DOI: 10.3745/KIPSTD.2010.17D.5.337

# 선형 계획법을 이용한 Timing Diagram의 테스트 입력 시퀀스 자동 생성 전략

이 홍 석[†] · 정 기 현[††] · 최 경 희[†††]


## 요 약

Timing diagram은 시간에 따른 시스템의 행동을 표현하기 용이하고 표현된 행동을 쉽게 인식할 수 있는다는 장점 때문에 널리 사용되고 있다. Timing diagram으로 기술된 시스템을 테스트 하기 위해서는 여러 기술이 필요하다. 그 중의 하나는 테스트 케이스 목표들이 존재할 때, 시스템 모델이 원하는 상태에 도달하도록 하기 위해 입력 값들의 시퀀스를 생성하는 기술이다. 본 논문은 Timing diagram모델에 대한 테스트 케이스 목표로부터 테스트 입력 시퀀스를 자동으로 생성하는 방법을 제안한다. Timing diagram에서 테스트 입력 시퀀스를 자동으로 생성하기 위해서는 입력 waveform과 시간 제약으로 이루어진 시점의 전이 조건을 만족시키는 적절한 입력 집합을 필요로 한다. 이와 같은 문제를 해결하기 위해, 본 논문에서는 선형 계획법을 이용한 접근 방식을 택하였는데, 해결과정은 다음과 같다. 1) Timing diagram 모델을 입력으로 받아 이를 선형 계획 문제로 변형한다. 2)변형된 선형 계획 문제를 선형 문제 해결 도구를 사용하여 해결한다. 3) 선형 계획 문제의 해답으로부터 Timing diagram모델의 테스트 입력 시퀀스를 생성한다. 본 논문에서는 임의의 Timing diagram 모델에 대해 이를 선형 계획법으로 모델링 하는 방법을 형식적으로 기술하였고, 증명을 통해 본 논문의 접근 방법의 타당성을 보였으며, 또한 도구를 구현하여 Timing diagram 예제 모델로부터 테스트 입력 시퀀스를 생성함으로써 본 논문의 유용성을 입증하였다.


**키워드: Timing diagram, 테스트 입력 시퀀스 자동 생성, 선형 계획법**

# Test Input Sequence Generation Strategy for Timing Diagram using Linear Programming

Hongseok Lee[†] · Kihyun Chung[††] · Kyunghee Choi[†††]


## ABSTRACT

Timing diagram is popularly utilized for the reason of its advantages; it is convenient for timing diagram to describe behavior of system and it is simple for described behaviors to recognize it. Various techniques are needed to test systems described in timing diagram. One of them is a technique to derive the system into a certain condition under which a test case is effective. This paper proposes a technique to automatically generate the test input sequence to reach the condition for systems described in timing diagram. It requires a proper input set which satisfy transition condition restricted by input waveform and timing constraints to generate a test input sequence automatically. To solve the problem, this paper chooses an approach utilizing the linear programming, and solving procedure is as follows: 1) Get a Timing diagram model as an input, and transforms the timing diagram model into a linear programming problem. 2) Solve the linear programming problem using a linear programming tool. 3) Generate test input sequences of a timing diagram model from the solution of linear programming problem. This paper addresses the formal method to drive the linear programming model from a given timing diagram, shows the feasibility of our approach by prove it, and demonstrates the usability of our paper by showing that our implemented tool solves an example of a timing diagram model.


**Keywords: Timing Diagram, Test Input Sequence Generation, Linear Programming**

## 1. 서 론

Timing diagram은 시간에 따른 시스템의 행동을 표현하는 도구로 매우 유용하게 사용되고 있다. Timing diagram은 과거에는 주로 전자공학에서 주로 사용되어 왔으나 최근에는 소프트웨어 분야에서도 그 사용이 확대되고 있다. UML2는 Timing diagram을 시스템을 표현하기 위한 도구로 채택하였으며, 그 의미도 기존의 전자공학에서 사용하던 의미에서 소프트웨어의 행동을 표현하기 위한 의미가 추가되었다. 다른 대부분의 시스템을 표현하는 도구들이 다 그

† 준 회 원 : 아주대학교 전자공학과 박사과정
†† 정 회 원 : 아주대학교 전자공학부 교수
††† 정 회 원 : 아주대학교 정보통신전문대학원 교수
논문접수 : 2010년 4월 28일
수 정 일 : 1차 2010년 6월 14일
심사완료 : 2010년 7월 6일

렇듯이 Timing diagram도 모델로부터 테스트 케이스를 생성할 수 있으며, 테스트 케이스 목표에 도달하기 위한 테스트 입력 시퀀스를 생성할 수 있다. 하지만, Timing diagram에 대한 테스트 입력 시퀀스를 생성하고자 하는 연구는 좀처럼 찾아보기 힘들다. 그래서 본 연구에서는 임의의 Timing diagram 모델과 테스트 케이스 목적으로부터 테스트 입력 시퀀스를 생성하는 전략에 대해서 기술하고자 한다.

Timing diagram 모델의 테스트 케이스 입력 시퀀스를 생성하는데 있어서 고려해야 하는 문제는 Timing diagram 모델에 존재하는 시간제약을 만족시키면서 적절한 waveform 입력 값을 생성해야 한다는 점이다. 특히 여러 개의 시간 제약이 복잡하게 얽혀 있는 모델의 경우 적절한 시간과 적절한 waveform입력 값의 생성은 쉽지 않은 문제이다. 이와 같은 문제를 해결하기 위해서 본 연구에서는 Timing diagram 모델을 시간 제약이 존재하지 않는 모델과 존재하는 모델로 나누고 각각에 대해서 해결하는 방법을 택하였다. 시간 제약이 존재하지 않는 모델은 입력 waveform의 값들을 적절하게 생성해 주면 되므로, 이런 모델에 대한 테스트 입력 시퀀스 생성은 쉬운 문제이다. 시간 제약이 존재하는 Timing diagram모델에서도 역시 입력 waveform의 값을 원하는 대로 생성하는 것은 쉬운 문제이므로 테스트 입력 시퀀스를 생성하기 위해서 시간 제약이 차지하고 있는 구간 내에서 소비하는 시간에 대해서만 고려하여 문제를 해결하고자 했으며, 이를 선형 계획법으로 모형을 모델링 하여 해결하는 접근 방식을 택하였다.

선형 계획법[7, 8]은 최적화 문제의 일종으로 주어진 선형 조건들의 제약 조건을 만족시키면서 여러 결정 변수의 값을 변화시켜서 선형인 목적함수를 최적화하는 문제로 산업공학이나, 미시 경제학, 네트워크 경로 최적화 등 많은 분야에서 사용되는 학문이다. 이와 같이 시간 제약들이 차지하고 있는 구간 내의 소비해야 하는 시간에 대해 선형 모형으로 모델링 하여 그 해를 구함으로써 시간 제약을 만족시키면서 적절한 입력 waveform의 값들을 생성하는 방법을 찾을 수 있었다. 본 논문에서는 임의의 Timing diagram 모델에 대해 모델 내의 임의의 시점에 도달하기 위한 테스트 입력 시퀀스 테이블을 정의하였으며, Timing diagram 모델로부터 선형 계획 모형으로 모델링 하는 방법을 형식적으로 기술하였다. 선형 계획 모형의 계산은 엑셀의 해 찾기 기능을 사용하여 선형 계획 문제를 해결하였으며, 이를 임의의 예제 Timing diagram 모델에 적용함으로 본 연구의 접근 방법으로 해결할 수 있음을 보였다.

본 논문의 구성은 다음과 같다. 2장에서는 테스트 입력 시퀀스를 자동으로 생성하는 연구에 대해서 기술하였으며, 3장에서는 선형 계획법의 의미와 Timing diagram의 문법 및 의미에 대해서 기술하였다. 4장에서는 테스트 입력 시퀀스 테이블을 정의하고, 입력 waveform을 자동으로 생성하는 방법에 대해서 정의하였고, 임의의 Timing diagram 모델에 존재하는 시간 제약상의 문제를 선형 계획 모형으로 모델링 하는 방법에 대해서 기술하였다. 5장에서는 Timing diagram 예제 모델에 대한 테스트 입력 시퀀스를 생성하는 방법을 설명하였고, 6장에서는 결론과 향후 연구로 끝을 맺는다.

## 2. 관련 연구

테스트 입력 시퀀스를 생성하는 기존의 연구는 많은 연구가 있었는데, 생성 방법에 따라 랜덤 방식, 경로 기반 방식, 목표 기반 방식으로 나눌 수 있다. 랜덤 방식은 테스트 입력 값을 무작위로 생성하여 입력하는 방식이다. 이 방식은 비교적 간단하게 구현할 수 있는 장점이 있지만 입력 값이 무작위로 생성되므로 생성된 테스트 케이스가 모든 테스트 요구사항을 만족시키지 못할 수 있다는 문제가 있다. Reactis[5] 도구 및 오정섭[13]이 랜덤 방식으로 테스트 입력 값을 생성한다. 경로 기반 방식[11, 14]은 모델을 분석하여 모델이 행동할 수 있는 경로들의 집합을 추출하고 이 경로들이 수행될 수 있도록 입력 값을 생성하는 방식이다. Nayak[11]은 UML로 표현된 모델로부터 제어 흐름 기반의 정보를 추출하여 Structured Composite Graph를 생성하고, 그래프로부터 모델의 실현 가능한 실행 경로를 찾아 적절한 입력을 찾는 방식의 연구를 수행하였고, Lee[14]는 Modechart로 표현된 사양서를 기반으로 기호 실행 방식을 통해 기호 수행 트리를 생성하여 테스트 시퀀스를 생성하는 연구를 수행하였다. 경로 기반의 테스트 입력 시퀀스를 생성하는 방법은 모델로부터 모델의 행위 경로를 추출하는 과정에서 실현 가능하지 않은 경로가 추출될 수 있기 때문에 임의의 경로가 실현 가능한지의 여부를 결정해야 하는데, 이 문제가 쉽지 않다는 단점이 있다. 그리고 목표 기반 방식[1-4, 6, 9-10, 12-13, 15, 17]은 경로 기반 방식의 실현 가능하지 않은 경로를 찾는 단점을 해결하기 위해 제안된 방식으로 모델의 초기 상태로부터 도달하고자 하는 목표 상태를 추출하고, 초기 상태로부터 목표 상태에 도달하도록 하기 위한 테스트 입력 데이터를 생성하는 방식이다. 이를 위해 많은 연구들[3-4, 10, 12, 15, 17]에서는 모델 체킹 도구를 이용하여 테스트 케이스를 생성하는 접근 방식을 택하였다. 모델 체킹 도구는 원래 모델과 그 모델이 만족해야 하는 속성을 입력으로 받아 모델의 모든 경우에 대한 행동을 계산하여 모든 행동에서 속성을 만족하는지를 검사하고, 만족하지 못하는 경우 모델의 행위를 반례로 생성함으로 모델을 검증하는 도구이다. 모델 체킹 도구를 이용한 테스트 입력 시퀀스를 생성방법은 모델 체킹 도구에 모델과 속성을 입력하는데, '모델이 초기상태로부터 목표 상태에 도달할 수 없음'을 속성으로 입력하여 모델 체킹 도구가 이와 같은 속성이 참이 아님을 반례를 통해 찾아내도록 하는 방식이다. Fröhlich[13]은 테스트 입력 시퀀스를 생성하는 문제를 계획(planning) 문제로 변환하여 문제를 해결하는 접근 방법을 택하였는데, statechart 모델을 인공 지능 분야에서 매우 넓게 사용되는 표현 방식인 STRIPS 계획 언어로 변환하여 테스트 케이스를 생성하는 연구를 수행했다. 그리고 Van[9]

은 제약 사항 해결 방식으로 테스트 입력 시퀀스를 생성했는데 UML의 Class diagram, OCL, statecharts를 분해하여 관계가 있는 것들끼리 조합하여 (사전 조건, 사후 조건, 전이 사전 조건, 전이 사후 조건)의 집합을 생성하여 statecharts의 각 전이에 대해 위의 제약 조건을 만족하는 테스트 입력 시퀀스를 생성하는 방식이다. 그리고 Assertion 기반 방식[1, 2, 6]은 일종의 목표 기반의 방식의 일종으로 소프트웨어의 디버깅, 테스팅 도중에 소프트웨어의 에러를 실시간으로 탐지하는 방식이다. Assertion 기반으로 테스트 입력 시퀀스를 생성하는 방법은, 소프트웨어가 행동하는 각각의 상태들에 대해 그 상태에서 행동을 하고 그 후에 만족해야 하는 속성을 assertion으로 기술하며, 만약 프로그램이 실행 도중 임의의 상태에서의 행동 이후 기술된 assertion의 속성을 만족하지 못하는 경우 초기 상태에서 현재 상태까지의 행동 경로를 출력 결과로 내보냄으로 소프트웨어의 테스트 입력 시퀀스를 자동으로 생성하는 방식이다. 목표 기반의 방식에서는 초기 상태로부터 목표 상태를 찾기 위해 탐색하는 시간이 오래 걸린다는 단점을 가지지만, 대부분의 연구에서 목표 기반의 방식으로 연구를 수행하고 있다.

본 연구는 목표 기반의 테스트 입력 시퀀스를 생성하는 방식을 택하고 있으며, Timing diagram의 테스트 입력 시퀀스를 생성하는 문제를 선형 계획 문제로 변환하여 해결하는 접근 방법을 택하였는데, 이와 같은 방법을 선택한 이유는 선형 계획 문제로 접근하게 되면 모든 Timing diagram 모델의 테스트 입력 시퀀스의 생성 여부를 결정할 수 있으며, 생성 가능하다고 판단된 Timing diagram 모델에 대해서 매우 빠른 속도로 테스트의 입력 시퀀스 생성이 가능하기 때문이다. 이 점은 기존의 연구들에서 모든 모델에 대해 완벽하게 테스트 입력 시퀀스를 생성하지 못한 것과 비교했을 때 매우 큰 장점이라고 할 수 있다.

## 3. 배경지식

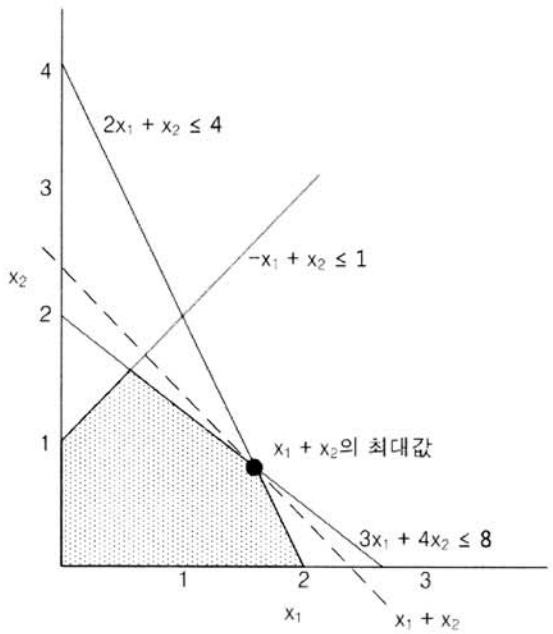
이 장에서는 본 논문의 핵심 내용인 선형 계획법과 Timing diagram에 대해 소개하고, 연구에서 해결하고자 하는 문제를 정의하였다. 3.1절에서 선형 계획법에 대해 소개하였고 정의 3-1에서 선형 계획법으로 해결하기 위한 선형 계획 모형을 정의하였다. 3.2절에서는 Timing diagram의 문법과 의미를 정의하였는데, 정의 3-2와 정의 3-4에서 각각 Timing diagram의 문법과 Timing diagram의 행동에 대해 형식적으로 정의하였다.

### 3.1 선형 계획법

선형 계획법(Linear Programming)은 여러 제약 조건을 만족시키면서 하나의 목적을 달성하기 위해 한정된 자원을 배분하는 수학적 계획법이다. 제약 조건 식과 목적은 선형적인 특징을 갖고 있으며, 두 개 이상의 의사 결정 변수가 존재해야 한다. 예를 들어 임의의 변수 $x_1$, $x_2$에 대하여 아래의 제약 조건을 만족하면서 $x_1+x_2$의 값이 최대가 되는 $x_1$, $x_2$의 값을 찾는다고 가정하자.

$$3x_1 + 4x_2 \le 8$$
$$-x_1 + x_2 \le 1$$
$$2x_1 + x_2 \le 4$$
$$x_1 \ge 0$$
$$x_2 \ge 0$$

이 문제에서 $x_1$, $x_2$는 의사 결정 변수가 되며 제약 조건식은 다섯 개가 되며, 목적함수는 $x_1+x_2$의 최대값이 된다. 위 예에서는 의사 결정 변수의 개수가 두 개 이므로, (그림 1)과 같이 그림을 그려서 문제를 해결 할 수 있다. (그림 1)의 음영처리 된 부분은 위의 문제에서 제약 조건을 만족할 수 있는 $x_1$, $x_2$의 값의 범위를 나타내며, $x_1 = 8/5$, $x_2 = 4/5$ 일 때, $x_1 + x_2$는 최대값을 갖는다. 이와 같은 선형 계획 모형에 대한 형식적 정의는 정의 3-1과 같다.




(그림 1) 그래프에 의한 선형 계획 문제의 해결

**[정의 3-1] 선형 계획 모형**

선형 계획 모형 lp는 <X, $f$, $\gamma$>로 구성되며 여기서 X는 결정 변수, $f$는 목적함수, $\gamma$는 제약 조건식의 집합을 의미한다. 결정변수 $X = \{ X_1, X_2, \cdots, X_k\}$는 목적함수 $f$의 파라미터이다. 제약 조건식 $\gamma = \{\gamma_1, \gamma_2, \cdots, \gamma_n\}$는 결정 변수로 이루어진 조건식이며, 목적함수는 결정 변수들의 함수 $f(X_1, X_2, \cdots, X_k)$로 표현되며 목적함수의 값을 최대화, 최소화, 혹은 임의의 지정된 값을 찾는 것이 선형 계획 모형의 목표이다.

### 3.2 Timing diagram의 문법과 의미

이 절에서는 본 논문에서 대상으로 하는 모델인 Timing diagram의 문법과 Timing diagram의 행동에 대해서 설명할 것이다. 정의 3-2과 정의 3-3에서 Timing diagram의 문법

을 기술하였고 정의 3-3에서 Timing diagram의 행동에 대해 정의하였다.

[정의 3-2] Timing diagram의 문법

Timing diagram TD는 <X, Y, Σ, D, C, ρ, δ>로 구성되어 있는데, X는 입력 waveform의 집합, Y는 출력 waveform의 집합, Σ = {Signal, Event}는 waveform 타입의 집합을 의미하며, D는 시점의 집합을 의미하는데, 시점은 입력 waveform의 값의 변화의 경계 점을 의미한다. 그리고 C는 시간 제약의 집합을 의미하며, 시간 제약에 대한 정의는 정의 3-3와 같다. ρ: (X ∪ Y) → Σ는 waveform에 대한 타입을 정의하는 함수를 의미하며, δ: (X ∪ Y) x D → VALUE는 waveform과 시점에 대한 값을 정의하는 함수를 의미하며, VALUE는 값의 집합을 의미한다. ■

[정의 3-3] 시간 제약

임의의 TD = <X, Y, Σ, D, C, ρ, δ>에 대해 시간 제약 C는 <$d_s$, $d_e$, t, v>로 구성되어 있으며 $d_s$ ∈ D, $d_e$ ∈ D는 각각 시작 시점, 끝 시점을 의미하고, T = {Duration, In, After}는 시간 제약의 타입의 집합일 때, t ∈ T는 시간 제약의 타입을 의미하고, v ∈ N (N은 자연수)는 시간 제약의 값을 의미한다. 시간 제약 타입 Duration은 시작 시점에서 끝 시점 기간의 시간이 시간 제약의 값이어야 함을 의미하며, In은 시간 제약의 값보다 구간 내의 시간이 작아야 함을 의미하며, After는 시간 제약의 값보다 구간내의 시간이 커야 함을 의미한다. ■

정의 3-3의 시간 제약에 대해 예를 들어 설명하면 시간 제약 tc1 = <$d_{s1}$, $d_{e1}$, Duration, _3sec>이 있을 때, tc1의 제약은 시간 제약 구간 [$d_{s1}$, $d_{e1}$]내에서 소비한 시간이 _3sec이어야 한다는 것이다. 만약 그 구간 내의 시간이 _3sec이면 시간 제약을 만족한 것이 되지만, _3sec보다 크면 시간 제약 tc1을 위반한 것이 된다. 또한 시간 제약 tc2 = <$d_{s2}$, $d_{e2}$, In, _5sec>의 제약은 구간 [$d_{s2}$, $d_{e2}$]내에서 소비한 시간이 _5sec 이내여야 할 것을 의미한다. 만약 그 구간 내에서 소비한 시간이 _5sec보다 작으면 시간제약을 만족한 것이지만 _5sec보다 같거나 크게 되면 그 조건은 tc2를 위반한 것이 된다. 그리고 시간 제약 tc3 = <$d_{s3}$, $d_{e3}$, After, _7sec>에 대해 tc3는 구간 [$d_{s3}$, $d_{e3}$]내에서 소비한 시간이 _7sec보다 클 것을 요구한다. 만약 그 구간 내에서 소비한 시간이 _7sec이내이면 시간 제약을 불만족한 것이 되며, _7sec보다 크게 되면 시간 제약을 만족한 것이 된다. After 타입의 시간 제약에 대해서는 제약의 위반은 발생하지 않는 특징이 있다. 모든 시간 제약은 불만족, 만족, 위반의 3가지 상태가 있을 수 있다. 시간 제약의 불만족은 현재는 시간 제약의 조건을 충족하지 않지만, 앞으로 시간이 지나면 조건을 충족할 수 있는 상태이고, 만족은 시간 제약의 조건을 충족하는 상태를 의미한다. 위반은 시간 제약의 조건을 현재 충족하지 않으며, 앞으로 계속 충족할 수 없는 상태를 의미한다.

Timing diagram의 매 시점마다 입력 waveform은 특정한 값을 가지지 않을 수도 있다. 예를 들어서 어떤 시점에서 어떤 입력 waveform의 값이 어떤 값이 되어도 상관없는 경우가 있을 수 있다. 이런 경우 무 상관(don't care)이란 의미를 가질 수 있으며, 그림 상으로 표현할 때에는 빗금으로 표현할 수 있다. (그림 2)는 도난 경보 모델을 Timing diagram 모델로 표현한 그림이다. 여기서 입력 Waveform의 집합 X = {Ready, Alarm}이고 출력 Waveform의 집합 Y= {Alarm_State, Indicator}이다. 시점의 집합 D = {$d_1$, $d_2$, $d_3$, $d_4$, $d_5$}이며, Σ(Ready) = Signal, Σ(Alarm) = Signal, Σ(Alarm_State) = Signal, Σ(Indicator) = Signal로 그림 2의 모델에서 모든 waveform의 타입은 시그널이며, Ready의 $d_1$, $d_2$, $d_3$에서의 값은 각각 δ(Ready, $d_1$) = 0, δ(Ready, $d_2$) = 1, δ(Ready, $d_3$) = 1이다. 그리고 하나의 시간 제약 tc1이 있는데, tc1 = <$d_3$, $d_4$, After, _3sec>이다.

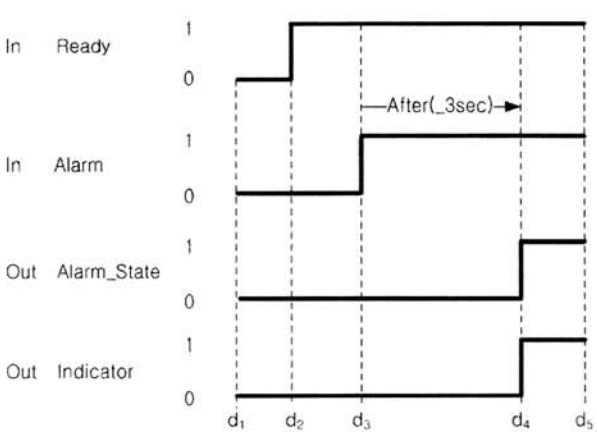
[정의 3-4] Timing diagram의 행동

임의의 Timing diagram TD = <X, Y, Σ, D, C, ρ, δ>에서 D = {$d_1$, $d_2$, ⋯ , $d_n$}일 때, $d_0$를 Timing diagram이 초기화되지 않을 때의 시점으로 정의하도록 하자. TD의 행동은 현재 시점 $d_{prev}$의 입력 waveform들의 벡터 값 $V_{inputs}$과 현재 시점으로부터 지난 시간 $t_{elapsed}$에 대한 변화된 시점 $d_{post}$과 출력 waveform들의 벡터 값 $V_{outputs}$으로 표현되며 $d_{post}$와 $V_{outputs}$은 [$d_{post}$ $V_{outputs}$] = $f_{Transition}$($d_{prev}$, $V_{inputs}$, $t_{elapsed}$)으로 정의된다.

(1) $f_{Transition}$에서 $d_{post}$의 정의

(a) i의 값이 (0 ≤ i ≤ n-2)이고 현재 시점($d_{prev}$)이 $d_i$일 때, 변화된 시점($d_{post}$)가 $d_{i+1}$이 되기 위한 조건: 현재 시점을 커버하고 있는 시간 제약들에 대해 제약 조건을 위반하지 않아야 하고 입력 waveform의 값이 Timing diagram에 기술된 다음 시점의 값과 같으면서 다음 시점에서 끝나는 시간 제약들의 조건을 모두 만족해야 한다.

(b) i의 값이 (0 ≤ i ≤ n-2)이고 현재 시점($d_{prev}$)이 $d_i$일




(그림 2) 도난 경보 모델에 대한 Timing diagram

때, 현재 상태를 유지하는 조건: 현재 시점을 커버하고 있는 시간 제약들에 대해 제약 조건을 위반하지 않아야 하고 입력 waveform의 값이 Timing diagram에 기술된 현재 시점의 값과 같아야 한다.

(c) i의 값이 ($1 \le i \le n-2$)이고 현재 시점($d_{prev}$)이 $d_i$일 때, TD가 비정상적으로 종료하는 조건: 현재 시점을 커버하고 있는 시간 제약들 중 적어도 하나가 제약 조건을 위반하거나 입력 waveform의 값이 현재 상태를 유지하는 조건을 만족하지 못하고 다음 시점으로 이동하는 조건을 만족하지 못해야 한다.

(d) 현재 시점($d_{prev}$)이 $d_{n-2}$일 때, TD가 정상적으로 종료하는 조건: 현재 시점을 커버하고 있는 시간 제약들에 대해 제약 조건을 위반하지 않아야 하고 입력 waveform의 값이 Timing diagram에 기술된 다음 시점의 값과 같으면서 다음 시점에서 끝나는 시간 제약들의 조건을 모두 만족해야 한다.

(2) $f_{Transition}$에서 $V_{outputs}$의 정의

(a) $f_{Transition}$에 의해 시점이 $d_i$에서 $d_{i+1}$로 이동하게 된 경우: $V_{outputs}$는 구간 $[d_{i+1}, d_{i+2}]$에 표현된 값으로 정의된다.

(b) $f_{Transition}$에 의해 현재 시점을 유지하는 경우: 출력 행동은 없다

(c) $f_{Transition}$에 의해 TD가 비정상적으로 종료하는 경우: 출력 행동은 없다

(d) $f_{Transition}$에 의해 TD가 정상적으로 종료하는 경우: $V_{outputs}$는 구간 $[d_{n-1}, d_n]$에 표현된 값으로 정의된다.■

정의 3-4에 정의되어 있는 Timing diagram의 행동에 대해 (그림 2)를 예를 들어서 설명하면, Timing diagram이 초기화 되어 있지 않을 때, Ready와 Alaram의 값이 모두 0이 되면 Timing diagram은 초기화되며, 이 때의 Alarm_State와 Indicator의 값은 모두 0이 되며, 시점은 $d_0$에서 $d_1$로 이동하게 된다. 그리고 시점이 $d_1$에 있을 때 $d_2$로 이동하기 위해서는 (Ready, Alaram)의 값은 (1, 0)이 되어야 한다. $d_2$에서 $d_3$로 이동하기 위해서는 (Ready, Alaram)의 값은 (1, 1)이 되어야 하며, $d_3$에서 $d_4$로 이동하기 위해서는 (Ready, Alaram)의 값이 (1, 1)인 상태를 유지한 채로 _3sec가 지나야 한다. $d_3$에서 $d_4$로 이동할 때의 출력 결과는 Alarm_State와 Indicator의 값은 모두 1이 되며 Timing diagram은 정상적으로 종료된다. 만약 $d_2$에서 (Ready, Alaram)의 값이 (0, 0)이 된다면 이 값은 현재 시점을 유지하기 위한 조건도 만족하지 못하며, 다음 시점으로 이동하기 위한 조건도 만족하지 못한다. 그러므로 이 경우 Timing diagram은 비 정상적으로 종료하게 된다.

## 4. Timing diagram의 테스트 입력 시퀀스 생성

이 장에서는 Timing diagram 모델 TD의 테스트 입력 시퀀스를 생성하는 방법에 대해서 설명한다. 정의 4-1에서 테스트 케이스 목표에 대해 정의하였으며, 문제 4-2에서 본 논문에서 해결하고자 하는 문제에 대해서 정의하였다. 문제 4-2의 테스트 입력 시퀀스 알고리즘이 동작하기 위한 전제 조건은 테스트 케이스 목표에 속한 시점(d)까지 TD를 이동시켜야 한다. 이를 위해서 정의 4-3에서 Timing diagram의 도달 가능성에 대해서 정의하였다. 그리고 TD를 d까지 이동시키기 위해서는 시간과 입력 waveform들의 값에 대한 벡터 정보가 필요하다. 이를 정의 4-4에서 테스트 입력 시퀀스 테이블을 정의하였는데, 여기서 시간과 입력 waveform은 제어 가능함을 전제로 하였다.

**[정의 4-1] 테스트 케이스 목표**

임의의 Timing diagram 모델 TD = <X, Y, Σ, D, C, ρ, δ>에 대한 TD의 테스트 케이스 목표 obj는 <d, $V_{Inputs}$, Constraints_Time>로 구성되어 있는데, $d \in D$는 Timing diagram의 시점을 의미하고, $V_{Inputs}$는 입력 waveform들의 벡터 값을 의미하며, Constraints_Time은 시점 d에서 $V_{Inputs}$이 들어오기 전에 소비해야 하는 시간을 의미한다. 예를 들어 어떤 obj = <$d_3$, $V_{inputs}$, 3>가 있다고 할 때 이 테스트 목적의 의미는 $d_3$시점에서 3 tick을 소비하고 $V_{inputs}$값이 입력으로부터 들어와야 함을 의미한다. ■

**[문제 4-2] Timing diagram의 테스트 입력 시퀀스 생성**

임의의 Timing diagram 모델 TD = <X, Y, Σ, D, C, ρ, δ>과 TD의 테스트 케이스 목표 obj = <d, $V_{Inputs}$, Constraints_Time>에 대해 테스트 입력 시퀀스 알고리즘($Alg_{TIS}$)은 TD, obj를 입력으로 받아서 테스트 입력 시퀀스를 결과로 생성하거나 혹은 생성 불가능함을 결과로 생성한다. ■

**[정의 4-3] Timing diagram 모델 TD에 속한 시점 $d_k$의 도달 가능성**

임의의 Timing diagram TD = <X, Y, Σ, D, C, ρ, δ>이고, D = {$d_1$, $d_2$, …, $d_n$}에 대해 초기 시점에서 $1 \le k \le n-1$인 $d_k$의 시점으로 이동할 수 있을 때, 그 TD를 k-reachable이라고 정의한다. 그리고 특별히 TD가 $d_{n-1}$시점까지 이동할 수 있을 때, 그 TD를 reachable-complete라고 정의한다. ■

**[정의 4-4] 테스트 입력 시퀀스 테이블**

X = {$w_1$, $w_2$, …, $w_m$}인 임의의 Timing diagram TD = <X, Y, Σ, D, C, ρ, δ>에 대한 테스트 입력 시퀀스 테이블 TBL은 <time, Vector>의 리스트로 구성되어 있는데, time은 초기 상태로부터 흐른 시간, Vector는 ($value_1$, $value_2$, …, $value_m$)로 구성되어 있는 입력 벡터로, $1 \le i \le m$인 i에 대한, $value_i$는 $w_i$에 대한 값을 의미한다. ■

앞으로의 편의를 위해 테스트 입력 테이블 TBL과 관련된 표현에서 TBL(j)를 TBL의 j번째 행으로, time(i)는 TBL의 i번째 행의 time값으로 표현하도록 하겠다.

**[정리 4-5] 시간 제약이 없는 Timing diagram의 도달 가능성**

X ={$w_1$, $w_2$, … $w_m$}이고 Y = {$w_{m+1}$, …, $w_k$}일 때, D =

$\{d_1, d_2, \cdots, d_n\}$, C = Ø인 임의의 Timing diagram 모델 TD = <X, Y, Σ, D, C, ρ, δ>에 대해 TD는 reachable-complete하다

**증명)** TD에 대한 테스트 입력 시퀀스 테이블을 TBL이라고 할 때, TBL는 다음과 같이 정의한다.

$$TBL(i) \equiv \begin{cases} < 0 & , f_{vector}(i) > , i = 1 \\ < time(i-1)+1 & , f_{vector}(i) > , 2 \le i \le n-1 \end{cases}$$

$f_{vector}(i) \equiv (\delta(w_1, d_i), \delta(w_2, d_i), \cdots, \delta(w_m, d_i))$

이 때, TBL(1)은 TD를 초기화 시키는 입력 값을 의미하는데, time은 0이고 각 waveform의 값은 $f_{vector}(1)$로 표현된다. $f_{vector}(i)$는 시점 $d_i$에 표현된 입력 waveform들에 대한 벡터 값을 의미한다. 입력 waveform은 제어 가능하다고 전제했으므로 임의의 시점 $d_i$에 대한 waveform값의 벡터는 생성 가능하다. 그리고 $2 \le i \le n-1$인 i에 대해 TBL(i)는 TD가 시점 $d_{i-1}$로부터 $d_i$로 진입하기 위한 시간 및 입력 waveform 벡터 값을 의미한다. TBL(i)에서의 time은 구간 $[d_{i-1}, d_i]$에서 소비하는 시간을 의미하는데, 이 구간 내의 시간 제약이 없으므로 최소값인 1로 정의할 수 있다. 즉 위의 테스트 입력 시퀀스 테이블 TBL은 매 tick마다 새로운 시점으로 전이가 일어나게 된다. 정리하면 위의 Timing diagram 모델 TD에서 시점 $d_i$까지 도달하기 위한 테스트 입력 시퀀스는 TBL(1)으로부터 TBL(i)까지의 입력 값에 대한 결합을 의미한다. TD는 TBL로부터 $1 < i \le n-1$인 i에 대해 임의의 시점 $d_i$에 진입할 수 있으며, 그러므로 TD는 reachable-complete이다. (증명끝) ■

정리 4-5에서 시간 제약이 없는 Timing diagram모델은 reachable-complete하며 결국 어떤 테스트 케이스 목적에 대해서 테스트 입력 시퀀스 생성이 가능함을 증명하였다. 이를 (그림 2)의 도난 경보 모델을 대상으로 예를 들어 설명해 보도록 하겠다. 도난 경보 모델의 시점 $d_3$에서 시간 제약이 시작하고 있으므로 초기 상태에서 $d_3$까지 도달하는 테스트 입력 시퀀스 테이블을 만들어보도록 하겠다. 정리 4-5에서 정의한 TBL(i)으로부터 도난 경보 모델의 초기상태에서 $d_3$까지 도달하는 테스트 입력 시퀀스 테이블은 표 1과 같다. 표 1의 비고 열은 표를 설명하기 위해 편의상 넣은 것이며 테스트 입력 시퀀스 테이블에 들어가는 열은 아니다. <표 1>의 첫 번째 행은 모델을 초기화 하는 입력이며, 두 번째 행은 모델의 시점 $d_1$으로부터 $d_2$로 전이시키는 입력이다. 마지막으로 세 번째 행은 모델의 시점 $d_2$로부터 $d_3$로 전이시키는 입력이다.

정리 4-5에서 증명한 Timing diagram 모델의 테스트 입력 시퀀스 생성 알고리즘은 시간 제약이 없는 특수한 모델에 대한 것이었으며, 앞으로 시간 제약이 존재하는 일반적인 Timing diagram모델에 대한 테스트 입력 시퀀스 생성 알고리즘에 대해서 설명하도록 할 것인데, 이 문제를 선형 계획법으로 모델링 하는 방법에 대해서 기술할 것이다.

일반적인 Timing diagram에 대해서 생각해 보기 위해 1개의 waveform과 m개의 시점과 n개의 시간 제약을 가지고

〈표 1〉 도난 경보 모델의 $d_3$까지 도달하는 테스트 입력 시퀀스 테이블

| time | Ready | Alarm | 비고 |
|---|---|---|---|
| 0 | 0 | 0 | 모델의 초기화 |
| 1 | 1 | 1 | $d_1 \rightarrow d_2$로 전이 |
| 2 | 0 | 1 | $d_2 \rightarrow d_3$로 전이 |

있는 임의의 모델 TD이 있다고 가정하자. 정리 4-5로부터 입력 waveform은 제어 가능하므로 시간 제약에 대한 제어 가능성의 여부에 따라 테스트 케이스 입력 시퀀스의 생성 여부가 결정된다. 즉 시퀀스 입력 테이블을 생성하기 위해서는 각 구간에서 소비해야 하는 시간과 waveform들의 벡터 값을 생성해야 하는데, waveform들의 벡터 값은 제어 가능하기 때문에 각 구간에서 소비해야 하는 시간에 대해서만 고려하면 테스트 케이스 입력 시퀀스를 생성하는 문제를 풀 수 있다. 이를 위해 waveform은 생략되고 시간 제약만 표현되어 있는 (그림 3)과 같은 모델을 예로 들어 각 구간 내에서 소비해야 하는 시간을 구하는 방법을 설명하고 이를 토대로 일반적인 Timing diagram 모델의 테스트 입력 시퀀스를 생성하는 알고리즘을 기술하도록 하겠다. 그림 3에서 세 개의 시간 제약 $tc_1$, $tc_2$, $tc_3$에 대해 $tc_1$ = <$d_2$, $d_5$, after, 5>, $tc_2$ = <$d_3$, $d_6$, during, 4>, $tc_3$ = <$d_4$, $d_7$, during, 5>라고 가정하자. 여기서 $\chi_i$를 구간 $[d_i, d_{i+1}]$에서 소비하는 시간이라 정의할 때, $tc_i$의 시간 제약 조건을 $\chi_i$의 함수인 $\gamma_i$로 표현하면 다음과 같다.

$$\begin{aligned} \gamma_1 &\equiv \chi_2 + \chi_3 + \chi_4 > 5 \\ \gamma_2 &\equiv \chi_3 + \chi_4 + \chi_5 = 4 \\ \gamma_3 &\equiv \chi_4 + \chi_5 + \chi_6 = 5 \end{aligned} \quad (1)$$

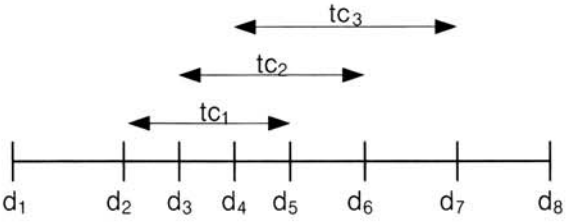
또한 각각의 구간 $[d_i, d_{i+1}]$은 최소한 1 tick의 시간을 소비하기 때문에 각각의 $\chi_i$는 1보다 크거나 같은 정수이어야 한다. 이를 $\gamma_4$로 표현하면 다음과 같다.

$$\gamma_4 \equiv \chi_2, \chi_3, \chi_4, \chi_5, \chi_6 \in N \quad (2)$$
(여기서 N은 자연수의 집합)

위 식을 만족하는 미지수 $\chi_2, \chi_3, \cdots, \chi_6$은 매우 많은 답이 있다. 그 중에서 각 구간 별로 최소한의 시간을 소비하는 미지수를 찾는 함수 $f$를 정의한다면 다음과 같이 정의할 수 있다.

$$f = \sum_{i=2}^{6} \chi_i \text{의 최소화} \quad (3)$$

$f$의 값을 만족하는 미지수 $\chi_2, \chi_3, \cdots, \chi_6$를 찾으면 TD는 reachable-complete하며, 값을 찾지 못한다면 구간 $[d_2, d_6]$내에 속한 어느 시점에 도달 불가능함을 의미한다. 여기서 위의 (1), (2), (3)에서 정의한 식이 선형 계획 모형 lp이며 이와 같이 임의의 Timing diagram 모델의 각 구간의 소비하는 시간을 구하기 위해서 선형 계획 모형으로 모델링 하여




(그림 3) 여러 개의 시간 제약들

문제를 해결할 수 있다. 임의의 Timing diagram 모델의 테스트 입력 시퀀스를 생성하기 위해서는 시점에 대한 각 구간의 소비 시간을 계산하는 것이 중요한데, 이 문제를 선형 계획 모형으로 정의 4-6과 같이 모델링 할 수 있다.

**[정의 4-6] 임의의 Timing diagram 모델에 대한 선형 계획 모형 모델링**

임의의 Timing diagram 모델 TD = <X, Y, Σ, D, C, ρ, δ>에 대해 D = {$d_1$, $d_2$, ⋯, $d_j$}이고 C = {$c_1$, $c_2$, ⋯, $c_k$}이고, 각각의 $c_i \in C$에 대해 $c_i$ = ($d_{si}$, $d_{ei}$, $t_i$, $v_i$)일 때, 각 시점의 구간의 소비 시간을 찾기 위한 선형 계획 모형 lp <χ, $f$, γ>는 다음과 같이 정의한다.

$\chi = \{\chi_1, \chi_2, \cdots, \chi_{j-1}\}$

$f = \sum_{i=1}^{j-1}\chi_i$의 최소화

$\gamma = \{\gamma_1, \gamma_2, \cdots, \gamma_{k+1}\}$,

$$\gamma_i \equiv \begin{cases} \sum_{i=d_{si}}^{d_{ei}} \Upsilon_i > v_i & , t_i = \text{after} \\ \sum_{i=d_{si}}^{d_{ei}} \Upsilon_i = v_i & , t_i = \text{during} \\ \sum_{i=d_{si}}^{d_{ei}} \Upsilon_i < v_i & , t_i = \text{in} \end{cases}$$

$\gamma_{k+1} = \chi_1, \chi_2, \cdots, \chi_k \in N$ (N은 자연수의 집합) ■

정리 4-5와 정의 4-6으로부터 임의의 Timing diagram 모델 TD와 그의 선형 계획 모형 lp에 대해 $f$를 만족하는 결정 변수가 존재할 경우 TD에 대한 테스트 입력 시퀀스 테이블 TBL은 정의 4-7과 같이 정의할 수 있다. TBL(1)은 초기 상태에 도달하기 위한 테스트 입력을 의미하며, TBL(i)은 시점 $d_{i-1}$에서 $d_i$로 진입하기 위한 테스트 입력을 의미한다.

**[정의 4-7] 임의의 Timing diagram모델에 대한 테스트 입력 시퀀스 자동 생성**

임의의 Timing diagram 모델 TD = <X, Y, Σ, D, C, ρ, δ>에 대해 D = {$d_1$, $d_2$, ⋯, $d_j$}이고 C = {$c_1$, $c_2$, ⋯, $c_k$}이고, 각각의 $c_i \in C$인 $c_i$ = ($d_{si}$, $d_{ei}$, $t_i$, $v_i$)라 가정하자. TD의 선형 계획 모형 lp <χ, $f$, γ>에 대해 만족하는 해가 존재하면 테스트 입력 시퀀스 테이블 TBL은 다음과 같이 정의한다.

$$TBL(i) \equiv \begin{cases} < 0 & , f_{vector}(i) > \quad , i = 1 \\ < time(i-1) + \chi_i & , f_{vector}(i) > \quad , i > 1 \end{cases}$$

$f_{vector}(i) \equiv (\delta(w_1, d_i), \delta(w_2, d_i), \cdots, \delta(w_m, d_i))$ ■

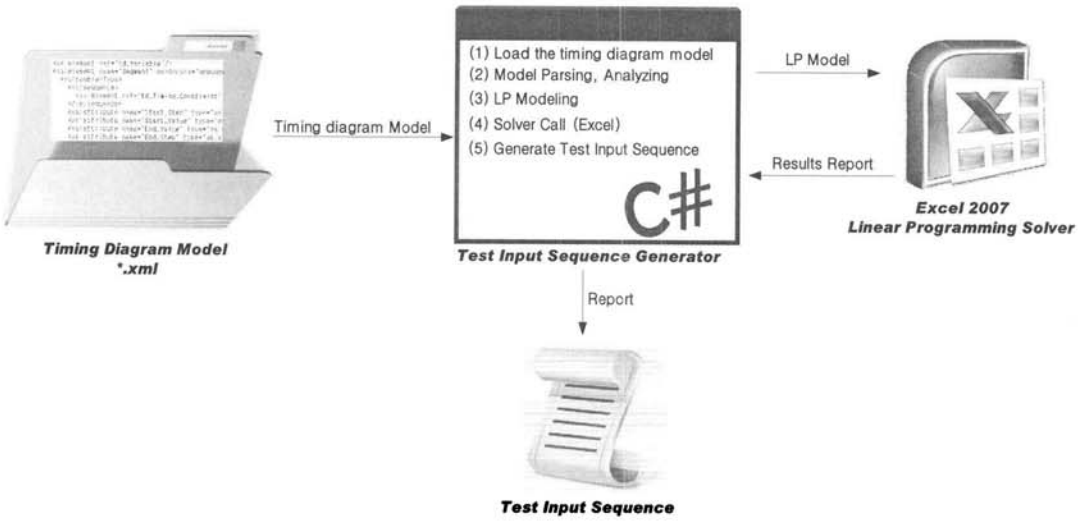
정리 4-5, 정의 4-6, 정의 4-7로부터 임의의 Timing diagram 모델에 대한 테스트 입력 시퀀스를 생성하기 위해 Timing diagram모델을 선형 계획 모형으로 모델링하고, 그 문제를 해결함으로써 테스트 입력 시퀀스를 생성하는 것을 보였다. 지금부터는 임의의 Timing diagram 모델과 임의의 테스트 목적이 존재할 때, 모델이 테스트 목적을 만족할 수 있는 테스트 입력 시퀀스가 존재하는지를 판별하는 방법에 대해서 설명하도록 하겠다. Timing diagram 모델 TD = <X, Y, Σ, D, C, ρ, δ>에 대한 테스트 목적 obj = <$d_{goal}$, $V_{Inputs}$, Constraints_Time>가 있을 때, 다음의 과정으로 테스트 입력 시퀀스를 생성할 수 있다. 1) TD가 $d_{goal}$까지 도달하는 테스트 입력 시퀀스를 생성한다. 2) Constraints_Time 기간 후에 $V_{Inputs}$를 생성하는 테스트 입력을 생성한다. 여기서 첫 번째 과정을 만족시키기 위해서는 정의 4-6을 이용할 수 있는데, TD로부터 구간 [$d_1$, $d_{goal}$]을 커버하는 시간 제약들의 집합 C′에 대해 새로운 Timing diagram 모델 TD′ = <X, Y, Σ, D, C′, ρ, δ>을 생성하고, TD′으로부터 선형 계획 모형을 생성하여 만족하는 해를 찾는다. 만족하는 해가 존재하면 $d_{goal}$까지 도달할 수 있는 테스트 입력 시퀀스를 생성할 수 있으며, 그렇지 않으면 테스트 입력 시퀀스를 생성할 수 없다. 첫 번째 과정이 성공적으로 수행하여 TBL(1), ⋯, TBL(goal)을 생성했으면 두 번째 과정인 Constraints_Time기간 후에 $V_{inputs}$를 생성하는 테스트 입력을 생성할 수 있는데, 이 테스트 입력은 TBL(goal+1)에 생성하며 다음과 같이 정의할 수 있다.

TBL(goal+1) ≡ < time(goal) + Constraints_Time, $V_{inputs}$ >

## 5. 도구의 구현과 예제

이 장에서는 4장에서 언급한 Timing diagram 모델과 테스트 목적으로부터 테스트 입력 시퀀스를 생성하는 도구의 구현 방법과 도구를 이용하여 Timing diagram 모델의 테스트 입력 시퀀스를 생성하는 예를 설명할 것이다. 본 논문에서 구현한 도구의 개발 환경은 Visual Studio C# 2008이며 선형 계획 문제 해결 도구는 Excel 2007의 해 찾기 기능을 이용하여 문제를 해결했다. 프로그램의 전체 흐름은 (그림 4)와 같다. 테스트 입력 시퀀스 생성기(TISG)는 xml파일 형식의 Timing diagram 모델을 읽어서 모델을 분석한다. 그리고 분석한 모델로부터 정의 4-6과 같은 선형 계획 모형으로 모델링을 하여 Excel 2007의 시트에 선형 계획 모형을 모델링을 하여, 해 찾기 도구를 이용하여 선형 계획 문제를 해결한다. TISG는 Excel 2007의 결과로부터 Timing diagram모델의 정의 4-7과 같은 테스트 입력 시퀀스를 생성한다.

TISG를 이용한 Timing diagram의 테스트 입력 시퀀스 생성이 유용함을 보이기 위해서 예제를 이용하여 설명하도록 하겠다. (그림 5)와 같은 Timing diagram 모델 예제로부




(그림 4) 테스트 입력 시퀀스 생성기(TISG)의 전체 흐름도

터 테스트 입력 시퀀스를 생성하고자 할 때, 이 모델의 선형 계획 모형은 다음과 같이 정의할 수 있다.

$X = \{x_1, x_2, \cdots, x_8\}$

$f = \sum_{i=1}^{j} x_i$의 최소값

$\gamma = \{\gamma_1, \gamma_2, \gamma_3, \gamma_4, \gamma_5\}$,

$\gamma_1 \equiv x_2 + x_3 + x_4 + x_5 > 8$

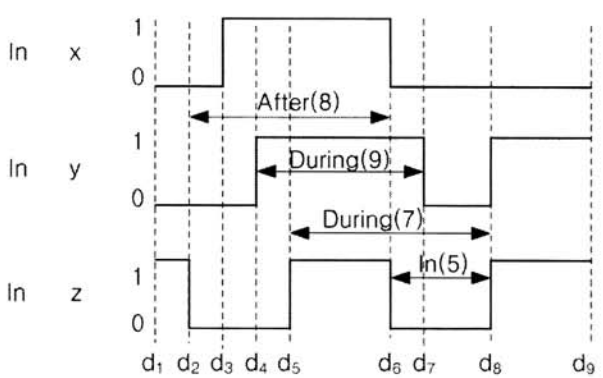
$\gamma_2 \equiv x_4 + x_5 + x_6 = 9$

$\gamma_3 \equiv x_5 + x_6 + x_7 = 7$

$\gamma_4 \equiv x_6 + x_7 < 5$

$\gamma_5 \equiv x_1, x_2, \cdots, x_8 \in N$ (N은 자연수의 집합)

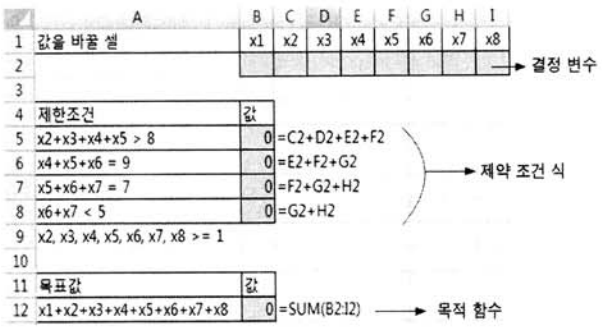
여기서 $x_i$는 구간 $[d_i, d_{i+1}]$을 소비해야 하는 시간을 의미하는 결정 변수이며, $X$는 결정 변수들의 집합을 의미한다. $f$는 목적 함수를 의미하며, $X$에 속한 모든 원소들의 합의 최소값으로 정의된다. $\gamma$는 제약 조건식의 집합으로 각각의 시간 제약 c에 대해, c가 포함하고 있는 결정 변수들이 만족해야 하는 식으로 정의된다. 예를 들어 $\gamma_1$는 시간 제약 c = <$d_2$, $d_6$, after, 8>에 대한 제약 조건식으로 c가 포함하는 구간들에서 소비해야 하는 시간이 8보다 커야 하므로 $\gamma_1$은 $x_2 + x_3 + x_4 + x_5 > 8$과 같이 정의된다. 그리고 $\gamma_5$는 결정 변수들이 가져야 하는 제약 조건 식으로 $\gamma_5$는 $x_1, x_2, \cdots, x_8 \in N$으로 정의된다.



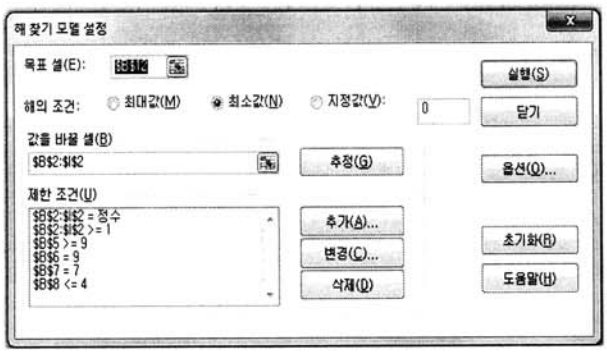

(그림 5) Timing diagram 모델 예제

위의 선형 계획 모델링 식을 TISG에서 Excel 2007 시트에 표현한 것이 (그림 6)이고, Excel 2007의 선형 계획 모형을 계산하기 위한 해 찾기 기능이 (그림 7)과 같다. (그림 6)에서 셀 [2, 2]~[2, 9]의 회색으로 음영 처리된 부분이 결정 변수 $x_1, x_2, \cdots, x_8$을 의미하며, 셀 [5, 2]~[8, 2]의 회색으로 음영 처리된 부분이 Timing diagram의 제약 조건에 의한 제약 조건 식에 사용되는 변수를 의미한다. 예를 들어 셀 [5, 2]의 셀 내용은 '=C2 + D2 + E2 + F2'로 표현되며, 이 의미는 $\sum_{i=2}^{5} x_i$를 의미한다. 즉 시간 제약 c = <$d_2$, $d_6$, after, 8>가 커버하는 구간에서 소비한 총 시간을 의미한다. 이 셀은 (그림 7)의 해 찾기 기능에서 '셀 [5, 2]의 값이 8보다 커야 한다'로 정의함으로써 시간 제약 c의 제약 조건 식을 정의하게 된다. 그리고 셀 [6, 2]은 '=E2 + F2 + G2'로 표현되며, 이 의미는 $\sum_{i=1}^{8} x_i$를 의미한다. 위 셀의 내용은 시간 제약 c = <$d_4$, $d_6$, duration, 9>가 커버하는 구간에서 소비하는 총 시간을 의미하며, (그림 7)의 해 찾기 기능에서 '셀 [6, 2]의 값이 9이어야 한다'고 정의함으로써 시간 제약 c의 제약 조건식을 정의한다. 각각의 결정 변수에 대한 제약 조건 식은 (그림 6)에서는 표현하지 않았으며 (그림 7)에서 표현하였다. 그리고 셀 [12, 2]의 회색으로 음영 처리된 부분이 목표 함수 $\sum_{i=1}^{8} x_i$이다.

(그림 7)은 선형 계획 모형을 계산하기 위한 Excel 2007의 해 찾기 기능에 대한 그림을 나타낸다. (그림 7)에서 목표 함수, 제약 조건 식, 결정 변수에 대하여 (그림 6)의 셀로 정의를 한다. 예를 들어 (그림 7)의 목표 셀이 $B$12로 표현되어 있는데, 이는 (그림 6)의 셀 [12, 2]를 의미하며, 값을 바꿀 셀이 $B$2:$I$2로 표현되어 있는데, 이는 그림 6의 셀 [5, 2]~[8, 2]로 결정 변수들을 의미한다. 그리고 제한 조건들은 시간 제약 조건 및 결정 변수의 조건을 의미한다. (그림




(그림 6) Excel 2007시트에 표현된 선형 계획 모형




(그림 7) Excel 2007의 해 찾기 모델 기능

7)의 제약 조건 식에서 제약조건 $\gamma_1 \equiv x_2 + x_3 + x_4 + x_5 > 8$에 대해 엑셀에서 '>'연산자가 존재하지 않아 $x_2 + x_3 + x_4 + x_5 \geq 9$로 변경하여 입력을 하였다. (그림 8)은 선형 계획 모형을 계산한 결과를 나타내며, 계산 결과 결정 변수 $x_1, x_2, \cdots, x_8$의 값은, 각각 1, 1, 1, 3, 4, 2, 1과 같이 되었다. (그림 8)에서 볼 수 있듯이 각각의 결정 변수에 대한 제약 조건, '모든 결정 변수는 자연수이어야 한다' 는 만족되었으며, 모든 Timing diagram의 시간 제약 조건 역시 셀 [5, 2]~[8, 2]의 값에서 알 수 있듯이 만족되었음을 알 수 있다.

(그림 8)과 같은 계산 결과로부터 테스트 입력 시퀀스를 생성할 수 있다. (그림 5)의 Timing diagram 모델에 대한 테스트 입력 시퀀스 테이블 TBL을 생성하면 <표 2>와 같다. <표 2>를 이용하여 임의의 테스트 목적에 대한 테스트 입력 시퀀스를 생성하는 방법은 생략하도록 하겠다.

| | A | B | C | D | E | F | G | H | I |
|---|---|---|---|---|---|---|---|---|---|
| 1 | 값을 바꿀 셀 | x1 | x2 | x3 | x4 | x5 | x6 | x7 | x8 |
| 2 | | 1 | 1 | 1 | 3 | 4 | 2 | 1 | 1 |
| 3 | | | | | | | | | |
| 4 | 제한조건 | 값 | | | | | | | |
| 5 | x2+x3+x4+x5 > 8 | 9 | =C2+D2+E2+F2 | | | | | | |
| 6 | x4+x5+x6 = 9 | 9 | =E2+F2+G2 | | | | | | |
| 7 | x5+x6+x7 = 7 | 7 | =F2+G2+H2 | | | | | | |
| 8 | x6+x7 < 5 | 3 | =G2+H2 | | | | | | |
| 9 | x2, x3, x4, x5, x6, x7, x8 >= 1 | | | | | | | | |
| 10 | | | | | | | | | |
| 11 | 목표값 | 값 | | | | | | | |
| 12 | x1+x2+x3+x4+x5+x6+x7+x8 | 13 | =SUM(B2:I2) | | | | | | |

(그림 8) 해 찾기 기능을 수행한 결과

〈표 2〉 그림 3의 Timing diagram 모델에 대한 테스트 입력 시퀀스 테이블

| time | x | y | z | 비고 |
|---|---|---|---|---|
| 0 | 0 | 0 | 1 | 모델의 초기화 |
| 1 | 0 | 0 | 0 | $d_1 \rightarrow d_2$로 전이 |
| 2 | 1 | 0 | 0 | $d_2 \rightarrow d_3$로 전이 |
| 3 | 1 | 1 | 0 | $d_3 \rightarrow d_4$로 전이 |
| 6 | 1 | 1 | 1 | $d_4 \rightarrow d_5$로 전이 |
| 10 | 0 | 1 | 0 | $d_5 \rightarrow d_6$로 전이 |
| 12 | 0 | 0 | 0 | $d_6 \rightarrow d_7$로 전이 |
| 13 | 0 | 1 | 1 | $d_7 \rightarrow$ 모델의 종료 |

## 6. 결론 및 향후 연구

본 연구에서는 Timing diagram의 테스트 수행 자동화를 위해 임의의 Timing diagram 모델과 임의의 테스트 케이스 목적으로부터 테스트 입력 시퀀스를 생성하기 위한 방법을 제시하였으며, 이를 위해 다음과 같은 단계를 거쳤다. 1) Timing diagram을 형식적으로 기술하였다. 2) Timing diagram 모델의 테스트 입력 시퀀스 생성 알고리즘을 제시하고 이를 증명하였다. 3) 예제를 통해 알고리즘을 설명하였다. 이 연구를 통해 Timing diagram으로 작성된 시스템의 사양에 대한 테스트 입력 시퀀스를 자동으로 생성할 수 있게 되었으며, Timing diagram 모델 기반으로 테스트를 수행하고자 할 때, 자동화 할 수 있는 기술을 얻게 되었다. 본 연구에서 제시한 선형 계획 모델 접근 방법은 목표 기반의 테스트 케이스 생성 전략 혹은 제약 조건을 가진 방정식이 있는 모델의 테스트 입력 시퀀스를 생성하는데 유용한 방법이다. 그러므로 향후 연구로 Stateflow모델과 같은 Timing diagram이외의 모델을 대상으로 선형 계획 방법을 이용하여 테스트 입력 시퀀스를 생성하는 방법을 연구할 계획이다.

## 참 고 문 헌

[1] Korel, B.; Qi Zhang; Li Tao;, "Assertion-Based Validation of Modified Programs," Software Testing Verification and Validation, 2009. ICST '09. International Conference on, Vol., No., pp.426-435, 1-4 April 2009.

[2] Ali M. Alakeel, "An Algorithm for Efficient Assertions-Based Test Data Generation", Journal of Software, Vol.5, No.6, pp. 644-653, 2010.

[3] S. Rayadurgam; M.P.E. Heimdahl, "Coverage based test-case generation using model checkers," Engineering of Computer Based Systems, 2001. ECBS 2001. Proceedings. Eighth Annual IEEE International Conference and Workshop on the, Vol., No., pp.83-91, 2001.

[4] Jee-Eun Yoo, "Using Model Checking to Generate Data-Flow Oriented Test Case from Statecharts," Master thesis, KAIST, 2002.

[5] Reactis: http://www.reactive-systems.com/

[6] J. Tong, M. Boule, Z. Zilic, "Airwolf-TG: A Test Generator for Assertion-Based Dynamic Verification," Proceedings of IEEE High-level Design Validation and Test Workshop, HLDVT'09, pp.106-113, Nov. 2009.

[7] Mokhtar Bazaraa et al, "Linear programming and network flows," Wiley Press (3rd edition), 2004.

[8] 박구현, 송한식, 원중연, "경영과학-엑셀활용," 교보문고, 2009.

[9] Resolution Lionel Van, Lionel Van Aertryck, Thomas Jensen, "UML-CASTING: Test synthesis from UML models using constraint," In Proc.AFADL'2003 (Approches Formelles dans l'Assistance au Dévelopment de Logiciel), 2003. Available on the internet: http://www.irisa.fr/triskell/AFADL2003/actes AFADL2003/test04.pdf

[10] Gr´egoire Hamon, Leonardo de Moura and John Rushby, "Generating Efficient Test Sets with a Model Checker," Software Engineering and Formal Methods, International Conference on Second International Conference on Software Engineering and Formal Methods (SEFM'04), pp.261-270, 2004.

[11] Ashalatha Nayak, Debasis Samanta: "Automatic Test Data Synthesis using UML Sequence Diagrams," in Journal of Object Technology, vol. 09, no. 2, pp. 75-104, 2010. Available on the internet: http://www.jot.fm/issues/issue201003/article2/

[12] Yongyan Zheng, Jiong Zhou, Paul Krause, "An Automatic Test Case Generation Framework for Web Services", in Journal of Software, Vol.2, No.3, pp.64-77, 2007.

[13] Peter Fröhlich, Johannes Link, "Automated Test Case Generation from Dynamic Models", In Proceedings of the 14th European Conference on Object-Oriented Programming, Lecture Notes In Computer Science, Vol.1850, pp.472-492, 2000.

[14] Nam Hee Lee, Sung Deok Cha, "Generating test sequences using symbolic execution for event-driven real-time systems," Microprocessors and Microsystems, Vol27, Issue 10, pp.523-531, 2003.

[15] Gr´egoire Hamon, Leonardo de Moura and John Rushby, "Automated Test Generation with SAL," CSL Technical Note, 2005. Available on the internet: http://fm.csl.sri.com/~rushby/abstracts/sal-atg

[16] 오정섭, 최경희, 정기현, "1대1 요구사항 모델링을 통한 테스트 케이스 자동 생성," 정보처리학회 논문지 D, Vol.17D, No.1, pp.41-52, 2010.

[17] 정인상, "SAT를 이용한 MC/DC 블랙박스 테스트 케이스 자동 생성," 정보처리학회 논문지 D, Vol.16D, No.6, pp.911-920, 2009.

**이 홍 석**

e-mail : hsyi98@naver.com

2003년 아주대학교 전자공학부(공학사)

2003년 아주대학교 정보및컴퓨터공학부 (공학사)

2005년 아주대학교 전자공학과(공학석사)

2005년~현 재 아주대학교 전자공학과 박사과정

관심분야 : 명세 기술, 정형 검증, 소프트웨어 공학, 임베디드 시스템

**정 기 현**

e-mail : khchung@ajou.ac.kr

1984년 서강대학교 전자공학과 졸업(학사)

1988년 미국 Ilinois주립대 EECS 졸업(석사)

1990년 미국 Purdue대학 전기전자공학부 졸업(박사)

1991년~1992년 현대반도체 연구소

1993년~현 재 아주대학교 전자공학부 교수

관심분야 : 컴퓨터구조, VLSI 설계, 멀티미디어 및 실시간 시스템 등

**최 경 희**

e-mail : khchoi@ajou.ac.kr

1976년 서울대학교 수학교육과 졸업(학사)

1979년 프랑스 그랑데꼴 Enseeiht대학 졸업 (석사)

1982년 프랑스 Paul Sabatier대학 정보공학부 졸업(박사)

1982년~현 재 아주대학교 정보통신전문대학원 교수

관심분야 : 운영 체제, 분산시스템, 실시간 및 멀티미디어시스템 등