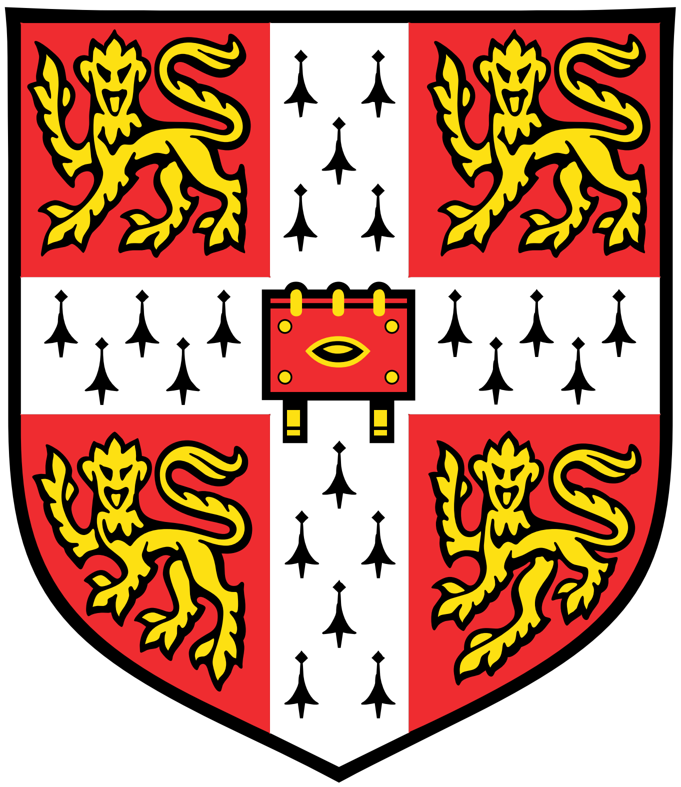
tf tries to explain the Priority Method to himself

tf (obviously!)

April 9, 2017



Theorem 1 *Friedberg-Muchnik*

There are $<_T$-incomparable degrees of semidecidable sets.

Proof:

We are trying to build two sets $A, B \subseteq \mathbb{N}$ such that neither is recursive in the other. In particular neither of them can be recursive *tout court*—decidable. Both A and B are constructed as a union of finite approximants: $\langle A_i : i < \omega \rangle$

and $\langle B_i : i < \omega \rangle$ and this will make them semidecidable, which is clearly the best we can hope for. The fact that neither A nor B are decidable means that the inclusion ordering on the A_i (resp. the B_i) is not end-extension, because that would mean A and B could be enumerated in increasing order and that would make them decidable. However the A_i *are* totally ordered by $\subseteq$. Do they start off empty, with $A_0 = B_0 = \emptyset$? A useful thought is that it actually doesn't matter a damn what finite sets A_0 and B_0 are. (In fact i'm pretty sure we can even take A_0 and B_0 to be decidable moieties, like the odds and the evens.)

We wish to ensure that for no e is $\{e\}^A$ the characteristic function for B nor is $\{e\}^B$ the characteristic function for A. For these purposes the characteristic function χ_A is the total version: $\chi_A(n) = \texttt{if } n \in A \texttt{ then } 1 \texttt{ else } 0$. The **requirements**[1] are $\chi_A \neq \{e\}^B$; $\chi_B \neq \{e\}^A$, for each $e \in \mathbb{N}$. Thus $\chi_A \neq \{e\}^B$ is the eth A-requirement, and $\chi_B \neq \{e\}^A$ is the eth B-requirement.

Each requirement $\chi_A \neq \{e\}^B$ is looking for a **witness**, an $x \in \mathbb{N}$ s.t. $\{e\}^B(x) = 0$ (which will say that x is not a member of the eth set computable from B) or $\{e\}^B(x)\Uparrow$. When this happens we can put x into A. Thus x will be the desired witness to the fact that $\{e\}^B$ is not χ_A. (*Mutatis mutandis* swapping A and B.) [You might think that x could be a witness (of sorts) if $\{e\}^B(x) = 1$ (which will say that x *is* a member of the eth set recursive in B)—so that we then make sure never to put x into A—but we never put things into $\mathbb{N} \setminus A$, only into A.]

The reason why the Friedberg-Muchnik theorem is such a big deal is that the requirements, at each stage of the construction, have access to only *part* of the set (A or B) relative to which they are computing. This results in the construction having a hesitant, *ratcheting* quality that can make it quite hard to describe.

I have found it helpful to think of each requirement as a kind of dæmon, a process with a job of work to do. The construction of A and B is the result of all these dæmons working independently and at their own pace, and in parallel. This makes it sound as tho' the project is asynchronous. Altho' that is probably the best conception to start off with we will eventually want to make it a clocked process (of length ω) not least so that it looks more deterministic and makes it more obvious that A and B are genuinely semidecidable.

Definition 1
Dæmons that are putting stuff into B are going to be called B-dæmons; dæmons that are putting stuff into A are going to be called A-dæmons.

Each dæmon has its own supply of potential witnesses. (It might need more than one, because things can go wrong, what with the dæmon having access only to *part* of A—or B.) This supply is in the form of a *stream*, a data object of the kind that will always respond when you say to it "give me your next member". The stream contains infinitely many natural numbers, and the natural numbers in each stream are stored in increasing order. The streams are also pairwise

[1]Not all terminology used in this exposition is entirely standard, but "requirement" is!

disjoint, for reasons that will emerge later. It may be worth noting at this point that, although the dæmon will use only a finite initial segment of its stream, there is no way of bounding in advance how large that initial segment will be. For the moment you need not worry why this might be so, just minute it, co's it might help in setting the stage.

(*À propos* of the observation that A_0 and B_0 don't have to be empty but could be decidable moieties ... what we really do need is that all the streams of candidate witnesses for the A-requirements should be disjoint from A_0 (and the B-streams similarly of course).)

0.0.1 What does the world look like to the dæmon $\{17\}^A \neq \chi_B$?

Imagine—for the sake of illustration—that you are the B-dæmon whose job it is to look after the B-requirement $\{17\}^A \neq \chi_B$. Your long-term aim is to find an input x which is a witness to the fact that $\{17\}^A \neq \chi_B$, which is to say that either

(i) $\{17\}^A(x)\Uparrow$; or

(ii) $\{17\}^A(x)\Downarrow = 1$;

(iii) $\{17\}^A(x)\Downarrow =$ something other than 0 or 1; or

(iv) $\{17\}^A(x) = 0$.

If (i) you are happy beco's $\{17\}^A$ is distinct from χ_B because χ_B is total and $\{17\}^A$ isn't. You don't need to do anything.

If (ii) you are happy.

Granted, if the result was 1 the possibility remains open that $\{17\}^A$ is a characteristic function, but then you can ostentatiously **not** put x into B. Yet again, you do nothing. This does the trick, since your stream of candidate witnesses is disjoint from everyone else's stream, so no-one else gets a chance to put x into B... so x is not in B, and is a witness to the distinctness of $\{17\}^A$ and χ_B;

Case (iii) If the result was anything other than 1 or 0 then evidently $\{17\}^A$ is not a characteristic function and so *a fortiori* is certainly not χ_B, and (as in case (i)) you don't need to do anything.

Case (iv) is where $\{17\}^A(x) = 0$. By now putting x into B you ensure that $\chi_B(x) = 1$ and therefore $\chi_B \neq \{17\}^A$.

Injury

If you look at the four possibilities considered in the preceding paragraph it seems that the dæmon is satisfied whatever happens—the four cases cover everything! The real problem with your life if you are this dæmon is that when you computed $\{17\}^A(x)$ you consulted A, but some A-dæmon wants to put something into A, which could invalidate the computation you performed of

$\{17\}^A(x)$. We have to find some way of mitigating this. This writing-stuff-into-A has to be done, of course, but in a controlled way.

Consider the wrangle going on between the two dæmons $\{0\}^A \neq \chi_B$ and $\{0\}^B \neq \chi_A$. Each might upset the other by writing stuff into the area that the other is trying to consult during the course of its computation. Every time one of them does that the other one has to restart. This could go on for ever! There has to be a way of preventing this loop of interference. That technique is a *hierarchy of importance* among dæmons.

Priority

The dæmon asks its stream for its first member, x, say. It then starts to compute $\{17\}^A(x)$, consulting A where necessary. If it halts and gets 1 it does nothing. If it halts and gets 0 it puts x into B. Its decision that $\{17\}^A(x) = 0$ was based on imperfect, incomplete information about A, and it wants to ensure that no other dæmon sabotage the assumption on which that decision relied. This of course cannot be ensured, because other dæmons are in the same boat, wanting to write stuff into A or B. What it *can* do is insist that dæmons of lower priority (a concept yet to be explained) are not allowed to put stuff into that initial segment of A that it has consulted.

Two things now happen.

$\{17\}^A \neq \chi_B$ has just put x into B, and may therefore have sabotaged some computations made by some A-dæmons that had been consulting B. These dæmons have been "injured" and have to restart.

So it "raises the barriers" for all A-dæmons (of lower priority) that might put stuff into A. ("raising barriers" is standard terminology.) A dæmon that is having its barrier raised experiences this as an instruction to discard that initial segment of its stream-of-potential-witnesses that contains numbers in the initial segment of A that $\{17\}^A \neq \chi_B$ has consulted. This raising-of-the-barriers for dæmons of lower priority ensures that the initial segment of A that the dæmon $\{17\}^A \neq \chi_B$ has just consulted is not altered. New stuff might be added to A, but it will consist of numbers bigger than any that $\{17\}^A \neq \chi_B$ enquired about. This "raising of the barriers" explains why no dæmon knows in advance how much of its stream-of-candidate-witnesses it might need.

Of course the dæmon $\{17\}^A \neq \chi_B$ might itself be interfered with ("injured") by A-dæmons of higher priority—that is to say it might be on the receiving end of the treatment we have just considered it dishing out to requirements of lower priority. There are two ways in which this might happen, depending on whether or not the dæmon $\{17\}^A \neq \chi_B$ was working at the time.

> (i) The dæmon $\{17\}^A \neq \chi_B$ might have completed a calculation and be lying on its back in the sun. It now has to start again from scratch with the next member of its own, personalised, stream of candidate witnesses. If you are this dæmon... don't even think about the previous candidate witness wot you put into B; water under the bridge. Let go. On the other hand

(ii) you might be in the middle of a calculation. You don't need to discard the argument you were working on but you do need to abort the current calculation and restart it using the updated version of A.

If you are an A-dæmon then all the things that might injure you and the things you might injure are B-dæmons; if you are a B-dæmon then all the things that might injure you and the things you might injure are A-dæmons. We absolutely have to ensure that each dæmon can be injured only finitely often. (After all, if it is interfered with (injured) infinitely often then it will never complete its task.) This requires some thought. After all, a dæmon that is allowed to injure you may itself be injured—by some still higher life form—and have to restart. So it may injure you more than once.

The best way to ensure that a dæmon is injured only finitely often is to ensure that each dæmon can be injured by only finitely many other dæmons and that the transitive closure of the "I might injure you" relation is wellfounded. If this condition is satisfied, fix a dæmon $\mathcal{D}$ and consider the tree of dæmons whose root is $\mathcal{D}$ and where there is an edge from a dæmon $\mathcal{D}_1$ to a dæmon $\mathcal{D}_2$ iff $\mathcal{D}_2$ can injure $\mathcal{D}_1$. This is a finite branching tree with no infinite paths and so must hold only finitely many dæmons. Every injury to $\mathcal{D}$ corresponds to a path through this tree, so there are only finitely many possible injuries to $\mathcal{D}$. Shades of König's Lemma.

Pretty clearly, among the binary relations one can impose on the set of dæmons, there are lots that have this property. We have to find one and stick too it. It seems that the usual way to go about it is to allow the dæmon $\{n\}^A \neq \chi_B$ to injure the dæmon $\{m\}^B \neq \chi_A$ as long as $n < m$ ...and analogously allow the dæmon $\{n\}^B \neq \chi_A$ to injure the dæmon $\{m\}^A \neq \chi_B$ as long as $n < m$. If $n = m$ then we have to make a choice. If $n = m$ we say that the A-dæmon can injure the B-dæmon but not the other way round.

Thus—in the usual scheme of things— any A-dæmon can injure any B-dæmon with a higher gnumber (the number inside the curly brackets) and if their gnumbers are the same the A-dæmon can injure the B-dæmon but not *vice versa.*

This gives us the ordering

$$\{0\}^A \neq \chi_B \ > \ \{0\}^B \neq \chi_A \ > \{1\}^A \neq \chi_B \ > \ \{1\}^B \neq \chi_A \ > \ldots$$

$$\ldots \{n\}^A \neq \chi_B \ > \ \{n\}^B \neq \chi_A \ > \{n+1\}^A \neq \chi_B \ > \ \{n+1\}^B \neq \chi_A \ > \ldots$$

However, life cannot be that simple. I have been describing this process as if it were asynchronous, but actually it can't be. Consider the B-dæmon $\{0\}^A \neq \chi_B$. It has highest priority—nothing is allowed to injure it. However ... beavering away in a back room somewhere is the A-dæmon $\{17\}^B \neq \chi_A$. It wants to put something into A. When we start the process, nobody has put anything into anything, and no barriers have been raised. $\{17\}^B \neq \chi_A$ might halt very quickly, and want to put something x into A. Since $\{17\}^B \neq \chi_A$'s barrier has not been raised, this x might be small enough for $\{0\}^A \neq \chi_B$ to take an interest, and $\{0\}^A \neq \chi_B$ might still be running. If the A-dæmon $\{17\}^B \neq \chi_A$ were to be allowed to write something in A while the B-dæmon $\{0\}^A \neq \chi_B$ were reading A then it would in effect be injuring $\{0\}^A \neq \chi_B$ and that is not to be borne.

This means that we have to *clock* the process somehow. We want a modification (clocking) along the following lines. You aren't allowed to put x into A or into B as soon as the thought occurs to you, but you have to wait until the next quarter-day when things get sorted out. The clock ticks are when A_n and B_n get updated to A_{n+1} and B_{n+1}. No injury or barrier pushing or anything takes places between clock ticks. Between clock ticks the dæmons go about their business in complete isolation.

What happens at the first clock tick? Suppose the dæmon $\{17\}^B \neq \chi_A$ has been computing $\{17\}^B(x)$ and got it to halt with output 0. $\{17\}^B \neq \chi_A$ will be itching to put x into A. Will this be allowed? That all depends on what $\{0\}^A \neq \chi_B$ has been doing. If $\{0\}^A \neq \chi_B$ has halted, it will want to freeze the part of A that it has been consulting. If x lies beyond that part then $\{17\}^B \neq \chi_A$ will be allowed to say that its value is 0 and it will be allowed to put x into A. If x is too small then the dæmon $\{17\}^B \neq \chi_A$ will be injured.

But what if $\{0\}^A \neq \chi_B$ is still running? Then we allow the A-dæmon of lower priority to write into A and tell $\{0\}^A \neq \chi_B$ to restart!

So the picture is this. When the clock ticks, every dæmon stops, and they all compare notes. The system managers then process the dæmons in decreasing order of priority. A dæmon that hasn't halted has nothing to contribute, so we ignore it (except we might raise its barrier).

Then we restart. In the case we have been considering, where $\{0\}^A \neq \chi_B$ was still running but $\{17\}^B \neq \chi_A$ had halted and wanted to put x into A we allow $\{17\}^B \neq \chi_A$ to put x into A. Thus, at the restart, the version of A that $\{0\}^A \neq \chi_B$ consults now contains x, whereas it hadn't done earlier. And it starts this computation from scratch of course.

This last detail should raise suspicions in the mind of the reader. If $\{0\}^A \neq \chi_B$ can be interrupted in this way, it might be that its first computation (with A_0) would have halted if it had been given the chance. Indeed, on the face of it, this could keep happening—in perpetuity, so that $\{0\}^A \neq \chi_B$ never halts, even tho (it seems) it should.

Let us think about this very carefully. Suppose that *every* time the bell strikes and every dæmon stops, that $\{0\}^A \neq \chi_B$ is still running, and that infinitely often stuff gets put into A by dæmons of lower priority. This had better imply that $\{0\}(x)$ diverges (x is the first member of this dæmon's stream) when given access to the whole of A. We prove the contrapositive.

Suppose that $\{0\}^A(x)$ halts. In so doing it has consulted only a finite amount of A. Now all thee naturals that $\{0\}^A(x)$ asks about have appeared by some finite time, in A_n, say. So what is preventiing it from halting? Reflect that from the nth clock tick on, it is starting (restarting) one and the same computation every time. It is working on the same argument x and the changes to A do not affect it. The way to ensure that it will halt is to ensure that the time interval between successive clock ticks increases without bound. Eventually the interval between successive clock ticks will be long enuff for $\{0\}^A(x)$ to halt.