5. 조건 명제

Conditionals

진리함수 논리연산자 Truth-functional Connectives

- 진리함수 논리연산자란?
  - 피연산자의 진리값만으로 복합문장의 진리값을 결정할 수 있는 논리연산자를 진리함수 논리연산자라고 한다.
- 불 논리연산자($\neg$, $\wedge$, $\vee$)는 진리함수 논리연산자이다.
- 진리함수가 아닌 논리연산자는 FOL에 포함될 수 없다.
- <u>목표</u>: 진리함수적인 조건 논리연산자($\rightarrow$, $\leftrightarrow$)를 FOL에 포함시켜보자.

조건 명제 기호Conditional Symbol: →

구문Syntax: $P \to Q$

antecedent(전항,가정,전제) consequent(후항,결과)

의미Semantics:

- 문장 $P \to Q$ 가 참true이다

  iff

  P가 거짓false이거나 Q가 참true이다
  (또는 P와 Q의 진리 값이 같다)
- $P \to Q \equiv \neg P \lor Q$
- $P \to Q \approx$ if P then Q (다양한 영어 표현)

  = P only if Q

  = Q provided P

  = Q if P

진리표Truth Table:

P	Q	P→Q
T	T	T
T	F	F
F	T	T
F	F	T

필요 조건 Necessary Conditions

- 뭔가를 얻기 위해서 반드시 성립해야 하는 조건
- 영어: “only if” 뒤에 필요조건이 나온다.
- 예제
  - ✓ 영어: You will pass the course only if you turn in all the homework assignments.
  - ✓ 이 과목을 이수하기 위해서는 숙제를 모두 낼 필요가 있다.
  - ✓ 숙제를 내는 것이 이 과목 이수의 필요 조건이다.
  - ✓ 숙제를 내지 않으면 탈락이다.
  - ✓ 그렇지만 숙제를 다 내도 빵꾸 날 수 있다.

충분 조건 Sufficient Conditions

- 뭔가 얻을 수 있다는 보장을 해주는 조건
- 영어: “if” 뒤에 충분조건 나온다.
- 예제
  - ✓ 영어: You will pass the course if you turn in all the homework assignments
  - ✓ 이 과목을 이수하기 위해서 숙제를 다 내면 충분하다.
  - ✓ 숙제를 제출하는 것이 이 과목 이수의 충분 조건이다.
  - ✓ 숙제만 다 내면 과목을 이수한다.
  - ✓ 즉, 숙제를 다 내고도 빵꾸 나는 일은 절대로 없다.

→ 의 사용

- Q는 $P_1, \ldots, P_n$의 논리적 결과이다
  <u>if and only if</u>
  문장 $(P_1 \wedge \ldots \wedge P_n) \rightarrow Q$ 은 논리적으로 진리이다.
- Unless P, Q
  = Q unless P
  = P 아니면 Q이다
  = $\neg P \rightarrow Q$

상호 조건 명제 기호 Biconditional Symbol: ↔

구문Syntax: **P ↔ Q**

영어 : if and only if ≡ iff
우리말 : 필요충분조건

진리표Truth Table:

P	Q	P↔Q
T	T	T
T	F	F
F	T	F
F	F	T

의미Semantics:

- 문장 P ↔ Q 가 참true이다.

  iff

  P와 Q의 진리 값이 같다.
- $P \leftrightarrow Q \equiv (P \rightarrow Q) \wedge (Q \rightarrow P)$
- 주목: ⇔와 혼돈하지 말자

  ⇔ : FOL의 논리식(문장)을 만드는 기호가 아니라, 두 문장이 논리적으로 동일한지(“is logically equivalent to”)를 나타내는 기호

  ↔ : FOL의 논리식(문장)을 만드는 기호; 논리식(문장)의 논리연산자

진리함수 논리연산자의 완전성

Truth-functional Completeness

- 진리함수 논리연산자 = ¬, ∧, ∨, →, ↔
  - ✓ 더 필요한가?
  - ✓ 다 필요한가?
  - ✓ →와 ↔를 추가해서 표현력이 향상되었나?
- 정의Definition
  - ✓ 진리함수 논리연산자의 부분집합 O를 가지고 다른 모든 진리함수 논리연산자를 모두 표현 가능하면, 그 O를 진리 함수적으로 완전하다truth-functionally complete고 한다.
- 정리Theorem
  - ✓ 불 논리연산자 {¬, ∧, ∨}는 진리함수적으로 완전하다.
  - ✓ 증명해보자!

이항 논리연산자 Binary Connectives

P	Q	P*Q
T	T	1
T	F	2
F	T	3
F	F	4

where

$C_1 = (P \land Q)$

$C_2 = (P \land \neg Q)$

$C_3 = (\neg P \land Q)$

$C_4 = (\neg P \land \neg Q)$

1	2	3	4	¬,∨ ∧ 만으로 표현한 문장
F	F	F	F	$P \land \neg P \land Q \land \neg Q$
T	F	F	F	$C_1 = (P \land Q)$
F	T	F	F	C_2
F	F	T	F	C_3
F	F	F	T	C_4
T	T	F	F	$C_1 \lor C_2$
T	F	T	F	$C_1 \lor C_3$
T	F	F	T	$C_1 \lor C_4$
F	T	T	F	$C_2 \lor C_3$
F	T	F	T	$C_2 \lor C_4$
F	F	T	T	$C_3 \lor C_4$
T	T	T	F	$C_1 \lor C_2 \lor C_3 = (P \lor Q)$
T	T	F	T	$C_1 \lor C_2 \lor C_4$
T	F	T	T	$C_1 \lor C_3 \lor C_4$
F	T	T	T	$C_2 \lor C_3 \lor C_4$
T	T	T	T	$C_! \lor C_2 \lor C_3 \lor C_4$

단항 논리연산자 Unary Connectives

P	#P
T	1
F	2

1	2	¬,∨ ∧ 만으로 표현한 문장
F	F	$P \wedge \neg P$
T	F	P
F	T	$\neg P$
T	T	$P \vee \neg P$

삼항 논리연산자 Ternary Connectives

P	Q	R	♣(P,Q,R)
T	T	T	T
T	T	F	T
T	F	T	F
T	F	F	F
F	T	T	T
F	T	F	F
F	F	T	T
F	F	F	F

where ♣(P,Q,R) = if P then Q else R

¬,∨ ∧ 만으로 표현한 문장:

$$(P \wedge Q \wedge R)$$
$$\vee\ (P \wedge Q \wedge \neg R)$$
$$\vee\ (\neg P \wedge Q \wedge R)$$
$$\vee\ (\neg P \wedge \neg Q \wedge R)$$

일반화

- ♠를 n항 논리연산자라고 하자.
- 진리표의 2^n개 각 행에 해당하는 연어連語명제conjunction (논리곱으로 연결된 문장) $C_1, \ldots, C_{2^n}$ 을 정의한다.
- 진리표에서 진리값이 참이 되는 행에 해당되는 연어명제를 논리합으로 연결하여 선언選言명제disjunction(논리합으로 연결된 문장) D를 만든다. 즉, k번째 행의 진리 값이 참이면, C_k는 는 이 선언명제에 포함된다.
- 만약 모든 행이 거짓false이라면, $D \equiv P_1 \wedge \neg P_1$
- 그러면 이 선언명제 D는 ♠$(P_1,\ldots,P_n)$과 항진적으로 동일하다 tautologically equivalent.

증명 끝□

주요 정리Theorem들

- 정리: 불 논리연산자 $\neg$와 $\vee$는 진리함수적으로 완전하다.
- 정리: 불 논리연산자 $\neg$와 $\wedge$는 진리함수적으로 완전하다.
- 정리: P↓Q가 "P도 아니고 Q도 아니다"("neither P nor Q")를 의미한다고 하자. 그러면 불 논리연산자 ↓는 진리함수적으로 완전하다.

  $\neg P = P \downarrow P$ = neither P nor P

  $P \wedge Q = (P \downarrow P) \downarrow (Q \downarrow Q)$ = neither not P nor not Q

사용하는 논리연산자의 개수가 적다고 꼭 좋지만은 않은 이유

- 사용하는 논리연산자의 개수가 적을 수록 문장을 이해하기가 더 어려워진다.
- 사용하는 논리연산자의 개수가 적을 수록 증명이 더 복잡해진다.

조건 명제의 비정형 증명의 타당한 절차들

- 조건 제거 Modus Ponens (Conditional Elimination)

  $P \rightarrow Q$ 와 P에서, Q를 유추
- 상호 조건 제거 Biconditional Elimination

  P 와 ($P \leftrightarrow Q$ 또는 $Q \leftrightarrow P$) 에서, Q를 유추
- 대우 對偶 Contraposition

  $P \rightarrow Q \Leftrightarrow \neg Q \rightarrow \neg P$
- 그 밖의 주요 동일 문장

  $P \rightarrow Q \Leftrightarrow \neg P \vee Q$

  $\neg(P \rightarrow Q) \Leftrightarrow P \wedge \neg Q$

  $P \leftrightarrow Q \Leftrightarrow (P \rightarrow Q) \wedge (Q \rightarrow P)$

  $P \leftrightarrow Q \Leftrightarrow (P \wedge Q) \vee (\neg P \wedge \neg Q)$

비정형 증명 : 조건 증명법 Conditional Proof

- 증명 방법: P → Q를 증명하려면, P를 가정하고 Q를 증명한다.
- 정리: Tet(a) → Tet(c)는 Tet(a) → Tet(b) 와 Tet(b) → Tet(c)의 논리적 결과이다. 다시 말하면, →는 이행성移行性transitive 이다.
- 증명: (조건 증명법)
  - 전제로서 Tet(a)→Tet(b)와 Tet(b)→Tet(c)가 주어졌다. 우리가 증명하려고 하는 건 Tet(a)→Tet(c) 이다.
  - 조건 증명법으로 증명하기 위해서 Tet(a)를 가정하자. 그러면 이것을 첫번째 전제와 Modus Ponens를 적용하면 Tet(b)이다. 이것을 두번째 전제와 Modus Ponens를 적용하면 Tet(c)이다.
  - Tet(a)를 가정하고 Tet(c)를 증명했으므로 조건부 증명법 규칙에 의해서 Tet(a)→Tet(c)이다.

비정형 증명 : 조건 증명법 Conditional Proof

- 정리: $Even(n^2) \rightarrow Even(n)$
- 증명1: (대우법)
  - 대우인 $\neg Even(n) \rightarrow \neg Even(n^2)$을 증명하면 된다.
  - n이 짝수가 아니라고 가정하면, 홀수이므로 2m+1로 표현할 수 있다. 그러면 $n^2 = (2m+1)^2 = 4m^2+4m+1 = 2(2m^2+2m)+1$ 이다. 그러면 n^2도 홀수이다. 증명 끝.

비정형 증명 : 조건 증명법 Conditional Proof

- 정리: Even(n^2) → Even(n)
- 증명2: (조건증명 + 모순유도증명)
  - Even(n^2)를 가정하고 Even(n)을 증명하면 된다.
  - n^2이 짝수라고 가정하고, n이 짝수임을 증명한다.
  - n이 홀수라고 가정하자. 그러면 n은 임의의 정수 m에 대해서 2m+1로 표현할 수 있다. 그러면 $n^2 = (2m+1)^2 = 4m^2+4m+1 = 2(2m^2+2m)+1$ 따라서 n^2은 홀수이라고 할 수 있다. 이는 가정과 모순이다.
  - 따라서 n은 짝수임에 틀림없다.

비정형 증명: 상호조건 증명법 Biconditionals

- 접근법: 상호조건문을 싸이클이 되도록 조건열결자를 사용하여 문장을 재배열한다.
- 주장: Q_1, Q_2, Q_3은 모두 같다. 즉, $Q_1 \leftrightarrow Q_2$, $Q_2 \leftrightarrow Q_3$, $Q_1 \leftrightarrow Q_3$
- 증명방법: 문장이 다음과 같이 배열한 뒤 증명하면 된다.

$$Q_1 \rightarrow Q_2 \rightarrow Q_3 \rightarrow Q_1$$

- 다음 문장들이 모두 같음을 증명하시오.
  - n 은 짝수이다.
  - n^2 은 짝수이다.
  - n^2 은 4로 나누어진다.

contraposition

Proof: 3) $\rightarrow$ 2) $\rightarrow$ 1) $\rightarrow$ 3)

→의 정형 증명 규칙

조건 제거 Conditional Elimination (→ Elim)

	P → Q
	⋮
	P
	⋮
▷	**Q**

Modus Ponens

조건 생성 Conditional Introduction (→ Intro)

		P
		⋮
		Q
▷	**P → Q**	

Conditional Proof

Fitch로 증명해보자

- Conditional 1
- Prove: A → ¬¬A
- Conditional 2

↔의 정형 증명 규칙

상호조건 제거 Biconditional Elimination (↔ Elim)

$$
\begin{array}{l|l}
 & \mathbf{P \leftrightarrow Q} \quad \mathbf{(or\ Q \leftrightarrow P)} \\
 & \vdots \\
 & \mathbf{P} \\
 & \vdots \\
\triangleright & \mathbf{Q}
\end{array}
$$

상호조건 생성Biconditional Introduction (↔ Intro)

$$
\begin{array}{l|l}
 & \begin{array}{|l} P \\ \hline \vdots \\ Q \end{array} \\
 & \begin{array}{|l} Q \\ \hline \vdots \\ P \end{array} \\
\triangleright & \mathbf{P \leftrightarrow Q}
\end{array}
$$

<u>Fitch로 증명해보자</u>

- Prove: A ↔ ¬¬A
- Conditional 3