FASTER TEMPORAL REASONING FOR INFINITE-STATE PROGRAMS

Heidy Khlaaf$^1$  Byron Cook$^{1,2}$  Nir Piterman$^3$

$^1$University College London  
$^2$Microsoft Research  
$^3$University of Leicester
BRIEF OVERVIEW

- A new symbolic model checking procedure for CTL verification of infinite-state programs.

- Counterexample-guided precondition synthesis strategy to compute location(PC) specific preconditions.

- Existing tools not scalable. We propose:
  
  - Compositional strategy via exploiting the natural decomposition of the program’s state space.
  
  - Performance improvement and scalability!
CTL - A REFRESHER

- Temporal logic reasoning about sets of states.
- Implemented via formula structure.
- Reasoning about non-deterministic (branching) programs.
- Termination - AFAX false, Non-Termination - EGEX true, AG - Safety
- These operators can be nested to generate complex liveness and safety properties.
- Used to uncover bugs in device drivers, operating systems, servers, etc.
A \( \varphi \) – All: \( \varphi \) has to hold on all paths starting from all initial states.

E \( \varphi \) – Exists: there exists at least one path starting from all initial states where \( \varphi \) holds.

G \( \varphi \) – Globally: \( \varphi \) has to hold on the all states along a path.

F \( \varphi \) – Finally: \( \varphi \) eventually has to hold.

Other operators include X \( \varphi \) – Next, \( \varphi_1 \lor \varphi_2 \) – Until, \( \varphi_1 W \varphi_2 \) – Weak until.
Verify $\text{EF } y > z$

**INTUITION**

- $\tau_1: y' = 0$
  - $x' = 10$
- $\tau_2: y \leq x$
  - $y' = y + 1$
- $\tau_3: z' = x$
- $\tau_4: y > z$
  - $y' = 0$
- $\tau_5: x > 0$
  - $x' = x - 1$
The set of states satisfying EF y > z before a program command is very often the same as the set of states respecting EF y > z after the command.

**INTUITION**

- The set of states satisfying EF y > z before a program command is very often the same as the set of states respecting EF y > z after the command.
We can infer whether a command is likely to affect the truth of $EF \ y > z$.

- From one counterexample, we use precondition synthesis to infer the pre-image of all locations within a counterexample.
EXAMPLE (PT. 1)

- Prove the CTL property $\text{AGEF } y = 1$. 

\[
\begin{align*}
\rho_1 : & \quad x' = * \\
& \quad y' = 0 \\
\rho_2 : & \quad x \leq 0 \\
& \quad x' = x + 1 \\
\rho_3 : & \quad x \leq 0 \\
\rho_4 : & \quad x > 0 \\
\rho_5 : & \quad y' = 1
\end{align*}
\]
EXAMPLE (PT. 1)

- Recurse over the structure of the CTL formula.
- Find $\varphi$ such that $\text{AG} \varphi$ holds, and $\varphi \models \text{EF} y = 1$.
- $\varphi(\varphi)$ takes the form $\bigwedge_{i} (\text{pc} = l_{i} \Rightarrow \varphi(l_{i}, \varphi))$. 

\[\begin{array}{l}
\rho_{1} : \ x' = * \\
y' = 0
\end{array}\]
\[\begin{array}{l}
\rho_{2} : \ x \leq 0 \\
x' = x + 1
\end{array}\]
\[\begin{array}{l}
\rho_{3} : \ x \leq 0
\end{array}\]
\[\begin{array}{l}
\rho_{4} : \ x > 0
\end{array}\]
\[\begin{array}{l}
\rho_{5} : \ y' = 1
\end{array}\]
EXAMPLE (PT. 1)

- Verify the universal dual of the existential property and seek a set of counterexamples to serve as witnesses.

- Transform the program for the property $\varphi = EF \ y = 1$ using its dual $\text{AG} \ y \neq 1$. 
EXAMPLE (PT. 1)

- Initially $\varnothing(\varphi) = \text{false}$ as only failures to proving $\text{AG } y \neq 1$ imply that there exists a witness such that $\text{EF } y = 1$. 

\[\begin{align*}
\rho_1 &: \quad x' = * \quad y' = 0 \\
\rho_2 &: \quad x \leq 0 \land y' = 0 \\
\rho_3 &: \quad x \leq 0 \land y \neq 1 \\
\rho_4 &: \quad x > 0 \land y \neq 1 \\
\rho_5 &: \quad y \neq 1 \\
\rho_6 &: \quad y = 1 \\
\rho_7 &: \quad y = 1
\end{align*}\]
EXAMPLE (PT. 1)

- Use a safety prover to check reachability of ERR, and begin from $l_1$.
- For every reachable location $l \in L$ in CEX$_1$, we compute a pre-image using the suffix of CEX$_1$ from $l$ onwards.
EXAMPLE (PT. 1)

- \( \text{pre}(\text{CEX}_1) = y = 0 \).
- When a new counterexample is discovered, we refine
  \( \varnothing \langle l, \varphi \rangle \) resulting in
  \( \varnothing \langle l, \varphi \rangle = \bigvee_{n \in \mathbb{N}} \text{pre}(\text{CEX}_n) \)
EXAMPLE (PT. 1)

- $\text{pre(CEX}_{1'}) = x > 0$.

- Computed a refinement for $l_2$ from a counterexample generated for $l_1$. No need to verify $l_2$ independently!
EXAMPLE (PT. 1)

- Ensure $\text{EF } y = 1$ satisfies all initial states: Rule out CEX by adding $\neg \text{pre}(\text{CEX}_1)$ to each transition from $l$ to the error state.

- Re-run the safety checker.

- No more counterexamples are generated and all locations covered:

$$\diamond \langle \text{EF } y = 1 \rangle = (\text{pc}=l_1 \Rightarrow y=0) \land (\text{pc}=l_2 \Rightarrow x>0).$$
EXAMPLE (PT. 2)

- Modify $\varphi = \text{AGEF } y = 1$ by using $\varphi \langle \text{EF } y = 1 \rangle$:
  - $\varphi = \text{AG } ((\text{pc} = l_1 \Rightarrow y = 0) \land (\text{pc} = l_2 \Rightarrow x > 0))$. 

\begin{itemize}
  \item $\rho_1 : x' = 0$
  \item $\rho_2 : x \leq 0 \land y = 0$
  \item $\rho_3 : x \leq 0 \land y = 0$
  \item $\rho_4 : x > 0$
  \item $\rho_5 : x > 0$
  \item $\rho_6 : y \neq 0$
  \item $\rho_7 : x \leq 0$
\end{itemize}
EXAMPLE (PT. 2)

- $\varphi(l, \varphi)$ resulting in $\varphi(l, \varphi) = \bigwedge_{n \in \mathbb{N}} \neg \text{pre}(\text{CEX}_n)$

- Universal: the initial precondition $\varphi(\varphi) = \text{true}$. No counterexamples are generated thus $\varphi(\text{AGEF } y = 1) = \text{true}$!
Partition CTL formula preconditions by program location:

- $\varnothing (\varphi)$ takes the form $\bigwedge_{l_i} \ (pc = l_i \Rightarrow \varnothing (l, \varphi))$.

Universal location preconditions:

- $\varnothing (l, \varphi) = \bigwedge_{n \in \mathbb{N}} \neg \text{pre}(\text{CEX}_n)$

Existential location preconditions:

- $\varnothing (l, \varphi) = \bigvee_{n \in \mathbb{N}} \text{pre}(\text{CEX}_n)$
EXPERIMENTS

- Built as an extension to the open source project T2
- Input: C files converted to t2 file format + CTL specification.
- Compared our tool to:


## EXPERIMENTS

<table>
<thead>
<tr>
<th>LoC</th>
<th>Property</th>
<th>T2</th>
<th>[1]</th>
<th>[2]</th>
</tr>
</thead>
<tbody>
<tr>
<td>1050</td>
<td>$\text{AG}(b = 1 \implies \text{AF}(u = 0))$</td>
<td>67.3</td>
<td>T/O</td>
<td>T/O</td>
</tr>
<tr>
<td>1050</td>
<td>$\text{EG}(b = 1 \implies \text{EF}(u = 0))$</td>
<td>36.2</td>
<td>T/O</td>
<td>T/O</td>
</tr>
<tr>
<td>370</td>
<td>$\text{AG}(a = 1 \implies \text{EF}(r = 1))$</td>
<td>6.8</td>
<td>35.45</td>
<td>90.0</td>
</tr>
<tr>
<td>370</td>
<td>$\text{EF}(a = 1 \land \text{AG}(r \neq 1))$</td>
<td>4.7</td>
<td>T/O</td>
<td>T/O</td>
</tr>
<tr>
<td>370</td>
<td>$\text{EG}(\text{io} \neq 1) \land \text{EG}(\text{ret} \neq 1)$</td>
<td>13.5</td>
<td>T/O</td>
<td>7.6</td>
</tr>
<tr>
<td>370</td>
<td>$\text{AG}(\text{io} \neq 1) \lor \text{AG}(\text{ret} \neq 1)$</td>
<td>8.0</td>
<td>0.1</td>
<td>T/O</td>
</tr>
<tr>
<td>90</td>
<td>$\text{AGEF} w = 1$</td>
<td>2.0</td>
<td>0.7</td>
<td>T/O</td>
</tr>
<tr>
<td>90</td>
<td>$\text{EFAG} w \neq 1$</td>
<td>2.0</td>
<td>0.1</td>
<td>T/O</td>
</tr>
<tr>
<td>90</td>
<td>$\text{EFEG} w \neq 1$</td>
<td>0.1</td>
<td>0.1</td>
<td>35.2</td>
</tr>
</tbody>
</table>
LIMITATIONS

- Divergence can occur due to infinitely many counterexamples.
  - Take pre-image $\alpha$ of a PC, quantify out all variables that are updated proceeding a program location.

- Can lead to unsoundness due to over-approximation of the set of states for existential path quantifiers.

- Check that the precondition is sound e.g. that $\phi_1 \Rightarrow EG \phi_2$, we can use SMT based strategies to double check the small lemma on initial locations.
SUMMARY

- A new symbolic model checking procedure for CTL verification of infinite-state programs.

- Use a counterexample-guided precondition synthesis strategy to compute location-specific preconditions.

- Reduces the amount of irrelevant reasoning traditionally performed as several preconditions for each location can be computed simultaneously.

- Performance improvement and scalability!
BACKGROUND

- **P** = (L, E, Vars),
- Each edge **T** = (l, ρ, l’) in E, where l, l’ ∈ L and ρ is a condition, specifies possible transitions in the program.
- **T** = (S, R)
  - **S** = L × (Vars → Vals)
  - **R** ⊆ S × S
- A **cut-point** is a set C such that C ⊆ L and every cycle in the program’s graph contains at least one cut-point.

**Pre-images**: For a path **Π** = (l₀, ρ₀, l’₀), (l₁, ρ₁, l’₁),…, (lₙ, ρₙ, l’ₙ), we compute a pre-image for every possible suffix of **Π**:

- **pre**ₙ₊₁ = S and **pre**ᵢ = **pre**[(lᵢ, ρᵢ, l’ᵢ),…, (lₙ, ρₙ, l’ₙ)] as the set of states such that **pre**ᵢ = {s | ∃s’ ∈ **pre**ᵢ₊₁ s.t. ((lᵢ, s), (l’ᵢ, s’)) |= ρᵢ}.
FINDING TEMPORAL PRECONDITIONS

• Recurse over the structure of the given CTL formula.

• For each CTL sub-formula $\varphi$ we find a precondition $\wp\langle \varphi \rangle$ that ensures its satisfaction.

• Each sub-formula $\varphi$ is then replaced with $\wp\langle \varphi \rangle$ within the original formula.
  
  • Note: It is only necessary to handle formulas of nesting depth 1.

• To utilize sequential locality of a counterexample’s control-flow graph:
  
  • $\wp\langle \varphi \rangle$ takes the form $\bigwedge_{i} (pc = l_{i} \Rightarrow \wp\langle l_{i}, \varphi \rangle)$. 

24
FINDING TEMPORAL PRECONDITIONS

- For a universal CTL sub-property:
  - A precondition $\varphi(l, \phi)$ for a program location $l$ is initialized to true.
  - When a new counterexample is discovered, we refine
    $\varphi(l, \phi)$ resulting in $\varphi(l, \phi) = \bigwedge_{n \in \mathbb{N}} \neg \text{pre}(\text{CEX}_n)$

- For an existential CTL sub-property:
  - Verify the universal dual of the existential property and seek a set of counterexamples to serve as witnesses.
  - A precondition $\varphi(l, \phi)$ for a program location $l$ is initialized to false.
  - When a new counterexample is discovered, we refine
    $\varphi(l, \phi)$ resulting in $\varphi(l, \phi) = \bigvee_{n \in \mathbb{N}} \text{pre}(\text{CEX}_n)$