Recenzja rozprawy doktorskiej*

"On the determination of extended classes and some other problems in Java programming language"

mgr Marka Warpechowskiego

29 października 2009

1 Problematyka i zawartość rozprawy

1.1 Tematyka rozprawy

Rozprawa mieści w nurcie badań nad semantyką języków programowania obiektowego, a dokładniej - nad semantyką klas i dziedziczenia w języku Java. Samo programowanie obiektowe pojawiło się jako osobna dziedzina już ponad czterdzieści lat temu wraz z powstaniem języka Simula 67 i przeżyło okres bardzo intensywnych badań w latach siedemdziesiątych i osiemdziesiątych minionego stulecia. Pozycja podejścia obiektowego jako głównego paradygmatu stosowanego w wytwórstwie oprogramowania utrwaliła się wraz z powstaniem języka C++ oraz mniej lub bardziej udanym dodaniem cech obiektowych do języków programowania strukturalnego. Obecnie istnieje wiele języków obiektowych, jak Python, Perl, Object Pascal, Java, C#, PHP5, itd. Również w Polsce powstał język Loglan oferujący praktycznie wszystkie (poza wielodziedziczeniem) mechanizmy obiektowe. Prace nad tym językiem są jednym ze źródeł rozprawy doktorskiej przedłożonej przez M. Warpechowskiego, będącego współautorem wielu ciekawych rozwiązań związanych z koncepcją i implementacją obiektowości.

Projektowanie i programowanie obiektowe jest już od dawna stosowane w przemyśle informatycznym. Kompilatory i interpretery języków obiektowych są tworzone przez wyspecjalizowane firmy komercyjne, pojawia się więc pytanie

*Autor recenzji: prof. dr hab. Andrzej Szałas, Wydział Matematyki, Informatyki i Mechaniki Uniwersytetu Warszawskiego, 02-097 Warszawa, andsz@mimuw.edu.pl.

czy jest miejsce na uprawianie tej tematyki w ramach badań naukowych. Omawiana rozprawa pokazuje, że dziedzina ta wymaga stawia nadal szereg niebanalnych problemów koncepcyjnych i implementacyjnych, wymagających prowadzenia badań. Okazuje się bowiem, że nawet tak fundamentalne pytania, jak podanie semantyki dla wystąpień identyfikatorów, nie doczekały się satysfakcjonującej odpowiedzi ze strony producentów narzędzi do tworzenia programów obiektowych. Ma to miejsce, w szczególności, w języku Java, który jest jednym z najpopularniejszych języków tworzenia takich programów. Okazuje się, że rozumienie semantyki tego języka nie jest jednolite w różnych środowiskach twórców kompilatorów. Różnie rozumieją je także programiści. Pokazuje to choćby przykład 1.2.4 z rozprawy, który testowany na kilku kompilatorach dawał kilka różnych wyników. Przykład ten to zaledwie kilka linii kodu, co wyraźnie pokazuje, że nawet bardzo dobrze rozpoznane i opracowane języki mają nadal istotne luki semantyczne.

1.2 POSTAWIONY PROBLEM

Głównym celem rozprawy jest określanie znaczenia identyfikatorów, którego szczególnym przypadkiem jest wyznaczanie dla każdej klasy występującej w danym programie jej *klasy rozszerzanej*, czyli klasy z której dana klasa bezpośrednio dziedziczy. Problem wyznaczania klasy rozszerzającej okazuje się trudny w przypadku języków dopuszczających zagnieżdżanie klas, w tym w języku Java, gdzie problem dodatkowo komplikuje się w związku z możliwością użycia typów kwalifikowanych.

Problem postawiony w rozprawie polega na odpowiednim uściśleniu semantyki Javy i podaniu algorytmów rozwiązujących następujący problem:

> dla danej struktury klas sprawdzić czy program jest poprawnie zbudowany i w przypadku pozytywnej odpowiedzi podać funkcję wyznaczającą klasę rozszerzaną dla dowolnej klasy rozszerzającej.

Aby dokładniej przedstawić problem, niezbędne jest przytoczenie kilku definicji. Przez *typ kwalifikowany* rozumiemy (skończony) ciąg identyfikatorów klas. Dziedziczenie może być wyrażone poprzez jedną z poniższych klauzul:

- class A extends B (1)
- class A extends $B_1.B_2 \ldots B_n$ (2)

  gdzie $B_1.B_2 \ldots B_n$ jest typem kwalifikowanym.

Brak klauzuli „extends" oznacza, że klasa dziedziczy po systemowej klasie „Object".

Oczywiście w danym programie wiele klas może mieć ten sam identyfikator.

Kluczową rolę w rozprawie odgrywają funkcje *bind* i *inh*, dla których:

- $bind(T, C) = D$ oznacza, że semantyką typu T w klasie C jest klasa D

- $inh(C) = D$ oznacza, że bezpośrednią nadklasą klasy C jest klasa D.

Dokładna indukcyjna definicja *bind* i *inh* jest podana w rozprawie jako definicja 2.1.2.

Z punktu widzenia rozważań Autora wystarczy przyjąć, że program jest *poprawnie zbudowany* jeśli dla jego struktury klas istnieje funkcja *inh* określona dla każdej klasy i spełniająca warunki:

1. $inh(K) = bind(ext(K), decl(K))$, gdzie $ext(K)$ oznacza typ określony w klauzuli dziedziczenia dla K (czyli w klauzuli postaci (1) lub (2)), zaś $decl(K)$ oznacza klasę, w której bezpośrednio (tekstowo) jest zadeklarowana klasa K
2. relacja zależności klas (odzwierciedlająca zależności w strukturze dziedziczenia) nie zawiera cyklu (dzięki czemu definicja żadnej klasy nie zależy od niej samej).

Uważam, że tak postawione zadania badawcze jest niebanalne i dobrze umotywowane. Z pewnością może stanowić podstawę dla rozprawy doktorskiej.

1.3 ZAWARTOŚĆ ROZPRAWY

Kolejne rozdziały rozprawy zawierają:

- wstęp, sformułowanie problemu i motywacje dla jego podjęcia
- algorytm niedeterministyczny i jego analizę
- algorytm deterministyczny, jego analizę i aspekty implementacyjne, dotyczące efektywności, reakcji na błędy i ich sygnalizacji
- dyskusję wcześniejszych rozwiązań, pokazującą że Autor rozprawy zapoznał się z nimi bardzo rzetelnie, w szczególności przedstawiona została dogłębna analiza propozycji Igarash'ego i Pierce'a (pozycja [11] w literaturze cytowanej w rozprawie)
- dyskusję wiązania statycznego i dynamicznego, wykonywanego w czasie obliczeń
- podsumowanie i konkluzje.

Szereg dyskusji i rozwiązań opartych jest na pracach, których M. Warpechowski jest współautorem, w tym szczególnie pozycji zacytowanych w rozprawie jako [16, 17, 18, 19] i - w mniejszym stopniu - pozycji [21, 22, 23].

2 OCENA ROZPRAWY

2.1 ORYGINALNY WKŁAD AUTORA PRACY

Oryginalnym wkładem Autora jest:

- formalne zdefiniowanie funkcji *bind* oraz *inh*, w oparciu o wyspecyfikowaną w rozprawie strukturę $\mathcal{S}_P$ (por. str. 8), odzwierciedlające i wyjaśniające bardzo zagmatwane definicje semantyki języka Java
- podanie algorytmów dla obliczania *bind* oraz *inh* i przeprowadzenie ich analizy
- dyskusja przytoczonych rozwiązań.

Analiza algorytmu niedeterministycznego dotyczy jego poprawności i jednoznaczności wyniku. W przypadku algorytmu deterministycznego została również pokazana jego złożoność. Ponadto Autor przeprowadził eksperymenty obliczeniowe.

2.2 NAJCIEKAWSZE WYNIKI

Do najciekawszych wyników zawartych w rozprawie zaliczam podanie klarownej definicji funkcji *bind* oraz algorytm deterministyczny i jego analizę (rozdział 4). Algorytm ten - poza rozwiązaniem postawionego problemu - dostarcza szeregu praktycznych mechanizmów związanych z obsługą i sygnalizacją błędów. Odpowiednia reakcja na błędy pozwala na kontynuację analizy programu przez kompilator, ma więc ważne znaczenie praktyczne. Algorytm jest niebanalny i analiza jego poprawności jest nietrywialna. Przebiega ona poprzez kilka lematów pokazujących niezmienniki spełnione w różnych momentach obliczeń.

2.3 UWAGI KRYTYCZNE

2.3.1 UWAGI MERYTORYCZNE

1. Nie dostrzegłem motywacji dla umieszczenia w rozprawie algorytmu niedeterministycznego. Wprawdzie był on ważnym krokiem na drodze do uzyskania algorytmu deterministycznego, ale jego rola w rozprawie wydaje się głównie historyczna zwłaszcza w kontekście zaprezentowanego algorytmu deterministycznego.
2. Rozdział 6, ważny z punktu widzenia zastosowania proponowanych rozwiązań, jest nieco zbyt pobieżnie napisany. Dyskutuje ważną problematykę semantyki programów, widzianą nie tylko poprzez statyczną analizę ich tekstów, ale także w trakcie obliczeń. Autor porusza tu szereg problemów związanych z semantyką zmiennych, czy generowania obiektów, jednak przedstawiona dyskusja pozostawia pewien niedosyt.

3. W pracy nie ma nawiązań do koncepcji interfejsów Javy i przyjętego w nich wielodziedziczenia. Nie bardzo widać na ile koncepcja ta ma wpływ na rozwiązania zaproponowane w rozprawie i możliwość ich stosowania gdy program korzysta z tych mechanizmów.

4. Z punktu widzenia przyszłej popularyzacji wyników przydałyby się także przykłady wzięte z praktyki programistycznej, ilustrujące napotkane problemy semantyczne.

2.3.2 UWAGI REDAKCYJNE

Praca na ogół jest zredagowana starannie. Przygotowane z dużą dbałością i dobrze przemyślane rysunki bardzo ułatwiają czytanie i rozumienie trudnych fragmentów rozprawy. Zdarzają się jednak drobne niedociągnięcia, jak np.:

1. nie pojawia się dokładna definicja sygnatury funkcji *inh* (podobna do sygnatury *bind* podanej na str 9)

2. punkty w definicjach pojawiają się w środku tekstu, co zmniejsza komfort czytania rozprawy (por. np. warunek I_1 w definicji 2.1.5)

3. zdarzają się drobne niezręczności językowe

4. pozycje [16] i [20] w cytowanej literaturze są identyczne.

3 PODSUMOWANIE I WNIOSEK

Bez wątpienia przedstawiona rozprawa wymagała dużych umiejętności badawczych i znajomości nietrywialnej tematyki, a także pokazała rzetelne opanowanie warsztatu naukowego przez doktoranta, w tym biegłe poruszanie się po zagadnieniach badanej dziedziny.

Biorąc pod uwagę wartość merytoryczną rozprawy i jakość uzyskanych w niej wyników, stwierdzam, że spełnia ona w pełni kryteria stawiane przed rozprawami doktorskimi i wnoszę o dopuszczenie jej do dalszych faz przewodu doktorskiego.

A. SZAŁAS