Solving (Quantified) Horn Clauses

Andrey Rybalchenko
Microsoft Research Cambridge and TUM

joint work with

T. Beyene, N. Bjørner,
S. Chaudhuri, K. McMillan, and C. Popeea

October 4, 2013

Program Verification and Synthesis as (Horn) Constraint Solving

- Universal temporal properties, e.g., LTL
- Quantifier free invariants/auxiliary assertions
- Proof rule formalization as Horn clauses

Transition System

- v - program variables
- $init(v)$ - initial states
- $step(v, v')$ - transition relation
- $safe(v)$ - safe states

Safety and Termination (WF) of Transition System

$\exists inv(v):$

$$init(v) \rightarrow inv(v)$$
$$inv(v) \wedge step(v, v') \rightarrow inv(v')$$

Safety and Termination (WF) of Transition System

$\exists inv(v)$:

$$init(v) \rightarrow inv(v)$$
$$inv(v) \wedge step(v, v') \rightarrow inv(v')$$
$$inv(v) \rightarrow safe(v) \qquad \text{safety}$$
$$wf(inv(v) \wedge step(v, v')) \qquad \text{well-foundedness}$$

From WF to DWF

$wf(rel(v, v'))$

iff

$\exists ti(v, v')$:

$rel(v, v') \rightarrow ti(v, v')$

$ti(v, v') \wedge rel(v', v'') \rightarrow ti(v, v'')$

$dwf(ti(v, v'))$ disjunctive well-foundedness

Program with procedures

- v - program variables
- $init(v)$ - initial states of main procedure
- $step(v, v')$ - intra-procedural transition relation
- $safe(v)$ - safe states

Program with procedures

- v - program variables
- $init(v)$ - initial states of main procedure
- $step(v, v')$ - intra-procedural transition relation
- $safe(v)$ - safe states
- $call(v, v')$ - parameter passing relation
- $ret(v, v')$ - return value passing

Safety and Termination of Program with Procedures

$\exists sum(v, v')$:

$$init(v_0) \rightarrow sum(v_0, v_0)$$
$$sum(v_0, v_1) \wedge step(v_1, v_2) \rightarrow sum(v_0, v_2)$$
$$sum(v_0, v_1) \wedge call(v_1, v_2) \rightarrow sum(v_2, v_2)$$
$$sum(v_0, v_1) \wedge call(v_1, v_2) \wedge sum(v_2, v_3) \wedge ret(v_3, v_4) \rightarrow sum(v_0, v_4)$$

Safety and Termination of Program with Procedures

$\exists sum(v, v')$:

$$init(v_0) \rightarrow sum(v_0, v_0)$$
$$sum(v_0, v_1) \wedge step(v_1, v_2) \rightarrow sum(v_0, v_2)$$
$$sum(v_0, v_1) \wedge call(v_1, v_2) \rightarrow sum(v_2, v_2)$$
$$sum(v_0, v_1) \wedge call(v_1, v_2) \wedge sum(v_2, v_3) \wedge ret(v_3, v_4) \rightarrow sum(v_0, v_4)$$
$$sum(v_0, v_1) \rightarrow safe(v_1)$$
$$wf(\exists v_1 : sum(v_0, v_1) \wedge call(v_1, v_2))$$

Multi-Threaded Program

- $v = (g, l_1, l_2)$ - global and thread-local variables
- $init(v)$ - initial states
- $safe(v)$ - safe states

Multi-Threaded Program

- $v = (g, l_1, l_2)$ - global and thread-local variables
- $init(v)$ - initial states
- $safe(v)$ - safe states
- $step_1(v, v')$ - transition relation of 1st thread, preserves l_2
- $step_2(v, v')$ - transition relation of 2nd thread, preserves l_1

Rely/Guarantee Rule for Safety

$$\exists inv_1(v)\ \exists inv_2(v)\ \exists env_1(v, v')\ \exists env_2(v, v') :$$

$$init(v) \rightarrow inv_1(v)$$
$$inv_1(v) \wedge step_1(v, v') \rightarrow inv_1(v') \wedge env_2(v, v')$$
$$inv_1(v) \wedge env_1(v, v') \rightarrow inv_1(v')$$
$$\ldots$$
$$inv_1(v) \wedge inv_2(v) \rightarrow safe(v)$$

Clauses for preservation of $inv_2(v)$ are symmetric

Resolving Rely/Guarantee Rule

$$\exists env_2(v, v') :$$

$$\begin{aligned}
&\ldots \\
&inv_1(v) \land step_1(v, v') \rightarrow env_2(v, v') \\
&\ldots \\
&inv_2(v) \land env_2(v, v') \rightarrow inv_2(v') \\
&\ldots
\end{aligned}$$

Into Owicki/Gries Rule

$$\ldots$$

$$env_2(v, v') := inv_1(v) \land step_1(v, v')$$

$$\ldots$$

$$inv_2(v) \land inv_1(v) \land step_1(v, v') \rightarrow inv_2(v')$$

$$\ldots$$

Owicki/Gries Rule for Safety

$$\exists inv_1(v)\ \exists inv_2(v) :$$

$$init(v) \rightarrow inv_1(v)$$
$$inv_1(v) \wedge step_1(v, v') \rightarrow inv_1(v')$$
$$inv_1(v) \wedge inv_2(v) \wedge step_2(v, v') \rightarrow inv_1(v')$$
$$\ldots$$
$$inv_1(v) \wedge inv_2(v) \rightarrow safe(v)$$

Clauses for preservation of $inv_2(v)$ are symmetric

Thread-Modular Rule for Safety

$$\exists \mathit{inv_1}(g, l_1)\ \exists \mathit{inv_2}(g, l_2)\ \exists \mathit{env}(g, g') :$$

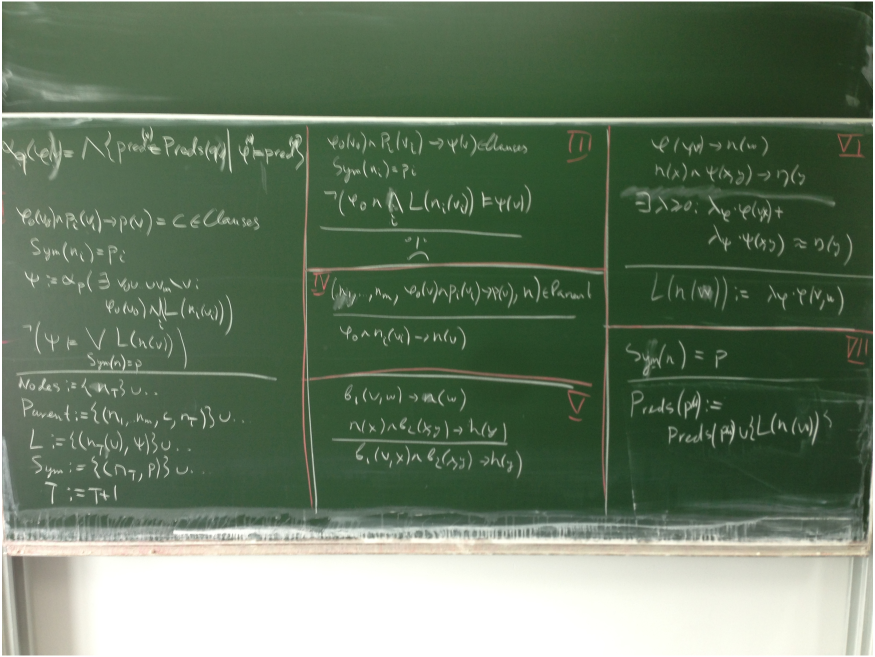
$$\mathit{init}(v) \rightarrow \mathit{inv_1}(g, l_1)$$
$$\mathit{inv_1}(g, l_1) \wedge \mathit{step_1}(v, v') \rightarrow \mathit{inv_1}(g', l_1') \wedge \mathit{env}(g, g')$$
$$\ldots$$
$$\mathit{inv_1}(g, l_1) \wedge \mathit{inv_2}(g, l_2) \rightarrow \mathit{safe}(v)$$

Clauses for preservation of $\mathit{inv_2}(v)$ are symmetric

Quantifier Free Horn Clauses

$$\forall v \; \forall w : \mathit{body}(v, w) \rightarrow \mathit{head}(v)$$

$\mathit{body}(v, w)$ and $\mathit{head}(v)$ are quantifier free



Sym(n_i) = p_i
Nodes := {n_T} ∪ ..
T := T+1
Sym(n) = p

Quantified Horn Clauses

- Existential temporal properties, e.g., CTL
- Program synthesis and infinite-state game solving
- Inference of transactions for concurrent programs

$$\forall v \; \forall w : \mathit{body}(v, w) \rightarrow \exists x : \mathit{head}(v, x)$$

- Quantified invariants/auxiliary assertions

$$\forall v \; \forall w : (\forall y : \mathit{body}(v, w, y)) \rightarrow \mathit{head}(v)$$

Existentially Quantified Horn Clauses

$$\forall v \ \forall w : body(v, w) \rightarrow \exists x : head(v, x)$$

$body(v, w)$ and $head(v, x)$ are quantifier free

Proving CTL Properties

$$(init(v), step(v, v')) \models EF(q(v))$$

$$(init(v), step(v, v')) \models EG(EU(p(v), q(v)))$$

Based on proof system for CTL* by Kesten and Pnueli [TCS'05]

Proving $EF(q(v))$

$$\exists inv(v)\ \exists round(v, v') :$$

$$\begin{aligned}
& init(v) \rightarrow inv(v) \\
& inv(v) \wedge \neg q(v) \rightarrow \exists v' : step(v, v') \\
& \qquad\qquad\qquad\qquad\qquad\quad \wedge\ inv(v') \\
& \qquad\qquad\qquad\qquad\qquad\quad \wedge\ round(v, v') \\
& wf(round(v, v'))
\end{aligned}$$

Decomposing $EG(EU(p(v), q(v)))$

$$(init(v), step(v, v')) \models EG(EU(p(v), q(v)))$$

iff

$\exists mid(v)$:

$$(init(v), step(v, v')) \models EG(mid(v))$$
$$(mid(v), step(v, v')) \models EU(p(v), q(v))$$

Proving $(init(v), step(v, v')) \models EG(mid(v))$ and $(mid(v), step(v, v')) \models EU(p(v), q(v))$

$$\exists mid(v)\ \exists inv_1(v)\ \exists inv_2(v)\ \exists round(v, v') :$$

$$init(v) \rightarrow inv_1(v)$$
$$inv_1(v) \rightarrow mid(v) \wedge \exists v' : step(v, v') \wedge inv_1(v')$$

$$mid(v) \rightarrow inv_2(v)$$
$$inv_2(v) \wedge \neg q(v) \rightarrow p(v) \wedge \exists v' : step(v, v') \wedge inv_2(v') \wedge round(v, v')$$
$$wf(round(v, v'))$$

Solving Infinite-State Game

Given five empty bottles arranged in circle and jar full of water

- Stepmother pours all water from jar into some bottles
- Cinderella empties pair of adjucent bottles
- Jar is refilled for next round

Stepmother wins if some bottle overflows

Formalization of Game Arena

- $v = (v_1, \ldots, v_5)$
- B - bottle volume
- J - jar volume

$$
\begin{aligned}
init(v) \quad &= (v_1 = \cdots = v_5 = 0) \\
cindy(v, v') &= (v'_1 = v'_2 = 0 \wedge same(v_3, v_4, v_5) \vee \\
&\quad \cdots \\
&\quad \vee\ v'_5 = v'_1 = 0 \wedge same(v_2, v_3, v_4)) \\
step(v, v') \ &= (v'_1 \geq v_1 \wedge \cdots \wedge v'_5 \geq v_5 \wedge \\
&\quad v'_1 + \cdots + v'_5 - (v_1 + \cdots + v_5) = J) \\
over(v) \quad &= (v_1 > B \vee \cdots \vee v_5 > B)
\end{aligned}
$$

Stepmother's Victory as Constraint Satisfaction

$$\exists win(v)\ \exists round(v, v') :$$

$$init(v) \rightarrow win(v)$$

$$win(v) \wedge \neg over(v) \wedge cindy(v, v') \rightarrow \exists v'' : step(v', v'') \wedge win(v'') \wedge round(v, v'')$$

$$wf(round(v, v'))$$

Inference of Transactions

- $v = (g, l_1, l_2)$ - global and thread-local variables
- $init(v)$ - initial states
- $safe(v)$ - safe states

Inference of Transactions

- $v = (g, l_1, l_2)$ - global and thread-local variables
- $init(v)$ - initial states
- $safe(v)$ - safe states
- $a(v, v'), b(v, v')$ - transitions of 1st thread

Inference of Transactions

- $v = (g, l_1, l_2)$ - global and thread-local variables
- $init(v)$ - initial states
- $safe(v)$ - safe states
- $a(v, v'), b(v, v')$ - transitions of 1st thread
- $p(v, v'), q(v, v')$ - transitions of 2nd thread

Transaction Inference Constraints

- Abstract transitions
- Commutativity
- Transactions
- Invariance

Abstract Transitions: Thread 1

$$inv(v) \land a(v, v') \rightarrow \hat{a}(v, v')$$

Abstract Transitions: Thread 1

$$inv(v) \wedge a(v, v') \rightarrow \hat{a}(v, v')$$

$$inv(v) \wedge \hat{a}(v, v') \wedge b(v', v'') \rightarrow \hat{b}(v', v'')$$

Commutativity: Thread 1

$\hat{a}$ is a right mover and $\hat{b}$ is a left mover

$$inv(v) \wedge \hat{a}(v, v') \wedge \hat{p}(v', v'') \rightarrow \exists v''' : \hat{p}(v, v''') \wedge \hat{a}(v''', v'')$$

Commutativity: Thread 1

$\hat{a}$ is a right mover and $\hat{b}$ is a left mover

$$inv(v) \wedge \hat{a}(v, v') \wedge \hat{p}(v', v'') \rightarrow \exists v''' : \hat{p}(v, v''') \wedge \hat{a}(v''', v'')$$

$$inv(v) \wedge \hat{a}(v, v') \wedge \hat{q}(v', v'') \rightarrow \exists v''' : \hat{q}(v, v''') \wedge \hat{a}(v''', v'')$$

Commutativity: Thread 1

$\hat{a}$ is a right mover and $\hat{b}$ is a left mover

$$inv(v) \wedge \hat{a}(v, v') \wedge \hat{p}(v', v'') \rightarrow \exists v''' : \hat{p}(v, v''') \wedge \hat{a}(v''', v'')$$

$$inv(v) \wedge \hat{a}(v, v') \wedge \hat{q}(v', v'') \rightarrow \exists v''' : \hat{q}(v, v''') \wedge \hat{a}(v''', v'')$$

$$inv(v) \wedge \hat{p}(v, v') \wedge \hat{b}(v', v'') \rightarrow \exists v''' : \hat{b}(v, v''') \wedge \hat{p}(v''', v'')$$

Commutativity: Thread 1

$\hat{a}$ is a right mover and $\hat{b}$ is a left mover

$$inv(v) \land \hat{a}(v, v') \land \hat{p}(v', v'') \rightarrow \exists v''' : \hat{p}(v, v''') \land \hat{a}(v''', v'')$$

$$inv(v) \land \hat{a}(v, v') \land \hat{q}(v', v'') \rightarrow \exists v''' : \hat{q}(v, v''') \land \hat{a}(v''', v'')$$

$$inv(v) \land \hat{p}(v, v') \land \hat{b}(v', v'') \rightarrow \exists v''' : \hat{b}(v, v''') \land \hat{p}(v''', v'')$$

$$inv(v) \land \hat{q}(v, v') \land \hat{b}(v', v'') \rightarrow \exists v''' : \hat{b}(v, v''') \land \hat{q}(v''', v'')$$

Transactions

$$inv(v) \wedge \hat{a}(v, v') \wedge \hat{b}(v', v'') \to \widehat{ab}(v, v'')$$

Transactions

$$inv(v) \land \hat{a}(v, v') \land \hat{b}(v', v'') \rightarrow \widehat{ab}(v, v'')$$

$$inv(v) \land \hat{p}(v, v') \land \hat{q}(v', v'') \rightarrow \widehat{pq}(v, v'')$$

Invariance

$$init(v) \rightarrow inv(v)$$

Invariance

$$init(v) \rightarrow inv(v)$$

$$inv(v) \wedge \widehat{ab}(v, v') \rightarrow inv(v')$$

Invariance

$$init(v) \rightarrow inv(v)$$

$$inv(v) \land \widehat{ab}(v, v') \rightarrow inv(v')$$

$$inv(v) \land \widehat{pq}(v, v') \rightarrow inv(v')$$

Invariance

$$init(v) \rightarrow inv(v)$$

$$inv(v) \wedge \widehat{ab}(v, v') \rightarrow inv(v')$$

$$inv(v) \wedge \widehat{pq}(v, v') \rightarrow inv(v')$$

$$inv(v) \rightarrow safe(v)$$

Universally Quantified Horn Clauses

$$\forall v \; \forall w : (\forall y : body(v, w, y)) \rightarrow head(v)$$

$body(v, w, y)$ and $head(v)$ are quantifier free

Vefication with Universally Quantified Invariants

```
for(i = 0; i < n; i++) { a[i] = i; }
assert("forall p: i <= p && p < n -> a[p] == p");
```

State-of-the-art recipe (e.g., Gopan et al. POPL'05,Gulwani et al. POPL'08, Halbwachs et al. PLDI'08, Dillig et al. ESOP'10, Logozzo et al. POPL'11, Alberti et al. CAV'12, Larraz et al. VMCAI'13):

- Quantification template
- Instantiation template
- Shape template with abstract domains

Templates for Universally Quantified Invariants

- Quantification template

$$\forall p : inv(i, n, p, a(p))$$

- Instantiation template

$$inv(i, n, e_1, a(e_1)) \wedge \cdots \wedge inv(i, n, e_k, a(e_k))$$

- Shape template with abstract domains

$$inv(i, n, p, a(p)) = (guard(i, n, p) \rightarrow property(i, n, p, a(p)))$$

*under-approximation for $guard(i, n, p)$

Universally Quantified Clauses

```
for(i = 0; i < n; i++) { a[i] = i; }
assert("forall p: i <= p && p < n -> a[p] == p");
```

- Quantification template $\forall p : inv(i, n, p, a(p))$

$$\exists inv(i, n, p, \overbrace{v}^{a(p)}) :$$

$$i = 0 \rightarrow (\forall p : inv(i, n, p, a(p)))$$

$$(\forall p : inv(i, n, p, a(p))) \wedge i < n \wedge i' = i + 1 \wedge a' = a\{i := i\} \rightarrow$$
$$(\forall q : inv(i', n, q, a'(q)))$$

$$(\forall p : inv(i, n, p, a(p))) \rightarrow (\forall q : 0 \leq q < n \rightarrow a(q) = q)$$

Universally Quantified Clauses

```
for(i = 0; i < n; i++) { a[i] = i; }
assert("forall p: i <= p && p < n -> a[p] == p");
```

- Quantification template $\forall p : inv(i, n, p, a(p))$

$$\exists inv(i, n, p, \overbrace{v}^{a(p)}) :$$

$$i = 0 \rightarrow inv(i, n, p, a(p))$$

$$(\forall p : inv(i, n, p, a(p))) \wedge i < n \wedge a' = a\{i := i\} \rightarrow inv(i + 1, n, q, a'(q))$$

$$(\forall p : inv(i, n, p, a(p))) \wedge 0 \leq q < n \rightarrow a(q) = q$$

Quantifier Instantiation Heuristic

Instantiation Constraint Generation

- If $a(p)$ occcurs in clause then $p \in inst(a)$
- If $a' = a\{\cdot := \cdot\}$ occcurs in clause then $inst(a') \subseteq inst(a)$

Example

- Clause

$$(\forall p : inv(i, n, p, a(p))) \land i < n \land a' = a\{i := i\}$$
$$\rightarrow inv(i + 1, n, q, a'(q))$$

- Instantiation constaints $q \in inst(a')$ and $inst(a') \subseteq inst(a)$
- Instantiation solution $inst(a) = inst(a') = \{q\}$

Quantifier Instantiation Validation

$inv(i, n, p, v) = (0 \leq p < i \rightarrow v = p)$ and $inst(a) = inst(a') = \{q\}$

$$
\begin{aligned}
&(0 \leq q < i \rightarrow q = a(q)) \\
&\wedge\ i < n \\
&\wedge\ a'(i) = i \wedge (q \neq i \rightarrow a'(q) = a(q)) \\
&\rightarrow (0 \leq q < i + 1 \rightarrow a'(q) = q)
\end{aligned}
$$

Instantiation Based Proof

$$(0 \leq q < i \rightarrow a(q) = q) \wedge i < n \wedge a'(i) = i \wedge (q \neq i \rightarrow a'(q) = a(q))$$

$$(0 \leq q < i + 1 \rightarrow a'(q) = q)$$

Instantiation Based Proof

$$(0 \leq q < i \rightarrow a(q) = q) \wedge i < n \wedge a'(i) = i \wedge (q \neq i \rightarrow a'(q) = a(q))$$

$$0 \leq q < i + 1$$

$$a'(q) = q$$

$$(0 \leq q < i + 1 \rightarrow a'(q) = q)$$

Instantiation Based Proof

$$(0 \leq q < i \rightarrow a(q) = q) \wedge i < n \wedge a'(i) = i \wedge (q \neq i \rightarrow a'(q) = a(q))$$

$$0 \leq q < i + 1$$

$$q \neq i \qquad\qquad q = i$$

$$a'(q) = q$$

$$(0 \leq q < i + 1 \rightarrow a'(q) = q)$$

Instantiation Based Proof

$$(0 \leq q < i \rightarrow a(q) = q) \wedge i < n \wedge a'(i) = i \wedge (q \neq i \rightarrow a'(q) = a(q))$$

$0 \leq q < i + 1$

$q \neq i$ $\qquad\qquad$ $q = i$

$a(q) = q$

$a'(q) = q$

$(0 \leq q < i + 1 \rightarrow a'(q) = q)$

Instantiation Based Proof

$$(0 \leq q < i \rightarrow a(q) = q) \wedge i < n \wedge a'(i) = i \wedge (q \neq i \rightarrow a'(q) = a(q))$$

$0 \leq q < i + 1$

$q \neq i$ | $q = i$

$a(q) = q$

$a'(q) = a(q)$

$a'(q) = q$

$$(0 \leq q < i + 1 \rightarrow a'(q) = q)$$

Instantiation Based Proof

$$(0 \leq q < i \to a(q) = q) \wedge i < n \wedge a'(i) = i \wedge (q \neq i \to a'(q) = a(q))$$

$$0 \leq q < i + 1$$

$$q \neq i \qquad\qquad\qquad q = i$$

$$a(q) = q$$

$$a'(q) = a(q)$$

$$a'(q) = q$$

$$a'(q) = q$$

$$(0 \leq q < i + 1 \to a'(q) = q)$$

Instantiation Based Proof

$$(0 \leq q < i \rightarrow a(q) = q) \wedge i < n \wedge a'(i) = i \wedge (q \neq i \rightarrow a'(q) = a(q))$$

$0 \leq q < i + 1$

$q \neq i$	$q = i$
$a(q) = q$	$a'(q) = q$
$a'(q) = a(q)$	
$a'(q) = q$	

$a'(q) = q$

$$(0 \leq q < i + 1 \rightarrow a'(q) = q)$$

Universally Quantified Invariant for Termination

```
for(i = 0; i < n; i++) {
  a[i] = 1;
}
while (x > 0) {
  for(i = 0; i < n; i++) {
    x = x-a[i];
  }
}
```

$$
\begin{aligned}
inv_1(i, n, x, p, v) &= (0 \leq p < i \rightarrow v \geq 1) \\
inv_2(i, n, x, p, v) &= (0 \leq x \wedge 0 \leq i < n \wedge (0 \leq p < n \rightarrow v \geq 1)) \\
&= inv_3(i, n, x, p, v)
\end{aligned}
$$

Further Pointers

- Solving recursion-free clauses over LI+UIF, [APLAS'11]
- Solving quantifier free clauses and well-foundedness, [PLDI'12]
- Solving existentially quantified clauses: [CAV'13]
- Solving universally quantified clauses: [SAS'13]
- Proof rules for multi-threaded programs [POPL'11, CAV'11, TACAS'12]
- Proof rules for functional programs [CAV'11, SAS'12]
- Software verification competition [SV-COMP'12, SV-COMP'13]
- Separation logic modulo theories [APLAS'13]
- A Constraint-Based Approach to Solving Games on Infinite Graphs [POPL'2014]