Proving Languages not to be Regular

SITE: http://www.sir.blois.univ-tours.fr/~mirian/
Proving Languages not to be Regular

- Regular languages has at least four different descriptions:
  1. Languages accepted by DFA
  2. Languages accepted by NFA
  3. Languages accepted by $\epsilon$-NFA
  4. Languages defined by regular expressions

- **NOT every language is a regular language**

- Powerful technique for showing certain languages not to be regular: Pumping Lemma
Theorem: The pumping lemma for regular languages

Let $L$ be a regular language. Then there exists a constant $n$ (which depends on $L$) such that for every string $w$ in $L$ such that $|w| \geq n$, we can break $w$ into three strings, $w = xyz$, such that:

1. $y \neq \epsilon$
2. $|xy| \leq n$
3. For all $k \geq 0$, the string $xy^kz$ is also in $L$

That is, we can always find a nonempty string $y$ not too far from the beginning of $w$ that can be "pumped"; that is, repeating $y$ any number of times, or deleting it (case $k = 0$, keeps the resulting string in the language $L$
Proof

- Suppose \( L \) is regular and recognized by a DFA \( A \). Suppose \( A \) has \( n \) states.
- Let \( w \) be a string of length \( n \) or more, i.e., \( w = a_1 a_2 \ldots a_m \) where \( m \geq n \) and each \( a_i \) is an input symbol.
- For \( i = (0, 1, \ldots n) \) define state \( p_i \) to be \( \hat{\delta}(q_0, a_1 a_2 \ldots a_i) \), where \( \delta \) is the transition function of \( A \) and \( q_0 \) is the start state of \( A \).

That is, \( p_i \) is the state \( A \) is after reading the first \( i \) symbols of \( w \). Note that \( p_0 = q_0 \).
- It is not possible for the \( n + 1 \) different \( p_i \) (for \( i = (0, 1, \ldots n) \)) to be distinct, since there are only \( n \) different states (Pigeonhole principle).
- Thus we can find two different integers \( i \) and \( j \) (with \( 0 \leq i < j \leq n \)) such that \( p_i = p_j \).
Proof (cont.)

- Now we can break \( w = xyz \) as follows:
  1. \( x = a_1 a_2 \ldots a_i \)
  2. \( y = a_{i+1} a_{i+2} \ldots a_j \)
  3. \( z = a_{j+1} a_{j+2} \ldots a_m \)

- That is, \( x \) takes us to \( p_i \) once; \( y \) takes us from \( p_i \) back to \( p_i \) (since \( p_i \) is also \( p_j \)), and \( z \) is the balance of \( w \).
- Note that \( x \) and \( z \) may be empty.
- However \( y \) cannot be empty (\( i \) is strictly less than \( j \)).
Now consider when $A$ receives input $xy^kz$ for $k \geq 0$:

- If $k = 0$ then the automaton goes from the start state to $p_i$ on input $x$. Since $p_i$ is also $p_j$, it must be that $A$ goes from $p_i$ to the accepting state on input $z$. Thus $A$ accepts $xz$.

- If $k > 0$ then $A$ goes from $q_0$ to $p_i$ on input $x$, circles from $p_i$ to $p_i$, $k$ times on input $y^k$, and then goes to the accepting state on input $z$. Thus for any $k \geq 0$, $xy^kz$ is also accepted by $A$. 
The Pumping Lemma as an Adversarial Game

Theorems whose statement involves several alternatives of *for all* and *there exists* quantifiers can be thought of as a game between two players. The pumping lemma is an important example of this type of theorem.

We can see the application of the pumping lemma as a game, in which

1. **Player 1** picks the language $L$ to be proved non regular.
2. **Player 2** picks $n$, but does not reveal to player 1 what $n$ is; player 1 must devise a play for all possible $n$.
3. **Player 1** picks $w$, which may depend on $n$ and which must be of length at least $n$.
4. **Player 2** divides $w$ into $x, y$ and $z$, obeying the constraints that are stipulated in the pumping lemma; $y \neq \epsilon$ and $|x.y| \leq n$. Again, player 2 does not have to tell player 1 what $x, y$ and $z$ are, although they must obey the constraints.
5. **Player 1** wins by picking $k$, which may be a function of $n, x, y$ and $z$, such that $x.y^k.z$ is NOT in $L$. 

Automata Theory, Languages and Computation - Mírian Halfeld-Ferrari – p. 7/8