Division by four

Peter G. Doyle Cecil Qiu

Version 8A1 dated 10 April 2015
No Copyright[*]

Abstract

Write $A \preceq B$ if there is an injection from A to B, and $A \asymp B$ if there is a bijection. We give a simple proof that for finite n, $n \times A \preceq n \times B$ implies $A \preceq B$. From the Cantor-Bernstein theorem it then follows that $n \times A \asymp n \times B$ implies $A \asymp B$. These results have a long and tangled history, of which this paper is meant to be the culmination.

For John

1 The gist of it

To show: If there is a one-to-one map from $4 \times A$ to $4 \times B$ (which need not hit all of the range $4 \times B$), then there is a one-to-one map from A to B.
A word to the wise: Check out what Rich Schwartz has to say in [5].
Proof. Think of $4 \times B$ as a deck of cards where for each x in B there are cards of rank x in each of the four suits spades, hearts, gems, clubs. Note that while we use the word 'rank', in this game all ranks will be worth the same: Who is to say that a king is worth more or less than a queen?

Think of A as a set of racks, where each rack has four spots to hold cards, and think of $4 \times A$ as the set of all the spots in all the racks.

[*]The authors hereby waive all copyright and related or neighboring rights to this work, and dedicate it to the public domain. This applies worldwide.

Think of a one-to-one map from $4 \times A$ to $4 \times B$ as a way to fill the racks with cards, so that all the spots have cards, though some of the cards may not have been dealt out and are still in the deck.

Name the four spots in each rack by the four suits as they come in bridge, with spades on the left, then hearts, gems, clubs. Call a spade 'good' if it is in the spades spot of its rack, and 'bad' if not.

Do these two rounds in turn. (As you read what is to come, look on to where we have worked out a case in point.)

Shape Up: If a rack has at least one bad spade and no good spade, take the spade that is most to the left and swap it to the spades spot so that it is now good. Do these swaps all at the same time in all of the hands, so as not to have to choose which swap to do first.

Ship Out: Each bad spade has a good spade in its rack, thanks to the Shape Up round. Swap each bad spade for the card whose rank is that of the good spade in its rack, and whose suit is that of its spot. To see how this might go, say that in some rack the queen of spades is in the spades spot, while the jack of spades is in the hearts spot. In this case we should swap the jack of spades for the queen of hearts. (Take care not to swap it for the jack of hearts!) Note that some spades may need to be swapped with cards that were left in the deck, but this is fine. Do all these swaps at once, for all the bad spades in all the racks. This works since no two bad spades want to swap with the same card.

Note. If you want, you can make it so that when there is more than one bad spade in a rack, you ship out just the one that is most to the left.

Now shape up, ship out, shape up, ship out, and so on. At the end of time there will be no bad spades to be seen. (Not that all bad spades will have shaped up, or been put back in the deck: Some may be shipped from spot to spot through all of time.) Not all the cards in the spades spots need be spades, but no card to the right of the spades spot is a spade. So if we pay no heed to the spades spots, we see cards of three suits set out in racks with three spots each, which shows a one-to-one map from $3 \times A$ to $3 \times B$.

You see how this goes, right? We do a new pass, and get rid of the hearts from the last two spots in each rack. Then one last pass and the clubs spots have just clubs in them. So the clubs show us a one-to-one map from the set A of racks to the set B of ranks. Done.

Is this clear? It's true that we have left some things for you to think through on your own. You might want to look at [5], where Rich Schwartz has put in things that we have left out.

Here's the first pass in a case where all the cards have been dealt out. Note that in this case we could stop right here and use the spades to match A with B, but that will not work when A and B get big.

```
Start:
 4g    6h    Qg    8g    9h   *Qs*   4c    Ag    6c   *4s*
 Jh    Ah    9c    8h   *As*   Tc    Tg    5h    Qc   *Js*
 Kc   *6s*   4h    6g   *Ts*  *9s*   Jc    Kg   *8s*   8c
 5c    5g   *Ks*  *5s*   Th    Jg    Ac    Qh    9g    Kh

Shape up:
 4g   *6s*  *Ks*  *5s*  *As*  *Qs*   4c    Ag   *8s*  *4s*
 Jh    Ah    9c    8h    9h    Tc    Tg    5h    Qc   *Js*
 Kc    6h    4h    6g   *Ts*  *9s*   Jc    Kg    6c    8c
 5c    5g    Qg    8g    Th    Jg    Ac    Qh    9g    Kh

Ship out:
 4g   *6s*  *Ks*  *5s*  *As*  *Qs*   4c   *Ts*  *8s*  *4s*
 Jh    Ah    9c    8h    9h    Tc    Tg    5h    Qc   |4h|
 Kc    6h   *Js*   6g   |Ag|  |Qg|   Jc    Kg    6c    8c
 5c    5g   *9s*   8g    Th    Jg    Ac    Qh    9g    Kh

Ship out:
 4g   *6s*  *Ks*  *5s*  *As*  *Qs*   4c   *Ts*  *8s*  *4s*
 Jh    Ah    9c    8h    9h    Tc    Tg    5h    Qc   |4h|
*9s*   6h   |Kg|   6g   |Ag|  |Qg|   Jc   *Js*   6c    8c
 5c    5g   |Kc|   8g    Th    Jg    Ac    Qh    9g    Kh

Shape up:
*9s*  *6s*  *Ks*  *5s*  *As*  *Qs*   4c   *Ts*  *8s*  *4s*
 Jh    Ah    9c    8h    9h    Tc    Tg    5h    Qc   |4h|
 4g    6h   |Kg|   6g   |Ag|  |Qg|   Jc   *Js*   6c    8c
 5c    5g   |Kc|   8g    Th    Jg    Ac    Qh    9g    Kh

Ship out:
*9s*  *6s*  *Ks*  *5s*  *As*  *Qs*   4c   *Ts*  *8s*  *4s*
 Jh    Ah    9c    8h    9h    Tc   *Js*   5h    Qc   |4h|
 4g    6h   |Kg|   6g   |Ag|  |Qg|   Jc   |Tg|   6c    8c
 5c    5g   |Kc|   8g    Th    Jg    Ac    Qh    9g    Kh

Shape up:
*9s*  *6s*  *Ks*  *5s*  *As*  *Qs*  *Js*  *Ts*  *8s*  *4s*
 Jh    Ah    9c    8h    9h    Tc    4c    5h    Qc   |4h|
 4g    6h   |Kg|   6g   |Ag|  |Qg|   Jc   |Tg|   6c    8c
 5c    5g   |Kc|   8g    Th    Jg    Ac    Qh    9g    Kh
```

2 Discussion and plan

2.1 Division by any finite number

Write $A \preceq B$ ('A is less than or equal to B') if there is an injection from A to B. Write $A \asymp B$ ('A equals B', with apologies to the equals sign) if there is a bijection.

The method we've described for dividing by four works fine for any finite n, so we have:

Theorem 1. *For any finite n, $n \times A \preceq n \times B$ implies $A \preceq B$.*

From the Cantor-Bernstein theorem (Prop. 5 below) we then get

Theorem 2. *For any finite n, $n \times A \asymp n \times B$ implies $A \asymp B$.*

As an application of the Theorem 1, Lindenbaum and Tarski (cf. [4, p. 305], [8, Theorem 13]) proved the following:

Theorem 3. *If $m \times A \asymp n \times B$ where $\gcd(m, n) = 1$, then there is some R so that $A \asymp n \times R$ and $B \asymp m \times R$.*

We reproduce Tarski's proof in Section 6.2 below.

Combining Theorems 2 and 3 yields this omnibus result for division of an equality [8, Corollary 14]:

Theorem 4. *If $m \times A \asymp n \times B$ where $\gcd(m, n) = d$, then there is some R so that $A \asymp (n/d) \times R$ and $B \asymp (m/d) \times R$.*

2.2 Pan Galactic Division

We call the shape-up-or-ship-out algorithm for eliminating bad spades *Shipshaping*. As we've seen, Shipshaping is the basis for a division algorithm that we'll call *Pan Galactic Long Division*. As the name suggests, there is another algorithm called *Pan Galactic Short Division*, which we'll come to presently. 'Pan Galactic' indicates that we think this is the 'right way' to approach division, and some definite fraction of intelligent life forms in the universe will have discovered it. Though if this really is the right way to divide, there should be no need for this Pan Galactic puffery, we should just call these algorithms *Long Division* and *Short Division.* Which is what we will do.

2.3 What is needed for the proof?

Shipshaping and its associated division procedures are effective (well-defined, explicit, canonical, equivariant, . . .), and do not require the well-ordering principle or any other form of the axiom of choice. Nor do we need the axiom of power set, which is perhaps even more suspect than the axiom of choice. In fact we don't even need the axiom of infinity, in essence because if there are no infinite sets that all difficulties vanish. Still weaker systems would suffice: It would be great to know just how strong a theory is needed.

2.4 Whack it in half, twice

Division by 2 is easy (cf. Section 3), and hence so is division by 4:

$$4 \times A \preceq 4 \times B \implies 2 \times A \preceq 2 \times B \implies A \preceq B.$$

We made it hard for ourselves in order to show a method that works for all n. It would have been more natural to take $n = 3$, which is the crucial test case for division. If you can divide by 2, and hence by 4, there is no guarantee that you can divide by 3; whereas if you can divide by 3, you can divide by any n. This is not meant as a formal statement, it's what you might call a Thesis. We chose $n = 4$ instead of $n = 3$ because there are four suits in a standard deck of cards, and because there is already a paper called 'Division by three' [2], which this paper is meant to supersede.

2.5 Plan

In Section 3 we discuss division by two, and explain why it is fundamentally simpler than the general case. In Section 4 we introduce Short Division. In Section 5 we take a short break to play Pan Galactic Solitaire. In Section 6 we reproduce classical results on subtraction and division, so that this work can stand on its own as the definitive resource for these results, and as preparation for Section 7, where we discuss how long the Long and Short Division algorithms take to run. In Section 8 we discuss the tangled history of division. In Section 9 we wrap up with some platitudes.

3 Division by two

Why is division by two easy? The flippant answer is that $2 - 1 = 1$. Take a look at Conway and Doyle [2] to see one manifestation of this. Here is how this shows up in the context of Shipshaping.

The reason Shipshaping has a Shape Up round is that for $n > 2$, there can be more than one bad spade. When $n = 2$ there can't be more than one bad spade. In light of this, we can leave out the Shape Up rounds. When a bad spade lands in the hearts spot of a rack with a heart in the spades spot, we just leave it there. It's probably easier to understand this if we give up the good-bad distinction, and just say that the rule is that when both cards in a rack are spades, we ship out the spade in the hearts spot. At the end of time, there will be at most one spade in each rack, so in the Long Division setup there will be at least one heart; assigning to each rack the rank of the rightmost heart gives us an injection from racks to ranks.

In this approach to division by 2, we find that there is no need to worry about doing everything in lockstep. We can do the Ship Out steps in any order we please, without organizing the action into rounds. As long as any two-spade rack eventually gets attended to, we always arrive at the same final configuration of cards. This is the kind of consideration that typically plays an important role in the discussion of distributed computation, where you want the result not to depend on accidents of what happens first. It's quite the opposite of what concerns us here, where, in order to keep everything canonical, we can't simply say, 'Do these steps in any old order.' Without the axiom of choice, everything has to be done synchronously.

Now in fact the original Shipshaping algorithm works fine as an asynchronous algorithm for any n, but the limiting configuation will depend on the order in which swaps are carried out, so the result won't be canonical. More to the point, without the Axiom of Choice we can't just say, 'Do these operations in whatever order you please.' In contrast to the real world, where the ability to do everything synchronously would be a blessing, for us it is an absolute necessity.

4 Short Division

In Short Division we reverse the roles of A and B, so that A is the set of ranks and B the set of racks. An injection from $4 \times A$ to $4 \times B$ now shows a way to deal out all the cards, leaving no cards in the deck, though some of the spots in the racks may remain empty. We do just one round of Shipshaping. This works just as before, the only new twist being that if the spades spot is empty when we come to shape up a bad spade, we simply move the spade over into the spades spot. Since now all the cards have been dealt out, we don't ever have occasion to swap with a card still in the deck.

When A is finite, all the spades will shape up, and show a bijection from ranks (A) to racks (B).

When A is infinite, some of the bad spades may get 'lost', having been shipped out again and again without ever shaping up. These lost spades will each have passed through an infinite sequence of spots, and all these infinite sequences will be disjoint. We use these sequences to hide the lost spades, as follows.

Let $A_{\mathbf{good}}$ denote the ranks of the good spades at the end of the game, and $B_{\mathbf{good}} \asymp A_{\mathbf{good}}$ the racks where they have landed. $A_{\mathbf{bad}} = A - A_{\mathbf{good}}$ is the the set of ranks of the lost spades. To each element of $A_{\mathbf{bad}}$ there is an infinite chain of spots in $3 \times B_{\mathbf{good}}$, and these chains are all disjoint. Using these disjoint chains we can use the usual 'Hilbert Hotel' scheme to define a bijection between $A_{\mathbf{bad}} \cup 3 \times B_{\mathbf{good}}$ and $3 \times B_{\mathbf{good}}$, i.e. we make $3 \times B_{\mathbf{good}}$ 'swallow' $A_{\mathbf{bad}}$. But if $3 \times B_{\mathbf{good}}$ can swallow $A_{\mathbf{bad}}$ then so can $B_{\mathbf{good}}$ (cf. Proposition 13 below): $A_{\mathbf{bad}} \cup B_{\mathbf{good}} \asymp B_{\mathbf{good}}$. So

$$A = A_{\mathbf{bad}} \cup A_{\mathbf{good}} \asymp A_{\mathbf{bad}} \cup B_{\mathbf{good}} \asymp B_{\mathbf{good}} \preceq B.$$

Note. To make apparent the sequences of cards accumulated by the lost spades, we can modify the game by making stacks of cards accumulate under the spades, to record their progress. The Shape Up round is unchanged, except that we swap the whole spade stack into the spades spot. In the Ship Out round, when a spade ships out it takes its stack with it, and places it on top of the card it was meant to swap with. Spades that eventually shape up will have finite piles beneath them, but lost spades will accumulate an infinite stack of cards.

5 Pan Galactic Solitaire

Let us take a short break to play Pan Galactic Solitaire.

Deal the standard 52-card deck out in four rows of 13 cards. The columns represent the racks, with spots in each rack labelled spades, hearts, diamonds, clubs going from top to bottom. (Though as you'll see it makes no difference how we label the spots, this game is suit-symmetric.) The object is to 'fix' each column so that the ranks of the four cards are equal and each card is in the suit-appropriate spot. We move one card at a time. There is no shaping up; shipping out swaps are allowed based on any suit, not just spades. So for example if in some column the 3 of hearts is in the hearts spot and the 6 of hearts is in the diamonds spot, we may swap the 6 of hearts for the 3 of diamonds.

We don't know a good strategy for this game. Computer simulations show that various simple strategies yield a probability of winning of about 1.2 per cent. Younger players find this frustratingly low, and either play with a smaller deck or allow various kinds of cheating. Older players are not put off by the challenge, and at least one has played the game often enough to have won twice. Though because the game recovers robustly from an occasional error, he cannot be certain that he won these games fair and square.

A couple of apps have been written to implement this game on the computer. In some versions if you click on the 6 of hearts (as in the example above) the app locates the 3 of diamonds for you and carries out the swap. This makes the game go much faster, but it is much less fun to play than if you must locate and click on the 3 of diamonds, or better yet, drag the 6 of hearts over onto the 3 of diamonds. One theory as to why the automated version of the game is less fun to play is that the faster you can play the game, the more frequently you lose.

This game is called Pan Galactic Solitaire from the conviction that something like it will have occurred to a definite fraction of all life forms that have discovered Short and Long Division.

6 Cancellation laws

In this section we reproduce basic results about cancellation. These results are all cribbed from Tarski [8], though the notation is new and (we hope) improved.

To simplify notation, write $+$ for disjoint union and nA for $n \times A$. (This abbreviation is long overdue.) $A - B$ will mean set theoretic complement, with the implication that B is a subset of A.

The results below guarantee the existence of certain injections and bijections, and the proofs are backed by algorithms. For finite sets these can all be made to run snappily.

6.1 Subtraction laws

The only ingredients here are the Cantor-Bernstein construction and the closely related Hilbert Hotel construction.

We start with Cantor-Bernstein.

Proposition 5 (Cantor-Bernstein)**.**

$$A \preceq B \wedge B \preceq A \implies A \asymp B.$$

Proof. For a proof, draw the picture and follow your nose (cf. [2]).

Here is a version of the proof emphasizing that the desired bijection is the pointwise limit of a sequence of finitely-computable bijections, which will be important to us in Section 7 below.

It suffices to show that from an injection

$$f : A \to B \subset A$$

we can get a bijection.

Say we have any function $f : A \to B$, not necessarily injective (this relaxation of the conditions is useful, so that we can think of finite examples). Every $a \in A$ makes a Valentine card. To start, every $a \in A - B$ gives their Valentine to $f(a) \in B$. Any $b \in B$ that gets a Valentine then gives their own Valentine to $f(b)$. Repeat this procedure ad infinitum, and at the end of the day every $b \in B$ has at least one Valentine, and every Valentine has a well-defined location (it has moved at most once). Let g associate to $a \in A$ the $b \in B$ that has a's Valentine. g is always a surjection, and if f is an injection, g is a bijection.

Here's a variation on the proof. Again each $a \in A$ makes a Valentine, only this time every $a \in A$ gives their Valentine to $f(a)$. Now any $b \in B$ that has no Valentine demands its Valentine back. Repeat this clawing-back ad infinitum, and at the end of the day, every Valentine has a well-defined location, and if f was injective, or more generally if f was finite-to-one, every $b \in B$ has a Valentine, and we're done as before. The twist here is that if f is not finite-to-one, at the end of the day some b's may be left without a Valentine. So they demand their Valentine back, continuing a transfinite chain of clawings-back. We may or may not be comfortable concluding that after some transfinite time, every $b \in B$ will have a Valentine. For present purposes we needn't worry about this, since when f is injective the fuss is all over after at most ω steps. $\square$

Notation. Write $A \ll B$ ('A is swallowed by B' or 'A hides in B') if $A + B \preceq B$. By Cantor-Bernstein this is equivalent to $A + B \asymp B$. Another very useful equivalent condition is that there exist disjoint infinite chains inside B, one for each element of A.

By repeated swallowing we have:

Proposition 6. *For any n, if*

$$A_i \ll B,\ i = 0, \ldots, n-1$$

then

$$A_0 + \ldots + A_{n-1} \ll B. \qquad \square$$

Here are two close relatives of Cantor-Bernstein, proved by the same back-and-forth construction.

Proposition 7.

$$A + C \preceq B + C \implies A - A_0 \preceq B,$$

where $A_0 \ll C$. $\square$

Proposition 8.

$$A + C \asymp B + C \implies A - A_0 \asymp B - B_0,$$

where $A_0, B_0 \ll C$. $\square$

Proposition 9.

$$A + C \preceq B + 2C \implies A \preceq B + C$$

Proof.

$$A + C \preceq B + C + C \implies A - A_0 \preceq B + C$$

with $A_0 \ll C$. So

$$A = (A - A_0) + A_0 \preceq B + C + A_0 \preceq B + C. \qquad \square$$

Proposition 10. *For finite $m < n$,*

$$A + mC \preceq B + nC \implies A \preceq B + (n-m)C.$$

Proof. The proof is by induction, and we can get the number of recursive steps down to $O(\log(m))$. $\square$

From this we get

Proposition 11. *For* $n \geq 1$

$$A + nC \preceq B + nC \implies A + C \preceq B + C. \qquad \square$$

From Cantor-Bernstein we then get

Proposition 12. *For* $n \geq 1$

$$A + nC \asymp B + nC \implies A + C \asymp B + C. \qquad \square$$

Here's the key result for Short Division:

Proposition 13. *For* $n \geq 1$

$$A \ll nC \implies A \ll C.$$

Proof. This is the special case of Proposition 11 when B is empty. Here we redo the proof in this special case, in preparation for the timing considerations of Section 7.

Think of nC as a deck of cards, where (i, c) represents a card of suit i and rank c. Since $A \ll nC$, there are disjoint infinite chains

$$s_a : \omega \to nC,\ a \in A.$$

Let $\alpha(a)$ to be the smallest i such that s_a contains infinitely many cards of suit i, and let ρ_a be the sequence of ranks of those cards. (For future reference, note that we could trim this down to the ranks of those cards of suit $\alpha(a)$ that come after the last card of a lower-numbered suit.)

Let

$$A_i = \{a \in A : \alpha(a) = i\}.$$

The infinite chains

$$\rho_a : \omega \to C,\ a \in A_i$$

are disjoint, so

$$A_i \ll C$$

So by Proposition 6,

$$A = A_0 + \ldots + A_{n-1} \ll C.$$

The required injection from $A + C$ to C is obtained as a composition of injections f_i which map $A+C$ onto $A-A_i+C$, leaving $A-A_i$ fixed. $\square$

Here's a result that will be handy when we come to the Euclidean algorithm.

Proposition 14. *For $m < n$*

$$A + mC \asymp nC \implies A + E \asymp (n-m)C,$$

where $E \ll nC$, and hence $E \ll C$.

Proof. From above we have

$$A \preceq (n-m)C.$$

Write

$$A + E \asymp (n-m)C,$$

so that

$$A + E + mC \asymp nC.$$

Since $A + mC \asymp nC$ this gives

$$E + nC \asymp nC \implies E \ll nC \implies E \ll C. \qquad \square$$

Finally, here's the result Conway and Doyle needed to make their division method work.

Proposition 15. *For $n \geq 1$,*

$$nA \preceq nB \;\wedge\; B \preceq A \implies A \preceq B,$$

and hence by Cantor-Bernstein $A \asymp B$.

Proof. Write

$$A \asymp B + C.$$

From

$$nA \preceq nB$$

and

$$A = B + C$$

we get

$$nB + nC \preceq nB,$$

i.e.

$$nC \ll nB.$$

But

$$nC \ll nB \implies C \ll nB \implies C \ll B,$$

so

$$A \asymp B + C \preceq B. \qquad \square$$

6.2 Division laws

Finally we come to division. We've already proved Theorems 1 and 2. All that remains now is Theorem 3.

Proof of Theorem 3. As you would expect, the proof is a manifestation of the Euclidean algorithm. If $m = 1$ we are done. Otherwise we will use the usual recursion.

$$mB \preceq nB \asymp mA.$$

Using division we get

$$B \preceq A.$$

Write

$$A \asymp B + C.$$

$$mB + mC \asymp mA \asymp nB.$$

From Lemma 14 we have

$$mC + E \asymp (n - m)B,$$

with $E \ll B$, and hence $E \ll A$. We think of E as 'practically empty'. If it were actually empty, we'd recur using C in place of A. Ditto if we knew that $E \ll C$. As it is we use $C + E$ in the recursion, hoping that this amounts to practically the same thing.

$$m(C + E) \asymp mC + E + (m - 1)E \asymp (n - m)B + (m - 1)E \asymp (n - m)B.$$

So by induction, for some R we have

$$C + E \asymp (n - m)R,$$

$$B \asymp mR.$$

Since $E \ll A$,

$$A \asymp A + E \asymp B + C + E \asymp mR + (n-m)R \asymp nR$$

Done.

We won't undertake to determine the best possible running time here. But in order to make sure it requires at most a logarithmic number of divisions, we will want to check that for the recursion we can subtract any multiple km from n as along as $km < n$. Here's the argument again, in abbreviated form.

$$kmB \preceq nB \asymp mA;$$

$$kB \preceq A;$$

$$A \asymp kB + C;$$

$$mkB + mC \asymp mA \asymp nB;$$

$$mC + E \asymp (n-km)B,\ E \ll B \preceq A;$$

$$m(C+E) \asymp mC + E + (m-1)E \asymp (n-km)B + (m-1)E \asymp (n-km)B;$$

$$C + E \asymp (n-km)R,\ B \asymp mR;$$

$$A \asymp A + E \asymp kB + C + E \asymp kmR + (n-km)R \asymp nR. \qquad \square$$

7 How long does it take?

7.1 The finite case

In Shipshaping, every swap puts at least one card in a spot where it will stay for good, so the number of swaps is at most $n|A|$: This holds both in the Long Division setup where nA is the set of spots, and in the Short Division setup where nA is the set of cards. For a distributed process where all Shape Up and Ship Out rounds take place simultaneously the number of rounds could still be nearly this big, despite the parallelism, because one bad spade could snake its way through nearly all the non-spades spots. If we simulate this distributed process on a machine with one processor, the running time will be $O(n|A|)$ (or maybe a little longer, depending on how much you charge for various operations).

Note. While $|B|$ might be as large as $n|A|$, nothing requires us to allocate storage for all $n|B|$ cards (in Long Division) or spots (in Short Divison).

For finite A, in Short Division all spades will shape up after one pass of Shipshaping, and show an injection from ranks to racks. So the running time for Short Division is $O(n|A|)$, running either as a distributed process or on a single processor. This is as good as we could hope for.

Still for finite A, Long Division with the naive recursion takes $n-1$ passes of Shipshaping. If we divide by 2 whenever an intermediate value of n is even, we can get this down to at most $O(\log(n))$ passes. The number of suits remaining gets halved at least once every other pass, so the total number of swaps over all rounds of Shipshaping will be at most

$$|A|(n+n+n/2+n/2+n/4+n/4+\ldots)=4n|A|.$$

Hence the total time for Long Division is $O(n|A|)$, running either as a distributed process or on a single-processor machine. This is the same order as for Short Division, though Short Division will win out when you look at explicit bounds.

7.2 The infinite case

For A infinite, to talk about running times we will need a notion of transfinite synchronous distributed computation. The general idea is to support taking the kind of pointwise limits that show up in Shipshaping, where in the limit the contents of each spot is well defined, as is the goodness (but not the location) of each spade. (See 7.7 below for more specifics.)

One round of Shipshaping will take time ω. For Long Division sped up as in the finite case so that as to take $O(\log(n))$ passes, the running time will be $\omega \cdot O(\log(n))$. For Short Division, the swallowing step can be implemented by a recursion which, like Long Division, can be sped up to take $O(\log(n))$ passes, so again the running time is $\omega \cdot O(\log(n)$.

Note. Here and throughout, we round infinite running times down to the nearest limit ordinal, so as not to have to worry about some finite number of post-processing steps. In the cases at hand these post-processing steps can be eliminated, in essence by running them simultaneously with the computation of the results to be post-processed, and taking a limit. We imagine that with an appropriate formulate of transfinite computation, the possibility of eliminating post-processing steps could hold in general.

The big advantage of Short Division stems from the fact that the swallowing can be sped up to run in time ω, in essence by running the steps of the recursion in tandem. And the swallowing can be configured to run simultaneously with the Shipshaping. Sped up in this way, Short Division can be done in time at most ω. We discuss this further in Section 7.5 below.

By contrast, we've never found a way to run the division stages of Long Division in tandem.

7.3 Dividing a bijection

To divide a bijection to get a bijection, we can simultaneously compute injections each way, and then combine them using the Cantor-Bernstein construction in additional time ω. (Cf. Proposition 5.) So if dividing an injection takes time ω, dividing a bijection takes time at most $\omega \cdot 2$.

Can this be improved upon? It is tempting to start running the Cantor-Bernstein algorithm using partial information about the two injections being computed, but we haven't made this work. The case to look at first is $n = 2$, where Long Division is all done after one pass of Shipshaping.

7.4 Speeding up Long Division

A division algorithm takes as input an injection $f_n : nA \to nB$, and produces as output an injection $f_1 : A \to B$. In the Long Division setup, where all the spots are filled though not all the cards need have been dealt out, one pass of Shipshaping gets us from f_n to f_{n-1}, or more generally, from f_n to $f_{n(k-1)/k}$, for any k dividing n. In particular, when n is even one pass gets us $f_{n/2}$. This allows us to get from f_n to f_1 in $O(\log(n))$ passes. As one Shipshaping pass takes time at most ω, this makes for a total running time of at most $\omega \cdot O(\log(n))$.

Various tricks can be used to cut down the number of passes in Long Division. We can run Shipshaping using any divisor of n we please, or better yet, run it simultaneously for all divisors. Knowing f_m for as many values of m as possible can be useful because if we know injections from m_kA to m_kB then we can paste them together to get an injection from mA to mB for any positive linear combination $m = \sum_k r_k m_k$. This pasting takes only finite time, which in this context counts as no time at all. Combining these observations we can shave down the number of Shipshaping passes needed, and in the process we observe some intriguing phenomena. But we can never get the number of passes below $\lceil \log_2 n \rceil$, which is at most a factor of two better than what we achieve with the naive method of dividing by 2 when any intermediate n is even.

7.5 Speeding up Short Division

For real speed, we use Short Division. In this setup, all the cards are dealt out, though not all the spots need be filled. As the Shipshaping algorithm runs, we observe a steadily decreasing collection of bad spades, together with steadily lengthening disjoint sequences in $(n - \{0\}) \times B$ telling the spots through which these bad spades have passed. In the limit, the bad spades that remain have wandered forever, and they index disjoint injective sequences in $(n - \{0\}) \times B$. From these sequences we can determine how to hide the lost spades, and as indicated above, we can arrange to compute where the spades go while the Shipshaping algorithm is running, so everything is finished in time ω.

Here's roughly how it works. As we run the Shipshaping pass for Short Division, the set of bad spades decreases, while the sequences of spots they have visited steadily increases. As these preliminary results trickle in, we can be computing how we propose to hide the shrinking set of bad spades among the growing set of good spades. Eventually every bad spade knows where it will go, as does every good spade. In the limit we have an injection from spades, and hence ranks, to racks.

The main thing missing here is how we end up hiding the lost spades, and how we compute this. The fundamental idea is to trim the sequence of spots visited by a lost spade down to the subsequence consisting of all the hearts visited, then all the diamonds after the last heart, then all the clubs after the last diamond. These trimmed sequences can be computed in tandem with Shipshaping, so they are ready for use after time ω. (See Section 7.6 below.)

Once these sequences have been computed, it is possible to compute an injection from A to B with a finite amount of post-processing, and in fact we can arrange to compute approximations that converge to this injection at the end of ω steps. This is illustrated in Section 7.6 below.

For now, let us content ourselves with showing that we can divide in time $\omega \cdot 2$. To see this, note that because the trimmed sequences are increasing, once we are done computing them we can determine the limiting suit of all these sequences by running along them, keeping track of the suit, which eventually stops changing. So after another ω steps we've computed what in the notation of Proposition 13 was the function α. Then we can distinguish the lost spades according to this limiting value. We first hide those lost spades that limit with hearts, using the disjoint infinite chains of hearts locations in these trimmed sequences. Then we hide those spades whose trimmed sequences end with diamonds, and then clubs. This hiding takes a finite number of post-processing steps, which we can disregard, for a total running time of $\omega \cdot 2$: ω to compute and trim the tracks of the lost spades; ω to determine the limiting suits of the lost spades; then a little post-processing.

To get the running time down to ω is a little trickier. The idea is that we start using the trimmed sequences before we are done computing them. Be aware that the injection we compute will be different from that just described, because of artifacts associated to the cards in the trimmed sequences that come before those of the limiting suit (e.g. a finite number of hearts coming before an infinite number of diamonds).

7.6 Chipshaping

Here is a variation on Shipshaping that incorporates the simultaneous determination of the trimmed sequences of spots that the lost spades have passed through.

In this variation, we begin by placing a red poker chip on top of each spade, which will serve as its representative during the Shape Up and Ship Out rounds. The red chips will move just as the spades would have moved in Shipshaping.

(If the racks are tilted, we'll have to lean the chip against the card, maybe it would be better to think of the cards laid out in rows as in Pan Galactic Solitaire.)

Now we do the following.

Shape Up: We swap cards to the spades spot just as in Shipshaping, only now it is the red chips that determine which swaps take place, rather than spades. When a chip swaps to the spades spot it brings its card with it, and gets replaced by the card in the spades spot (if there is such a card), which of course has no chip on it, since otherwise we would not have been permitted to do this swap.

Hose Down: Say that a card is *above its station* if it has a red chip on it, and is to the left of the spot designated for cards of its suit. (In the Solitaire layout the card really is above the spot where cards of its suit belong.) Note that no spade is ever above its station, while a non-spade that is in the spades spot and has a red chip on it is always above its station. Swap any such card to where it belongs, namely, the spot for cards of its suit in the rack for cards of its rank. This spot is guaranteed to exist and to have a white chip on it. When swapping the cards, leave the red chip where it is and remove the white chip.

Repeat Hose Down rounds until no card is above its station. The number of rounds is at most the number of Ship Out rounds that have taken place.

Ship Out: This is just as in Shipshaping, only now we move *only the red chip*, leaving a white chip in its place.

Now Shape Up, Hose Down as many times as needed, Ship Out, repeat. Of course we stop after finite time if at some point there are no red chips in non-spades spots.

In the limit, each bad spade will begin an infinite trail of white chips, which is precisely the trimmed sequence of the non-spades spots its red chip (which is now lost) has visited, and which the spade itself would have visited under Shipshaping.

Now we can proceed as described about to divide in time $\omega \cdot 2$. But here's how we can do better. After each hosing down we compute an injection from spaded racks (racks with spades in the left slot) to spades as follows. Start at the right end of the rack and scan left along the rack until you find a chip. If it covers a spade, that's your answer. If not, go to where the card underneath the chip belongs, and continue scanning left from there. Because the chips cover trimmed sequences, we move steadily leftward, and we encounter a spade after at most n steps. This gives, an injection from spaded racks to spades. In the limit, this is a bijection from the eventually spaded racks to spades. Turned around, this is a bijection from spades to eventually spaded racks, which form a subset of B. So we have determined a bijection from A to a subset of B.

7.7 Transfinite synchronous distributed computation

We're using here a loose notion of transfinite computation. Roughly speaking, we're imagining that we have a processor for each element of A and B. Each processor has some finite set of registers that can store the name of an element of A or B, and a finite set of ports. The processors can communicate by sending messages to a designated port of another processor; if two processors simultaneously attempt to send to the same port of another processor, the whole computation crashes. Certain registers may be designated as suitable for reading at limit times $\omega, \omega \cdot 2, \ldots$. If the state contents of one of these registers fails to take on a limiting value, the whole computation crashes.

This is as precise as we want to get here. Really, we're hoping to find that a suitable formulation has already been made and explored. It would be great if there is essentially one suitable formulation, but we are by no means certain of this.

8 Some history

Here's a brief rundown on the history of Theorems 1 and 2.

Theorems 1 and 2 follow easily from the well-ordering principle, since then $n \times A \asymp A$ when A is infinite. The well-ordering principle is a consequence of the power set axiom and the axiom of choice (which without the power set axiom may take several inequivalent forms). The effective proofs we are interested in don't use either axiom.

Briefly put, Bernstein stated Theorem 2 in 1905 and gave a proof for $n = 2$, but nobody could make sense of what he had written about extending the proof to the general case. In the 20's Lindenbaum and Tarski found proofs of Theorems 2 and 1 but didn't publish them. Lindenbaum died and Tarski forgot how their proofs went. In the 40's Tarski found and published two new proofs of Theorem 1. In the 90's Conway and Doyle found a proof, after finally peeking at Tarski [8]; based on what Tarski had written, they decided that their proof was probably essentially the same as Lindenbaum and Tarski's lost proof.

For the gory details, here is Tarski [8], from 1949. (We'll change his notation slightly to match that used here.)

> Theorem 2 for $n = 2$ was first proved by F. Bernstein [1, pp. 122 ff.]; for the general case Bernstein gave only a rough outline of a proof, the understanding of which presents some difficulties. Another very elegant proof of Theorem 2 in the case $n = 2$ was published later by W. Sierpinski [6]; and a proof in the general case was found, but not published, by the late A. Lindenbaum [4, p. 305]. Theorem 1 — from which Theorem 2 can obviously be derived by means of the Cantor-Bernstein equivalence theorem — was first obtained for $n = 2$ by myself, and then extended to

> the general case by Lindenbaum [4, p. 305]; the proofs, however, were not published. Recently Sierpinski [7] has published a proof of Theorem 1 for $n = 2$.
>
> A few years ago I found two different proofs of Theorem 1 (and hence also, indirectly, of Theorem 2). ... The second proof is just the one which I should like to present in this paper. It is in a sense an extension of the original proof given by me for $n = 2$, and is undoubtedly related to Lindenbaum's proof for the general case. Unfortunately, I am not in a position to state how close this relation is. The only facts concerning Lindenbaum's proof which I clearly remember are the following: the proof was based on a weaker though related result previously obtained by me, which will be given below ...; the idea used in an important part of the proof was rather similar to the one used by Sierpinski in the above-mentioned proof of Theorem 2 for $n = 2$. Both of these facts apply as well to the proof I am going to outline. On the other hand, my proof will be based upon a lemma ..., which seems to be a new result and which may present some interest in itself; it is, however, by no means excluded that the proof of this lemma could have been easily obtained by analyzing Lindenbaum's argument

Observe Tarski's delicate description of Bernstein's attempts at the general case. Conway and Doyle [2] were less circumspect:

> We are not aware of anyone other than Bernstein himself who ever claimed to understand the argument.

Around 2010 Arie Hinkis claimed to understand Bernstein's argument (see [3]). Peter proposed to Cecil that he look into this claim for his Darmouth senior thesis project. Between 2011 to 2013 we spent many long hours trying to understand Hinkis's retelling of Bernstein's proof, and exploring variations on it. In the end, we hit upon the proofs given here. These proofs are very much in the spirit of Bernstein's approach, but not close enough that we believe that Bernstein knew anything very like this, and we remain skeptical that Bernstein ever knew a correct proof. There are very many possible variations of Bernstein's general idea, a lot of which seem to *almost* work. Still, these two proofs can be seen as a vindication of Bernstein's approach. Our finding them owes everything to Hinkis's faith in Bernstein.

9 Valediction

Tarski [8, p. 78] wrote that 'an effective proof of [Theorems 1 and 2] is by no means simple.' We hope that you will disagree, if not from this treatment then from Rich Schwartz's enchanting presentation in [5].

The reason that division can wind up seeming simple to us is that combinatorial arguments and algorithms are second nature to us now. A different way of looking at the problem makes its apparent difficulties disappear.

For an even more persuasive example of this, consider the Cantor-Bernstein theorem, and the confusion that accompanied it when it was new. From a modern perspective, you just combine the digraphs associated to the two injections and follow your nose.

After a century of combinatorics and computer science, it's easy to understand how Pan Galactic Division works. Can we hope that someday folks will marvel at how hard it was to discover it?

References

[1] F. Bernstein. Untersuchungen aus der Mengenlehre. *Math. Ann.*, 61:117–154, 1905.

[2] Peter G. Doyle and John Horton Conway. Division by three, 1994, arXiv:math/0605779 [math.LO]. `http://arxiv.org/abs/math/0605779`.

[3] Arie Hinkis. *Proofs of the Cantor-Bernstein Theorem: A Mathematical Excursion.* Birkhauser, 2013.

[4] A. Lindenbaum and A. Tarski. Communication sur les recherches de la théorie des ensembles. *Comptes Rendus des séances de la Société des Sciences et des Letters de Varsovie, Classe III*, 19:299–330, 1926.

[5] Rich Schwartz. Pan Galactic Division. *Mathematical Intelligencer*, to appear, arXiv:1504.02179 [math.CO]. `http://arxiv.org/abs/1504.02179`.

[6] W. Sierpinski. Sur l'égalité $2m = 2n$ pour les nombres cardinaux. *Fund. Math.*, 3:1–6, 1922.

[7] W. Sierpinski. Sur l'implication $2m \leq 2n \to m \leq n$ pour les nombres cardinaux. *Fund. Math.*, 34:148–154, 1947.

[8] A. Tarski. Cancellation laws in the arithmetic of cardinals. *Fundamenta Mathematica*, 36:77–92, 1949.