Given a language $L$, we let $\sim_L$ be the strange-looking relation from before

- $w_1 \sim_L w_2$ if and only if, for every $z$ in $\Sigma^*$, $w_1 z$ and $w_2 z$ are either BOTH in $L$ or BOTH not in $L$

- We are very interested in the equivalence classes defined by the relation $\sim_L$

Let $[w]_L$ be the equivalence class of $w$ under the relation $\sim_L$

Thus, $w_1$ is in $[w]_L$ if and only if, for all $z$ in $\Sigma^*$, $w z$ and $w_1 z$ are either both in $L$ or both not in $L$

Suppose $L$ is regular language over $\Sigma$ and has a DFA $M = (Q, \Sigma, \delta, q_0, F)$ (which probably is not minimal)

Define a relation $\sim_M$ on $\Sigma^*$ by $w_1 \sim_M w_2$ if $\delta^*(q_0, w_1) = \delta^*(q_0, w_2)$
**M** defines finitely many classes

- How many equivalence classes are there?
- There is exactly one equivalence class for each reachable state in \( M \):
- Take \( w \) in \( S^* \), then \( \delta^*(q_0, w) = q_i \) for some reachable state \( q_i \) then \([w]_M = \{ w' \in S^* : \delta^*(q_0, w') = q_i \}\)

---

The key step

- The relation \( \sim_L \) has, at most, the same number of equivalence classes as \( \sim_M \)
- We claim that if \( w_1 \sim_M w_2 \) then \( w_1 \sim_L w_2 \), hence \([w]_M \subseteq [w]_L\) for every \( w \)

---

If there are \( n \) reachable states in \( M \) there are at most \( n \) classes \([w]_M\)
- Since \([w]_M \subseteq [w]_L\) for every \( w \) in \( S^* \), there are at most \( n \) classes \([w]_L\)

---

Conclusions - I

- If \( L \) is regular, there are finitely many equivalence classes \([w]_L\)
- Note the corollary: \( M_L = (Q, \Sigma, q_0, \delta, F) \) is a DFA with the smallest possible number of states, where
  - \( Q \) is the finite set \( \{ [w]_L : w \in \Sigma^* \} \)
  - \( F = \{ [w]_L : w \in L \} \)
  - \( \delta ([w]_L, a) = [wa]_L \)
Conclusions - II

- We have just seen that the number of equivalence classes for the relation $\sim_L$ is the minimum of the number of equivalence classes for the relation $\sim_M$ for all the DFA's $M$ for the language $L$
- Thus, $M_L$ is a minimal DFA

Conclusion - III

- Further, since $[w]_M \subseteq [w]_L$ and each $[w]_M$ corresponds to a reachable state $q$ in $M$, the states $[w]_L$ of $M_L$ are formed by merging states in $M$:

Conclusion - IV

- If we start with a DFA $M$, we obtain a copy of $M_L$ as follows:
  - discard unreachable states
  - obtain a state $[w]_L$ in $M_L$ by merging all the states $q$ in $M$ such that $\delta^*(q_0, w') = q$ where $w \sim_L w'$

A second look at $\sim_L$

- We defined the relation $\sim_L$ on $\Sigma^*$
- Look first at the set $\Sigma^*$ using the example $(0+1)^*$
- The set expands exponentially
- If $\Sigma$ has $k$ elements, then there are $k^n$ strings of length $n$
The equivalence classes

- As an example, consider the language $0^* + 11$
- The black points are in the language, the 3 classes are $\varepsilon$, $0^+$, $11$
- The classes outside the language are $1$ and $(0+1+10+110+111)(0+1)^*$

The minimal DFA

- The equivalence classes are determined by the states of the minimal automaton:

Equivalence classes: $\varepsilon$

- There are other characterizations of the equivalence classes in terms of $\sim_L$:
- $\varepsilon$ is the only string $w$ with the property $w11$ and $w0^n \in L$ for all $n$ but $wz \notin L$ for all other $z$

$0^+$ and $11$

- $0^+$ is the set of strings $w$ with the property $w0^n \in L$ for all $n$ but $wz \notin L$ for all other $z$
- $11$ is the only string $w$ with the property $w\varepsilon \in L$ but $wz \notin L$ for all other $z$
1 and \((0^*1+10)(0+1)^* + 11(0+1)^+\)

- 1 is the only string \(w\) with the property \(w1 \in L\) but \(wz \notin L\) for all other \(z\)
- \((0^*1+10)(0+1)^* + 11(0+1)^+\) is the set of all strings \(w\) such that \(wz \notin L\) for all \(z\)

### Equivalence class

- The equivalence classes once again
- The minimal *complete* DFA

---

**Returning to notes from class 10**

- If we start with a DFA \(M\), we obtain a copy of \(M_L\) as follows:
  - discard unreachable states
  - obtain a state corresponding to \([w]_L\) in \(M_L\) by merging all the states \(q\) in \(M\) such that \(\delta^*(q_0, w') = q\) where \(w \sim_L w'\)

---

**The minimization process**

- Take two reachable states \(q_1\) and \(q_2\) in \(M\), should we merge them?
- For sure, there are strings \(w_1\) and \(w_2\) such that \(\delta^*(q_0, w_1) = q_1\) and \(\delta^*(q_0, w_2) = q_2\), since \(q_1\) and \(q_2\) are reachable
Condition for the merge

- We merge \( q_1 \) and \( q_2 \) if \( w_1 \sim_L w_2 \), i.e. if \( w_1z \) and \( w_2z \) are either both in \( L \) or both not in \( L \) for all \( z \) in \( \Sigma^* \).
- Now, this condition can be restated as \( \delta^*(q_0, w_1z) \) and \( \delta^*(q_0, w_2z) \) are either both final or both non-final for all \( z \) in \( \Sigma^* \).

Simpler form of merge condition

- However, \( \delta^*(q_0, w_1z) = \delta^*(\delta^*(q_0, w_1), z) = \delta^*(q_1, z) \)
  and \( \delta^*(q_0, w_2z) = \delta^*(q_2, z) \)
- So, the condition is: \( \delta^*(q_1, z) \) and \( \delta^*(q_2, z) \) are either both final or both non-final for all \( z \) in \( \Sigma^* \).

Condition for not merging

- Hence, \( q_1 \) and \( q_2 \) should not be merged if there is at least one \( z \) such that only one of \( \delta^*(q_1, z) \) and \( \delta^*(q_2, z) \) is final.
- Now, suppose we follow the path of transitions in \( \delta^*(q_1, z) \) and \( \delta^*(q_2, z) \) for some \( z \) such that only one of the two states is final, where \( |z| > n^2 \) (\( n \) is the number of states in \( M \)).

Beginnings of a pumping lemma

- Let \( z = a_1 a_2 a_3... a_m \) (\( m > n^2 \))
- Consider the \( m+1 \) pairs
  \( (q_1, q_2) \)
  \( (\delta^*(q_1, a_1), \delta^*(q_2, a_1)) \)
  \( (\delta^*(q_1, a_1a_2), \delta^*(q_2, a_1a_2)) \)
  \( (\delta^*(q_1, a_1a_2a_3), \delta^*(q_2, a_1a_2a_3)) \)
  \( (\delta^*(q_1, a_1a_2a_3a_4), \delta^*(q_2, a_1a_2a_3a_4)) \)
  ...
  \( (\delta^*(q_1, a_1a_2a_3...a_m), \delta^*(q_2, a_1a_2a_3...a_m)) \)
Pigeon-hole principle

- How many pairs of states are in our list? -- \( m + 1 > n^2 + 1 \)
- How many pairs of states are possible? -- \( n^2 \)
- By the pigeon-hole principle, there is a duplicate pair:
  \((\delta^*(q_1, a_1a_2a_3...a_i), \delta^*(q_2, a_1a_2a_3...a_i))\)
  and
  \((\delta^*(q_1, a_1a_2a_3...a_j), \delta^*(q_2, a_1a_2a_3...a_j))\)
  for some \( i < j \)

Conclusion

- Let \( z = uvw \), where \( u = a_1a_2a_3...a_i \),
  \( v = a_{i+1}a_2a_3...a_j \), \( w = a_{j+1}a_2a_3...a_m \), then
  \((\delta^*(q_1, uvw), \delta^*(q_2, uvw))\)
  \(= (\delta^*(q_1, uw), \delta^*(q_2, uw))\)
- This means that if \( z \) can be used to distinguish \( q_1 \) and \( q_2 \) and \(|z| > n^2\),
  then there is a shorter string, namely \( uw \), that also distinguishes \( q_1 \) and \( q_2 \)

Algorithm to distinguish states

- We only need to test strings of length \(< n^2 \) to decide if \( q_1 \) and \( q_2 \) are distinguished
- That provides an algorithm

The minimization method

- The method for minimizing automata relies on this algorithm and on the following simple idea:
  - if we can distinguish \( \delta^*(q_1, a) \) and \( \delta^*(q_2, a) \), then \( q_1 \) and \( q_2 \) are distinguished,
  - i.e. IF there is an X in the table we drew for \( \delta^*(q_1, a) \) and \( \delta^*(q_2, a) \), THEN put an X for \( q_1 \) and \( q_2 \)
An algorithm for $\sim_L$

- Finally, deciding whether $w_1 \sim_L w_2$ for a regular language $L$, has an algorithm
- We are simply asking if $[w_1]_L = [w_2]_L$
- This can be done by checking all $[w_1z]_L$ and $[w_2z]_L$ to see if one is inside $L$ and one is not, for all $z$ with $|z| < n^2$, where $n$ is the number of equivalence classes of $\sim_L$

Pictures and examples

- We began with an example of a regular language
- Now consider the famous context-free but not regular language $L_1 = \{0^n1^n : n > 0\}$

Infinitely many equivalence classes

- There are infinitely many equivalence classes
- First there is a break-down into the strings $w$ such that $wz \in L_1$ for some $z$ and those $w$ such that $wz \notin L_1$ for all $z$

Class 1

- The strings $w$ in $\{0^n1^q : q > n > 0\} \cup (0+1)^*10(0+1)^*$ satisfy $wz \notin L_1$ for all $z$, so this is one equivalence class
Class 2

- The strings in \(\{0^n1^n : n > 0\}\) form an equivalence class since this part of \(L_1\) consists of all the \(w\) such that \(we \in L_1\) but \(wz \notin L_1\) for all other \(z\).

Class 3

- \(\varepsilon\) is a class on its own because \(\varepsilon\) is the only string \(w\) such that \(w0^n1^n \in L_1\) for all \(n > 0\) but \(wz \notin L_1\) for all other \(z\).

The first infinite family of classes

- For each \(n > 0\), the singleton set \(0^n\) is an equivalence class since \(0^n\) is the only string \(w\) such that \(w0^n1^n+r \in L_1\) for this fixed \(n\) and any \(r > 0\) but \(wz \notin L_1\) for all other \(z\).

The second infinite family

- For each fixed \(q > 0\), the language \(\{0^{n+q}1^n : n > 0\}\) is an equivalence class because the elements are the only \(w\) such that \(w1^q \in L_1\) for this fixed \(q\) but \(wz \notin L_1\) for all other \(z\).
The partition of \((0+1)^*\)

- We can now write

\[
(0 + 1)^* = \bigcup_{n \geq 0} 0^n \cup \bigcup_{q \geq 0} \{0^{n+q}1^n : n > 0\}
\]

\[
\cup \{0^n1^q : q > n > 0\} \cup (0+1)^*10(0+1)^*
\]

An infinite-state machine for \(L_1\)

- If you make an automaton with infinitely many states (one for each equivalence class) then you have a machine that recognizes the context-free language \(\{0^n1^n : n > 0\}\)

A picture

- We cannot draw infinitely many states but

![Diagram of an infinite-state machine for \(\{0^n1^n : n \geq 0\}\)]

The picture for \(\{0^n1^n : n \geq 0\}\)

- The initial state becomes final

![Diagram of the initial state becoming final in a machine for \(\{0^n1^n : n \geq 0\}\)]
A grammar is a 4-tuple $G = (V, T, P, S)$ where

- $V$ is a finite set of symbols called the variables of the grammar
- $T$ is a finite set of symbols called the terminals of the grammar (was $\Sigma$)
- $P$ is a finite set of transformation rules called productions
- $S$ is a specific, fixed symbol in $V$ called the start symbol

Productions are rules, which can be used to transform one symbol or string of symbols into another string (the resulting string can be empty)

For an unrestricted grammar, the most general kind proposed by Naom Chomsky, the productions have the form

$$\alpha \rightarrow \beta$$

where $\alpha$ and $\beta$ are arbitrary strings of symbols from $V \cup T$

- We must have $\alpha \neq \epsilon$
- It is usually OK if $\beta = \epsilon$
one-step derivations

- Productions are used to define derivations
- A one-step derivation is simply the application of a production somewhere in a string:
  - If $\alpha \rightarrow \beta$ is a production, then $\gamma\alpha\eta \Rightarrow \gamma\beta\eta$ is a one-step derivation
- We have used the production to transform part of the string

derivations

- We need to apply sequences of many derivations to obtain the kind of transformations we need:
- Suppose we have a sequence of strings that are transformed one to the next by one-step derivations:

derivations-II

- $\alpha = \alpha_0$ and $\beta = \alpha_n$
- $\alpha_0 \Rightarrow \alpha_1 \Rightarrow \alpha_2 \Rightarrow \alpha_3 \Rightarrow \ldots \Rightarrow \alpha_{n-1} \Rightarrow \alpha_n$
- In such a case we write $\alpha \Rightarrow \beta$
- We also write $\alpha \Rightarrow \alpha$
- which uses 0 one-step derivations

derivations-III

- If $\alpha \Rightarrow \beta$, we say that $\alpha$ derives $\beta$ in the grammar $G$
- The relation $\Rightarrow$ is the “reflexive, transitive closure” of the relation $\Rightarrow$
- It is the smallest relation that is reflexive and transitive and that also contains the whole of the “one-step derivation” relation
The language of a grammar

- It is easy to write down the definition of the language generated by a grammar $G$

  $$L(G) = \{ w \in T^* : S \Rightarrow^* w \}$$

  It is the set of all terminal strings that can be derived from the start symbol.