machines with output

- We need to discuss automata with outputs before demonstrating that there really is a unique minimal DFA.
- The easier one is the one we will use: Moore machines.
- Simply rethink the DFA: let each state give an output in an alphabet $\Gamma$.
- We could also allow an NFA $\epsilon$.

Moore machines

- After processing an input string $w$ in $\Sigma^*$, we reach a state $q$ which outputs $g$ in $\Gamma$.
- For example, take $\Sigma = \{0, 1\}$ and $\Gamma = \{0, 1, 2\}$ and a Moore machine that determines the remainder modulo 3 of any positive number $n$, where $n$ is expressed in binary:
  
  0, 11, 110, 1001 return 0
  1, 100, 111, 1010 return 1
  10, 101, 1000, 1011 return 2
Example: modulo 3 machine

- To simplify things, let input $\varepsilon$ return output 0.
- Adding a 0 at the right-hand end is a left-shift, i.e. the number is doubled.
- A remainder of 0 is unchanged:

\[ q_0/0 \quad q_1/1 \quad q_2/2 \]

\[ \xrightarrow{0} \]

Doubling

- If you double a number, a remainder of 1 becomes a remainder of 2 and a remainder of 2 becomes a remainder of 1 (= 4 modulo 3).

\[ q_0/0 \quad q_1/1 \quad q_2/2 \]

\[ \xrightarrow{0} \]

Doubling plus 1

- If you add a 1 at the right-hand end of the input, you are doubling the number and adding 1.
- Remainder 0 becomes remainder 1.
- Remainder 1 becomes remainder 0.
- Remainder 2 stays remainder 2.

\[ q_0/0 \quad q_1/1 \quad q_2/2 \]

\[ \xrightarrow{1} \quad \xrightarrow{0} \]

Another way to view an automaton

- An automaton can be regarded as a Moore machine, where final states output “y” and the non-final states output “n”.
- We expect a full automaton.
Mealy machine

- In a Mealy machine, the output is generated by the transition, not by the state
- An input \( w = a_1 a_2 \ldots a_n \) in \( \Sigma^* \) generates an output \( g_1 g_2 \ldots g_n \) in \( \Gamma^* \)

Mealy machine

- Note that the same input at a different state can generate a different output

Transducer

- A more general idea is that every input symbol can generate a string output, including the empty string

Encoding

- Such a transducer is an excellent machine to describe encoding and decoding: input: \( aababa \), output: \( x_1g g x_1 x_2 x_3 \)
decoding?

- The hard thing is to ensure there is another transducer so that input $x_1gx_1x_2x_3$ produces output $aababa$

Down to business

- Take a given language $L$ over $\Sigma$
- We speculate on the existence of a machine $M$ that allows the input of any string $w$ in $\Sigma^*$ and has the following ability:
  - If $w$ is in $L$, the machine outputs “y”
  - If $w$ is not in $L$, the machine outputs “n”
- If $L$ is regular this is like our new version of a DFA on slide 8-13

We do not know how $M$ looks inside

We can only speculate on how $M$ works and if it has well-defined states.

However, assuming there are states inside $M$, we want to tell whether two input strings $w_1$ and $w_2$ take the machine to the same state
Do \( w_1 \) and \( w_2 \) have the same effect?

- Does \( q_1 = q_2 \) ?

\[
\begin{align*}
\text{\( M \) reaches } q_1 & \quad \text{\( y/n \)} \\
\text{\( M \) reaches } q_2 & \quad \text{\( y/n \)}
\end{align*}
\]

Who knows?

- We cannot say, but perhaps we can say that, for all practical purposes \( w_1 \) and \( w_2 \) take \( M \) to the same state

- All we can perhaps tell is whether \( w_1 \) and \( w_2 \) take the machine to states \( q_1 \) and \( q_2 \) that will appear to be the same according to their behavior

The states \( q_1 \) and \( q_2 \) have the same behavior if any string \( z \) applied to \( q_1 \) and \( q_2 \) reaches a state or states that give the same “\( y/n \)” output

\[
\begin{align*}
\text{\( q_1 \) \( \to \) } q_1' & \quad \text{\( y/n \)} \\
\text{\( q_2 \) \( \to \) } q_2' & \quad \text{\( y/n \)}
\end{align*}
\]

The question is: “do \( w_1z \) and \( w_2z \) always agree?”

- As is clear from the diagram, we are simply asking if \( w_1z \) and \( w_2z \) give the same \( y/n \) output for all \( z \)

- Now forget \( M \), there is a special relationship between \( w_1 \) and \( w_2 \) :
  - if \( w_1z \) and \( w_2z \) are either both in \( L \)
  - or both not in \( L \)
  - for all \( z \)
Unique Minimal DFA?

- Why is there a *unique* minimal DFA?
- This is derived from the Myhill-Nerode Theorem, which we demonstrate later
- We will demonstrate the minimal automaton by building a DFA, whose *states are languages!!!*

preamble to equivalence relations

- Relations
  - A relation on a set is a rule $R$ that relates elements of the set: $x R y$
  - On the set of integers we have the relations: $=, <, <=, >, >=, |x - y| > 0$
    - $5 < 10, 6 <= 6, 9 = 9, |8 - 4| > 0$

reflexive relation, symmetric relation

- Relations may be *reflexive*: $x R x$
  - $=, <=, >=$ are reflexive, $>, <$ are not
- Relations may be *symmetric*:
  - IF $x R y$ THEN (ALWAYS) $y R x$
    - (i.e. for all $x$ and $y$)
  - Among the example relations above, only $=$ and $|x - y|$ are symmetric

transitive relation

- Relations may be *transitive*:
  - IF $x R y$ AND $y R z$
    - THEN (ALWAYS) $x R z$
    - (i.e. for all $x$, $y$ and $z$)
  - All the relations we gave as examples are transitive, except for $|x - y|$: $|6 - 8| > 0$ and $|8 - 6| > 0$ but $|6 - 6| = 0$
An equivalence relation is one that has the properties of reflexive, symmetric, and transitive.

An equivalence relation on a set is used to divide the set into equivalence classes.

Suppose $S$ is a set and $R$ is an equivalence relation on the set.

Divide $S$ into subsets using $R$.

Two elements $x$ and $y$ of $S$ are in the same subset if (and only if) $x \sim y$.

Because $R$ is an equivalence relation, any $x$ in $S$ can only be in ONE subset, called an equivalence class.

We denote it by $[x]$. Hence, if $x \sim y$ then $y$ is in $[x]$.

Is it possible that an element $z$ in $S$ could be in two classes $[x]$ and $[y]$, without $[x]$ being identical to $[y]$? i.e. could $[x]$ and $[y]$ overlap without being equal? No, if $z \sim x$ and $z \sim y$, then by symmetry $x \sim z$ and by transitivity $x \sim y$.

$x$ and everything related to $x$ is in $[y]$. Hence $x \sim y$ means $[x] = [y]$. 

Each $x$ belongs to a unique class.

Change the notation to \( \sim \)

- We will use \( \sim \) as the notation for an equivalence relation in place of \( R \)
- These ideas get really exciting when the set \( S \) is \( \Sigma^* \)

The relation \( \sim_L \)

- 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 \( \Sigma^* \), \( w_1z \) and \( w_2z \) are either BOTH in \( L \) or BOTH not in \( L \)
- There is a practical problem (not a theoretical one): there are infinitely many \( z \) so, for a general \( L \), we could never actually finish checking if \( w_1 \sim_L w_2 \)

It is an equivalence relation

- The relation \( \sim_L \) is an equivalence relation
- Reflexive: take any \( w \) in \( \Sigma^* \), then \( w \sim_L w \):
  - for any \( z \) in \( \Sigma^* \), it is obvious that \( wz \) and \( wz \) are always the same, so they are both in \( L \) or both not in \( L \)

Symmetry

- Symmetric: take any \( w_1 \) and \( w_2 \) in \( \Sigma^* \), then \( w_1 \sim_L w_2 \) implies \( w_2 \sim_L w_1 \)
  - for any \( z \) in \( \Sigma^* \), it is obvious that
    - IF \( w_1z \) and \( w_2z \) are always both in \( L \) or both not in \( L \)
    - THEN \( w_2z \) and \( w_1z \) are always both in \( L \) or both not in \( L \)
transitivity

- Transitive: take any $w_1$, $w_2$ and $w_3$ in $\Sigma^*$, then $w_1 \sim_L w_2$ and $w_2 \sim_L w_3$ imply $w_1 \sim_L w_3$

- for any $z$ in $\Sigma^*$, it is obvious that IF $w_1z$ and $w_2z$ are always both in $L$ or both not in $L$
  AND $w_2z$ and $w_3z$ are always both in $L$ or both not in $L$

- THEN $w_1z$ and $w_3z$ are always both in $L$ or both not in $L$

Equivalence classes

- 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 $\Sigma^*$, $wz$ and $w_1z$ are either both in $L$ or both not in $L$

Go back to our magic machine

- Given a language $L$ we can now think about a machine $M_L$ that can recognize $L$

- Populate $M_L$ with states $[w]_L$, where $w$ can be any string in $\Sigma^*$

- The number of states can be infinite

Transitions

- Transitions in this machine are easy:
  $\delta([w]_L, a) = [wa]_L$

- The initial state is easy, it is $[\epsilon]_L$

- The “yes/no” outputs are easy: $[w]_L$ gives “y” if and only if $w$ is in $L$

- The problem is the number of states may be infinite
Is $\delta$ OK?

- We have to ask if $\delta$ is well-defined, i.e. if $[w_1]_L = [w_2]_L$, do we really get the same answer for $\delta([w_1]_L, a)$ and $\delta([w_2]_L, a)$. Does $[w_1 a]_L = [w_2 a]_L$?
- We need $w_1 a z$ and $w_2 a z$ both in $L$ or both not in $L$ for all $z$ in $\Sigma^*$
- But, since $[w_1]_L = [w_2]_L$, we know $w_1 z'$ and $w_2 z'$ are both in $L$ or both not in $L$ for all $z'$ in $\Sigma^*$ and $az$ is just a special case of $z'$

$L(M_L) = L$?

- Does the machine $M_L$ recognize $L$?
- Suppose we give input $w = a_1 a_2 \ldots a_n$ at the start state $[\varepsilon]_L$
- The first $\delta$ transition takes us to $[a_1]_L$, the second to $[a_1 a_2]_L$, the third to $[a_1 a_2 a_3]_L$ and so on
- Hence input $w$ takes us to state $[w]_L$
- The output is “y” if and only if $w$ is in $L$

Myhill-Nerode

- The problem is the number of states (equivalence classes of strings) may be infinite
- The Myhill-Nerode theorem states that $L$ is regular if and only if the number of states in $M_L$ is finite

Two more results

- Further, if $L$ is regular, the machine $M_L$ we just constructed is the minimal complete DFA for $L$
- Also, by a version of the pumping lemma, we will be able to check if $w_1 \sim_L w_2$
Proof of Myhill-Nerode-easy parts

- If the number of states is finite then we have built a DFA for $L$.
- The final states are those that output "y", i.e. all $[w]_L$ where $w$ is in $L$, e.g.

Example

- For this language $[aa]_L = [ab]_L, [a]_L = [ba]_L, [b]_L = [aaab]_L$ and so on.
- The language looks horrible but it is regular.

The converse

- 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 $\delta^*(q_0, w_1) = \delta^*(q_0, w_2)$.

$\sim_M$ is reflexive

- Reflexive: take any $w$ in $\Sigma^*$, then $w \sim_M w$:
  obviously $\delta^*(q_0, w) = \delta^*(q_0, w)$.
\(\sim_M\) is symmetric

- Symmetry: take any \(w_1\) and \(w_2\) in \(\Sigma^*\):
  if \(w_1 \sim_M w_2\) then \(w_2 \sim_M w_1\):
  obviously if \(\delta^*(q_0, w_1) = \delta^*(q_0, w_2)\) then
  \(\delta^*(q_0, w_2) = \delta^*(q_0, w_1)\)

\(\sim_M\) is transitive

- Transitive: take any \(w_1, w_2\) and \(w_3\) in \(\Sigma^*\), then \(w_1 \sim_M w_2\) and \(w_2 \sim_M w_3\) imply
  \(w_1 \sim_M w_3\):
  IF \(\delta^*(q_0, w_1) = \delta^*(q_0, w_2)\)
  AND \(\delta^*(q_0, w_2) = \delta^*(q_0, w_3)\)
  THEN \(\delta^*(q_0, w_1) = \delta^*(q_0, w_3)\)

\(\sim_M\) 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 \(\Sigma^*\), then \(\delta^*(q_0, w) = q_i\) for some reachable state \(q_i\), then
  \([w]_M = \{w' \in \Sigma^* : \delta^*(q_0, w') = q_i\}\)

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 \(w\)
proof of key step

- If \( w_1 \sim_M w_2 \) then \( \delta^*(q_0, w_1) = \delta^*(q_0, w_2) \), hence \( \delta^*(q_0, w_1z) = \delta^*(q_0, w_2z) \) for any \( z \) in \( \Sigma^* \)
- It follows that \( \delta^*(q_0, w_1z) \) is final if and only if \( \delta^*(q_0, w_2z) \) because they are the same state!
- In other words, for any \( z \) in \( \Sigma^* \), \( w_1z \) is in \( L \) if and only if \( w_2z \) is in \( L \), i.e. \( w_1 \sim_L w_2 \)

finitely many classes

- 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 \( \Sigma^* \), 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: \( M_L \) is also a DFA and the number of states in \( M_L \) is equal to the number of equivalence classes for the relation \( \sim_L \)

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