# Advanced Automata Theory 7 Automatic Functions

**Frank Stephan**

**Department of Computer Science**

**Department of Mathematics**

**National University of Singapore**

**fstephan@comp.nus.edu.sg**

# Repetition 1

Automaton $(\mathbf{Q}, \mathbf{\Sigma}, \delta, \mathbf{s}, \mathbf{F})$ with $\mathbf{Q}, \mathbf{\Sigma}, \delta, \mathbf{s}, \mathbf{F}$ being defined as a usual non-deterministic automaton but a different semantic for dealing with infinite words.

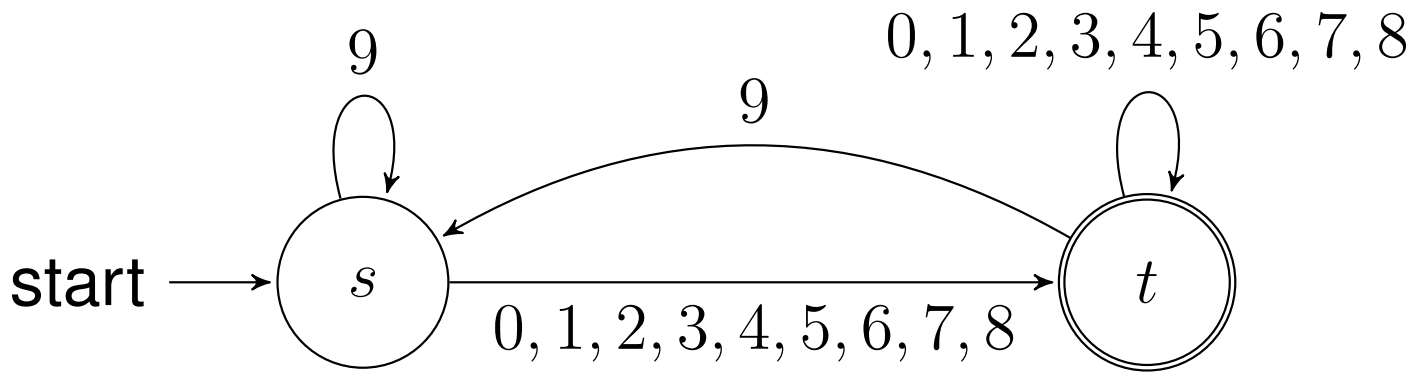
Given an infinite word $\mathbf{b_0 b_1 b_2} \ldots \in \mathbf{\Sigma}^\omega$, a run is a sequence $\mathbf{q_0 q_1 q_2} \ldots \in \mathbf{Q}^\omega$ of states such that $\mathbf{q_0} = \mathbf{s}$ and $(\mathbf{q_k}, \mathbf{b_k}, \mathbf{q_{k+1}}) \in \delta$ for all $\mathbf{k}$. Let

$$\mathbf{U} = \{\mathbf{p} \in \mathbf{Q} : \exists^\infty \mathbf{k} \, [\mathbf{q_k} = \mathbf{p}]\}$$

be the set of infinitely often visited states on this run. The run is accepting iff $\mathbf{U} \cap \mathbf{F} \neq \emptyset$. The Büchi automaton accepts an $\omega$-word iff it has an accepting run on this $\omega$-word, otherwise it rejects the $\omega$-word.

# Repetition 2

The following deterministic Büchi automaton accepts all $\omega$-words of reals between $\mathbf{0}$ and $\mathbf{1}$ which are not almost always $\mathbf{9}$.




Here a Büchi automaton is called deterministic iff for every $\mathbf{p} \in \mathbf{Q}$ and $\mathbf{a} \in \mathbf{\Sigma}$ there is at most one $\mathbf{q} \in \mathbf{Q}$ with $(\mathbf{p}, \mathbf{a}, \mathbf{q}) \in \delta$; in this case one also writes $\delta(\mathbf{p}, \mathbf{a}) = \mathbf{q}$.

This automaton goes infinitely often through the accepting state $\mathbf{t}$ iff there is infinitely often one of the digits $\mathbf{0}, \mathbf{1}, \mathbf{2}, \mathbf{3}, \mathbf{4}, \mathbf{5}, \mathbf{6}, \mathbf{7}, \mathbf{8}$ and therefore the word is not of the form $\mathbf{w9}^{\omega}$.

# Repetition 3

Theorem [Büchi 1960]
The following are equvivalent for a language $\mathbf{L}$ of $\omega$-words:
**(a)** $\mathbf{L}$ is recognised by a non-deterministic Büchi automaton;
**(b)** $\mathbf{L} = \cup_{\mathbf{m} \in \{1,\ldots,n\}} \mathbf{A_m B_m^\omega}$ for some $\mathbf{n}$ and $\mathbf{2n}$ regular languages $\mathbf{A_1}, \mathbf{B_1}, \ldots, \mathbf{A_n}, \mathbf{B_n}$.

Here $\mathbf{B_m^\omega}$ is the concatenation of infinitely many non-empty strings from $\mathbf{B_m}$.

Theorem [Büchi 1960]
There are $\omega$-languages which are only recognised by a non-deterministic Büchi automaton but not by a deterministic one; for example, $\{\mathbf{0},\mathbf{1}\}^* \cdot \{\mathbf{0}\}^\omega$.

# Repetition 4

Theorem [McNaughton 1966, Safra 1988]
The following conditions are equivalent for an $\omega$-language $\mathbf{L}$.
**(a)** $\mathbf{L}$ is recognised by a non-deterministic Büchi automaton;
**(b)** $\mathbf{L}$ is recognised by a deterministic Muller automaton;
**(c)** $\mathbf{L}$ is recognised by a non-deterministic Muller automaton.

Application
If a language $\mathbf{L}$ is recognised by a non-deterministic Büchi automaton, so is its complement. There is an algorithm which constructs from a Büchi automaton for $\mathbf{L}$ a Büchi automaton for its complement. The number of states grows exponentially.

# Automatic Relations and Functions

## Lexicographical Order

$\mathbf{a_1 a_2 \ldots a_n <_{lex} b_1 b_2 \ldots b_m}$ iff either the first string is a proper prefix of the second or there is a $\mathbf{k}$ with $\mathbf{k \leq n \wedge k \leq m \wedge a_k < b_k \wedge a_h = b_h}$ for all $\mathbf{h}$ with $\mathbf{1 \leq h < k}$.

$\mathbf{NUH <_{lex} NUHS <_{lex} NUS <_{lex} SOC}$.

## Algorithm

Processing inputs $\mathbf{x, y}$ symbol by symbol.

1. If $\mathbf{x}$ is exhausted and $\mathbf{y}$ not, then $\mathbf{x <_{lex} y}$, Halt.
2. If $\mathbf{y}$ is exhausted and $\mathbf{x}$ not, then $\mathbf{y <_{lex} x}$, Halt.
3. If $\mathbf{x}$ and $\mathbf{y}$ are exhausted, then $\mathbf{x = y}$, Halt.
4. Read symbol $\mathbf{a}$ from $\mathbf{x}$ and $\mathbf{b}$ from $\mathbf{y}$.
5. If $\mathbf{a = b}$ then go to 1.
6. If $\mathbf{a < b}$ then $\mathbf{x <_{lex} y}$ else $\mathbf{y <_{lex} x}$. Halt.

# More Formally

Use special symbol $\#$ to be returned from exhausted inputs.

Relation $\mathbf{R} \subseteq \mathbf{X} \times \mathbf{Y}$ is automatic iff there is an automaton reading both inputs at the same speed (one symbol per cycle) such that $(\mathbf{x}, \mathbf{y}) \in \mathbf{R}$ iff the automaton is in an accepting state after having read both, $\mathbf{x}$ and $\mathbf{y}$, completely.

Similarly one can define that a relation of several parameters is automatic.

A function $\mathbf{f} : \mathbf{X} \to \mathbf{Y}$ is automatic iff the relation $\{(\mathbf{x}, \mathbf{y}) : \mathbf{x} \in \mathbf{dom}(\mathbf{f}) \wedge \mathbf{y} = \mathbf{f}(\mathbf{x})\}$ is automatic.

# Convolution

Relations can be translated into singular sets using convolution.

Convolution has combined characters of matching positions in the words with $\#$ used for exhausted words ($\#$ is not in the alphabet).

$\mathbf{conv}(\mathbf{00110}, \mathbf{0123456789}) =$
$\binom{\mathbf{0}}{\mathbf{0}}\binom{\mathbf{0}}{\mathbf{1}}\binom{\mathbf{1}}{\mathbf{2}}\binom{\mathbf{1}}{\mathbf{3}}\binom{\mathbf{0}}{\mathbf{4}}\binom{\#}{\mathbf{5}}\binom{\#}{\mathbf{6}}\binom{\#}{\mathbf{7}}\binom{\#}{\mathbf{8}}\binom{\#}{\mathbf{9}}$.

Now one can formalise automaticity of a relation using a convolution.

For example, a ternary relation $\mathbf{R}$ of words over $\mathbf{\Sigma}$ is automatic iff the set $\{\mathbf{conv}(\mathbf{x}, \mathbf{y}, \mathbf{z}) : (\mathbf{x}, \mathbf{y}, \mathbf{z}) \in \mathbf{R}\}$ is regular.

# Example

Let $\Sigma = \{0, 1\}$.

Then $\{\mathbf{conv}(x, y) : x, y \in \Sigma^* \wedge |x| = |y|\}$ is the set $\{\binom{0}{0}, \binom{0}{1}, \binom{1}{0}, \binom{1}{1}\}^*$ or, more general, $(\Sigma \times \Sigma)^*$.

Quiz

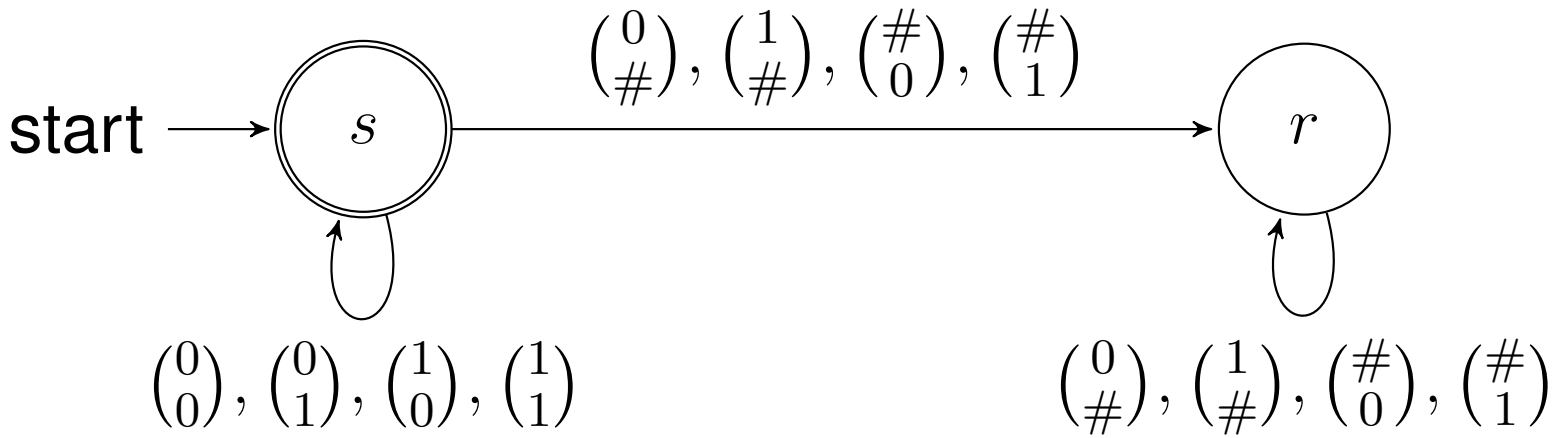
Which relations are coded by the sets

$$\{\binom{0}{0}, \binom{0}{1}, \binom{1}{0}, \binom{1}{1}\}^* \cdot \{\binom{\#}{0}, \binom{\#}{1}\}^*$$

and

$$\{\binom{0}{0}, \binom{0}{1}, \binom{1}{0}, \binom{1}{1}\}^* \cdot \{\binom{\#}{0}, \binom{\#}{1}\} \cdot \{\binom{\#}{0}, \binom{\#}{1}\}^* \quad ?$$

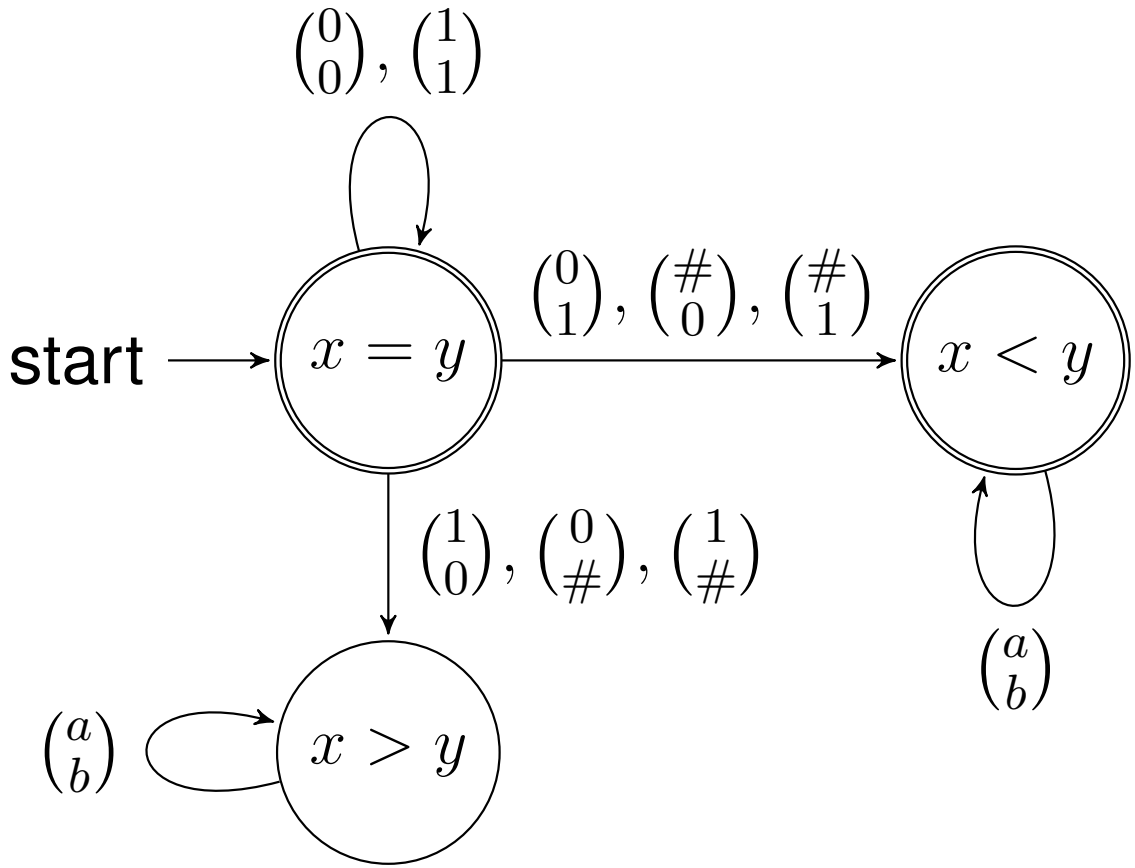
# Automaton comparing string length




Automaton processes strings correctly when it is the convolution of two binary strings.

# Lexicographic Order as AR

For binary alphabet $\{0, 1\}$, the following automaton recognises lexicographic ordering.




Here $\binom{a}{b}$ on an arrow means that the automaton always goes this way.

# Automatic Functions

An automatic relation $\mathbf{R} \subseteq \mathbf{X} \times \mathbf{Y}$ defines a function $\mathbf{f}$ iff $\forall \mathbf{x} \forall \mathbf{y}\, [\mathbf{f}(\mathbf{x}) = \mathbf{y} \Leftrightarrow (\mathbf{x}, \mathbf{y}) \in \mathbf{R}]$.

For example, $\mathbf{f}(\mathbf{x}) = \mathbf{x01}$ has the graph $\{\binom{\mathbf{0}}{\mathbf{0}}, \binom{\mathbf{1}}{\mathbf{1}}\}^* \cdot \binom{\#}{\mathbf{0}}\binom{\#}{\mathbf{1}}$.

Quiz. Consider the following expressions:

- $\{\binom{\mathbf{0}}{\mathbf{0}}, \binom{\mathbf{0}}{\mathbf{1}}, \binom{\mathbf{1}}{\mathbf{0}}, \binom{\mathbf{1}}{\mathbf{1}}\}^* \cdot \{\binom{\mathbf{0}}{\mathbf{0}}, \binom{\mathbf{1}}{\mathbf{1}}\} \cdot (\{\binom{\#}{\mathbf{0}}, \binom{\#}{\mathbf{1}}\}^* \cup \{\binom{\mathbf{0}}{\#}, \binom{\mathbf{1}}{\#}\}^*)$,
- $\{\binom{\mathbf{0}}{\mathbf{1}}, \binom{\mathbf{1}}{\mathbf{0}}\}^* \cdot (\{\varepsilon\} \cup \{\binom{\mathbf{2}}{\#}\} \cdot \{\binom{\mathbf{0}}{\#}, \binom{\mathbf{1}}{\#}, \binom{\mathbf{2}}{\#}\}^*)$,
- $\{\binom{\mathbf{0}}{\mathbf{0}}, \binom{\mathbf{1}}{\mathbf{0}}, \binom{\mathbf{2}}{\mathbf{0}}, \ldots, \binom{\mathbf{9}}{\mathbf{0}}\}^*$.

Which of these three relations define functions? What are the domains and ranges of these functions?

# Exercise 7.6

Which of the following relations are automatic (where $\mathbf{x_k}$ is the $\mathbf{k}$-th symbol of $\mathbf{x = x_1 x_2 \ldots x_n}$ and $\mathbf{|x| = n}$):

- $\mathbf{R_1(x, y, z) \Leftrightarrow \forall k \in \{1, 2, \ldots, \min\{|x|, |y|, |z|\}\}\,[x_k = y_k \vee x_k = z_k \vee y_k = z_k]}$;
- $\mathbf{R_2(x, y, z) \Leftrightarrow |x| + |y| = |z|}$;
- $\mathbf{R_3(x, z) \Leftrightarrow \exists y\,[|x| + |y| = |z|]}$;
- $\mathbf{R_4(x, y, z) \Leftrightarrow \exists k \in \{1, 2, \ldots, \min\{|x|, |y|, |z|\}\}\,[x_k = y_k = z_k]}$;
- $\mathbf{R_5(x, y, z) \Leftrightarrow \exists i, j, k\,[x_i = y_j = z_k]}$;
- $\mathbf{R_6(x, y) \Leftrightarrow y = 012 \cdot x \cdot 012}$.

Give a short explanations why certain relations are automatic or not; it is not needed to construct the corresponding automata by explicit tables or diagrammes.

# Theorem 7.7

Theorem

If a relation or function is first-order definable from automatic parameters then it is automatic.

Example

Length-lexicographic ordering:

$$\mathbf{x} <_{\mathbf{ll}} \mathbf{y} \Leftrightarrow |\mathbf{x}| < |\mathbf{y}| \vee (|\mathbf{x}| = |\mathbf{y}| \wedge \mathbf{x} <_{\mathbf{lex}} \mathbf{y}).$$

Length-lexicographic successor:

$$\mathbf{y} = \mathbf{Succ}(\mathbf{x}) \Leftrightarrow \mathbf{x} <_{\mathbf{ll}} \mathbf{y} \wedge \forall \mathbf{z} \left[\mathbf{z} <_{\mathbf{ll}} \mathbf{x} \vee \mathbf{z} = \mathbf{x} \vee \mathbf{z} = \mathbf{y} \vee \mathbf{y} <_{\mathbf{ll}} \mathbf{z}\right].$$

Range $\mathbf{R}$ of a function $\mathbf{f}$ with domain $\mathbf{D}$:

$$\mathbf{y} \in \mathbf{R} \Leftrightarrow \exists \mathbf{x} \left[\mathbf{x} \in \mathbf{D} \wedge \mathbf{y} = \mathbf{f}(\mathbf{x})\right].$$

# Exercise 7.9: Automatic Family

A family $\mathbf{L_e}$ of sets with a regular index set $\mathbf{I}$ is automatic iff $\{\mathbf{conv(e, x)} : \mathbf{x} \in \mathbf{L_e}\}$ is a regular set.

The set $\mathbf{D} = \{\mathbf{x} : \exists \mathbf{e} \in \mathbf{I}\,[\mathbf{x} \in \mathbf{L_e}]\}$ is regular.

Show that the following relations are also automatic.

- $\{(\mathbf{i}, \mathbf{j}) \in \mathbf{I} \times \mathbf{I} : \mathbf{L_i} = \mathbf{L_j}\}$;
- $\{(\mathbf{i}, \mathbf{j}) \in \mathbf{I} \times \mathbf{I} : \mathbf{L_i} \subseteq \mathbf{L_j}\}$;
- $\{(\mathbf{i}, \mathbf{j}) \in \mathbf{I} \times \mathbf{I} : \mathbf{L_i} \cap \mathbf{L_j} = \emptyset\}$;
- $\{(\mathbf{i}, \mathbf{j}) \in \mathbf{I} \times \mathbf{I} : \mathbf{L_i} \cap \mathbf{L_j}$ is infinite$\}$.

Show this by showing that the corresponding relations are first-order definable from given automatic relations. You can use for the fourth the length-lexicographic order in the first-order definition.

# Example 7.10

Let $(\mathbf{N}, \mathbf{\Sigma}, \mathbf{P}, \mathbf{S})$ be a grammar and $\mathbf{R} = \{(\mathbf{x}, \mathbf{y}) \in (\mathbf{N} \cup \mathbf{\Sigma})^* \times (\mathbf{N} \cup \mathbf{\Sigma})^* : \mathbf{x} \Rightarrow \mathbf{y}\}$ be the set of all pairs of words where $\mathbf{y}$ can be derived from $\mathbf{x}$ in one step. The relation $\mathbf{R}$ is automatic.

Furthermore, for each fixed $\mathbf{n}$, the relation $\{(\mathbf{x}, \mathbf{y}) : \exists \mathbf{z_0}, \mathbf{z_1}, \ldots, \mathbf{z_n} [\mathbf{x} = \mathbf{z_0} \wedge \mathbf{y} = \mathbf{z_n} \wedge \mathbf{z_0} \Rightarrow \mathbf{z_1} \wedge \mathbf{z_1} \Rightarrow \mathbf{z_2} \wedge \ldots \wedge \mathbf{z_{n-1}} \Rightarrow \mathbf{z_n}]\}$ of all pairs of words such that $\mathbf{y}$ can be derived from $\mathbf{x}$ in exactly $\mathbf{n}$ steps is automatic.

Similarly, the relation of all $(\mathbf{x}, \mathbf{y})$ such that $\mathbf{y}$ can be derived from $\mathbf{x}$ in at most $\mathbf{n}$ steps is automatic.

# Remark 7.11

The relation $\Rightarrow^*$ is usually not automatic for a non-regular grammar, even if the language generated is regular.

The grammar $(\{\mathbf{S}\}, \{\mathbf{0}, \mathbf{1}, \mathbf{2}\}, \{\mathbf{S} \to \mathbf{SS}|\mathbf{0}|\mathbf{1}|\mathbf{2}\}, \mathbf{S})$ generating all non-empty words over $\{\mathbf{0}, \mathbf{1}, \mathbf{2}\}$. Consider a derivation $\mathbf{S} \Rightarrow^* \mathbf{S01^m2S} \Rightarrow^* \mathbf{0^k1^m2^n}$. If $\Rightarrow^*$ would be automatic, then also the set

$$\mathbf{R} = \{\mathbf{conv}(\mathbf{S01^m2S}, \mathbf{0^k1^m2^n}) : \mathbf{k} > \mathbf{1}, \mathbf{m} > \mathbf{0}, \mathbf{n} > \mathbf{1}\}.$$

Use pumping lemma and choose $\mathbf{n} = \mathbf{h} + \mathbf{4}$, $\mathbf{m} = \mathbf{h}$, $\mathbf{k} = \mathbf{h} + \mathbf{4}$ for $\mathbf{h}$ much larger than the pumping constant. Then for every $\mathbf{r}$ the string

$$\binom{\mathbf{S}}{\mathbf{0}}\binom{\mathbf{0}}{\mathbf{0}}\binom{\mathbf{1}}{\mathbf{0}}^{\mathbf{c}}\binom{\mathbf{1}}{\mathbf{0}}^{\mathbf{dr}}\binom{\mathbf{1}}{\mathbf{0}}^{\mathbf{h-c-d}}\binom{\mathbf{2}}{\mathbf{0}}\binom{\mathbf{S}}{\mathbf{0}}\binom{\#}{\mathbf{1}}^{\mathbf{h}}\binom{\#}{\mathbf{2}}^{\mathbf{h+4}}$$

is in $\mathbf{R}$, where $\mathbf{c} \geq \mathbf{0}$, $\mathbf{d} > \mathbf{0}$ and $\mathbf{h} - \mathbf{c} - \mathbf{d} \geq \mathbf{0}$ are some parameters. For the given $\mathbf{r}$, the substring $\mathbf{01^{h+(r-1)d}2}$ becomes $\mathbf{01^h2}$ in the derivation, impossible for $\mathbf{r} \neq \mathbf{1}$.

# Derivarability in Regular Grammars

However, the relation $\Rightarrow^*$ is regular grammar $(\mathbf{N}, \mathbf{\Sigma}, \mathbf{P}, \mathbf{S})$ in the case that the grammar used is regular.

For every $\mathbf{A}, \mathbf{B} \in \mathbf{N}$, let $\mathbf{L_{A,B}} = \{\mathbf{w} \in \mathbf{\Sigma}^* : \mathbf{A} \Rightarrow^* \mathbf{wB}\}$ and $\mathbf{L_A} = \{\mathbf{w} \in \mathbf{\Sigma}^* : \mathbf{A} \Rightarrow^* \mathbf{w}\}$. All sets $\mathbf{L_A}$ and $\mathbf{L_{A,B}}$ are regular. Note that the concatenation of regular languages is regular.

Now $\{\mathbf{conv}(\mathbf{x}, \mathbf{y}) : \mathbf{x} \Rightarrow^* \mathbf{y}\}$ is the union of all sets $\{\mathbf{conv}(\mathbf{vA}, \mathbf{vwB}) : \mathbf{v} \in \mathbf{\Sigma}^*, \mathbf{w} \in \mathbf{L_{A,B}}\}$ and $\{\mathbf{conv}(\mathbf{vA}, \mathbf{vw}) : \mathbf{v} \in \mathbf{\Sigma}^*, \mathbf{w} \in \mathbf{L_A}\}$ with $\mathbf{A}, \mathbf{B} \in \mathbf{N}$; this union is a regular set.

# Exercise 7.12

## Exercise

Let $\mathbf{R}$ be an automatic relation over $\mathbf{\Sigma^* \cup \Gamma^*}$ such that whenever $\mathbf{(v, w) \in R}$ then $\mathbf{|v| \leq |w|}$ and let $\mathbf{L}$ be the set of all words $\mathbf{x \in \Sigma^*}$ for which there exists a sequence $\mathbf{y_0, y_1, \ldots, y_m \in \Gamma^*}$ with $\mathbf{y_0 = \varepsilon}$, $\mathbf{(y_k, y_{k+1}) \in R}$ for all $\mathbf{k < m}$ and $\mathbf{(y_m, x) \in R}$. Note that $\varepsilon \in \mathbf{L}$ iff $(\varepsilon, \varepsilon) \in \mathbf{R}$.

Show that $\mathbf{L}$ is context-sensitive.

## Comment

The converse direction is also true, as one could take a grammar for $\mathbf{L} - \{\varepsilon\}$ where each rule $\mathbf{v \to w}$ satisfies $\mathbf{|v| \leq |w|}$ and either $\mathbf{v, w \in N^+}$ or $\mathbf{|v| = |w| \wedge v \in N^+ \wedge w \in \Sigma^+}$. Then $\mathbf{(x, y) \in R}$ if either $\mathbf{x, y \in N^+ \wedge x \Rightarrow y}$ or $\mathbf{x \in N^+ \wedge y \in \Sigma^+ \wedge (x \Rightarrow^* y}$ by rules making non-terminals to terminals$)$ or $\mathbf{(x, y) = (\varepsilon, \varepsilon) \wedge \varepsilon \in L}$.

# Context-Sensitive Languages

Theorem [Immerman and Szelepcsényi 1987]
The complement of a context-sensitive language is context-sensitive.

Representation of $\mathbf{\Sigma^* - L}$
The complent of $\mathbf{L}$ will be represented by an automatic relation $\mathbf{R}$ such that $(\mathbf{v}, \mathbf{w}) \in \mathbf{R} \Rightarrow |\mathbf{v}| \leq |\mathbf{w}|$ and $\mathbf{x} \in \mathbf{L}$ iff $\exists \ell \exists \mathbf{d_0}, \mathbf{d_1}, \ldots, \mathbf{d}_\ell [\mathbf{d_0} = \varepsilon \wedge (\mathbf{d_0}, \mathbf{d_1}) \in \mathbf{R} \wedge (\mathbf{d_1}, \mathbf{d_2}) \in \mathbf{R} \wedge \ldots \wedge (\mathbf{d}_{\ell-\mathbf{1}}, \mathbf{d}_\ell) \in \mathbf{R} \wedge (\mathbf{d}_\ell, \mathbf{x}) \in \mathbf{R}]$.

Proof-Method: Nondeterministic Counting
If $\mathbf{i}$ strings can be derived in $\ell$ steps then one can non-deterministically check which string $\mathbf{y}$ can be derived in $\mathbf{i} + \mathbf{1}$ steps and count their number $\mathbf{j}$.

# Basic Algorithm

1. For any $\mathbf{x} \in \mathbf{\Sigma}^*$, try to verify $\mathbf{x} \notin \mathbf{L}$ as follows;
2. Let $\mathbf{u}$ be the length-lexicographically largest string in $(\mathbf{N} \cup \mathbf{\Sigma})^{|\mathbf{x}|}$;
3. Let $\mathbf{i} = \mathbf{Succ_{ll}}(\varepsilon)$;
4. For $\ell = \varepsilon$ to $\mathbf{u}$ Do Begin
5. Let $\mathbf{j} = \varepsilon$;
6. For all $\mathbf{y} \leq_{\mathbf{ll}} \mathbf{u}$ Do Begin
7. Derive words $\mathbf{w_1}, \mathbf{w_2}, \ldots, \mathbf{w_i}$ non-deterministically in length-lexicographic order in up to $\ell$ steps each and check:
8. If some $\mathbf{w_m}$ satisfies $\mathbf{w_m} \Rightarrow \mathbf{y}$ or $\mathbf{w_m} = \mathbf{y}$ then let $\mathbf{j} = \mathbf{Succ_{ll}}(\mathbf{j})$;
9. If some $\mathbf{w_m}$ satisfies $\mathbf{w_m} = \mathbf{x}$ or $\mathbf{w_m} \Rightarrow \mathbf{x}$ then abort computation (as $\mathbf{x} \in \mathbf{L}$); End (of For-Loop 6);
10. Let $\mathbf{i} = \mathbf{j}$; let $\mathbf{j} = \varepsilon$; End (of For-Loop 4);
11. If the algorithm has not been aborted then $\mathbf{x} \notin \mathbf{L}$.

# Refined Algorithm I

1: Choose an $\mathbf{x} \in \mathbf{\Sigma}^+$ and initial all other variables as $\varepsilon$;

2: Let $\mathbf{u} = (\max_{\mathbf{ll}}(\mathbf{N} \cup \mathbf{\Sigma}))^{|\mathbf{x}|}$;

3: Let $\mathbf{i} = \mathbf{Succ}_{\mathbf{ll}}(\varepsilon)$ and $\ell = \varepsilon$;

4: While $\ell <_{\mathbf{ll}} \mathbf{u}$ Do Begin

  5: Let $\mathbf{j} = \varepsilon$;

  6: Let $\mathbf{y} = \varepsilon$;

  7: While $\mathbf{y} <_{\mathbf{ll}} \mathbf{u}$ Do Begin

    8: Let $\mathbf{y} = \mathbf{Succ}_{\mathbf{ll}}(\mathbf{y})$;

    9: $\mathbf{h} = \varepsilon$ and $\mathbf{w} = \varepsilon$;

    10: While $\mathbf{h} <_{\mathbf{ll}} \mathbf{i}$ Do Begin

      11: Nondeterministically replace $\mathbf{w}$ by $\mathbf{w}'$ with $\mathbf{w} <_{\mathbf{ll}} \mathbf{w}' \leq_{\mathbf{ll}} \mathbf{u}$;

      12: Let $\mathbf{v} = \mathbf{S}$;

      13: Let $\mathbf{k} = \varepsilon$;

# Refined Algorithm II

**14:** While $(\mathbf{v} \neq \mathbf{w}) \wedge (\mathbf{k} <_{\mathbf{ll}} \ell)$ Do Begin

**15:** Nondeterministically replace $(\mathbf{k}, \mathbf{v})$ by $(\mathbf{k}', \mathbf{v}')$ with $\mathbf{k} <_{\mathbf{ll}} \mathbf{k}'$ and $\mathbf{v} \Rightarrow \mathbf{v}'$ End (of While in 14);

**16:** If $\mathbf{v} \neq \mathbf{w}$ Then abort the computation;

**17:** If $\mathbf{w} = \mathbf{x}$ or $\mathbf{w} \Rightarrow \mathbf{x}$ Then abort the computation;

**18:** If $\mathbf{w} \neq \mathbf{y}$ and $\mathbf{w} \not\Rightarrow \mathbf{y}$

**19:** Then Let $\mathbf{h} = \mathbf{Succ_{ll}(h)}$

**20:** Else Let $\mathbf{h} = \mathbf{i}$

**21:** End (of While in 10);

**22:** If $\mathbf{w} = \mathbf{y}$ or $\mathbf{w} \Rightarrow \mathbf{y}$ Then $\mathbf{j} = \mathbf{Succ_{ll}(j)}$

**23:** End (of While in 7);

**24:** Let $\mathbf{i} = \mathbf{j}$;

**25:** Let $\ell = \mathbf{Succ_{ll}}(\ell)$ End (of While in 4);

**26:** If the algorithm has not yet aborted Then generate $\mathbf{x}$;

# Exercise 7.14

## Algorithm

Generate all nonempty strings which do not have as length a power of $\mathbf{2}$.

1. Guess $\mathbf{x} \in \mathbf{\Sigma}^+$; Let $\mathbf{y} = \mathbf{0}$;
2. If $|\mathbf{x}| = |\mathbf{y}|$ then abort;
3. Let $\mathbf{z} = \mathbf{y}$;
4. If $|\mathbf{x}| = |\mathbf{y}|$ then generate $\mathbf{x}$ and halt;
5. Remove last $\mathbf{0}$ in $\mathbf{z}$;
6. Let $\mathbf{y} = \mathbf{y0}$;
7. If $\mathbf{z} = \varepsilon$ then goto 2 else goto 4.

Make an $\mathbf{R}$ using the subset of $\mathbf{\Gamma}^*$ consisting of $\mathbf{conv(line, x, y, z)}$. $\mathbf{R}$ should have rules mapping $\varepsilon$ to possible outcomes of line 1 (before line 2), updates from line to line and final moves producing the output from line 4.

# Additional Exercises

Let $\mathbf{x = a_0 a_1 a_2 \ldots a_n \in \{0,1,2\}^n}$ represent the natural number $\sum_{\mathbf{m \leq n}} \mathbf{a_m \cdot 3^m}$. Construct for these ternary function dfas which recognise the graphs of the following automatic functions (as convolutions of input and output), the dfas need only to be correct on inputs of the form of a convolution.

7.15: $\mathbf{x \mapsto x + 1}$.

7.16: $\mathbf{x \mapsto x + x + x + x}$.

7.17: $\mathbf{x \mapsto x + x + 1}$.

7.18: $\mathbf{x \mapsto 3^{n+1} - x - 1}$.

7.19: $\mathbf{x \mapsto (x - a_0)/3 + 3^n \cdot a_0}$.

7.20: $\mathbf{x \mapsto Even(x)}$ ($\{0,1\}$-valued function).