Solving Constrained Horn Clauses over ADTs by Finite Model Finding[1]

Yurii Kostyukov[1] Dmitry Mordvinov[1] Grigory Fedyukovich[2]

[1]Saint Petersburg State University
JetBrains Research

[2]Florida State University

March 28, 2021

[1](conditionally) accepted at PLDI'21

Motivating Example: STLC Type Inhabitation

STLC Typing Rules

$$\frac{x{:}T \in \Gamma}{\Gamma \vdash x : T} \quad \text{(T-VAR)}$$

$$\frac{\Gamma, x{:}T_1 \vdash t_2 : T_2}{\Gamma \vdash \lambda x{:}T_1 . t_2 : T_1 \to T_2} \quad \text{(T-ABS)}$$

$$\frac{\Gamma \vdash t_1 : T_{11} \to T_{12} \qquad \Gamma \vdash t_2 : T_{11}}{\Gamma \vdash t_1\ t_2 : T_{12}} \quad \text{(T-APP)}$$

Motivating Example: STLC Type Inhabitation

STLC Typing Rules

$$\frac{\mathtt{x}:\mathtt{T} \in \Gamma}{\Gamma \vdash \mathtt{x} : \mathtt{T}} \quad \text{(T-VAR)}$$

$$\frac{\Gamma, \mathtt{x}:\mathtt{T}_1 \vdash \mathtt{t}_2 : \mathtt{T}_2}{\Gamma \vdash \lambda \mathtt{x}:\mathtt{T}_1 . \mathtt{t}_2 : \mathtt{T}_1 \to \mathtt{T}_2} \quad \text{(T-ABS)}$$

$$\frac{\Gamma \vdash \mathtt{t}_1 : \mathtt{T}_{11} \to \mathtt{T}_{12} \qquad \Gamma \vdash \mathtt{t}_2 : \mathtt{T}_{11}}{\Gamma \vdash \mathtt{t}_1\ \mathtt{t}_2 : \mathtt{T}_{12}} \quad \text{(T-APP)}$$

Type Inhabitation

- Is there a term of type $(a \to a) \to a$?

Motivating Example: STLC Type Inhabitation

STLC Typing Rules

$$\frac{\mathtt{x}:\mathtt{T} \in \Gamma}{\Gamma \vdash \mathtt{x} : \mathtt{T}} \qquad \text{(T-VAR)}$$

$$\frac{\Gamma, \mathtt{x}:\mathtt{T}_1 \vdash \mathtt{t}_2 : \mathtt{T}_2}{\Gamma \vdash \lambda \mathtt{x}:\mathtt{T}_1.\mathtt{t}_2 : \mathtt{T}_1 \to \mathtt{T}_2} \qquad \text{(T-ABS)}$$

$$\frac{\Gamma \vdash \mathtt{t}_1 : \mathtt{T}_{11} \to \mathtt{T}_{12} \qquad \Gamma \vdash \mathtt{t}_2 : \mathtt{T}_{11}}{\Gamma \vdash \mathtt{t}_1\ \mathtt{t}_2 : \mathtt{T}_{12}} \qquad \text{(T-APP)}$$

Type Inhabitation

- Is there a term of type $(a \to a) \to a$?
- Is there a term e which for all atomic substitutions of free types in T has type T?

STLC Type Inhabitation as a CHC ADT Problem

STLC Type Inhabitation Example

$$
\begin{aligned}
\textcolor{red}{tc}(\Gamma, e, t) &\leftarrow \Gamma = cons(v, t, \Gamma') \land e = var(v) & \\
\textcolor{red}{tc}(\Gamma, e, t) &\leftarrow \Gamma = cons(v', t', \Gamma') \land \textcolor{red}{tc}(\Gamma', e, t) & \text{(T-VAR)} \\
\textcolor{red}{tc}(\Gamma, e, t) &\leftarrow e = abs(v, e') \land t = arrow(t', u) \land \textcolor{red}{tc}(cons(v, t', \Gamma), e', u) & \text{(T-ABS)} \\
\textcolor{red}{tc}(\Gamma, e, t) &\leftarrow e = app(e_1, e_2) \land \textcolor{red}{tc}(\Gamma, e_1, arrow(u, t)) \land \textcolor{red}{tc}(\Gamma, e_2, u) & \text{(T-APP)}
\end{aligned}
$$

STLC Type Inhabitation as a CHC ADT Problem

STLC Type Inhabitation Example

$$
\begin{aligned}
tc(\Gamma, e, t) &\leftarrow \Gamma = cons(v, t, \Gamma') \land e = var(v) \\
tc(\Gamma, e, t) &\leftarrow \Gamma = cons(v', t', \Gamma') \land tc(\Gamma', e, t)
\end{aligned} \qquad \text{(T-VAR)}
$$

$$
tc(\Gamma, e, t) \leftarrow e = abs(v, e') \land t = arrow(t', u) \land tc(cons(v, t', \Gamma), e', u) \qquad \text{(T-ABS)}
$$

$$
tc(\Gamma, e, t) \leftarrow e = app(e_1, e_2) \land tc(\Gamma, e_1, arrow(u, t)) \land tc(\Gamma, e_2, u) \qquad \text{(T-APP)}
$$

STLC Type Inhabitation as a CHC ADT Problem

STLC Type Inhabitation Example

$$
\begin{aligned}
tc(\Gamma, e, t) &\leftarrow \Gamma = cons(v, t, \Gamma') \land e = var(v) \\
tc(\Gamma, e, t) &\leftarrow \Gamma = cons(v', t', \Gamma') \land tc(\Gamma', e, t) && \text{(T-Var)} \\
tc(\Gamma, e, t) &\leftarrow e = abs(v, e') \land t = arrow(t', u) \land tc(cons(v, t', \Gamma), e', u) && \text{(T-Abs)} \\
tc(\Gamma, e, t) &\leftarrow e = app(e_1, e_2) \land tc(\Gamma, e_1, arrow(u, t)) \land tc(\Gamma, e_2, u) && \text{(T-App)} \\
\bot &\leftarrow \forall a \,.\, tc(empty, e, arrow(arrow(a, a), a))
\end{aligned}
$$

STLC Type Inhabitation as a CHC ADT Problem

STLC Type Inhabitation Example

$$
\begin{array}{rll}
tc(\Gamma, e, t) & \leftarrow \Gamma = cons(v, t, \Gamma') \land e = var(v) & \\
tc(\Gamma, e, t) & \leftarrow \Gamma = cons(v', t', \Gamma') \land tc(\Gamma', e, t) & \text{(T-VAR)} \\
tc(\Gamma, e, t) & \leftarrow e = abs(v, e') \land t = arrow(t', u) \land tc(cons(v, t', \Gamma), e', u) & \text{(T-ABS)} \\
tc(\Gamma, e, t) & \leftarrow e = app(e_1, e_2) \land tc(\Gamma, e_1, arrow(u, t)) \land tc(\Gamma, e_2, u) & \text{(T-APP)} \\
\bot & \leftarrow \forall a \;.\; tc(empty, e, arrow(arrow(a, a), a)) &
\end{array}
$$

- An invariant $\text{lfp}(tc) = \{\langle \Gamma, e, t \rangle \mid \Gamma \vdash e : t\}$ is undefinable in FOL:

STLC Type Inhabitation as a CHC ADT Problem

STLC Type Inhabitation Example

$$\begin{aligned}
tc(\Gamma, e, t) &\leftarrow \Gamma = cons(v, t, \Gamma') \land e = var(v) \\
tc(\Gamma, e, t) &\leftarrow \Gamma = cons(v', t', \Gamma') \land tc(\Gamma', e, t) && \text{(T-Var)} \\
tc(\Gamma, e, t) &\leftarrow e = abs(v, e') \land t = arrow(t', u) \land tc(cons(v, t', \Gamma), e', u) && \text{(T-Abs)} \\
tc(\Gamma, e, t) &\leftarrow e = app(e_1, e_2) \land tc(\Gamma, e_1, arrow(u, t)) \land tc(\Gamma, e_2, u) && \text{(T-App)} \\
\bot &\leftarrow \forall a \,.\, tc(empty, e, arrow(arrow(a, a), a))
\end{aligned}$$

- An invariant $\mathrm{lfp}(tc) = \{\langle \Gamma, e, t \rangle \mid \Gamma \vdash e : t\}$ is undefinable in FOL:
  - ADT admits quantifier elimination
  - $\Rightarrow$ **finite** number of constructor and selector accesses

STLC Type Inhabitation as a CHC ADT Problem

STLC Type Inhabitation Example

$$
\begin{aligned}
tc(\Gamma, e, t) &\leftarrow \Gamma = cons(v, t, \Gamma') \land e = var(v) \\
tc(\Gamma, e, t) &\leftarrow \Gamma = cons(v', t', \Gamma') \land tc(\Gamma', e, t) && \text{(T-VAR)} \\
tc(\Gamma, e, t) &\leftarrow e = abs(v, e') \land t = arrow(t', u) \land tc(cons(v, t', \Gamma), e', u) && \text{(T-ABS)} \\
tc(\Gamma, e, t) &\leftarrow e = app(e_1, e_2) \land tc(\Gamma, e_1, arrow(u, t)) \land tc(\Gamma, e_2, u) && \text{(T-APP)} \\
\bot &\leftarrow \forall a \, . \, tc(empty, e, arrow(arrow(a, a), a))
\end{aligned}
$$

- An invariant $\mathrm{lfp}(tc) = \{\langle \Gamma, e, t \rangle \mid \Gamma \vdash e : t\}$ is undefinable in FOL:
  - ADT admits quantifier elimination
  - $\Rightarrow$ **finite** number of constructor and selector accesses
- There are **no invariants definable** in the assertion language!

STLC Type Inhabitation as a CHC ADT Problem

STLC Type Inhabitation Example

$tc(\Gamma, e, t) \leftarrow \Gamma = cons(v, t, \Gamma') \land e = var(v)$ (T-VAR)

$tc(\Gamma, e, t) \leftarrow \Gamma = cons(v', t', \Gamma') \land tc(\Gamma', e, t)$

$tc(\Gamma, e, t) \leftarrow e = abs(v, e') \land t = arrow(t', u) \ldots , \Gamma), e', u)$ (T-ABS)

$tc(\Gamma, e, t) \leftarrow e = app(e_1, e_2) \land tc(\Gamma, e \ldots tc(\Gamma, e_2, u)$ (T-APP)

$\bot \leftarrow \forall a \,.\, tc(empty, e, a \ldots , a))$

This undefinability makes most solvers[2] diverge!

- An in… $\{\langle \Gamma, e, t\rangle \mid \Gamma \vdash e : t\}$ is undefinable in FOL:
  - …s quantifier elimination
  - …ite number of constructor and selector accesses
- …here are **no invariants definable** in the assertion language!

[2] We tried Z3, Eldarica, HoIce

STLC Type Inhabitation: Regular Invariants

Classical Interpretation

$$\mathcal{I} \equiv \{\langle \Gamma, e, t \rangle \mid \text{for all } M, \text{ if } \forall u.u \in \Gamma \Rightarrow M \models u, \text{ then } M \models t\}$$

STLC Type Inhabitation: Regular Invariants

Classical Interpretation

$$\mathcal{I} \equiv \{\langle \Gamma, e, t \rangle \mid \text{for all } M, \text{ if } \forall u.u \in \Gamma \Rightarrow M \models u, \text{ then } M \models t\}$$

$$\left.\begin{array}{ll} arrow(1, 0) & \mapsto 0 \\ arrow(*, *) & \mapsto 1 \end{array}\right\} M \models t$$

STLC Type Inhabitation: Regular Invariants

Classical Interpretation

$$\mathcal{I} \equiv \{\langle \Gamma, e, t \rangle \mid \text{for all } M, \text{ if } \forall u. u \in \Gamma \Rightarrow M \models u, \text{ then } M \models t\}$$

$$\left.\begin{array}{ll} arrow(1,0) & \mapsto 0 \\ arrow(*,*) & \mapsto 1 \end{array}\right\} M \models t$$

$$\left.\begin{array}{ll} empty & \mapsto \notin \\ cons(v,1,\notin) & \mapsto \notin \\ cons(v,*,*) & \mapsto \in \end{array}\right\} \forall u. u \in \Gamma \Rightarrow M \models u$$

STLC Type Inhabitation: Regular Invariants

Classical Interpretation

$$\mathcal{I} \equiv \{\langle \Gamma, e, t\rangle \mid \text{for all } M, \text{ if } \forall u.u \in \Gamma \Rightarrow M \models u, \text{ then } M \models t\}$$

for all i: $Var_i \mapsto v$

for all i: $PrimType_i \mapsto 0$

$var(v) \mapsto e$

$abs(v, e) \mapsto e$

$app(e, e) \mapsto e$

$$\left.\begin{array}{ll} arrow(1,0) & \mapsto 0 \\ arrow(*,*) & \mapsto 1 \end{array}\right\} M \models t$$

$$\left.\begin{array}{ll} empty & \mapsto \notin \\ cons(v,1,\notin) & \mapsto \notin \\ cons(v,*,*) & \mapsto \in \end{array}\right\} \forall u.u \in \Gamma \Rightarrow M \models u$$

Constructed Tree Automaton

- $A = (\{0, 1, \in, \notin, v, e\}, \Sigma_F, \{\langle \in, 0\rangle, \langle \notin, 1\rangle, \langle \in, 1\rangle\}, \Delta)$

STLC Type Inhabitation: Regular Invariants

Classical Interpretation

$$\mathcal{I} \equiv \{\langle \Gamma, e, t \rangle \mid \text{for all } M, \text{ if } \forall u.u \in \Gamma \Rightarrow M \models u, \text{ then } M \models t\}$$

for all i: $Var_i \mapsto v$

for all i: $PrimType_i \mapsto 0$

$var(v) \mapsto e$

$abs(v, e) \mapsto e$

$app(e, e) \mapsto e$

$$\left.\begin{array}{ll} arrow(1,0) & \mapsto 0 \\ arrow(*,*) & \mapsto 1 \end{array}\right\} M \models t$$

$$\left.\begin{array}{ll} empty & \mapsto \notin \\ cons(v,1,\notin) & \mapsto \notin \\ cons(v,*,*) & \mapsto \in \end{array}\right\} \forall u.u \in \Gamma \Rightarrow M \models u$$

Constructed Tree Automaton

- $A = (\{0, 1, \in, \notin, v, e\}, \Sigma_F, \{\langle \in, 0 \rangle, \langle \notin, 1 \rangle, \langle \in, 1 \rangle\}, \Delta)$
- $tc(empty, e, arrow(arrow(0,0),0))$

STLC Type Inhabitation: Regular Invariants

Classical Interpretation

$$\mathcal{I} \equiv \{\langle \Gamma, e, t \rangle \mid \text{for all } M, \text{ if } \forall u. u \in \Gamma \Rightarrow M \models u, \text{ then } M \models t\}$$

$$
\begin{array}{ll}
\text{for all } i\text{: } Var_i & \mapsto v \\
\text{for all } i\text{: } PrimType_i & \mapsto 0 \\
var(v) & \mapsto e \\
abs(v, e) & \mapsto e \\
app(e, e) & \mapsto e
\end{array}
$$

$$
\left.\begin{array}{ll}
arrow(1, 0) & \mapsto 0 \\
arrow(*, *) & \mapsto 1
\end{array}\right\} M \models t
$$

$$
\left.\begin{array}{ll}
empty & \mapsto \notin \\
cons(v, 1, \notin) & \mapsto \notin \\
cons(v, *, *) & \mapsto \in
\end{array}\right\} \forall u. u \in \Gamma \Rightarrow M \models u
$$

Constructed Tree Automaton

- $A = (\{0, 1, \in, \notin, v, e\}, \Sigma_F, \{\langle \in, 0 \rangle, \langle \notin, 1 \rangle, \langle \in, 1 \rangle\}, \Delta)$
- $tc(empty, e, arrow(arrow(0, 0), 0)) \rightarrow_A tc(\notin, e, arrow(1, 0))$

STLC Type Inhabitation: Regular Invariants

Classical Interpretation

$$\mathcal{I} \equiv \{\langle \Gamma, e, t\rangle \mid \text{for all } M, \text{ if } \forall u.u \in \Gamma \Rightarrow M \models u, \text{ then } M \models t\}$$

for all i: $Var_i \mapsto v$

for all i: $PrimType_i \mapsto 0$

$var(v) \mapsto e$

$abs(v, e) \mapsto e$

$app(e, e) \mapsto e$

$$\left.\begin{array}{ll} arrow(1,0) & \mapsto 0 \\ arrow(*,*) & \mapsto 1 \end{array}\right\} M \models t$$

$$\left.\begin{array}{ll} empty & \mapsto \notin \\ cons(v, 1, \notin) & \mapsto \notin \\ cons(v, *, *) & \mapsto \in \end{array}\right\} \forall u.u \in \Gamma \Rightarrow M \models u$$

Constructed Tree Automaton

- $A = (\{0, 1, \in, \notin, v, e\}, \Sigma_F, \{\langle \in, 0\rangle, \langle \notin, 1\rangle, \langle \in, 1\rangle\}, \Delta)$
- $tc(empty, e, arrow(arrow(0, 0), 0)) \rightarrow_A tc(\notin, e, arrow(1, 0)) \rightarrow_A tc(\notin, e, 0)$

Regular Invariants

Tree Automata

A *deterministic finite tree n-automaton* over Σ_F is a quadruple $(S, \Sigma_F, S_F, \Delta)$, where

- S is a finite set of states,
- $S_F \subseteq S^n$ is a set of final states,
- Δ is a transition relation with rules of the form: $f(s_1, \ldots, s_m) \to s$, where $f \in \Sigma_F$, $ar(f) = m$ and $s, s_1, \ldots, s_m \in S$, and there are no two rules in Δ with the same left-hand side.

Regular Invariants

Tree Automata

A *deterministic finite tree n-automaton* over Σ_F is a quadruple $(S, \Sigma_F, S_F, \Delta)$, where

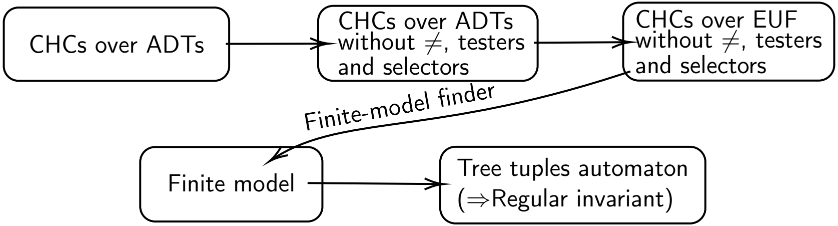
- S is a finite set of states,
- $S_F \subseteq S^n$ is a set of final states,
- Δ is a transition relation with rules of the form: $f(s_1, \ldots, s_m) \to s$, where $f \in \Sigma_F$, $ar(f) = m$ and $s, s_1, \ldots, s_m \in S$, and there are no two rules in Δ with the same left-hand side.

Regular Relations

A relation $X \subseteq |\mathcal{H}|_{\sigma_1} \times \ldots \times |\mathcal{H}|_{\sigma_n}$ is called *regular* iff there is an n-automaton A over Σ_F, s.t.:

$$X = \{\langle a_1, \ldots, a_n \rangle \mid \langle a_1, \ldots, a_n \rangle \text{ is accepted by } A, a_i \in |\mathcal{H}|_{\sigma_i}\}.$$

Regular Invariant Inference by Finite-Model Finding



Regular Invariant Inference by Finite-Model Finding

CHC system

$$x = Z \rightarrow even(x)$$
$$x = S(S(y)) \land even(y) \rightarrow even(x)$$
$$even(x) \land even(y) \land y = S(x) \rightarrow \bot$$

Regular Invariant Inference by Finite-Model Finding

CHC system as a EUF-formula

$$\forall x.(x = Z \rightarrow even(x)) \land$$
$$\forall x, y.(x = S(S(y)) \land even(y) \rightarrow even(x)) \land$$
$$\forall x, y.(even(x) \land even(y) \land y = S(x) \rightarrow \bot)$$

Regular Invariant Inference by Finite-Model Finding

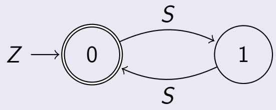
CHC system as a EUF-formula

$$\forall x.(x = Z \rightarrow even(x)) \land$$
$$\forall x, y.(x = S(S(y)) \land even(y) \rightarrow even(x)) \land$$
$$\forall x, y.(even(x) \land even(y) \land y = S(x) \rightarrow \bot)$$

Finite Model

$$|\mathcal{M}|_{Nat} = \{0, 1\}$$
$$\mathcal{M}(Z) = 0$$
$$\mathcal{M}(S)(x) = 1 - x$$
$$\mathcal{M}(even) = \{0\}$$

Model Representation



Finite Models to Regular Invariants

Tree Automata Construction

$$\mathcal{A}_P = \left(\left|\mathcal{M}\right|, \Sigma_F, \mathcal{M}(P), \tau\right), \text{ where for all } x_i \in |\mathcal{M}|_{\sigma_i},$$
$$\tau\left(f(x_1, \ldots, x_n)\right) = \mathcal{M}(f)(x_1, \ldots, x_n)$$

Finite Model

$$|\mathcal{M}|_{Nat} = \{0, 1\}$$
$$\mathcal{M}(Z) = 0$$
$$\mathcal{M}(S)(x) = 1 - x$$
$$\mathcal{M}(even) = \{0\}$$

Finite Models to Regular Invariants

Tree Automata Construction

$\mathcal{A}_P = \left(|\mathcal{M}|, \Sigma_F, \mathcal{M}(P), \tau\right)$, where for all $x_i \in |\mathcal{M}|_{\sigma_i}$,

$$\tau\big(f(x_1, \ldots, x_n)\big) = \mathcal{M}(f)(x_1, \ldots, x_n)$$

Finite Model

$$|\mathcal{M}|_{Nat} = \{0, 1\}$$
$$\mathcal{M}(Z) = 0$$
$$\mathcal{M}(S)(x) = 1 - x$$
$$\mathcal{M}(even) = \{0\}$$

Finite Models to Regular Invariants

Tree Automata Construction

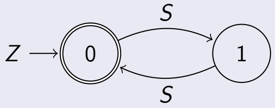
$$\mathcal{A}_P = \left(|\mathcal{M}|, \Sigma_F, \mathcal{M}(P), \tau\right), \text{ where for all } x_i \in |\mathcal{M}|_{\sigma_i},$$
$$\tau\left(f(x_1, \dots, x_n)\right) = \mathcal{M}(f)(x_1, \dots, x_n)$$

Finite Model

$$|\mathcal{M}|_{Nat} = \{0, 1\}$$
$$\mathcal{M}(Z) = 0$$
$$\mathcal{M}(S)(x) = 1 - x$$
$$\mathcal{M}(even) = \{0\}$$

Tree Automaton

$$\tau(Z) = \mathcal{M}(Z) = 0$$
$$\tau(S(0)) = \mathcal{M}(S)(0) = 1$$
$$\tau(S(1)) = \mathcal{M}(S)(1) = 0$$



Selector and Tester Support

Selectors and testers can be supported by introducing new clauses, e.g.:

$$cons?(x) \leftarrow x = cons(y, ys)$$

$$car(x, r) \leftarrow x = cons(r, xs)$$

Disequality Elimination

Problem

$\forall x, y.(\bot \leftarrow \neg(x = y))$ is always satisfied by a model of size 1

Disequality Elimination

Problem

$\forall x, y.(\bot \leftarrow \neg(x = y))$ is always satisfied by a model of size 1

Solution

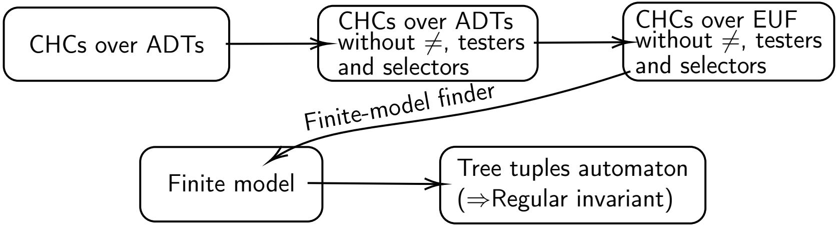
Substitute disequalities with new atoms, e.g.:

$$diseq_{Nat}(Z, S(x)) \leftarrow \top$$

$$diseq_{Nat}(S(x), Z) \leftarrow \top$$

$$diseq_{Nat}(S(x), S(y)) \leftarrow diseq_{Nat}(x, y)$$

Regular Invariant Inference by Finite-Model Finding



Implementation

- All the presented transformations are proved to be *sound*

Implementation

- All the presented transformations are proved to be *sound*
- The approach was implemented in F# in a tool named RInGen

Implementation

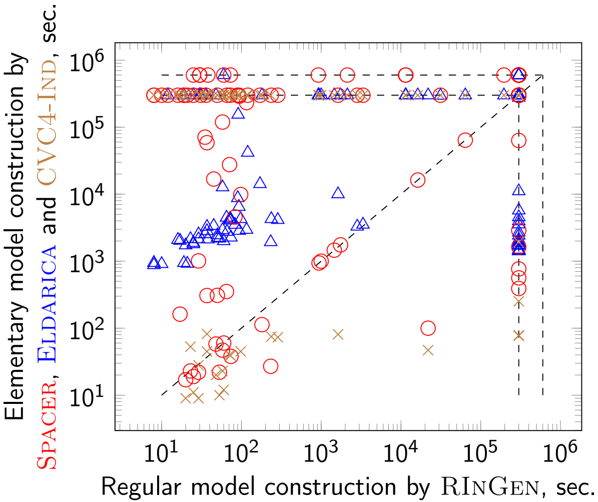
- All the presented transformations are proved to be *sound*
- The approach was implemented in F# in a tool named RInGen
- RInGen uses CVC4 as a backend finite-model finder

Evaluation

Benchmark	#	Result	RInGen	Eldarica	Z3	CVC4-Ind
PositiveEq[1,2]	35	SAT	**27**	1	4	0
Diseq[1,2]	25	SAT	**4**	0	2	0
		UNSAT	**1**	**1**	**1**	**1**
TIP[3]	377	SAT	18	**24**	0	0
		UNSAT	36	**40**	31	22
Total	437	SAT	**49**	25	6	0
		UNSAT	37	**41**	32	23

1. Yang et al. "Lemma Synthesis for Automating Induction over Algebraic Data Types". In: *CP*. 2019
2. De Angelis et al. "Solving Horn Clauses on Inductive Data Types Without Induction". In: *TPLP* (2018)
3. Claessen et al. "TIP: tons of inductive problems". In: *CICM*. 2015

Evaluation: Efficiency



Summary

- The paper is (conditionally) accepted at PLDI'21
- RInGen participated in CHC-COMP'21 on ADT tracks
- RInGen code: `https://github.com/Columpio/RInGen`
- Benchmarks from the paper: `https://gitlab.com/Columpio/adt-benchmarks`
- Also on CHC-COMP: `https://github.com/chc-comp/ringen-adt-benchmarks`
- Future / Ongoing Work:
  - Proper support for CHCs with quantifier alternation
  - Combining FOL-based invariants and regular invariants