Information-Aware Type Systems

Philippa Cowderoy

August 30, 2018

What are Information-Aware Type Systems?

An Information-Aware Type System is a type system where:

- It is clear where information is introduced and eliminated
- It is clear (or at least clearer) how information flows within the type system

This is achieved by using *information effects* to track where information is created and destroyed - or if you prefer, where the system violates *conservation of information*. We hope inferences tell us something new!

Why Bother?

Our standard notation hides things from us.

$$\frac{\Gamma \vdash Tp : \tau p \qquad \Gamma \vdash Tf : \tau p \rightarrow \tau r}{\Gamma \vdash Tf\ Tp : \tau r} App1 \qquad \frac{\Gamma \vdash Tf : \tau f \qquad \Gamma \vdash Tp : \tau p \qquad \tau p \rightarrow \tau r = \tau f}{\Gamma \vdash Tf\ Tp : \tau r} App2$$

- While we are used to *App*1, *App*2 is easier for beginners to understand – an implicit constraint is made explicit.
- Generating that constraint is an information effect.
- Information-Awareness means more syntax, but makes possibilities clearer.

How To Make A System Information-Aware

This is just one recipe, but it's pretty reliable:

- Linear logic variables: one +ve occurrence, one -ve
- Constraints:
  - Constraint generation is an information effect
  - Constraints give us an abstraction tool
  - Constraints help avoid *overconstraining* data flow
- Duplication effects: track dataflow branches and merges
- Mode analysis: keep track of which way data flows, which forms of constraints we can solve

Constraints for the Simply-Typed Lambda Calculus

$\tau = \tau$	Type equality
$\tau \prec^{\tau l}_{\tau r}$	Type duplication
$x : \tau \in \Gamma$	Binding in context
$\Gamma' := \Gamma \,;\, x : \tau$	Context extension
$\Gamma \prec^{\Gamma L}_{\Gamma R}$	Context duplication

Note that the context constraints encode the structural rules. An alternative interpretation could give us a minimal linear calculus.

Playing with $\prec$ might lead the adventurous thinker down other paths entirely!...

Information-Aware Simply-Typed λ-Calculus (unannotated)

$$\frac{x : \tau \in \Gamma}{\Gamma \vdash x : \tau}\,Var \qquad \frac{\begin{array}{c}\Gamma f := \Gamma \,;\, x : \tau p \\ \Gamma f \vdash T : \tau r \\ \tau f = \tau p \to \tau r\end{array}}{\Gamma \vdash \lambda x.T \;:\; \tau f}\,Lam$$

$$\frac{\begin{array}{c}\Gamma \prec^{\Gamma L}_{\Gamma R} \\ \Gamma L \vdash Tf : \tau f \qquad \Gamma R \vdash Tp : \tau p \\ \tau p \to \tau r = \tau f\end{array}}{\Gamma \vdash Tf\ Tp : \tau r}\,App$$

Annotations, Duplication & Bidirectionality

Let's support annotations!

- We are forced to duplicate a type
- We could duplicate the function type to check then return
- Better: send the annotation both 'in' and 'out'

$$
\frac{\begin{array}{c} \tau a \prec \begin{smallmatrix}\tau ap \\ \tau af\end{smallmatrix} \\ \Gamma f := \Gamma \,;\, x : \tau ap \\ \Gamma f \vdash T : \tau r \\ \tau f = \tau af \rightarrow \tau r \end{array}}{\Gamma \vdash \lambda x : \tau a \,.\, T \;:\; \tau f} ALam
$$

Different Modes of a Type System

Mode	Unidirectional	Bidirectional
$\Gamma^+ \vdash T^+ : \tau^+$	Type Checking	Checking
$\Gamma^+ \vdash T^+ : \tau^-$		Synthesis
$\Gamma^- \vdash T^+ : \tau^+$	Free Variable Types	Checked type
$\Gamma^- \vdash T^+ : \tau^-$		Synthesised Type
$\Gamma^+ \vdash T^- : \tau^+$	Proof search	
	Program Synthesis	

- Systems that only support checking modes may not be *algorithms*, but they're typecheckers and not type systems.
- I'm not aiming to actively *support* program synthesis. Without syntax direction, it's search as usual.

$\rightarrow$ - The Other Information Effect

- The function arrow $\rightarrow$ doesn't appear in the source language, but it does appear in our types.
  - Not simply isomorphic to something in the term
  - Part of our (abstract) *interpretation* of a term

- Information we *generate* from or *create* about terms

- I assign two different modes to $\rightarrow$
  - based on how the solver handles $=$ constraints
  - Convention: LHS of $=$ is being 'assigned to' in some form

Modes for $\rightarrow$ - 1

- $\tau 1^{+} = \tau 2^{-} \rightarrow^{+} \tau 3^{-}$
  - $\rightarrow$ behaves as a *constructor* assigned to $\tau 1$
  - Variable parameters to $\rightarrow^{+}$ have -ve mode – they are being consumed to construct something to match against
- $\tau 1^{+} \rightarrow^{-} \tau 2^{-} = \tau 3^{-}$
  - $\rightarrow$ behaves as a *pattern* matched against $\tau 3$
  - Variable parameters with $+$ve mode act as variable patterns, producing something to use elsewhere
  - Variables are matched against when -ve, but generate no new local information

Modes for $\rightarrow$ - 2

During solving:

- $\rightarrow^+$ creates or introduces information
- $\rightarrow^-$ destroys or eliminates information

Why mention introduction and elimination? Well, $\rightarrow^+$ appears in the *Lam* rule, aka $\rightarrow I$. And $\rightarrow^-$ in *App*, aka $\rightarrow E$. The modes are telling us about introducing and eliminating connectives!

Contextual Behaviour

Context extension and binding constraints also have a relationship.

Read one way:

- $\Gamma' := \Gamma \,;\, x : \tau$ introduces the need for a binding
- $x : \tau \in \Gamma$ makes use of - or especially in linear and affine systems eliminates a binding

This can also be read in reverse:

- Using a variable requires it to be bound
- Providing a binding meets that requirement!

Likewise, $\Gamma {\prec}\langle^{\Gamma L}_{\Gamma R}$ can be read as merging ΓL and ΓR.

Information-Aware Simply-Typed λ-Calculus (moded)

Mode: $\Gamma^+ \vdash T^+ : \tau^-$ (Synthesis or 'typechecking')

$$\frac{x^- : \tau^+ \in \Gamma^-}{\Gamma^+ \vdash x^+ : \tau^-}\,Var \qquad \frac{\begin{array}{c}\Gamma f^+ := \Gamma^- \,;\, x^- : \tau p^+ \\ \Gamma f^- \vdash T^- : \tau r^+ \\ \tau f^+ = \tau p^- \to^+ \tau r^-\end{array}}{\Gamma^+ \vdash \lambda x^+ . T^+ \;:\; \tau f^-}\,Lam$$

$$\frac{\begin{array}{c}\Gamma^- \prec^{\Gamma f^+}_{\Gamma p^+} \\ \Gamma f^- \vdash Tf^- : \tau f^+ \qquad \Gamma p^- \vdash Tp^- : \tau p^+ \\ \tau p^- \to^- \tau r^+ = \tau f^-\end{array}}{\Gamma^+ \vdash Tf^+ \; Tp^+ : \tau r^-}\,App$$

Proofs and Symmetries Undone

Conservation of information requires a symmetry which our information effects can break.

If we restrict ourselves to a linear system then we can hopefully implement our context constraints with no violations – the symmetry is between introduction and elimination.

Typings are proofs – what's the proof theoretic angle on all this?