EC3204: Programming Languages and Compilers

Lecture 4 — Lexical Analysis (3)

Construction of String Recognizers

Sunbeom So
Fall 2024

This Lecture: Construction of DFA

Methodology: transform a lexical specification (regular expression) into an equivalent string recognizer (DFA).

- RE to NFA: Thompson's construction
- NFA to DFA: subset construction

cf) The transformations are instances of compilation. Their correctness is defined by the semantic equivalence:

- $L(RE) = L(NFA)$ for Thomson's construction
- $L(NFA) = L(DFA)$ for subset construction

Thompson's construction: RE to NFA

Recall RE from Lec. 2:

$$\begin{array}{rcll} \boldsymbol{R} & \rightarrow & \emptyset \mid \epsilon \mid a \in \Sigma & \text{(base cases)} \\ & \mid & R_1 \mid R_2 \mid R_1 \cdot R_2 \mid R_1^* \mid (R) & \text{(inductive cases)} \end{array}$$

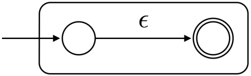
Method: use two kinds of transformation rules

- **Basic rules** for transforming primitive regexs into NFA
- **Inductive rules** for constructing larger NFA from sub-regexs' NFA

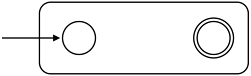
A final NFA will have exactly one start and one accepting state.

Basic Rules

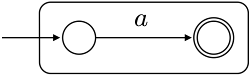
- $R = \epsilon$



- $R = \emptyset$



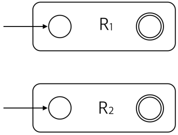
- $R = a \ (\in \Sigma)$



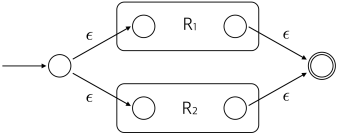
Clearly, $L(NFA) = L(R)$ in every case.

Inductive Rules

- $R = R_1 | R_2$:
  1. Compile R_1 and R_2:



  2. Construct $R_1 | R_2$ using the intermediate results:

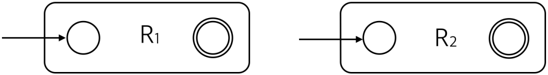


$L(NFA) = L(R_1) \cup L(R_2)$

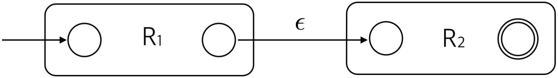
- Any path from the start to the final must path through either NFA_{R_1} or NFA_{R_2}, which accept $L(R_1)$ and $L(R_2)$, respectively.
- Strings (labels) are not changed by ϵ-transitions.

Inductive Rules

- $R = R_1 \cdot R_2$:
  1. Compile R_1 and R_2:



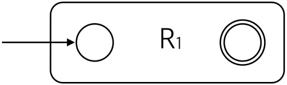
  2. Construct $R_1 \cdot R_2$ using the intermediate results:



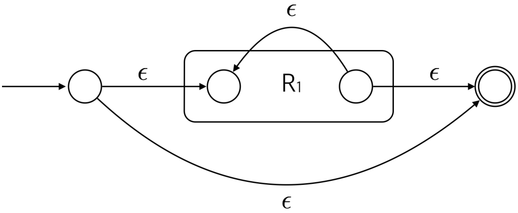
$$\begin{aligned} L(NFA) &= \{x\epsilon y \mid x \in L(R_1) \land y \in L(R_2)\} \\ &= \{xy \mid x \in L(R_1) \land y \in L(R_2)\} = L(R_1)L(R_2) \end{aligned}$$

Compilation

- $R = R_1^*$:
  1. Compile R_1:



  2. Construct R_1^* using the intermediate results:



$$L(NFA) = \{\epsilon\} \cup (L(R_1))^+ = (L(R_1))^0 \cup (L(R_1))^+ = (L(R_1))^*$$

Exercises

Construct NFAs that accept the languages described by the following regular expressions.

- $\mathbf{0 \cdot 1^*}$
- $\mathbf{(0|1) \cdot 0 \cdot 1}$
- $\mathbf{(0|1)^* \cdot 1 \cdot (0|1)}$

Our Context

- RE to NFA: Thompson's construction
- NFA to DFA: subset construction

NFA to DFA

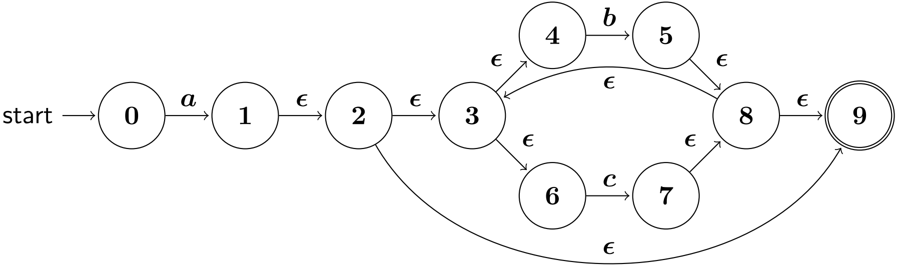
Transform an NFA

$$(N, \Sigma, \delta_N, n_0, N_A)$$

into an equivalent DFA

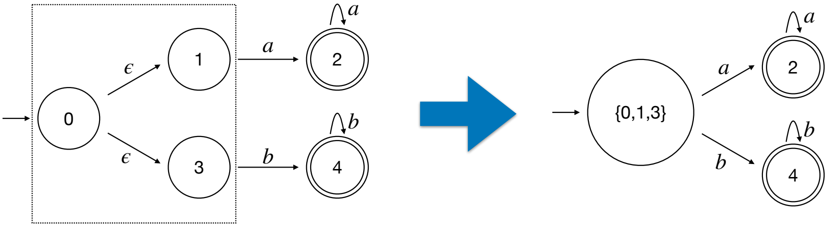
$$(D, \Sigma, \delta_D, d_0, D_A).$$

Running example ($a \cdot (b|c)^*$):



Subset Construction

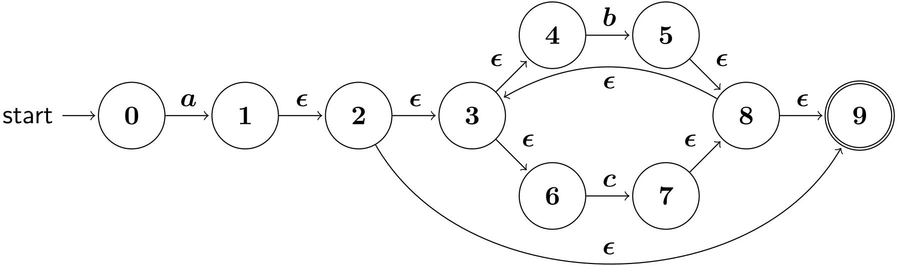
- Input: an NFA $(N, \Sigma, \delta_N, n_0, N_A)$.
- Output: a DFA $(D, \Sigma, \delta_D, d_0, D_A)$.
- Key Idea: eliminate non-deterministic choices in NFA.
  - How? By merging states whose internal labels do not change strings.



Internal labels can be ignored!

Preliminary: ϵ-Closure

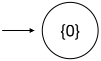
- ϵ-**closure**(I): the set of states reachable from I without consuming any symbols.



$$\begin{aligned} \epsilon\text{-}\mathbf{closure}(\{1\}) &= \{1,2,3,4,6,9\} \\ \epsilon\text{-}\mathbf{closure}(\{1,5\}) &= \{1,2,3,4,6,9\} \cup \{3,4,5,6,8,9\} \end{aligned}$$

Running Example (1/5)

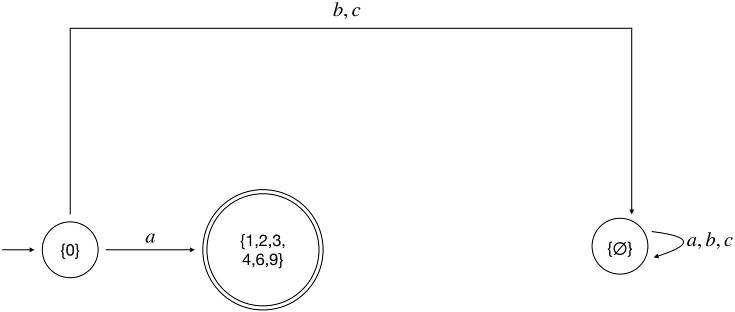
The initial DFA state $\boldsymbol{d_0} = \epsilon$-**closure**$(\{\mathbf{0}\}) = \{\mathbf{0}\}$.



Running Example (2/5)

For the initial state $d_0 = \{0\}$, consider every $x \in \Sigma$ and compute the corresponding next states:

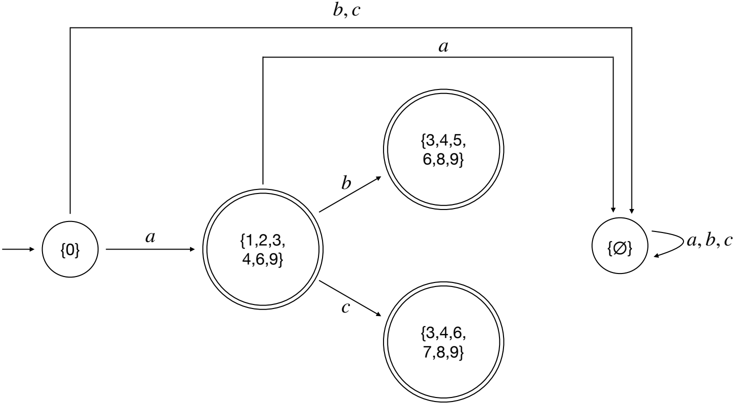
$$\begin{array}{rcl} \epsilon\text{-}\mathbf{closure}(\bigcup_{s \in \{0\}} \delta(s, a)) & = & \{1, 2, 3, 4, 6, 9\} \\ \epsilon\text{-}\mathbf{closure}(\bigcup_{s \in \{0\}} \delta(s, b)) & = & \emptyset \\ \epsilon\text{-}\mathbf{closure}(\bigcup_{s \in \{0\}} \delta(s, c)) & = & \emptyset \end{array}$$



Running Example (3/5)

For the state $\{1, 2, 3, 4, 6, 9\}$, compute the next states:

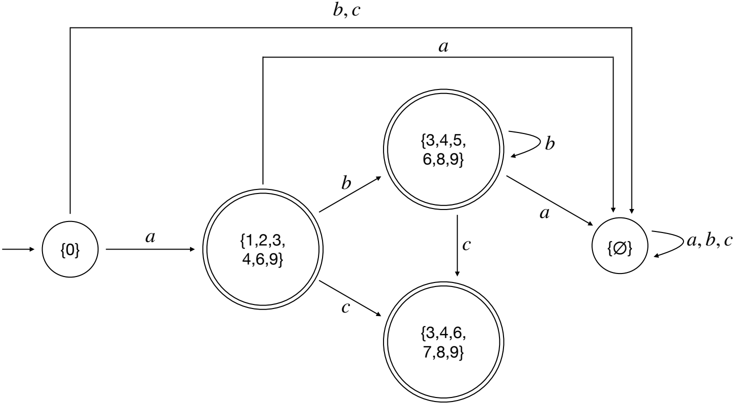
$$\epsilon\text{-closure}(\bigcup_{s\in\{1,2,3,4,6,9\}} \delta(s, a)) = \emptyset$$
$$\epsilon\text{-closure}(\bigcup_{s\in\{1,2,3,4,6,9\}} \delta(s, b)) = \{3, 4, 5, 6, 8, 9\}$$
$$\epsilon\text{-closure}(\bigcup_{s\in\{1,2,3,4,6,9\}} \delta(s, c)) = \{3, 4, 6, 7, 8, 9\}$$



Running Example (4/5)

Compute the next states of $\mathbf{\{3, 4, 5, 6, 8, 9\}}$:

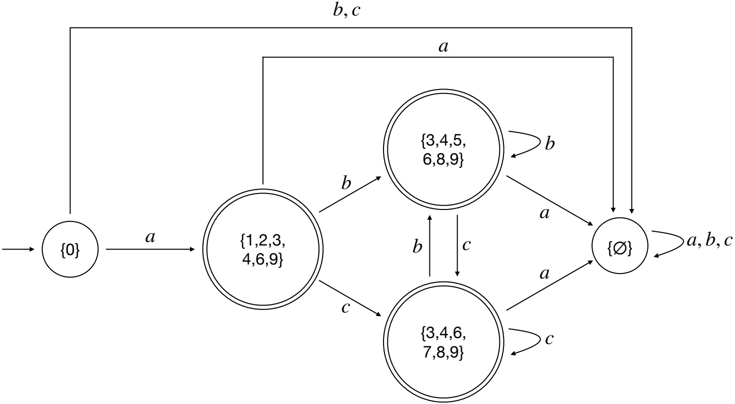
$$\begin{aligned}
&\epsilon\text{-}\mathbf{closure}(\textstyle\bigcup_{s\in\{3,4,5,6,8,9\}} \delta(s,a)) = \emptyset\\
&\epsilon\text{-}\mathbf{closure}(\textstyle\bigcup_{s\in\{3,4,5,6,8,9\}} \delta(s,b)) = \{3,4,5,6,8,9\}\\
&\epsilon\text{-}\mathbf{closure}(\textstyle\bigcup_{s\in\{3,4,5,6,8,9\}} \delta(s,c)) = \{3,4,6,7,8,9\}
\end{aligned}$$



Running Example (5/5)

Compute the next states of $\{3, 4, 6, 7, 8, 9\}$:

$$\epsilon\text{-}\mathbf{closure}(\bigcup_{s \in \{3,4,6,7,8,9\}} \delta(s, a)) = \emptyset$$
$$\epsilon\text{-}\mathbf{closure}(\bigcup_{s \in \{3,4,6,7,8,9\}} \delta(s, b)) = \{3, 4, 5, 6, 8, 9\}$$
$$\epsilon\text{-}\mathbf{closure}(\bigcup_{s \in \{3,4,6,7,8,9\}} \delta(s, c)) = \{3, 4, 6, 7, 8, 9\}$$



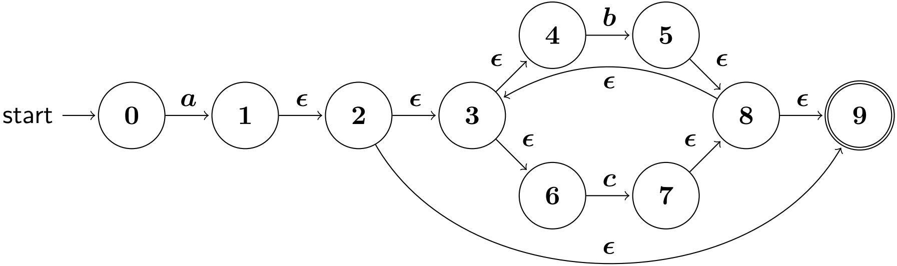
Subset Construction Algorithm

Algorithm 1 Subset Construction

```
Input: An NFA $(N, \Sigma, \delta_N, n_0, N_A)$
Output: An equivalent DFA $(D, \Sigma, \delta_D, d_0, D_A)$
 1: $d_0 \leftarrow \epsilon$-closure$(\{n_0\})$
 2: $D \leftarrow \{d_0\}$                                  ▷ $D$: a set of DFA states
 3: $W \leftarrow \{d_0\}$                ▷ $W$ (workset): a set of DFA states to process
 4: while $W \neq \emptyset$ do
 5:     pick and remove $q$ from $W$
 6:     for $x \in \Sigma$ do                          ▷ consider each input symbol
 7:         $t \leftarrow \epsilon$-closure$(\bigcup_{s \in q} \delta_N(s, x))$
 8:         $D \leftarrow D \cup \{t\}$
 9:         $\delta_D(q, x) \leftarrow t$
10:         if $t$ has not been added to $W$ before then
11:             $W \leftarrow W \cup \{t\}$
12: $D_A \leftarrow \{q \mid q \in D, q \cap N_A \neq \emptyset\}$
13: return $(D, \Sigma, \delta_D, d_0, D_A)$
```

- Note (small optimization): At line 10, if $t = \emptyset$, we can skip line 11, and instead update $\delta_D(\emptyset, x)$ as $\emptyset$ for all $x \in \Sigma$.

Running Example (1/5)



The initial state $d_0 = \epsilon$-**closure**$(\{0\}) = \{0\}$. Initialize D and W:

$$D = \{\{0\}\}, \qquad W = \{\{0\}\}$$

Running Example (2/5)

Choose $q = \{0\}$ from W.

- When $x = a$:
  - ϵ-**closure**$(\bigcup_{s\in\{0\}} \delta_N(s, a)) = \{1, 2, 3, 4, 6, 9\}$
  - $D = \{\{0\}, \{1, 2, 3, 4, 6, 9\}\}$
  - $W = \{\{1, 2, 3, 4, 6, 9\}\}$
- When $x = b$:
  - ϵ-**closure**$(\bigcup_{s\in\{0\}} \delta_N(s, b)) = \emptyset$
  - $D = \{\emptyset, \{0\}, \{1, 2, 3, 4, 6, 9\}\}$
  - $W = \{\{1, 2, 3, 4, 6, 9\}\}$
- When $x = c$:
  - ϵ-**closure**$(\bigcup_{s\in\{0\}} \delta_N(s, c)) = \emptyset$
  - $D = \{\emptyset, \{0\}, \{1, 2, 3, 4, 6, 9\}\}$
  - $W = \{\{1, 2, 3, 4, 6, 9\}\}$

Running Example (3/5)

Choose $q = \{1, 2, 3, 4, 6, 9\}$ from W.

- When $x = a$:
  - ϵ-**closure**$(\bigcup_{s \in \{1,2,3,4,6,9\}} \delta_N(s, a)) = \emptyset$
  - $D = \{\emptyset, \{0\}, \{1, 2, 3, 4, 6, 9\}\}$
  - $W = \emptyset$
- When $x = b$:
  - ϵ-**closure**$(\bigcup_{s \in \{1,2,3,4,6,9\}} \delta_N(s, b)) = \{3, 4, 5, 6, 8, 9\}$
  - $D = \{\emptyset, \{0\}, \{1, 2, 3, 4, 6, 9\}, \{3, 4, 5, 6, 8, 9\}\}$
  - $W = \{\{3, 4, 5, 6, 8, 9\}\}$
- When $x = c$:
  - ϵ-**closure**$(\bigcup_{s \in \{1,2,3,4,6,9\}} \delta_N(s, c)) = \{3, 4, 6, 7, 8, 9\}$
  - $D = \{\emptyset, \{0\}, \{1, 2, 3, 4, 6, 9\}, \{3, 4, 5, 6, 8, 9\}, \{3, 4, 6, 7, 8, 9\}\}$
  - $W = \{\{3, 4, 5, 6, 8, 9\}, \{3, 4, 6, 7, 8, 9\}\}$

Running Example (4/5)

Choose $q = \{3, 4, 5, 6, 8, 9\}$ from W.

- When $x = a$:
  - ϵ-**closure**$(\bigcup_{s \in \{3,4,5,6,8,9\}} \delta_N(s, a)) = \emptyset$
  - $D = \{\emptyset, \{0\}, \{1, 2, 3, 4, 6, 9\}, \{3, 4, 5, 6, 8, 9\}, \{3, 4, 6, 7, 8, 9\}\}$
  - $W = \{\{3, 4, 6, 7, 8, 9\}\}$
- When $x = b$:
  - ϵ-**closure**$(\bigcup_{s \in \{3,4,5,6,8,9\}} \delta_N(s, b)) = \{3, 4, 5, 6, 8, 9\}$
  - $D = \{\emptyset, \{0\}, \{1, 2, 3, 4, 6, 9\}, \{3, 4, 5, 6, 8, 9\}, \{3, 4, 6, 7, 8, 9\}\}$
  - $W = \{\{3, 4, 6, 7, 8, 9\}\}$
- When $x = c$:
  - ϵ-**closure**$(\bigcup_{s \in \{3,4,5,6,8,9\}} \delta_N(s, c)) = \{3, 4, 6, 7, 8, 9\}$
  - $D = \{\emptyset, \{0\}, \{1, 2, 3, 4, 6, 9\}, \{3, 4, 5, 6, 8, 9\}, \{3, 4, 6, 7, 8, 9\}\}$
  - $W = \{\{3, 4, 6, 7, 8, 9\}\}$

Running Example (5/5)

Choose $q = \{3, 4, 6, 7, 8, 9\}$ from W.

- When $x = a$:
  - ϵ-**closure**$(\bigcup_{s \in \{3,4,6,7,8,9\}} \delta_N(s, a)) = \emptyset$
  - $D = \{\emptyset, \{0\}, \{1, 2, 3, 4, 6, 9\}, \{3, 4, 5, 6, 8, 9\}, \{3, 4, 6, 7, 8, 9\}\}$
  - $W = \emptyset$
- When $x = b$:
  - ϵ-**closure**$(\bigcup_{s \in \{3,4,6,7,8,9\}} \delta_N(s, b)) = \{3, 4, 5, 6, 8, 9\}$
  - $D = \{\emptyset, \{0\}, \{1, 2, 3, 4, 6, 9\}, \{3, 4, 5, 6, 8, 9\}, \{3, 4, 6, 7, 8, 9\}\}$
  - $W = \emptyset$
- When $x = c$:
  - ϵ-**closure**$(\bigcup_{s \in \{3,4,6,7,8,9\}} \delta_N(s, c)) = \{3, 4, 6, 7, 8, 9\}$
  - $D = \{\emptyset, \{0\}, \{1, 2, 3, 4, 6, 9\}, \{3, 4, 5, 6, 8, 9\}, \{3, 4, 6, 7, 8, 9\}\}$
  - $W = \emptyset$

Running Example: Termination

The while-loop terminates:

$$\boldsymbol{D = \{\emptyset, \{0\}, \{1,2,3,4,6,9\}, \{3,4,5,6,8,9\}, \{3,4,6,7,8,9\}\}}$$

Since $\boldsymbol{N_A = \{9\}}$, the accepting states of DFA is:

$$\boldsymbol{D_A = \{\{1,2,3,4,6,9\}, \{3,4,5,6,8,9\}, \{3,4,6,7,8,9\}\}}$$

The final transition table can be obtained by incorporating $\boldsymbol{\delta_D}$ computed so far:

	$\boldsymbol{a}$	$\boldsymbol{b}$	$\boldsymbol{c}$
$\boldsymbol{\{0\}}$	$\boldsymbol{\{1,2,3,4,6,9\}}$	$\emptyset$	$\emptyset$
$\boldsymbol{\{1,2,3,4,6,9\}}$	$\emptyset$	$\boldsymbol{\{3,4,5,6,8,9\}}$	$\boldsymbol{\{3,4,6,7,8,9\}}$
$\boldsymbol{\{3,4,5,6,8,9\}}$	$\emptyset$	$\boldsymbol{\{3,4,5,6,8,9\}}$	$\boldsymbol{\{3,4,6,7,8,9\}}$
$\boldsymbol{\{3,4,6,7,8,9\}}$	$\emptyset$	$\boldsymbol{\{3,4,5,6,8,9\}}$	$\boldsymbol{\{3,4,6,7,8,9\}}$

Algorithm for Computing ϵ-Closures

- The definition "ϵ-**closure**(I) is the set of states reachable from I without consuming any symbols." is neither formal nor constructive.

Algorithm for Computing ϵ-Closures

- The definition "ϵ-**closure**(I) is the set of states reachable from I without consuming any symbols." is neither formal nor constructive.
- A formal definition:
  $T = \epsilon$-**closure**(I) is the smallest set such that

$$I \cup \bigcup_{s \in T} \delta(s, \epsilon) \subseteq T.$$

- Equivalently, T is the smallest solution X of the equation

$$F(X) \subseteq X$$

where

$$F(X) = I \cup \bigcup_{s \in X} \delta(s, \epsilon).$$

Such a solution is called **the least fixed point** of F, denoted $\mathit{fix} F$.

Why Smallest Solution?

- Recall ϵ-**closure**$(\{1\}) = \{1, 2, 3, 4, 6, 9\}$. Is this a unique solution that satisfies $F(X) \subseteq X$?

Why Smallest Solution?

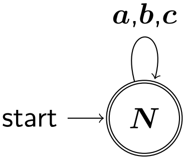
- Recall ϵ-**closure**$(\{1\}) = \{1, 2, 3, 4, 6, 9\}$. Is this a unique solution that satisfies $F(X) \subseteq X$?
- $X = N = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ is also the solution. So what happens if we use this solution that is not the least?

Why Smallest Solution?

- Recall $\boldsymbol{\epsilon}$-**closure**$(\{1\}) = \{1, 2, 3, 4, 6, 9\}$. Is this a unique solution that satisfies $F(X) \subseteq X$?
- $X = N = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ is also the solution. So what happens if we use this solution that is not the least?

Why Smallest Solution?

- Recall ϵ-**closure**$(\{1\}) = \{1, 2, 3, 4, 6, 9\}$. Is this a unique solution that satisfies $F(X) \subseteq X$?
- $X = N = \{0, 1, 2, 3, 4, 5, 6, 7, 8, 9\}$ is also the solution. So what happens if we use this solution that is not the least?
- We may accept invalid lexemes (e.g., c).



cf) In programming language theories, we are mainly interested in computing the least fixed point (typically indicates the most precise analysis results). We will revisit this concept later in this course (semantic analysis and optimization).

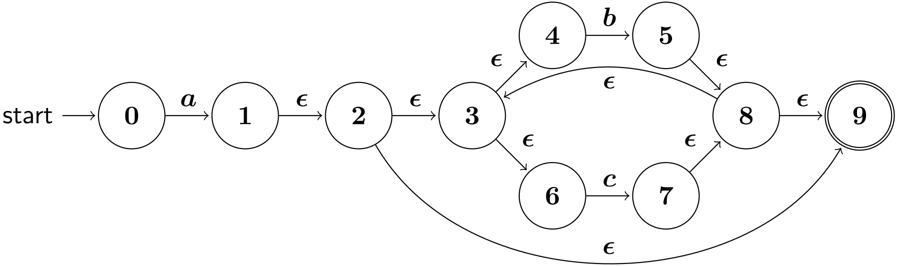
Fixed Point Iteration

The least fixed point of a function can be computed by the **fixed point iteration**.

```
$T := \emptyset$
repeat
    $T' := T$
    $T := T' \cup F(T')$
until $T = T'$
return $T$
```

In words: After initializing T with $\emptyset$, iteratievly apply F until T converges.

Example



ϵ-**closure**$(\{1\})$:

Iteration	T'	T
1	$\emptyset$	$\{1\}$
2	$\{1\}$	$\{1, 2\}$
3	$\{1, 2\}$	$\{1, 2, 3, 9\}$
4	$\{1, 2, 3, 9\}$	$\{1, 2, 3, 4, 6, 9\}$
5	$\{1, 2, 3, 4, 6, 9\}$	$\{1, 2, 3, 4, 6, 9\}$

Questions about Fixed Point Iteration

- Q. Why is the resulting $\boldsymbol{T}$ the least solution?

- Q. Why does the loop terminate? Why does $\boldsymbol{T}$ eventually converge?

Questions about Fixed Point Iteration

- Q. Why is the resulting T the least solution?
  A. T is obtained by initializing it with $\emptyset$, and then progressively expanding it.
- Q. Why does the loop terminate? Why does T eventually converge?
  A. The function F is monotonic

  $$F(A) \subseteq F(B) \text{ if } A \subseteq B$$

  and the solution has the upperbound N. Thus, at some point, F will not induce new information and T will not grow.

Summary

Construction of string recognizers (DFA)

- RE to NFA: Thompson's construction
- NFA to DFA: subset construction
  - Key idea: eliminate non-deterministic transitions in NFA.
  - More specifically, to make every transition unique, we simulate all possibilities at once for each input symbol, where all possibilities for each input symbol are computed using ϵ-**closure**.

Next class: functional programming in OCaml. Bring your laptop after installing OCaml!