Las tres leyes de la generación infinita

Para resolver un problema de generación infinita, es importante recordar estas tres leyes:

1. Nunca debe usarse más de un generador infinito.
2. El generador infinito siempre va a la izquierda de cualquier otro generador.
3. Los generadores infinitos deben usarse únicamente para generar infinitas soluciones[1].

Otra cosa a tener en cuenta: es importante es que las soluciones generadas en cada paso sean finitas, y que entre todas cubran todo el espacio de soluciones que se busca generar. Por ejemplo, si queremos generar todas las listas finitas de enteros positivos, no nos sirve que el generador infinito nos vaya dando la longitud de la lista, porque para cada longitud hay infinitas listas posibles. Tampoco nos sirve que vaya generando el primer elemento de la lista y que los demás estén acotados por el primero, porque entonces nunca generaríamos las listas cuyo primer elemento no es el máximo. En cambio la suma de los elementos de la lista sí es un buen valor para ir generando en cada paso, ya que hay una cantidad finita de listas para cada suma posible, y entre todas las sumas obtenemos todas las listas (además las soluciones en cada paso son disjuntas, lo cual siempre es útil, ya que no tenemos que ocuparnos de eliminar soluciones repetidas).

En caso de que haya más de una forma posible de ir recorriendo el espacio de búsqueda generando finitas soluciones en cada paso, conviene pensar cuál es la forma más sencilla. Por ejemplo, para generar árboles binarios, no es muy útil que el generator infinito genere la altura, ya que la altura de un subárbol no determina la del otro. No hay una receta para resolver todos los problemas posibles, pero si ven que el problema de cómo generar las soluciones en cada paso es muy complejo, probablemente haya otra forma de usar el generador infinito que divida el espacio en particiones más sencillas de generar.

[1] La tercera ley puede tener excepciones en Haskell si el problema se salva con la evaluación Lazy, pero siempre es importante asegurarse de que el programa no se cuelgue al terminar de generar las soluciones que queremos (y, obviamente, tampoco antes).