# 다익스트라의 '프로그래밍의 수련(修練)': 여덟 번째, 유클리드 알고리즘의 재고찰(再考察) (Dijkstra's "A Discipline of Programming": The Eighth Lecture, Euclid's Algorithm Revisited)

김도형
성신여자대학교 컴퓨터정보학부
Do-Hyung Kim
School of CSE, Sungshin Women's University
E-mail: dkim@cs.sungshin.ac.kr; URL: http://cs.sungshin.ac.kr/~dkim

## 요약

이번 튜토리얼에서는 두 양의 정수의 최대공약수를 구하는 유클리드 알고리즘을 다시 살펴본다. 이 알고리즘은 이미 첫 번째 튜토리얼인 '수행의 추상화'에서 '판지 기계(cardboard machine)'를 사용하여 고찰한 바 있다. 그러나 이번에는 두 번째부터 일곱 번째 튜토리얼 동안 설명하여 온 다익스트라의 프로그래밍 방법론을 사용하여 이 알고리즘을 다시 살펴보는 것이다.

[1]독자들이 지겨워 할 위험성을 무릅쓰고, 나는 이제 또 다른 하나의 장(章)을 유클리드 알고리즘(Euclid's algorithm)에 할당하려고 한다. 여기까지 오는 동안 독자들 중 일부는 이미 그 알고리즘을 다음

$$
\begin{array}{l}
x, y := X, Y; \\
\textbf{do } x \neq y \rightarrow \textbf{ if } x > y \rightarrow x := x - y \\
\qquad\qquad\qquad\;\; \mid\; y > x \rightarrow y := y - x \\
\qquad\qquad\qquad\;\; \textbf{fi} \\
\textbf{od}; \\
print(x)
\end{array}
$$

과 같은 형태로 작성하였으리라고 기대한다. 여기에서 반복 구조의 가드는 선택 구조가 취소(abortion)로 귀결되지 않도록 보장한다. 또 다른 독자들은 이 알고리즘이 다음

$$
\begin{array}{l}
x, y := X, Y; \\
\textbf{do } x > y \rightarrow x := x - y \\
\;\;\mid\; y > x \rightarrow y := y - x \\
\textbf{od}; \\
print(x)
\end{array}
$$

과 같이 보다 간단하게 작성될 수 있음을 발견했을 것이다.

이제 판지(cardboard) 놀이는 잊도록 하고, 두 양수(陽數) $X$와 $Y$의 최대공약수(greatest common divisor)[2]를 구하기 위한 유클리드 알고리즘을 새로이 만들어 보도록 하자. 이와 같은 문제에 마주쳤을 때, 원론적으로는 항상 두 가지 길이 우리 앞에 펼쳐져 있는 법이다.

1) 이번 튜토리얼은 다익스트라의 원저에서 제7장에 해당하는 부분이다.

2) 다익스트라가 제0장에서 판지 기계를 이용하여 최대공약수 문제를 다룰 때, 제일 처음 문제를 제시할 때를 제외하고는 최대공약수를 나타내기 위해 'GCD'를 사용하였다. 그런데 여기에서 약어(略語)를 사용하지 않고 단어들을 다 적은 것은 다음 단락에 나오는 '해답의 정의'를 따라간다는 방법을 자연스럽게 이끌어내기 위해서이다.

한 가지 길은 요구되는 해답의 정의(定義)를 가능한 한 엄밀하게 따라가 보는 것이다. 생각건대 우리는 $X$의 약수(約數; divisor)들의 표를 만들 수 있을 것이다; 이 표에 있는 항목들의 개수는 단지 유한할 것이고, 그 중에는 1이 가장 작은 것이고 $X$가 가장 큰 항목일 것이다. (만약 $X=1$이라면 최소와 최대인 항목들이 일치할 것이다.) 그런 뒤 우리는 $Y$의 약수들의 표도 비슷하게 만들 수 있을 것이다. 이 두 표로부터 우리는 두 표에 모두 나타나는 숫자들의 표를 만들 수 있을 것이다; 그렇게 되면 이것은 $X$와 $Y$의 *공통(common)* 약수들의 표가 되고, 1이라는 항목은 포함하고 있을 것이므로 비어 있는 표는 확실히 아니다. 따라서 우리는 이 세 번째 표로부터 크기가 가장 큰 항목을 고를 수 있고(이 표는 유한하기도 하므로!), 그것이 *최대(greatest)* 공약수가 될 것이다.

앞에서 개략적으로 보인 것처럼, 때로는 정의를 꼼꼼하게 따라가는 것이 우리가 할 수 있는 최상의 일이다. 그러나 만약 계산할 함수의 성질들을 안다면(혹은 찾아낼 수 있다면) 시도해 볼 다른 대안(代案)이 있다. 우리가 (함수에 관해) 충분히 많은 성질들을 알아서 그것들을 함께 결합하면 그 함수를 결정지을 수도 있으므로, 우리는 이 성질들을 활용함으로써 해답의 구성을 도모할 수도 있다.

예컨대, 우리는 최대공약수의 경우 $-x$의 약수들은 $x$ 자신의 약수들과 같으므로, GCD($x$, $y$)가 음수 인자(因子)들에 대해서도 정의되며 우리가 인자들의 부호를 바꾼다 해도 (GCD($x$, $y$)는) 변하지 않는다는 것을 관찰한다. 이 함수는 인자들 중 하나가 0일 때도 역시 정의된다; 그 인자는 무한히 많은 약수들의 표를 가지나(따라서 우리는 그 표를 만들려고 시도하지 말아야 한다!) 나머지 한 인자($\neq 0$)가 유한한 약수들의 표를 가지므로, 공통 약수들의 표는 여전히 비어 있지 않고 또한 유한하다. 그래서 우리는 GCD($x$, $y$)가 $(x, y) \neq (0, 0)$인 $(x, y)$ 각각에 대하여 정의된다는 결론에 이른다. 거기에 더해, '공통'이라는 개념의 대칭성(對稱性) 때문에 두 숫자의 최대공약수는 두 인자들의 대칭 함수이다. 조금만 더 생각해 보면, 두 인자의 최대공약수는 인자들 중 하나를 (두 인자의) 합(合)이나 차(差)로 대치한다 하더라도 변하지 않는다는 사실을 우리는 확인할 수 있다. (여기까지) 우리가 안 것을 정리하면, 다음과 같이 적을 수 있다:

$(x, y) \neq (0, 0)$인 $(x, y)$에 대해,

(a) GCD($x$, $y$) = GCD($y$, $x$).
(b) GCD($x$, $y$) = GCD($-x$, $y$).
(c) GCD($x$, $y$) = GCD($x + y$, $y$)
= GCD($x - y$, $y$), 등등.
(d) GCD($x$, $y$) = $abs(x)$, 만약 $x = y$이면.

논증(論證)을 위해, 위의 네 가지 성질들이 우리가 GCD-함수에 대해 알고 있는 것의 전부라고 가정하자. 이것들로 충분할까? 여러분이 보다시피, 앞에서 세 가지 관계들은 $x$와 $y$의 최대공약수를 다른 쌍의 최대공약수로 표현하고 있으나, 마지막 것은 $x$로 직접 표현하고 있다. 그리고 이러한 점은 우선

$$P = (\mathrm{GCD}(X, Y) = \mathrm{GCD}(x, y))$$

의 참을 확립하는 알고리즘을 강력하게 시사(示唆)하고 있으며(이 조건은 배정문 "$x, y := X, Y$"에 의해 손쉽게 달성될 수 있다), 여기에서부터 우리는 $P$ 관계가 변하지 않도록 값의 쌍 $(x, y)$를 (a)와 (b) 또는 (c)에 따라 '주무른다(massage)'. 만약 우리가 $x = y$를 만족하는 상태에 이르도록 이 조작 과정을 해낼 수 있다면, $x$의 절대치(絶對値)를 취함으로써 해답을 찾은 것이다.

우리의 궁극적 목표는 $P$의 불변(不變) 하에서 $x = y$의 참을 확립하는 것이므로, 단조 감소 함수로서 $t = abs(x - y)$를 시도해 볼 수 있겠다.

다시 $P$가 불변(invariant) 조건이라고 하자. 즉, 모든 상태에 대해

$$(P \textbf{ and } BB) \Rightarrow \mathrm{wp}(\mathrm{IF}, P) \tag{1}$$

라고 하고, 추가로 $t$가 모든 상태에 대해

$$(P \textbf{ and } BB) \Rightarrow (t > 0) \tag{2}$$

이 성립하는 유한(有限) 정수 함수라고 하고, 여기에 모든 상태에서 임의의 값 $t0$에 대해

$$(P \textbf{ and } BB \textbf{ and } t \le t0+1) \Rightarrow \mathrm{wp}(\mathrm{IF}, t \le t0) \tag{3}$$

이라고 하자. 그렇다면 우리는 모든 상태에 대해

$$P \Rightarrow \mathrm{wp}(\mathrm{DO},\ T) \tag{4}$$

임을 증명할 것이며, 반복을 위한 기본 정리와 더불어 이것으로부터 우리는 모든 상태에 대해

$$P \Rightarrow \mathrm{wp}(\mathrm{DO},\ P \textbf{ and non } BB) \tag{5}$$

라는 결론을 내릴 수 있다.[3)]

우리는 이것을 수학적 귀납법을 통해 모든 상태에서

$$(P \textbf{ and } t \le k) \Rightarrow H_k(T) \tag{6}$$

가 $k \ge 0$인 모든 $k$에 대해 성립함을 증명함으로써 보인다. 우리는 먼저 $k = 0$에 대해 (6)의 참을 확인한다. $H_0(T) = \textbf{non } BB$이므로, 우리는 모든 상태에 대해

$$(P \textbf{ and } t \le 0) \Rightarrow \textbf{non } BB \tag{7}$$

임을 증명하여야 한다. 그런데 (7)은 다름 아닌 (2)와 같은 식이다; 두 식 모두

$$\textbf{non } P \textbf{ or } \textbf{non } BB \textbf{ or } (t > 0)$$

과 같고,[4)] 따라서 (6)은 $k = 0$에 대해 성립한다.

이제 $k = K$에 대해 (6)이 성립한다고 가정한다; 그러면

$$\begin{aligned}(P \textbf{ and } BB \textbf{ and } t \le K+1) &\Rightarrow \mathrm{wp}(\mathrm{IF},\ P \textbf{ and } t \le K) \\ &\Rightarrow \mathrm{wp}(\mathrm{IF},\ H_K(T));\end{aligned}$$ [5)]

$$\begin{aligned}(P \textbf{ and non } BB \textbf{ and } t \le K+1) &\Rightarrow \textbf{non } BB \\ &= H_0(T)\end{aligned}$$ [6)]

가 성립한다. 그리고 이 두 개의 함축(含蓄; implication)들은 다음:

$$\begin{aligned}(P \textbf{ and } t \le K+1) \Rightarrow \mathrm{wp}(\mathrm{IF},\ H_K(T)) \textbf{ or} \\ H_0(T) = H_{K+1}(T)\end{aligned}$$

과 같이 결합될 수 있고($A \Rightarrow C$와 $B \Rightarrow D$로부터 우리는 ($A$ **or** $B$) $\Rightarrow$ ($C$ **or** $D$)가 성립한다고 결론지을 수 있다), 따라서 (6)의 참은 $k \ge 0$인 모든 $k$에 대해 성립함이 확인되었다. $t$는 유한 함수이므로,

$$(\mathbf{E}\ k\text{: } k \ge 0\text{: } t \le k)$$ [7)]

와

$$\begin{aligned}P &\Rightarrow (\mathbf{E}\ k\text{: } k \ge 0\text{: } P \textbf{ and } t \le k) \\ &\Rightarrow (\mathbf{E}\ k\text{: } k \ge 0\text{: } H_k(T)) \\ &= \mathrm{wp}(\mathrm{DO},\ T)\end{aligned}$$ [8)]

가 성립하며, 따라서 (4)가 증명되었다.

직관적으로 이 정리는 매우 명백하다. 한편으로는 $P$가 참으로 유지될 것이므로 $t \ge 0$도 역시 참으로 유지되며[9)]; 다른 한편으로는 관계 (3)은 가드 명령을 매번 선택할 때마다 $t$가 적어도 1은 실질적으로 감소하게 만들 것이라는 점을 기술하고 있다. 가드 명령이 선택되는 횟수가 무제한적이라면 $t$를

---

3) 지난번 강의에서 다룬 '반복을 위한 기본 정리'는 모든 상태에서 (1) 식이 성립한다면

$$(P \textbf{ and } \mathrm{wp}(\mathrm{DO},\ T)) \Rightarrow \mathrm{wp}(\mathrm{DO},\ P \textbf{ and non } BB)$$

가 성립한다는 것이다. 따라서 (4) 식이 성립한다면 위의 함축에서 전제는 $P$만으로 줄어든다.

4) 잘 알다시피 함축 ($P \Rightarrow Q$)는 (**non** $P$ **or** $Q$)와 동치이다. 또한 (**non** ($P$ **and** $Q$))는 (**non** $P$ **or non** $Q$)와 동치이다.

5) 이 식에서 첫 번째 함축은 (1) 식과 (3) 식, 그리고 술어 변환자의 성질 3에 의해 성립한다. 즉, (1) 식과 $t0$ 자리에 $K$를 넣은 (3) 식을 결합하면

$$\begin{aligned}(P \textbf{ and } BB \textbf{ and } t \le K+1) \\ \Rightarrow \mathrm{wp}(\mathrm{IF},\ P) \textbf{ and } \mathrm{wp}(\mathrm{IF},\ t \le K)\end{aligned}$$

가 성립하며, 여기에 술어 변환자의 세 번째 성질을 적용하면 된다. 두 번째 함축은 (6) 식의 귀납 가정(induction hypothesis)과 술어 변환자의 성질 2에 의해 성립한다. 참고로 술어 변환자의 성질은 다익스트라의 원저 제3장에서 다루어졌다.

6) 이 식의 첫 번째 함축은 자명하다는 것을 노파심에 환기시킨다. (($P$ **and** $Q$) $\Rightarrow$ $P$)는 항상 성립하기 때문이다.

7) $t$가 유한 함수이므로 이 식은 당연히 성립한다.

8) 이 식의 첫 번째 함축은 $t$가 유한 함수라는 것으로부터 성립한다. $t$가 유한 함수이므로 $t$의 값보다 작지 않은 정수는 항상 존재한다. 즉, ($\mathbf{E}$ $k$: $k \ge 0$: $t \le k$) $= T$이다. 따라서 ($P \Rightarrow$ ($P$ **and** ($\mathbf{E}$ $k$: $k \ge 0$: $t \le k$)))가 성립한다. 두 번째 함축은 (6) 식에 의한 것이며, 마지막 동치는 DO 구조의 의미 정의에 따른 것이다(다익스트라의 원저 제4장에 IF와 DO 구조의 의미 정의가 나온다).

9) 식 (1)과 (2)에 의해서 그러하다.

제한 없이 감소시킬 것이며, 이것은 모순(矛盾)으로 이끌 것이다[10].

이 정리의 적용가능 여부는 (2)와 (3)의 성립 여부에 달려 있다. 관계 (2)는 다소 뻔한 것이나,[11] 관계 (3)은 약간 까다로운 면이 있다. 선택 구조를 위한 우리의 기본 정리에서

$$Q = (P \textbf{ and } BB \textbf{ and } t \le t0+1)$$
$$R = (t \le t0)$$

으로 하면—두 술어 모두에 자유 변수 $t0$이 나타난다는 것이 우리가 '술어 쌍'에 대해 언급해 온 이유이다—(그 정리는) 만약 다음

$$(\textbf{A } j{:}\, 1 \le j \le n{:}\, (P \textbf{ and } B_j \textbf{ and } t \le t0+1) \Rightarrow \mathrm{wp}(SL_j,\, t \le t0))$$

이 성립한다면 (3)도 성립한다는 결론을 우리가 내릴 수 있다고 말해 준다.[12] 달리 말하자면, 우리는 각 가드 명령에 대해서 그것을 선택하면 $t$의 실질적인 감소를 일으킬 것이라는 점을 증명해야 한다. 우리는 $t$가 현재 상태의 함수라는 것을 염두에 두고 다음 (술어)

$$\mathrm{wp}(SL_j,\, t \le t0) \tag{8}$$

을 고찰해 볼 수 있다. 이것은 상태 공간의 좌표 변수들 이외에도 자유 변수 $t0$ 또한 포함하고 있는 술어이다. 여태까지 우리는 이러한 술어를 상태들의 부분집합을 규정짓는 것으로서 간주해 왔다. 그러나 어떤 상태가 하나 주어졌을 때, 우리는 이 술어를 $t0$에 부여된 조건으로 취급할 수도 있다.[13] $t0 = tmin$을 식 (8)의 $t0$을 위한 최소의 해라고 하자; 그렇게 되면 우리는 값 $tmin$을 $t$의 최종값을 위한 최저상한(最低上限; lowest upper bound)으로서 해석할 수 있다. $t$ 자신과 꼭 마찬가지로 $tmin$ 역시 현재 상태의 함수라는 점을 상기하면, 술어

$$tmin \le t-1$$

은 $SL_j$가 실행되면 $t$의 값이 적어도 1은 감소된다는 것을 보장하는 최약 사전 조건으로 해석될 수 있다. 여기서—반복하자면—두 번째 인자 $t$는 정수값을 가지는 현재 상태의 함수인 이 사전 조건을

$$\mathrm{wdec}(SL_j,\, t)$$

로 나타내면, $P$의 불변과 $t$의 실질적인 감소는 모든 $j$에 대해

$$(P \textbf{ and } B_j) \Rightarrow (\mathrm{wp}(SL_j,\, P) \textbf{ and } \mathrm{wdec}(SL_j,\, t)) \tag{9}$$

가 성립하면 보장된다.

적당한 $B_j$를 찾기 위한 대개의 실제 방법은 다음과 같다. 식 (9)는

$$(P \textbf{ and } Q) \Rightarrow R$$

---

10) 식 (2)를 만족시키지 못 하기 때문이다. 어떤 정리가 자기 자신의 전제에 위배되므로 자가당착(自家撞着)이고 모순이다. 그 결과로 도출되는 결론은 DO 구조의 반복 횟수가 무제한적이 되지 않고, 언젠가는 '적절히 종료한다'는 것이다.

11) 관계 (2)를 만족하는 유한 정수 함수 $t$를 찾는 것은 전혀 어렵지 않다. 예컨대 $t = k$ $(k > 0)$와 같은 상수(常數) 함수는 관계 (2)를 만족시킨다.

12) 선택 구조를 위한 기본 정리에 따르면, 술어 쌍 $Q$와 $R$이 모든 상태에 대해

$$Q \Rightarrow BB$$

와

$$(\textbf{A } j{:}\ 1 \le j \le n{:}\ (Q \textbf{ and } B_j) \Rightarrow \mathrm{wp}(SL_j,\, R))$$

을 만족하면,

$$Q \Rightarrow \mathrm{wp}(\mathrm{IF},\, R)$$

역시 모든 상태에 대해 성립한다는 것이다. 여기서 $Q$와 $R$을 본문에서처럼 정의하면 선택 구조를 위한 기본 정리의 첫 번째 조건

$$(P \textbf{ and } BB \textbf{ and } t \le t0+1) \Rightarrow BB$$

는 당연히 만족되므로,

$$(\textbf{A } j{:}\ 1 \le j \le n{:}\ (P \textbf{ and } B_j \textbf{ and } t \le t0+1) \Rightarrow \mathrm{wp}(SL_j,\, t \le t0))$$

이 만족되면

$$(P \textbf{ and } BB \textbf{ and } t \le t0+1) \Rightarrow \mathrm{wp}(\mathrm{IF},\, t \le t0),$$

즉 (3) 식이 만족된다.

13) 지금까지처럼 술어를 변수 $t$에 부여된 조건이 아니라, 이제 $t$가 하나의 상태에서 값이 고정된 경우이므로 $t0$에 부여된 조건으로 보자는 뜻이다.

과 같은 형태이고, 여기서 주어진 $P$와 $R$에 대한 $Q$—실제 계산가능한(computable)!—를 하나 찾아야만 하는 것이다. 우리는 다음과

1. $Q = R$은 하나의 해(解)다.
2. 만약 $Q = (Q1$ **and** $Q2)$가 하나의 해고 $P \Rightarrow Q2$라면, $Q1$ 역시 해다.
3. 만약 $Q = (Q1$ **or** $Q2)$가 하나의 해고 $P \Rightarrow$ **non** $Q2$라면(또는 결국 같은 것이 되지만 ($P$ **and** $Q2) = F$), $Q1$ 역시 해다.
4. 만약 $Q$가 하나의 해고 $Q1 \Rightarrow Q$라면, $Q1$ 역시 해다.

같은 점들을 살펴본다.

주의 1. 이렇게 하는 과정에서[14] 만약 우리가 $P \Rightarrow$ **non** $Q$와 같은 $B_j$를 위한 $Q$의 후보에 이르렀다면, 이 후보 $Q$는 $Q = F$(앞의 단계 (3)에 따르면, 모든 $Q$에 대해 $Q = (F$ **or** $Q)$이므로)로 보다 더 간단해질 수 있다; 이것은 현재 고려중인 가드 명령이 쓸모가 없으며, 결코 선택되지 않을 것이기 때문에 (가드 명령) 집합에서 빼버릴 수 있음을 의미한다. (주의 1의 끝.)

주의 2. 종종 (9) 식을 두 개의 식

$$(P \text{ and } B_j) \Rightarrow \text{wp}(SL_j, P) \tag{9a}$$

와

$$(P \text{ and } B_j) \Rightarrow \text{wdec}(SL_j, t) \tag{9b}$$

로 분리하여 별도로 다루는 것이 실용적이다. 따라서 두 개의 고려사항이 분리된 셈이다: (9a)는 불변으로 유지되는 것과 관련이 있고, 반면에 (9b)는 진행(進行)을 보장하는 것과 관련이 있다. 만약 식 (9a)를 다루는 도중에 $P \Rightarrow B_j$와 같은 $B_j$에 이르렀다면, 그러한 $B_j$를 가지고는 $P$의 불변성이 비종결(nontermination)을 보장할 것이기 때문에 이 조건[15]은 (9b)를 만족하지 않을 것이라는 점이 확실하다.[16] (주의 2의 끝.)

14) 바로 앞 부분에서 이야기하고 있는 (9) 식을 만족하는 가드들 $B_j$를 구하는 과정을 뜻한다.

15) $B_j$를 뜻한다.

그러므로 우리는

$$P \Rightarrow \text{wp}(\text{DO}, P \textbf{ and non } BB)$$

와 같은 DO 메커니즘을 만들 수 있다. 우리가 선택한 $B_j$들은 함축 (9)를 만족시킬 수 있도록 충분히 강해야만 하며,[17] 그 결과로 이제 보장되는 사후 조건 $P$ **and non** $BB$는 너무 약해져서 요구하는 사후 조건 $R$을 함축하지 못 할 지도 모른다.[18] 그런 경우, 우리는 아직 문제를 해결하지 못한 것이며 다른 가능성들을 고려해야 한다.

## 참고 문헌

[1] Bergin, T. J. and R. G. Gibson (Eds.), *History of Programming Languages*, Addison Wesley, New York, 1996.

16) 만약 어떤 가드 $B_j$가 불변 조건 $P$가 성립하는 동안 항상 참을 유지한다면, 즉 ($P \Rightarrow B_j$)가 참이라면, 이러한 불변 조건과 가드를 가지는 반복 구조는 결코 끝나지 않는다. 참인 가드가 항상 하나 이상 존재하기 때문이다. 이런 경우 그러한 불변 조건과 가드는 당연히 (9b) 식을 만족하지 않을 것이다.

17) 함축이 성립하기 위해서는 전제를 이루는 조건이 강하면 강할수록 용이하다. 가장 강한—어떻게 해도 만족시킬 수 없을 정도로 강한—조건인 $F$가 전제이면, 결론을 이루는 조건에 관계없이 함축은 참이 되는 것이다. (9) 식의 전제를 이루는 $B_j$가 강하면 강할수록 그 함축은 참이 되기가 쉽다.

18) 반복 구조가 종료할 때, 우리는 불변 조건 $P$에 더해서 참인 가드가 하나도 없다는 조건, 즉 **non** $BB$를 부가적으로 만족하게 된다. 이 조건, 곧 ($P$ **and non** $BB$)가 우리가 원하는 사후 조건 $R$을 함축하면 우리가 원하는 바가 달성되는 것이다. 그런데

$$\begin{aligned} BB &= (\textbf{E } j: 1 \le j \le n: B_j) \\ &= (B_1 \textbf{ or } B_2 \textbf{ or } \cdots \textbf{ or } B_n) \end{aligned}$$

이므로,

$$\textbf{non } BB = (\textbf{non } B_1 \textbf{ and non } B_2 \textbf{ and } \cdots \textbf{ and non } B_n)$$

이 되고, 가드 $B_j$가 강해지면 강해질수록 **non** $BB$는 약해진다. 따라서 함축

$$(P \textbf{ and non } BB) \Rightarrow R$$

이 성립되기 더 어려워지는 것이다.

[2] Dahl, O.-J. E. W. Dijkstra, and C. A. R. Hoare, *Structured Programming*, Academic Press, New York, 1972.
[3] Dijkstra, E. W., "Guarded Commands, Nondeterminacy, and Formal Derivation of Programs," *Communications of the ACM*, Vol. 18, No. 8, pp. 453-457, 1975.
[4] Dijkstra, E. W., *A Discipline of Programming*, Prentice Hall, Englewood Cliffs, NJ, 1976.
[5] Ghezzi, C. and M. Jazayeri, *Programming Language Concepts*, 2nd Ed., Wiley, New York, 1987.
[6] Hoare, C. A. R., "An Axiomatic Basis of Computer Programming," *Communications of the ACM*, Vol. 12, No. 10, pp. 576-580, 1969.
[7] Hoare, C. A. R., and N. Wirth, "An Axiomatic Definition of the Programming Language Pascal," *Acta Informatica*, Vol. 2, pp. 335-355, 1973.
[8] Hoare, C. A. R., "The Emperor's Old Clothes," *Communications of the ACM*, Vol. 24, No. 2, pp. 75-83, 1981.
[9] Kafura, D., *Object-Oriented Software Design and Construction with C++*, Prentice Hall, Upper Saddle River, NJ, 1998.
[10] Louden, K. C., *Programming Languages: Principles and Practice*, PWS Publishing Company, 1993.
[11] Sammet, J. E., "Programming Languages: History and Future," *Communications of the ACM*, Vol. 15, No. 7, pp. 601-610, 1972.
[12] Sebesta, R. W., *Concepts of Programming Languages*, 4th Ed., Addison Wesley Longman, 1999.
[13] Wexelblat, R. L. (Ed.), *History of Programming Languages*, Academic Press, New York, 1981.

**김도형**

1981년 ~ 1985년:
서울대학교 공과대학 컴퓨터공학과(학사)
1985년 ~ 1987년:
한국과학기술원 전산학과(석사)
1987년 ~ 1992년:
한국과학기술원 전산학과(박사)
1992년:
한국과학기술원 정보전자연구소 연수연구원
1992년 ~ 현재:
성신여자대학교 컴퓨터정보학부 부교수
1997년 ~ 1998년:
스토니 브룩 소재 뉴욕주립대학교 컴퓨터과학과 객원교수

관심분야:
프로그래밍 언어