Języki programowania

II UWr 2019/20

Lista zadań nr 9 (bonusowa)

Na ćwiczenia 10 grudnia 2019

W tym tygodniu mamy kolejną bonusową listę zadań, w ramach której będziemy rozszerzać język **W** z ubiegłego tygodnia.

Zadanie 1. Proste *rekordy* możemy intuicyjnie rozumieć jako produkty indeksowane $\Pi_{i\in I}\tau_i$ rodziny typów $(\tau_i)_{i\in I}$ (dla skończonego zbioru indeksów I). W tym ujęciu typ produktu znany z systemu typów prostych to po prostu szczególny przypadek dla zbioru $I = \{1, 2\}$.

Bazując na tej intuicji:

- Rozszerz składnię języka **W** o odpowiednie konstrukcje dla rekordów (należy wymyślić odpowiednie formy składniowe) tak aby w wyrażeniach oprócz liczb i działań na liczbach mogły pojawiać się rekordy i odpowiednie działania na nich;
- Zadaj *system typów* dla rozszerzonego języka rozszerzając odpowiednio pojęcie *dobrze sformowanych* wyrażeń/instrukcji/programów
- Rozszerz *semantykę operacyjną* (w wybranym formalizmie, ale patrz kolejne zadanie) o dodane do języka konstrukcje.

Czy w rozszerzonym języku umiemy wyrazić typ unit?

Zadanie 2. Udowodnij bezpieczeństwo typów dla rozszerzonego języka, korzystając z metodologii *progress/preservation*.

Zadanie 3. Analogicznie do rekordów możemy myśleć o prostych *wariantach*, które uogólniają typ sumy rozłącznej na indeksowane rodziny typów postaci, intuicyjnie $\Sigma_{i\in I}\tau_i$ (znów, dla skończonego zbioru indeksów I).

Analogicznie do zadania pierwszego, rozszerz składnię, system typów i semantykę języka **W** o warianty. Czy w rozszerzonym języku umiemy wyrazić typy void lub bool? Czy (być może dodając odpowiednie wbudowane operacje na liczbach) możemy pozbyć się instrukcji if i zmienić semantykę instrukcji while na bardziej naturalną (jeśli chodzi o warunek pętli)?

Zadanie 4. Udowodnij bezpieczeństwo typów dla języka rozszerzonego o warianty, jak zwykle korzystając z metodologii *progress/preservation*.

Zadanie 5. Do języka **W** rozszerzonego o rekordy i warianty dodajemy globalne definicje funkcji (à la C) rozszerzając definicję składni w następujący sposób (język nazywamy **W+**):

$$
\begin{aligned}
d &::= f(x_1, \ldots, x_n) = \mathsf{vars}\ (x := e)^*\ \mathsf{in}\ c;\ \mathsf{return}\ e \\
c &::= \ldots \mid x := f(e_1, \ldots, e_n) \\
p &::= d^* \mathsf{in\ vars}\ (x := v)^*\ \mathsf{in}\ c
\end{aligned}
$$

Po pierwsze, dla wygody zastępujemy blok w którym deklarujemy jedną zmienną blokiem w którym możemy zadeklarować dowolnie dużo zmiennych. Wyrażenia pozostają bez zmian, jednak do instrukcji dodajemy nową konstrukcję: wywołanie funkcji $f(e_1, \ldots, e_n)$. Program składa się z ciągu deklaracji funkcji, deklaracji zmiennych których ostateczne wartości chcemy uzyskać jako wynik (zwróć uwagę że uogólniamy $\underline{n}$ do dowolnych *wartości* v) i instrukcji. Każda z deklaracji zaś składa się z nazwy (f), listy *parametrów formalnych* (które zachowują się jak pozostałe zmienne i są związane w ciele funkcji)

i ciała. Ciało składa się z deklaracji zmiennych, instrukcji i wyrażenia które zostanie obliczone po wykonaniu instrukcji, a którego wartość (intuicyjnie) funkcja ma zwrócić. Zadeklarowane zmienne są związane zarówno w instrukcji, jak i w zwracanym wyrażeniu (zauważ że w przeciwnym wypadku nie byłyby potrzebne).

Zaproponuj system typów (nie musisz dbać aby był bezpośrednio/łatwo implementowalny) i semantykę operacyjną (w wybranym formalizmie) dla języka **W+**. Możesz przyjąć że procedury a) mogą wywoływać tylko te zdefiniowane *przed* nimi (w ciągu procedur w definicji programu p); lub b) są *rekurencyjne*, a w szczególności wzajemnie rekurencyjne, tj. mogą wywoływać się nawzajem. Drugi wariant jest istotnie ciekawszy, pierwszy jest nieco prostszy.

Zadanie 6. Udowodnij bezpieczeństwo typów dla **W+** w wybranym przez siebie wariancie.