Finite State Acceptors

We’ve now seen three ways to present formal languages: recursive definitions, rewrite rules, and finite-state automata. We’re going to confuse you with a fourth: finite-state acceptors. They’re confusing because they’re also called “FSAs” and they look an awful lot like finite-state automata. To help you, we’ll call finite state acceptors “fsas.”

With finite-state automata, every path through the diagram is a legal wff and nothing else is a legal wff. In that sense, finite-state automata generate all the correct wffs. But if you’re given a string of characters and are asked “Is this a grammatically correct wff?” it’s hard to use the automaton to answer. You have to keep testing paths through the automaton, looking for a path to generate the string. If you find one, great, you know it’s a wff. If you don’t, then it’s not clear. How do you know you’ve looked hard enough? How do you know there isn’t a path that escaped your notice?

Finite-state acceptors work differently. You start with a string. You move from state to state depending on the characters in the string. If you end up in an “accepting” state, then you know the string is a wff. If you end up in a “non-accepting” state then you know the string is not a wff.

Here’s an example. Let’s say the alphabet of the language is $a$ and $b$. We have an initial state, which we’ll call $I$, and two other states, $A$ and $B$. Let’s say $B$ is accepting. We mark accepting states with a double circle. Here’s the diagram:

![Finite State Automaton Diagram]

Notice that leaving each state we have exactly one “a” arrow and one “b” arrow.
Let’s test $abbab$ to see if it’s a legal wff. We start in state $I$. Then we read off the letters one-by-one starting from the left. First we have $a$ and we go to $A$.

then to $B$, then to $I$,

then to $A$ and to $B$ where it finishes.

Since $B$ is an accepting state, $abbab$ is a legal wff.
If a string, when submitted to a finite state acceptor, terminates at an accepting state, we’ll say that the fsa accepts it. Note: finite state acceptors can have more than one accepting state.

Let’s call a language $L$ hunky-dory if there’s a finite state acceptor that accepts all and only the wfs of $L$.

**Exercises:**

1. Construct an fsa for the language where the only words are $x$, $yx$, $yyx$, $yyyyx$, ... (All $y$s except for an $x$ at the end.)

2. Construct an fsa for the language where the only words are $a$, $aaa$, $aaaaa$, ... In other words, strings with an odd number of repetitions of $a$.

3. Construct an fsa for the language where the only words are $abc$, $abbc$, $abbbc$, $abbbbc$, ...

4. Find a set of rewrite rules for the language in exercise 1.

5. Find a set of rewrite rules for the language in exercise 2.

6. Find a set of rewrite rules for the language in exercise 3.

All but three of the following languages are hunky-dory. Find which are hunky-dory and construct an fsa for each.

7. Consider the language consisting of nothing but ‘buffalo.’ A wff in this language is any legitimate English sentence.

8! Consider the language consisting of just $\Box$ and $\triangle$. A wff in this language is one where there is exactly one $\Box$.

9! Consider the language consisting of nothing but ‘bison’ and ‘intimidate.’. A wff in this language is any legitimate English sentence. Is this language hunky-dory? If so, find an fsa for it.

10!! Consider the language consisting of nothing but left and right parentheses. A wff in this language is one where the parentheses are paired and nested; that is, 

$$(), (()), ()()$$

are wffs, but

$$((),(),(), and (((())))()$$

are not. Is this language hunky-dory? If so, find an fsa for it.

11!!! Consider the language consisting of just $\heartsuit$ and $\spadesuit$. A wff in this language is one where there are equal numbers of $\heartsuit$’s and $\spadesuit$’s. Is this language hunky-dory? If so, find an fsa for it.
Suppose we use only the letters \( a \) and \( b \).

12. Can you construct an FSA so that the legitimate words are exactly those words with no \( a \)'s?

13. Can you construct an FSA so that the legitimate words are exactly those words with no two \( a \)'s in a row?

14. Can you construct an FSA so that the legitimate words are exactly those words with no isolated \( a \)'s (e.g., \( bab \) is not allowed)?

15. Can you construct an FSA so that the legitimate words are exactly those words with at least one \( a \)?

16. Can you find an FSA for the language represented by the FSA in the text?

17. Can you find a set of rewrite rules for the language represented by the FSA in the text?
Answers

1. $I \xrightarrow{y} A$
2. $B \xrightarrow{y} I$
3. $A \xrightarrow{y} I$
4. $B \xrightarrow{y} A$
5. $A \xrightarrow{y} B$

7. buffalo

9. Not hunky-dory

11. Not hunky-dory

13.

15.

1. $S \rightarrow Bb$
2. $B \rightarrow Ch$
3. $B \rightarrow$
4. $B \rightarrow Da$
5. $C \rightarrow Bh$
6. $D \rightarrow Da$
7. $D \rightarrow E$
8. $E \rightarrow Eh$
9. $E \rightarrow D$
10. $D \rightarrow$