granule-project.github.io

The Granule language:
Fine-grained program verification via types

Vilem-Benjamin Liepelt[1] Dominic Orchard[1] Harley Eades III[2] Preston Keel[2]

Q: How do we ensure a program does not leak sensitive data?

Granule: track & enforce security levels of data via the type system of a programming language

Granule allows domain-specific data-flow properties to be automatically checked at compile time

Describe data representation with **confidentiality types**

A value of type: `A [L]`

is a value of type ‘A’ that can be *accessed* at a security level ‘L’

e.g. a private integer has type `Int [Private]`

e.g.

```
data Patient where

  Patient :
      Int    [Private]   -- Patient id
   -> String [Private]   -- Patient name
   -> Int    [Public]    -- Patient age
   -> Patient
```

Programmers then write standard functional programs, but with **confidentiality specifications**

e.g. `meanAge : List Patient -> Int [Public]` ✔

Granule type checker rejects anything that leaks private values to a public context:

e.g. `allNames : List Patient -> String [Public]` ✘

More detail for the interested

Granule can track various different data-flow properties of computation in a similar way.

For example, it can track how many times values are used via a type:

`A [n]`

Meaning: a value of type ‘A‘ which can be used exactly ’n‘ times.

This allows tighter specifications of programs, reducing the number of possible implementations, and also provides resource reasoning.
For example:

to transform a list of A elements of length n into a list of B elements of length n requires a function which maps A elements to B elements which can be used exactly n times:

```
map : forall (a,b : Type, n : Nat)
    . (a -> b) [n]
   -> Vec n a
   -> Vec n b
```

This gives a guarantee on running time, and cuts out buggy implementations.

[1] University of Kent [2] Augusta University