

하이브리드 시스템 검증에 대한 소개

Dependable Software Laboratory

윤상현

2016. 05. 19

Software Verification

- Software verification (검증)
 - Software가 specification (명세)에 맞게 잘 개발 되었는지 확인하는 과정
 - 소프트웨어 개발에 사용되는 다양한 요구사항 문서, 디자인 모델, ...
- Software verification techniques
 - Testing
 - Simulation
 - Model checking
 - Theorem proving
 - ...

Software Testing

- (Functional) Software Testing
 - 다양한 목적에 따라 수행되는 대표적인 검증 기법
 - 실행 가능한 프로그램 또는 모델에 입력을 넣어 예상되는 출력이 나오는지 확인

테스트 케이스

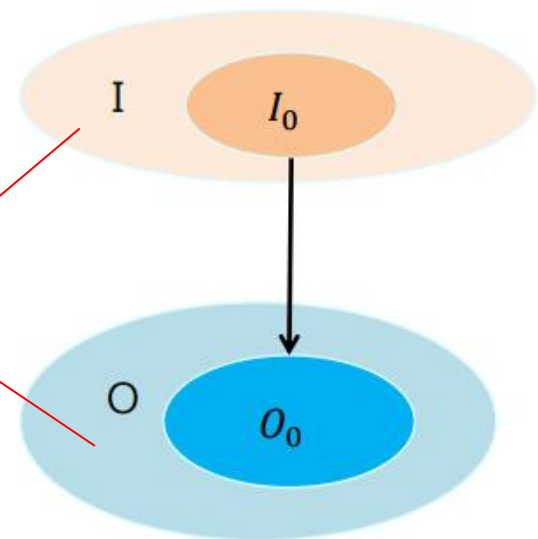
이를 생성하기 위한
많은 기법들이 있음



Why Testing Isn't Good Enough

- Testing can be used to show the presence of errors but not their absence.
(by Edsger Dijkstra)
 - Test result is all TRUE -> no error?
 - No errors about the input data

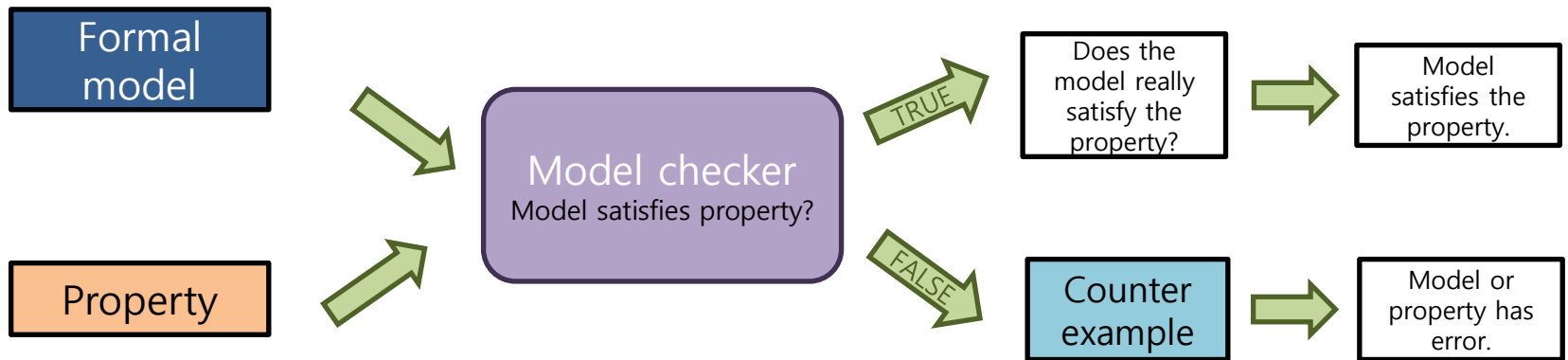
테스팅 해보지 않은 부분에
대해 확신을 가질 수 없으며
Full coverage를 달성하기 어렵다.



Model checking?

- 모델체킹이란?

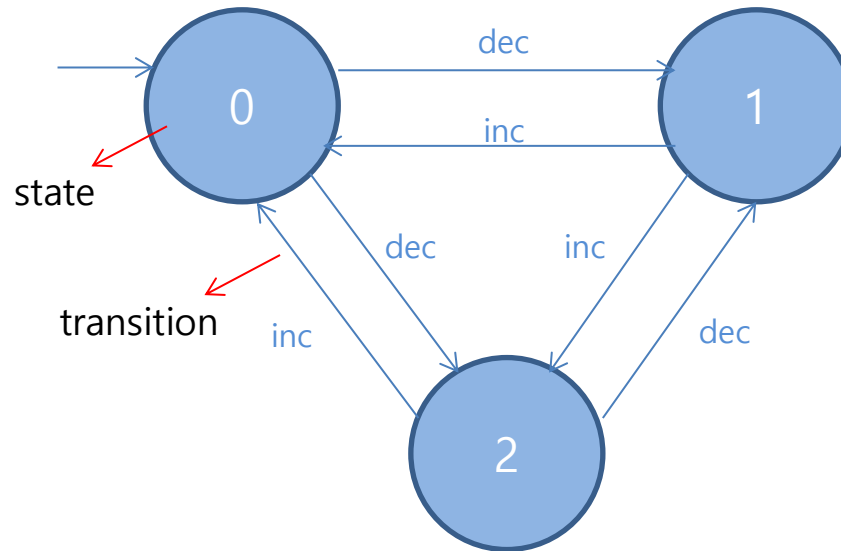
- 검증 도구가 가능한 모든 경우의 수를 계산하여 분석하기 때문에 예상하기 어려운 에러를 찾아낼 수 있음
- 테스트보다 비용(시간+자원)이 많이 필요하게 되며 주로 safety-critical system과 같은 critical system의 검증에 활용된다. ↳ 원자력 발전소, 철도, 항공,...
- Formal Model + Verification Requirements (property) + Model checker (verification tool)
↳ Formal definition + formal semantics를 가진 모델
주로 오토마타의 형태로 구성되어 있음



Model checker는 전문가를 도와주는 도구일 뿐, 사람이 직접 분석 & 판단해야 하는 부분이 많다.

Automata

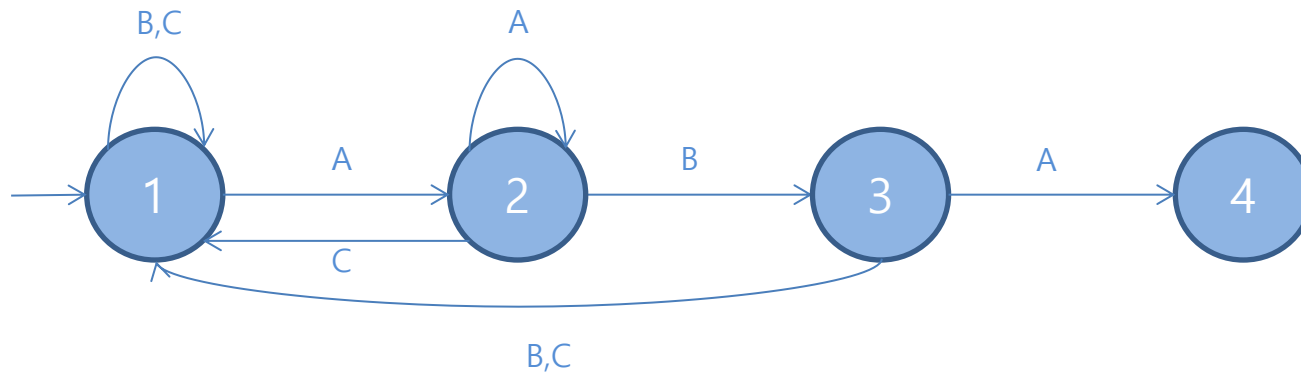
- (Finite) Automata
 - A machine evolving from one state to another under action of transitions
 - State는 시스템의 내부적인 상태, transition은 상태의 변화를 나타낸다.
- 3 counter example
 - (0~2)까지 숫자를 세는 카운터
 - inc 와 dec 입력 (이벤트)를 가짐
 - 숫자 증가(inc), 숫자 감소(dec)



Door lock example

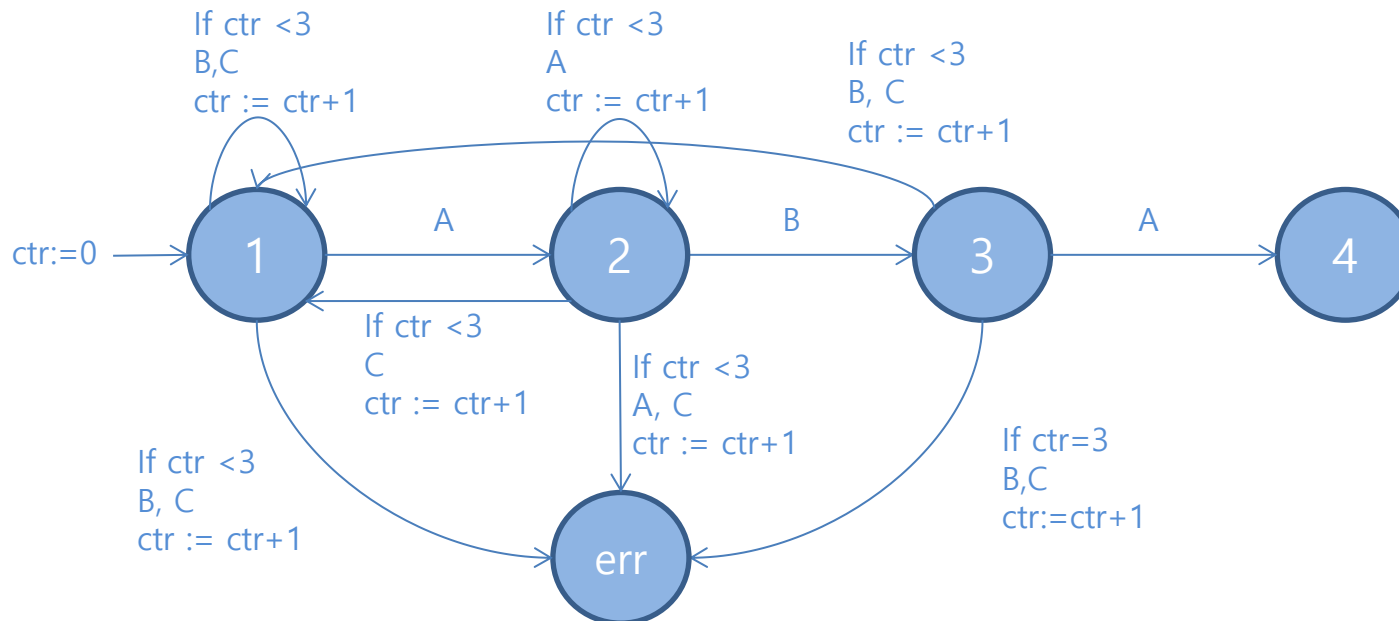
- A digicode door lock example

- 요구사항
- 사무실 문을 여는 것을 제어하는 시스템
 - 문을 열기 위해서 키를 정해진 순서 대로 입력해야 하며 잘못된 입력을 하면 다시 처음부터 입력해야 한다.
 - 3 keys (A, B, C)
 - Correct key sequence : ABA



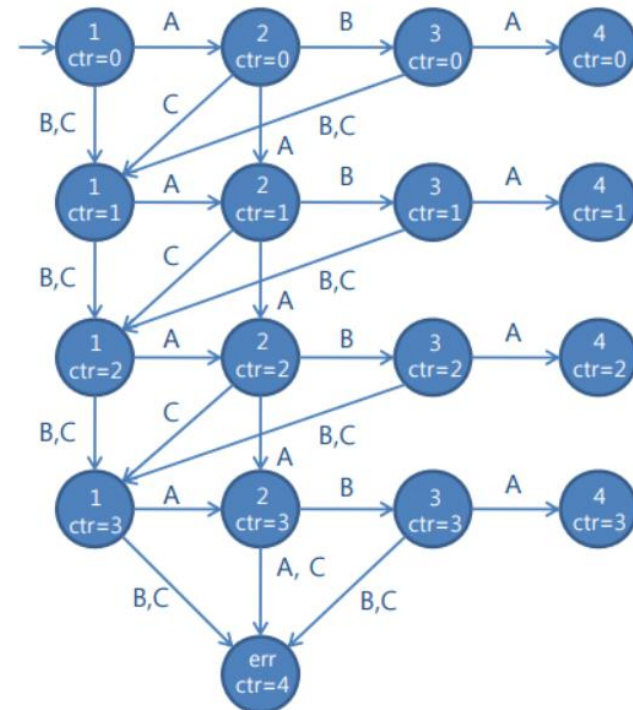
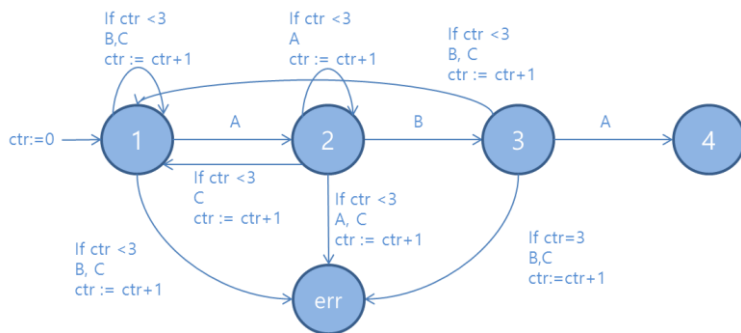
Door lock example

- A digicode door lock with counter
 - 사무실 문을 여는 것을 제어하는 시스템
 - 문을 열기 위해서 키를 정해진 순서 대로 입력해야 하며 잘못된 입력을 하면 다시 처음부터 입력해야 한다.
 - 3 keys (A, B, C)
 - Correct key sequence : ABA
 - **입력을 세 번 틀리면 시스템을 멈춘다.** ← 추가된 요구사항



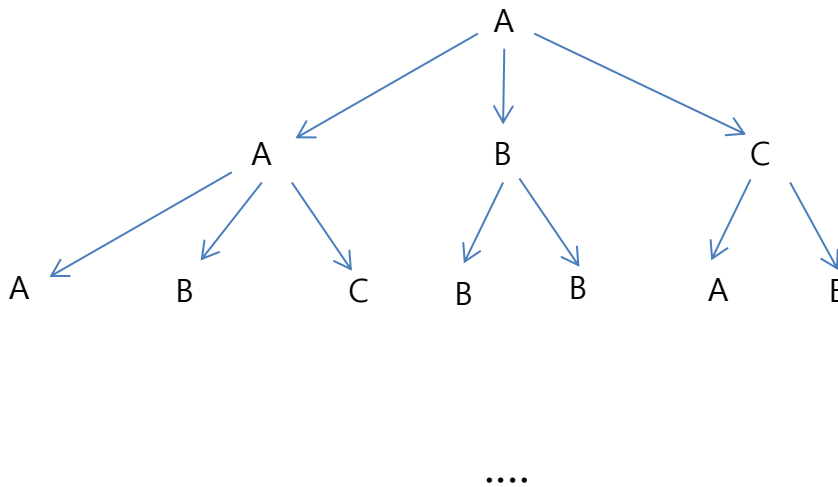
Door lock example

- 모델 체커는 내부적으로 입력 받은 모델을 unfolding하여 분석한다.
 - 실제 예제에서는 모델간의 message passing 등을 이용하여 가능한 state의 개수를 줄인다.
- State explosion problem
 - 분석 할 모델의 상태가 너무 많아져 분석을 할 수 없는 문제
 - 모델에서 사용하는 변수의 개수가 많고 범위가 클수록 특히 많이 발생하게 된다.



Door lock example

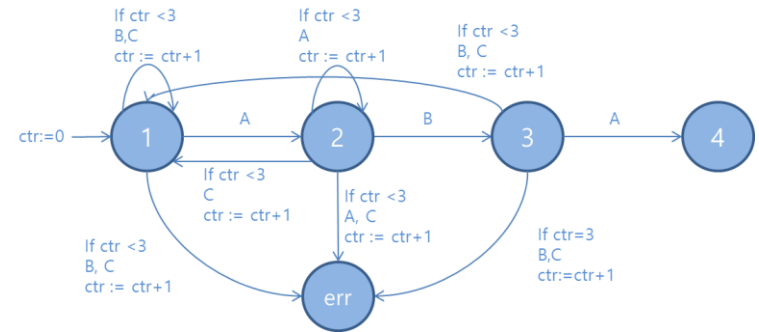
- Execution
 - A sequence of states describing one possible evolution of the system
 - e.g.) ABA, ABBA, AAAABA, ...
- Execution tree
 - A set of all possible executions of the system in the form of a tree



Door lock example

- Temporal Property
 - 대표적인 예 : CTL, LTL
 - 모델이 요구사항을 만족하는지 확인하기 위해 사용한다.
 - 모델의 execution을 고려하여 작성할 필요가 있다.

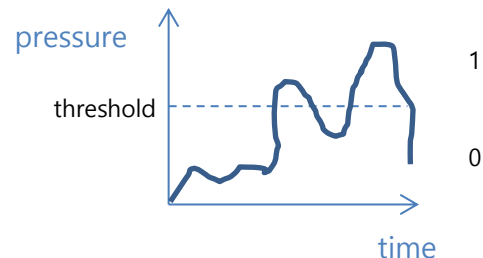
- Property를 이용한 검증 (CTL) 예제
 - 시스템은 데드락 상태에 빠지지 않는가?
 - FALSE => err state에서 연결되는 state가 없음
 - ABA의 순서로 입력을 넣었을 때 항상 문이 열리는가?
 - FALSE => ctr이 3 이상이면 문이 열리지 않음



Hybrid System

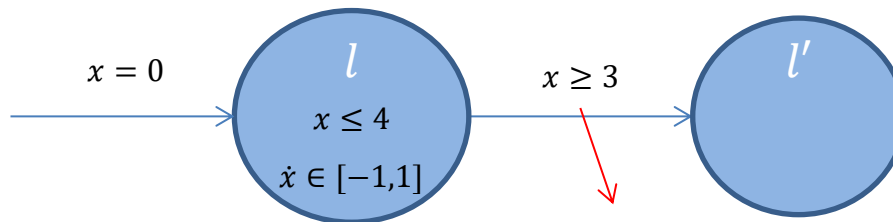
Hybrid system

- Hybrid system
 - Discrete system + continuous system
 - 주로 continuous part는 물리적인 변화를, discrete part는 시스템의 내부적인 상태를 표현하는데 사용된다.
 - E.g.) 원자력 시스템, 차량 시스템, 에너지, ...
- Analog signal
 - 일반적인 시스템은 센서로부터 받은 입력 값을 digitalize하여 처리
 - E.g.) 자판기의 버튼을 세게 누르든 약하게 누르든 일정 수준 이상(threshold)이면 동일하게 인식
 - Hybrid system은 analog signal 입력을 직접 받아 이에 대한 처리, 제어 수행을 한다.



Hybrid Automata

- Hybrid Automata
 - Hybrid system을 모델링하는데 사용되는 오토마타
 - Non-deterministic transition (jump)



x 의 값이 3이 되더라도 jump가 발생하지 않을 수 있다

$$HA = \langle X, V, flow, inv, init, E, jump, \Sigma, syn \rangle$$

- X is a set of continuous, real-valued variables
- V is a set of control modes (locations)
- $flow$ is a labeling function that assigns a flow condition to each control mode $v \in V$. The flow condition $flow(v)$ is a predicate over the variables in $X \cup \dot{X}$, refers to the first derivative of $x_i \in \dot{X}$. While the control of the hybrid automaton A is in mode v , the values of the variables and their first derivatives satisfy the flow condition $flow(v)$.
- inv is a labeling function that assigns an invariant condition to each control mode $v \in V$. While the control of the hybrid automaton A is in mode v , the variables in X must satisfy the invariant condition $inv(v)$.
- $init$ is a labeling function that assigns initial condition to each control mode $v \in V$.
- E is a finite multiset of control switches. Each control switch (v, v') is a directed edge between a source mode $v \in V$ and a target mode $v' \in V$.
- $jump$ is a labeling function that assigns a jump condition to each control switch $e \in E$. The jump condition $jump(e)$ is a predicate over variables in $X \cup X'$.
- Σ is a finite set of events, and a labeling function syn that assigns an event in Σ to each control switch $e \in E$.

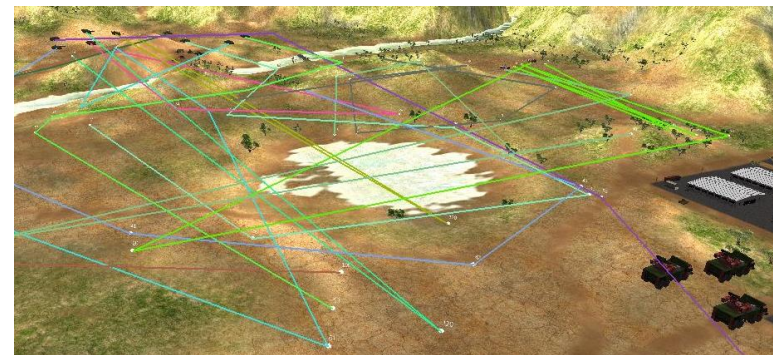
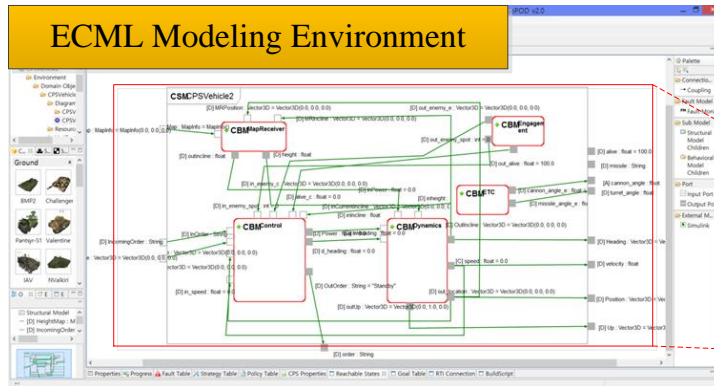
Hybrid system의 분석

- 하이브리드 시스템은 연속 값을 갖는 변수를 사용하기 때문에 일반적인 모델체킹 기법을 사용하기 어렵다.
 - E.g.) 0과 1사이의 연속적인 변수 값을 셀 수 있는가?
 - Region (값의 범위)를 이용한 분석

- 대표적인 검증 도구
 - HyTech, SpaceEx, Flow*, d/dt, ...

ECML (ETRI CPS Modeling Language)

- ECML
 - 하이브리드 시스템을 시뮬레이션 하기 위한 모델링 언어
 - 모델링 및 시뮬레이션 환경 제공



SpaceEx

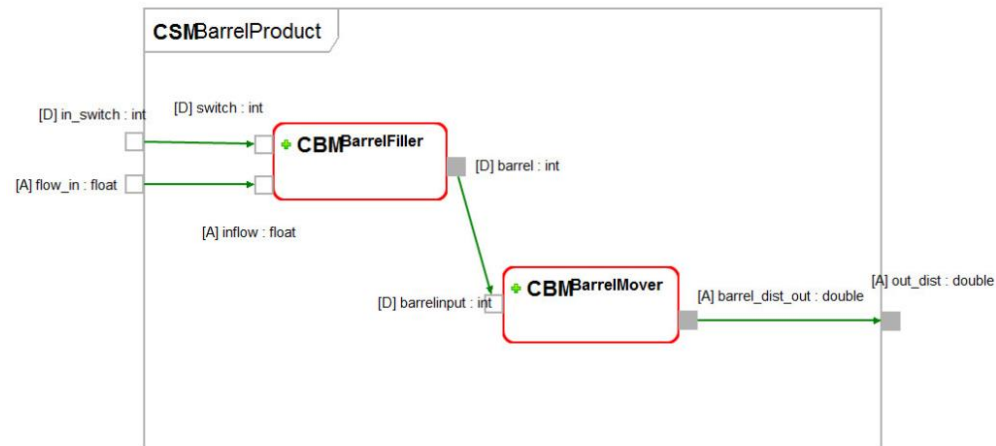
The screenshot displays the SpaceEx Virtual Machine Server interface. The top navigation bar includes links for Home, About SpaceEx, Documentation, Run SpaceEx, Downloads, and Contact. The main interface is divided into several sections:

- System:** Shows the current system as 'Ctl_Fill_System' with an 'Update' button.
- Initial states:** Contains a complex logical expression: `Controller_1.e==0 & valve==0 & motor==0 & position==0 & level==0 & Controller_1.e_sx==0 & Filler_1.e==0 & Filler_1.e_sx==0 & Conveyer_1.e_sx==0 & t=0 & loc(Controller_1)==init_asap & loc(Filler_1)==Stop_asap & loc(Conveyer_1)==Stop_asap`.
- Forbidden states:** A red box highlights the expression `level>10 || position >10`.
- Console:** Shows the simulation progress, including iterations (555 to 561) and a final message: `Forbidden states are not reachable.` and `Output of reachable states... 0s`. A red box highlights this message.
- Reports:** Displays simulation statistics: `2.31s elapsed`, `2992KB memory`, and `SpaceEx output file : output (txt).`
- Graphics:** A red box highlights the text `Forbidden states are not reachable.`.
- Analysis:** Features 'Start' and 'Stop' buttons.
- Execution terminated:** A status bar at the bottom indicates the simulation has ended.

Level 이나 Position 이 10을 초과하지 않음

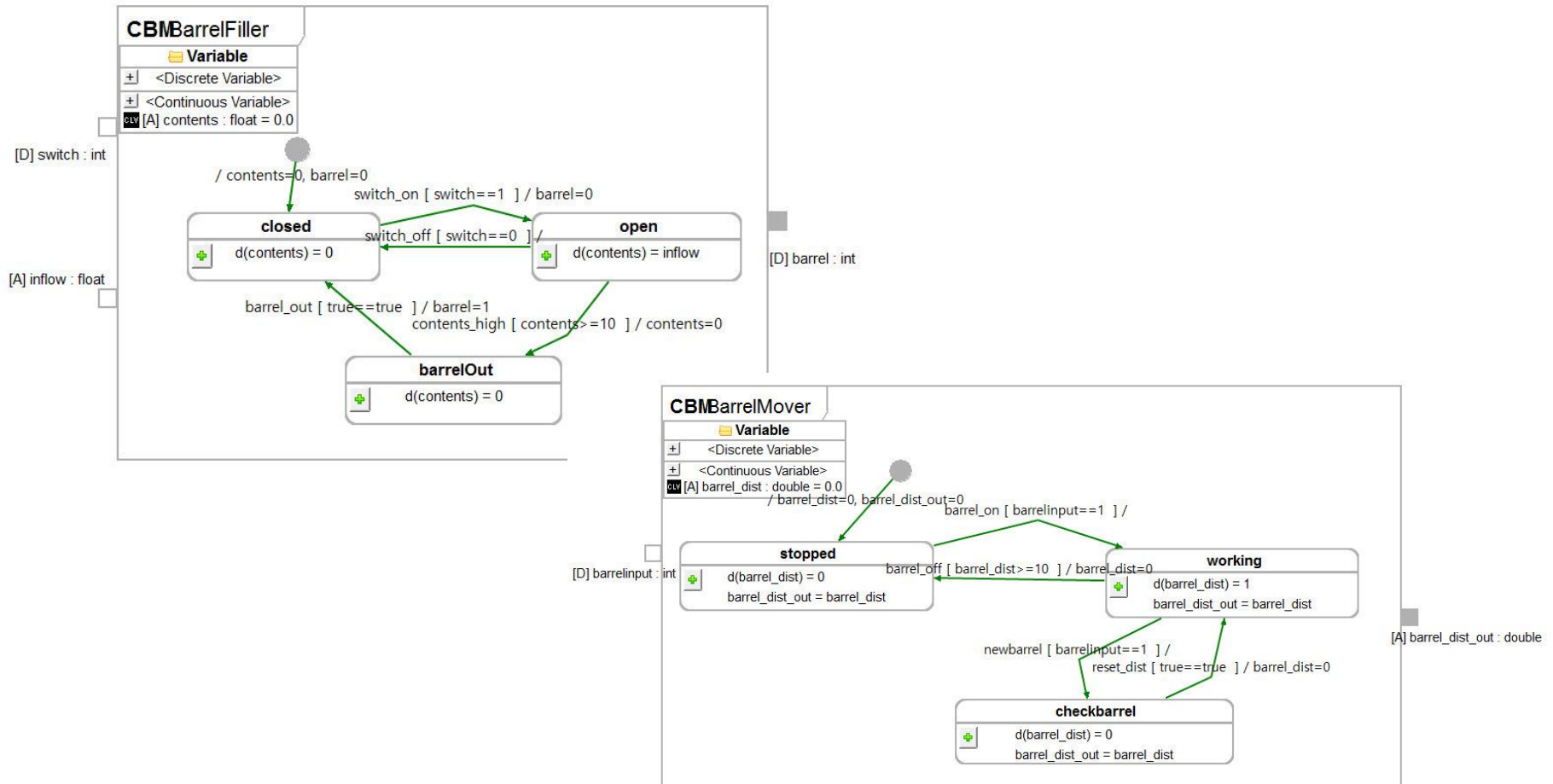
Barrel Product Example

- Barrel Product example
 - Barrel에 물을 채우고 물이 가득찬 barrel을 이동시키는 시스템
 - BarrelFiller와 BarrelMover의 subsystem으로 구성

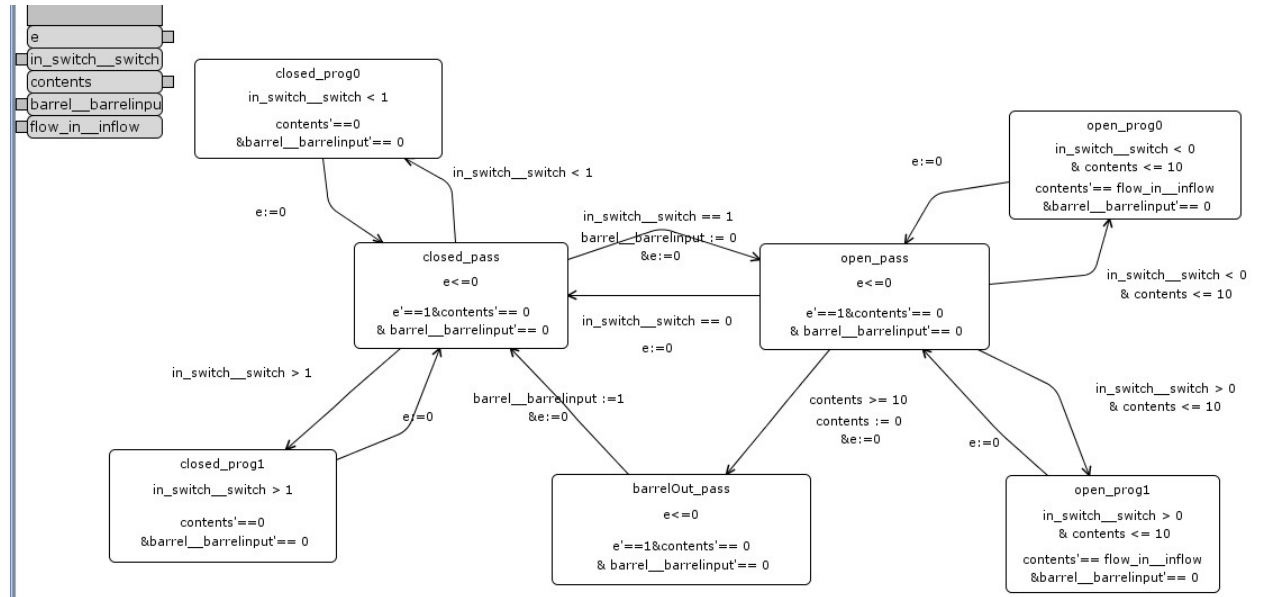
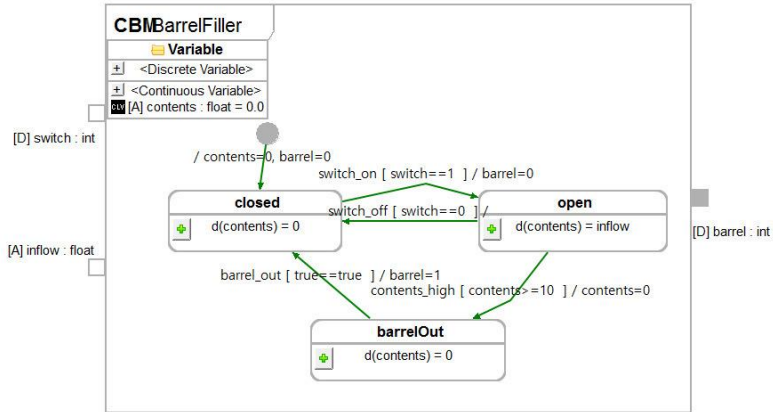


Barrel Product Example

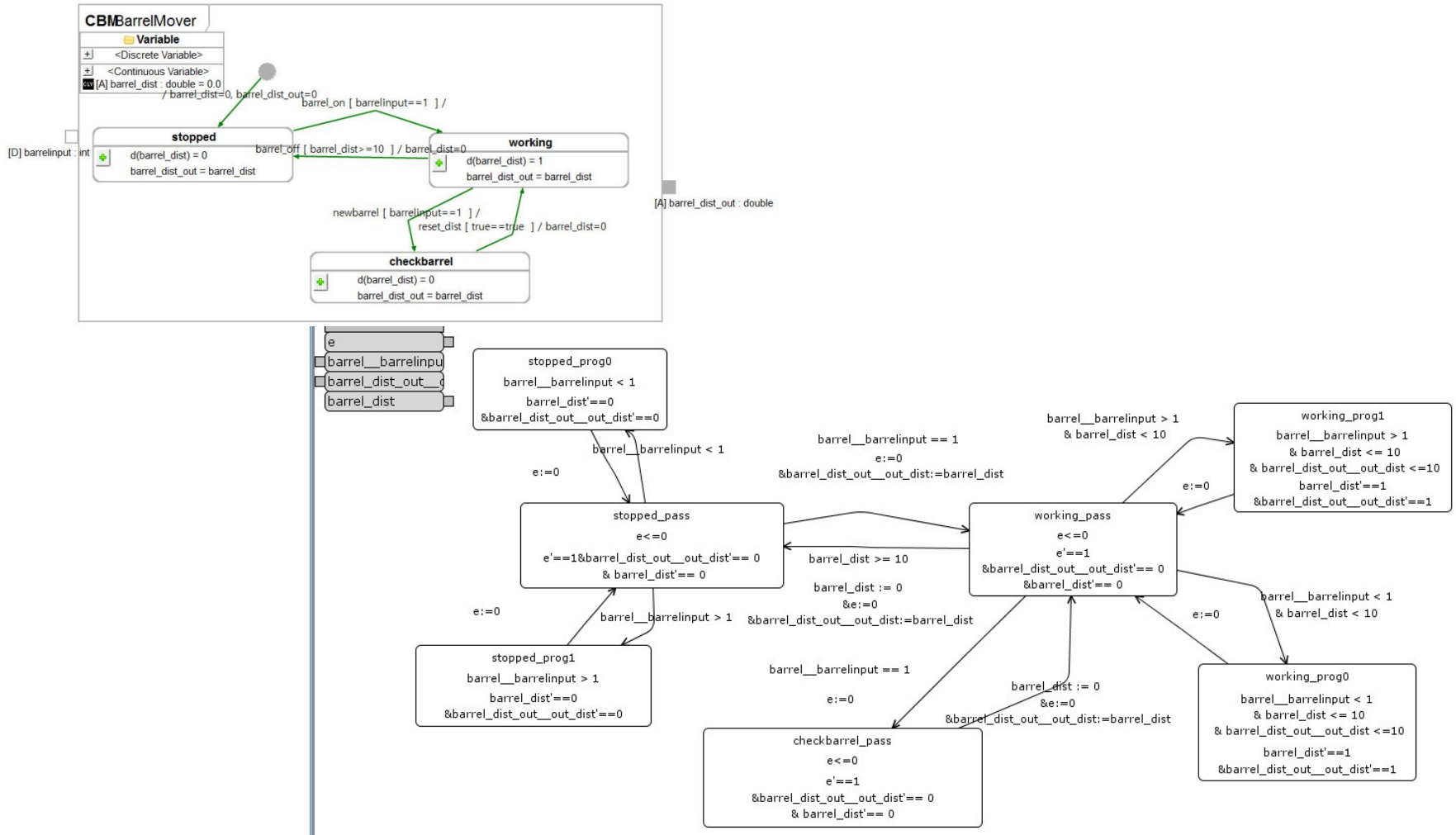
- Sub-systems



Barrel Product Example



Barrel Product Example



Barrel Product Example

- 물이 가득찬 barrel이 conveyor 벨트 끝까지 옮겨지는가?

The screenshot displays the SpaceEx Virtual Machine Server interface. The main window shows the 'System' tab for 'BarrelProduct'. The 'Console' panel on the right shows the simulation progress, including iterations and state counts. A red box highlights the message 'Forbidden states are reachable.' in the console output. The 'Initial states' and 'Forbidden states' sections are also visible.

SpaceEx [Virtual Machine Server] Home

Model Specification Options Output Advanced

System: BarrelProduct [Update]

BarrelProduct

Controlled : in_switch__switch, flow_in__inflow, t, BarrelProduct_BarrelFiller.e, contents, barrel__barrelinput, BarrelProduct_BarrelMover.e, barrel_dist_out__out_dist, barrel_dist

Base-components : BarrelProduct_Input

Initial states

```
in_switch__switch==0 & flow_in__inflow==0 & t==0 & BarrelProduct_BarrelFiller.e==0 & contents==0 & barrel__barrelinput==0 & BarrelProduct_BarrelMover.e==0 & barrel_dist_out__out_dist==0 & barrel_dist==0 & loc(BarrelProduct_Input)=loc1 & loc(BarrelProduct_BarrelFiller)==closed_pass & loc(BarrelProduct_BarrelMover)==stopped_pass
```

Forbidden states

```
(loc(BarrelProduct_BarrelMover)==working_prog0 | loc(BarrelProduct_BarrelMover)==working_prog1) & barrel__barrelinput==0 & barrel_dist==10
```

Console

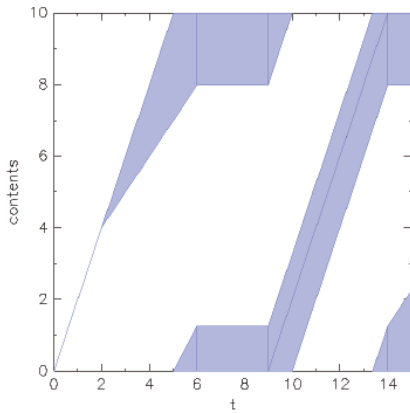
```
Iteration 3078... 1023 sym states passed, 6 waitin
Iteration 3079... 1022 sym states passed, 5 waitin
Iteration 3080... 1022 sym states passed, 4 waitin
Iteration 3081... 1022 sym states passed, 3 waitin
Iteration 3082... 1022 sym states passed, 2 waitin
Iteration 3083... 1021 sym states passed, 1 waitin
Iteration 3084... 1020 sym states passed, 0 waitin
Found fixpoint after 3085 iterations.
Computing reachable states done after 19.465s
Forbidden states are reachable.
Output of reachable bad states... 0.009s
```

Graphics

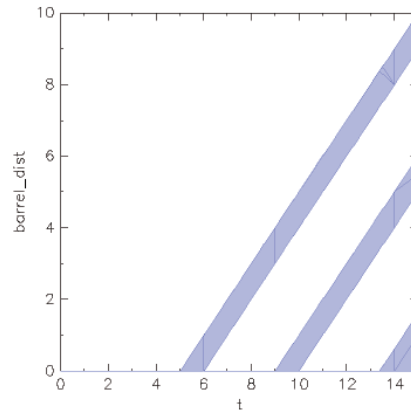
```
{(loc(BarrelProduct_Input)=loc8 & loc(t = 20 & barrel_dist_out__out_dist == BarrelProduct_BarrelFiller.e == 0 & barrel_dist_out__out_dist == 10 & flow_in__inflow == 1 & in_switch__barrel__barrelinput == 0) | (loc(BarrelProduct_BarrelMover)==working_prog0 & in_switch__switch == 0 & contents == 10 & loc(BarrelProduct_Input)=loc8 & loc(R
```

Barrel Product Example

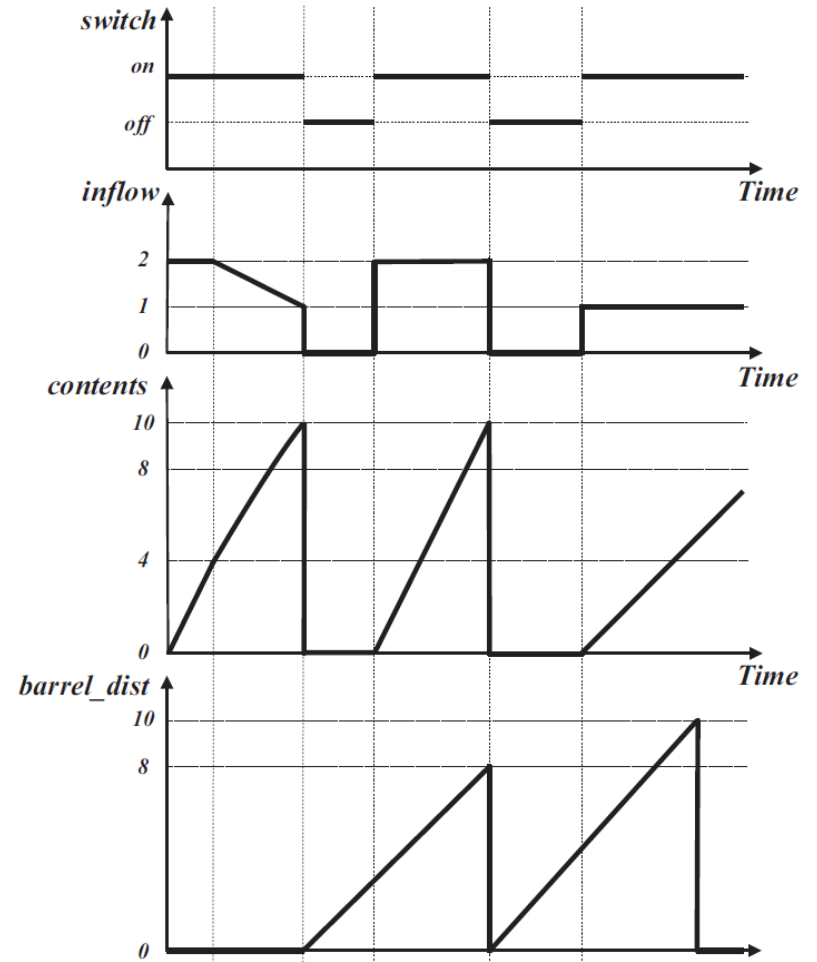
- Behavior



(a) Sequential change of *contents*



(b) Sequential change of *barrel_dist*



감사합니다.