The Impact of Linearity Information on the Performance of TyCO

Francisco Martins
University of Azores
Luís Lopes
University of Porto
Vasco T. Vasconcelos
University of Lisbon

Goals

- **Study a linear channel inference system for the TyCO programming language.**
- **Apply linear channel information to optimize code generation for a multithreaded runtime system.**

Contributions

Extend Igarashi and Kobayashi work to handle

1. **Mutually recursive definitions;**
2. **Free variables within procedures.**

Outline

- TyCO via an an example
- Linear type assignment system
- Linear type inference system
- Optimizing linear channels
- Future work

An example—Sieve of Eratosthenes I

```
Sieve (inStream, grain, outStream) =
   inStream ? (n) :
      if n % grain /= 0
      then
         outStream ! [n] ;
         Sieve [inStream, grain, outStream]
      else
         Sieve [inStream, grain, outStream]
```

Sieve of Eratosthenes II

```
Sink (inStream) =
   inStream ? (n) :
      io ! puti [n] ;
      new newSieve
      Sink [newSieve]  |
      Sieve [inStream, n, newSieve]
```

Sieve—Putting everything together

```
def
   Ints (..) = . . .
   Sieve (..) = . . .
   Sink (..) = . . .
in
   new aStream
   Ints [aStream] |
   Sink [aStream]
```

Syntax of core-TyCO

$$
\begin{array}{lcl}
P & ::= & a\,![\vec{v}] \;\mid\; a\,?\,\{l_i(\vec{x}_i) = P_i\}_{i\in I} \;\mid\; P \mid Q \;\mid\; 0 \;\mid\; \textsf{new}\; x\; P \;\mid \\
 & & X[\vec{v}] \;\mid\; \textsf{def}_{i\in I}\; X_i(\vec{x}_i) = P_i \;\textsf{in}\; Q
\end{array}
$$

- **Add a conditional construct, and expressions built from channels, base types (integers, booleans, strings, floats), and primitive operations.**
- **The remaining constructs are translated at parsing time into the core.**

Reduction

$$\textsc{Com}\quad a\,!\,l_j[\vec{v}] \mid a\,?\,\{l_i(\vec{x}_i) = P_i\}_{i\in I} \to \{\vec{v}/\vec{x}_j\}P_j$$

$$\textsc{Call}\quad \mathsf{def}_{i\in I}\ X_i(\vec{x}_i) = P_i\ \mathsf{in}\ X_j[\vec{v}] \mid Q \to \mathsf{def}_{i\in I}\ X_i(\vec{x}_i) = P_i\ \mathsf{in}\ \{\vec{v}/\vec{x}_j\}P$$

$$\textsc{Res}\ \frac{P \to Q}{\mathsf{new}\ x\ P \to \mathsf{new}\ x\ Q} \qquad \textsc{Par}\ \frac{P \to Q}{P \mid R \to Q \mid R}$$

$$\textsc{Def}\ \frac{P \to Q}{\mathsf{def}\ D\ \mathsf{in}\ P \to \mathsf{def}\ D\ \mathsf{in}\ Q} \qquad \textsc{Str}\ \frac{P \equiv R \quad R \to S \quad S \equiv Q}{P \to Q}$$

Linear type inference system—Uses

0 **no communication is allowed on the channel;**

1 **at most one communication—a linear channel;**

ω **unbound number of communications on the channel.**

$\kappa_1 + \kappa_2$	0	1	ω
0	0	1	ω
1	1	ω	ω
ω	ω	ω	ω

$\kappa_1 \times \kappa_2$	0	1	ω
0	0	0	0
1	0	1	ω
ω	0	ω	ω

$\kappa_1 \sqcup \kappa_2$	0	1	ω
0	0	1	ω
1	1	1	ω
ω	ω	ω	ω

Types

$$\rho \quad ::= \quad \{l_i : \vec{\rho}_i\}_{i \in I}^{(\kappa_1, \kappa_2)} \quad | \quad t$$

Examples from the Sieve of Eratosthenes:

ack: $\{$**done:** $\}^{(1,1)}$

outStream: IntegerStream$^{(0,w)}$

Sieve: IntegerStream$^{(w,0)}$ **Integer IntegerStream**$^{(0,w)}$

Counting procedure calls

Let $D \stackrel{\text{def}}{=} (X_i(\vec{x}_i) = P_i)_{i \in I}$**. A function** $\mathcal{U}$ **is a *call counting function* if it satisfies the following requirements.**

1. $\mathcal{U}(X, D, Q) \geq \mathcal{U}(X, D, R)$**, if** $Q \rightarrow R$**,**
2. $\mathcal{U}(X, D, X_i[\vec{v}] \mid Q) =$

$$\begin{cases} 1 + \mathcal{U}(X, D, \{\vec{v}/\vec{x}_i\}P_i \mid Q) & \textbf{if } X = X_i \textbf{ for some } i, \\ \mathcal{U}(X, D, \{\vec{v}/\vec{x}_i\}P_i \mid Q) & \textbf{otherwise}. \end{cases}$$

Type assignment system

Subtyping

$$\frac{\kappa_1 \geq \mu_1 \qquad \kappa_2 \geq \mu_2 \qquad \mu_1 \geq 1 \textbf{ implies } \vec{\rho}_i \preceq \vec{\sigma}_i \qquad \mu_2 \geq 1 \textbf{ implies } \vec{\sigma}_i \preceq \vec{\rho}_i}{\{l_i : \vec{\rho}_i\}_{i \in I}^{(\kappa_1, \kappa_2)} \preceq \{l_i : \vec{\rho}_i\}_{i \in I}^{(\mu_1, \mu_2)}}$$

Type assignment system: not syntax directed

$$\frac{\Gamma, x : \rho \vdash P \qquad \sigma \preceq \rho}{\Gamma, x : \sigma \vdash P}$$

plus weak rules.

Subject reduction I

Two ingredients:

1. **Explicitly typed processes: new** $x : \rho\ P$**.**
2. **Use-aware reduction. Labels:** ϵ **communication on a bound channel or procedure call.**

$$\textsc{Com}_a \quad a\,!\,l_j[\vec{v}] \mid a\,?\,\{l_i(\vec{x}_i) = P_i\}_{i\in I} \xrightarrow{a} \{\vec{v}/\vec{x}_j\}P_j$$

$$\textsc{Res}_\epsilon \frac{P \xrightarrow{x} R}{\textsf{new}\ x : \alpha^{(\kappa_1,\kappa_2)} P \xrightarrow{\epsilon} \textsf{new}\ x : \alpha^{(\kappa_1^-,\kappa_2^-)} R}$$

$$\textsc{Res}_l \frac{P \xrightarrow{\ell} R \qquad \ell \neq x}{\textsf{new}\ x : \rho\ P \xrightarrow{\ell} \textsf{new}\ x : \rho\ R}$$

Subject reduction II

If $\Gamma \vdash P$ **and** $P \xrightarrow{\ell} Q$**, then** $\Gamma^{-\ell}$ **is defined and** $\Gamma^{-\ell} \vdash Q$**.**

$$\Gamma^{-\ell}(a) = \begin{cases} \Gamma(a) & \textbf{if } a \neq \ell, \\ \alpha^{(\kappa_1^-, \kappa_2^-)} & \textbf{if } \Gamma(a) = \alpha^{(\kappa_1, \kappa_2)} \textbf{ and } \kappa_1^-, \kappa_2^- \textbf{ defined}, \\ \textbf{undefined} & \textbf{otherwise}. \end{cases}$$

$\omega^- = \omega$

$1^- = 0$

0^- **undefined**

Type Inference—Constraints

$$\kappa \quad ::= \quad 0 \quad | \quad 1 \quad | \quad \omega \quad | \quad u \quad | \quad \kappa_1 + \kappa_2 \quad | \quad \kappa_1 \cdot \kappa_2 \quad | \quad \kappa_1 \sqcup \kappa_2$$

(Subtype) constraint set C is a set of subtype expressions $\rho_1 \preceq \rho_2$. Extend to typings.

Substitution S is a map from type vars into types (ρ/t), and from use vars into use expressions (κ/u).

Ground substitution S is a *solution* of C, if $S\rho_1 \preceq S\rho_2$ holds for all $\rho_1 \preceq \rho_2$ in C.

C_1 *satisfies* C_2 ($C_1 \models C_2$), if every solution of C_1 is also a solution of C_2.

Type Inference System

$$\textsc{Par}\ \frac{\Gamma_1; C_1 \vdash P_1 \qquad \Gamma_2; C_2 \vdash P_2 \qquad C \models \Gamma \preceq \Gamma_1 + \Gamma_2 \qquad C \models C_1 \cup C_2}{\Gamma; C \vdash P_1 \mid P_2}$$

$$\textsc{Msg}\ \frac{C \models \Gamma \preceq (a : \{l_i : \vec{\rho}_i\}_{i \in I}^{(0,1)}, \vec{v} : \vec{\rho}_i) \qquad j \in I}{\Gamma; C \vdash a\,!\,l_j[\vec{v}]}$$

$$\textsc{Res}\ \frac{\Gamma, x : \rho; C \vdash P}{\Gamma; C \vdash \textsf{new}\ x\ P} \qquad \textsc{Call}\ \frac{C \models \vec{\sigma} \preceq \vec{\rho}}{\Gamma, X : \vec{\rho}, \vec{v} : \vec{\sigma}; C \vdash X[\vec{v}]}$$

$$\textsc{Def}\ \frac{\begin{array}{c}\bigcup_{j \in I} X_j : \vec{\rho}_j, \Gamma_i, \vec{x}_i : \vec{\sigma}_i; C_i \vdash P_i, \quad \forall i \in I \qquad \bigcup_{j \in I} X_j : \vec{\rho}_j, \Delta; C' \vdash Q \\ C \models \Gamma \preceq (\Delta + \sum_{j \in I} \mathcal{U}(X_j, (X_i(\vec{x}_i) = P_i)_{i \in I}, Q) \times \Gamma_j) \\ C \models C' \qquad\qquad C \models \bigcup_{j \in I} (C_j \cup \{\vec{\sigma}_j \preceq \vec{\rho}_j\})\end{array}}{\Gamma; C \vdash \textsf{def}_{i \in I}\ X_i(\vec{x}_i) = P_i\ \textsf{in}\ Q}$$

Type Inference System—Main result

Let P be an explicitly typed process.

1. **If $\Gamma; C \vdash \mathrm{erase}(P)$, and S is a solution of C whose its domain includes all type/use variables in Γ and in P, then $S\Gamma \vdash SP$.**
2. **If $\Gamma \vdash P$, then $(\Gamma, \emptyset) \vdash \mathrm{erase}(P)$.**

Type Reconstruction

The pair $\Gamma; C$ is *principal* for P, if

1. $\Gamma; C \vdash P$**, and**
2. **If $\Delta; C' \vdash P$, then $\Delta; C'$ is an *instance* of $\Gamma; C$, that is, if there is a substitution S such that $C' \models SC$ and $S\Gamma \subseteq \Gamma'$.**

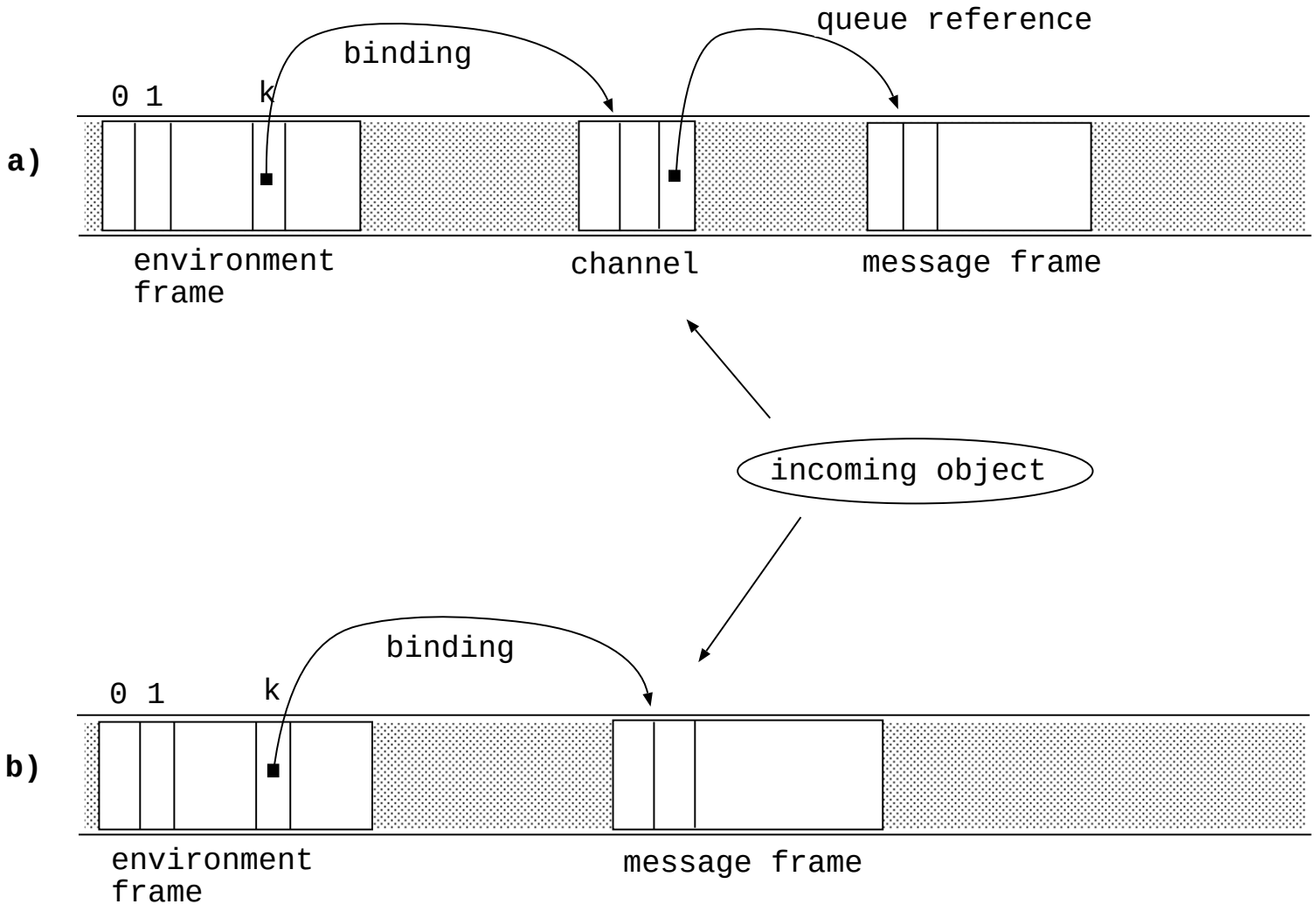
There is an algorithm, LTR *linear type reconstruction*,

1. **If LTR(P) outputs the pair $\Gamma; C$, then $\Gamma; C$ is principal for P;**
2. **If LTR(P) outputs *FAIL*, then $\Gamma; C \not\models P$ for all pairs $\Gamma; C$.**

Computing the use of a process variable

$$
\begin{aligned}
\mathcal{W}(Y, D, \mathsf{0}, V) &= \mathcal{W}(Y, D, a\,!\,l[\vec{v}], V) \;=\; 0 \\
\mathcal{W}(Y, D, P \mid Q, V) &= \mathcal{W}(Y, D, P, V) + \mathcal{W}(Y, D, Q, V) \\
\mathcal{W}(Y, D, a\,?\,\{l_i(\vec{x}_i) = P_i\}_{i \in I}, V) &= \bigsqcup_{i \in I} \mathcal{W}(Y, D, P_i, V) \\
\mathcal{W}(Y, D, \textsf{new}\ x\ P, V) &= \mathcal{W}(Y, D, P, V) \\
\mathcal{W}(Y, D, Y[\vec{v}], V) &= 1, \quad \textbf{if } Y \notin \mathrm{bv}(D) \\
\mathcal{W}(Y, D, Z[\vec{v}], V) &= 0, \quad \textbf{if } Z \notin \mathrm{bv}(D) \textbf{ and } Y \neq Z \textbf{ or} \\
&\qquad\qquad Z \in \mathrm{bv}(D), Z \in V\textbf{, and } Z \not\to Y \\
\mathcal{W}(Y, D, Z[\vec{v}], V) &= \omega, \quad \textbf{if } Z \in \mathrm{bv}(D), Z \in V\textbf{, and } Z \to Y \\
\mathcal{W}(X_i, D, X_i[\vec{v}], V) &= 1 + \mathcal{W}(X_i, D, P_i, V \cup \{X_i\}), \quad \textbf{if } X_i \notin V \\
\mathcal{W}(Y, D, X_i[\vec{v}], V) &= \mathcal{W}(Y, D, P_i, V \cup \{X_i\}), \\
&\qquad\qquad \textbf{if } X_i \notin V\textbf{, and } Y \neq X_i \\
\mathcal{W}(Y, D, \textsf{def}\ D'\ \textsf{in}\ Q, V) &= \mathcal{W}(Y, D', Q, V) + \\
&\qquad \sum_{i \in I} \mathcal{W}(X_i, D', Q, V) \times \mathcal{W}(Y, D, P_i, V
\end{aligned}
$$

Optimizing linear channels



Results

program	time(s)			heap(kw)			♯gc		
	¬opt	opt	%	¬opt	opt	%	¬opt	opt	%
`tak 22,16,8`	2.03	1.78	87	19458	15035	77	278	208	74
`fib 30`	3.55	3.16	89	32501	24376	75	460	331	71
`sieve 10000`	2.93	2.48	84	22203	18387	82	414	331	79

Further work

1. **Recursive types; (predicative) polymorphism.**
2. **Fixing the compiler ;-)**
3. **More on call-counting functions.**