Nondeterminism

Monday Feb 8, 2021
Logistics

• HW 0, HW 1 done

• HW 2 due Sunday 2/14 11:59pm EST

• No class next Monday 2/15

• (Some) HW 0 solutions posted

• Questions?
A Brief Intro to XML

• What is it?
  • It’s a widely-used “data interchange format”
    • I.e., A standard, language-agnostic way for programs to send/recv data
  • (JSON is another popular interchange format)

• Example, when querying web apis:
  • https://api.etrade.com/v1/market/quote/GOOG
XML in this class: 2 purposes

1. Grader uses it to send/get state machines to/from HW
   - E.g.

```xml
<automaton>
    <!--The list of states.-->
    <state id="0" name="q1"><initial/></state>
    <state id="1" name="q2"><final/></state>
    <state id="2" name="q3"></state>

    <!--The list of transitions.-->
    <transition>
        <from>0</from>
        <to>0</to>
        <read>0</read>
    </transition>
    <transition>
        <from>1</from>
    </transition>
</automaton>
```

- **Open tags** may contain **attributes**
- **attribute name**
- **attribute value**
- **element** = open/close tag + everything in between
- **Open tag**
- **Close tag**

Elements nest, i.e., they may contain other elements
XML in this class: 2 purposes

2. Running example of a “language”, to compare/contrast computation models
   - E.g.,

   "Language" of all possible open tag strings is regular

   "Language" of all XML strings is not regular, because a DFA cannot do open/close tag matching

A language is a set of strings.

if $A = \{w | M \text{ accepts } w\}$
Last time: “Closed” Operations

A set is **closed** under an operation if applying the operation to members of the set returns an element still in the set

- E.g., Natural numbers = \{0, 1, 2, \ldots\}
  - closed under addition,
  - **not** closed under subtraction
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 $A_1 \cup A_2$.

**Proof** (implement this algorithm for HW2)

- 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 a **new** machine $M = (Q, \Sigma, \delta, q_0, F)$ using $M_1$ and $M_2$

$M$ simulates running its input on both $M_1$ and $M_2$ in “parallel”; accept if either accepts
DFA Union Example

• $A_1 = \{ \text{"a"} \}$, $A_2 = \{ \text{"b"} \}$, $A_1 \cup A_2 = \{ \text{"a"}, \text{"b"} \}$

• $M_1 = (Q_1, \Sigma, \delta_1, \text{start}_1, F_1)$
• $M_2 = (Q_2, \Sigma, \delta_2, \text{start}_2, F_2)$

• $M$ recognizing $\{ \text{"a"}, \text{"b"} \} = (Q, \Sigma, \delta, \text{start}, F)$
  • $Q = Q_1 \times Q_2 = \{(q0, q3), (q0, q4), (q0, q5), \ldots\}$
  • $\Sigma = \{a, b\}$
  • $\delta((q0, q3), a) = (\delta_1(q0), \delta_2(q3)) = (q1, q5)$
  • ...
  • $\text{start} = (q0, q3)$
  • $F = \{(q1, q3), (q1, q4), (q1, q5), (q4, q0), \ldots\}$
**Last time:** Is Concatenation Closed?

**Theorem 1.26**

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 $A_1 \circ A_2$.

- **Proof:** Construct a new machine? (like union)
- How does $N$ know when to switch from $N_1$ to $N_2$?
- Can only read input once

100 Morrissey Blvd

$N$: CONCAT

$N_1$: recognize numbers

$N_2$: recognize words
Let $N_1$ recognize $A_1$, and $N_2$ recognize $A_2$.

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

$\varepsilon$ = “empty string” (ie, 0 length str) = transition that reads no input

Enables $N$ to run input on two machines that are at different input positions

$N$ must simultaneously:
- Keep checking with $N_1$ and
- Move to $N_2$ to check 2\textsuperscript{nd} part

But this is a different kind of machine. So is concatenation closed???
Nondeterminism

Deterministic computation

- start

- ... (states)

- accept or reject

Nondeterministic computation

- ... (reject)

- accept
Example Fig 1.27 (JFLAP demo): 010110
A **nondeterministic finite automaton** is a 5-tuple \((Q, \Sigma, \delta, q_0, F)\), where

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

• A power set is the set of all subsets of a set

• **Example:** \( S = \{a, b, c\} \)

• Power set of \( S = \)
  - \( \{\{\}, \{a\}, \{b\}, \{c\}, \{a, b\}, \{a, c\}, \{b, c\}, \{a, b, c\}\} \)
Formal Definition of “Computation”

• DFA (from before): Let \( w = w_1 w_2 \cdots w_n \)

\( M \) accepts \( w \) if a sequence of states \( r_0, r_1, \ldots, r_n \) in \( Q \) exists with three conditions:

1. \( r_0 = q_0 \),
2. \( \delta(r_i, w_{i+1}) = r_{i+1} \), for \( i = 0, \ldots, n - 1 \), and
3. \( r_n \in F \).

• NFA: Let \( w = y_1 y_2 \cdots y_m \)

\( N \) accepts \( w \) if a sequence of states \( r_0, r_1, \ldots, r_m \) exists in \( Q \) with three conditions:

1. \( r_0 = q_0 \),
2. \( r_{i+1} \in \delta(r_i, y_{i+1}) \), for \( i = 0, \ldots, m - 1 \), and
3. \( r_m \in F \).

This is now a set

Nondeterministic computation requires only one path to accept state in the computation tree
In-class exercise

• Come up with a formal description of the following NFA:

**Definition 1.37**

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

1. \(Q\) is a finite set of states,
2. \(\Sigma\) is a finite alphabet,
3. \(\delta : Q \times \Sigma_e \rightarrow \mathcal{P}(Q)\) is the transition function,
4. \(q_0 \in Q\) is the start state, and
5. \(F \subseteq Q\) is the set of accept states.
The formal description of $N_1$ is $(Q, \Sigma, \delta, q_1, F)$, where

1. $Q = \{q_1, q_2, q_3, q_4\}$,
2. $\Sigma = \{0, 1\}$,
3. $\delta$ is given as

4. $q_1$ is the start state, and
5. $F = \{q_4\}$.
So is Concatenation Closed for Reg Langs?

• Concatenation of DFAs produces an NFA

• But: A language is called a *regular language* if some DFA recognizes it.

• To show that concatenation is closed for regular languages, we must *prove* that NFAs also recognize regular languages.

• Specifically, we must *prove*:
  • NFAs $\Leftrightarrow$ regular languages
How to Prove a Theorem: $X \iff Y$

- $X \iff Y$ = “$X$ if and only if $Y$” = $X$ iff $Y$ = $X \iff Y$
- Proof at minimum has 2 parts:
  1. $\implies$ if $X$, then $Y$
     * i.e., assume $X$, then use it to prove $Y$
     * “forward” direction
  2. $\impliedby$ if $Y$, then $X$
     * i.e., assume $Y$, then use it to prove $X$
     * “reverse” direction
Proving NFAs recognize regular langs

**Theorem:**
- A language $A$ is regular if and only if some NFA $N$ recognizes it.

**Must prove:**
- $\Rightarrow$ If $A$ is regular, then some NFA $N$ recognizes it
  - Easy
  - We know: if $A$ is regular, then a **DFA** recognizes it.
  - Easy to convert DFA to an NFA! (how?)
- $\Leftarrow$ If an NFA $N$ recognizes $A$, then $A$ is regular.
  - Hard
  - Idea: Convert NFA to DFA
Need a way to convert NFA -> DFA

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. \(\delta : Q \times \Sigma \rightarrow Q\) is the transition function,
4. \(q_0 \in Q\) is the start state, and
5. \(F \subseteq Q\) is the set of accept states.

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

1. \(Q\) is a finite set of states,
2. \(\Sigma\) is a finite alphabet,
3. \(\delta : Q \times \Sigma_\varepsilon \rightarrow \mathcal{P}(Q)\) is the transition function,
4. \(q_0 \in Q\) is the start state, and
5. \(F \subseteq Q\) is the set of accept states.

**Proof idea:**
Each “state” of the DFA must be a set of states in the NFA
In a DFA, all these states at each step must be only one state.

So design a state in the converted DFA to be a set of NFA states!
Example:

**Figure 1.42**
The NFA $N_4$

**Figure 1.43**
A DFA $D$ that is equivalent to the NFA $N_4$
Next time: Convert NFA -> DFA, Formally

- Let $\text{NFA } N = (Q, \Sigma, \delta, q_0, F)$

- Then equivalent DFA $M$ has states $Q' = \mathcal{P}(Q)$ (power set of $Q$)

- (implement this algorithm for HW3)
Check-in Quiz 2/8

On gradescope

In-Class Survey

See course website