Finite Automata

Motivation

An Example

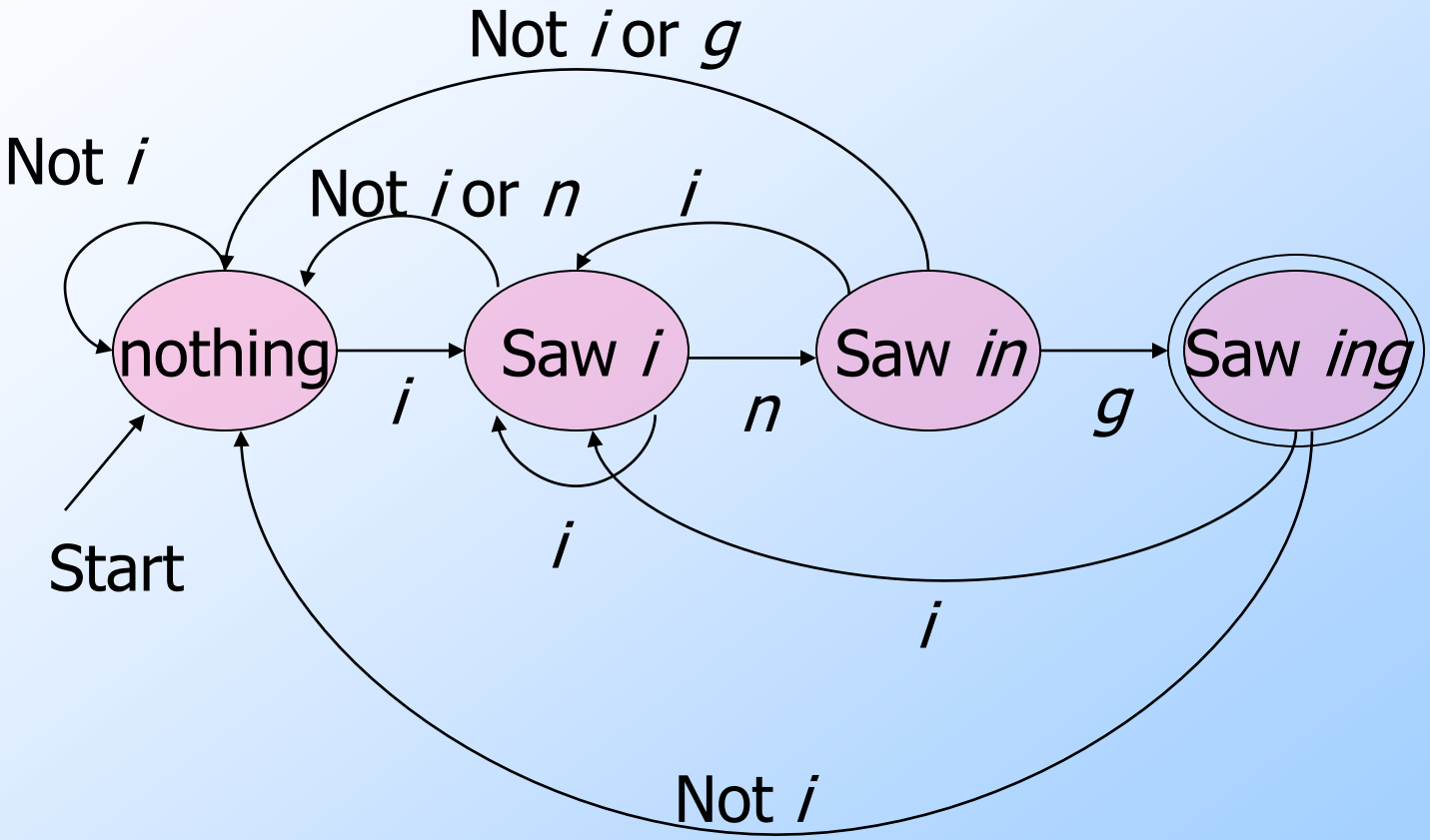
Informal Explanation

- Finite automata are finite collections of states with transition rules that take you from one state to another.
- Original application was sequential switching circuits, where the "state" was the settings of internal bits.
- Today, several kinds of software can be modeled by FA.

Representing FA

- Simplest representation is often a graph.
  - Nodes = states.
  - Arcs indicate state transitions.
  - Labels on arcs tell what causes the transition.

Example 1: Recognizing Strings Ending in "ing"



Automata to Code (by hand)

- In C/C++/Java:
  1. Initialize state q to start state.
  2. Loop through the string one character at a time.
  3. Make a switch statement with a case for each state for q, where each case sets q according to the transitions for that state.
  4. Accept if you end in a final state.

Example in Java

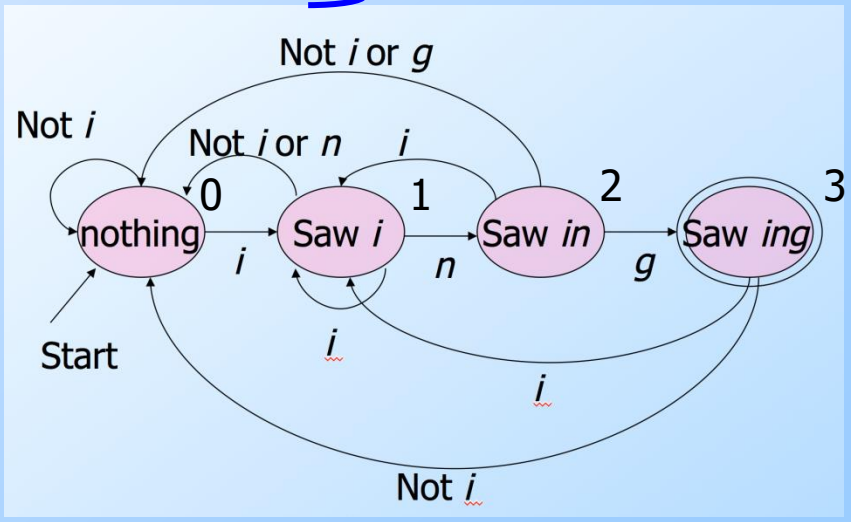
```
Scanner scan = new Scanner(System.in);
String s = scan.next();
int q = 0;
for (char c : s.toCharArray()) {
  switch (q) {
    case 0: q = (c=='i')? 1  : 0; break;
    case 1: q = (c=='n')? 2  : ((c=='i')? 1  : 0); break;
    case 2: q = (c=='g')? 3  : ((c=='i')? 1  : 0); break;
    case 3: q = (c=='i')? 1  :  0;
            }
      }
if (q==3)
  System.out.println("accept.");
else
  System.out.println("reject.");
```

Start state

Loop through string s

Transitions

Final state



Example 1: Automata to Code

```
2: /* i seen */
  c = getNextInput();
  if (c == 'n') goto 3;
  else if (c == 'i') goto 2;
  else goto 1;
3: /* "in" seen */
  . . .
```

Automata to Code – General

- It would be nice to have an automatic way to generate such code…
- Rather than do it by hand, a code generator takes a “regular expression” describing the pattern(s) you are looking for and produces the code for it.
  - Example: `.*ing` works in grep.

Example 2: Set of all strings that start with 0 over {0,1}

Example 3: Set of all strings of length 2 over {0,1}

General Comments

- Some things are easy with finite automata:
  - Substrings (…abcabc…)
  - Subsequences (…a…b…c…b…a…)
  - Modular counting (odd number of 1's)
- Some things are impossible with finite automata (we will prove this later):
  - An equal number of a's and b's
  - More 0's than 1's