Formal Verification of Robust State Estimator

Ashish Tiwari
SRI International

Problem Description

LTI system:

$$\begin{aligned}\vec{x}_{k+1} &= A\vec{x}_k + B\vec{u}_k + \vec{v}_k \\ \vec{y}_k &= C\vec{x}_k + \vec{w}_k + \vec{e}_k\end{aligned}$$

where

$\vec{v}, \vec{w}$: noise

$\vec{e}$: attack

Estimate state $\vec{x}$ and prove error in estimation is bounded

Assumptions

- Model validated: The equations describe the real system
- Bounded noise: $|\vec{v}| \leq \epsilon_v, \qquad |\vec{w}| \leq \epsilon_w$
- Know $\vec{u}$
- Same set of q_{max} sensors attacked for $t = 0, \ldots, N-1$
- Optimization procedure always returns a feasible solution

Resilient State Estimation

$$\begin{aligned}(\vec{x}_0^*, \vec{E}^*) &= argmin_{\vec{x},\vec{E}} |E|_0 \\ & s.t.\ Y - \Phi(\vec{x}) - E \leq \Delta\end{aligned}$$

where

Y = actual observations $[\vec{y}_0 \mid \ldots \mid \vec{y}_{N-1}]$

$\phi(\vec{x})$ = ideal observations if there was no noise/attack
$\phi(\vec{x}) = [C\vec{x} \mid \ldots \mid CA^{N-1}\vec{x} + \sum_{i=0}^{N-2} CA^{N-2-i}B\vec{u}_i]$

E = variables denoting guessed attacks $[e_0 \mid \ldots \mid e_{N-1}]$

Δ = upper bound in expected deviation due to noise
$\Delta = [\ldots \mid \quad |C| \sum_{i=0}^{k-1} |A^{k-1-i}|\epsilon_v + \epsilon_w \quad \mid \ldots]$

Claim

The state estimation error $|\vec{x}_0^* - \vec{x}_0|$ is bounded

The above claim is proved in [Pajic et al 2014] for any A, B, C and any values of all the other parameters

Formal verification approach:

- We instantiate the model and RSE algorithm for landshark
- We model it in SAL as an infinite-state transition system
- We formally verify the claim (using k-induction prover)

Formal Model of RSE on Landshark

- Landshark model has 2 state variables: position, velocity
  $A = 2 \times 2$ matrix; $B = 2 \times 1$ (1 control input)
- Three sensors observing velocity
  $C = [0, 1; 0, 1; 0, 1]$
- Noise: All noise terms in $[-0.1, 0.1]$ unchanging with time
  $\epsilon_v = \epsilon_w = 0.1$
- One fixed sensor under attack
  $q_{max} = 1$
- Estimator: Optimization-based estimator replaced by feasibility-based estimator
  Non-deterministically returns any $\vec{x}_0$ consistent with observations of any 2 out of 3 sensors

Landshark Model

```
plant: MODULE =
BEGIN
 OUTPUT velocity, u: REAL
 LOCAL  position, noise1: REAL
 INITIALIZATION
  position IN { z : REAL | -1 <= z AND z <= 1 } ;
  velocity IN { z : REAL | -1 <= z AND z <= 1 } ;
  noise1 IN { z : REAL | -bound <= z AND z <= bound } ;
  u IN { z : REAL | -1 <= z AND z <= 1 } ;
 TRANSITION [
  TRUE --> position' = a11 * position + a12 * velocity + b11 * u' + noise1';
           velocity' = a21 * position + a22 * velocity + b21 * u' + noise1';
           noise1' IN { z : REAL | -bound  <= z AND z <= bound };
           u' IN { z : REAL | -1  <= z AND z <= 1 }; ]
END ;
```

Sensor Model Normal

```
sensor_normal[i:SensorType]: MODULE =
BEGIN
 INPUT velocity:REAL
 LOCAL noise: REAL
 OUTPUT y: REAL
 INITIALIZATION
  y = 0;
  noise IN { z : REAL | -bound <= z AND z <= bound } ;

 TRANSITION
 [
  TRUE --> y' = velocity' + noise' ;
           noise' IN { z : REAL | -bound <= z AND z <= bound } ;
 ]
END;
```

Sensor Model Attack

```
sensor_attack: MODULE =
BEGIN
 INPUT velocity: REAL
 LOCAL noise2, attack: REAL
 OUTPUT y1: REAL
 INITIALIZATION
  y1 = 0;
  noise2 IN { z : REAL | -bound  <= z AND z <= bound  } ;
  attack IN { z : REAL | -10 <= z AND z <= 10 } ;
 TRANSITION
 [ TRUE --> y1' = velocity' + noise2' + attack' ;
            noise2' IN { z : REAL | -bound  <= z AND z <= bound } ;
            attack' IN { z : REAL | -10 <= z AND z <= 10 } ;
 ]
END;
```

Estimator Model

```
estimator: MODULE =
BEGIN
 INPUT u, y1, y2, y3: REAL
 OUTPUT x_estimate: REAL
 LOCAL x0: REAL
 INITIALIZATION x0 = 0;
 DEFINITION x_estimate = a22 * x0 + b21 * u ;
 TRANSITION
 [
   a22 * x0' + b21 * u' - y1' <= 2*bound AND
   a22 * x0' + b21 * u' - y2' <= 2*bound AND
   a22 * x0' + b21 * u' - y1' >= -2*bound AND
   a22 * x0' + b21 * u' - y2' >= -2*bound) -->
    x0' IN {z: REAL | TRUE };
  []
  a22 * x0' + b21 * u' - y1' <= 2*bound AND
  a22 * x0' + b21 * u' - y3' <= 2*bound AND
```

```
  a22 * x0’ + b21 * u’ - y1’ >= -2*bound AND
  a22 * x0’ + b21 * u’ - y3’ >= -2*bound -->
    x0’ IN {z: REAL | TRUE };
  []
  a22 * x0’ + b21 * u’ - y2’ <= 2*bound AND
  a22 * x0’ + b21 * u’ - y3’ <= 2*bound AND
  a22 * x0’ + b21 * u’ - y2’ >= -2*bound AND
  a22 * x0’ + b21 * u’ - y3’ >= -2*bound) -->
    x0’ IN {z: REAL | TRUE };
 ]
END;
```

System Model and Property

```
system: MODULE = plant || sensor_attack || estimator ||
(WITH OUTPUT y2: REAL RENAME y TO y2 IN sensor_normal[1]) ||
(WITH OUTPUT y3: REAL RENAME y TO y3 IN sensor_normal[2]) ;

correct : THEOREM
  system |- X( G( velocity - x_estimate <= 3*bound ) );
```

Proof of Correctness Claim

Prove the desired bounded error property

sal-inf-bmc -d 2 -i landsharkPennEstimator correct

Check that we can not prove false properties

```
wrong: THEOREM
  system |- X( G( velocity - x_estimate <= 0.2 ) );
```

sal-inf-bmc -d 2 landsharkPennEstimator wrong

Check that the model does not deadlock

sal-inf-bmc -d 10 landsharkPennEstimator wrong

Full SAL Model

```
landsharkPennEstimator: CONTEXT =
BEGIN

SensorType: TYPE = [1..2] ; % Two good sensors
bound: REAL = 0.1 ; % Noise bounds

a11: REAL = 1.0 ; % A matrix
a12: REAL = 0.0194 ;
a21: REAL = 0.0 ;
a22: REAL = 0.94 ;

b11: REAL = 0.00175 ; % B matrix
b21: REAL = 0.174 ;

plant: MODULE = ...

sensor_attack: MODULE = ...
```

```
sensor_normal[i:SensorType]: MODULE = ...

estimator: MODULE =  ...

system: MODULE = ...

correct : THEOREM ...

wrong: THEOREM ...

END
```

Sal Infinite Bounded Model Checker

The analysis flow (all steps are automatic):

	Hybrid system = Composition of (hybrid) automata
HybridSal (Time-aware) rel. abs.	$\downarrow$
Sal language	Infinite state transition system
sal-inf-bmc/k-ind	$\downarrow$
Yices language	Eϕ
yices	$\downarrow$
	Proved/Counter-example

Conclusions

Correctness of robust state estimation procedure is easy – for particular instances

k-induction is an effective technique

- works for infinite state systems
- correctness mostly oblivious to initial state(s)

SAL used for discrete-time model

HybridSal would be required if continuous-time model were used