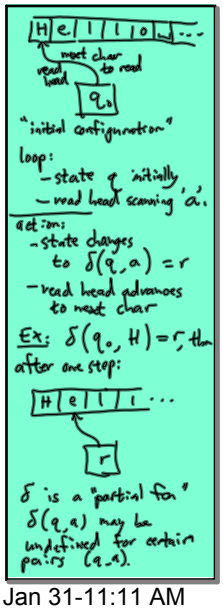
How flex works

Regexp → finite automaton

Def: A deterministic finite automaton (DFA) is
- a finite set Q of "states"
- a transition function $\delta(q, a) = q'$ (state, character → 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)

Behavior:
- discrete, state transitions, one per char read.

Jan 31-11:01 AM



"initial configuration"

loop:
- state q initially
- read head scanning 'a'.

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

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

δ is a "partial fcn"
$\delta(q, a)$ 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:
$[0\text{-}9]+$ (int consts)

Transition diagram:

$\delta(q_1, 0) = \delta(q_1, 1) = \cdots = \delta(q_1, 9) = q_1$.

→○ start state
◎ accepting state

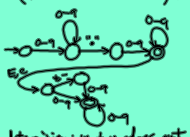
At any stage let x be the string read so far. If DFA is in accepting state, then we say the DFA accepts x. Otherwise the DFA rejects x (if DFA is either in a nonaccepting state or it died before reading all of x)

Jan 31-11:17 AM

Real constants in Pascal:

$[0\text{-}9]+"."[0\text{-}9]+$

$\left([eE][+-]?[0\text{-}9]+\right)?$

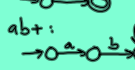


determinism: no two edges out of the same state with same label.

Accepting binary strings with an even number of 0s

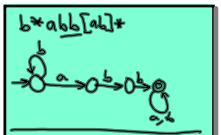
ab*:

ab+:



Jan 31-11:28 AM

b*abb[ab]*



For each rule:
exp_1 action$_1$
exp_2 action$_2$
exp_n action$_n$

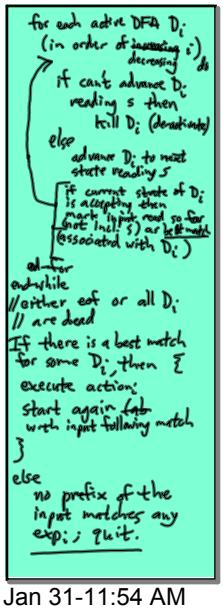
For each exp_i flex builds a DFA D_i that accepts strings matched by exp_i (and no others).

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

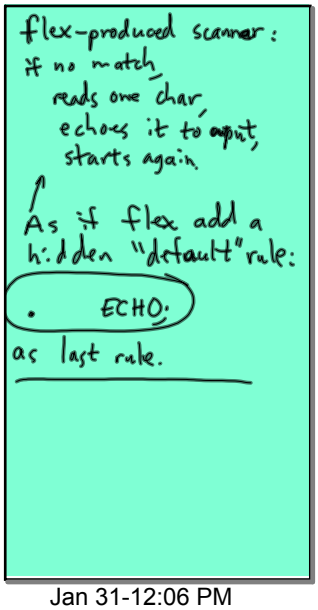
initially: nothing read on stream
all D_i in start states
all D_i are active (alive)
(no best matching string yet)
while (not eof and there are still active DFAs) do
  read next char s

Jan 31-11:45 AM

Jan 31-11:54 AM



Jan 31-11:54 AM



Jan 31-12:06 PM