

Preverjanje tipa

false: bool true: bool 42: int

10: int | if false then true else 42

ni pravila,
izraz nima tipa

10 + (if false then true else 42)

0 + (if false then true else 42)

0 + 42 → 42

kljub temu, da izraz nima tipa, se še vedno lahko evaluiira...

0 * (if true then true else 42)

0 * true

...ali pa ne (množenje z 0 po semantiki MiniML deluje samo za števila)

f: int → int

f 0 se zavičkla

0 * (f 0)

Ker pravilo za množenje zahteva, da se vedno evaluirata oba faktorja, zgornji izraz divergira.

Ob primerni spremembi semantike bi lahko dosegli, da se desni faktor ne evaluiira in se tako izraz evaluiira v 0.

Preverjanje tipa se izide, izraz ima tip int

$$\begin{array}{c}
 \frac{\frac{\frac{\Gamma(m)=int}{\Gamma(n:int) \quad \Gamma(1:int)}}{\Gamma(m)=int} \quad \frac{\frac{\frac{\frac{\Gamma(f)=int \rightarrow int}{\Gamma(m:int) \quad \Gamma(1:int)}}{\Gamma(f)=int \rightarrow int} \quad \frac{\Gamma(m-1):int}{\Gamma(m-1):int}}{\Gamma(m)=int \quad \Gamma(f:int \rightarrow int) \quad \Gamma(m-1):int}}{\Gamma(m:int) \quad \Gamma(f(m-1):int)}}{\Gamma(m < 1:bool) \quad \Gamma(1:int) \quad \Gamma(m * f(m-1):int)}
 \end{array}$$

$$\frac{\Gamma(m:int, f:int \rightarrow int) \text{ if } m < 1 \text{ then } 1 \text{ else } m * f(m-1):int}{\Gamma(m:int, f(int):int \text{ is if } m < 1 \text{ then } 1 \text{ else } m * f(m-1):int} \quad \Gamma(3:int)$$

$$\frac{\Gamma(m:int, f(int):int \text{ is if } m < 1 \text{ then } 1 \text{ else } m * f(m-1)) \quad \Gamma(3:int)}{\Gamma(m:int, f(int):int \text{ is if } m < 1 \text{ then } 1 \text{ else } m * f(m-1)) \quad \Gamma(3:int)}$$

Izvajanje po semantiki malih korakov

(fun f(n: int): int is if n < 1 then 1 else n * f(n-1)) 2

if 2 < 1 then 1 else 2 * (fun f(n: int): int is if n < 1 then 1 else n * f(n-1)) (2-1)

if false then 1 else 2 * (fun f(n: int): int is if n < 1 then 1 else n * f(n-1)) (2-1)

2 * (fun f(n: int): int is if n < 1 then 1 else n * f(n-1)) (2-1)

2 * (fun f(n: int): int is if n < 1 then 1 else n * f(n-1)) 1

2 * (if 1 < 1 then 1 else 1 * (fun f...)) (1-1)

2 * (if false ...)

2 * (1 * (fun f...)) (1-1)

2 * (1 * (fun f...)) 0

2 * (1 * if 0 < 1 then 1 else 0 * (fun f...)) (0-1)

2 * (1 * if true then 1 else ...)

2 * (1 * 1) → 2 * 1 → 2