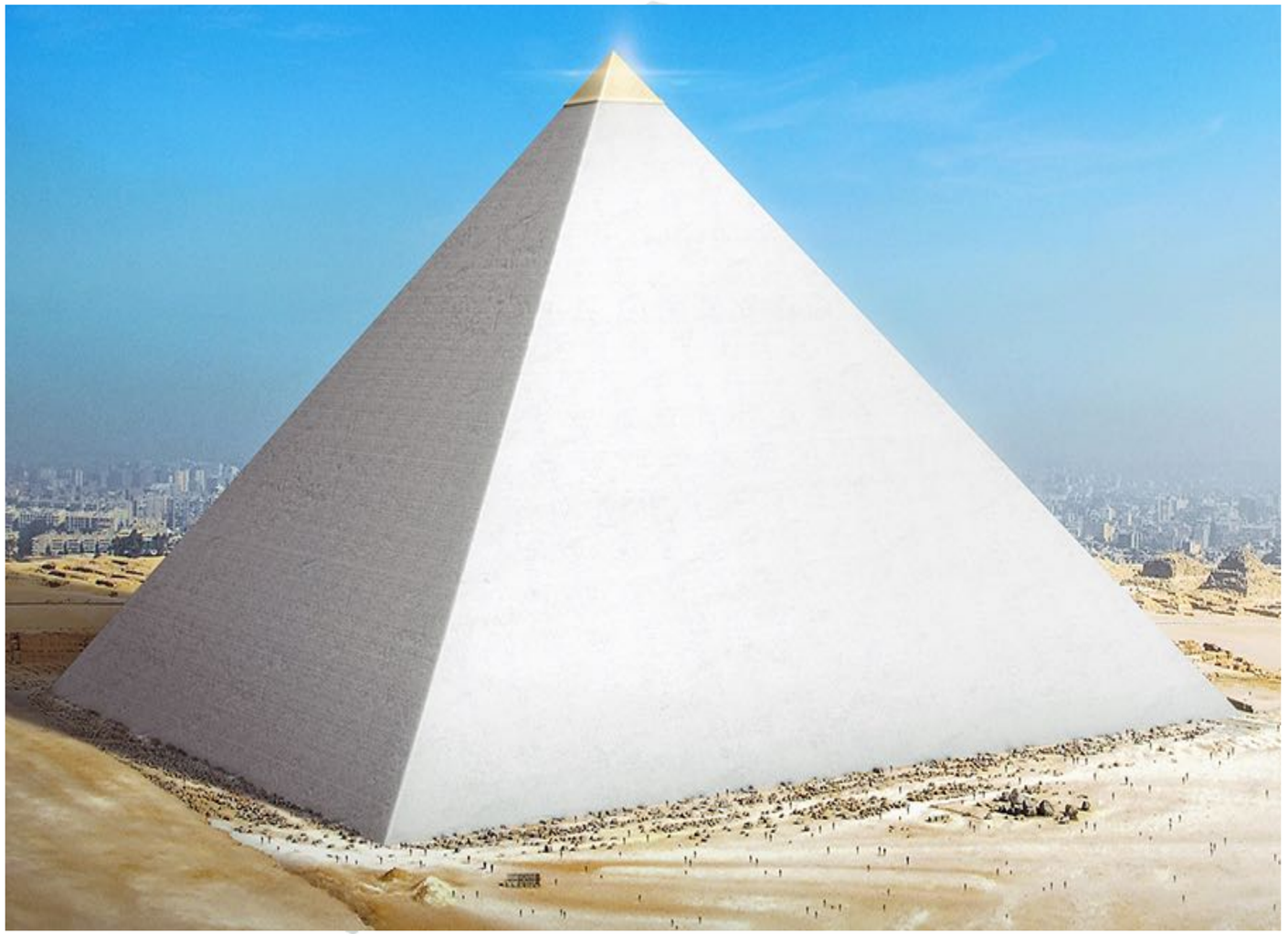
# Verification of Software Network Functions with No Verification Expertise

Arseniy Zaostrovnykh, Solal Pirelli, Rishabh Iyer, Matteo Rizzo,
Luis Pedrosa, Katerina Argyraki, George Candea

# Bright Future With Verification



# Bright Future With Verification

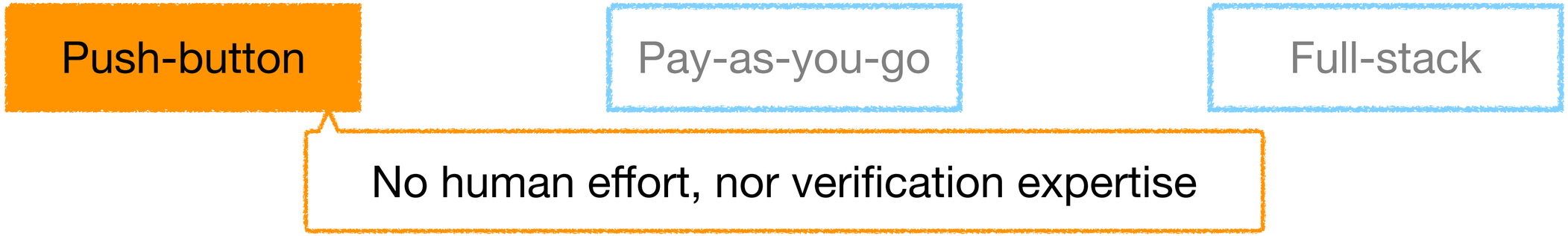
Push-button

Pay-as-you-go

Full-stack

Vigor enables practical verification of software network functions.

- Virtualized networks
- Critical infrastructure, but unreliable software
- NF operators and devs have no verification expertise




- Traditional verification cost: 5-10x development effort†
- Ultimate goal: zero-cost verification

†seL4 [SOSP 2009]
Ironclad [OSDI 2014]
HACL* [CCS 2017]

Push-button

**Pay-as-you-go**

Full-stack

Support for partial specifications

```
EXT_PORT = 2

if received_on_port != EXT_PORT: # Heartbeat
    return ([],[])
```

```
from state import flow_emap, flow_id_to_backend_id, backends, backend_ip_emap, cht
EXP_TIME = 10 * 1000
BACKEND_EXP_TIME = 3600000000 * 1000
EXT_PORT = 2

if a_packet_received:
    flow_emap.expire_all(now - EXP_TIME)
    backend_ip_emap.expire_all(now - BACKEND_EXP_TIME)

h3 = pop_header(tcpudp, on_mismatch=([],[]))
h2 = pop_header(ipv4, on_mismatch=([],[]))
h1 = pop_header(ether, on_mismatch=([],[]))

assert a_packet_received
assert h1.type == 8 # 0x0800 == IPv4 in big endian
assert h2.npid == 6 or h2.npid == 17 # 6/17 -> TCP/UDP

if received_on_port == EXT_PORT: # Packet from the external network - client
    packet_flow = LoadBalancedFlow(h2.saddr, h2.daddr, h3.src_port, h3.dst_port, h2.npid)
    alloc_flow_and_process_packet = False;
    if flow_emap.has(packet_flow):
        flow_id = flow_emap.get(packet_flow)
        backend_id = flow_id_to_backend_id.get(flow_id)
        if backend_ip_emap.has_idx(backend_id):
            flow_emap.refresh_idx(flow_emap.get(packet_flow), now)
            backend = backends.get(backend_id)
            return ([backend.nic],
                    [ether(h1, saddr=..., daddr=backend.mac),
                     ipv4(h2, cksum=..., daddr=backend.ip),
                     tcpudp(h3)])
        else:
            flow_emap.erase(packet_flow)
            alloc_flow_and_process_packet = True
    else:
        alloc_flow_and_process_packet = True
    if alloc_flow_and_process_packet:
        if backend_ip_emap.exists_with_cht(cht, _LoadBalancedFlow_hash(packet_flow)):
            bknd = backend_ip_emap.choose_with_cht(cht, _LoadBalancedFlow_hash(packet_flow))
            if not flow_emap.full():
                idx = the_index_allocated
                flow_emap.add(packet_flow, idx, now)
                flow_id_to_backend_id.set(idx, bknd)
            backend = backends.get(bknd)
            return ([backend.nic],
                    [ether(h1, saddr=..., daddr=backend.mac),
                     ipv4(h2, cksum=..., daddr=backend.ip),
                     tcpudp(h3)])
        else:
            return ([],[])
else: # A heartbeat from a backend
    bknd_addr = ip_addrc(h2.saddr)
    if backend_ip_emap.has(bknd_eddr):
        backend_ip_emap.refresh_idx(backend_ip_emap.get(bknd_addr), now)
    else:
        if not backend_ip_emap.full():
            idx = the_index_allocated
            backend_ip_emap.add(bknd_addr, idx, now)
            backends.set(idx, LoadBalancedBackendc(received_on_port, h1.saddr, h2.saddr))
    return ([],[])
```

Push-button

Pay-as-you-go

Full-stack

Whole SW stack: framework, OS, drivers

**NF: 3 KLOC**

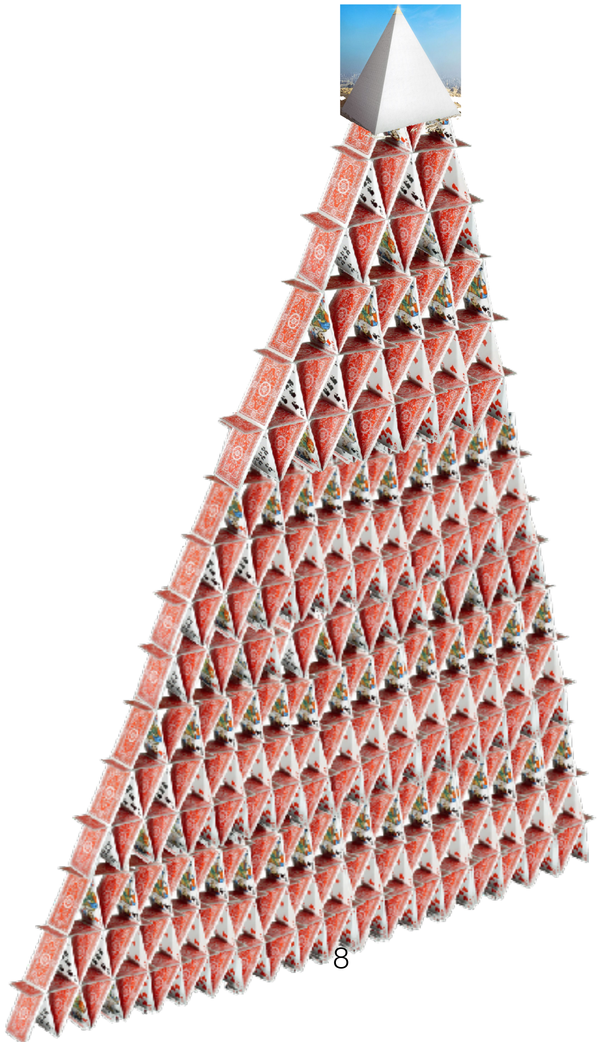
**Framework, driver: 85 KLOC**

Push-button

Pay-as-you-go

Full-stack

Whole SW stack: framework, OS, drivers



**NF: 3 KLOC**

**Framework, driver: 85 KLOC**

**Linux: 23 MLOC**

**Vigor enables**

Push-button **,**

Pay-as-you-go **, and**

Full-stack **verification of software NFs.**

# Outline

Push-button

Pay-as-you-go

Usage scenario

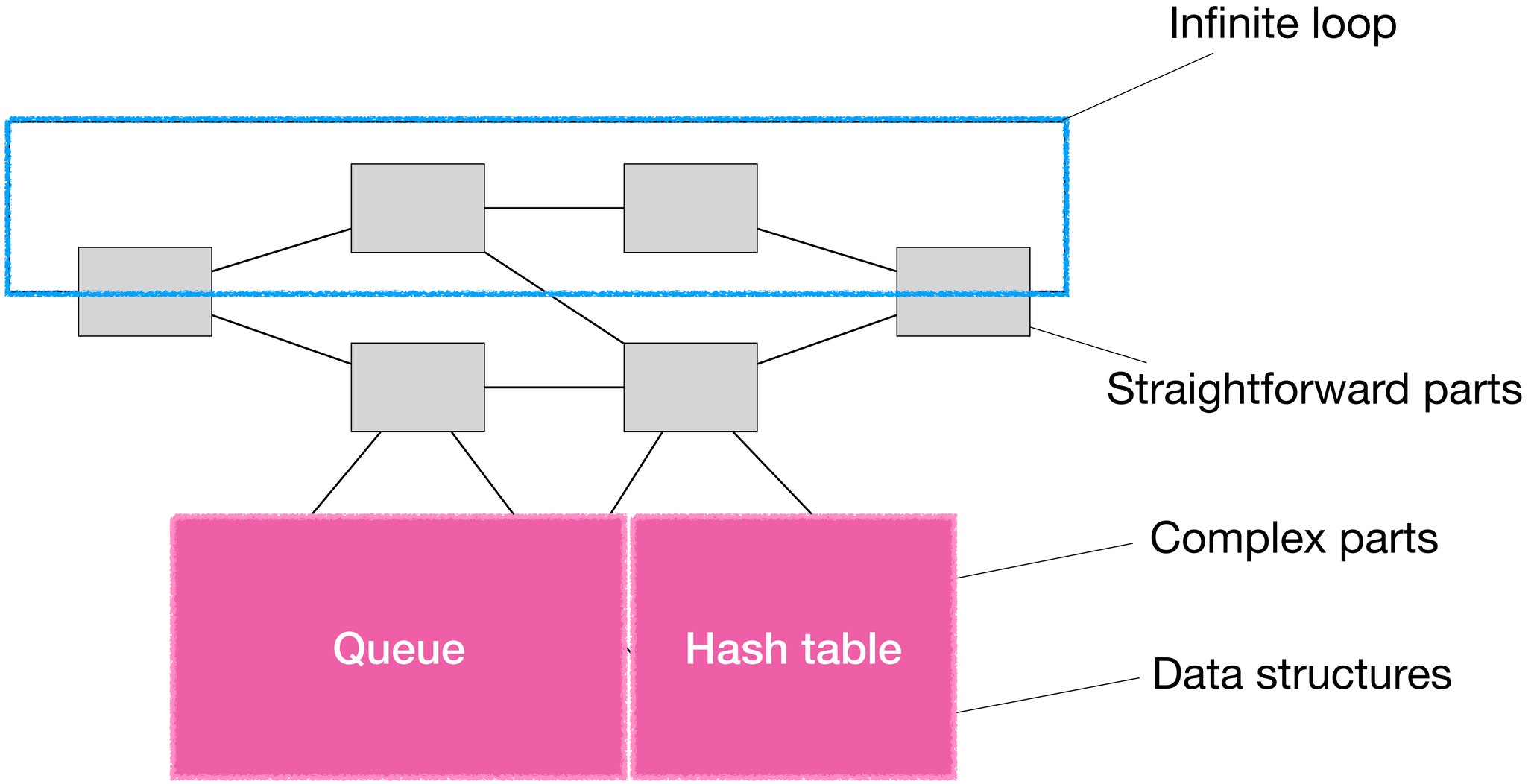
Full-stack

Evaluation [ Practical | No performance overhead ]

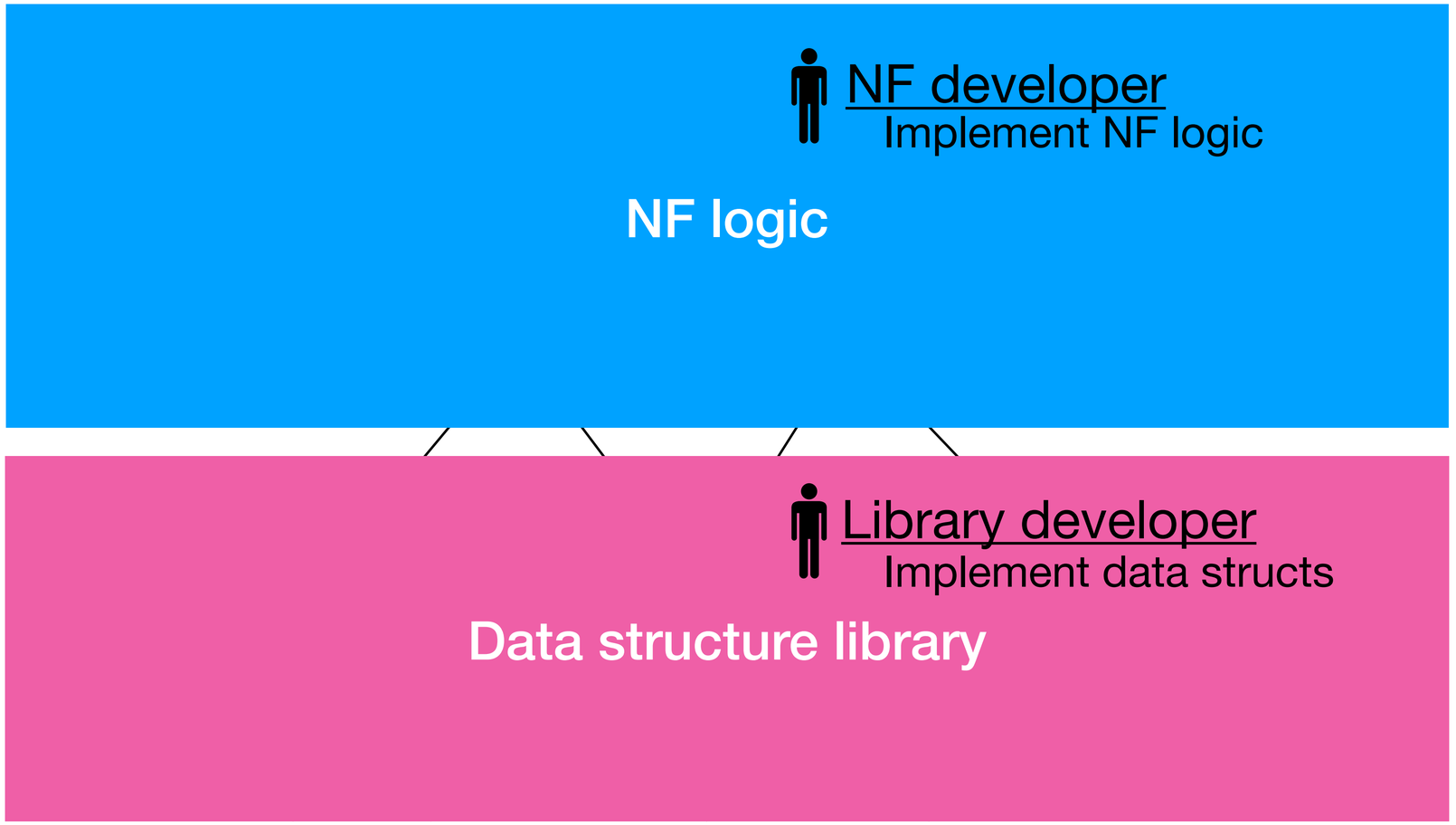
vigor.epfl.ch

# Network Function (NF) is the core of a network middlebox

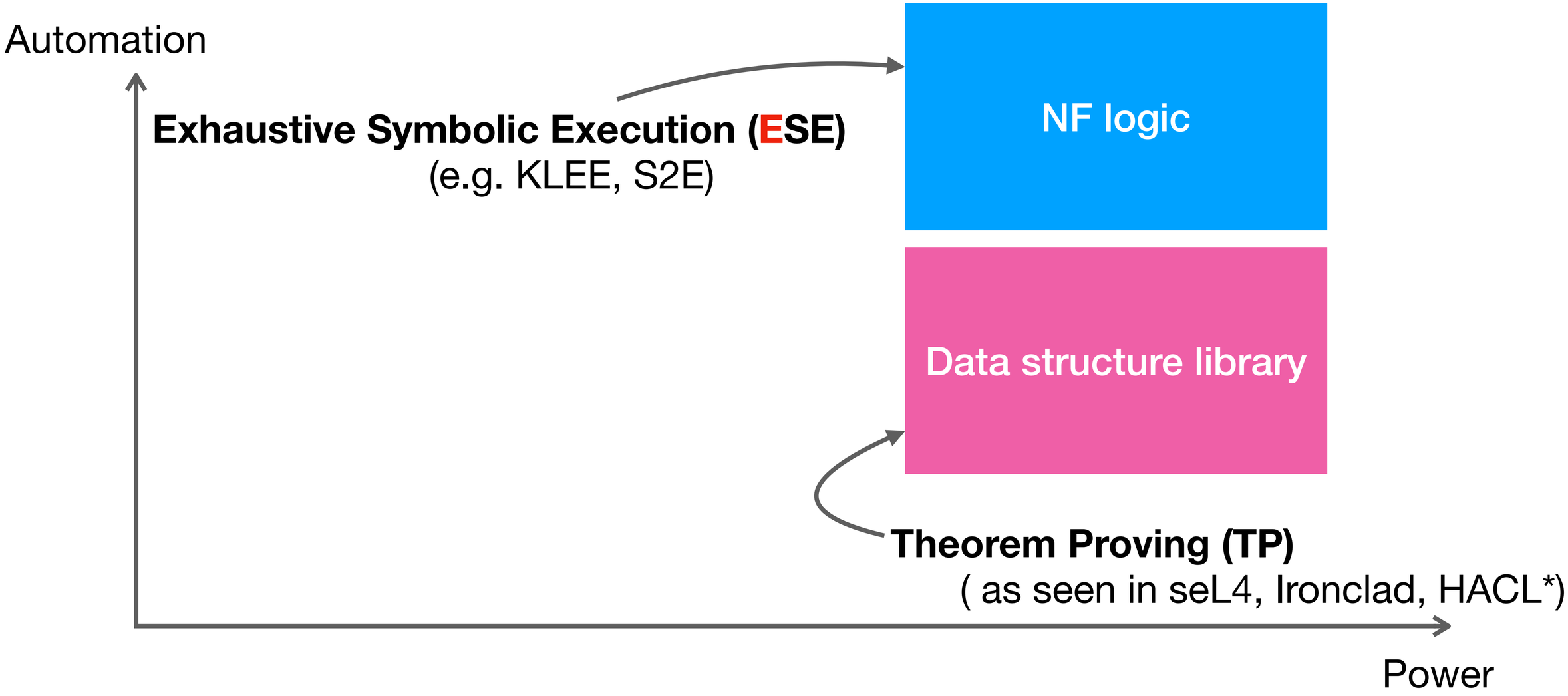
# NF in a Nutshell



# NF = NF Logic + Data Structs



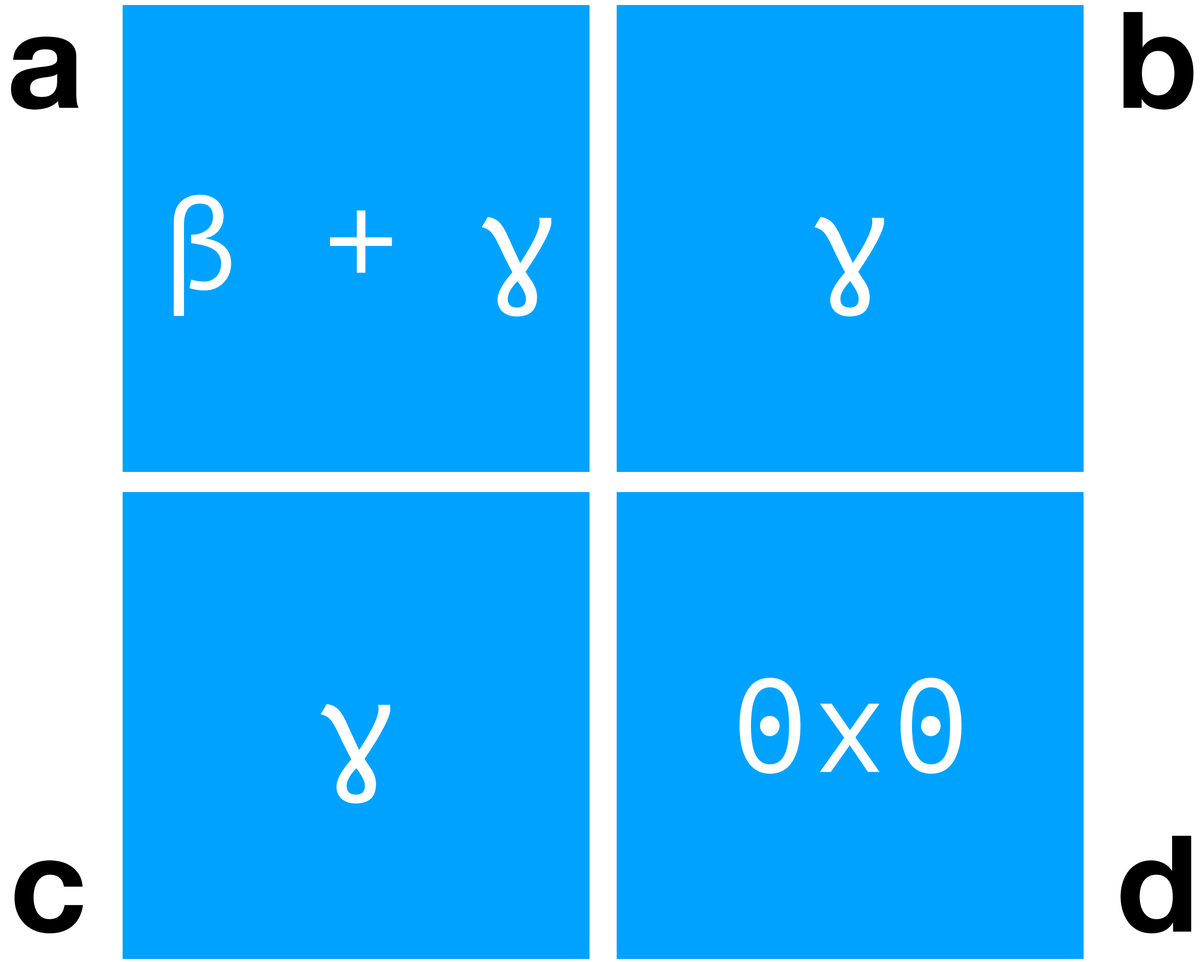
# 2 Verification Approaches



# SE = Low-Level & Automatic

**a <- b + c**

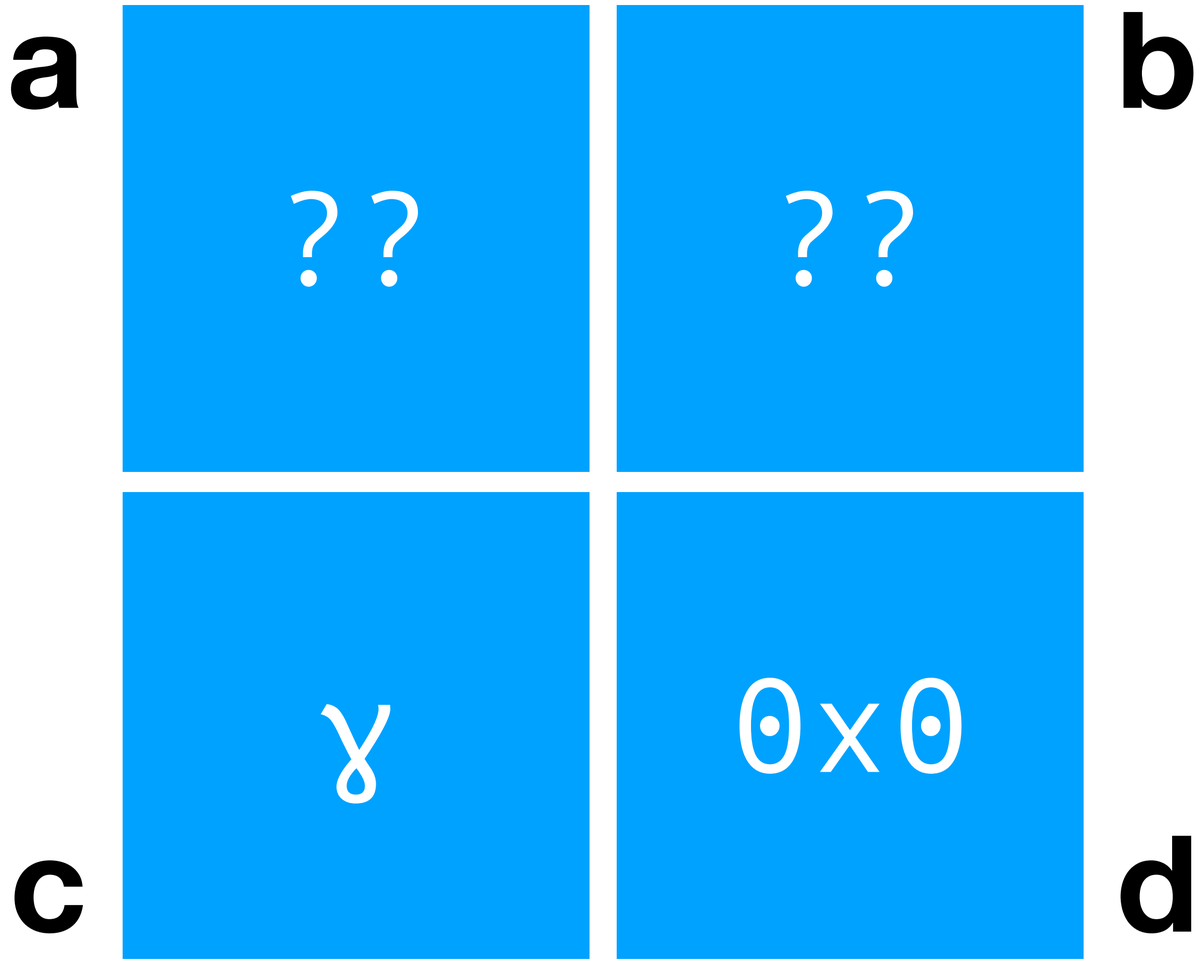
**→ b <- c + d**



# TP = High-Level & Manual

a <- *b

→ b <- *(b + 1)



# TP = High-Level & Manual

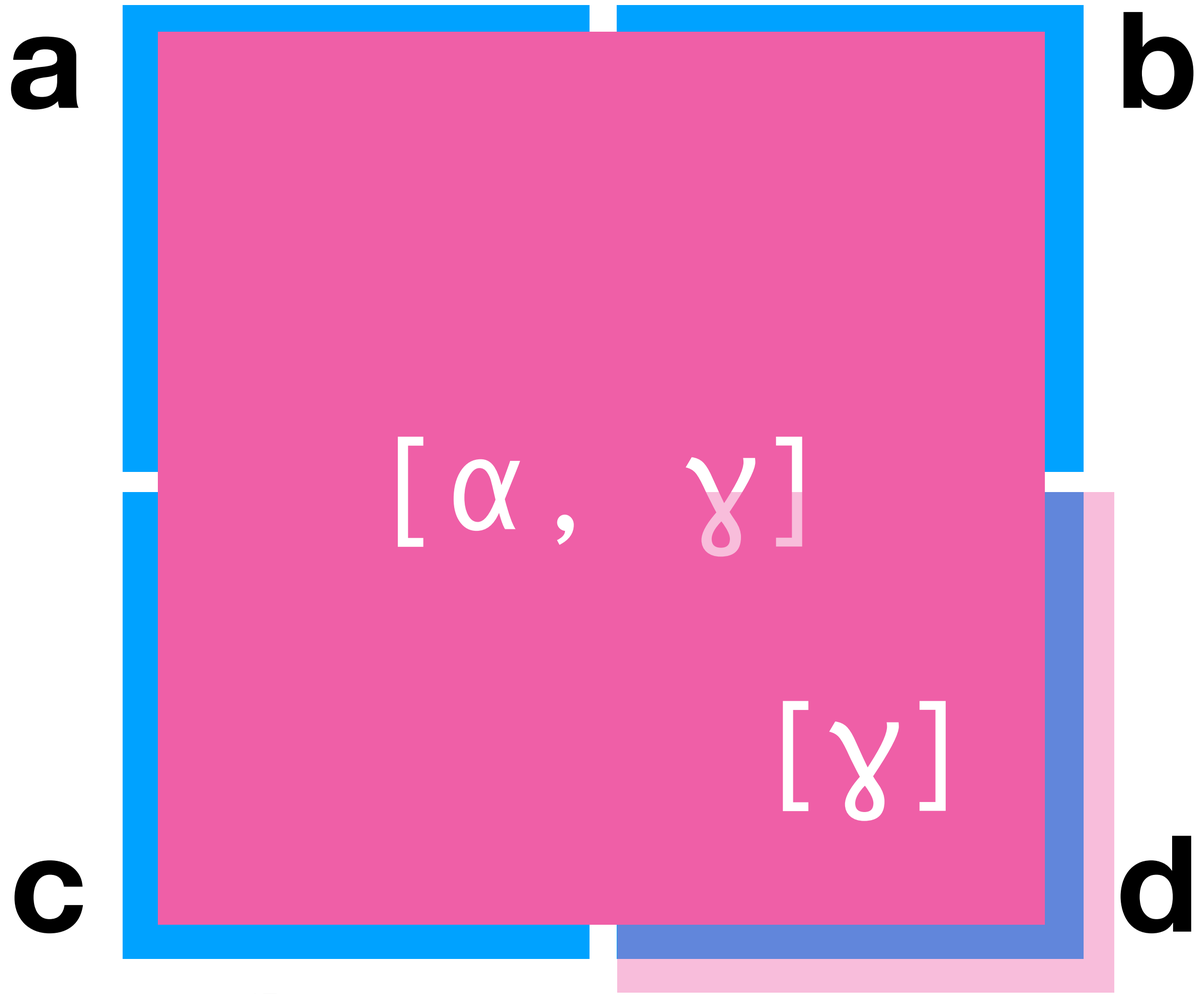
/*@ a - head, b - tail @*/

**a <- *b**

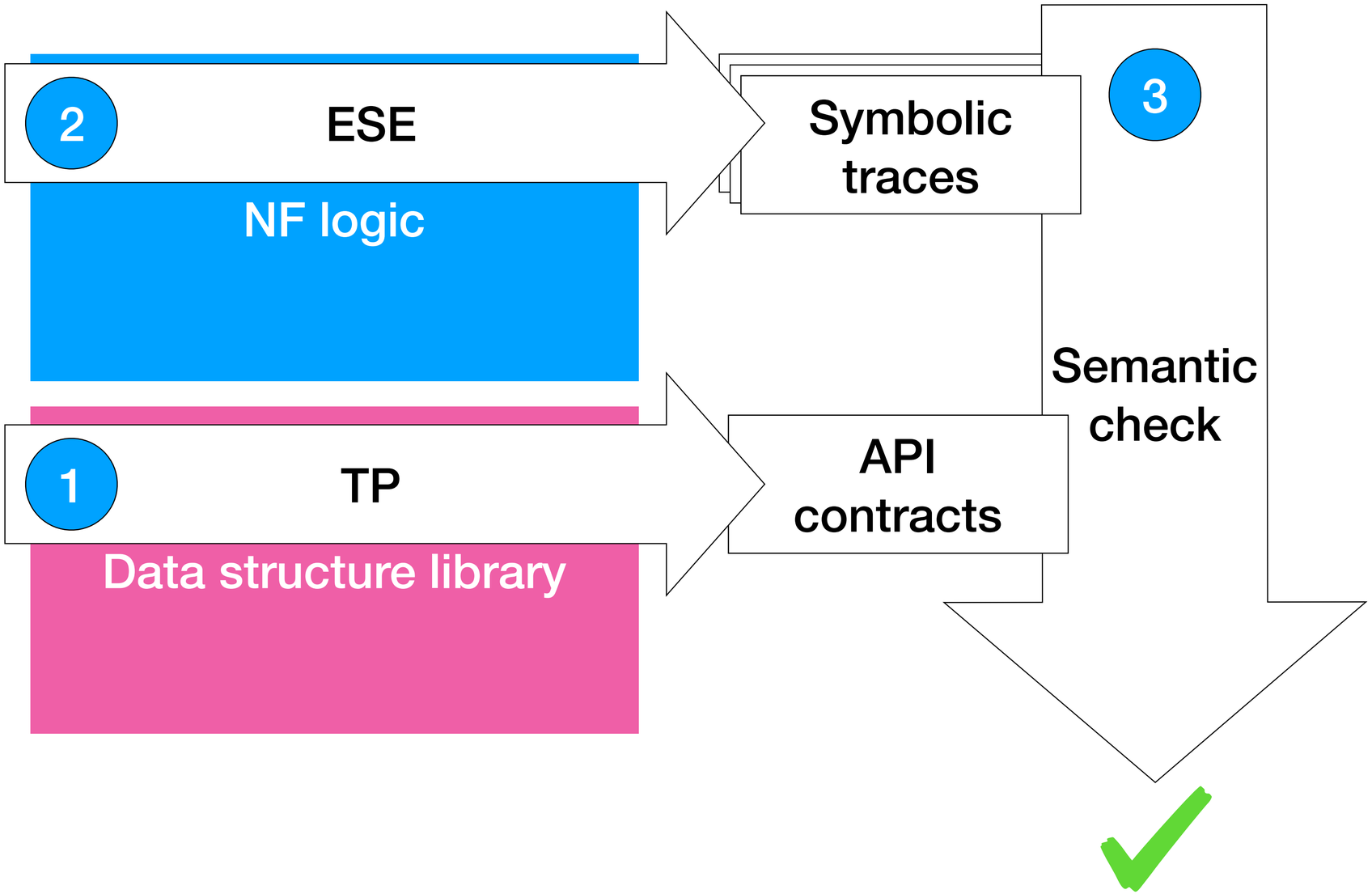
/*@ This is replacing
the list head @*/

→ **b <- *(b + 1)**

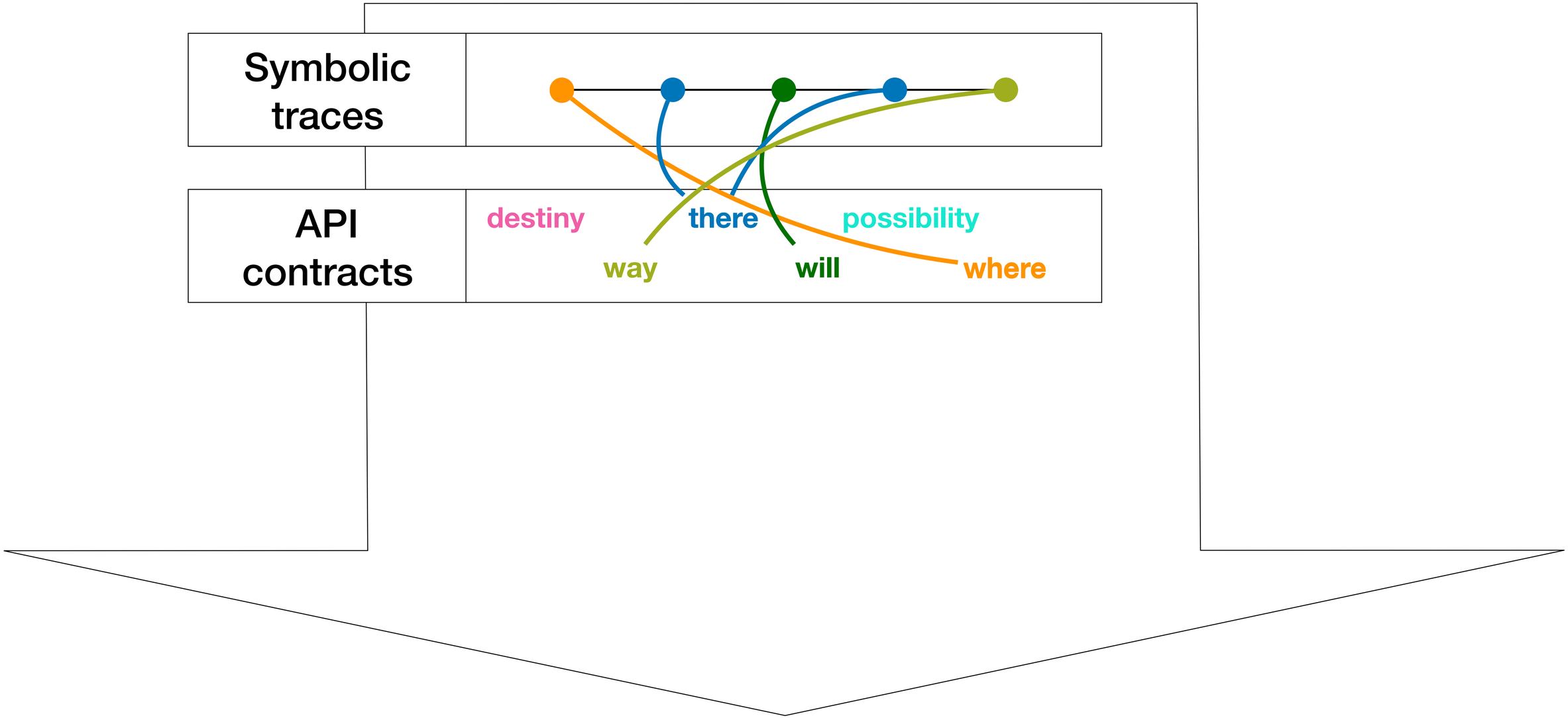
/*@ This is replacing
the list tail @*/



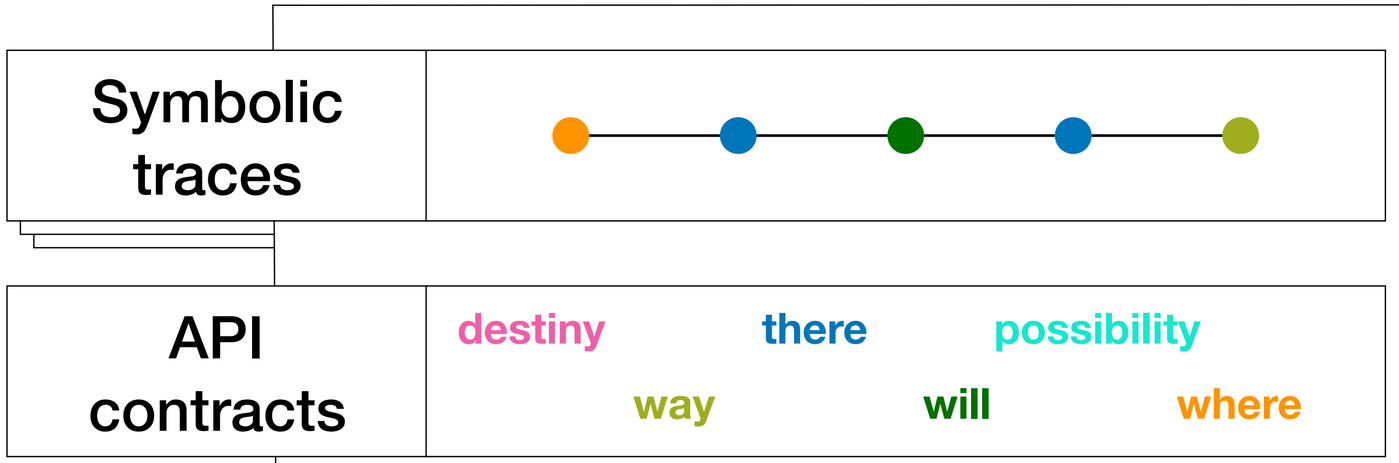
# Verification Steps



# Semantic Check

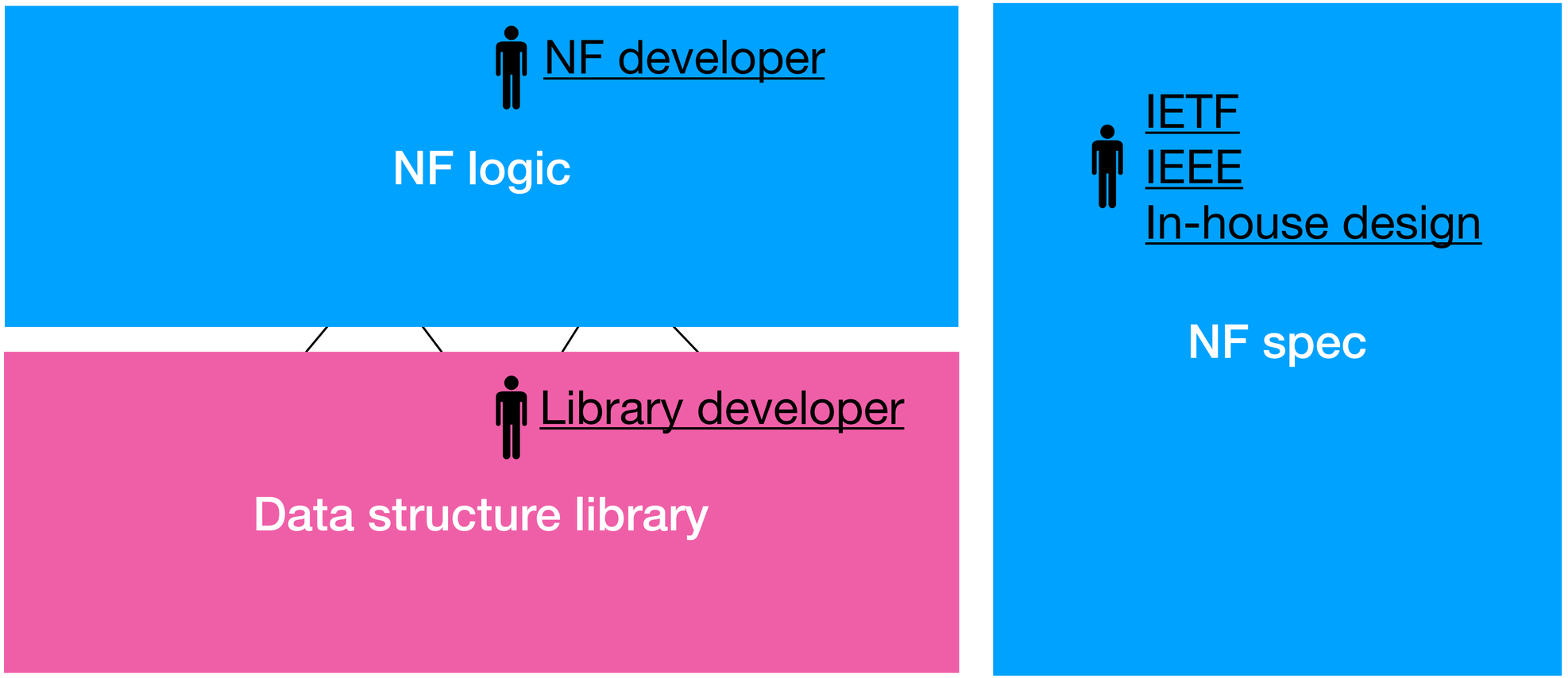


# Semantic Check




**Where there's a will, there's a way.**

# Who Writes The Specs



# Specification Abstraction

**Where there’s a will, there’s a way.**

Vouloir, c’est pouvoir.

Wo ein Wille ist, ist auch ein Weg.

Было бы желание, а способ найдётся.

# Common Abstractions

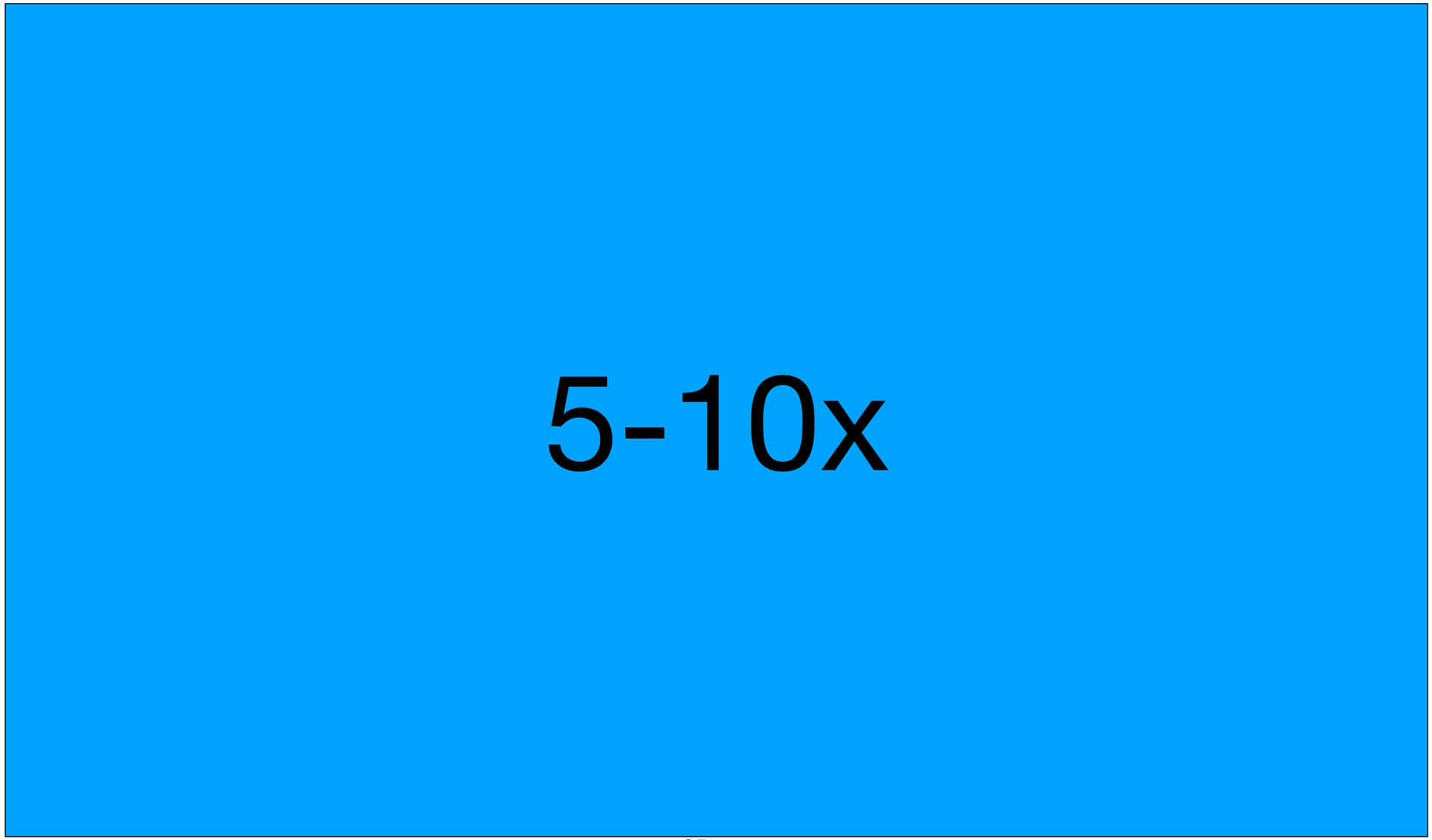
| Party | Abstractions used |
|---|---|
| IETF (NAT, Firewall) | same data structs |
| IEEE (bridge) | same data structs |
| In-house design (Maglev) | same data structs |
| Developers | same data structs |

**Common abstractions do exist: NF data structures**

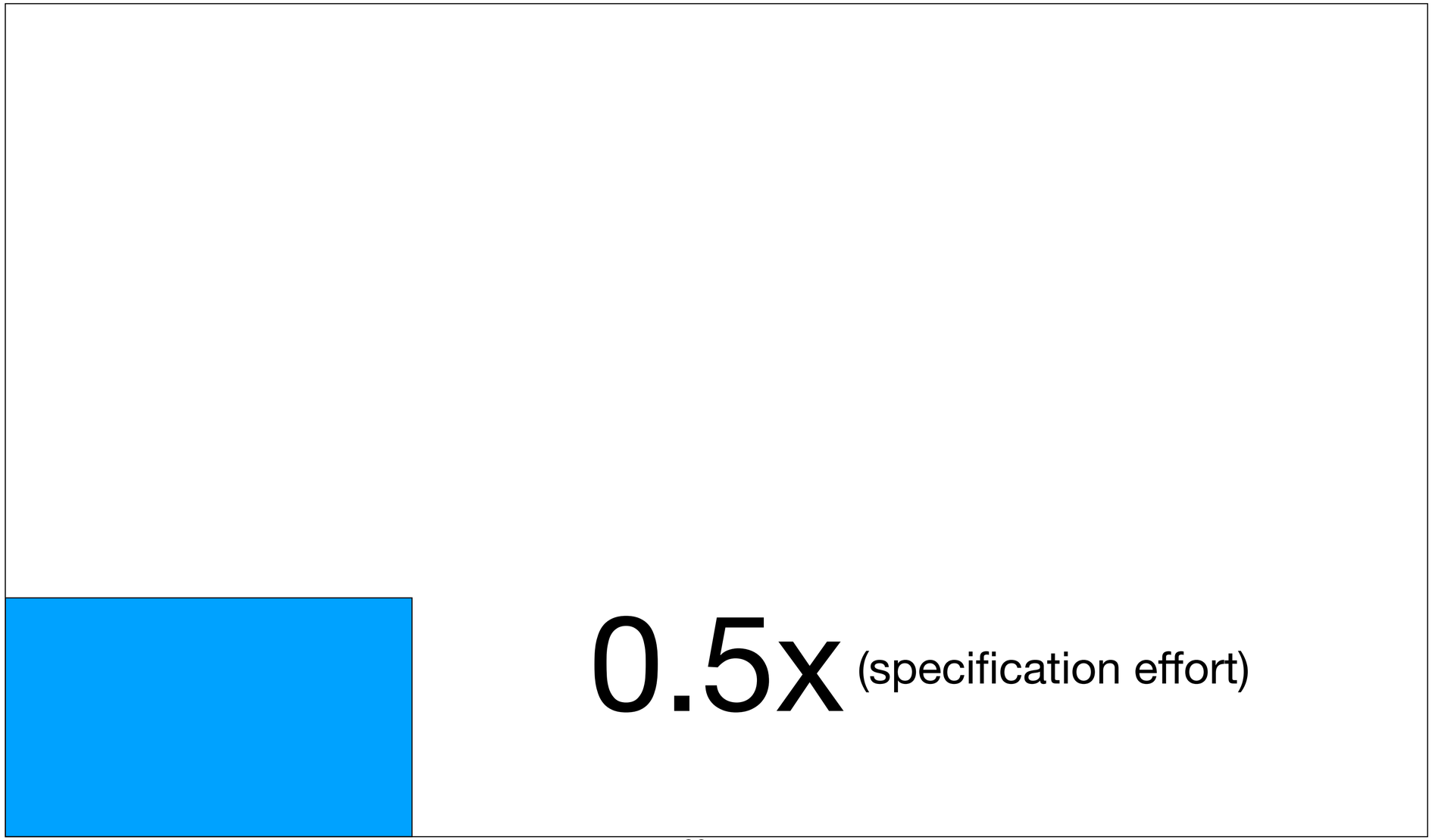
# Insight

NF implementations and specifications use the same set of abstractions

# Verification Effort



# Verification Effort



# Outline

Push-button

Pay-as-you-go

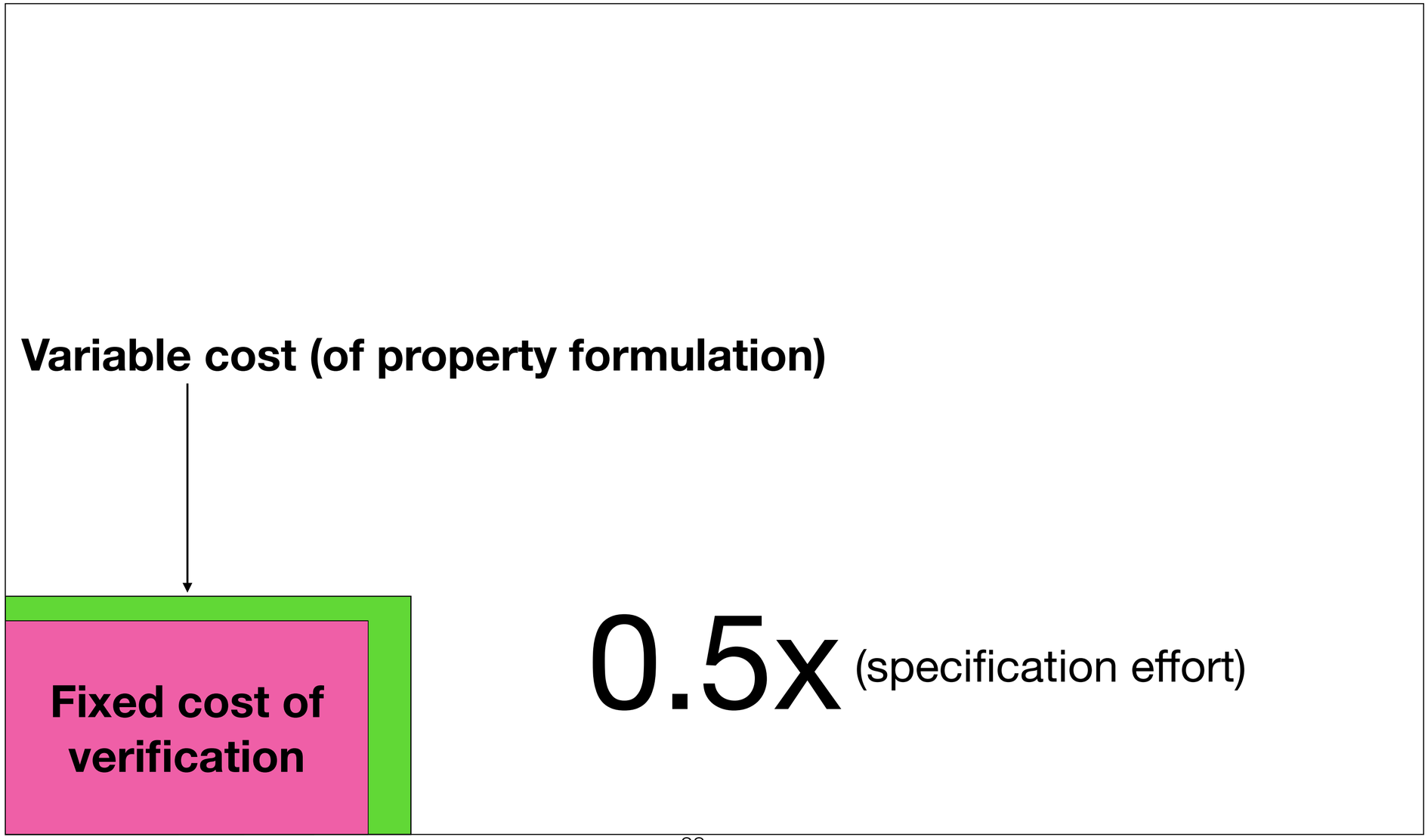
Usage scenario

Full-stack

Evaluation [ Practical | No performance overhead ]

vigor.epfl.ch

# Verification Effort



# Verification Effort

# Specification Example

```
from state import flow_emap, int_devices
EXP_TIME = 10 * 1000
EXT_DEVICE = 1

if a_packet_received:
    flow_emap.expire_all(now - EXP_TIME)

h3 = pop_header(tcpudp, on_mismatch=([],[]))
h2 = pop_header(ipv4, on_mismatch=([],[]))
h1 = pop_header(ether, on_mismatch=([],[]))

if received_on_port == EXT_DEVICE:
    internal_flow = FlowIdc(h3.dst_port, h3.src_port, h2.daddr, h2.saddr, h2.npid)
    if flow_emap.has(internal_flow):
        fl_id = flow_emap.get(internal_flow)
        flow_emap.refresh_idx(fl_id, now)
        out_port = vector_get(int_devices, fl_id)
        return ([out_port],[ether(h1, saddr=..., daddr=...), ipv4(h2, cksum=...), tcpudp(h3)])
    else:
        return ([],[])
else:
    internal_flow = FlowIdc(h3.src_port, h3.dst_port, h2.saddr, h2.daddr, h2.npid)
    if flow_emap.has(internal_flow):
        fl_id = flow_emap.get(internal_flow)
        flow_emap.refresh_idx(fl_id, now)
    else:
        if not flow_emap.full():
            fl_id = the_index_allocated
            flow_emap.add(internal_flow, fl_id, now)
            vector_set(int_devices, fl_id, received_on_port)
    return ([EXT_DEVICE],[ether(h1, saddr=..., daddr=...), ipv4(h2, cksum=...), tcpudp(h3)])
```

# One-Off Property

```
1  if received_on_port == EXT_DEVICE and flow_emap.has(internal_flow):
2      fl_id = flow_emap.get(internal_flow)
3      flow_emap.refresh_idx(fl_id, now)
4      out_port = vector_get(int_devices, fl_id)
5      return ([out_port],[ether(h1, saddr=..., daddr=...), ipv4(h2, cksum=...), tcpudp(h3)])
6  else:
7      pass
```

# Pay-as-you-go: ‘pass’

```
1  if received_on_port == EXT_DEVICE and flow_emap.has(internal_flow):
2      fl_id = flow_emap.get(internal_flow)
3      flow_emap.refresh_idx(fl_id, now)
4      out_port = vector_get(int_devices, fl_id)
5      return ([out_port],[ether(h1, saddr=..., daddr=...), ipv4(h2, cksum=...), tcpudp(h3)])
6  else:
7      pass
```

# Pay-as-you-go: '...'(ellipsis)

```
1  if received_on_port == EXT_DEVICE and flow_emap.has(internal_flow):
2      fl_id = flow_emap.get(internal_flow)
3      flow_emap.refresh_idx(fl_id, now)
4      out_port = vector_get(int_devices, fl_id)
5      return ([out_port],[ether(h1, saddr=..., daddr=...), ipv4(h2, cksum=...), tcpudp(h3)])
6  else:
7      pass
```

# Outline

Push-button

Pay-as-you-go

Usage scenario

Full-stack

Evaluation [ Practical | No performance overhead ]

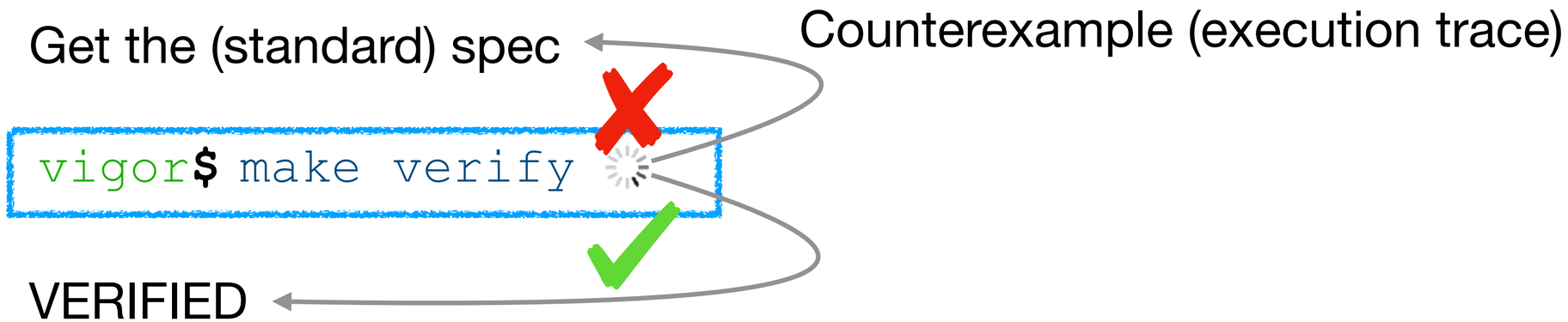
vigor.epfl.ch

# Usage Scenario

1. Implement, using Vigor library
2. Get the (standard) spec
3. ???
4. VERIFIED

# Usage Scenario

1. Implement, using Vigor library
2. Get the (standard) spec
3. `vigor$ make verify`
4. VERIFIED



# Outline

Push-button

Pay-as-you-go

Usage scenario

Full-stack

Evaluation [ Practical | No performance overhead ]

vigor.epfl.ch

# Recap: Insights and Design

Observations:

- NFs naturally split into NF logic + data structures
- NF specs and code use common abstractions
- Writing full specs is costly
- Most code in SW stack has no impact on desired properties

How Vigor leverages them:

- Verify different parts with different techniques, then stitch
- Adopt those abstractions for the Vigor library API and contracts
- Support incremental specs
- Exclude irrelevant code and verify the rest

# Outline

Push-button

Pay-as-you-go

Usage scenario

Full-stack

Evaluation [ Practical | No performance overhead ]

vigor.epfl.ch

# Vigor NFs

- NAT
- Load balancer (Maglev)
- Traffic policer
- MAC-learning bridge
- Firewall

# Verify On Every Commit

| NF | NF-only | | Full-stack | |
|---|---|---|---|---|
| NAT | 7 sec + | 54 × 88 sec | 8 min + | 488 × 88 sec |
| Load Balancer | 23 sec + | 146 × 219 sec | 26 min + | 1,336 × 219 sec |
| Traffic Policer | 14 sec + | 37 × 82 sec | 6 min + | 309 × 82 sec |
| Bridge | 7 sec + | 69 × 80 sec | 10 min + | 601 × 80 sec |
| Firewall | 6 sec + | 43 × 88 sec | 7 min + | 369 × 88 sec |

- Verify NF in minutes
- Verify Full-stack in hours
- Embarrassingly parallel
- Found bugs in:
  DPDK pkt proc framework
  Intel NIC driver
  Our NFs

# Concise and incremental specs

- **1 day** to formulate a full spec
- **30-60 LOC** size of a full spec
- **1-13 LOC** size of a minimal property
- **4-9 LOC** size of a property increment

# Outline

Push-button

Pay-as-you-go

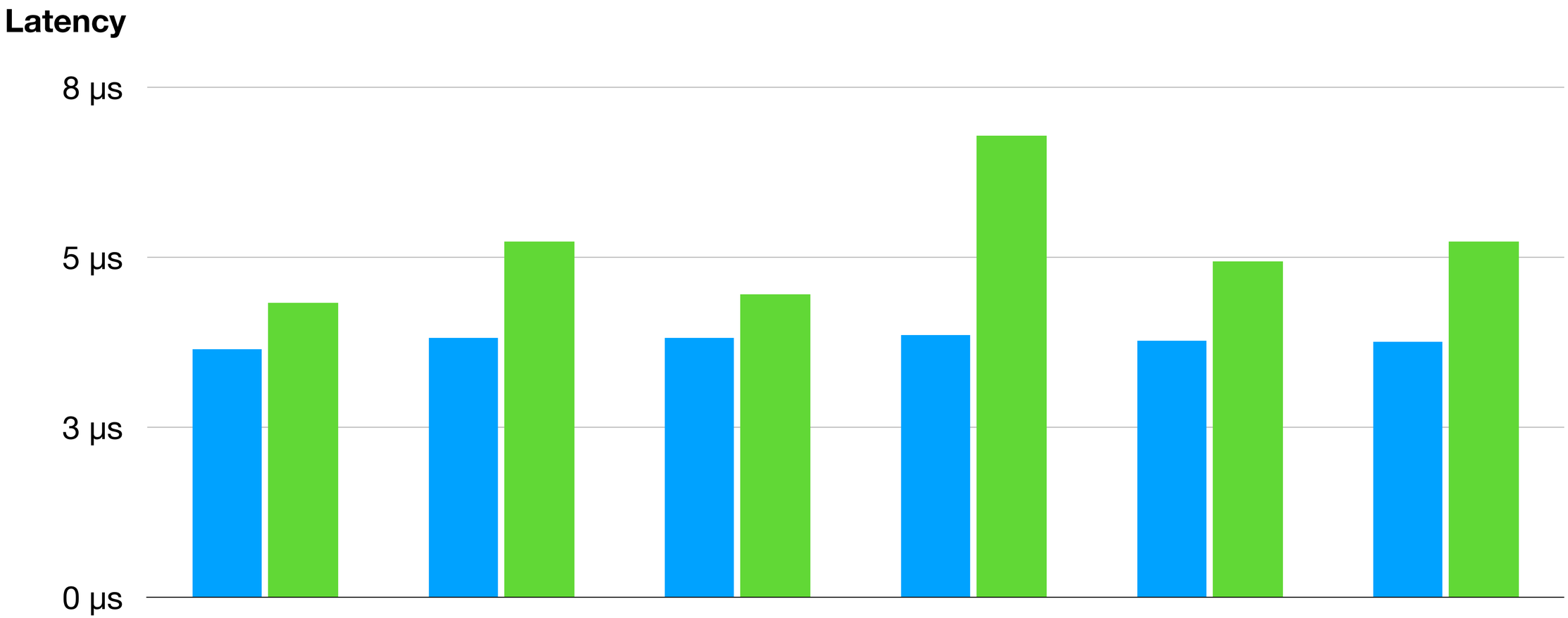
Usage scenario

Full-stack

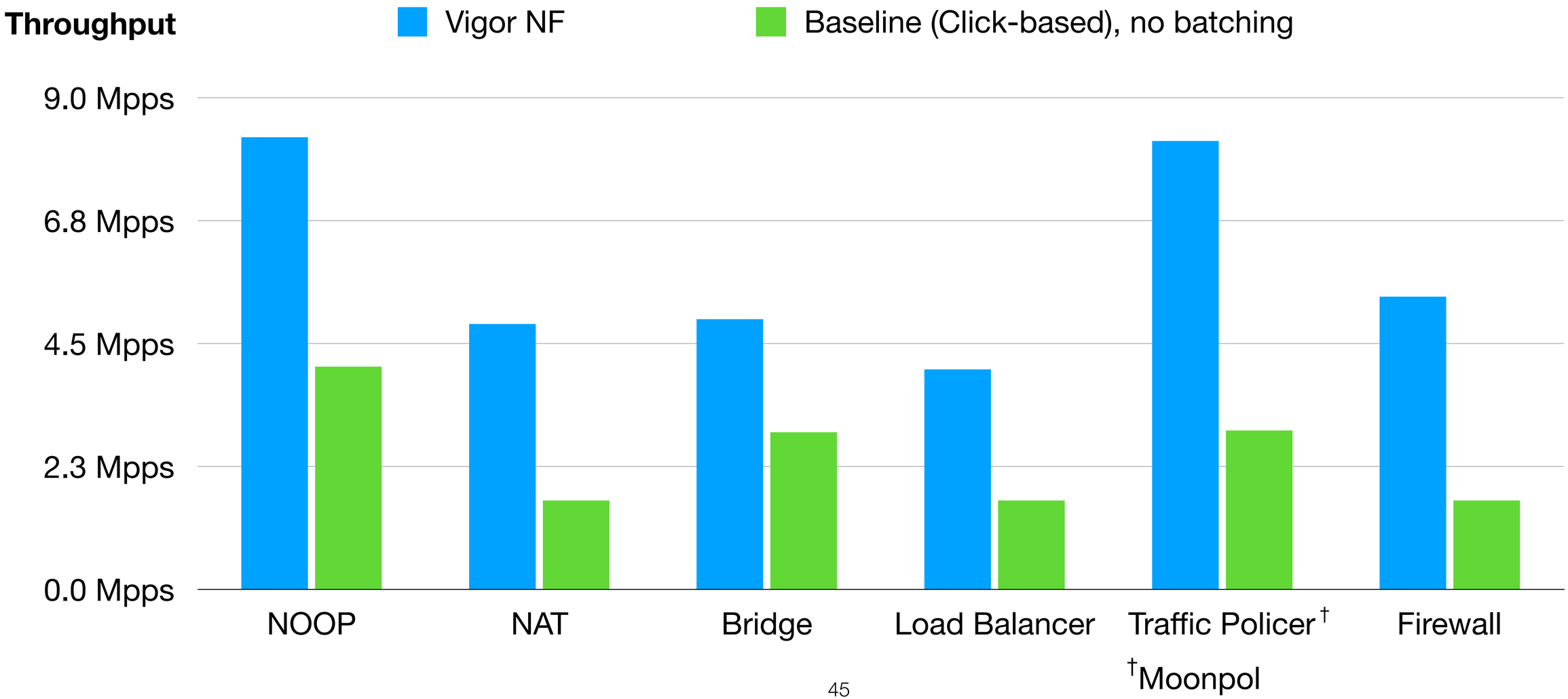
Evaluation [ Practical | No performance overhead ]

vigor.epfl.ch

# Latency is Competitive



# Throughput is Competitive



# Recap: Eval

- Fast: can be done on every commit
- Easy: properties take just a few lines
- No performance overhead (modulo batching support)

# Vigor enables NF developers to verify the Full stack of their NFs in a Pay-as-you-go manner with no verification expertise or effort ( Push-button )

Vigor enables practical NF verification

Get it online! vigor.epfl.ch