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

– Opisać temat i cel pracy.

1.1. Motywacja pracy

– Opisać problemy Erlanga,

– Umotywować powstanie ThesisVM,

1.2. Zawartość pracy

– Opisać zakres pracy.

– Opisać zawartość poszczególnych rozdziałów.

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

[1] R. Carlsson, "An introduction to Core Erlang," in *In Proceedings of the PLI'01 Erlang Workshop*, 2001.

[2] R. Carlsson, B. Gustavsson, E. Johansson, T. Lindgren, S.-O. Nyström, M. Pettersson, and R. Virding, "Core Erlang 1.0.3 language specification," tech. rep., Department of Information Technology, Uppsala University, Nov. 2004.

[3] J. Fairbairn and S. Wray, "TIM: A simple, lazy abstract machine to execute supercombinators," in *Proc. Of a Conference on Functional Programming Languages and Computer Architecture*, (London, UK, UK), pp. 34–45, Springer-Verlag, 1987.

[4] J. D. Ramsdell, "The Tail-Recursive SECD Machine," *Journal of Automated Reasoning*, vol. 23, no. 1, pp. 43–62, 1999.

[5] D. Van Horn and M. Might, "Abstracting abstract machines," in *Proceedings of the 15th ACM SIGPLAN International Conference on Functional Programming*, ICFP '10, (New York, NY, USA), pp. 51–62, ACM, 2010.

[6] H. Abelson and G. J. Sussman, *Structure and Interpretation of Computer Programs*. Cambridge, MA, USA: MIT Press, 2nd ed., 1996.

[7] G. L. Steele Jr and G. J. Sussman, "The art of the interpreter of the modularity complex (parts zero, one, and two)," 1978.

[8] D. Gudeman, "Representing type information in dynamically typed languages," 1993.

[9] S. P. Jones and D. Lester, *Implementing functional languages: a tutorial*. Prentice Hall, 1992. Free online version.

[10] W. R. Cook, "Anatomy of programming languages." Free online version.

[11] O. Kaser, S. 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*, pp. 335–344, ACM, 1992.

[12] R. Shahriyar, S. M. Blackburn, and D. Frampton, "Down for the count? getting reference counting back in the ring," in *Proceedings of the 2012 International Symposium on Memory Management*, ISMM '12, (New York, NY, USA), pp. 73–84, ACM, 2012.

[13] H.-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, (New York, NY, USA), pp. 210–219, ACM, 2004.

[14] L. Huelsbergen and P. Winterbottom, "Very concurrent mark-&-sweep garbage collection without fine-grain synchronization," in *Proceedings of the 1st International Symposium on Memory Management*, ISMM '98, (New York, NY, USA), pp. 166–175, ACM, 1998.

[15] J. Armstrong and R. Virding, "One pass real-time generational mark-sweep garbage collection," in *IN INTERNATIONAL WORKSHOP ON MEMORY MANAGEMENT*, pp. 313–322, Springer-Verlag, 1995.

[16] D. F. Bacon, P. 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, (New York, NY, USA), pp. 50–68, ACM, 2004.

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

[18] C. Hewitt, P. Bishop, and R. Steiger, "A universal modular actor formalism for artificial intelligence," in *Proceedings of the 3rd International Joint Conference on Artificial Intelligence*, IJCAI'73, (San Francisco, CA, USA), pp. 235–245, Morgan Kaufmann Publishers Inc., 1973.

[19] W. D. Clinger, "Foundations of actor semantics," tech. rep., Cambridge, MA, USA, 1981.

[20] C. S. Gordon, M. J. Parkinson, J. Parsons, A. Bromfield, and J. 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, (New York, NY, USA), pp. 21–40, ACM, 2012.

[21] M. M. Michael and M. 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, (New York, NY, USA), pp. 267–275, ACM, 1996.

[22] L. 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, (Darlinghurst, Australia, Australia), pp. 133–142, Australian Computer Society, Inc., 2008.

[23] A. Kogan and E. 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, (New York, NY, USA), pp. 223–234, ACM, 2011.

[24] M. Herlihy, V. Luchangco, P. Martin, M. Moir, D. sized Lockfree, D. Structures, M. Herlihy, V. Luchangco, P. Martin, and M. Moir, "Dynamic-sized lockfree data structures," tech. rep., 2002.

[25] E. Ladan-Mozes and N. Shavit, "An optimistic approach to lock-free fifo queues," 2004.

[26] H. Sundell and P. 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, (Washington, DC, USA), pp. 84.2–, IEEE Computer Society, 2003.

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