Nr rejestracyjny: 2019/33/B/ST6/00289; Kierownik projektu: prof. dr hab. Witold Charatonik

Maszyny abstrakcyjne dla języków programowania: podejście derywacyjne

Wszyscy na co dzień korzystamy z komputerów, które działają na podstawie oprogramowania napisanego w jakimś języku programowania wysokiego poziomu i polegamy na tym, że spełnione są dwa istotne warunki: po pierwsze, że programista zakodował w języku programowania algorytm (czyli opis tego, co komputer ma wykonać) w sposób poprawny i zgodny ze specyfikacją; po drugie, że komputer poprawnie (i efektywnie) wykonuje polecenia zawarte w tym programie.

Języki programowania to sztuczne języki używane w komunikacji ludzi z komputerami do opisu zadań, które komputer ma wykonać. Powodzenie takiej komunikacji – podobnie jak w komunikacji międzyludzkiej przy użyciu języków naturalnych – zależy od tego, czy obie strony dobrze rozumieją znaczenie konstrukcji językowych, którymi się posługują. Języki programowania od języków naturalnych różnią się m. in. tym, że znaczenie każdego "zdania" w takim języku powinno być bardzo precyzyjnie określone tak, żeby zarówno ludzie jak i komputery rozumieli je w ten sam sposób. Konsekwencje błędów czy niejednoznaczności w komunikacji między człowiekiem a komputerem mogą być poważne, szczególnie w przypadku niektórych systemów, np. tych używanych w transporcie czy medycynie, gdzie od poprawności działania komputerów może zależeć życie i zdrowie ludzi.

Dlatego języki programowania – oprócz precyzyjnych zasad budowania poprawnych konstrukcji składniowych – powinny mieć precyzyjnie określone znaczenie tych konstrukcji. Taki opis znaczenia konstrukcji języka nazywamy jego formalną semantyką, a do jej opisu używamy języka matematyki, który umożliwia precyzyjne wnioskowanie o różnych własnościach samego języka, jak i poszczególnych programów w nim napisanych. W informatyce wykorzystuje się wiele różnych sposobów definiowania semantyki języków programowania, w zależności od potrzeb i od tego, dla kogo jest przeznaczona. Oprócz wysokopoziomowej, zrozumiałej dla programisty semantyki musimy umieć dany język zaimplementować na komputerze – wtedy musimy stworzyć niskopoziomowy opis realizacji poszczególnych konstrukcji języka, który będzie bezpośrednio wykonywany przez maszyny. Wszystkie takie opisy semantyki dla danego języka powinny być równoważne tak, aby można było uznać, że komputer wykonuje dokładnie to, co opisano w programie.

Ogólnym celem badań prowadzonych w ramach tego projektu jest dostarczenie nowych narzędzi – zarówno teoretycznych, jak i praktycznych – do badania funkcyjnych języków programowania, w szczególności różnych sposobów opisu ich formalnej semantyki. Funkcyjne języki programowania stanowią grupę języków o kapitalnym znaczeniu w informatyce. Niektóre z nich (np. ML czy Haskell) znajdują zastosowanie w aplikacjach komercyjnych, choć nie są tak popularne jak np. języki obiektowe. Jednak w miarę rozwoju badań, a także zmieniających się potrzeb przemysłu informatycznego, konstrukcje i narzędzia rozwijane w ramach paradygmatu funkcyjnego są adaptowane także w innych językach powszechnie używanych przez programistów, takich jak Scala, Ruby czy Java.

Jedną z form opisu semantyki języków programowania, w tym języków funkcyjnych, są maszyny abstrakcyjne. Jest to forma pośrednia pomiędzy wysokopoziomową semantyką—zrozumiałą dla programisty, ale pozbawioną wielu szczegółów dotyczących implementacji—a niskopoziomową semantyką precyzującą jak należy wykonywać poszczególne konstrukcje. Maszyny abstrakcyjne są używane jako narzędzie teoretyczne do badania języków programowania, ale również jako prototypowy model implementacji, w którym istotnym aspektem jest efektywność działania. Celem projektu jest rozwój formalnego opisu semantyki operacyjnej języków programowania oraz metod automatycznego generowania jednej semantyki z drugiej, ze szczególnym uwzględnieniem konstrukcji, analizy i optymalizacji maszyn abstrakcyjnych.