MP22 @ II UWr 25 maja 2022 r.

Lista zadań nr 11

Poniższe zadania rozwiąż w języku Plait.

Zadanie 1. (2 pkt)

Używając mechanizmu makr, a w szczególności kostrukcji `define-syntax`, zdefiniuj w języku Plait własne *wieloargumentowe* operacje koniunkcji `my-and` i alternatywy `my-or` w postaci form specjalnych, z leniwą semantyką, tzn. argumenty ewaluujemy od lewej do prawej i tylko tyle ile trzeba by wyznaczyć wartość danej operacji logicznej. Nie używaj form specjalnych **`and`** i **`or`**.

Zdefiniuj następnie formy specjalne `my-let` i `my-let*` odpowiadające konstrukcjom **`let`** i **`let*`** z Racketa.

Zadanie 2. (1 pkt)

Zmodyfikuj ewaluator z pliku `error-ans-monad-macros.rkt` tak, by błędy `''type error''` były zgłaszane przez zdefiniowaną przez nas funkcję `err`, a nie przez funkcję `error`, pochodzącą z Plaita. Jak zmienią się typy funkcji pomocniczych związanych z operatorami arytmetycznymi i logicznymi?

Dodatkowo, zadbaj o to by ewaluator zgłaszał błąd (przy pomocy funkcji `err`) w razie wystąpienia próby dzielenia przez 0.

Zadanie 3. (1 pkt)

Zaimplementuj funkcję `free-vars`, która wyznacza zbiór zmiennych wolnych w wyrażeniu z języka z pliku `let-subst.rkt`.

Zadanie 4. (2 pkt)

Zaimplementuj operację podstawienia dla języka z pliku `let-subst.rkt`, ale tak by była poprawna dla wszystkich podstawianych wyrażeń, a nie tylko tych zamkniętych (nie dopuść do przechwycenia zmiennych).

W swoim rozwiązaniu możesz potrzebować mechanizmu generowania nowych (świeżych) nazw zmiennych. Jedną z możliwości jest napisanie imperatywnego generatora nazw, który będzie trzymał lokalny, modyfikowalny stan licznika. Inną możliwością jest wyposażenie operacji podstawienia w dodatkowy argument – stan licznika.

Zadanie 5. (2 pkt)

Zmodyfikuj język i jego interpreter z pliku `let.rkt`, tak by **`let`** pozwalał na wiązanie wielu zmiennych jednocześnie, tak jak jest to w języku Racket. Rozszerz następnie swoje rozwiązanie o konstrukcję **`let*`**, znaną z języka Racket.

Zadanie 6. (1 pkt)

Wyjaśnij działanie translatora z wyrażeń z nazwanymi zmiennymi do wyrażeń z adresowaniem leksykalnym oraz ewaluatora tych drugich (plik `let-lex-addr.rkt`), na przykładzie wyrażenia

```
{let x 3
  {let y 4
    {+ {let x {+ y 5}
         {* x y}}
       x}}}
```

Zadanie 7. (3 pkt)

Dla języków z pliku `let-lex-addr.rkt` napisz funkcję tłumaczącą wyrażenia z adresami leksykalnymi (indeksami de Bruijna) na tradycyjną reprezentację z nazwanymi zmiennymi. Również w tym zadaniu będziesz potrzebować mechanizmu generowania świeżych nazw zmiennych.