Subtipado

Por qué subtipado

- El sistema de tipos que estudiamos descarta programas incorrectos.
- Pero también programas “buenos”.

Por qué subtipado

- El sistema de tipos que estudiamos descarta programas incorrectos.
- Pero también programas "buenos".
  - $(\lambda x : \{a : Nat\}.x.a)\{a = 1, b = true\}$
- Queremos mayor flexibilidad y disminuir la cantidad de programas buenos que se descartan.

Principio de sustitución (o "sustitutividad")

$$\sigma <: \tau$$

Principio de sustitución (o "sustitutividad")

$$\sigma <: \tau$$

- **Principio de sustitución** (Liskov)
  Las propiedades que se pueden predicar sobre los elementos de un tipo también se pueden predicar sobre los elementos de sus subtipos.
  $$(\forall x : \tau.\ P(x)) \rightarrow (\forall x : \sigma.\ P(x))$$

Principio de sustitución (o "sustitutividad")

$$\sigma <: \tau$$

- **Principio de sustitución** (Liskov)
  Las propiedades que se pueden predicar sobre los elementos de un tipo también se pueden predicar sobre los elementos de sus subtipos.
  $$(\forall x : \tau.\ P(x)) \to (\forall x : \sigma.\ P(x))$$

- **Ejemplo informal**

  Elefante $<:$ Mamífero $\quad P(x) =$ "x tiene sistema nervioso central"

Principio de sustitución (o "sustitutividad")

$$\sigma <: \tau$$

- **Interpretación operacional**
  "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:

$$\frac{\Gamma \triangleright M : \sigma \quad \sigma <: \tau}{\Gamma \triangleright M : \tau} \text{(T-Subs)}$$

Subtipado de tipos base

- Para los tipos base asumimos que nos informan de qué manera están relacionados; por ejemplo

$$\begin{array}{rcl} Nat & <: & Int \\ Int & <: & Float \\ Bool & <: & Nat \end{array}$$

Subtipado como preorden

Subtipado como preorden

$$\frac{}{\sigma <: \sigma}\ \text{(S-Refl)} \qquad \frac{\sigma <: \tau \quad \tau <: \rho}{\sigma <: \rho}\ \text{(S-Trans)}$$

Nota:

- Sin antisimetría, ni simetría

Tipado para LC con registros – Repaso

$$\frac{x : \sigma \in \Gamma}{\Gamma \triangleright x : \sigma}\,(\text{T-Var})$$

$$\frac{\Gamma, x : \sigma \triangleright M : \tau}{\Gamma \triangleright \lambda x : \sigma.M : \sigma \to \tau}\,(\text{T-Abs}) \quad \frac{\Gamma \triangleright M : \sigma \to \tau \quad \Gamma \triangleright N : \sigma}{\Gamma \triangleright M\,N : \tau}\,(\text{T-App})$$

Tipado para LC con registros – Repaso

$$\frac{x : \sigma \in \Gamma}{\Gamma \rhd x : \sigma}\ \text{(T-Var)}$$

$$\frac{\Gamma, x : \sigma \rhd M : \tau}{\Gamma \rhd \lambda x : \sigma.M : \sigma \to \tau}\ \text{(T-Abs)} \quad \frac{\Gamma \rhd M : \sigma \to \tau \quad \Gamma \rhd N : \sigma}{\Gamma \rhd M\, N : \tau}\ \text{(T-App)}$$

$$\frac{\Gamma \rhd M_i : \sigma_i \quad \forall i \in I = \{1..n\}}{\Gamma \rhd \{l_i = M_i\}_{i \in I} : \{l_i : \sigma_i\}_{i \in I}}\ \text{(T-Rcd)}$$

$$\frac{\Gamma \rhd M : \{l_i : \sigma_i\ ^{i \in 1..n}\} \quad j \in 1..n}{\Gamma \rhd M.l_j : \sigma_j}\ \text{(T-Proj)}$$

Tipado para LC con registros – Repaso

$$\frac{x : \sigma \in \Gamma}{\Gamma \rhd x : \sigma} \text{(T-Var)}$$

$$\frac{\Gamma, x : \sigma \rhd M : \tau}{\Gamma \rhd \lambda x : \sigma . M : \sigma \to \tau} \text{(T-Abs)} \quad \frac{\Gamma \rhd M : \sigma \to \tau \quad \Gamma \rhd N : \sigma}{\Gamma \rhd M\, N : \tau} \text{(T-App)}$$

$$\frac{\Gamma \rhd M_i : \sigma_i \quad \forall i \in I = \{1..n\}}{\Gamma \rhd \{l_i = M_i\}_{i \in I} : \{l_i : \sigma_i\}_{i \in I}} \text{(T-Rcd)}$$

$$\frac{\Gamma \rhd M : \{l_i : \sigma_i\ ^{i \in 1..n}\} \quad j \in 1..n}{\Gamma \rhd M.l_j : \sigma_j} \text{(T-Proj)}$$

$$\frac{\Gamma \rhd M : \sigma \quad \sigma <: \tau}{\Gamma \rhd M : \tau} \text{(T-Subs)}$$

Subtipado de registros a lo “ancho”

Subtipado de registros a lo “ancho”

`{nombre: String, edad: Int}` `{nombre: String}`

Subtipado de registros a lo "ancho"

`{nombre: String, edad: Int} <: {nombre: String}`

La regla general es

$$\frac{}{\{l_i : \sigma_i | i \in 1..n+k\} <: \{l_i : \sigma_i | i \in 1..n\}} \text{(S-RcdWidth)}$$

Subtipado de registros a lo “ancho”

`{nombre: String, edad: Int} <: {nombre: String}`

La regla general es

$$\frac{}{\{l_i : \sigma_i | i \in 1..n+k\} <: \{l_i : \sigma_i | i \in 1..n\}} \text{(S-RcdWidth)}$$

Nota:

- $\sigma <: \{\}$, para todo tipo registro σ

Subtipado de registros a lo “ancho”

`{nombre: String, edad: Int} <: {nombre: String}`

La regla general es

$$\frac{}{\{l_i : \sigma_i | i \in 1..n+k\} <: \{l_i : \sigma_i | i \in 1..n\}} \text{(S-RcdWidth)}$$

Nota:

- $\sigma <: \{\}$, para todo tipo registro σ
- ¿hay algún tipo registro τ tal que $\tau <: \sigma$, para todo tipo registro σ?

Subtipado de registros en “profundidad”

`{a: Nat, b: Int}` `{a: Float, b: Int}`

Subtipado de registros en "profundidad"

```
{a: Nat, b: Int}  <:  {a: Float, b: Int}
```

La regla general es

$$\frac{\sigma_i <: \tau_i \quad i \in I = \{1..n\}}{\{l_i : \sigma_i\}_{i \in I} <: \{l_i : \tau_i\}_{i \in I}} \text{(S-RcdDepth)}$$

Ejemplos

$$\{x : \{a : Nat, b : Nat\}, y : \{m : Nat\}\} \quad <: \quad \{x : \{a : Nat\}, y : \{\}\}$$

Ejemplos

$$\{x : \{a : Nat, b : Nat\}, y : \{m : Nat\}\} \quad <: \quad \{x : \{a : Nat\}, y : \{\}\}$$

$$\dfrac{\dfrac{}{\{a : Nat, b : Nat\} <: \{a : Nat\}}\,(\text{S-RcdWidth}) \qquad \dfrac{}{\{m : Nat\} <: \{\}}\,(\text{S-RcdWidth})}{\{x : \{a : Nat, b : Nat\}, y : \{m : Nat\}\} <: \{x : \{a : Nat\}, y : \{\}\}}\,(\text{S-RcdDepth})$$

Permutaciones de campos

- Los registros son descripciones de campos y no deberían depender del orden dado

Permutaciones de campos

- Los registros son descripciones de campos y no deberían depender del orden dado

$$\frac{\{k_j : \sigma_j | j \in 1..n\} \text{ es permutación de } \{l_i : \tau_i | i \in 1..n\}}{\{k_j : \sigma_j | j \in 1..n\} <: \{l_i : \tau_i | i \in 1..n\}} \text{ (S-RcdPerm)}$$

Permutaciones de campos

- Los registros son descripciones de campos y no deberían depender del orden dado

$$\frac{\{k_j : \sigma_j | j \in 1..n\} \text{ es permutación de } \{l_i : \tau_i | i \in 1..n\}}{\{k_j : \sigma_j | j \in 1..n\} <: \{l_i : \tau_i | i \in 1..n\}} \text{(S-RcdPerm)}$$

Nota:

- (S-RcdPerm) puede usarse en combinación con (S-RcdWidth) y (S-Trans) para eliminar campos en cualquier parte del registro

Combinando width, depth y permutation subtyping

$$\frac{\{l_i |\ i \in 1..n\} \subseteq \{k_j |\ j \in 1..m\} \qquad k_j = l_i \Rightarrow \sigma_j <: \tau_i}{\{k_j : \sigma_j |\ j \in 1..m\} <: \{l_i : \tau_i |\ i \in 1..n\}} \text{ (S-Rcd)}$$

Subtipado de tipos función

Subtipado de tipos función

$$\frac{\sigma' <: \sigma \quad \tau <: \tau'}{\sigma \to \tau <: \sigma' \to \tau'} \text{(S-Func)}$$

- Observar que el sentido de $<:$ 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

$$\frac{\sigma' <: \sigma \quad \tau <: \tau'}{\sigma \to \tau <: \sigma' \to \tau'} \text{(S-Func)}$$

Subtipado de tipos función

$$\frac{\sigma' <: \sigma \quad \tau <: \tau'}{\sigma \to \tau <: \sigma' \to \tau'} \text{(S-Func)}$$

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

Subtipado de referencias ("punteros")

¿Covariante? Imaginemos esta regla:

$$\frac{\sigma <: \tau}{\mathit{Ref}\ \sigma <: \mathit{Ref}\ \tau}$$

¿Qué ocurre?

Ref no es covariante

```
p := new Nat(1);
*p := 0.5;
succ(*p)
```

Ref no es covariante

```
p := new Nat(1);  // p : Ref Nat
*p := 0.5;
succ(*p)
```

$$\frac{\sigma <: \tau}{\mathit{Ref}\,\sigma <: \mathit{Ref}\,\tau} \qquad \frac{\mathit{Nat} <: \mathit{Float}}{\mathit{Ref}\ \ \mathit{Nat} <: \mathit{Ref}\ \ \mathit{Float}}$$

Ref no es covariante

```
p := new Nat(1);  // p : Ref Nat
*p := 0.5;        // usando Ref Nat <: Ref Float
succ(*p)          // Nat
```

$$\frac{\sigma <: \tau}{\mathit{Ref}\,\sigma <: \mathit{Ref}\,\tau} \qquad \frac{\mathit{Nat} <: \mathit{Float}}{\mathit{Ref}\ \mathit{Nat} <: \mathit{Ref}\ \mathit{Float}}$$

Ref no es covariante

```
p := new Nat(1);   // p : Ref Nat
*p := 0.5;         // usando Ref Nat <: Ref Float
succ(*p)           // Nat
```

¡Pero 0.5 no es un natural!

$$\frac{\sigma <: \tau}{\mathit{Ref}\,\sigma <: \mathit{Ref}\,\tau} \qquad \frac{\mathit{Nat} <: \mathit{Float}}{\mathit{Ref}\ \mathit{Nat} <: \mathit{Ref}\ \mathit{Float}}$$

¿Ref contravariante?

¿Contravariante? Imaginemos esta regla:

$$\frac{\sigma <: \tau}{\mathit{Ref}\,\tau <: \mathit{Ref}\,\sigma}$$

Otra vez, ¿qué ocurre?

Ref no es contravariante

```
p := new Float(0.5);
succ(*p)
```

$$\frac{\sigma <: \tau}{\mathit{Ref}\,\tau <: \mathit{Ref}\,\sigma} \qquad \frac{\mathit{Nat} <: \mathit{Float}}{\mathit{Ref}\ \ \mathit{Float} <: \mathit{Ref}\ \ \mathit{Nat}}$$

Ref no es contravariante

```
p := new Float(0.5);  // p : Ref Float
succ(*p)
```

$$\frac{\sigma <: \tau}{\mathit{Ref}\, \tau <: \mathit{Ref}\, \sigma} \qquad \frac{\mathit{Nat} <: \mathit{Float}}{\mathit{Ref}\ \ \mathit{Float} <: \mathit{Ref}\ \ \mathit{Nat}}$$

Ref no es contravariante

```
p := new Float(0.5);  // p : Ref Float
succ(*p)              // Nat
```

¡Pero 0.5 no es un natural!

$$\frac{\sigma <: \tau}{\mathit{Ref}\,\tau <: \mathit{Ref}\,\sigma} \qquad \frac{\mathit{Nat} <: \mathit{Float}}{\mathit{Ref}\ \ \mathit{Float} <: \mathit{Ref}\ \ \mathit{Nat}}$$

Ref es invariante

$$\frac{\sigma <: \tau \quad \tau <: \sigma}{\mathit{Ref}\,\sigma <: \mathit{Ref}\,\tau}$$

"Sólo se comparan referencias de tipos equivalentes."

Refinando el tipo Ref

Además de $Ref\,\sigma$ (referencias de lecto-escritura), se pueden identificar los tipos:

- $Source\ \sigma$ (referencias de sólo lectura)
- $Sink\ \sigma$ (referencias de sólo escritura)

$$\frac{\ldots \rhd M : Source\ \sigma}{\ldots \rhd *M : \sigma} \qquad \frac{\ldots \rhd M : Sink\ \sigma \quad \ldots \rhd N : \sigma}{\ldots \rhd *M := N : ()}$$

Refinando el tipo Ref

Además de $Ref\,\sigma$ (referencias de lecto-escritura), se pueden identificar los tipos:

- $\mathit{Source}\ \sigma$ (referencias de sólo lectura)
- $\mathit{Sink}\ \sigma$ (referencias de sólo escritura)

$$\frac{\ldots \triangleright M : \mathit{Source}\ \sigma}{\ldots \triangleright *M : \sigma} \qquad \frac{\ldots \triangleright M : \mathit{Sink}\ \sigma \quad \ldots \triangleright N : \sigma}{\ldots \triangleright *M := N : ()}$$

Source es covariante	Sink es contravariante
$\dfrac{\sigma <: \tau}{\mathit{Source}\,\sigma <: \mathit{Source}\,\tau}$	$\dfrac{\sigma <: \tau}{\mathit{Sink}\,\tau <: \mathit{Sink}\,\sigma}$

Refinando el tipo Ref

Además de $Ref\,\sigma$ (referencias de lecto-escritura), se pueden identificar los tipos:

- $Source\ \sigma$ (referencias de sólo lectura)
- $Sink\ \sigma$ (referencias de sólo escritura)

$$\frac{\ldots \rhd M : Source\ \sigma}{\ldots \rhd *M : \sigma} \qquad \frac{\ldots \rhd M : Sink\ \sigma \quad \ldots \rhd N : \sigma}{\ldots \rhd *M := N : ()}$$

Ref es subtipo de Source	Ref es subtipo de Sink
$\dfrac{}{Ref\ \sigma <: Source\,\sigma}$	$\dfrac{}{Ref\ \sigma <: Sink\,\sigma}$

Reglas de tipado como especificación de un algoritmo

- Las reglas de tipado sin subtipado son dirigidas por sintaxis.
- Ello hace que sea inmediato implementar un algoritmo de chequeo de tipos a partir de ellas.

$$\frac{x : \sigma \in \Gamma}{\Gamma \rhd x : \sigma} \text{(T-Var)}$$

$$\frac{\Gamma, x : \sigma \rhd M : \tau}{\Gamma \rhd \lambda x : \sigma.M : \sigma \to \tau} \text{(T-Abs)} \qquad \frac{\Gamma \rhd M : \sigma \to \tau \quad \Gamma \rhd N : \sigma}{\Gamma \rhd M\, N : \tau} \text{(T-App)}$$

$$\frac{\Gamma \rhd M_i : \sigma_i \quad \forall i \in I = \{1..n\}}{\Gamma \rhd \{l_i = M_i\}_{i \in I} : \{l_i : \sigma_i\}_{i \in I}} \text{(T-Rcd)}$$

$$\frac{\Gamma \rhd M : \{l_i : \sigma_i\ ^{i \in 1..n}\} \quad j \in 1..n}{\Gamma \rhd M.l_j : \sigma_j} \text{(T-Proj)}$$

Agregando subsumption

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

$$\frac{x : \sigma \in \Gamma}{\Gamma \rhd x : \sigma}\,(\text{T-Var}) \qquad \frac{\Gamma \rhd M : \sigma \quad \sigma <: \tau}{\Gamma \rhd M : \tau}\,(\text{T-Subs})$$

$$\frac{\Gamma, x : \sigma \rhd M : \tau}{\Gamma \rhd \lambda x : \sigma . M : \sigma \to \tau}\,(\text{T-Abs}) \quad \frac{\Gamma \rhd M : \sigma \to \tau \quad \Gamma \rhd N : \sigma}{\Gamma \rhd M\,N : \tau}\,(\text{T-App})$$

$$\frac{\Gamma \rhd M_i : \sigma_i \quad \forall i \in I = \{1..n\}}{\Gamma \rhd \{l_i = M_i\}_{i \in I} : \{l_i : \sigma_i\}_{i \in I}}\,(\text{T-Rcd})$$

$$\frac{\Gamma \rhd M : \{l_i : \sigma_i\ {}^{i \in 1..n}\} \quad j \in 1..n}{\Gamma \rhd M.l_j : \sigma_j}\,(\text{T-Proj})$$

“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

$$\frac{x : \sigma \in \Gamma}{\Gamma \mapsto x : \sigma}\,(\text{T-Var}) \qquad \frac{\Gamma, x : \sigma \mapsto M : \tau}{\Gamma \mapsto \lambda x : \sigma.M : \sigma \to \tau}\,(\text{T-Abs})$$

$$\frac{\Gamma \mapsto M : \sigma \to \tau \quad \Gamma \mapsto N : \rho \quad \rho <: \sigma}{\Gamma \mapsto M\,N : \tau}\,(\text{T-App})$$

$$\frac{\Gamma \mapsto M_i : \sigma_i \quad \forall i \in I = \{1..n\}}{\Gamma \mapsto \{l_i = M_i\}_{i \in I} : \{l_i : \sigma_i\}_{i \in I}}\,(\text{T-Rcd})$$

$$\frac{\Gamma \mapsto M : \{l_i : \sigma_i\ ^{i \in 1..n}\} \quad j \in 1..n}{\Gamma \mapsto M.l_j : \sigma_j}\,(\text{T-Proj})$$

Variante dirigida por sintaxis

- ¿Qué relación tiene con la formulación original?

Proposición:

1. $\Gamma \mapsto M : \sigma$ implica que $\Gamma \rhd M : \sigma$
2. $\Gamma \rhd M : \sigma$ implica que existe τ tal que $\Gamma \mapsto M : \tau$ con $\tau <: \sigma$

Hacia una implementación de chequeo de tipos

- Lo único que faltaría cubrir es de qué manera se implementa la relación $\sigma <: \tau$

$$\frac{x : \sigma \in \Gamma}{\Gamma \mapsto x : \sigma}\,(\text{T-Var}) \qquad \frac{\Gamma, x : \sigma \mapsto M : \tau}{\Gamma \mapsto \lambda x : \sigma . M : \sigma \to \tau}\,(\text{T-Abs})$$

$$\frac{\Gamma \mapsto M : \sigma \to \tau \quad \Gamma \mapsto N : \rho \quad \rho <: \sigma}{\Gamma \mapsto M\, N : \tau}\,(\text{T-App})$$

$$\frac{\Gamma \mapsto M_i : \sigma_i \quad \forall i \in I = \{1..n\}}{\Gamma \mapsto \{l_i = M_i\}_{i \in I} : \{l_i : \sigma_i\}_{i \in I}}\,(\text{T-Rcd})$$

$$\frac{\Gamma \mapsto M : \{l_i : \sigma_i \ ^{i \in 1..n}\} \quad j \in 1..n}{\Gamma \mapsto M.l_j : \sigma_j}\,(\text{T-Proj})$$

Reglas de subtipado – Recordatorio

$$\frac{}{Nat <: Int}\ \text{(S-NatInt)} \qquad \frac{}{Int <: Float}\ \text{(S-IntFloat)} \qquad \frac{}{Bool <: Nat}\ \text{(S-BoolNat)}$$

$$\frac{}{\sigma <: \sigma}\ \text{(S-Refl)} \qquad \frac{\sigma <: \tau \quad \tau <: \rho}{\sigma <: \rho}\ \text{(S-Trans)}$$

$$\frac{\sigma' <: \sigma \quad \tau <: \tau'}{\sigma \to \tau <: \sigma' \to \tau'}\ \text{(S-Func)}$$

$$\frac{\{l_i |\ i \in 1..n\} \subseteq \{k_j |\ j \in 1..m\} \qquad k_j = l_i \Rightarrow \sigma_j <: \tau_i}{\{k_j : \sigma_j |\ j \in 1..m\} <: \{l_i : \tau_i |\ i \in 1..n\}}\ \text{(S-Rcd)}$$

Reglas de subtipado – Recordatorio

$$\frac{}{Nat<:Int}\text{ (S-NatInt)} \qquad \frac{}{Int<:Float}\text{ (S-IntFloat)} \qquad \frac{}{Bool<:Nat}\text{ (S-BoolNat)}$$

$$\frac{}{\sigma<:\sigma}\text{ (S-Refl)} \qquad \frac{\sigma<:\tau \quad \tau<:\rho}{\sigma<:\rho}\text{ (S-Trans)}$$

$$\frac{\sigma'<:\sigma \quad \tau<:\tau'}{\sigma \to \tau<:\sigma' \to \tau'}\text{ (S-Func)}$$

$$\frac{\{l_i |\, i \in 1..n\} \subseteq \{k_j |\, j \in 1..m\} \qquad k_j = l_i \Rightarrow \sigma_j<:\tau_i}{\{k_j : \sigma_j |\, j \in 1..m\}<:\{l_i : \tau_i |\, i \in 1..n\}}\text{ (S-Rcd)}$$

- ¿Son dirigidas por sintáxis?

Reglas de subtipado – Recordatorio

$$\frac{}{Nat{<:}Int}\ \text{(S-NatInt)} \qquad \frac{}{Int{<:}Float}\ \text{(S-IntFloat)} \qquad \frac{}{Bool{<:}Nat}\ \text{(S-BoolNat)}$$

$$\frac{}{\sigma{<:}\sigma}\ \text{(S-Refl)} \qquad \frac{\sigma{<:}\tau \quad \tau{<:}\rho}{\sigma{<:}\rho}\ \text{(S-Trans)}$$

$$\frac{\sigma'{<:}\sigma \quad \tau{<:}\tau'}{\sigma \to \tau{<:}\sigma' \to \tau'}\ \text{(S-Func)}$$

$$\frac{\{l_i \mid i \in 1..n\} \subseteq \{k_j \mid j \in 1..m\} \qquad k_j = l_i \Rightarrow \sigma_j{<:}\tau_i}{\{k_j : \sigma_j \mid j \in 1..m\}{<:}\{l_i : \tau_i \mid i \in 1..n\}}\ \text{(S-Rcd)}$$

- ¿Son dirigidas por sintáxis? No.
- El problema está en (S-Refl) y (S-Trans)

Deshaciéndonos de (S-Refl) y (S-Trans)

- Se puede probar que $\sigma <: \sigma$ se puede derivar siempre que se tenga reflexividad para los tipos escalares:
  - $\mathit{Nat} <: \mathit{Nat}$
  - $\mathit{Bool} <: \mathit{Bool}$
  - $\mathit{Int} <: \mathit{Int}$
  - $\mathit{Float} <: \mathit{Float}$
- Agregamos estos cuatro axiomas y no consideramos explícitamente a la regla (S-Refl).

Deshaciéndonos de (S-Trans)

- Si consideramos el cálculo sin *Bool*, *Nat*, *Int* ni *Float*, se puede probar la transitividad.
- Es decir, no hace falta tenerla como una regla explícita.
- Si se desea agregar estos tipos, se pueden escribir los axiomas correspondientes.

El algoritmo de chequeo de subtipos (sin los axiomas de Nat, Bool, Int, Float)

```
subtype(S, T) =
     if S==S1 → S2 and T==T1 → T2
        then subtype(T1, S1) and subtype(S2, T2)
        else
         if S=={kj : Sj, j ∈ 1..m} and T=={li : Ti, i ∈ 1..n}
            then {li, i ∈ 1..n} ⊆ {kj, j ∈ 1..m} and
               ∀ i ∃ j kj = li and subtype(Sj, Ti)
            else false
```