Mathematisch-Naturwissenschaftliche Fakultät
Der Dekan

HEINRICH HEINE
UNIVERSITÄT DÜSSELDORF

Neue Promotionsordnung

An alle
hauptamtlichen Professoren/innen
und Privatdozenten/innen
des Faches Informatik der
Mathematisch-Naturwissenschaftlichen Fakultät

Promotionsangelegenheiten
Universitätsstr. 1
40225 Düsseldorf
Telefon: (0211) 81-15092
Telefax: (0211) 81-15090
E-Mail: promotion@mnf.uni-duesseldorf.de

14.05.2018

Promotionsverfahren von **Herrn M.Sc. John Witulski**
Auslage der Dissertation und Gutachten sowie Termin der mündlichen Prüfung
Anlage: Einseitige Zusammenfassung der Dissertation

Sehr geehrte Damen und Herren,

in dem oben genannten Promotionsverfahren wird die Annahme der Dissertation

A Python B Implementation - PyB A Second Tool-Chain

von den Berichterstattern Prof. Dr. M. Leuschel und Prof. Dr. M. Schöttner beantragt. Sie kann zusammen mit den Gutachten in der Zeit

vom 26.05.2018 bis 06.06.2018

im Promotionsbüro (Gebäude 25.32, Ebene 00, Raum 36) zu den Sprechzeiten eingesehen werden.

Einsprüche gegen diese Dissertation können nur zwei Tage nach der vorgenannten Frist geltend gemacht werden. Erfolgt kein Einspruch, so gilt die Dissertation als angenommen (§ 7 Ziffer (5) PO).

Sofern die Dissertation angenommen wird, findet die mündliche Prüfung am

11.06.2018 um 11:00 Uhr

im **im Hörsaal 5 A** statt. Als Prüfer sind vorgesehen:
Prof. Dr. G. Klau, Prof. Dr. S. Harmeling und Prof. Dr. E. Wanke.

Zuhörer sind bei der Befragung zugelassen.

Mit freundlichen Grüßen
im Auftrag

A. Stefanidou
Athina Stefanidou

Zusammenfassung

Diese Dissertation befasst sich mit den Themen Interpreterbau, Just-In-Time Kompilation und Software-Werkzeuge für die Spezifikationssprache B. Behandelt wurden zwei unterschiedliche Themen: Das Thema der zweiten Kette und das Thema des RPython-JIT. Beide Themen erfordern eine Implementierung von B. Bei dieser Implementierung handelt es sich um PyB, einen Interpreter und Modelprüfer geschrieben in Python. Der **Beitrag** dieser Arbeit ist PyB und die damit durchgeführten Experimente.

1. Zusammenfassung zweite Kette: Die **erste Problemstellung** ist die Entwicklung und Untersuchung einer zweiten Kette (PyB). Dies ist ein Software-Werkzeug, welches die Berechnungen eines anderen Werkzeuges (ProB) überprüft. Von Interesse war, wie bestimmte Aspekte der Sprache B besonders einfach implementiert werden können, für welche Aspekte dies nicht möglich ist und die Bestimmung von Nützlichkeit und Grenzen dieses Ansatzes. **Motivation** dieser Fragestellungen ist die Notwendigkeit einer zusätzlichen Überprüfung von ProB, einem Software-Werkzeug für die B-Methode, welche zur Entwicklung und Modellierung im Bereich sicherheitskritische Software eingesetzt wird, wo Zuverlässigkeit unabdingbar ist. Die zweite Kette ist ein Ansatz, der diese Zuverlässigkeit erhöhen soll. **Methode** der Entwicklung war die einer Cleanroom-Implementierung, welche fordert, dass PyB seine Berechnungen völlig unabhängig vom ersten Werkzeug (ProB) durchführt: Kein ProBCode ist bekannt. Die Grundannahme ist, dass es einfach ist, ein simpleres Werkzeug zu entwickeln um ein komplexes zu testen. Im **Ergebnis** hat sich diese Annahme nur als teilweise richtig herausgestellt. Sie ist falsch, wenn PyB zur Überprüfung einer Lösung selbst Werte finden muss. Dies ist in B in einigen Fällen unausweichlich. Simpel war die B-Implementierung mit Ausnahme von unendlichen Mengen und constraint solving. Die Implementierung unterstützt den vollen B Sprachumfang und wurde erfolgreich mit industriellen Maschinen getestet. Im **Fazit** ist dieser Ansatz für B nur nützlich, wenn die zweite Kette nicht selbst Lösungen finden muss, da diese sonst selbst zu einem komplexen fehleranfälligen Werkzeug wird.

2. Zusammenfassung RPython-JIT: Die **zweite Problemstellung** ist die Anwendung der RPython Technologie auf eine B-Implementierung. Hierbei ist die Anpassung des PyB-Quellcodes an die RPython Anforderungen (eine statische Python Untermenge) sowie das Hinzufügen eines JITs gemeint. Dieses wurde bisher nur auf Implementierungen von dynamischen Programmiersprachen angewandt. **Motivation** ist es, die Übertragbarkeit von bereits bestehenden Performanceergebnissen auf eine Spezifikationssprache wie B zu überprüfen. Da das Ziel, ein simples Werkzeug mit dem Ziel ein performantes Werkzeug zu schreiben, im Konflikt steht, wurde hierbei an zwei unterschiedlichen braches entwickelt. Überprüft wurde die Performance mit der **Methode** der Micro-Benchmarks und Benchmarks bestehend aus industriellen Beispielen. Im **Ergebnis** hat sich für das Modelprüfen von simplen Modellen ein speed up von einer Größenordnung (im Vergleich zu ProB) ergeben, während bei Beispielen welche constraint solving erfordern, die Performance um Größenordnungen schlechter sein kann. Als **Fazit** wird die Anwendung von der RPython-JIT Technologie auf B dennoch als nützlich bewertet. Wenn PyB um komplexe Features wie constraint solving erweitert wird, sind auch hier bessere Ergebnisse zu erwarten.