Computational Thinking

2017-2학기

3장-논리

박종혁 교수 (컴퓨터공학과)

jhpark1@seoultech.ac.kr

http://www.parkjonghyuk.net

서울과학기술대학교

SEOULTECH SEOUL NATIONAL UNIVERSITY OF SCIENCE & TECHNOLOGY

학습개요

- 3장. 논리
  1. 논리란 무엇인가?
  2. 부울논리
  3. 명제 논리의 응용

- 조별과제

학습목표

- 올바르고 이성적인 사고를 위해서 논리가 필요하며 유용하다는 것을 이해한다.
- 자연 언어의 논리가 어떻게 기호로 표현되는지 설명한다.
- 논리값과 논리 연산자 AND, OR, NOT, IMPLIES를 정의한다.
- 진리표, 항진 및 모순을 정의한다.
- 다양한 실세계 문제들을 해결하기 위하여 논리가 적용되는 방법을 설명한다.

❖ 논리 (Logic)

- 수많은 수학자 및 철학자들이 인간의 추론을 논리 시스템으로 정의
- 올바른 사고의 과학 (Science of correct thinking)
- 두개의 범주 – 귀납적 논리, 연역적 논리

❖ 귀납적 논리 (inductive logic)

- 관찰 또는 경험으로부터 어느 정도의 확실성을 가지고 결론이 도출
- 도출된 결론은 필연적으로 불확실

  - 결론이 기반을 두고 있는 경험들의 횟수와 직접적으로 관련하기 때문

- 예) 방울양배추(brussels sprouts)를 2번 먹은 후 병이 났다

  → 결론: 방울양배출에 알레르기가 있다.

  → 방울양배추를 먹을 때마다 병이 났었다면, 결론의 확실성이 증가

  =➔ 경험의 횟수에 따라 결론의 확실성 증가

❖ **연역적 논리** (deductive logic)

- 확실하다고 생각되는 것은 절대적으로 참이라는 가정으로 시작
- 확실하다고 생각되는 것으로부터 다른 사실들도 절대적으로 참이라고 생각
- 가정이 참이라고 한다면 결론은 확실한 참
- 예) 모든 인간은 죽는다. 아리스토텔레스는 인간이다

  → 아리스토텔레스는 죽는다

- **전제** (premises) : 참이라고 가정
- **결론**(conclusion) : 전제로부터 논리적으로 따라가면 도출되는 참인 진술
- **삼단논법**(syllogism) : 두 개의 참인 전제에서 논리적으로 따라가면 도출되는 참인 결론을 포함하는 논리적인 주장

예) 전제 : 계산적 사고는 모든 학생들에게 유익하다.
전제 : 제미마 판스워스(Jemimah Farnsworth)는 학생이다.
결론 : 계산적 사고는 제미마 판스워스에게 유익하다.

❖ **기호 논리**(symbolic logic)
- 진실에 대한 진술을 표현하기 위해 기호를 사용

❖ **논리 연산자**
- 논리적 사고를 표현

❖ **부울 논리**(boolean logic)
- 수학자 조지부울에 의해만들어진 기호 논리 시스템
- 현대의 컴퓨팅 시스템은 올바르고 신뢰할 결과를 생성하기 위해 부울 논리 규칙에 의존함

❖ **명제**(proposition)

- 부울 논리의 기본 단위
- 참 또는 거짓이 될 수 있는 진술
- 예) "에베레스트 산이 세상에서 가장 높은 산이다."
- 기본 또는 합성 둘 중의 하나가 될 수 있음

❖ **논리 값**(logical values)/**진리값**(truth values)

- 부울논리의 사용 값: 참 또는 거짓

❖ **배중률**(law of the excluded middle)

- 논리시스템에서 참 또는 거짓인 진리 값만 존재

기본 명제(simple proposition)

- 부분들로 쪼개질 수 없는 명제

합성 명제(compound proposition)

- 기본 명제들을 논리 연산자로 결합
- 논리 연산자 : **and, or, implies, not**

명제를 약어로 사용

- P = "I am hungry.", Q = "I am cold."

논리 연산자를 사용

- "I am hungry and I am cold." → P and Q

❖ **잘 작성(well formed)된 명제**
- 명제가 의미를 가지기 위해 잘 작성되어야 함
- 다음의 규칙을 따름

❖ **규칙 1 – 다음의 각각은 기본 명제이다.**

a. 어떤 단일 문자
b. 참
c. 거짓

❖ **규칙 2 – 박스(□)가 명제를 나타낸다고 하자.**
박스가 어떤 명제라고 가정, 다음의 각각 또한 명제이다.

a. □ and □
b. □ or □
c. □ implies □
d. □ ≡ □
e. Not □
f. (□)

❖ P and not (Q or R)은 잘 작성된 명제인가?

- P and not (Q or R) → □ and not (□ or □)(규칙 1a)
- □ and not (□ or □) → □ and not (□)(규칙 2b)
- □ and not (□) → □ and not □(규칙 2f)
- □ and not □ → □ and □(규칙 2e)
- □ and □ → □(규칙 2a)
- 따라서 주어진 명제는 잘 작성되었고 의미 있는 명제

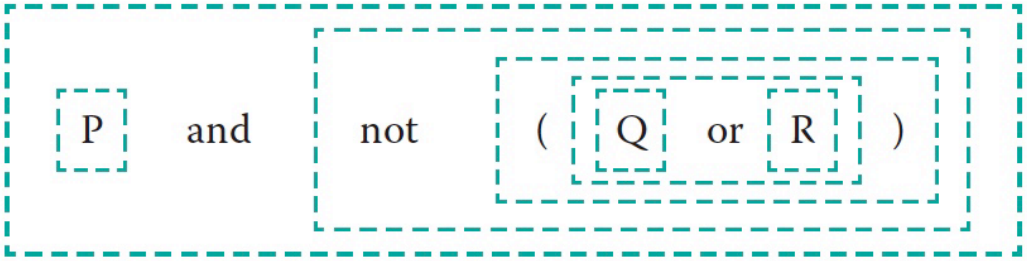

그림 3.1 각 점선 박스는 전체의 구성요소인 부분 명제에 해당하기 때문에 부울 명제 "P and not (Q or R)"은 잘 작성된 것으로 증명된다.

❖ **문자들과 연산자들의 일부 조합은**
- 의미있는 부울 명제처럼 보이지만 실제로 기호들의 의미 없는 혼합

❖ **P not Q은 잘 작성된 명제 인가?**
- P not Q → □ not □(규칙 1)
- □ not □ → □□(규칙 2e)
- □□을 단일 박스로 축소해줄 어떤 규칙도 없음

➔ 결론: "P not Q"는 명제가 아니다

- **명제판별의 목적:**
  잘 작성되었는가가 아니고 명제의 진리값을 결정하는 것
- **연산자**(operator): 입력 받고, 입력값을 처리하고, 단일 출력값을 만들어내는 기계같은 것
- **연산자의 애리티**(arity): 연산자에 대한 입력의 개수
- **이항 연산자**(binary operator): 애리티가 2인 연산자
- **단항 연산자**(unary operator): 애리티가 1인 연산자

연산자	기술적인 이름	애리티	사용 예
and	논리곱	2 (이항)	P and Q
or	논리합	2 (이항)	P or Q
implies	함축	2 (이항)	P implies Q
≡	동치	2 (이항)	P≡Q
not	논리적 부정	1 (단항)	not P

그림 3.2 논리 연산자

❖ **진리표**(truth table)
- 논리 연산자는 진리표로 정의됨

❖ **논리곱**
- 양쪽 입력 모두가 참일 때만 참값

논리곱		
P	Q	P and Q
거짓	거짓	거짓
거짓	참	거짓
참	거짓	거짓
참	참	참

그림 3.3 논리곱에 대한 진리표

❖ 논리합

- 입력들 중 어느 하나가 참이면 우리는 결과를 참으로 정의

논리합		
P	Q	P or Q
거짓	거짓	거짓
거짓	참	참
참	거짓	참
참	참	참

그림 3.4 논리합에 대한 진리표

3.2.2.3 함축(Implication: IMPLIES)

- **함축: 하나가 참이면, 논리적 필연성에 의해 일부 다른 것들도 참이어야 한다는 개념**
- **"P implies Q" 라는 명제 ($p \rightarrow q$)**
  - 전건(antecedent) : P // 가정, 전제조건
  - 후건(consequent) : Q // 결과

함축		
P	Q	P implies Q
거짓	거짓	참
거짓	참	참
참	거짓	거짓
참	참	참

그림 3.5 논리 함축에 대한 진리표

3.2.2.3 함축(Implication: IMPLIES)

❖ **P: My car battery is dead , Q: My car won't start**

1. P는 참이고 Q는 참일때 :
   - 차량 밧데리가 다 떨어져서 자동차의 시동이 걸리지 않을 것이기 때문에 "P implies Q"임을 주장하는 것이 참이다.

2. P는 참이고 Q는 거짓일때 :
   - (관찰) 차량 밧데리는 다 떨어졌고 자동차는 시동이 걸린다
   - P implies Q : 차량 밧데리가 다 떨어질때 마다 자동차는 시동이 걸린다고 주장하는 명제

   ➔ 논리적으로 일치하지 않음: 명제는 거짓

P: My car battery is dead , Q: My car won't start

3. P는 거짓이고 Q는 참일때 :

- 차량 밧데리가 다 떨어지지 않았지만, 차량 시동이 걸리지 않음. (아마도 자동차는 다른 이유로 시동이 걸리지 않을 수도 있음.)
- 이것은 "My car battery is dead implies my car won't start"라는 진술과 논리적인 모순이 없음

➔ 명제는 참

4. P는 거짓이고 Q는 거짓일때 :

- "차량 밧데리가 다 떨어지지 않았고 차량 시동이 걸린다."에 논리적인 모순이 없기 때문

➔ 함축은 참

3.2.2.3 함축(Implication: IMPLIES)

❖ **"P implies Q"의 역(converse) → "Q implies P"**

❖ **함축이 참일 때 명제의 역은 본질적으로 논리적인 반대임**

- 함축이 참이면 그역은 참이 아니다
- 예) "Humphrey (험프리) is a dog implies Humphrey is a mammal (포유동물)."

→ "Humphrey is a mammal implies Humphrey is a dog."(역, 거짓)

"Humphrey is a mammal"이 "Humphrey is a dog"을 의미하지는 않기 때문

❖ 동치 : 변수 P와 Q가 동일한 진리값을 가지는 경우

동치		
P	Q	P ≡ Q
거짓	거짓	참
거짓	참	거짓
참	거짓	거짓
참	참	참

그림 3.6 논리 동치에 대한 진리표

❖ 단항 연산자

부정	
P	not P
거짓	참
참	거짓

그림 3.7 논리적 부정에 대한 진리표

❖ **명제를 판별하는 목적: 명제의 진리값을 결정**

❖ **진리표 참조해서 명제를 판별함**

예) P=참, Q=거짓, R=참이 주어질 때

P and (Q or R)

→ P and True ((Q or R)는 True 이므로)

→주어진 명제는 True

3.2.2.6 합성 명제(Compound Propositions)

- **보통 왼쪽부터 오른쪽으로 진행**
- **괄호를 사용하는 경우 우선적으로 연산자를 판별함**
- **괄호 때문에 그 진리값이 달라지는 경우**

  예) P=참, Q=참, R=거짓인 경우

  P and Q implies R : 참

  P and (Q implies R) : 거짓

3.2.2.6 합성 명제(Compound Propositions)

예) P and not (Q or R)

- 진리표를 작성해서 풀이
- 논리 변수 P, Q 그리고 R 각각에 대하여 하나의 열을 가지도록 하고 마지막 열은 전체 명제를 가지도록 함
- 3개의 변수들이 있기 때문에 적어도 8개(23)의 행을 작성함
- 각 행은 P, Q, 그리고 R이라는 변수들에 대한 값들의 유일한 조합이 되도록 함

3.2.2.6 합성 명제(Compound Propositions)

P	Q	R	(Q or R)	not (Q or R)	P and not (Q or R)
거짓	거짓	거짓	거짓	참	거짓
거짓	거짓	참	참	거짓	거짓
거짓	참	거짓	참	거짓	거짓
거짓	참	참	참	거짓	거짓
참	거짓	거짓	거짓	참	참
참	거짓	참	참	거짓	거짓
참	참	거짓	참	거짓	거짓
참	참	참	참	거짓	거짓

그림 3.9 진리표를 생성하는 과정

❖ **동치(equivalent) : 두 개의 명제가 동일한 진리표를 가짐**

- 예) not (P or Q)
  (not P) or (not Q)

P	Q	not (P and Q)
거짓	거짓	참
거짓	참	참
참	거짓	참
참	참	거짓

P	Q	(not P) or (not Q)
거짓	거짓	참
거짓	참	참
참	거짓	참
참	참	거짓

그림 3.11 두 개의 다른 명제가 동일한 진리표를 가진다. 그러므로 이러한 두 개의 명제들은 동치라고 한다.

3.2.2.8 항진과 모순(Tautologies and Contradictions)

❖ **항진** (tautology)
- 입력에 관계없이 참의 값을 가지는 명제
- 예) P or not P

❖ **모순** (contradiction)
- 모든 입력에 대하여 거짓의 값을 가지는 명제
- 예) P and not P

❖ **무모순의 법칙** (law of noncontradiction)
- 한 명제가 동시에 거짓과 참 둘 다가 될 수 없다

3.3.1.1 검색 질의에서의 논리곱

❖ **검색 질의** (search query)
- 사용자가 찾고자 하는 정보를 설명

❖ **고등학교 동창 Hannah Garcia를 찾기위해 Hannah Garcia를 입력하면 Hannah AND Garcia로 이해함**

❖ **"Hannah"와 단어 "Garcia" 둘 다를 포함하는 검색결과를 보여줌**

3.3.1.2 검색 질의에서의 논리합

- 결혼 후 Mason 성을 가졌지만 Garcia성을 그대로 쓰기도 하는 경우
- "Garcia" OR "Mason"를 입력하면 가르시아 또는 메이슨 중의 하나만 포함하는 웹페이지를 찾음
- Hannah AND (Garcia OR Mason)를 입력

3.3.1.3 검색 질의에서의 부정

❖ **NOT 연산자**
- 어떤 단어를 포함하는 페이지를 배제

❖ **고등학교 동창은 축구선수가 아님을 확신하므로 Hannah AND (Garcia OR Mason) AND NOT soccer"를 입력**

❖ **이 질의는 "soccer"라는 단어를 포함하는 페이지는 배제하고 "Hannah" 와 "Garcia"라는 단어들이나 또는 "Hannah"와"Mason"라는 단어들 중에서 한쪽을 포함하는 모든 페이지를 반환**

➔ **Hannah AND (Garcia OR Mason) AND –soccer**

❖ **대부분 검색엔진에서 논리 연산자들은 모두 대문자로 입력**

❖ **논리 게이트** (logic gate)

- 부울 연산자를 구현하는 전자 장치
- 입력들을 받아들여 연산자에 해당하는 단일 출력을 발생

❖ **건전지는 전압 레벨 1로 표시**

❖ **회로는 0(전력이 없음) 또는 거짓 , 1(전체 전력) 또는 참으로 표시**

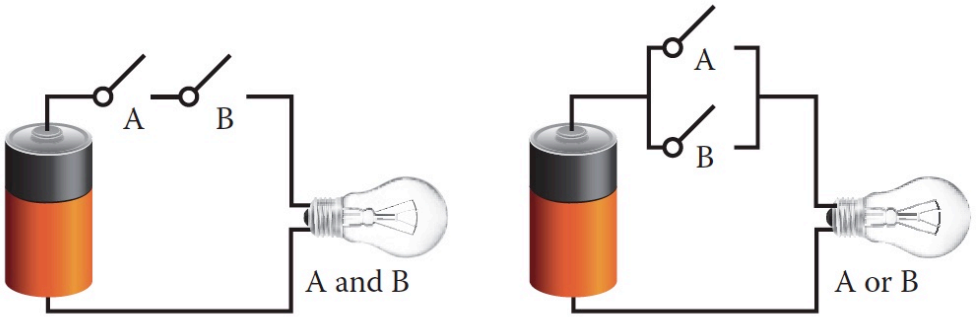

그림 3.12 스위치들이 논리 입력을 나타내고 전구의 상태가 출력을 표시하는 전기 회로

- P and not (Q or R)를 논리 게이트로 표현하기
- 논리회로에서 괄호로 묶인 "or" 게이트가 먼저 동작
- 가능한 한 적은 수의 게이트들을 사용하는 걸 더 선호

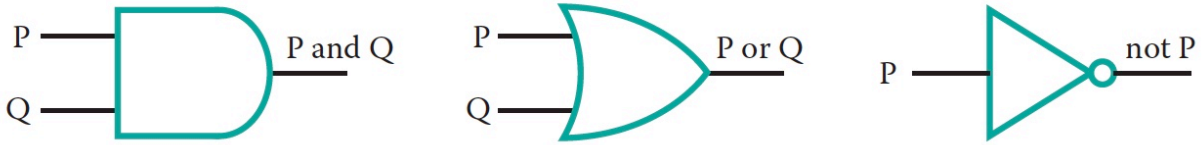

그림 3.13 부울 논리 게이트 AND, OR 및 NOT에 대한 회로 표기

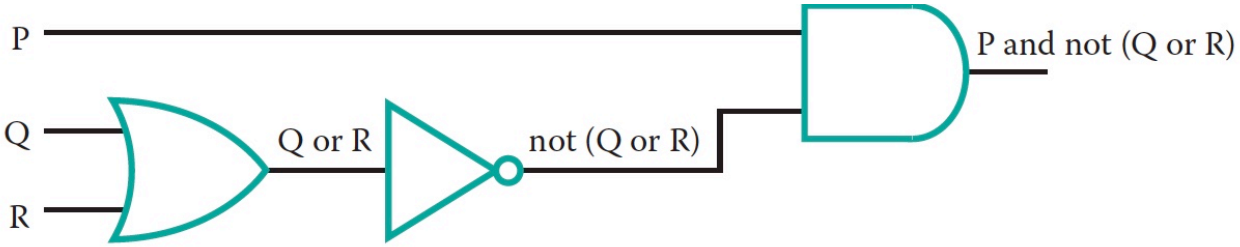

그림 3.14 술어 "P and not (Q or R)"을 구현하기 위해서 논리 게이트 사용하기

❖ P와 Q가 동치일때만 출력이 참인 논리회로

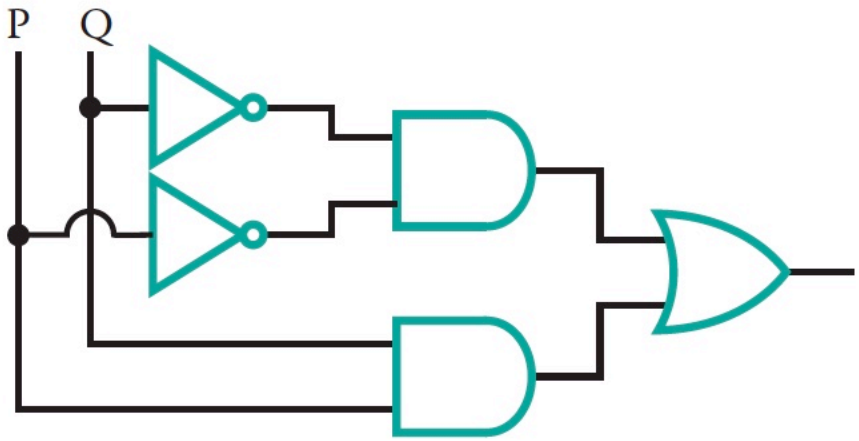

그림 3.15 동치회로. P와 Q가 동치일 때만 출력이 1(참)이다.

❖ **디지털 영상** (digital images)
- 픽셀로 이루어짐

❖ **이진 영상** (binary images)
- 흰색 또는 검정색 픽셀들로 이루어진 디지털 영상

❖ **디지털 합성** (digital compositing)
- 2개의 형상이나 영상을 결합
- 진리값 대신 색상 사용

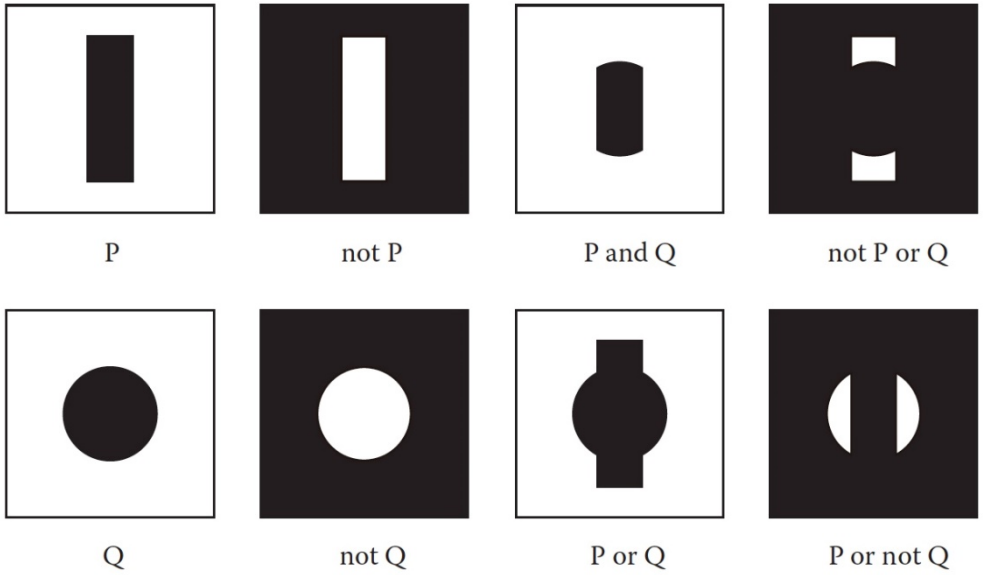

그림 3.18 논리 연산자를 사용한 영상 합성

3.3.4 데이터베이스 질의

- **데이터베이스 (database: DB)는 많은 양의 데이터를 효율적으로 저장하기 위해 설계된 소프트웨어 시스템**
  - DB는 테이블에 정보를 저장
- **레코드**(record)
  - 데이터베이스 테이블의 각 행
- **필드**(field)
  - 테이블의 각 열
- **구조적 질의 언어**(Structured Query Language: SQL)
  - 데이터베이스 질의는 보다 명확한 구조를 요구하는 SQL 언어로 작성

예) **SELECT** field1, field2, ..., fieldN **FROM** table **WHERE** criteria

예) **SELECT** First, Last **FROM** Donor **WHERE** Age < 40
AND Amount >= 500 **AND** (State ='NY' **OR** State ='CA')
뉴욕주 또는 켈리포니아주에 살고 있는, 과거에 최소한 $500를 기부했던, 40세 이하의 모든 기부자들을 찾음

❖ **요구공학**(requirements engineering)

- 소프트웨어가 무엇을 해야 하는지를 정의
- 소프트웨어 요구 문서(software requirements document)로 표현

❖ **논리 명제를 사용하여 소프트웨어의 중요한 기능들을 정의**

- (FLOOR=1 OR FLOOR=2) AND (NOT MOVING) AND BUTTON_PUSHED IMPLIES FRONT_DOOR_OPENS
- (FLOOR=3 OR FLOOR=4) AND (NOT MOVING) AND BUTTON_PUSHED IMPLIES REAR_DOOR_OPENS
- (EMERGENCY_KEY_INSERTED AND BUTTON_PUSHED) IMPLIES (FRONT_DOOR_OPENS AND REAR_DOOR_OPENS)
- 그 외의 경우에 시스템은 아무것도 하지 않는다.

❖ **안전중심 시스템** (safety-critical systems)

- 올바로 동작하지 않는다면 심각한 해를 끼칠 수도 있는 소프트웨어 시스템
- 수학을 사용하여 명시
- **정형 기법** (formal methods) : 안전중심 소프트웨어 시스템의 명세와 개발을 위한 수학적으로 정밀한 기술들

- 문제 1
  - 피보나치가 쓴 책에서 소개된 피보나치 수열은 알고리즘 시간에 가장 많이 만나게 되는 문제 중 하나이다. 이러한 피보나치 수열이 무엇인지 알아보고, 이 피보나치 수열이 왜 많은 사람들이 관심을 가지게 되는지에 대해 조사하시오.

- 문제 2
  - 당신 앞에 천국문과 지옥문이 있다. 이 중 천국문으로 들어가야 하는데, 두 문은 똑같이 생겨서 어느 곳이 천국문인지 지옥문인지 알 수 없다. 두 문 앞에 천사와 악마가 서 있고 역시 둘다 똑같이 생겨 누가 천사인지 악마인지 모르는 상황이다. 악마에게 질문을 하면 항상 거짓말로 대답하고 천사는 항상 옳은 말만 한다. 천사와 악마 중 단 한명에게만 한번 질문해서 천국에 갈 수 있는 방법은 무엇일까?