2.4. NONDETERMINISTIC FINITE AUTOMATA

Algorithm. Abstractly, there is an edge from $d_i$ to $d_j$ labeled with $c$ if $d_j = \text{DFAedge}(d_i, c)$. We let $\Sigma$ be the alphabet.

```
states[0] ← {}; states[1] ← closure({s1})
p ← 1; j ← 0
while j ≤ p
    foreach c ∈ $\Sigma$
        e ← DFAedge(states[j], c)
        if e = states[i] for some i ≤ p
            then trans[j, c] ← i
        else p ← p + 1
            states[p] ← e
            trans[j, c] ← p
    j ← j + 1
```

The algorithm does not visit unreachable states of the DFA. This is extremely important, because in principle the DFA has $2^n$ states, but in practice we usually find that only about $n$ of them are reachable from the start state. It is important to avoid an exponential blowup in the size of the DFA interpreter’s transition tables, which will form part of the working compiler.

A state $d$ is final in the DFA if any NFA state in states[$d$] is final in the NFA. Labeling a state final is not enough; we must also say what token is recognized; and perhaps several members of states[$d$] are final in the NFA. In this case we label $d$ with the token-type that occurred first in the list of