# Verification-Modulo-Theory techniques

## BMC, interpolation-based, k-induction, predicate abstraction and a new combination thereof

Stefano Tonetta


FBK-irst, Trento, Italy
{tonettas}@fbk.eu

RichModels Workshop, Turin, 4 October 2011

# Motivations I

- VMT applied to support the development of embedded systems.
- Industrial effort in model-based system engineering.
- Validation and verification integrated in the development process.
- Many languages used to describe requirements, system architectures, software and hardware components (UML, SysML, MARTE, CHESS, SystemC, Altarica, ...).
- Formal approaches to validation and verification assign a formal semantics to the design models.
- Complexity of embedded systems:
  - Hundreds of functions
  - Networked control
  - Real-time constraints
  - Complex execution model with mixture of real-time and event-based triggers

# Motivations II

- Transition systems are a generic formalism suitable to capture the peculiarities of the different languages (a relational form is sometimes necessary).
- Symbolic representation allows compact and easy translations.
- VMT reduces the verification to a (possibly infinite) series of satisfiability checks.
- SMT is fundamental to increase the richness of the models.

# Outline

# Outline

# Symbolic transition systems

- $V$: state variables.
- $V'$: next variables.
- A symbolic transition system is a tuple $S = \langle V, I, T \rangle$ where:
  - $I$ is a first order formula over $V$ (called initial condition),
  - $T$ is a first order formula over $V \cup V'$ (called transition condition),

  where the conditions are formulas of some decidable first-order logic.
- Models are sequences $\pi$ of assignments to $V$ such that:
  - $\pi_0 \models I$,
  - $\pi_i, \pi'_{i+1} \models T$, for all $i$, $0 \leq i < |\pi|$.

# First-order transition systems

- Refined definition.
- $\Sigma$: first-order signature;
- $\Sigma_r \subseteq \Sigma$ signature of rigid symbols;
- $\Sigma_f = \Sigma \setminus \Sigma_r$ flexible symbols;
- $\Sigma' = \{s'\}_{s \in \Sigma_f}$: next symbols;
- A first-order transition system is a tuple $S = \langle \Sigma, \Sigma_r, I, T \rangle$ where:
  - $I$ is a $\Sigma$-formula,
  - $T$ is a $\Sigma \cup \Sigma'$-formula.
- $\mathcal{T}$: a $\Sigma$-theory.
- Models are sequences $\pi$ of $\mathcal{T}$-models with the same domain and same interpretation of the rigid symbols in $\Sigma_r$ such that:
  - $\pi_0 \models I$,
  - $\pi_i, \pi'_{i+1} \models T$, for all $i$, $0 \leq i < |\pi|$.

# First-order transition systems

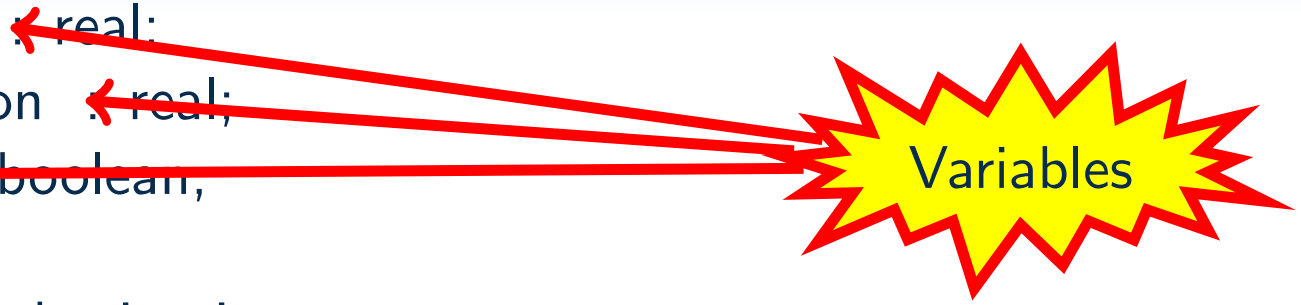
- Refined definition.
- $\Sigma$: first-order signature;
- $\Sigma_r \subseteq \Sigma$ signature of rigid symbols;
- $\Sigma_f = \Sigma \setminus \Sigma_r$ flexible symbols;
- $\Sigma' = \{s'\}_{s \in \Sigma_f}$: next symbols;
- A first-order transition system is a tuple $S = \langle \Sigma, \Sigma_r, I, T \rangle$ where:
  - $I$ is a $\Sigma$-formula,
  - $T$ is a $\Sigma \cup \Sigma'$-formula.
- $\mathcal{T}$: a $\Sigma$-theory.
- Models are sequences $\pi$ of $\mathcal{T}$-models with the same domain and same interpretation of the rigid symbols in $\Sigma_r$ such that:
  - $\pi_0 \models I$,
  - $\pi_i, \pi'_{i+1} \models T$, for all $i$, $0 \leq i < |\pi|$.
- We keep the standard notation where $V = \Sigma_f$ and $\Sigma_r$ is omitted (assuming no functions, no parameters, ...).

# A hybrid system example

VAR
location : real;
destination : real;
timed : boolean;
INIT
location$\leq$destination
TRANS
!timed $\rightarrow$
(next(location)=location &
next(destination)$\geq$location)
TRANS
timed $\rightarrow$
(location=destination $\rightarrow$
next(location)=location)

# A hybrid system example

VAR
location : real;
destination : real;
timed : boolean;
INIT
location$\leq$destination
TRANS
!timed $\rightarrow$
(next(location)=location &
next(destination)$\geq$location)
TRANS
timed $\rightarrow$
(location=destination $\rightarrow$
next(location)=location)

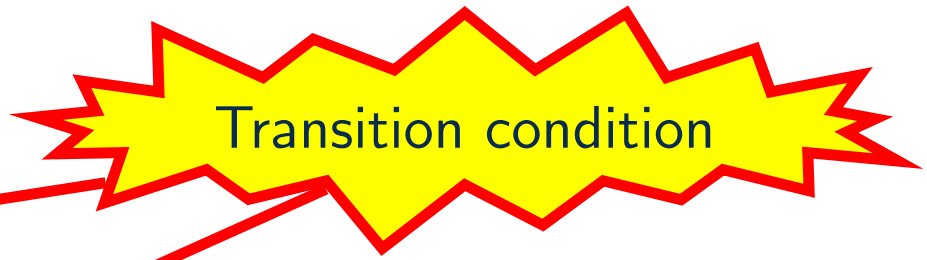


# A hybrid system example

VAR
location : real;
destination : real;
timed : boolean;
INIT
location$\leq$destination
TRANS
!timed $\rightarrow$
(next(location)=location &
next(destination)$\geq$location)
TRANS
timed $\rightarrow$
(location=destination $\rightarrow$
next(location)=location)

# A hybrid system example

```
VAR
location  : real;
destination  : real;
timed  : boolean;
INIT
location≤destination
TRANS
!timed →
  (next(location)=location &
    next(destination)≥location)
TRANS
timed →
  (location=destination →
    next(location)=location)
```



# Reachability modulo theory

## Reachability

Given a transition system $S$, a theory $\mathcal{T}$, and a formula $\phi$, is there a finite sequence $\pi$ of $\mathcal{T}$-models such that:

- $\pi$ is a run of $S$;
- $\pi_{|\pi|} \models \phi$.

## Language emptiness

Given a transition system $S$, a theory $\mathcal{T}$, and a formula $\phi$, is there an infinite sequence $\pi$ of $\mathcal{T}$-models such that:

- $\pi$ is a run of $S$;
- $\pi_i \models \phi$ for infinitely many $i$.

# Reachability modulo theory

## Reachability

Given a transition system $S$, a theory $\mathcal{T}$, and a formula $\phi$, is there a finite sequence $\pi$ of $\mathcal{T}$-models such that:

- $\pi$ is a run of $S$;
- $\pi_{|\pi|} \models \phi$.

## Language emptiness

Given a transition system $S$, a theory $\mathcal{T}$, and a formula $\phi$, is there an infinite sequence $\pi$ of $\mathcal{T}$-models such that:

- $\pi$ is a run of $S$;
- $\pi_i \models \phi$ for infinitely many $i$.

- Undecidable.
- Sound but incomplete solution, no guarantee on termination.
- Focus on reachability, but the techniques can be easily adapted to fairness.

# Outline

# Symbolic model checking

- VMT techniques are particular cases of Symbolic Model Checking (SMC)
- SMC characteristics: manipulate formulas instead of states.
- Typical operations are union, conjunction, sat, quantifier elimination.
- VMT uses SMT:
  - Satisfiability
  - Incrementality
  - Model extraction
  - Unsat core extraction
  - Interpolation
  - Quantifier elimination / ALLSMT
- Since quantifier elimination is expensive, many techniques are based on sat only.

# BMC

- Determine if $\phi$ is reachable in $k$ steps.
- State variables replicated $k+1$ times: $V^0, V^1, \ldots, V^{k-1}, V^k$.
- Given $\psi(V)$, denote $\psi[V^i/V]$ with $\psi^i$.
- Given $\psi(V, V')$, denote $\psi[V^i/V, V^{i+1}/V']$ with $\psi^i$.
- Encoding of an initial path reaching $\phi$:

$$I^0 \wedge T^0 \wedge T^1 \wedge \ldots \wedge T^{k-1} \wedge \phi^k$$

- Incremental approach:

$I^0 \wedge \phi^0$

$I^0 \wedge T^0 \wedge \phi^1$

$I^0 \wedge T^0 \wedge T^1 \wedge \phi^2$

$\ldots$

$I^0 \wedge T^0 \wedge T^1 \wedge \ldots \wedge T^{k-1} \wedge \phi^k$

# Interpolation-based model checking

- If $A \wedge B \models \bot$, the Craig *interpolant* of $A \wedge B$ is a formula $I$ such that $\models A \rightarrow I$, $B \wedge I \models \bot$, and which contains only variables common to $A$ and $B$.
- Interpolation-based model checking:
  1. $I^0 \wedge T^0 \wedge T^1 \wedge \ldots \wedge T^{k-1} \wedge \phi^k$

# Interpolation-based model checking

- If $A \wedge B \models \bot$, the Craig *interpolant* of $A \wedge B$ is a formula $I$ such that $\models A \rightarrow I$, $B \wedge I \models \bot$, and which contains only variables common to $A$ and $B$.
- Interpolation-based model checking:
  1. $I^0 \wedge T^0 \wedge T^1 \wedge \ldots \wedge T^{k-1} \wedge \phi^k$
     $\rightsquigarrow I_1$ interpolant over-approximating the reachable states

# Interpolation-based model checking

- If $A \wedge B \models \bot$, the Craig *interpolant* of $A \wedge B$ is a formula $I$ such that $\models A \rightarrow I$, $B \wedge I \models \bot$, and which contains only variables common to $A$ and $B$.
- Interpolation-based model checking:
  1. $I^0 \wedge T^0 \wedge T^1 \wedge \ldots \wedge T^{k-1} \wedge \phi^k$
  2. $I_1^0 \wedge T^0 \wedge T^1 \wedge \ldots \wedge T^{k-1} \wedge \phi^k$

# Interpolation-based model checking

- If $A \wedge B \models \bot$, the Craig *interpolant* of $A \wedge B$ is a formula $I$ such that $\models A \rightarrow I$, $B \wedge I \models \bot$, and which contains only variables common to $A$ and $B$.
- Interpolation-based model checking:
  1. $I^0 \wedge T^0 \wedge T^1 \wedge \ldots \wedge T^{k-1} \wedge \phi^k$
  2. $I_1^0 \wedge T^0 \wedge T^1 \wedge \ldots \wedge T^{k-1} \wedge \phi^k \rightsquigarrow I_2$

# Interpolation-based model checking

- If $A \wedge B \models \bot$, the Craig *interpolant* of $A \wedge B$ is a formula $I$ such that $\models A \rightarrow I$, $B \wedge I \models \bot$, and which contains only variables common to $A$ and $B$.
- Interpolation-based model checking:
  1. $I^0 \wedge T^0 \wedge T^1 \wedge \ldots \wedge T^{k-1} \wedge \phi^k$
  2. $I_1^0 \wedge T^0 \wedge T^1 \wedge \ldots \wedge T^{k-1} \wedge \phi^k$
  3. ... until fixpoint.
- If sat with abstract initial states, $k$ is increased.

# K-induction

- K-induction proves that if a set of states is not reachable in $k$ steps, then it is not reachable at all.
- It consists of a base step (bounded reachability problem), and an inductive step.
- Two ways:
  - check if the initial states cannot reach new states in $k+1$ steps
  - check if the target set of states cannot be reached in $k+1$ steps.
- Solved by means of satisfiability.

$$kindfw_k := I^0 \wedge T^0 \wedge T^1 \wedge \ldots \wedge T^{k-1} \wedge \bigwedge_{0 \leq i < j \leq k} V^i \neq V^j$$

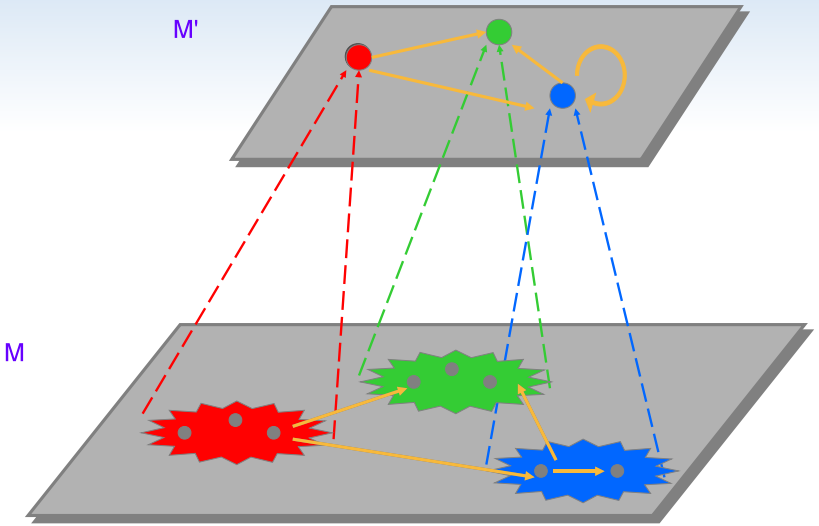
$$kindbw_{k,\phi} := T^0 \wedge T^1 \wedge \ldots \wedge T^{k-1} \wedge \phi^k \wedge \bigwedge_{0 \leq i < j \leq k} V^i \neq V^j$$

- If, for all $i \leq k$, $BMC_{i,\phi}$ is unsat and, either $kindfw_{k+1}$ or $kindbw_{k+1,\phi}$ is unsat as well, then $\phi$ is not reachable in $S$.

# Predicate abstraction

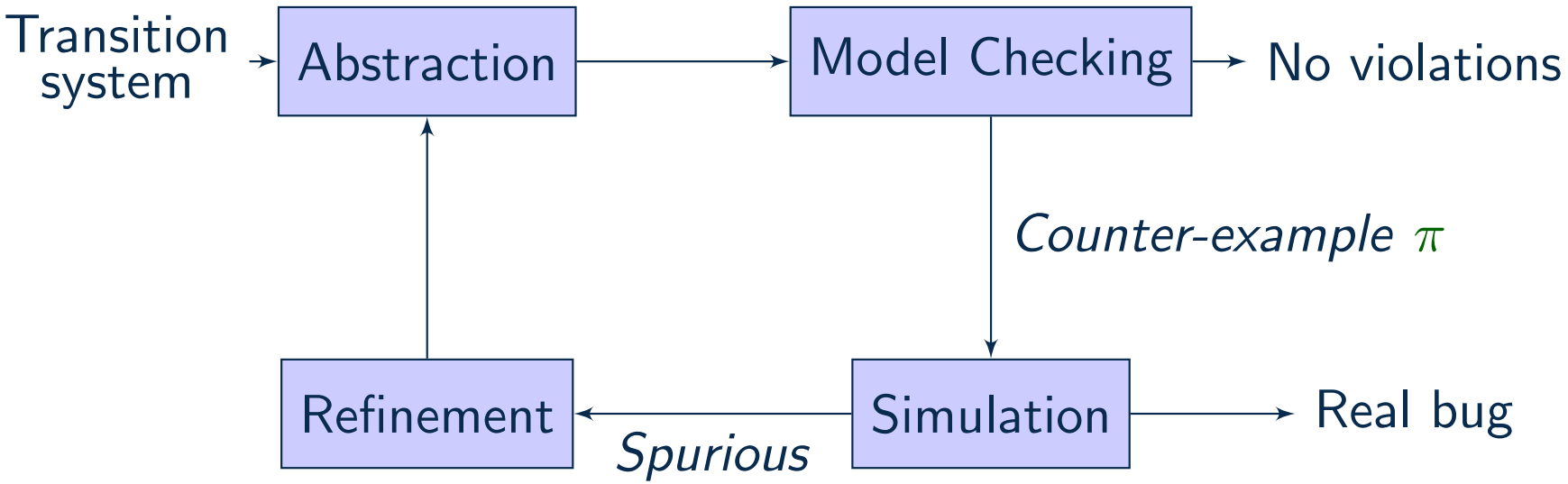
## Existential abstraction

- $\hat{I}_\alpha(\hat{V}) := \exists V(I(V) \wedge H_\alpha(V, \hat{V}))$
- $\hat{T}_\alpha(\hat{V}, \hat{V}') := \exists V \exists V'(T(V, V') \wedge H_\alpha(V, \hat{V}) \wedge H_\alpha(V', \hat{V}'))$




- Predicate abstraction: abstract state-space is described with a set of predicates $\mathbb{P}$.
- Each predicate is represented by an abstract variable ($V_\mathbb{P} = \{v_P\}_{P \in \mathbb{P}}$).
- Abstract relation:
  $H_\mathbb{P}(V, V_\mathbb{P}) := \bigwedge_{P \in \mathbb{P}} v_P \leftrightarrow P(V)$
- Quantifier elimination with ALL-SMT.

# CEGAR loop

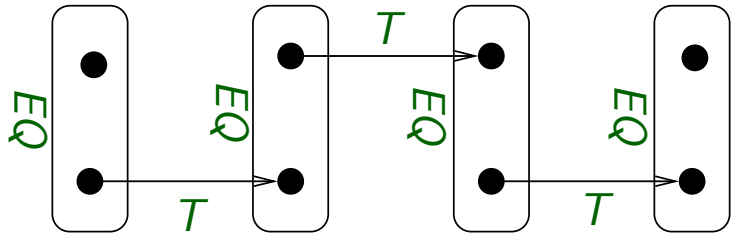

Transition system
Abstraction
Model Checking
No violations
Counter-example π
Refinement
Spurious
Simulation
Real bug

# Outline

# General idea

## Key idea

Encode the abstract path in terms of concrete variables.



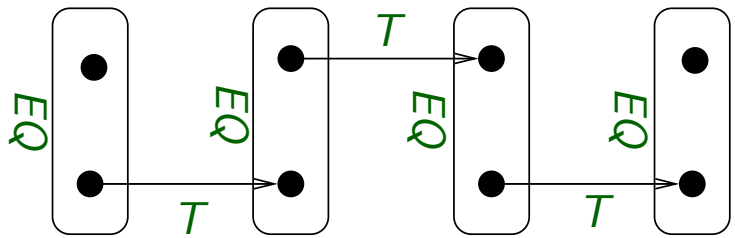

- An abstract path encoding represents a sequence of disconnected transitions where every gap lays in the same abstract state.
- Equivalence induced by abstraction:
  - $s, \overline{s} \models EQ_\alpha$ iff two concrete states correspond to the same abstract one.
  - In the case of predicate abstraction:

$$EQ_{\mathbb{P}}(V, \overline{V}) := \exists \hat{V}\left(\bigwedge_{P \in \mathbb{P}} \hat{v}_P \leftrightarrow P(V) \wedge \bigwedge_{P \in \mathbb{P}} \hat{v}_P \leftrightarrow P(\overline{V})\right)$$

$$\equiv \bigwedge_{P \in \mathbb{P}} P(V) \leftrightarrow P(\overline{V})$$

# Abstract encodings

- We use $EQ_\alpha$ to provide abstract versions of the formulas used for BMC and k-induction.
- Abstract path:
  $\widehat{PATH}_{\alpha,k} := \bigwedge_{1 \leq h < k}(T(V_{h-1}, \overline{V}_h) \wedge EQ_\alpha(\overline{V}_h, V_h)) \wedge T(V_{k-1}, \overline{V}_k)$
- The encoding represents a sequence of disconnected transitions where every gap lays in the same abstract state.



# Path encoding

## Rational

Embed the definition of the abstraction in the encoding of BMC and k-induction.

- Consider the BMC encoding of the abstract system:

$$I_\alpha(V_0) \wedge T_\alpha(V_0, V_1) \wedge \cdots \wedge T_\alpha(V_{k-1}, V_k) \wedge \phi_\alpha(V_k)$$

# Path encoding

## Rational

Embed the definition of the abstraction in the encoding of BMC and k-induction.

- Consider the BMC encoding of the abstract system:

$$\hat{I}_\alpha(\hat{V}_0) \wedge \hat{T}_\alpha(\hat{V}_0, \hat{V}_1) \wedge \cdots \wedge \hat{T}_\alpha(\hat{V}_{k-1}, \hat{V}_k) \wedge \hat{\phi}_\alpha(\hat{V}_k)$$

# Path encoding

## Rational

Embed the definition of the abstraction in the encoding of BMC and k-induction.

- Consider the BMC encoding of the abstract system:

$$\hat{I}_\alpha(\hat{V}_0) \wedge \hat{T}_\alpha(\hat{V}_0, \hat{V}_1) \wedge \cdots \wedge \hat{T}_\alpha(\hat{V}_{k-1}, \hat{V}_k) \wedge \hat{\phi}_\alpha(\hat{V}_k)$$

- If we substitute $\hat{I}_\alpha$, $\hat{T}_\alpha$, and $\hat{\phi}_\alpha$ with their definitions, we obtain:

## Rational

Embed the
k-induction.

## Abstract system

- $\hat{I}_\alpha(\hat{V}) := \exists V(I(V) \wedge H_\alpha(V, \hat{V}))$
- $\hat{T}_\alpha(\hat{V}, \hat{V}') := \exists V \exists V'(T(V, V') \wedge H_\alpha(V, \hat{V}) \wedge H_\alpha(V', \hat{V}'))$
- $\hat{F}_\alpha(\hat{V}) := \exists V(F(V) \wedge H_\alpha(V, \hat{V}))$
- $\hat{\phi}_\alpha = \exists V(\phi(V) \wedge H_\alpha(V, \hat{V}))$

- Consider the BMC encoding of the abstract system:

$$\hat{I}_\alpha(\hat{V}_0) \wedge \hat{T}_\alpha(\hat{V}_0, \hat{V}_1) \wedge \cdots \wedge \hat{T}_\alpha(\hat{V}_{k-1}, \hat{V}_k) \wedge \hat{\phi}_\alpha(\hat{V}_k)$$

- If we substitute $\hat{I}_\alpha$, $\hat{T}_\alpha$, and $\hat{\phi}_\alpha$ with their definitions, we obtain:

## Rational

Embed the
k-induction.

## Abstract system

- $\hat{I}_\alpha(\hat{V}) := \exists V(I(V) \wedge H_\alpha(V, \hat{V}))$
- $\hat{T}_\alpha(\hat{V}, \hat{V}') := \exists V \exists V'(T(V, V') \wedge H_\alpha(V, \hat{V}) \wedge H_\alpha(V', \hat{V}'))$
- $\hat{F}_\alpha(\hat{V}) := \exists V(F(V) \wedge H_\alpha(V, \hat{V}))$
- $\hat{\phi}_\alpha = \exists V(\phi(V) \wedge H_\alpha(V, \hat{V}))$

- Consider the BMC encoding of the abstract system:

$$\hat{I}_\alpha(\hat{V}_0) \wedge \hat{T}_\alpha(\hat{V}_0, \hat{V}_1) \wedge \cdots \wedge \hat{T}_\alpha(\hat{V}_{k-1}, \hat{V}_k) \wedge \hat{\phi}_\alpha(\hat{V}_k)$$

- If we substitute $\hat{I}_\alpha$, $\hat{T}_\alpha$, and $\hat{\phi}_\alpha$ with their definitions, we obtain:

$$I(V_0) \wedge H_\alpha(V_0, \hat{V}_0)$$

## Rational

Embed the c
k-induction.

## Abstract system

- $\hat{I}_\alpha(\hat{V}) := \exists V(I(V) \wedge H_\alpha(V, \hat{V}))$
- $\hat{T}_\alpha(\hat{V}, \hat{V}') := \exists V \exists V'(T(V, V') \wedge H_\alpha(V, \hat{V}) \wedge H_\alpha(V', \hat{V}'))$
- $\hat{F}_\alpha(\hat{V}) := \exists V(F(V) \wedge H_\alpha(V, \hat{V}))$
- $\hat{\phi}_\alpha = \exists V(\phi(V) \wedge H_\alpha(V, \hat{V}))$

- Consider the BMC encoding of the abstract system:

$$\hat{I}_\alpha(\hat{V}_0) \wedge \hat{T}_\alpha(\hat{V}_0, \hat{V}_1) \wedge \cdots \wedge \hat{T}_\alpha(\hat{V}_{k-1}, \hat{V}_k) \wedge \hat{\phi}_\alpha(\hat{V}_k)$$

- If we substitute $\hat{I}_\alpha$, $\hat{T}_\alpha$, and $\hat{\phi}_\alpha$ with their definitions, we obtain:

$$I(V_0) \wedge H_\alpha(V_0, \hat{V}_0) \wedge H_\alpha(\overline{V}_0, \hat{V}_0) \wedge T(\overline{V}_0, V_1) \wedge H_\alpha(V_1, \hat{V}_1)$$

## Rational

Embed the
k-induction.

## Abstract system

- $\hat{I}_\alpha(\hat{V}) := \exists V(I(V) \wedge H_\alpha(V, \hat{V}))$
- $\hat{T}_\alpha(\hat{V}, \hat{V}') := \exists V \exists V'(T(V, V') \wedge H_\alpha(V, \hat{V}) \wedge H_\alpha(V', \hat{V}'))$
- $\hat{F}_\alpha(\hat{V}) := \exists V(F(V) \wedge H_\alpha(V, \hat{V}))$
- $\hat{\phi}_\alpha = \exists V(\phi(V) \wedge H_\alpha(V, \hat{V}))$

- Consider the BMC encoding of the abstract system:

$$\hat{I}_\alpha(\hat{V}_0) \wedge \hat{T}_\alpha(\hat{V}_0, \hat{V}_1) \wedge \cdots \wedge \hat{T}_\alpha(\hat{V}_{k-1}, \hat{V}_k) \wedge \hat{\phi}_\alpha(\hat{V}_k)$$

- If we substitute $\hat{I}_\alpha$, $\hat{T}_\alpha$, and $\hat{\phi}_\alpha$ with their definitions, we obtain:

$$I(V_0) \wedge H_\alpha(V_0, \hat{V}_0) \wedge H_\alpha(\overline{V}_0, \hat{V}_0) \wedge T(\overline{V}_0, V_1) \wedge H_\alpha(V_1, \hat{V}_1) \wedge \cdots \wedge H_\alpha(\overline{V}_{k-1}, \hat{V}_{k-1}) \wedge T(\overline{V}_{k-1}, V_k) \wedge H_\alpha(V_k, \hat{V}_k)$$

## Rational

Embed the
k-induction.

## Abstract system

- $\hat{I}_\alpha(\hat{V}) := \exists V(I(V) \wedge H_\alpha(V, \hat{V}))$
- $\hat{T}_\alpha(\hat{V}, \hat{V}') := \exists V \exists V'(T(V, V') \wedge H_\alpha(V, \hat{V}) \wedge H_\alpha(V', \hat{V}'))$
- $\hat{F}_\alpha(\hat{V}) := \exists V(F(V) \wedge H_\alpha(V, \hat{V}))$
- $\hat{\phi}_\alpha = \exists V(\phi(V) \wedge H_\alpha(V, \hat{V}))$

- Consider the BMC encoding of the abstract system:

$$\hat{I}_\alpha(\hat{V}_0) \wedge \hat{T}_\alpha(\hat{V}_0, \hat{V}_1) \wedge \cdots \wedge \hat{T}_\alpha(\hat{V}_{k-1}, \hat{V}_k) \wedge \hat{\phi}_\alpha(\hat{V}_k)$$

- If we substitute $\hat{I}_\alpha$, $\hat{T}_\alpha$, and $\hat{\phi}_\alpha$ with their definitions, we obtain:

$$I(V_0) \wedge H_\alpha(V_0, \hat{V}_0) \wedge H_\alpha(\overline{V}_0, \hat{V}_0) \wedge T(\overline{V}_0, V_1) \wedge H_\alpha(V_1, \hat{V}_1) \wedge \cdots \wedge$$
$$H_\alpha(\overline{V}_{k-1}, \hat{V}_{k-1}) \wedge T(\overline{V}_{k-1}, V_k) \wedge H_\alpha(V_k, \hat{V}_k) \wedge H_\alpha(\overline{V}_k, \hat{V}_k) \wedge \phi(\overline{V}_k)$$

# Path encoding

## Rational

Embed the definition of the abstraction in the encoding of BMC and k-induction.

- Consider the BMC encoding of the abstract system:

$$\hat{I}_\alpha(\hat{V}_0) \wedge \hat{T}_\alpha(\hat{V}_0, \hat{V}_1) \wedge \cdots \wedge \hat{T}_\alpha(\hat{V}_{k-1}, \hat{V}_k) \wedge \hat{\phi}_\alpha(\hat{V}_k)$$

- If we substitute $\hat{I}_\alpha$, $\hat{T}_\alpha$, and $\hat{\phi}_\alpha$ with their definitions, we obtain:

$$I(V_0) \wedge H_\alpha(V_0, \hat{V}_0) \wedge H_\alpha(\overline{V}_0, \hat{V}_0) \wedge T(\overline{V}_0, V_1) \wedge H_\alpha(V_1, \hat{V}_1) \wedge \cdots \wedge$$
$$H_\alpha(\overline{V}_{k-1}, \hat{V}_{k-1}) \wedge T(\overline{V}_{k-1}, V_k) \wedge H_\alpha(V_k, \hat{V}_k) \wedge H_\alpha(\overline{V}_k, \hat{V}_k) \wedge \phi(\overline{V}_k)$$

- Note that the scope of abstract variables $\hat{V}_i$ is limited to two copies of the abstraction relation.

# Path encoding

## Rational

Embed the definition of the abstraction in the encoding of BMC and k-induction.

- Consider the BMC encoding of the abstract system:

$$\hat{I}_\alpha(\hat{V}_0) \wedge \hat{T}_\alpha(\hat{V}_0, \hat{V}_1) \wedge \cdots \wedge \hat{T}_\alpha(\hat{V}_{k-1}, \hat{V}_k) \wedge \hat{\phi}_\alpha(\hat{V}_k)$$

- If we substitute $\hat{I}_\alpha$, $\hat{T}_\alpha$, and $\hat{\phi}_\alpha$ with their definitions, we obtain:

$$I(V_0) \wedge EQ_\alpha(V_0, \overline{V}_0) \wedge T(\overline{V}_0, V_1) \wedge H_\alpha(V_1, \hat{V}_1) \wedge \cdots \wedge$$
$$H_\alpha(\overline{V}_{k-1}, \hat{V}_{k-1}) \wedge T(\overline{V}_{k-1}, V_k) \wedge EQ_\alpha(V_k, \overline{V}_k) \wedge \phi(\overline{V}_k)$$

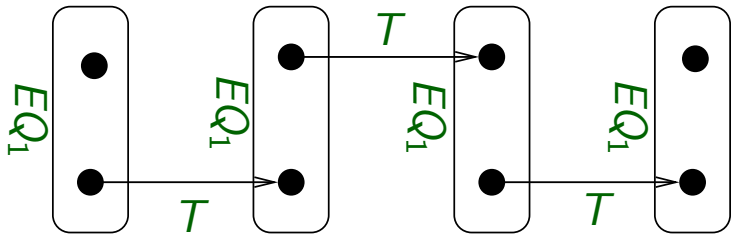
- Note that the scope of abstract variables $\hat{V}_i$ is limited to two copies of the abstraction relation.

# Abstract model checking

- Abstract BMC:
  $\widehat{BMC}_{\alpha,k,\phi} := I(V_0) \wedge EQ_\alpha(V_0, \overline{V}_0) \wedge \widehat{PATH}_{\alpha,k} \wedge EQ_\alpha(V_k, \overline{V}_k) \wedge \phi(V_k)$
- Abstract simple path:
  $\widehat{SIMPLEPATH}_{\alpha,k} := \widehat{PATH}_{\alpha,k} \wedge \bigwedge_{0 \leq i < j \leq k} \neg EQ_\alpha(V_i, V_j)$
- Abstract forward simple path:
  $\widehat{kindfw}_{\alpha,k} := I(V_0) \wedge EQ_\alpha(V_0, \overline{V}_0) \wedge \widehat{SIMPLEPATH}_{\alpha,k}$
- Abstract backward simple path:
  $\widehat{kindbw}_{\alpha,k,\phi} := \widehat{SIMPLEPATH}_{\alpha,k} \wedge EQ_\alpha(V_k, \overline{V}_k) \wedge \phi(V_k)$
- If, for all $i \leq k$, $\widehat{BMC}_{\alpha,i,\phi}$ is unsat and, either $\widehat{kindfw}_{\alpha,k+1}$ or $\widehat{kindbw}_{\alpha,k+1,\phi}$ is unsat as well, then $\hat{\phi}$ is not reachable in $\hat{S}$.

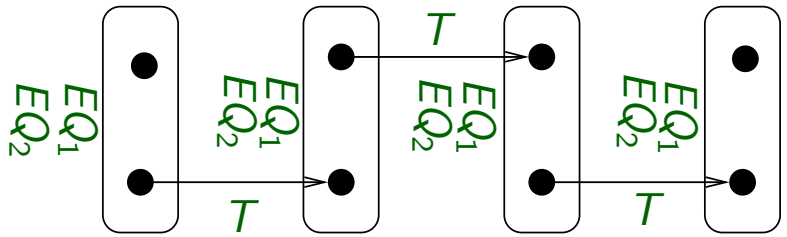
# Abstraction refinement

- Incremental abstraction: exploiting $EQ_{\mathbb{P}\cup\mathbb{P}'} = EQ_{\mathbb{P}} \wedge EQ_{\mathbb{P}'}$



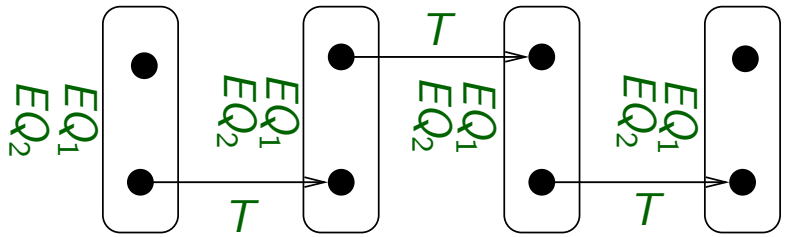
# Abstraction refinement

- Incremental abstraction: exploiting $EQ_{\mathbb{P}\cup\mathbb{P}'} = EQ_{\mathbb{P}} \wedge EQ_{\mathbb{P}'}$

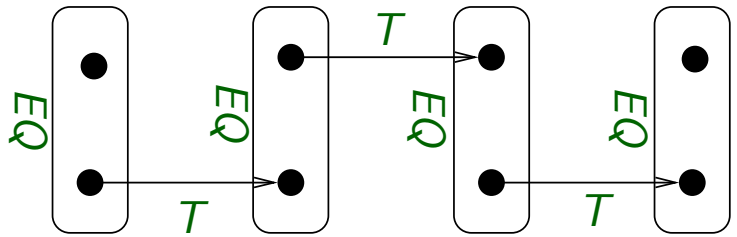


# Abstraction refinement

- Incremental abstraction: exploiting $EQ_{\mathbb{P}\cup\mathbb{P}'} = EQ_{\mathbb{P}} \wedge EQ_{\mathbb{P}'}$

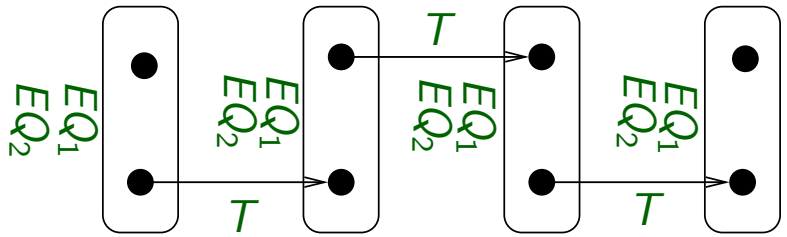



- Incremental concretization:

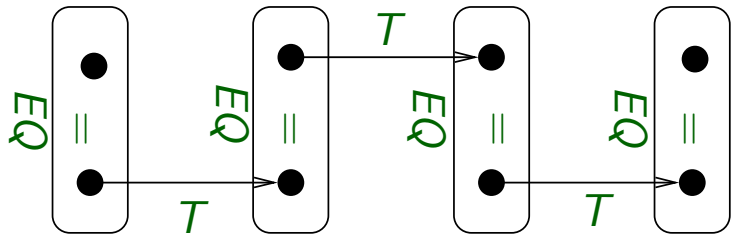


# Abstraction refinement

- Incremental abstraction: exploiting $EQ_{\mathbb{P}\cup\mathbb{P}'} = EQ_{\mathbb{P}} \wedge EQ_{\mathbb{P}'}$

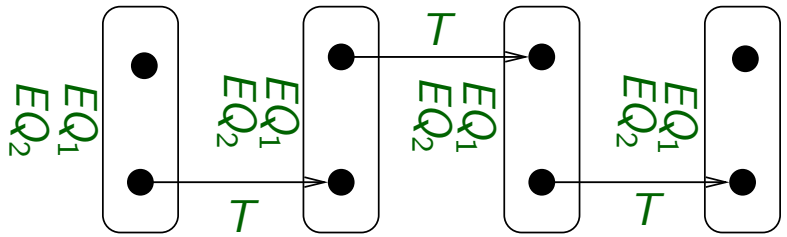



- Incremental concretization:

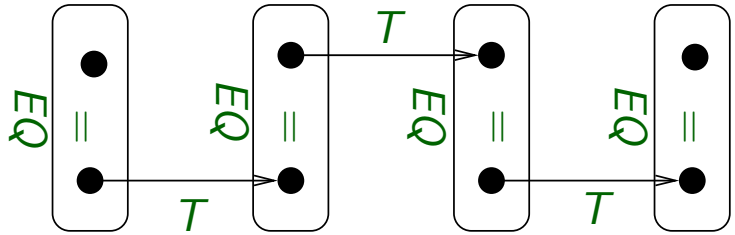


# Abstraction refinement

- Incremental abstraction: exploiting $EQ_{\mathbb{P}\cup\mathbb{P}'} = EQ_{\mathbb{P}} \wedge EQ_{\mathbb{P}'}$




- Incremental concretization:




- Predicate discovery based on interpolation.

# Outline

# Conclusions

- In this talk: BMC, k-induction, interpolation, predicate abstraction and combination thereof.
- At FBK, we work both on VMT engine and applications:
  - hybrid verification (Mover's thesis),
  - software verification (Kratos model checker),
  - requirements validation and safety assessment for expressive logic.
- Better to avoid quantifier elimination $\rightsquigarrow$ new VMT techniques.
- Other SAT-based techniques not described: interpolation-sequence and IC3.