Finite Model Theory
Lecture 9: Second Order Logic

Spring 2025

Second Order Logic

Definition

Definition

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

Definition

Definition

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

- Connectors: $\wedge, \vee, \neg$
- First order quantifiers: $\forall x, \exists y$
- Second order quantifiers: $\forall X, \exists Y$
- Atoms: $R(x, y), X(u, v, w),\ Y(v)$.

Definition

Definition

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

- Connectors: $\wedge, \vee, \neg$
- First order quantifiers: $\forall x, \exists y$
- Second order quantifiers: $\forall X, \exists Y$
- Atoms: $R(x,y), X(u,v,w),\ Y(v)$.

Each set variable has an arity, e.g arity$(X) = 3$, arity$(Y) = 1$

We only discuss SO over finite models. Standard meaning: $\boldsymbol{A} \vDash \varphi$

Example: Connectivity

Check if a directed graph $G = (V, E)$ is connected.

In FO: not possible.

Example: Connectivity

Check if a directed graph $G = (V, E)$ is connected.

In FO: not possible.

In SO:

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

“For every U, if it and its complement are nonempty, then there exist an edge between U and its complement”

3-Colorability

Check if an undirected graph $G = (V, E)$ is 3-colorable.

3-Colorability

Check if an undirected graph $G = (V, E)$ is 3-colorable.

$$
\begin{aligned}
\exists R \exists B \exists G \forall x (R(x) \vee B(x) \vee G(x)) \\
\wedge \forall x \forall y (E(x, y) \to \neg(R(x) \wedge R(y))) \\
\wedge \forall x \forall y (E(x, y) \to \neg(G(x) \wedge G(y))) \\
\wedge \forall x \forall y (E(x, y) \to \neg(B(x) \wedge B(y)))
\end{aligned}
$$

Example: Hamiltonean

Check if a directed graph $G = (V, E)$ admits a Hamiltonean path.

Example: Hamiltonean

Check if a directed graph $G = (V, E)$ admits a Hamiltonean path.

$$\begin{aligned}\exists <(\forall x \neg(x < x)) \wedge \\ (\forall x \forall y (x < y \vee y < x \vee x = y)) \\ (\forall x \forall y \forall z (x < y \wedge y < z \Rightarrow x < z))\end{aligned}$$

$$\forall x \forall y (x < y \wedge \neg \exists z (x < z \wedge z < y) \Rightarrow E(x, y))$$

"There exists a total order on the nodes such that for any successive nodes x, y there exists an edge $E(x, y)$."

At home: modify to check for a Hamiltonean cycle.

Quantifier Order

An SO sentence can be rewritten such that all 2nd order quantifier precede the 1st order.

$$\exists x \quad \underbrace{\forall \ldots \exists \ldots}_{\text{1st and 2nd order}} \quad \varphi \qquad\qquad \exists X \underbrace{\forall \ldots \exists \ldots}_{\text{keep them}} (\exists x (X(x) \wedge \varphi))$$

Quantifier Order

An SO sentence can be rewritten such that all 2nd order quantifier precede the 1st order.

$$\exists x \quad \underbrace{\forall \ldots \exists \ldots}_{\text{1st and 2nd order}} \quad \varphi \qquad\qquad \exists X \underbrace{\forall \ldots \exists \ldots}_{\text{keep them}} (\exists x (X(x) \wedge \varphi))$$

$$\forall x \quad \underbrace{\forall \ldots \exists \ldots}_{\text{1st and 2nd order}} \quad \varphi \qquad\qquad \forall X \underbrace{\forall \ldots \exists \ldots}_{\text{keep them}} (\forall x (X(x) \rightarrow \varphi))$$

Fragments of SO

Existential SO

Existential SO allows only existential quantifiers of the 2nd order.

$$\boxed{\exists X_1 \exists X_2 \cdots \exists X_k [\text{FO formula}]}$$

Notation: ESO or $\exists$SO

∃SO and NP

The following is a landmark result proven by Fagin:

Theorem (Fagin)

∃SO captures the class NP

More precisely: a problem is in NP iff it can be expressed in ∃SO

Example: 3-colorability, Hamiltonean path

∃SO and NP

The following is a landmark result proven by Fagin:

Theorem (Fagin)

∃SO captures the class NP

More precisely: a problem is in NP iff it can be expressed in ∃SO

Example: 3-colorability, Hamiltonean path

Proof ideas:

$\exists\mathrm{SO} \subseteq NP$: to evaluate φ "guess" the sets $X_1, X_2, \ldots$

$NP \subseteq \exists\mathrm{SO}$: given a non-deterministic PTIME Turing machine "guess" an execution trace: $\exists\mathrm{TRACE}(\ldots)$.

Monadic SO

Monadic Second Order Logic, MSO, restricts the 2nd order variables to be unary relations.

Examples: 3-colorability, connectivity:

$$\begin{aligned}\exists R \exists B \exists G \forall x(R(x) \vee B(x) \vee G(x))&\\ \wedge \forall x \forall y(E(x,y) \to \neg(R(x) \wedge R(y)))&\\ \wedge \forall x \forall y(E(x,y) \to \neg(G(x) \wedge G(y)))&\\ \wedge \forall x \forall y(E(x,y) \to \neg(B(x) \wedge B(y)))&\end{aligned}$$

$$\forall U \forall x \forall y \left((U(x) \wedge \neg U(y)) \to \exists u \exists v E(u,v) \wedge U(u) \wedge \neg U(v)\right)$$

Existential MSO

Monadic Existential SO, $\exists$MSO, restricts the 2nd order quantifiers to be existential, and applied only to the unary relations.

Example: 3-colorability, non-connectivity.

Discussion

- $\exists$SO captures NP; e.g. Hamiltonean path
- $\exists$MSO is sometimes called Monadic NP; e.g. 3-colorability.
- $\forall$SO captures coNP. $\exists\text{SO} \neq \forall\text{SO}$ is a major open problem
- Fagin proved $\exists\text{MSO} \neq \forall\text{MSO}$: Connectivity $\in \forall$MSO but $\notin \exists$MSO

MSO on Strings

Representing Strings

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

Example: representing *aabaca*

$<$

1	2
1	3
...	
5	6

P_a

1
2
4
6

$P_b = \cdots$

$P_c = \cdots$

Representing Strings

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

Example: representing *aabaca*

$<$

1	2
1	3
...	
5	6

P_a

1
2
4
6

$P_b = \cdots$

$P_c = \cdots$

Recall that we can define:
$\text{succ}(x, y) = (x < y) \wedge \neg\exists z(x < z \wedge z < y)$
$\text{isMin}(x) = \forall y(x = y \vee x < y)$
The elements: $\overline{\min}$, $\overline{\max}$

Logic over Strings

A sentence φ defines a language $\{w \mid w \models \varphi\}$.

- What languages can be define in FO?
- What languages can be define in MSO?

Examples

$\Sigma = \{a, b\}$.

Which words w satisfy these sentences:

$\exists x P_a(x)$

Examples

$\Sigma = \{a, b\}$.

Which words w satisfy these sentences:

$\exists x P_a(x)$ $\qquad$ $(a|b)^*.a.(a|b)^*$

Examples

$\Sigma = \{a, b\}$.

Which words w satisfy these sentences:

$\exists x P_a(x)$ $(a|b)^*.a.(a|b)^*$

$\forall x \forall y((x < y \wedge P_a(y)) \Rightarrow P_a(x))$

Examples

$\Sigma = \{a, b\}$.

Which words w satisfy these sentences:

$\exists x P_a(x)$ $(a|b)^*.a.(a|b)^*$

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

Examples

$\Sigma = \{a, b\}$.

Which words w satisfy these sentences:

$\exists x P_a(x)$ $\qquad$ $(a|b)^*.a.(a|b)^*$

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

$\forall x \forall y(x < y \land P_a(x) \land P_a(y) \Rightarrow \exists z(x < z \land z < y \land P_b(z)))$

Examples

$\Sigma = \{a, b\}$.

Which words w satisfy these sentences:

$\exists x P_a(x)$ $(a|b)^*.a.(a|b)^*$

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

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

Examples

$\Sigma = \{a, b\}$.

Which words w satisfy these sentences:

$\exists x P_a(x)$ $(a|b)^*.a.(a|b)^*$

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

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

$$\begin{aligned}
&\forall x P_a(x) \wedge \\
&\exists X(X(\overline{\min}) \wedge \neg X(\overline{\max}) \wedge \\
&\quad \forall u \forall v(\mathrm{succ}(u, v) \Rightarrow (X(u) \wedge \neg X(v)) \vee (\neg X(u) \wedge X(v))))
\end{aligned}$$

Examples

$\Sigma = \{a, b\}$.

Which words w satisfy these sentences:

$\exists x P_a(x)$ $\quad (a|b)^*.a.(a|b)^*$

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

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

$$\begin{aligned} &\forall x P_a(x) \wedge \\ &\exists X (X(\overline{\min}) \wedge \neg X(\overline{\max}) \wedge \\ &\quad \forall u \forall v (\mathrm{succ}(u, v) \Rightarrow (X(u) \wedge \neg X(v)) \vee (\neg X(u) \wedge X(v)))) \end{aligned}$$

$(a.a)^*$

Büchi's Theorem

Theorem

MSO on strings captures regular languages

Proof of Büchi's Theorem: Part 1

If L is a regular language, then there exists φ in MSO s.t. $L = \{w \mid w \vDash \varphi\}$.

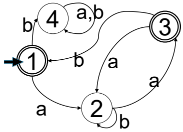
Proof of Büchi's Theorem: Part 1

If L is a regular language, then there exists φ in MSO s.t. $L = \{w \mid w \models \varphi\}$.

Assume L given by a deterministic automaton.

$$\varphi = \exists S_1 \exists S_2 \exists S_3 \exists S_4 (\psi)$$

where ψ asserts:



- $\forall x$ exactly one of $S_1(x), \dots, S_4(x)$ holds
- The minimum element x is in S_1.
- The maximum element is in S_1 or S_3.
- $\forall x, y\, (S_1(x) \wedge P_a(x) \wedge \mathrm{succ}(x, y) \Rightarrow S_2(y))$
- etc

Proof of Büchi's Theorem: Part 2

For every MSO sentence φ, the set $\{w \mid w \vDash \varphi\}$ is a regular language.

Proof by induction on φ.

Difficulty: φ is a sentence, but its subexpressions are formulas.

Need to given an interpretation to formulas.

Proof of Büchi's Theorem: Part 2

Fix $\Sigma = \{a, b\}$.

Proof of Büchi's Theorem: Part 2

Fix $\Sigma = \{a, b\}$.

To give meaning to $\varphi = P_a(x)$:

- Extend Σ to $\overline{\Sigma} = \{a, b, a^x, b^x\}$

Proof of Büchi's Theorem: Part 2

Fix $\Sigma = \{a, b\}$.

To give meaning to $\varphi = P_a(x)$:

- Extend Σ to $\overline{\Sigma} = \{a, b, a^x, b^x\}$
- a^x means: "symbol a and position x"

Proof of Büchi's Theorem: Part 2

Fix $\Sigma = \{a, b\}$.

To give meaning to $\varphi = P_a(x)$:

- Extend Σ to $\overline{\Sigma} = \{a, b, a^x, b^x\}$
- a^x means: "symbol a and position x"
- a means: "symbol a and position $\neq x$"

Proof of Büchi's Theorem: Part 2

Fix $\Sigma = \{a, b\}$.

To give meaning to $\varphi = P_a(x)$:

- Extend Σ to $\overline{\Sigma} = \{a, b, a^x, b^x\}$
- a^x means: "symbol a and position x"
- a means: "symbol a and position $\neq x$"
- Meaning of φ is $\overline{\Sigma}^*.a^x.\overline{\Sigma}^*$

Proof of Büchi's Theorem: Part 2

Fix $\Sigma = \{a, b\}$.

To give meaning to $\varphi = P_a(x)$:

- Extend Σ to $\overline{\Sigma} = \{a, b, a^x, b^x\}$
- a^x means: "symbol a and position x"
- a means: "symbol a and position $\neq x$"
- Meaning of φ is $\overline{\Sigma}^*.a^x.\overline{\Sigma}^*$

To give meaning to $\varphi = P_a(x) \wedge x < y$:

- $\overline{\Sigma} = \{a, b, a^x, b^x, a^y, b^y, a^{xy}, b^{xy}\}$

Proof of Büchi's Theorem: Part 2

Fix $\Sigma = \{a, b\}$.

To give meaning to $\varphi = P_a(x)$:

- Extend Σ to $\overline{\Sigma} = \{a, b, a^x, b^x\}$
- a^x means: "symbol a and position x"
- a means: "symbol a and position $\neq x$"
- Meaning of φ is $\overline{\Sigma}^*.a^x.\overline{\Sigma}^*$

To give meaning to $\varphi = P_a(x) \wedge x < y$:

- $\overline{\Sigma} = \{a, b, a^x, b^x, a^y, b^y, a^{xy}, b^{xy}\}$ denote $a^{x-} = (a^x|a^{xy})$:
- Meaning of φ is $\left(\overline{\Sigma}^*.a^{x-}.\overline{\Sigma}^*\right) \cap \left(\overline{\Sigma}^*.(a^{x-}|b^{x-}).\overline{\Sigma}^*.(a^{y-}|b^{y-}).\overline{\Sigma}^*\right)$

Proof of Büchi's Theorem: Part 2

$V = \mathrm{Vars}(\varphi)$. Extend Σ to $\overline{\Sigma} = \Sigma \times 2^V$.

Convert MSO formula φ to an automaton A_φ. Sketch:

Proof of Büchi's Theorem: Part 2

$V = \mathrm{Vars}(\varphi)$. Extend Σ to $\overline{\Sigma} = \Sigma \times 2^V$.

Convert MSO formula φ to an automaton A_φ. Sketch:

- $P_a(x)$: automaton for $\overline{\Sigma}^*.a^{x-}.\overline{\Sigma}^*$
  where a^{x-} means "any a-symbol that has at least annotation x"
  $$a^{x-} = (a^x|a^{xy}|\cdots|a^{xX}|a^{xyX}\cdots)$$
- $x < y$: similar to what we saw on previous slide.

Proof of Büchi's Theorem: Part 2

$V = \mathrm{Vars}(\varphi)$. Extend Σ to $\overline{\Sigma} = \Sigma \times 2^V$.

Convert MSO formula φ to an automaton A_φ. Sketch:

- $P_a(x)$: automaton for $\overline{\Sigma}^*.a^{x-}.\overline{\Sigma}^*$
  where a^{x-} means "any a-symbol that has at least annotation x"
  $$a^{x-} = (a^x | a^{xy} | \cdots | a^{xX} | a^{xyX} \cdots)$$
- $x < y$: similar to what we saw on previous slide.
- $X(x)$: $\overline{\Sigma}^*.(a^{x,X} | b^{x,X} | \cdots).\overline{\Sigma}^*$.
  In words: "some symbol has both annotations x and X."

Proof of Büchi's Theorem: Part 2

$V = \text{Vars}(\varphi)$. Extend Σ to $\overline{\Sigma} = \Sigma \times 2^V$.

Convert MSO formula φ to an automaton A_φ. Sketch:

- $P_a(x)$: automaton for $\overline{\Sigma}^*.a^{x-}.\overline{\Sigma}^*$
  where a^{x-} means "any a-symbol that has at least annotation x"
  $a^{x-} = (a^x | a^{xy} | \cdots | a^{xX} | a^{xyX} \cdots)$
- $x < y$: similar to what we saw on previous slide.
- $X(x)$: $\overline{\Sigma}^*.(a^{x,X} | b^{x,X} | \cdots).\overline{\Sigma}^*$.
  In words: "some symbol has both annotations x and X."
- $\neg\varphi$: automaton $A_{\neg\varphi}$ computes the complement of A_φ how??

Proof of Büchi's Theorem: Part 2

$V = \text{Vars}(\varphi)$. Extend Σ to $\overline{\Sigma} = \Sigma \times 2^V$.

Convert MSO formula φ to an automaton A_φ. Sketch:

- $P_a(x)$: automaton for $\overline{\Sigma}^*.a^{x-}.\overline{\Sigma}^*$
  where a^{x-} means "any a-symbol that has at least annotation x"
  $$a^{x-} = (a^x | a^{xy} | \cdots | a^{xX} | a^{xyX} \cdots)$$
- $x < y$: similar to what we saw on previous slide.
- $X(x)$: $\overline{\Sigma}^*.(a^{x,X} | b^{x,X} | \cdots).\overline{\Sigma}^*$.
  In words: "some symbol has both annotations x and X."
- $\neg\varphi$: automaton $A_{\neg\varphi}$ computes the complement of A_φ how??
- $\exists x\varphi$. Intersect A_φ with "x is unique", then drop the annotation x.

Proof of Büchi's Theorem: Part 2

$V = \text{Vars}(\varphi)$. Extend Σ to $\overline{\Sigma} = \Sigma \times 2^V$.

Convert MSO formula φ to an automaton A_φ. Sketch:

- $P_a(x)$: automaton for $\overline{\Sigma}^*.a^{x-}.\overline{\Sigma}^*$
  where a^{x-} means "any a-symbol that has at least annotation x"
  $a^{x-} = (a^x|a^{xy}|\cdots|a^{xX}|a^{xyX}\cdots)$
- $x < y$: similar to what we saw on previous slide.
- $X(x)$: $\overline{\Sigma}^*.(a^{x,X}|b^{x,X}|\cdots).\overline{\Sigma}^*$.
  In words: "some symbol has both annotations x and X."
- $\neg\varphi$: automaton $A_{\neg\varphi}$ computes the complement of A_φ how??
- $\exists x\varphi$. Intersect A_φ with "x is unique", then drop the annotation x.
- $\exists X\varphi$. Drop the annotation X from A_φ

Discussion

- MSO over strings captures regular languages.
- The data complexity of MSO over strings is linear time!
  Fixed φ: one can check $w \vDash \varphi$ in time $O(|w|)$

  Contrast with SO = NP.
- Every MSO sentence over strings is equivalent to an $\exists$MSO sentence!
  Contrast: $\exists$SO = $\forall$SO iff $NP = coNP$.

Courcelle's Theorem

Fix a number $k \in \mathbb{N}$.

Theorem (Courcelle)

Every formula in $\varphi \in MSO$ can be evaluated in linear time over structures with tree-width $\leq k$.

This is an amazing result! Caveats:

- The data complexity is linear time, but the expression complexity is tower of exponentials
- Complexity of tree decomposition:
  - Finding the optimal tree decomposition is NP-complete.
  - But given k, checking if a structure has tree-width $\leq k$ is in time $O(n^k)$

FO on Strings

FO on Strings

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:

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

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

Star-Free Languages

Fix an alphabet Σ. Regular expressions are:

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

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:

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

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:

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

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:

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

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:

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

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^* $\quad C(\varnothing)$
- b^* $\quad C(\Sigma^*.a.\Sigma^*)$
- $(a.b)^*$ $\quad C(\Sigma^*.a.a.\Sigma^* \cup \Sigma^*.b.b.\Sigma^* \cup b.\Sigma^* \cup \Sigma^*.a)$
- $(a.a)^*$

Star-Free Languages

Fix an alphabet Σ. Regular expressions are:

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

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

FO on Strings

Theorem

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

Proof: Part 1

Regular expression E. We construct Φ s.t. $w \vDash \Phi$ iff $w \in L(E)$.

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 $\neg\varphi_E(x, y)$

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 adapt that prove to construct a regular expression instead?

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 adapt that prove to construct a regular expression instead?

Seems promising: in most of the proof we constructed automata for intersection, union, complement: do the same for regular expressions.

Proof: Part 2

For each sentence φ, construct regular expression E_φ s.t. $w \models \varphi$ 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 adapt that prove to construct a regular expression instead?

Seems promising: in most of the proof we constructed automata for intersection, union, complement: do the same for regular expressions.

The problem is how we handled $\exists$ Will show next.

Handling $\exists$ in the Proof of Büchi's Theorem

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. $f(L(A)) = L(f(A))$

Handling ∃ in the Proof of Büchi's Theorem

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. $f(L(A)) = L(f(A))$

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

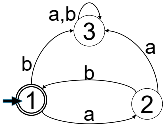
Handling $\exists$ in the Proof of Büchi's Theorem

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. $f(L(A)) = L(f(A))$

$(a.b)^*$:



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

Handling $\exists$ in the Proof of Büchi's Theorem

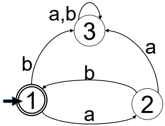
Fix two alphabets and a function $f : \overline{\Sigma} \to \Sigma$.

If A is an automaton, then:

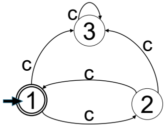
$\boxed{w \in L(f(A)) \text{ iff } \exists u(u \in L(A) \wedge f(u) = w)}$ i.e. $f(L(A)) = L(f(A))$

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

$(a.b)^*$:



$(c.c)^*$:



Handling $\exists$ in the Proof of Büchi's Theorem

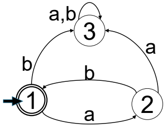
Fix two alphabets and a function $f : \overline{\Sigma} \to \Sigma$.

If A is an automaton, then:

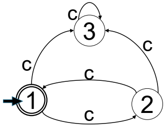
$\boxed{w \in L(f(A)) \text{ iff } \exists u(u \in L(A) \wedge f(u) = w)}$ i.e. $f(L(A)) = L(f(A))$

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

$(a.b)^*$:



$(c.c)^*$:



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

Handling $\exists$ in the Proof of Büchi's Theorem

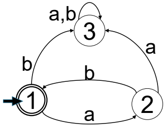
Fix two alphabets and a function $f : \overline{\Sigma} \to \Sigma$.

If A is an automaton, then:

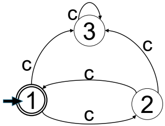
$\boxed{w \in L(f(A)) \text{ iff } \exists u(u \in L(A) \wedge f(u) = w)}$ i.e. $f(L(A)) = L(f(A))$

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

$(a.b)^*$:



$(c.c)^*$:



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

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

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

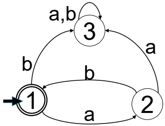
Handling ∃ in the Proof of Büchi's Theorem

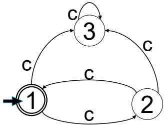
Fix two alphabets and a function $f : \overline{\Sigma} \to \Sigma$.

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. $f(L(A)) = L(f(A))$

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

$(a.b)^*$: 

$(c.c)^*$: 

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

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

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

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

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

Discussion

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

Will do induction on the quantifier depth k.

And we will use 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:

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

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.

$qr(\varphi) = 0$.
There are no such sentences. But we do need some sentences with $qr = 0$ for technical reasons (will be clear shortly).

So, add the constant $\overline{\mathrm{min}}$ to the vocabulary.

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

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. $\varphi = \exists x \psi(x)$

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

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

$$\boxed{w \models \varphi \text{ iff } \exists p \in \mathbb{N}, \exists(\sigma, \tau) \in S : \text{tp}_k(w^{\leq p}) = \sigma, \text{tp}_k(w^{>p}) = \tau}$$

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. $\varphi = \exists x \psi(x)$

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

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

$$\boxed{w \vDash \varphi \text{ iff } \exists p \in \mathbb{N}, \exists(\sigma, \tau) \in S : \text{tp}_k(w^{\leq p}) = \sigma, \text{tp}_k(w^{>p}) = \tau}$$

S is finite: $\{(\sigma_1, \tau_1), \ldots, (\sigma_n, \tau_n)\}$

The 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. $\varphi = \exists x \psi(x)$

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

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

$$\boxed{w \vDash \varphi \text{ iff } \exists p \in \mathbb{N}, \exists (\sigma, \tau) \in S : \mathrm{tp}_k(w^{\leq p}) = \sigma, \mathrm{tp}_k(w^{>p}) = \tau}$$

If $w \vDash \varphi$ then exists q s.t. $q \vDash \psi(q)$.

Take $\sigma = \mathrm{tp}_k(w^{\leq p})$, $\tau = \mathrm{tp}_k(w^{>p})$

By definition, $(\sigma, \tau) \in S$

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. $\varphi = \exists x \psi(x)$

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

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

$$\boxed{w \vDash \varphi \text{ iff } \exists p \in \mathbb{N}, \exists(\sigma, \tau) \in S : \text{tp}_k(w^{\leq p}) = \sigma, \text{tp}_k(w^{>p}) = \tau}$$

Assume $\exists p \in \mathbb{N}, \text{tp}_k(w^{\leq p}) = \sigma, \text{tp}_k(w^{>p}) = \tau$

Since $(\sigma, \tau) \in S$, $\exists u \in \Sigma^*, q \in \mathbb{N}, u \vDash \psi(q)$:

$$\text{tp}_k(u^{\leq q}) = \sigma, \text{tp}_k(u^{>q}) = \tau$$

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

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

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