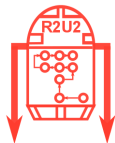


# R2U2 V3.0

Re-imagining a Toolchain for **Specification**, **Resource Estimation**,  
and Optimized **Observer Generation** for Runtime Verification in  
Hardware and Software

**Chris Johanssen**<sup>1</sup>, Phillip Jones<sup>1</sup>, Brian Kempa<sup>1</sup>, Kristin Yvonne  
Rozier<sup>1</sup>, Pei Zhang<sup>2</sup>



<sup>1</sup>Iowa State University  
[cgjohann@iastate.edu](mailto:cgjohann@iastate.edu)

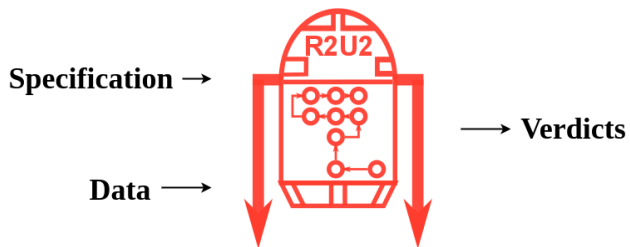
<sup>2</sup>Google LLC



35th International Conference on Computer Aided Verification  
July 2023

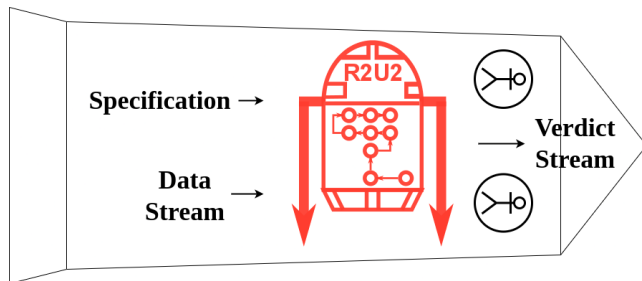
<https://r2u2.temporallogic.org/>

# What Does R2U2 Do?



Runtime Verification

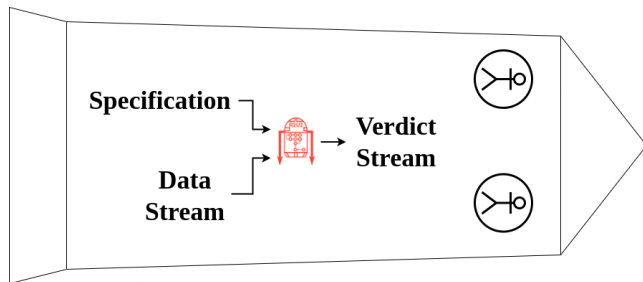
# What Does R2U2 Do?



~~Runtime Verification~~

Real-time Runtime Verification

# What Does R2U2 Do?

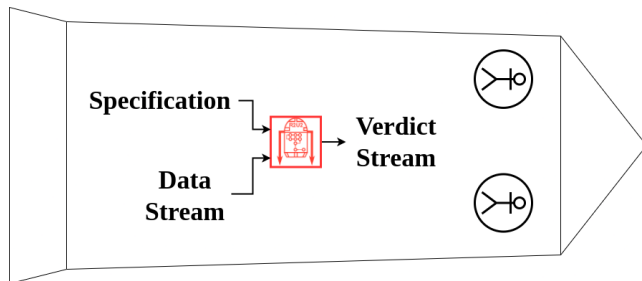


~~Runtime Verification~~

~~Real-time Runtime Verification~~

Resource-constrained Real-time Runtime Verification

# What Does R2U2 Do?



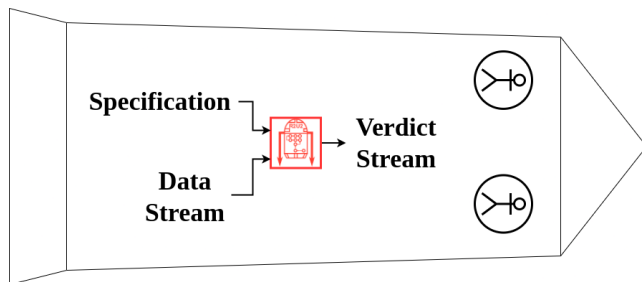
~~Runtime Verification~~

~~Real-time Runtime Verification~~

~~Resource-constrained Real-time Runtime Verification~~

Unobtrusive Resource-constrained Real-time Runtime Verification

# What Does R2U2 Do?



~~Runtime Verification~~

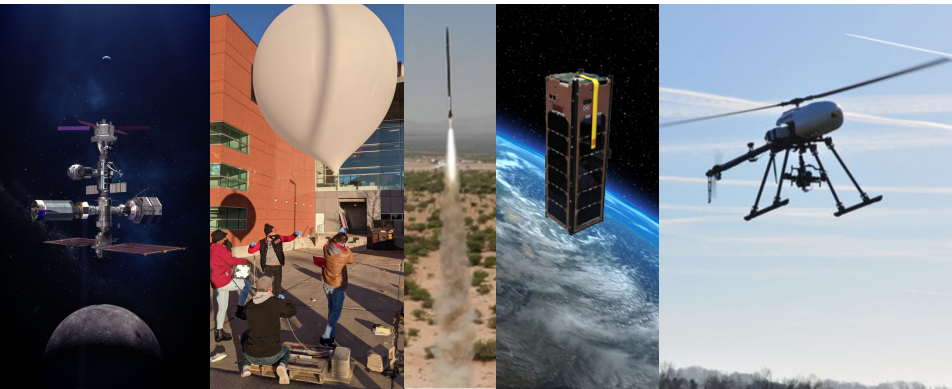
~~Real-time Runtime Verification~~

~~Resource-constrained Real-time Runtime Verification~~

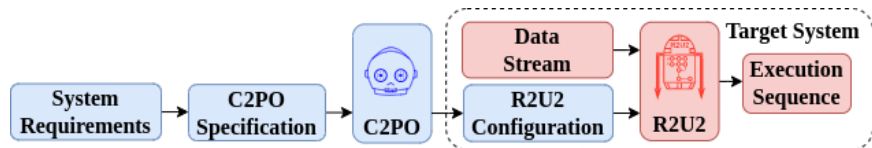
~~Unobtrusive Resource-constrained Real-time Runtime Verification~~

**Realizable, Responsive, Unobtrusive Unit (R2U2)**

# Flight-Certifiable Runtime Verification

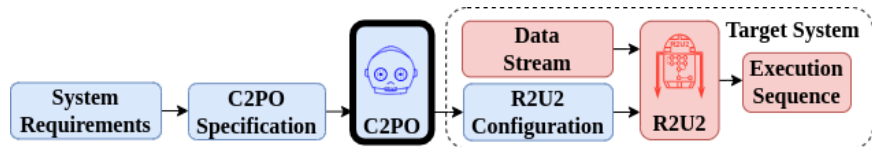


# R2U2 Toolchain Overview



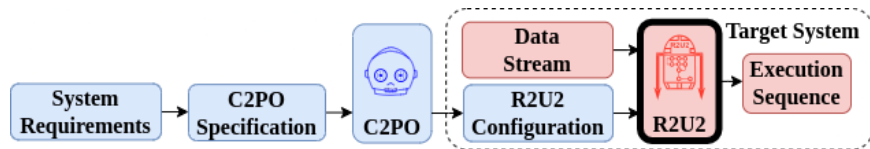


# R2U2 Toolchain Overview



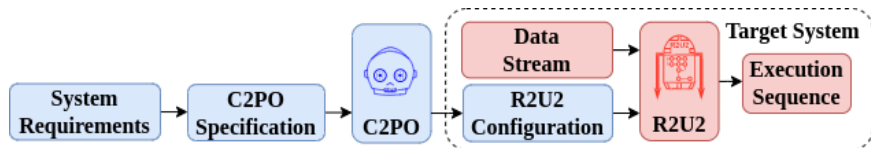
- Python: Configuration Compiler for Property Organization (C2PO)

# R2U2 Toolchain Overview



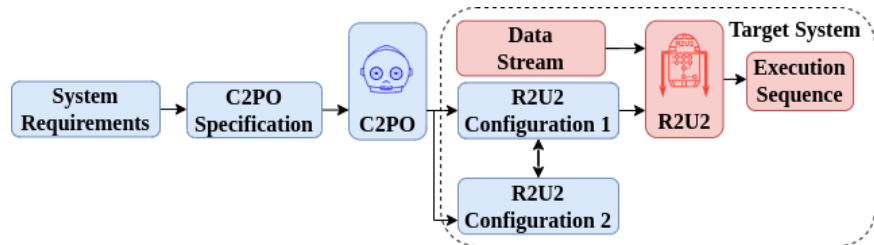
- Python: Configuration Compiler for Property Organization (C2PO)
- C: Observer (R2U2)

# R2U2 Toolchain Overview



- Python: Configuration Compiler for Property Organization (C2PO)
- C: Observer (R2U2)
  - VHDL
  - C++

# R2U2 Toolchain Overview



- Python: Configuration Compiler for Property Organization (C2PO)
- C: Observer (R2U2)
  - VHDL
  - C++

## R2U2 Example: Arbiter

### English Requirement

Every request shall be granted within 5 seconds where only 3 requests can be queued at once.

# R2U2 Example: Arbiter

## English Requirement

Every request shall be granted within 5 seconds where only 3 requests can be queued at once.

```
1  STRUCT
2  Request: { active, granted: bool; };
```

# R2U2 Example: Arbiter

## English Requirement

Every request shall be granted within 5 seconds where only 3 requests can be queued at once.

```
1  STRUCT
2    Request: { active, granted: bool; };
3
4  INPUT
5    a1,a2,a3,g1,g2,g3: bool;
```

# R2U2 Example: Arbiter

## English Requirement

Every request shall be granted within 5 seconds where only 3 requests can be queued at once.

```
1  STRUCT
2    Request: { active, granted: bool; };
3
4  INPUT
5    a1,a2,a3,g1,g2,g3: bool;
6
7  DEFINE
8    r1 := Request(a1,g1);
9    r2 := Request(a2,g2);
10   r3 := Request(a3,g3);
11   Reqs := {r1,r2,r3};
```



# R2U2 Example: Arbiter

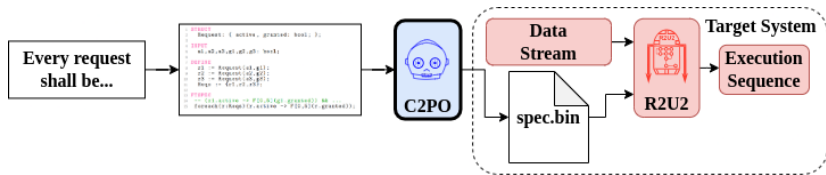
## English Requirement

Every request shall be granted within 5 seconds where only 3 requests can be queued at once.

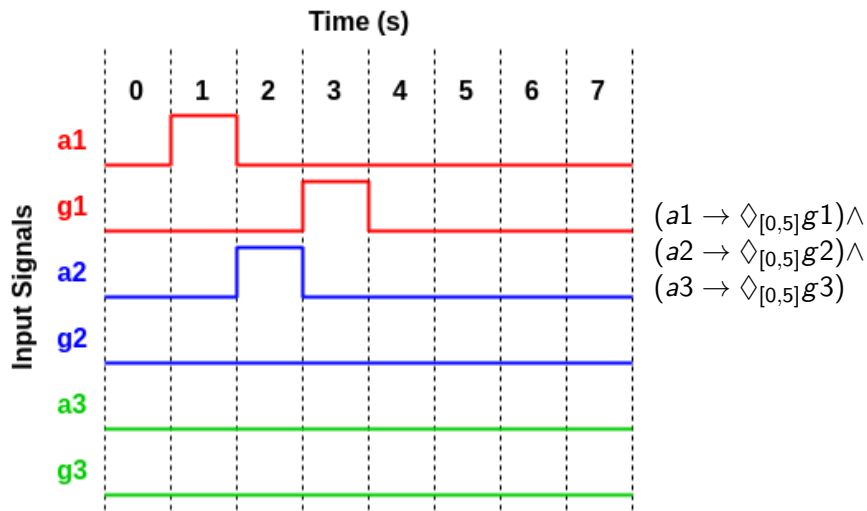
```
1  STRUCT
2    Request: { active, granted: bool; };
3
4  INPUT
5    a1,a2,a3,g1,g2,g3: bool;
6
7  DEFINE
8    r1 := Request(a1,g1);
9    r2 := Request(a2,g2);
10   r3 := Request(a3,g3);
11   Reqs := {r1,r2,r3};
12
13  FTSPEC
14   -- (r1.active -> F[0,5](g1.granted)) && ...
15   foreach(r:Reqs)(r.active -> F[0,5](r.granted));
```



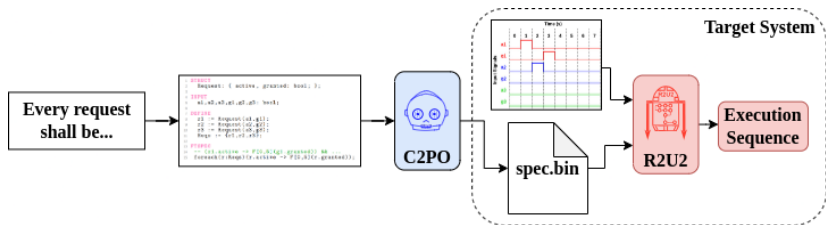
# R2U2 Example: Arbiter



## R2U2 Example: Arbiter



# R2U2 Example: Arbiter





R2U2 provides a framework for **real-time** runtime verification on **resource-constrained** systems (e.g., embedded systems).

Toolchain improvements:

- New **specification language** for defining MLTL Observers
- Implementation of **formula compiler** (C2PO)
- Improvements to internals of R2U2 **C implementation**
- Web-based GUI for **resource estimation**

<https://r2u2.temporallogic.org/>



# Specification Logic: Mission-time Linear Temporal Logic

**Mission-Time Temporal Logic (MLTL)** reasons about *integer-bounded, finite* timelines:

- finite set of atomic propositions  $\{p, q\}$
- Boolean connectives:  $\neg$ ,  $\wedge$ ,  $\vee$ , and  $\rightarrow$
- temporal connectives *with time bounds*:

Symbol	Operator	Timeline
$\square_{[2,6]}p$	ALWAYS <sub>[2,6]</sub>	
$\diamond_{[0,7]}p$	EVENTUALLY <sub>[0,7]</sub>	
$p\mathcal{U}_{[1,5]}q$	UNTIL <sub>[1,5]</sub>	
$p\mathcal{R}_{[3,8]}q$	RELEASE <sub>[3,8]</sub>	



# Specification Logic: Mission-time Linear Temporal Logic<sup>1</sup>

## Some important notes:

- Finite traces
- Finite intervals
- **U-semantics**:  $\pi \models \varphi \mathcal{U}_{[a,b]} \psi$  iff  $|\pi| > a$  and,  $\exists i \in [a, b], i < |\pi|$  such that  $\pi, i \models \psi$  and  $\forall j \in [a, b], j < i$  it holds that  $\pi, j \models \varphi$
- Intervals are closed, unit-less (generic)
- Signal processing compartmentalized

---

<sup>1</sup>Li, Vardi, Rozier. "Satisfiability checking for mission-time LTL." CAV, 2019.

# Resource Estimation GUI

**R2U2 Resource Estimator**

**C2PO Input**

```
INPUT
a0,a1,a2: bool;
b0,b1,b2: bool;

DEFINE
c := a1 || a2;

FTSPEC
a0: a0;
a1: a2;
a2: b0 || 0, 5; b2;
s3: 0(1, 3) b2;
s4: a2 a8 s3;
```

**Software Configuration**

**Clock Frequency (GHz)**  
10

**CPU Operator Latencies**  
Worst-case Exec. Time: 64.00000µs / 0.01562MHz  
Est. SCQ Memory: 0.0KB

**Hardware Configuration**

**Clock Frequency (MHz)**  
100

**LUT Type Select**  
LUT-3

**Resource to Observe**  
LUT

**Timestamp Length (Bits)**  
32

**Comparators per Node**  
33

**Adders per Node**  
32

**FPGA Operator Latencies**  
Worst-case Exec. Time: 7.80000µs / 0.12821MHz  
Total SCQ Memory Slots: 0

**Logic Diagram**

```
graph TD
  a0((a0)) --- or((or))
  a1((a1)) --- or
  or --- and((and))
  b0((b0)) --- and
  b1((b1)) --- and
  b2((b2)) --- global((global))
  and --- lut1((lut1))
  global --- lut1
```

**LUT Requirements**

Number of LUTs vs. Timestamp Width (Bits)

Timestamp Width (Bits)	LUT-3 Comparators	LUT-3 Adder/Subtractors	Current Configuration
0	0	0	0
20	~1600	~1000	~1000
40	~3200	~2000	~2000
60	~4800	~3000	~3000

**Mouseover Data**  
None Selected

**Assembly**

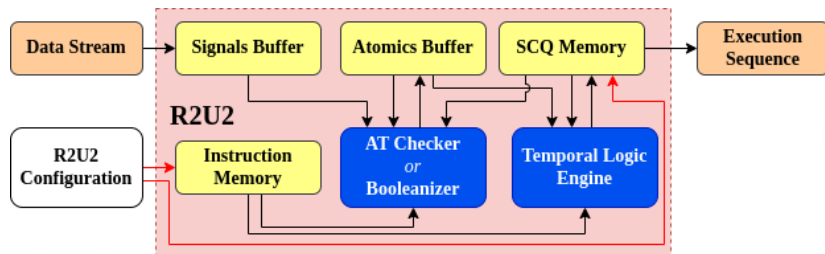
**BZ Assembly:**

```
b0 load s8 a0
b1 load s1 a1
b2 load s2 a2
b3 load s3 a3
b4 load s4 a4
b5 load s5 a5
```

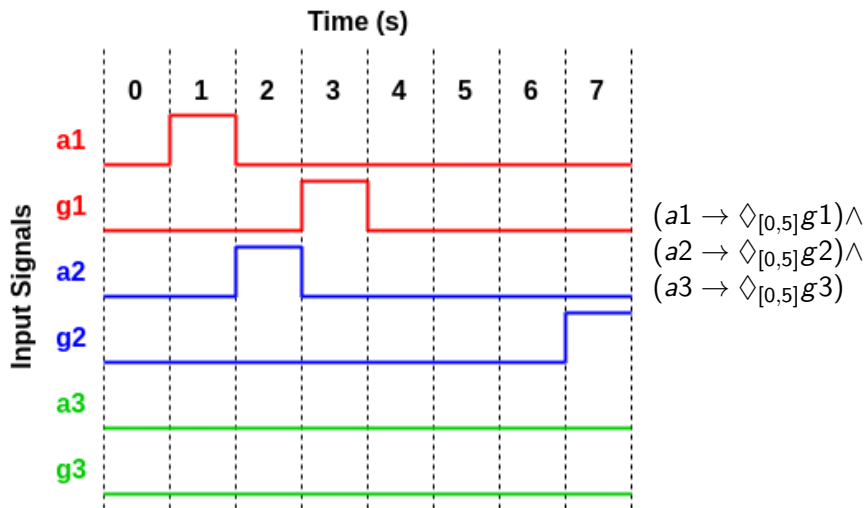
**FT Assembly:**

```
n0 load a0
n1 end n0 f0
n2 load a1
n3 load a2
```

# R2U2 Internal Improvements

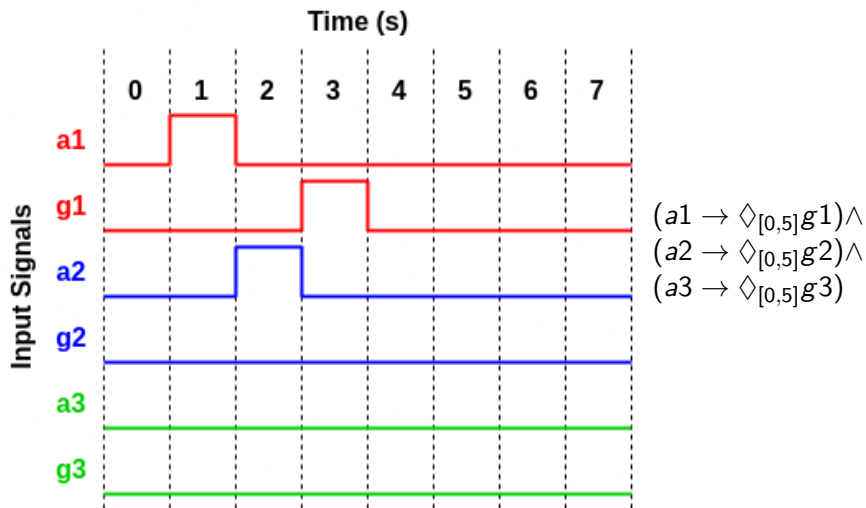


## R2U2 By Example: Passing Case<sup>2</sup>



<sup>2</sup>[https://cgjohannsen.com/movies/req\\_grant\\_pass.mp4](https://cgjohannsen.com/movies/req_grant_pass.mp4)

## R2U2 By Example: Failure Case<sup>3</sup>



<sup>3</sup>[https://cgjohannsen.com/movies/req\\_grant\\_fail.mp4](https://cgjohannsen.com/movies/req_grant_fail.mp4)

## R2U2 By Example: Map File

`https://cgjohannsen.com/movies/req\_grant\_map.mp4`