임베디드 시스템에서의 상황인식 제어를 위한 상태전이 기반 상황 모델링과 이를 응용한 상황-동작 변환기(SAC)의 설계

(A state transition based situation modeling and its application to design of SAC(Situation-Action Converter) for situation-aware control for embedded systems)

허 길 † 박 희 정 †† 조 위 덕 ††† 최 재 영 ††††
(Gil Heo) (Joshua Park) (We-duke Cho) (Jae-Young Choi)

요 약 지능형 서비스를 제공하는 환경에서 상황을 인식하기 위하여 임베디드 시스템이 갖는 제한적인 리소스와 컴퓨팅 파워 환경에 적합한 상태 전이 기반 상황 표현 모델을 제안하고, 이를 인식하는 상황 인식기와 제어 신호를 발생시키는 동작 발생기를 결합한 임베디드 시스템에 탑재하기에 적합한 상황-동작 변환기(SAC)를 설계하였다. 또한 ARM 프로세서 기반의 임베디드 보드에 설계된 상황-동작 변환기를 응용한 상황 관리기를 구현하여 이를 스마트 스케줄러 서비스에 활용하였다.

키워드 : 임베디드 시스템, 상황인식, 유한오토마타, 패턴인식, 홈네트워크, 가전제어

Abstract In order to recognize a situation from a environment which provides an intelligent service, we propose state-transition based situation modeling which is suitable for a low computing power and restricted resources like embedded systems, and we designed its application to a situation-action converter(SAC)which is consist of two parts; situation detector recognized wanted situations and action generator generated various control actions. Then, we implemented a situation manager for smart scheduler service by using a SAC which is installed to a ARM processor based embedded Linux evaluation board.

Key words : embedded system, context-awareness, finite automata, pattern recognition, home network, digital appliance control

1. 서 론

지능형 서비스를 제공하기 위한 매우 좋은 방법은 서비스 환경에 있는 임베디드 시스템이 상황을 인식하여 그에 맞는 장비들을 적절하게 제어를 하는 것이다. 그러기 위해서는 현재의 상황을 가능한 정확하게 인식하는 것이 무엇보다도 중요한데, 특히 컴퓨터의 성능이 향상되면서 과거에 느린 추론 속도로 인하여 실 시스템에 적용되기 힘들었던 퍼스트 오더 로직(first-order logic)에 기반을 둔 추론 기법이나 신경망을 활용한 기법, 규칙을 활용한 방법 등 인간의 생각에 가까운 지식 표현 기법에 기반을 둔 다양한 형태의 추론 시스템에 대한 연구가 그 어느 때보다도 활발히 진행되고 있다. CMU Aura의 CIS(Contextual Information Service)[1], UIUC Gaia의 Active Space[2], Stanford University의 ICrafter[3], MIT의 Oxygen[4], ASU의 RCSM (Reconfigurable Context-Sensitive Middleware for Pervasive Computing)[5], UMBC의 CoBrA(Context Broker Architecture)[6] 등은 그 좋은 예이다.

하지만, 추론에 기반한 상황 인식 기법은 컴퓨터 성능이 우수해 졌다고는 하나 여전히 빠른 컴퓨팅 속도와

· 본 연구는 21세기 프론티어 연구개발사업의 일환으로 추진되고 있는 정보통신부의 유비쿼터스컴퓨팅및네트워크원천기반기술개발사업의 지원에 의한 것임
† 학생회원 : 아주대학교 유비쿼터스시스템 연구센터 연구원
semylab@ajou.ac.kr
†† 정 회 원 : 아주대학교 유비쿼터스시스템 연구센터 연구원
joshua@ajou.ac.kr
††† 종신회원 : 아주대학교 전자공학부 교수
chowd@ajou.ac.kr
†††† 종신회원 : 숭실대학교 컴퓨터학부 교수
choi@ssu.ac.kr
논문접수 : 2006년 5월 24일
심사완료 : 2006년 8월 17일

많은 양의 리소스를 요구하며, 알고리즘 복잡도가 다항식 시간(polynomial time)내에 해결할 수 없기 때문에 추론에 필요한 지식(knowledge)과 사실(fact)의 수가 증가됨에 따라 기하급수적으로 계산량이 많아져서 실시간성을 보장하기 힘들고 많은 컴퓨팅 파워와 많은 양의 리소스를 요구하기 때문에 임베디드 시스템에 적용하기 힘들다[7,8]. 또한 논리연산을 활용하여 상황 판단을 사용하는 경우는 비교적 구조가 간단하고 추론 방법에 비하여 빠르기 때문에 홈네트워크와 같은 상용 시스템의 조건부 작동 설정 등에 활용되고 있다[14,15]. 그러나 표현상의 제약이 많고 특히 시간의 흐름에 따른 상황을 표현하기가 힘들다.

그러므로, 임베디드 시스템에서는 센싱 패턴 상황 인식 기법이 보다 적합한데, 이 방법은 시간의 흐름에 따른 센싱 값들의 변화를 비교하면서 모든 조건이 만족되면 원하는 상황을 찾아내는 방법으로, 상황을 인식하는 방법이 간단하기 때문에 작게 제작이 가능하고 정의된 상황을 정확하게 인식이 가능하며 적은 하드웨어 리소스와 작은 컴퓨팅 파워를 요구하기 때문에 리얼타임이 요구되는 시스템에 적합하다. 그러나 이러한 패턴 방식의 상황 인식 방법은 간단한 구조로 인하여 다양한 상황을 표현하는데 한계를 가지며, 정적으로 묘사되는 패턴 특성 때문에 미리 정해진 상황 만을 인식하기 위한 시스템으로 개발되는 것이 보통이다. 따라서, 센싱 패턴을 이용하면서도 추론기법에서와 같이 상황을 잘 표현할 수 있다면 센싱 패턴 인식기법이 갖는 빠른 실시간성 처리와 적은 리소스 사용의 장점을 가지면서도 다양한 형태의 상황에 대응하는 지능형 임베디드 시스템을 개발하는데 도움이 될 것이다.

본 논문에서는 인식해야 할 상황을 표현하기 위하여 센싱 패턴 상황 표현 방법을 개선한 상태 전이(state transition) 형식의 상황 모델 표현 방법을 제안한다. 이 표현 방법은 기존의 패턴 형식의 고정식 표현 방법보다 훨씬 단순하면서도 보다 다양한 상황 표현이 가능하다. 또한 본 논문의 방법으로 모델링 된 상황을 인식하는 인식기(automata)도 고정식 표현 인식기가 갖는 장점인 간단한 구조와 적은 리소스를 사용하기 때문에 내장형 시스템에 적용하기가 용이하며, O(n)의 시간 복잡도를 갖기 때문에 매우 빠른 응답시간을 보일 수 있어 실시간 처리 응용에 적합하다. 본 논문에서는 제안한 모델링을 이용한 상황인식기반 제어기인 SAC(Situation-Action Converter)를 설계하고 이를 활용한 ARM 프로세서 기반 임베디드 시스템을 구현하여 실제 가전제품이 제어되도록 하였다. 2장에서는 상태 전이를 이용한 상황 모델링에 대하여 설명하고, 3장에서는 상황 인식기와 동작 발생기에 대하여 설명한다. 4장에서는 본 논문에서 제시한 내용을 실제로 적용한 테스트 내용을 소개하며, 마지막으로 5장에서는 결론과 향후 연구방향에 대하여 설명한다.

2. 상태 전이(state transition) 상황 모델링

지능화 수준이 높아지고 그에 따른 훨씬 많은 다양한 상황을 인식해야 제공할 수 있는 서비스를 제공해야 하는 임베디드 시스템에 기존의 센싱 패턴기반의 인식 방법을 적용하기 위해서는 보다 다양한 고 수준의 상황 표현이 가능해야 하면서도 개발하기 쉬워야 한다. 이를 해결하기 위하여 본 논문에서는 상황을 인식하기 위한 센싱 패턴을 정규 표현(regular expression)으로 나타낼 수 있으면 이를 상태 전이도(state transition diagram)로 모델링하여 이를 인식하는 유한 오토마타(finite automata)를 설계할 수 있다.

2.1 센싱 패턴 기반 상황 표현

센싱 패턴을 이용한 상황 인식 방법은 상황 판단의 근거가 되는 각종 센서들의 값이 시간이 흐름에 따라 변화되는 것을 반영하여 값 패턴의 유사도를 비교하여 일치 허용 범위 안에 들어오면 상황이 발생되었다고 판단하여 그에 따른 적절한 처리를 하는 방식이다. 이와 같은 접근 방법은 필요에 따라 손쉽게 필요한 패턴을 만들어 사용할 수 있는데, 그것이 가능한 이유는 일반적으로 인식해야 할 상황이 복잡하지 않고 센싱하는 정보의 종류가 1~2가지 정도로 많지 않기 때문이다. 다음의 그림 1의 예는 실제 사람이 방에 들어오는 과정에 대한 센싱 값의 변화를 나타낸 것이다.

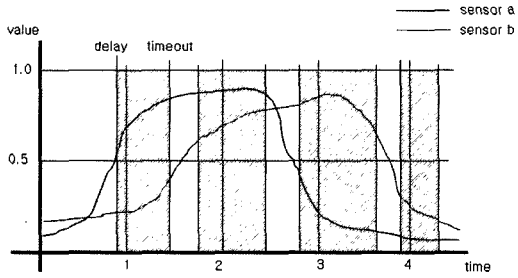

그림 1 시간에 따른 센싱 정보의 흐름 예

위의 그림 1에서 보이는 것과 같은 실제 센서 값을 시스템에서 인식하기 위해서는 우선 센싱된 값을 적당한 시간(예를 들어 1초)단위로 샘플링하고 그 샘플링 된 값들을 이용하여 읽어 들인 값이 오차가 허용되는 원하는 범위 안에 있으면 하나씩 그 다음 조건을 만족하는지 조사하는 형식으로 인식될 수 있다. 이와 같은 방법은 구조가 간단하고 패턴 설계가 쉬우며 지금까지 대부분의 상황을 인식하는 시스템에 적용되어 왔던 예에서

와 같이 인식에 대한 고 신뢰성이 보장된다. 하지만 패턴의 변화에 대처하기 쉽지 않으며, 패턴에 포함되는 센서의 수가 많아지면 복잡도가 기하급수적으로 증가하기 때문에 현실적으로 3~4개 이상의 센서 정보를 이용해야 하는 시스템에 적용하기가 쉽지 않다. 또한 설정된 패턴을 상황 변화의 감지 등에 의해서 프로그램으로 동적으로 변경하기가 쉽지 않아 상황 적응형 지능형 서비스 시스템에 적용하기가 쉽지 않다.

2.2 정규 표현을 이용한 센싱 패턴 표현

센싱 패턴 형식의 상황 표현 방법은 표준화된 표현 방법이 없기 때문에 시스템간의 패턴 호환이나 커뮤니티 등을 통한 상황별 유용한 패턴 개발 및 공유 등이 현실적으로 불가능하다. 따라서, 패턴의 표준화된 표기법이 필요한데 정규 표현은 패턴을 표현하는데 가장 적합한 표현법 중의 하나이다. 위의 그림 1로부터 입력 심볼을 추출하면 다음의 표 1과 같이 나타낼 수 있다.

표 1 추출한 Input Symbol들

{0.0 <= a < 0.5} ..... Input Symbol ① (I_1)
{0.5 <= a < 1.0} ..... Input Symbol ② (I_2)
{0.0 <= b < 0.5} ..... Input Symbol ③ (I_3)
{0.5 <= b < 1.0} ..... Input Symbol ④ (I_4)

위의 표 1에서 추출한 입력 심볼 들을 중심으로 예외 상황이 고려되지 않은 처음 기술했던 상황을 다시 표현하면 다음의 표 2와 같이 나타낼 수 있다.

표 2 정규 표현을 이용한 상황 표현 (1차)

{방에 들어옴} = (I_1 I_3) (I_2 I_3) (I_2 I_4) (I_1 I_4) (I_1 I_3)

센서a와 센서b 사이에 읽혀지는 순서가 정해진 것이 아니므로 먼저 읽혀지는 센서에 따른 상태변화까지 고려하면 다음의 표 3과 같이 표현할 수 있다.

표 3 정규 표현을 이용한 상황 표현 (2차)

{방에 들어옴} = ((I_1 I_3) | (I_3 I_1)) ((I_2 I_3) | (I_3 I_2)) ((I_2 I_4) | (I_4 I_2)) ((I_1 I_4) | (I_4 I_1)) ((I_1 I_3) | (I_3 I_1))

일반적인 정규 표현을 이용하여 상황을 모델링 할 경우 그 복잡도가 기하급수적으로 증가되는 이유는 입력 심볼의 순서에 의한 복잡도가 가장 큰데, 논리적으로 하나의 상태에서 다음 상태로 전이(transition)되기 위하여 요구되는 입력 심볼은 한 개이지만 실제로 상태 전이가 일어나기 위해서는 여러 개의 센서들의 값들이 조합되어 결정되므로 센서값 들의 순서에 따라 계승(factorial) 수를 따르기 때문에 알고리즘의 복잡도는 센서 수의 O(n!)의 복잡도를 갖는다. 따라서 이를 해결하기 위해서는 입력 심볼 들을 시간의 오차가 허용되는 범위 내에서 그룹화 하여 그룹 범위를 만족하는 경우는 순서가 없다고 가정하면 센서 수에 따른 복잡도가 O(n!)에서 O(n)으로 줄어들 수 있다. 시간 그룹화를 근거로 한 그림 1의 상황 정의를 다시 모델링 하면 다음의 표 4와 같이 나타낼 수 있다.

표 4 시간 그룹핑이 반영된 상황 모델링

{방에 들어옴} = I_1 + I_2 + I_3 + I_4+ I_1 Input Symbol : I_1 : {0, 5000}, {0.0 <= a < 0.5}, {0.0 <= b < 0.5} I_2 : {0, 5000}, {0.5 <= a < 1.0}, {0.0 <= b < 0.5} I_3 : {0, 5000}, {0.5 <= a < 1.0}, {0.5 <= b < 1.0} I_4 : {0, 5000}, {0.0 <= a < 0.5}, {0.5 <= b < 1.0} 단, 1) + 기호는 1번 이상 반복을 의미함 2) 각 Input Symbol의 첫번째 항목은 시간적 허용 범위를 나타내는 요소로써, state가 시작된 시점으로부터 {delay, timeout}을 나타내며 시간이 delay <= t < timeout 범위 내에 있을 때 이후에 주어진 조건들이 모두 만족되면 Input Symbol이 발생된 것으로 인정함

위의 표 4에서와 같이 입력 심볼에 대한 추상화를 가함으로써 상황 표현이 매우 간단해지며 시간 개념이 포함된 입력 심볼의 표현으로 입력 심볼에 대한 복잡도가 현저하게 감소될 수 있으며, 규칙 자체의 변화나 입력되는 값의 변화가 분리되어 있어 상황 대응적 시스템에 적용하기 유리하며, 표준화된 정규 표현을 그대로 따르므로 상황을 모델링하기 쉽고, 표준화가 가능하기 때문에 다양한 상황 패턴들에 대한 개발 및 공유가 가능하다.

3. 상황-동작 변환기(SAC)의 설계

상황이 상태 전이도로 모델링 되면 이를 인식하는 인식기를 설계해야 한다. 기본적으로 유한 오토마타 모델을 그대로 따르기 때문에 구조적으로는 기존의 유한 오토마타와 동일하지만, 하나의 입력 심볼이 실제로는 시간 단위로 그룹핑 된 여러 입력들의 집합이라는 점이나 입력 심볼의 존재 유무로써 상태전이가 일어나는 것이 아니라 아닌 값의 범위로 상태전이가 일어난다는 점 등이 다르다. 따라서, 기존의 유한 오토마타 구조를 그대로 따를 수 없기 때문에 제안된 모델을 인식할 수 있도록 적절히 변형된 새로운 결정적 유한 오토마타를 설계하였으며, 제어 신호가 발생될 수 있도록 동작 발생기를 설계하였다. 그리고 상황 인식기에서 인식된 상황을 바탕으로 제어 신호가 발생될 수 있도록 상황 인식기와

동작 발생기를 결합하였는데, 이처럼 함께 동작하는 상황 인식기와 동작 발생기를 본 논문에서는 *상황-동작 변환기(SAC:Situation Action Converter)*라는 명칭을 사용하였다.

3.1 상황인식을 위한 결정적 유한 오토마타

인식하고자 하는 상황이 상태 전이도로 모델링 되었다면 결정적 유한 오토마타(DFA: deterministic finite automata)로 인식될 수 있다[8]. 본 논문에서 제안하는 다음의 표 5에 나타낸 결정적 유한 오토마타 M은 모델링된 인식하고자 하는 상황을 인식하는 오토마타이다.

표 5 상태 전이로 표현된 상황 모델을 인식하는 결정적 유한 오토마타

DFA M = (S, ∑, δ, s_0, F)
where,
S : a set of states
∑ : a set of input symbols
δ : transition function
s_0 : start state
F : a set of final states

이 가운데 Transition Function δ 는 다음의 표 6과 같이 구성된다.

표 6 전이 함수 조건

$\delta(s_i, I_j) = s_n$
where,
s_i = current state
I_j = input symbol
s_n = next state

여기서 input symbol I_j 는 입력되는 심볼들을 유효성 함수인 σ에 적용하여 조건이 만족되었을 때 얻을 수 있다. 이를 수식으로 나타내면 다음의 표 7과 같다.

표 7 입력 심볼 유효성 함수

$I_j = \sigma(condition_j, symbol_j)$
where,
$condition_j$ = <id_j, min_val_j, max_val_j, $delay_j$, $timeout_j$, $start\_time_j$>
$symbol_j$ = set of <id_k, $value_k$, $timestamp_k$>
$\sigma(condition_j, symbol_j)$ is valid for all k,
(($id_j = id_k$) AND
($min\_val_j <= value_k <= max\_val_j$) AND
($start\_time_j + delay_j <= timestamp_k <$
($start\_time_j + timeout_j$))

심볼 식별자(id_k)는 실제로 *Plugin_ID*와 *Symbol_ID*로 구성된다. *Plugin_ID*는 심볼을 발생시킨 주체가 누구인지를 구분하는 식별자이며, *Symbol_ID*는 실제 값($value_k$)의 주체이다. 입력 심볼은 기존의 유한 오토마타에서와 같이 심볼의 존재여부로 결정되는 것이 아니라 심볼($symbol_j$) 내의 모든 심볼 식별자(id_k)의 값($value_k$)이 조건($condition_j$)을 만족하는지에 대한 것으로 결정되기 때문에, 인식기에서 필요로 하는 전체 입력 심볼의 개수는 읽어 들이는 센싱 데이타의 개수에 관계 없이 무한개가 될 수 있다. 따라서 기존의 유한 오토마타를 정의하는 것과 같이 입력되는 유한개의 입력 심볼들을 미리 정의하기가 쉽지 않을 뿐만 아니라 인식기를 설계하는 측면에서도 많은 개수의 입력 심볼에 대한 상태전이 테이블을 만들어 사용하는 것이 효율적이지 않기 때문에 각각의 스테이트마다 상태전이에 필요한 입력 심볼들과 함수들을 정의하는 것이 상황 모델링과 이를 인식하는 오토마타 설계에 유리하다.

또한, 편집 툴을 이용하여 상태전이 기반 상황 모델을 설계하여 저장하고 이를 상황 인식기에서 읽어 들이기 위하여 상황 기술 언어인 *uSDML(u-Situation Description Markup Language)*을 정의하였는데, 이는 상황 인식기를 상황 인식 규칙을 이해하는 범용 상태전이 엔진 형태로 설계했기 때문이다. *uSDML*은 XML 기반의 DTD로 설계된 언어로써 위의 유한 오토마타를 그대로 표현할 수 있으며, Apache Jakarta Project의 Digester XML 파서를 사용하여 PC 환경에서 동작하도록 구현하였다[19]. 그러나 임베디드 시스템 환경에서는 인식해야 할 상황이 그다지 많지 않고 Digester XML 파서가 크고 매우 느리기 때문에 *uSDML*로 정의된 상황인식 규칙과 XML 파서를 사용하지 않고 유한 오토마타 구조체를 직접 프로그램 코딩으로 생성하여 사용하였다.

다음의 그림 2는 상황 인식기의 구조와 동작원리를 나타내고 있다.

상황 표현을 *uSDML*로 기술한 경우에는 *uSDML* 파서를 거쳐 유한 오토마타의 구성 요소인 상태(state), 전이함수(transition function), 초기상태(init state), 마지막 상태들(final states)을 추출하여 상태 전이 엔진에 등록함으로써 상황 인식기는 상황을 인식할 준비가 완료된다. 이 때, *uSDML* 파서의 출력은 원하는 상황을 인식하기 위한 유한 오토마타 구조체이므로 이 구조체를 프로그램으로 직접 생성하면 *uSDML*과 파서 없이 직접 상황인식 규칙을 등록할 수 있다. 그리고, 입력된 센싱 정보나 장비의 상태정보, 그 외의 상황 인식에 필요한 여러 가지 정보들은 *CODEC* 내의 해당 인코더를 거쳐 정규화(normalize)된 입력심볼 형태로 인식기에 공

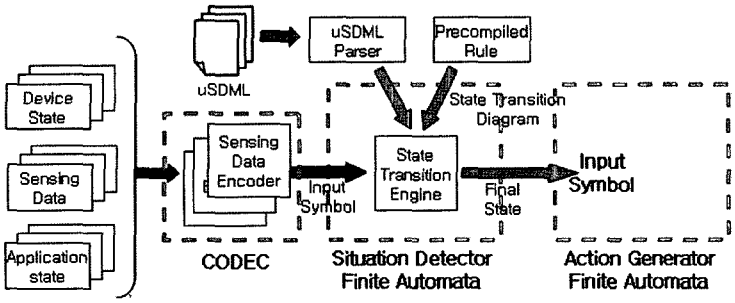

그림 2 상황 인식기의 동작원리

급된다. 상태 전이 엔진은 입력되는 심볼 들을 보면서 등록되어 있는 상태전이 함수들과 비교하여 상태전이 조건을 만족하면 상태전이가 지속적으로 일어난다. 만일 전이된 상태가 마지막상태(final state)중의 하나이면 상황이 인식되었음을 이벤트 형태로 외부에 알리고 다시 초기상태(init state)로 이동함으로써 다시 새로운 상황을 인식할 준비를 한다. 인식된 상황은 동작 발생기의 입력 심볼로 사용됨으로써 발생된 상황에 대한 동작이 수행되게 된다.

3.2 입력 심볼의 정규화

유한 오토마타를 인식기로 활용하는 예는 많이 있는데 어떤 유한개의 상태를 미리 알 수 있고 어떤 입력에 대하여 상태가 변화되는 것으로 표현할 수 있는 분야에 적용될 수 있는 좋은 인식 기법이다. 유한 오토마타는 고전적인 활용 분야인 문자열 인식이 활용되는 컴파일러나 텍스트 에디터를 비롯하여 응용 분야인 컴퓨터 키보드의 한글 입력기, 음성인식분야[9], 제스쳐인식분야[10], 실시간 시스템의 분석이나 검증[11,12]에 활용된다.

그런데, 이종의 입력을 갖는 경우에는 인식기에서 동일한 형태의 입력 심볼로 취급하기가 쉽지 않기 때문에 입력들을 추상화하여 컨텍스트화 하고 이들 컨텍스트의 형식을 일치시킴으로써 입력 심볼로 사용할 수 있다. 그러나 추상화한 컨텍스트의 레벨과 처리하기 위한 절차에는 상호 장단점이 있는데, 추상화 레벨이 높을 수록 표현 수준이 높아질 수 있지만 그만큼 처리하기 위해서는 많은 컴퓨팅 리소스를 요구하게 되고, 추상화 레벨이 낮을 수록 임베디드 시스템에 적합하지만 입력 심볼로 수용하는 데는 한계가 있다[17,20]. 따라서 높은 컨텍스트 수준을 유지하면서도 추상화 레벨이 낮은 표현이 가능하다면 임베디드 시스템에 적용 가능하면서도 다양한 입력 형식들을 수용하는 인식기가 될 수 있을 것이다.

본 논문에서는 다양한 형식의 입력 종류를 수용하면서도 임베디드 시스템에 적합한 입력 심볼로 가공하기 위하여 *CODEC*을 사용하는 방법을 적용하였다. 따라서 센싱 값, 장비 상태 등과 같은 모든 종류의 입력들은 그 입력 타입을 처리하는 *CODEC* 과정을 통해서 정규화된 저수준의 유한 오토마타의 입력 심볼로 가공된다. 본 논문에서 사용한 입력 심볼은 스트링 형태의 *CODEC* 식별자와 심볼 식별자, 그리고 16bit의 비부호 정수(unsigned integer)를 사용한 정규화된 형식을 따르고 있다. *CODEC* 변환 규칙은 실제 형식과 정규화된 형식 간의 변환만 가능하다면 제약은 두지 않았다. 본 논문에서는 실제 형식이 boolean인 경우는 0과 1로, 숫자인 경우 0~65535(=2^{15})의 값으로 변환하였으며, 문자열인 경우는 문자열 테이블을 만들고 index를 사용하는 방법으로 변환하였다.

3.3 동작 발생기(Action Generator)

지능형 서비스를 제공하기 위하여 인식된 상황에 맞는 적절한 제어를 하기 위해서는 이를 적절하게 나타낼 수 있어야 한다. 가장 손쉬운 방법은 테이블을 이용하여 상황이 발생되면 그에 따른 해당 동작을 수행하는 것이며 실제로 손쉽게 가장 많이 사용하는 방법이다. 그러나 점차로 서비스가 복잡해짐에 따라 현재의 상황에 따라서 제어되어야 할 내용이 변하는 환경에 적용하기가 쉽지 않다. 이를 해결하기 위하여 상황 조건에 따른 분기를 기술한 상황 흐름도로 서비스를 모델링하고 각 상황별로 제어될 내용을 기술하는 방법을 사용하였다. 이러한 접근방법은 워크플로우(workflow)[13]의 개념을 단순화한 것으로 상황 조건별 다양한 분기를 표현하기 용이하고, 구조적으로 상태 전이를 활용한 상황 인식기법과 동일한 구조로 작동될 수 있기 때문에 그림 2의 상태 전이 엔진을 그대로 활용할 수 있다. 다음의 그림 3은 동작 발생기의 동작원리를 나타내고 있다.

상태 변화에 따른 제어되는 내용들을 표현하기 위하여 상황 인식기에서와 마찬가지로 XML기반의 언어인 *uADML(u-Action Generator Markup Language)*를 정의하였는데 *uSDML*과 구조가 거의 동일하므로 같은 파서를 사용할 수 있다. *uSDML*과의 차이점은 모든 스테이트에 발생될 제어 내용을 포함할 수 있는데, 그 안에 한 개 이상의 제어될 내용들이 기술된다. 각각의 제

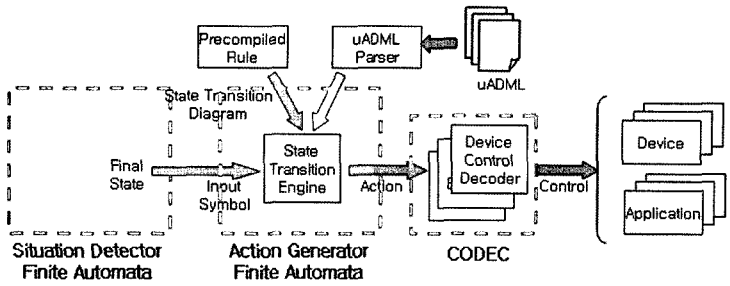

그림 3 동작 발생기의 동작원리

어 내용들은 기술된 순서대로 발생되는데, 이때 발생 지연시간을 추가할 수 있다. *uSDML*의 경우와 마찬가지로 본 논문에 적용된 동작 발생기에서는 *uADML*과 파서를 사용하지 않고 직접 프로그램으로 유한 오토마타 구조체를 생성하여 범용 상태전이 엔진에 등록하는 방법을 사용하였다.

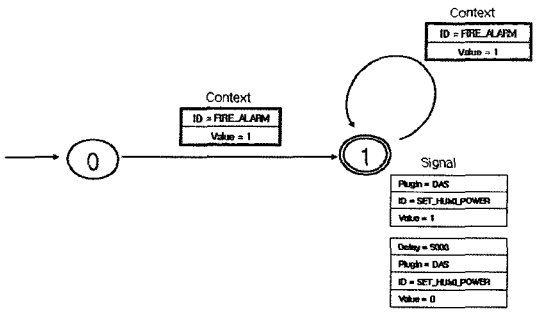

그림 4 Fire-Alarm을 정의한 Action의 예

위의 그림 4는 알람이 발생되는 서비스를 상태 전이도를 이용하여 모델링한 예를 나타내고 있다. 초기 상태인 *<state 0>*에서 알람이 발생되어야 하는 상황이 인식되면 상황 인식기에서 *FIRE_ALARM* 상태가 발생되고 그에 따른 *<state 1>*로의 상태전이를 일으키게 된다. *<state 1>*에는 제어될 내용이 정의되어 있는데 *DAS*라는 *CODEC*으로 가습기 전원을 켰다가 5000ms 후에 전원을 끄도록 하는 액션을 기술한 예이다.

4. 상황-동작 변환기의 응용

본 논문에서 제안한 상황-동작 변환기를 이용하여 상황 관리기를 구현하였다. 테스트에 사용된 시나리오는 스마트 스케줄링 서비스인데, 약속시간에 늦지 않기 위하여 지속적으로 목적지까지 걸리는 시간을 교통정보 서비스를 통하여 모니터링 하면서 출발시간을 자동으로 조절하여 출발해야 하는 시간에 알람을 울려주는 서비스이다. 이 시나리오에서 상황 관리기는 알람이 발생되어야 하는 상황을 감지하여 그에 따른 실제 알람 서비스에 제어 신호를 발생시키는 역할을 담당하였다. 다음의 그림 5는 상황 관리기가 실제로 탑재된 ARM기반 테스팅 보드와 테스팅 환경의 사진을 나타내고 있다.

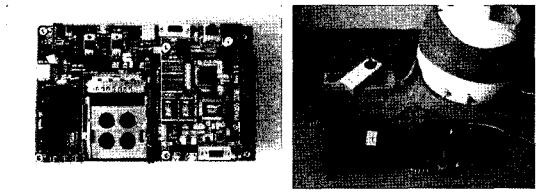

그림 5 임베디드 테스팅 보드와 테스팅 환경

상황 관리기는 OSGi Release 3 상에서 동작되도록 모두 9개의 OSGi 번들로 개발되었으며, OSGi Framework로는 Knopflerfish 1.3.4가 사용되었다[18,21]. JVM은 ARM 프로세서용 JVM인 Blackdown-1.3.1-RC1가 사용되었다[16]. 사용된 임베디드 보드는 PXA255 400Mhz 프로세서, 64MByte RAM, 64MByte Flash Memory가 장착된 범용 테스트보드이며 운영체제는 임베디드 리눅스 2.4.19 커널이 사용되었다. JDK와 OSGi Framework, 그리고 개발된 상황 관리기는 CF Memory에 저장되어 PCMCIA 슬롯에 장착되어 사용되었다. 실제 알람 서비스를 대체할 장비 대신 가습기가 사용되었으며, 가습기 제어는 전력선 제어를 위하여 *MAT*사의 iPlug CR310 게이트웨이와 On/Off가 지원되는 플러그가 사용되었고, 1개의 상황인식 규칙과 1개의 동작발생 규칙이 사용되었다.

상황 관리기의 처음 시작하는데 대략 30여초 걸렸으며 그 동안 CPU 점유율 100%를 유지했는데, 이는 JDK와 OSGi Framework, 상황 관리기 번들들 모두가 엑세스 속도가 상대적으로 느린 CF 메모리에 저장되어 있었기 때문으로 추측된다. 상황 관리기 실행 시에 모두 15개의 JVM이 동시에 작동되는데, 이는 Knopflerfish OSGi framework에서 번들들을 운영하는데 사용하기 때문이다. 다음의 그림 6은 상황처리기 작동시의 CPU

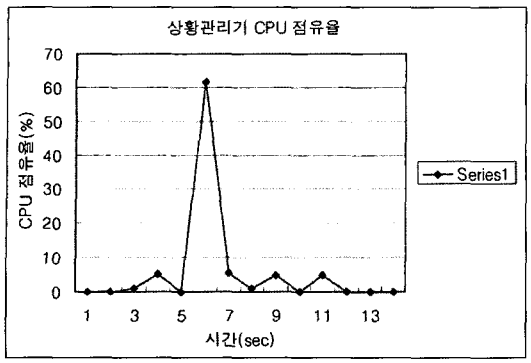

그림 6 상황처리기의 CPU 점유율

점유율을 나타내고 있다. CPU 점유율의 측정은 Linux 명령어인 top을 사용하여 프로세스별 CPU 점유율을 1초에 한 번씩 모니터링 하였다.

대부분의 경우 5~5.5% 정도의 CPU 점유율을 보이다가 상황 인식과 관련하여 순간적으로 CPU 점유율이 61.6%를 기록하였다. 하지만 실제 점유율은 이 수치보다는 낮을 것으로 예상되는데, 그 이유는 현재는 상황 인식 여부를 콘솔로 로그를 남기도록 처리했기 때문이다.

상황-동작 변환기의 정량적 성능분석은 쉽지 않은데, 그 이유는 상황 인식이라는 측면으로 보면 추론을 이용하는 방법, 규칙을 이용하는 방법 그리고, 본 논문에서와 같은 상태전이 방법 등이 서로 강점을 보이는 적용 도메인과 상황 인식 수준이 다르기 때문이다. 추론 방법과 같이 시간이 오래 걸리더라도 하나의 사실로부터 여러 가지의 현재 상황을 유추해 내는 것이 유리할 수도 있고, 각각의 장비에서 스스로 간단한 몇 가지의 상황을 빠르게 인식하여 동작하는 것이 유리할 수도 있기 때문에 이들 간의 상황 인식 성능 비교는 큰 의미가 없다. 다만, 본 논문에서 제안하는 상황-동작 발생기의 경우 기존의 홈네트워크 장비에서 사용되는 규칙등록 분야에 적용할 수 있기 때문에 이러한 장비와의 성능, 기능 비교는 가능하다. 홈네트워크 장비로 EIB를 따르는 *JUNG* 제품과의 성능 비교한 결과 상황에 따른 반응속도에서는 *JUNG*사의 제품이 대략 0.5초 이내로 본 논문의 1초 정도의 반응속도보다 빠르게 반응했으나 이는 본 논문에서 사용된 가전제어가 *LonWorks*기반의 전력선 제어 장비를 사용했기 때문에 상대적으로 느린 가전제어 반응속도를 나타낸 것으로 보인다. 반면에 *JUNG* 사의 제품은 등록될 수 있는 상황 규칙이 매우 제한적이고 무엇보다도 시간의 흐름에 따른 상황 변화를 반영할 수 없기 때문에 기능적인 면에서는 본 논문의 방법이 훨씬 다양한 상황을 표현할 수 있었다.

5. 결론 및 향후 연구과제

본 논문에서 제안한 상황 동작 변환기는 상황 모델링이 쉽고 기존의 센싱 패턴 방식과 유사하면서도 강력하고 실제 적용하기 쉬운 상황 표현 방법과 표준화된 입력 심볼의 CODEC 과정을 통한 다양한 형태의 입력을 수용할 수 있어 그 활용도가 매우 높았다. 데모 시스템을 임베디드 리눅스 기반의 OSGi Framework에서 구현하였기 때문에 동적 업데이트, 관리의 편의성 등과 같은 임베디드 시스템에서의 OSGi가 갖는 모든 장점들을 가질 뿐만 아니라, 유한 오토마타로 상황을 인식하기 때문에 인식기의 구조가 간단하고 많은 컴퓨팅 파워를 요구하지 않아서 임베디드 시스템에 적합한 방법임을 알 수 있었다. 또한 본 시스템의 입출력을 하나의 16bit 값만이 존재하는 시그널 형태로 처리하도록 배려하여 상황-동작 변환기의 H/W 칩화도 가능할 것이다.

그리고, 간단하고 표준화된 상태 전이 상황 모델은 자동화 도구와 이를 응용한 상황 모델 자동 생성기에 대한 연구도 함께 진행될 때 보다 가치 있을 것이다. 또한, 기본적인 상황 표현 모델이 음성 인식기반 제어 시스템의 개념과 유사하기 때문에 이미 연구 진행된 음성 인식기반 상황 인식에 적용된 연구 내용들을 적용한다면 보다 자동화 및 지능화를 개선할 수 있을 것이다. 그리고 연구의 결과물을 FPGA화 등 H/W 칩 화에 대한 연구 역시 함께 진행되어야 할 내용이다.

참 고 문 헌

[1] Glenn Judd and Peter Steenkiste, "Providing Contextual Information to Ubiquitous Computing Applications," http://www.cs.cmu.edu/~aura/services/, July 2002.

[2] "Gaia Active Space for Ubiquitous Computing," http://gaia.cs.uiuc.edu/

[3] Shankar R. Ponnekanti, Brian Lee, Armando Fox, Pat Hanrahan, and Terry Winograd, "ICrafter: A Service Framework for Ubiquitous Computing Environments," Ubicomp 2001.

[4] Bart Eisenberg, "People-centric Computing: MIT's Project Oxygen," Pacific Connection Web Site, October, 2002.

[5] Stephen S. Yau, Fariaz Karim, Yu Wang, Bin Wang, and Sandeep K.S. Gupta, "Reconfigurable Context-Sensitive Middleware for Pervasive Computing," Pervasive Computing, July-September 2002.

[6] Chen, Harry, Tim Finin, and Anupam Joshi. "An Intelligent Broker for Context-Aware Systems," Adjunct Proceedings of Ubicomp 2003, Seattle, Washington, USA, October 12-15, 2003.

[7] Ernest J. Friedman-Hill, "Jess 7.0 Manual Version

7.0b7 DREFT," http://www.jessrules.com/jess/index.shtml, 11 May 2006.
[8] Haarslev, V., Möller, R., "RACER System Description," Lecture Notes in Computer Science 2083 (2001) 701.
[9] "Speech Recognition Grammar Specification Version 1.0," W3C Recommendation, 16 March 2004.
[10] Buxton, W., "A Three-State Model of Graphical Input. In D. Diaper et al. (Eds), Human-Computer Interaction," INTERACT '90. Amsterdam: Elsevier Science Publishers B.V. (North-Holland), 449-456, 1990.
[11] Thomas A. Henzinger, Zohar Manna, Amir Pnueli, "Temporal Proof Methodologies for Real-time Systems," ACM 089791-419-8/90/0012/0353, 1990.
[12] Inhye Kang, Member, IEEE, Insup Lee, Senior Member, IEEE, and Young-Si Kim, "An Efficient State Space Generation for the Analysis of Real-Time Systems," IEEE TRANSACTIONS ON SOFTWARE ENGINEERING, VOL. 26, NO. 5, MAY 2000.
[13] 한주현, 김은희, 최재영, "유비쿼터스 환경을 위한 웹 서비스 기반 워크플로우 언어 설계", 한국정보과학회 2004 춘계학술발표논문집 제31권 제1호, pp.28-30, 2004년 4월.
[14] "EIB System: Logical Topology," Konnex Association, http://support.eiba.com/, Feb 25 2005.
[15] Pierre Guillemin, "The European Home Systems Protocol - Concepts and Product," http://www.domotics.com/homesys/HSpapers/EHSproto.htm, 1996.
[16] "Java Platform 2 for Linux," http://www.blackdown.org/java-linux/java2-status/index. html, Dec 2005.
[17] Xiao Hang Wang, Da Qing Zhang, Tao Gu, Hung Keng Pung, "Ontology Based Context Modeling and Reasoning using OWL," IEEE International Conference on Pervasive Computing and Communications, 2004.
[18] "OSGi Service Platform Release 3," IOS Press, http://www.osgi.org/, March 2003.
[19] "The Jakarta Project, commons digester 1.7 Release," http://jakarta.apache.org/commons/digester/, June 2005.
[20] "OWL Web Ontology Language: Overview," W3C, http://www.w3c.org/2004/OWL/, 2004.
[21] "Knopflerfish OSGi R3 framework," http://www.knopflerfish.org/index.html, 2006.

허 길

1995년 단국대학교 전자계산학과 학사 졸업. 1997년 숭실대학교 컴퓨터학과 석사 졸업. 2001년 숭실대학교 컴퓨터학과 박사 수료. 1996년~1998년 (주)세기통신 개발부 팀장. 2000년~2004년 (주)네오퍼스 개발팀 연구원. 2004년~현재 아주대학교 유비쿼터스시스템 연구센터 SA솔루션개발팀 책임연구원. 관심분야는 임베디드시스템, 컨텍스트 프로세싱, 폰트

박 희 정

2003년 광운대학교 환경공학 학사 졸업 2001년~2003년 플럼라인 개발부 연구원 2003년~2003년 미디어포드 개발부 연구원. 2003년~현재 아주대학교 유비쿼터스 시스템 연구센타 SA솔루션개발팀 전임연구원. 관심분야는 지능형 홈네트워크, 유비쿼터스 시스템, RFID

조 위 덕

1981년 서강대학교 전자공학과(학사) 1983년 한국과학기술원 전기 및 전자공학과(석사). 1987년 한국과학기술원 전기 및 전자공학과(박사). 1983년~1990년 금성전기(현LG전자) 기술연구소 DSP 연구실장. 1990년~1991년 한국생산기술연구원 전자정보시스템연구부 팀장/조교수. 1991년~2003년 전자부품연구원 시스템연구본부 본부장. 2003년~현재 유비쿼터스컴퓨팅사업단 단장, 아주대학교 전자공학부 교수. 관심분야는 유비쿼터스 컴퓨팅/네트워크, 센서 네트워크, Post-PC (차세대 Smart PDA), Interactive DTV 방송기술, 고품질 홈서버/게이트웨이기술, 디지털방송/이동통신 연계 융합플랫폼기술, 무선인터넷응용기술

최 재 영

정보과학회논문지 : 시스템 및 이론
제 33 권 제 7 호 참조