Program slicing is a well known family of techniques used to identify code fragments which depend on or are depended upon specific program entities. They are particularly useful in the areas of reverse engineering, program understanding, testing and software maintenance. Most slicing methods, usually oriented towards the imperative or object paradigms, are based on some sort of graph structure representing program dependencies. Slicing techniques amount, therefore, to (sophisticated) graph transversal algorithms. This paper proposes a completely different approach to the slicing problem for functional programs. Instead of extracting program information to build an underlying dependencies structure, we resort to standard program calculation strategies, based on the so-called Bird-Meertens formalism. The slicing criterion is specified either as a projection or a hiding function which, once composed with the original program, leads to the identification of the intended slice. Going through a number of examples, the paper suggests this approach may be an interesting, even if not completely general, alternative to slicing functional programs.