String searching looks to find a specific string as a substring of another string.

Pattern searching is when we want to look for a substring that is any member of a set of strings (potentially a set of infinite size).

Regular grammars, regular expressions and FSMs

The set of strings might be generalised/characterised as a pattern expression, called a "regular expression". A regular expression is a single string that corresponds to a regular grammar, which itself can be modeled as a finite state machine (FSM) for recognising strings (i.e. a language). Such a FSM can be implemented using three arrays, where each state is an index into all arrays such that one array indicates any character to be matched by the FSM at that state or marks the state as a branching state, and the other two arrays hold the indices of the one or two possible next states if a match is successful.

Parsing with a regexp

Searching for a pattern, or "parsing", is carried out with that FSM by using an abstract data type called a double-ended queue (also known as a dequeue or a deque). The front of the deque holds a list of all possible current states, and the end of the list holds all possible next states for when a current state is able to match a symbol of input. These two lists are separated by a token called the SCAN.

Details about regular grammars, regular expression, FSM string-recognizers and the process of pattern searching are included in the following lecture notes: this old URL link is broken, see WORD file attached below
http://old-www.cms.waikato.ac.nz/~tcs/COMP317/rgfsm.html

Context-free grammars and compilers

Before a FSM machine can be constructed for a regexp, the expression must first be checked to see if it is well-formed. That is, does the sequence of symbols actually represent a regexp? This suggests that regexps are themselves a language (i.e. a subset of all possible strings). Unfortunately, the language of regular expressions is not expressible as a regular grammar (if for no other reason than a regexp must have balanced-brackets). But the language can be expressed with a slightly stronger formalism called a context-free grammar. Strings for a context-free grammar can be recognised with a pushdown automaton ... basically a state machine with some stack memory. High level programming language are themselves context-free, and we can use this to our advantage to write a program which "parses" a string and checks to see if it conforms to a particular CFG (in this case, the CFG for all regexps). We can modify/extend such a "parser" so that it also builds a FSM for the regexp described by the string, and this results in what is known as a "compiler".

Details about context-free grammars, parsing and compiling of regexps can be found in the following lecture notes: this old URL link is broken, see WORD file attached below http://old-www.cms.waikato.ac.nz/~tcs/COMP317/cfgparsing.html