Lista zadań nr 11

Poniższe zadania rozwiąż w języku Plait.

Zadanie 1. (2 pkt)

Dodaj do języka z pliku `error-ans-monad-macros.rkt`, konstrukcję {`try` e_1 e_2}, której semantyka polega na próbie obliczenia wartości wyrażenia e_1, a w razie wystąpienia w tym obliczeniu (dowolnego) błędu, na obliczeniu wartości wyrażenia e_2. Jeżeli e_1 bezbłędnie wylicza się do wartości, to ta wartość staje się wartością wyrażenia `try`.

Zadanie 2. (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 3. (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 4. (2 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.

Zadanie 5. (1 pkt)

Zaimplementuj funkcję `fv`, która wyznacza zbiór zmiennych wolnych w wyrażeniu z języka z pliku `lambda.rkt`.

Zadanie 6. (3 pkt)

Zmodyfikuj język z pliku `lambda.rkt`, tak by funkcje mogły przyjmować więcej niż jeden argument. Wykonaj to ćwiczenie w dwóch wariantach:

1. zmieniając wyłącznie składnię konkretną języka, ale nie modyfikując ani składni abstrakcyjnej ani ewaluatora – parser powinien wykonać całą pracę generując odpowiednio zagnieżdżone funkcje i aplikacje jednoargumentowe;
2. modyfikując składnię konkretną i abstrakcyjną, a następnie dostosowując ewaluator – składnia abstrakcyjna powinna zawierać funkcje i aplikacje wieloargumentowe.