MODEL CHECKING SAFETY PROPERTIES

Orna Kupferman
Hebrew University

Moshe Y. Vardi
Rice University
Weizmann Inst.

MODEL CHECKING: SAFETY PROPERTIES

Orna Kupferman
Hebrew University

Moshe Y. Vardi
Rice University
Weizmann Inst.

Model Checking Linear Properties

M: system

$L(M)$: linear traces of M

φ: specification (LTL, automata)

$L(\varphi)$: linear traces satisfying φ

Correctness: $L(M) \subseteq L(\varphi)$

(Language containment)

$A_{\neg\varphi}$: Tester for φ (ω-automaton)

$L(A_{\neg\varphi})$: linear traces violating φ

Correctness:

$L(M) \subseteq L(\varphi) \Leftrightarrow L(M \,||\, A_{\neg\varphi}) = \emptyset$

(Language emptiness)

Invariance checking

φ: always p, where p is a state property

$A_{\neg\varphi}$: 

$L(M \| A_{\neg\varphi}) \neq \emptyset$ if M contains a path from an initial state to a state satisfying $\neg p$.

Forward MC: search from Init forward

Backward MC: search from $\neg p$ backward

Safety Properties

$L \subseteq \Sigma^\omega$: ω-language

Dfn: $x \in \Sigma^*$ is a <u>bad prefix</u> for L if $xy \notin L$ for all $y \in \Sigma^\omega$.

L is a <u>safety</u> language if each $u \in L$ has a bad prefix.

[Alpern + Schneider, 1985]

φ: safety property if $L(\varphi)$ is a safety language.

Examples:

- always $(p \rightarrow$ eventually $q)$: No
- always $(p \rightarrow$ next$^{15}\, q)$: Yes

Syntactic Safety

Dfn: φ is syntactically safe if its negation-normal form (i.e., negation pushed down to atomic propositions) is free of Until and Eventually.

Fact: [Sistla, 94]
syntactically safe formulas are safe.

Example: always$(p \rightarrow \text{next}^{15} q)$

M-C Safety Properties: Idea

bad-prefix(φ) = $\{x : x$ is a bad prefix for $L(\varphi)\}$

Note: bad-prefix(φ) $\subseteq \Sigma^*$ (finite words)

$B_{\neg\varphi}$: automaton for bad-prefix(φ)

Fact: $L(M \| A_{\neg\varphi}) \neq \emptyset$ iff $L(M \| B_{\neg\varphi}) \neq \emptyset$

$L(M \| A_{\neg\varphi}) \neq \emptyset$: ω-words

$L(M \| B_{\neg\varphi}) \neq \emptyset$: finite words

$L(M \| B_{\neg\varphi}) \neq \emptyset$ iff there is a path from an initial state of $M \| B_{\neg\varphi}$ to an accepting state:

reduction to invariance checking

Büchi Automata

$A = (\Sigma, S, S_0, \rho, F)$

Σ: alphabet (2^{Prop})

S: states

$S_0 \subseteq S$: initial states

$\rho : S \times \Sigma \to 2^S$: transition function

$F \subseteq S$: accepting states

Input word: $a_0, a_1, a_2, \ldots$

Accepting run: $\pi_0, \pi_1, \pi_2, \ldots$

$\pi_0 \in S_0$, $\pi_{i+1} \in \rho(\pi_i, a_i)$, F visited i.o.

Notation: $A^T = (\Sigma, S, T, \rho, F)$: T as initial states

Proviso: $L(A^T) \neq \phi$ for all $\pi \in S$.

Implementation: Compute $S' = \{\pi : L(A^T) \neq \phi\}$ by M-C.

M-C safety properties: Implementation

φ: specification

step 1: check if φ is a safety property

Sistla, 94:
- PSPACE-complete
- reduces to model checking

Let $A_\varphi = (\Sigma, S, S_0, \rho, F)$ be such that $L(A_\varphi) = L(\varphi)$ (tester for φ).

Let $A^\ell_\varphi = (\Sigma, S, S_0, \rho, S)$ (no acceptance condition).

Theorem: φ is a safety property iff $L(A^\ell_\varphi) = L(\varphi)$.

Implementation: check $A^\ell_\varphi \models \varphi$.

M-C Safety Properties: Implementation

Step 2: construct $B_{\neg\varphi}$ (automaton for bad-prefix(φ))

Let $A_{\neg\varphi} = (\Sigma, S, S_0, \rho, F)$ (tester for $\neg\varphi$)

$B_{\neg\varphi} = (\Sigma, 2^S, \{S_0\}, \rho', U)$ (subset construction)

$$\rho'(Q, a) = \bigcup_{q \in Q} \rho(q, a)$$

$$U = \{Q : L(A^Q_{\neg\varphi}) = \Sigma^\omega\}$$ (universal sets)

Notes:

1. Universality test: via M.C.
2. $B_{\neg\varphi}$ is deterministic: (can be minimized using MoDA techniques.
3. $B_{\neg\varphi}$ can be used as a simulation checker.

Complexity Issues

- $|\varphi| = n$
- $|A_{\neg\varphi}| = 2^{O(n)}$: unavoidable!
- $|B_{\neg\varphi}| = 2^{2^{O(n)}}$: can we do better?

Theorem: No. The blow-up is unavoidable!

But:

Symbolically -

- $|\varphi| = n$
- $|A_{\neg\varphi}| = O(n)$
- $|B_{\neg\varphi}| = 2^{O(n)}$: a single exponential blow-up

Who's Afraid of the Subset Construction

Subset construction: We do it every day!

$Front^0(M) = Init(M)$

$Front^i(M) = \{$ states reachable from $Init(M)$ in i steps $\}$

This is subset construction for M.

$Front^0(M \| A_{\neg\varphi}) = Init(M) \times Init(A_{\neg\varphi})$

$Front^i(M \| A_{\neg\varphi}) = \{$ states reachable from $Init(M) \times Init(A_{\neg\varphi})$ in i steps $\}$

Fix state w of M:

$\{q : (w, q) \in Front^i(M \| A_{\neg\varphi})\} =$

$\{$ states of $A_{\neg\varphi}$ reachable via a path in M from $Init(M)$ to $w\}$

This is subset construction for $A_{\neg\varphi}$

Implicit Subset Construction

step 1: perform forward μ-c on $M \| A_{\neg\varphi}$.

step 2: If $Front^i(M \| A_{\neg\varphi})$ gets too large, search for a state w of M s.t. $\{q : (w,q) \in Front^i(M \| A_{\neg\varphi})\}$ is universal for $A_{\neg\varphi}$.

semi-exhaustive search: underapprox. $Front^i(M \| A_{\neg\varphi})$ — drop states of M, but keep all states of $A_{\neg\varphi}$.

simulation: drop all but one state of M, but keep all states of $A_{\neg\varphi}$.

Explicit vs. Implicit Subset Construct

Two options:

1. $B_{\neg\varphi}$ is constructed before M-C.
   - +: one-time construction
   - +: deterministic
   - −: exponentially many vars

2. $B_{\neg\varphi}$ is constructed (implicitly) during M-C.
   - −: repeated construction
   - +: no explicit construction
   - +: linearly many vars

Tradeoff: long and skinny bdds vs. short and fat bdds.

Can We Do Better?

bad-prefix(φ) = $\{x : x$ is a bad prefix for $L(\varphi)\}$

$B_{\neg\varphi}$: tight for bad-prefix(φ),

i.e., $L(B_{\neg\varphi}) =$ bad-prefix(φ)

Suppose: $L(B) \subseteq$ bad-prefix(φ)

Defn: B is fine for bad-prefix(φ) if

every $w \in L(\neg\varphi)$ has a prefix in $L(B)$

Idea: Search $M \| B$ rather than $M \| B_{\neg\varphi}$

Facts:
- $|B| \leq |B_{\neg\varphi}|$ (tight $\Rightarrow$ fine)
- Fine automata can be exponentially smaller than tight automata.

Open: Is there a construction of fine automata that is polynomial in $A_{\neg\varphi}$

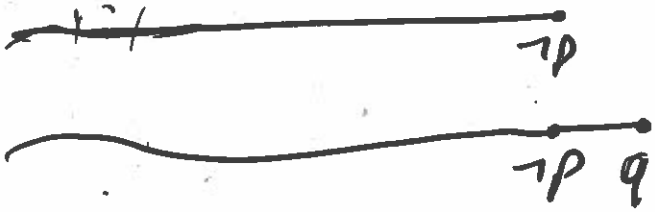
Safety Formulas

Examples:

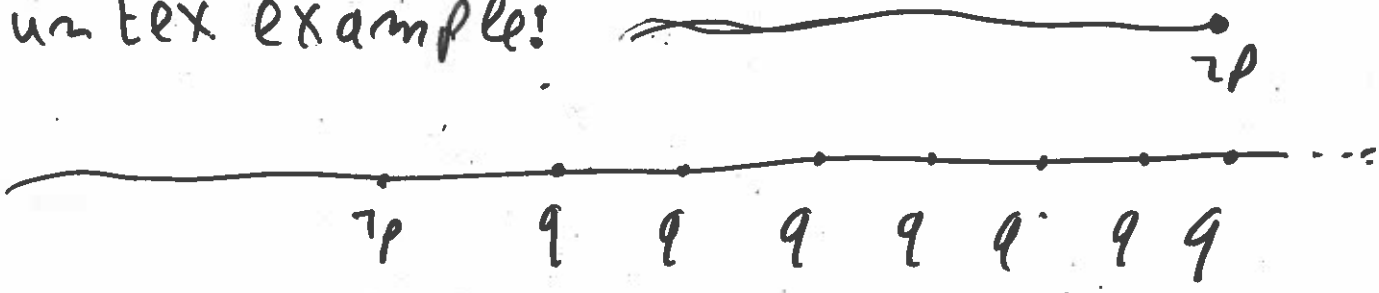
- $G\,p \overset{\neg}{\longmapsto} F\neg p$

  counterexample: 

- $G[p \vee (Xq \wedge X\neg q)] \overset{\neg}{\longmapsto} F[\neg p \wedge (X\neg q \vee Xq)]$

  counterexamples: 

- $G[p \vee (FGq \wedge FG\neg q)] \overset{\neg}{\longmapsto} F[\neg p \wedge (GF\neg q \vee GFq)]$

  Countex example: 

What kind of counterexample does the user expect?

Informative Witnesses

Finite witness: $\pi = \pi_0, \pi_1, \ldots, \pi_n$

$\pi_i \subseteq 2^{Prop}$

Dfn: π is an informative witness for ψ if there is a mapping $\xi : \{0, \ldots, n\} \to 2^{cl(\psi)}$

1. $\psi \in \xi(0)$
2. $\xi(n) = \emptyset$
3. If $p \in \xi(i)$, then $p \in \pi_i$
4. If $(\alpha \wedge \beta) \in \xi(i)$, then $\alpha \in \xi(i)$ and $\beta \in \xi(i)$
5. If $X\alpha \in \xi(i)$, then $\alpha \in \xi(i+1)$
6. If $G\alpha \in \xi(i)$, then $\alpha \in \xi(i)$ and $G\alpha \in \xi(i+1)$

⋮

Example: $GFp \vee FG\neg p$

has no informative finite witness

Classification

Recall: φ is a safety property if each $w \in L(\neg\varphi)$ has a bad prefix

Classification:

1. φ is intentionally safe if all bad prefixes for φ are informative. (for $\neg\varphi$)
   (e.g., GP)
2. φ is accidentally safe if each $w \in L(\neg\varphi)$ has an informative bad prefix. (e.g. $G(P \vee (Xq \wedge X\neg q))$)
3. φ is pathologically safe if there is some $w \in L(\neg\varphi)$ that does not have a bad informative prefix.
   (e.g., $G(P \vee (FGq \wedge FG\neg q))$)

Classification: Algorithms

Theorem: Deciding whether a given formula is pathologically safe is PSPACE-complete.

Proof: Via alternating automata.

Open: Deciding whether a given formula is accidentally safe but not intentionally safe.

But: Every syntactically safe formula is intentionally or accidentally safe.

Classification: Application

Theorem: For a given φ, we can construct an automaton C_φ s.t.

- If φ is intentionally safe, then C_φ is tight (i.e., accepts all bad prefixes)
- If φ is accidentally safe, then C_φ is fine (i.e., accepts "enough" bad prefixes).

Methodology:

1. Search $M \| C_\varphi$ (a bug found is real).
2. If $L(M \| C_\varphi) = \emptyset$, then check if φ is pathologically safe.
3. If φ is P.S., consult user, else M is correct.

In Conclusion

Open questions:

Theoretical:

1. Do fine automata of exponential size alway exist?
2. Characterize intentionally safe formulas

Practical:

1. Long and skinny BDDs vs. short and fat BDDs.

Model Checking Linear Properties

M: system

$L(M)$: linear traces of M

φ: specification (LTL, automata)

$L(\varphi)$: linear traces satisfying φ

Correctness: $L(M) \subseteq L(\varphi)$

(Language containment)

$A_{\neg\varphi}$: Tester for φ (W- automata)

$L(A_{\neg\varphi})$: linear traces violating φ

Correctness:

$L(M) \subseteq L(\varphi) \Leftrightarrow L(M \| A_{\neg\varphi}) = \emptyset$

(Language emptiness)

Invariance checking

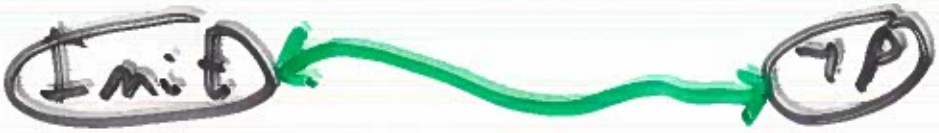
φ: always P, where P is a state property

$A_{\neg\varphi}$: 

$L(M \| A_{\neg\varphi}) \neq \emptyset$ if M contains a path from an initial state to a state satisfying $\neg P$.

Forward MC: search from Init forward

Backward MC: search from $\neg P$ backward



Safety properties

$L \subseteq \Sigma^\omega$: ω-language

Dfn: $x \in \Sigma^*$ is a bad prefix for L if $xy \notin L$ for all $y \in \Sigma^\omega$.

L is a safety language if each $u \notin L$ has a bad prefix.

[Alpern + Schneider, 1985]

φ: safety property if $L(\varphi)$ is a safety language.

Examples:

- always ($p \rightarrow$ eventually q): No
- always ($p \rightarrow$ next^{15} q): Yes

Syntactic Safety

Dfn: φ is syntactically safe if its negation-normal form (i.e., negation pushed down to atomic propositions) is free of Until and Eventually.

Fact: [Sistla, 94]

Syntactically safe formulas are safe.

Example: $always(p \rightarrow next^{15} q)$

M-C Safety Properties: Idea

bad-prefix(φ) = $\{x : x$ is a bad prefix for $L(\varphi)\}$

Note: bad-prefix(φ) $\subseteq \Sigma^*$ (finite words)

$B_{\neg\varphi}$: automaton for bad-prefix(φ)

Fact: $L(M \| A_{\neg\varphi}) \neq \emptyset$ iff $L(M \| B_{\neg\varphi}) \neq \emptyset$

ω-words — finite words

$L(M \| B_{\neg\varphi}) \neq \emptyset$ iff there is a path from an initial state of $M \| B_{\neg\varphi}$ to an accepting state:

reduction to invariance checking

Büchi Automata

$A = (\Sigma, S, S_0, \rho, F)$

Σ: alphabet (2^{Prop})

S: states

$S_0 \subseteq S$: initial states

$\rho : S \times \Sigma \to 2^S$: transition function

$F \subseteq S$: accepting states

Input word: $a_0, a_1, a_2, \ldots$

Accepting run: $\lambda_0, \lambda_1, \lambda_2, \ldots$

$\lambda_0 \in S_0$, $\lambda_{i+1} \in \rho(\lambda_i, a_i)$, F visited i.o.

Notation: $A^T = (\Sigma, S, T, \rho, F)$: T as initial states

Proviso: $L(A^\lambda) \neq \emptyset$ for all $\lambda \in S_0$.

Implementation: compute $S' = \{\lambda : L(A^\lambda) \neq \emptyset\}$ by M-C.

M-C Safety properties: Implementation

φ: specification

step 1: check if φ is a safety property

Sistla, 94:
- PSPACE-complete
- reduces to model checking

Let $A_\varphi = (\Sigma, S, s_0, \rho, F)$ be such that $L(A_\varphi) = L(\varphi)$ (tester for φ).

Let $A^\ell_\varphi = (\Sigma, S, s_0, \rho, S)$ (no acceptance condition)

Theorem: φ is a safety property iff $L(A^\ell_\varphi) = L(\varphi)$.

Implementation: check $A^\ell_\varphi \models \varphi$.

M-C Safety Properties: Implementation

step 2: construct $B_{\neg\varphi}$ (automaton for bad-prefix(φ))

Let $A_{\neg\varphi} = (\Sigma, S, S_0, \rho, F)$ (tester for $\neg\varphi$)

$B_{\neg\varphi} = (\Sigma, 2^S, \{S_0\}, \rho', U)$ (subset construction)

$$\rho'(Q, a) = \bigcup_{q \in Q} \rho(q, a)$$

$$U = \{Q : L(A^Q_{\neg\varphi}) = \Sigma^\omega\}$$ (universal sets)

Notes:

1. Universality test: via M.C.
2. $B_{\neg\varphi}$ is deterministic: (can be minimized using MONA techniques.
3. $B_{\neg\varphi}$ can be used as a simulation checker.

Complexity Issues

- $|\varphi| = n$
- $|A_{\neg\varphi}| = 2^{O(n)}$: unavoidable!
- $|B_{\neg\varphi}| = 2^{2^{O(n)}}$: can we do better?

Theorem: No. The blow-up is unavoidable!

But:

Symbolically -

- $|\varphi| = n$
- $|A_{\neg\varphi}| = O(n)$
- $|B_{\neg\varphi}| = 2^{2^{O(n)}}$: a single exponential blow-up

Who's Afraid of the Subset Construction

Subset construction: we do it every day!

$Front^0(M) = Init(M)$

$Front^i(M) = \{$ states reachable from $Init(M)$ in i steps $\}$

This is subset construction for M.

$Front^0(M \| A_{\neg\varphi}) = Init(M) \times Init(A_{\neg\varphi})$

$Front^i(M \| A_{\neg\varphi}) = \{$ states reachable from $Init(M) \times Init(A_{\neg\varphi})$ in i steps $\}$

Fix state w of M:

$\{q : (w, q) \in Front^i(M \| A_{\neg\varphi})\} =$

$\{$ states of $A_{\neg\varphi}$ reachable via a path in M from $Init(M)$ to $w\}$

This is subset construction for $A_{\neg\varphi}$

Implicit subset construction

step 1: perform forward M-C on $M \| A_{\neg\varphi}$.

step 2: If $Front^i(M \| A_{\neg\varphi})$ gets too large, search for a state w of M s.t. $\{q : (w,q) \in Front^i(M \| A_{\neg\varphi})\}$ is universal for $A_{\neg\varphi}$.

semi-exhaustive search: underapprox.

$Front^i(M \| A_{\neg\varphi})$ – drop states of M, but keep all states of $A_{\neg\varphi}$.

simulation: drop all but one state of M, but keep all states of $A_{\neg\varphi}$.

Explicit vs. Implicit Subset Construct.

Two options:

1. $B_{\neg\varphi}$ is constructed before M-C.

   +: one-time construction

   +: deterministic

   −: exponentially many vars

2. $B_{\neg\varphi}$ is constructed (implicitly) during M-C.

   −: repeated construction

   +: no explicit construction

   +: linearly many vars

Tradeoff: long and skinny bdd's vs. short and fat bdd's.

Can We Do Better?

bad-Prefix(φ) = $\{x : x$ is a bad prefix for $L(\varphi)\}$

$B_{\neg\varphi}$: tight for bad-Prefix(φ),

i.e., $L(B_{\neg\varphi}) =$ bad-Prefix(φ)

Suppose: $L(B) \subseteq$ bad-Prefix(φ)

Dfn: B is fine for bad-Prefix(φ) if

every $w \in L(\neg\varphi)$ has a prefix in $L(B)$

Idea: Search $M \| B$ rather than $M \| B_{\neg\varphi}$

Facts:
- $|B| \leq |B_{\neg\varphi}|$ (tight $\Rightarrow$ fine)
- Fine automata can be exponentially smaller than tight automata.

Open: Is there a construction of fine automata that is polynomial in $A_{\neg\varphi}$?

Safety Formulas

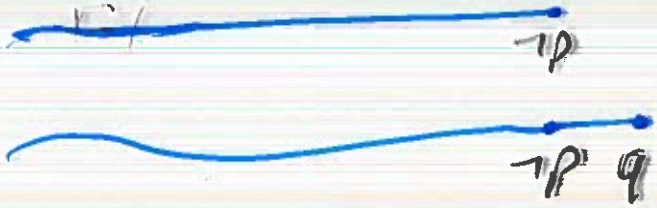
Examples:

- $G\,p \;\overset{\neg}{\longmapsto}\; F\,\neg p$

  counterexample:

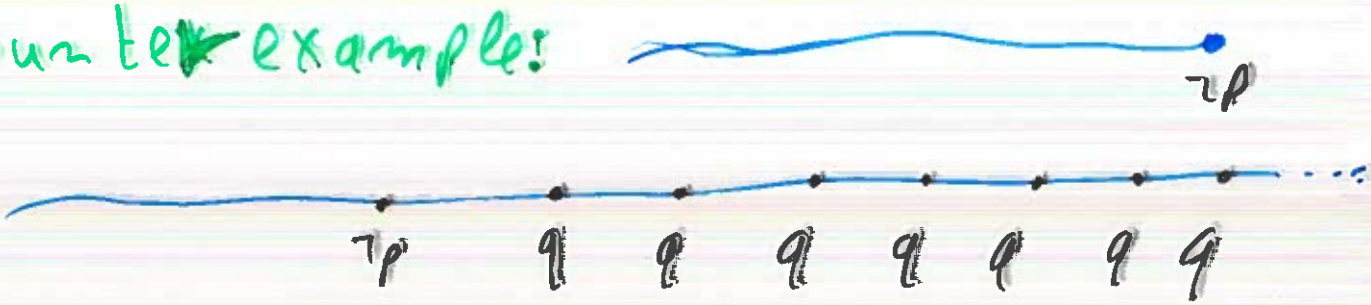
- $G[p \vee (Xq \wedge X\neg q)] \overset{\neg}{\longmapsto} F[\neg p \wedge (X\neg q \vee Xq)]$

  counterexamples:



- $G[p \vee (FGq \wedge FG\neg q)] \overset{\neg}{\longmapsto} F[\neg p \wedge (GF\neg q \vee GFq)]$

  counterexample:



What kind of counterexample does the user expect?

Informative Witnesses

Finite witness: $\Pi = \Pi_0, \Pi_1, \ldots, \Pi_m$

$\Pi_i \subseteq 2^{Prop}$

Dfn: Π is an informative witness for Ψ if there is a mapping $\xi : \{0, \ldots, m\} \to 2^{cl(\Psi)}$

1. $\Psi \in \xi(0)$
2. $\xi(m) = \emptyset$
3. If $p \in \xi(i)$, then $p \in \Pi_i$
4. If $(\alpha \wedge \beta) \in \xi(i)$, then $\alpha \in \xi(i)$ and $\beta \in \xi(i)$
5. If $X\alpha \in \xi(i)$, then $\alpha \in \xi(i+1)$
6. If $G\alpha \in \xi(i)$, then $\alpha \in \xi(i)$ and $G\alpha \in \xi(i+1)$

⋮

Example: $GFp \vee FG\neg p$ has no informative finite witness

Classification

Recall: φ is a safety property if each $w \in L(\neg\varphi)$ has a bad prefix

Classification:

1. φ is intentionally safe if all bad prefixes for φ are informative. (for $\neg\varphi$)
   (e.g., $G\,p$)
2. φ is accidentally safe if each $w \in L(\neg\varphi)$ has an informative bad prefix. (e.g. $G(p \vee (Xq \wedge X\neg q))$)
3. φ is pathologically safe if there is some $w \in L(\neg\varphi)$ that does not have a bad informative prefix.
   (e.g., $G(p \vee (FGq \wedge FG\neg q))$)

Classification: Algorithms

Theorem: Deciding whether a given formula is pathologically safe is PSPACE-complete.

Proof: Via alternating automata.

Open: Deciding* whether a given formula is accidentally safe but not intentionally safe.

But: Every syntactically safe formula is intentionally or accidentally safe.

*: in PSPACE

Classification: Application

Theorem: For a given φ, we can construct an automaton C_φ s.t.

- If φ is intentionally safe, then $C_{\neg\varphi}$ is tight (i.e., accepts all bad prefixes)
- If φ is accidentally safe, then $C_{\neg\varphi}$ is fine (i.e., accepts "enough" bad prefixes).
- $|C_{\neg\varphi}|$ is exponential in $|\varphi|$.

Methodology:

1. Search $M \| C_{\neg\varphi}$ (a bug found is real).
2. If $L(M \| C_{\neg\varphi}) = \emptyset$, then check if φ is pathologically safe.
3. If φ is P.S., consult user, else M is correct.