Polymorphic Time Systems for Estimating Program Complexity

Vincent Dornic, Pierre Jouvelot, and David K. Gifford
LOPLAS 1992

Presented by Oukseh Lee

Research On Program Analysis System, KAIST

April 15, 2000

Type and Language

$$
\begin{array}{llll}
\mathit{Id} & i & & \\
\mathit{Kind} & k ::= & \mathtt{type} \mid \mathtt{time} & \\
\mathit{Descr} & d ::= & \tau \mid m & \\
\mathit{Type} & \tau ::= & \mathtt{rec}\; i\; \tau & \text{recursive type} \\
 & \mid & \tau \xrightarrow{m} \tau & \text{abstraction type} \\
 & \mid & \forall(i{:}\, k).\tau & \text{polymorphic type} \\
 & \mid & i & \\
\mathit{Time} & m ::= & 1 \mid 2 \mid 3 \mid \cdots \mid \mathtt{long} & \\
 & \mid & m + m & \text{time accumulation} \\
 & \mid & i & \\
\mathit{Expr} & e ::= & i & \\
 & \mid & \lambda(i{:}\, \tau).e & \text{abstraction} \\
 & \mid & e\; e & \text{application} \\
 & \mid & \Lambda(i{:}\, k).e & \text{polymorphic abstraction} \\
 & \mid & e\,[d] & \text{projection}
\end{array}
$$

Semantics

$$\frac{\sigma(i) = v}{\sigma \vdash i \rightarrow v, 1}\,(D.Env)$$

$$\sigma \vdash \lambda(i{:}\,\tau).e \rightarrow \langle i, e, \sigma \rangle, 1 \quad (D.Lambda)$$

$$\frac{\begin{array}{l}\sigma \vdash e_0 \rightarrow \langle i, e', \sigma' \rangle, n_0 \\ \sigma \vdash e_1 \rightarrow v', n_1 \\ \sigma'[i \mapsto v'] \vdash e' \rightarrow v, n\end{array}}{\sigma \vdash e_0\ e_1 \rightarrow v, 1 + n + n_0 + n_1}\,(D.Apply)$$

$$\frac{\sigma_i \vdash e \rightarrow v, n}{\sigma \vdash \Lambda(i{:}\,k).e \rightarrow v, n}\,(D.Plambda)$$

$$\frac{\sigma \vdash e \rightarrow v, n}{\sigma \vdash e\,[d] \rightarrow v, n}\,(D.Proj)$$

Typing

$$\frac{\Gamma \vdash e : \tau/m \quad \tau \sim \tau' \quad m \sim m'}{\Gamma \vdash e : \tau'/m'} \,(S.Equiv)$$

$$\frac{\Gamma(i) = \tau}{\Gamma \vdash i : \tau/1} \,(S.Env)$$

$$\frac{\Gamma[i{:}\,\tau_1] \vdash e : \tau_0/m \quad \Gamma \vdash \tau_1 :: \texttt{type}}{\Gamma \vdash \lambda(i{:}\,\tau_1).e : \tau_1 \xrightarrow{m} \tau_0/1} \,(S.Lambda)$$

$$\frac{\Gamma \vdash e_0 : \tau' \xrightarrow{m} \tau/m_0 \quad \Gamma \vdash e_1 : \tau'/m_1}{\Gamma \vdash e_0\ e_1 : \tau/m_0 \,\texttt{+}\, m_1 \,\texttt{+}\, m \,\texttt{+}\, 1} \,(S.Apply)$$

$$\frac{\Gamma[i :: k] \vdash e : \tau/m}{\Gamma \vdash \Lambda(i{:}\,k).e : \forall(i{:}\,k).\tau/m} \,(S.Plambda)$$

$$\frac{\Gamma \vdash e : \forall(i{:}\,k).\tau/m \quad \Gamma \vdash d :: k}{\Gamma \vdash e\,[d] : \tau[d/i]/m} \,(S.Proj)$$

$\sim$ on *Type* : α-conversion.

$\sim$ on *Time* : **+** is associative, commutative, and absorbed by `long`.

Consistency and a Correct Algorithm

- **Theorem 3 (Consistency)**

$$\begin{array}{l} \sigma \vdash e \to n, v \\ \Gamma \vdash e : t/m \\ K \models \sigma : \Gamma \end{array} \quad \Rightarrow \quad \begin{array}{l} K \models v : t \\ n \leq m \end{array}$$

- **Theorem 4 (Correctness)**

$$\begin{array}{l} \Gamma \vdash e : \tau/m \\ (\tau', m') = \mathtt{TTCA}(\Gamma, e) \end{array} \quad \Rightarrow \quad \begin{array}{l} \tau \sim \tau' \\ m \sim m' \end{array}$$

Limitation

- If-expression: "$\texttt{if}\ e_1\ e_2\ e_3$" takes the **`sumtime`** of e_1, e_2 and e_3.
  - needs **`max`** operator for a better estimate.
- Every recursive function takes `long` time.
  - expected rule:

$$\frac{\Gamma[f\colon \tau_1 \stackrel{m}{\to} \tau_2] \vdash \lambda i.e : \tau_1 \stackrel{m}{\to} \tau_2 / m'}{\Gamma \vdash \texttt{fix}\ f\ \lambda i.e : \tau_1 \stackrel{m}{\to} \tau_2 / m'}\ (S.Rec)$$

  - recursive function definition: $\texttt{rec}\ (f\ i)\ e \equiv (\texttt{Y}\ \lambda f.\lambda i.e)$.
  - $\texttt{Y}$: $\forall (t : \texttt{type})(m_1 : \texttt{time})(m_2 : \texttt{time})$.
    $((\cdot \stackrel{6+m_1}{\to} (t \stackrel{m_2}{\to} t)) \stackrel{m_1}{\to} (t \stackrel{m_2}{\to} t)) \stackrel{6+m_1}{\to} (t \stackrel{m_2}{\to} t)$.
  - $\texttt{FACT}$: $\forall (m_3 : \texttt{time})(m_4 : \texttt{time})$.
    $(\cdot \stackrel{m_3}{\to} (\texttt{int} \stackrel{m_4}{\to} \texttt{int})) \stackrel{1}{\to} (\texttt{int} \stackrel{16+m_3+m_4}{\to} \texttt{int})$.