Parametric Symbolic Reachability

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Arie Gurfinkel

ongoing work with
Sharon Shoham-Buchbinder
SYMBOLIC REACHABILITY
Symbolic Reachability Problem

\[ P = (V, \text{Init}, \text{Tr}, \text{Bad}) \]

\( P \) is UNSAFE if and only if there exists a number \( N \) s.t.
\[
\text{Init}(X_0) \land \left( \bigwedge_{i=0}^{N-1} \text{Tr}(X_i, X_{i+1}) \right) \land \text{Bad}(X_N) \not\Rightarrow \perp
\]

\( P \) is SAFE if and only if there exists a safe inductive invariant \( \text{Inv} \) s.t.
\[
\begin{align*}
\text{Init} & \Rightarrow \text{Inv} \\
\text{Inv}(X) \land \text{Tr}(X, X' ) & \Rightarrow \text{Inv}(X') \\
\text{Inv} & \Rightarrow \neg \text{Bad}
\end{align*}
\]
Constrained Horn Clauses (CHC)

A Constrained Horn Clause (CHC) is a FOL formula of the form

\[ \forall V . (\phi \land p_1[X_1] \land \ldots \land p_n[X_n] \rightarrow h[X]), \]

where

- \( \phi \) is a constrained in the background theory \( A \)
- \( A \) is a combined theory of Linear Arithmetic, Arrays, Bit-Vectors, ...
- \( p_1, \ldots, p_n, h \) are n-ary predicates
- \( p_i[X] \) is an application of a predicate to first-order terms
CHC Terminology

Rule
\[ h[X] \leftarrow p_1[X_1], \ldots, p_n[X_n], \phi. \]

Query
\[ \text{false} \leftarrow p_1[X_1], \ldots, p_n[X_n], \phi. \]

Fact
\[ h[X] \leftarrow \phi. \]

Linear CHC
\[ h[X] \leftarrow p[X_1], \phi. \]

Non-Linear CHC
\[ h[X] \leftarrow p_1[X_1], \ldots, p_n[X_n], \phi. \]
for \( n > 1 \)
Example Horn Encoding

\[
\begin{align*}
    \text{l}_0 : & \quad x = 1 \\
    \text{l}_1 : & \quad b_1 = \text{nondet()} \\
    \text{l}_2 : & \quad x = x + y \\
    \text{l}_4 : & \quad \\
    \text{l}_3 : & \quad b_2 = x \geq y \\
\end{align*}
\]

\[
\begin{align*}
    \langle 1 \rangle & \quad \text{p}_0. \\
    \langle 2 \rangle & \quad \text{p}_1(x, y) \leftarrow \text{p}_0, \ x = 1, \ y = 0. \\
    \langle 3 \rangle & \quad \text{p}_2(x, y) \leftarrow \text{p}_1(x, y). \\
    \langle 4 \rangle & \quad \text{p}_3(x, y) \leftarrow \text{p}_1(x, y). \\
    \langle 5 \rangle & \quad \text{p}_1(x', y') \leftarrow \text{p}_2(x, y), \\
    & \quad x' = x + y, \\
    & \quad y' = y + 1. \\
    \langle 6 \rangle & \quad \text{p}_4 \leftarrow (x \geq y), \text{p}_3(x, y). \\
    \langle 7 \rangle & \quad \text{p}_{\text{err}} \leftarrow (x < y), \text{p}_3(x, y). \\
    \langle 8 \rangle & \quad \text{p}_4 \leftarrow \text{p}_4. \\
    \langle 9 \rangle & \quad \bot \leftarrow \text{p}_{\text{err}}. 
\end{align*}
\]
CHC Satisfiability

A model of a set of clauses $\Pi$ is an interpretation of each predicate $p_i$ that makes all clauses in $\Pi$ valid.

A set of clauses is **satisfiable** if it has a model, otherwise **unsatisfiable**.

A model is **A-definable**, if each $p_i$ is definable by a formula $\psi_i$ in $A$.

In the context of program verification:
- a program satisfies a property iff corresponding CHCs are satisfiable.
- verification certificates correspond to models of CHC.
- counterexamples correspond to derivations of false.
Spacer: Solving CHC in Z3

Spacer: a solver for SMT-constrained Horn Clauses
• stand-alone implementation in a fork of Z3
• http://bitbucket.org/spacer/code

Support for Non-Linear CHC
• model procedure summaries in inter-procedural verification conditions
• model assume-guarantee reasoning
• uses MBP to under-approximate models for finite unfoldings of predicates
• uses MAX-SAT to decide on an unfolding strategy

Supported SMT-Theories
• Best-effort support for arbitrary SMT-theories
  – data-structures, bit-vectors, non-linear arithmetic
• Full support for Linear arithmetic (rational and integer)
• Quantifier-free theory of arrays
  – only quantifier free models with limited applications of array equality
PARAMETRIC SYMBOLIC REACHABILITY
What we want to do ...

Form Methods Syst Des (2009) 34: 104–125

\[
\begin{align*}
\text{in} & \quad N : \text{natural where } N > 1 \\
\text{type} & \quad Pr\_id : [1..N] \\
& \quad Level : [0..N] \\
\text{local} & \quad y : \text{array } Pr\_id \text{ of } Level \text{ where } y = 0 \\
& \quad s : \text{array } Level \text{ of } Pr\_id \\
\text{loop forever do:} \\
& \quad \left[ l_0 : \text{Non-Critical} \\
& \quad l_1 : (y[i], s[1]) := (1, i) \\
& \quad l_2 : \text{while } y[i] < N \text{ do} \\
& \quad \left[ l_3 : \text{await } s[y[i]] \neq i \lor \forall j \neq i : y[j] < y[i] \\
& \quad l_4 : (y[i], s[y[i] + 1]) := (y[i] + 1, i) \\
& \quad l_5 : \text{Critical} \\
& \quad l_6 : y[i] := 0 \right] \\
\right]
\end{align*}
\]

**Fig. 2** PETERSON’S mutual exclusion protocol
What we want to do ...

```
global
    curr : array 0..(N - 1) of Location;
    next : array 0..(N - 1) of Location;

local
    desired : Q

def proc Proc(i):
    while true do
        CHOOSE: desired = f();
        TRY: ⟨await (\forall j. i < j \Rightarrow curr[j] \neq desired \land next[j] \neq desired)⟩;
            next[i] \leftarrow desired;
        WAIT: await (\forall j. j < i \Rightarrow next[i] \neq curr[j] \land next[i] \neq next[j]);
        MOVE: curr[i] \leftarrow next[i];

    def init(i,j):
        assume(curr[i] = next[i]);
        assume(i \neq j \Rightarrow curr[i] \neq curr[j]);

    def spec(i,j):
        assert(i \neq j \Rightarrow curr[i] \neq curr[j])

Algorithm 1: Collision avoidance.
```
Parametric Symbolic Reachability Problem

\[ T = ( \mathbf{v}, \text{Init}(\mathbf{v}), \text{Tr}(i, N, \mathbf{v}, \mathbf{v}'), \text{Bad}(\mathbf{v}) ) \]

- \( \mathbf{v} \) is a set of state variables
  - each \( v_k \in \mathbf{v} \) is a map \( \text{Nat} \rightarrow \text{Rat} \)
  - \( \mathbf{v} \) is partitioned into \( \text{Local}(\mathbf{v}) \) and \( \text{Global}(\mathbf{v}) \)
- \( \text{Init}(\mathbf{v}) \) and \( \text{Bad}(\mathbf{v}) \) are initial and bad states, respectively
- \( \text{Tr}(i, N, \mathbf{v}, \mathbf{v}') \) is a transition relation, parameterized by a process identifier \( i \) and total number of processes \( N \)

All formulas are over the combined theories of arrays and LRA

- \( \text{Init}(\mathbf{v}) \) and \( \text{Bad}(\mathbf{v}) \) contain at most 2 quantifiers
  - \( \text{Init}(\mathbf{v}) = \forall x,y \cdot \varphi_{\text{Init}}(x, y, \mathbf{v}) \), where \( \varphi_{\text{Init}} \) is quantifier free (QF)
  - \( \text{Bad}(\mathbf{v}) = \forall x,y \cdot \varphi_{\text{Bad}}(x, y, \mathbf{v}) \), where \( \varphi_{\text{Bad}} \) is QF

- \( \text{Tr} \) contains at most 1 quantifier
  - \( \text{Tr}(i, N, \mathbf{v}, \mathbf{v}') = \forall j \cdot \rho(i, j, N, \mathbf{v}, \mathbf{v}') \)
### A State of a Parametric System

<table>
<thead>
<tr>
<th>PID</th>
<th>Global</th>
<th>Local</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>$v_0$</td>
<td>$v_1$</td>
</tr>
<tr>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>1</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
</tr>
<tr>
<td>4</td>
<td></td>
<td></td>
</tr>
<tr>
<td>5</td>
<td></td>
<td></td>
</tr>
<tr>
<td>6</td>
<td></td>
<td></td>
</tr>
<tr>
<td>...</td>
<td></td>
<td></td>
</tr>
<tr>
<td>N</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Extra restrictions on the transition relation

Parametricity

• $Tr$ depends only on first $N$ entries of each state variable

$$(\forall j \in [0..N) . \upsilon(j) = \upsilon'(j)) \Rightarrow (Tr(i, N, \upsilon, \upsilon') \iff Tr(i, N, \upsilon, \upsilon'))$$

Locality

• $Tr$ does not modify local variables of other processes

$$Tr(i, N, \upsilon, \upsilon') \Rightarrow (\forall j . j \neq i \Rightarrow Local(\upsilon)(j) = Local(\upsilon')(j))$$

(Optional) Single-writer

• Every state-variable (including global) is written by exactly one process

$$Tr(i, N, \upsilon, \upsilon') \Rightarrow (\forall j \in [0..N) . j \neq i \Rightarrow \upsilon(j) = \upsilon'(j))$$
Parametric Symbolic Reachability

\[ T = (v, \text{Init}, Tr, \text{Bad}) \]

\( T \) is UNSAFE if and only if there exists a number \( K \) s.t.

\[ \text{Init}(v_0) \land (\bigwedge_{s \in [0, K]} \text{Tr}(i_s, N, v_s, v_{s+1})) \land \text{Bad}(v_K) \neq \bot \]

\( T \) is SAFE if and only if there exists a safe inductive invariant \( \text{Inv} \) s.t.

\[ \text{Init}(v) \Rightarrow \text{Inv}(v) \]

\[ \text{Inv}(v) \land \text{Tr}(i, N, v, v') \Rightarrow \text{Inv}(v') \]

\[ \text{Inv}(v) \Rightarrow \neg \text{Bad}(v) \]

\[ \text{Safe}(T) \]
Parametric vs Non-Parametric Reachability

\[ \text{Init}(v) \Rightarrow \text{Inv}(v) \]
\[ \text{Inv}(v) \land \text{Tr}(i, N, v, v') \Rightarrow \text{Inv}(v') \]
\[ \text{Inv}(v) \Rightarrow \neg \text{Bad}(v) \]

\{ \text{Safe}(T) \}

\text{Init, Bad, and Tr} might contain quantifiers
- e.g., “ALL processes start in unique locations”
- e.g., “only make a step if ALL other processes are ok”
- e.g., “EXIST two distinct process in a critical section”

\text{Inv} cannot be assumed to be quantifier free
- QF \text{Inv} is either non-parametric or trivial

Decide existence of quantified models for CHC
- stratify search by the number of quantifiers
- models with 1 quantifier, 2 quantifiers, 3 quantifiers, etc…
ONE QUANTIFIER
TWO QUANTIFIER
One Quantifier (Solution)

\[ \begin{align*}
Init(i, \nu) & \implies Inv_1(i, \nu) \\
Inv_1(i, \nu) \land Tr(i, \nu, \nu') & \implies Inv_1(i, \nu') \\
 j \neq i \land Inv_1(i, \nu) \land Inv_1(j, \nu) \land Tr(j, \nu, \nu') & \implies Inv_1(i, \nu') \\
 Inv_1(i, \nu) \land Inv_1(j, \nu) & \implies \neg Bad(i, j, \nu)
\end{align*} \]

Claim

- If Safe_1(T) is QF-SAT then Safe(T) is SAT
- If \( Tr \) does not contain functions that range over PIDs, then Safe_1(T) is QF-SAT only if Safe(T) admits a model definable with a single quantifier

Safe_1(T) is essentially Owicki-Gries for 2 processes \( i \) and \( j \)

If \( Tr \) is **single-writer** then the 3rd rule is not needed

- get linear CHC
One Quantifier explained (induction rule)

\[ Inv(v) \land Tr(i, v, v') \Rightarrow Inv(v') \]
( plug \( \forall j. Inv_1(j, v) \) for \( Inv(v) \) )

\[(\forall j . Inv_1(j, v)) \land Tr(i, v, v') \Rightarrow Inv_1(k, v') \]
( instantiate \( j \) by \( k \) and \( i \) )

\[ Inv_1(k, v) \land Inv_1(i, v) \land Tr(i, v, v') \Rightarrow Inv_1(k, v') \]

Unless \( Tr \) contains other PIDs, no other instantiations are possible

Split into two rules using \( i = k \)

If \( Tr \) contains quantifiers, they can be instantiated using \( i \) and \( k \) as well
Two Quantifier (Solution)

\[ i \neq j \land \text{Init}(i, j, v) \land \text{Init}(j, i, v) \Rightarrow \text{Inv}_2(i, j, v) \]
\[ i \neq j \land \text{Inv}_2(i, j, v) \land \text{Tr}(i, v, v') \Rightarrow \text{Inv}_2(i, j, v') \]
\[ i \neq j \land \text{Inv}_2(i, j, v) \land \text{Tr}(j, v, v') \Rightarrow \text{Inv}_2(i, j, v') \]
\[ i \neq j \land \text{Inv}_2(i, j, v) \Rightarrow \neg \text{Bad}(i, j, v) \]

Claim

- assume that \( \text{Tr} \) satisfies **single-writer**, then
- If \( \text{Safe}_2(T) \) is QF-SAT then \( \text{Safe}(T) \) is SAT
- If \( \text{Tr} \) does not contain functions that range over PIDs, then \( \text{Safe}_2(T) \) is QF-SAT only if \( \text{Safe}(T) \) admits a model definable with at most two quantifier

Single-writer => linear CHC

- still working out good solution for general case
Symmetric Models

Definition
• A formula $\varphi(x,y)$ is symmetric in $(x,y)$ iff $\varphi(x,y) \Leftrightarrow \varphi(y,x)$

Claim
• A set of CHC $S$ admits a quantified model of the form $\forall x, y. M(x, y)$ iff
• $S$ admits a quantified model of the form $\forall x, y. x \neq y \Rightarrow H(x,y)$,
• where $H(x,y)$ is symmetric in $(x, y)$
• (assuming that the sort of $x,y$ is $\geq 2$)
Two Quantifier Explained (induction rule)

\[ \text{Inv}(v) \land Tr(i, v, v') \Rightarrow \text{Inv}(v') \]

(plug \( \forall x, y. x \neq y \Rightarrow \text{Inv}_2(x, y, v) \) for \( \text{Inv}(v) \) )

\[ ((\forall x, y . x \neq y \Rightarrow \text{Inv}_2(x, y, v)) \land Tr(i, v, v') \land h \neq j) \Rightarrow \text{Inv}_2(h, j, v') \]

(by symmetry, only need 3 instantiations \((h,j),(i,h),(i,j)\))

\[ (h \neq j \land \text{Inv}_2(h, j, v) \land \]
\[ (i \neq h \Rightarrow \text{Inv}_2(i, h, v)) \land (i \neq j \Rightarrow \text{Inv}_2(i, j, v)) \land \]
\[ Tr(i, v, v') \) \Rightarrow \text{Inv}_2(h, j, v') \]

(split based on \( i \neq h \land i \neq j, i \neq h, i \neq j \))
Two quantifiers explained (cont’d)

\[ h \neq j \land \text{Inv}_2(h, j, v) \land i \neq h \land \text{Inv}_2(i, h, v) \land i \neq j \land \text{Inv}_2(i, j, v) \land \text{Tr}(i, v, v') \implies \text{Inv}_2(h, j, v') \]

\[ h \neq j \land \text{Inv}_2(h, j, v) \land \text{Inv}_2(h, j, v) \land \text{Tr}(h, v, v') \implies \text{Inv}_2(h, j, v') \]

\[ h \neq j \land \text{Inv}_2(h, j, v) \land \text{Inv}_2(h, j, v) \land \text{Tr}(j, v, v') \implies \text{Inv}_2(h, j, v') \]
Two Quantifiers (repeated)

\[ i \neq j \land \text{Init}(i, j, v) \land \text{Init}(j, i, v) \Rightarrow \text{Inv}_2(i, j, v) \]
\[ i \neq j \land \text{Inv}_2(i, j, v) \land \text{Tr}(i, v, v') \Rightarrow \text{Inv}_2(i, j, v') \]
\[ i \neq j \land \text{Inv}_2(i, j, v) \land \text{Tr}(j, v, v') \Rightarrow \text{Inv}_2(i, j, v') \]
\[ i \neq j \land \text{Inv}_2(i, j, v) \Rightarrow \neg \text{Bad}(i, j, v) \]

Claim

• assume that Tr satisfies single-writer, then
• If Safe_2(T) is QF-SAT then Safe(T) is SAT
• If Tr does not contain functions that range over PIDs, then Safe_2(T) is QF-SAT only if Safe(T) admits a model definable with at most two quantifier

Single-writer => linear CHC

• still working out good solution for general case
What we can do now

Peterson’s protocol 😊
• and similar small protocols

Input problem T
• Init, Transition, and Bad
• in (extension of) SMT-LIB format
• over combined theory of Linear Arithmetic and Arrays

Generate Constrained Horn Clauses
• Safe_1(T) or Safe_2(T)

Solve using QF CHC solver
• Spacer works for small protocols
Related Work

Kedar Namjoshi et al.
- Local Proofs for Global Safety Properties, and many other papers
- systematic derivation of proof rules for concurrent systems
- finite state and fixed number of processes

Andrey Rybalchenko et al.
- Compositional Verification of Multi-Threaded Programs, and others
- compositional proof rules for concurrent systems are CHC
- infinite state and fixed number of processes

Lenore Zuck et al.
- Invisible Invariants
- finite state and parametric number of processes
- finite model theorem for special classes of parametric systems

Nikolaj Bjørner, Kenneth L. McMillan, and Andrey Rybalchenko
- On Solving Universally Quantified Horn Clauses. SAS 2013:
Conclusion

Parametric Verification == Quantified Models for CHC

Quantifier instantiation to *systematically* derive proof rules for verification of safety properties of parametric systems
- parametric systems definable with SMT-LIB syntax

Lazy vs Eager Quantifier Instantiation
- eager instantiation in this talk
- “easy” to extend to lazy / dynamic / model-based instantiation

Connections with other work in parametric verification
- complete instantiation = decidability?
- relative completeness
- …
Contact Information

Arie Gurfinkel, Ph. D.
Sr. Researcher
CSC/SSD
Telephone: +1 412-268-5800
Email: info@sei.cmu.edu

Web
www.sei.cmu.edu
www.sei.cmu.edu/contact.cfm

U.S. Mail
Software Engineering Institute
Customer Relations
4500 Fifth Avenue
Pittsburgh, PA 15213-2612
USA

Customer Relations
Email: info@sei.cmu.edu
Telephone: +1 412-268-5800
SEI Phone: +1 412-268-5800
SEI Fax: +1 412-268-6257