정규언어

제 1 절 정규식

수식(arithmetic expression)과 유사하게 정규식(regular expression)을 정의한다.

정의 1 *알파벳 Σ 상의 정규식과 그것이 표시하는 집합은 다음과 같다*

1. *$\emptyset$는 정규식이고 공집합을 표시한다.*
2. *ϵ는 정규식이고 $\{\epsilon\}$을 표시한다.*
3. *각 $a \in \Sigma$는 정규식이고 $\{a\}$를 표시한다.*
4. *r과 s가 정규식이고 각각 R과 S 집합을 표시한다면, $(r+s)$, (rs), (r^*)는 정규식이고 각각 $R \cup S$, RS, R^* 집합을 표시한다.*

정규식 r의 언어 $L(r)$은 r이 표시하는 집합이다.

정규식 a와 글자 a는 서로 다른 것을 나타내나, 같은 형태임을 유의하라. 즉 $L(a) = \{a\}$이고 $L(\epsilon) = \{\epsilon\}$이다.

정규식에서 연산의 순서를 $*\cdot+$로 정하면 많은 괄호를 생략할 수 있다. 즉 $((1(0^*))+0) = 10^*+0$이고 이 정규식은 $\{0, 1, 10, 100, \ldots\}$을 나타낸다. 또한 다음과 같은 연산의 법칙을 적용할 수 있다.

- $+$ 연산은 합집합을 나타내므로, 교환법칙과 결합법칙이 성립한다. 즉 $r+s = s+r$이고 $(r+s)+t = r+(s+t)$이다.
- $\cdot$ 연산은 접합을 나타내므로, 교환법칙은 성립하지 않고 결합법칙은 성립한다. 즉 $(rs)t = r(st)$이다.
- $+$와 $\cdot$ 사이에 분배법칙이 성립한다. 즉 $r(s+t) = rs+rt$이고 $(r+s)t = rt+st$이다.

정의 2 *정규식으로 표시되는 언어를 정규언어라고 부른다.*

예제 1 *정규식*

$$(00)^*(11)^*1$$

은 짝수 개의 0 다음에 홀수 개의 1이 나오는 스트링의 집합을 표시한다.

예제 2 $\{0^n1^m : (n+m)$*은 짝수*$\}$*를 나타내는 정규식을 구하라.*

정규식의 연산 $+$는 "또는"의 의미이므로, 정규식을 구할 때 여러 경우로 나누어서 생각하는 것이 좋다. 0이 짝수 개 나오고 1이 짝수 개 나오든지 0이 홀수 개 나오고 1이 홀수 개 나와야 되므로, 답은 $(00)^*(11)^* + 0(00)^*1(11)^*$이다.

앞으로 예제에서 알파벳에 대한 별다른 언급이 없으면
$\Sigma = \{0, 1\}$이다.

예제 3 *1이 한 개 또는 두 개 있는 스트링의 집합을 표시하는 정규식을 구하라.*

1이 한 개 있는 스트링의 집합은 0^*10^*이고, 1이 두 개 있는 스트링의 집합은 $0^*10^*10^*$이므로 둘을 합치면 $0^*10^*(\epsilon + 10^*)$이다.

예제 4 *길이가 3의 배수인 스트링의 집합을 표시하는 정규식을 구하라. 길이가 3인 스트링을 표시하는 정규식이*
$(0+1)(0+1)(0+1)$*이므로 답은* $((0+1)(0+1)(0+1))^*$*이다.*
$(0+1)(0+1)(0+1)$*을 줄여서* $(0+1)^3$*으로 쓰기로 한다.*

예제 5 *11을 부분스트링으로 갖는 스트링의 집합을 표시하는 정규식은* $(0+1)^*11(0+1)^*$ *이다.*

예제 6 *111이 딱 한 번 나타나는 스트링의 집합을 표시하는 정규식을 구하라.*

예제 7 *11을 부분스트링으로 갖지 않는 스트링의 집합을 표시하는 정규식을 구하라.*

예제 8 $\Sigma = \{a, b, c\}$*일 때, 첫 글자가 다시 나타나지 않는 스트링의 집합을 표시하는 정규식을 구하라.*

언어를 표현하는 방식에는 정규식, 오토마타, 문법 등이 있다.

정의 3 *언어의 표현 방식* A, B*가* $L(A) = L(B)$*를 만족하면,* A*와* B*는 동등(equivalent)하다.*

제 2 절 유한 오토마타

컴퓨터는 일반적으로 중앙처리장치, 메모리, 입출력장치로 구성되어 있다. 앞으로 컴퓨터의 이론적인 모델들을 배우게 되는데, 그 중에서 가장 간단한 것이 유한 오토마타이다. 유한 오토마타는 중앙처리장치에 해당하는 유한제어기와 입력장치인 입력테입으로 구성된다. 유한 오토마타의 특징은 메모리를 가지고 있지 않다는 것이다. 입력테입은 칸들로 나누어지고 각 칸은 하나의 글자를 가질 수 있다. 입력테입에는 헤드(head)가 있어서 헤드가 가리키는 칸에 있는 글자를 읽을 수 있다. 유한제어기는 상태들로 구성된다. 그림 1를 보라.

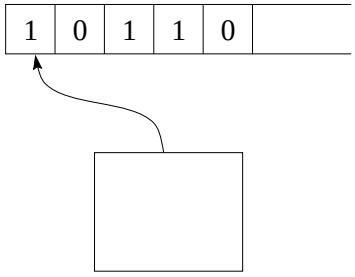

그림 1:

초기에 헤드는 입력테입의 맨왼쪽 칸을 가리키고, 유한제어기는 초기 상태에 놓여 있다. 각 단계에서 현재 상태와 테입의 글자에 의해서 다음 상태가 결정되고, 헤드는 한 칸 오른쪽으로 진행한다.

정의 4 *결정 유한 오토마타 M은 다섯 가지 요소 $(Q, \Sigma, \delta, q_0, F)$로 구성된다. 여기에서*

1. *Q는 상태들의 유한 집합이고*
2. *Σ는 알파벳이고*

3. *전이함수 δ는 $Q \times \Sigma$에서 Q로의 함수이고*
4. *$q_0 \in Q$는 초기상태이고*
5. *$F \subseteq Q$는 최종상태들의 집합이다.*

결정 유한 오토마타(deterministic finite automata)를 줄여서 DFA로 부르기로 하자.

전이 함수 δ를 확장하여 $Q \times \Sigma^*$에서 Q로의 함수 δ^*를 다음과 같이 정의한다.

- $\delta^*(q, \epsilon) = q$
- 모든 $w \in \Sigma^*$와 $a \in \Sigma$에 대하여, $\delta^*(q, wa) = \delta(\delta^*(q, w), a)$

스트링 $w \in \Sigma^*$에 대하여 $\delta^*(q_0, w)$이 DFA M의 최종상태이면, M이 w를 받아들인다고 말한다. M의 언어 $L(M)$은 M이 받아들이는 모든 스트링의 집합이다. 즉

$$L(M) = \{w \in \Sigma^* : \delta^*(q_0, w) \in F\}.$$

DFA의 상황(configuration)은 현재 상태와 입력 스트링의 읽지 않은 부분으로 결정된다. 입력 스트링이 w일 때, DFA의 시작 상황은 (q_0, w) 이고, 종료 상황은 어떤 $q \in Q$에 대하여 (q, ϵ) 이다.

한 번의 전이에 의해서 DFA M의 상황이 변화하는 과정을 $\vdash_M$으로 표시한다. 즉 현재 상황이 $(q, 10110)$이고 $\delta(q, 1) = q'$이면,

$$(q, 10110) \vdash_M (q', 0110)$$

이다. DFA M이 명확한 경우 $\vdash_M$ 대신 $\vdash$를 사용할 수 있다. 0번 이상의 전이를 $\vdash^*$로 표시한다.

예제 9 $M = (\{q_0, q_1\}, \{0, 1\}, \delta, q_0, \{q_0\})$ *이고,* δ*는 다음과 같다.*

	0	1
q_0	q_0	q_1
q_1	q_1	q_0

$L(M)$*은 짝수 개의 1을 가진 스트링의 집합이다.*

유한 오토마타를 이해하기 쉽도록 FA그림으로 나타낸다. 상태는 정점으로, 전이 $\delta(p, a) = q$는 상태 p에서 q로 가는 라벨 a가 붙은 간선으로, 최종상태는 이중원으로, 초기상태는 화살표로 표시한다. 예제 9의 DFA를 그림으로 나타내면 그림 2와 같다.

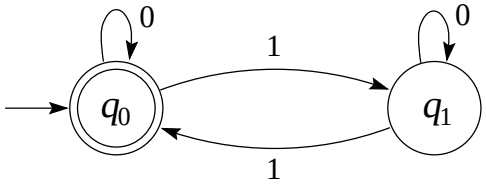

그림 2:

일반적으로 다음 두 단계에 의해 DFA를 만든다.

1. 현재까지 읽은 부분에서 어떤 정보를 기억해야 하는지를 정하고 이 정보를 상태로 표시한다.
2. 새로운 글자를 읽었을 때 기억해야 하는 정보가 어떻게 바뀌는지를 보고 전이함수를 정한다.

이 중에서 중요한 것은 기억해야 하는 정보를 정하는 것이다. 예제 9에서는 1의 개수가 짝수인지 홀수인지를 기억해야 한다. 따라서 상태 q_0은 1의 개수가 짝수, q_1은 1의 개수가 홀수임을 나타낸다.

예제 10 *$L = \{w : N_0(w) \bmod 2 > N_1(w) \bmod 2\}$을 받아들이는 DFA를 구하라. 여기에서 $N_a(w)$는 w에 있는 a의 개수를 나타낸다. 즉, 이 DFA는 0의 개수는 홀수이고 1의 개수는 짝수인 스트링을 받아들여야 된다.*

이 경우 $N_0(w) \bmod 2$와 $N_1(w) \bmod 2$를 기억하면 된다. 그림 3.

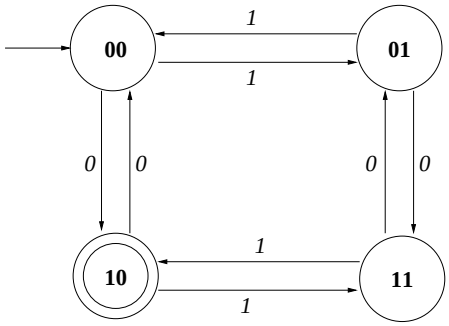

그림 3:

예제 11 *0101을 부분스트링으로 갖는 스트링을 받아들이는 DFA를 구하라.*

이 문제에서는 0101의 어두 중에서 현재까지 읽은 부분을 기억해야 한다. 따라서 0101의 어두를 상태로 표시하는 DFA를 만들면 된다. (그림 4) 이제 전이함수를 정한다. 예를 들면, 현재까지 읽은 어두가 0이고, 그 다음에 1을 읽으면 읽은 어두가 01이 되고, 0을 읽으

면 읽은 부분이 *00*이 되는데 *0101*의 어두에 해당되는 것은 *0*이다.

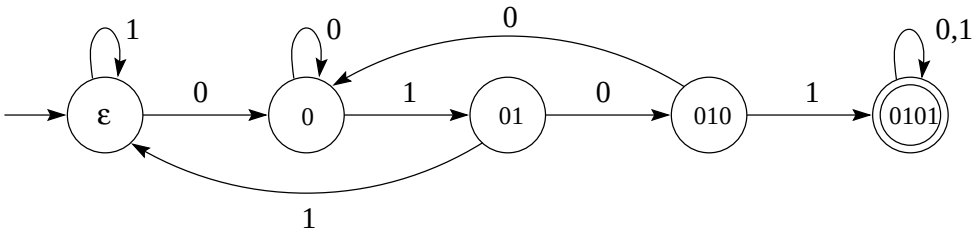

그림 4:

예제 12 *0101*을 부분스트링으로 갖지 않는 스트링을 받아들이는 *DFA*를 구하라.

*그림 4의 DFA*에서 최종상태와 그 외의 상태를 맞바꾸면 된다. *(*이와 같이 여집합을 표시하는 문제가 정규식에서는 어려웠는데, *DFA*에서는 쉬운 것을 주목하라.*)*

예제 13 스트링에서 연속*(run)*이란 같은 글자로만 구성된 *(*더 이

상 늘릴 수 없는*)* 부분스트링을 말한다. 예를 들면, *0110001*은 길이가 *2*인 *1*의 연속과 길이가 *3*인 *0*의 연속을 가지고 있다. 길이가 *2*인 연속을 갖지 않는 스트링을 받아들이는 *DFA*를 구하라. 즉 스트링이 길이가 *2*인 연속은 갖지 않아야 하고 길이가 *1*이나 *3*이상인 연속은 가질 수 있다.

이 문제에서는 가장 최근의 연속을 기억해야 하고 그 중에서 길이가 *3*이하인 연속을 기억하면 된다.

예제 14 *010*을 부분스트링으로 가지고, *100*을 부분스트링으로 갖지 않는 스트링을 받아들이는 *DFA*를 구하라.

이 문제는 약간 어렵다. *010*이나 *100*이 부분스트링으로 있는지 없는지를 알기 위해서는 앞의 예제에서처럼 *010*과 *100*의 어두를 기억해야 한다. 동시에 현재 *100*을 읽고 있다면 이전에 *010*이 있었는지 없었는지를 알고 있어야 한다. 반대로 현재 *010*을 읽고 있을 때 이전에 *100*이 있었는지를 기억할 필요는 없다. 왜냐하면 *100*이 있었다면 이 스트링은 받아들여지지 않기 때문이다.

δ가 부분함수일 때도 DFA로 볼 수 있다. 이 경우 위의 정의에 의한 DFA로 바꾸려면, 새로운 상태 $q' \notin F$을 도입하여

- 정의되지 않은 모든 전이는 q'으로 가도록 하고,
- 모든 $a \in \Sigma$에 대하여 $\delta(q', a) = q'$이 되도록 한다.

q'과 같이 한 번 들어가면 빠져 나올 수 없는 상태를 죽은 상태(dead state)라고 부른다. 예제 12에서 상태 0101도 죽은 상태이다.

제 3 절 비결정 유한 오토마타

비결정 유한 오토마타(nondeterministic finite automata)를 줄여서 NFA로 부르기로 하자. NFA는 다음 두 가지 면에서 DFA를 일반화한 것이다.

1. 각 상태와 글자가 주어졌을 때, 다음 상태가 없거나 하나 이상일 수 있다.

2. 각 상태에서 공스트링 ϵ에 의한 전이가 있을 수 있다.

정의 5 *비결정 유한 오토마타 M은 다섯 가지 요소 $(Q, \Sigma, \Delta, q_0, F)$로 구성된다. 여기에서 Q, Σ, q_0, F는 DFA에서와 같이 정의된다. Δ는 $Q \times (\Sigma \cup \{\epsilon\}) \times Q$의 부분집합인 전이 관계로, 또는 $Q \times (\Sigma \cup \{\epsilon\})$에서 2^Q로의 전이 함수로 정의할 수 있다.*

NFA를 정의할 때는 Δ를 DFA의 전이 함수에서 일반화된 전이 관계로 파악하는 것이 자연스러우나, 설명에서는 Δ를 전이 함수로 보는 것이 편리하므로 이후에는 Δ를 2^Q으로의 함수로 간주한다. 또한 설명의 편의를 위해서 Δ를 $2^Q \times (\Sigma \cup \{\epsilon\})$에서의 함수로 다음과 같이 확장한다. $P \subseteq Q$와 $a \in \Sigma \cup \{\epsilon\}$에 대하여

$$\Delta(P, a) = \bigcup_{q \in P} \Delta(q, a).$$

임의의 상태 q에 대하여, $E(q)$를 q에서부터 입력 글자를 읽지 않고 (ϵ 라벨이 붙은 간선을 따라) 도달할 수 있는 상태의 집합이라고 하자. E도 역시 확장하여, 상태 집합 P에 대하여 $E(P) = \bigcup_{q \in P} E(q)$.

Δ를 확장하여 $Q \times \Sigma^*$에서 2^Q으로의 함수 Δ^*를 다음과 같이 정의한다.

- $\Delta^*(q, \epsilon) = E(q)$
- 모든 $w \in \Sigma^*$와 $a \in \Sigma$에 대하여, $\Delta^*(q, wa) = E(\Delta(\Delta^*(q, w), a))$

스트링 $w \in \Sigma^*$에 대하여 $\Delta^*(q_0, w)$이 NFA M의 최종상태를 적어도 하나 포함하면, M이 w를 받아들인다고 말한다. M의 언어 $L(M)$은 M이 받아들이는 모든 스트링의 집합이다. 즉

$$L(M) = \{w \in \Sigma^* : \Delta^*(q_0, w) \cap F \neq \emptyset\}.$$

또는 전이과정을 사용하여 $L(M)$을 다음과 같이 정의할 수 있다.

$$L(M) = \{w \in \Sigma^* : (q_0, w) \vdash_M^* (q, \epsilon), q \in F\}.$$

일반적으로 NFA는 DFA보다 훨씬 쉽게 언어를 표현할 수 있다.

예제 15 *정규식 $(010 + 01)^*$가 표시하는 언어를 받아들이는 NFA를 구하라.*

그림 5. 입력 스트링이 010일 때,

- $\Delta^*(q_1, 0) = \{q_2\}$,
- $\Delta^*(q_1, 01) = \{q_3, q_1\}$,
- $\Delta^*(q_1, 010) = \{q_1, q_2\}$

이고, 여기에 최종상태 q_1이 포함되어 있으므로 이 NFA는 010을 받아들인다. 이것을 전이 과정으로 나타내면 다음과 같다.

$$(q_1, 010) \vdash (q_2, 10) \vdash (q_3, 0) \vdash (q_1, \epsilon)$$

즉 $(q_1, 010) \vdash^* (q_1, \epsilon)$이고 q_1이 최종상태이므로 이 NFA는 010을 받아들인다.

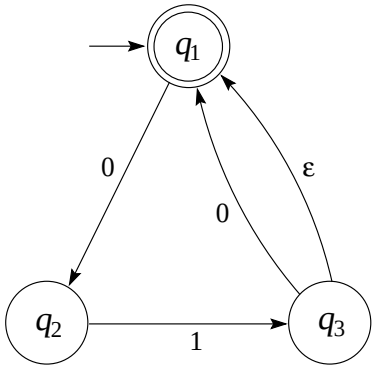

그림 5:

예제 16 *끝에서 두 번째 글자가 1인 스트링을 받아들이는 NFA를 구하라. 이 NFA는 q_1에 머물다가 끝에서 두 번째 글자를 추측하여 그 글자가 1이면 q_2로 이동한다. q_3는 이 추측이 맞았는지 확인한다.*

그림 6. 입력 스트링이 011일 때,

- $\Delta^*(q_1, 0) = \{q_1\}$,
- $\Delta^*(q_1, 01) = \{q_1, q_2\}$,
- $\Delta^*(q_1, 011) = \{q_1, q_2, q_3\}$

이므로, 이 NFA는 011을 받아들인다. 입력스크링이 100일 때,

- $\Delta^*(q_1, 1) = \{q_1, q_2\}$,
- $\Delta^*(q_1, 10) = \{q_1, q_3\}$,
- $\Delta^*(q_1, 100) = \{q_1\}$

이므로, 이 NFA는 100을 받아들이지 않는다.

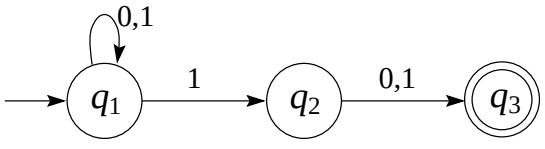

그림 6:

NFA의 최종상태는 한 개라고 가정할 수 있다. NFA의 Q에 두 개 이상의 최종상태가 있으면, 새로운 상태 q_f만을 최종상태로 하고 모든 $q \in F$에 대해 전이 (q, ϵ, q_f)를 더해준다.

왜 비결정성(nondeterminism)이 필요한가? 현재 사용되는 디지털 컴퓨터는 완전히 결정적으로 움직인다. 또 미래에 비결정적으로 작동되는 컴퓨터를 만들 가능성은 적다. (현재 quantum computer, DNA computing 등이 시도되고 있지만) 그럼에도 불구하고, 다음과 같은 이유로 비결정성은 중요한 역할을 한다.

- 비결정성을 이용하면 오토마타를 쉽게 만들 수 있다. 어떤 언어를 받아들이는 NFA를 만드는 것이 일반적으로 DFA를 만드는 것보다 쉽다.
- 문제의 세계와 문제를 푸는 도구의 세계를 생각해보자. 문제의 세계는 우리의 문제해결 능력과 관계없이 자연에 주어져있는 것이다. 이에 비해 디지털 컴퓨터 등은 우리가 가지고 있는 문제해결도구이다. 현재 우리가 가지고 있는 문제해결도구는 결

정적으로 움직이지만, 문제의 세계에는 비결정성이 이미 존재한다. 따라서 문제의 세계를 이해하기 위해서는 비결정성이라는 개념이 필요하다.

제 4 절 유한 오토마타의 동등성

DFA가 받아들이는 언어 종류와 NFA가 받아들이는 언어 종류가 같음을 보이고자 한다. DFA는 NFA의 일종이므로 DFA의 언어 종류는 NFA의 언어 종류의 부분집합이다. 이제 반대의 경우를 증명하자.

정리 1 *임의의 NFA에 대하여 이와 동등한 DFA가 존재한다.*

증명. $N = (Q_N, \Sigma, \Delta, q_0, F_N)$을 임의의 NFA라고 하자. 이제 N과 동등한 DFA $D = (Q_D, \Sigma, \delta, q', F_D)$를 만들고자 한다. 증명의 핵심은 어느 시점에서 NFA의 현재 상태들의 집합을 DFA의 하나의 상태로 간주하는 것이다. 즉 DFA D의 상태는 Q_N의 멱집합의 원소이

다.

DFA D의 초기 상태 q'는 $E(q_0)$이다. D의 상태 $P \subseteq Q_N$와 글자 $a \in \Sigma$에 대하여, D의 전이 함수는 $\delta(P, a) = E(\Delta(P, a))$이다. 이를 이용하여 다음과 같이 DFA D를 구한다.

$Q_D \leftarrow \{E(q_0)\}$
mark $E(q_0)$
while $\exists$ marked state $P \in Q_D$ do
    unmark P
    for each $a \in \Sigma$ do
        $R \leftarrow E(\Delta(P, a))$
        if R is not in Q_D then
            add R as marked state to Q_D fi
        $\delta(P, a) \leftarrow R$
    od
od

마지막으로, D의 상태 P가 F_N의 원소를 적어도 하나 포함하면 P는 F_D의 원소이다.

이제 NFA N과 DFA D가 동등함을 보이자. 입력 스트링의 길이에 대한 귀납법으로

$$\Delta^*(q_0, w) = \delta^*(E(q_0), w) \tag{1}$$

임을 증명한다.

1. $|w| = 0$일 때, $\Delta^*(q_0, \epsilon) = E(q_0)$ 이고 $\delta^*(E(q_0), \epsilon) = E(q_0)$이므로 (1)가 성립한다.
2. $|w| < k$일 때, (1)가 성립한다고 가정하자.
3. $|w| = k \geq 1$일 때, $w = ua$ $(u \in \Sigma^*, a \in \Sigma)$ 라고 하자. 정의에 의해 NFA N에서는 $\Delta^*(q_0, w) = E(\Delta(\Delta^*(q_0, u), a))$이고, DFA D에서는
   $\delta^*(E(q_0), w) = \delta(\delta^*(E(q_0), u), a) = E(\Delta(\delta^*(E(q_0), u), a)$ 이다.
   귀납법 가정에 의해 $\Delta^*(q_0, u) = \delta^*(E(q_0), u)$이므로 (1)가 성립한다.

따라서 $\Delta^*(q_0, w)$가 F_N의 원소를 적어도 하나 포함하는 경우에만 $\delta^*(E(q_0), w) \in F_D$이다. 즉 $L(N) = L(D)$. □

예제 17 *예제 16의 NFA와 동등한 DFA를 구하라.*

그림 7. 이 DFA를 직접 구하기 위해서는 DFA가 읽은 이전 두 글자를 기억해야 한다. 이전 두 글자는 00, 01, 10, 11 중 하나이므로 이들을 상태로 나타내면 된다. 그림 7에서 상태 $\{1\}$이 00, $\{1,2\}$가 01, $\{1,3\}$이 10, $\{1,2,3\}$이 11이 된다.

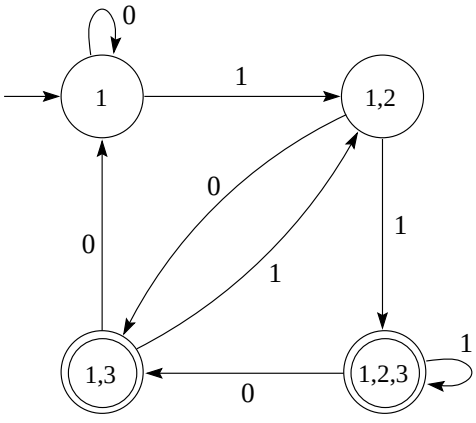

그림 7:

예제 18 *예제 15의 NFA와 동등한 DFA를 구하라.*

그림 8. 이 DFA에서 공집합을 나타내는 상태는 (항상) 죽은 상태이다.

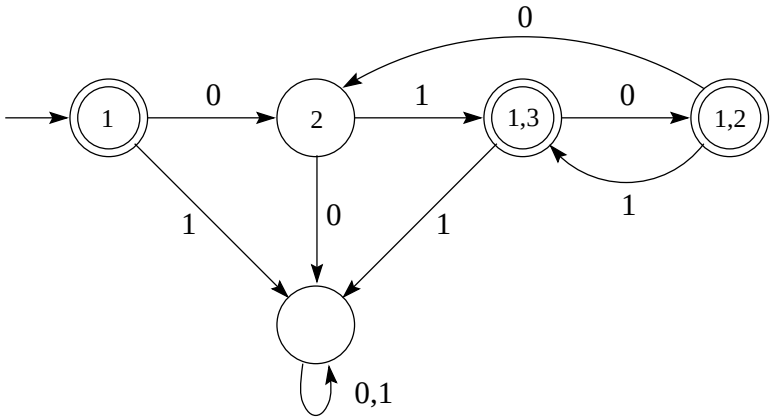

그림 8:

제 5 절 정규식과 유한 오토마타

앞 장에서 DFA의 언어 종류와 NFA의 언어 종류가 같음을 보았는데, 이제 이 언어 종류가 정규언어 종류임을 보이고자 한다.

정리 2 *r을 정규식이라고 하면, $L(r)$을 받아들이는 NFA가 존재한다.*

증명. 정규식은 정의 1에 의해 구성되므로 각 경우에 대하여 동등한 NFA가 있음을 보이면 된다. 이 과정에서 만들어지는 NFA는 항상 다음 조건을 만족한다.

- 최종상태는 하나이다.
- 초기상태로 들어가는 전이와 최종상태에서 나가는 전이가 없다.

정규식 $\emptyset, \epsilon, a \in \Sigma$와 동등한 NFA는 그림 9과 같다. 정규식 r, s와 동등한 NFA를 각각 R, S 라고 하면 $(r+s), (rs), (r^*)$와 동등한 NFA는 그림 10, 11, 12과 같다. □

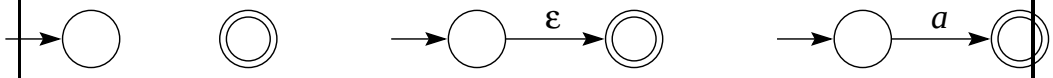

그림 9:

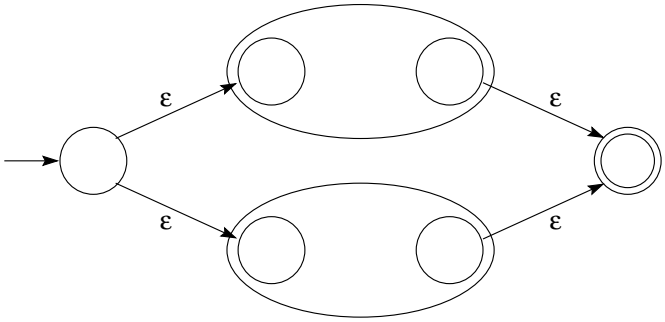

그림 10:

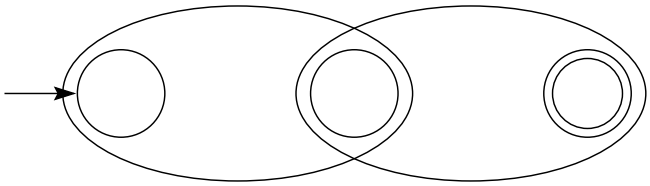

그림 11:

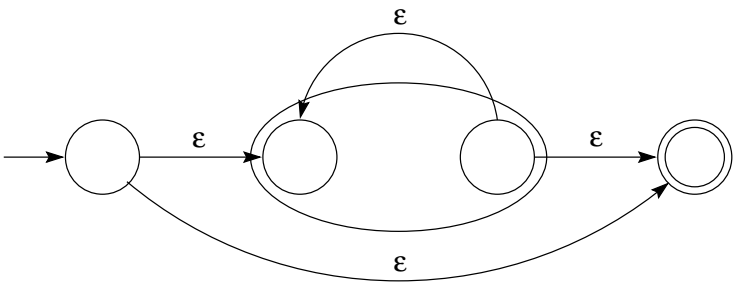

그림 12:

예제 19 *정규식* $(0+11)^*$*와 동등한 NFA를 구하라. 그림 13.*

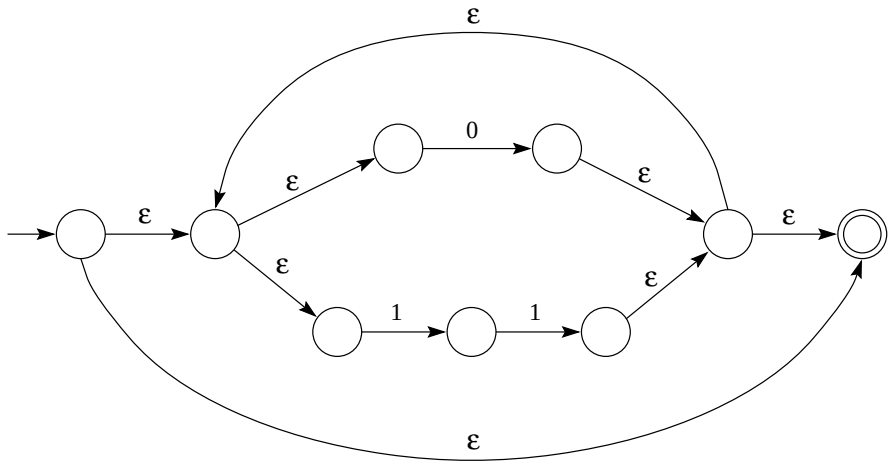

그림 13:

정리 3 *DFA* M*의 언어* $L(M)$*에 대하여,* $L(M)$*을 표시하는 정규식이 있다.*

증명. $M = (\{q_1, \ldots, q_n\}, \Sigma, \delta, q_1, F)$라고 하자. R_{ij}^k를 k이하의 (번호를 가진) 상태만을 지나면서 M을 q_i에서 q_j로 전이시키는 스트링의 집합이라고 하자. 모든 상태는 n이하이므로, R_{ij}^n는 q_i에서 q_j로 전이시키는 모든 스트링의 집합이다. 따라서

$$L(M) = \bigcup_{q_j \in F} R_{1j}^n.$$

R_{ij}^k는 다음과 같이 재귀적으로 구해질 수 있다.

$$\begin{aligned} R_{ij}^0 &= \begin{cases} \{a : \delta(q_i, a) = q_j\} & \text{if } i \neq j \\ \{a : \delta(q_i, a) = q_j\} \cup \{\epsilon\} & \text{if } i = j \end{cases} \\ R_{ij}^k &= R_{ij}^{k-1} \cup R_{ik}^{k-1}(R_{kk}^{k-1})^* R_{kj}^{k-1} \end{aligned} \tag{2}$$

식 2의 의미는 k이하의 상태를 지나면서 q_i에서 q_j로 가기 위해서는

1. $k-1$이하의 상태를 지나면서 q_i에서 q_j로 가든지

2. 각 경우에 $k-1$이하의 상태를 지나면서, q_i에서 q_k로 가고 q_k에서 q_k로 0번 이상 순환하고 q_k에서 q_j로 가면 된다는 것이다.

식 2에서 R_{ij}^k는 $*, \cdot, \cup$ 연산에 의해 표시되므로 정의 1에 의해 R_{ij}^k 집합을 표시하는 정규식이 있음을 k에 대한 귀납법으로 증명할 수 있다. $L(M)$은 각 $q_j \in F$에 대하여 R_{1j}^n의 합집합이므로, 이에 대한 정규식도 역시 존재한다. □

예제 20 *예제 18의 DFA M과 동등한 정규식을 구하라. M에서 $q_1 = \{1\}$, $q_2 = \{2\}$, $q_3 = \{1,3\}$, $q_4 = \{1,2\}$ 라고 하자. $\emptyset$은 죽은 상태이므로 고려하지 않아도 된다. 이후의 전개에서 다음 식을 이용한다.*

$$\begin{aligned}(st)^*s &= (\epsilon + st + stst + \cdots)s \\ &= s + sts + ststs + \cdots \\ &= s(\epsilon + ts + tsts + \cdots) \\ &= s(ts)^*\end{aligned}$$

집합 R_{ij}^k를 나타내는 정규식을 r_{ij}^k라고 하자 (즉 $R_{ij}^k = L(r_{ij}^k)$). 식 2에 의해, 구하는 정규식은

$$r = r_{11}^4 + r_{13}^4 + r_{14}^4$$

이다. $r_{11}^4 = \epsilon$임을 쉽게 알 수 있다.

$$\begin{aligned}r_{13}^4 &= r_{13}^3 + r_{14}^3 (r_{44}^3)^* r_{43}^3 \\ &= 01 + 010(\epsilon + 010 + 10)^*(01 + 1) \\ &= 01 + 010((01 + 1)0)^*(01 + 1) \\ &= 01 + 01(001 + 01)^*(001 + 01) \\ &= 01(001 + 01)^*\end{aligned}$$

마찬가지로 $r_{14}^4 = 010(010 + 10)^*$ 이다. 따라서

$$\begin{aligned}r &= \epsilon + 01(001 + 01)^* + 010(010 + 10)^* \\ &= \epsilon + (010 + 01)^*01 + (010 + 01)^*010 \\ &= (010 + 01)^*\end{aligned}$$

제 6 절 정규언어의 성질

정리 4 *정규언어 종류는 다음 연산에 대하여 닫혀 있다: (1) 합집합, (2) 접합, (3) Kleene 곱, (4) 여집합, (5) 교집합.*

증명. 정규식의 정의에 의해 정규언어는 합집합, 접합, Kleene 곱에 대해 닫혀 있다.
(4) 정규언어 L을 받아들이는 DFA를 $(Q, \Sigma, \delta, q, F)$라고 하자. 그러면 DFA $(Q, \Sigma, \delta, q, Q - F)$가 여집합 $\overline{L} = \Sigma^* - L$을 받아들인다.
(5) 다음 식에 의해 교집합에 대하여 닫혀 있다.

$$L_1 \cap L_2 = \overline{\overline{L_1} \cup \overline{L_2}}$$

□

정규언어에 대하여 여러 가지 질문을 던질 수 있다. 다음의 질문에 대하여 답하는 방법(알고리즘)이 있다.

예제 21 (소속문제) *정규언어 L과 스트링 w가 주어졌을 때, w가 L에 속하는지 결정하라.*

L이 정규식으로 주어지면 그와 동등한 유한 오토마타로 바꾼 후, w를 읽어서 받아들이는지 결정한다.

예제 22 *유한 오토마타 M이 주어졌을 때, $L(M) = \emptyset$ 인지 결정하라.*

M을 DFA로 바꾼 후, 초기상태에서 최종상태로 가는 경로가 있는지 깊이 우선 탐색(depth-first search)이나 너비 우선 탐색(breadth-first search)를 이용하여 확인한다.

예제 23 *두 정규언어 L_1, L_2가 주어졌을 때, $L_1 = L_2$ 인지 결정하라.*

$L_1 = L_2$를 증명하기 위해서는 $L_1 \subseteq L_2$와 $L_2 \subseteq L_1$을 보이면 된다. $L_1 \subseteq L_2$는 $L_1 \cap \overline{L_2} = \emptyset$과 같으므로, $L_1 \cap \overline{L_2}$를 받아들이는 DFA M을 만든 후 $L(M) = \emptyset$ 인지 확인한다.

어떤 언어가 정규언어가 아님을 보이기 위해서는 다음 정리를 사용한다.

정리 5 (펌프 정리) *L을 무한 정규언어라고 하면, 다음을 만족하는 양의 정수 t가 존재한다. 길이가 t 이상인 임의의 스트링 $w \in L$는 $w = xyz$로 표현되며 여기서*

1. $|xy| \leq t$,
2. $|y| \geq 1$,
3. *모든 $i \geq 0$에 대하여 $xy^iz \in L$ 이다.*

증명. L이 정규언어이므로 L을 받아들이는 DFA $M = (Q, \Sigma, \delta, q_0, F)$가 존재한다. M의 상태의 갯수를 t라 하자. $w = a_1a_2 \cdots a_n$ $(n \geq t)$에 대하여 $\delta^*(q_0, a_1 \cdots a_i) = q_i$라고 하자. M은 t 개의 상태만을 가지므로, 처음 $t+1$ 개의 상태 $q_0, q_1, \ldots, q_t$가 전부 다를 수는 없다. 즉 $q_j = q_k$인 두 수 j, k $(0 \leq j < k \leq t)$가 존재한다.

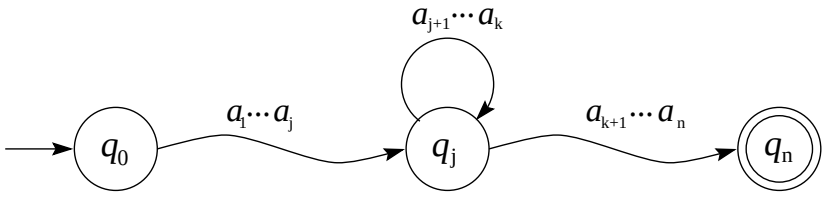

그림 14:

$x = a_1 \cdots a_j$, $y = a_{j+1} \cdots a_k$, $z = a_{k+1} \cdots a_n$이라고 하면, $|xy| \leq t$이고 $|y| \geq 1$이다. $w \in L$이므로 $q_n \in F$이다. 따라서 모든 $i \geq 0$에 대해서 $\delta^*(q_0, xy^iz) = q_n$이므로 $xy^iz \in L$이다. 즉 L에 속한 충분히 긴 스트링은 세 부분으로 나눌 수 있는데, 중간 부분인 y를 몇 번 펌프질 하든지 그 결과는 역시 L에 속한다는 의미로 펌프 정리라는 용어를 사용한다. □

펌프 정리를 적용하여 언어 L이 정규언어가 아님을 보이고자 할 때는 귀류법을 사용한다. 즉 L이 정규언어라고 가정하고, 펌프 정리

에 모순됨을 보이면 된다. 펌프 정리 중 t와 xyz는 '존재한다'는 조건을 갖고, w와 i는 '모든'의 조건을 가지므로 다음과 같이 증명을 전개한다. (t와 xyz는 존재한다는 사실만 알고 실제 값을 모르므로 어떤 값에 대해서든지 증명할 수 있어야 되고, w와 i는 모든 값에 대해서 성립하므로 증명에 가장 유리한 값을 선택하면 된다.)

1. 상대방(adversary)이 t를 선택한다.
2. 길이가 t 이상인 $w \in L$를 선택한다.
3. 상대방이 w를 $|xy| \leq t$, $|y| \geq 1$ 조건을 만족시키는 x, y, z로 나눈다.
4. i를 선택하여 xy^iz가 L에 속하지 않음을 보인다.

이 과정 중에서 가장 중요한 것은 2 단계로서 w를 선택할 때, 3 단계에서 상대방이 x, y, z를 선택할 여지가 적어지도록 하여야 한다.

예제 24 $L = \{0^n1^n : n \geq 0\}$*은 정규언어가 아님을 증명하라.*

L이 정규언어라고 가정하자. 그러면 정리 5를 만족하는 양의 정수 t가 존재한다. 스트링 $w = 0^t1^t \in L$을 고려하자. $|xy| \leq t$이므로 y는

0^k $(1 \leq k \leq t)$ 이어야 한다. 그러면 $i = 0$일 때 $0^{t-k}1^t$은 L에 속하지 않으므로 정리 5에 모순된다.

예제 25 $L = \{uu : u \in \{0,1\}^*\}$*는 정규언어가 아님을 증명하라.*

예제 26 $L = \{a^{n^2} : n \geq 1\}$*는 정규언어가 아님을 증명하라.*

예제 27 $L = \{0^m1^n : m \neq n\}$*은 정규언어가 아님을 증명하라.*

L이 정규언어라고 가정하면, 정리 5를 만족하는 t가 존재한다. 이 경우 $w = 0^t1^{t+1}$을 선택하면, 상대방은 $y = 00$를 선택하여 정리 5가 만족되므로 증명에 실패하게 된다. 따라서 상대방이 어떤 $y = 0^k$를 선택하든지 펌프질 하다보면 1의 개수와 같아지도록 w를 잡아야 된다. 즉 $w = 0^{t!}1^{(t+1)!}$을 선택한다. $|xy| \leq t$이므로 $y = 0^k$ $(1 \leq k \leq t)$ 이어야 한다. 그러면

$$i = \frac{tt!}{k} + 1$$

일 때, $xy^iz = 0^{t!}0^{k(i-1)}1^{(t+1)!} = 0^{(t+1)!}1^{(t+1)!}$이므로 L에 속하지 않는다.

보다 간단한 증명은 다음과 같다. L이 정규언어라고 가정하자. 그러면 $\overline{L} \cap L(0^*1^*)$는 정규언어이다. 그런데 이 언어가 $\{0^n 1^n : n \geq 0\}$이므로 모순이다.

제 7 절 유한 오토마타의 응용

유한 오토마타는 논리회로 설계에 광범위하게 사용된다.

예제 28 *패리티 검사기(parity checker)*는 $0, 1$로 구성된 스트링을 읽어서 *1*의 개수를 검사하는 회로이다. 짝수(홀수) 패리티 검사기는 *1*의 개수가 짝수(홀수)일 때 받아들이는 회로이다. 예제 *9*의 *DFA*는 짝수 패리티 검사기를 나타낸다. 이 *DFA*에 대한 하드웨어 구현은 *[Ka]*를 참고하라.

예제 29 *과자 한 봉지에 200원 받는 자동판매기가 있는데, 이것은 100원이나 50원 동전을 받아서 200원이 되면 과자 한 봉지를 내보낸다. 100원 동전을 a로, 50원 동전을 b로 표시할 때, 이 자동판매*

기의 제어기를 설계하라.

그림 15. 이 그림에서 150원 받은 상태에서 100원이 주어지는 경우가 없음에 유의하라. (즉 전이함수가 부분함수이다.) 이 제어기는 정확하게 200원이 주어졌을 때만 작동한다. 거스름돈을 내어주기 위해서는 제어기가 더 복잡해져야 된다.

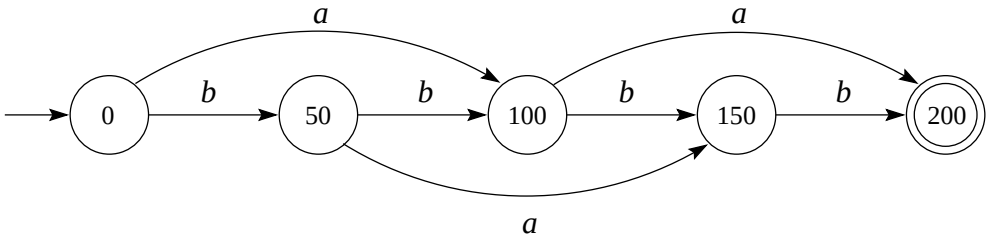

그림 15:

정규식을 유한 오토마타로 변환하는 것도 여러 곳에 응용된다.

예제 30 *Pascal 언어에서 변수 이름은 한 개의 영문자 뒤에 영문자*

또는 숫자를 반복적으로 붙인 것이므로,

$$(letter)(letter + digit)^*$$

로 표시할 수 있다. 여기에서 *letter*는 $A + B + \cdots + Z$를 나타내고 *digit*은 $0 + 1 + \cdots + 9$를 나타낸다. *FORTRAN* 언어에서 변수 이름은 *Pascal* 언어와 마찬가지이나 길이가 최대 6이므로

$$(letter)(\epsilon + letter + digit)^5$$

으로 표시할 수 있다. 컴파일러는 위의 정규식을 유한 오토마타로 바꾸어서 프로그램을 읽을 때 변수 이름을 인식한다.

예제 31 *Unix*의 `egrep` 명령어는 다음과 같은 형태로 사용된다.

```
egrep 'exp' file
```

여기에서 `exp`는 정규식이고 `file`은 파일의 이름이다. *Unix*에서 정규식 $r + s, rs, r^*, r^+, r + \epsilon$는 각각 `r|s`, `rs`, `r*`, `r+`, `r?`로 표시된다. 이 명령어는 `file`의 줄들을 차례로 읽으면서 각 줄에 `exp`에 속한

스트링이 포함되어 있으면 그 줄을 출력한다. 예를 들면

```
egrep '(aa|bb)+' file
```

에서 `file`이 `ababab`, `aaaa`, `ccbbaacc` 세 줄로 구성되어 있으면 이 명령어는 `aaaa`와 `ccbbaacc`를 출력한다. 이 경우 `exp`에 속한 스트링 w가 각 줄의 부분스트링이 될 수 있으므로, w 앞에 어떤 것이든지 붙을 수 있도록 정규식 Σ^*`exp`를 고려하여야 한다. `egrep`은 Σ^*`exp`를 *NFA*로 바꾼 후 각 줄을 읽다가 최종상태에 도달하면 그 줄을 출력한다.

제 8 절 유한 오토마타의 최소화

DFA를 이용하여 논리회로를 설계할 때, 가능한 한 DFA의 상태 개수를 적게 하는 것이 필요하다. 본 절에서는 정규언어 L이 주어졌을 때 L을 받아들이는, 상태의 개수가 최소인 DFA를 찾는 방법을 기술한다.

먼저 L을 받아들이는 임의의 DFA $M = (Q, \Sigma, \delta, q_0, F)$를 만든다. 물론 초기상태에서 도달할 수 없는 상태들은 쉽게 제거할 수 있다. 따라서 M의 모든 상태들은 초기상태에서 도달할 수 있다고 가정한다.

예제 32 *그림 16의 DFA를 고려하자.*

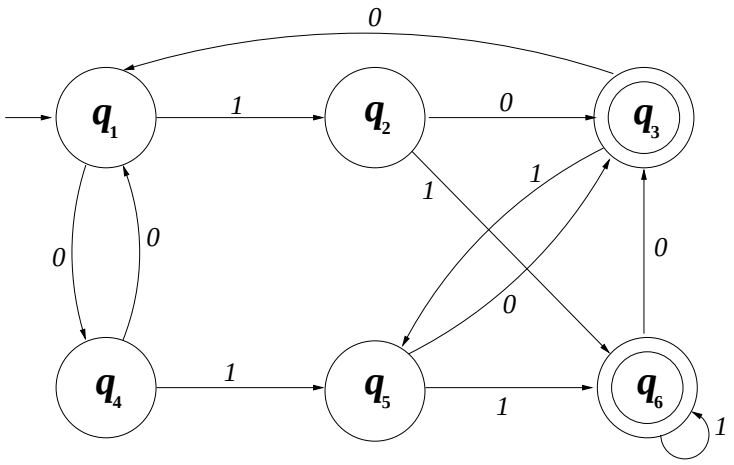

그림 16:

이제 M의 어떤 상태들을 하나의 상태로 묶을 수 있는지 찾고자 한다. 이를 위해 M의 상태들 $p, q \in Q$에 대하여 다음의 관계를 정의한다.

모든 스트링 $x \in \Sigma^*$에 대하여 $\delta^*(p,x) \in F$인 경우에만 $\delta^*(q,x) \in F$일 때 $p \sim q$라고 정의한다.

즉 p와 q 이후에 어떤 스트링이 오든지 결국 두 경우 다 M이 받아들이든지 두 경우 다 M이 받아들이지 않기 때문에 p와 q는 한 상태로 묶을 수 있다는 것이다. 그림 16에서 $q_2 \sim q_5$임을 쉽게 확인할 수 있다.

먼저 $\sim$은 정의에 의해서 반사적이고 대칭적이고 이행적이므로 동치관계이다. 따라서 M의 상태들이 $\sim$에 의해 동치류들로 분할되고 한 동치류에 속한 상태들을 한 상태로 묶으려고 한다.

$\sim$의 동치류들을 단계적으로 구하기 위해 $\sim_n$을 정의한다. 길이가 n 이하인 모든 스트링 x에 대하여 $\delta^*(p,x) \in F$인 경우에만 $\delta^*(q,x) \in F$일 때 $p \sim_n q$이다. 이제 $\sim_0$의 동치류들, $\sim_1$의 동치류들을 차례로 구하다가 $\sim_n$의 동치류들과 $\sim_{n+1}$의 동치류들이 동일하면 이것이 ($\sim$과 $\sim_n$의 정의에 의해) $\sim$의 동치류들이 된다.

$\sim_0$의 동치류는 F와 $Q-F$임을 쉽게 알 수 있다. $\sim_n$의 동치류들이 주어졌을 때 $\sim_{n+1}$의 동치류들은 다음 식에 의해 구할 수 있다.

$p \sim_n q$이고 모든 $a \in \Sigma$에 대하여 $\delta(p,a) \sim_n \delta(q,a)$일 경우에만 $p \sim_{n+1} q$이다.

$\sim$의 동치류들이 구해지면, $M = (Q, \Sigma, \delta, q_0, F)$으로부터 새로운 DFA $M = (Q', \Sigma, \delta', q_0', F')$을 만들 수 있다.

- Q'은 $\sim$의 동치류들의 집합이다.
- M에서 $\delta(p,a) = q$이면 M'에서 $\delta'([p],a) = [q]$이다.
- 초기상태 q'은 $[q_0]$이다.
- $q \in F$이면 $[q] \in F$이다.

예제 33 *그림 16의 DFA에 대하여 $\sim$의 동치류를 단계적으로 구하면 다음과 같다.*

$$\begin{aligned} \sim_0 &: \{q_1, q_2, q_4, q_5\}, \{q_3, q_6\} \\ \sim_1 &: \{q_1, q_4\}, \{q_2, q_5\}, \{q_3\}, \{q_6\} \end{aligned}$$

$$\sim_2 \quad : \quad \{q_1, q_4\}, \{q_2, q_5\}, \{q_3\}, \{q_6\}$$

$\sim$의 동치류는 네 개이고, 따라서 네 개의 상태를 가진 DFA를 만들 수 있다. 이렇게 만들어진 DFA가 그림 7의 DFA이다.

정리 6 (Myhill-Nerode) *정규언어 L을 받아들이는 최소 개의 상태를 가진 DFA는 유일하다.*

Myhill-Nerode의 정리를 이용하여 위에서 만들어진 M'이 최소 개의 상태를 가진 DFA임을 증명할 수 있다 [HU]. 그러므로 정규언어 L을 받아들이는 최소 상태 DFA를 구하기 위해서는 L을 받아들이는 임의의 DFA M을 만든 후 M으로부터 $\sim$을 이용하여 최소 개의 상태를 가진 DFA M'을 만들면 된다.