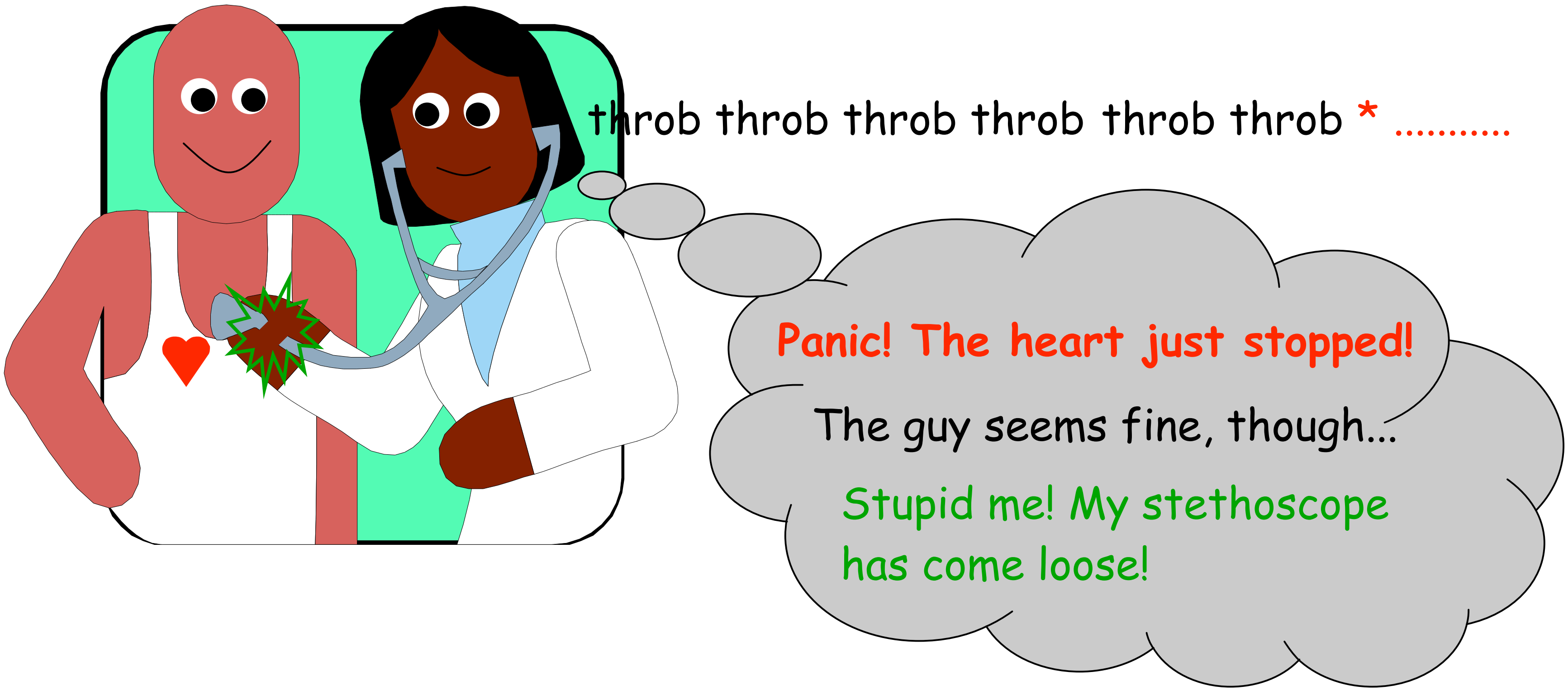
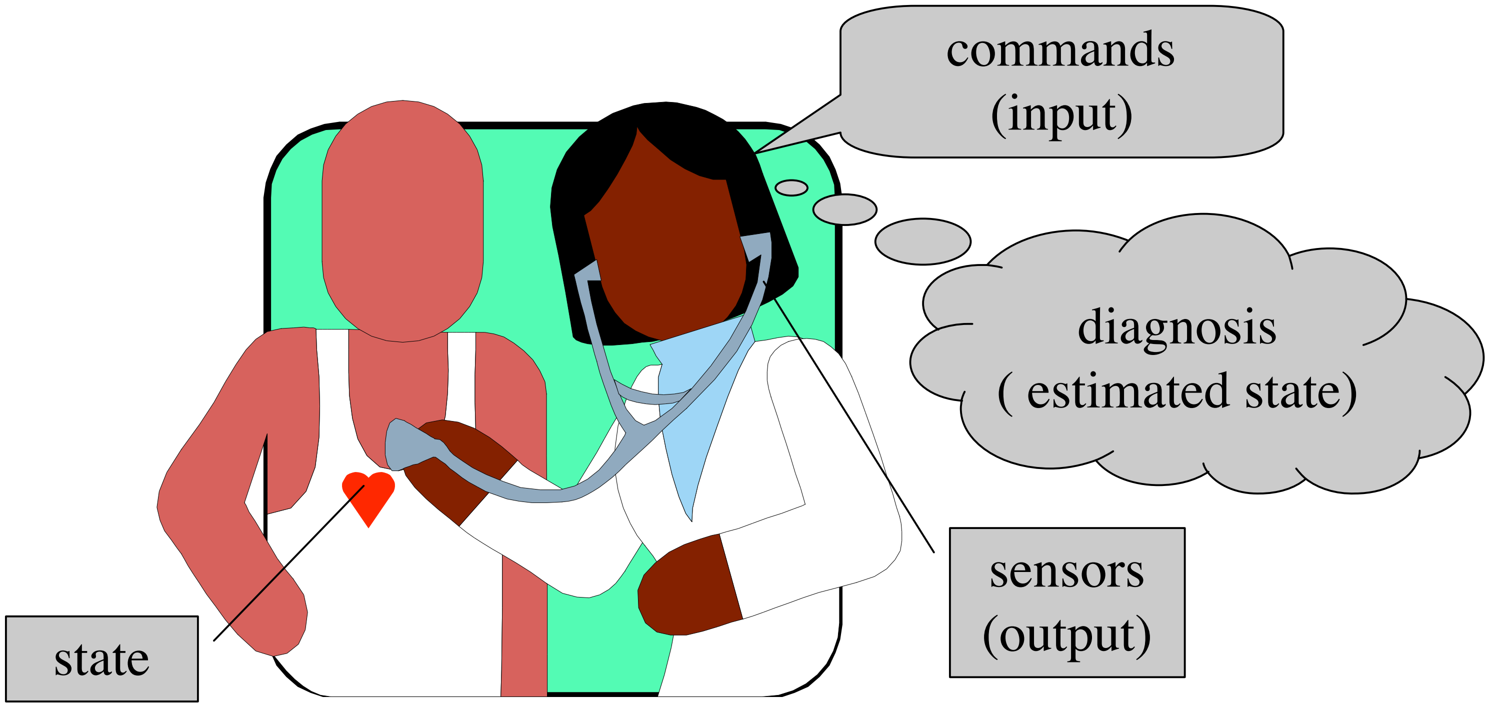
Verification of Diagnosability using Model Checking
(and why NASA cares)

Charles Pecheur, RIACS at NASA Ames

At the Doctor's



Diagnosability

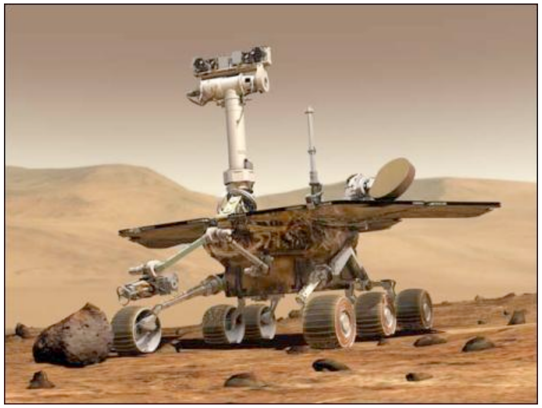


Can a (smart enough) doctor always make a proper diagnosis?

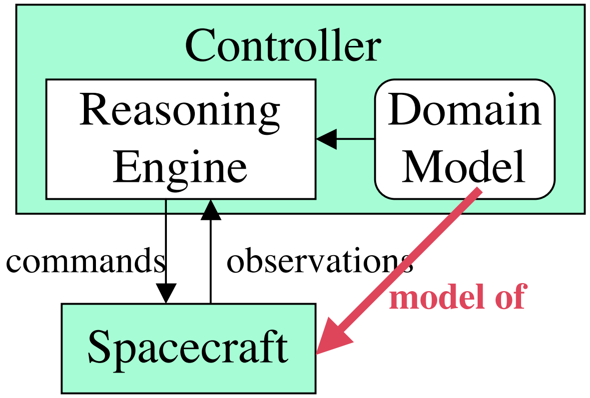
(... even if she cannot give commands?)

Autonomy at NASA

Autonomous spacecraft = on-board intelligence (AI)

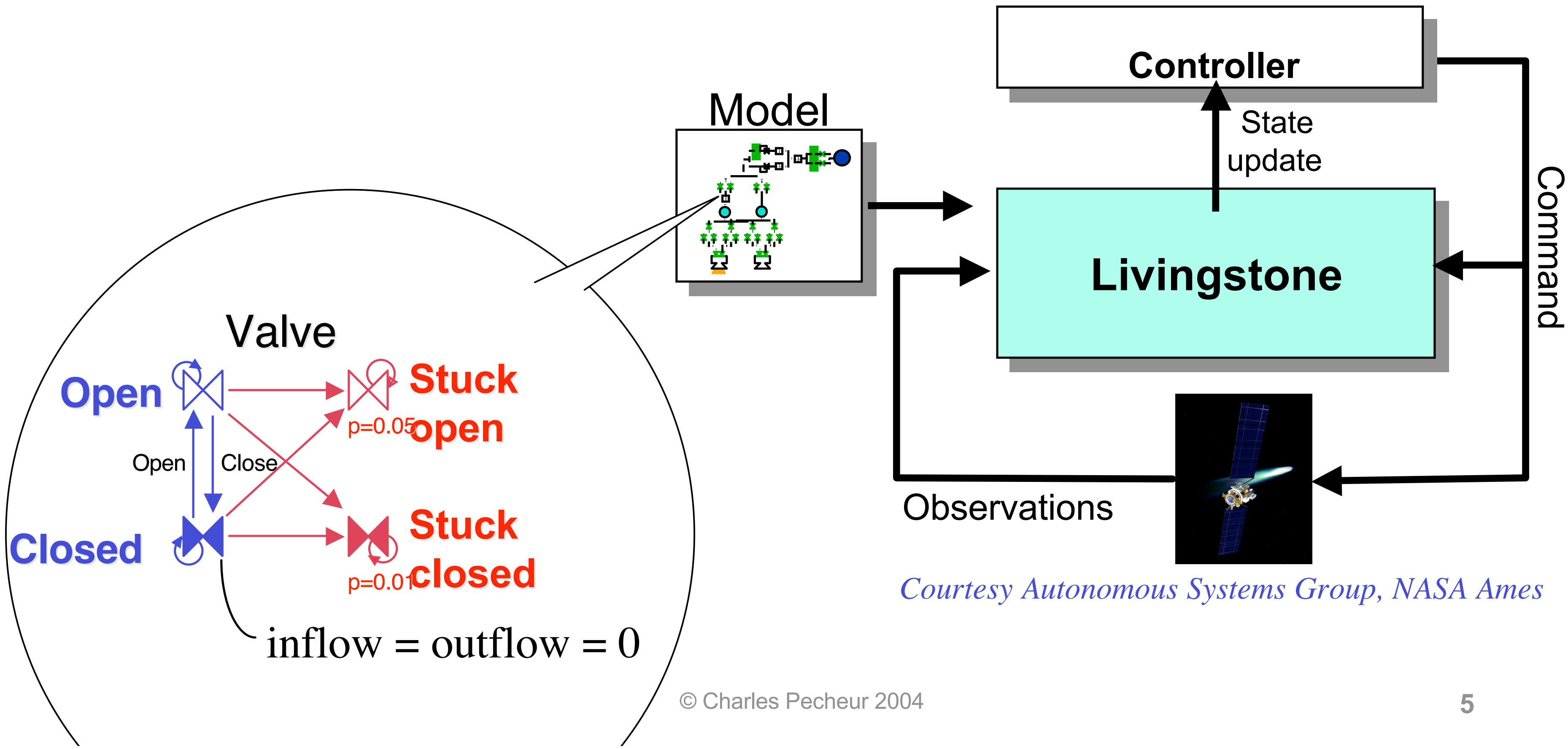


- **Goal:** Unattended operation in an unpredictable environment
- **Approach:** model-based reasoning
- **Pros**: smaller mission control crews, no communication delays/blackouts
- **Cons:** Verification and Validation ???
  Much more complex, huge state space
- Better verification is critical for adoption



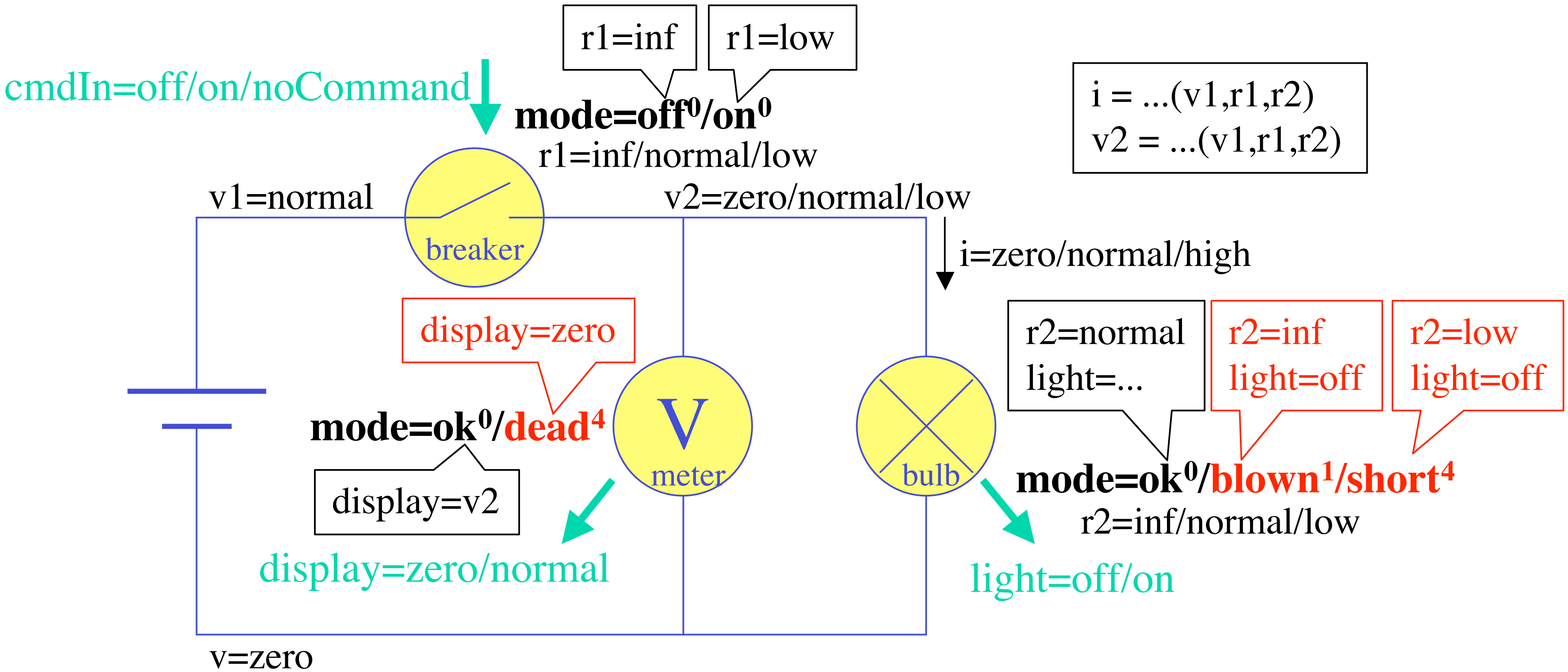
Model-Based Diagnosis

- Focus on **Livingstone** system from NASA Ames.
- Uses a discrete, qualitative model to reason about faults
  => naturally amenable to formal analysis



Courtesy Autonomous Systems Group, NASA Ames

A Simple Diagnosis Model

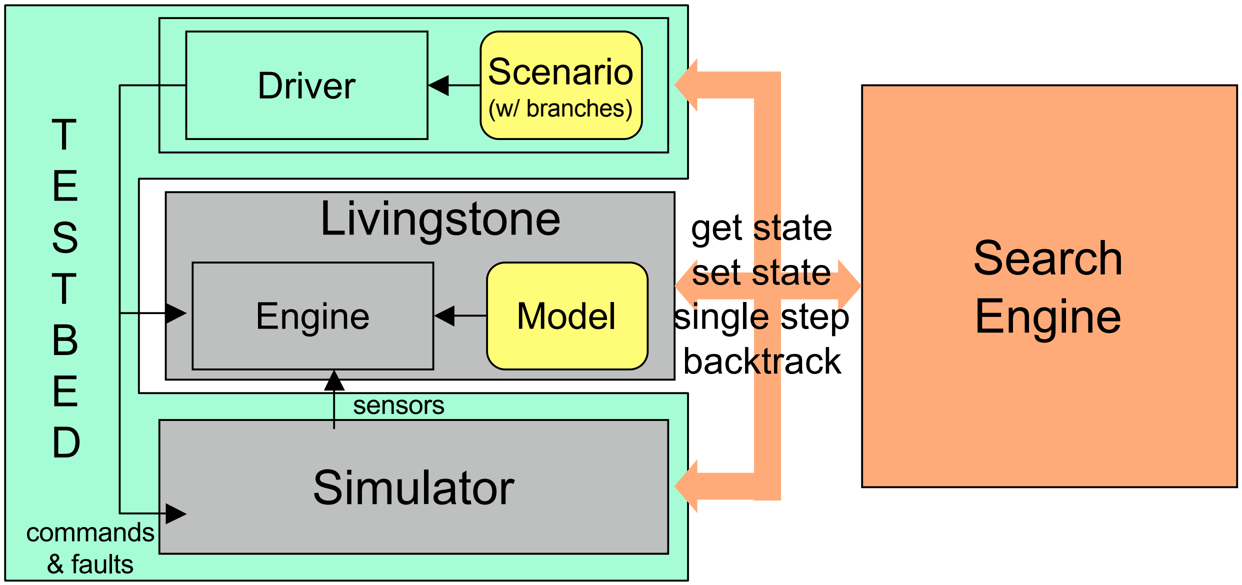


Goal: determine **modes** from observations
Generates and tracks *candidates*

breaker	bulb	meter	rank
off^0	**ok^0**	**ok^0**	**0**
off^0	**ok^0**	**blown^1**	**1**
on^0	**dead^4**	**short^4**	**8**

Livingstone PathFinder

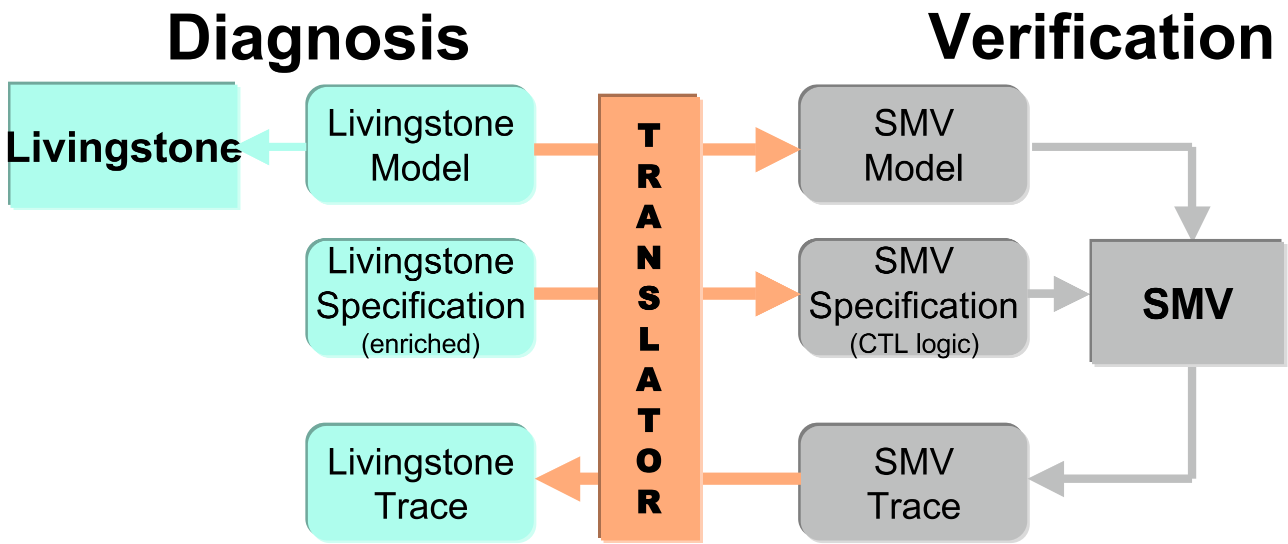
with Tony Lindsey (QSS @ ARC)



- An advanced testing/simulation framework for Livingstone applications
  - Executes the **Real Livingstone Program** in a simulated environment (testbed)
  - **Instrument** the code to be able to **backtrack** between alternate paths
- **Scenarios** = non-deterministic test cases (defined in custom language)
- **Modular** architecture with generic APIs (in Java)
  - allows different diagnosers, simulators (can use Livingstone), search algorithms (depth-first, breadth-first, heuristic, random, ...)
- Graphical interface, trace display, integration in Livingstone development tools
- See TACAS'04 paper

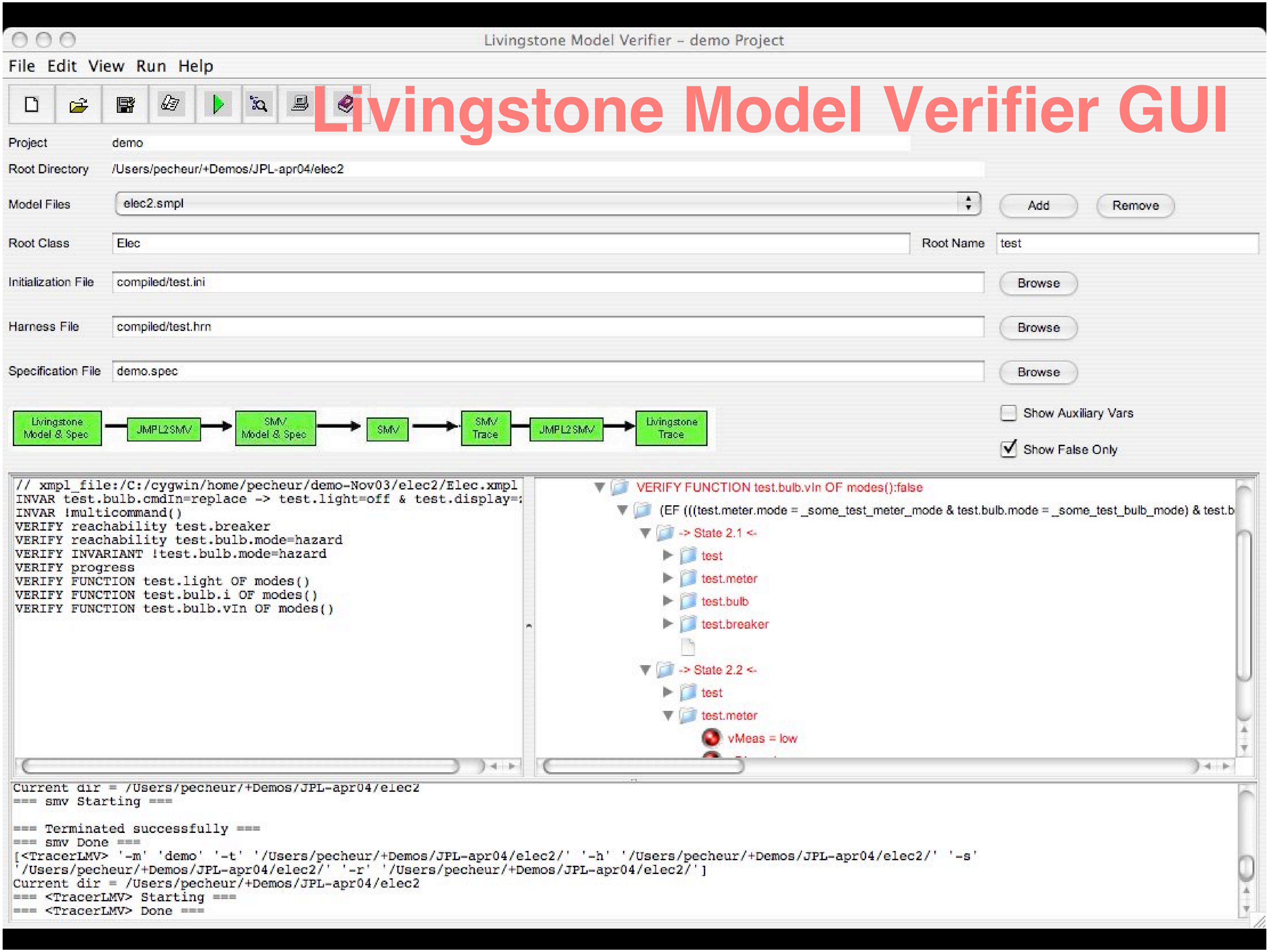
Livingstone-to-SMV Translator

Joint work with Reid Simmons (Carnegie Mellon)



- A translator that converts Livingstone models, specs, traces to/from SMV (in Java)
  - SMV: symbolic model checker (both BDD and SAT-based) allows exhaustive analysis of very large state spaces (10^{50+})
  - Translator hides away SMV, offers a model checker for Livingstone
- Enriched specification syntax (vs. SMV's core temporal logic)
- Graphical interface, trace display, integration in Livingstone development tools

Livingstone Model Verifier GUI

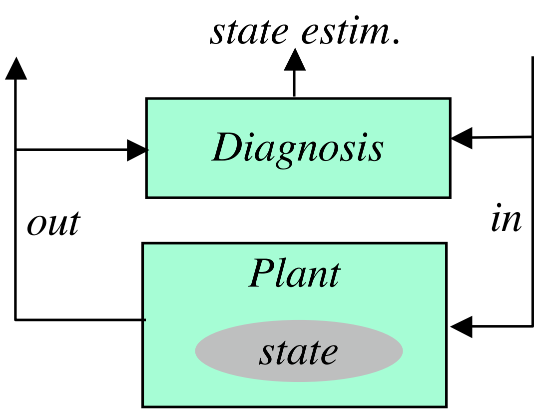


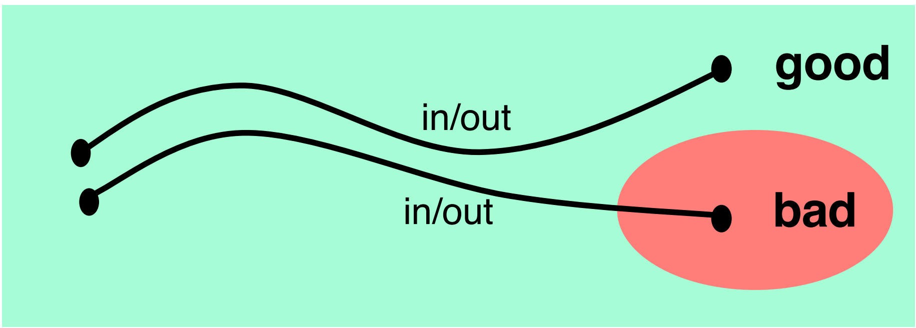
Verification of Diagnosis Models

- Coding Errors
  - e.g. Consistency, well-defined transitions, ...
  - Generic
  - Compare to Lint for C
- Model Correctness
  - Expected properties of modeled system
  - e.g. flow conservation, operational scenarios, ...
  - Application-specific
- **Diagnosability**
  - Are faults detectable/diagnosable?
    - Given available sensors
    - In all/specific operational situations (dynamic)
  - Innovative use of model checking using twin models

Diagnosability

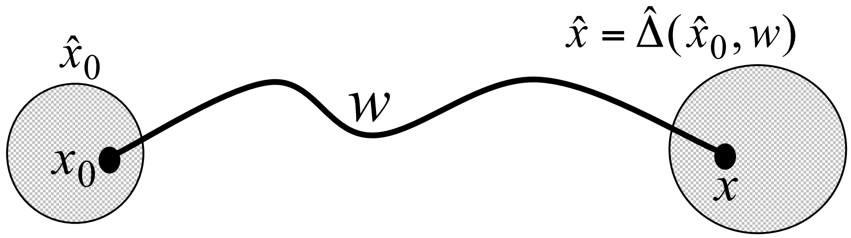
Joint work with Alessandro Cimatti (IRST)





- *Diagnosis* estimates the hidden state of *Plant*, given observable input and output of *Plant*.
- **Diagnosability**: Can (a smart enough) *Diagnosis* always tell when *Plant* comes to a **bad** state?
- **Intuition**: YES, if and only if there is no pair of executions, one reaching a **bad** state, the other reaching a **good** state, with identical observations.
- ... within a given context

Formalization of Diagnosis



Transition system $x \xrightarrow{u/y} x'$, executions $\sigma : x_0 \xrightarrow{w} x$

inputs u, outputs y are visible, states x, x' are hidden

trace $w = (u_1, y_1, \ldots, u_n, y_n)$

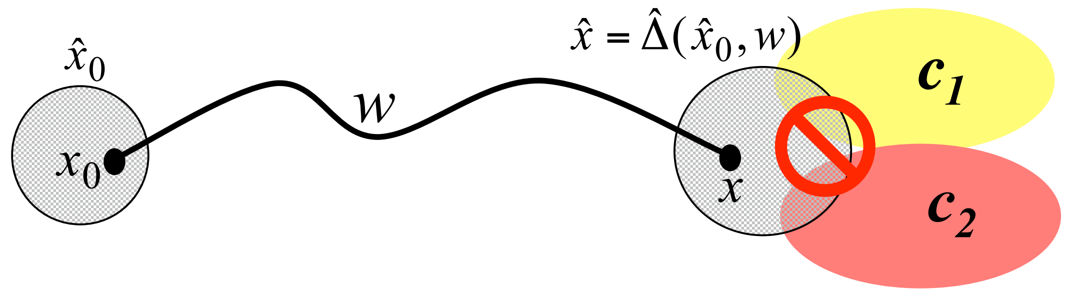
Diagnosis function $\hat{x} = \hat{\Delta}(\hat{x}_0, w)$ such that

$$x_0 \in \hat{x}_0, x_0 \xrightarrow{w} x \;\Rightarrow\; x \in \hat{\Delta}(\hat{x}_0, w)$$

updates belief state (set of possible states) according to observed trace

⇨

Formalization of Diagnosability



$$\text{Diagnosis condition } \hat{x} \models c_1 \perp c_2 \text{ iff } \hat{x} \cap c_1 = \varnothing \vee \hat{x} \cap c_2 = \varnothing$$

belief state never allows both c_1 and c_2

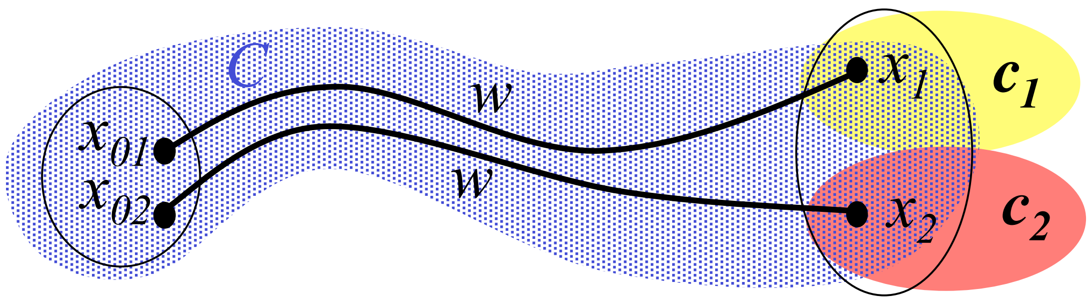
- Typical cases: *fault*$\perp\neg$*fault* (detection), $fault_1 \perp fault_2$ (identification)

$$\hat{\Delta} \models c_1 \perp c_2 \text{ iff } \forall x_0 \in \hat{x}_0, x_0 \xrightarrow{w} x \cdot \hat{\Delta}(\hat{x}_0, w) \models c_1 \perp c_2$$

$$c_1 \perp c_2 \text{ is diagnosable iff } \exists \hat{\Delta} \models c_1 \perp c_2$$

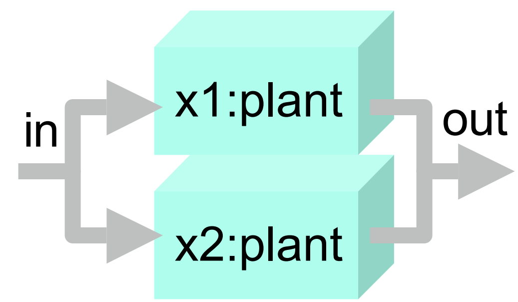
- ... in given context: conditions on execution σ and initial belief state $\hat{x}_0$

Diagnosability as Reachability



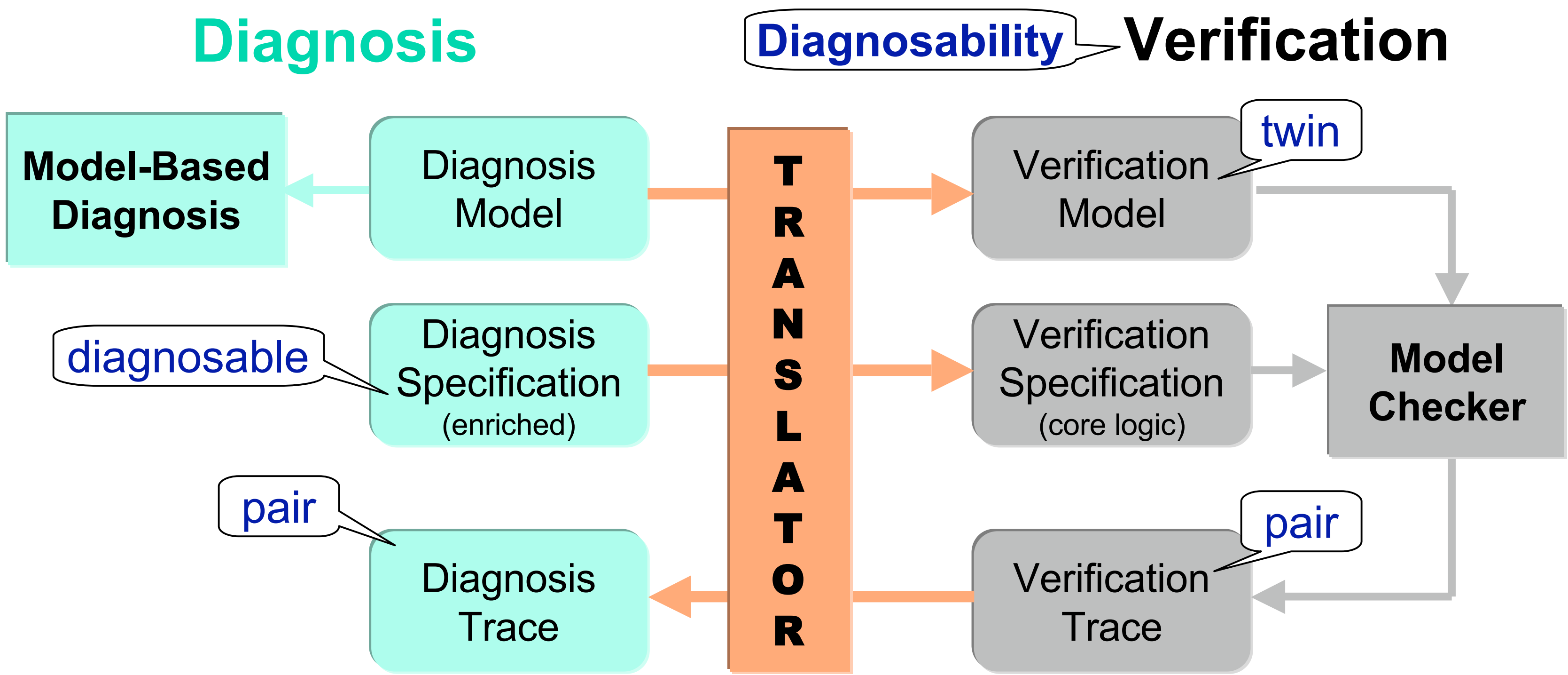
- Critical Pair σ_1, σ_2 for $c_1 \perp c_2$ (in context C) such that $w_1 = w_2 = w$ and $x_1 \in c_1$ and $x_2 \in c_2$ (and (σ_1, σ_2) satisfy C)
- C*oupled Twin Plant* P^2 = two copies of the plant P with merged inputs and outputs

	$c_1 \perp c_2$ diagnosable (in C)
iff	no critical pairs for $c_1 \perp c_2$ (in C)
iff	$c_1 \times c_2$ not reachable in P^2 (and C)



- Model checking: verify $\neg \mathbf{F}\ c_1(x_1) \wedge c_2(x_2)$ in P^2 (+ context)

Model Translation for Diagnosability



- Generate twin coupled model
- Support specific syntax for twin models and diagnosability properties
- Translate/correlate pairs of error traces

Diagnosability in SMV Translator

- Added generation of twin models
- Added syntax for properties of twin models
- Example: starting form known initial non-faulty state, with single faults, can we detect whether there is high current in the bulb?

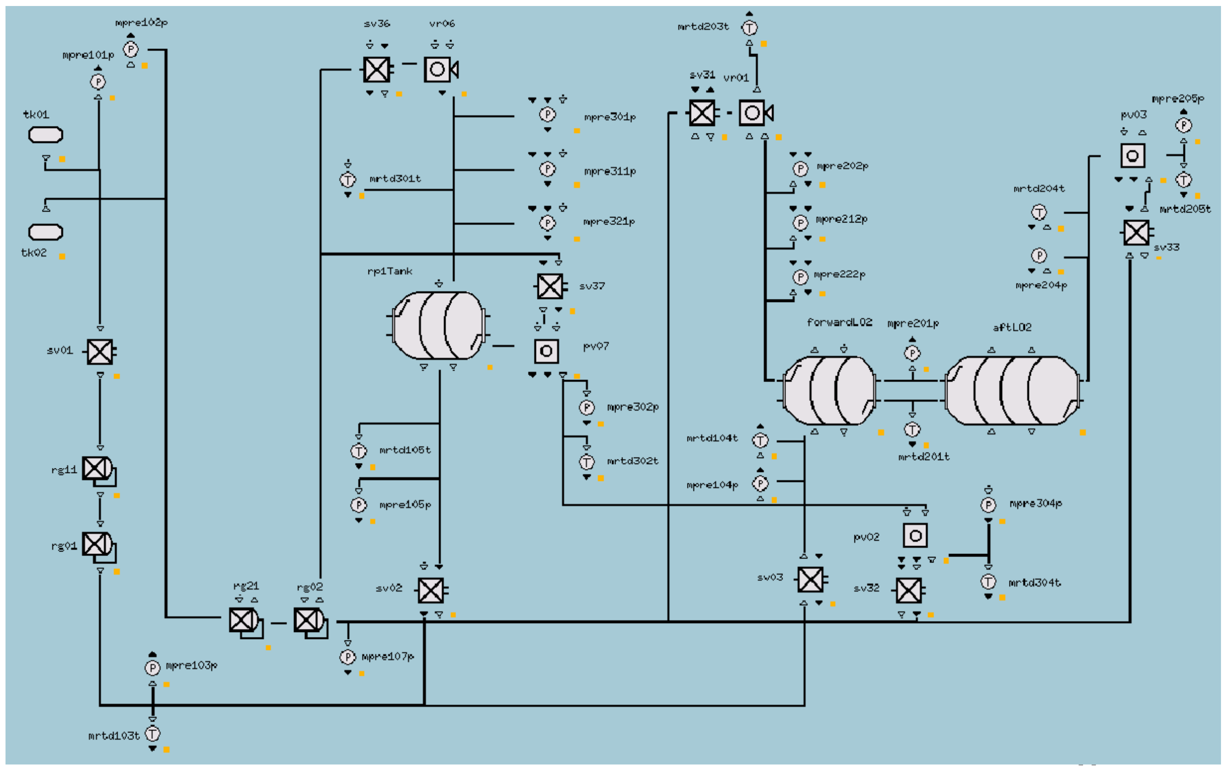
```
invar same(visibles())      // observations are the same on both sides

verify (same(modes()) & both(!broken()) ->
      !E[both(!multibroken()) U both(!multibroken()) & !same(test.bulb.i)]
```

- Coming soon: syntax for diagnosability property

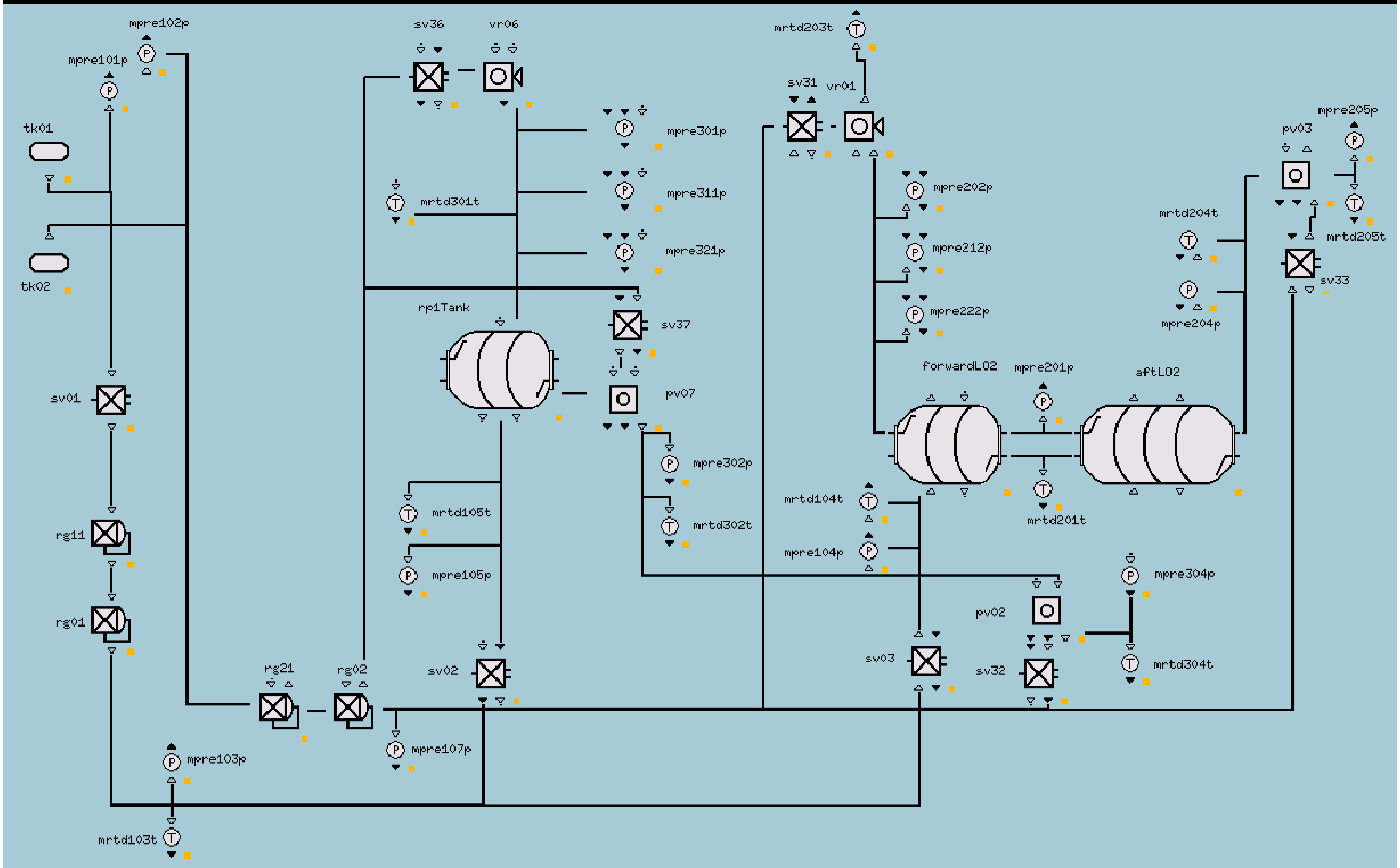
```
verify detection test.bulb.i=high
         from same(modes()) & both(!broken())
         keeping !multibroken()
```

X-34 / PITEX

- Propulsion IVHM Technology Experiment (ARC, GRC)
- Livingstone applied to propulsion feed system of space vehicle
- Livingstone model is $4 \cdot 10^{33}$ states



[redacted]


mpre102p
mpre101p
tk01
tk02
sv01
rg11
rg01
mpre103p
mrtd103t
rg21
rg02
sv36
vr06
mrtd301t
mpre301p
mpre311p
mpre321p
rp1Tank
sv37
pv07
mrtd105t
mpre105p
sv02
mpre107p
mpre302p
mrtd302t
mrtd203t
sv31
vr01
mpre202p
mpre212p
mpre222p
forwardLO2
mpre201p
aftLO2
mrtd201t
mrtd104t
mpre104p
sv03
pv02
sv32
mpre304p
mrtd304t
mpre205p
pv03
mrtd204t
mrtd205t
sv33
mpre204p

Diagnosability Verification on PITEX

with Roberto Cavada (IRST, NuSMV developer)

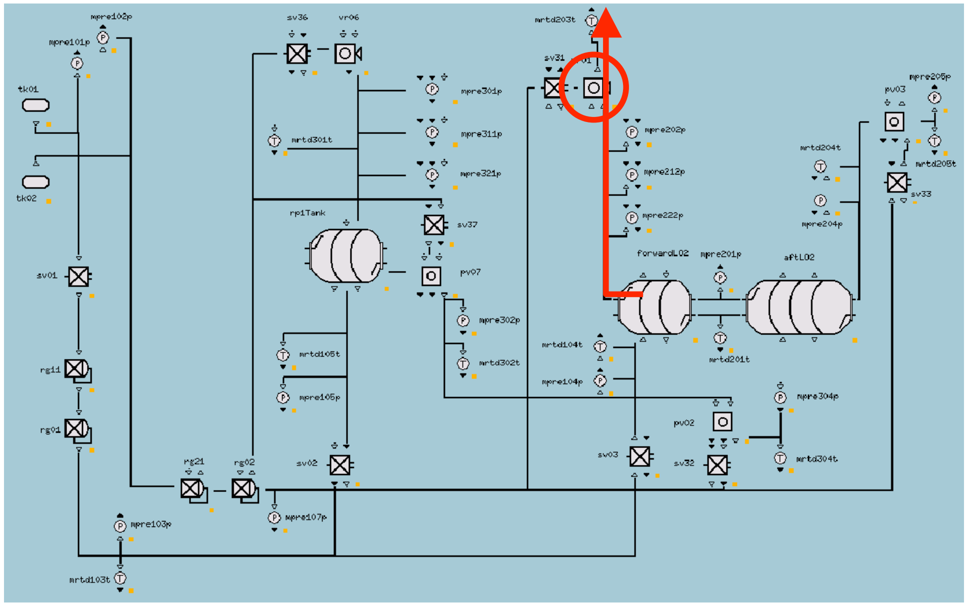
- Applied translator to PITEX model
- Goals:
  - Demonstrate scalability to real-size models
  - Demonstrate relevance wrt. application needs
- Compared BDD-based vs. SAT-based
  - BDD: single model done (with tuning), twin model too big
  - SAT: twin model done in a few seconds!
- Found application-relevant anomaly in PITEX model (unnoticed oxygen leak)
- See report: RIACS TR 03.03

PITEX Diagnosability Error

- *"Diagnosis can decide whether the venting valve VR01 is closed or stuck open (assuming no other failures)"*

  INVAR !test.multibroken() & twin(!test.broken())
  VERIFY INVARIANT !(test.vr01.mode=stuckOpen &
  twin(test.vr01.valvePosition=closed))

- Results show a pair of traces with same observations, one leading to **VR01 stuck open**, the other to **VR01 closed**. Application specialists fixed their model.



Publications

- Charles Pecheur, Alessandro Cimatti. Formal Verification of Diagnosability via Symbolic Model Checking .Workshop on Model Checking and Artificial Intelligence (MoChArt-2002), Lyon, France, July 22/23, 2002.

- Cavada, Roberto and Pecheur, Charles. Practical Formal Verification of Diagnosability of Large Models via Symbolic Model Checking. Technical Report TR03.03, RIACS, USRA, January 2003.

- Roberto Cavada, Alessandro Cimatti, Charles Pecheur. Formal Verification of Diagnosability via Symbolic Model Checking. IJCAI'03, Acapulco, Mexico, August 2003.

Perspectives

The Big Picture:

A Model-Based Failure Analysis Tool
Applicable to Dynamic Models

- Key concept: partial observability
- Demonstrated on concrete, real-size applications
  - Demonstrate scalability and relevance to practical needs
- Tools aimed at non-specialist users, integrated with development
  - Vision: build integrated "advanced debuggers"
  - GUI, visualization, documentation, integration, ...
  - Takes a lot of engineering work

Perspectives (cont'd)

Extensions:

- Extend from discrete to **real-time and hybrid models**
  - Build on new generalized solvers (MathSAT at IRST, ICS at SRI)
- Apply to **human-computer interaction**
  - Features partial observability issues
- Study relations with classical **risk analysis models**
  - Fault trees, FMEA, ...
- Generalize to verification of **epistemic logics**
  - applications to multi-agent systems, security protocols

CTLK Logic

- Reasoning about time and knowledge:
  = CTL + temporal operators
  - $K_a\ \varphi$ = *a* knows φ
  - $E_G\ \varphi$ = each one in *G* knows φ
  - $D_G\ \varphi$ = together, all in *G* know φ
  - $C_G\ \varphi$ = it is common knowledge in *G* that φ
- Interpreted over an *Interpreted System* =
  - *Transition system* (Kripke structure) *T* +
  - *Observation functions* $obs_a(\sigma)$ over runs σ of *T*, for each agent *a*

$$\sigma \sim_a \sigma' \quad \text{iff} \quad obs_a(\sigma) = obs_a(\sigma')$$

$$\sigma \models K_a\ \varphi \quad \text{iff} \quad \text{for all } \underline{\text{reachable}}\ \sigma' .\ \sigma \sim_a \sigma' \Rightarrow \sigma' \models \varphi$$

Knowledge views

- *Total recall*:
  - $\text{obs}_a(\sigma)$ = all that *a* saw since start of σ
  - Full CTLK non-elementary
  - Nice solution for $AX^k \varphi$, φ with only one actor
- *Observational*:
  - $\text{obs}_a(\sigma)$ = all that *a* sees in last state of σ
  - $\text{obs}_a(s_0 \ldots s_n) = \text{obs}_a(s_n)$, $s \sim_a s'$ becomes a state relation
  - $s \models K_a \varphi$ iff $(s \sim_a s' \leftarrow^* s'_0 \in Q_0) \Rightarrow s' \models \varphi$
  - Can be expressed as generalized CTL over multiple transition relations $\rightarrow$, $\sim_a$, $\leftarrow$
  - SMV-style symbolic model checking applies
- Variants: last + clock, all – clock

Diagnosability and CTLK

$$c_1 \perp c_2 \quad \text{iff} \quad AG\,(K_d{\sim}c_1 \vee K_d{\sim}c_2)$$

where agent *d* (diagnoser) sees all observable variables, with perfect recall.

- Bounded approximation corresponds to $AX^k\,\varphi$ case above
- Conversely, all φ with positive K over a single agent are equivalent to

$$K\,\varphi_1 \vee \ldots \vee K\,\varphi_n$$

and can be analyzed using the twin model approach

Diagnosability with Observational

- Franco Raimondi (King's College London)
  - At Ames for the summer
  - developed a BDD model checker for CTLK (using observational view)
  - studying connections between diagnosability and CTLK
- Using observational view for diagnosability
  - requires mapping memory of previous obs explicitly into diagnoser variables
  - inelegant, cumbersome and inefficient
  - flexible model for diagnoser's memory
  - work in progress!

CTLK + correctness

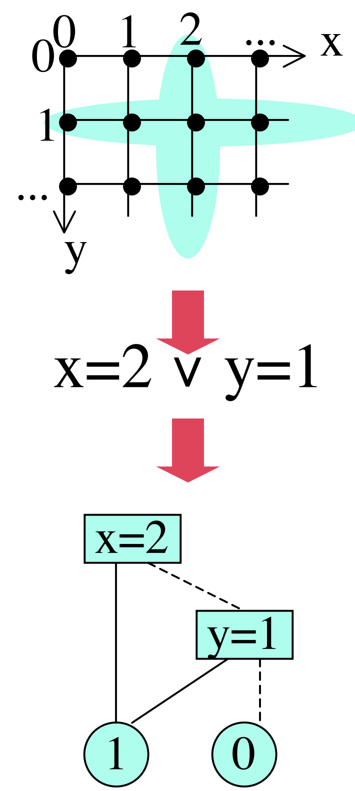
$K^{\wedge}{}_{a}^{G}\,\varphi$ = a knows φ, assuming everyone in G "works correctly"

- "works correctly" is a state condition
- Useful for diagnosis: one agent per component, works correctly iff non-fault mode
- Verification supported by Raimondi's tool (BDD based)
- Expressivity issue: correctness in present state vs. in future
- Work in progress!

Backup Slides

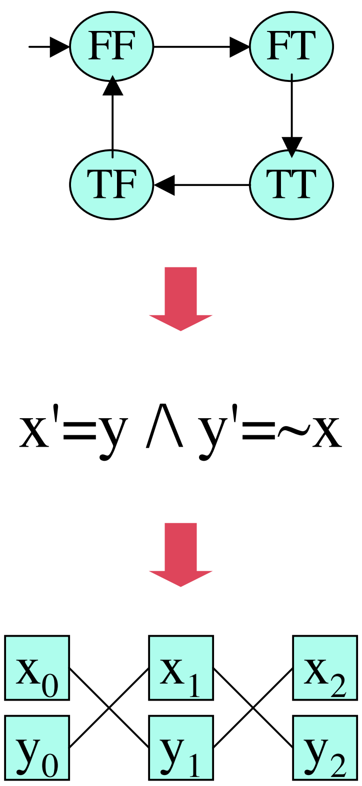
Symbolic Model Checking (BDD)

- Manipulates sets of states,
  Represented as boolean formulas,
  Encoded as binary decision diagrams.
- Can handle large state spaces (10^{50} and up).
- BDD computations:
  - Efficient algorithms for needed operations.
  - BDD size is still exponential in worst case.
  - Highly sensitive (e.g. to variable ordering) and hard to optimize.
- Example: SMV/NuSMV (Carnegie Mellon/IRST)

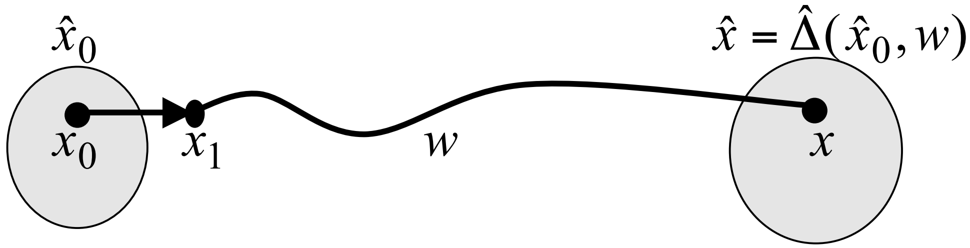


Bounded Model Checking (SAT)

- Symbolic model checking variant.
- Uses SAT (propositional satisfiability) rather than BDDs.
  - Idea: unroll transition relation a finite number of times into a (big) constraint network.
- Bounded-depth only, not complete.
- Very efficient
  - Polynomial space!
  - Exponential time in the worst-case **but** modern SAT solvers are very efficient in most practical cases.
- Example: NuSMV (using the Chaff solver from Princeton)



Formalization



Transition system $x \xrightarrow{u/y} x'$, execution $\sigma : x_0 \xrightarrow{w} x$

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

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

updates belief state according to observed trace

Correct iff $x_0 \in \hat{x}_0, x_0 \xrightarrow{w} x \;\Rightarrow\; x \in \hat{x}$

does not lose the actual state

Perfect diagnosis $\Delta_P(\hat{x}_0, w) = \{x \mid \exists x_0 \in \hat{x}_0 . x_0 \xrightarrow{w} x\}$

the best possible knowing the transition system

Formalization (cont'd)

$$\hat{x} \models c_1 \perp c_2 \quad \text{iff} \quad \hat{x} \cap c_1 = \emptyset \vee \hat{x} \cap c_2 = \emptyset$$

no ambiguity between c_1 and c_2

$$\hat{x}_0 \models \theta_C \quad \text{iff} \quad \hat{x}_0 \times \hat{x}_0 \subseteq \theta_C$$

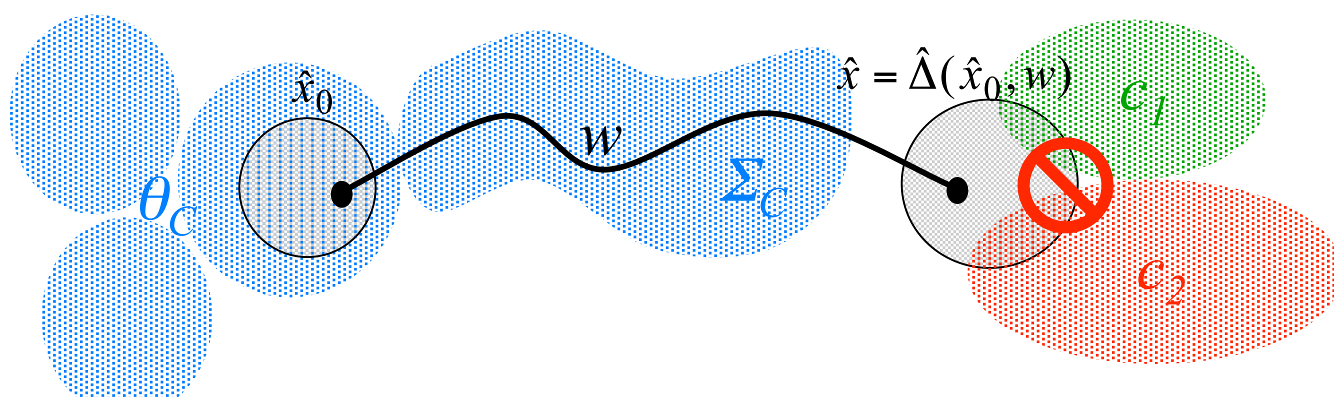
initial belief compatible with equivalence θ_C

$$(\hat{x}_0, w) \models (\Sigma_C, \theta_C) \quad \text{iff} \quad \hat{x}_0 \models \theta_C \wedge \exists \sigma : x_0 \xrightarrow{w} x \,.\, x_0 \in \hat{x}_0 \wedge \sigma \in \Sigma$$

idem. and trace compatible with some execution in Σ_C

$$\hat{\Delta}, (\Sigma_C, \theta_C) \models c_1 \perp c_2 \quad \text{iff} \quad (\hat{x}_0, w) \models (\Sigma_C, \theta_C) \Rightarrow \hat{\Delta}(\hat{x}_0, w) \models c_1 \perp c_2$$

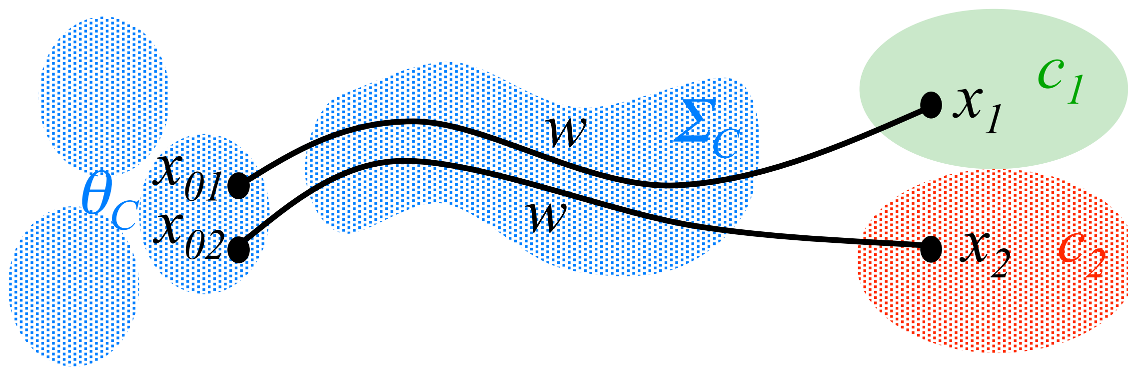
for all initial beliefs and executions within context, no ambiguity



Critical Pairs

Counter-example of a condition $c_1 \perp c_2$ in context (Σ_C, θ_C):
a pair of executions $\sigma_1|\sigma_2 : x_{01}|x_{02} -w\!\!-\!\!> x_1|x_2$ with the same observable trace w, such that

- $c_1(x_1)$ and $c_2(x_2)$, and
- ... $\sigma_1, \sigma_2 \in \Sigma_C$, and
- ... $x_{01}\ \theta_C\, x_{02}$

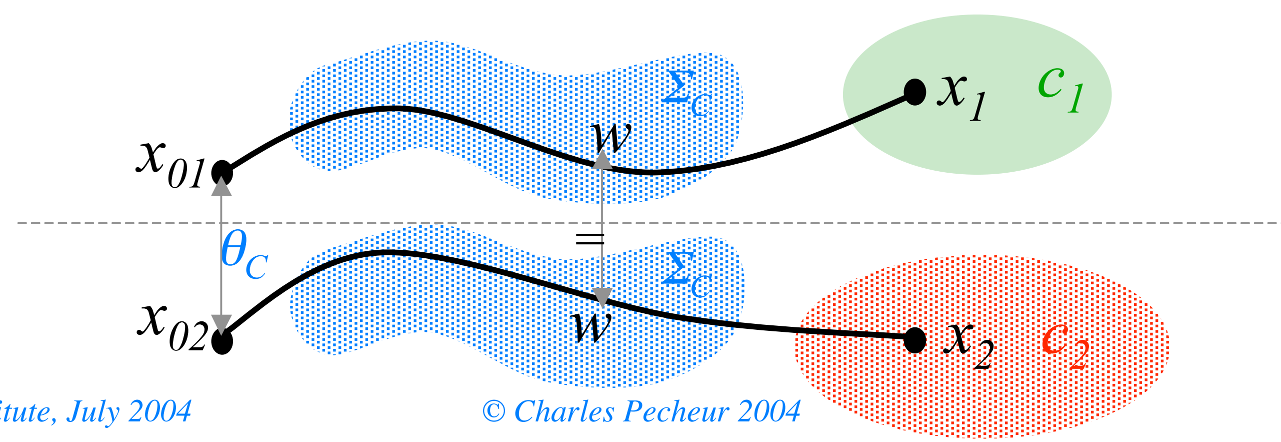


$c_1 \perp c_2$ diagnosable in (Σ_C, θ_C) iff no critical pairs

Coupled Twin Model

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

$$c_1 \perp c_2 \text{ diagnosable in } (\Sigma_C, \theta_C)$$
iff
$$c_1 \times c_2 \text{ not reachable from } \theta_C \text{ through } \Sigma_C \times \Sigma_C \text{ in } P^2$$



Temporal Epistemic Logic

- Reasoning about time and knowledge: **CTLK** logic

$\varphi \quad ::= \; p \mid \neg\varphi \mid \varphi\wedge\varphi$ — *atomic propositions, boolean ops*

$\quad\quad \mid \; \text{EX}\,\varphi \mid \text{E}[\varphi \text{ U } \varphi] \mid \text{EG}\,\varphi$ — *temporal ops*

$\quad\quad \mid \; \text{K}_a\,\varphi \mid \text{E}_G\,\varphi \mid \text{D}_G\,\varphi \mid \text{C}_G\,\varphi$ — *knowledge ops*

with $\varphi\vee\varphi' := \neg(\neg\varphi\wedge\neg\varphi')$, $\text{EF}\,\varphi := \text{E}[\text{true U }\varphi]$, $\text{AG}\,\varphi := \neg\text{EF}\,\neg\varphi$, ...

- Interpreted over an *Interpreted System* =
  - *Transition system* (Kripke structure) T +
  - *Observation functions* $\text{obs}_a(\sigma)$ over runs σ of T, for each agent a

$$\sigma \sim_a \sigma' \quad \text{iff} \quad \text{obs}_a(\sigma)=\text{obs}_a(\sigma')$$

$$\sigma \models \text{K}_a\,\varphi \quad \text{iff} \quad \text{for all } \underline{\text{reachable}}\ \sigma'\ .\ \sigma\sim_a\sigma' \Rightarrow \sigma' \models \varphi$$