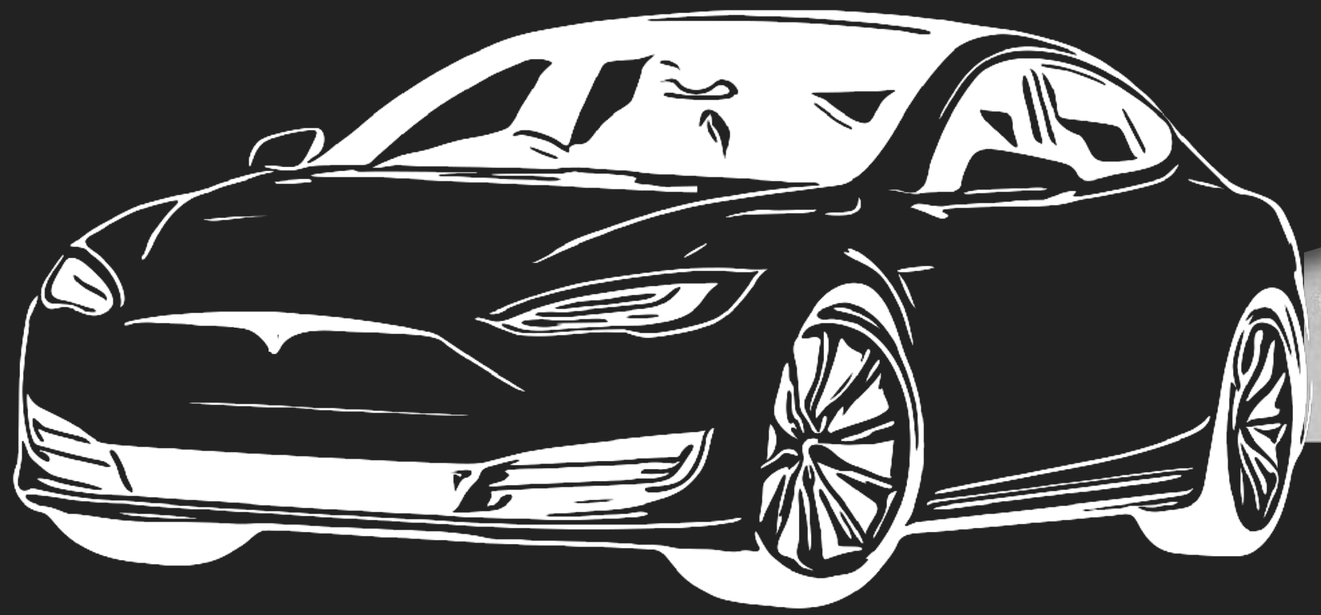
Nobody's Perfect: Monitoring Systems that Work *Most of the Time*

Maximilian Schwenger

Joint work with Jan Baumeister, Peter Faymonville, Bernd Finkbeiner, Malte Schledjewski, Marvin Stenger, Leander Tentrup, Hazem Torfah

Saarland
University

Why Should We Bother?



Our Vision

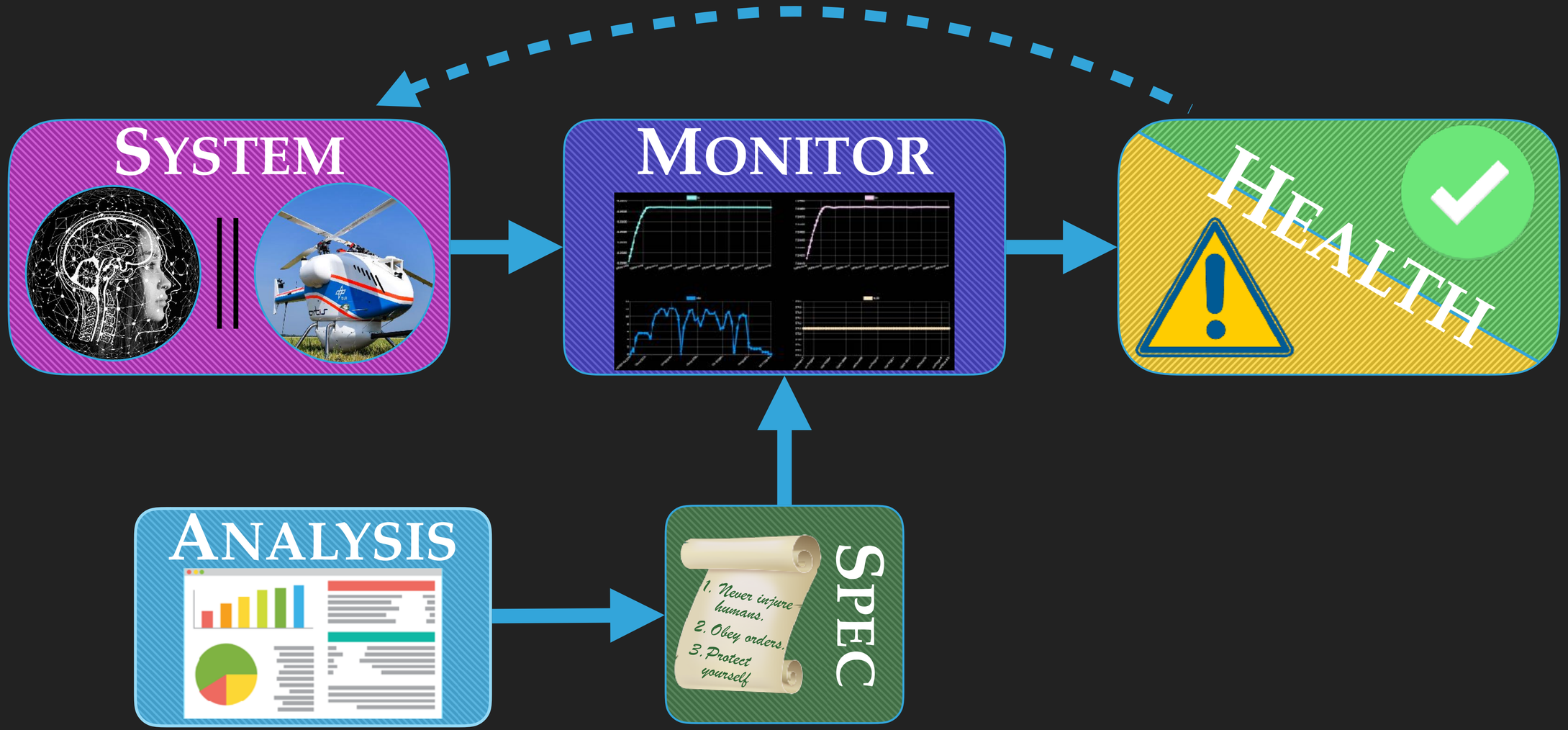
System

||

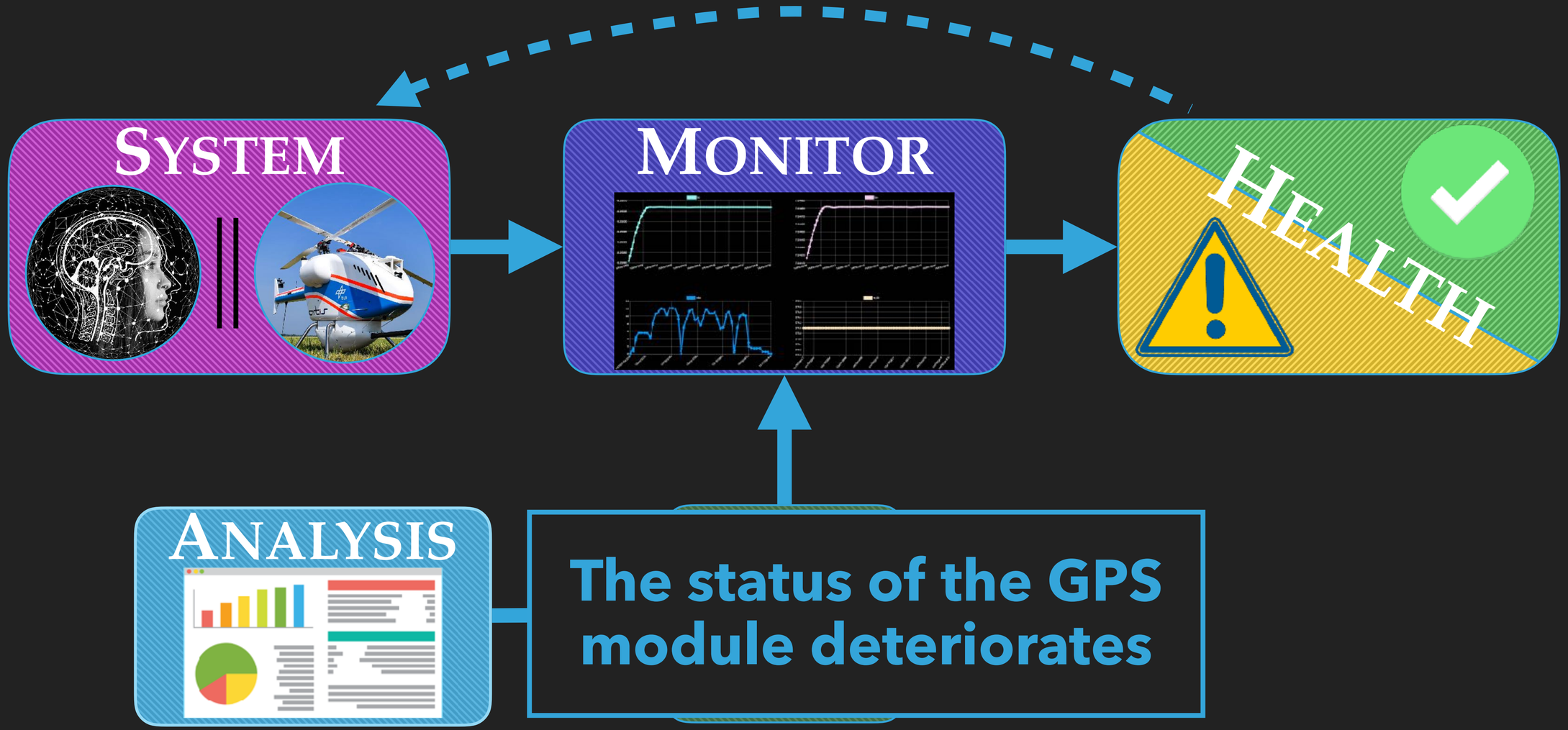
+

Formal Guarantees on Runtime Behavior

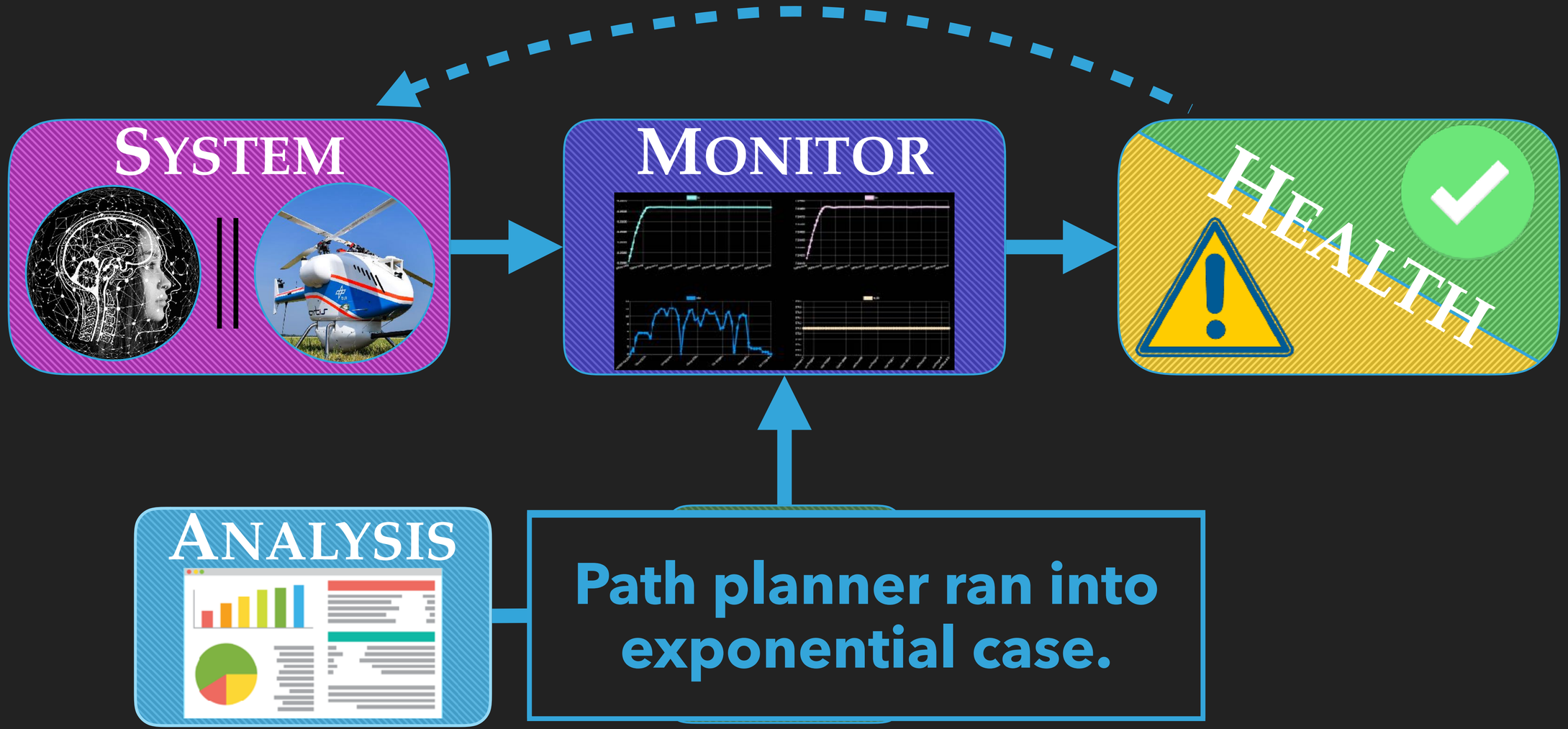
Our Take on Runtime Verification



Our Take on Runtime Verification



Our Take on Runtime Verification



Static Verification

System S

Controller C

Specification φ

Verify:

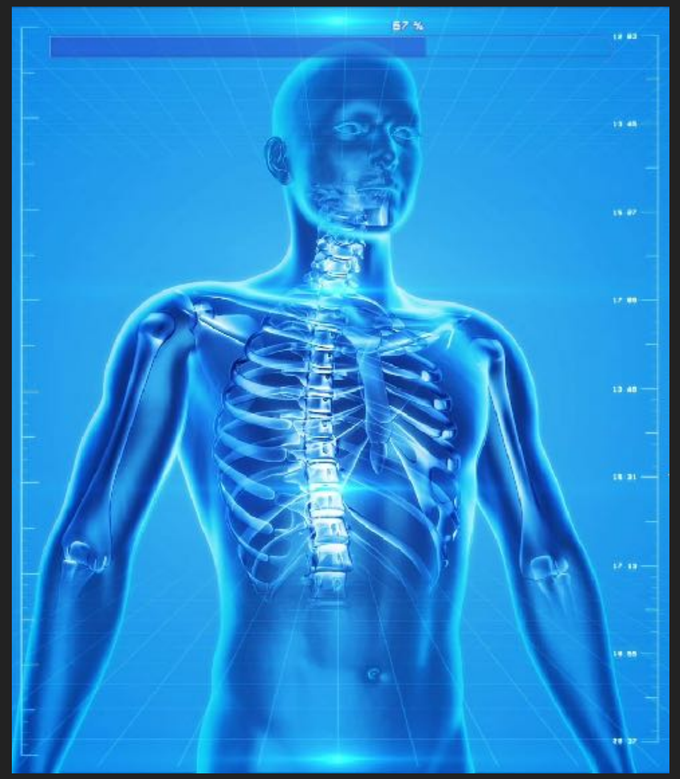
$$\forall \sigma \in \mathit{runs}(S \parallel C): \sigma \models \varphi$$

When Static Verification Fails

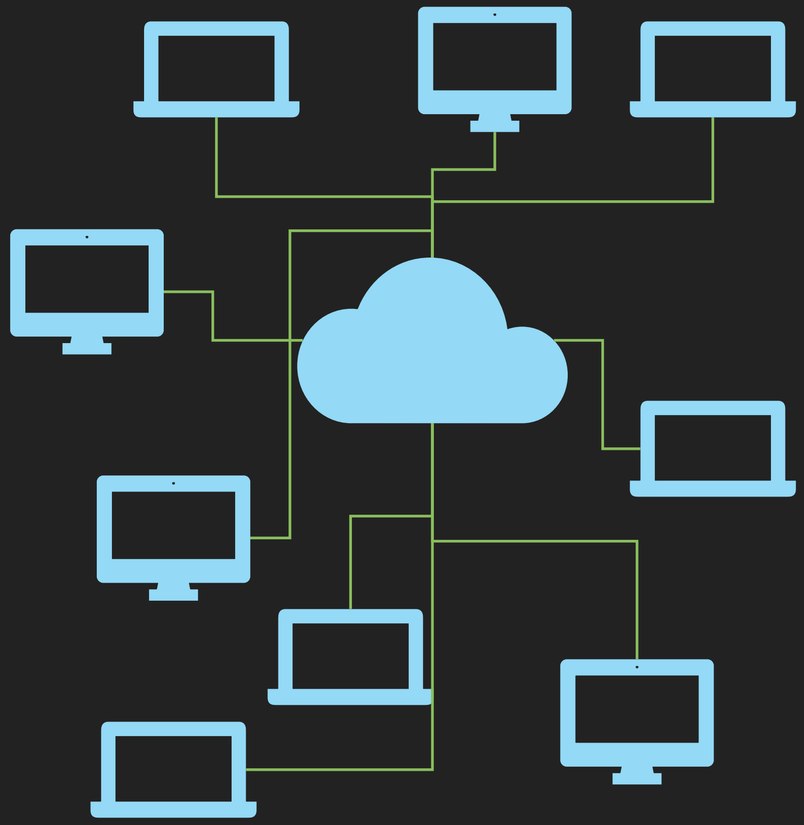
Scalability

$$\dot{p} = Rv$$
$$\dot{R} = R\hat{\omega}$$
$$\dot{v} = -\omega \times v + R^T \bar{g} + f_v(\omega, v, \alpha, \beta, \omega_r, \delta_c, \delta_r)$$
$$\dot{\omega} = -J^{-1}(\omega \times J\omega) + f_w(\omega, v, \alpha, \beta, \omega_r, \delta_c, \delta_r)$$
$$\dot{\alpha} = f_\alpha(\omega, v, \alpha, \beta, \omega_r, \delta_a, \delta_e)$$
$$\dot{\beta} = f_\beta(\omega, v, \alpha, \beta, \omega_r, \delta_a, \delta_e)$$
$$\dot{\omega}_r = f_r(\omega, v, \omega_r, \delta_c, \delta_r)$$

Model Dependency



Non-determinism



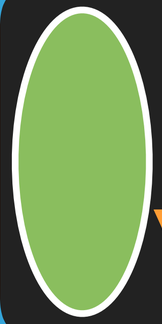
Static Verification

System S | Controller C | Specification φ

Verify:

$\forall \sigma \in \mathit{runs}(S \parallel C): \sigma \models \varphi$

Testing



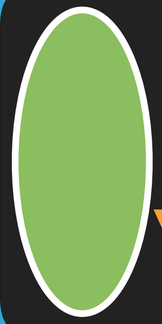
Verify:

$\forall \sigma \in \mathit{runs}(S \parallel C): \sigma \models \varphi$

$\exists S' \subseteq \mathit{runs}(S \parallel C):$

$\forall \sigma \in S': \sigma \models \varphi$

Runtime Verification



Verify:

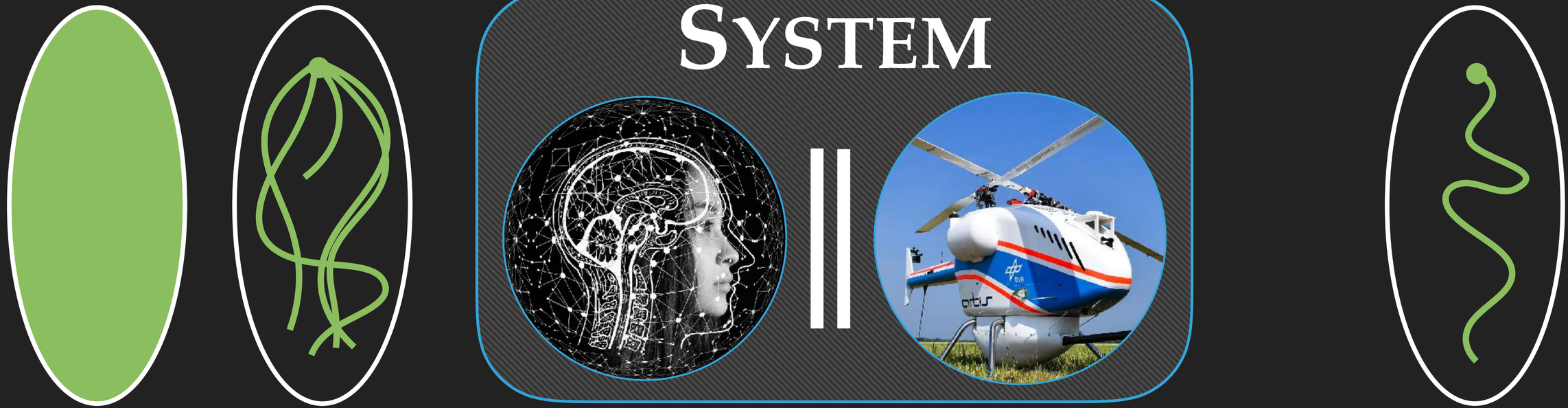
$\forall \sigma \in \mathit{runs}(S \parallel C): \sigma \models \varphi$

Given $\sigma \in \mathit{runs}(S \parallel C)$:

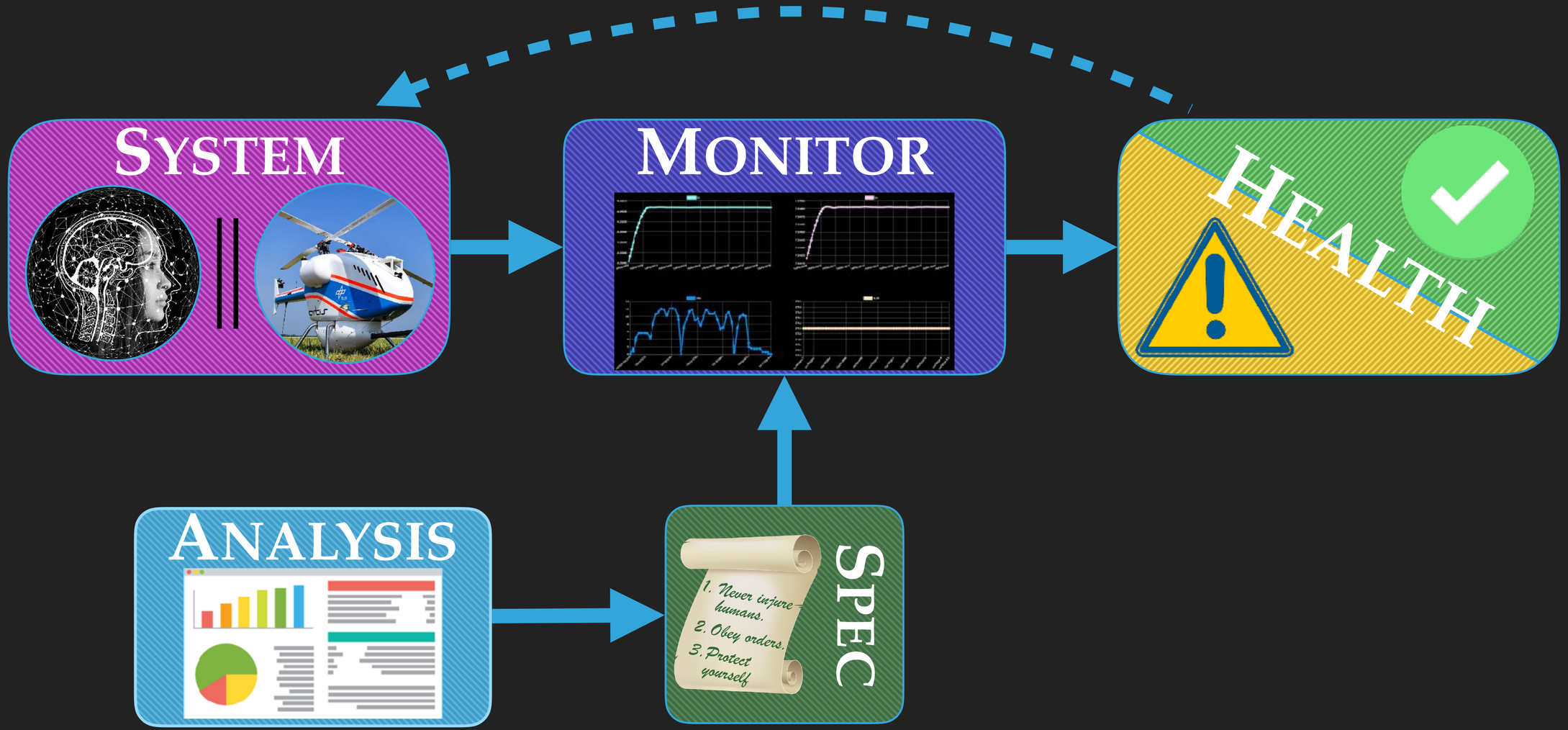
$\sigma \models \varphi$

Prior to Deployment

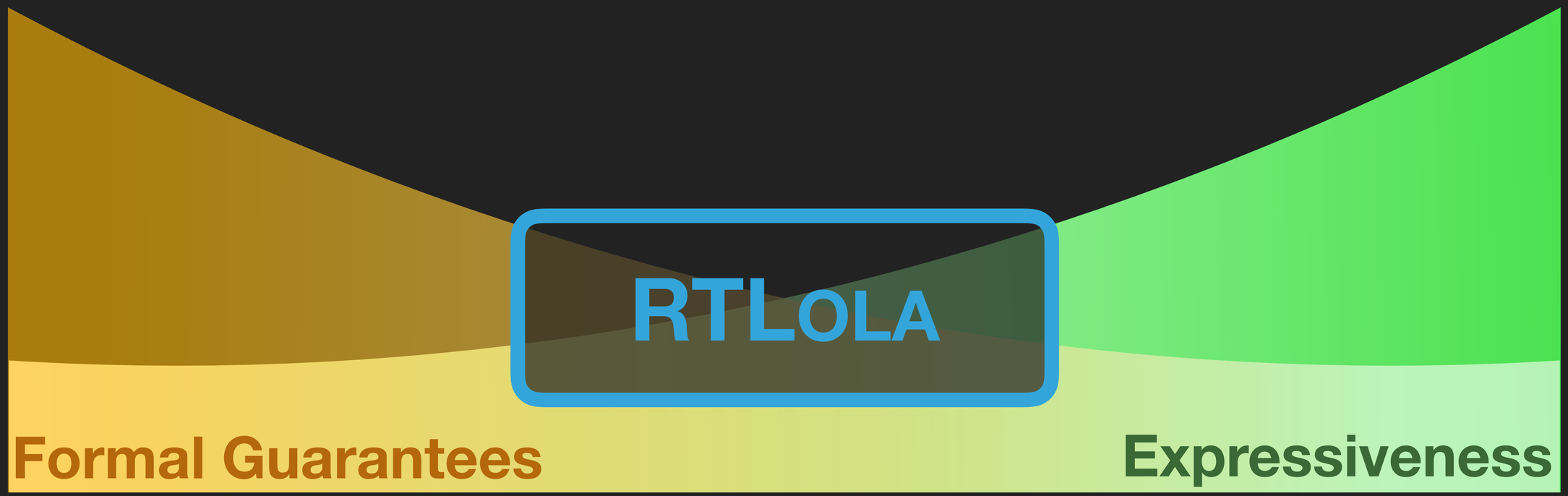
After Deployment



Our Take on Runtime Verification



Specification Languages



Stream-based Runtime Verification

Output Streams

Trigger

Trigger

Runtime Monitor

RTLola in a Nutshell

```
input lat, lon: Float64 // from GPS
input accel_x: Float64  // from accelerometer
input slow_down_cmd: Bool
```

The GPS module operates with at least 5Hz.

RTLola in a Nutshell

```
input lat, lon: Float64 // from GPS
input accel_x: Float64  // from accelerometer
input slow_down_cmd: Bool

output gps_samples @1Hz := lat.aggregate(over_exactly: 1s, using: count)
trigger gps_samples < 5 “GPS frequency less than 5 Hz.”
```

RTLola in a Nutshell

```
input lat, lon: Float64 // from GPS
input accel_x: Float64  // from accelerometer
input slow_down_cmd: Bool

output gps_samples @1Hz := lat.aggregate(over_exactly: 1s, using: count)
trigger gps_samples < 5 “GPS frequency less than 5 Hz.”
```

Accelerometer and GPS readings coincide.

RTLola in a Nutshell

```
input lat, lon: Float64 // from GPS
input accel_x: Float64  // from accelerometer
input slow_down_cmd: Bool

output gps_samples @1Hz := lat.aggregate(over_exactly: 1s, using: count)
trigger gps_samples < 5 “GPS frequency less than 5 Hz.”

output accel_velo @1Hz := accel_x.aggregate(over: 5s, using: ∫)
output gps_velo @1Hz := lon.aggregate(over: 5s, using: ∇)
trigger abs(accel_velo - gps_velo) > 0.1 “Conflicting measurements for velocity.”
```

RTLola in a Nutshell

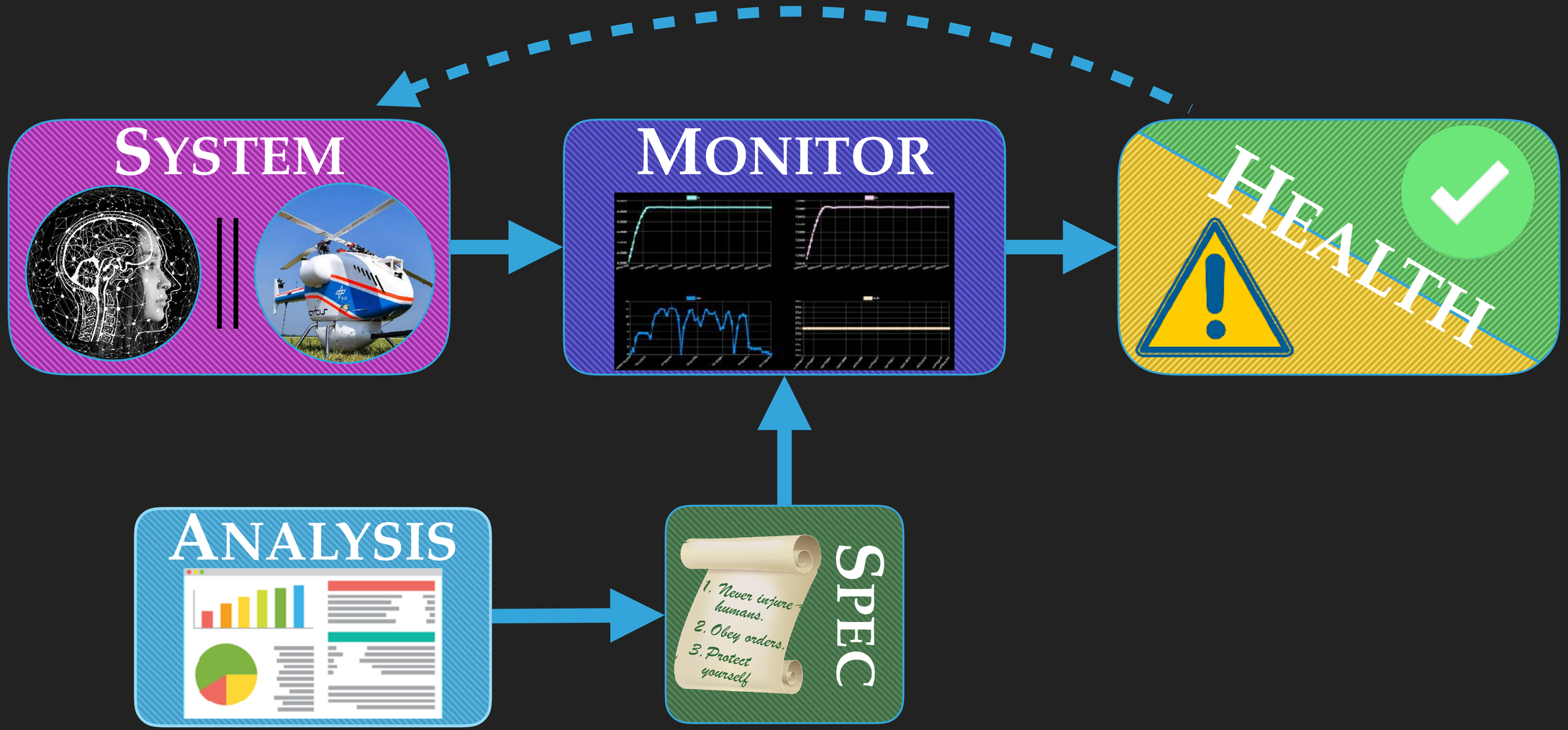
```
input lat, lon: Float64 // from GPS
input accel_x: Float64  // from accelerometer
input slow_down_cmd: Bool

output gps_samples @1Hz := lat.aggregate(over_exactly: 1s, using: count)
trigger gps_samples < 5 “GPS frequency less than 5 Hz.”

output accel_velo @1Hz := accel_x.aggregate(over: 5s, using: ∫)
output gps_velo @1Hz := lon.aggregate(over: 5s, using: ∇)
trigger abs(accel_velo - gps_velo) > 0.1 “Conflicting measurements for velocity.”
```

A slow-down is preceded by the respective command.

RTLola in a Nutshell

```
input lat, lon: Float64 // from GPS
input accel_x: Float64  // from accelerometer
input slow_down_cmd: Bool

output gps_samples @1Hz := lat.aggregate(over_exactly: 1s, using: count)
trigger gps_samples < 5 “GPS frequency less than 5 Hz.”

output accel_velo @1Hz := accel_x.aggregate(over: 5s, using: ∫)
output gps_velo @1Hz := lon.aggregate(over: 5s, using: ∇)
trigger abs(accel_velo - gps_velo) > 0.1 “Conflicting measurements for velocity.”

output fast := accel_velo > 700
output slow_down := fast.offset(by: -1).defaults(to: false) ∧ ¬fast
trigger @1Hz ¬slow_down_cmd.aggregate(over: 5s, using: ∃)
             ∧ slow_down.hold().defaults(to: false) “Spurious Slow-Down.”
```

Our Take on Runtime Verification



Strong Type System

```
input lat, lon: Float64 // from GPS
input accel_x: Float64  // from accelerometer
input slow_down_cmd: Bool

output gps_samples @1Hz := lat.aggregate(over_exactly: 1s, using: count)
trigger gps_samples < 5 “GPS frequency less than 5 Hz.”

output accel_velo @1Hz := accel_x.aggregate(over: 5s, using: ∫)
output gps_velo @1Hz := lon.aggregate(over: 5s, using: ∇)
trigger abs(accel_velo - gps_velo) > 0.1 “Conflicting measurements for velocity.”

output fast := accel_velo > 700
output slow_down := fast.offset(by: -1).defaults(to: false) ∧ ¬fast
trigger @0.5Hz ¬slow_down_cmd.aggregate(over: 5s, using: ∃)
                ∧ slow_down.hold().defaults(to: false) “Spurious Slow-Down.”
```

Strong Type System

```
input lat, lon: Float64 // from GPS
input accel_x: Float64  // from accelerometer
input slow_down_cmd: Bool

output gps_samples @1Hz := lat.aggregate(over_exactly: 1s, using: count)
trigger gps_samples < 5 “GPS frequency less than 5 Hz.”
output accel_velo @1Hz := accel_x.aggregate(over: 5s, using: ∫)
output gps_velo @1Hz := lon.aggregate(over: 5s, using: ∇)
trigger abs(accel_velo - gps_velo) > 0.1 “Conflicting measurements for velocity.”

output fast := accel_velo > 700
output slow_down := fast.offset(by: -1).defaults(to: false) ∧ ¬fast
trigger @0.5Hz ¬slow_down_cmd.aggregate(over: 5s, using: ∃)
    ∧ slow_down.hold().defaults(to: false) “Spurious Slow-Down.”
```

Strong Type System

```
input lat, lon: Float64 // from GPS
input accel_x: Float64  // from accelerometer
input slow_down_cmd: Bool
```

Float64	{lon}

Float64	{lat}
Float64	{acc}

Bool	{cmd}

```
output gps_samples @1Hz := lat.aggregate(over_exactly: 1s, using: count)
trigger gps_samples < 5 “GPS frequency less than 5 Hz.”
```

UInt64	1Hz

```
output accel_velo @1Hz := accel_x.aggregate(over: 5s, using: ∫)
output gps_velo @1Hz := lon.aggregate(over: 5s, using: ∇)
trigger abs(accel_velo - gps_velo) > 0.1 “Conflicting measurements for velocity.”
```

Float64	1Hz
Float64	1Hz

```
output fast := accel_velo > 700
output slow_down := fast.offset(by: -1).defaults(to: false) ∧ ¬fast
trigger @0.5Hz ¬slow_down_cmd.aggregate(over: 5s, using: ∃)
               ∧ slow_down.hold().defaults(to: false) “Spurious Slow-Down.”
```

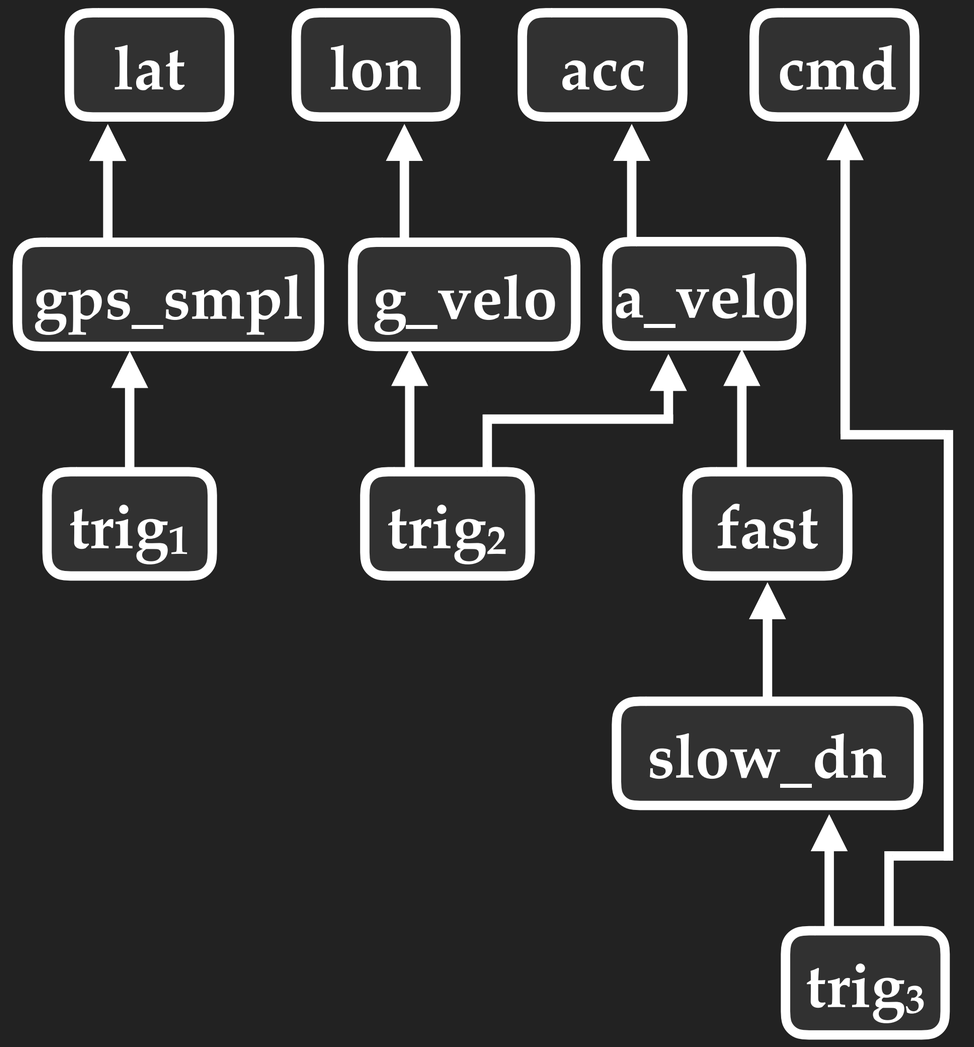
Bool	1Hz

Bool	1Hz
Bool	0.5Hz

Specification

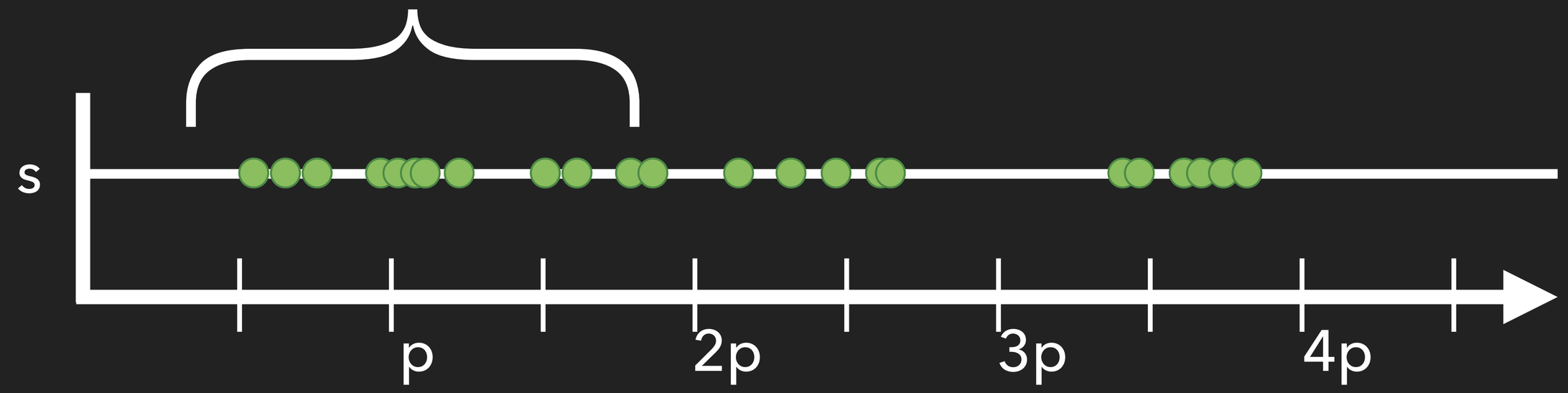
```
input lat, lon: Float64 // from GPS
input accel_x: Float64  // from accelerometer
input slow_down_cmd: Bool

output gps_samples @1Hz := lat.aggregate(over_exactly: 1s, using: count)
trigger gps_samples < 5 “GPS frequency less than 5 Hz.”

output accel_velo @1Hz := accel_x.aggregate(over: 5s, using: ∫)
output gps_velo @1Hz := lon.aggregate(over: 5s, using: ∇)
trigger abs(accel_velo - gps_velo) > 0.1
    “Conflicting measurements for velocity.”

output fast := accel_velo > 700
output slow_down := fast.offset(by: -1).defaults(to: false) ∧ ¬fast
trigger @1Hz ¬slow_down_cmd.aggregate(over: 5s, using: ∃)
    ∧ slow_down.hold().defaults(to: false) “Spurious Slow-Down.”
```

Dependency Graph



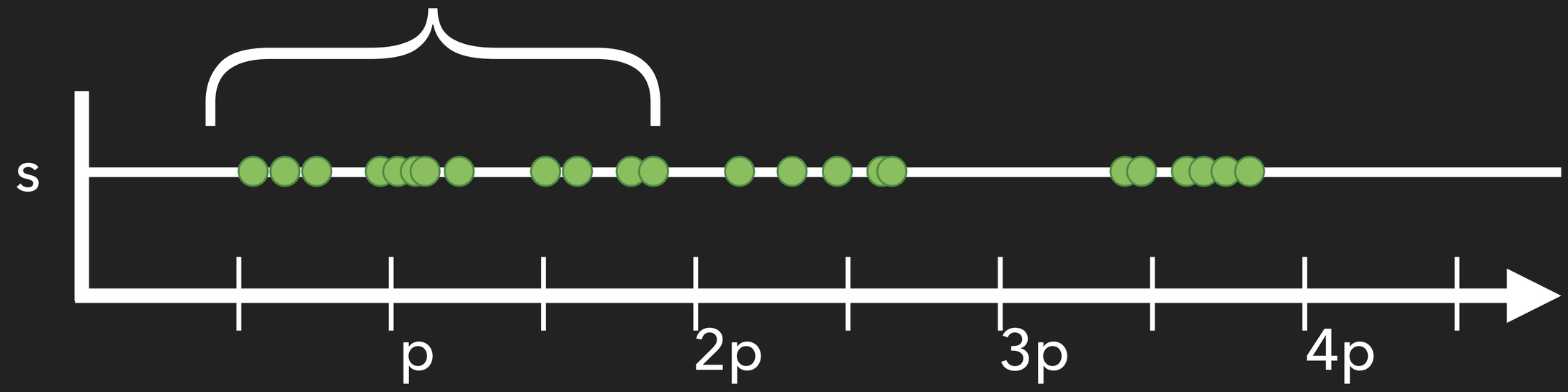
Sliding Windows

output h := **s**.aggr(over: 1.5p, using: γ)



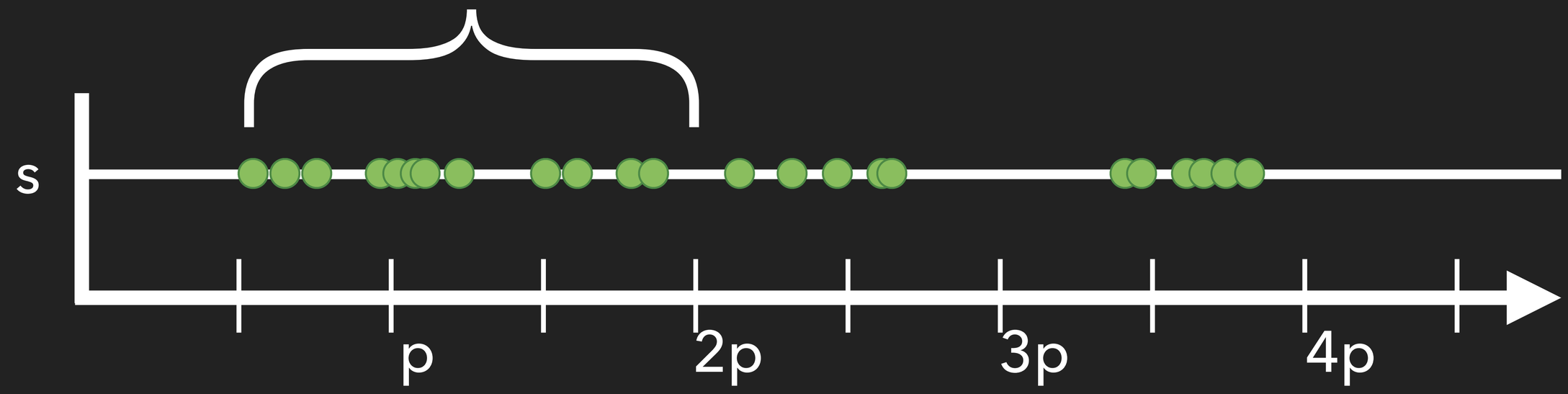
Sliding Windows

output h := **s**.aggr(over: 1.5p, using: γ)



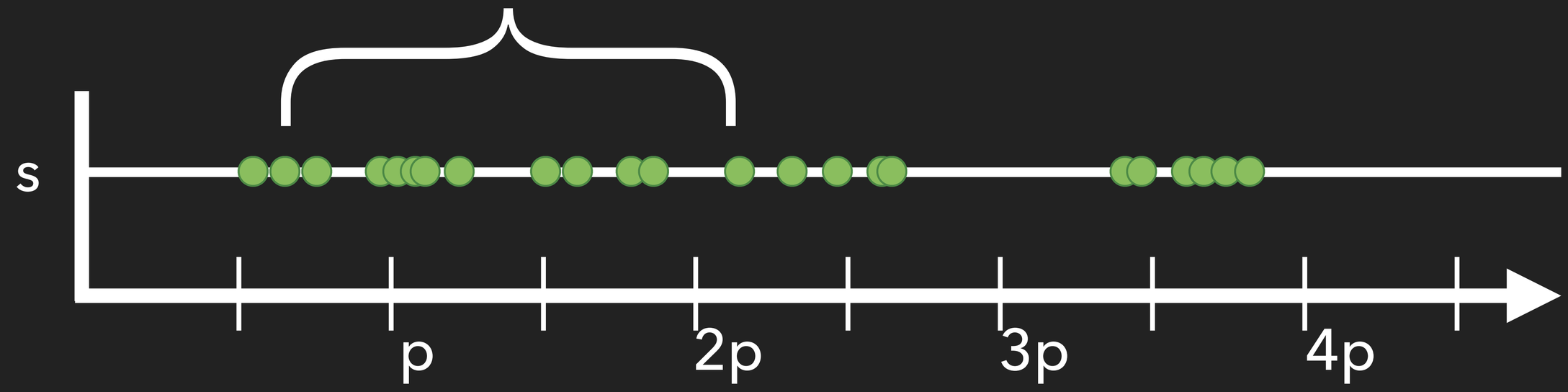
Sliding Windows

output h := **s**.aggr(over: 1.5p, using: γ)



Sliding Windows

output h := **s**.aggr(over: 1.5p, using: γ)



List Homomorphisms

output h := s.aggr(over: 1.5p, using: γ)

$\gamma : A^* \to B$

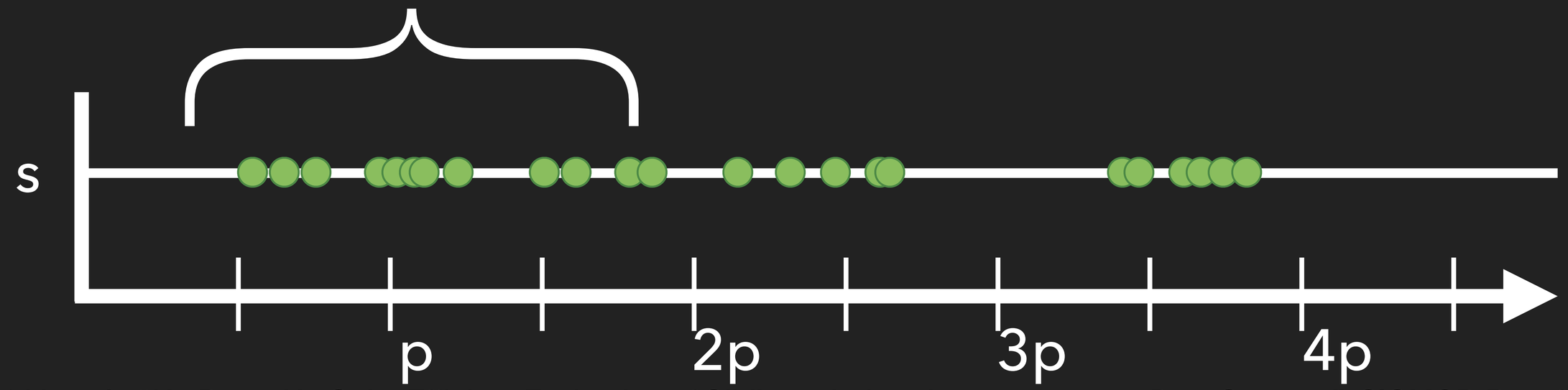
$\mathsf{map}_\gamma : A \to T \qquad \mathsf{fin}_\gamma : T \to B \qquad \circ_\gamma : T \times T \to T$

$\gamma(v_1, \ldots, v_n) = \mathsf{fin}_\gamma(\mathsf{map}_\gamma(v_1) \circ_\gamma \ldots \circ_\gamma \mathsf{map}_\gamma(v_n))$

- Meertens: "*Algorithmics: Towards programming as a mathematical activity*", 1986

Sliding Windows

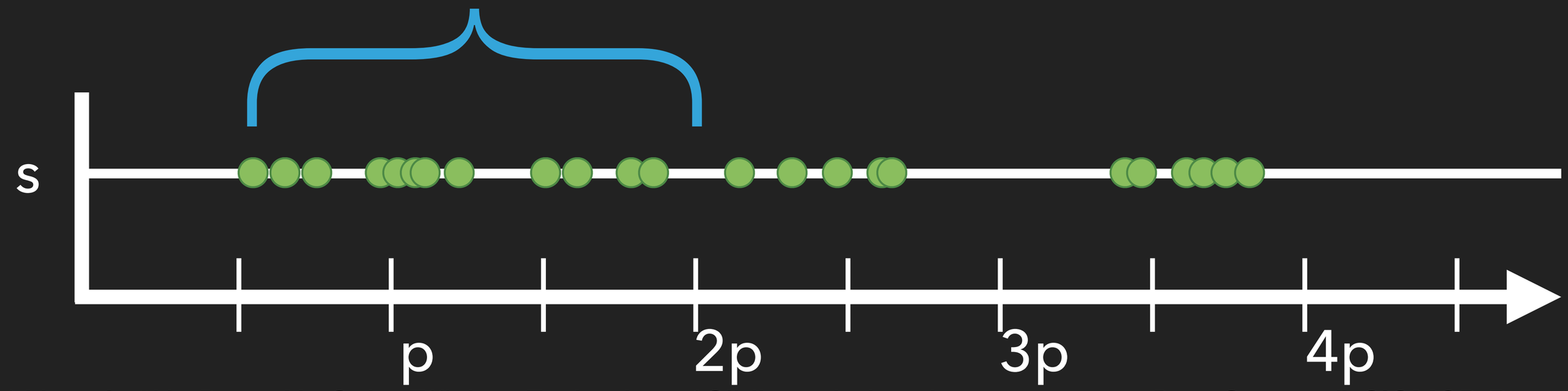
output h := s.aggr(over: 1.5p, using: γ)



- Li et al.: "*No Pane, No Gain: Efficient Evaluation of Sliding-window Aggregates over Data Streams*", SIGMOD Rec. 2005
- Schwenger: "*Let's not Trust Experience Blindly: Formal Monitoring of Humans and other CPS*", Master Thesis 2019

Sliding Windows

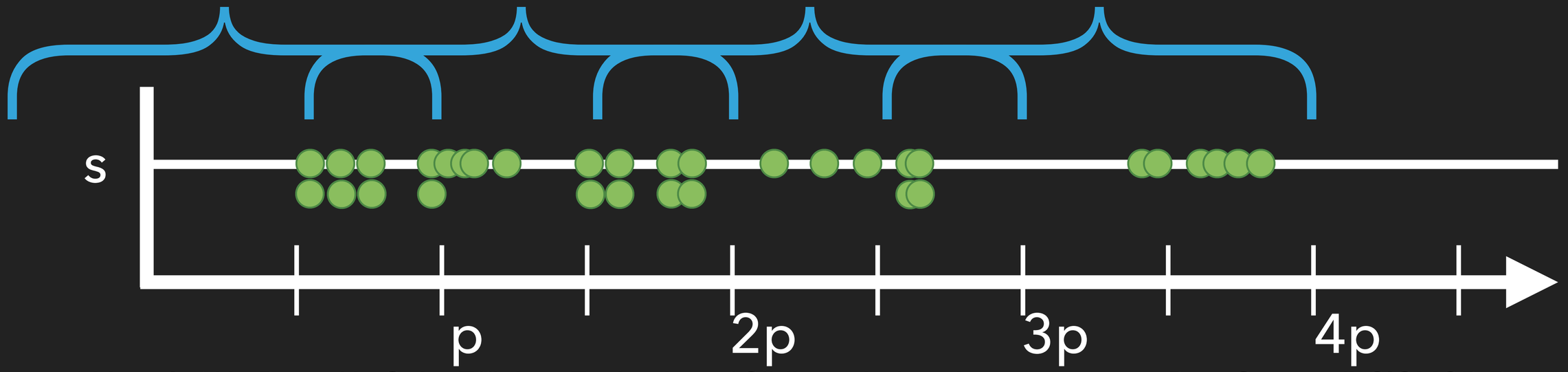
output **h** $@p^{-1}Hz$:= **s**.aggr(over: 1.5p, using: γ)



- Li et al.: "*No Pane, No Gain: Efficient Evaluation of Sliding-window Aggregates over Data Streams*", SIGMOD Rec. 2005
- Schwenger: "*Let's not Trust Experience Blindly: Formal Monitoring of Humans and other CPS*", Master Thesis 2019

Sliding Windows

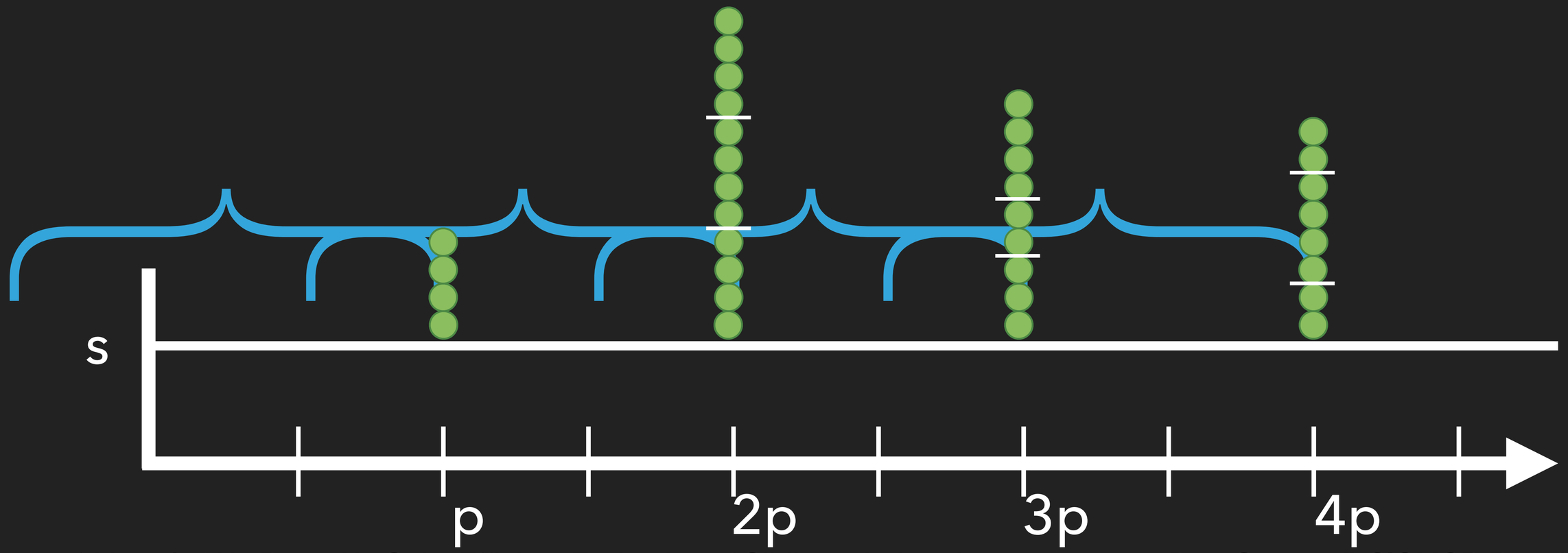
output **h** $@p^{-1}Hz$:= **s**.aggr(over: 1.5p, using: γ)



- Li et al.: "*No Pane, No Gain: Efficient Evaluation of Sliding-window Aggregates over Data Streams*", SIGMOD Rec. 2005
- Schwenger: "*Let's not Trust Experience Blindly: Formal Monitoring of Humans and other CPS*", Master Thesis 2019

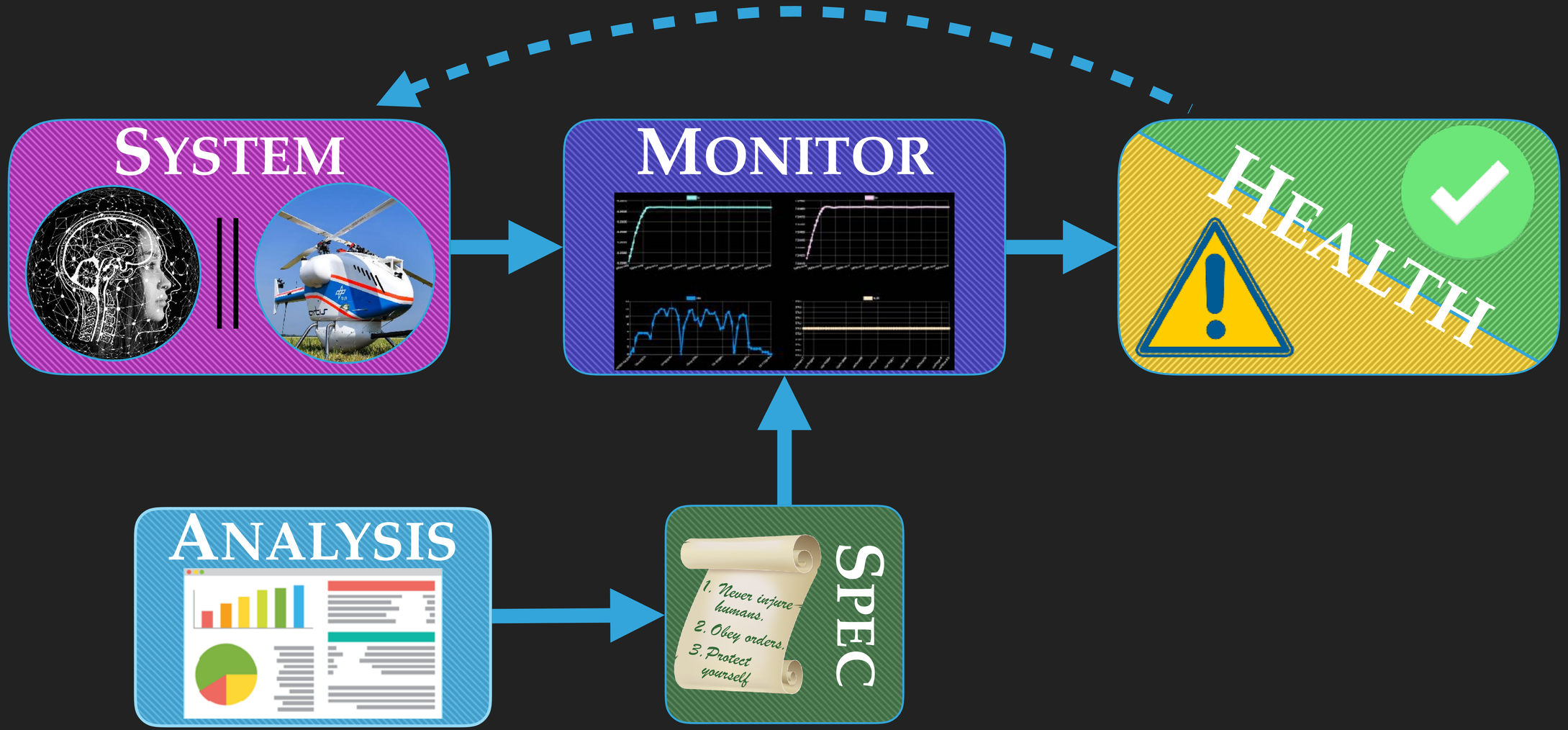
Sliding Windows

output **h** $@p^{-1}Hz$:= **s**.aggr(over: 1.5p, using: γ)

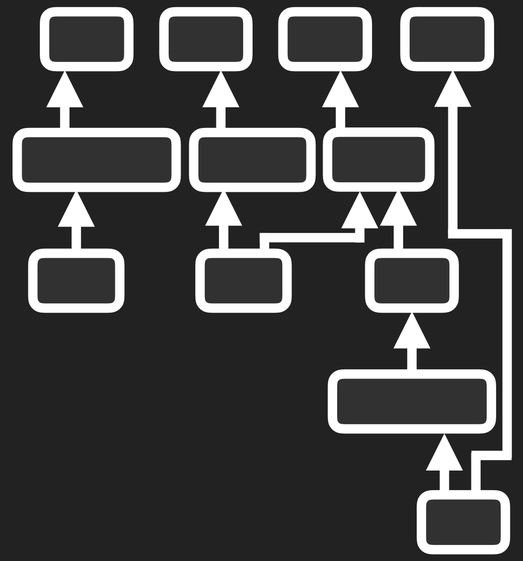


- Li et al.: "*No Pane, No Gain: Efficient Evaluation of Sliding-window Aggregates over Data Streams*", SIGMOD Rec. 2005
- Schwenger: "*Let's not Trust Experience Blindly: Formal Monitoring of Humans and other CPS*", Master Thesis 2019

Our Take on Runtime Verification



StreamLAB

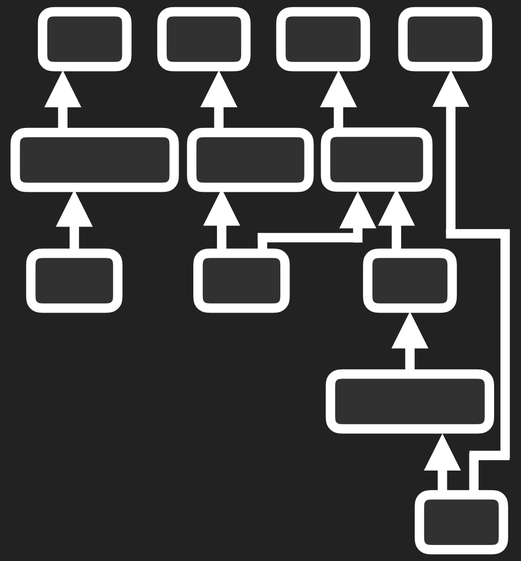


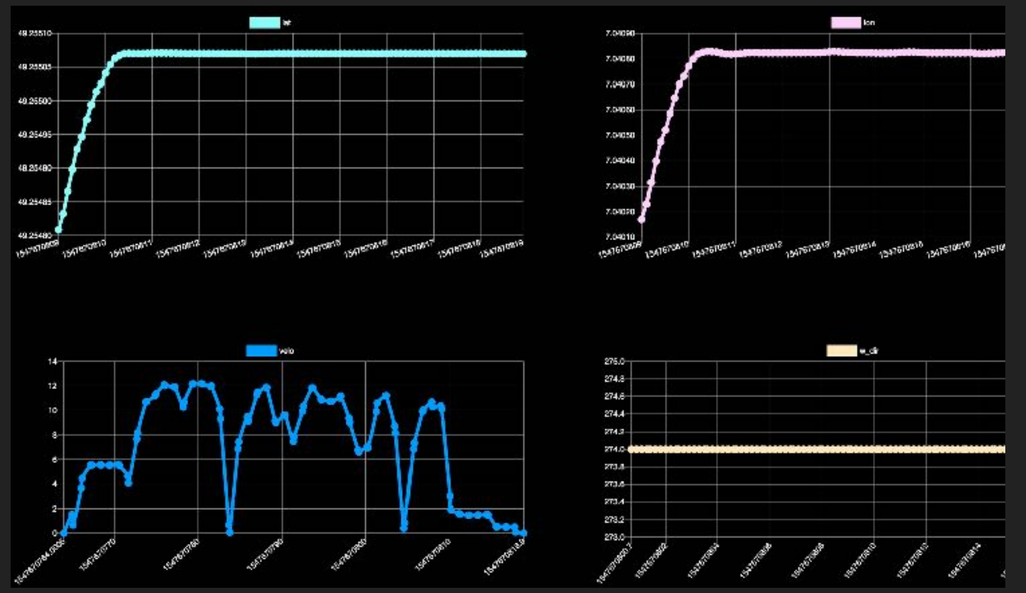
RTLola Specification

Annotated DG Intermediate Rep.

Backend

StreamLAB





Graphical UI developed by Sanny schmitt

RTLola Specification

Annotated DG Intermediate Rep.

Rust Interpretation

Rust Interpreter

Specification:

GPS frequency validation

GPS/IMU jump detection

Hover phase detection

Results:

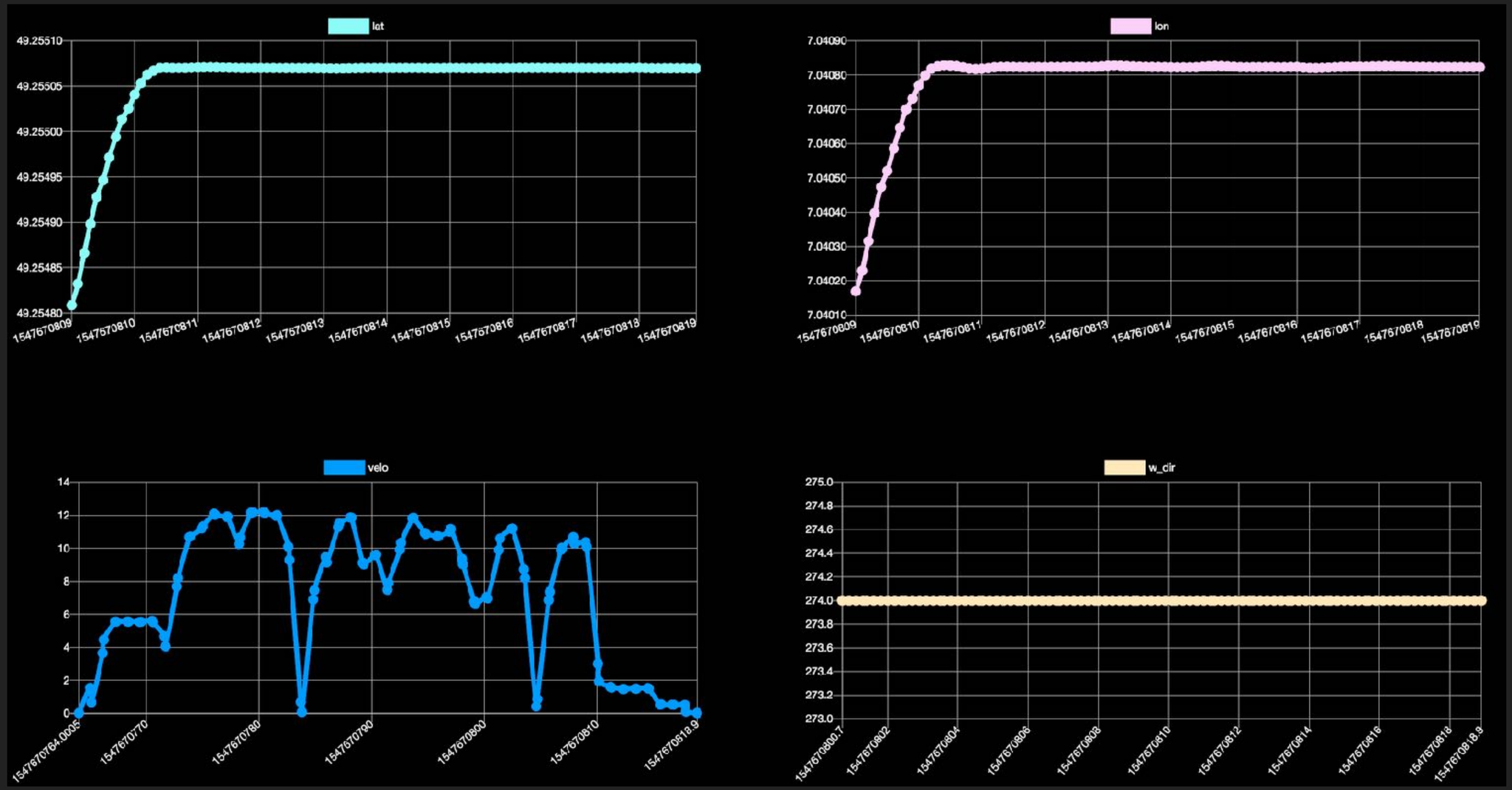
433,000 events

1,545ns per event @ 146%

Stack size < 1kB, no heap

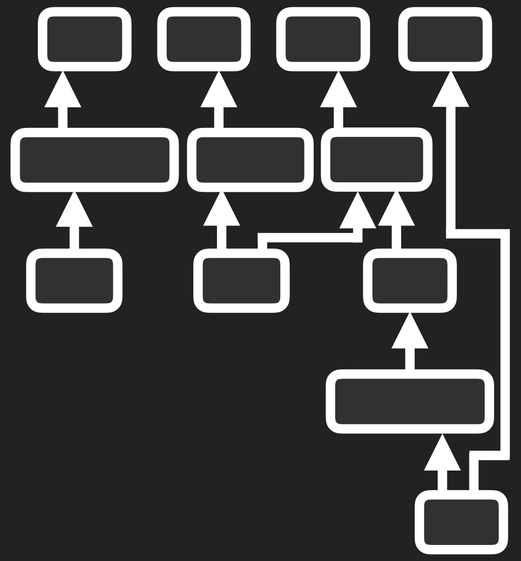
- Faymonville, Finkbeiner, Schledjewski, Schwenger, Stenger, Tentrup, Torfah, "*StreamLAB: Stream-based Monitoring of Cyber-physical Systems*", CAV 2019

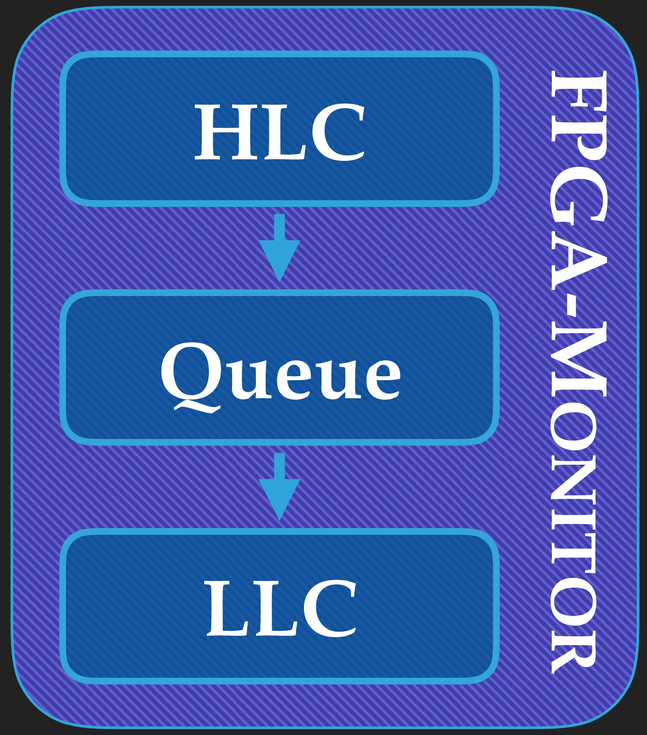
Rust Interpreter



Thanks to Sanny Schmitt for designing the interface!

StreamLAB



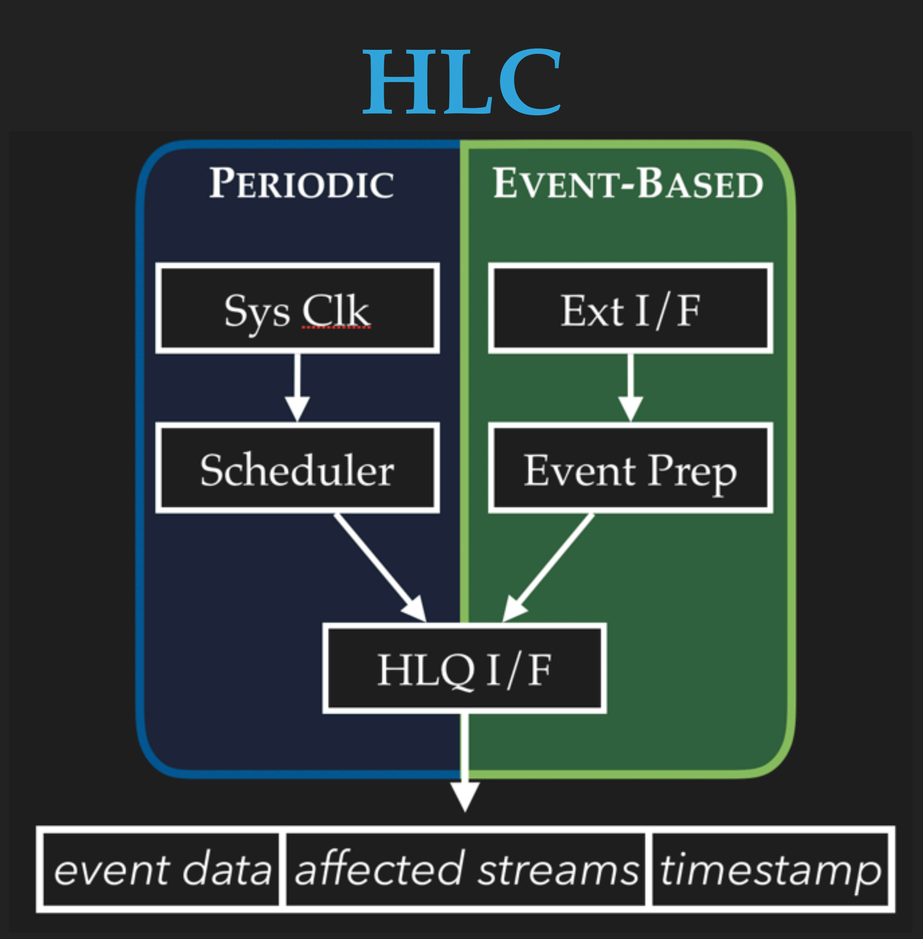


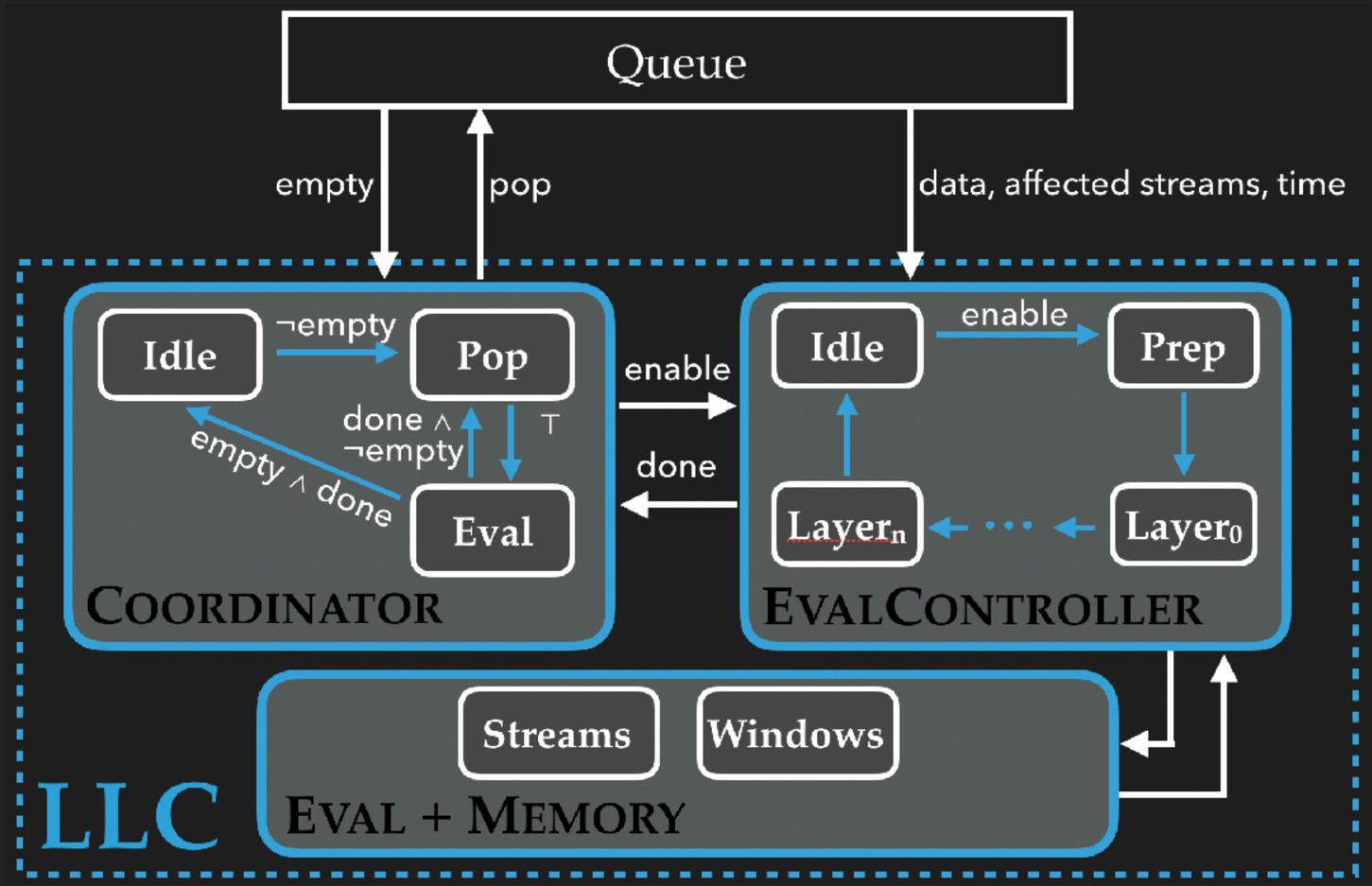
RTLola Specification

Annotated DG Intermediate Rep.

Hardware Compilation

VHDL/FPGA Compilation



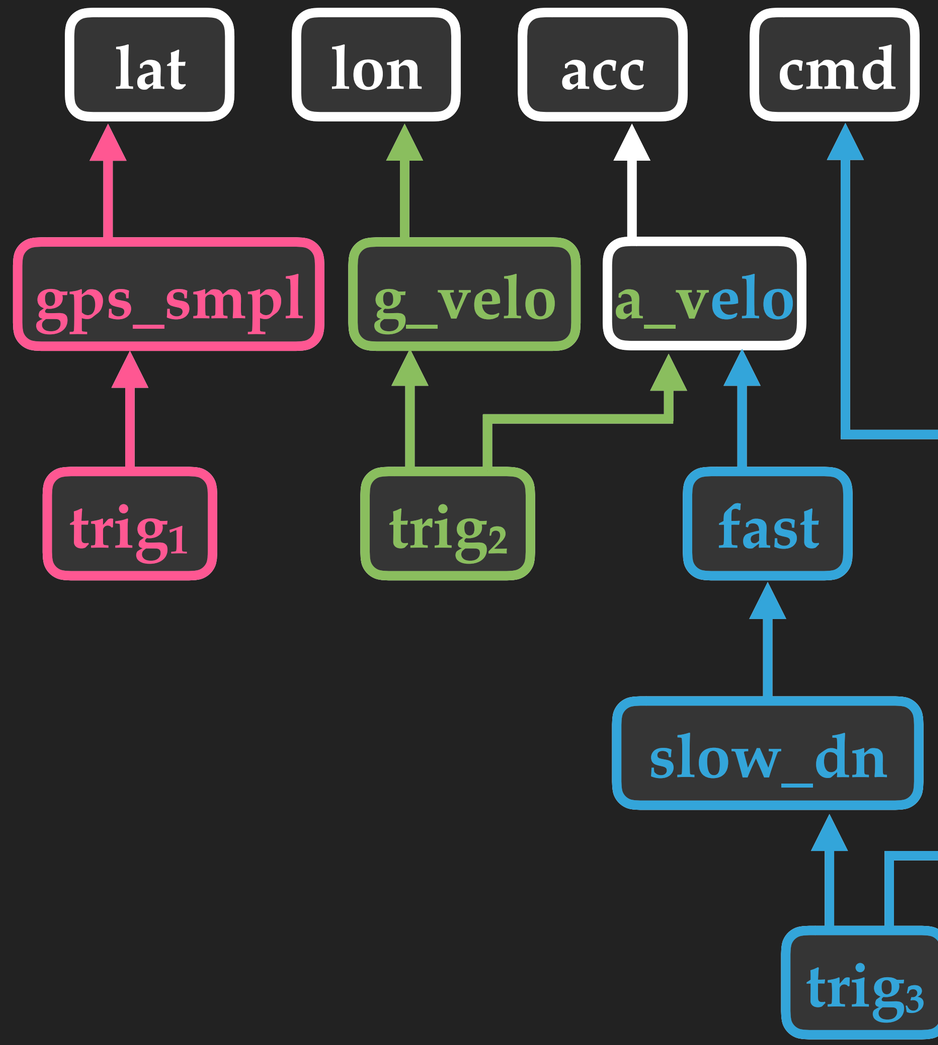


- Baumeister, Finkbeiner, Schwenger, Torfah, "*FPGA Stream-Monitoring of Real-Time Properties*", EMSOFT 2019
- Baumeister, Finkbeiner, Schwenger, Torfah, "*On the Similarities of Aircraft and Humans*", CyberCardia@ESWeek2019

Specification

```
input lat, lon: Float64 // from GPS
input accel_x: Float64  // from accelerometer
input slow_down_cmd: Bool

output gps_samples @1Hz := lat.aggregate(over_exactly: 1s, using: count)
trigger gps_samples < 5 “GPS frequency less than 5 Hz.”

output accel_velo @1Hz := accel_x.aggregate(over: 5s, using: ∫)
output gps_velo @1Hz := lon.aggregate(over: 5s, using: ∇)
trigger abs(accel_velo - gps_velo) > 0.1
    “Conflicting measurements for velocity.”

output fast := accel_velo > 700
output slow_down := fast.offset(by: -1).defaults(to: false) ∧ ¬fast
trigger @1Hz ¬slow_down_cmd.aggregate(over: 5s, using: ∃)
    ∧ slow_down.hold().defaults(to: false) “Spurious Slow-Down.”
```

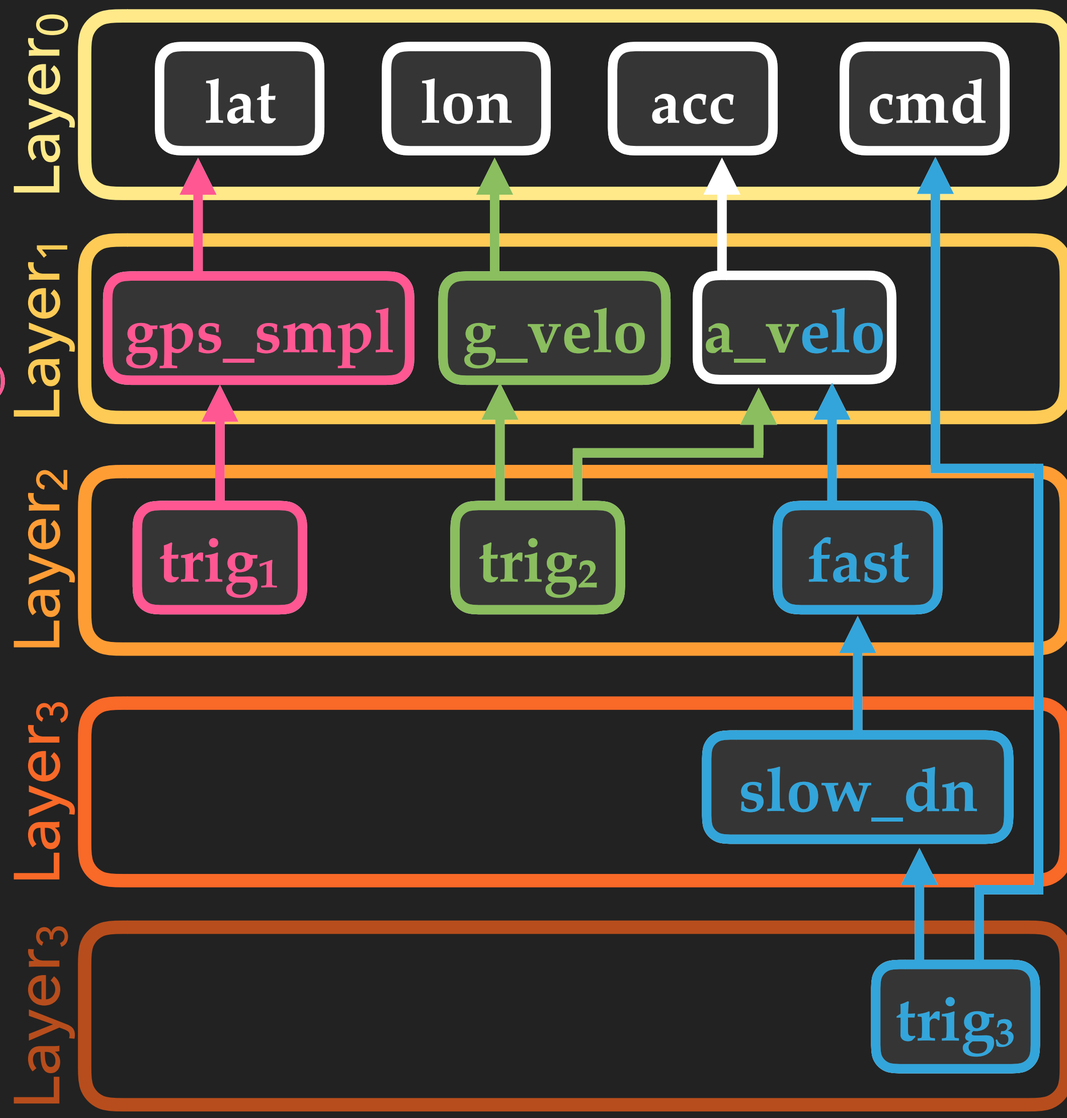
Dependency Graph



Specification

```
input lat, lon: Float64 // from GPS
input accel_x: Float64  // from accelerometer
input slow_down_cmd: Bool

output gps_samples @1Hz := lat.aggregate(over_exactly: 1s, using: count)
trigger gps_samples < 5 “GPS frequency less than 5 Hz.”

output accel_velo @1Hz := accel_x.aggregate(over: 5s, using: ∫)
output gps_velo @1Hz := lon.aggregate(over: 5s, using: ∇)
trigger abs(accel_velo - gps_velo) > 0.1
    “Conflicting measurements for velocity.”

output fast := accel_velo > 700
output slow_down := fast.offset(by: -1).defaults(to: false) ∧ ¬fast
trigger @1Hz ¬slow_down_cmd.aggregate(over: 5s, using: ∃)
    ∧ slow_down.hold().defaults(to: false) “Spurious Slow-Down.”
```

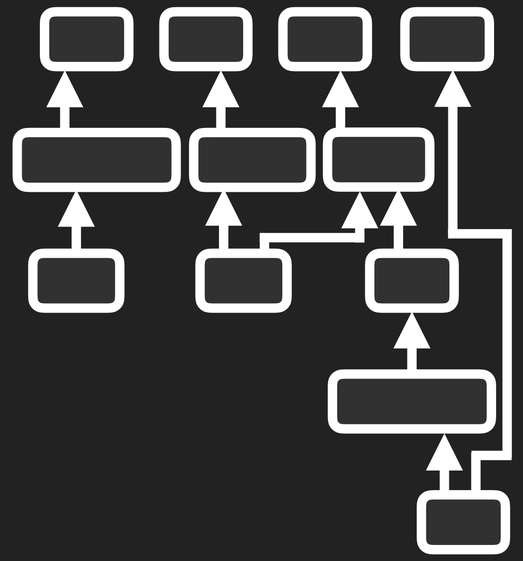
Dependency Graph



Evaluation

		FF	LUT	MUX	CA	MULT	Pwr [W]	Time [µs]
Drone	Mon	3036	3685	26	656	10	1.620	4.28
	HLC	901	156	0	22	0		
	Q	543	442	0	43	0		
	LLC	1281	2820	0	576	10		
Network	Mon	1905	1533	23	226	23	1.570	3.20
	HLC	550	161	0	37	0		
	Q	330	342	0	28	0		
	LLC	895	927	0	161	0		
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		

StreamLAB



```
invariant gm_s3[i-2] == r.s3_mem[0]
invariant |trigger1_ghost| == i-4
invariant |trigger2_ghost| == i-4
invariant forall j:Int :: {t3[j],t2[j]
invariant forall j:Int :: {t3[j],t2[j]
invariant forall j:Int :: {t3[j],t2[j]

var s1_i: Int := t1[i] + r.s2_mem[0]
var s2_i: Int := t2[i] + r.s3_mem[0]
var s3_i: Int := t3[i] + r.s1_mem[0]

var trigger1: Bool := s3_i -s2_i < 0
var trigger2: Bool := s1_i - t1[i] ==

assert trigger1 <==> (t3[i] + r.s1_mem
3  silicon  ✓ Successfully verified negative-cycle.vpr in
```

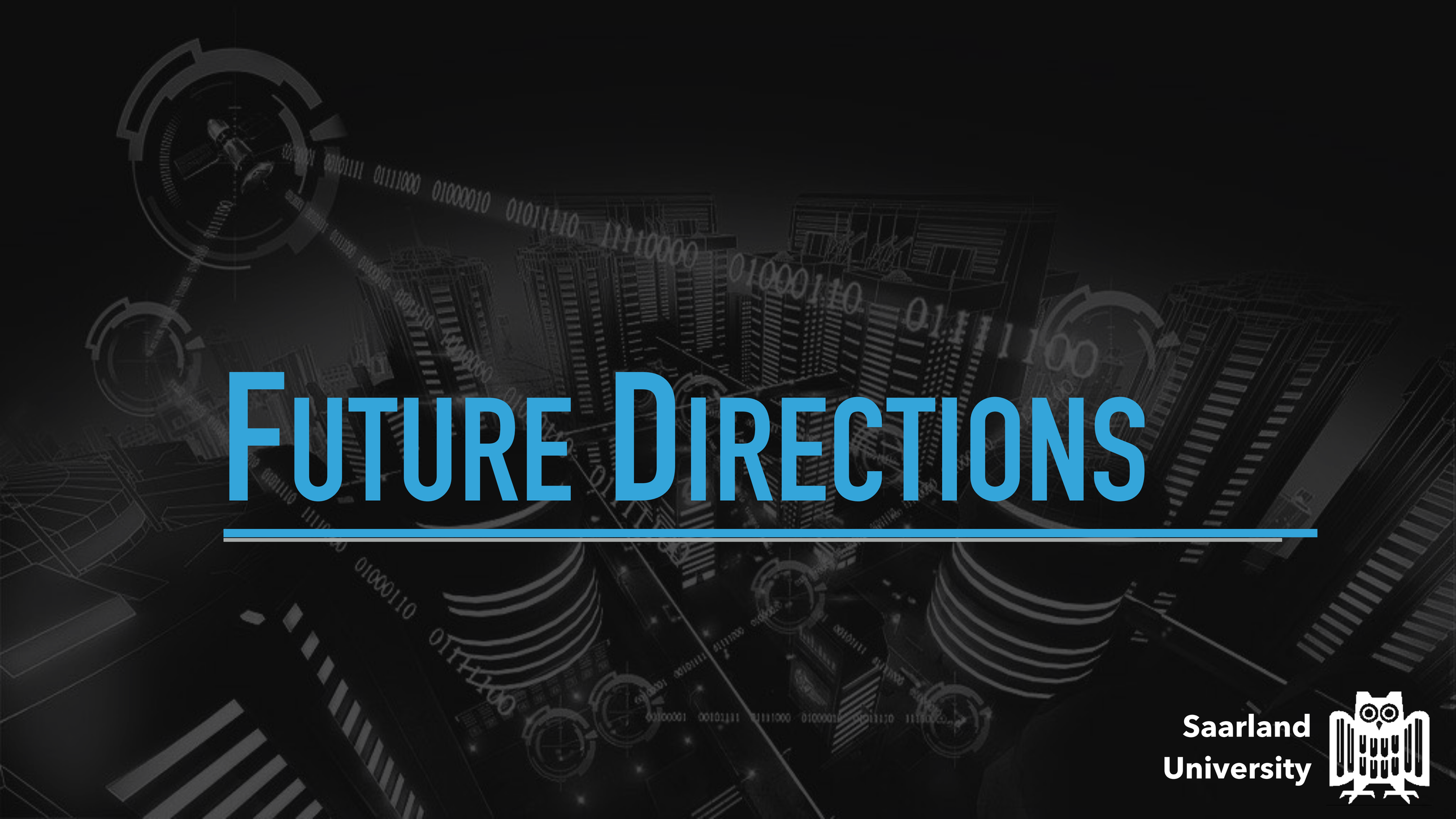
Developed by Stefan Oswald, co-advised by Noemi Passing

RTLola Specification

Annotated DG Intermediate Rep.

Viper Compilation

Future Directions



Saarland University

Language Development

```
input lat, lon: Float64 // from GPS
input accel_y: Float64  // from accelerometer
input slow_down_cmd: Bool

output imu_velo @1Hz := accel_y.aggregate(over: 1s, using: ∫) + imu_velo.offset(by: -1)
output imu_pos @1Hz := imu_velo.aggregate(over: 1s, using: ∫) + imu_pos.offset(by: -1)
trigger abs(imu_pos - lat) > 0.1 “Conflicting measurements for position estimation.”
```

$$v(t) = \int_0^t a(\tau)d\tau = \sum_{i=0}^{t-1} \left(\int_i^{i+1} a(\tau)d\tau \right)$$

Language Development

```
import integration // aggregations, functions (sqrt), macros (indef. integration, haversine…)

input lat, lon: Float64 // from GPS
input accel_y: Float64  // from accelerometer
input slow_down_cmd: Bool

output imu_velo @1Hz := accel_y.aggregate(over: ∞, using: ∫)
output imu_pos @1Hz := indef_integral(imu_velo)
trigger abs(imu_pos - lat) > 0.1 “Conflicting measurements for position estimation.”
```

$$v(t) = \int_0^t a(\tau)d\tau = \sum_{i=0}^{t-1} \left(\int_i^{i+1} a(\tau)d\tau \right)$$

Reactive Synthesis meets Runtime Verification

Agend A

System S

Controller C

Specification φ

Find C s.t.

$\forall \sigma \in \mathit{runs}(A \parallel S \parallel C)\colon \sigma \models \varphi$

Reactive Synthesis meets Runtime Verification

Agend A

System S

Controller C

Specification φ

Find C s.t.

$\forall \sigma \in \mathit{runs}(A \parallel S \parallel C)\colon \sigma \models \varphi$

Outlook

Outlook

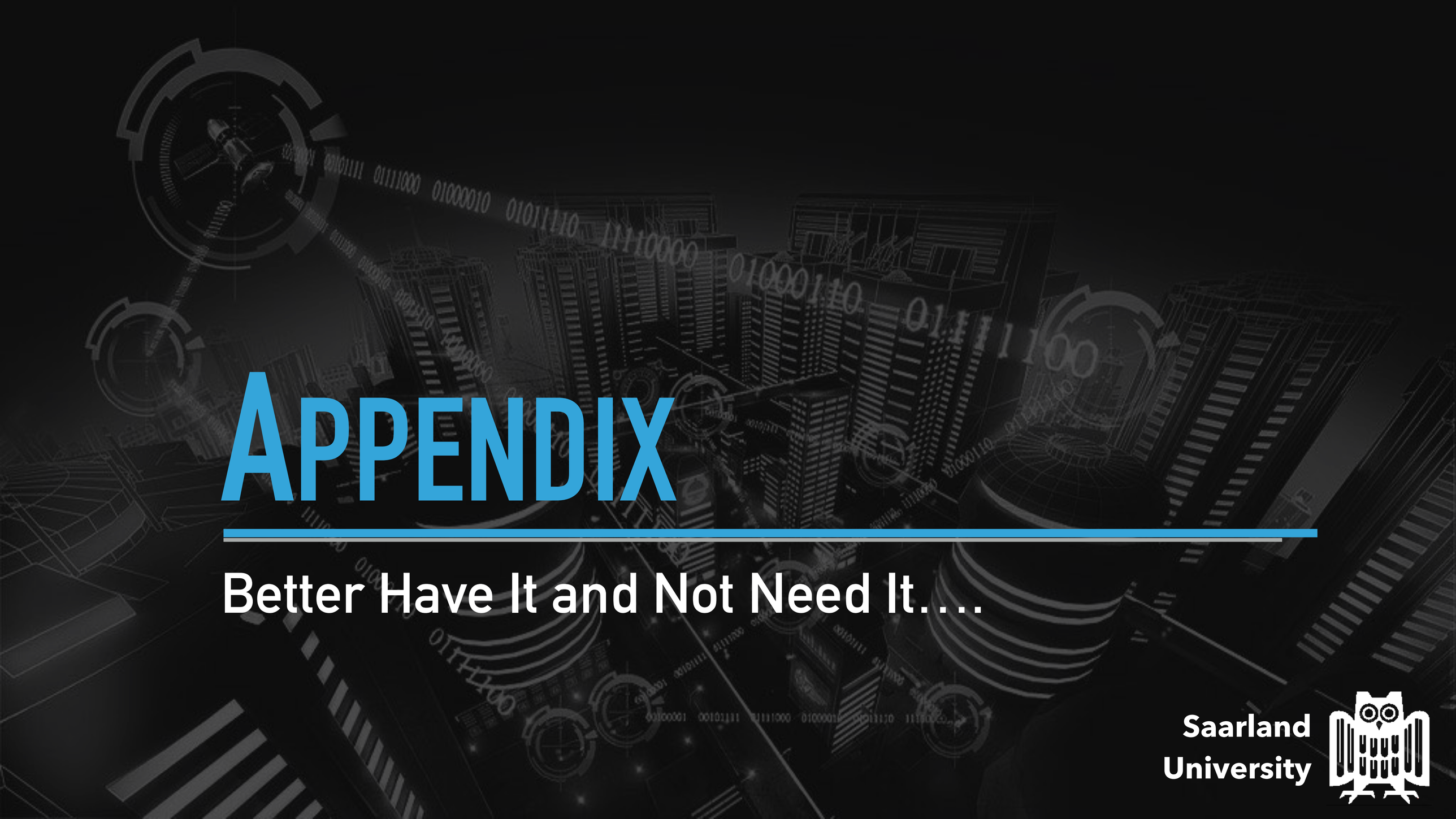
TOYOTA
RESEARCH INSTITUTE

?

Check out StreamLAB: stream-lab.eu **Contact: schwenger@react.uni-saarland.de**

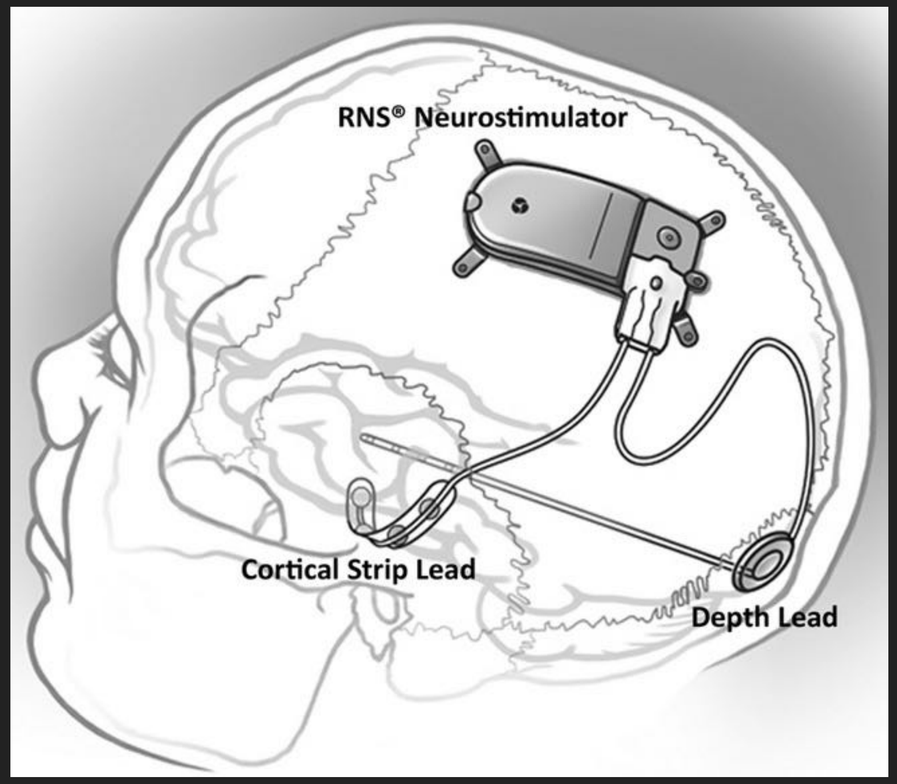
Appendix

Better Have It and Not Need It….



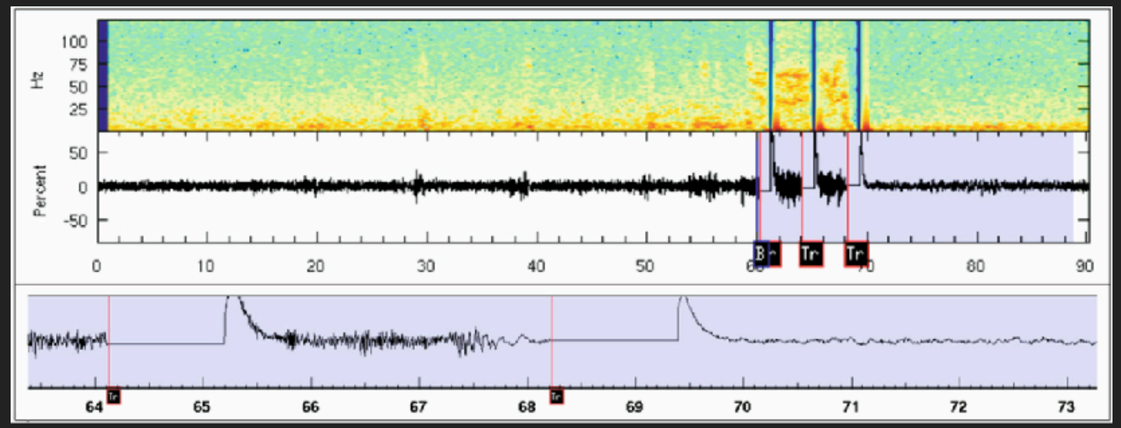
Saarland University

Responsive Neurostimulator



Heck et al., *"Two-year seizure reduction in adults with medically intractable partial onset epilepsy treated with responsive neurostimulation: Final results of the RNS System Pivotal trial"*, Epilepsia 2014

cortical electroencephalogram

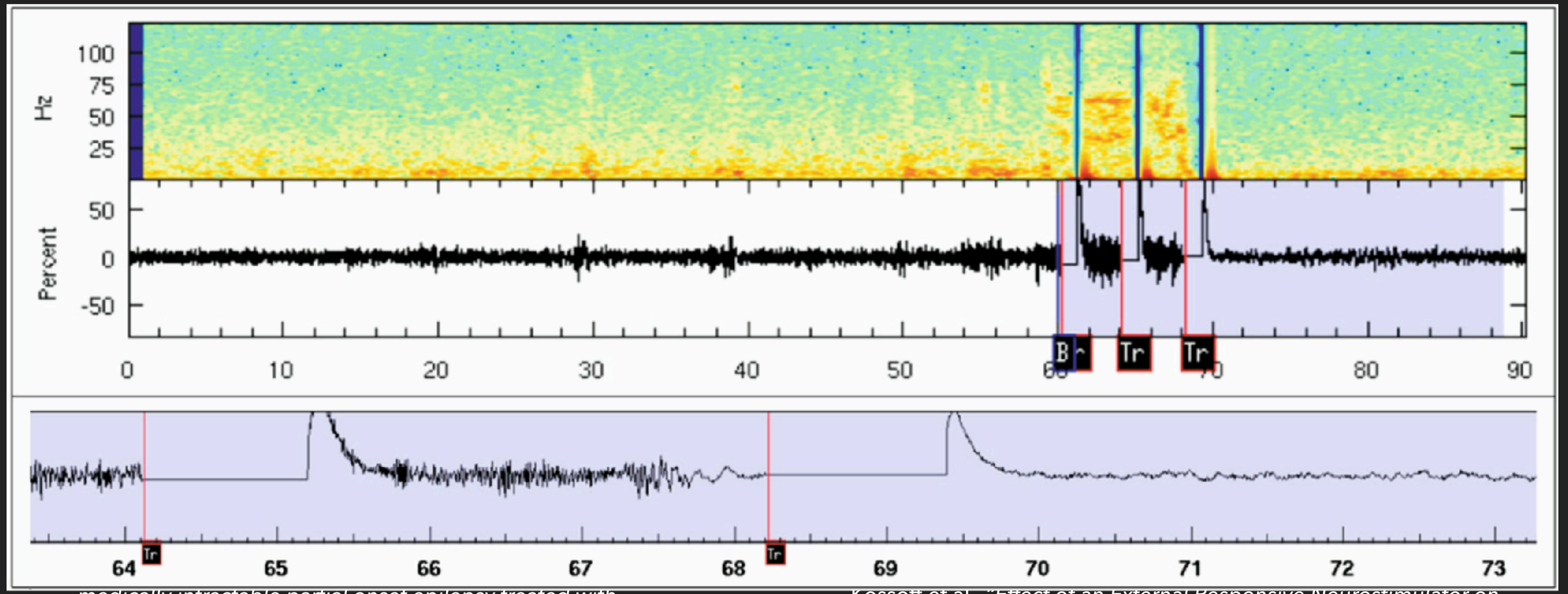


Sun et al., *"Responsive Cortical Stimulation for the Treatment of Epilepsy"*, Neurotherapeutics 2008

Kossoff et al., *"Effect of an External Responsive Neurostimulator on Seizures and Electrographic Discharges during Subdural Electrode Monitoring"*, Epilepsia 2004

Responsive Neurostimulator

cortical electroencephalogram

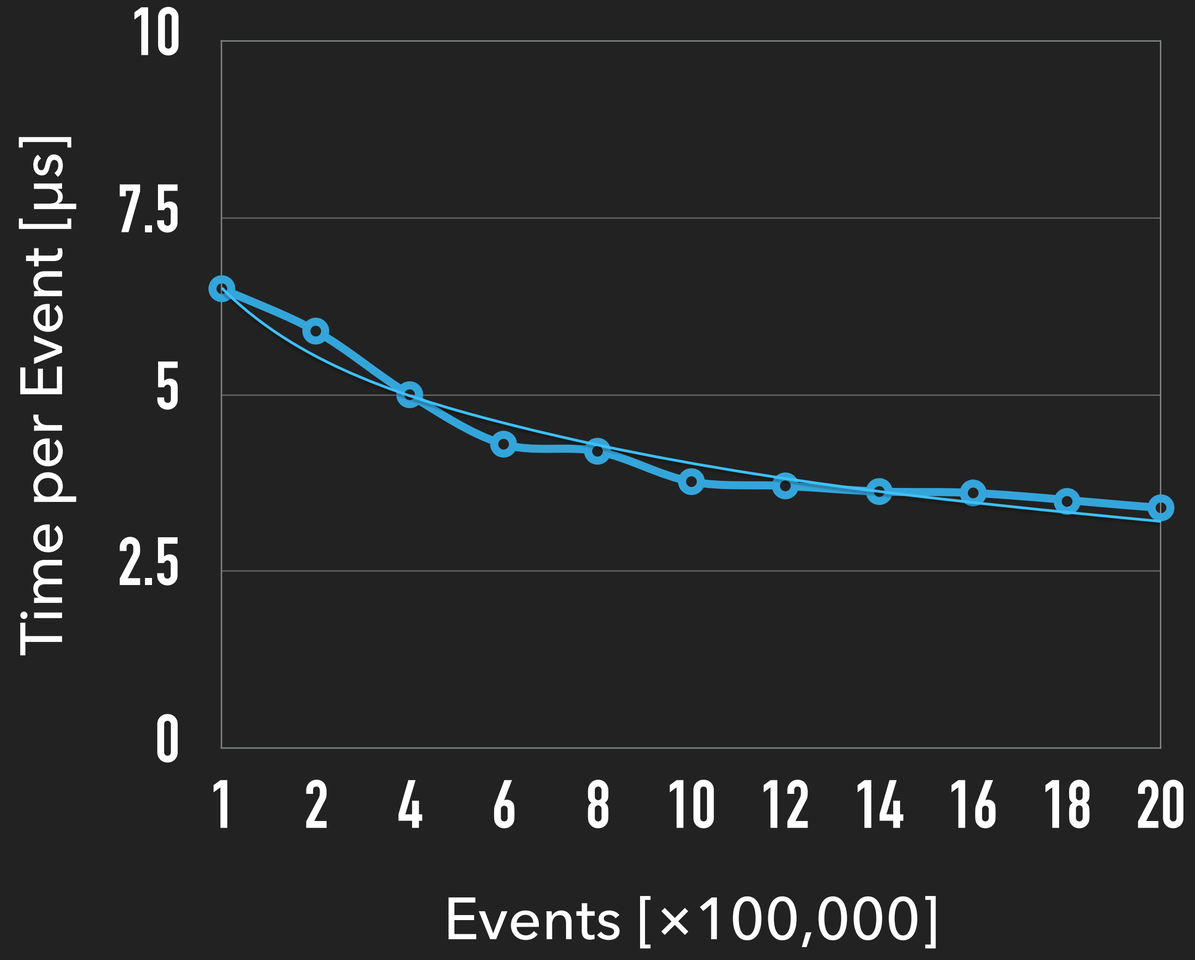


medically intractable partial onset epilepsy treated with responsive neurostimulation: Final results of the RNS System Pivotal trial", Epilepsia 2014

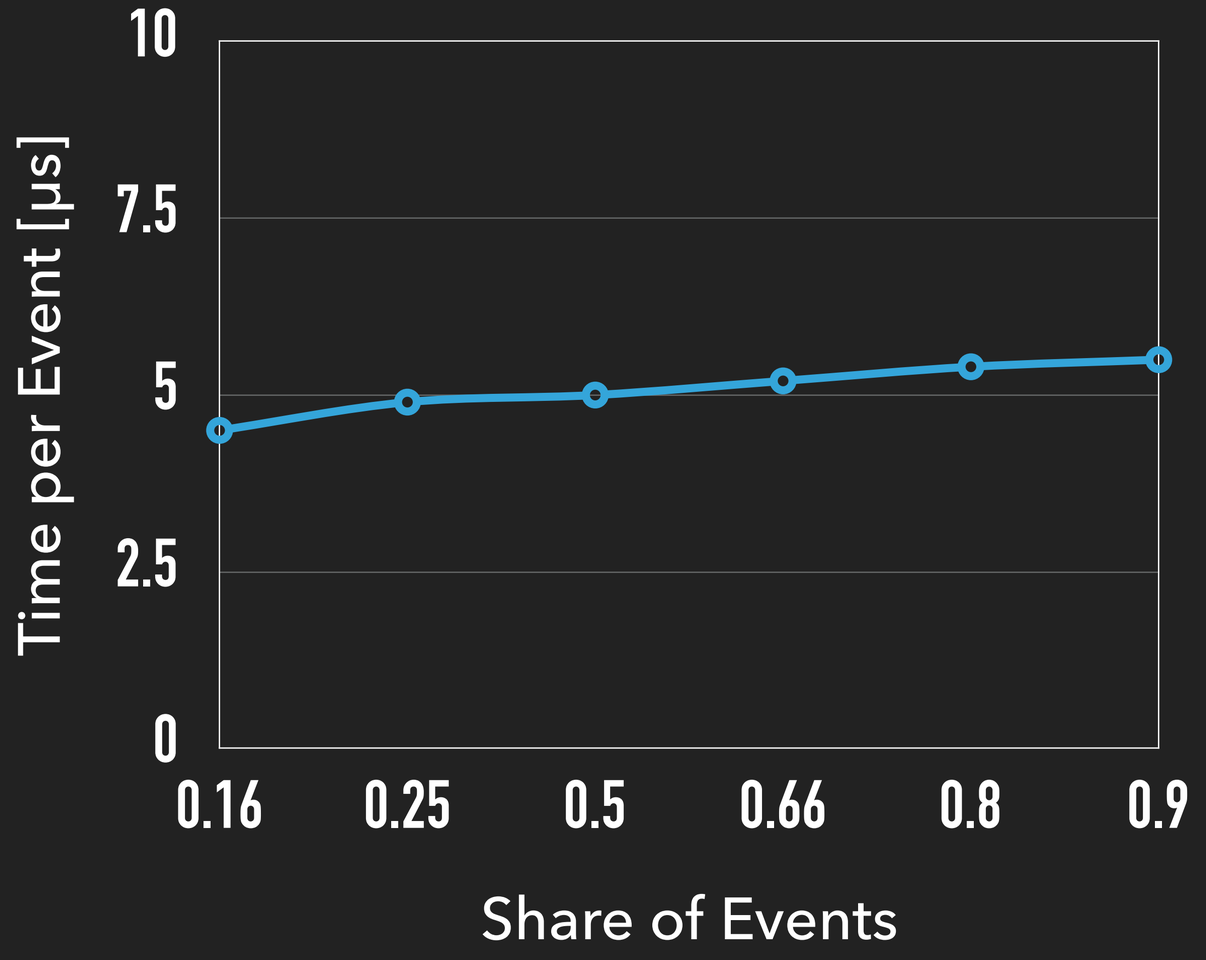
Kossoff et al., *"Effect of an External Responsive Neurostimulator on Seizures and Electrographic Discharges during Subdural Electrode Monitoring"*, Epilepsia 2004

Running Time

50% EV, 50% P



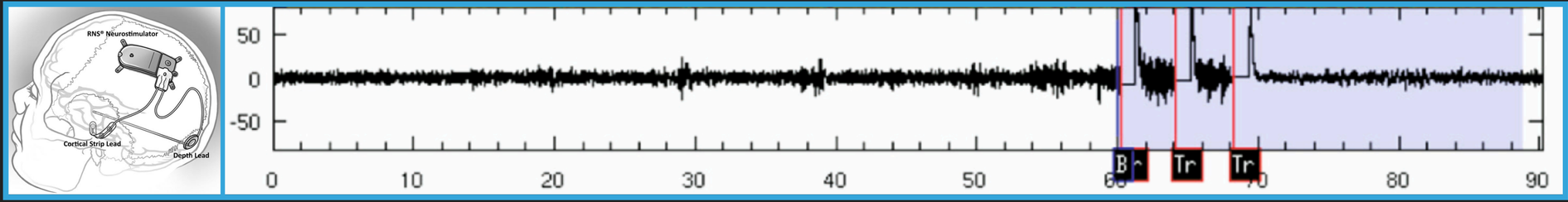
200K EV+P



Huge thanks to Leander, Marvin, and Malte!

RTLola: Medical Domain

```
input CLS: Float64
input rec, stim: Bool

output jerk := abs(derive(3, CLS)

output avg_long @100mHz := jerk.aggr(over: 2000s, using: avg)
output avg_short @1kHz := jerk.aggr(over: 2ms, using: avg)

output spike @1kHz := avg_short > avg_long.hold() + ε
trigger spike ∧ ¬rec.aggregate(over: 2ms, using: any) “Seizure not recognized”
trigger @1kHz rec.aggregate(over: 5ms, using: any) ∧
                    ¬stim.aggr(over: 3ms, using: any) “Stimulation not triggered”
```



Through the Zoo of RV Approaches*

Logics

↯ RT ↯ St

RV-LTL [2]

rLTL [3]

LTL_3 [1]

STL/MTL [4]

Data-Driven

↯ Fs ↯ Em

MTL + Aggr [5]

SW Tie-Ins

* rather a tiny fraction thereof

[1] A. Bauer, M. Leucker, C. Schallhart. "*Runtime verification for LTL and TLTL*". ACM Trans. Softw. Eng. Methodol. 2011

[2] A. Bauer, M. Leucker, C. Schallhart. "*The good, the bad, and the ugly, but how ugly is ugly*", RV 2007

[3] C. Mascle, D. Neider, M. Schwenger, P. Tabuada, A. Weinert, M. Zimmermann, "*From LTL to rLTL Monitoring: Improved Monitorability through Robust Semantics*", arxiv 2019

[4] O. Maler, D. Nickovic, "*Monitoring Temporal Properties of Continuous Signals*", FORMATS 2004

[5] D. Basin F. Klaedtke, S. Marinovic, E. Zalinescu, "*Monitoring of temporal first-order properties with aggregations*", FSMD 2015

Through the Zoo of RV Approaches*

Logics

RV-LTL [2]

rLTL [3]

LTL_3 [1]

STL/MTL [4]

Data-Driven

Fs Em

MTL + Aggr [5]

RT

Lustre [6]

(RT)

Striver [7]

RT

Lola [9] -2.0 [10];

RT- [11]; FPGA- [12]

SW Tie-Ins

* rather a tiny fraction thereof

[6] P. Caspi, D. Pilaud, N. Halbwachs, J. Plaice, *"Lustre: A Declarative Language for Programming Synchronous Systems"*, POPL 1987

[7] F. Gorostiaga, C. Sánchez, *"Striver: Stream Runtime Verification for Real-Time Event-Streams"*, RV 2018

[9] B. D'Angelo, S. Sankaranarayanan, C Sánchez, W. Robinson, B. Finkbeiner, H. Sipma, S. Mehrotra, Z. Manna, *"LOLA: Runtime Monitoring of Synchronous Systems"*, TIME 2005

[10] P. Faymonville, B. Finkbeiner, S. Schirmer, H. Torfah, *"A Stream-Based Specification Language for Network Monitoring"*, RV 2016

[11] P. Faymonville, B. Finkbeiner, M. Schledjewski, M. Schwenger, M. Stenger, L. Tentrup, H. Torfah, *"StreamLAB: Stream-based Monitoring of Cyber-Physical Systems"*, CAV 2019

[12] J. Baumeister, B. Finkbeiner, M. Schwenger, H. Torfah, *"FPGA-based Monitoring of Real-time Properties"*, EMSOFT 2019

Through the Zoo of RV Approaches*

↯ RT ↯ St

Logics

RV-LTL [2]

rLTL [3]

LTL_3 [1]

↯ Fs ↯ Em

MTL + Aggr [5]

STL/MTL [4]

Data-Driven

↯ RT

Lustre [6]

(↯ RT)

Striver [7]

↯ RT

Lola [9] -2.0 [10];

↯ RT

RT- [11]; FPGA- [12]

JavaMOP [13]

Aspects [14]

DTrace [15]

↯ Em

SW Tie-Ins

* rather a tiny fraction thereof

[13] F. Chen, G. Roşu, *"Java-MOP: A Monitoring Oriented Programming Environment for Java"*, TACAS 2005

[14] K. Havelund, E. Van Wyk, *"Aspect-Oriented Monitoring of C Programs"*

[15] C. Rosenberg, M. Steffen, V. Stolz, *"Leveraging DTrace for Runtime Verification"*, RV 2016