KAIST, School of Computing, Spring 2025 — **Scribed By: Eunjung KIM**
Graph classes, algorithms and logic (CS492) — **Introduction, MSO logic, regular language**
Lecture: Eunjung KIM — **Week 1: 25, 27 February 2025**

Contents

---

1 MSO and FO logic

We mostly following the notations from the textbook by Libkin [1].

1.1 Logic and relational structure

Monadic Second Order Logic, MSO-expressible property. A *vocabulary* τ is a set of relation names, or *predicates*, and *constant symbols*. Each predicate $R \in \tau$ is associated with a number $\mathsf{ar}(R) \in \mathbb{N}$, called the *arity* of R.

Let τ be a vocabulary. We assume that there is an infinite supply of symbols, all distinct from symbols in τ, for *individual variables* and for *unary relation variables* (a.k.a. *set variables*). We typically use a lowercase to denote an individual variable and an uppercase letter for a set variable.

A *formula in monadic second-order logic*, simply put mso-formula, over a vocabulary τ, is a string that can be built recursively using the logical connectives $\neg, \wedge, \vee, \rightarrow$, the *quantifiers* $\forall, \exists$ as well as the predicates in τ. A string in the form $x = y$, $X(x)$ for some set variable X or $R(x_1, \ldots, x_{\mathsf{ar}(R)})$ for some predicate $R \in \tau$ is an *atomic formula*. For two formulae ψ and ϕ, the strings $\neg\psi$, $\psi \wedge \phi$, $\psi \vee \phi$ and $\psi \rightarrow \phi$ are formulae. For a formula ψ, and for an individual variable x and a set variable X, $\forall x\psi$, $\exists x\psi$, $\forall X\psi$ and $\exists X\psi$ are all formulae. A variable in a formula ψ is *free* if it does not appear next to a quantifier. We often write the free variables of a formula ψ inside a parenthesis, e.g., $\psi(x, y, Z)$, to highlight[1] the free variables of ψ. A formula without a free variable is called a *sentence* and a formula without a quantifier is said to be *quantifier-free*. A formula is in *prenex normal form* if it is in the form $Q_1x_1 \cdots Q_\ell x_\ell\psi$ such that Q_i is either the universal quantifier $\forall$ or the existential quantifier $\exists$, x_i is an individual or a set variable, and ψ is quantifier-free.

Relational structure. Let τ be a vocabulary. For a set U, called a *universe of discourse* or simply a *universe*, an *interpretation of a predicate $R \in \tau$ in the universe U of discourse* is a subset of $U^{\mathsf{ar}}(R)$. In general, an interpretation of a symbol (it may be a predicate, a constant in the vocabulary, a variable be it a set or individual variable...) is to *give a meaning to the symbol in the universe of discourse by associating with the symbol a suitable tuple of elements from the universe.*

[1]The free variables of ψ is a subset of those in the parenthesis.

A *relational structure over* τ, or τ-structure, is a tuple $\mathbb{A} = (A, (R^{\mathbb{A}})_{R\in\tau})$ consisting of

- a *universe* A,
- an interpretation $R^{\mathbb{A}} \subseteq A^{\mathsf{ar}(R)}$ for each predicate $R \in \tau$, and
- an interpretation $c^{\mathbb{A}} \in A$ for each constant symbol $c \in \tau$.

Evaluating a formula. Fix a vocabulary τ and let $\mathbb{A}$ be a τ-structure on the universe U. An *interpretation* of an mso-formula $\psi(x_1, \dots, x_k, X_1, \dots, X_\ell)$ in $\mathbb{A}$ is an assignment s of variables in ψ (free variables and quantified variables) to an element of A, in the case of an individual variable, or to a subset of A in the case of a set variable. For a k-tuple $\vec{v} = (v_1, \dots, v_k) \in A^k$ of elements and an ℓ-tuple $\vec{V} = (V_1, \dots, V_\ell) \in (2^A)^\ell$ of sets, we write

$$\mathbb{A} \models \psi(v_1, \dots, v_k, V_1, \dots, V_\ell) \text{ or } \mathbb{A} \models \psi(\vec{v}, \vec{V})$$

if $\psi(\vec{x}, \vec{X})$ *evaluates to* TRUE ("$\psi(\vec{x}, \vec{X})$ holds on $\mathbb{A}$") when the variables x_i's are interpreted as v_i's and X_j's interpreted as V_j's. The *value* $\psi(\vec{v}, \vec{V})$ of a formula $\psi(x_1, \dots, x_k, X_1, \dots, X_\ell)$ under an interpretation s of its free variables as $(\vec{v}, \vec{V})$ is defined as follows.

- Atomic formula $X(x)$: note that x is an individual variable and X is a set variable here. Evaluates to TRUE if and only if $s(x) \in s(X)$.
- Atomic formula $x = y$: evaluates to TRUE if and only if $s(x) = s(y)$
- Atomic formula $R(x_1, \dots, x_{\mathsf{ar}(R)})$ for some $R \in \tau$: evaluates to TRUE if and only if $(s(x_1), \dots, s(x_{\mathsf{ar}(R)})) \in R^{\mathbb{A}}$.
- $\psi(\vec{x}, \vec{X}) = \neg\varphi(\vec{x}, \vec{X})$: flips the value of $\varphi(\vec{v}, \vec{V})$
- $\psi(\vec{x}, \vec{X}) = \psi_1(\vec{x}, \vec{X}) \wedge \psi_2(\vec{x}, \vec{X})$: the usual boolean operation on the values of $\psi_1(\vec{x}, \vec{X}) \wedge \psi_2(\vec{x}, \vec{X})$
- $\psi(\vec{x}, \vec{X}) = \exists x \varphi(x, \vec{x}, \vec{X})$: $\mathbb{A} \models \psi(\vec{v}, \vec{V})$ if and only if there exists $v \in A$ (or $X \subseteq A$) such that $\mathbb{A} \models \varphi(v, \vec{v}, \vec{V})$.
- $\psi(\vec{x}, \vec{X}) = \forall x \varphi(x, \vec{x}, \vec{X})$: $\mathbb{A} \models \psi(\vec{v}, \vec{V})$ if and only if for every $v \in A$ (or $X \subseteq A$) it holds that $\mathbb{A} \models \varphi(v, \vec{v}, \vec{V})$.

A τ-structure $\mathbb{A}$ *models* an mso-sentence ψ (or equivalently *satisfies* ψ, or ψ holds on $\mathbb{A}$) if $\mathbb{A} \models \psi$.

First Order logic. First order logic is monadic second order logic without set quantification.

1.2 Graphs and strings as relational structures

Graph as a relational structure. Of special interest is the vocabulary $\{\mathsf{edge}\}$, consisting of a single predicate edge of arity 2. An undirected graph $G = (V, E)$ can be represented as the $\{\mathsf{edge}\}$-structure $\mathbb{G} = (V, \mathsf{edge}^{\mathbb{G}})$ over $\{\mathsf{edge}\}$, where the vertex set V of G is the universe and for every $(u, v) \in V \times V$, $\mathsf{edge}^{\mathbb{G}}(u, v)$ if and only if uv is an edge of G. Another way to represent a graph $G = (V, E)$ as a relational structure is to represent its incidence graph. We shall denote the associated vocabulary $\tau = \{\mathsf{vtx}, \mathsf{edg}, \mathsf{inc}\}$, where vtx and edg are unary predicates and inc is a binary predicate. For a graph $G = (V, E)$, we associate the τ-structure $\mathbb{G} = (V \cap E, \mathsf{vtx}^{\mathbb{G}}, \mathsf{edg}^{\mathbb{G}}, \mathsf{inc}^{\mathbb{G}})$ where $\mathsf{vtx}^G = V$, $\mathsf{edg}^{\mathbb{G}} = E$ and $\mathsf{inc}^G = \{(v, e) \in V \times E \mid$

v is an endpoint of e in $G\}$. Such a τ-structure is a canonical representation of an incidence graph of G as a relational structure.

We say that a graph property $\mathcal{F}$ can be expressed in MSO, or MSO-*expressible* in short, if there is an mso-sentence ψ over $\{\mathsf{edge}\}$ such that $G \in \mathcal{F}$ if and only if $\mathbb{G} \models \psi$ where $\mathbb{G}$ is a $\{\mathsf{edge}\}$-structure. An MSO-expressible property is defined similarly, for which we take the vocabulary $\tau = \{\mathsf{vtx}, \mathsf{edg}, \mathsf{inc}\}$ in the place of $\{\mathsf{edge}\}$. A sentence over $\{\mathsf{edge}\}$, respectively over τ, is called an MSO-sentence, respectively an MSO-sentence, on graphs.

In this paper, we are mostly interested in an MSO or MSO-sentence as a graph property. By abusing the notation, we often write $G \models \psi$ for a graph and an MSO-sentence, respectively MSO-sentence, ψ as a shortcut to state the following: $\mathbb{G} \models \psi$ for an τ-structure and a sentence over τ, where $\tau = \{\mathsf{edge}\}$, respectively $\tau = \{\mathsf{vtx}, \mathsf{edg}, \mathsf{inc}\}$.

Definable graph property. We are often interested in graphs or strings (or other relational structures) with specific property. A set of graphs is called a *graph property*. A graph property Π is MSO-definable or MSO-expressible (MSO_2-definable) if there is an MSO-sentence ψ on graphs such that for every graph G

$$G \in \Pi \text{ if and only if } G \models \psi.$$

String as a relational structure. An *alphabet* is a set of symbols. A *string (word) over* τ is a finite sequence of symbols in τ. The string of length 0 (over any alphabet) is denoted by ϵ. A *language* L *over the alphabet* Σ is a set of strings over Σ, i.e. $K \subseteq \Sigma^*$. We restrict ourselves to finite strings over finite alphabet throughout this lecture note.

Fix a finite alphabet Σ. The *vocabulary for string over* Σ, τ_Σ, consists of the following predicates:

- (Linear order) Binary relation $<$; this predicate is to used to express "position i precedes position j in the string".
- (Symbol $a \in \Sigma$) Unary relation P_a for each $a \in \Sigma$; this predicate is used to express "The i-th position in the string carries the symbol a".

A string $w = w_1 \cdots w_n$ over Σ is seen as a τ_Σ-structure, obtained by interpreting the predicates in τ_Σ in the universe $\{1, \ldots, n\}$ as follows.

- Binary relation $<$; in the usual way.
- Unary relation P_a for each $a \in \Sigma$; $P_a = \{i \in [n] \mid w_i = a\}$.

Just like graphs, a set of strings (that is, a language) may be expressible in logic. We say that a language $L \subseteq \Sigma^*$ is MSO-definable, or equivalently MSO-expressible, if there is an MSO-sentence ψ on strings such that for every string s over Σ,

$$s \in \Pi \text{ if and only if } s \models \psi.$$

Example 1. *One can express the following languages over* $\{0, 1\}$ *in* MSO.

- $L_1 = \{w \in \{0,1\}^* \mid w$ *does not contain the substring 11*$\}$ *with the* MSO*-sentence*

$$\varphi = \forall x \forall y (x < y) \to \big((\exists z\ x < z < y) \vee x = 0 \vee y = 0\big).$$

- *The next formula expresses that "$S \subseteq [n]$ is an interval", i.e.* $(w, X) \models \varphi_{consec}$ *if and only if* X *defines an interval in the domain of* w.

$$\varphi_{int}(S) = \forall x \forall y \big((x \in S \wedge y \in S \wedge x \leq y) \rightarrow (\forall z \, (x \leq z \leq y) \rightarrow z \in S) \big)$$

- *The next formula expresses that "S forms a maximal interval in a set $X \subseteq [n]$".*

$$\varphi_{within}(X, S) = \forall x (x \in S \rightarrow x \in X) \wedge \varphi_{consec}(S) \wedge \forall y \big((y < S \vee S > y) \rightarrow y \notin S \big)$$

- *The next formula expresses that "$S \subseteq [n]$ is an interval with precisely two 1's".*

$$\varphi_{two1}(S) = \varphi_{int}(S) \wedge \exists x \in S \, \exists y \in S \big(P_1(x) \wedge P_1(y) \wedge x \neq y \big) \wedge \forall z \in S \big(P_1(z) \rightarrow (z = x \vee z = y) \big)$$

- $L_3 = \{w \in \{0,1\}^* \mid w$ *contains even number of 1's*$\}$ *with the* MSO*-sentence*

$$\varphi = \exists X \, \exists Y \big(\forall S \, (\varphi_{within}(X, S) \vee \varphi_{within}(Y, S)) \rightarrow \varphi_{two1}(S) \big)$$

2 Regular language and DFA

There are a range of excellent textbooks on formal language and automata, see [2] and [3]. Here we give a brief overview of key notations.

Definition 1. *A (deterministic) finite automaton is a 5-tuple* $M = (Q, \Sigma, \delta, q_0, F)$ *consisting of*

- Q*: a finite set called the* states*,*
- Σ*: a finite set called the* alphabet*,*
- δ*: a function from* $Q \times \Sigma$ *to* Q *called the* transition function*,*
- $q_0 \in Q$*: the* start state*,*
- $F \subseteq Q$*: the set of* accept states.

A finite automaton M *accepts a string* $s = s_1 \dots s_n \in \Sigma^*$ of length $n \geq 1$ if there is a sequence $r_0, r_1, \dots, r_\ell$ such that

- $r_0 = q_0$,
- $q_i = \delta(q_{i-1}, s_i)$ for every $i \in [n]$, and
- $q_n \in F$.

When the length of s is zero, i.e $s = \epsilon$, the automaton M accepts s if $q_0 \in F$.

In general, M changes the states and creates a sequence of states of the above form, except that the last condition $q_n \in F$ may not hold. Such a sequence of states that M goes through upon a string s the *run of* M *on string* s. We say that a language $L \subseteq \Sigma^*$ is *recognized by an automaton* M if L is precisely the set of strings on which M has an accepting run. A language recognized by some finite automaton is called a *regular language*.

A regular language is equivalently characterized by the existence of a regular expression generating it. Starting from the atom (single symbol from the alphabet, ϵ and $\emptyset$), an expression obtained by recursively applying any one of union, concatenation, complementation and Kleene star $(*)$ is again a regular expression. A regular expression defines a language (again recursively), and such a language is a regular language. Moreover, any regular language can be generated by some regular expression.

References

[1] Leonid Libkin. *Elements of Finite Model Theory*. Springer, 2004.

[2] Michael Sipser. *Introduction to the Theory of Computation*. Course Technology, Boston, MA, third edition, 2013.

[3] J. E. Hopcroft, R. Motwani, and J. D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison Wesley, 3rd edition, 2006.