Relativization and Complexity

Rod Downey
Victoria University
Wellington

World Logic Day, January 2026

Relativization

- *" making something relative or considering it in relation to something else, rather than as absolute"*
  - We all "know" about relativity in physics.
  - In logic likely began with set theory, where we relativize formulae to some set A and then consider quantifiers ranging over A, so $\forall x$ becomes $\forall x \in A$.
  - According to sources I have read, likely goes back to Thoralf Skolem (1922) in his critique of axiomatic set theory, exploring implications of the Downward Löwenheim-Skolem Theorem.
  - The "Skolem Paradox". How can a countable model have uncountable sets? The answer being that relative to the model there is no counting of the sets.
  - Extensive use in set theory especially by Gödel in his work on the constructable universe.

Relativization in Computability

- Began with Turing 1939: "Systems of logic based on ordinals"
- But really developed with Post (1944), who defined degrees.
- In Turing 1939 Turing first defines oracle computations, and what we call Turing reducibility.
- Implicitly (and explicly in Post 1944) Turing defining the B-computable sets as $\{A \mid A \leq_T B\}$. Same effect as adding χ_B to Kleene (partial) recursive functions.

Reductions

- The notion of reduction is old in mathematics. For example a square matrix is singular iff its determinant is 0.
- All reductions used in classical computability theory, in, say, undecidability proofs, were *m*-reductions $A \leq_m B$ iff $x \in A$ iff $f(x) \in B$, where f is the computable function for the *m*-reduction.
- $\leq_T$, Turing reducibility has $A \leq_T B$ for B written on an oracle tape (Read only memory) and a computable finite procedure allowed to query B.
- Evidently $A \leq_m B$ implies $A \leq_T B$, but it's not hard to give examples to show this is proper.
- Classical computability has many reducibilities, especially after the seminal paper of Post 1944.

- For example truth table reducibility has $A \leq_{tt} B$ via $x \in A$ iff $B \models \sigma_{f(x)}$.
- This means, that, on imput x we specify a boolean combination of queries we'll make to B. That is the queries don't depend on B but are set in advance. An old result of Nerode is that $A \leq_{tt} B$ iff $A \leq_T B$ via some procedure Φ which is total for all oracles.
- Many others: Q-reducibility, Ziegler reducibility, enumeration reducibility, etc.

- Relativization in classical computability is putting B as an oracle all objects in the proof.
- For example, the undecidability of the halting problem $K = \{x \mid \varphi_x(x) \downarrow\}$.
- Suppose it was computable. Then let $f(x) = \varphi_x(x) + 1$, if $\varphi_x(x) \downarrow$ and $f(x) = 0$ otherwise. Then f is computable if we can decide whether $\varphi_x(x) \downarrow$. It is total, by construction and hence has an index z, but then $f(z) = \varphi_z(z) \downarrow = \varphi_z(z) + 1 \downarrow$, hence $0 = 1$.
- The relativization is that $B' \not\leq_T B$ for any B, where $B' = \{x \mid \varphi^B(x) \downarrow\}$, the jump of B.
- The proofs are to put B's on everything. So $f^B(x) = \varphi_x^B(x) + 1$, etc.
- This is referred to as "Relativizing the proof".

In classical computability, I

- The general leitmotif is that “everything relativizes”.
- The led to a number of conjectures.
- For example, the homogeneity conjecture (Rogers 1967), that if A and B are sets the the upper cones $\{X \mid A \leq_T X\}$ and $\{Y \mid B \leq_T Y\}$ are elementarily equivalent. (Earlier isomorphic, or with the jump added to the language).
- These were all shown to be false by Feiner in restricted form, and finally in full strength by Shore around 1979.
- But computability-theorists think that everything normal relativizes.
- The reason is that the degree lower cone $\mathbf{a} = \{X \mid X \leq_T A\}$ coincides with the sets computable relative to B.

Complexity theory

- In the 1960's authors such as Hartmanis and Sterns, Blum, Edmonds and others developed the basic of computational complexity.
- Add a counter to count steps in computation on a Turing machine and then analyse the asymptotic complexity of this process.
- Using this model, $A \in \mathrm{DTIME}(g)$ if if we can accept (i.e. compute $\chi_A(x)$ in $O(g(|x|)$ many steps.)
- This is regarded as robust according to an extension of Church's Thesis. That is Turing machines with counters suffice so long as we allow a polynomial overhead in translation between models.

- In particular A has a feasible algorithm means that $A \in P$, where $P = \cup_n \mathrm{DTIME}(|x|^n)$.
- All of the obvious problems with this where known to the wokers at the time (e.g. n^{10000}) but the model has nice closure properties and is mathematically robust.
- We then can also miniaturise the apparatus of computability theory, and define $A \leq_T^P B$ means that we can accept A in polynomial time with an oracle Turing machine with oracle B where "$z \in B$?" costs (either) 1 or $|z|$ depending on your model.
- $A \leq_m^P B$ means that there is a polynomial time computable f with $x \in A$ iff $f(x) \in B$.
- (This is the reduction that launched uncountably many NP-completeness results, BTW.)

Relativization in Complexity 1

- In the early days of computational complexity, it was observed that results relativized in a manner analogous to those used in computability theory.
- To wit, of we have a complexity class $\mathcal{C}$ with associated reductions $\leq_{\mathcal{C}}$ then $\mathcal{C}^B = \{A \mid A \leq_{\mathcal{C}} B\}$.
- In particular, $P^B = \{A \mid A \leq_T^P B\}$.
- Many things relativized using this formulation. For instance the Hierarchy Theorems: If f is $o(g)$, then DTIME(f) is strictly contained in DTIME(g), so, for example, $P^B \subset \mathrm{EXP}^B$ for all B.
- The next results came therefore as a big shock at the time.

NP

- Famously, Levin and Cook (1971) and Karp (1972) initiated studies into NP.
- Recall $L \in$NP iff there is a polynomial time relation R such that $x \in L$ iff $\exists^P y R(x, y)$. (That is there is a witness y of length $\leq |x|^c$ with $R(x, y)$.)
- Working with rubrik above, NP^B would be defined identically, except to make everyting $\leq_T^P B$ (i.e. $\exists^{P,B} y R^B(x, y)$.).

THEOREM (BAKER, GILL AND SOLOVAY, 1975)

There are computable oracles X and Y with

1. $P^X \neq NP^X$ *(Also by Ladner).*
2. $P^Y = NP^Y$ *(Also by Meyer and Fischer).*

- Their conclusion:

  "Methods that relativize (such as diagonalization, and simulation) are not sufficient to settle the P=?NP question"

One proof

- We construct X with $\mathrm{P}^X \neq \mathrm{NP}^X$ as it is useful for later.
- We build X by finite extension and diagonalization.
- We define $Z = \{1^n \mid \exists y[|y| = n \wedge y \in X\}$. Clearly $Z \in \mathrm{NP}^X$.
- We satisfy the requirements

$$R_{\langle e,c\rangle} : \Phi_e^X \neq Z \text{ in time } c|x|^c.$$

- To meet $R_{\langle e,c\rangle}$ we will have already met higher priority requirements, constructed X_s and be wanting to construct X_{s+1} extending it. We now choose a long length n, sufficiently long that

$$2^n >> cn^c.$$

- Let X_s^* be an empty extension of X_s of long length. Run $\Phi_e^{X_s^*}(1^n)$, and se what the answer is.
- If $= 1$ let $X_{s+1} = X_s^*$. If $= 0$, find a y of length n not addressed in the computation and put y into X_s^* to define X_{s+1}.

Aftermath

- After this there were many oracle results showing nothing can be settled. (see e.g. Fortnow's survey)
- Some results were found that broke the "oracle barrier"..
- E.g. Shamir IP=PSPACE. It was known that with the model above, $\mathrm{IP}^A \neq \mathrm{PSPACE}^A$ with probability 1.
- Shamir used arithmetization and algebra over a big finite field.
- Also algebraization, however claimed limitations due to "natural proofs".
- Both of these in some sense have been shown to fail to suffice to separate e.g. P from NP.

Computability Theory 2

- Do “all normal things” in computability relativize.
- Relativization concerns $\leq_T$, whereas moving to different concepts in computability shows that relativization is strongly tied to Turing reducibillty.
- On startling example is in algorithmic randomness. There are oracles A and B such that A and B differ by a finite amount and yet Ω^A is relatively random with Ω^B. (Downey, Hirschfeldt, Miller, Nies, 2005)
- Relevant to complexity theory is that, even in computability theory, relativization and strong reducibilities have a strained relationship.

A DEGREE EXAMPLE

THEOREM (MOHRHERR, 1984)

The truth table degees above $\mathbf{0}'_{tt}$ *are dense.*

THEOREM (KOBZEV, 1978)

There is a minimal c.e. truth table degree.

- Does Mohrherr's result mean that Kobzev's Theorem does not relativize?
- Well no. The problem is that we need to fully relativize the theorem. So Kobzev's Theorem relativizes to say that there is a c.e. relative to $\emptyset'$ tt-degree $\mathbf{a}$ which is a minimal cover of $\mathbf{0}'_{tt}$ in the $\leq_{tt}^{\emptyset'}$ degrees.
- What does $\leq_{tt}^{\emptyset'}$ mean? It means that on input x we compute $\sigma_{f(x)}$ but now $f(x)$ is an $\emptyset'$-computable function, rather than a computable one from the definition of $\leq_{tt}$.

Complexity theory 2

- The point is that strong reducibilities are much more like bounded reducibilities met in computational complexity theory.
- By identifying P^A as $\{X \mid X \leq_T^P A\}$ for example, are we capturing what we mean by polynomial time relative to A?.
- We we see some X in P^A we often tell students that if A was in P so would X. But we do this to languages A which are surely not in P.
- For example, if A is very complicated (e.g. double exponential time computable) , don't we really mean that from computing A in polynomial time we can solve X quickly.
- I believe that for a true relativization we should relativize what we mean by polynomial in the same was as for countable models of ZFC we need to relativize what we mean by uncountable relative to the model.

Relativizing P

- I offer some ideas below. I will do this for polynomial time, but clearly any reasonable time class could be approached using the same methods.
- Clearly any notion of P^A should include $\{X \mid X \leq_T^P A\}$.
- I think we should formalize this as any algorithm which computes A should allow us to compute X in no more that a polyomial amount more.
- The point here is that classical relativization in computability theory is concerned with non-computable things, whereas in complexity theory, we are concerned with things that are computable and hence we need to relativize both the access mechanism (as in the tt-case) and the notion of running time, and concern ourselves with sets that do have running times.

Possible definitons

Definition

1. $A \in P^B$ iff for all procedures Φ and computable g, if Φ accepts B in time $g(|x|)$, then there is an n and a procedure Ψ accepting A in time $O(g(|x|))^n$.
2. $A \in P^B_{eff}$ iff there is a functional Θ such that for all Φ and computable g with Φ accepting B in time $g(|x|)$, then $\Theta(\Phi, g, x)$ accepts $A(x)$ in time $O(g(|x|))^n$, for some n.

- Notice that for both of these if $B \in P$ and $A \in P^B$ or P^B_{eff} then $A \in P$.
- In both definitions, we are allowed to use the actual computation time in the computation of A. Notice that 2. above is a an actual reduction, whereas 1 only gives a relative computation times.

- We remark that there are precursors to these ideas with work of Maass and Slaman (complexity types of computable setsi, early 1980's) and Geske, Huynh and Selman, early 1970's (polynomially related complexities).
- To my knowledge nothing questioning the notion of relativization.
- Notice also that there is also a weaker effective connection:

DEFINITION

$A \in P^B_{weak,eff}$ iff for all Φ and computable g with Φ accepting B in time $g(|x|)$, there is a Θ such that then $\Theta(\Phi, g, x)$ accepts $A(x)$ in time $O(g(|x|))^n$, for some n.

Refinements

- Suppose that we want to attribute meaning to $A \in P^B$ is the sense outlined above. Consider 2. of Definition 4. Here we are considering a procedure Θ which uses as input a procedure Φ accepting B, and it could be that Φ might accept inputs for B in differing runtimes. We might have $|z| = |y|$, but $y \in B?$ might run in time $h(y)$ and $z \in B?$ in time $h(z)$. Perhaps we should have then following which we will refer to as $P^B_{eff,ref}$.

Definition

$A \in P^B_{eff,ref}$ iff there is a functional Θ such that for all Φ and computable h with Φ accepting $B(z)$ in time $h(z)$, then $\Theta(\Phi, g, x)$ accepts $A(x)$ in time $O(g(m))^n$, for some n, where $m = \max\{h(z) \mid z \in B? \text{ is queried in the computation}\}$.

Models

- In the classical model of relativization we imagine the oracle tape as having B written as read only memory on an oracle tape, in the case of computational complexity, on a multitape oracle turing machine, where oracle queries have unit cost. In this speculation, I would prefer to think of $z \in B?$ as costing the time it takes for Φ to accept z.
- In the first model we can think of two machines run in parallel where one is emulating Φ accepting B in time g, and the other is accepting A via Ψ, in time $O(g(|x|)^n)$. Φ acts like a clock and is not consulted during the computation of A, but the latter must halt in the number of steps specified.
- In the second model computing Θ we think of part of the machine computing $\Phi = \Phi_e$ via some set of tapes, and the result computes A via the remainder of tapes.

Definition (NP-1)

1. $A \in NP^B_{strong}$ iff for all nondeterministic procedures Φ and computable g, if Φ accepts B in time $g(|x|)$, then there is an n and a procedure Ψ accepting A in time $O(g(|x|))^n$.
2. $A \in NP^B_{strong,eff}$ iff there is a functional Θ such that for all Φ and computable g with Φ accepting B in nondeterministic time $g(|x|)$ $\Theta(\Phi, g, x)$ accepts $A(x)$ in time $O(g(|x|))^n$, for some n.

Definition (NP-2)

1. $A \in NP^B_{weak}$ iff for all deterministic procedures Φ and computable g, if Φ accepts B in time $g(|x|)$, then there is an n and a nondeterministic procedure Ψ accepting A in time $O(g(|x|))^n$.
2. $A \in NP^B_{weak,eff}$ iff there is a nondeterministic functional Θ such that for all Φ and computable g with Φ accepting B in nondeterministic time $g(|x|)$ $\Theta(\Phi, g, x)$ accepts $A(x)$ in time $O(g(|x|))^n$, for some n.

- Indeed we could also simply put the new notions of relativization to the the characterization/definition $\exists y(|x| \leq f(|x|) \wedge R(x, y))$ so that
- Exists y, $|y| \leq f(|x|)$ becomes one where $f \in P^B$ and similarly R becomes R^B, etc.

What about Relativization?

Proposition (Fortnow)

If $P = NP$ and B is a computable oracle, then according to any of the definitions above, $P^B = NP^B$.

Proof.

Suppose $P = NP$ and fix computable B and a Turing Machine M that computes B in time $t(n)$. If L is in NP^B then L is in $NTIME(\text{poly}(t(n))$ which is in DTIME(poly($t(n)$)) by $P = NP$ and padding. (That is, define $L' = \{(x, 1^{t(|x|)}) | x \text{ is in } L\}$. L' is in NP so L' is in P which implies L is in DTIME(poly($t(n)$).) Since this holds for all M, by Definition 4 $L \in P^B$ □

A Question

At this stage it is not clear whether separation propogates. That is, is the following true? If $P \neq NP$, and B is a computable oracle, then according to Definition 4, $P^B \neq NP^B$.

Technical Separations

The following proof even works for Definition 6.

Theorem

There are computable $A \in P^B$ such that $A \not\leq_T^P B$.

Proof

We build $A = \lim_s A_s$ and $B = \lim_s B_s$. We meet the requirements

$$P_e : A \not\leq_T^P B \text{ via } \Phi_e \text{ in time } |x|^e.$$

We also build a collection of computable machines M_σ with running times Θ_σ, such that for suitably chosen σ, M_σ accepts B and

$$R_e : \Psi_e \text{ accepts } B \text{ in time } \Xi_e \text{ implies a.a. } x,\ \Xi_e(x) \geq \Theta_\sigma(x),$$

for suitably chosen σ of length e.
We use R_e to make sure that $A \in P^B$. To do this we need to build a procedure Λ which takes any procedure Ψ_e accepting B in time Ξ_e, and has $\Lambda(\Psi; x)$ accepting $A(x)$ in time polynomial in $\Xi_e(x)$, and do this by making it accept x in time polynomial in $\Theta_\sigma(x)$.

The construction works in substages $t \leq s$ at each stage s. To attend to R_t, we will examine at substage t each string x with $|x| = s$ we see if $\Psi_t(x)$ converges, and if so we will make sure that $B(x) \neq \Psi_t(x)$, and declare that R_e is no longer active. To work with many requirents of type R_e, we will examine all the active R_e on each x in substages $e \leq s$ in reverse order of priority. The fate of $B(x)$ is therefore decided by the highest priority active requirement we might diagonalize.

- Additionally, we need to attend the P_e. At stage s there will be a highest priority unmet P_e. Before we consider R_e we would first consider P_e. For this we would examine the values of $\Phi_e^{C_1}(1^n)$ and $\Phi_e^{C_2}(1^s)$. Here C_1 denotes $B_{s-1} * 1^s$ with an empty extension for all strings τ of length $\leq s^{e+1}$ and C_2 denotes $B_{s-1} * 01^{s+1}$ and similarly empty otherise.
- If both give answer 0, then this requirement wishes to set $A_s(1^s) = 1$. If either give answer 1, then chose one that does and declare that R_e wants to set B_s to be appropriate C_i. (Notice that this affects all strings of length up to s^e and hence the next s^e many stages, although whether this actually happes will only be determined by that stage, as we might see a higher priority R_e which declares otherwise. If this succeeds then at stage s^e we would declare that P_{e+1} becomes active.

- Thus we consider the strategies in reverse order of priority. Since we actually act for any requirement once, there is some σ of length e which codes the requirements R_j for $j \leq e$ which are infinitely examined by never acted on, and all P_k for $k \leq e$ are now met.
- The machine M_σ believes that at every stage $s >_\sigma$, only requirements of indices $> e$ will act, and it will be right. Thus once this σ has priority, we will never act to diagonalize Ψ_e and hence $\Theta_e(x)$ must be slower than the running time for M_σ thereafter on every x of length $> s_\sigma$.
- Λ simulates the construction to point σ and will know the fate of 1^d for all $d > s_\sigma$.

Conclusions

- Perhaps the Baker-Gill-Solovay conclusion should be:
  "Methods that partially relativize are not sufficient to settle the P=?NP question"
- There is nothing wrong with this but it points at a programme trying to understand what might relativize and not partially relativize.
- In particular, we should view BGS results as being about relationships between degree structures, the Polynomial time degrees and the nondeterministic polynomial time degrees. Not about relativization.
- In particular, we might explore the limits of Fortnow's observation.

- Unfortunately, if you try to use the Baker-Gill-Solovay proof methods, certainly the diagonal set is in P^B, but the obstacle to the proof seems to be something like P vs NP (unsurprisingly!)
- Explore other ideas of what we actually mean by relativization in the complexity setting. Bienvenu has suggested that we should apply a polynomial to the input first and then run the algorithm computing B second. This has not been explored.
- I should also mention that other authors have instead changed the kinds of allowable machines to limit relativization results; notably Ron Book and his co-authors. (Positive machines-e.g. “Controlled Relativizations of P and NP”.)

Thank You.