14주차 예비보고서

전공: 기계공학과 학년: 3학년 학번: 20191820 이름: 김형준

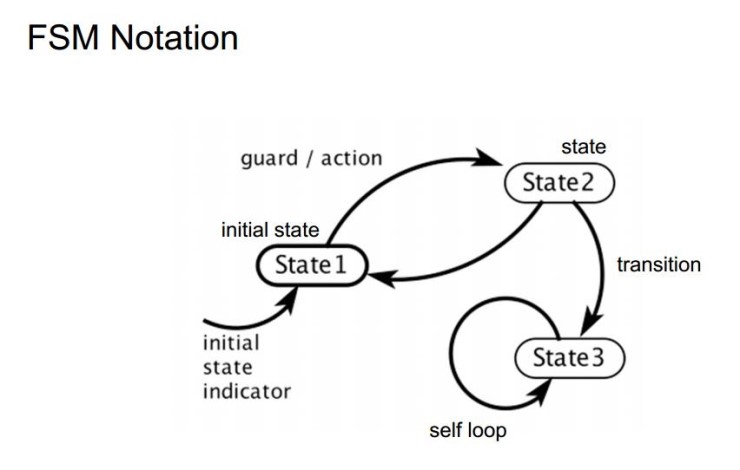
**1.**

FSM(finite state machine)은 유한 상태 기계 또는 유한 오토마타(finite automata)를 의미하며, 유한한 기억장치를 갖는 자동기계에 대한 추상적인 모형이다.

FSM은 컴퓨터 프로그램과 전자 논리 회로를 설계하는데 유용하게 사용되는 수학적 모델이며, 여러 개의 제한된 상태(State)가 존재하고, 그 상태들이 특정 조건에 물려 서로 연결된 형태로 이루어져 있다.

FSM은 주어진 모든 시간에 처해 있을 수 있는 유한개의 상태를 가지고, 한번에 하나의 상태(State)만을 가질 수 있으며, 이 때의 FSM의 상태를 현재 상태(Current State)라고 한다.

FSM의 상태가 변화하는 것을 전이(Transition)이라 하며, 상태간 전이를 유발시키는 사건(입력)을 이벤트(Event)라 하고, 행동(출력)을 내놓는다.

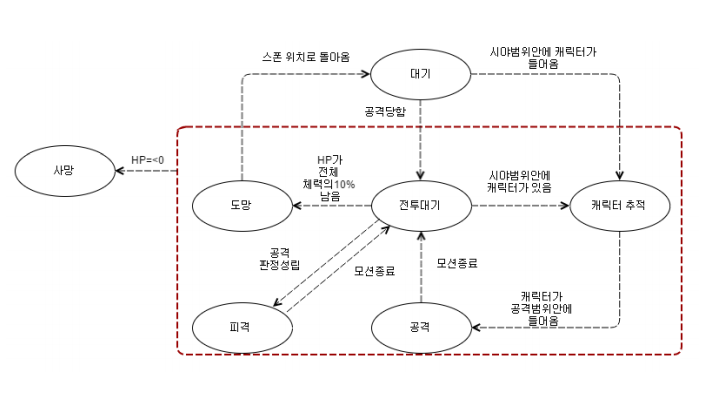


FSM의 각각의 상태들을 하나의 vertex로 두고 상태 간의 이동경로를 edge로 표현하면 위와 같이 directed graph가 되는데, 이 그래프를 상태도(state diagram)이라 한다.

FSM을 상태도를 사용하여 나타내면, 순차회로에서 각각의 상태를 구분하고 입력에 따른 상태변화와 현재 상태 및 입력에 대한 출력을 도식화 할 수 있다.

FSM에는 Moore FSM과 Mealy FSM이 있는데, Moore FSM은 출력이 현재 상태(Current State)에 따라서 출력이 결정되고, Mealy FSM은 입력과 현재 상태 모두에 의해 출력이 결정된다. Mealy FSM을 사용하면 Moore FSM보다 state의 개수를 줄일 수 있는 이점이 있는 경우가 있으나(Sequence Detector 등), 시스템이 복잡해질수록 Mealy FSM은 설계의 어려움이 커져 일반적으로 Moore FSM을 사용한다.

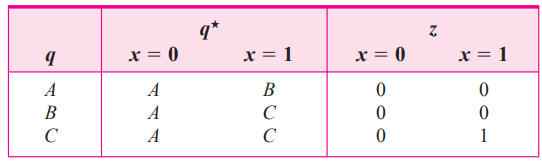
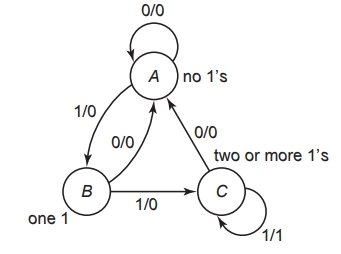
FSM은 여러 분야에서 많이 쓰이나, 특히 게임 분야에서 많이 사용된다.



예를 들어, RPG 게임 등에서 몬스터의 AI를 설계할 때 위와 같이 FSM을 사용할 수 있는데, 플레이어의 행동 및 기타 조건들에 따라 이벤트가 발생하여 몬스터의 상태가 달라지고, 상태에 따른 행동을 하는 것을 볼 수 있다. 이처럼 FSM을 사용하면 설계한 machine의 동작을 프로그래머 외에 기획자 또는 제 3자가 쉽게 이해할 수 있다는 장점이 있다.

FSM은 머신의 행동을 직관적이고 빠르게 구현할 수 있고 각각의 상태는 나눠져 있기 떄문에 새로운 상태의 추가 및 삭제가 용이해 유연성이 높고 오류의 수정이 용이하다는 장점이 있지만, 규모가 커지면 설계가 복잡해지고 제한된 범위의 문제에만 적용이 가능하다는 단점이 있다.

**2.**

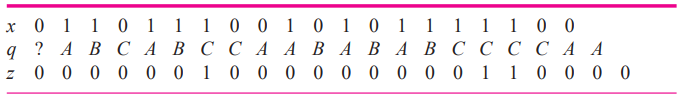


Mealy Machine은 FSM의 한 종류로, 출력값이 현재 상태와 입력값 모두에 의존한다.

위는 3번 연속으로 입력으로 1이 들어왔을 때만 출력으로 1을 갖는 Mealy Machine이다.

출력 z가 입력 x에 의해 값이 달라지는 점에서 이 machine이 Mealy Machine인 것을 확인할 수 있다.

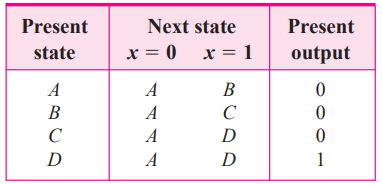
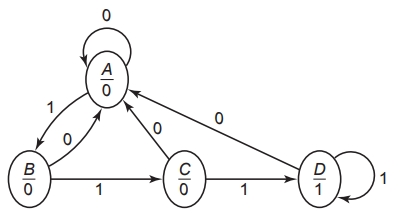
Mealy Machine은 같은 기능을 수행하는 Moore Machine보다 더 적은 개수의 state를 필요로 한다. 위의 Mealy Machine은 3개의 state만을 필요로 하지만, 후술할 같은 기능을 수행하는 Moore Machine은 4개의 state가 필요하다는 점에서 이를 확인할 수 있다.



또한, Mealy Machine은 Moore Machine보다 사용자 입력에 더 빠르게 반응하는데, Mealy Machine은 클럭 펄스를 한 사이클 기다리지 않고, 같은 사이클에서 input을 바로 output에 반영시킨다. 위의 Timing trace를 보면, 1이 연속으로 3번째 들어오는 순간, 즉시 출력으로 1이 나오는 것에서 이를 확인할 수 있다.

그러나, Mealy Machine은 이해가 어렵고 직관적이지 못해 설계가 어렵고, 입력에 의해 출력결과가 바뀔 수 있어 노이즈나 글리치(Glitch)같은 오동작 요소에 취약하다는 단점이 있다.

**3.**

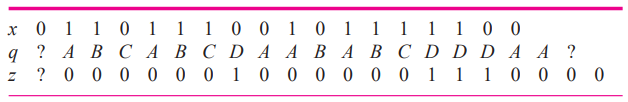


Moore Machine은 FSM의 한 종류로, 출력값이 현재 상태에만 의존한다.

위는 아까처럼 3번 연속으로 입력으로 1이 들어왔을 때만 출력으로 1을 갖는 Moore Machine이다.

출력 z가 입력 x에 관계없이 값이 Present state에 의해서만 결정되므로 이 machine이 Moore Machine인 것을 확인할 수 있다.

Moore Machine은 같은 기능을 수행하는 Mealy Machine보다 더 많은 개수의 state를 필요로 한다. 위의 Moore Machine은 4개의 state를 필요로 하지만, 이전의 같은 기능을 수행하는 Mealy Machine은 3개의 state가 필요하다는 점에서 이를 확인할 수 있다.



또한, Moore Machine은 Mealy Machine보다 사용자 입력에 더 느리게 반응하는데, Moore Machine은 출력값이 현재 상태에만 의존하므로, 현재 상태가 바뀌는 다음 사이클에서 input을 output에 반영한다.

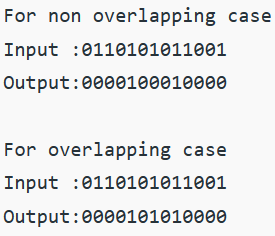
위의 Timing trace를 보면, 1이 연속으로 3번째 들어오는 순간에서 바로 출력값이 1이 되는 것이 아닌, 현재 상태가 D로 바뀌는 다음 클럭 펄스에서 출력값이 1이 되는 것에서 이를 확인할 수 있다.

Moore Machine은 설계가 직관적이고, 출력이 현재 상태에만 의존하므로 glitch issue에 비교적 안정하다는 장점이 있다. 또한, 시스템이 복잡해질수록 Mealy Machine은 설계의 어려움이 커져 일반적으로 Moore Machine을 설계에 사용한다.

**4.**

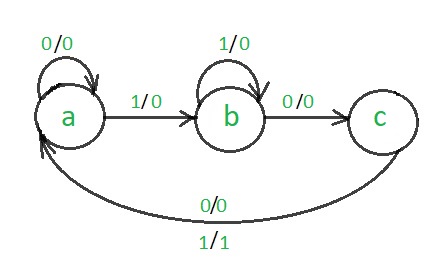
Sequence Detector(순차검출기)는 순차적 FSM의 일종으로, 입력 비트 스트링에서 target sequence가 감지될 때 마다 1을 출력하는 회로이다. 순차 검출기는 2가지 종류가 있는데, 중복(Overlapping)을 허용하는 경우와, 중복을 허용하지 않는(Non-Overlapping) 경우로 나눌 수 있다. 중복을 허용하지 않는 경우, target sequence로 감지된 비트들은 다음에 감지되는 target sequence에 포함될 수 없다. 예를 들어 101 Sequence 을 감지하는 경우, Mealy Machine으로 구현한 순차 검출기의 출력은 중복의 허용 여부에 따라 같은 입력 비트 스트링에 대해 다음과 같이

출력값에 차이가 나타난다.

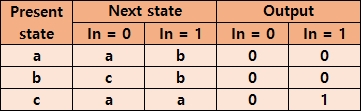


위 그림에서, 10101의 입력에 대해 중복을 허용하는 경우 뒤의 101도 target sequence로 감지하여 1을 출력하지만, 중복을 허용하지 않는 경우 뒤의 101중 첫번째 1이 앞의 101에 포함되므로 target sequence로 감지하지 않는다.

Sequence Detector의 예시로 Non-Overlapping 101 Mealy Sequence Detector를 설계하는 과정은 다음과 같다. 우선, Mealy machine을 만들기 위해 각 state를 정의하고 input과 next state를 고려하여 state diagram을 만든다. 초기 상태를 a, 1이 하나 들어온 상태를 b, 10이 들어온 상태를 c로 정의하고 state diagram을 그리면 아래와 같다.

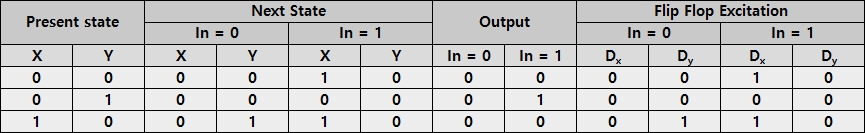


이를 state table로 나타내면 다음과 같고,



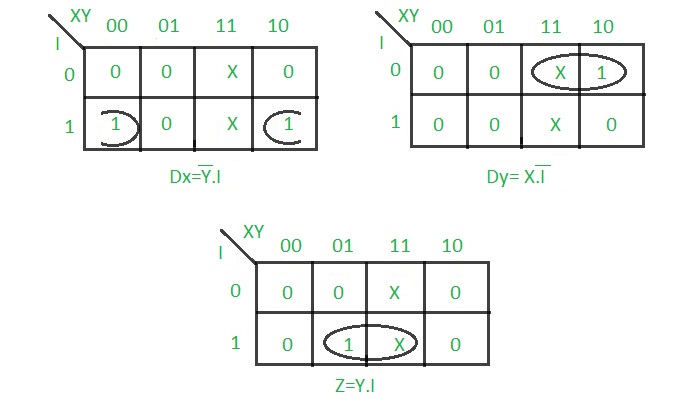
state의 개수가 총 3개이므로 state를 저장하기 위해서 2 bit의 플립플롭이 필요하다.

따라서 state a, b, c를 각각 으로 설정하고, D-FlipFlop 2개를 사용해 FlipFlop Excitation까지 같이 나타내면 다음과 같다.

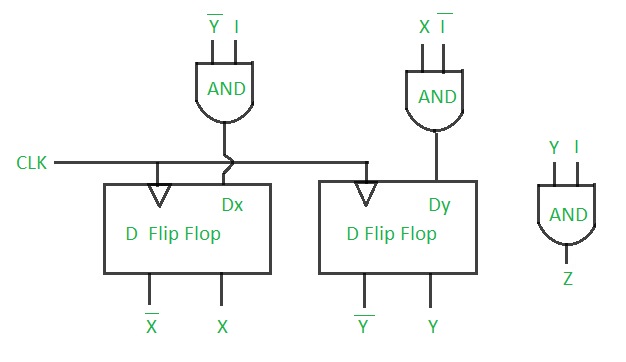


위 표에서 생략된 X=1, Y=1일때의 모든 값들은 don’t care로 처리한다.

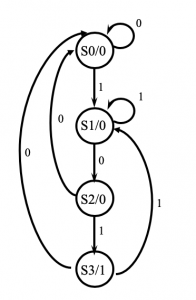
표의 값을 사용해 카르노맵을 그려 Output Z와 D-FlipFlop 의 부울 함수를 각각 구하면,



, , 을 구할 수 있고, 이를 통해 회로도를 그리면, 아래와 같이 나타낼 수 있다.

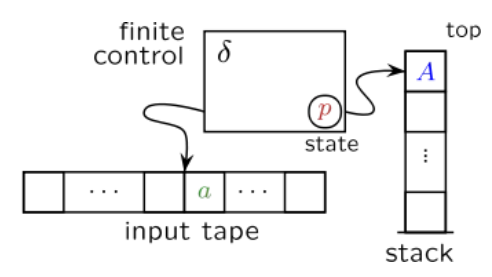


한편, Sequence Detector는 Moore Machine으로도 만들 수 있는데, 위에서 설계했던 Non-Overlapping 101 Sequence Detector를 Moore Machine으로 구현한 state diagram은 다음과 같다.



**5.**

- Pushdown Automata



Pushdown Automa(PDA)는 유한 오토마타의 확장으로, 유한 오토마타에 Stack 자료구조가 추가된 형태이다. 스택 자료구조가 추가되었기 때문에, 입력 심볼과 스택의 Top에 따라 State간 전이(Transition)가 일어난다.

푸시다운 오토마타는 7개의 요소를 가지는데, 유한 오토마타에서 2가지 개념()이 추가된 형태이다. 이 2가지 개념은 푸시다운 오토마타가 사용자의 입력을 저장할 수 있는 별도의 공간인 스택을 가지고 있기 때문에 추가되었다.

각 요소들에 대한 의미는 다음과 같다.

: 상태들의 유한 집합

: 입력 심볼의 유한 집합

: 스택 심볼의 유한 집합

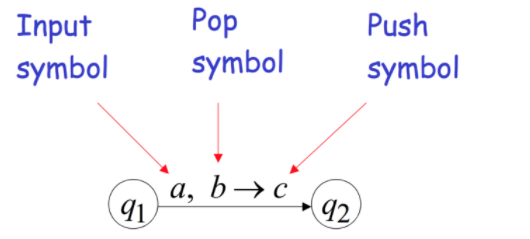
: 전이 함수

: 시작 상태

: 스택의 시작 심볼

: 종결 상태들의 집합

푸시다운 오토마타는 현재의 입력 심볼과 스택의 Top 심볼의 값에 따라 다음 행동을 결정하는데, 아래 그림에서 입력 심볼로 a가 들어오고, 스택의 Top 심볼이 b일 때 c로 변경한 다음, state를 에서 로 바꾸는 것을 확인할 수 있다.



푸시다운 오토마타는 전이함수에 따라 유형을 나눌 수 있는데, 전이가 결정적으로 이루어지는 DPDA(Deterministic PDA), 전이가 비결정적으로 이루어지는 NPDA(Non-deterministic PDA), 스택의 Top을 여러 개 확인할 수 있는 확장된 PDA(Extended PDA)가 있다.

푸시다운 오토마타는 문맥 자유 언어(context-free language)를 인식할 수 있어, 주로 자연 언어 처리(Natural Language Processing) 분야에서 사용된다.