KIPS Tr. Software and Data Eng. 휴리스틱 입력 분석을 이용한 RRT 기반의 Simulink/Stateflow 모델 테스트 케이스 생성 기법 829
Vol.2, No.12 pp.829~840 pISSN: 2287-5905 http://dx.doi.org/10.3745/KTSDE.2013.2.12.829

Generating Test Cases of Simulink/Stateflow Model Based on RRT Algorithm Using Heuristic Input Analysis

Hyeon Sang Park† · Kyung Hee Choi†† · Ki Hyun Chung†††

ABSTRACT

This paper proposes a modified RRT (Rapidly exploring Random Tree) algorithm utilizing a heuristic input analysis and suggests a test case generation method from Simulink/Stateflow model using the proposed RRT algorithm. Though the typical RRT algorithm is an efficient method to solve the reachability problem to definitely be resolved for generating test cases of model in a black box manner, it has a drawback, an inefficiency of test case generation that comes from generating random inputs without considering the internal states and the test targets of model. The proposed test case generation method increases efficiency of test case generation by analyzing the test targets to be satisfied at the current state and heuristically deciding the inputs of model based on the analysis during expanding an RRT, while maintaining the merit of RRT algorithm. The proposed method is evaluated with the models of ECUs embedded in a commercial passenger's car. The performance is compared with that of the typical RRT algorithm.

Keywords : Model Based Testing, Rapidly-Exploring Random Tree, Simulink/Stateflow, Test Case Generation, Satisfiability Modulo Theory

휴리스틱 입력 분석을 이용한 RRT 기반의 Simulink/Stateflow 모델 테스트 케이스 생성 기법

박 현 상† · 최 경 희†† · 정 기 현†††

요 약

본 논문은 Simulink/Stateflow 모델 기반의 테스트 케이스를 자동으로 생성하기 위하여, 휴리스틱 입력 분석을 이용한 Rapidly-exploring Random Tree(RRT) 기법을 제안한다. RRT는 모델 기반 블랙박스 테스트 케이스 생성 시 반드시 해결해야 되는 도달 가능성 문제를 효율적으로 해결할 수 있는 방법이지만, 모델의 내부 상태와 테스트 목표를 고려하지 않고 무작위로 모델의 입력을 생성하기 때문에 테스트 케이스 생성 효율이 떨어지는 단점이 있다. 제안하는 기법에서는 RRT를 확장해나갈 때 필요한 입력을, 모델의 현재 상태에서 만족 할 수 있는 테스트 목표를 분석하고 이를 달성할 수 있는 모델의 입력을 분석 결과에 따라 휴리스틱하게 결정함으로써, RRT의 장점을 보존하면서, 테스트 케이스 생성 효율을 높일 수 있다. 제안된 기법은 자동차에 사용되는 실 부품 ECU의 Simulink/Stateflow 모델을 대상으로 한 실험을 통해 성능이 평가되었으며, 기존 RRT와 비교하여 테스트 케이스 생성 효율이 높은 것을 보였다.

키워드 : 모델 기반 테스팅, Rapidly-Exploring Random Tree, Simulink/Stateflow, 테스트 케이스 생성, Satisfiability Modulo Theory

1. 서 론

산업에서의 내장 시스템에 대한 요구와 이용 사례가 급속도로 증가함에 따라 내장 시스템에 대한 테스트의 중요성이 증가하고 있다. 내장 시스템을 테스트 하는 여러 기법 중 블랙박스 상태에서의 모델 기반 테스트 기법[1]이 널리 사용되고 있다. 내장 시스템의 모델을 확보하게 되면, 모델을 기반으로 시스템의 개발에서부터 테스트까지의 과정을 진행할 수 있게 된다[2].

산업체에서 내장 시스템의 모델링 기법으로 많이 사용하고 있는 기법 중 Mathworks 사의 Simulink/Stateflow[3]가 있다. Simulink는 시스템의 동작을 동작 블록과 블록들의 연결 선으로 표현하며, 내장시스템의 기능 중 하드웨어적인 특성을 표현하기에 적합하다. Stateflow는 Simulink의 동작 블록 중 하나이지만, 블록의 동작을 상태 다이어그램으로 표현하며, C나 MATLAB 언어와 비슷한 코드로 시스템의

† 준 회 원 : 아주대학교 정보통신전문대학원 정보통신공학과 박사과정
†† 정 회 원 : 아주대학교 정보통신전문대학원 정보통신공학과 교수
††† 정 회 원 : 아주대학교 전자공학과 교수
논문접수 : 2013년 7월 19일
수 정 일 : 1차 2013년 9월 25일
심사완료 : 2013년 9월 30일
* Corresponding Author : Hyeon Sang Park(elvaimay@ajou.ac.kr)

상세한 동작을 모델링 할 수 있기 때문에, 내장 시스템의 기능 중 소프트웨어적인 특성을 표현하기에 적합하다.

모델 기반 테스트를 수행하기 위해서는 테스트에 필요한 테스트 케이스를 생성해야 한다. 기능이 단순하고, 시스템의 오 동작이 치명적인 결과를 발생시키지 않은 내장 시스템에 대해서는 시험자의 경험이나, 직관에 따라 생성한 테스트 케이스를 테스트에 이용 할 수 도 있다. 그러나 자동차, 항공기, 선박 등에 사용되는, 많은 기능이 요구 되고, 사용자의 안전 또한 중요한 내장 시스템에 대해서는 시험자가 수동으로 테스트 케이스를 생성하기에는 시스템의 규모와 복잡도가 매우 크다. 따라서 충분한 강도의 테스트를 수행할 수 있는 테스트 케이스를 내장 시스템의 모델로부터 자동으로 생성 할 수 있는 기법이 필요하다.

내장 시스템의 블랙박스 환경 모델 기반 테스트를 수행하기 위해 테스트 케이스를 생성하는 문제는, 도달 가능성 문제[4]와 같다. 블랙박스 환경에서 시스템을 테스트하기 위해서는 시스템의 외부 입력만을 이용하여 시스템의 내부 상태를 테스트 하기 위한 상태로 이끌어나가야 된다. 이 때 시스템의 특정 상태에 도달하는 시스템의 외부 입력의 순열을 찾는 문제가 도달 가능성 문제와 같게 된다. 도달 가능성 문제는 결정 불가능한 문제여서 해결 방법을 찾기가 매우 어려우며[5, 6], 그 근사치를 구하기 위한 기법도 매우 많은 비용을 요구한다[7]. 이러한 문제를 해결하고자 Simulink/Stateflow 모델 기반의 테스트 케이스 자동 생성 기법의 많은 선행 연구들이 이루어졌다. 상업적 도구로는 [8, 9]가 있으며, 학술 연구 결과로는 [10, 11]가 있다.

본 논문에서는 도달 가능성 문제를 해결하기 위한 효과적인 방법으로, 로봇 공학에서 로봇의 동작 경로를 탐색하기 위한 효과적인 기법인 Rapidly-exploring Random Tree (RRT)를 테스트 케이스 생성 기법으로 활용하였다. 테스트 케이스 생성을 위해 RRT를 사용한 연구는 주로 하이브리드 오토마타[12] 모델을 대상으로 이루어졌으며, [7]에서는 RRT를 테스트 케이스 생성에 이용하는 것과 도달 가능성 문제 해결과의 관계를 서술하고 있다.

RRT는 내장 시스템의 다양한 상태를 빠르고 조밀하게 탐색해 나갈 수 있으며, 모델의 동작 특성을 고려하지 않고, 적용이 가능하다는 장점이 있다. 그러나 원하는 목적의 테스트 케이스를 생성하기 위해서는 내장 시스템을 복수 개의 원하는 상태로 이끌어 나가야 된다. 이를 위해서는 시스템의 다양한 상태를 찾아 나가면서 목표 상태에 도달하는 RRT의 전략보다는 다수의 목표 상태가 존재하는 상황에서 RRT를 확장하여 최대한 많은 목표 상태에 빠르게 도달 할 수 있는 기법을 고안하는 것이 더 적합하다.

본 논문에서는 Simulink/Stateflow 모델의 전이 커버리지를 달성시키기 위한 테스트 케이스 생성 기법을 제안한다. 달성되지 않은 전이 목표를 달성시키기 위해서, RRT 확장 시 현재 RRT가 확장되고 있는 내장 시스템의 상태와, 그 상태에서 발생해야 되는 전이와, 전이의 조건을 고려하여, RRT 확장을 위한 시스템의 입력을 결정한다. 기존의 RRT 확장 기법은 시스템의 상태나 전이의 조건 등을 고려하지 않고, RRT 무작위 노드에 가까운 방향으로 확장 할 수 있도록 시스템의 입력을 결정한다. 그러나 제안한 기법에서는 RRT 무작위 노드에 가까운 방향이 아닌 목표 전이를 바로 발생시킬 수 있는 입력을 생성하여, 불필요한 RRT 확장 비용을 줄일 수 있어, 테스트 목표 달성율을 높일 수 있다. 또한 입력 분석이 불가능 할 때에는 기존 RRT 확장 기법에서 사용하는 입력 결정 방식을 그대로 사용함으로써, 기존 RRT 의 장점 또한 가져갈 수 있다.

본 논문의 구성은 다음과 같다. 2장에서는 Simulink/Stateflow 모델의 설명 및 Simulink/Stateflow 모델기반 테스트 케이스 생성 기법과, 본 논문에서 사용할 기법인 RRT와 SMT에 대해 설명하며, 3장에서는 모델의 상태에 따른 시스템 입력 분석 기법과 이를 이용한 테스트 케이스 생성 기법을 기술한다. 4장에서는 차량 ECU 명세를 모델링한 Simulink/Stateflow 모델을 이용하여 제안된 테스트 케이스 생성 기법의 성능을 평가한다. 5장에서는 지금까지 연구의 결론을 정리하여 기술한다.

2. 연구 배경 및 관련 연구

2.1 Simulink/Stateflow 모델

Simulink/Stateflow는 상태 기반의 시스템 모델링 기법이다. 대상 시스템의 명세를 몇 개의 주요한 이산적인 상태로 모델링 한 후 각 상태 간 전이 조건이나 전이 발생 후의 동작을 기술한다. 한 상태는 같은 상태로 전이 되거나, 다른 상태로 전이 될 수 있으며, 한 상태에서 벌어 질 수 있는 전이는 복수 개가 될 수 있으므로, 동시에 여러 전이가 발생할 수 있을 때의 문제를 해결하기 위한, 각 전이 간의 우선 순위가 존재한다.

Fig. 1은 Simulink/Stateflow 모델의 예제로써, 팬 모터를 제어하는 자동차 ECU의 명세를 모델링 한 것이다. 상태는 둥근 사각형으로 표시되며, 예제에서는 *"FanOff", "FanLo", "FanHi"*의 세 가지 상태를 가진다. 상태 사각형 안에는 해당 상태에 진입하였을 경우 수행해야 되는 동작이 기술되어 있다. 즉 모델이 *"FanLo"* 상태에 진입하면 *"MotorAim=0.75"*라는 동작이 수행된다. 전이는 화살표로 표시된다. 전이는 한 상태에서 다른 상태로 이어지며, 검은 점에서 상태로 연결되는 특수한 전이가 가리키는 상태가 시스템의 초기 상태가 된다. Fig. 1의 예제에서는 *"FanOff"*상태가 시스템의 초기 상태가 된다. 한 상태에서 나아가는 전이가 두 개 이상일 경우 전이의 시작 지점에 우선순위가 숫자로 표시 된다. 각 전이의 발생 조건은 대괄호 '[', ']' 사이에 기술 되며 전이가 발생하고 난 후 이루어져야 하는 동작은 전이 조건 다음 '/' 뒤에 기술된다. 예제 모델은 총 11개의 전이를 가지고 있다.

예제 Stateflow 모델에서 입력 변수는 *"TempSensor", "Mode"*이며 출력 변수는 *"MotorAim"*이다. 내부 변수는 *"LoTimer", "HiTimer"* 두 개가 존재하며 *"OFF", "LO", "HI", "LOTEMP", "HITEMP", "LOTIME", "HITIME"*은

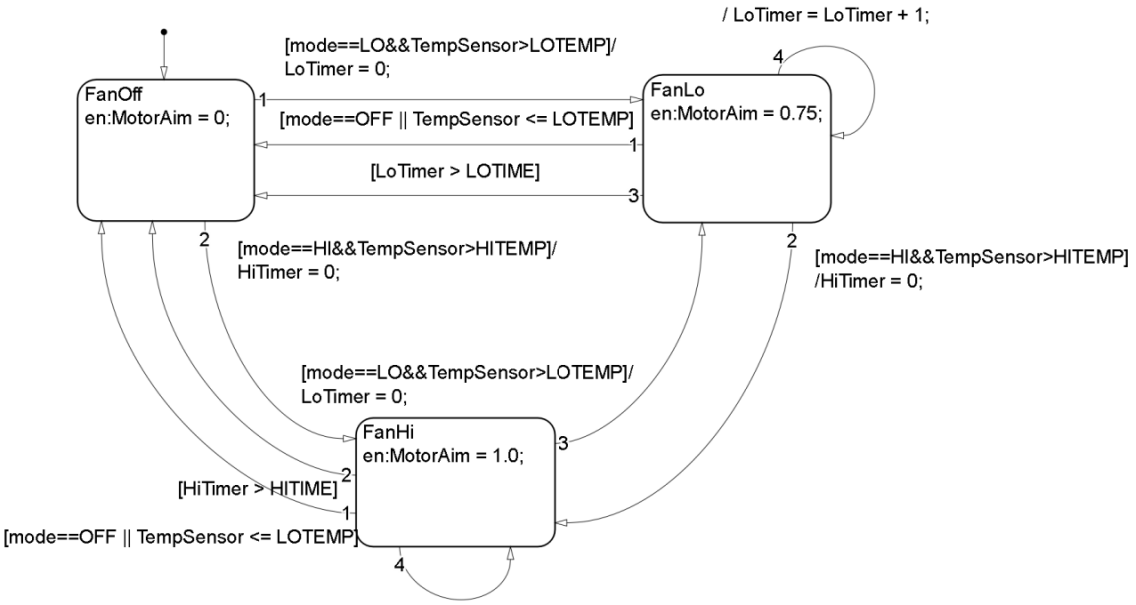

Fig. 1. Simulink/Stateflow diagram of fan control logic

상수로써, 각각 0, 1, 2, 20, 30, 120, 80의 값을 가진다. *"TempSensor"*는 0에서 100의 값, *"LoTimer", "HiTimer"*는 0에서 200의 값, "MotorAim"의 값은 0에서 1의 값을 가진다.

모델의 동작을 간단히 설명하면, 사용자 모드는 *"HIGH", "LOW", "OFF"*로 이루어져 있다. 사용자가 *"LOW"*모드를 선택했을 때, 현재 온도가 *"LOTEMP"* 보다 높다면 모터는 최대 출력의 75%로 동작한다. *"LOW"* 모드로 동작하는 시간이 *"LOTIME"*만큼 지나게 되거나, 현재 온도가 *"LOTEMP"* 이하일 경우, 또는 사용자 모드가 *"OFF"*가 될 경우 모터는 동작하지 않는다. 사용자가 *"HIGH"*모드를 선택했을 때, 현재 온도가 *"HITEMP"*보다 높다면 모터는 최대 출력의 100%로 동작한다. *"HIGH"*모드로 동작하는 시간이 *"HITEMP"*만큼 지나게 되거나, 현재 온도가 *"LOTEMP"* 이하일 경우, 또는 사용자 모드가 *"OFF"*가 될 경우 모터는 정지한다. 사용자 모드가 *"LOW"*거나 *"HIGH"*일 경우 온도 조건이 맞으면 *"HIGH"*나 *"LOW"*로 모드 변경이 가능하다.

2.2 Simulink/Stateflow 테스트 케이스 생성 기법

Simulink/Stateflow 모델을 이용한 테스트 케이스 생성 기법은 상업적 도구로나, 연구 결과로 많은 선행 연구가 이루어졌다. 먼저 상업적 도구로써 가장 대표되는 테스트 케이스 생성 도구는 Reactis[8]와 Design Verifier[9]가 있다. Ractis는 테스트 케이스를 생성하기 위한 입력을 휴리스틱 기반으로 모델을 분석 한 후 이를 이용하여 무작위로 입력 값을 선택하여 테스트 케이스를 생성한다. Design Verifier는 모델의 역함수 분석을 이용하여, 빠르게 테스트 케이스 생성이 가능하지만, 분석이 어려운 사용자 정의 블록 등이 모델에 포함되어 있을 경우, 테스트 케이스 생성 효율이 떨어진다.

비상업적인 연구 결과 중 대표적인 것을 살펴보면, [10]에서는 모델을 SMV[13] 프로그램 형태로 바꾼 다음 정적 분석을 이용하여 테스트 케이스를 생성한다. Simulink/Stateflow 모델을 다른 형태로 변환하여 테스트 케이스를 생성하는 기법의 문제점은, 모델의 크기가 커질 경우 분석 과정에서 매우 많은 비용을 소모하게 되며, 모든 Simulink/Stateflow 모델에 대해 변환을 할 수 있다는 것을 증명하기 어렵다는 것이다. [11]에서는 무작위 기반 테스트 케이스 생성을 바탕으로 모델 분석을 통해 원하는 테스트 케이스 생성을 위한 입력을 바로 찾아 낼 수 있는 경우는 분석된 입력을 사용하여 테스트 케이스 생성에 효율을 높였다. 이 연구와 본 논문의 차이점은 본 논문에서는 무작위 생성 기법으로 RRT를 선택하였으며, 입력 분석 방법으로 SMT 엔진[14]를 기반으로 한 여러 휴리스틱 기법들을 추가로 적용하였다.

2.3 Rapidly Exploring Random Trees(RRT)

시스템은 동작 중 특정 시점에서의 상태를 시스템의 내부 변수들의 값 조합으로 나타낼 수 있다. RRT 알고리즘에서는 특정 시점에서의 시스템의 상태를 트리의 노드로 정의하며, 시스템의 초기 상태를 트리의 루트 노드로 정한다. RRT 알고리즘의 목적은 트리의 루트 노드부터 RRT 확장을 시작, 도달하고자 하는 시스템의 상태에 해당하는 RRT 노드까지 RRT 트리를 확장하여, RRT 루트 노드로부터 RRT 목표 노드 까지의 경로를 구하는 것이다.

Table 1은 RRT 알고리즘의 기본적인 형태이다[15].

RRT 알고리즘에서 가장 중요한 것은 RRT 노드 간의 거리 함수를 정의하는 것이다. 거리 함수는 RRT 노드 간의 유사도를 나타내는 것으로 거리가 가까울 수록 두 RRT 노드의 상태는 유사하며, 반대로 거리가 멀수록 노드의 상태는 달라진다. 거리 함수가 RRT 노드 간의 거리를 얼마나 목적에 맞게 정확하게 정의하는가에 따라 RRT 알고리즘의 성능이 결정된다[16].

Table 1. Typical RRT extension algorithm[15]

```
Extends (nd_init, K, Δt)
T.init(nd_init)
for k=1 to K do
  nd_rand ← random_node();
  nd_near ← nearest_ node(nd_rand, T);
  v ← select_input(nd_rand, nd_near);
  nd_new ← new_ node(nd_near, v, Δt);
  T.add_vertex(nd_new);
  T.add_edge(nd_near, nd_new, v);
return T
```

RRT 확장 알고리즘의 파라메터는 트리 확장의 시작이 되는 시스템의 초기 상태 노드 nd_{init}, 트리의 확장 횟수 K, 트리 1회 확장 시 시스템의 시간 변화량 Δt가 된다. 알고리즘에서는 RRT의 루트 노드 nd_{init} 으로 트리를 초기화 한 후, K번의 RRT 확장이 이루어 진다. 처음으로, RRT 확장의 방향을 정하는 무작위 시스템 상태 값을 가진 RRT노드 nd_{rand}를 생성한다. 그리고 정의된 거리 함수를 이용하여 RRT 트리에 포함된 노드 들 중 nd_{rand}와 가장 가까운 노드 nd_{near}를 결정한다. 이 후 RRT는 nd_{near}에서 새로운 노드로의 확장을 수행한다. 이 때 시스템의 상태를 nd_{rand}로 가깝게 변화시키기 위한 입력 v를 결정한다. 마지막으로 결정된 입력 v를 Δt시간 만큼 시스템에 적용하여 생성된 새로운 시스템 상태 nd_{new}를 RRT의 새로운 노드로 추가한다.

2.4 Satisfiability modulo theory(SMT)

SMT 문제[17]는 일차 논리(first-order logic)식을 참으로 만족시키는 일차 논리를 구성하는 변수의 값들이 존재하는 가를 판단하는 문제이다. 컴퓨터 공학의 소프트웨어 분야에서는 많은 문제들이 일차 논리로 표현이 가능하기 때문에 이 문제를 풀기 위한 기법들이 꾸준히 연구되고 있다[18].

SMT 문제를 풀기 위한 SMT 엔진(SMT Solver)들은 많은 종류가 있다[14]. SMT 엔진은 문제의 대상이 되는 일차 논리식을 입력하면 일차 논리식을 참으로 만족시키는 변수의 순서쌍을 결정해준다. Fig. 1의 상태 *"FanHi"*에서 *"FanLo"*로 가는 전이 조건식을 예로 들면, 전이 조건식은 일차 논리로 다음 식과 같이 표현 된다.

$$\text{mode} = \text{LO} \wedge \text{TempSensor} < LOTEMP$$

이 일차 논리식의 구성을 살펴보면, 변수는 *"mode", "TempSensor"* 두 개가 되고, *"LO", "LOTEMP"*는 상수가 된다. 따라서 해당 일차 논리식의 SMT 문제를 풀기 위해서는 식을 참으로 만드는 변수 *"mode", "TempSensor"*의 값을 찾아야 하는 것이다.

SMT 엔진을 이용하여 위 식을 풀게 되면 다음과 같은 결과가 나온다.

$$\begin{cases} \text{mode} = 1 \\ \text{TempSensor} = 19 \end{cases}$$

상수 *"LO", "LOTEMP"*는 각각 1과, 20의 값을 가지므로, 위의 결과를 논리식에 대입하면 식이 참으로 만족 됨을 알 수 있다.

SMT 엔진은 논리식이 선형적이어야 문제를 풀 수 있다. 따라서, 전이 조건식이 선형적이지 않을 경우에는 SMT 엔진을 이용하여 조건 분석을 할 수 없다.

3. 입력 분석 기법 및 테스트 케이스 생성 기법

3.1 정의

1) Simulink/Stateflow 모델과 RRT

Simulink/Stateflow 모델에 RRT를 적용시키기 위한 RRT 노드와 RRT 거리 함수를 정의한다. 먼저 Simulink/Stateflow 모델의 구성 요소는 정의 1과 같다.

> 정의 1. Simulink/Stateflow model M은 (S, T, A, V)로 정의된다.
> $S=\{s_1, s_2, \ldots, s_m\}$: 모델의 상태 집합
> $T=\{t_1, t_2, \ldots, t_n\}$: 모델의 전이 집합
> $A=\{a_{s1}, a_{s2}, \ldots, a_{so}\}$: 모델의 활성화 된 상태 집합
> $V = \{V_1, V_2, \cdots, V_k\}$: 모델의 변수 집합

특정 시점에서 Simulink/Stateflow 모델의 상태는 모델의 활성화 되어 있는 Stateflow 다이어그램의 상태와, 그 때의 변수 값의 조합으로 모두 표현 가능하다. RRT 노드는 특정 시점에서의 시스템의 상태 값으로 이루어지므로, 정의 2와 같다.

> 정의 2. RRT 노드 nd는 (V, A)로 정의되어 모델의 변수의 값과 활성화 된 상태의 조합으로 나타낸다.

모델 M의 변수들은 각기 다른 데이터 형을 가지며 불리언, 정수, 실수 형이 그 종류이다. 각 변수들은 그 최대값과 최소값 등 성격이 모두 다르기 때문에 이를 반영해서 RRT 노드 간의 거리를 구할 수 있는 기법을 이용해야 한다. 본 연구에서는 Gower's General Similarity Coefficient[19]를 이용하여 변수 값들의 거리를 구하며, RRT 노드 간의 거리 함수는 정의 3과 같다.

> 정의 3. 두 RRT 노드 nd_x, nd_y의 변수 값의 거리 함수 $dv(nd_x, nd_y)$는 다음과 같이 정의 된다.
>
> $$dv(nd_x, nd_y) = \frac{\sum_{k=1}^{n(V)} ds_k(c_{kx}, c_{ky})}{n}$$
>
> $$\begin{cases} ds_k(a,b) = \begin{cases} 0 & if \quad a = b \\ 1 & if \quad a \neq b \end{cases} & if\ c_k\ is\ boolean\ or\ enum \\ ds_k(a,b) = \dfrac{|a-b|}{|c_{k\max} - c_{k\min}|} & if\ c_k\ is\ not\ boolean\ or\ enum \end{cases}$$

c_{kx}와 c_{ky}는 RRT 노드 nd_x, nd_y의 변수 c_k에 해당하는 값이며, c_{kmax}와 c_{kmin}은 시스템 명세에 기술된 c_k의 최대값과 최소값을 의미한다. $dv(nd_x, nd_y)$은 0에서 1 사이의 소수 값이며 노드 nd_x, nd_y의 변수 값이 같을수록 $dv(nd_x, nd_y)$의 값은 0에 가까워 지며 다를수록 1에 가까워진다.

2) 테스트 목표

제안하는 알고리즘에서는 Simulink/Stateflow 모델의 전이 커버리지를 달성하기 위한 테스트 케이스를 생성한다. Simulink/Stateflow 모델이 실행 될 때, 어떤 전이가 발생되었는지를 확인할 수 있어야 하며, 테스트 케이스 생성 시에 달성하고자 하는 전이 집합이 테스트 목표 집합이 된다. 정의 4는 모델 실행 시 발생한 전이 집합, 정의 5는 테스트 목표 집합이다.

정의 4. 전이 집합 *OT*는 Simulink/Stateflow 모델이 실행 되고 난 후, 발생된 전이들의 집합이다.

정의 5. 테스트 목표 집합 *TG*는 Simulink/Stateflow 모델을 테스트하기 위해 발생해야 되는 전이들의 집합이다.

테스트 케이스 생성 기법의 성능 지표가 되는 테스트 목표 달성율은 전체 테스트 목표 집합 *TG*의 전이 중 생성된 테스트 케이스가 몇 개의 전이를 발생시키는가로 나타낼 수 있다.

3.2 입력 분석 기법

RRT 확장이 이루어질 때, 입력이 결정되는 단계는2.3절에서 설명한 바와 같이, RRT 무작위 노드에서 가장 가까운 노드 nd_{near}가 결정이 된 다음 *select_input()*이 수행되는 단계이다. RRT 확장을 위한 입력 분석은 다음과 같은 단계로 이루어진다.

1) RRT 노드 nd_{near}에서 발생 시킬 수 있는 전이들 중 아직 발생된 적이 없는 전이를 선택
2) 선택된 전이의 조건 식을 구문 분석
3) 분석 결과에 따라 적용해야 할 입력 분석 기법 결정
4) 입력 분석 기법을 수행하여 입력 값을 결정

앞으로 각 단계에 대한 내용을 기술함으로써, RRT 확장 시 입력이 결정되는 과정을 설명한다.

1) 전이의 선택

RRT 노드의 정의에 따라 nd_{near}에서 활성화 되어 있는 Simulink/Stateflow 모델의 상태를 알 수 있다. 이 상태에서 나아가는 전이들 중 테스트 목표 집합에 속하는 전이를 임의로 선택 한다. 테스트 목표 집합에 속한 전이 중 이미 발생한 전이는 테스트 목표 집합에서 제외되므로, 이미 발생한 전이가 중복해서 선택되지는 않는다.

2) 조건식 구문 분석

발생시킬 전이가 결정되면 전이의 조건을 구문 분석한다. 구문 분석은 LEX/YACC[20]을 이용하여 수행하며, 구문 분석 결과로, 구문 분석 트리와, 조건에 사용된 변수들의 목록, 상수들의 목록과, 식의 연산자들을 추출 할 수 있다.

목표 전이를 발생시키기 위한 조건식 구문 분석의 결과는 크게 Table 2와 같이 나뉘어 지며 전이 조건 식의 구조와, 전이 조건 식에 사용된 변수들의 종류에 따라 각 경우가 나뉘며, 행해질 입력 분석 기법이 정해진다.

Table 2. The outcomes of transition condition parsing

Case a)	All variables are input variables.
Case b)	Some variables are input variables, other variables are local or output variables.
Case c)	All variables are local or output variables.
Case d)	The condition expression.is non-linear.

3) 입력 분석 기법 결정

조건 분석 구문 결과 경우에 따라 다른 형태의 입력 분석이 행해진다. 입력 분석은 기본적으로 SMT 엔진을 이용해서 이루어진다. 전이의 조건 식을 SMT 엔진에 입력하면, 조건 식을 참으로 만드는 변수들의 값이 구해진다. SMT 엔진은 Microsoft Z3[21] 엔진을 사용한다. SMT 엔진을 사용하여 입력을 분석하는 기법은 표 2와 같이 각 경우에 따라 나뉘어 적용된다.

a) 조건 식에 사용된 모든 변수가 입력 변수인 경우

이 경우는 가장 이상적인 경우로 조건식을 만족하는 변수 값이 SMT 엔진을 통해서 구해지면, RRT 확장의 입력 값으로 바로 사용하여, 목표 전이를 발생시키는 것이 가능하다.

b) 조건식에 사용된 변수 중 일부는 입력 변수이지만, 일부는 내부 변수, 또는 출력 변수인 경우

시스템의 내부 변수나 출력 변수는 테스트 케이스 생성시 직접 제어할 수 없는 변수이기 때문에SMT 엔진을 이용해서 조건식이 참이 되는 경우를 만드는 입력 변수의 값을 찾아 낸다 하더라도 변수에 직접 할당을 할 수 없게 된다. 따라서 내부 변수와 출력 변수는 상수로 취급하는데, 이 때의 변수 값은 RRT 확장의 기점이 되는 노드 nd_{near}의 해당 변수의 값이 되며, SMT 엔진을 이용하여 내부 변수와 출력 변수를 제외한 입력 변수 값을 결정하여 RRT 확장에 이용한다.

c) 조건식에 사용된 모든 변수가 내부 변수, 또는 출력 변수인 경우

이 경우는 SMT 엔진으로 분석 할 수 있는 변수가 없으므로, 목표 전이를 발생시키기 위한 입력 변수 값을 직접 결정 할 수 없다.

d) 조건식이 비선형적인 경우

이 경우는 SMT 엔진으로 조건 식을 분석할 수 없으므로, 목표 전이를 발생시키기 위한 입력 변수 값을 직접 결정 할 수 없다.

4) 입력 값 결정

3)에서 결정된 입력 분석 기법을 실행하여 입력 값을 결정한다. 입력 값이 결정될 때에는 목표 전이의 조건 식과 관련 있는 입력 변수 만이 값으로 결정되기 때문에, 값이 결정되지 않은 다른 입력 변수들의 값은 무작위로 결정된다.

그러나, 입력 값을 반드시 결정할 수 있는 것은 아니다. 시스템의 제약 조건을 위반한 경우, 전이 조건식이 잘못 기술된 경우 등에서는 입력 값을 결정 할 수 없게 된다. 따라서 표 2의 각 경우마다 입력 값이 결정되지 않을 경우의 상황을 처리해야 한다.

경우 a)에서 값을 찾는데 실패한다면, 해당 전이의 조건은 불가능한(infeasible) 조건이기 때문에 절대로 발생할 수 없게 되어 테스트 목표에서 제외된다.

경우 b)와 c), d)에서는 내부 변수와, 출력 변수 때문에 전이 조건을 만족하기 위한 입력 값을 결정 할 수 없다. 이 상황은, RRT 확장의 기점이 되는 RRT 노드 nd_{near}의 상태에서는 목표 전이를 바로 발생시킬 수 없는 상황일 가능성이 높기 때문에, 내부 변수나, 출력 변수의 값을 변화 시키는 입력 값을 이용해서, 시스템의 상태를 다른 상태로 변화 시켜, 목표 전이를 발생 시킬 수 있는 상태로 근접시키는 전략을 취해야 한다.

3.3 목표 전이를 발생 시킬 수 있는 상태의 근접 전략

이 전략에서는 RRT 확장을 위한 입력을 바로 결정 할 수 없을 때, 목표 전이의 조건 식을 만족시킬 수 있는 상태로 접근시키는 기법을 설명한다. 기법의 기본적인 전략은, L번 만큼의 무작위 입력 값으로 RRT 확장을 시도 하고 생성된 RRT 노드 들 중 목표 전이를 발생 시킬 수 있는 상태로 가장 근접한 RRT 노드를 생성한 입력을 RRT 확장의 입력으로 사용하는 것이다. 이 때 RRT 노드의 상태가 목표 전이의 발생에 얼마나 근접했느냐를 측정 할 수 있는 식의 정의가 필요하다.

Simulink/Stateflow의 한 상태 S로부터 나가는 우선순위 p의 전이의 발생 조건 $C_{s,p}$는 논리연산자 AND(&&)나 OR(||)로 나뉘어진 t개의 단위 조건 $\{ac_{s,p,1}, ac_{s,p,2}, ac_{s,p,3}, \cdots, ac_{s,p,t}\}$으로 이루어진다. 이 때, 3.2절의 2)의 조건 식 구문 분석 기법을 이용하여, 단위 조건에 사용된 Simulink/Stateflow 모델 변수를 분석한 후 조건 식에 내부 변수나, 출력 변수만을 포함하는 단위 조건만으로 이루어진 집합을 $LAC_{s,p}$이라고 하자. 이를 이용해서, Simulink/Stateflow의 전이에 대한 RRT 노드의 조건 유사도를 정의 6과 같이 정의 한다.

$CondSim_{S,p}(nd)$은 집합 $LACs,p$의 총 원소 개수 대비, $LACs,p$의 원소 중 참이 되는 단위 조건들의 비율을 나타낸다. 이 때 단위 조건이 참이 되는지 확인하기 위한 내부 변수와, 출력 변수의 값은 RRT 노드 nd의 해당 변수의 값을 이용한다. 조건 유사도를 구할 때, 입력 변수가 포함되어 있는 단위 조건을 제외하는 이유는 조건에 사용된 내부 변수와, 출력 변수의 값이 정해져 있더라도, 입력 변수의 값을 제어하여 단위 조건을 참으로 만들 수 있기 때문이다. 따라서 RRT 노드에 따라 조건 값이 바뀔 수 없는 단위 조건에 대해서만 조건 유사도를 평가한다.

> 정의 6. Simulink/Stateflow의 한 상태 S에서 나아가는 우선 순위 p의 전이에 대해서 RRT 노드 nd가 갖는 조건 유사도 $CondSim_{S,p}(nd)$는 식과 같다.
>
> $$CondSim_{S,p}(nd) = \frac{The\ \#\ of\ 'true'\ elements\ of\ the\ set\ LAC_{s,p}}{The\ \#\ of elements\ of\ the\ set\ LAC_{s,p}}$$

RRT 확장을 시도하기 위해 필요한 입력은 Table 2의 경우에 따라 다르게 된다. 먼저 경우 a)는 이미 SMT 엔진에 의해 입력이 결정된 상태이므로 고려하지 않는다. 경우 b)에서는 목표 전이의 조건식에 포함된 입력 변수에 대해서는 SMT 엔진을 이용하여 값을 결정 할 수 있다. 값이 결정된 변수는 그 값을 그대로 RRT 확장을 이용한 입력에 사용하며, 값이 결정되지 않은 변수의 값은 무작위로 결정된다. 경우 c), d)의 경우에는 모든 입력 변수의 값을 무작위로 결정한다.

RRT 확장의 시도는 L번 이루어지므로 확장에 필요한 입력 값들의 결정도 L번 이루어지며, 확장의 결과로 새로 생성된 RRT 노드도 L개가 된다. 이 L개의 RRT 노드 중 조건 유사도 $CondSim_{S,p}()$의 값이 가장 큰 RRT 노드의 확장에 사용된 입력 값이 RRT의 확장을 위한 최종 입력 값으로 선택이 된다. 만약 $CondSim_{S,p}()$의 값이 같은 노드들이 존재할 경우 Table 1과, 정의 3에 따라 RRT 무작위 노드 nd_{rand}에서 가장 가까운 RRT 노드로 확장 할 수 있는 입력이 선택된다. 그러나 예외적으로, L번 동안 입력 값을 결정하여, RRT 노드를 생성 중에 TG에 속한 전이가 발생이 되었으면, 해당 전이를 발생 시킨 입력 값이 RRT 확장에 필요한 최종 입력 값으로 결정된다.

3.4 입력 분석 기법이 적용된 RRT 확장 알고리즘

Table 3은 Table 1의 RRT 알고리즘 중 RRT 확장을 위한 입력을 선택하는 *select_input()*을 제안하는 RRT확장 기법에 따라 변형한 알고리즘이다. *SelectTransition()*에서는 RRT 노드 nd_{near}에서 발생시킬 수 있는 전이 중 TG에 포함되어 있는 전이를 선택한다. 전이가 선택되면 *Parsing Condition()*에서 전이 조건 식을 구문 분석한다. 구문 분석 결과와, Table 2의 어떤 경우에 해당하는지에 따라, 경우 a)일 때에는 *SolveSAT()*에서 조건식을 풀어, 결정된 입력을 반환한다. 입력을 결정하지 못하였을 경우 대상 전이 *Trans*는 유효하지 않으므로 TG에서 제외한다. 입력이 결정되었을 경우, *FillRemainInputs()*에서 값이 결정되지 않은 입력에 대해서, 무작위로 값을 결정한다. 경우 b)일 때는 *SolveSAT()*에서 조건 식에 포함된 입력에 대해서, 값을 결정한 후, 이 입력 값을 고정시킨 상태에서 *Approach()*에서 조건 식을 만족하는 상태로 가까워지기 위한 입력을 L번 무

작위로 시도하여 나머지 입력 값을 결정한다. 경우 c), d)에서는 SAT엔진을 이용하여 입력을 결정할 수 없기 때문에, 바로 *Approach()*를 이용하여, 조건을 만족할 수 있는 상태로 가까워지는 입력 값을 찾는다.

Table 4는 [15]의 기존 RRT 확장 알고리즘을 Table 3의 *mselect_input()*을 이용하여 변형한 RRT 확장 방법에 대한 알고리즘이다. *mselect_input()*에서 반환된 입력 값 v 는 새로운 RRT 노드를 확장하기 위한 입력으로 사용된다. 입력을 결정하지 못하였을 경우에는 RRT 확장을 수행하지 않고, 다시 RRT 무작위 노드를 생성하여 다른 방향으로 RRT가 확장되도록 한다.

*mselect_input()*에서 입력 값이 결정되면, *new_node()*에서 결정된 입력을 이용하여 Simulink/Stateflow 모델을 Δt 시간 동안 시뮬레이션 한다. 이 때, 모델이 시뮬레이션 된 결과, 생성된 RRT 노드 nd_{new} 와 OT(정의 4)가 반환된다.

Table 3. Modified select_input() function

```
mselect_input (nd_rand, nd_near, L, Δt, TG)
v= Φ;
v_sub= Φ;
Trans=SelectTransition(nd_near, , TG);
(ParseResult,Case) = ParseCondition(Trans);
If(case == 'a')
      v_sub = SolveSAT(ParseResult);
      if(v_sub == Φ) TG = TG - Trans
      v = FillRemainInputs(v_sub);
Else If(case == 'b')
      v_sub = SolveSAT(ParseResult);
      v = Approach(L, nd_rand, v_sub, Parsing_Result, Δt,
          TG);
Else If(case == 'c' || case == 'd')
      v = Approach(L, nd_rand, v_sub, Parsing_Result, Δt,
          TG);
return v;
```

Table 4. Modified RRT extension algorithm from [15] for test case generation

```
ModfiedExtends (nd_init, K, L, Δt, TG)
T.init(nd_init)
for k=1 to K do
  if(TG == Φ) return T;
  nd_rand ← random_node();
  nd_near ← nearest_ node(nd_rand, T);
  v ← mselect_input(nd_rand, nd_near);
  if(v == Φ) continue;
  (nd_new,OT)← new_ node(nd_near, v, Δt);
  T.add_vertex(nd_new); //Add nearest node
  T.add_edge(nd_near, nd_new, v);
  for Trans in OT  //Newly occurred transitions
    TG = TG - Trans
return T
```

*OT*의 전이 중 테스트 목표 *TG*(정의 5)에 포함된 전이는 달성된 전이므로 *TG*에서 제거된다.

3.5 예제

Table 4에서 제안한 RRT 확장 알고리즘 *Modified Extends*를 이용하여 Fig. 1의 예제 모델에 대한 테스트 케이스를 생성하는 과정을 설명한다. 예제 모델은 10ms의 실행 주기를 가지므로, 알고리즘의 파라메터 Δt는 10ms가 된다. RRT의 확장 횟수인 K와 RRT 확장을 위한 입력 값을 찾는 횟수인 L은 클수록 테스트 케이스를 정교하게 찾을 수 있다. 예제에서는 K와 L은 2로 설정한다.

시스템의 초기 상태를 표현하는 RRT 노드인 nd_{init}은 모든 변수의 초기 값으로 0을 가지며, 모델의 초기 상태는 *"FanOff"*이므로, 상태 활성화 변수 $a_{\text{"FanOff"}}$는 1이 되며, $a_{\text{"FanLo"}}$와 $a_{\text{"FanHi"}}$는 0이 된다. 즉 nd_{init}은 다음과 같은 값을 가진다.

nd_{init}={*"TempSensor"*=0, *"Mode"*=OFF, *"LoTimer"*=0, *"HiTimer"*=0, *"MotorAim"*= 0, $a_{\text{"FanOff"}}$= 1, $a_{\text{"FanLo"}}$= 0, $a_{\text{"FanHi"}}$=0}

테스트 목표 집합 *TG*는 상태 *"FanOff"*와 *"FanLo"* 사이의 전이 집합으로 한다. 즉, *TG*는 다음과 같은 원소로 구성된다.

TG = {상태 "FanOff"에서 나가는 우선순위 1의 전이,
상태 "FanLo"에서 나가는 우선순위 1의 전이,
상태 "FanLo"에서 나가는 우선순위 3의 전이}

지금부터 *ModifiedExtends* 알고리즘의 실행 흐름에 따라 알고리즘의 동작을 설명한다. 먼저 시스템의 초기 상태 RRT 노드 nd_{init}를 루트 노드로 RRT를 초기화 한다 (*T.init*(nd_{init})). 그리고 알고리즘의 파라메터 K 만큼 RRT 확장을 반복한다(*for k=1 to K do*). K가 2로 설정되었으므로 두 번의 RRT 확장이 이루어진다.

1) 첫 번째 RRT 확장

RRT 확장을 위한 첫 단계로 RRT 무작위 노드 nd_{rand}를 생성한다(nd_{rand}←*random_node()*). 그 후 RRT의 노드들 중 nd_{rand}와 가장 가까운 노드를 찾는다(nd_{near}←*nearest_node* (nd_{rand}, *T*)). 현재 RRT에는 nd_{init}만이 존재하므로 nd_{init}이 nd_{near}가 된다. 이제 nd_{near}에서 RRT를 확장하기 위한 입력 변수의 값을 결정하기 위한 Table 3의 알고리즘이 실행된다 (*mselect_input()*).

a) 만족하기 위한 대상 전이 결정

입력 변수의 값을 결정하기 위해 먼저 만족하기 위한 대상 전이를 결정한다(*Trans=SelectTransition*(nd_{near}, *TG*)). 현재 nd_{near}인 nd_{init}에서 발생할 수 있는 전이는 상태

*"FanOff"*가 활성화 된 상태이므로, *"FanOff"*로부터 나가는 전이 중 *TG*에 포함된, *"상태 "FanOff"에서 나아가는 우선순위 1의 전이"*가 목표 전이로 선택된다.

b) 선택된 전이의 구문 분석

전 단계에서 선택된 목표 전이 *"상태 "FanOff"에서 나아가는 우선순위 1의 전이"*의 조건식 *"[Mode==LO && TempSensor>LOTEMP]"*를 구문 분석 한다(*(ParseResult, Case)=ParseCondition(Trans)*). 분석 결과, 조건 식에 사용된 변수 *"Mode", "TempSensor"*는 모두 입력 변수이므로, Table 2의 경우 a에 해당한다(*If(case == 'a')*).

c) 입력 분석 기법 결정 및 입력 값 결정

경우 a에서는 SMT 엔진을 이용하여 목표 전이의 조건식을 풀 수 있다(*v_sub=SolveSAT(ParseResult)*). 조건식을 풀면, 입력 값으로 *v_sub={Mode="LO", TempSensor=21}*을 얻을 수 있다. 모델의 두 개의 입력 변수에 대해 모든 값이 결정되었으므로, FillRemainInputs()에서 추가로 결정해야 할 입력 값이 없고 v_sub가 그대로 RRT 확장의 입력 값으로 사용된다.

*mselect_input()*에서 결정된 입력 값 *v*를 이용하여 이를 이용하여, RRT 확장을 수행한다(*(nd_{new},OT)← new_node(nd_{near}, v, Δt)*). *new_node()*에서 모델을 시뮬레이션 하면, 모델의 상태 *"FanOff"*가 활성화 된 상태에서 목표 전이 *"상태 "FanOff"에서 나아가는 우선순위 1의 전이"*의 조건식 *"[Mode==LO && TempSensor>LOTEMP]"*이 참이 되어 전이가 발생 한다. 발생한 전이의 집합은 *new_node()*에서 *OT*로 반환된다. 시뮬레이션의 결과로 생성된 RRT 노드를 nd1이라고 하자. 따라서 nd_1은 다음과 같은 값을 가진다.

nd_1={*"TempSensor"=21, "Mode"="LO", "LoTimer"=0, "HiTimer"=0, "MotorAim"= 0.75, a"FanOff"= 0,* $a_{"FanLo"}$*= 1,* $a_{"FanHi"}$*=0}*

새 RRT 노드와 확장에 사용된 입력 값은 RRT에 추가된다(*T.add_vertex(nd_{new}); T.add_edge(nd_{near}, nd_{new}, v)*). 마지막으로 모델 시뮬레이션 중 발생한 전이들 중 테스트 목표 집합 *TG*에 포함된 전이는 테스트 목표에 해당하는 테스트 케이스를 찾은 것이므로, 목표 전이 *"상태 "FanOff"에서 나아가는 우선순위 1의 전이"*가 *TG*에서 제거되면서 RRT의 첫 번째 확장이 끝난다(*for Trans in OT TG = TG - Trans*).

2) 두 번째 RRT 확장

두 번째 RRT 확장의 시작으로 새로운 RRT 무작위 노드가 생성된다(*nd_{rand}←random_node()*). 그리고 nd_{rand}로부터 가장 가까운 노드를 찾는다(*nd_{near}←nearest_node(nd_{rand}, T)*). 첫 번째 확장과는 다르게 RRT에는 nd_{init}과 nd_1 두 개의 RRT 노드가 존재하기 때문에 두 노드 중 nd_{rand}에서 가까운 노드를 결정해야 한다. nd_{rand}가 다음과 같은 값을 가진다고 하자.

nd_{rand}={*"TempSensor"=20, "Mode"="OFF", "LoTimer"=40, "HiTimer"=60, "MotorAim"= 0.44,* $a_{"FanOff"}$*= 0,* $a_{"FanLo"}$*= 1,* $a_{"FanHi"}$*=1}*

먼저 nd_{rand}와 nd_{init}의 거리 *dv*(nd_{rand}, nd_{init})를 정의 3의 식에 따라 계산을 하면 (|20-0|/100 + 1 + |40-0/200| + |60-0|/200 + |0.44-0|/1 + 1 + 1 + 1)/8=0.52가 된다. 그리고 nd_{rand}와 nd_1의 거리 *dv*(nd_{rand}, nd_1)을 계산 하면(|20-21|/100 + 1 + |40-0/200| + |60-0|/200 + |0.44-0.75|/1 + 0 + 0 + 1)/8=0.35가 된다. 따라서 nd_{rand}에서 가까운 노드는 nd_1이 된다. 이제 nd_1에서 RRT를 확장하기 위한 입력을 결정한다.

a) 만족하기 위한 대상 전이 결정

*TG*에 속하는 전이 중 nd_1에서 나갈 수 있는 전이는 상태 *"FanLo"*에서 나가는 우선순위 1과 3 전이 두 개가 존재한다. 두 전이 중 우선순위 3 전이가 목표 전이로 선택되었다고 하자.

b) 선택된 전이의 구문 분석

목표 전이의 조건식 *"[LoTimer>LOTIME]"*을 분석하면, 조건식에 사용된 *"LoTimer"*는 내부 변수이므로, Table 2의 경우 c에 해당한다(Else If(case=='c' || case=='d').

c) 입력 분석 기법 결정 및 입력 값 결정

경우 c에서는 SMT 엔진을 이용하여 입력 변수의 값을 결정 할 수 없으므로, 목표 전이의 조건식을 만족하는 상태로 접근하는 기법을 사용한다(*v = Approach(L,* nd_{rand}*, v_sub, Parsing_Result, Δt, TG*).

*L*의 값이 2이므로, 조건식을 만족하는 상태로 접근하기 위해서, *Approach()*에서 두 번의 입력을 생성하여, 모델을 시뮬레이션 한다. 첫 번째 입력으로 *{Mode="LO", Temp Sensor=25}*가 생성이 되었다고 하자. 이 입력으로 모델을 시뮬레이션 하면, *"FanLo"*상태에서 나가는 우선순위 4의 전이가 발생하여 *"LoTimer"*의 값이 1 증가하며, 모델의 상태는 *"FanLo"*에 머물러 있게 된다. 이 결과로 생성된 RRT 노드를 nd_2라 하자. nd_2의 값은 다음과 같게 된다.

nd_2={*"TempSensor"=25, "Mode"="LO", "LoTimer"=1, "HiTimer"=0, "MotorAim"= 0.75,* $a_{"FanOff"}$*= 0,* $a_{"FanLo"}$*= 1,* $a_{"FanHi"}$*=0}*

두 번째 입력 생성 시도로 *{Mode="LO", TempSensor= 20}*이 입력으로 결정되었다고 하자. 이 입력으로 모델을 시뮬레이션 하면 *"FanLo"*에서 나가는 우선 순위 1의 전이가 발생하여, 모델의 상태는 *"FanOff"*로 이동하게 된다. 이 결과로 생성된 RRT 노드를 nd_3이라 하자. nd_3의 값은 다음과 같게 된다.

Table 5. Test case samples

RRT node path	Test cases	Achieved Transition
$nd_{init} \rightarrow nd_1$	{Mode= "LO", TempSensor=21}	Transition with priority 1 from state "FanOff"
$nd_{init} \rightarrow nd_1 \rightarrow nd_3$	{Mode= "LO", TempSensor=21} {Mode= "LO", TempSensor=20}	Transition with priority 1 from state "FanLo"

nd_3={"TempSensor"=20, "Mode"="LO", "LoTimer"=0, "HiTimer"=0, "MotorAim"= 0, $a_{\text{"FanOff"}}$= 1, $a_{\text{"FanLo"}}$= 0, $a_{\text{"FanHi"}}$=0}

이제 두 차례의 시도로 생성된 입력 중, 어느 입력을 RRT의 확장에 사용할 지 결정해야 한다. 이 때 RRT 노드 nd_3을 생성한 입력이 최종 입력으로 선택되는데, 그 이유는 RRT 노드 nd_3이 생성되면서, TG의 원소 중 하나인, "상태 "FanLo"에서 나가는 우선순위 1의 전이"가 발생 하였기 때문이다.

*mselect_input()*에서 결정된 입력 값 v를 이용하여 이를 이용하여, *new_node()*에서 모델을 시뮬레이션 하면, RRT 노드 nd_3이 생성되고, 발생한 전이 집합 OT에 "상태 "FanLo"에서 나가는 우선순위 1의 전이"가 포함된다. 생성된 노드와 입력 값은 RRT에 추가되며, 발생한 전이는 테스트 목표 집합 TG에서 제거된다.

예제를 통해 생성된 테스트 케이스는 Table 5와 같다.

4. 실 험

제시된 알고리즘을 평가하기 위해서 '실제 자동차에 사용되는 차선 이탈 경보 장치 제어 모듈, 에어 서스펜션 제어 모듈, 운전석 도어 램프 제어 모듈, ECU 모델에 대한 테스트 케이스 생성 실험을 수행한다. 실험은 논문에서 제안된 알고리즘과 [15]의 알고리즘을 이용하여 테스트 케이스를 생성한 결과를 테스트 목표 달성율을 비교한다.

테스트 목표 달성율은 정의 7과 같다.

정의 7. 테스트 케이스 생성 알고리즘의 테스트 목표 TG에 대한 테스트 목표 달성율 GoalRatio는 다음과 같이 정의된다.

$$GoalRatio(TG) = \frac{The\ \#\ of\ transition\ occured\ in\ TG}{The\ \#\ of\ transition\ in\ TG}$$

알고리즘의 RRT 확장 횟수 파라메터 K는 100, 새로운 RRT 노드 확장을 위해 입력을 생성하는 횟수인 파라메터 L은 5로 설정하였다. 확장 횟수를 100 초과로 하였을 때 증가하는 테스트 목표 달성율이 미미하였으므로, 100회의 확장 횟수는 테스트 케이스를 생성하기 위한 충분한 횟수라 할 수 있다. L을 5로 설정한 것은 실험 대상 모델에 사용되는 전이의 조건이 두 개의 불리언 변수로 표현되는 경우가 많기 때문이다. 두 불리언 변수의 값의 조합은 네 가지 경우가 있으며, 무작위로 변수 값 조합을 생성해서 목표 전이의 조건이 참이 되도록 근접하기 위해서 입력 생성 시도를 5회로 하여 충분한 입력 조합을 생성하도록 한다. K와 L의 크기가 커질수록 테스트 목표 달성율이 증가할 확률은 높아지지만 그만큼 테스트 케이스 생성 시간은 증가하게 된다. 실험 대상 컴퓨터의 CPU는 Intel Core i5 2.67GHz, RAM은 4.00GB이다.

실험에 사용 되는 세 ECU의 모델은 실제 자동차의 개발에 쓰이는 상업적인 모델로 그 복잡도가 높다. 먼저 차선 이탈 경보 장치 제어 모듈(Lane Departure Warning Systems Module)은 시동 여부와, 도로 감지 센서, 차량의 상태 정보 등의 6개의 입력을 받아서 차선 이탈 여부를 분석 한 다음 그 결과를 차량의 클러스터로 출력하는 모듈이다. 차량의 운전 여부, 차선 이탈 여부, 시스템 동작 오류 등의 시스템 상태를 Stateflow의 13가지 상태로 표현하여 모델링 하고 있다. 에어 서스펜션 제어 모듈(Air Suspension Control Module)은 차량의 시동 상태, 차축 상태, 높이 등의 21가지 입력을 이용하여 차량의 수평을 유지하기 위한 차축 제어, 유압 제어, 표시등 제어 등을 수행한다. 모델은 크게 전원 비인가 상태와 전원 인가 상태로 나뉘어지며, 전원 인가 상태에서는 각각 원격 제어, 상승축 제어, 수평 제어, 유압 제어, 높이 제어, 유압 벨브 제어, 디스플레이 제어 등의 기능이 상태로 모델링 되어 있으며 각 기능 상태는 그에 맞는 세부 상태를 가지게 된다. 마지막으로 운전석 도어 램프 제어 모듈(Drive Door Lamp Control Module)은 시동과, 도어 닫힘, 잠금 상태, 램프 스위치의 네 가지 입력을 이용하여 램프의 꺼짐과 켜짐을 제어한다. 이를 위하여, 램프의 동작을 제어하는 초기화, 꺼짐, 켜짐, 보류 등의 상태가 Stateflow로 모델링 되어있으며, 내부적으로 램프 점등의 시간을 조절하기 위한 타이머 제어 상태 등 총 20개의 상태로 모델링이 되어 있다.

Table 6은 위에서 설명한 세 가지 Simulink/Stateflow 모델의 통계 정보이다.

에어 서스펜션 제어 모듈 ECU의 모델이 가장 규모가 크며, 차선 이탈 경보 장치 제어 모듈, 운전석 도어 제어 모듈

Table 6. Statistics of example models

Model	States	Transitions	Variables
Lane Departure Warning Systems Module	13	205	46
Air Suspension Control Module	83	272	150
Drive Door Lamp Control Module	20	55	59

순으로 모델의 규모가 작아진다. 일반적으로 모델의 규모가 클 수록 모델의 복잡도가 증가하기 때문에, 테스트 케이스 생성 시 테스트 커버리지가 떨어지게 된다.

테스트 케이스 생성 시 알고리즘의 테스트 목표 집합 *TG*는 테스트 케이스 생성 대상 Simulink/Stateflow 모델의 모든 전이의 집합이 된다. Table 7은 기존 알고리즘과 제안된 알고리즘으로 테스트 케이스 생성을 수행한 실험 결과이다.

Table 7. Comparison of transition coverage

Model		Goal ratio
Lane Departure Warning Systems Module	Typical	47.95%
	Proposed	56.16%
Air Suspension Control Module	Typical	69.60%
	Proposed	84.00%
Drive Door Lamp Control Module	Typical	63.33%
	Proposed	86.67%

실험 대상 모델 별로 기존 알고리즘과, 제안된 알고리즘의 전이 기반 테스트 커버리지 (transition coverage)를 살펴보면, 기존 알고리즘에 비해 제안된 알고리즘의 커버리지가 더 높은 것을 확인할 수 있다.

기존의 알고리즘에서 만족하지 못한 전이 목표 중 제안된 알고리즘에서 만족한 전이 목표들을 조건 식 유형(Table 2)에 맞춰 실험 대상 모델 별로 분석해 보면 Table 8과 같다. 경우 d인 비선형적인 조건 식을 가지고 있는 경우는 없으며, 조건식이 입력 변수만으로 구성되어 있는 경우 a의 전이 목표가 가장 많이 추가적으로 달성되었다. 내부 변수만으로 조건식이 구성되어 있는 경우 c가 다음으로 많은 비율을 차지하였으며, 조건 식에 입력 변수와 내부 변수가 혼용된 경우 b가 가장 낮은 비율이다. 경우 a는 제안한 알고리즘에서는 SMT 엔진을 이용하면 목표 전이를 만족시킬 수 있는 입력 변수 값을 바로 생성 할 수 있기 때문에 입력 값을 목표 전이의 조건 식에 맞춰 생성하지 않는 기존 알고리즘 보다 추가적으로 달성한 비율이 높다고 할 수 있다. 경우 b)에서 추가적으로 달성된 전이 조건 식들은 입력 변수로 이루어진 단일 조건과 내부 변수로 이루어진 조건들의 OR 연산자로 결합된 경우가 대부분이다. 이 경우 제안한 알고리즘에서는 SMT 엔진을 이용하여 입력 변수의 조건을 만족하기 위한 값을 찾고, 나머지 조건에 대해서는 무작위로 접근하는 방법을 취하기 때문에, OR 연산자로 결합된 조건 식 중 입력 변수에 관련된 조건이 만족되어 전체 조건 식이 참이 되어 전이 목표를 달성하는 결과를 가져온다. 경우 c에 대한 전이 목표의 조건 식을 살펴보면 두 개 이상의 내부 변수들의 조건 식으로 이루어진 경우가 많으며, 이는 제안한 알고리즘에서 이용하는 조건 유사도가 높은 RRT 노드를 우선 선택하여 확장하는 전략이 유효하다는 것을 보인다.

Simulink/Stateflow 모델의 전이는 모델 작성자가 모델 대상 ECU의 기능을 표현하는 의도로 삽입한 것이기 때문에, 대상 ECU의 기능을 모두 테스트 하기 위해서는 반드시

Table 8. Case analysis for additionally covered transitions by proposed algorithm

Model		Percentages
Lane Departure Warning Systems Module	Case a)	50%
	Case b)	0%
	Case c)	50%
	Case d)	0%
Air Suspension Control Module	Case a)	47.83%
	Case b)	4.34%
	Case c)	47.83%
	Case d)	0%
Drive Door Lamp Control Module	Case a)	85.71%
	Case b)	14.29%
	Case c)	0%
	Case d)	0%

전이 목표들이 모두 발생하는 테스트 케이스가 확보되어야 한다. 그리고 모델에 오류가 없는 한, 모델에 존재하는 전이는 발생 가능한 전이이기 때문에 해당 전이를 발생 시키는 테스트 케이스는 반드시 존재한다.

그러나 제안한 알고리즘의 실험 결과 커버리지가 100%가 되지 않는다. 그 이유는 세 가지가 있다. 첫 번째로, 앞에서 설명하였듯이, Simulink/Stateflow 모델의 기술 오류로 인하여, 전이 조건식을 절대로 만족 시킬 수 없는 경우가 있다. 예를 들면 우선 순위 1의 전이 조건식이 [A>100]이었는데, 우선 순위 2의 전이 조건식이 [A>200]이라면, 우선 순위 2의 전이 조건이 참이 되면 자동으로 우선 순위 1의 전이 조건도 참이 되어 우선 순위 1의 전이가 먼저 발생하게 되므로, 우선 순위 2의 조건은 절대 발생 할 수 없게 된다.

두 번째로 테스트 목표 달성율이 100%가 되지 않는 이유는,, 자동차 ECU의 특성 상 시간과 관련된 기능들이 많이 들어 가기 때문이다. Fig. 2의 모델이 시간과 관련된 기능의 예제이다.

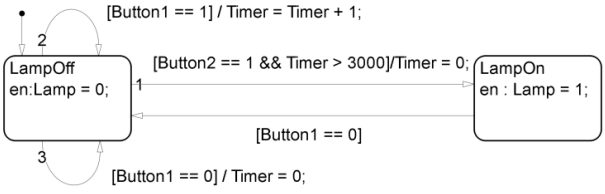

Fig. 2. An example model with timer logic

예제 모델에서는 Button1이 1이 되는 동안 Timer의 값이 증가한다. 모델의 상태 LampOff에서 LampOn으로 이동하는 전이가 발생하기 위해서는 Button1이 1이 된 상태에서 모델이 3000번 초과 실행 된 후 Button2가 1이 되어야 한다. 이를 위해서는 RRT 확장 시 Timer가 증가 하여 새롭게 생성된 RRT 노드를 다음 RRT 확장의 기점으로 선택해야 되는 과정이 Timer 값이 3000이 넘을 때까지 발생해야 하며, 이

동안에, Button1이 0이 되지 말아야 한다. Button1이 되면 Timer의 값이 0이 되기 때문이다. 무작위로 모델의 입력과, RRT 확장 노드를 결정하는 RRT의 특성 상 이 과정이 이루어지려면 매우 많은 RRT 확장 횟수가 필요하게 된다. 이 문제를 다루기 위해서는 제안한 알고리즘과 더불어, 다른 모델 분석 기법이 필요하다.

세 번째 이유는 RRT의 특성상, 무작위 기반의 테스트 케이스 생성 기법이기 때문에, RRT 확장을 위해 선택되는 노드나, 입력 생성에 따라, 달성 할 수 있는 테스트 목표가 바뀌기 때문이다. 물론 제안한 알고리즘에서는, 테스트 목표를 분석하여, 입력 값을 결정할 수 있는 경우에는 입력이 SMT 엔진으로부터 결정이 되지만, 입력 값을 결정할 수 없는 경우에는 기존 RRT와 마찬가지로 동작하게 된다. RRT와 같은 무작위 기반의 테스트 케이스 생성 기법은 다양한 특성의 모델에 범용적으로 적용할 수 있는 귀납적 방법이지만, 제안한 알고리즘과 같이 모델의 분석이 가능하여, 테스트 목표를 달성 할 수 있는 테스트 케이스를 연역적으로 찾아낼 수 있는 방법과 같이 사용되면 더 효율적이다. 그러므로 테스트 케이스 생성을 위한 다양한 모델 분석 기법을 추가로 연구하여, 테스트 케이스 생성 기법의 성능을 높일 필요가 있다.

Table 6의 차선 이탈 경보 장치 제어 모듈 ECU 모델의 테스트 목표 달성율은 다른 두 모델의 달성율보다 떨어지는데 그 이유는 차선 이탈 경보 장치 제어 모듈 ECU 모델의 요구사항이 외부 파라메터에 따라 바뀌어 동작하여 유효하지 않은 목표가 많이 생성되기 때문이다. 그 예를 들면 다음과 같다.

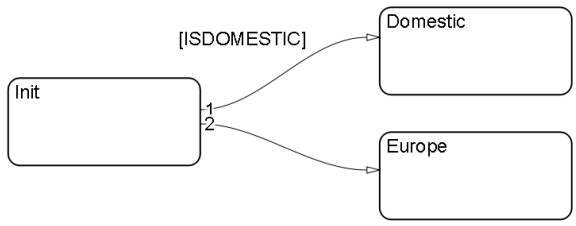

Fig. 3. An example model with an infeasible transition

Fig. 3은 모델의 *"ISDOMESTIC"* 파라메터에 따라 내수 사양의 기능이 동작하거나 유럽 사양의 기능이 동작하는 예이다 *"ISDOMESTIC"*은 모델을 시뮬레이션 하기 전 값이 결정이 되어 모델이 실행 중에는 값이 변하지 않기 때문에 *"ISDOMESTIC"*이 1로 결정된 상황에서는 *"Init"*에서 *"Europe"*으로의 전이와 *"Europe"* 상태에 대한 테스트 목표는 모두 유효하지 않게 된다.

또 다른 이유는 입력 신호의 에러 처리 기능 때문이다. 차선 이탈 경보 장치 제어 ECU 모델의 경우에는 오류에 해당하는 값이 입력될 경우에는 모든 기능을 멈추고 오류 처리 기능으로 모델의 상태가 이동하기 때문에 오류에 해당하지 않은 값을 입력하는 것이 중요하다. 그러나 테스트 케이스 생성 알고리즘에서 입력을 선택 할 때 입력의 오류 상황 같은 것을 고려하지 않은 상태에서 RRT 무작위 노드에 가까워지는 입력 만을 생성하기 때문에 RRT의 확장 방향이 오류 상태가 활성화 되는 쪽으로 치우칠 확률이 높다. 입력의 어떠한 값이 오류에 해당하는지는 모델의 분석만을 통해서 판단하기에 매우 어려운 문제이기 때문에 이러한 특성을 가진 모델의 테스트 케이스 생성은 그 달성율이 떨어 질 수 밖에 없다.

5. 결 론

본 논문에서는 Simulink/Stateflow 모델 기반의 테스트 케이스 생성 시 중요한 문제로 여겨지는 도달 가능성 문제를 해결 하기 위해 유용하게 사용 할 수 있는 RRT 알고리즘의 휴리스틱 기반 입력 생성 방식을 제안하고 있다.

휴리스틱 기반 입력 생성 방식은, 먼저 테스트 시 방문하고자 하는 목표 전이를 정한다. 그리고 목표 전이의 조건식을 구문 분석 한다. 그리고, 조건식에 사용된 변수의 종류를 분석하여, 각 경우마다 행해야 할 입력 결정 방식을 선택한다.

조건식에 사용된 변수가 모두 입력 변수일 경우에는 SMT 엔진를 이용하여 입력을 결정하며, 일부는 입력 변수, 일부는 내부 변수나 출력변수일 경우, SMT 엔진를 이용하여 입력 변수의 값을 구하고, 나머지 입력은 무작위로 생성함으로써, 조건식을 만족시킬 수 있는 상태로 접근하게 한다. 또한 조건식에 사용된 모든 변수가 내부 변수나, 출력변수라면, 모든 입력 변수를 무작위로 생성하여, 조건식을 만족시킬 수 있는 상태로 접근하는 입력을 선택한다.

제안된 알고리즘은 실제 산업체에서 사용되는 ECU의 Simulink/Stateflow 모델을 대상으로 그 성능이 평가되었으며, 기존 RRT를 사용하여 테스트 케이스를 생성한 결과보다, 향상된 성능을 보였다.

참 고 문 헌

[1] J. Zander, I. Schieferdecker, P. J. Mosterman, "Model-based Testing for Embedded systems", Vol.13, CRC Prss, 2012.

[2] K. Forsberg, H. Mooz, "The Relationship of System Engineering to the Project Cycle", in Proceedings of the First Annual Symposium of National Council on System Engineering, pp.57-65, 1991.

[3] MATLAB Simulink Stateflow http://www.mathworks.com/products/stateflow/ 1994-2013 The MathWorks, Inc.

[4] R. Alur, "Model checking of hierarchical state machines" in Proceedings of the 6th ACM SIGSOFT FSE, Vol.23, Issue 6, pp.175-188, 1998.

[5] T. A. Henzinger, P. W. Kopke, A. Puri, P. Varaiya, "What's Decidable About Hybrid Automata?", Journal of Computer and System Sciences Vol.57, Issue 1, pp.94-124, 1998.

[6] T. Brihaye, "A note on the undecidability of the reachability

problem for o-minimal dynamical systems" in Mathematical Logic Quarterly, Vol.52, Issue 2, pp.165-170, 2006.
[7] J.M. Esposito, J. Kim, V. Kumar "A Probabilistic Approach to Automated Test Case Generation for Hybrid Systems" in Hybrid Systems: Computation and Control, 2004.
[8] Reactis http://www.reactive-systems.com/products.msp Reactive Systems, Inc
[9] Design Verifier http://www.mathworks.com/products/sldeignverifier/ Mathworks, Inc.
[10] H. S. Hong, I. S. Lee, O. Sokolsky, S. D. Cha, "Automatic Test Generation From Statecharts Using Model Checking" in Technical Report, Department of Computer & Information Science University of Pennsylvania MS-CIS-01-07, 2001.
[11] M. Satpathy, A. Yeolekar, S. Ramesh, "Randomized Directed Testing(REDIRECT) for Simulink/Stateflow Models" in Proceedings of the 8th ACM international conference on Embedded software, pp.217-226, 2008.
[12] T.A. Henzinger, "The Theory of Hybrid Automata" in Proceedings of Logic in Computer Science, Eleventh Annual IEEE Symposium, pp.278-292, 1996.
[13] K. L. McMillan, "Symbolic Model Checking - an Approach to the State Explosion Problem" Kluwer Academic Publishers, 1993.
[14] L. de Moura, "SMT Solvers", 2006.
[15] S. M. LaValle, "Rapidly-Exploring Random Trees: A New Tool for Path Planning" TR 98-11 Computer Science Dept., Iowa State Univ. <http://janowiec.cs.iastate.edu/papers/rrt.ps>, 1998.
[16] J. M. Esposito, "Randomized Test Case Generation for Hybrid Systems: metric selection" in System Theory, Proceedings of the Thirty-Sixth Southeastern Symposium, pp.236-240, 2004.
[17] W., C, Barret, R. Sebastiani, S. A. Seshia, C. Tinelli, "Satisfiability Modulo Theories", Handbook of satisfiability, pp.825-885, 2009.
[18] D. R. Cok, A. Griggio, R. Bruttomesso, "The 2012 SMT Competition", 2012.
[19] J. C. Gower "A general coefficient of similarity and some of its properties" in Biometrics, Vol.27, pp.857-871, 1971.
[20] T. Mason, D. Brown, "Lex & yacc", O'Reilly, 1992.
[21] L. de Moura, N. Bjørner, "Z3: An efficient SMT solver.", Tools and Algorithms for the Construction and Analysis of Systems, pp.337-340, 2008.

박 현 상

e-mail : elvaimay@ajou.ac.kr
2005년 아주대학교 정보 및 컴퓨터 공학부 (공학사)
2007년 아주대학교 정보통신전문대학원 정보통신공학과(공학석사)
2007년~현 재 아주대학교 정보통신전문대학원 정보통신공학과 박사과정
관심분야 : 소프트웨어 공학, 임베디드 시스템, 운영 체제, 실시간 고속 네트워크 시스템 등

최 경 희

e-mail : khchoi@ajou.ac.kr
1976년 서울대학교 수학교육과(학사)
1979년 프랑스 그랑데꼴 Enseeiht 대학 (석사)
1982년 프랑스 Paul Sabatier 대학 정보공학부(박사)
1982년~현 재 아주대학교 정보통신전문대학원 교수
관심분야 : 운영체제, 분산시스템, 실시간/멀티미디어 시스템 등

정 기 현

e-mail : khchung@ajou.ac.kr
1984년 서강대학교 전자공학과(학사)
1988년 미국 Ilinois주립대 EECS(석사)
1990년 미국 Purdue대학 전기전자공학부 (박사)
1991년~1992년 현대반도체 연구소
1993년~현 재 아주대학교 전자공학부 교수
관심분야 : 컴퓨터구조, VLSI 설계, 멀티미디어/실시간 시스템 등