Akademia Górniczo-Hutnicza im. Stanisława Staszica w Krakowie

WYDZIAŁ ELEKTROTECHNIKI, AUTOMATYKI, INFORMATYKI I INŻYNIERII BIOMEDYCZNEJ

KATEDRA INFORMATYKI STOSOWANEJ

Praca dyplomowa inżynierska

Implementacja maszyny wirtualnej dla funkcyjnych języków programowania wspierających przetwarzanie współbieżne.

Implementation of a virtual machine for functional programming languages with support for concurrent computing.

Autor:	*Kajetan Rzepecki*
Kierunek studiów:	*Informatyka*
Opiekun pracy:	*dr inż. Piotr Matyasik*

Kraków, 2013

Oświadczam, świadomy odpowiedzialności karnej za poświadczenie nieprawdy, że niniejszą pracę dyplomową wykonałem osobiście i samodzielnie i nie korzystałem ze źródeł innych niż wymienione w pracy.

Serdecznie dziękuję opiekunowi pracy za wsparcie merytoryczne oraz dobre rady edytorskie pomocne w tworzeniu pracy.

Spis treści

1. Wstęp

Celem pracy i powiązanego z nią projektu jest implementacja oraz ewaluacja maszyny wirtualnej dla funkcyjnych języków programowania, które umożliwiają przetwarzanie współbieżne wykorzystując Model Aktorowy oraz asynchroniczne przekazywanie wiadomości.

Pubsy [HBS73], [Cli81], [FW87], [Ram99], [ST03], [VHM10], [Wil05], [BCR04], [AV95], [HW98], [Coo], [AS96], [Gud93], [GPP+12], [SJS78], [JL92], [LMS04], [HLM+02], [KP11], [Gro08], [MS96], [Boe04], [SBF12], [KPR+92].

Rysunek 1.1: Logo AGH

Rysunek 1.2: Logo inne

Tablica 1.1: Tabelka jakaś.

Tabelka	Hurr
Hurr	durr
Herp	derp

Tablica 1.2: Tabelka inna.

Tabelka	Hurr
Hurr	durr
Herp	derp

1.1. Problemy przetwarzania współbieżnego

– Opisać problemy Erlanga,

1.2. Próby rozwiązania problemu

– opisać próby ich rozwiązania w Erlangu,

1.3. Cel i zawartość pracy

– opisać proponowany sposób ich rozwiązania,

– umotywować powstanie ThesisVM.

2. Projekt i implementacja ThesisVM

TODO: Opisać ogólną strukturę maszyny wirtualnej (z GC i SMP) i opisać o czym będzie niniejsza sekcja.

2.1. Reprezentacja pośrednia programów

2.2. Reprezentacja prostych obiektów ThesisVM

2.3. Reprezentacja obiektów funkcyjnych ThesisVM

2.4. Reprezentacja kodu bajtowego ThesisVM

2.5. Ewaluacja argumentów i aplikacja funkcji

2.6. Operacje arytmetyczne

2.7. Implementacja wbudowanych operatorów

2.8. Kompilator kodu bajtowego ThesisVM

Opisać pipeline kompilatora.

3. Model zarządzania pamięcią

4. Model przetwarzania współbieżnego

4.1. Model Aktorowy

4.2. Notacja procesu w ThesisVM

4.3. Harmonogramowanie procesów

4.4. Przesyłanie wiadomości

5. Podsumowanie i analiza wydajności ThesisVM

Przeanalizować wydajność GC i SMP.

5.1. Leniwe zliczanie referencji

Przeanalizować szybkość, pauzy, zużycie pamięci.

5.2. Przesyłanie wiadomości

Przeanalizować szybkość przesyłania wiadomości/konieczność czekania procesów/wątków.

5.3. Porównanie szybkości działania ThesisVM

Porównać kilka implementacji prostych programów (z Haskell'em, leniwym Lispem itp).

Bibliografia

[AS96] Harold Abelson and Gerald J. Sussman. *Structure and Interpretation of Computer Programs*. MIT Press, Cambridge, MA, USA, 2nd edition, 1996.

[AV95] Joe Armstrong and Robert Virding. One pass real-time generational mark-sweep garbage collection. In *IN INTERNATIONAL WORKSHOP ON MEMORY MANAGEMENT*, pages 313–322. Springer-Verlag, 1995.

[BCR04] David F. Bacon, Perry Cheng, and V. T. Rajan. A unified theory of garbage collection. In *Proceedings of the 19th Annual ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications*, OOPSLA '04, pages 50–68, New York, NY, USA, 2004. ACM.

[Boe04] Hans-J. Boehm. The space cost of lazy reference counting. In *Proceedings of the 31st ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages*, POPL '04, pages 210–219, New York, NY, USA, 2004. ACM.

[Cli81] William D Clinger. Foundations of actor semantics. Technical report, Cambridge, MA, USA, 1981.

[Coo] William R. Cook. Anatomy of programming languages. Free online version.

[FW87] John Fairbairn and Stuart Wray. Tim: A simple, lazy abstract machine to execute supercombinators. In *Proc. Of a Conference on Functional Programming Languages and Computer Architecture*, pages 34–45, London, UK, UK, 1987. Springer-Verlag.

[GPP+12] Colin S. Gordon, Matthew J. Parkinson, Jared Parsons, Aleks Bromfield, and Joe Duffy. Uniqueness and reference immutability for safe parallelism. In *Proceedings of the ACM International Conference on Object Oriented Programming Systems Languages and Applications*, OOPSLA '12, pages 21–40, New York, NY, USA, 2012. ACM.

[Gro08] Lindsay Groves. Verifying michael and scott's lock-free queue algorithm using trace reduction. In *Proceedings of the Fourteenth Symposium on Computing:*

The Australasian Theory - Volume 77, CATS '08, pages 133–142, Darlinghurst, Australia, Australia, 2008. Australian Computer Society, Inc.

[Gud93] David Gudeman. Representing type information in dynamically typed languages, 1993.

[HBS73] Carl Hewitt, Peter Bishop, and Richard Steiger. A universal modular actor formalism for artificial intelligence. In *Proceedings of the 3rd International Joint Conference on Artificial Intelligence*, IJCAI'73, pages 235–245, San Francisco, CA, USA, 1973. Morgan Kaufmann Publishers Inc.

[HLM+02] Maurice Herlihy, Victor Luchangco, Paul Martin, Mark Moir, Dynamic sized Lockfree, Data Structures, Maurice Herlihy, Victor Luchangco, Paul Martin, and Mark Moir. Dynamic-sized lockfree data structures. Technical report, 2002.

[HW98] Lorenz Huelsbergen and Phil Winterbottom. Very concurrent mark-&-sweep garbage collection without fine-grain synchronization. In *Proceedings of the 1st International Symposium on Memory Management*, ISMM '98, pages 166–175, New York, NY, USA, 1998. ACM.

[JL92] Simon Peyton Jones and David Lester. *Implementing functional languages: a tutorial*. Prentice Hall, 1992. Free online version.

[KP11] Alex Kogan and Erez Petrank. Wait-free queues with multiple enqueuers and dequeuers. In *Proceedings of the 16th ACM Symposium on Principles and Practice of Parallel Programming*, PPoPP '11, pages 223–234, New York, NY, USA, 2011. ACM.

[KPR+92] Owen Kaser, Shaunak Pawagi, C. R. Ramakrishnan, I. V. Ramakrishnan, and R. C. Sekar. Fast parallel implementation of lazy languages - the equals experience. In *Journal of Functional Programming*, pages 335–344. ACM, 1992.

[LMS04] Edya Ladan-Mozes and Nir Shavit. An optimistic approach to lock-free fifo queues, 2004.

[MS96] Maged M. Michael and Michael L. Scott. Simple, fast, and practical non-blocking and blocking concurrent queue algorithms. In *Proceedings of the Fifteenth Annual ACM Symposium on Principles of Distributed Computing*, PODC '96, pages 267–275, New York, NY, USA, 1996. ACM.

[Ram99] John D. Ramsdell. The Tail-Recursive SECD Machine. *Journal of Automated Reasoning*, 23(1):43–62, 1999.

[SBF12] Rifat Shahriyar, Stephen M. Blackburn, and Daniel Frampton. Down for the count? getting reference counting back in the ring. In *Proceedings of the 2012 International Symposium on Memory Management*, ISMM '12, pages 73–84, New York, NY, USA, 2012. ACM.

[SJS78] Guy Lewis Steele Jr and Gerald Jay Sussman. The art of the interpreter of the modularity complex (parts zero, one, and two). 1978.

[ST03] Håkan Sundell and Philippas Tsigas. Fast and lock-free concurrent priority queues for multi-thread systems. In *Proceedings of the 17th International Symposium on Parallel and Distributed Processing*, IPDPS '03, pages 84.2–, Washington, DC, USA, 2003. IEEE Computer Society.

[VHM10] David Van Horn and Matthew Might. Abstracting abstract machines. In *Proceedings of the 15th ACM SIGPLAN International Conference on Functional Programming*, ICFP '10, pages 51–62, New York, NY, USA, 2010. ACM.

[Wil05] Jesper Wilhelmsson. *Efficient Memory Management for Message-Passing Concurrency — part I: Single-threaded execution.* Licentiate thesis, Department of Information Technology, Uppsala University, May 2005.

A. Wizualizacja stanu maszyny wirtualnej

Opisać narzędzie do rysowania grafów stanu.

B. Przykładowe programy

Dać kilka przykładów prostych programów razem z grafami stanów.

C. Spisy rysunków i tablic

Spis rysunków

Spis tablic