Regular languages

Building languages with concatenations, star-closures and unions
Motivation

As discussed in Lecture 6, building a language representation of a problem is a necessary (but not a sufficient) condition for its computational solvability.

There are at least two motivations for making this construction as structured as possible:

a. In practice, many strings are generated “on demand”, from construction rules.

b. Verifying the correctness, or establishing the “meaning” of a string requires a deconstruction. This is, identifying the sequence of operations and rules used in building it.
Induction and recursion

Building and deconstruction are step-by-step processes

• Building by **induction**
  – Strings are generated from a given finite set of basic strings, and a finite set of rules (*i.e.* relations)

• Deconstructing by **recursion**
  – The components and rules used in the generation of a given string are identified by repeatedly applying the same procedure to substrings, until the basic string components are reached
Regular languages

**Definition**: The operations of language concatenation, star-closure, and union are called regular operations.

**Definition**: A language $L$ over an alphabet $A$ is said to be regular if $L = \{\lambda\}$ or $L = \emptyset$; or either:

a. $L = \{a\}$, $a$ in $A$

b. $L = \text{ICat}(M,N)$, $M$ and $N$ regular languages

c. $L = M \cup N$, $M$ and $N$ regular languages

d. $L = M^*$, $M$ regular language
The previous definition builds not only a string but a whole language. An inductive step is a path in a tree of the form:

At the base of this induction are singletons with either symbols in the alphabet, the null string, or the empty language.
Deconstructing = recursion

The decomposition of $L$ can be visualized as a tree, as well. At each node, one asks the question: What are the regular languages and the regular operation that generated the language?

Thus, a step in the recursion corresponds to a sub-tree of the form of:

```
Regular
/   \  \
Regular Operation Regular
```

This recursion ends with singletons with either a symbol in the alphabet or the null string; or the empty language are reached.
Checking whether a language is regular

Let $Q(w): w \in \{0,1\}^* \land w \text{ ends with } 1$

Is $L = \{w: Q(w) = 1\}$ regular?

Answer: Clearly $L = lCat(\{0,1\}^*, \{1\})$.

$\{1\}$ is regular since $1 \in A = \{0,1\}$

Now, $\{0,1\}^*$ is the star-closure of $\{0,1\} = \{0\} \cup \{1\}$

$\{0\}$ is regular since $0 \in A = \{0,1\}$

Thus, $A = \{0,1\} = \{0\} \cup \{1\}$ is regular (union of regular sets)

Thus, $A^* = \{0,1\}^*$ is regular (*-closure of a regular set)

Thus, $L = lCat(A^*, \{1\})$ is regular (concatenation of regular sets)
Previous analysis cast as an inductive (generation) process

\[ L \]

Concatenation

\{0,1\}^* \quad \{1\}

Star

\{0,1\}

Union

\{0\} \quad \{1\}
Previous analysis cast as a recursive (deconstruction) process

\[
L = \{0,1\}^* \text{ Concatenation } \{1\}
\]

\[
\{0,1\} \text{ Star } \{0\} \text{ Union } \{1\}
\]
Example 2

Let $Q(w):"w$ is a string of length 3 over $\{0,1\}$"

Is $L = \{w: Q(w) = 1\}$ regular?

Answer:

$w \in L \iff (\forall i)(i \in \{1,2,3\})(\exists a_i \in A) \land w = a_1a_2a_3$

$\iff w \in A^3$

Therefore, $L = lCat(A, lCat(A, A)) = A^3$

From the previous example: $A$ is regular

Then, $lCat(A, A) = A^2$ is regular and thus,

$A^3 = lCat(A, A^2)$ is regular
Regularity of finite languages

**Theorem 7.1**: Finite languages are regular languages

**Proof:**

Let $L$ be a finite language. Then, there exist a nonnegative integer $n$, such that

$L = \{ w_i : (\exists i)(i \in \{1,\ldots,n\} \land w_i \in A^*) \}$. Thus, $L$ can be written as

$$L = \bigcup_{i=1}^{n} \{ w_i \}$$

Since union is a regular operation, all that is left to do is to show that

$$(\forall i)(i \in \{1,\ldots,n\})\{ w_i \}$$

is a regular language. Now, since $w_i \in A^*$:

$$(\forall i)(i \in \{1,\ldots,n\})(\exists n(i) \in N)(\forall j \in \{1,\ldots,n(i)\})(\exists a_j^{(i)} \in A \land w_i = a^{(i)}_1 \cdots a^{(i)}_{n(i)})$$

Therefore,

$$(\forall i)(i \in \{1,\ldots,n\})w_i = lCat(\{a^{(i)}_1\}, \cdots, lCat(\{a^{(i)}_{n(i)-1}\}, \{a^{(i)}_{n(i)}\}) \cdots)$$

Now, $(\forall i)(i \in \{1,\ldots,n\})(\forall j \in \{1,\ldots,n(i)\})(a_j^{(i)} \in A \Rightarrow \{a_j^{(i)}\} \text{ regular})$

Thus, $(\forall i)(i \in \{1,\ldots,n\})\{ w_i \}$ is a concatenation of regular languages and therefore, $\{ w_i \}$ is indeed a regular language.
Corollaries

The next statements follow directly from Theorem 7.1 and the definition of regular language

**Corollary 7.1**: The star-closure of a finite language is a regular language

**Corollary 7.2**: If $A$ is an alphabet, then $A^*$ is a regular language
Regular expressions and algebra

At the base of a construction or at the end of the analysis of a regular language is a collection of singletons of the form:

\{a\}, a \in A \text{ or } \{\lambda\}, \text{ or } \phi

This suggests constructing or analyzing with symbols in the alphabet, lambda and phi instead of sublanguages.

**Definition:** Let \(A\) be an alphabet. Then, a **regular expression** over \(A\) is a string of elements of \(A\), left and right parenthesis, and the symbols:

<table>
<thead>
<tr>
<th>SEMANTIC</th>
<th>SYMBOL</th>
</tr>
</thead>
<tbody>
<tr>
<td>Concatenation of symbols</td>
<td>(\circ)</td>
</tr>
<tr>
<td>Union of symbols</td>
<td>+</td>
</tr>
<tr>
<td>Star closure of a symbol</td>
<td>*</td>
</tr>
</tbody>
</table>

Inductively defined by:

a. The expressions

\(R = a, (\forall a \in A) \lor R = \lambda \lor R = \phi\)

are all regular

b. If \(R\) and \(S\) are regular expressions,

\((R), R + S, R \circ S, \text{ and } R *\) are regular expressions
Examples and counter examples

1. $R = (0+1)^* \circ 1$
   
is a regular expression

2. $R = (0+1) \circ (0+1) \circ (0+1)$
   
is a regular expression.

3. $R = (0+1)^3$
   
is NOT a regular expression!!!!
Operations hierarchy

To avoid using too many parenthesis it is assumed that the operations have the following hierarchy (do first), ordered from left to right:

– **Do first** *, then ᵇ, and then +

**Example:** $(0 + (1 \diamond (0^*)))$ is replaced with $0 + 1 \diamond 0^*$
The language of a regular expression

A regular expression describe a unique set of strings (i.e. formal language), which turns out to be a regular language

**Definition**: Let $R$ be a regular expression, then the language described by $R$ is denoted $L(R)$ and called **language of** $R$
The language of a regular expression - Examples

Example 1:

\[ L \left((0 + 1)^* \diamond 1 \right) = \{ w : w \in \{0,1\}^* \land w \text{ ends with } 1 \} \]

Example 2: As a matter of fact, regular expressions provide clues to the languages they represent. For example: a string in the language represented by

\[ L(1 + 0 \circ 1^*) \]

is clearly read as either 1, or 0, or a string that start with a 0 followed by a string of 1’s. This is,

\[ L(1 + 0 \circ 1^*) = \{1\} \cup \{ w : w = 01^n \land n \geq 0 \} \]
Operations for building or analyzing regular languages

Let $R$ and $S$ be regular expressions. Then,

\[
L(\emptyset) = \emptyset;
\]
\[
L(\lambda) = \{ \lambda \};
\]
\[
L(a) = \{ a \};
\]
\[
L(R + S) = L(R) \cup L(S);
\]
\[
L(R \circ S) = L(R)L(S);
\]
\[
L(R^*) = (L(R))^*
\]
Finding the language

Let’s use the previous operation rules to find a language from its regular expression:

\[
L(1 + 0 \cdot 1^*) = L(1) \cup L(0 \cdot 1^*)
\]
\[
= L(1) \cup lCat(L(0), L(1^*))
\]
\[
= L(1) \cup lCat(L(0), (L(1))^*)
\]
\[
= \{1\} \cup lCat(\{0\}, \{1\}^*)
\]
\[
= \{1\} \cup \{0, 01, 011, 0111, \ldots\}
\]
\[
= \{1, 0, 01, 011, 0111, \ldots\}
\]}
Further properties: + properties

Let $R$, $S$ and $T$ be regular expressions. Then,

\[ R + S = S + R; \]
\[ R + \emptyset = \emptyset + R = R; \]
\[ R + R = R; \]
\[ (R + S) + T = R + (S + T) \]
properties

Let $R$, $S$ and $T$ be regular expressions. Then,

$$R \circ \emptyset = \emptyset \circ R = \emptyset;$$

$$R \circ \lambda = \lambda \circ R = R;$$

$$(R \circ S) \circ T = R \circ (S \circ T)$$
Distributive properties

Let $R$, $S$ and $T$ be regular expressions. Then,

\[ R \circ (S + T) = (R \circ S) + (R \circ T); \]
\[ (S + T) \circ R = (S + T) \circ (T + R) \]
Closure properties (some of them)

Let $R$ and $S$ be regular expressions. Then,

\[
\begin{align*}
\phi^* &= \lambda^* = \lambda; \\
R^* &= (R^*)^* = R^*R^*; \\
R^*R &= RR^* = R^*; \\
(R+S)^* &= (R^*+S^*)^* = (R^*S^*)^* = (R^*S)^*R^* = R^*(SR^*)^*
\end{align*}
\]
Some seemingly different regular expressions are indeed the same

By applying the previously discussed operational rules it is often possible to reduce the usually long regular expressions that appears in practice. This is important when it comes to language identification

Example 1:

\[ L(0^* \circ (1 \circ 0^*)^*) = L((0 + 1)^*) \]

Example 2:

\[ L((0 + 1)^* + (0^* \circ 1^*)^* + 0^* \circ (1 \circ 0^*)^*) = L((0 + 1)^*) \]
Some general principles

**Theorem 7.1:** Every regular language admits a regular expression

**Proof:**
Let $n$ be the number of regular operations involved in constructing $L$. We proceed by induction on $n$.

- **Base case:** $n = 0$. In this case $L = \{\lambda\} \vee L = \phi \vee (\exists a \in A) \land L = \{a\}$. Then, $L$ is represented either by $\lambda \lor \phi \lor a$, respectively. Therefore, the statement is true for $n = 0$.
- **Hypothesis:** Assume that the statement is true for some $n \geq 1$.
- **This is, assume that:**

$$P(n): (\exists n \geq 0) \land L \text{ is built with at most } n \text{ regular operations} \implies L \text{ admits a regular expression}$$
Proof (cont.)

Let \( L \) be a regular language built with at most \( n + 1 \) regular operations. Then, \((\exists M, N)\) regular languages built with at most \( n \) regular operations, such that either:

\[
L = M \cup N \vee L = lCat(M, N) \vee L = M^*
\]

Since \( M \) and \( N \) are built with at most \( n \) regular operations, by the inductive hypothesis:

\[(\exists R, S)(R \land S \text{ regular expressions}) \land (M = L(R) \land N = L(S))\]

Therefore, \( L \) admits either: \( R + S \lor R \circ S \lor R^* \) as regular expression
A second theorem

Theorem 7.2: $L(R)$ is infinite if and only if $R$ contains at least one star operation over a non-null, non-empty expression.

Proof:
Assume first that $R$ contains $S^*$ with $S$ regular $\land S \neq \lambda \land S \neq \phi$. Then, $L(S^*) = L(S)^*$ is an infinite countable language. Then, there is an infinite sequence $s_j \in L(S)^*(\forall j \in N)$. Since $S^*$ is part of the regular expression $R$, $(\forall j \in N)(s_j \in L(R) \lor (\exists u_j \in L(R) \land s_j$ is a substring of $u_j))$. In each case, $L(R)$ contains an infinite string, and therefore, $L(R)$ is infinite.
Proof (cont.)
Assumenow that \( R \) does not contain the star of a regularexpression
Then, \( R \) is a string of regular expressions and the symbols + or \( \circ \).
To demonstrate that \( L(R) \) is finite we proceed by induction on the
number \( n \) of operationsin the string.
If \( n = 0 \), by hypothesis \( R = a \) for some \( a \in A \). Thus, \( L(R) = \{ a \} \) is finite.
Inductivehypothesis: \( \exists n \in N \)(\( n > 0 \land (\forall j) 0 \leq j \leq n \)) \( R \) has \( j \) or \( \circ \) operations \( \Rightarrow L(R) \) finite.
Inductivehesis: If \( R \) has \( n+1 \) \( \circ \) or + operations, then \( L(R) \) is finite.
Proof : If \( R \) has \( n+1 \) \( \circ \) or + operations then \( R \) decomposes
either as :
\( R = S \circ T \lor R = S + T \) where \( S \) and \( T \) have at most \( n \) \( \circ \) or + operations
By theinductivehypothesis \( L(S) \) and \( L(T) \) are finite. Now, by general
principles:
\( L(R) = L(S \circ T) = lCat(S,T) \lor L(R) = L(S + T) = L(S) \cup L(T) \).
In thefirst case, the number of elementsin \( L(R) \) is the product of
the number of elementsin \( L(S) \) by thenumber of elementsin \( L(T) \).
In thesecond case, the number of elementsin \( L(R) \) is at most thesum
of the elementsin \( L(S) \) and \( L(T) \). In both cases \( L(R) \) is finite.