Automated and Semi-Automated bug finding for Fortran

Dominic Orchard

25th May - RSE Seminar

UNIVERSITY OF CAMBRIDGE

Institute of Computing for Climate Science

University of Kent

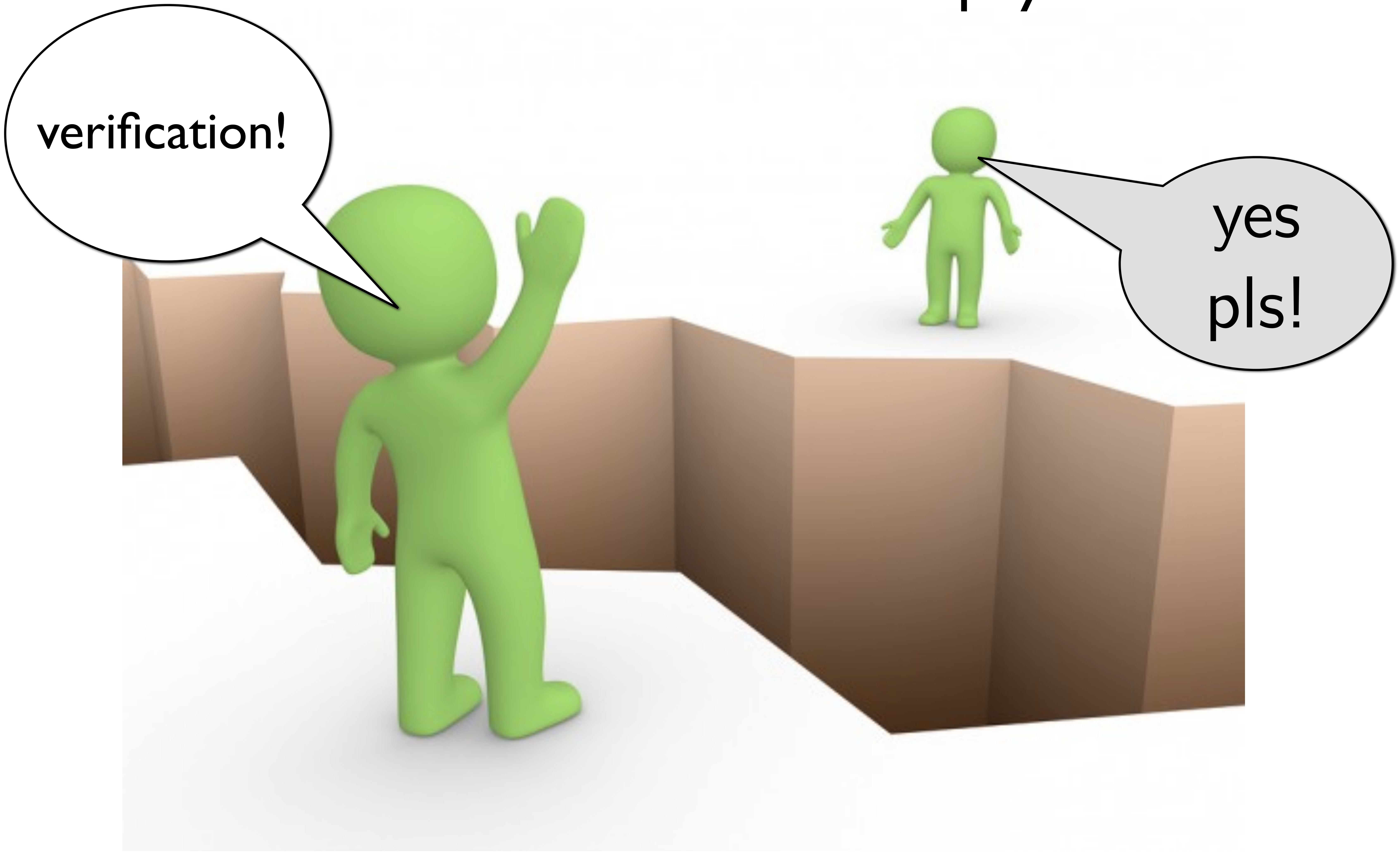
Programming Languages and Systems for Science laboratory

work also with Matthew Danish, Andrew Rice, Mistral Contrastin, Ben Orchard

thanks also to Bloomberg, UKRI Engineering and Physical Sciences Research Council, SCHMIDT FUTURES

2012/13

natural & physical sciences



computer science

natural & physical sciences

computer science

natural & physical sciences

computer science

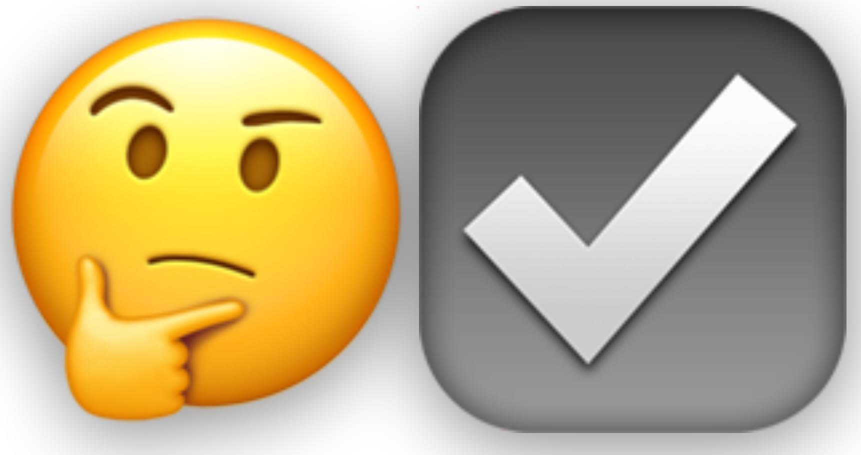
Let's bridge the chasm!

2015-today

https://github.com/camfort/camfort/

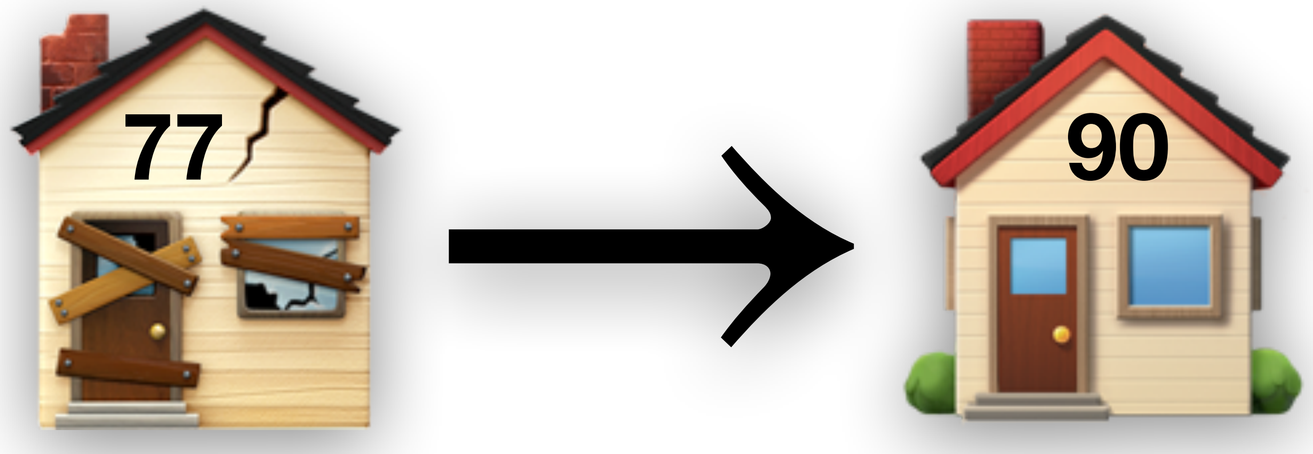
CamFort

Verification



Analysis

Refactoring



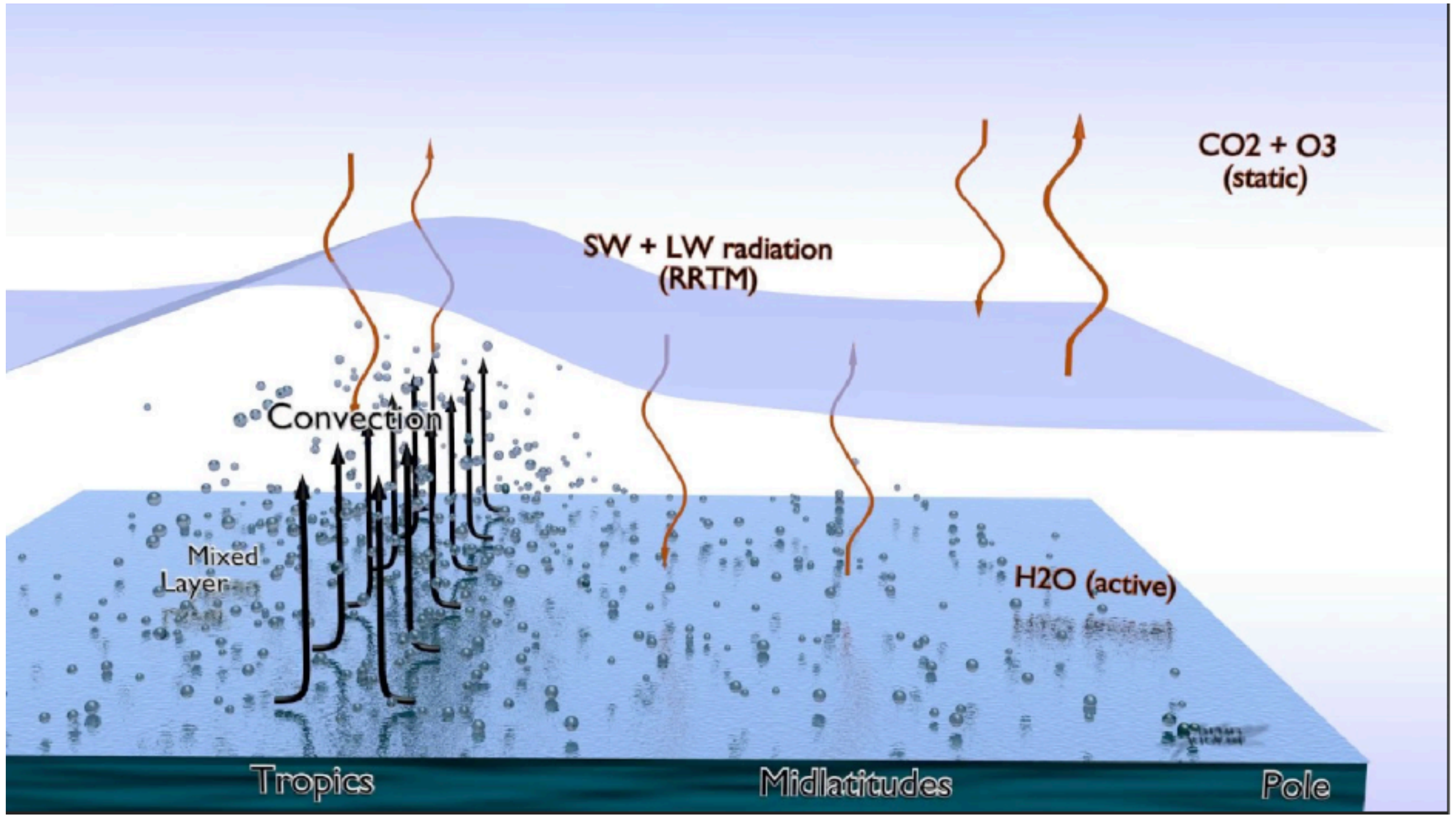
Bloomberg

UKRI Engineering and Physical Sciences Research Council

Met Office Hadley Centre

Demo using MiMA as target

https://github.com/mjucker/MiMA/blob/master/src/atmos_param/cg_drag/cg_drag.f90



`camfort alloc-check`

Memory performance & safety:
All allocated arrays freed, no double free, or use after free

`camfort fp-check`

Numerical stability:
No equality (or inequality) on FP

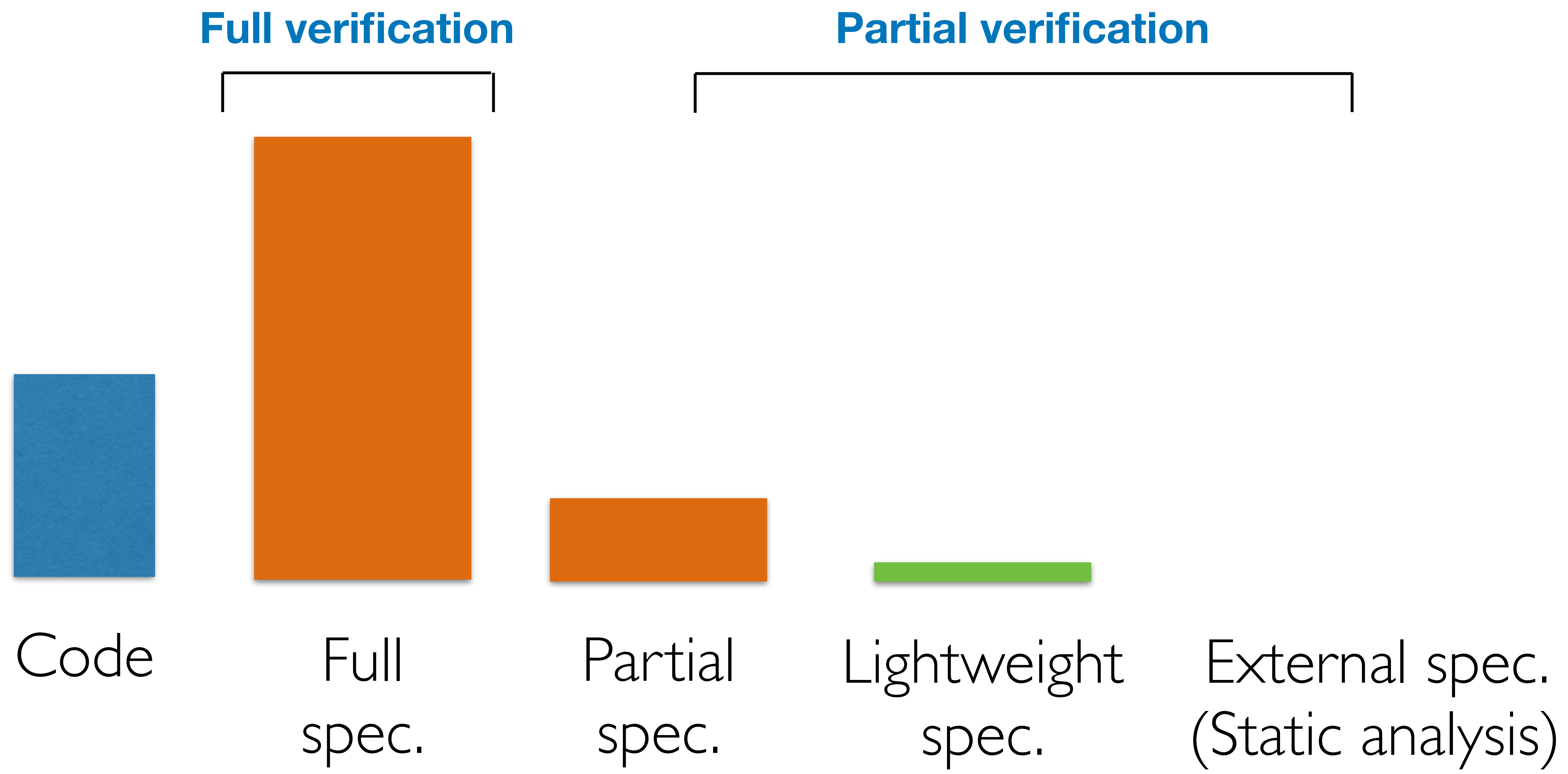
`camfort use-check`

Tidy code:
No equality (or inequality) on FP

`camfort array-check`

Computational performance:
Column-major order traversal

Approaches to verification



Time consuming
Specification completeness?

How to chose which parts?

Focussed on one aspect

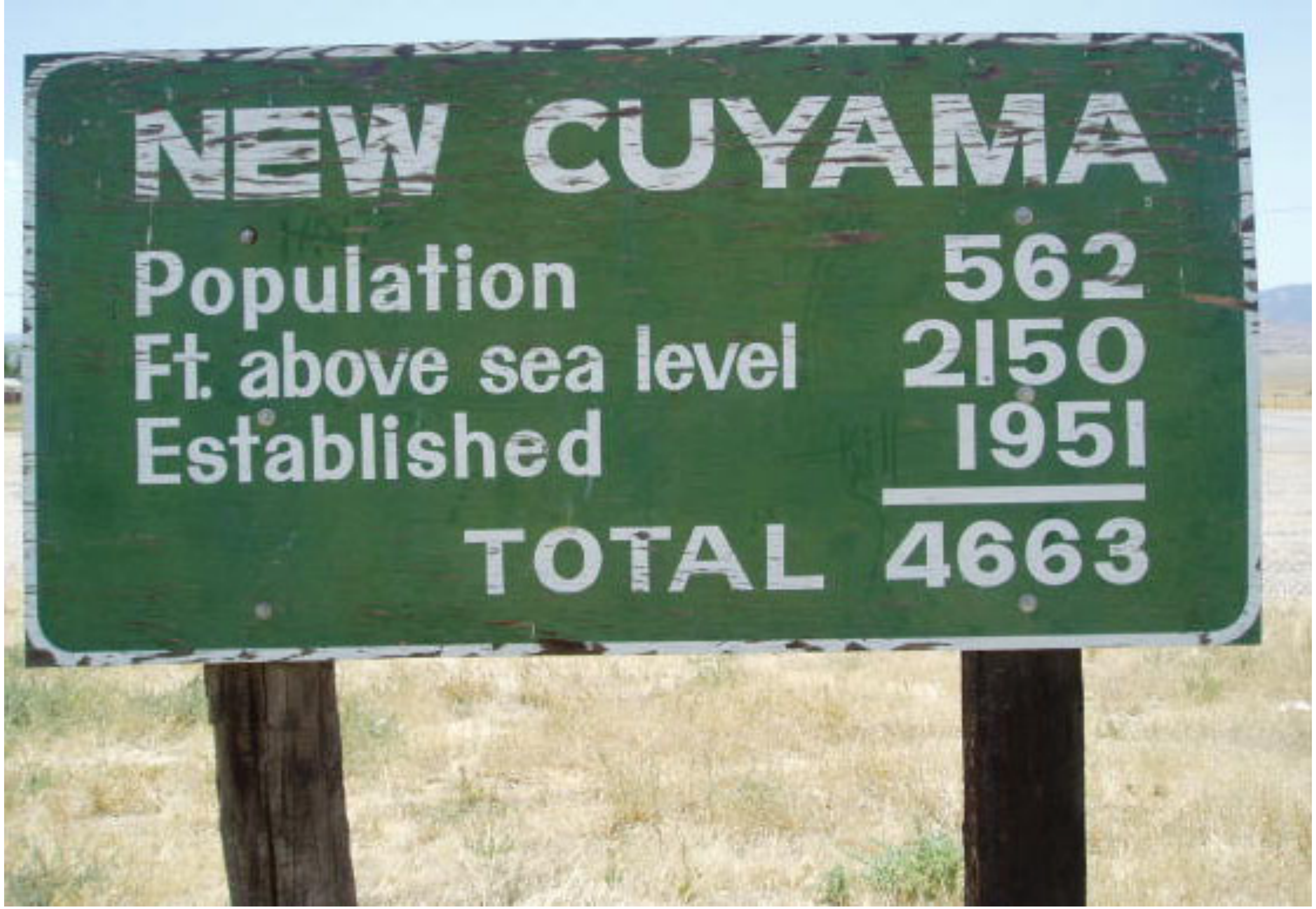


photo from Andrew Kennedy's website
http://research.microsoft.com/en-us/um/people/akenn/units/

Units-of-measure verification

```fortran
1  program energy
2    real :: mass = 3.00, gravity = 9.91, height = 4.20
3    real :: potential_energy
4
5    potential_energy = mass * gravity * height
6  end program energy
```

Suggest

```
$ camfort units-suggest energy1.f90
Suggesting variables to annotate with unit specifications in 'energy1.f90'
…
energy1.f90: 3 variable declarations suggested to be given a
specification:
    energy1.f90 (2:43)    height
    energy1.f90 (2:14)    mass
    energy1.f90 (3:14)    potential_energy
```

Units-of-measure verification

```
program energy
  != unit kg :: mass
  != unit m  :: height
  real :: mass = 3.00, gravity = 9.91, height = 4.20
  != unit kg m**2/s**2 :: potential_energy
  real :: potential_energy

  potential_energy = mass * gravity * height
end program energy
```

Check

```
$ camfort units-check energy1.f90

energy1.f90: Consistent. 4 variables checked.
```

Units-of-measure verification

```
program energy
  != unit kg :: mass
  != unit m  :: height
  real :: mass = 3.00, gravity = 9.91, height = 4.20
  != unit kg m**2/s**2 :: potential_energy
  real :: potential_energy

  potential_energy = mass * gravity * height
end program energy
```

Synthesise

```
$ camfort units-synth energy1.f90 energy1.f90

Synthesising units for energy1.f90
```

Units-of-measure verification

```
program energy
  != unit kg :: mass
  != unit m   :: height
  != unit m/s**2   :: gravity
  real :: mass = 3.00, gravity = 9.91, height = 4.20
  != unit kg m**2/s**2 :: potential_energy
  real :: potential_energy

  potential_energy = mass * gravity * height
end program energy
```

Synthesise

```
$ camfort units-synth energy1.f90 energy1.f90

Synthesising units for energy1.f90
```

Check

Does it do what I think it does?

Infer

What does it do?

Synthesise

Capture what it does for documentation & future-proofing

Suggest

Where should I add a specification to get the most information?

Navier-Stokes code, cf. 2007

```fortran
do i = 1, (imax-1)
  do j = 1, jmax
    ! only if both adjacent cells are fluid cells */
    if (toLogical(iand(flag(i,j), C_F)) .and.                              &
        toLogical(iand(flag(i+1,j), C_F))) then

      du2dx = ((u(i,j)+u(i+1,j))*(u(i,j)+u(i+1,j))+                      &
              gamma*abs(u(i,j)+u(i+1,j))*(u(i,j)-u(i+1,j))-               &
              (u(i-1,j)+u(i,j))*(u(i-1,j)+u(i,j))-                        &
              gamma*abs(u(i-1,j)+u(i,j))*(u(i-1,j)-u(i,j)))               &
              /(4.0*delx)
      duvdy = ((v(i,j)+v(i+1,j))*(u(i,j)+u(i,j+1))+                      &
              gamma*abs(v(i,j)+v(i+1,j))*(u(i,j)-u(i,j+1))-               &
              (v(i,j-1)+v(i+1,j-1))*(u(i,j-1)+u(i,j))-                    &
              gamma*abs(v(i,j-1)+v(i+1,j-1))*(u(i,j-1)-u(i,j)))           &
              /(4.0*dely)
      laplu = (u(i+1,j)-2.0*u(i,j)+u(i-1,j))/delx/delx+                   &
              (u(i,j+1)-2.0*u(i,j)+u(i,j-1))/dely/dely

      f(i,j) = u(i,j) + del_t*(laplu/Re-du2dx-duvdy)
    else
      f(i,j) = u(i,j)
    end if
  end do
end do
```

Correct?

Study corpus (v1)

	Package
climate	UM
economics	E3ME
bio/climate	Hybrid4
chem/climate	GEOS-Chem
fluids	Navier
physics	CP
library	BLAS
library	ARPACK-NG
geodynamics	SPECFEM3D
library	MUDPACK
seismology	Cliffs

11 packages

~1.1 million physical loc

FORTRAN 77 and Fortran 90

Analysis of patterns in corpus

Paper has more fine-grained analysis/data

- Array computations are common in science (133k / 1.1m)
- Mostly regular access patterns (72.12% of all array comps.)
- Many are *stencils* (55.86% of all array comps.)

(6.28% are "reductions")

Numerical analysis literature

Design of specification language for array access shape

```
do i = 1, (n-1)
      b(i) = a(i-1) - 2*a(i) + a(i+1)
end do
```

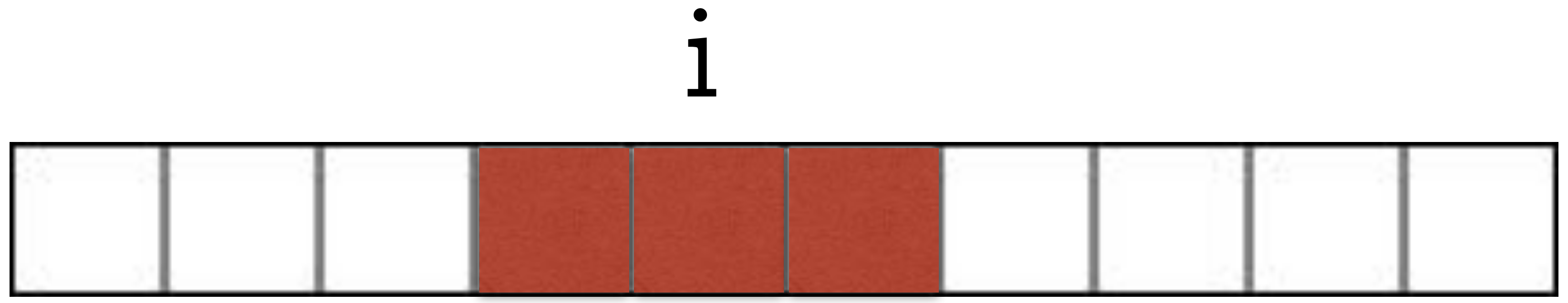
Spatial specifications as comments

```
do i = 1, (n-1)
      != stencil centered(dim=1, depth=1) :: a
      b(i) = a(i-1) - 2*a(i) + a(i+1)
end do
```

Spatial specification language

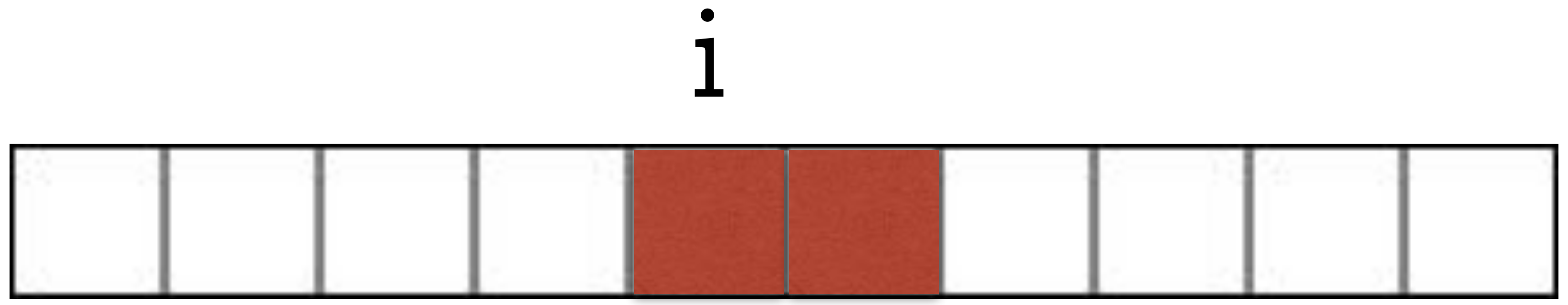
`centered(dim=1,depth=1)`

a(i-1), a(i), a(i+1)



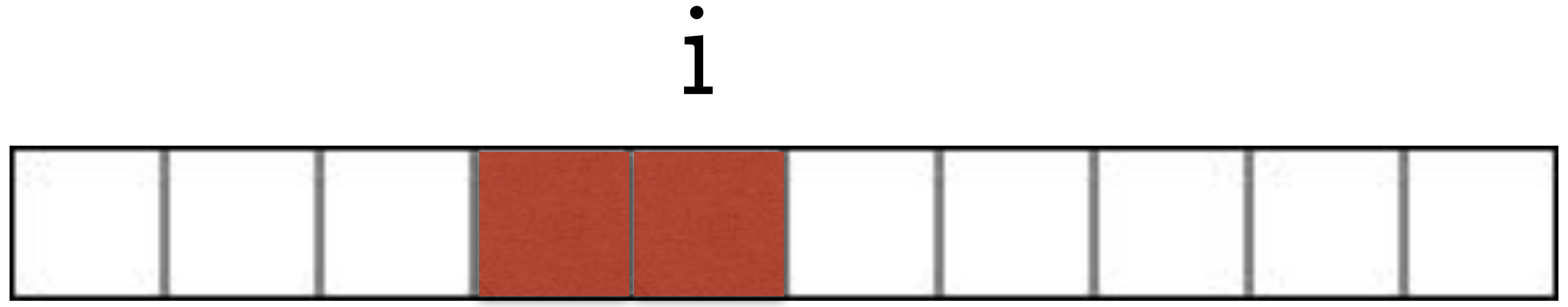
`forward(dim=1,depth=1)`

a(i), a(i+1)



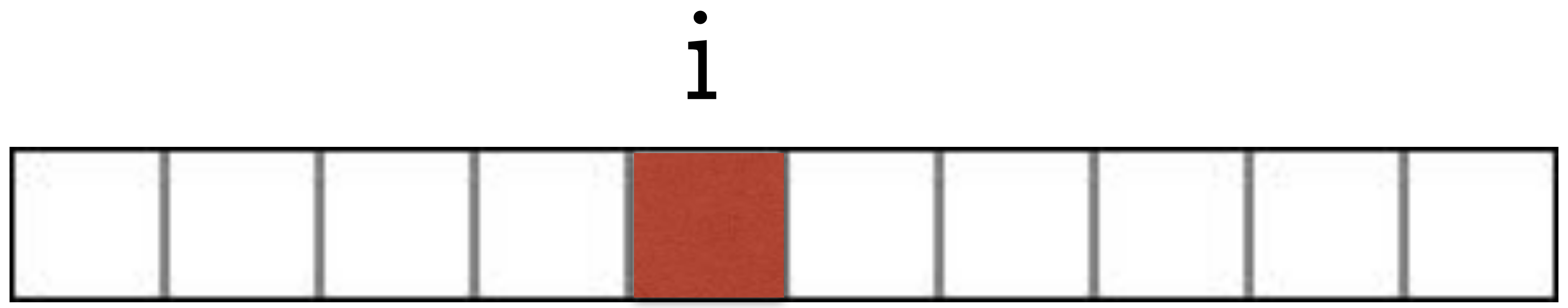
`backward(dim=1,depth=1)`

a(i-1), a(i)

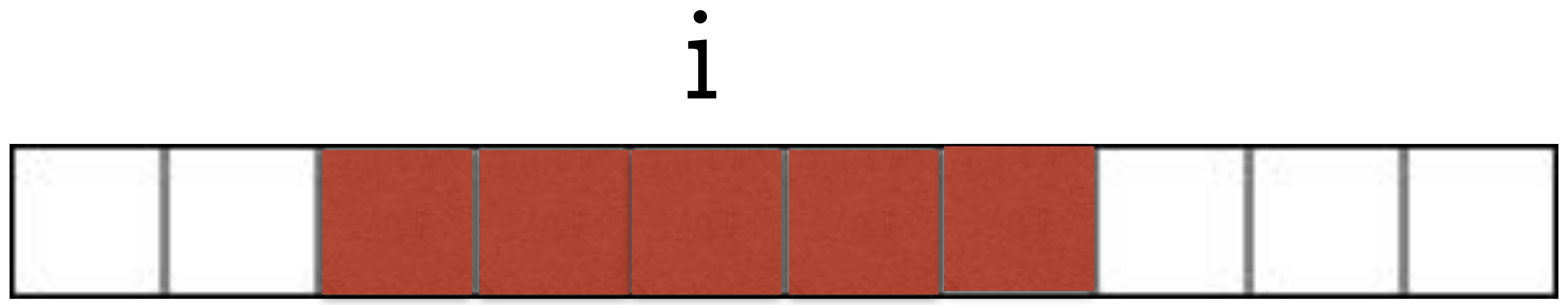


`pointed(dim=1)`

a(i)

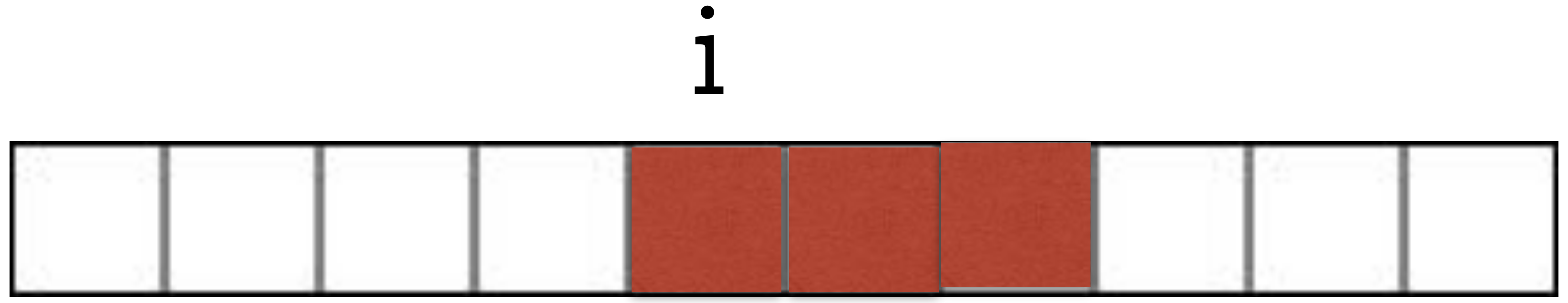


Spatial specification language



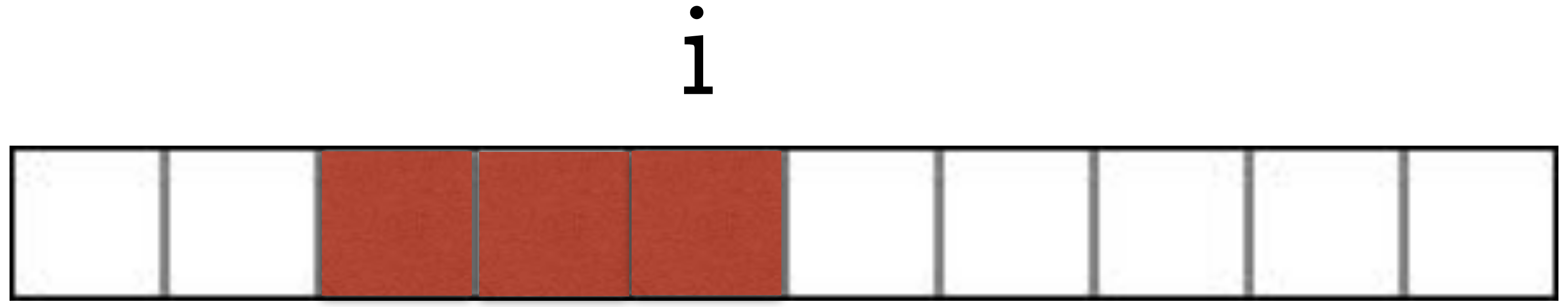
`centered(dim=1,depth=2)`

a(i-1), a(i-2), a(i), a(i+1), a(i+2)



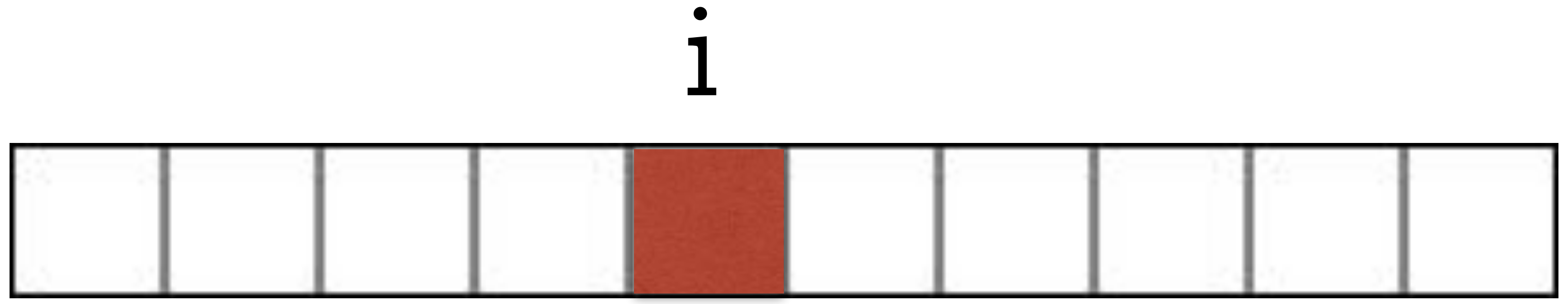
`forward(dim=1,depth=2)`

a(i), a(i+1), a(i+2)



`backward(dim=1,depth=2)`

a(i-2), a(i-1), a(i)

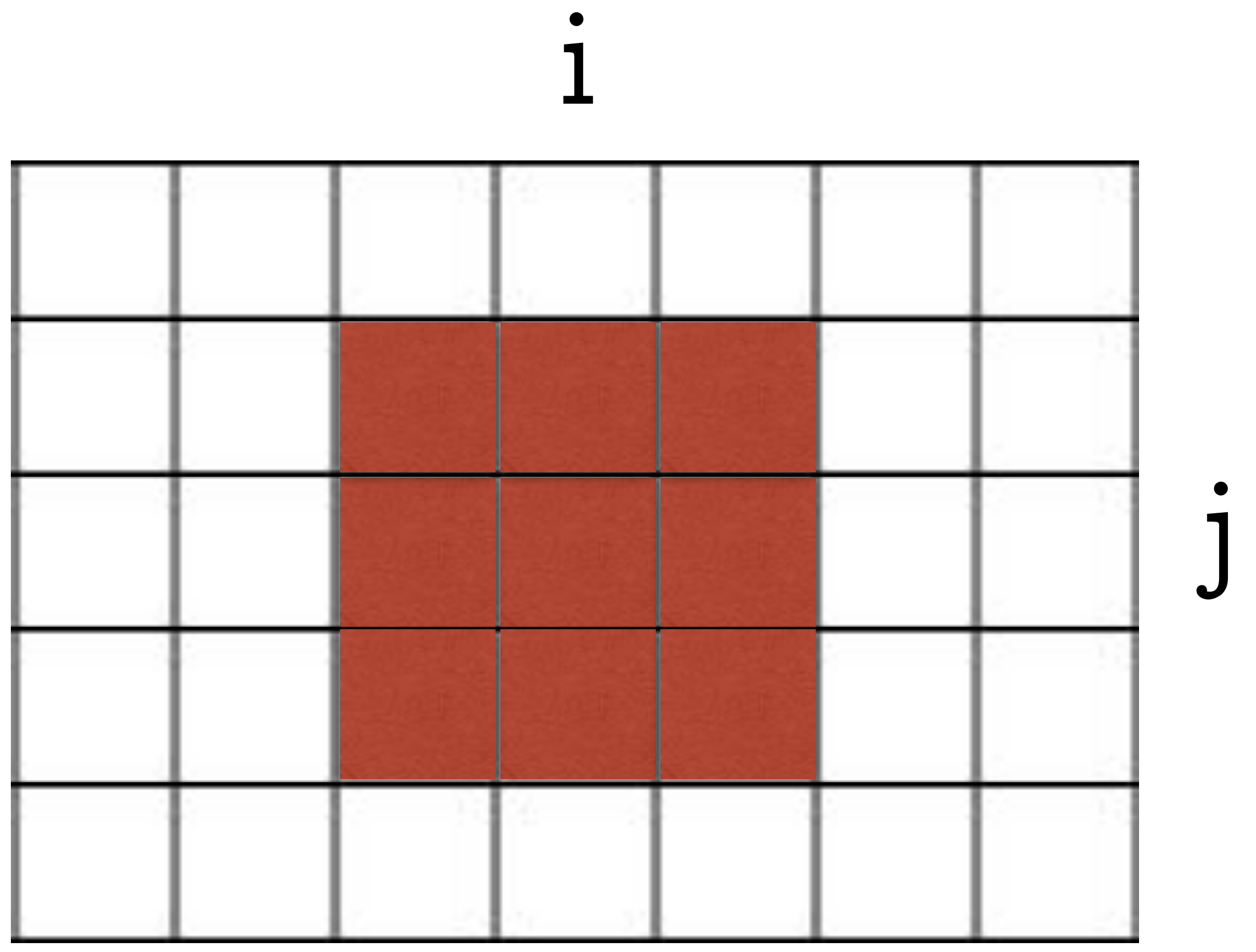


`pointed(dim=1)`

a(i)

Combining specifications with *

e.g. `centered(dim=1,depth=1)` * `centered(dim=2,depth=1)`



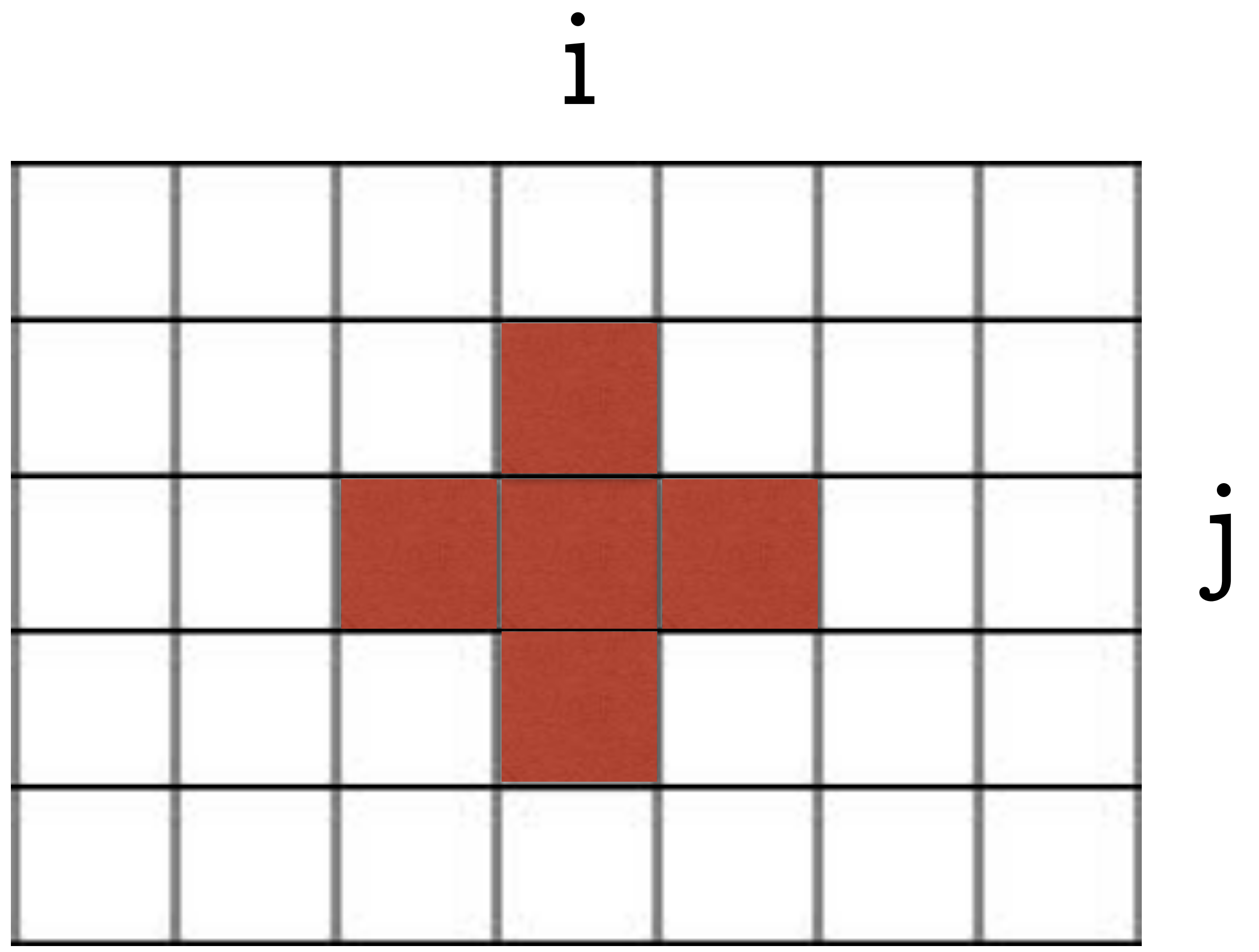
Corresponds to

```
  a(i-1, j-1) + a(i-1, j) + a(i-1, j+1)
+ a(i  , j-1) + a(i  , j) + a(i  , j+1)
+ a(i+1, j-1) + a(i+1, j) + a(i+1, j+1)
```

Combining specifications with +

e.g.

```
  centered(dim=1,depth=1)*pointed(dim=2)
+ centered(dim=2,depth=1)*pointed(dim=1)
```



"Five point stencil"

Corresponds to

```
                  a(i-1, j)
+ a(i, j-1) + a(i  , j) + a(i, j+1)
+                 a(i+1, j)
```

Analysis

Java

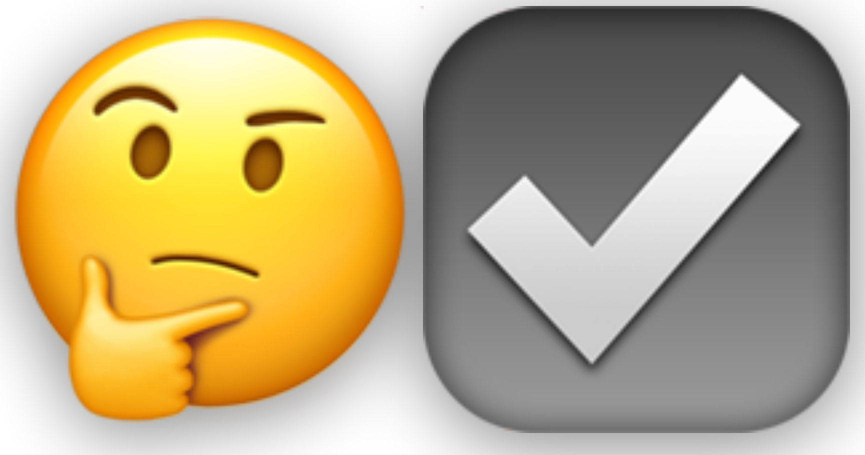
See general tools e.g.

sonar

coverity

Meta Infer

Verification



Interested in ideas for future tools….

dorchard.github.io

Thanks!

camfort.github.io