SMT-based Model Checking

Cesare Tinelli

The University of Iowa

The Intuition That Started it All

A software or hardware system S can be modeled as a *state transition system* $\mathcal{M} = (\mathcal{S}, \mathcal{I}, \mathcal{T}, \mathcal{L})$ where

- $\mathcal{S}$ is a set of *states*, the state space
- $\mathcal{I} \subseteq \mathcal{S}$ is a set of *initial states*
- $\mathcal{T} \subseteq \mathcal{S} \times \mathcal{S}$ is a (right-total) *transition relation*
- $\mathcal{L} : \mathcal{S} \to 2^{\mathcal{P}}$ is a *labeling function*
  where $\mathcal{P}$ is a set of *state predicates*

$\mathcal{M}$ can be seen as a Kripke structure

The Intuition That Started it All

Functional properties of S can be expressed in a suitable temporal logic that admits $\mathcal{M} = (\mathcal{S}, \mathcal{I}, \mathcal{T}, \mathcal{L})$ as a model

The Intuition That Started it All

Functional properties of S can be expressed in a suitable temporal logic that admits $\mathcal{M} = (\mathcal{S}, \mathcal{I}, \mathcal{T}, \mathcal{L})$ as a model

Checking properties of S then reduces to satisfiability checking in $\mathcal{M}$

The Intuition That Started it All

Functional properties of S can be expressed in a suitable temporal logic that admits $\mathcal{M} = (\mathcal{S}, \mathcal{I}, \mathcal{T}, \mathcal{L})$ as a model

Checking properties of S then reduces to satisfiability checking in $\mathcal{M}$

Model Checking!

Model Checking of Finite State Systems

A temporal logic extends a classical base logic $\mathbb{L}$ with temporal operators

Model Checking of Finite State Systems

A temporal logic extends a classical base logic $\mathbb{L}$ with temporal operators

Traditionally in model checking, $\mathbb{L}$ has been propositional logic

This limits model checking to finite-state systems

Model Checking of Finite State Systems

A temporal logic extends a classical base logic $\mathbb{L}$ with temporal operators

Traditionally in model checking, $\mathbb{L}$ has been propositional logic

This limits model checking to finite-state systems

Under the right conditions, more powerful logics $\mathbb{L}$ can be used

This is especially the case for safety checking and its dual, invariance checking

Logic-based Safety Checking

Necessary condition: can represent $\mathcal{M} = (\mathcal{S}, \mathcal{I}, \mathcal{T}, \mathcal{L})$ symbolically in some (classical) logic $\mathbb{L}$ with decidable entailment $\models_{\mathbb{L}}$

($\varphi \models_{\mathbb{L}} \psi$ iff $\varphi \wedge \neg\psi$ is unsatisfiable in $\mathbb{L}$)

Logic-based Safety Checking

Necessary condition: can represent $\mathcal{M} = (\mathcal{S}, \mathcal{I}, \mathcal{T}, \mathcal{L})$ symbolically in some (classical) logic $\mathbb{L}$ with decidable entailment $\models_{\mathbb{L}}$

($\varphi \models_{\mathbb{L}} \psi$ iff $\varphi \wedge \neg\psi$ is unsatisfiable in $\mathbb{L}$)

Examples of $\mathbb{L}$:

- Propositional logic
- Quantified Boolean Formulas
- Bernay-Schönfinkel logic
- Bit vector logic
- Quantifier-free real (or linear integer) arithmetic
- ...

Logical encodings of transitions systems

$\mathcal{M} = (\mathcal{S}, \mathcal{I}, \mathcal{T}, \mathcal{L})$ V: set of *values* in $\mathbb{L}$ $\mathbf{x}$: n-tuple of *variables*

THE UNIVERSITY OF IOWA

Logical encodings of transitions systems

$\mathcal{M} = (\mathcal{S}, \mathcal{I}, \mathcal{T}, \mathcal{L})$ V: set of *values* in $\mathbb{L}$ $\mathbf{x}$: n-tuple of *variables*

- states $\mathbf{s} \in \mathcal{S}$ encoded as n-tuples of V^n

The University of Iowa

Logical encodings of transitions systems

$\mathcal{M} = (\mathcal{S}, \mathcal{I}, \mathcal{T}, \mathcal{L})$ V: set of *values* in $\mathbb{L}$ $\mathbf{x}$: n-tuple of *variables*

- states $\mathbf{s} \in \mathcal{S}$ encoded as n-tuples of V^n
- $\mathcal{I}$ encoded as a formula $I[\mathbf{x}]$ such that

$$\mathbf{s} \in \mathcal{I} \text{ iff } \models_{\mathbb{L}} I[\mathbf{s}]$$

Logical encodings of transitions systems

$\mathcal{M} = (\mathcal{S}, \mathcal{I}, \mathcal{T}, \mathcal{L})$ V: set of *values* in $\mathbb{L}$ $\mathbf{x}$: n-tuple of *variables*

- states $\mathbf{s} \in \mathcal{S}$ encoded as n-tuples of V^n
- $\mathcal{I}$ encoded as a formula $I[\mathbf{x}]$ such that

$$\mathbf{s} \in \mathcal{I} \text{ iff } \models_{\mathbb{L}} I[\mathbf{s}]$$

- $\mathcal{T}$ encoded as a formula $T[\mathbf{x}, \mathbf{x}']$ such that

$$\models_{\mathbb{L}} T[\mathbf{s}, \mathbf{s}'] \text{ for all } (\mathbf{s}, \mathbf{s}') \in \mathcal{T}$$

Logical encodings of transitions systems

$\mathcal{M} = (\mathcal{S}, \mathcal{I}, \mathcal{T}, \mathcal{L})$ V: set of *values* in $\mathbb{L}$ $\mathbf{x}$: n-tuple of *variables*

- states $\mathbf{s} \in \mathcal{S}$ encoded as n-tuples of V^n
- $\mathcal{I}$ encoded as a formula $I[\mathbf{x}]$ such that

$$\mathbf{s} \in \mathcal{I} \text{ iff } \models_{\mathbb{L}} I[\mathbf{s}]$$

- $\mathcal{T}$ encoded as a formula $T[\mathbf{x}, \mathbf{x}']$ such that

$$\models_{\mathbb{L}} T[\mathbf{s}, \mathbf{s}'] \text{ for all } (\mathbf{s}, \mathbf{s}') \in \mathcal{T}$$

- State properties encoded as formulas $P[\mathbf{x}]$

Main Logic-based Approaches

- Bounded model checking
- Interpolation-based model checking
- Property Directed Reachability (IC3)
- Temporal induction
- Backward reachability
- ...

Past accomplishments: mostly based on propositional logic, with SAT solvers as reasoning engines

Main Logic-based Approaches

- Bounded model checking
- Interpolation-based model checking
- Property Directed Reachability (IC3)
- Temporal induction
- Backward reachability
- ...

Past accomplishments: mostly based on propositional logic, with SAT solvers as reasoning engines

New frontier: based on logics decided by solvers for

Satisfiability Modulo Theories

Safety Checking Modulo Theories

We invariably reason about computational systems in the context of some theory $\mathbb{T}$ of their data types

Examples

Pipelined microprocessors: theory of equality, atoms like

$$f(g(a,b),c) = g(c,a)$$

Timed automata: theory of integers/reals, atoms like

$$x - y < 2$$

General software: combination of theories, atoms like

$$a[2 * j + 1] + x \geq car(l) - f(x)$$

Such reasoning can be reduced to checking the satisfiability of certain formulas in (or *modulo*) $\mathbb{T}$

Satisfiability Modulo Theories

The satisfiability of quantifier-free formulas is decidable for many theories $\mathbb{T}$ of interest in model checking

THE UNIVERSITY OF IOWA

Satisfiability Modulo Theories

The satisfiability of quantifier-free formulas is decidable for many theories $\mathbb{T}$ of interest in model checking

- Equality with "Uninterpreted Function Symbols"
- Linear Arithmetic (Real and Integer)
- Bit vectors
- Arrays (i.e., updatable maps)
- Finite sets and multisets
- Strings
- Inductive data types (enumerations, lists, trees, . . .)
- . . .

Satisfiability Modulo Theories

The satisfiability of quantifier-free formulas is decidable for many theories $\mathbb{T}$ of interest in model checking

Thanks to advances in SAT and in decision procedures, this can be done very efficiently in practice by current SMT solvers

SMT Solvers

Provide additional functionalities besides satisfiability checking

- compute satisfying assignments
- evaluate terms
- identify unsatisfiable cores
- generate interpolants
- eliminate quantifiers
- construct proof objects
- optimize objective functions
- ...

SAT vs SMT in Safety Checking

SMT encodings provide several advantages over SAT encodings:

- more powerful language

  (unquantified) first-order formulas instead of Boolean formulas

- satisfiability still efficiently decidable
- similar high level of automation
- more natural and compact encodings
- greater scalability
- not limited to finite-state systems

Unifying Theme in SMT-based MC

Def. The *strongest inductive invariant (for $\mathcal{M}$ in $\mathbb{L}$)* is a formula $R[\mathbf{x}]$ such that $\models_{\mathbb{L}} R[\mathbf{s}]$ iff $\mathbf{s}$ is reachable

Unifying Theme in SMT-based MC

Def. The *strongest inductive invariant (for $\mathcal{M}$ in $\mathbb{L}$)* is a formula $R[\mathbf{x}]$ such that $\models_{\mathbb{L}} R[\mathbf{s}]$ iff $\mathbf{s}$ is reachable

Suppose we can compute R from an $\mathbb{L}$-encoding (I, T) of $\mathcal{M}$

Unifying Theme in SMT-based MC

Def. The *strongest inductive invariant (for $\mathcal{M}$ in $\mathbb{L}$)* is a formula $R[\mathbf{x}]$ such that $\models_{\mathbb{L}} R[\mathbf{s}]$ iff $\mathbf{s}$ is reachable

Suppose we can compute R from an $\mathbb{L}$-encoding (I, T) of $\mathcal{M}$

To check that some $P[\mathbf{x}]$ is invariant for $\mathcal{M}$ it suffices to check that $R[\mathbf{x}] \models_{\mathbb{L}} P[\mathbf{x}]$

Unifying Theme in SMT-based MC

Def. The *strongest inductive invariant (for $\mathcal{M}$ in $\mathbb{L}$)* is a formula $R[\mathbf{x}]$ such that $\models_{\mathbb{L}} R[\mathbf{s}]$ iff $\mathbf{s}$ is reachable

Suppose we can compute R from an $\mathbb{L}$-encoding (I, T) of $\mathcal{M}$

To check that some $P[\mathbf{x}]$ is invariant for $\mathcal{M}$ it suffices to check that $R[\mathbf{x}] \models_{\mathbb{L}} P[\mathbf{x}]$

Problem: R may be very expensive or impossible to compute or even represent in $\mathbb{L}$

Unifying Theme in SMT-based MC

Def. The *strongest inductive invariant (for $\mathcal{M}$ in $\mathbb{L}$)* is a formula $R[\mathbf{x}]$ such that $\models_{\mathbb{L}} R[\mathbf{s}]$ iff $\mathbf{s}$ is reachable

Suppose we can compute R from an $\mathbb{L}$-encoding (I, T) of $\mathcal{M}$

To check that some $P[\mathbf{x}]$ is invariant for $\mathcal{M}$ it suffices to check that $R[\mathbf{x}] \models_{\mathbb{L}} P[\mathbf{x}]$

SMT-based safety checking is about approximating R in $\mathbb{L}$ as efficiently as possible and as precisely as needed, with the help of SMT solvers

Main Idea

With the aid of a solver for $\mathbb{L}$, find or construct $\widehat{R}[\mathbf{x}]$ such that

1. $\widehat{R}$ is invariant
2. $\widehat{R}$ entails the input property P

Main Idea

With the aid of a solver for $\mathbb{L}$, find or construct $\widehat{R}[\mathbf{x}]$ such that

1. $\widehat{R}$ is invariant
2. $\widehat{R}$ entails the input property P

$\widehat{R}$ is a *witness* of P's invariance

Temporal Induction

Find $k \geq 0$ such that

1. $$I[\mathbf{x}_0] \wedge T[\mathbf{x}_0, \mathbf{x}_1] \wedge \cdots \wedge T[\mathbf{x}_{k-1}, \mathbf{x}_k] \models_{\mathbb{L}} P[\mathbf{x}_0] \wedge \cdots \wedge P[\mathbf{x}_k]$$

2. $$P[\mathbf{x}_0] \wedge \cdots \wedge P[\mathbf{x}_k] \wedge T[\mathbf{x}_0, \mathbf{x}_1] \wedge \cdots \wedge T[\mathbf{x}_{k-1}, \mathbf{x}_k] \models_{\mathbb{L}} P[\mathbf{x}_{k+1}]$$

$$\boxed{\widehat{R} = P}$$

Requires solver that:

- decides $\models_{\mathbb{L}}$

Interpolation-based MC

For some $k > 0$, compute a sequence $\widehat{R}^0[\mathbf{x}], \ldots, \widehat{R}^n[\mathbf{x}]$ such that

1. $R^i[\mathbf{x}] \models_{\mathbb{L}} \widehat{R}^i[\mathbf{x}]$ (R^i denotes states reachable in up to i steps)
2. $\widehat{R}^i[\mathbf{x}_1] \wedge T[\mathbf{x}_1, \mathbf{x}_2] \wedge \cdots \wedge T[\mathbf{x}_{k-1}, \mathbf{x}_k] \models_{\mathbb{L}} P[\mathbf{x}_1] \wedge \cdots \wedge P[\mathbf{x}_k]$
3. $\widehat{R}^i[\mathbf{x}] \models_{\mathbb{L}} \widehat{R}^{i+1}[\mathbf{x}]$
4. $\widehat{R}^n[\mathbf{x}] \models_{\mathbb{L}} \widehat{R}^{n-1}[\mathbf{x}]$

$$\boxed{\widehat{R} = \widehat{R}^n[\mathbf{x}]}$$

Requires solver that:

- decides $\models_{\mathbb{L}}$
- produces interpolants in $\mathbb{L}$

IC3

Compute a sequence $\widehat{R}^0[\mathbf{x}], \ldots, \widehat{R}^n[\mathbf{x}]$ such that

1. $R^i[\mathbf{x}] \models_{\mathbb{L}} \widehat{R}^i[\mathbf{x}]$ (R^i denotes states reachable in up to i steps)
2. $\widehat{R}^i[\mathbf{x}] \models_{\mathbb{L}} P[\mathbf{x}]$
3. $\widehat{R}^i[\mathbf{x}] \models_{\mathbb{L}} \widehat{R}^{i+1}[\mathbf{x}]$
4. $\widehat{R}^n[\mathbf{x}] \models_{\mathbb{L}} \widehat{R}^{n-1}[\mathbf{x}]$

$$\boxed{\widehat{R} = \widehat{R}^n[\mathbf{x}]}$$

Requires solver that:

- decides $\models_{\mathbb{L}}$
- generalizes induction counterexamples
- produces unsat cores

Some Future Directions

- New SMT techniques to work with quantified transition relations/interpolants/invariants/...
- Compositional model checking techniques built on Horn clause-based SMT encodings
- Synergistic combinations of SMT with traditional abstract interpretation techniques and tools
- Promising cross-fertilization between SMT-based model checking and SMT-based program synthesis
- Checking of non-functional properties
  (i.e., worst-case execution time)