Path Types in Tree Proofs

by Sven Nilsen, 2018

In this paper I argue that path types should share sub-nodes with other nodes in tree proofs.

A tree proof is a normalized type structure (no variables) where every term has a parent:

type	: U	`U` is the universe of types
bool	: type	Declaration of boolean
false	: bool	..
true	: bool	..
type → type	: type	
bool → bool	: type → type	
false → false	: bool → bool (ff)	Atomic functions
false → true	: bool → bool (ft)	..
true → false	: bool → bool (tf)	..
true → true	: bool → bool (tt)	..
[bool → bool]	: [type → type]	
[ff, tf]	: [bool → bool] (false$_1$)	Function bodies
[ft, tf]	: [bool → bool] (not)	..
[ff, tt]	: [bool → bool] (id)	..
[ft, tt]	: [bool → bool] (true$_1$)	..
false$_1$	: bool → bool	Named functions
not	: bool → bool	..
id	: bool → bool	..
true$_1$	: bool → bool	..
bool → bool → bool	: type → type → type	
false → false$_1$	: bool → bool → bool	Atomic functions
false → id	: bool → bool → bool	..
true → id	: bool → bool → bool	..
[bool → bool → bool]	: type → type → type	
[false → false$_1$, true → id]	: [bool → bool → bool] (and)	Function bodies
[false → id, true → true$_1$]	: [bool → bool → bool] (or)	..
and	: bool → bool → bool	Named functions
or	: bool → bool → bool	..

Now, consider `not(false)`, which returns a boolean:

not(false)	: bool

However, since it evaluates to `true`, it also belongs under the node `[id] true`:

[type → type] type	: type
[bool → bool] bool	: [type → type] type
[id] true	: [bool → bool] bool
not(false)	: [id] true