Advanced Logic

http://www-sop.inria.fr/members/Martin.Avanzini/teaching/2023/AL/

Martin Avanzini (martin.avanzini@inria.fr)
Etienne Lozes (etienne.lozes@univ-cotedazur.fr)

MASTER INFORMATIQUE
UNIVERSITÉ CÔTE D'AZUR

2nd Semester M1, 2023

Today's Lecture

First Order-Logic Recap

- ⋆ structures, formulas and satisfiability

Monadic Second-Order Logic

1. weak monadic second-order (WMSO) logic
2. Regularity and WMSO definability
3. Decision problems

First-Order Logic Recap

First-Order Logic

- let $\mathcal{V} = \{x, y, \ldots\}$ be a set of variables
- let $\mathcal{R} = \{P, Q, \ldots\}$ and $\mathcal{F} = \{f, g, \ldots\}$ be a vocabulary of predicate/function symbols
- predicate and function symbols are equipped with an arity $\mathsf{ar} : \mathcal{R} \cup \mathcal{F} \to \mathbb{N}$
- first-order terms and formulas over $\mathcal{V}$, $\mathcal{R}$ and $\mathcal{F}$ are given by the following grammar:

$$
\begin{array}{rcll}
s, t & ::= & x \mid f(t_1, \ldots, t_{\mathsf{ar}(f)}) & \textit{(terms)} \\
\phi, \psi & ::= & \top \mid \bot & \textit{(atomic truth values)} \\
& \mid & P(t_1, \ldots, t_{\mathsf{ar}(P)}) \mid s = t & \textit{(predicates and equality)} \\
& \mid & \phi \vee \psi \mid \neg \phi & \textit{(Boolean connectives)} \\
& \mid & \exists x. \phi & \textit{(existential quantification)}
\end{array}
$$

First-Order Logic

- let $\mathcal{V} = \{x, y, \dots\}$ be a set of variables
- let $\mathcal{R} = \{P, Q, \dots\}$ and $\mathcal{F} = \{f, g, \dots\}$ be a vocabulary of predicate/function symbols
- predicate and function symbols are equipped with an arity $\mathsf{ar} : \mathcal{R} \cup \mathcal{F} \to \mathbb{N}$
- first-order terms and formulas over $\mathcal{V}$, $\mathcal{R}$ and $\mathcal{F}$ are given by the following grammar:

$$
\begin{array}{rcll}
s, t & ::= & x \mid f(t_1, \dots, t_{\mathsf{ar}(f)}) & \textit{(terms)} \\
\phi, \psi & ::= & \top \mid \bot & \textit{(atomic truth values)} \\
& \mid & P(t_1, \dots, t_{\mathsf{ar}(P)}) \mid s = t & \textit{(predicates and equality)} \\
& \mid & \phi \vee \psi \mid \neg \phi & \textit{(Boolean connectives)} \\
& \mid & \exists x. \phi & \textit{(existential quantification)}
\end{array}
$$

- further connectives definable:

$$\phi \to \psi \triangleq \neg \phi \vee \psi \qquad s \neq t \triangleq \neg(s = t) \qquad \phi \wedge \psi \triangleq \neg(\neg \phi \vee \neg \psi) \qquad \forall x.\phi \triangleq \neg(\exists x. \neg \phi) \qquad \dots$$

First-Order Logic

- let $\mathcal{V} = \{x, y, \dots\}$ be a set of variables
- let $\mathcal{R} = \{P, Q, \dots\}$ and $\mathcal{F} = \{f, g, \dots\}$ be a vocabulary of predicate/function symbols
- predicate and function symbols are equipped with an arity $\mathsf{ar} : \mathcal{R} \cup \mathcal{F} \to \mathbb{N}$
- first-order terms and formulas over $\mathcal{V}$, $\mathcal{R}$ and $\mathcal{F}$ are given by the following grammar:

$$
\begin{array}{rll}
s, t ::= x \mid f(t_1, \dots, t_{\mathsf{ar}(f)}) & & \textit{(terms)} \\
\phi, \psi ::= \top \mid \bot & & \textit{(atomic truth values)} \\
\mid P(t_1, \dots, t_{\mathsf{ar}(P)}) \mid s = t & & \textit{(predicates and equality)} \\
\mid \phi \vee \psi \mid \neg \phi & & \textit{(Boolean connectives)} \\
\mid \exists x. \phi & & \textit{(existential quantification)}
\end{array}
$$

- further connectives definable:

$$\phi \to \psi \triangleq \neg \phi \vee \psi \quad s \neq t \triangleq \neg(s = t) \quad \phi \wedge \psi \triangleq \neg(\neg \phi \vee \neg \psi) \quad \forall x.\phi \triangleq \neg(\exists x. \neg \phi) \quad \dots$$

- to avoid parenthesis, we fix precedence $\neg \;>\; \wedge, \vee \;>\; \exists, \forall$

Free Variables, Open and Closed Formulas

- a quantifier $\exists x.\phi$ binds the variable x within ϕ
- variables not bound are called free
- the set of variables free in ϕ is denoted by $\mathrm{fv}(\phi)$

$$\mathrm{fv}(E(x, y)) = \{x, y\} \qquad \mathrm{fv}(\exists y.E(x, y)) = \{x\} \qquad \mathrm{fv}(\forall x.\exists y.E(x, y)) = \varnothing$$

Free Variables, Open and Closed Formulas

- a quantifier $\exists x.\phi$ binds the variable x within ϕ
- variables not bound are called free
- the set of variables free in ϕ is denoted by $\mathsf{fv}(\phi)$

$$\mathsf{fv}(E(x,y)) = \{x,y\} \qquad \mathsf{fv}(\exists y.E(x,y)) = \{x\} \qquad \mathsf{fv}(\forall x.\exists y.E(x,y)) = \varnothing$$

- the formulas without free variables are called sentences (or closed formulas)
- otherwise they are called open

Free Variables, Open and Closed Formulas

- a quantifier $\exists x.\phi$ binds the variable x within ϕ
- variables not bound are called free
- the set of variables free in ϕ is denoted by $\mathrm{fv}(\phi)$

$$\mathrm{fv}(E(x,y)) = \{x,y\} \qquad \mathrm{fv}(\exists y.E(x,y)) = \{x\} \qquad \mathrm{fv}(\forall x.\exists y.E(x,y)) = \emptyset$$

- the formulas without free variables are called sentences (or closed formulas)
- otherwise they are called open
- we consider formulas equal up to renaming of bound variables
  - $\exists y.E(x,y)$ is equal to $\exists z.E(x,z)$ but **neither** to $\exists y.E(x,z)$ nor $\exists y.E(z,y)$

Satisfiability, Informally

- a formula is evaluated to a truth value by assigning meaning to predicates and functions

Satisfiability, Informally

- a formula is evaluated to a truth value by assigning meaning to predicates and functions
- a (first-order) structure (or model) $\mathcal{M} = (D, \mathcal{I})$ on a vocabulary $\mathcal{R}$ consists of
  - a non-empty domain D; and
  - an interpretation $\mathcal{I}(P) \subseteq D^{\mathrm{ar}(P)}$ for each predicate $P \in \mathcal{R}$
  - an interpretation $\mathcal{I}(f) : D^{\mathrm{ar}(f)} \to D$ for each function $f \in \mathcal{F}$

Satisfiability, Informally

- a formula is evaluated to a truth value by assigning meaning to predicates and functions
- a (first-order) structure (or model) $\mathcal{M} = (D, \mathcal{I})$ on a vocabulary $\mathcal{R}$ consists of
  - a non-empty domain D; and
  - an interpretation $\mathcal{I}(P) \subseteq D^{\mathrm{ar}(P)}$ for each predicate $P \in \mathcal{R}$
  - an interpretation $\mathcal{I}(f) : D^{\mathrm{ar}(f)} \to D$ for each function $f \in \mathcal{F}$
- sentences describes properties of structures, consider e.g., $\forall x. \exists y. E(x, y)$:
  - on directed graphs, with E interpreted as "edge": every node has a successor
  - on natural numbers, with E interpreted as "$<$": for every number there is a strictly bigger one

Satisfiability, Informally

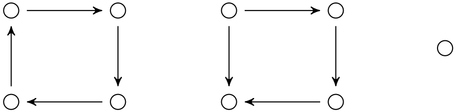
- a formula is evaluated to a truth value by assigning meaning to predicates and functions
- a (first-order) structure (or model) $\mathcal{M} = (D, \mathcal{I})$ on a vocabulary $\mathcal{R}$ consists of
  - a non-empty domain D; and
  - an interpretation $\mathcal{I}(P) \subseteq D^{\mathrm{ar}(P)}$ for each predicate $P \in \mathcal{R}$
  - an interpretation $\mathcal{I}(f) : D^{\mathrm{ar}(f)} \to D$ for each function $f \in \mathcal{F}$
- sentences describes properties of structures, consider e.g., $\forall x. \exists y. E(x, y)$:
  - on directed graphs, with E interpreted as "edge": every node has a successor
  - on natural numbers, with E interpreted as "$<$": for every number there is a strictly bigger one
- if a formula ϕ holds true in a model $\mathcal{M}$, we write

$$\mathcal{M} \models \phi$$

and say $\mathcal{M}$ models ϕ, or that ϕ is satisfiable with $\mathcal{M}$

Examples

1. consider the formula $\phi = \forall x. \exists y. E(x, y)$ and E interpreted by ...

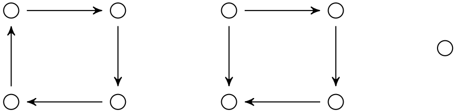


G_1 G_2 G_3

   – we have $G_1 \vDash \varphi$, $G_2 \nvDash \varphi$ and $G_3 \nvDash \varphi$

Examples

1. consider the formula $\phi = \forall x.\exists y.E(x,y)$ and E interpreted by ...



G_1 G_2 G_3

   - we have $G_1 \models \varphi$, $G_2 \not\models \varphi$ and $G_3 \not\models \varphi$

2. consider the formula $\exists x_1, x_2, x_3.\big(x_1 \neq x_2 \wedge x_2 \neq x_3 \wedge x_3 \neq x_1\big)$
   - the formula is satisfiable by all models with three objects in the domain

Consequence, Equivalence and Validity

- a sentence ϕ is a consequence of sentences $\Phi = \psi_1; \ldots; \psi_n$, in notation

$$\Phi \models \phi$$

  if all models satisfying all $\psi_i \in \Phi$ also satisfy ϕ

  - $\forall x.P(x) \rightarrow Q(x); \exists x.P(x) \models \exists x.Q(x)$

Consequence, Equivalence and Validity

- a sentence ϕ is a consequence of sentences $\Phi = \psi_1; \ldots; \psi_n$, in notation

$$\Phi \models \phi$$

  if all models satisfying all $\psi_i \in \Phi$ also satisfy ϕ

  – $\forall x.P(x) \rightarrow Q(x); \exists x.P(x) \models \exists x.Q(x)$

- two formulas ϕ and ψ are equivalent, in notation

$$\phi \equiv \psi$$

  if $\phi \models \psi$ and $\psi \models \phi$

  – $\forall x.P(x) \rightarrow Q(x) \equiv \forall x.\neg Q(x) \rightarrow \neg P(x)$

Consequence, Equivalence and Validity

- a sentence ϕ is a consequence of sentences $\Phi = \psi_1; \ldots; \psi_n$, in notation

$$\Phi \models \phi$$

if all models satisfying all $\psi_i \in \Phi$ also satisfy ϕ

  - $\forall x.P(x) \rightarrow Q(x); \exists x.P(x) \models \exists x.Q(x)$

- two formulas ϕ and ψ are equivalent, in notation

$$\phi \equiv \psi$$

if $\phi \models \psi$ and $\psi \models \phi$

  - $\forall x.P(x) \rightarrow Q(x) \equiv \forall x.\neg Q(x) \rightarrow \neg P(x)$

- a formula ϕ is valid if it is satisfiable for all models, in notation

$$\models \phi$$

  - this is to say that $\neg\phi$ is unsatisfiable
  - the formula $\forall x.x = x$ is trivially valid

Satisfiability, Formally

- an assignment (or valuation) for ϕ wrt. a model $\mathcal{M} = (D, \mathcal{I})$ is a function $\alpha : \mathsf{fv}(\phi) \to D$

Satisfiability, Formally

- ⋆ an assignment (or valuation) for ϕ wrt. a model $\mathcal{M} = (D, \mathcal{I})$ is a function $\alpha : \mathsf{fv}(\phi) \to D$
- ⋆ together with a model, we can now interpret open terms t in its domain D

$$\mathcal{I}_\alpha(x) \triangleq \alpha(x) \qquad \mathcal{I}_\alpha(f(t_1, \ldots, t_n)) \triangleq \mathcal{I}(f)(\mathcal{I}_\alpha(t_1), \ldots, \mathcal{I}_\alpha(t_n))$$

Satisfiability, Formally

- an assignment (or valuation) for ϕ wrt. a model $\mathcal{M} = (D, \mathcal{I})$ is a function $\alpha : \mathrm{fv}(\phi) \to D$
- together with a model, we can now interpret open terms t in its domain D

$$\mathcal{I}_\alpha(x) \triangleq \alpha(x) \qquad \mathcal{I}_\alpha(f(t_1, \ldots, t_n)) \triangleq \mathcal{I}(f)(\mathcal{I}_\alpha(t_1), \ldots, \mathcal{I}_\alpha(t_n))$$

- for a sentence ϕ, we can now define $\mathcal{M} \vDash \phi$ formally as $\mathcal{M}; \varnothing \vDash \phi$ where

$$\begin{array}{lll}
\mathcal{M}; \alpha \vDash \top \qquad \mathcal{M}; \alpha \nvDash \bot & & \\
\mathcal{M}; \alpha \vDash P(t_1, \ldots, t_n) & :\Longleftrightarrow & (\mathcal{I}_\alpha(t_1), \ldots, \mathcal{I}_\alpha(t_n)) \in \mathcal{I}(P) \\
\mathcal{M}; \alpha \vDash s = t & :\Longleftrightarrow & \mathcal{I}_\alpha(s) = \mathcal{I}_\alpha(t) \\
\mathcal{M}; \alpha \vDash \phi \vee \psi & :\Longleftrightarrow & \mathcal{M}; \alpha \vDash \phi \text{ or } \mathcal{M}; \alpha \vDash \psi \\
\mathcal{M}; \alpha \vDash \neg\phi & :\Longleftrightarrow & \mathcal{M}; \alpha \nvDash \phi \\
\mathcal{M}; \alpha \vDash \exists x.\phi & :\Longleftrightarrow & \mathcal{M}; \alpha[x \mapsto d] \vDash \phi \text{ for some } d \in D
\end{array}$$

Satisfiability, Formally

- an assignment (or valuation) for ϕ wrt. a model $\mathcal{M} = (D, \mathcal{I})$ is a function $\alpha : \mathrm{fv}(\phi) \to D$
- together with a model, we can now interpret open terms t in its domain D

$$\mathcal{I}_\alpha(x) \triangleq \alpha(x) \qquad \mathcal{I}_\alpha(f(t_1, \ldots, t_n)) \triangleq \mathcal{I}(f)(\mathcal{I}_\alpha(t_1), \ldots, \mathcal{I}_\alpha(t_n))$$

- for a sentence ϕ, we can now define $\mathcal{M} \vDash \phi$ formally as $\mathcal{M}; \varnothing \vDash \phi$ where

$$\begin{array}{lcl}
\mathcal{M}; \alpha \vDash \top \qquad \mathcal{M}; \alpha \nvDash \bot & & \\
\mathcal{M}; \alpha \vDash P(t_1, \ldots, t_n) & :\Longleftrightarrow & (\mathcal{I}_\alpha(t_1), \ldots, \mathcal{I}_\alpha(t_n)) \in \mathcal{I}(P) \\
\mathcal{M}; \alpha \vDash s = t & :\Longleftrightarrow & \mathcal{I}_\alpha(s) = \mathcal{I}_\alpha(t) \\
\mathcal{M}; \alpha \vDash \phi \vee \psi & :\Longleftrightarrow & \mathcal{M}; \alpha \vDash \phi \text{ or } \mathcal{M}; \alpha \vDash \psi \\
\mathcal{M}; \alpha \vDash \neg\phi & :\Longleftrightarrow & \mathcal{M}; \alpha \nvDash \phi \\
\mathcal{M}; \alpha \vDash \exists x.\phi & :\Longleftrightarrow & \mathcal{M}; \alpha[x \mapsto d] \vDash \phi \text{ for some } d \in D
\end{array}$$

Example

$$\begin{tikzcd}
\mathrm{a} \arrow[r] \arrow[dr, phantom, "\mathcal{G}"] & \mathrm{b} \arrow[d] \\
\mathrm{c} \arrow[u] & \mathrm{d} \arrow[l]
\end{tikzcd}$$

$$\begin{aligned}
\mathcal{G} \vDash \exists x.\exists y.E(x,y) &\Longleftrightarrow \mathcal{G}; \varnothing \vDash \exists x.\exists y.E(x,y) \\
&\Longleftarrow \mathcal{G}; x \mapsto \mathrm{a} \vDash \exists y.E(x,y) \\
&\Longleftarrow \mathcal{G}; x \mapsto \mathrm{a}; y \mapsto \mathrm{b} \vDash E(x,y) \\
&\Longleftrightarrow (\mathrm{a}, \mathrm{b}) \in \mathcal{I}(E)
\end{aligned}$$

Monadic Second-Order Logic

Monadic Second-Order Logic

Second Order-Logic

- in first-order logic, quantification confined to elements of the domain
- in second-order logic, quantification is permitted on relations
  - $\forall x. \exists X. \forall y. X(x, y) \leftrightarrow x = y$

Monadic Second-Order Logic

Second Order-Logic

- in first-order logic, quantification confined to elements of the domain
- in second-order logic, quantification is permitted on relations
  - $\forall x . \exists X . \forall y . X(x, y) \leftrightarrow x = y$

Monadic Second-Order Logic

- A predicate symbol P is monadic if its arity is 1

Monadic Second-Order Logic

Second Order-Logic

- in first-order logic, quantification confined to elements of the domain
- in second-order logic, quantification is permitted on relations
  - $\forall x.\exists X.\forall y.X(x,y) \leftrightarrow x = y$

Monadic Second-Order Logic

- A predicate symbol P is monadic if its arity is 1
- monadic second-order logic (MSO) confines second-order quantification to monadic predicates
  - monadic: $\forall x.\exists Y.\forall y.Y(y) \leftrightarrow x = y$
  - non-monadic: $\forall x.\exists X.\forall y.X(x,y) \leftrightarrow x = y$

Monadic Second-Order Logic

Second Order-Logic

- in first-order logic, quantification confined to elements of the domain
- in second-order logic, quantification is permitted on relations
  - $\forall x.\exists X.\forall y.X(x,y) \leftrightarrow x = y$

Monadic Second-Order Logic

- A predicate symbol P is monadic if its arity is 1
- monadic second-order logic (MSO) confines second-order quantification to monadic predicates
  - monadic: $\forall x.\exists Y.\forall y.Y(y) \leftrightarrow x = y$
  - non-monadic: $\forall x.\exists X.\forall y.X(x,y) \leftrightarrow x = y$
- quantification over sets, but not over arbitrary predicates
  - on graphs: quantification over nodes but not edges

Theories

- A theory is a set T of sentences such that for any sentence ϕ, if $T \models \phi$, then $\phi \in T$
  - a theory is closed under logical consequence

Theories

- A theory is a set T of sentences such that for any sentence ϕ, if $T \models \phi$, then $\phi \in T$
  - a theory is closed under logical consequence
- A theory is decidable if the problem of belonging to T is decidable
  - we have a decision procedure for reasoning about T

Theories

- A theory is a set T of sentences such that for any sentence ϕ, if $T \vDash \phi$, then $\phi \in T$
  - a theory is closed under logical consequence
- A theory is decidable if the problem of belonging to T is decidable
  - we have a decision procedure for reasoning about T
- A theory T is complete if for any sentence ϕ we have $\phi \in T$ or $\neg\phi \in T$.
  - a complete theory speaks about all formulas

Theories

- ⋆ A theory is a set T of sentences such that for any sentence ϕ, if $T \models \phi$, then $\phi \in T$
  - a theory is closed under logical consequence
- ⋆ A theory is decidable if the problem of belonging to T is decidable
  - we have a decision procedure for reasoning about T
- ⋆ A theory T is complete if for any sentence ϕ we have $\phi \in T$ or $\neg\phi \in T$.
  - a complete theory speaks about all formulas
- ⋆ for a class of structures $\mathcal{C}$, the theory of $\mathcal{C}$ is the set of sentences which are valid on all $\mathcal{M} \in \mathcal{C}$

Examples

1. The theory of Presburger Arithmetic, i.e., the theory of natural numbers with addition only is decidable
   - $\forall n.\exists m.(n = m + m) \vee (n = m + m + 1)$
   - Presburger Arithmetic admits a quantifier elimination procedure

Examples

1. The theory of Presburger Arithmetic, i.e., the theory of natural numbers with addition only is decidable
   - $\forall n.\exists m.(n = m + m) \vee (n = m + m + 1)$
   - Presburger Arithmetic admits a quantifier elimination procedure
2. The theory of Peano Arithmetic, i.e., the theory of natural numbers is undecidable
   - Gödels incompleteness theorem

Examples

1. The theory of Presburger Arithmetic, i.e., the theory of natural numbers with addition only is decidable
   - $\forall n.\exists m.(n = m + m) \vee (n = m + m + 1)$
   - Presburger Arithmetic admits a quantifier elimination procedure
2. The theory of Peano Arithmetic, i.e., the theory of natural numbers is undecidable
   - Gödels incompleteness theorem
3. The theory of graphs is undecidable

Examples

1. The theory of Presburger Arithmetic, i.e., the theory of natural numbers with addition only is decidable
   - $\forall n. \exists m. (n = m + m) \vee (n = m + m + 1)$
   - Presburger Arithmetic admits a quantifier elimination procedure
2. The theory of Peano Arithmetic, i.e., the theory of natural numbers is undecidable
   - Gödels incompleteness theorem
3. The theory of graphs is undecidable

Theorem (Büchi)

The theory of monadic second-order logic over $(\mathbb{N}, <)$ *is decidable*

Theorem (Rabin)

The theory of monadic second-order logic over trees is decidable

A First Step Towards Rabin's and Büchi's Result

consider only models over $\mathbb{N}$, ordered by $<$

Theorem (Büchi-Elgot-Trakhtenbrot)

The theory of weak monadic second-order logic over $(\mathbb{N}, <)$ is decidable

quantification over finite sets

Weak Monadic Second-Order Logic

Weak Monadic Second-Order Logic (WMSO)

- let $\mathcal{V}_1 = \{x, y, \dots\}$ be a set of first-order variables (ranging over $\mathbb{N}$)
- let $\mathcal{V}_2 = \{X, Y, \dots\}$ be monadic second-order variables (ranging over finite sets of $\mathbb{N}$)
- $\mathcal{R} = \{<\}$ and $\mathcal{F} = \varnothing$ is fixed, with $\mathrm{ar}(<) = 2$
- the set of WMSO formulas over $\mathcal{V}_1, \mathcal{V}_2$ is given by the following grammar:

$$\phi, \psi ::= \top \mid \bot \mid x < y \mid X(x) \mid \phi \vee \psi \mid \neg\phi \mid \exists x.\phi \mid \exists X.\phi$$

Weak Monadic Second-Order Logic (WMSO)

- let $\mathcal{V}_1 = \{x, y, \dots\}$ be a set of first-order variables (ranging over $\mathbb{N}$)
- let $\mathcal{V}_2 = \{X, Y, \dots\}$ be monadic second-order variables (ranging over finite sets of $\mathbb{N}$)
- $\mathcal{R} = \{<\}$ and $\mathcal{F} = \varnothing$ is fixed, with $\mathrm{ar}(<) = 2$
- the set of WMSO formulas over $\mathcal{V}_1, \mathcal{V}_2$ is given by the following grammar:

$$\phi, \psi ::= \top \;\big|\; \bot \;\big|\; x < y \;\big|\; X(x) \;\big|\; \phi \vee \psi \mid \neg\phi \;\big|\; \exists x.\phi \;\big|\; \exists X.\phi$$

- further definable connectives / formulas

$$\forall X.\phi \triangleq \neg(\exists X.\neg\phi) \quad x = 0 \triangleq \neg(\exists y.y < x) \quad x \leq y \triangleq \neg(y < x) \quad x = y \quad X(y + c) \quad \text{(exercise)}$$

Weak Monadic Second-Order Logic (WMSO)

- let $\mathcal{V}_1 = \{x, y, \dots\}$ be a set of first-order variables (ranging over $\mathbb{N}$)
- let $\mathcal{V}_2 = \{X, Y, \dots\}$ be monadic second-order variables (ranging over finite sets of $\mathbb{N}$)
- $\mathcal{R} = \{<\}$ and $\mathcal{F} = \varnothing$ is fixed, with $\mathsf{ar}(<) = 2$
- the set of WMSO formulas over $\mathcal{V}_1, \mathcal{V}_2$ is given by the following grammar:

$$\phi, \psi ::= \top \;\big|\; \bot \;\big|\; x < y \;\big|\; X(x) \;\big|\; \phi \vee \psi \mid \neg\phi \;\big|\; \exists x.\phi \;\big|\; \exists X.\phi$$

- further definable connectives / formulas

$$\forall X.\phi \triangleq \neg(\exists X.\neg\phi) \quad x = 0 \triangleq \neg(\exists y.y < x) \quad x \leq y \triangleq \neg(y < x) \quad x = y \quad X(y + c) \quad \text{(exercise)}$$

- weak: second-order variables refer to finite sets
  - $X(y)$ means informally $y \in X$ where X is finite set over $\mathbb{N}$
  - $\models \exists X.\forall x.X(x) \rightarrow \exists y.x < y \wedge X(y)$ $\qquad \alpha(X) = \varnothing$
  - $\not\models \exists X.(\forall x.x = 0 \rightarrow X(x)) \wedge (\forall x.X(x) \rightarrow \exists y.x < y \wedge X(y))$

Satisfiability

- since the model $(\mathbb{N}, \{<\})$ is fixed, the valuation of a formula depends only on an assignment α
- α maps first-order variables $x \in \mathcal{V}_1$ to $\mathbb{N}$, and second-order variables $X \in \mathcal{V}_2$ to finite subsets of $\mathbb{N}$

Satisfiability

- since the model $(\mathbb{N}, \{<\})$ is fixed, the valuation of a formula depends only on an assignment α
- α maps first-order variables $x \in \mathcal{V}_1$ to $\mathbb{N}$, and second-order variables $X \in \mathcal{V}_2$ to finite subsets of $\mathbb{N}$
- satisfiability relation takes the form $\alpha \models \phi$ and is inductively defined as expected:

$$\alpha \models \top \qquad \alpha \not\models \bot$$

$$\begin{aligned}
\alpha &\models x < y &&:\Longleftrightarrow && \alpha(x) < \alpha(y) \\
\alpha &\models X(x) &&:\Longleftrightarrow && \alpha(x) \in \alpha(X) \\
\alpha &\models \phi \vee \psi &&:\Longleftrightarrow && \alpha \models \phi \text{ or } \alpha \models \psi \\
\alpha &\models \neg\phi &&:\Longleftrightarrow && \alpha \not\models \phi \\
\alpha &\models \exists x.\phi &&:\Longleftrightarrow && \alpha[x \mapsto n] \models \phi \text{ for some } n \in \mathbb{N} \\
\alpha &\models \exists X.\phi &&:\Longleftrightarrow && \alpha[x \mapsto M] \models \phi \text{ for some \textit{finite} } M \subset \mathbb{N}
\end{aligned}$$

Connections to Formal Languages

- to encode words $w \in \Sigma^*$ over alphabet Σ we use to kinds of variables
  - first-order variables $x \in \mathcal{V}_1$ refer to positions within w
  - for each letter $\mathtt{a} \in \Sigma$, second-order variables $P_{\mathtt{a}} \in \mathcal{V}_2$ indicate the positions of $\mathtt{a}$ in w

w	a b b a
$P_{\mathtt{a}}$	$\{0, \quad 3\}$
$P_{\mathtt{b}}$	$\{\quad 1, 2 \quad\}$

Connections to Formal Languages

- to encode words $w \in \Sigma^*$ over alphabet Σ we use to kinds of variables
  - first-order variables $x \in \mathcal{V}_1$ refer to positions within w
  - for each letter $\mathtt{a} \in \Sigma$, second-order variables $P_{\mathtt{a}} \in \mathcal{V}_2$ indicate the positions of $\mathtt{a}$ in w

$$\left.\begin{array}{l|l} w & \mathtt{a\ b\ b\ a} \\ \hline P_{\mathtt{a}} & \{0, \quad\quad 3\} \\ P_{\mathtt{b}} & \{\quad 1, 2 \quad\} \end{array}\right\} \underline{\mathtt{abba}}$$

- thereby each word $w \in \Sigma^*$ uniquely determines an assignment, in notation $\underline{w}$

Connections to Formal Languages

- to encode words $w \in \Sigma^*$ over alphabet Σ we use to kinds of variables
  - first-order variables $x \in \mathcal{V}_1$ refer to positions within w
  - for each letter $\mathtt{a} \in \Sigma$, second-order variables $P_{\mathtt{a}} \in \mathcal{V}_2$ indicate the positions of $\mathtt{a}$ in w

$$\left.\begin{array}{l|l} w & \mathtt{a}\ \mathtt{b}\ \mathtt{b}\ \mathtt{a} \\ \hline P_{\mathtt{a}} & \{0, \quad 3\} \\ P_{\mathtt{b}} & \{\quad 1, 2 \quad\} \end{array}\right\} \underline{\mathtt{abba}}$$

- thereby each word $w \in \Sigma^*$ uniquely determines an assignment, in notation $\underline{w}$

Examples

- $\underline{\mathtt{ab}} \models \exists x. P_{\mathtt{a}}(x)$
- $\underline{\mathtt{ab}} \not\models \exists x. P_{\mathtt{c}}(x)$
- $\underline{\mathtt{ab}} \not\models \exists x. \exists y. x < y \wedge P_{\mathtt{b}}(x) \wedge P_{\mathtt{a}}(y)$
- $\underline{\mathtt{ab}} \not\models \exists X. \forall x. (X(x) \rightarrow P_{\mathtt{b}}(x)) \wedge \exists y. y = 0 \wedge X(y)$

Language of a WMSO Formula

- for alphabet Σ and WMSO formula ϕ s.t. $\mathsf{fv}(\phi) \subseteq \{P_{\mathsf{a}} \mid \mathsf{a} \in \Sigma\}$, we let

$$\mathsf{L}(\phi) \triangleq \{w \in \Sigma^* \mid \underline{w} \vDash \phi\}$$

  denote the language of ϕ
- a language L is WMSO definable iff there is some ϕ as above s.t. $L = \mathsf{L}(\phi)$

Language of a WMSO Formula

- for alphabet Σ and WMSO formula ϕ s.t. $\mathrm{fv}(\phi) \subseteq \{P_{\mathrm{a}} \mid \mathrm{a} \in \Sigma\}$, we let

$$\mathrm{L}(\phi) \triangleq \{w \in \Sigma^* \mid \underline{w} \vDash \phi\}$$

  denote the language of ϕ
- a language L is WMSO definable iff there is some ϕ as above s.t. $L = \mathrm{L}(\phi)$

Examples

ϕ	$L(\phi)$
$\exists x. P_{\mathrm{a}}(x)$	?
$\exists x. \exists y. x < y \wedge P_{\mathrm{b}}(x) \wedge P_{\mathrm{a}}(y)$	?
$\exists X. \forall x. (X(x) \rightarrow P_{\mathrm{b}}(x)) \wedge \exists y. y = 0 \wedge X(y)$	?

Language of a WMSO Formula

- for alphabet Σ and WMSO formula ϕ s.t. $\mathsf{fv}(\phi) \subseteq \{P_a \mid a \in \Sigma\}$, we let

$$\mathsf{L}(\phi) \triangleq \{w \in \Sigma^* \mid \underline{w} \vDash \phi\}$$

  denote the language of ϕ
- a language L is WMSO definable iff there is some ϕ as above s.t. $L = \mathsf{L}(\phi)$

Examples

ϕ	$L(\phi)$
$\exists x.P_a(x)$	$\{vaw \mid v, w \in \Sigma^*\}$
$\exists x.\exists y.x < y \wedge P_b(x) \wedge P_a(y)$	?
$\exists X.\forall x.(X(x) \to P_b(x)) \wedge \exists y.y = 0 \wedge X(y)$	?

Language of a WMSO Formula

- for alphabet Σ and WMSO formula ϕ s.t. $\mathsf{fv}(\phi) \subseteq \{P_a \mid a \in \Sigma\}$, we let

$$\mathsf{L}(\phi) \triangleq \{w \in \Sigma^* \mid \underline{w} \vDash \phi\}$$

  denote the language of ϕ
- a language L is WMSO definable iff there is some ϕ as above s.t. $L = \mathsf{L}(\phi)$

Examples

ϕ	$L(\phi)$
$\exists x.P_a(x)$	$\{vaw \mid v, w \in \Sigma^*\}$
$\exists x.\exists y.x < y \wedge P_b(x) \wedge P_a(y)$	$\{ubvaw \mid u, v, w \in \Sigma^*\}$
$\exists X.\forall x.(X(x) \rightarrow P_b(x)) \wedge \exists y.y = 0 \wedge X(y)$	?

Language of a WMSO Formula

- for alphabet Σ and WMSO formula ϕ s.t. $\mathsf{fv}(\phi) \subseteq \{P_\mathsf{a} \mid \mathsf{a} \in \Sigma\}$, we let

$$\mathrm{L}(\phi) \triangleq \{w \in \Sigma^* \mid \underline{w} \vDash \phi\}$$

  denote the language of ϕ
- a language L is WMSO definable iff there is some ϕ as above s.t. $L = \mathrm{L}(\phi)$

Examples

ϕ	$L(\phi)$
$\exists x. P_\mathsf{a}(x)$	$\{v\mathsf{a}w \mid v, w \in \Sigma^*\}$
$\exists x. \exists y. x < y \wedge P_\mathsf{b}(x) \wedge P_\mathsf{a}(y)$	$\{u\mathsf{b}v\mathsf{a}w \mid u, v, w \in \Sigma^*\}$
$\exists X. \forall x. (X(x) \rightarrow P_\mathsf{b}(x)) \wedge \exists y. y = 0 \wedge X(y)$	$\{\mathsf{b}w \mid w \in \Sigma^*\}$

Regularity and WMSO Definability

Büchi-Elgot-Trakhtenbrot

Theorem

Let $L \subseteq \Sigma^*$ *be a language. The following are equivalent:*

- ⋆ L *is regular*
- ⋆ L *is recognizable by a finite automata*
- ⋆ L *is WMSO definable*

Büchi-Elgot-Trakhtenbrot

Theorem

Let $L \subseteq \Sigma^*$ be a language. The following are equivalent:

- ⋆ *L is regular*
- ⋆ *L is recognizable by a finite automata*
- ⋆ *L is WMSO definable*

Proof Outline.

- ⋆ (1) $\Longleftrightarrow$ (2) Kleene's Theorem.
- ⋆ (2) $\Longrightarrow$ (3) Given an Automata $\mathcal{A}$, we define a WMSO formula $\phi_{\mathcal{A}}$ s.t. $\mathrm{L}(\mathcal{A}) = \mathrm{L}(\phi_{\mathcal{A}})$
- ⋆ (3) $\Longrightarrow$ (1) Given a WMSO formula ϕ, define a regular Language L_ϕ s.t. $\mathrm{L}(\phi) = L_\phi$

From Automatons to Formulas

Encoding for given $\mathcal{A} = (Q, \Sigma, q_I, \delta, F)$

- ⋆ first-order variables $m, n, \ldots$ refer to positions in input words w
- ⋆ for $\mathtt{a} \in \Sigma$: second-order variables $P_{\mathtt{a}}$ encode words: as before
- ⋆ for $q \in Q$: second-order variables X_q encode run: $X_q(m) \iff q_I \xrightarrow{\mathtt{a}_0} \ldots \xrightarrow{\mathtt{a}_m} q$

From Automatons to Formulas

Encoding for given $\mathcal{A} = (Q, \Sigma, q_I, \delta, F)$

- first-order variables $m, n, \ldots$ refer to positions in input words w
- for $\mathtt{a} \in \Sigma$: second-order variables $P_\mathtt{a}$ encode words: as before
- for $q \in Q$: second-order variables X_q encode run: $X_q(m) \iff q_I \xrightarrow{\mathtt{a}_0} \ldots \xrightarrow{\mathtt{a}_\mathtt{m}} q$

Example

example run	$p \xrightarrow{\mathtt{a}} q \xrightarrow{\mathtt{b}} p \xrightarrow{\mathtt{b}} r$
$P_\mathtt{a}$	$\{\ 0\ \}$
$P_\mathtt{b}$	$\{\ 1,\ 2\ \}$
X_p	$\{\ (-1)\ \ 1\ \}$
X_q	$\{\ 0\ \}$
X_r	$\{\ 2\ \}$

From Automatons to Formulas

Encoding for given $\mathcal{A} = (Q, \Sigma, q_I, \delta, F)$

- ⋆ first-order variables $m, n, \ldots$ refer to positions in input words w
- ⋆ for $\mathtt{a} \in \Sigma$: second-order variables $P_{\mathtt{a}}$ encode words: as before
- ⋆ for $q \in Q$: second-order variables X_q encode run: $X_q(m) \iff q_I \xrightarrow{\mathtt{a}_0} \ldots \xrightarrow{\mathtt{a}_m} q$

Example

example run	$p \xrightarrow{\mathtt{a}} q \xrightarrow{\mathtt{b}} p \xrightarrow{\mathtt{b}} r$
$P_{\mathtt{a}}$	$\{ \; 0 \; \}$
$P_{\mathtt{b}}$	$\{ \; 1, \; 2 \; \}$
X_p	$\{ (-1) \; 1 \; \}$
X_q	$\{ \; 0 \; \}$
X_r	$\{ \; 2 \}$

- ⋆ ultimately, $\phi_{\mathcal{A}} \triangleq \exists X_{q_1}. \ldots. \exists X_{q_n}. \psi_{\mathcal{A}}$ with $\psi_{\mathcal{A}}$ saying that X_{q_i} encode an accepting run of $\mathcal{A}$ on input word described by $P_{\mathtt{a}}$.

Linking Run-Variables

for all word lengths *len*, we define:

- ★ $\psi_{setup} \triangleq \forall m. m < len \rightarrow (\bigvee_{q \in Q} X_q(m)) \wedge (\bigwedge_{p \neq q} \neg(X_q(m) \wedge X_p(m)))$
  - reading $m < len$ symbols ends up in a state, and this state is unique

Linking Run-Variables

for all word lengths *len*, we define:

- $\psi_{setup} \triangleq \forall m. m < len \to (\bigvee_{q \in Q} X_q(m)) \land \left(\bigwedge_{p \neq q} \neg(X_q(m) \land X_p(m))\right)$
  - reading $m < len$ symbols ends up in a state, and this state is unique
- $\psi_{initial} \triangleq len = 0 \lor \bigvee_{\mathtt{a} \in \Sigma, p \in \delta(q_I, \mathtt{a})} \left(P_{\mathtt{a}}(0) \land X_p(0)\right)$
  - encoding of the initial transition

Linking Run-Variables

for all word lengths *len*, we define:

- $\psi_{setup} \triangleq \forall m. m < len \to (\bigvee_{q \in Q} X_q(m)) \land \left(\bigwedge_{p \neq q} \neg(X_q(m) \land X_p(m))\right)$
  - reading $m < len$ symbols ends up in a state, and this state is unique
- $\psi_{initial} \triangleq len = 0 \lor \bigvee_{\mathtt{a} \in \Sigma, p \in \delta(q_I, \mathtt{a})} \left(P_{\mathtt{a}}(0) \land X_p(0)\right)$
  - encoding of the initial transition
- $\psi_{run} \triangleq \forall m. m < len \to \bigvee_{\mathtt{a} \in \Sigma, q \in Q, p \in \delta(q, \mathtt{a})} \left(X_q(m) \land P_{\mathtt{a}}(m+1) \land X_p(m+1)\right)$
  - encoding of intermediate transitions

Linking Run-Variables

for all word lengths *len*, we define:

- $\psi_{setup} \triangleq \forall m. m < len \rightarrow (\bigvee_{q \in Q} X_q(m)) \wedge (\bigwedge_{p \neq q} \neg(X_q(m) \wedge X_p(m)))$
  - reading $m < len$ symbols ends up in a state, and this state is unique
- $\psi_{initial} \triangleq len = 0 \vee \bigvee_{\mathrm{a} \in \Sigma, p \in \delta(q_I, \mathrm{a})} (P_{\mathrm{a}}(0) \wedge X_p(0))$
  - encoding of the initial transition
- $\psi_{run} \triangleq \forall m. m < len \rightarrow \bigvee_{\mathrm{a} \in \Sigma, q \in Q, p \in \delta(q, \mathrm{a})} (X_q(m) \wedge P_{\mathrm{a}}(m+1) \wedge X_p(m+1))$
  - encoding of intermediate transitions
- $\phi_{accept} \triangleq (len = 0 \wedge \ulcorner q_I \in F \urcorner) \vee \exists m. len = m + 1 \wedge \bigvee_{q \in F} (X_q(m))$
  - encoded transition of word $\mathrm{a}_0 \ldots \mathrm{a}_m$ of length $m+1$ lands in a final state

$$\phi_{\mathcal{A}} \triangleq \exists X_{q_1} . \cdots \exists X_{q_n} .$$

$$\forall len. \underbrace{\left(\bigwedge_{\mathrm{a} \in \Sigma} \neg P_{\mathrm{a}}(len) \wedge \forall m. \bigwedge_{\mathrm{a} \in \Sigma} P_{\mathrm{a}}(m) \rightarrow m \leq len \right)}_{len \text{ gives length of input}} \rightarrow \psi_{setup} \wedge \psi_{initial} \wedge \psi_{run} \wedge \psi_{accept}$$

Büchi-Elgot-Trakhtenbrot

Theorem

Let $L \subseteq \Sigma^*$ be a language. The following are equivalent:

- ⋆ *L is regular*
- ⋆ *L is recognizable by a finite automata*
- ⋆ *L is WMSO definable*

Proof Outline.

- ⋆ (1) $\iff$ (2) Kleene's Theorem.
- ⋆ (2) $\implies$ (3) Given an Automata $\mathcal{A}$, we define a WMSO formula $\phi_{\mathcal{A}}$ s.t. $\mathrm{L}(\mathcal{A}) = \mathrm{L}(\phi_{\mathcal{A}})$
  - $\phi_{\mathcal{A}}$ given on previous slide satisfies the case
- ⋆ (3) $\implies$ (1) Given a WMSO formula ϕ, define a regular Language L_ϕ s.t. $\mathrm{L}(\phi) = L_\phi$

Büchi-Elgot-Trakhtenbrot

Theorem

Let $L \subseteq \Sigma^*$ be a language. The following are equivalent:

- ⋆ L *is regular*
- ⋆ L *is recognizable by a finite automata*
- ⋆ L *is WMSO definable*

Proof Outline.

- ⋆ (1) $\Leftrightarrow$ (2) Kleene's Theorem.
- ⋆ (2) $\Rightarrow$ (3) Given an Automata $\mathcal{A}$, we define a WMSO formula $\phi_{\mathcal{A}}$ s.t. $\mathrm{L}(\mathcal{A}) = \mathrm{L}(\phi_{\mathcal{A}})$
  - $\phi_{\mathcal{A}}$ given on previous slide satisfies the case
- ⋆ (3) $\Rightarrow$ (1) Given a WMSO formula ϕ, define a regular Language L_ϕ s.t. $\mathrm{L}(\phi) = L_\phi$

From Formulas to Regular Languages

Encoding for given ϕ over $\mathcal{V}_2 = \{X_1, \ldots, X_m\}$ and $\mathcal{V}_1 = \{y_{m+1}, \ldots, y_{m+n}\}$

- ⋆ the alphabet Σ_ϕ is given by $m + n$ bit-vectors, i.e., $\Sigma_\phi \triangleq \{0, 1\}^{n+m}$

From Formulas to Regular Languages

Encoding for given ϕ over $\mathcal{V}_2 = \{X_1, \ldots, X_m\}$ and $\mathcal{V}_1 = \{y_{m+1}, \ldots, y_{m+n}\}$

- the alphabet Σ_ϕ is given by $m + n$ bit-vectors, i.e., $\Sigma_\phi \triangleq \{0, 1\}^{n+m}$
- word $w \in \Sigma_\phi^*$ can then be seen as a bit-matrix, encoding a valuation α:
  - rows $1 \leq i \leq m$ encode valuations of $X_i \in \mathcal{V}_2$: 1 at column $1 \leq j \leq |w| \iff j \in \alpha(X_i)$
  - rows $m + 1 \leq i \leq m + n$ encode valuations of $y_i \in \mathcal{V}_1$: 1 at column $1 \leq j \leq |w| \iff j = \alpha(y_i)$

$$
\begin{array}{ll}
v & \alpha(v) \\
\hline
X_1 & \{0,2\} \\
X_2 & \{1,3,4\} \\
y_3 & 3 \\
y_4 & 0
\end{array}
\quad \equiv \quad
\begin{array}{ccccc}
w[0] & w[1] & w[2] & w[3] & w[4] \\
\hline
\begin{pmatrix}1\\0\\0\\1\end{pmatrix} & \begin{pmatrix}0\\1\\0\\0\end{pmatrix} & \begin{pmatrix}1\\0\\0\\0\end{pmatrix} & \begin{pmatrix}0\\1\\1\\0\end{pmatrix} & \begin{pmatrix}0\\1\\0\\0\end{pmatrix}
\end{array}
$$

From Formulas to Regular Languages

Encoding for given ϕ over $\mathcal{V}_2 = \{X_1, \ldots, X_m\}$ and $\mathcal{V}_1 = \{y_{m+1}, \ldots, y_{m+n}\}$

- the alphabet Σ_ϕ is given by $m + n$ bit-vectors, i.e., $\Sigma_\phi \triangleq \{0,1\}^{n+m}$
- word $w \in \Sigma_\phi^*$ can then be seen as a bit-matrix, encoding a valuation α:
  - rows $1 \leq i \leq m$ encode valuations of $X_i \in \mathcal{V}_2$: 1 at column $1 \leq j \leq |w| \iff j \in \alpha(X_i)$
  - rows $m+1 \leq i \leq m+n$ encode valuations of $y_i \in \mathcal{V}_1$: 1 at column $1 \leq j \leq |w| \iff j = \alpha(y_i)$

$$\begin{array}{cc} v & \alpha(v) \\ \hline X_1 & \{0,2\} \\ X_2 & \{1,3,4\} \\ y_3 & 3 \\ y_4 & 0 \end{array} \quad \equiv \quad \begin{array}{ccccc} w[0] & w[1] & w[2] & w[3] & w[4] \\ \hline \begin{pmatrix}1\\0\\0\\1\end{pmatrix} & \begin{pmatrix}0\\1\\0\\0\end{pmatrix} & \begin{pmatrix}1\\0\\0\\0\end{pmatrix} & \begin{pmatrix}0\\1\\1\\0\end{pmatrix} & \begin{pmatrix}0\\1\\0\\0\end{pmatrix} \end{array}$$

- for a valuation α for ϕ, let us write $\underline{\alpha} \in \Sigma_\phi^*$ for its encoding

The Main Lemma

let us denote by $\hat{\mathsf{L}}(\phi) \subseteq \Sigma_\phi^*$ the language of coded valuations making ϕ true:

$$\hat{\mathsf{L}}(\phi) \triangleq \{\underline{\alpha} \mid \alpha \models \phi\}$$

The Main Lemma

let us denote by $\hat{\mathsf{L}}(\phi) \subseteq \Sigma_\phi^*$ the language of coded valuations making ϕ true:

$$\hat{\mathsf{L}}(\phi) \triangleq \{\underline{\alpha} \mid \alpha \models \phi\}$$

Lemma

For any WMSO formula ϕ, $\hat{\mathsf{L}}(\phi)$ is regular

The Main Lemma

let us denote by $\hat{\mathsf{L}}(\phi) \subseteq \Sigma_\phi^*$ the language of coded valuations making ϕ true:

$$\hat{\mathsf{L}}(\phi) \triangleq \{\underline{\alpha} \mid \alpha \models \phi\}$$

Lemma

For any WMSO formula ϕ, $\hat{\mathsf{L}}(\phi)$ *is regular*

Proof Outline.

By induction on the structure of ϕ.

The Main Lemma

let us denote by $\hat{\mathsf{L}}(\phi) \subseteq \Sigma_\phi^*$ the language of coded valuations making ϕ true:

$$\hat{\mathsf{L}}(\phi) \triangleq \{\underline{\alpha} \mid \alpha \vDash \phi\}$$

Lemma

For any WMSO formula ϕ, $\hat{\mathsf{L}}(\phi)$ is regular

Proof Outline.

By induction on the structure of ϕ.

- $\phi = \top$, $\phi = \bot$: ?

The Main Lemma

let us denote by $\hat{\mathsf{L}}(\phi) \subseteq \Sigma_\phi^*$ the language of coded valuations making ϕ true:

$$\hat{\mathsf{L}}(\phi) \triangleq \{\underline{\alpha} \mid \alpha \vDash \phi\}$$

Lemma

For any WMSO formula ϕ, $\hat{\mathsf{L}}(\phi)$ is regular

Proof Outline.

By induction on the structure of ϕ.

- ⋆ $\phi = \top$, $\phi = \bot$: In these cases $\hat{\mathsf{L}}(\phi)$ is Σ_ϕ^* or $\varnothing$, thus regular.

The Main Lemma

let us denote by $\hat{\mathsf{L}}(\phi) \subseteq \Sigma_\phi^*$ the language of coded valuations making ϕ true:

$$\hat{\mathsf{L}}(\phi) \triangleq \{\underline{\alpha} \mid \alpha \vDash \phi\}$$

Lemma

For any WMSO formula ϕ, $\hat{\mathsf{L}}(\phi)$ is regular

Proof Outline.

By induction on the structure of ϕ.

- $\phi = \top$, $\phi = \bot$: In these cases $\hat{\mathsf{L}}(\phi)$ is Σ_ϕ^* or $\varnothing$, thus regular.
- $\phi = (x < y)$: ?

The Main Lemma

let us denote by $\hat{\mathsf{L}}(\phi) \subseteq \Sigma_\phi^*$ the language of coded valuations making ϕ true:

$$\hat{\mathsf{L}}(\phi) \triangleq \{\underline{\alpha} \mid \alpha \models \phi\}$$

Lemma

For any WMSO formula ϕ, $\hat{\mathsf{L}}(\phi)$ is regular

Proof Outline.

By induction on the structure of ϕ.

- $\phi = \top$, $\phi = \bot$: In these cases $\hat{\mathsf{L}}(\phi)$ is Σ_ϕ^* or $\varnothing$, thus regular.
- $\phi = (x < y)$: Then $\hat{\mathsf{L}}(\phi) = \binom{0}{0}^* \binom{1}{0} \binom{0}{0}^* \binom{0}{1} \binom{0}{0}$ or $\hat{\mathsf{L}}(\phi) = \binom{0}{0}^* \binom{0}{1} \binom{0}{0}^* \binom{1}{0} \binom{0}{0}$, both of them regular.

The Main Lemma

let us denote by $\hat{\mathsf{L}}(\phi) \subseteq \Sigma_\phi^*$ the language of coded valuations making ϕ true:

$$\hat{\mathsf{L}}(\phi) \triangleq \{\underline{\alpha} \mid \alpha \models \phi\}$$

Lemma

For any WMSO formula ϕ, $\hat{\mathsf{L}}(\phi)$ is regular

Proof Outline.

By induction on the structure of ϕ.

- $\phi = \top$, $\phi = \bot$: In these cases $\hat{\mathsf{L}}(\phi)$ is Σ_ϕ^* or $\varnothing$, thus regular.
- $\phi = (x < y)$: Then $\hat{\mathsf{L}}(\phi) = \binom{0}{0}^* \binom{1}{0} \binom{0}{0}^* \binom{0}{1} \binom{0}{0}$ or $\hat{\mathsf{L}}(\phi) = \binom{0}{0}^* \binom{0}{1} \binom{0}{0}^* \binom{1}{0} \binom{0}{0}$, both of them regular.
- $\phi = X(y)$: ?

The Main Lemma

let us denote by $\hat{\mathsf{L}}(\phi) \subseteq \Sigma_\phi^*$ the language of coded valuations making ϕ true:

$$\hat{\mathsf{L}}(\phi) \triangleq \{\underline{\alpha} \mid \alpha \vDash \phi\}$$

Lemma

For any WMSO formula ϕ, $\hat{\mathsf{L}}(\phi)$ is regular

Proof Outline.

By induction on the structure of ϕ.

- $\phi = \top$, $\phi = \bot$: In these cases $\hat{\mathsf{L}}(\phi)$ is Σ_ϕ^* or $\varnothing$, thus regular.
- $\phi = (x < y)$: Then $\hat{\mathsf{L}}(\phi) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}^* \begin{pmatrix} 1 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \end{pmatrix}^* \begin{pmatrix} 0 \\ 1 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \end{pmatrix}$ or $\hat{\mathsf{L}}(\phi) = \begin{pmatrix} 0 \\ 0 \end{pmatrix}^* \begin{pmatrix} 0 \\ 1 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \end{pmatrix}^* \begin{pmatrix} 1 \\ 0 \end{pmatrix} \begin{pmatrix} 0 \\ 0 \end{pmatrix}$, both of them regular.
- $\phi = X(y)$: Then $\hat{\mathsf{L}}(\phi) = \left(\begin{pmatrix} 0 \\ 0 \end{pmatrix} \cup \begin{pmatrix} 1 \\ 0 \end{pmatrix}\right)^* \begin{pmatrix} 1 \\ 1 \end{pmatrix} \left(\begin{pmatrix} 0 \\ 0 \end{pmatrix} \cup \begin{pmatrix} 1 \\ 0 \end{pmatrix}\right)^*$ is regular.

The Main Lemma

let us denote by $\hat{\mathsf{L}}(\phi) \subseteq \Sigma_\phi^*$ the language of coded valuations making ϕ true:

$$\hat{\mathsf{L}}(\phi) \triangleq \{\underline{\alpha} \mid \alpha \models \phi\}$$

Lemma

For any WMSO formula ϕ, $\hat{\mathsf{L}}(\phi)$ is regular

Proof Outline.

By induction on the structure of ϕ.

- $\phi = \top$, $\phi = \bot$: In these cases $\hat{\mathsf{L}}(\phi)$ is Σ_ϕ^* or $\varnothing$, thus regular.
- $\phi = (x < y)$: Then $\hat{\mathsf{L}}(\phi) = \binom{0}{0}^* \binom{1}{0} \binom{0}{0}^* \binom{0}{1} \binom{0}{0}$ or $\hat{\mathsf{L}}(\phi) = \binom{0}{0}^* \binom{0}{1} \binom{0}{0}^* \binom{1}{0} \binom{0}{0}$, both of them regular.
- $\phi = X(y)$: Then $\hat{\mathsf{L}}(\phi) = \left(\binom{0}{0} \cup \binom{1}{0}\right)^* \binom{1}{1} \left(\binom{0}{0} \cup \binom{1}{0}\right)^*$ is regular.
- $\phi \vee \psi$, $\exists x.\phi$: ?

Homomorphisms

Consider $h : \Sigma \to \Gamma^*$ and extend it to words w by replacing each letter $\mathtt{a}$ in w by $h(w)$:

$$h(\epsilon) \triangleq \epsilon \qquad h(\mathtt{a}w) \triangleq h(\mathtt{a}) \cdot h(w)$$

- ⋆ each function $h : \Sigma^* \to \Gamma^*$ defined this way is called a homomorphism

Homomorphisms

Consider $h : \Sigma \to \Gamma^*$ and extend it to words w by replacing each letter $\mathtt{a}$ in w by $h(w)$:

$$h(\epsilon) \triangleq \epsilon \qquad h(\mathtt{a}w) \triangleq h(\mathtt{a}) \cdot h(w)$$

- each function $h : \Sigma^* \to \Gamma^*$ defined this way is called a homomorphism
- for a language $L \subseteq \Sigma^*$ we let $h(L) \triangleq \{h(w) \mid w \in L\}$ be the application of h to L

Homomorphisms

Consider $h : \Sigma \to \Gamma^*$ and extend it to words w by replacing each letter $\mathtt{a}$ in w by $h(w)$:

$$h(\epsilon) \triangleq \epsilon \qquad h(\mathtt{a}w) \triangleq h(\mathtt{a}) \cdot h(w)$$

- each function $h : \Sigma^* \to \Gamma^*$ defined this way is called a homomorphism
- for a language $L \subseteq \Sigma^*$ we let $h(L) \triangleq \{h(w) \mid w \in L\}$ be the application of h to L
- for a language $L \subseteq \Gamma^*$ we let $h^{-1}(L) \triangleq \{w \mid h(w) \in L\}$ be the inverse application of h to L

Homomorphisms

Consider $h : \Sigma \to \Gamma^*$ and extend it to words w by replacing each letter $\mathtt{a}$ in w by $h(w)$:

$$h(\epsilon) \triangleq \epsilon \qquad h(\mathtt{a}w) \triangleq h(\mathtt{a}) \cdot h(w)$$

- each function $h : \Sigma^* \to \Gamma^*$ defined this way is called a homomorphism
- for a language $L \subseteq \Sigma^*$ we let $h(L) \triangleq \{h(w) \mid w \in L\}$ be the application of h to L
- for a language $L \subseteq \Gamma^*$ we let $h^{-1}(L) \triangleq \{w \mid h(w) \in L\}$ be the inverse application of h to L

Lemma (Closure of $REG(\Sigma)$ under homomorphism)

The set of regular languages is closed under (inverse) applications of homomorphisms.

Example

For $1 \le i \le k$, let $del_{i,k} : \{0,1\}^k \to \{0,1\}^{k-1}$ delete the i-th entry of its argument, e.g.,

$$del_{1,3}\left(\begin{pmatrix} a \\ b \\ c \end{pmatrix}\right) \triangleq \begin{pmatrix} b \\ c \end{pmatrix} \qquad del_{2,3}\left(\begin{pmatrix} a \\ b \\ c \end{pmatrix}\right) \triangleq \begin{pmatrix} a \\ c \end{pmatrix} \qquad del_{3,3}\left(\begin{pmatrix} a \\ b \\ c \end{pmatrix}\right) \triangleq \begin{pmatrix} a \\ b \end{pmatrix}$$

Example

For $1 \leq i \leq k$, let $del_{i,k} : \{0,1\}^k \to \{0,1\}^{k-1}$ delete the i-th entry of its argument, e.g.,

$$del_{1,3}\left(\begin{pmatrix} a \\ b \\ c \end{pmatrix}\right) \triangleq \begin{pmatrix} b \\ c \end{pmatrix} \qquad del_{2,3}\left(\begin{pmatrix} a \\ b \\ c \end{pmatrix}\right) \triangleq \begin{pmatrix} a \\ c \end{pmatrix} \qquad del_{3,3}\left(\begin{pmatrix} a \\ b \\ c \end{pmatrix}\right) \triangleq \begin{pmatrix} a \\ b \end{pmatrix}$$

and thus

$$del_{1,3}\left(\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}^*\right) = \begin{pmatrix} 1 \\ 0 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix}^* \qquad del_{1,3}^{-1}\left(\begin{pmatrix} 1 \\ 0 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix}^*\right) = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}^* \cup \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}^* \cup \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}^* \cup \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}^*$$

Example

For $1 \leq i \leq k$, let $del_{i,k} : \{0,1\}^k \to \{0,1\}^{k-1}$ delete the i-th entry of its argument, e.g.,

$$del_{1,3}\left(\begin{pmatrix} a \\ b \\ c \end{pmatrix}\right) \triangleq \begin{pmatrix} b \\ c \end{pmatrix} \qquad del_{2,3}\left(\begin{pmatrix} a \\ b \\ c \end{pmatrix}\right) \triangleq \begin{pmatrix} a \\ c \end{pmatrix} \qquad del_{3,3}\left(\begin{pmatrix} a \\ b \\ c \end{pmatrix}\right) \triangleq \begin{pmatrix} a \\ b \end{pmatrix}$$

and thus

$$del_{1,3}\left(\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}^*\right) = \begin{pmatrix} 1 \\ 0 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix}^* \qquad del_{1,3}^{-1}\left(\begin{pmatrix} 1 \\ 0 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix}^*\right) = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}^* \cup \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}^* \cup \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}^* \cup \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}^*$$

Concretely, for WMSO formulas ϕ over $\mathcal{V}_2 = \{X_1, \ldots, X_m\}$, $\mathcal{V}_1 = \{y_{m+1}, \ldots, y_{m+n}\}$:

Example

For $1 \leq i \leq k$, let $del_{i,k} : \{0,1\}^k \to \{0,1\}^{k-1}$ delete the i-th entry of its argument, e.g.,

$$del_{1,3}\left(\begin{pmatrix} a \\ b \\ c \end{pmatrix}\right) \triangleq \begin{pmatrix} b \\ c \end{pmatrix} \qquad del_{2,3}\left(\begin{pmatrix} a \\ b \\ c \end{pmatrix}\right) \triangleq \begin{pmatrix} a \\ c \end{pmatrix} \qquad del_{3,3}\left(\begin{pmatrix} a \\ b \\ c \end{pmatrix}\right) \triangleq \begin{pmatrix} a \\ b \end{pmatrix}$$

and thus

$$del_{1,3}\left(\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}^*\right) = \begin{pmatrix} 1 \\ 0 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix}^* \qquad del_{1,3}^{-1}\left(\begin{pmatrix} 1 \\ 0 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix}^*\right) = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}^* \cup \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}^* \cup \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}^* \cup \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}^*$$

Concretely, for WMSO formulas ϕ over $\mathcal{V}_2 = \{X_1, \ldots, X_m\}$, $\mathcal{V}_1 = \{y_{m+1}, \ldots, y_{m+n}\}$:

- ★ for $1 \leq i \leq n$, $del_{i,n+m}(\hat{\mathrm{L}}(\phi)) = del_{i,n+m}(\{\underline{\alpha} \mid \alpha \vDash \phi\})$
  $\approx \{\underline{\beta} \mid \beta[X_i \mapsto S] \vDash \phi \text{ for some } S \subseteq \mathbb{N}\} = \hat{\mathrm{L}}(\exists X_i.\phi)$

Example

For $1 \leq i \leq k$, let $del_{i,k} : \{0,1\}^k \to \{0,1\}^{k-1}$ delete the i-th entry of its argument, e.g.,

$$del_{1,3}\left(\begin{pmatrix} a \\ b \\ c \end{pmatrix}\right) \triangleq \begin{pmatrix} b \\ c \end{pmatrix} \qquad del_{2,3}\left(\begin{pmatrix} a \\ b \\ c \end{pmatrix}\right) \triangleq \begin{pmatrix} a \\ c \end{pmatrix} \qquad del_{3,3}\left(\begin{pmatrix} a \\ b \\ c \end{pmatrix}\right) \triangleq \begin{pmatrix} a \\ b \end{pmatrix}$$

and thus

$$del_{1,3}\left(\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}^*\right) = \begin{pmatrix} 1 \\ 0 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix}^* \qquad del_{1,3}^{-1}\left(\begin{pmatrix} 1 \\ 0 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix}^*\right) = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}^* \cup \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}^* \cup \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}^* \cup \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}^*$$

Concretely, for WMSO formulas ϕ over $\mathcal{V}_2 = \{X_1, \ldots, X_m\}$, $\mathcal{V}_1 = \{y_{m+1}, \ldots, y_{m+n}\}$:

- for $1 \leq i \leq n$, $del_{i,n+m}(\hat{\mathrm{L}}(\phi)) = del_{i,n+m}(\{\underline{\alpha} \mid \alpha \vDash \phi\})$
  $$\approx \{\underline{\beta} \mid \beta[X_i \mapsto S] \vDash \phi \text{ for some } S \subseteq \mathbb{N}\} = \hat{\mathrm{L}}(\exists X_i.\phi)$$
- inversely, $del_{i,1+n+m}^{-1}(\hat{\mathrm{L}}(\phi)) = \{\underline{\alpha[X \mapsto S]} \mid \alpha \vDash \phi \text{ and } S \subseteq \mathbb{N}\}$ extends valid assignments

Example

For $1 \le i \le k$, let $del_{i,k} : \{0,1\}^k \to \{0,1\}^{k-1}$ delete the i-th entry of its argument, e.g.,

$$del_{1,3}\left(\begin{pmatrix} a \\ b \\ c \end{pmatrix}\right) \triangleq \begin{pmatrix} b \\ c \end{pmatrix} \qquad del_{2,3}\left(\begin{pmatrix} a \\ b \\ c \end{pmatrix}\right) \triangleq \begin{pmatrix} a \\ c \end{pmatrix} \qquad del_{3,3}\left(\begin{pmatrix} a \\ b \\ c \end{pmatrix}\right) \triangleq \begin{pmatrix} a \\ b \end{pmatrix}$$

and thus

$$del_{1,3}\left(\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}^*\right) = \begin{pmatrix} 1 \\ 0 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix}^* \qquad del_{1,3}^{-1}\left(\begin{pmatrix} 1 \\ 0 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix}^*\right) = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}^* \cup \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}^* \cup \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}^* \cup \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}^*$$

Concretely, for WMSO formulas ϕ over $\mathcal{V}_2 = \{X_1, \ldots, X_m\}$, $\mathcal{V}_1 = \{y_{m+1}, \ldots, y_{m+n}\}$:

- for $1 \le i \le n$, $del_{i,n+m}(\hat{\mathrm{L}}(\phi)) = del_{i,n+m}(\{\underline{\alpha} \mid \alpha \models \phi\})$
  $\approx \{\underline{\beta} \mid \beta[X_i \mapsto S] \models \phi \text{ for some } S \subseteq \mathbb{N}\} = \hat{\mathrm{L}}(\exists X_i.\phi)$
- inversely, $del^{-1}_{i,1+n+m}(\hat{\mathrm{L}}(\phi)) = \{\underline{\alpha[X \mapsto S]} \mid \alpha \models \phi \text{ and } S \subseteq \mathbb{N}\}$ extends valid assignments
- similar for first order variables y_i $(m + 1 \le i \le m + n)$

Example

For $1 \le i \le k$, let $del_{i,k} : \{0,1\}^k \to \{0,1\}^{k-1}$ delete the i-th entry of its argument, e.g.,

$$del_{1,3}\left(\begin{pmatrix} a \\ b \\ c \end{pmatrix}\right) \triangleq \begin{pmatrix} b \\ c \end{pmatrix} \qquad del_{2,3}\left(\begin{pmatrix} a \\ b \\ c \end{pmatrix}\right) \triangleq \begin{pmatrix} a \\ c \end{pmatrix} \qquad del_{3,3}\left(\begin{pmatrix} a \\ b \\ c \end{pmatrix}\right) \triangleq \begin{pmatrix} a \\ b \end{pmatrix}$$

and thus

$$del_{1,3}\left(\begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}^*\right) = \begin{pmatrix} 1 \\ 0 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix}^* \qquad del_{1,3}^{-1}\left(\begin{pmatrix} 1 \\ 0 \end{pmatrix}\begin{pmatrix} 0 \\ 1 \end{pmatrix}^*\right) = \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}^* \cup \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}^* \cup \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}\begin{pmatrix} 0 \\ 0 \\ 1 \end{pmatrix}^* \cup \begin{pmatrix} 1 \\ 1 \\ 0 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \\ 1 \end{pmatrix}^*$$

Concretely, for WMSO formulas ϕ over $\mathcal{V}_2 = \{X_1, \ldots, X_m\}$, $\mathcal{V}_1 = \{y_{m+1}, \ldots, y_{m+n}\}$:

- ⋆ for $1 \le i \le n$, $del_{i,n+m}(\hat{L}(\phi)) = del_{i,n+m}(\{\underline{\alpha} \mid \alpha \models \phi\})$
  $\approx \{\underline{\beta} \mid \beta[X_i \mapsto S] \models \phi \text{ for some } S \subseteq \mathbb{N}\} = \hat{L}(\exists X_i.\phi)$
- ⋆ inversely, $del_{i,1+n+m}^{-1}(\hat{L}(\phi)) = \{\underline{\alpha[X \mapsto S]} \mid \alpha \models \phi \text{ and } S \subseteq \mathbb{N}\}$ extends valid assignments
- ⋆ similar for first order variables y_i $(m+1 \le i \le m+n)$
- ⋆ Attention: One has to be slightly more careful with codings.

$$\phi \rightsquigarrow \begin{matrix} X \\ Y \end{matrix}\begin{pmatrix} a_1 \\ b_1 \end{pmatrix}\cdots\begin{pmatrix} a_n \\ b_n \end{pmatrix}\begin{pmatrix} a_{n+1} \\ 1 \end{pmatrix}\begin{pmatrix} 1 \\ 0 \end{pmatrix} \qquad \exists X.\phi \rightsquigarrow (\, b_1 \,)\cdots(\, b_n \,)(\, 1 \,)(\, 0 \,)$$

The Main Lemma (Continued)

Lemma

For any WMSO formula ϕ, $\hat{\mathsf{L}}(\phi)$ is regular

Proof Outline.

- $\phi = \psi_1 \vee \psi_2$:
  - by induction hypothesis, $L_1 \triangleq \hat{\mathsf{L}}(\psi_1)$ and $L_2 \triangleq \hat{\mathsf{L}}(\psi_2)$ are regular
  - L_1 and L_2 speak about assignments to variables in ψ_1 and ψ_2
  - inverse applications of $del_{i,*}$ extends these codings to valuations over $\mathsf{fv}(\psi_1 \vee \psi_2)$
  - their union yields $\hat{\mathsf{L}}(\psi_1 \vee \psi_2)$ and is thus regular

The Main Lemma (Continued)

Lemma

For any WMSO formula ϕ, $\hat{\mathsf{L}}(\phi)$ is regular

Proof Outline.

- $\phi = \psi_1 \vee \psi_2$:
  - by induction hypothesis, $L_1 \triangleq \hat{\mathsf{L}}(\psi_1)$ and $L_2 \triangleq \hat{\mathsf{L}}(\psi_2)$ are regular
  - L_1 and L_2 speak about assignments to variables in ψ_1 and ψ_2
  - inverse applications of $del_{i,*}$ extends these codings to valuations over $\mathrm{fv}(\psi_1 \vee \psi_2)$
  - their union yields $\hat{\mathsf{L}}(\psi_1 \vee \psi_2)$ and is thus regular
- $\phi = \neg\psi$: Then $\hat{\mathsf{L}}(\phi) = \overline{\hat{\mathsf{L}}(\psi)} \cap L_{valid}$.
  - $L_{valid} \in REG$ constrains Σ_ϕ to valid codings (e.g., for FO variables, only one bit is set)
  - by induction hypothesis and closure properties of REG, $\hat{\mathsf{L}}(\phi)$ is valid

The Main Lemma (Continued)

Lemma

For any WMSO formula ϕ, $\hat{\mathsf{L}}(\phi)$ is regular

Proof Outline.

- ⋆ $\phi = \psi_1 \vee \psi_2$:
  - by induction hypothesis, $L_1 \triangleq \hat{\mathsf{L}}(\psi_1)$ and $L_2 \triangleq \hat{\mathsf{L}}(\psi_2)$ are regular
  - L_1 and L_2 speak about assignments to variables in ψ_1 and ψ_2
  - inverse applications of $del_{i,*}$ extends these codings to valuations over $\mathsf{fv}(\psi_1 \vee \psi_2)$
  - their union yields $\hat{\mathsf{L}}(\psi_1 \vee \psi_2)$ and is thus regular
- ⋆ $\phi = \neg\psi$: Then $\hat{\mathsf{L}}(\phi) = \overline{\hat{\mathsf{L}}(\psi)} \cap L_{valid}$.
  - $L_{valid} \in REG$ constrains Σ_ϕ to valid codings (e.g., for FO variables, only one bit is set)
  - by induction hypothesis and closure properties of REG, $\hat{\mathsf{L}}(\phi)$ is valid
- ⋆ $\phi = \exists X_i.\psi$ or $\phi = \exists y_j.\psi$: from induction hypothesis, using homomorphism $del_{i,*}$ to drop the rows referring to X_i or y_j; taking care of trailing zero-vectors (see previous slide)

Büchi-Elgot-Trakhtenbrot

Theorem

Let $L \subseteq \Sigma^*$ be a language. The following are equivalent:

- L *is regular*
- L *is recognizable by a finite automata*
- L *is WMSO definable*

Proof Outline.

- (1) $\Longleftrightarrow$ (2) Kleene's Theorem.
- (2) $\Longrightarrow$ (3) Given an Automata $\mathcal{A}$, we define a WMSO formula $\phi_{\mathcal{A}}$ s.t. $\mathrm{L}(\mathcal{A}) = \mathrm{L}(\phi_{\mathcal{A}})$
- (3) $\Longrightarrow$ (1) Given a WMSO formula ϕ, define a regular Language L_ϕ s.t. $\mathrm{L}(\phi) = L_\phi$
  - we can define a homomorphism $h : \{0,1\}^{|\Sigma|} \to \Sigma$, and thereby a function from codings $\underline{\alpha}$ to words w
  - this homomorphism maps $\hat{\mathrm{L}}(\phi)$ to $\mathrm{L}(\phi)$ (how?)

Büchi-Elgot-Trakhtenbrot

Theorem

Let $L \subseteq \Sigma^*$ be a language. The following are equivalent:

- ⋆ *L is regular*
- ⋆ *L is recognizable by a finite automata*
- ⋆ *L is WMSO definable*

Proof Outline.

- ⋆ (1) $\Longleftrightarrow$ (2) Kleene's Theorem.
- ⋆ (2) $\Longrightarrow$ (3) Given an Automata $\mathcal{A}$, we define a WMSO formula $\phi_{\mathcal{A}}$ s.t. $\mathrm{L}(\mathcal{A}) = \mathrm{L}(\phi_{\mathcal{A}})$
- ⋆ (3) $\Longrightarrow$ (1) Given a WMSO formula ϕ, define a regular Language L_ϕ s.t. $\mathrm{L}(\phi) = L_\phi$
  - we can define a homomorphism $h : \{0, 1\}^{|\Sigma|} \rightarrow \Sigma$, and thereby a function from codings $\underline{\alpha}$ to words w
  - this homomorphism maps $\hat{\mathrm{L}}(\phi)$ to $\mathrm{L}(\phi)$ (how?)
  - as the former is regular and $REG(\Sigma)$ closed under homomorphisms, the direction follows

Decision Problems

Decision Problems for WMSO

The Satisfiability Problem

- Given: WMSO formula ϕ
- Question: is there α s.t $\alpha \models \phi$?

The Validity Problem

- Given: WMSO formula ϕ
- Question: $\alpha \models \phi$ for all assignments α?

Decision Problems for WMSO

The Satisfiability Problem

- Given: WMSO formula ϕ
- Question: is there α s.t $\alpha \models \phi$?

The Validity Problem

- Given: WMSO formula ϕ
- Question: $\alpha \models \phi$ for all assignments α?

Theorem

Satisfiability and Validity are decidable for WMSO.

Proof Outline.

through the construction of corresponding DFAs, checking emptiness

Complexity

- Emptiness for an DFA $\mathcal{A}_\phi$ is in PTIME (in the number $|\mathcal{A}_\phi|$ of nodes of $\mathcal{A}_\phi$)
- the complexity of satisfiability/validity thus essentially depends on the size of $\mathcal{A}_\phi$

Complexity

- Emptiness for an DFA $\mathcal{A}_\phi$ is in PTIME (in the number $|\mathcal{A}_\phi|$ of nodes of $\mathcal{A}_\phi$)
- the complexity of satisfiability/validity thus essentially depends on the size of $\mathcal{A}_\phi$
- $\mathcal{A}_\phi$ is constructed recursively on the structure of ϕ

Complexity

- Emptiness for an DFA $\mathcal{A}_\phi$ is in PTIME (in the number $|\mathcal{A}_\phi|$ of nodes of $\mathcal{A}_\phi$)
- the complexity of satisfiability/validity thus essentially depends on the size of $\mathcal{A}_\phi$
- $\mathcal{A}_\phi$ is constructed recursively on the structure of ϕ
  - base cases $\phi = \top, \bot, (x < y), X(y)$: DFAs of constant size $O(1)$

Complexity

- ⋆ Emptiness for an DFA $\mathcal{A}_\phi$ is in PTIME (in the number $|\mathcal{A}_\phi|$ of nodes of $\mathcal{A}_\phi$)
- ⋆ the complexity of satisfiability/validity thus essentially depends on the size of $\mathcal{A}_\phi$
- ⋆ $\mathcal{A}_\phi$ is constructed recursively on the structure of ϕ
  - base cases $\phi = \top, \bot, (x < y), X(y)$: DFAs of constant size $O(1)$
  - disjunction $\phi = \psi_1 \vee \psi_2$: $\mathcal{A}_\phi$ DFA-union of $\mathcal{A}_{\psi_1}$ and $\mathcal{A}_{\psi_2}$ $O(|\mathcal{A}_{\psi_1}| + |\mathcal{A}_{\psi_2}|)$

Complexity

- Emptiness for an DFA $\mathcal{A}_\phi$ is in PTIME (in the number $|\mathcal{A}_\phi|$ of nodes of $\mathcal{A}_\phi$)
- the complexity of satisfiability/validity thus essentially depends on the size of $\mathcal{A}_\phi$
- $\mathcal{A}_\phi$ is constructed recursively on the structure of ϕ
  - base cases $\phi = \top, \bot, (x < y), X(y)$: DFAs of constant size $O(1)$
  - disjunction $\phi = \psi_1 \vee \psi_2$: $\mathcal{A}_\phi$ DFA-union of $\mathcal{A}_{\psi_1}$ and $\mathcal{A}_{\psi_2}$ $O(|\mathcal{A}_{\psi_1}| + |\mathcal{A}_{\psi_2}|)$
  - negations $\phi = \neg\psi$: $\mathcal{A}_\phi$ DFA-complement of $\mathcal{A}_\psi$ $O(|\mathcal{A}_\psi|)$

Complexity

- Emptiness for an DFA $\mathcal{A}_\phi$ is in PTIME (in the number $|\mathcal{A}_\phi|$ of nodes of $\mathcal{A}_\phi$)
- the complexity of satisfiability/validity thus essentially depends on the size of $\mathcal{A}_\phi$
- $\mathcal{A}_\phi$ is constructed recursively on the structure of ϕ
  - base cases $\phi = \top, \bot, (x < y), X(y)$: DFAs of constant size — $O(1)$
  - disjunction $\phi = \psi_1 \vee \psi_2$: $\mathcal{A}_\phi$ DFA-union of $\mathcal{A}_{\psi_1}$ and $\mathcal{A}_{\psi_2}$ — $O(|\mathcal{A}_{\psi_1}| + |\mathcal{A}_{\psi_2}|)$
  - negations $\phi = \neg\psi$: $\mathcal{A}_\phi$ DFA-complement of $\mathcal{A}_\psi$ — $O(|\mathcal{A}_\psi|)$
  - existentials $\phi = \exists x.\psi$ or $\phi = \exists X.\psi$: homomorphism application and determinisation — $2^{|\mathcal{A}_\psi|}$

Complexity

- Emptiness for an DFA $\mathcal{A}_\phi$ is in PTIME (in the number $|\mathcal{A}_\phi|$ of nodes of $\mathcal{A}_\phi$)
- the complexity of satisfiability/validity thus essentially depends on the size of $\mathcal{A}_\phi$
- $\mathcal{A}_\phi$ is constructed recursively on the structure of ϕ
  - base cases $\phi = \top, \bot, (x < y), X(y)$: DFAs of constant size $O(1)$
  - disjunction $\phi = \psi_1 \vee \psi_2$: $\mathcal{A}_\phi$ DFA-union of $\mathcal{A}_{\psi_1}$ and $\mathcal{A}_{\psi_2}$ $O(|\mathcal{A}_{\psi_1}| + |\mathcal{A}_{\psi_2}|)$
  - negations $\phi = \neg\psi$: $\mathcal{A}_\phi$ DFA-complement of $\mathcal{A}_\psi$ $O(|\mathcal{A}_\psi|)$
  - existentials $\phi = \exists x.\psi$ or $\phi = \exists X.\psi$: homomorphism application and determinisation $2^{|\mathcal{A}_\psi|}$

Theorem (Hardness)

Satisfiability and validity are in DTIME$(2^c_{O(n)})$, *where* 2^c_k *is a tower of exponentials* $2^{2^{\cdot^{\cdot^{\cdot^{2^c}}}}}$ *of height* k.

Complexity

- Emptiness for an DFA $\mathcal{A}_\phi$ is in PTIME (in the number $|\mathcal{A}_\phi|$ of nodes of $\mathcal{A}_\phi$)
- the complexity of satisfiability/validity thus essentially depends on the size of $\mathcal{A}_\phi$
- $\mathcal{A}_\phi$ is constructed recursively on the structure of ϕ
  - base cases $\phi = \top$, $\bot$, $(x < y)$, $X(y)$: DFAs of constant size $O(1)$
  - disjunction $\phi = \psi_1 \vee \psi_2$: $\mathcal{A}_\phi$ DFA-union of $\mathcal{A}_{\psi_1}$ and $\mathcal{A}_{\psi_2}$ $O(|\mathcal{A}_{\psi_1}| + |\mathcal{A}_{\psi_2}|)$
  - negations $\phi = \neg\psi$: $\mathcal{A}_\phi$ DFA-complement of $\mathcal{A}_\psi$ $O(|\mathcal{A}_\psi|)$
  - existentials $\phi = \exists x.\psi$ or $\phi = \exists X.\psi$: homomorphism application and determinisation $2^{|\mathcal{A}_\psi|}$

Theorem (Hardness)

Satisfiability and validity are in DTIME$(2^c_{O(n)})$, *where* 2^c_k *is a tower of exponentials* $2^{2^{\cdot^{\cdot^{\cdot^{2^c}}}}}$ *of height* k.

Theorem (Completeness)

Any language L *decidable in time* DTIME$(2^c_{O(n)})$ *can be reduced (within polynomial time) to the satisfiability of formulas* ϕ_w $(w \in L)$ *of size polynomial in* $|w|$.

WMSO and Alternating Finite Automata

- What if we translate WMSO formulas to AFAs?
  - for basic formulas $x < y$ and $X(y)$, the construction is as seen previously
  - Boolean connectives are reflected directly in the transition
  - Quantifier elimination through projection homomorphisms

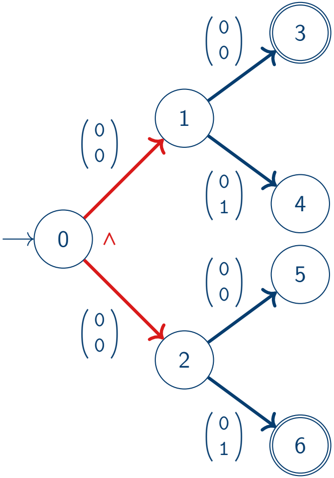
WMSO and Alternating Finite Automata

- What if we translate WMSO formulas to AFAs?
  - for basic formulas $x < y$ and $X(y)$, the construction is as seen previously
  - Boolean connectives are reflected directly in the transition
  - Quantifier elimination through projection homomorphisms

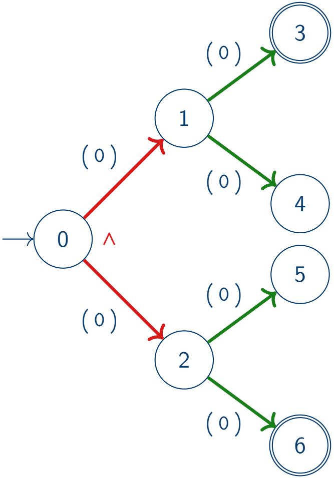
- this suggests resulting automaton is linear in size of formula

$\Longrightarrow$ WMSO model-checking in exponential time, contradicting the lower-bound result!

Projections and AFAs



$\Longrightarrow$



$L(\mathcal{A}) = \varnothing$

$L(\mathrm{del}_{2,2}(\mathcal{A})) = \{00\}$

WMSO and Alternating Finite Automata

- What if we translate WMSO formulas to AFAs?
  - for basic formulas $x < y$ and $X(y)$, the construction is as seen previously
  - Boolean connectives are reflected directly in the transition
  - Quantifier elimination through projection homomorphisms

- this suggests resulting automaton is linear in size of formula

  $\Rightarrow$ WMSO model-checking in exponential time, contradicting the lower-bound result!

Problem:

We do not have a polytime algorithm for homorphism applications on AFAs