5주차 예비보고서

전공: 컴퓨터공학과 학년: 3학년 학번: 20212022 이름: 이예준

**1.**

De-Morgan의 정리는 AND, OR 또는 NOT 사이의 관계를 나타내며

논리식 전체에 NOT을 적용시키면 AND는 OR로, OR은 AND로 반전되고 각 변수들도 동시에 NOT이 씌워지는 정리이다.

De-Morgan의 정리는 두가지 조건, 다른 말로 두가지 법칙이 존재한다.

제 1법칙 : (A B)’ A’ B’

제 2법칙 : (A B)’ A’ B’

도표, 라인, 폰트이(가) 표시된 사진

자동 생성된 설명

\*제 1법칙과 제 2법칙

이 법칙을 맞다는 것을 보이기 위해 진리표와 증명을 이용해본다.

-진리표

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| A | B | A’ | B’ | (A B)’ | A’ B’ | (A B)’ | A’ B’ |
| 0 | 0 | 1 | 1 | 1 | 1 | 1 | 1 |
| 0 | 1 | 1 | 0 | 1 | 1 | 0 | 0 |
| 1 | 0 | 0 | 1 | 1 | 1 | 0 | 0 |
| 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 |

\*제 1법칙과 제 2법칙의 진리표

진리표를 보면 알 수 있듯이 제1,2 법칙이 성립하다는 것을 알 수 있다.

-증명

R = (A B)’ , S = A’ B’이라고 두자.

y를 R의 임의의 요소라고 했을 때 y R = y (A B)’이다.

따라서 y (A B)’

⇒ y ∉ (A B)

⇒ y ∉ A or y ∉ B

⇒ y A' or y B'

⇒ y A' B'

⇒ y S ……………………… (1)

가 되므로 R S가 성립한다.

x를 S의 임의의 요소라고 했을 때 x S = x A' B'이다.

따라서x A' B'

⇒ x A' or x B'

⇒ x ∉ A or x ∉ B

⇒ x ∉ (A B)

⇒ x (A B)’

⇒ x R ……………………… (2)

가 되므로 S R이 성립한다.

(1)과 (2)를 결합하게 되면 S = R = (A B)’ = A’ B’ 이라고 말할 수 있다.

**2.**

Boolean 대수를 이용하여 하나 이상의 입력값을 논리계산하여 하나의 출력값을 얻는 전자회로를 논리회로라고 한다. 논리회로는 기본 논리게이트(AND, OR, NOT)를 조합해

복합적인 논리기능을 제공한다. 따라서 논리회로는 기본적으로 논리식의 조합으로 나타나기 때문에 이런 논리식을 간소화함으로써 더 적은 게이트를 이용해 효율적인 회로를

구성할 수 있다. 대표적인 방법으로는 Boolean 대수의 기본정리, 카르노 맵, 퀸-맥클러스키 방법 등이 있다.

|  |  |
| --- | --- |
| 동일 법칙(Identity Laws) | X 1 = X |
| X + 0 = X |
| 지배 법칙(Domination Laws) | X 0 = 0 |
| X + 1 = X |
| 등멱 법칙(Idempotent Laws) | X X = X |
| X + X = X |
| 부정 법칙(Negation Laws) | X X’ = 0 |
| X + X’ = 1 |
| 이중부정 법칙(Double Negation Laws) | (X’)’ = X |
| 교환 법칙(Commutative Laws) | X Y = Y X |
| X + Y = Y + X |
| 결합 법칙(Associative Laws) | X (Y Z) = (X Y) Z |
| X (Y Z) = (X Y) Z |
| 분배 법칙(Distributive Laws) | X (Y Z) = X Y + X Z |
| X + Y Z = (X + Y) (X + Z) |
| 드 모르간 법칙(De-Morgan’s Laws) | (X Y)’ = X’ + Y’ |
| (X Y)’ = X’ Y’ |
| 흡수 법칙(Absorption Laws) | X + X Y’ = X |
| X + X’ Y = X + Y |

\* Boolean 대수의 기본정리표

예시) Z = [A + B’C + D + EF][ A + B’C + (D + EF)’]

= (A + B’C) + (D + EF) ( D + EF)’ <분배 법칙>

= A + B’C <부정 법칙>

**3.**

Boolean 대수의 기본 정리는 복잡하고 실수할 가능성이 있어 간소화되었는지 검증도 어렵다.

그래서 논리식을 빠르게 간소화하는 방법 중 하나로 카르노 맵을 사용한다.

\*(그렇다고 카르노 맵을 통해 얻은 식이 가장 간소화된 식이라고 장담할 수 없다.)

먼저 카르노 맵을 그릴 때는 변수의 개수에 따라 나올 수 있는 모든 경우의 수를 표에 표현한다. 2변수일때는 4칸, 3변수일 때는 8칸 또는 4변수일 때는 16칸을 그려준다.

그 다음 카르노 맵을 사용하는 데는 몇가지 규칙이 있다.

1. 출력이 같은 항끼리 반드시 직사각형 또는 정사각형으로 최대한 크게 2n개씩 묶는다.

2. 바로 이웃한 항들끼리 묶는다.

3. 중복하여 묶어 간소화된다면 묶는다.

예시) F = AB’C + A’B’C + A’BC + AB’C’ + A’B’C

|  |  |  |  |
| --- | --- | --- | --- |
| A | B | C | AB’C + A’B’C + A’BC + AB’C’ + A’B’C |
| 0 | 0 | 0 | 1 |
| 0 | 0 | 1 | 1 |
| 0 | 1 | 0 | 0 |
| 0 | 1 | 1 | 1 |
| 1 | 0 | 0 | 1 |
| 1 | 0 | 1 | 1 |
| 1 | 1 | 0 | 0 |
| 1 | 1 | 1 | 0 |

\*진리표

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A BC | 00 | 01 | 11 | 10 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 |

\*카르노 맵

변수는 3개이기 때문에 8개의 칸을 그려주었고, 모든 경우의 수를 맵에 적어주었다.

이제 앞서 설명한 사용규칙에 맞게 ‘1’을 묶어준다.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| A BC | 00 | 01 | 11 | 10 |
| 0 | 1 | 1 | 1 | 0 |
| 1 | 1 | 1 | 0 | 0 |

\*카르노 맵

빨간색의 상자를 나타내면 B’C’가 고정이고 B’C가 고정이다. 따라서 B’C’ + B’C 이다.

이를 분배 법칙과 부정 법칙을 이용해 한번 더 정리해주면 B’(C’ + C) = B’ 이다.

파란색의 상자를 나타내면 A’C가 고정이다. 따라서 A’C 이다.

최종적으로 간소화된 식은 F = B’ + A’C가 된다.

\*다른 값들이 어떤 값을 취하든 변하지 않는 값들이 있다면

그 고정된 값만 있어도 논리식에는 아무런 문제가 없기 때문에 고정된 값만 남겨두고

나머지는 버려도 된다.

**4.**

변수가 적을 때는 카르노 맵이 효과적인 방법이지만 5변수 이상이거나 여러 함수를

간소화할 때는 카르노 맵을 그리기 어려워Quine-McCluskey 방법을 사용하는 것이

바람직하다. 카르노 맵은 그림을 그렸지만 Quine-McCluskey방법은 도표를 이용해

간소화한다. 또한 컴퓨터를 이용해 프로그램을 만들 수 있는 체계적인 간소화 과정을

가지고 있다.

Quine-McCluskey방법은 다음과 같은 과정을 거친다.

1. 이진 표현으로 나타내는 최소항들의 1의 개수에 따라 그룹을 만들고

그 그룹내에 있는 수들은 모두 오름차순으로 정렬한다.

2. 서로 이웃하는 항들에 대해서만 비교하며 두 개의 항이 서로 1비트만 다르다면 결합한다.

3. 결합할 때 서로 다른 비트는 ‘-‘기호로 대체하고 나머지 비트는 동일하게 유지한다.

4. 2~3단계를 모든 최소항이 두 비트 차이 나거나 중복될 때까지 반복한다.

중복항들은 제거한다.

5. 이렇게 구한 항들을 ‘주항’이라고 하며 이 주항과 최소항을 가지고 테이블을 만든다.

열에는 주항을, 행에는 최소항을 놓는다.

6. 주항에 최소항이 포함되어 있다면 그 주항과 최소항의 교차점에 ‘1’을 적는다.

이때 해당 최소항이 포함되어 있는 주항이 오직 하나라면 필수주항이라고 하며

간소화된 식에 반드시 포함되어야 한다.

7. 필수 주항이 있는 최소항들은 그 행을 삭제하고 그 삭제하는 곳에 1이 있다면

그 열도 제거한다. 그렇게 제거했을 때 1이 전혀 지워지지 않은 최소항과 필수주항이

간소화식의 항들이 된다.

예시) F(A,B,C,D) =∑ m(0,1,2,4,6,8,9,11,13,15)

|  |  |  |
| --- | --- | --- |
| 그룹 | 최소항 | 항 |
| 그룹 1 | 0 | 0000 |
| 그룹 2 | 1 | 0001 |
| 2 | 0010 |
| 4 | 0100 |
| 8 | 1000 |
| 그룹 3 | 9 | 1001 |
| 그룹 4 | 11 | 1011 |
| 13 | 1101 |
| 그룹 5 | 15 | 1111 |

\*한 비트 차이나는 항들끼리 묶기

|  |  |  |
| --- | --- | --- |
| 그룹 | 최소항 | 항 |
| 그룹 1 | 0,1 | 000- |
| 0,2 | 00-0 |
| 0,4 | 0-00 |
| 0,8 | -000 |
| 그룹 2 | 1,9 | -001 |
| 2,6 | 0-10 |
| 4,6 | 01-0 |
| 8,9 | 100- |
| 그룹 3 | 9,11 | 10-1 |
| 9,13 | 1-01 |
| 그룹 4 | 11,15 | 1-11 |
| 13,15 | 11-1 |

\*결합하기

|  |  |  |
| --- | --- | --- |
| 그룹 | 최소항 | 항 |
| 그룹 1 | 0,1,8,9 | -00- |
| 0,2,4,6 | 0--0 |
| 그룹 2 | 9,11,13,15 | 1--1 |

\*결합하기

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 4 | 6 | 8 | 9 | 11 | 13 | 15 |
| (0,1,8,9) | 1 | 1 |  |  |  | 1 | 1 |  |  |  |
| (0,2,4,6) | 1 |  | 1 | 1 | 1 |  |  |  |  |  |
| (9,11,13,15) |  |  |  |  |  |  | 1 | 1 | 1 | 1 |

\*테이블

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | 0 | 1 | 2 | 4 | 6 | 8 | 9 | 11 | 13 | 15 |
| (0,1,8,9) | 1 | 1 |  |  |  | 1 | 1 |  |  |  |
| (0,2,4,6) | 1 |  | 1 | 1 | 1 |  |  |  |  |  |
| (9,11,13,15) |  |  |  |  |  |  | 1 | 1 | 1 | 1 |

\*제거

따라서 Quine-McCluskey방법에 따라 간소화된 식은 F = **B'C' + A'D' + AD이 된다.**

**5.**

Boolean 변수들로 이루어진 Boolean Expression이 있을 때

각 변수에 대해서 True, False결정을 해야 할 때, 전체 식의 결과를 True로 만들 수 있느냐 에 대한 문제를 충족 가능성 문제(Satisfiability Problem)라고 한다.

만족성 문제, 만족도 문제, 만족 문제, 불린 충족 가능성 문제

(Boolean Satisfiability Problem)라고도 부른다.

그 중 괄호안에 각 절에 대해 최대 두가지 변수가 존재할 때 2-SAT, 최대 세가지 변수가 존재할 때 3-SAT라고 한다.

예를 들어 **f = (B ∨ A’)∧(A’ ∨ B’)∧(A ∨ C)∧(B’ ∨ C’)∧(A ∨ D)**

라는 식이 있을 때 A는 false, B는 false, C는 true 그리고 D는 true인 경우에

참이 된다.