MA/CSSE 474

Theory of Computation

Regular Expressions

Questions?

- Homework
- Tomorrow's Exam material
  (no NFA->DFA proof on this exam)
- Reading
- Anything else

The Myhill-Nerode Theorem

Theorem: A language is regular iff the number of equivalence classes of $\approx_L$ is finite.

Proof: *S*how the two directions of the implication:

L regular → the number of equivalence classes of $\approx_L$ ***is finite:*** If *L* is regular, then there exists some FSM *M* that accepts *L*. *M* has some finite number of states *m*. The cardinality of $\approx_L \leq m$. So the cardinality of $\approx_L$ is finite.

The number of equivalence classes of $\approx_L$ ***is finite → L regular:*** If the cardinality of $\approx_L$ is finite, then the construction that was described in the proof of the previous theorem will build an FSM that accepts *L*. So *L* must be regular.

Q1

Summary

- Given any regular language *L*, there exists a minimal DFSM *M* that accepts *L*.
- *M* is unique up to the naming of its states.
- Given any DFSM *M*, there exists an algorithm *minDFSM* that constructs a minimal DFSM that also accepts *L*(*M*).

Canonical Forms

A ***canonical form*** for some set of objects C assigns exactly one representation to each class of "equivalent" objects in C.

Further, each such representation is distinct, so two objects in C share the same representation iff they are "equivalent" in the sense for which we define the form.

A Canonical Form for FSMs

buildFSMcanonicalform(M: FSM) =

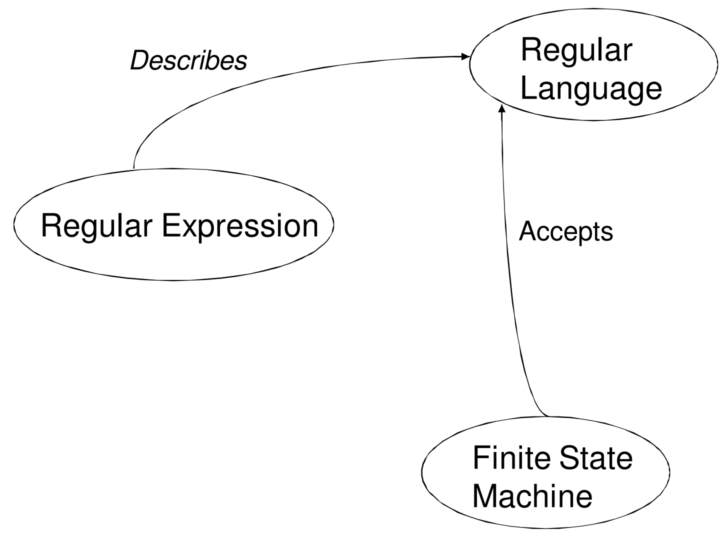
1. M' = *ndfsmtodfsm*(M).
2. M^* = *minDFSM*(M').
3. Create a unique assignment of names to the states of M^*.
4. Return M^*.

Given two FSMs M_1 and M_2:

buildFSMcanonicalform(M_1)
=
buildFSMcanonicalform(M_2)

iff $L(M_1) = L(M_2)$.

Regular Languages



Regular Expressions

The regular expressions over an alphabet Σ are the strings that can be obtained as follows:

1. $\varnothing$ is a regular expression.
2. ε is a regular expression.
3. Every element of Σ is a regular expression.
4. If α , β are regular expressions, then so is $\alpha\beta$.
5. If α , β are regular expressions, then so is $\alpha\cup\beta$.
6. If α is a regular expression, then so is α^*.
7. α is a regular expression, then so is α^+.
8. If α is a regular expression, then so is (α).

Regular Expression Examples

If $\Sigma = \{a, b\}$, the following are regular expressions:

$\varnothing$
ε
a
$(a \cup b)^*$
$abba \cup \varepsilon$

Regular Expressions Define Languages

Define *L*, a **semantic interpretation function** for regular expressions (Let α and β be arbitrary regular expressions over alphabet Σ.

1. $L(\varnothing) = \varnothing$.
2. $L(\varepsilon) = \{\varepsilon\}$.
3. *If* $c \in \Sigma$, $L(c) = \{c\}$.
4. $L(\alpha\beta) = L(\alpha)\ L(\beta)$.
5. $L(\alpha \cup \beta) = L(\alpha) \cup L(\beta)$.
6. $L(\alpha^*) = (L(\alpha))^*$.
7. $L(\alpha^+) = L(\alpha\alpha^*) = L(\alpha)\ (L(\alpha))^*$. If $L(\alpha)$ is equal to $\varnothing$, then $L(\alpha^+)$ is also equal to $\varnothing$. Otherwise $L(\alpha^+)$ is the language that is formed by concatenating together one or more strings drawn from $L(\alpha)$.
8. $L((\alpha)) = L(\alpha)$.

The Role of the Rules

- Rules 1, 3, 4, 5, and 6 give the language its power to define sets.
- Rule 8 has as its only role grouping other operators.
- Rules 2 and 7 appear to add functionality to the regular expression language, but they don't.

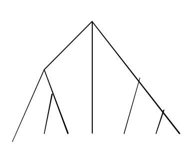
2. ε is a regular expression.

7. α is a regular expression, then so is α^+.

Q2

Operator Precedence in Regular Expressions

	Regular Expressions	Arithmetic Expressions
Highest	Kleene star	exponentiation
↕	concatenation	multiplication
Lowest	union	addition



$a\ b^* \cup c\ d^*$ $\quad$ $x\ y^2 + i\ j^2$

Analyzing a Regular Expression

$$L((\mathtt{a} \cup \mathtt{b})^*\mathtt{b}) = L((\mathtt{a} \cup \mathtt{b})^*)\ L(\mathtt{b})$$

$$= (L((\mathtt{a} \cup \mathtt{b})))^*\ L(\mathtt{b})$$

$$= (L(\mathtt{a}) \cup L(\mathtt{b}))^*\ L(\mathtt{b})$$

$$= (\{\mathtt{a}\} \cup \{\mathtt{b}\})^*\ \{\mathtt{b}\}$$

$$= \{\mathtt{a}, \mathtt{b}\}^*\ \{\mathtt{b}\}.$$

Examples

$L(\ \mathtt{a}^*\mathtt{b}^*\ ) =$

$L(\ (\mathtt{a} \cup \mathtt{b})^*\ ) =$

$L(\ (\mathtt{a} \cup \mathtt{b})^*\mathtt{a}^*\mathtt{b}^*\ ) =$

$L(\ (\mathtt{a} \cup \mathtt{b})^*\mathtt{abba}(\mathtt{a} \cup \mathtt{b})^*\ ) =$

Going the Other Way

$L = \{w \in \{\mathtt{a}, \mathtt{b}\}^*: |w| \text{ is even}\}$

L = $\{w \in \{\mathtt{0}, \mathtt{1}\}^*$: w is a binary representation of a multiple of 4}

$L = \{w \in \{\mathtt{a}, \mathtt{b}\}^*$: w contains an odd number of a's}

Q3-5

Hidden: Going the Other Way

$L = \{w \in \{\mathtt{a}, \mathtt{b}\}^*: |w| \text{ is even}\}$

$(\mathtt{a} \cup \mathtt{b})\ (\mathtt{a} \cup \mathtt{b}))^*$

$(\mathtt{aa} \cup \mathtt{ab} \cup \mathtt{ba} \cup \mathtt{bb})^*$

L = $\{w \in \{\mathtt{0}, \mathtt{1}\}^*$: w is a binary representation of a multiple of 4}

$0 \cup 1(0 \cup 1)^*00$

$L = \{w \in \{\mathtt{a}, \mathtt{b}\}^*$: w contains an odd number of a's}

$\mathtt{b}^*\ (\mathtt{ab}^*\mathtt{ab}^*)^*\ \mathtt{a}\ \mathtt{b}^*$

$\mathtt{b}^*\ \mathtt{a}\ \mathtt{b}^*\ (\mathtt{ab}^*\mathtt{ab}^*)^*$

The Details Matter

$a^* \cup b^* \neq (a \cup b)^*$

$(ab)^* \neq a^*b^*$

More Regular Expression Examples

$L($ $(aa^*) \cup \varepsilon$ $) =$

$L($ $(a \cup \varepsilon)^*$ $) =$

$L = \{w \in \{a, b\}^*$: there is no more than one b in $w\}$

$L = \{w \in \{a, b\}^*$: no two consecutive letters in w are the same$\}$

Q6-7

The Details Matter

L_1 = {$w \in$ {a, b}* : every a is immediately followed a b}

A regular expression for L_1:

A FSM for L_1:

L_2 = {$w \in$ {a, b}* : every a has a matching b somewhere}

A regular expression for L_2:

A FSM for L_2:

Kleene's Theorem

Finite state machines and regular expressions define the same class of languages.

To prove this, we must show:

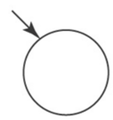
***Theorem**:* Any language that can be defined by a regular expression can be accepted by some FSM and so is regular.

Theorem: Every regular language (i.e., every language that can be accepted by some DFSM) can be defined with a regular expression.

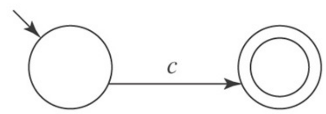
For Every Regular Expression There is a Corresponding FSM

We'll show this by construction. An FSM for:

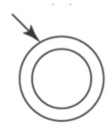
$\varnothing$:



A single element of Σ:

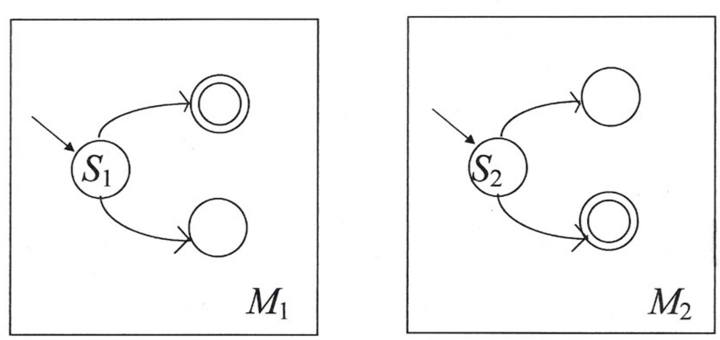


ε ($\varnothing$*):



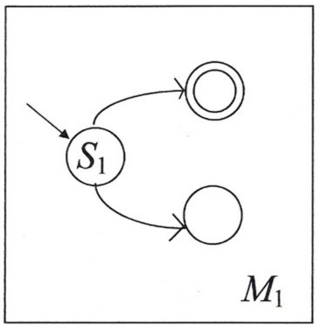
Union

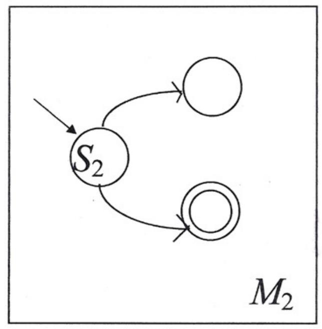
If α is the regular expression $\beta \cup \gamma$ and if both $L(\beta)$ and $L(\gamma)$ are regular:



Concatenation

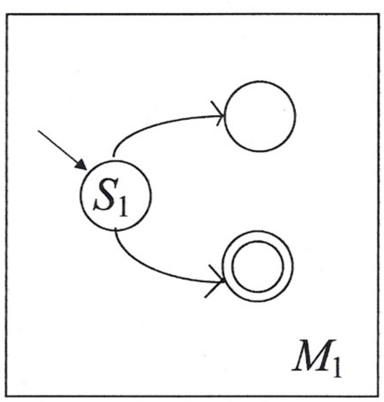
If α is the regular expression $\beta\gamma$ and if both $L(\beta)$ and $L(\gamma)$ are regular:





Kleene Star

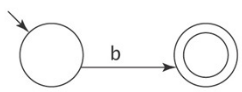
If α is the regular expression β^* and if $L(\beta)$ is regular:



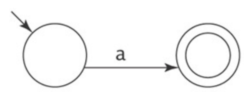
An Example

(b ∪ ab)*

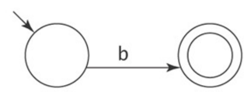
An FSM for b



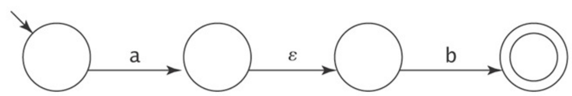
An FSM for a



An FSM for b



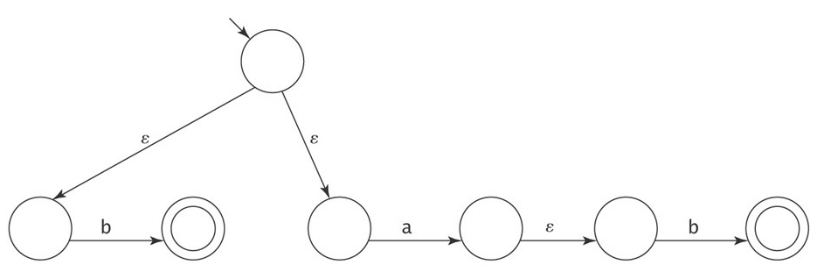
An FSM for ab:



An Example

(b ∪ ab)*

An FSM for (b ∪ ab):



An Example

(b ∪ ab)*

An FSM for (b ∪ ab)*: