Yet Another Relativization-Avoiding Diagonalization

Howon Lee

2022

Abstract

A method of diagonalization with contrary relativizations is given that is not the familiar $IP = PSPACE$-style arithmetization technique. Instead of computing wrong answers for queries, we compute the correct answers, but too slowly. We call it *stupid diagonalization* for humor value and for contrast to lazy diagonalization. Also given is a putative chain of reasoning to a class separation with great humor value known to have contrary relativizations, given a relaxation.

1 Introduction

Oracles do not actually exist. Methods of proof which are compatible with arbitrary oracles, like diagonalization by simulation of TMs, or *ordinary diagonalization*, are not sufficient to resolve the question of $P \stackrel{?}{=} NP$ and related questions because of contrary relativizations, as proved by Baker et al [3].

At the same time, Kozen [7] showed that any complexity class separation needs to be a member of a class of *generalized* diagonalization. Generalized diagonalization, as imagined by Kozen (or diagonalizations which are not necessarily simple, to use his words), is so general that this says less than one would imagine, really tantamount to "if you found a class separation, there is a function that can produce the separating member of that class".

Of course there are circuit complexity methods, which are not relativizing but which are bedevilled by Razborov et al's natural proof problem [11] and arithmetization techniques like those used for IP=PSPACE [13] which are bedevilled by Aaronson et al's algebrization [1].

But one may actually note that there are less-ordinary diagonalizations less common than arithmetization techniques. One is given by Fortnow et al [6]. Buss et al [4] proved that it had limits which have already been reached. Another putative less-ordinary diagonalization is given in this document.

The gist of the novel *stupid* diagonalization is to adversarially use resources *stupidly* by creating an adversarial *relation* that *knows about and wastes the time of* the TM producing the relation itself while getting the correct answer, rather than creating an adversarial TM that gets the wrong answer as in ordinary diagonalization. Hence the name. One is reminded of a labor union work-to-rule action, where the work eventually gets done but only after bringing up every rule possible to reduce productivity. Not to be confused with *lazy* diagonalization of Ladner et al [8].

Why this has contrary relativizations is because oracles are one step, so the chain of reasoning doesn't go through with certain oracles because they have *too much* numinous power and fail to waste enough resources.

2 Putative Chain of reasoning (Sketch)

We aim to look at function problems and nondeterminism and find that $FP_{stupid} \neq FNP_{stupid}$, where the complexity classes are touched by *stupidity*. We define *stupidity* below. Nearly the whole content of the chain of reasoning is in the definition of R, the separating relation. As mentioned above, the

definition of R depends upon the definition of the TM that calculates R, call it TM_R - which in turn depends upon the definition of R. Therefore R is defined in terms of itself, but at least indirectly.

Now, it's a pretty textbook notion [10] that the *search* version of $FP \stackrel{?}{=} FNP$ are closely related to the question of $P \stackrel{?}{=} NP$. But this is extremely fiddly upon the definition, so let us examine with respect to the definition of [12] and forget about P and NP for a bit.

Definition: A relation Q is in FP iff there is a deterministic polytime algorithm that, given an arbitrary input x, can find some y such that $(x, y) \in Q$.

Definition: A relation Q is in FNP iff there is a nondeterministic polytime algorithm that, given an arbitrary input x, can find some y such that $(x, y) \in Q$.

There is also a witness definition for FNP which we omit.

2.1 FP_{stupid} and FNP_{stupid}

Call FP_{stupid} and FNP_{stupid} variants of FP and FNP with the same relaxations on the definition of what *relation* is as noted in the introduction. That is, ordinarily the relation is implicitly not allowed to depend upon the TM that computes it, but this requirement is relaxed. We will be showing that $R \in (FNP_{stupid} \setminus FP_{stupid})$.

Note that there is no explicit requirement that the relation does not depend upon the TM that calculates it in the definitions given, but we shall posit that there is an explicit requirement, *and then* relax it, so I can sleep at night.

2.2 Defining R

WLOG, the alphabet of R, for inputs and outputs, is binary. If the input length of TM_R is held to be N, then R is a relation with arbitrary input producing a specific output y of length N per input. This output is chosen by a sort of keep-away with the TM_R and the input.

In order to possibly compute a putative output of R, TM_R has to have the string that represents the putative output on tape. Call that representation of the string on tape the *search leaf*.

We aim to adversarily demand by way of the definition of R a complete search from TM_R when TM_R is deterministic and to not do so if TM_R is nondeterministic - we want to get TM_R to look at all 2^N possible search leaves by writing them on the tape, but only in the deterministic case.

If the TM takes only deterministic steps (if the TM is in the set of TM's that computes members of FP_{stupid}), there exists a total ordering over the search leaves. Get the initial search leaf y_0 from simulating the TM_R on whatever input. Then, instead of making a contrary answer as in ordinary diagonalization, *fail to halt on the search leaf, despite the fact that it's written on the output tape.* Fail to halt in a similar manner for arbitrary search leaf y_i to get to search leaf y_{i+1}, with the condition that for all $j \in \{0...i\}$, y_{i+1} must be different from all y_j. The relation of the total ordering $<_{search}$ is whether the search leaf y_m is written on the output tape before another search leaf y_n.

If the TM takes nondeterministic steps (viz., if the transition relation is a non-function relation), there exists just a nontotal partial ordering over the search leaves $\leq_{search}$. This is because the TM advances multiple possible steps at a time, so we cannot have the previous total order relation $<_{search}$: the multiple possible steps are noncomparable with respect to each other.

But if we know that the ordering is a partial ordering which is not a total ordering, and cardinality of possible search leaves in a TM producing FNP_{stupid} is finite because the TM terminates, you can linearize the partial ordering to get a total ordering (taking a topological sort, if you think of the partial order as a DAG). All total orderings are also partial orderings, but only linearize if the ordering of search leaves is a partial ordering which is not a total ordering.

WLOG, our linearization of the nontotal partial order over the search leaves for TM in FNP_{stupid} is just lexicographical on the string the search leaf represents, instead of the bad-faith keep-away in the FP_{stupid} construction. You can use anything that's fast to find the last member of: if you do so, then things get very fast for the nondeterministic TM. But if the TM is deterministic we do not

linearize because we already have the total ordering, so the relation stays difficult for the deterministic TM.

Because there's a total ordering, which might already be present or that we might have made, and finite elements in the set of possible search leaves, there's a last search leaf looked at with respect to the total ordering for that input. Call the string corresponding to that last search leaf y_R. That is the output of R for any input.

2.3 $FP_{stupid} \neq FNP_{stupid}$

We know that R is in FNP_{stupid} because the nondeterministic machine can just go through the possible outputs and take the y_R that is last in the 'friendly linearized' (lexicographically linearized) partial order quickly, by construction.

Assert for contradiction's sake that R is in FP_{stupid}. Note that for the TM to compute R for input of length N, it needs to go through 2^N search leaves because if it went through any fewer it wouldn't be the last member of the total ordering $<_{search}$. But that busts out of polytime. Contradiction.

3 Putative Chain of Reasoning (Detail)

- *Definition*: A relation Q is in FP iff there is a deterministic polytime algorithm that, given an arbitrary input x, can find some y such that $(x, y) \in Q$. A deterministic polytime algorithm has a transition function for tape symbols and TM state.
- *Definition*: A relation Q is in FNP iff there is a nondeterministic polytime algorithm that, given an arbitrary input x, can find some y such that $(x, y) \in Q$. A nondeterministic polytime algorithm has a transition *relation* for tape symbols and TM state, which can take multiple values at the same time.
- There are multiple ways to define FNP, the above is the one we're working with right now.
- *Definition*: The process of turning a partial order into a total order is called *linearization*, or *linear extension*. All partial orders can be turned into a total order [9]. There are multiple ways to linearize in many cases. Note that the axiom of choice is needed for infinite cases.
- *Observation*: In order to compute the output of a computable relation, a TM has to have a representation of the putative output member of the relation on tape and *then* halt. Usually we think of this as one step, but it is trivially thought of as two steps (put a trivial state in going directly to the halting state in the TM).
- *Definition*: A representation of a putative output member of a relation being computed on tape we will call a *search leaf*.

3.1 FP_{stupid} and FNP_{stupid}

- Languages are defined over some alphabet Σ and are subsets of $\Sigma*$, where $*$ is the ordinary Kleene star operator. Relations are a subset of the Cartesian product of inputs and outputs, which are also defined over the Kleene star of some alphabet. So the notion of the subset is important here.
- What the notion of a subset is, is covered by the ZFC axiom of restricted comprehension, which they restricted to get rid of Russell's paradox. It goes, for some sets A and x and some putative subset B and some predicate ϕ,

$$\forall w_1, \ldots, w_n \, \forall A \, \exists B \, \forall x \, (x \in B \Leftrightarrow [x \in A \wedge \phi(x, w_1, \ldots, w_n, A)])$$

Now, nobody strictly said that this predicate ϕ cannot depend upon the machine that computes it, given that we're talking about computable relations. However, it really does seem unsporting to make that ϕ depend upon the machine that computes it in a complexity class.

- *Definition*: Therefore we state explicitly that we're imagining complexity classes where we can do that (make the predicate for a subset that's defined by the machine that computes it). That's FP_{stupid} and FNP_{stupid}, relaxations of the ordinary FP and FNP.

3.2 Towards defining R

We have to coerce the TM that calculates R into looking at 2^N search leaves if it's a deterministic TM but not if it's a nondeterministic TM. This is the reason we need the introspection of the TM, as well as for the iterative coercion itself.

- WLOG, the alphabet of R, for inputs and outputs, is binary. This is because you can expand into binary representations for everything.
- R is a relation with arbitrary input x producing a corresponding output y of length equal to the input, call that length N.
- The output is chosen by a keep-away with the TM_R and the input. You can think of an iterative construction with respect to the TM_R, detailed below.
- *Observation*: For a TM to compute a function, given input, is to have the output on tape while halting. Therefore, in order to compute a function on tape, TM_R has to have a representation of the *putative* output string on tape somewhere, the search leaf, and then it needs to *halt* on that putative output.
- Because we are talking of FP_{stupid} and FNP_{stupid}, not their non-stupid variants, we can look into the transition relations of the TM_R while defining R and see if they are non-function relations or relations which are functions. These correspond to nondeterministic and deterministic TM, respectively.
- We want to get TM_R to look at all 2^N possible search leaves by writing them on the tape in order to bust out of FP_{stupid}, but only in the deterministic case.
- In order to do so, we define a *case-based* total ordering $<_{search}$ over the search leaves which is adversarial in the deterministic case and not adversarial in the nondeterministic case.
- *Case*: TM_R transition relation is a function (TM_R deterministic)
  - In order to compute a relation (in this case the relation R), TM_R must write the output of the relation on tape and then halt. Note how this is two steps, if we're persnickety about it, which we will be.
  - An arbitrary x input is given. TM_R then does arbitrary polytime computations. In order to *have computed* the *initial putative R output*, TM_R writes down a putative output search leaf on the output tape, call it y_0.
  - 'In between' the writing down of the putative output search leaf y_0 and actually halting, we *perseverate*.
  - Unlike in ordinary diagonalization, where we are in a decision problem, simulate the TM and give a TM that outputs a contrary result, we instead give a TM that *refuses to halt here* and go and demand another output search leaf, if possible, despite the fact that we have y_0 already. Call this TM TM_1.

- Given y_i written in the output, we perseverate by not halting despite the fact that it's written down and demanding the next y_{i+1}, if possible, in TM_{i+1}.
- What is meant by "if possible"? Given all y_j, $j \in \{1...i\}$, we require of y_{i+1} that it not be the same as any search leaf y_j. Therefore the next y_{i+1} is impossible if the search leaf space is exhausted.
- When the search leaf space is exhausted, we halt and accept. So we do halt and accept, just after an incredible amount of procrastination, an incredible amount of perseveration.
- Note because of the iterative construction all TM_i have to write down the search leaves y_j, $j \in \{1...i\}$.
- This halts in finite alphabet size and finite length for search leaves because of the above uniqueness constraint.
- For TM_{2^N} with transition relation as a function (for deterministic TM_R, the total ordering $<_{search}$ is the y_j, ordered by j.
- *Observation*: This sometimes reminds me of the original Baker et al construction [3] for the relativized $P^O \neq NP^O$ world. But then again, sometimes it does not.

- *Case*: TM_R transition relation is a nonfunction relation (TM_R nondeterministic)
  - In the case of TM_R transition relation being a nonfunction relation, there is only a nontotal partial ordering on the possible output search leaves.
  - This is because there are multiple outputs from each input in nonfunction relations, so there can be multiple search leaves at once related to a previous search leaf.
  - Because there is only a nontotal partial ordering on the search leaves, if we want the total ordering we are free to linearize however we'd like, given that any partial ordering can be linearized.
  - As noted before, any partial ordering can be linearized with the axiom of choice, but a construction exists with finite orders, the topological sort. But we are free to choose among multiple linearizations, because there are multiple possible linearizations.
  - The linearization we want is a 'friendly' one that can be solved easily in nondeterministic polytime, in the case of nondeterminism of the TM.
  - We arbitrarily treat for our convenience $<_{search}$ in the nondeterministic TM case to be a lexicographical ordering on the search leaves' lexicographical content, where the search leaf is the string and the individual characters of the search leaf are the letters in the lexicographical ordering.

3.2.1 Definition of R

- Therefore, for machines in FP_{stupid} or FNP_{stupid} there is a total ordering on the search leaves $<_{search}$ (parameterized by the input).
- The cardinality of the search leaves is finite for any one machine and one input for that machine because we put a finite limit on the adversarial iterative construction, from the uniqueness constraint.
- Therefore there is a last member of the set of search leaves by that total ordering on that machine and input.
- The output of R on a given input x, denoted y_R, is that last member of the set of search leaves by that total ordering $<_{search}$ on that machine which is computing R for the input.

3.3 $FP_{stupid} \neq FNP_{stupid}$

(same as in sketch, reproduced here)

When TM_R is a nondeterministic machine, the machine can just go through the possible outputs and take the y_R that is last in the 'friendly linearized' (lexicographically linearized) partial order quickly, by construction, because the lexicographical ordering is known beforehands. WLOG in the case of binary search leaves and in the lexicographical ordering we mentioned we just output 1^N. Therefore, $R \in FNP_{stupid}$.

Assert for contradiction's sake that R is in FP_{stupid}. Note that for a deterministic TM to compute R for input of length N, it needs to go through writing 2^N search leaves because if it went through any fewer it wouldn't be the last member of the total ordering $<_{search}$. But that busts out of polytime. Contradiction.

(Note that if we can write new search leaves over tape from previous search leaves we don't necessarily need exponential space. But by construction, we do need to do all of that writing.)

4 Observations

4.1 Has the same phenomenon of contradictory relativizations as $P \stackrel{?}{=} NP$ itself

We give an oracle O such that with regards to the above chain of reasoning $FP^O_{stupid} = FNP^O_{stupid}$. The thing we get from the oracle is an excessively fast reduction in the set of search leaves.

- O takes a representation of the index $i \in 0..N$ of the one output of R (given an input) as a query. It outputs 1 if the output y_R of R at index i for the input is 1, 0 otherwise (Taking, WLOG, alphabet as $\{0, 1\}$).
- Ordinarily the computation would take the whole 2^n time to go through and get the last search leaf, and getting the contents of R without the running of TM_R would seem to not make sense.
- However, people prove stuff about oracles for, like, the halting problem, so despite the seeming contradiction it would seem that the oracle has numinous power and just gets that one index of the output of R in one step, skipping the adversarial lengthening of time.
- This way, the deterministic TM with oracle can just call O n times and produce the output in linear time, so we fail to bust out of polytime and the whole chain of reasoning of the previous sections doesn't work.

Of course note that null oracle is compatible with the chain of reasoning as is. So we are compatible with the known contrariness of relativizations for the related $P \stackrel{?}{=} NP$ because the chain of reasoning has contrary relativizations itself.

The usual counter to attacks on relativization (see [2]) is to find problems with space bounds, partial relativization or most importantly oracle access (see [5]). But you can have any kind of oracle access you'd like, the path by which we escape relativization is the *essential* property of oracles that they decide in one step and don't waste resources stupidly, and how that relates with the fundamental difference between the TM and NTM that the transition is a function in the TM and a nonfunction relation in the NTM.

4.2 Does not algebrize

We don't use arithmetization, instead issuing a total ordering on the search leaves to realize the separation.

4.3 Does not pretend that 2SAT, HORNSAT, XORSAT is in NP

This gets way more putative proofs of the related $P \stackrel{?}{=} NP$ than you would think, which is why I mention it especially. It would be good, when the counterexample is found, if it were less quotidian than the P-time SATs.

This is still a diagonalization. The produced example is not an instance of SAT except inasmuch as completeness makes all NP problems instances of SAT, and the Cook-Levin reduction, at least, always creates a big instance of SAT that is neither 2-SAT, HORNSAT or XORSAT or any other P-time SAT.

5 Conclusion

So why is it not a proof? It is the goal of computational complexity to go find out why we cannot have nice things with computers. But there is also a surfeit of evidence for the *empirical* corollary that computational complexity *itself* cannot have nice things.

Therefore I have *empirical* conviction that some cool peeps will go about and not only refute the above chain of reasoning, but show that this entire class of chains of reasoning, stupid diagonalizations, cannot lead to a proof, just as they did with arithmetization, with natural proofs, and with ordinary relativizable diagonalization itself. It's the thing to do. I eagerly await the counterexample.

Even given the inevitable failure, I tend to believe this would still be an interesting contribution. And some peeps might find some less humorous class separations with the underlying putative idea of diagonalization being a general class of bad-faith constructions not necessarily needing wrong answers. ¯_(ツ)_/¯

References

[1] Aaronson, S., and Wigderson, A. Algebrization: A new barrier in complexity theory. *ACM Transactions on Computation Theory (TOCT) 1*, 1 (2009), 1–54.

[2] Arora, S., Impagliazzo, R., and Vazirani, U. Relativizing versus nonrelativizing techniques: the role of local checkability.

[3] Baker, T., Gill, J., and Solovay, R. Relativizations of the $P \stackrel{?}{=} NP$ question. *SIAM Journal on computing 4*, 4 (1975), 431–442.

[4] Buss, S. R., and Williams, R. Limits on alternation trading proofs for time–space lower bounds. *computational complexity 24*, 3 (2015), 533–600.

[5] Fortnow, L. The role of relativization in complexity theory. *Bulletin of the EATCS 52* (1994), 229–243.

[6] Fortnow, L., Lipton, R., Van Melkebeek, D., and Viglas, A. Time-space lower bounds for satisfiability. *Journal of the ACM (JACM) 52*, 6 (2005), 835–865.

[7] Kozen, D. Indexings of subrecursive classes. *Theoretical Computer Science 11*, 3 (1980), 277–301.

[8] Ladner, R. E. On the structure of polynomial time reducibility. *Journal of the ACM (JACM) 22*, 1 (1975), 155–171.

[9] Marczewski, E. Sur l'extension de l'ordre partiel. *Fundamenta Mathematicae 16* (1930), 386–389.

[10] Papadimitriou, C. H. Complexity theory. *Reading: Addison Wesley* (1994).

[11] Razborov, A. A., and Rudich, S. Natural proofs. *Journal of Computer and System Sciences 55*, 1 (1997), 24–35.

[12] Rich, E. *Automata, computability and complexity: theory and applications.* Pearson Prentice Hall Upper Saddle River, 2008.

[13] Shamir, A. $IP = PSPACE$. *Journal of the ACM (JACM) 39*, 4 (1992), 869–877.