
Lamar University

COSC 5369, Fall 2003

Graduate Projects

Advisor: Dr. Chung-Chih Li

Office: 69 Maes, Tel: (409) 880-8748

URL: <http://hal.lamar.edu/~licc>

E-mail: licc@hal.lamar.edu

Office Hours: MWF 10:00 ~ 11:00 AM or by appointment

Class meeting times and place:

For first 2 weeks: MW 12:30 ~ 1:45 PM, Maes 113
(by appointment for each group afterwards)

Course Description and Purpose:

Independent study and research of a specific problem in a field of computer science or its application. A report is required defining the problem and developing a solution. The work will be supervised by the instructor.

4 Teams: at most 5 students for each team; at most 2 teams are allowed to work on the same topic.

Prerequisites: 10 hours of graduate computer science credit including COSC 5100 with grades of A or B; prior approval of written plan by the faculty supervisor and by the computer science department chair. May not be repeated for credit.

Time Frame:

Time length	Achievement	Deadline
1 week	Form a team and Select a topic	Sep. 8
3 weeks	Background study and Collect papers	Sep. 22
2 weeks	Form and Polish the idea	Oct. 6
3 weeks	Implement and test the idea	Nov. 3
3 weeks	Write up the report	Dec. 1
2 weeks	Revise the report	Dec. 15
1 week	Present the project	Dec 19
1 week	Finalize the report	—
Total: 16 weeks		

Tentative Topics for COSC 5369, Fall 2003

1. Machine Learning (either form practical side or theoretical side).
2. Parallel primality test; Pseudo big prime numbers and how “prime” they are.
See [12, 38] and any textbooks on parallel computing.
3. Case study and comparison on big-integer packages (libraries) and possible improvement.
Search the Internet for a good verity of big-integer packages.
4. E-mail encryption; a cryptosystem that combines the Linear Congruential Method and the feature of public key cryptosystem such as RSA for transmitting lengthy text through insecure channel.
See [12, 38, 26].
5. Study topological spaces between Baire topology and the relative topology determined by the concerned computation to see if there is an alternative space that can reinstall the transitivity property without hurting the finite computation properties.
Note: We strongly feel that such an alternative does not exist. If this turns out to be the case, we shall give an impossibility proof to settle down the search.
See [35, 31] for Baire topology and [29, 27] for the problem setup. For a not so technical glance, see [28].
6. Define a type-2 complexity class analog to the one given by Hartmanis and Stearns. Also, a type-2 big-O notation should follow the definition of type-2 complexity class.
See [18, 19, 14] for Hartmanis and Stearns’ complexity classes.
7. The condition for the enumerability (and hence the existence of a programming system) of type-2 complexity classes.
See [35, 31, 32, 14] for enumerability, [40, 6] for complexity classes’ enumerability. For acceptable programming system see [36, 34].
8. Prove a type-2 analog of Rabin’s theorem stating that there are arbitrarily complex and *dishonest* type-2 algorithms. Intuitively, a *dishonest* program is one that uses a lot of resources but produces a small output. A proof of type-2 Rabin’s theorem is a good demonstration of the use of asymptotic notation.
Rabin’s theorem [33], Honesty Theorem [30].
9. Prove a type-2 analog of the Union theorem. If our type-2 complexity classes can lead to a type-2 union theorem under some rather weak conditions, then we can be sure that our definition of type-2 complexity classes is a right one that can be applied to both practical and theoretical applications.
Union Theorem [30].

10. Examine some type-2 computational problems such as machine learning to see if our notion of type-2 complexity classes can introduce a better complexity characterization on the problems.

Related references:

- Speedup Theorem [4, 41], Gap Theorem [5, 9, 41], Compression Theorem [4],
- [21] used the notion of reduction to compare the difficulty of identification between different classes of languages. Some earlier works such as [13] also addressed the complexity issue in great details. There are some common complexity measures for computational learning surveyed in [20].
- [10] was the first paper explicitly dealing with higher order complexity theory. Early results in the subject can be found in [24, 25, 15]. Also, in early 70's, Symes presented a work on type-2 complexity theory with an axiomatic approach [39].
- Kapron and Cook's [11, 22, 23] syntactically defined a class of functions called *second-order polynomials* and the *length function* of any type-1 function f (denoted by $|f|$).
- Seth in [37] suggested a type-2 complexity class similar to Cook's.
- With regard to machine learning, Gold in [17] established a theoretical framework, and from his paradigm, there had been many models for machine learning such as, the Inductive Inference Machine (IIM) which had been extensively studied over the past few decades (e.g., [3, 2, 8, 7, 20] or Chapter VII.5 of [32]). Typically, an IIM is a Turing Machine M that works as follows:
 1. An external agent that carries a certain concept is fixed.
 2. M starts from an initial state with the input and output tapes empty.
 3. From time to time, one of the following situations happens:
 - (a) M enters a special state in which M holds its execution and waits for some inputs from the external agent. After the input has been prepared on the input tape, M resumes and proceeds to the next state.
 - (b) M starts to write a number on the output tape. When it finishes the writing, M either simply stops or continues its execution.

M may keep asking for inputs from the external agent or changing the contents of the output tape as many times as needed.

The set of *primitive recursive* functions is identifiable but the set of *recursive* functions is not [3]. Changing the criteria of identification may result in different identifiable classes [8].

The Student-Teacher learning model can be found in [1, 16] and the book [20].

References

- [1] Dana Angluin. Queries and concept learning. *Machine Learning*, 2:319–342, 1988.
- [2] Dana Angluin and Carl Smith. Inductive inference: Theory and methods. *Computing Surveys*, 15(3):237–269, September 1983.
- [3] Lenore Blum and Manuel Blum. Toward a mathematical theory of inductive inference. *Information and Control*, 28:125–155, 1975.
- [4] Manuel Blum. A machine-independent theory of the complexity of recursive functions. *Journal of the ACM*, 14(2):322–336, 1967.
- [5] A. Borodin. Computational complexity and the existence of complexity gaps. *Journal of the ACM*, 19(1):158–174, 1972.
- [6] Critian Calude. *Theories of Computational Complexity*. Number 35 in Annals of Discrete Mathematics. North-Holland, Elsevier Science Publisher, B.V., 1988.
- [7] John Case. Learning machines. In W. Demopoulos and A. Marras, editors, *Language Learning and Concept Acquisition*. Ablex Publishing Co., 1986.
- [8] John Case and Carl Smith. Comparison of identification criteria for machine inductive inference. *Theoretical Computer Science*, 25:193–220, 1983.
- [9] Robert L. Constable. The operator gap. *Journal of the ACM*, 19:175–183, 1972.
- [10] Robert L. Constable. Type two computational complexity. *Proceedings of the Fifth ACM Symposium on the Theory of Computing*, pages 108–122, 1973.
- [11] Stephen A. Cook and Bruce M. Kapron. Characterization of the basic feasible functions of finite type. *Proceedings of the 30th Annual IEEE Symposium on the Foundations of Computer Science*, pages 154–159, 1989.
- [12] Thomas H. Cormen, Charles E. Leiserson, and Ronald L. Rivest. *Introduction to Algorithms*. The MIT Press, Cambridge, Massachusetts, 1989.
- [13] Robert P. Daley and Carl Smith. On the complexity of inductive inference. *Information and Control*, 69:12–40, 1986.
- [14] Martin D. Davis and Elaine J. Weyuker. *Computability, Complexity, and Languages*. Academic Press, San Diego, 1983.
- [15] R.O. Gandy and J.M.E. Hyland. Computable and recursively countable functions of higher type. *Logic Colloquium*, 76:405–438, 1977.
- [16] William I. Gasarch and Carl H. Smith. Learning via queries. *Journal of the ACM*, 39(3):649–674, 1992.
- [17] E. Mark Gold. Language identification in the limit. *Information and Control*, 10:447–474, 1967.

- [18] J. Hartmanis and R. E. Stearns. On the computational complexity of algorithms. *Transitions of the American Mathematics Society*, pages 285–306, May 1965.
- [19] John E. Hopcroft and Jeffrey D. Ullman. *Introduction to Automata Theory, Languages, and Computation*. Addison-Wesley, 1979.
- [20] Sanjay Jain, Daniel Osherson, James S. Royer, and Arun Sharma. *Systems That Learn*. The MIT Press, Cambridge, Massachusetts, second edition, 1999.
- [21] Sanjay Jain and Arun Sharma. On the intrinsic complexity of language identification. In *Proceedings of the Seventh Annual ACM Conference on Computational Learning Theory*, pages 278–286, 1994.
- [22] Bruce M. Kapron and Stephen A. Cook. A new characterization of Mehlhorn’s polynomial time functionals. *Proceedings of the 32th Annual IEEE Symposium on the Foundations of Computer Science*, pages 342–347, 1991.
- [23] Bruce M. Kapron and Stephen A. Cook. A new characterization of type 2 feasibility. *SIAM Journal on Computing*, 25:117–132, 1996.
- [24] Steve C. Kleene. Turing-machine computable functionals of finite types II. *Proceedings of London Mathematical Society*, 12:245–258, 1962.
- [25] Steve C. Kleene. Recursive functionals and quantifies of finite types II. *Transitions of the American Mathematics Society*, 108:106–142, 1963.
- [26] Donald Ervin Knuth. *The Art of Computer Programming*, volume 2. Addison-Wesley, second edition, 1981.
- [27] Chung-Chih Li. Type-2 complexity theory. Ph.d. dissertation, Syracuse University, New York, 2001.
- [28] Chung-Chih Li. A new approach to measuring the computational complexity of type-2 algorithms. Reg proposal, Lamar University, New York, 2003. (not so technical written).
- [29] Chung-Chih Li and James S. Royer. On type-2 complexity classes: Preliminary report. *Proceedings of the Third International Workshop on Implicit Computational Complexity*, pages 123–138, May 2001.
- [30] E. McCreight and A. R. Meyer. Classes of computable functions defined by bounds on computation. *Proceedings of the First ACM Symposium on the Theory of Computing*, pages 79–88, 1969.
- [31] Piergiorgio Odifreddi. *Classical Recursion Theory*, volume 125 of *Studies in Logic and the Foundations of Mathematics*. Elsevier Science Publishing, North-Holland, Amsterdam, 1989.
- [32] Piergiorgio Odifreddi. *Classical Recursion Theory, Volume II*, volume 143 of *Studies in Logic and the Foundations of Mathematics*. Elsevier Science Publishing, North-Holland, Amsterdam, 1999.

- [33] M.O. Rabin. Degree of difficulty of computing a function and a partial ordering of recursive sets. Technical Report 2, Hebrew University, 1960.
- [34] Hartley Rogers, Jr. *Theory of Recursive Functions and Effective Computability*. McGraw-Hill, 1967. First paperback edition published by MIT Press in 1987.
- [35] Hartley Rogers, Jr. *Theory of Recursive Functions and Effective Computability*. The MIT Press, third edition, 1992. Original edition published by McGraw-Hill in 1967. First paperback edition published by MIT Press in 1987.
- [36] James Royer and John Case. *Subrecursive Programming Systems: Complexity & Succinctness*. Birkhäuser, 1994.
- [37] Anil Seth. Complexity theory of higher type functionals. Ph.d. dissertation, University of Bombay, 1994.
- [38] Douglas R. Stinson. *Cryptography, Theory and Practice*. Chapman & Hall/CRC Press, New York, second edition, 2002.
- [39] D.M. Symes. The extension of machine independent computational complexity theory to oracle machine computation and the computation of finite functions. Ph.d. dissertation, University of Waterloo, Oct. 1971.
- [40] Klaus Wagner and Gerd Wechsung. *Computational Complexity*. Mathematics and its applications (East European Series). D. Reidel Publishing Company, Dordrecht, 1985.
- [41] Paul Young. Easy construction in complexity theory: Gap and speed-up theorems. *Proceedings of the American Mathematical Society*, 37(2):555–563, February 1973.