Języki i paradygmaty programowania

Interpreter

Mateusz Perlik

Język LatteFun jest językiem imperatywnym. Jest on wersją języka Latte (lekko zmodyfikowanego w podstawowych aspektach) rozbudowaną o aspekty funkcyjne. Dokładniej, wspiera zagnieżdżone definicje funkcji, lambdy i zmienne o typach funkcyjnych. Ponadto język będzie statycznie typowany.

Doprecyzowanie podpunktów z tabelki:

1. Są trzy podstawowe typy, tj. `int`, `bool` i `string`, a ponadto
   - typ `void`, który nie ma żadnego odpowiadającego mu wyrażenia;
   - typy funkcyjne zdefiniowane indukcyjnie jako postaci $(T_1, ..., T_n) \to T_0$, gdzie T_i dla $i = 0, ..., n$ są typami (potencjalnie funkcyjnymi). Składniowo typ ten reprezentowany jest jako `[(T1, ..., Tn) -> T0]`.
2. Standardowo.
3. Zmienne należy deklarować pojedynczo.

   Niezainicjalizowane zmienne mają domyślną wartość dla swojego typu, tj. 0 dla `int`, `false` dla `bool`, `""` (puste słowo) dla `string`, zaś dla typu funkcyjnego $(T_1, ..., T_n) \to T_0$ domyślną wartością jest funkcja stale równa domyślnej wartości typu T_0.

   [Opcjonalne] Przy deklaracji może również wystąpić słowo kluczowe `auto` zamiast typu, co spowoduje jego inferencję (w statycznej kontroli typów). Może zostać użyte jedynie przy
   - deklaracji zmiennej, przy czym konieczna jest jej inicjalizacja;
   - definicji funkcji (lub lambdy) jako typ przez nią zwracany (nie może być użyty przy argumentach). Nie ma gwarancji sukcesu dla każdej poprawnej funkcji, np. nie są wspierane funkcje rekurencyjne. Inferencja typu powiedzie się, gdy przejścia po wszystkich rozgałęziach kodu kończą się instrukcją `return` o tym samym, możliwym do ustalenia typie.
4. Występują wbudowane funkcje print i println. Przyjmują jeden argument podstawowego typu. Funkcja println po wypisaniu wartości przechodzi do nowej linii.
5. Instrukcje `if`, `else` oraz `while` po warunku wymagają bloku otoczonego klamrami dla jednoznaczności.
6. Funkcje z rekurencją, funkcje zwracające void jako procedury. Jeśli wywołanie funkcji zakończy się bez wykonania instrukcji `return`, to zwracana jest domyślna wartość typu zwracanego przez funkcję. Nie dotyczy to funkcji z typem wartości oznaczonym jako `auto`.
7. Przekazywanie przez wartość i przez zmienną. Domyślne przekazywanie przez wartość. Przekazywanie przez zmienną wymaga dopisania w definicji funkcji między typem owej zmiennej i jej identyfikatorem słowa kluczowego `ref`. Tak samo przy deklaracji zmiennej funkcyjnej (oczywiście z pominięciem identyfikatora). Sposób przekazywania zmiennej odróżnia typy funkcyjne, np. typy `[(int) -> int]` oraz `[(int ref) -> int]` są różne.
9. Standardowo zmienne lokalne i globalne oraz zagnieżdżone funkcje.
10. Informatywny komunikat o błędzie i zatrzymanie interpretera.
11. jw.
12. Informatywny komunikat o błędzie i zatrzymanie type checkera.

13. jw.

17. Zmienne o typach funkcyjnych jw., mogą być przekazywane jako parametry i mogą być wartościami funkcji (domknięcia). Ponadto występują lambdy. Składnia jest następująca:

    - `lambda (T1 x1, ..., Tn xn) -> T0 { B }`, gdzie `B` jest blokiem będącym poprawnym ciałem funkcji o odpowiednim typie, lub
    - `lambda (T1 x1, ..., Tn xn) -> T0 . E`, gdzie `E` jest wyrażeniem typu T_0, będącym wartością funkcji.

    Słowo kluczowe `lambda` można również zastąpić unicode'owym symbolem λ.

    Aby wywołać lambdę nieprzypisaną do zmiennej, trzeba otoczyć ją nawiasem, np. `(lambda (int x, int y) -> int . x + y)(2, 3)` jest poprawnym wyrażeniem. W konsekwencji łańcuchy wywołań funkcji wyższych rzędów wymagają nawiasowania, np. jeśli zadeklarowana jest funkcja `[(int) -> [(int) -> int]] f`, to poprawnym wyrażeniem będzie `(f(2))(3)`, ale `f(2)(3)` już nie.