

# Type inference su linguaggio base + `Nat<:Bool`

Simone Ballarin, approfondimento AALP 2018/19  
matricola 1207245

# Linguaggio Base senza subtyping

## ALGORITMO DI MITCHELL WAND

Semplice algoritmo guidato dalla forma dei termini

Risaliva l'albero di derivazione con i Lemmi di Inversione, fino ad ottenere solo assiomi

Otteneva dei vincoli sul tipo dell'espressione

Risolve il sistema di vincoli con un algoritmo di unificazione

# Subtyping con subsumption

Le derivazioni non sono più guidate dai termini, non si può risalire l'albero come prima. Questo vale sia per i giudizi di tipo, sia per i giudizi di subtyping.

1. T-Subsumption (si può applicare ad ogni termine sempre)
2. T-Trans (si possono usare per qualsiasi coppia di tipi S,T. Inoltre bisogna indovinare la variabile U.)
3. T-Refl (regola non strettamente necessaria, a patto di inserire NatNat e BoolBool)

Un algoritmo “bottom up” non sa’ se applicare la regola guidata dalla sintassi o queste regole sempre applicabile.

$$\frac{\Gamma \vdash t : S \quad S <: T}{\Gamma \vdash t : T}$$

$$\frac{S <: U \quad U <: T}{S <: T}$$

# Sottotipo algoritmico

Basato unicamente sulla struttura del tipo, quindi senza usare S-Trans e S-Refl, avendo gli assiomi `Nat<:Bool` , `Nat<:Nat` e `Bool <:Bool` in aggiunta.

```
data Type =  
    TBool | TNat | TArrow Type Type | Top  
deriving (Show, Eq)
```

## Lemma 1: $S \leq S$ senza S-Refl

induzione strutturale su  $S$

## Lemma 2: $S \leq T$ derivabile implica $S \leq T$ derivabile senza S-Trans

induzione strutturale su  $S$

Questi due lemmi ci dicono che le due regole non guidate dalla sintassi non sono necessarie se si aggiungono NatNat e BoolBool (che sono guidate dalla sintassi).

## Lemma 3: $S \leq T$ derivabile sse $S \leq T$ derivabile alg

Questo lemma combina i due precedenti formalizzando la correttezza e completezza del typing algoritmico

## Correttezza codice

Lemma 4:  $S \leq T$  derivabile algebricamente sse  $(\leq) S T$  ha valore True.

Lemma 5: Se  $S \leq T$  derivabile algebricamente allora  $(\leq) S T = \text{True}$  altrimenti  $(\leq) S T = \text{False}$

# Typing algoritmico

Stessa problema dell'algoritmo di sottotipaggio, siccome è presente la regola T-Sub allora l'algoritmo non risulta più guidato dalla struttura del termine.

Gli unici momenti in cui T-Sub è essenziale sono:

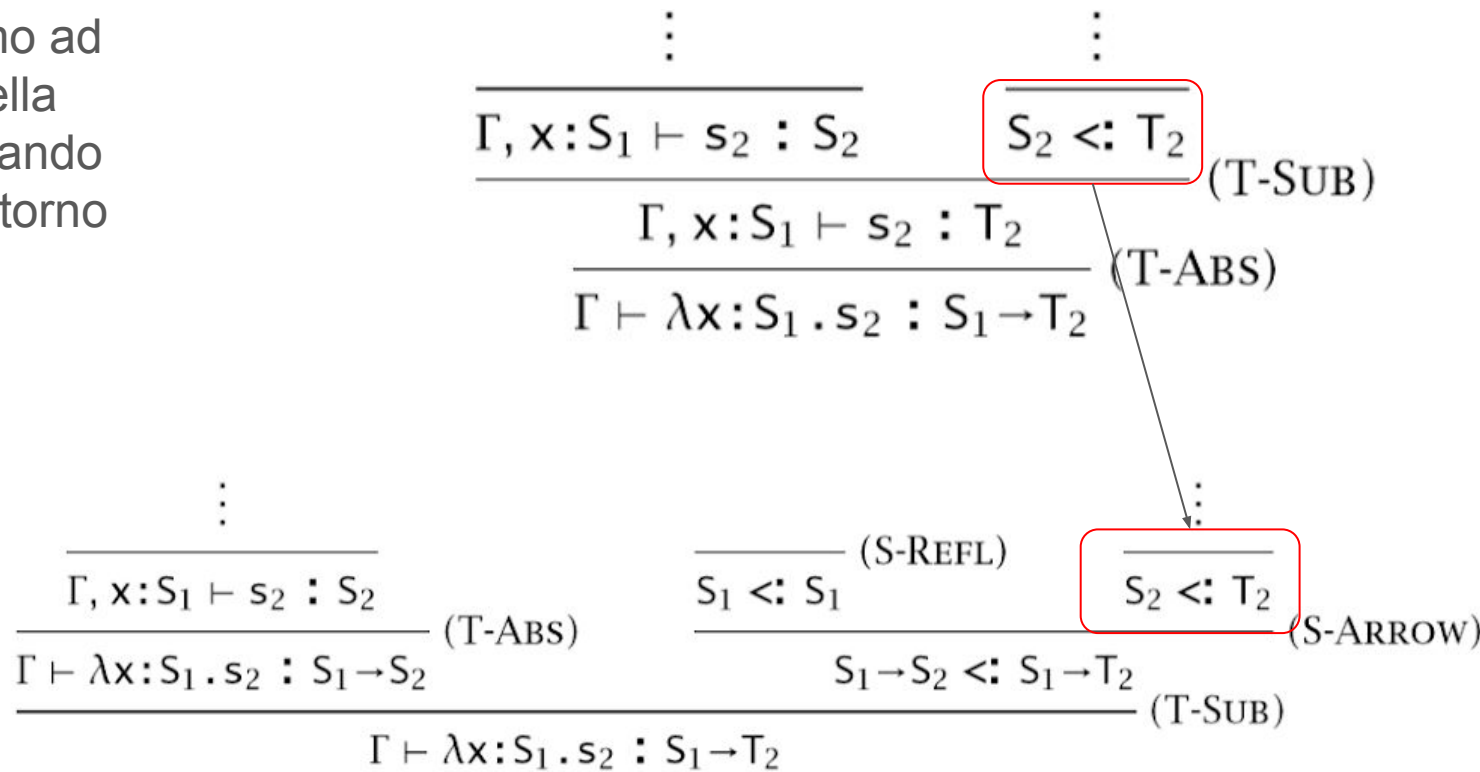
1. quando dobbiamo tipare il parametro di una funzione con un suo sottotipo in un'applicazione. *(fn x:Bool . x) 3*
2. quando dobbiamo tipare la guardia di *if 3 then 4 else 7*

In tutti gli altri casi T-Sub può essere portato più verso la radice e quindi si può fare in modo che l'albero finisca con T-Sub.

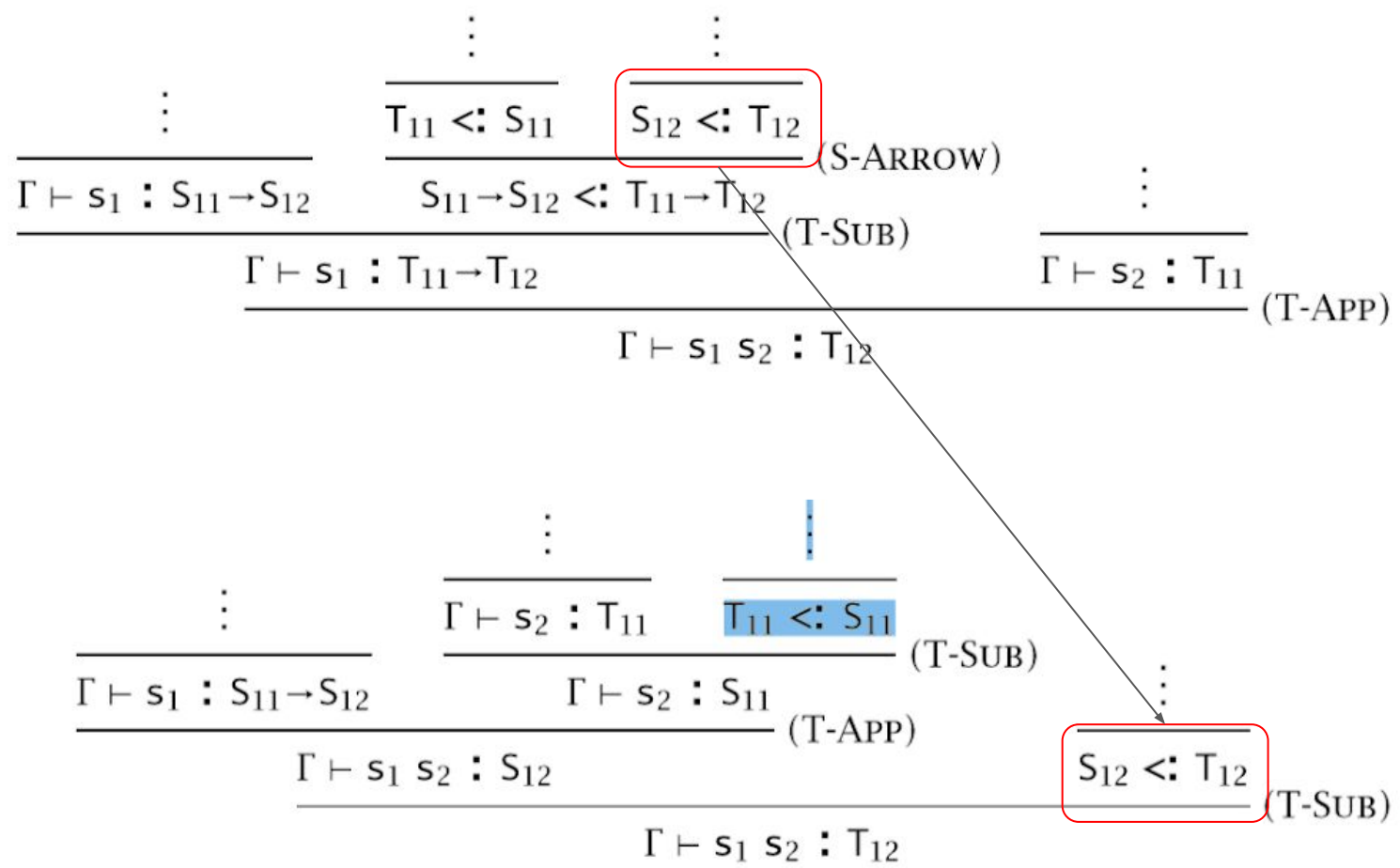


# Esempio 1:

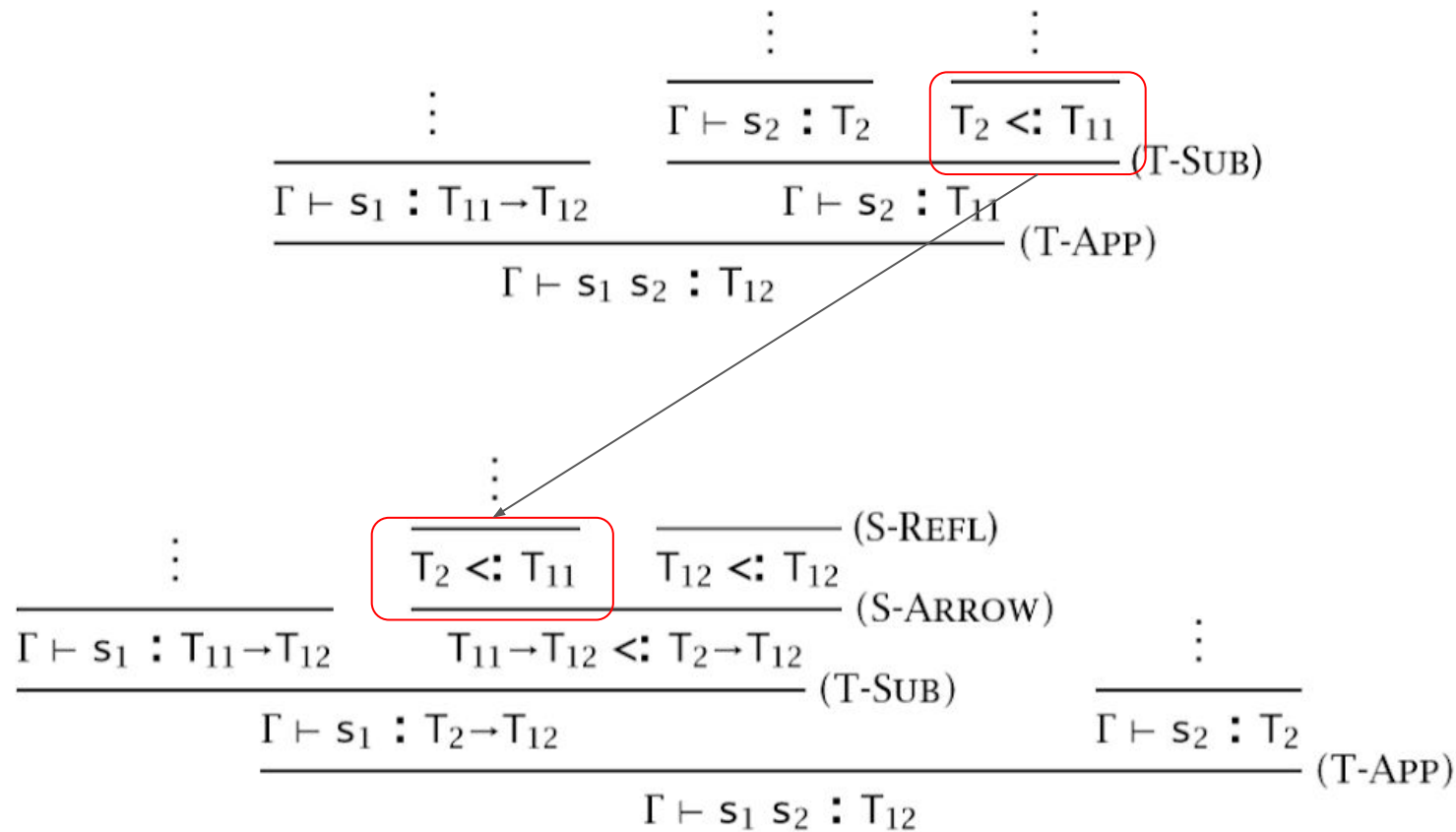
da adattare il termine  
 ritornato passiamo ad  
 adattare il tipo della  
 funzione specificando  
 meglio il tipo di ritorno



# Esempio 2



# Esempio 3



# Esempio 4:

collassamento tramite  
S-Trans

$$\frac{\frac{\frac{\vdots}{\Gamma \vdash s : S} \quad \frac{\frac{\vdots}{S <: U}}{\Gamma \vdash s : U}}{\Gamma \vdash s : T} \text{ (T-SUB)} \quad \frac{\frac{\vdots}{U <: T}}{\Gamma \vdash s : T} \text{ (T-SUB)}$$

$$\frac{\frac{\frac{\vdots}{\Gamma \vdash s : S} \quad \frac{\frac{\frac{\vdots}{S <: U} \quad \frac{\vdots}{U <: T}}{S <: T} \text{ (S-TRANS)}}{\Gamma \vdash s : T} \text{ (T-SUB)}$$

# Typing Algoritmico

Attuando queste modifiche ripetutamente possiamo ottenere degli alberi in cui TSub e' usato solo in tre posti:

1. nell'albero sinistro di TApp (Necessaria, per adattare il parametro);
2. nella tipaggio della guardia di un If then else;
3. o come ultima regola dell'albero (eliminare questa non e' un problema in quanto vuol dire che possiamo tipare solo col tipo più specifico).

# Typing Algoritmico

Si rimuove T-Sub

Si aggiunge T-AppAlg

La nuova regola TAppAlg ingloba TApp  
preceduta da TSub del parametro

$$\frac{\Gamma \vdash t_1 : T_{11} \rightarrow T_{12} \quad \Gamma \vdash t_2 : T_2 \quad T_2 <: T_{11}}{\Gamma \vdash t_1 t_2 : T_{12}}$$

Si deve aggiungere anche T-IFAlg

La nuova regola TIfAlg ingloba TIf  
preceduta da TSub sulla guardia

$$\frac{\Gamma \vdash t_1 : T_1 \quad T_1 <: \text{Bool} \quad \Gamma \vdash t_2 : T_2 \quad \Gamma \vdash t_3 : T_3 \quad T_2 \vee T_3 = T}{\Gamma \vdash \text{if } t_1 \text{ then } t_2 \text{ else } t_3 : T}$$

Ora siamo guidati dalla struttura del termine e possiamo tipare solo con il tipo piu' specifico. In sostanza si può scrivere l'algoritmo.

# Soundness typing algoritmico

Se  $r \mapsto t:T$  allora  $r \vdash t:T$

Si procede per induzione sull'altezza di  $r \mapsto t:T$ .

# Completezza typing algoritmico [tipo minimo]

Se  $r \vdash t:T$  allora  $r \mapsto t:S$  per qualche  $S <: T$

Si procede per induzione sull'altezza di  $r \vdash t:T$ .

# Correttezza codice

se  $\text{typeOf } r \ t = \text{Just } T$  allora  $r \mapsto t:T$  [correttezza]

se  $r \mapsto t:T$  allora  $\text{typeOf } r \ t = \text{Just } T$  [completezza]

se  $r \mapsto t:T$  derivabile allora  $\text{typeOf } t = \text{Just } T$  altrimenti  $\text{typeOf } t = \text{Nothing}$