RTLola: Specifying Real-time Monitors for Unmanned Aircraft Systems

Bernd Finkbeiner
CISPA Helmholtz Center for Information Security

CISPA
HELMHOLTZ CENTER FOR
INFORMATION SECURITY

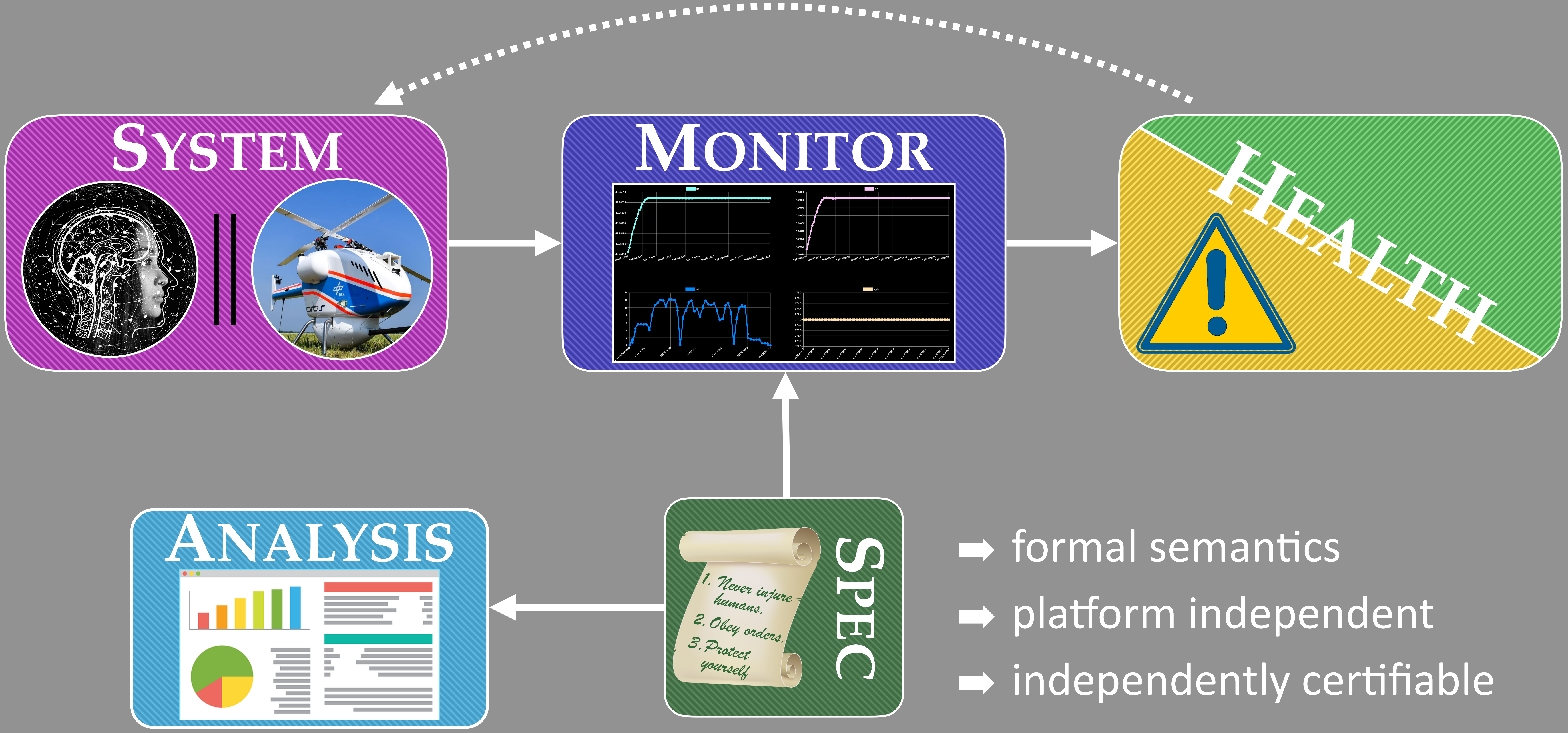
NSF-PIRE Workshop on Assured CPS Autonomy for 3D Urban Transportation
June 9, 2021

Monitor real-time properties

over rich data streams

with limited resources

➡ Since 2015 collaboration with German Aerospace Center (DLR)
➡ Since 2020 VoloStream project in collaboration with VOLOCOPTER
➡ Open source release (RTLola interpreter, VHDL compiler)



SYSTEM

MONITOR

HEALTH

ANALYSIS

SPEC

- ➡ formal semantics
- ➡ platform independent
- ➡ independently certifiable

RTLola

➡ Expressiveness of a general-purpose language

➡ Formal semantics and strong guarantees of a logic

➡ Highly efficient

Formal semantics
Guarantees

Expressiveness

Logics (LTL, STL, MTL, MFOTL ...)

General-purpose languages

Property Spectrum

	Sensor Level	Mission Level
Timeliness	critical	relaxed
Arithmetic difficulty	low (bounds checks, counting)	high (statistics, prediction)
Input data	raw	aggregated
Locality	local	system-wide
Example	Data Validation: *“Altimeter must produce positives values.”*	Mission Statistics: *“Low correlation between WP distance and relative path deviation.”*

Examples

Sensor Validation 1: *Altimeter readings must be non-negative.*

```
input altitude: Float32
trigger altitude < 0 “Altimeter reports negative values.”
```

Sensor Validation 2: *Barometer must produce 9 — 11 readings per second.*

```
input pressure: Float32
output read_ps @ 1Hz := pressure.aggregate(over: 1s, using: count)
trigger read_ps > 11 ∨ read_ps < 9 “Barometer count irregular.”
```

Examples

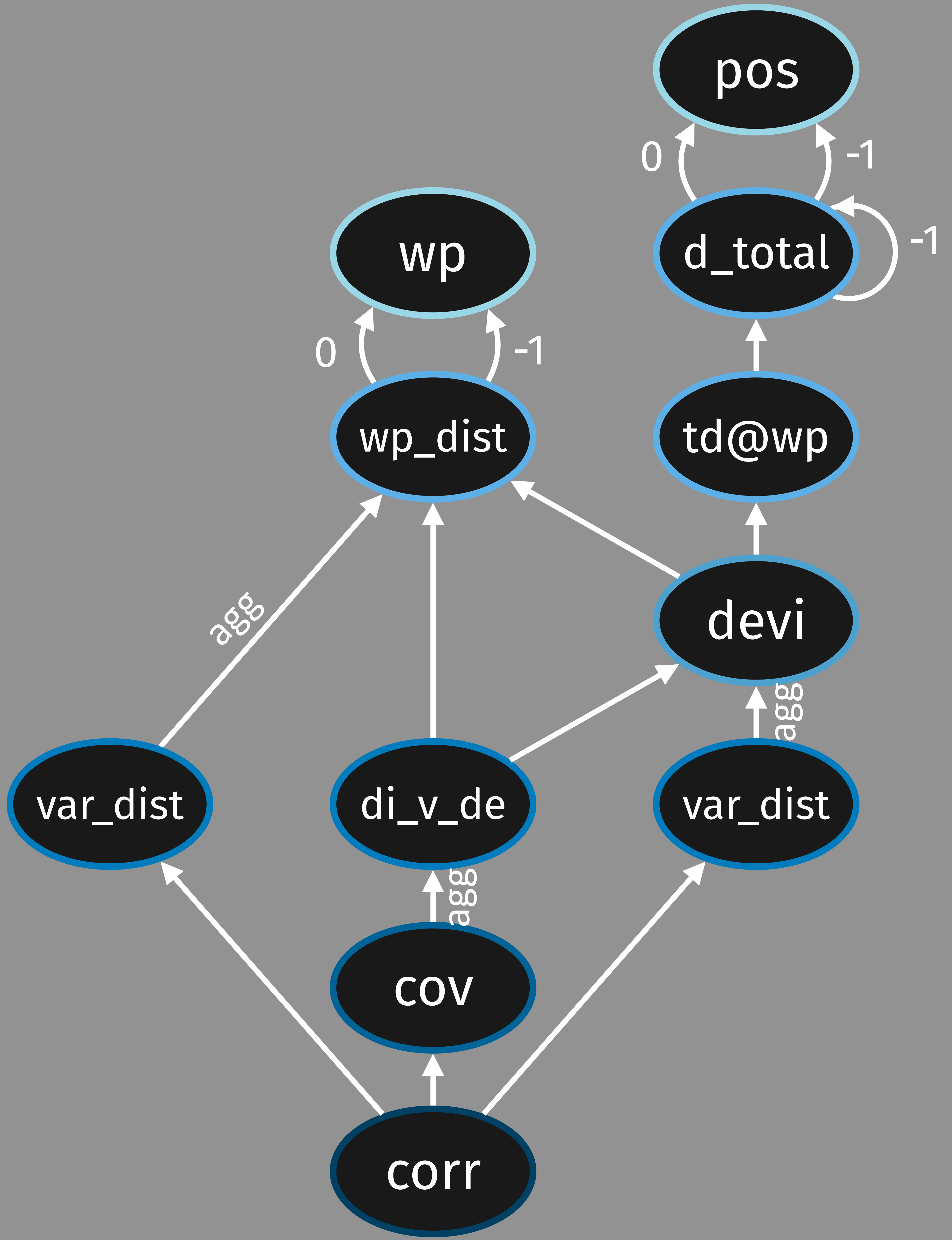
Mission Statistics: *Does the WP-distance correlate with the relative path deviation?*

```
input wp, pos: (Float64, Float64)

output wp_dist := abs(wp - wp.offset(by: -1, dft: wp))
output dist_total := pos - pos.offset(by: -1, dft: START)
                     + dist_total.offset(by: -1, dft: 0)
output total_dist_at_wp @ wp := dist_total.hold(or: 0)
output devi @ wp := abs( wp_dist.offset(by: -1, dft: 0) -
    (total_dist_at_wp - total_dist_at_wp.offset(by: -1, dft: 0) )

output dist_v_devi @ wp := (wp_dist, devi)
output cov @ 1Hz := dist_v_devi.aggregate(over: ∞, using: cov)
output var_dist @ 1Hz := wp_dist.aggregate(over: ∞, using: var)
output var_devi @ 1Hz := devi.aggregate(over: ∞, using: var)
output corr := cov / (var_devi^2 * var_dist^2)
```

Static Analysis

Stream	#values	Size	Windows	Total
pos	2	128		256
wp	2	128		256
d_total	2	64		128
wp_dist	1	64		64
d_s_wp	1	64		64
devi	1	64		64
var_dist	1	64	128	192
di_v_de	1	64		64
var_dist	1	64	128	192
cov	1	64	128	192
corr	1	64		64

Σ 1536B



Interpretation vs. Compilation

Observation

Compilation

Interpretation

```
Impl Monitor {
  while let Some(i)
      = get_input() {
  …
}}
```

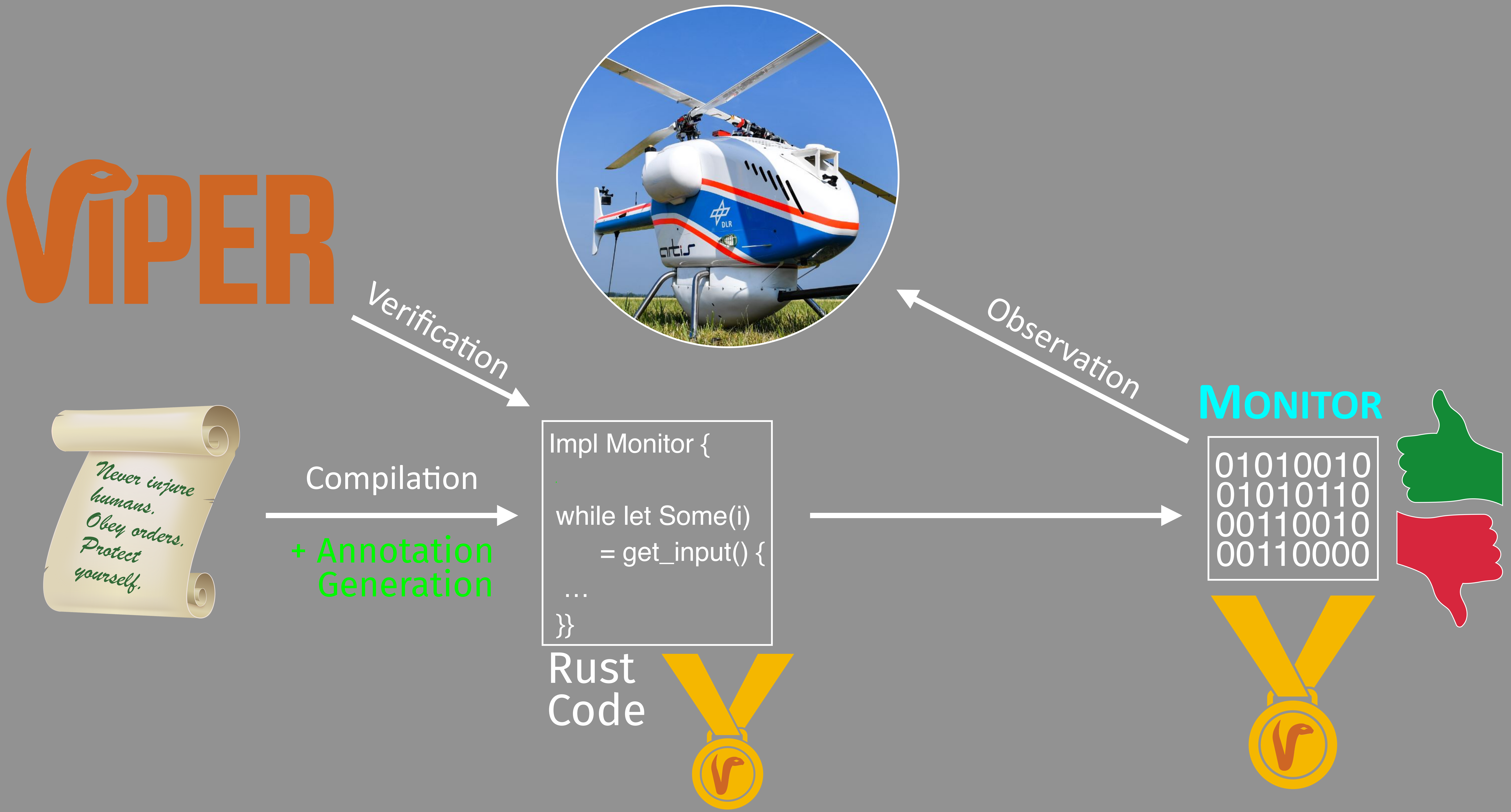
Rust/VHDL Code

MONITOR

```
01010010
01010110
00110010
00110000
```

Interpretation	438ns	1.535µs
Compilation	6ns (1.4%)	63ns (4%)

Verifying Compilation

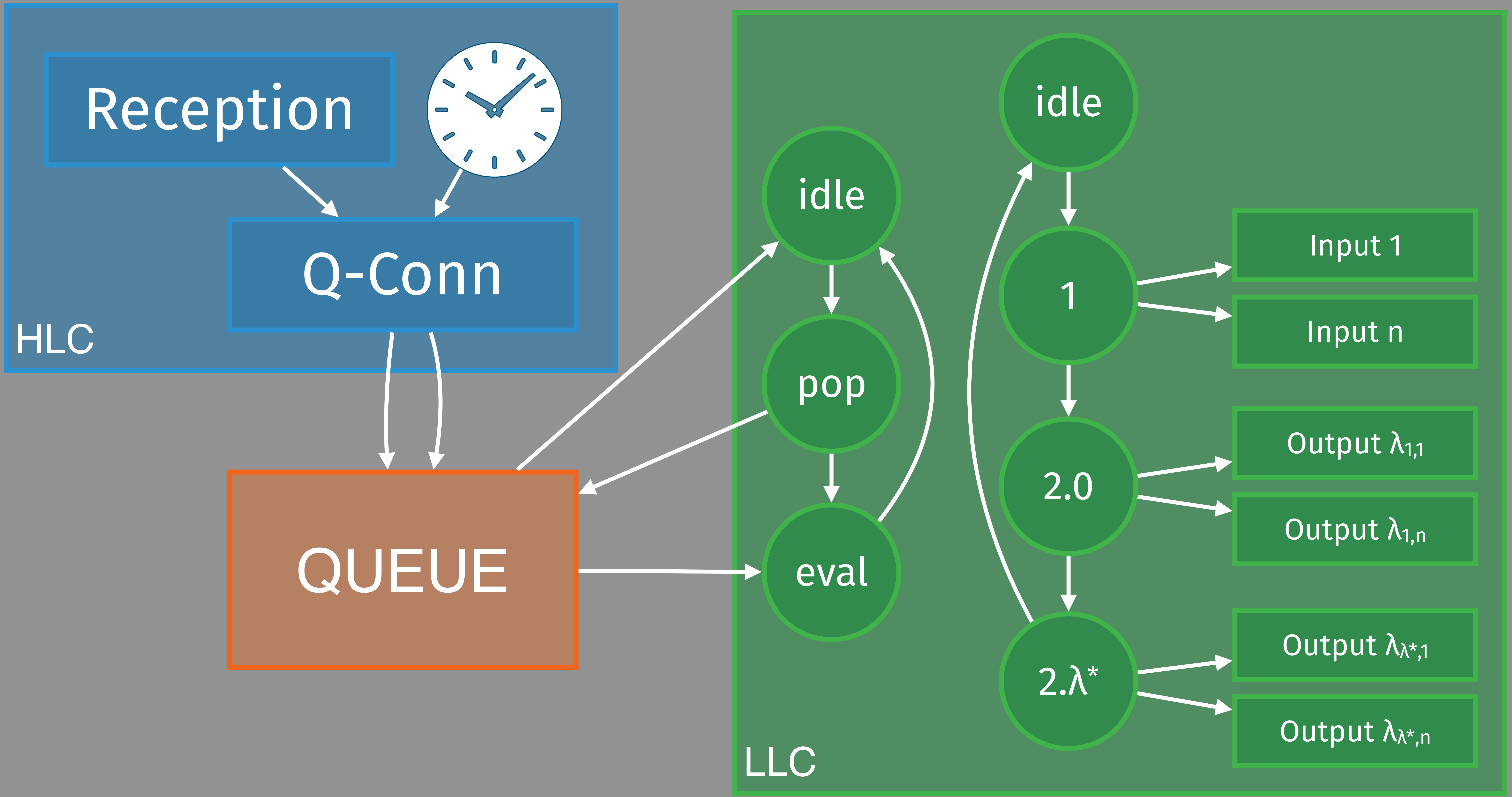


Never injure humans.
Obey orders.
Protect yourself.

```
Impl Monitor {

 while let Some(i)
       = get_input() {
  …
}}
```

```
01010010
01010110
00110010
00110000
```

Parallelization in Hardware



		FF	LUT	MUX	CA	MULT	Pwr [W]	Time [µs]
Cmd **Resp** **Parallel**	Mon	6379	13794	0	849	0	1,582	3,77
	HLC	936	232	0	30	0		
	Q	540	326	0	28	0		
	LLC	4903	13236	0	971	0		
Cmd **Resp** **Sequential**	Mon	6909	14768	0	851	0	1,581	43,83
	HLC	936	232	0	30	0		
	Q	534	326	0	28	0		
	LLC	5433	14210	0	973	0		

Real-time Monitoring with RTLola

➡ **Highly expressive + formal** specificaton language

➡ **Strong guarantees** (static analysis + verifying compiler)

➡ **Highly efficient** (HW/SW compilation, code optimization)

➡ **Maintainable** and **highly reusable** specs (stream structure)

The RTLola interpreter and VHDL compiler are available open source

www.rtlola.org