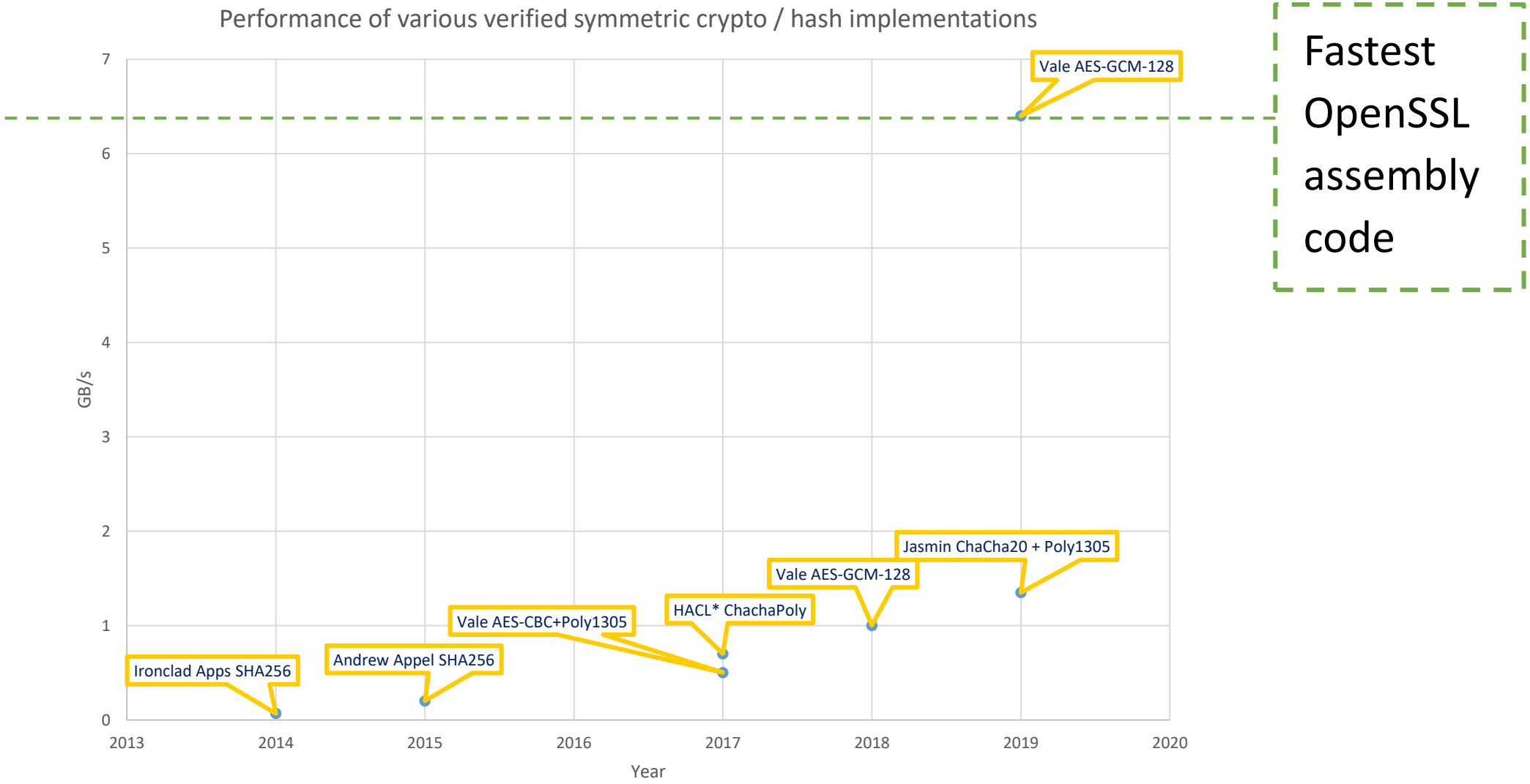
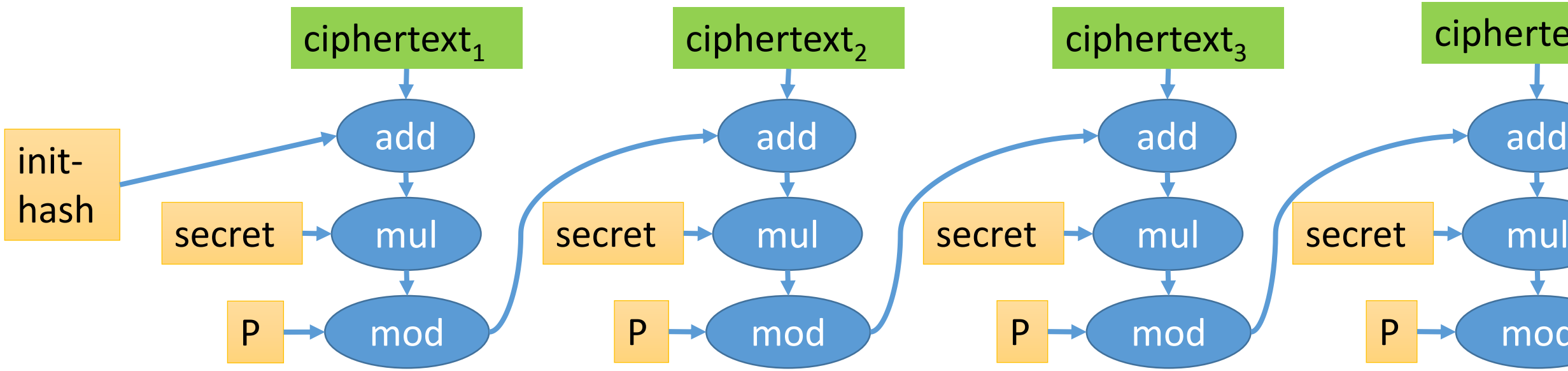
Verified Assembly Language in Vale / F*

We have a ***fast*** verified AES-GCM



Optimizing AES-GCM



Important optimizations:

- delay mod operations
- parallelize add/mul operations
- math+bitwise tricks for mod
- careful instruction scheduling

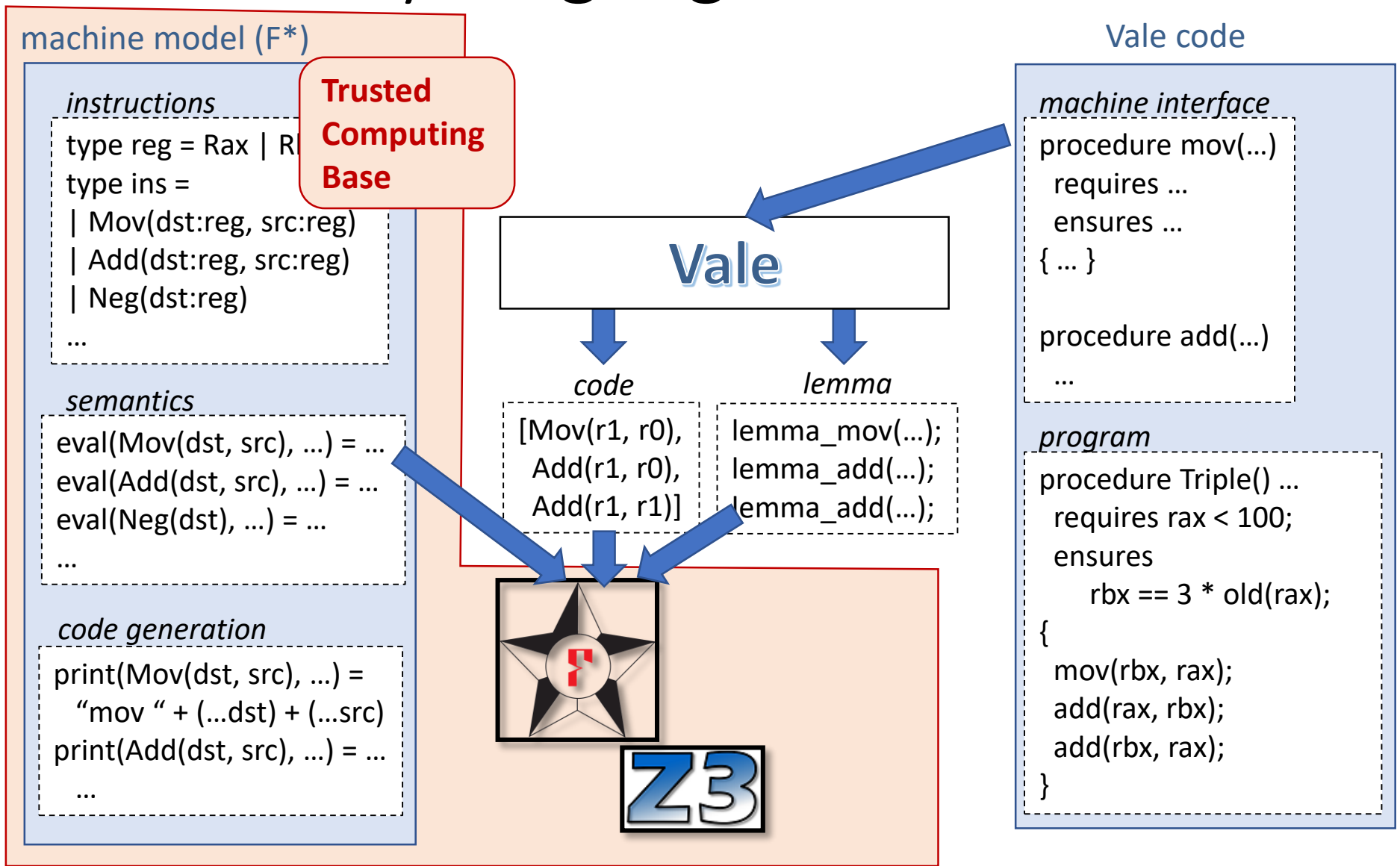
$(((init + c_1) * s \% P + c_2) * s \% P + c_3) * s \% P$

→ $(((init + c_1) * s + c_2) * s + c_3) * s \% P$

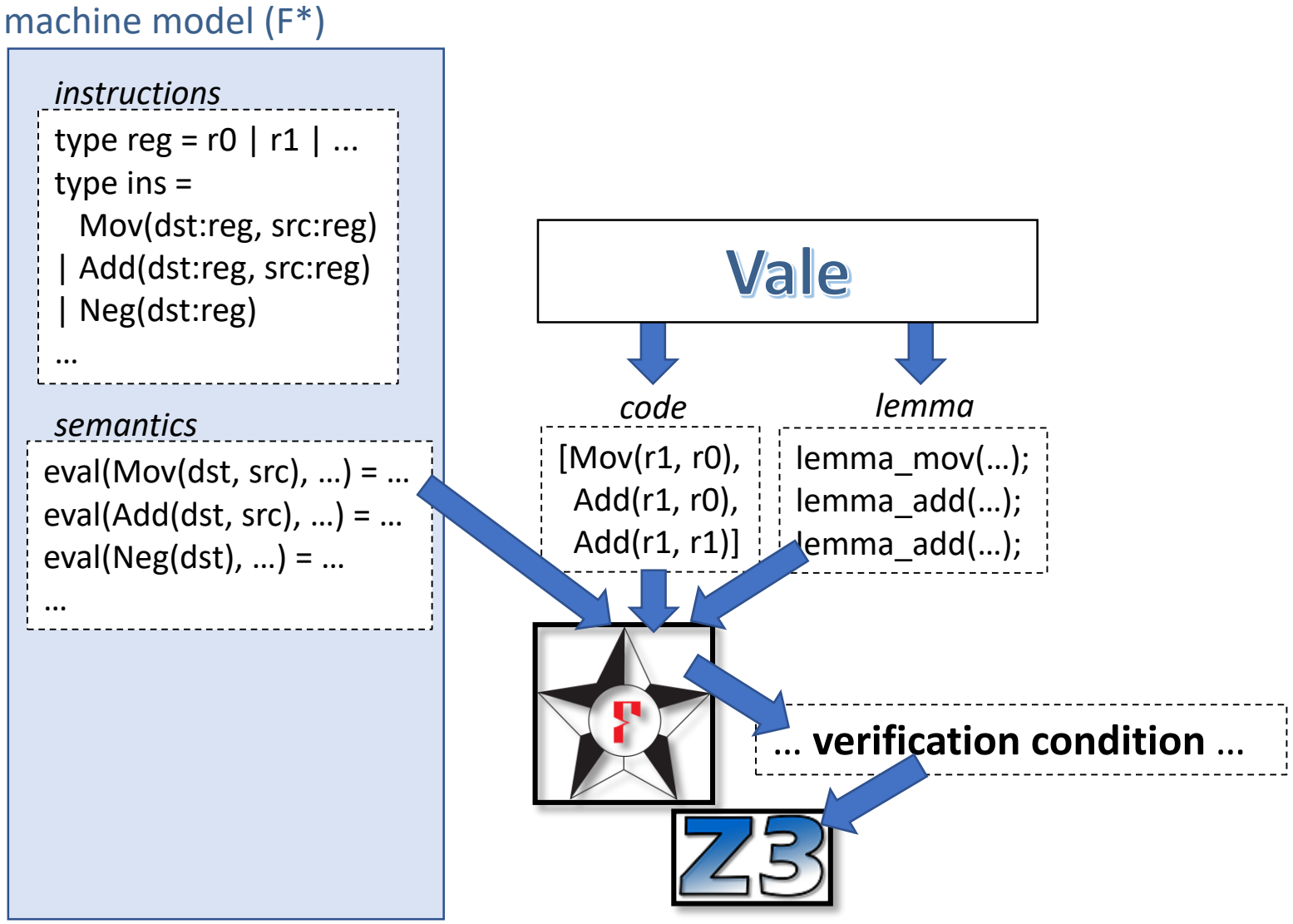
→ $((init + c_1) * s^3 + c_2 * s^2 + c_3 * s^1) \% P$

→ $((init + c_1) * (s^3 \% P) + c_2 * (s^2 \% P) + c_3 * s^1) \% P$

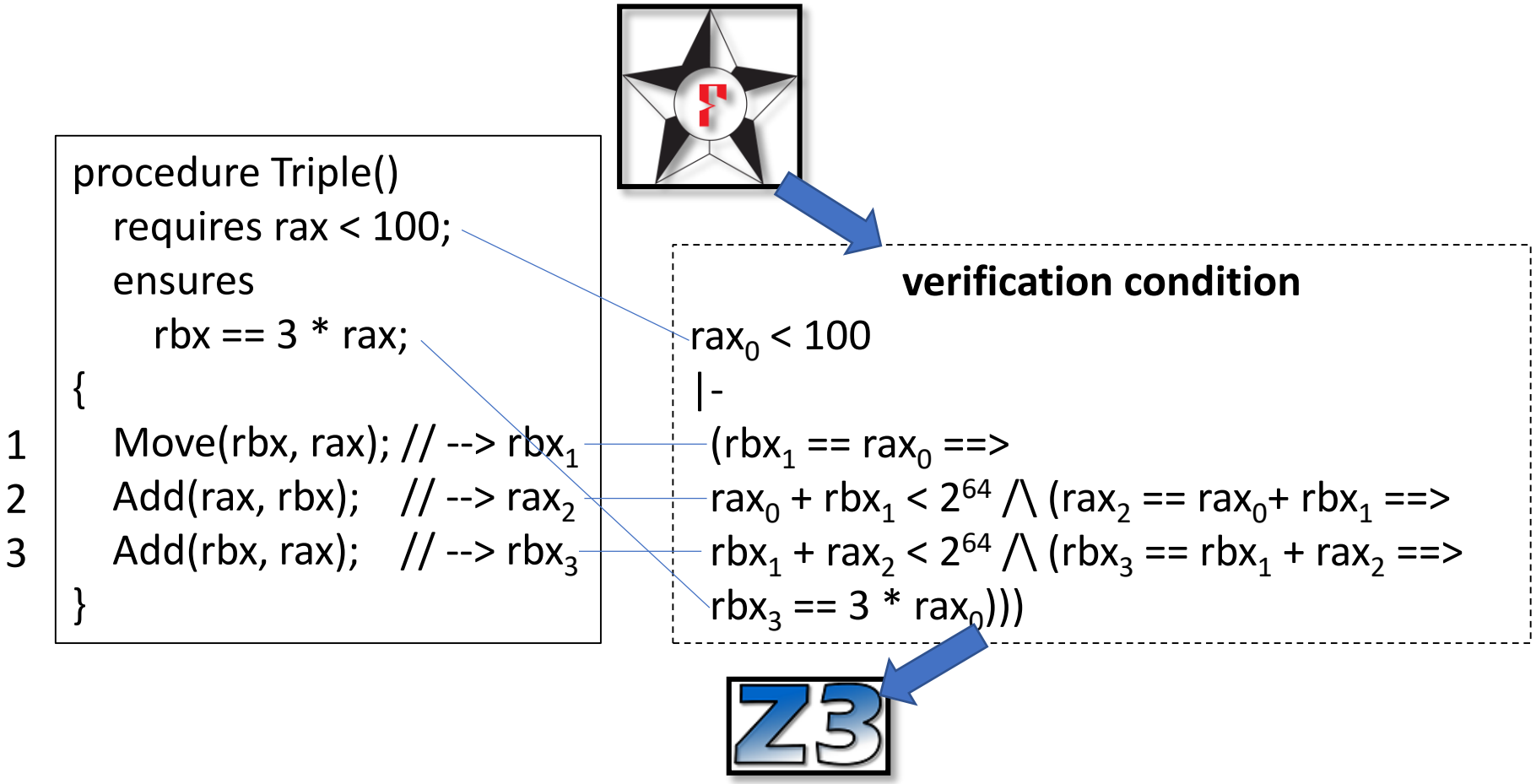
Vale: extensible, automated assembly language verification



Vale: extensible, automated assembly language verification



Verification condition



Ugh! Default SMT query looks awful!

verification condition we want:

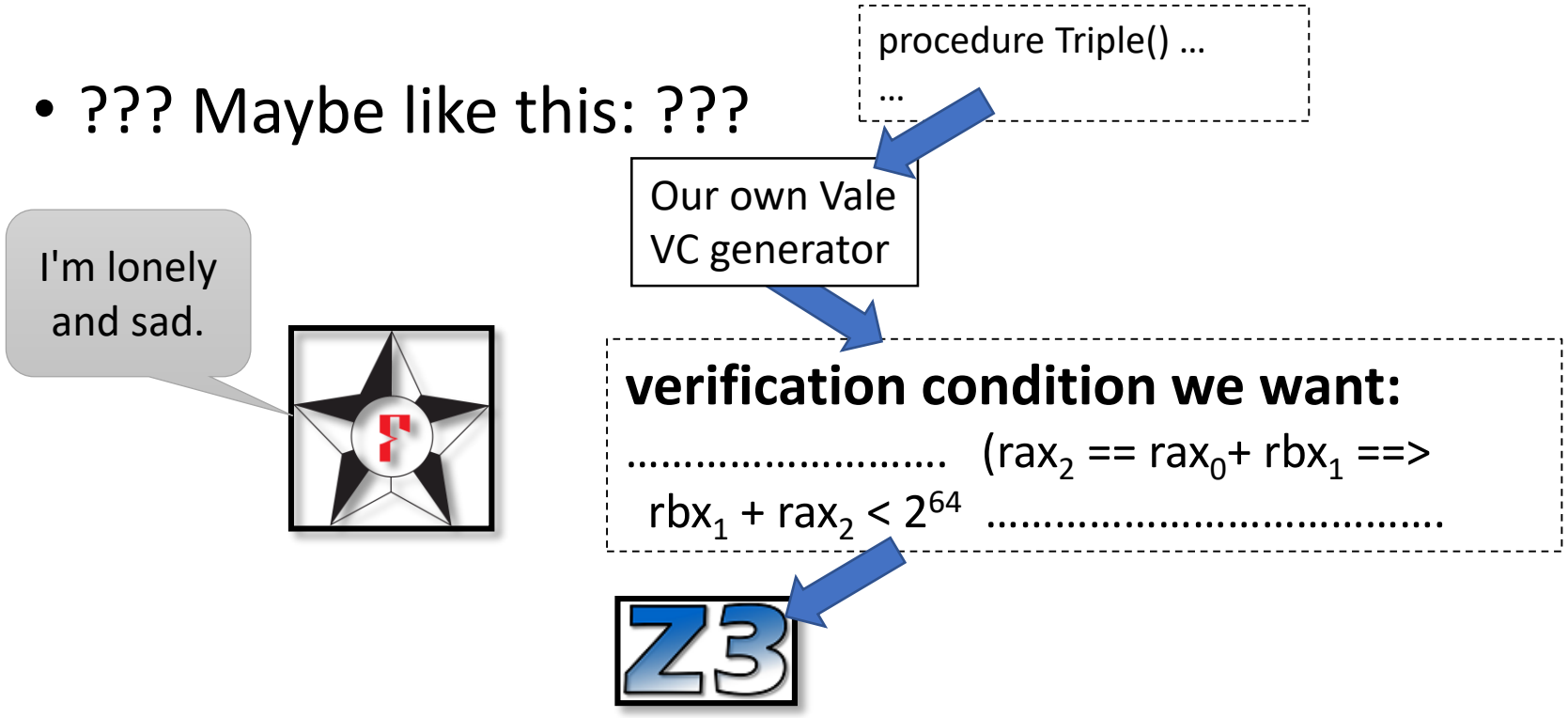
…… (rax_2 == rax_0+ rbx_1 ==>
rbx_1 + rax_2 < 2^{64} ……

verification condition we get:

```
…
(forall (ghost_result_0:(state * fuel)).
  (let (s3, fc3) = ghost_result_0 in
    eval_code (Ins (Add64 (OReg (Rax)) (OReg (Rbx)))) fc3 s2 == Some s3 /\
    eval_operand (OReg Rax) s3 == eval_operand (OReg Rax) s2 + eval_operand (OReg Rbx) s2 /\
    s3 == update_state (OReg Rax).r s3 s2) ==>
  lemma_Add s2 (OReg Rax) (OReg Rbx) == ghost_result_0 ==>
  (forall (s3:state) (fc3:fuel). lemma_Add s2 (OReg Rax) (OReg Rbx) == Mktuple2 s3 fc3 ==>
    Cons? codes_Triple.tl /\
    (forall (any_result0:list code). codes_Triple.tl == any_result0 ==>
      (forall (any_result1:list code). codes_Triple.tl.tl == any_result1 ==>
        OReg? (OReg Rbx) /\ eval_operand (OReg Rbx) s3 + eval_operand (OReg Rax) s3 < 2^64
…
```

Let's write our own VC generator!

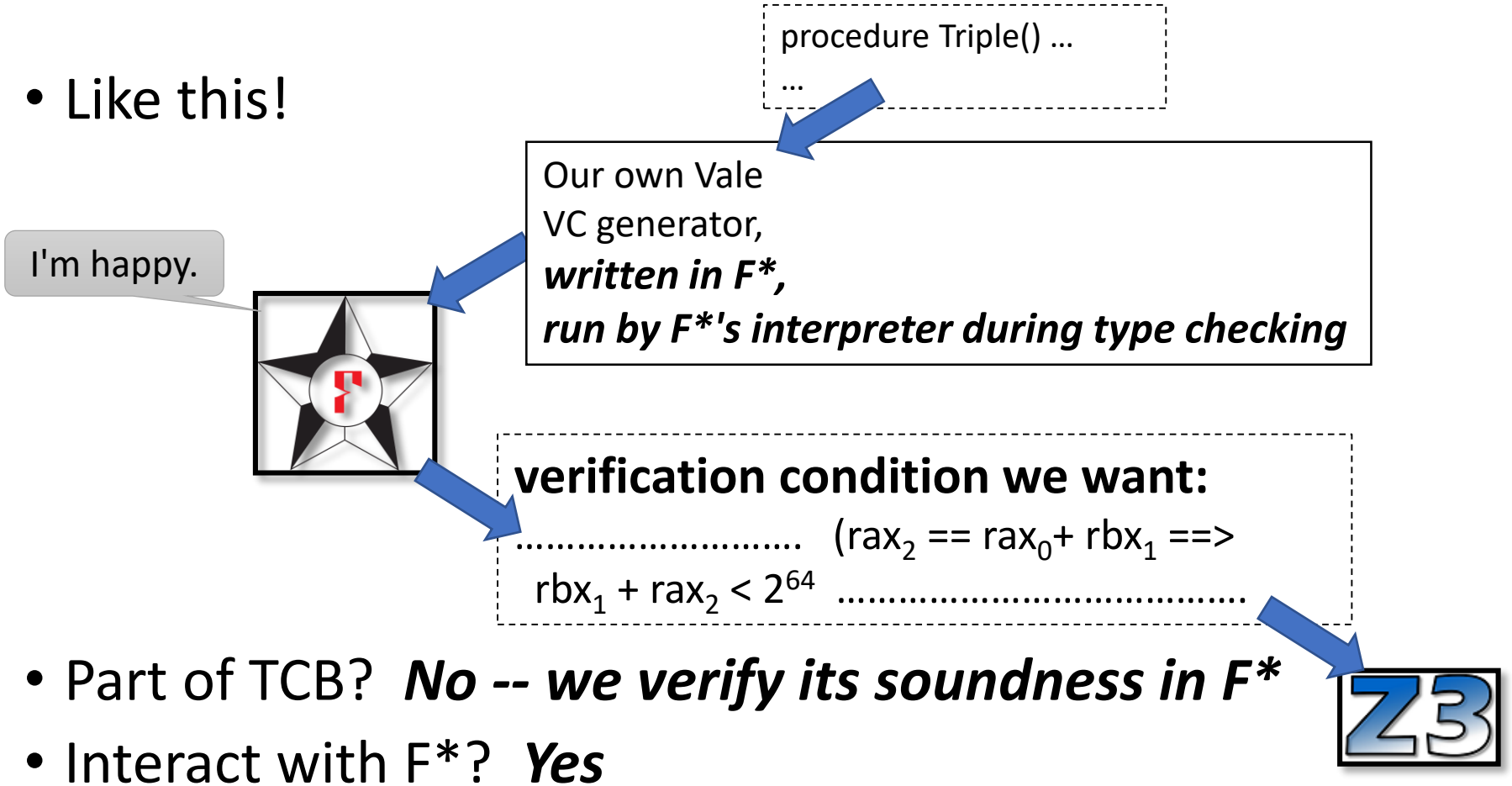
- ??? Maybe like this: ???



- But won't it be part of TCB?
- And how do we interact with F*?
- Can we reuse F* features and libraries?

Let's write our own VC generator!

- Like this!



- Part of TCB? ***No -- we verify its soundness in F****
- Interact with F*? ***Yes***
- Reuse F* features and libraries? ***Yes***

Let's write our own VC generator!

procedure Triple() …
…

Our own Vale
VC generator,
written in F*,
run by F*'s interpreter

verification condition we want:
……(rax$_2$ == rax$_0$+ rbx$_1$ ==>
rbx$_1$ + rax$_2$ < 2^{64} ……

A ~~b…ing~~?
A datatype:

```
type quickCode = ...
type quickCodes =
| QEmpty
| QSeq of quickCode * quickCodes ...
| QLemma of ... (Lemma pre post) * ...
```

Like our earlier code AST,
but with assertions, lemma calls,
ghost variables, etc.

A ~~b…ing~~?
A ~~d…pe~~?
An F* term:

(forall rbx$_1$. rbx$_1$ == rax$_0$ ==>
rax$_0$ + rbx$_1$ < 2^{64} /\
(forall rax$_2$. rax$_2$ == rax$_0$+ rbx$_1$ ==>
rbx$_1$ + rax$_2$ < 2^{64} /\ …

Demo

- Verification condition generation for Vale