MA/CSSE 474

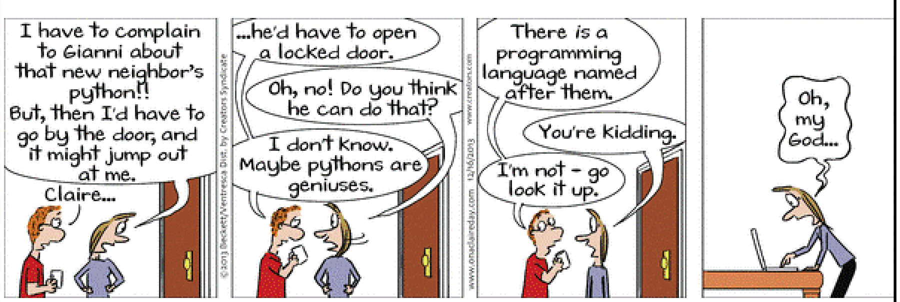
Theory of Computation

Proofs of several things

(as much as we have time for)

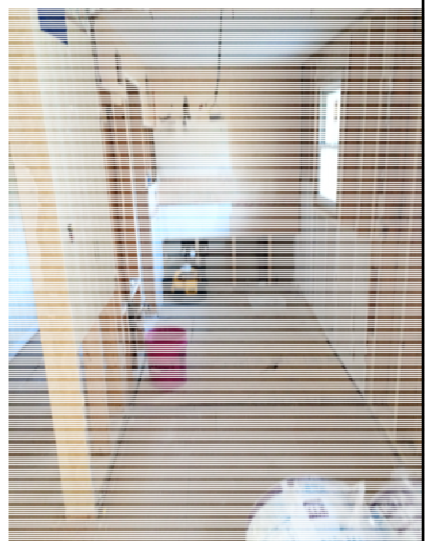
Your Questions?

- Previous class days' material
- Reading Assignments
- HW5 problems
- Tuesday's Exam
- Anything else



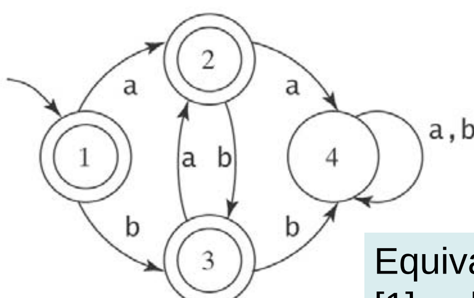
My life these days

Luke Thomas
Agapie
Born 3/18/2018
Grandchild #11



Example (continued)

L = { $w \in$ {a, b}* : no two adjacent characters are the same }



Equivalence classes of $\approx_L$:

[1] [ε]

[2] [a, aba, ababa,

[3] [b, ab, bab, abab, …]

[4] [aa, abaa, ababb…]

Lower bound on number of states

***Theorem*:** Let M be a DFSM that accepts the regular language L. The number of states in M is greater than or equal to the number of equivalence classes of $\approx_L$.

***Proof*:**

1. Suppose that the number of states in M were less than the number of equivalence classes of $\approx_L$.
2. Then, by the pigeonhole principle, there must be at least one state q that "contains" strings from more than one equivalence classes of $\approx_L$.
3. But then M's future behavior on those strings will be identical, which is not consistent with the fact that they are in different equivalence classes of $\approx_L$.

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

The number of equivalence classes of $\approx_L$ ***is finite → L regular:*** If the cardinality of $\approx_L$ is finite, then

NDFSMtoDFSM Correctness

We will probably not have time
to finish this in class;
we will do as much as we can.
Details are in the textbook
(Appendix C)

The Algorithm *ndfsmtodfsm*

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. *$A' = \{Q \in K' : Q \cap A \neq \varnothing\}$.*

Correctness Proof of *ndfsmtodfsm*

To prove:

From any NDFSM $M = (K, \Sigma, \Delta, s, A)$, *ndfsmtodfsm* constructs a DFSM $M' = (K', \Sigma, \delta', s', A')$, which is equivalent to M.

$K' \subseteq \mathcal{P}(K)$ (a.k.a. 2^K)

$s' = eps(s)$

$A' = \{Q \subseteq K : Q \cap A \neq \emptyset\}$

Union

$\delta'(Q, a) = \cup\{eps(p): p \in K$ and $(q, a, p) \in \Delta$ for some $q \in Q\}$

Correctness Proof of *ndfsmtodfsm*

From any NDFSM *M*, *ndfsmtodfsm* constructs a DFSM *M'*, which is:

(1)Deterministic: By the definition in step 3 of δ', we are guaranteed that δ' is defined for all reachable elements of *K'* and all possible input characters. Further, step 3 inserts a single value into δ' for each state-input pair, so *M'* is deterministic.

(2) **Equivalent to *M*:** We constructed δ' so that *M'* mimics an "all paths" simulation of *M*. We must now prove that that simulation returns the same result that *M* would.

A Useful Lemma

Lemma: Let w be any string in Σ^*, let p and q be any states in K, and let P be any state in K'. Then:

$$(q, w) \vdash_M^* (p, \varepsilon) \textbf{ iff } ((eps(q), w) \vdash_{M'}^* (P, \varepsilon) \text{ and } p \in P) \ .$$

INFORMAL RESTATEMENT OF LEMMA: In other words, the original NDFSM M starts in state q and, after reading the string w, can land in state p (along at least one of its paths)
iff
the new DFSM M' must behave as follows:

Furthermore, the only-if part implies:

A Useful Lemma

Lemma: Let w be any string in Σ^*, let p and q be any states in K, and let P be any state in K'. Then:

$$(q, w) \vdash_M^* (p, \varepsilon) \textbf{ iff } ((eps(q), w) \vdash_{M'}^* (P, \varepsilon) \text{ and } p \in P)$$

Recall: NDFSM $M = (K, \Sigma, \Delta, s, A)$, DFSM $M' = (K', \Sigma, \delta', s', A')$,

It turns out that we will only need this lemma for the case where q = s, but the more general form is easier to prove by induction. This is common in induction proofs.

Proof: We must show that δ' has been defined so that the individual steps of M', when taken together, do the right thing for an input string w of any length. Since the definitions describe one step at a time, we will prove the lemma by induction on $|w|$.

Base Case: |w| = 0, so w = ε

- *if* part: Prove:

  $(eps(q), \varepsilon) \vdash_{M'}^{*} (P, \varepsilon) \wedge p \in P \longrightarrow (q, \varepsilon) \vdash_{M}^{*} (p, \varepsilon)$

Base Case

- *only if* part: We need to show:

  $(q, \varepsilon) \vdash_{M}^{*} (p, \varepsilon) \rightarrow [\ (eps(q), \varepsilon) \vdash_{M'}^{*} (P, \varepsilon)$ and $p \in P\ ]$

Induction Step

Let w have length $k + 1$. Then w = zc where $z \in \Sigma^*$ has length k, and $c \in \Sigma$.

Induction assumption. The lemma is true for z.

So we show that, assuming that *M* and *M′* behave identically for the first *k* characters, they behave identically for the last character also and thus for the entire string of length $k + 1$.

The Definition of δ'

$\delta'(Q, a) = \bigcup\{eps(p) : \exists q \in Q\ ((q, a, p) \in \Delta)\}$

What We Need to Prove

The relationship between:

- The computation of the NDFSM *M*:

$$(q, w) \vdash_M^* (p, \varepsilon)$$

and

- The computation of the DFSM *M′*:

$$(eps(q), w) \vdash_{M'}^* (P, \varepsilon) \text{ and } p \in P$$

What We Need to Prove

Rewriting *w* as *zc*:

- The computation of the NDFSM *M*:

$$(q, zc) \mid\!\!-_M^* (p, \varepsilon)$$

and

- The computation of the DFSM *M'*:

$$(eps(q), zc) \mid\!\!-_{M'}^* (P, \varepsilon) \text{ and } p \in P$$

What We Need to Prove

Breaking w into two pieces:

- The computation of the NDFSM *M*:

$$(q, zc) \mid\!\!-_M^* (s_i, c) \mid\!\!-_M^* (p, \varepsilon)$$

and

- The computation of the DFSM *M'*:

$$(eps(q), zc) \mid\!\!-_{M'}^* (Q, c) \mid\!\!-_{M'} (P, \varepsilon) \text{ and } p \in P$$

In other words, after processing z, M will be in some set of states S, whose elements we write as s_i. M' will be in some "set" state that we call Q. Again, well split the proof into two parts:

In the M derivation above, the second $\mid\!\!-_M$ has a * due to the possibility of epsilon moves. In the M' derivation there is no * because of no epsilon moves in a deterministic machine.

If Part

We must prove:

$$(eps(q), zc) \mid\!-_{M'}^{*} (Q, c) \mid\!-_{M'} (P, \varepsilon) \text{ and } p \in P \;\rightarrow$$
$$(q, zc) \mid\!-_{M}^{*} (s_i, c) \mid\!-_{M}^{*} (p, \varepsilon)$$

Only If Part

We must prove:

$$(q, zc) \mid\!-_{M}^{*} (s_i, c) \mid\!-_{M}^{*} (p, \varepsilon) \rightarrow$$
$$(eps(q), zc) \mid\!-_{M'}^{*} (Q, c) \mid\!-_{M'} (P, \varepsilon) \text{ and } p \in P$$

Back to the Theorem

If $w \in L(M)$ then:

- The original machine M, when started in its start state, can consume w and end up in an accepting state.

- ($eps(s)$, w) $|\text{-}_{M'}{}^*$ (Q, ε) for some Q containing some *state* $r \in A$.
  - In the statement of the lemma, let q equal s and $p = r$ for some $r \in A$.
  - Then M', when started in its start state, $eps(s)$, will consume w and end in a state that contains r.
  - But if M' does that, then it has ended up in one of its accepting states (by the definition of A' in step 5 of the algorithm).
  - So M' accepts w (by the definition of what it means for a machine to accept a string).

Back to the Theorem 2

If $w \notin L(M)$ (i.e. the original NDFSM does not accept w):

- The original machine M, when started in its start state, will not be able to end up in an accepting state after reading w.

- If ($eps(s)$, w) $|\text{-}_{M'}{}^*$ (Q, ε), then Q contains no state $r \in A$. This follows directly from the lemma.

The two cases, taken together, show that M' accepts exactly the same strings that M accepts.