Automata and Formal Languages - CM0081
Algebraic Laws for Regular Expressions

Andrés Sicard-Ramírez

EAFIT University

Semester 2015-2
Algebraic Laws for Regular Expressions

Definition (Equivalence of regular expressions)

Two regular expressions with variables are equivalent if whatever languages we substitute for the variables, the results of the two expressions are the same language.
Algebraic Laws for Regular Expressions

Definition (Equivalence of regular expressions)

Two regular expressions with variables are equivalent if whatever languages we substitute for the variables, the results of the two expressions are the same language.

Notation

Let $L$, $M$ and $N$ be regular expression variables.
Algebraic Laws for Regular Expressions

Definition (Equivalence of regular expressions)
Two regular expressions with variables are \textit{equivalent} if whatever languages we substitute for the variables, the results of the two expressions are the same language.

Notation
Let $L$, $M$ and $N$ be regular expression variables.

Sugar syntax
\[
L^+ \overset{\text{def}}{=} LL^*, \\
L? \overset{\text{def}}{=} \varepsilon + L.
\]
Algebraic Laws for Regular Expressions

Some laws for the union

\[(L + M) + N = L + (M + N)\]  \hspace{1cm} \text{(associativity)}

\[L + \emptyset = \emptyset + L = L\]  \hspace{1cm} \text{(identity)}

\[L + M = M + L\]  \hspace{1cm} \text{(commutativity)}

\[L + L = L\]  \hspace{1cm} \text{(idempotence)}

Remark: There is no inverse for union.
Algebraic Laws for Regular Expressions

Some laws for the concatenation

\[(LM)N = L(MN)\]  
(associativity)

\[L\varepsilon = \varepsilon L = L\]  
(identity)

\[LM \neq ML\]  
(non-commutativity)

\[L\emptyset = \emptyset L = \emptyset\]  
(\emptyset is the annihilator for concatenation)

Remark: There is no inverse for concatenation.
Algebraic Laws for Regular Expressions

Some laws for the union and the concatenation

\[ L(M + N) = LM + LN \]  
\[ (L + M)N = LN + LM \]  

(distributive)
Algebraic Laws for Regular Expressions

Some laws for the closures

\[(L^*)^* = L^*\]  
\[\emptyset^* = \varepsilon\]  
\[\varepsilon^* = \varepsilon\]  
\[(\varepsilon + L)^* = L^*\]  
\[L^* = L^+ + \varepsilon\]  
(idempotence)
Simplification of Regular Expressions

Example

\[ 0 + (\varepsilon + 1)(\varepsilon + 1)^*0 = 0 + (\varepsilon + 1)1^*0 \]
\[ = 0 + (\varepsilon1^* + 11^*)0 \]
\[ = 0 + (1^* + 11^*)0 \]
\[ = 0 + (1^* + 1^+ )0 \]
\[ = 0 + 1^*0 \]
\[ = 1^*0 \]

\[ (\varepsilon + L)^* = L^* \]

(distributive)

(Identity)

(def. \( L^+ \))

(equivalence)
Let \( E \) and \( F \) be two regular expressions with the same set of variables \( \{L_1, \ldots, L_n\} \).

To test if \( E = F \):

1. Convert \( E \) and \( F \) to concrete regular expressions \( C \) and \( D \), replacing each \( L_i \) by a different symbol \( a_i \), for \( i = 1, 2, \ldots, n \).
2. Test whether \( L(C) = L(D) \). If so, then \( E = F \), and if not \( E \neq F \).
Discovering Laws for Regular Expressions

Let $E$ and $F$ be two regular expressions with the same set of variables \{$L_1, \ldots, L_n$\}.

To test if $E = F$:

1. Convert $E$ and $F$ to concrete regular expressions $C$ and $D$, replacing each $L_i$ by a different symbol $a_i$, for $i = 1, 2, \ldots, n$.
2. Test whether $L(C) = L(D)$. If so, then $E = F$, and if not $E \neq F$.

Observation

We are proving by example!
Example
Prove or disprove that $L^* = L^* L^*$. 

1. We replace the variable $L$ by the concrete regular expression $a$.
2. $a^* \equiv a^* a^*$.
3. Because $L(a^*) = L(a^* a^*)$, we conclude that $L^* = L^* L^*$. 
Discovering Laws for Regular Expressions

Example

Prove or disprove that $L^* = L^* L^*$.

1. We replace the variable $L$ by the concrete regular expression $a$. 

Automata and Formal Languages - CM0081. Algebraic Laws for Regular Expressions
Discovering Laws for Regular Expressions

Example

Prove of disprove that $L^* = L^* L^*$.

1. We replace the variable $L$ by the concrete regular expression $a$.
2. $a^* \equiv a^* a^*$. 
Discovering Laws for Regular Expressions

Example

Prove of disprove that $L^* = L^* L^*$.

1. We replace the variable $L$ by the concrete regular expression $a$.
2. $a^* \not= a^* a^*$.
3. Because $L(a^*) = L(a^* a^*)$, we conclude that $L^* = L^* L^*$. 
Discovering Laws for Regular Expressions

Example

Prove or disprove that $L + ML = (L + M)L$. 

1. We replace the variables $L$ and $M$ by the concrete regular expressions $a$ and $b$ respectively.

2. $a + ba \not\equiv (a + b)a$.

3. $aa \notin L(a + ba)$ and $aa \in L((a + b)a)$ $\Rightarrow L(a + ba) \neq L((a + b)a)$ $\Rightarrow L + ML \neq (L + M)L$. 
Discovering Laws for Regular Expressions

Example

Prove of disprove that $L + ML = (L + M)L$.

1. We replace the variables $L$ and $M$ by the concrete regular expressions $a$ and $b$ respectively.
Discovering Laws for Regular Expressions

Example

Prove or disprove that \( L + ML = (L + M)L \).

1. We replace the variables \( L \) and \( M \) by the concrete regular expressions \( a \) and \( b \) respectively.
2. \( a + ba \neq (a + b)a \).
Discovering Laws for Regular Expressions

Example

Prove or disprove that $L + ML = (L + M)L$.

1. We replace the variables $L$ and $M$ by the concrete regular expressions $a$ and $b$ respectively.

2. $a + ba \neq (a + b)a$.

3. $aa \notin L(a + ba)$ and $aa \in L((a + b)a)$

   $\Rightarrow L(a + ba) \neq L((a + b)a)$

   $\Rightarrow L + ML \neq (L + M)L$
Example (Hopcroft, Motwani and Ullman (2007), Exercise 3.4.2.d)

Prove or disprove that \((L + M)^* M = (L^* M)^*\).
Discovering Laws for Regular Expressions

Example (Hopcroft, Motwani and Ullman (2007), Exercise 3.4.2.d)
Prove of disprove that \((L + M)^* M = (L^* M)^*\).

1. We replace the variables \(L\) and \(M\) by the concrete regular expressions \(a\) and \(b\) respectively.
Discovering Laws for Regular Expressions

Example (Hopcroft, Motwani and Ullman (2007), Exercise 3.4.2.d)

Prove or disprove that \((L + M)^* M = (L^* M)^*\).

1. We replace the variables \(L\) and \(M\) by the concrete regular expressions \(a\) and \(b\) respectively.

2. \((a + b)^* b \neq (a^* b)^*\).
Discovering Laws for Regular Expressions

Example (Hopcroft, Motwani and Ullman (2007), Exercise 3.4.2.d)

Prove or disprove that \((L + M)^* M = (L^* M)^*\).

1. We replace the variables \(L\) and \(M\) by the concrete regular expressions \(a\) and \(b\) respectively.

2. \((a + b)^* b \not\equiv (a^* b)^*\).

3. Since \(\varepsilon \not\in (a + b)^* b\) and \(\varepsilon \in (a^* b)^*\)

\[\Rightarrow (L + M)^* M \neq (L^* M)^*\]
Discovering Laws for Regular Expressions

Example (counter-example)

Extensions of the previous test beyond regular expressions may fail.

1. Add $\cap$ to the regular expression operators.
2. Test if $\mathcal{L} \cap \mathcal{M} \cap \mathcal{N} = \mathcal{L} \cap \mathcal{M}$.
3. From $\mathcal{L} = a$, $\mathcal{M} = b$ and $\mathcal{N} = c$, we conclude $\mathcal{L} \cap \mathcal{M} \cap \mathcal{N} = \mathcal{L} \cap \mathcal{M}$.
4. But if $\mathcal{L} = \mathcal{M} = a$ and $\mathcal{N} = \emptyset$, we have $\mathcal{L} \cap \mathcal{M} \cap \mathcal{N} \neq \mathcal{L} \cap \mathcal{M}$.
5. Therefore, the test is not valid!
Discovering Laws for Regular Expressions

Example (counter-example)

Extensions of the previous test beyond regular expressions may fail.

1. Add $\cap$ to the regular expression operators.
2. Test if $L \cap M \cap N = L \cap M$.
3. From $L = a$, $M = b$ and $N = c$, we conclude $L \cap M \cap N = L \cap M$.
4. But if $L = M = a$ and $N = \emptyset$, we have $L \cap M \cap N \neq L \cap M$.
5. Therefore, the test is not valid!
Derivative of a Regular Expression\(^1\)

**Definition (Derivative of a language by a symbol)**

Let \( L \subseteq \Sigma^* \) be a language and \( a \in \Sigma \) a symbol. We define \( a\backslash L \) (derivative of \( L \) by \( a \)) by

\[
a\backslash L = \{ x \in \Sigma^* | ax \in L \}.
\]

---

\(^1\)Brzozowski, Janusz A. (1964). Derivates of Regular Expressions.
Derivative of a Regular Expression\(^1\)

**Definition (Derivative of a language by a symbol)**

Let \( L \subseteq \Sigma^* \) be a language and \( a \in \Sigma \) a symbol. We define \( a \backslash L \) (derivative of \( L \) by \( a \)) by

\[
a \backslash L = \{ x \in \Sigma^* \mid ax \in L \}.
\]

**Examples**

\[
a \backslash \{abab, abba\} = \{bab, bba\},
\]

\[
a \backslash L(ab^*) = L(b^*),
\]

\[
b \backslash L(ab^*) = \emptyset.
\]

---

\(^1\)Brzozowski, Janusz A. (1964). Derivates of Regular Expressions.
Derivative of a Regular Expression

Definition (Derivative of a regular expression by a symbol)

We define recursively $a \backslash E$ (derivative of the regular expression $E$ by $a \in \Sigma$) by

- $a \backslash \emptyset = \emptyset$,
- $a \backslash \varepsilon = \emptyset$,
- $a \backslash a = \varepsilon$,
- $a \backslash b = \emptyset$ for $a \neq b$, 

- $a \backslash (E + F) = a \backslash E + a \backslash F$,
- $a \backslash (E F) = \begin{cases} (a \backslash E) F + a \backslash F & \text{if } \varepsilon \in L(E) \\ (a \backslash E) F & \text{otherwise} \end{cases}$,
- $a \backslash (E^*) = (a \backslash E) E^*$.
Derivative of a Regular Expression

Definition (Derivative of a regular expression by a symbol)

We define recursively $a \backslash E$ (derivative of the regular expression $E$ by $a \in \Sigma$) by

\[
\begin{align*}
    a \backslash \emptyset &= \emptyset, \\
    a \backslash \varepsilon &= \emptyset, \\
    a \backslash a &= \varepsilon, \\
    a \backslash b &= \emptyset \text{ for } a \neq b, \\
    a \backslash (E + F) &= a \backslash E + a \backslash F, \\
    a \backslash (EF) &= \begin{cases} 
        (a \backslash E)F + a \backslash F & \text{if } \varepsilon \in L(E), \\
        (a \backslash E)F & \text{otherwise}, 
    \end{cases} \\
    a \backslash (E^*) &= (a \backslash E)E^*.
\end{align*}
\]
Derivative of a Regular Expression

Definition (Derivative of a regular expression by a string)

We define recursively $w \backslash E$ (derivative of the regular expression $E$ by $w \in \Sigma^*$) by

$$
\varepsilon \backslash E = E, \\
a x \backslash E = a \backslash (x \backslash E).
$$
Derivative of a Regular Expression

Definition (Derivative of a regular expression by a string)
We define recursively \( w\backslash E \) (derivative of the regular expression \( E \) by \( w \in \Sigma^* \)) by

\[
\varepsilon \backslash E = E, \\
a\cdot x \backslash E = a\cdot (x \backslash E).
\]

Theorem (Brzozowski’s Theorem 4.2)

\( w \in L(E) \iff \varepsilon \in L(w\backslash E) \).
References