Locating Type Errors *Speedily* with Delta Debugging

Joanna Sharrad
University of Kent
jks31@kent.ac.uk

Olaf Chitil
University of Kent
oc@kent.ac.uk

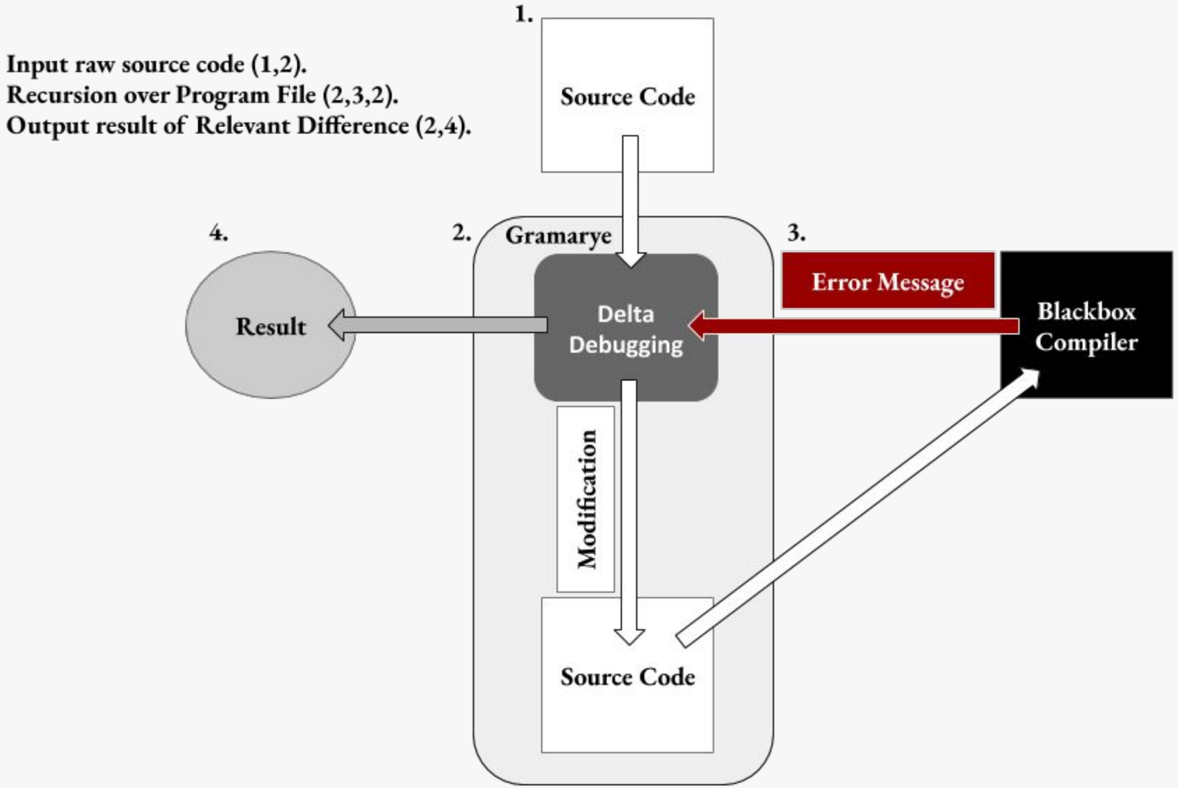
Example

Ill-typed case statement:

```
f x = case x of
    0 -> [0]
    1 -> 1
```

```
f x = case x of

Non type-variable argument in the
constraint: Num [a2]
```

Example from : Chen and Erwig. 2014. Counter-factual typing for debugging type errors.



Sharrad, J., Chitil, O., Wang, M. 2018. Delta Debugging Type Errors with a Blackbox Compiler.

Example

This code has a type error.

```
f x = case x of
    0 -> [0]
    1 -> 1
```

Example from : Chen and Erwig. 2014. Counter-factual typing for debugging type errors.

Example

Applying Delta Debugging:

```
f x = case x of
    0 -> [0]
    1 -> 1
```

Example

Applying Delta Debugging:

```
f x = case x of
    0 -> [0]

```

PASS (Compiles)

```


    1 -> 1
```

UNRESOLVED

Application of the Moiety Algorithm

Pre-processing to avoids line-splits causing unresolveds

```
f x = case x of
   0 -> [0]
   1 -> 1
plus :: Int -> Int -> Int
plus = (+)
fib x = case x of
  0 -> f x
  1 -> f x
  n -> fib (n-1) `plus` fib (n-2)
```

Application of the Moiety Algorithm

```

    0 -> [0]
```

```
parse error on input
```

Application of the Moiety Algorithm

```
    1 -> 1
```

```
parse error on input
```

Application of the Moiety Algorithm

```
plus :: Int -> Int -> Int
```

```
not parse error on input

(3,4)
```

Application of the Moiety Algorithm

```




plus = (+)




```

```
not parse error on input

(3,4) (4,5)
```

Application of the Moiety Algorithm

```
fib x = case x of
```

```
not parse error on input

(3,4) (4,5) (5,6)
```

Application of the Moiety Algorithm

```
    0 -> f x
```

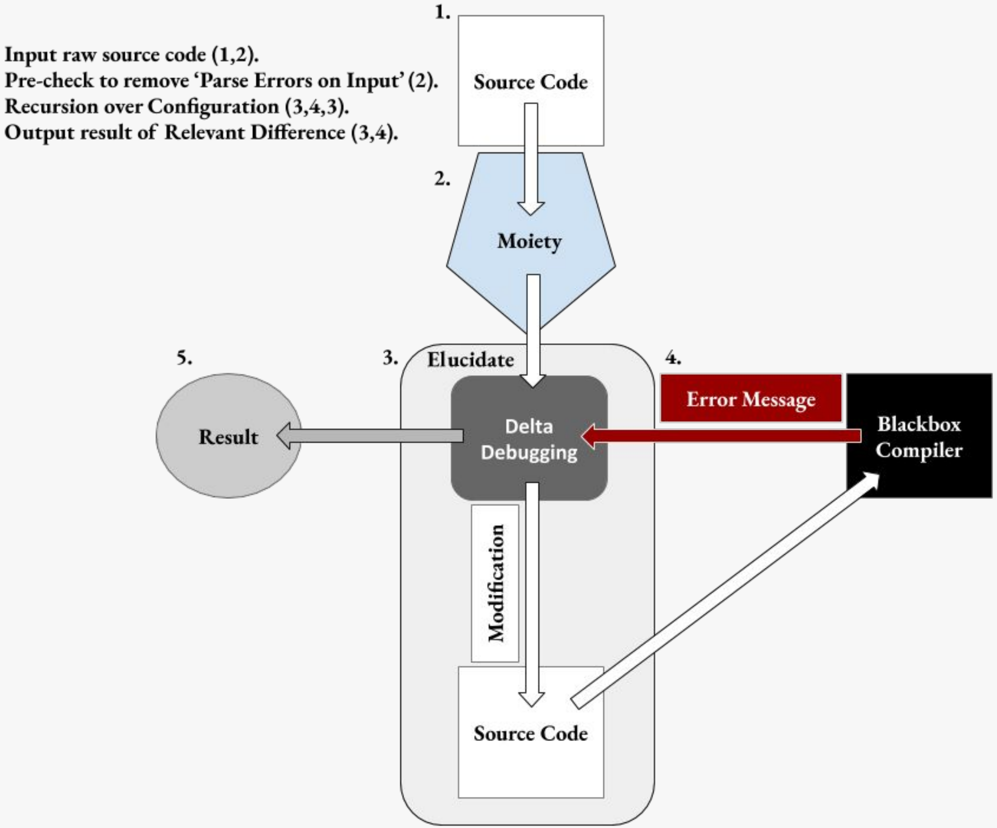
```
parse error on input

(3,4) (4,5) (5,6)
```

Application of the Moiety Algorithm

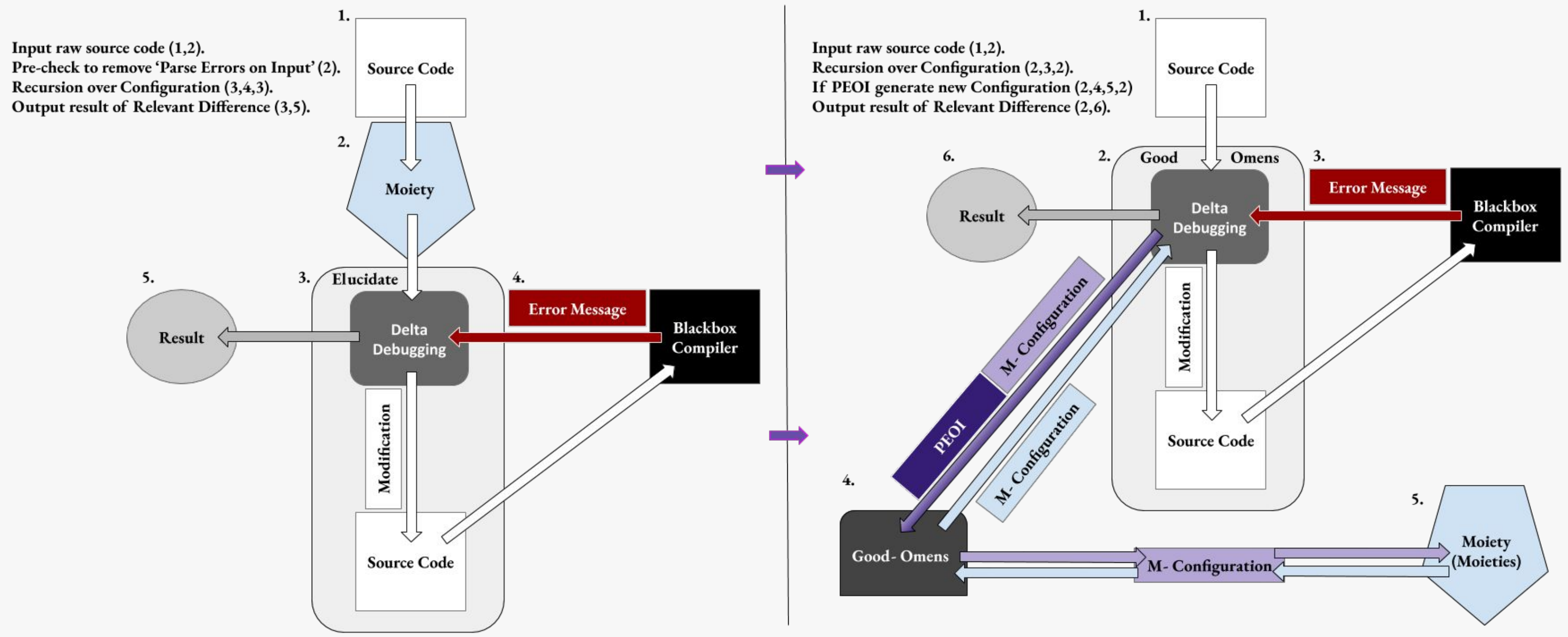
```
1| f x = case x of
2|    0 -> [0]
3|    1 -> 1
4| plus :: Int -> Int -> Int
5| plus = (+)
6| fib x = case x of
7|   0 -> f x
8|   1 -> f x
9|   n -> fib (n-1) `plus` fib (n-2)
```

Final Moieties (splitting points):

(3,4) (4,5) (5,6)



Sharrad and Chitil. 2020. Scaling Up Delta Debugging of Type Errors


Input raw source code (1,2).
Pre-check to remove 'Parse Errors on Input' (2).
Recursion over Configuration (3,4,3).
Output result of Relevant Difference (3,5).
1.
Source Code
2.
Moiety
3.
Elucidate
4.
5.
Result
Delta Debugging
Error Message
Blackbox Compiler
Modification
Source Code
Input raw source code (1,2).
Recursion over Configuration (2,3,2).
If PEOI generate new Configuration (2,4,5,2)
Output result of Relevant Difference (2,6).
1.
Source Code
6.
2.
Good
Omens
3.
Result
Delta Debugging
Error Message
Blackbox Compiler
M- Configuration
PEOI
M- Configuration
Modification
Source Code
4.
Good- Omens
M- Configuration
5.
Moiety (Moieties)

Applying the Good-Omens Algorithm

On request ‘parse error on input’ removal

```
f x = case x of
   0 -> [0]
   1 -> 1
plus :: Int -> Int -> Int
plus = (+)
fib x = case x of
  0 -> f x
  1 -> f x
  n -> fib (n-1) `plus` fib (n-2)
```

```
```

Applying the Good-Omens Algorithm

On request 'parse error on input' removal

```haskell
f x = case x of
   0 -> [0]
   1 -> 1
plus :: Int -> Int -> Int
plus = (+)
```

FAIL (Type Error)

```haskell
fib x = case x of
   0 -> f x
   1 -> f x
   n -> fib (n-1) `plus` fib (n-2)
```

UNRESOLVED

Applying the Good-Omens Algorithm

On request 'parse error on input' removal

```haskell
f x = case x of
   0 -> [0]
   1 -> 1
```

FAIL (Type Error)

```haskell
plus :: Int -> Int -> Int
plus = (+)
```

PASS (Compiles)

Applying the Good-Omens Algorithm

On request 'parse error on input' removal

```
f x = case x of
   0 -> [0]
```

PASS (Compiles)

```
   1 -> 1
plus :: Int -> Int -> Int
plus = (+)
```

Parse Error on Input

Applying the Good-Omens Algorithm

On request 'parse error on input' removal

```
f x = case x of
   0 -> [0]
```

PASS (Compiles)

```

   1 -> 1
plus :: Int -> Int -> Int
plus = (+)
```

Parse Error on Input

Applying the Good-Omens Algorithm

On request 'parse error on input' removal

```
Configuration Splitting Points:

{[1],[2],[3],[4],[5]}
```

```


   1 -> 1
plus :: Int -> Int -> Int
plus = (+)
```

Parse Error on Input

Applying the Good-Omens Algorithm

On request ‘parse error on input’ removal

```
Configuration Splitting Points:

{[1],[2],[3],[4],[5]}
```

```

   0 -> [0]
   1 -> 1
plus :: Int -> Int -> Int
plus = (+)




```

Applying the Good-Omens Algorithm

On request 'parse error on input' removal

```
Configuration Splitting Points:

{[1],[2,3],[4],[5]}
```

```

   0 -> [0]
   1 -> 1
plus :: Int -> Int -> Int
plus = (+)
```

Parse Error on Input

Applying the Good-Omens Algorithm

On request ‘parse error on input’ removal

```
Configuration Splitting Points:

{[1],[2,3],[4],[5]}
```

```
1| f x = case x of
2|    0 -> [0]
3|    1 -> 1
4| plus :: Int -> Int -> Int
5| plus = (+)
6|
7|
8|
9|
```

Applying the Good-Omens Algorithm

On request 'parse error on input' removal

```
Configuration Splitting Points:

{[1,2,3],[4],[5]}
```

```
f x = case x of
   0 -> [0]
   1 -> 1
plus :: Int -> Int -> Int
plus = (+)
```

FAIL (Type Error)

Applying the Good-Omens Algorithm

On request 'parse error on input' removal

```
f x = case x of
   0 -> [0]
   1 -> 1
```

FAIL (Type Error)

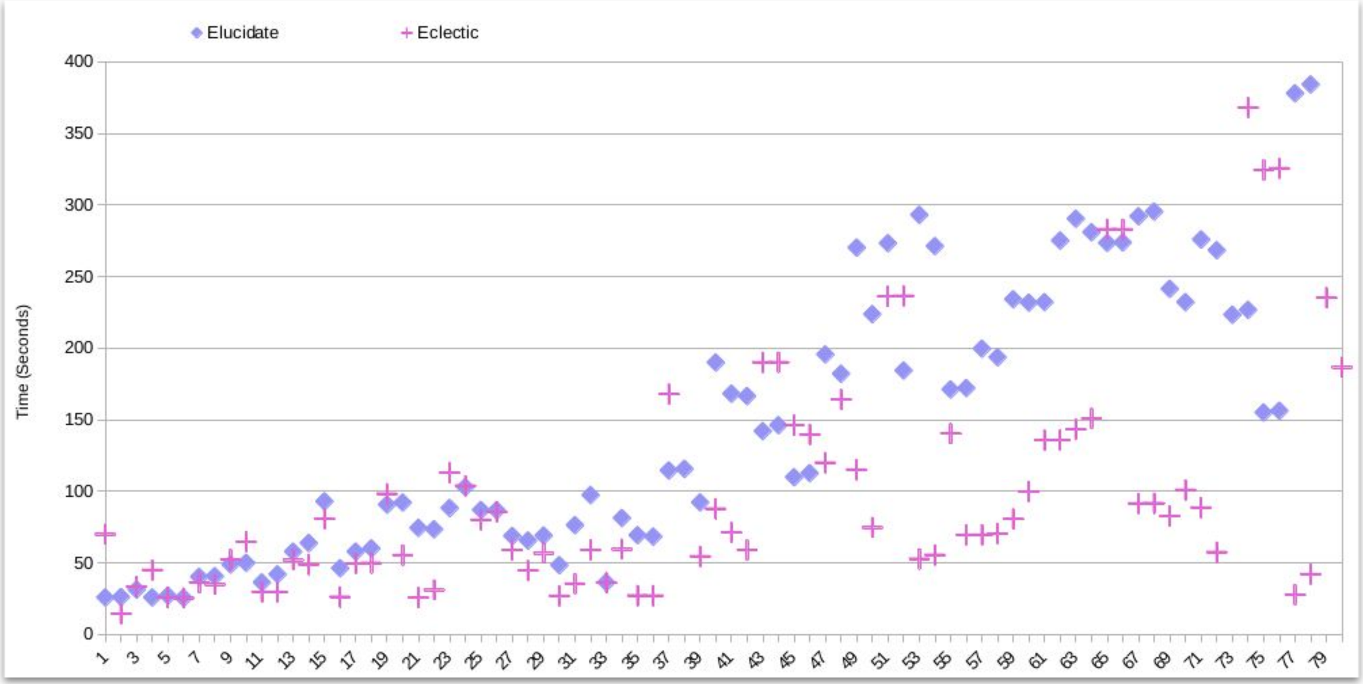
```
plus :: Int -> Int -> Int
plus = (+)
```

PASS (Compiles)

Evaluation

- Evaluated with a scalability data-set based on Pandoc
  - 80 type errors, 2 placed in each of 40 chosen modules
  - Modules have between 32 to 2305 lines of code
- Comparison with our moiety debugger Elucidate
- Can we reduce the debugger run-time?

Reduce the run-time of the



Evaluation

- Testing the Agnostic ability of the debugger
  - 11 test programs in both Haskell and Ocaml
  - Blackboxes: GHC and Ocamlc
- Promising initial results show debugger is agnostic
- Correctly located **8** identical type errors in both languages

Future Work

- Heuristics to choose pre-processing or request
- An empirical study with real programmers
- More evaluation of the debuggers agonistic behavior

Thank You

- Shown an agnostic type error debugger using a Blackbox compiler, Delta Debugging, Moiety and Good-Omens Algorithms
  - Increased the average speed by **1.5 minutes**
  - Best **38 minutes** faster, worst **8 minutes** slower
  - Agnostic debugging a success needing more research