Applications of Finite Automata

Applications of finite automata include string matching algorithms, network protocols and lexical analyzers.
Consider finding all occurrences of a short string (*pattern string*) within a long string (*text string*). This can be done by processing the text through a DFA: the DFA for all strings that *end* with the pattern string. Each time the accept state is reached, the current position in the text is output.
Example: Finding $1001$

To find all occurrences of pattern $1001$, construct the DFA for all strings ending in $1001$. 

![DFA Diagram]

$q_0 \xrightarrow{1} q_1 \xrightarrow{0} q_2 \xrightarrow{0} q_3 \xrightarrow{1} q_4$
$q_0 \xrightarrow{0}$
$q_1 \xrightarrow{1}$
$q_2 \xrightarrow{1}$
$q_3 \xrightarrow{0}$
$q_4 \xrightarrow{1}$

Goddard 5: 3
A finite-state machine is an FA together with actions on the arcs.

A trivial example for a communication link:

\begin{center}
\begin{tikzpicture}[node distance=2cm,auto,>=latex]
  \node (idle) [state] {idle};
  \node (wait) [state, right of=idle] {wait};
  \draw[->] (idle) edge node {send packet} (wait);
  \draw[->] (wait) edge node {receive ack} (idle);
  \draw[->] (wait) edge [loop above] node {if NAK then resend} (wait);
\end{tikzpicture}
\end{center}
A **bot** is a computer-generated character in a video game.

Note that using finite-state machine allows automation.
Statecharts model tasks as a set of states and actions. They extend FA diagrams.

Here is a simplified statechart for a stopwatch.
In compiling a program, the first step is *lexical analysis*. This isolates keywords, identifiers etc., while eliminating irrelevant symbols.

A *token* is a category, for example “identifier”, “relation operator” or specific keyword.

For example,

```
token RE
keyword then then
variable name [a-zA-Z] [a-zA-Z0-9]*
```

where latter RE says it is any string of alphanumeric characters starting with a letter.
A lexical analyzer takes source code as a string, and outputs sequence of *tokens*.

For example,

```plaintext
for i = 1 to max do
    x[i] = 0;
```

might have token sequence

```plaintext
for id = num to id do id [ id ] = num sep
```

As a token is identified, there may be an action. For example, when a number is identified, its value is calculated,