**REGULAR/NON REGULAR LANGUAGES**

**Closure:** Union, Intersection, Concat, \*, Homomorph, Complemnet, Reversal, Set Diff, EndsWith, Prefix, Suffix, (A\B) ∪(B\A)◊

**DFA**

**Formal:** (Q, Σ, δ, q0, F); Q = states finite; Σ = Alphabet; δ: Q X Σ -> Q transition function; q0 ∈ Q initial; F ⊆ Q final states

**NFA**

**Formal:** (Q, Σ, δ, q0, F); Q = states finite; Σ = Alphabet; δ: Q X Σepsilon  -> P(Q) transition function; q0 ∈ Q initial; F ⊆ Q final states

NFA = DFA

**Regex**

*1. a* for some *a* in Σ; 2. ε; 3. ∅; 4. (R1 ∪ R2), R1& R2 are regex; 5. (R1 ° R2), R1& R2 are regex; 6. (R1\*) R1 is regex;

R∪∅ = R; R°ε = R

**Pumping Lemma:**

1. Assume L is reg with pl p; 2. Pick w ∈ L |w| >= p; 3. Let x,y,z be arbitrary s.t. xyz = w, |xy| <= p, |y| > 0; 4. Pick i ≠ 1 s.t. xyiz ∉ L; 5. L not pump so not reg

**CONTEXT FREE LANGUAGES**

**Closure:** Union, Concat, \*, Intersect w/reg, Homomorph, CFL\REG

**CFG**

**Formal:**(V,Σ,R,S); V = variables; Σ = terminals; R = rules; S∈V start

**CNF**

**Formal:**1. S->ε; 2. A->BC; 3. A->a

1. Add new start var; 2. Replace all rules w/length > 2 with rules w/length = 2 ; 3. For each rule of the form A->ε replace accordingly B->A => B->ε, B->AA => B->A|ε, B->xA | Ax => B->x; 4. Remove the rules of the form A->B and for each B->u add A->u; 5. For each a ∈ Σ add rule of the form A->a

**PDA**

**Formal:**(Q, Σ, Γ, δ, q0, F); Q set of states; Σ input alphabet; Γ stack alphabet; δ: Q X Σepsilon X Γepsilon -> P(Q X Γepsilon); q0 ∈ Q initial; F ⊆ Q final states

**TURING MACHINES**

**Formal:**(Q, Σ, Γ, δ, q0, qaccept, qreject); Q set of states; Σ input alphabet; Γ tape alphabet; δ: Q X Γ -> Q X Γ X {L, R};q0 ∈ Q initial; qaccept ∈ Q accept state; qreject ∈ Q reject state;

**Decidable:** ADFA, ANFA, EQDFA, EDFA, ACFG, ECFG,AREX

**Undecidable:** EQCFG, ATM, HALTTM, ETM, EQTM, ALLCFG

**RE:** {Decidable}, {Undecidable}, ATM

**Not RE:** ETM, EQTM

**CO-RE:** {Decidable}, ETM

**Not CO-RE:** ATM, EQTM

**RE Closure:** Union, Concat, \*, Intersection, Homomorph

Language decidable iff RE and CO-RE;

**Mapping Reducibility:**

If A <=m B and B is decidable then, A is decidable.

If A <=m B and A is undecidable then B is undecidable.

If A <=m B and B is RE then A is RE.

To show A is undecidable; 1. Pick an undecidable language; 2. Build a TM T that takes “instances” of B and produces “instances” of A.

EX: ATM <=m EQTM

T:<M,w> |-> <M,M’>

Map Reduce EX: To see that T is undecidable we can show that ATM <=m T. Let y∈Σ\* be an arbitrary string s.t. y ≠ yR. The computable mapping, we want is <M, w> |-> <M’> where: M’ = “On input x; 1. If x = y, accept; 2. If x = yR run M on w and accept as M does; 3. Otherwise reject.” Note M’ will always accept y, and if w∈L(M) M will also accept yR. No other stings are accepted. If <M,w> ∈ ATM , then M accepts w so L(M’) = {y,yR} and <M’> ∈T. If <M,w> ∉ ATM , then M doesn’t accepts w so L(M’) = {y} and <M’> ∉ T