Advanced model checking for verification and safety assessment

Part 1

Stefano Tonetta

Fondazione Bruno Kessler (FBK)

Sixth Summer School on Formal Techniques (SSFT’16)

Lecture prepared in collaboration with

Alessandro Cimatti and Marco Gario

Slides on IC3 borrowed from Alberto Griggio (VTSA’15)
LECTURE 1

• Motivation

• Finite-State Model Checking
  • Invariant Checking
    • IC3
  • LTL Checking

• Infinite-State Model Checking

• Wrap-up
Motivation
Embedded Safety-Critical Systems

- Embedded with software to deliver intelligent:
  - Transportation
  - Communication
  - Automation

- Across domains:
  - Railways
  - Avionics
  - Automotive
  - Space
  - Health

- Key properties and challenges:
  - Interaction of components
  - Decomposition of services
  - Safety requirements

Stefano Tonetta, SSFT'16
Model-based system engineering

• **Models** used for system requirements, architectural design, analysis, validation and verification

• Different system-level analysis (safety, reliability, performance, ...)

• Formal methods as back-end
  • **Formal specification** to assign models a rigorous mathematical semantics
  • **Formal verification** to prove the properties on the models.

• Design models translated into input for verification engine

• Requirements formalized into properties

• Model checking appealing because integrated as push-button
AIR6110 Wheel Braking System

• Joint scientific study with Boeing

• Aerospace Information Report 6110:
  • Traditional Aircraft/System Development Process Example

• Wheel Brake System of a fictional dual-engine aircraft

• Objectives:
  • Analyze the system safety through formal techniques
  • Demonstrate the usefulness and suitability of formal techniques for improving the overall traditional development and supporting aircraft certification
NASA NextGen Air Traffic Control

- Joint project with NASA Ames and Langley
- Allocation of tasks between Aircraft and Ground
  - Model and Study a design space with more than 1600 configurations
- Objectives:
  - Apply Formal Methods to study the quality and Safety of many design proposals
  - Highlight Implicit assumptions
Finite-State Model Checking

Invariant Checking
Model checking

temporal formula

G(p \rightarrow Fq)

finite-state model

Model Checker

yes!

no!

counterexample
Mutual exclusion example

N: non-critical
T: trying
C: critical
User1
User2

Property: always not C1 or not C2 i.e. (C1 and C2) is not reachable
Symbolic representation

- Symbolic Boolean variables $V = \{v_1, \ldots, v_n\}$ to represent the state space
- A state is an assignment to the variables
- Symbolic formulas used to represent:
  - Set of states: $\phi(V) \equiv \{s \mid s \models \phi\}$
    - Abuse of notation $s \in \phi$ iff $s \models \phi$
  - Set of transitions: $\phi(V,V') \equiv \{\langle s, s' \rangle \mid \langle s, s' \rangle \models \phi\}$
    - Where the variables $V' = \{v'_1, \ldots, v'_n\}$ represent next state variables
- A transition system is a tuple $\langle V, I, T \rangle$ where:
  - $V$ is the set of variables
  - The set of initial states represented by the formula $I(V)$
  - The transition relation represented by the formula $T(V,V')$
Example

- $V := \{ u, v \}$
- $I := \neg u \land \neg v$
- $T := u' \leftrightarrow \neg u \land v' \leftrightarrow (u \text{ xor } v)$
Invariant properties

- A path of the system $S$ is a sequence $s_0, s_1, ..., s_k$ of states such that $s_0 \models I$ and for all $i, 0 \leq i < k$, $s_i, s_{i+1} \models T$

- A state $s$ is reachable iff there exists a path $s_0, s_1, ..., s_k$ such that $s = s_k$

- A formula $P(V)$ is an invariant iff for all paths $s_0, s_1, ..., s_k$, for all $i, s_i \models P$

- Equivalent to say that no state in $\neg P$ is reachable
Forward reachability checking

- **Forward image computation:**
  - Compute all states reachable from $Q$ in one transition:
    \[ \text{FwdImg}(Q) := \exists V(Q(V) \land T(V, V')[V/V')] \]

- **Prove that a set of states $Bad$ is not reachable:**
  - Start from initial states: $R := I$
  - Apply $FwdImg$ iteratively: $oldR := R; R := FwdImg(R)$
  - until fixpoint $oldR = R$

\[
R(X) := I(X) \quad \text{Bad}(X)
\]
Bounded Model Checking

- Reachability encoded into a satisfiability problem
  \[ I(V_0) \land T(V_0, V_1) \land T(V_1, V_2) \land \ldots \land T(V_{k-1}, V_k) \land \text{Bad}(V_k) \]
- The formula is sat iff there exists a path of length \( k \) that reaches \( \text{Bad} \)
- Checked for increasing values of \( k \)
- Exploited incrementality of SAT solvers
- Finite-state space \( \Rightarrow \) a completeness threshold \( K \) exists
  - If unsat for all \( k \leq K \) then \( \text{Bad} \) is not reachable
  - \( K \) is typically very large \( \Rightarrow \) unfeasible to reach in practice
Example

- $V := \{u, v \}$
- $I := \neg u \land \neg v$
- $T := u' \leftrightarrow u \land v' \leftrightarrow (u \text{ xor } v)$
- $Bad := u \land v$
- **BMC:**
  - $(\neg u_0 \land \neg v_0) \land (u_0 \land v_0)$ UNSAT
  - $(\neg u_0 \land \neg v_0) \land (u_1 \leftrightarrow u_0 \land v_1 \leftrightarrow (u_0 \text{ xor } v_0)) \land (u_1 \land v_1)$ UNSAT
  - ...
  - $(\neg u_0 \land \neg v_0) \land (u_1 \leftrightarrow u_0 \land v_1 \leftrightarrow (u_0 \text{ xor } v_0))$
    $(u_2 \leftrightarrow u_1 \land v_2 \leftrightarrow (u_1 \text{ xor } v_1)) \land$
    $(u_3 \leftrightarrow u_2 \land v_3 \leftrightarrow (u_2 \text{ xor } v_2)) \land (u_3 \land v_3)$ SAT
Induction and K-induction

• **Induction**
  • Base case: check if the initial state satisfies $P$ (invariant)
  • Inductive case: check if the transitions preserve the invariant $P(V) \land T(V, V') \models P(V')$
  • We say $P$ is inductive invariant

• **K-induction**
  • Base case: check if all initial path satisfies $P$ (invariant) up to $k$ steps
  • Inductive case: check if every path of $k + 1$ steps preserve the invariant
    
    
    
    $P(V_0) \land T(V_1, V_2) \land P(V_1) \land T(V_1, V_2) \land \cdots \land P(V_{k-1}) \land T(V_{k-1}, V_k) \models P(V')$
  • Strengthened with simple path condition to avoid repeating states
  • We say $P$ is $k$-inductive invariant

• **Typically however $P$ is not ($k$-)inductive**
  
  $\Rightarrow$ find $Inv$ such that $Inv$ is inductive invariant and $Inv \models P$
Example

- \( V := \{x_1, x_2, x_3\} \)
- \( I := \neg x_1 \land \neg x_2 \land \neg x_3 \)
- \( Bad := x_1 \land x_2 \)
- \( P := \neg x_1 \lor \neg x_2 \)
- Inductive?
  - No
- \( k\)-inductive?
  - Yes for \( k=3 \)
- Inductive invariant?
Finite State Model-Checking

IC3
IC3

- Very successful SAT-based model checking algorithm
- Based on induction
  - Given a symbolic transition system and invariant property $P$, build an inductive invariant $F$ s.t. $F \models P$
- Inductive invariant built incrementally
  - Trace of formulas $F_0 \equiv I, F_1, \ldots, F_k$ s.t:
    - for $i > 0$, $F_i$ is a set of clauses, overapproximation of states reachable in up to $i$ steps
    - $F_{i+1} \subseteq F_i$ (so $F_i \equiv F_{i+1}$)
    - $F_i \land T \models F'_{i+1}$
    - For all $i < k$, $F_i \models P$
- Strengthen formulas until $F_k = F_{k+1}$
- Exploiting efficient SAT solvers
A (very) high level view of IC3

- **Blocking phase**: incrementally strengthen trace until $F_k \models P$
  - Get bad cube $s$
A (very) high level view of IC3

- **Blocking phase**: incrementally strengthen trace until $F_k \models P$
  - Get bad cube $s$
  - Call SAT solver on $F_{k-1} \land \neg s \land T \land s'$
    (i.e., check if $F_{k-1} \land \neg s \land T \models \neg s'$)

Check if $\neg s$ is inductive relative to $F_{k-1}$
A (very) high level view of IC3

- **Blocking phase**: incrementally strengthen trace until $F_k \models P$
  - Get bad cube $s$
  - Call SAT solver on $F_{k-1} \land \neg s \land T \land s'$
    - **SAT**: $s$ is reachable from $F_{k-1}$ in 1 step
    - Get a cube $c$ in the preimage of $s$ and try (recursively) to prove it unreachable from $F_{k-2}, ...$
    - $c$ is a counterexample to induction (CTI)

If $I$ is reached, a counterexample to $P$ is found
A (very) high level view of IC3

- **Blocking phase**: incrementally strengthen trace until $F_k \models P$
  - Get bad cube $s$
  - Call SAT solver on $F_{k-1} \land \neg s \land T \land s'$
    - **UNSAT**: $\neg s$ is inductive relative to $F_{k-2}$
    - Generalize $c$ to $g$ and block by adding $\neg g$ to $F_{i-1}, F_{i-2}, \ldots, F_1$
A (very) high level view of IC3

- **Blocking phase**: incrementally strengthen trace until $F_k \models P$
  - Get bad cube $s$
  - Call SAT solver on $F_{k-1} \land \neg s \land T \land s'$
    - **UNSAT**: $\neg s$ is inductive relative to $F_{k-2}$
    - Generalize $c$ to $g$ and block by adding $\neg g$ to $F_{i-1}, F_{i-2}, \ldots, F_1$
A (very) high level view of IC3

- **Propagation**: extend trace to $F_{k+1}$ and push forward clauses
  - For each $i$ and each clause $c \in F_i$:
    - Call SAT solver on $F_i \land T \land \neg c'$
    - If UNSAT, add $c$ to $F_{i+1}$
A (very) high level view of IC3

- **Propagation**: extend trace to $F_{k+1}$ and push forward clauses
  - For each $i$ and each clause $c \in F_i$:
    - Call SAT solver on $F_i \land T \land \neg c'$
    - If **UNSAT**, add $c$ to $F_{i+1}$
A (very) high level view of IC3

- **Propagation**: extend trace to $F_{k+1}$ and push forward clauses
  - For each $i$ and each clause $c \in F_i$:
    - Call SAT solver on $F_i \land T \land \neg c'$
      - If **UNSAT**, add $c$ to $F_{i+1}$
    - If $F_i \equiv F_{i+1}$, P is proved,
    - otherwise start another round of blocking and propagation
Inductive Clause Generalization

- Crucial step of IC3
- Given a relatively inductive clause \( c \overset{\text{def}}{=} \{l_1, \ldots, l_n\} \)
- Compute a generalization \( g \subseteq c \) that is still inductive

\[
F_{i-1} \land T \land g \models g' \tag{1}
\]

- Drop literals from \( c \) and check that (1) still holds
  - Accelerate with unsat cores returned by the SAT solver
    - Using SAT under assumptions
- However, make sure the base case still holds
  - If \( I \not\models c \setminus \{l_j\} \), then \( l_j \) cannot be dropped

Stefano Tonetta, SSFT'16
Example

No counterexamples of length 0

\[ I = \neg x_1 \land \neg x_2 \land \neg x_3 \]
\[ P = \neg x_1 \lor \neg x_2 \]

\[ F_0 = I \]
\[ F_1 = \top \]

[borrowed and adapted from F. Somenzi]
Example

Get bad cube $c = x_1 \land x_2$ in $F_1 \land \neg P$

$I = \neg x_1 \land \neg x_2 \land \neg x_3$
$P = \neg x_1 \lor \neg x_2$

$F_0 = I$
$F_1 = \top$
Example

Is $\neg c$ inductive relative to $F_0$? $F_0 \land T \land \neg c \models \neg c'$

$I = \neg x_1 \land \neg x_2 \land \neg x_3$

$P = \neg x_1 \lor \neg x_2$

$F_0 = I$

$F_1 = \top$
Example

Yes, generalize $\neg c = \neg x_1 \lor \neg x_2$

$I = \neg x_1 \land \neg x_2 \land \neg x_3$

$P = \neg x_1 \lor \neg x_2$

$F_0 = I$

$F_1 = \top$
Yes, generalize $\neg c = \neg x_1 \lor \neg x_2$

Try dropping $\neg x_2$

$F_0 \land T \land \neg x_1 \not\vdash \neg x'_1$

$I = \neg x_1 \land \neg x_2 \land \neg x_3$

$P = \neg x_1 \lor \neg x_2$

$F_0 = I$

$F_1 = \top$
Example

Yes, generalize \( \neg c = \neg x_1 \lor \neg x_2 \)

Try dropping \( \neg x_1 \)

\[
F_0 \land T \land \neg x_2 \models \neg x'_2
\]

\[
I = \neg x_1 \land \neg x_2 \land \neg x_3
\]

\[
P = \neg x_1 \lor \neg x_2
\]

\[
F_0 = I
\]

\[
F_1 = \top
\]
Example

Yes, generalize $\neg c = \neg x_1 \lor \neg x_2$

Try dropping $\neg x_1$

$F_0 \land T \land \neg x_2 \models \neg x'_2$

$I = \neg x_1 \land \neg x_2 \land \neg x_3$
$P = \neg x_1 \lor \neg x_2$

$F_0 = I$
$F_1 = \top$
Example

Update $F_1$

$I = \neg x_1 \land \neg x_2 \land \neg x_3$
$P = \neg x_1 \lor \neg x_2$

$F_0 = I$
$F_1 = \neg x_2$
Example

Blocking done for $F_1$. Add $F_2$ and propagate forward

\[ I = \neg x_1 \land \neg x_2 \land \neg x_3 \]
\[ P = \neg x_1 \lor \neg x_2 \]

\[ F_0 = I \]
\[ F_1 = \neg x_2 \]
\[ F_2 = \top \]
Example

No clause propagates from $F_1$ to $F_2$

$I = \neg x_1 \land \neg x_2 \land \neg x_3$

$P = \neg x_1 \lor \neg x_2$

$F_0 = I$

$F_1 = \neg x_2$

$F_2 = \top$
Example

Get bad cube $c = x_1 \land x_2$ in $F_2 \land \neg P$

$I = \neg x_1 \land \neg x_2 \land \neg x_3$
$P = \neg x_1 \lor \neg x_2$

$F_0 = I$
$F_1 = \neg x_2$
$F_2 = \top$
Example

Is $\neg c$ inductive relative to $F_1$? $F_1 \land T \land \neg c \models \neg c'$

$I = \neg x_1 \land \neg x_2 \land \neg x_3$

$P = \neg x_1 \lor \neg x_2$

$F_0 = I$

$F_1 = \neg x_2$

$F_2 = \top$
Example

No, found CTI $s = \neg x_1 \land \neg x_2 \land x_3$

$I = \neg x_1 \land \neg x_2 \land \neg x_3$
$P = \neg x_1 \lor \neg x_2$

$F_0 = I$
$F_1 = \neg x_2$
$F_2 = \top$
Try blocking $\lnot s$ at level $0$: $F_0 \land T \land \lnot s \models \lnot s'$

$I = \lnot x_1 \land \lnot x_2 \land \lnot x_3$

$P = \lnot x_1 \lor \lnot x_2$

$F_0 = I$

$F_1 = \lnot x_2$

$F_2 = \top$
Example

Yes, generalize \( \neg s = x_1 \lor x_2 \lor \neg x_3 \)

Try dropping \( x_1 \)

\[
F_0 \land T \land x_2 \lor \neg x_3 \not\models x'_2 \lor \neg x'_3
\]

\[
I = \neg x_1 \land \neg x_2 \land \neg x_3 \\
P = \neg x_1 \lor \neg x_2 \\
F_0 = I \\
F_1 = \neg x_2 \\
F_2 = \top
\]
Example

Yes, generalize $\neg s = x_1 \lor x_2 \lor \neg x_3$

Try dropping $x_2$

$$F_0 \land T \land x_1 \lor \neg x_3 \models x_1' \lor \neg x_3'$$

$I = \neg x_1 \land \neg x_2 \land \neg x_3$

$P = \neg x_1 \lor \neg x_2$

$F_0 = I$

$F_1 = \neg x_2$

$F_2 = \top$
Example

Yes, generalize $\neg s = x_1 \vee x_2 \vee \neg x_3$

![Diagram]

Try dropping $x_3$

$I \not\models x_1$

$I = \neg x_1 \wedge \neg x_2 \wedge \neg x_3$

$P = \neg x_1 \vee \neg x_2$

$F_0 = I$

$F_1 = \neg x_2$

$F_2 = \top$
Example

Update $F_1$

$I = \neg x_1 \land \neg x_2 \land \neg x_3$

$P = \neg x_1 \lor \neg x_2$

$F_0 = I$

$F_1 = \neg x_2 \land (x_1 \lor \neg x_3)$

$F_2 = \top$
Example

Return to the original bad cube $c$

$I = \neg x_1 \land \neg x_2 \land \neg x_3$

$P = \neg x_1 \lor \neg x_2$

$F_0 = I$

$F_1 = \neg x_2 \land (x_1 \lor \neg x_3)$

$F_2 = \top$
Example

Is $\neg c$ inductive relative to $F_1$?  

\[ F_1 \land T \land \neg c \models \neg c' \]

\[ I = \neg x_1 \land \neg x_2 \land \neg x_3 \]
\[ P = \neg x_1 \lor \neg x_2 \]

\[ F_0 = I \]
\[ F_1 = \neg x_2 \land (x_1 \lor \neg x_3) \]
\[ F_2 = \top \]
Example

Yes, generalize $\neg c = \neg x_1 \lor \neg x_2$

Try dropping $\neg x_1$

$$F_1 \land T \land \neg x_2 \models \neg x'_2$$

$$I = \neg x_1 \land \neg x_2 \land \neg x_3$$
$$P = \neg x_1 \lor \neg x_2$$

$$F_0 = I$$
$$F_1 = \neg x_2 \land (x_1 \lor \neg x_3)$$
$$F_2 = \top$$
Example

Update $F_2$ and add new frame $F_3$

$I = \neg x_1 \land \neg x_2 \land \neg x_3$
$P = \neg x_1 \lor \neg x_2$

$F_0 = I$
$F_1 = \neg x_2 \land (x_1 \lor \neg x_3)$
$F_2 = \neg x_2$
$F_3 = \top$
Perform forward propagation

\[ F_1 \land T \models (x'_1 \lor \neg x'_3) \]

\[ I = \neg x_1 \land \neg x_2 \land \neg x_3 \]
\[ P = \neg x_1 \lor \neg x_2 \]

\[ F_0 = I \]
\[ F_1 = \neg x_2 \land (x_1 \lor \neg x_3) \]
\[ F_2 = \neg x_2 \]
\[ F_3 = \top \]
Example

Perform forward propagation

Found fixpoint!

\[ I = \neg x_1 \land \neg x_2 \land \neg x_3 \]
\[ P = \neg x_1 \lor \neg x_2 \]

\[ F_0 = I \]
\[ F_1 = \neg x_2 \land (x_1 \lor \neg x_3) \]
\[ F_2 = \neg x_2 \land (x_1 \lor \neg x_3) \]
\[ F_3 = \top \]
Example

Perform forward propagation

Inductive invariant:

\[ F_1 \equiv F_2 \equiv \neg x_2 \land (x_1 \lor \neg x_3) \]

\[ I = \neg x_1 \land \neg x_2 \land \neg x_3 \]

\[ P = \neg x_1 \lor \neg x_2 \]

\[ F_0 = I \]

\[ F_1 = \neg x_2 \land (x_1 \lor \neg x_3) \]

\[ F_2 = \neg x_2 \land (x_1 \lor \neg x_3) \]

\[ F_3 = \top \]
Finite State Model-Checking

Liveness Checking
Linear Temporal Logic

• Linear models: state sequences (traces)
• Built over set of atomic propositions $AP$
• LTL is the smallest set of formulas such that:
  • any atomic proposition $p \in AP$ is an LTL formula
  • if $\phi_1$ and $\phi_2$ are LTL formulas, then $\neg \phi_1, \phi_1 \land \phi_2$ and $\phi_1 \lor \phi_2$ are LTL formulas
  • if $\phi_1$ and $\phi_2$ are LTL formulas, then $X\phi_1, F\phi_1, G\phi_1$ and $\phi_1 U \phi_2$ are LTL formulas
LTL semantics

Semantics defined for every trace, for every $i \in \mathbb{N}$.

- Given an infinite trace $\pi = s_0, s_1, \ldots$

  - $\pi, i \models p$ iff $s_i \models p$
  - Standard definition for $\neg, \land, \lor$
  - $\pi, i \models X \phi$ iff $s_{i+1}, s_{i+2}, \ldots \models \phi$
  - $\pi, i \models \phi_1 U \phi_2$ iff there exists $j \geq i$, $\pi, j \models \phi_2$ and for all $k, i \leq k < j$, $\pi, k \models \phi_1$
  - $\pi, i \models F \phi$ iff there exists $j \geq i$, $\pi, j \models \phi$
  - $\pi, i \models G \phi$ iff for all $j \geq i$, $\pi, j \models \phi$
  - $M \models \phi$ iff $M, \pi, 0 \models \phi$ for every trace $\pi$ of $M$. 

Stefano Tonetta, SSFT'16
LTL examples

- \(Gp\) “always p” – like invariant (if we assume deadlock freedom)
- \(G(p \rightarrow Fq)\) “p is always followed by q” - reaction
- \(G(p \rightarrow Xq)\) “whenever p holds, q is set to true” – immediate reaction
- \(GFp\) “infinitely many times p” – fairness
- \(FGp\) “eventually permanently p”
- \(G(speed_{above\_limit} \rightarrow (brake \mathcal{U} \neg speed_{above\_limit}))\)
LTL verification

• Given an LTL property $\phi$, build a transition system $M_{\neg \phi}$ with a fairness condition $f_{\neg \phi}$, such that

$$M \times M_{\neg \phi} \models FG \neg f_{\neg \phi}$$

• $FG$ requires a doubly-nested fixpoint

• SAT-based approaches typically reduce the problem to safety
Liveness2safety

• Based on the existence of a lasso-shaped counterexample, with $f_{\neg \phi}$ at least once in the loop

• liveness to safety transformation: absence of lasso-shaped counterexamples as an invariant property
  • Duplicate the state variables $V_{copy} := \{v_c | v \in V\}$
  • Non-deterministically save the current state
  • Remember when $f_{\neg \phi}$ in extra state var $triggered$
  • Invariant: $G_{\neg}(X = X_{copy} \land triggered)$
K-liveness

- Simple but effective technique for LTL verification of finite-state systems
- Key insight: $M \times M_{\neg \phi} \models FG \neg f_{\neg \phi}$ iff there exists $k$ such that $f_{\neg \phi}$ is visited at most $k$ times
  - Again, a safety property
- K-liveness: increase $k$ incrementally
  - Liveness checking as a sequence of safety checks
- Using IC3 as safety checker
  - Exploits the highly incremental nature of IC3
Infinite State Model-Checking
Infinite State Transition System

- Same definition as before: \( \langle V, I, T \rangle \)
- First-order instead of propositional formulas:
  - Signature: set \( \Sigma \) of constant, functional, and relational symbols
  - Structure: a domain \( D \) and interpretation \( I \) of the symbols in the signature
  - Theory: set \( \mathcal{T} \) of axioms (a model of \( \mathcal{T} \) is a structure that satisfy \( \mathcal{T} \))
- Some constant symbols are used as the variables of the transition system
  - They have a flexible interpretation that varies along time
  - The other symbols are rigid
- In the following \( \models \) implicitly means \( \models_{\mathcal{T}} \), i.e. is restricted to the models of a given theory
Example

- \( V := \{x, y\} \)
- \( I := y \leq x \)
- \( T := (x' = x + 1) \land (y' \leq y) \)
- \( \Sigma := \{x, y, 0, 1, +, \leq, \ldots\} \)
- \( \mathcal{T} := \text{theory of reals} \)
- \( y \leq x \land T \models_{\mathcal{T}} y' \leq x' \)
From SAT to SMT

- Previous algorithms assume to have a solver for the satisfiability of formulas
- First developed for finite-state systems with the support of SAT solvers
- SAT solvers substituted by Satisfiability Modulo Theory (SMT) solvers:
  - Satisfiability for decidable fragments of first-order logic
  - SAT solver used to enumerate Boolean models
  - Integrated with decision procedure for specific theories, e.g., theory of real linear arithmetic
- Search algorithms applied to infinite-state systems (although in general undecidable)
- Lift to SMT straightforward for BMC and k-induction
- Not for IC3:
  - Requires alternative effective generalization
Predicate Abstraction

- Reduction to finite-state MC
- Predicates \( \mathbb{P} \) over concrete variables to define the abstraction
- Abstract state space given by Boolean variables, one for each predicate \( \hat{V} = \{v_p \mid p \in \mathbb{P}\} \)
- Abstract state \( \alpha(s) = \{v_p \mid s(p) = T\} \)
- Abstract transition iff there exists a concrete transition between two corresponding concrete states
  \[ \hat{T} = \{\langle \hat{s}, \hat{s}' \rangle \mid \exists s, s', \alpha(s) = \hat{s}, \alpha(s') = \hat{s'}, T(s, s')\} \]
- Transitions computed with ALLSMT:
  \[ \hat{T}(\hat{V}, \hat{V}') = \exists V, V'(T(V, V') \land \bigwedge_{p \in \mathbb{P}} v_p \leftrightarrow p(V) \land \bigwedge_{p \in \mathbb{P}} v'_p \leftrightarrow p(V')) \]
Abstraction Refinement

- Abstract traces are overapproximations
  - Spurious counterexamples can be generated
- Standard abstraction refinement techniques based on interpolation
  - Sequence of abstract states $\hat{s}_0, \hat{s}_1, \ldots, \hat{s}_k$
  - SMT check on $\hat{s}_0(V_0) \land T(V_0, V_1) \land \hat{s}_1(V_1) \land T(V_1, V_2) \land \ldots \land T(V_{k-1}, V_k) \land \hat{s}_k(V_k)$
  - If unsat, compute sequence of interpolants for
    \[
    [\hat{s}_0(V_0) \land T(V_0, V_1) \land \ldots \land T(V_{i-1}, V_i)] \\
    [\hat{s}_i(V_i) \land T(V_0, V_1) \land \ldots \land T(V_{k-1}, V_k) \land \hat{s}_k(V_k)]
    \]
using the same UNSAT proof (called sequence interpolants)
  - Add all the predicates in the interpolants to $\mathcal{P}$
Implicit Predicate Abstraction

- Abstract version of BMC and k-induction, avoiding explicit computation of the abstract transition relation
  - By embedding the abstraction in the SMT encoding
  - \( EQ(V_1, V_2) := \land_{p \in \mathbb{P}} p(V_1) \leftrightarrow p(V_2) \)
- The abstract unrolling is
  \[
  T(V_0, \overline{V}_1) \land EQ(\overline{V}_1, V_1) \land T(V_1, \overline{V}_2) \land EQ(\overline{V}_2, V_2) \land T(V_2, V_3) \land \ldots
  \]
Infinite State Model-Checking

IC3 with Implicit Abstraction
IC3 with Implicit Abstraction

- Integrate the idea of Implicit Abstraction within IC3
- Use abstract transition relation
- Learn clauses only over predicates
- Use abstract relative induction check:

\[
\text{AbsRelInd}(F, T, c, \mathbb{P}) \equiv F(V) \land c(V) \land T(V, \overline{V}') \land \bigwedge_{p \in \mathbb{P}} \left( p(V') \leftrightarrow p\left(\overline{V}'\right) \right) \land \neg c(V')
\]

- If UNSAT \( \Rightarrow \) inductive strengthening as in the Boolean case
- No theory-specific technique needed
IC3 with Implicit Abstraction

- Integrate the idea of Implicit Abstraction within IC3
- Use abstract transition relation
- Learn clauses only over predicates
- Use abstract relative induction check:

  \[ \text{AbsRelInd}(F, T, c, \mathcal{P}) \]
  \[ := F(V) \land c(V) \land T(V, \bar{V}) \land \bigwedge_{p \in \mathcal{P}} (p(V') \leftrightarrow p(\bar{V})) \land \neg c(V') \]

- If SAT ⇔ abstract predecessor from the SMT model
- No preimage needed
Example

- $T := (2x_1' - 3x_1 \leq 4x_2' + 2x_2 + 3) \land (3x_1 - 2x_2' = 0)$
- $\mathbb{P} := \{(x_1 - x_2 \geq 4), (x_1 < 3)\}$
- $s := \neg(x_1 - x_2 \geq 4) \land (x_1 < 3)$
- $\text{AbsRelInd}(\emptyset, T, \neg s, \mathbb{P}) = T(V, V') \land \neg s(V) \land s(V') \land (x_1 - x_2 \geq 4) \leftrightarrow (\overline{x}_1 - \overline{x}_2 \geq 4) \land (x_1 < 3) \leftrightarrow (\overline{x}_1 < 3)$
- $\text{AbsRelInd}(\emptyset, T, s, \mathbb{P})$ is SAT
- Compute a predecessor from SMT model:
  \[ \mu \overset{\text{def}}{=} \{x_1 \mapsto 0, x_2 \mapsto 1\} \]
  \[ \neg(x_1 - x_2 \geq 4) \land (x_1 < 3) \]
Abstraction refinement

- Abstract counterexample check can use incremental SMT
- Abstraction refinement is *fully incremental*
- No restart from scratch
- Can keep all the clauses of $F_1, \ldots, F_k$
- Refinements monotonically strengthen $T$

\[
T_{\text{new}} := T_{\text{old}} \land \bigwedge_{p \in \text{new} \mathbb{P}} \left( p(V) \leftrightarrow p(W) \right) \land \left( p(V') \leftrightarrow p(W') \right)
\]

- All IC3 invariants on $F_1, \ldots, F_k$ are preserved
  - $F_{i+1} \subseteq F_i$ (so $F_i \models F_{i+1}$)
  - $F_i \land T \models F_{i+1}'$
  - For all $i < k, F_i \models P$
Example

- System with 2 state vars c and d
  - Init: \( (d = 1) \land (c \geq d) \)
  - Trans: \( (c' = c + d) \land (d' = d + 1) \)
  - Property: \( (d > 2) \rightarrow (c > d) \)
- Check base case: Init \( \models \) Property

- Predicates \( \mathbb{P} \)
  
  \( (d = 1), (c \geq d), (d > 2), (c > d) \)
Example

- System with 2 state vars $c$ and $d$
  - Init: $(d = 1) \land (c \geq d)$
  - Trans: $(c' = c + d) \land (d' = d + 1)$
  - Property: $(d > 2) \rightarrow (c > d)$

- Get bad cube
  - SMT check $F_1 \land \neg P$
  - SAT with model $\mu := \{c = 0, d = 3\}$
  - Evaluate predicates wrt. $\mu$
    - Return
      $s := \{\neg(d = 1), \neg(c \geq d), (d > 2), \neg(c > d)\}$

- Predicates $\mathbb{P}$
  - $(d = 1), (c \geq d), (d > 2), (c > d)$

- Trace
  - $F_0 := \text{Init}$
  - $F_1 := \top$
Example

- System with 2 state vars $c$ and $d$
  - Init: $(d = 1) \land (c \geq d)$
  - Trans: $(c' = c + d) \land (d' = d + 1)$
  - Property: $(d > 2) \rightarrow (c > d)$

- Rec. block $s$
  - Check

$AbsRelInd(F_0, T, \neg s, \mathcal{P})$

$:= Init \land (\overline{c} = c + d) \land (\overline{d} = d + 1)$
$\land (d' = 1 \leftrightarrow \overline{d} = 1) \land (c' \geq d' \leftrightarrow \overline{c} \geq \overline{d})$
$\land (d' > 2 \leftrightarrow \overline{d} > 2) \land (c' > d' \leftrightarrow \overline{c} > \overline{d}) \land \neg s$
$\land s'$

- Predicates $\mathcal{P}$
  $(d = 1), (c \geq d), (d > 2), (c > d)$

- Trace
  - $F_0 := Init$
  - $F_1 := T$
Example

- System with 2 state vars $c$ and $d$
  - Init: $(d = 1) \land (c \geq d)$
  - Trans: $(c' = c + d) \land (d' = d + 1)$
  - Property: $(d > 2) \rightarrow (c > d)$

- Rec. block $s$
  - Check $\text{AbsRelInd}(F_0, T, \neg s, P)$: UNSAT
  - Generalize: $\{\neg(d > 2)\}$
  - Update $F_1 := F_1 \land \neg(d > 2)$

- Predicates $P$
  - $(d = 1), (c \geq d), (d > 2), (c > d)$

- Trace
  - $F_0 := \text{Init}$
  - $F_1 := T$
Example

- System with 2 state vars $c$ and $d$
  - Init: $(d = 1) \land (c \geq d)$
  - Trans: $(c' = c + d) \land (d' = d + 1)$
  - Property: $(d > 2) \rightarrow (c > d)$

- Forward propagation

- Predicates $\mathcal{P}$
  - $(d = 1), (c \geq d), (d > 2), (c > d)$

- Trace
  - $F_0 := Init$
  - $F_1 := \neg (d > 2)$
  - $F_2 := T$
Example

• System with 2 state vars $c$ and $d$
  • Init: $(d = 1) \land (c \geq d)$
  • Trans: $(c' = c + d) \land (d' = d + 1)$
  • Property: $(d > 2) \rightarrow (c > d)$

• Get bad cube at 2
  • $s := \{\neg(d = 1), \neg(c \geq d), (d > 2), \neg(c > d)\}$

• Predicates $\mathcal{P}$
  $(d = 1), (c \geq d), (d > 2), (c > d)$

• Trace
  • $F_0 := Init$
  • $F_1 := \neg(d > 2)$
  • $F_2 := T$
Example

- **System with 2 state vars c and d**
  - Init: \((d = 1) \land (c \geq d)\)
  - Trans: \((c' = c + d) \land (d' = d + 1)\)
  - Property: \((d > 2) \rightarrow (c > d)\)

- **Recursively block s**
  - ...
  - Update \(F_1 := F_1 \land (c \geq d)\)
  - ...
  - Update \(F_2 := F_2 \land (c \geq d) \lor \neg (d > 2)\)

- **Predicates** \(\mathcal{P}\)
  \((d = 1), (c \geq d), (d > 2), (c > d)\)

- **Trace**
  - \(F_0 := \text{Init}\)
  - \(F_1 := \neg (d > 2)\)
  - \(F_2 := \top\)
Example

- **System with 2 state vars** $c$ and $d$
  - **Init:** $(d = 1) \land (c \geq d)$
  - **Trans:** $(c' = c + d) \land (d' = d + 1)$
  - **Property:** $(d > 2) \rightarrow (c > d)$

- **Forward propagation**

- **Predicates** $\mathcal{P}$
  - $(d = 1), (c \geq d), (d > 2), (c > d)$

- **Trace**
  - $F_0 := \text{Init}$
  - $F_1 := \neg(d > 2) \land (c \geq d) \land F_2$
  - $F_2 := (c > d) \lor \neg(d > 2)$
  - $F_3 := \top$
Example

- System with 2 state vars $c$ and $d$
  - Init: $(d = 1) \land (c \geq d)$
  - Trans: $(c' = c + d) \land (d' = d + 1)$
  - Property: $(d > 2) \rightarrow (c > d)$

- Get cube at 3
  - $s := \{\neg (d = 1), \neg (c \geq d), (d > 2), \neg (c > d)\}$

- Predicates $\mathcal{P}$
  $(d = 1), (c \geq d), (d > 2), (c > d)$

- Trace
  - $F_0 := \text{Init}$
  - $F_1 := \neg (d > 2) \land (c \geq d) \land F_2$
  - $F_2 := (c > d) \lor \neg (d > 2)$
  - $F_3 := \top$
Example

- System with 2 state vars $c$ and $d$
  - Init: $(d = 1) \land (c \geq d)$
  - Trans: $(c' = c + d) \land (d' = d + 1)$
  - Property: $(d > 2) \rightarrow (c > d)$

- Recursively block $s$
  - $AbsRelInd$ is sat
  - SMT model:
    $\mu := \{c = 0, d = 2, c' = 0, d = 3, \overline{c} = 2, \overline{d} = 3\}$
  - Abstract predecessor:
    $\{\neg(d > 2), \neg(c > d), \neg(d = 1), \neg(c \geq d)\}$

- Predicates $\mathcal{P}$
  - $(d = 1), (c \geq d), (d > 2), (c > d)$

- Trace
  - $F_0 := Init$
  - $F_1 := \neg(d > 2) \land (c \geq d) \land F_2$
  - $F_2 := (c > d) \lor \neg(d > 2)$
  - $F_3 := \top$
Example

- System with 2 state vars c and d
  - Init: \((d = 1) \land (c \geq d)\)
  - Trans: \((c' = c + d) \land (d' = d + 1)\)
  - Property: \((d > 2) \rightarrow (c > d)\)
- Recursively block c
  - ...  
  - Reached level 0, abstract cex:  
    \(s_0 := \neg(d > 2), \neg(c > d), (d = 1), (c \geq d)\)  
    \(s_1 := \neg(d > 2), \neg(c > d), \neg(d = 1), (c \geq d)\)  
    \(s_2 := \neg(d > 2), \neg(c > d), \neg(d = 1), \neg(c \geq d)\)  
    \(s := \neg(d = 1), \neg(c \geq d), (d > 2), \neg(c > d)\)

- Predicates \(\mathcal{P}\)
  \((d = 1), (c \geq d), (d > 2), (c > d)\)
- Trace
  - \(F_0 := \text{Init}\)
  - \(F_1 := \neg(d > 2) \land (c \geq d) \land F_2\)
  - \(F_2 := (c > d) \lor \neg(d > 2)\)
  - \(F_3 := \top\)
Example

• System with 2 state vars $c$ and $d$
  • Init: $(d = 1) \land (c \geq d)$
  • Trans: $(c' = c + d) \land (d' = d + 1)$
  • Property: $(d > 2) \rightarrow (c > d)$

• Check abstract counterexample

  $s_0(V_0) \land T(V_0, V_1) \land s_1(V_1) \land T(V_1, V_2) \land s_2(V_2) \land T(V_2, V_3) \land s(V_3)$

  UNSAT

• Predicates $\mathcal{P}$

  $(d = 1), (c \geq d), (d > 2), (c > d)$

• Trace

  $F_0 := \text{Init}$
  $F_1 := \neg (d > 2) \land (c \geq d) \land F_2$
  $F_2 := (c > d) \lor \neg (d > 2)$
  $F_3 := T$
Example

- System with 2 state vars $c$ and $d$
  - Init: $(d = 1) \land (c \geq d)$
  - Trans: $(c' = c + d) \land (d' = d + 1)$
  - Property: $(d > 2) \rightarrow (c > d)$

- Check abstract counterexample

- Extract new predicates from sequence interpolants:
  $$d \geq 2, d \geq 3$$

- Update $\mathcal{P}$

- Predicates $\mathcal{P}$
  - $(d = 1), (c \geq d), (d > 2), (c > d), (d \geq 2), (d \geq 3)$

- Trace
  - $F_0 := \text{Init}$
  - $F_1 := \neg(d > 2) \land (c \geq d) \land F_2$
  - $F_2 := (c > d) \lor \neg(d > 2)$
  - $F_3 := \top$
Example

• System with 2 state vars $c$ and $d$
  • Init: $(d = 1) \land (c \geq d)$
  • Trans: $(c' = c + d) \land (d' = d + 1)$
  • Property: $(d > 2) \rightarrow (c > d)$

• Update abstract Trans

• Resume IC3 from level 3

• Predicates $\mathbb{P}$
  
  $(d = 1), (c \geq d),
  
  (d > 2), (c > d),
  
  (d \geq 2), (d \geq 3)$

• Trace

  $F_0 := \text{Init}$
  
  $F_1 := \neg (d > 2) \land (c \geq d) \land F_2$
  
  $F_2 := (c > d) \lor \neg (d > 2) \land F_3$
  
  $F_3 := (d = 1) \lor (d \geq 2) \land
  
  \neg (c \geq d) \land F_4$
  
  $F_4 := (c > d) \lor \neg (d > 2)$
Example

- System with 2 state vars \( c \) and \( d \)
  - Init: \((d = 1) \land (c \geq d)\)
  - Trans: \((c' = c + d) \land (d' = d + 1)\)
  - Property: 
    \[(d > 2) \rightarrow (c > d)\]

- Update abstract Trans
- Resume IC3 from level 3
- ...
- Forward propagation
  \[F_2 \land \widehat{T}_\mathcal{P} \models (c' \geq d') \lor \neg (d' \geq 2)\]

- Predicates \( \mathcal{P} \)
  \[
  (d = 1), (c \geq d),
  (d > 2), (c > d),
  (d \geq 2), (d \geq 3)
  \]

- Trace
  - \( F_0 := \text{Init} \)
  - \( F_1 := \neg(d > 2) \land (c \geq d) \land F_2 \)
  - \( F_2 := (c > d) \lor \neg(d > 2) \land F_3 \)
  - \( F_3 := (d = 1) \lor (d \geq 2) \land \neg(c \geq d) \land F_4 \)
  - \( F_4 := (c > d) \lor \neg(d > 2) \)
Example

- System with 2 state vars $c$ and $d$
  - Init: $(d = 1) \land (c \geq d)$
  - Trans: $(c' = c + d) \land (d' = d + 1)$
  - Property: $(d > 2) \rightarrow (c > d)$
- Update abstract Trans
- Resume IC3 from level 3
- ... 
- Forward propagation
  $$F_2 \land \hat{T}_P \models (c' \geq d') \lor \neg (d' \geq 2)$$
- Fixpoint $\Rightarrow$ Property is true

- Predicates $P$
  $$(d = 1), (c \geq d), (d > 2), (c > d), (d \geq 2), (d \geq 3)$$

- Trace
  - $F_0 := Init$
  - $F_1 := \neg (d > 2) \land (c \geq d) \land F_2$
  - $F_2 := F_3 := (c \geq d) \lor \neg (d \geq 2) \land (d = 1) \lor (d \geq 2) \land \neg (c \geq d) \land F_4$
  - $F_4 := (c > d) \lor \neg (d > 2)$
Infinite State Model-Checking

Liveness Checking
LTL from Finite to Infinite

- Use first-order predicates instead of propositions:
  - \( G(x \geq a \land x \leq b) \)
  - \( GF(x = a) \land GF(x = b) \)

- Predicates interpreted according to specific theory

- “next” variables to express changes/transitions:
  - \( G(x' = x + 1) \)
  - \( G(a' - a \leq b) \)

- BMC
  - Add encoding of lasso-shape and fairness
  - Sound for finding traces, but not complete
  - The only counterexample may be not lasso-shape

- K-liveness
  - No change
  - Sound to prove properties, but not complete
  - Property may hold, but fairness can be visited an unbounded number of times

Stefano Tonetta, SSFT'16
Liveness to Safety for Infinite States

• Unsound for infinite-state systems
  • Not all counterexamples are lasso-shaped

\[ I(S) \overset{\text{def}}{=} (x = 0) \quad T(S) \overset{\text{def}}{=} (x' = x + 1) \quad \varphi \overset{\text{def}}{=} F G(x < 5) \]

• Liveness to safety with Implicit Abstraction
  • Apply the \text{i2s} transformation to the abstract system
    • Save the values of the predicates instead of the concrete state
  • Do it on-the-fly, tightly integrating \text{i2s} with IC3
  • Sound but incomplete
    • When abstract loop found, simulate in the concrete and refine
    • Might still diverge during refinement
      • Intrinsic limitation of state predicate abstraction

Stefano Tonetta, SSFT'16
Wrap-up
Lecture Summary

• Overview of SAT-based model checking techniques
• Details on IC3, as currently the prominent algorithm
• Liveness reduced to safety
• Lifting SAT-based MC to SMT
  • For invariant checking
    • Easy for BMC and k-induction
    • Predicate abstraction to reduce to finite-state MC
    • Implicit abstraction to avoid explicit computation of abstract state space
    • Implicit abstraction to lift IC3 to SMT
  • For liveness
    • BMC and K-liveness sound but not complete
    • Liveness2safety on abstract state space
Not covered

• Other MC approaches: BDD-Based, Interpolation, ...

• Other Properties: CTL, PSL, termination, epistemic, ...

• Other kind of systems
  • Continuous-time/hybrid systems
  • Probabilistic Systems
  • Software (control-flow graphs)
  • ...

Stefano Tonetta, SSFT'16
Next lecture

L1
- Functional Verification
- Model-Checking

L2
- Safety Assessment
  - Model-Based Safety Assessment
- Hierarchical Decomposition
  - Contract-Based Design
  - Contract-Based Safety Assessment
Readings

A list of suggested readings on the topics of the course. The list is not meant to be complete.

- Model checking:
  - Edmund M. Clarke, Orna Grumberg, Doron A. Peled: Model Checking. The MIT Press, 1999

- Bounded Model Checking:
Readings

• **K-induction:**

• **IC3 for Finite-State Transition Systems:**
  - Aaron R. Bradley: SAT-Based Model Checking without Unrolling. VMCAI 2011: 70-87
  - Fabio Somenzi, Aaron R. Bradley: IC3: where monolithic and incremental meet. FMCAD 2011: 3-8
  - Aaron R. Bradley: Understanding IC3. SAT 2012: 1-14
Readings

- **LTL Model Checking:**
  - Amir Pnueli: The Temporal Logic of Programs. FOCS 1977: 46-57

- **Liveness to safety:**
  - Koen Claessen, Niklas Sörensson: A liveness checking algorithm that counts. FMCAD 2012: 52-59
Readings

- K-Induction for Infinite-State Systems:
  - Leonardo Mendonça de Moura, Harald Rueß, Maria Sorea: Bounded Model Checking and Induction: From Refutation to Verification (Extended Abstract, Category A). CAV 2003: 14-26
  - Jonathan Laurent, Alwyn Goodloe, Lee Pike: Assuring the Guardians. RV 2015: 87-101
Readings

- Interpolation-based Model Checking:
  - Kenneth L. McMillan: Applications of Craig Interpolants in Model Checking. TACAS 2005: 1-12

- Liveness to Safety for Infinite-State Systems:
Readings

• Implicit Abstraction:
  • Stefano Tonetta: Abstract Model Checking without Computing the Abstraction. FM 2009: 89-105

• IC3 for Infinite-State Systems:
  • Alessandro Cimatti, Alberto Griggio: Software Model Checking via IC3. CAV 2012: 277-293
  • Alessandro Cimatti, Alberto Griggio, Sergio Mover, Stefano Tonetta: IC3 Modulo Theories via Implicit Predicate Abstraction. TACAS 2014: 46-61
  • Johannes Birgmeier, Aaron R. Bradley, Georg Weissenbacher: Counterexample to Induction-Guided Abstraction-Refinement (CTIGAR). CAV 2014: 831-848
  • Yakir Vizel, Arie Gurfinkel: Interpolating Property Directed Reachability. CAV 2014: 260-276
  • Nikolaj Bjørner, Arie Gurfinkel: Property Directed Polyhedral Abstraction. VMCAI 2015: 263-281