Machine readable specifications at scale

Alastair Reid

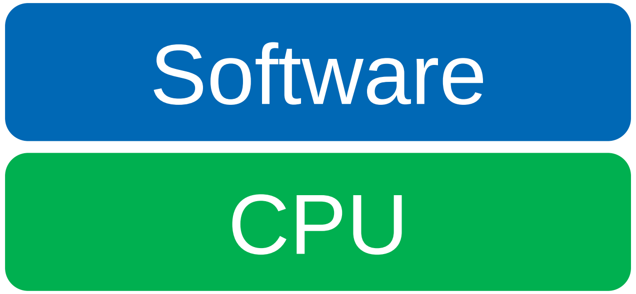
Strategic CAD Lab, Intel Labs

ZISC seminar, ETH Zurich, 2nd June 2022

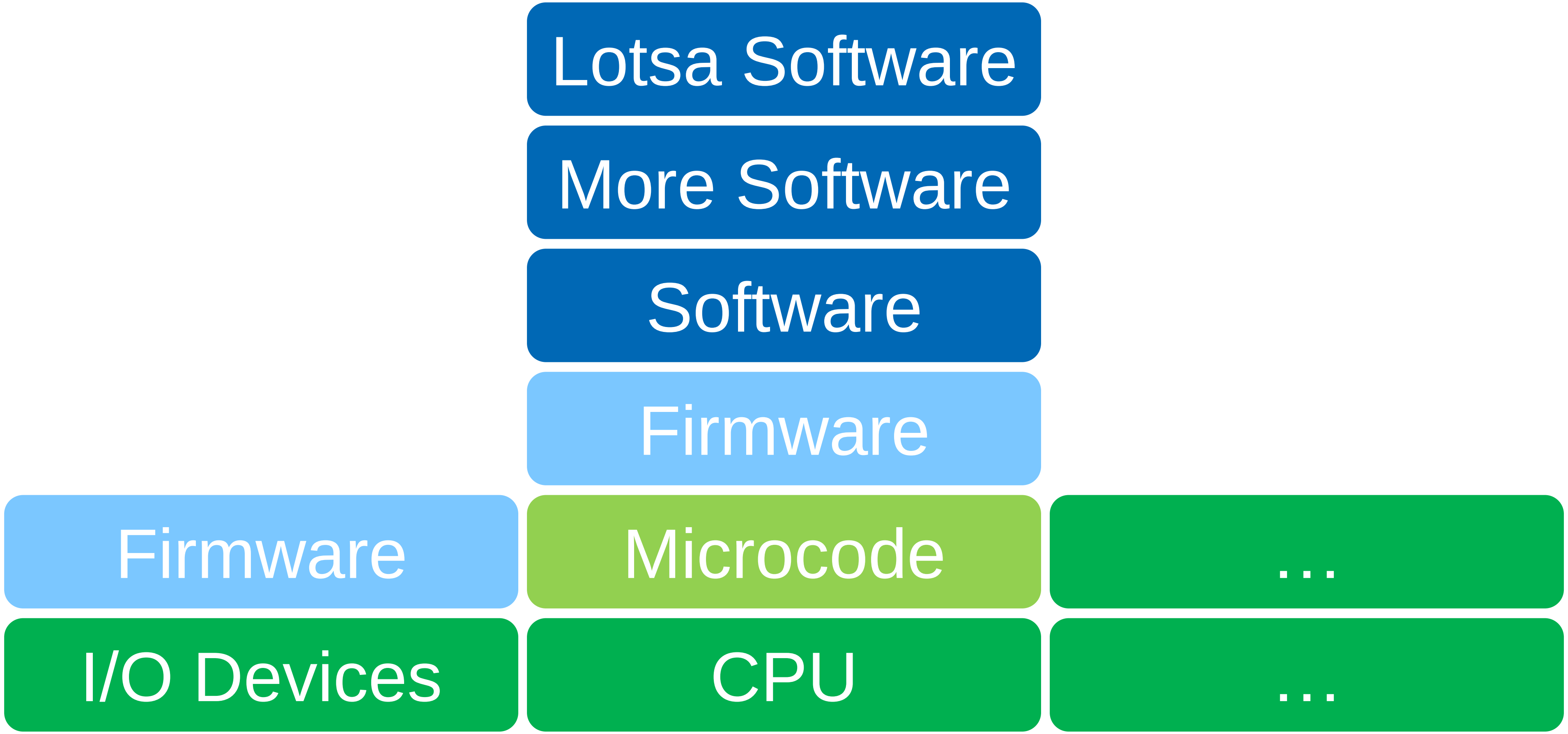
intel

See also:
https://alastairreid.github.io/mrs-at-scale/
https://alastairreid.github.io/uses-for-isa-specs/

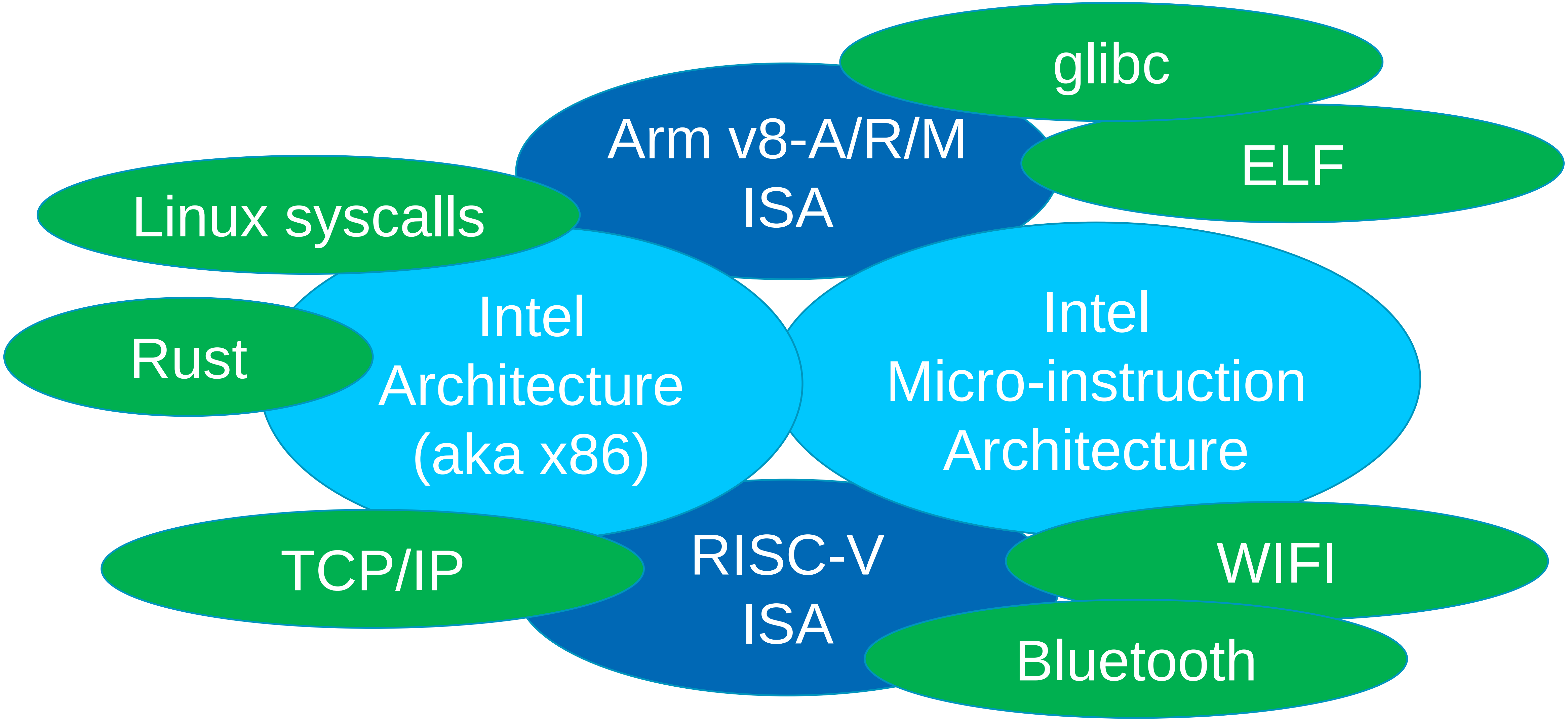
The hardware-software interface



So many critical interfaces …



The examples that I am thinking of



Machine readable specifications – part 1

Easy to parse

- CSV, XML, json, lex+yacc

Has a clear meaning

- “Pending is not set to 1”
- vs “Pending is unchanged”
- vs “Unchanged(Pending)”

Machine readable specifications – part 2

Usage

- Execute (Golden reference model)
- Enumerate legal behaviours
- Checker (Test oracle)

Automation

- Tool generation
- Verification / Testing

Machine readable specifications – part 3

Must be

great documentation

for **humans**

… at scale

Large, complex systems

Large implementations

Decades of history

Multiple implementation teams

Multiple companies

Thousands of spec users

Multiple universities / companies

Many groups

Many people per group

Diverse uses

Diverse uses

Documentation

Simulation

Testing

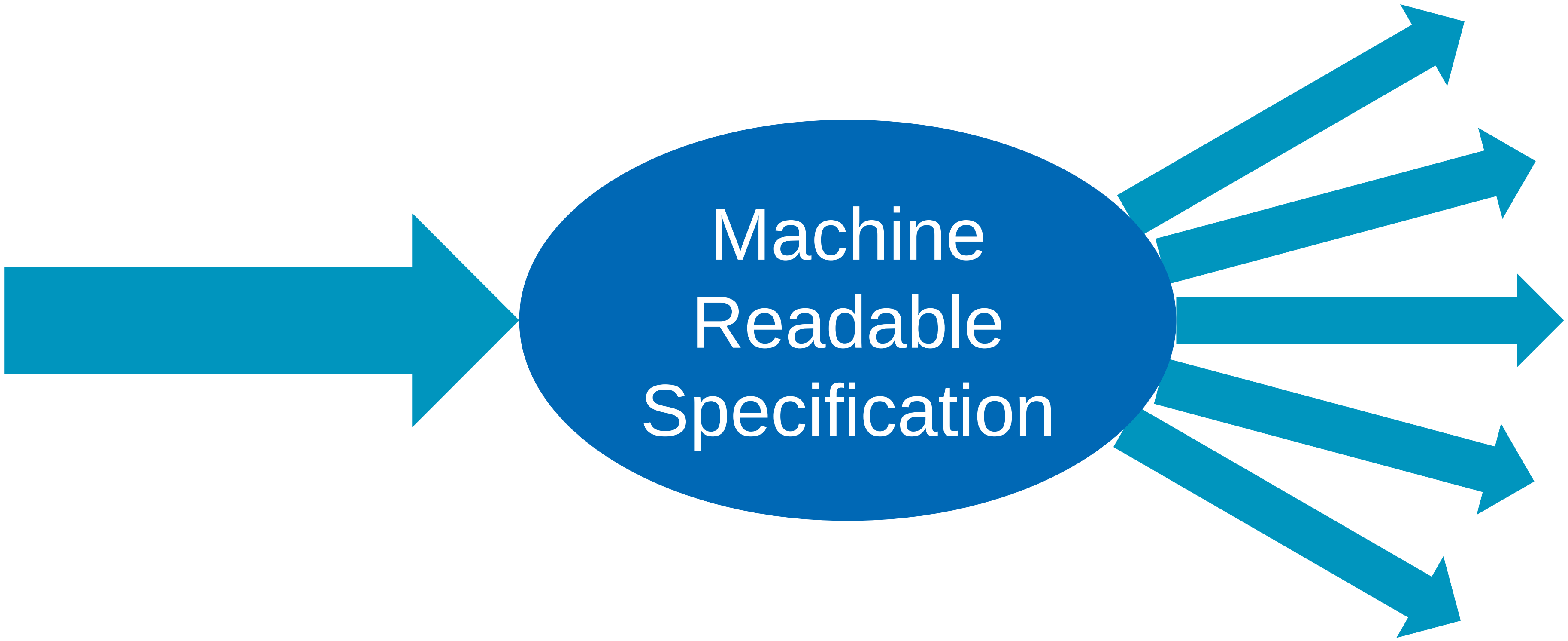
- reference model / checker, generate tests, measure coverage, fuzzing, …

Formal verification

- verify hardware and software sides of interface

Security

- malware analysis
- is architecture secure?


Machine
Readable
Specification

Barriers to this vision

- Making one size suit all
- Validation, validation, validation
- Effort
- Building confidence in engineers
- Building confidence in management
- Conway’s law
- Timing and brownfield sites
- Disadvantages of this approach

Making one size fit all

Diverse needs

- Readability (documentation)
- Performance (simulators)
- Ease of verification
- Match the implementation I am building
- Abstract over all implementations

The best specification language is

WEAK

and

INEXPRESSIVE

You can’t make everybody happy – you’re not an avocado

One of my wife’s favourite T-shirts

Be pragmatic

Aim for all users being able to use 97-99% of the spec.

Enable them to replace 1-3% of the spec with their own code that is

- Faster
- Better for interactive theorem proving
- …

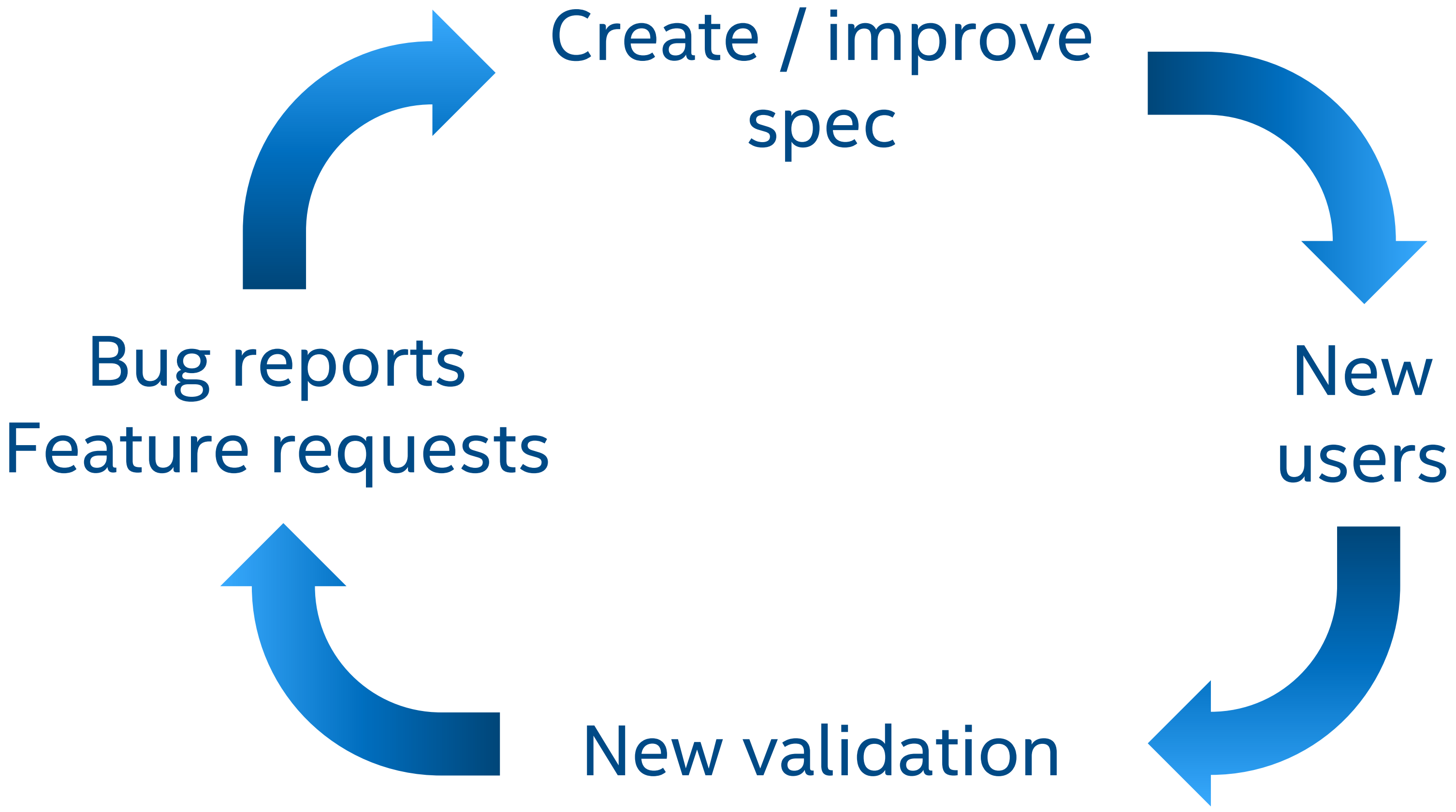
Address the resulting validation challenge

Design specification to be

USEFUL TO AS MANY USERS AS POSSIBLE

from the beginning

Creating a virtuous cycle



Conway's Law

... organizations which design systems [...] are constrained to produce designs which are copies of the communication structures of these organizations.

— *Melvin E. Conway (How do committees invent? 1968)*

If you have four groups working on a compiler, you'll get a 4-pass compiler

— *Eric S. Raymond's paraphrase of Conway*

Timing and brownfield sites

Timing is as important in industrial research as in academic research

Too early

→ Nobody cares

Too late

→You have to displace their solution

→You have to replicate many quirks of their solution

Disadvantages of this approach

Reduces redundancy

Reduces expertise

How to add useful redundancy?

We're hiring

Machine (and human) Readable Specs

... can dramatically improve quality and reduce effort

... many barriers to creating MRS at scale

Suggestions

- Spec languages should be weak and inexpressive
- Good enough for everybody vs. perfect for a few
- Create a virtuous cycle

See also:
https://alastairreid.github.io/mrs-at-scale/
https://alastairreid.github.io/uses-for-isa-specs/

Open questions

- How to integrate non-functional properties into ISA specs?
  - Eg bounds on speculative execution
- How to add useful redundancy to catch spec bugs early?
- What uses have we not even thought of?
- … And lots of details in making existing ideas better.

Example: ADD instruction

```
unsigned_sum = UInt(src1) + UInt(src2);
signed_sum   = SInt(src1) + SInt(src2);
result       = unsigned_sum[osize-1 : 0];

flags.CF = if UInt(result) == unsigned_sum then ‘0’ else ‘1’;
flags.OF = if SInt(result) == signed_sum then ‘0’ else ‘1’;
flags.PF = if ParityEven(result[0 +: 8]) then '1' else ‘0’;
flags.ZF = if IsZero(result) then '1' else '0';
flags.SF = result[osize-1];
```