Subtyping for Binary Sessions[1]

[1] Simon J. Gay, Malcolm Hole: Subtyping for session types in the pi calculus. Acta Inf. (2005)

Principio de sustitutividad

$$\sigma \leqslant \tau$$

- Lectura: "En todo contexto donde se espera una expresión de tipo τ, puede utilizarse una de tipo σ en su lugar sin que ello genere un error"
- Esto se refleja con una nueva regla de tipado llamada Subsumption:

$$\frac{\Gamma \vdash M : \sigma \qquad \sigma \leqslant \tau}{\Gamma \vdash M : \tau}\text{[T-Subs]}$$

Subtipado de tipos función

$$\frac{\sigma' \leqslant \sigma \quad \tau \leqslant \tau'}{\sigma \to \tau \ \leqslant \ \sigma' \to \tau'}\text{[S-Func]}$$

- Observar que el sentido de $\leqslant$ se da "vuelta" para el tipo del argumento de la función pero no para el tipo del resultado
- Se dice que el constructor de tipos función es contravariante en su primer argumento y covariante en el segundo.

Subtipado de tipos función

$$\frac{\sigma' \leqslant \sigma \quad \tau \leqslant \tau'}{\sigma \to \tau \;\leqslant\; \sigma' \to \tau'}\text{[S-Func]}$$

Si un contexto/programa P espera una expresión f de tipo $\sigma' \to \tau'$ puede recibir otra de tipo $\sigma \to \tau$ si dan las condiciones indicadas

- Toda aplicación de f se hace sobre un argumento de tipo σ'
- El argumento se coerciona al tipo σ
- Luego se aplica la función, cuyo tipo real $\sigma \to \tau$
- Finalmente se coerciona el resultado a τ', el tipo del resultado que espera P

Agregando subsumption

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma}\text{[T-Var]} \qquad \frac{\Gamma \vdash M : \sigma \to \tau \quad \Gamma \vdash N : \sigma}{\Gamma \vdash M\, N : \tau}\text{[T-App]}$$

$$\frac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma \vdash \lambda x : \sigma.M : \sigma \to \tau}\text{[T-Abs]} \qquad \frac{\Gamma \vdash M : \sigma \quad \sigma \leqslant \tau}{\Gamma \vdash M : \tau}\text{[T-Subs]}$$

- Con subsumption ya no son dirigidas por sintaxis.
- No es evidente cómo implementar un algoritmo de chequeo de tipos a partir de las reglas.

“Cableando” subsumption dentro de las demás reglas

- Un análisis rápido determina que el único lugar donde se precisa subtipar es al aplicar una función a un argumento
- Esto sugiere la siguiente formulación donde

$$\frac{x : \sigma \in \Gamma}{\Gamma \vdash x : \sigma}\,\text{[T-Var]} \qquad \frac{\Gamma \vdash M : \sigma \to \tau \quad \Gamma \vdash N : \rho \quad \rho \leqslant \sigma}{\Gamma \vdash M\,N : \tau}\,\text{[T-App]}$$

$$\frac{\Gamma, x : \sigma \vdash M : \tau}{\Gamma \vdash \lambda x : \sigma . M : \sigma \to \tau}\,\text{[T-Abs]}$$

Typing with subtyping

$$\frac{\Gamma_1 \vdash P_1 \qquad \Gamma_2 \vdash P_2}{\Gamma_1 + \Gamma_2 \vdash P_1 | P_2}\text{[T-Par]} \qquad \frac{\Gamma, x^+ : S, x^- : \overline{S} \vdash P}{\Gamma \vdash (\nu x{:}S)P}\text{[T-Res]}$$

$$\frac{\Gamma_1 \vdash v : t \quad \Gamma_2, x^p : S \vdash P \quad t \leqslant s}{\Gamma_1 + (\Gamma_2, x^p : \mathtt{!}s\mathtt{.}S) \vdash x^p\mathtt{!}v\mathtt{.}P}\text{[T-Out]} \qquad \frac{\Gamma, x^p : S, y : t \vdash P \quad s \leqslant t}{\Gamma, x^p : \mathtt{?}s\mathtt{.}S \vdash x^p\mathtt{?}(y{:}t)\mathtt{.}P}\text{[T-In]}$$

$$\frac{\Gamma, x^p : S_j \vdash P \qquad j \in I}{\Gamma, x^p : \oplus[\mathtt{l}_i : S_i]_{i \in I} \vdash x^p \triangleleft \mathtt{l}_j\mathtt{.}P}\text{[T-Choice]} \qquad \frac{I \subseteq J \quad \Gamma, x^p : S_i \vdash P_i \quad \forall i \in I}{\Gamma, x^p : \&[\mathtt{l}_i : S_i]_{i \in I} \vdash x^p \triangleright [\mathtt{l}_j : P_j]_{j \in J}}\text{[T-Branch]}$$

$$\frac{\Gamma \text{ completed}}{\Gamma \vdash 0}\text{[T-Nil]}$$

Subtyping for non-recursive types

$$\mathsf{end} \leqslant \mathsf{end} \text{ [S-End]}$$

$$\frac{s \leqslant t \qquad S \leqslant T}{?s.S \leqslant ?t.T}\text{[S-InS]}$$

$$\frac{t \leqslant s \qquad S \leqslant T}{!s.S \leqslant !t.T}\text{[S-OutS]}$$

$$\frac{I \subseteq J \qquad \forall i \in I. S_i \leqslant T_i}{\&[\mathtt{l}_i : S_i]_{i \in I} \leqslant \&[\mathtt{l}_j : T_{\mathtt{j}}]_{j \in J}}\text{[S-Branch]}$$

$$\frac{J \subseteq I \qquad \forall j \in J. S_j \leqslant T_j}{\oplus[\mathtt{l}_i : S_i]_{i \in I} \leqslant \oplus[\mathtt{l}_j : T_i]_{j \in J}}\text{[S-Choice]}$$

Infinite types

Several ways of writing a type

$$\begin{array}{l}\mu X.!\mathtt{int}.X\\ !\mathtt{int}.\mu Y.!\mathtt{int}.Y\\ !\mathtt{int}.!\mathtt{int}.\mu Y.!\mathtt{int}.Y\\ \mu X.!\mathtt{int}.!\mathtt{int}.X\\ \mu X.\mu Y.!\mathtt{int}.X\end{array}$$

unfold(_)

$$unfold(T) = \begin{cases} unfold(S\{\mu X.S/X\}) & \textit{if } T = \mu X.S \\ T & \text{otherwise} \end{cases}.$$

$unfold(T)$ terminates for all t (because types are contractive)

Type Simulation

$\mathbb{T}$ is the set of closed types, and assume the subtyping relation $\prec$ on basic types.

Type Simulation

A relation $\mathcal{R} \subseteq \mathbb{T} \times \mathbb{T}$ is a type simulation if $(s, t) \in \mathcal{R}$ implies:

1. If $unfold(s) = \mathsf{end}$ then $unfold(t) = \mathsf{end}$.
2. If $unfold(s) = ?t_1.S_1$ then $unfold(t) = ?t_2.S_2$ and $(S_1, S_2) \in \mathcal{R}$ and $(t_1, t_2) \in \mathcal{R}$.
3. If $unfold(s) = !t_1.S_1$ then $unfold(t) = !t_2.S_2$ and $(S_1, S_2) \in \mathcal{R}$ and $(t_2, t_1) \in \mathcal{R}$.
4. If $unfold(s) = \&[\mathtt{l}_i : S_i]_{i \in I}$ then $unfold(t) = \&[\mathtt{l}_j : T_j]_{j \in J}$ and $I \subseteq J$ and $(S_i, T_i) \in \mathcal{R}$ for all $i \in I$.
5. If $unfold(s) = \oplus[\mathtt{l}_i : S_i]_{i \in I}$ then $unfold(t) = \oplus[\mathtt{l}_j : T_j]_{j \in J}$ and $J \subseteq I$ and $(S_j, T_j) \in \mathcal{R}$ for all $j \in J$.
6. If $unfold(s) = [s']$ then $unfold(t) = [t']$ and $(s', t') \in \mathcal{R}$.
7. if s and t are basic types, then $s \prec t$.

(Coinductive) Subtyping

The *coinductive subtyping relation* $\leqslant_c$ is defined by $S \leqslant_c T$ if and only if there exists a type simulation $\mathcal{R}$ such that $(S, T) \in \mathcal{R}$.

Coinductive subtyping

Example

$$S = \mu X.\,!\mathtt{int}.X$$
$$T = \,!\mathtt{float}.\mu Y.\,!\mathtt{float}.Y$$

We show that $\mathcal{R} = \{(\mathtt{int}, \mathtt{float}), (T, S), (\mu Y.\,!\mathtt{float}.Y, S)\}$ is type simulation.
Hence $T \leqslant S$

Coinductive duality

$\mathbb{S}$ is the set of closed session types

Duality

A relation $\mathcal{R} \subseteq \mathbb{S} \times \mathbb{S}$ is a duality relation if $(S, T) \in \mathcal{R}$ implies:

1. If $\mathit{unfold}(S) = \mathsf{end}$ then $\mathit{unfold}(T) = \mathsf{end}$.
2. If $\mathit{unfold}(S) = \mathtt{?}t_1\mathtt{.}S_1$ then $\mathit{unfold}(T) = \mathtt{!}t_2\mathtt{.}S_2$ and $(S_1, S_2) \in \mathcal{R}$ and $t_1 \leqslant_c t_2$ and $t_2 \leqslant_c t_1$.
3. If $\mathit{unfold}(S) = \mathtt{!}t_1\mathtt{.}S_1$ then $\mathit{unfold}(T) = \mathtt{?}t_2\mathtt{.}S_2$ and $(S_1, S_2) \in \mathcal{R}$ and $t_1 \leqslant_c t_2$ and $t_2 \leqslant_c t_1$.
4. If $\mathit{unfold}(S) = \&[\mathtt{l}_i : S_i]_{i \in I}$ then $\mathit{unfold}(T) = \oplus[\mathtt{l}_i : T_i]_{i \in I}$ and $(S_i, T_i) \in \mathcal{R}$ for all $i \in I$.
5. If $\mathit{unfold}(S) = \oplus[\mathtt{l}_i : S_i]_{i \in I}$ then $\mathit{unfold}(T) = \&[\mathtt{l}_i : T_i]_{i \in I}$ and $(S_i, T_i) \in \mathcal{R}$ for all $i \in I$.

(Coinductive) Duality

The *coinductive duality relation* $\perp_c$ is defined by $S \perp_c T$ if and only if there exists a duality relation $\mathcal{R}$ such that $(S, T) \in \mathcal{R}$.

Typing

$$\frac{\Gamma_1 \vdash P_1 \qquad \Gamma_2 \vdash P_2}{\Gamma_1 + \Gamma_2 \vdash P_1 | P_2}\,[\text{T-Par}] \qquad \frac{\Gamma, x^+ : S, x^- : \overline{S} \vdash P}{\Gamma \vdash (\nu x{:}S)P}\,[\text{T-Res}]$$

$$\frac{\Gamma_1 \vdash v : t \quad \Gamma_2, x^p : S \vdash P \quad t \leqslant_c s}{\Gamma_1 + (\Gamma_2, x^p : !s.S) \vdash x^p ! v.P}\,[\text{T-Out}] \qquad \frac{\Gamma, x^p : S, y : t \vdash P \quad s \leqslant_c t}{\Gamma, x^p : ?s.S \vdash x^p ?(y{:}t).P}\,[\text{T-In}]$$

$$\frac{\Gamma_1 \vdash v : t \quad \Gamma_2, x : [s] \vdash P \quad t \leqslant_c s}{\Gamma_1 + (\Gamma_2, x : [s]) \vdash x^p ! v.P}\,[\text{T-Out-Un}] \qquad \frac{\Gamma, x : [s], y : t \vdash P \quad s \leqslant_c t}{\Gamma, x : [s] \vdash x ?(y{:}t).P}\,[\text{T-In-Un}]$$

$$\frac{\Gamma, x^p : S_j \vdash P \quad j \in I}{\Gamma, x^p : \oplus[\mathtt{l}_i : S_i]_{i \in I} \vdash x^p \triangleleft \mathtt{l}_j.P}\,[\text{T-Choice}] \qquad \frac{I \subseteq J \quad \Gamma, x^p : S_i \vdash P_i \quad \forall i \in I}{\Gamma, x^p : \&[\mathtt{l}_i : S_i]_{i \in I} \vdash x^p \triangleright [\mathtt{l}_j : P_j]_{j \in J}}\,[\text{T-Branch}]$$

$$\frac{\Gamma \text{ completed}}{\Gamma \vdash 0}\,[\text{T-Nil}] \qquad \frac{\Gamma \vdash P \quad \Gamma \text{ Unlimited}}{\Gamma \vdash !P}\,[\text{T-Rep}]$$