

Inf2A: Kleene's Theorem

Stuart Anderson

School of Informatics
University of Edinburgh

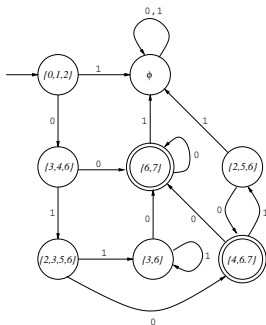
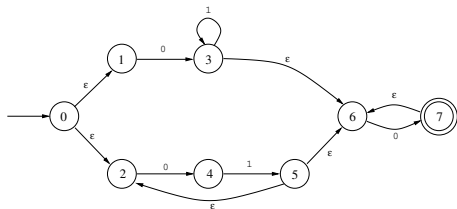
October 9, 2009



- Born 1909, Died 1994.
- Worked for most of his career at Univ of Wisconsin-Madison
- Significant contributor to the development of recursion theory (foundations of computation).
- Developed regular algebra.
- Contributions to Intuitionistic Mathematics.

Non-Deterministic Finite Automata

Conversion Theorem: Every language recognised by an *NFA* (with or without ϵ -transitions) is regular.



Examples

- `ls a*`
(lists all names of files in the current directory starting with a)
- `egrep '(.*aba.*) | (.*x.*y.*)'`
`/usr/dict/words`
(extracts all lines from the file `/usr/dict/words` that either contain `aba` or an `x` and after that a `y`).

Defining a Language

We take the following steps:

- 1 We define the syntax of the language by giving a grammar that defines the language. This is often called an *abstract syntax* for the language.
- 2 We say how the language will be interpreted, i.e. what kind of things the terms in the language will stand for.
- 3 We provide a mapping from the language to the things they stand for. Usually we give one rule for each rule in the abstract syntax.
- 4 Here we have the syntax of regular expressions, they are interpreted as standing for sets of finite strings, and we provide a map from regular expressions to sets of strings.

Syntax of Regular Expressions

Regular expressions over Σ are produced using the following formation rules:

- ε is a regular expression.
- \emptyset is a regular expression.
- Every symbol a in Σ is a regular expression.
- If R and S are regular expressions then so is $R + S$.
- If R and S are regular expressions then so is RS .
- If R is a regular expression then so is R^* .

Let $\Sigma = \{a, b, c\}$.

- $R_1 = ab + abc$
- $R_2 = (a(b + c))^*$
- $R_3 = (aaa)^* + (aaaaa)^*$

Semantics of Regular Expressions

With every regular expression R over the alphabet Σ we associate a language $L(R) \subseteq \Sigma^*$ (**syntax**, **semantics**):

- $L(\varepsilon) = \{\varepsilon\}$.
- $L(\emptyset) = \emptyset$.
- $L(a) = \{a\}$.

Now suppose that R and S are regular expressions and that we have already defined the languages $L(R)$ and $L(S)$. We define $L(R + S)$, $L(RS)$, and $L(R^*)$ as follows:

- $L(R + S) = L(R) \cup L(S)$,
- $L(RS) = \{xy \mid x \in L(R), y \in L(S)\}$,
- - $L(R^*) = \{\varepsilon\} \cup \{x \mid x \in L(R)\} \cup \{x_1x_2 \mid x_1, x_2 \in L(R)\} \cup \dots$
 $= \{x_1 \dots x_n \mid n \in \mathbb{N}_0, x_1, \dots, x_n \in L(R)\}$.

Examples

Let $\Sigma = \{a, b, c\}$.

- $R_1 = ab + abc$

$$L(R_1) = \{ab, abc\}$$

- $R_2 = (a(b + c))^*$

$$\begin{aligned} L(R_2) &= \{ax_1ax_2ax_3 \dots ax_n \mid n \in \mathbb{N}_0, x_1, \dots, x_n \in \{b, c\}\} \\ &= \{\varepsilon, ab, ac, abab, abac, acab, acac, ababab, \dots\}. \end{aligned}$$

- $R_3 = (aaa)^* + (aaaaa)^*$

$$L(R_3) = \{a^n \mid n \in \mathbb{N}_0 \text{ divisible by 3 or 5}\}$$

Proposition 1: For every regular expression R there exists an NFA with ε -transitions N such that

$$L(R) = L(N).$$

The *proof* is by induction on the structure of the regular expression R . For every regular expression R we construct an NFA with ε -transitions N , following the rules that were used to construct R .

Induction on the structure of regular expressions

To show that some proposition $P(R)$ is true for all regular expressions we often use induction on the structure of regular expressions. To do this, we show: The base cases:

- $P(\emptyset)$ is true
- $P(\varepsilon)$ is true
- $P(a)$ is true for each $a \in \Sigma$

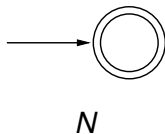
Then we assume that $P(R)$ and $P(S)$ are both true and we show that:

- $P(R + S)$ is true
- $P(RS)$ is true
- $P(R^*)$ is true

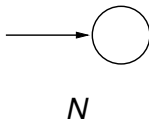
Example to try now - show for all regular expressions R , if R has no subexpression of the form S^* then $L(R)$ is a finite set.

The base cases

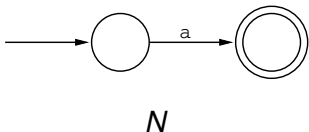
- $R = \varepsilon$



- $R = \emptyset$

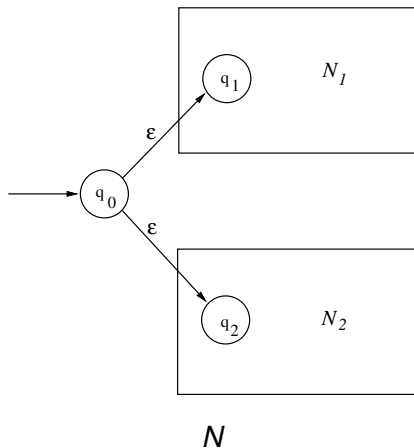


- $R = a$



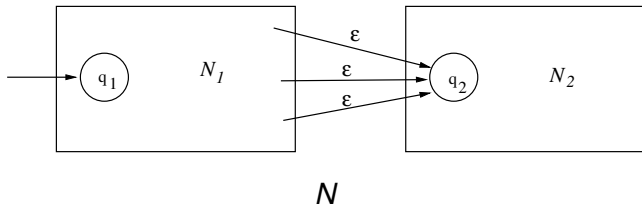
Addition

- $R = R_1 + R_2$



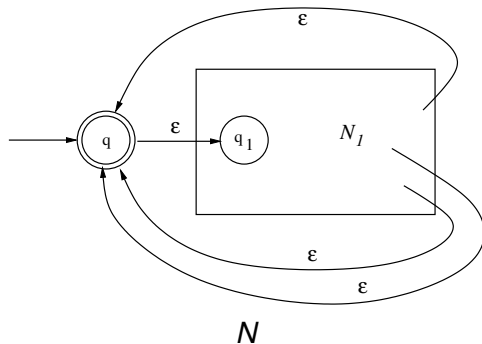
Concatenation

- $R = R_1 R_2$



Kleene Star - (Asterate in Kozen)

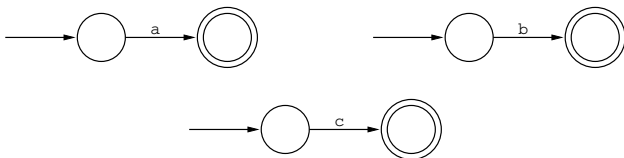
- $R = R_1^*$



Example

$$R = (a(b + c))^* a$$

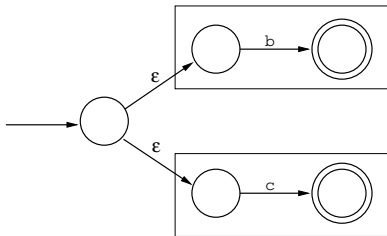
NFAs with ϵ -transitions for a , b , c :



Example (cont'd)

$$R = (a(b + c))^* a$$

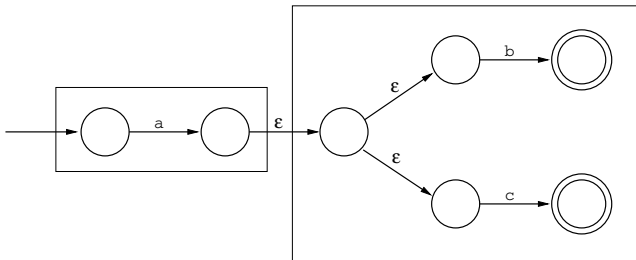
An NFA with ϵ -transitions for $b + c$:



Example (cont'd)

$$R = (a(b + c))^* a$$

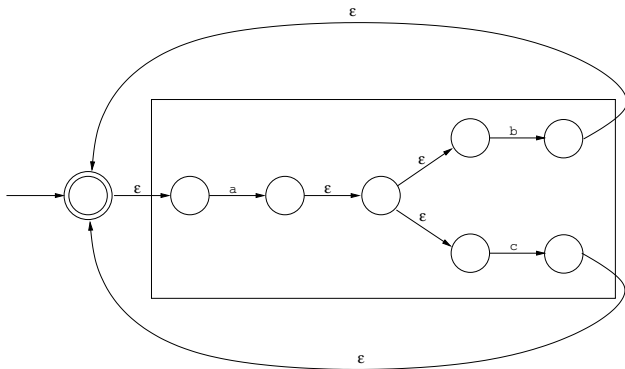
An NFA with ϵ -transitions for $a(b + c)$:



Example (cont'd)

$$R = (a(b + c))^* a$$

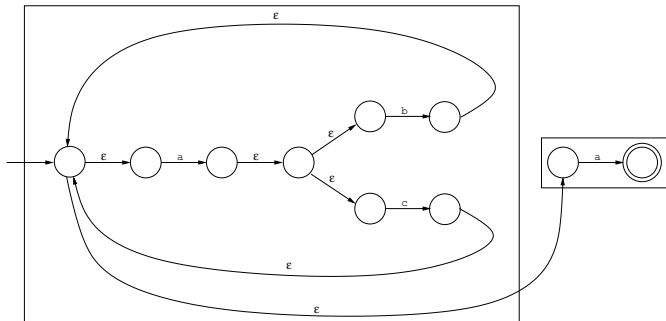
An NFA with ϵ -transitions for $(a(b + c))^*$:



Example (concluded)

$$R = (a(b + c))^* a$$

An NFA with ϵ -transitions for $(a(b + c))^* a$:



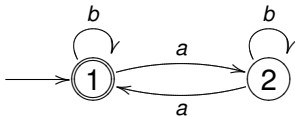
From FSAs to Regular Expressions

Proposition 2: For every FSM M there exists a regular expression R such that $L(M) = L(R)$.

- We construct a set of equations with one variable R_i for each state i of M
- For each variable we construct a sum of terms derived as follows:
 - If state i is accepting then ε is one of the summands
 - If there is a transition labelled a from state i to state j , then aR_j is one of the summands
 - We solve using the equations of regular algebra plus the observation that A^*B is a solution to equations of the form $R = AR + B$ (try it!!)
- The regular expression we derive for the initial state q_0 satisfies $L(R_{q_0}) = L(M)$

Example: From FSA to Regular Expression

- Consider the FSM:



- We construct the following equations:

$$R_1 = \varepsilon + bR_1 + aR_2 \quad (1)$$

$$R_2 = bR_2 + aR_1 \quad (2)$$

- Solving for R_2 : $R_2 = b^* aR_1$
- Substitute for R_2 :
 $R_1 = \varepsilon + bR_1 + ab^* aR_1 = (b + ab^* a)R_1 + \varepsilon$
- Solve for R_1 : $R_1 = (b + ab^* a)^* \varepsilon = (b + ab^* a)^*$

$$R = AR + B$$

- We claim a solution of $R = AR + B$ is A^*B .
- Substitute the solution into the RHS of the equation
 $A(A^*B) + B = AA^*B + \varepsilon B = (AA^* + \varepsilon)B$
- We know that from the definition of A^* that $A^* = AA^* + \varepsilon$,
so $(AA^* + \varepsilon)B = A^*B$ and we are done.
- A^*B is the smallest set solving this equation (try taking anything out of it!)

Summary

- Taken together Propositions 1 and 2 make Kleene's theorem, which tell us that every regular expression has a corresponding FSA that recognizes the language of the regular expression and every FSA has a regular expression whose language is exactly that recognised by the FSA.
- This tells us that despite looking quite different FSAs and REs define the same set of languages and we can use them interchangeably.
- Having two different representations is often useful. For example proving something by induction on the structure of the RE form of a regular language is often easier than trying to consider all possible FSMs and sometimes (e.g. in proving the pumping lemma) the finiteness of the stateset makes a proof easier than it would be with REs.