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?

Specification → R2U2 → Verdicts

Data → R2U2 → Verdicts

Runtime Verification
What Does R2U2 Do?

Runtime Verification
Real-time Runtime Verification

R2U2 V3.0  
CAV 2023
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?

Realizable, Responsive, Unobtrusive Unit (R2U2)
Flight-Certifiable Runtime Verification
R2U2 Toolchain Overview

System Requirements → C2PO Specification → C2PO → R2U2 Configuration → Data Stream → R2U2 → Target System

Execution Sequence
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.

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

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

```plaintext
1 STRUCT
2    Request: { active, granted: bool; };
3
4 INPUT
5    a1, a2, a3, g1, g2, g3: bool;
```
Every request shall be granted within 5 seconds where only 3 requests can be queued at once.

```plaintext
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};
```
**English Requirement**

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

```plaintext
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

Every request shall be...

C2PO

Data Stream

R2U2 Configuration

R2U2

Target System

Execution Sequence
Every request shall be...

C2PO

spec.bin

Data Stream

R2U2

Target System

Execution Sequence
R2U2 Example: Arbiter

\[(a_1 \rightarrow \Diamond [0,5] g_1) \land (a_2 \rightarrow \Diamond [0,5] g_2) \land (a_3 \rightarrow \Diamond [0,5] g_3)\]
R2U2 Example: Arbiter

Every request shall be...

C2PO

spec.bin

R2U2

Target System

Execution Sequence
R2U2 Example: Arbiter

Every request shall be...

C2PO

Target System

spec.bin

R2U2

0:0,T
0:1,T
0:2,F
0:7,T
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/
Mission-Time Temporal Logic (MLTL) reasons about *integer-bounded, finite* timelines:

- finite set of atomic propositions \{p, q\}
- Boolean connectives: \( \neg, \land, \lor, \text{and} \rightarrow \)
- temporal connectives *with time bounds*:

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Operator</th>
<th>Timeline</th>
</tr>
</thead>
<tbody>
<tr>
<td>(\Box_{[2,6]}p)</td>
<td><strong>ALWAYS</strong>([2,6])</td>
<td>0 1 2 3 4 5 6 7 8</td>
</tr>
<tr>
<td>(\Diamond_{[0,7]}p)</td>
<td><strong>EVENTUALLY</strong>([0,7])</td>
<td>0 1 2 3 4 5 6 7 8</td>
</tr>
<tr>
<td>(p\mathcal{U}_{[1,5]}q)</td>
<td><strong>UNTIL</strong>([1,5])</td>
<td>0 1 2 3 4 5 6 7 8</td>
</tr>
<tr>
<td>(p\mathcal{R}_{[3,8]}q)</td>
<td><strong>RELEASE</strong>([3,8])</td>
<td>0 1 2 3 4 5 6 7 8</td>
</tr>
</tbody>
</table>
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

---

Resource Estimation GUI

R2U2 Resource Estimator

C2PO Input

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

DEFINE
c := a1 || a2;

PTMPEC
a0: a0;
a1: or;
a2: b0 U(0.5) b1;
a3: 0(1.3) b2;
a4: a2 && a3;

uint8_t
float

Common Subexpression Elimination
Atomic Checker
Bitvectorizer
Extended Operators

Compile status: ok
C2PO Log

Software Configuration

Clock Frequency (GHz)
10

CPU Operator Latencies

EDIT
Worst-case Exec. Time
64.000000µs / 0.01562MHz
Est. SQM 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

EDIT
Worst-case Exec. Time
7.800000µs / 0.12821MHz

Total SQM Memory Slots
0

LUT Requirements

Number of LUT

0 2000 4000 8000
0 10 20 30 40 50

Timestamp Width (Bits)

Mouseover Data
None Selected

Assembly

BZ Assembly:
b0 load a0 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
n2 load a1
n3 load a2
R2U2 Internal Improvements
R2U2 By Example: Passing Case

\[(a_1 \rightarrow \Diamond [0,5] g_1) \land \]
\[(a_2 \rightarrow \Diamond [0,5] g_2) \land \]
\[(a_3 \rightarrow \Diamond [0,5] g_3)\]

\[2\text{https://cgjohannsen.com/movies/req_grant_pass.mp4}\]
R2U2 By Example: Failure Case

\[ (a_1 \rightarrow \Diamond [0,5]g_1) \land (a_2 \rightarrow \Diamond [0,5]g_2) \land (a_3 \rightarrow \Diamond [0,5]g_3) \]

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