Approfondimento di AALP

type inference sul linguaggio base + subtyping + NatBool

Simone Ballarin
matricola 1207245
anno accademico 2018/19

Subtyping algoritmico

Vigono le stesse regole del subtyping dichiarativo con eccezione di S-Trans e S-Refl, inoltre devono essere aggiunte le regole NatNat e BoolBool.
Le regole quindi presenti sono:

$\dfrac{}{S<:Top}$	S-Top
$\dfrac{}{Nat<:Nat}$	NatNat
$\dfrac{}{Bool<:Bool}$	BoolBool
$\dfrac{}{Nat<:Bool}$	NatBool
$\dfrac{T_1<:S_1 \quad S_2<:T_2}{S_1 \to S_2<:T_1 \to T_2}$	S-Arrow

Lemma 1
ENUNCIATO: S<:S senza S-Refl
DIMOSTRAZIONE:
induzione strutturale su S
(casi base)
S=Top, allora la tesi vale per l'assioma S-Top
S=Nat, allora la tesi vale per l'assioma Nat<:Nat
S=Bool, allora la tesi vale per l'assioma Bool<:Bool
(casi induttivi)
S=S1->S2, per ipotesi induttiva S1<:S1 e S2<:S2, quindi per per S-Arrow, S<:S

□

Lemma 2
ENUNCIATO: S<:T derivabile implica S<:T derivabile senza S-Trans
DIMOSTRAZIONE:
Se S<:T derivabile allora S<:T derivabile senza S-Refl.
procedo ora per induzione sull'albero di S<:T senza S-Refl:
(h=1)
In questi casi la regola S-Trans non può essere usata in quanto non è un assioma, quindi alberi di altezza uno non potranno mai contenere S-Trans
(h->h+1)
procedo in base all'ultima regola usata, se questa è diversa da S-Trans allora la tesi segue per ipotesi induttiva sui sottoalberi premessa.
Ultima regola usata S-Trans ottenuta da S<:U e U<:T di altezza al più h, per qualche U.
Procedo per casi in base all'ultima regola dei due sottoalberi premesse di S-Trans (si ricorda che l'albero non contiene S-Refl):
(NatNat, Any)
Se l'albero sinistro finisce con NatNat allora S=U=Nat, allora siccome U<:T allora T=Nat o T=Bool o T=Top. Nel primo caso la tesi vale per NatNat, nel secondo per NatBool, nel terzo per S-Top.
(BoolBool, Any)
Se l'albero sinistro finisce con BoolBool allora S=U=Bool, allora siccome U<:T allora T=Bool o T=Top. Nel primo caso la tesi vale per BoolBool, nel secondo per S-Top.

(NatBool, Any)
Se l'albero sinistro finisce con NatBool allora S=Nat e U=Bool. Siccome U<:T allora T=Bool o T=Top. Nel primo caso la tesi vale per BoolBool, nel secondo per S-Top.
(Any, NatNat)
Se l'albero destro finisce con NatNat allora U=Nat e T=Nat. Siccome S<:U allora S=Nat. La tesi vale per NatNat.
(Any, BoolBool)
Se l'albero destro finisce con BoolBool allora U=Bool e T=Bool. Siccome S<:U allora S=Nat o S=Bool. La tesi vale nel primo caso per NatBool, nel secondo per BoolBool.
(Any, NatBool)
Se l'albero destro finisce con NatBool allora U=Nat e T=Bool. Siccome S<:U allora S=Nat. La tesi vale per NatBool.
(Any, S-Top)
Se l'albero destro finisce con S-Top allora T=Top, quindi S<:Top viene per S-Top.
(S-Top, Any)
Se l'albero sinistro termina con S-Top allora U=Top. Per ip. ind. abbiamo che l'albero U<:T è derivabile senza S-Trans, quindi siccome non e' presente S-Refl, l'ultima regola usata per U<:T è S-Top (unica regola utilizzabilei). Quindi T=Top e S<:T ottenibile con S-Top.
(S-Arrow, S-Arrow)
Allora S=S1->S2, T=T1->T2, U=U1->U2 e U1<:S1, S2<:U2, T1<:U1 e U2<:T2.
Dalle precedenti posso ottenere T1<:S1 e S2<:T2 usando S-Refl per ip. ind. posso dire che T1<:S1 e S2<:T2 si possono ottenere senza T-Trans. Da queste due poi ottengo con S-Arrow S<:T senza transitività.

□

Lemma 3:
ENUNCIATO: S<:T derivabile sse S<:T derivabile alg
DIMOSTRAZIONE:
[S<:T -> S<:T alg]
procedo per induzione sull'albero S<:T.
(h=1)
Allora l'albero è ottenuto da un assioma. Nel caso di S-Top allora la derivazione è anche algoritmica, nel caso S-Refl basta usare Lemma 1, l'ultimo caso che rimane e NatBool che rimane anche nel typing algoritmico.
(h->h+1)
Procedo in base all'ultima regola usata.
S-Trans
Se è stata usata S-Trans allora segue per Lemma 2.
S-Arrow
Se è stata usata S-Arrow allora per ogni premessa esiste un derivazione algoritmica per ip. induttiva. Unendo questi due alberi con S-Arrow (presente anche nel typing algoritmico) si ottiene S<:T algoritmicamente.
[S<:T alg -> S<:T]
Le regole algoritmiche sono un sottoinsieme delle regole dichiarative ad eccezione delle regole NatNat e BoolBool. Questi due assiomi però possono essere sempre sostituiti da S-Refl per questo segue la tesi.

□

Correttezza e completezza rispetto subtyping algoritmico

Lemma 4:
ENUNCIATO: S<:T derivabile algoritmicamente sse $(<=)\ S\ T$ ha valore $True$.
DIMOSTRAZIONE:
[⇒]
Procedo per induzione sull'altezza:
(h=1)
Allora l'albero è ottenuto da STop, NatNat, NatBool, BoolBool. Per ognuno di questi casi c'è un pattern matching che ritorna $True$.
(h->h+1)
Allora è ottenuto da S-Arrow. Le due premesse sono alberi algoritmici di altezza al più h, quindi per ipotesi induttiva le due chiamate ricorsive fatte nel codice avranno come risultato $True$. Il calcolo evolve a $True\ \&\&\ True$ che quindi evolve a $True$.
[⇐]
Procedo per induzione sulla struttura del giudizio
(casi base)
allora si ricade in uno dei primi quattro casi del pattern matching, ognuno dei quali corrisponde ad un assioma algoritmico (essendo che ora ogni regola è guidata dalla forma del giudizio).
(caso induttivo)
Il quinto (il caso della funzione) si ottiene per ipotesi induttiva, se quest'ultimo caso è $True$ vuol dire che le due chiamate ricorsive ritornano $True$. Quindi per ipotesi induttiva esiste un albero algoritmico per le premesse di S-Arrow, quindi la tesi.

□

Lemma 5:
ENUNCIATO: Se S<:T derivabile algoritmicamente allora $(<=)\ S\ T = True$ altrimenti $(<=)\ S\ T = False$
DIMOSTRAZIONE:
La funzione termina sempre in quanto le chiamate vengono fatte sempre su tipi più piccoli. Per il lemma 4 la computazione termina a $True$ sse esiste una derivazione algoritmica. Se non esiste una derivazione algoritmica allora siccome il codice deve terminare allora essendo che il risultato non è $True$ è per forza di cose $False$.
L'algoritmo è corretto rispetto alla derivazione algoritmica la quale è corretta rispetto alla derivazione dichiarativa.

□

Typing Algoritmico

Vigono le stesse regole del typing dichiarativo con eccezione di T-Sub che deve essere rimossa. Inoltre devono essere modificate le regole TIf e TApp con una loro controparte algoritmica TIFAlg, TAppAlg.
Le regole quindi presenti sono:

1. TSumAlg, uguale a TSum;
2. TMinAlg, uguale a TMin;
3. TNatAlg, uguale a TNat;
4. TVarAlg, uguale a TVar;
5. TTrueAlg, uguale a TTrue;
6. TFalseAlg, uguale a TFalse;
7. TArrowAlg, uguale a TArrow
8. TIfAlg

$$\frac{\Gamma \vdash b:B \quad B<:Bool \quad \Gamma \vdash m:T \quad \Gamma \vdash n:S}{\Gamma \vdash \mathit{if}\ b\ \mathit{then}\ m\ \mathit{else}\ n : lub(T,S)}$$

9. TAppAlg

$$\frac{\Gamma \mapsto t_1:S_{11} \rightarrow S_{12} \quad \Gamma \mapsto t_2:S_2 \quad S_2<:S_{11}}{\Gamma \mapsto t_2:S_{12}}$$

Per ragioni di brevità nelle seguenti dimostrazione si darà per scontato che la conversione tra S<:T e S<:T algoritmico è sempre possibile per ogni S e T (in entrambi i versi). Quindi non si useranno sintassi diverse.

Correttezza Typing algoritmico

ENUNCIATO: Se $\Gamma \mapsto$t:T allora $\Gamma \vdash$t:T
DIMOSTRAZIONE:
procedo per induzione sull'altezza di $\Gamma \mapsto$t:T, sia questa h.
(h=1)
Allora l'ultima regola usata è <u>TVarAlg</u>, <u>TNatAlg</u>, <u>TTrueAlg</u> o <u>TFalseAlg</u>. In tutti questi casi la derivazione è già in forma dichiarativa siccome le regole sono uguali alla loro variante non algoritmica.
(h->h+1)
Procedo per casi sull'ultima regola usata.
<u>TArrowAlg</u>
Se l'ultima regola è TArrowAlg allora $\Gamma \mapsto$fn x:S.t:T ottenuta da Γ,x:S$\mapsto$fn.t:T con altezza al più h.
Per ipotesi induttiva abbiamo Γ,x:S$\vdash$fn.t:T quindi usando TArrow si ottiene $\Gamma \vdash$fn x:S.t:T.
<u>TAppAlg</u>
Se l'ultima regola usata è TAppAlg allora $\Gamma \mapsto t_1\ t_2 : T_{12}$ ottenuta da $\Gamma \mapsto t_1 : T_{11} \rightarrow T_{12}$, $\Gamma \mapsto t_2 : T_2$ e $T_2 <: T_{11}$, tutte ottenute con altezza al più h.
Per ipotesi induttiva sui primi due abbiamo $\Gamma \vdash t_1 : T_{11} \rightarrow T_{12}$, $\Gamma \vdash t_2 : T_2$, ora con TApp e TSub otteniamo la seguente derivazione che dimostra la tesi: $\dfrac{\Gamma \vdash t_1:T_{11} \rightarrow T_{12} \quad \dfrac{\Gamma \vdash t_2:T_2 \quad T_2<:T_{11}}{\Gamma \vdash t_2:T_{11}}}{\Gamma \vdash t_1\ t_2:T_{12}}$.
<u>TSumAlg</u>

Se l'ultima regola usata è TSumAlg allora $\Gamma \mapsto t_1 + t_2 : Nat$, ottenuto da $\Gamma \mapsto t_1 : Nat$ e $\Gamma \mapsto t_2 : Nat$ con altezza al più h. Per ipotesi induttiva allora abbiamo $\Gamma \vdash t_1 : Nat$ e $\Gamma \vdash t_2 : Nat$, da queste due per TSum ottengo la tesi.

<u>TMinAlg</u>

Se l'ultima regola usata è TMinAlg allora la dimostrazione è analoga al caso TSumAlg.

<u>TIFAlg</u>

Se l'ultima regola usata è TIFAlg allora $\Gamma \mapsto if\ b\ then\ m\ else\ n : L$, ottenuto da $\Gamma \mapsto b : B$, $B <: Bool$, $\Gamma \mapsto m : T$, $\Gamma \mapsto n : S$ con altezza al più h, dove $L = lub(T, S)$.

Per ipotesi induttiva abbiamo $\Gamma \vdash b : B$, $\Gamma \vdash m : T$, $\Gamma \vdash n : S$ e per definizione di lub abbiamo $T <: L$ e $S <: L$. Ora usando T-If e due volte T-Subs si può dimostrare la tesi nel seguente modo $\dfrac{\dfrac{B <: Bool}{\Gamma \vdash b : B} \quad \dfrac{\Gamma \vdash m : T \quad T <: L}{\Gamma \vdash m : L} \quad \dfrac{\Gamma \vdash n : S \quad S <: L}{\Gamma \vdash n : L}}{\Gamma \vdash if\ b\ then\ m\ else\ n : L}$.

Conclusione: il caso base e i casi induttivi sono dimostrati quindi per ipotesi induttiva si ha la tesi.

□

Completezza typing algoritmico (tipo minimo S)

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

DIMOSTRAZIONE:

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

(h=1)

Se l'ultima regola usata è <u>TVar</u> , <u>TTrue</u> , <u>TFalse</u> o <u>TNat</u> allora siccome queste regole sono identiche alla loro variante algoritmica l'albero è già in forma algoritmica.

(h->h+1)

Procedo per casi in base all'ultima regola usata.

<u>TArrow</u>

Se l'ultima regola usata è TArrow allora $\Gamma \vdash fn\ x : T_1 .\ t : T_1 \to T_2$ ottenuto da $\Gamma, x : T_1 \vdash t : T_2$ con altezza al più h. Per ipotesi induttiva $\Gamma, x : T_1 \mapsto t : S_2$ per qualche $S_2 <: T_2$, ora da questa segue per TArrowAlg $\dfrac{\Gamma, x : T_1 \mapsto t : S_2}{\Gamma \mapsto fn\ x : T_1 . t : T_1 \to S_2}$, siccome $S_2 <: T_2$ per SFun abbiamo $T_1 \to S_2 <: T_1 \to T_2$ che dimostra la tesi.

<u>TApp</u>

Se l'ultima regola usata è TApp allora $\Gamma \vdash t_1\ t_2 : T_{12}$ ottenuto da $\Gamma \vdash t_1 : T_{11} \to T_{12}$ e $\Gamma \vdash t_2 : T_{11}$ con altezza h o inferiore. Per ipotesi induttiva allora $\Gamma \mapsto t_1 : S_1$ e $\Gamma \mapsto t_2 : S_2$ con $S_1 <: T_{11} \to T_{12}$ e $S_2 <: T_{11}$. Per inversione abbiamo $S_1 = S_{11} \to S_{12}$ con $T_{11} <: S_{11}$ e $S_{12} <: T_{12}$. Per transitività abbiamo $S_2 <: S_{11}$.

Ora per TAppAlg abbiamo $\dfrac{\Gamma \mapsto t_1 : S_{11} \to S_{12} \quad \Gamma \mapsto t_2 : S_2 \quad S_2 <: S_{11}}{\Gamma \mapsto t_2 : S_{12}}$ che dimostra la tesi siccome $S_{12} <: T_{12}$.

<u>TSum</u>

Se l'ultima regola usata è TSum allora $\Gamma \vdash t_1 + t_2 : Nat$ ottenuto da $\Gamma \vdash t_1 : Nat$ e $\Gamma \vdash t_2 : Nat$ entrambi con altezza al più h. Per ipotesi induttiva $\Gamma \mapsto t_1 : L$ e $\Gamma \mapsto t_2 : S$ con $L <: Nat$ e $S <: Nat$, per inversione abbiamo $L = Nat$ e $T = Nat$. Dalle due derivazioni algoritmiche si ottiene per TSumAlg $\Gamma \mapsto t_1 + t_2 : Nat$, siccome $Nat <: Nat$ per la regola algoritmica NatNat la tesi è dimostrata.

TMin
Se l'ultima regola usata è TMin allora la dimostrazione è analoga a TSum.
TIF
Se l'ultima regola usata è TIF allora $\Gamma \vdash if\ b\ then\ m\ else\ n : T$ ottenuto da $\Gamma \vdash b : Bool$, $\Gamma \vdash m : Nat$ e $\Gamma \vdash n : Nat$, tutte di altezza al più h. Per ipotesi induttiva abbiamo $\Gamma \mapsto b : A$, $\Gamma \mapsto m : B$ e $\Gamma \mapsto n : C$ con $A <: Bool,\ B <: T\ e\ C <: T$, per inversione su questi abbiamo $A = Bool\ o\ A = Nat$.
Procediamo per casi in base al valore di A:

1. se $A = Nat$ per NatBool e TIfAlg abbiamo $\Gamma \mapsto if\ b\ then\ m\ else\ n : lub(B, C)$;
2. se $A = Bool$ per BoolBool e TIfAlg abbiamo $\Gamma \mapsto if\ b\ then\ m\ else\ n : lub(B, C)$.

Siccome T è maggiorante di B e C allora il lub esiste. Per definizione di lub possiamo dire $lub(B, C) <: T$. Questo dimostra la tesi.
TSub
Se l'ultima regola usata è TSub allora $\Gamma \vdash t : T$ ottenuta da $\Gamma \vdash t : U$ e $U <: T$ entrambe con altezza al più h. Per ipotesi induttiva $\Gamma \mapsto t : U'$ con $U' <: U$, per transitività abbiamo $U' <: T$ queste due cose dimostrano la tesi.
Conclusione: il caso base e i casi induttivi sono dimostrati quindi per ipotesi induttiva si ha la tesi.

□

Correttezza algoritmo di typing rispetto a typing algoritmico

ENUNCIATO: se $typeOf\ \Gamma\ t\ = Just\ T$ allora $\Gamma \mapsto t : T$
DIMOSTRAZIONE:
Procedo per induzione strutturale sul termine t.
(casi base)
$typeOf\ \ \Gamma\ (EVar\ x)\ =\ Just\ T$
In questo caso si deve avere $lookup\ x\ \Gamma\ = Just\ T$, questo vuol dire che $(x : T) \in \Gamma$, sapendo questo tramite la regola TVarAlg possiamo ottenere la tesi.
$typeOf\ \ \Gamma\ (ENum\ \ n)\ =\ Just\ Nat$
il costruttore $Enum$ rappresenta un valore naturale del linguaggio, da questo per TNatAlg deriva la tesi.
$typeOf\ \ \Gamma\ (EBool\ \ b)\ =\ Just\ Bool$
il costruttore $EBool$ rappresenta un valore booleano del linguaggio, quindi $b = True\ o\ b = False$ da questo per TTrueAlg o per TFalseAlf deriva la tesi.
(casi induttivi)
$typeOf\ \ \Gamma\ (Efn\ x\ S\ term)\ =\ Just\ S \rightarrow T$
nel codice viene fatta la seguente chiamata $typeOf\ ((x, S) : \Gamma)\ term\ =\ Just\ T$, per ipotesi induttiva allora $\Gamma, (x : S) \mapsto term : T$, da questa per TArrowAlg deriva la tesi.
$typeOf\ \ \Gamma\ (EAp\ m1\ m2)\ =\ Just\ T_{12}$
nel codice vengono fatte le seguenti chiamate :

1. $typeOf\ \Gamma m1\ =\ Just\ T_1$
2. $typeOf\ \Gamma m1\ =\ Just\ T_2$

per ipotesi induttiva quindi si ha $\Gamma \mapsto m1 : T_1$ e $\Gamma \mapsto m2 : T_2$. Siccome sappiamo che il risultato finale è $Just\ T_{12}$, possiamo dire che la guardia booleana dell'if è vera quindi $T_2 <: T_{11}$ e , il pattern matching sulla forma di T_1 è ricaduto nel secondo caso $T_1 = T_{11} \to T_{12}$, da quando appena detto applicando TAppAlg deriva la tesi.

$typeOf\ \ \Gamma\ (ECond\ b\ m1\ m2)\ =\ Just\ lub(T_1, T_2)$

le tre chiamate ricorsive presenti nel codice mi permettono di avere per ipotesi induttiva i seguenti giudizi algoritmici $\Gamma \mapsto m1 : T_1$, $\Gamma \mapsto m2 : T_2$ e $\Gamma \mapsto b : B$. Inoltre siccome il risultato finale sappiamo essere $Just\ lub(T_1, T_2)$ possiamo dire che la guardia dell'if è vera quindi $B <: Bool$, il codice prosegue nel $case\ of$ intermedio, e quindi $lub(T_1, T_2)$ esiste.

Ora usando TIfAlg possiamo deriva la tesi.

$typeOf\ \ \Gamma\ (ESum\ \ m1\ m2)\ =\ Just\ Nat$

le due chiamate ricorsive presenti nel codice mi permettono di avere per ipotesi induttiva i seguenti giudizi algoritmici $\Gamma \mapsto m1 : T_1$, $\Gamma \mapsto m2 : T_2$, essendo che il risultato finale è $Just\ Nat$, posso affermare che il controllo $T_1 == Nat\ \&\&\ T_2 == Nat$ da esito positivo. Applicando TSumAlg ai due giudizi ottengo la tesi.

$typeOf\ \ \Gamma\ (EMin\ \ m1\ m2)\ =\ Just\ Nat$

analogo a $typeOf\ \ \Gamma\ (ESum\ \ m1\ m2)\ =\ Just\ Nat$.

□

Completezza algoritmo di typing rispetto a typing algoritmico

ENUNCIATO: se $\Gamma \mapsto t : T$ allora $typeOf\ \ \Gamma\ t\ = Just\ T$

DIMOSTRAZIONE:

Procedo per induzione sull'altezza della derivazione $\Gamma \mapsto t : T$, sia questa h.

(h=1)

Procedo per casi in base alla regola usata.

<u>TVarAlg</u>

allora $\Gamma \mapsto t : T$ e $(t : T) \in \Gamma$, il codice esegue il lookup di t dalla struttura che rappresenta il contesto. Siccome la variabile è nel contesto allora il lookup avrà successo e l'algoritmo ritorna il tipo letto dal contesto.

<u>TNatAlg</u>

$\Gamma \mapsto n : Nat$, l'algoritmo grazie al pattern matching riconosce il costruttore $ENum$ e ritorna $Just\ Nat$

<u>TBoolAlg</u>

$\Gamma \mapsto b : Bool$, l'algoritmo grazie al pattern matching riconosce il costruttore $EBool$ e ritorna $Just\ Bool$

(h->h+1)

<u>TFunAlg</u>

allora $\Gamma \mapsto fn\ x : S.t : T$ ottenuta da $\Gamma, x : S \mapsto fn.t : T$ con altezza al più h. Per ipotesi induttiva la chiamata $typeOf\ \ ((x : S) : \Gamma)\ t\ = Just\ T$, siccome la chiamata da un risultato allora siamo nel secondo caso del $case\ of$ e infatti viene ritornato $Just\ (TArrow\ S\ T)$.

<u>TAppAlg</u>

allora $\Gamma \mapsto t_1\ t_2 : T_{12}$ ottenuta da $\Gamma \mapsto t_1 : T_{11} \to T_{12}$, $\Gamma \mapsto t_2 : T_2$ e $T_2 <: T_{11}$ con altezza al più h, per ipotesi induttiva abbiamo $typeOf\ \ \Gamma\ t_1\ = Just\ (TArrow\ T_{11}\ T_{12})$ e $typeOf\ \Gamma\ t_2\ =\ Just\ T_2$ inoltre per la correttezza dell'algoritmo di subtyping abbiamo $T_2 <= T_{11} = True$. Dato quanto appena detto il codice esegue il secondo caso del primo $case\ of$,e poi il secondo di quello più annidato. Nel $case\ of$ più annidato la guardia avrà esito positivo, quindi verrà ritornato $Just\ T_{12}$. Questo dimostra la tesi.

<u>TIFAlg</u>

allora $\Gamma \mapsto if\ b\ then\ m\ else\ n : L$, ottenuto da $\Gamma \mapsto b : Bool$, $\Gamma \mapsto m : T$, $\Gamma \mapsto n : S$ con altezza al più' h, dove $L = lub(T, S)$. Per ipotesi induttiva $typeOf\ \ \Gamma\ b\ = Just\ Bool$, $typeOf\ \ \Gamma\ m\ = Just\ T$, $typeOf\ \ \Gamma\ n\ = Just\ S$, per quanto appena detto la computazione procede fino al secondo caso del $case\ of$, dentro questo viene calcolato il lub siccome questo esiste ed è L allora viene trovato e ritornato $Just\ L$.

<u>TSumAlg</u>

allora $\Gamma \mapsto t_1 + t_2 : Nat$, ottenuto da $\Gamma \mapsto t_1 : Nat$ e $\Gamma \mapsto t_2 : Nat$. Per ipotesi induttiva $typeOf\ \ \Gamma\ t_1\ = Just\ Nat$ e $typeOf\ \ \Gamma\ t_2\ = Just\ Nat$. Per quanto appena detto siamo nel secondo caso del $case\ of$ e quindi ritorniamo $Just\ Nat$.

<u>TMinAlg</u>

analogo.

□

ENUNCIATO : $\Gamma \mapsto t : T$ derivabile allora $typeOf\ \ \Gamma\ t\ = Just\ T$ altrimenti $typeOf\ \ \Gamma\ t\ = Nothing$

DIMOSTRAZIONE:

Siccome l'algoritmo effettua sempre chiamate ricorsive con termini più piccoli allora termina certamente.

Da completezza e correttezza abbiamo $\Gamma \mapsto t : T$ sse $typeOf\ \Gamma\ t\ = Just\ T$. Siccome $\forall T . \Gamma \mapsto t : T$ non derivabile, allora $\forall T\ typeOf\ \Gamma\ t\ \neq Just\ T$. Dato che l'algoritmo termina certamente allora non potendo essere un valore $Just$ deve essere $typeOf\ \Gamma\ t\ =\ Nothing$.

□