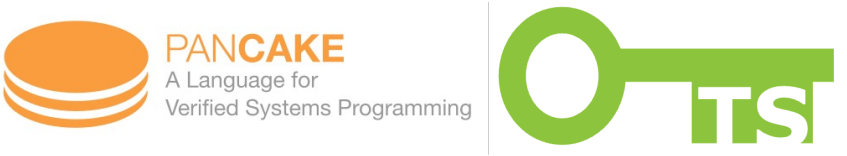
School of Computer Science & Engineering

Trustworthy Systems Group

Pancake at TS - Updates

Miki Tanaka

miki.tanaka@unsw.edu.au



PANCAKE

A Language for
Verified Systems Programming

TS

l4v (2009):
- Verified microkernel
- TCB still large for actual use
- Next: verify OS components

Cogent (2016~2021 in TS):
- Linear-typed functional language
- Proof-producing compilation to C
- No real uptake

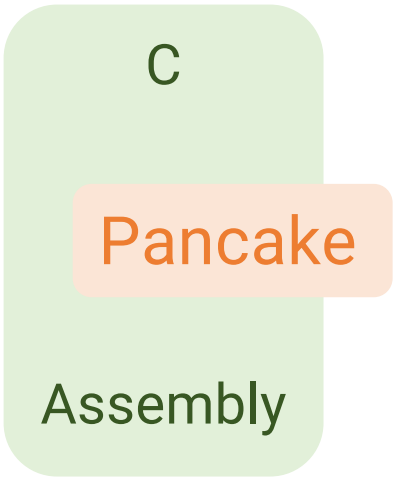
Early Pancake (around 2020):
- Magnus Myreen at Chalmers U. of Technology
- Experimental “thin” language out of CakeML

Pancake Development at TS
(+ ANU, Chalmers, GU) 2022~

It’s REAL! We are using it!!

Pancake: a quick tour

Philosophy



Similar feel to C but "Everything is a word"

- Assembly level memory abstraction
- No types but has "shapes"
- Simple semantics

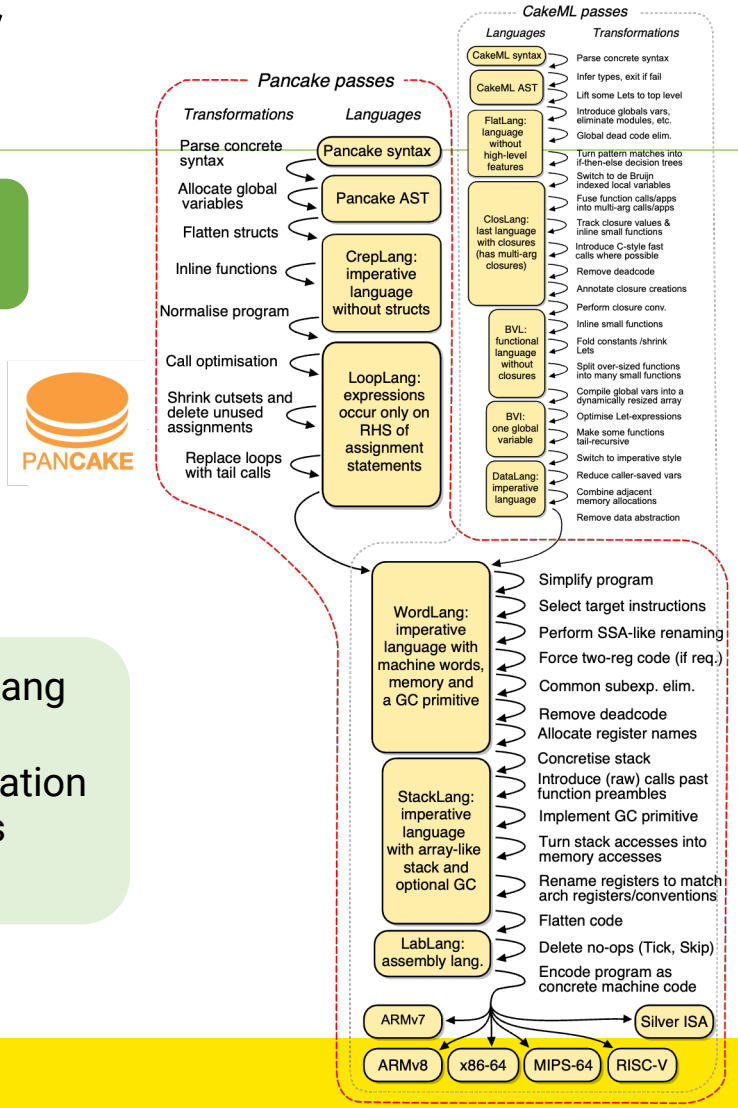
The programmers enforce their intended meaning of a word, not the compiler

```
export fun handle_irq() {
  var got_char = 0;
  got_char = getchar();

  if (got_char == -1) {
    return -1;
  } else {
    while true {
        var buffer_addr = @base + 544;
        var dequeue_ret = 0;
        // 0: dequeue_avail rx
        dequeue_ret = serial_dequeue_avail(0, buffer_
        if (dequeue_ret != 0) {
          // dequeue_avail ring is empty
          return -1;
        }
        var enqueue_ret = 0;
        // 0: enqueue_used rx
        enqueue_ret = serial_enqueue_used(0, got_char
        if (enqueue_ret != 0) {
          return -1;
        }
      }
  }
}
```

Verified compiler

Semantic preservation proved from source to binary



- compiles into CakeML WordLang
- implemented in HOL4
- many ILs, incremental compilation
- functional big-step semantics
- bootstrapped via CakeML

Compile with
"cake --pancake < file.pk"

More on syntax : shapes

Pancake data = one or more words

Shapes

- 1 represents one word: `var 1 x = 3;`
- 2 = {1,1}, …, N = {1, …, 1} (N times)
- can be nested: {1, 1, {1, 1}}

```
typedef struct {
    uint64_t tail;
    uint64_t head;
    uint64_t capacity;
    struct descriptor *descr;
} hw_ring_t;
```

This will have shape 4 as Pancake.

Initialisation: `var 4 ring = <0, 0, 128, descr>;`

But no field update (need to reassign as the whole):

`ring = <ring.0, ring.1, 256, ring.3>;`

(0-indexed)

More on syntax : shapes

```
void add_one(int *x) {
  *x = *x + 1;
}
```

```
fun add_one (1 x) {
  var val = lds 1 x;
  st x, (val + 1);
  return 0;
}
```

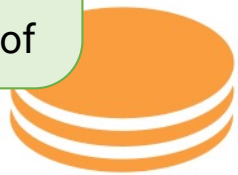
NEW: named structs

```
struct my_struct {
    2 tuple,
    1 value
  }
```

```
var my_struct x = my_struct <tuple = <0,1>,
                             value = 2>;
```

field access by x.value (no field update yet)

Development history



PANCAKE
A Language for
Verified Systems Programming

2023:
- Concrete syntax / parser
- Function declaration
- Multiplication
- Full compiler correctness proof

2024:
- Shared memory load/store (MMIO)
- Multiple entry points
- Logical and/or, misc other ops
- 32-bit shared memory load/store
- Interaction tree semantics v1
- Viper-frontend for Pancake

2025:
- 32-bit internal load/store
- Static checks / shape checks
- 16-bit shared memory load/store
- Global variables
- Interaction tree semantics v2
- More drivers

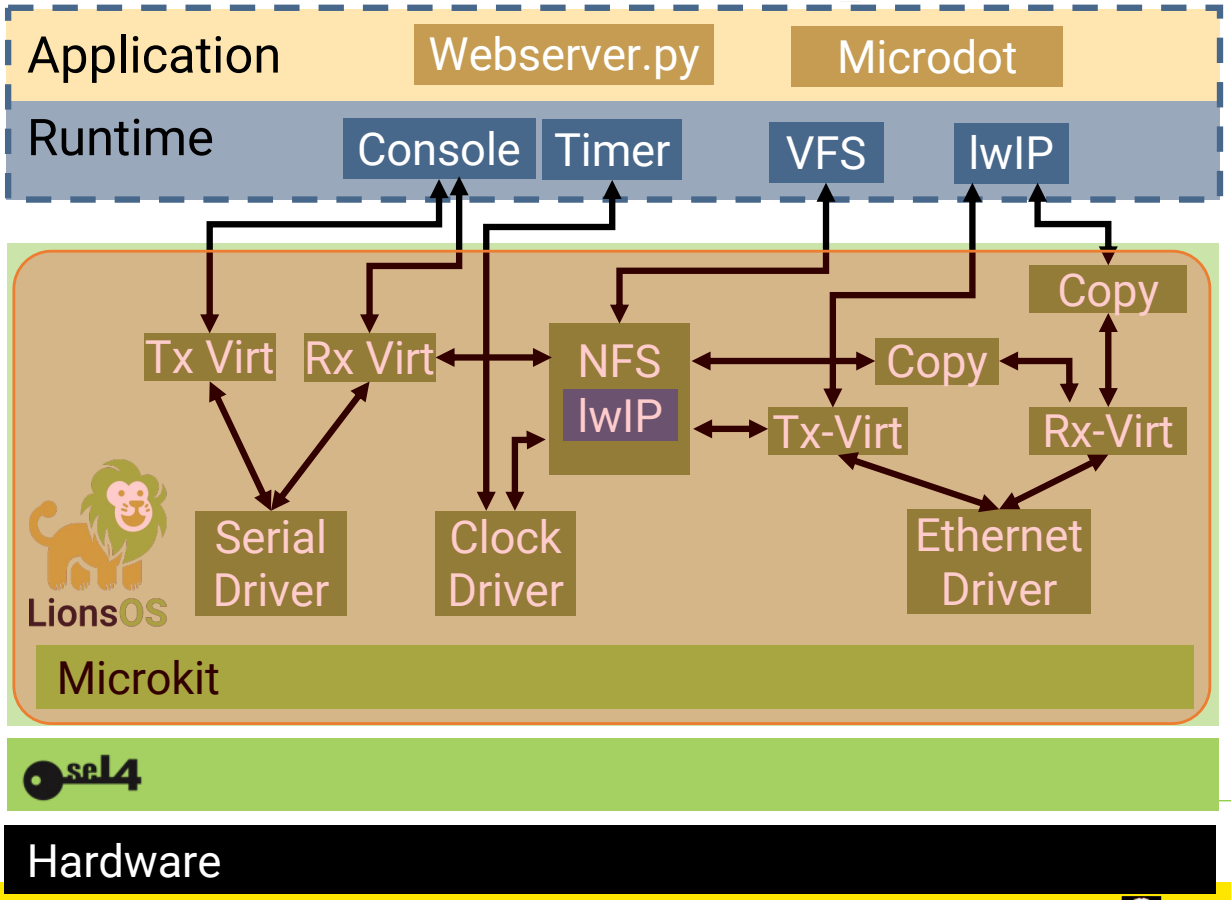
2026:
- Function inlining
- Named structs
- Decompiler into interaction trees
- Verified libmicrokit in Pancake
- Verified ethernet driver in Pancake
- More optimisations, memory barriers

Pancake and LionsOS

Implemented in Pancake:
- sDDF device drivers:
  serial, timer, ethernet
- libmicrokit (RISC-V)
- libsel4 binding for Microkit

Verification:
- Ethernet driver (imx)
- Virtualiser (in progress)
- libmicrokit (API, handler loop)

Next:
- implement and verify more drivers (I2C, etc.)
- composition proofs
- arch ports (libmicrokit, etc.)
- documentation, upstreaming

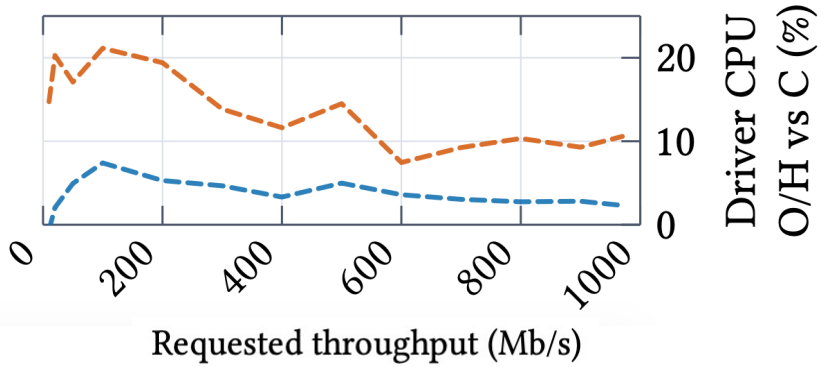


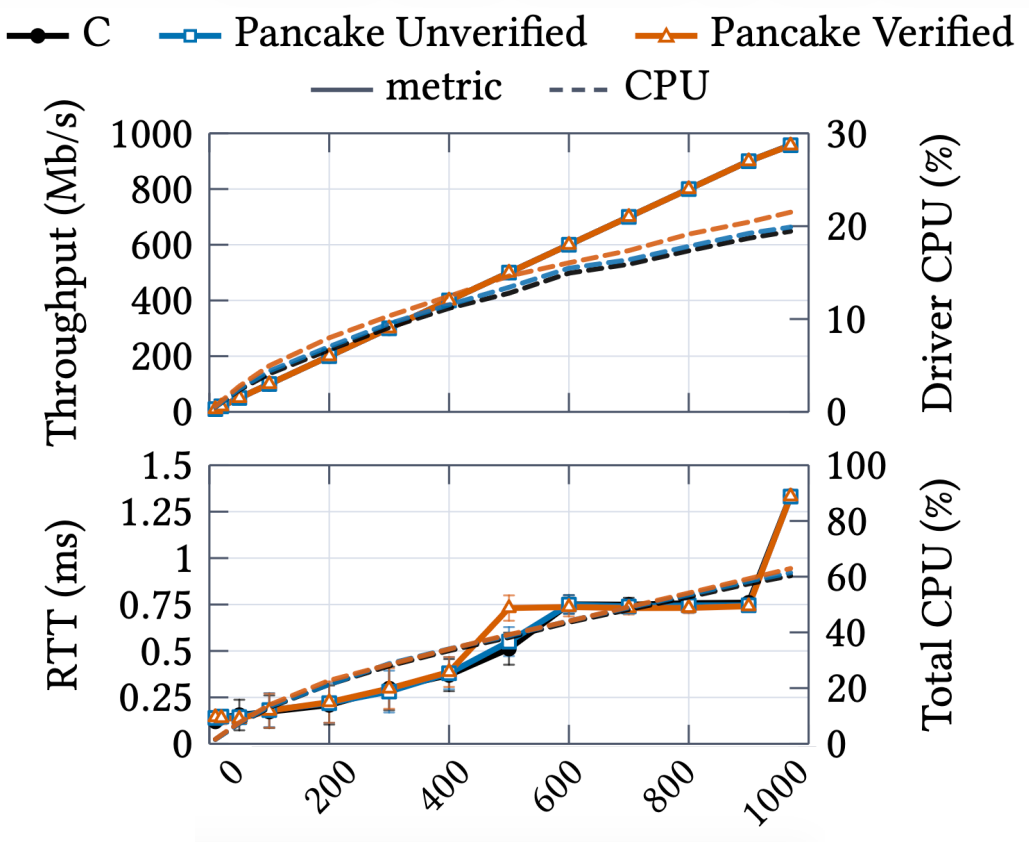
Performance

Significant improvement so far:

- reducing ffi calls
- function inlining
- current: global variable improvement

(Ethernet driver data)





Pancake: what's next

Extension & optimisation plans/ideas

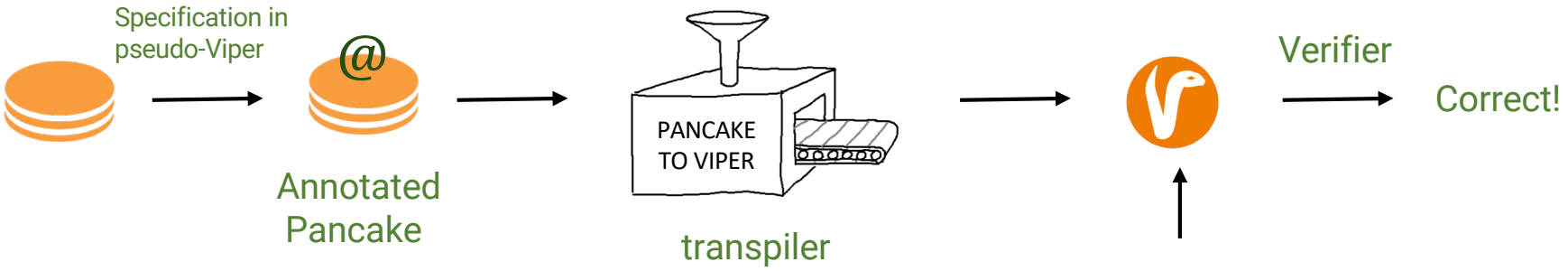
Language extensions:
- struct field update
- sum types
- division
- exceptions

Optimisations, others:
- global variable caching
- memcpy
- loop unrolling
- re-entry point proof

Usability features:
- multi-file compilation
- debugging tools
- better error messages

** CakeML: inline Pancake instead of assembly*

Automated verification



Verified:
- Ethernet driver (imx) [PLOS'26]
- network virtualiser (Tx only)
- libmicrokit & libsel4 binding

More targets / re-verification:
- virtualiser Rx, other drivers
- policy swapping
- arch ports (incl. libmicrokit)

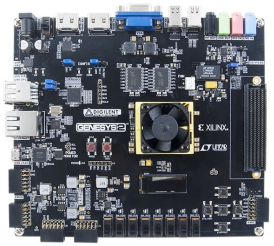
Misc others:
- transpiler verification (reimplementation, Viper semantics)
- annotation automation/abstraction
- backend improvements (hybrid?)
- bitvector optimisation

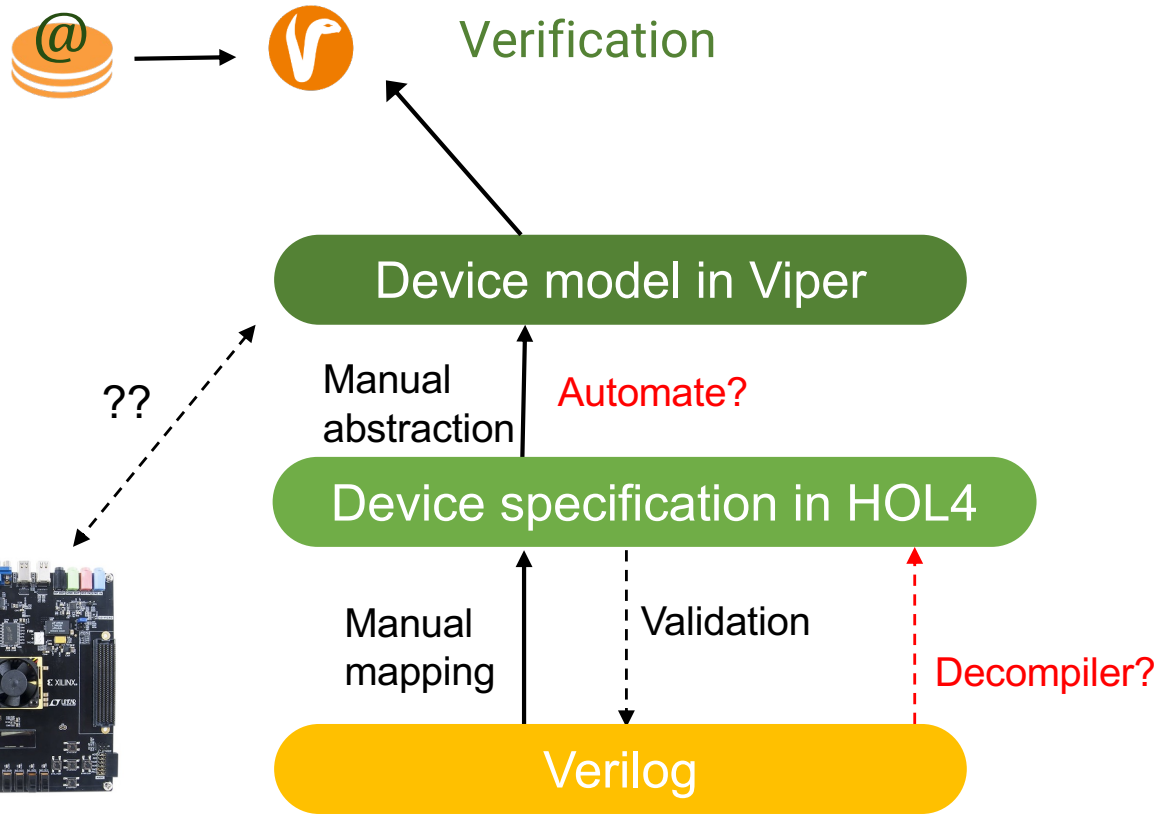
End-to-end verification

Verilog model in HOL4:
By Andreas Loow, with HOL-to-Verilog translator

Formalise open-source devices:
I2C done, SPI in progress [PLOS'25]

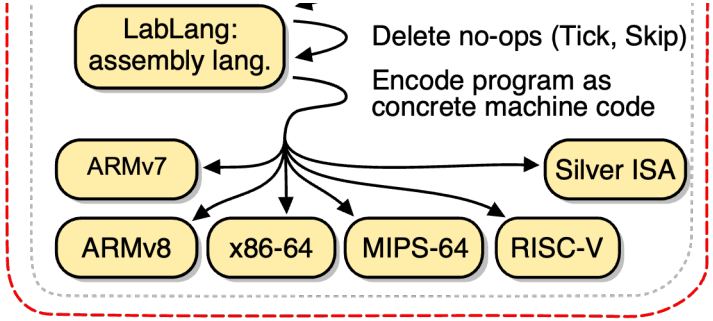
Next:
- Ethernet controller
- decompiler
- automated abstraction
- improve semantics

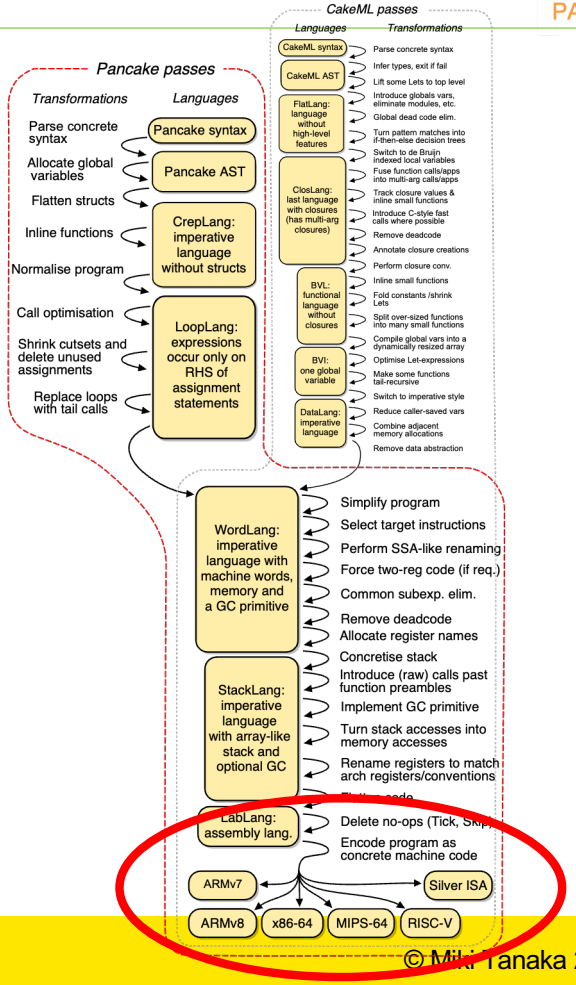




Binary validation

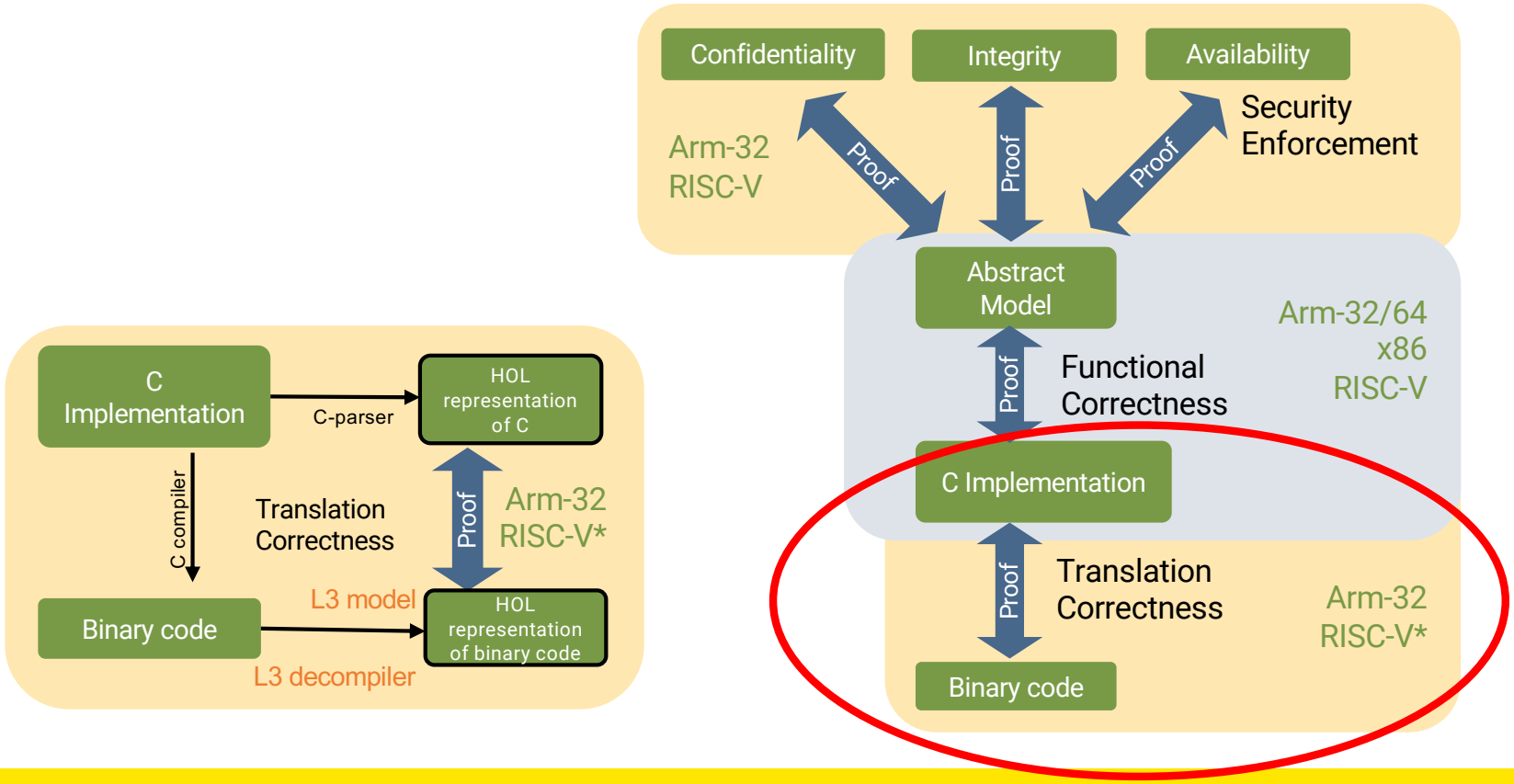
Formal ISA models:
- L3 (Cambridge, HOL4)
- Sail (Cambridge, misc formalisms)



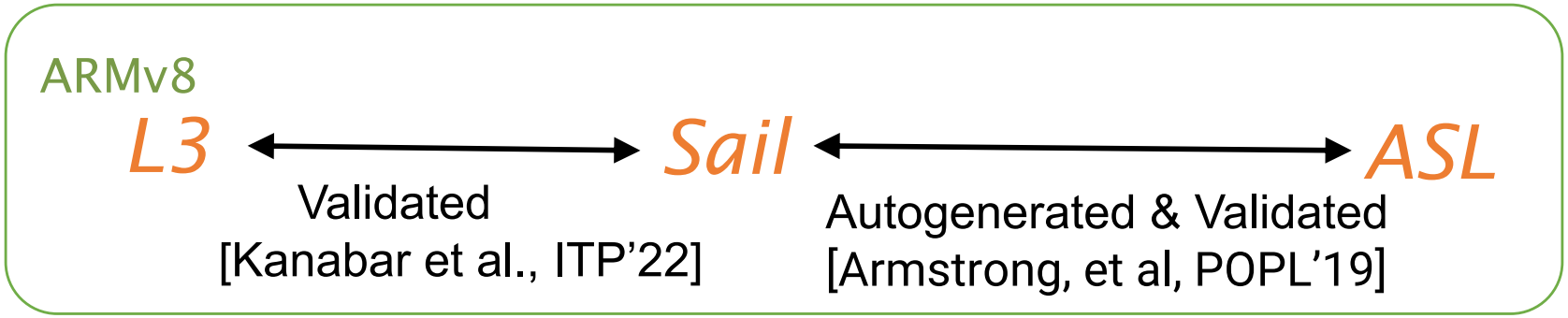


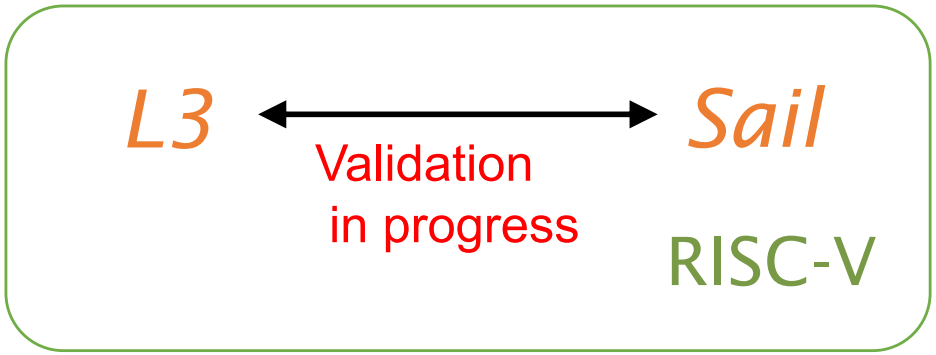
LabLang
↓
asmLang
↓
L3 models
(ARMv8, RISCV-V, x86-64, etc.)
↓ (L3 encoder)
Binary code

Binary verification in l4v



Validation of L3 RISC-V model





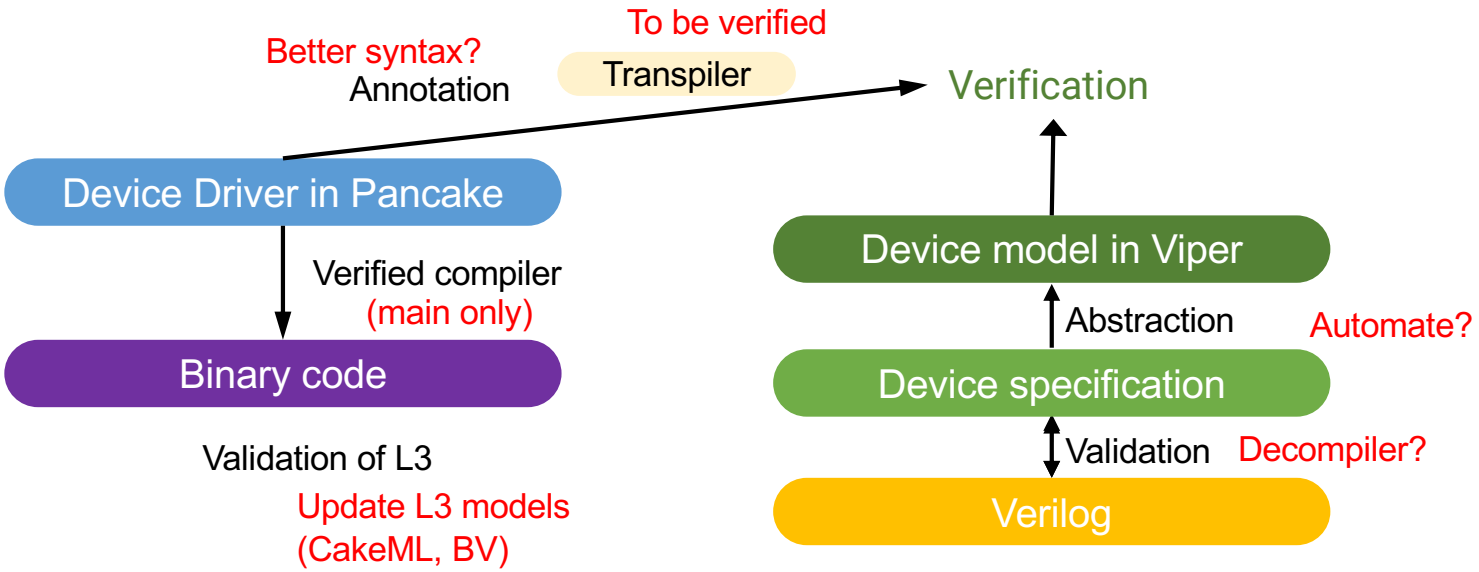
Next:
- complete RISC-V validation
- L3 model update (CakeML, BV)
- upstream HOL4 proofs for Sail

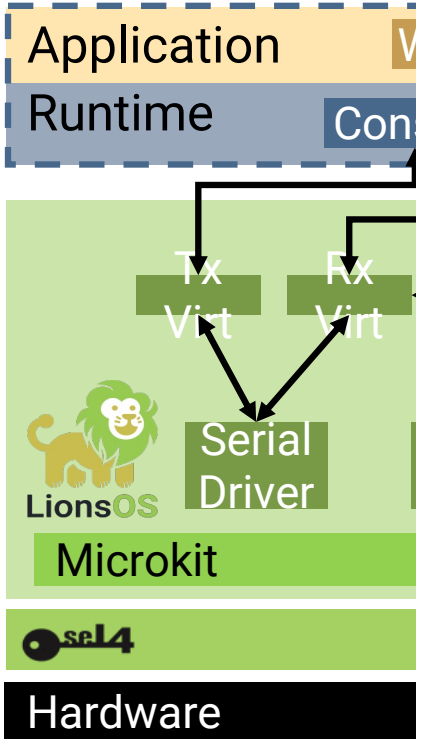
Conclusion

Perspectives

Further language extensions, compiler optimisations, semantics work for Pancake

Verification framework improvements, application to more components





Pancake has grown into a REAL usable language

for verified systems programming

Pancake has grown into a REAL usable language

for verified systems programming (in TS)

Pancake tutorial in seL4 Summit 2027?

in Sydney??

Thank you!