Abstractions for Reverse Engineering

Validating the Computerization of Relay-based Interlocking

PhD Candidate: **Anna Becchi**

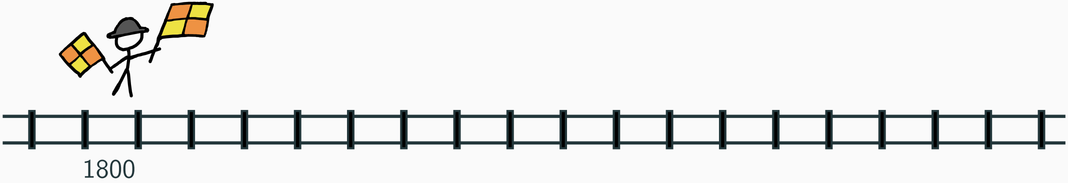
Advisor: Prof. Alessandro Cimatti

September 26th 2025

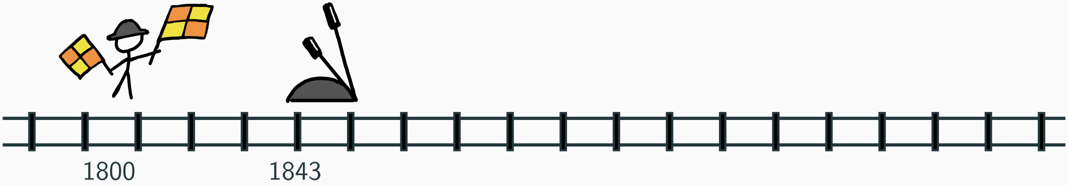
Motivation: modernizing railway interlocking systems

Motivation: modernizing railway interlocking systems

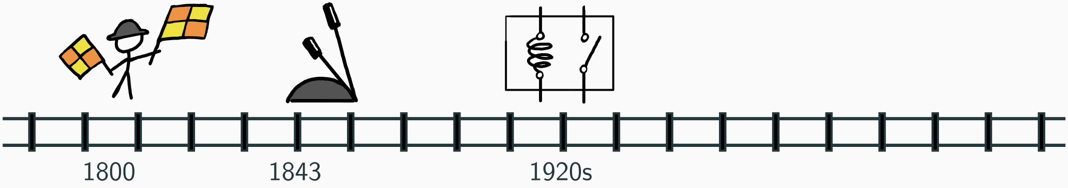
Motivation: modernizing railway interlocking systems



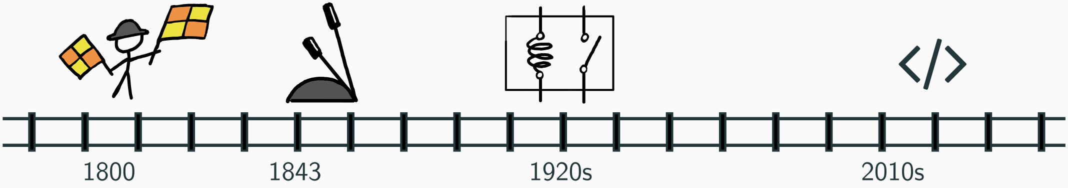
Motivation: modernizing railway interlocking systems



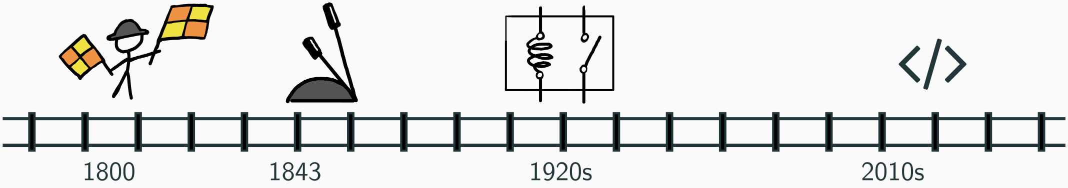
Motivation: modernizing railway interlocking systems



Motivation: modernizing railway interlocking systems



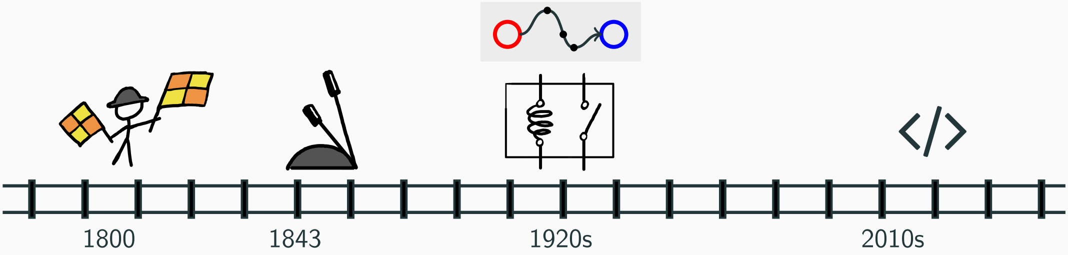
Motivation: modernizing railway interlocking systems



Embracing new technologies means:

- Improve efficiency, sustainability, safety
- Sometimes rethink the whole system

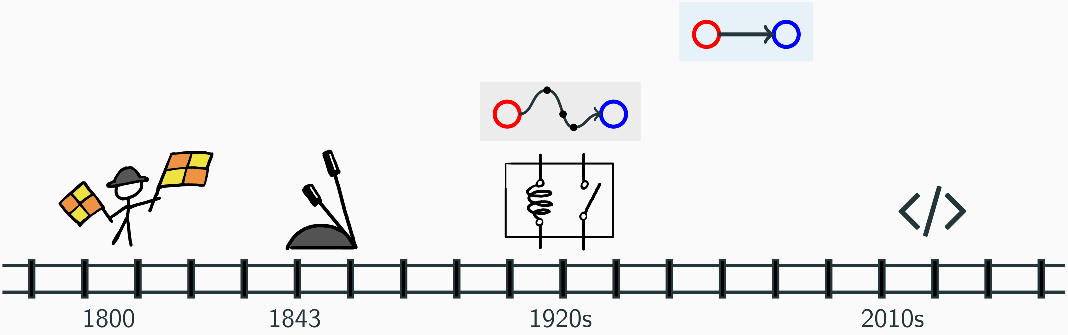
Motivation: modernizing railway interlocking systems



Embracing new technologies means:

- Improve efficiency, sustainability, safety
- Sometimes rethink the whole system

Motivation: modernizing railway interlocking systems

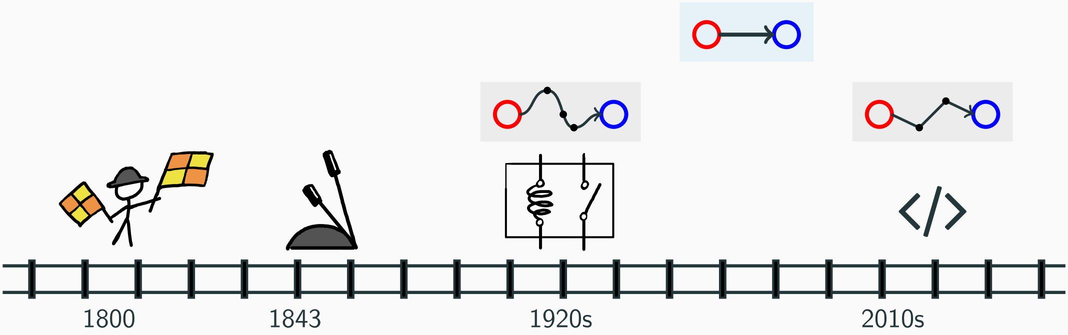


Embracing new technologies means:

Improve efficiency, sustainability, safety

Sometimes rethink the whole system

Motivation: modernizing railway interlocking systems

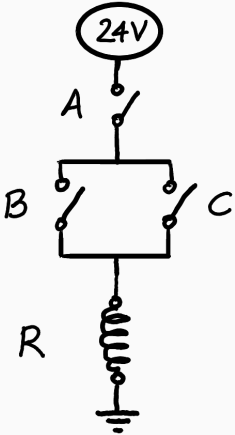


Embracing new technologies means:

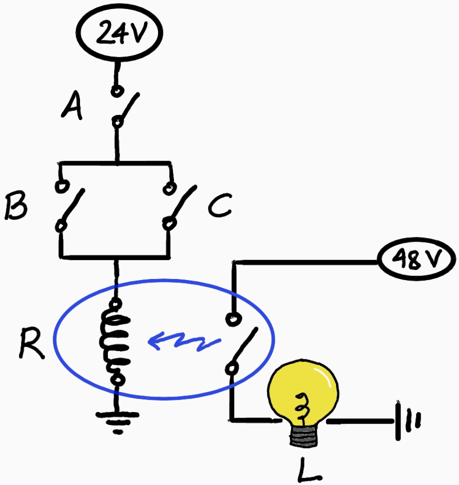
👍 Improve efficiency, sustainability, safety

⚠ Sometimes rethink the whole system

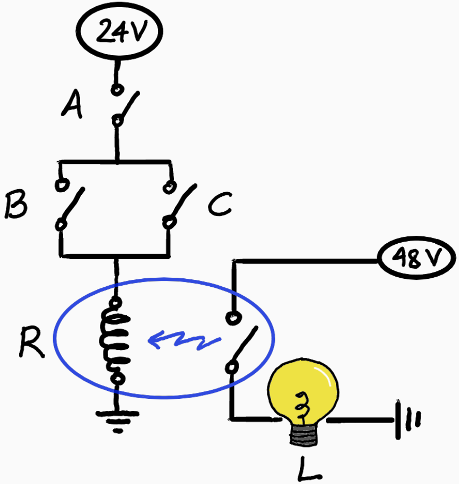
Example: understanding a relay-based circuit



Example: understanding a relay-based circuit

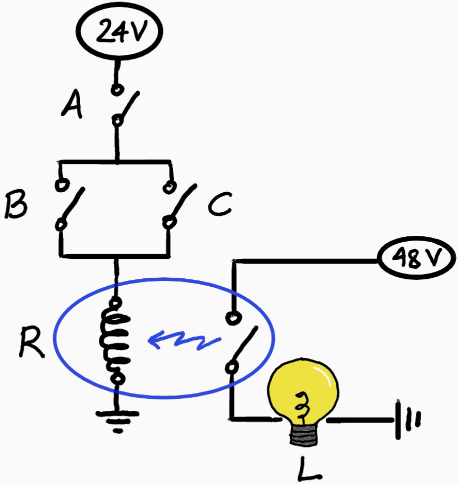


Example: understanding a relay-based circuit




```
def lamp_control(A, B, C) {
  R := A & (B | C);

  L := R;
}
```

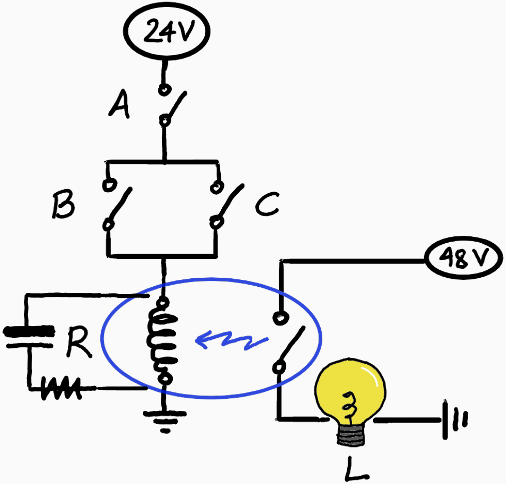
Example: understanding a relay-based circuit




```
def lamp_control(A, B, C) {
  R := A & (B | C);

  L := R;
}
```

`(A & (B | C)) <-> L`

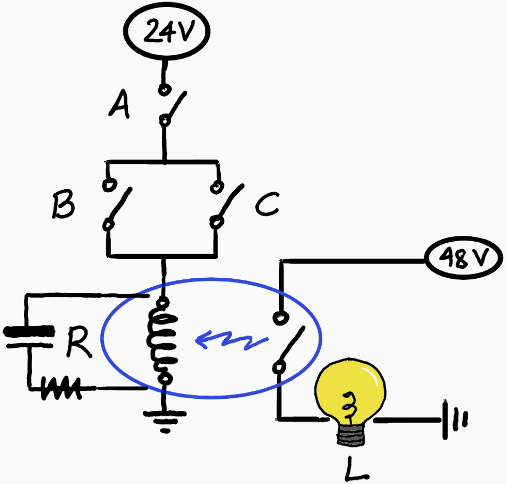
Example: understanding a relay-based circuit




```
def lamp_control(A, B, C) {
  R := A & (B | C);
  charge_capacitor()
  L := R;
}
```

`(A & (B | C)) <-> L`

Example: understanding a relay-based circuit




```
def lamp_control(A, B, C) {
  R := A & (B | C);
  charge_capacitor()
  L := R;
}
```

(A & (B | C)) <-> L

(A & (B | C)) -> F_{10s} L

This thesis

This thesis

Abstraction Modulo Stability (AMS) [CAV'22, FMSD'24]

This thesis

Abstraction Modulo Stability (AMS) [CAV'22, FMSD'24]

Minimal P-stable Abstraction [SoSyM'24]

This thesis

Abstraction Modulo Stability (AMS) [CAV'22, FMSD'24]

Minimal P-stable Abstraction [SoSyM'24]

SMT-based encoding and optimizations of relay circuits [TACAS'22]

This thesis

Abstraction Modulo Stability (AMS) [CAV'22, FMSD'24]

Minimal P-stable Abstraction [SoSyM'24]

SMT-based encoding and optimizations of relay circuits [TACAS'22]

New QE algorithm for LRA [TBD]

This thesis

Abstraction Modulo Stability (AMS) [CAV'22, FMSD'24]

Minimal P-stable Abstraction [SoSyM'24]

SMT-based encoding and optimizations of relay circuits [TACAS'22]

New QE algorithm for LRA [TBD]

Extract properties from relay circuits and validate the new software [CAV'24]

This thesis

- Abstraction Modulo Stability (AMS) [CAV'22, FMSD'24]
- Minimal P-stable Abstraction [SoSyM'24]
- SMT-based encoding and optimizations of relay circuits [TACAS'22]
- New QE algorithm for LRA [TBD]

- Extract properties from relay circuits and validate the new software [CAV'24]
- VALU3S PID controller stability verification [DSN-W'23]

This thesis

Abstraction Modulo Stability (AMS) [CAV'22, FMSD'24]

Minimal P-stable Abstraction [SoSyM'24]

SMT-based encoding and optimizations of relay circuits [TACAS'22]

New QE algorithm for LRA [TBD]

Extract properties from relay circuits and validate the new software [CAV'24]

PID controller stability verification [DSN-W'23]

NORMA: a tool for the analysis of relay circuits

This thesis

Abstraction Modulo Stability (AMS) [CAV'22, FMSD'24]

Minimal P-stable Abstraction [SoSyM'24]

SMT-based encoding and optimizations of relay circuits [TACAS'22]

New QE algorithm for LRA [TBD]

Extract properties from relay circuits and validate the new software [CAV'24]

PID controller stability verification [DSN-W'23]

NORMA: a tool for the analysis of relay circuits

Tool for reachability analysis in pwc hybrid systems

Plan of the talk

[TBD]

[CAV'22, FMSD'24]

[TACAS'22]

[CAV'24]

Plan of the talk

Prelude: SMT and Model Checking

[TBD]

[CAV'22, FMSD'24]

[TACAS'22]

[CAV'24]

Plan of the talk

Prelude: SMT and Model Checking

[TBD]

Act I: Abstraction Modulo Stability

[CAV'22, FMSD'24]

Extract high-level properties from timed transition systems

[TACAS'22]

[CAV'24]

Plan of the talk

Prelude: SMT and Model Checking [TBD]

Act I: Abstraction Modulo Stability [CAV'22, FMSD'24]

Extract high-level properties from timed transition systems

Act II: Model Checking Relay-based Interlocking [TACAS'22]

✔ SMT-based encoding and optimizations of relay circuits

[CAV'24]

Plan of the talk

Prelude: SMT and Model Checking [TBD]

Act I: Abstraction Modulo Stability [CAV'22, FMSD'24]

Extract high-level properties from timed transition systems

Act II: Model Checking Relay-based Interlocking [TACAS'22]

✔ SMT-based encoding and optimizations of relay circuits

Act III: Supporting RFI's migration towards software interlocking [CAV'24]

Use AMS to compare the relay and the new software interlocking

Background: super-dense time model

Infinite-state Timed Transition System $\mathcal{M} \doteq \langle X, C, I, \mathrm{Init}(X), \mathrm{Invar}(X), \mathrm{Trans}(X, I, X') \rangle$

Background: super-dense time model

Boolean&theory state vars

Infinite-state Timed Transition System $\mathcal{M} \doteq \langle X, C, I, \mathrm{Init}(X), \mathrm{Invar}(X), \mathrm{Trans}(X, I, X') \rangle$

Background: super-dense time model

clock vars

Boolean&theory state vars

Infinite-state Timed Transition System $\mathcal{M} \doteq \langle X, C, I, \mathrm{Init}(X), \mathrm{Invar}(X), \mathrm{Trans}(X, I, X') \rangle$

Background: super-dense time model

clock vars

Boolean&theory state vars

Boolean input vars

Infinite-state Timed Transition System $\mathcal{M} \doteq \langle X, C, I, \mathrm{Init}(X), \mathrm{Invar}(X), \mathrm{Trans}(X, I, X') \rangle$

Background: super-dense time model

Boolean&theory state vars

clock vars

Boolean input vars

Infinite-state Timed Transition System $\mathcal{M} \doteq \langle X, C, I, \mathrm{Init}(X), \mathrm{Invar}(X), \mathrm{Trans}(X, I, X') \rangle$

initial states

Background: super-dense time model

clock vars

Boolean&theory state vars

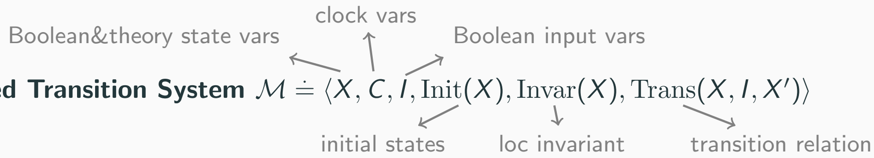
Boolean input vars

Infinite-state Timed Transition System $\mathcal{M} \doteq \langle X, C, I, \mathrm{Init}(X), \mathrm{Invar}(X), \mathrm{Trans}(X, I, X') \rangle$

initial states

loc invariant

Background: super-dense time model



Infinite-state Timed Transition System $\mathcal{M} \doteq \langle X, C, I, \mathrm{Init}(X), \mathrm{Invar}(X), \mathrm{Trans}(X, I, X') \rangle$

Background: super-dense time model

clock vars

Boolean&theory state vars

Boolean input vars

Infinite-state Timed Transition System $\mathcal{M} \doteq \langle X, C, I, \mathrm{Init}(X), \mathrm{Invar}(X), \mathrm{Trans}(X, I, X') \rangle$

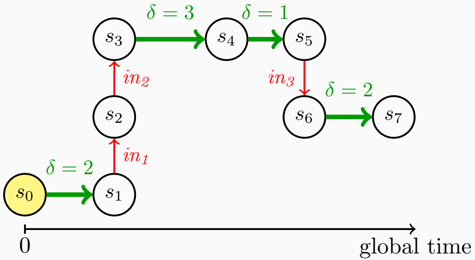
initial states

loc invariant

transition relation

Path of $\mathcal{M}$: $\pi = (s_0, s_1, \ldots)$

- discrete step:$(s_i, in, s'_{i+1}) \models \mathrm{Trans}$
- timed step:
  $\forall c \in C. \quad (s_i, s'_{i+1}) \models (c' = c + \delta)$
  $\forall x \in X \setminus C. \quad (s_i, s'_{i+1}) \models (x' = x)$



Background: super-dense time model

Boolean&theory state vars — clock vars — Boolean input vars

Infinite-state Timed Transition System $\mathcal{M} \doteq \langle X, C, I, \mathrm{Init}(X), \mathrm{Invar}(X), \mathrm{Trans}(X, I, X') \rangle$

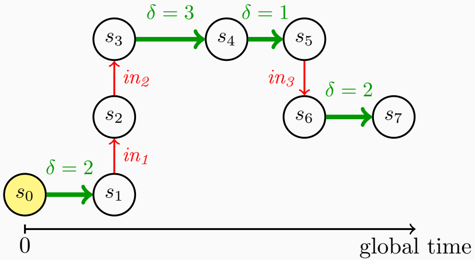
initial states — loc invariant — transition relation

Path of $\mathcal{M}$: $\pi = (s_0, s_1, \ldots)$

- discrete step:$(s_i, in, s'_{i+1}) \models \mathrm{Trans}$
- timed step:

  $\forall c \in C. \quad (s_i, s'_{i+1}) \models (c' = c + \delta)$

  $\forall x \in X \setminus C. \quad (s_i, s'_{i+1}) \models (x' = x)$



Linear Temporal Logic + First Order + Super-dense: **G**lobally, **F**inally, ne**X**t, timed ne**X̃**t

Background: super-dense time model

clock vars

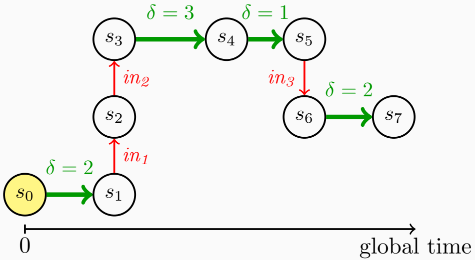
Boolean&theory state vars — Boolean input vars

Infinite-state Timed Transition System $\mathcal{M} \doteq \langle X, C, I, \mathrm{Init}(X), \mathrm{Invar}(X), \mathrm{Trans}(X, I, X')\rangle$

initial states — loc invariant — transition relation

Path of $\mathcal{M}$: $\pi = (s_0, s_1, \ldots)$

- discrete step:$(s_i, in, s'_{i+1}) \models \mathrm{Trans}$
- timed step:
  $\forall c \in C. \quad (s_i, s'_{i+1}) \models (c' = c + \delta)$
  $\forall x \in X \setminus C. \quad (s_i, s'_{i+1}) \models (x' = x)$



Linear Temporal Logic + First Order + Super-dense: **G**lobally, **F**inally, ne**X**t, timed ne**X̃**t

Model-checking: $\mathcal{M} \models_{\exists} \varphi$ iff there exists $\pi \in \Pi(\mathcal{M})$ s.t. $\pi \models \varphi$

Background: SMT and Quantifier Elimination

Satisfiability Modulo Linear Rational Arithmetic
(alternative representation: sets of convex polyhedra)

Background: SMT and Quantifier Elimination

Satisfiability Modulo Linear Rational Arithmetic
(alternative representation: sets of convex polyhedra)

Quantifier Elimination: $\exists X . \Phi(X, Y)$
Projection, predicate abstraction, image computation, reachability, SMT-based model-checking

Background: SMT and Quantifier Elimination

Satisfiability Modulo Linear Rational Arithmetic
(alternative representation: sets of convex polyhedra)

Quantifier Elimination: $\exists X.\Phi(X, Y)$
Projection, predicate abstraction, image computation, reachability, SMT-based model-checking

Pre-Image: $\exists X', I.\mathrm{Trans}(X, I, X') \wedge S(X')$

Background: SMT and Quantifier Elimination

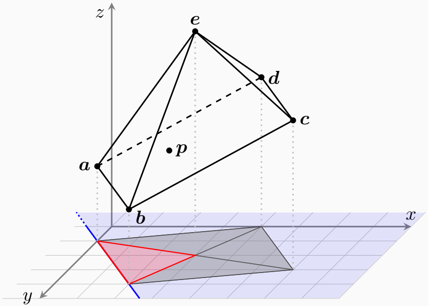
Satisfiability Modulo Linear Rational Arithmetic
(alternative representation: sets of convex polyhedra)

Quantifier Elimination: $\exists X . \Phi(X, Y)$
Projection, predicate abstraction, image computation, reachability, SMT-based model-checking

Pre-Image: $\exists X', I . \mathrm{Trans}(X, I, X') \wedge S(X')$

QE with Bidirectional Model-Based Projection

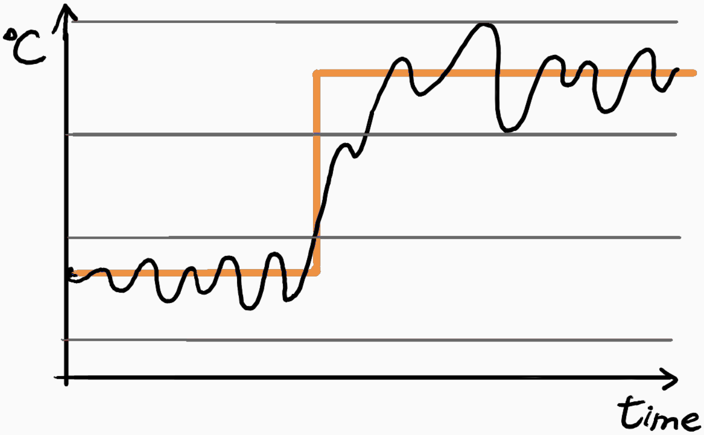
Solve QE by combining over- and under-approximations of convex polyhedra.



Act I

Abstraction Modulo Stability

Stability for Reverse-Engineering: Intuition

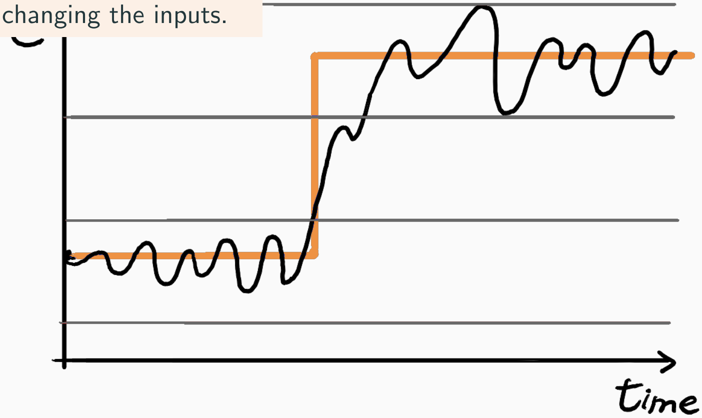


Stability for Reverse-Engineering: Intuition

Stimulus vs input

Inputs are values read by the system.

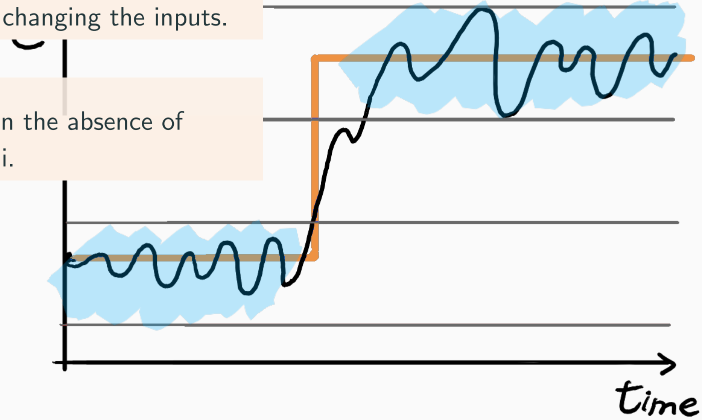
Stimuli are events changing the inputs.



Stability for Reverse-Engineering: Intuition

Stimulus vs input
Inputs are values read by the system.
Stimuli are events changing the inputs.

Closed system
Stability happens in the absence of interrupting stimuli.



Stability for Reverse-Engineering: Intuition

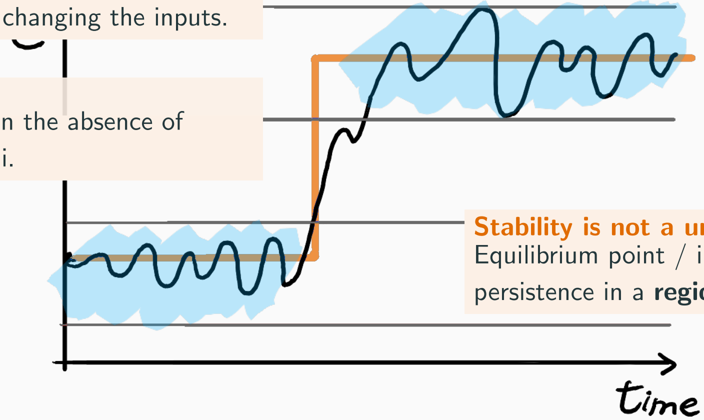
Stimulus vs input

Inputs are values read by the system.

Stimuli are events changing the inputs.

Closed system

Stability happens in the absence of interrupting stimuli.



Stability is not a unique concept

Equilibrium point / invariance / persistence in a **region** ...

Stability for Reverse-Engineering: Intuition

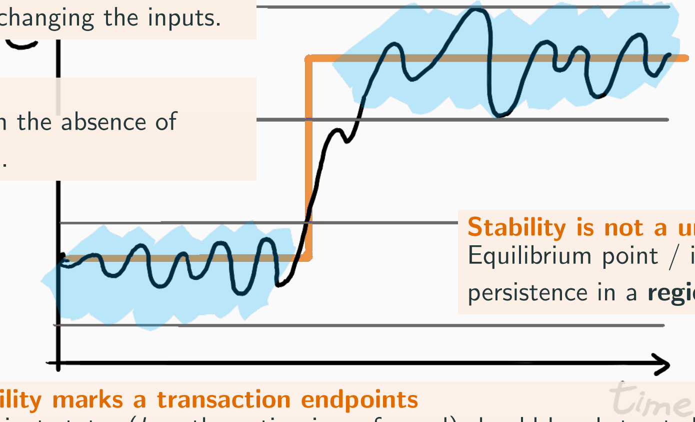
Stimulus vs input

Inputs are values read by the system.

Stimuli are events changing the inputs.

Closed system

Stability happens in the absence of interrupting stimuli.



Stability is not a unique concept

Equilibrium point / invariance / persistence in a **region** ...

Stability marks a transaction endpoints

Transient states (*how* the action is performed) should be abstracted.

Abstraction Modulo σ-Stability

Ingredients:

- concrete system $\mathcal{M} = \langle X, I, \mathrm{Init}(X), \mathrm{Trans}(X, I, X') \rangle$
- predicates $P \subseteq X$
- stability criterion $\sigma(X)$

Abstraction Modulo σ-Stability

Ingredients:

- concrete system $\mathcal{M} = \langle X, I, \mathrm{Init}(X), \mathrm{Trans}(X, I, X') \rangle$
- predicates $P \subseteq X$
- stability criterion $\sigma(X)$

Abstract system:

$$\mathbf{AMS}(\mathcal{M}, \mathbf{P}, \sigma) \doteq \langle \mathbf{P}, \mathbf{I}, \mathbf{Init}_{\mathcal{A}}(\mathbf{P}), \mathbf{Trans}_{\mathcal{A}}(\mathbf{P}, \mathbf{I}, \mathbf{P}') \rangle$$

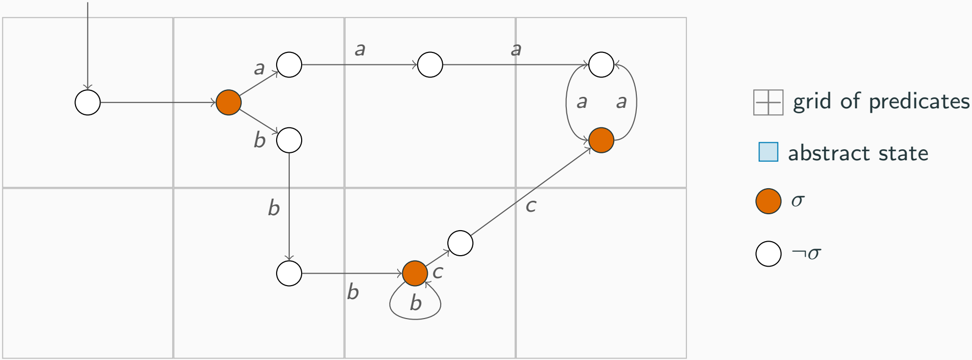
$$\mathrm{Init}_{\mathcal{A}}(P) = \left\{ p_0 \,\middle|\, \mathcal{M}^{\tau} \models_{\exists} (\neg\sigma \ \mathrm{U} \ (\sigma \wedge p_0)) \right\}$$

"at initialization, $\mathcal{M}$ may stabilize in these predicate assignments"

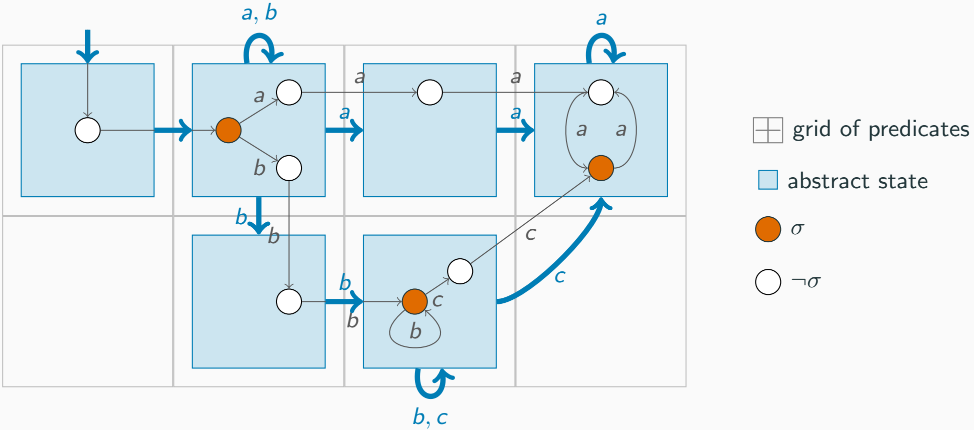
$$\mathrm{Trans}_{\mathcal{A}}(P, I, P') = \left\{ (p_1, in, p_2) \,\middle|\, \mathcal{M} \models_{\exists} \mathrm{F} \left((\sigma \wedge p_1) \wedge \left(\mathrm{G} in \wedge \mathrm{X}(\neg\sigma \ \mathrm{U} \ (\sigma \wedge p_2)) \right) \right) \right\}$$

"if $\mathcal{M}$ is stable in p_1 and receives input in, it will stabilize in p_2"

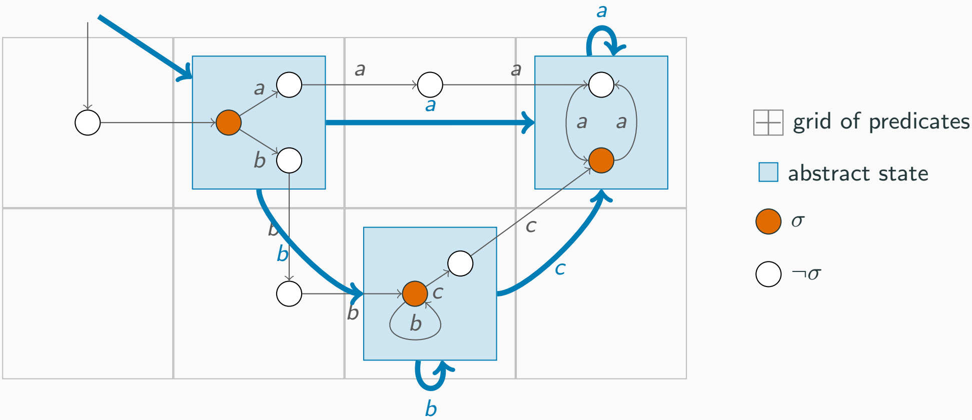
Abstraction modulo σ: more than predicate abstraction



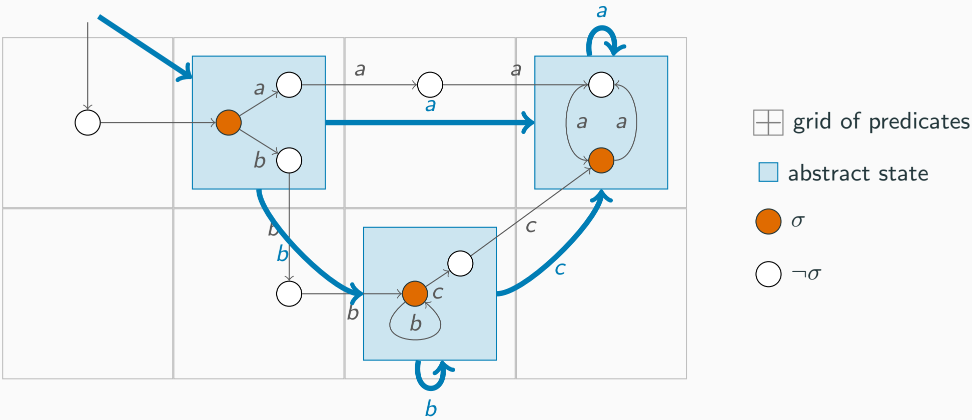
Abstraction modulo σ: more than predicate abstraction



Abstraction modulo σ: more than predicate abstraction



Abstraction modulo σ: more than predicate abstraction



Theorem: If $\sigma \implies \sigma'$, then $\mathcal{A}_\sigma \lesssim \mathcal{A}_{\sigma'}$.

How to: choose σ

Our menu:

- *predicate abstraction*: $\sigma_1 := \text{True}$
- *non-urgent abstraction*: $\sigma_2 := \exists I, X', \delta \;.\; \mathrm{Trans}(X, I, X') \wedge \delta > 0$
- *t-stable abstraction*: $\sigma_3 := \delta > t$
- *same-predicates abstraction*: $\sigma_4 := \left\{ s \in 2^X \;\middle|\; \mathcal{M}^\tau, s \models \mathrm{AG}\, Eq(P) \right\}$

Offline fix-point computation of σ_4

$$\sigma_4(V) \doteq [\![\mathrm{AG}(P = P')]\!] = \neg(\mu Z.\mathrm{preimage}(\mathrm{P} \neq \mathrm{P}') \cup \mathrm{preimage}(Z))$$

How to: build the abstract system

Recall

$$\begin{aligned} \mathrm{Init}_{\mathcal{A}}(P) &= \left\{ p_0 \,\middle|\, \mathcal{M}^{\tau} \models_{\exists} (\neg\sigma \mathbin{\mathrm{U}} (\sigma \wedge p_0)) \right\} \\ \mathrm{Trans}_{\mathcal{A}}(P, I, P') &= \left\{ (p_1, in, p_2) \,\middle|\, \mathcal{M} \models_{\exists} \mathrm{F}\left((\sigma \wedge p_1) \wedge \left(\mathrm{G}\, in \wedge \mathrm{X}(\neg\sigma \mathbin{\mathrm{U}} (\sigma \wedge p_2))\right)\right) \right\} \end{aligned}$$

Algorithms

- **Naïve Enumerative algorithm:** Exponential number of LTL checks.
- **Parameter synthesis:** IC3-based algorithm to find all parameter assignments
- **Bounded algorithm:** QE on BMC-like unrolling until K bound or completeness is checked.

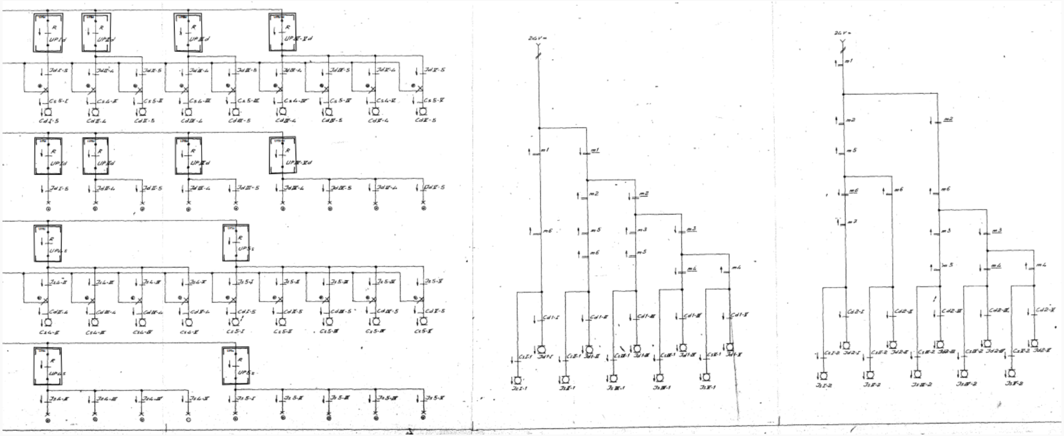
Wrapping up: AMS

- The user picks stability criterion and predicate variables
- Extraction of properties like:
  - *if stable here and this input is received, it should be possible to go there*
  - *this predicate configuration can never be stable*
  - *this input can never stably change this predicate value*
- Algorithms based on LTL model-checking and QE

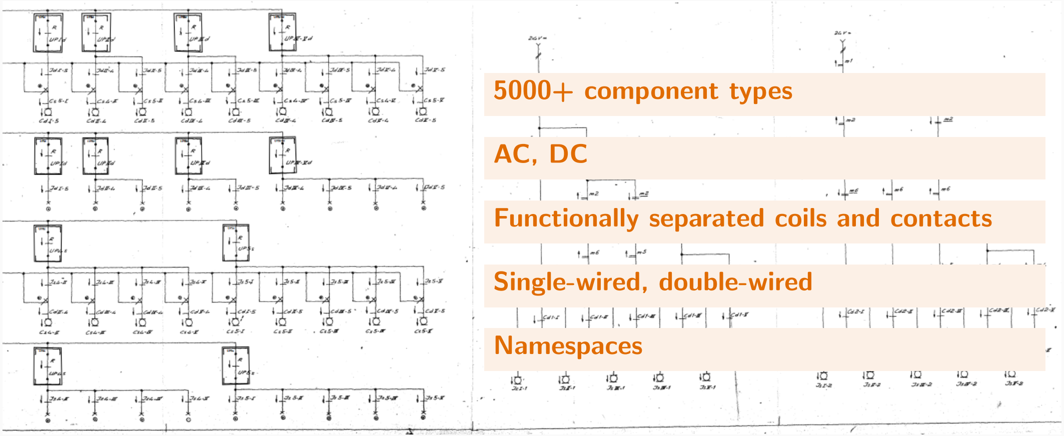
Act II

Model-checking relay-based circuits

Relay-based circuits: a complex domain

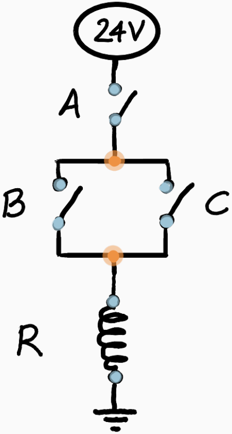


Relay-based circuits: a complex domain

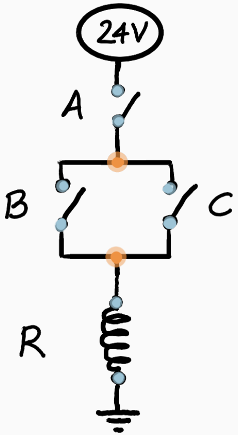


- 5000+ component types
- AC, DC
- Functionally separated coils and contacts
- Single-wired, double-wired
- Namespaces

Encoding relay-based circuits in timed transition systems



Encoding relay-based circuits in timed transition systems



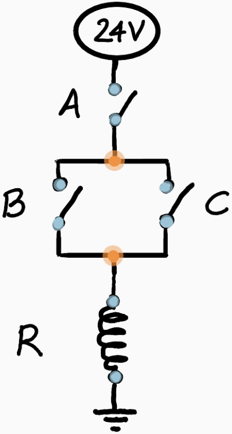
$\neg A \rightarrow (i_A = 0)$
$A \rightarrow (v_A^{in} = v_A^{out})$

$\neg B \rightarrow (i_B = 0)$
$B \rightarrow (v_B^{in} = v_B^{out})$

$\neg C \rightarrow (i_C = 0)$
$C \rightarrow (v_C^{in} = v_C^{out})$

$R \leftrightarrow (i_R > 650\text{Amp})$
$i_R = 190\Omega \cdot (v_R^{in} - v_R^{out})$

Encoding relay-based circuits in timed transition systems



$\neg A \rightarrow (i_A = 0)$
$A \rightarrow (v_A^{in} = v_A^{out})$

$i_A + i_B + i_C = 0$
$v_A^{out} = v_B^{in} = v_C^{in}$

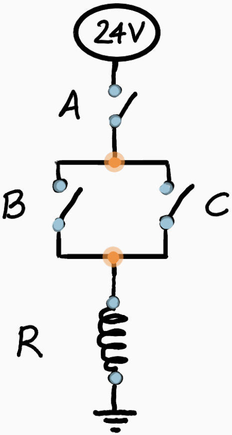
$\neg B \rightarrow (i_B = 0)$
$B \rightarrow (v_B^{in} = v_B^{out})$

$\neg C \rightarrow (i_C = 0)$
$C \rightarrow (v_C^{in} = v_C^{out})$

$i_R + i_B + i_C = 0$
$v_R^{in} = v_B^{out} = v_C^{out}$

$R \leftrightarrow (i_R > 650\text{Amp})$
$i_R = 190\Omega \cdot (v_R^{in} - v_R^{out})$

Encoding relay-based circuits in timed transition systems



$v_A = 24\text{Volt}$

$\neg A \rightarrow (i_A = 0)$
$A \rightarrow (v_A^{in} = v_A^{out})$

$i_A + i_B + i_C = 0$
$v_A^{out} = v_B^{in} = v_C^{in}$

$\neg B \rightarrow (i_B = 0)$
$B \rightarrow (v_B^{in} = v_B^{out})$

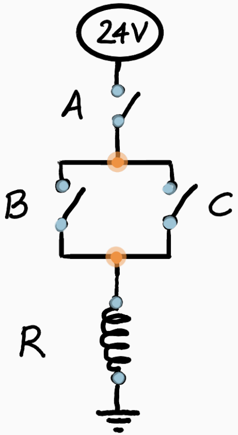
$\neg C \rightarrow (i_C = 0)$
$C \rightarrow (v_C^{in} = v_C^{out})$

$i_R + i_B + i_C = 0$
$v_R^{in} = v_B^{out} = v_C^{out}$

$R \leftrightarrow (i_R > 650\text{Amp})$
$i_R = 190\Omega \cdot (v_R^{in} - v_R^{out})$

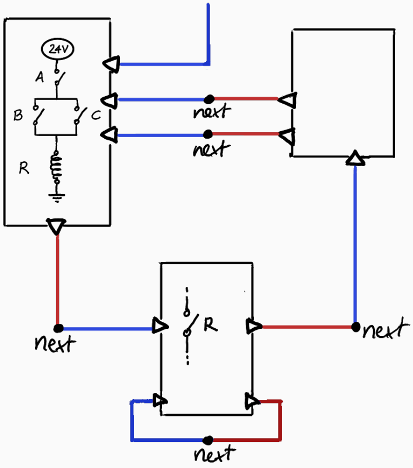
$v_R^{out} = 0$

Encoding relay-based circuits in timed transition systems



$$\text{Invar}\begin{pmatrix} \mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{R}, \\ v_A^{in}, v_A^{out}, i_A, \\ v_B^{in}, v_B^{out}, i_B, \\ v_C^{in}, v_C^{out}, i_C, \\ v_R^{in}, v_R^{out}, i_R \end{pmatrix} := \begin{pmatrix} v_A = 24\text{Volt} \\ \neg A \rightarrow (i_A = 0) \quad A \rightarrow (v_A^{in} = v_A^{out}) \\ i_A + i_B + i_C = 0 \quad v_A^{out} = v_B^{in} = v_C^{in} \\ \neg B \rightarrow (i_B = 0) \quad B \rightarrow (v_B^{in} = v_B^{out}) \qquad \neg C \rightarrow (i_C = 0) \quad C \rightarrow (v_C^{in} = v_C^{out}) \\ i_R + i_B + i_C = 0 \quad v_R^{in} = v_B^{out} = v_C^{out} \\ R \leftrightarrow (i_R > 650\text{Amp}) \quad i_R = 190\Omega \cdot (v_R^{in} - v_R^{out}) \\ v_R^{out} = 0 \end{pmatrix}$$

Encoding relay-based circuits in timed transition systems (cont'd)

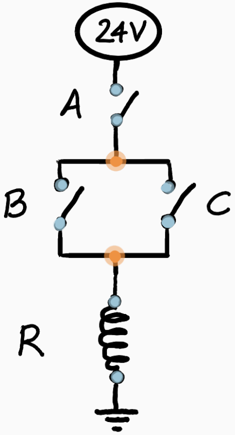


Compose circuits: Link relays with switches

Add a transition step:

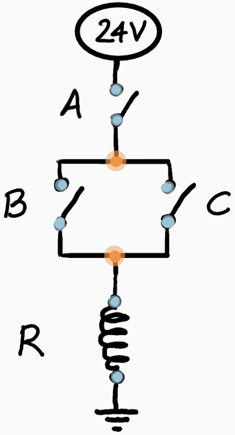
```
TRANS next(switch) = relay
URGENT switch != relay
```

Optimizing the encoding



$$\text{Invar}\begin{pmatrix} A, B, C, R, \\ v_A^{in}, v_A^{out}, i_A, \\ v_B^{in}, v_B^{out}, i_B, \\ v_C^{in}, v_C^{out}, i_C, \\ v_R^{in}, v_R^{out}, i_R \end{pmatrix}$$

Optimizing the encoding

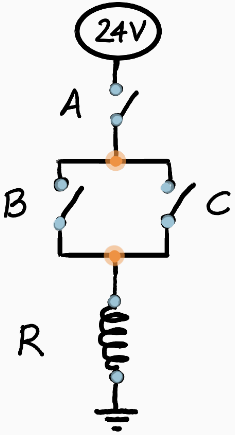


$$\mathrm{Invar}\begin{pmatrix} A, B, C, R, \\ v_A^{in}, v_A^{out}, i_A, \\ v_B^{in}, v_B^{out}, i_B, \\ v_C^{in}, v_C^{out}, i_C, \\ v_R^{in}, v_R^{out}, i_R \end{pmatrix}$$

Check determinism of relays

$$\begin{pmatrix} \mathrm{Invar} \wedge \mathrm{Invar}' \\ \wedge\ A = A' \\ \wedge\ B = B' \\ \wedge\ C = C' \end{pmatrix} \models (R = R')$$

Optimizing the encoding



$$\mathrm{Invar}\begin{pmatrix} A, B, C, R, \\ v_A^{in}, v_A^{out}, i_A, \\ v_B^{in}, v_B^{out}, i_B, \\ v_C^{in}, v_C^{out}, i_C, \\ v_R^{in}, v_R^{out}, i_R \end{pmatrix}$$

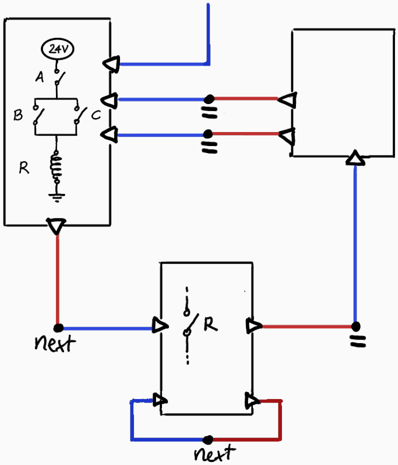
Check determinism of relays

$$\begin{pmatrix} \mathrm{Invar} \wedge \mathrm{Invar}' \\ \wedge\ A = A' \\ \wedge\ B = B' \\ \wedge\ C = C' \end{pmatrix} \models (R = R')$$

Optimized Invariant: remove electrical variables

$$\mathbf{Invar}^{opt}(\mathbf{A}, \mathbf{B}, \mathbf{C}, \mathbf{R}) := (\exists\ \mathbf{Reals}.\mathbf{Invar}) = (\mathbf{R} \leftrightarrow (\mathbf{A} \wedge (\mathbf{B} \vee \mathbf{C})))$$

Optimizing the encoding (cont'd)

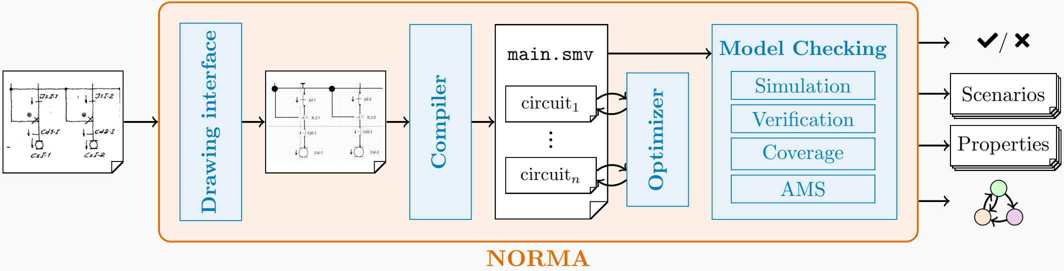


Only to break cyclic dependencies

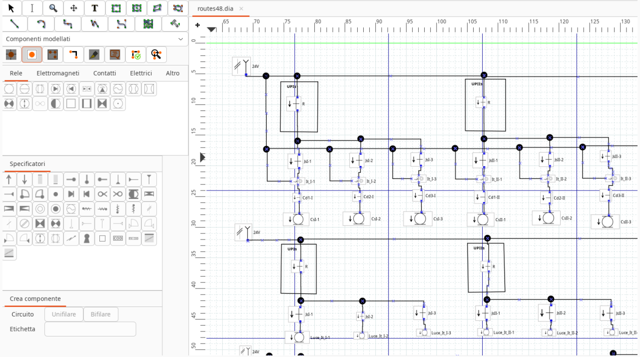
```
TRANS next(switch) = relay
URGENT switch != relay
```

Everywhere else switch = relay

Norma: a tool for the analysis of relay-based circuits

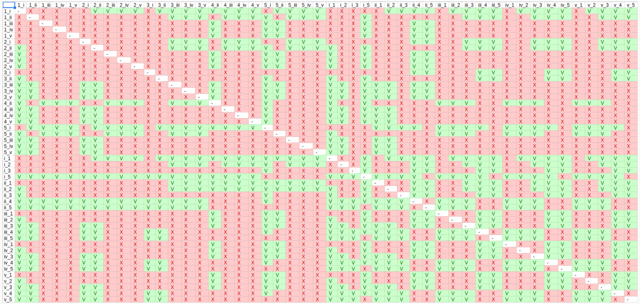


Norma: digital modelling, simulation and verification made possible



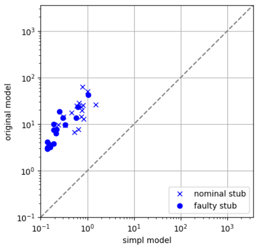

```
--------------------------------------------------------------------------
MODULE main

    -- Sub-modules:
    VAR root : CircuitRoot;

--------------------------------------------------------------------------
MODULE CircuitRoot

    -- Sub-modules:
    VAR net00 : Net00(
            stub."RTe",
            net14."A_f0$0935".interface,
            stub."CTR_normale",
            stub."CTR_rovescio");
    VAR net11 : Net11(stub."CTR_elettromagnete");
    VAR net06 : Net06(
            net05."AEDX_f0$01279".interface,
            stub."Ad",
            net11."CEDX_f0$0898".interface,
            net05."BEDX_f0$01281".interface);
    VAR net03 : Net03(
            net01."DX_f0$01022".interface,
            net13."M_f0$0934".interface,
            net08."CBi_f0$0848".interface,
            stub."bp",
            stub."bc",
            net05."BEDX_f0$01281".interface,
            net05."AEDX_f0$01279".interface,
            net11."CEDX_f0$0898".interface,
            net04."Racc_f0$01060".interface,
            net01."mX_f0$01014".interface,
            net01."_DX_f0$01206".interface,
            net01."_mX_f0$01207".interface);
```

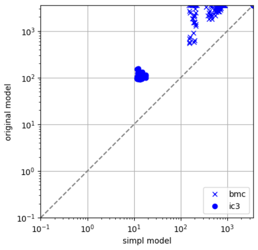
Norma: digital modelling, simulation and verification made possible



`table[i,j]` = ✖: if route i is accepted, route j must wait

`table[i,j]` = ✔: route i and j can be accepted together

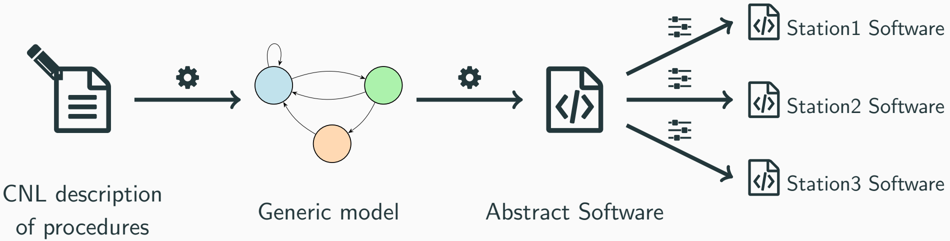




Act III

Validating the migration from relay-based to software interlocking

RFI's development of software interlocking



My goal: StationX Relays $\simeq$ StationX Software?

Use AMS to link the circuits with the new code

1. **Extraction of simulations from relay circuits $\mathcal{M}$**
2. **Translate simulations in sw system tests**
3. **Document differences / report bugs**

Use AMS to link the circuits with the new code

1. Extraction of simulations from relay circuits $\mathcal{M}$

800+ scenarios

Cover the AMS machine: for all $(p_1, in, p_2) \in \mathrm{Trans}_{\mathcal{A}}$, collect

$$\pi \models F((\sigma \wedge p_1) \wedge \mathrm{G}(in) \wedge \mathrm{X}(\neg\sigma \ \mathrm{U}\ (\sigma \wedge p_2)))$$

(+ other coverage criteria)

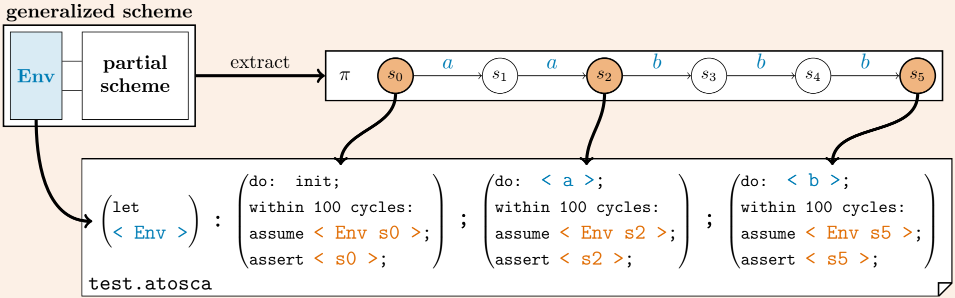
2. Translate simulations in sw system tests

3. Document differences / report bugs

Use AMS to link the circuits with the new code

1. Extraction of simulations from relay circuits $\mathcal{M}$ — 800+ scenarios

2. Translate stable simulations in sw system tests — 7K+ tests for 8 stations



3. Document differences / report bugs

Use AMS to link the circuits with the new code

1. Extraction of simulations from relay circuits $\mathcal{M}$ — **800+ scenarios**

2. Translate simulations in sw system tests — **7K+ tests for 8 stations**

3. Document differences / report bugs — **10+ bugs from errors in the specs!**

Execute tests on our platform simulator:

- ✔ pass : a behavior in the circuits is realizable in the sw
- fail with violation of environment assumptions: document
- fail : unexpected discrepancy between circuits and sw!

Conclusions

Abstractions for reverse-engineering: wrapping up

What we did

- AMS framework
- </> AMS implementation (nuXmv, pySMT)
- Optimal encoding of relay circuits
- </> Norma tool for verifying circuits
- Extract properties and tests from circuits
- Testing the new software
- BMBP: a new QE algorithm for LRA
- </> Tool for reachability analysis of pwc HS

Abstractions for reverse-engineering: wrapping up

What we did

- AMS framework
- AMS implementation (nuXmv, pySMT)
- Optimal encoding of relay circuits
- Norma tool for verifying circuits
- Extract properties and tests from circuits
- Testing the new software
- BMBP: a new QE algorithm for LRA
- Tool for reachability analysis of pwc HS

Future works

- Improve integration with RFI toolset
- Verify circuit properties on the code
- Component-based reasoning on circuits
- Improve QE calls in AMS algorithms
- Other applications for AMS
- Extend BMBP-QE to other theories
- BMBP-QE in model-checkers

Thank you!

Stability: related works

Lyapunov stability in control theory [Liberzon03]
For $\dot{x} = f(x)$, stability is witnessed by a Lyapunov function $V : \mathbb{R}^n \to \mathbb{R}$

- $V(\boldsymbol{x}_{eq}) = 0$ and $V(\boldsymbol{x}) > 0$ for all $\boldsymbol{x} \neq \boldsymbol{x}_{eq}$
- $\dot{V}(\boldsymbol{x}) < 0$ for all $\boldsymbol{x} \neq \boldsymbol{x}_{eq}$

Exponentially stable if: $\dot{V}(\boldsymbol{x}) \leq -\alpha V(\boldsymbol{x})$.

Synthesis of a Lyapunov function in dynamical systems

- Sum of squares decomposition [PapachristodoulouP02],
- counter-example guided synthesis [AbatePA20],
- numeric synthesis and SMT-based verification [Basagiannis+23]

Stability: related works

Stability in hybrid systems

- Global or piecewise quadratic Lyapunov function [Oehlerking11, JohanssonR98]
- guided synthesis [RavanbakhshS15],
- deductive verification [TanMP22]

Find conditions on switching sequences: average dwell time [BogomologMP10, MitraL04] and **slow switching constraints**.

Region Stability
Classical asymptotic stability is not adequate for most systems [PodelskiW06]. Verify invariance in a region with a model transformation.

Hybrid systems verification: related works

- Reachability analysis on polyhedra: HyTech [HHWT97], PHAVer [Frehse05,Frehse08, BecchiZ19]
- Flowpipe construction: HyPro [SchuppÀMK17, SchuppÀE22], Ariadne [CollinsBGV12], JuliaReach [BogomolovFFPS19], Flow* [ChenAS13]
- Theorem prover on differential dynamic logic: KeYmaera [PlatzerQ08]
- Model-checker via discretization: HyComp [CimattiGMT15]
- Model-checker for infinite-state timed TS: Timed nuXmv [CimattiGMRT19]

LTL model checking

Finite state systems: search for a cex as a lasso-shaped path $\pi = uv^\omega$.

- BMC [BiereCCZ99, BiereCCSZ03]: SAT-based search of a path of length k
- Liveness2Safety [BiereAS02]: reduce to $FG\phi$, then look for a cex where $\neg\phi$ holds infinitely often.
- K-Liveness [ClaessenS12, CimattiGMT14]: $\neg\phi$ holds at most k times
- rlive [XiaCGL24]: reachability of a bad state starting from an already proven reachable bad state

Infinite state systems: a non lasso-shaped cex may exist

- Liveness2Safety + implicit Predicate Abstraction [DanielCGTM16]
- rlive + implicit predicate abstraction + well founded sets [CimattiGJRT25]
- non lasso-shaped search [CimattiGM21, CimattiGM22]: cex are sequence of well founded funnels searched in a predicate space

Railway interlocking: related works

Interlocking Verification

- Verification of control tables from railway layout with: graph theory [SheSCY07], CSP [Winter02], Colored Petri Nets [SheZY14]
- Model-checking on general railway rules + configurations: BMP and K-induction [HongHP17, VuHP17]
- In Italy: computerized logic [CimattiGMRTT98] verified with SPIN.
- Compositional reasoning: [HaxthausenF23]

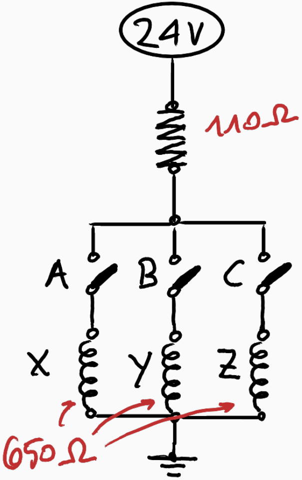
Generation of new software interlocking
French railways:

- generation of event-B code form UML [BougachaWKAD19]
- generation **from relays**: from colored petri nets, to abstract machines in B-method, to code [SunDB15, SunBD15, PereiraDPB19, Pereira20]

Relay-based interlocking verification

- Danish relay-based interlocking: modelled in SAL [HaxthausenKB11], Boolean encoding [Haxthausen14]. Verified against properties extracted from interlocking tables (LTL).
- Ladder logic [GhoshDBDK17], to sets of Boolean equations, to propositional logic
- CSP [PereiraOBBD22] for capturing transient behavior and verify against ring-bell effects
- Italian case: SMT-based encoding of multi-domain Kirchhoff networks[CimattiMS17, CavadaCMSC18]. Norma inherits **electrical accuracy**

Why electrical accuracy is important



Relay activation condition

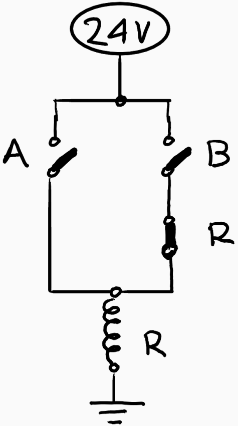
$$X \iff (i_X > 19/650)$$

If A, B and C are closed, then none of X, Y or Z can be active!

Correct invariant is:

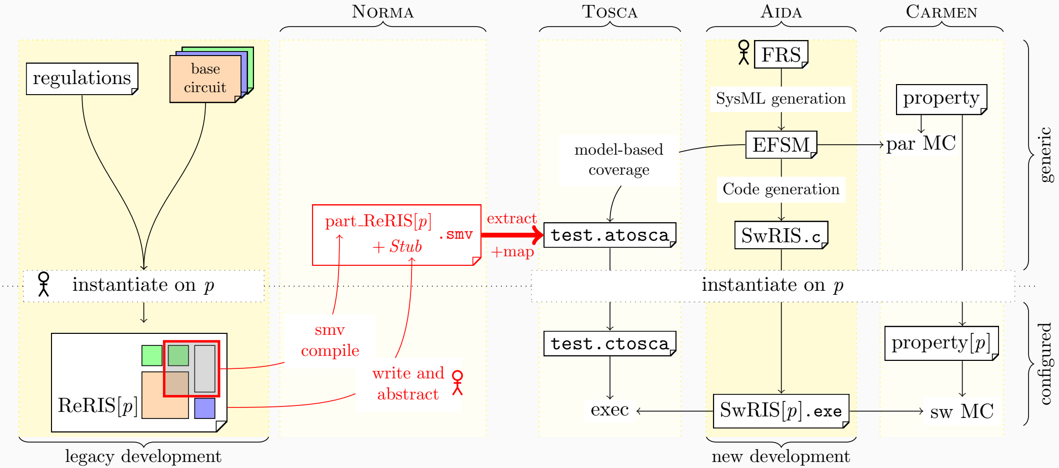
$$X \iff (A \land (\neg B \lor \neg C))$$
$$Y \iff (B \land (\neg A \lor \neg C))$$
$$Z \iff (C \land (\neg A \lor \neg B))$$

Stable transition properties



Only A can stably change R.
Whatever B does, it cannot stably change R.

RFI tolset



Norma
Tosca
Aida
Carmen
regulations
base circuit
FRS
SysML generation
property
model-based coverage
EFSM
par MC
Code generation
part_ReRIS[p] .smv
+ Stub
extract
+map
test.atosca
SwRIS.c
generic
instantiate on p
instantiate on p
smv compile
test.ctosca
property[p]
configured
write and abstract
ReRIS[p]
exec
SwRIS[p].exe
sw MC
legacy development
new development

Algorithms for AMS

(Naïve) Enumerative algorithm

- $\mathrm{Init}_{\mathcal{A}}$: $2^{|P|}$ checks like $\mathcal{M}^{\tau} \models_{\exists} \neg\hat{\sigma}\ \mathrm{U}\ (\hat{\sigma} \wedge p)$
- $\mathrm{Trans}_{\mathcal{A}}$: $2^{|P| \times |I| \times |P|}$ checks like $\mathcal{M} \models_{\exists} \mathrm{F}\,((\sigma \wedge p_1) \wedge \mathrm{G}\,in \wedge \mathrm{X}(\neg\sigma\ \mathrm{U}\ (\sigma \wedge p_2)))$

Parameter synthesis

- Introduce parameters $\overline{P_1}, \overline{I}, \overline{P_2}$.
- $\mathrm{Init}_{\mathcal{A}}$: find all $\overline{p_1}$ s.t. $\mathcal{M}^{\tau}[\overline{p_1}] \models_{\exists} \neg\hat{\sigma}\ \mathrm{U}\ (\hat{\sigma} \wedge (P = \overline{P}_1))$,
- $\mathrm{Trans}_{\mathcal{A}}$: find all $\overline{p_1}, \overline{in}, \overline{p_2}$ s.t. $\mathcal{M}[(\overline{p}_1, \overline{in}, \overline{p}_2)] \models_{\exists} \mathrm{F}\begin{pmatrix}(\hat{\sigma} \wedge P = \overline{P}_1) \wedge \mathrm{G}(I = \overline{I}) \\ \wedge\ \mathrm{X}(\neg\hat{\sigma}\ \mathrm{U}\ (\hat{\sigma} \wedge P = \overline{P}_2))\end{pmatrix}$

Algorithms for AMS (cont'd)

Bounded algorithm: QE on BMC-like unrolling until completeness

- $\mathrm{Init}_{\mathcal{A}} := \bigvee_k \exists(\mathrm{FV}(\pi_k) \setminus P_k)\ .\ \pi_k$

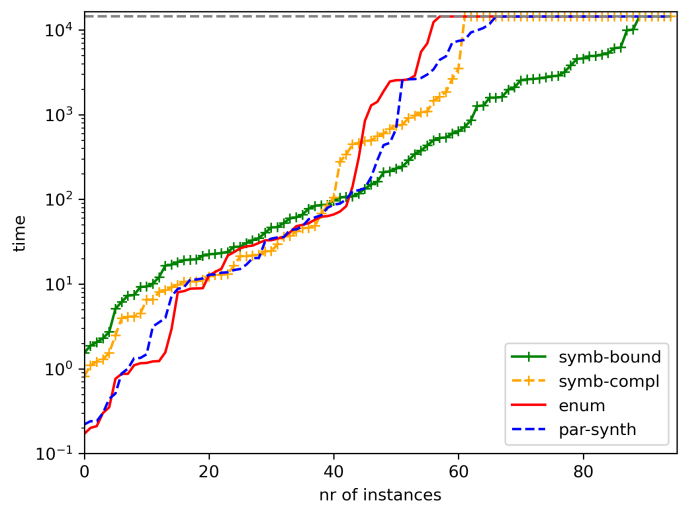
$$\pi_k := \mathrm{Init}(V_0) \wedge \bigwedge_{0 \le h < k} (\mathrm{Trans}(V_h, I_h, V_{h+1}) \wedge I_h = I_{h+1} \wedge \neg\hat{\sigma}(V_h)) \wedge \hat{\sigma}(V_k)$$

- completeness check: $\mathcal{M}^\tau \models \neg(\neg\hat{\sigma}\ \mathrm{U}\ (\hat{\sigma} \wedge \neg\mathrm{Init}_{\mathcal{A}}))$
- $\mathrm{Trans}_{\mathcal{A}} := \bigvee_{i<j} \exists(\mathrm{FV}(\pi_{i,j}) \setminus P_i \setminus I_j \setminus P_k)\ .\ \pi_{i,j}$

$$\pi_{i,j} \doteq \mathrm{Init}(V_0) \wedge \bigwedge_{0 \le h < j} \mathrm{Trans}(V_h, I_h, V_{h+1}) \wedge \begin{pmatrix} \hat{\sigma}(V_i) \wedge \hat{\sigma}(V_j) \wedge \\ \bigwedge_{i<h<j} (I_h = I_{h+1} \wedge \neg\hat{\sigma}(V_h)) \end{pmatrix}$$

- $\mathrm{iscomplete}(p_1, p_2) \doteq \neg\mathrm{F}\left((\hat{\sigma} \wedge p_1) \wedge \begin{pmatrix} \mathrm{G}(I = \mathrm{X}I) \wedge \\ \neg\mathrm{Trans}_\sigma[p_1, p_2] \end{pmatrix} \wedge \mathrm{X}(\neg\hat{\sigma}\ \mathrm{U}\ (\hat{\sigma} \wedge p_2))\right)$

Comparing AMS algorithms



Minimal P-stable abstraction

Intuition

- Instance of AMS to capture invariance in multiple predicates.
- Capture oscillation in regions larger than $p \in 2^P$.
- Find minimal region ϕ that are eventually invariant.

Definition

- $\textsc{attr}(s, \phi) \doteq (\mathcal{M}^\tau, s \models \mathrm{EFAG}\phi)$
- $\textsc{no-str-attr}(s, \phi) \doteq (\nexists \phi' \in \Phi. (\phi' \neq \phi \wedge (\phi' \models \phi) \wedge \textsc{attr}(s, \phi')))$
- $\mathrm{STABLE}_{\min}(s, \phi) \doteq (\mathcal{M}^\tau, s \models \mathrm{AG}\phi) \wedge \textsc{no-str-attr}(s, \phi)$
- $\mathrm{ATTR}_{\min}(s, \phi) \doteq \textsc{attr}(s, \phi) \wedge \textsc{no-str-attr}(s, \phi)$

Minimal P-stable abstraction (cont'd)

As Galois Connection $(\wp(\Sigma), \subseteq) \underset{\alpha_1}{\overset{\gamma_1}{\leftrightarrows}} (\wp(\Phi), \subseteq)$

- $\alpha_1(S) \doteq \left\{ \phi \in \Phi \,\middle|\, \exists s \in S . \text{ATTR}_{\min}(s, \phi) \right\}$
- $\gamma_1(F) \doteq \left\{ s \in \Sigma \,\middle|\, \forall \phi \in \Phi . \text{ATTR}_{\min}(s, \phi) \implies \phi \in F \right\}$

Further abstractions

$$(\wp(\Sigma), \subseteq) \underset{\alpha_1}{\overset{\gamma_1}{\leftrightarrows}} (\wp(\Phi), \subseteq) \underset{\alpha_2}{\overset{\gamma_2}{\leftrightarrows}} (\Phi, \models) \underset{\alpha_3}{\overset{\gamma_3}{\leftrightarrows}} (\mathbb{K} \models).$$

Minimal P-stable abstraction: Implemantation