigma = \{c\}$
$f(a) = f(b) = c.$

$(a.b)^*$:



$(c.c)^*$:



This fails for regular expressions with complement: $L(f(E)) \neq f(L(E))$

$$E = C((a|b)^*.a.a.(a|b)^* \cup (a|b)^*.b.b.(a|b)^* \cup b.(a|b)^* \cup (a|b)^*.a)$$

$L(E) = (a.a)^*$

$$f(E) = C(c^*.c.c.c^* \cup c^*.c.c.c^* \cup c.c^* \cup c^*.c) = \varepsilon$$

$L(f(E)) = \varepsilon.$

Discussion

We need a inductive proof that uses only sentences, no formulas.

Will do induction on the quantifier depth k.

And we will use $\mathrm{FO}[k]$ types.

Review: FO[k] types

FO[k] is FO where we restrict formulas to quantifer rank $\leq k$.

We defined $\text{tp}_{k,m}$ where m = number of free variables.

Today: we only need $m = 0$.

Definition

Let $\boldsymbol{A}$ be a structure. Its FO[k]-type is: $\text{tp}_k(\boldsymbol{A}) = \{\varphi \in \text{FO}[k] \mid \boldsymbol{A} \models \varphi\}$

Every FO[k]-type is a finite set of sentences: their $\bigwedge$ is a single sentence:

$$\tau = \text{tp}_k(\boldsymbol{A})$$

Review: EF-games on Linear Order

Let $L_m = ([m], <)$. Denote:[2]

$$L_m^{<a} \stackrel{\text{def}}{=} \{x \in L_m \mid x < a\} \qquad L_m^{>a} \stackrel{\text{def}}{=} \{x \in L_m \mid x > a\}$$

[2] Isomorphic to linear orders: $\boldsymbol{L}_m^{<a} \simeq \boldsymbol{L}_{a-1}$, $\boldsymbol{L}_m^{>a} \simeq L_{m-a}$.

Review: EF-games on Linear Order

Let $L_m = ([m], <)$. Denote:[2]

$$L_m^{<a} \overset{\text{def}}{=} \{x \in L_m \mid x < a\} \qquad L_m^{>a} \overset{\text{def}}{=} \{x \in L_m \mid x > a\}$$

Lemma

If $L_m^{<a} \sim_k L_n^{<b}$ and $L_m^{>a} \sim_k L_n^{>b}$, then $L_m \sim_k L_n$.

[2] Isomorphic to linear orders: $\boldsymbol{L}_m^{<a} \simeq \boldsymbol{L}_{a-1}$, $\boldsymbol{L}_m^{>a} \simeq L_{m-a}$.

EF Games on Words

$\sigma = (<, \overline{\min}, P_{a_1}, P_{a_2}, \ldots)$ $\sigma_1 = \sigma \cup \{c\}$ (add a constant c)

Lemma

If $w^{<p} \sim_k u^{<q}$ and $w^{\geq p} \sim_k u^{\geq q}$ then $(w, p) \sim_k (u, q)$ (i.e. $c^w = p, c^u = q$)

EF Games on Words

$\sigma = (<, \overline{\min}, P_{a_1}, P_{a_2}, \ldots)$ $\qquad$ $\sigma_1 = \sigma \cup \{c\}$ (add a constant c)

Lemma

If $w^{<p} \sim_k u^{<q}$ *and* $w^{\geq p} \sim_k u^{\geq q}$ *then* $(w, p) \sim_k (u, q)$ $\qquad$ *(i.e.* $c^w = p, c^u = q$*)*

$$w = \begin{array}{|c|c|c|c|c||c|c|c|c|} \multicolumn{1}{c}{1} & \multicolumn{4}{c}{} & \multicolumn{1}{c}{p} & \multicolumn{3}{c}{} \\ \hline a_1 & a_2 & a_1 & \ldots & a_1 & a_3 & a_1 & \ldots & \ldots \\ \hline \end{array}$$

$$u = \begin{array}{|c|c|c|c|c|c||c|c|c|c|c||} \multicolumn{1}{c}{1} & \multicolumn{5}{c}{} & \multicolumn{1}{c}{q} & \multicolumn{4}{c}{} \\ \hline a_1 & a_2 & a_2 & a_1 & \ldots & a_2 & a_3 & a_1 & a_1 & a_2 & \ldots \\ \hline \end{array}$$

EF Games on Words

$\sigma = (<, \overline{\min}, P_{a_1}, P_{a_2}, \ldots)$ $\sigma_1 = \sigma \cup \{c\}$ (add a constant c)

Lemma

If $w^{<p} \sim_k u^{<q}$ and $w^{\geq p} \sim_k u^{\geq q}$ then $(w, p) \sim_k (u, q)$ (i.e. $c^w = p, c^u = q$)

	1					p			
$w =$	a_1	a_2	a_1	$\ldots$	a_1	a_3	a_1	$\ldots$	$\ldots$

	1						q				
$u =$	a_1	a_2	a_2	a_1	$\ldots$	a_2	a_3	a_1	a_1	a_2	$\ldots$

Proof in class.

It is necessary to have $\overline{\min}$ in the vocabulary. why???

Proof: Part 2

For every sentence φ we construct a star-free regular expression E_φ.

Proof: Part 2

For every sentence φ we construct a star-free regular expression E_φ.

Note: no free variables. Instead we prove by induction on $k = qr(\varphi)$.

For each k we also do induction on the structure of φ:

- If $\varphi = \varphi_1 \vee \varphi_2$ then $E_\varphi = E_{\varphi_1} | E_{\varphi_2}$
- If $\varphi = \neg\varphi_1$ then $E_\varphi = C(E_{\varphi_1})$.

Proof: Part 2

For every sentence φ we construct a star-free regular expression E_φ.

$qr(\varphi) = 0$.

- If $\varphi = P_a(\overline{\text{min}})$
- If $\varphi = (\overline{\text{min}} < \overline{\text{min}})$

Proof: Part 2

For every sentence φ we construct a star-free regular expression E_φ.

$qr(\varphi) = 0$.

- If $\varphi = P_a(\overline{\min})$ then $E_\varphi = a$
- If $\varphi = (\overline{\min} < \overline{\min})$

Proof: Part 2

For every sentence φ we construct a star-free regular expression E_φ.

$qr(\varphi) = 0$.

- If $\varphi = P_a(\overline{\min})$ then $E_\varphi = a$
- If $\varphi = (\overline{\min} < \overline{\min})$ then $E_\varphi = \varnothing$

Proof: Part 2

For every sentence φ we construct a star-free regular expression E_φ.

$qr(\varphi) = k + 1$. Assume w.l.o.g. $\varphi = \exists x \psi(x)$

$$S \stackrel{\text{def}}{=} \{(\text{tp}_k(u^{<q}), \text{tp}_k(u^{\geq q})) \mid u \in \Sigma^*, q \in \mathbb{N}, u \vDash \psi(q), = \sigma\}$$

Proof: Part 2

For every sentence φ we construct a star-free regular expression E_φ.

$qr(\varphi) = k + 1$. Assume w.l.o.g. $\varphi = \exists x \psi(x)$

$$S \stackrel{\text{def}}{=} \{(\text{tp}_k(u^{<q}), \text{tp}_k(u^{\geq q})) \mid u \in \Sigma^*, q \in \mathbb{N}, u \vDash \psi(q), = \sigma\}$$

Claim: for every $w \in \Sigma^*$:

$$\boxed{w \vDash \varphi \text{ iff } \exists p \in \mathbb{N}, (\text{tp}_k(w^{<p}), \text{tp}_k(w^{\geq p})) \in S}$$

Proof: Part 2

For every sentence φ we construct a star-free regular expression E_φ.

$qr(\varphi) = k + 1$. Assume w.l.o.g. $\varphi = \exists x \psi(x)$

$$S \stackrel{\text{def}}{=} \{(\text{tp}_k(u^{<q}), \text{tp}_k(u^{\geq q})) \mid u \in \Sigma^*, q \in \mathbb{N}, u \models \psi(q), = \sigma\}$$

Claim: for every $w \in \Sigma^*$:

$$\boxed{w \models \varphi \text{ iff } \exists p \in \mathbb{N}, (\text{tp}_k(w^{<p}), \text{tp}_k(w^{\geq p})) \in S}$$

$S = \{(\sigma_1, \tau_1), \dots, (\sigma_n, \tau_n)\}$ (finite); claim implies $E_\varphi = E_{\sigma_1}.E_{\tau_1} | E_{\sigma_2}.E_{\tau_2} | \cdots$

Proof: Part 2

For every sentence φ we construct a star-free regular expression E_φ.

$qr(\varphi) = k + 1$. Assume w.l.o.g. $\varphi = \exists x \psi(x)$

$$S \stackrel{\text{def}}{=} \{(\text{tp}_k(u^{<q}), \text{tp}_k(u^{\geq q})) \mid u \in \Sigma^*, q \in \mathbb{N}, u \vDash \psi(q), = \sigma\}$$

Claim: for every $w \in \Sigma^*$:

$$\boxed{w \vDash \varphi \text{ iff } \exists p \in \mathbb{N}, (\text{tp}_k(w^{<p}), \text{tp}_k(w^{\geq p})) \in S}$$

If $w \vDash \varphi$ then $\exists q$ s.t. $w \vDash \psi(q)$ and $(\text{tp}_k(w^{<p}), \text{tp}_k(w^{\geq p})) \in S$

Proof: Part 2

For every sentence φ we construct a star-free regular expression E_φ.

$qr(\varphi) = k+1$. Assume w.l.o.g. $\varphi = \exists x \psi(x)$

$$S \stackrel{\text{def}}{=} \{(\text{tp}_k(u^{<q}), \text{tp}_k(u^{\geq q})) \mid u \in \Sigma^*, q \in \mathbb{N}, u \vDash \psi(q), = \sigma\}$$

Claim: for every $w \in \Sigma^*$:

$$\boxed{w \vDash \varphi \text{ iff } \exists p \in \mathbb{N}, (\text{tp}_k(w^{<p}), \text{tp}_k(w^{\geq p})) \in S}$$

If $(\text{tp}_k(w^{<p}), \text{tp}_k(w^{\geq p})) \in S$ then $\exists u \in \Sigma^*, q \in \mathbb{N}, u \vDash \psi(q)$:

$$\boxed{\text{tp}_k(w^{<p}) = \text{tp}_k(u^{<q})} \text{ and } \boxed{\text{tp}_k(w^{\geq p}) = \text{tp}_k(u^{\geq q})}$$

This implies $\text{tp}_k(w) = \text{tp}_k(u)$
Then $w \vDash \psi(p)$, and therefore $w \vDash \exists x \psi(x)$.

This completes the proof.

Discussion

- The language $(a.a)^*$ is not star-free
  because it checks if a^* has EVEN length; is not in FO
- Satisfiability of for MSO on strings is decidable.
- The data complexity for MSO on strings is in linear time
  In general, the data complexity of MSO is in NP; can be NP-conplete.
- On strings: $\exists$MSO = $\forall$MSO = MSO

$\exists\text{MSO} \neq \forall\text{MSO}$

Problem Setting

$\exists$SO $\neq$ $\forall$SO is as difficult as $NP \neq coNP$.

Surprisingly, Fagin proved $\exists$MSO $\neq$ $\forall$MSO.

We will prove this result next.

Fagin's Theorem

Theorem

(1) CONNECTIVITY is expressible in $\forall$MSO
(2) CONNECTIVITY is not expressible in $\exists$MSO

Proof We have seen (1):

$$\forall U\,(\exists x \exists y (U(x) \wedge \neg U(y)) \rightarrow \exists u \exists v (E(u,v) \wedge U(u) \wedge \neg U(v)))$$

For (2), we will use games for $\exists$MSO.

Review: Hanf's Lemma

d-neighborhood of $a \in A$:

$$N(a,d) \stackrel{\text{def}}{=} \{b \in A \mid d(a,b) \le d\} \cup \{\text{all constants in vocabulary}\}$$

Definition

The d-type of a is the isomorphism type of the substructure induced by $N(a,d)$, plus the constant a.

Definition

$\boldsymbol{A}, \boldsymbol{B}$ are *d-equivalent* if, for each d-type, they have the same number of elements of that type.

Hanf's Lemma

Theorem

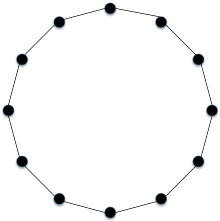
Let $d \geq 3^{k-1} - 1$. If $\boldsymbol{A}, \boldsymbol{B}$ are d-equivalent, then $\boldsymbol{A} \sim_k \boldsymbol{B}$.

The proof exhibits a winning strategy for the duplicator.

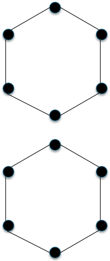
We will omit the proof.

CONNECTIVITY Not Expressible in FO

Claim duplicator has winning strategy with $k = 2$ pebbles.



$\mathbf{C_{12}}$

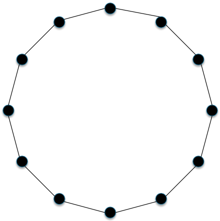


$\mathbf{C_6 \cup C_6}$

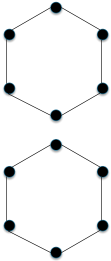
CONNECTIVITY Not Expressible in FO

Claim duplicator has winning strategy with $k = 2$ pebbles.

Proof Use Hanf's lemma. $k = 2$ and $d = 3^{k-1} - 1 = 2$



$\mathbf{C_{12}}$

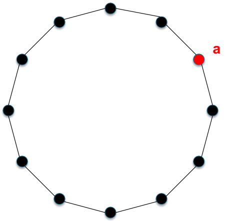


$\mathbf{C_6 \cup C_6}$

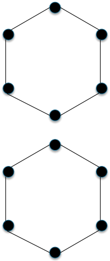
CONNECTIVITY Not Expressible in FO

Claim duplicator has winning strategy with $k = 2$ pebbles.

Proof Use Hanf's lemma. $k = 2$ and $d = 3^{k-1} - 1 = 2$



$\mathbf{C_{12}}$



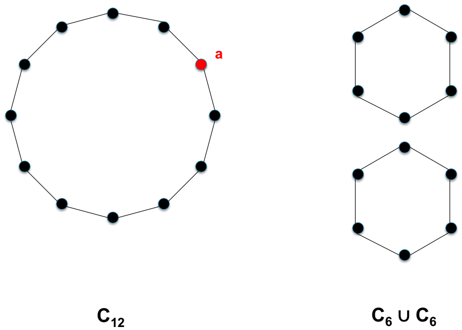
$\mathbf{C_6 \cup C_6}$

CONNECTIVITY Not Expressible in FO

Claim duplicator has winning strategy with $k = 2$ pebbles.

Proof Use Hanf's lemma. $k = 2$ and $d = 3^{k-1} - 1 = 2$

In class: describe $N(a, d)$

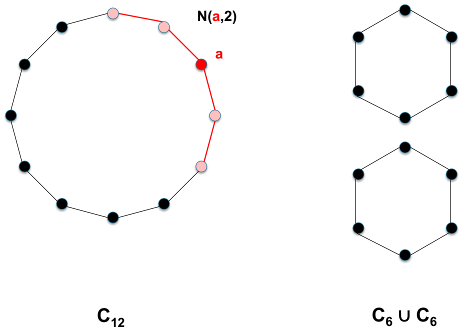


CONNECTIVITY Not Expressible in FO

Claim duplicator has winning strategy with $k = 2$ pebbles.

Proof Use Hanf's lemma. $k = 2$ and $d = 3^{k-1} - 1 = 2$

In class: describe $N(a, d)$

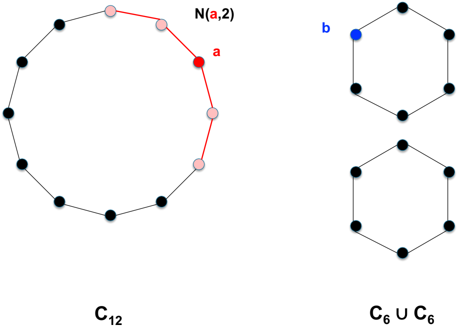


CONNECTIVITY Not Expressible in FO

Claim duplicator has winning strategy with $k = 2$ pebbles.

Proof Use Hanf's lemma. $k = 2$ and $d = 3^{k-1} - 1 = 2$

In class: describe $N(a, d)$

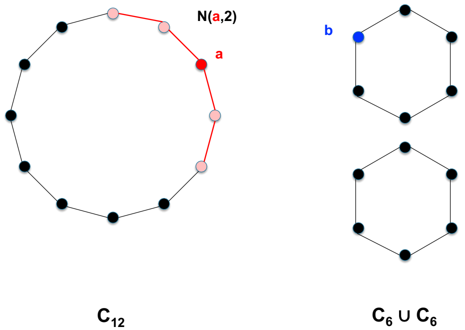


CONNECTIVITY Not Expressible in FO

Claim duplicator has winning strategy with $k = 2$ pebbles.

Proof Use Hanf's lemma. $k = 2$ and $d = 3^{k-1} - 1 = 2$

In class: describe $N(a, d)$ Describe $N(b, d)$

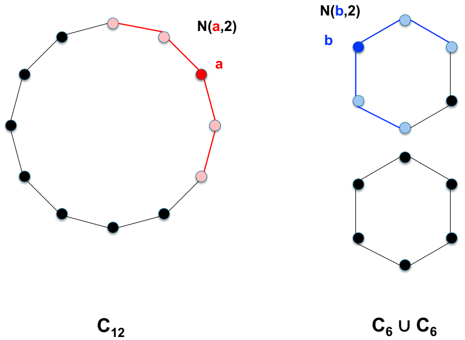


CONNECTIVITY Not Expressible in FO

Claim duplicator has winning strategy with $k = 2$ pebbles.

Proof Use Hanf's lemma. $k = 2$ and $d = 3^{k-1} - 1 = 2$

In class: describe $N(a, d)$ Describe $N(b, d)$



CONNECTIVITY Not Expressible in FO

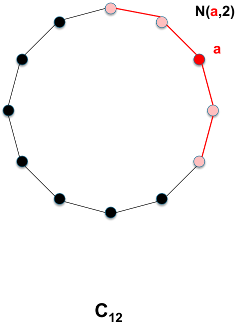
Claim duplicator has winning strategy with $k = 2$ pebbles.

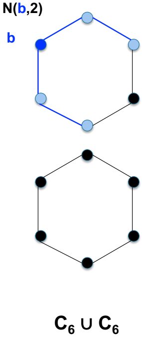
Proof Use Hanf's lemma. $k = 2$ and $d = 3^{k-1} - 1 = 2$

In class: describe $N(a, d)$ Describe $N(b, d)$

Their types are $\bullet - \bullet - X - \bullet - \bullet$. There are 12 in each structure.

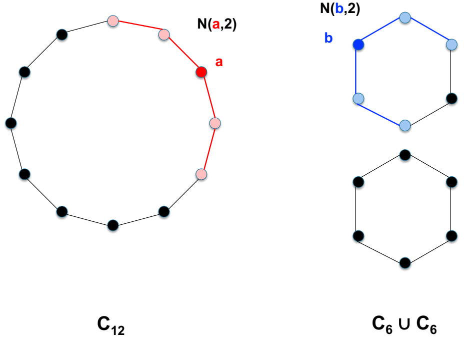
Therefore, duplicator wins with $k = 2$ pebbles.





CONNECTIVITY Not Expressible in FO

Claim duplicator has winning strategy with $k = 2$ pebbles.
Proof Use Hanf's lemma. $k = 2$ and $d = 3^{k-1} - 1 = 2$
In class: describe $N(a, d)$ Describe $N(b, d)$
Their types are $\bullet - \bullet - X - \bullet - \bullet$. There are 12 in each structure.
Therefore, duplicator wins with $k = 2$ pebbles.



At home: prove that spoiler wins in 3 rounds

From FO to MSO

The k-round Ehrenfeucht-Fraisse game is useful only for FO:

$$\boldsymbol{A} \equiv_k \boldsymbol{B} \text{ iff } \boldsymbol{A} \sim_k \boldsymbol{B}$$

Need to extend this to a game for ∃MSO.

(r, k)-Ajtai-Fagin game

The (r, k)-Ajtai-Fagin Game

The (r, k)-Ajtai-Fagin game for ∃MSO and a problem P is the following:

- Duplicator picks a structure $\boldsymbol{A}$ that satisfies P.
- Spoiler picks r unary relations $U_1^A, \ldots, U_r^A$ on $\boldsymbol{A}$.
- Duplicator picks a structure $\boldsymbol{B}$ that does not satisfy P.
- Duplicator picks $U_1^B, \ldots, U_r^B$ in $\boldsymbol{B}$.
- Spoiler and Duplicator play an EF game with k pebbles on the structures $(\boldsymbol{A}, U_1^A, \ldots, U_r^A)$ and $(\boldsymbol{B}, U_1^B, \ldots, U_r^B)$.

Games for ∃MSO

Lemma

If Duplicator wins the (r, k) game, then no ∃MSO sentence with r 2-nd order quantifiers and k 1-st order quantifiers can express P.

Games for ∃MSO

Lemma

If Duplicator wins the (r, k) game, then no ∃MSO sentence with r 2-nd order quantifiers and k 1-st order quantifiers can express P.

Proof Assume that $\boxed{\varphi = \exists U_1 \cdots \exists U_r \psi}$ expresses P. Then spoiler wins:

- Duplicator chooses $\boldsymbol{A}$ s.t. $P(\boldsymbol{A}) = \mathtt{TRUE}$. Then $\boldsymbol{A} \models \exists U_1 \cdots \exists U_r \psi$

Games for $\exists$MSO

Lemma

If Duplicator wins the (r, k) game, then no $\exists$MSO sentence with r 2-nd order quantifiers and k 1-st order quantifiers can express P.

Proof Assume that $\boxed{\varphi = \exists U_1 \cdots \exists U_r \psi}$ expresses P. Then spoiler wins:

- Duplicator chooses $\boldsymbol{A}$ s.t. $P(\boldsymbol{A}) = \texttt{TRUE}$. Then $\boldsymbol{A} \vDash \exists U_1 \cdots \exists U_r \psi$
- Spoiler chooses $U_1^A, \ldots, U_r^A$ s.t. $\boxed{(\boldsymbol{A}, U_1^A, \ldots, U_r^A) \vDash \psi}$

Games for $\exists$MSO

Lemma

If Duplicator wins the (r, k) game, then no $\exists$MSO sentence with r 2-nd order quantifiers and k 1-st order quantifiers can express P.

Proof Assume that $\boxed{\varphi = \exists U_1 \cdots \exists U_r \psi}$ expresses P. Then spoiler wins:

- Duplicator chooses $\boldsymbol{A}$ s.t. $P(\boldsymbol{A}) = \texttt{TRUE}$. Then $\boldsymbol{A} \models \exists U_1 \cdots \exists U_r \psi$
- Spoiler chooses $U_1^A, \ldots, U_r^A$ s.t. $\boxed{(\boldsymbol{A}, U_1^A, \ldots, U_r^A) \models \psi}$
- Duplicator must choose $\boldsymbol{B}$ s.t. $P(\boldsymbol{B}) = \texttt{FALSE}$. Thus, $\boldsymbol{B} \not\models \varphi$.

Games for $\exists$MSO

Lemma

If Duplicator wins the (r, k) game, then no $\exists$MSO sentence with r 2-nd order quantifiers and k 1-st order quantifiers can express P.

Proof Assume that $\boxed{\varphi = \exists U_1 \cdots \exists U_r \psi}$ expresses P. Then spoiler wins:

- Duplicator chooses $\boldsymbol{A}$ s.t. $P(\boldsymbol{A}) = \texttt{TRUE}$. Then $\boldsymbol{A} \vDash \exists U_1 \cdots \exists U_r \psi$
- Spoiler chooses $U_1^A, \ldots, U_r^A$ s.t. $\boxed{(\boldsymbol{A}, U_1^A, \ldots, U_r^A) \vDash \psi}$
- Duplicator must choose $\boldsymbol{B}$ s.t. $P(\boldsymbol{B}) = \texttt{FALSE}$. Thus, $\boldsymbol{B} \nvDash \varphi$.
- Duplicator chooses $U_1^B, \ldots, U_k^B$: $\boxed{(\boldsymbol{B}, U_1^B, \ldots, U_r^B) \nvDash \psi}$

Games for ∃MSO

Lemma

If Duplicator wins the (r, k) game, then no ∃MSO sentence with r 2-nd order quantifiers and k 1-st order quantifiers can express P.

Proof Assume that $\boxed{\varphi = \exists U_1 \cdots \exists U_r \psi}$ expresses P. Then spoiler wins:

- Duplicator chooses $\boldsymbol{A}$ s.t. $P(\boldsymbol{A}) = \texttt{TRUE}$. Then $\boldsymbol{A} \vDash \exists U_1 \cdots \exists U_r \psi$
- Spoiler chooses $U_1^A, \ldots, U_r^A$ s.t. $\boxed{(\boldsymbol{A}, U_1^A, \ldots, U_r^A) \vDash \psi}$
- Duplicator must choose $\boldsymbol{B}$ s.t. $P(\boldsymbol{B}) = \texttt{FALSE}$. Thus, $\boldsymbol{B} \nvDash \varphi$.
- Duplicator chooses $U_1^B, \ldots, U_k^B$: $\boxed{(\boldsymbol{B}, U_1^B, \ldots, U_r^B) \nvDash \psi}$
- Then $\boxed{(\boldsymbol{A}, U_1^A, \ldots, U_k^A) \not\sim_k (\boldsymbol{B}, U_1^B, \ldots, U_r^B)}$. Duplicator lost.

Games for ∃MSO

Lemma

If Duplicator wins the (r, k) game, then no ∃MSO sentence with r 2-nd order quantifiers and k 1-st order quantifiers can express P.

Proof Assume that $\boxed{\varphi = \exists U_1 \cdots \exists U_r \psi}$ expresses P. Then spoiler wins:

- Duplicator chooses $\boldsymbol{A}$ s.t. $P(\boldsymbol{A}) = \texttt{TRUE}$. Then $\boldsymbol{A} \models \exists U_1 \cdots \exists U_r \psi$
- Spoiler chooses $U_1^A, \ldots, U_r^A$ s.t. $\boxed{(\boldsymbol{A}, U_1^A, \ldots, U_r^A) \models \psi}$
- Duplicator must choose $\boldsymbol{B}$ s.t. $P(\boldsymbol{B}) = \texttt{FALSE}$. Thus, $\boldsymbol{B} \not\models \varphi$.
- Duplicator chooses $U_1^B, \ldots, U_k^B$: $\boxed{(\boldsymbol{B}, U_1^B, \ldots, U_r^B) \not\models \psi}$
- Then $\boxed{(\boldsymbol{A}, U_1^A, \ldots, U_k^A) \not\sim_k (\boldsymbol{B}, U_1^B, \ldots, U_r^B)}$. Duplicator lost.

Converse holds too, but we don't need it.

Proof of Fagin's Theorem

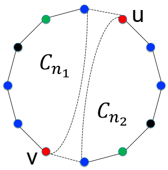
CONNECTIVITY is not expressible in ∃MSO.

Proof of Fagin's Theorem

CONNECTIVITY is not expressible in ∃MSO.

Claim: for all r, k, duplicator wins the (r, k)-Ajtai-Fagin game.

Duplicator chooses C_n; spoiler chooses $U_1, \ldots, U_r$; thus 2^r colors



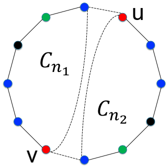
Proof of Fagin's Theorem

CONNECTIVITY is not expressible in ∃MSO.

Claim: for all r, k, duplicator wins the (r, k)-Ajtai-Fagin game.

Duplicator chooses C_n; spoiler chooses $U_1, \ldots, U_r$; thus 2^r colors

There are $t \leq 2^{r(2d+1)}$ d-types why?



Proof of Fagin's Theorem

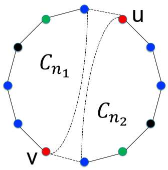
CONNECTIVITY is not expressible in ∃MSO.

Claim: for all r, k, duplicator wins the (r, k)-Ajtai-Fagin game.

Duplicator chooses C_n; spoiler chooses $U_1, \ldots, U_r$; thus 2^r colors

There are $t \leq 2^{r(2d+1)}$ d-types why?

If n is big: $\exists u, v$, same type, $d(u, v) \geq 2d + 2$.



Proof of Fagin's Theorem

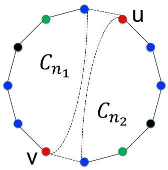
CONNECTIVITY is not expressible in ∃MSO.

Claim: for all r, k, duplicator wins the (r, k)-Ajtai-Fagin game.

Duplicator chooses C_n; spoiler chooses $U_1, \ldots, U_r$; thus 2^r colors

There are $t \leq 2^{r(2d+1)}$ d-types why?

If n is big: $\exists u, v$, same type, $d(u, v) \geq 2d + 2$.
One type occurs $\geq n/t$ times;
u, v = first, middle: $d(u, v) \geq n/(2t)$



Proof of Fagin's Theorem

CONNECTIVITY is not expressible in ∃MSO.

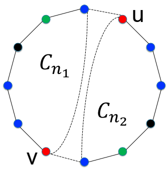
Claim: for all r, k, duplicator wins the (r, k)-Ajtai-Fagin game.

Duplicator chooses C_n; spoiler chooses $U_1, \ldots, U_r$; thus 2^r colors

There are $t \leq 2^{r(2d+1)}$ d-types why?

If n is big: $\exists u, v$, same type, $d(u, v) \geq 2d + 2$.
One type occurs $\geq n/t$ times;
u, v = first, middle: $d(u, v) \geq n/(2t)$

"Cut" C_n at u,v, obtain two cycles C_{n_1}, C_{n_2}



Proof of Fagin's Theorem

CONNECTIVITY is not expressible in ∃MSO.

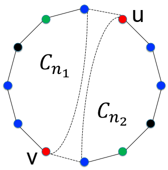
Claim: for all r, k, duplicator wins the (r, k)-Ajtai-Fagin game.

Duplicator chooses C_n; spoiler chooses $U_1, \ldots, U_r$; thus 2^r colors

There are $t \leq 2^{r(2d+1)}$ d-types why?

If n is big: $\exists u, v$, same type, $d(u, v) \geq 2d + 2$.
One type occurs $\geq n/t$ times;
u, v = first, middle: $d(u, v) \geq n/(2t)$

"Cut" C_n at u,v, obtain two cycles C_{n_1}, C_{n_2}



Hanf's Lemma: $C_n \sim_k (C_{n_1} \cup C_{n_2})$,

Discussion

- $\exists$MSO $\neq$ $\forall$MSO
- $\exists$SO $\neq$ $\forall$SO major open problem.
- Games are wonderful: EF games can be extended to Ajtai-Fagin, to MSO-games (which allows us to define MSO-types), to infinitary logics.

Next week: recursion, infinitary logics, pebble games