Activity 11

Treasure hunt—*Finite-state automata*

Age group	Middle elementary and up.
Abilities assumed	Simple map reading.
Time	15 minutes or more.
Size of group	Around ten or more.

Focus

Maps.

Abstract representations.

Recognizing patterns.

Summary

Computers programs often need to process a sequence of symbols. The symbols might be letters or words in a document, or the text of another computer program. Computer scientists often use a simple but powerful idea called a finite-state automaton to process these symbols. A finite-state automaton involves little more than following simple instructions on a map. This activity shows how the idea works, and what it can be used for.

Technical terms

Finite-state automata; languages; parsing; compilers.

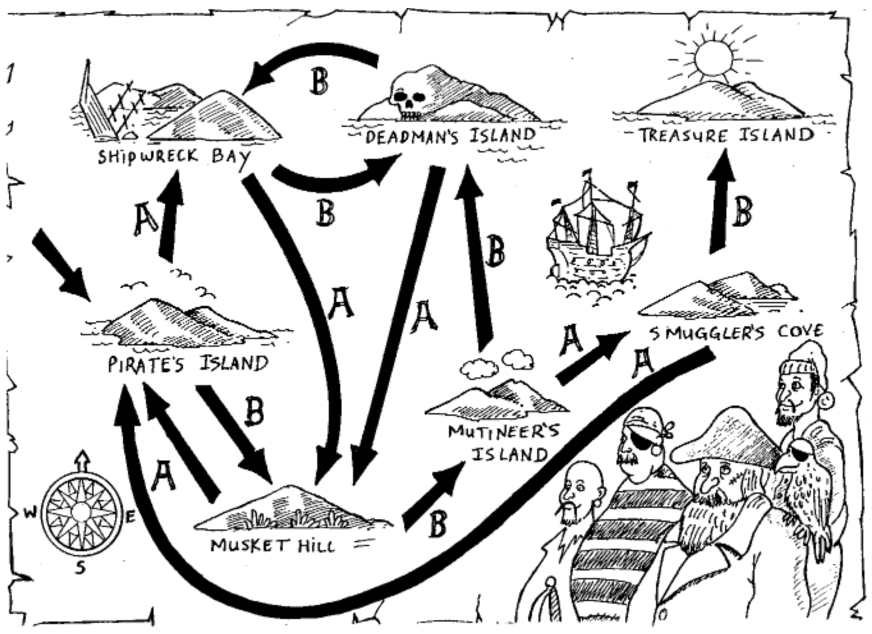


Figure 11.1: The final map

Materials

Each child will need:

a copy of the blackline master on page 116, and

a pen or pencil.

You will need:

one set of cards made from the blackline master on page 117,

cards or stickers to label the people at the islands with the names on page 116,

prizes (such as a stamp or stickers), and

blank paper, cards, and labels for creating new maps.

What to do

In this activity, seven people are positioned around the room (or further apart if outdoors), each representing an island. Pirate ships sail along a fixed set of routes between the islands (these are very well-organized pirates who offer a scheduled shuttle service for children). The seven "islands" are labeled with the names on page 116, either by holding a card or wearing a sticker or badge.

Each island has two departing ships, A and B. Figure 11.1 shows a map of the routes available. Unfortunately the pirates don't let children have this map because they want to keep the route to Treasure Island secret. The goal is to find a sequence of ship rides that will get from Pirate's Island to Treasure Island. The only way to find this is for the children to "travel" between islands, until they find a route to Treasure Island. For example, the sequence of choices AABAAAB is one of many that achieves the goal for Figure 11.1.

The children start by going to the person representing Pirate's Island, and saying whether they would like to take route A or B. The people looking after the islands have the corresponding instruction card from the blackline master on page 117. They use the card to tell arriving children where their chosen route will take them. For example, a child arriving at Pirate's Island who asks to take route A will be sent to Shipwreck Bay. The child continues to move around the islands by asking for route A or B. The children start with a blank map—the blackline master on page 116—and use it to note where each route takes them as they move between the islands. Without their record of the routes, they could easily end up going around in circles!

Here is a step-by-step description of the activity.

1. First demonstrate what the children are required to do by using a small example.

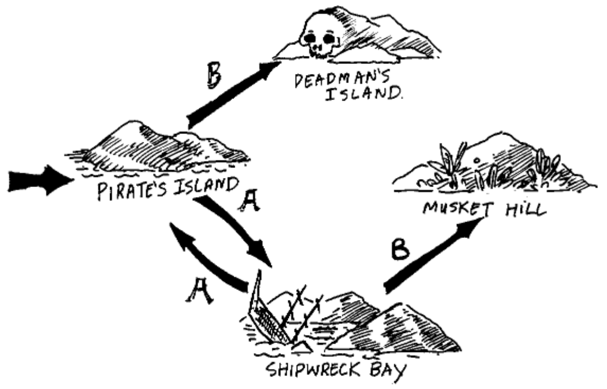
   A map with four islands, only two of which have associated directions, is shown in Figure 11.2a, and the cards for the islands are in Figure 11.2b. Choose two children (or, preferably, adult helpers) and give them the example cards and the labels *Pirate's Island* and *Shipwreck Bay*. Have them stand some distance apart. Show all the children a copy of page 116 (but do not hand it out yet) and explain that

   - it is a map of some islands, with a choice of two routes, A and B, from each one,
   - the goal is to find a route to get to Treasure Island, and
   - there is a prize at Treasure Island for getting there with a correct route marked on the map.

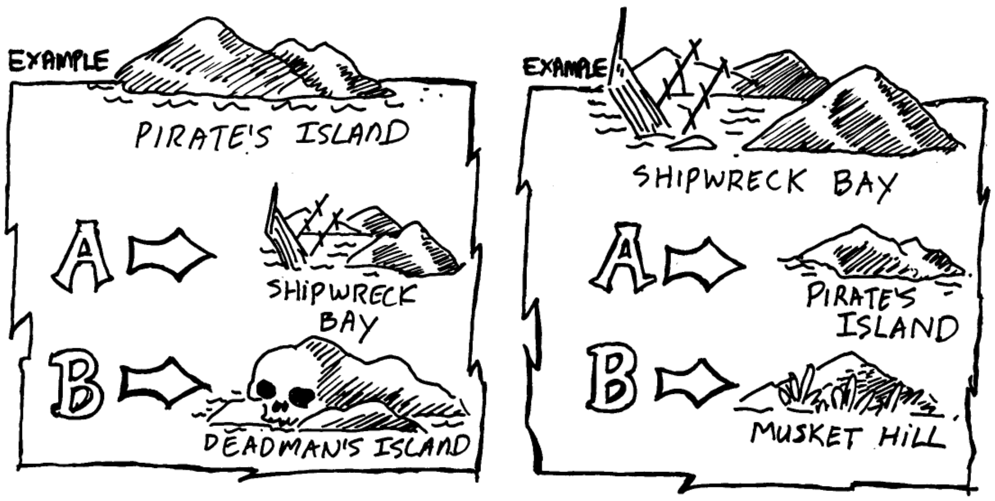
   Demonstrate the game by going to the person representing Pirate's Island and asking for route A. The person at Pirate's Island should direct you to Shipwreck Bay. Show the children how to mark this on the map by drawing an arrow from Pirate's Island to Shipwreck Bay, labeled with an A. At Shipwreck Bay, ask for route A again. You should be directed back to Pirate's Island. Mark this route on the map.

   Emphasize that they need to write down the routes as they go along—in the excitement children can forget about recording routes, and become hopelessly lost. Also, they often forget to label the route with A or B, which makes things difficult if they need to re-use the route later.

   The routes in the small example of Figure 11.2 are different from those in the main activity, so collect in the example cards to ensure that they don't cause confusion with the next map that will be used.



(a)



(b)

Figure 11.2: A small example: (a) map, (b) cards

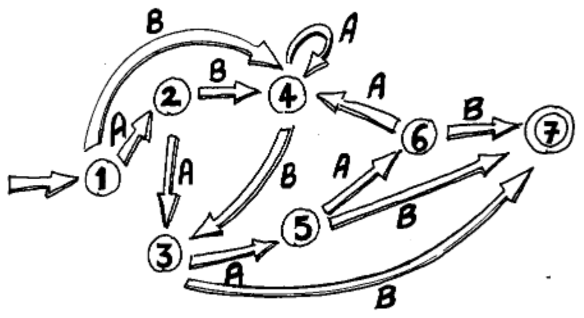


Figure 11.3: A map with a loop

2. Now it's time to set up all seven islands with a new set of routes, and have the children attempt to navigate their way to Treasure Island.

   Give out the instruction cards (from the blackline master on page 117) to the people who will be the islands, and position them around the room (or around a playground). It is safer if these people are adults, but older children will catch on quickly enough. A seventh "island" (Treasure Island) is required to check maps and administer prizes. Give out the blank maps (page 116) and send the children off one at a time to start at Pirate's Island.

   The people at the islands should check that children are writing down the routes as they move around. The person at Treasure Island needs to check that their map is correct (it needn't be complete, but it must show at least one path from Pirate's Island to Treasure Island). Correct maps receive a prize, which might be a stamp (see page 6) or sticker.

   Children who finish quickly can be asked to find an alternative route.

3. After everyone has found a route, gather the children together to discuss the routes they found to get to the treasure. A complete map (Figure 11.1) should be constructed on the classroom board, and the children can trace other routes. Point out routes that involve loops—for example, BBAAAAAB and BBAAAAAAAB both get to Treasure Island.

4. The game can be repeated with different layouts. Figure 11.3 shows an alternative way of expressing a map. The islands are shown as numbered circles, and the final island (with the treasure) has a double circle. Notice the loop from island number 4 back to itself. This may seem pointless, but we will explore uses for it later. The children moving around the islands can construct a map on a blank piece of paper this time, drawing the islands as a number in a circle.

   Children will probably enjoy designing their own layouts and creating the cards for the islands.

5. Back at the classroom board, the children can explore different maps. Figure 11.4 shows three that can be discussed. For each of the sample maps they should try to work out

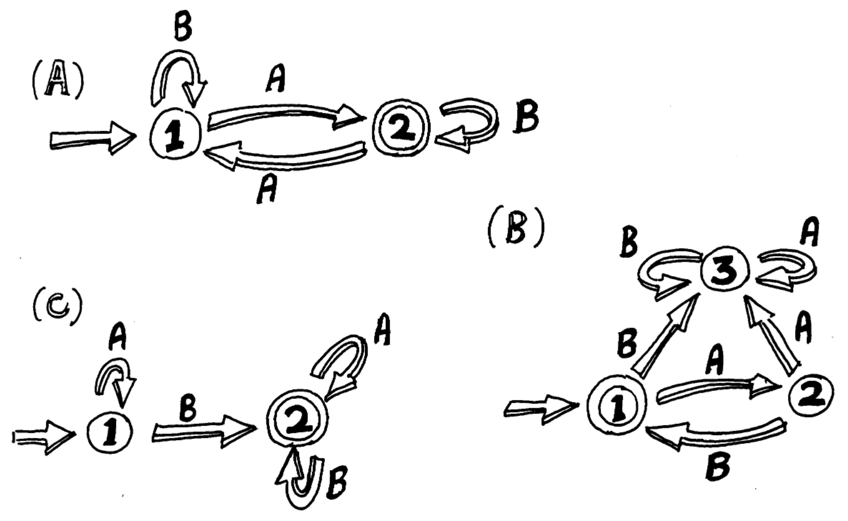


Figure 11.4: Sample maps for classroom discussion

a general description of the sequence of routes that get to the island with the double circle. The map in Figure 11.4a will finish at the double circle (island 2) only if the sequence has an odd number of A's (for example, AB, BABAA, or AAABABA). The one in Figure 11.4b only gets to the double circle with a strictly alternating sequences of A's and B's (AB, ABAB, ABABAB, . . .). The one in Figure 11.4c requires that the sequence contains at least one B (the only sequences not suitable are A, AA, AAA, AAAA, . . .).

Variations and extensions

Instead of having the children discover the map, each could be given a different sequence of A's and B's, and they could report which island they ended up at, receiving a reward if their answer is correct.

Various games and puzzles can be based on this kind of "map." For example, Figure 11.5 illustrates a way of constructing sentences by choosing random paths through the map and noting the words that are encountered.

The following puzzle has a solution that can be represented clearly by a map of the type we have been using. Figure 11.6a shows a sequence of coin tosses supposedly produced by an automatic coin-tossing machine. We want to find out if the machine is rigged. It looks like there may be a pattern in the tosses, but it is hard to see what it is. Show the sequence to the class, and see if they can identify a predictable pattern.

The pattern in Figure 11.6a can be explained using the map in Figure 11.6b. Following the sequence, the first *h* takes you from island 1 to island 2. The second *h* goes from 2 to 4. Notice that there is no route from island 2 on a *t*. This means we would be stuck if a *t* ever came up while we were at island 2. Tracing through the sequence in Figure 11.6a reveals that we never

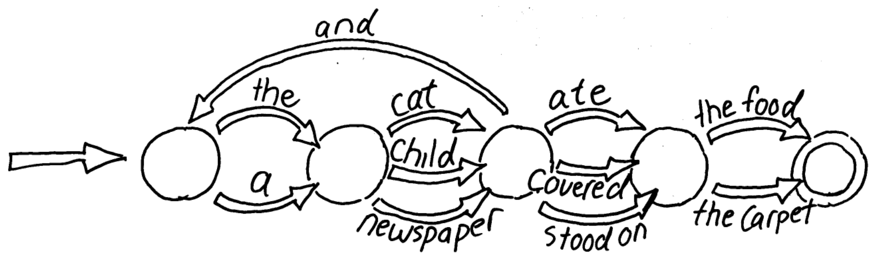
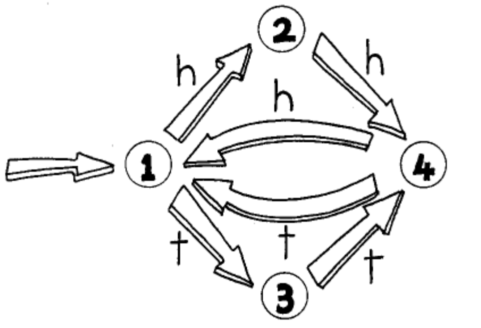


Figure 11.5: A map for generating sentences

```
h h t h h t h h h t t h h h h t t h t t t h h h h h t h h h t t t
h h h t t t h h h h h h t t h t t t t t h t t h t t t h h h t t h
h h t h h h h h h h h h t t h h h t t t t h h h h h t t t t t t t
```

(a)



(b)

Figure 11.6: (a) A sequence of coin-tosses; (b) a finite-state automaton that explains it

get stuck like this, and so the pattern is now revealed in the structure of the map: the first two coin tosses of every group of three will be the same. Every third toss is completely predictable! (You can see this from the sequence too: every third toss, starting with the second symbol in the sequence, is a copy of the previous one.)

What's it all about?

The "maps" used in this activity are a kind of representation that computer scientists call a *finite-state automaton*, or FSA for short. They are usually represented on paper with diagrams like the ones in Figure 11.4. We say that they are made up of *states* (the circles) and *transitions* (the arrows). The goal is to get to an *accepting state* (the double circle). An FSA begins in a specified starting state, and reads symbols from its input, one by one. After reading each symbol it moves to the next state indicated by the transition labeled with that symbol. For example, in Figure 11.3, suppose the input is ABAABB. We start in state 1, and the first A takes us to state 2. The B takes us to state 4, and the next two A's keep us in state 4. The next B takes us to state 3, and the final B takes us to state 7.

The double circle around state 7 means that we "accept" any input that leads to that state. Therefore the input ABAABB is accepted. The input ABAAAA would be rejected—it doesn't fit the definition of an "acceptable" input.

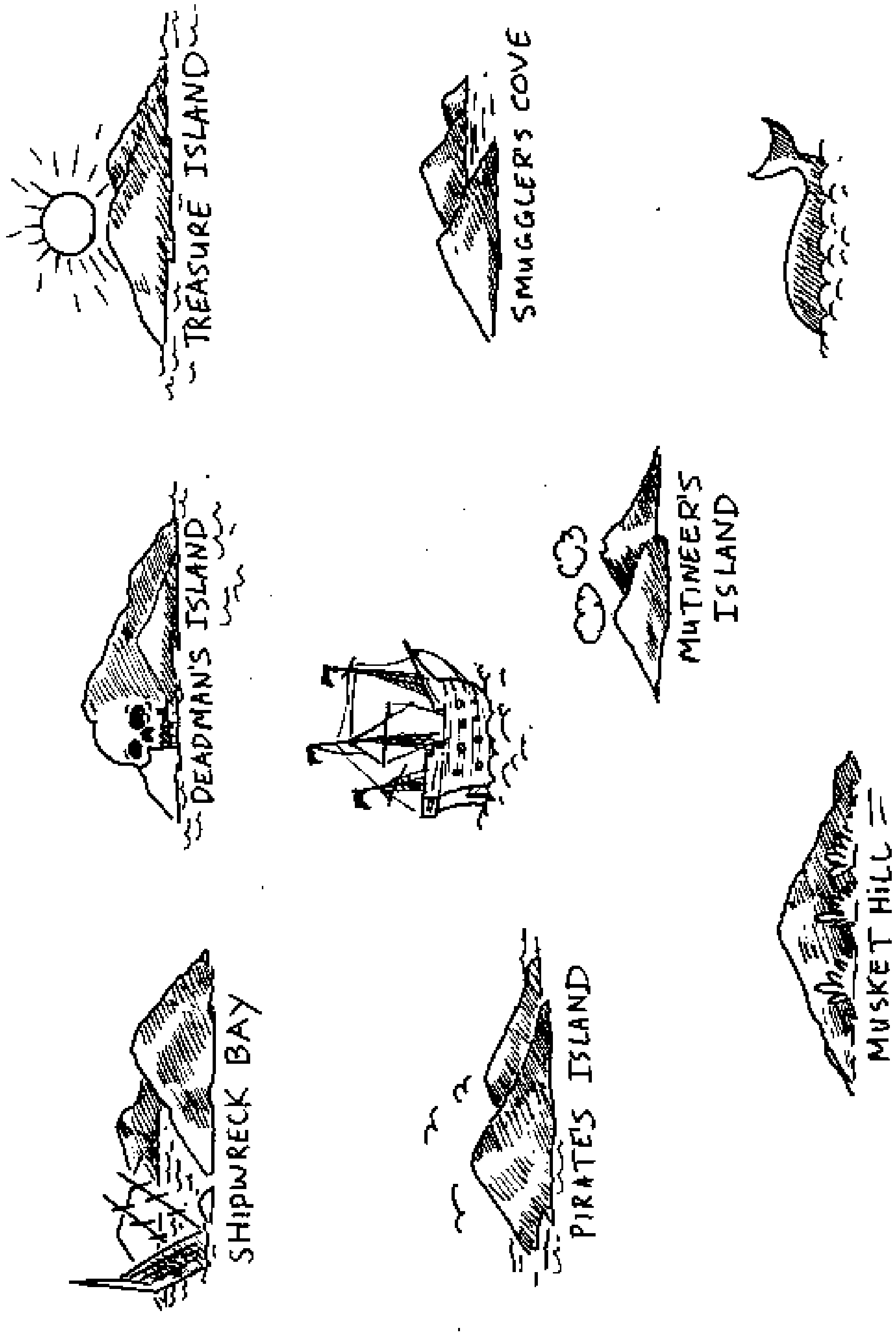
Although the examples have mainly used just A's and B's, FSAs normally work with all letters of the alphabet, as well as other characters such as digits and punctuation. They aren't usually built as real machines, but are simulated by computer programs instead. They are mainly useful as a tool for thinking about recognizing patterns in the computer's input. For example, a compiler is a program that converts programs from languages such as Pascal and C to a form that the computer can use. Compilers use finite-state automata to recognize parts of the language—such as numbers and keywords—in the input.

Finite-state automata are also used to explain or generate output. Suppose you want to get a computer to imitate the way that people can learn the structure of sequences of symbols from just a few examples. You might instruct the computer to construct a finite-state automaton that represents the patterns in the input, and can generate more output symbols that fit the pattern. The coin-tossing puzzle was an example of this: the automaton in 11.6b was generated by feeding the sequence of Figure 11.6a into a program that looks for small FSAs that could generate the sequence. This is a simple form of learning: the computer has been fed some raw information, the example sequence, and has generated a concise description or "explanation" for it that could be used to predict more symbols in the sequence. Finite-state automata are used in data compression to make predictions of what character will appear next in the input so that they don't have to be transmitted explicitly. They are also useful in human–computer interaction, where the transitions represent a series of interactions between a person and a computer.

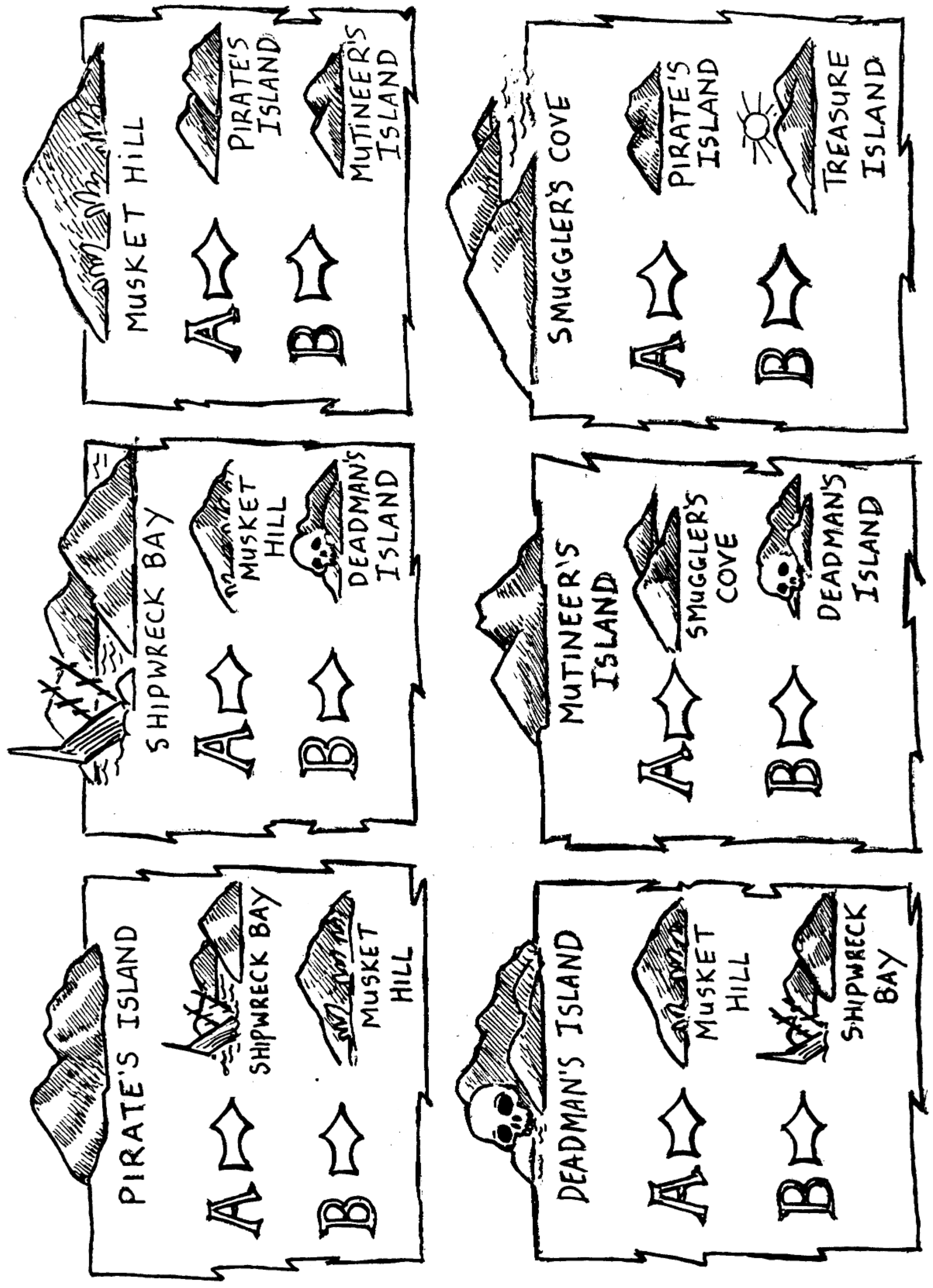
FSAs aren't the only sort of abstract machines that computer scientists use. Other models are used for different purposes, with names such as *push-down automaton*, *LR parser*, and *Turing machine*. Like the FSA, each is simple to describe, and represents a new way of approaching problems that enable us to deal with their complexity a lot more easily.

Further reading

Harel's *Algorithmics* and Dewdney's *Turing Omnibus* both have sections on finite-state automata. The rigged coin-tossing machine is introduced in the book *Thinking with a teachable machine* by John Andreae. A thorough technical treatment of the area is given by Hopcroft and Ullman in *Introduction to Automata Theory, Languages, and Computation*.



Instructions: *Start at Pirate's Island, and ask for either the A or B route. Write down the routes that you discover, and keep going until you find a way to get to Treasure Island.*



Instructions: *Give one of these cards to each of the people who is looking after an "island."*