

Whiley: a Platform for Research in Verifying Compilers

David J. Pearce

Victoria University of Wellington

@WhileyDave

<http://whiley.org>

<http://github.com/Whiley>

Who Am I?



David Pearce
DavePearce

Victoria University of Wellington
 New Zealand
 david.pearce@ecs.vuw.ac.nz
 <http://www.ecs.vuw.ac.nz/~djp>
 Joined on Nov 16, 2010

13 Followers
10 Starred
0 Following

Organizations



Contributions Repositories Public activity

Edit profile

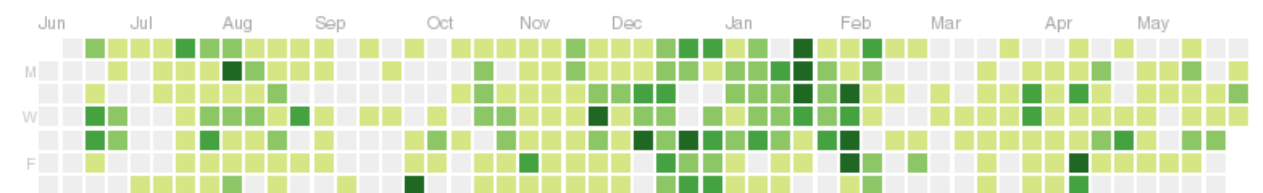
Popular repositories

jkit Java Compiler Kit (JKit)	2 ★
Wyscript A derivative of the Whiley language whic...	2 ★
Automark A Simple Web Framework used for Hobb...	1 ★
TuttePoly Tutte Polynomial Computation	1 ★
wyone A theorem prover generator	1 ★

Repositories contributed to

Whiley/WhileyCompiler The Whiley Compiler (WyC)	48 ★
Whiley/WhileyDocs Various documents relating to the While...	2 ★
Whiley/Wyclipse Eclipse Plugin for the Whiley Programmi...	3 ★
Whiley/WyBench Benchmark Suite for Whiley	4 ★
Whiley/WhileyCompilerCollection A compiler framework for managing the ...	0 ★

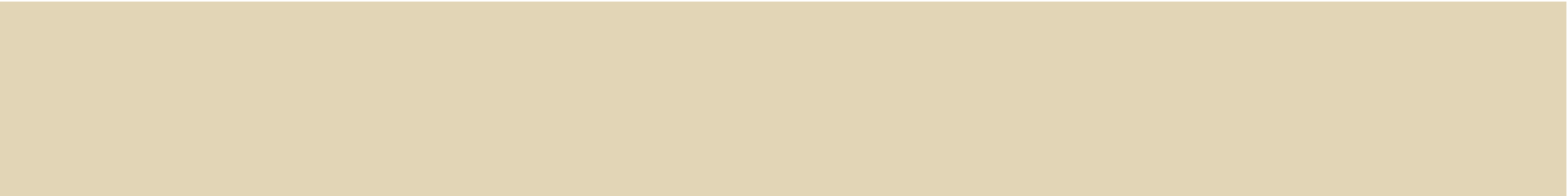
Contributions



Year of contributions
1,982 total
Jun 10 2013 - Jun 10 2014

Longest streak
45 days
October 29 - December 12

Current streak
3 days
June 09 - June 11



Background

Verification: A Challenge for Computer Science

*“A **verifying compiler** uses automated mathematical and logical reasoning methods to check the correctness of the programs that it compiles”*

—Hoare'03

Verification: Who Cares?

Search for content...

[National](#) [Opinion](#) [Business](#) [Technology](#) [World](#) [Sport](#) [Entertainment](#) [Life & Style](#)

[Business](#) [Small Business](#) [AroundNZ](#) [Economy](#) [Industries](#) [Property](#) [Personal Finance](#)

Provided by NZX

AIR	2.55	Open 2.525	High 2.555	Low 2.505	Bid Price 2.52
+\$0.025	+0.98%	Offer Price 2.555	Value 2910700.38	Volume 1142735	

Current as of 30/06/15 07:39PM NZST

**Grant Bradley** Aviation, tourism and energy writer for the Business Herald

Computers rebooted to tackle Dreamliner glitch

5:00 AM Tuesday May 5, 2015

[Air New Zealand Ltd](#) [Airlines](#) [Aviation](#)   116  19  0  0

Rebooting computers will overcome glitch that could cut all power.



Air New Zealand says it will comply with the directive to undertake a "power cycle" on its Dreamliners every three months. Photo / Brett Phibbs

**theguardian**

[home](#) [tech](#) [UK](#) [world](#) [politics](#) [sport](#) [football](#) [opinion](#) [cultu](#) [all](#)

Heartbleed

Heartbleed bug: what do you actually need to do to stay secure?

'Catastrophic' bug leaves thousands of sites vulnerable, but what exactly is Heartbleed and how does it affect me?



There is a very high chance that at least one online service that you use will be affected.
Photograph: Peter Titmus/Alamy

Samuel Gibbs
 @SamuelGibbs
Thursday 10 April 2014 14:29 BST

[Shares](#) [Comments](#)

1,577 **140**

Heartbleed is a [catastrophic bug that affects thousands of sites and services](#) across the internet, but what is it, and what do you need to do about it to protect yourself from cybercriminals?

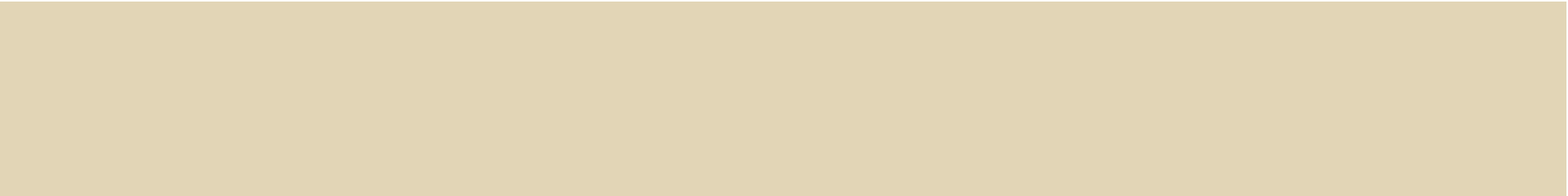


Whiley

Overview: What is Whiley?

```
function max(int x, int y)  $\rightarrow$  (int z)
// result must be one of the arguments
ensures x == z || y == z
// result must be greater-or-equal than arguments
ensures x <= z && y <= z :
    . . .
```

- A language designed specifically to simplify **verifying software**
- Several trade offs e.g. **performance for verifiability**
 - *Unbounded Arithmetic, value semantics, etc*
- **Goal:** to statically verify functions meet their specifications



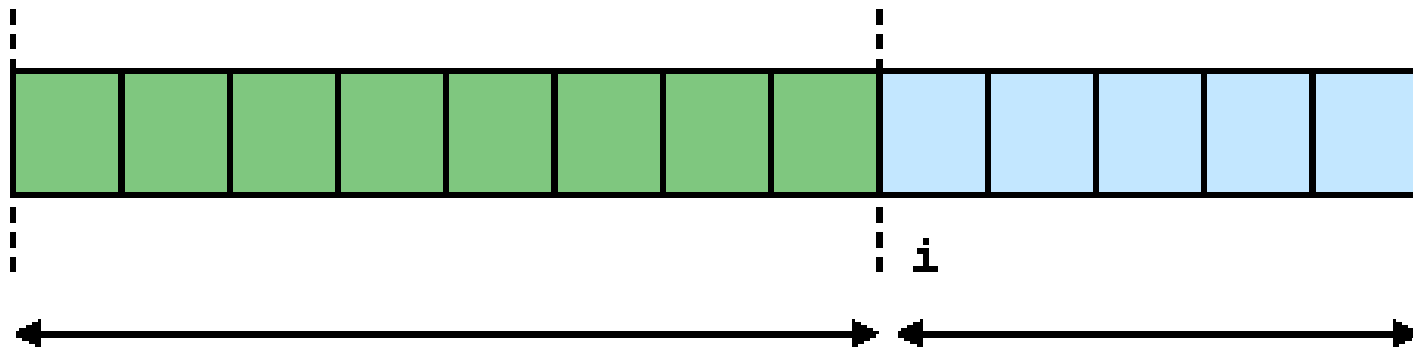
Welcome to the Future...

Example: `max (int [ ] )`

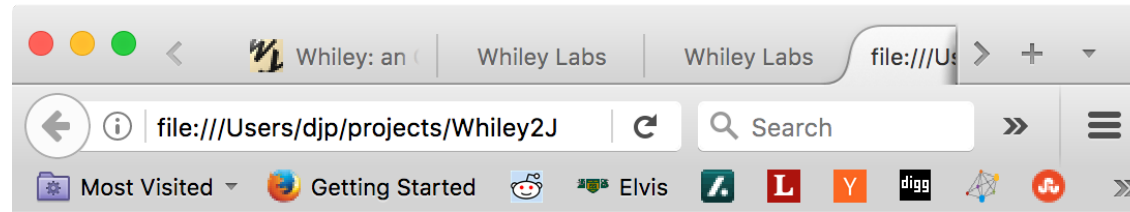
// Returns index of largest item in array

function `max (int [ ] items) → (int r)`

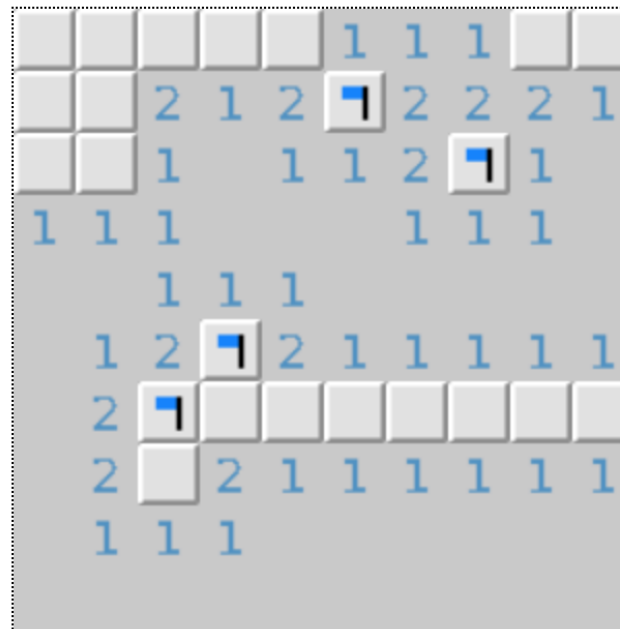
Diagram!



Minesweeper!



Minesweeper (in Whiley)

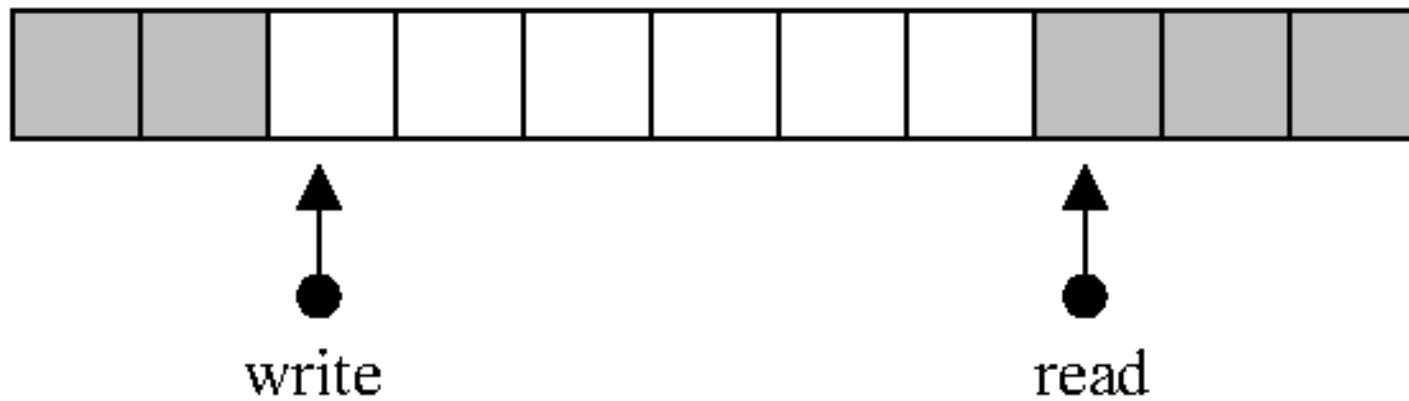


(use shift-click to flag squares)

Images



Cyclic Buffer!





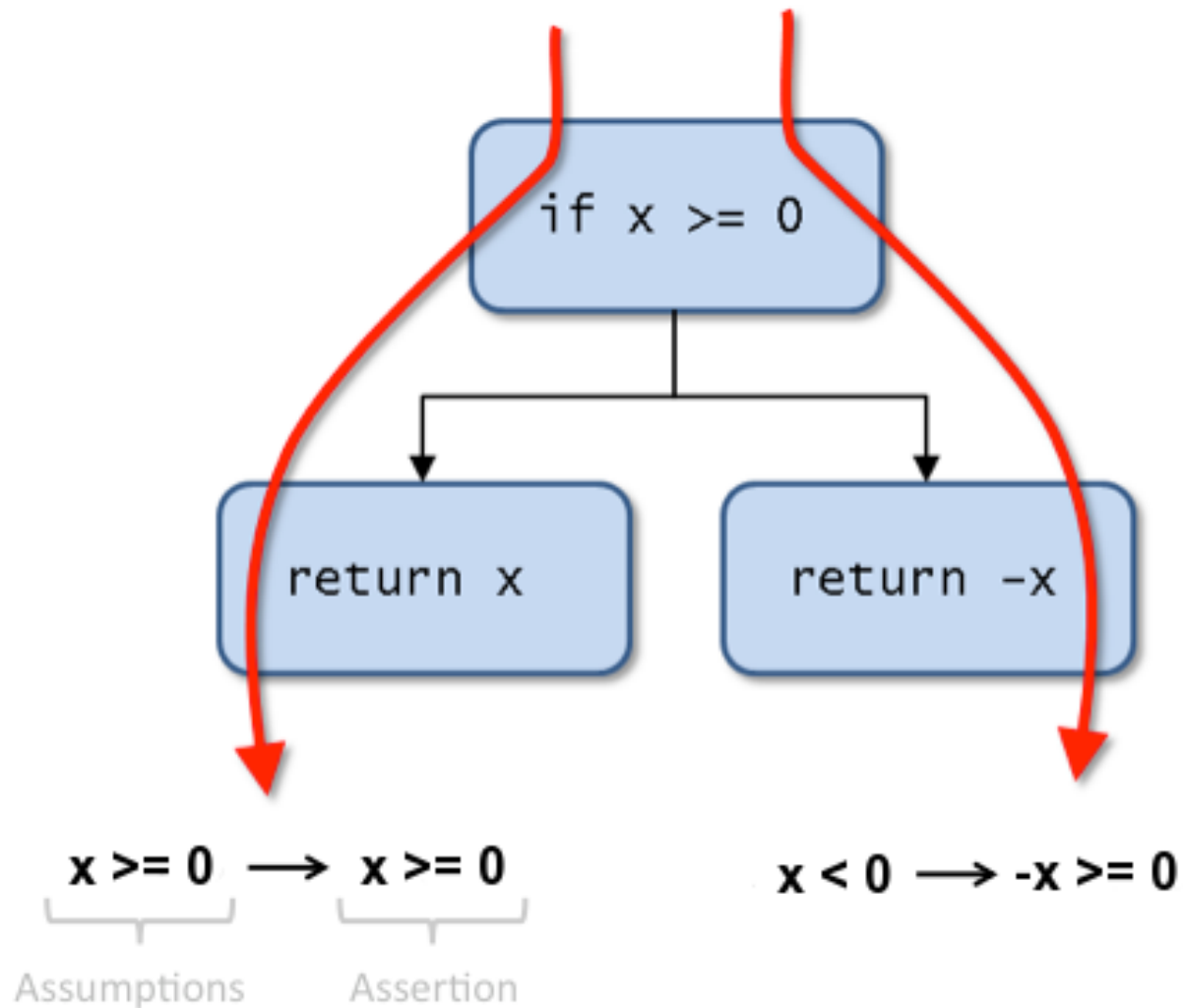
How does it work?

Verification: How does it work?

```
function abs (int x) => (int r)
// return value cannot be negative
ensures r >= 0:
    //
    if x >= 0:
        return x
    else:
        return -x
```

- To verify above function, compiler generates **verification conditions**
- Verification conditions are (roughly) **first-order logic formulas**

Verification: Verification Condition Generation



Verification: Assertion Language

- Whiley compiler emits verification conditions in **assertion language**

assert :

```
forall (int x) :  
    x >= 0 ==> x >= 0
```

assert :

```
forall (int x) :  
    x < 0 ==> -x >= 0
```

- Verification conditions from `abs()` example shown above
- In principle, can hook up different **automatic theorem provers**

People (so far)



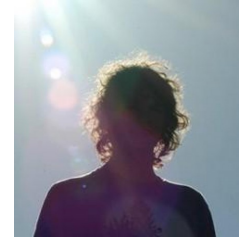
Art

(built C backend, 2012)



Melby

(built GPGPU backend,
2013)



Daniel

(helping with WhyleyWeb)



Matt

(compiling for a QuadCopter,
2014)



Henry

(improving verification, 2014)



Sam

(started PhD on
Parallelisation, 2014)



Lindsay

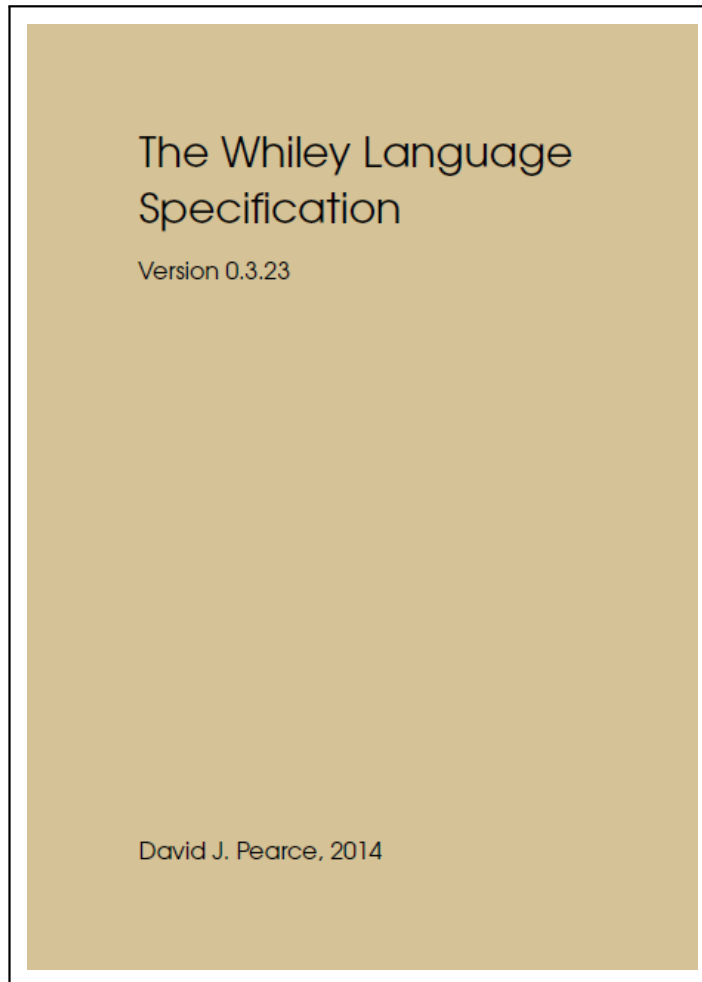
(A/Prof, Victoria University)



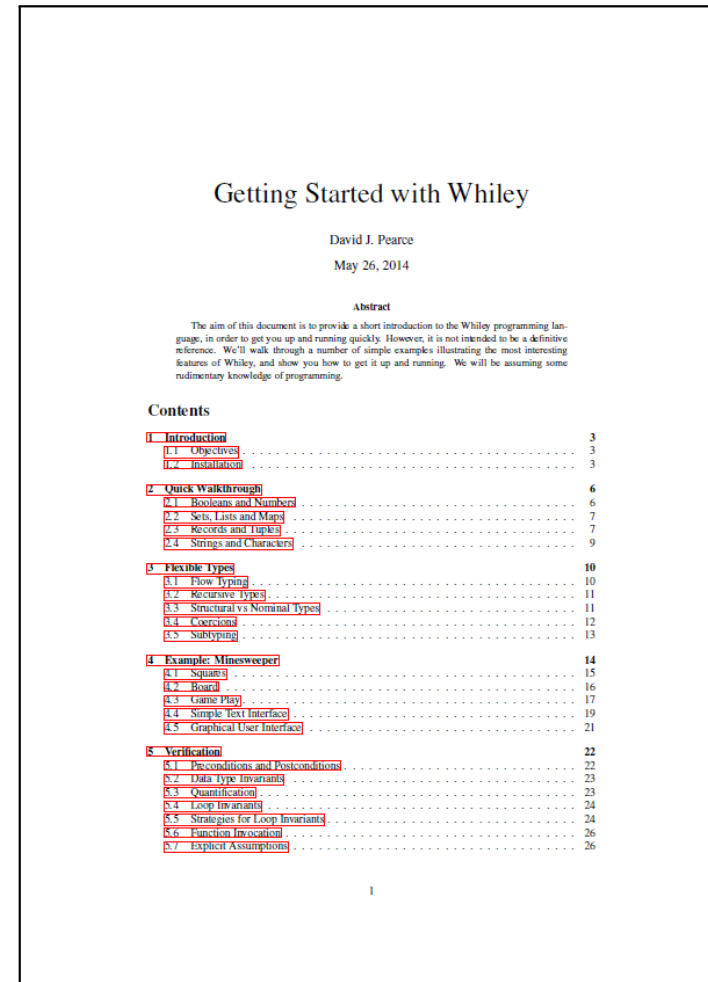
Mark

(A/Prof, University of
Waikato)

Documents (so far)



Whiley Language Specification



Getting Started with Whiley (Tutorial)

`http://whiley.org`

@WhileyDave
`http://github.com/Whiley`

Verification: Constrained Types

```
type N is (T x) where e
```

- Above defines **constrained type**
- **Invariant:** for any variable of type $\boxed{N}$, follows that $\boxed{e}$ always holds
- Constrained types can **simplify** specifications / invariants
- **Example:** *natural numbers*

```
type nat is (int n) where n >= 0
```

Verification: Structural Typing

```
type nat is (int n) where n >= 0
```

```
function cut(int x) → (nat y) :
```

```
  if x >= 0:
```

```
    return x
```

```
  else:
```

```
    return 0
```

- Variable types in Whiley are **ephemeral** ...

... and determined by what is **known** (not what was declared)

Verification: Flow Typing

```
function indexOf(int[] items, int item) → (int|null r)
// If integer value returned, must be index of item
ensures r is int ==> items[r] == item
// No element before integer r matches item
ensures r is int ==> all { k in 0..r | items[k] != item }
// If null returned, no matching item
ensures r is null ==> all { k in 0..|items| | items[k] != item }:
    //
    int i = 0
    //
    while i < |items|
    where i >= 0 && i <= |items|
    where all { j in 0..i | items[j] != item }:
        if items[i] == item:
            return i
        i = i + 1
    //
    return null
```