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.

Automata to Code

- In C/C++, make a piece of code for each state. This code:
  - Reads the next input.
  - Decides on the next state.
  - Jumps to the beginning of the code for that state.

Example: Protocol for Sending Data

- How would you do this in Java, which has no goto?
- You don't really write code like this.
- Rather, a code generator takes a "regular expression" describing the pattern(s) you are looking for.
  - Example: \.ing works in grep.
**Extended Example**

- Thanks to Jay Misra for this example.
- On a distant planet, there are three species, a, b, and c.
- Any two different species can mate. If they do:
  - The participants die.
  - Two children of the third species are born.

**Strange Planet – (2)**

- Observation: the number of individuals never changes.
- The planet *fails* if at some point all individuals are of the same species.
  - Then, no more breeding can take place.
- State = sequence of three integers – the numbers of individuals of species a, b, and c.

**Strange Planet – Questions**

- In a given state, must the planet eventually fail?
- In a given state, is it possible for the planet to fail, if the wrong breeding choices are made?

**Strange Planet – Transitions**

- An a-event occurs when individuals of species b and c breed and are replaced by two a’s.
- Analogously: b-events and c-events.
- Represent these by symbols a, b, and c, respectively.

**Strange Planet with 2 Individuals**

<table>
<thead>
<tr>
<th>State</th>
<th>a</th>
<th>b</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>002</td>
<td>a</td>
<td></td>
<td></td>
</tr>
<tr>
<td>001</td>
<td></td>
<td>b</td>
<td></td>
</tr>
<tr>
<td>011</td>
<td></td>
<td></td>
<td>c</td>
</tr>
</tbody>
</table>

Notice: all states are "must-fail" states.
Strange Planet with 3 Individuals

![Strange Planet with 3 Individuals diagram]

Notice: four states are "must-fail" states. The others are "can't-fail" states. State 111 has several transitions.

Strange Planet with 4 Individuals

![Strange Planet with 4 Individuals diagram]

Notice: states 400, etc. are must-fail states. All other states are "might-fail" states.

Taking Advantage of Symmetry

- The ability to fail depends only on the set of numbers of the three species, not on which species has which number.
- Let’s represent states by the list of counts, sorted by largest-first.
- Only one transition symbol, x.

The Cases 2, 3, 4

![The Cases 2, 3, 4 diagram]

Notice: for the case n = 4, there is nondeterminism: different transitions are possible from 211 on the same input.

5 Individuals

![5 Individuals diagram]

Notice: 500 is a must-fail state; all others are might-fail states.

6 Individuals

![6 Individuals diagram]

Notice: 600 is a must-fail state; 510, 420, and 321 are can't-fail states; 411, 330, and 222 are "might-fail" states.
Notice: 700 is a must-fail state; All others are might-fail states.

Questions for Thought

- Without symmetry, how many states are there with $n$ individuals?
- What if we use symmetry?
- For $n$ individuals, how do you tell whether a state is “must-fail,” “might-fail,” or “can’t-fail”?