Thorough Checking Revisited

Shiva Nejati    Mihaela Gheorghiu    Marsha Chechik
{shiva,mg,chechik}@cs.toronto.edu

University of Toronto
Automated Abstraction

SW/HW Artifact

Model Extraction

Finite Abstract Model

Inconclusive Answer

Model-Checker

Conclusive Answer

Correctness Property

Translation

Temporal Logic

Conclusive Answer
3-Valued Abstraction

Partial Models → Model-Checker

Maybe

Model-Checker → Universal + Existential Properties

Yes/No

Partial Models

Model-Checker

Universal + Existential Properties

Yes/No
3-Valued Abstraction

Partial Models

Model-Checker

Universal + Existential Properties

PKS [BG00]
MixedTS [DGG97]
HTS [SG04] [LX90]

Yes/No

Maybe

Partial Models

Universal + Existential Properties

PKS [BG00]
MixedTS [DGG97]
HTS [SG04] [LX90]

Yes/No

Maybe
3-Valued Abstraction

Partial Models

- PKS [BG00]
- MixedTS [DGG97]
- HTS [SG04] [LX90]

Model-Checker

- Universal + Existential Properties
- Compositional Semantics
- Thorough Semantics [Bruns & Godefroid 00]

Yes/No

Maybe
3-Valued Semantics: Example

P:
int \( x, y = 1, 1; \)
int \( t; \)
\( x, y = t, t+1; \)
\( x, y = 1, 1; \)

Property: \( \text{AG}(\text{odd}(y)) \land \text{A}[\text{odd}(x) \lor \neg \text{odd}(y)] \)

<table>
<thead>
<tr>
<th>Compositional Semantics</th>
<th>Thorough Semantics</th>
</tr>
</thead>
</table>

M:
\( \text{odd}(x) \)
\( \text{odd}(y) \)
\( \text{odd}(x)? \)
\( \text{odd(y)}? \)
\( \text{odd}(x) \)
\( \text{odd}(y) \)
3-Valued Semantics: Example

P:
\[
\begin{align*}
\text{int } x, y &= 1, 1; \\
\text{int } t; \\
x, y &= t, t+1; \\
x, y &= 1, 1;
\end{align*}
\]

Property: \( AG(\text{odd}(y)) \land A[\text{odd}(x) \cup \neg\text{odd}(y)] \)

<table>
<thead>
<tr>
<th>Semantics</th>
<th>( AG(\text{odd}(y)) \land A[\text{odd}(x) \cup \neg\text{odd}(y)] )</th>
</tr>
</thead>
<tbody>
<tr>
<td>Compositional</td>
<td>( AG(\text{odd}(y)) \land A[\text{odd}(x) \cup \neg\text{odd}(y)] )</td>
</tr>
<tr>
<td>Thorough</td>
<td>( AG(\text{odd}(y)) \land A[\text{odd}(x) \cup \neg\text{odd}(y)] )</td>
</tr>
</tbody>
</table>
3-Valued Semantics: Example

P:
    int x, y = 1, 1;
    int t;
    x, y = t, t+1;
    x, y = 1, 1;

M:
    odd(x)
    odd(y)
    odd(x) ?
    odd(y) ?
    odd(x)
    odd(y)

Property: $AG(\text{odd}(y)) \land A[\text{odd}(x) U \neg\text{odd}(y)]$

<table>
<thead>
<tr>
<th>Compositional Semantics</th>
<th>Maybe $\land A[\text{odd}(x) U \neg\text{odd}(y)]$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thorough Semantics</td>
<td></td>
</tr>
</tbody>
</table>
3-Valued Semantics: Example

P:
int x, y = 1, 1;
int t;
x, y = t, t+1;
x, y = 1, 1;

M:
odd(x)
odd(y)
odd(x) ?
odd(y) ?
odd(x)
odd(y)

Property: AG(odd(y)) \land A[odd(x) U \neg odd(y)]

<table>
<thead>
<tr>
<th>Compositional Semantics</th>
<th>Maybe \land Maybe</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thorough Semantics</td>
<td></td>
</tr>
</tbody>
</table>
3-Valued Semantics: Example

P:
int x, y = 1, 1;
int t;
x, y = t, t+1;
x, y = 1, 1;

Property: AG(\text{odd}(y)) \land A[\text{odd}(x) \cup \neg\text{odd}(y)]

<table>
<thead>
<tr>
<th>Compositional Semantics</th>
<th>Maybe</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thorough Semantics</td>
<td></td>
</tr>
</tbody>
</table>
3-Valued Semantics: Example

P:
int x, y = 1, 1;
int t;
x, y = t, t+1;
x, y = 1, 1;

M:
odd(x)
odd(y)
odd(x) ?
odd(y) ?
odd(x)
odd(y)
odd(x) ?
odd(y) ?
odd(x)
odd(y)

One concretization
odd(x)
odd(y)
odd(x)
odd(y)
¬odd(x)
¬odd(y)
¬odd(x)
¬odd(y)
¬odd(x)
¬odd(y)

Property: AG(odd(y)) ∧ A[odd(x) U ¬odd(y)]

<table>
<thead>
<tr>
<th>Compositional Semantics</th>
<th>Maybe</th>
</tr>
</thead>
</table>
| Thorough Semantics      | AG(odd(y)) ∧ A[odd(x) U ¬odd(y)]
                           | False over all Concretizations of M |
3-Valued Semantics: Example

P:
int x, y = 1, 1;
int t;
x, y = t, t+1;
x, y = 1, 1;

Property: $AG(\text{odd}(y)) \land A[\text{odd}(x) \lor \neg\text{odd}(y)]$

<table>
<thead>
<tr>
<th>Compositional Semantics</th>
<th>Maybe</th>
</tr>
</thead>
<tbody>
<tr>
<td>Thorough Semantics</td>
<td>False</td>
</tr>
</tbody>
</table>
Compositional vs Thorough

Partial Models

Model-Checker

Compositional Semantics

- ✔ Computationally cheap
- ✗ Less precise (more maybe’s)
- ✔ Various implementations

Thorough Semantics

- ✗ Computationally expensive
- ✔ More precise (less maybe’s)
- ✗ No implementation

Need to increase conclusiveness while avoiding too much overhead
Implementing Thorough via Compositional

→ Identify formulas where compositional = thorough

  ✶ **Self-minimizing formulas** [Godefroid & Huth 05]
  ✶ **E.g.** $AG(\text{odd}(y))$

→ Transform other formulas into equivalent self-minimizing ones

  ✶ **Semantic minimization** [Reps et. al. 02]
  ✶ **E.g.** $AG(\text{odd}(y)) \land A[\text{odd}(x) \lor \neg \text{odd}(y)]$
    \[
    \equiv
    \quad A[(\text{odd}(x) \land \text{odd}(y)) \lor \text{False}] \quad (\text{Self-minimizing})
    \]
Thorough Checking Algorithm

\[ \text{ThoroughCheck}(M, \varphi) \]
\[ (1): \quad \text{if } (v := \text{ModelCheck}(M, \varphi)) \neq \text{Maybe} \]
\[ \quad \text{return } v \]
\[ (2): \quad \text{if } \text{IsSelfMinimizing}(M, \varphi) \]
\[ \quad \text{return } \text{Maybe} \]
\[ (3): \quad \text{return } \text{ModelCheck}(M, \text{SemanticMinimization}(\varphi)) \]
Thorough Checking Algorithm

\textbf{ThoroughCheck}(M, \varphi)
(1): \text{if} (v := \text{ModelCheck}(M, \varphi)) \neq \text{Maybe}\checkmark
\text{return} v
(2): \text{if} \text{IsSelfMinimizing}(M, \varphi)
\text{return} \text{Maybe}
(3): \text{return} \text{ModelCheck}(M, \text{SemanticMinimization}(\varphi))
Our Goal

ThoroughCheck($M$, $\varphi$)
(1): if ($v := \text{ModelCheck}(M, \varphi)$) $\neq$ Maybe
  return $v$
(2): if $\text{IsSelfMinimizing}(M, \varphi)$
  return Maybe
(3): return ModelCheck($M$, SemanticMinimization($\varphi$))

→ Step (2):
  ➡ Identifying a large class of self-minimizing formulas

→ Step (3):
  ➡ Devising practical algorithms for semantic minimization of remaining formulas
Our Contributions

1. We prove that disjunctive/conjunctive $\mu$-calculus formulas are self-minimizing

   ➤ Related Work:
   ➤ [Gurfinkel & Chechik 05] [Godefroid & Huth 05] checking pure polarity
   ➤ Only works for PKSs, not for all partial models

2. We provide a semantic minimization algorithm via the tableau-based translation of [Janin & Walukiewicz 95]

   ➤ Related Work:
   ➤ [Godefroid & Huth 05]: $\mu$-calculus is closed under semantic-minimization
   ➤ But no implementable algorithm
Main Idea

→ Thorough checking can be as hard as satisfiability checking

▷ Satisfiability checking is linear for disjunctive μ-calculus
  ➢ Then, can we show that disjunctive μ-calculus is self-minimizing?
  ➢ But, a naive inductive proof does not work for the greatest fixpoint formulas [Godefroid & Huth 05]

→ Our proof uses an automata characterization of thorough checking

▷ reducing checking self-minimization to deciding an automata intersection game
Outline

→ Need for thorough checking

→ Thorough via compositional

→ Main Result: Disjunctive/Conjunctive $\mu$-calculus is self-minimizing
 pherd by
  
  • Intuition
  • Background
  • Proof

→ Our thorough checking algorithm

→ Conclusion and future work
Background

→ **Disjunctive μ-calculus** [Janin and Walukiewicz 95]

느 Conjunctions are restricted (special conjunctions)

느 **Examples**

\[ \varphi_1 = \text{EX}p \land \text{EX}\lnot q \land \text{AX}(p \lor \lnot q) \] ✔

\[ \varphi_2 = \text{AX}(p \land q) \] ✔

\[ \varphi_3 = \text{AX}p \land \text{AX}q \] ✘

 느 **Syntax**

\[ \varphi ::= p \mid \lnot p \mid Z \mid \varphi \lor \varphi \mid p \land \bigwedge_{\psi \in \Gamma} \text{EX} \psi \land \text{AX} \bigvee_{\psi \in \Gamma} \psi \mid \nu(Z) \cdot \varphi(Z) \mid \mu(Z) \cdot \varphi(Z) \]

→ **Conjunctive μ-calculus is dual**

→ **Disjunctive μ-calculus is equal to μ-calculus**
Background:

Abstraction as Automata [Dams & Namjoshi 05]

→ Formulas = automata, abstract models = automata

💡 Model Checking
Model M satisfies formula \( \varphi \) \( \mathcal{L}(A_M) \subseteq \mathcal{L}(A_\varphi) \)

💡 Refinement Checking
Model M abstracts model M' \( \mathcal{L}(A_M) \subseteq \mathcal{L}(A_M') \)

→ We use \( \mu \)-automata [Janin & Walukiewicz 95]

💡 Similar to non-deterministic tree automata

💡 But
- no fixed branching degree
- no ordering over successors
Self-minimization and Automata

A formula $\varphi$ is self-minimizing if

1. For every abstract model $M$ over which $\varphi$ is non-false (true or maybe)
   there is a completion of $M$ satisfying $\varphi$

2. For every abstract model $M$ over which $\varphi$ is non-true (false or maybe)
   there is a completion of $M$ refuting $\varphi$
A formula $\varphi$ is self-minimizing if

1. For every abstract model $M$ over which $\varphi$ is non-false (true or maybe)

$$\mathcal{L}(A_M) \cap \mathcal{L}(A_\varphi) \neq \emptyset$$

2. For every abstract model $M$ over which $\varphi$ is non-true (false or maybe)

there is a completion of $M$ refuting $\varphi$
Self-minimization and Automata

→ A formula $\varphi$ is self-minimizing if

1. For every abstract model $M$ over which $\varphi$ is non-false (true or maybe)

$$\mathcal{L}(A_M) \cap \mathcal{L}(A_\varphi) \neq \emptyset$$

2. For every abstract model $M$ over which $\varphi$ is non-true (false or maybe)

$$\mathcal{L}(A_M) \cap \mathcal{L}(A_{\neg \varphi}) \neq \emptyset$$
Self-minimization and Automata

→ A formula $\varphi$ is self-minimizing if

1. For every abstract model $M$ over which $\varphi$ is non-false (true or maybe)

$$\mathcal{L}(A_M) \cap \mathcal{L}(A_\varphi) \neq \emptyset$$

2. For every abstract model $M$ over which $\varphi$ is non-true (false or maybe)

$$\mathcal{L}(A_M) \cap \mathcal{L}(A_{\neg \varphi}) \neq \emptyset$$

→ Existing partial model formalisms can be translated to $\mu$-automata

→ There exists a linear syntactic translation from disjunctive $\mu$-calculus to $\mu$-automata

[Janin & Walukiewicz 95]
Outline

→ Need for thorough checking

→ Thorough via compositional

→ Main Result: Disjunctive/Conjunctive $\mu$-calculus is self-minimizing
  ➤ Intuition
  ➤ Background
  ➤ Proof

→ Our thorough checking algorithm

→ Conclusion and future work
Main Result

\( \rightarrow \) Let \( \varphi \) be a disjunctive formula. Show:

for every abstract model \( M \) over which \( \varphi \) is non-false

\[ \mathcal{L}(A_M) \cap \mathcal{L}(A_{\varphi}) \neq \emptyset \]

\( \rightarrow \) The case for conjunctive \( \varphi \) is dual

\( \rightarrow \) Proof Steps:

1. Translate models and formulas to \( \mu \)-automata

2. Find a winning strategy for an intersection game between \( A_M \) and \( A_{\varphi} \) (by structural induction)
Illustrating the Proof

→ Show that $AG_p$ is self-minimizing

⇔ i.e., $\forall M$ over which $\phi$ is non-false

$L(A_M) \cap L(A_{AG_p}) \neq \emptyset$

Choose $M$

\[ p \]
\[ q \]

$AG_p$

\[ \neg p \]
\[ q \]

\[ \neg q \]

\[ p \]
\[ q \]

\[ p \]
\[ q \]

\[ p \]
\[ q \]
Illustrating the Proof

→ Show that AGp is self-minimizing

⇒ i.e., ∀M over which φ is non-false

\[ \mathcal{L}(A_M) \cap \mathcal{L}(A_{AGP}) \neq \emptyset \]

1. Translate models and formulas to μ-automata

Choose M

\[ \begin{array}{c}
 p \\
 q \\
 \end{array} \]

AGp

\[ \begin{array}{c}
 s_0 \\
 s_1 \\
 s_2 \\
 s_3 \\
 \end{array} \]

\[ \begin{array}{c}
 \neg p \\
 q \\
 \end{array} \]

\[ \begin{array}{c}
 p \\
 \neg q \\
 \end{array} \]

\[ \begin{array}{c}
 p \\
 q \\
 \end{array} \]
Illustrating the Proof

→ Show that $AGp$ is self-minimizing

⇒ i.e., $\forall M$ over which $\varphi$ is non-false

$L(A_M) \cap L(A_{AGP}) \neq \emptyset$

1. Translate models and formulas to $\mu$-automata
Illustrating the Proof

→ Show that AGp is self-minimizing

⇔ i.e., ∀M over which ϕ is non-false

\[ \mathcal{L}(A_M) \cap \mathcal{L}(A_{AGp}) \neq \emptyset \]

1. Translate models and formulas to μ-automata
Illustrating the Proof

Show that $\mathcal{AGp}$ is self-minimizing

i.e., $\forall M$ over which $\varphi$ is non-false

$L(\mathcal{A}_M) \cap L(\mathcal{A}_{AGp}) \neq \emptyset$

2. Find a winning strategy for an intersection game
Show that $AGp$ is self-minimizing
\[ L(A_M) \cap L(AG) \neq \emptyset \]

2. Find a winning strategy for an intersection game
2. Find a winning strategy for an intersection game

→ Show that $AGp$ is self-minimizing

\[ \forall M \text{ over which } \varphi \text{ is non-false} \]

\[ \mathcal{L}(A_M) \cap \mathcal{L}(A_{AGp}) \neq \emptyset \]

Illustrating the Proof

Proof by structural induction (see the paper)
Main Result

→Proof Steps:

1. Translate models and formulas to $\mu$-automata
2. Find a winning strategy for an intersection game

→In conclusion:

\(\Rightarrow\) Disjunctive/conjunctive $\mu$-calculus formulas are self-minimizing
\(\Rightarrow\) Every $\mu$-calculus formula can be translated to its disjunctive/conjunctive form
Outline

→ Need for thorough checking

→ Thorough via compositional

→ Main Result: Disjunctive/Conjunctive $\mu$-calculus is self-minimizing
  - Intuition
  - Background
  - proof

→ Our thorough checking algorithm

→ Conclusion and future work
Thorough Checking Algorithm

\[
\text{ThoroughCheck}(M, \varphi) \\
(1): \text{ if } (v := \text{ModelCheck}(M, \varphi)) \neq \text{Maybe} \rightarrow \text{return } v \\
(2): \text{ if } \text{IsSelfMinimizing}(M, \varphi) \rightarrow \text{return } \text{Maybe} \\
(3): \text{return } \text{ModelCheck}(M, \text{SemanticMinimization}(\varphi))
\]
Self-Minimization

IsSelfMinimizing($M, \varphi$)
(i) if $M$ is a PKS or an MixTS and $\varphi$ is monotone
    return true
(ii) if $M$ is an HTS and $\varphi$ is disjunctive
     return true
(iii) return false

Example

Property $AGq \land A[p \lor \neg q]$ over
- PKSs and MixTSs violates condition (i)
- HTSs violates condition (ii)

Thus, $AGq \land A[p \lor \neg q]$ is not self-minimizing
Semantic Minimization

Semantic Minimization(φ)
(i) convert φ to its disjunctive form φ^v
(ii) replace all special conjunctions in φ^v containing p and ¬p with False
(iii) return φ^v

→ Example: semantic minimization of AGq ∧ A[p U ¬q]

دخل

Step (i) AGq ∧ A[p U ¬q] \rightarrow A[p ∧ q U q ∧ ¬q ∧ AXAGq]
Step (ii) A[p ∧ q U q ∧ ¬q ∧ AXAGq] \rightarrow A[p ∧ q U False]
Complexity

\[ \text{ThoroughCheck}(M, \varphi) \]
(1):  \(\text{if } (v := \text{ModelCheck}(M, \varphi)) \neq \text{Maybe} \)
\hspace{1cm} \text{return } v \\
(2):  \text{if } \text{IsSelfMinimizing}(M, \varphi) \\
\hspace{1cm} \text{return } \text{Maybe} \\
(3):  \text{return } \text{ModelCheck}(M, \text{SemanticMinimization}(\varphi)) \\

→ Step (1)

\(\text{Model checking } \mu\text{-calculus formulas } O((|\varphi| \cdot |M|)^{\lfloor d/2 \rfloor + 1})\)

→ Step (2)

\(\text{Self-minimization check is linear in the size of formulas}\)

→ Step (3)

\(\text{Semantic minimization } O((2^{O(|\varphi|)} \cdot |M|)^{\lfloor d/2 \rfloor + 1})\)
Conclusion

→ Studied thorough checking over partial models

▷ An automata-based characterization for thorough checking

▷ Simple and syntactic self-minimization checks
  ➢ Grammars for identifying self-minimizing formulas in CTL

▷ A semantic-minimization procedure
Future Work

→ Studying the classes of formulas for which thorough checking is cheap
  ≪linear in the size of models≫

→ Identifying commonly used formulas in practice that are self-minimizing
Thank You!
Questions?