Outline (Meeting 9 Feb, 2021)

- Paper (A) Stealthy attacks
- Paper (B) Mitigation Strategy

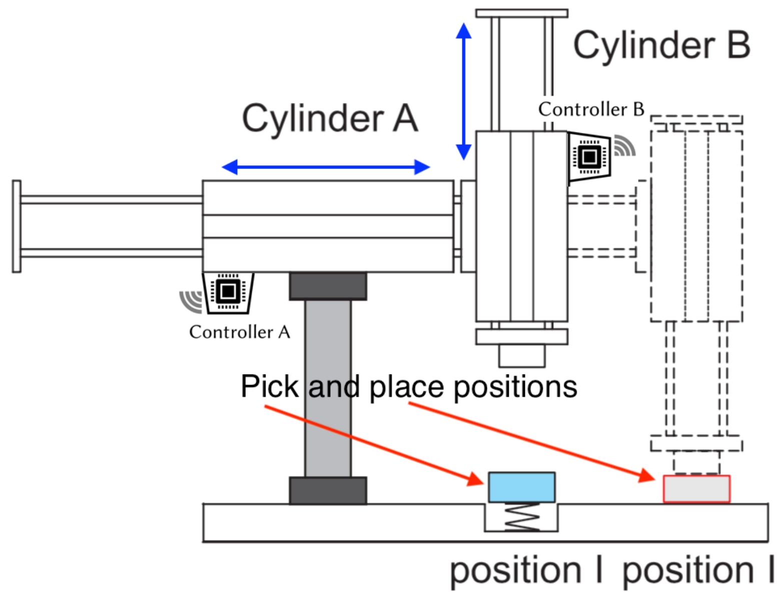
Reconfigurable Pneumatic System

Reconfigurable Pneumatic System

System's work cycle:

B + B – A + B + B – A–

where X+ denotes advancement and X– retracting of cylinder X (X ∈ {A, B})

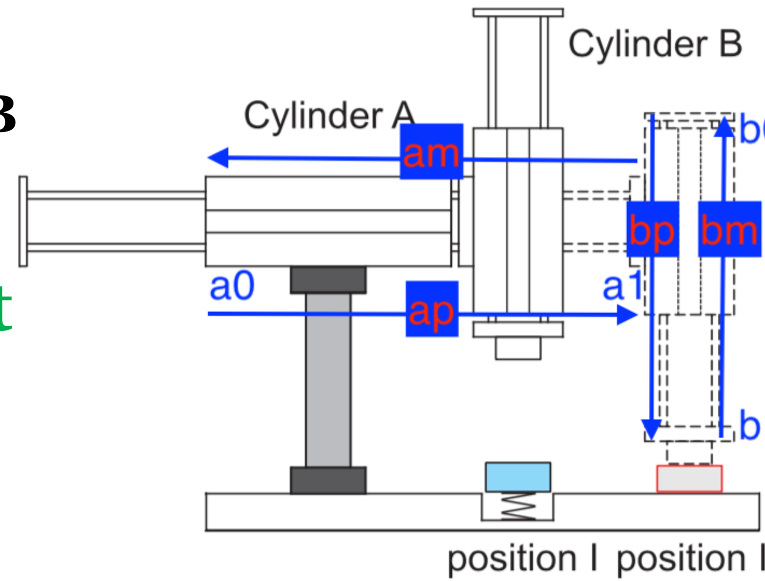


Input and output signals

Controller's input and output signals:

	Controller A	**Controller B**
Sensing signals:	a0, a1	b0, b1
Commands:	ap, am	bp, bm, st

Reconfigurable Pneumatic System



Paper (A)

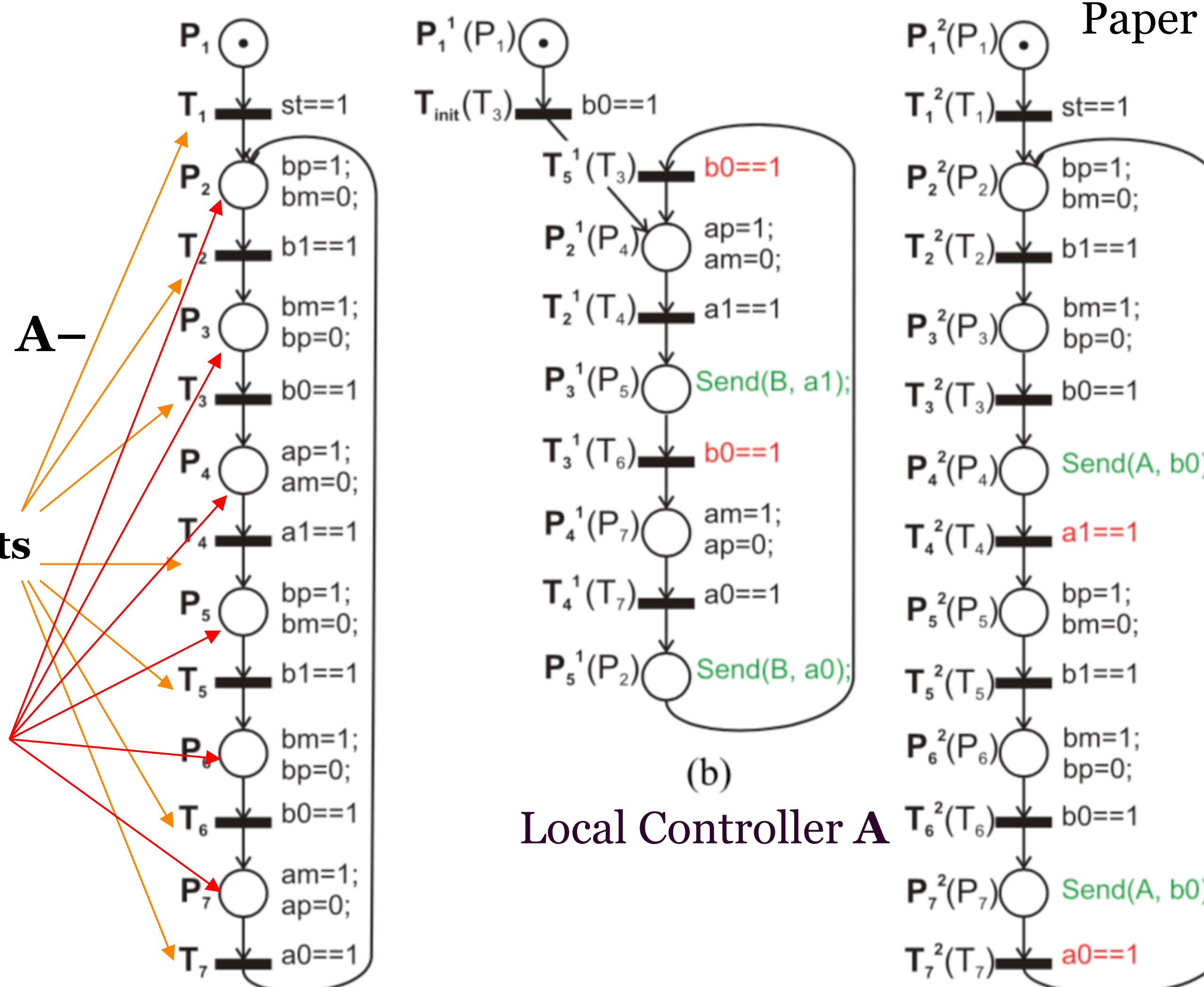
CIPN

System's work cycle:

B + B − A + B + B − A−

Sensing Events

Commands

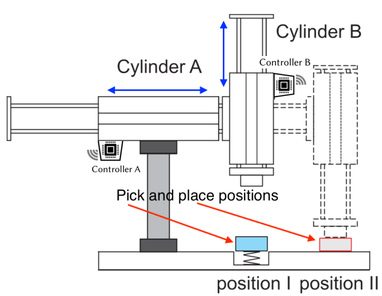


(b)

Local Controller **A**

Global (a)

Local Controller **B** (c)



CIPN: Control interpreted Petri Nets

Cylinders A and B (FSMi) Controllable and Uncontrollable events

Possible behaviours of Cylinder **A** and **B** represented as FSA.

$$(Q^i, E^i, f^i, q_0^i)$$

$$E^i = E_c^i \cup E_{uc}^i$$

$$E_o^i \cup E_{uo}^i$$

Local Controller **A** and **B**:

Actuator signals
(**Controllable** events)

Sensor signals
(**Uncontrollable** events)

- Each Cylinder **A** and **B** modeled as FSA that is locally controlled by a Local Controller specified by **CIPN**.
- **Sensor signals** assigned to **CIPN**i transitions belong to **uncontrollable** events, while **actuator signals** assigned to the places are **controllable**.

Controlled loop behavior of system

- Each Local controller **A** and **B** provides controlled behaviour of the Cylinder **A** and **B** through a feedback control loop by imposing a **Supervisor**.

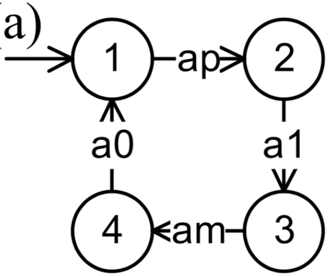
The system as a whole can be represented as an FSA:

$$(Q^i, E^i, f^i, q_0^i) \times (Q^i, E^i, f^i, q_0^i)$$

$$S \times G$$

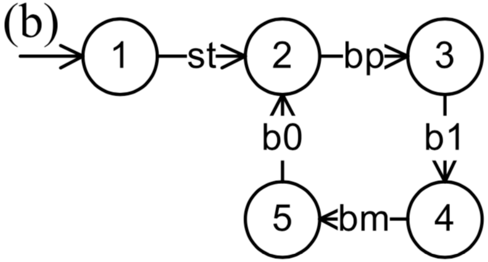
Model of system as FSAs

(a)



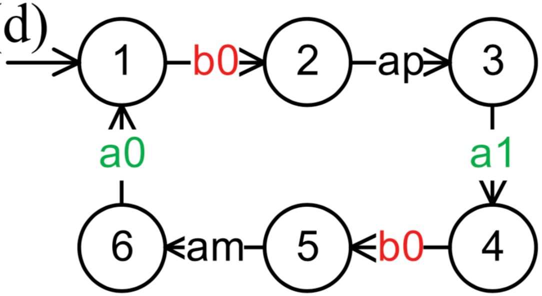
Cylinder **A (Global_A)**

(b)



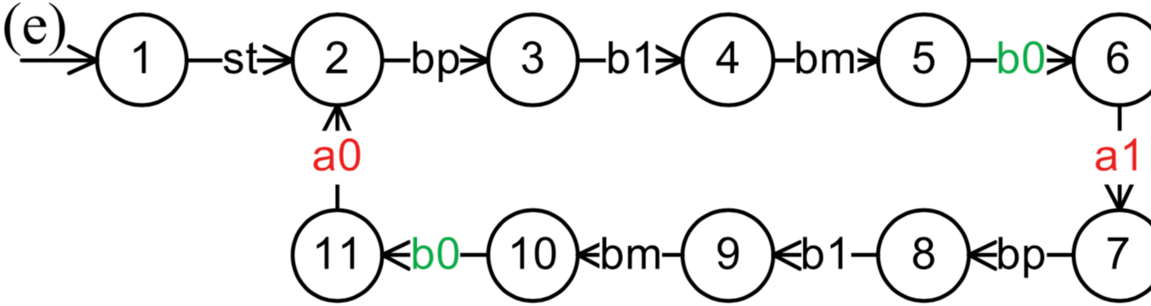
Cylinder **B (Global_B)**

(d)



Local Controller **A (Supervisor_A)**

(e)



Local Controller **A (Supervisor_B)**

Observable events: $\{ap, a1, am, a0\}$ $\{bp, b1, bm, b0, st\}$

Controllable events: $\{ap, am\}$ $\{bp, bm\}$

Modeling impacts of attacks

Two possible types of attacks:

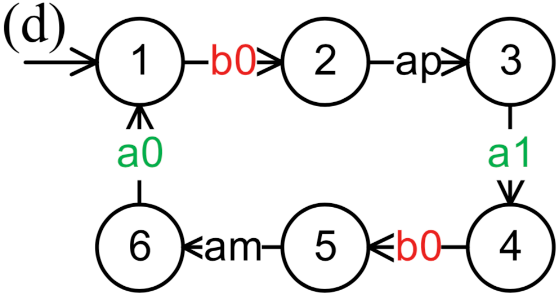
- **Event insertion:** A controller **Si** receives an event before **Sj** sends it.
- **Event removal:** An event sent to a controller **Si** from a controller **Sj** is not received.

Assumption **(Stealthy attacks):**

- We assume that the attacker's goal is to affect the performance of the system **without being immediately revealed**;
- Attacker knows the current states of the cylinders and supervisors.

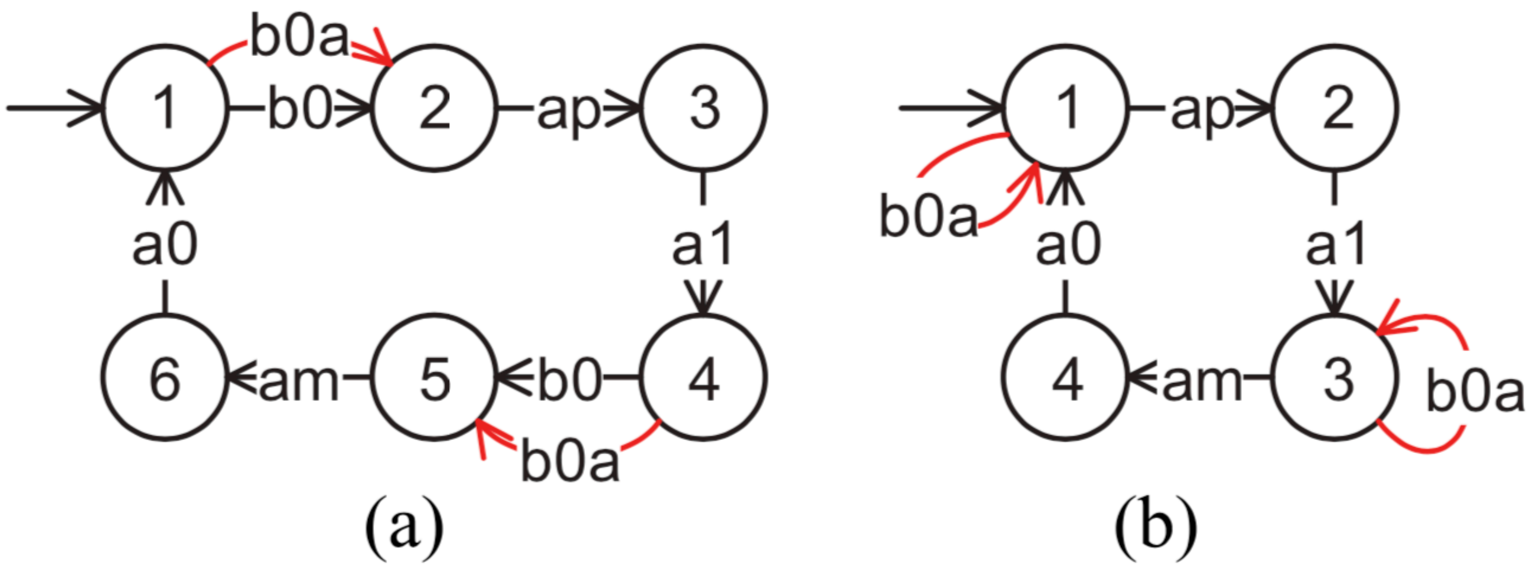
Simple event insertion can be easily detected.
Such as inserting events like **bo** when automaton **Supervisor_A** is in, e.g., state3.

(d)



Local Controller **A** **(Supervisor_A)**

Modeling event insertion attack



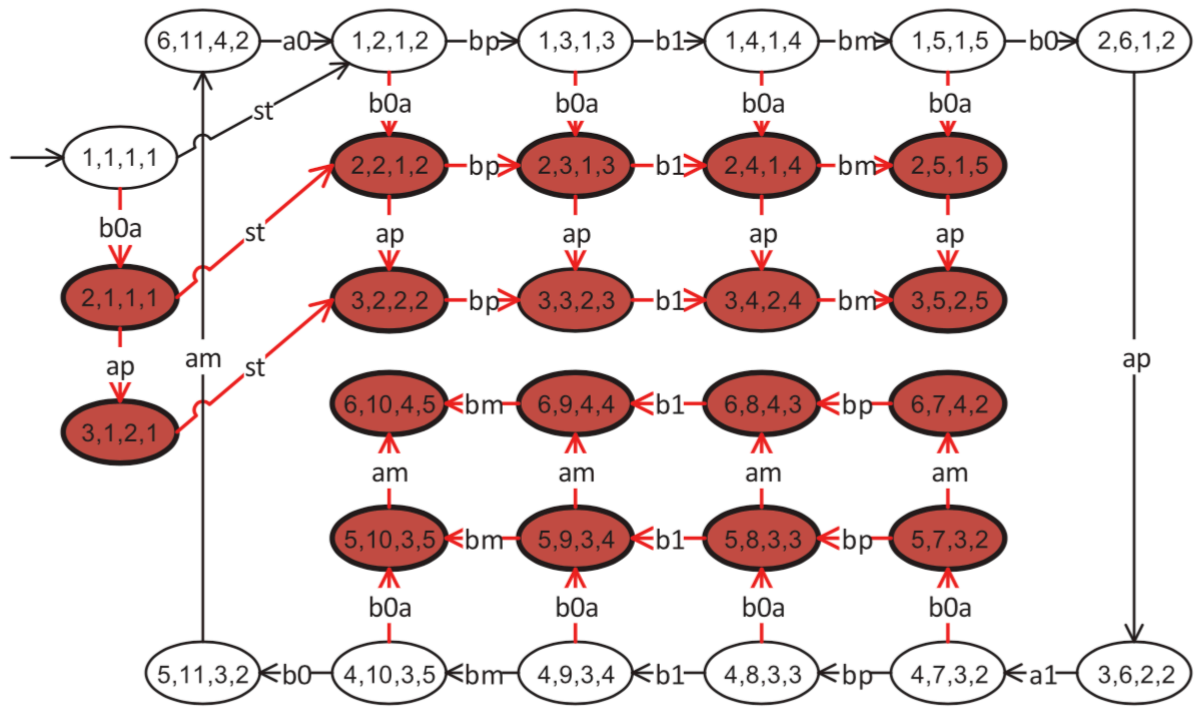
Local Controller **A (Supervisor_A)**
Under attack

Cylinder **A (Global_A)**
Under attack

Paper (A)

Model of system under event insertion attack

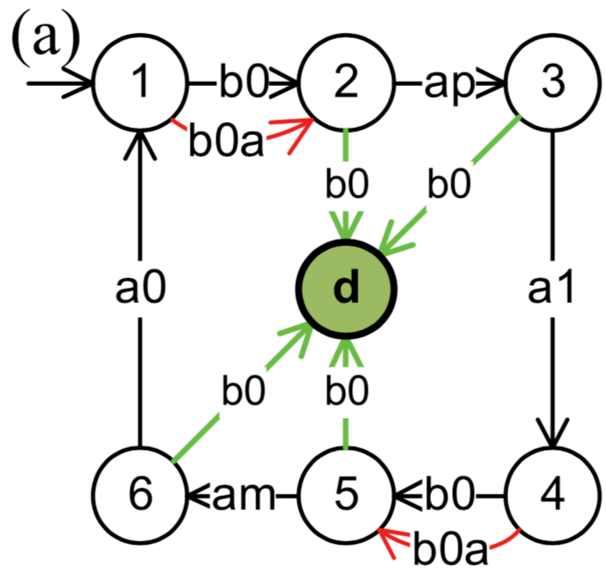
The model of the system under such attack



state(x, y, z, u)

X: Supervisor_A state,
Y: Supervisor_B state,
Z: Global_A state,
U: Global_B state.

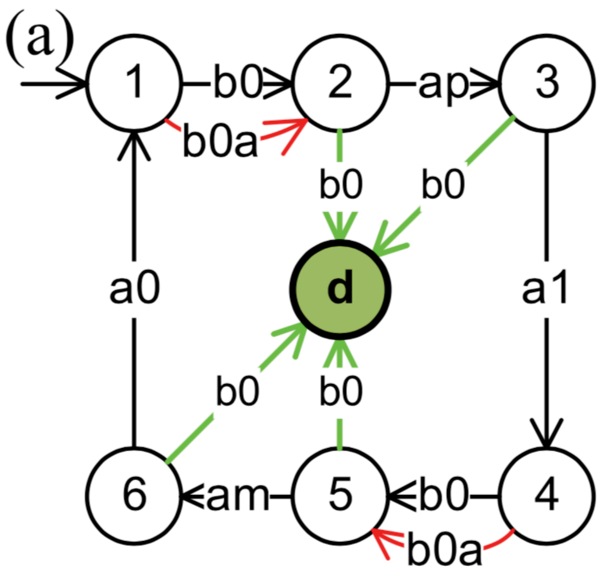
Modeling a supervisor integrated with a detected state



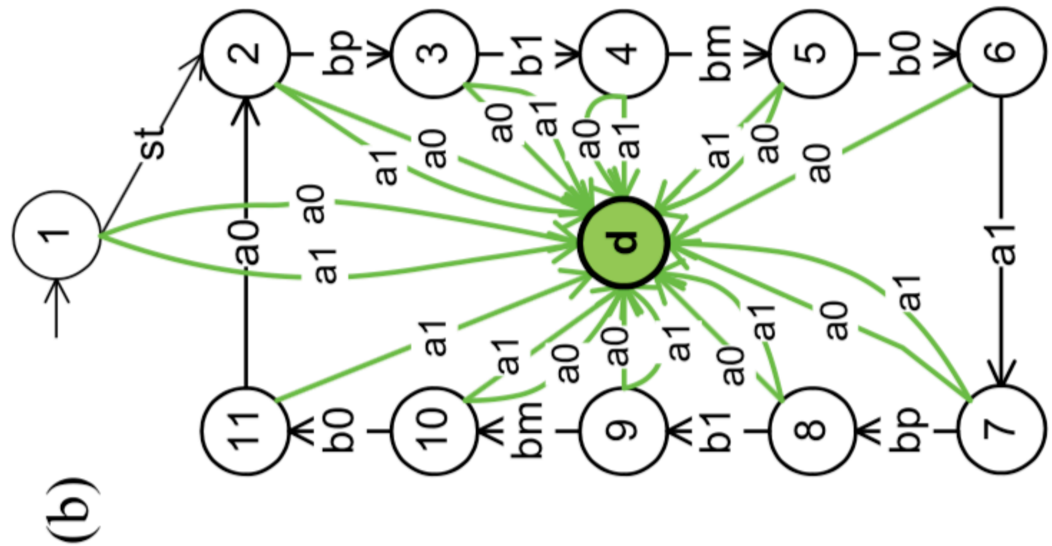
Local Controller **A (Supervisor_A)**
Integrated with a detect state

Paper (A)

Modeling a supervisor integrated with a detected state



Local Controller **A (Supervisor_A)**
Integrated with a detect state



Local Controller **B (Supervisor_B)**
Integrated with a detect state

Identification of undesired system behavior

The question is whether the system under attack will lead to a catastrophic damage before the attack is revealed.

Two **situations** that endanger the quality of the process in Reconfigurable Pneumatic System:

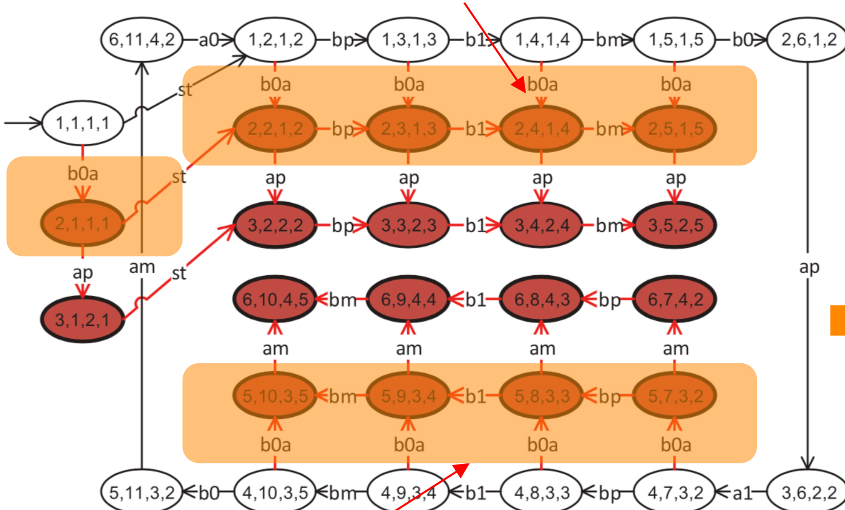
- (CD1)- Cylinder B **enters** position II from **horizontal direction**,
- (CD2)- Cylinder B **leaves** position II in **horizontal direction**.

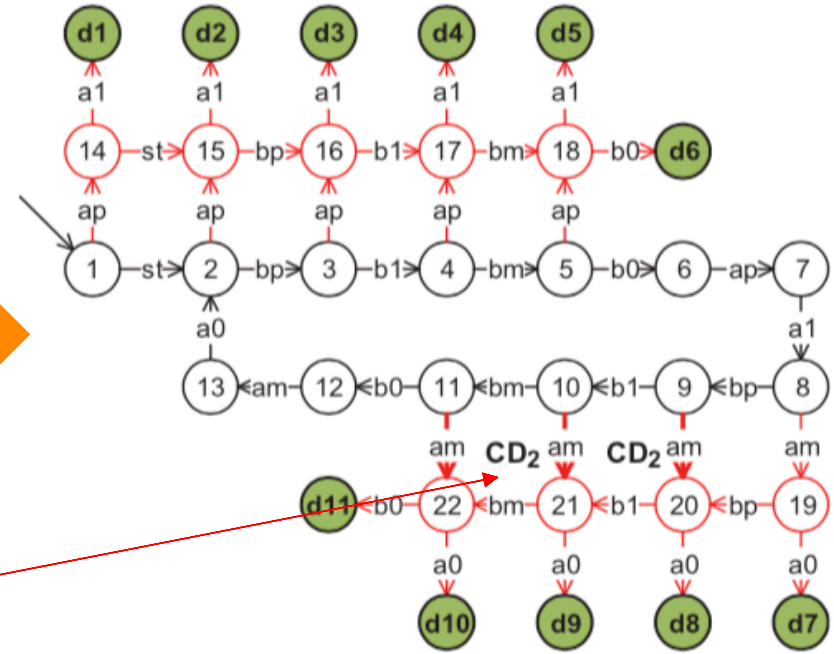
CD1 and CD are presented as **events strings** set.

CD_1	$w_{c,1}^{1} = w_{r1}(ama0ap)^{*}bp$, $w_{c,2}^{1} = w_{r1}(amap)^{*}bp$ where $w_{r1} = w_r bpb1bmb0ap(a1ama0ap)^{*}$
CD_2	$w_{c,1}^{2} = w_{r2}am$, $w_{c,2}^{2} = w_{r2}b1am$, $w_{c,3}^{2} = w_{r2}b1bmam$ where $w_{r2} = w_r bpb1bmb0apa1bp(b1bmb0bp)^{*}$

Late Detection (Stealthy attacks)

Unobservable events





Applied Projection

$$\mathrm{CD}_2 \quad w_{c,1}^2 = w_{r2}am,\ w_{c,2}^2 = w_{r2}b1am,\ w_{c,3}^2 = w_{r2}b1bmam$$

Catastrophic damage where $w_{r2} = w_r bpb1bmb0apa1bp(b1bmb0bp)^*$

Attack Mitigation

Target problem:

- Taking actions **too late** when the attack is detected,

Solution:

- Formulates the attack mitigation problem as a **tolerant** control problem under partial observation.
  - Controllable events
  - Defendable events

The goal is to prevent the new system from reaching **unsafe states** while **maximizing the desirable behavior**, which is the closed-loop language without attack.

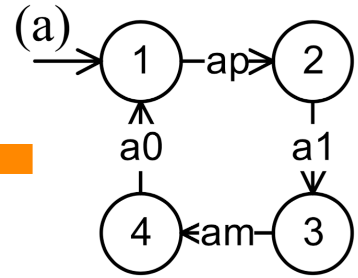
- Example

Outline (Meeting 16 Feb, 2021)

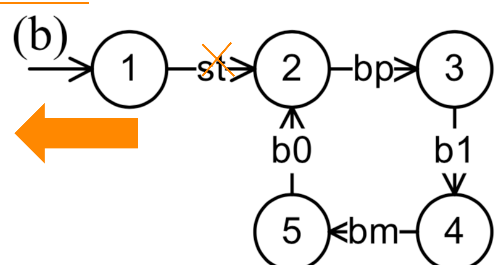
- Reconfigurable Pneumatic System Rebeca Model (Rebeca codes)
  - Plant without controller components (Physical Layer)
  - Plant with controller components (Cyber and Physical Layers)
  - Integrate controller components with detectors
  - Attacks (Event Insertion)
- Mitigation Module (LF code, implementation will be next week)

*Pneumatic_Manipulator_ABT_FSM_Plant_Sensor.rebeca

```
//system's work cycle: all possible states of plant
//intitial state: a0, b0
reactiveclass Cylinder_A(4){
    knownrebecs{}
    statevars{}
    Cylinder_A(){self.ap();}
    msgsrv ap() {self.a1();}
    msgsrv a1() {self.am();}
    msgsrv am() {self.a0();}
    msgsrv a0() {self.ap();}
}
reactiveclass Cylinder_B(4){
    knownrebecs{}
    statevars{}
    Cylinder_B(){self.bp();}
    msgsrv bp() {self.b1();}
    msgsrv b1() {self.bm();}
    msgsrv bm() {self.b0();}
    msgsrv b0() {self.bp();}
}
main{
    Cylinder_A cylinder_A():(),
    Cylinder_B cylinder_B():();
}
```

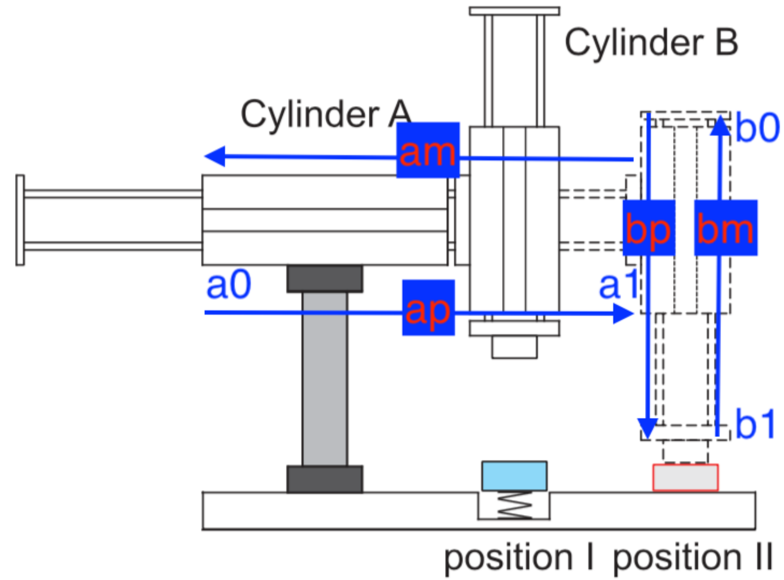


Cylinder **A**



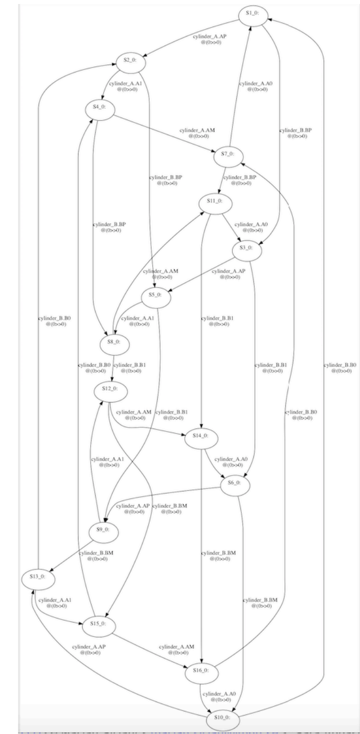
Cylinder **B**

Reconfigurable Pneumatic System



Rebeca model without controller components

Generated state-space by Afra (Rebeca model without controllers)

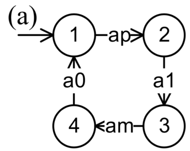


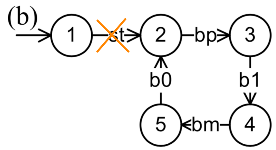
Pneumatic_Manipulator_ABT_FSM_Plant_Sensor_Controller.rebeca   Pneumatic_Manipulator_ABT_FSM_Plant_Sensor.rebeca

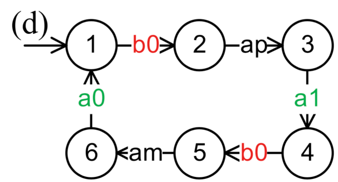
```
//system's work cycle: bp, b1, bm, b0, bp, b1, bm, b0, am, a0
//intitial state: a0, b0
reactiveclass Cylinder_A(4){
    knownrebecs{LocalControllerA lc_A;}
    statevars{}
    Cylinder_A(){}
    msgsrv ap() {self.a1();}
    msgsrv a1() {lc_A.a1();}
    msgsrv am() {self.a0();}
    msgsrv a0() {lc_A.a0();}
}
reactiveclass Cylinder_B(4){
    knownrebecs{LocalControllerB lc_B;}
    statevars{}
    Cylinder_B(){}
    msgsrv bp() {self.b1();}
    msgsrv b1() {lc_B.b1();}
    msgsrv bm() {self.b0();}
    msgsrv b0() {lc_B.b0();}
}
reactiveclass LocalControllerA(4){
    knownrebecs{Cylinder_A Cyl_a;LocalControllerB lc_B;}
    statevars{boolean turn;}
    LocalControllerA(){turn = true;}
    msgsrv a0() {lc_B.a0();} //Input signals
    msgsrv a1() {lc_B.a1();}
    msgsrv b0() {
        if(turn){   Cyl_a.ap(); turn = false;} else {
                Cyl_a.am(); turn = true;} //Output signals
    }
}
reactiveclass LocalControllerB(4){
    knownrebecs{Cylinder_B Cyl_b;LocalControllerA lc_A;}
    statevars{}
    LocalControllerB(){self.st();}
    msgsrv st() {Cyl_b.bp();} //start bottom
    msgsrv b1() {Cyl_b.bm();}
    msgsrv b0() {lc_A.b0();}
    msgsrv a0() {Cyl_b.bp();}
    msgsrv a1() {Cyl_b.bp();}

}

main{
    Cylinder_A cylinder_A(lc_A):();
    Cylinder_B cylinder_B(lc_B):();
    LocalControllerA lc_A(cylinder_A, lc_B):();
    LocalControllerB lc_B(cylinder_B, lc_A):();
}
```

Rebeca model with controller components

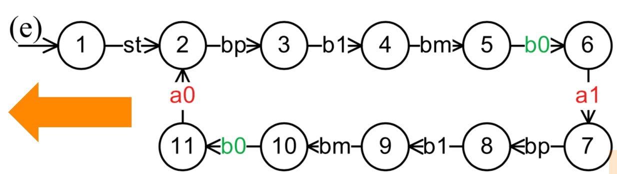
Desirable Scenario







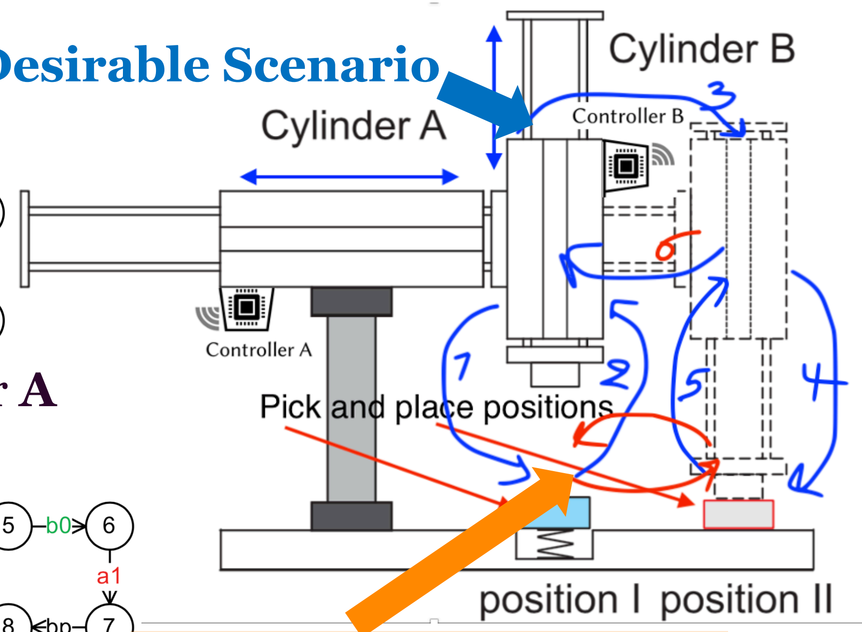
Local Controller A



Local Controller B

```
/*
CTL{
Safety_1: E(ap U bp);
Safety_2: E(am U bp);
    }
*/
```

Reconfigurable Pneumatic System

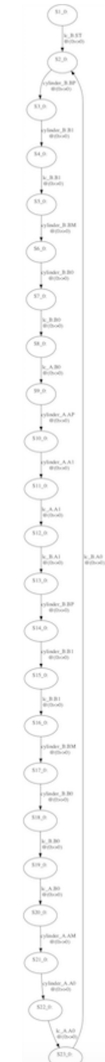


Safety Properties:

Safety_1: !(bp ->ap);

Safety_2: !(bp ->am);

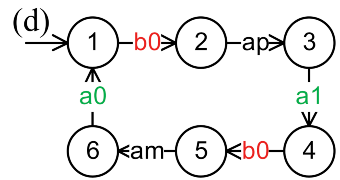
Generated state-space by Afra
(Rebeca model with controller components)



Pneumatic_Manipulator_ABT_FSM_Plant_Controller_Detector.rebeca | Pneumatic_Manipulator_ABT_FSM_Plant_Sensor_Controller.rebeca | Pneumatic_Manipulator_

```
//system's work cycle: bp, b1, bm, b0, bp, b1, bm, b0, am, a0
//intitial state: a0, b0
reactiveclass Cylinder_A(4){
    knownrebecs{LocalControllerA lc_A;}
    statevars{}
    Cylinder_A(){}
    msgsrv ap() {delay(1);self.a1();}
    msgsrv a1() {lc_A.a1();}
    msgsrv am() {delay(1);self.a0();}
    msgsrv a0() {lc_A.a0();}
}
reactiveclass Cylinder_B(4){
    knownrebecs{LocalControllerB lc_B;}
    statevars{}
    Cylinder_B(){}
    msgsrv bp() {delay(1);self.b1();}
    msgsrv b1() {lc_B.b1();}
    msgsrv bm() {delay(1);self.b0();}
    msgsrv b0() {lc_B.b0();}
}

reactiveclass LocalControllerA(4){
    knownrebecs{Cylinder_A Cyl_a;LocalControllerB lc_B;}
    statevars{boolean [4]s; boolean turn,alarm;} // expected inputs order s = (b0,a1,b0,a0)*
    LocalControllerA(){
        //self.b0() after(0); //Event-Insetion *The integrated detector will not recognize this injection before violation (Bad-Event).*
        s[0] = true; s[1] = false; s[2] = false; s[3] = false; // initial state
        turn = true;
        alarm = false;
    }
    msgsrv a0() {self.d(3);}
    msgsrv a1() {self.d(1);}
    msgsrv b0() {if(turn){ self.d(0); turn = false;}
                else {self.d(2); turn = true;}
    }
```

Attack



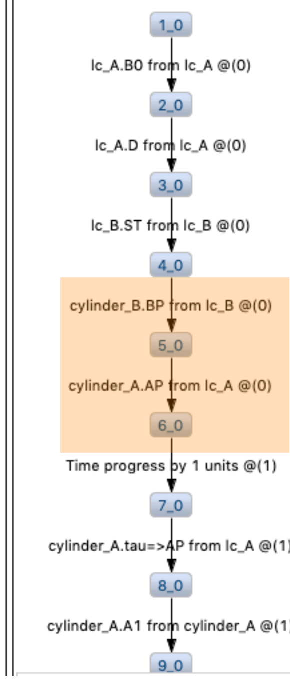
Detector component

```
    msgsrv d(int c) { //detector
        if(c == 0 && s[0] == true){Cyl_a.ap();       s[0] = false; s[1] = true; s[2] = false; s[3] = false;}
        else if(c == 1 && s[1] == true){lc_B.a1();   s[0] = false; s[1] = false; s[2] = true; s[3] = false;}
        else if(c == 2 && s[2] == true){Cyl_a.am();  s[0] = false; s[1] = false; s[2] = false; s[3] = true;}
        else if(c == 3 && s[3] == true){lc_B.a0();   s[0] = true; s[1] = false; s[2] = false; s[3] = false;}
        else {alarm = true;}
    }
}
reactiveclass LocalControllerB(4){
    knownrebecs{Cylinder_B Cyl_b;LocalControllerA lc_A;}
    statevars{}
    LocalControllerB(){self.st();}
    msgsrv st() {Cyl_b.bp();}
    msgsrv b1() {Cyl_b.bm();}
    msgsrv b0() {lc_A.b0();}

    msgsrv a0() {Cyl_b.bp();}
    msgsrv a1() {Cyl_b.bp();}
}

main{
    Cylinder_A cylinder_A(lc_A):();
    Cylinder_B cylinder_B(lc_B):();
    LocalControllerA lc_A(cylinder_A, lc_B):();
```

Late Detection

Pneumatic_Ma... | Counter Example

```
property {

    define {
        alarm = lc_A.alarm;
    }
    Assertion{
        Safety_Alarm: !(alram);
    }
    /*
    CTL{
    Violation1: E(ap U bp);
    Violation2: E(am U bp);
        }
    */
}
```



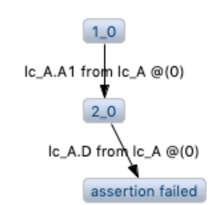
Event Injection

Self.bo() after(o);

On-time Detection

Pneumatic_Ma... | Counter Example

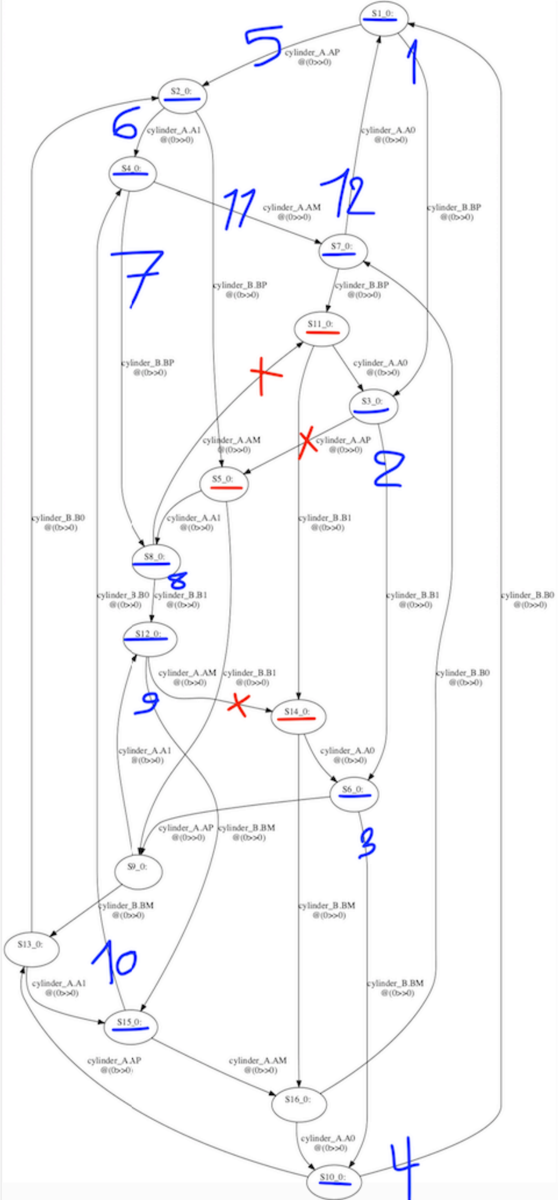
```
wnrebecs{Cylinder_A Cyl_a;Loc
tevars{boolean [4]s; boolean
alControllerA(){
 self.a1() after(0); //Event-
 s[0] = true; s[1] = false; s
 turn = true;
 alarm = false;

srv a0() {self.d(3);}
srv a1() {self.d(1);}
srv b0() {if(turn){ self.d(0)
        else {self.d(2);
```



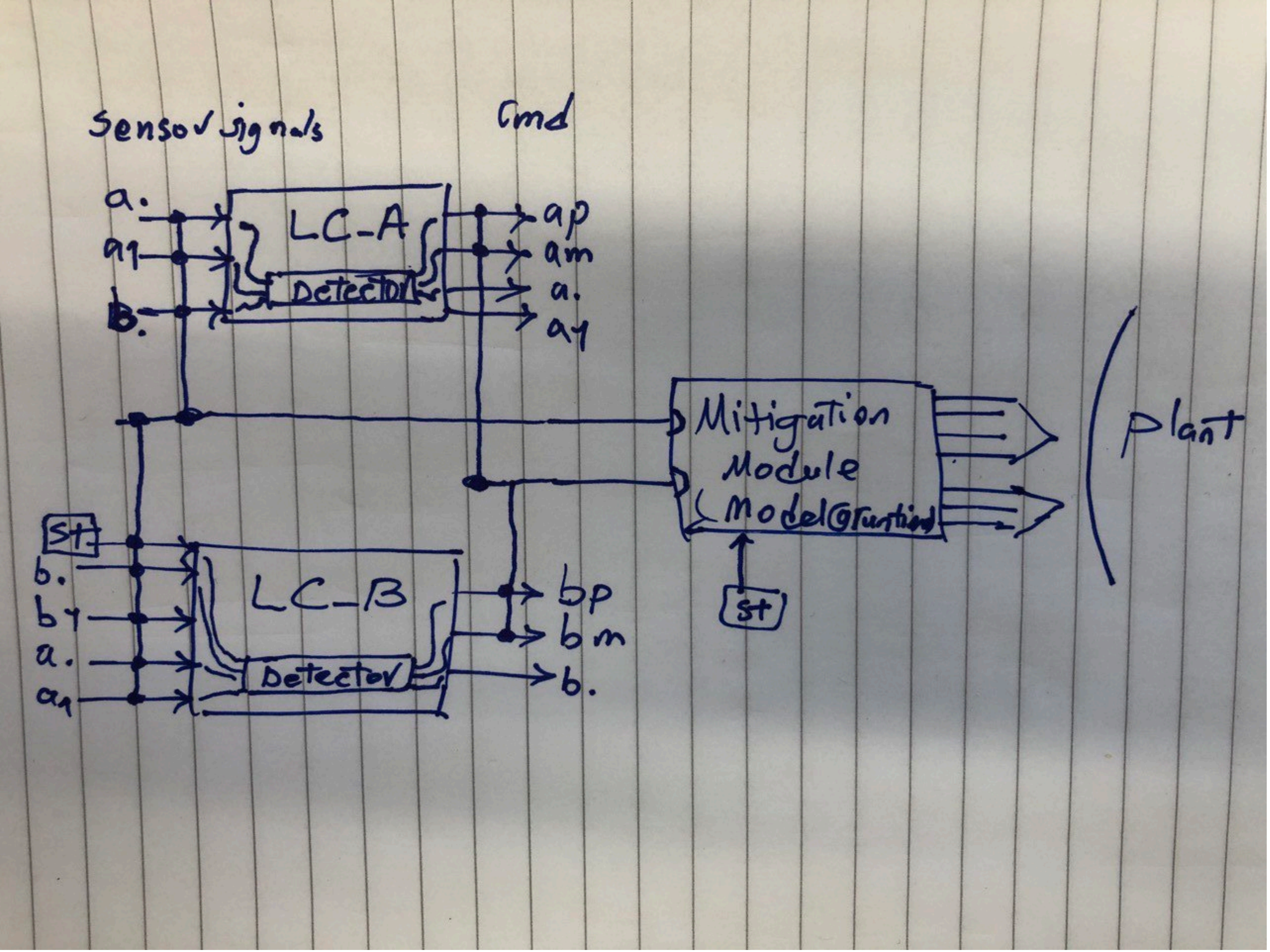
Event Injection

Self.a1() after(o);

Mitigation Module (Model@runtime)



LF Modules



Sensor signals
Cmd
a.
a1
B.
LC-A
Detector
ap
am
a.
a1
Mitigation
Module
Model@function
plant
St
b.
b1
a.
a1
LC-B
Detector
bp
bm
b.
St