MathWorks Automotive Conference

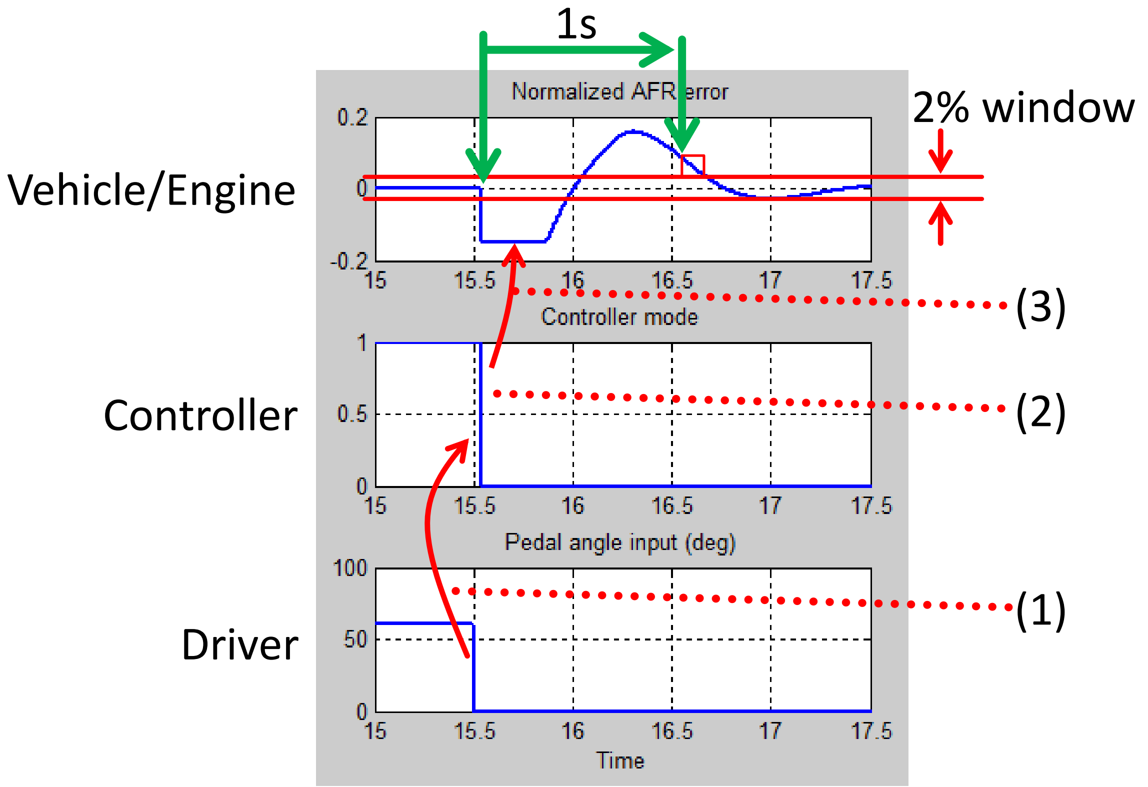
Simulation-Guided Verification & Validation for Large-Scale Automotive Control Systems

Hisahiro “Isaac” Ito, Jim Kapinski, Jyotirmoy Deshmukh, Xiaoqing Jin, Ken Butts

TOYOTA TECHNICAL CENTER

May 12, 2015
Plymouth, MI, USA

Motivation … System-Level Control Requirement Development



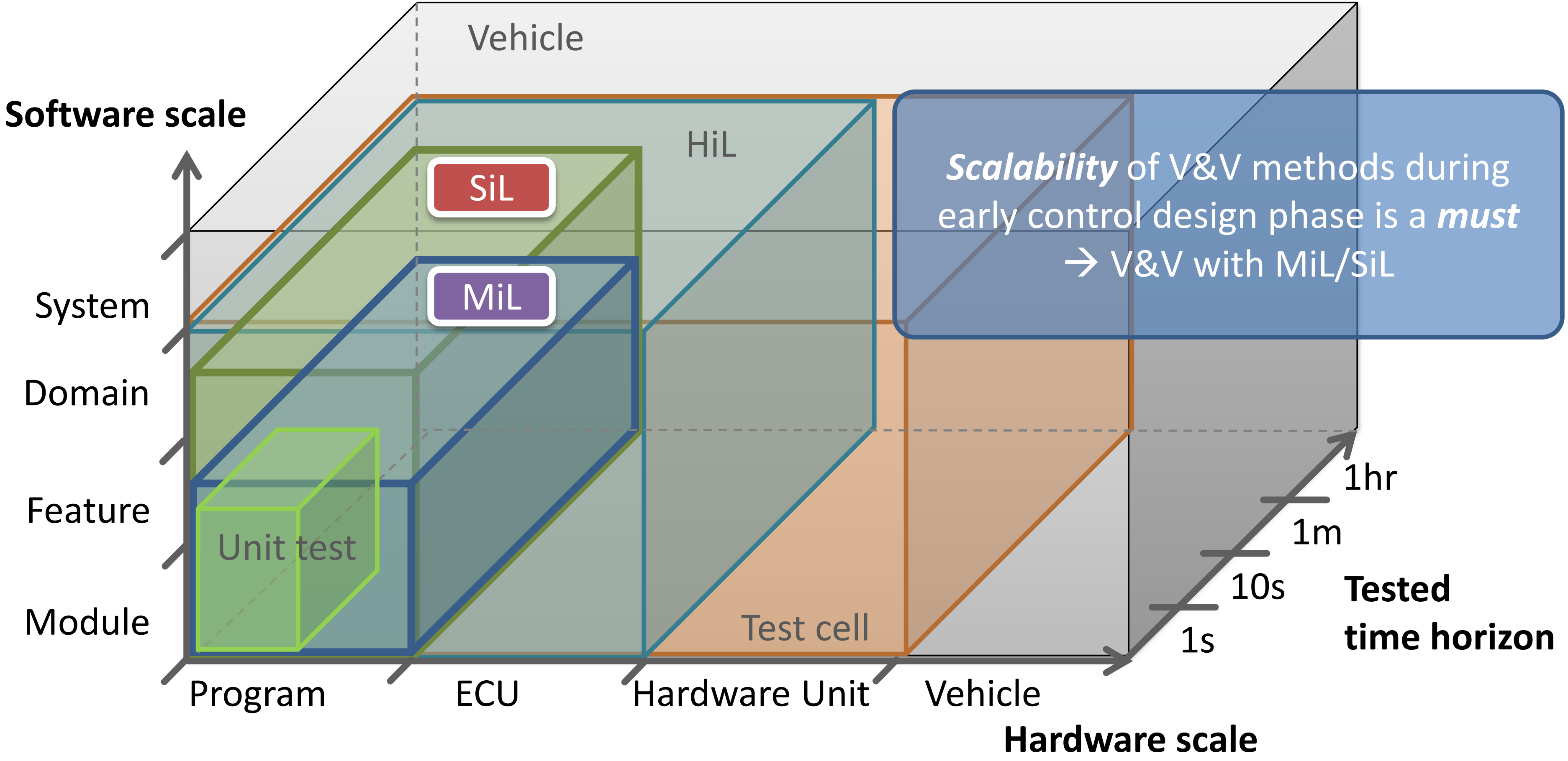
Example scenario

Suppose air-to-fuel ratio should settle within x% error window in τ seconds after a certain controller mode change…

Verifying a part of software in the controller does not help frontload the system level requirement development.

System-level requirement development needs to deal with driver, controller and plant.
How to do it in early control design phase?

Scales of Testing in Automotive Control System Development

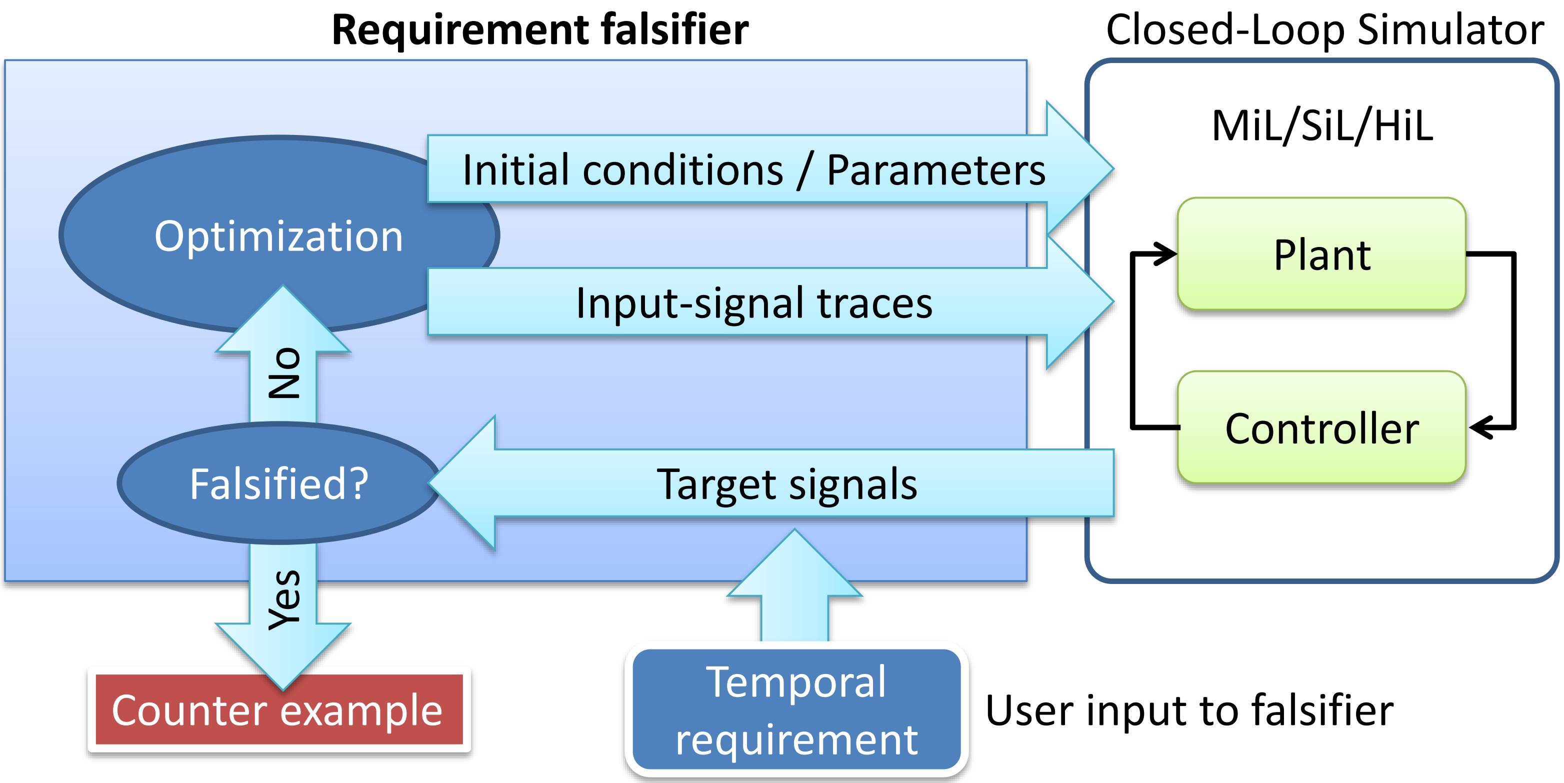


Scalability of V&V methods during early control design phase is a ***must***
→ V&V with MiL/SiL

Overview

- Requirement Falsification using S-TaLiRo
  - Example – Benchmark Closed-Loop Model for Air-to-Fuel Ratio Control
  - Designing Requirements using Metric Temporal Logic & Signal Temporal Logic
  - Setting up Falsification Process
  - Example outputs from S-TaLiRo
  - Potential Improvement Points
- Requirement Mining using Breach
- Summary & Conclusion

An Emerging, Scalable V&V Method – Requirement Falsification Simulation



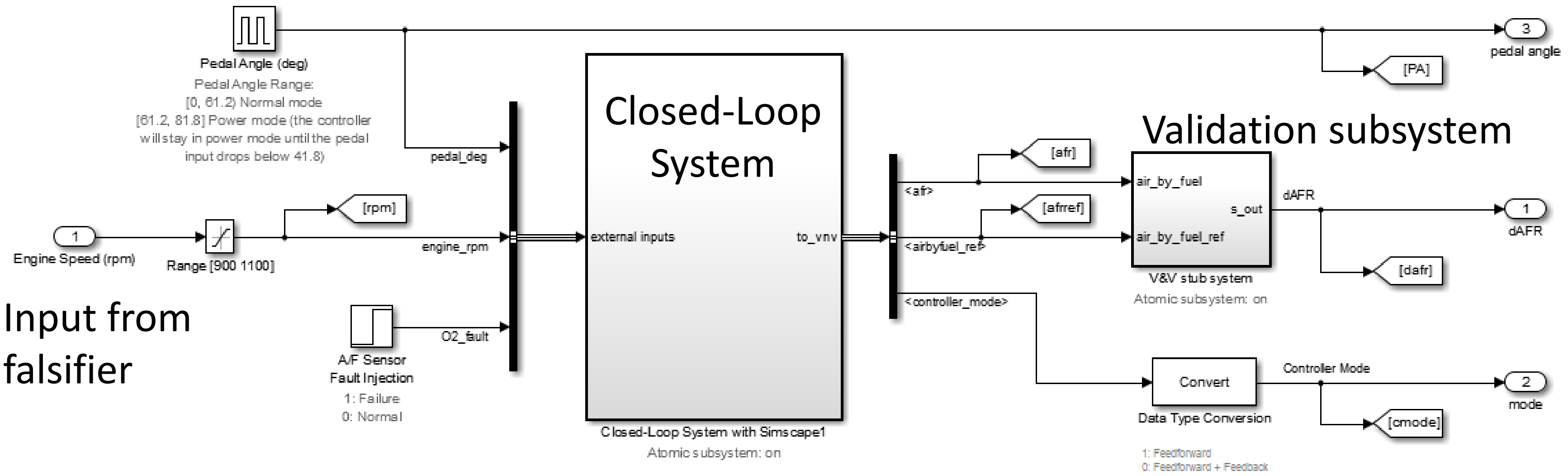
S-TaLiRo (Arizona State U, Colorado U)

`https://sites.google.com/a/asu.edu/s-taliro/s-taliro`

Breach (UC Berkeley)

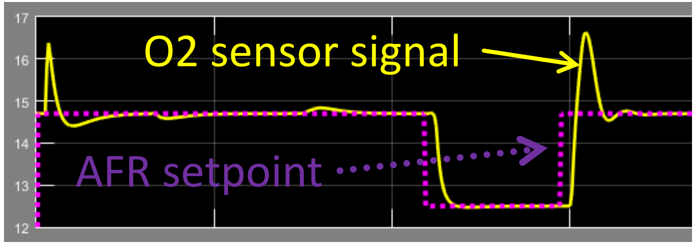
`http://www.eecs.berkeley.edu/~donze/breach_page.html`

Example – Air-to-Fuel Ratio Control Validation



Input from falsifier

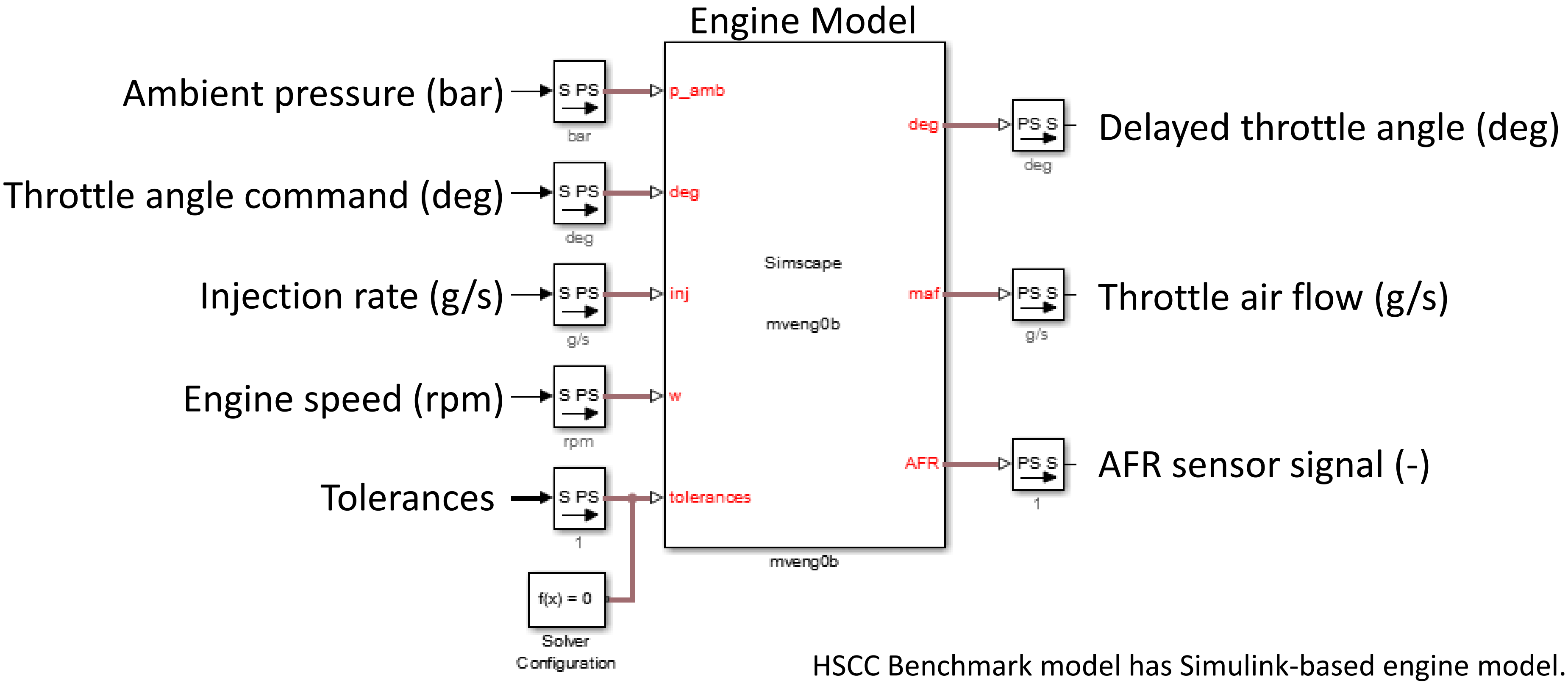
Outputs to falsifier



time

The model is based on HSCC 2014 benchmark model:
Jin, X., Deshmukh, J. V., Kapinski, J., Ueda, K., Butts, K., “Powertrain Control Verification Benchmark”, HSCC 2014

Engine Model Input/Output

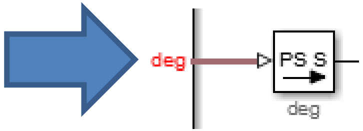


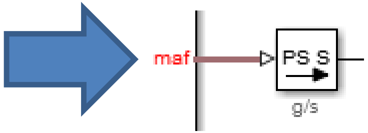
HSCC Benchmark model has Simulink-based engine model.
It was replaced with Simscape-based model here.
Their simulation results are identical.

Engine Dynamics 1 of 2 … Throttle, Intake & Fuel Injection

Throttle Air Flow Rate

```
der(delayed_thr) == {-10 '1/s'}*(delayed_thr - thr_cmd);
thr_deg_noguard == delayed_thr + thr_rest;
%...
thetaHat == c6 + c7*thr_deg + c8*thr_deg^2 + c9*thr_deg^3;
%...
mdot_thr == {1 'g/s'}*dir*thetaHat*2*sqrt(p_d/p_u - (p_d/p_u)^2);
%...
```





Intake Manifold Pressure

```
der(p_mani) == RT_V*(mdot_thr - mdot_air_tocyl);  % pdot=(R*T/V)*mdot
mdot_air_tocyl == tol_pump*(c2 + c3*w*p_mani + c4*w*p_mani^2 + c5*w^2*p_mani);
```

Fuel Injection and Port Wet

```
kappa == tol_kappa * tablelookup(kappa_x1data, kappa_x2data, kappa_ydata, ...
                                 eng_rpm, cyl_chg);
tau_ww == tol_tau_ww * tablelookup(tau_ww_x1data, tau_ww_x2data, tau_ww_ydata, ...
                                   eng_rpm, cyl_chg);
der(m_fuel) == (1-kappa)*inj_cmd - m_fuel/tau_ww;
mdot_fuel_tocyl == kappa*inj_cmd + m_fuel/tau_ww;
```

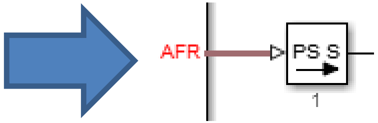
Engine Dynamics 2 of 2 … Mean-Value Cylinder & Exhaust Dynamics

Simplistic Mean-Value Cylinder

```
cyl_chg == mdot_air_tocyl/w * (4*pi) / Ncyl;  % Air charge per cylinder
cyl_afr == mdot_air_tocyl / mdot_fuel_tocyl;
```

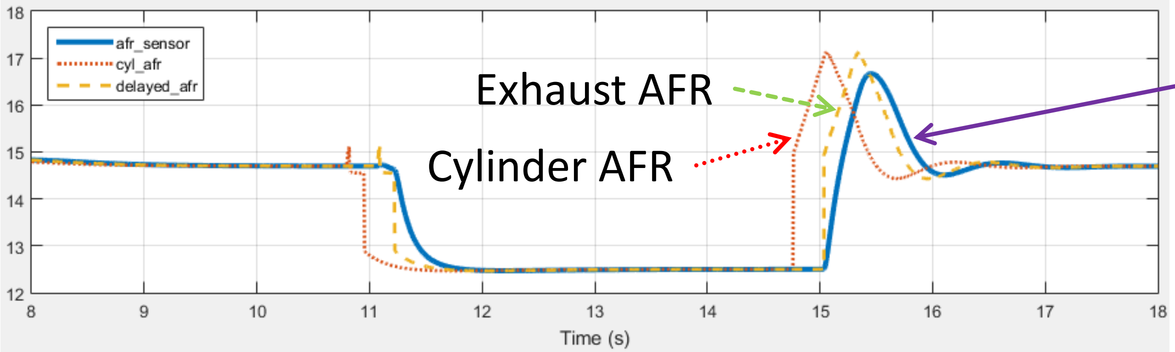
Exhaust AFR

```
cyl_delay == tablelookup(cyl_delay_x1data, cyl_delay_x2data, cyl_delay_ydata, ...
                         eng_rpm, cyl_chg);
delayed_afr == delay(cyl_afr, cyl_delay, ...
                     History=cyl_afr_pre, MaximumDelay=afr_delay_max);
der(exh_afr) == {-10 '1/s'}*(exh_afr - delayed_afr);
der(afr_sensor) == {-50 '1/s'}*(afr_sensor - exh_afr);
```



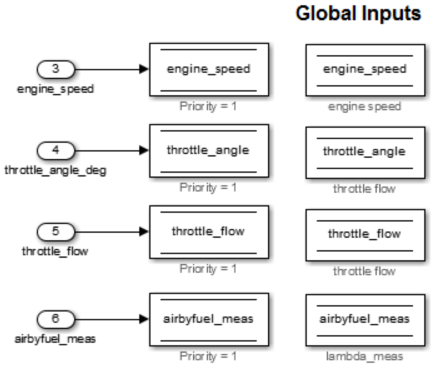
e.g.

Engine speed 1000RPM



O2 sensor signal

Controller (Sample Model)

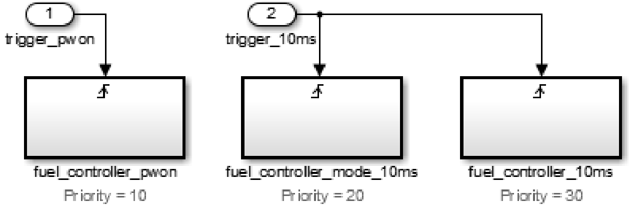


Inputs to Controller

- Engine speed (rad/s)
- Throttle angle (deg)
- Throttle air flow (g/s)
- O2 sensor signal (-)

Tasks

- Power-on function
- Controller mode (normal or power) @10ms
- Injection command @10ms

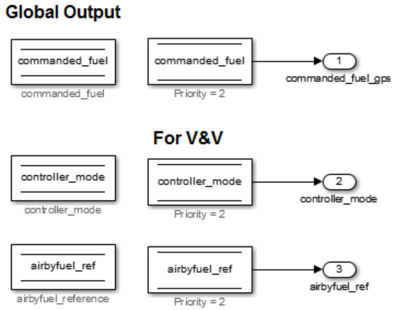


Output from Controller

- Injection command (g/s)

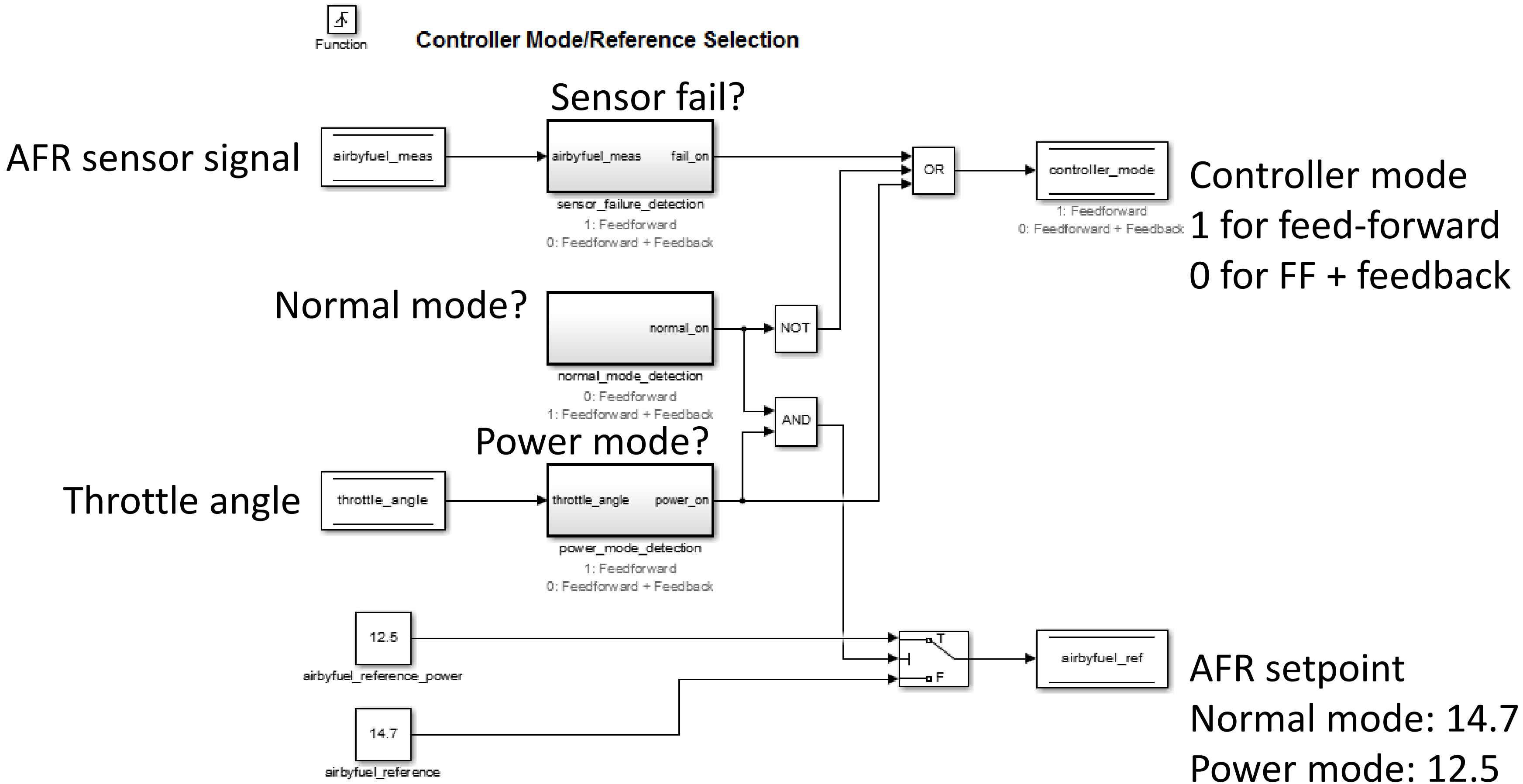
Output for Validation

- Fuel control mode
- AFR setpoint

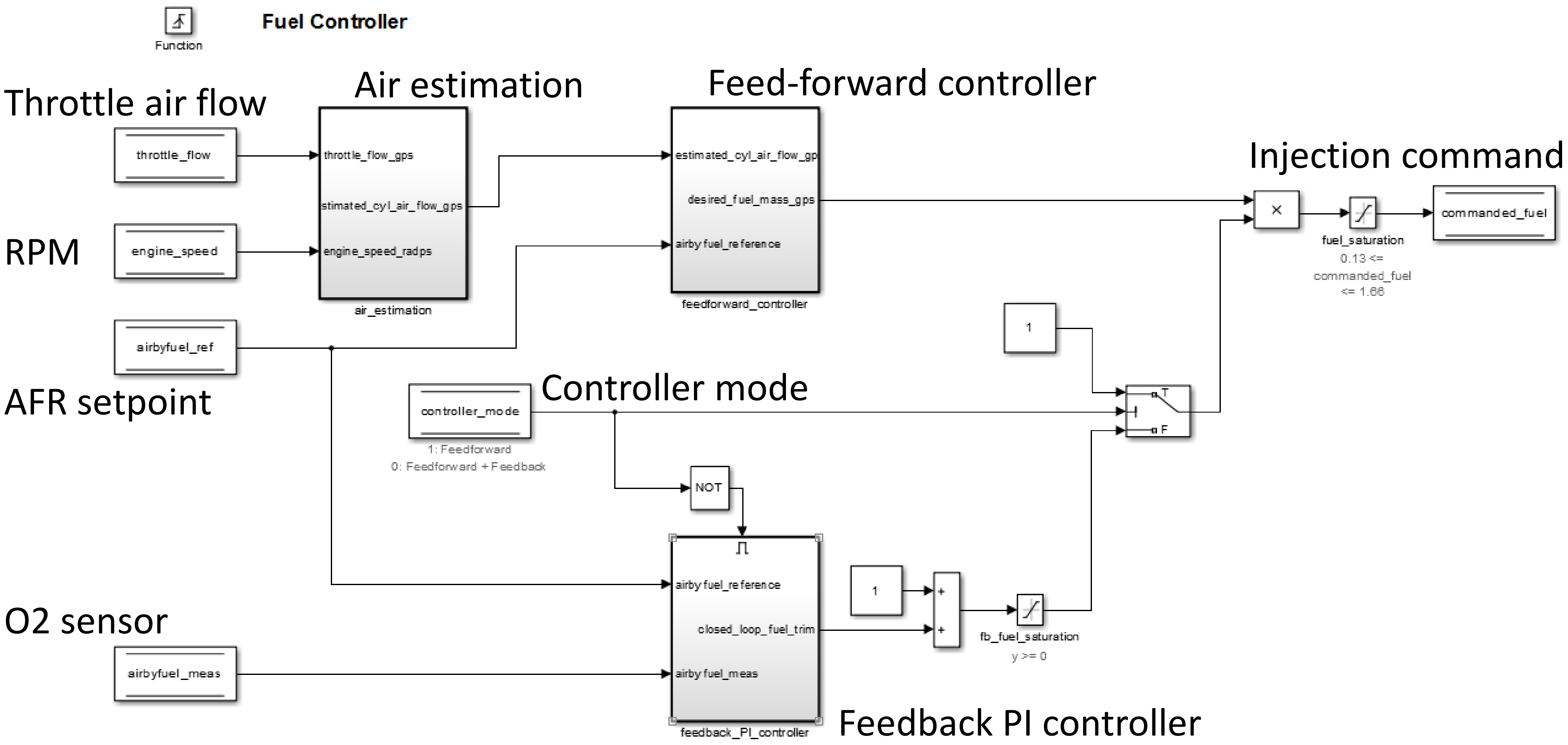


Following MAAB Guideline
Control model architecture, Type A

Controller Mode/Reference Selection .. 10ms timer



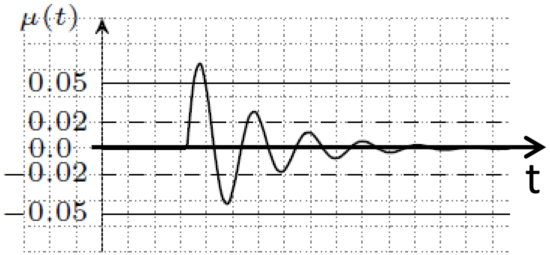
Fuel Controller .. 10ms timer



Designing Requirements

e.g. $A \Rightarrow B$

If A happens, B must happen.



Normalized AFR error

$$\mu(t) = \frac{\lambda_{sens}(t) - \lambda_{ref}(t)}{\lambda_{ref}(t)}$$

e.g. Settling Time Requirement

- A: Control mode switches from "power" to "normal" within 20ms.
- B: μ must settle within ± 0.02 within 1 second and must stay there for 4 seconds.
- Always "$A \Rightarrow B$" must be true, i.e., whenever A happens, B must happen.

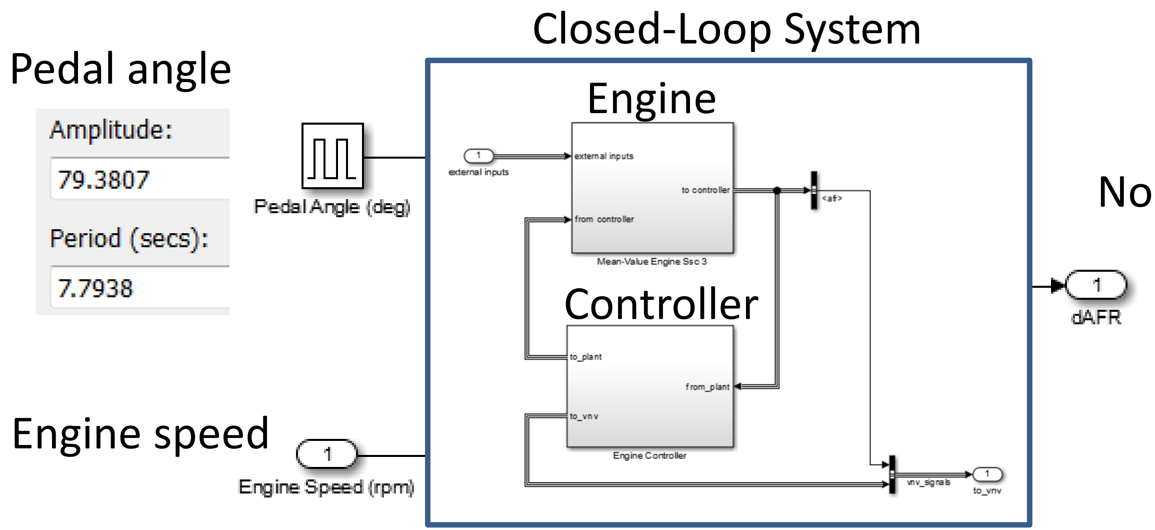
Metric Temporal Logic (MTL) and Signal Temporal Logic (STL) allow the description of temporal properties like above in a machine readable manner:

$$\varphi \coloneqq always\big(\underbrace{\ell = \text{power} \wedge eventually_{(0,0.02)}\ell = \text{normal}}_{A} \Rightarrow \underbrace{always_{(1,5)}|\mu| < 0.02}_{B}\big)$$

MTL/STL can represent system-level real-time control requirements.

Requirement Falsification

e.g. Transient requirement $\varphi := always|\mu| < 0.02$



Normalized AFR error

$$\mu(t) = \frac{\lambda_{sens}(t) - \lambda_{ref}(t)}{\lambda_{ref}(t)}$$

Initialize → Simulate → Estimate robustness → Falsified? —Yes→ Counter example found

e.g. $\min_t(\mu - 0.02)$

No → Initialize

Requirement falsifier tries to falsify requirements by simulation.
This is not property proving, not exhaustive, but can handle large-scale system.

Writing Temporal Requirement for S-TaLiRo

e.g. Settling Time Requirement

$$\varphi \coloneqq always\big(\ell = \text{power} \wedge eventually_{(0,0.02)}\ell = \text{normal} \Rightarrow always_{(1,5)}|\mu| < 0.02\big)$$

```
phi = ['[] ((pwr /\ <>_(0, 0.02) norm) -> ([]_(1, 5) mulow /\ muhigh))'];
```

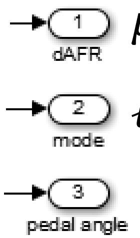
Predicates

```
preds(1).str = 'norm';
preds(1).A = [0 1 0];
preds(1).b = 0.5 ;

preds(2).str = 'pwr';
preds(2).A = [0 -1 0];
preds(2).b = -0.5 ;

preds(3).str = 'muhigh';
preds(3).A =  [1 0 0];
preds(3).b =  0.02 ;

preds(4).str = 'mulow';
preds(4).A =  [-1 0 0];
preds(4).b =  0.02 ;
```



μ x_1

ℓ x_2

x_3

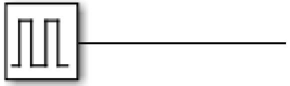
$A \cdot x \leq b$

Constraint

norm	pwr	mulow	muhigh
$\ell \leq 0.5$	$0.5 \leq \ell$	$-0.02 \leq \mu$	$\mu \leq 0.02$

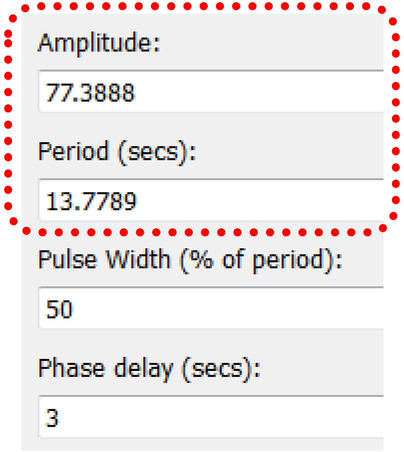
Designing Parameters & Initial Conditions

e.g. Pulse Generator Block



Pedal Angle (deg)

Pedal Angle Range:
[0, 61.2) Normal mode
[61.2, 81.8] Power mode (the controller will stay in power mode until the pedal input drops below 41.8)



Suppose falsifier wants to vary amplitude (deg) and period (s) for each simulation run.

X0(1) range

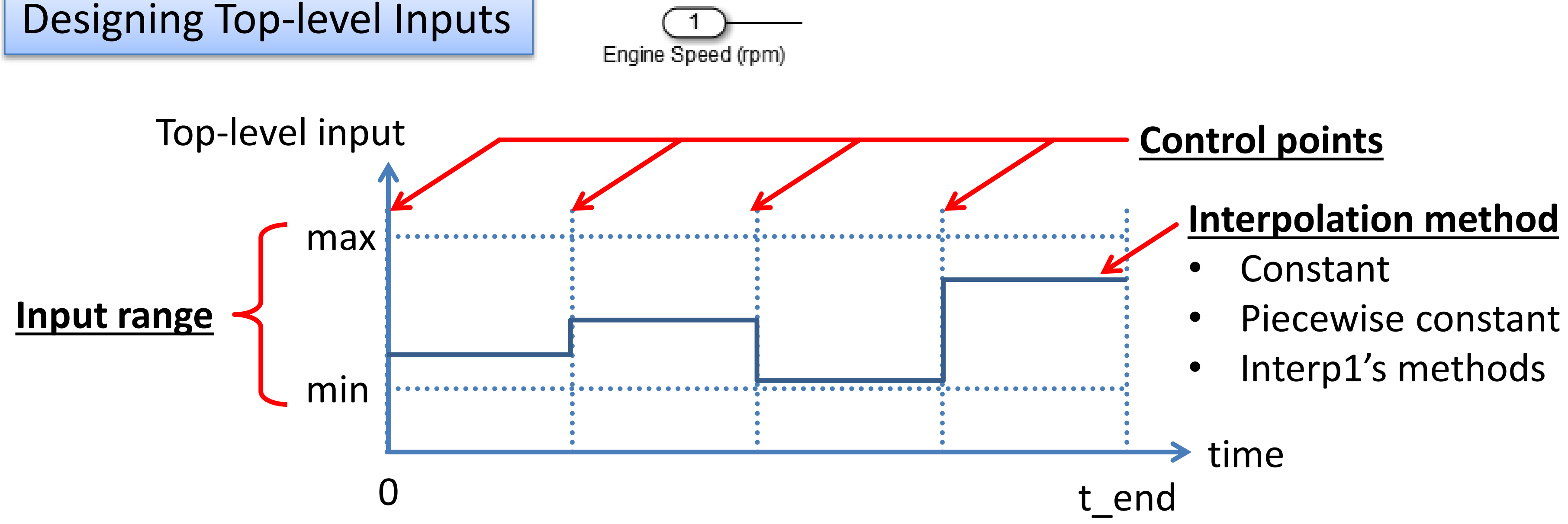
```
Initial_cond = [ 61.3  81.2 ; 10  20 ];
```

min max

```
set_param([model,'/Pedal Angle (deg)'],'Amplitude',num2str(X0(1)));
set_param([model,'/Pedal Angle (deg)'],'Period',num2str(X0(2)));
```

Initial conditions (block parameters) are randomly chosen within the specified range for each simulation run by S-TaLiRo.

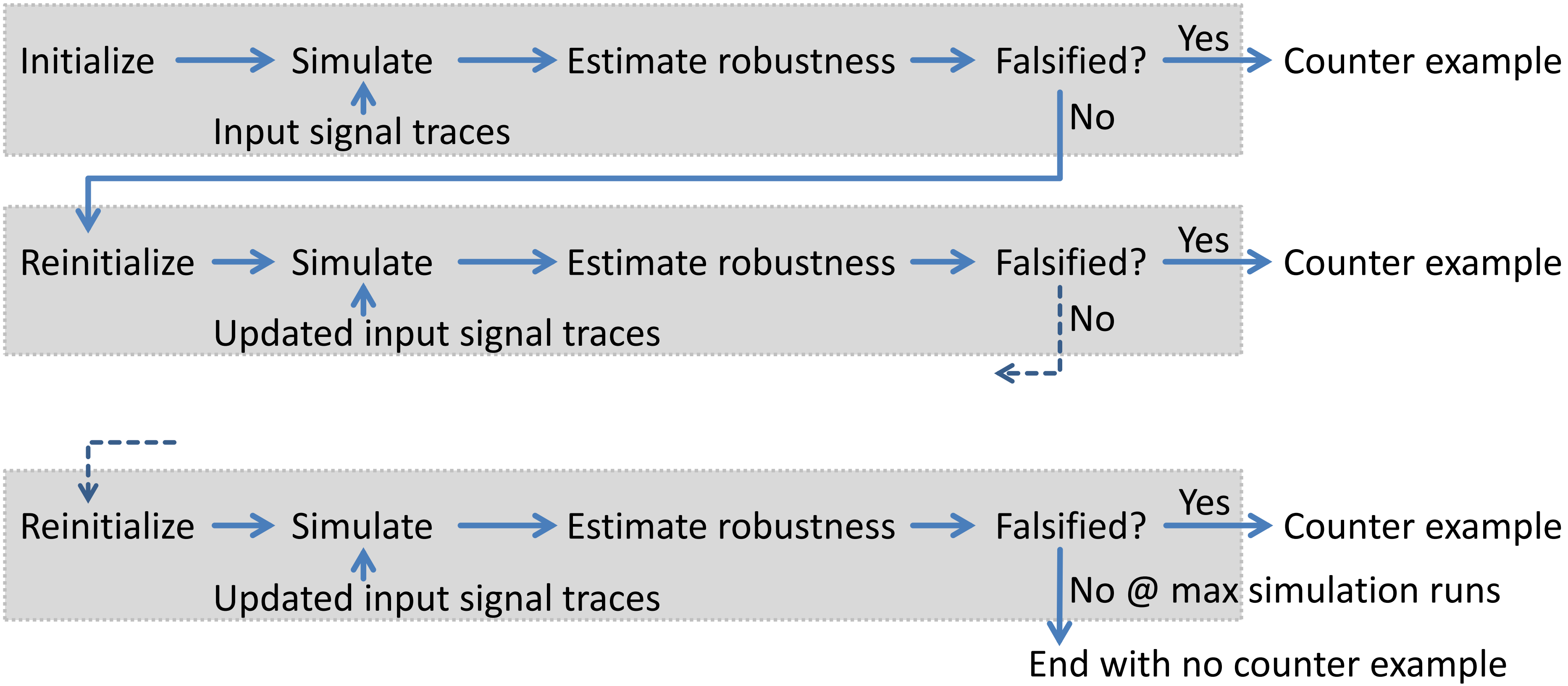
Designing Top-level Inputs




```
input_range = [ 900  1100 ];
cp_array = 1 ;
opt.interpolationtype={'const'};
```

Top-level inputs are manipulated by S-TaLiRo during a simulation run. The top-level inputs receives different input traces from S-TaLiRo for each simulation run.

Falsification Process



Example Outputs from S-TaLiRo, #1

e.g. Transient requirement

$$\varphi \coloneqq always_{(11,50)}|\mu| < 0.02$$ (Normal mode only)

Number of global resets

Best/min robustness value

Number of simulation runs

User-specified max simulation runs

Optimization progress hint

```
Run number 1 / 1
Best ==> 0.042666
Best ==> 0.04235
50 Acceptance Ratio=1 beta=-22.5,0.825
Best ==> 0.042269
Best ==> 0.042093
Best ==> 0.042061
100 Acceptance Ratio=1 beta=-33.75,0.9075
150 Acceptance Ratio=1 beta=-50.625,0.99
Best ==> 0.042037
200 Acceptance Ratio=1 beta=-75.9375,0.99
250 Acceptance Ratio=0.98 beta=-113.9063,0.99
300 Acceptance Ratio=0.96 beta=-170.8594,0.99
```

```
900 Acceptance Ratio=0.66 beta=-9852.6125,0.99
950 Acceptance Ratio=0.54 beta=-9852.6125,0.99
Best ==> 0.041919
1000 Acceptance Ratio=0.56 beta=-14778.9188,0.99
  Running time of run 1: 974.1314 sec
fx >>
```

No counter example was found

Example Outputs from S-TaLiRo, #2

e.g. Settling Time Requirement

$$\varphi \coloneqq always(\underbrace{\ell = \text{power} \wedge eventually_{(0,0.02)}\ell = \text{normal}}_{} \Rightarrow \underbrace{always_{(1,5)}|\mu| < 0.02}_{4s})$$

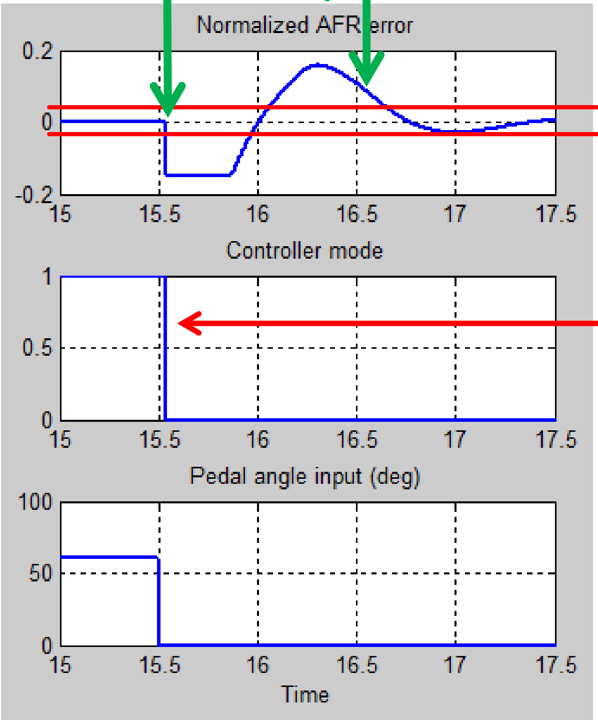
1s

4s

Negative robustness value

```
Run number 1 / 1
Best ==> -0.12968
FALSIFIED at sample ,3!
  Running time of run 1: 16.9284 sec
fx >> |
```

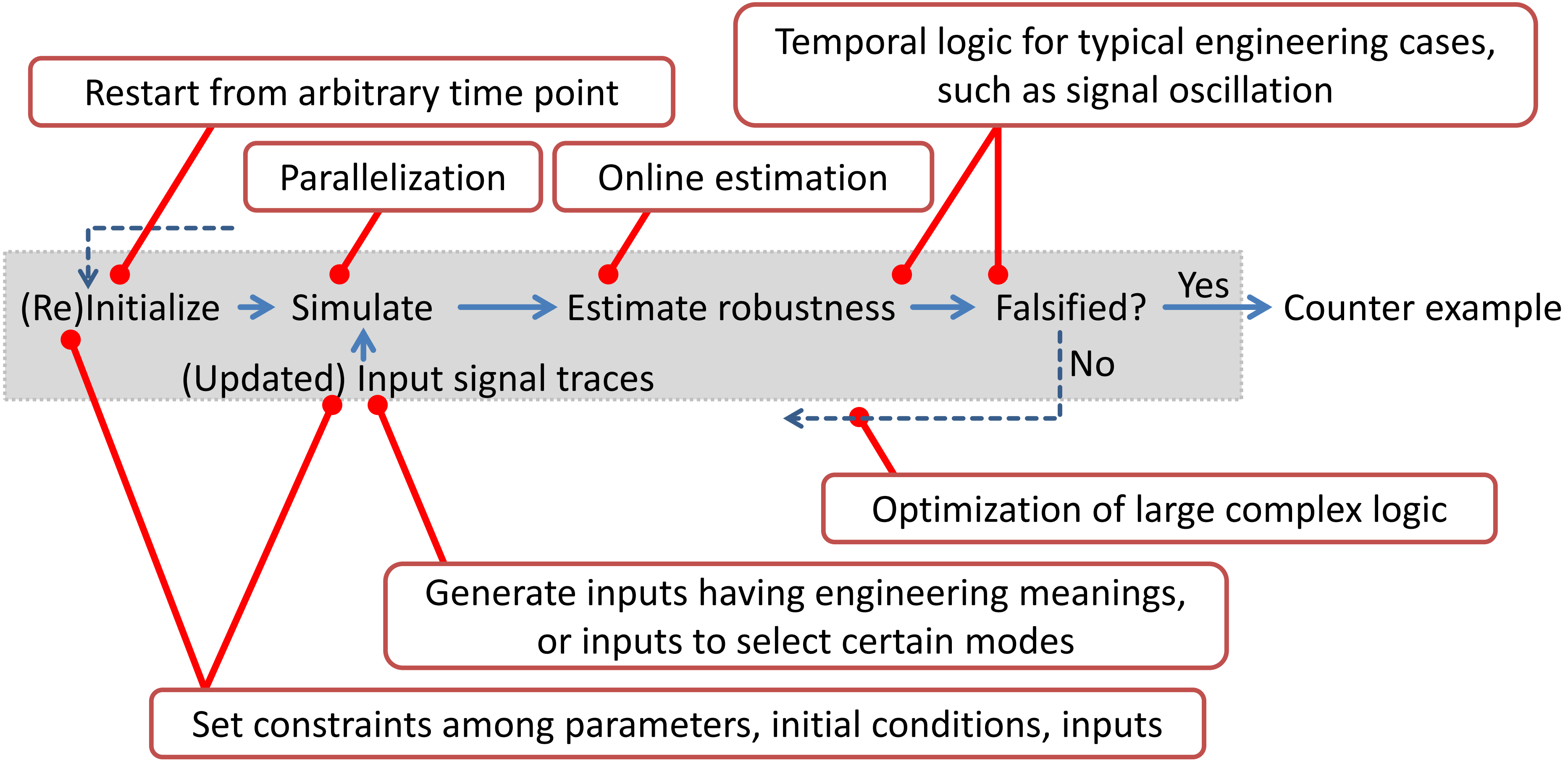
Falsified!



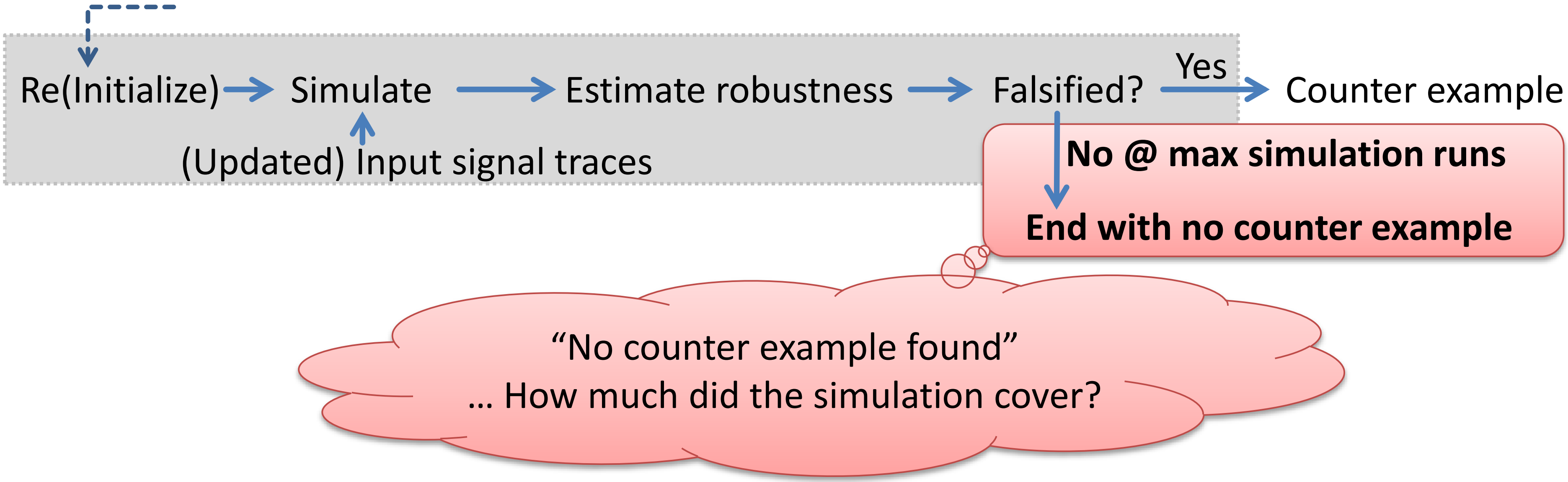
$|\mu| < 0.02$

Power mode (1) to Normal mode (0)

Potential Improvement Points

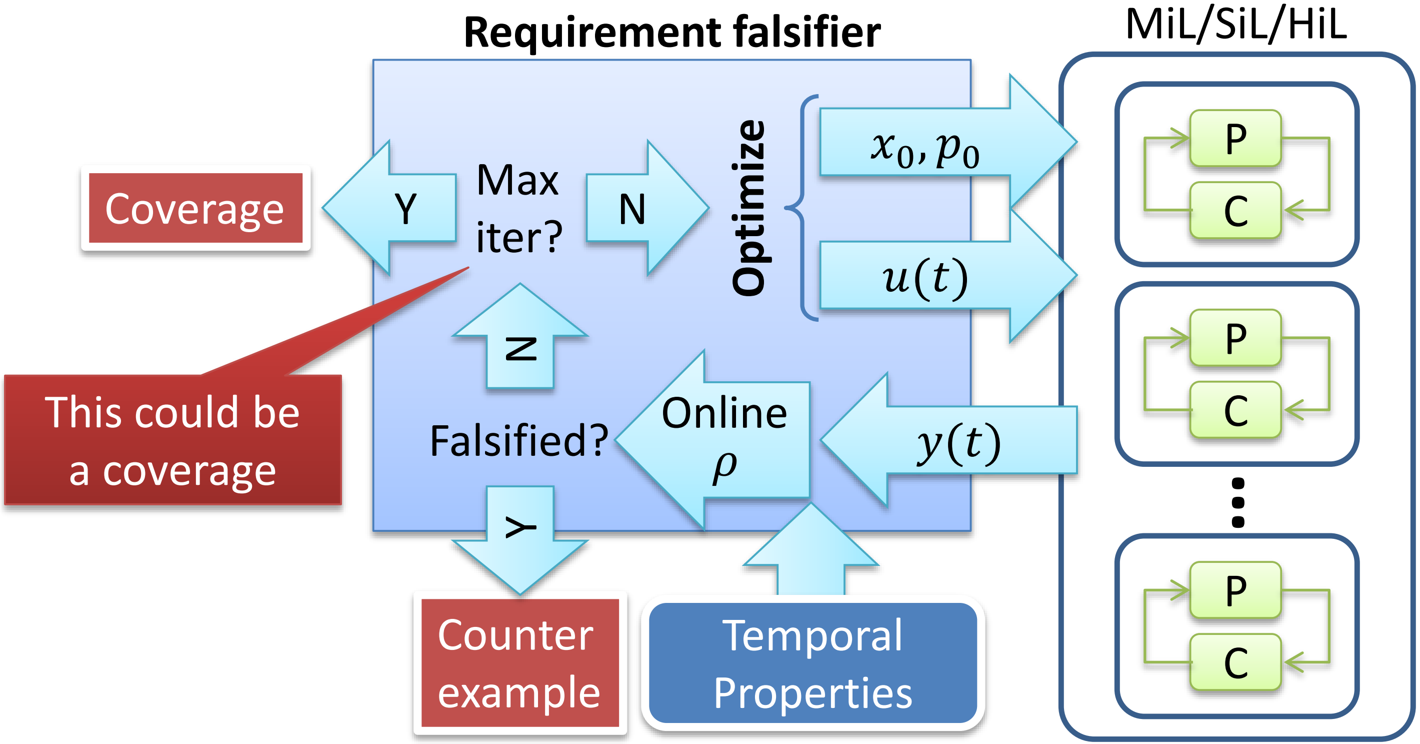


Coverage for No-counter Example Case



Some coverage could be used as a stop condition, rather than maximum simulation runs. However, such a coverage should cover both plant and controller.

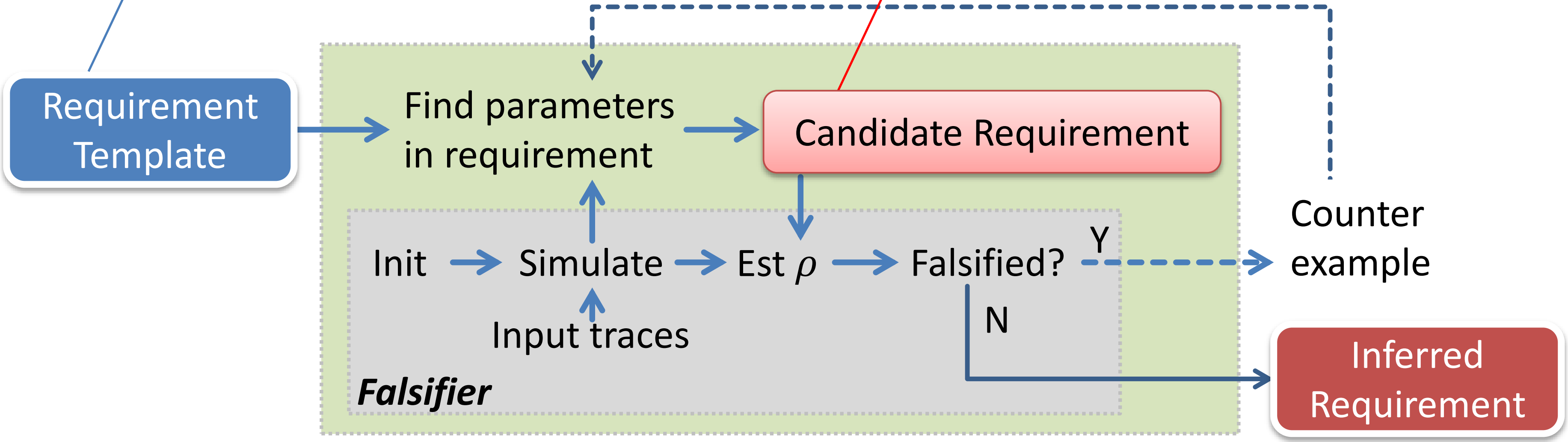
Simulation-guided V&V Framework … Revisited and Updated



Closed-loop simulator, plant modeling, code generation and high-performance computing are also very import technologies to realize practical V&V environment.

Application of Falsification – Requirement Mining by Breach

e.g. $alw_{[0,\tau_1]}\left(x_1 < \pi_1 \Rightarrow ev_{[0,\tau_2]}(x_2 > \pi_2)\right)$ $\quad$ $alw_{[0,3]}\left(x_1 < 700 \Rightarrow ev_{[0,0.2]}(x_2 > 0.1)\right)$



Potential uses of requirement mining:

- Worst-case testing
- Signal range mining

Summary & Conclusion

- Simulation-guided V&V methods were introduced.
  - Requirement falsification
  - Requirement mining
  - (There are other simulation-guided V&V methods.)
  - Not proving, not exhaustive, but can handle large-scale system
- MTL/STL can represent system-level real-time control requirements.
- Potential improvement points were identified.
  - In dire need of a good engineering coverage.

- Given its scalability, simulation-guided V&V such as requirement falsification is promising and already practical technology for large-scale control system development.
- Gaps towards widespread use in industry need to be filled fast.

Thank you.