A New, Very Very Simple, Constructive Proof of Kannan's Theorem

Sunny Daniels
e-mail: sdaniels@lycos.com

Abstract:

I appreciate the feedback from Fortnow (the author of [1]) on my recent viXra article [2], and in particular him supplying me with a definition of the standard polynomial-time universal Turing machine[4]. In [3], I clarified the way in which my construction and proofs in [2] could be re-worked to be analogous to the more usual definition of a universal \Sigma_2^P Turing machine.

Shortly after publishing [2], I realised that my construction in [2] or [3] gives rise to a very, very simple new constructive proof of Kannan's theorem: much simpler than anything that has been published before, if I am not mistaken, and even simpler than my proposed "layer by layer" proof in [2]. I suspect that Fornow realised this soon after reading [2]: his e-mail correspondence with me seems to hint at this. If so, I appreciate him giving me the opportunity to pubish this new proof myself!

I present a sketch of this new proof here.

Introduction:

I appreciate the feedback from Fortnow (the author of [1]) on my recent viXra article [2], and in particular him supplying me with a definition of the standard polynomial-time universal Turing machine[4]. In [3], I clarified the way in which my construction and proofs in [2] could be re-worked to be analogous to the more usual definition of a universal \Sigma_2^P Turing machine.

Shortly after publishing [2], I realised that my construction in [2] or [3] gives rise to a very, very simple new constructive proof of Kannan's theorem: much simpler than anything that has been published before, if I am not mistaken, and even simpler than my proposed "layer by layer" proof in [2]. I suspect that Fornow realised this soon after reading [2]: his e-mail correspondence with me seems to hint at this. If so, I appreciate him giving me the opportunity to pubish this new proof myself!

A Sketch of the Proof:

I present a sketch of this new proof here. We will use the conventions in [3]; I think that it should be obvious to the reader as to how to adapt our proof to the conventions in [2], if there is any point at all in attempting to adapt our proof to the conventions in [2], which are now sort-of obsolete IMHO!

Let the exponent k, for which we want our language not to have circuits of size big oh of n^k, be given.

Let the language L_3 be as defined on page 2 of [3] (sorry, \Sigma^2 should have been \Sigma_2 in [3]):

L_3 = {(<M>,x,1^q) | M is a \Sigma_2 machine and M(x) accepts in P(q) steps}

We have not yet decided on a value for the polynomial P: we will discuss this very soon! We then suppose, for the sake of a contradiction, (I just found [5] with a Google search and looked at it to check that my memory of the terminology that I am using here is correct), that L_3 has circuits of size big oh of n^k.

Now, it is very well-known that SAT is in NP, and hence in \Sigma_2^P. Now we set the degree and coefficients (which we can take as positive) of P to be big enough to accommodate the running time of this SAT machine (so that the L_3 machine can't terminate the simulation of M before M finishes) and set M to be this SAT machine. Then, if we hardwire the resulting circuits to make them simulate SAT in an way analogous to "Proof that the Construction Works" in [3], we get a circuit family of size big oh of n^k for SAT.

Then, by the Karp-Lipton Theorem ([6] I think, cited by [7]) , the polynomial hierarchy collapses down to \Sigma_2^P, i.e. PH=\Sigma_2^P.

Now, by Kannan's Theorem ([8] I think, cited by [7])), there is a language in \Sigma_4^P that does not have circuits of size big oh of n^k: we can call this L_4. Now, since we have already established that the polynomial hierarchy collapses down to \Sigma_2^P under our assumption for the sake of a contradiction, L_4 is also in \Sigma_2^P. So if we initially set the degree and coefficients of P to be high enough to be able to simulate both the \Sigma_2^P machine for L_4 and the \Sigma_2^P machine for SAT, we can hardwire our circuits for L_3, in a way analogous to "Proof that the Construction Works" in [3], to produce a circuit family of size big oh of n^k for L_4.

But we have already established the fact that L_4 does not have circuits of size big oh of n^k. This gives the required contradiction.

Making Proof even more Constructive?:

I think that it is fair to regard this proof as being constructive. However, this proof does not, of course, explicitly give the value of P (degree and coefficients) that we need in our definition of L_3 to make the construction work. Explicitly constructing an NP machine for SAT, which I presume has been done before, and looking at the construction, should give the degree and coefficients of P required to make the simulation of SAT work.

I presume that it is easy to see, from analysis of Kannan's[8] construction of his \Sigma_4^P machine, how to derive an exact (not asymptotic) polynomial upper bound on its running time. However, I have not yet looked into this.

Then, I presume that it would be at least moderately easy to see, from careful analysis of Karp and Lipton's original proof[6], exactly what effect their simulation of a \Sigma_4 machine by a \Sigma_2 machine has on a polynomial upper bound on the running time.

If I am not mistaken, we can assume WLOG that all of the coefficients of both polynomials are non-negative, so we can simply add both polynomials together to give a usable value of P for our L_3 construction.

Do readers think that it is worthwhile for me (or someone else) to explicitly try to work out a usable value for P in our L_3 construction in this (or some other) way?

Conclusion:

So I think that we now have a very simple, intuitive, constructive proof of Kannan’s Theorem (much simpler than any previous ones) that we can make even more constructive (putting in an explicit construction of P) if we want!

Conjectured Possible Next Step: Replace [2] Constructions Completely

My proposed “layer by layer” proof in [2] and [3] is now obsolete, if I am not mistaken, because the sketch proof that I presented above is much simpler!

Acknowledgements:

1) I thank Professor Lance Fortnow, currently at Illinois Institute of Technology I believe, for his ongoing feedback (I have been corresponding with him about [2] by e-mail since I published it) on this.

2) I thank Dr Sione Ma’u, of the University of Auckland Mathematics Department, for his ongoing interest in my complexity theory research and ongoing logistical support with my communication about it with Professor Lance Fortnow.

3) I thank Professor Tava Olsen of the Melbourne Business School (in Australia: www.mbs.edu) for her ongoing interest in my Complexity Theory research, and her offer to try to read and understand my Complexity Theory research herself (although I think that she is not normally a Complexity Theory researcher herself).

References:

[1] https://blog.computationalcomplexity.org/2014/08/sixteen-years-in-making.html

[2] A Constructive Proof of Kannan’s Theorem Based Upon a Much Simpler Construction: https://vixra.org/abs/2307.0031

[3] My New Constructive Kannan’s Theorem Construction Using More Standard Conventions: https://vixra.org/abs/2308.0031

[4] https://blog.computationalcomplexity.org/2002/12/foundations-of-complexity-lesson-11-np.html

[5] https://www.math-cs.gordon.edu/courses/mat231/notes/proof-contradiction.pdf

[6] *Karp, R. M.; Lipton, R. J. (1980), "Some connections between nonuniform and uniform complexity classes", Proceedings of the Twelfth Annual ACM Symposium on Theory of Computing, pp. 302–309, doi:10.1145/800141.804678, S2CID 1458043.*

[7] https://en.wikipedia.org/wiki/Karp%E2%80%93Lipton_theorem

[8] *Kannan, R. (1982). "Circuit-size lower bounds and non-reducibility to sparse sets". Information and Control.* ***55*** *(1–3): 40–56. doi:10.1016/S0019-9958(82)90382-5.*