Logic

Logics on words

- Regular expressions give operational descriptions of regular languages.
- Often the natural description of a language is declarative:
  - **even number of a's and even number of b's** vs.

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

  - **words not containing ‘hello’**
- Goal: find a declarative language able to express all the regular languages, and only the regular languages.

Logics on words

- Idea: use a logic that has an interpretation on words
- A formula expresses a property that each word may satisfy or not, like
  - **the word contains only a's**
  - **the word has even length**
  - **between every occurrence of an a and a b there is an occurrence of a c**
- Every formula (indirectly) defines a language: the language of all the words over the given fixed alphabet that satisfy it.

First-order logic on words

- Atomic formulas:
  - for each letter a we introduce the formula $Q_a(x)$, with intuitive meaning: **the letter at position x is an a**.
  - for every two variables $x, y \in V$ we introduce the formula $x < y$ with intuitive meaning: **position x is to the left of position y**.

First-order logic on words: Syntax

- Formulas constructed out of atomic formulas by means of standard "logic machinery":
  - Alphabet $\Sigma = \{a, b, \ldots\}$ and position variables $V = \{x, y, \ldots\}$
  - $Q_a(x)$ is a formula for every $a \in \Sigma$ and $x \in V$.
  - $x < y$ is a formula for every $x, y \in V$
  - If $\varphi, \varphi_1, \varphi_2$ are formulas then so are $\neg\varphi$ and $\varphi_1 \vee \varphi_2$
  - If φ is a formula then so is $\exists x\ \varphi$ for every $x \in V$

Abbreviations

$$\varphi_1 \wedge \varphi_2 \quad \coloneqq \quad \neg(\neg\, \varphi_1 \vee \neg \varphi_2)$$

$$\varphi_1 \rightarrow \varphi_2 \quad \coloneqq \quad \neg \varphi_1 \vee \varphi_2$$

$$\varphi_1 \leftrightarrow \varphi_2 \quad \coloneqq \quad (\varphi_1 \wedge \varphi_2) \vee (\neg \varphi_1 \wedge \neg \varphi_2)$$

$$\forall x\, \varphi \quad \coloneqq \quad \neg\, \exists x\, \neg \varphi$$

Abbreviations

$$\mathrm{first}(x) \coloneqq \neg\exists y \; y < x$$

$$\mathrm{last}(x) \coloneqq \neg\exists y \; x < y$$

$$y = x + 1 \coloneqq x < y \wedge \neg\exists z\,(x < z \wedge z < y)$$

$$y = x + 2 \coloneqq \exists z\,(z = x + 1 \wedge y = z + 1)$$

$$\ldots$$

$$y = x + k \coloneqq \exists z\,(z = x + 1 \wedge y = z + (k - 1))$$

$$x < k \coloneqq \forall y \forall z\,(\mathrm{first}(y) \wedge z = y + k - 1) \rightarrow x < z)$$

$$\mathrm{last} < k \coloneqq \forall x\,(\mathrm{last}(x) \rightarrow x < k)$$

Examples (without semantics yet)

- "The last letter is a b and before it there are only a's."

- "Every a is immediately followed by a b."

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

- "Between every a and every later b there is a c."

Examples (without semantics yet)

- "The last letter is a b and before it there are only a's."

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

- "Every a is immediately followed by a b."

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

- "Between every a and every later b there is a c."

Examples (without semantics yet)

- "The last letter is a b and before it there are only a's."

$$\exists x\, Q_b(x) \land \forall x\, (\mathrm{last}(x) \rightarrow Q_b(x) \;\land\; \neg \mathrm{last}(x) \rightarrow Q_a(x))$$

- "Every a is immediately followed by a b."

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

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

- "Between every a and every later b there is a c."

Examples (without semantics yet)

- "The last letter is a b and before it there are only a's."

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

- "Every a is immediately followed by a b."

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

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

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

- "Between every a and every later b there is a c."

Examples (without semantics yet)

- "The last letter is a b and before it there are only a's."

$$\exists x\, Q_b(x) \wedge \forall x\, (\mathrm{last}(x) \to Q_b(x) \ \wedge \ \neg\mathrm{last}(x) \to Q_a(x))$$

- "Every a is immediately followed by a b."

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

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

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

- "Between every a and every later b there is a c."

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

First-order logic on words: Semantics

- Formulas are interpreted on pairs $(w, \mathcal{V})$ called interpretations, where
  - w is a word, and
  - $\mathcal{V}$ assigns positions to the free variables of the formula (and maybe to others too).
- It does not make sense to say a formula is true or false: it can only be true or false for a given interpretation.
- If the formula has no free variables (if it is a sentence), then for each word it is either true or false.

- Satisfaction relation:

$$
\begin{array}{lll}
(w, \mathcal{V}) \vDash Q_a(x) & \text{iff} & w[\mathcal{V}(x)] = a \\
(w, \mathcal{V}) \vDash x < y & \text{iff} & \mathcal{V}(x) < \mathcal{V}(y) \\
(w, \mathcal{V}) \vDash \neg\varphi & \text{iff} & w \nvDash \varphi \\
(w, \mathcal{V}) \vDash \varphi_1 \vee \varphi_2 & \text{iff} & w \vDash \varphi_1 \text{ or } w \vDash \varphi_2 \\
(w, \mathcal{V}) \vDash \exists x\, \varphi & \text{iff} & w \neq \epsilon \text{ and } (w, \mathcal{V}[i/x]) \vDash \varphi_2 \\
 & & \text{for some } 1 \leq i \leq |w|
\end{array}
$$

- Observe that the empty word does not satisfy any formula of the form $\exists x\, \varphi$

- More logic jargon:
  - A formula is valid if it is true for all its interpretations
  - A formula is satisfiable if is is true for at least one of its interpretations
  - Two formulas are equivalent if they have the same interpretations and the same models

Can FOL express non-regular languages? Can FOL express all regular languages?

- The language $L(\varphi)$ of a sentence φ is the set of words that satisfy φ.
- A language L is expressible in first-order logic or FO-definable if some sentence φ satisfies $L(\varphi) = L$.
- Proposition: a language over a one-letter alphabet is expressible in first-order logic iff it is finite or co-finite (its complement is finite).
- Consequence: we can only express regular languages, but not all, not even the language of words of even length.

Proof sketch

1. If L is finite, then it is FO-definable

2. If L is co-finite, then it is FO-definable.

Proof sketch

3. If L is FO-definable (over a one-letter alphabet), then it is finite or co-finite.
   1) We define a new logic QF (quantifier-free fragment)
   2) We show that a language is QF-definable iff it is finite or co-finite
   3) We show that a language is QF-definable iff it is FO-definable.

1) The logic QF

- $x < k$ $\quad x > k$

  $x < y + k$ $\quad x > y + k$

  $k <$ last $\quad k >$ last

  are formulas for every variable x, y and every $k \geq 0$.
- If f_1, f_2 are formulas, then so are $f_1 \vee f_2$ and $f_1 \wedge f_2$

2) L is QF-definable iff it is finite or co-finite

($\rightarrow$) Let f be a sentence of QF.

Then f is a positive boolean combination of formulas k < last and k > last.

$L(k < \text{last}) = \{k+1, k+2, \ldots\}$ is co-finite (we identify words and numbers)

$L(k > \text{last}) = \{0,1, \ldots, k\}$ is finite

$L(f_1 \vee f_2) = L(f_1) \cup L(f_2)$ and so if $L(f_1)$ and $L(f_2)$ finite or co-finite then L is finite or co-finite.

$L(f_1 \wedge f_2) = L(f_1) \cap L(f_2)$ and so if $L(f_1)$ and $L(f_2)$ finite or co-finite then L is finite or co-finite.

2) L is QF-definable iff it is finite or co-finite

($\leftarrow$) If $L = \{k_1, \dots, k_n\}$ is finite, then

$$(k_1 - 1 < \text{last} \land \text{last} < k_1 + 1) \lor \cdots \lor$$
$$(k_n - 1 < \text{last} \land \text{last} < k_n + 1)$$

expresses L.

If L is co-finite, then its complement is finite, and so expressed by some formula. We show that for every f some formula $\text{neg}(f)$ expresses $\overline{L(f)}$

- $\text{neg}(k < \text{last}) = (k - 1 < \text{last} \land \text{last} < k + 1) \lor \text{last} < k$
- $\text{neg}(f_1 \lor f_2) = \text{neg}(f_1) \land \text{neg}(f_2)$
- $\text{neg}(f_1 \land f_2) = \text{neg}(f_1) \lor \text{neg}(f_2)$

3) Every first-order formula φ has an equivalent QF-formula $QF(\varphi)$

- $QF(x < y) = x < y + 0$
- $QF(\neg\varphi) = \text{neg}\big(QF(\varphi)\big)$
- $QF(\varphi_1 \vee \varphi_2) = QF(\varphi_1) \vee QF(\varphi_2)$
- $QF(\varphi_1 \wedge \varphi_2) = QF(\varphi_1) \wedge QF(\varphi_2)$
- $QF(\exists x\ \varphi) =$
  - Put $QF(\varphi)$ in disjunctive normal form. Assume $QF(\varphi)$= ($\varphi_1 \vee \ldots \vee \varphi_n$), where each φ_i is a conjunction of atomic formulas.
  - Since ∃x ($\varphi_1 \vee \ldots \vee \varphi_n$) ≡ ∃x $\varphi_1 \vee \ \ldots\ \vee$ ∃x φ_n, it suffices to define $QF(\exists x\ \varphi)$ for the case in which φ is a conjunction of atomic formulas of QF
  - For this case, see example in the next slide.

- Consider the formula

$$\exists x \quad x < y + 3 \quad \wedge$$
$$z < x + 4 \quad \wedge$$
$$z < y + 2 \quad \wedge$$
$$y < x + 1$$

- The equivalent QF-formula is

$$z < y + 8 \;\; \wedge \;\; y < y + 5 \;\; \wedge \;\; z < y + 2$$

Monadic second-order logic (MSOL)

- First-order variables: interpreted on positions
- Monadic second-order variables: interpreted on sets of positions.
  - Diadic second-order variables: interpreted on relations over positions
  - Monadic third-order variables: interpreted on sets of sets of positions
- New atomic formula: $x \in X$
- New quantification: $\exists X \varphi$

Expressing „even length“

- Express

  There is a set X of positions such that
  - **X contains exactly the even positions, and**
  - **the last position belongs to X.**
- Express

  X contains exactly the even positions

  as

A position is in X iff it is the second position or the second successor of another position of X

Syntax and semantics of MSOL

- New set $\{X, Y, Z, \ldots\}$ of second-order variables
- New syntax: $x \in X$ and $\exists X\ \varphi$
- New semantics:
  - Interpretations now also assign sets of positions to the free second-order variables.
  - Satisfaction defined as expected.

Expressing „even length“

- $\text{second}(x) = \exists y\,(\text{first}(y) \wedge x = y + 1)$
- $\text{Even}(X) = \forall y \left(x \in X \leftrightarrow \begin{pmatrix} \text{second}(x) \\ \vee \\ \exists y\,(x = y + 2 \wedge y \in X) \end{pmatrix} \right)$
- $\text{EvenLength} = \exists X \begin{pmatrix} \text{Even}(X) \\ \wedge \\ \forall x\,(\text{last}(x) \rightarrow x \in X) \end{pmatrix}$

Expressing $c^*(ab)^*d^*$

- Express:

  There is a block X of consecutive positions such that
  - **before X there are only c‘s;**
  - **after X there are only d‘s;**
  - **a‘s and b‘s alternate in X;**
  - **the first letter in X is an a, and the last is a b.**
- Then we can take the formula

$$\exists X \begin{pmatrix} \mathrm{Block}(X) \wedge \mathrm{Boc}(X) \wedge \mathrm{Aod}(X) \\ \wedge \\ \mathrm{Alt}(X) \wedge \mathrm{Fa}(X) \wedge \mathrm{Lb}(X) \end{pmatrix}$$

- X is a block of consecutive positions

- Before X there are only c's

- In X a's and b's alternate

- X **is a block of consecutive positions**

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

- **Before** X **there are only** c**'s**

- **In** X a**'s and** b**'s alternate**

- **X is a block of consecutive positions**

$$\text{Block}(X) := \forall x \in X \;\; \forall y \in X \;\; \forall z \; ((x < z \wedge z < y) \rightarrow z \in X)$$

- **Before X there are only c‘s**

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

$$\text{Boc}\ (X) \; := \; \forall x \; (\text{Before}(x, X) \rightarrow Q_c(x))$$

- **In X a‘s and b‘s alternate**

- X **is a block of consecutive positions**

$$\text{Block}(X) := \forall x \in X \;\; \forall y \in X \;\; \forall z \; ((x < z \wedge z < y) \to z \in X)$$

- **Before** X **there are only** c**'s**

$$\begin{aligned} \text{Before}(x, X) \;\; &:= \;\; \forall y \in X \; x < y \\ \text{Boc}\,(X) \;\; &:= \;\; \forall x \; (\text{Before}(x, X) \to Q_c(x)) \end{aligned}$$

- **In** X a**'s and** b**'s alternate**

$$\text{Alt}\,(X) := \forall x \in X \, \forall y \in X \begin{pmatrix} y = x + 1 \\ \to \\ \big(Q_a(x) \wedge Q_b(y)\big) \vee \big(Q_b(x) \wedge Q_a(y)\big) \end{pmatrix}$$

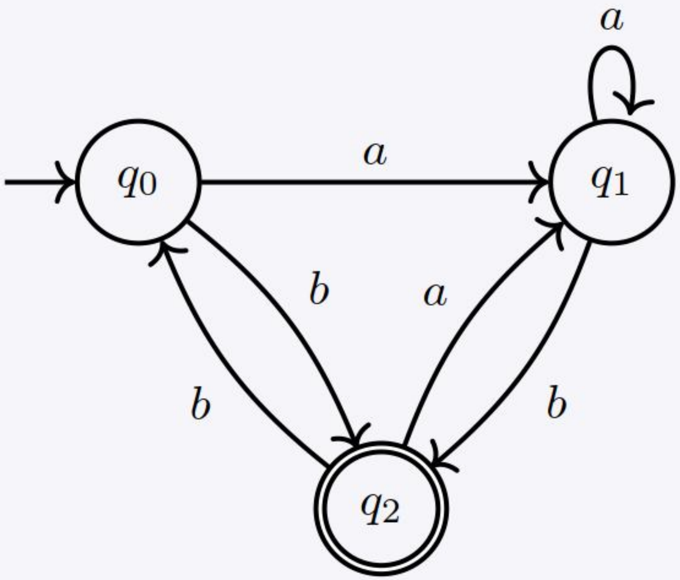
Every regular language is expressible in MSOL

- Goal: given an arbitrary regular language L, construct an MSO sentence φ s.t. $L = L(\varphi)$.
- It suffices to construct φ s.t. $w \in L$ iff $w \in L(\varphi)$ for every nonempty word w.
  (Avoid the corner-case of the empty word.)
- We use: if L is regular, then there is a DFA A recognizing L.
- Idea: construct a formula expressing

the run of A on this word ends in an accepting state

- Fix a regular language L.
- Fix a DFA A with states $q_0, \ldots, q_n$ recognizing L.
- Fix a nonempty word $w = a_1 a_2 \ldots a_m$.
- Let $R(q)$ be the set of positions i such that after reading $a_1 a_2 \ldots a_i$ the automaton A is in state q.
- We have:

  A accepts w iff $m \in R(q)$ for some final state q.



Run:	q_0	$\xrightarrow{a} q_1$	$\xrightarrow{a} q_1$	$\xrightarrow{b} q_2$	$\xrightarrow{b} q_0$	$\xrightarrow{b} q_2$
Position:		1	2	3	4	5

$$R_w(q_0) = \{4\}$$
$$R_w(q_1) = \{1, 2\}$$
$$R_w(q_2) = \{3, 5\}$$

- Assume we can construct a formula

$$\text{Visits}(X_0, \ldots, X_n)$$

which is true for $(w, \mathcal{I})$ iff

$$\mathcal{I}(X_0) = R(q_0), \ldots, \mathcal{I}(X_n) = R(q_n)$$

- Then $(w, \mathcal{I})$ satisfies the formula

$$\forall X_0 \cdots \forall X_n \, \forall x \left((\text{Visits}(X_0, \ldots, X_n) \wedge \text{last}(x)) \rightarrow \bigvee_{q_i \in F} x \in X_i \right)$$

iff the state after the last position is accepting, and we easily get a formula expressing L .

- To construct $\text{Visits}(X_0, \ldots, X_n)$ we observe that the sets $R(q)$ are the unique sets satisfying
  a) $1 \in R(\delta(q_0, a_1))$
  After reading the first letter the DFA is in state $\delta(q_0, a_1)$.
  b) If $i \in R(q)$ then $i + 1 \in R(q')$ iff $\delta(q, a_{i+1}) = q'$
  The sets „match“ δ.

- We give formulas for a) and b).

Formula for a):

$$\mathrm{InX}_i(x) := \left(x \in X_i \wedge \bigwedge_{j \neq i} x \notin X_j \right)$$

$$\mathrm{Init}(X_0, \ldots, X_n) := \forall x \bigwedge_{a \in \Sigma} \left((\mathrm{first}(x) \wedge Q_a(x)) \to \mathrm{InX}_{\delta(0,a)}(x) \right)$$

Formula for b):

$$\mathrm{Respect}(X_0, \ldots, X_n) :=$$

$$\forall x \, \forall y \left(y = x + 1 \to \bigwedge_{\substack{a \in \Sigma, \\ i \in \{0, \ldots, n\}}} (Q_a(x) \wedge x \in X_i) \to \mathrm{InX}_{\delta(i,a)}(y) \right)$$

Every language expressible in MSOL is regular

- An interpretation of a formula is a pair $(w, \mathcal{V}_1, \mathcal{V}_2)$ consisting of a word w and assignments $\mathcal{V}_1, \mathcal{V}_2$ to the free first and second-order variables (and perhaps to others).

$$\left(aab, \begin{Bmatrix} x \mapsto 1 \\ y \mapsto 3 \end{Bmatrix}, \begin{Bmatrix} X \mapsto \{2,3\} \\ Y \mapsto \{1,2\} \end{Bmatrix} \right)$$

$$\left(ba, \begin{Bmatrix} x \mapsto 2 \\ y \mapsto 1 \end{Bmatrix}, \begin{Bmatrix} X \mapsto \emptyset \\ Y \mapsto \{1\} \end{Bmatrix} \right)$$

- We encode interpretations as words.

$$\left(aab, \begin{Bmatrix} x \mapsto 1 \\ y \mapsto 3 \end{Bmatrix}, \begin{Bmatrix} X \mapsto \{2,3\} \\ Y \mapsto \{1,2\} \end{Bmatrix}\right)$$

$$\begin{array}{rccc} w: & a & a & b \\ x: & 1 & 0 & 0 \\ y: & 0 & 0 & 1 \\ X: & 0 & 1 & 1 \\ Y: & 1 & 1 & 0 \end{array}$$

$$\left(ba, \begin{Bmatrix} x \mapsto 2 \\ y \mapsto 1 \end{Bmatrix}, \begin{Bmatrix} X \mapsto \emptyset \\ Y \mapsto \{1\} \end{Bmatrix}\right)$$

$$\begin{array}{rcc} w: & b & a \\ x: & 0 & 1 \\ y: & 1 & 0 \\ X: & 0 & 0 \\ Y: & 1 & 0 \end{array}$$

- Given a formula with n free variables, we encode an interpretation $(w, \mathcal{V}_1, \mathcal{V}_2)$ as a word $enc(w, \mathcal{V}_1, \mathcal{V}_2)$ over the alphabet $\Sigma \times \{0,1\}^n$.

- The language of the formula φ , denoted by $L(\varphi)$, is given by

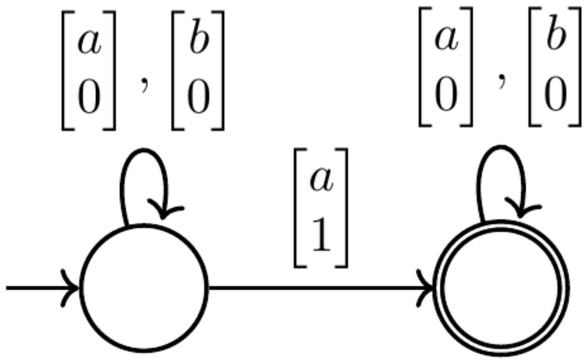
$$L(\varphi) := \{enc(w, \mathcal{V}_1, \mathcal{V}_2) \mid (w, \mathcal{V}_1, \mathcal{V}_2) \models \varphi\}$$

- We prove by induction on the structure of φ that $L(\varphi)$ is regular (and explicitely construct an automaton for it).

Case $\varphi = Q_a(x)$

- φ has one free variable, and so its interpretations are encoded as words over $\Sigma \times \{0,1\}$

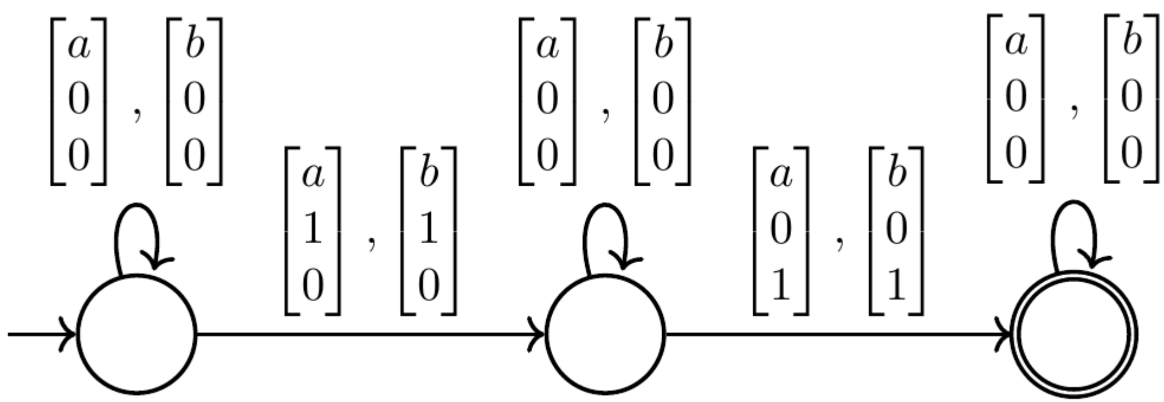
$$\mathcal{L}(\varphi) = \left\{ \begin{bmatrix} a_1 \\ \beta_1 \end{bmatrix} \begin{matrix} \cdots \\ \cdots \end{matrix} \begin{bmatrix} a_k \\ \beta_k \end{bmatrix} : \begin{array}{l} k \geq 1; \\ a_1 \dots a_k \in \Sigma^k, \beta_1 \dots \beta_k \in \{0,1\}^k; \text{ and} \\ \beta_i = 1 \text{ for a single index } i \in \{1, \dots, k\} \\ \text{such that } a_i = a. \end{array} \right\}$$



Case $\varphi = x < y$

- φ has two free variables, and so its interpretations are encoded as words over $\Sigma \times \{0,1\}^2$

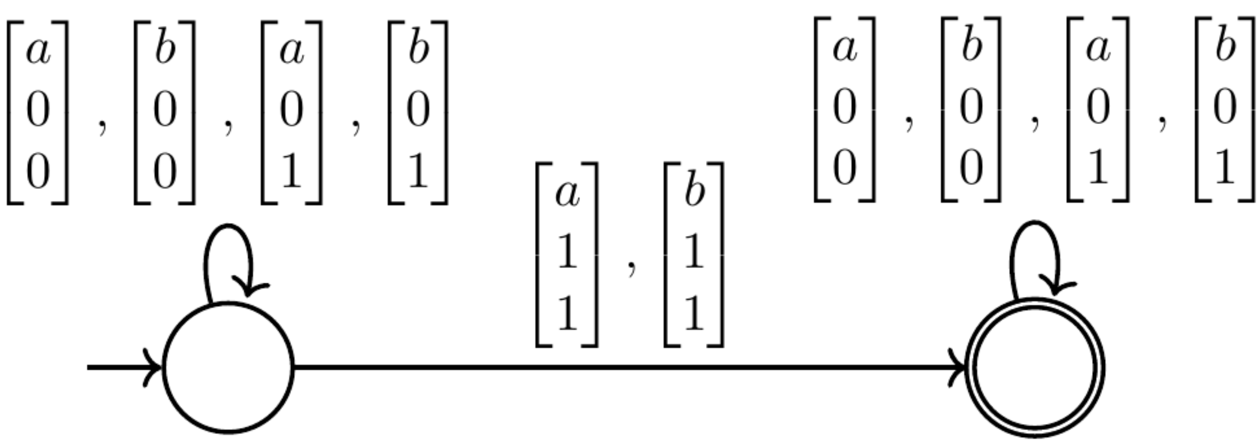
$$\mathcal{L}(\varphi) = \left\{ \begin{bmatrix} a_1 \\ \beta_1 \\ \gamma_1 \end{bmatrix} \begin{matrix} \cdots \\ \cdots \\ \cdots \end{matrix} \begin{bmatrix} a_k \\ \beta_k \\ \gamma_k \end{bmatrix} : \begin{array}{l} k \geq 1; \\ a_1 \ldots a_k \in \Sigma^k, \beta_1 \ldots \beta_k, c_1 \ldots c_k \in \{0,1\}^k; \\ \beta_i = 1 \text{ for a single index } i \in \{1, \ldots, k\}; \\ \gamma_j = 1 \text{ for a single index } j \in \{1, \ldots, k\}; \text{ and} \\ i < j. \end{array} \right\}$$



Case $\varphi = x \in X$

- φ has two free variables, and so its interpretations are encoded as words over $\Sigma \times \{0,1\}^2$

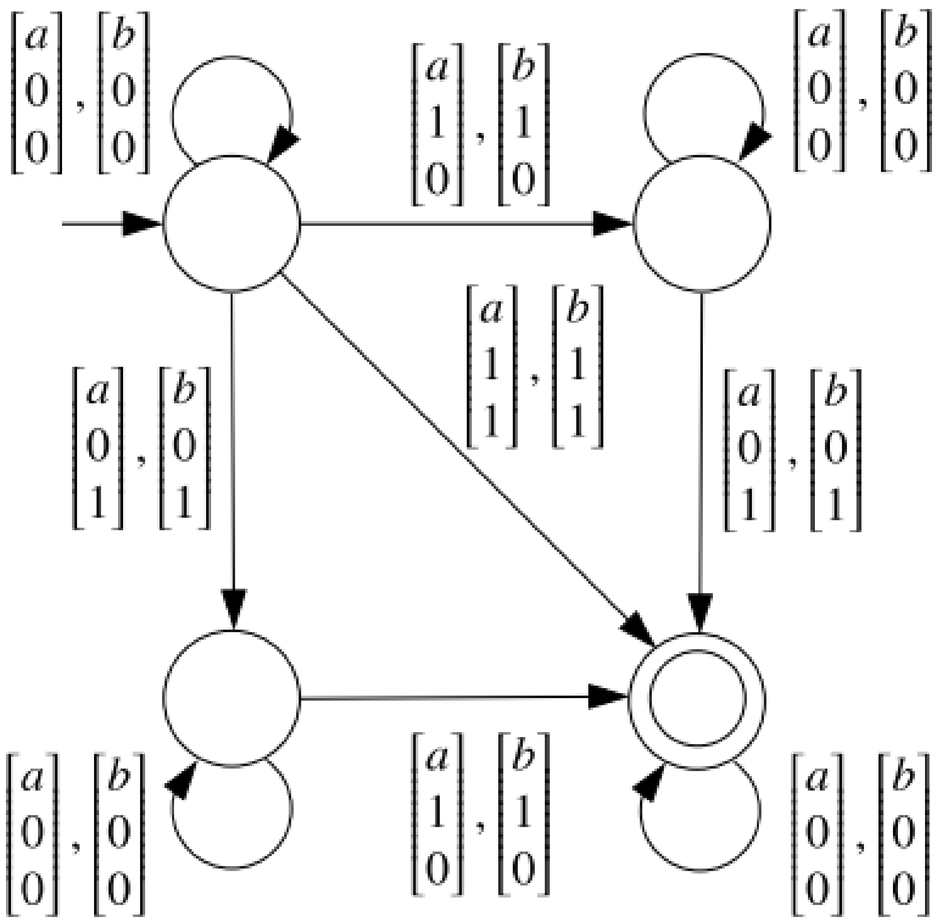
$$\mathcal{L}(\varphi) = \left\{ \begin{bmatrix} a_1 \\ \beta_1 \\ c_1 \end{bmatrix} \begin{matrix} \cdots \\ \cdots \\ \cdots \end{matrix} \begin{bmatrix} a_k \\ \beta_k \\ c_k \end{bmatrix} : \begin{array}{l} k \geq 1, \\ a_1 \ldots a_k \in \Sigma^k, \beta_1 \ldots \beta_k, \gamma_1 \ldots \gamma_k \in \{0,1\}^k; \\ \beta_i = 1 \text{ for a single index } i \in \{1, \ldots, k\}; \text{ and} \\ \beta_i = 1 \text{ implies } \gamma_i = 1 \text{ for all } i \in \{1, \ldots, k\}. \end{array} \right\}$$



Case $\varphi = \neg\psi$

- Then free(φ) = free(ψ) . By i.h. $L(\psi)$ is regular.
- $L(\varphi)$ is equal to $\overline{L(\psi)}$ minus the words that do not encode any implementation („the garbage“).
- Equivalently, $L(\varphi)$ is equal to the intersection of $\overline{L(\psi)}$ and the encodings of all interpretations of ψ.
- We show that the set of these encodings is regular.
  – Condition for encoding: Let x be a free first-oder variable of ψ . The projection of an encoding onto x must belong to 0^*10^* (because it represents one position).
  – So we just need an automaton for the words satisfying this condition for every free first-order variable.

Example: free(φ) = $\{x, y\}$

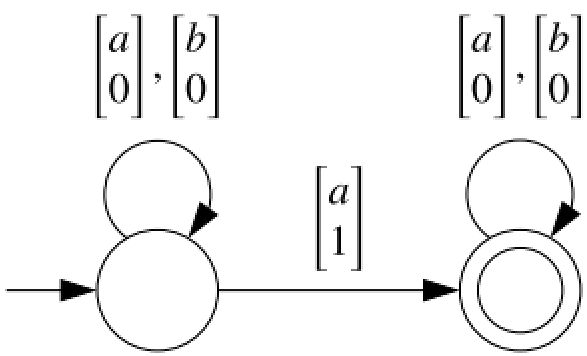


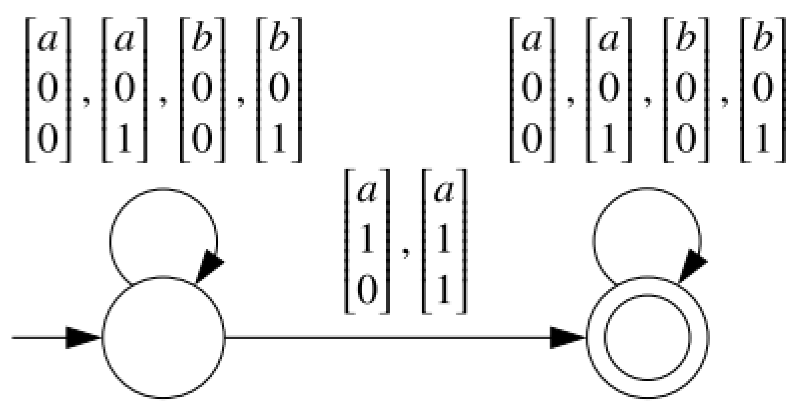
Case $\varphi = \varphi_1 \vee \varphi_2$

- Then $\text{free}(\varphi) = \text{free}(\varphi_1) \cup \text{free}(\varphi_2)$. By i.h. $L(\varphi_1)$ and $L(\varphi_2)$ are regular.
- If $\text{free}(\varphi_1) = \text{free}(\varphi_2)$ then $L(\varphi) = L(\varphi_1) \cup L(\varphi_2)$ and so $L(\varphi)$ is regular.
- If $\text{free}(\varphi_1) \neq \text{free}(\varphi_2)$ then we extend $L(\varphi_1)$ to L_1 encoding all interpretations of $\text{free}(\varphi_1) \cup \text{free}(\varphi_2)$ whose projection onto $\text{free}(\varphi_1)$ belongs to $L(\varphi_1)$. Similarly we extend $L(\varphi_2)$ to L_2. We have
  - L_1 and L_2 are regular.
  - $L(\varphi) = (L_1 \cup L_2) \cap Enc(\varphi)$, where $Enc(\varphi)$ is the set of encodings of all interpretations of φ.

Example: $\varphi = Q_a(x) \vee Q_b(y)$

- L_1 contains the encodings of all interpretations $(w, \{x \mapsto n_1, y \mapsto n_2\})$ such that the encoding of $(w, \{x \mapsto n_1\})$ belongs to $L(Q_a(x))$.
- Automata for $L(Q_a(x))$ and L_1:



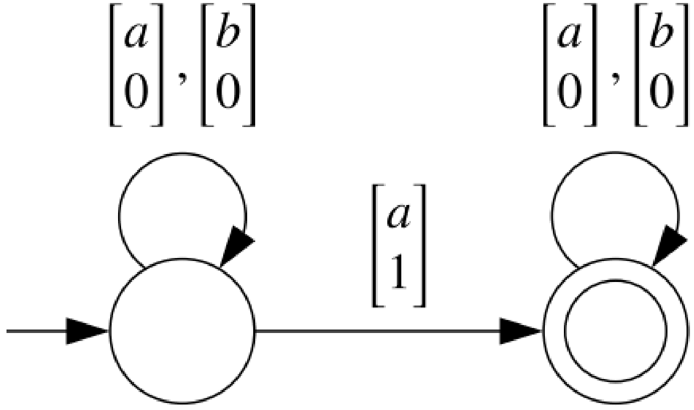


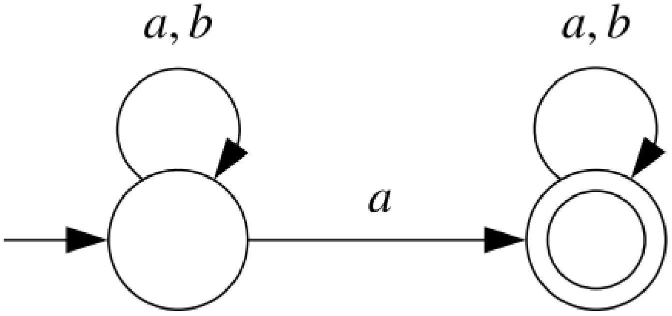
Cases $\varphi = \exists x\ \psi$ and $\varphi = \exists X\ \psi$

- Then free(φ)= free(ψ)\ $\{x\}$ or
  free(φ)= free(ψ)\ $\{X\}$
- By i.h. $L(\psi)$ is regular.
- $L(\varphi)$ is the result of projecting $L(\psi)$ onto the components for free(ψ)\ $\{x\}$ or for free(ψ)\ $\{X\}$.

Example: $\varphi = Q_a(x)$

- Automata for $Q_a(x)$ and $\exists x\, Q_a(x)$



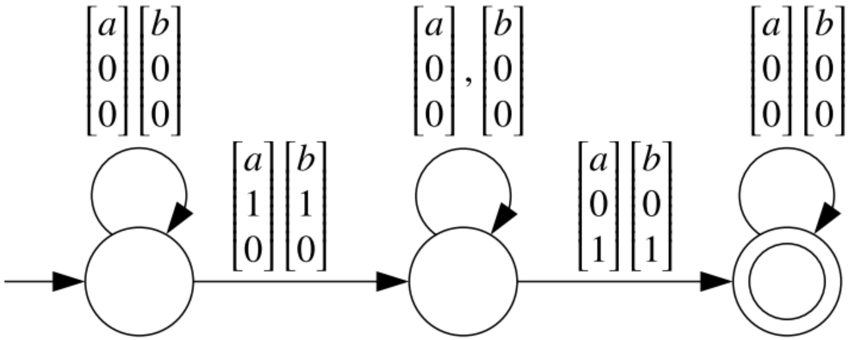


The mega-example

- We compute an automaton for
  $\exists x\,\big(\mathrm{last}(x) \land Q_b(x)\big) \land \forall x\,\big(\neg\mathrm{last}(x) \to Q_a(x)\big)$
- First we rewrite it into
  $\exists x\,\big(\mathrm{last}(x) \land Q_b(x)\big) \land \neg\exists x\,\big(\neg\mathrm{last}(x) \land \neg Q_a(x)\big)$
- In the next slides we
  1. compute a DFA for $\mathrm{last}(x)$
  2. compute DFAs for $\exists x\,\big(\mathrm{last}(x) \land Q_b(x)\big)$ and $\neg\exists x\,\big(\neg\mathrm{last}(x) \land \neg Q_a(x)\big)$
  3. compute a DFA for the complete formula.
- We denote the DFA for a formula ψ by $[\psi]$.

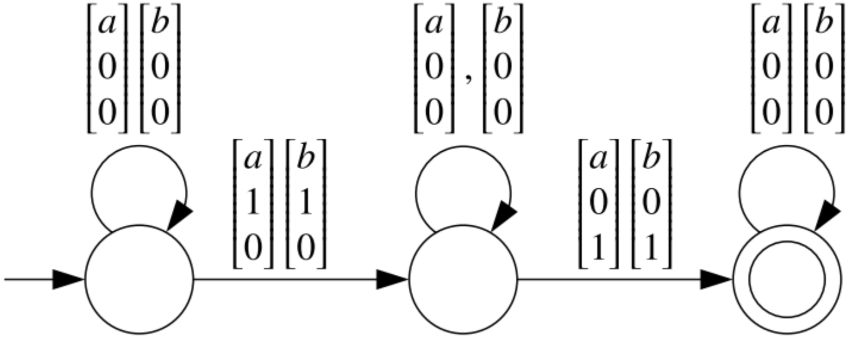
$[\mathrm{last}(x)]$

$[x < y]$

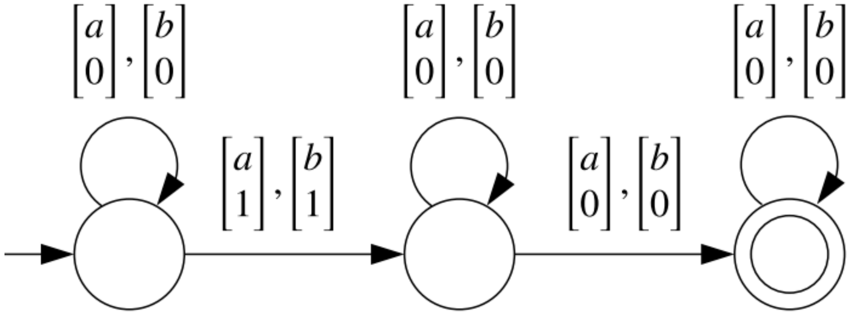


$[\mathrm{last}(x)]$

$[x < y]$

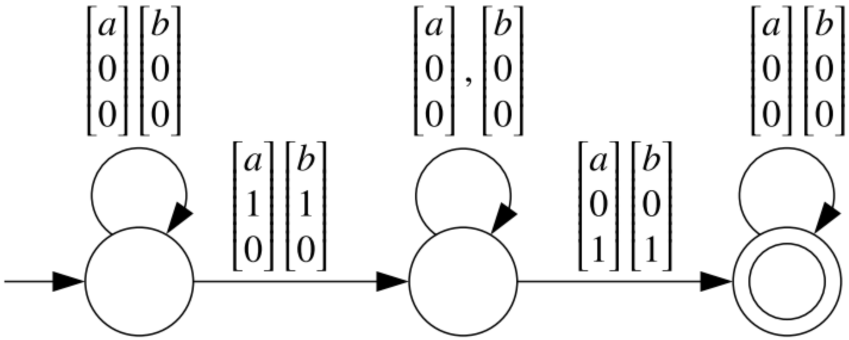


$[\exists y\, x < y]$

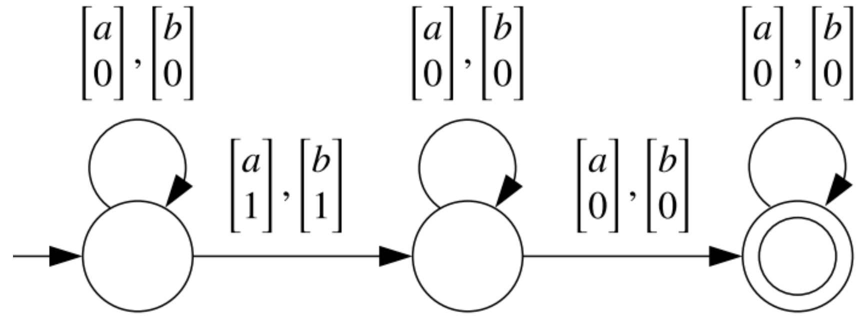


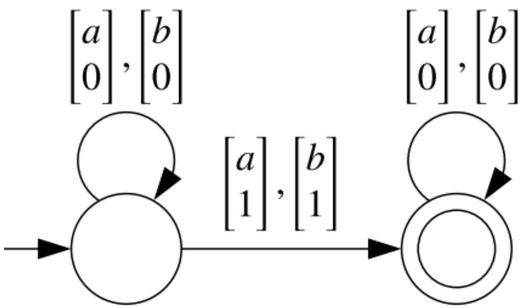
[last(x)]

$[x < y]$

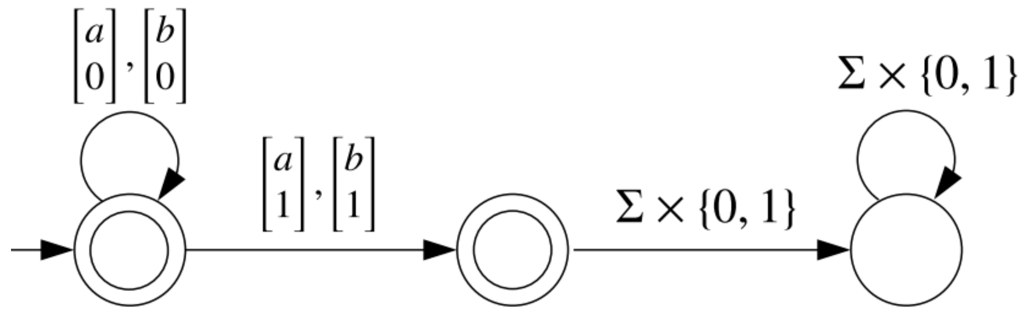


$[\exists y\ x < y]$





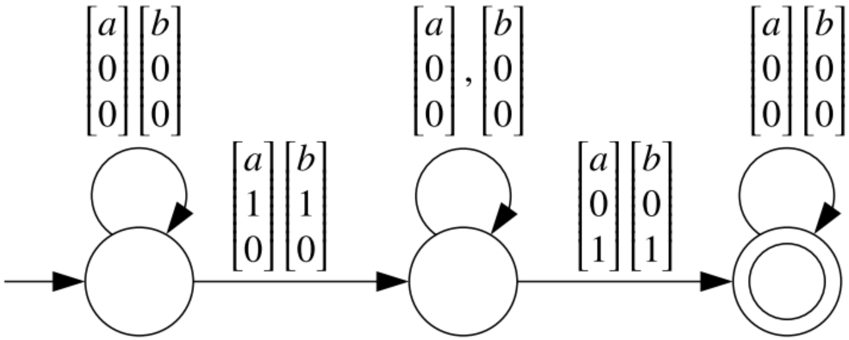
$[Enc(\exists y\ x < y)]$



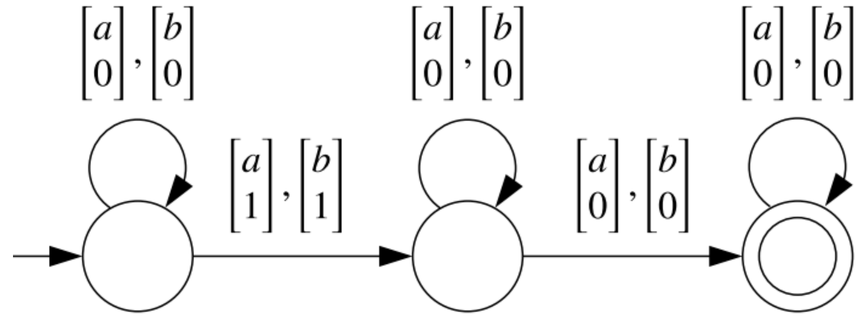
$\overline{[\exists y\ x < y]}$

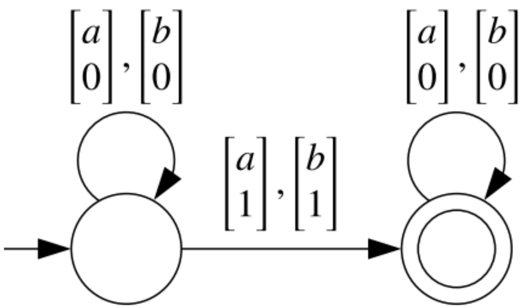
[last(x)]

[$x < y$]

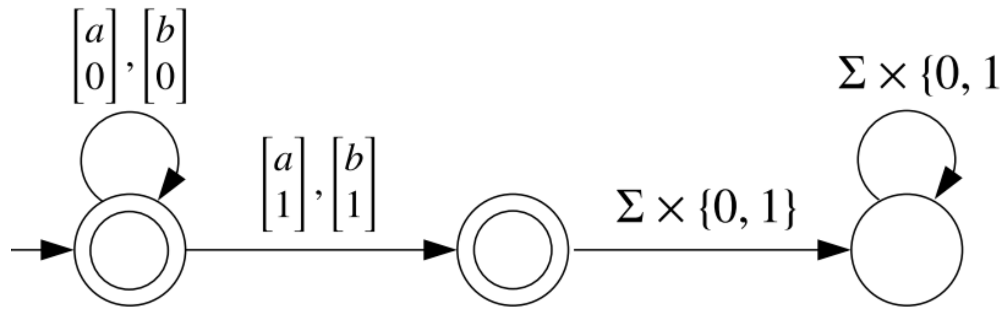


[$\exists y\ x < y$]

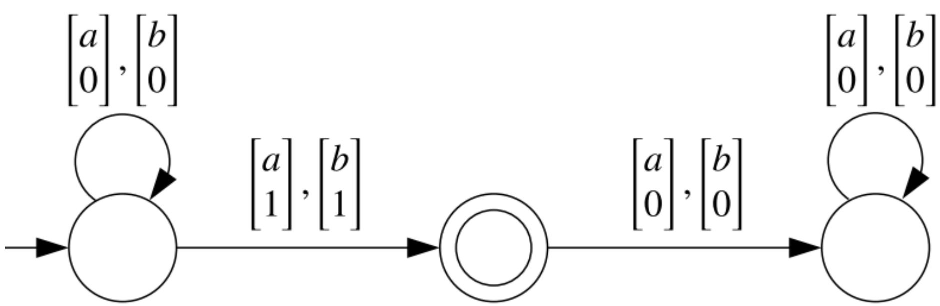


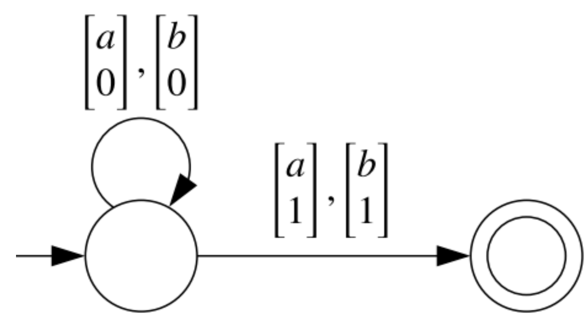


[$Enc(\exists y\ x < y)$]



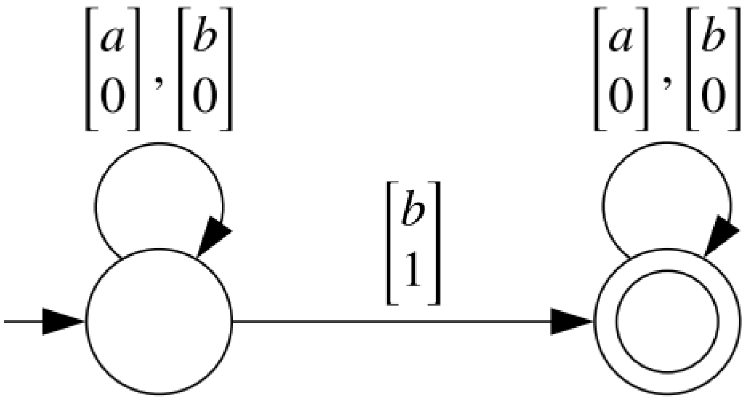
$\overline{[\exists y\ x < y]}$



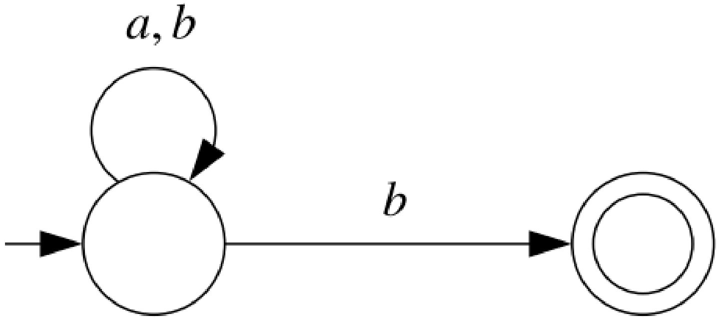


[last(x)]

$\left[\exists x\left(\mathrm{last}(x) \wedge Q_b(x)\right)\right]$

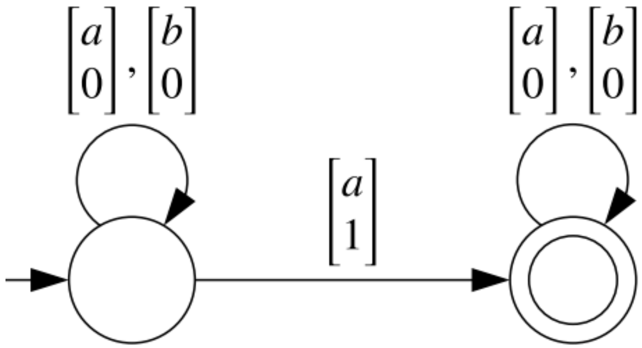


$\left[\boldsymbol{Q_b(x)}\right]$

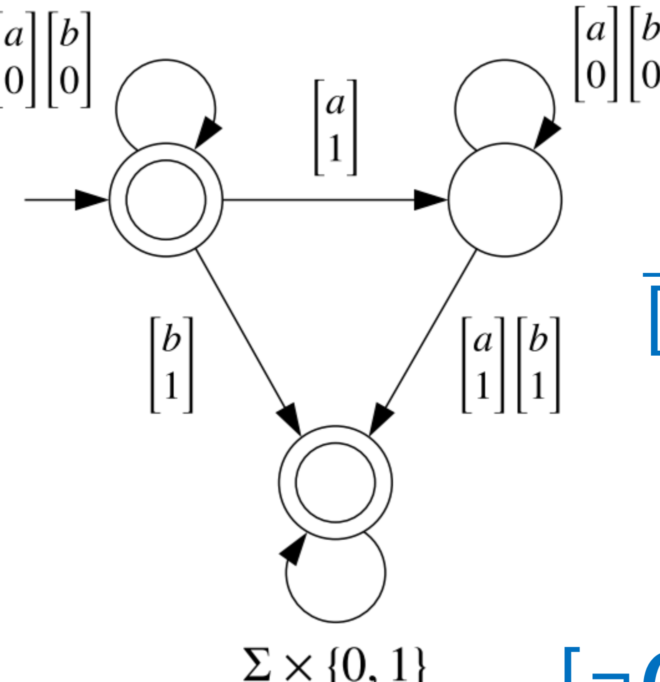


$\left[\exists \boldsymbol{x}\left(\mathbf{last}(\boldsymbol{x}) \wedge \boldsymbol{Q_b}(\boldsymbol{x})\right)\right]$

$[\neg Q_a(x)]$

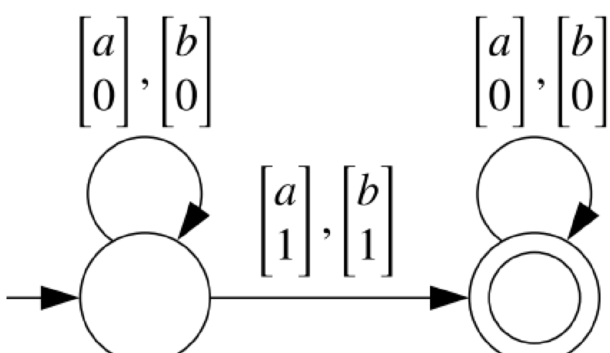


$[Q_a(x)]$

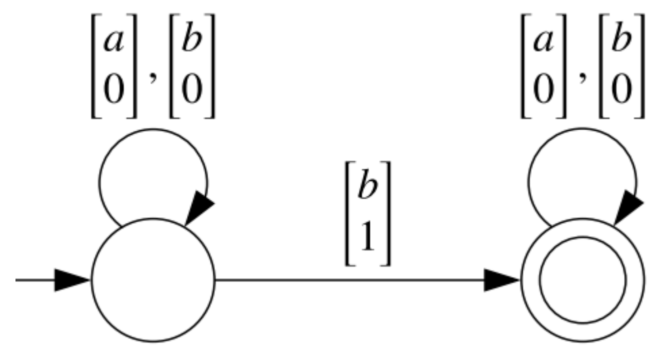


$\overline{[Q_a(x)]}$

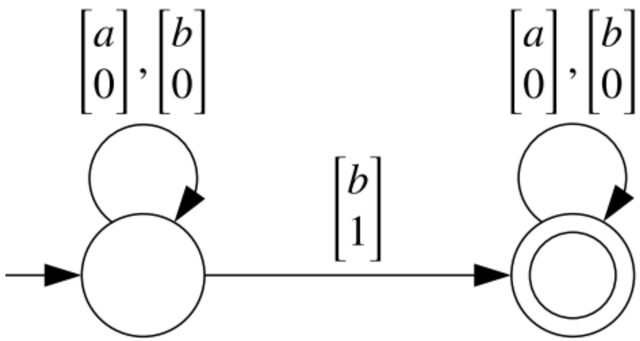
$[Enc(Q_a(x))]$



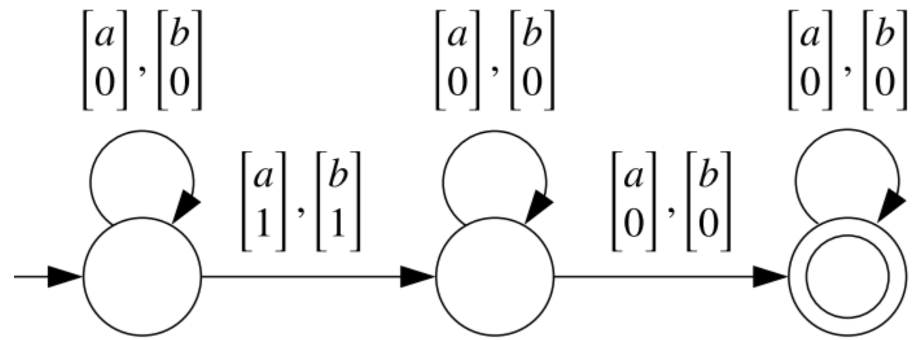
$[\neg Q_a(x)]$



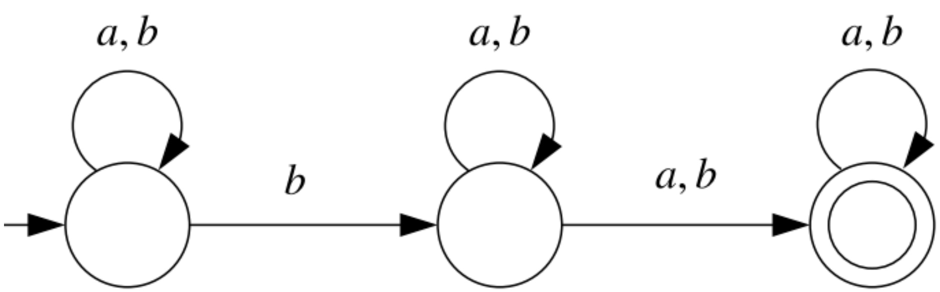
$[\neg\exists x\,(\neg\mathrm{last}(x) \wedge \neg Q_a(x))]$



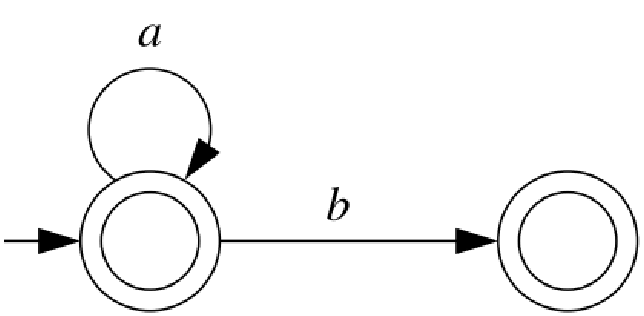
$[\neg Q_a(x)]$



$[\neg\mathrm{last}(x)]$

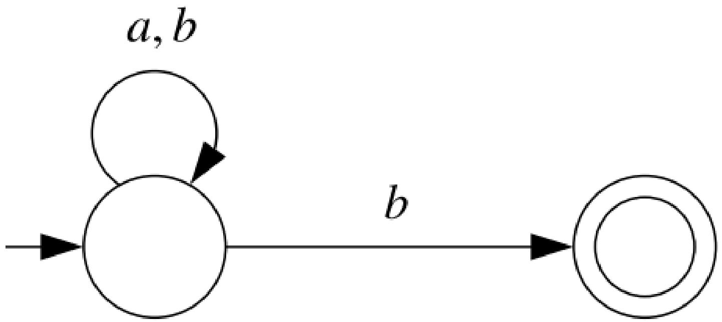


$[\exists x\,(\neg\mathrm{last}(x) \wedge \neg Q_a(x))]$

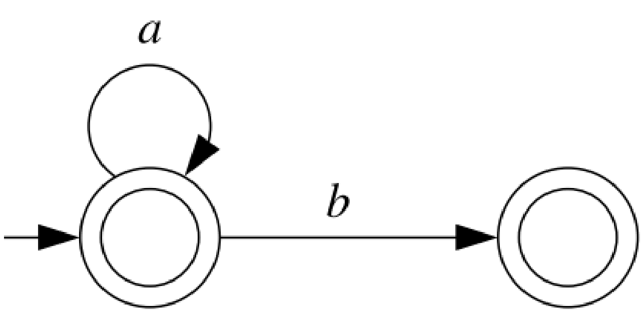


$[\neg\exists x\,(\neg\mathrm{last}(x) \wedge \neg Q_a(x))]$

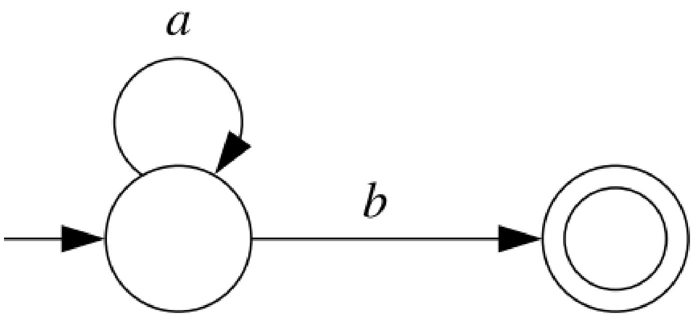
$$\left[\exists x\,\left(\mathrm{last}(x) \wedge Q_b(x)\right) \wedge \neg\exists x\,\left(\neg\mathrm{last}(x) \wedge \neg Q_a(x)\right)\right]$$



$\left[\exists x\,\left(\mathrm{last}(x) \wedge Q_b(x)\right)\right.$



$\left[\neg\exists x\,\left(\neg\mathrm{last}(x) \wedge \neg Q_a(x)\right)\right]$



$\left[\exists x\,\left(\mathrm{last}(x) \wedge Q_b(x)\right) \wedge \neg\exists x\,\left(\neg\mathrm{last}(x) \wedge \neg Q_a(x)\right)\right]$