University of Pretoria etd – Ketcha Ngassam, E (2007)

Part IV

Epilogue: Conclusion and Future Work

University of Pretoria etd – Ketcha Ngassam, E (2007)

CHAPTER 11

SUMMARY AND CONCLUSION

11.1 Summary

This work can be summarized as follows:

- In part I, we introduced basic elements of automata theory and computer architecture that served as basis for understanding various aspects of the thesis. In particular, the formal definition of a string recognizer in terms of its denotational semantics was given. The chapter also depicted the complex operational diagram of modern processors which showed some of the detail of how cache memory plays an important role in an algorithm's performance.

- Part II was devoted to the investigation of alternative algorithms for string recognition. To this end, we started off (in Chapter 3) by presenting the core TD and HC algorithms which led to the design of the core MM algorithm. Furthermore, the denotational semantics of the core algorithms was specified in terms of a single function. The next three chapters (Chapters 4, 5 & 6) of part II were devoted to investigations of new strategies for the implementation of DFA-based string recognizers. The function defining the recognizer's denotational semantics was then expanded to incorporate appropriate variables that specify whether or not, and —where relevant— the way in which, each of the respective strategies is to be deployed. In Chapter 7, a unified version of this denotational semantics function was suggested, taking into account all strategy variables discussed in previous chapters. The formalism resulted in the suggestion of a total of 168 algorithms. This was then used for the construction of a taxonomy tree whose nodes represent the different algorithms. The taxonomy was further mapped into a class-diagram (Chapter 8) that represented the architectural view for a toolkit of DFA-based string recognizers.

- In part III, attention was given on the implementation of some algorithms. An introductory note on the way our experiments were conducted as well as the software and hardware used for the experiments was given in Chapter 9. In Chapter 10, experimental results of some selected algorithms were discussed. In general, experiments conducted revealed that algorithms suggested throughout the thesis are of interest and could prove useful when processing particular kinds of strings. It was also shown in this part that each of the investigated algorithms could be processed at optimum, as long as the kind of string to

be processed reflects the algorithm's best case behaviour. Moreover, although it was already known that the core HC algorithm outperforms the core TD algorithm for automata of size in the order of hundreds, it was also shown that the MM algorithm could be used as a performance booster since it has been explicitly designed to take advantage of both HC and TD capabilities.

Over the past few years, various aspects of our work have been published in scientific journals, peer reviewed conferences, and workshops:

1. [NKW06a] represents the initial work that lead us to the investigation of new implementation strategies since it was established that there is a correlation between algorithm performance and cache memory capabilities.

2. In [NWK04, NWK05] various ideas were being shaped and we suggested a framework for the dynamic implementation of FAs whose original concept is still under investigation, and the notion of dynamic implementation resulted in the investigation of the DSA strategy whose early publication in [NKW05b] referred the strategy as *state reordering*, and an enhanced version in [NKW06b] kept the terminology DSA. The hardcoded version of the DSA strategy was published in [NKW06c].

3. Based on previous results, the idea of SpO and AVC strategies were investigated. Resulting in a formal characterization of string recognizers using the notion of denotational semantics. An early version of the idea was presented at a workshop ([Nga05]) where the notion of taxonomy was first suggested. Further improvements on the idea yielded the publication in [NKW06d] of a taxonomy of DFA-based string processors.

4. Finally, the performance of all the TD algorithms was published in [NWK06].

In the next section, we provide a conclusion to our work.

11.2 Conclusion

We believe that the followings have been achieved in this thesis:

- **Denotational Semantics of String Recognizers:** Our work has established the basic foundation for mathematically representing DFA-based string recognizers. Although our parametric function relied on the suggested implementation strategies, many other strategies could be investigated, resulting therefore to more algorithms, and of course several challenges for solving FA-based string processing problems.

- **New DFA-recognition algorithms:** To date, DFA-based string recognition has been limited to two alternative solutions: the core TD and HC algorithms. Our work has provided for up to 166 different algorithms.

- **Knowledge of hardware:** Although the design of optimal algorithms requires sound theoretical knowledge, their implementation requires sound knowledge of the computation medium on which the implemented algorithm is to be processed. This work has empirically proven that data/instructions organization plays an important role on the overall efficiency of any given algorithm.

- **Processor Performance:** Although our work only relied on the design of cache optimized algorithms, the complex operational diagram of modern processors is made of various other components that may be regarded as time consumers. This work has thus raised the need to algorithmically investigate the effect of those components on running programs in general. Such investigation may lead to the design of algorithms that take advantage of the capabilities of those various components.

Based on the above mentioned lessons learnt from our work, a personal perspective could be summarized as follows: the design of efficient algorithms based on theoretical abstractions is good; but a design that takes into account the potentialities and weaknesses of the computational medium on which the algorithm is to be processed, is better. It is thus of importance to account for hardware capabilities when designing algorithms that have to account for efficiency. In the next chapter, we present further directions that this work could take.

University of Pretoria etd – Ketcha Ngassam, E (2007)

CHAPTER 12

FUTURE WORK

Although this research has closed the gap in the availability of alternative implementation strategies for FA-based string processors, there is a variety of future challenges that require further investigations. Each subsection below contains a list of projects that could be undertaken by future researchers.

12.1 Projects of limited scale

The following projects are of limited scale and could conceivably be undertaken by junior graduate students (for example, in their 4^{th} year of study).

- **Algorithm Performance Analysis**: Carry out an analysis of the performance of the various algorithms, based on *artificial* data, in order to capture their strengths and weaknesses. In this project the researcher should select an algorithm from the 168 available. The best case behaviour of each algorithm should be determined as well as a complete analysis of the effect of caching on the algorithm. The researcher should also determine the effect of any other hardware component (pipeline, trace-cache, branch prediction, etc.) on the algorithm.

- **Applied Algorithms Analysis**: This involves an investigation of each algorithm as applied to specific problems such as network intrusion detection, tandem repeat finding, natural and computer virus scanning, etc. In this project, an algorithm should be identified and various analysis on it should be performed, using a real-life application. There is a need to investigate whether or not the chosen algorithm will outperform the conventional one (usually the TD). If the algorithm appears to under-perform its conventional counterpart, then there would be a need to investigate possible ways of improving the algorithm.

12.2 Medium-scale projects

- **A toolkit for FA-based string recogintion**: This project aims at improving and implementing the architectural design suggested in Chapter 8. The project is currently being undertaken as a postgraduate (masters level) exercise.

- **Extension of the taxonomy for DFA-based string recognizers.** The taxonomy suggested in Chapter 7 relied on the three implementation strategies that were used as parameter variables associated with the core algorithms. However, investigations could also be conducted not only on new strategies, but also on any other data-structure based recognizers such as linked-lists, trees, graphs, etc. A new taxonomy of DFA-based string processing algorithms could be proposed.

- **Improvement of the HC, TD, and MM algorithms**. This would be a project in high-performance computing. Alternatives ways should be investigated of how to obtain even faster TD, HC, and MM algorithms.

- **Platform specific investigation of the algorithms**. Various hardware platforms (such as Intel, Power-PC, Silicon Graphics, and the like) should be used to conduct experiments testing for performance of the various algorithms. The effects of cache, pipelining, branch prediction on various platforms should be investigated.

12.3 Advanced research projects

- **Hardware implementation of DFA-based string processing.** The hard-coded algorithm would be the starting point of this challenging project. All the HC algorithms should be translated to hardware. The advantages/disadvantages of implementing string recognizers on hardware should be explored. This might result in the implementation of specialized DFA-based applications on hardware.

- **Design and implementation of other cache optimized applications**. There may be other computational problems, unrelated to DFA-based string recognition, that are performance sensitive and that could be investigated following the same approach used throughout this thesis.

- **Investigation of other hardware performance metrics:** The operational diagram suggested in Chapter 3 requires further investigations in that we only restricted ourselves to the cache's locality of references. However, several other aspects could be envisaged for performance enhancement. Further analysis of the operational diagram could lead to suggestions on whether and how it might be possible to algorithmically influence positively the performance of those components.

12.4 End note

The source code for the various experiments conducted have intentionally not been released, but are available on request by e-mailing the author.

University of Pretoria etd – Ketcha Ngassam, E (2007)

BIBLIOGRAPHY

[Ale01] Andrei Alexandrescu, *Modern C++ Design: Generic Programming and Design Patterns Applied*, first ed., Addison-Wesley Professional, February 2001.

[ASU86] Alfred V. Aho, Ravi Sethi, and Jeffrey D. Ullman, *Compilers: Principles, Techniques, and Tools*, second ed., Addison Wesley, 1986.

[AU73] Alfred V. Aho and Jeffrey D. Ullman, *The Theory of Parsing, Translation, and Compiling*, vol. 2, Prentice Hall, 1973.

[Bac99] Rebecca Gurley Bace, *Intrusion Detection*, first ed., Sams Publishing, 1999.

[Bro83] Mandfred Broy, *Program Construction by Transformations: A Family Tree of Sorting Programs*, Computer Program Synthesis Methodologies, vol. 95, 1983, pp. 1–49.

[CH91] J. M. Champarnaud and G. Hansel, *Automate: A Computing Package for Automata and Finite Semigroups*, Journal of Symbolic Computation (G. Rozenberg and A. Salomaa, eds.), vol. 12, 1991, pp. 197–220.

[CH97] Maxime Crochemore and Christophe Hancart, *Automata for Matching Patterns*, Handbook of Formal Languages (G. Rozenberg and A. Salomaa, eds.), vol. 2, Springer-Verlag, 1997, pp. 399–462.

[Cor02] Intel Corporation, *The Intel Optimization Reference Manual*, http://www.intel.com/design/pentiumiii/manuals/, [last date accessed: 22 July 2005], 2002.

[Cra03] Chris Crawford, *Chris Crawford on Game Design*, first ed., New Riders Games, 2003.

[Cro02] Tim Crothers, *Implementing Intrusion Detection Systems : A Hands-On Guide for Securing the Network*, first ed., Wiley, 2002.

[CW05] Loek Cleophas and Bruce W. Watson, *TAxonomy-Based Software COnstruction of SPARE time: A Case Study*, IEE Proceedings Software, vol. 152, February 2005, pp. 29–37.

[DC04] Rael Dornfest and Tara Calishain, *Google Hacks : Tips & Tools for Smarter Searching*, second ed., O'Reilly Media, Inc., 2004.

[Deu99] A. Deutsch, *Principles of Biological Pattern Formation: Swarming and Aggregation viewed as Self-organization Phenomena*, Journal of Bioscience, vol. 24, 1999, pp. 115–120.

[DF88] Edsger W. Dijkstra and W. H. J. Feijen, *A Method of Programming*, Addison Wesley, 1988.

[DHI+00] C. C. Douglas, J. Hu, M. Iskandarani, M. Kowarschik, U. Rude, and C. Weiss, *Maximizing Cache Memory usage for Multigrid Algorithms*, In Multiphase Flows and Transport in Porous Media: State of the Art, Springer, Berlin, 2000, pp. 124–137.

[Dij76] Edsger W. Dijkstra, *A Discipline of Programming*, Prentice Hall, 1976.

[DWM00] R. D. Dowsing, F.W.D Woodhams, and I. Marshall, *Computers from Logic to Architecture*, second ed., McGraw-Hill, 2000.

[Epp95] Susanna S. Epp, *Discrete Mathematics with Applications*, second ed., Thompson Publishing, 1995.

[FCH02] R. Franklin, D. Carver, and B. L. Hutchings, *Network Intrusion Detection with Reconfigurable Hardware*, in IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM) (Napa, CA, USA), 2002.

[FH91] Christopher W. Fraser and Robert R. Henry, *Hard-coding Bottom-up Code Generation Tables to save Time and Space*, Software—Practice&Experience, vol. 21, January 1991, pp. 1–12.

[Fri05] Michiel Frishert, *FIRE Station: a FInite automata and Regular Expression playground*, Master's thesis, Department of Mathematics and Computer Science, Eindhoven, The Netherlands, 2005.

[Ger98] Richard Gerber, *The Software Optimization Cookbook: High-perfromance Recipes for the Intel Architecture*, third ed., Intel Press, 1998.

[GJ91] Dick Grune and Ceriel J. H. Jacob, *Parsing Techniques: A Practical Guide*, Prentice Hall, 1991.

[Gov01] S. Govindarajan, *Inside the Pentium 4*, http://www.pcquest.com/content/technology/101021101.asp, [last date accessed: 22 July 2005], 2001.

[Gus97] Dan Gusfield, *Algorithms on Strings, Trees, and Sequences: Computer Science and Computational Biology*, first ed., Cambridge University Press, 1997.

[Hau01] Roland R Hausser, *Foundations of Computational Linguistics*, second ed., Springer, 2001.

[Hay98] John P. Hayes, *Computer Architecture and Organization*, third ed., McGraw-Hill, 1998.

[HMU01] John E. Hopcroft, Rajeev Montwani, and Jeffrey D. Ullman, *Introduction to Automata Theory, Languages, and Computation*, second ed., Addison Wesley, 2001.

[HP03] John L. Hennessy and David A. Patterson, *Computer Architecture: A Quantitative Approach*, third ed., McGraw-Hill, 2003.

[HVZ02] C. Hamacher, Z. Vranesic, and S. Zaky, *Computer Organization*, fifth ed., McGraw-Hill, 2002.

[Hyd03] Randall Hyde, *The Art of Assembly Language*, first ed., No Strach Press, September 2003.

[iCS03] Lecture Notes in Computer Science, *Field-Programmable Logic and Applications*, vol. 2778, Springer Berlin / Heidelberg, Lisbon, Portugal, 2003, 13^{th} International Conference, FPL, Proceedings.

[Jon82] H. B. M. Jonkers, *Abstraction, Specification and Implementation Techniques*, Ph.D. thesis, Faculty of Mathematics and Computer Science, Eindhoven, the Nederlands, September 1982.

[JPTW90] V. Jansen, A. Pothoff, W. Thomas, and U. Wertmuth, *A Short Guide to the AMORE System*, vol. 90, Aachener Informatik-Berichte, 1990.

[KMP77] D. E. Knuth, J. H. Jr. Morris, and V. R. Pratt, *Fast Pattern Matching in Strings*, SIAM Journal on Computing, vol. 6, 1977, pp. 368–387.

[LMK+03] John W. Lockwood, James Moscola, Matthew Kulig, David Reddick, and Tim Brooks, *Internet Worm and Virus Protection in Dynamically Reconfigurable Hardware*, In Military and Aerospace Programmable Logic Device (MAPLD) (Washington DC), NASA Office of Logic Design, September 2003.

[Los98] David Loshin, *Efficient Memory Programming*, Mcgraw-Hill, November 1998.

[LP81] Harry R. Lewis and Christo H. Papadimitrou, *Elements of the Theory of Computation*, Prentice Hall, 1981.

[McC97] Roger A. McCain, *Cellular Genetic Automata in Computer Simulation of Economic Growth and Development with Romer Externalities*, Computing in Economics and Finance, vol. 41, 1997.

[McN82] Robert McNaughton, *Elementary Computability, Formal Languages and Automata*, Prentice Hall, 1982.

[Mey90] Bertrand Meyer, *Introduction to the Theory of Programming Languages*, c.a.r hoare series ed., Prentice Hall, 1990.

[MHP02] McGraw-Hill and Sybil P. Parker, *Mcgraw-Hill Dictionary of Scientific and Technical Terms*, sixth ed., McGraw-Hill Professional, September 2002.

[MLLP03] J. Moscola, J. Lockwood, R. P. Loui, and M. Pachos, *Implementation of a Content-Scanning Module for an Internet Firewall*, In Proceedings of the Eleventh Annual IEEE Symposium on Field-Programmable Custom Computing Machines, vol. 31, 2003.

[Mor94] Carroll Morgan, *Programming from Specifications*, second ed., Prentice Hall, 1994.

[Nga03] Ernest Ketcha Ngassam, *Hardcoding Finite Automata*, Master's thesis, Department of computer Science, Pretoria 0002, South Africa, November 2003.

[Nga05] _____, *Characterization of Finite Automata Implementations: A Preliminary Taxonomy*, The second FASTAR Worskhop, Finite Automata Systems Thoeretical and Applied Research, October 2005.

[NKW05a] Ernest Ketcha Ngassam, Derrick G. Kourie, and Bruce W. Watson, *Reordering Finite Automata States for Fast String Recognition*, In Proceedings of the Prague Stringology Conference (Prague, Czech Republic), Czech Technical University, August 2005.

[NKW05b] _____, *Reordering Finite Automata States for Fast String Recognition*, In Proceedings of the Prague Stringology Conference (Prague, Czech Republic), Czech Technical University, August 2005.

[NKW06a] Ernest Ketcha Ngassam, Derrick G. Kourie, and Bruce Watson, *Performance of Hardcoded Finite Automata*, Software Practice and Experience, vol. 36, 2006, pp. 525–538.

[NKW06b] Ernest Ketcha Ngassam, Derrick G. Kourie, and Bruce W. Watson, *Dynamic Allocation of Finite Automata States for Fast String Recognition*, International Journal of Foundations of Computer science (to appear), 2006.

[NKW06c] _____, *FA-based String Processing: The Hardcoded Dynamic State Allocation Algorithm*, In Proceedings of the African Conference on Research in Computer Science, Benin, ARIMA, November 2006.

[NKW06d] _____, *A Taxonomy of DFA-based String Processors*, In Proceedings of the SAICSIT Conference (Gordon's Bay, South Africa), ACM, October 2006, pp. 111–121.

[NN02] Stephen Northcutt and Judy Novak, *Network Intrusion Detection*, third ed., Sams Publishing, 2002.

[NR02] Gonzalo Navarro and Mathieu Raffinot, *Flexible Pattern Matching in Strings Practical: on-line search algorithms for texts and biological sequences*, Cambridge University Press, 2002.

[NWK03a] Ernest Ketcha Ngassam, Bruce W. Watson, and Derrick G. Kourie, *Hardcoding Finite State Automata Processing*, In Proceedings of the SAICSIT Conference (Johannesburg, South Africa), ACM, September 2003, pp. 111–121.

[NWK03b] _____, *Preliminary Experiments in Hardcoding Finite Automata*, In Proceedings of the 10^{th} Conference on Implementation and Application of Automata (Santa Barbara, CA, USA), Springer, July 2003, pp. 299–300.

[NWK04] _____, *A Framework for the Dynamic Implemenation of Finite Automata for Performance Enhancement*, In Proceedings of the Prague Stringology Conference (Prague, Czech Republic), Czech Technical University, August 2004.

[NWK05] _____, *A Framework for the Dynamic Implementation of Finite Automata for Performance Enhancement*, International Journal of Foundations of Computer Science, vol. 16, December 2005, pp. 1193–1206.

[NWK06] _____, *On Implementation and Performance of Table-driven DFA-based String Processors*, In Proceedings of the Prague Stringology Conference (Prague, Czech Republic), Czech Technical University, August 2006.

[PD04] Thomas J. Pennello and Frank DeRemer, *Efficient Computation of LALR(1) Look-Ahead Sets*, ACM Press Special Issue, vol. 39, April 2004, pp. 14–27.

[Pen86] Thomas J. Pennello, *Very Fast LR Parsing*, In Proceedings of the SIGPLAN Symposium on Compiler Construction, 1986, pp. 145–151.

[PH05] David A. Patterson and John L. Hennessy, *Computer Organization and Design*, third ed., Morgan Kaufmann, 2005.

[PTVF02] William H. Press, Saul A. Teukolsky, William T. Vetterling, and Brian P. Flannery, *Numerical Recipes in C++ : The Art of Scientific Computing*, second ed., Prentice Hall, February 2002.

[RBS99] E. Rotenberg, S. Bennett, and J. E. Smith, *A Trace Cache Microarchitecture and Evaluation*, IEEE Trans. Computers, vol. 48, 1999, pp. 111–120.

[RP93] D. R. Raymond and D Pwood, *The GRAIL papers: Version 2.0*, Technical Report University of Waterloo, Canada, January 1993.

[SLT02] T. Sproull, J. W. Lockwood, and D. E. Taylor, *Control and Configuration Software for a Reconfigurable Networking Hardware Platform*, In IEEE Symposium on Field-Programmable Custom Computing Machines (FCCM) (Napa, CA, USA), 2002.

[Tho68] Ken Thompson, *Regular Expression Search Algorithm*, Communications of the ACM, vol. 11, 1968, pp. 323–350.

[TPS05] Andrew Turplin, Simon J. Puglisi, and William F. Smyth, *A Taxonomy of Suffix Array Construction Algorithms*, In Proceedings of the Prague Stringology Conference (Prague, Czech Republic), Czech Technical University, August 2005.

[Vic84] Gerard Y. Vichniac, *Simulating Physics with Cellular Automata*, Physica, vol. D, 1984, pp. 96–116.

[VM03] John Viega and Matt Messier, *Secure Programming Cookbook for C and C++ : Recipes for Cryptography, Authentication, Input Validation & More*, first ed., O'Reilly Media, Inc.;, July 2003.

[Wat94] B. W. Watson, *The Design and Implementation of FIRE Engine: A C++ Toolkit for Finite Automata and Regular Expressions*, Technical Report, Technical University of Eindhoven, 1994.

[Wat95a] Michael S. Waterman, *Introduction to Computational Biology: Maps, Sequences and Genomes*, last ed., Chapman & Hall CRC, 1995.

[Wat95b] Bruce W. Watson, *Taxonomies and Toolkits of Regular Languages Algorithms*, Ph.D. thesis, Faculty of Mathematics and Computer Science, Eindhoven University of Technology, the Nederlands, September 1995.

[WC93] William M. Waite and Lynn R. Carter, *An Introduction to Compiler Contruction*, Harper Collins, 1993.

[WC04] Bruce W. Watson and Loek Cleophas, *SPARE PARTS: A C++ Toolkit for String Pattern Recognition*, Technical Report, Technical University of Eindhoven, vol. 34, 2004, pp. 697–710.

[Yao79] A. C. Yao, *The Complexity of Pattern Matching for a Random String*, SIAM Journal on Computing, vol. 8, 1979, pp. 368–387.