Electronic Colloquium on Computational Complexity, Report No. 57 (2002)

ECCC

Non-uniform Depth of Polynomial Time and Space Simulations

Richard J. Lipton*
and Anastasios Viglas**

Abstract. We discuss some connections between polynomial time and non-uniform, small depth circuits. A connection is shown with simulating deterministic time in small space. The well known result of Hopcroft, Paul and Valiant [HPV77] showing that space is more powerful than time can be improved, by making an assumption about the connection of deterministic time computations and non-uniform, small depth circuits. To be more precise, we prove the following: If every linear time deterministic computation can be done by non-uniform circuits of polynomial size and sub-linear depth (for example, if $\mathcal{P}$ is in non-uniform $\mathcal{NC}$), then $\mathcal{DTIME}(t) \subseteq \mathcal{DSPACE}(t^{1-\epsilon})$ for some constant $\epsilon > 0$.
We can also apply the same techniques to prove an unconditional result, a trade-off type of theorem for the size and depth of a non-uniform circuit that simulates a uniform computation. We prove that time t has non-uniform circuits of depth $\sqrt{t}$, with polynomially bounded fan-in for one type of gates ("semi-unbounded" fan-in type circuits).

1 Introduction

We present an interesting connection between non-uniform characterizations of Polynomial time and time versus space results.

Hopcroft Paul and Valiant [HPV77] proved that space is more powerful than time: $\mathcal{DTIME}(t) \subseteq \mathcal{DSPACE}(t/\log t)$. The proof of this trade-off result is based on pebbling techniques and the notion of *block respecting* computation. Improving the space simulation of deterministic time has been a long standing open problem. Paul Tarjan and Celoni [PTC77] proved an $n/\log n$ lower bound for pebbling a certain family of graphs. This lower bound implies that the trade-off result $\mathcal{DTIME}(t) \subseteq \mathcal{DSPACE}(t/\log t)$ of [HPV77] cannot be improved using similar pebbling arguments.

In this work we present a connection between space simulations of deterministic time and the depth of non-uniform circuits simulating polynomial time computations. If every problem in linear deterministic time can be solved by polynomial size non-uniform circuits of small (sub-linear) depth then every deterministic computation of time t can be simulated in space $t^{1-\epsilon}$ for some constant $\epsilon > 0$ (that depends only on our assumption about the non-uniform depth of linear time):

* College of Computing, Georgia Institute of Technology, rjl@cc.gatech.edu, and Telcordia Applied Research. Supported in part by NSF CCR-9633103.
** Computer Science Department, University of Toronto, aviglas@cs.toronto.edu.

ISSN 1433-8092

$$\begin{aligned} \mathcal{DTIME}(n) \subseteq & \mathcal{SIZE}\text{–}\mathcal{DEPTH}(poly(n), n^{\delta}) \\ & \Longrightarrow \mathcal{DTIME}(t) \subseteq \mathcal{DSPACE}(t^{1-\epsilon}) \end{aligned} \tag{1}$$

where $\delta < 1$ and $\epsilon > 0$. Note that we allow the size of the non-uniform circuit to be *any* polynomial. Since $\mathcal{DTIME}(t) \subseteq \mathcal{SIZE}(t \cdot \log t)$ (proved in [PF79]), our assumption basically asks to reduce the depth of the non-uniform circuit by a small amount, allowing the size to increase by any polynomial factor.

It is interesting to note that in this result, a non-uniform assumption is used ($\mathcal{P}$ has small non-uniform depth) to prove a purely uniform result (deterministic time can be simulated in small space). This can also be considered as an interesting result for the power of non-uniformity: If non-uniformity is powerful enough to allow small depth circuits for linear time deterministic computations, then we can improve the space-bounded simulation of deterministic time given by Hopcroft Paul and Valiant.

A related result was shown by Sipser [Sip86,Sip88] from the point of view of reducing randomness required for randomized algorithms. His result considers the problem of constructing expanders with certain properties. Assuming that those expanders can be constructed efficiently, the main theorem proved is that either $\mathcal{P}$ is equal to $\mathcal{RP}$ *or* the space simulation of Hopcroft, Paul and Valiant [HPV77] can be improved:

Theorem 1 (Sipser [Sip86]). *Under the hypothesis that certain expanders have explicit constructions, there exists an $\epsilon > 0$ such that*

$$(\mathcal{P} = \mathcal{RP}) \text{ or } (\mathcal{DTIME}(t) \cap 1^*) \subseteq \mathcal{DSPACE}(t^{1-\epsilon}) \tag{2}$$

An explicit construction for the expanders mentioned above was given by Saks, Srinivasan and Zhou [SSZ98]. The theorem mentioned above reveals a deep connection between pseudo-randomness and efficient space simulations (for unary languages): either space bounded simulations for deterministic time can be improved, or we can construct (pseudorandom) sequences that can be used to improve the derandomization of certain algorithms. On the other hand, the result we are going to present in this work, gives a connection between the power of non-uniformity and the power of space bounded computations.

Other related results include Dymond and Tompa [DT85] where it is shown that $\mathcal{DTIME}(t) \subseteq \mathcal{ATIME}(t/\log t)$, improving the Hopcroft Paul Valiant theorem, and Paterson and Valiant [PV76] proving $\mathcal{SIZE}(t) \subseteq \mathcal{DEPTH}(t/\log t)$.

We also show how to apply the same techniques to prove an unconditional trade-off type of result for the size and depth of a non-uniform circuit that simulates a uniform computation. Any deterministic time t computation can be simulated by a non-uniform circuit of size roughly $2^{\sqrt{t}}$ and depth $\sqrt{t}$, which has "semi-unbounded" fan-in: all AND gates have polynomially bounded fan-in and OR gates are unbounded, or vice versa. Similar results were given in [DT85] showing that time t is in PRAM time $\sqrt{t}$.

2 Notation - Definitions

We use the standard notation for time and space complexity classes $\mathcal{DTIME}(t)$ and $\mathcal{DSPACE}(t)$. $\mathcal{SIZE}$–$\mathcal{DEPTH}(s,d)$ will denote the class of non-uniform circuits with size (number of gates) $O(s)$ and depth $O(d)$. We also use $\mathcal{NC}/poly$ ($\mathcal{NC}$ with polynomial advice) to denote the class of non-uniform circuits of polynomial size and poly-logarithmic depth, $\mathcal{SIZE}$–$\mathcal{DEPTH}(poly, polylog)$. At some points in the paper, we will also avoid writing poly-logarithmic factors in detail and use the notation $\tilde{O}(n)$ to denote $O(n \log^k n)$ for constant k. In this work we consider time complexity functions that are time constructible: A function $t(n)$ is called fully time constructible if there exists a deterministic Turing Machine that on input of length n halts after exactly $t(n)$ steps. In general a function $f(n)$ is t-time constructible, if there is a deterministic Turing Machine that on input x outputs $1^{f(|x|)}$ and runs in time $O(t)$. (t,s)-time-space constructible functions are defined similarly. We also use "TM" for "deterministic Turing Machine".

For the proof of the main result we use the notion of block respecting Turing machines introduced by Hopcroft Paul and Valiant in [HPV77].

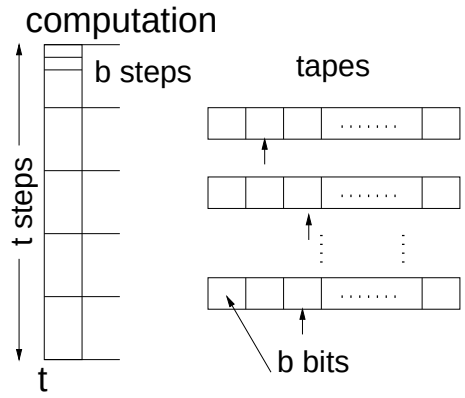

Fig. 1. Block respecting computation

Definition 1. *Let M be a machine running in time $t(n)$, where n is the length of its input x. Let the computation of M be partitioned in $a(n)$* segments, *where each segment consists of $b(n)$ consecutive steps, $a(n) \cdot b(n) = t(n)$. Let also the tapes of M be partitioned into $a(n)$* blocks *each consisting of $b(n)$ bits (cells) on each tape. We will call M* block respecting *if during each segment of its computation, each head visits only one block on each tape.*

Every Turing Machine can be converted to a block respecting machine with only a constant factor slow down in its running time. The construction is simple:

Let M be a deterministic Turing Machine running in time t. Break the computation steps $(1 \dots t)$ in segments of size B. Break the work tapes in blocks of the same size B. If at the start of a computation segment σ the work tape head is in block b_j, then during the computation steps (b steps) of that segment, the head could only visit the adjacent blocks, b_{j-1} or b_{j+1}. Keep a copy of those two blocks along with b_j and do all the computation of segment σ reading and updating from those copies (if needed). At the end of the computation of every segment, there is a clean-up step: update the blocks b_{j-1} and b_{j+1} and move the work tape head to the appropriate block to start the computation of the next segment. This construction can be done for different block sizes B. For our purposes B will be t^c for a small constant $c < 1$.

Block respecting Turing machines are also used in [PPST83] to prove that non-deterministic linear time is more powerful than deterministic linear time (see also [PR81] for a generalization of the results from [HPV77] for RAMs and other machine models).

3 Main Results

We show that if linear time has small non-uniform circuit depth (for polynomial size circuits) then $\mathcal{DTIME}(t) \subseteq \mathcal{DSPACE}(t^{1-\epsilon})$ for a constant $\epsilon > 0$.

To be more precise, the strongest form of the main result is the following: if (deterministic) linear time has polynomial size, non-uniform circuits of sublinear depth (for example depth n^δ for $0 < \delta < 1$), then $\mathcal{DTIME}(t) \subseteq \mathcal{DSPACE}(t^{1-\epsilon})$ for a small positive $\epsilon > 0$:

$$\mathcal{DTIME}(n) \subseteq \mathcal{SIZE\text{–}DEPTH}(poly, n^\delta) \implies \mathcal{DTIME}(t) \subseteq \mathcal{DSPACE}(t^{1-\epsilon}) \tag{3}$$

The main idea is the following: Start with a deterministic Turing machine M running in time t and convert it to a block respecting machine M_B with block size B. In each segment of the computation, M_B reads and/or writes in exactly one block on each tape. We will argue that we can check the computation in each such segment with the *same* sub-circuit and we can actually construct this sub-circuit with polynomial size and small (poly-logarithmic or sub-linear) depth. Combining all these sub-circuits together we can build a larger circuit that will check the entire computation of M_B in small depth. The final step is a technical lemma that shows how to evaluate this circuit in small space (equal to its depth).

We start by proving the main theorem using the assumption $\mathcal{P} \subseteq \mathcal{NC}/poly$. It is easy to see that an assumption of the form $\mathcal{DTIME}(n) \subseteq \mathcal{NC}/poly$ implies $\mathcal{P} \subseteq \mathcal{NC}/poly$ by padding arguments.

Theorem 2. *Let t be a polynomial time complexity function. If $\mathcal{P} \subseteq \mathcal{NC}/poly$ then $\mathcal{DTIME}(t) \subseteq \mathcal{DSPACE}(t^{1-\epsilon})$ for some constant $\epsilon > 0$.*

Proof. (Any "reasonable" time complexity function could be used in the statement of this theorem.) Consider any Turing Machine M running in deterministic time t. Here is how to simulate M in small space using the assumption that polynomial time has shallow (poly-logarithmic depth) polynomial size circuits:

1. Convert given TM in a block respecting machine with block size B.
2. Construct the graph that describes the computation. Each vertex corresponds to a computation segment of B steps.
3. The computation on each vertex can be checked by the *same* TM U that runs in polynomial time (linear time)
4. Since $\mathcal{P} \subseteq \mathcal{NC}/poly$, there is a circuit U_C that can replace U. U_C has polynomial size and polylogarithmic depth.
5. Construct U_C by trying all possible circuits.
6. Plug in the sub-circuit U_C to the entire graph. This graph is the description of a circuit of small depth, that corresponds to the computation of the given TM. Evaluate the circuit (in small space)

In more detail: Convert M to a block respecting machine M_B. Break the computation of M_B (on input x) in segments of size B each; the number of segments is t/B. Consider the directed graph G corresponding to the computation of the block respecting machine as described in [HPV77]: G has one vertex for every time segment (that is t/B vertices) and the edges are defined from the sequence of head positions. Let $v(\Delta)$ denotes the vertex corresponding to time segment Δ then and Δ_i is the last time segment before Δ during which the i-th head was scanning the same block as during segment Δ. Then the edges of G are $v(\Delta - 1) \to v(\Delta)$ and for all $1 \leq i \leq l$, $v(\Delta_i) \to v(\Delta)$. The number of edges can be at most $O(\frac{t}{B})$ and therefore the number of bits required to describe the graph is $O\left(\frac{t}{B} \log \frac{t}{B}\right)$.

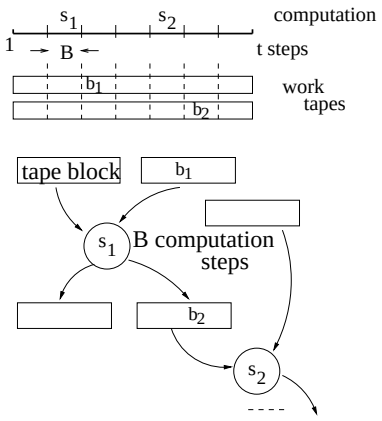

Fig. 2. Graph description of a block respecting computation.

Figure 2 shows the idea behind the construction of the graph for the block respecting computation. The computation is partitioned in segments of size B. Every segment corresponds to a vertex (denoted by a circle in figure 2). Each segment will access only one block on each tape. Figure 2 shows the tape blocks blocks which are read during a computation segment (input blocks for that vertex) and those that will be written during the same segment (shown as output blocks). If a block is written during a segment and the same block is read by another computation segment later in the computation, then the second segment depends directly from the previous one and there will be an edge connecting the corresponding vertices in our graph.

Each vertex of this graph corresponds to B computation steps of M_B. During this computation, M_B reads and writes only in one block from each tape. In order to check the computation that corresponds to a vertex of this graph, we would need to simulate M_B for B steps and check $O(B)$ bits from M_B's tapes. For each vertex we need to check/simulate a different segment of M_B's computation: this can be done by a Turing machine that will check the corresponding computation of M_B. We argue that the same Turing machine can be used on every vertex. The computation we need to do on each vertex of the graph is essentially the same: given the "input" and "output" contents of certain tape blocks, simulate the machine M_B for B steps and check if the output contents are correct. The only thing that changes is the actual segment of the computation of M_B that we are going to simulate (which B steps of M_B we should simulate). This means that the exact same "universal" Turing machine checks the computation for each segment/vertex, and this universal machine also takes as input the description (for example the index of the part of the computation of the initial machine M_B it will need to simulate or any reasonable encoding) of the computation that it needs to actually simulate on each vertex. Therefore we have the same machine U on all vertices of the graph which runs in deterministic polynomial time.

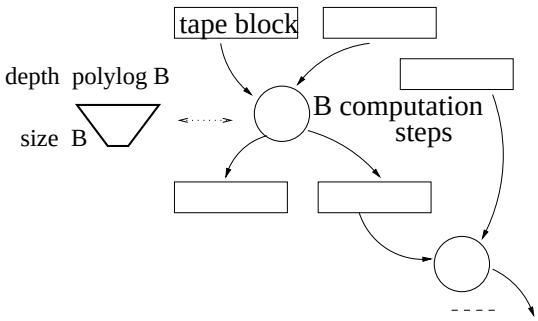

Fig. 3. Insert the (same) sub-circuit on all vertices

If $\mathcal{P} \subseteq \mathcal{SIZE}$–$\mathcal{DEPTH}(n^k, \log^l n)$ then U can be simulated by a circuit U_C of size $O(B^k)$ and small depth $O(\log^l B)$, for some k, l. The same circuit

is used on all vertices of the graph. In order to construct this circuit, we can try all possible circuits and simulate them on all possible inputs. This requires exponential time, but only small amount of space: the size of the circuit is B^k and its depth polylogarithmic in B. We need $\tilde{O}(B^k)$ bits to write down the circuit and only polylog space to evaluate it (using lemma 1).

Once we have constructed U_C, we can build the entire circuit that will simulate M_B. This circuit derives directly from the (block-respecting) computation graph where each vertex is an instance of the sub-circuit U_C. The size of the entire circuit is too big to write down. We have up to t/B sub-circuits (U_C) that would require a size of $\tilde{O}(\frac{t}{B}B^k)$ for some constant k. But since it is the same sub-circuit U_C that appears throughout the graph, we can implicitly describe the entire circuit in much less space. For the evaluation of the circuit, we only need to be able to describe the exact position of a vertex in the graph, and determine the immediate neighbors of a given vertex (previous and next vertices). This can easily be done in space $\tilde{O}(t/B + B^k)$.

In order to complete the simulation we need to show how to evaluate a small-depth circuit in small space (see Borodin [Bor77]).

Lemma 1. *Consider a directed acyclic graph G with one source (root). Assume that the leaves are labeled from $\{0,1\}$, its inner nodes are either AND or OR nodes and the depth is at most d. Then we can evaluate the graph in space at most $O(d)$.*

Proof. (of lemma. See [Bor77] for more details).

Convert the graph to a tree (by making copies of the nodes). The tree will have much bigger size but the depth will remain the same. We can prove (by induction) that the value of the tree is the same as the value of the graph from which we started. Evaluating the tree corresponds to computing the value of its root. In order to find the value of any node v in the tree, proceed as follows: Let $u_1, \ldots, u_k$ denote the child-nodes of v.

If v is an AND node, then compute (recursively) the value of its first child u_1. If $value(u_1) = 0$ then the value of v is also 0. Otherwise continue with the next child. If the last child has value 1 then the value of v is 1. Notice that we do not need to remember the value of the child-nodes that we have evaluated. If v is an OR node, the same idea can be applied. We can use a stack for the evaluation of the tree. It is easy to see that the size of the stack will be at most $O(d)$, that is as big as the depth of the tree. ∎

The total amount of space used is:

$$\tilde{O}(B^{2k} + \frac{t}{B}\log^l B) \tag{4}$$

To get the desired result, we need to choose the size B of the blocks appropriately to balance the two terms in (4). B will be $t^{1/c}$ for some constant c that is larger than k.

∎

As mentioned above, the exact same proof would work even if we allow almost linear depth for the non-uniform circuits for just linear deterministic time instead of $\mathcal{P}$. The stronger theorem is the following:

Theorem 3. *If* $\mathcal{DTIME}(n) \subseteq \mathcal{SIZE}$–$\mathcal{DEPTH}(n^k, n^\delta)$ *for some* $k > 0$ *and* $\delta < 1$, *then* $\mathcal{DTIME}(t) \subseteq \mathcal{DSPACE}(t^{1-\epsilon})$ *where* $\epsilon = 1 - \frac{1-\delta}{2k+1}$.

Proof. From the proof of theorem 2 we can calculate the space required for the simulation: In order to find the correct sub-circuit which has size B^k and depth B^δ, we need $O(B^{2k} \log B)$ space to write it down and $O(B^\delta)$ to evaluate it. To evaluate the entire circuit which has depth $\frac{t}{B} \cdot B^\delta$) we are only using space

$$O(\frac{t}{B} \cdot B^\delta \log B + \frac{t}{B} \log t + B^{2k} \log B) \tag{5}$$

The first term in equation (5), is the space required to evaluate the entire circuit that has depth $\frac{t}{B} \cdot B^\delta$ and the second and third term is the space required to write down an implicit description of the entire circuit (description of the graph from the block respecting computation, and the description of the smaller sub-circuit)

Total space used (to find the correct sub-circuit and to evaluate the entire circuit) is

$$O(\frac{t}{B} \cdot B^\delta \log B + B^{2k} \log B) \tag{6}$$

If we set $B = t^{1/2k+1}$ then the space bound is

$$O(t^{1-\frac{1-\delta}{2k+1}}) \tag{7}$$

In these calculations $2k + 1$ means just something greater than $2k$.

∎

These proof ideas seem to fail if we try to simulate non-deterministic time in small space. In that case, evaluating the circuit would be more complicated: we would need to use more space in order to make sure that the non-deterministic guesses are consistent throughout the evaluation of the circuit.

4 Semi-unbounded circuits

These simulation ideas using block respecting computation can also be used to prove an *unconditional* result relating uniform polynomial time and non-uniform small depth circuits. The simulation of the previous section implies unconditionally a trade-off type of result for the size and depth of non-uniform circuits that simulate uniform computations. The next theorem proves that any deterministic time t computation can be simulated by a non-uniform circuit of size $\sqrt{t} \cdot 2^{\sqrt{t}}$ or $2^{O(\sqrt{t})}$ and depth $\sqrt{t}$, which has "semi-unbounded" fan-in. Previous work by Dymond and Tompa [DT85] also present similar results showing that deterministic time t is in PRAM time $\sqrt{t}$.

Theorem 4. *Let t be a reasonable time complexity function. Then $\mathcal{DTIME}(t) \subseteq \mathcal{SIZE}$–$\mathcal{DEPTH}(2^{O(\sqrt{t})}, \sqrt{t})$, and the simulating circuits require exponential fan-in for AND gates and polynomial for OR gates (or vice-versa)*

Proof. Given a Turing machine running in $\mathcal{DTIME}(t)$, construct the block respecting version, and repeat the exact same construction as the one presented in the proof of theorem 2: Construct the graph describing the block respecting computation, which has t/B nodes, and every node corresponds to a segment of B (we will chose the size B later in the proof) computation steps. Use this graph to construct the non-uniform circuit: For every node, build a circuit, say in DNF, that corresponds to the computation that takes place on that node. This circuit has size exponential in B in the worst case, $2^{O(B)}$, and depth 2. The entire graph describes a circuit of size $\frac{t}{B}2^{O(B)}$ and depth $O(B)$. Also, note that for every sub-circuit that corresponds to each node, the input gates (AND gates as described in the proof) have a fan-in of at most $O(B)$, while the second level might need exponential fan-in. This construction yields a circuit of "semi-unbounded" type fan-in. ∎

5 Discussion - Open Problems

In this work we have shown a connection between the power of non-uniformity and the power of space bounded computation. The proof of the main theorem is based on the notion of block respecting computation and various techniques for simulating Turing Machine computation. The main result states that if Polynomial time has small non-uniform depth then space can simulate deterministic time fast(-er). An interesting open question is to see if the same ideas can be used to prove a similar space simulation for non-deterministic time. It seems also possible that a result could be proved for probabilistic classes. A different approach would be to make a stronger assumption (about complexity classes) and reach a contradiction with some hierarchy theorem or other diagonalization result thus proving a complexity class separation.

Acknowledgments

We would like to thank Charlie Rackoff and Toni Pitassi for many discussions on these ideas. Also many thanks to Dieter van Melkebeek and Lance Fortnow.

References

[Bor77] A. Borodin. On relating time and space to size and depth. *SIAM Journal of Computing*, 6(4):733–744, December 1977.

[DT85] Patrick W. Dymond and Martin Tompa. Speedups of deterministic machines by synchronous parallel machines. *Journal of Computer and System Sciences*, 30(2):149–161, April 1985.

[HPV77] J. Hopcroft, W. Paul, and L. Valiant. On time versus space. *Journal of the ACM.*, 24(2):332–337, April 1977.

[PF79] Nicholas Pippenger and Michael J. Fischer. Relations among complexity measures. *Journal of the ACM*, 26(2):361–381, April 1979.

[PPST83] Wolfgang J. Paul, Nicholas Pippenger, Endre Szemerédi, and William T. Trotter. On determinism versus non-determinism and related problems (preliminary version). In *24th Annual Symposium on Foundations of Computer Science*, pages 429–438, Tucson, Arizona, 7–9 November 1983. IEEE.

[PR81] W. Paul and R. Reischuk. On time versus space II. *Journal of Computer and System Sciences*, 22(3):312–327, June 1981.

[PTC77] Wolfgang J. Paul, Robert Endre Tarjan, and James R. Celoni. Space bounds for a game on graphs. *Mathematical Systems Theory*, 10:239–251, 1977.

[PV76] M. S. Paterson and L. G. Valiant. Circuit size is nonlinear in depth. *Theoretical Computer Science*, 2(3):397–400, September 1976.

[Sip86] M. Sipser. Expanders, randomness, or time versus space. In Alan L. Selman, editor, *Proceedings of the Conference on Structure in Complexity Theory*, volume 223 of *LNCS*, pages 325–329, Berkeley, CA, June 1986. Springer.

[Sip88] M. Sipser. Expanders, randomness, or time versus space. *Journal of Computer and System Sciences*, 36:379–383, 1988.

[SSZ98] Michael Saks, Aravind Srinivasan, and Shiyu Zhou. Explicit OR-dispersers with polylogarithmic degree. *Journal of the ACM*, 45(1):123–154, January 1998.

ECCC ISSN 1433-8092
http://www.eccc.uni-trier.de/eccc
ftp://ftp.eccc.uni-trier.de/pub/eccc
ftpmail@ftp.eccc.uni-trier.de, subject 'help eccc'