Lista zadań nr 10

Poniższe zadania rozwiąż w języku Plait.

Zadanie 1. (2 pkt)

Zmodyfikuj język wyrażeń arytmetycznych i jego interpreter z pliku `arith.rkt`, tak by operatory arytmetyczne mogły przyjmować dowolną, w tym zerową, liczbę argumentów.

Zadanie 2. (1 pkt)

Rozszerz język wyrażeń logicznych z pliku `bool.rkt` o operatory koniunkcji, alternatywy, negacji i o unarny minus. Dostosuj odpowiednio interpreter.

Zadanie 3. (2 pkt)

W pliku `arith-am.rkt` znajduje się definicja maszyny abstrakcyjnej dla wyrażeń arytmetycznych. Zaimplementuj podobną maszynę abstrakcyjną dla języka wyrażeń z pliku `bool.rkt`, z którego usuwamy wyrażenia **`cond`**, ale dodajemy stałe `#t` i `#f`. Otrzymana maszyna powinna być ekstensjonalnie zgodna (na wspólnym podzbiorze wyrażeń) z interpreterem wyrażeń logicznych z pliku `bool.rkt`.

Zadanie 4. (2 pkt)

Interpreter wyrażeń z pliku `bool.rkt` przeplata ewaluację z odcukrzaniem wyrażeń **`cond`**. Zaimplementuj odcukrzanie jako oddzielną fazę w łańcuchu wykonania programu, która następuje po parsowaniu a przed ewaluacją. Zastanów się czy nie warto wprowadzić dodatkowego języka, który nie będzie zawierał wyrażeń **`cond`**.

Zadanie 5. (3 pkt)

Rozszerz język i interpreter wyrażeń z pliku `bool.rkt` o listy. W szczególności, język powinien zawierać operacje `cons`, `car`, `cdr`, `null?`, `null` i `list`.

Zadanie 6. (2 pkt)

W pliku `arith-vm.rkt` znajduje się definicja maszyny wirtualnej i kompilatora dla wyrażeń arytmetycznych. Zaimplementuj funkcję `decompile`, która dla danego ciągu instrukcji

(typu `Code`) zwróci wyrażenie (typu `Exp`) takie, że `(decompile(compile` e`))` $\equiv e$ dla dowolnego wyrażenia e. Podejrzenie zachodzenia takiego twierdzenia wzmocnij odpowiednim zestawem testów.