Higher-order Constrained Horn Clauses and Automatic Program Verification

Luke Ong

(Joint with: Steven Ramsay[†], Toby Cathcart Burn, Jerome Jochems, Long Pham, and Dominik Wagner)

University of Bristol[†] and University of Oxford

Logic Mentoring Workshop, 8 July 2018

Motivation: two developments in formal verification

1. **Model checking** (from 1980s)—an approach to verification that promises accurate analysis with push-button automation—has been a successful application of logic to computer science.

- 2007 ACM Turing Award (Clarke, Emerson and Sifakis) "for their rôle in developing model checking into a highly effective verification technology, widely adopted in hardware and software industries".
- Higher-order model checking (from 2000s) has had some success in both theory and practice: Carayol, Hague, K-N-U, Kobayashi, Mellies, O., Ramsay, Salvati, Serre, Tsukada, Unno, Walukiewicz, etc.

2. "**Constrained Horn clauses** are a suitable basis for automatic program verification" (Bjørner, McMillan & Rybalchenko 2012)

- Expressive framework, promoting separation of concerns (PL specifics vs purely logical), exploiting the phenomenal efficiency of SMT solvers.

Outline

1. Higher-order constrained Horn clauses (HoCHC): satisfiability and safety problems
2. Semantics of higher-order logic (standard / monotone / continuous) and least model property
3. Solutions of HoCHC systems 1 & 2, and automation via prototype tools DefMono & Horus
4. Conclusion and future directions

A simple functional program

let $\mathit{add}\ x\ y \ = \ x + y$
letrec $\mathit{iter}\ f\ s\ n \ = $ if $n \leq \mathit{0}$ then s else $f\ n\ (\mathit{iter}\ f\ s\ (n - \mathit{1}))$
in $\mathit{iter}\ \mathit{add}\ \mathit{2}\ \mathit{2}$

- $(\mathit{iter}\ f\ s\ n)$ computes $f\,n\,\big(f\,(n-1)\,(f\,(n-2)\,(\cdots(f\,1\,s)\cdots))\big)$.

Running the program:

$$\begin{aligned} \mathit{iter}\ \mathit{add}\ \mathit{2}\ \mathit{2} &\to \mathit{add}\ \mathit{2}\ (\mathit{iter}\ \mathit{add}\ \mathit{2}\ \mathit{1}) \\ &\to^* \mathit{2} + (\mathit{add}\ \mathit{1}\ (\mathit{iter}\ \mathit{add}\ \mathit{2}\ \mathit{0})) \\ &\to^* \mathit{2} + (\mathit{1} + \mathit{2}) \\ &\to^* 5 \end{aligned}$$

Verification task: Prove: $\forall n \geq 2 \,.\, (\mathit{iter}\ \mathit{add}\ n\ n) > n + n$.

Question. Does it hold for $n = 1$?

Our approach: translation into **higher-order logic**

let $\mathit{add}\ x\ y = x + y$
letrec $\mathit{iter}\ f\ s\ n =$ if $n \leq \mathit{0}$ then s else $f\ n\ (\mathit{iter}\ f\ s\ (n - \mathit{1}))$
in $\mathit{iter}\ \mathit{add}\ \mathit{2}\ \mathit{2}$

Transform each function (iter) to its *inductive invariant* (Iter) i.e. a relation over-approximating (containing) its graph.

$$D \begin{cases} \forall x, y, z\, .\, \big(z = x + y \rightarrow \mathit{Add}\ x\ y\ z\big) \\ \forall f, s, n, r\, .\, \big(n \leq 0 \wedge r = s \rightarrow \mathit{Iter}\ f\ s\ n\ r\big) \\ \forall f, s, n, r\, .\, \big(n > 0 \wedge (\exists p.\ \mathit{Iter}\ f\ s\ (n - \mathit{1})\ p \wedge f\ n\ p\ r) \rightarrow \mathit{Iter}\ f\ s\ n\ r\big) \end{cases}$$

with goal

$$G \qquad \forall n, r\, .\, \big(n > 1 \wedge \mathit{Iter}\ \mathit{Add}\ n\ n\ r \rightarrow r > n + n\big)$$

Idea. We wish to prove: "canonical model" of D is a model of G.

Is this a sensible algorithmic approach to verification?

- First-order logic is undecidable but semi-decidable: the validities are computably enumerable.
  - (Tarski) $\mathbf{V}^1(=)$ is decidable; $\mathbf{V}^1(P(\text{-},\text{-}))$ is r.e.
  - If a formula is unsatisfiable then it is provable by resolution (Davis & Putnam 1960; Robinson 1965).
- Second-order logic (standard semantics) is not even semi-decidable.
  - $\mathbf{V}^2(=)$ is not *analytical* (not definable in arithmetic by any 2nd-order formula), let alone *arithmetical*!

Bad news? An **aside.** Why not consider higher-order logic in general (Henkin) semantics (which is nothing but many-sorted 1st-order logic with comprehension axioms)?

$\because$ Standard semantics is simple and natural; it is widely adopted in verification (e.g. Gordon's HOL, and MSO logic)

Some syntactic features

$$D \begin{cases} \forall x, y, z. \big(z = x + y \rightarrow \mathit{Add}\ x\ y\ z\big) \\ \forall f, s, n, r. \big(n \leq 0 \wedge r = s \rightarrow \mathit{Iter}\ f\ s\ n\ r\big) \\ \forall f, s, n, r. \big(n > 0 \wedge (\exists p.\ \mathit{Iter}\ f\ s\ (n - \mathit{1})\ p \wedge f\ n\ p\ r) \rightarrow \mathit{Iter}\ f\ s\ n\ r\big) \end{cases}$$

$$G \qquad \forall n, r\,.\, \big(n > 1 \wedge \mathit{Iter}\ \mathit{Add}\ n\ n\ r \rightarrow r > n + n\big)$$

- Higher-order relations (predicates):

  $\mathit{Iter} : (\mathsf{int} \rightarrow \mathsf{int} \rightarrow \mathsf{int} \rightarrow \mathsf{bool}) \rightarrow \mathsf{int} \rightarrow \mathsf{int} \rightarrow \mathsf{int} \rightarrow \mathsf{bool}$

- Quantification at higher types: $f : \mathsf{int} \rightarrow \mathsf{int} \rightarrow \mathsf{int} \rightarrow \mathsf{bool}$

- Literals may be headed by variables: $f\ n\ p\ m$

- Each D-clause is *definitional Horn* (:= at most one positive literal, which has "definitional form" $R\, x_1 \cdots x_n$). This restriction turns out to be the saving grace!

Higher-order constrained Horn clauses (HoCHC)

Constrained means truth of formula is relative to a decidable 1st-order background theory with vocab. Σ (e.g. LIA).

Higher-order relational types: $\sigma ::= \mathsf{int} \to \mathsf{bool} \mid \mathsf{int} \to \sigma \mid \sigma \to \sigma'$

Fix vocab. Δ of higher-order "free" relational symbols.

$$\begin{array}{rrcl} \text{goal} & G & ::= & A \mid \varphi \mid G \wedge G \mid G \vee G \mid \exists x{:}\sigma.\, G \\ \text{definite} & D & ::= & \mathsf{true} \mid \forall x{:}\sigma.\, D \mid D \wedge D \mid G \to R\, x_1 \ldots x_n \end{array}$$

- A ranges over foreground formulas e.g. $\mathit{Iter}\, f\, m\, (n-1)\, p$, $f\, n\, p\, r$
- φ ranges over constraints (background formulas) e.g. $x \leq 0$
- R ranges over Δ e.g. Iter

Satisfiability Problem $\langle \Delta, D \rangle$ is solvable if for all models $\mathcal{A}$ of background theory Th, there are Δ-expansion $\mathcal{B}$ and valuation α s.t. $\mathcal{B}, \alpha \models D$.

- Satisfiability Problem $\langle \Delta, D \rangle$ is solvable if for all models $\mathcal{A}$ of background theory Th, there are Δ-expansion $\mathcal{B}$ and valuation α s.t. $\mathcal{B}, \alpha \vDash D$.
- Safety Problem $\langle \Delta, D, G \rangle$ is solvable if for all models $\mathcal{A}$ of Th, there are Δ-expansion $\mathcal{B}$ and valuation α s.t. $\mathcal{B}, \alpha \vDash D$, yet $\mathcal{B}, \alpha \nvDash G$.

Standard semantics of higher-order logic

Syntax: standard presentation as a simply-typed λ-calculus with

- types: $\sigma \ ::= \ \mathsf{one} \mid \mathsf{bool} \mid \mathsf{int} \mid \sigma_1 \to \sigma_2$ (Bkgrd theory: LIA)
- logical constants: $\neg, \wedge, \vee, \forall_\sigma, \exists_\sigma$, etc.

$$\neg : \mathsf{bool} \to \mathsf{bool} \qquad \forall_\sigma, \exists_\sigma : (\sigma \to \mathsf{bool}) \to \mathsf{bool}$$

Write $\exists_\sigma(\lambda x{:}\sigma.\, M)$ as $\exists x{:}\sigma.\, M$.

Semantics: standard!

$$\begin{array}{rcl} \mathcal{S}[\![\mathsf{one}]\!] & := & \{\star\} \\ \mathcal{S}[\![\mathsf{bool}]\!] & := & \{0, 1\} \\ \mathcal{S}[\![\mathsf{int}]\!] & := & \mathbb{Z} \\ \mathcal{S}[\![\sigma_1 \to \sigma_2]\!] & := & \mathcal{S}[\![\sigma_1]\!] \to \mathcal{S}[\![\sigma_2]\!] \quad (\textit{all}\ \text{functions}) \end{array}$$

Example: $\mathcal{A} \ \models_{\mathcal{S}} \ \exists x : \big((\mathsf{int} \to \mathsf{bool}) \to \mathsf{bool}\big)\, .\, G$

"There is some predicate x, on sets of integers, that makes G true in $\mathcal{A}$."

Failure of least model property in standard semantics!

Counterexample:

$$\begin{cases} P & : & ((\mathsf{one} \to \mathsf{bool}) \to \mathsf{bool}) \to \mathsf{bool} \\ Q & : & \mathsf{one} \to \mathsf{bool} \end{cases}$$

$$\forall x : (\mathsf{one} \to \mathsf{bool}) \to \mathsf{bool}\, .\, \big(x\ Q \to P\ x \big)$$

Question. Does Q occur positively in $x\,Q$?

Theorem

Satisfiable systems of higher-order constrained Horn clauses do not necessarily have (unique) least models.

(Least with respect to inclusion of relations.)

$\forall x.\,(x\ Q \to P\ x)$ has 2 minimal models: $\mathcal{B}$, $\mathcal{C}$

$P : ((\mathsf{one} \to \mathsf{bool}) \to \mathsf{bool}) \to \mathsf{bool}$ $\qquad$ $Q : \mathsf{one} \to \mathsf{bool}$

$$\mathcal{S}[\![\mathsf{one}]\!] \coloneqq \{\star\}$$
$$\mathcal{S}[\![\mathsf{one} \to \mathsf{bool}]\!] \coloneqq \{ \underbrace{\{\star \mapsto 0\}}_{-}, \quad \underbrace{\{\star \mapsto 1\}}_{+} \}$$

$\mathcal{S}[\![(\mathsf{one} \to \mathsf{bool}) \to \mathsf{bool}]\!] \coloneqq$

$$\left\{ \underbrace{\left\{ \begin{array}{c} - \mapsto 0 \\ + \mapsto 1 \end{array} \right\}}_{\mathrm{id}}, \underbrace{\left\{ \begin{array}{c} - \mapsto 0 \\ + \mapsto 0 \end{array} \right\}}_{\mathrm{cst0}}, \underbrace{\left\{ \begin{array}{c} - \mapsto 1 \\ + \mapsto 1 \end{array} \right\}}_{\mathrm{cst1}}, \underbrace{\left\{ \begin{array}{c} - \mapsto 1 \\ + \mapsto 0 \end{array} \right\}}_{\mathrm{neg}} \right\}$$

$$\begin{array}{rcl} \mathcal{B}(Q)(\star) & = & 0 \\ \mathcal{B}(P)(\mathrm{id}) & = & 0 \\ \mathcal{B}(P)(\mathrm{cst0}) & = & 0 \\ \mathcal{B}(P)(\mathrm{cst1}) & = & 1 \\ \mathcal{B}(P)(\mathrm{neg}) & = & 1 \end{array} \qquad \begin{array}{rcl} \mathcal{C}(Q)(\star) & = & 1 \\ \mathcal{C}(P)(\mathrm{id}) & = & 1 \\ \mathcal{C}(P)(\mathrm{cst0}) & = & 0 \\ \mathcal{C}(P)(\mathrm{cst1}) & = & 1 \\ \mathcal{C}(P)(\mathrm{neg}) & = & 0 \end{array}$$

Monotone semantics of higher-order logic

$\mathcal{M}$ interpret $\to$ as the monotone function space.

$$\begin{array}{rcl} \mathcal{M}[\![\mathsf{int}]\!] & := & \text{poset } \mathbb{Z} \text{ (ordered discretely)} \\ \mathcal{M}[\![\mathsf{bool}]\!] & := & \text{poset } \{0,1\} \text{ with } 0 \sqsubseteq 1 \\ \mathcal{M}[\![\sigma_1 \to \sigma_2]\!] & := & \mathcal{M}[\![\sigma_1]\!] \to_{\mathrm{m}} \mathcal{M}[\![\sigma_2]\!] \quad \text{(monotone fns)} \end{array}$$

Example: $\mathcal{A} \vDash_{\mathcal{M}} \exists x : \big((\mathsf{int} \to \mathsf{bool}) \to \mathsf{bool}\big) \,.\, G$

> "There is some monotone predicate x, on sets of integers, that makes G true in $\mathcal{A}$."

In monotone semantics, satisfiable Horn clauses have least models (because "immediate consequence operator" is monotone) and constructible by Knaster-Tarski Fixpoint Theorem.

Examples

$\mathcal{M}[\![(\mathsf{int} \to \mathsf{bool}) \to \mathsf{bool}]\!]$	All upward-closed (w.r.t. $\subseteq$) sets of sets of integers
$\mathcal{M}[\![((\mathsf{int} \to \mathsf{bool}) \to \mathsf{bool}) \to \mathsf{bool}]\!]$	All upward-closed sets of upward-closed sets of sets of integers

$\mathcal{M}[\![-]\!]$ is counter-intuitive!

Take $x : (\mathsf{int} \to \mathsf{bool}) \to \mathsf{bool}$, and

$$\varphi \;:=\; \exists y : (\mathsf{int} \to \mathsf{bool}) \,.\, \exists z : \mathsf{int} \,.\, \big(x\, y \wedge y\, z\big)$$

which means: "x contains a nonempty set".

Thus "$x \mapsto \{\{1\}\} \vDash \varphi$" should hold, but doesn't $\because$ the valuation is invalid i.e.

$$\{\{1\}\} \;\not\in\; \mathcal{M}[\![(\mathsf{int} \to \mathsf{bool}) \to \mathsf{bool}]\!])$$

"Each is good for something"

Standard Semantics

☺ Completely standard satisfiability problem (modulo background theory) in higher-order logic.

☹ No least model.

Monotone Semantics

☹ Bespoke satisfiability problem with a restricted class of models.

☺ Least model arising in the usual way.

Can we have the best of both worlds?

I.e. can we specify verification problems in standard semantics, but solve / compute them in monotone semantics?

Standard and monotone semantics are equivalent for the HoCHC Satisfiability Problem

We *can* have the best of both worlds!

Theorem (Model correspondence)

Given a definite clause H, H is satisfiable in the standard semantics iff H is satisfiable in the monotone semantics.

Proof idea

For each relational type ρ, monotone and standard semantics are locked in two-sided **adjunctions** (or Galois connections):

$$\mathcal{S}[\![\rho]\!] \;\underset{\mathsf{L}_\rho}{\overset{\mathsf{I}_\rho}{\leftrightarrows}}\; \mathcal{M}[\![\rho]\!] \;\underset{\mathsf{J}_\rho}{\overset{\mathsf{U}_\rho}{\leftrightarrows}}\; \mathcal{S}[\![\rho]\!]$$

Standard   Monotone   Standard

Define, by recursion over types:

$$\begin{aligned} \mathsf{I}_{\mathsf{bool}}(b) &:= b & \qquad \mathsf{J}_{\mathsf{bool}}(b) &:= b \\ \mathsf{I}_{\mathsf{int}\to\rho}(r) &:= \mathsf{I}_\rho \circ r & \qquad \mathsf{J}_{\mathsf{int}\to\rho}(r) &:= \mathsf{J}_\rho \circ r \\ \mathsf{I}_{\rho_1\to\rho_2}(r) &:= \mathsf{I}_{\rho_2} \circ r \circ \mathsf{L}_{\rho_1} & \qquad \mathsf{J}_{\rho_1\to\rho_2}(r) &:= \mathsf{J}_{\rho_2} \circ r \circ \mathsf{U}_{\rho_1} \end{aligned}$$

where

- U_ρ is the right adjoint of J_ρ. Thanks to Adjunct Functor Theorem, each is uniquely determined by the other, via:

$$\forall a, b \,.\, \big(J_\rho\, a \subseteq b \;\Leftrightarrow\; a \subseteq U_\rho\, b\big)$$

- Similarly, I_ρ is the right adjoint of L_ρ.

Standard and monotone semantics are equivalent also for HoCHC Safety Problem

Theorem (Equivalence 1)

Fix a background theory Th of vocabulary Σ. For all Δ, D and G, T.F.A.E.

(i) *HoCHC Safety Problem $\langle \Delta, D, G \rangle$ in standard semantics is solvable*

(ii) *HoCHC Safety Problem $\langle \Delta, D, G \rangle$ in monotone semantics is solvable*

(iii) *For all models $\mathcal{A}$ of the background theory Th, $\mathcal{B}, \alpha \vDash D$, yet $\mathcal{B}, \alpha \nvDash G$, where $\mathcal{B}$ is the least Δ-expansion of Σ that models D.*

Thus: we can specify problems using the standard semantics, but solve them in the monotone semantics.

Monotone and continuous semantics are equivalent also for HoCHC Safety Problem

Theorem (Equivalence 2)

For all Δ, D and G, T.F.A.E.

(i) *HoCHC Safety Problem $\langle \Delta, D, G \rangle$ in monotone semantics is solvable*

(ii) *HoCHC Safety Problem $\langle \Delta, D, G \rangle$ in continuous semantics is solvable*

(Continuity is w.r.t. Scott topology of the relevant poset.)

Thus: we can specify problems using the standard semantics, and then solve them in either the monotone or continuous semantics.

Each semantics is useful in its own way (more anon).

Solving HoCHC systems 1: refinement types

- Sound but incomplete. (Cathcart Burn, O. & Ramsay, POPL18)
- Web interface to Horus: `http://mjolnir.cs.ox.ac.uk/horus`

Tests

Verification problems taken from MoCHi test suite (Kobayashi et al. PLDI'11) but rexpressed as HoCHC safety problems.

In all the examples (without local assertions), except `neg`:

- Horus takes around 0.01s to transform the system of clauses and
- Z3 takes around 0.02s to solve the transfromed 1st-order system.

Example. In Problem `mc91` (McCarthy's 91 function), we verify:

$$\forall n \in \mathbb{Z} \,.\, \big(n \leq 101 \to M(n) = 91\big).$$

Solving HoCHC systems 2: Reynolds' defunctionalisation

Reynolds' defunctionalisation (1972). Given a simply-typed closed term M of base type, there is effectively a first-order term $\mathsf{defun}(M)$ such that $[\![M]\!] = [\![\mathsf{defun}(M)]\!]$.

Theorem. Given a HoCHC safety problem instance $\mathcal{I} = \langle \Delta, D, G \rangle$, the defunctionalisation algorithm:

(i) constructs a first-order (well-typed) constrained Horn clause problem instance $\mathcal{I}'$

(ii) moreover, $\mathcal{I}$ is solvable iff $\mathcal{I}'$ is solvable.

(Proof uses continuous semantics of HoL.)

DefMono: `http://mjolnir.cs.ox.ac.uk/dfhochc`
- a prototype implementation that first defunctionalises an HoCHC problem instance and then feeds it to a backend SMT solver.

We believe that HoCHC are an easy-to-use, efficient, and expressively adequate framework for the analysis and verification of higher-order programs.

(Cathcart Burn, O. & Ramsay: POPL 2018)

Future directions

- Construct prototypical implementation of the HoCHC resolution proof system.

- Find extensions of the HoCHC fragment which is still semi-decidable.

- Find interesting decidable fragments (using finite instantiations or ordered resolution / superposition).