Refinement Types and Higher-Order Constrained Horn Clauses

Some work in progress with Toby Cathcart Burn, Martin Lester and Luke Ong.

Motivation:

Constrained Horn clauses in program verification.

Safety verification as constrained horn clause solving.

```
let rec sum n =
  if n <= 0 then 0 else n + sum (n-1)
in fun n -> assert (n <= sum n)
```

$$\forall n.\ n \leq 0 \ \Rightarrow\ \boldsymbol{P}\ n\ 0$$

$$\forall nm.\ n > 0 \ \wedge\ \boldsymbol{P}\ (n-1)\ m \ \Rightarrow\ \boldsymbol{P}\ n\ (n+m)$$

$$\forall nm.\ \boldsymbol{P}\ n\ m \ \Rightarrow\ n \leq m$$

$$\forall n.\ n \leq 0 \ \Rightarrow\ \boldsymbol{P}\ n\ 0$$

$$\forall nm.\ n > 0 \ \wedge\ \boldsymbol{P}\ (n-1)\ m \ \Rightarrow\ \boldsymbol{P}\ n\ (n+m)$$

$$\forall nm.\ \boldsymbol{P}\ n\ m \ \Rightarrow\ n \leq m$$

Fix a constraint logic (quantifier free integer linear arithmetic).

Constrained horn clause: At most one uninterpreted positive literal.

One possible model is the assignment $\boldsymbol{P} \mapsto \{\, (x, y) \in \mathbb{Z} \times \mathbb{Z} \mid x \leq y \,\}$

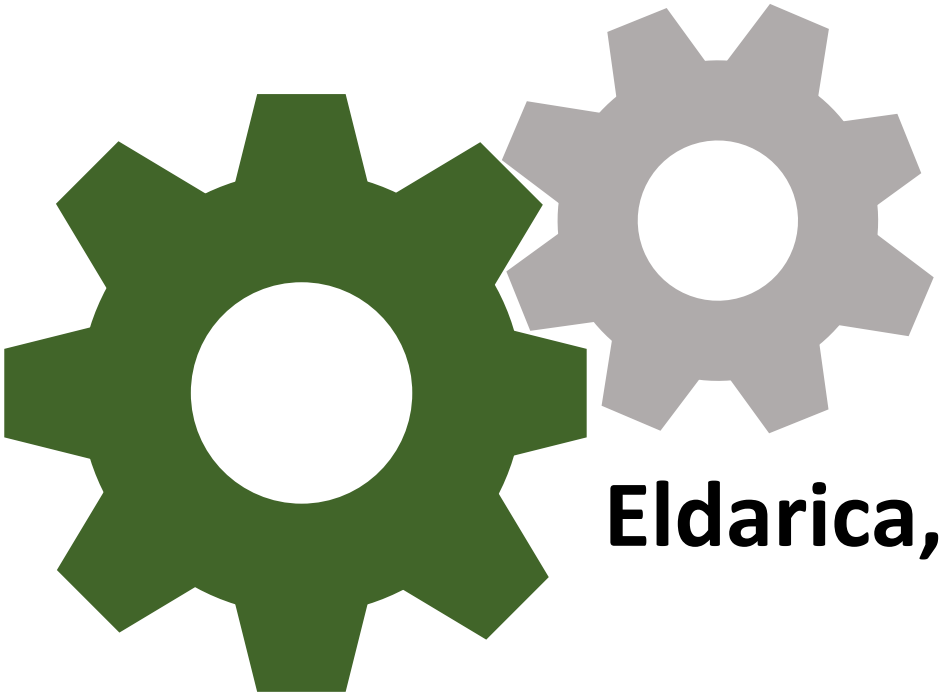
$$P \mapsto \lambda nm.\, n \leq m$$

$$\forall n.\, n \leq 0 \;\Rightarrow\; \boldsymbol{P}\; n\; 0$$

$$\forall nm.\, n > 0 \;\wedge\; \boldsymbol{P}\; (n-1)\; m \;\Rightarrow\; \boldsymbol{P}\; n\; (n+m)$$

$$\forall nm.\, \boldsymbol{P}\; n\; m \;\Rightarrow\; n \leq m$$

Symbolic models are expressible within the constraint language.



Eldarica, SeaHorn, HSF, μZ …

```
let add x y = x + y
let rec iter f m n =
  if n <= 0 then m else f n (iter f m (n-1))
in fun n -> assert (n <= iter add 0 n)
```

$$\forall xyz.\ z = x + y \Rightarrow \boldsymbol{Add}\ x\ y\ z$$

$$\forall fmn.\ n \leq 0 \Rightarrow \boldsymbol{Iter}\ f\ m\ n\ m$$

$$\forall fmnpr.\ n > 0 \wedge \boldsymbol{Iter}\ f\ m\ (n-1)\ p \wedge f\ n\ p\ r \Rightarrow \boldsymbol{Iter}\ f\ m\ n\ r$$

$$\forall nr.\ \boldsymbol{Iter}\ \boldsymbol{Add}\ 0\ n\ r \Rightarrow n \leq r$$

Higher-order constrained horn clauses:

Constrained horn clauses in higher-order logic.

Higher-order "unknown" relations:

$Iter : (\text{int} \to \text{int} \to \text{int} \to \text{bool}) \to \text{int} \to \text{int} \to \text{int} \to \text{bool}$

(We assume everything

$$\forall xyz.\ z = x + y \Rightarrow \boldsymbol{Add}\ x\ y\ z$$

$$\forall fmn.\ n \leq 0 \Rightarrow \boldsymbol{Iter}\ f\ m\ n\ m$$

$$\forall fmnpr.\ n > 0 \wedge \boldsymbol{Iter}\ f\ m\ (n-1)\ p \wedge f\ n\ p\ r \Rightarrow \boldsymbol{Iter}\ f\ m\ n\ r$$

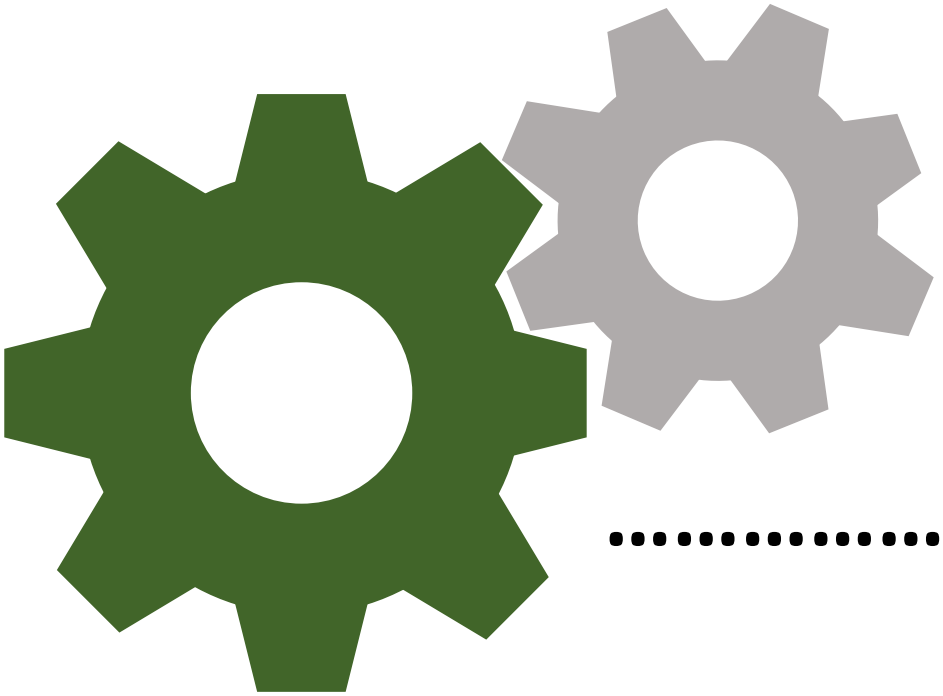
$$\forall nr.\ \boldsymbol{Iter}\ \boldsymbol{Add}\ 0\ n\ r \Rightarrow n \leq r$$

Quantification at higher-sorts:

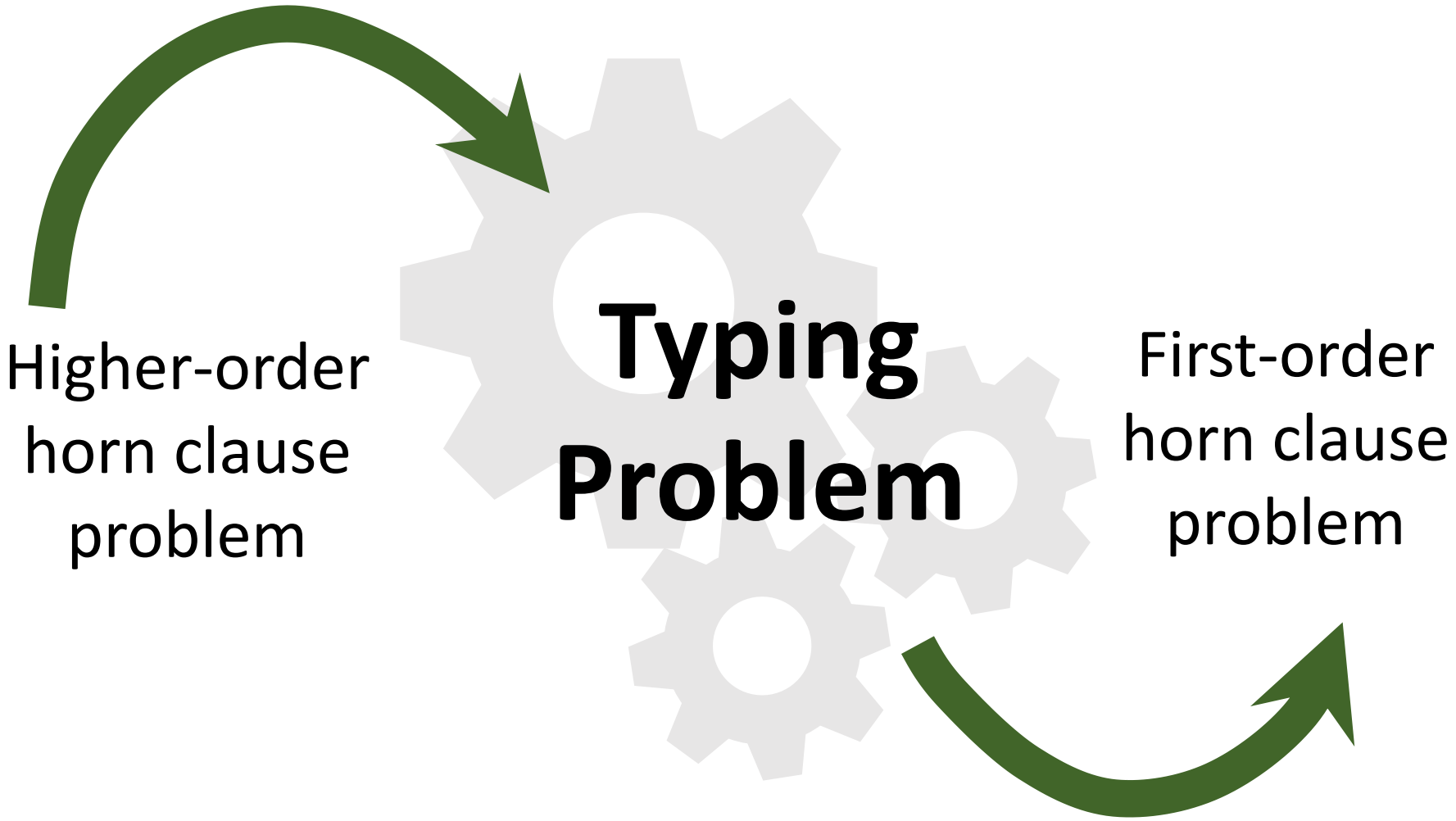
$\forall : ((\text{int} \to \text{int} \to \text{int} \to \text{bool}) \to \text{bool}) \to \text{bool}$

Literals headed by (bound) variables:

$\forall x.\ x\ (y\ z) : \text{bool}$



.....?


Higher-order horn clause problem
Typing Problem
First-order horn clause problem

Symbolic models at higher orders:

Refinement types as higher-order invariants.

$$Add \mapsto \lambda xyz.\, z = x + y$$

$$Iter \mapsto \lambda fmnr.\, (\forall xyz.\, f\ x\ y\ z \Rightarrow 0 < x \Rightarrow z > y) \Rightarrow m \geq 0 \Rightarrow n \leq r$$

$$\forall xyz.\ z = x + y \Rightarrow \boldsymbol{Add}\ x\ y\ z$$

$$\forall fmn.\ n \leq 0 \Rightarrow \boldsymbol{Iter}\ f\ m\ n\ m$$

$$\forall fmnpr.\ n > 0 \wedge \boldsymbol{Iter}\ f\ m\ (n-1)\ p \wedge f\ n\ p\ r \Rightarrow \boldsymbol{Iter}\ f\ m\ n\ r$$

$$\forall nr.\, \boldsymbol{Iter}\ \boldsymbol{Add}\ 0\ n\ r \Rightarrow n \leq r$$

Higher-type symbolic models:
expressible in the constraint language?

$$R : \big((\text{int} \to \text{bool}) \to \text{bool}\big) \to \text{bool}$$

Symbolic model:

$$R \mapsto \lambda f.\,(\forall g.\ f\ g \Rightarrow (\forall x.\, g\ x \Rightarrow \phi) \Rightarrow \psi) \Rightarrow \chi$$

Dependent refinement type:

$$R : \ f\colon (g\colon (x\colon \text{int} \to \text{bool}\langle\phi\rangle) \to \text{bool}\langle\psi\rangle) \to \text{bool}\langle\chi\rangle$$

Dependent refinement types:

$$T \quad ::= \quad \text{int} \;\mid\; \text{bool}\langle\phi\rangle \;\mid\; x{:}\,T_1 \to T_2$$

Refinement at bool:
ϕ is a FO formula of
the constraint language

Dependence at int:
x can occur freely in
T_2 only if $T_1 = \text{int}$

$$[\![\text{int}]\!](\alpha) = \mathbb{Z}$$

$$[\![\text{bool}\langle\phi\rangle]\!](\alpha) = \{\text{false}, [\![\phi]\!](\alpha)\}$$

$$[\![x{:}\,T_1 \to T_2]\!](\alpha) = \Pi d \in [\![T_1]\!](\alpha).\, [\![T_2]\!](\alpha[x \mapsto d])$$

Example refinement:

$[\![\text{bool}\langle x \leq y \rangle]\!](x \mapsto 1, y \mapsto 2)$

$= \{\text{false, true}\}$

Consequence of defn:

$b \in [\![\text{bool}\langle \phi \rangle]\!](\alpha)$

iff $\alpha \vDash b \Rightarrow \phi$

Refinements of relational sorts:

$[\![x\colon \text{int} \to \text{bool}\langle \phi \rangle]\!]$

$= \Pi n \in \mathbb{Z}.\, [\![\text{bool}\langle \phi[n/x] \rangle]\!]$

$= \{\, f \mid \forall n \in \mathbb{Z}.\, f\ n \in [\![\text{bool}\langle \phi[n/x] \rangle]\!] \,\}$

$= \{\, f \mid \forall n \in \mathbb{Z}.\, f\ n \Rightarrow \phi[n/x] \}$

$= \{\, f \mid \forall x.\, f\ x \Rightarrow \phi \,\}$

Type schemes:

$$S \quad ::= \quad T \quad | \quad \forall X.S$$

Rank-1, second-order universal quantification:

X is a second order relational variable

(i.e. its sort has shape int $\rightarrow \cdots \rightarrow$ int $\rightarrow$ bool)

Type environment:

Finite map from variables to monotypes (not schemes).

$$R_1 : T_1$$
$$R_2 : T_2$$
$$\vdots$$
$$R_m : T_m$$

Symbolic Model

Type system:

Working with given symbolic models.

Decidable judgement

$$\Gamma \vdash \phi : T$$

Refinement type:
Monotype
(i.e. not a scheme)

Type env:
Finite map from variables to monotypes

Term:
A subterm of the body of a horn clause

$\Gamma \vdash \phi : \mathrm{bool}\langle\psi\rangle$

In **symbolic model** Γ, **higher-order** formula ϕ implies **first-order** constraint formula ψ

$\mathsf{n} : \mathsf{int}$

$+ : \mathsf{int} \to \mathsf{int} \to \mathsf{int}$

$\leq : x\colon \mathsf{int} \to y\colon \mathsf{int} \to \mathsf{bool}\langle x \leq y \rangle$

$\wedge : \forall XY.\, \mathsf{bool}\langle X \rangle \to \mathsf{bool}\langle Y \rangle \to \mathsf{bool}\langle X \wedge Y \rangle$

$\exists : \forall X.\, (x\colon \mathsf{int} \to \mathsf{bool}\langle X\ x \rangle) \to \mathsf{bool}\langle \exists x.\, X\ x \rangle$

Abbreviation: $\exists x.\, \phi \equiv \exists\, (\lambda x.\, \phi)$

$$\frac{}{\Gamma_1, x{:}\,T, \Gamma_2 \vdash x{:}\,T} \quad \text{(Var)}$$

$$\frac{c : \forall X_1 \ldots X_k.\, T}{\Gamma \vdash c : T[\phi_1 \ldots \phi_k / X_1 \ldots X_k]} \quad \text{(Const)}$$

$$\frac{\Gamma \vdash s : (x{:}\,T_1 \to T_2) \qquad \Gamma \vdash t : T_1}{\Gamma \vdash s\ t : T_2[t/x]} \quad \text{(App)}$$

$$\frac{\Gamma, x{:}\,T_1 \vdash s{:}\,T_2}{\Gamma \vdash \lambda x.\, s : (x{:}\,T_1 \to T_2)} \quad \text{(Abs)}$$

$$\frac{\Gamma \vdash s : T_1 \quad T_1 \sqsubseteq T_2}{\Gamma \vdash s : T_2}\ \text{(Sub)}$$

$$\frac{\mathbb{Z}LA \models \phi \Rightarrow \psi}{\text{bool}\langle\phi\rangle \sqsubseteq \text{bool}\langle\psi\rangle}\ \text{(SubB)}$$

$$\frac{T_1' \sqsubseteq T_1 \qquad T_2 \sqsubseteq T_2'}{x{:}\,T_1 \to T_2 \sqsubseteq x{:}\,T_1' \to T_2'}\ \text{(SubArr)}$$

For example…

r : int

n : int

Add : $(x$: int $\to \cdots \to$ bool$\langle z = x + y \rangle)$

$Iter$: $(x$: int $\to \cdots \to$ bool$\langle 0 < x \Rightarrow y < z \rangle) \to \cdots \to$ bool$\langle 0 \leq m \Rightarrow n \leq r \rangle$

Γ

$$\dfrac{\dfrac{}{\Gamma \vdash > : (x\colon \mathsf{int} \to y\colon \mathsf{int} \to \mathsf{bool}\langle x > y\rangle)} \quad \dfrac{}{\Gamma \vdash n : \mathsf{int}}}{\Gamma \vdash > n : (y\colon \mathsf{int} \to \mathsf{bool}\langle n > y\rangle)} \quad \dfrac{}{\Gamma \vdash r : \mathsf{int}}$$

$$\Gamma \vdash n > r : \mathsf{bool}\langle n > r\rangle$$

$$\mathrm{T}_1 = (x\colon \mathsf{int} \to \cdots \to \mathsf{bool}\langle 0 < x \Rightarrow y < z\rangle) \to m\colon \mathsf{int} \to \cdots \to \mathsf{bool}\langle 0 \leq m \Rightarrow n \leq r\rangle$$

$$\dfrac{\dfrac{}{\Gamma \vdash Iter : T_1} \quad \dfrac{\dfrac{}{\Gamma \vdash Add\ : (x\colon \mathsf{int} \to \cdots \to \mathsf{bool}\langle z = x + y\rangle)}}{\Gamma \vdash Add\ : (x\colon \mathsf{int} \to \cdots \to \mathsf{bool}\langle 0 < x \Rightarrow y < z\rangle)}}{\dfrac{\Gamma \vdash Iter\ Add\ : x\colon int \to \cdots \to \mathsf{bool}\langle 0 \leq x \Rightarrow y \leq z\rangle}{\vdots}}$$

$$\dfrac{}{\Gamma \vdash Iter\ Add\ 0\ n\ r : \mathsf{bool}\langle n \leq r\rangle}$$

$$\wedge : \forall XY.\,\mathsf{bool}\langle X\rangle \to \mathsf{bool}\langle Y\rangle \to \mathsf{bool}\langle X \wedge Y\rangle$$

$$\Gamma \vdash \wedge : \mathsf{bool}\langle n \leq r\rangle \to \mathsf{bool}\langle n > r\rangle \to \mathsf{bool}\langle n \leq r \wedge n > r\rangle$$

$$\Gamma \vdash (Iter\ Add\ 0\ n\ r) \wedge (n > r) : \mathsf{bool}\langle n \leq r \wedge n > r\rangle$$

$$\Gamma \vdash (Iter\ Add\ 0\ n\ r) \wedge (n > r) : \mathsf{bool}\langle \mathsf{false}\rangle$$

Given HO horn clauses H and type environment Γ :

$$B_1 \Rightarrow R_1 x_1 \cdots x_{k_1} \qquad R_1 : T_1$$

$$B_2 \Rightarrow R_2 x_1 \cdots x_{k_2} \qquad R_2 : T_2$$

$$\vdots \qquad\qquad \vdots$$

$$B_m \Rightarrow R_m x_1 \cdots x_{k_m} \qquad R_m : T_m$$

$$B \Rightarrow \phi$$

If, for all *i*, $\Gamma \vdash \lambda x_1 \ldots x_{k_i}. B_i : T_i$ **and** $\Gamma \vdash B \wedge \neg\phi : \text{bool}\langle\text{false}\rangle$
then Γ **represents a solution to the HO horn clauses** H

Type inference:

How to find symbolic models.

From higher-order relational variables:

$$R : (\text{int} \to \text{bool}) \to \text{bool}$$
$$S : \text{int} \to \text{int} \to \text{bool}$$

Create refinement templates:

$$R : (x\text{: int} \to \text{bool}\langle Z_1\ x\rangle) \to \text{bool}\langle Z_2\rangle$$
$$S : (y\text{: int} \to x\text{: int} \to \text{bool}\langle Z_3\ x\ y\rangle)$$

Check that this type environment is a model…

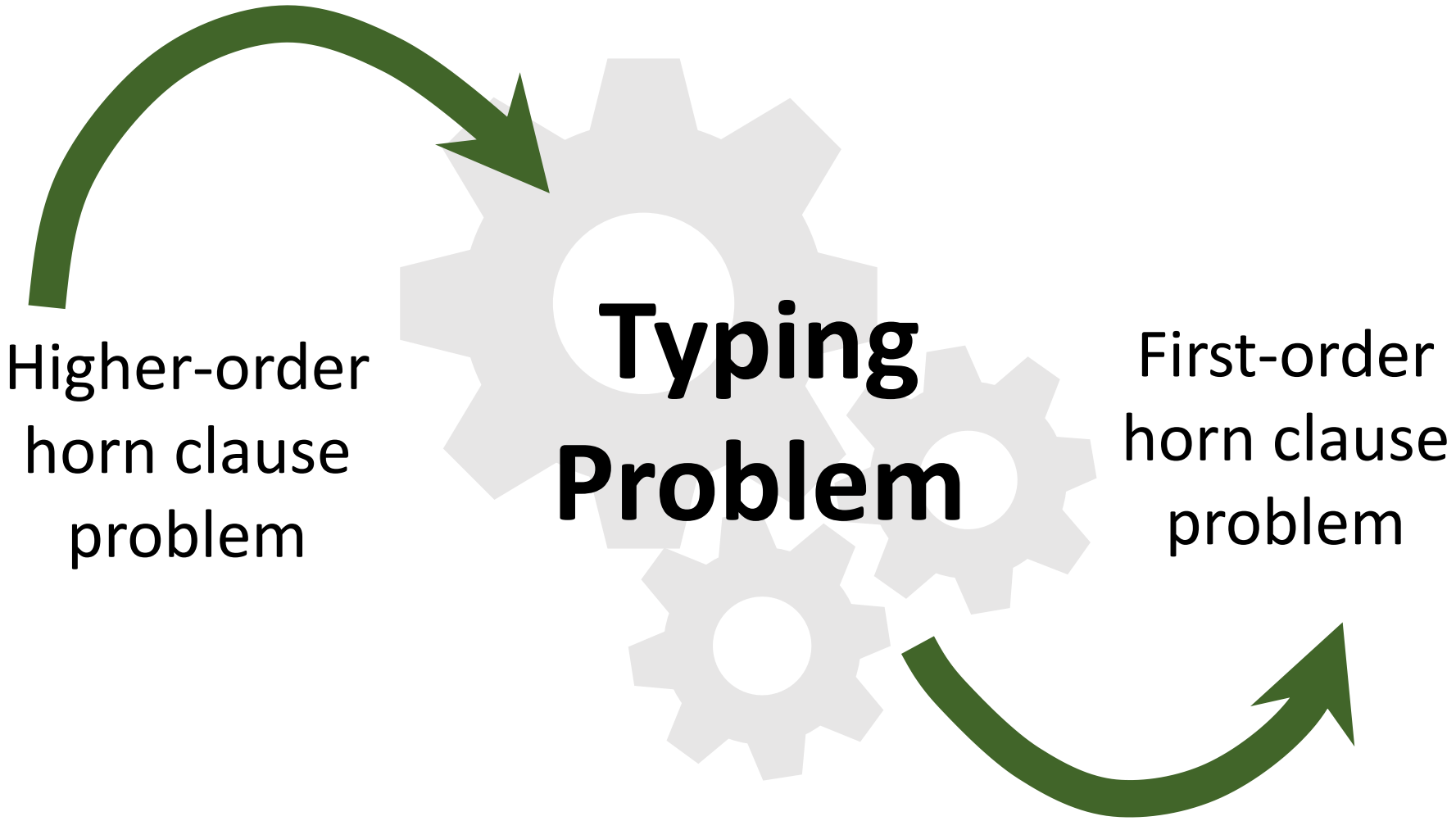
$$R : (x\text{: int} \to \text{bool}\langle Z_1\ x\rangle) \to \text{bool}\langle Z_2\rangle$$

$$S : (y\text{: int} \to x\text{: int} \to \text{bool}\langle Z_3\ x\ y\rangle)$$

…except, whenever you would be forced to check the validity of an implication:

$$\frac{\vDash Z_3 n\ z \Rightarrow Z_1 z}{bool\langle Z_3 n\ z\rangle \sqsubseteq bool\langle Z_1 z\rangle} \quad \text{(Sub-Bool)}$$

Instead record it as a (first-order Horn) constraint.


Higher-order horn clause problem
Typing Problem
First-order horn clause problem

In progress...

Completeness for the type system:

If HO ϕ implies FO constraint ψ then $\vdash \phi : \mathsf{bool}\langle\psi\rangle$

Lots more implementation work.

Richer sort systems to allow easier encoding of programs.