A Proof of the Low Basis Theorem

Duarte Maia

July 31, 2023

Lemma. Let $T \subseteq 2^{<\omega}$ be a tree which is computable in the Halting problem, by an oracle program P which is inversely monotone in the oracle, in the following sense: If $A \subseteq B$ are oracles, then $P^A \geq P^B$ pointwise.

Then, there is a computable tree T' which contains the same paths. Moreover, T' can be obtained effectively from P.

Proof: Let P be a program which, using the Halting problem as an oracle, returns whether a finite string σ is in T or not. Then, we determine whether σ is in T' as follows. First, we pick an increasing approximation to the Halting problem H which improves with depth, let's say $H_{\operatorname{len}\sigma}$. Then, we verify that, with this oracle, P thinks that σ, and all its initial segments, are in T.

The resulting set T' is computable. We verify that it is a tree. Indeed, suppose that

$$P^{H_{\operatorname{len}\sigma}}(\varepsilon) = P^{H_{\operatorname{len}\sigma}}(\sigma_0) = P^{H_{\operatorname{len}\sigma}}(\sigma_0\sigma_1) = \cdots = P^{H_{\operatorname{len}\sigma}}(\sigma) = 1. \tag{1}$$

Then, by monotonicity of P as a function of the oracle, all of the above equalities still hold when using any $n < \operatorname{len}\sigma$. Thus, in particular, any prefix of σ is in T'.

Now, let us verify that T and T' have the same paths. One inclusion is due to the fact that, due to a reasoning similar to the one we just did, $T \subseteq T'$, and so any path in T is also in T'. So, we focus on the opposite inclusion.

Let f be a path in T', and let f_n be its n-th initial segment. We wish to verify, for arbitrary n, that $P^H(f_n) = 1$. Note that the computation of $P^H(f_n)$, whatever the output may be, uses the oracle only finitely many times. Thus, there is some approximation H_N such that $P^{H_N}(f_n) = P^H(f_n)$. Now, since the entirety of f is in T', then in particular $f_{\max(N,n)}$ is in T', and so by definition we have $P^{H_{\max(N,n)}}(f_n) = 1$. By monotonicity in the oracle, we conclude that

$$P^H(f_n) = P^{H_N}(f_n) \geq P^{H_{\max(N,n)}}(f_n) = 1. \tag{2}$$

■

Theorem. Let $T \subseteq 2^{<\omega}$ be an infinite computable binary tree. Then, there exists a path A through T of low degree.

Proof: We will construct A at the same time as we conclude (or force) things about its jump. More precisely, we will iterate over all programs P that make use of an oracle, and whenever possible we will ensure that A is built such that P does not halt when given A as an oracle.

Given the computable tree T, and fixed an oracle program P, we construct the subtree T_0 given as follows. To decide whether a string $\sigma \in T$ is in T_0, we check whether P halts (on the empty tape) when given σ as an oracle, with the stipulation that if P tries to access past the bounds of σ, it crashes and loops indefinitely. If P^σ runs forever (or crashes), we declare that σ is in T_0. It is trivial that T_0 is a tree. Moreover, P^σ requires no oracle to execute, and so T_0 is computable in the Halting problem, and furthermore in a way that is inversely monotone in the oracle. Thus, there is a computable tree $T_0' \supseteq T_0$ whose paths are the same as those of T_0, and moreover by intersecting with T we may in fact assume that $T_0' \subseteq T$. (Note that intersecting with T does not remove any paths.)

Now that we have performed the construction in the previous paragraph, we construct A itself, by constructing a decreasing sequence T_n of infinite trees. The first tree is T itself. Then, we iterate over all oracle programs P, and at each step construct the computable subtree (say U_{n+1}) of the current tree (say T_n) given by the above procedure. Now, there are two cases. Either U_{n+1} is infinite, or U_{n+1} is finite, and with an oracle for the Halting problem it is possible to tell effectively which of these two cases holds.

If the first case holds, and U_{n+1} is infinite, then there is an oracle in T_n which causes P never to halt, and indeed any oracle in U_{n+1} does this. So we set $T_{n+1} = U_{n+1}$. If the second case holds, then any oracle in T_n *will* cause P to halt in finite time. Thus, we set $T_{n+1} = T_n$, confident that whatever future restrictions we place on A, P will halt when given A as an oracle.

We thus obtain a decreasing sequence of trees, all of which contain at least one path, and so we may find a path f which is in all of them. [Note: This path is unique, because we can, by considering specific programs P, force the process to make a decision on whether any particular $n \in \mathbb{N}$ is in A.] The jump of f is determined by the above process, which is effective in the Halting problem, and so f has low degree. ■