Lexical Analysis:
DFA Minimization

Automating Scanner Construction

PREVIOUSLY

RE→NFA *(Thompson's construction)*

- Build an NFA for each term
- Combine them with ε-moves

NFA →DFA (*subset construction*)

- Build the simulation

TODAY

DFA →Minimal DFA

- Hopcroft's algorithm

DFA Minimization

Details of the algorithm

- Group states into maximal size sets, *optimistically*
- Iteratively subdivide those sets, as needed
- States that remain grouped together are equivalent

DFA Minimization

Remember DFA =(Q,Σ,δ,q_0,F)

Initial partition, P_0 , has two sets: *{D_F}* and *{D-D_F}*

Splitting a set s ("partitioning a set by a")

- Assume q_i, & $q_j \in s$ and $\delta(q_i,\underline{a}) = q_x$ and $\delta(q_j,\underline{a}) = q_y$
- If q_x and q_y are not in the same set, then *s* must be split
  - → q_i has transition on a, q_j does not ⇒ a splits *s*
- One state in the final DFA cannot have two transitions on a (otherwise we have an NFA!)

DFA Minimization (the algorithm)

$P \leftarrow \{ D_F, \{D-D_F\}\}$

while (P is still changing)

$\quad T \leftarrow \varnothing$

$\quad$ *for each set* $p \in P$

$\qquad T \leftarrow T \cup Split(p)$

$\quad P \leftarrow T$

Split(S)

for each $\alpha \in \Sigma$

if α *splits S into* s_1 *and* s_2

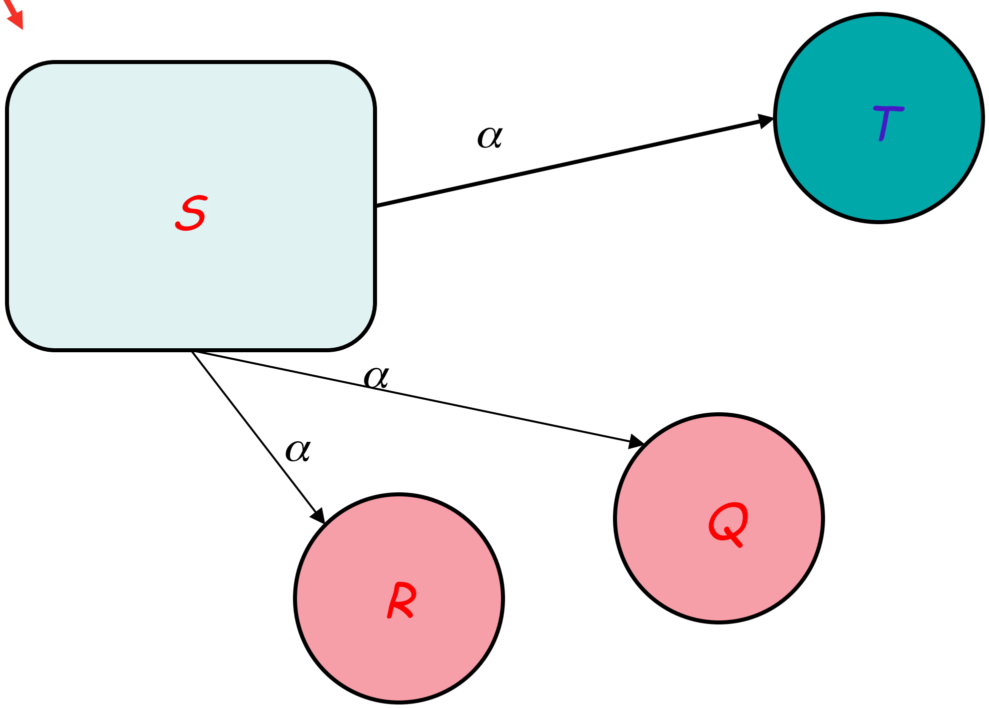
then return $\{s_1,s_2\}$

return S

This is a another fixed-point algorithm!

Key Idea: Splitting S around α

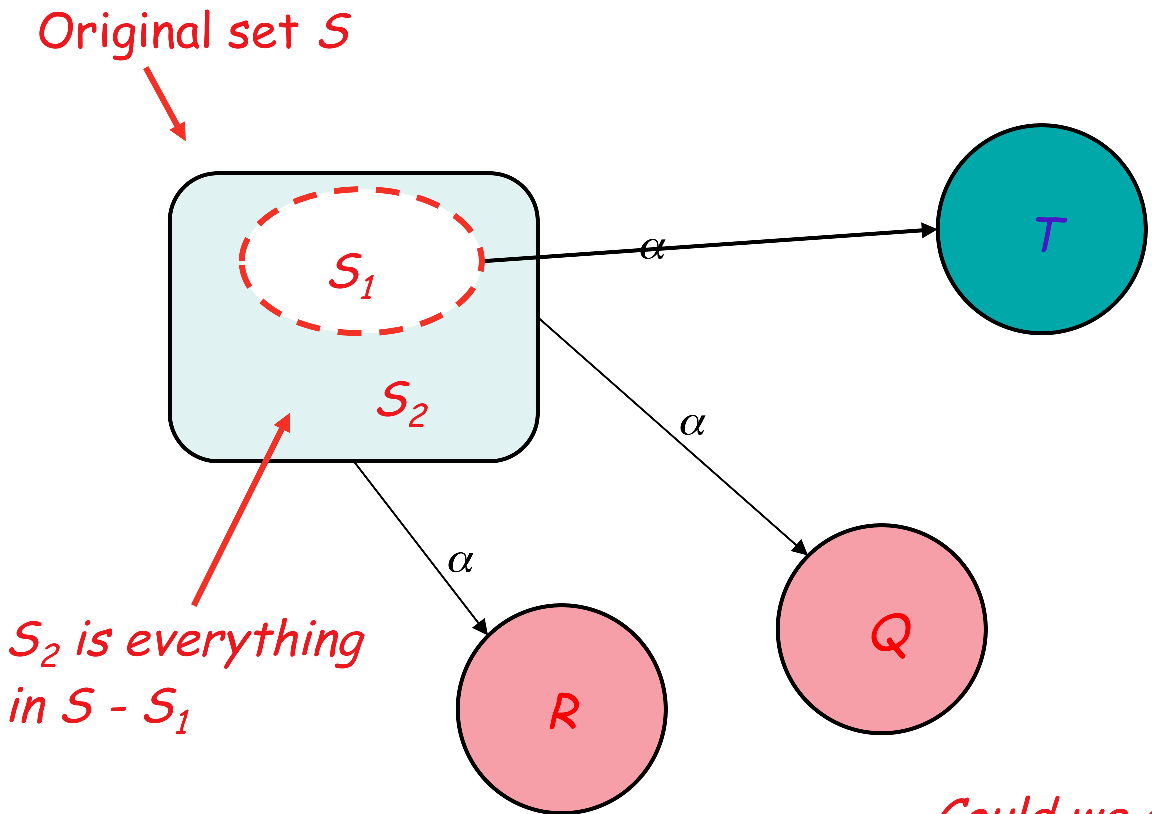
Original set S



S has transitions on α to R, Q, & T

The algorithm partitions S around α

Key Idea: Splitting S around α

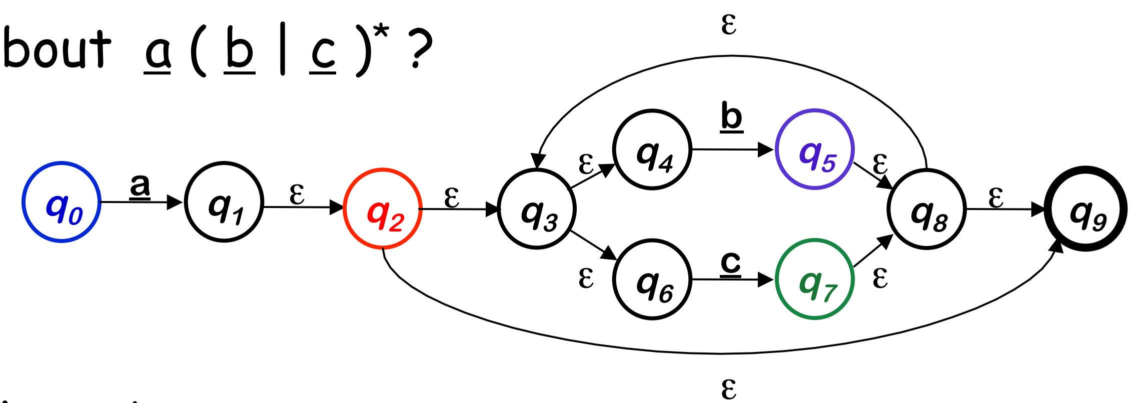


Could we split S_2 further?

Yes, will do this in another iteration!

DFA Minimization

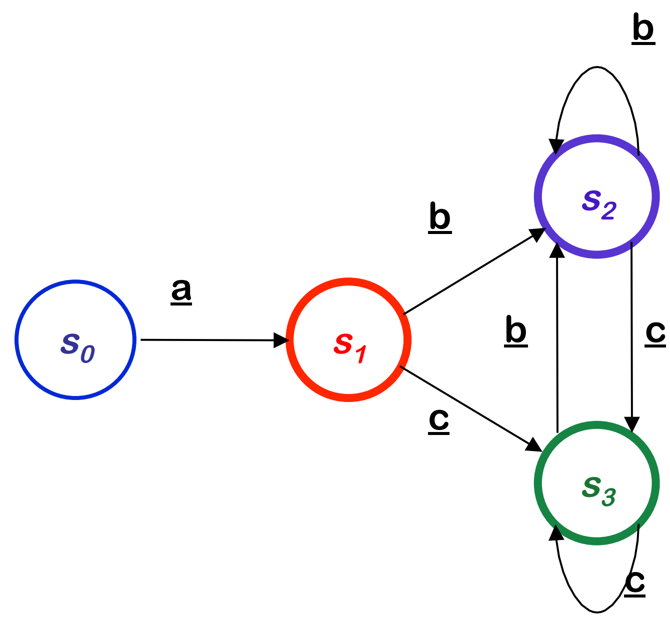
What about $\underline{a}\ (\ \underline{b}\ |\ \underline{c}\ )^*$?



First, the subset construction:

		ε-closure(Delta(s,*))		
	NFA states	a	b	c
s_0	q_0	$q_1, q_2, q_3, q_4, q_6, q_9$	none	none
s_1	$q_1, q_2, q_3, q_4, q_6, q_9$	none	$q_5, q_8, q_9, q_3, q_4, q_6$	$q_7, q_8, q_9, q_3, q_4, q_6$
s_2	$q_5, q_8, q_9, q_3, q_4, q_6$	none	s_2	s_3
s_3	$q_7, q_8, q_9, q_3, q_4, q_6$	none	s_2	s_3

Final states



Apply DFA Minimization algorithm

$P \leftarrow \{\, D_F, \{D-D_F\}\}$

while (P is still changing)

$\quad T \leftarrow \emptyset$

$\quad$ *for each set* $p \in P$

$\qquad T \leftarrow T \cup Split(p)$

$\quad P \leftarrow T$

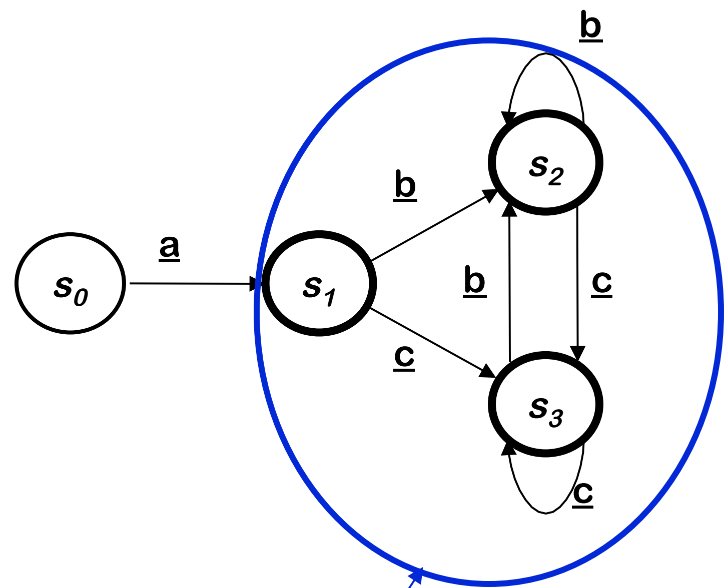
Split(S)

$\quad$ *for each* $\alpha \in \Sigma$

$\qquad$ *if* α *splits S into* s_1 *and* s_2

$\qquad\quad$ *then return* $\{s_1, s_2\}$

$\quad$ *return S*



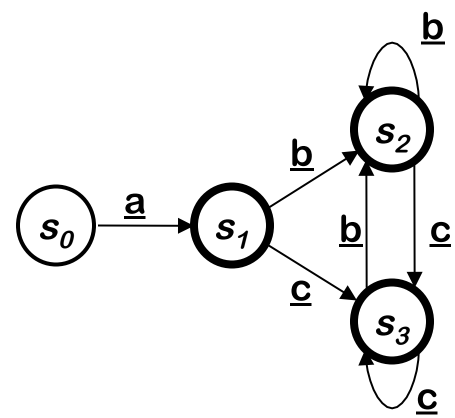
final states

DFA Minimization

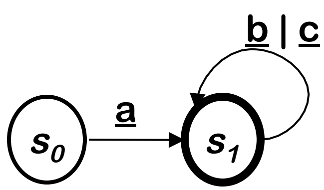
Then, apply the minimization algorithm

	Current Partition	Split on		
		a	b	c
P_0	$\{s_1, s_2, s_3\}$ $\{s_0\}$	*none*	*none*	*none*

final states



To produce the minimal DFA



In a previous lecture, we observed that a human would design a simpler automaton than Thompson's construction & the subset construction did.

Minimizing that DFA produces the one that a human would design!

Abbreviated Register Specification

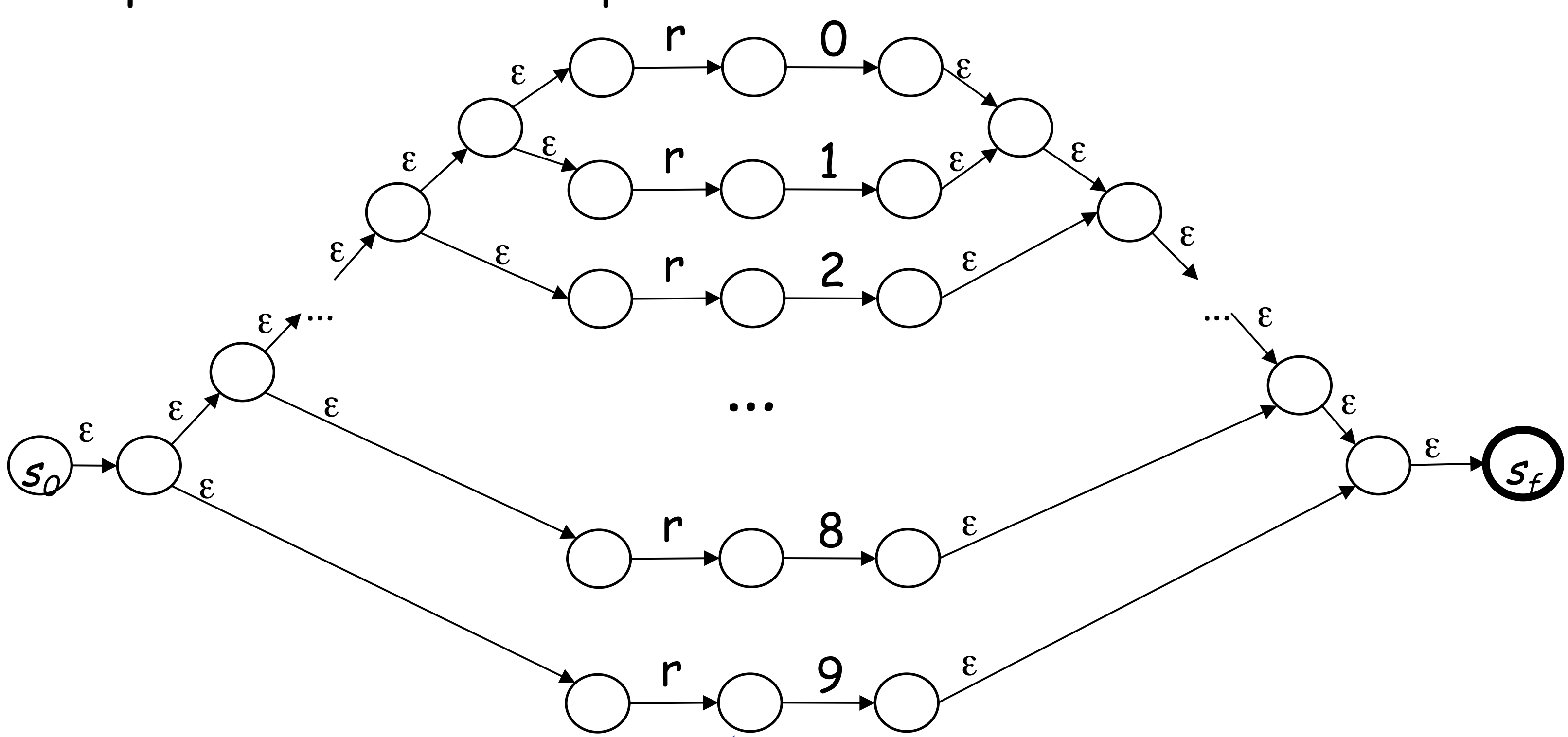
Start with a regular expression

r0 | r1 | r2 | r3 | r4 | r5 | r6 | r7 | r8 | r9

The Cycle of Constructions

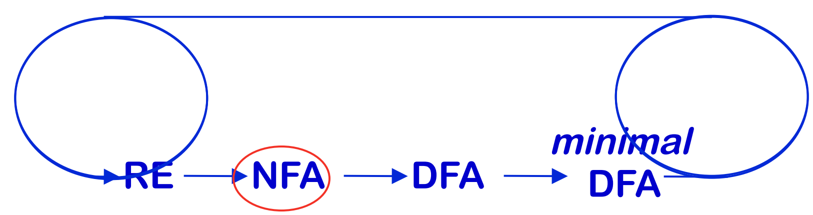
Abbreviated Register Specification

Thompson's construction produces



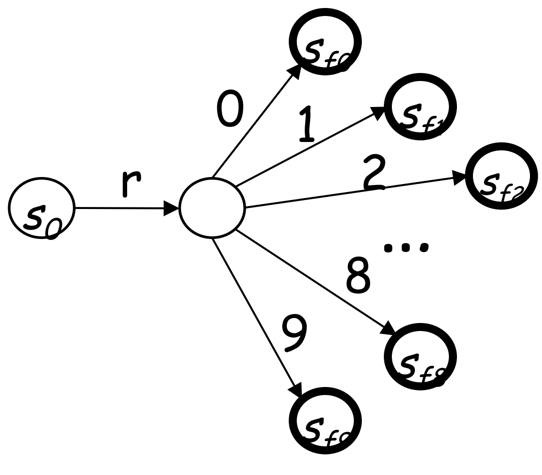
To make it fit, we've eliminated the ε-transition between "r" and "0...9".

The Cycle of Constructions

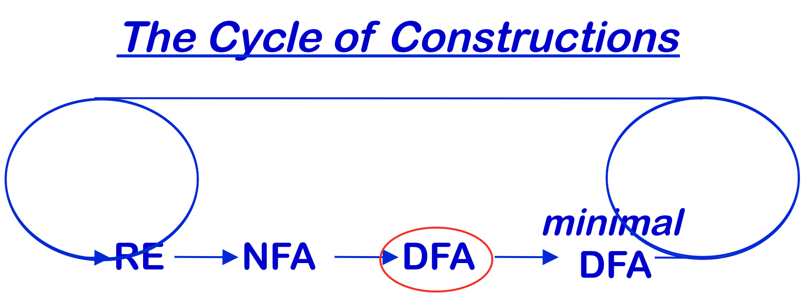


Abbreviated Register Specification

The subset construction builds

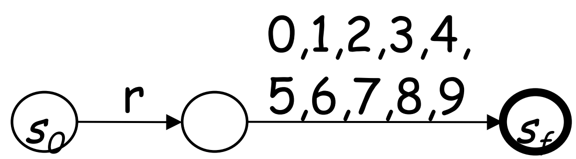


This is a DFA, but it has a lot of states …



Abbreviated Register Specification

The DFA minimization algorithm builds



This looks like what a skilled compiler writer would do!

The Cycle of Constructions