Automata Theory and Formal Grammars: Lecture 4

Minimal Deterministic Automata

Minimal Deterministic Automata

Last Time:

- Regular Expressions and Regular Languages
- Properties of Regular Languages
- Relating NFAs and regular expressions: Kleene's Theorem

Today:

- Decision procedures for FAs
- Distinguishing Strings with respect to a Language
- Minimum-state DFAs for Regular Languages
- Minimizing DFAs using Partition Refinement

Decision Procedures for FAs

Decision Procedures for FAs

A decision procedure is an algorithm for answering a yes/no question.

A number of yes/no questions involving FAs have decision procedures.

- Given FA M and $x \in \Sigma^*$, is $x \in \mathcal{L}(M)$?
- Given FA M, is $\mathcal{L}(M) = \emptyset$?
- Given FAs M_1 and M_2, is $\mathcal{L}(M_1) \subseteq \mathcal{L}(M_2)$?

Answering the first is easy ... but what about the other two?

Deciding Whether $\mathcal{L}(M) = \emptyset$

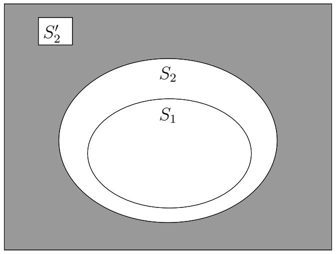
$$\begin{aligned}\mathcal{L}(M) = \emptyset &\iff \forall x \in \Sigma^*.\, x \notin \mathcal{L}(M) \\ &\iff \forall x \in \Sigma^*.\, \delta^*(q_0, x) \notin A\end{aligned}$$

The latter property can be checked using reachability analysis: do all paths from the start state lead to nonaccepting states?

Deciding Whether $\mathcal{L}(M_1) \subseteq \mathcal{L}(M_2)$

For any sets S_1 and S_2 we can reason as follows.

$$\begin{aligned}S_1 \subseteq S_2 &\iff S_1 - S_2 = \emptyset \\ &\iff S_1 \cap \overline{S_2} = \emptyset\end{aligned}$$



Deciding Whether $\mathcal{L}(M_1) \subseteq \mathcal{L}(M_2)$ (cont.)

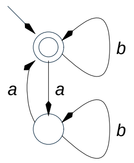
So how can we decide whether or not $\mathcal{L}(M_1) \subseteq \mathcal{L}(M_2)$?

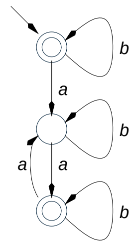
- Build a FA for $\mathcal{L}(M_1) - \mathcal{L}(M_2)$.
  - Complement M_2 to get $\overline{M_2}$.
  - Apply the product construction to get $\Pi(M_1, \overline{M_2})$.
- Check whether or not $\mathcal{L}(\Pi(M_1, \overline{M_2})) = \emptyset$.

Minimizing Automata

How Many States Do You Need in a DFA?

Here are two DFAs recognizing the same language.





The right automaton seems to have a redundant state!

Questions about States in DFAs

- How many states does an DFA need to accept a given language?
- Can a DFA be "minimized" (i.e. can "unnecessary" states be identified and removed)?

We now devote ourselves to answering these questions. All involve a study of the notion of *indistinguishability* of strings.

Indistinguishability

Definition Let $L \subseteq \Sigma^*$ be a langauge. Then the *indistinguishability relation* for L, $\stackrel{L}{\bowtie} \subseteq \Sigma^* \times \Sigma^*$, is defined as follows.

$$x \stackrel{L}{\bowtie} y \text{ iff } \forall z \in \Sigma^*.\, xz \in L \iff yz \in L$$

Intuitively, if $x \stackrel{L}{\bowtie} y$, then any common "extension" to x, y (the "z" in the definition) either makes both xz and yz, or neither, elements of L.

Notes

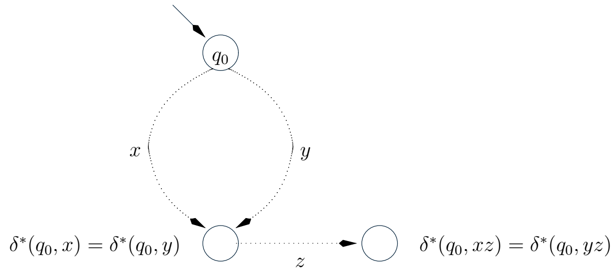
- $x \stackrel{L}{\bowtie} y$ means x, y are indistinguishable *with respect to language* L. (That is, L must be given in order for $\stackrel{L}{\bowtie}$ to be well-defined.)
- $\stackrel{L}{\bowtie}$ relates arbitrary strings, not just elements in L.
- If $x \in L$ and $x \stackrel{L}{\bowtie} y$ then $y \in L$ also (why?).
- Is it true that $x \in L$ and $y \in L$ imply that $x \stackrel{L}{\bowtie} y$?

Examples of Indistinguishability

- Let $L = \{\, w \in \{0,1\}^* \mid w \text{ ends in } 00 \,\}$. Is:
  - $\varepsilon \stackrel{L}{\bowtie} 1$? *Yes*
  - $1 \stackrel{L}{\bowtie} 011$? *Yes*
  - $0 \stackrel{L}{\bowtie} 10$? *Yes*
  - $1 \stackrel{L}{\bowtie} 0$? *No; consider* $z = 0$
- Let $L = \{\, 0^n 1^n \mid n \geq 0 \,\}$. Is:
  - $\varepsilon \stackrel{L}{\bowtie} 1$? *No; consider* $z = 01$
  - $0 \stackrel{L}{\bowtie} 00$? *No; consider* $z = 1$
  - $01 \stackrel{L}{\bowtie} 0011$? *Yes*

Relating $\stackrel{L}{\bowtie}$ and DFAs for L

Let $M = \langle Q, \Sigma, q_0, \delta, A \rangle$ be a DFA accepting L, and suppose $x, y \in \Sigma^*$ are such that $\delta^*(q_0, x) = \delta^*(q_0, y)$.



Then $x \stackrel{L}{\bowtie} y$!

Formally...

Lemma Let $M = \langle Q, \Sigma, q_0, \delta, A \rangle$ be a DFA, and let $x, y \in \Sigma^*$ be such that $\delta^*(q_0, x) = \delta^*(q_0, y)$. Then $x \stackrel{\mathcal{L}(M)}{\bowtie} y$.

Proof Fix $x, y \in \Sigma^*$, and suppose that $\delta^*(q_0, x) = \delta^*(q_0, y)$.
We must prove that $x \stackrel{\mathcal{L}(M)}{\bowtie} y$, i.e.
for any $z \in \Sigma^*$, $xz \in \mathcal{L}(M)$ iff $yz \in \mathcal{L}(M)$. So fix z.
By induction on z, one may establish that $\delta^*(q_0, xz) = \delta^*(q_0, yz)$.
Hence $\delta^*(q_0, xz) \in A$ iff $\delta^*(q_0, yz) \in A$.
This implies that $xz \in \mathcal{L}(M)$ iff $yz \in \mathcal{L}(M)$.

Note The contrapositive of the lemma says that if $x \stackrel{\mathcal{L}(M)}{\not\bowtie} y$ then $\delta^*(q_0, x) \neq \delta^*(q_0, y)$; in other words, if $x \stackrel{\mathcal{L}(M)}{\not\bowtie} y$ then x and y must lead to *different* states in any DFA accepting $\mathcal{L}(M)$.

$\stackrel{L}{\bowtie}$ and Minimum-state Automata

The previous lemma says that if $x \stackrel{L}{\not\bowtie} y$ then any DFA accepting L must have different states for x and y.

Question Suppose $x \stackrel{L}{\bowtie} y$. Could an DFA for L equate the states to which x, y lead to from the start state?

The answer turns out to be "yes". To establish this, we will show how to construct an automaton M_L for L with the property that if $x \stackrel{L}{\bowtie} y$ then $\delta^*(q_0, x) = \delta^*(q_0, y)$.

A Fact About $\stackrel{L}{\bowtie}$

Theorem Let $L \subseteq \Sigma^*$. Then $\stackrel{L}{\bowtie}$ is an equivalence relation on Σ^*.

Proof Outline To prove this we need to show that $\stackrel{L}{\bowtie}$ is:

Reflexive: For any $x \in \Sigma^*$, $x \stackrel{L}{\bowtie} x$.

Symmetric: For any $x, y \in \Sigma^*$, if $x \stackrel{L}{\bowtie} y$ then $y \stackrel{L}{\bowtie} x$.

Transitive: For any $x, y, z \in \Sigma^*$, if $x \stackrel{L}{\bowtie} y$ and $y \stackrel{L}{\bowtie} z$ then $x \stackrel{L}{\bowtie} z$.

$\overset{L}{\bowtie}$ and Equivalence Classes

Since $\overset{L}{\bowtie}$ is an equivalence relation over Σ^*, every $x \in \Sigma^*$ belongs to a unique *equivalence class*.

$$[x]_{\overset{L}{\bowtie}} = \{\, y \in \Sigma^* \mid x \overset{L}{\bowtie} y \,\}$$

Example Let $L = \{\, w \in \{0,1\}^* \mid w \text{ ends in } 00 \,\}$.

- $[\varepsilon]_{\overset{L}{\bowtie}} = \{\, y \in \{0,1\}^* \mid y \text{ does not end in } 0 \,\}$
- What are the other equivalence classes of $\overset{L}{\bowtie}$?

$$\begin{aligned} [0]_{\overset{L}{\bowtie}} &= \{\, y \in \{0,1\}^* \mid y \text{ ends in exactly one } 0 \,\} \\ [00]_{\overset{L}{\bowtie}} &= \{\, y \in \{0,1\}^* \mid y \text{ ends in at least two } 0\text{s} \,\} \end{aligned}$$

Note that every string in $\{0,1\}^*$ falls into one of these three equivalence classes!

Building M_L

In M_L strings indistinguishable with respect to L should lead to the same state.

Idea (for M_L)

- Introduce a state for each equivalence class of $\overset{L}{\bowtie}$.
- Define the transitions so that $\delta^*(q_0, x)$ is $[x]_{\overset{L}{\bowtie}}$.

Questions

- What should the start state be?
  The state corresponding to $[\varepsilon]_{\overset{L}{\bowtie}}$
- What should the accepting states be?
  The states corresponding to $[x]_{\overset{L}{\bowtie}}$ *for each* $x \in L$.
- What should the a-transition of the state for $[x]_{\overset{L}{\bowtie}}$ be?
  The state corresponding to $[xa]_{\overset{L}{\bowtie}}$.

Formalizing the Construction of M_L

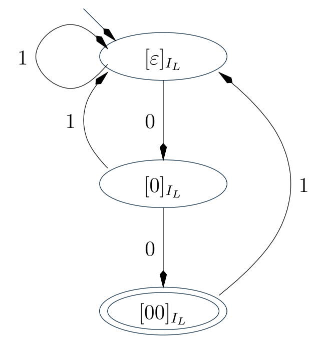
Theorem Let $L \subseteq \Sigma^*$, and consider the automaton $M_L = \langle Q_L, \Sigma, q_L, \delta_L, A_L \rangle$ given as follows.

$$\begin{aligned} Q_L &= \{\, [w]_{\overset{L}{\bowtie}} \mid w \in \Sigma^* \,\} \\ q_L &= [\varepsilon]_{\overset{L}{\bowtie}} \\ \delta_L([w]_{\overset{L}{\bowtie}}, a) &= [wa]_{\overset{L}{\bowtie}} \\ A_L &= \{\, [w]_{\overset{L}{\bowtie}} \mid w \in L \,\} \end{aligned}$$

Then $\mathcal{L}(M_L) = L$, and no automaton recognizing L can have fewer states.

Example of M_L

Let $L = \{\, w \in \{0,1\}^* \mid w \text{ ends in } 00 \,\}$. Then M_L looks like this.



Hmmm...

Is this an algorithm?

Constructive proofs need not be algorithmic.

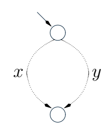
Why is the Theorem True?

- What is $\delta_L^*(q_L, x)$?
  One can show by induction that it is $[x]_{\stackrel{L}{\bowtie}}$.
- When does M_L accept x?
  When $[x]_{\stackrel{L}{\bowtie}} \subseteq L$.
- Suppose $\delta_L^*(q_L, x) = \delta_L^*(q_L, y)$. What is the relationship between x, y?
  $x \stackrel{L}{\bowtie} y$
- Suppose $\delta_L^*(q_L, x) \neq \delta_L^*(q_L, y)$. What is the relationship between x, y?
  $x \stackrel{L}{\not\bowtie} y$

The first two points guarantee that $\mathcal{L}(M_L) = L$; the last two ensure that no DFA for L can have fewer states (why?)!

Reviewing $\stackrel{L}{\bowtie}$

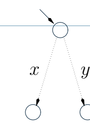
- What does "$x \stackrel{L}{\bowtie} y$" mean?
  That x and y are indistinguishable with respect to language L; that is, for any $z \in \Sigma^*, xz \in L \iff yz \in L$.
- Suppose $x \stackrel{L}{\bowtie} y$ and $y \stackrel{L}{\bowtie} z$. What can we say about x and z, and why?
  $x \stackrel{L}{\bowtie} z$ because $\stackrel{L}{\bowtie}$ is an equivalence relation on $\Sigma^* \times \Sigma^*$.
- Suppose machine M and strings $x, y \in \Sigma^*$ are such that:



  Why is $x \stackrel{\mathcal{L}(M)}{\bowtie} y$?
  Because xz and yz will lead to the same state too, for any $z \in \Sigma^*$!

Reviewing $\stackrel{L}{\bowtie}$ (cont.)

- Suppose machine M and strings $x, y \in \Sigma^*$ are such that:



  Is $x \stackrel{\mathcal{L}(M)}{\not\bowtie} y$?
  Not necessarily.
- Suppose that $x \stackrel{L}{\not\bowtie} y$ and that $M = \langle Q, \Sigma, q_0, \delta, A \rangle$ accepts L. Can $\delta^*(q_0, x) = \delta^*(q_0, y)$?
  No!
- What are the equivalence classes of $\stackrel{L}{\bowtie}$ when $L = \{\, w \in \{0,1\}^* \mid w \text{ has an even number of 1's} \,\}$?
  One is L itself; the other is $\{\, w \in \{0,1\}^* \mid w \text{ has an odd number of 1's} \,\}$.

A Minimum-State DFA for L

- If L is regular, what are the states of the minimum-state DFA M_L for L?

  The equivalence classes of $\overset{L}{\bowtie}$.

- Let L be regular, let $M_L = \langle Q_L, \Sigma, q_L, \delta_L, A_L \rangle$ be the minimum-state DFA for L, and let $x \in \Sigma^*$. What is $\delta_L^*(q_L, x)$?

  $[x]_{\overset{L}{\bowtie}}$, *i.e. the equivalence class of* L*!*

- Let L be regular, and let M_L be the minimum-state DFA M_L for L. What are the accepting states of M_L?

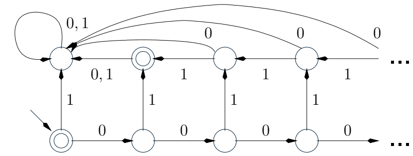
  The equivalence classes of elements of L.

Languages That Are Not Regular

Do nonregular languages exist?

Yes! Consider $L = \{ 0^n 1^n \mid n \geq 0 \}$.

- What would a "FA" look like for this language?



- What can you say about the strings 0^i and 0^j if $i \neq j$?

  If $i \neq j$ *then* $0^i \overset{L}{\not\bowtie} 0^j$*!*

In this case $\overset{L}{\bowtie}$ *has an infinite number of equivalence classes!*

We will revisit this issue next lecture.

Minimizing DFAs

Minimizing DFAs

What we know:

> For any regular language L there is a minimum-state DFA M_L with $\mathcal{L}(M_L) = L$.

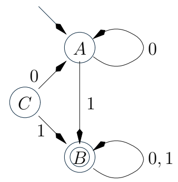
So any DFA for L must have at least as many states as M_L.

Question Suppose we have a DFA M for L. Is there a way to *minimize* M, i.e. generate the mimimum-state DFA M_L by eliminating "unnecessary" states from M?

We'll see....

Unnecessary States in DFAs

Certainly, unreachable states in DFAs are unnecessary:



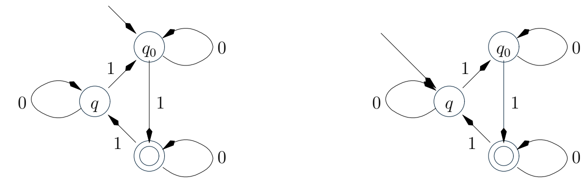
In what follows we will assume these states have already been removed.

Other Unneccessary States: Preliminaries

In what follows, fix DFA $M = \langle Q, \Sigma, q_0, \delta, A \rangle$.

Definition Let $q \in Q$. Then M_q is the DFA $\langle Q, \Sigma, q, \delta, A \rangle$.

M_q is like M except that the start state has been changed to q from q_0:



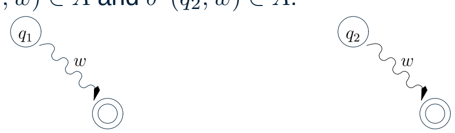
M $\qquad$ M_q

What is $\mathcal{L}(M_q)$? The words leading from q to an accepting state in M!

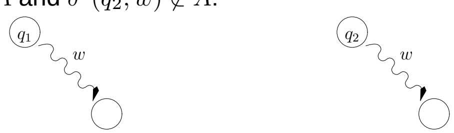
Language Equivalence and States

Definition Let $q_1, q_2 \in Q$. Then $q_1 \overset{M}{\sim} q_2$ if $\mathcal{L}(M_{q_1}) = \mathcal{L}(M_{q_2})$.

- $q_1 \overset{M}{\sim} q_2$ holds if for all $w \in \Sigma^*, \delta^*(q_1, w) \in A$ iff $\delta^*(q_2, w) \in A$. So either $\delta^*(q_1, w) \in A$ and $\delta^*(q_2, w) \in A$:



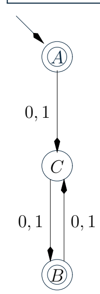
or $\delta^*(q_1, w) \notin A$ and $\delta^*(q_2, w) \notin A$:



Example for $\overset{M}{\sim}$

Let $M = \langle Q, \Sigma, \delta, q_0, A \rangle$ be a DFA, and let $q_1, q_2 \in Q$ be states. Intuitively, $q_1 \overset{M}{\sim} q_2$ holds if the states "accept" the same strings.

Example Consider the following M.



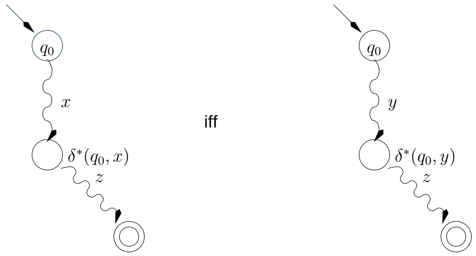
$A \overset{M}{\sim} B$: every string leading from A to accepting state also leads from B to accepting state, and vice versa.

$A \overset{M}{\not\sim} C$: string 0 leads from A to rejecting state but from C to accepting state.

Relating $\overset{M}{\sim}$ and $\overset{\mathcal{L}(M)}{\bowtie}$

What happens if $\delta^*(q_0, x) \overset{M}{\sim} \delta^*(q_0, y)$?

- This means for all $z \in \Sigma^*$, $\delta^*(q_0, xz) \in A$ iff $\delta^*(q_0, yz) \in A$, i.e.:

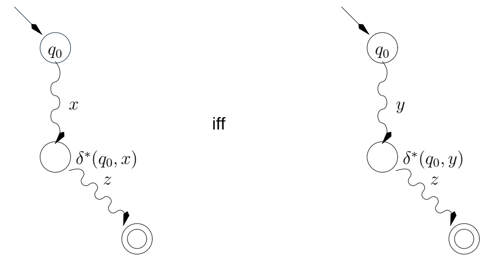


- In other words, $xz \in \mathcal{L}(M)$ iff $yz \in \mathcal{L}(M)$!

Relating $\overset{M}{\sim}$ and $\overset{\mathcal{L}(M)}{\bowtie}$ (cont.)

Now suppose that $x \overset{\mathcal{L}(M)}{\bowtie} y$.

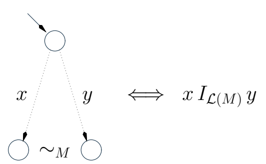
- This means that for all $z \in \Sigma^*$, $xz \in \mathcal{L}(M)$ iff $yz \in \mathcal{L}(M)$.
- In other words:



Lemma Let $M = \langle Q, \Sigma, q_0, \delta, A\rangle$ be a DFA, and let $x, y \in \Sigma^*$. Then $\delta^*(q_0, x) \overset{M}{\sim} \delta^*(q_0, y)$ if and only if $x \overset{\mathcal{L}(M)}{\bowtie} y$.

Facts about $\overset{M}{\sim}$

- Let $x, y \in \Sigma^*$. Then $\delta^*(q_0, x) \overset{M}{\sim} \delta^*(q_0, y)$ iff $x \overset{\mathcal{L}(M)}{\bowtie} y$.



- $\overset{M}{\sim}$ is an *equivalence relation* and thus has *equivalence classes*.
- If $q_1 \overset{M}{\sim} q_2$ then $\delta(q_1, a) \overset{M}{\sim} \delta(q_2, a)$ for any $a \in \Sigma$.

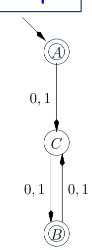
- In contrast with $\overset{\mathcal{L}(M)}{\bowtie}$, $\overset{M}{\sim}$ is an equivalence relation over a *finite* set (Q) rather than an infinite one (Σ^*).

Constructing Minimum-State DFAs

The previous facts suggest a means for minimizing DFAs.

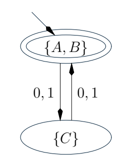
- "Collapse" $\overset{M}{\sim}$ states into a single state
- "Merge" transitions.

Example



M

$\overset{M}{\sim}$ equivalence classes: $\{A, B\}, \{C\}$



Minimized M

Merging Language-Equivalent States

We just established this:

$\boxed{\text{Lemma}}$ Let $M = \langle Q, \Sigma, q_0, \delta, A \rangle$ be a DFA, and suppose that $q_1 \overset{M}{\sim} q_2$. Then for any $a \in \Sigma$, $\delta(q_1, a) \overset{M}{\sim} \delta(q_2, a)$.

We can now "merge" redundant states as follows!

$\boxed{\text{Theorem}}$ Let $M = \langle Q, \Sigma, q_0, \delta, A \rangle$ be a DFA. Then the automaton $M_L = \langle Q_L, \Sigma, q_L, \delta_L, A_L \rangle$ given below is a mimimum-state DFA accepting $\mathcal{L}(M)$.

$$\begin{aligned} Q_L &= \{\, [q]_{\overset{M}{\sim}} \mid q \in Q \,\} \\ q_L &= [q_0]_{\overset{M}{\sim}} \\ \delta([q]_{\overset{M}{\sim}}, a) &= [\delta(q, a)]_{\overset{M}{\sim}} \\ A_L &= \{\, [q]_{\overset{M}{\sim}} \mid q \in A \,\} \end{aligned}$$

Computing Equivalence Classes of $\overset{M}{\sim}$

In order to minimize DFAs mechanically, we need to be able to compute the equivalence classes of $\overset{M}{\sim}$ for a given DFA M.

This can be done using a *partition refinement* algorithm.

- We initially make crude assumptions about which states are related by $\overset{M}{\sim}$. (I.e. we assume a small number of large equivalence classes.)
- Based on an analysis of outgoing transitions, we may split some equivalence classes when they are found to contain states not related by $\overset{M}{\sim}$.
- When we can't split any more, we're done.

List of equivalence classes: *partition*.
Splitting equivalence classes: *refinement*.

The Initial Partition

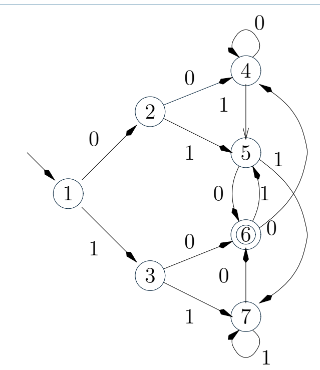
Where do we start our partition-refinement algorithm? In other words, which states are guaranteed not to be $\overset{M}{\sim}$ related?

$\boxed{\text{Claim}}$ If $q_1 \in A$ and $q_2 \notin A$ then $q_1 \overset{M}{\not\sim} q_2$.

$\boxed{\text{Why?}}$ Because $\varepsilon \in \mathcal{L}(M_{q_1})$ and $\varepsilon \notin \mathcal{L}(M_{q_2})$!

So the partition refinement algorithm starts off with an initial partition containing two equivalence classes: A and $Q - A$.

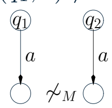
Example



Initial partition: $\{6\}$, $\{1, 2, 3, 4, 5, 7\}$

Refining Partitions

Suppose q_1, q_2 are such that $\delta(q_1, a) \overset{M}{\not\sim} \delta(q_2, a)$ for some $a \in \Sigma$.



Then $q_1 \overset{L}{\not\sim} q_2$! (Why?)

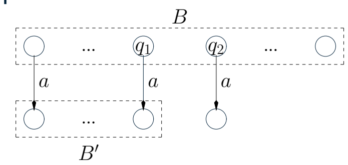
This means that if we have an equivalence class (or *block*) B such that

- q_1, q_2 are in B, but
- there is an a such that $\delta(q_1, a)$ and $\delta(q_2, a)$ are in different blocks,

then B should be split into two new classes: one containing q_1, and one containing q_2!

Splitting Blocks

More precisely, suppose we have:



That is, $q_1, q_2 \in B$ and het $\delta(q_1, a) \in B'$ but $\delta(q_2, a) \notin B'$. Then B should be split into:

$$\begin{aligned} B_1 &= \{\, q \in B \mid \delta(q, a) \in B' \,\} \\ B_2 &= \{\, q \in B \mid \delta(q, a) \notin B' \,\} \end{aligned}$$

Example

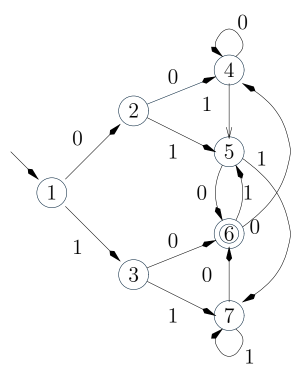
Initial partition: $\{6\}$, $\{1, 2, 3, 4, 5, 7\}$
In $B = \{1, 2, 3, 4, 5, 7\}$:

- $3, 5, 7$ have 0 transitions to $B' = \{6\}$.
- $1, 2, 4$ do not.

So B should be split into:

- $B_1 = \{3, 5, 7\}$, and
- $B_2 = \{1, 2, 4\}$.

New partition: $\{6\}, \{3, 5, 7\}, \{1, 2, 4\}$.



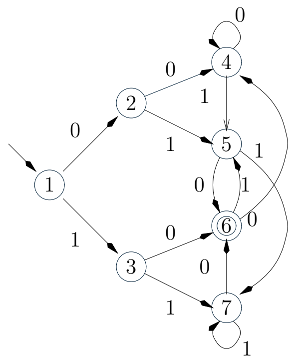
The Algorithm for Computing Equivalence Classes of $\overset{M}{\sim}$

- Start with partition $\{A, Q - A\}$.
- While there is a block B that should be split, generate a new partition by replacing B with B_1 and B_2.
- Halt when no more splitting is possible.

It turns out that when the algorithm terminates, the blocks are exactly the equivalence classes of $\overset{M}{\sim}$!

These can then be used to generated the minimized version M_L of M.

Example



Summary: Regular Languages...

- are defined using regular expressions
- are processed mechanically via DFAs/NFAs
- are closed with respect to $\circ$, *, $\cup$, complement, $\cap$, ...
- have a characterization in terms of equivalence classes of "indistinguishability"
- have minimum-state DFA acceptors