


Using FFI's pragmas in Agda

Andrés Eugenio Castaño Cardenas
acastan7@eafit.edu.co

Universidad EAFIT
June 20, 2011

Overview

- Motivation
- FFI's pragmas
- Database example

MOTIVATION

- "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."

The Power of Pi by Nicolas Oury and Wouter Swierstra page 8

MOTIVATION

- "Critical Systems are interactive. We need to be able to prove the correctness of interactive programs."
- "Programming with Dependent Types only convincing, if we can write interactive programs."

Interactive programs in dependent type theory by Anton Setzer page 2

The Main Idea

- Make a connection to a PostgreSQL database and do a query to a table using Agda through the FFI's pragmas and HDBC from Haskell.

FFI's pragmas

- Agda has a foreign function interface for calling Haskell functions from Agda. Foreign functions are only executed in compiled programs.

IMPORT pragma

- The IMPORT pragma instructs the compiler to generate a Haskell import statement in the compiled code. The pragma above will generate the following Haskell code:

```
import qualified HsModule
```

IMPORT pragmas can appear anywhere in a file.

Syntax: `{-# IMPORT HsModule #-}`

COMPILED_TYPE pragma

- 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.

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

COMPILED_DATA pragma

- 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.

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

COMPILED pragma

- 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.

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

Database example

- In accordance with the order of each mention of the pragmas I will show an example that is immerse in the code of DB_project_p.agda

Database example

- First of all we will use the IMPORT pragma, that will tell Agda which modules of Haskell we need to have for our propose.

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

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

Database example

- 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.

Database example

```
postulate

  SqlValue    : Set
  SqlColDesc : Set

{-# COMPILED_TYPE SqlColDesc
Database.HDBC.SqlColDesc #-}

{-# COMPILED_TYPE SqlValue
Database.HDBC.SqlValue #-}
```

Database example

- The COMPILED_DATA pragma we will use it to associate a datatype in Agda and its constructors to the ones in Haskell's. Such is the case of dbList that has the same definition that the datatype List in the standard library.

Database example

```
data dbList (a : Set) : Set where
  []  : dbList a
  _::_ : a → dbList a → dbList a
```

```
{-# COMPILED_DATA dbList [] [] (:) #-}
```

Database example

- 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 the querying, the commit, run and some others.

Database example

```
postulate
  dbconnect : String → IO Connection
  dbquery : Connection → Query → dbList Value → IO
    (dbList Row)
  dbcommit : Connection → IO Unit
{-# COMPILED dbquery Database.HDBC.quickQuery' #-}
{-# COMPILED dbotrans Database.HDBC.run #-}
{-# COMPILED dbcommit Database.HDBC.commit #-}
```

Conclusions

- 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.
- This work is 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.