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. 
    \((aa + bb + (ab + ba)(aa + bb)^*(ba + ab))^*\)
  – 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 $a$. 
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 $\varphi_1 \lor \varphi_2$.
  - If $\varphi$ is a formula then so is $\exists x \varphi$ for every $x \in V$. 
Abbreviations

\[ \varphi_1 \land \varphi_2 \equiv \neg (\neg \varphi_1 \lor \neg \varphi_2) \]
\[ \varphi_1 \rightarrow \varphi_2 \equiv \neg \varphi_1 \lor \varphi_2 \]
\[ \varphi_1 \leftrightarrow \varphi_2 \equiv \neg (\varphi_1 \lor \varphi_2) \lor \neg (\neg \varphi_1 \lor \neg \varphi_2) \]
\[ \forall x \varphi \equiv \neg \exists x \neg \varphi \]

first(x) := 
last(x) :=
y = x + 1 :=
y = x + 2 :=
y = x + (k + 1) :=
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.”

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

• “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.”

\[ \exists x \ Q_b(x) \land \forall x \ (\text{last}(x) \rightarrow Q_b(x) \land \neg \text{last}(x) \rightarrow Q_a(x)) \]

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

\[ \forall x \ (Q_a(x) \rightarrow \exists y \ (y = x + 1 \land Q_b(y))) \]

- “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.”

\[ \exists x \ Q_b(x) \land \forall x \ (\text{last}(x) \rightarrow Q_b(x) \land \neg \text{last}(x) \rightarrow Q_a(x)) \]

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

\[ \forall x \ (Q_a(x) \rightarrow \exists y \ (y = x + 1 \land Q_b(y))) \]

- “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 $c$.”
Examples (without semantics yet)

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

\[ \exists x \ Q_b(x) \land \forall x (\text{last}(x) \rightarrow Q_b(x) \land \neg \text{last}(x) \rightarrow Q_a(x)) \]

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

\[ \forall x (Q_a(x) \rightarrow \exists y (y = x + 1 \land Q_b(y))) \]

- “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 $c$."

\[ \forall x \forall y (Q_a(x) \land Q_b(y) \land x < y \rightarrow \exists z (x < z \land z < y \land Q_c(z))) \]
First-order logic on words: Semantics

• Formulas are interpreted on pairs \((w, J)\) called interpretations, where
  
  – \(w\) is a word, and
  
  – \(J\) assigns positions to the free variables of the formula (and maybe to others too—who cares)

• 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:

\[(w, J) \models Q_a(x) \iff w[J(x)] = a\]
\[(w, J) \models x < y \iff J(x) < J(y)\]
\[(w, J) \models \neg \varphi \iff (w, J) \not\models \varphi\]
\[(w, J) \models \varphi_1 \lor \varphi_2 \iff (w, J) \models \varphi_1 \text{ or } (w, J) \models \varphi_2\]
\[(w, J) \models \exists x \varphi \iff |w| \geq 1 \text{ and some } i \in \{1, \ldots, |w|\} \text{ satisfies } (w, J[i/x]) \models \varphi\]

• More logic jargon:
  – A formula is **valid** if it is true for all its interpretations
  – A formula is **satisfiable** if it is true for at least one of its interpretations
The empty word ...

- ... is as usual a pain in the eh, neck.
- It satisfies all universally quantified formulas, and no existentially quantified formula.
Can we only express regular languages? Can we express all regular languages?

- The language $L(\varphi)$ of a sentence $\varphi$ is the set of words that satisfy $\varphi$.
- A language $L$ is expressible in first-order logic or FO-definable if some sentence $\varphi$ 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.
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

- $x < k$     $x > k$
  $x < y + k$     $x > y + k$
  $k < \text{last}$     $k > \text{last}$

are formulas for every variable $x$, $y$ and every $k \geq 0$.

- If $f_1, f_2$ are formulas, then so are $f_1 \lor f_2$ and $f_1 \land f_2$
2) $L$ is QF-definable iff it is finite or co-finite

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

Then $f$ is an and-or combination of formulas $k < \text{last}$ and $k > \text{last}$.

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

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

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

$L(f_1 \land f_2) = L(f_1) \cap L(f_2)$ and so if $L(f)$ and $L(g)$ 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 $L = \{k_1, \ldots, k_n\}$ 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 $\neg f(L(f))$ expresses $L(f)$

- $\neg (k < \text{last}) = (k - 1 < \text{last} \land \text{last} < k + 1) \lor \text{last} < k$
- $\neg (f_1 \lor f_2) = \neg f_1 \land \neg f_2$
- $\neg (f_1 \land f_2) = \neg f_1 \lor \neg f_2$
3) Every first-order formula $\varphi$ has an equivalent QF-formula $QF(\varphi)$

- $QF(x < y) = x < y + 0$
- $QF(\neg \varphi) = neg(QF(\varphi))$
- $QF(\varphi_1 \lor \varphi_2) = QF(\varphi_1) \lor QF(\varphi_2)$
- $QF(\varphi_1 \land \varphi_2) = QF(\varphi_1) \land QF(\varphi_2)$
- $QF(\exists x \varphi) =$
  - Put $QF(\varphi)$ in disjunctive normal form. Assume $QF(\varphi) = (\varphi_1 \lor ... \lor \varphi_n)$, where each $\varphi_i$ is a conjunction of atomic formulas.
  - Since $\exists x (\varphi_1 \lor ... \lor \varphi_n) \equiv \exists x \varphi_1 \lor ... \lor \exists x \varphi_n$, it suffices to define $QF(\exists x \varphi)$ for the case in which $\varphi$ is a conjunction of atomic formulas of QF.
  - For this case, see example in the next slide.
• Consider the formula
  \[ \exists x \quad x < y + 3 \quad \land \\
  z < x + 4 \quad \land \\
  z < y + 2 \quad \land \\
  y < x + 1 \]

• The equivalent QF-formula is
  \[ z < y + 8 \quad \land \quad y < y + 5 \quad \land \quad z < y + 2 \]
Monadic second-order logic

- 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 formulas: \( x \in X \)
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 MSO

• New set \( \{X,Y,Z,...\} \) 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“

- $\text{second}(x) = \exists y \ (\text{first}(y) \land x = y + 1)$

- $\text{Even}(X) =
\forall y \ (x \in X \leftrightarrow (\text{second}(x) \lor \exists y \ (x = y + 2 \land y \in X)))$

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

\[
\exists X \ (Cons(X) \land Boc(X) \land Aod(X) \land Alt(X) \\
\land Fa(X) \land Lb(X))
\]
• $X$ 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

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

• Before $X$ there are only $c$'s

• In $X$ $a$'s and $b$'s alternate
• **X** is a block of consecutive positions

\[
\text{Cons}(X) := \forall x \in X \forall y \in X \ (x < y \rightarrow (\forall z \ (x < z \land z < y) \rightarrow z \in X))
\]

• **Before** \(X\) there are only **c**‘s

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

\[
\text{Before\_only\_c}(X) := \forall x \ \text{Before}(x, X) \rightarrow Q_c(x)
\]

• **In** \(X\) **a**‘s and **b**‘s alternate
• **$X$ is a block of consecutive positions**

\[
\text{Cons}(X) := \forall x \in X \ \forall y \in X \ (x < y \rightarrow (\forall z \ (x < z \land z < y) \rightarrow z \in X))
\]

• **Before $X$ there are only $c$‘s**

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

\[
\text{Before\_only\_c}(X) := \forall x \ \text{Before}(x, X) \rightarrow Q_c(x)
\]

• **In $X$ $a$‘s and $b$‘s alternate**

\[
\text{Alternate}(X) := \forall x \in X \ (Q_a(x) \rightarrow \forall y \in X \ (y = x + 1 \rightarrow Q_b(y)) \land Q_b(x) \rightarrow \forall y \in X \ (y = x + 1 \rightarrow Q_a(y)))
\]
Every regular language is expressible in MSO logic

- **Goal**: given an arbitrary regular language $L$, construct an MSO sentence $\varphi$ such having $L = L(\varphi)$.

- 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 is accepting
• Fix a regular language \( L \).
• Fix a DFA \( A \) with states \( q_0, \ldots, q_n \) recognizing \( L \).
• Fix a word \( w = a_1 a_2 \ldots a_m \).
• Let \( P_q \) 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:

\[
A \text{ accepts } w \text{ iff } m \in P_q \text{ for some final state } q.
\]
• Assume we can construct a formula

\[ \text{Visits}(X_0, \ldots, X_n) \]

which is true for \((w, J)\) iff

\[ J(X_0) = P_{q_0}, \ldots, J(X_n) = P_{q_n} \]

• Then \((w, J)\) satisfies the formula

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

iff \(w\) has a last letter and \(w \in L\), and we easily get a formula expressing \(L\).
• To construct \( \text{Visits}(X_0, \ldots, X_n) \) we observe that the sets \( P_q \) are the unique sets satisfying
  
  a) \( 1 \in P_{\delta(q_0,a_1)} \) i.e., after reading the first letter the DFA is in state \( \delta(q_0, a_1) \).
  
  b) The sets \( P_q \) build a partition of the set of positions, i.e., the DFA is always in exactly one state.
  
  c) If \( i \in P_q \) and \( \delta(q, a_{i+1}) = q' \) then \( i + 1 \in P_{q'} \), i.e., the sets "match" \( \delta \).

• We give formulas for a), b), and c)
\[
\text{Init}(X_0, \ldots, X_n) = \exists x \left( \text{first}(x) \land \left( \bigvee_{a \in \Sigma} (Q_a(x) \land x \in X_{i_a}) \right) \right)
\]

\[
\text{Partition}(X_0, \ldots, X_n) = \forall x \left( \bigvee_{i=0}^n x \in X_i \land \bigwedge_{i, j=0}^n (x \in X_i \implies x \notin X_j) \right)
\]
• Formula for c)

\[
\text{Respect}(X_0, \ldots, X_n) = \exists x \forall y \left( y = x + 1 \rightarrow \bigvee_{a \in \Sigma} (x \in X_i \land Q_a(x) \land y \in X_j) \right)
\]

\[
i, j \in \{0, \ldots, n\}
\]

\[
\delta(q_i, a) = q_j
\]

• Together:

\[
\text{Visits}(X_0, \ldots X_n) := \text{Init}(X_0, \ldots, X_n) \land \\
\text{Partition}(X_0, \ldots, X_n) \land \\
\text{Respect}(X_0, \ldots, X_n)
\]
Every language expressible in MSO logic is regular

- Recall: an interpretation of a formula is a pair \((w, I)\) consisting of a word \(w\) and assignments \(I\) to the free first and second order variables (and perhaps to others).
• We encode interpretations as words.

<table>
<thead>
<tr>
<th></th>
<th>a</th>
<th>a</th>
<th>b</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>1</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>y</td>
<td>0</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>X</td>
<td>0</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Y</td>
<td>1</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>b</th>
<th>a</th>
</tr>
</thead>
<tbody>
<tr>
<td>x</td>
<td>0</td>
<td>1</td>
</tr>
<tr>
<td>y</td>
<td>1</td>
<td>0</td>
</tr>
<tr>
<td>X</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>Y</td>
<td>1</td>
<td>0</td>
</tr>
</tbody>
</table>
• Given a formula with \( n \) free variables, we encode an interpretation \((w, \mathcal{I})\) as a word \( enc(w, \mathcal{I}) \) over the alphabet \( \Sigma \times \{0,1\}^n \).

• The language of the formula \( \varphi \), denoted by \( L(\varphi) \), is given by

\[
L(\varphi) = \{ enc(w, \mathcal{I}) \mid (w, \mathcal{I}) \models \varphi \}
\]

• We prove by induction on the structure of \( \varphi \) that \( L(\varphi) \) is regular (and explicitly construct an automaton for it).
Case  \( \varphi = Q_\alpha(x) \)

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

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

and is recognized by

[Diagram of a finite automaton]
Case $\varphi = x < y$

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

$$L(\varphi) = \left\{ \begin{bmatrix} a_1 \\ b_1 \\ c_1 \end{bmatrix}, \ldots, \begin{bmatrix} a_k \\ b_k \\ c_k \end{bmatrix} \right\}$$

- $k \geq 0,$
- $a_i \in \Sigma$ and $b_i, c_i \in \{0, 1\}$ for every $i \in \{1, \ldots, k\}$,
- $b_i = 1$ for exactly one index $i \in \{1, \ldots, k\}$,
- $c_j = 1$ for exactly one index $j \in \{1, \ldots, k\}$, and
- $i < j$

and is recognized by

![Diagram](image-url)
Case $\varphi = x \in X$

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

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

and is recognized by

[Diagram of a nondeterministic finite automaton with states and transitions labeled with symbols indicating the language $L(\varphi)$.]
Case $\varphi = \neg \psi$

- Then $\text{free}(\varphi) = \text{free}(\psi)$. 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 $\psi$.
- We show that the set of these encodings is regular.
  - Condition for encoding: Let $x$ be a free first-order variable of $\psi$. 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: \( \text{free}(\varphi) = \{x, y\} \)
\textbf{Case } \varphi = \varphi_1 \lor \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.
  - \( L(\varphi) = L_1 \cup L_2. \)
Example: \( \varphi = Q_a(x) \lor Q_b(y) \)

- \( L_1 \) 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 \( L_1 \):

![Automata for \( L(Q_a(x)) \) and \( L_1 \).]
Cases $\varphi = \exists x \psi$ and $\varphi = \exists X \psi$

- Then $\text{free}(\varphi) = \text{free}(\psi) \setminus \{x\}$ or $\text{free}(\varphi) = \text{free}(\psi) \setminus \{X\}$
- By i.h. $L(\psi)$ is regular.
- $L(\varphi)$ is the result of projecting $L(\psi)$ onto the components for $\text{free}(\psi) \setminus \{x\}$ or for $\text{free}(\psi) \setminus \{X\}$. 
Example: $\varphi = Q_\alpha(x)$

- Automata for $Q_\alpha(x)$ and $\exists x Q_\alpha(x)$
The mega-example

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

\[x < y\]

\[
\begin{array}{c|c}
[0|0] & [a|b] \\
[1|0] & [0|1] \\
\end{array}
\]

\[
\begin{array}{c|c}
[0|0] & [a|b] \\
[1|1] & [0|0] \\
\end{array}
\]

\[
\begin{array}{c|c}
[0|0] & [a|b] \\
[0|0] & [0|0] \\
\end{array}
\]
\[ \text{last}(x) \]

\[ [x < y] \]

\[ [\exists y \ x < y] \]
$[\text{last}(x)]$

$[x < y]$

$[\forall y \ x < y]$

$\text{Enc}(\forall y \ x < y)$

$\Sigma \times \{0, 1\}$
\[ \text{last}(x) \]

\[ x < y \]

\[ \exists y \ x < y \]

\[ \text{Enc}(\exists y \ x < y) \]

\[ \Sigma \times \{0, 1\} \]

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

$[Q_b(x)]$

$[\exists x \ (\text{last}(x) \land Q_b(x)))]$
$[-Q_a(x)]$

Enc($Q_a(x)$)

$\Sigma \times \{0, 1\}$

$[-Q_a(x)]$
$$\lnot \exists x (\lnot \text{last}(x) \land \lnot Q_a(x))$$
$[\exists x \left( \text{last}(x) \land Q_b(x) \right) \land \neg \exists x \left( \neg \text{last}(x) \land \neg Q_a(x) \right)]$