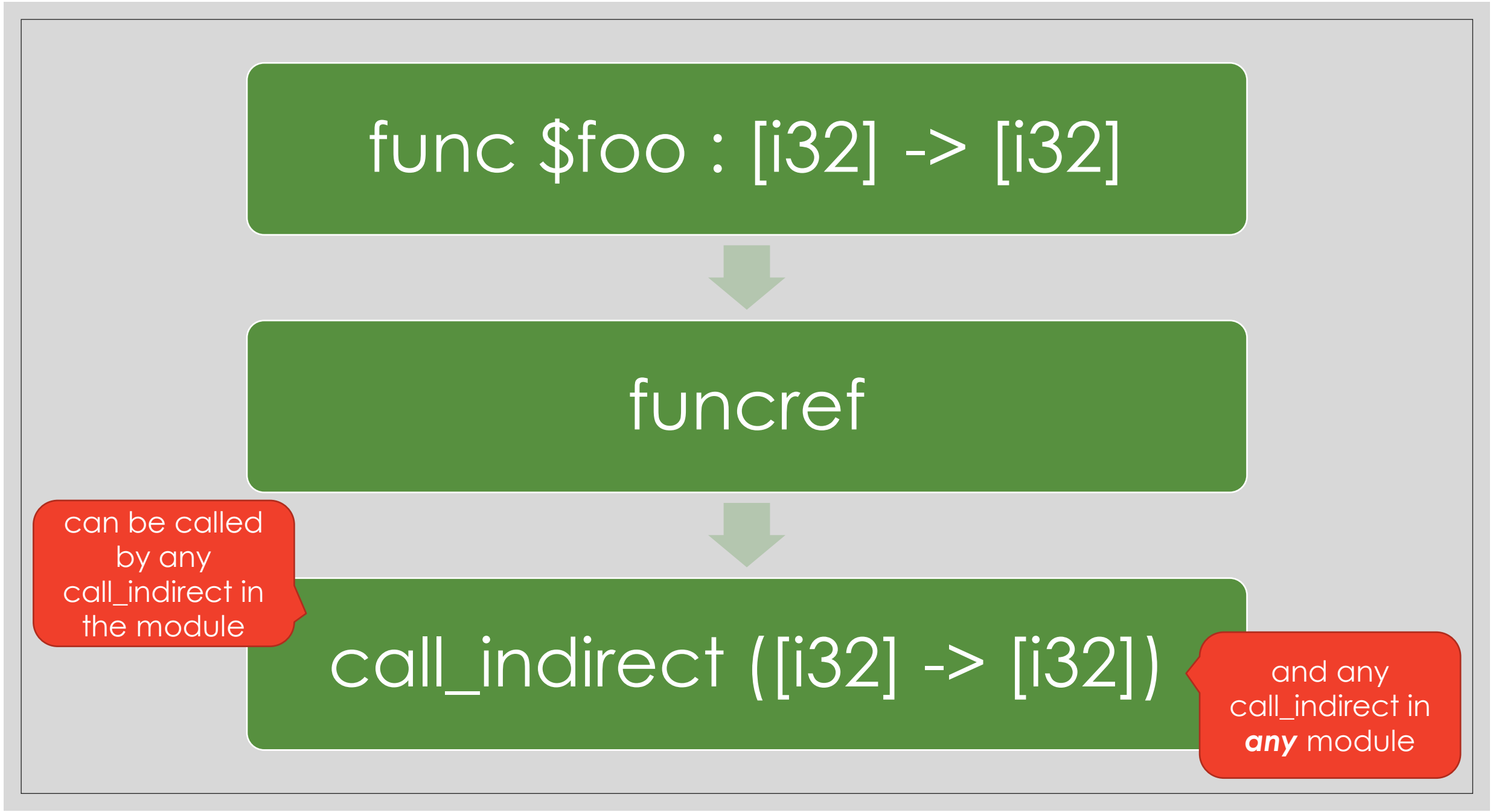
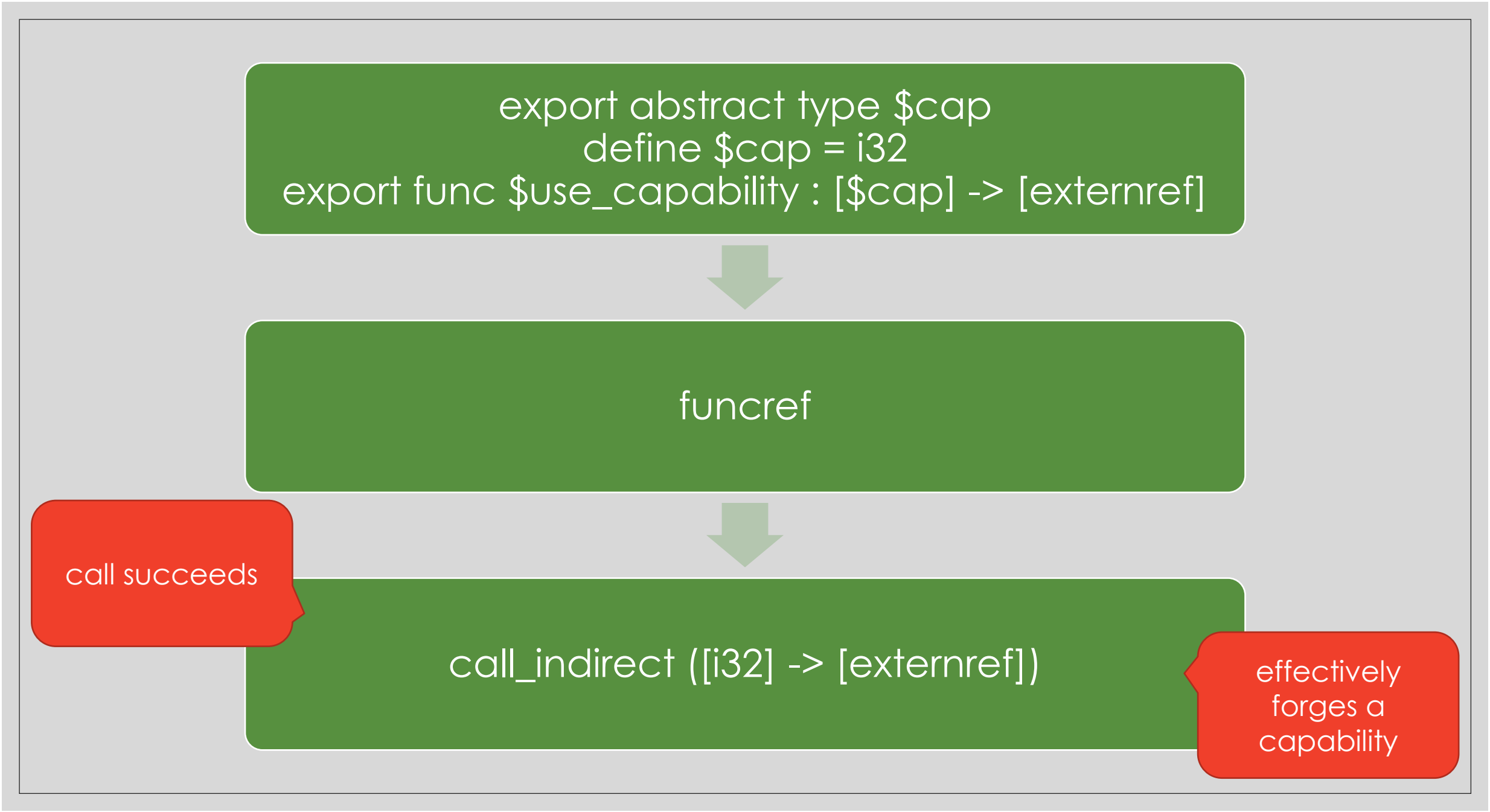
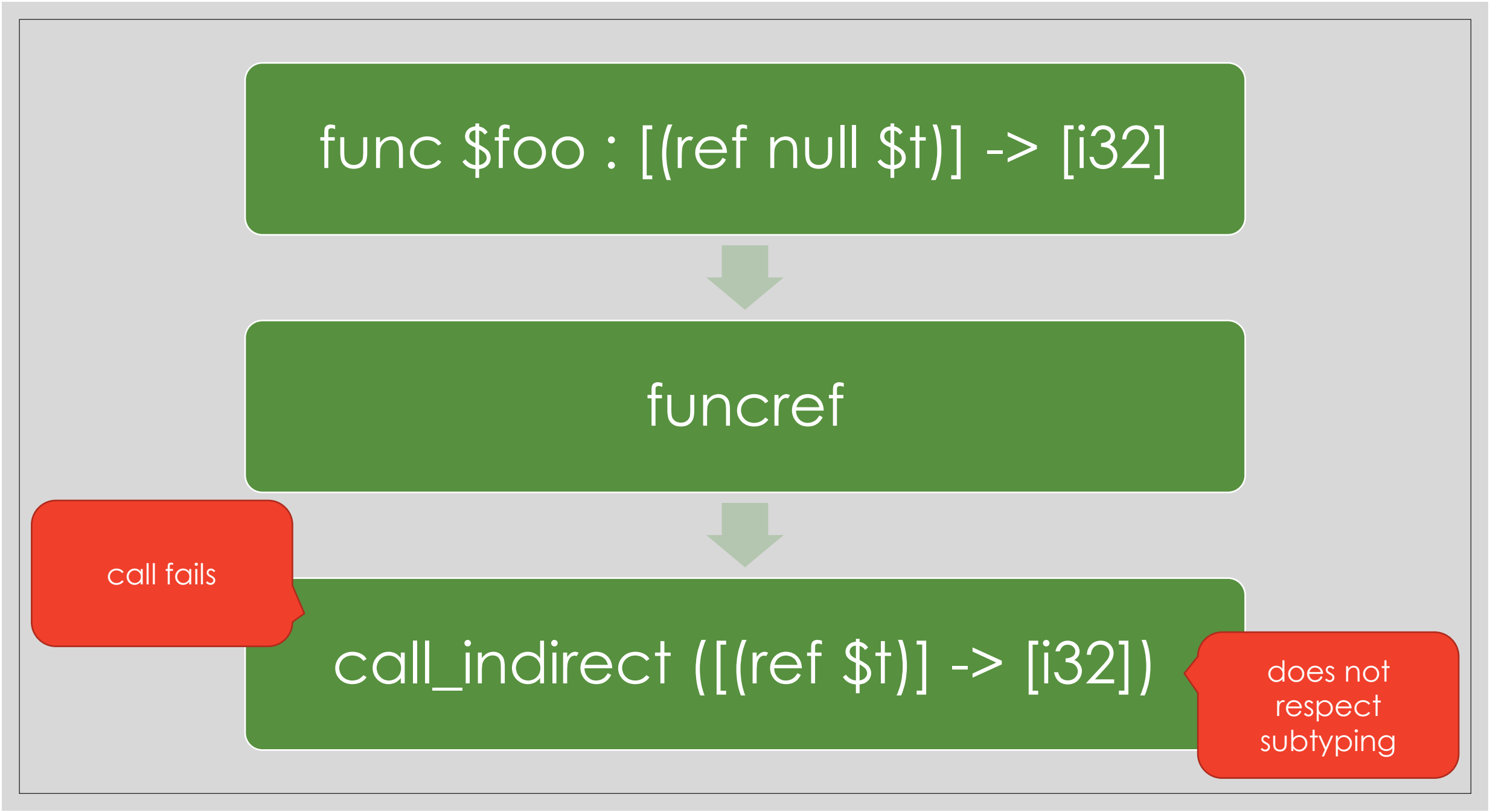
DISPATCH TAGS

Ross Tate

PROBLEM


func $foo : [i32] -> [i32]
funcref
call_indirect ([i32] -> [i32])
can be called by any call_indirect in the module
and any call_indirect in ***any*** module


export abstract type $cap
define $cap = i32
export func $use_capability : [$cap] -> [externref]
funcref
call succeeds
call_indirect ([i32] -> [externref])
effectively forges a capability


func $foo : [(ref null $t)] -> [i32]
funcref
call_indirect ([(ref $t)] -> [i32])
call fails
does not respect subtyping

PROPOSAL

Dispach Tags

Old

- call_indirect $type (ignoring tables)

New

- call_indirect $tag (ignoring tables)
- canonical $tag for each $type
  - backwards compatibility
  - $type cannot mention imports/exports
- declare new tags in event section
  - dispatch_tag $tag : [i32 i32] -> [i64]
- tags can be imported and exported
  - supports *intended* cross-module calls
  - respects abstract types

Associating Tags

Old

- func $foo (param i32) (result i32)
  - automatically associates canonical tag

New

- func $foo (…) (tag $tag1 $tag2 …)
  - explicitly specifies associated tags
    - if no associated tags are exported then function cannot be indirectly called by other modules
  - tag list can be empty
    - ensures no indirect calls
  - tag types must be *compatible* with sig
    - but do not have to be identical to sig
    - supports subtyping

EXTENSION

Tag Switching

```
(dispatch_func $funcref
    (on_tag $tag1 $func1)
    (on_tag $tag2 $func2)
    …
    (trap)
)
```

- Defines a funcref
- Funcref checks which tag it was indirectly called to
- Calls a function with compatible signature
- Traps if there is no match

Applications

Interface-Method Dispatch

- new tag for each interface method
- V-Table Creation
  - have an interface funcref that switches on tag of each interface method implemented by class
- Interface-Method Invocation
  - load interface funcref from v-table
  - indirect call with interface method's tag

Closure Dispatch in Functional Langs

- new tag for each arity
- Closure creation
  - have a funcref that switches on tag specifying the number of args supplied
- Closure application
  - indirect call the closure funcref with tag specifying the number of args supplied