Copilot : Traceability and Verification of a Low Level Automatically Generated C Source Code

Georges-Axel Jaloyan

École Normale Suprieure, NASA Langley Research center, National Institute of Aerospace

August 24, 2015

Copilot language

Copilot is an *EDSL* (embedded domain specific language), embedded in *Haskell* and used for writing *runtime monitors* for hard real-time, distributed, reactive systems written in C.

A Copilot program, can either be :

- compiled to C using two back-ends : SBV, ATOM
- interpreted
- analyzed using static analysis tools (CBMC, Kind)

Copilot syntax

A program is a list of streams that can be either external or internal which are defined by mutually recursive stream equations.

Each stream has a type which can be `Bool`, `Int8`, `Int16`, `Int32`, `Int64`, `Word8`, `Word16`, `Word32`, `Word64`, `Float`, `Double`.

```
x :: Stream Word16
x = 0
-- x = {0, 0, 0, ...}
y :: Stream Bool
y = x `mod` 2 == 0
-- y = {T, T, ...}
nats :: Stream Word64
nats = [0] ++ (1 + nats)
-- nats = {0,1,2, ..., 2^64-1, 0, 1, ..}
```

Operators

Each operator and constant has been lifted to Streams (working pointwise).

Two temporal operations working on Streams :

- ++ : which prepends a finite list to a Stream

  ```
  (++) :: [a] -> Stream a -> Stream a
  ```

- drop : which drops a finite number of elements at the beginning of a Stream

  ```
  drop :: Int -> Stream a -> Stream a
  ```

Casts and unsafe casts are also provided :

```
cast :: (Typed a, Typed b) => Stream a -> Stream b
unsafeCast :: (Typed a, Typed b) => Stream a -> Stream b
```

Examples

Fibonacci sequence :

```
fib :: Stream Word64
fib = [1,1] ++ (fib + drop 1 fib)
-- fib = {1,1,2,3,5,8,13,...,
--        7540113804746346429,-6246583658587674878,...}
```

Interaction

Questions

Questions ?