Quotient Complexity of Regular Languages

Janusz Brzozowski

David R. Cheriton School of Computer Science

University of Waterloo

Tallinn University of Technology
Tallinn, Estonia
June 9, 2011

Outline

- Regular Languages

Outline

- Regular Languages
- Quotient Complexity

Outline

- Regular Languages
- Quotient Complexity
- State Complexity

Outline

- Regular Languages
- Quotient Complexity
- State Complexity
- Upper Bounds on Complexity of Operations

Outline

- Regular Languages
- Quotient Complexity
- State Complexity
- Upper Bounds on Complexity of Operations
- Results

Outline

- Regular Languages
- Quotient Complexity
- State Complexity
- Upper Bounds on Complexity of Operations
- Results
- Conclusions

Languages

- Alphabet Σ a finite set of letters
- Set of all words Σ^* free monoid generated by Σ
- Empty word ε
- Language $L \subseteq \Sigma^*$

Operations on Languages

- complement $\overline{L} = \Sigma^* \setminus L$
- union $K \cup L$ intersection $K \cap L$
- difference $K \setminus L$ symmetric difference $K \oplus L$
- general binary boolean operation $K \circ L$

Operations on Languages

- complement $\overline{L} = \Sigma^* \setminus L$
- union $K \cup L$ intersection $K \cap L$
- difference $K \setminus L$ symmetric difference $K \oplus L$
- general binary boolean operation $K \circ L$

- product or (con)catenation, $K \cdot L = \{w \in \Sigma^* \mid w = uv, u \in K, v \in L\}$
- star $K^* = \bigcup_{i \geq 0} K^i$ positive closure $K^+ = \bigcup_{i \geq 1} K^i$
- reverse L^R $\varepsilon^R = \varepsilon$, $(wa)^R = aw^R$

Regular or Rational Languages

- basic languages $\{\emptyset, \{\varepsilon\}\} \cup \{\{a\} \mid a \in \Sigma\}$
- use a finite number of rational operations $\cup$, $\cdot$, *, $(^-)$

Regular or Rational Languages

- basic languages $\{\emptyset, \{\varepsilon\}\} \cup \{\{a\} \mid a \in \Sigma\}$
- use a finite number of rational operations $\cup$, $\cdot$, *, $(^-)$

- Notation clumsy $L = (\{\varepsilon\} \cup \{a\})^* \cdot \{b\}$
- Free algebra over $\{\varepsilon, \emptyset\} \cup \Sigma$ with function symbols $\cup$, $\cdot$, *
- Use regular expression $E = (\varepsilon \cup a)^* \cdot b$

Regular or Rational Languages

- basic languages $\{\emptyset, \{\varepsilon\}\} \cup \{\{a\} \mid a \in \Sigma\}$
- use a finite number of rational operations $\cup$, $\cdot$, *, $(^-)$

- Notation clumsy $L = (\{\varepsilon\} \cup \{a\})^* \cdot \{b\}$
- Free algebra over $\{\varepsilon, \emptyset\} \cup \Sigma$ with function symbols $\cup$, $\cdot$, *
- Use regular expression $E = (\varepsilon \cup a)^* \cdot b$

- Mapping $\mathcal{L}$ from free algebra to regular languages
- $\mathcal{L}(\emptyset) = \emptyset, \quad \mathcal{L}(\varepsilon) = \{\varepsilon\}, \quad \mathcal{L}(a) = \{a\}$
- $\mathcal{L}(E \cup F) = \mathcal{L}(E) \cup \mathcal{L}(F)$
- $\mathcal{L}(E \cdot F) = \mathcal{L}(E) \cdot \mathcal{L}(F), \quad \mathcal{L}(E^*) = (\mathcal{L}(E))^*$
- $\mathcal{L}(\overline{E}) = \overline{\mathcal{L}(E)}$

Quotient Complexity of a Language

- Left quotient, or quotient of a language L by a word w
- The language $L_w = \{x \in \Sigma^* \mid wx \in L\}$

Quotient Complexity of a Language

- Left quotient, or quotient of a language L by a word w
- The language $L_w = \{x \in \Sigma^* \mid wx \in L\}$

- The quotient complexity of L is the number of quotients of L
- Denoted by $\kappa(L)$ (kappa for both kwotient and komplexity)
- $\kappa(L)$ defined for any language, and may be finite or infinite

Quotient Complexity of a Language

- Left quotient, or quotient of a language L by a word w
- The language $L_w = \{x \in \Sigma^* \mid wx \in L\}$

- The quotient complexity of L is the number of quotients of L
- Denoted by $\kappa(L)$ (kappa for both kwotient and komplexity)
- $\kappa(L)$ defined for any language, and may be finite or infinite

Example

- Example: $\Sigma = \{a, b\}$, $L = a\Sigma^*$ $\kappa(L) = 3$
  - $L_\varepsilon = L$
  - $L_a = \Sigma^* = L_{aa} = L_{ab}$
  - $L_b = \emptyset = L_{ba} = L_{bb}$

Finding Quotients: The ε-Function

Does L contain the empty word?

$$x^\varepsilon = \begin{cases} \emptyset, & \text{if } x = \emptyset, \text{ or } x \in \Sigma; \\ \varepsilon, & \text{if } x = \varepsilon \end{cases}$$

$$(\overline{L})^\varepsilon = \begin{cases} \emptyset, & \text{if } L^\varepsilon = \varepsilon; \\ \varepsilon, & \text{if } L^\varepsilon = \emptyset \end{cases}$$

Finding Quotients: The ε-Function

Does L contain the empty word?

$$x^\varepsilon = \begin{cases} \emptyset, & \text{if } x = \emptyset, \text{ or } x \in \Sigma; \\ \varepsilon, & \text{if } x = \varepsilon \end{cases}$$

$$(\overline{L})^\varepsilon = \begin{cases} \emptyset, & \text{if } L^\varepsilon = \varepsilon; \\ \varepsilon, & \text{if } L^\varepsilon = \emptyset \end{cases}$$

$$\begin{aligned} (K \cup L)^\varepsilon &= K^\varepsilon \cup L^\varepsilon \\ (KL)^\varepsilon &= K^\varepsilon \cap L^\varepsilon \\ (L^*)^\varepsilon &= \varepsilon \end{aligned}$$

Quotient by a Letter

$$x_a = \begin{cases} \emptyset, & \text{if } x \in \{\emptyset, \varepsilon\}, \text{ or } x \in \Sigma \text{ and } x \neq a; \\ \varepsilon, & \text{if } x = a \end{cases}$$

Quotient by a Letter

$$x_a = \begin{cases} \emptyset, & \text{if } x \in \{\emptyset, \varepsilon\}, \text{ or } x \in \Sigma \text{ and } x \neq a; \\ \varepsilon, & \text{if } x = a \end{cases}$$

$$\begin{aligned} (\overline{L})_a &= \overline{(L_a)} \\ (K \cup L)_a &= K_a \cup L_a \\ (KL)_a &= K_a L \cup K^\varepsilon L_a \\ (L^*)_a &= L_a L^* \end{aligned}$$

Quotient by a Word

$$
\begin{aligned}
L_\varepsilon &= L \\
L_w &= L_a, \text{ if } w = a \in \Sigma \\
L_{wa} &= (L_w)_a
\end{aligned}
$$

Quotient by a Word

$$
\begin{aligned}
L_\varepsilon &= L \\
L_w &= L_a, \text{ if } w = a \in \Sigma \\
L_{wa} &= (L_w)_a
\end{aligned}
$$

- Calculating quotients, we get expressions called derivatives
- There is an infinite number of distinct derivatives

Quotient by a Word

$$\begin{aligned} L_\varepsilon &= L \\ L_w &= L_a, \text{ if } w = a \in \Sigma \\ L_{wa} &= (L_w)_a \end{aligned}$$

- Calculating quotients, we get expressions called derivatives
- There is an infinite number of distinct derivatives

Example

$(a^*)_a = (a)_a a^* = \varepsilon a^*$
$(a^*)_{aa} = (\varepsilon a^*)_a = \varepsilon_a a^* \cup \varepsilon (a^*)_a = \emptyset a^* \cup \varepsilon(\varepsilon a^*)$, etc.

Similarity Laws

$$
\begin{aligned}
L \cup L &= L \\
K \cup L &= L \cup K \\
K \cup (L \cup M) &= (K \cup L) \cup M \\
L \cup \emptyset &= L \\
L\emptyset &= \emptyset L = \emptyset \\
L\varepsilon &= \varepsilon L = L
\end{aligned}
$$

Quotients, Equations, Automata

Example

- Example: $\Sigma = \{a, b\}$, $L = a\Sigma^*$
  - $L_\varepsilon = L$
  - $L_a = \Sigma^* = L_{aa} = L_{ab}$
  - $L_b = \emptyset = L_{ba} = L_{bb}$

Quotients, Equations, Automata

Example

- Example: $\Sigma = \{a, b\}$, $L = a\Sigma^*$
  - $L_\varepsilon = L$
  - $L_a = \Sigma^* = L_{aa} = L_{ab}$
  - $L_b = \emptyset = L_{ba} = L_{bb}$

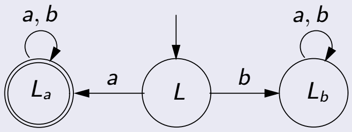
$$\begin{aligned} L &= aL_a \cup bL_b, \\ L_a &= aL_a \cup bL_a \cup \varepsilon, \\ L_b &= aL_b \cup bL_b. \end{aligned}$$

Quotients, Equations, Automata

Example

- Example: $\Sigma = \{a, b\}$, $L = a\Sigma^*$
  - $L_\varepsilon = L$
  - $L_a = \Sigma^* = L_{aa} = L_{ab}$
  - $L_b = \emptyset = L_{ba} = L_{bb}$

$$\begin{aligned} L &= aL_a \cup bL_b, \\ L_a &= aL_a \cup bL_a \cup \varepsilon, \\ L_b &= aL_b \cup bL_b. \end{aligned}$$



Extended Regular Expressions

Example ($L = \Sigma^* a \Sigma^* \cap \overline{\Sigma^* bb \Sigma^*}$)

- $L_\varepsilon = L \quad L_a = \overline{\Sigma^* bb \Sigma^*}$
- $L_b = \Sigma^* a \Sigma^* \cap \overline{\Sigma^* bb \Sigma^* \cup b\Sigma^*} \quad L_{aa} = L_a$
- $L_{ab} = \overline{\Sigma^* bb \Sigma^* \cup b\Sigma^*} \quad L_{ba} = \overline{\Sigma^* bb \Sigma^*} = L_a$
- $L_{bb} = \emptyset \quad L_{aba} = L_a \quad L_{abb} = \emptyset$

Extended Regular Expressions

Example ($L = \Sigma^* a \Sigma^* \cap \overline{\Sigma^* bb \Sigma^*}$)

- $L_\varepsilon = L \quad L_a = \overline{\Sigma^* bb\Sigma^*}$
- $L_b = \Sigma^* a \Sigma^* \cap \overline{\Sigma^* bb\Sigma^* \cup b\Sigma^*} \quad L_{aa} = L_a$
- $L_{ab} = \overline{\Sigma^* bb\Sigma^* \cup b\Sigma^*} \quad L_{ba} = \overline{\Sigma^* bb\Sigma^*} = L_a$
- $L_{bb} = \emptyset \quad L_{aba} = L_a \quad L_{abb} = \emptyset$

$$\begin{aligned} L &= aL_a \cup bL_b, \\ L_a &= aL_a \cup bL_{ab} \cup \varepsilon, \\ L_b &= aL_a \cup b\emptyset, \\ L_{ab} &= aL_a \cup b\emptyset \end{aligned}$$

Solving Equations $X = AX \cup B \Longrightarrow X = A^*B$

Example ($L = \Sigma^* a \Sigma^* \cap \overline{\Sigma^* bb \Sigma^*}$)

$$
\begin{aligned}
L &= aL_a \cup bL_b \\
L_a &= aL_a \cup bL_{ab} \cup \varepsilon \\
L_b &= aL_a \cup b\emptyset \\
L_{ab} &= aL_a \cup b\emptyset
\end{aligned}
$$

Solving Equations $X = AX \cup B \implies X = A^*B$

Example ($L = \Sigma^* a \Sigma^* \cap \overline{\Sigma^* bb \Sigma^*}$)

$$\begin{aligned}
L &= aL_a \cup bL_b \\
L_a &= aL_a \cup bL_{ab} \cup \varepsilon \\
L_b &= aL_a \cup b\emptyset \\
L_{ab} &= aL_a \cup b\emptyset
\end{aligned}$$

$$\begin{aligned}
L &= aL_a \cup bL_b \\
L_a &= aL_a \cup bL_b \cup \varepsilon \\
L_b &= aL_a \cup b\emptyset
\end{aligned}$$

Solving Equations $X = AX \cup B \Longrightarrow X = A^*B$

Example ($L = \Sigma^* a \Sigma^* \cap \overline{\Sigma^* bb \Sigma^*}$)

$$\begin{aligned} L &= aL_a \cup bL_b \\ L_a &= aL_a \cup bL_{ab} \cup \varepsilon \\ L_b &= aL_a \cup b\emptyset \\ L_{ab} &= aL_a \cup b\emptyset \end{aligned}$$

$$\begin{aligned} L &= aL_a \cup bL_b \\ L_a &= aL_a \cup bL_b \cup \varepsilon \\ L_b &= aL_a \cup b\emptyset \end{aligned}$$

$$\begin{aligned} L &= (a \cup ba)L_a \\ L_a &= (a \cup ba)L_a \cup \varepsilon = (a \cup ba)^* \\ L &= (a \cup ba)(a \cup ba)^* \end{aligned}$$

Automata

Deterministic finite automaton DFA $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$

- Q set of states
- $\delta : Q \times \Sigma \to Q$ transition function
- q_0 initial state
- $F \subseteq Q$ set of final or accepting states

Automata

Deterministic finite automaton DFA $\mathcal{A} = (Q, \Sigma, \delta, q_0, F)$

- Q set of states
- $\delta : Q \times \Sigma \to Q$ transition function
- q_0 initial state
- $F \subseteq Q$ set of final or accepting states

Nondeterministic finite automaton NFA $\mathcal{N} = (Q, \Sigma, \delta, I, F)$

- Q set of states
- $\delta : Q \times \Sigma \to 2^Q$ transition function
- I set of initial states
- $F \subseteq Q$ set of final or accepting states

Quotient Automaton

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

- $Q = \{L_w \mid w \in \Sigma^*\}$
- $\delta(L_w, a) = L_{wa}$
- $q_0 = L_\varepsilon = L$
- $F = \{L_w \mid \varepsilon \in L_w\}$ accepting or final quotients

Quotient Automaton

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

- $Q = \{L_w \mid w \in \Sigma^*\}$
- $\delta(L_w, a) = L_{wa}$
- $q_0 = L_\varepsilon = L$
- $F = \{L_w \mid \varepsilon \in L_w\}$ accepting or final quotients

L is recognizable if and only if the number of quotients is finite (Nerode, 1958; Brzozowski, 1962)

State Complexity

State complexity of L is the number of states in the minimal DFA recognizing L

State Complexity

State complexity of L is the number of states in the minimal DFA recognizing L

- Why define the complexity of a language by the size of its automaton, a different object?

State Complexity

State complexity of L is the number of states in the minimal DFA recognizing L

- Why define the complexity of a language by the size of its automaton, a different object?
- Quotient DFA of L is isomorphic to the minimal DFA of L

State Complexity

State complexity of L is the number of states in the minimal DFA recognizing L

- Why define the complexity of a language by the size of its automaton, a different object?
- Quotient DFA of L is isomorphic to the minimal DFA of L
- State complexity = quotient complexity

State Complexity

State complexity of L is the number of states in the minimal DFA recognizing L

- Why define the complexity of a language by the size of its automaton, a different object?
- Quotient DFA of L is isomorphic to the minimal DFA of L
- State complexity = quotient complexity
- Quotient complexity is more natural

State Complexity

State complexity of L is the number of states in the minimal DFA recognizing L

- Why define the complexity of a language by the size of its automaton, a different object?
- Quotient DFA of L is isomorphic to the minimal DFA of L
- State complexity = quotient complexity
- Quotient complexity is more natural
- Quotients have some advantages

Complexity of Operations

- A subclass $\mathcal{C}$ of regular languages
- $L_1, \ldots, L_k \in \mathcal{C}$ with quotient complexities $n_1, \ldots, n_k$
- A k-ary operation f on $L_1, \ldots, L_k$
- Quotient complexity of $f(L_1, \ldots, L_k)$
- Quotient complexity of f in $\mathcal{C}$ is the worst case quotient complexity of $f(L_1, \ldots, L_k)$ as $L_1, \ldots, L_k$ range over $\mathcal{C}$
- A function of $n_1, \ldots, n_k$

Complexity of Operations

- A subclass $\mathcal{C}$ of regular languages
- $L_1, \dots, L_k \in \mathcal{C}$ with quotient complexities $n_1, \dots, n_k$
- A k-ary operation f on $L_1, \dots, L_k$
- Quotient complexity of $f(L_1, \dots, L_k)$
- Quotient complexity of f in $\mathcal{C}$ is the worst case quotient complexity of $f(L_1, \dots, L_k)$ as $L_1, \dots, L_k$ range over $\mathcal{C}$
- A function of $n_1, \dots, n_k$

Example

- Regular languages K and L with $\kappa(K) = m$, $\kappa(L) = n$
- Union: $\kappa(K \cup L) \leq mn$
- Complement: $\kappa(\overline{L}) = \kappa(L) = n$

Some Early Work on State Complexity

- 1957, Rabin and Scott: upper bound of mn for intersection
- 1962, Brzozowski: upper bounds for union, product and star
- 1963, Lupanov: NFA to DFA conversion bound of 2^n is tight
- 1964, Lyubich: unary case
- 1966, Mirkin: 2^n bound for reversal is attainable
- 1970, Maslov: examples meeting bounds for union, concatenation, star and other operations
- 1971, Moore: NFA to DFA conversion bound of 2^n is tight (rediscovered)

Some Early Work on State Complexity

- 1957, Rabin and Scott: upper bound of mn for intersection
- 1962, Brzozowski: upper bounds for union, product and star
- 1963, Lupanov: NFA to DFA conversion bound of 2^n is tight
- 1964, Lyubich: unary case
- 1966, Mirkin: 2^n bound for reversal is attainable
- 1970, Maslov: examples meeting bounds for union, concatenation, star and other operations
- 1971, Moore: NFA to DFA conversion bound of 2^n is tight (rediscovered)

Renewed interest

- 1991, Birget: "state complexity"
- 1994, Yu, Zhuang, K. Salomaa: complexity of operations

Upper Bounds Using Automata

- Given automata $\mathcal{A}$, $\mathcal{B}$ of languages K, L, find $\kappa(f(K, L))$

Upper Bounds Using Automata

- Given automata $\mathcal{A}$, $\mathcal{B}$ of languages K, L, find $\kappa(f(K, L))$
- Check if automata "complete"

Upper Bounds Using Automata

- Given automata $\mathcal{A}$, $\mathcal{B}$ of languages K, L, find $\kappa(f(K, L))$
- Check if automata "complete"
- If there is a "sink state", is there only one?

Upper Bounds Using Automata

- Given automata $\mathcal{A}$, $\mathcal{B}$ of languages K, L, find $\kappa(f(K, L))$
- Check if automata "complete"
- If there is a "sink state", is there only one?
- Is every state "useful"?

Upper Bounds Using Automata

- Given automata $\mathcal{A}$, $\mathcal{B}$ of languages K, L, find $\kappa(f(K, L))$
- Check if automata "complete"
- If there is a "sink state", is there only one?
- Is every state "useful"?

Upper Bounds Using Automata

- Given automata $\mathcal{A}$, $\mathcal{B}$ of languages K, L, find $\kappa(f(K,L))$
- Check if automata "complete"
- If there is a "sink state", is there only one?
- Is every state "useful"?

- Construct DFA for $f(K,L)$ directly

Upper Bounds Using Automata

- Given automata $\mathcal{A}$, $\mathcal{B}$ of languages K, L, find $\kappa(f(K,L))$
- Check if automata "complete"
- If there is a "sink state", is there only one?
- Is every state "useful"?

- Construct DFA for $f(K,L)$ directly
- Construct NFA, convert to DFA

Upper Bounds Using Automata

- Given automata $\mathcal{A}$, $\mathcal{B}$ of languages K, L, find $\kappa(f(K, L))$
- Check if automata "complete"
- If there is a "sink state", is there only one?
- Is every state "useful"?

- Construct DFA for $f(K, L)$ directly
- Construct NFA, convert to DFA
- Use NFA with ε transitions

Upper Bounds Using Automata

- Given automata $\mathcal{A}$, $\mathcal{B}$ of languages K, L, find $\kappa(f(K, L))$
- Check if automata "complete"
- If there is a "sink state", is there only one?
- Is every state "useful"?

- Construct DFA for $f(K, L)$ directly
- Construct NFA, convert to DFA
- Use NFA with ε transitions
- Use NFA with multiple initial states

Upper Bounds Using Automata

- Given automata $\mathcal{A}$, $\mathcal{B}$ of languages K, L, find $\kappa(f(K, L))$
- Check if automata "complete"
- If there is a "sink state", is there only one?
- Is every state "useful"?

- Construct DFA for $f(K, L)$ directly
- Construct NFA, convert to DFA
- Use NFA with ε transitions
- Use NFA with multiple initial states

Upper Bounds Using Automata

- Given automata $\mathcal{A}$, $\mathcal{B}$ of languages K, L, find $\kappa(f(K,L))$
- Check if automata "complete"
- If there is a "sink state", is there only one?
- Is every state "useful"?

- Construct DFA for $f(K,L)$ directly
- Construct NFA, convert to DFA
- Use NFA with ε transitions
- Use NFA with multiple initial states

- There is an alternative: Quotient complexity

Formulas for Boolean Operations and Product

Theorem

If K and L are regular expressions, then

$$(\overline{L})_w = \overline{L_w}$$

$$(K \circ L)_w = K_w \circ L_w$$

$$(KL)_w = K_w L \cup K^\varepsilon L_w \cup \left(\bigcup_{\substack{w=uv \\ u,v \in \Sigma^+}} K_u^\varepsilon L_v \right)$$

Example Formula for Product: $(KL)_w = K_w L \cup K^\varepsilon L_w \cup \left(\bigcup_{\substack{w=uv \\ u,v \in \Sigma^+}} K_u^\varepsilon L_v \right)$

Example

- $\kappa(G) = n$
- $(\Sigma^* G)_w = \Sigma^* G \cup G_w \cup \bigcup_{w=uv} G_v$
- G is always present on the right-hand side
- At most 2^{n-1} subsets of quotients to be added to $\Sigma^* G$
- $\kappa(\Sigma^* G) \leq 2^{n-1}$

Formula for Star

Theorem

For the empty word

$$(L^*)_\varepsilon = \varepsilon \cup LL^*$$

and for $w \in \Sigma^+$

$$(L^*)_w = \left(L_w \cup \bigcup_{\substack{w=uv \\ u,v\in\Sigma^+}} (L^*)_u^\varepsilon L_v \right) L^*$$

Quotient Formulas

All you have to do is count!

Upper bounds for operations

Theorem

For any languages K and L with $\kappa(K) = m$ and $\kappa(L) = n$,

- $\kappa(\overline{L}) = n$. $\kappa(K \circ L) \leq mn$.
- *If K (L) has k (ℓ) accepting quotients, then*
  - *If $k = 0$ or $\ell = 0$, then $\kappa(KL) = 1$.*
  - *If $k, \ell > 0$ and $n = 1$, then $\kappa(KL) \leq m - (k - 1)$.*
  - *If $k, \ell > 0$ and $n > 1$, then $\kappa(KL) \leq m2^n - k2^{n-1}$.*

Upper bounds for operations

Theorem

For any languages K and L with $\kappa(K) = m$ and $\kappa(L) = n$,

- $\kappa(\overline{L}) = n$. $\kappa(K \circ L) \le mn$.
- *If K (L) has k (ℓ) accepting quotients, then*
  - *If $k = 0$ or $\ell = 0$, then* $\kappa(KL) = 1$.
  - *If $k, \ell > 0$ and $n = 1$, then* $\kappa(KL) \le m - (k - 1)$.
  - *If $k, \ell > 0$ and $n > 1$, then* $\kappa(KL) \le m2^n - k2^{n-1}$.

Claim for boolean operations is obvious since $(\overline{L})_w = \overline{L_w}$ and $(K \cup L)_w = K_w \cup L_w$

Proof for product $(KL)_w = K_w L \cup K^\varepsilon L_w \cup (\bigcup_{\substack{w=uv \\ u,v\in\Sigma^+}} K_u^\varepsilon L_v)$

- if $k = 0$ or $\ell = 0$, then $KL = \emptyset$ and $\kappa(KL) = 1$
- If $k, l > 0$, $n = 1$, then $L = \Sigma^*$ and $w \in K \Rightarrow (KL)_w = \Sigma^*$
- All k accepting quotients of K produce Σ^* in KL (1)
- For each rejecting quotient of K, we have two choices for the union of quotients of L: the empty union or Σ^*
- If we choose the empty union, at most $m - k$ quotients of KL
- Choosing Σ^* results in $(KL)_w = \Sigma^*$, which has been counted
- Altogether, there are at most $1 + m - k$ quotients of KL

Proof for product $(KL)_w = K_w L \cup K^\varepsilon L_w \cup (\bigcup_{\substack{w=uv \\ u,v\in\Sigma^+}} K_u^\varepsilon L_v)$

- $k, l > 0$ and $n > 1$
- If $w \notin K$, then we can choose K_w in $m - k$ ways, and the union of quotients of L in 2^n ways
- If $w \in K$, then we can choose K_w in k ways, and the set of quotients of L in 2^{n-1} ways, since L is then always present
- Thus we have $(m - k)2^n + k2^{n-1}$

Star

Let $M = L^*$, $w \neq \varepsilon$

$$M_w = (L_w \cup M_w^\varepsilon L \cup \bigcup_{\substack{w=uv \\ u,v\in\Sigma^+}} M_u^\varepsilon L_v)M$$

Theorem

- *If $n = 1$, then $\kappa(L^*) \leq 2$.*
- *If $n > 1$ and only L_ε accepts, then $\kappa(L^*) = n$.*
- *If $n > 1$ and L has $l > 0$ accepting quotients $\neq L$, then $\kappa(L^*) \leq 2^{n-1} + 2^{n-l-1}$.*

Witnesses to bounds

- This is a challenging problem
- Take a guess
- How do you prove the guess meets the bound?
- Use quotients, of course!

Witnesses to bounds

Example

- Symmetric difference, $K \oplus L$

Witnesses to bounds

Example

- Symmetric difference, $K \oplus L$
- $K = (b^*a)^{m-1}(a \cup b)^*$, $L = (a^*b)^{n-1}(a \cup b)^*$

Witnesses to bounds

Example

- Symmetric difference, $K \oplus L$
- $K = (b^*a)^{m-1}(a \cup b)^*$, $L = (a^*b)^{n-1}(a \cup b)^*$
- Words $a^i b^j$, $0 \le i \le m-1$, $0 \le j \le n-1$

Witnesses to bounds

Example

- Symmetric difference, $K \oplus L$
- $K = (b^*a)^{m-1}(a \cup b)^*$, $L = (a^*b)^{n-1}(a \cup b)^*$
- Words a^ib^j, $0 \le i \le m-1$, $0 \le j \le n-1$
- Let $x = a^ib^j$ and $y = a^kb^\ell$

Witnesses to bounds

Example

- Symmetric difference, $K \oplus L$
- $K = (b^*a)^{m-1}(a \cup b)^*$, $L = (a^*b)^{n-1}(a \cup b)^*$
- Words $a^i b^j$, $0 \leq i \leq m-1$, $0 \leq j \leq n-1$
- Let $x = a^i b^j$ and $y = a^k b^\ell$
- If $i < k$, let $u = a^{m-1-k} b^n$

Witnesses to bounds

Example

- Symmetric difference, $K \oplus L$
- $K = (b^*a)^{m-1}(a \cup b)^*$, $L = (a^*b)^{n-1}(a \cup b)^*$
- Words $a^i b^j$, $0 \leq i \leq m-1$, $0 \leq j \leq n-1$
- Let $x = a^i b^j$ and $y = a^k b^\ell$
- If $i < k$, let $u = a^{m-1-k} b^n$
- xu not full of a's, is full of b's, yu full of a's and b's

Witnesses to bounds

Example

- Symmetric difference, $K \oplus L$
- $K = (b^*a)^{m-1}(a \cup b)^*$, $L = (a^*b)^{n-1}(a \cup b)^*$
- Words $a^i b^j$, $0 \leq i \leq m-1$, $0 \leq j \leq n-1$
- Let $x = a^i b^j$ and $y = a^k b^\ell$
- If $i < k$, let $u = a^{m-1-k} b^n$
- xu not full of a's, is full of b's, yu full of a's and b's
- Then $xu \notin K$, $yu \in K$, and $xu, yu \in L$

Witnesses to bounds

Example

- Symmetric difference, $K \oplus L$
- $K = (b^*a)^{m-1}(a \cup b)^*$, $L = (a^*b)^{n-1}(a \cup b)^*$
- Words $a^i b^j$, $0 \le i \le m-1$, $0 \le j \le n-1$
- Let $x = a^i b^j$ and $y = a^k b^\ell$
- If $i < k$, let $u = a^{m-1-k} b^n$
- xu not full of a's, is full of b's, yu full of a's and b's
- Then $xu \notin K$, $yu \in K$, and $xu, yu \in L$
- $xu \in K \oplus L$, and $yu \notin K \oplus L$, *i.e.*, $(K \oplus L)_x \neq (K \oplus L)_y$

Witnesses to bounds

Example

- Symmetric difference, $K \oplus L$
- $K = (b^*a)^{m-1}(a \cup b)^*$, $L = (a^*b)^{n-1}(a \cup b)^*$
- Words $a^i b^j$, $0 \leq i \leq m-1$, $0 \leq j \leq n-1$
- Let $x = a^i b^j$ and $y = a^k b^\ell$
- If $i < k$, let $u = a^{m-1-k} b^n$
- xu not full of a's, is full of b's, yu full of a's and b's
- Then $xu \notin K$, $yu \in K$, and $xu, yu \in L$
- $xu \in K \oplus L$, and $yu \notin K \oplus L$, *i.e.*, $(K \oplus L)_x \neq (K \oplus L)_y$
- Case $j < \ell$ is similar

Witnesses to bounds

Example

- Symmetric difference, $K \oplus L$
- $K = (b^*a)^{m-1}(a \cup b)^*$, $L = (a^*b)^{n-1}(a \cup b)^*$
- Words $a^i b^j$, $0 \le i \le m-1$, $0 \le j \le n-1$
- Let $x = a^i b^j$ and $y = a^k b^\ell$
- If $i < k$, let $u = a^{m-1-k} b^n$
- xu not full of a's, is full of b's, yu full of a's and b's
- Then $xu \notin K$, $yu \in K$, and $xu, yu \in L$
- $xu \in K \oplus L$, and $yu \notin K \oplus L$, *i.e.*, $(K \oplus L)_x \neq (K \oplus L)_y$
- Case $j < \ell$ is similar
- All quotients of $K \oplus L$ by these mn words are distinct

Recent work on quotient complexity

- TCS 1994: Yu, Zhuang, K. Salomaa: regular languages (state complexity)
- WIA 2001, Câmpeanu, Culik, Salomaa, Yu: finite languages
- DCFS 2009: Brzozowski: regular languages (quotients)
- TCS 2009: Han Salomaa: suffix-free languages
- 2009: Han, Salomaa, Wood: prefix-free languages
- LATIN 2010, Brzozowski, Jirásková, Li: ideal languages
- CSR 2010, Brzozowski, Jirásková, Zou: closed languages
- AFL 2011, Brzozowski, Liu: star-free languages
- AFL 2011, Brzozowski, Jirásková, Li, Smith: bifix-, factor-, subword-free languages

Prefixes, Suffixes, Factors and Subwords

- $w = uv$ u is a prefix of w

Prefixes, Suffixes, Factors and Subwords

- $w = uv$ $\quad u$ is a prefix of w
- üks is a prefix of üksteist

Prefixes, Suffixes, Factors and Subwords

- $w = uv$ u is a prefix of w
- üks is a prefix of üksteist
- $w = uv$ v is a suffix of w

Prefixes, Suffixes, Factors and Subwords

- $w = uv$ u is a prefix of w
- üks is a prefix of üksteist
- $w = uv$ v is a suffix of w
- päev is suffix of laupäev

Prefixes, Suffixes, Factors and Subwords

- $w = uv$ u is a prefix of w
- üks is a prefix of üksteist
- $w = uv$ v is a suffix of w
- päev is suffix of laupäev
- $w = uxv$ x is a factor of w

Prefixes, Suffixes, Factors and Subwords

- $w = uv$ u is a prefix of w
- üks is a prefix of üksteist
- $w = uv$ v is a suffix of w
- päev is suffix of laupäev
- $w = uxv$ x is a factor of w
- serv is a factor of konservatiivsus

Prefixes, Suffixes, Factors and Subwords

- $w = uv$ u is a prefix of w
- üks is a prefix of üksteist
- $w = uv$ v is a suffix of w
- päev is suffix of laupäev
- $w = uxv$ x is a factor of w
- serv is a factor of konservatiivsus
- $w = w_0 a_0 w_1 a_1 \cdots a_n w_n$ $a_0 \cdots a_n$ is a subword of w

Prefixes, Suffixes, Factors and Subwords

- $w = uv$ u is a prefix of w
- üks is a prefix of üksteist
- $w = uv$ v is a suffix of w
- päev is suffix of laupäev
- $w = uxv$ x is a factor of w
- serv is a factor of konservatiivsus
- $w = w_0 a_0 w_1 a_1 \cdots a_n w_n$ $a_0 \cdots a_n$ is a subword of w
- kaks is a subword of kaheksa

Convex Languages

- A language L is prefix-convex if u is a prefix of v, v is a prefix of w and $u, w \in L$ implies $v \in L$
- L is prefix-closed if u is a prefix of v and $v \in L$ implies u in L
- L is converse prefix-closed if u is a prefix of v, and $u \in L$ implies $v \in L$ right ideal
- L is prefix-free if $u \neq v$ is a prefix of v and $v \in L$ implies $u \notin L$ prefix code

Convex Languages

- A language L is prefix-convex if u is a prefix of v, v is a prefix of w and $u, w \in L$ implies $v \in L$
- L is prefix-closed if u is a prefix of v and $v \in L$ implies u in L
- L is converse prefix-closed if u is a prefix of v, and $u \in L$ implies $v \in L$ right ideal
- L is prefix-free if $u \neq v$ is a prefix of v and $v \in L$ implies $u \notin L$ prefix code

- L is suffix-convex
- L is factor-convex
- L is subword-convex
- L is bifix-convex

Closed Languages

- L is prefix-closed
- L is suffix-closed
- L is factor-closed
- L is subword-closed
- L is bifix-closed if it is both prefix- and suffix-closed if and only if it is factor closed

Ideal Languages

L is nonempty

- Right ideal $L = L\Sigma^*$
- Left ideal $L = \Sigma^* L$
- Two-sided ideal $L = \Sigma^* L \Sigma^*$
- All-sided ideal $L = \Sigma^* \sqcup\!\sqcup\, L$

Ideal Languages

L is nonempty

- Right ideal $L = L\Sigma^*$
- Left ideal $L = \Sigma^* L$
- Two-sided ideal $L = \Sigma^* L \Sigma^*$
- All-sided ideal $L = \Sigma^* \sqcup\!\sqcup\, L$

- Shuffle: let $w = a_1 a_2 \cdots a_k, \quad a_i \in \Sigma$
  $\Sigma^* \sqcup\!\sqcup\, w = \Sigma^* \sqcup\!\sqcup\, (a_1 a_2 \cdots a_k) = \Sigma^* a_1 \Sigma^* a_2 \Sigma^* \cdots \Sigma^* a_k \Sigma^*$
  $\Sigma^* \sqcup\!\sqcup\, L = \bigcup_{w \in L} (\Sigma^* \sqcup\!\sqcup\, w)$

X-Free Languages

- L is prefix-free:
- L is suffix-free
- L is factor-free
- L is subword-free
- L is bifix-free if it is both prefix- and suffix-free

Star-Free Languages

- $\emptyset$, $\{\varepsilon\}$, $\{a\}$, $a \in \Sigma$ are star-free
- If K and L are star-free, then so are
  - $\overline{L}$
  - $K \cup L$
  - KL

Star-Free Languages

- $\emptyset$, $\{\varepsilon\}$, $\{a\}$, $a \in \Sigma$ are star-free
- If K and L are star-free, then so are
  - $\overline{L}$
  - $K \cup L$
  - KL
- The smallest class of languages containing finite languages and closed under boolean operations and product

Tight Upper Bounds for Union ($|\Sigma|$)

- mn regular (2), star-free (2), prefix-, factor-, subword-closed (2), suffix-closed (4), left ideal (4)
- $mn - 2$ prefix-free (2)
- $mn - (m + n - 2)$ suffix-free (2), right, two-sided, all-sided ideal (2)
- $mn - (m + n)$ bifix-, factor-free (3), subword-free $(m + n - 3)$, finite $(mn - 2(m + n) + 5)$
- $max(m, n)$ free unary, closed unary
- $min(m, n)$ ideal unary

Similar results for intersection, difference, symmetric difference

Tight Upper Bounds for Product ($|\Sigma|$)

- $(m-1)2^n + 2^{n-1}$ regular (2), star-free (4)
- $(m-1)2^{n-1} + 1$ suffix-free (3)
- $(m+1)2^{n-2}$ prefix-closed (3)
- $m + 2^{n-2}$ right ideal (3)
- $(m-1)n + 1$ suffix-closed (3)
- $m + n - 1$ left, two-sided, all-sided ideal (1), unary ideal, factor-closed (2), subword-closed (2)
- $m + n - 2$ closed unary, free unary, prefix-, bifix-, factor, subword-free (1)

Tight Upper Bounds for Star ($|\Sigma|$)

- $2^{n-1} + 2^{n-2}$ regular (2), star-free (4)
- $2^{n-2} + 1$ prefix-closed (3), suffix-free (2)
- $2^{n-3} + 2^{n-4}$ finite (3)
- $n^2 - 7n + 13$ finite unary, star-free unary
- $n + 1$ left, right, two-sided, all-sided ideals (2)
- n free unary, suffix-closed (2), prefix-free (2)
- $n - 1$ bifix-, factor-, subword-free (2)
- 2 closed unary, factor-, subword-closed (2)

Tight Upper Bounds for Reversal ($|\Sigma|$)

- 2^n regular (2),
- $2^n - 1$ star-free ($n - 1$)
- $2^{n-1} + 1$ suffix-closed (3), left ideal (3)
- 2^{n-1} prefix-closed (2), right ideal (2)
- $2^{n-2} + 1$ free unary, prefix-, suffix-free (3), factor-closed (3), subword-closed (2n), two-sided, all-sided ideal (3)
- $2^{n-3} + 2$ bifix-, factor-free (3), subword-free ($2^{n-3} - 1$)
- $2^{(n+1)/2} - 1$ finite, n odd (2)
- $3 \cdot 2^{n/2-1} - 1$ finite, n even (2)
- n unary

Conclusions

- Quotients provide a uniform approach

Conclusions

- Quotients provide a uniform approach
- Upper bounds for the complexity of operations

Conclusions

- Quotients provide a uniform approach
- Upper bounds for the complexity of operations
- Verifying that witnesses meet these bounds

Conclusions

- Quotients provide a uniform approach
- Upper bounds for the complexity of operations
- Verifying that witnesses meet these bounds
- A step towards a theory of complexity of languages

Conclusions

- Quotients provide a uniform approach
- Upper bounds for the complexity of operations
- Verifying that witnesses meet these bounds
- A step towards a theory of complexity of languages
- An interesting theory

Conclusions

- Quotients provide a uniform approach
- Upper bounds for the complexity of operations
- Verifying that witnesses meet these bounds
- A step towards a theory of complexity of languages
- An interesting theory
- Difficult problems

Conclusions

- Quotients provide a uniform approach
- Upper bounds for the complexity of operations
- Verifying that witnesses meet these bounds
- A step towards a theory of complexity of languages
- An interesting theory
- Difficult problems
- State complexity useful when implementing regular operations

Related work

- Combined operations, for example, KL^*

Related work

- Combined operations, for example, KL^*
- Other operations, for example, shuffle

Related work

- Combined operations, for example, KL^*
- Other operations, for example, shuffle
- Nondeterministic complexity

Related work

- Combined operations, for example, KL^*
- Other operations, for example, shuffle
- Nondeterministic complexity
- Transition complexity

Related work

- Combined operations, for example, KL^*
- Other operations, for example, shuffle
- Nondeterministic complexity
- Transition complexity
- Syntactic complexity

$$\tilde{L}OPP$$