The C-variadic type `...` has been nested inside another type.

Erroneous code example:

```
#![feature(c_variadic)]

fn foo2(x: u8, y: &...) {} // error!
```

Only foreign functions can use the C-variadic type (`...`). In such functions, `...` may only occur non-nested. That is, `y: &'a ...` is not allowed.

A C-variadic type is used to give an undefined number of parameters to a given function (like `printf` in C). The equivalent in Rust would be to use macros directly (like `println!` for example).