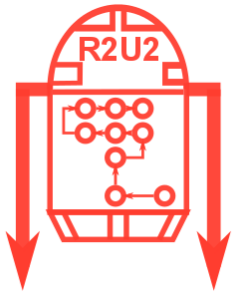
R2U2 V3.0

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

Chris Johannsen[1], Phillip Jones[1], Brian Kempa[1], Kristin Yvonne Rozier[1], Pei Zhang[2]



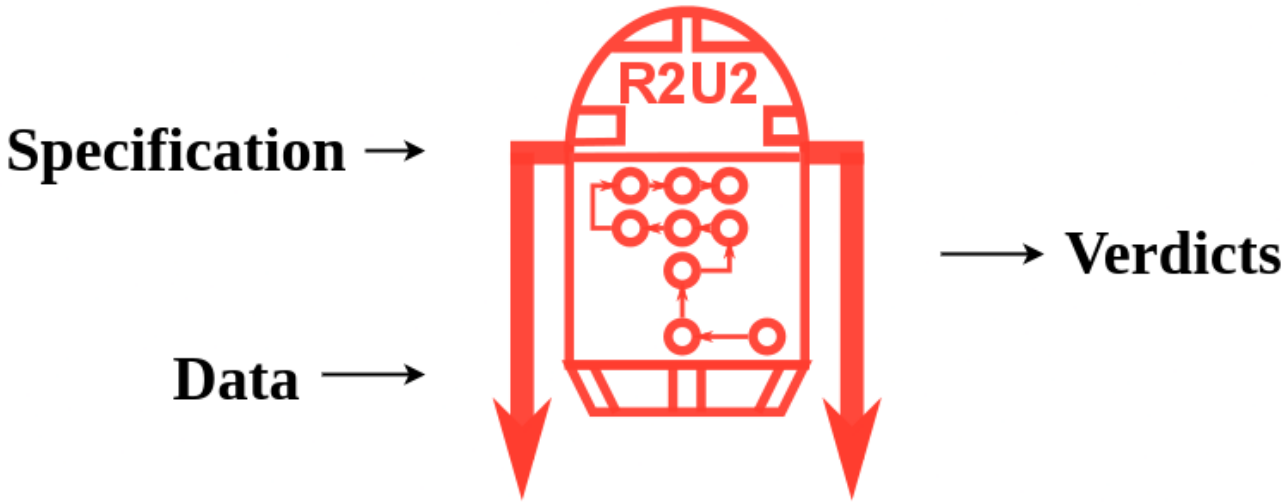
[1]Iowa State University
cgjohann@iastate.edu

[2]Google LLC

35th International Conference on Computer Aided Verification
July 2023

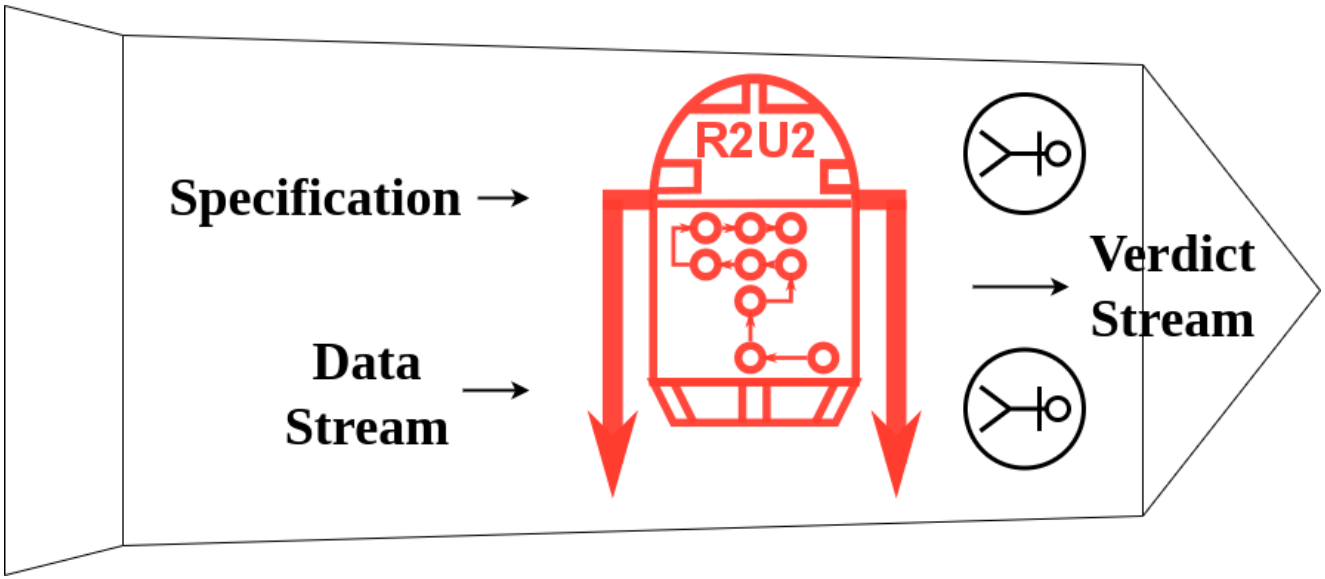
https://r2u2.temporallogic.org/

What Does R2U2 Do?



Runtime Verification

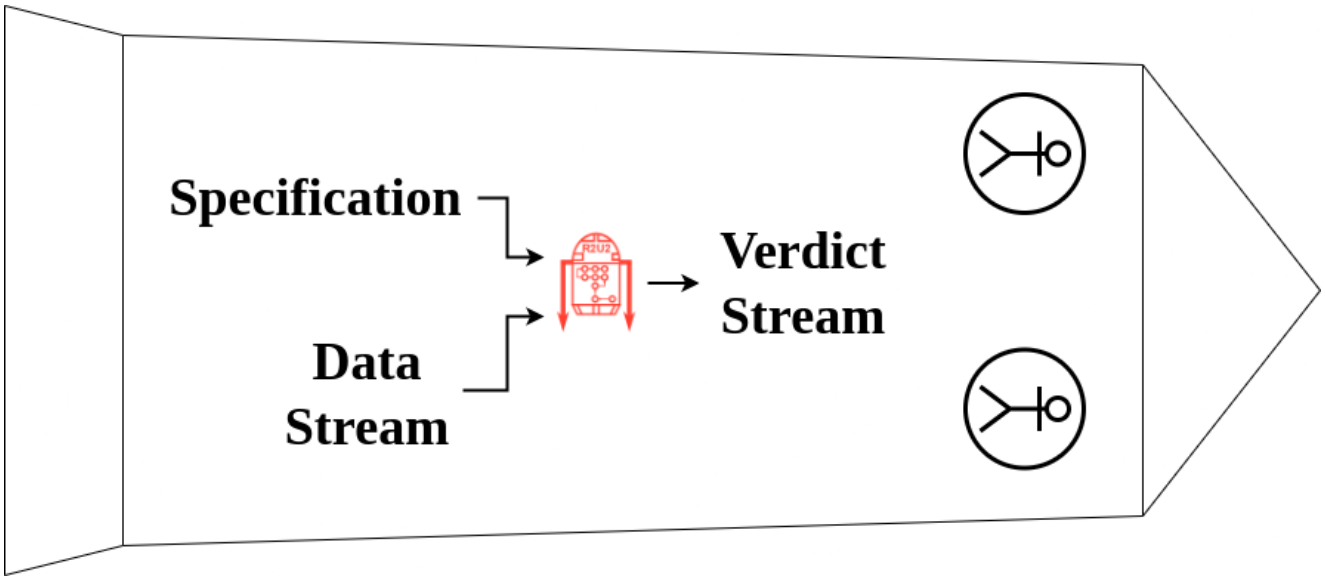
What Does R2U2 Do?



~~Runtime Verification~~

Real-time Runtime Verification

What Does R2U2 Do?

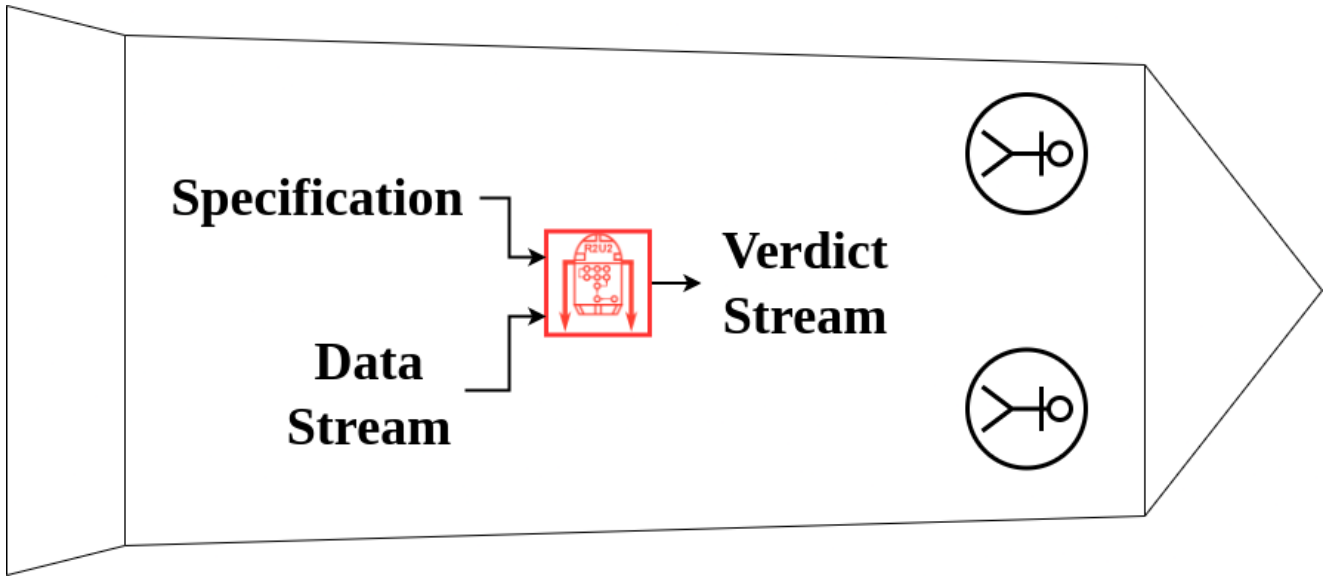


~~Runtime Verification~~

~~Real-time Runtime Verification~~

Resource-constrained Real-time Runtime Verification

What Does R2U2 Do?



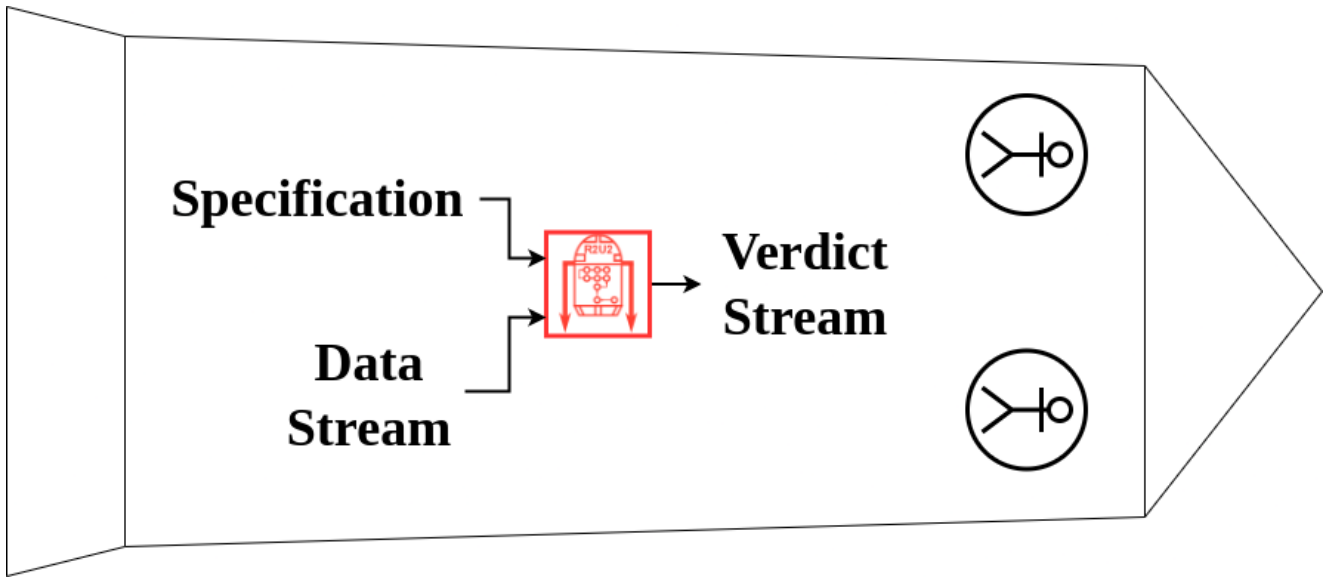
~~Runtime Verification~~

~~Real-time Runtime Verification~~

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

Unobtrusive Resource-constrained Real-time Runtime Verification

What Does R2U2 Do?



~~Runtime Verification~~

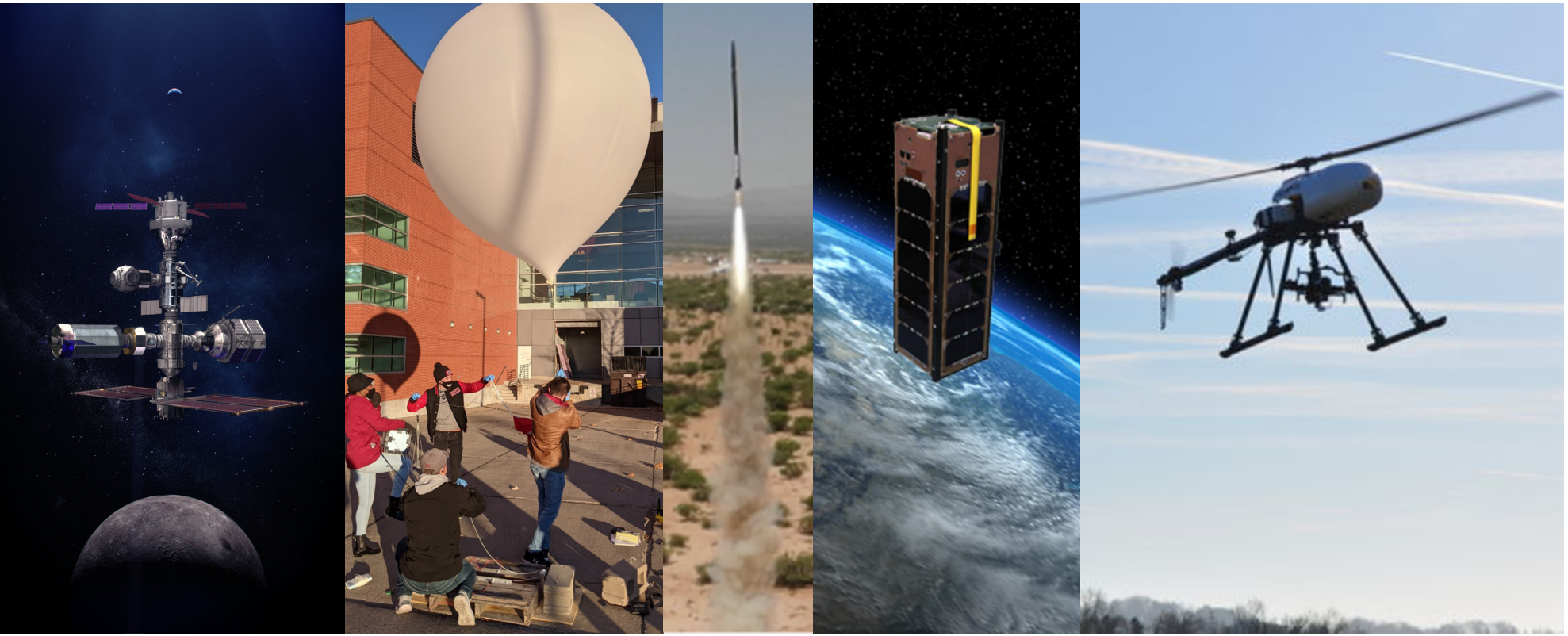
~~Real-time Runtime Verification~~

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

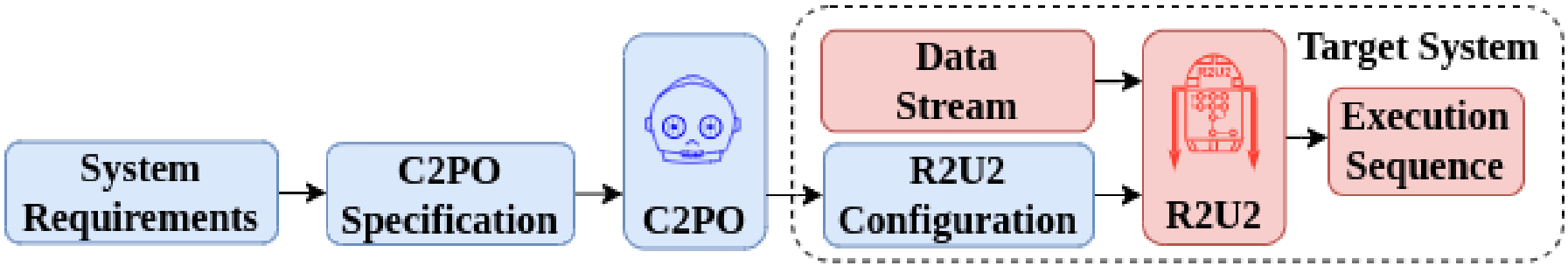
Unobtrusive Resource-constrained Real-time Runtime Verification

Realizable, Responsive, Unobtrusive Unit (R2U2)

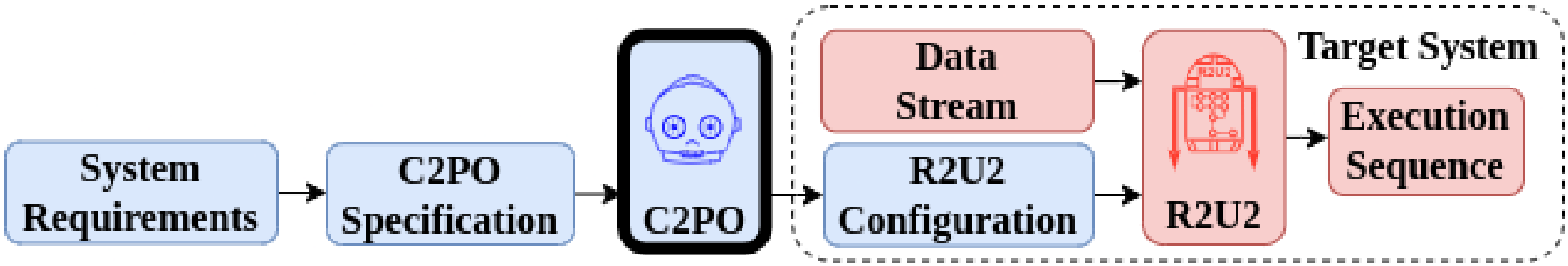
Flight-Certifible Runtime Verification



R2U2 Toolchain Overview

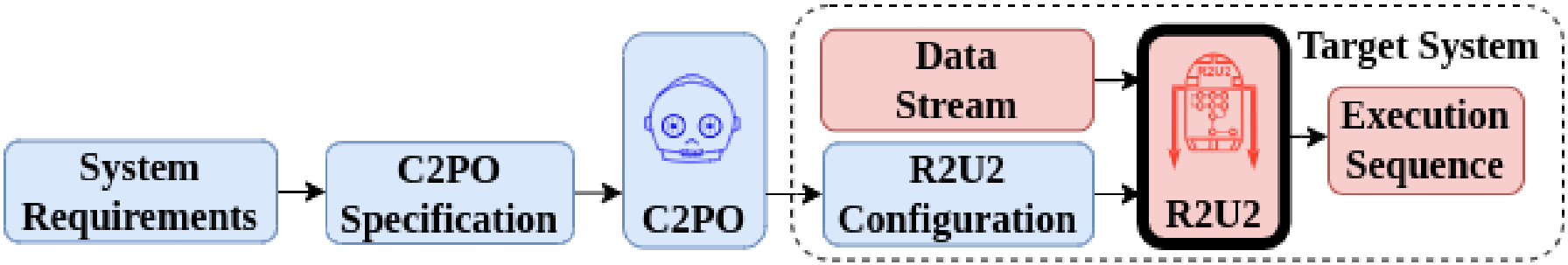


R2U2 Toolchain Overview



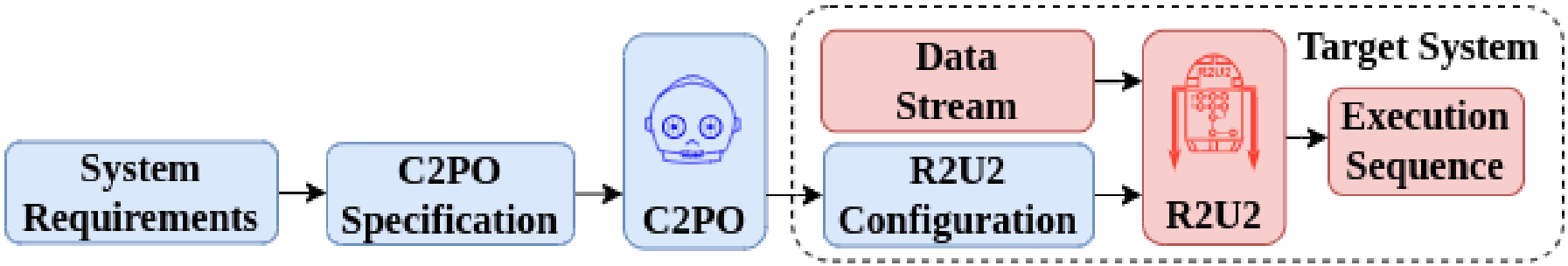
- **Python: Configuration Compiler for Property Organization (C2PO)**

R2U2 Toolchain Overview



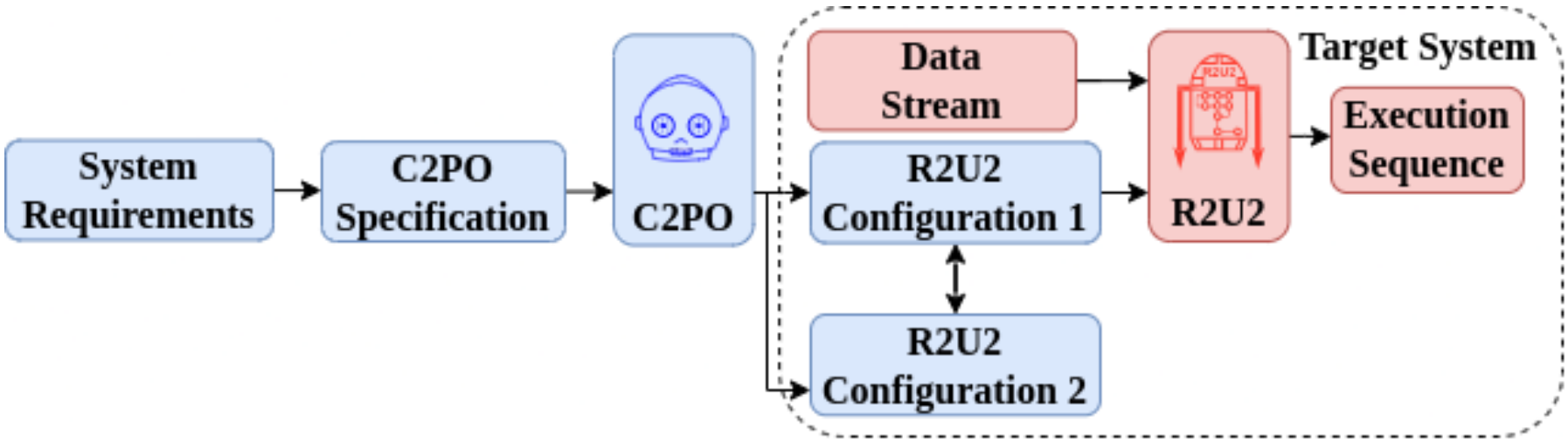
- Python: Configuration Compiler for Property Organization (C2PO)
- **C: Observer (R2U2)**

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};
```

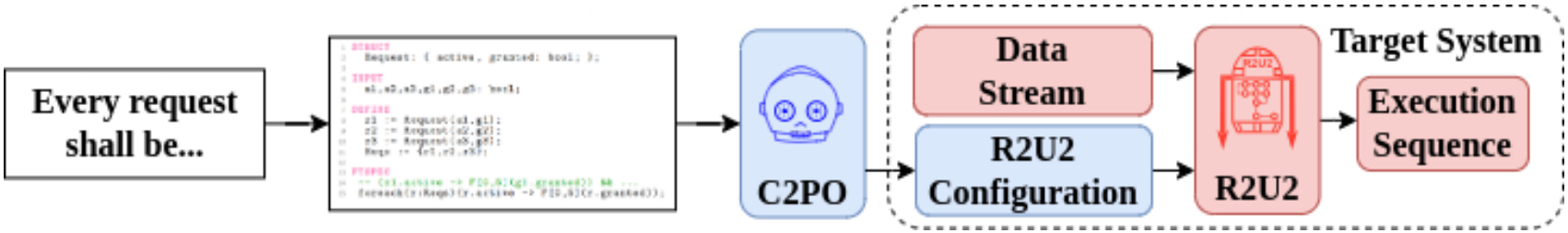
R2U2 Example: Arbiter

English Requirement

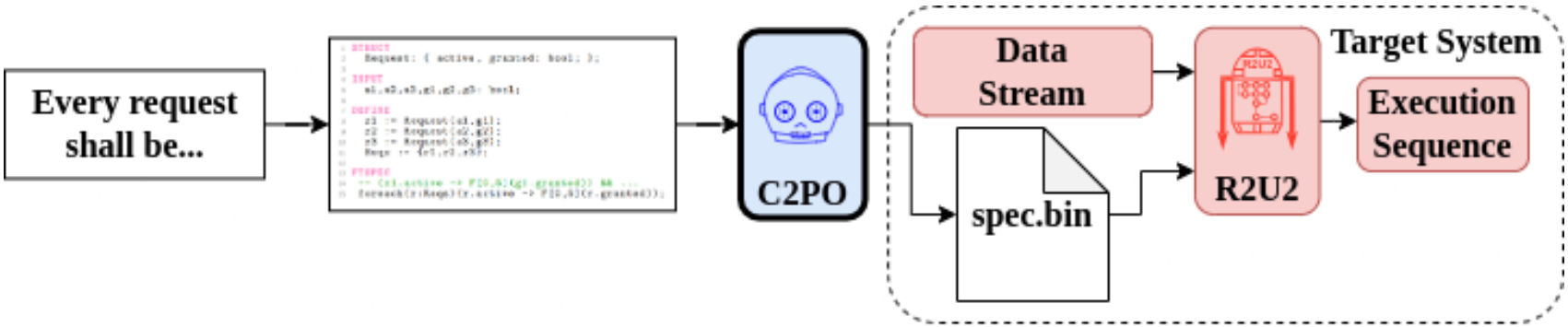
Every request shall be granted within 5 seconds where only 3 requests can be queued at once.

```
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));
```

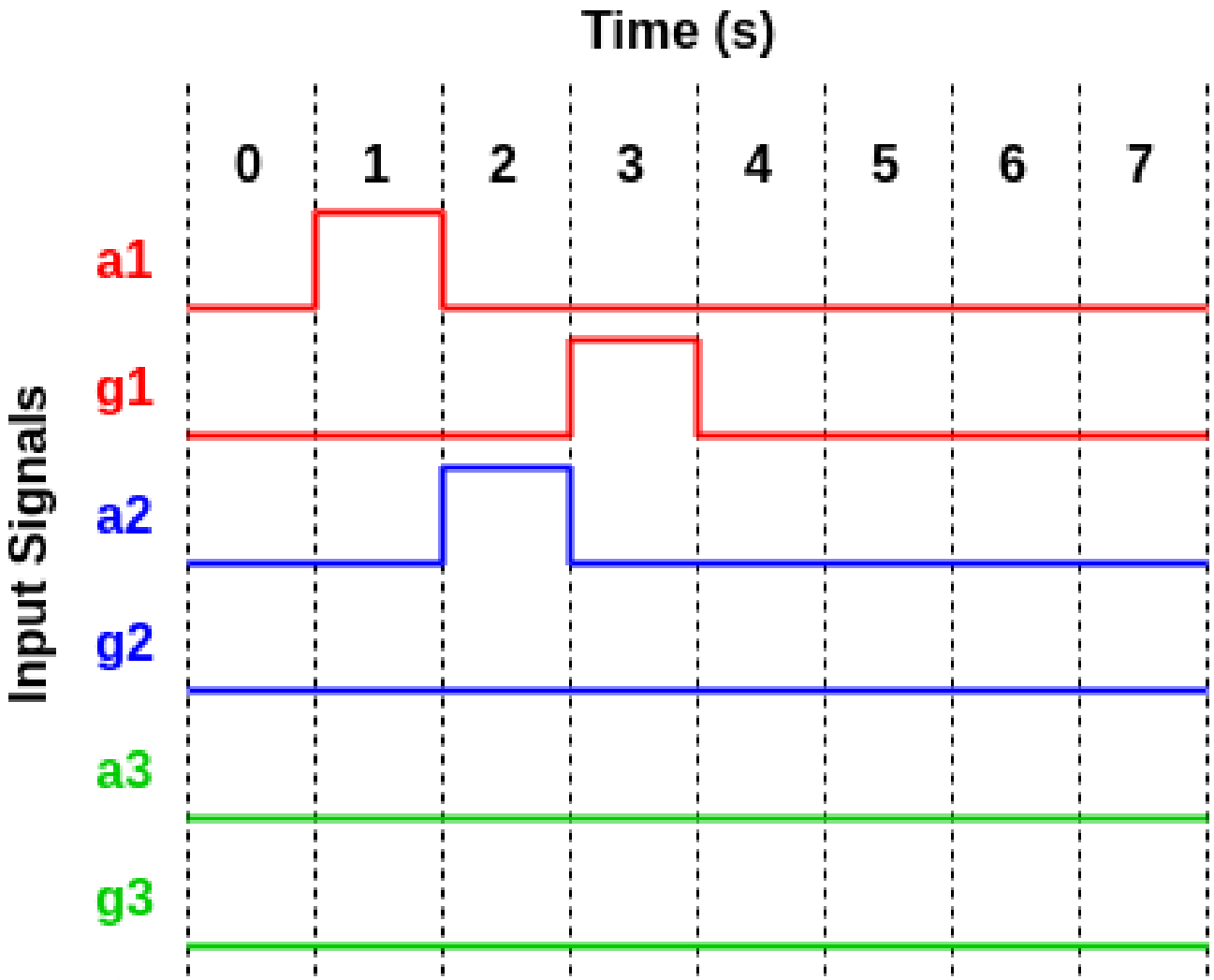
R2U2 Example: Arbiter



R2U2 Example: Arbiter

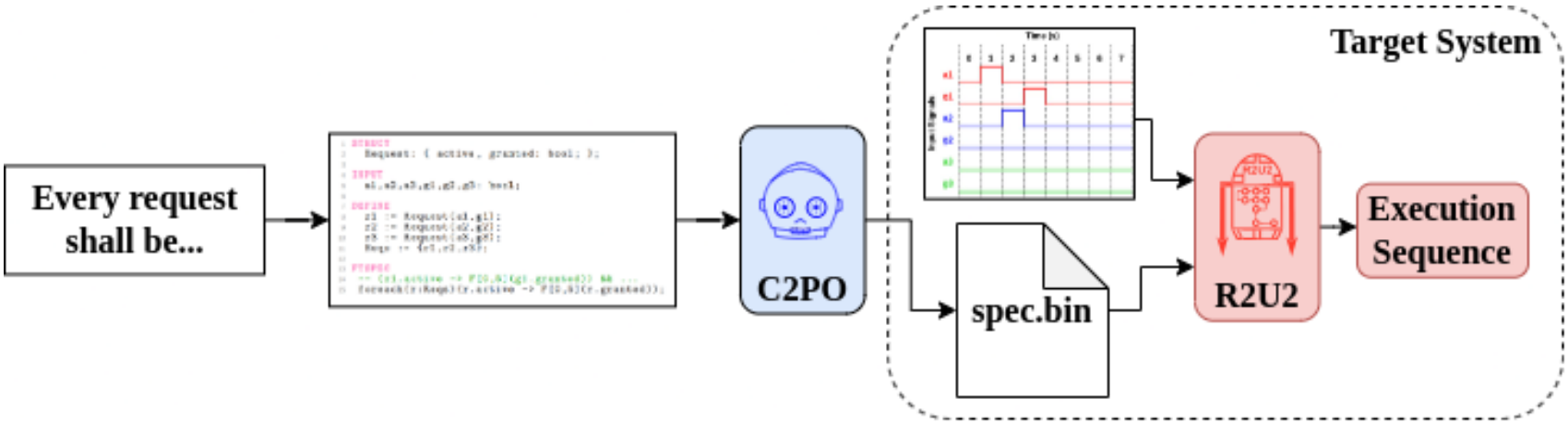


R2U2 Example: Arbiter

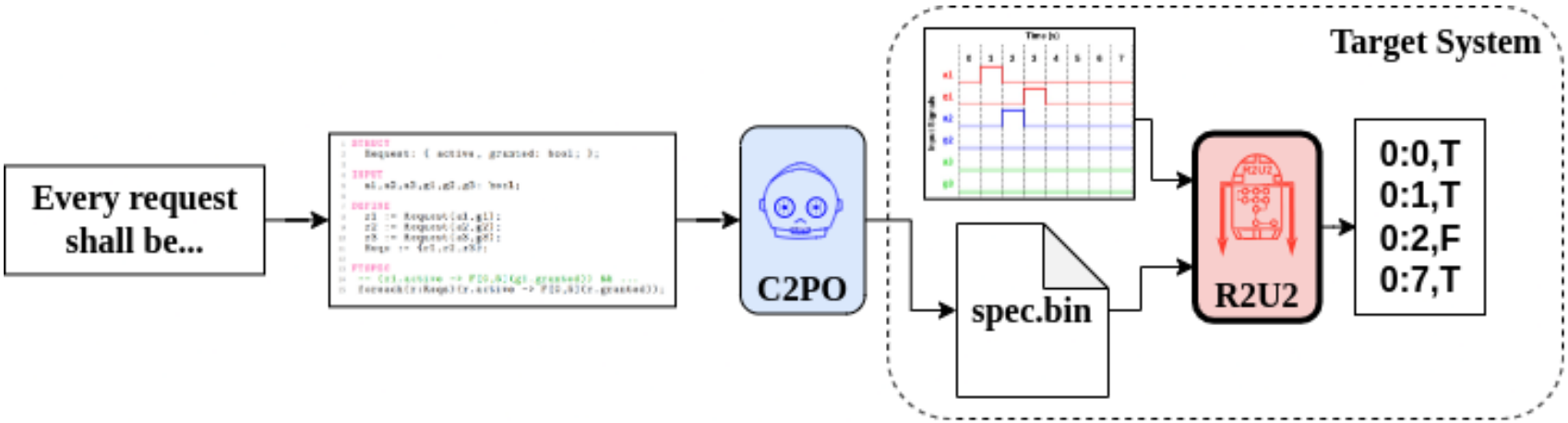


$$(a1 \rightarrow \Diamond_{[0,5]} g1) \wedge$$
$$(a2 \rightarrow \Diamond_{[0,5]} g2) \wedge$$
$$(a3 \rightarrow \Diamond_{[0,5]} g3)$$

R2U2 Example: Arbiter



R2U2 Example: Arbiter



R2U2 V3.0

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**

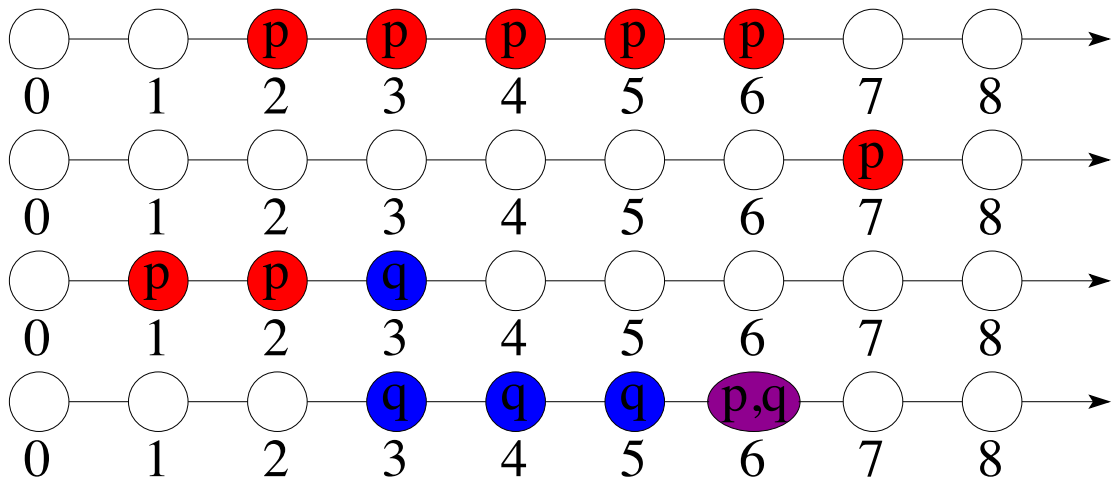
`https://r2u2.temporallogic.org/`

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
$\Box_{[2,6]} p$	$\textsc{Always}_{[2,6]}$
$\Diamond_{[0,7]} p$	$\textsc{Eventually}_{[0,7]}$
$p \mathcal{U}_{[1,5]} q$	$\textsc{Until}_{[1,5]}$
$p \mathcal{R}_{[3,8]} q$	$\textsc{Release}_{[3,8]}$



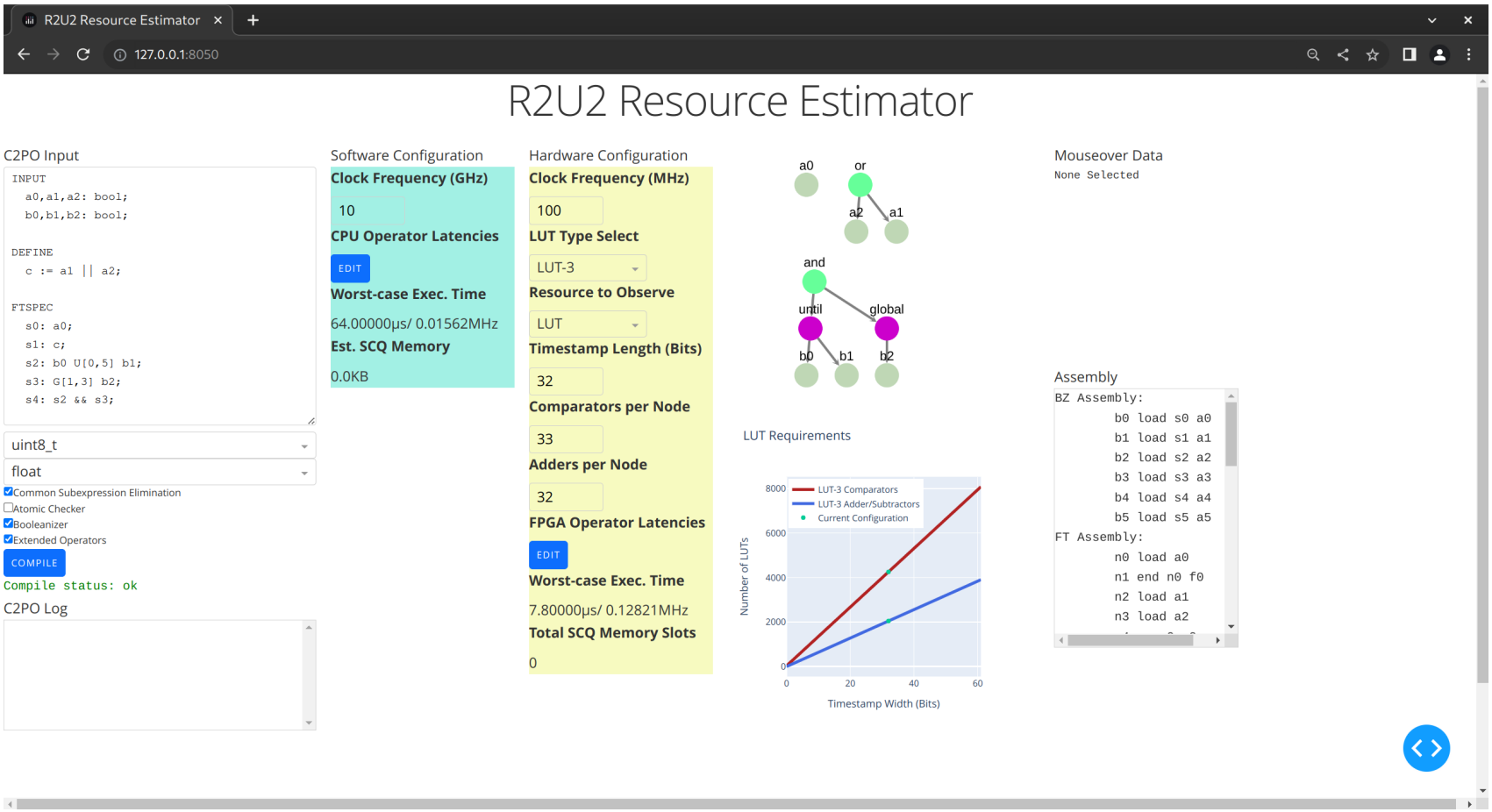
Specification Logic: Mission-time Linear Temporal Logic[1]

Some important notes:

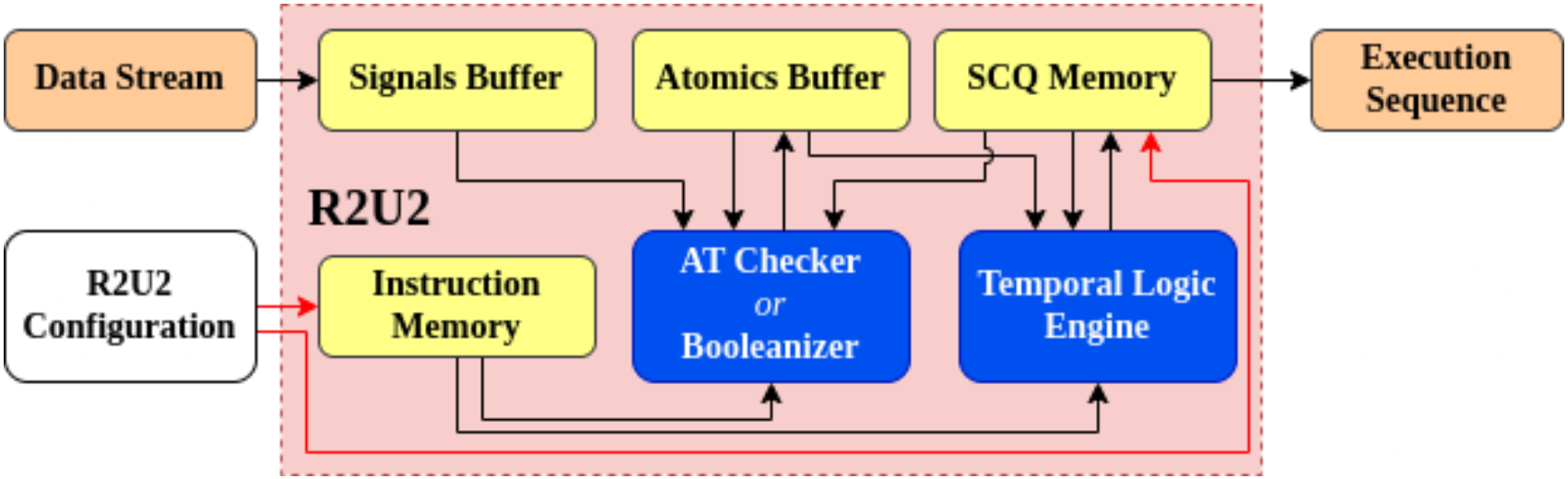
- Finite traces
- Finite intervals
- **U-semantics**: $\pi \models \varphi \, \mathcal{U}_{[a,b]} \psi$ iff $|\pi| > a$ and, $\exists i \in [a,\ b], i < |\pi|$ such that $\pi, i \models \psi$ and $\forall j \in [a,\ b], j < i$ it holds that $\pi, j \models \varphi$
- Intervals are closed, unit-less (generic)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

[1] Li, Vardi, Rozier. "Satisfiability checking for mission-time LTL." CAV, 2019.

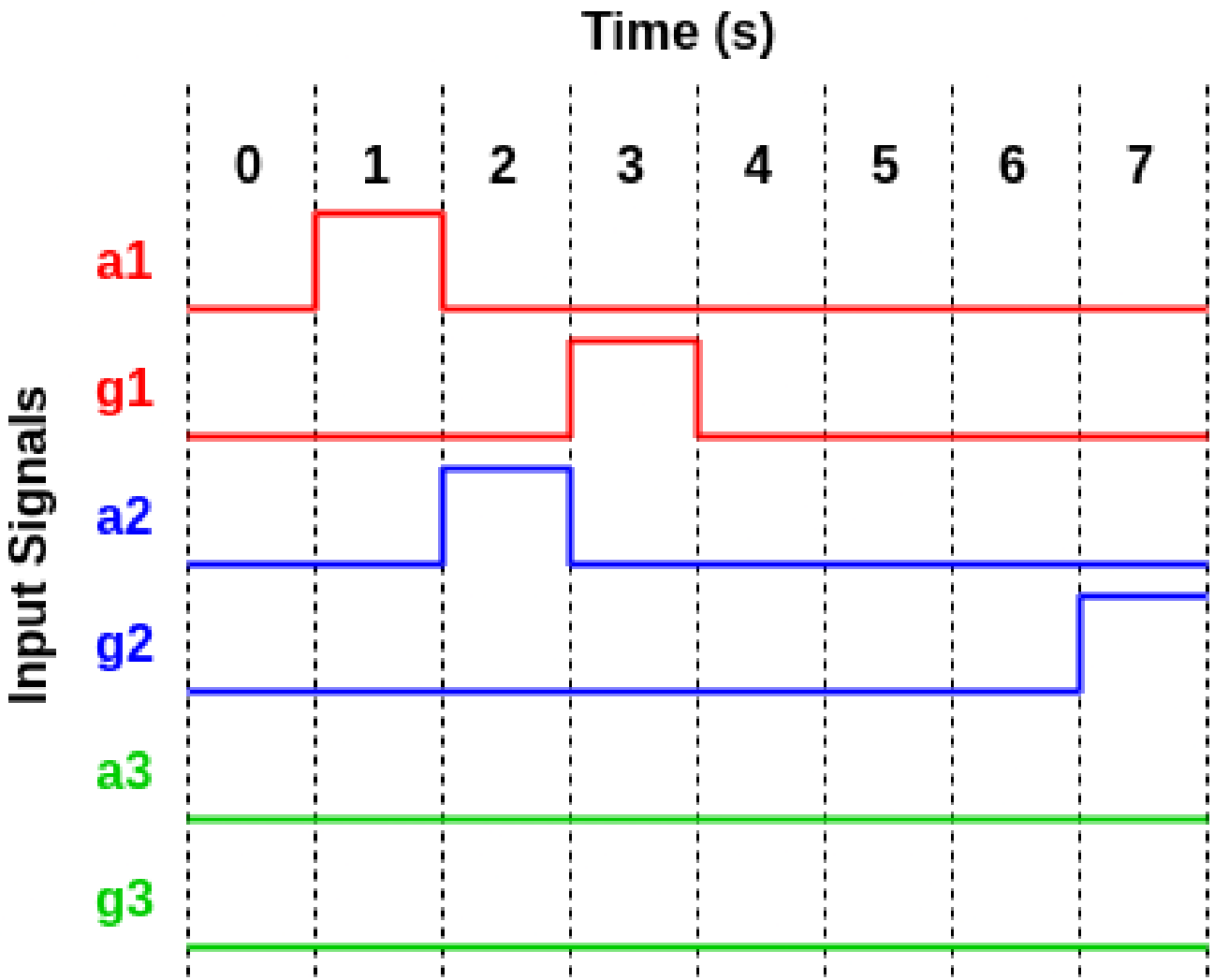
Resource Estimation GUI



R2U2 Internal Improvements



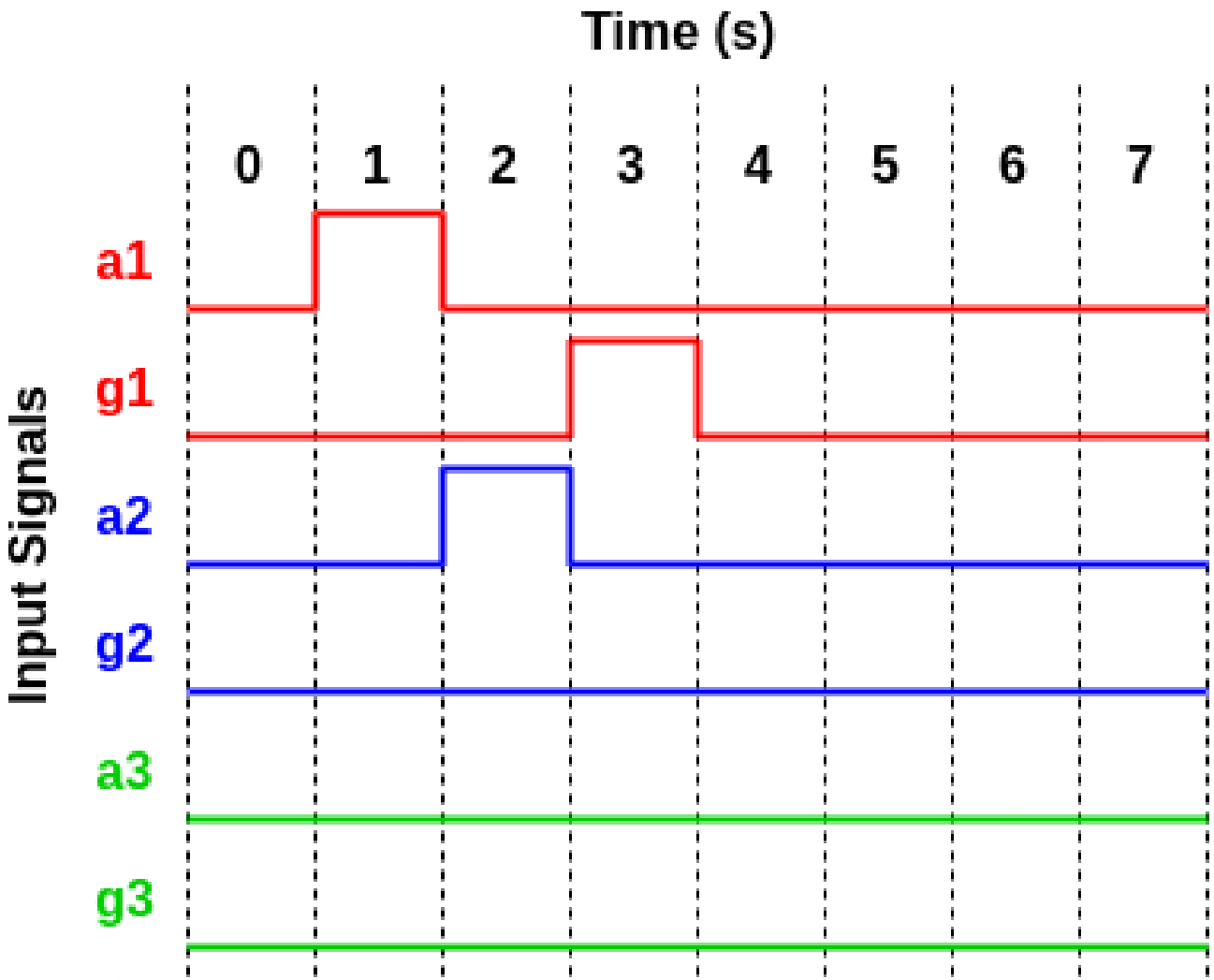
R2U2 By Example: Passing Case[2]



$$(a1 \rightarrow \Diamond_{[0,5]} g1) \wedge$$
$$(a2 \rightarrow \Diamond_{[0,5]} g2) \wedge$$
$$(a3 \rightarrow \Diamond_{[0,5]} g3)$$

[2] https://cgjohannsen.com/movies/req_grant_pass.mp4

R2U2 By Example: Failure Case[3]



$$(a1 \rightarrow \Diamond_{[0,5]} g1) \wedge$$
$$(a2 \rightarrow \Diamond_{[0,5]} g2) \wedge$$
$$(a3 \rightarrow \Diamond_{[0,5]} g3)$$

[3] https://cgjohannsen.com/movies/req_grant_fail.mp4

R2U2 By Example: Map File

https://cgjohannsen.com/movies/req_grant_map.mp4