



# 인공지능 논리 입문

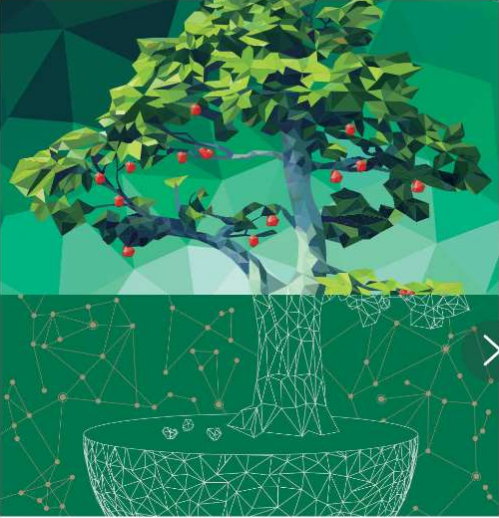
ZEROPAGE 33기 정현승



# 인공지능 논리 입문 - 명제논리

2023.08.21 OMS

안녕하세요, 저는 오늘 수리논리 중 명제논리에 대해 발표할 Zeropage 33기 정현승입니다.



**인공지능 논리 입문**  
수리논리로 이해하는 AI논리  
이은정 지음

한빛아카데미

서지상세

자료유형 국내서단행본

서명책임사항 **인공지능 논리 입문 : 수리논리로 이해하는 AI논리 / 이은정 지음**

개인저자 이은정 李銀貞, 1965-

발행사항 서울 : 한빛아카데미, 2021

형태사항 260 p. : 천연색삽화 ; 26 cm

CC BY : Naver

알라딘

▼ 더보기

개인보관함 ▼ 임시보관함 ▼ 내보내기 ▼

소장정보 가상서가

서울학술정보원

등록번호	소장위치	청구기호	위치	상태	서비스
C1400248	단행본(2층 대출자료실)	006.3 이은정인	서가번호 011-C	대출중 ~ 2023.08.30	
C1400249	보존서고	006.3 이은정인 c.2		대출가능	보
보존번호 U0059603					
C1419342	단행본(2층 대출자료실)	006.3 이은정인 c.3	서가번호 011-C	대출가능	무

(2023.08.21 OMS)시작하기에 앞서 이 내용은 제가 방학 동안에 개인적으로 공부하려고 읽은 책 "인공지능 논리 입문"의 내용을 정리한 것입니다. 저는 이걸 학교 도서관에서 빌려서 읽었고, 다른 분들도 필요하시다면 도서관에서 책 빌려서 읽어 보시면 됩니다. 다만 책 이름이 "인공지능 논리 입문"인데, 정작 저는 이 책에 나온 인공지능과 논리의 연결 부분을 제대로 이해하지 못했습니다. 근데 아는 사람에게 여쭙어보니 이걸 연결하는 게 부적절하다고 하더라고요.. 네.. 사기당했어요.. 그래도 제가 다룰 부분이 이거밖에 없고 의미가 전혀 없는 건 아닌 거 같아서 다루긴 하겠습니다. 혹시 책에서 말하는 인공지능과 연관성을 직접 알아보고 싶으신 분들은 책 빌려서 읽어보시기 바랍니다. 또한 저는 학교를 아직 1학년 1학기만 다녀본 사람이라 각 과목에서 뭐를 배우는지는 잘 모르고 있는 상태입니다. 이 내용은 학교 커리큘럼에 없다고 생각하고 가져왔는데 다 알고 계신 내용이라면 죄송하다는 말씀을 드립니다.

(2024.02.07 OMS)시작하기에 앞서 이 내용은 제가 2023년 여름방학 동안에 개인적으로 공부하려고 읽은 책 "인공지능 논리 입문"의 내용을 정리한 것입니다. 네 지금이 아니에요.. 지금도 읽고 있는 책이 있기는 하지만, 아직 이걸 발표할 정도는 안 되는 것 같아 이전 방학 것을 가져왔습니다. 네... 사진도 저번에 찍은 거 그대로입니다. 저는 이걸 학교 도서관에서 빌려서 읽었고, 다른 분들도 필요하시다면 도서관에서 책 빌려서 읽어 보시면 됩니다. 다만 책 이름이 "인공지능 논리 입문"인데, 정작 다른 사람에게 여쭙어보니 인공지능과 논리를 연결하는 게 부적절하다고 하더라고요.. 사기를 당한 것 같지만, 그래도 제가 다룰 부분이 이거밖에 없고, 이번 술어논리 정리하면서 보니 책 외적으로 둘의 연결에 대한 무언가를 찾은 것 같아서 다루겠습니다. 혹시 책에서 직접적으로 말하는 인공지능과 연관성을 직접 알아보고 싶으신 분들은 책 빌려서 읽어보시기 바랍니다.

## 기본 연산자

- $\wedge$  : AND
- $\vee$  : OR
- $\sim$  : NOT

원래 다른 개념을 설명하기에 앞서서 '명제'라는 단어의 개념을 알아야 하지만, 이 개념은 대충이라도 다 아실 거라 생각합니다. 그래서 슬라이드도 할당을 안 해 놨는데요, 혹시나 모르는 분이 있을 수 있으니 책에 쓰여 있는 정의를 읽어드리자면, "명제는 참 또는 거짓을 구별할 수 있는 객관적 상태가 포함된 문장이다."라고 되어 있습니다. 먼저 기본 연산자입니다. 기본연산자는 AND, OR, NOT인데 다 알고 계시겠죠. 넘어가겠습니다.

## 조건연산자

- $\rightarrow$  : if (만약 ~이면 ~이다)
- $\leftarrow$  : only if (if에서 순서가 바뀐 것)
- $\leftrightarrow$  : if and only if (둘을 합친 것)

$p$	$q$	$p \rightarrow q$
T	T	T
T	F	F
F	T	T
F	F	T

공허한 참  
(vacant true)

다음으로 조건연산자입니다. 여기 표현들은 알고 계실 수도 있고, 몰라도 직관적이어서 이해가 어렵지 않으니 넘어가겠습니다. 오른쪽은 두 명제가 각각 True와 False일 때, 조건연산자를 사용하여 이 두 명제를 연결한 명제가 어떨 때 True이고 어떨 때 False인지를 나타낸 것입니다. 이때 주의할 점은 앞에 연결된 이  $p$ 가 False이면  $q$ 가 어떻든 이 명제는 True가 됩니다. 따라서 이 조건연산자를 이용한 명제가 False가 되는 경우는  $p$ 가 True인데  $q$ 가 거짓이 되는 경우 한가지 밖에는 없고요, 이때  $p$ 가 False여서 이 명제가 True가 될 때 이를 공허한 참, vacant true라고 부릅니다.

## 계산 순서

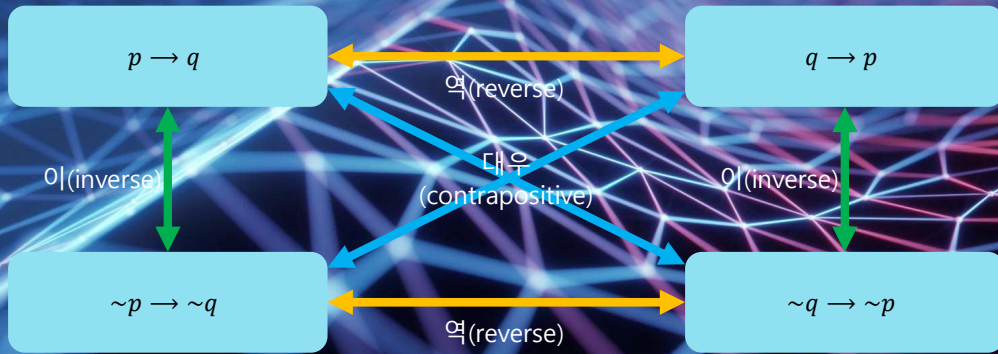
- ~, ^, V, →, ↔

- OR보다 AND를 먼저 계산함

```
*IDLE Shell 3.10.2*
File Edit Shell Debug Options Window Help
Python 3.10.2 (tags/v3.10.2:a58ebcc, Jan 17 2022, 14:12:15) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> True or (True and False)
True
>>> Python도 원래 이랬나...]
```

이 연산자들의 계산 순서는 다음과 같습니다. 왼쪽부터 먼저 계산합니다. 이때 주의할 점은 AND와 OR의 우선도가 같은 게 아니라는 점입니다. OR보다 AND를 먼저 계산합니다. 예를 들어서, 이걸 PyCharm 켜기 귀찮아서 Python IDLE에 실험해본 건데요, 이런 걸 계산하면 뒤의 and로 묶인 부분이 먼저 계산되어 전체 결과는 True가 나오게 됩니다... 저는 이전까지 AND와 OR의 우선도가 같은 줄 알았는데 아니더라고요...

## 역, 이, 대우



- 대우명제는 원래 명제와 항상 진리값이 같다
- 즉 원래 명제가 True였으면 그 대우명제도 True, 원래 False였으면 그 대우명제도 False

그 다음으로는 조건연산자의 역, 이, 그리고 대우입니다. 아직 제가 1학년이라 그런지 저는 고1때 이것을 배운 게 기억나는데 다른 분들은 어떠실 지 모르니 설명하겠습니다. 역은 조건연산자에서 명제의 순서를 바꾼 것이고요, 이는 두 명제에 NOT을 붙인 겁니다. 그리고 대우는 이 두개를 동시에 한 것입니다. 여기서 포인트는 대우명제는 원래 명제와 항상 진리값이 같다는 겁니다. 그러니까 원래 명제가 True였으면 이에 대우를 취한 것도 True, 원래 명제가 False였으면 이에 대우를 취한 것도 False입니다.

## 역, 이, 대우

- 대우명제는 원래 명제와 항상 진리값이 같다
- 즉 원래 명제가 True였으면 그 대우명제도 True, 원래 False였으면 그 대우명제도 False

$p$	$q$	$p \rightarrow q$	$\sim q$	$\sim p$	$\sim q \rightarrow \sim p$
T	T	T	F	F	T
T	F	F	T	F	F
F	T	T	F	T	T
F	F	T	T	T	T

공허한 참  
(vacant true)

- 동치: 두 조건식이 단순명제의 모든 True, False의 경우에 대해 항상 같은 진위값을 가짐
- 대우를 동치를 이용해 표현하면  $p \rightarrow q \equiv \sim q \rightarrow \sim p$

왜 그런지는 직접 경우의 수 다 따져서 구해보도록 하겠습니다. 4개의 모든 경우에서 이 초록색으로 해놓은 부분이 전부 같음을 알 수 있습니다. 그리고 잘 보면 둘이 True로 같은 경우는 공허한 참이 하나씩 끼어 있다는 점도 볼 수 있습니다. 공허한 참이 아니라 공허한 거짓(?)이었다면 이건 성립하지 않았겠죠. 이때 이렇게 대우처럼 두 식이 모든 경우에서 항상 같은 값을 가지는 경우 이를 '동치'라고 합니다. 대우를 동치를 이용해 표현하면 이렇게 되고요. 동치를 표현하는 기호는 이것입니다. (삼각형의 합동이 생각하시는 분이 있으실지는 모르겠네요. 그때 쓰던 기호입니다.)



## 동치규칙

규칙	식
모순규칙	$p \wedge \sim p \equiv F$
항진규칙	$p \vee \sim p \equiv T$
동일규칙	$T \wedge p \equiv p, F \vee p \equiv p$
우위규칙	$T \vee p \equiv T, F \wedge p \equiv F$
이중부정규칙	$\sim(\sim p) \equiv p$
역등규칙	$p \wedge p \equiv p, p \vee p \equiv p$
교환규칙	$p \wedge q \equiv q \wedge p$
배분규칙	$(p \wedge q) \vee r \equiv (p \vee r) \wedge (q \vee r)$
대우규칙	$p \rightarrow q \equiv \sim q \rightarrow \sim p$

규칙	식
드모르간 규칙	$\sim(p \wedge q) \equiv \sim p \vee \sim q$ $\sim(p \vee q) \equiv \sim p \wedge \sim q$

p	q	$p \vee q$	$\sim(p \vee q)$	$\sim p$	$\sim q$	$\sim p \wedge \sim q$
T	T	T	F	F	F	F
T	F	T	F	F	T	F
F	T	T	F	T	F	F
F	F	F	T	T	T	T

이렇게 항상 동치가 되는 경우가 몇 가지 있습니다. 이를 '동치규칙'이라고 합니다. 왼쪽에 있는 규칙들은 대우명제는 방금 확인했고요, 나머지 8개는 상식 수준이니 넘어가도록 하겠습니다. 혹시나 배분규칙을 모르시는 분이 계실지 모르겠는데 그냥... 눈으로 보고 확인하시면 됩니다. 교환규칙, 배분규칙에 있는 연산자는 모두 상하반전해도 동치가 성립합니다. 식을 하나만 써 놨을 뿐이에요. 그리고 드모르간 규칙인데요, 식은 이겁니다. NOT을 두 명제에 붙이고 기호는 상하반전하면 됩니다. 증명은 다른 거 없고 대우처럼 다 따져보면 됩니다... 기호 바꾸면 이렇게요.

## 동치규칙

규칙	식
흡수규칙	$p \vee (p \wedge q) \equiv p$

$p$	$q$	$p$	$\vee$	$(p \wedge q)$	$p \vee (p \wedge q)$
T	T	T	T	F	T
T	F	T	T	F	T
F	T	F	F	F	F
F	F	F	F	F	F

규칙	식
변환규칙	$p \rightarrow q \equiv \sim p \vee q$

$p$	$q$	$p \rightarrow q$
T	T	T
T	F	F
F	T	T
F	F	T

공허한 참  
(vacant true)

그 다음 흡수규칙입니다. 식은 이거인데요, p가 True이면 이 OR에서 전체가 True가 되고, p가 False이면 앞의 p가 False, 괄호 안도 p가 False라서 AND로 묶여 False가 되어 전체로도 False가 되는 겁니다. 이것 아까처럼 표로 정리하면 이렇게 됩니다. 그리고 마지막으로 변환규칙인데요, 앞에서 조건연산자 설명할 때 공허한 참 때문에 이게 False가 되는 때는 한 가지밖에 없다고 말씀을 드렸잖아요? 그거를 OR을 써서 나타낸 겁니다.

## 동치규칙 사용예시

$(X \rightarrow Y) \vee (X \rightarrow \sim Y) \equiv T$	$X \wedge Y \rightarrow Z \equiv (X \rightarrow Z) \vee (Y \rightarrow Z)$
$(X \rightarrow Y) \vee (X \rightarrow \sim Y)$ $\equiv (\sim X \vee Y) \vee (\sim X \vee \sim Y)$ (변환규칙) $\equiv (\sim X \vee \sim X) \vee (Y \vee \sim Y)$ (교환규칙) $\equiv \sim X \vee T$ (멱등규칙, 항진규칙) $\equiv T$ (우위규칙)	$X \wedge Y \rightarrow Z$ $\equiv \sim(X \wedge Y) \vee Z$ (변환규칙) $\equiv \sim X \vee \sim Y \vee Z$ (드모르간 규칙) $\equiv (\sim X \vee Z) \vee (\sim Y \vee Z)$ (멱등규칙) $\equiv (X \rightarrow Z) \vee (Y \rightarrow Z)$ (변환규칙)

그냥 말로만 하면 이해가 안 될 수 있으니 책에 있는 예제를 가지고 와서 풀어보겠습니다. 왼쪽을 먼저 풀면 먼저 변환규칙으로 조건연산자를 지워줍니다. 그러면 OR밖에 안 남네요? 순서 바꿀 수 있으니 이렇게 묶어줍니다. 앞에 괄호는 같은 명제이니 멱등규칙을 적용해 이렇게 써주고요, 뒤 괄호는 둘 중 하나는 참이고 하나는 거짓일테니 항진규칙을 적용해 True로 쓸 수 있습니다. 마지막으로 True를 OR로 다른 명제와 묶으면 그것은 또 True이니 우위규칙을 적용시키면 이걸 항상 True임을 알 수 있습니다. 오른쪽을 풀면, 먼저 방금 전과 같이 조건연산자부터 지워줍니다. 괄호로 묶인 명제에 NOT이 달려 있으니 드모르간 규칙을 이용해 괄호를 풀어줍니다. 그러면 아까처럼 전부 OR만 남고요, 오른쪽의 형태를 만들어야 하니 멱등규칙을 사용해 Z를 하나 더 넣어줍니다. 그냥 Z나 Z 또는 Z나 같으니까요. 이렇게 묶어주면 다시 변환규칙을 사용해 조건연산자로 돌려놓으면 끝입니다.

## 주장

- 가설과 결론으로 이루어진 명제의 연속
- 주장을 식으로 바꾸면 가설→결론
- 가설이 참일 때 결론이 항상 참이면 타당하다고 한다

$p$	$q$	$p \rightarrow q$
T	T	T
T	F	F
F	T	T
F	F	T

이런 경우가 없으면  
타당한 주장  
알 필요  
없음

다음은 주장입니다. 주장은 “가설과 결론으로 이루어진 명제의 연속”으로 정의되며, 이를 식으로 나타내면 계속 봐 왔던 그 식이 나옵니다. 이때 그냥 조건절과 다른 점은, 가설이 참일 때 결론이 항상 참이면 타당하다고 한다는 점입니다. 따라서 공허한 참이 성립되는 경우는 그냥 무시하고  $p$ 가 True일때만 보면 됩니다. 여기서 초록색 경우만 존재하는 경우 해당 주장은 타당한 주장이 됩니다. 그리고 ‘가설→결론’이라고 써 놔는데, 보통은 가설 여러 개가 AND로 묶여서 들어오는 경우가 대부분입니다.

## 주장, 결론 찾기

- A는 공연 티켓이 남아 있고 티켓 살 돈이 있으면 반드시 공연을 보러 갔을 것이다.
- $T \wedge M \rightarrow P$
- 티켓이 남아 있었지만 A는 공연을 보러 가지 않았다.
- $T, \sim P$
- 그러므로 A는 돈이 없었던 것이 분명하다.
- $\sim M$
- $(T \wedge M \rightarrow P) \wedge T \wedge \sim P \rightarrow \sim M$

또 이해를 돕기 위해 책의 예제를 가져오겠습니다. 이러한 주장이 있다고 합시다. 여기서 명제라고 할 만한 것은 이 정도가 있을 것입니다. 이제 식을 써야 하니 편의를 위해 이 각각을 T, M, P라고 하도록 하겠습니다. 다 쓰고 깨달았는데, 이 T가 True랑 다르다는 점은 알아두셨으면 좋겠습니다. 책에서 Tk라 하는데 그 이유를 이제야 알 것 같아요.. 그러면 이 첫 줄을 식으로 표현하면 다음과 같이 됩니다. 티켓이 남아 있고(T), 돈이 있다(M) 두개가 모두 True이면 공연을 보러 간다(P) 입니다. 두번째 줄을 바꾸면 티켓이 남아 있고(T), 공연을 안 보러 갔다( $\sim P$ )입니다. 둘이 AND로 묶이는데 여기서는 따로 쓰고 나중에 묶겠습니다. 마지막 줄은 돈이 없다( $\sim M$ ) 하나가 끝입니다. 이를 식으로 표현하면 이렇게 됩니다.

## 주장논리식 표현

- 주장의 식을 간결하게 표현하는 방법
- 가설 여러 개를 '/'로 구분하여 한 줄에 쓰고, 마지막에 '//로 결론을 표시함 ( $p \rightarrow q / p // q$ )

- $T \wedge M \rightarrow P$
- $T$
- $\sim P$
- 결론  $\sim M$
- $(T \wedge M \rightarrow P) \wedge T \wedge \sim P \rightarrow \sim M$
- $(T \wedge M \rightarrow P) / T / \sim P // \sim M$

그 다음은 주장논리식 표현입니다. 이거는 제가 아까 설명한대로, 명제와 같은 방식으로 식을 쓰지 않겠다는 말인데요, 다른 건 없고 그냥 가설들을 '/'로 구분하고, 결론은 '//로 구분해 적는 방식입니다. 예시를 들기 위해 방금 봤던 그것을 가져왔습니다. 애를 주장논리식 표현으로 바꿔 쓰면 다음과 같이 됩니다.

## 반례

- $(T \wedge M \rightarrow P) / T / \sim P // \sim M$

$T$	$P$	$M$	$T \wedge M \rightarrow P$	$\sim P$	$\sim M$
T	F	T	T	T	F
T	F	F	T	T	T

반례

조건이 맞는  
경우만 보면 됨

- -> 돈이 있다고 꼭 공연을 보는 것은 아님

그 다음은 반례인데요, 뭐 설명은 필요없겠죠. 반례 찾는 법만 설명하고 넘어가겠습니다. 아까 그 식을 또 가져왔습니다. 이걸 아까 설명했듯 다 볼 필요 없이 가설이 True인 경우만 보면 됩니다. 따라서 T는 True, P은 False로 고정되니 M만 보면 됩니다. 이렇게 보니 결론이 False인 경우를 볼 수 있는데 이게 반례입니다. 저걸 글로 쓰면 돈이 있다고 꼭 공연을 보는 건 아니라는 말이 됩니다. 이 주장은 타당한 주장이 아닌거죠.

## 추론, 추론규칙

추론: 알려진 사실에 타당한 주장을 적용하여 새로운 결론에 도달하는 과정

- 추론규칙: 추론에 사용하는 타당한 주장

그 다음은 추론인데요, 다른 건 없고 가설들로부터 결론을 만드는, 그러니까 주장을 만드는 과정입니다. 추론규칙은 여기에 쓰는 타당한 주장이고요.



## 기본 추론규칙

규칙	이론	설명이나 예시
→ 제거 규칙	$X \rightarrow Y / X // Y$	추론의 기본이고 가장 많이 사용되는 규칙 발표자로 지목되면 발표를 한다.( $X \rightarrow Y$ ) / 발표자로 지목되었다.( $X$ ) // 발표를 한다.( $Y$ )
∨ 제거 규칙	$X \vee Y / \sim X // Y$	앞에서 봤던 동일규칙 ( $F \vee p \equiv p$ ) 김성민 선배: 33기분 또는 선지훈 씨 다음 주에 발표하세요.( $X \vee Y$ ) / 33기: 주제도 안 올렸고 아는 것도 없는 1학년인데 무슨 발표를 해요?( $\sim X$ ) // 회장님: 그럼 지훈 씨...( $Y$ )
∨ 도입 규칙	$X // X \vee Y$	이미 X가 True이므로 X or Y는 X에 의해 True 확정 (우위규칙 $T \vee p \equiv T$ ) 나는 OMS 발표를 한다.( $X$ ) // 나는 시사 또는 OMS 발표를 한다.( $X \vee Y$ )
∧ 제거 규칙	$X \wedge Y // X$	$X \wedge Y$ 가 True이므로 X, Y 전부 True 오늘 수업도 있고 동아리 정모도 있다.( $X \wedge Y$ ) // 오늘 동아리 정모가 있다.( $X$ )
∧ 도입 규칙	$X/Y // X \wedge Y$	위 규칙의 역(위 규칙은 역이 성립함) 목요일 1,2,3,4 수업 있음 / 목요일 5,6 수업 있음 / 목요일 7,8,9 수업 있음 / 목요일 11,12,13 수업 있음 // 목요일 오전 9시부터 밤 10시까지 저녁식사 1시간 제외 12시간 풀강

그 추론규칙들을 알아보겠습니다. 제가 주장을 설명할 때 가설이 True일 때만 보면 된다고 했던 거 기억하시죠? 이 점을 기억하면서 추론규칙을 봐 주시기 바랍니다. 추론규칙은 또 기본 추론규칙과 이를 이용한 추가 추론규칙이 있는데, 먼저 기본 추론규칙부터 알아보겠습니다.

먼저 → 제거규칙입니다. X가 True이면 Y도 True인데 X가 True이니 Y는 자동으로 True가 되는 것입니다. 이 규칙은 추론의 기본이고, 가장 많이 사용되는 규칙입니다. OMS를 이용해 예시를 들어보자면, 제가 발표자로 지목되었으면 제가 발표를 해야겠죠? 그리고 저번 주에 제가 발표자로 지목되었으니, 지금 발표를 해야 하는 것입니다. 다음은 OR 제거규칙입니다. X 또는 Y가 True인데 X가 True가 아니면 Y가 True가 됩니다. 이는 앞 동치규칙에서 봤던 동일규칙으로도 설명할 수 있습니다. 저번 주 OMS 발표자 지목 상황을 예로 들어 보겠습니다. 저번 주에 발표를 진행한 김성민 선배님께서 33기, 33기가 안되면 선지훈 선배가 발표하라고 지목했었습니다. 그때 제가 발표를 할 수 없다고 했다면 지훈 선배님께서 발표를 하게 되었겠죠. 물론 제가 거부를 하지 않았기 때문에 오늘은 제가 발표를 하게 됐습니다. 발표를 선지훈 선배가 하지 않는데 이 Y가 True라고 할 수 있는 이유는, 앞에서 말했듯 주장은 가설이 참인 경우만 따지기 때문입니다. 오늘은 이 가설에서 ~X가 False이니 오늘같이 제가 발표 거부를 안 한 경우는 고려를 하면 안됩니다.

다음 OR 도입규칙입니다. X가 True이면 여기에 어떤 다른 명제를 OR로 묶어도 True겠죠. 이것도 동치규칙 중 우위규칙으로도 설명할 수 있습니다. 또 정모로 예를 들어 보겠습니다. 제가 OMS 발표를 하는 게 True이니, 제가 시사 발표는 안 했지만 제가 시사 또는 OMS 발표를 하는 것은 사실이겠죠.

다음은 AND 제거 규칙입니다. X 그리고 Y가 모두 True이니 당연히 X도 True이고 Y도 True가 될 것입니다. 또 정모로 예를 들면, 오늘이 학기 중 월요일이라 수업도 있고

동아리 정모도 있다면, 오늘 동아리 정모가 있다는 것도 당연히 사실입니다. 제가 2학기에는 월요일 공강을 설계하고 있긴 하지만, 이것이 성공한다 하더라도 앞서 말했듯 주장은 가설이 거짓인 경우는 생각하지 않으니 여전히 사실일 겁니다. 마지막은 AND 도입 규칙입니다. X가 True이고 Y가 True면 X 그리고 Y 둘 다 True가 될 것입니다. 이번에는 정모가 아니라 제 2학기 시간표 중 한 가지 경우의 수를 예시로 들어 보겠습니다. 그 시간표에는 목요일 1,2,3,4교시에 수업이 있습니다. 또한 목요일 5,6교시에도 수업이 존재합니다. 또 목요일 7,8,9교시에도 수업이 있고, 목요일 11,12,13교시에도 수업이 있습니다. 혹시 모르실까봐 알려드리면 우리 학교에서 13교시는 오후 9시부터 10시입니다. 이게 전부 사실이니, 이 시간표대로라면 저는 오전 9시부터 오후 10시까지, 저녁식사 1시간을 제외하면 12시간을 강의만 듣게 된다는 점도 True가 될 것입니다. 이 5개가 기본 추론규칙입니다. 이해하면 알겠지만 이것도 전부 상식 수준입니다.

## 기본 추론규칙

- $(G \rightarrow (M \rightarrow U)) \rightarrow 0 / G \rightarrow (M \rightarrow U) // 0$
- $(B \wedge P) \vee (H \rightarrow I) // ((B \wedge P) \vee (H \rightarrow I)) \vee (L \vee (F \wedge \sim E))$

또 이해를 돕기 위해 또 책에서 예시를 가져왔습니다. 첫 번째의 경우에는 이 부분이  $\rightarrow$  제거규칙에서 X 부분이 되니  $\rightarrow$  제거규칙을 써 줄 수 있고요, 두 번째 것도 이 부분을 하나로 보면 OR 도입규칙에 문제가 없습니다. 이렇게 복잡해 보이는 것도.. 이번에는 다루지 않겠지만.. 묶어서 하나의 명제로 보면 규칙 적용에 문제가 없습니다.

## 추가 추론규칙

규칙	이론	증명	설명
대우 → 제거 규칙	$X \rightarrow Y / \sim Y$ // $\sim X$	1. $X \rightarrow Y$ 2. $\sim Y$ // $\sim X$ 3. $\sim Y \rightarrow \sim X$ (1, 대우규칙) 4. $\sim X$ (2, 3, → 제거규칙)	대우규칙과 → 제거규칙을 연달아 사용하는 것
삼단 규칙	$X \rightarrow Y / Y \rightarrow Z$ // $X \rightarrow Z$	→ 도입규칙(조건증명)	삼단논법
딜레마 규칙	$X \vee Y / X \rightarrow Z$ / $Y \rightarrow Z$ // $Z$	~ 도입규칙(간접증명)	"바다나 산으로 휴가를 가려 한다. 바다에 가면 햇볕이 뜨거워 싫고 산에 가면 힘들어서 싫다." // "휴가를 안 간다"
모순 규칙	$X \rightarrow False$ // $\sim X$	1. $X \rightarrow False$ // $\sim X$ 2. $True$ (가설에 추가(동일규칙)) 3. $\sim X$ (1, 2, 대우 → 제거규칙)	"~는 있을 수 없는 일이다", "~는 상상도 할 수 없다" 등의 표현으로 사용

다음은 이 기본 추론규칙을 이용한 추가 추론규칙입니다. 증명을 표기하는 방법은 설명 없이 넘어가도록 하겠습니다. 설명 없어도 이해에는 문제가 없을 겁니다. 먼저 대우 → 제거규칙입니다. 이건 다른 거 없고 대우규칙과 → 제거규칙을 연달아 사용하는 것입니다. 이렇게 두 개를 연달아 사용하는 경우가 많아 따로 빼놓았을 뿐입니다.

삼단규칙은 삼단논법으로 많이 들어 보셨을텐데, 지금까지 다른 규칙만으로는 설명이 안 되어서 뒤에서 설명하겠습니다.

딜레마 규칙 또한 같은 이유로 뒤에서 설명할 텐데요, 이건 생소할 수 있으니 책의 예시를 가져왔습니다. 바다나 산으로 휴가를 가야 합니다. 그런데 바다에 가면 햇볕이 뜨거워서 싫고 산에 가면 힘들어서 싫습니다. 그러면 결론은 그냥 휴가를 안 가는 것입니다. 다음으로 모순규칙입니다. X가 True이면 False가 True가 되어야 합니다. 그러니까 조건절의 결과는 False인데 조건절 자체는 가설에 포함되므로 True가 되어야 합니다. 이를 충족하려면 공허한 참을 이용할 수밖에 없습니다. 이때 공허한 참을 이용해야 하기 때문에, X는 False가 됩니다. 규칙을 이용한 증명 과정은 먼저 동일규칙을 이용해 가설에 True를 추가하고, 기존 가설에 대우 → 제거규칙을 사용하면 됩니다. 사용 예시로는 "~는 있을 수 없는 일이다" 아니면 "~는 상상도 할 수 없다" 같은 표현이 모순 규칙에 해당합니다.

## 조건증명

규칙	이론	증명	설명
→ 도입 규칙	(가설( $X$ 제외)) // $X \rightarrow Y$	<ol style="list-style-type: none"> <li>1. (가설(<math>X</math>제외)) // <math>X \rightarrow Y</math></li> <li>2. <math>X</math> (가정)</li> <li>3. <math>Y</math> (<math>X</math>를 가정하고 결론 <math>Y</math>를 (다른 규칙을 써서)얻음)</li> <li>4. <math>X \rightarrow Y</math> (<math>\rightarrow</math> 도입)</li> </ol> <p>증명시 가정한 부분을 들여쓰기로 구분함</p>	$R \rightarrow (M \vee B) / \sim B // R \rightarrow M$ <ol style="list-style-type: none"> <li>1. <math>R \rightarrow (M \vee B)</math></li> <li>2. <math>\sim B // R \rightarrow M</math></li> <li>3. <math>R</math> (가정)</li> <li>4. <math>M \vee B</math> (1, 3, <math>\rightarrow</math> 제거)</li> <li>5. <math>M</math> (2, 4, <math>\vee</math> 제거)</li> <li>6. <math>R \rightarrow M</math> (3~5, <math>\rightarrow</math> 도입)</li> </ol> <p>삼단규칙</p> <ol style="list-style-type: none"> <li>1. <math>A \rightarrow B</math></li> <li>2. <math>B \rightarrow C // A \rightarrow C</math></li> <li>3. <math>A</math> (가정)</li> <li>4. <math>B</math> (1, 3, <math>\rightarrow</math> 제거)</li> <li>5. <math>C</math> (2, 4, <math>\rightarrow</math> 제거)</li> <li>6. <math>A \rightarrow C</math> (3~5, <math>\rightarrow</math> 도입)</li> </ol>

다음은 조건증명입니다. 조건증명은 증명을 위해 추가 가설을 세우는 증명 방법입니다. 그냥 이렇게 들어서는 이해가 안될 테니 이를 이용한 규칙인  $\rightarrow$  도입규칙을 설명하겠습니다. 식도 이해가 안 될 수도 있는데, 여기서 가설은 결론을 유도할 수는 있지만,  $X$ 가 직접적으로 들어가지 않은 형태가 되어야 합니다. 그렇기에 규칙 증명도  $X$ 를 통해 어떻게든  $Y$ 를 얻을 수 있다고 써 놓고 다른 말이 없습니다. 이해 안 되는 것 투성이라고 생각하실 수 있지만 예시를 보면 쉽습니다. 예... 또 책에서 가져왔습니다. 이 주장의 경우  $R$ 을 가정하고  $\rightarrow$  제거와 OR 제거를 이용해  $M$ 을 얻었으며, 이에 따라 이 결론을 얻을 수 있습니다. 증명이나 예시를 보면 알겠지만, 가정을 한 부분은 반드시 들여쓰기로 구분해 주어야 합니다. 이를 이용해 앞에서 언급했던 삼단규칙을 증명해보겠습니다.  $A$ 를 가정하고,  $\rightarrow$  제거를 통해 각각  $B$ 와  $C$ 를 얻고,  $A$ 를 가정해  $C$ 를 얻었으니 이러한 결론을 끌어낼 수 있는 것입니다.

## 간접증명

- 귀류법, 모순에 의한 증명,  $\sim$  도입규칙
- 결론의 부정을 가설로 하고 시작하여 기존의 가정과 모순된다는 것을 보이면 됨
- 모순을 증명하는 방법:  $p \wedge \sim p \equiv T$ 임을 보이기

딜레마규칙 증명 ( $A \vee B / A \rightarrow C / B \rightarrow C // C$ )

1.  $A \vee B$

2.  $A \rightarrow C$

3.  $B \rightarrow C // C$

4.  $\sim C$  (결론 부정)

5.  $\sim A$  (2, 4, 대우  $\rightarrow$  제거)

6.  $\sim B$  (3, 4, 대우  $\rightarrow$  제거)

7.  $\sim A \wedge \sim B$  (5, 6,  $\wedge$  도입)

8.  $\sim(A \vee B)$  (7, 드모르간 규칙)

9.  $(A \vee B) \wedge \sim(A \vee B)$  (1, 8,  $\wedge$  도입, 모순)

10.  $C$  (4~9,  $\sim$ 도입)

마지막으로 간접증명입니다. 이 증명은 귀류법, 모순에 의한 증명, NOT 도입규칙이라고도 불리며, 결론의 부정을 가설로 하고 시작해 기존의 가정과 모순됨을 보이는 방법입니다. 여기서 모순을 보이는 방법은 어떤 명제와 그 명제의 부정이 AND로 묶었는데 그게 True이어야 하는 경우를 보이면 됩니다. 예시로 앞에서 언급했던 딜레마규칙을 증명해보겠습니다. 이것도 새로운 가설을 만드는 것이라 들여쓰기를 해주어야 합니다. 먼저 결론 부정으로 시작합니다. 다음 대우  $\rightarrow$  제거규칙을 이용해  $\sim A$ 와  $\sim B$ 를 얻을 수 있습니다. 이 둘을 AND로 묶고 드모르간 규칙을 사용하면, 가설은 모두 True이어야 하는데 1번에 따라 모순이 발생합니다. 따라서 결론 C를 얻을 수 있습니다. 제 발표는 여기까지입니다. 뒤에 술어논리 부분을 빼서 책의 앞 반절만 요약하게 되었는데, 계속하면 시간 문제가 생길 것 같아 뺐습니다.



# 인공지능 논리 입문 - 술어논리

2024.02.07 OMS

안녕하세요, 저는 오늘 수리논리 중 술어논리에 대해 발표할 Zeropage 33기 정현승입니다.

\*(2페이지가 이어짐)\*

앞에서, 그러니까 저번 8월 21일 OMS에서, 명제논리에 대한 설명을 했기 때문에, 바로 그 다음부터 진행을 하도록 하겠습니다. 앞에 내용 못 들으셨어도 내용이 이산수학과 중복되는지라 별 문제는 없을 것이라고 생각합니다.

## 정언명제

- 주어 집합과 술어 집합의 포함과 배제관계를 서술하는 명제
- 주어와 술어를 각각 집합으로 보고, 두 집합 간의 관계를 서술함
- 주어와 술어를 분리할 수 없어서 생기는 명제의 한계 보완

먼저 정언명제입니다. 정언명제란 주어 집합과 술어 집합의 포함과 배제관계를 서술하는, 그러니까 주어와 술어를 각각 집합으로 보고, 두 집합 간의 관계를 서술하는 명제입니다. 이 정언명제는 주어와 술어를 분리할 수 없고, 집합 중 일부의 성질을 표현할 수 없으며, 같은 대상에 대해 두 개 이상의 속성이나 성질 관계를 표현할 수 없는 기존 명제의 한계를 보완할 수 있는 명제입니다.



## 한정자

- $\forall$ : 전체 한정자
- $\exists$ : 존재 한정자

여기서 한정자라는 개념이 나오는데요, 이걸 전부 다 알고 계시겠죠. 책에 쓰여 있는 한정자의 정의만 읽어 드리고 넘어가겠습니다. "한정자는 변수에 적용되어 변수가 나타내는 대상 집합을 정한다." 라고 되어 있습니다.

## 정언명제

규칙	설명	예시	식
A형식	모든 S는 P다.	모든 나무는 식물이다.	$\forall S \rightarrow P$
E형식	모든 S는 P가 아니다.	모든 식물은 동물이 아니다.	$\forall S \rightarrow \sim P$
I형식	S 중에 P인 것이 있다.	어떤 동물은 광합성을 한다.	$\exists S \rightarrow P$
O형식	S 중에 P가 아닌 것이 있다.	어떤 식물은 땅에서 자라지 않는다.	$\exists S \rightarrow \sim P$

정언명제는 A, E, I, O 네 가지 형식이 있다고는 하는데, 말이 어렵지 뭐 소린지는 다 아시겠죠. 저도 왜 이런 걸 구분했는지 모르겠습니다만, 일단 넘어가겠습니다.

## 술어논리

- 주어나 대상을 변수로 설정하고 술어라는 심볼을 이용해 기술하는 것
- 주어와 술어가 분리되어 있어 결합과 분할이 가능해짐
- 술어와 변수로 구성

그리고 술어논리입니다. 술어논리는 정언명제의 수학적 표현체계입니다. 술어와 변수로 구성되며, 주어나 대상을 변수로 표현하고 술어라는 심볼을 이용해 기술하는 것인데 주어와 술어가 분리되어 있어 결합과 분할이 가능하다는 특징이 있습니다. 이는 다음 예시에서 자세히 알아보겠습니다. 여기서 변수는 무엇이든 될 수 있는 기호이고, 술어는 여러 개의 대상을 가질 수 있다라고 책에 쓰여 있는데, 저게 무슨 소리인지는 예시와 함께 알아보도록 하겠습니다.

## 술어논리

- 철수가 영희에게 책을 줬다.
- 지훈이 중혁에게 빵을 줬다.
- 경희가 정연에게 요리를 해 주었다.
- x가 y에게 z를 줬다.
- 전문가가 문제 발생 원인으로 행인의 가방을 지목했다.
- 정현승이 다음 OMS로 김도엽 선생의 "맥도날드 포인트 누적 44,660의 메뉴 리뷰" 주제를 지목했다.
- 권하연 선생이 다음 OMS로 정현승의 자유주제를 지목했다.
- x가 y로 z의 w를 지목했다.

주다(x, y, z)

Boolean Give(Object x, Object y, Object z)

Boolean Pick(Person x, function y, Object z, Object w)

Boolean PickOMS(Person x, Person z, Subject w)

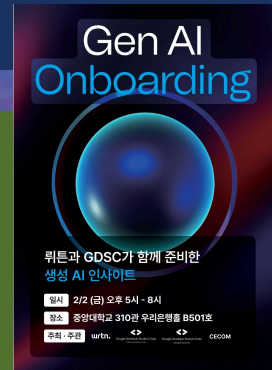
네 이게 예시입니다. 앞에 있는 한 세트는 책에 있는 걸 그대로 가져왔고요, 아래의 한 세트는 제가 직접 만든 예시라 약간 부적절할 수도 있습니다. 앞 3개 문장을 보면 주어, 목적어 정도만 다르고 서술어가 같습니다. 아까 술어논리가 주어나 대상을 변수로 설정한다고 말씀을 드렸는데요, 이 문장에서 변수를 찾아보겠습니다. 첫 번째 문장 "철수가 영희에게 책을 줬다."에서 변수는 철수, 영희, 책이 될 수 있고요, 두 번째 문장 "지훈이 중혁에게 빵을 줬다."에서는 지훈, 중혁, 빵이 변수, 세 번째 문장 "경희가 정연에게 요리를 해 주었다."에선 경희, 정연, 요리를 하다 이게 변수가 될 수 있을 겁니다. 이 세 문장에서 변수를 빼서 기호로 바꿔보면 이렇게, "x가 y에게 z를 줬다."이렇게 표현할 수 있게 됩니다. 네 이게 술어입니다. 아까 주어와 술어가 분리되어 있어 결합과 분할이 가능하다 이렇게 설명을 드렸는데요, 이게 이겁니다. 변수의 조합이 가능해지는 겁니다. 이걸 토대로 다시 돌아가 보면 첫 번째 문장에선 x가 철수, y가 영희, z가 책이 될 수 있을 것이고요, 두 번째 문장에서는 x가 지훈, y가 중혁, z가 빵이 되고, 세 번째 문장에서는 x가 경희, y가 정연, z가 요리를 하다가 될 수 있는 겁니다. 예시 하나를 더 들어보겠습니다. 첫 번째 문장 "전문가가 문제 발생 원인으로 행인의 가방을 지목했다"에서는 변수가 전문가, 문제 발생 원인, 행인, 가방이 될 수 있을 것이고, 두 번째 문장 "정현승이 다음 OMS로 김도엽 선생의 "맥도날드 포인트 누적 44,660의 메뉴 리뷰" 주제를 지목했다." 에서는 정현승, 다음 OMS, 김도엽 선생, "맥도날드 포인트 누적 44,660의 메뉴 리뷰" 주제가, 세 번째 문장 "권하연 선생이 다음 OMS로 정현승의 자유주제를 지목했다." 에서는 권하연 선생, 다음 OMS, 정현승, 자유주제가, 변수가 될 수 있을 겁니다. 이 세 문장에서 다시 변수를 기호로 바꾸면, "x가 y로 z의 w를 지목했다." 이렇게 표현이 가능하게 됩니다. 이를 토대로 다시 돌아가 보면 첫 번째 문장은 x가 전문가, y가 문제 발생 원인, z가 행인, y가 가방이 되고요, 두 번째 문장에서는 x가 정현승, y가 다음 OMS, z가 김도엽 선생, w가 "맥도날드 포인트 누적

44,660의 메뉴 리뷰" 주제가 되어 "정현승이 다음 OMS로 김도엽 선배의 "맥도날드 포인트 누적 44,660의 메뉴 리뷰" 주제를 지목했다."라는 문장이 완성됩니다. 세 번째 문장도 x가 권하연 선배, y가 다음 OMS, z가 정현승, w가 자유주제가 되어서 "권하연 선배가 다음 OMS로 정현승의 자유주제를 지목했다." 라는 문장이 완성되는 겁니다. 근데 여기서 끝이 아니라 이 술어를 심볼로 또 바꿀 수 있습니다. 책에서는 이렇게 표현하고 있습니다. 이렇게 보니까 뭔가 Python에서 볼 것 같이 생겼죠? 술어가 함수가 되고, 각 변수를 인자로 받는 형태입니다. 근데 이렇게 보면 좀 덜 직관적이니까, JAVA에서 볼 것같이 생긴 함수로 바꿔보겠습니다. 이러니까 뭔가 익숙하죠? 술어는 Give이고, 이 함수는 인자를 3개 받는데, x와 y는 사람이고, z는 그냥 Object 형의 클래스나 구조체를 받는다고 되어 있습니다. 근데 비록 예시에서는 x와 y가 사람이지만 꼭 여기에 사람만 들어갈 필요는 없죠? 애초에 앞에서 변수는 무엇이든 될 수 있는 기호라고 설명하기도 했고요, 따라서 이걸 약간 수정해 보면 이렇게 되겠네요. 물론 꼭 Object에 사람이 아닌 어떤 것이 반드시 들어가야 한다는 말은 아닙니다. 애초에 JAVA에서 Object 클래스란 모든 클래스의 최상위 클래스이기 때문에 그냥 어떤 형태의 클래스가 들어가도 좋다는 의미에 가깝죠.

아래 예시도 비슷하게 바꿔보겠습니다. 바꾸면 이렇게 될 겁니다. 여기서 function은 생각나는 거 없어서 그냥 써 놓은 거니 무시하셔도 되고요, Pick이란 술어, 그러니까 함수는 사람 x, 함수 y, Object형 클래스 z와 w를 인수로 받는 거죠. 여기서 첫 번째 문장을 지우면 OMS가 중복되니, 이를 술어로 빼서, 이렇게 사람 x, 사람 z, 주제 w를 받는 PickOMS라는 술어, 그러니까 함수를 만들 수도 있겠죠. 다만 여러 형태로 사용할 수 있게 하기 위해, 술어는 구체적인 것을 피하는 게 좋다고 하네요.

## 술어논리..?

- |                                  |            |
|----------------------------------|------------|
| ▪ 1형식: S(주어)+V(동사)               | ▪ V(S)     |
| ▪ 2형식: S(주어)+V(동사)+C(보어)         | ▪ V(S,C)   |
| ▪ 3형식: S(주어)+V(동사)+O(목적어)        | ▪ V(S,O)   |
| ▪ 4형식: S(주어)+V(동사)+O(목적어)+O(목적어) | ▪ V(S,O,O) |
| ▪ 5형식: S(주어)+V(동사)+O(목적어)+C(보어)  | ▪ V(S,O,C) |
- 
- |                            |  |
|----------------------------|--|
| ▪ 안은문장, 안긴문장, ...          | ▪ interface 체언{...}, class 명사 implements 체언{...} |
| ▪ 체언, 용언, 관형사, 부사, 조사, 감탄사 | ▪ 체언 관형사(체언)                                     |



여기서부터는 책에 없는 내용이기도 하지만, 그러면 우리 같은 주입식 교육을 받은 사람들은, 이런 게 생각이 날 것입니다. 앞에 이건 한국식 영어 문법이고요, 아래 이건 국어 연매의 언어 부분입니다. 한국의 주입식 교육을 받지 않은 사람은 둘 다 모를 수도 있어요. 네... 그 흔히 말하는 주입식 교육의 폐해죠... 영어의 경우에는 이렇게 1형식부터 5형식까지 있는데 각 형식을 컴퓨터 언어식 함수로 바꾸면 이렇게 될 겁니다. 국어에서는 이런 게 있었어요. 뭐 언어와 매체를 선택하지 않고 화법과 작문을 선택한 사람은 이걸 모를 수도 있습니다. 뭐 저도 연매 골랐는데 까먹어서 다시 찾아봤어요. 명사를 이렇게 JAVA식으로 선언하면, 관형사는 체언을 수식하고 이게 세트로 묶여 체언 자리, 그러니까 여기 위에 S, O, C 자리에 들어가야 하니, return 값도 체언, 수식할 단어도 체언이니 이런 식으로 만들어질 수 있겠죠. 뭔가 이해 되시나요? 이건 사람의 언어와 컴퓨터 언어 간의 관계로 보시면 되고, 이 주제의 핵심이죠. 제가 저번 금요일에 뉘튼에서 진행하는 컨퍼런스에 다녀왔는데요, 사진 찍은 게 없어서 관련 자료가 이 포스터 하나밖에 없지만, 생성형 AI는 질문을 읽거나 답변을 쓸 때, 단어를 token이라는 단위로 끊어서 인식한다고 하더라고요.

## 술어논리

- 철수가 영희에게 책을 줬다.
- 지훈이 중혁에게 빵을 줬다.
- 경희가 정연에게 요리를 해 주었다.
- x가 y에게 z를 줬다.
- 전문가가 문제 발생 원인으로 행인의 가방을 지목했다.
- 정현승이 다음 OMS로 김도엽 선배의 "맥도날드 포인트 누적 44,660의 메뉴 리뷰" 주제를 지목했다.
- 권하연 선배가 다음 OMS로 정현승의 자유주제를 지목했다.
- x가 y로 z의 w를 지목했다.

주다(x, y, z)

Boolean Give(Object x, Object y, Object z)

Boolean Pick(Person x, function y, Object z, Object w)

Boolean PickOMS(Person x, Person z, Subject w)

아까 거기서, 그 token의 단위가 여기 노란 색으로 표시해 놓은 이런 단위였습니다. 조사도 전부 token에 들어가고 술어도 형태소 단위로 들어가는 것 같긴 하나, 제가 전문가가 아닌 관계로, 그걸 여기서 세세히 짚을 순 없으므로 넘어가겠습니다. 그러니까 문득, "술어논리에서 다루었던 이 변수와 술어가 죄다 token으로 인식되는 것은 아닐까" 하는 생각이 들더라고요. 행사 끝나고 생각난 거라 다른 분들께 여쭙어보지는 못했지만, 이러한 술어논리의 변수와 술어 구분은 생성형 AI의 키 포인트가 될 수 있지 않을까 싶습니다. 근데 책 이름이 "인공지능 논리 입문"인데 이 부분에 대해서는 언급이 전혀 없네요..

## 술어논리

- 함수 그 자체로는 명제가 아님(참과 거짓을 구분할 수 없기 때문)
- 함수에 넘기는 인수가 모두 정해지면, 그때서야 함수는 명제의 역할을 할 수 있음
- 함수 실행에 필요한 인수를 넣지 않으면 컴파일 에러를 뿜어내는 것과 같음
- 전체집합: 술어의 대상이 될 수 있는 모든 객체의 집합(인자의 자료형의 집합)

주다(x, y, z)

Boolean Give(Object x, Object y, Object z)

Boolean Pick(Person x, function y, Object z, Object w)

Boolean PickOMS(Person x, Person z, Subject w)

다시 책 내용으로 돌아오겠습니다. 아까 그 함수 그 자체로는 명제가 될 수 없고, 함수에 넘기는 인수가 전부 정해져야 함수는 명제의 역할을 할 수 있습니다. 앞에서, 그러니까 8월 OMS에서 명제의 정의가 "참 또는 거짓을 구별할 수 있는 객관적 상태가 포함된 문장"으로 되어 있다고 말씀드렸는데, 인수가 정해지지 않고서는 이를 구별할 수가 없기 때문입니다. 제가 앞 함수 표현식의 return 값을 전부 Boolean으로 만들어 놓은 이유도 여기에 있습니다. 함수에 넘기는 인수가 전부 정해지지 않을 때 명제의 역할을 할 수 없다는 부분은, 프로그램 짤 때 함수 실행에 필요한 인수를 전부 넣지 않으면 컴파일 에러가 뜨는 것과 같은 맥락으로 이해하시면 될 것 같습니다. 기본값 같은 경우는 어쨌든 인수를 준 것이기 때문에 상관 없고요.

또 전체집합이라는 개념이 있는데요, 이건 술어의 대상이 될 수 있는 모든 객체의 집합인데요, 아까 인자에 써 놓았던 Person이나 Subject, Object같은 것으로 보면 됩니다.



## 술어논리

술어	함수 형태	연산자 오버로딩(C++)	기호를 이용한 간단한 표현식
x가 y보다 크다	bigger(x, y)	bool operator>(int y)	x>y
x가 y보다 작다	smaller(x, y)	bool operator<(int y)	x<y
x와 y가 같다	equal(x, y)	bool operator==(int y)	x==y
x와 y가 같지 않다	notequal(x, y)	bool operator!=(int y)	x!=y
	주다(x, y, z)		

그리고 술어를 반드시 저런 식으로만 표현해야 하는 건 또 아닙니다. 몇몇 술어는 기호를 이용해 쉽게 표현하기도 하는데, 이건 연산자 오버로딩과 같은 개념으로 이해하시면 됩니다. 예를 들어 x가 y보다 크다는 술어는 지금까지 봤던 표현식, 그러니까 이런 Python식 표현식입니다. 이 표현식으로는 bigger(x,y) 같은 형태로 표현이 가능한데, 이걸 이렇게 표현할 수 있다는 내용입니다. 이걸 프로그래밍 언어로 바꿔 보면 이런 식으로 될 것입니다. 여기도 JAVA로 넣으려 했는데 JAVA에서 연산자 오버로딩 어떻게 하는지 몰라서 그냥 C++ 것을 가져왔어요. 나머지는 이해가 어렵지 않으실 테니 설명 없이 넘어가겠습니다. 이건 그냥 모든 술어를 무식하게 이런 식으로만 표현하지는 않는다 이 정도를 표현하기 위해 가져온 거지 더 특별한 무언가는 없습니다. 이게 끝입니다. 책에는 뒤에 단일변수 술어논리, 다중변수 술어논리 해서 더 내용이 있는데, 그건 그냥 이걸 활용하는 형태라 더 설명할 게 더 없습니다.

- $(G \rightarrow (M \rightarrow U)) \rightarrow O / G \rightarrow (M \rightarrow U) // O$
- $(B \wedge P) \vee (H \rightarrow I) // ((B \wedge P) \vee (H \rightarrow I)) \vee (L \vee (F \wedge \sim E))$
  
- $\forall x. \forall y. \text{Loves}(x, y) \rightarrow \forall z. \text{Happy}(z)$

이건 저번 8월 OMS때 다뤘던 예시인데 여기서 문자로 표시한 각종 명제에 한정자와 결합한 술어논리가 가득 들어있는 거, 뭐 이런 식으로 정도 외에는 뭐가 없기 때문에 발표는 여기서 마무리하도록 하겠습니다.