A Library for Processing Ad hoc Data in Haskell

Embedding a Data Description Language

Yan Wang and Verónica Gaspes

Halmstad University
{yan.wang,veronica.gaspes}@hh.se

Abstract. Ad hoc data formats, i.e. semistructured non-standard data formats, are pervasive in many domains that need software tools — bioinformatics, demographic surveys, geophysics and network software are just a few. Building tools becomes easier if parsing and other standard input-output processing can be automated. Modern approaches for dealing with ad hoc data formats consist of domain specific languages based on type systems. Compilers for these languages generate data structures and parsing functions in a target programming language in which tools and applications are then written. We present a monadic library in Haskell that implements a data description language. Using our library, Haskell programmers have access to data description primitives that can be used for parsing and that can be integrated with other libraries and application programs without the need of yet another compiler.

1 Introduction

Imagine your favorite online travel agency dealing with data from many sources: airlines, train companies, car rental companies, hotels and maybe some more. It has to understand all these formats for which parsers and converters have to be programmed. Moreover, as the agency comes to serve more companies this work has to be done again for new data formats. It is most likely that these companies have legacy data formats that have not been upgraded to XML or other standardized formats for which tools exist. Furthermore, these data formats frequently evolve over time, leave some fields unused or use them for new purposes etc, making some of the data seem erroneous according to the latest version of the format. This scenario is not at all unique. In bioinformatics, demographic applications, geophysics applications, network traffic monitoring, web servers logs, etc, most of the data formats are *ad hoc*, i.e., semistructured non-standard data formats. Typical tools programmed for ad hoc data sources of this kind include generating reports, exporting data to other formats, collecting statistics and reporting errors in input data.

We came across a similar problem when designing a domain specific language for the implementation of protocol stacks. Packet formats are often described in packet specifications. The physical organization, the dependencies among field contents and the constraints over the values of some fields are provided using a combination of figures and explanations. In protocol implementations there are