dr hab. inż. Remigiusz Wiśniewski, prof. UZ
Instytut Sterowania i Systemów Informatycznych
Uniwersytet Zielonogórski

Zielona Góra, 09.12.2021

# Recenzja rozprawy doktorskiej mgra inż. Michała Wypycha
## *Methods of generation of transition systems for Alvis language*

### 1. Obszar problemowy rozprawy

Praca doktorska Pana mgr. inż. Michała Wypycha wpisuje się w aktualny nurt modelowania, analizy oraz weryfikacji systemów współbieżnych. Systemy tego typu mogą zostać formalnie opisane z zastosowaniem języka Alvis, który w przejrzysty sposób umożliwia specyfikację oraz analizę modelu w formie graficznej. W ramach rozprawy Doktorant zaproponował szereg nowatorskich algorytmów i metod usprawniających proces testowania oraz analizy modeli opisanych za pomocą języka Alvis. W tym celu Autor zastosował pośrednią reprezentację systemu za pomocą języka Haskell. Ponadto, opracowany został autorski algorytm umożliwiający wyznaczenie etykietowanego systemu przejść dla systemu modelowanego w języku Alvis.

Uważam, że temat recenzowanej rozprawy doktorskiej Pana mgr. inż. Michała Wypycha jest interesujący, aktualny i ważny technicznie. Teza oraz cele naukowe pracy zostały precyzyjne sformułowane, a stopień złożoności, znaczenie naukowe i zakres zadań odpowiadają ustawowym i zwyczajowym wymogom stawianym rozprawie doktorskiej.

### 2. Ocena merytoryczna rozprawy

Rozprawa stanowi spójną tematycznie całość. Składa się z 7 rozdziałów wraz ze wstępem i podsumowaniem. Zasadniczą część rozprawy stanowią rozdziały 4-6 poświęcone zaproponowanej autorskiej metodzie automatycznego generowania etykietowanego systemu przejść.

Rozdział 1 przedstawia motywację podjęcia tematu, formułuje hipotezę, cele oraz zakres badawczy rozprawy. Teza oraz zadania pracy zostały precyzyjnie sformułowane. Czytelnik nie ma wątpliwości, co Autor chce zbadać i jakie metody naukowo-badawcze zastosować.

Rozdział 2 przedstawia aktualny stan wiedzy. Doktorant dokonał bardzo rzetelnego przeglądu innych, alternatywnych narzędzi oraz języków modelowania. Na szczególne podkreślenie zasługuje gruntowna analiza, w której Autor wypunktował nie tylko zalety i wady innych rozwiązań, ale potrafił także wskazać na ich tle zalety i ograniczenia języka Alvis. Moim zdaniem świadczy to o uczciwym i rzetelnym podejściu do tematu rozpatrywanego w rozprawie i znacząco podnosi wartość całego doktoratu.

Wprowadzenie do języka Alvis zawarto w rozdziale 3. Przedstawiono poszczególne warstwy języka, omówiono modele formalne, a także powiązano Alvis z etykietowanym systemem przejść (ang. *„Labeled Transition System", LTS*). Niewielkim mankamentem utrudniającym lekturę pracy jest drobiazgowe zestawienie reguł („rules"), które zajmują aż 10 kolejnych stron. W mojej opinii wystarczyło je ogólnie omówić, ewentualnie opisać reprezentatywne przykłady (pozostałe umieścić np. w dodatku).

Rozdział 4 poświęcono autorskiej metodzie reprezentacji modelowanego systemu z zastosowaniem języka Haskell. Autor podjął się bardzo ambitnego zadania, polegającego na opracowaniu szeregu algorytmów i metod umożliwiających transformację modelu opisanego w języku Alvis do języka Haskell (wymiernym efektem jest praktyczna implementacja metod w postaci kompilatora). Poszczególne podrozdziały drobiazgowo omawiają kolejne aspekty, a prezentowane algorytmy zilustrowano poglądowymi przykładami. Na podkreślenie zasługuje wyjątkowa precyzja i skrupulatność Autora, czego rezultatem jest rzadko spotykana obszerność rozdziału (66 stron). Niemniej, w tym przypadku uważam, że dokładna prezentacja oraz omówienie autorskich metod i algorytmów jest w pełni uzasadnione.

Rozdział 5 prezentuje autorski mechanizm generowania etykietowanego systemu przejść. Algorytm jest bardzo wnikliwe omówiony, choć w mojej opinii zabrakło przedstawienia ogólnej koncepcji, np. w formie pseudo-kodu (pseudo-algorytmu), co ułatwiłoby zrozumienie prezentowanych zagadnień.

Rozdział 6 przedstawia wyniki ewaluacji algorytmu zaproponowanego w poprzednim rozdziale. Autor gruntownie omówił zastosowaną metodykę weryfikacji, zaprezentował wyniki testów, które zostały poparte wnikliwą analizą. Dodatkowym, niewątpliwym atutem tego rozdziału są informacje odnośnie napotkanych problemów oraz zastosowanych (zaproponowanych) przez Autora rozwiązań.

Pracę kończy rozdział 7, w którym Autor podsumował rozprawę oraz przedstawił możliwe kierunki dalszych prac.

Za najważniejsze, oryginalne elementy rozprawy uważam:

- Zaproponowanie autorskiej koncepcji reprezentacji systemu modelowanego w języku Alvis z zastosowaniem języka Haskell.
- Opracowanie szeregu algorytmów i metod umożliwiających transformację modelu opisanego w języku Alvis do języka Haskell.
- Opracowanie autorskiego algorytmu umożliwiającego wyznaczenie etykietowanego systemu przejść dla systemu modelowanego w języku Alvis.
- Opracowanie oraz realizacja kompilatora dla języka Alvis, umożliwiającego automatyczną konwersję do języka Haskell.
- Opracowanie modelów testowych w celu ewaluacji zaproponowanych algorytmów.
- Analiza skuteczności oraz sprawności opracowanego algorytmu generowania etykietowanego systemu przejść.

Podsumowując ten fragment recenzji stwierdzam, że cele pracy zostały osiągnięte.

Rozprawa doktorska ma również swoje słabe strony i pewne niedociągnięcia, które podzieliłem na dwie części, grupując je w postaci uwag krytycznych oraz uwag szczegółowych. Jednocześnie chciałbym wyraźnie zaznaczyć, że sekcja „uwagi krytyczne" ma charakter dyskusyjny i dotyczy moich pytań oraz wątpliwości, które pojawiły się w trakcie recenzji rozprawy.

## 3. Uwagi krytyczne, wątpliwości, pytania

1. Uwaga ogólna: praca jest napisana poprawnie i spójnie, jednakże układ rozdziałów oraz sposób prezentacji zagadnień jest moim zdaniem w niektórych miejscach chaotyczny. Przykładowo, rozdział pierwszy (wprowadzenie) zawiera aż 11 podrozdziałów, wydaje się, że część z nich z powodzeniem można zamieścić w rozdziale 2. Niemniej, chciałbym w tym miejscu wyraźnie podkreślić, że jest to uwaga mocno subiektywna, która absolutnie nie obniża mojej ogólnej, wysokiej oceny pracy.
2. Rozdział 1.2: autor omawia sposoby oraz metody testowania, nawiązując do technik weryfikacji („software verification", „(...) product to be verified though testing (...)"). Mam tu spore wątpliwości czy sposoby i techniki testowania wymienione na stronach 4-5 dotyczą tylko weryfikacji, czy może także walidacji systemu? Natomiast wydaje mi się, że słowo „testowanie" jest tu użyte jak najbardziej poprawnie.
3. Rozdział 1.6, pierwsze równanie na dole strony 13: Autor pisze o „formalnym" „zdefiniowaniu" pojęcia bezpieczeństwa, jednakże pominięto wyjaśnienie oznaczeń użytych w definicji, przez co jest ona niezrozumiała. Co oznaczają poszczególne symbole? Ponadto, skoro jest to definicja formalna, powinna także zostać odpowiednio (formalnie) oznaczona i ponumerowana. Podobnie, w przypadku kolejnej definicji (żywotności) – tu także brakuje formalnych oznaczeń oraz numeracji. Warto także zauważyć, że logika temporalna użyta w definicjach jest wprowadzona dopiero w kolejnym rozdziale (1.7), wobec czego wydaje się, że te rozdziały powinny zostać zamienione kolejnością. Poza tym mocno wskazany byłby ogólny wykaz oznaczeń i skrótów użytych w pracy.
4. To bardziej pytanie niż uwaga: w rozdziale 1.3. przedstawiono problemy związane ze spójnością, dostępnością oraz partycjonowaniem systemu współbieżnego, które wydają się być bardzo mocno powiązane się z twierdzeniem CAP („consistency", „availability", „partition tolerance"). Czy to twierdzenie (CAP) odnosi się w jakikolwiek sposób do metod proponowanych w doktoracie i czy ma zastosowanie?
5. Strona 43: stwierdzenie „Main differences between Alvis and more classical formal methods, like Petri nets and process algebras, include the syntax that is *more user-friendly* from engineers' point of view (..)" jest autorytatywne i dyskusyjne. W jaki sposób język Alvis jest „bardziej przyjazny dla użytkownika" („more user friendly") niż np. sieci Petriego? Jaka jest podstawa takiego założenia? Czy stwierdzenie jest poparte odpowiednimi badaniami/analizą, że tak jest rzeczywiście? Wydaje mi się, że to sformułowanie jest w ogóle mocno niefortunne, gdyż sieć Petriego nie jest językiem

modelowania, a jedynie aparatem (narzędziem, sposobem) umożliwiającym opis systemu (specyfikację) na bardzo zróżnicowanym poziomie abstrakcji (począwszy od ogólnych modeli graficznych bez jednoznacznego powiązania formalnego, poprzez specyfikacje formalne, aż po precyzyjne opisy systemów oraz procesów z zastosowaniem sygnałów oraz tzw. sieci interpretowanych), który znajduje zastosowanie w najprzeróżniejszych dziedzinach nauki/przemysłu. Natomiast Alvis jest językiem modelowania, który z natury jest formalny, posiada precyzyjną (i jednoznaczną) składnię, warstwy („layers"), kompilator, itp.

6. Strona 47: "Splitting the system into smaller parts is one of the most basic concepts in software engineering. Fragments of functionality can be split into their respective modules, which encapsulate the separated pieces of logic. Then, working on a particular module does not require immediate consideration of the implications of the work on other parts of the system. Modularisation is invaluable for working efficiently. There are many other benefits of breaking the system into smaller modules, e.g. the model is more maintainable, testable and reusable. Moreover, such a module can be used as a reusable component."
   Powyższe zdania są z jednej strony stanowcze, a z drugiej mocno ogólne i – moim zdaniem – niekoniecznie zawsze adekwatne w kontekście modelowanego systemu:
   - „Splitting the system into smaller parts is one of the most basic concepts in software engineering" – co Autor rozumie poprzez "basic concepts in software engineering"?
   - „Modularisation is invaluable for working efficiently" – bardzo mocne stwierdzenie; dlaczego Autor uważa, że modularyzacja jest „invaluable" (niezbędna, nieoceniona) do wydajnej pracy?
   - Co Autor rozumie poprzez: „model is more maintainable, testable"? Np. w przypadku testowania – istotnie – modularyzacja umożliwia testowanie pojedynczych modułów, ale docelowo należy przeprowadzić testowanie całego systemu?
7. Algorytm przedstawiony w rozdziale 5: w mojej opinii brakuje przedstawienia ogólnej koncepcji, tzn. pseudo-kodu (pseudo-algorytmu). To znacznie ułatwiłoby zrozumienie prezentowanych zagadnień. Autor pisze, że celem jest „optymalny algorytm" eksploracji grafu. Następnie pokazane są znane metody przeszukwiania grafu (DFS, BFS), na podstawie których określona jest złożoność obliczeniowa (także ogólnie znana). Wobec tego pojawia się pytanie: jaki jest cel szacowania złożoności obliczeniowej tego typu algorytmów? Dalej: w jaki sposób algorytmy DFS/BFS pozwalają na znalezienie „optymalnego" rozwiązania? Co jest uznane za kryterium „optymalności"? Czy analizowany graf jest szczególnego typu (np. należy do klasy grafów doskonałych – np. grafów porównywalności „comparability graphs"), skoro ogólnie znane algorytmy DFS/BFS potrafią znaleźć optymalne rozwiązanie?

## 4. Uwagi szczegółowe

1. Uwaga ogólna: od strony językowej (angielski) praca została – w mojej opinii - przygotowana bardzo dobrze. Wprawdzie w kilku miejscach dostrzegłem drobne błędy językowe czy gramatyczne, jednakże są to niuanse absolutnie nie wpływające na wartość pracy, dlatego też pomijam w zestawieniu tego typu uwagi.
2. W rozprawie pojawiają się stanowcze sformułowania typu „IT systems are currently in everyday use in every branch of industry". To prawda, że systemy tego typu są powszechnie stosowane w przemyśle, ale nie „everyday" i nie „in every branch" – wg mnie to zbyt stanowcze określenie.
3. Brak rozwinięcia skrótów (np. „IT", „ISO", „LOTOS", itp.).
4. Standard „IEEE Std 1012-2012" wymieniony na stronie 3 jako „existing standard" nie jest już aktualny, został zastąpiony przez „IEEE Std 1012-2016".
5. W literaturze zdecydowanie przydałyby się odnośniki (linki URL) do pozycji, zwłaszcza tych ogólnodostępnych w Internecie (np. standardów – patrz uwaga wyżej).
6. Strony 8, 81, 208 słowo „naive" jest napisane z podwójną kropką (nad „i").
7. Na stronie 11 pojawia się odniesienie do pojęcia żywotności („liveness"), które zostaje wyjaśnione dopiero w kolejnym podrozdziale.
8. Rozdział 1.5 omawia algorytmy nieblokujące „non-blocking algorithms", są informajce o korzyściach, sposobach implementacji, jednakże brakuje podstawowych informacji, co to w ogóle jest za rodzaj algorytmów.
9. Rozdział 1.6: sformułowanie „Hoare's logic" nie zostało zdefiniowane ani wyjaśnione.
10. Strona 29: „Fire is related ..." -> „Transition firing is related ...".
11. Strona 31: zdanie „Modelling language allows..." jest nieprecyzyjnie sformułowane, z którego wynika, że każdy język modelowania pozwala na specyfikację systemu jako kompozycja modułów sekwencyjnych. Brakuje tu jawnego odniesienia (wskazania) do NuXmv, lub nawiązania do poprzedniego zdania (np. „Such a modelling language...").
12. Strona 45, rysunek 3.2 jest niewyraźny.

Należy jednoznacznie stwierdzić, że powyższe uwagi szczegółowe wyspecyfikowano z opiniodawczego obowiązku i nie obniżają one wartości merytorycznej rozprawy.

## 5. Ocena końcowa rozprawy

Podsumowując stwierdzam, że rozprawa doktorska Pana mgr. inż. Michała Wypycha stanowi oryginalne rozwiązanie problemu naukowego, a autorskie osiągnięcia przedstawione w rozprawie wnoszą istotny wkład w rozwój dyscypliny informatyka techniczna i telekomunikacja. Recenzowana rozprawa spełnia warunki określone w art. 13 ust. 1 ustawy z dnia 14 marca 2003 r. *o stopniach naukowych i tytule naukowym oraz o stopniach i tytule w zakresie sztuki*, wobec czego wnoszę o przyjęcie rozprawy doktorskiej i dopuszczenie jej do publicznej obrony.

Remigiusz Wiśniewski