MA/CSSE 474 Exam 1 Notation and Definitions page Name _______________ (turn this in with your exam)

Unless specified otherwise, r,s,t,u,v,w,x,y,z are strings over alphabet Σ; while a, b, c, d are individual alphabet symbols.

DFSM notation: M= $(K, \Sigma, \delta, s, A)$, where:

K is a finite set of ***states***, Σ is a finite ***alphabet***

$s \in K$ is start state, $A \subseteq K$ is set of ***accepting states***

$\delta: (K \times \Sigma) \rightarrow K$ is the ***transition function***

Extend δ's definition to $\delta: (K \times \Sigma^*) \rightarrow K$ by the recursive definition $\delta(q, \varepsilon)=q$, $\delta(q, xa) = \delta(\delta(q, x), a)$

M accepts w iff $\delta(s, w) \in A$. $L(M) = \{w \in \Sigma^* : \delta(s, w) \in A\}$

Alternate notation:

(q, w) is a *configuration* of M. (current state, remaining input)

The *yields-in-one-step* relation: $|\text{-}_M$:

(q, w) $|\text{-}_M$ (q', w') iff $w = a\,w'$ for some symbol $a \in \Sigma$, and $\delta(q, a) = q'$

The *yields-in-zero-or-more-steps* relation: $|\text{-}_M{}^*$ is the reflexive, transitive closure of $|\text{-}_M$.

A ***computation*** by M is a finite sequence of configurations $C_0, C_1, \ldots, C_n$ for some $n \geq 0$ such that:

- C_0 is an initial configuration,
- C_n is of the form (q, ε), for some state $q \in K_M$,
- $\forall i \in \{0, 1, \ldots, n\text{-}1\}$ $(C_i$ $|\text{-}_M$ $C_{i+1})$

M **accepts** w iff the state that is part of the last step in w is in A.

A language L is **regular** if L=L(M) for some DFSM M.

In an **NDFSM**, the function δ is replaced by the relation Δ: $\Delta \subseteq (K \times (\Sigma \cup \{\varepsilon\})) \times K$

ndfsmtodfsm(*M*: NDFSM) =
1. For each state *q* in K_M do:
   1.1 Compute *eps*(*q*).
2. *s' = eps*(*s*)
3. Compute δ':
   3.1 active-states = {*s'*}.
   3.2 $\delta' = \varnothing$.
   3.3 While there exists some element *Q* of *active-states* for which δ' has not yet been computed do:
      For each character *c* in Σ_M do:
         new-state = $\varnothing$.
         For each state *q* in *Q* do:
            For each state *p* such that $(q, c, p) \in \Delta$ do:
               new-state = *new-state* $\cup$ *eps*(*p*).
         Add the transition (*q*, *c*, *new-state*) to δ'.
         If *new-state* $\notin$ *active-states* then insert it.
4. *K'* = *active-states.*
5. $A' = \{Q \in K' : Q \cap A \neq \varnothing\}$.

Some functions over languages:

maxstring(*L*) =

$\{w \in L: \forall z \in \Sigma^* (z \neq \varepsilon \rightarrow wz \notin L)\}$.

chop(*L*) =

$\{w : \exists x \in L\ (x = x_1 c x_2,\ x_1 \in \Sigma_L^*,\ x_2 \in \Sigma_L^*,\ c \in \Sigma_L,$
$|x_1| = |x_2|, \text{ and } w = x_1 x_2)\}$.

firstchars(*L*) =

$\{w : \exists y \in L\ (y = cx \wedge c \in \Sigma_L \wedge x \in \Sigma_L^* \wedge w \in \{c\}^*)\}$.

Equivalent strings relative to a language: Given a language L, two strings w and x in Σ_L^* are *indistinguishable* with respect to L, written $w \approx_L x$, iff $\forall z \in \Sigma^* (xz \in L \text{ iff } yz \in L)$.

[x] is a notation for "the equivalence class that contains the string x".

The construction of a minimal-state DSFM based on $\approx_L$:

$M = (K, \Sigma, \delta, s, A)$, where K contains n states, one for each equivalence class of $\approx_L$.

$s = [\varepsilon]$, the equivalence class containing ε under $\approx_L$,

$A = \{[x] : x \in L\}$,

$\delta([x], a) = [xa]$.

Enumerator (generator) for a language:When it is asked, enumerator gives us the next element of the language. Any given element of the language will appear within a finite amount of time. It is allowed that some may appear multiple times.

Recognizer: Given a string s, recognizer halts and accepts s if s is in the language. If not, recognizer either halts and rejects s or keeps running forever. This is a **semidecision procedure**. If recognizer is guaranteed to always halt and (accept or reject) no matter what string it is given as input, it is a **decision procedure**.