Using FFI's pragmas in Agda

Andrés Eugenio Castaño Cardenas, Universidad EAFIT

Abstract**—**This paper exhibits the power that Agda can have besides all the capability of programming with dependent types, Agda is a proof assistant, it is an interactive system for writing and checking proofs [1], that so named interactivity is seen from an other point of view in this work, the capability of interact with an Agda program, the option that we used to achieve this goal for writing interactive programs is the foreign function interface a way of calling Haskell functions from Agda [2].

Index Terms**—**Agda,database,FFI,foreign function interface, Haskell, HDBC, interactive, PostgreSQL, proof assistant.

I. Introduction

This document is a brief summary through the using of the foreign function interface (FFI) pragmas applied to a specific example, there are 5 types of pragmas a IMPORT, COMPILED_TYPE, COMPILED_DATA, COMPILED and recently has been added a new one the COMPILED_EPIC this is used by the EPIC backend and to give EPIC code for postulated definitions, though in the present paper we are going to use the four first mentioned pragmas these ones compile via Haskell but the COMPILED_EPIC pragma compiles via C, not Haskell [3].

One of the main motivations of doing a paper like this borns after the reading of The Power Pi and the section of relational database. Databases are everywhere. When you book a flight , order a book, or rent a movie online, all you are really doing under the hood is querying and updating a database. For this reason, a programming language must be able to interface with a database. Most of the time such interface consists of a pair of functions to send a request—as a simple string containing an SQL query—and to receive a response—usually in the form of a string or some dynamic type [4]. So we will use an interface called HDBC to interact with a database through FFI from Agda to Haskell, the FFI pragmas mentioned above will let us interact to the Haskell interface that handle's the database.

In section II we are going to make a short description about what is a FFI, and in the sections below we are going to make some insight with each of the pragmas that were needed to integrate Agda with Haskell in this work, in section VII there is an example in which we applied the pragmas shown.

II. FOREIGN FUNCTION INTERFACE

A foreign function interface is a mechanism that lets a program written in a programming language [13] make a call to some function or use a bunch of services written in other one. Agda has a foreign function interface for calling Haskell functions from Agda. Foreign functions are only executed in compiled programs [2].

III. IMPORT PRAGMA

The first part of writing the program is to identify which modules are needed for the construction of the code, this pragma instructs the compiler to generate a Haskell import statement in the compiled code. For our purpose we import the libraries needed to interact to a Postgres database, the following: Database.HDBC[6] and Database.HDBC.PostgreSQL [7], HDBC provides an abstraction layer between Haskell programs and SQL relational databases. This lets us write database code once, in Haskell, and have it work with any number of backend SQL databases (MySQL, Oracle, PostgreSQL, ODBC-compliant databases, etc.) [5]. There are more options for interacting with a database from Haskell, like Takusen [8] and HaskellDB [9].

IV. COMPILED_TYPE PRAGMA

The COMPILED_TYPE pragma will help us to define the types in the HDBC types library, such as Connection, SqlValue and SqlColDesc, a reminder is that this pragma only works with types that are postulated in Agda. The syntax of this pragma is:

```
{-# COMPILED_TYPE D HsType #-}
```

The COMPILED_TYPE pragma tells the compiler that the postulated Agda type D corresponds to the Haskell type HsType. This information is used when checking the types of COMPILED functions and constructors [2].

V. COMPILED_DATA PRAGMA

According with the syntax:

```
{-# COMPILED_DATA D HsD HsC1...HsCn #-}
```

The COMPILED_DATA pragma tells the compiler that the Agda datatype D corresponds to the Haskell datatype HsD and that its constructors should be compiled to the Haskell constructors HsC1 ... HsCn. The compiler checks that the Haskell constructors have the right types and that all constructors are covered [2].

VI. COMPILED PRAGMA

The COMPILED pragma is the one we use to seize some functions of the module Database.HDBC.PostgreSQL such as the connection, and from Database.HDBC: query, commit, run and other functions.

According to the syntax:

```
{-# COMPILED f HsCode #-}
```

The COMPILED pragma tells the compiler to compile the postulated function f to the Haskell code HsCode. HsCode can be an arbitrary Haskell term of the right type. This is checked by translating the given Agda type of f into a Haskell type and checking that this is the type of HsCode [2].

VII. DATABASE EXAMPLE

The idea behind this implementation is gain all the facilities of HDBC interface calling the right functions for the manipulation of the database [11], first of all for using functions like connectPosgreSQL we need to import the right modules to start working, Database.HDBC and Database.HDBC.PostgreSQL

```
{-# IMPORT Database.HDBC #-}
{-# IMPORT Database.HDBC.PostgreSQL #-}
```

Once we have done this we can start thinking in which are the types that we need to get connectPostgreSQL function up and running, so we have to use another pragma.

```
postulate
  SqlValue : Set
{-# COMPILED_TYPE SqlValue
          Database.HDBC.SqlValue #-}
```

Finally for using the PostgreSQL function connect we need to postulate a function in the Agda side and associate it with the Haskell throw the compiled pragma.

```
{-# COMPILED dbconnect Database.HDBC.
          PostgreSQL.connectPostgreSQL #-}
```

The complete code [12] shows different function calls, the only thing that is left is to compile the Agda program and execute the Haskell result.

VIII. CONCLUSION

In some point Agda gain interactivity with the use of the FFI's applied to access a PostgreSQL database. The foreign function interface allows the use of a bunch of functions and it facility encourage it use, one of the things is that the implementation of database interaction in Agda was simple approach and it has numerous drawbacks, cause the interface is unsafe in the meaning that there are not static checks on the queries and it is all too easy to formulate a syntactically incorrect or semantically incoherent query; an unexpected response from the database server results in a runtime error. There is some work that address this issues in an implementation level [10] that is safe and totally embedded. Other works sketch how to write a domain-specific embedded language for relational algebra in Agda [4]. One of issues with the implementation of the database example is that in some points some of the pragmas couldn't disappear the level parameters, where they suppose to.

REFERENCES

[1] *Agda Intro.* May 2011.http://wiki.portal.chalmers.se/Agda/pmwiki.php

[2] *Foreign Function Interface.* November 2009. http://wiki.portal.chalme .se/Agda/pmwiki.php?n=ReferenceManual.ForeignFunctionInterface# ToHaskellType

[3] *Release notes for Agda 2 version 2.2.10.* February 2011. http://wiki. portal.chalmers.se/Agda/pmwiki.php?n=Main.Version-2-2-10

[4] Nicolas Oury and Wouter Swierstra, *The Power of pi.* 2008 http://www.cs.nott.ac.uk/~wss/Publications/ThePowerOfPi.pdf

[5] John Goerzen, *Welcome to HDBC, Haskell Database Connectivity.* September 2010. https://github.com/jgoerzen/hdbc/wiki

[6] hackageDB, *The HDBC package.* April 2011. http://hackage.Haskell.o rg/package/HDBC.

[7] John Goerzen, PostgreSQL driver for HDBC. February 2011. https://github.com/jgoerzen/hdbc-PostgreSQL

[8] *Takusen Making DBMS Access Efficient and Safe.* July 2010. http://projects.Haskell.org/takusen/

[9] *HaskellDB.* September 2010. http://www.Haskell.org/Haskellwiki/ Applications_and_libraries/Database_interfaces/HaskellDB

[10] Ulf Norell, *Database.* September 2009. http://www.cse.chalmers.se /~ulfn/code/database/Main.html

[11] Bryan O'Sullivan, Don Stewart, and John Goerzen, *Real World Haskell.* http://book.realworldHaskell.org/read/using-databases.html

[12] Andrés Castaño. *Using FFI's pragmas in Agda.* June 2011. http://www1.eafit.edu.co/asicard/teaching/dtfl-CB0683/projects/andres -eugenio-castano-cardenas/src.zip

[13] Wikipedia, *Foreign function interface.* May 2011. http://en.wikipedia.org/wiki/Foreign_function_interface