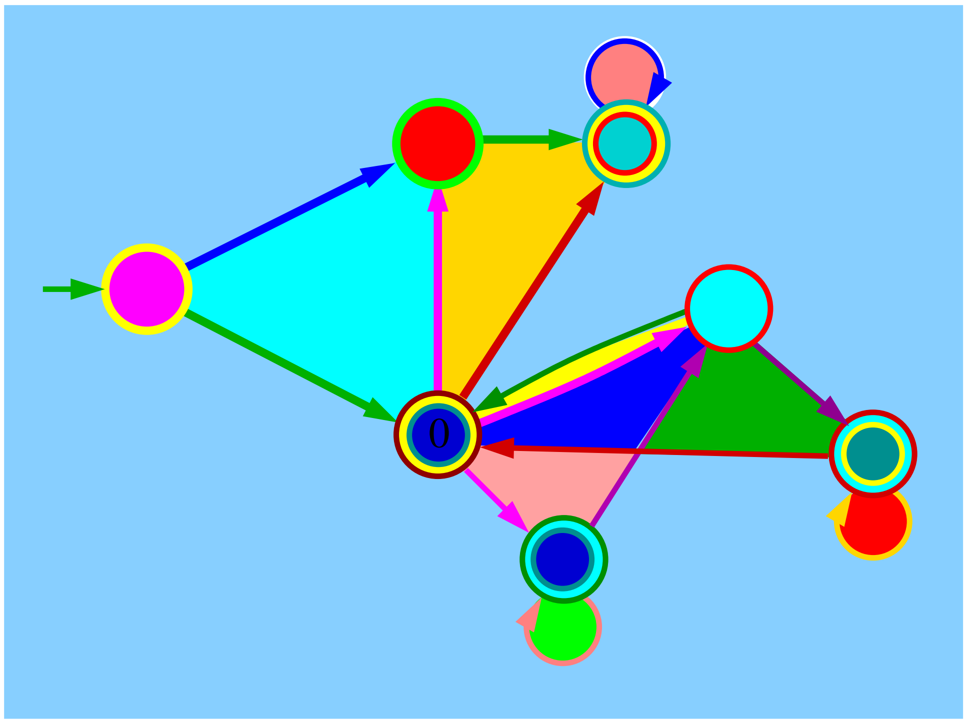
Automata theory

An algorithmic approach



Lecture Notes

Javier Esparza

July 20, 2012

Chapter 9

Automata and Logic

A regular expression can be seen as a set of instructions (a 'recipe') for generating the words of a language. For instance, the expression $aa(a+b)^*b$ can be interpreted as "write two a's, repeatedly write a or b an arbitrary number of times, and then write a b". We say that regular expressions are an *operational* description language.

Languages can also be described in *declarative* style, as the set of words that satisfy a property. For instance, "the words over $\{a, b\}$ containing an even number of a's and an even number of b's" is a declarative description. A language may have a simple declarative description and a complicated operational description as a regular expression. For instance, the regular expression

$$(aa + bb + (ab + ba)(aa + bb)^*(ba + ab))^*$$

is a natural operational description of the language above, and it is arguably less intuitive than the declarative one. This becomes even more clear if we consider the language of the words over $\{a, b, c\}$ containing an even number of a's, of b's, and of c's.

In this chapter we present a logical formalism for the declarative description of regular languages. We use logical formulas to describe properties of words, and logical operators to construct complex properties out of simpler ones. We then show how to automatically translate a formula describing a property of words into an automaton recognizing the words satisfying the property. As a consequence, we obtain an algorithm to convert declarative into operational descriptions, and vice versa.

9.1 First-Order Logic on Words

In declarative style, a language is defined by its *membership predicate*, i.e., the property that words must satisfy in order to belong to it. Predicate logic is the standard language to express membership predicates. Starting from some natural, "atomic" predicates, more complex ones can be constructed through boolean combinations and quantification. We introduce atomic predicates $Q_a(x)$, where a is a letter, and x ranges over the positions of the word. The intended meaning is "the letter at position x is an a." For instance, the property "all letters are as" is formalized by the formula $\forall x\, Q_a(x)$.

In order to express relations between positions we add to the syntax the predicate $x < y$, with intended meaning "position x is smaller than (i.e., lies to the left of) position y". For example, the property "if the letter at a position is an a, then all letters to the right of this position are also as" is formalized by the formula

$$\forall x \forall y\, ((Q_a(x) \wedge x < y) \rightarrow Q_a(y))\ .$$

Definition 9.1 *Let $V = \{x, y, z, \ldots\}$ be an infinite set of* variables*, and let $\Sigma = \{a, b, c, \ldots\}$ be a finite alphabet. The set $FO(\Sigma)$ of* first-order formulas *over Σ is the set of expressions generated by the grammar:*

$$\varphi := Q_a(x) \mid x < y \mid \neg\varphi \mid (\varphi \vee \varphi) \mid \exists x\, \varphi\ .$$

As usual, variables within the scope of an existential quantifier are *bounded*, and otherwise *free*. A formula without free variables is a *sentence*. Sentences of $FO(\Sigma)$ are interpreted on words over Σ. For instance, $\forall x\, Q_a(x)$ is true for the word aa, but false for word ab. Formulas with free variables cannot be interpreted on words alone: it does not make sense to ask whether $Q_a(x)$ holds for the word ab or not. A formula with free variables is interpreted over a pair $(w, \mathfrak{I})$, where $\mathfrak{I}$ assigns to each free variable (and perhaps to others) a position in the word. For instance, $Q_a(x)$ is true for the pair $(ab, x \mapsto 1)$, because the letter at position 1 of ab is a, but false for $(ab, x \mapsto 2)$.

Definition 9.2 *An* interpretation *of a formula φ of $FO(\Sigma)$ is a pair $(w, \mathfrak{I})$ where $w \in \Sigma^*$ and $\mathfrak{I}$ is a mapping that assigns to every free variable x a* position $\mathfrak{I}(x) \in \{1, \ldots, |w|\}$ *(the mapping may also assign positions to other variables).*

Notice that if φ is a sentence then a pair $(w, \mathcal{E})$, where $\mathcal{E}$ is the empty mapping that does not assign any position to any variable, is an interpretation of φ. Instead of $(w, \mathcal{E})$ we write simply w.

We now formally define when an interpretation satisfies a formula. Given a word w and a number k, let $w[k]$ denote the letter of w at position k.

Definition 9.3 *The satisfaction relation $(w, \mathfrak{I}) \models \varphi$ between a formula φ of $FO(\Sigma)$ and an interpretation $(w, \mathfrak{I})$ of φ is defined by:*

$$\begin{array}{lcll}
(w, \mathfrak{I}) & \models & Q_a(x) & \textit{iff} \quad w[\mathfrak{I}(x)] = a \\
(w, \mathfrak{I}) & \models & x < y & \textit{iff} \quad \mathfrak{I}(x) < \mathfrak{I}(y) \\
(w, \mathfrak{I}) & \models & \neg\varphi & \textit{iff} \quad (w, \mathfrak{I}) \not\models \varphi \\
(w, \mathfrak{I}) & \models & \varphi 1 \vee \varphi_2 & \textit{iff} \quad (w, \mathfrak{I}) \models \varphi_1 \textit{ or } (w, \mathfrak{I}) \models \varphi_2 \\
(w, \mathfrak{I}) & \models & \exists x\, \varphi & \textit{iff} \quad |w| \geq 1 \textit{ and some } i \in \{1, \ldots, |w|\} \textit{ satisfies } (w, \mathfrak{I}[i/x]) \models \varphi
\end{array}$$

where $w[i]$ is the letter of w at position i, and $\mathfrak{I}[i/x]$ is the mapping that assigns i to x and otherwise coincides with $\mathfrak{I}$. (Notice that $\mathfrak{I}$ may not assign any value to x.) If $(w, \mathfrak{I}) \models \varphi$ we say that $(w, \mathfrak{I})$ is a model *of φ. Two formulas are* equivalent *if they have the same models.*

It follows easily from this definition that if two interpretations $(w, \mathcal{I}_1)$ and $(w, \mathcal{I}_2)$ of φ differ only in the positions assigned by $\mathcal{I}_1$ and $\mathcal{I}_2$ to bounded variables, then either both interpretations are models of φ, or none of them is. In particular, whether an interpretation $(w, \mathcal{I})$ of a sentence is a model or not depends only on w, not on $\mathcal{I}$.

We use some standard abbreviations:

$$\forall x\, \varphi := \neg \exists\, x \neg \varphi \qquad \varphi_1 \wedge \varphi_2 := \neg\,(\neg \varphi_1 \vee \neg \varphi_2) \qquad \varphi_1 \rightarrow \varphi_2 := \neg \varphi_1 \vee \varphi_2$$

Notice that according to the definition of the satisfaction relation the empty word ϵ satisfies no formulas of the form $\exists x\, \varphi$, and all formulas of the form $\forall x\, \varphi$. While this causes no problems for our purposes, it is worth noticing that in other contexts it may lead to complications. For instance, the formulas $\exists x\, Q_a(x)$ and $\forall y \exists x\, Q_a(x)$ do not hold for exactly the same words, because the empty word satisfies the second, but not the first. Further useful abbreviations are:

$$\begin{array}{rcll} \text{first}(x) & := & \neg \exists y\, y < x & \text{“}x\text{ is the first position”} \\ \text{last}(x) & := & \neg \exists y\, x < y & \text{“}x\text{ is the last position”} \\ y = x + 1 & := & x < y \wedge \neg \exists\, z(x < z \wedge z < y) & \text{“}y\text{ is the successor position of }x\text{”} \\ y = x + 2 & := & \exists\, z(z = x + 1 \wedge y = z + 1) & \\ y = x + (k+1) & := & \exists\, z(z = x + k \wedge y = z + 1) & \end{array}$$

Example 9.4 Some examples of properties expressible in the logic:

- “The last letter is a b and before it there are only a’s.”

$$\exists x\, Q_b(x) \wedge \forall x\, (\text{last}(x) \rightarrow Q_b(x) \ \wedge\ \neg\text{last}(x) \rightarrow Q_a(x))$$

- “Every a is immediately followed by a b.”

$$\forall x\, (Q_a(x) \rightarrow \exists y\, (y = x + 1 \wedge Q_b(y)))$$

- “Every a is immediately followed by a b, unless it is the last letter.”

$$\forall x\, (Q_a(x) \rightarrow \forall y\, (y = x + 1 \rightarrow Q_b(y)))$$

- “Between every a and every later b there is a c.”

$$\forall x \forall y\, (Q_a(x) \wedge Q_b(y) \wedge x < y \rightarrow \exists z\, (x < z \wedge z < y \wedge Q_c(z)))$$

□

9.1.1 Expressive power of $FO(\Sigma)$

Once we have defined which words satisfy a sentence, we can associate to a sentence the set of words satisfying it.

Definition 9.5 *The language $L(\varphi)$ of a sentence $\varphi \in FO(\Sigma)$ is the set $L(\varphi) = \{w \in \Sigma^* \mid w \models \phi\}$. We also say that φ* expresses *$L(\varphi)$. A language $L \subseteq \Sigma^*$ is* FO-definable *if $L = L(\varphi)$ for some formula φ of $FO(\Sigma)$.*

The languages of the properties in the example are FO-definable by definition. To get an idea of the expressive power of $FO(\Sigma)$, we prove a theorem characterizing the FO-definable languages in the case of a 1-letter alphabet $\Sigma = \{a\}$. In this simple case we only have one predicate $Q_a(x)$, which is always true in every interpretation. So every formula is equivalent to a formula without any occurrence of $Q_a(x)$. For example, the formula $\exists y\, (Q_a(y) \wedge y < x)$ is equivalent to $\exists y\, y < x$.

We prove that a language over a one-letter alphabet is FO-definable if and only if it is finite or *co-finite*, where a language is co-finite if its complement is finite. So, for instance, even a simple language like $\{a^n \mid n \text{ is even }\}$ is not FO-definable. The plan of the proof is as follows. First, we define the *quantifier-free fragment* of $FO(\{a\})$, denoted by QF; then we show that 1-letter languages are QF-definable iff they are finite or co-finite; finally, we prove that 1-letter languages are FO-definable iff they are QF-definable.

For the definition of QF we need some more macros whose intended meaning should be easy to guess:

$$\begin{array}{rcl} x+k<y & := & \exists z\,(z = x+k \wedge z < y) \\ x<y+k & := & \exists z\,(z = y+k \wedge x < z) \\ k < last & := & \forall x\,(\text{last}(x) \rightarrow x > k) \end{array}$$

In these macros k is a constant, that is, $k < last$ standa for the infinite family of macros $1 < last, 2 < last, 3 < last \ldots$. Macros like $k > x$ or $x + k > y$ are defined similarly.

Definition 9.6 *The logic QF (for quantifier-free) is the fragment of $FO(\{a\})$ with syntax*

$$f := x \approx k \mid x \approx y + k \mid k \approx last \mid f_1 \vee f_2 \mid f_1 \wedge f_2$$

where $\approx \in \{<, >\}$ and $k \in \mathbb{N}$.

Proposition 9.7 *A language over a 1-letter alphabet is QF-definable iff it is finite or co-finite.*

Proof: ($\Rightarrow$): Let f be a sentence of QF. Since QF does not have quantifiers, f does not contain any occurrence of a variable, and so it is a positive (i.e., negation-free) boolean combination of formulas of the form $k < last$ or $k > last$. We proceed by induction on the structure of f. If $f = k < last$, then $L(\varphi)$ is co-finite, and if $f = k > last$, then $L(\varphi)$ is finite. If $f = f_1 \vee f_2$, then by induction hypothesis $L(f_1)$ and $L(f_2)$ are finite or co-finite; if $L(f_1)$ and $L(f_2)$ are finite, then so is $L(f)$, and otherwise $L(f)$ is co-finite. The case $f = f_1 \wedge f_2$ is similar.

($\Leftarrow$): A finite language $\{a^{k_1}, \dots, a^{k_n}\}$ is expressed by the formula $(last > k_1 - 1 \wedge last < k_1 + 1) \vee \dots \vee (last > k_1 - 1 \wedge last < k_1 + 1)$. To express a co-finite language, it suffices to show that for every formula f of QF expressing a language L, there is another formula $\overline{f}$ expressing the language $\overline{L}$. This is easily proved by induction on the structure of the formula. □

Theorem 9.8 *Every formula φ of $FO(\{a\})$ is equivalent to a formula f of QF.*

Proof: *Sketch.* By induction on the structure of φ. If $\varphi(x, y) = x < y$, then $\varphi \equiv y < x + 0$. If $\varphi = \neg\psi$, the result follows from the induction hypothesis and the fact that negations can be removed using De Morgan's rules and equivalences like $\neg(x < y + k) \equiv x \geq y + k$. If $\varphi = \varphi_1 \vee \varphi_2$, the result follows directly from the induction hypothesis. Consider now the case $\varphi = \exists x\ \psi$. By induction hypothesis, ψ is equivalent to a formula f of QF, and we can assume that f is in disjunctive normal form, say $f = D_1 \vee \dots \vee D_n$. Then $\varphi \equiv \exists x\ D_1 \vee \exists x\ D_2 \vee \dots \vee \exists x\ D_n$, and so it suffices to find a formula f_i of QF equivalent to $\exists x\ D_i$.

The formula f_i is a conjunction of formulas containing all conjuncts of D_i with no occurrence of x, plus other conjuncts obtained as follows. For every *lower bound* $x < t_1$ of D_i, where $t_1 = k_1$ or $t_1 = x_1 + k_1$, and every *upper bound* of the form $x > t_2$, where $t_2 = k_1$ or $t_2 = x_1 + k_1$ we add to f_i a conjunct equivalent to $t_2 + 1 < t_1$. For instance, $y + 7 < x$ and $x < z + 3$ we add $y + 5 < z$. It is easy to see that $f_i \equiv \exists x\ D_i$. □

Corollary 9.9 *The language Even* $= \{a^{2n} \mid n \geq 0\}$ *is not first-order expressible.*

These results show that first-order logic cannot express all regular languages, not even over a 1-letter alphabet. For this reason we now introduce monadic second-order logic.

9.2 Monadic Second-Order Logic on Words

Monadic second-order logic extends first-order logic with variables $X, Y, Z, \dots$ ranging over *sets* of positions, and with predicates $x \in X$, meaning "position x belongs to the set X. [1] It is allowed to quantify over both kinds of variables. Before giving a formal definition, let us informally see how this extension allows to describe the language *Even*. The formula states that the last position belongs to the set of even positions. A position belongs to this set iff it is the second position, or the second successor of another position in the set.

The following formula states that X is the set of even positions:

$$\begin{array}{rcl} \text{second}(x) & := & \exists y\ (\text{first}(y) \wedge x = y + 1) \\ Even(X) & := & \forall x\ \ (x \in X \leftrightarrow (\text{second}(x) \vee \exists y\ \ (x = y + 2 \wedge y \in X))) \end{array}$$

For the complete formula, we observe that the word has even length if its last position is even:

$$EvenLength := \exists X\ (Even(X) \wedge \forall x\ (\text{last}(x) \rightarrow x \in X)\ )$$

[1]More generally, second-order logic allows for variables ranging over relations of arbitrary arity. The monadic fragment only allows arity 1, which corresponds to sets.

We now define the formal syntax and semantics of the logic.

Definition 9.10 *Let $X_1 = \{x, y, z, \ldots\}$ and $X_2 = \{X, Y, Z, \ldots\}$ be two infinite sets of* first-order *and* second-order variables. *Let $\Sigma = \{a, b, c, \ldots\}$ be a finite alphabet. The set $MSO(\Sigma)$ of* monadic second-order formulas *over Σ is the set of expressions generated by the grammar:*

$$\varphi := Q_a(x) \mid x < y \mid x \in X \mid \neg\varphi \mid \varphi \vee \varphi \mid \exists x\, \varphi \mid \exists X\, \varphi$$

An interpretation *of a formula φ is a pair $(w, \mathfrak{I})$ where $w \in \Sigma^*$, and $\mathfrak{I}$ is a mapping that assigns every free first-order variable x a* position *$\mathfrak{I}(x) \in \{1, \ldots, |w|\}$ and every free second-order variable X a set of positions $\mathfrak{I}(X) \subseteq \{1, \ldots, |w|\}$. (The mapping may also assign positions to other variables.)*

The satisfaction relation $(w, \mathfrak{I}) \models \varphi$ between a formula φ of $MSO(\Sigma)$ and an interpretation $(w, \mathfrak{I})$ of φ is defined as for $FO(\Sigma)$, with the following additions:

$$\begin{array}{llll} (w,\mathfrak{I}) & \models & x \in X & \textit{iff} \quad \mathfrak{I}(x) \in \mathfrak{I}(X) \\ (w,\mathfrak{I}) & \models & \exists X\, \varphi & \textit{iff} \quad |w| > 0 \textit{ and some } S \subseteq \{1, \ldots, |w|\} \\ & & & \qquad \textit{satisfies } (w, \mathfrak{I}[S/X]) \models \varphi \end{array}$$

where $\mathfrak{I}[S/X]$ is the interpretation that assigns S to X and otherwise coincides with $\mathfrak{I}$ — whether $\mathfrak{I}$ is defined for X or not. If $(w, \mathfrak{I}) \models \varphi$ we say that $(w, \mathfrak{I})$ is a model *of φ. Two formulas are* equivalent *if they have the same models. The language $L(\varphi)$ of a sentence $\varphi \in MSO(\Sigma)$ is the set $L(\varphi) = \{w \in \Sigma^* \mid w \models \phi\}$. A language $L \subseteq \Sigma^*$ is* MSO-definable *if $L = L(\varphi)$ for some formula $\varphi \in MSO(\Sigma)$.*

Notice that in this definition the set S may be empty. So, for instance, ay interpretation that assigns the empty set to X is a model of the formula $\exists X\, \forall x\, \neg(x \in X)$.

We use the standard abbreviations

$$\forall x \in X\, \varphi \;:=\; \forall x\, (x \in X \rightarrow \varphi) \qquad \exists x \in X\, \varphi \;:=\; \exists x\, (x \in X \wedge \varphi)$$

9.2.1 Expressive power of *MSO*(Σ)

We show that the languages expressible in monadic second-order logic are exactly the regular languages. We start with an example.

Example 9.11 Let $\Sigma = \{a, b, c, d\}$. We construct a formula of $MSO(\Sigma)$ expressing the regular language $c^*(ab)^*d^*$. The membership predicate of the language can be informally formulated as follows:

> There is a block of consecutive positions X such that: before X there are only c's; after X there are only d's; in X b's and a's alternate; the first letter in X is an a and the last letter is a b.

The predicate is a conjunction of predicates. We give formulas for each of them.

- "X is a block of consecutive positions."

$$\text{Cons}(X) := \forall x \in X\ \forall y \in X\ \ (x < y \to (\forall z\ (x < z \land z < y) \to z \in X))$$

- "x lies before/after X."

$$\text{Before}(x, X) := \forall y \in X\ x < y \qquad \text{After}(x, X) := \forall y \in X\ y < x$$

- "Before X there are only c's."

$$\text{Before\_only\_c}(X) := \forall x\ \text{Before}(x, X) \to Q_c(x)$$

- "After X there are only d's."

$$\text{After\_only\_d}(X) := \forall x\ \text{After}(x, X) \to Q_d(x)$$

- "a's and b's alternate in X."

$$\begin{array}{rl} \text{Alternate}(X) := \ \forall x \in X & (\ Q_a(x) \to \forall y \in X\ (y = x + 1 \to Q_b(y)\ ) \\ & \qquad\qquad \land \\ & \ \ Q_b(x) \to \forall y \in X\ (y = x + 1 \to Q_a(y)\ )\ ) \end{array}$$

- "The first letter in X is an a and the last is a b."

$$\begin{array}{rcl} \text{First\_a}(X) & := & \forall x \in X\ \forall y\ (y < x \to \neg y \in X) \to Q_a(x) \\ \text{Last\_b}(X) & := & \forall x \in X\ \forall y\ (y > x \to \neg y \in X) \to Q_a(x) \end{array}$$

Putting everything together, we get the formula

$$\begin{array}{l} \exists X(\ \text{Cons}(X) \land \text{Before\_only\_c}(X) \land \text{After\_only\_d}(X) \land \\ \qquad \text{Alternate}(X) \land \text{First\_a}(X) \land \text{Last\_b}(X)\ ) \end{array}$$

Notice that the empty word is a model of the formula. because the empty set of positions satisfies all the conjuncts. □

Let us now directly prove one direction of the result.

Proposition 9.12 *If $L \subseteq \Sigma^*$ is regular, then L is expressible in MSO(Σ).*

Proof: Let $A = (Q, \Sigma, \delta, q_0, F)$ be a DFA with $Q = \{q_0, \ldots, q_n\}$ and $L(A) = L$. We construct a formula φ_A such that for every $w \neq \epsilon$, $w \models \varphi_A$ iff $w \in L(A)$. If $\epsilon \in L(A)$, then we can extend the formula to $\varphi_A \vee \varphi'_A$, where φ'_A is only satisfied by the empty word (e.g. $\varphi'_A = \forall x\, x < x$).

We start with some notations. Let $w = a_1 \ldots a_m$ be a word over Σ, and let

$$P_q = \left\{ i \in \{1, \ldots, m\} \mid \hat{\delta}(q_0, a_0 \ldots a_i) = q \right\} .$$

In words, $i \in P_q$ iff A is in state q immediately *after* reading the letter a_i. Then A accepts w iff $m \in \bigcup_{q \in F} P_q$.

Assume we were able to construct a formula $\text{Visits}(X_0, \ldots X_n)$ with free variables $X_0, \ldots X_n$ such that $\mathfrak{I}(X_i) = P_{q_i}$ holds for *every* model $(w, \mathfrak{I})$ and for every $0 \leq i \leq n$. In words, $\text{Visits}(X_0, \ldots X_n)$ is only true when X_i takes the value P_{q_i} for every $0 \leq i \leq n$. Then $(w, \mathfrak{I})$ would be a model of

$$\psi_A := \exists X_0 \ldots \exists X_n \ \text{Visits}(X_0, \ldots X_n) \wedge \exists x \left(\text{last}(x) \wedge \bigvee_{q_i \in F} x \in X_i \right)$$

iff w has a last letter, and $w \in L$. So we could take

$$\varphi_A := \begin{cases} \psi_A & \text{if } q_0 \notin F \\ \psi_A \vee \forall x\, x < x & \text{if } q_0 \in F \end{cases}$$

Let us now construct the formula $\text{Visits}(X_0, \ldots X_n)$. The sets P_q are the unique sets satisfying the following properties:

(a) $1 \in P_{\delta(q_0, a_1)}$, i.e., after reading the letter at position 1 the DFA is in state $\delta(q_0, a_1)$;

(b) every position i belongs to exactly one P_q, i.e., the P_q's build a partition of the set positions; and

(c) if $i \in P_q$ and $\delta(q, a_{i+1}) = q'$ then $i + 1 \in P_{q'}$, i.e., the P_q's "respect" the transition function δ.

We express these properties through formulas. For every $a \in \Sigma$, let $q_{i_a} = \delta(q_0, a)$. The formula for (a) is:

$$\text{Init}(X_0, \ldots, X_n) = \exists x \left(\text{first}(x) \wedge \left(\bigvee_{a \in \Sigma} (Q_a(x) \wedge x \in X_{i_a}) \right) \right)$$

(in words: if the letter at position 1 is a, then the position belongs to X_{i_a}).
Formula for (b):

$$\text{Partition}(X_0, \ldots, X_n) = \forall x \left(\bigvee_{i=0}^{n} x \in X_i \ \wedge \bigwedge_{\substack{i, j = 0 \\ i \neq j}}^{n} (x \in X_i \rightarrow x \notin X_j) \right)$$

Formula for (c):

$$\text{Respect}(X_0,\dots,X_n) = \forall x \forall y \left(y = x+1 \rightarrow \bigvee_{\substack{a \in \Sigma \\ i,j \in \{0,\dots,n\} \\ \delta(q_i,a)=q_j}} (x \in X_i \wedge Q_a(x) \wedge y \in X_j) \right)$$

Altogether we get

$$\text{Visits}(X_0,\dots X_n) := \text{Init}(X_0,\dots,X_n) \wedge \text{Partition}(X_0,\dots,X_n) \wedge \text{Respect}(X_0,\dots,X_n)$$

□

It remains to prove that MSO-definable languages are regular. Given a sentence $\varphi \in MSO(\Sigma)$ show that $L(\varphi)$ is regular by induction on the structure of φ. However, since the subformulas of a sentence are not necessarily sentences, the language defined by the subformulas of φ is not defined. We correct this. Recall that the interpretations of a formula are pairs $(w, \mathfrak{I})$ where $\mathfrak{I}$ assigns positions to the free first-order variables and sets of positions to the free second-order variables. For example, if $\Sigma = \{a, b\}$ and if the free first-order and second-order variables of the formula are x, y and X, Y, respectively, then two possible interpretations are

$$\left(aab\ ,\ \begin{array}{c} x \mapsto 1 \\ y \mapsto 3 \\ X \mapsto \{2,3\} \\ Y \mapsto \{1,2\} \end{array} \right) \qquad \left(ba\ ,\ \begin{array}{c} x \mapsto 2 \\ y \mapsto 1 \\ X \mapsto \emptyset \\ Y \mapsto \{1\} \end{array} \right)$$

Given an interpretation $(w, \mathfrak{I})$, we can encode each assignment $x \mapsto k$ or $X \mapsto \{k_1,\dots,k_l\}$ as a bitstring of the same length as w: the string for $x \mapsto k$ contains exactly a 1 at position k, and 0's everywhere else; the string for $X \mapsto \{k_1,\dots,k_l\}$ contains 1's at positions $k_1,\dots,k_l$, and 0's everywhere else. After fixing an order on the variables, an interpretation $(w, \mathfrak{I})$ can then be encoded as a tuple $(w, w_1,\dots,w_n)$, where n is the number of variables, $w \in \Sigma^*$, and $w_1,\dots,w_n \in \{0,1\}^*$. Since all of $w, w_1,\dots,w_n$ have the same length, we can as in the case of transducers look at $(w, w_1,\dots,w_n)$ as a word over the alphabet $\Sigma \times \{0,1\}^n$. For the two interpretations above we get the encodings

$$\begin{array}{c|ccc} & a & a & b \\ x & 1 & 0 & 0 \\ y & 0 & 0 & 1 \\ X & 0 & 1 & 1 \\ Y & 1 & 1 & 0 \end{array} \quad \text{and} \quad \begin{array}{c|cc} & b & a \\ x & 0 & 1 \\ y & 1 & 0 \\ X & 0 & 0 \\ Y & 1 & 0 \end{array}$$

corresponding to the words

$$\begin{bmatrix}a\\1\\0\\0\\1\end{bmatrix}\begin{bmatrix}a\\0\\0\\1\\1\end{bmatrix}\begin{bmatrix}b\\0\\1\\1\\0\end{bmatrix} \quad \text{and} \quad \begin{bmatrix}b\\0\\1\\0\\1\end{bmatrix}\begin{bmatrix}a\\1\\0\\0\\0\end{bmatrix} \quad \text{over } \Sigma \times \{0,1\}^4$$

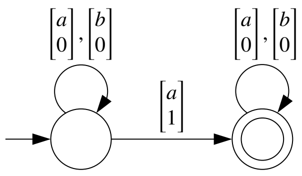
Definition 9.13 *Let φ be a formula with n free variables, and let $(w, \mathfrak{I})$ be an interpretation of φ. We denote by* $\mathrm{enc}(w, \mathfrak{I})$ *the word over the alphabet* $\Sigma \times \{0,1\}^n$ *described above. The language of φ is $L(\varphi) = \{\mathrm{enc}(w, \mathfrak{I}) \mid (w, \mathfrak{I}) \models \varphi\}$.*

Now that we have associated to every formula φ a language (whose alphabet depends on the free variables), we prove by induction on the structure of φ that $L(\varphi)$ is regular. We do so by exhibiting automata (actually, transducers) accepting $L(\varphi)$. For simplicity we assume $\Sigma = \{a, b\}$, and denote by *free*(φ) the set of free variables of φ.

- $\varphi = Q_a(x)$. Then *free*$(\varphi) = x$, and the interpretations of φ are encoded as words over $\Sigma \times \{0,1\}$. The language $L(\varphi)$ is given by

$$L(\varphi) = \left\{ \begin{bmatrix}a_1\\b_1\end{bmatrix} \begin{matrix}\cdots\\\cdots\end{matrix} \begin{bmatrix}a_k\\b_k\end{bmatrix} \;\middle|\; \begin{array}{l} k \geq 0, \\ a_i \in \Sigma \text{ and } b_i \in \{0,1\} \text{ for every } i \in \{1,\ldots,k\}, \text{ and} \\ b_i = 1 \text{ for exactly one index } i \in \{1,\ldots,k\} \text{ such that } a_i = a \end{array} \right\}$$

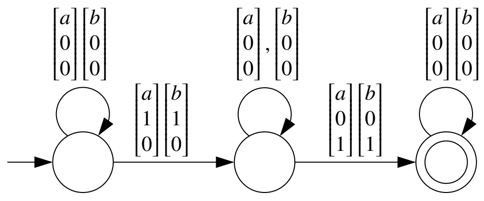
and is recognized by



- $\varphi = x < y$. Then *free*$(\varphi) = \{x, y\}$, and the interpretations of ϕ are encoded as words over $\Sigma \times \{0,1\}^2$. The language $L(\varphi)$ is given by

$$L(\varphi) = \left\{ \begin{bmatrix}a_1\\b_1\\c_1\end{bmatrix} \begin{matrix}\cdots\\\cdots\end{matrix} \begin{bmatrix}a_k\\b_k\\c_k\end{bmatrix} \;\middle|\; \begin{array}{l} k \geq 0, \\ a_i \in \Sigma \text{ and } b_i, c_i \in \{0,1\} \text{ for every } i \in \{1,\ldots,k\}, \\ b_i = 1 \text{ for exactly one index } i \in \{1,\ldots,k\}, \\ c_j = 1 \text{ for exactly one index } j \in \{1,\ldots,k\}, \text{ and} \\ i < j \end{array} \right\}$$

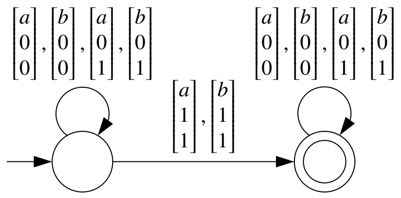
and is recognized by



- $\varphi = x \in X$. Then *free*$(\varphi) = \{x, X\}$, and interpretations are encoded as words over $\Sigma \times \{0,1\}^2$. The language $L(\varphi)$ is given by

$$L(\varphi) = \left\{ \begin{bmatrix} a_1 \\ b_1 \\ c_1 \end{bmatrix} \begin{matrix} \dots \\ \dots \end{matrix} \begin{bmatrix} a_k \\ b_k \\ c_k \end{bmatrix} \left| \begin{array}{l} k \geq 0, \\ a_i \in \Sigma \text{ and } b_i, c_i \in \{0,1\} \text{ for every } i \in \{1, \dots, k\}, \\ b_i = 1 \text{ for exactly one index } i \in \{1, \dots, k\}, \text{ and} \\ \text{for every } i \in \{1, \dots, k\}, \text{ if } b_i = 1 \text{ then } c_i = 1 \end{array} \right. \right\}$$

and is recognized by



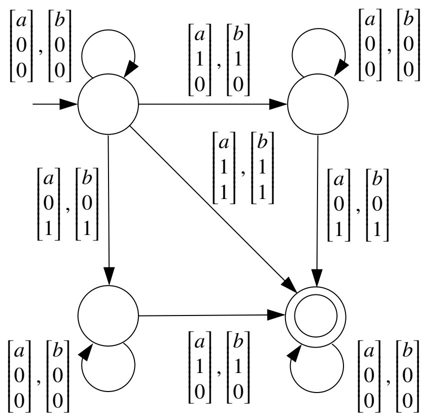
- $\varphi = \neg\psi$. Then *free*(φ) = *free*(ψ), and by induction hypothesis there exists an automaton A_ψ s.t. $L(A_\psi) = L(\psi)$.

  Observe that $L(\varphi)$ is *not* in general equal to $\overline{L(\psi)}$. To see why, consider for example the case $\psi = Q_a(x)$ and $\varphi = \neg Q_a(x)$. The word

$$\begin{bmatrix} a \\ 1 \end{bmatrix} \begin{bmatrix} a \\ 1 \end{bmatrix} \begin{bmatrix} a \\ 1 \end{bmatrix}$$

  belongs neither to $L(\psi)$ nor $L(\varphi)$, because it is not the encoding of any interpretation: the bitstring for x contains more than one 1. What holds is $L(\varphi) = \overline{L(\psi)} \cap Enc(\psi)$, where $Enc(\psi)$ is the language of the encodings of all the interpretations of ψ (whether they are models of ψ or not). We construct an automaton A_ψ^{enc} recognizing $Enc(\psi)$, and so we can take $A_\varphi = A_\psi \cap A_\psi^{enc}$.

  Assume ψ has k first-order variables. Then a word belongs to $Enc(\psi)$ iff each of its projections onto the 2nd, 3rd, ..., $(k+1)$-th component is a bitstring containing exactly one 1. As states of A_ψ^{enc} we take all the strings $\{0,1\}^k$. The intended meaning of a state, say state 101 for the case $k = 3$, is "the automaton has already read the 1's in the bitstrings of the first and third variables, but not yet read the 1 in the second." The initial and final states are 0^k and 1^k, respectively. The transitions are defined according to the intended meaning of the states. For instance, the automaton $A_{x<y}^{enc}$ is

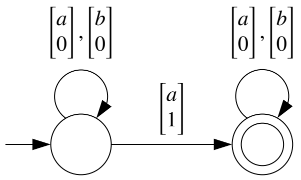


Observe that the number of states of A_{ψ}^{enc} grows exponentially in the number of free variables. This makes the negation operation expensive, even when the automaton A_{ϕ} is deterministic.

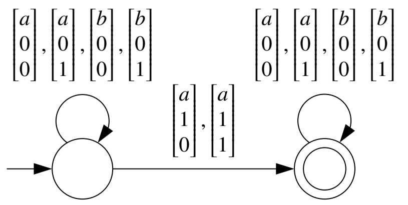
- $\varphi = \varphi_1 \vee \varphi_2$. Then *free*$(\varphi)$ = *free*$(\varphi_1) \cup$ *free*(φ_2), and by induction hypothesis there are automata $A_{\varphi_i}, A_{\varphi_2}$ such that $\mathcal{L}(A_{\varphi_1}) = \mathcal{L}(\varphi_1)$ and $\mathcal{L}(A_{\varphi_2}) = \mathcal{L}(\varphi_2)$.

  If *free*(φ_1) = *free*(φ_2), then we can take $A_\varphi = A_{\varphi_1} \cup A_{\varphi_2}$. But this need not be the case. If *free*$(\varphi_1) \neq$ *free*(φ_2), then $\mathcal{L}(\varphi_1)$ and $\mathcal{L}(\varphi_2)$ are languages over different alphabets Σ_1, Σ_2, or over the same alphabet, but with different intended meaning, and we cannot just compute their intersection. For example, if $\varphi_1 = Q_a(x)$ and $\varphi_2 = Q_b(y)$, then both $\mathcal{L}(\varphi_1)$ and $\mathcal{L}(\varphi_2)$ are languages over $\Sigma \times \{0, 1\}$, but the second component indicates in the first case the value of x, in the second the value of y.

  This problem is solved by extending $\mathcal{L}(\varphi_1)$ and $\mathcal{L}(A_{\varphi_2})$ to languages L_1 and L_2 over $\Sigma\times\{0, 1\}^2$. In our example, the language L_1 contains the encodings of all interpretations $(w, \{x \mapsto n_1, y \mapsto n_2\})$ such that the projection $(w, \{x \mapsto n_1\})$ belongs to $\mathcal{L}(Q_a(x))$, while L_2 contains the interpretations such that $(w, \{y \mapsto n_2\})$ belongs to $\mathcal{L}(Q_b(y))$. Now, given the automaton $A_{Q_a(x)}$ recognizing $\mathcal{L}(Q_a(x))$



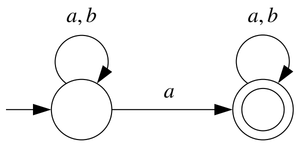
  we transform it into an automaton A_1 recognizing L_1



$\begin{bmatrix}a\\0\\0\end{bmatrix},\begin{bmatrix}a\\0\\1\end{bmatrix},\begin{bmatrix}b\\0\\0\end{bmatrix},\begin{bmatrix}b\\0\\1\end{bmatrix}$
$\begin{bmatrix}a\\1\\0\end{bmatrix},\begin{bmatrix}a\\1\\1\end{bmatrix}$
$\begin{bmatrix}a\\0\\0\end{bmatrix},\begin{bmatrix}a\\0\\1\end{bmatrix},\begin{bmatrix}b\\0\\0\end{bmatrix},\begin{bmatrix}b\\0\\1\end{bmatrix}$

After constructing A_2 similarly, take $A_\varphi = A_1 \cup A_2$.

- $\varphi = \exists x\ \psi$. Then $free(\varphi) = free(\psi) \setminus \{x\}$, and by induction hypothesis there is an automaton A_ψ s.t. $L(A_\psi) = L(\psi)$. Define $A_{\exists\, x\psi}$ as the result of the projection operation, where we project onto all variables but x. The operation simply corresponds to removing in each letter of each transition of A_σ the component for variable x. For example, the automaton $A_{\exists x\ Q_a(x)}$ is obtained by removing the second components in the automaton for $A_{Q_a(x)}$ shown above, yielding



Observe that the automaton for $\exists x\ \psi$ can be nondeterministic even if the one for ψ is deterministic, since the projection operation may map different letters into the same one.

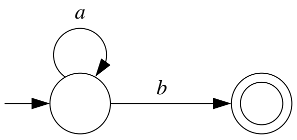
- $\varphi = \exists X\ \varphi$. We proceed as in the previous case.

Size of A_φ. The procedure for constructing A_φ proceeds bottom-up on the syntax tree of φ. We first construct automata for the atomic formulas in the leaves of the tree, and then proceed upwards: given automata for the children of a node in the tree, we construct an automaton for the node itself.

Whenever a node is labeled by a negation, the automaton for it can be exponentially bigger than the automaton for its only child. This yields an upper bound for the size of A_φ equal to a tower of exponentials, where the height of the tower is equal to the largest number of negations in any path from the root of the tree to one of its leaves.

It can be shown that this very large upper bound is essentially tight: there are formulas for which the smallest automaton recognizing the same language as the formula reaches the upper bound. This means that MSO-logic allows to describe some regular languages in an extremely *succinct* form.

Example 9.14 Consider the alphabet $\Sigma = \{a, b\}$ and the language $a^*b \subseteq \Sigma^*$, recognized by the NFA



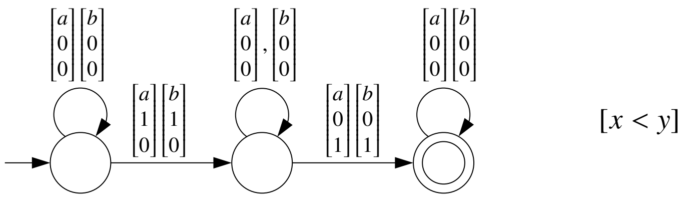
We derive this NFA by giving a formula φ such that $L(\varphi) = a^*b$, and then using the procedure described above. We shall see that the procedure is quite laborious. The formula states that the last letter is b, and all other letters are a's.

$$\varphi = \exists x \ (\text{last}(x) \land Q_b(x)) \ \land \ \forall x \ (\neg\text{last}(x) \rightarrow Q_a(x))$$

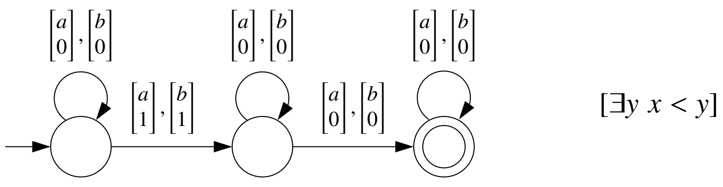
We first bring φ into the equivalent form

$$\psi = \exists x \ (\text{last}(x) \land Q_b(x)) \ \land \ \neg\exists x \ (\neg\text{last}(x) \land \neg Q_a(x))$$

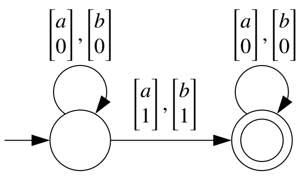
We transform ψ into an NFA. First, we compute an automaton for $\text{last}(x) \ = \ \neg\exists y \ x < y$. Recall that the automaton for $x < y$ is



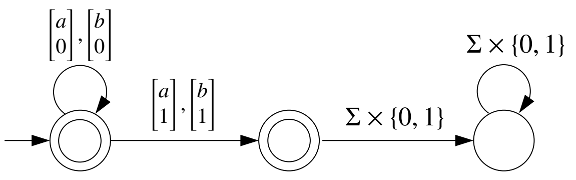
Applying the projection operation, we get following automaton for $\exists y \ x < y$



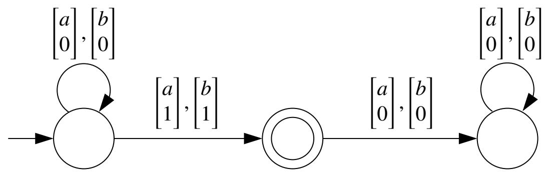
Recall that computing the automaton for the negation of a formula requires more than complementing the automaton. First,we need an automaton recognizing $Enc(\exists y \ x < y)$.



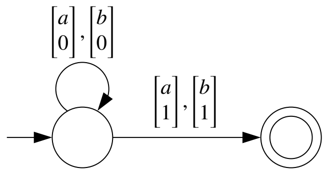
Second, we determinize and complement the automaton for $\exists y \ x < y$:



And finally, we compute the intersection of the last two automata, getting

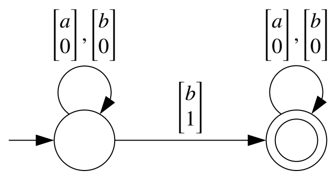

$\begin{bmatrix}a\\0\end{bmatrix},\begin{bmatrix}b\\0\end{bmatrix}$
$\begin{bmatrix}a\\1\end{bmatrix},\begin{bmatrix}b\\1\end{bmatrix}$
$\begin{bmatrix}a\\0\end{bmatrix},\begin{bmatrix}b\\0\end{bmatrix}$
$\begin{bmatrix}a\\0\end{bmatrix},\begin{bmatrix}b\\0\end{bmatrix}$

whose last state is useless and can be removed, yielding the following NFA for last(x):


$\begin{bmatrix}a\\0\end{bmatrix},\begin{bmatrix}b\\0\end{bmatrix}$
$\begin{bmatrix}a\\1\end{bmatrix},\begin{bmatrix}b\\1\end{bmatrix}$

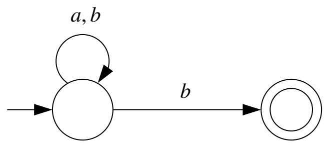
[last(x)]

Next we compute an automaton for $\exists x\ (\text{last}(x) \wedge Q_b(x))$, the first conjunct of ψ. We start with an NFA for $Q_b(x)$


$\begin{bmatrix}a\\0\end{bmatrix},\begin{bmatrix}b\\0\end{bmatrix}$
$\begin{bmatrix}b\\1\end{bmatrix}$
$\begin{bmatrix}a\\0\end{bmatrix},\begin{bmatrix}b\\0\end{bmatrix}$

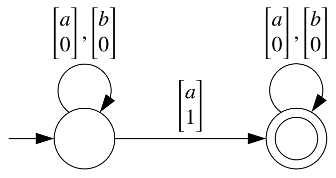
[$Q_b(x)$]

The automaton for $\exists x\ (\text{last}(x) \wedge Q_b(x))$ is the result of intersecting this automaton with the NFA for last(x) and projecting onto the first component. We get



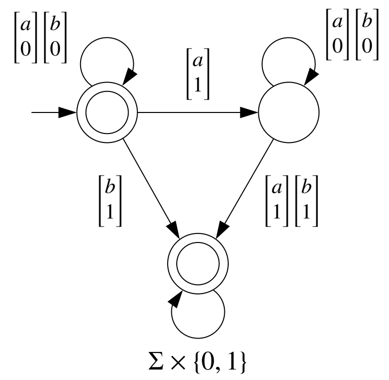
[$\exists x\ (\text{last}(x) \wedge Q_b(x))$]

Now we compute an automaton for $\neg\exists x\ (\neg\text{last}(x) \wedge \neg Q_a(x))$, the second conjunct of ψ. We first obtain an automaton for $\neg Q_a(x)$ by intersecting the complement of the automaton for $Q_a(x)$ and the automaton for $Enc(Q_a(x)$. The automaton for $Q_a(x)$ is

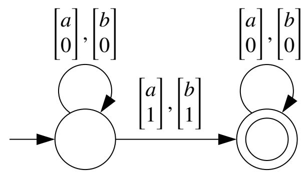

$\begin{bmatrix}a\\0\end{bmatrix},\begin{bmatrix}b\\0\end{bmatrix}$
$\begin{bmatrix}a\\1\end{bmatrix}$
$\begin{bmatrix}a\\0\end{bmatrix},\begin{bmatrix}b\\0\end{bmatrix}$

[$Q_a(x)$]

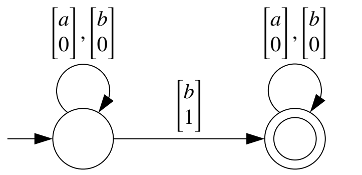
and after determinization and complementation we get


$\begin{bmatrix}a\\0\end{bmatrix}\begin{bmatrix}b\\0\end{bmatrix}$
$\begin{bmatrix}a\\1\end{bmatrix}$
$\begin{bmatrix}a\\0\end{bmatrix}\begin{bmatrix}b\\0\end{bmatrix}$
$\begin{bmatrix}b\\1\end{bmatrix}$
$\begin{bmatrix}a\\1\end{bmatrix}\begin{bmatrix}b\\1\end{bmatrix}$
$\Sigma \times \{0,1\}$

For the automaton recognizing $Enc(Q_a(x))$, notice that $Enc(Q_a(x)) = Enc(\exists y\ x < y)$, because both formulas have the same free variables, and so the same interpretations. But we have already computed an automaton for $Enc(\exists y\ x < y)$, namely


$\begin{bmatrix}a\\0\end{bmatrix}, \begin{bmatrix}b\\0\end{bmatrix}$
$\begin{bmatrix}a\\1\end{bmatrix}, \begin{bmatrix}b\\1\end{bmatrix}$
$\begin{bmatrix}a\\0\end{bmatrix}, \begin{bmatrix}b\\0\end{bmatrix}$

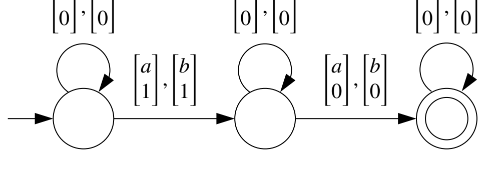
The intersection of the last two automata yields a three-state automaton for $\neg Q_a(x)$, but after eliminating a useless state we get


$\begin{bmatrix}a\\0\end{bmatrix}, \begin{bmatrix}b\\0\end{bmatrix}$
$\begin{bmatrix}b\\1\end{bmatrix}$
$\begin{bmatrix}a\\0\end{bmatrix}, \begin{bmatrix}b\\0\end{bmatrix}$

$[\neg Q_a(x)]$

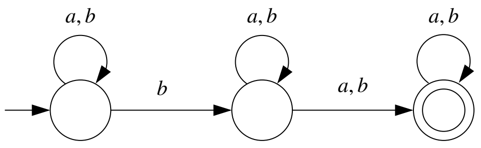
Notice that this is the same automaton we obtained for $Q_b(x)$, which is fine, because over the alphabet $\{a, b\}$ the formulas $Q_b(x)$ and $\neg Q_a(x)$ are equivalent.

To compute an automaton for $\neg last(x)$ we just observe that $\neg last(x)$ is equivalent to $\exists y\ x < y$, for which we have already compute an NFA, namely


$\begin{bmatrix}a\\0\end{bmatrix}, \begin{bmatrix}b\\0\end{bmatrix}$
$\begin{bmatrix}a\\1\end{bmatrix}, \begin{bmatrix}b\\1\end{bmatrix}$
$\begin{bmatrix}a\\0\end{bmatrix}, \begin{bmatrix}b\\0\end{bmatrix}$
$\begin{bmatrix}a\\0\end{bmatrix}, \begin{bmatrix}b\\0\end{bmatrix}$
$\begin{bmatrix}a\\0\end{bmatrix}, \begin{bmatrix}b\\0\end{bmatrix}$

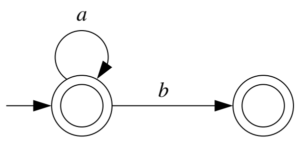
$[\neg last(x)]$

Intersecting the automata for $\neg last(x)$ and $\neg Q_a(x)$, and subsequently projecting onto the first component, we get an automaton for $\exists x\ (\neg last(x) \wedge \neg Q_a(x))$



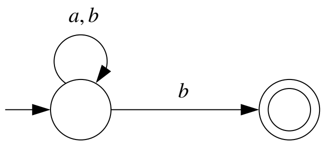
$[\exists x\ (\neg last(x) \wedge \neg Q_a(x))]$

Determinizing, complementing, and removing a useless state yields the following NFA for $\neg\exists x\ (\neg last(x) \wedge \neg Q_a(x))$:

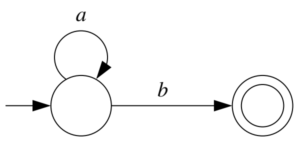


$[\neg\exists x\ (\neg last(x) \wedge \neg Q_a(x))]$

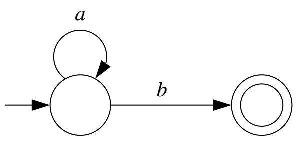
Summarizing, the automata for the two conjuncts of ψ are



and



whose intersection yields a 3-state automaton, which after removal of a useless state becomes



$[\exists x\ (\text{last}(x) \wedge Q_b(x)) \ \wedge\ \neg\exists x\ (\neg\text{last}(x) \wedge \neg Q_a(x))]$

ending the derivation. □

Exercises

Exercise 76 Characterize the languages described by the following formulas and give a corresponding automaton:

1. $\exists x\, first(x)$
2. $\forall x\, first(x)$
3. $\neg\exists x \exists y\ (x < y \wedge Q_a(x) \wedge Q_b(y)) \ \wedge\ \forall x\ (Q_b(x) \rightarrow \exists y\ x < y \wedge Q_a(y)) \ \wedge\ \exists x\, \neg\exists y\ (x < y \wedge Q_a(x))$

Exercise 77 Give a defining MSO-formula, an automaton, and a regular expression for the following languages over $\{a, b\}$.

- The set of words of even length and containing only a's or only b's.
- The set of words, where between each two b's with no other b in between there is a block of an odd number of letters a.
- The set of words with odd length and an odd number of occurrences of a.

Exercise 78 For every $n \geq 1$, give a FO-formula of polynomial length in n abbreviating $y = x+2^n$. (Notice that the abbreviation $y = x + k$ of page 9.1 has length $\mathcal{O}(k)$, and so it cannot be directly used.) Use it to give another FO-formula φ_n, also of of polynomial length in n, for the language $L_n = \{ww \in \{a, b\}^* \mid |w| = 2^n\}$.

Remark: Since the minimal DFA for L_n has 2^{2^n} states (Exercise 10), this shows that the number of states of a minimal automaton equivalent to a given FO-formula may be double exponential in the length of the formula.

Exercise 79 *MSO* over a unary alphabet can be used to automatically prove some simple properties of the natural numbers. Consider for instance the following property: every finite set of natural numbers has a minimal element[2]. It is easy to see that this property holds iff the formula

$$\forall Z \exists x \forall y\, (y \in Z \rightarrow (x \leq y \wedge x \in Z))$$

is a *tautology*, i.e., if it is satisfied by every word. Construct an automaton for the formula, and check that it is universal.

Exercise 80 Give formulas $\varphi_1, \ldots, \varphi_4$ for the following abbreviations:

$Sing(X)$	$:=$	φ_1	"X is a singleton, i.e., X contains one element"
$X \subseteq Y$	$:=$	φ_2	"X is a subset of Y"
$X \subseteq Q_a$	$:=$	φ_3	"every position of X contains an a"
$X < Y$	$:=$	φ_4	"X and Y are singletons $X = \{x\}$ and $Y = \{y\}$ satisfying $x < y$"

Exercise 81 Express addition in $MSO(\{a\}$. More precisely, find a formula $+(X, Y, Z)$ of $MSO(\{a\})$ that is true iff $x + y = z$, where x, y, z are the numbers encoded by the sets X, Y, Z, respectively, in *lsbf* encoding. You may use any abbrevation defined in the chapter.

[2]Of course, this also holds for every infinite set, but we cannot prove it using MSO over finite words.

Exercise 82 The *nesting depth* $d(\varphi)$ of a formula φ of $FO(\{a\})$ is defined inductively as follows:

- $d(Q_a(x)) = d(x < y) = 0$;
- $d(\neg\varphi) = d(\varphi)$, $d(\varphi_1 \vee \varphi_2) = \max\{d(\varphi_1), d(\varphi_2)\}$; and
- $d(\exists x\, \varphi) = 1 + d(\varphi)$.

Prove that every formula φ of $FO(\{a\})$ of nesting depth n is equivalent to a formula f of QF having the same free variables as φ, and such that every constant k appearing in f satisfies $k \geq 2^n$.

Hint: The proof is similar to that of Theorem 9.8. The difficult case is the one where φ has the form $\exists x\, \psi$ and ψ is a conjunction. Define f as the following conjunction. All conjuncts of ψ not containing x are also conjuncts of f; for every conjunct of D of the form $x \geq k$ or $x \geq y + k$, f contains a conjunct $last \geq k$; for every two conjuncts of D containing x, f contains a conjunct obtained by "quantifying x away": for example, if the conjuncts are $x \geq k_1$ and $y \geq x + k_2$, then f has the conjunct $y \geq k_1 + k_2$. Since the constants in the new conjuncts are the sum of two old constants, the new constants are bounded by $2 \cdot 2^d = 2^{d+1}$.