

Ideas and Techniques: Bounding Critical Triples

★ *Priority trees* are a way to organize priority arguments. Each node is assigned to a requirement based on its length, and has children named after the possible outcomes of the requirement's action. The children are ordered so that the stronger ones are to the left of the weaker ones (infinitary outcomes, which aren't settled at any finite stage of the construction, generally go on the left). At a given stage s of the construction one finite path of nodes will be *accessible*, meaning those nodes are allowed to act at stage s to try to satisfy their respective requirements. At the end of each stage, nodes weaker than the last accessible node are initialized, meaning their memories are wiped and afterward they only see numbers which are larger than the stage at which they were initialized. We define the *true path* as the leftmost path of nodes which are accessible infinitely often. One proves the nodes on the true path actually do what is needed to meet all the requirements, proving along the way that they are each initialized only finitely many times (in some cases one must prove that the true path is infinite and is actually the \liminf of the approximation).

★ *Permitting* is a way to build sets which are Turing-below some given set D (some constructions use permission even as they build D). The basic idea is to only allow x to enter A (where we want $A \leq_T D$) at stage s if after s some number below x enters D . This is modified so permission is actually given by some number below a value n entering D , where n is assigned to x in some recursive way. The general mode when (the enumeration of) D is given to you at the start is to say “ D has a certain property, and if I don't get enough permissions to build A the way I want it, I can contradict that property of D .” While waiting for D to change appropriately, numbers which we desire to put into A will sit in a permitting bin, where they may remain forever – we only need *enough* permissions, not in general *all* permissions.

★ In both directions of the equivalence between being not totally ω -c.e.¹ and bounding weak critical triples², we have a somewhat circular dependency: $A_i \leq_T B \oplus A_{1-i}$ for $i = 0, 1$. When building a WCT below the not totally ω -c.e. D , we use *continuous tracing* to maintain the dependency while avoiding breakage of B -computations. That is, to a number targeted for A_0 we assign a *trace* targeted for A_1 and vice-versa, until we have an entire *entourage* attached to the original follower. Only after we see the B computation halt can we safely assign a B -trace to the end of the entourage, capping it off.

In the other direction, showing any purported WCT beneath a totally ω -c.e. D is not actually one, we know the dependency must hold. We think of it then as *layers*, where the demarcations between layers are the uses of the back-and-forth reductions. Any change to one set can create a chain reaction of changes in the others, each succeeding change happening in a lower layer, called *peeling*. The trick is to set up enough layers that they cannot be peeled back far enough to damage a computation you are preserving. Since we do not know what D will do we cannot *know* how many layers we will need, but since D is totally ω -c.e. we can try all the ω -c.e. approximations and know one will work.

¹A function g is ω -c.e. if there is a recursive approximation $g(x, s) \rightarrow_s g(x)$ and a recursive function h such that the number of times $g(x, s) \neq g(x, s + 1)$ is bounded by $h(x)$. D has totally ω -c.e. degree if every $g \leq_T D$ is ω -c.e.

²A weak critical triple is incomparable A_0, A_1, B such that $B \oplus A_i \equiv_T B \oplus A_{1-i}$, and no $C \leq_T A_0, A_1$ is such that $A_0 \leq_T B \oplus C$. For a critical triple that last condition is changed to $C \leq_T A_0, A_1 \Rightarrow C \leq_T B$.