특집원고

하이브리드 시스템 정형 검증 분야 소개 및 동향

국방과학연구소 ▌윤상현
건국대학교 ▌유준범

1. 서 론

하이브리드 시스템[1]은 연속 시스템(Continuous System)과 이산 시스템(Discrete System)이 융합된 형태의 시스템으로, 연속적인 부분은 주로 시스템의 주변의 환경을 나타내는데 사용되며 이산적인 부분은 환경의 변화에 따른 시스템의 작동 모드(Operational Mode)를 명세하기 위해 사용된다. 이러한 하이브리드 시스템은 항공, 철도, 전장 시스템과 같은 안전 필수 시스템(Safety-Critical System)을 명세하기 위해 사용되므로 정형 검증(Formal Verification)과 같은 다양한 검증 기법을 적용하여 고신뢰성(Dependability)을 확보하는 것이 필수적이라 할 수 있다.

정형 검증은 시스템의 주요 행위를 담은 정형 모델을 대상으로 수학·논리를 사용하여 검증 속성을 만족하는지 확인하는 검증 기법이다. 정형 검증을 적용하기 위해서는 대상 시스템의 특징과 검증하고자 하는 검증 속성에 맞는 명세·검증 기법을 선택해서 적용해야하는 특징이 있다. 대표적인 정형 검증 기법으로는 모델 체킹(Model Checking)[2]이 있다. 모델 체킹은 모델 체커(Model Checker)라는 자동 검증 도구를 이용하여 모델의 모든 상태를 자동으로 분석하는 기법으로, 검증된 시스템의 에러가 없음을 확인할 수 있는 장점이 있지만 비용이 널리 사용되는 검증 기법인 테스팅 및 시뮬레이션에 비해 비용이 높고 적용에 제약이 있다는 단점이 있다. 테스팅 및 시뮬레이션으로 검증을 수행한 후 시스템의 컨트롤러, 스케줄러와 같은 중요한 부분에 정형 검증을 추가로 적용한다면 두 기법을 상호 보완하여 적용한다면 효과적으로 높은 수준의 고신뢰성을 확보 할 수 있을 것이다.

하이브리드 시스템을 명세 및 검증하기 위해 널리 사용되는 모델은 하이브리드 오토마타(Hybrid Automata)[3]이다. 기존의 이산시스템을 명세하기 위한 오토마타와는 다르게 모든 변수가 실수 값(Real Value)을 가지며 상미분방정식(ODE, Ordinary Differential Equation)으로 변수 값의 시간에 따른 변화율을 나타내는 큰 차이가 있다. 하이브리드 오토마타는 무한한 수의 상태를 가지며, 강한 제약사항이 없이는 모델의 초기상태에서 특정 상태까지 도달하는지 여부(Reachability Problem)를 확신할 수 없는 문제가 있어[4] 기존의 이산 시스템을 위한 모델 체킹 기법을 활용하는 것은 불가능하다. 많은 연구들이 모델 행위의 근사치(Approximation)를 구하는 방식의 최적화 알고리즘과 지원 도구들을 제안해 왔으며, 모델 체킹 알고리즘 외에도 시뮬레이션, 명제 증명(Theorem Proving)과 같은 기법으로 하이브리드 시스템을 검증하려는 연구들도 수행되고 있다.

본 논문에서는 하이브리드 시스템에 대한 소개와 이를 모델링 및 검증하기 위한 연구들에 대해 소개한다. 논문의 구성은 다음과 같다. 2장에서는 하이브리드 시스템을 명세하기위해 사용되는 하이브리드 오토마타를 소개한다. 3장에서는 하이브리드 오토마타의 reachability analysis를 위한 기본적인 기법들을 설명한다. 4장에서는 하이브리드 시스템을 검증하기 위한 기법들을 관련 연구로서 소개하며, 5장에서 결론을 내린다.

2. 배경 지식

2.1 하이브리드 오토마타

하이브리드 시스템은 이산적인 시스템과 연속적인 시스템이 결합된 시스템으로, 주로 물리적인 환경 안에 존재하는 소프트웨어로 제어되는 이산적인 제어기로 구성된다. <그림 1> 은 하이브리드 시스템 예제로 자주 사용 되는 온도계 시스템(*Thermostat System* [5])을 보여주고 있다. 온도계 시스템은 온도계로 방의 온도를 측정하여 일정 수준이하로 온도가 떨어지면 자동으로 히터를 켜서 온도를 올리고, 히터에 의해 온도가 어느 정도 높아지면 히터를 끄는 시스템이다. 이 시스템에서 방의 온도를 물리적인 환경, 온도에 따른 스위치의 상태 변화를 제어기의 이산 상태 변화로 볼 수 있다.

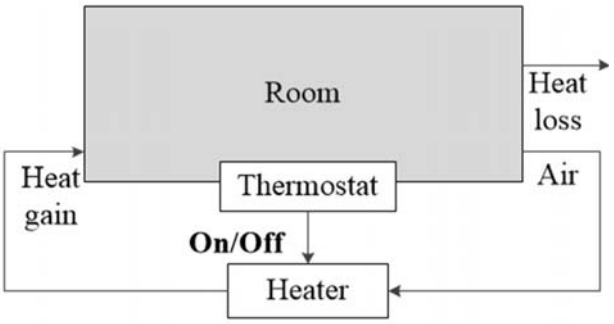


그림 1 온도계 시스템의 개념도

하이브리드 오토마타는 하이브리드 시스템을 명세하기 위한 유한 상태 기계(*Finite State Machine*)이다. 하이브리드 시스템의 이산 상태변화는 하이브리드 오토마타의 *control mode*로 표현되며, 물리적인 환경의 연속적인 변화는 실수 값을 갖는 변수들로 표현된다. 변수들의 행위(*Behavior*)는 각 *control mode*가 갖는 상미분방정식들로 정의된다.

<그림 2>는 <그림 1>의 온도계 시스템을 모델링한 하이브리드 오토마타를 보여주고 있다. 온도계 시스템은 스위치의 상태에 따라 *on*과 *off*의 두 가지 *control mode*를 가진다 할 수 있다. 각각의 *control mode*에서 모델이 한 *control mode*에 머무르기 위한 조건인 *invariant*와 방의 온도를 나타내는 변수 x의 연속적인 변화를 표현하는 *flow condition*이 정의 된다. <그림 2>의 모델에서 초기 *control mode*는 *on*이며 x의 초기 값은 2로 정의 된다. 모델이 *on*에 머물기 위해서는 *invariant*인 $1 \le x \le 3$를 만족해야하며, x의 값은 *flow condition*인 $\dot{x} = -x + 5$에 따라 증가하게 된다. x가 3까지 오르면 *jump condition*인 $x = 3$을 만족하여 *off*로 이동하게 되고, *off*의 *flow condition*인 $\dot{x} = -x$에 따라 x의 값이 감소하여 1 까지 떨어지면 모델이 *on*으로 돌아오게 된다.

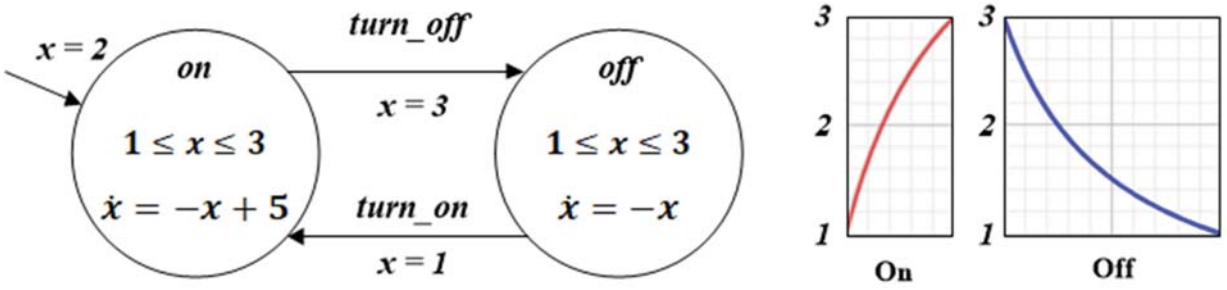
하이브리드 오토마타로 명세된 시스템을 분석하기 위해서는 모델이 가질 수 있는 상태를 분석할 필요가 있으나, 하이브리드 오토마타의 변수는 실수값을 가지기에 이산 시스템과 같이 상태를 변수가 가질 수 있는 모든 값으로 표현하는 것은 불가능하다. 따라서 하이브리드 오토마타에서 상태는 변수들의 값을 범위로 나타낸 집합 형태로 표현하게 된다. 이것을 *region*이라하며, *control mode*의 이름과 *control mode*에서 가질 수 있는 변수들의 값의 범위로 표현된다. 예를 들어, <그림 2> 모델 행위변화에 따라 $\{(on,\ x = 2),\ (on,\ 2 < x \le 3),\ (off,\ 1 \le x \le 3),\ (on,\ 1 \le x \le 3)\}$와 같이 *region*들을 정의해 볼 수 있다. 이 후, 논문에서 언급하는 상태 혹은 상태 집합은 *region*을 말한다.

2.2 Reachability Problem과 하이브리드 시스템 모델 체킹 알고리즘

모델 체커들은 모델의 상태들을 분석하여 모델이 요구사항, 혹은 검증 속성을 만족하는지 확인한다. 명세 가능한 요구사항들은 기법에 따라 다양하지만 크게 나누어 보면 다음과 같이 나누어 볼 수 있다.

- 모델이 특정 상태에 도달 할 수 있는가? (*Reachability Property*)
- 특정 조건하에, 항상 (원하지 않는) 특정 상태가 발생하지 않는가? (*Safety Property*)
- 특정 조건하에, 항상 (원하는) 특정 상태가 발생하는가? (*Liveness Property*)

가장 기본이 되는 검증 속성은 *rechability property*라고 할 수 있으며, 모델 체커가 초기 상태에서 대상 상태를 도달 가능한지 확인하는 분석 과정을 *reachability analysis*라고 한다. *Reachability property* 외의 속성들은 *reachability analysis* 알고리즘을 응용하여 확인할 수 있다. 예를 들어, *safety property*는 원하지 않는 상태에 대한 도달 가능한 상태를 찾아내는 것으로 확인할 수 있으며, *liveness property*의 경우, 조건이 만족되었을 때 도달 가능한 상태를 모두 찾아내어 확인 할 수 있다. 즉, *reachability analysis*가 불가능 하다면 그 외의 속성



(a) 온도계 시스템을 모델링한 하이브리드 오토마타 (b) 스위치 상태에 따른 온도변화

그림 2 온도계 시스템을 모델링한 하이브리드 오토마타[5]

들도 만족여부를 확인하기 어려우며, 모델의 초기 상태에서 특정 상태까지 도달할 수 있는지 판단하는 이론적인 문제를 *reachability problem*이라고 한다.

이산 시스템의 모델 체킹은 상태 폭발 문제가 발생하여 분석할 수 있는 모델의 규모가 한정되는 문제가 있지만, 이론적으로는 무한한 메모리와 시간이 주어진다면 모든 상태에 대해 *rechability analysis*가 가능하다. 반면에 하이브리드 시스템은 이론적으로도 도달 가능한 상태를 확인할 수 없을 수 있다는 것이 증명되어[4], 분석하는 알고리즘도 모델의 모든 상태를 분석하는 것이 아니라 모델이 가질 수 있는 상태의 근사치(*Approximation*)를 계산하여 가능한 수준만큼 *reachability analysis*를 수행하는 방향으로 연구가 진행되어 왔다.

2.3 하이브리드 오토마타의 종류

하이브리드 오토마타는 다양한 변형들이 제안되어 왔으며, 모델의 dynamics에 대한 제약사항이 강한 순으로 *timed*, *linear*, (*piecewise*) *affine*, *non-linear* 등으로 분류해 볼 수 있다. *Timed automata* [6]는 시간의 흐름을 표현하는 *clock* 변수만 실수 값을 가지고 다른 변수들은 이산 시스템과 같이 정수 값을 갖는 특징을 가지며 *UPPAAL* [7], *TIMES* [8]와 같은 모델 체커를 이용하여 모델 체킹 및 반례(*Counterexample*) 생성이 가능하다. *Linear hybrid automata* [9]와 *affine hybrid automata* [10]는 공통적으로 선형 상미분방정식을 사용하여 변수의 변화율을 표현 하며, *linear hybrid automata*는 *HyTech* [5], *PHAVer* [10]와 같은 도구로, *affine hybrid automata*의 경우에는 *SpaceEx* [11] 등으로 오차 범위를 갖는 *reachability analysis*를 수행할 수 있다. *Non-linear hybrid automata*는 제약사항이 없는 하이브리드 오토마타로 분석해야할 모델의 복잡도가 매우 높아, *Flow** [12]와 같이 *reachability analysis*를 위한 도구들이 제안되어 왔으나, dynamics가 매우 복잡하여 전문가가 직접 결정해 주어야하는 설정 값에 의해 *reachability analysis*의 정확성이 달라지는 문제가 있다[13, 14]. *Reachability analysis*의 오차에 대한 내용은 다음 장에서 다룬다.

3. 하이브리드 시스템의 Reachability Analysis

하이브리드 시스템을 모델 체킹하기 위해서는 많은 이슈들이 남아 있으며[15], 다양한 연구들이 각각 하이브리드 오토마타의 한 종류를 대상으로 최적화 알고리즘들을 제시하였다. 이번 장에서는 많은 *reachability analysis* 알고리즘들에서 공통적으로 적용되는 *flow-pipe construction*을 소개 한다.

3.1 Flow-pipe construction

하이브리드 오토마타를 검증하는 알고리즘 및 지원 도구들은 하이브리드 오토마타로 명세된 모델을 분석하기 위해 *flow-pipe construction* [16]을 수행한다. *Flow-pipe*는 한 범위시간(*Time-Step*) 동안의 도달 가능한 상태 집합(*Region*)을 이야기 하며, 모델의 초기 상태에서 반복해서 *flow-pipe*를 계산하여 주어진 자원으로 가능한 많은 수의 도달 가능한 상태 집합들을 찾아낸다. 그리고 <그림 3>과 같이 찾아낸 집합 중 대상 *region*과 교집합이 있는 부분을 찾아내면 모델이 대상 *region*에 도달 가능하다고 할 수 있으며, 최대한 도달 가능한 *region*을 찾아내었는데 대상 *region*과 교집합이 있는 부분을 찾지 못하면 도달하지 않는다가 아니라, 결정할 수 없다(*Undecidable*)이라고 결론을 내린다. 이는 하이브리드 시스템을 위한 *reachability analysis*가 근본적으로 모든 상태를 찾아 낼 수 없어 가능한 만큼 분석을 한다는 점에서 근거한다.

3.2 Flow-pipe Approximation

Flow-pipe를 계산하기 위해서는 이전 시간의 모델의 상태 집합과, 도구나 사용자에 의해 지정되는 범위

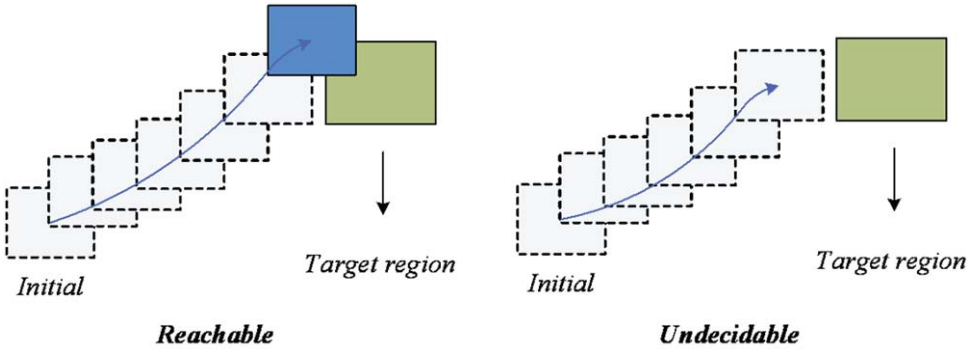
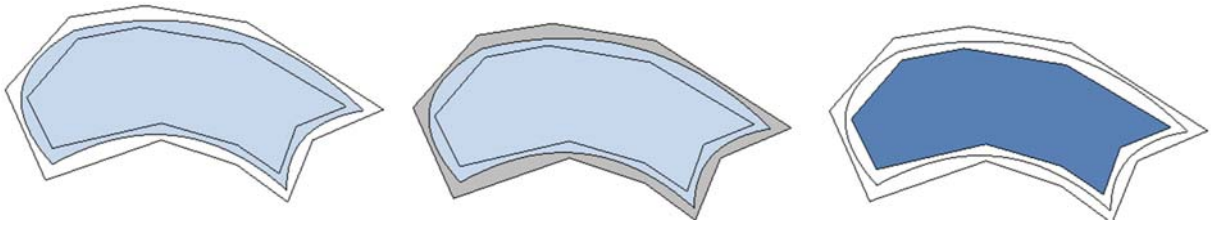


그림 3 Reachability analysis 결과의 도달 가능한(Reachable) 경우와 결정 할 수 없는(Undecidable) 경우



(a) Reachable states (b) Over-approximated states (c) Under-approximated states

그림 4 Flow-pipe approximation의 개념도

시간, 변수의 변화율을 명세한 상미분방정식을 활용한다. 제약사항이 강한 일부 하이브리드 오토마타는 *flow-pipe*를 정확하게 계산할 수 있지만[9, 16] 대부분의 경우에는 어렵다. 따라서 *flow-pipe*를 계산할 때 정확한 상태 범위가 아닌 근사치(*Approximation*) 활용하게 된다. <그림 4>는 모델의 실제 도달 가능한 상태와 *approximation*이 된 상태간의 차이를 보여주고 있다. *Over-approximation* 방식과 *under-approximation* 방식이 있는데, 두 방식은 각각 실제 상태보다 더 넓은, 좁은 범위로 *flow-pipe*를 계산한다.

*Reachability analysis*에서는 일반적으로 over-approximation 방식이 활용되고 있다. 각 기법들은 목적에 맞게 polytope [17, 18], *zono-tope* [19], *support function* [20], *Taylor model* [21] 등의 상태 표현을 활용하여 실제 도달 가능한 상태를 포함하는 convex hull을 생성하고 이를 *flow-pipe*로 활용한다. 따라서 *over-approximation* 방식으로 수행된 *reachability analysis*에는 근본적으로 실제로는 모델이 도달하지 않는 오차 상태 범위가 포함되게 되며, 오차 범위를 줄일수록 *reachability analysis*의 정확성(Correctness)는 높아지지만 효율(Efficiency)은 낮아지는 trade-off가 존재하게 된다.

이러한 *over-approximation*의 특징을 고려해 보면, *reachability analysis*의 결과가 대상 상태에 '도달 가능'으로 나와도 분석에 오차 범위가 들어가기 때문에 실제로 모델이 도달했는지는 확신할 수 없다는 것을 알 수 있다. 다시 말하면 over-approximation을 이용한 reachability analysis에서는 reachability property를 보는 것은 어렵고 분석결과를 바탕으로 시뮬레이션 등의 다른 방법을 이용하여 실제 도달여부를 확인하는 것을 생각해 볼 수 있을 것이다. 반면에, reachability analysis의 대상 상태를 원하지 않는 상태로 지정한다면 우리는 한정적으로 *safety property*를 확인할 수 있다는 것을 알 수 있다. 실제 도달 가능한 범위와 오차 범위를 포함해서 분석한 결과가 '도달 가능하지 않다'로 나왔다면 실제로도 모델이 원하지 않는 상태에 도달하지 않을 것이며, 분석되지 않은 범위에 대해서는 확신할 수 없다(*Undecidable*). 하이브리드 시스템 reachabiltiy analysis를 수행하는 도구들은 분석된 region들의 집합을 분석결과와 함께 출력해 주고 있다.

3.3 시뮬레이션과 Reachability Analysis 검증 결과 비교

하이브리드 시스템을 분석하는데 가장 널리 사용되는 기법은 시뮬레이션이라고 할 수 있으며, 모델 체킹의 관점과는 다른 방향성을 취하고 있다. 시뮬레이션의 경우, 모델이 가질 수 있는 전체 행위중 하나의 경우의 수를 고려하여 초기 상태에서 특정 시간, 혹은 상태에 도달할 때까지의 경로를 여러개 생성해낸다. 반면에 모델 체킹의 관점에서는 초기상태에서부터 모든 경우의 수를 고려하여 상태 범위를 주어진 자원으로 가능한 많이 찾아내는 방식으로 볼 수 있다.

전체 상태를 분석할 수 없는 하이브리드 시스템의 특징을 고려해 보았을 때, 시뮬레이션으로는 *reachability property*를 확인할 수 있지만 *safety property*를 확인하는 것은 어려울 수 있으며, 모델 체킹 방식으로는 분석된 범위, 혹은 모델의 시작 시간부터 특정 시간까지에 대한 부분적인 *safety property*를 보기에 적합한 대신 *flow-pipe construction*에 *over-approximation*을 수행하는 현황을 고려해 보았을 때 실제로 모델이 대상 상태에 도달하는지 *reachability property*를 확인하기에는 부적합할 수 있다는 것을 알 수 있다. 따라서 두 기법을 상호 보완하여 적용할 필요가 있다.

4. 관련 연구

하이브리드 시스템을 검증하기 위한 *reachability analysis*외의 연구들도 꾸준히 제안되어 왔다. 대표적으로는 *monitoring*이 있다[22]. *Monitoring*은 앞서 언급한 시뮬레이션을 이용한 방법으로 모델이 검증 속성을 만족하는지 확인하기 위해서 일정 수 이상의 시나리오들을 생성하고, 생성된 시나리오에 의한 시뮬레이션 결과가 속성을 모두 만족하는지 확인한다. 기본적으로 생성한 시나리오 및 시나리오로 지나는 모델의 path가 많을수록 상대적으로 더 높은 신뢰도를

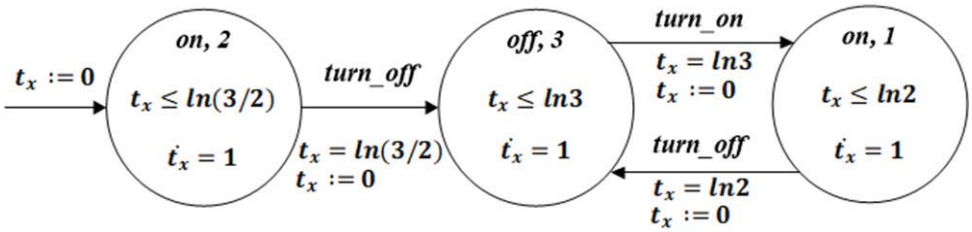


그림 5 *Thermostat system*의 하이브리드 오토마타를 변환한 timed automata [5]

표 1 하이브리드 시스템을 명세, 분석, 검증하기 위한 도구들 [26]

이름	목적	입출력	검증 기법
CHARON	Modeling, Simulation	CHARON language	None
CheckMate[a]	Verification	Autonomous liner hybrid automata	Rectangular polytopes automation
d/dt	Verification	Linear hybrid automata	Over-approximation
Ellopsodial ToolBox[a]	Verification	Controlled linear hybrid system	Parallellotope method
GBTa	Computation	Polytope, ellipsoid	Convex hull determination
HSIF	Modeling, simulation	Network (collection of hybrid automata)	None
HSolver	Verification	Input hybrid system	Constraint propagation[b]
HyTech	Verification	Linear hybrid automata	Quantifier elimination, validity checking
HyVisual	Modeling	Embedded systems	None
KeYmaera	Verification	Differential dynamic logic	Symbolic decomposition[b]
Level Set ToolBox[a]	Verification	Partial differential equation	Hamilton-Jacobi equation solutions
MATISSEa	Verification	Transition system	Bisimulation
MultiParametic ToolBox[a]	Simulation, verification	Piecewise affine systems	Linear / quadratic programming solver
PHAVer	Verification	Linear I/O hybrid automata	
Ptolemy II	Modeling, simulation	Embedded systems (contains hybrid system)	Non-hybrid verifier
SHIFT	Modeling, translation	SHIFT language	None
SpaceEx	Verification	Linear, affine hybrid automata	Time-step flow-pipe computation
STeP	Verification	Real-time system	Invariant generation[b]
Flow*	Verification	Non-linear hybrid automata	Flow-pipe construction

a: Requiring Matlab, b: Theorem proving

가진다고 할 수 있으며, *monitoring*과 관련된 연구에서는 테스팅에서 fault를 찾아낼 수 있는 테스트 케이스를 만들어내는 연구들과 유사하게 시뮬레이션 시나리오의 *robustness*를 향상 시켜[23] fault를 효율적으로 찾아 낼 수 있는 방향으로 연구가 진행되고 있다.

HySAT [24]과 같은 SMT (SAT-Modulo-Theories) solver를 이용하여 하이브리드 오토마타의 상미분 방정식의 해를 찾아내거나, *theorem proving*을 적용하는 방법도 생각해 볼 수 있다. *Theorem proving*은 이론적으로 거의 모든 증명을 할 수 있는 매우 강력한 기법이나, 기법을 적용하는 전문가가 많은 경험과 지식을 가지고 직접 분석하는 방식으로, 적용하기 매우 어려운 단점이 있다.

또 다른 방법으로는 모델을 변환하여 모델 체킹을 수행하는 방법이 있다[25]. 하이브리드 오토마타의 많은 변형들은 dynamics가 단순할수록 분석 난이도는 줄어든다고 할 수 있다. Dynamics가 다를 경우 완벽한 변환은 어려울 수 있어, 가정과 제약사항을 가지고 모델을 변환하여 검증한다. 특히, *timed automata*의 경우에는 *UPPAAL*과 같은 도구로 이산 시스템의 모델 체킹과 유사한 수준의 다양한 검증 속성을 확인할 수 있다. <그림 5>은 온도계 시스템을 나타낸 모델인

<그림 2>을 전문가가 직접 *clock* 변수를 기준으로 *timed automata*로 변환한 모델이다[5]. 변환 전과 모델 행위가 동일하지는 않으나, 어느 시점에 모델이 특정 *control mode*에 머무는지 확인할 수 있다.

우리는 이전 연구에서[26] 하이브리드 시스템을 명세하고 검증하기 위한 도구들을 조사하고, ECML [27] 이라는 시뮬레이션 언어로 명세된 모델을 제약사항과 가정을 갖는 변환을 통하여 검증하는 기법을 제안하였다. ECML 모델은 *linear hybrid automata*로 자동 변환되었으며, 이를 *SpaceEx*를 이용하여 *reachability analysis*를 수행하였다. <표 1>은 하이브리드 시스템을 명세하고 분석 및 검증하기 위한 도구들을 조사한 것이다. 각각의 도구들이 명세 기법에 따라 다른 검증 기법을 활용하는 것을 확인할 수 있다.

5. 결론 및 향후 연구

하이브리드 시스템은 연속 시스템과 이산 시스템이 융합된 시스템으로 안전 필수 시스템에 많이 활용되므로 다양한 검증 기법 적용을 통한 고신뢰성 확보가 필수적이다. 하이브리드 시스템을 정형 검증하기 위해서는 기존의 이산 시스템의 관점뿐만 아니라 연속 시스템의 관점 또한 필요하다. 하이브리드 시스템은 다양한 종류의 하이브리드 오토마타로 모델링 될 수 있으며, 많은 연구들이 *over-approximation*을 통한 *reachability analysis*를 수행하고 있다. 하이브리드 오토마타의 *reachability problem*은 결정할 수 없는 문제이기에 *reachability analysis* 알고리즘의 최적화만으로는 이산 시스템과 같은 수준의 모델 체킹을 기대하기는 어렵다.

하이브리드 시스템 정형 검증은 적용이 어렵고 비용이 큰 특징을 고려하여, 효율적인 검증을 위해 시뮬레이션을 기본적으로 적용하고 스케줄러 등의 중요한 부분에 추가로 정형 검증을 적용하는 것을 고려해 볼 수 있을 것이다. 우리는 하이브리드 시스템 정형 검증의 이슈들을 보완하는 연구를 수행하고 있으며 시뮬레이션, 정형 검증을 상호 보완하는 검증 환경 구축을 계획하고 있다.

참고문헌

[1] P. J. Antsaklis, J. A. Stiver, M. D. Lemmon, "Interface and Controller Design for Hybrid Control Systems," Hybrid Systems II, LNCS 999, pp. 462-492, 1995.

[2] E. M. Clarke, E. A. Emorson, A. P. Sistla, “Automatic Verification of Finite-State Concurrent Systems using Temporal Logic Specifications”, ACM Trans. Programming Languages and Systems, vol. 8, no. 2, pp. 244-263, 1986.

[3] R. Alur, C. Courcoubetis, T. A. Henzinger, P. -H. Ho, "Hybrid Automata: An Algorithmic Approach to the Specification and Verification of Hybrid Systems," Hybrid Systems, pp. 209-229, 1993.

[4] T. A. Henzinger, P. W. Kopke, A. Puri, P. Varaiya, "What's Decidable about Hybrid Automata?," Proceedings of the twenty-seventh annual ACM symposium on theory of computing, pp. 373-382, 1995.

[5] T. A. Henzinger, P.-H. Ho, H. Wong-Toi, "HyTech: A Model Checker for Hybrid Systems," International Journal on Software Tools for Technology Transfer, vol. 1, no. 1-2, pp. 110-122, 1997.

[6] R. Alur, D. L. Dill, "A Theory of Timed Automata," Theoretical Computer Science, vol. 126, no. 2, pp. 183-235, 1994.

[7] K. G. Larsen, P. Pettersson, W. Yi, "UPPAAL in a Nutshell," International Journal on Software Tools for Technology Transfer (STTT), vol. 1, no. 1, pp. 134-152, 1997.

[8] T. Amnell, E. Fersman, L. Mokrushin, P. Pettersson, W. Yi, "Times: A Tool for Schedulability Analysis and Code Generation of Real-Time Systems," International Conference on Formal Modeling and Analysis of Timed Systems, pp. 60-72, 2003.

[9] R. Alur, T. A. Henzinger, P.-H. Ho, "Automatic Symbolic Verification of Embedded Systems," IEEE Transactions on Software Engineering, vol. 22, no. 3, pp. 181-201, 1996.

[10] G. Frehse, "PHAVer: Algorithmic Verification of Hybrid Systems Past HyTech," International Journal on Software Tools for Technology Transfer (STTT), vol. 10, no. 3, pp. 263-279, 2008.

[11] G. Frehse, C. Le Guernic, A. Donzé, S. Cotton, R. Ray, O. Lebeltel, R. Ripado, A. Girard, T. Dang, O. Maler, "SpaceEx: Scalable Verification of Hybrid Systems," Proc. 23rd International Conference on Computer Aided Verification (CAV), pp. 379-395, 2011.

[12] X. Chen, E. Ábrahám, S. Sankaranarayanan, “Flow*: An Analyzer for Non-linear Hybrid Systems,” International Conference on Computer Aided Verification, pp. 258-263, 2013.

[13] M. Althoff, S. Bak, D. Cattaruzza, X. Chen, G. Freshe, R. Ray, S. Schupp, "Arch-comp17 Category Report:

Continuous and Hybrid Systems with Linear Continuous Dynamics," ARCH17, 4th International Workshop on Applied Verification of Continuous and Hybrid Systems, pp. 143-159, 2017.
[14] X. Chen, M. Althoff, F. Immler, "Arch-comp17 Category Report: Continuous Systems with Nonlinear Dynamics," ARCH17, 4th International Workshop on Applied Verification of Continuous and Hybrid Systems, pp. 160-169, 2017.
[15] S. Schupp, E. Ábrahám, X. Chen, I. B. Makhlouf, G. Freshe, S. Sankaranarayanan, S. Kowaleski, "Current Challenges in the Verification of Hybrid Systems," International Workshop on Design, Modeling, and Evaluation of Cyber Physical Systems, pp. 8-24, 2015.
[16] A. Chuntinan, B. H. Krogh, "Computing Approximating Automatata for a Class of Linear Hybrid Systems," International Hybrid Systems Workshop, pp. 16-37, 1997.
[17] R. Alur, T. Dang, F. Ivančić, "Predicate Abstraction for Reachability Analysis of Hybrid Systems," ACT Transactions on Embedded Computing Systems (TECS), vol. 5, no. 1, pp. 152-199, 2006.
[18] S. Sankaranarayanan, T. Dang, F. Ivančić, "Symbolic Model Checking of Hybrid Systems using Template Polyhedral," Tools and Algorithms for the Construction and Analysis of Systems, pp. 188-202, 2008.
[19] A. Girard, "Reachability of Uncertain Linear Systems using Zonotopes," International Workshop on Hybrid Systems: Computation and Control, pp. 291-305, 2005.
[20] G. Frehse, R. Ray, "Flow-pipe-guard Intersection for Reachability Computations with Support Functions," IFAC Proceedings, vol. 45, no. 9, pp. 94-101, 2012.
[21] X. Chen, E. Abraham, S. Sankaranarayanan, "Taylor Model Flowpipe Construction for Non-linear Hybrid Systems," Real-Time Systems Symposium (RTSS), 2012 IEEE 33rd, pp. 183-192, 2012.
[22] O. Maler, D. Nickovic, "Monitoring Temporal Properties of Continuous Signals," Formal Techniques, Modeling and Analysis of Timed and Fault-Tolerant Systmes, pp. 152-166, 2004.
[23] A. Donzé, T. Ferrere, O. Maler, "Efficient Robust Monitoring for STL," International Conference on Computer Aided Verification, pp. 264-279, 2013.
[24] M. Fränzle, C. Herde, "Hysat: An Efficient Proof Engine for Bounded Model Checking of Hybrid Systems," Formal Methods in System Design, vol. 30, no. 3, pp. 179-198, 2007.
[25] T. A. Henzinger, H. Wong-Toi, "Linear Phase-portrait Approximations for Nonlinear Hybrid Systems," International Hybrid Systems Workshop, pp. 377-388, 1995.
[26] S. Yoon, J. Yoo, "Formal Verification of ECML Hybrid Models with SpaceEx," Information and Software Technology, vol. 92, pp. 121-144, 2017.
[27] 윤상현, 전인걸, 김원태, 조재연, 유준범, “하이브리드 시스템을 명세하기 위한 ETRI CPS 모델링 언어”, 정보과학회 논문지: 시스템 및 이론, 제42권, 제7호, pp.823-833, 2015.

약 력

윤 상 현

2010 건국대학교 컴퓨터공학과 졸업 (학사)
2012 건국대학교 컴퓨터공학과 졸업 (석사)
2018 건국대학교 컴퓨터공학과 졸업 (박사)
2019~현재 국방과학연구소 선임연구원
관심분야: 소프트웨어 공학, 하이브리드 시스템, 정형 기법
Email: pctkdgus@konkuk.ac.kr

유 준 범

2005 KAIST 전자전산학과 전산학전공 졸업 (박사).
2008 삼성 전자주식회사 통신연구소 책임연구원.
2008~현재 건국대학교 컴퓨터공학부 교수.
관심분야: 소프트웨어 공학, 안전성 분석, 정형 기법
Email: jbyoo@konkuk.ac.kr