

# “고델의 불완전성 정리들” 해설

## 임기정

이 글에는 논리학자 Raymond M. Smullyan 교수님의 위대한 저서 “Gödel’s Incompleteness Theorems”의 번역과 더불어 제 사견도 들어 있습니다.

### 1 Gödel의 증명 뒤의 일반적인 아이디어

다음 몇 개의 단원에서 다양한 [산술의 공리화]에 대한 불완전성 증명들을 공부할 것이다. Gödel은 1931년 공리적 집합론에 대한 그의 원래 증명을 이끌어냈지만, 그 방법은 공리적 수론에도 (동등하게) 적용될 수 있다. 공리적 수론에 대한 불완전성은, 공리적 집합론의 불완전성을 쉽게 도출해 낼 수 있으므로, 실제로 더 강한 결과이다.

Gödel은 그의 개념비적인 논문을 다음의 놀라운 낱말들로 시작한다.

“더 정확한 방향으로의 [수학의 발전]은 그것의 광범위한 분야가 형식화되도록 이끌었고, 그리하여 증명들은 몇 개 되지 않는 기계적인 규칙들을 따라 이끌어 낼 수 있다. 오늘날 가장 포괄적인 형식 중 하나는 Whitehead와 Russel의 Principia Mathematica이고, 다른 하나는 공리적 집합론의 Zermelo-Fraenkel 체계이다. 두 체계 모두 너무 확장적이어서, 오늘날 사용되는 모든 증명 기법들이 그것들 안에서 형식화 될 수 있다 – 즉, 몇 가지 안 되는 공리들과 추론 규칙들로 축약될 수 있다. 그러므로, 이 공리들과 추론 규칙들이 관심 있는 체계 안에서 형식화될 수 있는 모든 수학적 질문들을 결정하기에<sup>1</sup> 충분하다고 받아들이는 게 합리적으로 보인다. 후술할 내용에서 이는 사실이 아니라는 것과, 오히려, 언급된 두 체계 모두에서 상대적으로 쉬운 정수론의 문제들 중에 이 공리들을 기저로 하여 결정될 수 없는 문제가 존재한다는 것을 보일 것이다.”

Gödel은 그리고는 이러한 상황이 고려 중인 두 체계의 특별한 본성에 의존하지 않고, 어떤 수학 체계들의 대규모 집단에서 성립함을 설명하기 시작한다. 이 수학 체계들의 “대규모 집단”이란 과연 무엇일까? 이 어구에 대하여 다양한 해석들이 있어왔고, Gödel의 그 정리는 몇 가지

---

<sup>1</sup>역주: 여기서, “결정”의 뜻은 증명 또는 반증을 하여 참인지 거짓인지 결정한다는 뜻이다.

방법으로 일반화될 수 있는데, 이 책에서 많은 그러한 일반화들을 고려할 것이다. 그 일반화들 중 일반적인 독자에게 가장 직접적이고 가장 쉽게 접근 가능한 하나가 가장 덜 알려져 있는 것으로 보이는데, 이는 매우 희한하다. 이를 더욱 희한하게 만드는 것은, 문제의 그 방법이 Gödel 그 자신이 그의 원래 논문의 서론에서 시사한 바로 그것이라는 것이다! 나는 곧 이것으로 (혹은 더 일반화된 것으로) 방향을 틀 것이다. 그러나 그 전에 나는 독자가, Gödel의 아이디어의 본질을 단순하고 교육적인 방법으로 설명해 주는, 다음 퍼즐을 보길 바란다.

**A Gödelian Puzzle.** 다음의 다섯 가지 기호로 이루어진 다양한 표현식을 출력해 내는 어떤 계산하는 기계를 고려하자:

$$\sim \quad P \quad N \quad ( \quad )$$

표현식(expression)이란 위의 다섯 가지 기호들의 문자열 중 길이가 0은 아니지만 유한한 것을 의미한다. 그 기계가 표현식  $X$ 를 출력해 낼 수 있다면,  $X$ 를 출력 가능(printable)하다고 한다. 그 기계는 그것이 출력해 낼 수 있는 표현식이라면 무엇이든지 언젠가는 출력해 내도록 설계되어 있다고 가정하자.

표현식  $X$ 의 노름(norm)이란, 표현식  $X$  ( $X$ )를 의미한다 – 예를 들어,  $P \sim$ 의 노름은  $P \sim (P \sim)$ 이다. 문장(sentence)이란, 다음 네 가지 꼴 중 하나의 형태를 띤 모든 표현식을 의미한다 – 여기서,  $X$ 에는 임의의 표현식이 올 수 있다:

$$(1) P(X)$$

$$(2) PN(X)$$

$$(3) \sim P(X)$$

$$(4) \sim PN(X)$$

비형식적으로,  $P$ 는 “printable”을;  $N$ 은 “the norm of”를;  $\sim$ 은 “not”을 의미한다. 그런즉, 표현식  $X$ 에 대하여,  $X$ 가 출력 가능할 때 (그리고 그럴 때에만)  $P(X)$ 를 참(true)이라고 정의하고;  $X$ 의 노름이 출력 가능할 때 (그리고 그럴 때에만)  $PN(X)$ 를 참(true)이라고 정의하고;  $X$ 가 출력 가능하지 않을 때 (그리고 그럴 때에만)  $P(X)$ 를 참(true)이라고 정의하고;  $X$ 의 노름이 출력 가능하지 않을 때 (그리고 그럴 때에만)  $PN(X)$ 를 참(true)이라고 정의한다.<sup>2</sup>

이로써 무슨 문장이 참인지에 대한 완벽한 정의를 갖추었는데, 그것은 자기 참조의 흥미로운 경우이다: 그 기계는 자신이 출력할 수 있는 문장은 무엇이고 출력할 수 없는 문장은 무엇인지에 대한 여러 가지의 문장들을 출력하기 때문에, 자신의 행동을 기술하는 셈이다!<sup>3</sup>

---

<sup>2</sup>마지막 문장은 “Not printable the norm of  $X$ ”라고 읽거나, 혹은 더 좋은 영어로 “The norm of  $X$  is printable”라고 읽는다.

<sup>3</sup>이것은 자의식 있는 생물과 다소 닮았다. 또한 그러한 컴퓨들이 왜 인공지능 분야에서 일하는 사람들에게 관심이 있는지를 알 수 있다.

그 기계가 완벽하게 작동하여 그 기계에 의하여 출력되는 모든 문장들이 참이라는 사실이 주어졌다. 그러므로, 예를 들어, 그 기계가  $P(X)$ 를 출력할 때마다  $X$ 가 정말로 출력 가능하다 –  $X$ 는 그 전 혹은 그 후에 출력될 것이다. 마찬가지로, 그 기계가  $P N(X)$ 를 출력할 때마다  $X$ 의 노름인  $X(X)$ 도 출력 가능하다.  $X$ 가 출력 가능하다고 가정하자. 그렇다면  $P(X)$ 가 출력 가능해질까? 그럴 필요는 없다.  $X$ 가 출력 가능하다면,  $P(X)$ 는 참이지만, 그 기계가 거짓인 문장을 절대로 출력하지 않을 뿐이지, 모든 참인 문장을 출력할 수 있다는 사실은 주어지지 않았기 때문이다.<sup>4</sup>

그렇다면, 그 기계가 모든 참인 문장을 출력할 수 있을까? 대답은 아니오이다. 이제 독자를 위한 수수께기를 내겠다: 그 기계가 출력할 수 없지만 참인 문장을 하나 찾아라.<sup>5</sup>

**A Variant of the Puzzle.** 위의 수수께기의 변종인 다음 문제는 독자들에게 괴델 넘버링(*Gödel numbering*)의 개념을 소개할 것이다.

이제 다음 다섯 가지의 기호들로 구성된 문자열들을 출력해 내는 또 다른 기계가 주어졌다고 하자:

~       $P$        $N$       1      0

우리는 자연수를 이진 표기법으로 (1과 0의 문자열로서) 자연수를 나타낼 것이고, 이 문제의 목적에 맞추어, 그 둘을 나타내는 이진 숫자들을 비교함으로써 두 자연수를 식별할 것이다.

각 표현마다 괴델 수(*Gödel number*)라고 불리는 수를 할당할 것이다. 그렇게 하기 위하여, 다음 규칙을 따르자: 단일 기호 ~,  $P$ ,  $N$ , 1, 0의 괴델 수는 각각 10, 100, 1000, 10000, 100000이다. 그리고, 복합식의 괴델 수는 각 기호들을 그것의 괴델 수로 교체하여 얻는다 – 예를 들어  $P N P$ 의 괴델 수는 1001000100이다. 표현식  $X$ 의 노름(*norm*)을  $X$  뒤에  $X$ 의 괴델 수를 붙인 표현식으로 재정의한다. 이제 문장(*sentence*)이란 다음 네 가지 꼴 중 하나의 형태를 띤 표현식이다:  $P X$ ,  $P N X$ ,  $\sim P X$  그리고  $\sim P N X$  – 여기서  $X$ 에는 임의의 이진 숫자가 올 수 있다. 이진 숫자  $X$ 에 대하여, 어떤 표현식의 괴델 수가  $X$ 이고 그 표현식이 출력 가능할 때 그리고 그럴 때에만,  $P(X)$ 를 참이라고 한다. 이진 숫자  $X$ 에 대하여, 어떤 표현식의 괴델 수가  $X$ 이고 그 표현식의 노름이 출력 가능할 때 그리고 그럴 때에만,  $P N(X)$ 를 참이라고 한다. 이진 숫자  $X$ 에 대하여,  $P(X)$ 가 참이 아닐 때 그리고 그럴 때에만,  $\sim P(X)$ 를 참이라고 한다. 이진 숫자  $X$ 에 대하여,  $P N(X)$ 가 참이 아닐 때 그리고 그럴 때에만,  $\sim P N(X)$ 를 참이라고 한다.

전과 마찬가지로 그 기계는 거짓인 문장을 절대로 출력하지 않는다고 한다. 참이지만 그 기계가 출력할 수 없는 문장을 하나 찾아라.

**Solutions.** 첫 번째 문제의 답은  $\sim P N(\sim P N)$ 이다. “참”의 정의에 의하여, 이 문장이 참일 때 그리고 그럴 때에만  $\sim P N$ 의 노름이 출력 가능하지 않다. 그런데  $\sim P N$ 의 노름이 바로 문장

<sup>4</sup> 그 기계가 문장이 아닌 표현식을 출력하는지는 전혀 중요하지 않다. 중요한 것은 그 기계에 의하여 출력되는 문장들은 모두 참이라는 사실이다.

<sup>5</sup> 도움말: 자신의 출력 불가능성을 선언하는 문장을 찾아라 – 즉, 그 문장은 자신이 그 기계에 의하여 출력될 수 없을 때 그리고 그럴 때에만 참이다. 해답은 다음 문제 뒤에 줄 것이다.

$\sim PN$  ( $\sim PN$ )이다. 그런즉, 그 문장이 참일 때 그리고 그럴 때에만 그것은 출력 가능하지 않다. 이는 그 문장이 참이고 출력 가능하지 않는 경우와 거짓이고 출력 가능한 경우 밖에 없음을 의미한다. 그러나 후자는 참이지 않은 문장은 절대로 출력하지 않는다는 가정을 위반한다. 그러므로 그 문장은 참이지만 출력 가능하지 않다.

당연히, 다섯 가지의 기호들로 구성된 다양한 표현식들을 출력하는 기계를 논하는 대신, 같은 기호들로 구성된 다양한 문장들을 증명하는 수학 체계를 논할 수도 있다. 우리는 문자  $P$ 를, 그 기계에 의하여 출력 가능하다는 뜻 대신, 그 체계에서 증명 가능(provable)하다는 뜻으로 재해석할 수 있다. 그러면, 만약 그 체계가 완전히 정확하다고 할 때 (즉, 거짓인 문장은 절대 증명 가능하지 않을 때), 문장  $\sim PN$  ( $\sim PN$ )는 참이지만 그 체계에서 증명 가능하지 않는 문장이 될 것이다.

추가로, 문장  $PN$  ( $\sim PN$ )이 거짓임을 알 수 있다 – 왜냐하면 그것의 부정이 참이기 때문이다. 그러므로 (그 체계가 정확하다고 가정할 때) 그것 또한 그 체계에서 증명 가능하지 않아야 한다. 그러므로 문장  $PN$  ( $\sim PN$ )는 결정 불가능(undecidable)한 문장의 예이다 – 즉, 자신도 자신의 부정도 그 체계에서 증명 가능하지 않다.

두 번째 문제의 답은  $\sim PN101001000$ 이다. ¬

이제 일반적인 설정에서의 몇 가지 불완전성 논증들로 방향을 틀자: 특정 성질이 있는 수학 체계에는 반드시 괴델의 논증이 통한다는 것을 보일 것이다. 뒷 단원들에서는 특정 체계들을 살펴보고 그들이 정말로 그러한 성질을 가지고 있음을 보일 것이다.

## 1.1 Gödel과 Tarski의 정리의 추상적인 형태

괴델의 논증을 적용할 수 있는 각각의 체계  $\mathcal{L}$ 은 적어도 다음과 같은 물건들을 가지고 있다.

1. 그것의 원소가  $\mathcal{L}$ 의 표현식(expression)이라고 불리는 가부번 집합  $\mathcal{E}$ .
2. 그것의 원소가 문장(sentence)이라고 불리는 집합  $\mathcal{S} \subseteq \mathcal{E}$ .
3. 그것의 원소가 증명 가능(provable) 문장이라고 불리는 집합  $\mathcal{P} \subseteq \mathcal{S}$ .
4. 그것의 원소가 반증 가능(refutable) 문장이라고 불리는 집합  $\mathcal{R} \subseteq \mathcal{S}$ .
5. 그것의 원소가  $\mathcal{L}$ 의 술어(predicate)라고 불리는 집합  $\mathcal{H} \subseteq \mathcal{E}$ .<sup>6</sup>
6.  $(\forall H \in \mathcal{H}) (\forall n \in \mathbb{N}) [\Phi(H, n) \in \mathcal{S}]$ <sup>7</sup>를 만족시키는 함수  $\Phi : \mathcal{E} \times \mathbb{N} \rightarrow \mathcal{E}$ .<sup>8</sup>

---

<sup>6</sup>괴델의 서론에서 술어는 *class name*이라고 불린다. 비형식적으로, 각 술어는 자연수들의 집합의 이름으로 생각할 수 있다.

<sup>7</sup>비형식적으로, 문장  $H(n)$ 은 자연수  $n$ 이  $H$ 라는 이름을 가진 집합에 속한다는 명제를 표현한다.

<sup>8</sup>표기법의 남용을 허용하여, 각  $E \in \mathcal{E}$ 와 각  $n \in \mathbb{N}$ 에 대하여 표현식  $\Phi(E, n)$ 을 “ $E(n)$ ”으로 표기한다.

7. 그것의 원소가 참(true)인 문장이라고 불리는 집합  $\mathcal{T} \subseteq \mathcal{S}$ .

특정 체계  $\mathcal{L}$ 에 대한 제 1 불완전성을 증명할 때, Alfred Tarski[1936]에 의하여 정확하게 된 참의 개념을 사용할 것이다.

이로써, 뒤의 여러 단원에서 공부할 체계들의 유형에 대한 추상적인 묘사를 마무리 짓겠다.

**Expressibility in  $\mathcal{L}$ .** 우리가 곧 정의할  $\mathcal{L}$ 에서의 표현 가능성(expressibility)의 개념은 진리 집합  $\mathcal{T}$ 를 다루지만  $\mathcal{P}$ 나  $\mathcal{R}$ 은 다루지 않는다.

이 책의 나머지 부분에서 “수”라는 단어는 자연수(natural number)를 의미한다. 술어  $H$ 와 수  $n$ 에 대하여,  $\Phi(H, n)$ 이 참인 문장일 때 – 즉,  $H(n) \in \mathcal{T}$ 일 때 –  $H$ 가  $n$ 에 대하여 참(true)이라고 하거나,  $n$ 이  $H$ 를 만족시킨다고 한다. 술어  $H$ 에 의하여 표현된(expressed) 집합이란,  $H$ 를 만족시키는 모든 수들의 집합을 의미한다. 그러므로, 임의의  $H \in \mathcal{H}$ 와 임의의  $A \subseteq \mathbb{N}$ 에 대하여,  $H$ 가  $A$ 를 표현할 때 그리고 그럴 때에만

$$(\forall n \in \mathbb{N}) [H(n) \in \mathcal{T} \leftrightarrow n \in A]$$

가 성립한다.  $\dashv$

**Definition.**  $\mathcal{L}$ 에서 집합  $A$ 를 표현하는 술어가 존재할 때,  $A$ 를  $\mathcal{L}$ 에서 표현될 수 있다(expressible)고 하거나,  $A$ 가  $\mathcal{L}$ 에서 이름을 가진다(nameable)고 한다.  $\dashv$

$\mathcal{E}$ 는 가부번 집합이므로,  $\mathcal{H}$ 는 유한 집합이거나 가부번 집합이다. 그러나 Cantor의 잘 알려진 정리에 의하여, 모든  $\mathbb{N}$ 의 부분 집합들의 집합은 비가산 집합이다. 그러므로, 모든 집합들이  $\mathcal{L}$ 에서 표현될 수 있는 것은 아니다.

**Definition.** 체계  $\mathcal{L}$ 의 모든 증명 가능한 문장들이 참이고 모든 반증 가능한 문장들이 거짓이면,  $\mathcal{L}$ 이 정확(correct)하다고 한다. 이것은  $\mathcal{P}$ 이  $\mathcal{T}$ 의 부분 집합이고  $\mathcal{R}$ 이  $\mathcal{T}$ 와 서로소 집합이라는 것을 의미한다. 이제 충분한 조건들 안에서  $\mathcal{L}$ 이 정확하다면 참이지만 증명 가능하지 않은 문장을 포함한다는 데 집중하자.  $\dashv$

**Gödel Numbering and Diagonalization.**  $\mathfrak{g}$ 를 각  $E \in \mathcal{E}$ 에게 고델 수(Gödel number)라고 불리는 자연수  $\mathfrak{g}(E)$ 를 부여하는 단사 함수라고 하자. 이 단원의 나머지 부분에서  $\mathfrak{g}$ 는 고정될 것이다.<sup>9</sup>  $\mathfrak{g}$ 가 전사함수라고 가정하는 것이 기술적으로 편리할 것이다.<sup>10</sup> 이제 각각의 수  $n$ 마다  $n$ 을 고델 수로 하는 표현식이 유일하다고 가정하고, 그 유일한 표현식을  $E_n$ 이라 하겠다. 즉, 임의의  $n \in \mathbb{N}$ 에 대하여  $\mathfrak{g}(E_n) = n$ 이다.

$E_n$ 의 대각화란, 표현식  $E_n(n)$ 을 의미한다.  $E_n$ 이 술어라면, 그것의 대각화는 당연히 문장이다; 이 문장은,  $E_n$ 이 그것의 고델 수  $n$ 에 의하여 만족할 때 그리고 그럴 때에만, 참이다.

---

<sup>9</sup> 뒷 단원들에서 공부할 구체적인 체계에서는 특정 고델 넘버링이 주어질 것이다. 지금 다루는 순수하게 추상적인 논의에서는 임의의 고델 넘버링에 대하여 적용될 수 있다.

<sup>10</sup> 고델의 원래 넘버링은 이러한 성질이 없지만, 뒷 단원들에서 사용할 고델 넘버링은 이러한 성질을 가진다. 그러나, 이 단원의 결과들은, 소소한 수정을 거치면, 이러한 제한 없이도 증명될 수 있다. (cf. Ex. 5).

각각의  $n \in \mathbb{N}$ 마다,  $d(n)$ 을  $E_n(n)$ 의 괴델 수로 둔다. 함수  $d : \mathbb{N} \rightarrow \mathbb{N}$ 은 뒤의 내용에서 항상 핵심적인 역할을 할 것이다;  $d$ 를 그 체계의 대각 함수(diagonal function)라고 부른다.

각각의  $A \subseteq \mathbb{N}$ 마다,  $A^* := \{n \in \mathbb{N} : d(n) \in A\}$ 로 놓자. 즉,  $A^*$ 의 정의에 의하여 동등성

$$(\forall n \in \mathbb{N}) [n \in A^* \leftrightarrow g(E_n(n)) \in A]$$

이 성립한다.<sup>11</sup>

¬

**An Abstract Form of Gödel's Theorem.**  $P$ 를 모든 증명 가능한 문장들의 괴델 수를 모은 집합이라고 하자. 그리고, 각  $A \subseteq \mathbb{N}$ 마다  $\tilde{A}$ 를  $\mathbb{N} \setminus A$ 로 두자 – 즉,  $\tilde{A}$ 는  $A$ 의 여집합이다.

¬

**Theorem (GT)–After Gödel with shades of Tarski.** 집합  $\tilde{P}^*$ 가  $\mathcal{L}$ 에서 표현될 수 있고  $\mathcal{L}$ 이 정확하다면,  $\mathcal{L}$ 에서 참이지만 증명 가능하지 않은 문장이 있다.

¬

*Proof.*  $\mathcal{L}$ 이 정확하고  $\tilde{P}^*$ 가  $\mathcal{L}$ 에서 표현될 수 있다고 가정하자. 그러면  $\mathcal{L}$ 에서  $\tilde{P}^*$ 를 표현하는 술어  $H$ 가 존재한다. 이때  $h$ 를  $H$ 의 괴델 수라고 하고,  $G$ 를  $H$ 의 대각화(즉, 문장  $H(h)$ )라고 하자. 우리는  $G$ 가 참이지만 증명 가능하지 않음을 보일 것이다.  $H$ 가  $\mathcal{L}$ 에서  $\tilde{P}^*$ 를 표현하므로,

$$H(n) \in \mathcal{T} \iff n \in \tilde{P}^* \quad \text{for } \forall n \in \mathbb{N}$$

이다. 이 동등성이 임의의  $n$ 에 대하여 성립하므로,  $n$ 이  $h$ 일 때도 성립한다. 그러므로  $n$ 에  $h$ 를 대입하여 – 이 부분의 논증은 대각화하기(diagonalizing)라고 불린다 –

$$H(h) \in \mathcal{T} \iff h \in \tilde{P}^*$$

를 얻는다. 한편,

$$h \in \tilde{P}^* \iff d(h) \in \tilde{P} \iff d(h) \notin P$$

인데,  $H$ 의 괴델 수가  $h$ 이기 때문에  $d(h)$ 의 괴델 수가  $H(h)$ 이므로,  $H(h) \in \mathcal{T} \iff H(h) \notin P$ 이다. 그러므로,

1.  $H(h)$ 가 참일 때 그리고 그럴 때에만  $H(h)$ 가 증명 가능하지 않다. 이것은  $H(h)$ 가 참이고 증명 가능하지 않은 경우와 거짓이고 증명 가능한 경우 밖에 없음을 의미한다. 그런데 후자는  $\mathcal{L}$ 이 정확하다는 가정을 위반한다. 따라서  $H(h)$ 는 참이지만 증명 가능하지 않아야 한다.

라는 결론을 얻는다. □

---

<sup>11</sup>  $A^*$ 은 대각 함수  $d$  아래의  $A$ 의 역상이기 때문에  $d^{-1}[A]$ 로도 쓰일 수 있다.

우리가 공부할 특정 체계  $\mathcal{L}$ 에 대하여,  $\tilde{P}^*$ 가  $\mathcal{L}$ 에서 표현 될 수 있다는 전제 조건이 성립함을, 다음 세 조건을 따로따로 검사함으로써, 입증할 것이다:

$G_1$ : 임의의  $A \subseteq \mathbb{N}$ 에 대하여,  $A$ 가  $\mathcal{L}$ 에서 표현될 수 있으면  $A^*$ 도  $\mathcal{L}$ 에서 표현될 수 있다.

$G_2$ : 임의의  $A \subseteq \mathbb{N}$ 에 대하여,  $A$ 가  $\mathcal{L}$ 에서 표현될 수 있으면  $\tilde{A}$ 도  $\mathcal{L}$ 에서 표현될 수 있다.

$G_3$ :  $\mathcal{P}$ 가  $\mathcal{L}$ 에서 표현될 수 있다.

조건  $G_1$ 과  $G_2$ 가 성립하면,  $\mathcal{L}$ 에서 표현될 수 있는 임의의  $A \subseteq \mathbb{N}$ 에 대하여  $A^*$ 도  $\mathcal{L}$ 에서 표현될 수 있다. 그러므로  $P$ 가  $\mathcal{L}$ 에서 표현될 수 있으면,  $\tilde{P}^*$ 도  $\mathcal{L}$ 에서 표현될 수 있다.

우리는  $G_1$ 의 검증은 상대적으로 단순한 것으로 밝혀지고;  $G_2$ 의 검증은 완전히 사소하지만;  $G_3$ 의 증명은 극도로 정교하다고 밝혀지는 데 주목할 수 있다.

**Gödel Sentences.** Rudolf Carnap [1934]에 의하여 명시화된 매우 중요한 원리가 정리 GT의 증명에 얹혀있는데, 이것은 곧 다를 Tarski의 정리와도 밀접한 관련이 있다.

$A \subseteq \mathbb{N}$ 와  $X \in \mathcal{S}$ 에 대하여,

$$X \in \mathcal{T} \leftrightarrow g(X) \in A$$

가 성립하면,  $X$ 를  $A$ 에 대한 괴델 문장이라고 부른다.<sup>12</sup>

¬

다음 보조정리는  $\mathcal{T}$ 와만 관련이 있다. 집합  $\mathcal{P}$ 와 집합  $\mathcal{R}$ 과는 무관하다.

**Lemma (D)–A Diagonal Lemma.**

- (a) 임의의  $A \subseteq \mathbb{N}$ 에 대하여,  $A^*$ 가  $\mathcal{L}$ 에서 표현될 수 있으면  $A$ 에 대한 괴델 문장이 존재한다. 더 자세히는, 술어  $H$ 가  $A^*$ 를  $\mathcal{L}$ 에서 표현할 때,  $H$ 의 대각화  $H(h)$ 는  $A$ 에 대한 괴델 문장이다  
– 여기서  $h$ 는  $H$ 의 괴델 수이다.
- (b)  $\mathcal{L}$ 이 조건  $G_1$ 을 만족시키면,  $\mathcal{L}$ 에서 표현될 수 있는 임의의  $A \subseteq \mathbb{N}$ 에 대하여  $A$ 에 대한 괴델 문장이 존재한다.

¬

*Proof.* (a)  $d(h)$ 는  $H(h)$ 의 괴델 수이다. 술어  $H$ 가  $A^*$ 를  $\mathcal{L}$ 에서 표현하므로,

$$H(n) \iff n \in A^* \quad \text{for } \forall n \in \mathbb{N}$$

이다. 따라서,  $H(h)$ 가 참일 때 그리고 그럴 때에만  $h \in A^*$ 이다. 그런데  $d(h) \in A \leftrightarrow h^* \in A$ 가 성립하므로,  $H(h)$ 가 참일 때 그리고 그럴 때에만  $d(h) \in A$ 이다. 그런데  $g(H(h)) = d(h)$ 이므로,  $H(h)$ 가  $A$ 에 대한 괴델 문장임을 알 수 있다.

<sup>12</sup>비형식적으로,  $A$ 에 대한 괴델 문장은 그 자신의 괴델 수가  $A$ 에 속한다고 선언하는 문장으로 볼 수 있다. 그 문장이 참이면 그것의 괴델 수가  $A$ 에 속하고, 그 문장이 거짓이면 그것의 괴델 수가  $A$ 에 속하지 않는다.

(b) (a)에 의하여 즉각적으로 따라온다.

□

만일 보조정리 D를 먼저 증명했다면, 다음과 같은 신속한 증명을 얻었을 것이다:  $\tilde{P}^*$ 가 이름을 가진다면, D의 (a)에 의하여,  $\tilde{P}$ 에 대한 괴델 문장  $G$ 가 존재한다.  $\tilde{P}$ 에 대한 괴델 문장은 자신이 참일 때 그리고 그럴 때에만 증명 가능하지 않은 문장 그 이상도 그 이하도 아니다. 그러므로, 임의의 정확한 체계  $\mathcal{L}$ 에서,  $\tilde{P}$ 에 대한 괴델 문장은  $\mathcal{L}$ 에서 참이지만 증명 가능하지 않은 문장이다.<sup>13</sup>

**An Abstract Form of Tarski's Theorem.** 보조정리 D는 또 다른 중요한 결과를 낳는다:  $T$ 를 모든 참인 문장들의 괴델 수를 모은 집합이라고 하자. 그러면 다음 정리가 성립한다.

**Theorem (T) (After Tarski).**

1. 집합  $\tilde{T}^*$ 는  $\mathcal{L}$ 에서 이름을 가지지 않는다.
2. 조건  $G_1$ 이 성립하면,  $\tilde{T}$ 는  $\mathcal{L}$ 에서 이름을 가지지 않는다.
3. 조건  $G_1$ 과  $G_2$ 가 성립하면,  $T$ 는  $\mathcal{L}$ 에서 이름을 가지지 않는다.

□

*Proof.* 시작하기에 앞서,  $\tilde{T}$ 에 대한 괴델 문장은 없다는 것에 주목하자 – 왜냐하면, 그러한 문장은 자신의 괴델 수가 참인 문장의 괴델 수가 아닐 때 그리고 그럴 때에만 참인 문장이기 때문이다.

1.  $\tilde{T}^*$ 가  $\mathcal{L}$ 에서 이름을 가진다면, 보조정리 D의 (a)에 의하여,  $\tilde{T}$ 에 대한 괴델 문장이 존재하는 데, 이는 방금 보인 바에 따르면 불가능하다. 그러므로  $\tilde{T}^*$ 가  $\mathcal{L}$ 에서 이름을 가지지 않는다.
2. 조건  $G_1$ 이 성립한다고 가정하자. 그러면,  $\tilde{T}$ 가  $\mathcal{L}$ 에서 이름을 가질 때,  $\tilde{T}^*$ 도  $\mathcal{L}$ 에서 이름을 가지는데, 이는 (1)을 위반한다.
3. 조건  $G_2$  역시 성립한다면,  $T$ 가  $\mathcal{L}$ 에서 이름을 가질 때,  $\tilde{T}$ 도  $\mathcal{L}$ 에서 이름을 가지는데, 이는 (2)를 위반한다.

□

### Remarks.

1. 위의 결론 (3)은 종종 다음과 같이 비유된다: 충분히 강력한 체계에서는, 그 체계 안에서의 진리는 그 체계 안에서 정의될 수 없다. 어구 “충분히 강력한”은 몇 가지 방법으로 해석될 수 있다.  $G_1$ 과  $G_2$ 가 “충분히 강력한”에 부합되기에 충분하다는 것을 짚고 넘어간다.

---

<sup>13</sup>그러한 문장은  $\mathcal{L}$ 에서의 자신의 증명 불가능성을 선언하는 문장으로 생각할 수 있다.

2. Gödel(1931)은 [크레타인들은 모두 거짓말쟁이라고 말하는 어떤 크레타인]의 유명한 역설에 빗댄다.<sup>14</sup> 괴델의 비유에 조금 더 근접한 비유는 이것이다: 항상 참말만을 말하는 주민들과 항상 거짓말만을 말하는 주민들 밖에 없는 나라를 상상하여라. 주민들 중 일부는 아테네인이고 또 다른 일부는 크레타인이다. 그 나라의 모든 아테네인들은 참말만을 말하고 그 나라의 모든 크레타인들은 거짓말만을 말한다고 한다. 한 주민이 자신은 항상 참말만을 말하지만 아테네인이 아니라는 것을 너에게 설득시키기 위하여 그 주민이 무슨 말을 하면 될까?

그는 “나는 아테네인이 아니다”라고만 말하면 된다. 거짓말쟁이는 그러한 주장을 만들 수 없기 때문이다 – 왜냐하면, 거짓말쟁이는 실제로 아테네인이 아니기 때문이다; 아테네인들은 참말쟁이 밖에 없기 때문이다. 그러므로 그의 말을 믿을 수 있다. 그가 말한 문장이 참이었기 때문에 그는 정말로 아테네인이 아니다. 그러므로 그는 참말쟁이이지만 아테네인이 아니다.

아테네인들이  $\mathcal{L}$ 에서 참이면서 증명 가능한 문장들의 역할을 하는 것으로 간주하면, 자신이 아테네인이 아니라고 주장하는 주민은  $\tilde{P}$ 에 대한 괴델 문장  $G$ 의 역할을 하는 셈인데, 이는  $\mathcal{L}$ 에서의 자신의 증명 불가능성을 선언하기 때문이다.<sup>15</sup>

–

## 1.2 $\mathcal{L}$ 의 결정 불가능한 문제들

지금까지 모든 반증 가능한 문장들의 집합  $\mathcal{R}$ 은 아무 역할도 하지 않았다. 이제 그것은 중요한 역할을 할 것이다.

$\mathcal{L}$ 의 문장 중 증명 가능한 동시에 반증 가능한 문장이 없을 때  $\mathcal{L}$ 이 일관성 있다(*consistent*)고 하고 – 즉,  $\mathcal{P}$ 와  $\mathcal{R}$ 이 서로소임을 의미한다 – 그렇지 않을 경우 일관성 없다(*inconsistent*)고 한다. 이 정의는 집합  $\mathcal{P}$ 와  $\mathcal{R}$ 을 언급하지만  $\mathcal{T}$ 는 언급하지 않는다. 그럼에도 불구하고,  $\mathcal{L}$ 이 정확하면, 자동적으로 일관성 있게 된다 – 왜냐하면  $\mathcal{P}$ 가  $\mathcal{T}$ 의 부분 집합이고  $\mathcal{R}$ 이  $\mathcal{T}$ 와 서로소 집합이면  $\mathcal{P}$ 는  $\mathcal{R}$ 과 서로소 집합이어야 하기 때문이다. 그 역이 참일 필요는 없다 – 우리는 후에 일관성 있지만 정확하지 않은 체계들을 다룰 것이다.

문장  $X$ 가  $\mathcal{L}$ 에서 증명 가능하거나 반증 가능할 경우  $X$ 를 결정 가능(*decidable*)하다고 하고, 그렇지 않으면 결정 불가능(*undecidable*)하다고 한다. 체계  $\mathcal{L}$ 의 모든 문장이 결정 가능하면 완전(*complete*)하다고 하고 그렇지 않으면 불완전(*incomplete*)하다고 한다.

<sup>14</sup>실제로는, 거짓말쟁이의 역설은 Gödel의 정리보다는 Tarski의 정리에 더 가깝다.

<sup>15</sup>크레타인들은 당연히  $\mathcal{L}$ 의 반증 가능한 문장들의 역할을 하는데, 그것의 기능이 잠시 뒤에 나타날 것이다.

이제  $\mathcal{L}$ 이 정리 GT의 전제 조건들을 만족시킨다고 가정하자. 그러면 어떤 문장  $G$ 가  $\mathcal{L}$ 에서 참이지만 증명 가능하지 않다.  $G$ 는 참이므로 반증 역시 가능하지 않다 – 정확성을 가정하였기 때문이다. 그러므로  $G$ 는  $\mathcal{L}$ 에서 결정 불가능하다. 그리고 즉시 다음 정리를 얻는다.

**Theorem 1.**  $\mathcal{L}$ 이 정확하고  $\mathcal{L}$ 에서  $\tilde{P}^*$ 가 표현될 수 있으면,  $\mathcal{L}$ 은 불완전하다.  $\dashv$

**A Dual of Theorem 1.** T.F.S.(Theory of Formal Systems, 1961)에서 나는 Gödel의 논증의 “쌍대 형식”이라고 부르는 것이 적절한 무언가를 도입했는데, 이를 먼저 비형식적으로 설명하겠다. “나는 증명 가능하지 않아”라고 말하는 문장을 구성하는 대신에, “나는 반증 가능해”라고 말하는 문장을 구성하겠다. 그러한 문장 역시 ( $\mathcal{L}$ 이 정확하다면) 결정 불가능해야 함을 곧 보게 될 것이다.

집합  $P$ 를 모든 증명 가능한 문장들의 괴델 수로 정의해왔다. 이제 집합  $R$ 을 모든 반증 가능한 문장들의 괴델 수로 정의하자.  $\dashv$

**Theorem (1°)–(A Dual of Theorem 1).**  $\mathcal{L}$ 이 정확하고  $R^*$ 가  $\mathcal{L}$ 에서 표현될 수 있으면,  $\mathcal{L}$ 은 불완전하다. 더 구체적으로는,  $\mathcal{L}$ 이 정확하고 술어  $K$ 가  $\mathcal{L}$ 에서  $R^*$ 를 표현하면, 그것의 대각화  $K(k)$ 는  $\mathcal{L}$ 에서 결정 불가능하다 – 여기서  $k$ 는  $K$ 의 괴델 수이다.  $\dashv$

*Proof.* 전제 조건들을 가정하자.  $K$ 가  $R^*$ 를 표현하기에, 보조정리 D의 (a)에 의하여, 문장  $K(k)$ 는  $R$ 에 대한 괴델 문장이다. 즉,  $K(k)$ 는 그것의 괴델 수가  $R$ 에 속할 때 그리고 그럴 때에만 참이다. 다시 말해,  $K(k)$ 가 참일 때 그리고 그럴 때에만  $K(k)$ 가 반증 가능하다. 이는  $K(k)$ 가 참이고 반증 가능한 경우와 거짓이고 반증 불가능한 경우 밖에 없음을 의미한다. 정확성을 가정하였으므로,  $K(k)$ 는 참이면서 반증 가능할 수는 없다. 따라서 그것은 거짓이며 반증 가능하지 않다. (또다시  $\mathcal{L}$ 이 정확하다는 가정에 의하여) 그 문장은 거짓이기에 증명 역시 가능하지 않다. 그러므로  $K(k)$ 는  $\mathcal{L}$ 에서 증명도 반증도 가능하지 않다.  $\square$

**Remarks.**  $\tilde{P}$ 에 대한 괴델 문장  $H(h)$ 가 “나는  $\mathcal{L}$ 에서 증명 가능하지 않아”라고 말한다고 생각할 수 있는 것처럼,  $K(k)$ 는 “나는  $\mathcal{L}$ 에서 반증 가능해”라고 말하는 것으로 생각할 수 있다. 아테네인들과 크레타인들에 관한 비유로 돌아가자. 자신이 아테네인이 아니라고 말하는 주민이  $H(h)$ 에 대응하는 것처럼, 문장  $K(k)$ 는 자신이 크레타인이라고 말하는 주민에 대응된다. 그는 거짓 말쟁이지만 크레타인이 아니어야 한다. 따라서 (자신이 아테네인이 아니라고 주장하는 주민의 경우와 같이) 그는 아테네인도 크레타인도 아니어야 한다.

정확한 체계  $\mathcal{L}$ 이 다음 두 조건을 만족시킨다고 가정하자:

$G_1$ : 임의의  $A \subseteq \mathbb{N}$ 에 대하여,  $A$ 가  $\mathcal{L}$ 에서 표현될 수 있으면  $A^*$ 도  $\mathcal{L}$ 에서 표현될 수 있다.

$G'_3$ :  $R$ 을  $\mathcal{L}$ 에서 표현될 수 있다.

그러면 당연히  $R^*$ 은  $\mathcal{L}$ 에서 표현될 수 있다. 그렇기 때문에, 정리 1°에 의하여, 정확하지 않거나 불완전하다. 여집합 조건  $G_2$ 는 이 증명에서 필요하지 않음에 주목하자.  $\dashv$

다음 첫 번째 연습문제는 정리 1°의 흥미로운 변종이다.

**Exercise 1.** 정확한 체계  $\mathcal{L}$ 이 다음 두 조건을 만족시킨다고 가정하자.

1. 집합  $P^*$ 이  $\mathcal{L}$ 에서 표현될 수 있다.
2. 임의의 술어  $H$ 에 대하여, 어떤 술어  $H'$ 이 존재하여

$$(\forall n \in \mathbb{N}) [H'(n) \in \mathcal{P} \leftrightarrow H(n) \in \mathcal{R}]$$

이 성립한다.

이때  $\mathcal{L}$ 이 불완전함을 증명하여라.  $\neg$

**Exercise 2.**  $H \in \mathcal{H}$ 와  $A \subseteq \mathbb{N}$ 가

$$(\forall n \in \mathbb{N}) [H(n) \in \mathcal{P} \leftrightarrow n \in A]$$

를 만족시키면  $H$ 가  $\mathcal{L}$ 에서  $A$ 를 나타낸다(represent)고 한다.<sup>16</sup>

정확한 체계  $\mathcal{L}$ 에서 집합  $R^*$ 를 나타낼 수 있으면,  $\mathcal{L}$ 은 불완전함을 보여라.  $\neg$

**Exercise 3.** 어떤  $R^*$ 의 상위 집합  $A \subseteq \mathbb{N}$ 가 존재하여  $P^*$ 가  $A$ 와 서로소 집합이고  $\mathcal{L}$ 에서  $A$ 를 나타낼 수 있으면,  $\mathcal{L}$ 은 불완전함을 보여라.<sup>17</sup>  $\neg$

**Exercise 4.**  $H \in \mathcal{H}$ 와  $A \subseteq \mathbb{N}$ 가

$$(\forall n \in \mathbb{N}) [H(n) \in \mathcal{R} \leftrightarrow n \in A]$$

를 만족시키면  $H$ 가  $\mathcal{L}$ 에서  $A$ 를 거꾸로 나타낸다(contrarepresent)고 한다. 일관성 있는 체계  $\mathcal{L}$ 에서 집합  $P^*$ 를 거꾸로 나타낼 수 있다면,  $\mathcal{L}$ 은 불완전함을 보여라.<sup>18</sup>  $\neg$

**Exercise 5.** 괴델 넘버링  $\mathbf{g}$ 가 전사 함수가 아닌 경우를 고려하자. 이때

$$(\forall E \in \mathcal{E}) (\forall e \in \mathbb{N}) [\mathbf{g}(E) = e \rightarrow \mathbf{g}(\Phi(E, e)) = d(e)]$$

를 만족시키는 함수  $d : \mathbb{N} \rightarrow \mathbb{N}$ 를 한 대각 함수로 보자. 대각 함수  $d$ 가 주어졌을 때, 임의의  $A \subseteq \mathbb{N}$ 에 대하여  $d^{-1}[A]$ 가  $\mathcal{L}$ 에서 표현될 수 있다면  $A$ 에 대한 괴델 문장이 존재한다는 것을 보여라.  $\neg$

**Exercise 6.** 임의의  $A \subseteq \mathbb{N}$ 에 대하여,  $\tilde{A}^*$ 와  $\widetilde{A}^*$ 가 같은 집합일 필요가 있을까?  $\neg$

**Exercise 7.** Gödel의 증명의 온전히 구성적인 본질을 강조하기 위하여,  $\mathcal{L}$ 이 다음 세 조건을 만족시키는 정확한 체계라고 하자.

<sup>16</sup>이 정의는 진리 집합  $T$ 를 언급하지 않고 오직 증명가능성 집합  $\mathcal{P}$ 만을 언급한다.

<sup>17</sup>집합  $A$ 가 집합  $B$ 의 부분 집합이면,  $B$ 를  $A$ 의 상위 집합(superset)이라고 한다.

<sup>18</sup>이 결과와 연습문제 2의 결과는 5단원에서 확장될 것이다; 연습문제 3의 결과는 6단원에서 배울 Rosser의 불완전성 정리와 관련 있다.

1. 표현식  $E_7$ 는  $P$ 를 표현하는 술어이다.
2. 임의의  $n \in \mathbb{N}$ 와 임의의  $A \subseteq \mathbb{N}$ 에 대하여, 표현식  $E_n$ 이  $A$ 를 표현하는 술어이면 표현식  $E_{3n}$ 은  $\tilde{A}$ 를 표현하는 술어이다.
3. 임의의  $n \in \mathbb{N}$ 와 임의의  $A \subseteq \mathbb{N}$ 에 대하여, 표현식  $E_n$ 이  $A$ 를 표현하는 술어이면 표현식  $E_{3n+1}$ 은  $A^*$ 를 표현하는 술어이다.

이때 다음 세 물음에 답하여라:

- (a) 문장  $E_a(b)$ 가 참이지만 증명 가능하지 않은 문장임을 보장할 수 있는  $a \in \mathbb{N}$ 와  $b \in \mathbb{N}$ 의 순서쌍  $(a, b)$ 를 찾아라.<sup>19</sup>
- (b) 문장  $E_a(b)$ 가 참이지만 증명 가능하지 않은 문장임을 보장할 수 있는  $a \in \mathbb{N}$ 와  $b \in \mathbb{N}$ 의 순서쌍  $(a, b)$ 이 무한히 많음을 보여라.
- (c) 표현식  $E_{10}$ 이 술어일 때,  $E_c(d)$ 가  $E_{10}$ 이 표현하는 집합에 대한 괴델 문장임을 보장할 수 있는  $c \in \mathbb{N}$ 와  $d \in \mathbb{N}$ 의 순서쌍  $(c, d)$ 를 하나 찾아라.

→

### Solutions.

1. 모든 전제 조건들을 가정하자. 먼저, 조건 1에 의하여, 어떤 술어  $H$ 가  $\mathcal{L}$ 에서  $P^*$ 를 표현하는 것을 알 수 있다. – 즉,

$$(\forall n \in \mathbb{N}) [H(n) \in \mathcal{T} \leftrightarrow n \in P^*]$$

이 성립한다. 또한, 조건 2에 의하여, 어떤 술어  $H'$ 이 존재하여

$$(\forall n \in \mathbb{N}) [H'(n) \in \mathcal{P} \leftrightarrow H(n) \in \mathcal{R}]$$

이 성립한다. 이제,  $h' := g(H')$ 에 대하여,  $H(h')$ 이  $\mathcal{L}$ 에서 증명도 반증도 가능하지 않는 것을 보일 것이다. 먼저,  $H(h')$ 이  $\mathcal{L}$ 에서 증명 가능한 경우,

$$\begin{aligned} H(h') \in \mathcal{P} &\implies H(h') \in \mathcal{T} \\ &\implies h' \in P^* \\ &\implies H'(h') \in \mathcal{P} \\ &\implies H(h') \in \mathcal{R} \end{aligned}$$

---

<sup>19</sup> $a$ 와  $b$  모두 100 미만인 해는 두 개 있다. 과연 독자가 둘 다 찾아낼 수 있을까?

인데, 이는  $\mathcal{L}$ 이 정확하다는 가정에 모순이다. 한편,  $H(h')$ 이  $\mathcal{L}$ 에서 반증 가능한 경우,

$$\begin{aligned} H(h') \in \mathcal{R} &\implies H'(h') \in \mathcal{P} \\ &\implies h' \in P^* \\ &\implies H(h') \in \mathcal{T} \end{aligned}$$

인데, 이것 역시  $\mathcal{L}$ 이 정확하다는 가정에 모순이다. 그러므로,  $H(h')$ 이  $\mathcal{L}$ 에서 결정 불가능하기 때문에,  $\mathcal{L}$ 은 불완전하다.

2. 모든 전제 조건들을 가정하자.  $R^*$ 를  $\mathcal{L}$ 에서 나타낼 수 있으므로, 어떤 술어  $K$ 가 존재하여

$$(\forall n \in \mathbb{N}) [K(n) \in \mathcal{P} \leftrightarrow n \in R^*]$$

이 성립한다. 이제,  $k := g(K)$ 에 대하여,  $K(k)$ 가  $\mathcal{L}$ 에서 증명도 반증도 가능하지 않다는 것을 보일 것이다. 먼저,  $K(k)$ 가 증명 가능한 경우,

$$\begin{aligned} K(k) \in \mathcal{P} &\implies k \in R^* \\ &\implies K(k) \in \mathcal{R} \end{aligned}$$

인데, 이는  $\mathcal{L}$ 이 정확하다는 가정에 모순이다. 한편,  $K(k)$ 가 반증 가능한 경우,

$$\begin{aligned} K(k) \in \mathcal{R} &\implies K \in R^* \\ &\implies K(k) \in \mathcal{P} \end{aligned}$$

인데, 이것 역시  $\mathcal{L}$ 이 정확하다는 가정에 모순이다. 그러므로,  $K(k)$ 이  $\mathcal{L}$ 에서 결정 불가능하기 때문에,  $\mathcal{L}$ 은 불완전하다.

3. 전제 조건들을 가정하자.  $A$ 를  $\mathcal{L}$ 에서 나타낼 수 있으므로, 어떤 술어  $K$ 가 존재하여

$$(\forall n \in \mathbb{N}) [K(n) \in \mathcal{P} \leftrightarrow n \in A]$$

이 성립한다. 이제,  $k := g(K)$ 에 대하여,  $K(k)$ 가  $\mathcal{L}$ 에서 증명도 반증도 가능하지 않다는 것을 보일 것이다. 먼저,  $K(k)$ 가 증명 가능하다고 가정하자. 그러면,

$$K(k) \in \mathcal{P} \implies k \in P^*$$

이고

$$K(k) \in \mathcal{P} \implies k \in A$$

인데, 이는  $P^*$ 가  $A$ 와 서로소 집합이라는 가정에 모순이다. 한편,  $K(k)$ 가 반증 가능하다고 가정하자. 그러면,  $R^* \subseteq A$ 임을 이용하여,

$$\begin{aligned} K(k) \in \mathcal{R} &\implies k \in R^* \\ &\implies k \in A \\ &\implies K(k) \in \mathcal{P} \\ &\implies k \in P^* \end{aligned}$$

임과

$$\begin{aligned} K(k) \in \mathcal{R} &\implies k \in R^* \\ &\implies k \in A \end{aligned}$$

임을 알 수 있다. 그러나 이것 역시  $P^*$ 가  $A$ 와 서로소 집합이라는 가정에 모순이다.

4. 전제 조건들을 가정하자.  $P^*$ 를  $\mathcal{L}$ 에서 거꾸로 나타낼 수 있으므로, 어떤 술어  $K$ 가 존재하여

$$(\forall n \in \mathbb{N}) [K(n) \in \mathcal{R} \leftrightarrow n \in P^*]$$

이 성립한다. 이제,  $k := \mathfrak{g}(K)$ 에 대하여, 문장  $K(k)$ 가  $\mathcal{L}$ 에서 결정 불가능함을 보이겠다. 먼저,  $K(k)$ 를  $\mathcal{L}$ 에서 증명 가능한 경우,

$$\begin{aligned} K(k) \in \mathcal{P} &\implies k \in P^* \\ &\implies K(k) \in \mathcal{R} \end{aligned}$$

인데, 이는  $\mathcal{L}$ 이 일관성 있다는 가정에 모순이다. 한편,  $K(k)$ 를  $\mathcal{L}$ 에서 반증 가능한 경우,

$$\begin{aligned} K(k) \in \mathcal{R} &\implies k \in P^* \\ &\implies K(k) \in \mathcal{P} \end{aligned}$$

인데, 이것 역시  $\mathcal{L}$ 이 일관성 있다는 가정에 모순이다. 그러므로,  $K(k)$ 가  $\mathcal{L}$ 에서 결정 불가능하기 때문에,  $\mathcal{L}$ 은 불완전하다.

5. 대각 함수  $d$ 와 집합  $A$ 이 주어졌다고 하고,  $d^{-1}[A]$ 가  $\mathcal{L}$ 에서 표현될 수 있다고 하자. 그러면 어떤 술어  $H$ 가 존재하여

$$(\forall n \in \mathbb{N}) [H(n) \in \mathcal{T} \leftrightarrow n \in d^{-1}[A]]$$

이 성립한다. 이제,  $h := g(H)$ 에 대하여,  $H(h)$ 가  $A$ 에 대한 괴델 문장이라는 것을 보일 것이다. 먼저,  $h = g(H)$ 이므로, 대각 함수의 제약 조건

$$(\forall E \in \mathcal{E}) (\forall e \in \mathbb{N}) [g(E) = e \rightarrow g(E(e)) = d(e)]$$

로부터,  $g(H(h)) = d(h)$ 를 얻는다. 그러면,

$$\begin{aligned} H(h) \in \mathcal{T} &\iff h \in d^{-1}[A] \\ &\iff d(h) \in A \\ &\iff g(H(h)) \in A \end{aligned}$$

의 동등성을 얻는데, 이는  $H(h)$ 가  $A$ 에 대한 괴델 문장 중 하나라는 것을 의미한다.

6.  $A \subseteq \mathbb{N}$ 이 주어졌다고 하자. 그러면, 임의의  $n \in \mathbb{N}$ 에 대하여

$$\begin{aligned} n \in \tilde{A}^* &\iff n \in d^{-1}[\tilde{A}] \\ &\iff d(n) \in \tilde{A} \\ &\iff d(n) \notin A \\ &\iff n \notin d^{-1}[A] \\ &\iff n \notin A^* \\ &\iff n \in \widetilde{A}^* \end{aligned}$$

이므로,  $\tilde{A}^* = \widetilde{A}^*$ 이어야 한다.

7. (a) 먼저,  $(a, b) = (64, 64)$ 가 한 해임을 보이겠다.  $64 = 3 \times (3 \times (7)) + 1$ 으로,  $E_{64}$ 는 집합  $\tilde{P}^*$ 을 표현하는 술어이다. 정리 GT에 의하여,  $E_{64}$ 는  $\mathcal{L}$ 에서 참이지만 증명 가능하지 않다. 그리고,  $(a, b) = (66, 66)$ 이 또 다른 해임을 보이겠다.  $66 = 3 \times (3 \times (7) + 1)$  이므로,  $E_{66}$ 은 집합  $\widetilde{P}^*$ 을 표현하는 술어이다 – 즉,

$$E_{66}(n) \in \mathcal{T} \iff n \in \widetilde{P}^* \quad \text{for } \forall n \in \mathbb{N}$$

이다. 따라서,

$$\begin{aligned}
 E_{66}(66) \in \mathcal{T} &\iff 66 \in \tilde{P}^* \\
 &\iff 66 \notin P^* \\
 &\iff d(66) \notin P^* \\
 &\iff g(E_{66}(66)) \notin P \\
 &\iff E_{66}(66) \notin \mathcal{P}
 \end{aligned}$$

인데,  $\mathcal{L}$ 의 정확성을 가정하였으므로, 문장  $E_{66}(66)$ 은  $\mathcal{L}$ 에서 참이지만 증명 가능하지 않아야 한다.

(b)  $n \geq 1$ 에 대하여,  $a_n := 7 \times 9^n + 1$ 라고 하자. 이제, 임의의  $n \geq 1$ 에 대하여,  $E_{a_n}(b_n)$ 이  $\mathcal{L}$ 에서 참이지만 증명 가능하지 않다는 것을 보이겠다.  $n \geq 1$ 을 임의로 잡자. 그러면 술어  $E_{a_n}$ 은 집합  $\tilde{P}^*$ 를 표현한다. 그러므로,

$$\begin{aligned}
 E_{a_n}(b_n) \in \mathcal{T} &\iff b_n \in \tilde{P}^* \\
 &\iff d(b_n) \in \tilde{P} \\
 &\iff g(E_{a_n}(b_n)) \in \tilde{P} \\
 &\iff g(E_{a_n}(b_n)) \notin P \\
 &\iff E_{a_n}(b_n) \notin \mathcal{P}
 \end{aligned}$$

인데,  $\mathcal{L}$ 의 정확성에 의하여,  $E_{a_n}(b_n)$ 은  $\mathcal{L}$ 에서 참이지만 증명 가능하지 않아야 한다. 그런데  $n \mapsto (a_n, b_n) : \mathbb{N}_+ \rightarrow \mathbb{N} \times \mathbb{N}$ 은 정의역이  $\mathbb{N}_+$ 인 단사 함수이므로,  $E_a(b)$ 가  $\mathcal{L}$ 에서 참이지만 증명 가능하지 않다는 것을 보장할 수 있는,  $a \in \mathbb{N}$ 와  $b \in \mathbb{N}$ 의 순서쌍  $(a, b)$ 은 무한히 많음을 알 수 있다.

(c) 집합  $A \subseteq \mathbb{N}$ 가 술어  $E_{10}$ 에 의하여 표현된다고 하자. 그러면

$$(\forall n \in \mathbb{N}) [E_{10}(n) \in \mathcal{T} \leftrightarrow n \in A]$$

이 성립한다. 이제,  $E_{31}(31)$ 이  $A$ 에 대한 괴델 문장임을 보이겠다. 먼저,  $31 = 3 \times (10) + 1$ 이므로, 술어  $E_{31}$ 는 집합  $A^*$ 를 표현한다 – 즉,

$$E_{31}(n) \in \mathcal{T} \iff n \in A^* \quad \text{for } \forall n \in \mathbb{N}$$

이다. 따라서,

$$\begin{aligned} E_{31}(31) \in \mathcal{T} &\iff 31 \in A^* \\ &\iff d(31) \in A \\ &\iff g(E_{31}(31)) \in A \end{aligned}$$

인데, 이는  $E_{31}(31)$ 이  $A$ 에 대한 괴델 문장이라는 것을 의미한다. 그러므로

$$(c, d) = (31, 31)$$

이 해 중 하나이다.

¬

### Comments.

- 정리 GT의 증명에서, 문장  $G$ 가 참일 때 그리고 그럴 때에만 증명 가능하지 않다는 것으로부터  $G$ 는 참이지만 증명 가능하지 않음을 보일 때, 배증률을 이용하여 증명했지만, 직관주의 논리에서도 증명할 수 있다 – 즉, 명제  $H_1$ 과 명제  $H_2$ 를 각각

$$H_1 : \equiv (G \in \mathcal{T}) \leftrightarrow \neg(G \in \mathcal{P})$$

와

$$H_2 : \equiv (\forall X \in \mathcal{S}) [X \in \mathcal{P} \rightarrow X \in \mathcal{T}]$$

로 두었을 때, 직관주의 체계에서  $H_1, H_2 \vdash G \in \mathcal{T} \wedge G \notin \mathcal{P}$ 를 도출할 수 있다.

*Proof.* 먼저,  $H_1$ 과  $H_2$ 를 가정하자. 추가로, 명제

$$H_3 : \equiv G \in \mathcal{P}$$

가 성립한다고 가정하자. 그러면  $H_2$ 에  $H_3$ 를 적용하여  $G \in \mathcal{T}$ 를 얻는데, 이것을  $H_3$ 와 함께  $H_1$ 의 순방향에 적용하면 모순을 얻는다. 이제 가정  $H_3$ 을 해제하고,  $\neg H_3$ 를 얻는다. 이것을  $H_1$ 의 역방향에 적용하면,  $G \in \mathcal{T}$ 를 얻는다. 그런데  $\neg H_3 \equiv G \notin \mathcal{P}$ 이므로, 원하던  $[G \in \mathcal{T} \wedge G \notin \mathcal{P}]$ 를 얻는다. □

¬

## 2 산술에 대한 Tarski의 정리

전 단원에서, 상당히 일반적인 방법으로 수학적 언어를 다루었다. 이제는 특정 수학적 언어를 다루겠다. 페아노 산술(Peano Arithmetic)으로 알려진 체계에 대한 Gödel의 불완전성 정리에 도달하는 것이 우리의 목표 중 하나인데, 이 중요한 결과에 대한 몇 가지 증명을 볼 것이다; 그 중 가장 간단한 것은, 곧 다룰, Tarski의 정리를 부분적인 기반으로 삼는다.

### 2.1 언어 $\mathcal{L}_E$

#### §1. 구문론적 예비 사항

우리가 공부할 첫 번째 구체적인 언어는 덧셈과 곱셈 그리고 제곱을 기반으로 하는 1차 산술의 언어이다.<sup>20</sup> (편리한 괴델 넘버링을 구현하기 위하여) 오직 유한한 알파벳만을 사용하여 그 언어를 형식화할 것이다; 구체적으로는 다음 13개의 기호들을 사용할 것이다.

$$0 \quad ' \quad ( \quad ) \quad f \quad \circ \quad v \quad \neg \quad \rightarrow \quad \forall \quad = \quad \leq \quad \#$$

표현식  $0, 0', 0'', 0''', \dots$ 은 각각 자연수  $0, 1, 2, 3, \dots$ 의 형식적인 이름으로서 숫자(numeral)라고 불린다. 악센트 기호는 프라임(prime)이라고도 불리는데, 다음 수 함수의 형식적인 이름이다. 또한 덧셈과 곱셈 그리고 제곱 연산의 이름이 필요한데;  $f_0, f_{00}, f_{000}$ 을 각각 이 세 함수의 이름이라 하자. 그러나  $f_0$ 을 “+”로,  $f_{00}$ 을 “×”로,  $f_{000}$ 을 “E”로 줄여서 표기하자.

기호  $\neg$ 과  $\rightarrow$ 는 명제 논리의 친숙한 기호로서, 각각 부정과 실질 함축을 의미한다.<sup>21</sup> 기호  $\forall$ 는 전칭 양화사(universal quantifier)라고 불리는데, “for all”을 의미한다. 우리는 집합이나 자연수들 사이의 관계 위에 양화하지 않고 오직 자연수만을 양화한다.<sup>22</sup>

여느 때처럼, 기호 “=”는 좌변과 우변이 같다는 관계를 의미하고 기호 “ $\leq$ ”는 좌변이 우변보다 같거나 작은 관계를 의미한다.

또한 개체 변수(individual variable)라고 불리는 가부번하게 많은 표현식들  $v_1, v_2, \dots, v_n, \dots$ 이 필요하다. 이 표현식들을 13 개의 알파벳만으로 표현하기 위하여,  $v_1, v_2, v_3, \dots$ 을 각각  $(v_0), (v_{00}), (v_{000}), \dots$ 로 표현하자.<sup>23</sup>

**Terms and Formulas.** 다음 두 규칙들의 결과로서 존재하는 표현식을 논리항(term)이라고 부른다.

<sup>20</sup>다음 수 함수와 이하의 관계도 넣을 것이지만, 이것들은 불필요하다.

<sup>21</sup>화살표 기호가 익숙하지 않은 독자에게: 두 명제  $p$ 와  $q$ 에 대하여,  $p \rightarrow q$ 는  $p$ 가 거짓인 경우와  $p$ 와  $q$  모두 참인 경우에 그리고 이 두 경우에만 참인 명제이다.

<sup>22</sup>기술적으로, 우리는 1차 산술을 다루는 것이지, 2차 산술을 다루는 것은 아니다.

<sup>23</sup>그러므로  $v_n$ 은 “ $v$ ” 뒤에  $n$ 개의 ○가 따라온 표현식에 팔호를 감싼 것이다.

1. 모든 개체 변수들과 숫자들은 논리항이다.
2.  $t_1$ 과  $t_2$ 가 논리항일 때,  $(t_1 + t_2)$ ,  $(t_1 \times t_2)$ ,  $(t_1 \rightarrow t_2)$ 와  $t'_1$ 도 항이다.

논리항이 개체 변수를 포함하고 있지 않으면, 그것을 상항(constant term)이라고 부르거나 닫혀 있다(closed)고 한다.

원자 논리식(atomic formula)이란,  $t_1 = t_2$ 와  $t_1 \leq t_2$  중 한 형태의 표현식을 의미한다 – 여기서  $t_1$ 과  $t_2$ 에는 임의의 논리항이 올 수 있다.

모든 논리식(formula)들의 집합은 다음 규칙에 의하여 귀납적으로 정의된다:

1. 모든 원자 논리식은 논리식이다.
2.  $F$ 와  $G$ 가 논리식이라면,  $\neg F$ 과  $F \rightarrow G$  그리고  $\forall v_i F$ 도 논리식이다 – 여기서  $v_i$ 는 임의의 개체 변수이다.

¬

**Free and Bound Occurrences of Variables.**  $v_i$ 를 개체 변수라고 하자. 항  $t$ 에 대하여,  $t$ 에서의  $v_i$ 의 모든 나타남들은 자유 나타남(free occurrence)이라고 불린다. 또한, 원자 논리식  $A$ 에 대하여서도,  $A$ 에서의  $v_i$ 의 모든 나타남들은 자유 나타남이다. 논리식  $F$ 와  $G$ 에 대하여,  $F \rightarrow G$ 에서의  $v_i$ 의 모든 자유 나타남들은  $F$ 와  $G$ 의 그것과 같다. 논리식  $F$ 에 대하여,  $\neg F$ 에서의  $v_i$ 의 모든 자유 나타남들은  $F$ 의 그것과 같다. 이때  $v_i$ 는  $\forall v_i F$ 에서 자유롭게 나타나지 않는다;  $\forall v_i F$ 에서의  $v_i$ 의 모든 나타남들은 속박 나타남(bound occurrence)이라고 불린다.  $j \neq i$ 인 각  $j \in \mathbb{N}_+$ 마다  $\forall v_j F$ 에서의  $v_i$ 의 모든 자유 나타남들은  $F$ 의 그것과 같다.

¬

**Sentences.** 문장(sentence)이란, 어떤 개체 변수도 자유롭게 나타나지 않는 논리식을 의미한다. 문장들은 때때로 닫힌(closed) 논리식이라고도 불린다. 열린(open) 논리식이란, 닫히지 않은 논리식을 의미한다 – 즉, 적어도 하나의 개체 변수가 자유롭게 나타나는 논리식이다.

¬

**Substitution of Numerals for Variables.** 자연수  $n$ 에 대하여,  $\bar{n}$ 은  $n$ 을 가르키는 숫자를 의미한다 – 즉, 기호 “0” 뒤에  $n$ 개의 악센트 기호가 따라오는 표현식이다.<sup>24</sup>

개체 변수  $v_i$ 에 대하여, 우리는 논리식  $F$ 에서  $v_i$ 의 모든 자유 나타남들을  $\bar{n}$ 으로 대체한 논리식을  $F[v_i := \bar{n}]$ 으로 적을 것이다. 더 일반적으로,  $F[v_{i_1} := \bar{n}_1, \dots, v_{i_k} := \bar{n}_k]$ 은 논리식  $F$ 에서  $v_{i_1}, \dots, v_{i_k}$ 을 동시에 각각  $\bar{n}_1, \dots, \bar{n}_k$ 으로 치환한 결과를 의미한다.

자유롭게 나타나는 개체 변수가  $v_1, \dots, v_n$  뿐인 논리식을 정규(regular) 논리식이라고 한다. 그러므로, 정규 논리식  $F$ 에서 개체 변수  $v_i$ 가 자유롭게 나타난다면,  $j \leq i$ 인 임의의  $j \in \mathbb{N}_+$ 에 대하여  $v_j$ 도  $F$ 에서 자유롭게 나타나는 개체 변수이다.

¬

**Degrees and Induction.** 논리식의 복잡도(degree)란, 그 논리식에서  $\neg, \rightarrow, \forall$ 이 나타나는 횟수이다. 그러므로,

<sup>24</sup> 예를 들어,  $\bar{5}$ 는 표현식  $0^{''''}$ 이다.

1. 원자 논리식의 복잡도는 0이다.
2. 논리식  $F$ 의 복잡도가  $d_1$ 이고 논리식  $G$ 의 복잡도가  $d_2$ 이면,  $\neg F$ 의 복잡도는  $d_1 + 1$ 이고;  $(F \rightarrow G)$ 의 복잡도는  $d_1 + d_2 + 1$ 이고;  $\forall v_i F$ 의 복잡도는  $d_1 + 1$ 이다 – 여기서  $v_i$ 는 임의의 개체 변수이다.

다음의 수학적 귀납법에 익숙하다고 가정하겠다: 주어진 성질이 모든 논리식에 대하여 성립함을 보이기 위하여 모든 원자 논리식에 대하여 그 성질이 성립함과 임의의 논리식  $F$ 에 대하여  $F$ 보다 낮은 복잡도의 모든 논리식들에 대하여 그 성질이 성립한다고 가정하고  $F$ 에서도 성립함을 보이면 충분하다.  $\dashv$

**Abbreviations.** 우리는 다음의 표준적인 약어를 쓸 것이다:

$$\begin{aligned}
 (F \vee G) &:= (\neg F \rightarrow G) \\
 (F \wedge G) &:= \neg(F \rightarrow \neg G) \\
 (F \leftrightarrow G) &:= ((F \rightarrow G) \wedge (G \rightarrow F)) \\
 \exists v_i F &:= \neg \forall v_i \neg F \\
 t_1 \neq t_2 &:= \neg t_1 = t_2 \\
 t_1 < t_2 &:= (t_1 \leq t_2 \wedge t_1 \neq t_2) \\
 t_1^{t_2} &:= (t_1 \mathbf{E} t_2) \\
 (\forall v_i \leq t) F &:= \forall v_i (v_i \leq t \rightarrow F) \\
 (\exists v_i \leq t) F &:= \neg (\forall v_i \leq t) \neg F
 \end{aligned}$$

여기서  $F$ 와  $G$ 에는 임의의 논리식이 올 수 있고,  $t_1$ 과  $t_2$ 에는 임의의 논리항이 올 수 있으며,  $v_i$ 에는 임의의 개체 변수가 올 수 있다.

논리항과 논리식을 표기할 때, 괄호를 생략하여도 모호함이 생기지 않는 경우에는 종종 괄호를 생략할 것이다. 예를 들어, 독립적인 논리항과 논리식을 표기할 때, 가장 바깥쪽 괄호를 떨어뜨릴 수 있다 – [예 1]  $(F \rightarrow G)$  대신에  $F \rightarrow G$ 라고 쓴다; [예 2] 독립적인 항  $(v_1 + v_2)$ 는  $v_1 + v_2$ 로 단축할 수 있다; [예 3]  $((v_1 + v_2) \times v_3)$ 은  $(v_1 + v_2) \times v_3$ 으로 단축될 수 있다.  $\dashv$

**Designation.** 상항이 개체 변수가 나타나지 않는 항이라는 사실을 상기하자. 임의의 상항은 다음 규칙에 따라 유일한 자연수에 대응한다.

1. 숫자  $\bar{n}$ 은  $n$ 에 대응한다.
2. 두 상항  $c_1$ 과  $c_2$ 가 각각  $n_1$ 과  $n_2$ 에 대응할 때,  $(c_1 + c_2)$ 는  $n_1$ 과  $n_2$ 의 합에 대응하고;  $(c_1 \times c_2)$ 는  $n_1$ 과  $n_2$ 의 곱에 대응하고;  $(c_1 \mathbf{E} c_2)$ 는  $n_1$ 의  $n_2$  제곱에 대응한다.

한 예로, 상항  $((0'' + 0') \times (0'' \mathbf{E} 0''))'$ 는 수  $((3 + 1) \times 2^3) + 1 = 33$ 에 대응한다.  $\dashv$

## §2. $\mathcal{L}_E$ 에서의 참의 개념.

이제  $\mathcal{L}_E$ 의 어떠한 문장들이 참이라는 게 무엇을 의미하는지를 정의할 때가 되었다. 이 정의는 문장의 복잡도에 대한 귀납법에 의하여 적용된다. 다음 조건들은 진리에 대한 귀납적인 정의를 제공한다.

$T_0$ : 1. 상항  $c_1$ 과  $c_2$ 에 대하여, 원자 논리식인 문장  $c_1 = c_2$ 는  $c_1$ 과  $c_2$ 가 같은 자연수에 대응할 때 그리고 그럴 때에만 참이다.

2. 상항  $c_1$ 과  $c_2$ 에 대하여, 원자 논리식인 문장  $c_1 \leq c_2$ 는  $c_1$ 이 대응하는 자연수가  $c_2$ 가 대응하는 자연수보다 작거나 같을 때 그리고 그럴 때에만 참이다.

$T_1$ :  $\neg X$  꼴의 문장은  $X$ 가 참이 아닐 때 그리고 그럴 때에만 참이다.

$T_2$ :  $X \rightarrow Y$ 는  $X$ 가 참이 아닌 경우 또는  $X$ 와  $Y$  모두 참인 경우에 그리고 이 두 경우에만 참이다.

$T_3$ :  $\forall v_i F$ 은 모든 자연수  $n$ 에 대하여 문장  $F[v_i := \bar{n}]$ 이 참일 때 그리고 그럴 때에만 참이다.

조건  $T_0$ 는 노골적으로 원자 논리식인 문장이 참일 조건을 진술한다. 조건  $T_1$ ,  $T_2$ 와  $T_3$ 은, 더 낮은 복잡도의 문장들에 대한 진리를 이용하여, 원자 논리식이 아닌 문장이 참일 조건을 기술한다.<sup>25</sup>

열린 논리식은 참이라고 혹은 거짓이라고 말할 수는 없지만, 임의의  $n_1 \in \mathbb{N}$ , ...,  $n_k \in \mathbb{N}$ 에 대하여  $F[v_{i_1} := \bar{n_1}, \dots, v_{i_k} := \bar{n_k}]$ 가 참일 때 자유롭게 나타나는 개체 변수가  $v_{i_1}, \dots, v_{i_k}$  뿐인 논리식  $F$ 를 정확(*correct*)하다고 한다 – 단,  $k \geq 1$ 이다.

### Exercise 1.

(a) 임의의 문장  $X$ 와  $Y$ 에 대하여,  $X \wedge Y$ 는  $X$ 와  $Y$  모두 참일 때 그리고 그럴 때에만 참이다.

(b) 임의의 문장  $X$ 와  $Y$ 에 대하여,  $X \vee Y$ 는  $X$ 와  $Y$  중 적어도 하나가 참일 때 그리고 그럴 때에만 참이다.

(c) 자유롭게 나타나는 개체 변수가  $v_i$  뿐인 논리식  $F$ 에 대하여,  $\exists v_i F$ 는 어떤 자연수  $n$ 이 존재하여  $F[v_i := \bar{n}]$ 이 참일 때 그리고 그럴 때에만 참이다.

위의 세 명제가 모두 참임을 보여라.  $\dashv$

**Substitution of Variables.** 자유롭게 나타나는 개체 변수가  $v_1$  뿐인 논리식  $F$ 를 고려하자.  $i > 1$ 일 때, 개체 변수  $v_i$ 에 대하여,  $F[v_1 := v_i]$ 를 다음과 같이 정의한다:

---

<sup>25</sup> $T_3$ 에서  $F[v_i := n]$ 의 복잡도가  $\forall v_i F$ 의 복잡도보다 낮은 것과  $\forall v_i F$ 가 문장이므로  $F[v_i := n]$ 도 문장임에 주목하여라.

1.  $v_i$ 가  $F$ 에서 속박되어 나타나지 않을 때,  $F[v_1 := v_i]$ 는  $v_1$ 의 모든 자유 나타남을  $v_i$ 로 치환한 결과이다.

2.  $v_i$ 가  $F$ 에서 속박되어 나타날 때,  $v_j$ 가  $F$ 에서 나타나지 않는 개체 변수가 되게 하는 가장 작은  $j$ 를 취하여,  $v_i$ 의 모든 나타남을  $v_j$ 으로 치환한 논리식에서  $v_j$ 의 모든 자유 나타남을  $v_i$ 로 치환한 결과이다.

예를 들어, 논리식  $\exists v_2 v_2 \neq v_1$ 을  $F$ 라고 하자. 그러면  $F$ 는 정확하다.  $F[v_1 := v_2]$ 이란 정확한 논리식  $\exists v_3 v_3 \neq v_2$ 를 의미하지, 정확하지 않은 논리식  $\exists v_2 v_2 \neq v_2$ 를 의미하지는 않는다.

자유롭게 나타나는 개체 변수가  $n$  개인 정규 논리식에 대하여서도 비슷한 정의를 적용한다. 개체 변수  $v_{i_1}, \dots, v_{i_n}$ 에 대하여,  $F[v_1 := v_{i_1}, \dots, v_n := v_{i_n}]$ 을  $v_{i_1}, \dots, v_{i_n}$  중 어느 것도  $F$ 에서 속박되어 나타나지 않도록  $F$ 를 재작성하여 얻은 논리식에  $v_1, \dots, v_n$ 을 동시에  $v_{i_1}, \dots, v_{i_n}$ 으로 치환하여 얻은 논리식으로 정의한다.  $\dashv$

두 문장이 동시에 참이거나 거짓일 때, 두 문장이 (산술적으로) 동등(*equivalent*)하다고 한다. 자유롭게 나타나는 개체 변수가  $v_{i_1}, \dots, v_{i_k}$  뿐인 논리식  $F$ 와  $G$ 에 대하여, 임의의  $n_1 \in \mathbb{N}, \dots, n_k \in \mathbb{N}$ 에 대하여  $F[v_{i_1} := \bar{n}_1, \dots, v_{i_k} := \bar{n}_k]$ 이 동등한 문장일 때,  $F$ 와  $G$ 를 동등한 논리식이라고 한다.

### §3. 쌈술적인 그리고 산술적인 집합과 관계

자유롭게 나타나는 개체 변수가  $v_1$  뿐인 논리식  $F$ 에 대하여,  $F$ 는 문장  $F[v_1 := \bar{n}]$ 이 참이게 하는 자연수  $n$ 을 모두 모은 집합을 표현한다(*express*)고 한다; 즉, 임의의 집합  $A \subseteq \mathbb{N}$ 에 대하여,

$$(\forall n \in \mathbb{N}) [F[v_1 := \bar{n}] \in \mathcal{T} \leftrightarrow n \in A]$$

이 성립할 때 그리고 그럴 때에만  $F$ 는  $A$ 를 표현한다. 자유롭게 나타나는 개체 변수가  $k$ 개인 정규 논리식  $F$ 에 대하여,  $F$ 는 문장  $F[v_{i_1} := \bar{n}_1, \dots, v_{i_k} := \bar{n}_k]$ 이 참이게 하는  $k$ -튜플  $(n_1, \dots, n_k)$ 을 모두 모은 집합을 표현한다(*express*)고 한다; 즉, 임의의 관계  $R \subseteq \mathbb{N}^k$ 에 대하여,

$$(\forall n_1 \in \mathbb{N}) \cdots (\forall n_k \in \mathbb{N}) [F[v_{i_1} := \bar{n}_1, \dots, v_{i_k} := \bar{n}_k] \in \mathcal{T} \leftrightarrow R(n_1, \dots, n_k)]$$

이 성립할 때 그리고 그럴 때에만  $F$ 는  $R$ 을 표현한다.

예를 들어, 모든 자연수는 짝수일 때 그리고 그럴 때에만 2로 나누어 떨어지므로, 모든 짝수들의 집합은 논리식  $\exists v_2 v_1 = 0'' \times v_2$ 에 의하여 표현된다.

집합이나 관계가 어떤  $\mathcal{L}_E$ 의 논리식에 의하여 표현될 때 그것을 쌈술적(*Arithmetic*)이라고 한다. 그리고 집합이나 관계가 **E**를 포함하지 않는 어떤  $\mathcal{L}_E$ 의 논리식에 의하여 표현될 때 그것을 산술적(*arithmetic*)이라고 한다. 비형식적으로, 쌈술적인 집합과 관계는 1차 논리에서 덧셈과 곱셈 그리고 제곱에 의하여 정의될 수 있다는 특성을 가지고 있다; 그리고 산술적인 집합과 관계는

1차 논리에서 덧셈과 곱셈에만 의하여 정의될 수 있다는 특성을 가지고 있다.<sup>26</sup>

뒷 단원에서, 관계  $x^y = z$ 를 덧셈과 곱셈만으로 정의할 수 있기 때문에 모든 쌍술적인 관계가 산술적인 관계라는, 자명하지 않은 Gödel의 결론을 증명할 것이다. 그러나 이것을 증명하기 전까지, “쌍술적인”이라는 용어를 계속 사용할 것이다.

함수  $f : \mathbb{N}^k \rightarrow \mathbb{N}$ 를, 관계

$$f(x_1, \dots, x_k) = y$$

가 쌍술적일 때, 쌍술적(Arithmetic)이라고 부를 것이다. 다시 말해, 임의의 함수  $f : \mathbb{N}^k \rightarrow \mathbb{N}$ 에 대하여,  $f$ 가 쌍술적일 때 그리고 그럴 때에만 자유롭게 나타나는 개체 변수가  $k + 1$ 개인 어떤 정규 논리식  $F$ 가 존재하여 임의의  $x_1 \in \mathbb{N}, \dots, x_k \in \mathbb{N}, y \in \mathbb{N}$ 에 대하여

$$F[v_1 := \bar{x_1}, \dots, v_k := \bar{x_k}, v_{k+1} := \bar{y}] \in \mathcal{T} \leftrightarrow f(x_1, \dots, x_k) = y$$

가 성립한다.

성질(property)  $P$ 를 만족시키는 모든 자연수들의 집합이 쌍술적일 때  $P$ 를 쌍술적이라고 할 것이다. 또한 “조건”이란 용어를 관계나 성질을 의미하는 용어로서 사용할 것이다.

### Exercise 2.

(a) 관계 “ $x$ 가  $y$ 의 약수이다”는 산술적이다.

(b) 모든 소수들의 집합은 산술적이다.

위의 두 명제가 모두 참임을 보여라.  $\dashv$

**Exercise 3.** 임의의 쌍술적인 집합  $A \subseteq \mathbb{N}$ 와 임의의 쌍술적인 함수  $f : \mathbb{N} \rightarrow \mathbb{N}$ 에 대하여, 집합

$$f^{-1}[A] = \{n \in \mathbb{N} : f(n) \in A\}$$

도 쌍술적임을 보여라. “산술적인”에 대하여 같은 것을 보여라.  $\dashv$

### Exercise 4.

(a) 임의의 두 함수  $f : \mathbb{N} \rightarrow \mathbb{N}$ 과  $g : \mathbb{N} \rightarrow \mathbb{N}$ 가 쌍술적일 때 함수  $x \mapsto f(g(x))$ 도 쌍술적이다.

(b) 임의의 두 함수  $f : \mathbb{N} \rightarrow \mathbb{N}$ 과  $g : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ 가 쌍술적일 때 세 함수  $(x, y) \mapsto g(f(x), y)$  와  $(x, y) \mapsto g(x, f(y))$  그리고  $(x, y) \mapsto f(g(x, y))$  모두 쌍술적이다.

위의 두 명제가 모두 참임을 보여라.  $\dashv$

---

<sup>26</sup>관계  $\leq$ 를 넣지 않은 이유는 다음과 같다: 논리식  $(v_1 \leq v_2 \leftrightarrow (\exists v_3 v_1 + v_3 = v_2))$ 이 정확하기 때문에,  $\leq$ 는 덧셈만으로 표현될 수 있다.

**Exercise 5.** 무한 집합  $A \subseteq \mathbb{N}$ 가 짠술적이라고 하자. 그러면  $A$ 가 무한 집합이므로

$$(\forall x \in \mathbb{N}) (\exists y \in \mathbb{N}) [y \in A \wedge x \leq y]$$

가 성립한다. 이제  $R(x, y)$ 를 “ $x$ 보다 큰  $A$ 의 원소들의 최솟값이  $y$ 이다”는 관계로 두자. 그러면 관계  $R \subseteq \mathbb{N}^2$ 이 짠술적임을 보여라.  $\dashv$

## 2.2 접합과 괴델 넘버링

### §4. $b$ 를 기저로 하는 접합

각  $b \geq 2$ 에 대하여,  $b$ 를 기저로 하는 접합(concatenation to the base  $b$ )라고 불릴 함수

$$(x, y) \mapsto x *_b y : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$$

를 정의할 것인데, 이 함수는 이 책에서 기본적인 역할을 할 것이다.

저 정의에 친숙한 기저인 10을 주자. 임의의  $m \in \mathbb{N}$ 과  $n \in \mathbb{N}$ 에 대하여,  $m *_{10} n$ 을  $m$ 의 십진법 표기 뒤에  $n$ 의 십진법 표기를 붙인 수로 정의하자. 예를 들어,  $53 *_{10} 792 = 53792$ 이다. 우리는  $53792 = 53000 + 792 = 53 \times 10^3 + 792$ 임에 주목한다. 3은 (십진법으로 표기된) 792의 자릿수이다 – 또는, 우리는 3을 792의 길이(length)라고 말할 것이다.

더 일반적으로는,  $m *_{10} n = m \times 10^{\text{len}_{10}(n)} + n$ 이다 – 여기서  $\text{len}_{10}(n)$ 은 (십진법으로 표기된)  $n$ 의 길이이다.

더욱 일반적으로는, 각  $b \geq 2$ 에 대하여, 우리는  $b$ 진 표기법에서  $m *_b n$ 을  $m$  뒤에  $n$ 을 붙인 수로 정의할 것이다. 그러면  $m *_b n = m \times b^{\text{len}_b(n)} + n$ 이다 – 여기서  $\text{len}_b(n)$ 은  $n$ 을  $b$ 진 표기법으로 나타냈을 때 그것의 자릿수이다. 다음 정리는 기본적이다.

**Proposition 1.** 각각의  $b \geq 2$ 에 대하여, 관계  $x *_b y = z$ 는 짠술적이다.  $\dashv$

증명을 하기에 앞서, 우리는 본질적인 아이디어를 진술한다. 먼저 친숙한 기저인 10을 고려하자. 임의의  $n > 0$ 에 대하여,  $\text{len}_{10}(n)$ 은  $10^k > n$ 을 만족시키는 가장 작은  $k$ 이고,  $10^{\text{len}_{10}(n)}$ 은  $n$ 보다 큰 10의 거듭제곱 중 가장 작은 것이다. (예를 들어,  $10^{\text{len}_{10}(5368)} = 10^4 = 10,000$ 인데, 이는 5368보다 큰 10의 거듭제곱 중 가장 작은 것이다.) 일반적으로, 임의의  $b \geq 2$ 와 임의의  $n \in \mathbb{N}$ 에 대하여,  $b^{\text{len}_b(n)}$ 은  $n > 0$ 이면  $n$ 보다  $b$ 의 거듭제곱 중 가장 작은 것과 같고 그렇지 않으면  $b$ 와 같다.

*Proof.*  $b \geq 2$ 를 잡자.

1.  $\text{Pow}_b(x)$ 을  $x$ 가  $b$ 의 거듭제곱 중 하나라는 조건으로 두자. 이 조건은 짠술적이다. 왜냐하면

$(\exists y \in \mathbb{N}) [x = b^y]$ 일 때 그리고 그럴 때에만  $\text{Pow}_b(x)$ 가 성립하기 때문이다.<sup>27</sup>

2. ( $x$ 와  $y$  사이의 관계로서의) 관계  $b^{\text{len}_b(x)} = y$ 는, 우리가 주목했듯이, 조건

$$(x = 0 \wedge y = b) \vee (x \neq 0 \wedge s(x, y))$$

과 동등하다 – 여기서,  $s(x, y)$ 는 “ $y$ 가  $x$ 보다 큰 거듭제곱 중 가장 작다”는 관계이다. 이것은 쌍술적인데, 왜냐하면

$$(\text{Pow}_b(y) \wedge x < y) \wedge (\forall z \in \mathbb{N}) [(\text{Pow}_b(z) \wedge x < z) \rightarrow y \leq z]$$

일 때 그리고 그럴 때에만  $s(x, y)$ 가 성립하기 때문이다.

3. 최종적으로, 관계  $x \times b^{\text{len}_b(y)} + y = z$  ( $\equiv$ ,  $x *_b y = z$ )는 다음 조건과 같다:

$$(\exists w \in \mathbb{N}) [b^{\text{len}_b(y)} = w \wedge x \times w + y = z].$$

따라서, 관계  $x *_b y = z$ 는 쌍술적이다.

□

임의의  $x > 0$ 와  $y > 0$  그리고  $z > 0$ 에 대하여,

$$(x *_b y) *_b z = x *_b (y *_b z)$$

임에 주목하자. 그러나  $y = 0$ 이면, 이것은 실패할 수도 있다.<sup>28</sup> 그러므로 괄호를 생략하는 경우 그 식에 좌측으로 결합하는 괄호가 있는 것으로 간주하겠다. (예를 들어,  $x *_b y *_b z$ 는  $(x *_b y) *_b z$ 를 의미하지만  $x *_b (y *_b z)$ 를 의미하지는 않는다.)

**Corollary 1.** 각각의  $n \geq 2$ 과 각각의  $b \geq 2$ 에 대하여, ( $y, x_1, x_2, \dots, x_n$  사이의  $(n+1)$ -항 관계로서의) 관계

$$y = x_1 *_b x_2 *_b \cdots *_b x_n$$

는 쌍술적이다. ¬

*Proof.*  $n$ 에 대한 귀납법을 적용하자. 우리는 이미  $n = 2$ 인 경우에 대하여 증명했다. 이제  $n \geq 2$

<sup>27</sup>더 형식적으로는, 논리식  $\exists v_2 v_1 = (\bar{b} \mathbf{E} v_2)$ 을 모든  $b$ 의 거듭제곱들의 집합을 표현한다. 앞으로는 이렇게 형식적으로 나타내지 않을 것이다.

<sup>28</sup>Quine의 예를 빌리자면,  $(5 *_{10} 0) *_{10} 3 = 50 *_{10} 3 = 503$ 이지만,  $0 *_{10} 3 = 3$ 이므로  $5 *_{10} (0 *_{10} 3) = 5 *_{10} 3 = 53$ 이기 때문에 실패한다.

에 대하여 관계  $y = x_1 *_b x_2 *_b \cdots *_b x_n$ 가 쌍술적이라고 하자. 그러면

$$y = x_1 *_b x_2 *_b \cdots *_b x_n *_b x_{n+1}$$

일 때 그리고 그럴 때에만

$$(\exists z \in \mathbb{N}) [z = x_1 *_b x_2 *_b \cdots *_b x_n \wedge y = z *_b x_{n+1}]$$

이므로, 관계  $y = x_1 *_b x_2 *_b \cdots *_b x_n *_b x_{n+1}$ 는 쌍술적이다.  $\square$

## §5. 괴델 넘버링

쌍술적인 문장들(즉,  $\mathcal{L}_E$ 의 문장들)은 수에 대하여 이야기하지,  $\mathcal{L}_E$ 의 표현식들을 이야기하지는 않는다. 표현식들에게 괴델 수를 할당하는 목적은 바로 문장이 문장들에 대하여 그것들의 괴델 수를 이용하여 간접적으로 말하는 것을 허용하기 위함이다.

우리가 사용할 괴델 넘버링은 Quine[1940]을 수정한 것이다. Quine은 그의 언어를 9개의 알파벳  $S_1, S_2, \dots, S_9$ 으로 이루어진 언어로 형식화하였다. 그리고서는 각 표현식  $S_{i_1}S_{i_2}\cdots S_{i_n}$ 에 입진 표기의  $i_1i_2\cdots i_n$ 를 괴델 수를 할당하였다. (예를 들어, 표현식  $S_3S_1S_2$ 는 괴델 수로 312를 할당받는다.)

우리의 언어  $\mathcal{L}_E$ 는 13개의 기호들을 사용하므로, 접합의 기저로서 10 대신 13을 취하여 Quine의 아이디어를 차용할 것이다.<sup>29</sup>

우리는 13진수의 숫자로서 10, 11, 12로서 각각 “ $\eta$ ”, “ $\varepsilon$ ”, “ $\delta$ ”를 사용할 것이다. 그리고 13개의 기호에 각각 다음과 같은 괴델 수를 할당하겠다:

0	'	(	)	$f$	$\circ$	$v$	$\neg$	$\rightarrow$	$\forall$	$=$	$\leq$	#
1	0	2	3	4	5	6	7	8	9	$\eta$	$\varepsilon$	$\delta$

그 다음, 이 기호들의 임의의 문자열에 대하여, 그것의 괴델 수로 그것의 각 기호를 그에 대응하는 13진법 숫자로 대체하여 13진법으로 읽은 수를 부여한다. 예를 들어 4번째 기호 다음에 7번째 기호 다음에 3번째 기호가 나오는 길이 3의 문자열의 괴델 수를 13진법으로 표기한 것은 “362”이다 – 즉, 수  $2 + 6 \times 13 + 3 \times 13^2$ 이다.

각  $n > 0$ 에 대하여,  $E_n$ 을  $n$ 을 괴델 수로 할당받은 유일한 문자열로 정의하자. 한편, 악센트 기호로만 이루어진 문자열은 모두 괴델 수로 0을 부여받는데,  $E_0$ 을 악센트 기호 한 개 그 자체로 정의하겠다. 그러므로 우리는 표현식(expression)이라는 단어를 악센트 기호로 시작하지 않거나 악센트 기호 한 개 그 자체인 그 13개의 기호들의 문자열이라는 뜻으로 사용하겠다.

---

<sup>29</sup> 13이라는 수는 우연히도 소수인데, 우리는 뒷 단원에서 합성수를 접합의 기저로 취하는 것보다 기술적 이득이 있음을 보게 될 것이다.

0을 악센트 기호의 괴델 수로 택한 까닭은 다음과 같다: 임의의  $n \in \mathbb{N}$ 에 대하여, (다른 표현식들과 마찬가지로) 숫자  $\bar{n}$ 은 괴델 수를 가진다. 이때 함수  $n \mapsto g(\bar{n}) : \mathbb{N} \rightarrow \mathbb{N}$ 이 쌍술적이기를 원한다. 당연히,  $\bar{n}$ 은 “0” 뒤에  $n$ 개의 악센트 기호가 붙은 것이므로, 그것의 괴델 수를 13진법으로 표기하면 “1” 뒤에 “0”이  $n$ 번 나타나는 문자열과 같다. 따라서  $g(\bar{n}) = 13^n$ 이다.

**Discussion.** 실제로, 이 단원과 다음 두 단원의 모든 결과들은 다음 두 성질을 가지는 임의의 괴델 수에 대하여 통한다:

1. 쌍술적인 함수  $(x, y) \mapsto x \bullet y : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ 가 존재하여,

$$(\forall x \in \mathbb{N}) (\forall y \in \mathbb{N}) (\forall z \in \mathbb{N}) [E_x E_y \equiv E_z \rightarrow x \bullet y = z]$$

가 성립한다.

2. 함수  $n \mapsto g(\bar{n}) : \mathbb{N} \rightarrow \mathbb{N}$ 가 쌍술적이다.

저 두 성질을 최대한 빨리 달성하기 위하여 저러한 특정 괴델 넘버링을 택하였다. 또한 첫 번째 성질을 가지는 괴델 넘버링은 두 번째 성질도 가진다는 데에 주목할 수 있다. (이것은 다음 몇 개의 단원 전까지는 분명하지 않다; 그것은 우리가 자연수의 이름을 선택한 특정 방식에 달려있다.)  $\dashv$

당연히, 기호들에게 다음과 같은 괴델 수를 부여함으로써, 13 대신 10을 괴델 넘버링의 기저로 취할 수 있었고,

$$\begin{array}{ccccccccccccc} 0 & ' & ( & ) & f & \circ & v & \neg & \rightarrow & \forall & = & \leq & \# \\ 1 & 0 & 2 & 3 & 4 & 5 & 6 & 7 & 89 & 899 & 8999 & 89999 & 899999 & \end{array}$$

십진 표기법을 더 편안하게 여기는 독자들은 그가 원하면 이러한 괴델 넘버링을 쓸 수 있다. 이러한 괴델 넘버링 하에서는,  $\bar{n}$ 의 괴델 수는  $13^n$ 이 아니라  $10^n$ 이 된다. 그러나 우리가 전에 주목했듯이, 소수인 13을 기저로 하는 데에 특정한 기술적 이득이 있다.

이 장의 나머지 부분(과 다음 나머지 단원들)에서, 기저를 13으로 하는 괴델 수만을 다룰 것이며, 따라서,  $x * y$ 는  $x *_{13} y$ 를 의미할 것이다.<sup>30</sup>

## 2.3 Tarski의 정리

### §6. 대각화와 괴델 문장

---

<sup>30</sup> 10을 기저로 취하고자 하는 독자는 “ $x * y$ ”를  $x *_{10} y$ 로 읽을 수 있어도 좋다. 그러나  $13^n$ 이 쓰일 때마다 그 또는 그녀는  $10^n$ 을 써야할 것이다.

집합  $T \subseteq \mathbb{N}$ 을  $T := \{\mathbf{g}(X) : X \in \mathcal{T}\}$ 으로 놓자. 이  $T$ 라는 집합은 완벽하게 정의된 [자연수들의 집합]이다. 그런데 이것이 짠술적일까? 우리는 그렇지 않다는 것을 보일 것이다 – Tarski의 정리.

1단원에서처럼, 집합  $A \subseteq \mathbb{N}$ 에 대하여  $X$ 가 참이고 그것의 괴델 수가  $A$ 에 속하거나  $X$ 가 거짓이고 그것의 괴델 수가  $A$ 에 속하지 않을 때 문장  $X$ 을  $A$ 에 대한 괴델 문장(Gödel sentence)이라고 하겠다. 이제 (Tarski의 정리가 쉽게 따라오도록) 임의의 짠술적인 집합  $A \subseteq \mathbb{N}$ 에 대하여  $A$ 에 대한 괴델 문장이 있음을 보이는 데 초점을 맞추겠다.

거의 모든 책이 괴델 문장을 구성하는 여러 가지 영리한 방법으로 쓰여질 수 있다. Gödel의 원래 방법은 임의의 논리식  $F$ 와  $G$  그리고 임의의 수  $n$ 에 대하여,

$$F[v_1 := \bar{n}] \equiv G \rightarrow \text{sub}(\mathbf{g}(F), n) = \mathbf{g}(G)$$

만족시키는 짠술적인 함수  $\text{sub} : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ 의 존재를 보이는 것을 수반한다. 이것을 수행하는 것은 개체 변수를 숫자로 치환하는 연산을 산술화하는 것과 관련되어 있는데, 이것은 꽤나 복잡한 일이다. 대신에, Tarski[1953]의 단순하지만 영리한 생각을 활용할 것이다.

비형식적으로, 성질  $P$ 가 수  $n$ 에 대하여 성립한다고 말하는 것은 임의의 수  $x$ 에 대하여  $x$ 가  $n$ 과 같을 때마다  $P$ 가  $x$ 에 대하여 성립한다고 말하는 것과 같다. 형식적으로, 개체 변수  $v_1$ 만이 자유롭게 나타나는 논리식  $F$ 에 대하여, 문장  $F[v_1 := \bar{n}]$ 은 문장  $\forall v_1 (v_1 = \bar{n} \rightarrow F)$ 과 동등하다.<sup>31</sup> 이제 문장  $F[v_1 := \bar{n}]$ 의 괴델 수가  $F$ 의 괴델 수와  $n$ 의 짠술적인 함수라는 게 중점이다.

각각의 논리식  $F$ 와 각각의 수  $n$ 에 대하여  $F(n) := \forall v_1 (v_1 = \bar{n} \rightarrow F)$ 로 표기하자. 재차 강조하자면, 논리식  $F$ 에  $v_1$ 만이 자유롭게 나타나는 개체 변수라면,  $F[v_1 := \bar{n}]$ 과  $F(n)$ 은 같진 않더라도 동등한 문장이라는 것이다 – 즉, 동시에 참이거나 동시에 거짓이다. 사실, 임의의 표현식  $E$ 에 대하여,  $E$ 가 논리식이든 아니든,  $\forall v_1 (v_1 = \bar{n} \rightarrow E)$ 은 ( $E$ 가 논리식이 아닐 경우 의미 없는 표현식일지라도) 잘 정의된 표현식이고, (표현식  $E(n)$ 의 의미가 없을 수도 있지만)

$$E(n) := \forall v_1 (v_1 = \bar{n} \rightarrow E)$$

로 단축하겠다. 이때,  $E$ 가 논리식일 경우  $E(n)$ 은 논리식이지만 문장일 필요는 없으며,  $E$ 가 변수  $v_1$ 만이 자유롭게 나타나는 논리식인 경우 당연히  $E(n)$ 은 문장이나, 이 모든 경우에서  $E(n)$ 은 잘 정의된 논리식이다.

함수  $r : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ 을

$$r(x, y) := \mathbf{g}(E_x(y))$$

으로 정의하자. 즉, 임의의  $e \in \mathbb{N}$ 과 임의의  $n \in \mathbb{N}$ 에 대하여,  $e$ 를 괴델 수로 받은 표현식을  $E$ 라

<sup>31</sup>덧붙여서, 이는  $\exists v_1 (v_1 = \bar{n} \wedge F)$ 와도 동등하다.

하면,  $r(e, n)$ 은 표현식  $E(n)$ 의 괴델 수이다.  $r$ 은 이 책에서 매우 중요한 역할을 하는 함수이며, 우리는 곧 이 함수가 쌍술적임을 보이겠다.

$x \in \mathbb{N}$ 과  $y \in \mathbb{N}$ 을 임의로 잡자.  $E_x(y)$ 는 곧  $\forall v_1 (v_1 = \bar{y} \rightarrow E_x)$ 이라는 표현식과 같다. 표현식 “ $\forall v_1 (v_1 =$ ”의 괴델 수를  $k$ 라고 하자.<sup>32</sup> 함의 기호 “ $\rightarrow$ ”는 괴델 수 8을 부여받았고 오른쪽 괄호 “ $)$ ”는 괴델 수 3을 부여받았고  $\bar{y}$ 의 괴델 수는  $13^y$ 이며  $E_x$ 의 괴델 수는  $x$ 이다 때문에,  $E_x(y)$ 의 괴델 수는  $k *_{13} 13^y *_{13} 8 *_{13} x *_{13} 3$ 이다. 즉, 그림으로 나타내면 다음과 같다:

$$\underbrace{\forall v_1}_{k}(v_1 = \underbrace{\bar{y}}_{13^y} \underbrace{\rightarrow}_{8} \underbrace{E_x}_{x} \underbrace{)}_{3}$$

결국,

$$(\forall x \in \mathbb{N}) (\forall y \in \mathbb{N}) [r(x, y) = k *_{13} 13^y *_{13} 8 *_{13} x *_{13} 3]$$

이므로, 관계  $r(x, y) = z$ 는,  $(\exists w \in \mathbb{N}) [w = 13^y \wedge z = k *_{13} w *_{13} 8 *_{13} x *_{13} 3]$ 으로 나타낼 수 있기 때문에, 명백하게 쌍술적이다. 따라서 다음 명제를 얻는다.

**Proposition 2.**  $r$ 은 쌍술적인 함수이다.  $\neg$

함수  $r$ 은 이 책에서 많이 튀어나올 것인데, 그것을  $\mathcal{L}_E$ 의 나타내기(representation) 함수라고 부를 것이다.

**Diagonalization.** 함수  $d : \mathbb{N} \rightarrow \mathbb{N}$ 를

$$d(x) := r(x, x) \quad \text{for } x \in \mathbb{N}$$

으로 정의하고 이것을 대각(diagonal)함수라고 부르겠다.  $r$ 이 쌍술적이었기 때문에  $d$  또한 쌍술적이다. 임의의  $n \in \mathbb{N}$ 에 대하여  $d(n) = g(E_n(n))$ 이다.  $\neg$

집합  $A \subseteq \mathbb{N}$ 에 대하여  $A^* := \{n \in \mathbb{N} : d(n) \in A\}$ 로 정의하자.<sup>33</sup>

**Lemma 1.**  $A$ 가 쌍술적이면  $A^*$ 도 그러하다.  $\neg$

*Proof.* 전제 조건을 가정하자.  $A^*$ 은  $(\exists y \in \mathbb{N}) [d(x) = y \wedge y \in A]$ 을 만족시키는 모든  $x \in \mathbb{N}$ 들의 집합이다. 대각함수  $d$ 는 쌍술적이므로 어떤 논리식  $D$ 가 존재하여

$$(\forall x \in \mathbb{N}) (\forall y \in \mathbb{N}) [D[v_1 := \bar{x}, v_2 := \bar{y}] \in \mathcal{T} \leftrightarrow d(x) = y]$$

이 성립한다. 한편,  $A$ 가 쌍술적이라고 가정하였으므로, 어떤 논리식  $F$ 가 존재하여

$$(\forall n \in \mathbb{N}) [F[v_1 := \bar{n}] \in \mathcal{T} \leftrightarrow n \in A]$$

<sup>32</sup>독자는 원한다면  $k$ 를 계산할 수 있다.

<sup>33</sup>즉,  $A^* = d^{-1}[A]$ 이다.

이 성립한다. 그러면  $A^*$ 는 논리식  $\exists v_2 (D \wedge F[v_1 := v_2])$ 에 의하여 표현되기 때문에 짠술적이다.<sup>34</sup> □

**Theorem 1.** 임의의  $A \subseteq \mathbb{N}$ 에 대하여  $A$ 가 짠술적이면  $A$ 에 대한 괴델 문장이 존재한다. ▶

*Proof.* 정말로 이것은 위의 보조정리와 1단원의 보조정리 D에 의하여 따라온다 – 밑의 주목을 보라. 그러나  $\mathcal{L}_E$ 이라는 특정 언어에서 반복하기 위하여,  $A$ 를 짠술적이라고 가정하자. 그러면  $A^*$ 은 위의 보조정리에 의하여 짠술적이다. 그러므로 어떤 술어  $H$ 가 존재하여  $A^*$ 를 표현한다.  $H$ 의 괴델 수를  $h$ 라 하자. 그러면

$$H(h) \in \mathcal{T} \iff h \in A^* \iff d(h) \in A$$

인데,  $d(h)$ 은 문장  $H(h)$ 의 괴델 수이므로,  $H(h)$ 가  $A^*$ 에 대한 괴델 문장이다. □

**Remarks.** 지금 우리가 다루는 언어  $\mathcal{L}_E$ 를 1단원의 추상적인 프레임워크에 적용하기에 앞서, 각각의 수  $n$ 에 대하여 각 표현식  $E$ 에 표현식  $E(n)$ 을 할당했던 함수  $\Phi : \mathcal{E} \times \mathbb{N} \rightarrow \mathcal{E}$ 를 떠올려보자. 언어  $\mathcal{L}_E$ 에서는  $\Phi(E, n) := \forall v_1 (v_1 = \bar{n} \rightarrow E)$ 로 취했다. 그러면 “술어”는 개체 변수  $v_1$ 만이 자유롭게 나타나는 논리식이 되어야 할 것이다. 그러면 정리 1에 쓰이는 보조정리는 1단원에 있던 조건  $G_1$ 이 체계  $\mathcal{L}_E$ 에서 적용됨을 말해준다. 그러므로, 정리 1은 1단원의 보조정리 D의 (b)의 특별한 경우이다. ▶

**Tarski's Theorem.** 임의의  $A \subseteq \mathbb{N}$ 에 대하여 논리식  $F$ 가  $A$ 를 표현하면  $\neg F$ 는  $\tilde{A}$ 를 표현하기 때문에, 모든 짠술적인 집합들의 모임은 명백히 여집합에 대하여 닫혀있다. 따라서  $\mathcal{L}_E$ 에서는 1단원에서의 조건  $G_1$ 과  $G_2$ 가 모두 성립한다. 그러므로, 1단원의 정리 T에 의하여  $\mathcal{L}_E$ 의 모든 참인 문장들의 괴델 수들의 집합  $T$ 는  $\mathcal{L}_E$ 에서 표현 가능하지 않다 – 즉, 그것은 짠술적이지 않다. ▶

$\mathcal{L}_E$ 의 특별한 경우에서 증명을 반복하자.  $\tilde{T}$ 에 대한 괴델 문장은 그것이 참일 때 그리고 그럴 때에만 거짓이어야 하기 때문에 존재하지 않는다. 그러나  $\tilde{T}$ 가 짠술적인 집합이었다면 정리 1에 의하여 그것에 대한 괴델 문장이 존재한다. 그러므로 집합  $\tilde{T}$ 은 짠술적일 수 없고  $T$ 도 짠술적이지 않다. 따라서 다음 정리를 얻는다.

**Theorem 2–Tarski's Theorem.** 모든 짠술적인 문장들 중 참인 문장의 괴델 수를 모은 집합은 짠술적이지 않다. ▶

다음 단원에서 우리는 산술을 위한 형식적인 공리 시스템으로 넘어갈 것이다. 표면적으로는 모든 참인 문장이 시스템에서 증명 될 수 있다는 것이 그럴 듯해 보이지만, 집합  $T$ 와는 다르게 체계에서 증명 가능한 모든 문장들의 괴델 수를 모은 집합은 짠술적이라는 것이 밝혀질 것이다. 그러므로, 정리 2에 의하여, 진실과 증명 가능성은 일치하지 않는다. 사실 정리 1을 사용하면 체계에서 참이지만 증명 가능하지 않은 문장을 보일 수 있다.

<sup>34</sup> 대안으로  $\forall v_2 (D \rightarrow F[v_1 := v_2])$ 가 있다.

**Exercise 6.**

- (1) 기저를 10으로 취한 괴델 넘버링을 이용하여, 우리가 방금 공부했던 방법으로, 모든 짹수들의 집합에 대한 괴델 문장을 하나 찾아라. 그것은 참인가 거짓인가?
- (2) 같은 것을 모든 홀수들의 집합에 대하여서도 하라.

위 두 물음에 대하여 답하여라.  $\dashv$

**Exercise 7.** 임의의 논리식  $F$ 와 임의의 수  $n$ 에 대하여  $F$ 에 자유롭게 나타나는 개체 변수가  $v_1$  뿐이고  $F$ 의 괴델 수가  $n$ 일 때  $f(n)$ 이  $F$ 가 표현하는 집합에 대한 괴델 문장의 괴델 수가 되게 하는 쌍술적인 함수  $f : \mathbb{N} \rightarrow \mathbb{N}$ 을 찾아라.<sup>35</sup>  $\dashv$

**Exercise 8. [어려움 주의!]** 임의의  $A \subseteq \mathbb{N}$ 과 임의의  $B \subseteq \mathbb{N}$ 에 대하여,  $A$ 와  $B$  모두 쌍술적이면 어떤 두 문장  $X$ 와  $Y$ 가 존재하여  $X \in \mathcal{T} \leftrightarrow g(Y) \in A$ 와  $Y \in \mathcal{T} \leftrightarrow g(X) \in B$ 가 성립함을 보여라. 이것은 상호 참조(cross-reference)의 한 예로 생각될 수 있다;  $X$ 를  $Y$ 의 괴델 수가  $A$ 에 속한다고 선언하는 문장으로 생각할 수 있고,  $Y$ 를  $X$ 의 괴델 수가  $B$ 에 속한다고 선언하는 문장으로 생각할 수 있다.  $\dashv$

**Solutions.**

1. (a)  $X$ 와  $Y$ 를 문장이라고 하자. 그러면  $(X \wedge Y)$ 는 문장  $\neg(X \rightarrow \neg Y)$ 의 약어이다.
  - (i)  $X$ 와  $Y$  모두 참인 경우를 고려하자. 이때  $\neg Y$ 는 거짓이고, 또한  $X \rightarrow \neg Y$ 는 거짓이다. 따라서  $\neg(X \rightarrow \neg Y)$ 는 참이고  $(X \wedge Y)$ 도 참이다.
  - (ii)  $X$ 는 참이지만  $Y$ 는 참이 아닌 경우를 고려하자. 이때  $\neg Y$ 는 참이고  $X \rightarrow \neg Y$ 도 참이다. 따라서  $\neg(X \rightarrow \neg Y)$ 는 거짓이고  $(X \wedge Y)$ 도 거짓이다.
  - (iii)  $X$ 는 참이 아니지만  $Y$ 는 참인 경우를 고려하자. 이때  $X \rightarrow \neg Y$ 는 참이다. 따라서  $\neg(X \rightarrow \neg Y)$ 는 거짓이고  $(X \wedge Y)$ 도 거짓이다.
  - (iv)  $X$ 와  $Y$  모두 참이 아닌 경우를 고려하자. 이때  $X \rightarrow \neg Y$ 는 참이다. 따라서  $\neg(X \rightarrow \neg Y)$ 는 거짓이고  $(X \wedge Y)$ 도 거짓이다.

그러므로 위의 네 경우로부터  $(X \wedge Y)$ 가 참일 때 그리고 그럴 때에만  $X$ 와  $Y$  모두 참임을 알 수 있다.

- (b)  $X$ 와  $Y$ 를 문장이라고 하자. 그러면  $(X \vee Y)$ 는 문장  $(\neg X \rightarrow Y)$ 의 약어이다.
  - (i)  $X$ 와  $Y$  모두 참인 경우를 고려하자.  $\neg X$ 가 거짓이므로  $(\neg X \rightarrow Y)$ 는 참이다. 따라서  $(X \vee Y)$ 도 참이다.
  - (ii)  $X$ 는 참이지만  $Y$ 는 참이 아닌 경우를 고려하자.  $\neg X$ 가 거짓이므로  $(\neg X \rightarrow Y)$ 는 참이다. 따라서  $(X \vee Y)$ 도 참이다.

<sup>35</sup> 그러한  $f$ 는 괴델라이저(Gödelizer)라고 적절하게 불릴 수 있을 것이다.

(iii)  $X$ 는 참이 아니지만  $Y$ 는 참인 경우를 고려하자.  $\neg X$ 가 참이고  $(\neg X \rightarrow Y)$ 도 참이므로,  $(X \vee Y)$ 도 참이다.

(iv)  $X$ 와  $Y$  모두 참이 아닌 경우를 고려하자. 그러면  $\neg X$ 가 참이지만  $(\neg X \rightarrow Y)$ 는 거짓이다. 따라서  $(X \vee Y)$ 도 거짓이다.

그러므로 위의 네 경우로부터  $(X \vee Y)$ 가 참일 때 그리고 그럴 때에만  $X$ 와  $Y$  중 적어도 하나가 참임을 알 수 있다.

(c)  $F$ 를 자유롭게 나타나는 개체 변수가  $v_i$  뿐인 논리식이라고 하자. 그러면  $\exists v_i F$ 는 문장  $\neg \forall v_i \neg F$ 의 약어이다. 먼저  $\neg \forall v_i \neg F$ 가 참이라고 하자. 그러면  $\forall v_i \neg F$ 는 거짓이다. 이때, 모든  $n \in \mathbb{N}$ 에 대하여  $F[v_i := \bar{n}]$ 이 거짓이라면,  $\forall v_i \neg F$ 는 거짓임에 모순이다. 따라서 어떤  $n \in \mathbb{N}$ 가 존재하여  $F[v_i := \bar{n}]$ 이 거짓이 아니어야 한다. 이제 어떤  $n \in \mathbb{N}$ 가 존재하여  $F[v_i := \bar{n}]$ 이 참이라고 하자. 그러면  $\forall v_i \neg F$ 은 거짓이다. 따라서  $\neg \forall v_i \neg F$ 는 거짓이다. 이상에서  $\exists v_i F$ 가 참일 때 그리고 그럴 때에만 어떤  $n \in \mathbb{N}$ 가 존재하여  $F[v_i := \bar{n}]$ 이 참임을 알 수 있다.

2. (a) 논리식  $\exists v_3 v_2 = (v_1 \times v_3) \circ ]$  “ $x$ 가  $y$ 의 약수이다”라는 관계를 표현하기 때문이다.

(b) 논리식  $(v_1 \neq \bar{1} \wedge (\forall v_2 \forall v_3 ((v_1 = v_2 \times v_3) \rightarrow (v_2 = \bar{1} \vee v_2 = v_1)))) \circ ]$  모든 소수들의 집합을 표현하기 때문이다.

3. 집합  $A \subseteq \mathbb{N}$ 과 함수  $f : \mathbb{N} \rightarrow \mathbb{N}$ 가 쌍술적이라고 하자. 그러면 자유롭게 나타나는 개체 변수가  $v_1$  뿐인 어떤 논리식  $F_A$ 와 자유롭게 나타나는 개체 변수가  $v_1$ 과  $v_2$  뿐인 어떤 논리식  $F_f$ 가 존재하여

$$(\forall n \in \mathbb{N}) [F_A[v_1 := \bar{n}] \in \mathcal{T} \leftrightarrow n \in A]$$

와

$$(\forall x \in \mathbb{N}) (\forall y \in \mathbb{N}) [F_f[v_1 := \bar{x}, v_2 := \bar{y}] \in \mathcal{T} \leftrightarrow f(x) = y]$$

가 성립한다. 이때 논리식  $\exists v_2 (F_f \wedge F_A[v_1 := v_2])$ 을 고려하자. 이 논리식은

$$(\exists y \in \mathbb{N}) [f(x) = y \wedge y \in A]$$

을 만족시키는 모든  $x \in \mathbb{N}$ 들의 집합을 표현한다. 그런데 그 집합은 결국  $f^{-1}[A]$ 를 표현하므로,  $f^{-1}[A]$ 는 쌍술적임을 알 수 있다.  $A$ 과  $f$ 가 산술적인 경우에는, 고려 중인 논리식이 산술적이게 되므로  $f^{-1}[A]$ 도 산술적임을 알 수 있다.

4. (a)  $f : \mathbb{N} \rightarrow \mathbb{N}$ 와  $g : \mathbb{N} \rightarrow \mathbb{N}$  모두 쌍술적이라고 하자. 그러면 개체 변수  $v_1$ 과  $v_2$ 만이

자유롭게 나타나는 어떤 두 논리식  $F_f$ 와  $F_g$ 가 존재하여

$$(\forall x \in \mathbb{N}) (\forall y \in \mathbb{N}) [F_f [v_1 := \bar{x}, v_2 := \bar{y}] \in \mathcal{T} \leftrightarrow f(x) = y]$$

와

$$(\forall x \in \mathbb{N}) (\forall y \in \mathbb{N}) [F_g [v_1 := \bar{x}, v_2 := \bar{y}] \in \mathcal{T} \leftrightarrow g(x) = y]$$

가 성립한다. 이제 논리식  $\forall v_3 (F_f [v_2 := v_3] \rightarrow F_g [v_1 := v_3])$ 를 고려하자. 이것은 관계

$$\forall z (f(x) = z \rightarrow g(z) = y)$$

을 표현하므로, 함수  $x \mapsto f(g(x)) : \mathbb{N} \rightarrow \mathbb{N}$ 도 쌍술적이다.

- (b)  $f : \mathbb{N} \rightarrow \mathbb{N}$ 과  $g : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$  모두 쌍술적이라고 하자. 그러면 개체 변수  $v_1$ 과  $v_2$ 만이 자유롭게 나타나는 어떤 논리식  $F_f$ 와 개체 변수  $v_1, v_2$ 와  $v_3$ 만이 자유롭게 나타나는 어떤 논리식  $F_g$ 가 존재하여

$$(\forall x \in \mathbb{N}) (\forall y \in \mathbb{N}) [F_f [v_1 := \bar{x}, v_2 := \bar{y}] \in \mathcal{T} \leftrightarrow f(x) = y]$$

와

$$(\forall x \in \mathbb{N}) (\forall y \in \mathbb{N}) (\forall z \in \mathbb{N}) [F_g [v_1 := \bar{x}, v_2 := \bar{y}, v_3 := \bar{z}] \in \mathcal{T} \leftrightarrow g(x, y) = z]$$

가 성립한다. 이때 세 논리식  $F_1, F_2$ 와  $F_3$ 를 개체 변수  $v_1, v_2$ 와  $v_3$ 만이 자유롭게 나타나도록 다음과 같이 정의하자:

$$F_1 := \forall v_4 (F_f [v_2 := v_4] \rightarrow F_g [v_1 := v_4])$$

$$F_2 := \forall v_4 (F_f [v_2 := v_4] [v_1 := v_2] \rightarrow F_g [v_2 := v_4])$$

$$F_3 := \forall v_4 (F_g [v_3 := v_4] \rightarrow F_f [v_1 := v_4] [v_2 := v_3])$$

그러면, 함수  $(x, y) \mapsto g(f(x), y) : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ 은  $F_1$ 에 의하여 표현되고 함수  $(x, y) \mapsto g(x, f(y)) : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ 은  $F_2$ 에 의하여 표현되고 함수  $(x, y) \mapsto f(g(x, y))$ 는  $F_3$ 에 의하여 표현되므로, 모두 쌍술적이다.

- (b)  $A$ 가 쌍술적이므로, 개체 변수  $v_1$ 만이 자유롭게 나타나는 논리식  $F_A$ 가 존재하여

$$(\forall n \in \mathbb{N}) [F_A [v_1 := \bar{n}] \leftrightarrow n \in A]$$

이 성립한다. 이때 논리식

$$((F_A [v_1 := v_2] \wedge v_1 < v_2) \wedge \forall v_3 ((F_A [v_1 := v_3] \wedge v_1 < v_3) \rightarrow v_2 \leq v_3))$$

을 고려하자. 이것은 관계  $(y \in A \wedge x < y) \wedge (\forall z \in \mathbb{N}) [(z \in A \wedge x < z) \rightarrow y \leq z]$ 를 표현하는데, 임의의  $x \in \mathbb{N}$ 와 임의의  $y \in \mathbb{N}$ 에 대하여 이 관계가  $x$ 와  $y$ 에 대하여 성립할 때 그리고 그럴 때에만  $R(x, y)$ 이므로, 관계  $R \subseteq \mathbb{N}^2$ 은 쌍술적이다.

- (c) 문제를 풀기에 앞서, 괴델 넘버링의 기저를 10으로 취했을 때의 대각 함수  $d$ 를 계산해 보자. 임의의  $x \in \mathbb{N}$ 에 대하여,  $\mathbf{g}(\bar{x}) = 10^x$ 임과

$$\mathbf{g}(v_1) = \mathbf{g}((v \circ)) = 2 *_{10} 6 *_{10} 5 *_{10} 3 = 2653$$

임에 주목하면,

$$\begin{aligned} d(x) &= r(x, x) \\ &= \mathbf{g}(\forall v_1 (v_1 = \bar{x} \rightarrow E_x)) \\ &= \mathbf{g}(\forall) *_{10} \mathbf{g}(v_1) *_{10} \mathbf{g}(() *_{10} \mathbf{g}(v_1) *_{10} \mathbf{g}(=) *_{10} \mathbf{g}(\bar{x}) *_{10} \mathbf{g}(\rightarrow) *_{10} \mathbf{g}(E_x) * \mathbf{g}()) \\ &= 899 *_{10} 2653 *_{10} 2 *_{10} 2653 *_{10} 10^x *_{10} 89 *_{10} x *_{10} 3 \\ &= 899265322653 *_{10} 10^x *_{10} 89 *_{10} x *_{10} 3 \end{aligned}$$

를 얻는다.  $k := 899265322653$ 로 놓자. 그러면, 관계  $k *_{10} 10^x *_{10} 89 *_{10} x *_{10} 3 = y$ 는 쌍술적이므로, 개체 변수  $v_1$ 과  $v_2$ 만이 자유롭게 나타나는 논리식  $D$ 가 존재하여

$$(\forall x \in \mathbb{N}) (\forall y \in \mathbb{N}) [D[v_1 := \bar{x}, v_2 := \bar{y}] \in \mathcal{T} \leftrightarrow k *_{10} 10^x *_{10} 89 *_{10} x *_{10} 3 = y]$$

가 성립한다.

- (1) 이제 모든 짝수들의 집합을  $A$ 라고 하자. 임의의  $x \in \mathbb{N}$ 에 대하여

$$\begin{aligned} x \in A^* &\iff (\exists y \in \mathbb{N}) [d(x) = y \wedge y \in A] \\ &\iff (\exists y \in \mathbb{N}) [k *_{10} 10^x *_{10} 89 *_{10} x *_{10} 3 = y \wedge (\exists z \in \mathbb{N}) [y = 2z]] \end{aligned}$$

이므로, 논리식

$$H_A := \exists v_2 (D \wedge \exists v_3 (v_2 = \bar{2} \times v_3))$$

은 집합  $A^*$ 를 표현한다. 이때  $H_A$ 의 괴델 수를  $h_A$ 라 하자. 그러면  $H_A(h_A)$ 은 원하던  $A$ 의 괴델 문장이 된다. 한편,  $k *_{10} 10^{h_A} *_{10} 89 *_{10} h_A *_{10} 3$ 가 짹수일 때 그리고 그럴 때에만 이 문장이 참인데, 이 수의 10진 표기의 마지막 숫자는 3이므로  $H_A(h_A)$ 는 거짓임을 알 수 있다.

(2) 이제 모든 홀수들의 집합을  $B$ 라고 하자. 임의의  $x \in \mathbb{N}$ 에 대하여

$$\begin{aligned} x \in B^* &\iff (\exists y \in \mathbb{N}) [d(x) = y \wedge y \in B] \\ &\iff (\exists y \in \mathbb{N}) [k *_{10} 10^x *_{10} 89 *_{10} x *_{10} 3 = y \wedge (\forall z \in \mathbb{N}) [y \neq 2z]] \end{aligned}$$

이므로, 논리식

$$H_B : \equiv \exists v_2 (D \wedge \forall v_3 \neg (v_2 = \bar{2} \times v_3))$$

은 집합  $B^*$ 를 표현한다. 이때  $H_B$ 의 괴델 수를  $h_B$ 라 하자. 그러면  $H_B(h_B)$ 은 원하던  $B$ 의 괴델 문장이 된다. 한편,  $k *_{10} 10^{h_B} *_{10} 89 *_{10} h_B *_{10} 3$ 가 홀수일 때 그리고 그럴 때에만 이 문장이 참인데, 이 수의 10진 표기의 마지막 숫자는 3이므로  $H_B(h_B)$ 은 참임을 알 수 있다.

5. 먼저 대각 함수  $d$ 는 쌍술적이므로,  $v_1$ 과  $v_2$ 만이 자유롭게 나타나는 개체 변수인 논리식  $D$ 가 존재하여

$$(\forall x \in \mathbb{N}) (\forall y \in \mathbb{N}) [D[v_1 := \bar{x}, v_2 := \bar{y}] \in \mathcal{T} \leftrightarrow d(x) = y]$$

가 성립한다. 임의의  $n \in \mathbb{N}$ 에 대하여, 표현식  $E_n$ 이 자유롭게 나타나는 개체 변수가  $v_1$  뿐인 논리식일 때, 논리식

$$F : \equiv \exists v_2 (D \wedge \forall v_1 (v_1 = v_2 \rightarrow E_n))$$

에 대하여  $F(g(F))$ 은  $E_n$ 이 표현하는 집합의 괴델 문장이다. 그러므로

$$f(n) := g\left(\forall v_1 \left(v_1 = \overline{g(F)} \rightarrow F\right)\right)$$

로 두자. 그러면  $f$ 는 쌍술적이므로 원하는 답을 얻었다.

6. 다음은 전한울님의 풀이임을 밝힙니다:

$A \subseteq \mathbb{N}$ 와  $B \subseteq \mathbb{N}$ 을 쌍술적이라고 하자. 그러면 개체 변수  $v_1$ 만이 자유롭게 나타나는 어떤 두 논리식  $F_A$ 와  $F_B$ 가 존재하여

$$(\forall n \in \mathbb{N}) [F_A[v_1 := \bar{n}] \in \mathcal{T} \leftrightarrow n \in A]$$

와

$$(\forall n \in \mathbb{N}) [F_B [v_1 := \bar{n}] \in \mathcal{T} \leftrightarrow n \in B]$$

가 성립한다. 각각의 표현식  $E$ 에 대하여 각  $x \in \mathbb{N}$ 와 각  $y \in \mathbb{N}$ 마다

$$E(x, y) := \forall v_1 \forall v_2 (v_1 = \bar{x} \rightarrow (v_2 = \bar{y} \rightarrow E))$$

라고 표기하자. 그리고 함수  $\text{fst} : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ 와  $\text{snd} : \mathbb{N} \times \mathbb{N} \rightarrow \mathbb{N}$ 을 각각

$$\text{fst}(x, y) := \mathbf{g}(E_x(x, y))$$

와

$$\text{snd}(x, y) := \mathbf{g}(E_y(x, y))$$

로 두자. 그런데  $\text{fst}$ 와  $\text{snd}$ 는 모두 쌍술적인 함수이므로, 개체 변수  $v_1, v_2$ 와  $v_3$ 만이 자유롭게 나타나는 어떤 두 논리식  $\text{Fst}$ 와  $\text{Snd}$ 가 존재하여

$$(\forall x \in \mathbb{N}) (\forall y \in \mathbb{N}) (\forall z \in \mathbb{N}) [\text{Fst} [v_1 := \bar{x}, v_2 := \bar{y}, v_3 := \bar{z}] \in \mathcal{T} \leftrightarrow \text{fst}(x, y) = z]$$

와

$$(\forall x \in \mathbb{N}) (\forall y \in \mathbb{N}) (\forall z \in \mathbb{N}) [\text{Snd} [v_1 := \bar{x}, v_2 := \bar{y}, v_3 := \bar{z}] \in \mathcal{T} \leftrightarrow \text{snd}(x, y) = z]$$

가 성립한다. 이제 논리식  $F_1$ 과  $F_2$ 를 각각

$$F_1 := \forall v_3 (\text{Snd} \rightarrow F_A [v_1 := v_3])$$

와

$$F_2 := \forall v_3 (\text{Fst} \rightarrow F_B [v_1 := v_3])$$

로 놓자. 그러면,  $F_1$ 과  $F_2$ 에는 개체 변수  $v_1$ 와  $v_2$ 만이 자유롭게 나타나며,

$$F_1 (\mathbf{g}(F_1), \mathbf{g}(F_2)) \in \mathcal{T} \iff F_A \left[ v_1 := \overline{\mathbf{g}(F_2(\mathbf{g}(F_1), \mathbf{g}(F_2)))} \right] \in \mathcal{T}$$

이고

$$F_2 (\mathbf{g}(F_1), \mathbf{g}(F_2)) \in \mathcal{T} \iff F_B \left[ v_1 := \overline{\mathbf{g}(F_1(\mathbf{g}(F_1), \mathbf{g}(F_2)))} \right] \in \mathcal{T}$$

이다. 이때

$$X := F_1(\mathbf{g}(F_1), \mathbf{g}(F_2))$$

와

$$Y := F_2(\mathbf{g}(F_1), \mathbf{g}(F_2))$$

를 취하자. 그러면,

$$\begin{aligned} X \in \mathcal{T} &\iff F_1(\mathbf{g}(F_1), \mathbf{g}(F_2)) \in \mathcal{T} \\ &\iff F_A \left[ v_1 := \overline{\mathbf{g}(F_2(\mathbf{g}(F_1), \mathbf{g}(F_2)))} \right] \in \mathcal{T} \\ &\iff \mathbf{g}(F_2(\mathbf{g}(F_1), \mathbf{g}(F_2))) \in A \\ &\iff \mathbf{g}(Y) \in A \end{aligned}$$

이고,

$$\begin{aligned} Y \in \mathcal{T} &\iff F_2(\mathbf{g}(F_1), \mathbf{g}(F_2)) \in \mathcal{T} \\ &\iff F_B \left[ v_1 := \overline{\mathbf{g}(F_1(\mathbf{g}(F_1), \mathbf{g}(F_2)))} \right] \in \mathcal{T} \\ &\iff \mathbf{g}(F_1(\mathbf{g}(F_1), \mathbf{g}(F_2))) \in B \\ &\iff \mathbf{g}(X) \in B \end{aligned}$$

이다. 이로써

$$X \in \mathcal{T} \leftrightarrow \mathbf{g}(Y) \in A$$

와

$$Y \in \mathcal{T} \leftrightarrow \mathbf{g}(X) \in B$$

가 성립함을 알 수 있다.

→

**Comments.** 이 장의 내용을 Coq으로 형식화해보자.

1. requirements:

```
From Coq.Bool Require Export Bool.
From Coq.micromega Require Export Lia.
From Coq.Lists Require Export List.
From Coq.Arith Require Export PeanoNat.
```

2. utility:

```

Import ListNotations.

Lemma forallb_true_iff {A : Type} (f : A -> bool) :
  forall xs : list A,
  forallb f xs = true <-> (forall x : A, In x xs -> f x = true).

Proof.
  intros xs; induction xs as [| x' xs].
  - simpl; constructor.
    + intros H x H0; contradiction.
    + tauto.
  - simpl; rewrite andb_true_iff; constructor.
    + intros H; destruct H as [H H0]; intros x H1; destruct H1.
      { subst; apply H.
      }
      { apply IHxs.
        - apply H0.
        - apply H1.
      }
    + intros H; constructor.
      { apply H; left; reflexivity.
      }
      { apply IHxs.
        intros x H0; apply H; right; apply H0.
      }
  Qed.

Definition fold_right_max_0 : list nat -> nat :=
  fold_right max 0

.

Lemma fold_right_max_0_in :
  forall ns : list nat,
  forall n : nat,
  In n ns ->
  fold_right_max_0 ns >= n.

Proof.
  unfold fold_right_max_0; intros ns; induction ns as [| n' ns].
  - simpl; lia.
  - simpl; intros n H; destruct H.
    + lia.
    + assert (fold_right max 0 ns >= n).
      { apply IHns.
        apply H.
      }

```

```

        }

lia.

Qed.

Lemma fold_right_max_0_app :
  forall ns1 : list nat,
  forall ns2 : list nat,
  fold_right_max_0 (ns1 ++ ns2) = max (fold_right_max_0 ns1) (
    fold_right_max_0 ns2).

Proof.
  unfold fold_right_max_0; intros ns1; induction ns1 as [|n1 ns1].
  - simpl; intros n; lia.
  - simpl; intros n; rewrite IHns1; lia.

Qed.

```

## 3. term, formula:

```

Definition vr : Set :=
  nat
  .

Inductive tm : Set :=
| ivar_tm : vr -> tm
| zero_tm : tm
| succ_tm : tm -> tm
| plus_tm : tm -> tm -> tm
| mult_tm : tm -> tm -> tm
| expo_tm : tm -> tm -> tm
  .

Inductive form : Set :=
| eqn_form : tm -> tm -> form
| leq_form : tm -> tm -> form
| neg_form : form -> form
| imp_form : form -> form -> form
| all_form : vr -> form -> form
  .

```

## 4. free occurrence:

```

Lemma vr_eq_dec :
  forall x1 : vr,
  forall x2 : vr,
  {x1 = x2} + {x1 <> x2}.

```

```

Proof.

apply Nat.eq_dec.

Qed.

Fixpoint occursFreeIn_tm (z : vr) (t : tm) : bool :=
  match t with
  | ivar_tm x => if vr_eq_dec z x then true else false
  | zero_tm => false
  | succ_tm t1 => occursFreeIn_tm z t1
  | plus_tm t1 t2 => occursFreeIn_tm z t1 || occursFreeIn_tm z t2
  | mult_tm t1 t2 => occursFreeIn_tm z t1 || occursFreeIn_tm z t2
  | expo_tm t1 t2 => occursFreeIn_tm z t1 || occursFreeIn_tm z t2
  end

.

Fixpoint occursFreeIn_form (z : vr) (f : form) : bool :=
  match f with
  | eqn_form t1 t2 => occursFreeIn_tm z t1 || occursFreeIn_tm z t2
  | leq_form t1 t2 => occursFreeIn_tm z t1 || occursFreeIn_tm z t2
  | neg_form f1 => occursFreeIn_form z f1
  | imp_form f1 f2 => occursFreeIn_form z f1 || occursFreeIn_form z f2
  | all_form x f1 => if vr_eq_dec z x then false else occursFreeIn_form
    z f1
  end

.

Fixpoint getFreeVars_tm (t : tm) : list vr :=
  match t with
  | ivar_tm x => [x]
  | zero_tm => []
  | succ_tm t1 => getFreeVars_tm t1
  | plus_tm t1 t2 => getFreeVars_tm t1 ++ getFreeVars_tm t2
  | mult_tm t1 t2 => getFreeVars_tm t1 ++ getFreeVars_tm t2
  | expo_tm t1 t2 => getFreeVars_tm t1 ++ getFreeVars_tm t2
  end

.

Theorem the_rule_of_getFreeVars_tm :
  forall t : tm,
  forall x : vr,
  In x (getFreeVars_tm t) <-> occursFreeIn_tm x t = true.

Proof.

intros t; induction t.
- intros vr; simpl; constructor.

```

```

+ intros H; destruct H.
{ subst.
  destruct (vr_eq_dec vr vr).
- reflexivity.
- contradiction.
}
{ contradiction.
}

+ intros H; destruct (vr_eq_dec vr v).
{ rewrite e; left; reflexivity.
}
{ inversion H.
}

- intros vr; simpl; constructor.
+ tauto.
+ intros H; inversion H.

- intros vr; simpl; apply IHt.

- intros vr; simpl; rewrite orb_true_iff; rewrite in_app_iff; rewrite
  IHt1; rewrite IHt2; reflexivity.

- intros vr; simpl; rewrite orb_true_iff; rewrite in_app_iff; rewrite
  IHt1; rewrite IHt2; reflexivity.

- intros vr; simpl; rewrite orb_true_iff; rewrite in_app_iff; rewrite
  IHt1; rewrite IHt2; reflexivity.

Qed.

Fixpoint getFreeVars_form (f : form) : list vr :=
match f with
| eqn_form t1 t2 => getFreeVars_tm t1 ++ getFreeVars_tm t2
| leq_form t1 t2 => getFreeVars_tm t1 ++ getFreeVars_tm t2
| neg_form f1 => getFreeVars_form f1
| imp_form f1 f2 => getFreeVars_form f1 ++ getFreeVars_form f2
| all_form x f1 => remove vr_eq_dec x (getFreeVars_form f1)
end

.

Theorem the_rule_of_getFreeVars_form :
  forall f : form,
  forall x : vr,
  In x (getFreeVars_form f) <-> occursFreeIn_form x f = true.

Proof.
  intros f; induction f.
- intros vr; simpl; rewrite orb_true_iff; rewrite in_app_iff; rewrite

```

```

the_rule_of_getFreeVars_tm; rewrite the_rule_of_getFreeVars_tm;
reflexivity.

- intros vr; simpl; rewrite orb_true_iff; rewrite in_app_iff; rewrite
the_rule_of_getFreeVars_tm; rewrite the_rule_of_getFreeVars_tm;
reflexivity.

- intros vr; simpl; apply IHf.

- intros vr; simpl; rewrite orb_true_iff; rewrite in_app_iff; rewrite
IHf1; rewrite IHf2; reflexivity.

- intros vr; simpl; constructor.

+ intros H.

  assert (In vr (getFreeVars_form f) /\ vr <> v).

  { apply (in_remove vr_eq_dec (getFreeVars_form f) vr v); apply H.
  }

  destruct H0 as [H0 H1]; destruct (vr_eq_dec vr v).

  { contradiction H1.
  }

  { apply IHf; apply H0.
  }

+ intros H; destruct (vr_eq_dec vr v).

{ inversion H.
}

{ apply in_in_remove.

- apply n.

- apply IHf; apply H.
}

```

Qed.

## 5. designation:

```

Definition value_assignment : Set :=
vr -> nat

.

Fixpoint eval_tm (va : value_assignment) (t : tm) : nat :=
match t with
| ivar_tm x => va x
| zero_tm => 0
| succ_tm t1 => S (eval_tm va t1)
| plus_tm t1 t2 => eval_tm va t1 + eval_tm va t2
| mult_tm t1 t2 => eval_tm va t1 * eval_tm va t2
| expo_tm t1 t2 => (eval_tm va t1)^(eval_tm va t2)
end

```

```

.
Lemma eval_tm_extensionality :
  forall t : tm,
  forall va1 : value_assignment,
  forall va2 : value_assignment,
  (forall x : vr, occursFreeIn_tm x t = true -> va1 x = va2 x) ->
  eval_tm va1 t = eval_tm va2 t.

Proof.
  intros t; induction t.
  - simpl; intros va1 va2 H; apply H; destruct (vr_eq_dec v v).
    { reflexivity. }
    { contradiction. }
  - simpl; intros va1 va2 H.
    reflexivity.
  - simpl; intros va1 va2 H.
    rewrite (IHt va1 va2).
    reflexivity.
    apply H.
  - simpl; intros va1 va2 H.
    rewrite (IHt1 va1 va2).
    rewrite (IHt2 va1 va2).
    reflexivity.
    intros x H0; apply H; apply orb_true_iff; tauto.
    intros x H0; apply H; apply orb_true_iff; tauto.
  - simpl; intros va1 va2 H.
    rewrite (IHt1 va1 va2).
    rewrite (IHt2 va1 va2).
    reflexivity.
    intros x H0; apply H; apply orb_true_iff; tauto.
    intros x H0; apply H; apply orb_true_iff; tauto.
  - simpl; intros va1 va2 H.
    rewrite (IHt1 va1 va2).
    rewrite (IHt2 va1 va2).
    reflexivity.
    intros x H0; apply H; apply orb_true_iff; tauto.
    intros x H0; apply H; apply orb_true_iff; tauto.

Qed.

Fixpoint eval_form (va : value_assignment) (f : form) : Prop :=

```

```

match f with
| eqn_form t1 t2 => eval_tm va t1 = eval_tm va t2
| leq_form t1 t2 => eval_tm va t1 <= eval_tm va t2
| neg_form f1 => ~ eval_form va f1
| imp_form f1 f2 => eval_form va f1 -> eval_form va f2
| all_form x f1 => forall n : nat, eval_form (fun z : vr => if
  vr_eq_dec x z then n else va z) f1
end

.

Lemma eval_form_extensionality :
forall f : form,
forall va1 : value_assignment,
forall va2 : value_assignment,
(forall x : vr, occursFreeIn_form x f = true -> va1 x = va2 x) ->
eval_form va1 f <-> eval_form va2 f.

Proof.
intros f; induction f.
- simpl; intros va1 va2 H.
  rewrite (eval_tm_extensionality t va1 va2).
  rewrite (eval_tm_extensionality t0 va1 va2).
  reflexivity.
  intros x H0; apply H; apply orb_true_iff; tauto.
  intros x H0; apply H; apply orb_true_iff; tauto.
- simpl; intros va1 va2 H.
  rewrite (eval_tm_extensionality t va1 va2).
  rewrite (eval_tm_extensionality t0 va1 va2).
  reflexivity.
  intros x H0; apply H; apply orb_true_iff; tauto.
  intros x H0; apply H; apply orb_true_iff; tauto.
- simpl; intros va1 va2 H.
  rewrite (IHf va1 va2).
  reflexivity.
  intros x H0; apply H; tauto.
- simpl; intros va1 va2 H.
  rewrite (IHf1 va1 va2).
  rewrite (IHf2 va1 va2).
  reflexivity.
  intros x H0; apply H; apply orb_true_iff; tauto.
  intros x H0; apply H; apply orb_true_iff; tauto.
- simpl; intros va1 va2 H.

```

```

cut (
    forall n : nat,
    eval_form (fun z : vr => if vr_eq_dec v z then n else va1 z) f <->
    eval_form (fun z : vr => if vr_eq_dec v z then n else va2 z)
    f
).
{ constructor.
- intros H1 n; apply H0; apply H1.
- intros H1 n; apply H0; apply H1.
}
intros n.
rewrite (IHf (fun z : vr => if vr_eq_dec v z then n else va1 z) (fun
z : vr => if vr_eq_dec v z then n else va2 z)).
reflexivity.
intros x H0; destruct (vr_eq_dec v x).
{ reflexivity.
}
{ apply H.
destruct (vr_eq_dec x v).
- contradiction n0; rewrite e; reflexivity.
- apply H0.
}
Qed.

Fixpoint make_numeral (n : nat) : tm :=
match n with
| 0 => zero_tm
| S n => succ_tm (make_numeral n)
end

.

Lemma eval_tm_make_numeral :
forall n : nat,
forall va : value_assignment,
eval_tm va (make_numeral n) = n.

Proof .
intros n; induction n.
- simpl; intros va; reflexivity.
- simpl; intros va; rewrite IHn; reflexivity.
Qed.

```

```

Definition substitution : Set :=
list (vr * tm)

```

```

.
Fixpoint substitute_vr (sigma : substitution) (x : vr) : tm :=
  match sigma with
  | [] => ivar_tm x
  | (x1, tm1) :: sigma' =>
    if vr_eq_dec x x1
    then tm1
    else substitute_vr sigma' x
  end

.
Fixpoint substitute_tm (sigma : substitution) (t : tm) : tm :=
  match t with
  | ivar_tm x => substitute_vr sigma x
  | zero_tm => zero_tm
  | succ_tm t1 => succ_tm (substitute_tm sigma t1)
  | plus_tm t1 t2 => plus_tm (substitute_tm sigma t1) (substitute_tm sigma t2)
  | mult_tm t1 t2 => mult_tm (substitute_tm sigma t1) (substitute_tm sigma t2)
  | expo_tm t1 t2 => expo_tm (substitute_tm sigma t1) (substitute_tm sigma t2)
  end

.
Theorem substitute_tm_preserves_eval_tm :
  forall t : tm,
  forall sigma : substitution,
  forall va : value_assignment,
  eval_tm (fun z : vr => eval_tm va (substitute_vr sigma z)) t = eval_tm
    va (substitute_tm sigma t).

Proof.
  intros t; induction t.
  - intros sigma va; simpl; reflexivity.
  - intros sigma va; simpl; reflexivity.
  - intros sigma va; simpl; rewrite IHt; reflexivity.
  - intros sigma va; simpl; rewrite IHt1; rewrite IHt2; reflexivity.
  - intros sigma va; simpl; rewrite IHt1; rewrite IHt2; reflexivity.
  - intros sigma va; simpl; rewrite IHt1; rewrite IHt2; reflexivity.

Qed.

Definition getMaxNumOfFreeVars_tm (t : tm) : vr :=
  fold_right_max_0 (getFreeVars_tm t)

```

```

.
Lemma getMaxNumOfFreeVars_tm_lt :
  forall t : tm,
  forall x : vr,
  getMaxNumOfFreeVars_tm t < x ->
  occursFreeIn_tm x t = false.

Proof.
unfold getMaxNumOfFreeVars_tm; intros t; induction t.
- simpl; intros x H; destruct (vr_eq_dec x v).
  + lia.
  + reflexivity.
- simpl; intros x H; tauto.
- simpl; intros x H; apply IHt; apply H.
- simpl; intros x H; apply orb_false_iff; rewrite fold_right_max_0_app
  in H; constructor.
  + apply IHt1; lia.
  + apply IHt2; lia.
- simpl; intros x H; apply orb_false_iff; rewrite fold_right_max_0_app
  in H; constructor.
  + apply IHt1; lia.
  + apply IHt2; lia.
- simpl; intros x H; apply orb_false_iff; rewrite fold_right_max_0_app
  in H; constructor.
  + apply IHt1; lia.
  + apply IHt2; lia.

Qed.

Definition isFreshVarIn_substitute_tm (sigma : substitution) (z : vr) (t
  : tm) : Prop :=
forallb (fun x : vr => negb (occursFreeIn_tm z (substitute_vr sigma x)
  )) (getFreeVars_tm t) = true

.

Definition isFreshVarIn_substitute_form (sigma : substitution) (z : vr)
  (f : form) : Prop :=
forallb (fun x : vr => negb (occursFreeIn_tm z (substitute_vr sigma x)
  )) (getFreeVars_form f) = true

.

Definition chi (sigma : substitution) (f : form) : vr :=
S (fold_right_max_0 (map (fun x : vr => getMaxNumOfFreeVars_tm (
  substitute_vr sigma x)) (getFreeVars_form f)))
.
```

```

Theorem the_rule_of_chi :
  forall f : form,
  forall sigma : substitution,
  isFreshVarIn_substitute_form sigma (chi sigma f) f.

Proof.
  assert ( claim1 :
    forall sigma : substitution,
    forall f : form,
    forall x : vr,
    occursFreeIn_form x f = true ->
    occursFreeIn_tm (chi sigma f) (substitute_vr sigma x) = false
  ).

  { intros sigma f x H.
    assert (getMaxNumOfFreeVars_tm (substitute_vr sigma x) < chi sigma f
      ).

    { unfold chi; unfold fold_right_max_0.
      cut (fold_right max 0 (map (fun z : vr => getMaxNumOfFreeVars_tm (
        substitute_vr sigma z)) (getFreeVars_form f)) >=
        getMaxNumOfFreeVars_tm (substitute_vr sigma x)).
      { lia.
      }
      rewrite <- the_rule_of_getFreeVars_form in H.
      apply fold_right_max_0_in; apply in_map_iff.
      exists x; constructor.
      - reflexivity.
      - apply H.
    }

    apply getMaxNumOfFreeVars_tm_lt; apply H0.
  }

  unfold isFreshVarIn_substitute_form; intros f sigma; apply
  forallb_true_if.
  intros x H; apply negb_true_if; apply claim1; apply
  the_rule_of_getFreeVars_form; apply H.

Qed.

Fixpoint substitute_form (sigma : substitution) (f : form) : form :=
  match f with
  | eqn_form t1 t2 => eqn_form (substitute_tm sigma t1) (substitute_tm
    sigma t2)
  | leq_form t1 t2 => leq_form (substitute_tm sigma t1) (substitute_tm
    sigma t2)

```

```

| neg_form f1 => neg_form (substitute_form sigma f1)
| imp_form f1 f2 => imp_form (substitute_form sigma f1) (
    substitute_form sigma f2)
| all_form x f1 =>
    let z : vr := chi sigma f in
    all_form z (substitute_form ((x, ivar_tm z) :: sigma) f1)
end

.

Theorem substitute_form_preserves_eval_form :
  forall f : form,
  forall sigma : substitution,
  forall va : value_assignment,
  eval_form (fun z : vr => eval_tm va (substitute_vr sigma z)) f <->
    eval_form va (substitute_form sigma f).

Proof.
  intros f; induction f.
  - intros sigma va; simpl.
    rewrite substitute_tm_preserves_eval_tm; rewrite
      substitute_tm_preserves_eval_tm; reflexivity.
  - intros sigma va; simpl.
    rewrite substitute_tm_preserves_eval_tm; rewrite
      substitute_tm_preserves_eval_tm; reflexivity.
  - intros sigma va; simpl.
    rewrite IHf; reflexivity.
  - intros sigma va; simpl.
    rewrite IHf1; rewrite IHf2; reflexivity.
  - intros sigma va. simpl.
    cut (
      forall n : nat,
      eval_form (fun z : vr => if vr_eq_dec v z then n else eval_tm va (
        substitute_vr sigma z)) f <-> eval_form (fun z : vr => if
        vr_eq_dec (chi sigma (all_form v f)) z then n else va z) (
        substitute_form ((v, ivar_tm (chi sigma (all_form v f))) :: sigma) f)
    ).  

    { intros H; constructor.
      - intros H0 n; apply H; apply H0.
      - intros H0 n; apply H; apply H0.
    }
    intros n; rewrite <- (IHf ((v, ivar_tm (chi sigma (all_form v f))))
```

```

    :: sigma) (fun z : vr => if vr_eq_dec (chi sigma (all_form v f))
      z then n else va z)); apply eval_form_extensionality.

intros x H; simpl; destruct (vr_eq_dec v x).

{ destruct (vr_eq_dec x v).
  - simpl; destruct (vr_eq_dec (chi sigma (all_form v f)) (chi sigma
    (all_form v f))).
    + reflexivity.
    + contradiction.
  - contradiction n0; rewrite e; reflexivity.
}

{ destruct (vr_eq_dec x v).
  - contradiction n0; rewrite e; reflexivity.
  - apply eval_tm_extensionality.

intros x' H0; destruct (vr_eq_dec (chi sigma (all_form v f)) x')

.

{ subst.

  assert (isFreshVarIn_substitute_form sigma (chi sigma (
    all_form v f)) (all_form v f)).
    apply the_rule_of_chi.

  unfold isFreshVarIn_substitute_form in H1; rewrite
    forallb_true_iff in H1.

  assert (negb (occursFreeIn_tm (chi sigma (all_form v f)) (
    substitute_vr sigma x)) = true).

{ apply H1; apply the_rule_of_getFreeVars_form; simpl;
  destruct (vr_eq_dec x v).
  - contradiction.
  - tauto.
}

  rewrite H0 in H2; inversion H2.

}

{ reflexivity.

}

Qed.

```

### 3 제곱 연산이 있는 페아노 산술의 불완전성

#### 3.1 P.E. 공리계

##### §1. P.E. 공리계

이제 제곱 연산이 있는 페아노 산술(Peano Arithmetic with Exponentiation)이라고 불리는 공리계로 넘어갈 것인데, 이 형식 체계를 약어로 “P.E.”라고 할 것이다. 공리(axiom)라고 불리는, 정확한 특정 논리식들을 잡고; 이미 증명된 [정확한 논리식]들로부터 새로운 [정확한 논리식]을 증명할 수 있게 하는 두 추론 규칙(inference rule)을 제공한다. 공리의 개수는 무한히 많지만, 각 공리는 쉽게 인식할 수 있는 열아홉 가지의 꼴 중 하나이다; 이러한 꼴들을 공리꼴(axiom schemes)라고 한다. 19가지의 공리꼴을 4개의 그룹으로 분류하는 것은 편리하다. (cf. 공리꼴이 소개된 다음의 논의) 그룹 I와 II는 이른바 논리적 공리(logical axioms)라고 하는데, Tarski[1965]을 기초로 한 Kalish, Montague[1965]의 등호가 있는 일차논리의 적절한 형식화의 기둥이 된다. 그룹 III와 그룹 IV는 산술(arithmetic) 공리라고 불린다.

이 공리꼴들을 보일 때에,  $F, G, H$ 는 임의의 논리식이며,  $v_i, v_j$ 는 임의의 변수이며,  $t$ 는 임의의 항이다. 예를 들어, 첫 번째 공리꼴  $L_1$ 은 임의의 논리식  $F$ 와  $G$ 에 대하여 논리식

$$(F \rightarrow (G \rightarrow F))$$

를 공리 중 하나로 둔다는 걸 의미하고; 공리꼴  $L_4$ 는 임의의 변수  $v_i$ 와 임의의 논리식  $F$ 와  $G$ 에 대하여 논리식

$$(\forall v_i (F \rightarrow G) \rightarrow (\forall v_i F \rightarrow \forall v_i G))$$

를 공리 중 하나로 둔다는 것을 의미한다.

##### Group I–Axiom Schemes for Propositional Logic

$$L_1 : (F \rightarrow (G \rightarrow F))$$

$$L_2 : ((F \rightarrow (G \rightarrow H)) \rightarrow ((F \rightarrow H) \rightarrow (G \rightarrow H)))$$

$$L_3 : ((\neg F \rightarrow \neg G) \rightarrow (G \rightarrow F))$$

##### Group II–Additional Axiom Schemes for First-Order Logic with Identity

$$L_4 : (\forall v_i (F \rightarrow G) \rightarrow (\forall v_i F \rightarrow \forall v_i G))$$

$$L_5 : (F \rightarrow \forall v_i F) \text{ (단, } v_i \text{는 } F \text{에서 자유롭게 나타나지 않아야 함)}$$

$$L_6 : \exists v_i v_i = t \text{ (단, } v_i \text{는 } t \text{에 자유롭게 나타나지 않아야 함)}$$

$L_7 : (v_i = t \rightarrow (X_1 v_i X_2 \rightarrow X_1 t X_2))$  (단,  $X_1$ 과  $X_2$ 는  $X_1 v_i X_2$ 가 원자 논리식이 되게 하는 임의의 표현식임)<sup>36</sup>

### Group III–Eleven Axiom Schemes Having Only One Axiom Apiece

$$N_1 : (v'_1 = v'_2 \rightarrow v_1 = v_2)$$

$$N_2 : \neg \bar{0} = v'_i$$

$$N_3 : (v_1 + \bar{0}) = v_1$$

$$N_4 : (v_1 + v'_2) = (v_1 + v_2)'$$

$$N_5 : (v_1 \times \bar{0}) = \bar{0}$$

$$N_6 : (v_1 \times v'_2) = v_1 \times v_2 + v_1$$

$$N_7 : (v_1 \leq \bar{0} \leftrightarrow v_1 = \bar{0})$$

$$N_8 : (v_1 \leq v'_2 \leftrightarrow (v_1 = v_2 \vee v_1 = v'_2))$$

$$N_9 : (v_1 \leq v_2 \vee v_2 \leq v_1)$$

$$N_{10} : (v_1 \mathbf{E} \bar{0}) = \bar{1}$$

$$N_{11} : (v_1 \mathbf{E} v'_2) = ((v_1 \mathbf{E} v_2) \times v_2)$$

**Group IV** 이 그룹에는 오직 하나의 공리꼴만이 있다 – 수학적 귀납법의 공리꼴 – 그러나 그 것은  $v_1$ 만이 자유롭게 나타나는 논리식  $F$ 마다 무한히 많은 공리를 준다. 이 공리꼴을 나타내기 위하여,  $F$ 를  $v_1$ 만이 자유롭게 나타나는 논리식이라고 하자.  $G$ 를

$$\forall v_i (v_i = v'_1 \rightarrow \forall v_1 (v_1 = v_i \rightarrow F))$$

꼴의 아무 논리식이라고 하되, 여기서  $v_i$ 는  $F$ 에 자유롭게 나타나지 않는다고 하자.<sup>37</sup> 이때,

$$N_{12} : (F(0) \rightarrow (\forall v_1 (F \rightarrow G) \rightarrow \forall v_1 F))$$

가 수학적 귀납법의 공리꼴이다.

---

<sup>36</sup>이 공리꼴의 대안으로는  $(v_i = t \rightarrow (Y_1 \rightarrow Y_2))$ 가 있다. 여기서  $Y_1$ 은 원자 논리식이고,  $Y_2$ 는  $Y_1$ 에서  $v_i$ 의 나타남 중 아무거나 하나를  $t$ 로 교체하여 얻은 식이다.

<sup>37</sup>이러한 꼴의 논리식은  $F[v_1 := v'_1]$ 와 동등하다 – 즉,  $F$ 에서의  $v_1$ 의 모든 자유 나타남들을  $v'_1$ 로 교체한 결과와 같다.