Generalized LR Parsing in Haskell

João Fernandes
jpaulo@di.uminho.pt

Departamento de Informática,University of Minho
Braga, Portugal

Abstract. Parser combinators elegantly and concisely model generalised LL parsers in a purely functional language. They nicely illustrate the concepts of higher-order functions, polymorphic functions and lazy evaluation. Indeed, parser combinators are often presented as a motivating example for functional programming. Generalised LL, however, has an important drawback: it does not handle (direct nor indirect) left recursive context-free grammars.
In a different context, the (non-functional) parsing community has been doing a considerable amount of work on generalised LR parsing. Such parsers handle virtually any context-free grammar. Surprisingly, little work has been done on generalised LR by the functional programming community.
In this presentation, we present a concise and elegant implementation of an incremental generalised LR parser generator and interpreter in Haskell. For good computational complexity, such parsers rely heavily on lazy evaluation. Incremental evaluation is obtained via function memoisation.
An implementation of our generalised LR parser generator is available as the HaGLR tool. We assess the performance of this tool with some benchmark examples.