

2.1.3 ε-move를 허용한다

(해결책) ε-closure(ε*)를 구한다.

오토마타를 더 확장해보자. ε-move를 허용한다. 즉 기존에 NFA에서 입력문자열을 보지 않고 상태를 바꿀 수 있게 한다.

이 상황을 상태변화함수 δ로 표시하면 아래와 같다.

$$\delta: Q \times (\Sigma \cup \{\epsilon\}) \rightarrow 2^Q.$$

즉 상태변화함수의 두 번째 도메인에 ε을 추가하여, δ(q, ε) = {p₁, p₂, ..., p_n}을 허용한다.

(정의 10) ε-move를 허용하는 오토마타 M_{ε-NFA} = (Q, Σ, δ_{ε-NFA}, q₀, F)는 아래와 같이 정의한다.

(1), (2), (4), (5)의 Q, Σ, q₀, F는 기본정의 M_{DFA}와 같다. 단

(3) δ_{ε-NFA}: Q × (Σ ∪ {ε}) → 2^Q 상태변화함수만이 다르다.

(정의 11) M_{ε-NFA}를 M_{ε-NFA} 전체의 오토마타 클래스라 부른다.

(쉬운정리 3) M_{DFA} ⊆ M_{부분} ⊆ M_{NFA} ⊆ M_{ε-NFA}.

M_{NFA} ⊆ M_{ε-NFA}이므로 쉽다.

(정의 10)와 (쉬운정리 3)으로 확장의 첫 번째와 두 번째 작업인 확장된 오토마타 클래스의 (A) 정의와 (B) 확장을 마치었다. 확장에 마지막 작업으로 같은 일을 하면서 상태변화함수 δ가 전체함수인 오토마타 클래스로 바꾸어보자.

(정의 12) 상태변화함수 δ의 첫 번째 정의역을 상태(Q)에서 상태집합(2^Q)으로 확장하자.

$$\begin{aligned} \delta: 2^Q \times \Sigma &\rightarrow 2^{2^Q} \\ \delta(P, a) &\stackrel{\text{def}}{=} \{q \in Q \mid p \in P, \delta(p, a) = q\} \\ &= \bigcup_{p \in P} \delta(p, a) \end{aligned}$$

(정의 13) ε-move를 반복적으로 수행하는 ε*를 recursive하게 정의하자.

$$\begin{aligned} \epsilon^* : 2^Q &\rightarrow 2^Q \\ \epsilon^*(P) &\stackrel{\text{def}}{=}_{\text{B}} P \quad \text{단 } P \in 2^Q. \\ \epsilon^*(P) &\stackrel{\text{def}}{=}_{\text{R}} \epsilon^*(\delta_{\epsilon-NFA}(P, \epsilon)) \quad \text{단 } P \in 2^Q. \end{aligned}$$

즉 ε*는 ε-NFA에서 추가로 정의된 δ(q, ε)을 여러 번 하는 것이다.

임의의 M_{ε-NFA}를 같은 일을 하는 M_{DFA}로 바꾸어주는 알고리즘을 보자.

(알고리즘) **function** ε-NFA_to_DFA(Q_{ε-NFA}(ε-NFA상태집합), Σ, δ_{ε-NFA}(ε-NFA상태변화함수), q₀ ∈ Q_{ε-NFA}, F_{ε-NFA} ⊆ Q_{ε-NFA}) **returns** (Q_{DFA}(DFA상태집합), Σ, δ_{DFA}(DFA상태변화함수), q₀^{DFA} ∈ Q_{DFA}, F_{DFA} ⊆ Q_{DFA});
variable P ⊆ Q_{ε-NFA}; a ∈ 입력문자집합(Σ);

1) 앞 장의 (정의 8)과 같으나, δ' 대신 δ를 함께 쓰고 있음에 주의하자.

$$q_0^{DFA} = \epsilon^*({q_0}); q_0^{DFA} \subseteq Q_{DFA}^2);$$

repeat

for $P \in Q_{DFA}$ **do**

for $a \in \Sigma$ **do**

$$\epsilon^*(\delta_{\epsilon-NFA}(P, a)) \subseteq Q_{DFA}; (\delta_{DFA}(P, a) = \epsilon^*(\delta_{\epsilon-NFA}(P, a))) \subseteq \delta_{DFA};^3)$$

od

if $(P \cap F_{\epsilon-NFA}) \neq \emptyset \rightarrow P \subseteq F_{DFA}$ **else skip fi**

od

until no more new states are added to Q_{DFA}

return $M_{DFA} = (Q_{DFA}, \Sigma, \delta_{DFA}, q_0^{DFA}, F_{DFA});$

end function NFA_to_DFA;

(부분정리) $Q_{DFA} = 2^{Q_{\epsilon-NFA}} = \{P \mid P \subseteq Q_{\epsilon-NFA}\} = \{P \mid P \subseteq Q_{DFA}\}$

$$P \subseteq Q_{\epsilon-NFA} (\equiv P \subseteq 2^{Q_{\epsilon-NFA}} \equiv P \subseteq Q_{DFA}), a \in \Sigma.$$

$$\delta_{DFA}(P, a) = \epsilon^*(\delta_{\epsilon-NFA}(P, a)).$$

$$F_{DFA} = \{P \in 2^{Q_{DFA}} \mid P \cap F \neq \emptyset\}$$

(증명1) 위의 (알고리즘)은 임의의 ε-NFA에서 위의 (부분정리)을 만족하는 DFA를 만든다.

(증명2) $L(M_{\epsilon-NFA}) = L(M_{DFA})$

(생략)

(중요정리 3) \mathbb{M}_{DFA} 와 $\mathbb{M}_{부분}$, \mathbb{M}_{NFA} , $\mathbb{M}_{\epsilon-NFA}$ 는 모두 같은 일을 하는(같은) 오토마타 클래스이다.

2.1.4 입력문자열(Σ^*)에 대한 상태변환을 허용한다

(해결책) 중간에 입력문자 하나씩 보고 상태를 바꾸는 중간상태들을 추가한다.

오토마타를 더 확장해보자. ε-move 뿐만이 아니라 길이 2 이상의 입력문자열에 의한 상태 변화도 허용한다. 즉 기존에 ε-NFA에서 입력문자 여러 개를 보고 한꺼번에 상태를 바꿀 수 있게 한다.

이 상황을 상태변화함수 δ로 표시하면 아래와 같다.

$$\delta: Q \times \Sigma^* \rightarrow 2^Q.$$

즉 상태변화함수의 두 번째 도메인을 Σ^* 로 확장하여, $\delta(q, a_1 a_2 \dots a_k) = \{p_1, p_2, \dots, p_n\}$ 을 허용한다.

2) 이러한 표현은 declarative 언어에서 쓰는 표현으로 앞 장의 procedural 언어의 $Q_{DFA} := Q_{DFA} \cup \{q_0^{DFA}\}$ 와 같다. 또한 엄밀히 쓰면 $\{q_0^{DFA}\} \subseteq Q_{DFA}$ 로 써야하나, 집합을 위한 {}기호를 빼고 $q_0^{DFA} \subseteq Q_{DFA}$ 로 쓰기도 한다.

3) $Q = \epsilon^*(\delta_{\epsilon-NFA}(P, a));$ 라 하고, $Q \subseteq Q_{DFA}; (\delta_{DFA}(P, a) = Q) \subseteq \delta_{DFA}$ 와 같다.

(정의 12) 확장된 오토마타 $M_{XFA} = (Q, \Sigma, \delta_{XFA}, q_0, F)$ 는 아래와 같이 정의한다.

(1), (2), (4), (5)의 Q, Σ, q_0, F 는 기본정의 M_{DFA} 와 같다. 단

(3) $\delta_{XFA}: Q \times \Sigma^* \rightarrow 2^Q$ 상태변화함수만이 다르다.

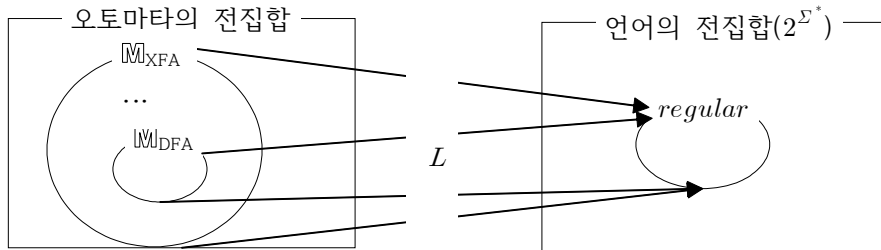
(정의 13) \mathbb{M}_{XFA} 를 M_{XFA} 전체의 오토마타 클래스라 부른다.

(쉬운정리 4) $\mathbb{M}_{DFA} \subsetneq \mathbb{M}_{부분} \subsetneq \mathbb{M}_{NFA} \subsetneq \mathbb{M}_{\epsilon-NFA} \subsetneq \mathbb{M}_{XFA}$.

$\mathbb{M}_{\epsilon-NFA} \subsetneq \mathbb{M}_{XFA}$ 이므로 쉽다.

(정의 12)와 (쉬운정리 4)로 확장의 첫 번째와 두 번째 작업인 확장된 오토마타 클래스의 (A) 정의와 (B) 확장을 마치었다. 확장에 세 번째 작업으로 같은 일을 하면서 상태변화함수 δ 가 입력문자 하나에 대하여 전체함수인 오토마타로(DFA) 바꾸는 작업은, (1) 길이가 0인 ϵ -move는 2.1.3의 알고리즘을 이용하고, (2) 길이 2 이상인 문자열(길이 n)에 관한 상태변환은 중간에 문자 하나만 보고 상태를 바꾸는 중간 상태 $(n-1)$ 개를 넣어줌으로 쉽게 해결되므로 생략한다.

(중요정리 4) \mathbb{M}_{DFA} 와 $\mathbb{M}_{부분}$, \mathbb{M}_{NFA} , $\mathbb{M}_{\epsilon-NFA}$, \mathbb{M}_{XFA} 는 모두 같은 일을 하는(같은) 오토마타 클래스이다.



확장된 오토마타 \mathbb{M}_{XFA} 를 줄여서 오토마타 \mathbb{M}_{FA} 라 부르기도 한다.

(마지막 중요정리) \mathbb{M}_{DFA} , $\mathbb{M}_{부분}$, \mathbb{M}_{NFA} , $\mathbb{M}_{\epsilon-NFA}$, \mathbb{M}_{FA} 는 모두 같은 언어 클래스(정규언어)를 정의하는 오토마타 클래스이고 서로 바꿀 수 있으므로, 구분 없이 사용한다.

| | |
|-----------------------------|--|
| \mathbb{M}_{DFA} | $\delta_{DFA}: Q \times \Sigma \rightarrow Q$ |
| $\mathbb{M}_{부분}$ | $\delta_{부분}: Q \times \Sigma \rightarrow Q \cup \{\emptyset\}$ |
| \mathbb{M}_{NFA} | $\delta_{NFA}: Q \times \Sigma \rightarrow 2^Q$ |
| $\mathbb{M}_{\epsilon-NFA}$ | $\delta_{\epsilon-NFA}: Q \times \{\Sigma \cup \{\epsilon\}\} \rightarrow 2^Q$ |
| \mathbb{M}_{FA} | $\delta_{FA}: Q \times \Sigma^* \rightarrow 2^Q$ |