Haskell compiler for Android written in Haskell (pre-release)

(In English / In Japanese[*1])

written : 2020-12-15 by @unnohideyuki[*2]

Introduction

This is an article for the 15th day of the Haskell Advent Calendar 2020[*3].

I'm going release, a prototype pre-release however, a Haskell compiler which I introduced in the article Let us write a Haskell compiler![*4] As I mentioned in it, I'm planning to release it as Version 1 when it meets the Haskell 2010 specification and supports FFI, but since it hasn't reached that point yet, I'm releasing it as version 0.9.0.

Bunny - A Haskell compiler for Android

Bunny is a Haskell compiler targeting at building applications that run on Android. Because I learned the basics of how to write a compiler in the book "Modern Compiler Implementation in ML", I named it Bunny after the nick name of the book, "Tiger Book".

Bunny is a Haskell compiler written in Haskell, which reads Haskell programs as source code and emits Java code as object code. By building it with the runtime library (also included in the Bunny project), you can create an Android application. The process of compiling a Haskell program into an object Java program is described in the article I wrote 4 years ago. The general structure has not changed since then.

Bunny runs on a Windows or Linux machine with GHC (Haskell Platform) and Android Studio installed, see the Bunny project page for information on how to get Bunny and build it.

[*1] `https://uhideyuki.sakura.ne.jp/studs/index.cgi/ja/BunnyHaskellCompiler009`
[*2] `https://twitter.com/unnohideyuki`
[*3] `https://qiita.com/advent-calendar/2020/haskell`
[*4] I'm sorry, this is in Japanese only.

Sample Usage

Although Bunny version 0.9.0 still has many limitations (see the release note for the limitations)[*5], you can compile a simple program to create an Android application.

Using the problem C[*6] from the AtCoder Beginner Contest 185 as an example. To solve the problem, it can be written as follows:

```
getInteger :: IO Integer
getInteger = do
  s <- getLine
  return $ read s

main = do
  l <- getInteger
  let ans = product [(l-11)..(l-1)] `div` product [1..11]
  print ans
```

Save this program as abc185c.hs in an appropriate working directory, and compile it using Bunny to create an Android project.

```
bunny android abc185.hs
```

When successfully compiled, you can see the message like the following:

```
------------------------------------------------------------
An android project has been created!
  path: $HOME/BunnyProjects/abc185c

then you can try:
$ cd $HOME/BunnyProjects/abc185c
$ ./gradlew assembleDebug
$ adb install app/build/outputs/apk/debug/app-debug.apk
------------------------------------------------------------
```

[*5] There may also be a lot of bugs that are not mentioned in the note.

[*6] `https://atcoder.jp/contests/abc185/tasks/abc185_c`

Following the recommendation in this message, move to the directory of the Android project and run `./gradlew` (or `./gradlew.bat` on Windows) to build an Android app. For this time, create a debug APK with the assemblyDebug task.

```
cd $HOME/BunnyProjects/abc185c
./gradlew assembleDebug
```

Next, let's run the apk on an Android virtual machine. To do this, we'll use `adb` command included in the Android SDK, but before that, we need to start a virtual machine. Start Android Studio, select "Open AVD Manager" from the menu to launch AVD Manager, and start up a virtual machine of your choice.

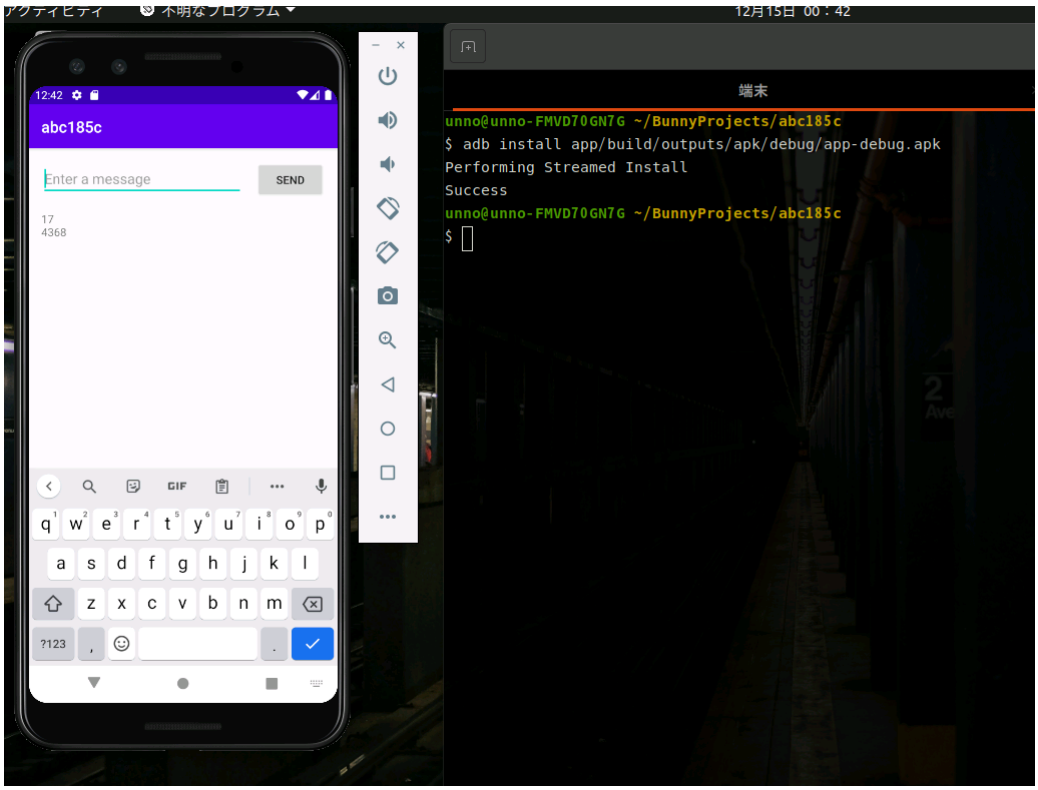
After launching the virtual machine, use the adb command to install the apk file.

```
adb install app/build/outputs/apk/debug/app-debug.apk
```

After that, operate the virtual machine to launch the application named abc185c. This program reads one line from the standard input and displays the calculation results accordingly, so nothing is displayed at first.

For example, enter 17, the same as in the Sample Input 3 of ABC185C[*7], and press the SEND button. If you see two lines 17 and 4368 in the display area, you have succeeded.

[*7] `https://atcoder.jp/contests/abc185/tasks/abc185_c`



Towards Version 1

If I could, I would have said today, "I will release Version 1!", but there were so many items remaining that I couldn't finish in time. However, I think it gets much closer to the completion of the tool by trying to release it even as a pre-release.

For Version 1, the goals are to meet the Haskell 2010 specification and to implement FFI that enables to use Java libraries. I'm hoping to release Version 1 in 6 months time, around May 2021 (see also our roadmap).

I'll keep working on it!

Wishing you a happy holiday season and a wonderful good New Year!