Scaling Up Delta Debugging of Type Errors

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

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

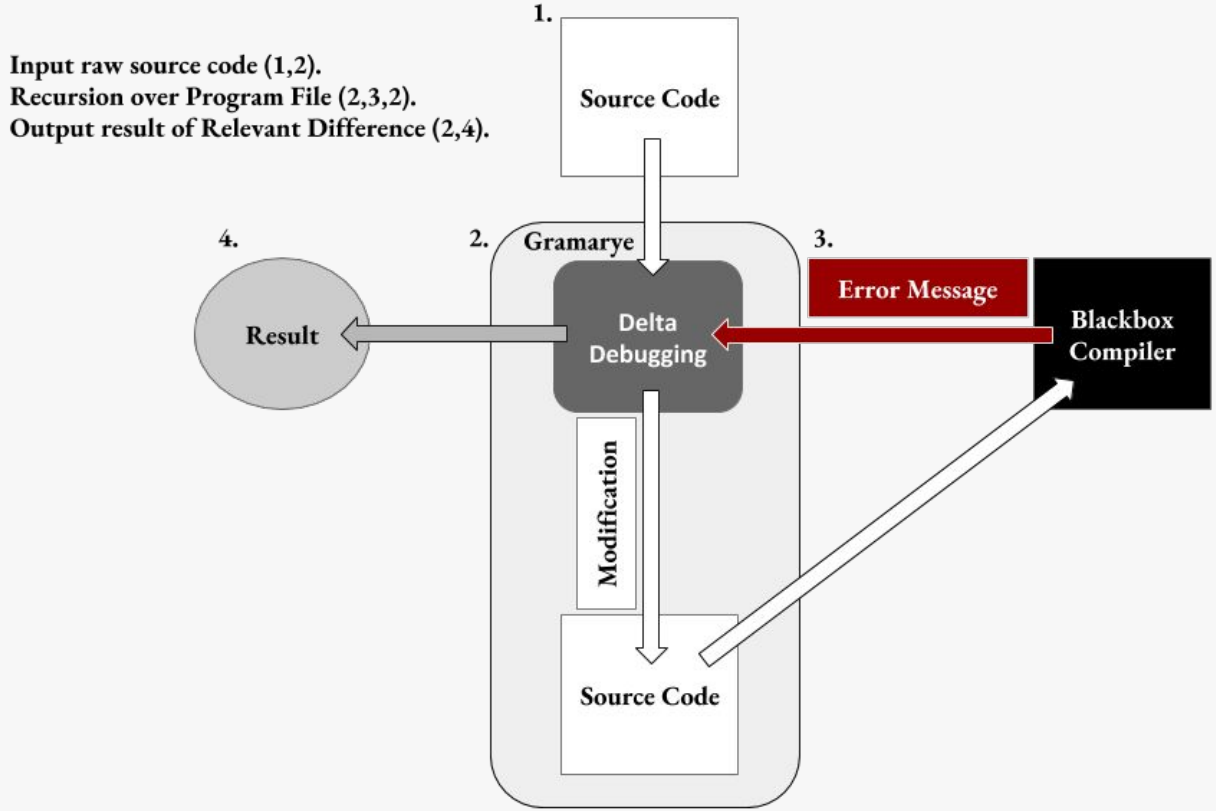
Example

Insert an element into an ordered list:

```
1. insert x [] =  x
2. insert x (y:ys) | x > y = y : insert x ys
3.                 | otherwise = x : y : ys
```

```
Insert.hs:2:27: error:
• Occurs check: cannot construct the
infinite type: a ~ [a]
• In the expression: y : Main.insert x ys
```

Example from : Stuckey, P., Sulzmann, M., Wazny, J. 2004. Improving type error diagnosis.



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

Example

This code has a type error.

```
1.   insert x [] =  x
2.   insert x (y:ys) | x > y = y : insert x ys
3.                  | otherwise = x : y : ys
```

Example from : Stuckey, P., Sulzmann, M., Wazny, J. 2004. Improving type error diagnosis.

Example

Applying Delta Debugging:

```
1. insert x [] = x
2. insert x (y:ys) | x > y = y : insert x ys
3.
```

```
1.
2.
3.                  | otherwise = x : y : ys
```

Example

Applying Delta Debugging:

```
1. insert x [] = x
2. insert x (y:ys) | x > y = y : insert x ys
3.
```

FAIL (Type Error)

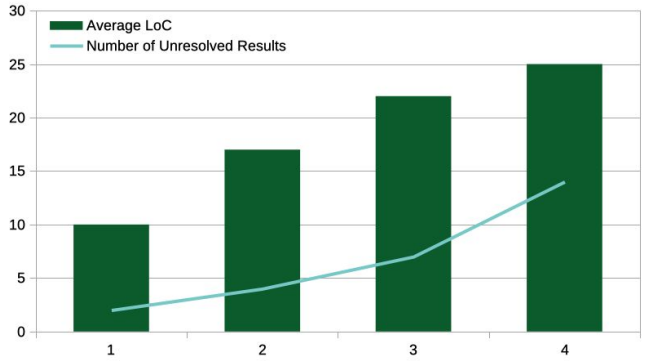
```
1.
2.
3.                    | otherwise = x : y : ys
```

UNRESOLVED

Can we scale up our debugger?

Average unresolved result for 900 program:

# lines	# unresolveds
10	2
17	4
22	7
25	14



Can we scale up our debugger?

Pandoc Module - 87 lines of code

error message	#
The last statement in a 'do' block must be an expression	4
Variable not in scope	4
Not in scope:	5
Empty 'do' block	5
Parse error (incorrect indentation or mismatched brackets)	7
Empty list of alternatives in case expression	8
The type signature...lacks an accompanying binding	16
Parse error on input	77
Total	126

Application of the Moiety Algorithm

Pre-processing to avoids line-splits causing unresolveds

```
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)
```

Application of the Moiety Algorithm

```
1|
2|    0 -> [0]
3|
4|
5|
6|
7|
8|
9|
```

```
parse error on input
```

Application of the Moiety Algorithm

```
1|
2|
3|    1 -> 1
4|
5|
6|
7|
8|
9|
```

```
parse error on input
```

Application of the Moiety Algorithm

```
1|
2|
3|
4| plus :: Int -> Int -> Int
5|
6|
7|
8|
9|
```

```
not parse error on input

(3,4)
```

Application of the Moiety Algorithm

```
1|
2|
3|
4|
5| plus = (+)
6|
7|
8|
9|
```

```
not parse error on input

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

Application of the Moiety Algorithm

```
1|
2|
3|
4|
5|
6| fib x = case x of
7|
8|
9|
```

```
not parse error on input

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

Application of the Moiety Algorithm

```
1|
2|
3|
4|
5|
6|
7|    0 -> f x
8|
9|
```

```
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)

Evaluation Framework

- A new type error evaluation framework for all
- Quantify the quality of the debugger
- Data Science - Accuracy, Recall, Precision, and F1 Score

Evaluation Framework

- **Accuracy:** Number of lines correctly excluded plus correctly reported lines containing a type error.
- **Recall:** Number of errors that are reported correctly compared to the number of errors within the source code.
- **Precision:** Number of correct lines of code reported by the debugger compared to the total number of lines returned.

Application of the Moiety Algorithm

The need for multiple metrics:

```
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)
```

- **L**ines of Code = 9
- **E**rrors in Code = 1
- **R**eturned **L**ines = 9
- Successfully **R**eturned **E**rrors = 1

Application of the Moiety Algorithm

The need for multiple metrics:

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

Recall:

$$\frac{R_E}{E} = \frac{1}{1} = 100\%$$

- **L**ines of Code = 9
- **E**rrors in Code = 1
- **R**eturned **L**ines = 9
- Successfully **R**eturned **E**rrors = 1

Application of the Moiety Algorithm

Recall = 100%

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

Precision:

$$\frac{R_E}{R_L} = \frac{1}{8} = 12.5\%$$

- **L**ines of Code = 9
- **E**rrors in Code = 1
- **R**eturned **L**ines = 9
- Successfully **R**eturned **E**rrors = 1

Application of the Moiety Algorithm

Recall = 100%, Precision = 12.5%

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

F1:

$$F_1 = 2\frac{R_E}{E + R_L} = 2\frac{1}{1+8} = 22\%$$

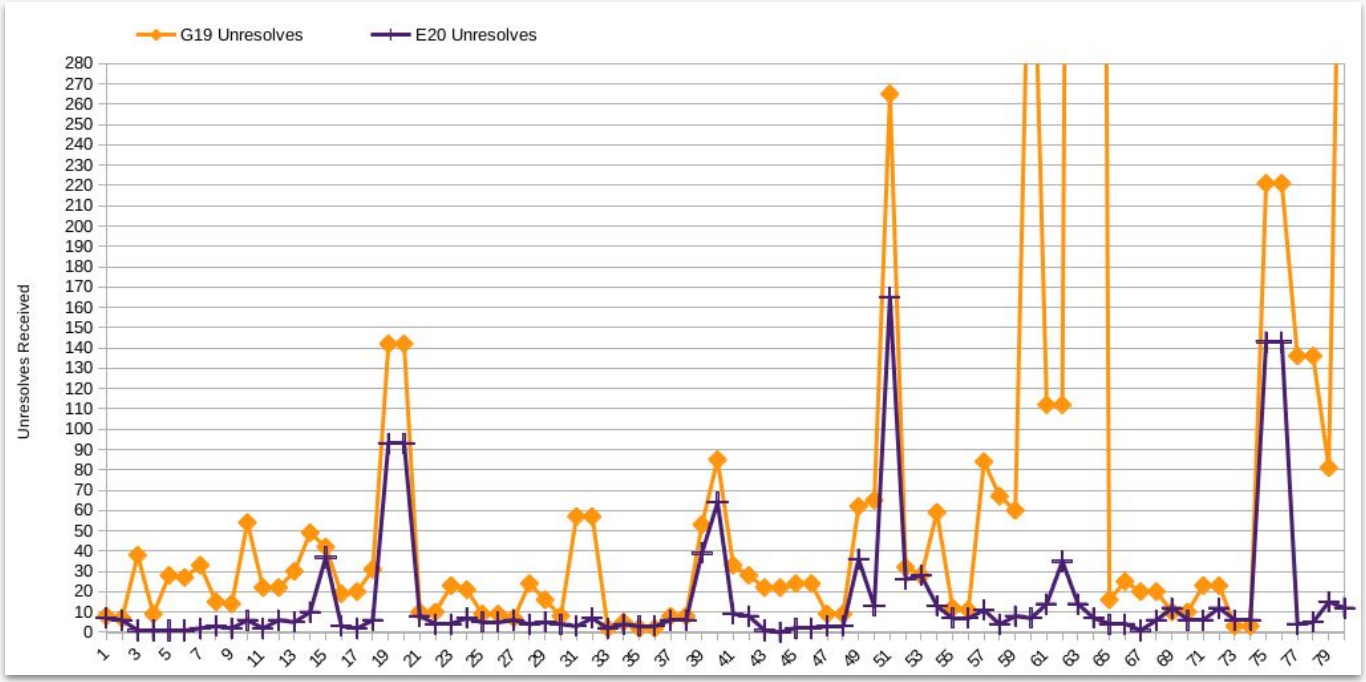
- **L**ines of Code = 9
- **E**rrors in Code = 1
- **R**eturned **L**ines = 9
- Successfully **R**eturned **E**rrors = 1

F1 gives us the harmony mean of the two metrics

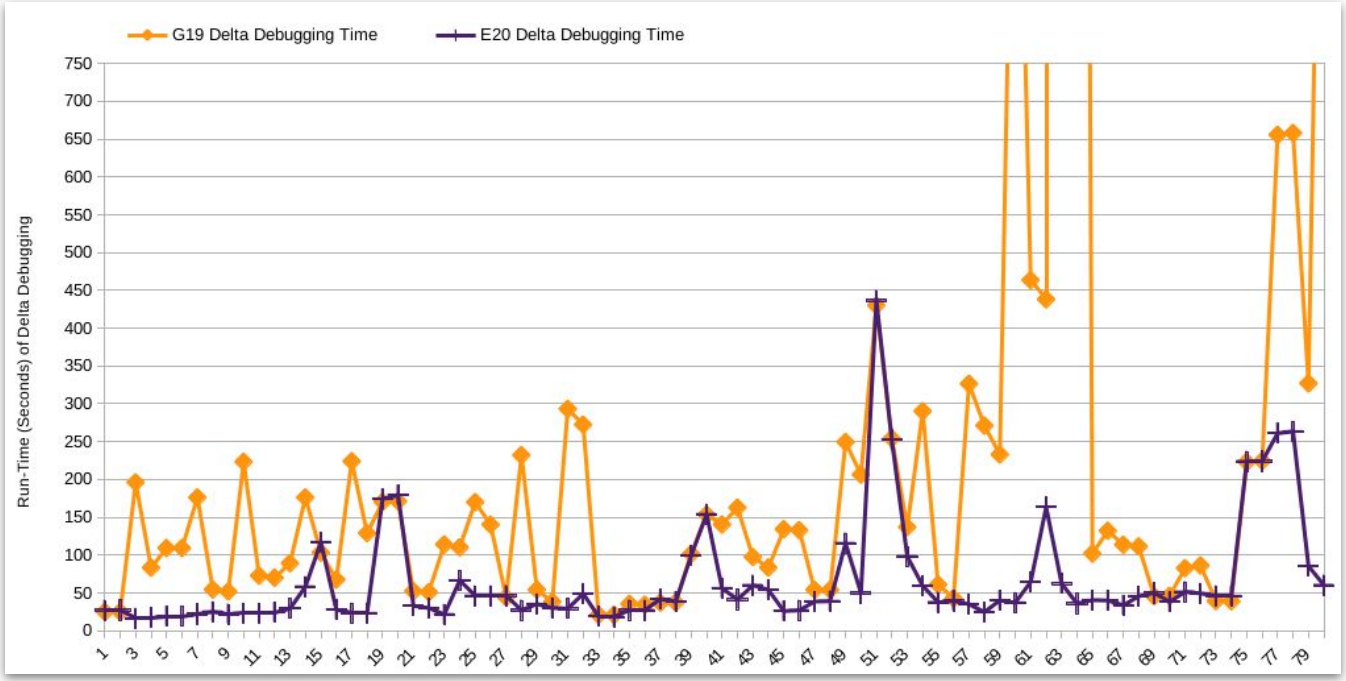
Evaluation

- A new 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 non-moiety debugger
- Can we reduce unresolved results and algorithm time?
- Does our framework quantify the quality of the debugger?

Reduce the number of unresolveds



Reduce Delta Debugging Time



The Evaluation Framework Figures

	Gramarye19(G19)	Elucidate20(E20)
Accuracy	94%	88%
Recall	38%	59%
Precision	16%	14%
F1	**20%**	**19%**

Future Work

- Reduction of the time Moiety takes
- Increase our scalability data-set with more large programs
- Make our debugger programming language agnostic

Thank You

- Shown a type error debugger using Delta Debugging, Blackbox compiler, and a **Moiety algorithm**
  - Introduced a scalability data-set
  - Introduced a new evaluation framework
  - Unresolved outcomes **lowered** by **82%**
  - **Reduced** Delta Debuggings run-time by **77%**