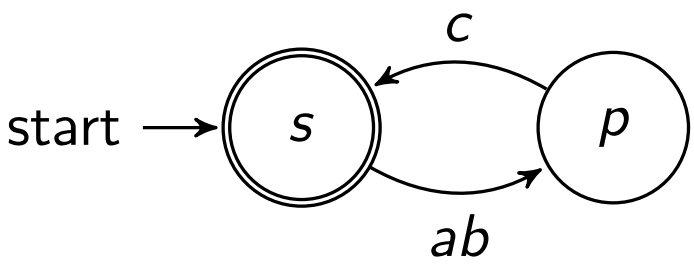
Membership Problem for Two-Dimensional Jumping Finite Automata

Grzegorz Madejski, Andrzej Szepietowski

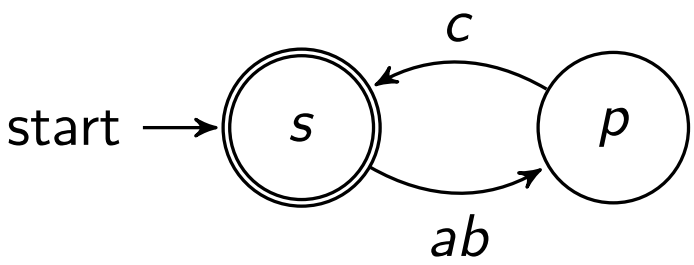
Institute of Informatics,
Faculty of Mathematics, Physics and Informatics,
University of Gdańsk, 80-308 Gdańsk, Poland

August 16, 2017

We begin with an example!



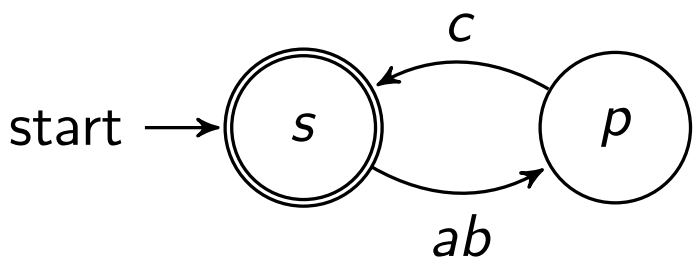
We begin with an example!



- Classical, sequential mode:

$$sabcabc \Rightarrow pcabc \Rightarrow sabc \Rightarrow pc \Rightarrow s$$

We begin with an example!



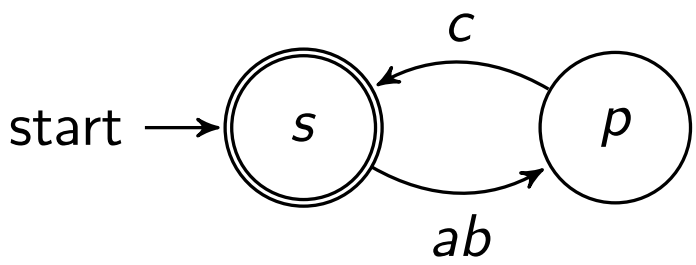
- Classical, sequential mode:

$$sabcabc \Rightarrow pcabc \Rightarrow sabc \Rightarrow pc \Rightarrow s$$

- Deleting jumping mode:

$$csabacb \curvearrowright capcb \curvearrowright csab \curvearrowright pc \curvearrowright s$$

We begin with an example!



- Classical, sequential mode:

$$sabcabc \Rightarrow pcabc \Rightarrow sabc \Rightarrow pc \Rightarrow s$$

- Deleting jumping mode:

$$csabacb \curvearrowright capcb \curvearrowright csab \curvearrowright pc \curvearrowright s$$

- Erasing jumping mode:

$$csababc \curvearrowright c\boxtimes\boxtimes abpc \curvearrowright c\boxtimes\boxtimes sab\boxtimes \curvearrowright pc\boxtimes\boxtimes\boxtimes\boxtimes\boxtimes \curvearrowright s\boxtimes\boxtimes\boxtimes\boxtimes\boxtimes\boxtimes$$

Some background

- Finite automata with deleting jumping mode [Meduna & Zemek, 2012, 2014].
- JFA (jumping finite automata) have rules $py \curvearrowright q$ with $|y| \leq 1$, GFJA (general jumping automata) have no restriction on rule length.

Some background

- Finite automata with deleting jumping mode [Meduna & Zemek, 2012, 2014].
- JFA (jumping finite automata) have rules $py \curvearrowright q$ with $|y| \leq 1$, GFJA (general jumping automata) have no restriction on rule length.
- The complexity of JFA and GJFA was studied in [Fernau, Paramasivan, Schmid, Vorel, 2017].

Some background

- Finite automata with deleting jumping mode [Meduna & Zemek, 2012, 2014].
- JFA (jumping finite automata) have rules $py \curvearrowright q$ with $|y| \leq 1$, GFJA (general jumping automata) have no restriction on rule length.
- The complexity of JFA and GJFA was studied in [Fernau, Paramasivan, Schmid, Vorel, 2017].
- Two-Dimensional Row Jumping Finite Automata (2-RJFA, 2-GRJFA) were introduced and studied in [Immanuel, Thomas, 2016].
- The erasing mode is used in the two-dimensional case.

How does a 2D jumping automaton work?

- The input is a rectangular array $X \in \Sigma^{**}$.
- Every row of X has a guarding symbol $\$ \notin \Sigma$ on the left and $\# \notin \Sigma$ on the right. X with these two additional columns is denoted by $\widehat{X}$.

How does a 2D jumping automaton work?

- The input is a rectangular array $X \in \Sigma^{**}$.
- Every row of X has a guarding symbol $\$ \notin \Sigma$ on the left and $\# \notin \Sigma$ on the right. X with these two additional columns is denoted by $\widehat{X}$.
- The 2D jumping row automaton A works on input $\widehat{X}$ as follows:
  - Choose nondeterministcally a row and a position in this row for the starting state.

How does a 2D jumping automaton work?

- The input is a rectangular array $X \in \Sigma^{**}$.
- Every row of X has a guarding symbol $\$ \notin \Sigma$ on the left and $\# \notin \Sigma$ on the right. X with these two additional columns is denoted by $\widehat{X}$.
- The 2D jumping row automaton A works on input $\widehat{X}$ as follows:
  - Choose nondeterministcally a row and a position in this row for the starting state.
  - Process this row using the erasing jumping mode.

How does a 2D jumping automaton work?

- The input is a rectangular array $X \in \Sigma^{**}$.
- Every row of X has a guarding symbol $\$ \notin \Sigma$ on the left and $\# \notin \Sigma$ on the right. X with these two additional columns is denoted by $\widehat{X}$.
- The 2D jumping row automaton A works on input $\widehat{X}$ as follows:
  - Choose nondeterministcally a row and a position in this row for the starting state.
  - Process this row using the erasing jumping mode.
  - Using row checking rules, check if all symbols were read.

How does a 2D jumping automaton work?

- The input is a rectangular array $X \in \Sigma^{**}$.
- Every row of X has a guarding symbol $\$ \notin \Sigma$ on the left and $\# \notin \Sigma$ on the right. X with these two additional columns is denoted by $\widehat{X}$.
- The 2D jumping row automaton A works on input $\widehat{X}$ as follows:
  - Choose nondeterministcally a row and a position in this row for the starting state.
  - Process this row using the erasing jumping mode.
  - Using row checking rules, check if all symbols were read.
  - Using a row jump rule, jump to another row. Delete from array the already processed row.

How does a 2D jumping automaton work?

- The input is a rectangular array $X \in \Sigma^{**}$.
- Every row of X has a guarding symbol $\$ \notin \Sigma$ on the left and $\# \notin \Sigma$ on the right. X with these two additional columns is denoted by $\widehat{X}$.
- The 2D jumping row automaton A works on input $\widehat{X}$ as follows:
  - Choose nondeterministcally a row and a position in this row for the starting state.
  - Process this row using the erasing jumping mode.
  - Using row checking rules, check if all symbols were read.
  - Using a row jump rule, jump to another row. Delete from array the already processed row.
  - Process all rows, one by one, until the array is empty.

How does a 2D jumping automaton work?

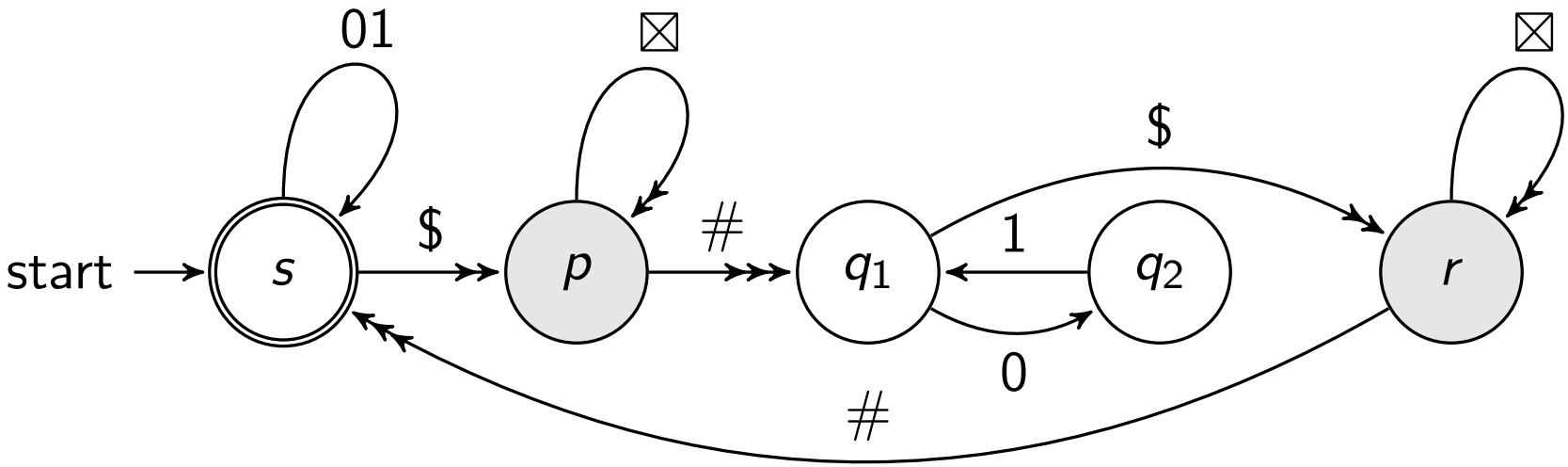
- The input is a rectangular array $X \in \Sigma^{**}$.
- Every row of X has a guarding symbol $\$ \notin \Sigma$ on the left and $\# \notin \Sigma$ on the right. X with these two additional columns is denoted by $\widehat{X}$.
- The 2D jumping row automaton A works on input $\widehat{X}$ as follows:
  - Choose nondeterministcally a row and a position in this row for the starting state.
  - Process this row using the erasing jumping mode.
  - Using row checking rules, check if all symbols were read.
  - Using a row jump rule, jump to another row. Delete from array the already processed row.
  - Process all rows, one by one, until the array is empty.
- Formally, a 2D jumping row automaton is an octuple $(Q, Q', \Sigma, R_1, R_2, R_3, s, F)$.

An example

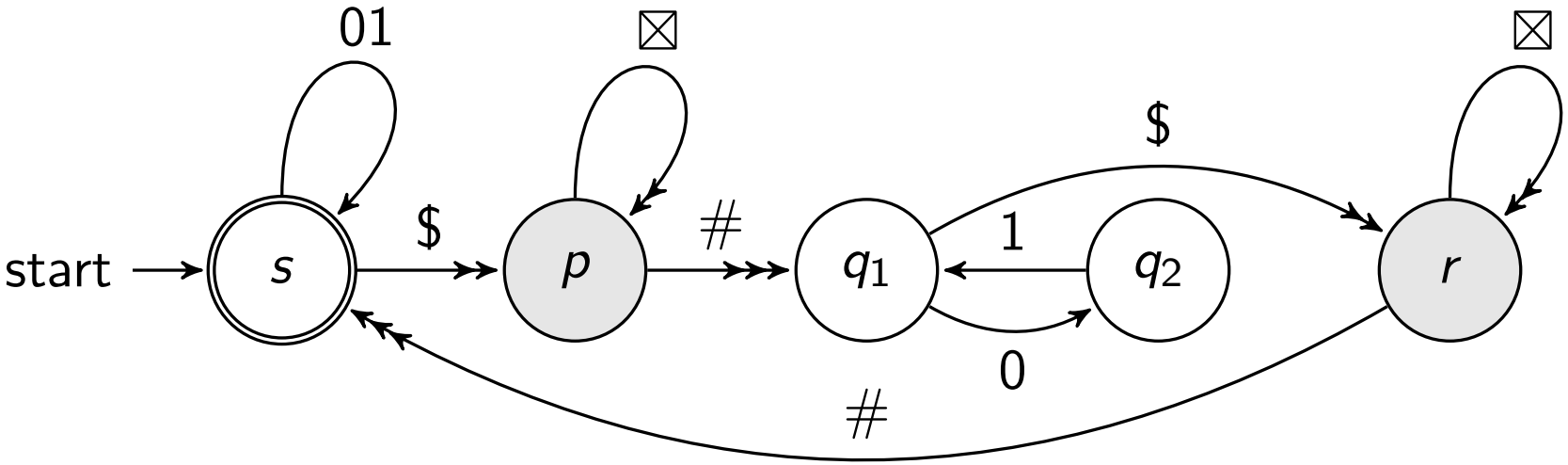
- Representation as a formula:
  $A = (\{s, p, q_1, q_2, r\}, \{p, r\}, \{0, 1\}, \{s01 \curvearrowright s, q_1 0 \curvearrowright q_2, q_2 1 \curvearrowright q_1, \},$
  $\{s\$ \to p, p\boxtimes \to p, q_1\$ \to r, r\boxtimes \to r\}, \{p\# \curvearrowleft q_1, r\# \curvearrowleft s\}, s, \{s\})$

An example

- Representation as a formula:
  $A = (\{s, p, q_1, q_2, r\}, \{p, r\}, \{0, 1\}, \{s01 \curvearrowright s, q_1 0 \curvearrowright q_2, q_2 1 \curvearrowright q_1, \}, \{s\$ \to p, p\boxtimes \to p, q_1\$ \to r, r\boxtimes \to r\}, \{p\# \rightsquigarrow q_1, r\# \rightsquigarrow s\}, s, \{s\})$
- Graphical representation:

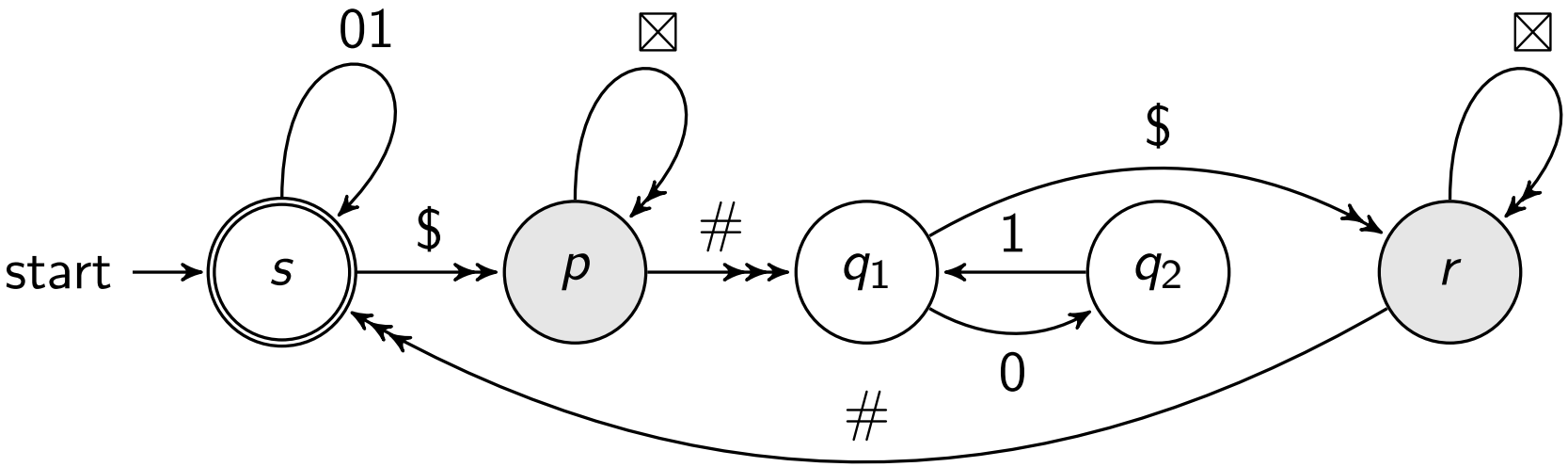


An example



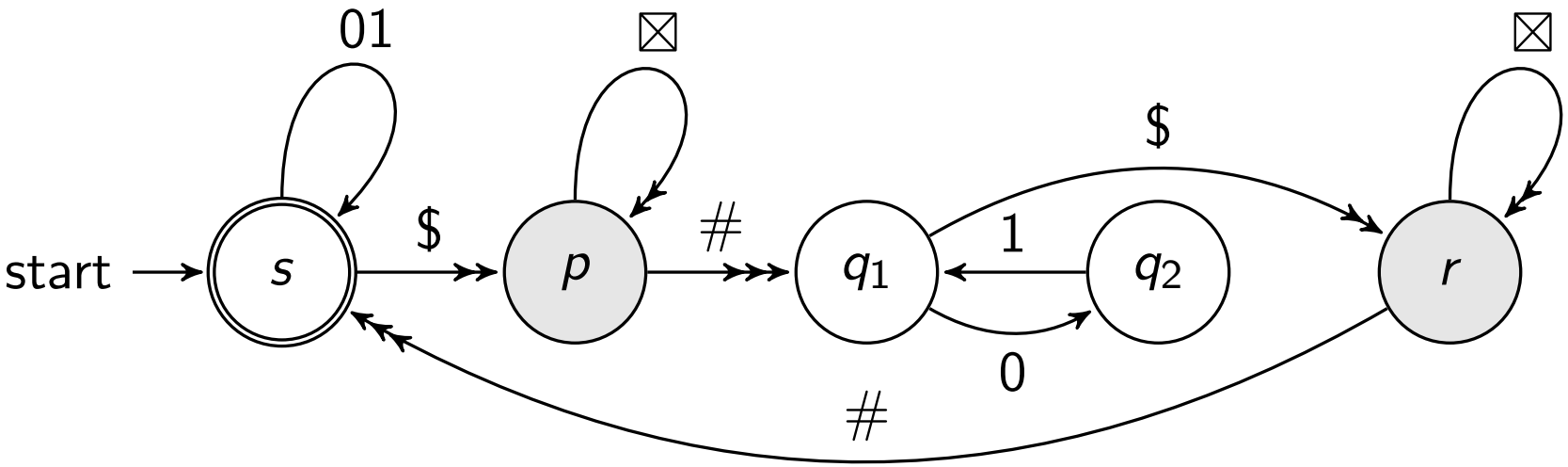
$	0	1	0	1	#
$	0	1	1	0	#
$	1	1	0	0	#
$	0	1	s0	1	#

An example



$	0	1	0	1	#
$	0	1	1	0	#
$	1	1	0	0	#
$	$s0$	1	⊠	⊠	#

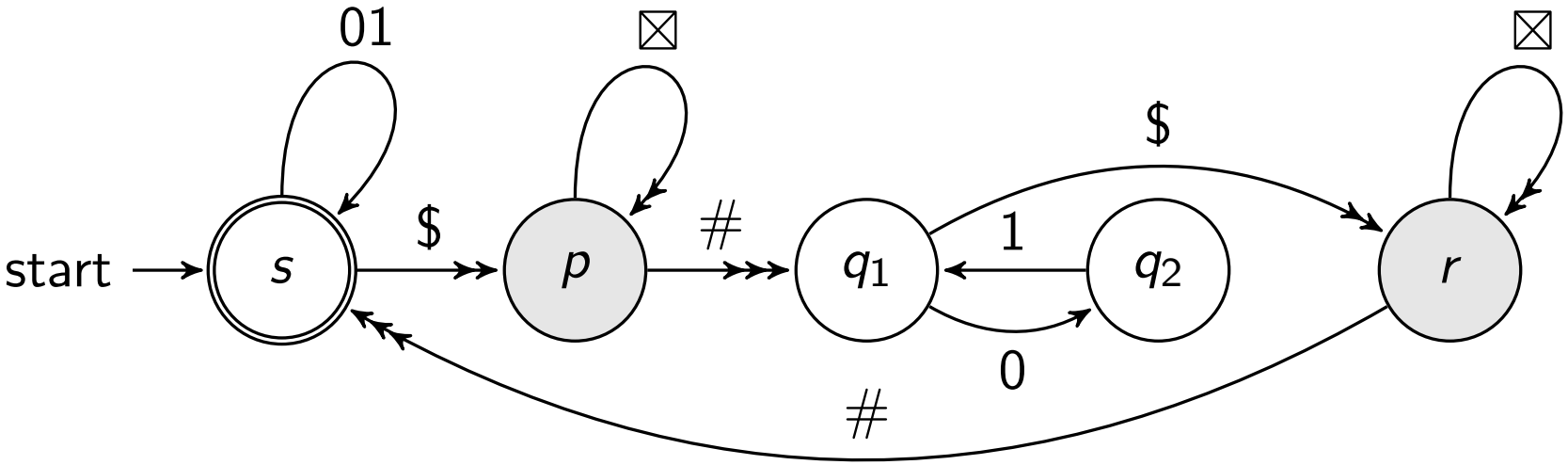
An example



$	0	1	0	1	#
$	0	1	1	0	#
$	1	1	0	0	#
s	⊠	⊠	⊠	⊠	#

$\rightarrow$

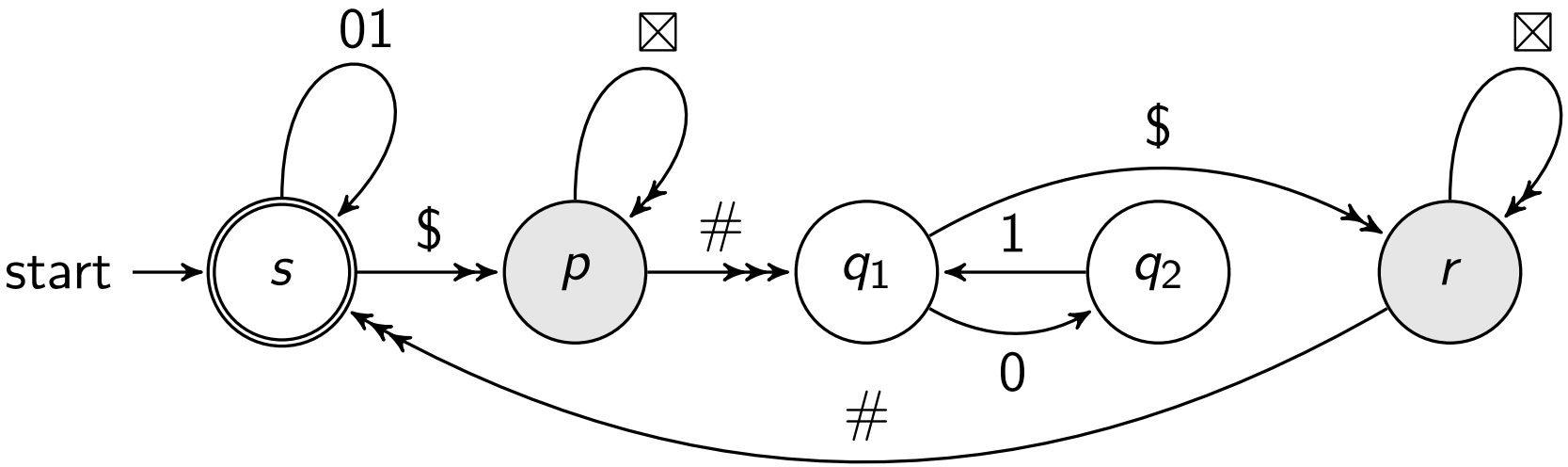
An example



$	0	1	0	1	#
$	0	1	1	0	#
$	1	1	0	0	#
$	$p\boxtimes$	⊠	⊠	⊠	#

$\rightarrow$

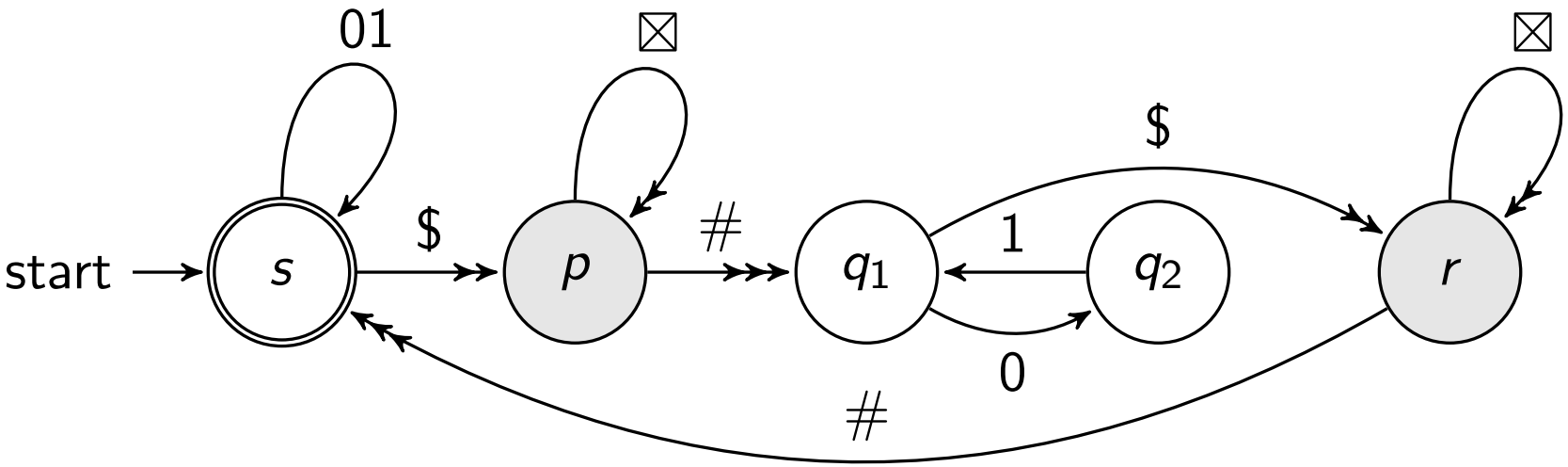
An example



$	0	1	0	1	#
$	0	1	1	0	#
$	1	1	0	0	#
$	⊠	p⊠	⊠	⊠	#

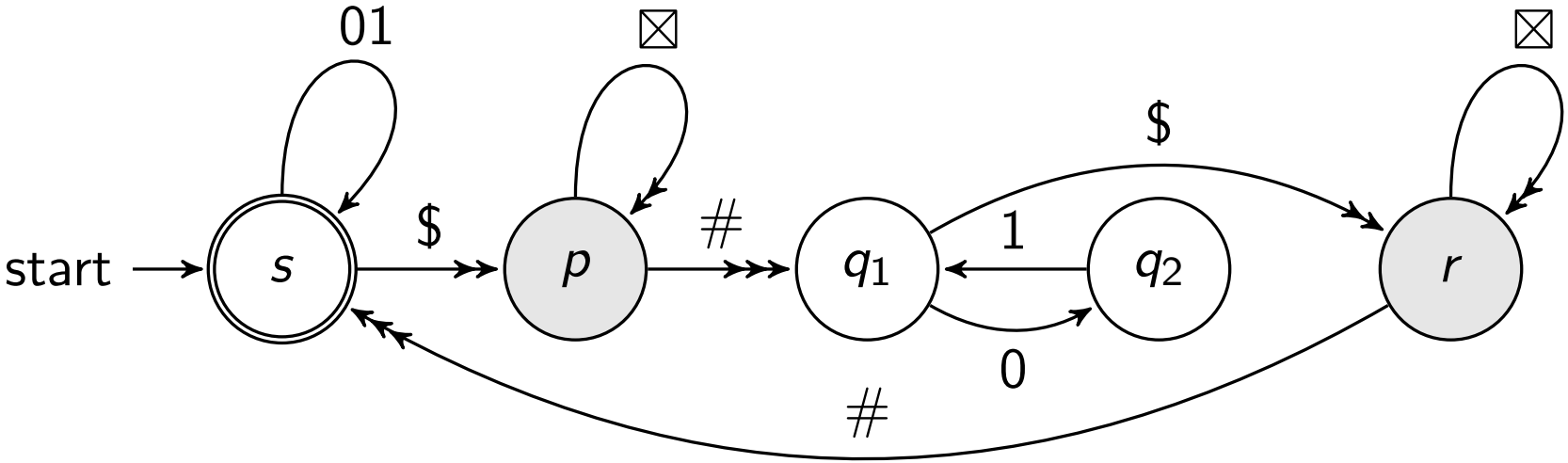
$\rightarrow \cdots \rightarrow$

An example



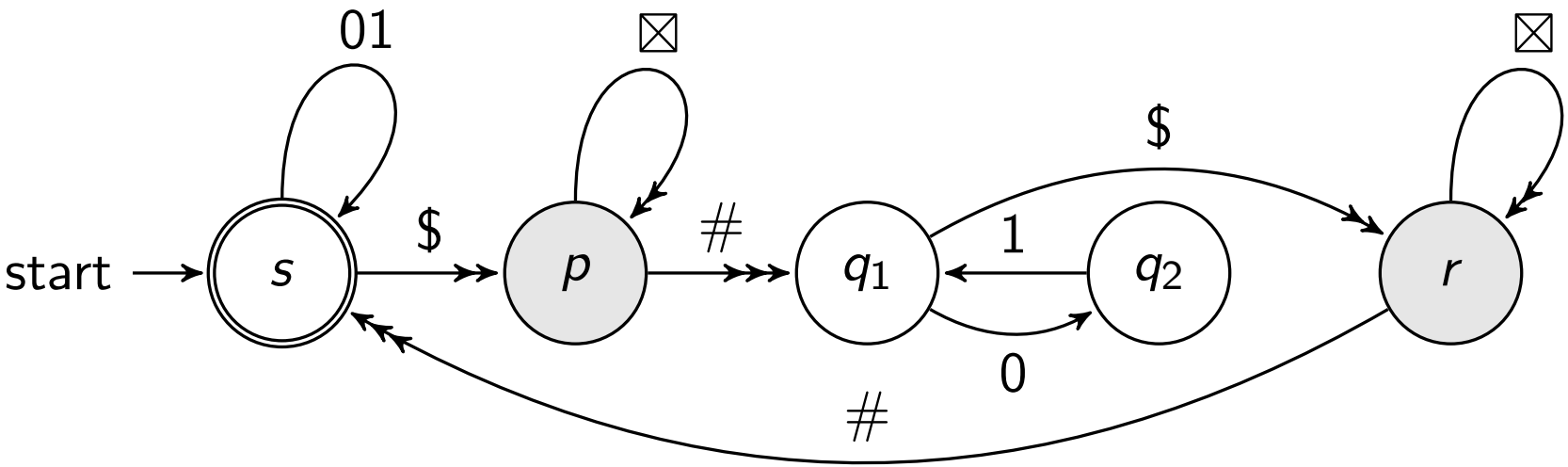
$	0	1	0	1	#
$	0	1	1	0	#
$	1	1	0	0	#
$	⊠	⊠	⊠	⊠	p#

An example



$	0	1	0	1	#
$	0	1	1	$q_1 0$	#
$	1	1	0	0	#

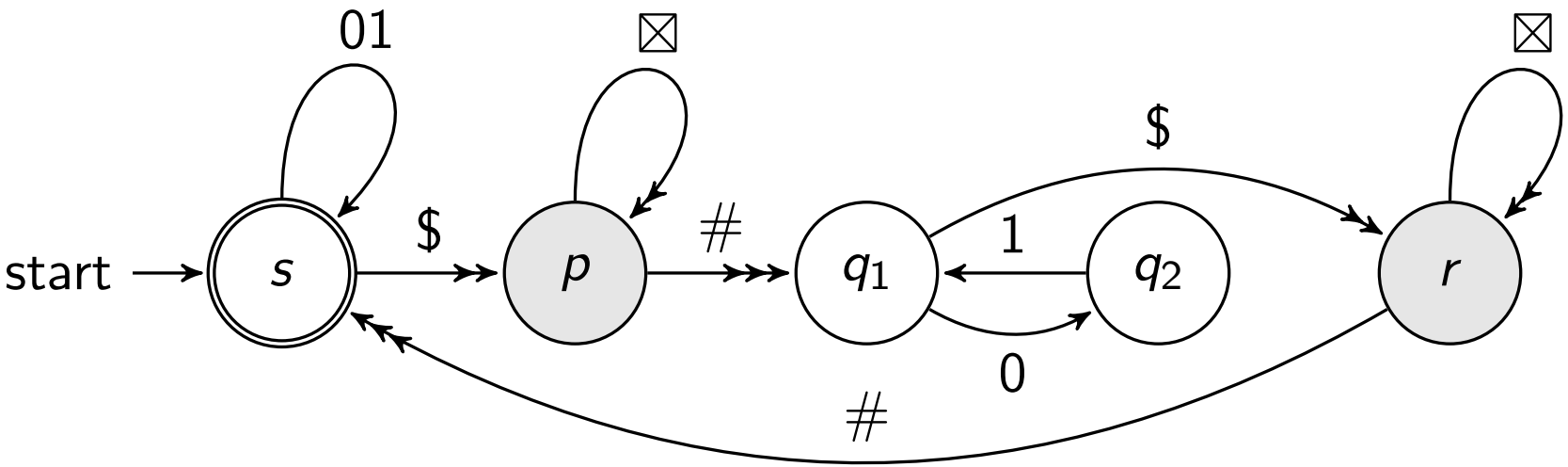
An example



$	0	1	0	1	#
$	0	$q_2 1$	1	⊠	#
$	1	1	0	0	#

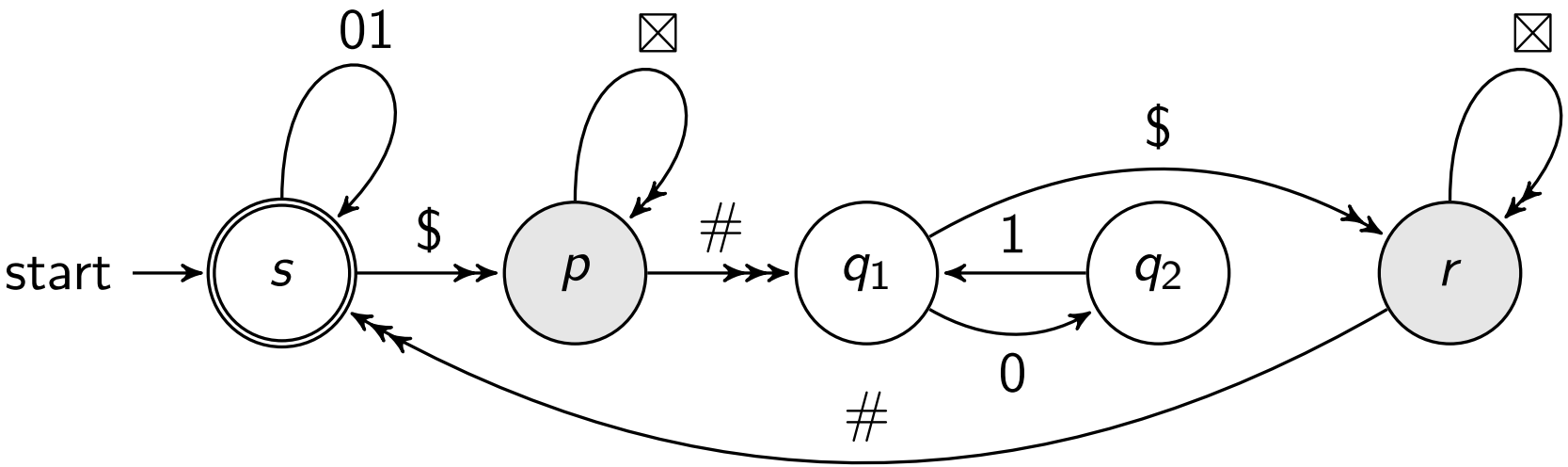
$\curvearrowright \cdots \curvearrowright$

An example



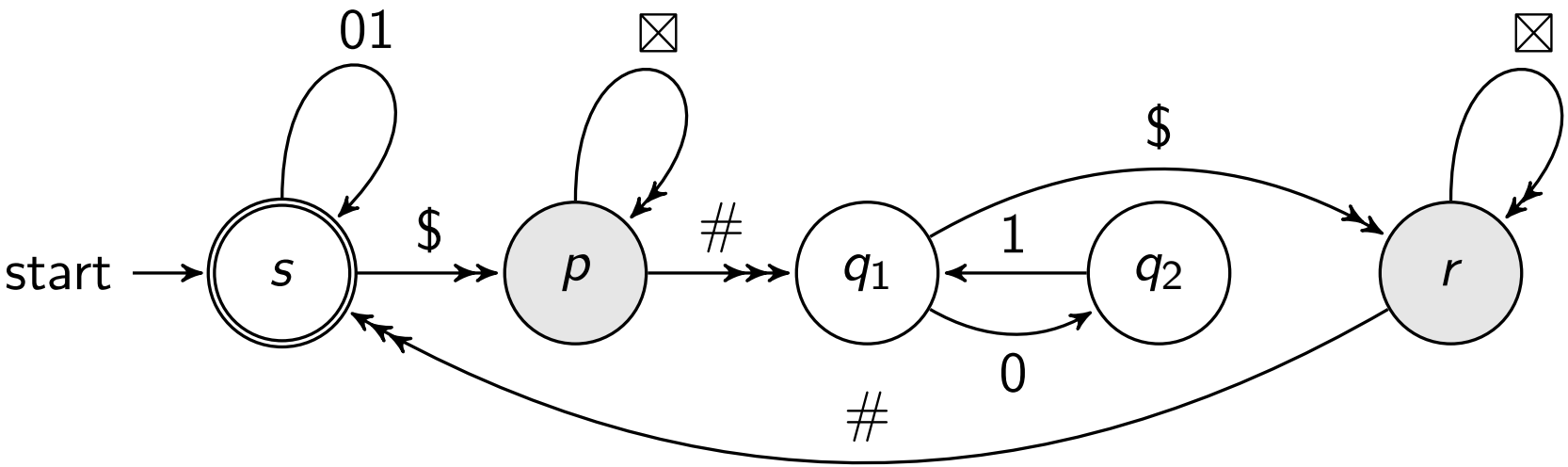
$	0	1	0	1	#	
q_1 $	⊠	⊠	⊠	⊠	#	→
$	1	1	0	0	#	

An example



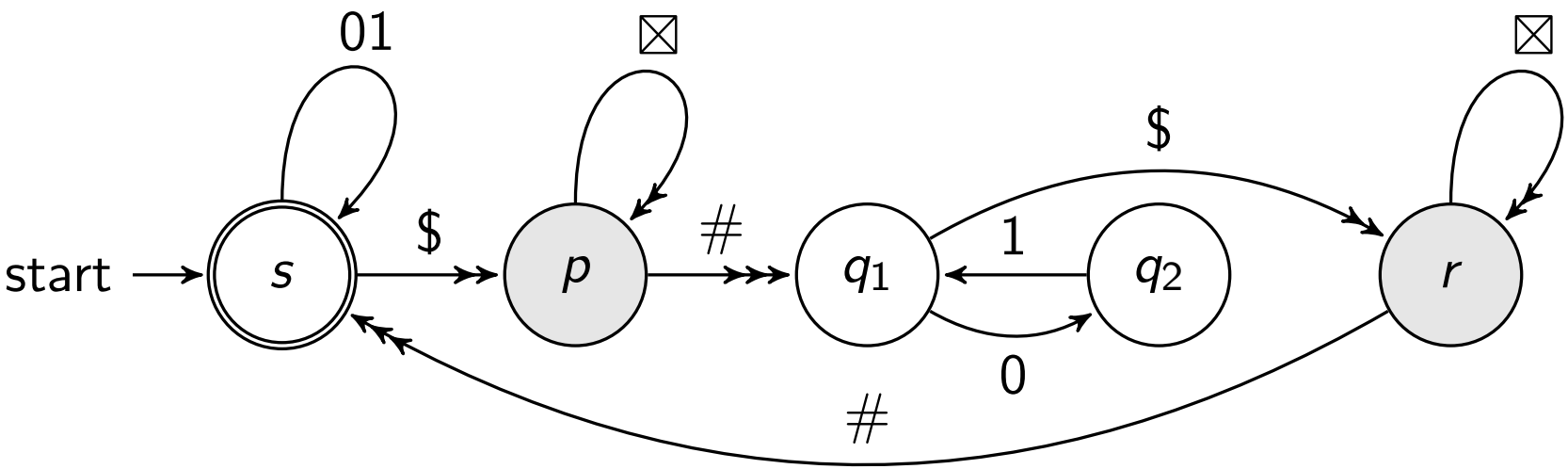
$	0	1	0	1	#	
$	r⊠	⊠	⊠	⊠	#	$\rightarrow \cdots \rightarrow$
$	1	1	0	0	#	

An example



$	0	1	0	1	#
$	⊠	⊠	⊠	⊠	r#
$	1	1	0	0	#

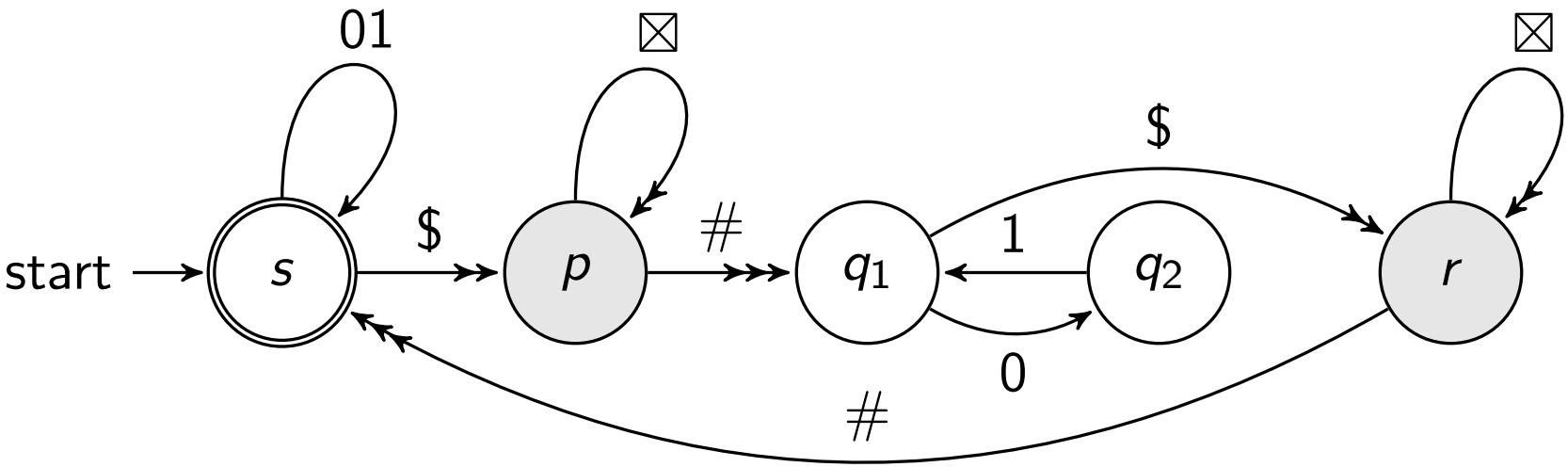
An example



$	$s0$	1	0	1	#
$	1	1	0	0	#

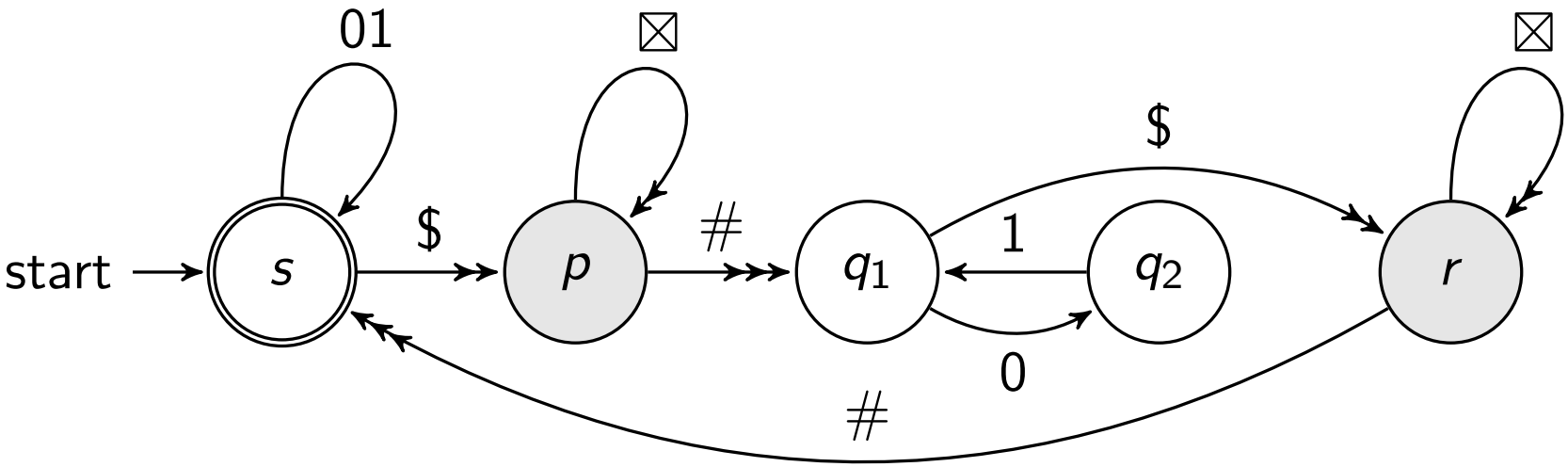
$\curvearrowright \cdots \curvearrowright$

An example



$	1	1	$q_1 0$	0	#	$\curvearrowright \cdots \curvearrowleft s$

An example



$L(A) = \{X \in \{0,1\}^{**} : X$ has n rows of the form $(01)^*$ and n rows containing the same number of 0's and 1's, where $n \geq 0\}$

Results

The aim of this work was to analyse the time complexity of the following two problems:

UNIFORM M.P. FOR 2-RJFA
INPUT: a 2-RJFA A and an array $X \in \Sigma^{**}$
QUESTION: $X \in L(A)$?

UNIFORM M.P. FOR 2-GRJFA
INPUT: a 2-GRJFA A and an array $X \in \Sigma^{**}$
QUESTION: $X \in L(A)$?

Results for 2-RJFA

Theorem

Uniform m.p. for 2-rjfa *is NP-complete, even under the restriction that* $|\Sigma| = 2$.

Results for 2-RJFA

Theorem

UNIFORM M.P. FOR 2-RJFA *is NP-complete, even under the restriction that* $|\Sigma| = 2$.

Proof.

(Idea) To prove NP-hardness, we use the problem:
3-PARTITION
INPUT: a sequence of natural numbers $S = (n_1, n_2, ..., n_{3m})$ given in unary, where $m \geq 0$,
QUESTION: can S be partitioned into triplets $(n_{i_1}, n_{j_1}, n_{k_1})$,$(n_{i_2}, n_{j_2}, n_{k_2})$, ..., $(n_{i_m}, n_{j_m}, n_{k_m})$, such that $\bigcup_{1 \leq r \leq m} \{i_r, j_r, k_r\} = \{1, 2, ..., 3m\}$ and for each r, $n_{i_r} + n_{j_r} + n_{k_r} = B$ (each triplet sums to the same number)?
Based on it, we construct an automaton A and an array X such that S has 3-partition if and only if $X \in L(A)$. □

Results for 2-GRJFA

Theorem

The uniform membership problem for 2-GRJFA is NP-complete, even under the restrictions that:

- $|\Sigma| = 2$ *(binary alphabet), and*
- *all inputs are single-row arrays (one dimensional case).*

Results for 2-GRJFA

Theorem

The uniform membership problem for 2-GRJFA is NP-complete, even under the restrictions that:

- $|\Sigma| = 2$ *(binary alphabet), and*
- *all inputs are single-row arrays (one dimensional case).*

Proof.

The idea of proving NP-hardness is similar. Based on a 3-partition problem, we construct an automaton A and an array X such that S has 3-partition if and only if $X \in L(A)$. □

Summary of results

	JFA	GJFA	2-RJFA	2-GRJFA
Uniform M.P.	NPC	NPC	**NPC**	**NPC**†
Uniform M.P. with $\lvert\Sigma\rvert = 2$	P	NPC	**NPC**	**NPC**†
† even under the restriction that the input arrays have one row				

The results of this paper are presented in bold. The results for one-dimensional case with deleting mode are from the work [Fernau, Paramasivan, Schmid, Vorel, 2017].

Some remarks and questions

- A broader study of the time complexity will be published in an extended paper.
- Deleting vs erasing jumping mode.
- Study other variations of 2D automata (left, right, one-way jumps or row jumps).

Thank you for your attention!