Uniwersytet Wrocławski

dr hab. Dariusz Biernacki, prof. UWr
Instytut Informatyki UWr
ul. Joliot-Curie 15, 50-383 Wrocław
e-mail: dabi@cs.uni.wroc.pl

Wrocław, 07.07.2022 r.

# Recenzja rozprawy doktorskiej mgr. Michała Szynwelskiego zatytułowanej „A functional programming language for sets with atoms"

## Wstęp

Rozprawa doktorska mgr. Michała Szynwelskiego zatytułowana „A functional programming language for sets with atoms" stanowi prezentację semantycznych podstaw, implementacji oraz zastosowań funkcyjnego języka programowania N$\lambda$, opracowanego i rozwijanego przez autora rozprawy pod kierunkiem prof. Klina. Podstawowym celem realizowanym przez język N$\lambda$, odróżniającym go od większości istniejących języków programowania, jest umożliwienie programiście wyrażania obliczeń na pewnych strukturach nieskończonych, ale przy ich *skończonej* reprezentacji w języku, opartej na formułach logiki pierwszego rzędu. Strukturami relacyjnymi, które posiadają własność skończonej reprezentacji są w szczególności *skończenie orbitowe zbiory z atomami* i to one stanowią podstawę modelu obliczeniowego języka N$\lambda$.

Rozprawa zasadniczo zawiera dwie główne części: część teoretyczną, która prezentuje pewien model języka N$\lambda$ rozumianego jako rozszerzenie rachunku $\lambda$ oraz część praktyczną, w której autor opisuje implementację N$\lambda$ w języku Haskell wraz z przykładami użycia języka. Rozprawę uzupełnia wstęp do teorii zbiorów z atomami, jak również porównanie z istniejącymi językami operującymi na takich strukturach.

## Wyniki zaprezentowane w rozprawie

Teoretyczny model języka N$\lambda$ został wprowadzony w rozdziale 3 rozprawy, jako rozszerzenie rachunku $\lambda$ z typami prostymi o typ atomów (rachunek jest sparametryzowany strukturą relacyjną atomów $\mathcal{A}$), typ wartości logicznych, konstruktor typu zbioru nad (niefunkcyjnym) typem elementów oraz o zestaw stałych operujących na wartościach powyższych typów. Semantyka denotacyjna języka zadana jest w tradycyjny sposób: typ $\alpha$ reprezentuje odpowiedni zbiór, a dobrze typowane (zamknięte) wyrażenie $M$ typu $\alpha$ reprezentuje element zbioru reprezentującego $\alpha$. Ważnym założeniem w definicji semantyki typów jest wymóg by konstruktor zbioru oraz przestrzeń funkcyjna ograniczone były, odpowiednio, do podzbiorów oraz funkcji ze *skończonym wsparciem*. Tak skonstruowana semantyka denotacyjna bardzo czytelnie definiuje znaczenie języka, jednak nie nadaje się ona do bezpośredniej implementacji w postaci ewaluatora – np. semantyka stałych takich jak `map` wymaga przetworzenia każdego elementu potencjalnie nieskończonego zbioru.

W celu uzyskania modelu implementacji języka, autor wprowadza jego semantykę operacyjną w formacie SOS (ang. structural operational semantics), a zatem strukturalnej semantyki operacyjnej małych kroków. Taki wybór wymaga rozszerzenia języka o zmienne atomowe, a także o wyrażenia pośrednie, niedostępne dla programisty: formuły logiczne pierwszego rzędu (uogólnienie wartości logicznych), wyrażenia reprezentujące zbiory oraz warianty. Reguły redukcji zaproponowane przez autora posiadają dwie szczególne cechy. Po pierwsze, przejście między termami definiowane jest w kontekście formuły logicznej nad zmiennymi atomowymi, który może się zmieniać w drzewie wyprowadzenia danego kroku redukcji. Po drugie, znaczna część reguł redukcji warunkowana jest zachodzeniem w wybranej strukturze relacyjnej $\mathcal{A}$ pewnych formuł logicznych zależących od bieżącego kontekstu, przy dowolnym wartościowaniu zmiennych atomowych. Zaprezentowana semantyka jest niedeterministyczna i co więcej, pozwala na redukcję w ciele $\lambda$-abstrakcji – na fragmencie odpowiadającym rachunkowi $\lambda$ jest to pełna $\beta$-redukcja, a zatem nie definiuje żadnej konkretnej strategii ewaluacji.

W rozdziale 4, autor przedstawia standardowy dowód twierdzenia o zachowaniu typu (Twierdzenie 4.2). Jak zwykle w przypadku tego twierdzenia, dowód w kluczowy sposób korzysta z lematu o podstawieniu (lemat 4.1), którego szczegółowy dowód również został zaprezentowany.

W rozdziale 5, znajdujemy potwierdzenie oczekiwanego rezultatu: semantyka operacyjna zachowuje semantykę denotacyjną wyrażeń. Ponownie, dowód prowadzony jest wedle klasycznego schematu, z wykorzystaniem standardowego lematu o podstawieniu (Lemat 5.1). Warto natomiast odnotować, że zachowanie denotacji wyrażeń zaangażowanych w redukcję, warunkowane jest prawdziwością kontekstu, w którym redukcja ma miejsce (Twierdzenie 5.2).

W związku z tym, że autor nie zdecydował się na wybranie strategii redukcji w semantyce operacyjnej języka, w rozdziale 6 pokazany jest dowód twierdzenia Churcha-Rossera, które zapewnia jednoznaczność postaci normalnych czy też pozwala scharakteryzować równość termów (zdefiniowaną jako symetryczne domknięcie wielokrokowej redukcji) w terminach ich normalizacji. Dowód twierdzenia prowadzony jest klasyczną metodą redukcji równoległych, ale wymaga dodatkowych zabiegów. W szczególności, wykorzystany zostaje lemat o przeniesieniu relacji redukcji równoległych do bardziej restrykcyjnego kontekstu (Lemat 6.8).

Rozdział 7 zawiera dowód silnej normalizacji rachunku N$\lambda$. I tym razem autor sięgnął po klasyczną metodę dowodu, znaną z literatury, a mianowicie po metodę predykatów redukowalności Taita. Ponownie, trudnością w zaadaptowaniu znanego dowodu do rozważanego rachunku było zarządzanie kontekstem redukcji. W szczególności, definicja własności redukowalności (Definicja 7.4) w przypadku typu funkcyjnego uwzględnia wszystkie bardziej restrykcyjne konteksty niż ten rozważany.

Domknięciem prezentacji rachunku N$\lambda$ jest rozdział 8, w którym autor omawia wybrane konstrukcje programistyczne, jakie udostępnia pełny język N$\lambda$, a które albo są semantycznie nieinteresujące i dlatego zostały pominięte w części teoretycznej, albo też poważnie komplikują model języka, potencjalnie zmieniając jego własności. Do tej drugiej kategorii należy operator punktu stałego, który wymaga zupełnie innych struktur matematycznych do zdefiniowania jego semantyki denotacyjnej, a także pozbawia rachunek własności normalizacji.

W rozdziale 9 autor dokonuje porównania języka N$\lambda$ z innymi językami operującymi na zbiorach z atomami. Najwięcej miejsca poświęca poprzednikowi języka N$\lambda$, w porównaniu do którego język prezentowany w rozprawie oferuje znacznie bardziej oszczędną, ze względu na rozmiar, reprezentację zbiorów opartą na formułach logiki pierwszego rzędu. Rozdział zawiera też wyczerpujący

przegląd literatury poruszającej temat zbiorów z atomami oraz technik nominalnych.

Część praktyczna rozprawy została zrealizowana w rozdziałach 10 i 11. Rozdział 10 szczegółowo opisuje szereg aspektów związanych z implementacją pełnego języka N$\lambda$ jako pakietu języka Haskell. Implementacja wykorzystuje mechanizmy języka Haskell do zrealizowania większości konstrukcji językowych w N$\lambda$, które nie są bezpośrednio związane z obsługą wyrażeń reprezentujących zbiory z atomami. Pozostałe konstrukcje natomiast zostały zaimplementowane jako stałe, których typy wyrażone zostały w terminach typów reprezentujących atomy, formuły oraz zbiory, a także korzystających z metod klasy `Nominal`, która dostarcza metod operujących na zmiennych atomowych. Implementacja w kluczowy sposób korzysta z SMT Solvera Z3 do sprawdzania prawdziwości formuł powstających przy reprezentacji nieskończonych struktur danych. Dodatkowo, użytkownik języka ma możliwość definiowania własnego kontekstu obliczeń, który jawnie występował w formalnej semantyce operacyjnej języka.

Rozdział 11 prezentuje język N$\lambda$ w działaniu. Zestaw przykładowych zastosowań zaprezentowanych w tej części rozprawy zawiera algorytmy grafowe oraz dotyczące automatów. Wśród tych pierwszych wyróżniony zostaje algorytm $k$-kolorowania grafu, który w przeciwieństwie do innych algorytmów grafowych pokazanych w pracy, wymaga od użytkownika świadomego przejścia z grafów skończonych na nieskończone.

**Ocena rozprawy**

Głównym wynikiem rozprawy doktorskiej mgr. Szynwelskiego jest według mnie implementacja języka N$\lambda$ oraz infrastruktura zbudowana wokół niego. Jestem pod dużym wrażeniem profesjonalizmu z jakim język został zaimplementowany, udokumentowany i udostępniony wszystkim zainteresowanym. A zainteresowanych nie powinno brakować, ponieważ język N$\lambda$ w udany i stosunkowo przystępny dla programisty sposób stara się umożliwić programowanie algorytmów na strukturach nieskończonych, ale w taki sposób by kod programu nie różnił się znacząco od tego dla struktur skończonych. W związku z tym, uważam, że cały projekt jest bardzo dobrze uzasadniony.

Jak zawsze w przypadku poważnych projektów badawczych o charakterze implementacyjnym, pojawia się tu problem publikacji naukowych – rozprawa mgr. Szynwelskiego została oparta na pojedynczej publikacji, na forum o niezbyt dużym zasięgu. Uważam jednak, że tego typu przedsięwzięcia są niezwykle wartościowe i w tym konkretnym przypadku nie mam wątpliwości co do tego, że projekt posiada istotne walory naukowe, jednocześnie przenosząc dalece nietrywialną teorię na poziom praktyki programistycznej. Tak zaawansowana implementacja, w mistrzowski sposób wykorzystująca wiedzę teoretyczną oraz głęboką znajomość narzędzi takich jak Haskell, mogła być zrealizowana tylko w ramach projektu doktoranckiego i to przez doktoranta o nieprzeciętnym potencjale oraz wszechstronnej wiedzy i umiejętnościach.

Część teoretyczna rozprawy jest wartościowym uzupełnieniem implementacji. Właściwie wszystkie techniki użyte do uzyskania rezultatów zaprezentowanych w tej części są standardowe i dobrze znane. Nie oznacza to, że zaadaptowanie ich do języka N$\lambda$ nie stanowiło wyzwania. W szczególności, takim wyzwaniem z pewnością było uwzględnienie kontekstu redukcji w semantyce operacyjnej i te fragmenty teorii języka N$\lambda$ wydają mi się interesujące i zasługujące na szczególną uwagę.

Jeśli miałbym wyrazić jakieś drobne zastrzeżenie co do części teoretycznej pracy, to dotyczyłoby ono semantyki operacyjnej języka N$\lambda$. Wydaje mi się, że rozprawa zyskałaby na spójności, gdyby autor, oprócz niedeterministycznej relacji redukcji, zdefiniował semantykę operacyjną ję-

zyka jako deterministyczną relację redukcji wyznaczoną przez określoną strategię redukcji, w tym przypadku zgodną z semantyką języka Haskell. Semantyka operacyjna z rozprawy stanowi pewien ogólny model obliczeń w rachunku lambda rozszerzonym o wyrażenia reprezentujące zbiory z atomami, ale nie odpowiada bezpośrednio implementacji, np. brakuje bezpośredniego związku między postaciami normalnymi w tych dwóch formalizmach (w Haskellu ewaluacja zatrzymuje się na $\lambda$-abstrakcji, a w semantyce operacyjnej N$\lambda$ – nie).

**Podsumowanie**

Podsumowując, mgr Michał Szynwelski zaprojektował, sformalizował i zaimplementował bardzo interesujący język programowania, opierając go na solidnych podstawach teoretycznych. Uważam, że przedstawiona przez niego rozprawa doktorska zawiera ciekawe wyniki znajdujące się w aktywnym nurcie badań współczesnej informatyki i że spełnia ona zwyczajowe i ustawowe wymogi stawiane pracom doktorskim, w związku z czym wnoszę o dopuszczenie jej do publicznej obrony.

*Dariusz Biernacki*