barkhausen institut

Debug, Execute, Verify!

Development-Verification Co-Design Made Practical

František Farka, **Carmine Abate**, Shuanglong Kan, Sebastian Ertel

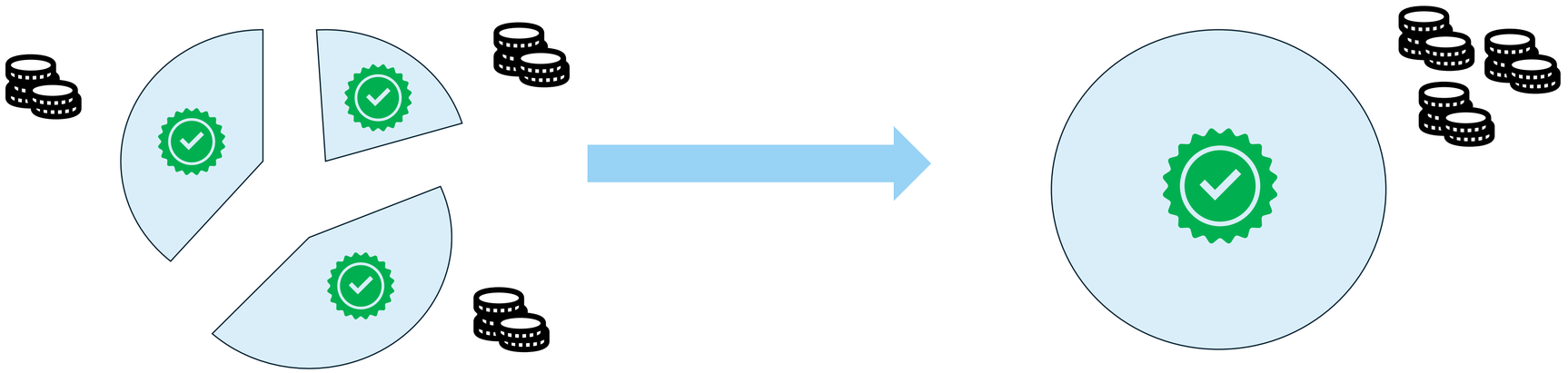
MOTIVATION

SYSTEM CODE

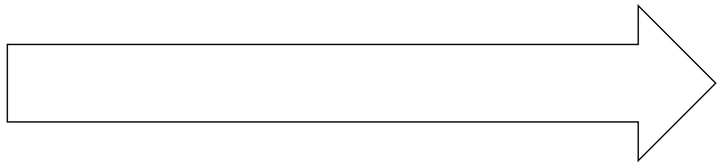
- Large, fast growing/evolving
- Complex: multiple components, concurring to resources, e.g. memory

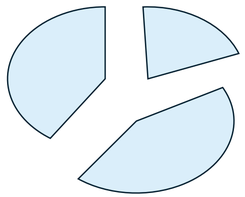
VERIFICATION

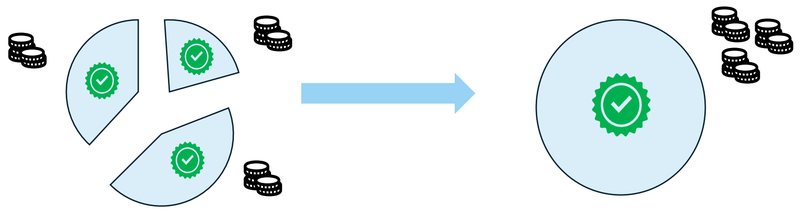
- Co-Design with dev, for NON-Expert
- Logic: higher order, concurrent separation logic



OUR FRAMEWORK





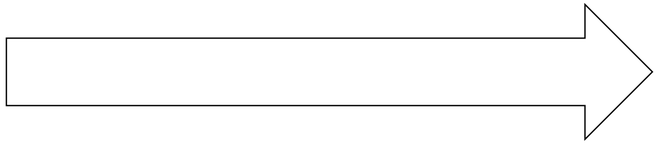


OUR FRAMEWORK IN ACTION

```
fn sub_42(a : &u8) -> u8 {
  let x = *a;
  let mut y = 0;
  if (x > 41) {
    y = x - 42;
    y }
  else { panic!() }
}
```

"if $* a > 41$, then no panic"

OUR FRAMEWORK IN ACTION



```
fn sub_42(a : &u8) -> u8 {
  let x = *a;
  let mut y = 0;
  if (x > 41) {
    y = x - 42;
    y }
  else { panic!() }
}
```

v_a : val

$v_a > 41$

$a \longmapsto \&u8\ l_a$

$l_a \longmapsto u8\ v_a$

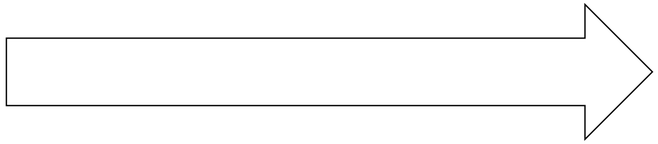
NO PANIC

Variables, Hypothesis

Resources

Goal

OUR FRAMEWORK IN ACTION



```
fn sub_42(a : &u8) -> u8 {
  let x = *a;
  let mut y = 0;
  if (x > 41) {
    y = x - 42;
    y }
  else { panic!() }
}
```

v_a : val

$v_a > 41$

$a \mapsto \&u8\ l_a$

$l_a \mapsto u8\ v_a$

$x \mapsto u8\ v_a$

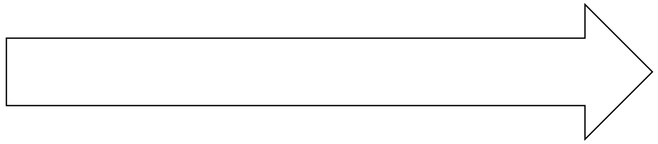
NO PANIC

Variables, Hypothesis

Resources

Goal

OUR FRAMEWORK IN ACTION



```
fn sub_42(a : &u8) -> u8 {
  let x = *a;
  let mut y = 0;
  if (x > 41) {
    y = x - 42;
    y }
  else { panic!() }
}
```

v_a : val

$v_a > 41$

$a \mapsto \&u8\ l_a$

$l_a \mapsto u8\ v_a$

$x \mapsto u8\ v_a$

$y \mapsto u8\ 0$

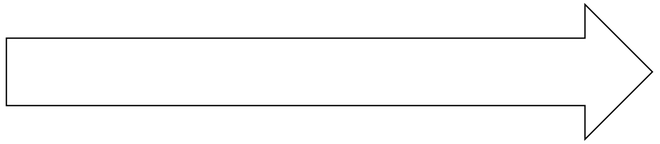
NO PANIC

Variables, Hypothesis

Resources

Goal

OUR FRAMEWORK IN ACTION



```
fn sub_42(a : &u8) -> u8 {
  let x = *a;
  let mut y = 0;
  if (x > 41) {
    y = x - 42;
    y }
  else { panic!() }
}
```

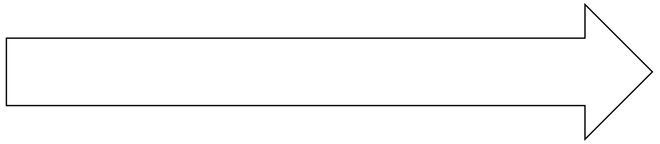
v_a : val	v_a : val
$v_a > 41$	$v_a > 41$
$v_a > 41$	$v_a \not> 41$
$a \longmapsto \&u8\ l_a$	$a \longmapsto \&u8\ l_a$
$l_a \longmapsto u8\ v_a$	$l_a \longmapsto u8\ v_a$
$x \longmapsto u8\ v_a$	$x \longmapsto u8\ v_a$
$y \longmapsto u8\ 0$	$y \longmapsto u8\ 0$
NO PANIC	NO PANIC

Variables, Hypothesis

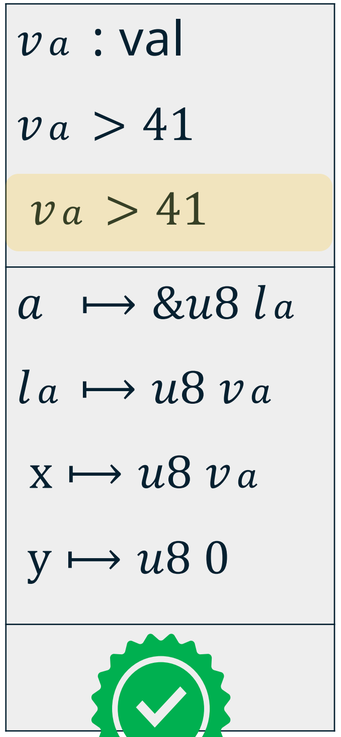
Resources

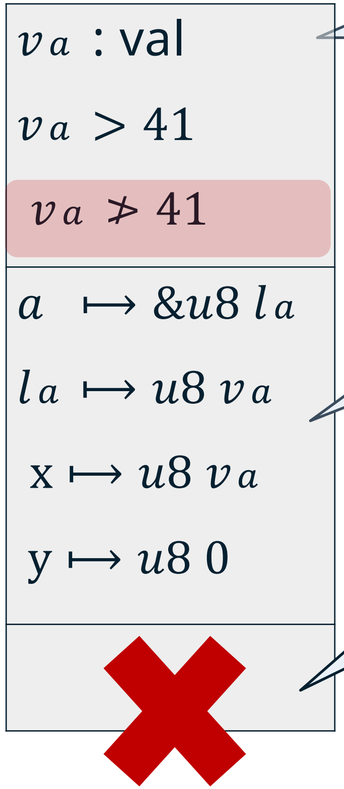
Goal

OUR FRAMEWORK IN ACTION



```
fn sub_42(a : &u8) -> u8 {
  let x = *a;
  let mut y = 0;
  if (x > 41) {
    y = x - 42;
    y }
  else { panic!() }
}
```



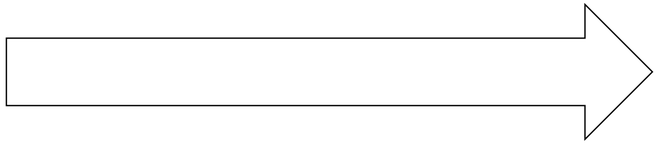


Variables, Hypothesis

Resources

Goal

OUR FRAMEWORK IN ACTION

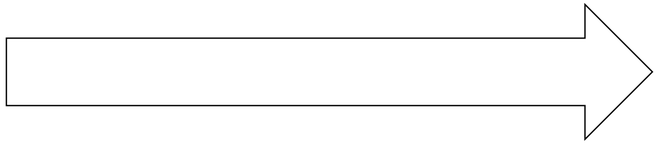


```
fn sub_42(a : &u8) -> u8
```

“if $* a > 41$, then no panic”

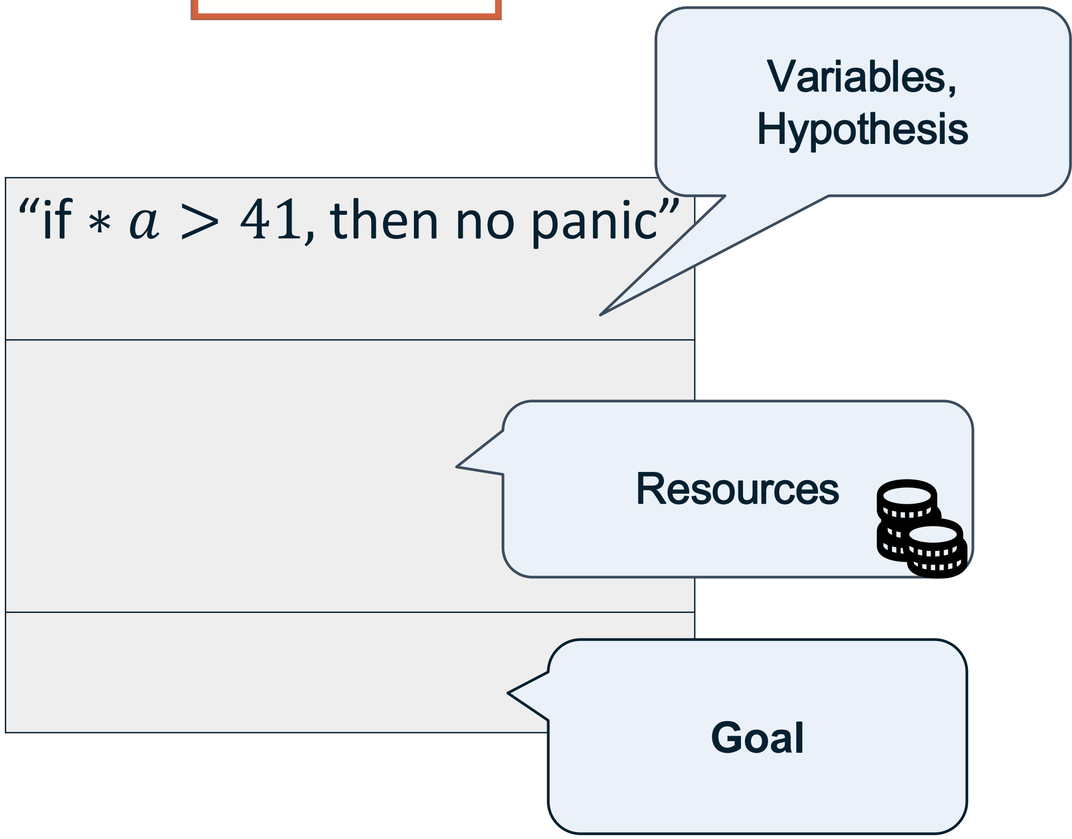
```
fn caller (..) {
    ..
    a := .. ;
    sub_42(a);
}
```

OUR FRAMEWORK IN ACTION

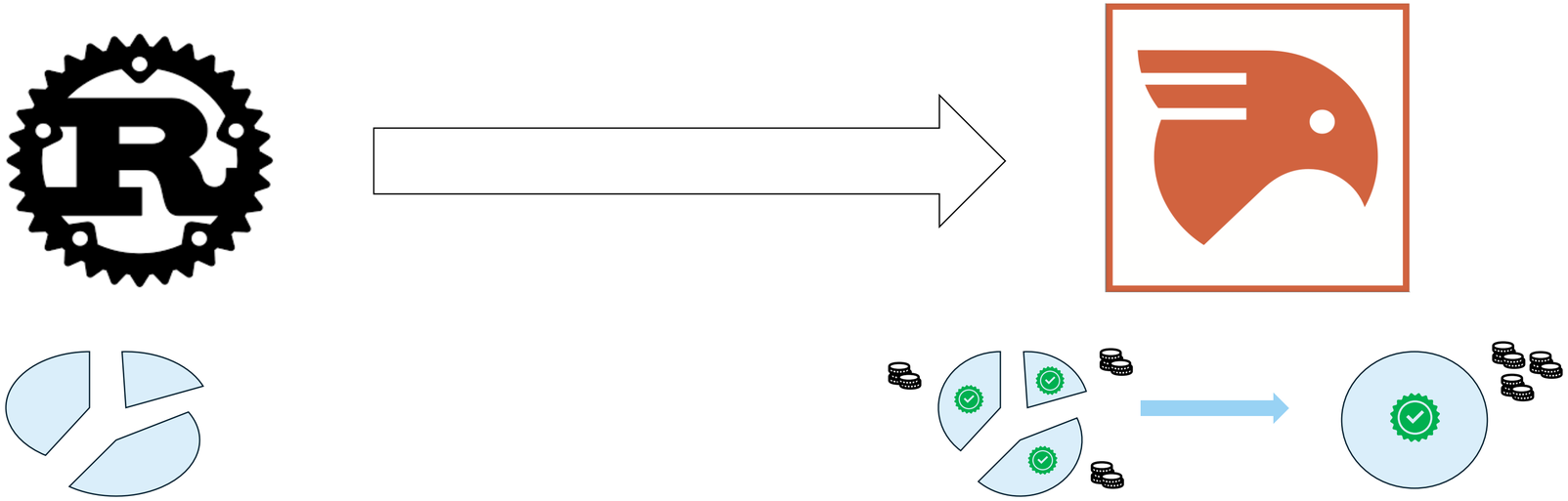


```rust
fn sub_42(a : &u8) -> u8
```

```rust
fn caller (..) {
    ..
    a := .. ;
    sub_42(a);
}
```

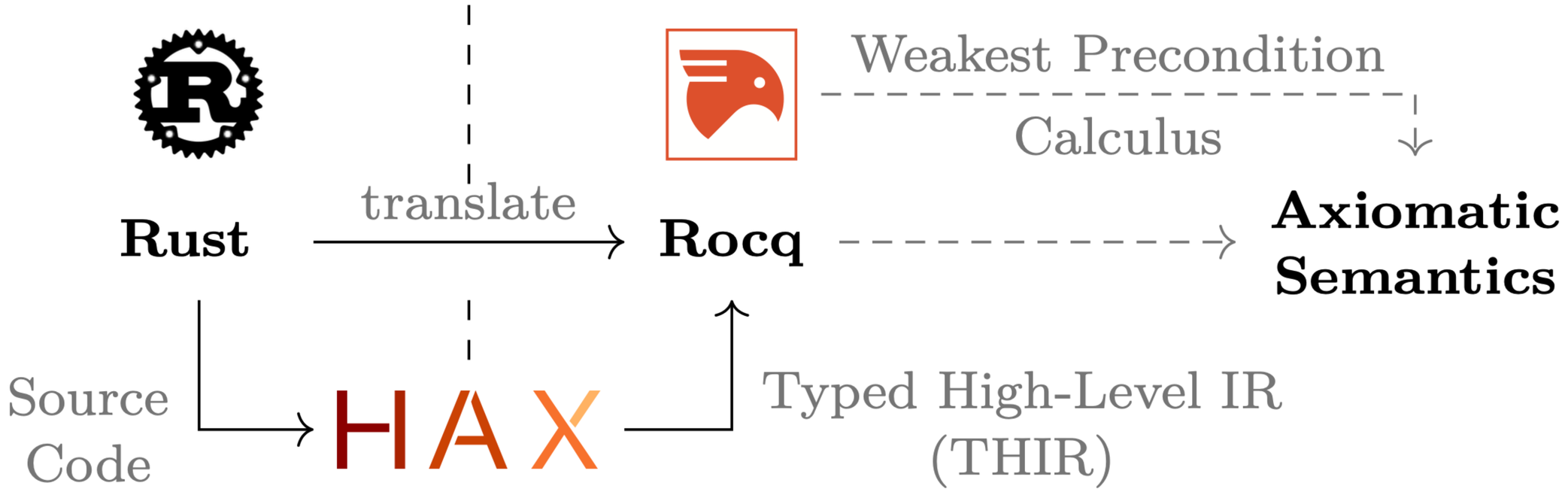


CONCLUSIONS



Binary Trie (In the Paper)
Unikernel (Work in Progress)

BACKUP


Weakest Precondition
Calculus
translate
Rust
Rocq
Axiomatic
Semantics
Source
Code
HAX
Typed High-Level IR
(THIR)

Formal Verification of Rust Code

- unsafe { … }
- liveness, confidentiality, functional correctness ≠ memory safety

FULLY AUTOMATIC	INTERACTIVE
e.g., Verus, Creusot, Prusti, Kani ..	e.g., Aeneas, Refined Rust
+ Rust code + accessible to non-experts - hard to recover from failure - limited expressiveness - large TCB	- IR - require expertise + high expressiveness + small TCB

Our Solution:
+ (very close to) Rust code
+ accessible to non-experts
+ high expressiveness
+ small TCB

Formal Verification of Rust Code

- unsafe { … }
- memory safety ≠ liveness, confidentiality, functional correctness ..

```
1  fn sub_42(a : &u8) -> u8 {
2    let x = *a;
3    let mut y = 0;
4
5    y = x - 42;
6    y
7
8  }
```

Runtime Crash
or
(Silent)
Wrap Arithmetic?