PLP - Práctica 4: Subtipado

Zamboni, Gianfranco

19 de marzo de 2018

Reglas de subtipado

4.1. Ejercicio 1

1)

$$\frac{\{y\} \subseteq \{x, y, z\} \qquad y : Nat = y : Nat}{\{x : Nat,\ y : Nat,\ z : Nat\} <: \{y : Nat\}}\ \text{S-Rcd}$$

Esta demostración no es única porque existe la regla $S - Trans$ que nos permite probar la transitividad de los tipos, si hubiesemos decidido usarla para primero conseguir un supertipo $\{x : Nat, y : Nat\}$ del primer término y luego probar que ese supertipo es subtipo de $\{y : Nat\}$ entonces también hubiese sido una demostración válida.

2)

$$\frac{\emptyset \subseteq \{x, y, z\}}{\{x : Nat,\ y : Nat\} <: \{\}}\ \text{S-Rcd}$$

$$\frac{\dfrac{\{x\} \subseteq \{x, y\} \qquad x : Nat = x : Nat}{\{x : Nat,\ y : Nat\} <: \{x : Nat\}}\ \text{S-Rcd} \qquad \dfrac{\emptyset \subseteq \{x, y, z\}}{\{x : Nat\} <: \{\}}\ \text{S-Rcd}}{\{x : Nat,\ y : Nat\} <: \{\}}\ \text{S-Trans}$$

4.2. Ejercicio 2

1) Los registros tienen subtipos infinitos porque dado el tipo de un registro $\omega = \{l_i : \sigma_i\}_{i \in 1..n}$, entonces cualquier tipo de la forma $\omega' = \{l_i : \tau_i\}_{i \in 1..k}$ con $k \geqslant n$ tal que $\tau_i <: \sigma_i^{i \in 1..n}$ es subtipo del tipo de ω.

Top tiene como subtipo a los registros y los registros tienen infinitos subtipos, entonces Top tiene infinitos subtipos.

Por S-Arrow, los subtipos de una función $\sigma \to \tau$ son los tipos de la forma $\sigma' \to \tau'$ tal que $\sigma <: \sigma'$ y $\tau' <: \tau$. En particular si τ es de tipo registro, entonces τ tiene infinitos subtipos, por lo que $\sigma \to \tau$ tambien los tiene (son las funciones que devuelven registros).

2) Top no tiene supertipos.

Los registros tiene una cantidad finita de supertipos, siendo el máximo registro $\{\} <: Top$

Otra vez, hay casos en que las funciones tienen infinitos supertipos y es cuando toman como parámetro a un registro. Esto es porque la regla S-Arrow es contravariante respecto del tipo del párametro de la función, es decir, para que un tipo $\sigma \to \tau$ sea supertipo de $\sigma' \to \tau$ tiene que valer que $\sigma <: \sigma'$. Y si $sigma'$ es un registro, entonces tiene infinitos subtipos.

4.3. Ejercicio 3

1) $S = Top$

2) Si solo consideramos los tipos básicos $Bool$, Nat, Int, $Float$, entonces $S = Bool$, pero cuando empezamos a considerar registros o listas u otros tipos, entonces, los "mínimos" de cada tipo no están relacionados de ninguna forma, incluso, en el caso de los registros, ese mínimo nisiquiera existe.

3) Por S-Arrow, tenemos que $S_1 \to S_2 <: T_1 \to T_2$ si $T_1 <: S_1$ y $T_2 <: S_2$. El primer caso es el punto **1)**, el segundo es el punto **2)**.

4) Es similar al anterior pero con los casos invertidos.

4.4. Ejercicio 4

1) $T <: S \underset{\text{S-Trans}}{\iff} T <: T \ \wedge \ T <: S \underset{\text{S-Arrow}}{\iff} S \to T <: T \to T$

2) Si $S = Bool$ y $T = Top$, entonces $\{x : Bool, y : Top\}$ tiene 26 supertipos y el tipo $Bool \to Top$ solo tiene como supertipo a Top, porque $Bool$ no tiene subtipos y Top no tiene supertipos.

3) Si $S = Top$ y $T = Top$, entonces $\{x : Top, y : Top\}$ tiene como supertipos a $\{x : Top\}$, $\{y : Top\}$, $\{\}$ y a Top y el tipo $Top \to Top$ tiene infinitos por el ejercicio 2.

4.5. Ejercicio 5

Podemos pensar la demostración por inducción, los casos bases serían las relaciones entre los dos tipos básicos y las funciones que tienen como parámetro y argumento alguna combinación de estos tipos.

Primero, $Bool <: Nat$ y no hay más tipos que puedan ser subtipo de $Bool$, ni supertipo de Nat, por lo que esta relación es finita.

Las funciones "base" pueden ser de tipo $Bool \to Bool$, $Bool \to Nat$, $Nat \to Nat$ ó $Nat \to Bool$. Y las relaciones de subtipado quedan como siguen:

$$Nat \to Bool <: Bool \to Bool <: Bool \to Nat$$
$$Nat \to Bool <: Nat \to Nat <: Bool \to Nat$$

Luego están los tipos de funciones "recursivos" que tienen la forma $\sigma \to \tau$ con σ y τ cualquier tipo posible. Por inducción, podemos decir que σ y τ tienen finitos subtipos y finitos supertipos, por lo que la cantidad de supertipos $\#supertipos(\sigma \to \tau) = \#supertipos(\sigma) \times \#subtipos(\tau)$ y la cantidad de subtipos $\#subtipos(\sigma \to \tau) = \#subtipos(\sigma) \times \#supertipos(\tau)$ es finita.

Subtipado en el contexto de tipado

4.6. Ejercicio 6

a)

$$\dfrac{\dfrac{\dfrac{\dfrac{y : Nat \in \Gamma'}{\Gamma' \mapsto y : Nat}\text{ T-Var}}{\Gamma' = \Gamma, \{y : Nat\} \mapsto succ(y) : Nat}\text{ T-Succ}}{\Gamma \mapsto \lambda y : Nat.succ(y) : Nat \to Nat}\text{ T-Abs} \quad \dfrac{x : Bool \in \Gamma}{\Gamma \mapsto x : Bool}\text{ T-Var} \quad \dfrac{}{Bool <: Nat}\text{ S-BoolNat}}{\dfrac{\Gamma = \{x : Bool\} \mapsto (\lambda y : Nat.succ(y))\ x : Nat}{\varnothing \mapsto \lambda x : Bool.(\lambda y : Nat.succ(y))\ x : Bool \to Nat}\text{ T-Abs}}\text{ T-App}$$

b) $\Gamma = \{r : \{l_1 : Bool, l_2 : Float\}\}$

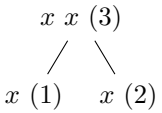
$$\dfrac{\textbf{(1)} \qquad \textbf{(2)} \qquad \dfrac{\{l_1, l_2\} \subseteq \{l_1, l_2, l_3\} \qquad l_1 \Rightarrow Bool = Bool \qquad \dfrac{}{l_2 \Rightarrow Int <: Float}\text{ S-IntFloat}}{\{l_1 : Bool,\ l_2 : Int,\ l_3 : Float\} <: \{l_1 : Bool,\ l_2 : Float\}}\text{ S-RCD}}{\varnothing \mapsto (\lambda r : \{l_1 : Bool,\ l_2 : Float\}.\text{if } r.l_1 \text{ then } r.l_2 \text{ else } 5, 5)\ \{l_1 = \texttt{true},\ l_2 = -8,\ l_3 = 9, 0\} : Float}\text{ T-App}$$

$$\textbf{(1)} \quad \dfrac{\dfrac{r : \{l_1 : Bool,\ l_2 : Float\} \in \Gamma \qquad 1 \in \{1, 2\}}{\Gamma \mapsto r.l_1 : Bool}\text{ T-Proj} \quad \dfrac{r : \{l_1 : Bool,\ l_2 : Float\} \in \Gamma \qquad 1 \in \{1, 2\}}{\Gamma \mapsto r.l_2 : Float}\text{ T-Proj} \qquad \Gamma \mapsto 5, 5 : Float}{\Gamma \mapsto \text{if } r.l_1 \text{ then } r.l_2 \text{ else } 5, 5 : Float}\text{ T-If}$$

$$\textbf{(2)} \quad \dfrac{\Gamma \mapsto \texttt{true} : Bool \qquad \Gamma \mapsto -8 : Int \qquad \Gamma \mapsto 9, 0 : Float}{\Gamma \mapsto \{l_1 = \texttt{true},\ l_2 = -8,\ l_3 = 9, 0\} : \{l_1 : Bool,\ l_2 : Int,\ l_3 : Float\}}\text{ T-RCD}$$

4.7. Ejercicio 7

Cálculo λ clásico Aplicamos el algoritmo de inferencia a $x\ x$



(1) $\mathbb{W}(x) = \{x : \sigma\} \rhd x : \sigma$
(2) $\mathbb{W}(x) = \{x : \tau\} \rhd x : \tau$
(3) $\mathbb{W}(x\ x) = S\{x : \sigma\} \cup S\{x : \tau\} \rhd S(x\ x) : S\theta$

$$S = MGU(\{\sigma \doteq \tau,\ \sigma \doteq \tau \to \theta\}) \underset{\tau/\sigma}{\overset{4}{\rightsquigarrow}} \{\tau \doteq \tau \to \theta\} \overset{6}{\rightsquigarrow} \texttt{falla}$$

Cálculo λ subtipado :

$$\dfrac{\dfrac{x : \sigma \to \tau \in \Gamma}{\Gamma \rhd x : \sigma \to \tau} \quad \dfrac{x : \theta \in \Gamma}{\Gamma \rhd x : \theta} \quad \theta <: \sigma}{\Gamma \mapsto x\ x : \tau}\ \text{T-App}$$

Entonces debemos unificar $\sigma \to \tau \doteq \theta$ y $\sigma \doteq \theta$ de tal manera que se cumpla $\theta <: \sigma$. Si elegimos que $\sigma = \theta = Top$, entonces nos queda que x es de tipo $Top \to \tau$ y de tipo Top simultaneamente. Y se cumple que $Top <: Top$. Además, como x es una función, puede tomar cualquier cosa que sea subtipo de Top. En particular $Top \to \tau <: Top$, por lo que el jucio de tipado $\{x : Top \to \tau\} \mapsto x\ x : \tau$ es válido.

4.8. Ejercicio 8

a) Si valiese S-Arrow', la expresión $M = succ((\lambda x : Float.x)\ 0{,}5)$ es de tipo Int, pero si evaluamos M, entonces tenemos que $M \to succ(0{,}5)$, lo que no tiene sentido, por que $succ$ solo está definido para enteros.

b) Si la regla fuese covariante tanto en el argumento como en el resultado, entonces en la expresión $M = (\lambda x : Int.succ(x))\ 0{,}5 : Int$, sin embargo si realizamos la aplicación nos queda que $M = succ(0{,}5)$, por lo que tenemos el mismo error que en **a)**.

b)

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{x : Float \in \{x : Float\}}{\{x : Int\} \mapsto x : Int}\text{ T-Var}}{\{x : Int\} \mapsto succ(x) : Int}\text{ T-Succ}}{\varnothing \rhd \lambda x : Int.succ(x) : Int \to Int}\text{ T-Abs} \quad \dfrac{\dfrac{}{Float <: Float}\text{ S-Refl} \quad \dfrac{}{Int <: Float}\text{ S-IntFloat}}{Float \to Float <: Float \to Int}\text{ S-Arrow'}}{\varnothing \rhd \lambda x : Float.x : Float \to Int}\text{ T-Sub} \quad \varnothing \mapsto 0{,}5 : Float \quad \dfrac{}{Float <: Float}\text{ S-Refl}}{\varnothing \mapsto (\lambda x : Float.x)\ 0{,}5) : Int}\text{ T-App}}{\varnothing \mapsto succ((\lambda x : Float.x)\ 0{,}5) : Int}\text{ S-Succ}$$

a)

$$\dfrac{\dfrac{\dfrac{\dfrac{\dfrac{x : Float \in \{x : Float\}}{\{x : Float\} \mapsto x : Float}\text{ T-Var}}{\varnothing \rhd \lambda x : Float.x : Float \to Float}\text{ T-Abs} \quad \dfrac{\dfrac{}{Int <: Float}\text{ S-IntFloat} \quad \dfrac{}{Int <: Float}\text{ S-IntFloat}}{Int \to Int <: Float \to Int}\text{ S-Arrow''}}{\varnothing \rhd \lambda x : Int.succ(x) : Float \to Int}\text{ T-Sub} \quad \varnothing \mapsto 0{,}5 : Float \quad \dfrac{}{Float <: Float}\text{ S-Refl}}{\varnothing \mapsto (\lambda x : Int.succ(x))\ 0{,}5) : Int}\text{ T-App}$$

4.9. Ejercicio 9

a) Si la regla de referencias es covariante, entonces $\tau <: \sigma$, si intentamos subtipar una referencia $M : Ref\ \sigma$ con $Ref\ \tau$, entonces:

$$let\ r = \text{ref } 3\ in\ r := 2{,}1;$$
$$!r$$

En este caso, definimos $r : Ref\ Int$ en el let, por lo que cuando derreferenciemos r esperaremos entero. En la asignación, queremos asignar un $Float$ a r, por lo que esperariamos que r fuese de tipo $Ref\ Float$ y como la regla de subtipado es covariante y $Int <: Float$, entonces la asignación se realiza sin ningún problema.

Ahora, cuando derreferenciemos r obtendremos 2,1 que es un $Float$, no un Int por lo que el programa falla.

b) Si la regla es contravariante, entonces en el programa:

$$let\ r = \text{ref } 2{,}1\ in\ !r$$

Definimos a r como una referencia de $Float$, sin embargo, como r es subtipable a $Ref\ Int$, podemos usar la derreferenciación de enteros para derreferenciarla, lo que provocaría el error en el programa.

4.10. Ejercicio 10

a) El término es tipable. Demostración:

$$\dfrac{\dfrac{\dfrac{c : Comp_{\{x:Int\}} \in \Gamma}{\Gamma \mapsto c : Comp_{\{x:Int\}}}\text{T-Var} \quad \dfrac{(1)}{\Gamma \mapsto \{x = 1,\ y = 2\} : \{x : Int\}} \quad \dfrac{}{\Gamma \mapsto \{x = 0\} : \{x : Int\}}\text{T-Rcd}}{\Gamma = \{c : Comp_{\{x:Int\}}\} \mapsto \text{mejorSegún}(c, \{x = 1,\ y = 2\}, \{x = 0\}) : Bool}\text{T-Comp}}{\varnothing \mapsto \lambda c : Comp_{\{x:Int\}}.\text{mejorSegún}(c, \{x = 1,\ y = 2\}, \{x = 0\}) : Comp_{\{x:Int\}} \to Bool}\text{T-Abs}$$

$$\textbf{(1)} \quad \dfrac{\dfrac{}{\Gamma \mapsto \{x = 1,\ y = 2\} : \{x : Int,\ y : Int\}}\text{T-Rcd} \quad \dfrac{\{x\} \subseteq \{x, y\} \qquad \dfrac{}{Int <: Int}\text{S-Refl}}{\{x : Int,\ y : Int\} <: \{x : Int\}}\text{S-Rcd}}{\Gamma \mapsto \{x = 1,\ y = 2\} : \{x : Int\}}\text{T-Sub}$$

b)

$$\dfrac{\tau <: \sigma}{Comp_\sigma <: Comp_\tau}\text{S-Comp}$$

c)

$$\dfrac{\dfrac{\dfrac{x : Comp_{Nat} \in \Gamma'}{\Gamma' \mapsto x : Comp_{Nat}}\text{T-Var} \qquad \Gamma' \mapsto 3 : Nat \qquad \Gamma' \mapsto 4 : Nat}{\Gamma' = \Gamma, \{x : Comp_{Nat}\} \mapsto \text{mejorSegún}(x, 3, 4)) : Bool}\text{T-Comp}}{\Gamma \mapsto \lambda x : Comp_{Nat}.\text{mejorSegún}(x, 3, 4)) : Comp_{Nat} \to Bool}\text{T-Abs} \qquad \dfrac{c : Comp_{Float} \in \Gamma}{\Gamma \mapsto c : Comp_{Float}}\text{T-Var} \qquad \dfrac{\dfrac{}{Int <: Float}\text{S-IntFloat}}{Comp_{Float} <: Comp_{Int}}\text{S-Comp}$$

$$\dfrac{\Gamma = \{c : Comp_{Float}\} \mapsto (\lambda x : Comp_{Nat}.\text{mejorSegún}(x, 3, 4))\ c : Bool}{\varnothing \mapsto \lambda c : Comp_{Float}.(\lambda x : Comp_{Nat}.\text{mejorSegún}(x, 3, 4))\ c : Comp_{Float} \to Bool}\text{T-Abs}$$

4.11. Ejercicio 11

1)

$$\dfrac{}{< A \times B > \to C <: A \to B \to C}\text{(S-Curry)} \qquad \dfrac{}{A \to B \to C\ <:\ < A \times B > \to C}\text{(S-Uncurry)}$$

2)

$$\dfrac{\dfrac{\dfrac{\{x\} \subseteq \{x, y\} \qquad A <: A'}{\{x : A, y : A'\} <: \{x : A'\}}\text{S-Rcd} \qquad \dfrac{B <: B' \qquad C <: C}{B' \to C <: B \to C}\text{S-Arrow}}{\{x : A'\} \to B' \to C <: \{x : A, y : A'\} \to B \to C}\text{S-Arrow} \qquad \dfrac{}{\{x : A, y : A'\} \to B \to C\ <:\ < \{x : A, y : A'\} \times B > \to C}\text{S-Unurry}}{\{x : A'\} \to (B' \to C)\ <:\ < \{x : A, y : A'\} \times B > \to C}\text{S-Trans}$$

3) Con las nuevas reglas, entonces tenemos que
$M = ((\lambda x :< Nat \times Nat > .\pi_2(x))\ 5)\ 1)$ es una expresión tipable, esto es porque por S-Curry, podemos tipar $\lambda x :< Nat \times Nat > .\pi_2(x)$ como una función de tipo $Nat \to Nat \to Nat$. Sin embargo, cuando vamos a la función, vemos que necesitamos proyectar la segunda coordenada de una tupla, por lo que no tiene sentido pasarle como parámetro un entero a esa función.

4.12. Ejercicio 12

1)

$$\dfrac{\dfrac{\dfrac{}{\varnothing \mapsto Clarabelle : Vaca}\,\text{T-Clara} \quad \dfrac{}{Vaca <: Animal}\,\text{S-Vaca}}{\varnothing \mapsto Clarabelle : Animal}\,\text{T-Sub} \quad \dfrac{\dfrac{}{\varnothing \mapsto Clarabelle : Vaca}\,\text{T-Clara} \quad \overset{\textbf{(a)}}{\dfrac{}{Vaca <: AlimentoPara(Animal)}}}{\varnothing \mapsto Clarabelle : AlimentoPara(Animal)}\,\text{T-Subs}}{\varnothing \mapsto \text{comer}(Clarabelle, Clarabelle) : Animal}\,\text{T-Comer}$$

$$\textbf{(a)} \quad \dfrac{\dfrac{}{Vaca <: AlimentoPara(Leon)}\,\text{S-VacaLeon} \quad \dfrac{\dfrac{}{Leon <: Animal}\,\text{S-Leon}}{AlimentoPara(Leon) <: AlimentoPara(Animal)}\,\text{S-Alim}}{Vaca <: AlimentoPara(Animal)}\,\text{S-Trans}$$

2.

$$\dfrac{\sigma <: \tau \qquad \tau <: \sigma}{AlimentoPara(\tau) <: AlimentoPara(\tau)}\,\text{S-Align}$$

Esto quiere decir que $AlimentoPara(\sigma)$ es subtipo de $AlimentoPara(\tau)$ si τ y σ son equivalente. Si la regla, la hacemos covariante, entonces tenemos el problema del ejercicio anterior, y si es contravariante, entonces podriamos darle de comer pasto a un león.

4.13. Ejercicio 13

1)

$$\frac{\sigma <: \tau}{\times_1(\sigma) <: \times_1(\tau)}(\text{S-Proj1}) \qquad \frac{\sigma <: \tau}{\times_2(\sigma) <: \times_2(\tau)}(\text{S-Proj2})$$

$$\frac{}{< \sigma \times \tau > \;\; <: \;\; \times_1(\sigma)}(\text{S-TuplaProj1})$$

$$\frac{}{< \sigma \times \tau > \;\; <: \;\; \times_2(\tau)}(\text{S-TuplaProj2})$$

Todas las reglas son covariantes, siempre que usemos un elemento del tipo proyección podremos remplazar su tipo por uno más especifico sin ningún problema. Con las reglas S-TuplaProj, consideramos a las tuplas como subtipos de las proyecciones, pues las tuplas tienen la operacion para proyectar cada una de sus coordenadas.

Además, no tendría sentido la inversa, ya que si quisieramos remplazar una tupla por una proyección y necesitamos hacer uso de las dos coordenadas de las tuplas, estariamos perdiendo información.

2)

$$\dfrac{\dfrac{\dfrac{\text{(a)}}{\Gamma \mapsto \lambda y : \times_2(Int).\pi_2(y) : \times_2(Int) \to Float} \quad \dfrac{p :< Nat, Nat > \in \Gamma}{\Gamma \mapsto p :< Nat, Nat >}\,\text{T-Var} \quad \dfrac{\dfrac{}{< Nat, Nat ><: \times_2(Nat)}\,\text{S-TuplaProj2} \quad \dfrac{Nat <: Int}{\times_2(Nat) <: \times_2(Int)}\,\text{S-Proj2}}{< Nat, Nat ><: \times_2(Int)}\,\text{S-Trans}}{\Gamma = \{p :< Nat, Nat >\} \mapsto (\lambda y : \times_2(Int).\pi_2(y))\; p : Float}\,\text{T-App}$$

(a)

$$\dfrac{\dfrac{\dfrac{\dfrac{y : \times_2(Int) \in \Gamma'}{\Gamma' \mapsto y : \times_2(Int)}\,\text{T-Var}}{\Gamma' = \Gamma, \{y : \times_2(Int)\} \mapsto \pi_2(y) : Int}\,\text{T-Proj1}}{\Gamma \mapsto \lambda y : \times_2(Int).\pi_2(y) : \times_2(Int) \to Int}\,\text{T-Abs} \quad \dfrac{\dfrac{}{\times_2(Int) <: \times_2(Int)}\,\text{S-Refl} \quad \dfrac{}{Int <: Float}\,\text{S-IntFloat}}{\times_2(Int) \to Int <: \times_2(Int) \to Float}\,\text{S-Arrow}}{\Gamma \mapsto \lambda y : \times_2(Int).\pi_2(y) : \times_2(Int) \to Float}\,\text{T-Subs}$$

4.14. Ejercicio 14

a)

$$\frac{\sigma <: \tau}{\texttt{det}(\sigma) <: \texttt{det}(\tau)}(\text{S-Det}) \qquad \frac{\sigma <: \tau}{\sigma <: det(\tau)}(\text{S-Det1})$$

Las reglas son covariantes, porque si queremos detener un Int, entonces con el detener de tipo entero deberemos poder detener $Bool$s y Nats. Además, si tenemos un $\texttt{det}(Nat)$ entonces querremos querer poder tratarlo como a un Nat, en muchos casos, por lo que tenemos la segunda regla.

b)

$$\dfrac{\dfrac{\textbf{(1)}}{\varnothing \mapsto \lambda x : det(Nat).\text{if } True \text{ then } x \text{ else } 0 : \texttt{det}(Nat) \to \texttt{det}(Nat)} \quad \dfrac{\dfrac{}{Nat <: \texttt{det}(Nat)}\text{S-Det1} \quad \dfrac{}{\texttt{det}(Nat) <: \texttt{det}(Int)}\text{S-Det}}{\texttt{det}(Nat) \to \texttt{det}(Nat) <: Nat \to \texttt{det}(Int)}\text{S-Arrow}}{\varnothing \mapsto \lambda x : det(Nat).\text{if } True \text{ then } x \text{ else } 0 : Nat \to \texttt{det}(Int)}\text{T-sub}$$

$$\textbf{(1)} \quad \dfrac{\dfrac{}{\Gamma \mapsto True : Bool}\text{T-True} \quad \dfrac{x : \texttt{det}(Nat) \in \Gamma}{\Gamma \mapsto x : \texttt{det}(Nat)}\text{T-Var} \quad \dfrac{\Gamma \mapsto 0 : Nat \qquad Nat <: \texttt{det}(Nat)}{\Gamma \mapsto 0 : det(Nat)}\text{T-Sub}}{\Gamma = \{x : det(Nat)\} \mapsto \text{if } True \text{ then } x \text{ else } 0 : \texttt{det}(Int)}$$