

IMPLICIT COMPUTATIONAL COMPLEXITY

Examination reading list

Other titles may be added. Feel free to propose some title you would like to read and comment upon.

If you choose an argument, write its number next to your name in the participants list. Please mark also with ``HERE'' if you want to submit your report by next Friday.

Basic formal systems (discussed in the lectures)

1. S. Bellantoni, S. Cook. A NEW RECURSION-THEORETIC CHARACTERIZATION OF THE POLYTIME FUNCTIONS. Computational Complexity vol 2. pp. 97-199, 1992.
2. D. Leivant. Stratified functional programs and computational complexity. ACM 20th symp. on Princ. of Progr. Lang. (Popl) 1993.
3. D. Leivant. Ramified recurrence and computational complexity I: Word recurrence and poly-time, in Peter Clote and Jeffrey Remmel (editors), Feasible Mathematics II, Perspective in Computer Science, Birkhauser 1994, pp. 320–343.
4. D. Leivant, J.-Y. Marion. Ramified recurrence and computational complexity II: substitutions and poly-space. in J. Tiuryn and L. Pacholsky (editors), Computer Science Logic, Lecture Notes in Computer Science 933, Proceedings of CSL '94, Springer-Verlag, 1995, 486–500. **(somehow specialistic; not discussed)**
5. D. Leivant. Ramified recurrence and computational complexity III: Higher type recurrence and elementary complexity, Annals of Pure and Applied Logic 96/3, 1999, p.209-229. **(not discussed)**

Higher order systems (cited in the lectures: something discussed on Friday)

6. M. Hofmann. Linear types and non-size increasing polynomial time computation. Information and Computation 183(1), 57-85, 2003
7. M. Hofmann. A mixed modal/linear lambda-calculus with applications to Bellantoni-Cook safe recursion. Conference on Computer Science Logic 1997. Springer LNCS 1414, 1998, pages 275-294

First order rewriting systems (not discussed)

8. Bonfante, Marion, Moyen. On complexity analysis by Quasi-interpretation. proceedings of APPSEM 2004.

Interesting, but not really implicit complexity (not discussed)

9. Neil Jones. Constant time factors do matter. ACM STOC '93. Symposium on Theory of Computing (1993) 602--611.

Basics of Curry-Howard correspondence

10. Girard, Lafont, Taylor. Proofs and Types. Cambridge Univ. Press. (It is a book!)

Basics of Linear Logic (very logically oriented; no computational complexity)

11. Linear Logic Primer; in

<http://www.pps.jussieu.fr/~dicosmo/CourseNotes/LinLog/> (some of the first chapters).

Fragments of Linear Logic (discussed on Thursday; somehow specialistic)

12. A. Asperti. Light Affine Logic. Proceedings of the 13th Annual IEEE Symposium on Logic in Computer Science (LICS) 1998. pp 300ss
13. Y. Lafont. Soft linear logic and polynomial time. Theoretical Computer Science, Volume 318 , Issue 1-2 (June 2004). pag. 163-180.
14. P. Baillot, K. Terui. Light types for polynomial time computation in lambda-calculus, in Proceedings of LICS 2004, pp. 266-275, IEEE Computer Society Press, 2004. (discussed on Friday)

Applications to type systems and/or programming languages (not really discussed in the lectures; various levels of difficulty and background)

15. Lennart Beringer, Martin Hofmann, Alberto Momigliano and Olha Shkaravska. Automatic Certification of Heap Consumption. In Logic for Programming, Artificial Intelligence, and Reasoning: 11th International Conference, LPAR 2004, Montevideo, Uruguay, March 14-18, 2005. Proceedings, Publisher: Springer-Verlag GmbH, February 2005, Vol. 3452, pages 347-362.
16. M. Hofmann and S. Jost. Static prediction of heap space usage for first-order functional programs. In Proceedings of the 30th ACM Symposium on Principles of Programming Languages, ACM Press, New York, January 2003, Vol. 38, No. 1, pages 185-197.
17. Kenneth MacKenzie and Nicholas Wolverson. Camelot and Grail: resource-aware functional programming on the JVM. In Trends in Functional Programming, Intellect, 2004, Vol. 4, pages 29-46.
18. Lennart Beringer, Martin Hofmann, Alberto Momigliano and Olha Shkaravska. Towards certificate generation for linear heap consumption. In Proceedings of ICALP/LICS Workshop on Logics for Resources, Processes, and Programs (LRPP2004), July 2004.
19. Lennart Beringer, Kenneth MacKenzie and Ian Stark. Grail: a Functional Form for Imperative Mobile Code. In Foundations of Global Computing: Proceedings of the 2nd EATCS Workshop, Elsevier, June 2003, No. 85.1.
20. David Aspinall and Adriana Compagnoni. Heap-Bounded Assembly Language. In Journal of Automated Reasoning, Kluwer, Dordrecht, The Netherlands, 2003, Vol. 31, No. 3-4, pages 261-302.
21. Hans-Wolfgang Loidl and Kenneth MacKenzie. A Gentle Introduction to Camelot. LFCS, Univ of Edinburgh & Inst for Informatics, LMU Univ, September 2004.
22. Brian Campbell. Folding stack memory usage prediction into heap. In Proceedings of Quantitative Aspects of Programming Languages Workshop, ETAPS 2005, April 2005.