Parametric Symbolic Reachability

Software Engineering Institute
Carnegie Mellon University
Pittsburgh, PA 15213

Arie Gurfinkel

ongoing work with
Sharon Shoham-Buchbinder

This material is based upon work funded and supported by the Department of Defense under Contract No. FA8721-05-C-0003 with Carnegie Mellon University for the operation of the Software Engineering Institute, a federally funded research and development center.

SYMBOLIC REACHABILITY

Symbolic Reachability Problem

P = (V, Init, Tr, Bad)

P is UNSAFE if and only if there exists a number *N* s.t.

$$Init(X_0) \wedge \left(\bigwedge_{i=0}^{N-1} Tr(X_i, X_{i+1}) \right) \wedge Bad(X_N) \not\Rightarrow \bot$$

P is SAFE if and only if there exists a *safe inductive invariant Inv* s.t.

$$\left.\begin{array}{c} Init \Rightarrow Inv \\ Inv(X) \wedge Tr(X, X') \Rightarrow Inv(X') \end{array}\right\} \quad \text{Inductive}$$

$$Inv \Rightarrow \neg Bad \quad \text{Safe}$$

Constrained Horn Clauses (CHC)

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

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

where

- ϕ 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 $\overset{\text{head}}{h[X]} \leftarrow \overbrace{p_1[X_1], \ldots, p_n[X_n], \overset{\text{constraint}}{\phi}}^{\text{body}}.$

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

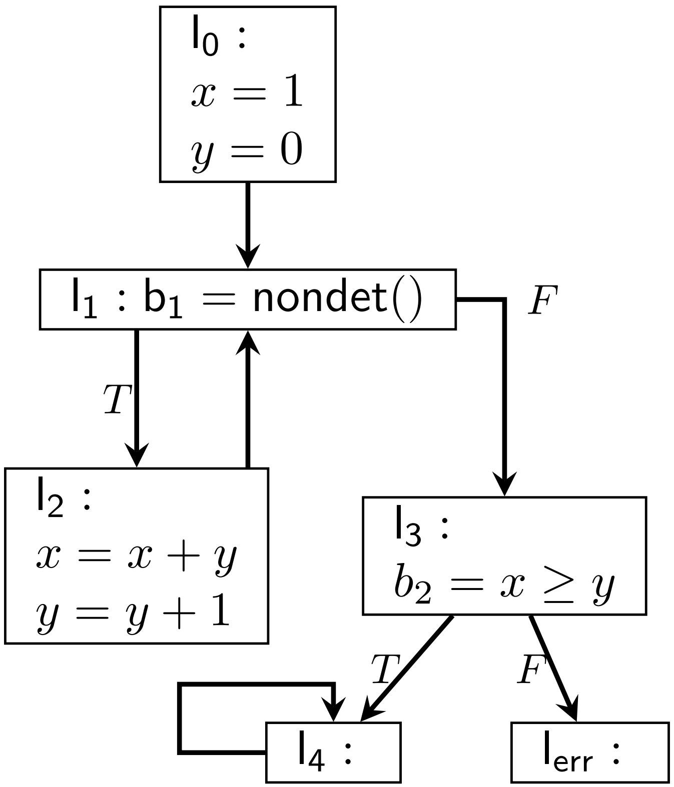
Fact $h[X] \leftarrow \phi.$

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

$$
\begin{aligned}
&\text{int } x = 1;\\
&\text{int } y = 0;\\
&\mathsf{while}\ (*)\ \{\\
&\quad x = x + y;\\
&\quad y = y + 1;\\
&\}\\
&\mathsf{assert}(x \geq y);
\end{aligned}
$$



$$
\begin{aligned}
&\langle 1 \rangle\ \mathsf{p}_0.\\
&\langle 2 \rangle\ \mathsf{p}_1(x, y) \leftarrow\\
&\qquad\quad \mathsf{p}_0, x = 1, y = 0.\\
&\langle 3 \rangle\ \mathsf{p}_2(x, y) \leftarrow \mathsf{p}_1(x, y)\ .\\
&\langle 4 \rangle\ \mathsf{p}_3(x, y) \leftarrow \mathsf{p}_1(x, y)\ .\\
&\langle 5 \rangle\ \mathsf{p}_1(x', y') \leftarrow\\
&\qquad\quad \mathsf{p}_2(x, y),\\
&\qquad\quad x' = x + y,\\
&\qquad\quad y' = y + 1.\\
&\langle 6 \rangle\ \mathsf{p}_4 \leftarrow (x \geq y), \mathsf{p}_3(x, y).\\
&\langle 7 \rangle\ \mathsf{p}_{\mathsf{err}} \leftarrow (x < y), \mathsf{p}_3(x, y).\\
&\langle 8 \rangle\ \mathsf{p}_4 \leftarrow \mathsf{p}_4.\\
&\langle 9 \rangle\ \bot \leftarrow \mathsf{p}_{\mathsf{err}}.
\end{aligned}
$$

CHC Satisfiability

A **model** of a set of clauses Π is an interpretation of each predicate p_i that makes all clauses in Π valid

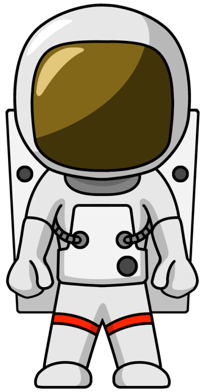
A set of clauses is **satisfiable** if it has a model, otherwise **unsatisfiable**

A model is **A-definable**, it each p_i is definable by a formula ψ_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

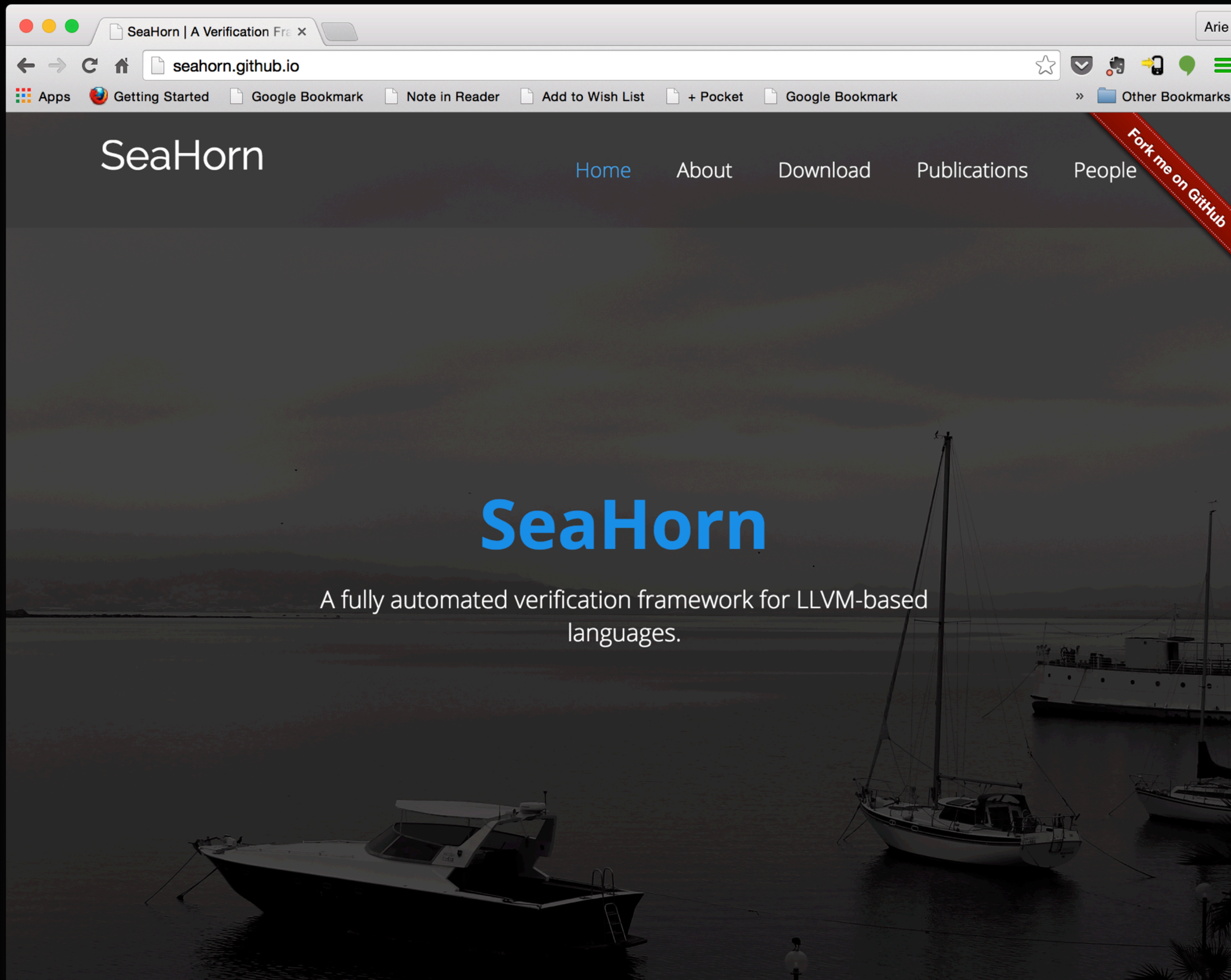
- stand-alone implementation in a fork of Z3
- http://bitbucket.org/spacer/code

Support for Non-Linear CHC

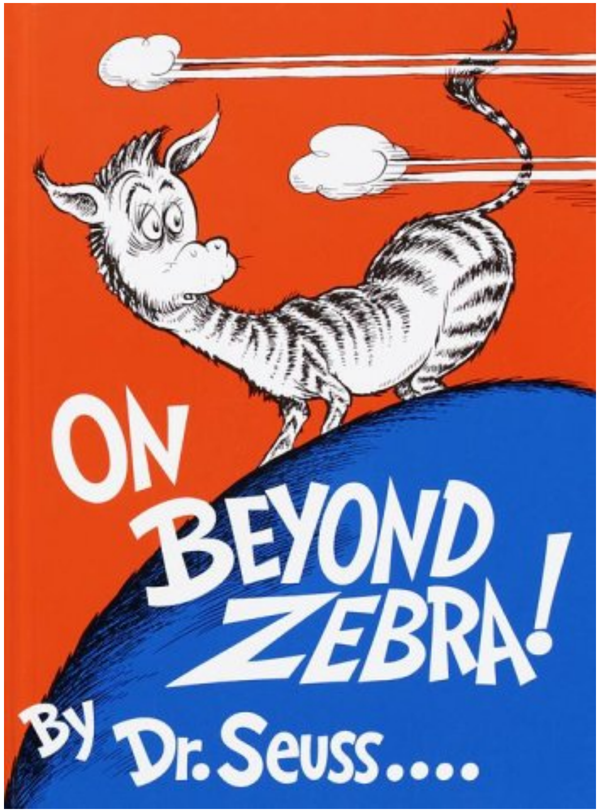
- model procedure summaries in inter-procedural verification conditions
- model assume-guarantee reasoning
- uses MBP to under-approximate models for finite unfoldings of predicates
- uses MAX-SAT to decide on an unfolding strategy

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



http://seahorn.github.io



PARAMETRIC SYMBOLIC REACHABILITY

What we want to do …

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

$$
\left[\begin{array}{llll}
\textbf{in} & N & : & \textbf{natural where } N > 1 \\
\textbf{type} & Pr\_id & : & [1..N] \\
 & Level & : & [0..N] \\
\textbf{local} & y & : & \textbf{array } Pr\_id \textbf{ of } Level \textbf{ where } y = 0 \\
 & s & : & \textbf{array } Level \textbf{ of } Pr\_id \\
\overset{N}{\underset{i=1}{\Vert}} & P[i] :: & & \left[\begin{array}{l}
\textbf{loop forever do}: \\
\quad \left[\begin{array}{l}
l_0: \ \textit{Non-Critical} \\
l_1: \ (y[i], s[1]) \ := (1, i) \\
l_2: \ \textbf{while } y[i] < N \textbf{ do} \\
\quad \left[\begin{array}{l}
l_3: \ \textbf{await } s[y[i]] \neq i \ \vee \ \forall j \neq i: \ y[j] < y[i] \\
l_4: \ (y[i], s[y[i]+1]) \ := (y[i]+1, i)
\end{array}\right] \\
l_5: \ \textit{Critical} \\
l_6: \ y[i] \ := 0
\end{array}\right]
\end{array}\right]
\end{array}\right]
$$

Fig. 2 PETERSON'S mutual exclusion protocol

A. Cohen, K. S. Namjoshi: Local proofs for global safety properties. FMSD 34(2): 104-125 (2009)

What we want to do …

global

$curr$: **array** $0..(N-1)$ **of** Location;

$next$: **array** $0..(N-1)$ **of** Location;

local

$desired : \mathbb{Q}$

def proc Proc(i) **:**

while *true* **do**

CHOOSE: $desired = f()$;

TRY: $\langle$**await**$(\forall j. i < j \Rightarrow curr[j] \neq desired \land next[j] \neq desired)$;

$next[i] \leftarrow desired\rangle$;

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 = (**v**, *Init*(**v**), *Tr*(*i*, N, **v**, **v**'), *Bad* (**v**))

- **v** is a set of state variables
  - each $v_k \in$ **v** is a map *Nat*→*Rat*
  - **v** is partitioned into *Local*(**v**) and *Global*(**v**)
- *Init*(**v**) and *Bad*(**v**) are initial and bad states, respectively
- *Tr*(*i*, N, **v**, **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

Init(**v**) and *Bad*(**v**) contain at most 2 quantifiers

- Init(**v**) = $\forall$ x,y . φ_{Init}(x, y, **v**), where φ_{Init} is quantifier free (QF)
- Bad(**v**) = $\forall$ x,y . φ_{Bad}(x, y, **v**), where φ_{Bad} is QF

Tr contains at most 1 quantifier

- *Tr*(*i*, N, **v**, **v**') = $\forall$ *j* . ρ (*i*, *j*, N, **v**, **v**')

A State of a Parametric System

PID	Global				Local					
	V_0	V_1	V_2	V_3	V_4	V_5	V_6	V_7	V_8	V_9
0										
1										
2										
3										
4										
5										
6										
…										
N										

Extra restrictions on the transition relation

Parametricity

- *Tr* depends only on first N entries of each state variable

$$(\forall j \in [0..N) \,.\, \boldsymbol{v}(j) = \boldsymbol{u}(j)) \Rightarrow (Tr(i, N, \boldsymbol{v}, \boldsymbol{v}') \iff Tr(i, N, \boldsymbol{u}, \boldsymbol{v}'))$$

Locality

- *Tr* does not modify local variables of other proceses

$$Tr(i, N, \boldsymbol{v}, \boldsymbol{v}') \Rightarrow (\forall j \,.\, j \neq i \Rightarrow Local(\boldsymbol{v})(j) = Local(\boldsymbol{v}')(j))$$

(Optional) **Single-writer**

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

$$Tr(i, N, \boldsymbol{v}, \boldsymbol{v}') \Rightarrow (\forall j \in [0..N) \,.\, j \neq i \Rightarrow \boldsymbol{v}(j) = \boldsymbol{v}'(j))$$

Parametric Symbolic Reachability

T = ($\mathbf{v}$, *Init*, Tr, *Bad*)

T is UNSAFE if and only if there exists a number *K* s.t.

$$Init(\boldsymbol{v}_0) \wedge (\bigwedge_{s \in [0,K)} Tr(i_s, N, \boldsymbol{v}_s, \boldsymbol{v}_{s+1})) \wedge Bad(\boldsymbol{v}_K) \not\Rightarrow \bot$$

T is SAFE if and only if there exists a *safe inductive invariant* *Inv* *s.t.*

$$\left.\begin{array}{r} Init(\boldsymbol{v}) \Rightarrow Inv(\boldsymbol{v}) \\ Inv(\boldsymbol{v}) \wedge Tr(i, N, \boldsymbol{v}, \boldsymbol{v}') \Rightarrow Inv(\boldsymbol{v}') \\ Inv(\boldsymbol{v}) \Rightarrow \neg Bad(\boldsymbol{v}) \end{array}\right\} \textbf{Safe(T)}$$

Parametric vs Non-Parametric Reachability

$$\left.\begin{aligned} Init(\boldsymbol{v}) &\Rightarrow Inv(\boldsymbol{v}) \\ Inv(\boldsymbol{v}) \land Tr(i, N, \boldsymbol{v}, \boldsymbol{v}') &\Rightarrow Inv(\boldsymbol{v}') \\ Inv(\boldsymbol{v}) &\Rightarrow \neg Bad(\boldsymbol{v}) \end{aligned}\right\} \textbf{Safe(T)}$$

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"*

Inv cannot be assumed to be quantifier free

- QF *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)

$$\left.\begin{array}{rcl} Init(i, \boldsymbol{v}) & \implies & Inv_1(i, \boldsymbol{v}) \\ Inv_1(i, \boldsymbol{v}) \land Tr(i, \boldsymbol{v}, \boldsymbol{v}') & \implies & Inv_1(i, \boldsymbol{v}') \\ j \neq i \land Inv_1(i, \boldsymbol{v}) \land Inv_1(j, \boldsymbol{v}) \land Tr(j, \boldsymbol{v}, \boldsymbol{v}') & \implies & Inv_1(i, \boldsymbol{v}') \\ Inv_1(i, \boldsymbol{v}) \land Inv_1(j, \boldsymbol{v}) & \implies & \neg Bad(i, j, \boldsymbol{v}) \end{array}\right\} \textbf{Safe}_{\mathbf{1}}\textbf{(T)}$$

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)

$$Inv(\boldsymbol{v}) \wedge Tr(i, \boldsymbol{v}, \boldsymbol{v}') \Rightarrow Inv(\boldsymbol{v}')$$

(plug $\forall j.Inv_1(j, \mathbf{v})$ for $Inv(\mathbf{v})$)

$$(\forall j \,.\, Inv_1(j, \boldsymbol{v})) \wedge Tr(i, \boldsymbol{v}, \boldsymbol{v}') \Rightarrow Inv_1(k, \boldsymbol{v}')$$

(instantiate *j* by *k* and *i*)

$$Inv_1(k, \boldsymbol{v}) \wedge Inv_1(i, \boldsymbol{v}) \wedge Tr(i, \boldsymbol{v}, \boldsymbol{v}') \Rightarrow Inv_1(k, \boldsymbol{v}')$$

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)

$$\left.\begin{array}{r} i \neq j \wedge Init(i,j,\boldsymbol{v}) \wedge Init(j,i,\boldsymbol{v}) \Rightarrow Inv_2(i,j,\boldsymbol{v}) \\ i \neq j \wedge Inv_2(i,j,\boldsymbol{v}) \wedge Tr(i,\boldsymbol{v},\boldsymbol{v}') \Rightarrow Inv_2(i,j,\boldsymbol{v}') \\ i \neq j \wedge Inv_2(i,j,\boldsymbol{v}) \wedge Tr(j,\boldsymbol{v},\boldsymbol{v}') \Rightarrow Inv_2(i,j,\boldsymbol{v}') \\ i \neq j \wedge Inv_2(i,j,\boldsymbol{v}) \Rightarrow \neg Bad(i,j,\boldsymbol{v}) \end{array}\right\} \textbf{Safe}_{\mathbf{2}}\textbf{(T)}$$

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

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)

$$Inv(\boldsymbol{v}) \wedge Tr(i, \boldsymbol{v}, \boldsymbol{v}') \Rightarrow Inv(\boldsymbol{v}')$$

(plug $\forall$x,y.x≠y $\Rightarrow$ *Inv*$_2$(x,y,**v**) for *Inv*(**v**))

$$\big((\forall x, y \,.\, x \neq y \Rightarrow Inv_2(x, y, \boldsymbol{v})) \wedge Tr(i, \boldsymbol{v}, \boldsymbol{v}') \wedge h \neq j\big) \implies Inv_2(h, j, \boldsymbol{v}')$$

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

$$\begin{aligned}\big(h \neq j \wedge Inv_2(h, j, \boldsymbol{v}) \wedge \\ (i \neq h \Rightarrow Inv_2(i, h, \boldsymbol{v})) \wedge (i \neq j \Rightarrow Inv_2(i, j, \boldsymbol{v})) \wedge \\ Tr(i, \boldsymbol{v}, \boldsymbol{v}')\big) \implies Inv_2(h, j, \boldsymbol{v}')\end{aligned}$$

(split based on i≠h$\wedge$i≠j , i ≠ h, i ≠ j)

Two quantifiers explained (cont'd)

single-writer

$$h \neq j \wedge Inv_2(h, j, \boldsymbol{v}) \wedge \cancel{i \neq h \wedge Inv_2(i, h, \boldsymbol{v})} \wedge$$
$$\cancel{i \neq j} \wedge Inv_2(i, j, \boldsymbol{v}) \wedge Tr(i, \boldsymbol{v}, \boldsymbol{v}') \implies Inv_2(h, j, \boldsymbol{v}')$$

duplicate

$$h \neq j \wedge Inv_2(h, j, \boldsymbol{v}) \wedge \cancel{Inv_2(h, j, \boldsymbol{v})} \wedge Tr(h, \boldsymbol{v}, \boldsymbol{v}') \implies Inv_2(h, j, \boldsymbol{v}')$$

symmetry

$$h \neq j \wedge Inv_2(h, j, \boldsymbol{v}) \wedge \cancel{Inv_2(j, h, \boldsymbol{v})} \wedge Tr(j, \boldsymbol{v}, \boldsymbol{v}') \implies Inv_2(h, j, \boldsymbol{v}')$$

Two Quantifiers (repeated)

$$\left.\begin{array}{r} i \neq j \wedge Init(i,j,\boldsymbol{v}) \wedge Init(j,i,\boldsymbol{v}) \Rightarrow Inv_2(i,j,\boldsymbol{v}) \\ i \neq j \wedge Inv_2(i,j,\boldsymbol{v}) \wedge Tr(i,\boldsymbol{v},\boldsymbol{v}') \Rightarrow Inv_2(i,j,\boldsymbol{v}') \\ i \neq j \wedge Inv_2(i,j,\boldsymbol{v}) \wedge Tr(j,\boldsymbol{v},\boldsymbol{v}') \Rightarrow Inv_2(i,j,\boldsymbol{v}') \\ i \neq j \wedge Inv_2(i,j,\boldsymbol{v}) \Rightarrow \neg Bad(i,j,\boldsymbol{v}) \end{array}\right\} \textbf{Safe}_{\mathbf{2}}\textbf{(T)}$$

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