INTRO. TO LOGIC & FUNCT. PROG.

In this assignment, you will write a type-checker for a simple functional language based on the lambda-calculus for simply-typed Propositional Natural Deduction proofs. You need to write a Prolog predicate *hastype(Gamma, E, T)*, where

Gamma is a list of variable-type pairs, representing type assumptions on variables

E is an object language expression, where E ranges over *v(X)* | *****X.E* | **(***E1 E2***)** | **<***E1***,** *E2***>** | **proj** *^(2)_i E* | **inl***(E)* | **inr***(E)* | **case** *Eo* **of inl***(X) => E1* || **inr***(Y) => E2*

and

T is a type ranging over TypeVar(A) | T1 -> T2 | T1 * T2 | T1 + T2

-

You need to provide enough test examples to show your type checker works correctly.

--

Note that this checker can work as a type inference engine. However it does not work for polymorphic type inference. Show with counter-examples that this is the case.

--

For extra credit (50% extra)

-- extend the pairing and projection operation for arbitrary tuples. Hint: use lists and use check for projection operations being of the right kind.

-- generalise from just the two given constructors, to work with arbitrary sums that can have user-defined constructors and take tuples of arguments are arguments.

Skip NavigationSkip Administration