Model Checking Vs. Generalized Model Checking: Semantic Minimizations for Temporal Logics

Patrice Godefroid  
Bell Laboratories

Michael Huth  
Imperial College
Verification via Automatic Abstraction

• Implemented in software model checkers like SLAM, BLAST,…

• Traditional iterative abstraction procedure:
  1. Abstract: generate a finite abstraction $A$ from the concrete program $C$ such that $A$ simulates $C$ (using predicate abstraction, theorem proving)
  2. Check: given any universal temporal-logic formula $\phi$, compute $[A \models \phi]$:
     if $[A \models \phi] = \text{true}$, then return true (we then know $[C \models \phi] = \text{true}$)
  3. Refine: (checking of $A$ is inconclusive) refine $A$, then go to Step 1
     • Ex: with predicate abstraction, add predicates to refine the model

• Limitations:
  – Restricted to universal properties (no existential properties)
  – $[A \models \phi] = \text{false}$ does not imply anything about $C$
  – Could the analysis be more precise for an acceptable cost?
A Solution: 3-Valued Models and Logics

- Richer models A that distinguish what is true/false/unknown of C
  - Example: “partial Kripke structure” [Fitting92, Bruns-G99]
  - Other example: “Modal Transition System” (may/must trans.) [Larsen+88]
  - These formalisms are all equally expressive [G-Jagadeesan03]

- Reasoning about 3-valued models requires 3-valued temporal logic
  - Ex: \([(M,s) \models p] = \text{true}, \quad [(M,s) \models AXp] = \text{false}, \quad [(M,s) \models EXp] = \bot\]

- Complexity of 3-valued MC = complexity of MC [Bruns-G00]
New Abstract-Check-Refine Process

• New procedure for automatic abstraction: (3 improvements)
  [G-Jagadeesan02, G-Huth-Jagadeesan01,…]

  1. Abstract: generate a 3-valued abstraction \( A \) from the concrete program \( C \) that preserves \( true, false, unknown \) properties of \( C \) (same cost)
     (Formally, \( A \preceq C \) where \( \preceq \) is the abstraction preorder on 3-valued models)

  2. Check: given any temporal-logic formula \( \phi \),
     • (3-valued model checking) compute \([A \models \phi] : \) (same cost)
       if \([A \models \phi] = true\) or \(false\), then return \(true\) or \(false\) (respectively)
     • (generalized model checking)
       if there is no concretization \( C \) of \( A \) such that \( C \) satisfies \( \phi \), return false
       if there is no concretization \( C \) of \( A \) such that \( C \) violates \( \phi \), return true

  3. Refine: (checking of \( A \) is inconclusive) refine \( A \), then go to Step 1
Generalized Model Checking (GMC)

• Definition: [Bruns-G00]
  Given a program abstraction A and a temporal logic formula φ, does there exist a concretization C of A such that C satisfies φ?

• GMC is a generalization of both
  – Satisfiability (SAT)
  – Model Checking (MC)

• GMC can be more expensive than MC (since it includes SAT)
  – in |φ| (but worst-case and φ is usually short) [Bruns-G00]
  – in |A| (quadratic) but linear for persistence (incl. safety) properties [G-J02]

• GMC can also be more precise than MC…
Example where GMC is more precise than MC

Program $P()$ {
    int $x, y = 1, 0$;
    $x, y = 2*f(x), f(y)$;
    $x, y = 1, 0$;
}

Property “(eventually $y$ is odd) and (always, $x$ is odd or $y$ is even)”

is represented by the LTL formula $\phi = F(q) \land G(p \lor \neg q)$

$MC(A, \phi) = \bot$ …but $GMC(A, \phi) = false$!
How often is GMC more precise than MC?

• Motivation for this paper!

• More generally, how to reduce GMC(A,φ) to MC(A,φ’)? (independently of A)

• φ’ is called a semantic minimization of φ; problem already studied for Propositional Logic (PL) [Blamey80, Reps-Loginov-Sagiv02]

• Theorem: [Blamey80] for all φ ∈ PL, there is a sem min φ’ ∈ PL

• Note: |φ’| can be much larger than |φ|! (since computing φ’ is as hard as GMC, which is as hard as SAT, hence NP-hard for PL)

• What about temporal logics?
Semantic Minimizations for PML and $\mu L$

- Propositional Modal Logic (PML): $\phi ::= p \mid \neg \phi \mid \phi \land \phi \mid \text{EX} \phi$

- **Theorem:** For all $\phi \in \text{PML}$, there is a sem min $\phi' \in \text{PML}$

- **Proof idea:**
  - build a tree automaton $A^3(\phi)$ that accepts a 3-valued labeled tree $T^3$ iff there exists a 2-valued tree $T$ such that $T^3 \leq T$ and $T$ satisfies $\phi$
  - Translate $A^3(\phi)$ back into a PML formula $\phi'$ of modal depth $O(|\phi|)$ (possible because $A^3(\phi)$ cannot distinguish trees at depths greater than $|\phi|$)

- Modal mu-calculus ($\mu L$): $\phi ::= p \mid \neg \phi \mid \phi \land \phi \mid \text{EX} \phi \mid Z \mid \mu Z.\phi$

- **Theorem:** For all $\phi \in \mu L$, there is a sem min $\phi' \in \mu L$

- **Proof idea:** similar as above
Semantic Minimizations for CTL, LTL and CTL*

- For CTL, LTL and CTL*, GMC is PTIME-hard in $|A|$ [G-03]…

- …while MC is known to be NLOGSPACE-complete in $|A|$

- … therefore, GMC($A, \phi$) cannot be reduced to MC($A, \phi'$) unless NLOGSPACE = PTIME !

- **Theorem:**
  The sem min of the CTL formula $\phi = A[(EX q_1)U(q_1 \rightarrow q_2)]$ is the $\mu$L formula $\phi' = \mu Z_1.(q_1 \rightarrow q_2) \lor [\mu Z_2.AX Z_1 \land EX(q_1 \land (q_2 \lor Z_2))],$ which is not expressible in CTL*

- Proof idea: $\phi'$ is obtained with previous construction of $A^3(\phi)$; CTL* cannot express unbounded alternation (and/or graph reach.)
Semantic Minimizations: Summary

- PL, PML and $\mu$L are closed under semantic minimizations
- while CTL, LTL and CTL* are not
- But for all $\phi \in$ CTL* (thus CTL,LTL), there is a sem min $\phi' \in \mu$L
- Note on first-order logic over binary relations (FOL): since SAT is undecidable, so is GMC, while MC is decidable. Thus, sem min cannot exist for all formulas of FOL.
Self-Minimization

- When do we have $\phi' = \phi$? Such $\phi$ are called self-minimizing.

- For any self-minimizing formula $\phi$, $\text{GMC}(A,\phi) = \text{MC}(A,\phi)$ that is, GMC and MC have the same precision.

- Checking for self-minimization semantically:
  - Compare the automaton $A^3(\phi)$ with an automaton $A^3(\models \phi)$ that accepts exactly all 3-valued labeled trees $T^3$ such that $[T^3 \models \phi] = \text{true}$.
  - By construction, $L(A^3(\phi)) \subseteq L(A^3(\models \phi))$.
  - If $L(A^3(\models \phi)) \subseteq L(A^3(\phi))$, then $\phi$ is self-minimizing.
  - If $\phi \in \mu L$, these automata are parity tree automata and $A^3(\phi)$ can be of size exponential in $|\phi|$.
  - Thus, such a semantic (automata-based) check is precise but expensive!
Syntactic Tests for Self-Minimization

- Checking for self-minimization **syntactically**:
  - linear in |ϕ| but incomplete (less precise than semantic check)

- Example of **sufficient** condition: (*)
  Any formula that does not contain any atomic proposition in mixed polarity (in its negation normal form) is self-minimizing.

- Many frequently-used formulas satisfy this condition:
  - (absence) AG(q → AG(¬p)),  (universality) AG(q → AG p)
  - (existence) EF p,  (response) AG(p → AF q),  etc.

- (*) is not necessary: (¬q₁ ∨ q₂) ∧ (¬q₂ ∨ q₁) is self-minimizing
Temporal Patterns of Self-Minimization

• In the paper, we present grammars to identify syntactically self-minimizing formulas

(details omitted here)

• Related work: study of temporal logics and normal forms for which satisfiability is efficiently decidable
  – [Emerson-Evangelist-Srinivasan90, Janin-Walukiewicz95, Demri-Schnoebelen99, Henzinger-Kupferman-Majumdar03, etc.]
Conclusions

- Study of precision of MC vs. GMC for verification via abstraction
- More generally, study of how to reduce GMC(A,φ) to MC (A,φ’)
- φ’ is called a semantic minimization of φ
- Like PL, PML and µL are closed under semantic minimizations, but CTL, LTL and CTL* are not
- Checking for self-minimizing formulas:
  - semantically (precise but expensive automata-based algorithms)
  - syntactically (sufficient conditions only, linear in |φ|)
- Good news: in practice, many formulas are self-minimizing, and MC is as precise as GMC for those