Challenge

Spare Time Teaching

July 14, 2015

Introduction

Interpreters are cool. When the language we are interpreting can be reduced to a normal form, we sometimes call the interpreter a *normalizer*.

Most people have written normalizers, usually a negational normalizer for a boolean language, or maybe even an evaluator for Simply Typed Lambda Calculus.

As with our compilers (see Block challenge) we expect our interpreters to be fast, and correct.

Problem

You have to implement, in OCaml, a one-pass conjunctional normalizer. That is: a function with type `bool_exp -> norm_conj` using at most n recursive calls.

```
type bool_exp =
  | True
  | False
  | Var of string
  | Conj of bool_exp * bool_exp
  | Disj of bool_exp * bool_exp

type norm_disj =
  | NFalse
  | NDisj of string * norm_disj

type norm_conj =
  | NTrue
  | NConj of norm_disj * norm_conj
```