2.6 Monadic second-order logic

Monadic first-order logic (see Section 1.3.2, page 42) is a decidable fragment of FOL which relies on purely relational vocabularies with only monadic (i.e., unary) predicate symbols. Since MFO cannot specify relations between two or more objects, it fails to provide characterizations of many interesting properties. We now consider the monadic fragment of second-order logic. The general version of monadic second-order logic (MSO) does not impose any restrictions on the vocabularies (and allows for predicate and function symbols of arbitrary arity), but restricts to second-order quantification over unary predicate variables. Since unary predicate variables have to be interpreted by sets (namely subsets of the domain), we may refer to them as *set variables* and write "$x \in X$" for the atomic formula $X(x)$. Similarly, $x \notin X$ stands short for $\neg X(x)$.

As standard FOL is subsumed by the general version of monadic second-order logic, no decidability results can be established. However, when restricting to certain vocabularies and structures for them, monadic second-order logic might become decidable. We will concentrate here on the interpretation of monadic second-order formulas over *finite words* and establish a connection between MSO and regular languages. This connection allows for a reduction of the satisfiability problem of MSO-formulas to the nonemptiness problem of finite automata and yields the decidability of the satisfiability problem for MSO over finite words.

For the purpose of this course, the syntax of MSO relies on a specific vocabulary that serves to formalize certain graph structures with labeled nodes and an edge relation E that specifies the direct successors of any node. That is, we use here a vocabulary $Voc_{\Sigma,\text{graph}}$ that extends Voc_{graph} by unary predicate symbols that allow to define atoms to reason about the node-labels. For the labeling of the nodes we assume a finite alphabet Σ and use atomic formulas of the form $P_a(x)$ where a is a symbol in Σ and P_a a monadic predicate symbol. The intuitive meaning of the atomic formula $P_a(x)$ is that the label of node x is a. Furthermore, we deal with MSO with equality, i.e., we have atomic formulas of the form "$x = y$" where x and y are first-order variables. Thus, in the approach of this course, MSO-formulas are built by four types of atomic formulas:

$P_a(x)$	with $x \in \mathit{Var}$, $a \in \Sigma$
$E(x,y)$	with $x, y \in \mathit{Var}$
$x \in X$	with $x \in \mathit{Var}$ and X an unary predicate variable
$x = y$	with $x, y \in \mathit{Var}$

Furthermore, we have the standard boolean connectives, first-order quantification and second-order quantification over set variables. Instead of a single relation E one might also use several relations $E_\alpha, E_\beta, \ldots$, which is, e.g., useful to reason about graph with labeled edges. This is irrelevant here and a single relation E is sufficient for our purposes. The abstract syntax of MSO (as we will use it here) is:

$$\varphi \;::=\; \mathit{true} \mid P_a(x) \mid E(x,y) \mid x \in X \mid x = y \mid \varphi_1 \wedge \varphi_2 \mid \neg\varphi \mid \forall x.\varphi \mid \forall X.\varphi$$

where $x \in \mathit{Var}$ and $X \in \mathit{PVar}$. As before, we use here only $\wedge$ and $\neg$ for the propositional logic fragment and universal quantification. Other boolean operators and existential quantification can be derived as usual. MSO-formulas are interpreted over structures that describe a graph

$$
\begin{array}{lll}
(\mathcal{G},\mathcal{V}) \models \mathit{true} & & \\
(\mathcal{G},\mathcal{V}) \models P_a(x) & \text{iff} & \mathcal{V}(x) \in P_a^{\mathcal{G}} \\
(\mathcal{G},\mathcal{V}) \models E(x,y) & \text{iff} & (\mathcal{V}(x),\mathcal{V}(y)) \in E^{\mathcal{G}} \\
(\mathcal{G},\mathcal{V}) \models x \in X & \text{iff} & \mathcal{V}(x) \in \mathcal{V}(X) \\
(\mathcal{G},\mathcal{V}) \models x = y & \text{iff} & \mathcal{V}(x) = \mathcal{V}(y) \\
(\mathcal{G},\mathcal{V}) \models \phi_1 \wedge \phi_2 & \text{iff} & (\mathcal{G},\mathcal{V}) \models \phi_1 \text{ and } (\mathcal{G},\mathcal{V}) \models \phi_2 \\
(\mathcal{G},\mathcal{V}) \models \neg\phi & \text{iff} & (\mathcal{G},\mathcal{V}) \not\models \phi \\
(\mathcal{G},\mathcal{V}) \models \forall x.\phi & \text{iff} & (\mathcal{G},\mathcal{V}[x := a]) \models \phi \text{ for all } a \in A \\
(\mathcal{G},\mathcal{V}) \models \forall X.\phi & \text{iff} & (\mathcal{G},\mathcal{V}[X := B]) \models \phi \text{ for all } B \subseteq A
\end{array}
$$

Figure 18: Satisfaction relation $\models$ for MSO

where the nodes are labeled with symbols from Σ. Thus, we refer to them as *labeled graph structures*. Formally, a labeled graph structure is a tuple

$$\mathcal{G} \;=\; \big(A, E^{\mathcal{G}}, (P_a^{\mathcal{G}})_{a\in\Sigma}\big)$$

where A is a nonempty set of nodes, $E^{\mathcal{G}} \subseteq A \times A$ the edge relation and $(P_a^{\mathcal{G}})_{a\in\Sigma}$ is a family of predicates that encode the labeling of the nodes. As before, we often identify the predicate symbol E with the edge relation of $\mathcal{G}$ and simply write E rather than $E^{\mathcal{G}}$. Given a node $a \in A$ and an element $a \in \Sigma$, we have: $a \in P_a^{\mathcal{G}}$ if and only if node a is labeled by a.[5] Given a variable valuation $\mathcal{V}$ that assigns to each first-order variable x a node $\mathcal{V}(x) \in A$ and to each set variable X a node-set $\mathcal{V}(X) \subseteq A$, the satisfaction relation $\models$ is defined in the expected way as shown in Figure 18.

Example 2.6.1 (Singleton and other conditions for sets). The MSO-formula

$$\mathit{singleton}(X) \;\stackrel{\text{def}}{=}\; \exists!x.\, x \in X \;=\; \exists x.\Big(\, x \in X \wedge \forall y.\big(y \in X \to x = y\big)\,\Big)$$

states that X contains exactly one element. The requirement X to be an nonempty set is specified in MSO by the formula:

$$X \neq \varnothing \;\stackrel{\text{def}}{=}\; \exists x.\, x \in X$$

Similarly, $X = \varnothing$ is a shorthand notation for $\neg\exists x.\, x \in X$. The MSO-formula

$$X \subseteq Y \;\stackrel{\text{def}}{=}\; \forall x.\big(x \in X \to x \in Y\big)$$

asserts that X is a subset of Y. ∎

Example 2.6.2 (Partition). The following formula $\mathit{partition}(X_1,\ldots,X_k)$ states that $X_1,\ldots,X_k$ provide a partition of the domain into disjoint subsets:

$$\mathit{partition}(X_1,\ldots,X_k) \;\stackrel{\text{def}}{=}\; \forall x.\bigvee_{1\leqslant i\leqslant k} x \in X_i \;\wedge\; \forall x.\bigwedge_{1\leqslant i<j\leqslant k} (\,x \notin X_i \vee x \notin X_j\,)$$

[5] As long as we do not require that the predicates $P_a^{\mathcal{G}}$ are pairwise disjoint, any node could be labeled with several symbols $a \in \Sigma$ or with none of them. That is, the labeling of a node is a subset of Σ. For the moment this is irrelevant. For the word structures considered later, any node has a unique label $a \in \Sigma$, except for the word structure of the empty word.

Note that the first subformula states that each element of the domain is contained in some X_i, while the second subformula states that the X_i's are pairwise disjoint. ■

Example 2.6.3 (Reachability). MSO is powerful enough to formalize reachability. In fact, the MSO-formula

$$\phi_{\text{reach}}(x,y) \stackrel{\text{def}}{=} \forall Z.\,\big(x \in Z \wedge \forall u \forall v.\,((u \in Z \wedge E(u,v)) \rightarrow v \in Z) \rightarrow y \in Z\big)$$

asserts that y is reachable from x (see Example 2.4.1 on page 150). ■

Existential MSO means the fragment of MSO consisting of formulas that start with a prefix of existential second-order quantifiers, followed by an FOL-formula.

Definition 2.6.4 (Existential MSO). An *existential MSO-formula* (EMSO-formula for short) is a MSO-formula of the type

$$\exists X_1 \ldots \exists X_k.\,\phi(X_1,\ldots,X_k,\overline{Y},\overline{z})$$

where $k \geqslant 0$, $X_1,\ldots,X_k$ are pairwise distinct set variables. $\overline{Y}$ stands for a (possibly empty) tuple of set variables, while $\overline{z}$ is a (possibly empty) tuple of first-order variables, and ϕ is a FOL-formula, when treating the X- and Y-variables as predicate symbols. ■

Example 2.6.5 (EMSO-formula for reachability in finite acyclic graphs). An EMSO-formula that specifies reachability in finite acyclic directed graphs is the following one:

$$\phi'_{\text{reach}}(x,y) \stackrel{\text{def}}{=} \exists Z.\,\psi(x,y,Z)$$

where x, y are FO-variables and Z is a set variable and where formula $\psi(x,y,Z)$ is defined by:

$$\begin{aligned}\psi(x,y,Z) &\stackrel{\text{def}}{=} x \in X \wedge y \in X \wedge \theta(y,Z)\\ \theta(y,Z) &\stackrel{\text{def}}{=} \forall z.\,\big(z \in X \wedge z \neq y \rightarrow \exists u.\,(u \in X \wedge E(z,u))\big)\end{aligned}$$

Intuitively, formula $\phi'_{\text{reach}}(x,y)$ states the existence of a set Z that contains nodes x and y and such that all elements $z \in X \setminus \{y\}$ have a successor in X. If $a_0\,a_1\,\ldots\,a_k$ is a path from $a = a_0$ to $b = a_k$ then $Z = \{a_0, a_1, \ldots, a_k\}$ is a witness that $\phi'_{\text{reach}}(x,y)$ holds for the interpretation $(\mathcal{G}, a, b)$. This shows that for each graph $\mathcal{G}$ and nodes a, b in $\mathcal{G}$ that serve as meanings for x and y, respectively, we have:

$$\text{If } b \text{ is reachable from } a \text{ then } (\mathcal{G}, a, b) \models \phi'_{\text{reach}}(x,y).$$

For infinite or cyclic graphs, $(\mathcal{G}, a, b)$ might be a model for $\phi'_{\text{reach}}(x,y)$, even if b is not reachable from a. E.g., if D is the set of nodes that constitute a cycle where a belongs to, then for $C = D \cup \{b\}$, we have: $(\mathcal{G}, a, b, C) \models \psi(x,y,Z)$. That is, node-set D witnesses the truth of $\phi'_{\text{reach}}(x,y)$ in $(\mathcal{G}, a, b)$, no matter whether b is reachable from a. However, for each finite acyclic graph $\mathcal{G}$ and nodes a, b in $\mathcal{G}$, we have:

$$(\mathcal{G}, a, b) \models \phi'_{\text{reach}}(x,y) \quad \text{iff} \quad b \text{ is reachable from } a$$

Let us see why the "only-if-part" holds. Suppose that $\mathcal{G}$ is finite and acyclic and that a, b are nodes in $\mathcal{G}$ with $(\mathcal{G}, a, b) \models \phi'_{\text{reach}}(x, y)$. Let C be a node-set such that $(\mathcal{G}, a, b, C) \models \psi(x, y, Z)$. Then, $a \in C$, $b \in C$ and whenever $c \in C \setminus \{b\}$ then there is some node $d \in C$ with $(c, d) \in E$. We now define inductively a sequence $a_0\, a_1\, a_2 \ldots$ consisting of nodes in C.

We start with $a_0 \stackrel{\text{def}}{=} a$. For $i \geqslant 0$ where $a_i \in C \setminus \{b\}$, we choose a node $a_{i+1} \in C$ with $(a_i, a_{i+1}) \in E$. In this way we generate a path $a_0\, a_1 \ldots a_k$ of nodes in C that starts in a. If $\mathcal{G}$ has n nodes then this procedure must end with $a_k = b$ for some $k < n$ (which yields a path from a to b), since otherwise $a_i = a_j$ for some indices i, j with $0 \leqslant i < j \leqslant n$ and $\mathcal{G}$ would be cyclic. ■

Example 2.6.6 (3-colorability). A graph $\mathcal{G} = (A, E)$ is called 3-colorable if there exists a partition of the node-set into three pairwise disjoint subsets A_1, A_2, A_3 such that for each edge $(a, a') \in E$ the source node a and the target node a' belong to different blocks of the partition. The 3-colorability of a graph can be specified by the following EMSO-sentence:

$$\exists X_1 \exists X_2 \exists X_3.\ \Big(\mathit{partition}(X_1, X_2, X_3) \wedge \forall x \forall y.\, \big(E(x, y) \rightarrow \bigwedge_{i=1,2,3} (x \in X_i \rightarrow y \notin X_i) \big) \Big)$$

where $\mathit{partition}(X_1, X_2, X_3)$ is as in Example 2.6.2. ■

MSO over finite words

In the remainder of this section we stick to special labeled graph structures that encode *finite words* over some finite and nonempty alphabet Σ. We will present the famous result by Büchi and Elgot stating the connection between MSO over finite words and regular languages over Σ. This result states that the class of MSO-definable languages of finite words agrees with the class of regular languages. Indeed, there is an algorithmic transformation that constructs a finite automaton for a given MSO-formula. Checking emptiness in this automaton then yields a decision procedure for checking satisfiability of MSO-formulas.

Finite words over Σ are finite sequences (strings) $w = a_1 \ldots a_n$ of elements $a_i \in \Sigma$. n is called the length of w and denoted by $|w|$. The empty word, i.e., the word of length 0, is denoted by ε. We shall use here an alternative, but equivalent view on finite words and consider them as labeled directed graphs where the nodes represent the positions and the label of the node at position i stands for the symbol a_i. That is, we identify any finite word $w = a_1 \ldots a_n \in \Sigma^+$ of length n with the directed labeled graph

$$\text{Graph}(w) \ \stackrel{\text{def}}{=}\ \big(A^w, E^w, (P^w_a)_{a \in \Sigma}\big)$$

where

- the node-set $A^w = \{1, \ldots, n\}$ is the set of word positions in w,
- $P^w_a = \big\{ i \in A^w : a_i = a \big\}$ is the set of word positions where w contains letter a,
- the edge relation is the successor relation on the word positions, i.e.,

$$E^w \ = \ \big\{ (i, i{+}1) : 1 \leqslant i < n \big\}.$$

Note that the transitive and reflexive closure of E^w yields a linear order of length $n = |w|$.

Since structures are required to have a nonempty domain, for the word structure of the empty word we have to depart from the above definition of $\mathrm{Graph}(w)$. We define $\mathrm{Graph}(\varepsilon)$ to be the following structure with the singleton domain $\{0\}$:

$$\mathrm{Graph}(\varepsilon) \stackrel{\text{def}}{=} \left(\{0\}, E^\varepsilon, (P_a^\varepsilon)_{a\in\Sigma}\right)$$

where the edge relation E^ε and the predicates P_a^ε are empty, i.e., $E^\varepsilon = \varnothing$ and $P_a^\varepsilon = \varnothing$ for all $a \in \Sigma$. The word structures for nonempty words can be distinguished from the word structure $\mathrm{Graph}(\varepsilon)$ by the fact that in $\mathrm{Graph}(w)$ for $w \in \Sigma^+$ any node is labeled with some $a \in \Sigma$ (i.e., is contained in some P_a^w), while the element 0 in $\mathrm{Graph}(\varepsilon)$ has no label in Σ.

The graph structures $\mathrm{Graph}(w)$ for $w \in \Sigma^*$ are called *(finite) word structures*. The satisfaction relation $\models$ for MSO-formulas and word structures is obtained by the standard MSO-semantics, see Figure 18 on page 167. If ϕ is a MSO-sentence and $\mathrm{Graph}(w) \models \phi$ then $\mathrm{Graph}(w)$ is called a (finite) *word model* for ϕ.

The predicate $\leqslant$ is defined by the EMSO-formula the EMSO-formula $\phi'_{\text{reach}}(x,y)$ of Example 2.6.5, which asserts that y is reachable from x. When interpreted over word structures this gives the standard meaning "less than or equal" of $\leqslant$. The predicate $<$ is derived from $\leqslant$ in the standard way by $x < y \stackrel{\text{def}}{=} x \leqslant y \wedge x \neq y$.

We will often use the following MSO-formulas to denote the first and last position of a finite word:

$$\begin{aligned} \mathit{first}(x) &\stackrel{\text{def}}{=} \bigvee_{a\in\Sigma} P_a(x) \wedge \neg\exists y.E(y,x) \\ \mathit{last}(x) &\stackrel{\text{def}}{=} \bigvee_{a\in\Sigma} P_a(x) \wedge \neg\exists y.E(x,y) \end{aligned}$$

In both cases, the subformula $\bigvee_{a\in\Sigma} P_a(x)$ serves to ensure that x is a proper word position and not the special element 0 of $\mathrm{Graph}(\varepsilon)$. Thus:

$$\begin{aligned} \mathrm{Graph}(\varepsilon) &\not\models \exists x.\, \mathit{first}(x) \vee \exists x.\, \mathit{last}(x) \\ \mathrm{Graph}(w) &\models \exists! x.\, \mathit{first}(x) \wedge \exists! y.\, \mathit{last}(y) \quad \text{for all } w \in \Sigma^+ \end{aligned}$$

The formula $\exists x.\mathit{first}(x)$ characterizes the nonempty words in the sense that for all finite words $w \in \Sigma^*$:

$$\mathrm{Graph}(w) \models \exists x.\, \mathit{first}(x) \quad \text{iff} \quad w \neq \varepsilon$$

The analogous statement holds for $\exists x.\mathit{last}(x)$. Other intuitive shorthand notations that will often be used later are:

$$\begin{aligned} x \notin X &\stackrel{\text{def}}{=} \neg(x \in X) \\ \mathit{first} \in X &\stackrel{\text{def}}{=} \exists x.\left(\mathit{first}(x) \wedge x \in X\right) \\ \mathit{last} \in X &\stackrel{\text{def}}{=} \exists x.\left(\mathit{last}(x) \wedge x \in X\right) \end{aligned}$$

For any word structure $\mathrm{Graph}(w)$ where $w \neq \varepsilon$ the predicates P_a^w yield a partition of $\{1,\ldots,n\}$ where $n = |w|$:

$$\mathrm{Graph}(w) \;\models\; \forall x.\, \bigwedge_{\substack{a,b\in\Sigma \\ a\neq b}} \big(\neg P_a(x) \vee \neg P_b(x)\big) \;\wedge\; \forall x.\, \bigvee_{a\in\Sigma} P_a(x)$$

For the empty word structure $\mathrm{Graph}(\varepsilon)$ the above MSO-sentence does not hold, since the subformula $\forall x.\, \bigvee_{a\in\Sigma} P_a(x)$ is violated.

Definition 2.6.7 (MSO-definable languages). Let $L \subseteq \Sigma^*$, i.e., L is a language of finite words over Σ. L is said to be *MSO-definable* if there exists a MSO-sentence ϕ over the vocabulary $\mathit{Voc}_{\Sigma,\mathrm{graph}}$ such that

$$L \;=\; \big\{\, w \in \Sigma^* : \mathrm{Graph}(w) \models \phi \,\big\}$$

In this case, we say that ϕ *defines* L. ■

Example 2.6.8 (MSO-sentence for a regular language). The following MSO-sentence defines the language L consisting of all nonempty finite words of even length:

$$\exists X.\, \Big(\mathit{first} \in X \wedge \mathit{last} \notin X \wedge \forall x \forall y.\, \big(E(x,y) \rightarrow (x \in X \leftrightarrow y \notin X) \big) \Big)$$

where $\mathit{last} \notin X$ stands for $\exists x.\, \big(\mathit{last}(x) \wedge x \notin X\big)$. This formula asserts the existence of a set X of positions that contains the first position, but not the last one and such that exactly any second position is contained in X. Thus, X contains exactly the odd positions $1,3,5,\ldots$. The subformula $\mathit{last} \notin X$ then requires that the last position (and hence the length of the word) is even. ■

In the sequel, we often will use regular expressions. These constitute a formalism for regular languages based on the fact that the class of regular languages over some alphabet Σ is the smallest class of languages that contains $\varnothing$, the singletons $\{\varepsilon\}$ and $\{a\}$ for $a \in \Sigma$ and is closed under union, concatenation and Kleene star (finite repetition). Formally, *regular expressions* are built by the symbols $\emptyset$, ε and $a \in \Sigma$ and the operators union, concatenation and Kleene star:

$$\alpha \;::=\; \emptyset \;\mid\; \varepsilon \;\mid\; a \;\mid\; \alpha_1 + \alpha_2 \;\mid\; \alpha_1\,\alpha_2 \;\mid\; \alpha^*$$

where a ranges over all symbols in Σ. The associated (regular) language $\mathfrak{L}(\alpha) \subseteq \Sigma^*$ of a regular expression is defined in the obvious way, see Figure 19. In what follows, we often identify a regular expression α with its language $\mathfrak{L}(\alpha)$.

Example 2.6.9 (MSO-sentences for regular expressions). Let $\Sigma = \{a,b\}$. The regular language given by the regular expression $a(a+b)^*a$ is defined by the MSO-sentence

$$\exists x.\, \big(\mathit{first}(x) \wedge P_a(x) \;\wedge \exists y.(\,\mathit{last}(y) \wedge x \neq y \wedge P_a(y)\,)\big)$$

which states that the first and last symbol has to be letter a and that the length of the word is at most 2 (by requiring that the last and first position are different). The MSO-sentence

$$\forall x \forall y.\, \big(\, P_a(x) \wedge P_b(y) \;\rightarrow\; x < y \,\big)$$

$$
\begin{aligned}
\mathfrak{L}(\emptyset) &\stackrel{\text{def}}{=} \varnothing \\
\mathfrak{L}(\varepsilon) &\stackrel{\text{def}}{=} \{\varepsilon\} \\
\mathfrak{L}(\mathfrak{a}) &\stackrel{\text{def}}{=} \{\mathfrak{a}\} \\
\mathfrak{L}(\,\alpha_1 + \alpha_2\,) &\stackrel{\text{def}}{=} \mathfrak{L}(\alpha_1) \cup \mathfrak{L}(\alpha_2) \\
\mathfrak{L}(\,\alpha_1\,\alpha_2\,) &\stackrel{\text{def}}{=} \mathfrak{L}(\alpha_1)\,\mathfrak{L}(\alpha_2) \;=\; \big\{ w_1 w_2 : w_i \in \mathfrak{L}(\alpha_i), i = 1,2 \big\} \\
\mathfrak{L}(\,\alpha^*\,) &\stackrel{\text{def}}{=} \mathfrak{L}(\alpha)^* \;=\; \{\varepsilon\} \cup \bigcup_{n \geqslant 1} \big\{ w_1 \ldots w_n : w_i \in \mathfrak{L}(\alpha), i = 1, \ldots, n \big\}
\end{aligned}
$$

Figure 19: Semantics of regular expressions

specifies the regular language $\big\{\mathfrak{a}^n\mathfrak{b}^m : n, m \in \mathbb{N}\big\}$ given by the regular expression $\mathfrak{a}^*\mathfrak{b}^*$.

The regular language given by $(\mathfrak{a}\mathfrak{b})^*$ is defined by the MSO-sentence $\neg\exists x.\mathit{first}(x) \vee \phi$ where ϕ states that (i) the first symbol is $\mathfrak{a}$, (ii) letters $\mathfrak{a}$ and $\mathfrak{b}$ alternate and (iii) the last symbol is $\mathfrak{b}$:

$$
\begin{aligned}
\phi \;\stackrel{\text{def}}{=}\; & \exists x.\big(\mathit{first}(x) \wedge P_{\mathfrak{a}}(x)\big) \\
\wedge\; & \forall z \forall z'.\big(E(z,z') \rightarrow \big((P_{\mathfrak{a}}(z) \rightarrow P_{\mathfrak{b}}(z')) \wedge (P_{\mathfrak{b}}(z) \rightarrow P_{\mathfrak{a}}(z'))\big)\big) \\
\wedge\; & \exists y.\big(\mathit{last}(y) \wedge P_{\mathfrak{b}}(y)\big)
\end{aligned}
$$

Then, $\mathrm{Graph}(w) \models \phi$ iff w is a word in $(\mathfrak{a}\mathfrak{b})^+$. Thus, $\mathrm{Graph}(w)$ is a model for $\neg\exists x.\mathit{first}(x) \vee \phi$ if and only if w is a word in $(\mathfrak{a}\mathfrak{b})^*$. ∎

In the remainder of this section, we show that the class of MSO-definable languages over finite words agrees with the class of regular languages. To prove the MSO-definability of regular languages we shall use a representation of the given regular language by a *nondeterministic finite automaton* (NFA). Let us briefly explain our notations. An NFA is a tuple

$$\mathcal{M} \;=\; (Q, \Sigma, \delta, Q_0, Q_F)$$

where Q is a finite set of states, Σ the input alphabet, $\delta : Q \times \Sigma \to 2^Q$ the transition function that assigns to each state q and input letter $\mathfrak{a} \in \Sigma$ a set $\delta(q,\mathfrak{a})$ of potential successor states. Furthermore, $Q_0 \subseteq Q$ is the set of starting states and $Q_F \subseteq Q$ the set of final (accept) states. Given a word $w = \mathfrak{a}_1 \ldots \mathfrak{a}_n \in \Sigma^*$, a *run* for w in $\mathcal{M}$ is a sequence $\pi = p_0 p_1 \ldots p_n \in Q^+$ such that $p_0 \in Q_0$ and $p_{i+1} \in \delta(p_i, \mathfrak{a}_{i+1})$ for $i = 0, 1, \ldots, n-1$. Run π is called *accepting* if $p_n \in Q_F$. The language recognized by $\mathcal{M}$, denoted $\mathfrak{L}(\mathcal{M})$, also often called the accepted language of $\mathcal{M}$, is the set of all words $w \in \Sigma^*$ that have an accepting run in $\mathcal{M}$.

It is well-known that the class of languages recognizable by NFA agrees with the class of regular languages. Furthermore, each NFA has an equivalent *deterministic finite automaton* (DFA), which means an NFA $\mathcal{M}$ as above with a single initial state and exactly one successor for each state q and symbol $\mathfrak{a}$, i.e., $|Q_0| = 1$ and $|\delta(q,\mathfrak{a})| = 1$ for all $q \in Q$ and $\mathfrak{a} \in \Sigma$. By Kleene's Theorem, the class of regular languages is closed under union, intersection, concatenation, the Kleene star operator and complementation.

Theorem 2.6.10 (EMSO-definability of regular languages). *For each regular language* $L \subseteq \Sigma^*$ *there exists an EMSO-sentence* ϕ *over the vocabulary* $\mathrm{Voc}_{\Sigma,\mathit{graph}}$ *such that:*

$$L \;=\; \big\{ w \in \Sigma^* : \mathit{Graph}(w) \models \phi \big\}$$

Proof. Since L is regular there is a nondeterministic finite automaton $\mathcal{M} = (Q, \Sigma, \delta, Q_0, Q_F)$ that recognizes L. Let $Q = \{q_1, \ldots, q_k\}$, where $q_1, \ldots, q_k$ are pairwise distinct. The idea is now to formalize the runs $p_0 p_1 \ldots p_n$ in $\mathcal{M}$ by means of partitions $X_1, \ldots, X_k$ where X_j stands for the set of word positions ℓ such that when reading the symbol a_ℓ state q_j is entered.

We define the EMSO-sentence ϕ as follows:

$$\phi \stackrel{\text{def}}{=} \exists X_1 \ldots \exists X_k. \big(\mathit{partition}(X_1, \ldots, X_k) \wedge \phi_{\text{init}} \wedge \phi_{\text{trans}} \wedge \phi_{\text{accept}} \big)$$

where $\mathit{partition}(X_1, \ldots, X_k)$ is as in Example 2.6.2 on page 167. Note that $\mathit{partition}(X_1, \ldots, X_k)$ is in fact a FOL-formula, when the X_i's are viewed as predicate symbols. The same holds for the subformulas ϕ_{init}, ϕ_{accept} and ϕ_δ, which are shown in Figure 20. Intuitively, subformulas ϕ_{accept} and ϕ_{init} state that the run encoded by the partition $X_1, \ldots, X_k$ begins with a starting state $q \in Q_0$ and ends in final state. Subformula ϕ_{trans} encodes the transition function of $\mathcal{M}$ by asserting that if at some word position x the current state is q_i, i.e., $x \in X_i$, and the current letter is $a_{x+1} = a$, then the state for word position $y = x{+}1$ belongs to $\delta(q_i, a)$. That is, $y \in X_j$ for some $q_j \in \delta(q_i, a)$.

$$\begin{aligned}
\phi_{\text{init}} &\stackrel{\text{def}}{=} \forall x. \Big(\mathit{first}(x) \rightarrow \bigvee_{a \in \Sigma} \bigvee_{q_0 \in Q_0} \bigvee_{q_i \in \delta(q_0, a)} \big(P_a(x) \wedge x \in X_i \big) \Big) \\
\phi_{\text{trans}} &\stackrel{\text{def}}{=} \forall x \forall y. \Big(E(x,y) \rightarrow \bigvee_{1 \leqslant i \leqslant k} \bigvee_{a \in \Sigma} \bigvee_{q_j \in \delta(q_i, a)} \big(x \in X_i \wedge P_a(y) \wedge y \in X_j \big) \Big) \\
\phi_{\text{accept}} &\stackrel{\text{def}}{=} \forall x. \Big(\mathit{last}(x) \rightarrow \bigvee_{q_i \in Q_F} x \in X_i \Big)
\end{aligned}$$

Figure 20: Formulas ϕ_{init}, ϕ_{trans} and ϕ_{accept}

A short remark on the formula for the initial state. One might expect a formula ϕ_{init} that is similar to ϕ_{accept}. However, the X_i's stand for subsets of $\{1, \ldots, n\}$ (where n is the length of the input word) and do not contain the initial position 0 where the first symbol a_1 of the input word $w = a_1 \ldots a_n$ is not yet consumed. The above formula ϕ_{init} describes that at position 1 (i.e., after having consumed the first letter) the automaton is in a state $p_1 = q_j$ that is reachable from some starting state $p_0 = q_0 \in Q_0$ via a transition labeled with the first symbol a_1 of the input word.

Let us check that for all nonempty words $w \in \Sigma^+$:

$$w \in L \quad \text{iff} \quad \text{Graph}(w) \text{ is a model for } \phi$$

"$\Longrightarrow$:" Suppose $w = a_1 \ldots a_n \in L = \mathfrak{L}(\mathcal{M})$. Let $p_0 p_1 \ldots p_n$ be an accepting run for w in $\mathcal{M}$. For $1 \leqslant j \leqslant k$ we define:

$$B_j \stackrel{\text{def}}{=} \big\{ \, \ell \in \{1, \ldots, n\} : p_\ell = q_j \, \big\}$$

Then, the sets $B_1, \ldots, B_k$ constitute a partition of the set $\{1, \ldots, n\}$ of word positions. For all word positions $\ell \in \{1, \ldots, n\}$ we have: $\ell \in P^w_{a_\ell}$ and

$$\text{if } \ell \in B_i \text{ and } \ell < n \text{ then } \ell{+}1 \in B_j \text{ for some state } q_j \in \delta(q_i, a_{\ell+1})$$

As $p_0 \in Q_0$ and $p_n \in Q_F$ we have: $1 \in \bigcup_{\substack{q_0 \in Q_0 \\ q_j \in \delta(q_0, a_1)}} B_j$ and $n \in \bigcup_{q_j \in Q_F} B_j$.

$$\begin{array}{ccccccccccc} p_0 & \xrightarrow{a_1} & p_1 & \xrightarrow{a_2} & \dots & \xrightarrow{a_\ell} & p_\ell & \xrightarrow{a_{\ell+1}} & p_{\ell+1} & \xrightarrow{a_{\ell+2}} & \dots \xrightarrow{a_n} p_n \\ \uparrow & & & & & & \uparrow & & \uparrow & & \uparrow \\ \in Q_0 & & & & & & q_i & & q_j & & \in Q_F \\ & & & & & & \ell \in B_i & & \ell{+}1 \in B_j & & \end{array}$$

This yields $(\mathrm{Graph}(w), B_1, \dots, B_k) \models \mathit{partition}(X_1, \dots, X_k) \wedge \phi_{\text{init}} \wedge \phi_{\text{trans}} \wedge \phi_{\text{accept}}$. Therefore, $\mathrm{Graph}(w) \models \phi$.

"$\Longleftarrow$:" Suppose $\mathrm{Graph}(w) \models \phi$ where $w = a_1 \dots a_n$ is a nonempty word over Σ. Then, there exists a partition $B_1, \dots, B_k$ of the set $\{1, \dots, n\}$ of word positions such that

$$(\mathrm{Graph}(w), B_1, \dots, B_k) \models \phi_{\text{init}} \wedge \phi_{\text{trans}} \wedge \phi_{\text{accept}}$$

For each word position ℓ, let $p_\ell \in Q$ be the unique state q_j such that $\ell \in B_j$. As $(\mathrm{Graph}(w), B_1, \dots, B_k)$ is a model for ϕ_{init}, there exists an initial state $p_0 \in Q_0$ and symbol $a \in \Sigma$ such that

$$p_1 \in \delta(p_0, a) \quad \text{and} \quad 1 \in P_a^w.$$

Then, $a = a_1$ is the first letter of w and $p_0 \xrightarrow{a_1} p_1$ is a transition in $\mathcal{M}$. As

$$(\mathrm{Graph}(w), B_1, \dots, B_k) \models \phi_{\text{init}} \wedge \phi_{\text{trans}},$$

the state sequence $p_0 p_1 \dots p_n$ is a run for w in $\mathcal{M}$. Note that $(\mathrm{Graph}(w), B_1, \dots, B_k) \models \phi_{\text{trans}}$ yields that for each word position $\ell \in \{1, \dots, n{-}1\}$ there is some input symbol $a \in \Sigma$ such that

$$p_{\ell+1} \in \delta(p_\ell, a) \quad \text{and} \quad \ell{+}1 \in P_a^w.$$

The condition $\ell{+}1 \in P_a^w$ is equivalent to the requirement that a is the $(\ell{+}1)$-st letter in w, i.e., $a = a_{\ell+1}$. This yields that

$$p_\ell \xrightarrow{a_{\ell+1}} p_{\ell+1}$$

is a transition in $\mathcal{M}$ for $1 \leqslant \ell < n$. Together with the observation made above, we get that $p_0 p_1 \dots p_n$ is indeed a run for w in $\mathcal{M}$. Furthermore, $p_n \in Q_F$ is a final state. Note that $(\mathrm{Graph}(w), B_1, \dots, B_k) \models \phi_{\text{accept}}$ yields that $n \in B_j$ for some $j \in \{1, \dots, k\}$ such that $q_j \in Q_F$. But then $p_n = q_j$ for some state $q_j \in Q_F$. Hence, $p_0 p_1 \dots p_n$ is an accepting run for w in $\mathcal{M}$. Therefore, $w \in \mathfrak{L}(\mathcal{M}) = L$. ⌋

Since all three subformulas ϕ_{trans}, ϕ_{accept} and ϕ_{init} hold for the empty word, we get:

$$L = \left\{ w \in \Sigma^* : \mathrm{Graph}(w) \models \phi \right\} \quad \text{if } \varepsilon \in L = \mathfrak{L}(\mathcal{M})$$

A slight modification of ϕ is required when $\varepsilon \notin L$, i.e., the automaton $\mathcal{M}$ does not accept the empty word. (This holds iff $Q_0 \cap Q_F = \varnothing$.) In this case, we switch to the formula

$$\phi' \stackrel{\text{def}}{=} \phi \wedge \exists x.\mathit{first}(x),$$

which states additionally the existence of a first position. Thus, if $\varepsilon \notin L = \mathfrak{L}(\mathcal{M})$ then

$$w \in L \quad \text{iff} \quad \mathrm{Graph}(w) \models \phi'.$$

for all words $w \in \Sigma^*$. □