

Logic and Automata

Lecture 2

Büchi's Theorem

MSO on strings = Finite-State Automata

Sebastian Maneth

NICTA & UNSW

Logic Summer School - Canberra, December 2006

Outline

1. Finite-State Automata Recap
2. Büchi's Theorem
3. MSO to automaton: non-elementary blow up

FINITE-STATE AUTOMATON

$$\mathbf{A} = (Q, \Sigma, q_0, F, \delta)$$

Q finite set of **states**

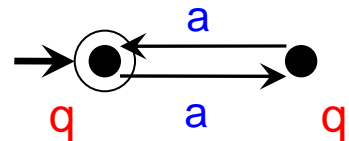
Σ input alphabet

$q_0 \in Q$ initial state

$F \subseteq Q$ set of final states

$\delta: Q \times \Sigma \rightarrow P(Q)$ transition function

Example $A = (\{q, q'\}, \{a\}, q, \{q\}, \delta)$ with $\delta(q, a) = \{q'\}$
 $\delta(q', a) = \{q\}$

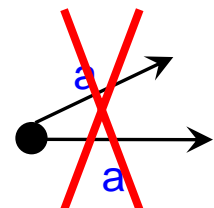


Extend δ to $Q \times \Sigma^*$

$$\underline{\delta}(q, a w) = \underline{\delta}(\delta(q, a), w)$$
$$\underline{\delta}(q, w a) = \delta(\underline{\delta}(q, w), a)$$

Language accepted by \mathbf{A} $L(\mathbf{A}) = \{ w \in \Sigma^* \mid \underline{\delta}(q_0, w) \cap F \neq \emptyset \}$

\mathbf{A} is **deterministic**, if $|\delta(q, a)| \leq 1$ for all $q \in Q$ and $a \in \Sigma$.



Basic Constructions

$$A = (Q, \Sigma, q_0, F, \delta)$$

→ **Deterministic Automaton** (subset/powerset construction)

$$\text{Det}(A) = (P(Q), \Sigma, \{ q_0 \}, \{ S \in P(Q) \mid S \cap F \neq \emptyset \}, \delta')$$

for $S \in P(Q)$ and $a \in \Sigma$: $\delta'(S, a) = \{ \delta(p, a) \mid p \in S \}$

→ **Complement Automaton** accepts $\Sigma^* - L(A)$

Given deterministic A, simply exchange final w non-final states!

→ **Intersection Automaton** (product construction)
accepts $L(A) \cap L(B)$

$$AB = (Q \times P, \Sigma, \langle q_0, p_0 \rangle, F \times G, d)$$

$$d(\langle q, p \rangle, a) = \{ \langle q', p' \rangle \mid q' \in \delta(q, a), p' \in \gamma(p, a) \}$$

→ **Minimal deterministic Automaton**
(remove useless states & merge equivalent states)

1. Büchi's Theorem

MSO definable string languages = Regular languages

(1) Automaton to Logic

For every finite-state automaton A there (effectively) exists an MSO sentence $\psi(A)$ such that $L(\psi(A)) = L(A)$.

(2) Logic to Automaton

For every MSO sentence ϕ there (effectively) exists a finite-state automaton $A(\phi)$ such that $L(A(\phi)) = L(\phi)$.

Automaton to Logic

Idea use a set variable X_q for every state q of the automaton A .
 X_q contains positions of input string, for which A is in state q

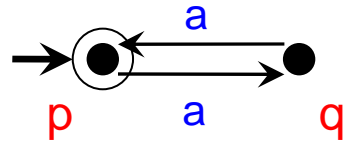
$\text{first}(x) \equiv \neg(\exists x) \text{ edge}(x,y)$

$\text{last}(x) \equiv \neg(\exists y) \text{ edge}(x,y)$

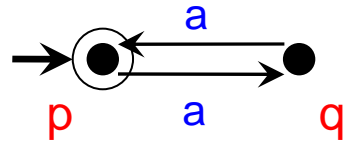
Given a **deterministic automaton** $A = (Q, \Sigma, q_0, F, \delta)$

$$\begin{aligned} \psi(A) = & (\exists X_q)_{q \in Q} && \text{total run} \\ & [(\forall i)(\bigvee_{q \in Q} i \in X_q) \\ & \wedge (\forall i) \neg(\bigvee_{p, q \in Q, p \neq q} (i \in X_p \wedge i \in X_q)) && \text{partition} \\ & \wedge (\forall i)(\text{first}(i) \rightarrow \bigvee_{(q_0, a, q) \in \delta} (\text{lab-}a(i) \wedge i \in X_q)) && \text{first step} \\ & \wedge (\forall i)(\forall j)(\text{edge}(i, j) \rightarrow \bigvee_{(p, a, q) \in \delta} (i \in X_p \wedge \text{lab-}a(j) \wedge j \in X_q)) && \text{transition} \\ & \wedge (\forall i)(\text{last}(i) \rightarrow \bigvee_{p \in F} (i \in X_p)) && \text{final state} \\ &] \end{aligned}$$

$$\begin{aligned}
& (\exists X_q)_{q \in Q} \\
& [(\forall i) (\bigvee_{q \in Q} i \in X_q) \\
& \wedge (\forall i) \neg (\bigvee_{p, q \in Q, p \neq q} (i \in X_p \wedge i \in X_q)) \\
& \wedge (\forall i) (\text{first}(i) \rightarrow \bigvee_{(q_0, a, q) \in \delta} (\text{lab-}a(i) \wedge i \in X_q)) \\
& \wedge (\forall i) (\forall j) (\text{edge}(i, j) \rightarrow \bigvee_{(p, a, q) \in \delta} (i \in X_p \wedge \text{lab-}a(j) \wedge j \in X_q)) \\
& \wedge (\forall i) (\text{last}(i) \rightarrow \bigvee_{p \in F} (i \in X_p))]
\end{aligned}$$

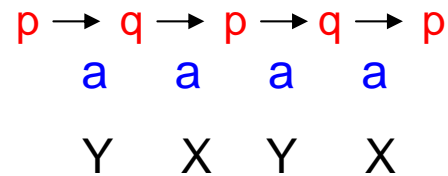
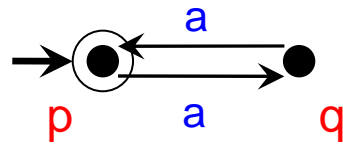


$$\begin{aligned}
& (\exists X_q)_{q \in Q} [(\forall i) (\forall_{q \in Q} i \in X_q) \\
& \quad \wedge (\forall i) \neg (\forall_{p, q \in Q, p \neq q} (i \in X_p \wedge i \in X_q)) \\
& \quad \wedge (\forall i) (\text{first}(i) \rightarrow \forall_{(q_0, a, q) \in \delta} (\text{lab-}a(i) \wedge i \in X_q)) \\
& \quad \wedge (\forall i) (\forall j) (\text{edge}(i, j) \rightarrow \forall_{(p, a, q) \in \delta} (i \in X_p \wedge \text{lab-}a(j) \wedge j \in X_q)) \\
& \quad \wedge (\forall i) (\text{last}(i) \rightarrow \forall_{p \in F} (i \in X_p))]
\end{aligned}$$



$$\begin{aligned}
& (\exists X)(\exists Y) [(\forall i) (i \in X \vee i \in Y) \\
& \quad \wedge (\forall i) \neg (i \in X \wedge i \in Y) \\
& \quad \wedge (\forall i) (\text{first}(i) \rightarrow (\text{lab-}a(i) \wedge i \in Y)) \\
& \quad \wedge (\forall i) (\text{edge}(i, j) \rightarrow ((i \in X \wedge \text{lab-}a(j) \wedge j \in Y) \vee (i \in Y \wedge \text{lab-}a(j) \wedge j \in X))) \\
& \quad \wedge (\forall i) (\text{last}(i) \rightarrow i \in X)]
\end{aligned}$$

$$\begin{aligned}
& (\exists X_q)_{q \in Q} [(\forall i) (\forall_{q \in Q} i \in X_q) \\
& \quad \wedge (\forall i) \neg (\forall_{p, q \in Q, p \neq q} (i \in X_p \wedge i \in X_q)) \\
& \quad \wedge (\forall i) (\text{first}(i) \rightarrow \forall_{(q_0, a, q) \in \delta} (\text{lab-}a(i) \wedge i \in X_q)) \\
& \quad \wedge (\forall i) (\forall j) (\text{edge}(i, j) \rightarrow \forall_{(p, a, q) \in \delta} (i \in X_p \wedge \text{lab-}a(j) \wedge j \in X_q)) \\
& \quad \wedge (\forall i) (\text{last}(i) \rightarrow \forall_{p \in F} (i \in X_p))]
\end{aligned}$$



$$\begin{aligned}
& (\exists X)(\exists Y) [(\forall i) (i \in X \vee i \in Y) \\
& \quad \wedge (\forall i) \neg (i \in X \wedge i \in Y) \\
& \quad \wedge (\forall i) (\text{first}(i) \rightarrow (\text{lab-}a(i) \wedge i \in Y)) \\
& \quad \wedge (\forall i) (\text{edge}(i, j) \rightarrow ((i \in X \wedge \text{lab-}a(j) \wedge j \in Y) \vee (i \in Y \wedge \text{lab-}a(j) \wedge j \in X))) \\
& \quad \wedge (\forall i) (\text{last}(i) \rightarrow i \in X)]
\end{aligned}$$

$$\begin{aligned} \psi(A) = & (\exists X)(\exists Y)[(\forall i) (i \in X \vee i \in Y) \\ & \wedge (\forall i) \neg(i \in X \wedge i \in Y) \\ & \wedge (\forall i)(\text{first}(i) \rightarrow (\text{lab-}a(i) \wedge i \in Y) \\ & \wedge (\forall i)(\text{edge}(i,j) \rightarrow ((i \in X \wedge \text{lab-}a(j) \wedge j \in Y) \vee (i \in Y \wedge \text{lab-}a(j) \wedge j \in X)) \\ & \wedge (\forall i)(\text{last}(i) \rightarrow i \in X) \end{aligned}$$

Of course, we can simplify $\psi(A)$.

→ if a is only label, remove $\text{lab-}a$'s

→ First two lines imply $(i \in X) \leftrightarrow \neg(i \in Y)$

Hence, we can remove first two lines

$$\begin{aligned} \psi(A) = & (\exists X)[\begin{array}{cccc} & a & a & a & a \\ \neg X & X & \neg X & X & \end{array} \\ & \wedge (\forall i)(\text{first}(i) \rightarrow \neg(i \in X) \\ & \wedge (\forall i)(\text{edge}(i,j) \rightarrow ((i \in X \wedge \neg(j \in X)) \vee (\neg(i \in X) \wedge j \in X)) \\ & \wedge (\forall i)(\text{last}(i) \rightarrow i \in X) \end{aligned}$$

$$A = (\{p,q\}, \{a\}, p, \{p\}, \{ (p,a,q), (q,a,p) \})$$

Automaton to Logic

Idea use a set variable X_q for every state q of the automaton A .
 X_q contains positions of input string, for which A is in state q

$$\begin{aligned} \psi(A) = & (\exists X_q)_{q \in Q} \\ & [(\forall i)(\bigvee_{q \in Q} i \in X_q) \\ & \wedge (\forall i) \neg (\bigvee_{p, q \in Q, p \neq q} (i \in X_p \wedge i \in X_q)) \\ & \wedge (\forall i)(\text{first}(i) \rightarrow \bigvee_{(q_0, a, q) \in \delta} (\text{lab-}a(i) \wedge i \in X_q)) \\ & \wedge (\forall i)(\forall j)(\text{edge}(i, j) \rightarrow \bigvee_{(p, a, q) \in \delta} (i \in X_p \wedge \text{lab-}a(j) \wedge j \in X_q)) \\ & \wedge (\forall i)(\text{last}(i) \rightarrow \bigvee_{p \in F} (i \in X_p)) \\ &] \end{aligned}$$

What is the **size** of $\psi(A)$, in terms of **size** of A ?

Automaton to Logic

Idea use a set variable X_q for every state q of the automaton A .
 X_q contains positions of input string, for which A is in state q

$$\begin{aligned} \psi(A) = & (\exists X_q)_{q \in Q} \\ & [(\forall i)(\forall_{q \in Q} i \in X_q) \\ & \wedge (\forall i) \neg (\exists_{p, q \in Q, p \neq q} (i \in X_p \wedge i \in X_q)) \\ & \wedge (\forall i)(\text{first}(i) \rightarrow \exists_{(q_0, a, q) \in \delta} (\text{lab-}a(i) \wedge i \in X_q)) \\ & \wedge (\forall i)(\forall j)(\text{edge}(i, j) \rightarrow \exists_{(p, a, q) \in \delta} (i \in X_p \wedge \text{lab-}a(j) \wedge j \in X_q)) \\ & \wedge (\forall i)(\text{last}(i) \rightarrow \exists_{p \in F} (i \in X_p)) \\ &] \end{aligned}$$

What is the **size** of $\psi(A)$, in terms of **size** of A ?

$$\text{size}(\psi(A)) = O(\text{size}(A))$$

→ grows only by a **constant factor**, i.e.,
 $\exists c: \text{size}(\psi(A)) \leq c \cdot \text{size}(A)$

Logic to Automaton

Idea: inductive definition, show it for subformulas

But, subformulas contain variables.

How can we determine
the truth value of, e.g., $(\forall i)(\text{lab-a}(i) \rightarrow i \in X)$,
if we do not know the value of X ?

For an arbitrary formula ϕ , an element in the language $L(\phi)$ shall not only describe a Σ -string, but also a **valuation** β of the FREE variables in ϕ .

Example $\phi = (\forall i)(\text{lab-a}(i) \rightarrow i \in X)$

$L(\phi)$ contains the string $w =$

a	b	b	a	b
1	0	1	1	0

← “am I in X?”

This word determines $\beta_w(X) = \{1, 3, 4\}$ and $[w] = a b b a b$

Logic to Automaton

Example $\phi = i \in X \wedge i \in Y$

$L(\phi)$ contains $w =$ a a b b b

0	0	1	0	0	←	“am I the i?”	☺
1	1	1	1	0	←	in X?	
1	0	1	1	0	←	in Y?	

for which $\beta_w(i) = 3,$
 $\beta_w(X) = \{ 1, 2, 3, 4 \}$
 $\beta_w(Y) = \{ 1, 3, 4 \}$

But NOT

a	a	b	b	b	and	a	a	b	b	b
0	0	1	1	0		0	0	1	0	0
1	1	1	1	0		1	1	0	1	0
1	0	1	1	0		1	0	1	1	0

$L(\exists i \phi)$ contains

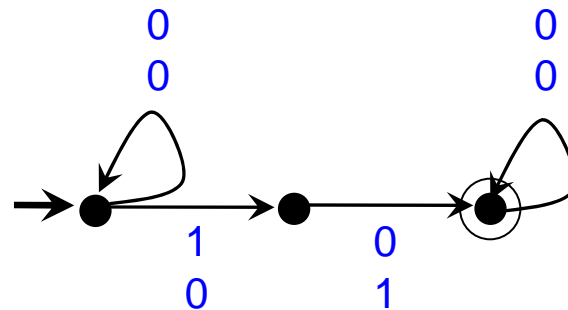
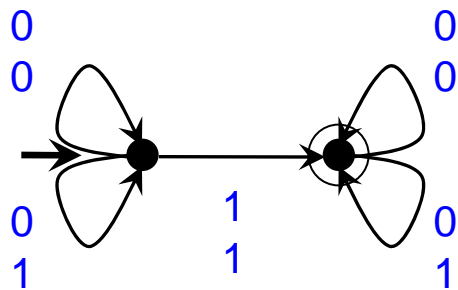
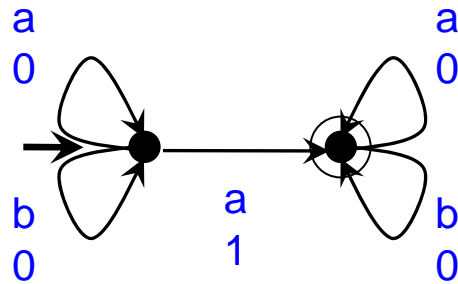
a	a	b	b	b
1	1	1	1	0
1	0	1	1	0

Logic to Automaton

Lemma 1 Let $\Sigma = \{a, b\}$.

the languages $L(\text{lab-a}(i))$, $L(i \in X)$, and $L(\text{edge}(i,j))$

Are accepted by the following automata



Σ -label irrelevant for the last two...

Logic to Automaton

Let V be a set of variables, and $U \subseteq V$

The homomorphism $v_U: \Sigma_V^* \rightarrow \Sigma_U$ deletes the $\{0,1\}$ -components for $V - U$

Thus, $v_U((a, f)) = (a, f|_U)$, where $f|_U$ is the restriction of f to U .

Lemma 2 Let ϕ, ϕ_1, ϕ_2 be $\text{MSO}(\Sigma)$ formulas and V, V_1, V_2 the sets of variables that occur free in them.

$$(1) \quad \mathbf{L}(\neg \phi) = B_V(\Sigma) - \mathbf{L}(\phi)$$

$$(2) \quad \mathbf{L}(\exists i) \phi = v_{V - \{i\}}(\mathbf{L}(\phi))$$

$$(3) \quad \mathbf{L}(\exists X) \phi = v_{V - \{X\}}(\mathbf{L}(\phi))$$

$$(4) \quad \mathbf{L}(\phi_1 \wedge \phi_2) = v_{V_1}^{-1}(\mathbf{L}(\phi_1)) \cap v_{V_2}^{-1}(\mathbf{L}(\phi_2))$$

Logic to Automaton

Lemma 2 Let ϕ, ϕ_1, ϕ_2 be MSO(Σ) formulas and V, V_1, V_2 the sets of variables that occur free in them.

$$(1) \quad \mathbf{L}(\neg \phi) = \mathbf{B}_V(\Sigma) - \mathbf{L}(\phi).$$

$$(2) \quad \mathbf{L}(\exists i \phi) = v_{V - \{i\}}(\mathbf{L}(\phi))$$

$$(3) \quad \mathbf{L}(\exists X \phi) = v_{V - \{X\}}(\mathbf{L}(\phi))$$

$$(4) \quad \mathbf{L}(\phi_1 \wedge \phi_2) = v_{V_1}^{-1}(\mathbf{L}(\phi_1)) \cap v_{V_2}^{-1}(\mathbf{L}(\phi_2))$$

→ Given automata A, A_1, A_2 for $\mathbf{L}(\phi), \mathbf{L}(\phi_1), \mathbf{L}(\phi_2)$, we can construct

new automata for

$\mathbf{L}(\neg \phi)$	complement construction
$\mathbf{L}(\exists i \phi)$	
$\mathbf{L}(\exists X \phi)$	label change
$\mathbf{L}(\phi_1 \wedge \phi_2)$	product construction

Logic to Automaton

Theorem Let ϕ be an MSO(Σ) formula. There effectively exists a finite-state automaton $A(\phi)$ such that $L(A(\phi)) = L(\phi)$.

Proof Induction on the structure of ϕ .
Lemma 1 gives base case, and
Lemma 2 & automata constructions give inductive step.
Büchi, we are Finished. 😊

→ Given automata A, A_1, A_2 for $L(\phi), L(\phi_1), L(\phi_2)$, we can construct

new automata for

$L(\neg\phi)$	complement construction
$L(\exists i \phi)$	
$L(\exists X \phi)$	label change
$L(\phi_1 \wedge \phi_2)$	product construction

Logic to Automaton

Lemma 2 Let ϕ, ϕ_1, ϕ_2 be $\text{MSO}(\Sigma)$ formulas and V, V_1, V_2 the sets of variables that occur free in them.

$$(1) \mathbf{L}(\neg \phi) = \mathbf{B}_V(\Sigma) - \mathbf{L}(\phi).$$

$$(2) \mathbf{L}(\exists i \phi) = v_{V - \{i\}}(\mathbf{L}(\phi))$$

$$(3) \mathbf{L}(\exists X \phi) = v_{V - \{X\}}(\mathbf{L}(\phi))$$

$$(4) \mathbf{L}(\phi_1 \wedge \phi_2) = v_{V_1}^{-1}(\mathbf{L}(\phi_1)) \cap v_{V_2}^{-1}(\mathbf{L}(\phi_2))$$

→ Given automata A, A_1, A_2 for $\mathbf{L}(\phi), \mathbf{L}(\phi_1), \mathbf{L}(\phi_2)$, we can construct

new automata for

$\mathbf{L}(\neg \phi)$	complement construction
$\mathbf{L}(\exists i \phi)$	
$\mathbf{L}(\exists X \phi)$	label change
$\mathbf{L}(\phi_1 \wedge \phi_2)$	product construction

automaton
A must be
deterministic!

Logic to Automaton

Lemma 2 Let ϕ, ϕ_1, ϕ_2 be MSO(Σ) formulas and V, V_1, V_2 the sets of variables that occur free in them.

$$(1) \mathbf{L}(\neg \phi) = \mathbf{B}_V(\Sigma) - \mathbf{L}(\phi).$$

$$(2) \mathbf{L}(\exists i \phi) = v_{V - \{i\}}(\mathbf{L}(\phi))$$

$$(3) \mathbf{L}(\exists X \phi) = v_{V - \{X\}}(\mathbf{L}(\phi))$$

$$(4) \mathbf{L}(\phi_1 \wedge \phi_2) = v_{V_1}^{-1}(\mathbf{L}(\phi_1)) \cap v_{V_2}^{-1}(\mathbf{L}(\phi_2))$$

→ Given automata A, A_1, A_2 for $\mathbf{L}(\phi), \mathbf{L}(\phi_1), \mathbf{L}(\phi_2)$, we can construct

new automata for

$\mathbf{L}(\neg \phi)$	complement construction
$\mathbf{L}(\exists i \phi)$	
$\mathbf{L}(\exists X \phi)$	label change
$\mathbf{L}(\phi_1 \wedge \phi_2)$	product construction

automaton
A must be
deterministic!

→ Where do we break determinism of automaton??

Logic to Automaton

Lemma 2 Let ϕ, ϕ_1, ϕ_2 be $\text{MSO}(\Sigma)$ formulas and V, V_1, V_2 the sets of variables that occur free in them.

$$(1) \mathbf{L}(\neg \phi) = \mathbf{B}_V(\Sigma) - \mathbf{L}(\phi).$$

$$(2) \mathbf{L}(\exists i \phi) = v_{V - \{i\}}(\mathbf{L}(\phi))$$

$$(3) \mathbf{L}(\exists X \phi) = v_{V - \{X\}}(\mathbf{L}(\phi))$$

$$(4) \mathbf{L}(\phi_1 \wedge \phi_2) = v_{V_1}^{-1}(\mathbf{L}(\phi_1)) \cap v_{V_2}^{-1}(\mathbf{L}(\phi_2))$$

→ Given automata A, A_1, A_2 for $\mathbf{L}(\phi), \mathbf{L}(\phi_1), \mathbf{L}(\phi_2)$, we can construct

new automata for $\mathbf{L}(\neg\phi)$ complement construction

$\mathbf{L}(\exists i \phi)$

$\mathbf{L}(\exists X \phi)$ label change

$\mathbf{L}(\phi_1 \wedge \phi_2)$ product construction

automaton
A must be
deterministic!

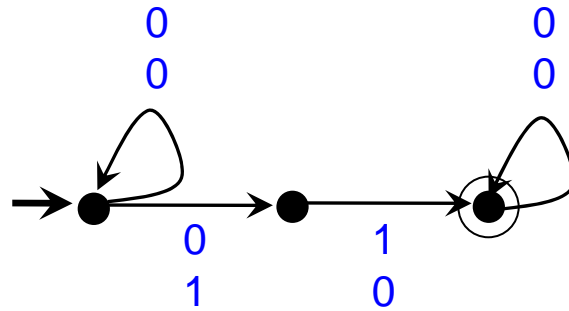
→ Where do we **break determinism** of automaton??

Logic to Automaton

Example. $\phi = (\exists i)(\text{first}(i) \wedge \text{lab-a}(i))$
 $= (\exists i) (\neg(\exists y) \text{edge}(y,x) \wedge \text{lab-a}(i))$

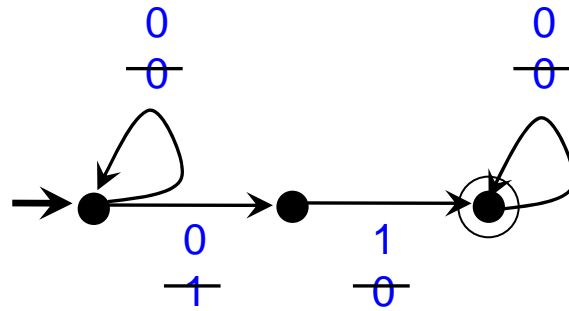
Logic to Automaton

Example. $\phi = (\exists i)(\text{first}(i) \wedge \text{lab-a}(i))$
 $= (\exists i) (\neg(\exists y) \text{edge}(y,x) \wedge \text{lab-a}(i))$



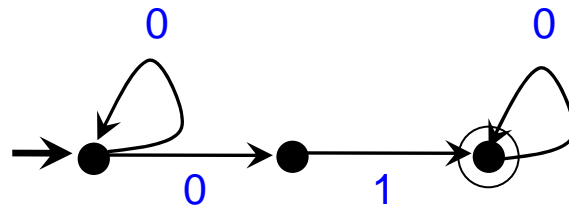
Logic to Automaton

Example. $\phi = (\exists i)(\text{first}(i) \wedge \text{lab-a}(i))$
 $= (\exists i) (\text{---} \text{edge}(y,x) \wedge \text{lab-a}(i))$



Logic to Automaton

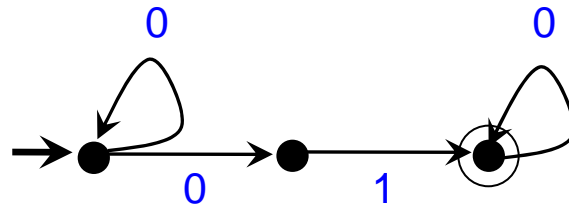
Example. $\phi = (\exists i)(\text{first}(i) \wedge \text{lab-a}(i))$
 $= (\exists i) (\neg(\exists y) \text{edge}(y,x) \wedge \text{lab-a}(i))$



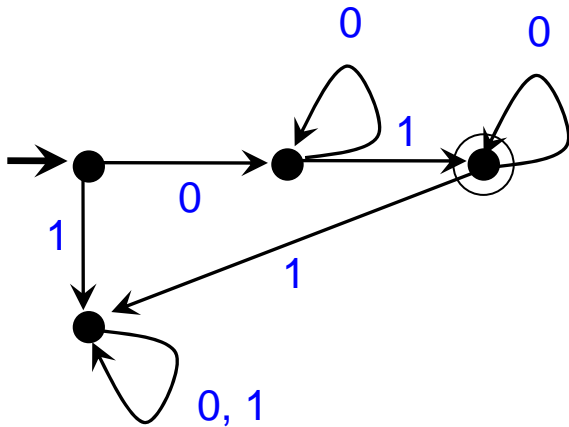
NONdeterministic

Logic to Automaton

Example. $\phi = (\exists i)(\text{first}(i) \wedge \text{lab-a}(i))$
 $= (\exists i) (\neg(\exists y) \text{edge}(y,x) \wedge \text{lab-a}(i))$

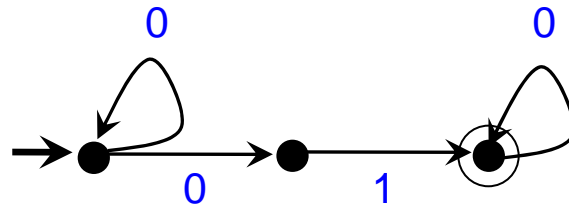


(1) Make deterministic & complete



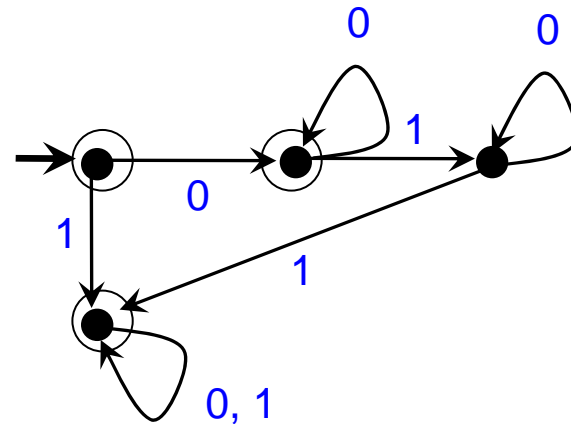
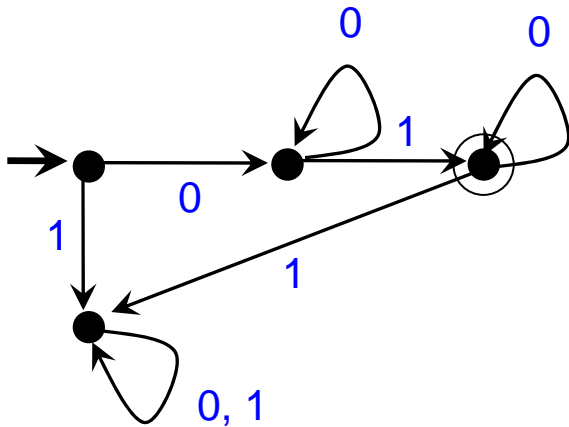
Logic to Automaton

Example. $\phi = (\exists i)(\text{first}(i) \wedge \text{lab-a}(i))$
 $= (\exists i) (\neg(\exists y) \text{edge}(y,x) \wedge \text{lab-a}(i))$



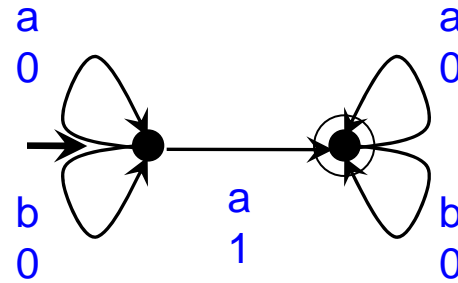
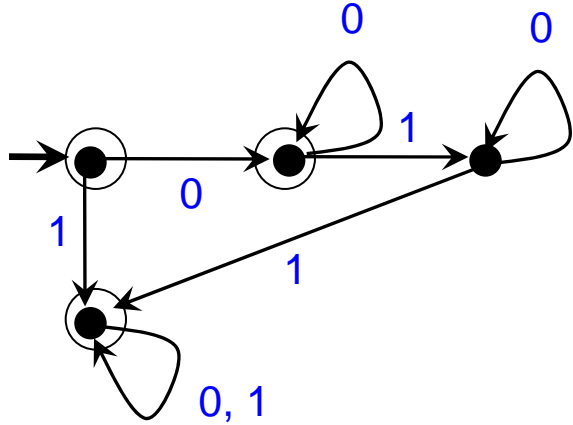
(1) Make deterministic & complete

(2) Complement

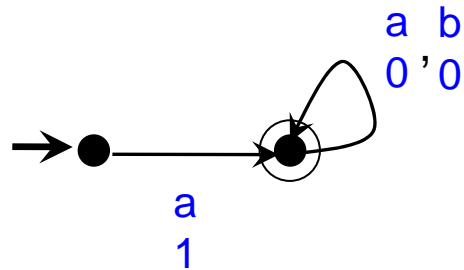


Logic to Automaton

Example. $\phi = (\exists i)(\text{first}(i) \wedge \text{lab-a}(i))$
 $= (\exists i) (\neg(\exists y) \text{edge}(y,x) \wedge \text{lab-a}(i))$

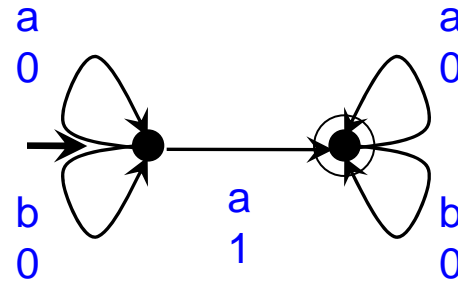
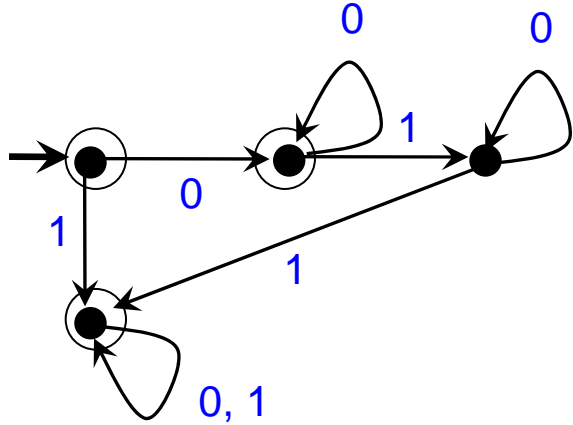


Product
Construction gives

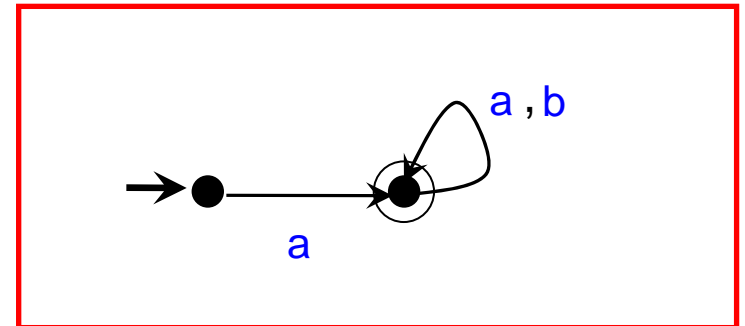
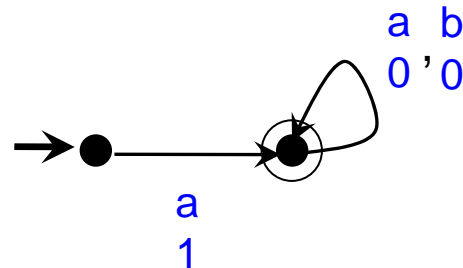


Logic to Automaton

Example. $\phi = (\exists i)(\text{first}(i) \wedge \text{lab-a}(i))$
 $= (\exists i) (\neg(\exists y) \text{edge}(y,x) \wedge \text{lab-a}(i))$



“Product Construction” gives



Automaton for $L(\phi)$

Büchi's Theorem

MSO definable string languages = Regular languages

(1) Automaton to Logic

For every finite-state automaton A there (effectively) exists an MSO sentence $\psi(A)$ such that $L(\psi(A)) = L(A)$.

Question How large is $\psi(A)$ at most wrt size of A ?

(2) Logic to Automaton

For every MSO sentence ϕ there (effectively) exists a finite-state automaton $A(\phi)$ such that $L(A(\phi)) = L(\phi)$.

Question How large is $A(\phi)$ at most wrt size of ϕ ?

Büchi's Theorem

MSO definable string languages = Regular languages

(1) Automaton to Logic

For every finite-state automaton A there (effectively) exists an MSO sentence $\psi(A)$ such that $L(\psi(A)) = L(A)$.

$\text{size}(\psi(A)) = O(\text{size}(A))$ (linearly related)

(2) Logic to Automaton

For every MSO sentence ϕ there (effectively) exists a finite-state automaton $A(\phi)$ such that $L(A(\phi)) = L(\phi)$.

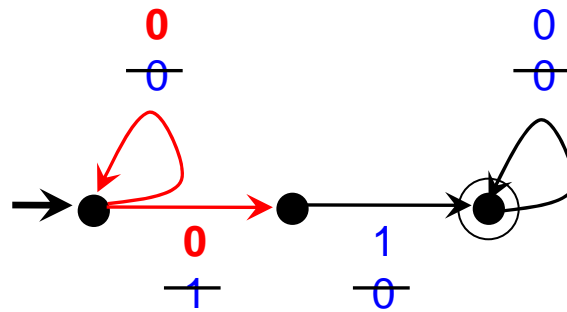
$\exists \phi$ such that

$\text{size}(\text{minimal } A(\phi))$ is NON-ELEMENTARY larger than ϕ .

→ “MSO is non-elementary **more succinct** than finite-state automata!”

Logic to Automaton

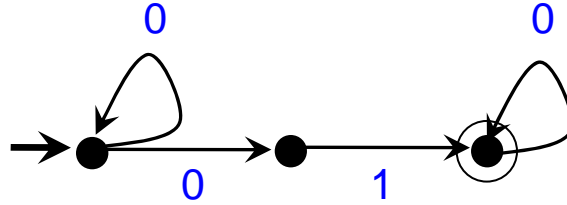
Example. $\phi = (\exists i)(\text{first}(i) \wedge \text{lab}(a,i))$
 $= (\exists i) (\neg(\exists y) \text{edge}(y,x) \wedge \text{lab}(a,i))$



existential quantifier
causes
nondeterminism

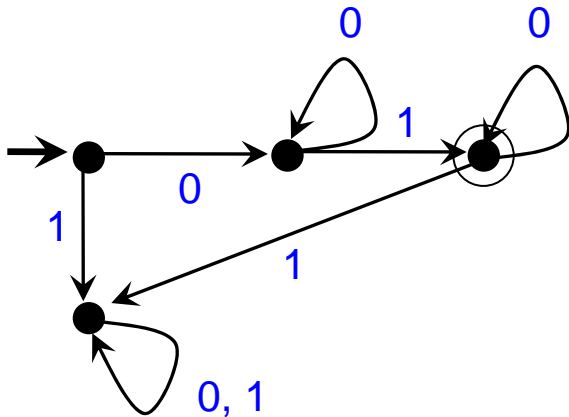
Logic to Automaton

Example. $\phi = (\exists i)(\text{first}(i) \wedge \text{lab}(a,i))$
 $= (\exists i) (\neg(\exists y) \text{edge}(y,x) \wedge \text{lab}(a,i))$

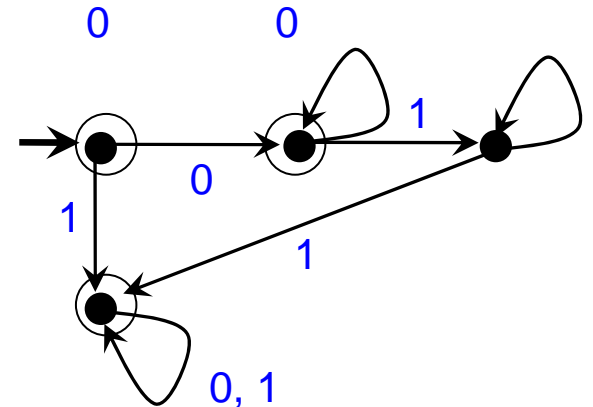


existential quantifier
causes
nondeterminism

Determinize  can cause
EXPONENTIAL blow-up



... to do **negation** ...



Logic to Automaton

Thus, nested existential quantifiers, alternating with negation, can cause *iterated EXPONENTIAL blow-up of size of automaton!!*

Logic to Automaton

Thus, nested existential quantifiers, alternating with negation, can cause *iterated EXPONENTIAL blow-up of size of automaton!!*

Concrete example:

$$F(1) = 1$$

$$F(n+1) = F(n) 2^{F(n)}$$

$$L_n = 0 \dots 0 \ 1 \ 0^{F(n)-1} \ 1 \ 0 \dots 0$$

Logic to Automaton

Thus, nested existential quantifiers, alternating with negation, can cause *iterated EXPONENTIAL blow-up of size of automaton!!*

Concrete example:

$$F(1) = 1$$

$$F(n+1) = F(n) \cdot 2^{F(n)}$$

$$F(2) = 2^1 = 2$$

$$F(3) = 2 \cdot 2^2 = 8$$

$$F(4) = 8 \cdot 2^8 = 2048$$

$$L_n = 0 \dots 0 \ 1 \ 0^{F(n)-1} \ 1 \ 0 \dots 0$$

$$L_1 = 0..0 \ 11 \ 0..0$$

$$L_2 = 0..0 \ 101 \ 0..0$$

$$L_3 = 0..0 \ 100000001 \ 0..0$$

$$L_4 = 0..0 \ 1000 \dots 0001 \ 0..0$$



2047

Logic to Automaton

Thus, nested existential quantifiers, alternating with negation, can cause *iterated EXPONENTIAL blow-up of size of automaton!!*

Concrete example:

$$F(1) = 1$$

$$F(n+1) = F(n) \cdot 2^{F(n)}$$

$$F(2) = 2^1 = 2$$

$$F(3) = 2 \cdot 2^2 = 8$$

$$F(4) = 8 \cdot 2^8 = 2048$$

$$L_n = 0 \dots 0 \ 1 \ 0^{F(n)-1} \ 1 \ 0 \dots 0$$

$$L_1 = 0..0 \ 11 \ 0..0$$

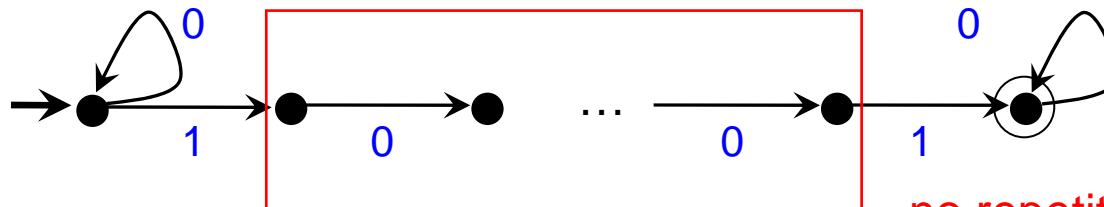
$$L_2 = 0..0 \ 101 \ 0..0$$

$$L_3 = 0..0 \ 100000001 \ 0..0$$

$$L_4 = 0..0 \ 1000 \dots 0001 \ 0..0$$

2047

Clearly, automaton for $L(n)$ needs $\geq F(n)$ states!!



cannot count..

no repetition of states

Logic to Automaton

Thus, nested existential quantifiers, alternating with negation, can cause *iterated EXPONENTIAL blow-up of size of automaton!!*

Concrete example:

$$F(1) = 1$$

$$F(n+1) = F(n) \cdot 2^{F(n)}$$

$$F(2) = 2^1 = 2$$

$$F(3) = 2 \cdot 2^2 = 8$$

$$F(4) = 8 \cdot 2^8 = 2048$$

$$L_n = 0 \dots 0 \ 1 \ 0^{F(n)-1} \ 1 \ 0 \dots 0$$

$$L_1 = 0..0 \ 11 \ 0..0$$

$$L_2 = 0..0 \ 101 \ 0..0$$

$$L_3 = 0..0 \ 100000001 \ 0..0$$

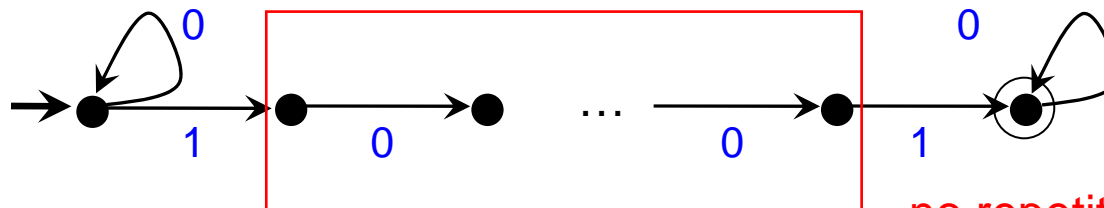
$$L_4 = 0..0 \ 1000 \dots 0001 \ 0..0$$

2047

→ Find formula ϕ_n^A that defines L_n

But with $\text{size}(\phi_n^A)$ only linear in $n!!$

Clearly, automaton for $L(n)$ needs $\geq F(n)$ states!!



cannot count..

no repetition of states

$$\begin{aligned}
L_1 &= 0..0 11 0..0 \\
L_2 &= 0..0 101 0..0 \\
L_3 &= 0..0 100000001 0..0 \\
L_4 &= 0..0 \underbrace{1000\dots0001}_{2047} 0..0
\end{aligned}$$

→ Find formula ϕ_n^A that defines L_n
but $\text{size}(\phi_n^A)$ only linear in $n!!$

(A is a predicate that is true for
a position x iff x is labeled 1)

Inductive definition

$$\phi_1^A = \exists x(A(x) \wedge A(x+1) \wedge \forall y(y=x \vee y=x+1 \vee \neg A(y)))$$

“exactly two positions, x and $x+1$ are labeled 1”

$$\begin{aligned}
L_1 &= 0..0 \ 11 \ 0..0 \\
L_2 &= 0..0 \ 101 \ 0..0 \\
L_3 &= 0..0 \ 100000001 \ 0..0 \\
L_4 &= 0..0 \ 1000\dots0001 \ 0..0
\end{aligned}$$



2047

→ Find formula ϕ_n^A that defines L_n
but **size(ϕ_n^A) only linear in n!!**

(A is a predicate that is true for a position x iff x is labeled 1)

Inductive definition

$$\phi_1^A = \exists x(A(x) \wedge A(x+1) \wedge \forall y(y=x \vee y=x+1 \vee \neg A(y)))$$

“exactly two positions, x and $x+1$ are labeled 1”

ϕ_{n+1}^A : use ϕ_n^A to determine if two positions have **distance $F(n)$**

Distance $F(n)$ is now the length of a counter, that counts from

0 ... 0 to 1 ... 1

E.g. For L_4 we use counters of length $F(3) = 8$.

00000000, 10000000, 01000000, 11000000, ... , 11111111

256 blocks
each 8 letters
= **2048**

A: 0 . . 0 10000000 00000000 00000000 ... 00000000 00000000 10...
 C: 00000000 10000000 01000000 ... 01111111 11111111 00...
 B: 10000000 10000000 10000000 10000000 10000000 10000000 10...

A: 0 . . 0 10000000 10...
 C: 00100111 00...
 B: 10101010 10...

A: 0 . . 0 10000000 10...
 C: 00100111 00...
 B: 10101010 10...

A: 0 . . 0 1010...
 C: 0100...
 B: 1110...

$$\phi_{n+1}^A = \exists x \exists y (A(x) \wedge A(y) \wedge \forall z (z=x \vee z=y \vee \neg A(z)))$$

$$\exists B \exists C \forall z \forall b ((\exists A (\phi_n^A \wedge a < b \leq y \wedge A(a) \wedge A(b)) \rightarrow (\text{InitC} \wedge \text{StartINC} \wedge \text{Carry} \wedge \text{FinalC})))$$

$$\text{InitC} = (a=x) \rightarrow (B(a) \wedge \neg C(a) \wedge \forall c (a < c < b \rightarrow (\neg C(c) \wedge \neg B(c))))$$

$$\text{StartINC} = (B(a) \vee B(b)) \rightarrow (B(b) \wedge \neg (C(a) \leftrightarrow C(b)))$$

$$\text{Carry} = ((C(a) \wedge \neg C(b)) \leftrightarrow (\neg C(a+1) \leftrightarrow C(b+1))) \vee B(a+1)$$

$$\text{FinalC} = (b=y) \leftrightarrow (B(a) \wedge \forall c (a \leq c < b \rightarrow C(c)))$$

Recap

MSO = first-order logic (FO) + quantification over **sets**

FO theory of finite graphs: $\{ \phi \in \text{FO} \mid g \models \phi \text{ for every finite graph } g \}$
is undecidable.

MSO: If **C** contains infinitely many square grids,
then MSO theory of **C** is undecidable.

→ **Restrict class C**, so that MSO theory becomes decidable!

(1) **C = STRINGS**

Theorem (Büchi) *Set of strings is **MSO definable** if and only if it is accepted by a **finite-state automaton**.*

Corollary ***MSO theory** of **strings** is decidable.*