한국게임학회 논문집
제2권 제2호 2002년 11월

확률 유한오토마타의 추론을 이용한 다양한 NPC의 행동양식 생성에 관한 기법 연구

조경은, 조형제
안양대학교 디지털미디어학부, 동국대학교 멀티미디어학과
cke@aycc.anyang.ac.kr, chohj@dongguk.edu

Generating various NPCs Behavior using Inference of Stochastic Finite Automata

Kyungeun Cho, Hyungje Cho
Division of Digital Media, Anyang University
Dept. of Multimedia, Dongguk University

요 약

이 논문에서는 FSM과 확률적 FSM, NFA 등이 게임에서 NPC의 행동 지정에 쓰인 방식을 소개하고, 기존 방법에서 확률적 FSM이나 NFA의 단점을 보완할 수 있는 새로운 확률적 FSM 방식을 제안한다. 즉, 확률 유한오토마타의 추론 방식을 이용하여 다양한 NPC나 컴퓨터 플레이어의 인성이나 특성을 자동적으로 게임에 반영하기 위한 방법을 제안한다. 이 방법으로 수 많은 게이머들의 인성이나 특성을 자동적으로 파악하여, 실제 게임에서 사용되는 NPC나 컴퓨터 플레이어에게 부여해 줄 수 있고, 또한 NPC들의 인성을 다양하게 부여함으로써 게임의 재미를 더 향상시킬 수가 있다.

Abstract

This paper introduces FSM, statistical FSM and NFA that are used for assigning behaviors of NPCs in computer games. We propose a new method for remedy of the weakness of previous studies. We use the method of inferencing stochastic grammars to generate NPCs behaviors. Using this method we can generate a lot of NPCs or Computer Players behaviors automatically and the games will be more enjoyable.

Key Words : Game AI, NPCs behavior, Inference of Stochastic Finite Automata, Finite state machine(FSM), NFA(Nondeterministic Finite Automata)

1. 서론

게임의 인공지능이란, 게임 객체의 주위 환경이나 또 다른 게임 객체와의 반응을 결정하는 로직으로, 실제적인 게임의 재미를 주는 요소이다. 기본적인 게임의 인공지능 기법은 게임루프 상에서 어떤 이벤트나 유저의 입력으로 인해 게임 객체의 상태가 변경되는 것이다. 게임에서 이러한 인공지능의 역할은 NPC(Non Player Character)의 동작을 제어하거나, 게임에서 상대역을 하기도 한다. 게임에서 등장하는 인공지능의 이슈는 경로찾기(pathfinding), 지능적인 행동, 계획(planning), 속임수(cheating) 등이 있다. 그리고 이러한 이슈들을 처리하기 위한 인공지능 구현 기술로는 FSM(Finite State Machine, Finite Automata)과 퍼지 이론

본 연구는 2002년도 한국과학재단 특정기초연구(과제 번호: R01-2002-000-00298-0) 내용의 일부임

들이 쓰이고 이것을 이용하여 게임 객체의 상태변화나 반응을 구현한다. 그 외에도 결정트리, 규칙 기반 시스템 등의 인공지능 알고리즘들도 게임 인공지능에서 쓰인다[1,2,3].

FSM은 말그대로 게임 객체가 가질 수 있는 일련의 상태와 각 상태의 전이를 위해 필요한 이벤트를 정의하여 객체의 상태 변이를 구현하는 것이다. FSM을 컴퓨터 게임의 인공지능에 도입한 사례를 보면 다음과 같다. 퀘이크(Quake)는 NPC의 행동 패턴을 제어하는데 FSM을 사용하였으며, Halflife는 계층적 FSM(HFSM)을 사용하였다. 또는 NPC의 감정을 흉내내기 위한 용도로도 사용되어졌다. FSM은 기본적인 아이디어나 알고리즘은 간단하지만, 전통적으로 게임에서 가장 많이 쓰이는 기술 중의 하나이다[1,4].

본 논문에서는 FSM이나 FSM의 응용인 확률적 FSM이나 NFA 등이 게임에서 NPC의 행동 지정에 쓰인 방식을 설명하고, 기존 방법에서 확률적 FSM의 단점을 보완할 수 있는 새로운 확률적 FSM 방식을 제안한다. 이 방법을 제안함으로써 NPC들의 인성을 다양하게 부여할 수 있게 되고 게임의 재미를 더 향상시킬 수가 있을 것이며, 인성 부여를 자동화시키는 것 자체에도 그 의의가 있다고 볼 수 있다.

이 논문의 구성은 다음과 같다. 2장에서는 기존에 사용되어졌던 NPC 행동 지정 방법으로서 기본적인 FSM 방식, 확률적 FSM 그리고 NFA를 이용한 방법을 설명하며, 3장에서는 본 논문에서 제안한 확률값을 적용한 NFA방식의 기본이론이 되는 확률 유한 오토마타의 추론에 관한 내용을 설명한다. 4장에서는 제안한 방법인 확률 유한 오토마타의 추론을 이용한 다양한 NPC의 행동 양식 생성방법을 소개하도록 하며 마지막으로 결론을 맺도록 한다.

2. FSM을 이용한 NPC 행동 지정 방법

이 장에서는 가장 기본이 되는 FSM, 확률적 FSM 그리고 NFA(비결정 유한 오토마타)를 NPC의 행동지정에 적용한 기존 방법들을 소개한다. FSM은 하나의 '입력'을 받고 그에 의거해서 현재 상태로부터 다른 어떤 상태로 '전이'(transition)하는 식으로 작동한다. 즉, 어떤 상태를 현재 상태로 만들 것인지 판단하는 간단한 상태 전이 함수를 가진다. 따라서 FSM은 유한한 개수의 상태들을 가진 하나의 기계이고, 그 상태들 중 하나가 현재 상태인 것이다. FSM을 게임의 인공지능에 적용한 예를 기본 FSM, 확률적 FSM 그리고 NFA 방식의 3가지 응용 예를 들어 살펴보면 다음과 같다[4,5].

기본 FSM을 게임 인공지능에 적용한 예를 살펴보자. 임의의 전투 게임에서 플레이어가 NPC들을 파괴하거나 또는 죽이면서 돌아다니는 게임이라고 가정하자. 이때 NPC들이 끊임없는 공격에 대항해서 살아남기 위해 다양한 행동들을 취하게 된다. 객체의 상태를 구체적으로 구분하는 것이 상태 기계를 만드는 가장 첫 번째 단계이다. NPC의 행동은 공격, 후퇴, 임의로 이동, 정지라는 4 가지의 상태를 가진다고 가정하자. 다음으로는 입력을 정의해야 한다. 플레이어의 등장, 플레이어의 죽음, 플레이어의 공격, 플레이어의 후퇴, 플레이어 안보임 등으로 입력을 정의내릴 수 잇다. 표 1은 적들이 취할 수 있는 상태들과 발생할 수 있는 입력 심볼을 나타낸 것이며 이들의 관계를 상태전이도로 표시한 것이 그림 1이다.

상태	공격
	후퇴
	임의로 이동
	정지
입력	플레이어의 등장
	플레이어의 죽음
	플레이어의 공격
	플레이어의 후퇴
	플레이어 안보임

표 1. 상태 및 입력 심볼

그림 1에서 보는 바와 같이 게임상의 NPC는 임의의 상태에 있을 경우 항상 같은 방식으로 행동한다. 예를 들어, 임의로 이동하고 있다가 플레이어의 공격을 받게 되면 공격 상태로 이동하게 된다. 공격 상태였다가 플레이어가 안보이면 NPC는 후퇴한다. 예에서도 볼 수 있듯이 기본적인 FSM을 사용할 경우에는 모든 NPC들이 행동하는 형태가 일관적이므로 게임은 단순해질 수 밖에 없다.

이 단점을 보완하기 위해서 사용하게 되는 것이 확률적 FSM이다. 즉, 이 확률적 FSM을 적용함으로써 NPC들의 성격을 부여할 수가 있게 되고, 따라서 NPC의 성격에 따라 서로 다른 행동을 취할 수 있는 것처럼 게임을 진행하게 되는 것이다. 예를 들어 임의의 NPC가 매우 터프한 성격을 지닌 NPC인 경우와 매우 소극적인 성격을 지닌 NPC를 FSM으로 표현한다고 가정하자. 이때에는 몇 가지 행동 특성들을 추

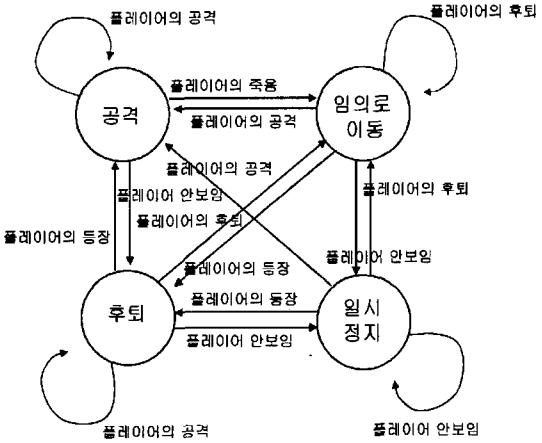


그림 1. NPC의 상태 전이도

적해서 각 특성에 확률을 정하는 논리와 확률 분포를 사용하여 성격의 유형을 모델로 설정할 수가 있어야하고, 이와 같은 확률 그래프는 상태를 변화시키는 데 사용될 수 있다. 예를 들어 다음과 같은 네 가지의 상태가 있다 : 공격, 후퇴, 정지, 임의로 이동. 그리고 임의의 NPC들이 새로운 상태를 결정할 때에는 각각의 확률 분포에 의해서 다음 상태를 결정하는 데 영향을 주게 된다. NPC1은 공격하기를 좋아하는 성격으로서 성격 확률 분포는 공격 50%, 후퇴 20%, 임의로 이동 20%, 정지 10%라고 하고, NPC2는 조금 소극적인 성격으로서 성격 확률 분포는 공격 15%, 후퇴 40%, 정지 30%, 임의로 이동이 15%이다. 그래서 이러한 성격 확률 분포에 의해서 다음 상태를 선택하게 된다. 다음 그림 2는 확률적 FSM을 사용하여 NPC1에 대한 상태선택을 하는 과정을 나타낸 그림이다. 각 상태의 일어날 확률이 각 NPC의 성격을 부여한 것이다. 상태 선택은 다음과 같이 간단하게 구현할 수 있다. 즉, state_table에는 성격을 반영하는 각 상태에 대한 비율대로 테이블에 표기된다. 예를 들어 공격상태가 50%이므로 state_table에서 10개 중에 5개가 상태 0으로 표기된다. 새로운 상태를 구할 때에는 간단하게 랜덤함수로 구할 수가 있다.

```
int state_table[10] = {0,0,0,0,0,1,1,2,2,3} ;
new_state = state_table[rand()%10];
```

이 방법은 기본적인 FSM을 사용한 방법보다는 좀 더 인공지능이 향상된 방법으로서 NPC들의 서로 다른 성격을 부여할 수는 있지만, 확률을 고려하여 상태를 랜덤하게 선택하도록 할 뿐, 실제현재 상태에서 임의의 입력에 대한 반응을 전혀 고려하지 않은 방법이다. 그 외에 확률적 FSM의 확장방법으로는 NPC가 위치하고 있는 지역 범위에 따라 다른 확률표를 사용하는 다른 응용방법도 있다 [5].

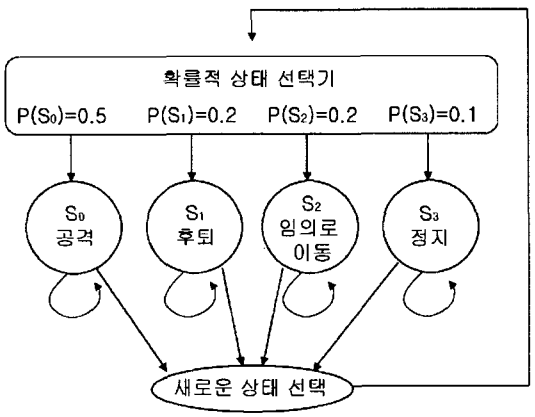


그림 2. 확률적인 상태 선택

마지막으로 소개할 FSM의 응용 방법으로는 NFA를 적용한 방법이다. FSM(FA)은 DFA(Deterministic Finite Automata)와 NFA(Nondeterministic Finite Automata)로 구분된다. 일반적으로 FA는 DFA를 의미한다. DFA는 하나의 입력 심볼을 보고 갈 수 있는 다음 상태가 반드시 한 개인 것에 반해, NFA는 하나의 입력 심볼을 보고 선택할 수 있는 다음 상태가 2개 이상이 될 수 있으므로 여러 가지의 가능성을 고려해야 하므로 비결정적이라고 한다. NFA가 게임에 적용된 예를 살펴보자. 예로써, 캐릭터가 가지고 있는 탄약에 따라 선택할 수 있는 상태가 달라지게 할 수 있다. 아주 많은 탄약, 적당한 탄약, 적은 탄약(또는 전혀 없음)의 세 가지 가능한 상황이 있을 때, 분기할 수 있는 상태는 아주 많은 탄약을 지닌 경우에는 공격상태로 상태를 전이하며, 반면에 적은 탄약 또는 전혀 없는 경우에는 탄약을 구할 수 있는 상태로 전이하게 되는 NFA가 필요하다. 앞의 예에서처럼 인위적으로 적당량을 분배하여 상황에 따라 적당한 상태를 선택할 수 있으나, 때로는 게임에서 이 인위적인 값을 좀 더 사실적인 값으로 추정해야 할 경우도 있다. 그러나 그

사실적인 값을 추정하는 것이 쉽지 않다.

확률적 FSM이 게임에 적용된 예를 NFA와 결부지어서 살펴보도록 하자. 즉, 인성을 부여할 필요성이 있는 경우가 있다고 보자. 그림 3에서 볼 수 있듯이 공격상태에서 플레이어의 공격이 들어왔을 때, 계속 공격상태를 유지할 수도 있지만, 후퇴할 수도 있는 경우가 발생한다. 따라서 게임에 따라서는 같은 상황이 발생했어도 그 NPC나 컴퓨터 플레이어의 인성에 따라 다른 상태를 선택할 가능성이 많다. 이에 NFA는 이를 반영해 줄 수 있는 좋은 FSM이다. 그러나, 이때 비결정적인 경우에 다음 상태를 결정할 때 확률을 도입해야 하는데, 앞에서 언급했던 확률적 FSM을 적용하는 방법을 그대로 적용할 수는 없다. 왜냐하면, 전이가 발생하는 확률을 임의로 결정하기가 어렵기 때문이다. 따라서, 전이하는 확률값을 추정하기 위한 방법이 필요하다. 그림 4는 확률적 요소를 NFA에 적용해 보았을 경우의 상태전이도를 나타낸다. 이 그림에서 보면 상태 q0에서 a라는 입력 심볼을 보고 갈 수 있는 상태가 2가지가 있다. 한 상태는 q1이고 다른 상태는 q3이다. 이들이 전이할 때 가지는 확률값은 서로 다르다.

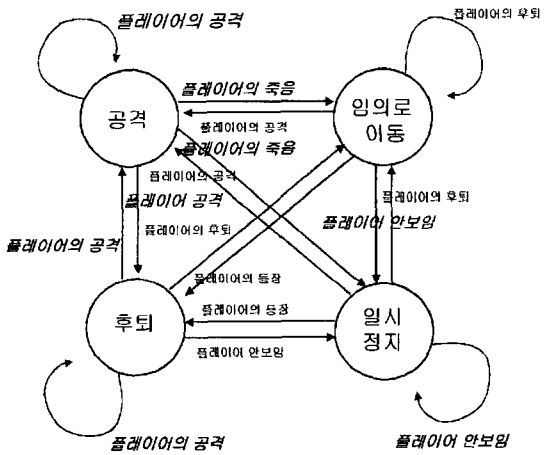


그림 3. NFA를 이용한 상태 전이도

본 논문에서는 이 확률값을 자동으로 추정하기 위한 방법을 제시하고자 한다. 즉, 다양한 플레이어들의 각 인성을 학습시키기 위해서, 확률값을 추정하기 위한 학습게임에서 실제 게이머들을 게임하게 한다. 그런 후 다양한 플레이어들의 인성을 학습시킨 후 이들의 인성 확률 테이블을 실제 게임에서 활용할 수 있는 방법을 사용한다. 그렇게 하여, 다양한 인성을 지닌 NPC나 컴퓨터 플레이어를 자동으로 생성해 내고, 더욱더 재미요소를 향상 시킬 수 있는 게임을 만들 수가 있게 된다.

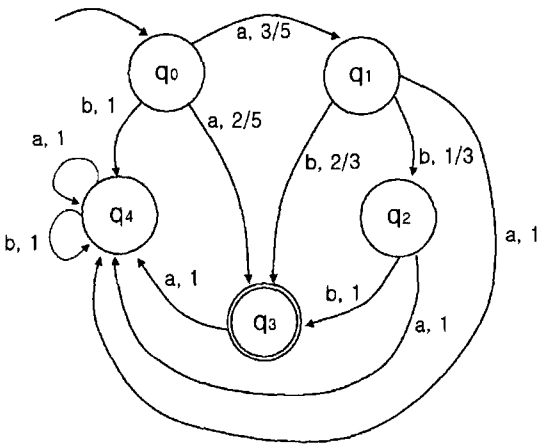


그림 4. 확률적 NFA의 상태전이도

3. 확률 유한 오토마타의 추론

본 논문에서는 기존 게임 인공지능에서 사용되었던 방법보다 좀 더 지능적으로 보이는 NPC들의 행동 유형을 생성해내기 위한 방법으로 확률적 NFA의 사용을 제안한다. 이 확률적 NFA에 NPC들의 인성을 반영하기 위해서는 각 상태와 각 상태들 사이의 전이 확률이 추정되어져야 한다. 이 장에서는 본 논문에서 제안한 확률적 NFA의 구축을 위해서 확률값을 자동적으로 추정할수 있는 방법에 사용되는 기본적인 이론을 설명한다.

확률적 문법 추론 방법 (inference of stochastic grammars)은 이미 형식 언어(Formal Language) 분야에서 많이 연구되어진 분야이다. 본 논문에서는 확률적 문법 추론 방법을 게임에서의 NPC들의 행동양식을 추출하기 위한 방법으로 적용하려고 하는 것이다. 다음은 확률적 문법 추론 방법을 기술한다.

주어진 문법과 학습 패턴들을 가지고 각 문법의 생성규칙에 해당하는 확률값을 구하는 것이 확률적 문법 추론방식

이다. 각 전이 확률을 추론하기 위한 절차로 게임에서 쓰이는 상태전이도를 문법으로 표현해야 한다. 그리고 학습 패턴이라는 것은 실제 게이머가 수행한 상태 전이를 임의의 문자열로 표현한 것이라고 말할 수 있다. 주어진 학습 패턴들은 서로 다른 문법을 생성하는 패턴들과 각각의 문법에 해당하는 확률값으로 이루어져 있다. 여기서는 확률적 문법 추론을 NPC들의 행동 분석에 이용하기 위해 각각의 생성규칙에 해당하는 확률값을 추정하는 방법을 설명한다.

일반적으로 각 생성규칙의 확률을 학습하는 방법인 확률적 문법 추론 방법 7,8 은 다음과 같다. $G_{sk} = (N_k, \Sigma_k, P_k, D_k, S_k)$에서 k = 1인 확률적 문법으로 표시되는 문제를 고려할 때 N_k는 비종단 기호의 집합, Σ_k는 종단 기호의 집합, P_k는 확률적 생성 규칙의 집합, D_k는 추정되어져야 할 각 생성규칙의 확률값의 집합 그리고 S_k는 시작 기호를 각각 의미한다. 여기서 N_k, Σ_k, P_k 그리고 S_k는 이미 알고 있는 정보라고 가정하고 G_{sk}는 모호하지 않은 문맥자유(context-free) 또는 정규 문법(regular grammar)이다. 여기서 k의 의미는 현재 수행되고 있는 게임을 의미한다. 만약 게임의 스테이지가 바뀐다면 k의 값을 여러 개로 확장할 수도 있다. 이 가정들을 바탕으로 학습 패턴 집합 $X = x_1, x_2, \ldots, x_m$ 에서 $D_k(k = 1, 2, \ldots, M)$의 생성규칙들의 확률을 추정하는 것이 요구된다. 학습 패턴 집합 X의 각 패턴들은 종단 기호들의 조합으로 이루어진 것으로 미리 정해진 확률적 문법들을 만족해야 한다. 이 학습 패턴들은 게이머가 상태들을 선택한 순서를 의미한다. 각 생성규칙의 확률값의 집합인 D_k를 추정하기 위해서는 G_{sk} 문법에서 각 생성 규칙들이 모든 학습 패턴 집합 X의 생성에 이용되었을 확률(P_{kij})들을 구해야 한다. G_{sk} 문법에서 Ai j이 모든 X의 생성에 이용되었을 확률을 Pkij라 하고 다음 수식과 같이 구할 수 있다.

$$\text{estimated } P_{kij} = \hat{P}_{kij} = \frac{n_{kij}}{\sum_r n_{kir}} \qquad (1)$$

수식(1)의 n_{kij}는 모든 학습 패턴에 대해서 G_{sk}문법에서 하나의 생성규칙인 $A_i - \beta$가 사용되어진 총 평균 횟수를 의미하며 다음 수식에 의해서 구하게 된다.

$$n_{kij} = \sum_{x_h \text{ in } X} n(x_h) \cdot p(G_{sk}/x_h) \cdot N_{kij}(x_h) \qquad (2)$$

수식(2)에서 각 항목이 의미하는 바는 다음과 같다. $n(x_h)$는 학습 패턴 집합의 모든 패턴에 대해서 이들이 학습 패턴 집합에서 발생한 빈도수를 의미한다. $N_{kij}(x_h)$는 하나의 패턴 x_h를 파싱할때 G_{sk} 문법의 하나의 생성규칙인 $A_i - \beta$가 사용되어진 횟수를 나타낸다. 비록 문법들의 각 생성규칙들에 해당하는 확률은 알고 있지 않지만, 사용하는 생성규칙들은 이미 알고 있기 때문에 이 파싱 단계가 가능한 것이다. $p(G_{sk}/x_h)$는 하나의 패턴 x_h가 G_{sk} 문법으로부터 생성되었을 확률을 의미한다. 수식(1)의 $\sum_r n_{kir}$ 는 $A_i - \beta_r$의 경우에 대한 총합을 의미하며 G_{sk} 문법에서 같은 left part nonterminal A_i를 가지는 모든 생성규칙에 대하여 위의 방법으로 n_{kir}을 계산하여 총합을 구한다.

모든 문법의 생성 규칙에 해당하는 확률을 표현하는 집합 D_k가 구해지면 확률적 문법의 추론 단계인 학습 단계가 완료되고 이 확률테이블들은 추후에 게임에서 각 NPC나 컴퓨터 플레이어의 인성으로 활용되는 값이 되는 것이다.

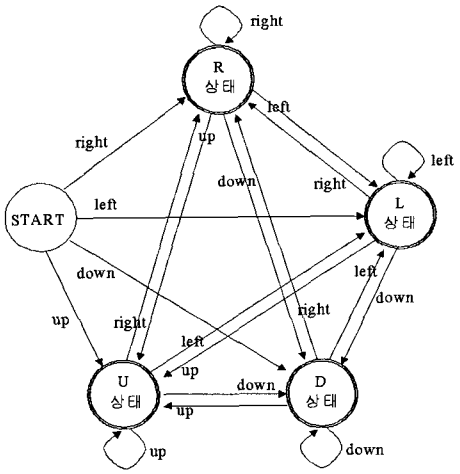
4. 확률 유한 오토마타의 추론을 이용한 다양한 NPC의 행동방식 생성

이 논문에서는 확률적 정규 문법 추론으로 게이머들의 상태 전이 방식을 분석함으로써 게이머들의 특성을 추출하려고 한다.

여기서는 확률 유한 오토마타의 추론이 진행되는 단계를 설명하기 위한 간단한 실험으로 한 게이머가 2차원 게임에서 상하좌우로 상태를 전이한 것을 학습하기 위한 것을 예로 들어 설명한다. 추후 실제 게임에서는 이 상태들의 수를 확장하고 입력 심볼들을 필요한 만큼 정의한 후에 적용하면 된다.

여기서 정규문법을 사용하는 이유는 게이머들의 상태 전이를 정규문법만으로도 충분히 표현할 수가 있기 때문이다.

게이머들의 게임 패턴들을 파싱해주는 유한오토마타의 상태전이도(그림 5(a))와 이로부터 파생된 정규문법(그림 5(b))은 다음과 같다.



(a) 유한오토마타 M의 상태 전이도

G ={S, R, L, U, D}, {right, left, up, down}, P, S) with productions

S -> right R	R -> right R	L -> right R
S -> left L	R -> left L	L -> left L
S -> up U	R -> up U	L -> up U
S -> down D	R -> down D	L -> down D
	R -> right	L -> right
	R -> left	L -> left
	R -> up	L -> up
	R -> down	L -> down
	U -> right R	D -> right R
	U -> left L	D -> left L
	U -> up U	D -> up U
	U -> down D	D -> down D
	U -> right	D -> right
	U -> left	D -> left
	U -> up	D -> up
	U -> down	D -> down

(b) 파생된 정규문법 G

그림 5. 유한오토마타 M의 상태 전이도와 파생된 정규문법 G

3장에서 설명한 확률적 문법 추론 방법을 이용하여 게이머의 학습 패턴 집합의 모든 데이터에 대해서 앞에서 정의한 정규문법의 모든 생성규칙에 해당하는 확률값을 추정한다. 만약 게임 스테이지가 여러 개라면 m은 스테이지 수만큼 늘어날 수도 있다.

$G_{s1} = (N, \Sigma, P, D_1, S)$

$G_{s2} = (N, \Sigma, P, D_2, S)$

$G_{sm} = (N, \Sigma, P, D_m, S)$

N = {S, R, L, U, D}

Σ = {right, left, up, down}

P = 유한오토마타 M에 의해서 파생된 정규문법의 생성규칙들

D_k~추정되어져야 할 각 생성규칙의 확률값

S = 시작 상태

	생성규칙	대치된 생성규칙	Stage 1		Stage m	[대치 규칙]
1	S -> right R	A1->β1	0.12217		P_{m11}	A1 = S
2	S -> left L	A1->β2	0.56045		P_{m12}	A2 = R
3	S -> up U	A1->β3	0.30227		P_{m13}	A3 = L
4	S -> down D	A1->β4	0.01511		P_{m14}	A4 = U
5	R -> right R	A2->β1	0.69052		P_{m21}	A5 = D
6	R -> left L	A2->β2	0.03151		P_{m22}	
7	R -> up U	A2->β3	0.16288		P_{m23}	β1 = right R
8	R -> down D	A2->β4	0.10734	...	P_{m24}	β2 = left L
9	R -> right	A2->β5	0.00694		P_{m25}	β3 = up U
	. . .	. . .	. . .		. . .	β4 = down D
						β5 = right
						β6 = left
23	U -> up U	A4->β3	0.86688		P_{m43}	β7 = up
24	U -> down D	A4->β4	0.02035		P_{m44}	β8 = down
	. . .	. . .	. . .		. . .	
36	D -> down	A5->β8	0.05257		P_{m58}	

표 2. 각 생성규칙에 해당하는 확률표

실질적으로 각 생성규칙에 해당하는 확률은 다음 표 2에서와 같이 구한다. 생성 규칙들은 확률을 표기하는 방법 P_{kij}을 따르기 위해서 정의한 대치규칙에 의해서 대치된다. G_{sk} 문법에서 각 생성 규칙들이 모든 학습 패턴 집합 X의 생성에 이용되었을 확률을 P_{kij}로 표기한다. 여기서 첨자 k는 소속 문법 클래스, 첨자 i는 생성규칙의 좌측항(left-hand side)의 첨자 그리고 첨자 j는 생성규칙의 우측항(right-hand side)의 첨자를 나타낸다. 각 생성규칙에 해당하는 확률값을 구하는 방법은 3 장에서 설명한 방법과 같다. 실제적으로 추정되어진 확률값에 대한 예를 표 2의 스테이지 1열에서 보여준다. 실질적으로 이 확률값들이 게임에서의 각 NPC 또는 컴퓨터 플레이어의 특성으로 활용되는 값들이다.

5. 결론 및 향후 연구

이 논문에서는 FSM이나 FSM의 응용인 확률적 FSM이나 NFA 등이 게임에서 NPC의 행동 지정에 쓰인 방식을 설명하였고, 기존 방법에서 확률적 FSM의 단점을 보완할 수 있는 새로운 확률적 FSM 방식을 제안했다. 즉, 확률 유한오토마타의 추론 방식을 이용하여 다양한 NPC나 컴퓨터 플레이어의 인성이나 특성을 자동적으로 게임에 반영하기 위한 방법을 제시한다. 이 방법을 사용하여 수많은 게이머들의 인성이나 특성을 자동적으로 파악하여, 실제 게임에서 사용되는 NPC나 컴퓨터 플레이어에게 부여해 줄 수 있다. 또한 NPC들의 인성을 다양하게 부여함으로써 게임의 재미를 더 향상시킬 수가 있을 것이며, 인성 및 특성 추출을 자동화시키는 것 자체에 이 연구의 큰 의미를 둔다.

현재까지는 게이머의 인성 및 특성 추출 알고리즘을 간단한 시뮬레이션만으로 실험하여 이 알고리즘의 타당성을 확인하였다. 향후 연구과제로는 다양한 상태 전이를 지니고, 다양한 입력들이 주어진 규모가 큰 게임에서의 실제 상황을 구현하여, 게이머들의 인성 및 특성 추출 자동 알고리즘을 적용한 후 이들의 결과를 여러 NPC나 컴퓨터 플레이어에게 부여하여 본 연구에서 제안한 방법을 실제 게임에 적용해보는 것이다.

참고문헌

[1] 동국대학교 산학기술협력센터, "기술 수준을 중심으로 한 게임 산업의 업계 실태 조사 연구", 2002

[2] 이만재, "게임에서의 인공지능 기술", 정보처리학회지 제 9권 제 3호, 2002

[3] S. Rabin, "AI Game Programming Wisdom", Charles River Media, Inc., 2002

[4] M. Deloura, "Game Programming Gems 1", Charles River Media, 1999

[5] A. Lamothe, "Tricks of the Windows Game Programming Gurus", Macmillian USA, 2000

[6] Lamothe 저, 김숙자 역, "3D 게임 프로그래밍", 성안당, 1999

[7] K.S. Fu, "Syntactic Methods in Pattern Recognition," Academic Press, 1974

[8] R.C. Gonzalez and M.G. Thomason, "Syntactic Pattern Recognition," Addison-Wesley Publishing Company, 1978

조 경 은

1989.3 ~ 1993.2 동국대학교, 전자계산학과(공학사)
1993.3 ~ 1995.2 동국대학교, 컴퓨터공학과 대학원(공학석사)
1995.3 ~ 2001.8 동국대학교, 컴퓨터공학과 대학원(공학박사)
2001.8 ~ 2002.2 동국대학교, 산업기술연구소 연구원
2002.3 ~ 현재 안양대학교, 디지털미디어학부 강의전담교수
관심분야 : 컴퓨터 게임 알고리즘, 게임 인공지능, 컴퓨터 그래픽스, 멀티미디어 정보처리

조 형 제

1969.3 ~ 1973.2 부산대학교, 전자공학과(공학사)
1973.9 ~ 1975.8 한국과학기술원, 전기&전자공학과 대학원(공학석사)
1982.9 ~ 1986.2 한국과학기술원, 전기&전자공학과 대학원(공학박사)
1975.8 ~ 1982.8 금성통신(주)연구소 실장
1986. 2 ~ 현재 동국대학교 멀티미디어학과 교수
관심분야 : 게임공학, 멀티미디어 정보처리, 컴퓨터비젼, 컴퓨터 그래픽스