Loss Bounds and Time Complexity for Speed Priors: Supplementary Material

Daniel Filan **Jan Leike** **Marcus Hutter**
College of Engineering and Computer Science, Australian National University

Similar definitions for $S_{\textbf{Fast}}$ and S_{Kt}

Proposition 3.

$$S_{Fast}(x) \stackrel{\times}{=} \sum_{p \to x} \frac{2^{-2|p|}}{t(p,x)}$$

Proof. First, we note that for each program p and string x, if $p \to_i x$, then for all $j \geq i$, $p \to_j x$. Now,

$$\sum_{j=i}^{\infty} 2^{-j} \times 2^{-|p|} = 2 \times 2^{-i} \times 2^{-|p|}$$

$$\Rightarrow \sum_{i=1}^{\infty} 2^{-i} \sum_{p \to_i x} 2^{-|p|} \stackrel{\times}{=} \sum_{i=1}^{\infty} 2^{-i} \sum_{\substack{p \to_i x \\ p \not\to_{i-1} x}} 2^{-|p|} \tag{14}$$

since all of the contributions to $S_{\text{Fast}}(x)$ from program p in phases $j \geq i$ add up to twice the contribution from p in PHASE i alone.

Next, suppose $p \to_i x$. Then, by the definition of FAST,

$$\begin{aligned} & t(p,x) \leq 2^{i-|p|} \\ \Leftrightarrow\ & \log t(p,x) \leq i - |p| \\ \Leftrightarrow\ & |p| + \log t(p,x) \leq i \end{aligned}$$

Also, if $p \not\to_{i-1} x$, then either $|p| > i-1$, implying $|p| + \log t(p,x) > i-1$, or $t(p,x) > 2^{i-1-|p|}$, also implying $|p| + \log t(p,x) > i-1$. Therefore, if $p \to_i x$ and $p \not\to_{i-1} x$, then

$$i - 1 < |p| + \log t(p,x) \leq i$$

implying

$$-|p| - \log t(p,x) - 1 < -i \leq -|p| - \log t(p,x) \tag{15}$$

Subtracting $|p|$ and exponentiating yields

$$\frac{2^{-2|p|-1}}{t(p,x)} \leq 2^{-i-|p|} \leq \frac{2^{-2|p|}}{t(p,x)}$$

giving

$$2^{-i-|p|} \stackrel{\times}{=} \frac{2^{-2|p|}}{t(p,x)}$$

Therefore,

$$\sum_{i=1}^{\infty} 2^{-i} \sum_{\substack{p \to_i x \\ p \not\to_{i-1} x}} 2^{-|p|} \stackrel{\times}{=} \sum_{p \to x} \frac{1}{t(p,x)} 2^{-2|p|} \tag{16}$$

which, together with equation (14), proves the proposition. □

Note that in (14) equality actually holds up to a factor af 2, and the sides of (16) are within a factor of two of each other, meaning that $S_{\text{Fast}}(x)$ is actually within a factor of 4 of $\sum_{p \to x} 2^{-2|p|}/t(p,x)$.

Proposition 4.

$$S_{Kt}(x) \stackrel{\times}{=} \sum_{i=1}^{\infty} 2^{-i} \sum_{p \to_i x} 1$$

Proof. Using equation (15), we have that if $p \to_i x$ and $p \not\to_{i-1} x$, then

$$\frac{2^{-|p|-1}}{t(p,x)} \leq 2^{-i} \leq \frac{2^{-|p|}}{t(p,x)}$$

so

$$2^{-i} \stackrel{\times}{=} \frac{2^{-|p|}}{t(p,x)}$$

Summing over all programs p such that $p \to_i x$ and $p \not\to_{i-1} x$, we have

$$2^{-i} \sum_{\substack{p \to_i x, \\ p \not\to_{i-1} x}} 1 \stackrel{\times}{=} \sum_{\substack{p \to_i x, \\ p \not\to_{i-1} x}} \frac{2^{-|p|}}{t(p,x)}$$

Then, summing over all phases i, we have

$$\sum_{i=1}^{\infty} 2^{-i} \sum_{\substack{p \to_i x, \\ p \not\to_{i-1} x}} 1 \stackrel{\times}{=} \sum_{p \to x} \frac{2^{-|p|}}{t(p,x)} \tag{17}$$

Now, as noted in the proof of Proposition 3, if $q \to_i x$, then $q \to_j x$ for all $j \geq i$. Similarly to the start of that proof, we note that

$$\sum_{i=j}^{\infty} 2^{-j} \times 1 = 2 \times 2^{-i} \times 1$$

The left hand side is the contribution of q to the sum

$$\sum_{i=1}^{\infty} 2^{-i} \sum_{p \to_i x} 1$$

and the right hand side is twice the contribution of q to the sum

$$\sum_{i=1}^{\infty} 2^{-i} \sum_{\substack{p \to_i x, \\ p \not\to_{i-1} x}} 1$$

Therefore,

$$\sum_{i=1}^{\infty} 2^{-i} \sum_{p \to_i x} 1 \overset{\times}{=} \sum_{i=1}^{\infty} 2^{-i} \sum_{\substack{p \to_i x, \\ p \not\to_{i-1} x}} 1$$

which, together with (17), proves the proposition.

□

Again, the two sides of the 'equation' established in this proposition are within a factor of 4 of each other.

S_{Kt} is a speed prior

Proposition 6. *Let $x_{1:\infty} \in \mathbb{B}^\infty$ be such that that there exists a program $p^x \in \mathbb{B}^*$ which outputs $x_{1:n}$ in $f(n)$ steps for all $n \in \mathbb{N}$. Let $g(n)$ grow faster than $f(n)$, i.e. $\lim_{n\to\infty} f(n)/g(n) = 0$. Then,*

$$\lim_{n\to\infty} \frac{\sum_{p \xrightarrow[\geq g(n)]{} x_{1:n}} 2^{-|p|}/t(p, x_{1:n})}{\sum_{p \xrightarrow[\leq f(n)]{} x_{1:n}} 2^{-|p|}/t(p, x_{1:n})} = 0$$

where $p \xrightarrow[\leq t]{} x$ iff program p computes string x in no more than t steps.

Proof.

$$\lim_{n\to\infty} \frac{\sum_{p \xrightarrow[\geq g(n)]{} x_{1:n}} 2^{-|p|}/t(p, x_{1:n})}{\sum_{p \xrightarrow[\leq f(n)]{} x_{1:n}} 2^{-|p|}/t(p, x_{1:n})}$$

$$\leq \lim_{n\to\infty} \frac{\sum_{p \xrightarrow[\geq g(n)]{} x_{1:n}} 2^{-|p|}/g(n)}{2^{-|p^x|}/f(n)} \tag{18}$$

$$\leq \lim_{n\to\infty} \frac{f(n)}{g(n)} \frac{\sum_{p \to x_{1:n}} 2^{-|p|}}{2^{-|p^x|}} \tag{19}$$

$$\leq \lim_{n\to\infty} \frac{f(n)}{g(n)} \frac{1}{2^{-|p^x|}} \tag{20}$$

$$= 0$$

Equation (18) comes from increasing $1/t(p, x_{1:n})$ to $1/g(n)$ in the numerator, and decreasing the denominator by throwing out all terms of the sum except that of p^x, which takes $f(n)$ time to compute $x_{1:n}$. Equation (19) takes $f(n)/g(n)$ out of the fraction, and increases the numerator by adding contributions from all programs that compute $x_{1:n}$. Equation (20) uses the Kraft inequality to bound $\sum_{p \to x_{1:n}} 2^{-|p|}$ from above by 1. Finally, we use the fact that $\lim_{n\to\infty} f(n)/g(n) = 0$. □

Time complexity: Upper bounds

Theorem 14 (S_{Kt} *computable in doubly-exponential time*)**.** *For any $\varepsilon > 0$, there exists an approximation S^ε_{Kt} of S_{Kt} such that $|S^\varepsilon_{Kt}/S_{Kt} - 1| \leq \varepsilon$ and S^ε_{Kt} is computable in time doubly-exponential in $|x|$.*

Proof. We again use the general strategy of computing k PHASEs of FAST, and adding up all the contributions to $S_{Kt}(x)$ we find. Once we have done this, the other contributions come from computations with Kt-cost $> k$. Therefore, the programs making these contributions either have a program of length $> k$, or take time $> 2^k$ (or both).

First, we bound the contribution to $S_{Kt}(x)$ by computations of time $> 2^k$:

$$\sum_{p \xrightarrow[> 2^k]{} x} \frac{2^{-|p|}}{t(p, x)} < \frac{1}{2^k} \sum_{p \to x} 2^{-|p|} \leq \frac{1}{2^k}$$

Next, we bound the contribution by computations with programs of length $|p| > k$. We note that since we are dealing with monotone machines, the worst case is that all programs have length $k+1$, and the time taken is only $k+1$ (since, by the definition of monotone machines, we need at least enough time to read the input). Then, the contribution from these programs is $2^{k+1} \times (1/(k+1)) \times 2^{-k-1} = 1/(k+1)$, meaning that the total remaining contribution after k PHASEs is no more than $2^{-k} + 1/(k+1) \leq 2/(k+1)$.

So, in order for our contributions to add up to $\geq 1 - \varepsilon$ of the total, it suffices to use k such that

$$k = \left\lfloor 2(\varepsilon S_{Kt}(x))^{-1} \right\rfloor \tag{21}$$

Now, again since λ is finitely computable in polynomial time, we substitute it into equation (5) to obtain

$$S_{Kt}(x) \overset{\times}{\geq} \frac{1}{|x|^{O(1)} 2^{|x|}} \tag{22}$$

Substituting equation (22) into equation (21), we get

$$k \leq O(|x|^{O(1)} 2^{|x|})/\varepsilon \tag{23}$$

So, substituting equation (23) into equation (10), we finally obtain

$$\begin{aligned} \#\text{ steps} &\leq 2^{O(|x|^{O(1)} 2^{|x|})/\varepsilon} \left(\frac{O(|x|^{O(1)} 2^{|x|})}{\varepsilon} \right) + 2 \\ &\leq 2^{2^{O(|x|)}} \end{aligned}$$

Therefore, S^ε_{Kt} is computable in doubly-exponential time. □

Computability along polynomial time computable sequences

Theorem 19 (S_{Fast} *computable in polynomial time on polynomial time computable sequence*)**.** *If $x_{1:\infty}$ is computable in polynomial time, then $S^\varepsilon_{Fast}(x_{1:n}0)$ and $S^\varepsilon_{Fast}(x_{1:n}1)$ are also computable in polynomial time.*

Proof. Suppose some program p^x prints $x_{1:\infty}$ in time $f(n)$, where f is a polynomial. Then,

$$S_{\text{Fast}}(x_{1:n}) \geq \frac{2^{-2|p^x|}}{f(n)}$$

Substituting this into equation (11), we learn that to compute $S^\varepsilon_{\text{Fast}}(x_{1:n})$, we need to compute FAST for k PHASEs where

$$k \leq \left\lfloor \log(2^{2|p^x|} f(n)/\varepsilon) \right\rfloor$$

Substituting this into equation (10) gives

$$\begin{aligned}\#\text{ steps} &\leq 2^{\log(2^{2|p^x|} f(n)/\varepsilon)} (\log(2^{2|p^x|} f(n)/\varepsilon) - 1) + 2 \\ &= O(f(n) \log f(n)) = O(f(n) \log n)\end{aligned}$$

Therefore, we only require a polynomial number of steps of the FAST algorithm to compute $S^\varepsilon_{\text{Fast}}(x_{1:n})$. To prove that it only takes a polynomial number of steps to compute $S^\varepsilon_{\text{Fast}}(x_{1:n}b)$ for any $b \in \mathbb{B}$ requires some more careful analysis.

Let $\langle n \rangle$ be a prefix-free coding of the natural numbers in $2 \log n$ bits. Then, if $b \in \mathbb{B}$, then there is some program prefix p^b such that $p^b \langle n \rangle q$ runs program q until it prints n symbols on the output tape, after which it stops running q, prints b, and then halts. In addition to running q (possibly slowed down by a constant factor), it must run some sort of timer to count down to n. This involves reading and writing the integers 1 to n, which takes $O(n \log n)$ time. Therefore, $p^b \langle n \rangle p^x$ prints $x_{1:n}b$ in time $O(f(n)) + O(n \log n)$, so

$$\begin{aligned}S_{\text{Fast}}(x_{1:n}b) &\geq \frac{2^{-2|p^b \langle n \rangle p^x|}}{O(f(n)) + O(n \log n)} \\ &= \frac{1}{O(f(n)) + O(n \log n)} \frac{1}{n^4 2^{2|p^b| + 2|p^x|}} \\ &= \frac{1}{g(n)}\end{aligned}$$

for some polynomial g of degree 4 greater than the degree of f. Using equations (11) and (10) therefore gives that we only need $O(g(n) \log g(n)) = O(g(n) \log n)$ timesteps to compute $S^\varepsilon_{\text{Fast}}(x_{1:n}b)$. Therefore, both $S^\varepsilon_{\text{Fast}}(x_{1:n}0)$ and $S^\varepsilon_{\text{Fast}}(x_{1:n}1)$ are computable in polynomial time. □

Note that the above proof easily generalises to the case where f is not a polynomial.

Theorem 20 (S_{Kt} *computable in exponential time on polynomial time computable sequence*)**.** *If $x_{1:\infty}$ is computable in polynomial time, then $S^\varepsilon_{Kt}(x_{1:n}0)$ and $S^\varepsilon_{Kt}(x_{1:n}1)$ are computable in time $2^{n^{O(1)}}$.*

Proof. The proof is almost identical to the proof of Theorem 19: supposing that p^x prints $x_{1:n}$ in time $f(n)$ for some polynomial f, we have

$$S_{Kt}(x_{1:n}) \geq \frac{2^{-|p^x|}}{f(n)}$$

The difference is that we substitute this into equation (21), getting

$$k \leq \left\lfloor 2^{|p^x|+1} f(n)/\varepsilon \right\rfloor$$

and substitution into equation (10) now gives

$$\begin{aligned}\#\text{ steps} &\leq 2^{2^{|p^x|+1} f(n)/\varepsilon} \left(2^{|p^x|+1} f(n)/\varepsilon - 1\right) + 2 \\ &= 2^{O(f(n))}\end{aligned}$$

The other difference is that when we bound $S_{Kt}(x_{1:n}b) \geq 1/g(n)$, the degree of g is only 2 greater than that of the degree of f. Therefore, we can compute $S^\varepsilon_{Kt}(x_{1:n}0)$ and $S^\varepsilon_{Kt}(x_{1:n}1)$ in time $2^{n^{O(1)}}$. □