Pointers on Proofs

In a proof, we have to lay out our approach in detail, so one can follow our reasoning. We'll take the problem in the Example Homework Answer handout as an example. It asks you to prove that the set of unsafe protection systems is recursively enumerable.

In what follows, the proof is in red. How we get to each step is in the normal color.

Let's begin by remembering what *recursively enumerable* means. It simply means that the entities in question can be listed. Now we know where we're going.

We will use the construction in the proof of the HRU result. There, an access control matrix is mapped onto a Turing machine, and the right to be leaked corresponds to the halting state. So we'll work the problem in terms of executions of Turing machines. A protection system that leaks a right is unsafe; similarly, we will call a Turing machine execution that enters the halting state unsafe. If a right does not leak, the Turing machine execution will not enter the halting state. So, if the execution is not unsafe (i.e., safe), then it will not halt. If it is unsafe, it will.

Our first approach is to list all the Turing machine executions so we can run them. We can assign numbers arbitrarily, but that may result in two Turing machine executions that are the same having different numbers. So we need to tie the numbering to the Turing machine execution itself. Fortunately, there is an easy way to do this — use Gö]del numbers. These encode the symbols of the tape on the Turing machine, and hence if two outputs are the same, the Gödel numbers are the same. So our list consists of Turing machine executions in the order of their Gödel numbers.

Represent the set of all possible systems as a set of executions of Turing machines. Each such execution has a unique Gödel number. Order the executions by their Gödel numbers (each such execution is represented by TM_i, where i is the appropriate Gödel number).

We also have to define when an execution stops. When a Turing machine execution stops, it enters the halting state. We'll call this state q_f (the "f" for "final"). So in terms of this problem, TM_i stops if, and only if, it enters state q_f.

Now, TM_i halts in state q_f if, and only if, the right in question leaks; that is, if, and only if, the system halts in an unsafe state.

If it doesn't halt at a given time, it may halt later on, or it may never halt.

If TM_i does not halt, it may be safe (and so never halt), or it may simply not yet have reached its unsafe (halting) state.

It's tempting to run the first machine until the execution stops, and then the second, and so forth. But what happens if the first machine execution is safe? It will never halt, so we'll never get to the second execution.

This means we cannot serially execute the systems, proceeding to TM_{i+1} when TM_i halts. If we did that, and TM_i never halted, we would never begin executing TM_{i+1} and so could not enumerate the unsafe systems with Gödel numbers greater than i.

So we have to interleave machine executions. The obvious approach is to run the first machine for one step, then another, then run the second machine for one step, the first for another step, then the second for another step, then the third, and so forth. This is called a *diagonalization* approach. The advantage of this is that any machine that is unsafe and will eventually halt, and we can drop it from the pattern. But other, safe machines will not halt. Even so, every machine advances.

So, use a diagonizational technique. Execute the first instruction in TM_1. Execute the second instruction in TM_1. Execute the first instruction in TM_2. Execute the third instruction in TM_1. Execute the second instruction in TM_2. Execute the first instruction in TM_3. Execute the fourth instruction in TM_1. Execute the third instruction in TM_2. Continue this pattern of execution, indicated in the following picture by numbers representing the order in which the steps are executed:

TM_1	1	2	4	7	11	$\cdots$
TM_2	3	5	8	12	$\cdots$	
TM_3	6	9	13	$\cdots$		
TM_4	10	14	$\cdots$			
TM_5	15	$\cdots$				
$\cdots$	$\cdots$					

So we can enumerate the unsafe executions simply by recording the numbers of those that halt.

Now, when TM_i halts, it is added to the list of unsafe systems. The diagonalization procedure is modified to skip over the halted TM_i. Thus, all systems which halt in state q_f will be enumerated. So we can list the unsafe Turing machine executions.

Now we map this back into protection systems.

As we can list the unsafe Turing machine executions, we can list the corresponding protection systems in which the right leaks. Hence the set of unsafe protection systems is recursively enumerable.