IAS/PCMI Summer Session 2000
Clay Mathematics Undergraduate Program
Advanced Course on Computational Complexity

Lecture 9: Towards Logspace Division

David Mix Barrington and Alexis Maciel
July 27, 2000

1. Overview

We continue our progress toward the result that integer division and related problems are in the class FOMP (first-order formulas with majority quantifiers and a numerical predicate for powering modulo short numbers) and thus in L-uniform TC^0 and in L itself. Last time we introduced Chinese Remainder Representation and showed that we can compute ITERATED MULTIPLICATION if the input and output are in CRR. By the end of this lecture we will have solved an important special case of DIVISION with input and output in CRR. This will require a series of technical results:

- We show how to compute (in FOMP) the *rank* of a number represented in CRR, which tells us the difference between the number calculated in the proof of the Chinese Remainder Theorem and our original number.
- We show how the rank function allows us to compare two numbers each represented in CRR (with the same modulus M) and tell which is larger.
- Given a number in CRR, we show how to find (in FOMP) its remainder modulo a *new* short prime number p, not one of the divisors of M. This will allow us to convert a CRR number to a new CRR representation with different moduli.
- Using this we solve the special case of DIVISION where the divisor Y is *nice*, meaning that it is the product of short prime powers.
- Finally we show that given an arbitrary Y, we can find a nice divisor D between $Y/2$ and Y. From the above we can find X/D, leaving us in our next lecture with the problem of computing (actually of approximating) the rational number D/Y. We indicate how this last step will finish the proof that DIVISION is in FOMP.

2. Computing the Rank of a CRR Number

Recall that if $X = \langle x_1, \ldots, x_k \rangle$ is a number in CRR_M, that is, CRR with moduli $m_1, \ldots, m_k$ where $M = m_1 \cdots m_k$, we chose numbers $D_1, \ldots, D_k$ so that each D_i is congruent to 1 modulo m_i and congruent to 0 modulo m_j for all $j \neq i$. X is congruent mod M to the number $\sum_i x_i D_i$, and the rank of X is defined to be the number r such that $X = (\sum_i x_i D_i) - rM$.

Davida and Litow recognized the importance of rank and gave an L-uniform NC^1 algorithm to compute it. Here we present an FOMP algorithm due to Macarie:

Theorem 1 *Given X in* CRR_M*, the rank of X can be computed in* FOMP*.*

Proof As we noted before, the rank is the floor of the rational number $\sum_i x_i(D_i/M)$. By the definition of D_i, this is equal to $\sum_i x_i(h_i/m_i)$ where h_i is the multiplicative inverse of the integer M/m_i modulo m_i. To calculate h_i, we must first use iterated product modulo m_i (in FOMP using discrete logs and iterated addition) to get M/m_i modulo m_i. Then the inverse may be obtained in $\mathrm{FO} + \mathrm{BIT}$.

For any number $c = O(1)$, we can approximate each rational number $x_i h_i/m_i$ to $c \log n + \log k$ bits of accuracy in $\mathrm{FO} + \mathrm{BIT}$, and then add these approximations in FOM. The error in each approximation is at most n^{-c}/k, so the overall answer is off by at most n^{-c}. In all likelihood this approximation will enable us to tell for sure what the floor of the real sum is, but we must guard against the possibility that this sum is very close to an integer. To do this we pick c so that n^c is considerably bigger than any of the m_i's, and reduce our problem to a simpler one in the case that the approximation might be wrong.

Let s be the integer with smallest absolute value such that $\sum_i x_i D_i$ is equal to s plus a multiple of M. (Thus s is either equal to $X = (\sum_i x_i D_i) - rM$ or is equal to the negative number $X - M$.) We call s the symmetric remainder of X modulo m.

Lemma 2 *If s is an integer such that $|s| < M/3m_k$, then s is also equal to the symmetric remainder of s modulo M/m_k.*

Proof Write s as $aM/m_k + b$ where b is the symmetric remainder modulo M/m_k. Note that $|b| \leq M/2m_k$. Even if s and b have their maximum possible absolute values, it is not possible for a to be nonzero. Therefore the equality holds. □

Our algorithm for the rank function is as follows. For every number j with $1 \leq j \leq k$, compute a rational approximation r_j for the rank as defined above with modulus

$M_j = m_1 \ldots m_j$. If r_k (the approximation for the original modulus M) is not within n^{-c} of an integer, then $\lfloor r_k \rfloor$ is our answer. Otherwise choose ℓ as large as possible so that r_ℓ is not within n^{-c} of an integer. By repeated application of the lemma, r_ℓ is just below an integer iff the true sum $\sum_{i=1}^{k} x_i D_i$ is just below an integer. We use this fact to determine whether $\lfloor r_k \rfloor$ or $\lceil r_k \rceil$ is the right answer, and then report that. The number ℓ must exist because $n^c > m_1$ and so r_1 at least cannot be close to an integer.

We have done only a polynomial number of FOMP calculations in parallel, followed by some FO + BIT calculation to determine our final answer. Thus this algorithm is in FOMP. □

3. Comparing Two CRR Numbers

Our next problem is to compare two numbers X and Y, each given in CRR_M. It is sufficient to consider the number $A = X - Y$ and determine whether it is above or below $M/2$. Of course, we can compute A in CRR_M by letting $a_i = x_i - y_i$.

The idea is to look at the number C whose CRR is $\langle c_i \rangle$, where c_i is equal to $2a_i$ if $2a_i < m_i$ or to $2a_i - m_i$ otherwise. Since C is congruent to $2A$ modulo M we know that either $C = 2A$ or $C = 2A - M$. The first case holds iff $A < M/2$, so we merely need to find out which case holds.

Let r be the rank of A and let s be the rank of C. By the definition of rank, $A = (\sum_i a_i D_i) - rM$, so $2A = (\sum_i 2a_i D_i) - 2rM$. Similarly, $C = (\sum_i c_i D_i) - sM$. The difference between $\sum_i c_i D_i$ and $\sum_i 2a_i D_i$ is exactly a term of the form $m_i D_i = h_i M$ each time that c_i is $2a_i - m_i$ rather than $2a_i$. Defining t to be the sum of the h_i in these "rollover cases", we have $C = (\sum 2a_i D_i) - tM - sM$. We can compute t easily given the values of the h_i's, and these are available in FOMP. All we need to do to determine whether $2A = C$, then, is to check whether $2r = s + t$. Since rank is computable in FOMP, we can decide this, and thus compare CRR numbers, in FOMP.

4. Changing the Moduli of a CRR Number

Suppose that X is a number in CRR_M, and that p is a short prime power that does not divide M. We have seen that in FOMP we can calculate iterated products modulo p using discrete logarithms (slightly modified if p is not prime) and ITERATED ADDITION. Thus we can compute the remainders modulo p of C_i, h_i, and thus D_i.

Then, again using ITERATED ADDITION, we get the remainder of $\sum x_i D_i$ modulo p. To find the remainder of X modulo p, all we have left to do is to subtract off the remainder of rM modulo p. Can we calculate rM modulo p in FOMP?

Yes. We saw above that we can get the actual number r in FOMP, and because r is short we can then get the remainder of r modulo p using only $\mathrm{FO} + \mathrm{BIT}$. M is an iterated product of some given numbers, and we have observed that in FOMP we can find this modulo p as well.

It follows from this result that if B is any product of short powers of distinct primes, we can take a number X in CRR_M and convert it to CRR_B. This will allow us to get started with DIVISION.

5. Dividing by a Nice Number

We are now finally prepared to carry out an important special case of DIVISION. Define a number to be *nice* if all of its prime power factors are short. If X is an arbitrary number given in CRR with respect to M, and B is nice, we will show how to compute $\lfloor X/B \rfloor$ in FOMP. (Note that this problem is only interesting if B is less than X and thus less than M.)

We have just seen that we can convert X from CRR_M to CRR_B. The remainder of X modulo B, which we will call E, is the unique number less than B that has this particular CRR_B form. By finding the rank of E with respect to B, we can now use the argument above to find the remainders of E modulo each of the numbers m_i and thus the CRR_M form of E. Subtracting this CRR_M from that of X, we get the CRR_M for the number $X - E$, which is a multiple of B.

It is easy in the same way to find the CRR_M for B. *If* B has an inverse modulo M, we can compute it in CRR_M and then compute $(X - E)B^{-1}$ in CRR_M, and we are done because $(X - E)B^{-1}$ is exactly $\lfloor X/B \rfloor$. Unfortunately things are a little more complicated if B and M share a prime factor.

In this case, let N be the product of all the prime divisors of M that are relatively prime to B. (Here is where we need the technical assumption mentioned earlier, that M is a product of distinct short primes and not just short prime powers.) We can certainly find B^{-1} modulo N in CRR_N, and then compute $(X - E)B^{-1}$ in CRR_N. This number is congruent modulo N to $\lfloor X/B \rfloor$, but is that good enough? *Because* M is the product of its prime divisors, we can say that $BN \geq M$ and thus M/B and hence X/B are smaller than N. So we have the CRR_N for the right number, and our only remaining problem is to get the CRR_M for it. We do this by finding its remainder modulo m_i for any of the m_i's that divide B. This requires finding the rank of the number modulo N and computing some more iterated products.

6. Finding a Nice Approximate Divisor

We can now divide by a nice number (though our result is in CRR rather than in binary). Of course our actual divisor Y may not be nice. Our strategy will be to find a nice number D that is close to Y, specifically, between $Y/2$ and Y. The rational number X/Y is exactly equal to $(X/D)(D/Y)$. In the next lecture we will show how to find a fraction N/A where A is nice and N/A is a very close approximation to D/Y. Then when we use "nice division" to get $\lfloor XN/AD \rfloor$, this will be within one of $\lfloor X/Y \rfloor$ and we can solve DIVISION (with input and output in CRR) by testing the possible quotients. (Recall, though, that with DIVISION in CRR we can extract any given bit of the binary form of a CRR number, and thus convert freely from CRR to binary. So we will then have solved the original DIVISION problem as well.)

To finish for today, then, we need to find this nice divisor that is between $Y/2$ and Y. We take a list of short odd primes and begin multiplying them together until their product exceeds Y. Then we remove the last prime and begin multiplying by two until the product exceeds Y. Removing the last two from the product gets our desired approximation, which is nice because it is a product of short primes and a power of two that is also short.

This search algorithm requires comparing the trial products with Y. Earlier in this lecture we gave an FOMP algorithm to compare any two numbers given in CRR_M. We are given Y in CRR_M, and we can easily compute the trial product in this form because we can perform iterated multiplication on numbers in CRR. We merely have to be sure that our list of primes is long enough that we can eventually exceed Y.

7. Exercises

1. If we can compare any two numbers in CRR_M, we can use binary search to get some of the high-order bits of the binary representation of a CRR_M number. How far will this method get us in FOMP, assuming we use comparison but not DIVISION?

2. Describe the CRR_M representation of the number $\lfloor M/2 \rfloor$ for general M (assuming that all the primes dividing M are odd). What is the remainder modulo each m_i?

3. If X and m are any numbers, the *symmetric remainder* of X modulo m is the unique number r that is congruent to X modulo m and that satisfies $-m/2 \leq r < m/2$. Show that CRR could equally well be defined with symmetric remainders in place of remainders, and that the rank problem reduces to finding whether the symmetric remainder of a long number modulo a short one is positive or negative.

4. If M is a product of short odd primes and X is any number such that $0 \leq X < M$, the inverse of X modulo M is the unique number Z (if any) such that XZ is congruent to 1 modulo M. Which numbers have inverses? Show that the CRR for the inverse, if it exists, can be computed in FO + BIT from the CRR for X.

5. Show that if X happens to be a short number, we can convert it from CRR to binary in FO + BIT.

6. How difficult is it to test whether a number is nice, or to find its prime factorization if it is? (We may assume that whenever we are "given a nice number" in our algorithms above, we are given its factorization.)