

REGULAR EXPRESSIONS

CHAPTER 1.3

Outline

- ◇ Examples of
- ◇ Formal Definition of Regular Expressions
- ◇ Equivalence of Regular Expressions and FAs

Introduction

In arithmetic, use operations: $+$, $-$, $*$, $/$ to build expressions of values:

$$(4 + 8)/2 - 1$$

We can do the same with regular languages and operations on them. That is, we can build an expression whose value is a language:

$$(0 \cup 1)0^*$$

This is called a *regular expression*.

What language is this? $(0 \cup 1)$ shorthand for $(\{0\} \cup \{1\}) = \{0, 1\}$.

0^* means $\{0\}^* = \{\epsilon, 0, 00, 000, \dots\}$

NOTE: This is *clearly* a language. Why?

Introduction (cont)

$(0 \cup 1)0^*$ means $(0 \cup 1) \circ 0^*$ so we have

$$\{0, 1\} \circ \{\epsilon, 0, 00, 000, \dots\} =$$

$$\{0, 1, 00, 10, 000, 100, \dots\}$$

NOTE: $\{0, 1\} \circ \{\epsilon\} = \{0, 1\}$ Check the NFA!

$\{0, 1\} \circ \{\} = \{\}$ Check the NFA!

Example

$$(0 \cup 1)^* = \{0, 1\}^* = \{\epsilon, 0, 1, 00, 01, 10, 11, \dots\}$$

Introduction (cont)

If $\Sigma = \{a, b, c\}$, we can talk about Σ as a language of all strings length one over the alphabet Σ .

Therefore, Σ^* = the language of all strings over the alphabet Σ including ϵ .
(NOTE: This is an infinite language)

Σ^*c = the language of all string over Σ that end in a c

$(a\Sigma^*) \cup (\Sigma^*b)$ = the language of all strings over Σ that start with an a or end with a b.

Order of Operations: *, \circ , \cup , parentheses change order

Formal Definition of a Regular Expression

Defn: R is a **regular expression** if R is

1. a where $a \in \Sigma$, Σ an alphabet
2. ϵ - the language $\{\epsilon\}$
3. \emptyset - the language with no strings in it $\{\}$
4. $(R_1 \cup R_2)$ where R_1 and R_2 are regular expressions
5. $(R_1 \circ R_2)$ where R_1 and R_2 are regular expressions
6. R_1^* where R_1 is a regular expression

NOTE: This is an inductive definition.

NOTE: A regular expression R denotes a language just like an arithmetic expression denotes a number.

Defn: If R a regular expression, then $L(R)$ is the language that R denotes.

Examples

Example $\Sigma = \{0, 1\}$

1. $L = \{\omega \mid \omega \text{ has exactly a single } 1 \text{ but possibly other symbols}\} = 0^*10^*$
2. $L = \{\omega \mid \omega \text{ has at least one } 1\} =$
3. $L = \{\omega \mid \omega \text{ contains the substring } 001\} =$
4. $L = \{\omega \mid \omega \text{ is even length}\} =$
5. $L = \{\omega \mid \omega \text{ the length of } \omega \text{ is a multiple of } 3\} =$
6. $L = \{01, 10\} =$
7. $L = \{\omega \mid \omega \text{ starts with and ends with the same symbol}\} =$
8. $L = \{\omega \mid \omega \text{ starts with and ends with the same symbol and there is at least one character in between}\} =$

Some Regular Expression Identities

1. $\epsilon\epsilon = \epsilon$

2. $R\epsilon = R$

3. $\epsilon R = R$

4. $R\emptyset = \emptyset$

5. $\emptyset R = \emptyset$

6. $\emptyset^* = \{\epsilon\}$

7. $R \cup \epsilon = R$ iff $\epsilon \in R$

Example

Let $D = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$

We want a language to express floats.

Examples: 72, 3.14159, +7., -.01

Equivalence of Regular Expressions with FAs

Reminder: A regular language is one recognized by some FA.

Theorem: A language is regular iff some regular expression denotes it.

Lemma 1: If some regular expression denotes a language, then the language is regular.

Lemma 2: If a language is regular, then some regular expression denotes it.

Lemma 1: Proof Sketch

Assume we have a regular expression R denoting language A .

Show how to convert R into an NFA recognizing A .

By Corollary in 1.2, if an NFA recognizes a language, then it is regular.

Equivalence of Regular Expressions with FAs

Lemma 1: Proof: Show how to convert R into an NFA recognizing A .

Consider each of the six cases in the formal definition of a regular expression R .

1. If R is a (where $a \in \Sigma$, the alphabet):

Then $L(R) = \{a\}$, and the following NFA recognizes this language:

2. If R is ϵ :

Then $L(R) = \{\epsilon\}$, and the following NFA recognizes this language:

3. If R is “nothing”

Then $L(R) = \{\} = \emptyset$, and the following NFA recognizes this language:

Equivalence of Regular Expressions with FAs

Lemma 1: Proof: Show how to convert R into an NFA recognizing A .

Consider each of the six cases in the formal definition of a regular expression R (cont).

4. If $R = R_1 \cup R_2$ where R_1 and R_2 are regular expressions, and $L(R_1)$ and $L(R_2)$ are regular by one of the previous cases, then the $L(R_1 \cup R_2)$ is regular by Theom. (Regular languages are closed under \cup .)
5. If $R = R_1 \circ R_2$ where R_1 and R_2 are regular expressions, and $L(R_1)$ and $L(R_2)$ are regular by one of the previous cases, then the $L(R_1 \circ R_2)$ is regular by Theorem. (Regular languages are closed under \circ .)
6. If $R = R_1^*$ where R_1 is a regular expression, and $L(R_1)$ is regular by one of the previous cases, then the $L(R_1^*)$ is regular by Theorem. (Regular languages are closed under $*$.)

Equivalence of Regular Expressions with FAs

Theorem: A language is regular iff some regular expression denotes it.

Lemma 2: If a language is regular, then some regular expression denotes it.

NOTE: This direction is “ugly”.

Proof idea: Assume the “if” part, that is, we have language A and it is regular. Therefore, by definition, a DFA, say M_1 , recognizes it.

We must show how to convert M_1 into a regular expression.

Hard to do! Convert the DFA M_1 to a *Generalized Nondeterministic Finite Automaton* (GNFA) and then convert the GNFA to a regular expression.

DFA \rightarrow GNFA \rightarrow regular expression

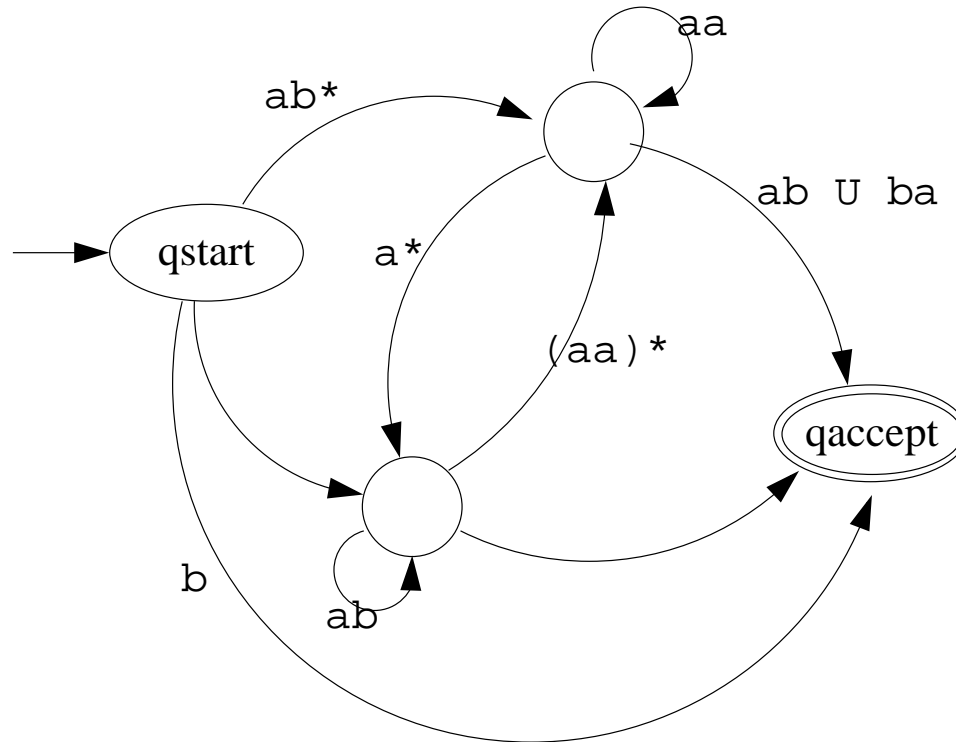
GNFAs

In a GNFA,

- transition arcs may be labelled with regular expressions in addition of elements of Σ and ε .
- if there is no arc between two states, we represent this explicitly by adding an arc labelled \emptyset .
- the start state has arcs going to every other state but no arcs coming in.
- there is a single accept state. It has arcs coming in from every other state. The accept state \neq start state.
- all other states each have an arc going to every other state (except the start state) and also an arc from every other state to itself.

GNFAs

Example:

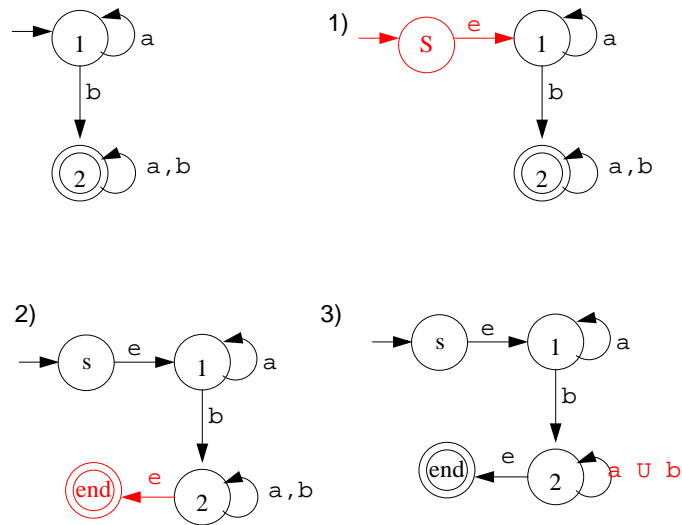


GNFAs

To convert a DFA to a GNFA with this special form:

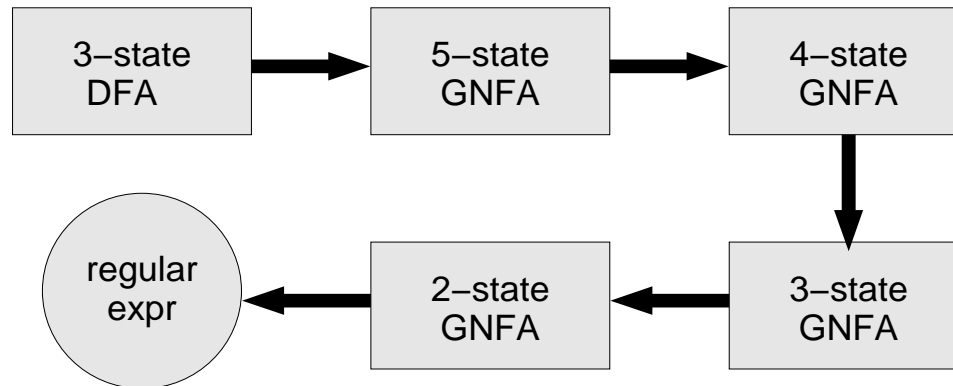
1. Add a new start state with ε arc to the old start state.
2. Add a new accept state with ε arcs to the old accept states.
3. If any arcs have multiple labels or multiple arcs, replace with one arc that is the union of the two previous labels. (regular exp)

Example:



Equivalence of Regular Expressions with FAs

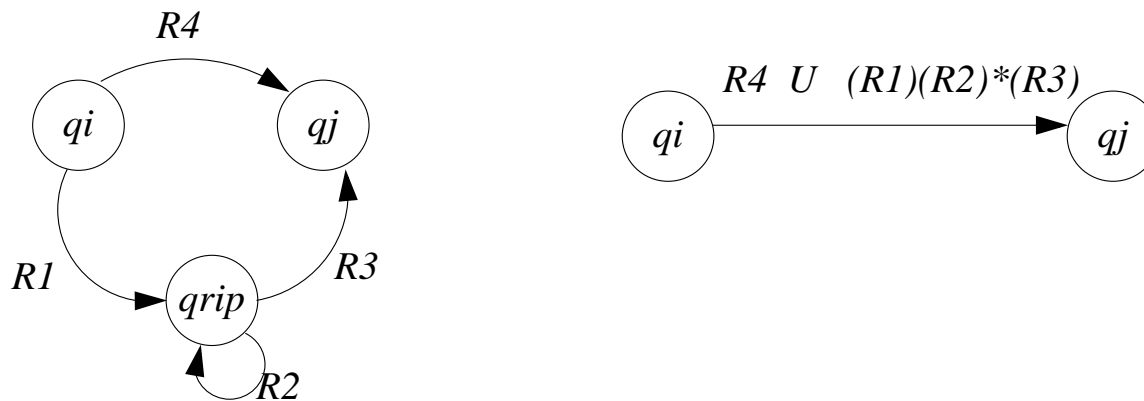
The idea for transforming a DFA to a regular expression:



Equivalence of Regular Expressions with FAs

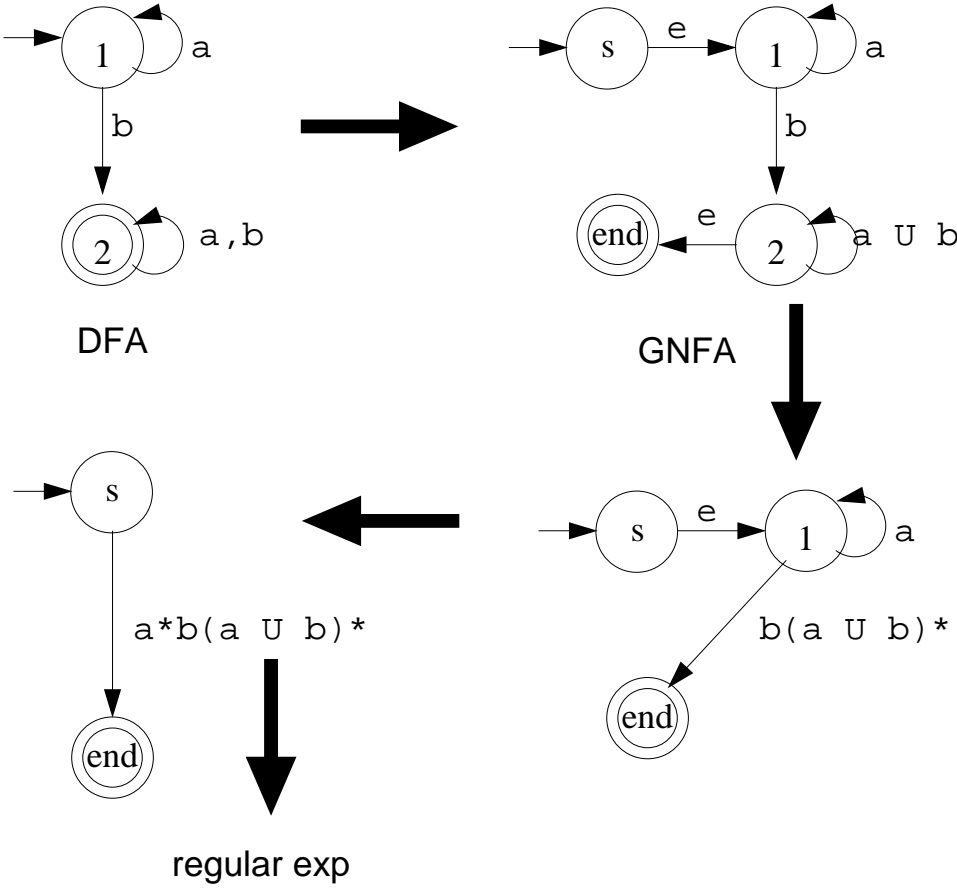
Select one state at a time to rip out of the GNFA until we are down to a single arc between the start state and the accept state.

Example of ripping out a state:



Equivalence of Regular Expressions with FAs

Our example:



Equivalence of Regular Expressions with FAs

Another example:

