Finite Automata

Non-Regular Languages

One of the questions we can now attempt to answer has to do with the existence of **non-regular languages**.

Ex. 2.54: prove that $L = \{xx^R \mid x \in \{0, 1\}^*\}$ is not regular (R denotes reversal).

Soln.: We prove that $\mathrm{Index}(R_L) = \infty$.

Let $x, y \in 0^*1$, with $x \neq y$. Then $xx^R \in L$ and $yx^R \notin L$. This implies that any two different strings in 0^*1 belong to two different equivalence classes. And this implies that the number of equivalence classes is infinite.

Finite Automata

Non-Regular Languages

Ex. 2.55: $L = \{0^m1^n \mid \gcd(m, n) = 1\}$ is not regular.

Soln.: the prime number theorem states that there are infinitely many primes; furthermore any two primes are relatively prime (= their $\gcd$ is 1). Let p and q be distinct primes. Then $0^p1^p \notin L$ and $0^q1^p \in L$, and 0^p and 0^q do not belong to the same equivalence class. Thus $\mathrm{Index}(R_L) = \infty$.

Ex. 2.58: $\{0^p \mid p \text{ is a prime}\}$ is not a regular language.

Soln.: attempt. The prime number theorem states that there are infinitely many primes. Consider two string of 0s, x and y, $x \neq y$. We have to show that the number of classes $[x]_{R_L}$ of $\Sigma^* = 0^*$ is infinite.

$$x\, R_L\, y \quad \Leftrightarrow [\forall z \in \Sigma^*, xz \in L \Leftrightarrow yz \in L]$$
$$\Leftrightarrow [\forall z \in \Sigma^*, xz \text{ has prime length} \Leftrightarrow yz \text{ has prime length}]???$$

Finite Automata

Pumping Lemmas

As it turns out - we will see a proof of this later - most languages are not regular. It would be nice to have several techniques to attempt the proof that some specific language is not regular: the index method seems to lead to very complex analyses (except for relatively trivial cases), and may thus be "too hard" for general use. One of the results we obtained used the fact that, for a DFA of cardinality n, any string that is longer than n and is accepted by it must complete at least one cycle in the DFA graph. From this string we must be able to create arbitrarily long strings accepted by the DFA and containing multiple successive copies of the substring responsible for the cycle. If this process were to lead to strings that are clearly not in the language we would have to conclude that the language is not regular.

We move to formalize these ideas.

Finite Automata

Pumping Lemmas

Lemma 2.57. (Pumping Lemma - weak form). If a language L is accepted by a DFA M with s states, then every string $x \in L$ with $|x| \geq s$ can be written as $x = uvw$ such that $v \neq \varepsilon$ and $uv^*w \subseteq L$.

Proof. A string with at least s "labels" must correspond to a path with the same number of arcs (and labels), connecting $s + 1$ states (nodes). Since M has s states, at least one state, say, q_i, must appear twice: $q_0q_1...q_{i-1}q_iq_{i+1}... q_{j-1}q_iq_{j+1}... q_f$, where q_0 is the start state and q_f is a final state. Let u be the substring providing the labels for $q_0q_1...q_{i-1}q_i$, let v be the substring providing the labels for $q_{i+1}... q_{j-1}q_i$ and let w be the substring providing the labels for $q_{j+1}... q_f$. Then $x = uvw$, $v \neq \varepsilon$, and elision or arbitrary finite repetition of v will still provide a member of L: $\delta(q_0, uv^*w) = \delta(\delta(\delta(q_0, u), v^*), w)$.

Finite Automata

Pumping Lemmas - Example

Ex. 2.58: $\{0^p \mid p \text{ is a prime}\}$ is not a regular language.

Soln.: Contradiction - **all** of these proofs are by contradiction...

Assume $L = \{0^p \mid p \text{ is a prime}\}$ is regular. The L is accepted by a DFA M with s states. Consider a prime $p > s$. Since $0^p \in L$ and $|0^p| = p > s$, the pumping lemma guarantees that $0^p = uvw$, with $v \neq \varepsilon$ and $uv^*w \subseteq L$. Let $i = |u| + |w|$, and $j = |v|$. $uv^*w \subseteq L$ implies that, for any $k \geq 0$, $uv^kw = 0^{i+kj} \in L$. Equivalently, for every $k \geq 0$, $i + kj$ is a prime. In particular it is a prime for $k = 0$, and thus i is a prime, which implies that that $i \geq 2$. When $k = i$, this means that $i(1 + j)$ is prime. But $j = |v| \geq 1$, and thus $i(1 + j)$ cannot be a prime. Contradiction.

NOTE: the pumping lemmas CANNOT be used to prove a language regular - they can only be used to prove it NOT regular, and ONLY using proof by contradiction.

Finite Automata

Pumping Lemmas

Since the form of the pumping lemma just given provides no control over where the "middle string" v is located in the original string (requiring care with cases), we observe that any substring of the original string which is long enough, must also contain cycles.

Lemma 2.59. (Pumping Lemma - strong form) Let L be accepted by a DFA M with s states. Let α be a string of L with $|\alpha| \geq s$. Assume one can rewrite $\alpha = xyz$, with $|y| \geq s$. Then y can be written as $y = uvw$ such that $v \neq \varepsilon$ and $xuv^*wz \subseteq L$.

Proof. Almost identical to the previous one: y must provide labels for a cycle. Use them to either eliminate the cycle or to "pump it up".

Finite Automata

Pumping Lemmas - Example

Ex. 2.60: $L = \{0^n1^n \mid n \geq 0\}$ is not regular.

Soln.: By contradiction. Assume it regular, and thus accepted by a DFA M with s states. Let $\alpha = 0^s1^s \in L$. Choose $x = \varepsilon$, $y = 0^s$, $z = 1^s$, where $|y| = |0^s| = s$. The Pumping Lemma states that α can be written as $\alpha = xuvwz$, with $v \neq \varepsilon$ and $xuv^*wz \subseteq L$. This implies that, for any $k \geq 0$, $xuv^kwz \in L$. When $k = 0$: $xuwz = 0^{s-|v|}1^s \in L$. Since $v \neq \varepsilon$, s-$|v| < s$: contradiction.

Note: using the weak form of the Pumping Lemma, we would have $\alpha = uvw$, with $uv^*w \subseteq L$. Then v could be: a) all 0s; b) some 0s followed by some 1s; c) all 1s. We would have to show a contradiction for ALL 3 cases separately. Proving that $\text{Index}(L) = \infty$ is, perhaps, easier.

Finite Automata

Pumping Lemmas - Example

Ex. 2.61: $L = \{\beta\beta^R \mid \beta \in \{0, 1\}^+\}$ is not regular.

Soln.: By contradiction. Assume L regular, M an accepting DFA with s states. Select $\alpha = 0^s110^s \in L$. Let $x = \varepsilon$, $y = 0^s$, $z = 110^s$. The pumping lemma implies that $y = uvw$, s.t. $v \neq \varepsilon$, and $xuv^*wz \subseteq L$. This gives that, for any $k \geq 0$, $xuv^kwz \in L$. For $k = 0$: $xuwz = 0^{s-|v|}110^s \in L$. When $|v|$ is odd, the length of this string is odd; when $|v|$ is even, the first half of the string contains two 1s, while the second half contains none. Contradiction.

Finite Automata

Pumping Lemmas - Example

Ex. 2.63: $L = \{0^n 1 0^m 1 0^p 1 0^q \mid n, m, p \geq 1, q \equiv nm \pmod{p}\}$ is not regular.

Soln.: Assume L regular, $L = \mathrm{L(M)}$, $|\mathrm{M}| = s$. Consider the string $\alpha = 010^{s+1}10^{s+1}10^{s+1}$. Apply the "strong pumping lemma" to α, with $x = 010^{s+1}10^{s+1}10$, $y = 0^s$ and $z = \varepsilon$. Then y can be written as $y = 0^s = uvw$, s.t. $v \neq \varepsilon$ and $xuv^kw \in L\ \forall\ k \geq 0$. Let $k = 0$. $xuw = 010^{s+1}10^{s+1}10^t$, with $1 \leq t \leq s$. Since $t\ (\neg \equiv)\ 1(s+1) (\mathrm{mod}\ s+1) \equiv 0\ (\mathrm{mod}\ s + 1)$, contradiction.

Finite Automata

Pumping Lemmas - Example

Ex. 2.67: $L = \{\mathrm{w} \in \{0, 1\}^* \mid \#_0(w) \neq \#_1(w)\}$ is not regular.

Soln. 1. Let $\alpha = 0^s 1^{(s!)+s}$, $x = \varepsilon$, $y = 0^s$, $z = 1^{(s!)+s}$. The pumping lemma gives $y = uvw$, with $v \neq \varepsilon$ and $xuv^kwz = 0^{s+(k-1)|v|}1^{(s!)+s} \notin L$ when $k = (s!)/|v| + 1$. (since $|v| \leq |y| = s$, k must be an integer.)

Soln. 2.

1. $L_1 = \{0^n 1^n \mid n \geq 0\}$ is not regular (easy to prove).
2. $L_2 = \{w \in \{0, 1\}^* \mid \#_0(w) = \#_1(w)\}$ is not regular, since $L_2 \cap 0^*1^* = L_1$.
3. L is not regular since $\overline{L} = L_2$ is not regular.

Finite Automata

Pumping Lemmas - Example

Ex. 2.68: $L = \{a^n b^m c^k \mid n, m, k \geq 0, n \neq m \text{ or } m \neq k \text{ or } k \neq n\}$ is not regular.

Soln. It should be easy to see that $\overline{L} \cap a^*b^*c^* = \{a^n b^n c^n \mid n \geq 0\}$. One can use the strong form of the pumping lemma to prove that this new language is not regular (just take a string with $n >$ number of states of the automaton and "pump" the substring of a^n implied by the strong form of the pumping lemma). We also know that the complement of a regular language is regular. Thus, if L were regular, $\overline{L}$ would have to be also regular, implying that $\overline{L} \cap a^*b^*c^*$ is regular. Contradiction.

Finite Automata

Pumping Lemmas - Advice

How do we apply the Pumping Lemmas?

We concentrate on the "strong form", since the weak form is just a special case.

1). The proof will always be by contradiction: assume the language L is regular and prove that the Pumping Lemma implies that L contains strings that it can't possibly contain.

- If L is regular there must exist a DFA with a finite number of states - say s - which accepts the strings in L and rejects all others. You need not know exactly what s is, just that it exists.
- Take a string α, with $|\alpha| \geq s$. Picking the right string may not be trivial, and may require thought and more than one try. If needed, you can rewrite α as $\alpha = xyz$, with $|y| \geq s$ - just make sure the string y occurs "in the right place": that's where you will "pump".

Finite Automata

Pumping Lemmas - Advice

2). At this point the further decompositions of the string are out of your control.

- The Pumping Lemma gives that y can be rewritten as $y = uvw$, with $1 \leq |v| \leq s$. It further says that $xuv^*wy \subseteq L$ ($xuv^nwy \in L$, $\forall\ n \geq 0$).
- The problem is that v can fall anywhere in y - you can make no assumptions.
- The only thing you can do is enumerate all the different cases that can occur and show that for each case at least some of the strings xuv^nwy are not in L. This is the only way you can reacquire control of the process.
- If you miss even a single case, you have proven nothing.

Finite Automata

Pumping Lemmas - Example

Ex. 2.61, revisited: $L = \{\beta\beta^R \mid \beta \in \{0, 1\}^+\}$ is not regular.

Soln.: By contradiction.

1. Assume L regular, M an accepting DFA with s states.
2. Select $\alpha \in L$. What α do we select? We could choose $\alpha = 0^{2n}$, where $2n \geq s$.
   1. The Pumping Lemma (either form, really) will give that $\alpha = xyz$, with $|y| \geq 1$, and $xy^nz \in L\ \forall\ n \geq 0$. The only way to generate a string not in L is if $|y|$ is an odd number. If $|y|$ is even, you get only strings in L. Why?
   2. The Pumping Lemma gives only that $1 \leq |y| \leq s$, but no information about $|y|$ being even or odd: dead end.

Finite Automata

Pumping Lemmas - Example

3. Re-select $\alpha \in L$. We must find some way to separate β from β^R: use 1s to separate the 0s. $\alpha = 0^n110^n$, where $2n+2 \geq s$ and $\beta = 0^n1$. The Pumping Lemma (weak form) will give that $\alpha = xyz$, with $1 \leq |y| \leq s$, and $xy^nz \in L\ \forall\ n \geq 0$. We look at all the cases.
   1. y could could be a string of 0s in the left half of α: removing it or pumping it up will give a different number of 0s before the 1s as after the 1s, thus violating the "mirror image" requirement.
   2. y could be a string of 0s in the right half of α and the same argument as before will give us strings not in L.
   3. y could contain one or both 1s. This has a number of subcases:
      1. $y = 0^k1$: $k < s$. If $k = 0$, then we can generate strings of odd length. Contradiction. If k > 0, we can still generate strings with an odd number of 1s, again reaching a contradiction. An identical argument applies to $y = 10^k : k < s$.
      2. $y = 0^k110^l$: $k+l < s$-1. The important subcase is when $k+l = 0$, and $y = 11$. In that case $xy^kz = 0^n(11)^k0^n = 0^n1^k1^k0^n$, which is in L for all $k \geq 0$.

   Since we cannot control what y is, except for length, we cannot rule out this last case, and so the proof breaks down.

Finite Automata

Pumping Lemmas - Example

4. What do we need to do? We must control where the "pumpable" substring lies, and that is where the strong form of the Pumping Lemma comes in: we must confine the substring to a segment of α that, when pumped, will take us out of the language. So α must be chosen with an appropriate substring. This is where the choice $\alpha = 0^s110^s$ comes in, and where the proof on slide #8 finally works: we have confined y to be of the form 0^k, and in the left half of α.