RECENZJA ROZPRAWY DOKTORSKIEJ

„GADTs for Reconstruction of Invariants and Postconditions" mgra Łukasza Stafiniaka

AUTOR RECENZJI: DR HAB. ALEKSY SCHUBERT, PROF. UW
Wydział Matematyki, Informatyki i Mechaniki
Uniwersytetu Warszawskiego

1 Problematyka i zawartość pracy

Tematyka rozprawy Tematyka pracy związana jest z wytwarzaniem oprogramowania wysokiej jakości. Przez wysoką jakość rozumieć będziemy tutaj takie kształtowanie bazy kodu źródłowego, aby istniało dobrze ugruntowane przekonanie, że wykonuje ono to, czego od oprogramowania oczekujemy.

Warto tutaj nadmienić, że już samo określenie oczekiwań wobec oprogramowania jest wysoce nietrywialnym zagadnieniem. W typowych realizacjach oczekiwania wobec oprogramowania wyrażane są w języku naturalnym, co w zasadzie przekreśla możliwość ich automatycznej weryfikacji względem wytworzonego kodu. Jednocześnie jednak istnieją języki formalnego wyrażania własności programów (np. metoda Z, CASL, Java Modeling Language etc.), które pozwalają na zapis własności z możliwością ich weryfikacji względem dostępnego kodu programów. Jednak konieczność wyrażania własności w tych językach jest postrzegana przez wytwórców oprogramowania jako dodatkowe obciążenie w procesie realizacji założonej funkcjonalności. Często słyszy się z ich strony narzekanie – po co pisać dodatkowo specyfikacje, skoro i tak oprogramowanie działa? To narzekanie słychać pomimo licznych dowodów wskazujących na to, że wprowadzenie specyfikacji do procesu wytwórstwa poprawia jakość kodu.

Jednym z ważnych nurtów w programowaniu ze specyfikacjami jest tzw. programowanie funkcyjne (ew. funkcjonalne). W ramach języków programowania tego rodzaju wykształciła się kultura systematycznego dokumentowania wymagań dotyczących programów w postaci statycznie sprawdzanych typów określających funkcjonalność jednostki wykonawczej (najczęściej funkcji) w postaci bardzo ogólnego opisu wyrażonego w języku tzw. typów prostych (wzbogacanych do schematów typowych w językach z rodziny ML). Z czasem tak ogólne specyfikacje przestały wystarczać i w ostatnich czasach coraz częściej proponowane są języki programowania lub też rozszerzenia istniejących języków wyposażone w systemy typów zależnych (ang. dependent type systems), pozwalające na daleko bardziej szczegółowe wyrażanie własności programów, zbliżające języki funkcyjne do języków programowania z własnym systemem specyfikacji (takich jak Eiffel, Spec#, czy nawet Java w połączeniu z Java Modeling Language).

Wkład pracy zlokalizowany jest w obrębie badań nad poszerzaniem możliwości funkcyjnych języków programowania z typami zależnymi i przynosi daleko idące wsparcie dla automatycznego

A.S.

generowania specyfikacji na podstawie istniejącego kodu, a także już dostępnych fragmentów specyfikacji.

Problem naukowy i jego znaczenie W tak określonej dziedzinie tematycznej kandydat podążał następującą wielostopniową ścieżką badawczą:

1. Opracowanie nowego systemu typów, który pozwalałby na automatyczne wyprowadzanie szerokiego zakresu specyfikacji programów funkcyjnych w postaci niezmienników dla wyrażeń rekurencyjnych oraz warunków końcowych (ang. postconditions) dla funkcji. Wskazane niezmienniki i warunki końcowe powinny być wyrażane za pomocą odpowiednio określonych typów zależnych z obecnością wyrażeń arytmetycznych.

2. Przeniesienie problemu wyprowadzania typów na grunt wykazywania spełnialności dla odpowiednio dobranego problemu spełnialności ograniczeń (ang. constraint satisfaction problem).

3. Znalezienie satysfakcjonującej metody rozwiązywania ograniczeń uzyskanych w poprzednim kroku.

Z tak obranego programu wynikły następujące problemy badawcze:

1. Określenie systemu typów tak, aby wnosił on nowe wartości do istniejącego zasobu technik weryfikowania własności programów. Tutaj tym systemem typów są uogólnione algebraiczne typy danych (GADT) z dodatkowo wprowadzonymi abstrakcjami konstruktorowymi, ograniczeniami numerycznymi oraz typami egzystencjalnymi.

2. Ustalenie poprawności zaproponowanego systemu typów. Tutaj zrealizowane przez translację do systemu $\mathrm{HMG}(X)$.

3. Zaproponowanie translacji termów-programów na ograniczenia i udowodnienie jej poprawności. Tutaj zaproponowano rozszerzenie techniki znanej z pracy na temat $\mathrm{HMG}(X)$.

4. Zaproponowanie metody rozwiązywania ograniczeń. Tutaj przez zaproponowane nowe algorytmy abdukcji.

5. W związku z istniejącymi już pracami, które idą tą ścieżką, konieczne jest też w ramach prac badawczych dokonanie porównania z istniejącymi rozwiązaniami. Tutaj przez określenie odpowiednio wyczerpującego zestawu programów porównawczych (ang. benchmarku) i pokazanie sposobu działania zaproponowanego rozwiązania w porównaniu z istniejącymi.

Przyjęło się, że metody formalnej weryfikacji własności oprogramowania nie są postrzegane jako praktyczne: ich stosowanie wymaga pokonania dużej bariery wejścia i to w dwóch wymiarach, mianowicie personel je stosujący wymaga bardzo specjalistycznego wyszkolenia, ale też i stosowanie tych metod jest czasochłonne. W związku z tym wszelkie badania z tego zakresu nie są uważane za główny nurt informatyki. Tym niemniej stały wysiłek społeczności naukowej sprawił, iż w ostatnim dziesięcioleciu odnotowany został znaczący postęp w zakresie technik wspomagających wytwarzanie niezawodnego oprogramowania tymi metodami. Zaproponowana praca doktorska jest przyczynkiem wnoszącym poszerzenie zakresu programów funkcyjnych, dla których możliwe jest automatyczne generowanie specyfikacji. Pozwala ona także na lepsze zrozumienie zakresu praktycznych technik specyfikowania programów funkcyjnych za pomocą typów zależnych.

A.S

Treść rozprawy Praca składa się z 6 rozdziałów, 4 dodatków i referencji. Rozdział 1 zawiera wprowadzenie w tematykę pracy oraz podsumowuje jej wkład. W rozdziale 2 omówione zostają pokrewne systemy typów, w szczególności omówiony jest system typów $\mathrm{HMG}(X)$, który stanowi istotną bazę metodologiczną zaproponowanego w rozprawie rozwiązania – jego poprawność wykazana jest przez zaproponowanie odpowiedniej translacji do systemu $\mathrm{HMG}(X)$, a zatem ten ostatni znajduje się bazie zaufania ocenianej tutaj rozprawy. Rozdział 3 poświęcony jest przedstawieniu zaproponowanego przez kandydata systemu typów, $\mathrm{MMG}(X)$. Rozdział 4 zawiera opis zaproponowanych technik rozwiązywania ograniczeń. W rozdziale 5 znajdujemy porównanie zakresu działania zaproponowanej metody w stosunku do innych podejść znanych z literatury, a także porównanie wydajności czasowej podejść. Rozdział 6 przynosi wnioski końcowe oraz propozycje pracy na przyszłość.

Integralną częścią rozprawy są dodatki. W dodatku A znajdujemy dowód poprawności zaproponowanego systemu typów oraz dowód poprawności translacji do systemu $\mathrm{HMG}(X)$. Zawiera on także rezultaty pozwalające odnieść rozwiązywanie wygenerowanych ograniczeń do początkowego zadania typowania, a także pozwalające odnieść wynik działania algorytmu abdukcji do tegoż. Dodatek B zawiera rozszerzony opis algorytmu abdukcji. W dodatku C znajdujemy kod programów, stanowiących punkt odniesienia dla testów wydajnościowych oraz testów zakresu działania przedstawianej metody (czyli szczegółowy opis wspomnianego benchmarku). Wreszcie dodatek D zawiera podręcznik obsługi programu InvarGenT, będącego praktycznym rezultatem przedstawionej rozprawy.

2 Ocena

Wiedza teoretyczna i umiejętność samodzielnego prowadzenia pracy naukowej

Przedstawiona rozprawa świadczy o dużej i głębokiej wiedzy kandydata w zakresie tematyki typów zależnych i ich zastosowania dla funkcyjnych języków programowania. Autor rozprawy musi się także wykazywać doskonałą znajomością zagadnień związanych z heurystykami stosowanymi w abdukcji. Pewne zastrzeżenia jednak budzi kwestia umiejętności samodzielnego prowadzenie pracy naukowej. Bez wątpienia autor jest w stanie zajmować się badaniem języków programowania i twórczo rozwijać napotykane tam zagadnienia. W szczególności zaprezentowany materiał przedstawia badanie w zakresie, który nie był wcześniej tak dokładnie i szczegółowo rozwijany przez pracowników Instytutu Informatyki Uniwersytetu Wrocławskiego. Jednak, jak to dalej wykażę, styl napisania doktoratu nie jest zbyt komunikatywny, co może oznaczać, że kandydat bez istotnego wsparcia w otoczeniu na etapie wyrażania uzyskanych pomysłów, może nie być w stanie uzyskać publikowalnych rezultatów.

Zastrzeżenie to jednak jest mało istotne, gdyż współcześnie badania informatyczne są prowadzone głównie w zespołach, a ze zdobytą wiedzą i wykazaną kreatywnością kandydat ma duże szanse być wsparciem dla grupy, w której się znajdzie – zresztą już znalazł zatrudnienie w dobrym zagranicznym ośrodku.

Uwagi merytoryczne

Kompozycja pracy Przede wszystkim zaproponowana praca doktorska została napisana, jak typowa praca z dziedziny języków programowania: główna część to omówienie przedstawionego rozwiązania, natomiast wszelkiego rodzaju dowody zostały przeniesione do dodatków. Tego rodzaju układ – może adekwatny dla konferencji – nie jest dobrym układem dla pracy doktorskiej z nauk matematycznych. Tutaj podstawą metody badawczej jest jednak metoda dedukcyjna, bez której właściwie

A.S

matematyka nie ma racji bytu. Warto nadmienić, że metoda ta ma ten niezwykły aspekt, nieobecny w naukach eksperymentalnych takich jak fizyka czy chemia, iż jesteśmy w stanie dostarczyć nie tylko sam wynik, ale też i całość procesu, jaki do niego doprowadził. Jednocześnie z drugiej strony patrząc, nauka to działanie, w którym istotny jest nie tylko wynik, ale też i metoda – w naukach doświadczalnych wynik jest niewiele wart, jeśli doświadczenia, na których się opiera, nie dadzą się powtórzyć niezależnie w innych laboratoriach niż laboratorium autora artykułu. Dlatego integralną częścią pracy naukowej jest skrupulatny opis eksperymentu, czego odpowiednikiem w naukach dedukcyjnych jest przedstawienie dowodu. Przedstawienie dowodu, który jest w stanie zrozumieć inny matematyk.

Dodatki do pracy to miejsce, gdzie umieszcza się informacje mniej ważne, ale z jakiegoś drugorzędnego powodu pożyteczne. Stąd przeniesienie dowodów do dodatku jest znakiem nadania im takiej wagi. Tymczasem to, co może jest pożyteczne w artykule konferencyjnym, gdzie obowiązują ograniczenia na długość wypracowania, w formacie bez takiego ograniczenia staje się niewłaściwe i może zagrażać odpowiedniemu przeprowadzeniu matematycznej metody naukowej.

Przedmiot badań Jako przedmiot badań przedstawiony jest system typów dla małego języka programowania reprezentowanego jako język wyrażeń zdefiniowany w tabeli 3.2. Istotny problem stanowi tutaj zrozumienie, jaka własność jest gwarantowana przez poprawne otypowanie w systemie $\mathrm{MMG}_{\exists}(X)$? Zwykły sposób postępowania polega tutaj na określeniu w jakiś standardowy sposób semantyki (np. metodą małych lub dużych kroków czy też metodą denotacyjną), a następnie wyrażenie oczekiwanej własności w terminach tejże semantyki. Wtedy klarowne jest to, o co chodzi. Raczej też istotne jest, aby tę własność sformułować stosunkowo szybko i na odpowiednio wyróżnionym miejscu. Tutaj ta kluczowa część, zamiast znaleźć opracowanie tuż po określeniu składni języka, została wyrzucona na koniec rozdziału 3. W związku z czym przez cały ten rozdział w zasadzie nie wiadomo, o co chodzi w badanym przedmiocie.

Warto tutaj naświetlić, jak taka metoda jest niewłaściwa. Przez cały bowiem rozdział 3 czytelnik nabywa pewnego intuicyjnego rozumienia semantyki omawianego języka. Dzieje się to przez kontakt ze wszystkimi definicjami i dowodami tej części rozprawy – trochę na zasadzie czytania rozległej i rozproszonej aksjomatyzacji. Zdobyte w ten sposób przekonanie na temat działania języka zostaje następnie skonfrontowane z właściwym określeniem i to bez wyraźnego (co jest w zasadzie niemożliwe, gdyż „aksjomatyzacja" nie ma tutaj siłą rzeczy charakteru systematycznego) porównania tych dwóch określeń.

Warto też nadmienić, że w zaproponowanym w pracy podejściu docelowa własność wyrażona została nie w sposób bezpośredni, ale poprzez szereg pośrednich pojęć (*computational equivalence, HMG-form, tag erasure*), które zaciemniają obraz. Obraz i tak nie do końca jasny, bo przecież użyty w pracy język jest tylko modelem dla zjawisk w większym, prawdziwym języku programowania. Należy tutaj podkreślić, że semantyka to sposób wyjaśniania pojęć nieznanych za pomocą znanych. Uciekanie się do niestandardowych konstrukcji daje tutaj zawsze w jakimś zakresie efekt wyjaśniania pojęć nieznanych przez nieznane.

W sytuacji, gdy sama dziedzina jest uważana za dosyć hermetyczną, takie prowadzenie wywodu nie przyczynia się do zwiększenia zaufania do uzyskanych w ten sposób wyników.

Spójność definicji Praca pełna jest mniej lub bardziej istotnych niedopowiedzeń. W istotnej części zapewne wynika to z decyzji o umieszczeniu dowodów w dodatku. Przenosząc dowody tamże, autor pozbawił się – ale także swoich czytelników, w tym mnie – jasnego kryterium dotyczącego umiejscowienia różnych informacji: pojęcie powinno być wprowadzone przed jego użyciem, najlepiej w niedalekiej odległości od tego lub też przy okazji definiowania spójnej tematycznie grupy

A.S.

pojęć. W zaproponowanym doktoracie zdarzają się niejednokrotnie sytuacje, kiedy jakaś istotna dla zrozumienia rzecz nie ma właściwego określenia. Oto dla ilustracji, zdecydowanie niepełna, lista przykładów:

- kategoria syntaktyczna v_{type} w tabeli 3.1 nie ma ani wyjaśnienia w samej tabeli, ani w tekście omawiającym zawartość tabeli;
- wyrażenie postaci **runtime failure** s nie ma swojej pozycji w tabeli 3.2, gdzie znajduje się składnia wyrażeń języka MMG, podobnie trudno się dopatrzyć w składni języka czy zestawie dostępnych typów typu String, co daje taki efekt, że przypadek tego typu w niektórych sytuacjach nie jest obsługiwany, jak to ma miejsce na przykład w tabeli 3.7;
- obecna w regule CLAUSE koniunkcja $\wedge$ z wielokropkiem oznaczającym wyliczenie wyrażeń jakiejś postaci nie była nigdzie wprowadzona, a zwyczajowe znaczenie takiego wyrażenia nie daje jednoznacznego określenia, o co chodzi; podobnie dla alternatywy $\vee$ w regule NEGCLAUSE;
- nie znalazłem wyjaśnienia dla subtelnej różnicy między $\overline{\tau}$ a $\overline{\tau}$ w sformułowaniu twierdzenia 3.2;
- notacja $(e_i, \mathbf{F})$ dla $i = 1, 2$ w dwóch ostatnich pozycjach tabeli 3.10 jest zapewne wynikiem opuszczenia operatora $\mathcal{E}$;
- nie znalazłem w rozprawie definicji notacji $\mathrm{RetType}(\tau, \alpha)$ kluczowej dla zrozumienia definicji w tabeli 3.13;
- zaskoczenie też budzi obecność notacji $\overline{\tau}'$ w argumencie konstruktora ε_K w założeniach reguły EXABS w tabeli 3.13; przecież zgodnie z opisem na stronie 53 te konstruktory typowe mają być jednoargumentowe;
- brakuje określenia oznaczenia usorts występującego w definicji algebry wolnej na stronie 60,
- nie jest jasny status sortów postaci s_{ty} w definicji algorytmu abdukcji w tabeli 4.1,
- nie była wprowadzona notacja $A \backslash B$ w punkcie 1 definicji 4.12,
- nie zostało wprowadzone określenie *one side of the atom* c na stronie 69 w omówieniu procedury Split,
- w głównym tekście nie były wprowadzone operacje DisjElim, Simpl, Atomized użyte w tabeli 4.3,
- nie było wprowadzone określenie *potential solution to an inference problem* w sformułowaniu twierdzenia 4.15.

Dopiero długie i mozolne, a także wielokrotne czytanie całej rozprawy pozwala na rozwikłanie tych pojęć, które czasem są wprowadzane w dodatku, czasem ich znaczenie wynika z treści – tak niedocenionych przez autora – dowodów, a czasem wymaga własnego dookreślenia tak, aby zapewnić logiczną spójność wywodów.

Innym problemem podobnego rodzaju są zakresy, po których biegną, różnego rodzaju metazmienne. Na przykład tabela 3.1 określa, że w formule uniwersalnej (schemacie typu) w nawiasach kwadratowych mamy $\exists \overline{\alpha}.\overline{a}$, ale na początku sekcji 3.2 mamy inne określenie: może tam występować formuła w postaci rozwiązanej $\exists \overline{\alpha}.E$ lub zmienna predykatowa $\chi(\beta)$, co zmienia zakres przebiegu

A.S.

meta-zmiennej D i w zasadzie nie wiadomo, co tam może się pojawiać. Inny przykład to sformułowanie reguły ExAbs w tabeli 3.13. Widoczny tam kwantyfikator $\forall i$ nie wiąże w żaden widoczny sposób zmiennej i, co każdemu czytelnikowi będzie kazało szukać tutaj jakiegoś ukrytego znaczenia (w tekście, który przez swoją konstrukcję co chwila każe czytelnikowi takiego szukać).

Zjawisko to jest także zauważalne przy różnego rodzaju wyrażeniach, które używają symbolu $\models$. Znaczenie na przykład przypadków od **f-Enrich** po **f-And** w sformułowaniu faktu A.2 jest nie do odcyfrowania na podstawie definicji podanych na początku sekcji A.1.2. Brak dowodu pozostawia w zasadzie czytelnika bezradnym. Ja sam osobiście w dalszym ciągu rozważań dodatku A nie korzystałem z tego stwierdzenia, a raczej posługiwałem się bezpośrednio własnym rozumieniem definicji.

Z kolei w tablicy 4.3 wzór

$$\wedge_\chi(A^+_\chi \subset A) \wedge \mathcal{M} \models \mathcal{Q}.(\cup_\chi A^+_\chi \Rightarrow A)[\overline{\alpha} := \overline{t}]$$

w jakiś sposób ma definiować minimalność, przy czym nie jest jasne, czy chodzi o to, że definiuje on nową relację przypadkiem oznaczoną przez $\subset$, czy też może ogranicza zakres, po którym przebiega zawieranie. Przy czym sam ten warunek też jest niejasny. Co bowiem oznacza $\wedge$ przed symbolem $\mathcal{M}$? Czy pierwsza koniunkcja ($\wedge_\chi$) jest w jakiś sposób częścią wyrażenia przed $\models$ i ogranicza jakoś zakres podstawień w ramach modelu $\mathcal{M}$, czy też może stanowi samoistne stwierdzenie, będące pierwszym koniunktem warunku, którego drugi koniunkt zaczyna się od $\mathcal{M} \models \ldots$

Podobnej klasy problem to konflikty oznaczeń, których chyba najznaczniejsze przykłady to konflikt między typami τ_1, τ_2 a τ_{m_i}, τ_{n_i} w tabeli 3.5 czy między β_1, β_2 a $\overline{\beta}$ (zgodnie z konwencją wprowadzoną na stronie 41) w definicji *no escaping variables* na stronie 59.

Kolejny przykład to definicja pojęcia *alien subterm*. Autor definiuje to pojęcie z użyciem określenia *maximally large*. Niestety nie określa on w jakim porządku ta maksymalność jest określana. Domyślnie w takich wypadkach przyjmuje się wielkość termu mierzoną liczbą wystąpień symboli składowych, ale po przeczytaniu dalszych kilku stron należy uznać, że chodzi nie o podtermy, a o wystąpienia podtermów rozumiane jako ścieżki i o porządek na tych ścieżkach.

Czytelność dowodów Jak już to zaznaczyliśmy powyżej sformułowania nie tylko definicji, ale też faktów lub lematów pozostawiają wiele do życzenia. Warto jednak naświetlić też kwestię samych dowodów.

W swoim życiu naukowym widziałem wiele przypadków, kiedy błąd w rozumowaniu matematycznym był ukryty w miejscu, które w tekście wywodu było opatrzone komentarzem: *łatwy dowód pozostawiamy czytelnikowi*. Dlatego praktykę omijania tzw. „nudnych" czy „rutynowych" fragmentów dowodów uważam za dosyć podejrzaną – zwłaszcza, gdy chodzi o materię rzadką i mniej powszechnie spotykaną.[1] Dlatego pominięte dowody w dodatkach A.1.2, A.1.4, A.2.1, A.3 uważam za istotny mankament pracy. Tym bardziej, że dowody te dałyby szansę na lepsze i szybsze zrozumienie pojęć użytych w rozprawie przez czytelnika. Na pewno też autorowi dałyby okazję do dokładniejszego przejrzenia wprowadzonej notacji i poprawienia jej.

Dodatkowo brak dowodów przy jednoczesnym braku przykładów ilustrujących wprowadzane pojęcia powoduje, że znaczne obszary pracy wymagają od czytelnika nakładu pracy znacząco wykraczającego poza typowe sytuacje.

Istniejące dowody też zawierają luki czy mylące wskazania, które nierzadko są trudne do uzupełnienia. Oto, ponownie dla ilustracji, kilka przykładów:

[1] Warto tutaj się zapoznać z przykładami błędów wśród zawodowych matematyków wymienionymi w artykule *Social Processes and Proofs of Theorems and Programs* autorstwa Richarda A. De Millo, Richarda J. Liptona i Alana J. Perlisa oraz *The role and function of proof in mathematics* Michaela de Villiersa

A.S.

- Wysoce nieoczywisty jest sposób wprowadzenia równości $\overline{\alpha\beta} \doteq \overline{\alpha'\beta'}$ w kroku 7. dowodu lematu A.7 (str. 91).
- W dowodzie lematu A.8 (str. 96) w przypadku P-HIDE wykonuje się pomocniczy dowód faktu $FV([\![\vdash p \downarrow \tau]\!]) = FV(\tau)$ tymczasem fakt ten przydałby się już na etapie lematu A.7, żeby zrozumieć prawidłowość kwantyfikacji. Wiele podobnych faktów należałoby wyodrębnić do osobnych lematów, co tradycyjnie się robi.
- W dowodzie twierdzenia 3.7 dla przypadku $\lambda_K \overline{c}$ dopiero w punkcie 4. może odbyć się wprowadzenie kwantyfikatora $\exists$ za pomocą reguły HIDE. Osłabianie rzeczywiście przydaje się do wprowadzenia do formuły predykatu $\chi_K(\alpha_1)$.
- W dowodzie tegoż twierdzenia dla przypadku pełności (completeness) używa się określenia *premise* zarówno na konkluzję reguły jak i jej założenia.
- Obecność $\wedge$ na przed $\mathrm{RetType}(\tau, \alpha_0)$ budzi podejrzenie, że wykonuje się tutaj koniunkcji po jakimś implicite obecnym zbiorze, ale za to przerzucenie $[\![\Gamma \vdash \lambda\overline{c} : \tau]\!]$ utrudnia porównanie ze wzorem docelowym w tabeli 3.13.
- Struktura dowodu twierdzenia 4.5 (część correctness) jest bardzo myląca. Treść twierdzenia wymaga dowiedzenia czterech warunków cząstkowych (relevance, validity, consistency and no escaping variables), a zatem chciałoby się widzieć cztery wyodrębnione cząstki. Tymczasem własności powyższe są zgromadzone na początku, w konwencji takiej, jakby druga miała wynikać z pierwszej, trzecia z drugiej itd. Taka struktura bardzo utrudnia zrozumienie, o co w dowodzie chodzi.

Rola specyfikacji W pracy (w rozdziale 5 zawierającym testy) autor skupił się prawie wyłącznie nad wykazywaniem użyteczności systemu w sytuacji, gdy program pozbawiony jest specyfikacji. Tymczasem jest to metodologicznie dosyć słabe podejście. Chodzi o to, że często słabym ogniwem w implementacji nie jest złe jej wykonanie, a niekoniecznie prawidłowa interpretacja wymagań wyrażonych w języku naturalnym. Wprowadzenie do procesu implementacji dwóch formalnych języków: języka programów oraz języka specyfikacji daje szansę na wykonanie interpretacji języka naturalnego na dwa sposoby i skonfrontowanie ich ze sobą. W wyniku możliwe jest dużo dokładniejsze wyeliminowanie błędów wynikających z fałszywego interpretowania pierwotnego źródła wiedzy na temat programu. Samo automatyczne wyprowadzanie specyfikacji jest potrzebne, ale w miejscach, które są w zasadzie uważane za dosyć oczywiste i bezpośrednio niezwiązane z realizacją docelowej funkcjonalności. Zaproponowany system powinien więc być szeroko sprawdzony na okoliczność konfrontacji istniejących specyfikacji ze specyfikacjami wygenerowanymi.

Uwagi podsumowujące

Metoda matematyczna w bardzo istotnym stopniu opiera się na możliwości utrwalenia wywodu. Dzięki takiej formie możliwe staje się przeprowadzenie długich i skomplikowanych konstrukcji, gdzie konieczne jest stałe nawracanie do konkretnej postaci tej lub innej definicji. Co ważne, do postaci ustabilizowanej. Gdy jednak tekst prowadzony jest tak, iż możliwe jest interpretowanie danego określenia na wiele sposobów, znaczenie staje się rozmyte i nie można łatwo skorzystać z zalet zapisu. Co więcej, gdy ktoś będzie chciał zrozumieć treść twierdzeń, to nie ma szybkiej i klarownej ścieżki, a to niweluje wartość tych twierdzeń jako szybkiego podsumowania zdobytej wiedzy. W rozprawie te wątki zostały potraktowane po macoszemu, co obniża jej wartość.

A. S.

Przy tak skonstruowanym wywodzie matematycznym raczej trudno jest wskazać poważne usterki, czy błędy (a może raczej nietrudno, ale byłoby ich bardzo dużo oraz byłyby one na poziomie szkolnym) – sprawniejsza jest weryfikacja proponowanych stwierdzeń przez własne dowody niż czytanie zaproponowanego tekstu, co też w dużej mierze czyniłem.

Przedstawiony tekst daje wiele możliwości interpretacji jego treści. Ja znalazłem swoją. Było to trudne przedsięwzięcie, zapewne możliwe tylko dzięki wielkiemu szacunkowi, jaki żywię do promotora, profesora Leszka Pacholskiego, ale pozostawia ono otwarte pytanie, czy inny czytelnik nie znajdzie interpretacji odmiennej od mojej?

Szczęśliwie jednak udało mi się przebrnąć przez całe to przedsięwzięcie i uzyskać zadowolenie wynikające ze zrozumienia oraz nabyć przekonanie o solidnej wartości całej pracy.

3 Wnioski

Jak wynika z przeprowadzonego omówienia, praca zawiera trudne i istotne dla dziedziny konstruowania poprawnego oprogramowania wyniki. Niewątpliwie rezultaty te są oryginalne i świadczą o dostatecznej sprawności kandydata w opracowywaniu modeli matematycznych dla zjawisk występujących w językach programowania oraz o jego dogłębnej wiedzy na temat przedstawiony w rozprawie. Przy całym moim krytycznym stosunku do metodologicznej strony pracy stoję jednak na stanowisku, że przedstawiony materiał świadczy o wystarczającej dojrzałości naukowej osoby, która go przygotowała. Dlatego mogę stwierdzić, że praca **spełnia warunki stawiane rozprawom doktorskim** przez *Ustawę o stopniach naukowych i tytule naukowym oraz o stopniach i tytule w zakresie sztuki* i wnioskuję o dopuszczenie mgra Łukasza Stafiniaka do dalszych etapów przewodu doktorskiego.

Berlin, 8 września 2015
Aleksy Schubert