

How flex works

Regex $\rightarrow$ finite automaton

Def: A deterministic finite automaton (DFA) is

- a finite set Q of "states"
- a transition function $\delta(q, a) = q'$

δ $\uparrow$ $\uparrow$ $\uparrow$
 state char state
 state $\uparrow$ state

- one state $q_0 \in Q$ is designated the start state
- set of states $F \subseteq Q$ is the set of accepting states (final states)

Behaviour:

- discrete, state transition one per char read.

Jan 31-11:01 AM

The diagram shows a horizontal sequence of boxes representing states, labeled 'H', 'e', 'l', 'l', 'o', followed by an ellipsis. An arrow labeled 'next char read' points to the 'H' box. Below the 'H' box is a box labeled 'q0'. A bracket under the 'H' box is labeled 'to read'. Below the 'q0' box is the text '"initial configuration"'.

loop:

- state q initially
- read head scanning a_i

actions:

- state changes to $\delta(q, a) = r$
- read head advances to next char

Ex: $\delta(q_0, H) = r$, then after one step:

The diagram shows the 'H' box with an arrow pointing down to a box labeled 'r'.

δ is a "partial fun"

$\delta(q, a)$ may be undefined for certain pairs (q, a) .

Jan 31-11:11 AM

DFA dies (stops) without accepting if some pair (q, a) is encountered where $\delta(q, a)$ is undef.

Example:

$[0-9]^+$ (int const)

Transition diagram:

The diagram shows a start state q0 with a self-loop labeled '0' and an arrow to an accepting state q1 labeled '1'. There is also a self-loop on q1 labeled '1'.

$\delta(q_0, 0) = \delta(q_0, 1) = \dots = \delta(q_1, 0) = \delta(q_1, 1) = q_1$

$\rightarrow$ start state
 $\odot$ accepting state

At any stage let x be the string read so far. If DFA is in accepting state, then we say the DFA accepts x . Otherwise the DFA rejects x (if x is either in a non-accepting state or it died before reading all of x).

Jan 31-11:17 AM

Real constants in Pascal:

$[0-9]^+ \cdot [0-9]^+ \cdot ([E|e][+-]?[0-9]^+)?$

The diagram shows a complex DFA with multiple states and transitions, including a loop for the decimal part.

determinism: no two edges out of the same state with same label.

Accepting binary strings with an even number of 0s

The diagram shows a DFA with two states: a start state q0 and an accepting state q1. Transitions: q0 to q0 on '1', q0 to q1 on '0', q1 to q1 on '1', q1 to q0 on '0'.

ab^* :

 The diagram shows a DFA with two states: a start state q0 and an accepting state q1. Transitions: q0 to q1 on 'a', q1 to q1 on 'b'.

ab^+ :

 The diagram shows a DFA with three states: a start state q0, an intermediate state q1, and an accepting state q2. Transitions: q0 to q1 on 'a', q1 to q1 on 'b', q1 to q2 on 'b'.

Jan 31-11:28 AM

$b^*abb[ab]^+$

The diagram shows a DFA with four states: a start state q0, a state q1, a state q2, and an accepting state q3. Transitions: q0 to q1 on 'b', q1 to q1 on 'b', q1 to q2 on 'a', q2 to q3 on 'b', q3 to q3 on 'a' and 'b'.

For each rule:

- exp action_i
- exp_j action_j
- exp_n action_n

For each exp_i flex builds a DFA D_i that accepts strings matched by exp_i (and no others).

When scanner is executed, all the D_i are simulated on the input simultaneously.

initially: nothing read on screen

- all D_i in start states
- all D_i are active (alive) (no best matching string yet)

while (not eof and there are still active DFAs) do read next char s

Jan 31-11:45 AM



Jan 31-11:54 AM

```

for each active DFA  $D_i$ 
  (in order of increasing  $i$ )
  if can't advance  $D_i$ 
    reading  $s$  then
    kill  $D_i$  (dead state)
  else
    advance  $D_i$  to next
    state reading  $s$ 
    if current state of  $D_i$ 
    is accepting then
      mark input read so far
      (not incl.  $s$ ) as best match
      (associated with  $D_i$ )
  end-for
end-while
// either eof or all  $D_i$ 
// are dead
If there is a best match
for some  $D_i$ , then {
  execute action;
  start again for
  with input following match
}
else
  no prefix of the
  input matches any
  exp.; quit.

```

Jan 31-11:54 AM

```

flex-produced scanner:
if no match
  reads one char,
  echoes it to output,
  starts again.
^
As if flex add a
hidden "default" rule:
. ECHO;
as last rule.

```

Jan 31-12:06 PM