Zadania na pracownię nr 8

Uwaga: W tym tygodniu aż dwa zadania! Termin oddania rozwiązań to poniedziałek, 23 kwietnia 2018, godz 6:00.

Zadanie A: Ewaluator wyrażeń arytmetycznych

Dzięki zeszłotygodniowym ćwiczeniom, nasz interpretowany język stał się już dość bogaty. Ma on wiele struktur danych: liczby, wartości boolowskie, pary, listy. Na ostatnim wykładzie zyskał także rekurencjyne procedury. Na SKOS-ie dołączony jest kod, w którym interpreter rozbudowany jest także o symbole. Składnia abstrakcyjna jest dana przez racketowe cytowanie, a symbole można porównać formą specjalna eq?. Skoro nasz język jest tak bogaty, to napiszmy w nim jakiś bardziej zaawansowany program!

Zadanie A to napisanie ewaluatora wyrażeń arytmetycznych **w interpretowanym języku** z wykładu. Składnia abstrakcyjna tych wyrażeń powinna być dana analogicznie do składni abstrakcyjnej wyrażeń arytmetycznych, które ewaluowaliśmy w Rackecie: stałe jako liczby, a operatory binarne jako trzyelementowe listy zawierające symbol reprezentujący operację i dwa podwyrażenia.

Uwaga: Napisanie ewalatora nie powinno być trudne, w końcu widzieliśmy podobny kawałek kodu (ale w innym języku) na wykładzie. Trudnością w tym zadaniu jest zorientowanie się, co jest bytem w którym języku.

Kolejna uwaga: Nasz język jest nadal dość ubogi. Jeśli będzie wygodniej najpierw rozbudować go o jakąś formę, która sprawi, że będzie łatwiej napisać ewaluator (albo łatwiej konstruować wyrażenia arytmetyczne w celu testowania), proszę tak zrobić.

Zadanie B: Leniwe let-wyrażenia

Na wykładzie widzieliśmy, że leniwość pozwala nam na zrobienie kilku sztuczek, np. na zdefiniowanie nieskończonej listy. Leniwość uzyskiwaliśmy dzięki obserwacji, że wyrażenia „pod lambdami" ewaluowane są dopiero, gdy wywołamy procedurę. Zadanie B polega na rozbudowaniu naszego języka o otwarcie leniwą konstrukcję: leniwe `let`-wyrażenia, czyli takie, które definiują pewne

wyrażenie, ale obliczają jego wartość dopiero w miejscu użycia. Tak więc chcemy, by następujący program **nie** kończył się błędem:

```
(lazy-let [x (/ 5 0)] 7)
```

Leniwe let-wyrażenia powinny pozwolić nam napisać silnię w następujący sposób (czy rozumiesz czemu nie możemy użyć zwykłych let-ów?):

```
(lambda-rec (fact n)
  (lazy-let [t 1]
  (lazy-let [f (n * (fact (- n 1)))]
  (if (= n 0) t f))))
```

Wskazówka: są trzy naturalne sposoby rozwiązania tego zadania. (Żaden nie jest wybitnie lepszy od pozostałych):

- Sposób pierwszy to rozszerzyć typ środowisk tak, by mogły przechowywać nie tylko wartości, ale całe wyrażenia. Wartość takich wyrażeń obliczana jest dopiero wtedy, gdy wyszukujemy w środowisku zmienną, do której takie wyrażenie jest przypisane. Ale uwaga na zmienne wolne w definiowanym wyrażeniu! Trzeba zadbać, by były wiązane statycznie. Dla przykładu, rozważ poniższy program:

  ```
  (let [x 4]
  (lazy-let [y (+ x 1)]
  (let [x 10]
    y)))
  ```

  Jego wartością powinno być 5. Wniosek z tego taki, że rozszerzone środowisko powinno przechowywać nie tylko wyrażenie do obliczenia, ale coś w stylu domknięcia: wyrażenie ze środowiskiem.

- Widzieliśmy na wykładzie, że leniwość można uzyskać zamykając wyrażenie pod bezargumentową lambdą. Sposób drugi to użyć tej obserwacji i przekształcić cały program: leniwe let-y tłumaczymy do zwykłych let-ów ale z wyrażeniem pod lambdą, a miejsca użycia definiowanych zmiennych zmieniamy na bezargumentowe wywołania procedur. Jak zrobić taką transformację? Używając rekurencyjnej procedury ze środowiskiem, mówiącym, które wystąpienia zmiennych stają się wywołaniem procedury.

- Podstawić treść definiowanej formuły za zmienną w reszcie programu. Należy jednak uważać: definiowane wyrażenie może mieć zmienne wolne, więc trzeba bardzo uważać, żeby nie przechwycić takiej wolnej zmiennej pod jakąś konstrukcją wiążącą (czyli: przy podstawieniu trzeba czasem podmienić nazwy związanych zmiennych – tak jak w przypadku zadania na pracownię nr 7.

Przesyłanie rozwiązań

Rozwiązania należy **porządnie** przetestować i zamieścić testy w przesłanym pliku. Rozwiązania powinny być rozszerzeniem pliku `interp-labdarec-symbols.rkt` dostępnego na SKOS-ie pod hasłem „Interpreter języka z rekurencyjnymi lambdami i symbolami".

Rozwiązanie każdego zadania należy przesłać jako plik o nazwie w formacie `nazwisko-imie.rkt`, jak zwykle bez spacji i polskich znaków.