On the complexity of unary tiling-recognizable languages

A. Bertoni, M. Goldwurm, V. Lonati

Università degli studi di Milano
Dipartimento di Scienze dell'Informazione

Two-dimensional languages

Given a finite alphabet Σ:

- a (non-empty) picture, or two-dimensional string, is a two-dimensional array of elements of Σ

Two-dimensional languages

Given a finite alphabet Σ:

- a (non-empty) picture, or two-dimensional string, is a two-dimensional array of elements of Σ
- the set of all pictures over Σ is denoted by Σ^{**}

Two-dimensional languages

Given a finite alphabet Σ:

- a (non-empty) picture, or two-dimensional string, is a two-dimensional array of elements of Σ
- the set of all pictures over Σ is denoted by Σ^{**}
- a two-dimensional language over Σ is any set $L \subseteq \Sigma^{**}$

Two-dimensional languages

Given a finite alphabet Σ:

- a (non-empty) picture, or two-dimensional string, is a two-dimensional array of elements of Σ
- the set of all pictures over Σ is denoted by Σ^{**}
- a two-dimensional language over Σ is any set $L \subseteq \Sigma^{**}$
- the size of a picture p is the pair (r_p, c_p) where
  - $r_p =$ number of rows of p
  - $c_p =$ number of columns of p

Two-dimensional languages

Given a finite alphabet Σ:

- a (non-empty) picture, or two-dimensional string, is a two-dimensional array of elements of Σ
- the set of all pictures over Σ is denoted by Σ^{**}
- a two-dimensional language over Σ is any set $L \subseteq \Sigma^{**}$
- the size of a picture p is the pair (r_p, c_p) where
  - $r_p =$ number of rows of p
  - $c_p =$ number of columns of p
- the border of p is defined as follows

Two-dimensional languages

Given a finite alphabet Σ:

- a (non-empty) picture, or two-dimensional string, is a two-dimensional array of elements of Σ
- the set of all pictures over Σ is denoted by Σ^{**}
- a two-dimensional language over Σ is any set $L \subseteq \Sigma^{**}$
- the size of a picture p is the pair (r_p, c_p) where
  - $r_p =$ number of rows of p
  - $c_p =$ number of columns of p
- the border of p is defined as follows

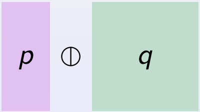
$$\begin{array}{ccc} \sharp & \sharp\cdots\sharp & \sharp \\ \sharp & & \sharp \\ \vdots & p & \vdots \\ \sharp & & \sharp \\ \sharp & \sharp\cdots\sharp & \sharp \end{array} = \hat{p}$$

Operations on two-dimensional languages

- Set operations

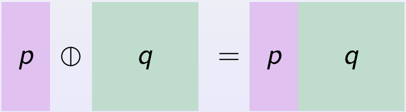
Operations on two-dimensional languages

- Set operations
- Column concatenation
  between pictures with the same number of rows



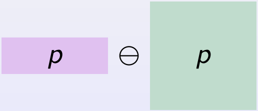
Operations on two-dimensional languages

- Set operations
- Column concatenation
  between pictures with the same number of rows



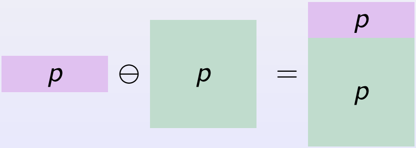
Operations on two-dimensional languages

- Set operations
- Column concatenation
  between pictures with the same number of rows
- Row concatenation
  between pictures with the same number of columns



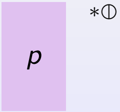
Operations on two-dimensional languages

- Set operations
- Column concatenation
  between pictures with the same number of rows
- Row concatenation
  between pictures with the same number of columns



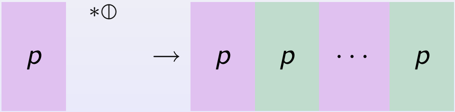
Operations on two-dimensional languages

- Set operations
- Column concatenation
  between pictures with the same number of rows
- Row concatenation
  between pictures with the same number of columns
- Column and row closures



Operations on two-dimensional languages

- Set operations
- Column concatenation
  between pictures with the same number of rows
- Row concatenation
  between pictures with the same number of columns
- Column and row closures



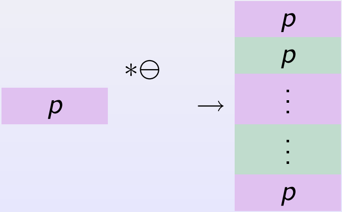
Operations on two-dimensional languages

- Set operations
- Column concatenation
  between pictures with the same number of rows
- Row concatenation
  between pictures with the same number of columns
- Column and row closures

$$\boxed{p}^{*\ominus}$$

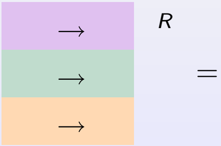
Operations on two-dimensional languages

- Set operations
- Column concatenation
  between pictures with the same number of rows
- Row concatenation
  between pictures with the same number of columns
- Column and row closures



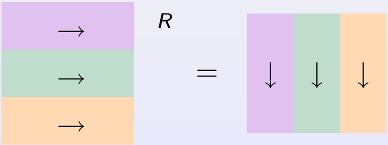
Operations on two-dimensional languages

- Set operations
- Column concatenation
  between pictures with the same number of rows
- Row concatenation
  between pictures with the same number of columns
- Column and row closures
- Rotation



Operations on two-dimensional languages

- Set operations
- Column concatenation
  between pictures with the same number of rows
- Row concatenation
  between pictures with the same number of columns
- Column and row closures
- Rotation



The class Rec

[Giammarresi, Restivo '97 - Handbook of formal languages]

The class Rec

[Giammarresi, Restivo '97 - Handbook of formal languages]

- Rec is a class of two-dimensional languages

The class Rec

[Giammarresi, Restivo '97 - Handbook of formal languages]

- Rec is a class of two-dimensional languages
- Rec tries to extends the concept of regular string language

The class REC

[Giammarresi, Restivo '97 - Handbook of formal languages]

- REC is a class of two-dimensional languages
- REC tries to extends the concept of regular string language
- REC can be defined using different approach:
  - regular expressions
  - online tessellation automata
  - logic formulas
  - tiling systems

Tiling recognizability

- A tile is a square picture of size 2. Given any picture p, we write $T(p)$ to denote the set of tiles contained in p.

Tiling recognizability

- A tile is a square picture of size 2. Given any picture p, we write $T(p)$ to denote the set of tiles contained in p.
- $L \subseteq \Gamma^{**}$ is a local language if there exists a finite set Θ of tiles such that $L = \{p \in \Gamma^{**} \mid T(\hat{p}) \subseteq \Theta\}$. We write $L = \mathcal{L}(\Theta)$.

Tiling recognizability

- A tile is a square picture of size 2. Given any picture p, we write $T(p)$ to denote the set of tiles contained in p.
- $L \subseteq \Gamma^{**}$ is a local language if there exists a finite set Θ of tiles such that $L = \{p \in \Gamma^{**} \mid T(\hat{p}) \subseteq \Theta\}$. We write $L = \mathcal{L}(\Theta)$.
- A tiling system is defined by
  - a projection $\pi : \Gamma \rightarrow \Sigma$ between two alphabets
  - a finite set of tiles $\Theta \subseteq (\Gamma \cup \{\sharp\})^{**}$

Tiling recognizability

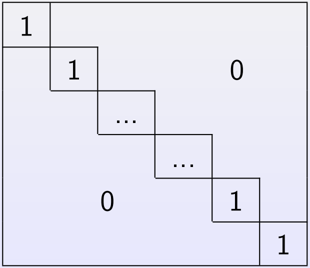
- A tile is a square picture of size 2. Given any picture p, we write $T(p)$ to denote the set of tiles contained in p.
- $L \subseteq \Gamma^{**}$ is a local language if there exists a finite set Θ of tiles such that $L = \{p \in \Gamma^{**} \mid T(\hat{p}) \subseteq \Theta\}$. We write $L = \mathcal{L}(\Theta)$.
- A tiling system is defined by
  - a projection $\pi : \Gamma \to \Sigma$ between two alphabets
  - a finite set of tiles $\Theta \subseteq (\Gamma \cup \{\sharp\})^{**}$
- A language L is in REC if there exists a tiling system such that $L = \pi(\mathcal{L}(\Theta))$.

Tiling recognizability

- A tile is a square picture of size 2. Given any picture p, we write $T(p)$ to denote the set of tiles contained in p.
- $L \subseteq \Gamma^{**}$ is a local language if there exists a finite set Θ of tiles such that $L = \{p \in \Gamma^{**} \mid T(\hat{p}) \subseteq \Theta\}$. We write $L = \mathcal{L}(\Theta)$.
- A tiling system is defined by
  - a projection $\pi : \Gamma \to \Sigma$ between two alphabets
  - a finite set of tiles $\Theta \subseteq (\Gamma \cup \{\sharp\})^{**}$
- A language L is in $\textsc{Rec}$ if there exists a tiling system such that $L = \pi(\mathcal{L}(\Theta))$.
- $\textsc{Rec}$ is closed w.r.t the operations $\cup, \varobar, \ominus, {}^{*\varobar}, {}^{*\ominus}, {}^{R}$.

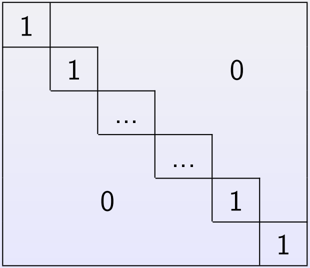
Example: squares

- The set of squares with 1 on the main diagonal and 0 in all other position is a local language



Example: squares

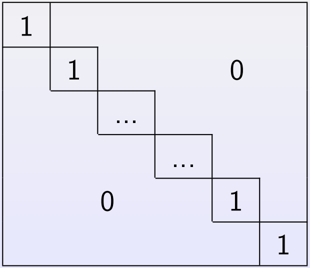
- The set of squares with 1 on the main diagonal and 0 in all other position is a local language
- The set of tiles is given by $T(p)$



$$p = \begin{array}{|c|c|c|c|c|c|} \hline \sharp & \sharp & \sharp & \sharp & \sharp & \sharp \\ \hline \sharp & 1 & 0 & 0 & 0 & \sharp \\ \hline \sharp & 0 & 1 & 0 & 0 & \sharp \\ \hline \sharp & 0 & 0 & 1 & 0 & \sharp \\ \hline \sharp & 0 & 0 & 0 & 1 & \sharp \\ \hline \sharp & \sharp & \sharp & \sharp & \sharp & \sharp \\ \hline \end{array}$$

Example: squares

- The set of squares with 1 on the main diagonal and 0 in all other position is a local language
- The set of tiles is given by $T(p)$
- The set of unary squares is in $\textsc{Rec}$ (it is the projection of the previous languages).



$$p = \begin{array}{|c|c|c|c|c|c|} \hline \sharp & \sharp & \sharp & \sharp & \sharp & \sharp \\ \hline \sharp & 1 & 0 & 0 & 0 & \sharp \\ \hline \sharp & 0 & 1 & 0 & 0 & \sharp \\ \hline \sharp & 0 & 0 & 1 & 0 & \sharp \\ \hline \sharp & 0 & 0 & 0 & 1 & \sharp \\ \hline \sharp & \sharp & \sharp & \sharp & \sharp & \sharp \\ \hline \end{array}$$

Quasi-unary strings

- Unary pictures are pictures over a one-letter alphabet $\{\circ\}$.

Quasi-unary strings

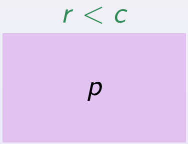
- Unary pictures are pictures over a one-letter alphabet $\{\circ\}$.
- A unary picture p is identified by its size (r_p, c_p).

Quasi-unary strings

- Unary pictures are pictures over a one-letter alphabet $\{\circ\}$.
- A unary picture p is identified by its size (r_p, c_p).
- We represent unary pictures by quasi-unary strings over the alphabet $\{\circ, h, v\}$:

Quasi-unary strings

- Unary pictures are pictures over a one-letter alphabet $\{\circ\}$.
- A unary picture p is identified by its size (r_p, c_p).
- We represent unary pictures by quasi-unary strings over the alphabet $\{\circ, h, v\}$:



Quasi-unary strings

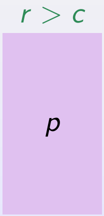
- Unary pictures are pictures over a one-letter alphabet $\{\circ\}$.
- A unary picture p is identified by its size (r_p, c_p).
- We represent unary pictures by quasi-unary strings over the alphabet $\{\circ, h, v\}$:

$r < c$

$$\phi(p) = \overbrace{\circ\cdots\circ}^{r}\, h\, \overbrace{\circ\cdots\circ}^{c-r-1} \qquad \in Q_h$$

Quasi-unary strings

- Unary pictures are pictures over a one-letter alphabet $\{\circ\}$.
- A unary picture p is identified by its size (r_p, c_p).
- We represent unary pictures by quasi-unary strings over the alphabet $\{\circ, h, v\}$:



Quasi-unary strings

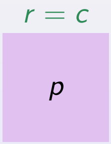
- Unary pictures are pictures over a one-letter alphabet $\{\circ\}$.
- A unary picture p is identified by its size (r_p, c_p).
- We represent unary pictures by quasi-unary strings over the alphabet $\{\circ, h, v\}$:

$r > c$

$$\phi(p) = \overbrace{\circ\cdots\circ}^{c} \; v \; \overbrace{\circ\cdots\circ}^{r-c-1} \qquad \in Q_v$$

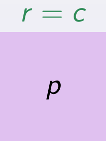
Quasi-unary strings

- Unary pictures are pictures over a one-letter alphabet $\{\circ\}$.
- A unary picture p is identified by its size (r_p, c_p).
- We represent unary pictures by quasi-unary strings over the alphabet $\{\circ, h, v\}$:



Quasi-unary strings

- Unary pictures are pictures over a one-letter alphabet $\{\circ\}$.
- A unary picture p is identified by its size (r_p, c_p).
- We represent unary pictures by quasi-unary strings over the alphabet $\{\circ, h, v\}$:



$$\phi(p) = \overbrace{\circ \cdots \circ}^{r} \; v \; \overbrace{\circ \cdots \circ}^{r} \qquad \in Q_s$$

Size of quasi-unary strings

- Given any quasi-unary string $x \in Q_h \cup Q_v \cup Q_s$:
  - the length of x is denoted by $|x|$
  - the length of the longest prefix in $\circ^*$ is denoted by ${}_\circ|x|$

Size of quasi-unary strings

- Given any quasi-unary string $x \in Q_h \cup Q_v \cup Q_s$:
  - the length of x is denoted by $|x|$
  - the length of the longest prefix in $\circ^*$ is denoted by $_\circ|x|$
- Given any unary picture p, if $\phi(p)$ is the quasi-unary string representing p we have:

$$|\phi(p)| = \max(r_p, c_p)$$

Size of quasi-unary strings

- Given any quasi-unary string $x \in Q_h \cup Q_v \cup Q_s$:
  - the length of x is denoted by $|x|$
  - the length of the longest prefix in $\circ^*$ is denoted by $_\circ|x|$
- Given any unary picture p, if $\phi(p)$ is the quasi-unary string representing p we have:

$$|\phi(p)| = \max(r_p, c_p) \quad \text{and} \quad {}_\circ|\phi(p)| = \min(r_p, c_p)$$

Size of quasi-unary strings

- Given any quasi-unary string $x \in Q_h \cup Q_v \cup Q_s$:
  - the length of x is denoted by $|x|$
  - the length of the longest prefix in $\circ^*$ is denoted by $_\circ|x|$
- Given any unary picture p, if $\phi(p)$ is the quasi-unary string representing p we have:

$$|\phi(p)| = \max(r_p, c_p) \quad \text{and} \quad {}_\circ|\phi(p)| = \min(r_p, c_p)$$

Ex: p

$r < c$

$$\phi(p) = \overbrace{\circ \cdots \circ}^{r} \, h \, \overbrace{\circ \cdots \circ}^{c-r-1} \qquad \in Q_h$$

The class $\textsc{N-LinSpaceRev}_{QU}$

Complexity class for quasi-unary strings

$\textsc{N-LinSpaceRev}_{QU}$ is the class of quasi-unary languages that can be recognized by a Turing machine M

The class $\textsc{N-LinSpaceRev}_{QU}$

Complexity class for quasi-unary strings

$\textsc{N-LinSpaceRev}_{QU}$ is the class of quasi-unary languages that can be recognized by a Turing machine M

- with 1-tape
- nondeterministic

The class N-LinSpaceRev$_{QU}$

Complexity class for quasi-unary strings

N-LinSpaceRev$_{QU}$ is the class of quasi-unary languages that can be recognized by a Turing machine M

- with 1-tape
- nondeterministic

such that, on every input x,

The class $\text{N-LinSpaceRev}_{QU}$

Complexity class for quasi-unary strings

$\text{N-LinSpaceRev}_{QU}$ is the class of quasi-unary languages that can be recognized by a Turing machine M

- with 1-tape
- nondeterministic

such that, on every input x,

- M works within $|x|$ space,
- M executes ${}_{\circ}|x|$ head reversals at most.

Characterization of $\textsc{Rec}_1$

Theorem

Given any two-dimensional unary language L, the following statements are equivalents

- *L is in* $\textsc{Rec}_1$
- $\phi(L)$ *belongs to* $\textsc{N-LinSpaceRev}_{QU}$.

Recognizability implies the complexity bound

$$L \in \textsc{Rec} \qquad \Longrightarrow \qquad \phi(L) \in \text{N-}\textsc{LinSpaceRev}_{QU}$$

Recognizability implies the complexity bound

$$L \in \textsc{Rec} \qquad \Longrightarrow \qquad \phi(L) \in \text{N-}\textsc{LinSpaceRev}_{QU}$$

Sketch of the proof

- Consider a tiling system for L and let Θ be its set of tiles.

Recognizability implies the complexity bound

$$L \in \text{REC} \qquad \Longrightarrow \qquad \phi(L) \in \text{N-LINSPACEREV}_{QU}$$

Sketch of the proof

- Consider a tiling system for L and let Θ be its set of tiles.
- $\phi(L)$ corresponds to the problem SIZE REPR (Θ):

Recognizability implies the complexity bound

$$L \in \textsc{Rec} \quad \Longrightarrow \quad \phi(L) \in \text{N-}\textsc{LinSpaceRev}_{QU}$$

Sketch of the proof

- Consider a tiling system for L and let Θ be its set of tiles.
- $\phi(L)$ corresponds to the problem $\textsc{Size Repr}\ (\Theta)$:
  - Instance: a quasi-unary string $x \in Q$.
  - Question: does there exist $p \in \mathcal{L}(\Theta)$ whose size is represented by x?

Recognizability implies the complexity bound

$$L \in \ \textsc{Rec} \qquad \Longrightarrow \qquad \phi(L) \in \textsc{N-LinSpaceRev}_{QU}$$

Sketch of the proof

- Consider a tiling system for L and let Θ be its set of tiles.
- $\phi(L)$ corresponds to the problem $\textsc{Size Repr}$ (Θ):
  - Instance: a quasi-unary string $x \in Q$.
  - Question: does there exist $p \in \mathcal{L}(\Theta)$ whose size is represented by x?
- The problem $\textsc{Size Repr}$ (Θ) is in $\textsc{N-LinSpaceRev}_{QU}$ for every finite set Θ of tiles.

Recognizability implies the complexity bound

$$L \in \textsc{Rec} \quad \Longrightarrow \quad \phi(L) \in \textsc{N-LinSpaceRev}_{QU}$$

Sketch of the proof

- Consider a tiling system for L and let Θ be its set of tiles.
- $\phi(L)$ corresponds to the problem $\textsc{Size Repr}\ (\Theta)$:
  - Instance: a quasi-unary string $x \in Q$.
  - Question: does there exist $p \in \mathcal{L}(\Theta)$ whose size is represented by x?
- The problem $\textsc{Size Repr}\ (\Theta)$ is in $\textsc{N-LinSpaceRev}_{QU}$ for every finite set Θ of tiles.

  $\Longrightarrow \quad \phi(L)$ is in $\textsc{N-LinSpaceRev}_{QU}$.

The problem Size Repris in $\text{N-LinSpaceRev}_{QU}$

The following Turing machine solve the problem Size Repr (Θ):

- M tries to generate some $p \in \mathcal{L}(\Theta)$ of the required size

The problem SIZE REPRis in N-LINSPACEREV$_{QU}$

The following Turing machine solve the problem SIZE REPR (Θ):

- M tries to generate some $p \in \mathcal{L}(\Theta)$ of the required size
  - first M establishes if the input x is Q_h, Q_v, or Q_s;

The problem $\textsc{Size Repr}$is in $\textsc{N-LinSpaceRev}_{QU}$

The following Turing machine solve the problem $\textsc{Size Repr}$ (Θ):

- M tries to generate some $p \in \mathcal{L}(\Theta)$ of the required size
  - first M establishes if the input x is Q_h, Q_v, or Q_s;
  - if $x \in Q_h$ or $x \in Q_s$, then the generation is performed row by row,

The problem SIZE REPRis in N-LINSPACEREV$_{QU}$

The following Turing machine solve the problem SIZE REPR (Θ):

- M tries to generate some $p \in \mathcal{L}(\Theta)$ of the required size
  - first M establishes if the input x is Q_h, Q_v, or Q_s;
  - if $x \in Q_h$ or $x \in Q_s$, then the generation is performed row by row,
  - otherwise the generation has to be done column by column.

The problem SIZE REPRis in $\text{N-LINSPACEREV}_{QU}$

The following Turing machine solve the problem SIZE REPR (Θ):

- M tries to generate some $p \in \mathcal{L}(\Theta)$ of the required size
  - first M establishes if the input x is Q_h, Q_v, or Q_s;
  - if $x \in Q_h$ or $x \in Q_s$, then the generation is performed row by row,
  - otherwise the generation has to be done column by column.
- The input is accepted if and only if such a generating process can be accomplished.

Tape

$\circ$	$\cdots$	$\circ$	h	$\circ$	$\cdots$	$\circ$

Tape

$\sharp$ p_{11}	$\cdots$	$\circ$	h	$\circ$	$\cdots$	$\circ$

$$\begin{array}{|c|c|}\hline \sharp & \sharp \\ \hline \sharp & p_{11} \\ \hline \end{array} \in \Theta$$

Tape

$\sharp$ p_{11}	$\sharp$ $\cdots$	$\circ$	h	$\circ$	$\cdots$	$\circ$

$$\begin{array}{|c|c|}\hline \sharp & \sharp \\ \hline p_{11} & \cdots \\ \hline \end{array} \in \Theta$$

Tape

$\sharp$ p_{11}	$\sharp$ $\cdots$	$\sharp$ p_{1m}	h	$\circ$	$\cdots$	$\circ$

$$\begin{array}{|c|c|}\hline \sharp & \sharp \\ \hline \cdots & p_{1m} \\ \hline \end{array} \in \Theta$$

Tape

$\sharp$ p_{11}	$\sharp$ $\cdots$	$\sharp$ p_{1m}	$\sharp$ p_{1m+1}	$\circ$	$\cdots$	$\circ$

$$\begin{array}{|c|c|} \hline \sharp & \sharp \\ \hline p_{1m} & p_{1m+1} \\ \hline \end{array} \in \Theta$$

Tape

$\sharp$	$\sharp$	$\sharp$	$\sharp$	$\sharp$	$\cdots$	$\circ$
p_{11}	$\cdots$	p_{1m}	p_{1m+1}	p_{1m+2}		

$$\begin{array}{|c|c|}\hline \sharp & \sharp \\ \hline p_{1m+1} & p_{1m+2} \\ \hline \end{array} \in \Theta$$

Tape

$\sharp$ p_{11}	$\sharp$ $\cdots$	$\sharp$ p_{1m}	$\sharp$ p_{1m+1}	$\sharp$ p_{1m+2}	$\sharp$ $\cdots$	$\circ$

$\sharp$	$\sharp$
p_{1m+2}	$\cdots$

$\in \Theta$

Tape

$\sharp$	$\sharp$	$\sharp$	$\sharp$	$\sharp$	$\sharp$	$\sharp$
p_{11}	$\cdots$	p_{1m}	p_{1m+1}	p_{1m+2}	$\cdots$	p_{1n}

$\sharp$	$\sharp$	$\in \Theta$
$\cdots$	p_{1n}	

Tape

$\sharp$	$\sharp$	$\sharp$	$\sharp$	$\sharp$	$\sharp$	$\sharp$
p_{11}	$\cdots$	p_{1m}	p_{1m+1}	p_{1m+2}	$\cdots$	p_{1n}

$$\begin{array}{|c|c|} \hline \sharp & \sharp \\ \hline p_{1n} & \sharp \\ \hline \end{array} \in \Theta$$

Tape

$\sharp$	$\sharp$	$\sharp$	$\sharp$	$\sharp$	$\sharp$	p_{1n}
p_{11}	$\cdots$	p_{1m}	p_{1m+1}	p_{1m+2}	$\cdots$	p_{2n}

$$\begin{array}{|c|c|}\hline p_{1n} & \sharp \\ \hline p_{2n} & \sharp \\ \hline \end{array} \in \Theta$$

Tape

$\sharp$	$\sharp$	$\sharp$	$\sharp$	$\sharp$	$\cdots$	p_{1n}
p_{11}	$\cdots$	p_{1m}	p_{1m+1}	p_{1m+2}	$\cdots$	p_{2n}

$$\begin{array}{|c|c|} \hline \cdots & p_{1n} \\ \hline \cdots & p_{2n} \\ \hline \end{array} \in \Theta$$

Tape

$\sharp$	$\sharp$	$\sharp$	$\sharp$	p_{1m+2}	$\cdots$	p_{1n}
p_{11}	$\cdots$	p_{1m}	p_{1m+1}	p_{2m+2}	$\cdots$	p_{2n}

$$\begin{array}{|c|c|}\hline p_{1m+2} & \cdots \\ \hline p_{2m+2} & \cdots \\ \hline \end{array} \in \Theta$$

Tape

$\sharp$	$\sharp$	$\sharp$	p_{1m+1}	p_{1m+2}	$\cdots$	p_{1n}
p_{11}	$\cdots$	p_{1m}	p_{2m+1}	p_{2m+2}	$\cdots$	p_{2n}

$$\begin{array}{|c|c|}\hline p_{1m+1} & p_{1m+2} \\ \hline p_{2m+1} & p_{2m+2} \\ \hline \end{array} \in \Theta$$

Tape

$\sharp$ p_{11}	$\sharp$ $\cdots$	p_{1m} p_{2m}	p_{1m+1} p_{2m+1}	p_{1m+2} p_{2m+2}	$\cdots$ $\cdots$	p_{1n} p_{2n}

$$\begin{array}{|c|c|} \hline p_{1m} & p_{1m+1} \\ \hline p_{2m} & p_{2m+1} \\ \hline \end{array} \in \Theta$$

Tape

$\sharp$ p_{11}	$\cdots$ $\cdots$	p_{1m} p_{2m}	p_{1m+1} p_{2m+1}	p_{1m+2} p_{2m+2}	$\cdots$ $\cdots$	p_{1n} p_{2n}

$$\begin{array}{|c|c|} \hline \cdots & p_{1m} \\ \hline \cdots & p_{2m} \\ \hline \end{array} \in \Theta$$

Tape

p_{11} p_{21}	$\cdots$ $\cdots$	p_{1m} p_{2m}	p_{1m+1} p_{2m+1}	p_{1m+2} p_{2m+2}	$\cdots$ $\cdots$	p_{1n} p_{2n}

$$\begin{array}{|c|c|} \hline p_{11} & \cdots \\ \hline p_{21} & \cdots \\ \hline \end{array} \in \Theta$$

Tape

p_{11} p_{21}	$\cdots$ $\cdots$	p_{1m} p_{2m}	p_{1m+1} p_{2m+1}	p_{1m+2} p_{2m+2}	$\cdots$ $\cdots$	p_{1n} p_{2n}

$$\begin{array}{|c|c|} \hline \sharp & p_{11} \\ \hline \sharp & p_{21} \\ \hline \end{array} \in \Theta$$

The complexity bound implies recognizability

For any unary two-dimensional language L

$$\phi(L) \in \text{N-LinSpaceRev}_{QU} \quad \Longrightarrow \quad L \in \text{Rec}$$

The complexity bound implies recognizability

For any unary two-dimensional language L

$$\phi(L) \in \text{N-}\textsc{LinSpaceRev}_{QU} \quad \Longrightarrow \quad L \in \textsc{Rec}$$

Sketch of the proof

The complexity bound implies recognizability

For any unary two-dimensional language L

$$\phi(L) \in \text{N-LinSpaceRev}_{QU} \quad \Longrightarrow \quad L \in \text{Rec}$$

Sketch of the proof

- We introduce some two-dimensional languages
  - the accepting-computation language of the Turing machine accepting $\phi(L)$

The complexity bound implies recognizability

For any unary two-dimensional language L

$$\phi(L) \in \text{N-LinSpaceRev}_{QU} \quad \Longrightarrow \quad L \in \text{Rec}$$

Sketch of the proof

- We introduce some two-dimensional languages
  - the accepting-computation language of the Turing machine accepting $\phi(L)$
  - the mask languages for squares and for horizontal and vertical rectangles

The complexity bound implies recognizability

For any unary two-dimensional language L

$$\phi(L) \in \text{N-LinSpaceRev}_{QU} \quad \Longrightarrow \quad L \in \text{Rec}$$

Sketch of the proof

- We introduce some two-dimensional languages
  - the accepting-computation language of the Turing machine accepting $\phi(L)$
  - the mask languages for squares and for horizontal and vertical rectangles
- We overlap them to obtain a tiling-recognizable language that is projected onto L

The complexity bound implies recognizability

For any unary two-dimensional language L

$$\phi(L) \in \text{N-LinSpaceRev}_{QU} \quad \Longrightarrow \quad L \in \text{Rec}$$

Sketch of the proof

- We introduce some two-dimensional languages
  - the accepting-computation language of the Turing machine accepting $\phi(L)$
  - the mask languages for squares and for horizontal and vertical rectangles
- We overlap them to obtain a tiling-recognizable language that is projected onto L $\Longrightarrow$ L is tiling-recognizable

End of the proof

Accepting computations of a Turing machine

Given any 1-tape nondeterministic Turing machine M:

Accepting computations of a Turing machine

Given any 1-tape nondeterministic Turing machine M:

- a configuration of M is a string $C = x \, \sigma_q \, y$
  where $x, y \in \Lambda^*$ and $\sigma_q \in \Lambda_q$ (Λ is the working alphabet)

Accepting computations of a Turing machine

Given any 1-tape nondeterministic Turing machine M:

- a configuration of M is a string $C = x\ \sigma_q\ y$ where $x, y \in \Lambda^*$ and $\sigma_q \in \Lambda_q$ (Λ is the working alphabet)
- Given two configuration C and D we write $C \triangleright D$ whenever we can go from C to D without head reversals

Accepting computations of a Turing machine

Given any 1-tape nondeterministic Turing machine M:

- a configuration of M is a string $C = x\ \sigma_q\ y$ where $x, y \in \Lambda^*$ and $\sigma_q \in \Lambda_q$ (Λ is the working alphabet)
- Given two configuration C and D we write $C \triangleright D$ whenever we can go from C to D without head reversals
- An accepting computation is a sequence

$$W_1 \triangleright W_2 \triangleright \cdots \triangleright W_n$$

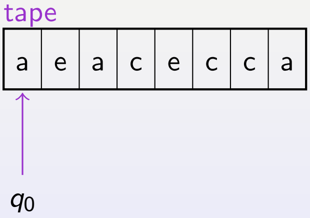
  where
  - W_1 is an initial configuration
  - W_n is an accepting configuration
  - at W_i there is a head reversal at W_i

Picture associated with an accepting-computation

Given any accepting computation $W_1 \rhd W_2 \rhd \cdots \rhd W_n$ on input w, let $m = \max |W_i|$ and consider the picture of size $n \times m$ containing W_i on the i-th row

Picture associated with an accepting-computation

Given any accepting computation $W_1 \rhd W_2 \rhd \cdots \rhd W_n$ on input w, let $m = \max |W_i|$ and consider the picture of size $n \times m$ containing W_i on the i-th row

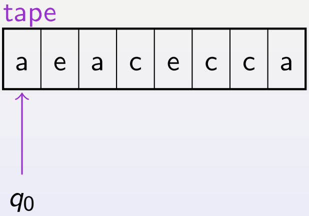


picture

a_0	e	a	c	e	c	c	a

Picture associated with an accepting-computation

Given any accepting computation $W_1 \triangleright W_2 \triangleright \cdots \triangleright W_n$ on input w, let $m = \max |W_i|$ and consider the picture of size $n \times m$ containing W_i on the i-th row



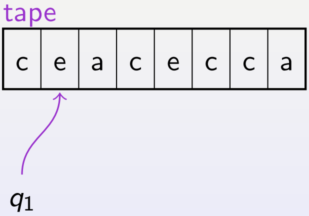
$\delta(q_0, a) \ni (q_1, c, +)$

picture

a_0	e	a	c	e	c	c	a

Picture associated with an accepting-computation

Given any accepting computation $W_1 \triangleright W_2 \triangleright \cdots \triangleright W_n$ on input w, let $m = \max |W_i|$ and consider the picture of size $n \times m$ containing W_i on the i-th row



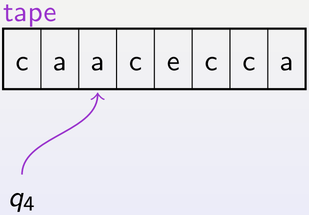
$$\delta(q_1, e) \ni (q_4, a, +)$$

picture

a_0	e	a	c	e	c	c	a

Picture associated with an accepting-computation

Given any accepting computation $W_1 \triangleright W_2 \triangleright \cdots \triangleright W_n$ on input w, let $m = \max |W_i|$ and consider the picture of size $n \times m$ containing W_i on the i-th row



$$\delta(q_4, a) \ni (q_2, c, +)$$

picture

a_0	e	a	c	e	c	c	a

Picture associated with an accepting-computation

Given any accepting computation $W_1 \rhd W_2 \rhd \cdots \rhd W_n$ on input w, let $m = \max |W_i|$ and consider the picture of size $n \times m$ containing W_i on the i-th row

tape

c	a	c	c	e	c	c	a

q_2

$$\delta(q_2, c) \ni (q_1, e, +)$$

picture

a_0	e	a	c	e	c	c	a

Picture associated with an accepting-computation

Given any accepting computation $W_1 \triangleright W_2 \triangleright \cdots \triangleright W_n$ on input w, let $m = \max |W_i|$ and consider the picture of size $n \times m$ containing W_i on the i-th row

tape

c	a	c	e	e	c	c	a

q_1

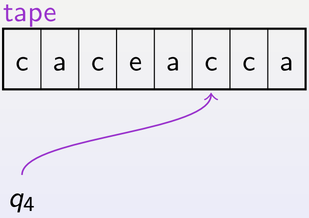
$$\delta(q_1, e) \ni (q_4, a, +)$$

picture

a_0	e	a	c	e	c	c	a

Picture associated with an accepting-computation

Given any accepting computation $W_1 \triangleright W_2 \triangleright \cdots \triangleright W_n$ on input w, let $m = \max|W_i|$ and consider the picture of size $n \times m$ containing W_i on the i-th row



$$\delta(q_4, c) \ni (q_5, e, -)$$

Head reversal!

picture

a_0	e	a	c	e	c	c	a

Picture associated with an accepting-computation

Given any accepting computation $W_1 \triangleright W_2 \triangleright \cdots \triangleright W_n$ on input w, let $m = \max |W_i|$ and consider the picture of size $n \times m$ containing W_i on the i-th row

tape

c	a	c	e	a	c	c	a

q_4

$\delta(q_4, c) \ni (q_5, e, -)$

Head reversal!

picture

a_0	e	a	c	e	c	c	a
c	a	c	e	a	c_4	c	a

Picture associated with an accepting-computation

Given any accepting computation $W_1 \triangleright W_2 \triangleright \cdots \triangleright W_n$ on input w, let $m = \max |W_i|$ and consider the picture of size $n \times m$ containing W_i on the i-th row

tape

c	a	c	e	a	e	c	a

q_5

$$\delta(q_5, a) \ni (q_1, c, -)$$

picture

a_0	e	a	c	e	c	c	a
c	a	c	e	a	c_4	c	a

Picture associated with an accepting-computation

Given any accepting computation $W_1 \triangleright W_2 \triangleright \cdots \triangleright W_n$ on input w, let $m = \max |W_i|$ and consider the picture of size $n \times m$ containing W_i on the i-th row

tape

c	a	c	e	c	e	c	a

q_1

$\delta(q_1, e) \ni (q_4, a, +)$

Head reversal!

picture

a_0	e	a	c	e	c	c	a
c	a	c	e	a	c_4	c	a

Picture associated with an accepting-computation

Given any accepting computation $W_1 \triangleright W_2 \triangleright \cdots \triangleright W_n$ on input w, let $m = \max |W_i|$ and consider the picture of size $n \times m$ containing W_i on the i-th row

tape

c	a	c	e	c	e	c	a

q_1

$\delta(q_1, e) \ni (q_4, a, +)$

Head reversal!

picture

a_0	e	a	c	e	c	c	a
c	a	c	e	a	c_4	c	a
c	a	c	e_1	c	e	c	a

Picture associated with an accepting-computation

Given any accepting computation $W_1 \rhd W_2 \rhd \cdots \rhd W_n$ on input w, let $m = \max |W_i|$ and consider the picture of size $n \times m$ containing W_i on the i-th row

tape

c	a	c	a	c	e	c	a

q_4

$\delta(q_4, c) \ni (q_0, e, +)$

picture

a_0	e	a	c	e	c	c	a
c	a	c	e	a	c_4	c	a
c	a	c	e_1	c	e	c	a

Picture associated with an accepting-computation

Given any accepting computation $W_1 \triangleright W_2 \triangleright \cdots \triangleright W_n$ on input w, let $m = \max |W_i|$ and consider the picture of size $n \times m$ containing W_i on the i-th row

tape

c	a	c	a	e	e	c	a

q_0

$$\delta(q_0, e) \ni (q_2, a, +)$$

picture

a_0	e	a	c	e	c	c	a
c	a	c	e	a	c_4	c	a
c	a	c	e_1	c	e	c	a

Picture associated with an accepting-computation

Given any accepting computation $W_1 \rhd W_2 \rhd \cdots \rhd W_n$ on input w, let $m = \max |W_i|$ and consider the picture of size $n \times m$ containing W_i on the i-th row

tape

c	a	c	a	e	a	c	a

q_2

$$\delta(q_2, c) \ni (q_3, e, +)$$

picture

a_0	e	a	c	e	c	c	a
c	a	c	e	a	c_4	c	a
c	a	c	e_1	c	e	c	a

Picture associated with an accepting-computation

Given any accepting computation $W_1 \triangleright W_2 \triangleright \cdots \triangleright W_n$ on input w, let $m = \max |W_i|$ and consider the picture of size $n \times m$ containing W_i on the i-th row

tape

c	a	c	a	e	a	e	a

q_3

$\delta(q_3, a) \ni (q_2, e, -)$

Head reversal!

picture

a_0	e	a	c	e	c	c	a
c	a	c	e	a	c_4	c	a
c	a	c	e_1	c	e	c	a

Picture associated with an accepting-computation

Given any accepting computation $W_1 \triangleright W_2 \triangleright \cdots \triangleright W_n$ on input w, let $m = \max |W_i|$ and consider the picture of size $n \times m$ containing W_i on the i-th row

tape

c	a	c	a	e	a	e	a

q_3

$\delta(q_3, a) \ni (q_2, e, -)$

Head reversal!

picture

a_0	e	a	c	e	c	c	a
c	a	c	e	a	c_4	c	a
c	a	c	e_1	c	e	c	a
c	a	c	a	e	a	e	a_3

Picture associated with an accepting-computation

Given any accepting computation $W_1 \triangleright W_2 \triangleright \cdots \triangleright W_n$ on input w, let $m = \max |W_i|$ and consider the picture of size $n \times m$ containing W_i on the i-th row

tape

c	a	c	a	e	a	e	e

q_2

$$\delta(q_2, e) \ni (q_4, a, -)$$

picture

a_0	e	a	c	e	c	c	a
c	a	c	e	a	c_4	c	a
c	a	c	e_1	c	e	c	a
c	a	c	a	e	a	e	a_3

Picture associated with an accepting-computation

Given any accepting computation $W_1 \triangleright W_2 \triangleright \cdots \triangleright W_n$ on input w, let $m = \max |W_i|$ and consider the picture of size $n \times m$ containing W_i on the i-th row

tape

c	a	c	a	e	a	a	e

q_4

$\delta(q_4, a) \ni (q_2, c, +)$

Head reversal!

picture

a_0	e	a	c	e	c	c	a
c	a	c	e	a	c_4	c	a
c	a	c	e_1	c	e	c	a
c	a	c	a	e	a	e	a_3

Picture associated with an accepting-computation

Given any accepting computation $W_1 \triangleright W_2 \triangleright \cdots \triangleright W_n$ on input w, let $m = \max |W_i|$ and consider the picture of size $n \times m$ containing W_i on the i-th row

tape

c	a	c	a	e	a	a	e

q_4

$\delta(q_4, a) \ni (q_2, c, +)$

Head reversal!

picture

a_0	e	a	c	e	c	c	a
c	a	c	e	a	c_4	c	a
c	a	c	e_1	c	e	c	a
c	a	c	a	e	a	e	a_3
c	a	c	a	e	a_4	a	e

Picture associated with an accepting-computation

Given any accepting computation $W_1 \triangleright W_2 \triangleright \cdots \triangleright W_n$ on input w, let $m = \max |W_i|$ and consider the picture of size $n \times m$ containing W_i on the i-th row

tape

c	a	c	a	e	c	a	e

q_2

$\delta(q_2, a) \ni (q_0, e, -)$

Head reversal!

picture

a_0	e	a	c	e	c	c	a
c	a	c	e	a	c_4	c	a
c	a	c	e_1	c	e	c	a
c	a	c	a	e	a	e	a_3
c	a	c	a	e	a_4	a	e

Picture associated with an accepting-computation

Given any accepting computation $W_1 \rhd W_2 \rhd \cdots \rhd W_n$ on input w, let $m = \max |W_i|$ and consider the picture of size $n \times m$ containing W_i on the i-th row

tape

c	a	c	a	e	c	a	e

q_2

$\delta(q_2, a) \ni (q_0, e, -)$

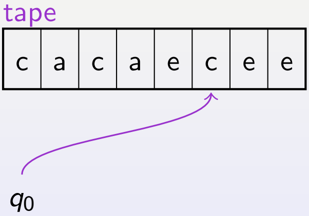
Head reversal!

picture

a_0	e	a	c	e	c	c	a
c	a	c	e	a	c_4	c	a
c	a	c	e_1	c	e	c	a
c	a	c	a	e	a	e	a_3
c	a	c	a	e	a_4	a	e
c	a	c	a	e	c	a_2	e

Picture associated with an accepting-computation

Given any accepting computation $W_1 \rhd W_2 \rhd \cdots \rhd W_n$ on input w, let $m = \max |W_i|$ and consider the picture of size $n \times m$ containing W_i on the i-th row



$$\delta(q_0, c) \ni (q_y, e, -)$$

picture

a_0	e	a	c	e	c	c	a
c	a	c	e	a	c_4	c	a
c	a	c	e_1	c	e	c	a
c	a	c	a	e	a	e	a_3
c	a	c	a	e	a_4	a	e
c	a	c	a	e	c	a_2	e

Picture associated with an accepting-computation

Given any accepting computation $W_1 \triangleright W_2 \triangleright \cdots \triangleright W_n$ on input w, let $m = \max |W_i|$ and consider the picture of size $n \times m$ containing W_i on the i-th row

tape

c	a	c	a	e	e	e	e

q_y

picture

a_0	e	a	c	e	c	c	a
c	a	c	e	a	c_4	c	a
c	a	c	e_1	c	e	c	a
c	a	c	a	e	a	e	a_3
c	a	c	a	e	a_4	a	e
c	a	c	a	e	c	a_2	e

Picture associated with an accepting-computation

Given any accepting computation $W_1 \rhd W_2 \rhd \cdots \rhd W_n$ on input w, let $m = \max |W_i|$ and consider the picture of size $n \times m$ containing W_i on the i-th row

tape

c	a	c	a	e	e	e	e

q_y

picture

a_0	e	a	c	e	c	c	a
c	a	c	e	a	c_4	c	a
c	a	c	e_1	c	e	c	a
c	a	c	a	e	a	e	a_3
c	a	c	a	e	a_4	a	e
c	a	c	a	e	c	a_2	e
c	a	c	a	e_y	e	e	e

Picture associated with an accepting-computation

Given any accepting computation $W_1 \triangleright W_2 \triangleright \cdots \triangleright W_n$ on input w, let $m = \max |W_i|$ and consider the picture of size $n \times m$ containing W_i on the i-th row

picture

a_0	e	a	c	e	c	c	a
c	a	c	e	a	c_4	c	a
c	a	c	e_1	c	e	c	a
c	a	c	a	e	a	e	a_3
c	a	c	a	e	a_4	a	e
c	a	c	a	e	c	a_2	e
c	a	c	a	e_y	e	e	e

Accepting-computation language

The accepting-computation language of M is defined as the set $A(M)$ of all pictures corresponding to any accepting computation of M.

Accepting-computation language

The accepting-computation language of M is defined as the set $A(M)$ of all pictures corresponding to any accepting computation of M.

Proposition

The accepting-computation language of a Turing machine is in $\textsc{Rec}$

Accepting-computation language

The accepting-computation language of M is defined as the set $A(M)$ of all pictures corresponding to any accepting computation of M.

Proposition

The accepting-computation language of a Turing machine is in $\textsc{Rec}$

a_0	e	a	c	e	c	c	a
c	a	c	e	a	c_4	c	a
c	a	c	e_1	c	e	c	a
c	a	c	a	e	a	e	a_3
c	a	c	a	e	a_4	a	e
c	a	c	a	e	c	a_2	e
c	a	c	a	e_y	e	e	e

Accepting-computation language

The accepting-computation language of M is defined as the set $A(M)$ of all pictures corresponding to any accepting computation of M.

$\overrightarrow{a_0}$	e	a	c	e	c	c	a
c	a	c	e	a	$\overleftarrow{c_4}$	c	a
c	a	c	$\overrightarrow{e_1}$	c	e	c	a
c	a	c	a	e	a	e	$\overleftarrow{a_3}$
c	a	c	a	e	$\overrightarrow{a_4}$	a	e
c	a	c	a	e	c	$\overleftarrow{a_2}$	e
c	a	c	a	$\overrightarrow{e_y}$	e	e	e

Proposition

The accepting-computation language of a Turing machine is in REC

Accepting-computation language

The accepting-computation language of M is defined as the set $A(M)$ of all pictures corresponding to any accepting computation of M.

Proposition

The accepting-computation language of a Turing machine is in $\textsc{Rec}$

$\overrightarrow{a_0}$	e	a	c	e	c	c	a
c	a	c	e	a	$\overleftarrow{c_4}$	c	a
c	a	c	$\overrightarrow{e_1}$	c	e	c	a
c	a	c	a	e	a	a	$\overleftarrow{a_3}$
c	a	c	a	e	$\overrightarrow{a_4}$	a	e
c	a	c	a	e	c	$\overleftarrow{a_2}$	e
c	a	c	a	$\overrightarrow{e_y}$	e	e	e

The accepting-computation language is in $\textsc{Rec}$

tape

a	e	a	c	e	c	c	a

q_0

$$\delta(q_0, a) \ni (q_1, c, +)$$

$\overrightarrow{a_0}$	e	a	c	e	c	c	a
c	a	c	e	a	$\overleftarrow{c_4}$	c	a
c	a	c	$\overrightarrow{e_1}$	c	e	c	a
c	a	c	a	e	a	a	$\overleftarrow{a_3}$
c	a	c	a	e	$\overrightarrow{a_4}$	a	e
c	a	c	a	e	c	$\overleftarrow{a_2}$	e
c	a	c	a	$\overrightarrow{e_y}$	e	e	e

The accepting-computation language is in $\textsc{Rec}$

tape

c	e	a	c	e	c	c	a

q_1

$$\delta(q_1, e) \ni (q_4, a, +)$$

$\overrightarrow{a_0}$	e	a	c	e	c	c	a
c	${}^1a$	c	e	a	$\overleftarrow{c_4}$	c	a
c	a	c	$\overrightarrow{e_1}$	c	e	c	a
c	a	c	a	e	a	a	$\overleftarrow{a_3}$
c	a	c	a	e	$\overrightarrow{a_4}$	a	e
c	a	c	a	e	c	$\overleftarrow{a_2}$	e
c	a	c	a	$\overrightarrow{e_y}$	e	e	e

The accepting-computation language is in REC

tape

c	a	a	c	e	c	c	a

q_4

$$\delta(q_4, a) \ni (q_2, c, +)$$

$\overrightarrow{a_0}$	e	a	c	e	c	c	a
c	${}^1a$	${}^4c$	e	a	$\overleftarrow{c_4}$	c	a
c	a	c	$\overrightarrow{e_1}$	c	e	c	a
c	a	c	a	e	a	a	$\overleftarrow{a_3}$
c	a	c	a	e	$\overrightarrow{a_4}$	a	e
c	a	c	a	e	c	$\overleftarrow{a_2}$	e
c	a	c	a	$\overrightarrow{e_y}$	e	e	e

The accepting-computation language is in REC

tape

c	a	c	c	e	c	c	a

q_2

$$\delta(q_2, c) \ni (q_1, e, +)$$

$\overrightarrow{a_0}$	e	a	c	e	c	c	a
c	${}^1a$	${}^4c$	${}^2e$	a	$\overleftarrow{c_4}$	c	a
c	a	c	$\overrightarrow{e_1}$	c	e	c	a
c	a	c	a	e	a	a	$\overleftarrow{a_3}$
c	a	c	a	e	$\overrightarrow{a_4}$	a	e
c	a	c	a	e	c	$\overleftarrow{a_2}$	e
c	a	c	a	$\overrightarrow{e_y}$	e	e	e

The accepting-computation language is in $\textsc{Rec}$

tape

c	a	c	e	e	c	c	a

q_1

$\delta(q_1, e) \ni (q_4, a, +)$

$\overrightarrow{a_0}$	e	a	c	e	c	c	a
c	^1a	^4c	^2e	^1a	$\overleftarrow{c_4}$	c	a
c	a	c	$\overrightarrow{e_1}$	c	e	c	a
c	a	c	a	e	a	a	$\overleftarrow{a_3}$
c	a	c	a	e	$\overrightarrow{a_4}$	a	e
c	a	c	a	e	c	$\overleftarrow{a_2}$	e
c	a	c	a	$\overrightarrow{e_y}$	e	e	e

The accepting-computation language is in $\textsc{Rec}$

tape

c	a	c	e	a	c	c	a

q_4

$$\delta(q_4, c) \ni (q_5, e, -)$$

$\overrightarrow{a_0}$	e	a	c	e	c	c	a
c	${}^1a$	${}^4c$	${}^2e$	${}^1a$	$\overleftarrow{c_4}$	c	a
c	a	c	$\overrightarrow{e_1}$	c	e	c	a
c	a	c	a	e	a	a	$\overleftarrow{a_3}$
c	a	c	a	e	$\overrightarrow{a_4}$	a	e
c	a	c	a	e	c	$\overleftarrow{a_2}$	e
c	a	c	a	$\overrightarrow{e_y}$	e	e	e

The accepting-computation language is in $\textsc{Rec}$

$\overrightarrow{a_0}$	e	a	c	e	c	c	a
c	${}^1a$	${}^4c$	${}^2e$	${}^1a$	$\overleftarrow{c_4}$	c	a
c	a	c	$\overrightarrow{e_1}$	${}^5c$	e	c	a
c	a	c	a	${}^4e$	${}^0a$	${}^2e$	$\overleftarrow{a_3}$
c	a	c	a	e	$\overrightarrow{a_4}$	${}^2a$	e
c	a	c	a	e	c	$\overleftarrow{a_2}$	e
c	a	c	a	$\overrightarrow{e_y}$	${}^0e$	e	e

The accepting-computation language is in REC

The marked picture
can be described locally!

$\overrightarrow{a_0}$	e	a	c	e	c	c	a
c	${}^1a$	${}^4c$	${}^2e$	${}^1a$	$\overleftarrow{c_4}$	c	a
c	a	c	$\overrightarrow{e_1}$	${}^5c$	e	c	a
c	a	c	a	${}^4e$	${}^0a$	${}^2e$	$\overleftarrow{a_3}$
c	a	c	a	e	$\overrightarrow{a_4}$	${}^2a$	e
c	a	c	a	e	c	$\overleftarrow{a_2}$	e
c	a	c	a	$\overrightarrow{e_y}$	${}^0e$	e	e

The accepting-computation language is in $\textsc{Rec}$

The marked picture
can be described locally!

Hence the accepting-computation
language is in $\textsc{Rec}$.

$\overrightarrow{a_0}$	e	a	c	e	c	c	a
c	${}^1a$	${}^4c$	${}^2e$	${}^1a$	$\overleftarrow{c_4}$	c	a
c	a	c	$\overrightarrow{e_1}$	${}^5c$	e	c	a
c	a	c	a	${}^4e$	${}^0a$	${}^2e$	$\overleftarrow{a_3}$
c	a	c	a	e	$\overrightarrow{a_4}$	${}^2a$	e
c	a	c	a	e	c	$\overleftarrow{a_2}$	e
c	a	c	a	$\overrightarrow{e_y}$	${}^0e$	e	e

Overlap of languages

$$p = \begin{array}{|c|c|c|} \hline p_{11} & \cdots & p_{1m} \\ \hline p_{21} & \cdots & p_{2m} \\ \hline \vdots & \cdots & \vdots \\ \hline p_{n1} & \cdots & p_{nm} \\ \hline \end{array}$$

$$q = \begin{array}{|c|c|c|} \hline q_{11} & \cdots & q_{1m} \\ \hline q_{21} & \cdots & q_{2m} \\ \hline \vdots & \cdots & \vdots \\ \hline q_{n1} & \cdots & q_{nm} \\ \hline \end{array}$$

Overlap of languages

Product $p \times q$

$$p = \begin{array}{|c|c|c|}\hline p_{11} & \cdots & p_{1m} \\ \hline p_{21} & \cdots & p_{2m} \\ \hline \vdots & \cdots & \vdots \\ \hline p_{n1} & \cdots & p_{nm} \\ \hline \end{array}$$

$$q = \begin{array}{|c|c|c|}\hline q_{11} & \cdots & q_{1m} \\ \hline q_{21} & \cdots & q_{2m} \\ \hline \vdots & \cdots & \vdots \\ \hline q_{n1} & \cdots & q_{nm} \\ \hline \end{array}$$

$$\begin{array}{|c|c|c|}\hline (p_{11}, q_{11}) & \cdots & (p_{1m}, q_{1m}) \\ \hline (p_{21}, q_{21}) & \cdots & (p_{2m}, q_{2m}) \\ \hline \vdots & \cdots & \vdots \\ \hline (p_{n1}, q_{n1}) & \cdots & (p_{nm}, q_{nm}) \\ \hline \end{array}$$

Overlap of languages

Product $p \times q$

$$p = \begin{array}{|c|c|c|} \hline p_{11} & \cdots & p_{1m} \\ \hline p_{21} & \cdots & p_{2m} \\ \hline \vdots & \cdots & \vdots \\ \hline p_{n1} & \cdots & p_{nm} \\ \hline \end{array}$$

$$q = \begin{array}{|c|c|c|} \hline q_{11} & \cdots & q_{1m} \\ \hline q_{21} & \cdots & q_{2m} \\ \hline \vdots & \cdots & \vdots \\ \hline q_{n1} & \cdots & q_{nm} \\ \hline \end{array}$$

$$\begin{array}{|c|c|c|} \hline (p_{11}, q_{11}) & \cdots & (p_{1m}, q_{1m}) \\ \hline (p_{21}, q_{21}) & \cdots & (p_{2m}, q_{2m}) \\ \hline \vdots & \cdots & \vdots \\ \hline (p_{n1}, q_{n1}) & \cdots & (p_{nm}, q_{nm}) \\ \hline \end{array}$$

$L_1 \diamond L_2$ is defined as the set of pictures $p_1 \times p_2$, $p_i \in L_i$, such that

- p_1 and p_2 have the same size
- the first row of p_1 equals the first row of p_2

$\phi(L) \in \text{N-LinSpaceRev}_{QU} \Longrightarrow L \in \text{Rec}$

- Assume $\phi(L) \in \text{N-LinSpaceRev}_{QU}$

$\phi(L) \in \textsc{N-LinSpaceRev}_{QU} \Longrightarrow L \in \textsc{Rec}$

- Assume $\phi(L) \in \textsc{N-LinSpaceRev}_{QU}$
- Let A be the accepting-computation language associated with the Turing machine that accept $\phi(L)$

$\phi(L) \in \text{N-LinSpaceRev}_{QU} \Longrightarrow L \in \text{Rec}$

- Assume $\phi(L) \in \text{N-LinSpaceRev}_{QU}$
- Let A be the accepting-computation language associated with the Turing machine that accept $\phi(L)$
- Add paddings to tune the sizes

$\phi(L) \in \textsc{N-LinSpaceRev}_{QU} \Longrightarrow L \in \textsc{Rec}$

- Assume $\phi(L) \in \textsc{N-LinSpaceRev}_{QU}$
- Let A be the accepting-computation language associated with the Turing machine that accept $\phi(L)$
- Add paddings to tune the sizes

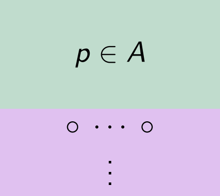
$$p \in A$$

$\phi(L) \in \text{N-LinSpaceRev}_{QU} \Longrightarrow L \in \text{Rec}$

- Assume $\phi(L) \in \text{N-LinSpaceRev}_{QU}$
- Let A be the accepting-computation language associated with the Turing machine that accept $\phi(L)$
- Add paddings to tune the sizes

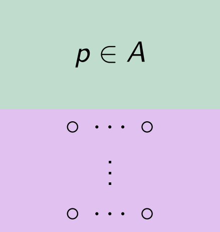
$\phi(L) \in \text{N-LINSPACEREV}_{QU} \Longrightarrow L \in \text{REC}$

- Assume $\phi(L) \in \text{N-LINSPACEREV}_{QU}$
- Let A be the accepting-computation language associated with the Turing machine that accept $\phi(L)$
- Add paddings to tune the sizes



$\phi(L) \in \text{N-LinSpaceRev}_{QU} \Longrightarrow L \in \text{Rec}$

- Assume $\phi(L) \in \text{N-LinSpaceRev}_{QU}$
- Let A be the accepting-computation language associated with the Turing machine that accept $\phi(L)$
- Add paddings to tune the sizes, obtaining $A' = A \ominus \circ^{**}$



$$A' = A \ominus \circ^{**}$$

$\phi(L) \in \text{N-LinSpaceRev}_{QU} \Longrightarrow L \in \text{Rec}$

- Assume $\phi(L) \in \text{N-LinSpaceRev}_{QU}$
- Let A be the accepting-computation language associated with the Turing machine that accept $\phi(L)$
- Add paddings to tune the sizes, obtaining $A' = A \ominus \circ^{**}$
- Consider the mask languages

$\phi(L) \in \textsc{N-LinSpaceRev}_{QU} \Longrightarrow L \in \textsc{Rec}$

- Assume $\phi(L) \in \textsc{N-LinSpaceRev}_{QU}$
- Let A be the accepting-computation language associated with the Turing machine that accept $\phi(L)$
- Add paddings to tune the sizes, obtaining $A' = A \ominus \circ^{**}$
- Consider the mask languages M_s

$$\begin{matrix} \circ & \cdots & \circ \\ & \vdots & \\ \circ & \cdots & \circ \end{matrix}$$

$M_s = \{\text{Unary squares}\}$

$\phi(L) \in \text{N-LinSpaceRev}_{QU} \Longrightarrow L \in \text{Rec}$

- Assume $\phi(L) \in \text{N-LinSpaceRev}_{QU}$
- Let A be the accepting-computation language associated with the Turing machine that accept $\phi(L)$
- Add paddings to tune the sizes, obtaining $A' = A \ominus \circ^{**}$
- Consider the mask languages M_s, M_h

$$\begin{array}{c|c|c} \circ \cdots \circ & h & \circ \cdots \cdots \circ \\ \vdots & h & \vdots \\ \circ \cdots \circ & h & \circ \cdots \cdots \circ \end{array}$$

$$M_h = M_s \varobar h^{*\ominus} \varobar \circ^{**}$$

$\phi(L) \in \text{N-LinSpaceRev}_{QU} \Longrightarrow L \in \text{Rec}$

- Assume $\phi(L) \in \text{N-LinSpaceRev}_{QU}$
- Let A be the accepting-computation language associated with the Turing machine that accept $\phi(L)$
- Add paddings to tune the sizes, obtaining $A' = A \ominus \circ^{**}$
- Consider the mask languages M_s, M_h and M_v

$$\begin{array}{c|c|c} \circ \cdots \circ & v & \circ \cdots \cdots \circ \\ \vdots & v & \vdots \\ \circ \cdots \circ & v & \circ \cdots \cdots \circ \end{array}$$

$$M_v = M_s \obar v^{*\ominus} \obar \circ^{**}$$

$\phi(L) \in \text{N-LinSpaceRev}_{QU} \Longrightarrow L \in \text{Rec}$

- Assume $\phi(L) \in \text{N-LinSpaceRev}_{QU}$
- Let A be the accepting-computation language associated with the Turing machine that accept $\phi(L)$
- Add paddings to tune the sizes, obtaining $A' = A \ominus \circ^{**}$
- Consider the mask languages M_s, M_h and M_v

$$\text{Set } L' = (A' \diamond M_s) \cup (A' \diamond M_h) \cup (A' \diamond M_v)^R$$

$\phi(L) \in \text{N-LinSpaceRev}_{QU} \Longrightarrow L \in \text{Rec}$

- Assume $\phi(L) \in \text{N-LinSpaceRev}_{QU}$
- Let A be the accepting-computation language associated with the Turing machine that accept $\phi(L)$
- Add paddings to tune the sizes, obtaining $A' = A \ominus \circ^{**}$
- Consider the mask languages M_s, M_h and M_v

$$\text{Set } L' = (A' \diamond M_s) \cup (A' \diamond M_h) \cup (A' \diamond M_v)^R$$

Then $\pi(L') = L$, where π is the morphism that maps all symbols onto $\circ$.

$\phi(L) \in \text{N-LinSpaceRev}_{QU} \Longrightarrow L \in \text{Rec}$

- Assume $\phi(L) \in \text{N-LinSpaceRev}_{QU}$
- Let A be the accepting-computation language associated with the Turing machine that accept $\phi(L)$
- Add paddings to tune the sizes, obtaining $A' = A \ominus \circ^{**}$
- Consider the mask languages M_s, M_h and M_v

$$\text{Set } L' = (A' \diamond M_s) \cup (A' \diamond M_h) \cup (A' \diamond M_v)^R$$

Then $\pi(L') = L$, where π is the morphism that maps all symbols onto $\circ$.

$$\Longrightarrow \; L \text{ is in } \text{Rec}$$

Summary

- Two dimensional tiling-recognizable pictures

Summary

- Two dimensional tiling-recognizable pictures
- Representation of two-dimensional unary pictures by quasi-unary strings

Summary

- Two dimensional tiling-recognizable pictures
- Representation of two-dimensional unary pictures by quasi-unary strings
- Complexity class for quasi-unary languages: space and number of head reversals bounded.

Summary

- Two dimensional tiling-recognizable pictures
- Representation of two-dimensional unary pictures by quasi-unary strings
- Complexity class for quasi-unary languages: space and number of head reversals bounded.
- Characterization of two-dimensional unary languages in terms of complexity of the corresponding quasi-unary languages