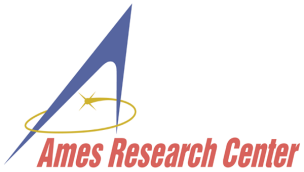
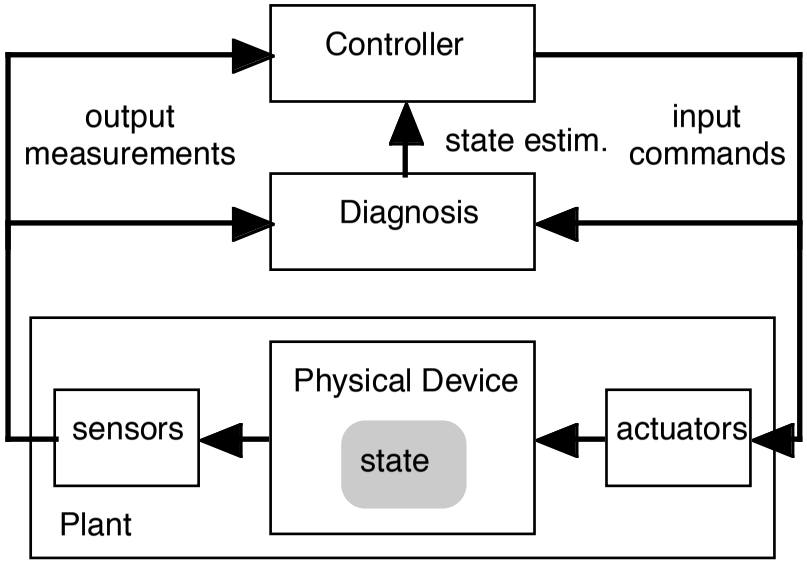


Formal Verification of Diagnosability via Symbolic Model Checking

Charles Pecheur (RIACS / NASA Ames)
Alessandro Cimatti (IRST)

Diagnosis

- *Diagnosis* estimates the state of *Plant*
- Partial observability: *state* is hidden, only *input* and *output* are visible
- Goal: prevent faults (deflated tire) from becoming failures (car crash) by detecting and identifying them timely and appropriately
- How can we verify that?



Ames Research Center

Diagnosis Model

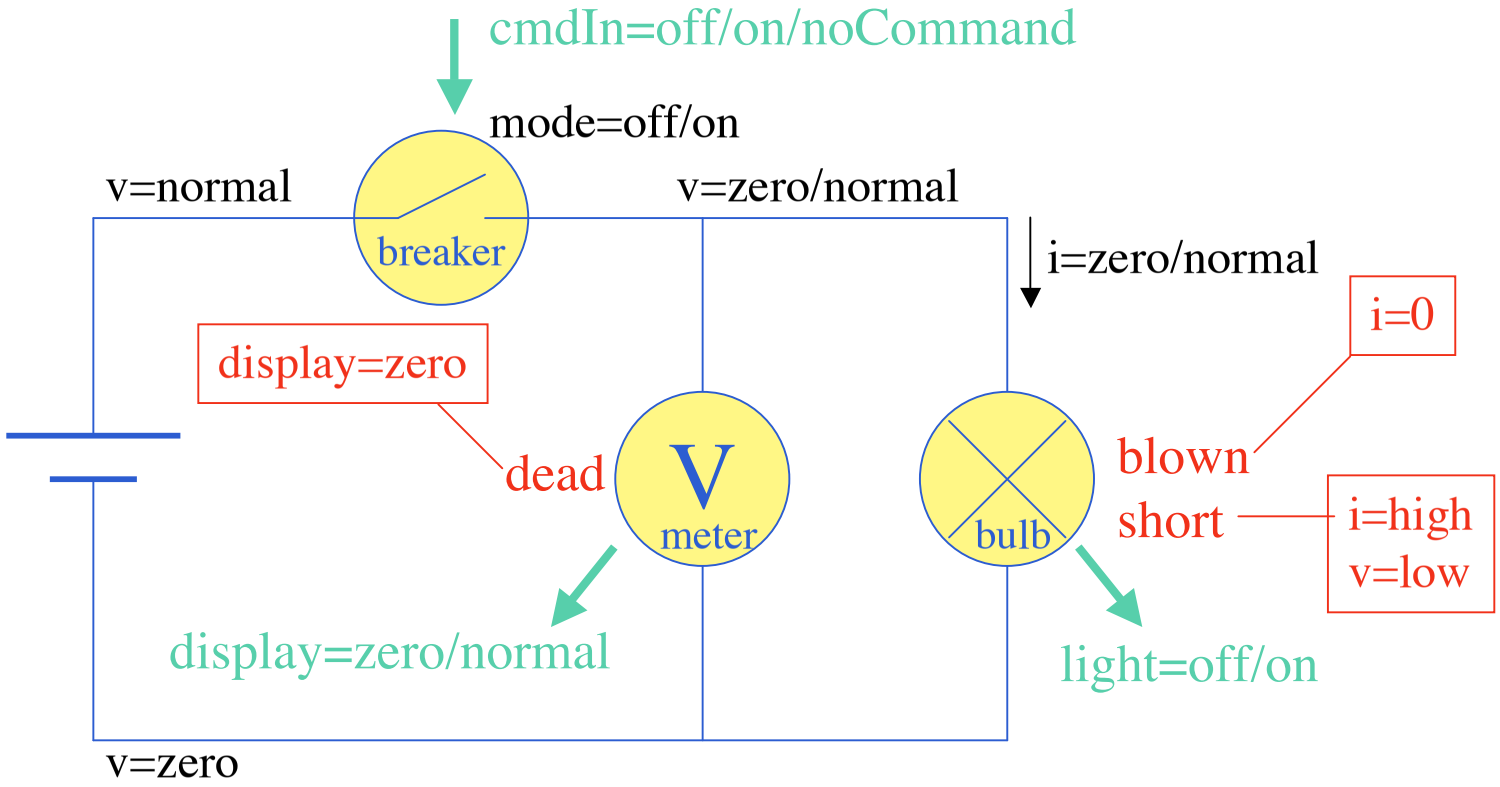
We assume that we have a **model** of the diagnosed system

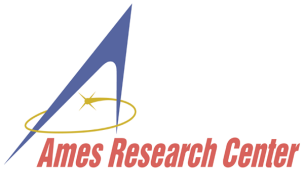
- models nominal and faulty behavior
- identifies hidden and observable parts
- we focus on *discrete* and *finite* models
- Same model can be compiled/interpreted in model-based diagnosis system

Case in point: ***Livingstone***

- model-based diagnosis system (NASA Ames)
- generic engine interprets application-specific model
- qualitative (discrete, finite) model

Elec: A Sample Model





Verification of Diagnosis systems

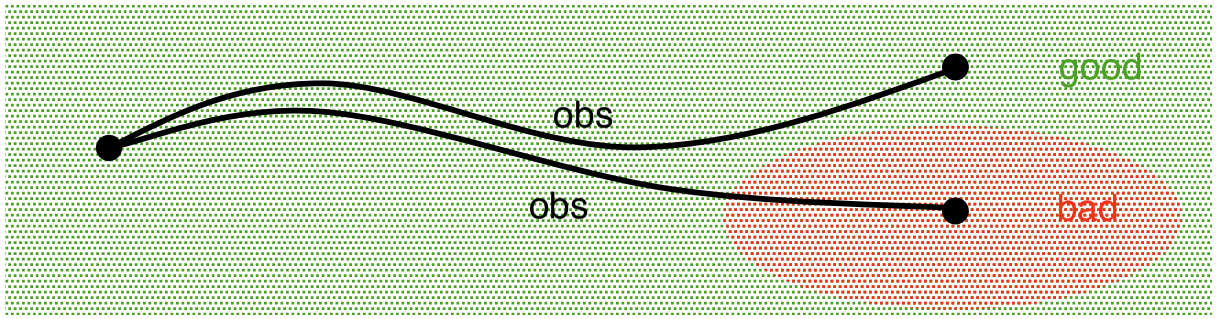
Verify what?

1. Model Correctness: the model is OK
   i.e. the model is a valid abstraction of the plant
2. Engine Correctness: the software is OK
   i.e. all that can be diagnosed is correctly diagnosed
3. **Diagnosability: the design is OK**
   i.e. all that needs to be diagnosed can be diagnosed

In principle, 1+2+3 => diagnosis will be correct

Here we look at 3 only!

Diagnosability as Reachability



Diagnosis can always tell when plant comes to a bad state

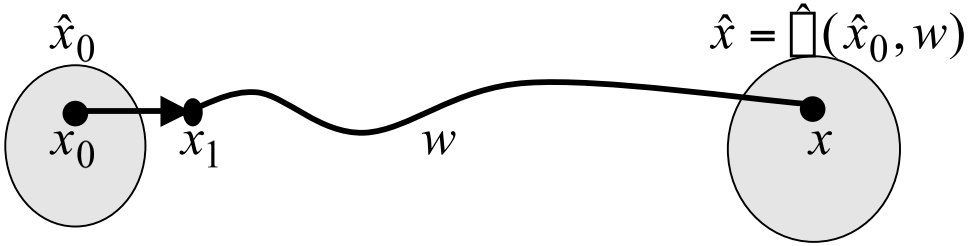
iff Observations always allow to tell when plant is in bad state

iff Whenever bad state is reached, no good state could have been reached with the same observations

iff There is no pair of executions, one reaching a bad state, the other reaching a good state, with identical observations

- ... within a specific context
- variant: tell between two kinds of bad

Formalization



Transition system $x \Box \xrightarrow{u/v} \Box x'$, execution $\rho : x_0 \Box\Box^{w} \Box x$

trace w is visible, states x, x' are hidden

Diagnosis function $\hat{x} = \hat{\Box}(\hat{x}_0, w)$

updates belief state according to observed trace

Correct iff $x_0 \Box \hat{x}_0, x_0 \Box\Box^{w} \Box x \Box x \Box \hat{x}$

does not lose the actual state

Perfect diagnosis $\Box_P(\hat{x}_0, w) = \{x \mid \Box x_0 \Box \hat{x}_0 . x_0 \Box\Box^{w} \Box x\}$

the best possible knowing the transition system

Diagnosis Condition

Diagnosis condition $c_1 \not\equiv c_2$ in context $C=(\Box_C, \Box_C)$:

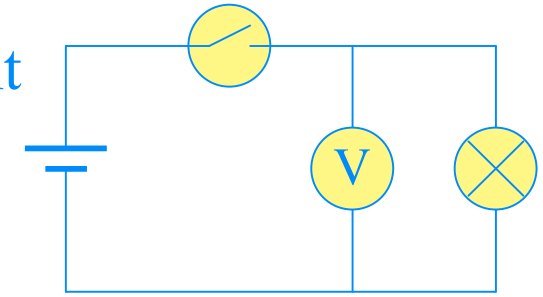
- Distinguish between two state conditions $c_1(x)$, $c_2(x)$
- ... assuming executions in $\Box_C$
- ... and initially indistinguishable states $x \Box_C x'$ (equiv.)

Fault detection: $fault \Box\Box fault$

Fault diagnosis: $fault_1 \Box fault_2$

Example:

- Distinguish between current and no current
- ... assuming single faults
- ... and known initial breaker state



Formalization (cont'd)

$$\hat{x} \models c_1 \Box c_2 \quad \text{iff} \quad \hat{x} \Box c_1 = \varnothing \quad \hat{x} \Box c_2 = \varnothing$$

no ambiguity between c_1 and c_2

$$\hat{x}_0 \models \Box_C \quad \text{iff} \quad \hat{x}_0 \Box \hat{x}_0 \Box \Box_C$$

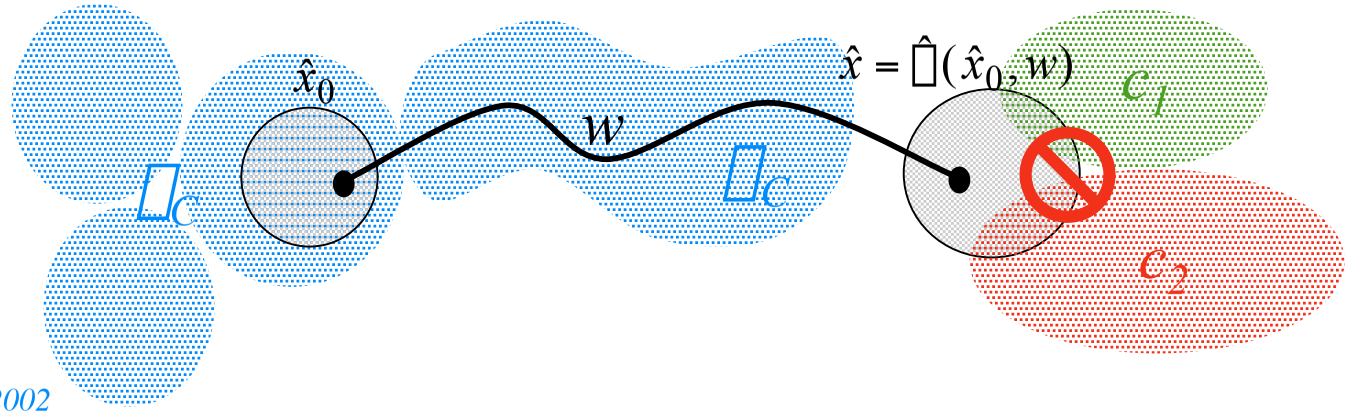
initial belief compatible with equivalence $\Box_C$

$$(\hat{x}_0, w) \models (\Box_C, \Box_C) \quad \text{iff} \quad \hat{x}_0 \models \Box_C \Box\Box : x_0 \Box\Box^{w} \Box \; x.x_0 \Box \hat{x}_0 \Box \Box \Box$$

idem. and trace compatible with some execution in $\Box_C$

$$\hat{\Box}, (\Box_C, \Box_C) \models c_1 \Box c_2 \quad \text{iff} \quad (\hat{x}_0, w) \models (\Box_C, \Box_C) \Box \; \hat{\Box}(\hat{x}_0, w) \models c_1 \Box c_2$$

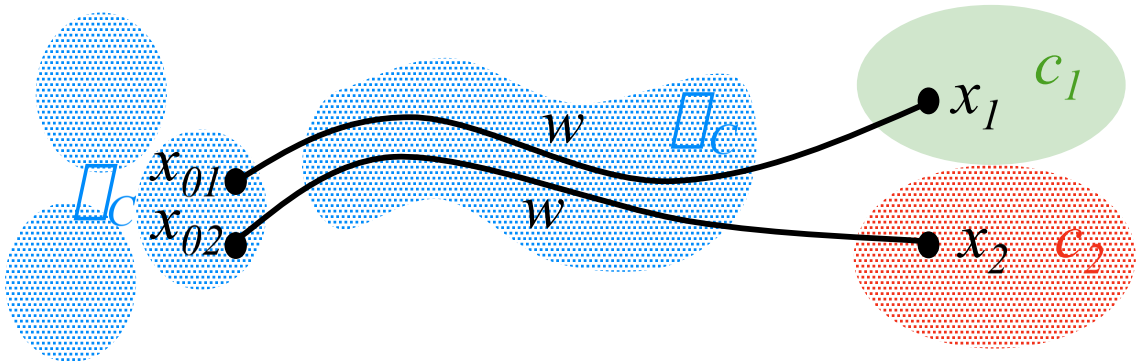
for all initial beliefs and executions within context, no ambiguity



Critical Pairs

Counter-example of a condition $c_1 \# c_2$ in context $(\Box_C, \Box_C)$:
a pair of executions $\Box_1 | \Box_2 : x_{01} | x_{02} \text{ –}w\text{–>} x_1 | x_2$ with the same observable trace w, such that

- $c_1(x_1)$ and $c_2(x_2)$, and
- ... $\Box_1, \Box_2 \Box \Box_C$, and
- ... $x_{01} \Box_C x_{02}$

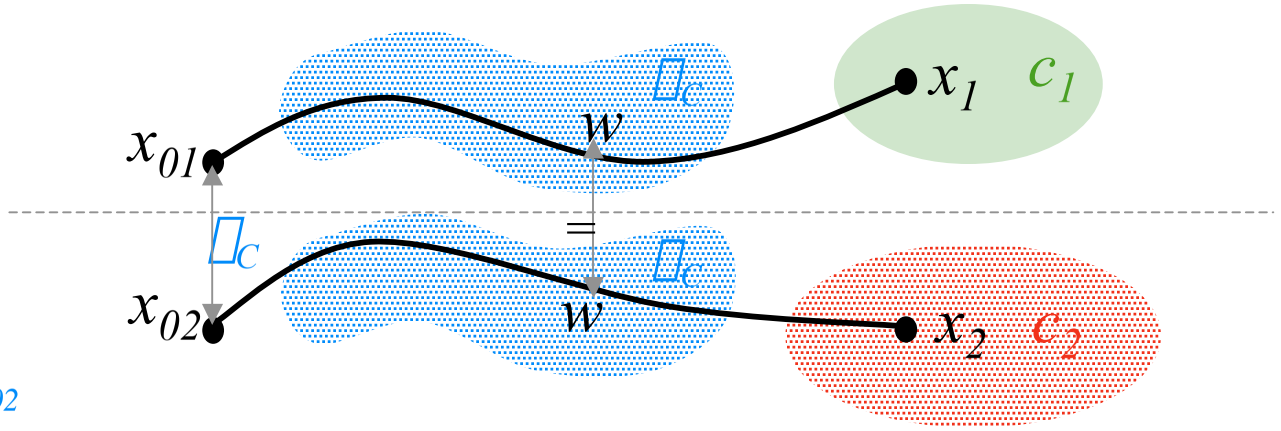


$c_1 \# c_2$ diagnosable in $(\Box_C, \Box_C)$ **iff** no critical pairs

Coupled Twin Model

- Coupled twin plant P^2 = two copies of the plant P with merged inputs and outputs

$c_1 \square c_2$ diagnosable in $(\square_C, \square_C)$
iff
$c_1 \square c_2$ not reachable from $\square_C$ through $\square_C \square \square_C$ in P^2



Model Checking of Diagnosability

- Coupled Twin Plant P^2 as Kripke Structure K_P
  - turn inputs/outputs into state variables
  - state vector $\langle v_{x1}, v_{x2}, v_u, v_y \rangle$
  - c_1, c_2 as propositional formulae $c_1(v_{x1})$, $c_2(v_{x2})$
  - Φ_C as propositional formula/constraint $\Phi_C(v_{x1}, v_{x2})$
  - $\Box_C$ as temporal formulae/constraints $\Box_C(v_{x1}, v_u, v_y)$, $\Box_C(v_{x2}, v_u, v_y)$

$c_1 \Box c_2$ diagnosable in $(\Phi_C, \Box_C)$

iff

$K_P \oplus \Box_C(v_{x1}, v_u, v_y) \oplus \Box_C(v_{x2}, v_u, v_y), \Phi_C(v_{x1}, v_{x2}) \models \Box$ **EF** $c_1(v_{x1}) \Box c_2(v_{x2})$

iff (if $\Box_C$ expressible in LTL)

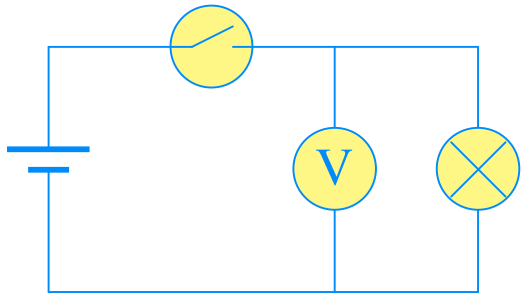
$K_P \models \Phi_C(v_{x1}, v_{x2}) \Box \Box_C(v_{x1}, v_u, v_y) \Box \Box_C(v_{x2}, v_u, v_y) \Box \Box$ **F** $c_1(v_{x1}) \Box c_2(v_{x2})$

error in paper

Model Checking Example (single model)

In SMV (Carnegie Mellon), verified using NuSMV (IRST)

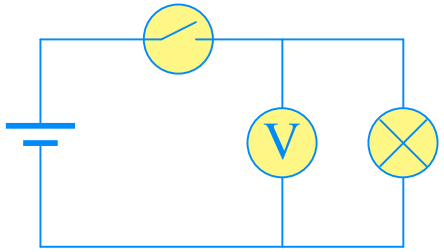
```
MODULE bulb
VAR light : {on,off};  VAR i : {zero,normal,high}; ...
VAR mode : {ok,blown,short};
DEFINE _broken := mode in {blown, short};
TRANS ...
INVAR ...

MODULE breaker ...
MODULE meter ...

MODULE elec ... -- single model
VAR meter : meter; VAR bulb : bulb; VAR breaker : breaker;
DEFINE _brokenCount := meter._broken + bulb._broken + breaker._broken;
```



Model Checking Example (twin model and spec)

```
MODULE main  -- twin model
VAR L : elec; -- "left" copy
VAR R : elec; -- "right" copy
-- coupled inputs and outputs
DEFINE same_commands := L.cmdIn = R.cmdIn;
DEFINE same_obs := L.light = R.light & L.display = R.display;
DEFINE same_diagnosis := same_commands & same_obs;
-- initial context: known modes
DEFINE same_modes := L.breaker.mode = R.breaker.mode &
       L.bulb.mode = R.bulb.mode & L.meter.mode = R.meter.mode;
-- trace context: no initial fault, single faults
DEFINE unbroken := !L._brokenCount & !R._brokenCount;
DEFINE single_faults := !L._brokenCount > 1 & !R._brokenCount > 1;
-- diagnosis condition: is there high current in the bulb?
DEFINE dontknow_high_i := L.bulb.i=high & !R.bulb.i=high;
```

Model Checking Example (specifications)



- Starting form known initial non-faulty state, tell whether there is high current in the bulb

```
SPEC (same_modes & unbroken) ->
        !E[same_diagnosis U
                same_diagnosis & dontknow_high_i]
```

  - false: if meter and bulb both fail...
- Idem with single faults

```
SPEC (same_modes & unbroken) ->
        !E[same_diagnosis & single_failures U
                same_diagnosis & single_failures & dontknow_high_i]
```

  - true

Ames Research Center

Model Checking Example (specifications cont'd)

- Idem remembering only the last two observations

```
SPEC (same_modes & unbroken) ->
        !E[single_failures U
                same_diagnosis & single_failures & EX (
                same_diagnosis & single_failures & EX (
                same_diagnosis & single_failures & dontknow_high_i))]
```

  - false: "forgets" the state of the circuit breaker
- Note: initial and trace contexts could be hardwired into the model
  - reduces the number of variables => improved performance
  - loses some flexibility w.r.t. properties that can be checked

Tools

- **NuSMV** (IRST): symbolic model checker
  - re-engineering of SMV (Carnegie Mellon)
  - supports both BDD and SAT
  - modular, documented, extensible, open-source
- **JMPL2SMV** (NASA Ames): translates Livingstone to SMV
  - enables diagnosability verification on Livingstone models
  - portable (Java)
- **Variable Elimination** (Bwolen Yang [CAV99]):
  - eliminates model variables by turning them into macros
  - implemented as customized (Carnegie Mellon) SMV
  - port to NuSMV under consideration

Ames Research Center

Experimental Results

- (Still) only early, small experiments in verification of diagnosability
  - Toy examples above processed in negligible time
- Verification of other properties on larger livingstone models:
  - check internal consistency, correctness w.r.t. real system
  - weakly coupled models ⇒ huge but shallow state spaces (e.g. 10^{50} states but only 3 steps deep)
  - SMV (and NuSMV) chokes
  - Yang's variable elimination saves the day!
- Lessons learned: symbolic model checking (with variable elimination) is a must, SAT-based has great potential (because of shallow depth)

Ames Research Center

Perspectives

- Larger experiments
  - See limits of scalability
  - Compare BDD vs. SAT
  - See how useful the results are in practice
  - Large-scale Livingstone models readily available
    Ex: X-34 spacecraft fuel feeding subsystem (800+ variables)
- Extend/refine/adjust model and specification patterns
  - according to feedback from real applications
- Integrated support in Livingstone toolset
  - first step: generate twin models in translator

Closely related to K-CTL (cf A. Cimatti's talk yesterday)