# Diary to release a Haskell Compiler for Android in 30 days

(In English / In Japanese[*1])

Four years have passed since I wrote the article, Let us wite a Haskell compiler in Haskell[*2], and eventually I'd like to finish it soon.

I was hoping to release the first edition on the same day when I'll post this year's article for the Haskell advent calendar, but it's not going to be possible. There are still a lot of unimplemented parts of Haskell 2010's specification, and there are a lot of bugs that have not been solved, and also a lot of lax descriptions. So I'd like to release version 0.9 as a base camp for the first version.

There are a lot of bugs, unknown parts, lack of features, etc., but somehow, I will organize them in some way[*3] and release version 0.9!

By the way, I'm not sure if it's really in 30 days, but I think I'm getting an "off-by-one" error. Does 1-origin save me?

## 2020-11-26 (Thu)

Day 11

### Run the main routine in a separate thread from the UI thread

As the UI will stop when the main routine runs on MainActivity, the UI thread, I will modify to make it run on a separate thread. To do so, I will add an IntentService and place the main routine in it.

```
public class MainIntentService extends IntentService {
    private static Deque<String> dequeIn = new ArrayDeque<>();
    public static TextView textView;

    public MainIntentService() {
        super("MainIntentService");
    }

    /**
     * Starts this service to perform action Main.
     * If the service is already performing a task this action will be queued.
     *
     * @see IntentService
     */
```

[*1] https://uhideyuki.sakura.ne.jp/studs/index.cgi/ja/thirtydays2release09
[*2] It is only in Japanese, Sorry.
[*3] some way!, lol

```
public static void startMain(Context context) {
    Intent intent = new Intent(context, MainIntentService.class);
    context.startService(intent);
}

public static void setTextview(TextView tv){
    textView = tv;
}

@Override
protected void onHandleIntent(Intent intent) {
    handleActionMain();
}

/**
 * Handle action Main in the provided background thread.
 */
private void handleActionMain() {
    while (true){
        String s = getLine();
        String[] t = s.split(" +");
        double sx = Double.parseDouble(t[0]);
        double sy = Double.parseDouble(t[1]);
        double gx = Double.parseDouble(t[2]);
        double gy = Double.parseDouble(t[3]);
        double ans = (sx*gy + gx*sy) / (sy + gy);
        String r = Double.toString(ans);
        putStrLn(r);
    }
}

public static void putline(String s){
    dequeIn.addLast(s);
}

private String getLine(){
    while (dequeIn.size() == 0){ /* wait */ }
    String r = dequeIn.removeFirst();
    return r;
}
```

```
    private void putStrLn(String str){
        textView.append(str + "\n");
    }
}
```

In this example, the so-called main routine is directly in handleActionMain. There is an infinite loop with `while (true)`, but this time it runs on a different thread than the UI thread, so it won't stop the UI.

The following is the MainActivity changed accordingly. The `onStart` function is used to start the Main service. We have a few additional functions for exchanging strings emulating standard inputs/outputs.

```
public class MainActivity extends AppCompatActivity {
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);
    }

    @Override
    protected void onStart() {
        super.onStart();
        TextView textView = findViewById(R.id.textView);
        MainIntentService.setTextview(textView);
        MainIntentService.startMain(this);
    }

    /** Called when the user taps the Send button */
    public void sendMessage(View view) {
        EditText editText = (EditText) findViewById(R.id.editText);
        String message = editText.getText().toString();
        editText.setText("");

        TextView textView = findViewById(R.id.textView);
        textView.append(message + "\n");

        MainIntentService.putline(message);
    }
}
```

It behaves the same way as yesterday's prototype. This is a little bad manner, since the interaction between Service and MainActivity is direct rather than intent, but with mutext locks added, it can be OK?.

This time, I put the sample program directly in the Service's handleActionMain function, and we can modify it call the program compiled from Haskell.

From tomorrow onwards, I will first try to integrate the program compiled from Haskell to this prototype manually. After that, I'll automate the process.

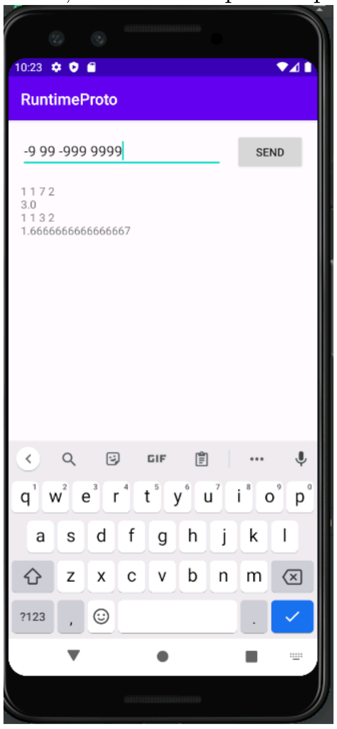
## 2020-11-25 (Wed)

Day 10

### Runtime UI Prototype on Android (continued)

I'll modify MyFirstApp written yesterday to create a runtime UI prototype. It has editText and buttons for input and TextView for display, that simply emulates stdin /stdout.

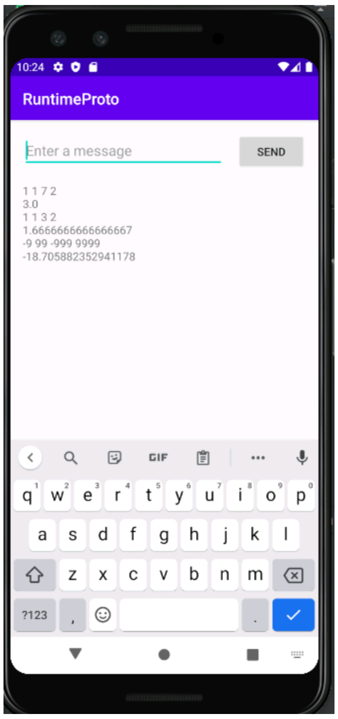
Then, attach a simple sample program that solves ABC183B[*4], and let it run.



```
-9 99 -999 9999
```

```
1 1 7 2
3.0
1 1 3 2
1.6666666666666667
```

→



```
1 1 7 2
3.0
1 1 3 2
1.6666666666666667
-9 99 -999 9999
-18.705882352941178
```

The main contents of MainActivity.java are as follows:

```java
public class MainActivity extends AppCompatActivity {
```

[*4] `https://atcoder.jp/contests/abc183/tasks/abc183_b`

```
@Override
protected void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.activity_main);
    // mainProgram();
}

public static Deque<String> dequeIn = new ArrayDeque<>();

/** Called when the user taps the Send button */
public void sendMessage(View view) {
    EditText editText = (EditText) findViewById(R.id.editText);
    String message = editText.getText().toString();
    editText.setText("");

    dequeIn.addLast(message);

    TextView textView = findViewById(R.id.textView);
    textView.append(message + "\n");

    mainProgram();
}

public void putStr(String s) {
    TextView textView = findViewById(R.id.textView);
    textView.append(s);
}

public void putStrLn(String s) {
    putStr(s + "\n");
}

public String getLine() {
    while (dequeIn.size() == 0) {
        // wait
    }
    String r = dequeIn.removeFirst();
    return r;
}
```

```
    public void mainProgram(){
        String s = getLine();
        String[] t = s.split(" +");
        double sx = Double.parseDouble(t[0]);
        double sy = Double.parseDouble(t[1]);
        double gx = Double.parseDouble(t[2]);
        double gy = Double.parseDouble(t[3]);
        double ans = (sx*gy + gx*sy) / (sy + gy);
        String r = Double.toString(ans);
        putStrLn(r);
    }
}
```

First, the mainProgram does an infinite loop with a while(true) and called at the end of onCreate, but it didn't finish onCreate and the app didn't work.

I think it should be called from the appropriate handler instead of onCreate, and it should do something like a thread switching when waiting for input, instead of just an endless loop. I should learn how to write such kind of work for Android.

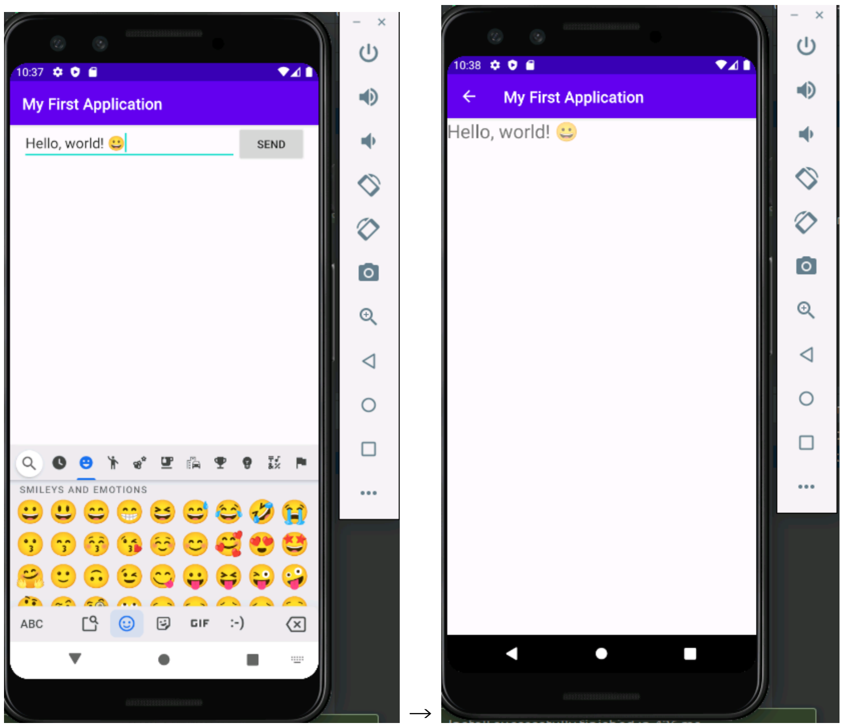
## 2020-11-24 (Tue)

Day 9

### Making an Android App

I wrote an android app today, referring to the Android Developer Guide*[5]. "Build Your First App" has been finished.

*5 https://developer.android.com/training/basics/firstapp



From now on, I will modify this step by step to create a simple UI that connects to the runtime library of the compiler I'm creating.

## 2020-11-23 (Mon)

Day 8. No progress.

## 2020-11-22 (Sun)

Day 7. No progress.

## 2020-11-21 (Sat)

Day 6

### Corrections around the synonym

The compiler can handle simple synonyms like `String`, but it fails with the synonyms that take type variables like `ReadS a`. I am going to fix it this time.

Yesterday, I almost finished modifications of `Rename.hs` and `RenUnil.hs` for this. I originally tried to solve the synonym in renSigdoc, as the original code did, but that didn't work as the fix affected other parts of the code and led to other problems.

So, I made synonyms resolved at the beginning of `A.SynonymDecl` processing:

```
renDecl (A.TypeSigDecl ns (maybe_sigvar, sigdoc')) = do
  ns' <- mapM renameVar ns
  sigdoc <- actualType sigdoc' -- Here!
  let kdict = kiExpr sigdoc []
  ps <- renSigvar maybe_sigvar kdict
  t <- renSigdoc sigdoc kdict
  return [(n, Just (ps :=> t), []) | n <- ns']
```

It works for now (for the input program I'm experimenting with), but the compiler fails kind inference in the `kiExpr` (for now, I just give a warning and ignore this failure). I wrote this kiExpr in the very early days, so I think I'll have to rewrite it properly.

Another part that needs to resolve synonyms is in the renaming of data declarations. I've taken care of that as well, however, the case of complex synonyms in data declarations still doesn't work.

Added test cases and modified lib/Prelude.hs to reflect the above situation.

- lib/Prelude.hs : some parts are modified using `ReadS a`
- testcases/typesyn.hs: An example using the complex type synonym, OK.
- testcases/typesyn2.hs: typesyn.hs with redundant parentheses. Errors.
- testcases/typesyn2b.hs: Removes redundant parentheses from typesyn2b.hs on the type declaration side. It's Ok
- testcases/typesyn3.hs and typesyn3b.hs: Samples have complex synonyms in the data declaration. Errors in both cases.

Among the above testcases/typesyn*.hs, add the ones that were ok to test/samples (used in make check).

## 2020-11-20 (Fri)

Day 5

I did some work around the synonym. Tomorrow, I will do some additional work and write here about it.

## 2020-11-19 (Thu)

Day 4

I only made really small progress today. Instead, my wife and I were on a sightseeing trip to Jindaiji.

### Just a few steps forward

Preperation for the modification to deal with more complex synonyms, I first changed the synonym dictionary from `[(A.Type, A.Type)]` to `[(A.Type, ([A.Type], A.Type))]`.

We also have to deal with cases with redundant parentheses, so I prepared such test cases:

```
type ((Foo a) (b)) = String -> (a, b)

foo :: (Foo (Int)) (Char)
foo s = (length s, head s)

main = print $ foo "Hello!"
```

## 2020-11-18 (Wed)

Day 3

### Investigate the problem of type synonym

I was thinking of starting some work to get it running on Android next, but since I only had a little time to spare today, I decided to work on a smaller task.

I would like to determine the cause of the problem with the type synonym and how to fix it. If the left side does not contain any type variables, such as `type String = [Char]`, it still works fine. A small test program that contains an error case:

```
type Foo a b = String -> (a, b)

foo :: Foo Int Char
foo s = (length s, head s)

main = print $ foo "Hello!"
```

This will be an error when compiled:

```
# 1. tcompile
source file: testcases/typesyn.hs
dst dir: /typesyn
doCompile ... done.
implicitPrelude ... done.
doCompile ... bunnyc: renSigDoc $ A.Tycon Foo
CallStack (from HasCallStack):
  error, called at src/Rename.hs:711:33 in main:Rename
```

Parts to be fixed:

- The part that reads TypeSynonymDecl and creates a synonym dictionary
- The part that replaces the synonym with the actual type using the synonym dictionary (actual_ty)

actual_ty converts from A.Type to A.Type, i.e., Absyn stage before converting to Typing.Type. This will be followed after the modification.

The first part is like this:

```
    scandecl (A.SynonymDecl t1 t2) = do
      st <- get
      let syn = rnSyn st
      put st{rnSyn=((t1, t2):syn)}
      return ([], [], [])
```

For synonym declarations like type `String = [Char]`, an entry like `(String, [Char])` will be made, and the replacement will happen later as expected.

But for type `Foo a b = String -> (a, b)`, an entry like `(Foo a b, String -> (a, b))` will be created, which doesn't work.

There are two places where this dictionary will be used, one of which is in the following part:

```
renSigdoc (A.AppTy e1 e2) kdict = do
  t1 <- renSigdoc e1 kdict
  t2 <- renSigdoc e2 kdict
  return (TAp t1 t2)
```

```
renSigdoc t@(A.Tycon n) kdict = do
  issyn <- isSynonym t
  if issyn
    then do t' <- actualTy t
            renSigdoc t' kdict
    else do let n' = origName n
            t <- lookupTConst n'
            return $ fromMaybe (error $ "renSigDoc $ A.Tycon " ++ n') t
```

For a constructor, the synonym dictionary is used in `isSynonym`. However, `Foo` will not hit in the dictionary because there is only `Foo a b` in the dictionary.

So we should make an entry like `(Foo, Synonym [a, b] (String -> (a, b)))` instead of `(Foo a b, String -> (a, b)))` when registering the dictionary. And the user of the dictionary needs to check whether the type constructor is a synonym at `A.AppTy (A.AppTy c v1) v2)` and replace it if it is, instead of decomposing it to A.Tycon and checking if it is a synonym[*6].

Note that I only mentioned renSigdoc here, but there is one more part that uses the synonym dictionary (search isSynonym in Rename.hs to find), where the same thing needs to be done.

That's all I have to do today (time is up). I'll actually fix it tomorrow or later.

## 2020-11-17 (Tue)

Day 2

### Make Int and Integer instances of the Read class

Yesterday, when I set up the target sample program, it got stuck where Int was not an instance of Read, so I'll fix that today. It will be done by adding instance declarations and support functions for them in lib/Prelude.hs.

I wrote the code in lib/Prelude.hs with reference to the code in Haskell98[*7], but then I got a run-time error saying that the pattern match was not exhaustive.

Apparently, pattern matching in list comprehension expressions, as seen in the definition of readSigned, is not treated well by the compiler.

```
readSigned :: (Real a) => ReadS a -> ReadS a
readSigned readPos = readParen False read'
                     where read' r  = read'' r ++
                                      [(-x,t) | ("-",s) <- lex r,
                                                (x,t)   <- read'' s]
```

[*6] Since there are simple cases like String, there remains the process of replacing A.Tycon alone, as it is now

[*7] because Haskell 2010 Language Report doesn't show the code that I want today.

```
                              read'' r = [(n,s)  | (str,s) <- lex r,
                                                   (n,"")  <- readPos str]
```

For example, `("-", s) <- lex r` will fail if `lex r` returns a result that does not match `("-", s)`. This should not be a run-time error, but the current compiler cannot handle it well.

As a temporary workaround, I removed such kind of description from it (there is the same problem with readParen and I removed it as well):

```
readSigned :: (Real a) => (String -> [(a, String)]) -> (String -> [(a, String)])
readSigned readPos = {- readParen False -}  read'
  where read' r | head r == '-' = [(-n, s) | (n, s) <- readPos (tail r)]
                | otherwise     = readPos r
```

Anyway, we are now close to being able to run the target program. In today's version, we can run the following program:

```
getIntList :: IO [Int]
getIntList = do s <- getLine
                return $ (map read . words) s

main = do
  [sx, sy, gx, gy] <- getIntList
  print (fromIntegral (sx*gy + gx*sy) / fromIntegral (sy + gy) :: Double)
```

The results of the execution are as follows: for each of the ABC183B[*8] input examples, the results appear to be as expected:

```
$ ./trun testcases/abc183b_t.hs
# 1. tcompile
source file: testcases/abc183b_t.hs
dst dir: /abc183b_t
doCompile ... done.
implicitPrelude ... done.
doCompile ... done.
# 2. jout/compile
```

[*8] https://atcoder.jp/contests/abc183/tasks/abc183_b

```
#!/bin/bash -v

target=$1
d=‘dirname $1‘

s=":"
if [ -d /c ]; then
    s=";"
fi

javac -J-Duser.language=en -cp "../../brt/src$s$d" $target
# 3. jout/run
1 1 7 2
3.0
$ cd jout/
$ ./run abc183b_t/Sample.java
1 1 3 2
1.6666666666666667
$ ./run abc183b_t/Sample.java
-9 99 -999 9999
-18.705882352941178
```

Today’s progress:

- Albeit they are provisional version, Int and Integer are now Read instances

Glitches identified today:

Not that I’m making anything new up, but I’ll make a note of some things I noticed/remember from today’s work:

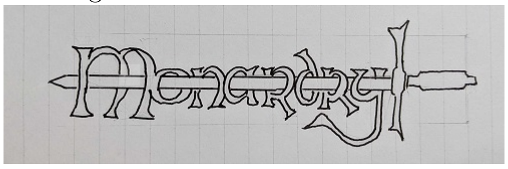
- Type synonym `type ReadS a = String -> [(a, String)]` doesn’t work (so I wrote it without the synonym today)
- Everything in lib/Prelude.hs is exported, including the support functions (namespace contamination).
- In the list comprehension expression, the behavior of pattern-match assignment is wrong when it fails (see above, new).

Next up is some work to get it running on Android (with a few additions to the runtime to get it running on Android, and some build instructions (scripts etc.) for Android)

### digression

For the next version 0.9, I've set up a pretty small sample program as the goal. I think that's fine for this time, but for version 1.0, I'd like to make a small demo application.

It will be a small game or something. And in fact, I've already decided on a title and logo of it. Something like this



Oh no, I wonder what kind of game it is.

## 2020-11-16 (Mon)

Day 1

### The target program

I want to set a specific goal like to say "the compiler will work for this program, for example." Let's make this the one to solve the last B problem on the ABC[*9]. Here's the one:

```
getIntList :: IO [Int]
getIntList = do s <- getLine
                return $ (map read . words) s

main = do
  [sx, sy, gx, gy] <- getIntList
  print $ fromIntegral (sx*gy + gx*sy) / fromIntegral (sy + gy)
```

As of today, the compiler doesn't succeed to compile it. Maybe it's because of the following:

- No read function yet (Int is not yet an instance of Read)
- Defaulting from Fractional to Double is not yet implemented

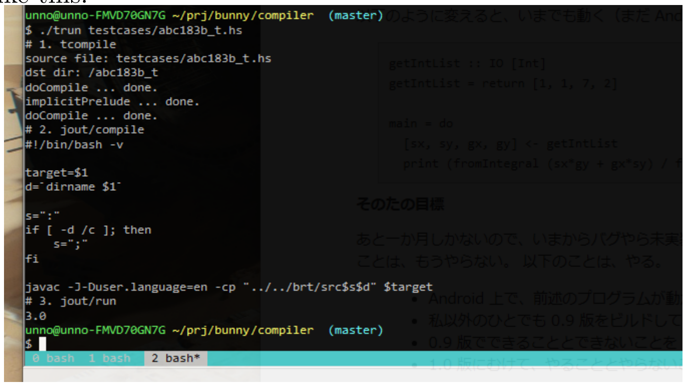
When we change the program as follows, it works even today (not on Android yet, but on Windows / Linux where the compiler works):

```
getIntList :: IO [Int]
getIntList = return [1, 1, 7, 2]
```

[*9] https://atcoder.jp/contests/abc183/tasks/abc183_b

```haskell
main = do
  [sx, sy, gx, gy] <- getIntList
  print (fromIntegral (sx*gy + gx*sy) / fromIntegral (sy + gy) :: Double)
```

Like this:



```
unno@unno-FMVD70GN7G ~/prj/bunny/compiler  (master)
$ ./trun testcases/abc183b_t.hs
# 1. tcompile
source file: testcases/abc183b_t.hs
dst dir: /abc183b_t
doCompile ... done.
implicitPrelude ... done.
doCompile ... done.
# 2. jout/compile
#!/bin/bash -v

target=$1
d=`dirname $1`

s=":"
if [ -d /c ]; then
    s=";"
fi

javac -J-Duser.language=en -cp "../../brt/src$s$d" $target
# 3. jout/run
3.0
unno@unno-FMVD70GN7G ~/prj/bunny/compiler  (master)
$
 0 bash  1 bash  2 bash*
```

## other goals

There is only one month left to go, and I shouldn't try to fix a lot of bugs and to add lacking features etc., that makes it difficult to release version 0.9. So, I'm not going to do that anymore. So, I'll do the following.

- Make the program above work on Android (I will add missing features and fix bugs only within this range).
- Write a kind of instruction document to build and use version 0.9.
- Identify (as much as possible) what you can and cannot do in version 0.9
- Identify the do's and don'ts for the 1.0 version

These outcomes will be published as an article for the advent calendar on December 15. And then, I hope to release later version 1 in May[*10]

[*10] I expect to need about a half year, and May is the month where my birthday is!