CS420
Combining DFAs and Closed Operations
Monday, February 12, 2024
UMass Boston Computer Science
Announcements

• HW 1 in
  • Due Mon 2/12 12pm

• HW 2 out
  • Due Mon 2/19 12pm

• Check previous Piazza posts before posting!
Languages Are Computation Models

- The **language** of a machine = set of strings that it **accepts**.

  - E.g., a DFA \( M = (Q, \Sigma, \delta, q_0, F) \) **recognizes** language \( A \): if \( A = \{ w \mid M \text{ accepts } w \} \).

- A **computation model** = set of machines it defines.

  - E.g., all possible DFAs are a computation model.

Thus: a **computation model** equivalently = a set of **languages**.

This class is really about studying **sets of languages**!
Languages Are Computation Models

- first set of languages we will study: **regular languages**

  If a DFA recognizes a language $L$, then $L$ is a **regular language**

Thus: a **computation model** equivalently = a **set of languages**

This class is **really** about studying **sets of languages!**
Is it regular?: strings with odd # 1s

**States:**
- 2 states:
  - seen even 1s so far
  - seen odds 1s so far

**Alphabet:** $\emptyset$ and 1

**Transitions:**

**Start / Accept states:**

(Part of Proof requires)
Creating DFA:

So a DFA’s computation recognizes simple string patterns?

Yes!

Have you ever used a programming language (feature) for writing string matching computation?

Regular Expressions! (stay tuned!)
Combining DFAs?

Password Requirements

- Passwords must have a minimum length of ten (10) characters - but more is better!
- Passwords **must include at least 3** different types of characters:
  - upper-case letters (A-Z)
  - lower-case letters (a-z)
  - symbols or special characters (%, &, *, $, etc.)
  - numbers (0-9)
- Passwords cannot contain all or part of your email address
- Passwords cannot be re-used

To match all requirements, combine smaller DFAs into one big DFA?

umb.edu/it/software-systems/password/

(We do this with programs all the time)
Password Checker DFAs

To combine more than once, this must be a DFA

$M_1$: Check special chars

$M_2$: Check uppercase

$M_3$: “OR”

$M_4$: Check length

$M_5$: “AND”

Want to be able to easily combine DFAs, i.e., *composability*

We want these operations:

“OR” : DFA × DFA → DFA

“AND” : DFA × DFA → DFA

To combine more than once, operations must be closed!
“Closed” Operations

- Set of Natural numbers = \{0, 1, 2, ...\}
  - Closed under addition:
    - if \(x\) and \(y\) are Natural numbers,
    - then \(z = x + y\) is a Natural number
  - Closed under multiplication?
    - yes
  - Closed under subtraction?
    - no

- Integers = {..., -2, -1, 0, 1, 2, ...}
  - Closed under addition and multiplication
  - Closed under subtraction?
    - yes
  - Closed under division?
    - no

- Rational numbers = \{\(x\) | \(x = y/z\), \(y\) and \(z\) are Integers\}
  - Closed under division?
    - No?
    - Yes if \(z \neq 0\)

A set is **closed** under an operation if: the result of applying the operation to members of the set is in the same set

i.e., input set(s) = output set
We Want “Closed” Ops For Regular Langs!

- Set of Regular Languages = \{L_1, L_2, \ldots\}
  - **Closed** under ...?
    - OR (union)
    - AND (intersection)
    - ...

**A set is closed** under an operation if: the **result** of applying the operation to members of the set is in the same set

i.e., input set(s) = output set
Why Care About Closed Ops on Reg Langs?

- Closed operations for regular langs preserve “regularness”
- I.e., it preserves the same computation model!
- Allows “combining” smaller “regular” computations to get bigger ones:

  For Example:
  OR: Regular Lang × Regular Lang → Regular Lang

- So this semester, **we will look for operations that are closed!**
Password Checker: "OR" = "Union"

$M_3$: "OR"

$M_1$: Check special chars

$M_2$: Check uppercase

A \hspace{2cm} B
Union of Languages

Let the alphabet $\Sigma$ be the standard 26 letters \( \{a, b, \ldots, z\} \).

If $A = \{\text{fort, south}\}$, $B = \{\text{point, boston}\}$

\[
A \cup B = \{\text{fort, south, point, boston}\}
\]
Is Union Closed For Regular Langs?

In this course, we are interested in closed operations for a set of languages (here the set of regular languages). The class of regular languages is closed under the union operation. In other words, if $A_1$ and $A_2$ are regular languages, so is $A_1 \cup A_2$.

(In general, a set is closed under an operation if applying the operation to members of the set produces a result in the same set.)
Is Union Closed For Regular Langs?

**Theorem**

The class of regular languages is closed under the union operation.

In other words, if $A_1$ and $A_2$ are regular languages, so is $A_1 \cup A_2$.

**In general,** a set is **closed** under an operation if applying the operation to members of the set produces a result in the same set.

A member of the set of regular languages is ...

... a regular language, which itself is a set (of strings) ...

... so the operations we’re interested in are **set operations**.
Is Union Closed For Regular Langs?

**Theorem**

The class of regular languages is closed under the union operation. In other words, if $A_1$ and $A_2$ are regular languages, so is $A_1 \cup A_2$.
Flashback: Mathematical Statements: IF-THEN

Using:

- If we know: \( P \rightarrow Q \) is TRUE, what do we know about \( P \) and \( Q \) individually?
  - Either \( P \) is FALSE (not too useful, can’t prove anything about \( Q \)), or
  - If \( P \) is TRUE, then \( Q \) is TRUE (modus ponens)

Proving:

\[
\begin{array}{|c|c|c|}
\hline
p & q & p \rightarrow q \\
\hline
\text{True} & \text{True} & \text{True} \\
\text{True} & \text{False} & \text{False} \\
\text{False} & \text{True} & \text{True} \\
\text{False} & \text{False} & \text{True} \\
\hline
\end{array}
\]
Flashback: Mathematical Statements: IF-THEN

Theorem

1. The class of regular languages is closed under the union operation. In other words, if $A_1$ and $A_2$ are regular languages, so is $A_1 \cup A_2$.

   - If $P$ is TRUE, then $Q$ is TRUE (modus ponens)

   - If $P \rightarrow Q$ is TRUE

   - $P \rightarrow Q$ is TRUE

Proving:

- To prove: $P \rightarrow Q$ is TRUE:
  - Prove $P$ is FALSE (usually hard or impossible)
  - Assume $P$ is TRUE, then prove $Q$ is TRUE


Would have to prove there are no regular languages (impossible)

<table>
<thead>
<tr>
<th></th>
<th></th>
<th>$p \rightarrow q$</th>
</tr>
</thead>
<tbody>
<tr>
<td>True</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>True</td>
<td>False</td>
<td>False</td>
</tr>
<tr>
<td>False</td>
<td>True</td>
<td>True</td>
</tr>
<tr>
<td>False</td>
<td>False</td>
<td>True</td>
</tr>
</tbody>
</table>
Is Union Closed For Regular Langs?

**Statements**

1. $A_1$ and $A_2$ are regular languages
2. A DFA $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognizes $A_1$
3. A DFA $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognizes $A_2$
4. Construct DFA $M = (Q, \Sigma, \delta, q_0, F)$ (todo)
5. $M$ recognizes $A_1 \cup A_2$
6. $A_1 \cup A_2$ is a regular language
7. The class of regular languages is closed under the union operation.

In other words, if $A_1$ and $A_2$ are regular languages, so is $A_1 \cup A_2$.

**Justifications**

1. Assumption
2. Def of Regular Language
3. Def of Regular Language
4. Def of DFA
5. See examples
6. Def of Regular Language
7. From stmt #1 and #6
Wait! If $A$ Then $B$ =?= If $B$ Then $A$

1. $A_1$ and $A_2$ are regular languages
2. A DFA $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognizes $A_1$
3. A DFA $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognizes $A_2$

If a DFA recognizes a language $L$, then $L$ is a regular language

If $L$ is a regular language, then a DFA recognizes $L$ ???
Equivalence of Conditional Statements

• Yes or No? “If X then Y” is equivalent to:

  • “If Y then X” (**converse**)
    • No!
If Regular, Then DFA?

If a DFA recognizes a language $L$, then $L$ is a regular language

- **Prove:** If $L$ is a **regular language**, then a **DFA** recognizes $L$

- **Proof (Sketch)**
  
  **Case analysis:**
  - **Look at all** if-then statements of the form:
    - “If ... language $L$, then $L$ is a regular language”
  - (At least one is true!)
  - **Figure out which one(s) led to conclusion:**
    - “$L$ is a regular language”
    - (There’s only 1!)

- **So it must be that:**
  If $L$ is a **regular language**, then a **DFA** recognizes $L$
A finite automaton is a 5-tuple \((Q, \Sigma, \delta, q_0, F)\), where

1. \(Q\) is a finite set called the **states**,
2. \(\Sigma\) is a finite set called the **alphabet**,
3. \(\delta: Q \times \Sigma \to Q\) is the **transition function**,
4. \(q_0 \in Q\) is the **start state**, and
5. \(F \subseteq Q\) is the **set of accept states**.

If \(L\) is a **regular language**, then a **DFA** recognizes \(L\).
Rough sketch Idea: $M$ is a combination of $M_1$ and $M_2$ that checks whether its input is accepted by either $M_1$ or $M_2$.

But, a DFA can only read its input once!

Need to somehow simulate “being in” both an $M_1$ and $M_2$ state simultaneously.

**THEOREM**

The class of regular languages is closed under the union operation.

In other words, if $A_1$ and $A_2$ are regular languages, so is $A_1 \cup A_2$. 

**Goal:** $M$ recognizes $A_1 \cup A_2$ 

(to prove $A_1 \cup A_2$ is regular)
Is Union Closed For Regular Langs?

**Statements**

1. $A_1$ and $A_2$ are regular languages
2. A DFA $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognizes $A_1$
3. A DFA $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognizes $A_2$
4. Construct DFA $M = (Q, \Sigma, \delta, q_0, F)$ (todo)
5. $M$ recognizes $A_1 \cup A_2$
6. $A_1 \cup A_2$ is a regular language
7. The class of regular languages is closed under the union operation.

**Justifications**

1. Assumption
2. Def of Regular Language
3. Def of Regular Language
4. Def of DFA
5. See examples
6. Def of Regular Language
7. From stmt #1 and #6

In other words, if $A_1$ and $A_2$ are regular languages, so is $A_1 \cup A_2$. 

Previously
Union is Closed For Regular Languages

Proof (continuation)

• Given: $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$, recognize $A_1$, $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$, recognize $A_2$.

• Construct: $M = (Q, \Sigma, \delta, q_0, F)$, using $M_1$ and $M_2$, that recognizes $A_1 \cup A_2$.

• states of $M$:

  $Q = \{(r_1, r_2) \mid r_1 \in Q_1 \text{ and } r_2 \in Q_2\} = Q_1 \times Q_2$

  This set is the **Cartesian product** of sets $Q_1$ and $Q_2$.

A finite automaton is a 5-tuple $(Q, \Sigma, \delta, q_0, F)$, where

1. $Q$ is a finite set called the **states**,
2. $\Sigma$ is a finite set called the **alphabet**,
3. $\delta: Q \times \Sigma \to Q$ is the **transition function**,\(^1\)
4. $q_0 \in Q$ is the **start state**, and
5. $F \subseteq Q$ is the set of **accept states**.

Want: $M$ that can simultaneously “be in” both an $M_1$ and $M_2$ state.

A state of $M$ is a pair:
- the **first** part is a state of $M_1$ and
- the **second** part is a state of $M_2$.

So the states of $M$ is all possible combinations of the states of $M_1$ and $M_2$.\(^1\)
Proof (continuation)

• Given: \( M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1) \), recognize \( A_1 \),
  \( M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2) \), recognize \( A_2 \),

• Construct: \( M = (Q, \Sigma, \delta, q_0, F) \), using \( M_1 \) and \( M_2 \), that recognizes \( A_1 \cup A_2 \)

• states of \( M \): \( Q = \{(r_1, r_2) | r_1 \in Q_1 \text{ and } r_2 \in Q_2\} = Q_1 \times Q_2 \)
This set is the \textit{Cartesian product} of sets \( Q_1 \) and \( Q_2 \)

A \textit{finite automaton} is a 5-tuple \((Q, \Sigma, \delta, q_0, F)\), where \( (a) = (\delta_1(r_1, a), \delta_2(r_2, a)) \)

A step in \( M \) is both:
- a step in \( M_1 \), and
- a step in \( M_2 \)
Union is Closed For Regular Languages

Proof (continuation)

• Given: $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$, recognize $A_1$,
  $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$, recognize $A_2$,

• Construct: $M = (Q, \Sigma, \delta, q_0, F)$, using $M_1$ and $M_2$, that recognizes $A_1 \cup A_2$

• states of $M$: $Q = \{(r_1, r_2) | r_1 \in Q_1 \text{ and } r_2 \in Q_2\} = Q_1 \times Q_2$
  This set is the Cartesian product of sets $Q_1$ and $Q_2$

• $M$ transition fn: $\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a))$

• $M$ start state: $(q_1, q_2)$  
  Start state of $M$ is both
  start states of $M_1$ and $M_2$
Union is Closed For Regular Languages

Proof (continuation)

• Given: $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$, recognize $A_1$,
  $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$, recognize $A_2$,

• Construct: $M = (Q, \Sigma, \delta, q_0, F)$, using $M_1$ and $M_2$, that recognizes $A_1 \cup A_2$

• states of $M$: $Q = \{(r_1, r_2) | r_1 \in Q_1 \text{ and } r_2 \in Q_2\} = Q_1 \times Q_2$
  This set is the *Cartesian product* of sets $Q_1$ and $Q_2$

• $M$ transition fn: $\delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a))$

• $M$ start state: $(q_1, q_2)$

• $M$ accept states: $F = \{(r_1, r_2) | r_1 \in F_1 \text{ or } r_2 \in F_2\}$

  Accept if either $M_1$ or $M_2$ accept

Remember: Accept states must be subset of $Q$

Q.E.D.?
Is Union Closed For Regular Langs?

**Statements**

1. $A_1$ and $A_2$ are regular languages
2. A DFA $M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$ recognizes $A_1$
3. A DFA $M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$ recognizes $A_2$
4. Construct DFA $M = (Q, \Sigma, \delta, q_0, F)$
5. $M$ recognizes $A_1 \cup A_2$
6. $A_1 \cup A_2$ is a regular language
7. The class of regular languages is closed under the union operation.

**Justifications**

1. Assumption
2. Def of Regular Language
3. Def of Regular Language
4. Def of DFA
5. See examples
6. Def of Regular Language
7. From stmt #1 and #6

In other words, if $A_1$ and $A_2$ are regular languages, so is $A_1 \cup A_2$. 

*Previously*
“Prove” that DFA recognizes a language

Let $s_1 \in A_1$ and $s_2 \in A_2$
Let $s_3 \notin A_1$ and $s_4 \notin A_2$

Be careful when choosing examples!

<table>
<thead>
<tr>
<th>String</th>
<th>In lang $A_1 \cup A_2$?</th>
<th>Accepted by $M$?</th>
</tr>
</thead>
<tbody>
<tr>
<td>$s_1$</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>$s_2$</td>
<td>Yes</td>
<td></td>
</tr>
<tr>
<td>$s_3$</td>
<td>???</td>
<td></td>
</tr>
<tr>
<td>$s_4$</td>
<td>???</td>
<td></td>
</tr>
</tbody>
</table>

Don’t know $A_1$ and $A_2$ exactly ...

... but we know ...

... they are sets of strings!

$M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1)$, recognize $A_1$,
$M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2)$, recognize $A_2$,
constructed $M = (Q, \Sigma, \delta, q_0, F)$ recognizes $A_1 \cup A_2$?

In this class, a table like this is sufficient to “prove” that a DFA recognizes a language
“Prove” that DFA recognizes a language

Let \( s_1 \in A_1 \) and \( s_2 \in A_2 \)

Let \( s_3 \notin A_1 \) and \( s_4 \notin A_2 \)

Let \( s_5 \notin A_1 \) and \( \notin A_2 \)

<table>
<thead>
<tr>
<th>String</th>
<th>In lang ( A_1 \cup A_2 )?</th>
<th>Accepted by ( M )?</th>
</tr>
</thead>
<tbody>
<tr>
<td>( s_1 )</td>
<td>Yes</td>
<td>( \text{???} )</td>
</tr>
<tr>
<td>( s_2 )</td>
<td>Yes</td>
<td>( \text{???} )</td>
</tr>
<tr>
<td>( s_3 )</td>
<td>( \text{???} )</td>
<td>( \text{???} )</td>
</tr>
<tr>
<td>( s_4 )</td>
<td>( \text{???} )</td>
<td>( \text{???} )</td>
</tr>
<tr>
<td>( s_5 )</td>
<td>No</td>
<td>( \text{???} )</td>
</tr>
</tbody>
</table>

\( M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1) \), recognize \( A_1 \),

\( M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2) \), recognize \( A_2 \),

constructed \( M = (Q, \Sigma, \delta, q_0, F) \) recognizes \( A_1 \cup A_2 \)?
Union is Closed For Regular Languages

Proof (continuation)

• Given: 
  
  \[ M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1) \], recognize \( A_1 \),
  
  \[ M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2) \], recognize \( A_2 \),

• Construct: \( M = (Q, \Sigma, \delta, q_0, F) \), using \( M_1 \) and \( M_2 \), that recognizes \( A_1 \cup A_2 \)

• states of \( M \): 
  
  \[ Q = \{(r_1, r_2) | r_1 \in Q_1 \text{ and } r_2 \in Q_2\} = Q_1 \times Q_2 \]
  
  This set is the Cartesian product of sets \( Q_1 \) and \( Q_2 \)

• \( M \) transition fn: 
  
  \[ \delta((r_1, r_2), a) = (\delta_1(r_1, a), \delta_2(r_2, a)) \]

• \( M \) start state: 
  
  \[ (q_1, q_2) \]

• Accept if either \( M_1 \) or \( M_2 \) accept

• \( M \) accept states: 
  
  \[ F = \{(r_1, r_2) | r_1 \in F_1 \text{ or } r_2 \in F_2\} \]
“Prove” that DFA recognizes a language

Let \( s_1 \in A_1 \) and \( s_2 \in A_2 \)

Let \( s_5 \notin A_1 \) and \( \notin A_2 \)

<table>
<thead>
<tr>
<th>String</th>
<th>In lang ( A_1 \cup A_2 )</th>
<th>Accepted by ( M )</th>
</tr>
</thead>
<tbody>
<tr>
<td>( s_1 )</td>
<td>Yes</td>
<td>Accept</td>
</tr>
<tr>
<td>( s_2 )</td>
<td>Yes</td>
<td>Accept</td>
</tr>
<tr>
<td>( s_3 )</td>
<td>???</td>
<td>???</td>
</tr>
<tr>
<td>( s_4 )</td>
<td>???</td>
<td>???</td>
</tr>
<tr>
<td>( s_5 )</td>
<td>No</td>
<td>Reject</td>
</tr>
</tbody>
</table>

\( M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1) \), recognize \( A_1 \),
\( M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2) \), recognize \( A_2 \),
constructed \( M = (Q, \Sigma, \delta, q_0, F) \)

Accept if either \( M_1 \) or \( M_2 \) accept
Is Union Closed For Regular Langs?

**Statements**

1. \( A_1 \) and \( A_2 \) are regular languages
2. A DFA \( M_1 = (Q_1, \Sigma, \delta_1, q_1, F_1) \) recognizes \( A_1 \)
3. A DFA \( M_2 = (Q_2, \Sigma, \delta_2, q_2, F_2) \) recognizes \( A_2 \)
4. Construct DFA \( M = (Q, \Sigma, \delta, q_0, F) \)
5. \( M \) recognizes \( A_1 \cup A_2 \)
6. \( A_1 \cup A_2 \) is a regular language
7. The class of regular languages is closed under the union operation.

**Justifications**

1. Assumption
2. Def of Regular Language
3. Def of Regular Language
4. Def of DFA
5. See examples
6. Def of Regular Language
7. From stmt #1 and #6

In other words, if \( A_1 \) and \( A_2 \) are regular languages, so is \( A_1 \cup A_2 \).
Another operation: Concatenation

Example: Recognizing street addresses

212 Beacon Street

\[ M_3: \text{"CONCAT"} \]

\[ M_1: \text{recognize numbers} \]

\[ M_2: \text{recognize words} \]
Concatenation of Languages

Let the alphabet $\Sigma$ be the standard 26 letters $\{a, b, \ldots, z\}$.

If $A = \{\text{fort, south}\} \quad B = \{\text{point, boston}\}$

$$A \circ B = \{\text{fortpoint, fortboston, southpoint, southboston}\}$$
Is Concatenation Closed?

**THEOREM**

The class of regular languages is closed under the concatenation operation.

In other words, if $A_1$ and $A_2$ are regular languages then so is $A_1 \circ A_2$.

- **Construct a new machine $M$ recognizing $A_1 \circ A_2$?** (like union)
  - Using DFA $M_1$ (which recognizes $A_1$),
  - and DFA $M_2$ (which recognizes $A_2$)
Let $M_1$ recognize $A_1$, and $M_2$ recognize $A_2$.

**Want:** Construction of $M$ to recognize $A_1 \circ A_2$.

**Problem:** Can only read input once, can't backtrack.
Overlapping Concatenation Example

• Let $M_1$ recognize language $A = \{ \text{jen, jens} \}$
• and $M_2$ recognize language $B = \{ \text{smith} \}$
• Want: Construct $M$ to recognize $A \circ B = \{ \text{jensmith, Jensssmith} \}$

• If $M$ sees jen ...
• $M$ must decide to either:
Overlapping Concatenation Example

- Let $M_1$ recognize language $A = \{ \text{jens}, \text{jen} \}$
- and $M_2$ recognize language $B = \{ \text{smith} \}$
- Want: Construct $M$ to recognize $A \circ B = \{ \text{jenssmith}, \text{jenssmith} \}$

- If $M$ sees jen ...
- $M$ must decide to either:
  - stay in $M_1$ (correct, if full input is jenssmith)
Overlapping Concatenation Example

- Let $M_1$ recognize language $A = \{\text{jens} \}$
- and $M_2$ recognize language $B = \{\text{smith}\}$
- Want: Construct $M$ to recognize $A \circ B = \{\text{jensmith, jenssmith}\}$

- If $M$ sees jen ...
- $M$ must decide to either:
  - stay in $M_1$ (correct, if full input is jenssmith)
  - or switch to $M_2$ (correct, if full input is jenssmith)

- But to recognize $A \circ B$, it needs to handle both cases!!
  - Without backtracking
Is Concatenation Closed?

**FALSE?**

**THEOREM**

The class of regular languages is closed under the concatenation operation.

In other words, if $A_1$ and $A_2$ are regular languages then so is $A_1 \circ A_2$.

- Cannot **combine** $A_1$ and $A_2$’s machine because:
  - Need to switch from $A_1$ to $A_2$ at some point …
  - … but we don’t know when! (we can only read input once)
- **This requires a new kind of machine!**
- **But** does this mean concatenation **is not closed** for regular langs?