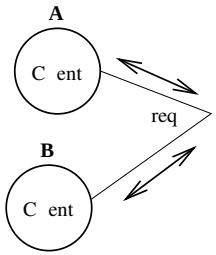
Assigning Types to Processes

NOBUKO YOSHIDA and MATTHEW HENNESSY

ABSTRACT In wide area distributed systems it is now common for *higher-order code*

➖YPING P OCE SS S E S

β and co un cat on co ¯Both these requ re a de n t on of subst tut on of va ues for var ab e


A
C ent
req
B
C ent

on part A.

–o der ve the. udge ent t s suf c ent to prove that for any w n dom$(\Delta \sqcap \Delta_2)$, $\Gamma \vdash \Delta(w) \leq (\Delta \sqcap \Delta_2)(w)$.– here are three poss b t es for w t s e ther n dom(Δ) ∩ dom(Δ_2), n dom(Δ) − dom(Δ_2) or dom(Δ_2) − dom(Δ). In the rst case we have, fro the hypothes s, that $\Gamma \vdash \Delta(w) \leq \Delta_i(w)$ and we ay app y nduct on, on part A to obta n $\Gamma \vdash \Delta(w) \leq \Delta(w) \sqcap \Delta_2(w)$ and the resu t fo ows, because n th s case $(\Delta \sqcap \Delta_2)(w) = \Delta(w) \sqcap \Delta_2(w)$.

–he other two poss b t es for w are s ar but s p er the nduct ve step s not requ red.–

Parts C and D are a so proved s u taneous y, th s t e by s u taneous nduct on on the de n t on of the operators $\sqcap$ and $\sqcup$.

(Common)

$$\text{(AL)}\ \frac{\vdash \Gamma, u\bullet \tau, \Gamma'\bullet \mathtt{Env}}{\Gamma, u\bullet \tau, \Gamma' \vdash u\bullet \tau} \qquad \text{(CON)}\ \frac{\vdash \Gamma\bullet \mathtt{Env}}{\Gamma \vdash \ \bullet \mathtt{nat}} \quad \text{etc}^{-}$$

$$\text{(SUB}_H\text{)}\ \frac{\Gamma \vdash P\bullet \rho \quad \Gamma \vdash \rho \leq \rho'}{\Gamma \vdash P\bullet \rho'} \qquad \text{(SUB}_N\text{)}\ \frac{\Gamma \vdash u\bullet \sigma \quad \Gamma \vdash \sigma \leq \sigma'}{\Gamma \vdash u\bullet \sigma'}$$

(Function)

$$\text{(ABS}_H\text{)}\ \frac{\Gamma, X\bullet \sigma_H \vdash P\bullet \rho}{\Gamma \vdash \lambda(X\bullet \sigma_H)P\bullet \sigma_H \to \rho} \qquad \text{(APP}_H\text{)}\ \frac{\Gamma \vdash P\bullet \sigma_H \to \rho \quad \Gamma \vdash Q\bullet \sigma_H}{\Gamma \vdash PQ\bullet \rho}$$

$$\text{(ABS}_N\text{)}\ \frac{\Gamma, x\bullet \sigma \vdash P\bullet \rho}{\Gamma \vdash \lambda(x\bullet \sigma)P\bullet (x\bullet \sigma) \to \rho} \qquad \text{(APP}_N\text{)}\ \frac{\Gamma \vdash P\bullet (x\bullet \sigma) \to \rho \quad \Gamma \vdash u\bullet \sigma}{\Gamma \vdash Pu\bullet \rho\{u/x\}}$$

(Process)

$$\text{(NIL)}\ \frac{\vdash \Gamma\bullet \mathtt{Env}}{\Gamma \vdash \mathbf{0}\bullet [\,]} \qquad \text{(PA)}\ \frac{\Gamma \vdash P\ ,\bullet \pi}{\Gamma \vdash P\ |P,\bullet \pi} \qquad \text{(REP)}\ \frac{\Gamma \vdash P\bullet \pi}{\Gamma \vdash *P\bullet \pi} \qquad \text{(RES)}\ \frac{\Gamma, a\bullet \sigma \vdash P\bullet \pi}{\Gamma \vdash (\nu a\bullet \sigma)P\bullet \pi/a}$$

$$\text{(OUT)}\ \frac{\pi \vdash_\Gamma u\bullet (\tau\ ,\ldots,\tau_n)^{\mathrm{O}} \qquad \Gamma \vdash P\bullet \pi \qquad \Gamma \vdash V_i\bullet \tau_i \quad \tau_i = \sigma_i \Rightarrow \pi \vdash_\Gamma V_i\bullet \sigma_i}{\Gamma \vdash u\ \langle V\ ,\ldots,V_n\rangle P\bullet \pi}$$

$$\text{(IN)}\ \frac{\pi \vdash_\Gamma u\bullet (\tau\ ,\ldots,\tau_n)^{\mathrm{I}} \qquad \Gamma, x\ \bullet \tau\ ,\ldots,x_n\bullet \tau_n \vdash P\bullet \pi, x\ \bullet \tau\ ,\ldots,x_n\bullet \tau_n}{\Gamma \vdash u\ (x\ \bullet \tau\ ,\ldots,x_n\bullet \tau_n)P\bullet \pi}$$

FIG. E ⁻-yp ng yste for $\lambda\pi_{\mathrm{v}}$

-he correspond ng e nat on (APP$_N$) a ows dyna c channe nstant at on nto types dur ng β reduct on⁻ If a ter P has a type $(x\bullet \sigma) \to \rho$ we can ap p y a na e a whose type s ess than σ to P⁻-hen a s subst tuted for x n ρ⁻

$$\frac{\Gamma \vdash P\bullet (x\bullet \sigma) \to \rho, \quad \Gamma \vdash a\bullet \sigma}{\Gamma \vdash Pa\bullet \rho\{a/x\}}$$

As an exa p e of the use of th s ru e cons der the channe abstract on $P \equiv \lambda(x\bullet \mathtt{nat})(x\ \langle\ \rangle\ |\ b$

s the process type wh ch aps b to the sa e type $(\texttt{int})^{\mathsf{O}}$ – hen w th the output ru e, together w th NIL and the abstract on ru es, we can estab sh

$$\Delta_{ab} \vdash b \langle\ \rangle \mathbf{0}\bullet\ [\Delta_b]$$

and therefore

$$\Delta_{ab} \vdash a\ \langle b \langle\ \rangle \mathbf{0}\rangle \mathbf{0}\bullet\ [a\bullet\ \langle \Delta_b \rangle^{\mathsf{O}}]$$

– HE INP – LE, IN • – he ru e for pre x ng s a stra ghtforward genera sa t on of that n [illegible]

$$\pi \vdash_\Gamma u\bullet\ (\tau)^{\mathsf{I}} \qquad \Gamma, x\bullet\ \tau \vdash P\bullet\ \pi, x\bullet\ \tau$$

An application of the rule (Out) gives the judgement

$$x\bullet\,(\mathtt{int})^{\mathtt{I}},\ y\bullet\,(\mathtt{int})^{\mathtt{O}},z\bullet\,\mathtt{int}\vdash y\ \langle z\rangle\bullet\,[\Delta_{xy}]$$

where Δ_{xy} denotes the interface $\{x\bullet\,(\mathtt{int})^{\mathtt{I}},\ y\bullet\,(\mathtt{int})^{\mathtt{O}}\}^{-}$ An application of the input rule (In) followed by an application of (Rep) now gives

$$x\bullet\,(\mathtt{int})^{\mathtt{I}},\ y\bullet\,(\mathtt{int})^{\mathtt{O}}\vdash *\,x\ (z\bullet\,\mathtt{int})\,y\ \langle z\rangle\bullet\,[\Delta_{xy}]$$

Now we may apply the channel abstraction rule (Abs$_N$) twice to obtain the following typing for the forwarder

$$\vdash \mathsf{Fw}\bullet\,(x\bullet\,(\mathtt{int})^{\mathtt{I}})\to(y\bullet\,(\mathtt{int})^{\mathtt{O}})\to[\Delta_{xy}]$$

Let us now see how we can use this typing to assign a type to the process R, as discussed in the Introduction.

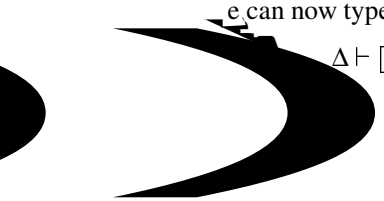
$$R\Longleftarrow s\ \langle c\rangle\ c\ (y\bullet\,\tau_{\mathrm{fw}})\ (y\,a\,b)$$

For convenience τ_{fw} denotes the type assigned to the forwarder and let us define

$$\Delta_R\stackrel{\mathrm{def}}{=}\{a\bullet\,(\mathtt{int})^{\mathtt{I}},b\bullet\,(\mathtt{int})^{\mathtt{O}},c\bullet\,(

e can now type the co b ned syste ¯By s proc n F gure we now

$\Delta \vdash [\text{req}_\bullet((\tau$



ub ect educt on aga n t ay be v ewed as a genera sat on of Le a

LEMMA

$$a\ (x_1 \bullet \tau_1, \ldots, x_n \bullet \tau_n)P \xrightarrow{\Gamma,\pi}_{err} \quad \text{if } \Gamma \not\vdash [a \bullet (\tau_1, \ldots, \tau_n)^{\mathrm{I}}] \leq \pi$$

$$a\ \langle V_1, \ldots, V_n \rangle P \xrightarrow{\Gamma,\pi}_{err} \quad \text{if no } \tau_i \text{ is } \overline{\Gamma} \text{ such that } \Gamma \vdash [a \bullet (\tau_1, \ldots, \tau_n)^{\mathrm{O}}] \leq \pi \text{ and } \Gamma \vdash V_i \bullet \tau_i$$

$$\frac{P \xrightarrow{(\Gamma, a \bullet \sigma),\pi}_{err}}{(\nu\, a \bullet \sigma)P \xrightarrow{\Gamma,(\pi/a)}_{err}} \qquad \frac{P \xrightarrow{\Gamma,\pi} \text{ or } Q \xrightarrow{\Gamma,\pi}}{P \mid Q \xrightarrow{\Gamma,\pi}_{err}} \qquad \frac{P \xrightarrow{\Gamma,\pi}_{err}}{*P \xrightarrow{\Gamma,\pi}_{err}}$$

FIGURE — untagged errors

Analysing the hypothesis we obtain

$$\Gamma, x \bullet \sigma \vdash P \bullet [\Delta_1, x \bullet \sigma] \quad \text{with } \Gamma, x \bullet \sigma \vdash [u \bullet (\sigma)^{\mathrm{I}}] \leq [\Delta_1] \leq [\Delta] \quad x \notin \mathrm{fv}(\Delta_1)$$
$$\Gamma \vdash Q \bullet [\Delta_2] \quad \text{with } \Gamma \vdash [u \bullet (\sigma')^{\mathrm{O}}, v \bullet \sigma'] \leq [\Delta_2] \leq [\Delta]$$
$$\Gamma \vdash v \bullet \sigma'$$

Noting $x \notin \mathrm{fv}(\sigma)$, we can apply Channel narrowing. Let Γ' be $\Gamma, u \bullet (\sigma)^{\mathrm{I}}$ to obtain $\Gamma' \vdash [\Delta_1] \leq [\Delta]$. Then we have $\Gamma' \vdash \Gamma(u) \leq \Delta(u) \leq \Delta_1(u) \leq (\sigma)^{\mathrm{I}}$ and $\Gamma' \vdash \Gamma(u) \leq \Delta(u) \leq \Delta_2(u) \leq (\sigma')^{\mathrm{O}}$, which by applying $\Gamma \vdash \sigma' \leq \sigma$

using subsumption we then have $\Gamma \vdash v \bullet \sigma$ and so we can apply substitution. Let Γ'' be $\Gamma, v \bullet \sigma$. Let Δ' be $\Delta, x \bullet \sigma$ to obtain $\Gamma'' \vdash P\{v/x\} \bullet [\Delta', x \bullet \sigma]\{v/x\}$. By calculation on the sorts $[\Delta_1] \sqcup [v \bullet \sigma]$ and we have $\Gamma'' \vdash [\Delta_1] \sqcup [v \bullet \sigma] \leq [\Delta_1] \sqcup [v \bullet \sigma'] \leq [\Delta_2] \leq [\Delta]$. Hence by subsumption we have the required $\Gamma'' \vdash P\{v/x\} \bullet [\Delta]$ □

Type safety

Our typing system is an extension of that for the λ-calculus from [] and that for the π-calculus from [], consequently it guarantees the absence of the type calculus run time errors associated with these languages. Rather than duplicate the formulation of these kinds of errors, which involves the development of complicated *tagging* notation, here we concentrate on the novel run time type errors which our typing system can catch.

Intuitively $\Gamma \vdash P \bullet \pi$ should mean that, assuming the environment Γ, the process P satisfies the *interface* π. If π is the undifferentiated type `proc` then, viewed as an interface, it provides no information. However if it has the form $[\Delta]$ this means that P can use *at most* the resources mentioned in Δ. Moreover these resources can only be used according to the capabilities they are assigned in Δ. As a step for a satisfaction of this intuitive idea, given in Figure, using a unary predicate $P \xrightarrow{\Gamma,\pi}_{err}$ — the first two clauses are the most significant — the first says that, relative to Γ, P violates the interface π if it can input on the channel a but the interface π does not assign any input capability to a. The second is similar, but for output.

Syntax: others fro F gure 2

yste	$M, N, \ldots ::=$	$P \mid N \parallel M \mid (\nu a, \sigma) N \mid \mathbf{0}$	
S–er	$P, Q, \ldots ::=$	$\mathtt{Spawn}(P) \mid \cdots$	as n F gure 2

–YPED BEHA IO AL EQ ALI–Y –ypes constra n the behav our of processes and the r env ron ents and consequent y have an pact on when the r behav our shou d be dee ed to be equ va ent – –yped behav oura equ va ences have a ready been nvest gated for var ous process ca cu n papers such as ar techn ques cou d be app ed to our anguage resu t ng n a new typed equ va ence where equa t es are n uenced by the presence of ne gra ned pro cess types – Invest gat on of such equ va ences s an nterest ng research top c part cu ar y n ts app cat on to the re ne ent of the context equa ty of we eave th s for future wor –

–YPE LIMI–A–ION One tat on of our typ ng syste s that wh e na e var ab es n types can be abstracted by channe dependency types

(Free Names)

Terms

$\mathsf{fn}(\mathbf{0}) = \mathsf{fn}(l) = \mathsf{fn}(x) = \emptyset \quad \mathsf{fn}(a) = \{a\}$

$\mathsf{fn}(P \mid Q) = \mathsf{fn}(PQ) = \mathsf{fn}(P) \cup \mathsf{fn}(Q)$

$\mathsf{fn}(*P) = \mathsf{fn}(P)$

$\mathsf{fn}(u\ (x_1 : \tau_1, \ldots, x_n : \tau_n)P)$
$= \mathsf{fn}(u) \cup \mathsf{fn}(\tau_1) \cup \ldots \cup \mathsf{fn}(P$

G aca one A M stra P and Prasad Operat ona and A gebra c e ant cs for Fac o A
y etr c Integrat on of Concurrent and Funct ona Progra ng