COSE312: Compilers

Lecture 5 — Lexical Analysis (4)

Hakjoo Oh
2015 Fall

Part 3: Automation

Transform the lexical specification into an executable string recognizers:

$$\text{RE} \xrightarrow[\text{Thompson's construction}]{\hspace{3em}} \text{NFA} \xrightarrow[\text{subset construction}]{\hspace{3em}} \text{DFA}$$

From NFA to DFA

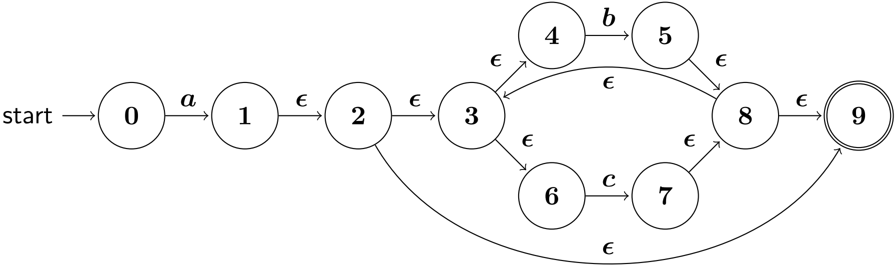
Transform an NFA

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

into an equivalent DFA

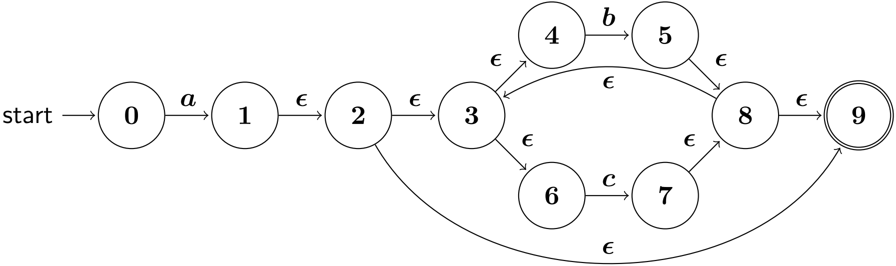
$$(D, \Sigma, \delta_D, d_0, D_A).$$

Running example:



ϵ-Closures

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



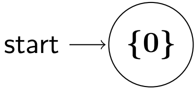
$$
\begin{aligned}
\epsilon\text{-closure}(\{1\}) &= \{1, 2, 3, 4, 6, 9\} \\
\epsilon\text{-closure}(\{1, 5\}) &= \{1, 2, 3, 4, 6, 9\} \cup \{3, 4, 5, 6, 8, 9\}
\end{aligned}
$$

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: the DFA simulates the NFA by considering every possibility at once. A DFA state $d \in D$ is a set of NFA state, i.e., $d \subseteq N$.

Running Example (1/5)

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

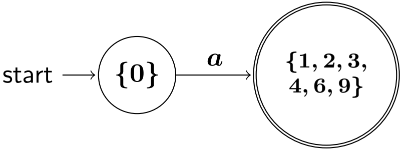


Running Example (2/5)

For the initial state S, consider every $x \in \Sigma$ and compute the corresponding next states:

$$\epsilon\text{-}\mathbf{closure}(\bigcup_{s \in S} \delta(s, a)).$$

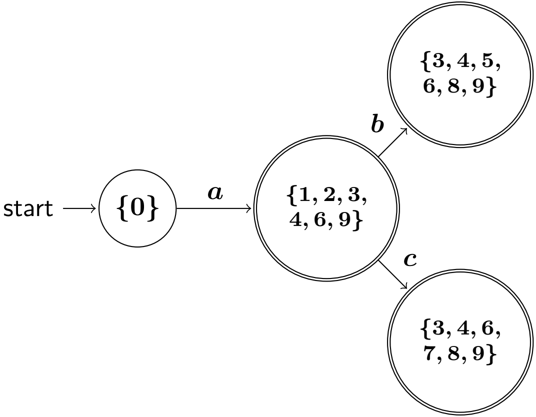
$$\begin{array}{rcl}
\epsilon\text{-}\mathbf{closure}(\bigcup_{s \in \{0\}} \delta(s, a)) & = & \{1, 2, 3, 4, 6, 9\} \\
\epsilon\text{-}\mathbf{closure}(\bigcup_{s \in \{0\}} \delta(s, b)) & = & \emptyset \\
\epsilon\text{-}\mathbf{closure}(\bigcup_{s \in \{0\}} \delta(s, c)) & = & \emptyset
\end{array}$$



Running Example (3/5)

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

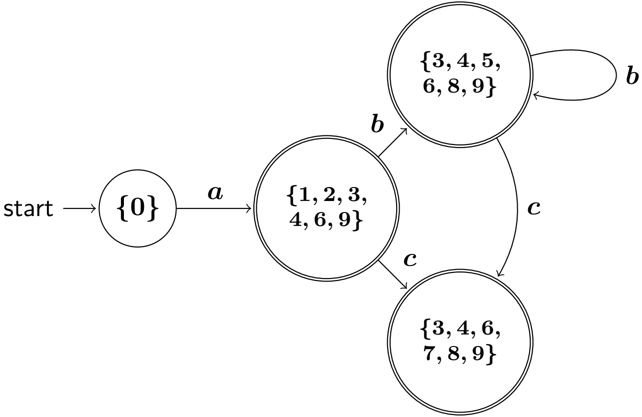
$$\epsilon\text{-}\textbf{closure}(\bigcup_{s\in\{1,2,3,4,6,9\}} \delta(s,a)) = \emptyset$$
$$\epsilon\text{-}\textbf{closure}(\bigcup_{s\in\{1,2,3,4,6,9\}} \delta(s,b)) = \{3,4,5,6,8,9\}$$
$$\epsilon\text{-}\textbf{closure}(\bigcup_{s\in\{1,2,3,4,6,9\}} \delta(s,c)) = \{3,4,6,7,8,9\}$$



Running Example (4/5)

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

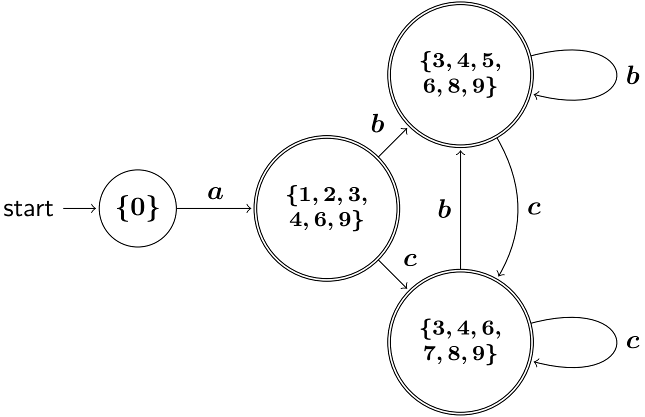
$$\epsilon\text{-}\mathbf{closure}(\bigcup_{s\in\{3,4,5,6,8,9\}} \delta(s,a)) = \emptyset$$
$$\epsilon\text{-}\mathbf{closure}(\bigcup_{s\in\{3,4,5,6,8,9\}} \delta(s,b)) = \{3,4,5,6,8,9\}$$
$$\epsilon\text{-}\mathbf{closure}(\bigcup_{s\in\{3,4,5,6,8,9\}} \delta(s,c)) = \{3,4,6,7,8,9\}$$



Running Example (5/5)

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

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



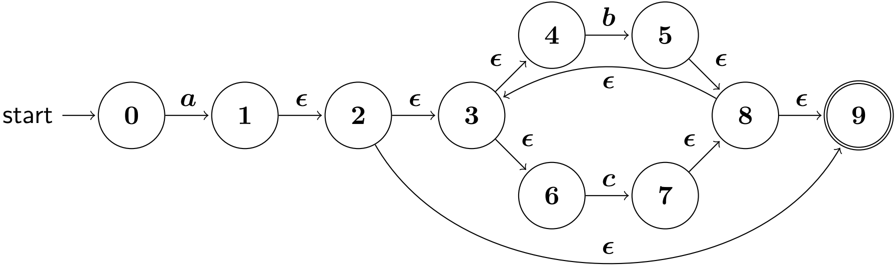
Subset Construction Algorithm

Algorithm 1 Subset construction

Input: An NFA $(N, \Sigma, \delta_N, n_0, N_A)$
Output: An equivalent DFA $(D, \Sigma, \delta_D, d_0, D_A)$

```
d0 = ε-closure({n0})
D = {d0}
W = {d0}
while W ≠ ∅ do
    remove q from W
    for c ∈ Σ do
        t = ε-closure(⋃_{s∈q} δ(s, c))
        δ_D(q, c) = t
        if t ∉ D then
            D = D ∪ {t}
            W = W ∪ {t}
        end if
    end for
end while
D_A = {q ∈ D | q ∩ N_A ≠ ∅}
```

Running Example (1/5)



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

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

Running Example (2/5)

Choose $q = \{0\}$ from W. For all $c \in \Sigma$, update δ_D:

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

Update D and W:

$$D = \{\{0\}, \{1, 2, 3, 4, 6, 9\}\}, \qquad W = \{\{1, 2, 3, 4, 6, 9\}\}$$

Running Example (3/5)

Choose $q = \{1, 2, 3, 4, 6, 9\}$ from W. For all $c \in \Sigma$, update δ_D:

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

Update D and W:

$$\begin{array}{rcl} D & = & \{\{0\}, \{1,2,3,4,6,9\}, \{3,4,5,6,8,9\}, \{3,4,6,7,8,9\}\} \\ W & = & \{\{3,4,5,6,8,9\}, \{3,4,6,7,8,9\}\} \end{array}$$

Running Example (4/5)

Choose $q = \{3,4,5,6,8,9\}$ from W. For all $c \in \Sigma$, update δ_D:

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

D and W:

$$\begin{array}{rcl} D & = & \{\{0\},\{1,2,3,4,6,9\},\{3,4,5,6,8,9\},\{3,4,6,7,8,9\}\} \\ W & = & \{\{3,4,6,7,8,9\}\} \end{array}$$

Running Example (5/5)

Choose $q = \{3,4,6,7,8,9\}$ from W. For all $c \in \Sigma$, update δ_D:

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

D and W:

$$\begin{array}{rcl} D & = & \{\{0\},\{1,2,3,4,6,9\},\{3,4,5,6,8,9\},\{3,4,6,7,8,9\}\} \\ W & = & \emptyset \end{array}$$

The while loop terminates. The accepting states:

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

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.

- To be formal and constructive,
  1. define ϵ-**closure**(I) by inductive definition,
  2. compute the set by *fixed point computation*.

Inductive Definition

Let I be a set of NFA states. The ϵ closure, $T = \epsilon\text{-}\mathbf{closure}(I)$, is the smallest set that satisfies the two conditions:

1. $I \subseteq T$.
2. If $S \subseteq T$, then $\bigcup_{s \in S} \delta(s, \epsilon) \subseteq T$.

or alternatively, $T = \epsilon\text{-}\mathbf{closure}(I)$ is the smallest set that satisfies the two conditions.

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

or alternatively, $T = \epsilon\text{-}\mathbf{closure}(I)$ is the smallest set such that

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

The inductively defined set can be computed by formulating the set by a *least fixed point* of a function F, and compute the least fixed point via *fixed point iteration*.

Least Fixed Point

- F: a function defined over sets: e.g.,
  - $F_1(X) = X \cup \{1, 2, 3\}$
  - $F_2(X) = X = \{1, 2\}$
- A set X is a *(pre-)fixed point* of F if

$$X \supseteq F(X).$$

- $\mathit{fix} F$: the *least fixed point* of F, i.e.,
  - $\mathit{fix} F \supseteq F(\mathit{fix} F)$
  - $X \supseteq F(X) \implies X \supseteq \mathit{fix} F$
- $\mathit{fix} F$ can be computed by the algorithm:

$$\begin{array}{l} T = \emptyset \\ \textbf{repeat} \\ \quad T' = T \\ \quad T = T' \cup F(T') \\ \textbf{until } T = T' \end{array}$$

Computing ϵ-Closures

To compute $T = \epsilon$-**closure**(I),

1. define a function F such that $T = \mathit{fix} F$, and
2. compute $\mathit{fix} F$ by fixed point iteration.

Computing ϵ-Closures

1. The inductive definition:

$$T = \epsilon\text{-}\mathbf{closure}(I) \text{ is the smallest set such that}$$

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

can be re-stated by:

$$T = \epsilon\text{-}\mathbf{closure}(I) \text{ is the smallest set such that}$$

$$T \supseteq F(T)$$

where

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

Thus, $T = \mathit{fix} F$.

Computing ϵ-Closures

2. Compute $\mathit{fix}F$ via fixed point iteration algorithm:

$$
\begin{array}{l}
T = \emptyset \\
\textbf{repeat} \\
\quad T' = T \\
\quad T = T' \cup F(T') \\
\textbf{until } T = T'
\end{array}
$$

ex) 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	$\{1,2,3,4,6,9\}$	$\{1,2,3,4,6,9\}$

cf) Computer Science is full of fixed points

Every inductively defined set is defined by fixed points.

- The set $N = \{0, 1, 2, 3, \ldots\}$ of natural numbers can be defined by a least fixed point

$$N = \mathit{fix} F.$$

  What is F?

- Let $G = (N, \rightarrow)$ be a graph, where N is the set of nodes and $(\rightarrow) \subseteq N \times N$ denotes edges. Let $I \subseteq N$ be a set of initial nodes. The set R_I of all nodes reachable from I can be defined by a least fixed point:

$$R_I = \mathit{fix} F$$

  What is F?

Efficient Fixed Point Computation via Worklist Algorithm

Recall the fixed point algorithm:

$$
\begin{array}{l}
\boldsymbol{T = \emptyset} \\
\textbf{repeat} \\
\quad \boldsymbol{T' = T} \\
\quad \boldsymbol{T = T' \cup F(T')} \\
\textbf{until } \boldsymbol{T = T'}
\end{array}
$$

and the computation of 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	$\{1, 2, 3, 4, 6, 9\}$	$\{1, 2, 3, 4, 6, 9\}$

Efficient Fixed Point Computation via Worklist Algorithm

The algorithm involves many redundant computations.

- The first iteration:

$$\boldsymbol{F(\emptyset) = \{1\} \cup \Big(\bigcup_{s \in \emptyset} \delta(s, \epsilon)\Big) = \{1\}}$$

- The second iteration:

$$\boldsymbol{F(\{1\}) = \{1\} \cup \Big(\bigcup_{s \in \{1\}} \delta(s, \epsilon)\Big) = \{1\} \cup \delta(1, \epsilon)}$$

- The third iteration:

$$\boldsymbol{F(\{1,2\}) = \{1\} \cup \Big(\bigcup_{s \in \{1,2\}} \delta(s, \epsilon)\Big) = \{1\} \cup \delta(1, \epsilon) \cup \delta(2, \epsilon)}$$

- The fourth iteration:

$$\boldsymbol{\begin{array}{rcl} F(\{1,2,3,9\}) & = & \{1\} \cup \big(\bigcup_{s \in \{1,2,3,9\}} \delta(s, \epsilon)\big) \\ & = & \{1\} \cup \delta(1, \epsilon) \cup \delta(2, \epsilon) \cup \delta(3, \epsilon) \cup \delta(9, \epsilon) \end{array}}$$

Efficient Fixed Point Computation via Worklist Algorithm

- The fifth iteration:

$$
\begin{aligned}
&F(\{1,2,3,4,6,9\}) \\
&= \{1\} \cup \left(\bigcup_{s\in\{1,2,3,4,6,9\}} \delta(s,\epsilon)\right) \\
&= \{1\} \cup \delta(1,\epsilon) \cup \delta(2,\epsilon) \cup \delta(3,\epsilon) \cup \delta(4,\epsilon) \cup \delta(6,\epsilon) \cup \delta(9,\epsilon)
\end{aligned}
$$

Efficient Fixed Point Computation via Worklist Algorithm

The worklist algorithm can compute fixed points with less redundancies:

Input: A set I of initial states.
Output: $T = \epsilon\text{-closure}(I)$

```
T = I
W = I
while W ≠ ∅ do
   remove a state q from W
   S = δ(q, ε)
   for s ∈ S do
      if s ∉ T then
         T = T ∪ {s}
         W = W ∪ {s}
      end if
   end for
end while
```

- T and W are initially $T = W = \{1\}$.
- Choose 1 and compute $\delta(1, \epsilon) = \{2\}$. Add 2 to T and W:

$$T = \{1, 2\}, \quad W = \{2\}$$

- Choose 2 and compute $\delta(2, \epsilon) = \{3, 9\}$. Add them to T and W:

$$T = \{1, 2, 3, 9\}, \quad W = \{3, 9\}$$

- Choose 3 and compute $\delta(3, \epsilon) = \{4, 6\}$. Add them to T and W:

$$T = \{1, 2, 3, 4, 6, 9\}, \quad W = \{4, 6, 9\}$$

- Choose 4 and compute $\delta(4, \epsilon) = \emptyset$. Nothing is added.

$$T = \{1, 2, 3, 4, 6, 9\}, \quad W = \{6, 9\}$$

- Choose 6 and compute $\delta(6, \epsilon) = \emptyset$. Nothing is added.

$$T = \{1, 2, 3, 4, 6, 9\}, \quad W = \{9\}$$

- Choose 9 and compute $\delta(9, \epsilon) = \emptyset$. Nothing is added.

$$T = \{1, 2, 3, 4, 6, 9\}, \quad W = \{\}$$

The worklist is empty and the algorithm terminates.

Summary

Subset construction:

- Goal: convert an NFA to an equivalent DFA
- Key idea: simulate the NFA by considering every possibility at once

ϵ-closures:

- defined by least fixed points
- computed by fixed point algorithms
  - naitve iterative algorithm
  - worklist algorithm