CSI: Haskell

Fault-Localization in Lazy Languages using Runtime Tracing

MATTHÍAS PÁLL GISSURARSON*, Chalmers University of Technology, Sweden

CHALMERS

WASP | WALLENBERG AI, AUTONOMOUS SYSTEMS AND SOFTWARE PROGRAM

Problem:

```haskell
module Main where
divs :: Int -> [Int]
divs n = go 2
  where go i | i == n = []
        go i = if d i
               then i:(go (i+1))
               else go (i+1)
        d i = n `mod` i == 0

smallestDiv n = head (divs n)

main :: IO ()
main = print (smallestDiv 13)
```

Problem:

```haskell
module Main where
divs :: Int -> [Int]
divs n = go 2
  where go i | i == n = []
        go i = if d i
               then i:(go (i+1))
               else go (i+1)
        d i = n `mod` i == 0

smallestDiv n = head (divs n)

main :: IO ()
main = print (smallestDiv 13)
```

Problem:

Error

```
divs: Prelude.head: empty list
```

```haskell
module Main where
divs :: Int -> [Int]
divs n = go 2
  where go i | i == n = []
        go i = if d i
               then i:(go (i+1))
               else go (i+1)
        d i = n `mod` i == 0

smallestDiv n = head (divs n)

main :: IO ()
main = print (smallestDiv 13)
```

Problem:

Error

```
divs: Prelude.head: empty list
```

```haskell
module Main where
divs :: Int -> [Int]
divs n = go 2
  where go i | i == n = []
        go i = if d i
               then i:(go (i+1))
               else go (i+1)
        d i = n `mod` i == 0

smallestDiv n = head (divs n)

main :: IO ()
main = print (smallestDiv 13)
```

Problem:

Error

```
divs: Prelude.head: empty list
```

```haskell
module Main where
divs :: Int -> [Int]
divs n = go 2
  where go i | i == n = []
        go i = if d i
               then i:(go (i+1))
               else go (i+1)
        d i = n `mod` i == 0

smallestDiv n = head (divs n)

main :: IO ()
main = print (smallestDiv 13)
```

Problem:

Error

Stack trace

```
divs: Prelude.head: empty list
CallStack (from HasCallStack):
  error, called at libraries/…/List.hs:1643:3 in base:GHC.List
  errorEmptyList, called at libraries/…/List.hs:82:11 in base:GHC.List
  badHead, called at libraries/…/List.hs:78:28 in base:GHC.List
  head, called at Div.hs:10:17 in main:Main
```

```haskell
1  module Main where
2  divs :: Int -> [Int]
3  divs n = go 2
4    where go i | i == n = []
           go i = if d i
                  then i:(go (i+1))
                  else go (i+1)
           d i = n `mod` i == 0

10 smallestDiv n = head (divs n)
11
12 main :: IO ()
13 main = print (smallestDiv 13)
```

Problem:

Error

```
divs: Prelude.head: empty list
CallStack (from HasCallStack):
  error, called at libraries/…/List.hs:1643:3 in base:GHC.List
  errorEmptyList, called at libraries/…/List.hs:82:11 in base:GHC.List
  badHead, called at libraries/…/List.hs:78:28 in base:GHC.List
  head, called at Div.hs:10:17 in main:Main
```

Stack trace

```haskell
module Main where
divs :: Int -> [Int]
divs n = go 2
  where go i | i == n = []
        go i = if d i
               then i:(go (i+1))
               else go (i+1)
        d i = n `mod` i == 0

smallestDiv n = head (divs n)

main :: IO ()
main = print (smallestDiv 13)
```

Problem:

Error

Stack trace

```
divs: Prelude.head: empty list
CallStack (from HasCallStack):
  error, called at libraries/…/List.hs:1643:3 in base:GHC.List
  errorEmptyList, called at libraries/…/List.hs:82:11 in base:GHC.List
  badHead, called at libraries/…/List.hs:78:28 in base:GHC.List
  head, called at Div.hs:10:17 in main:Main
```

```haskell
module Main where
divs :: Int -> [Int]
divs n = go 2
  where go i | i == n = []
        go i = if d i
               then i:(go (i+1))
               else go (i+1)
        d i = n `mod` i == 0

smallestDiv n = head (divs n)

main :: IO ()
main = print (smallestDiv 13)
```

Problem:

Error

Stack trace

```
divs: Prelude.head: empty list
CallStack (from HasCallStack):
  error, called at libraries/…/List.hs:1643:3 in base:GHC.List
  errorEmptyList, called at libraries/…/List.hs:82:11 in base:GHC.List
  badHead, called at libraries/…/List.hs:78:28 in base:GHC.List
  head, called at Div.hs:10:17 in main:Main
```

```haskell
module Main where
divs :: Int -> [Int]
divs n = go 2
  where go i | i == n = []
        go i = if d i
               then i:(go (i+1))
               else go (i+1)
        d i = n `mod` i == 0

smallestDiv n = head (divs n)

main :: IO ()
main = print (smallestDiv 13)
```

Problem:

Error

Stack trace

```
divs: Prelude.head: empty list
CallStack (from HasCallStack):
  error, called at libraries/…/List.hs:1643:3 in base:GHC.List
  errorEmptyList, called at libraries/…/List.hs:82:11 in base:GHC.List
  badHead, called at libraries/…/List.hs:78:28 in base:GHC.List
  head, called at Div.hs:10:17 in main:Main
```

The error mentions only the

CONSUMER

but the fault originates in the

PRODUCER

```
1  module Main where
2  divs :: Int -> [Int]
3  divs n = go 2
4    where go i | i == n = []
           go i = if d i
                  then i:(go (i+1))
                  else go (i+1)
         d i = n `mod` i == 0

10 smallestDiv n = head (divs n)
11
12 main :: IO ()
13 main = print (smallestDiv 13)
```

Insight:

Insight:

- In lazy languages, evaluation is ***demand*** driven

Insight:

- In lazy languages, evaluation is ***demand*** driven
- So ***producers*** and ***consumers*** are close!

Insight:

- In lazy languages, evaluation is ***demand*** driven
- So ***producers*** and ***consumers*** are close!
- By *tracing recent expressions*, we can localize faults!

Background: Haskell Program Coverage (HPC) [1]

[1] Andy Gill and Colin Runciman. 2007. Haskell Program Coverage. In *Proceedings of the ACM SIGPLAN Workshop on Haskell Workshop (Freiburg, Germany) (Haskell '07)*. Association for Computing Machinery, New York, NY, USA, 1–12.

Background: Haskell Program Coverage (HPC) [1]

- GHC feature that adds *ticks* to every expression

[1] Andy Gill and Colin Runciman. 2007. Haskell Program Coverage. In *Proceedings of the ACM SIGPLAN Workshop on Haskell Workshop (Freiburg, Germany) (Haskell '07)*. Association for Computing Machinery, New York, NY, USA, 1–12.

Background: Haskell Program Coverage (HPC) [1]

- GHC feature that adds *ticks* to every expression
- RTS keeps track of how often each tick occurred

[1] Andy Gill and Colin Runciman. 2007. Haskell Program Coverage. In *Proceedings of the ACM SIGPLAN Workshop on Haskell Workshop (Freiburg, Germany) (Haskell '07)*. Association for Computing Machinery, New York, NY, USA, 1–12.

Background: Haskell Program Coverage (HPC) [1]

- GHC feature that adds *ticks* to every expression
- RTS keeps track of how often each tick occurred
- Used for program coverage

[1] Andy Gill and Colin Runciman. 2007. Haskell Program Coverage. In *Proceedings of the ACM SIGPLAN Workshop on Haskell Workshop (Freiburg, Germany) (Haskell '07)*. Association for Computing Machinery, New York, NY, USA, 1–12.

Background: Haskell Program Coverage (HPC) [1]

- GHC feature that adds *ticks* to every expression
- RTS keeps track of how often each tick occurred
- Used for program coverage
- **Built-in** to GHC:
  - Scalable
  - Easy to use (one flag)
  - Low-cost (bumps an array entry)

[1] Andy Gill and Colin Runciman. 2007. Haskell Program Coverage. In *Proceedings of the ACM SIGPLAN Workshop on Haskell Workshop (Freiburg, Germany) (Haskell '07)*. Association for Computing Machinery, New York, NY, USA, 1–12.

Approach:

Error

Stack trace

```
divs: Prelude.head: empty list
CallStack (from HasCallStack):
  error, called at libraries/…/List.hs:1643:3 in base:GHC.List
  errorEmptyList, called at libraries/…/List.hs:82:11 in base:GHC.List
  badHead, called at libraries/…/List.hs:78:28 in base:GHC.List
  head, called at Div.hs:10:17 in main:Main
```

```haskell
module Main where
divs :: Int -> [Int]
divs n = go 2
  where go i | i == n = []
        go i = if d i
               then i:(go (i+1))
               else go (i+1)
        d i = n `mod` i == 0

smallestDiv n = head (divs n)

main :: IO ()
main = print (smallestDiv 13)
```

Approach:

Error

Stack trace

```
divs: Prelude.head: empty list
CallStack (from HasCallStack):
  error, called at libraries/…/List.hs:1643:3 in base:GHC.List
  errorEmptyList, called at libraries/…/List.hs:82:11 in base:GHC.List
  badHead, called at libraries/…/List.hs:78:28 in base:GHC.List
  head, called at Div.hs:10:17 in main:Main
```

- **Extend HPC**

```
1  module Main where
2  divs :: Int -> [Int]
3  divs n = go 2
4    where go i | i == n = []
           go i = if d i
                  then i:(go (i+1))
                  else go (i+1)
         d i = n `mod` i == 0

10 smallestDiv n = head (divs n)
11
12 main :: IO ()
13 main = print (smallestDiv 13)
```

Approach:

Error

Stack trace

```
divs: Prelude.head: empty list
CallStack (from HasCallStack):
  error, called at libraries/…/List.hs:1643:3 in base:GHC.List
  errorEmptyList, called at libraries/…/List.hs:82:11 in base:GHC.List
  badHead, called at libraries/…/List.hs:78:28 in base:GHC.List
  head, called at Div.hs:10:17 in main:Main
```

- **Extend HPC**
- **Track** *recently* evaluated expressions in the RTS

```
1  module Main where
2  divs :: Int -> [Int]
3  divs n = go 2
4    where go i | i == n = []
           go i = if d i
                  then i:(go (i+1))
                  else go (i+1)
          d i = n `mod` i == 0

10 smallestDiv n = head (divs n)
11
12 main :: IO ()
13 main = print (smallestDiv 13)
```

Approach:

Error

Stack trace

```
divs: Prelude.head: empty list
CallStack (from HasCallStack):
  error, called at libraries/…/List.hs:1643:3 in base:GHC.List
  errorEmptyList, called at libraries/…/List.hs:82:11 in base:GHC.List
  badHead, called at libraries/…/List.hs:78:28 in base:GHC.List
  head, called at Div.hs:10:17 in main:Main
```

- **Extend HPC**
- **Track** *recently* evaluated expressions in the RTS
- **Summarize** the trace and report!

```
1  module Main where
2  divs :: Int -> [Int]
3  divs n = go 2
4    where go i | i == n = []
           go i = if d i
                  then i:(go (i+1))
                  else go (i+1)
          d i = n `mod` i == 0

10 smallestDiv n = head (divs n)
11
12 main :: IO ()
13 main = print (smallestDiv 13)
```

DEMO

```
[nix-shell:~/Code/ghc]$ lat Div.hs
```

File: Div.hs

```haskell
module Main where
divs :: Int -> [Int]
divs n = go 2
  where go i | i == n = []
        go i = if d i
               then i:(go (i+1))
               else go (i+1)
        d i = n `mod` i == 0

smallestDiv n = head (divs n)

main :: IO ()
main = print (smallestDiv 13)
```

```
[nix-shell:~/Code/ghc]$
```

```
[nix-shell:~/Code/ghc]$ lat Div.hs
```

File: Div.hs

```haskell
module Main where
divs :: Int -> [Int]
divs n = go 2
  where go i | i == n = []
        go i = if d i
               then i:(go (i+1))
               else go (i+1)
        d i = n `mod` i == 0

smallestDiv n = head (divs n)

main :: IO ()
main = print (smallestDiv 13)
```

```
[nix-shell:~/Code/ghc]$ rm -rf Div.hi Div.o divs &&
> _build/stage1/bin/ghc -fhpc -hpc-trace250 Div.hs -o divs &&
> ./divs
```

```
[nix-shell:~/Code/ghc]$ rm -rf Div.hi Div.o divs &&
> _build/stage1/bin/ghc -fhpc -hpc-trace250 Div.hs -o divs &&
> ./divs
[1 of 2] Compiling Main             ( Div.hs, Div.o )
[2 of 2] Linking divs
divs: Prelude.head: empty list
CallStack (from HasCallStack):
  error, called at libraries/base/GHC/List.hs:1643:3 in base:GHC.List
  errorEmptyList, called at libraries/base/GHC/List.hs:82:11 in base:GHC.List
  badHead, called at libraries/base/GHC/List.hs:78:28 in base:GHC.List
  head, called at Div.hs:10:17 in main:Main
Recently evaluated locations (from HPC):
  Div.hs:4:25-4:26  alternative branch taken
  Div.hs:4:16-4:21  guarded branch taken
  repeats (11 times):
    Div.hs:4:9-7:28   Main:divs>go
    Div.hs:7:21-7:28  alternative branch taken
    Div.hs:5:19-5:21  else branch taken
    Div.hs:8:9-8:28   Main:divs>d
    Div.hs:5:16-7:28  alternative branch taken
    Div.hs:4:16-4:21  guarded branch not taken
  Div.hs:4:9-7:28   Main:divs>go
  Div.hs:3:1-8:28   Main:divs
  Div.hs:10:1-10:29 Main:smallestDiv
  Div.hs:13:1-13:29 Main:main
  Div.hs:4:25-4:26  alternative branch taken
  Div.hs:4:16-4:21  guarded branch taken
  Div.hs:4:9-7:28   Main:divs>go
  Div.hs:7:21-7:28  alternative branch taken
  Div.hs:5:19-5:21  else branch taken
  Div.hs:8:9-8:28   Main:divs>d
  Div.hs:5:16-7:28  alternative branch taken
  Div.hs:4:16-4:21  guarded branch not taken
```

Results:

Error

Stack trace

```
divs: Prelude.head: empty list
CallStack (from HasCallStack):
  error, called at libraries/…/List.hs:1643:3 in base:GHC.List
  errorEmptyList, called at libraries/…/List.hs:82:11 in base:GHC.List
  badHead, called at libraries/…/List.hs:78:28 in base:GHC.List
  head, called at Div.hs:10:17 in main:Main

Recently evaluated locations (from CSI):
  Div.hs:4:25-4:26 alternative branch taken
  Div.hs:4:16-4:21 guarded branch taken
  repeats (11 times):
    Div.hs:4:9-7:28 Main:divs>go
    Div.hs:7:21-7:28 alternative branch taken
    Div.hs:5:19-5:21 else branch taken
    Div.hs:8:9-8:28 Main:divs>d
    Div.hs:5:16-7:28 alternative branch taken
    Div.hs:4:16-4:21 guarded branch not taken
  Div.hs:4:9-7:28 Main:divs>go
  Div.hs:3:1-8:28 Main:divs
  Div.hs:10:1-10:29 Main:smallestDiv
  Div.hs:13:1-13:29 Main:main
```

```haskell
1  module Main where
2  divs :: Int -> [Int]
3  divs n = go 2
4    where go i | i == n = []
           go i = if d i
                  then i:(go (i+1))
                  else go (i+1)
          d i = n `mod` i == 0

10 smallestDiv n = head (divs n)
11
12 main :: IO ()
13 main = print (smallestDiv 13)
```

Results:

Error

Stack trace

```
divs: Prelude.head: empty list
CallStack (from HasCallStack):
  error, called at libraries/…/List.hs:1643:3 in base:GHC.List
  errorEmptyList, called at libraries/…/List.hs:82:11 in base:GHC.List
  badHead, called at libraries/…/List.hs:78:28 in base:GHC.List
  head, called at Div.hs:10:17 in main:Main

Recently evaluated locations (from CSI):
  Div.hs:4:25-4:26 alternative branch taken
  Div.hs:4:16-4:21 guarded branch taken
  repeats (11 times):
    Div.hs:4:9-7:28 Main:divs>go
    Div.hs:7:21-7:28 alternative branch taken
    Div.hs:5:19-5:21 else branch taken
    Div.hs:8:9-8:28 Main:divs>d
    Div.hs:5:16-7:28 alternative branch taken
    Div.hs:4:16-4:21 guarded branch not taken
  Div.hs:4:9-7:28 Main:divs>go
  Div.hs:3:1-8:28 Main:divs
  Div.hs:10:1-10:29 Main:smallestDiv
  Div.hs:13:1-13:29 Main:main
```

```
1  module Main where
2  divs :: Int -> [Int]
3  divs n = go 2
4    where go i | i == n = []
           go i = if d i
                  then i:(go (i+1))
                  else go (i+1)
           d i = n `mod` i == 0

10 smallestDiv n = head (divs n)
11
12 main :: IO ()
13 main = print (smallestDiv 13)
```

Results:

PRODUCER

Error

is now mentioned!

Stack trace

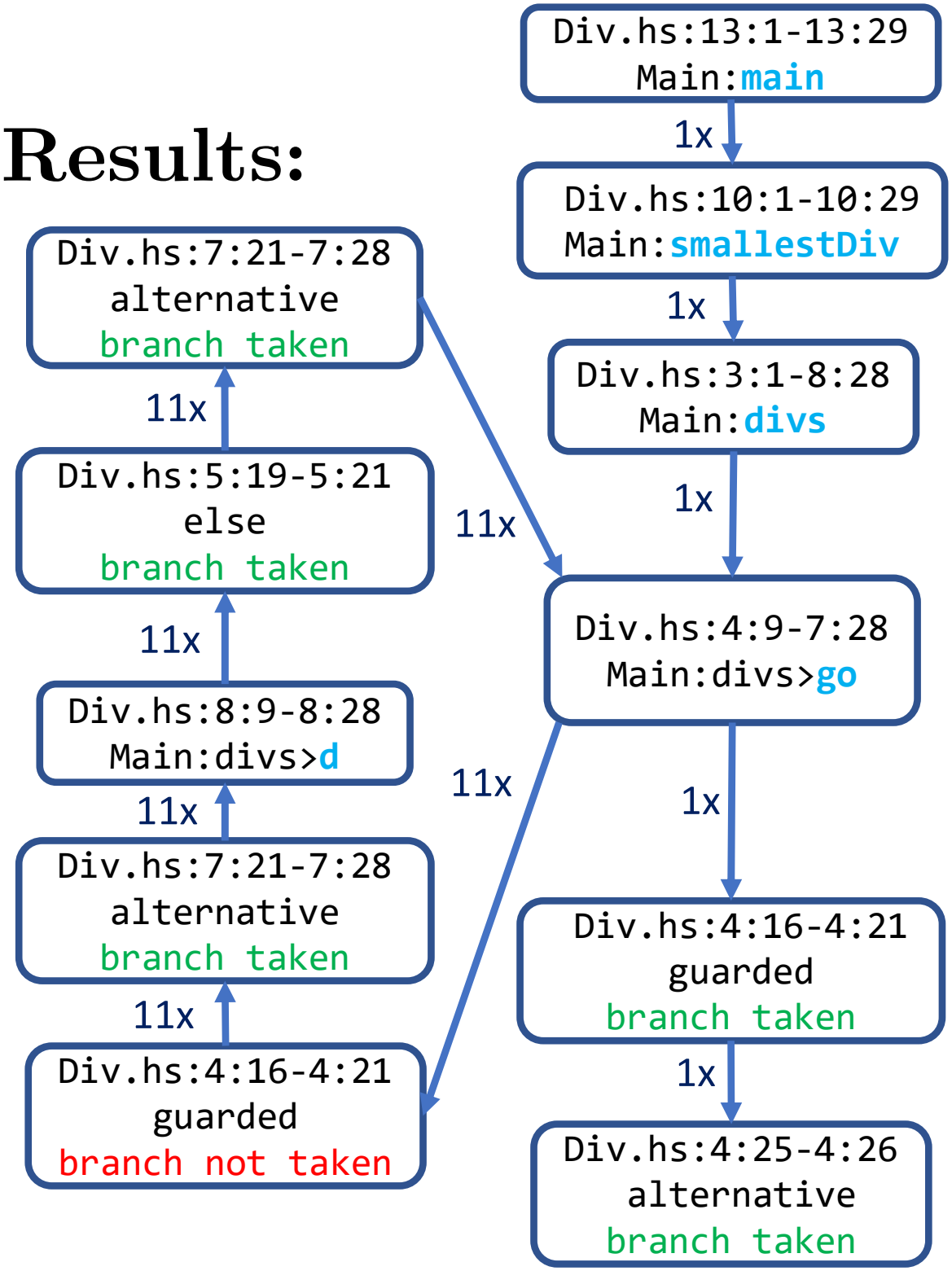
```
divs: Prelude.head: empty list
CallStack (from HasCallStack):
  error, called at libraries/…/List.hs:1643:3 in base:GHC.List
  errorEmptyList, called at libraries/…/List.hs:82:11 in base:GHC.List
  badHead, called at libraries/…/List.hs:78:28 in base:GHC.List
  head, called at Div.hs:10:17 in main:Main

Recently evaluated locations (from CSI):
  Div.hs:4:25-4:26 alternative branch taken
  Div.hs:4:16-4:21 guarded branch taken
  repeats (11 times):
    Div.hs:4:9-7:28 Main:divs>go
    Div.hs:7:21-7:28 alternative branch taken
    Div.hs:5:19-5:21 else branch taken
    Div.hs:8:9-8:28 Main:divs>d
    Div.hs:5:16-7:28 alternative branch taken
    Div.hs:4:16-4:21 guarded branch not taken
  Div.hs:4:9-7:28 Main:divs>go
  Div.hs:3:1-8:28 Main:divs
  Div.hs:10:1-10:29 Main:smallestDiv
  Div.hs:13:1-13:29 Main:main
```

```
1  module Main where
2  divs :: Int -> [Int]
3  divs n = go 2
4    where go i | i == n = []
           go i = if d i
                  then i:(go (i+1))
                  else go (i+1)
           d i = n `mod` i == 0

10 smallestDiv n = head (divs n)
11
12 main :: IO ()
13 main = print (smallestDiv 13)
```

Results:

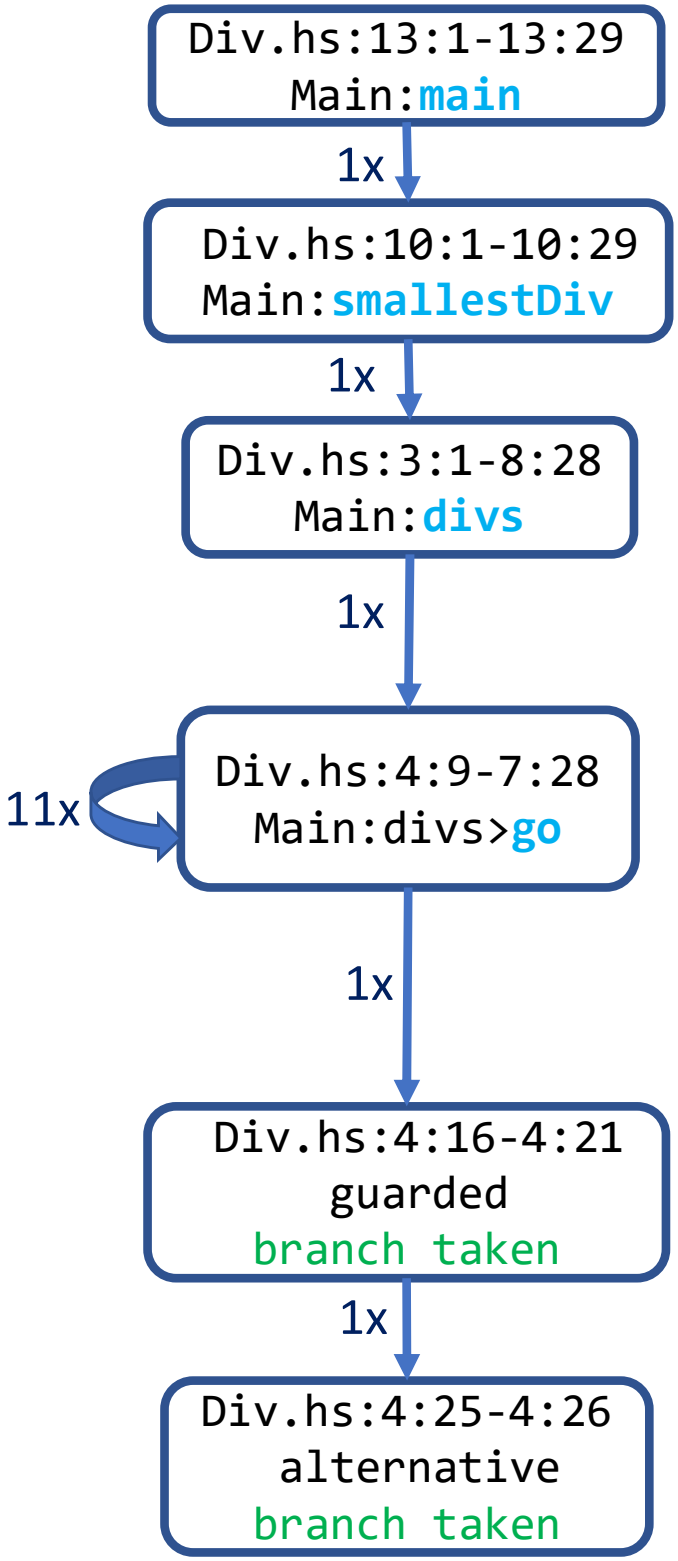



```haskell
module Main where
divs :: Int -> [Int]
divs n = go 2
  where go i | i == n = []
             go i = if d i
                    then i:(go (i+1))
                    else go (i+1)
           d i = n `mod` i == 0

smallestDiv n = head (divs n)

main :: IO ()
main = print (smallestDiv 13)
```

Results:

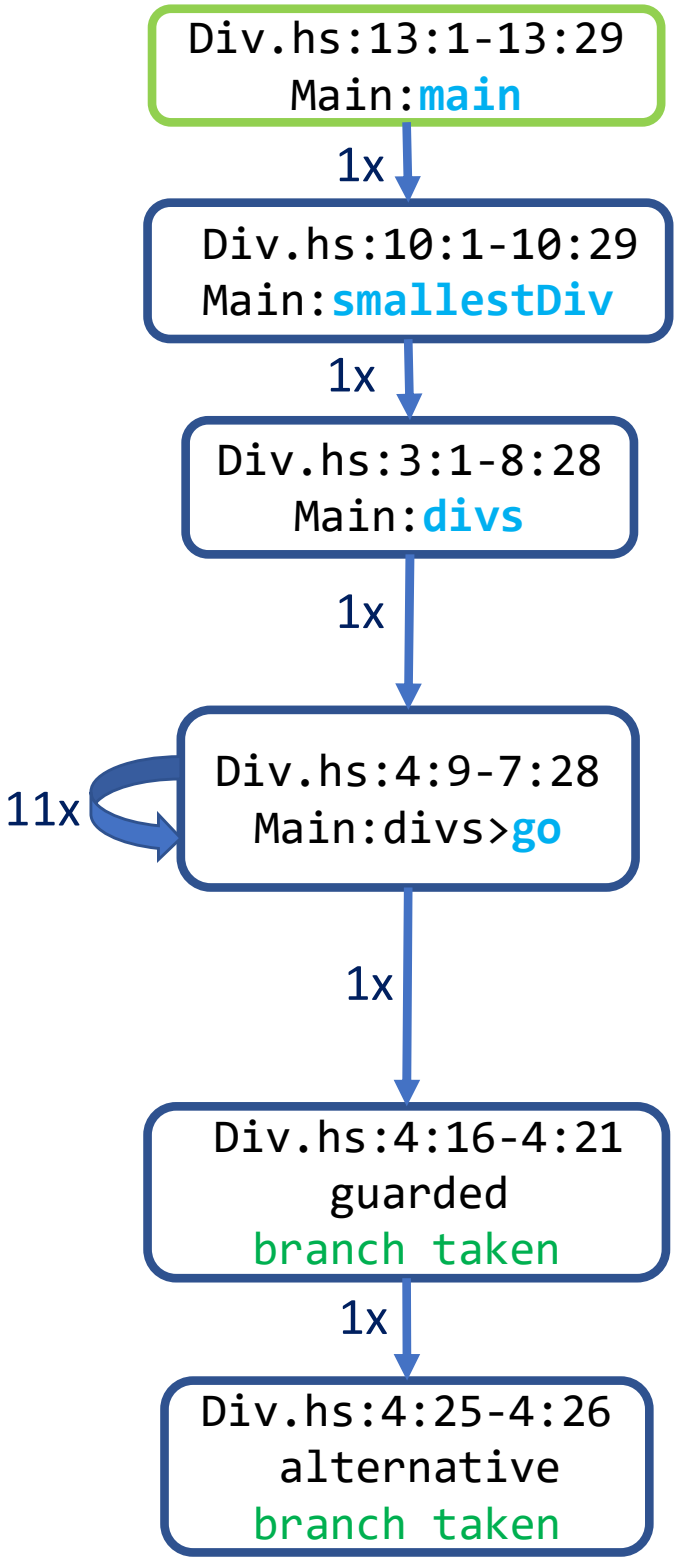



```haskell
module Main where
divs :: Int -> [Int]
divs n = go 2
  where go i | i == n = []
             go i = if d i
                    then i:(go (i+1))
                    else go (i+1)
             d i = n `mod` i == 0

smallestDiv n = head (divs n)

main :: IO ()
main = print (smallestDiv 13)
```

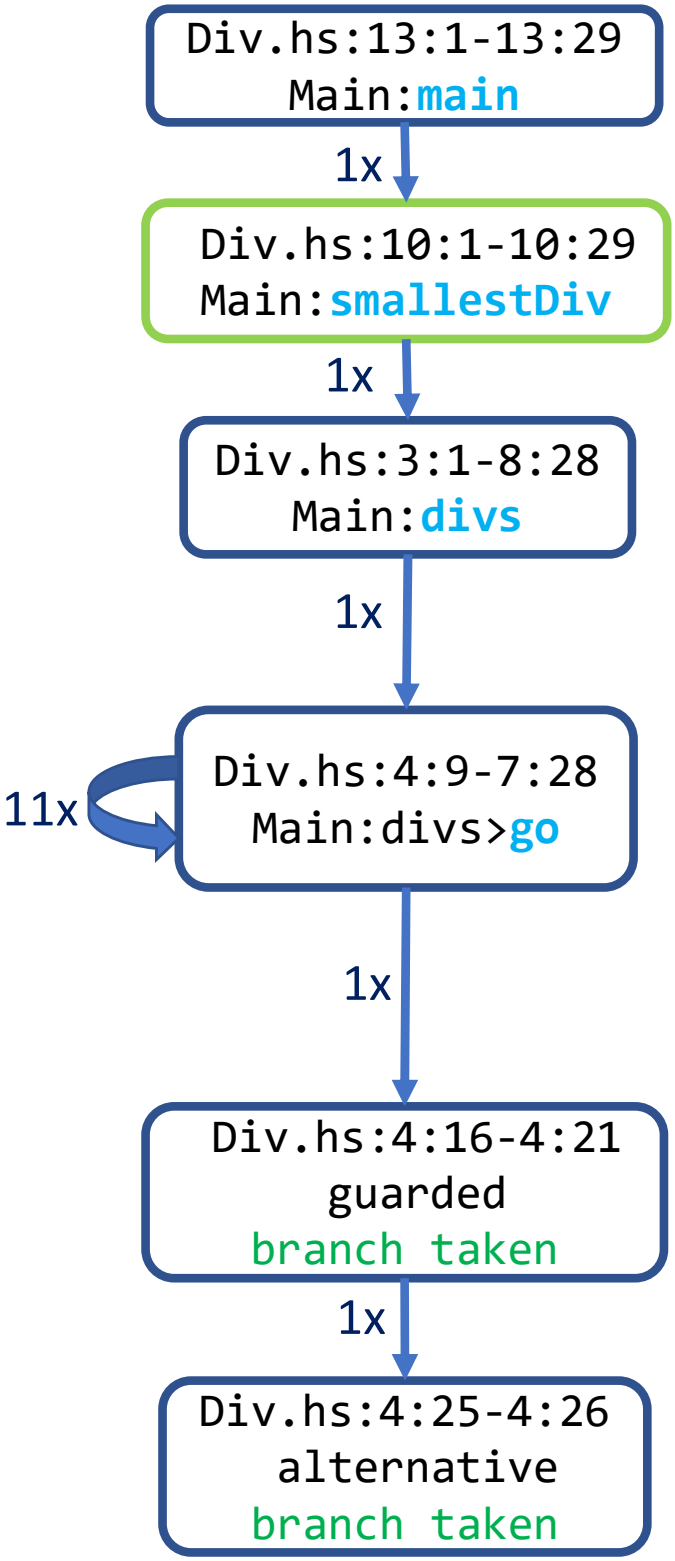
Results:




```
1  module Main where
2  divs :: Int -> [Int]
3  divs n = go 2
4    where go i | i == n = []
           go i = if d i
                  then i:(go (i+1))
                  else go (i+1)
         d i = n `mod` i == 0

10 smallestDiv n = head (divs n)
11
12 main :: IO ()
13 main = print (smallestDiv 13)
```

Results:

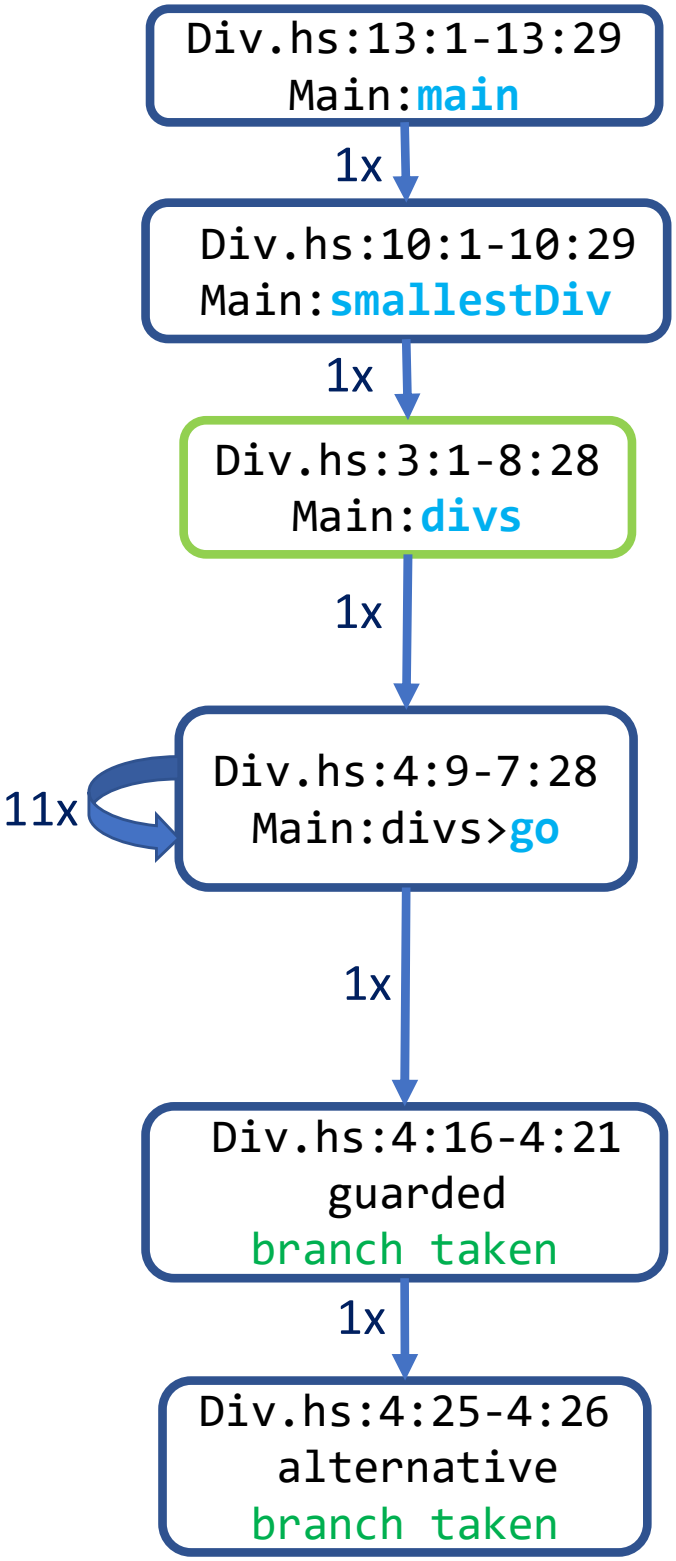



```haskell
module Main where
divs :: Int -> [Int]
divs n = go 2
  where go i | i == n = []
             go i = if d i
                    then i:(go (i+1))
                    else go (i+1)
           d i = n `mod` i == 0

smallestDiv n = head (divs n)

main :: IO ()
main = print (smallestDiv 13)
```

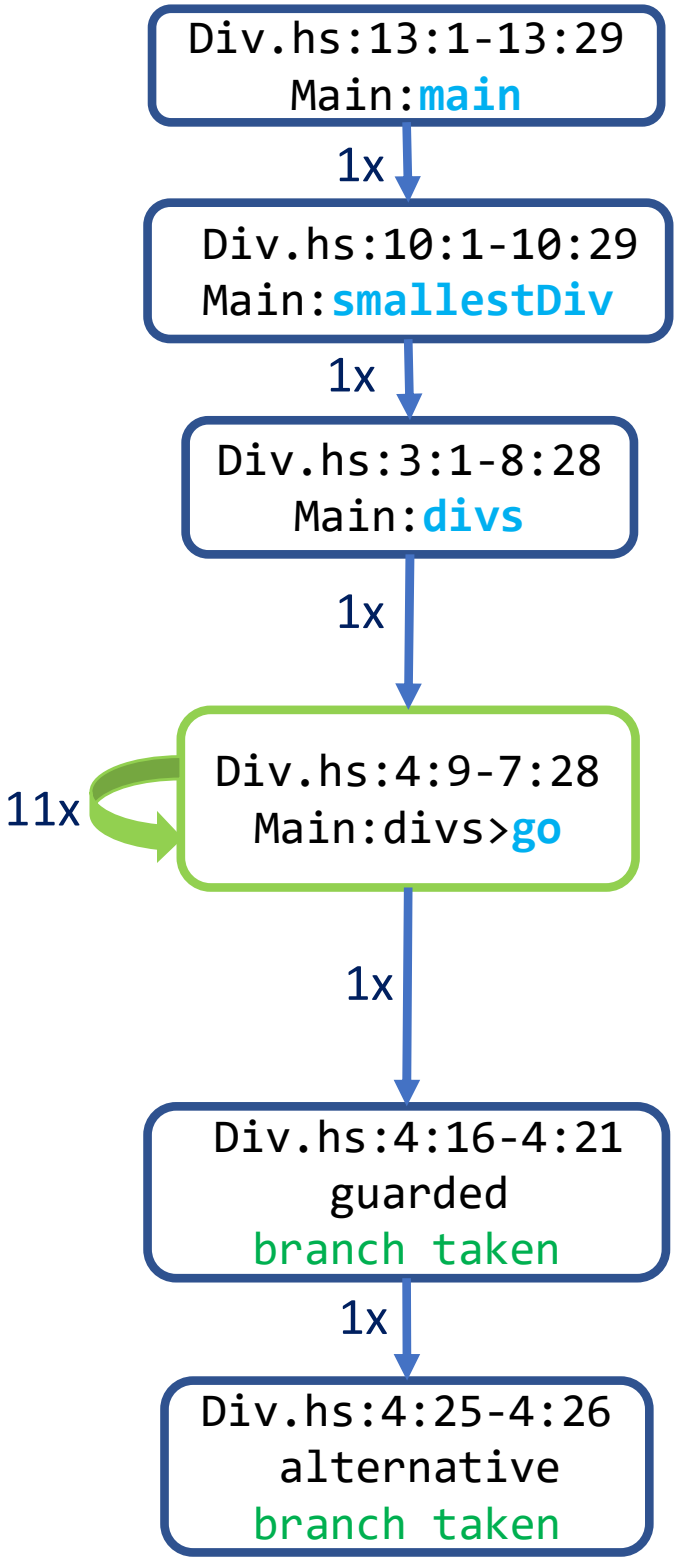
Results:




```
1  module Main where
2  divs :: Int -> [Int]
3  divs n = go 2
4    where go i | i == n = []
           go i = if d i
                  then i:(go (i+1))
                  else go (i+1)
           d i = n `mod` i == 0

10 smallestDiv n = head (divs n)
11
12 main :: IO ()
13 main = print (smallestDiv 13)
```

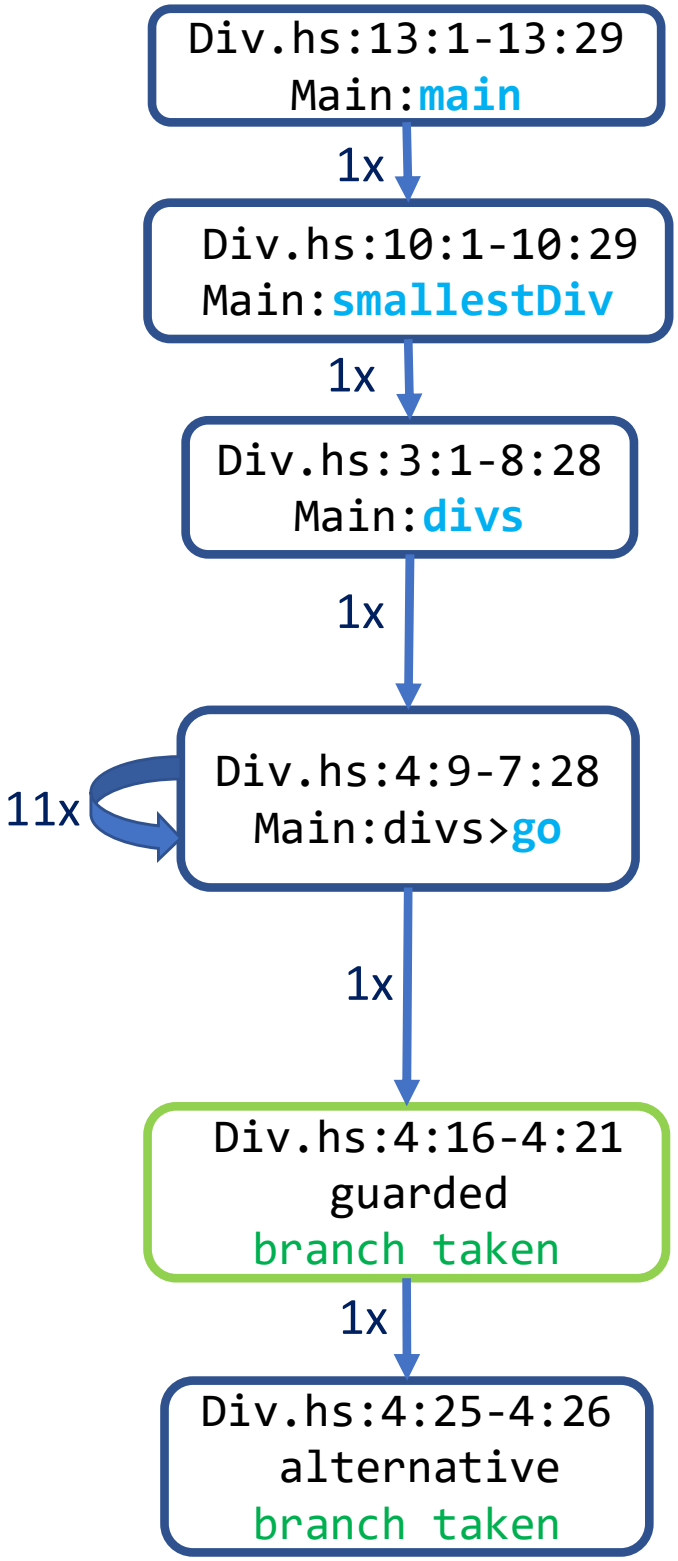
Results:




```
1  module Main where
2  divs :: Int -> [Int]
3  divs n = go 2
4    where go i | i == n = []
           go i = if d i
                  then i:(go (i+1))
                  else go (i+1)
           d i = n `mod` i == 0

10 smallestDiv n = head (divs n)
11
12 main :: IO ()
13 main = print (smallestDiv 13)
```

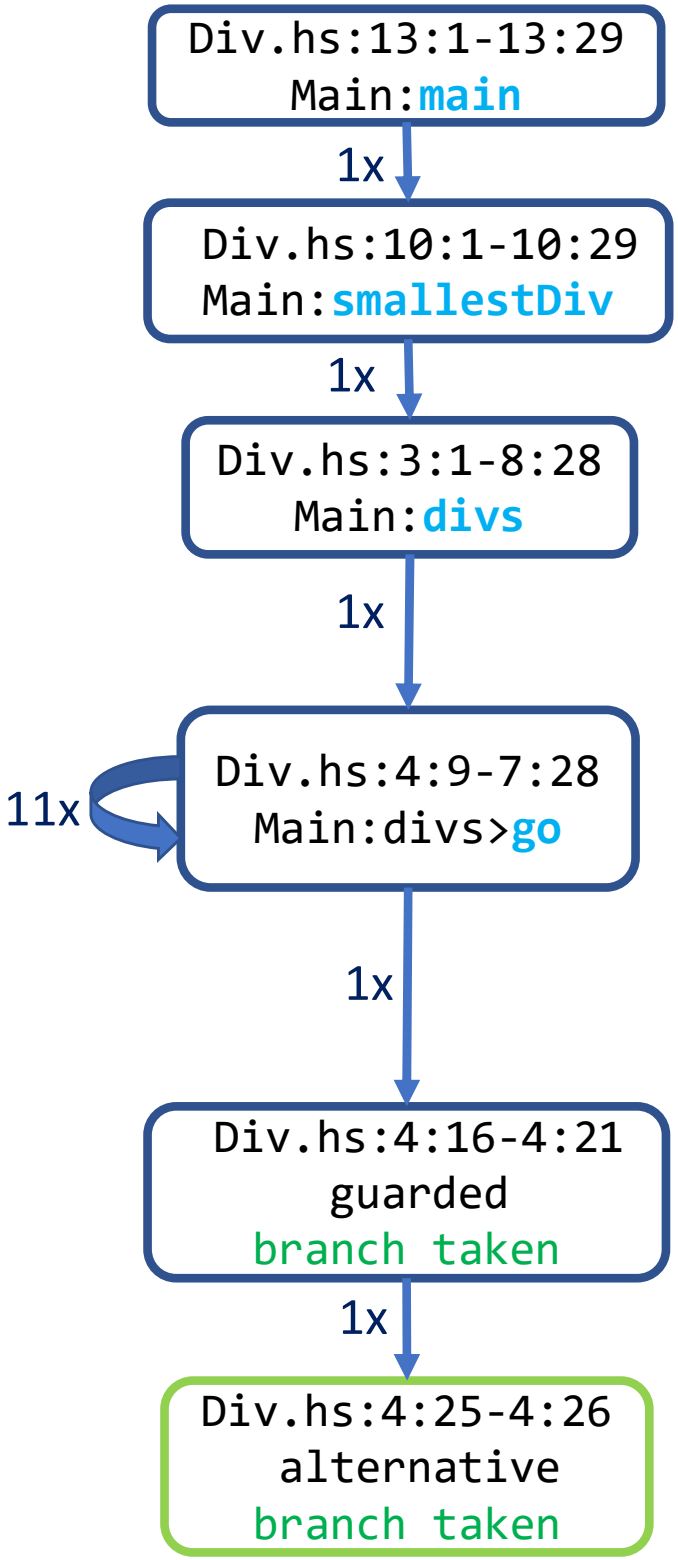
Results:




```
1  module Main where
2  divs :: Int -> [Int]
3  divs n = go 2
4    where go i | i == n = []
           go i = if d i
                  then i:(go (i+1))
                  else go (i+1)
         d i = n `mod` i == 0

10 smallestDiv n = head (divs n)
11
12 main :: IO ()
13 main = print (smallestDiv 13)
```

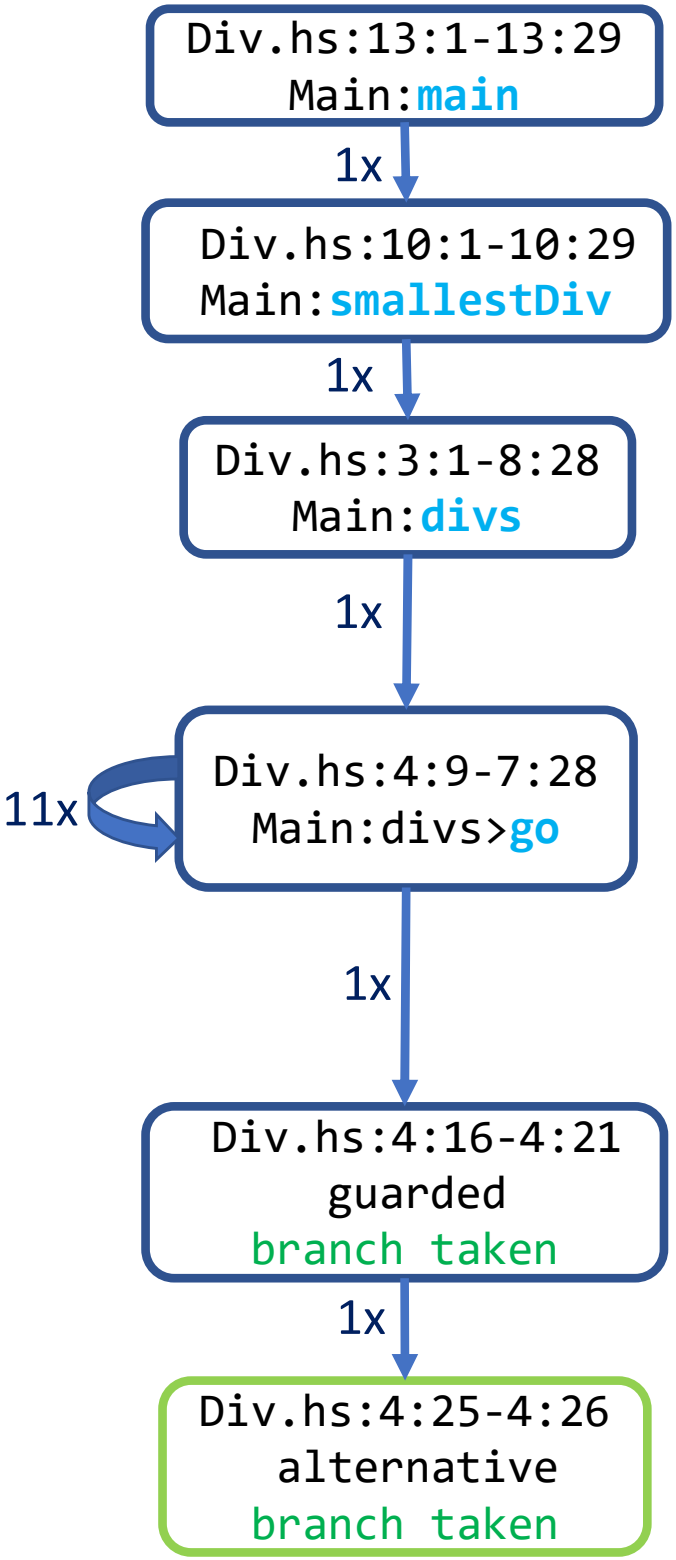
Results:




```
1  module Main where
2  divs :: Int -> [Int]
3  divs n = go 2
4    where go i | i == n = []
           go i = if d i
                  then i:(go (i+1))
                  else go (i+1)
         d i = n `mod` i == 0

10 smallestDiv n = head (divs n)
11
12 main :: IO ()
13 main = print (smallestDiv 13)
```

Results:

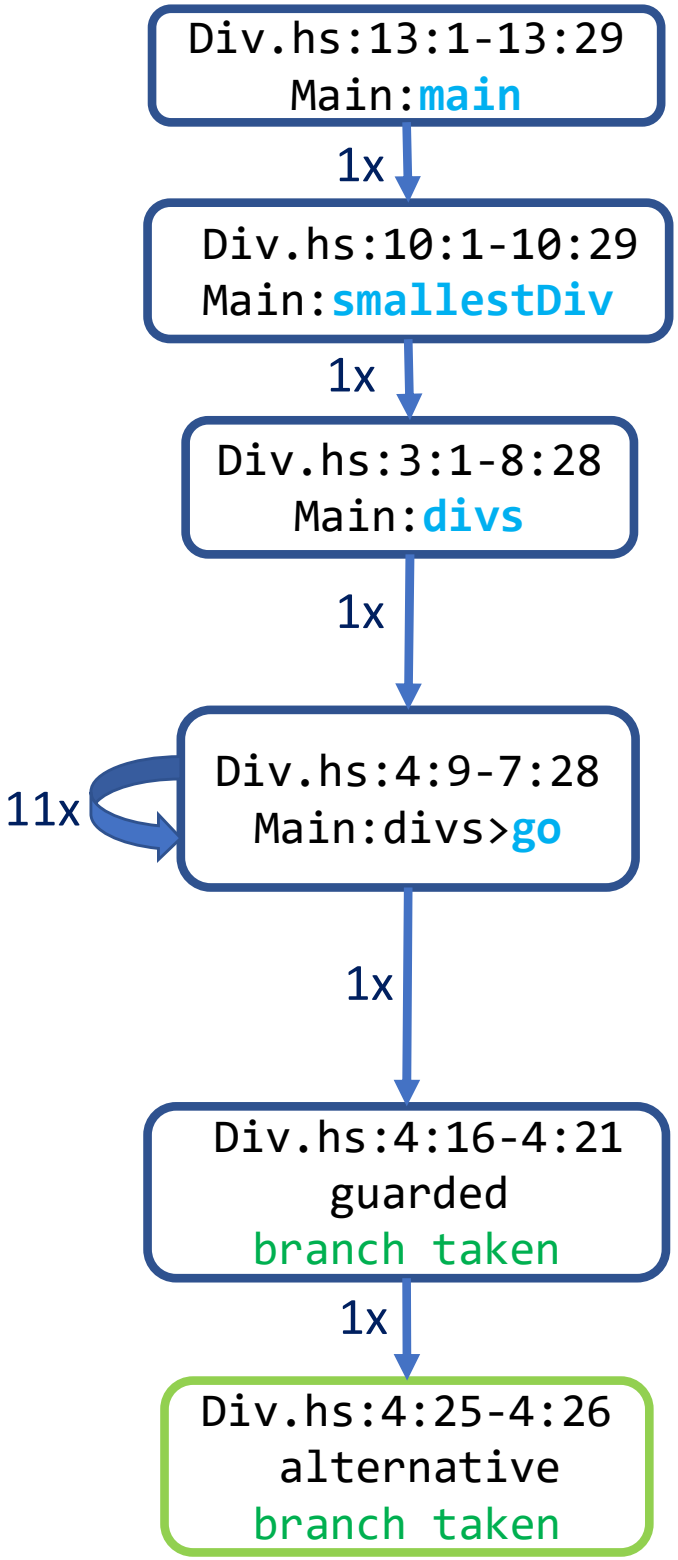


Like **HPC**, **CSI** is:

```
1  module Main where
2  divs :: Int -> [Int]
3  divs n = go 2
4    where go i | i == n = []
           go i = if d i
                  then i:(go (i+1))
                  else go (i+1)
         d i = n `mod` i == 0

10 smallestDiv n = head (divs n)
11
12 main :: IO ()
13 main = print (smallestDiv 13)
```

Results:



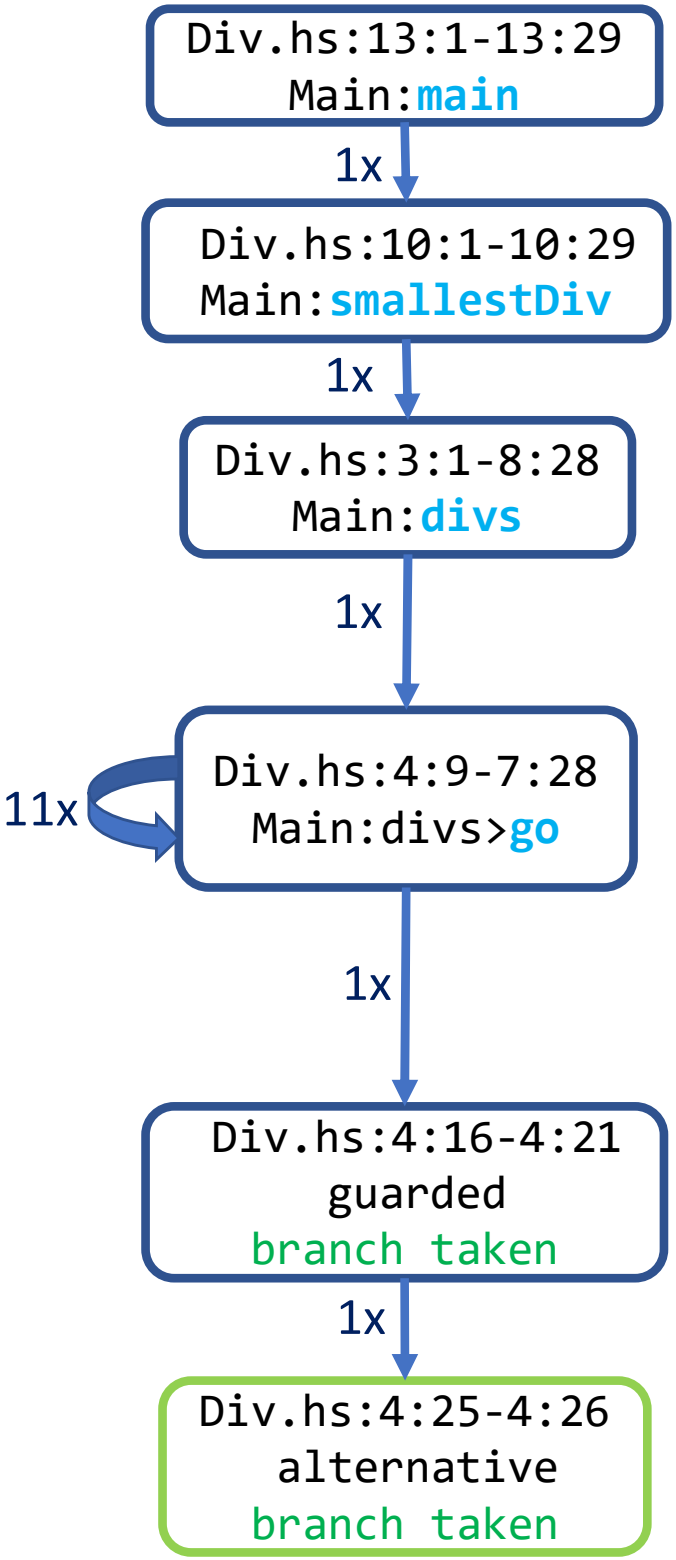
Like **HPC**, **CSI** is:

- Scalable

```
1  module Main where
2  divs :: Int -> [Int]
3  divs n = go 2
4    where go i | i == n = []
           go i = if d i
                  then i:(go (i+1))
                  else go (i+1)
         d i = n `mod` i == 0

10 smallestDiv n = head (divs n)
11
12 main :: IO ()
13 main = print (smallestDiv 13)
```

Results:



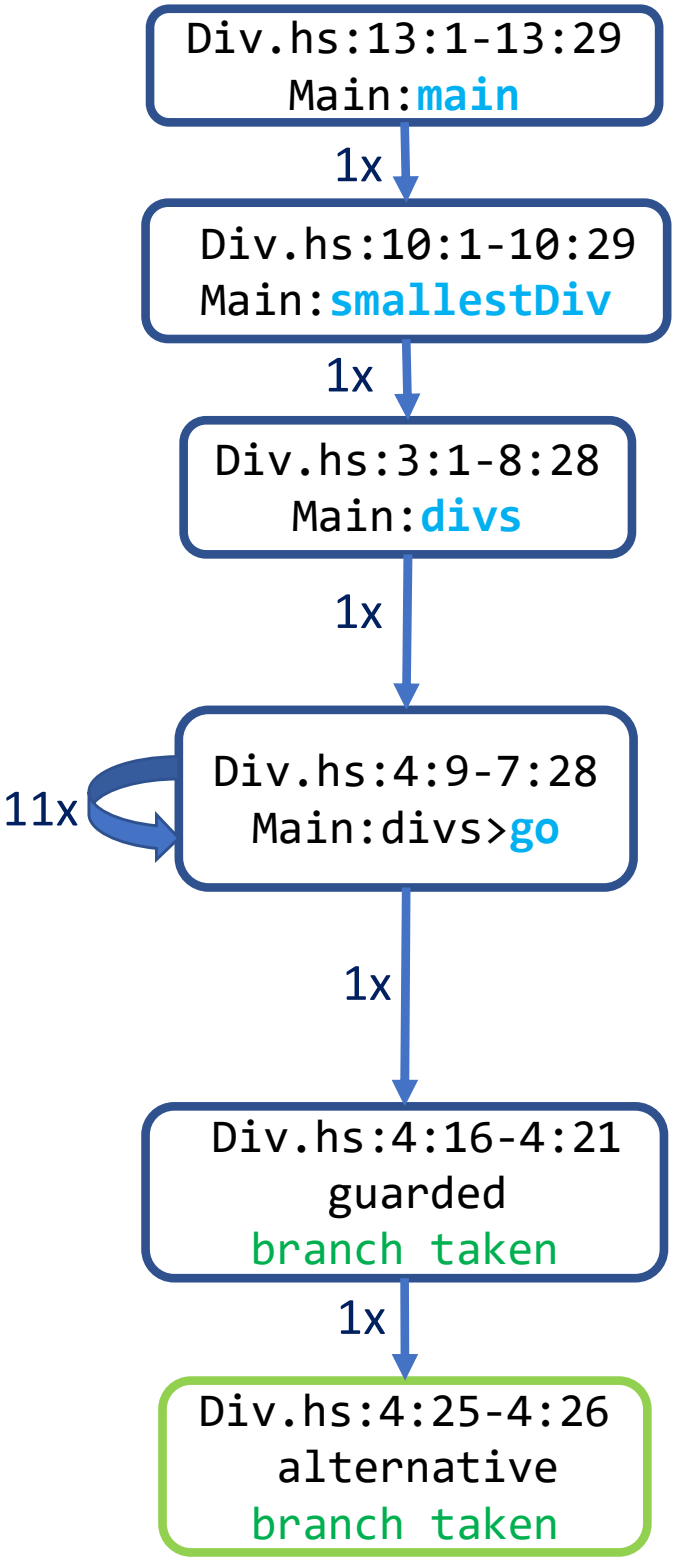
Like **HPC**, **CSI** is:

- Scalable
- Easy to use (two flags)

```
1  module Main where
2  divs :: Int -> [Int]
3  divs n = go 2
4    where go i | i == n = []
           go i = if d i
                  then i:(go (i+1))
                  else go (i+1)
         d i = n `mod` i == 0

10 smallestDiv n = head (divs n)
11
12 main :: IO ()
13 main = print (smallestDiv 13)
```

Results:



Like **HPC**, **CSI** is:

- Scalable
- Easy to use (two flags)
- Low-overhead

```
1  module Main where
2  divs :: Int -> [Int]
3  divs n = go 2
4    where go i | i == n = []
           go i = if d i
                  then i:(go (i+1))
                  else go (i+1)
         d i = n `mod` i == 0

10 smallestDiv n = head (divs n)
11
12 main :: IO ()
13 main = print (smallestDiv 13)
```

Future work

Future work

- Get HIW **feedback** and write GHC proposal!

Future work

- Get HIW **feedback** and write GHC proposal!
- How “close” are **producers** and **consumers**?

Future work

- Get HIW **feedback** and write GHC proposal!
- How “close” are **producers** and **consumers**?
- Better **summarization** via graph/source analysis

Future work

- Get HIW **feedback** and write GHC proposal!
- How “close” are **producers** and **consumers**?
- Better **summarization** via graph/source analysis
- How does strictness/sharing affect our analysis?

Future work

- Get HIW **feedback** and write GHC proposal!
- How “close” are **producers** and **consumers**?
- Better **summarization** via graph/source analysis
- How does strictness/sharing affect our analysis?
  - Could we turn these off? [2,3,4]

[2] Buiras, Pablo, and Alejandro Russo. "Lazy programs leak secrets." *Nordic Conference on Secure IT Systems*. Springer, Berlin, Heidelberg, 2013.
[3] Vassena, Marco, Joachim Breitner, and Alejandro Russo. "Securing concurrent lazy programs against information leakage." *2017 IEEE 30th Computer Security Foundations Symposium (CSF)*. IEEE, 2017.
[4] Breitner, Joachim. "dup--Explicit un-sharing in Haskell." *arXiv preprint arXiv:1207.2017* (2012).

Thank You!

Questions?

@tritlo

Artifact available!

https://l.mpg.is/ghc-csi

Bonus:
Comparison to HAT[5]

- No program transformation beyond HPC
- Easy to use (two flags)
- Low-cost (write to circular buffer)
- Only traces *recent* expressions, bounding costs
- Extendible by external tools

[5] Olaf Chitil, Colin Runciman, and Malcolm Wallace. 2002. Transforming Haskell for tracing. In *Symposium on Implementation and Application of Functional Languages*. Springer, 165–181.