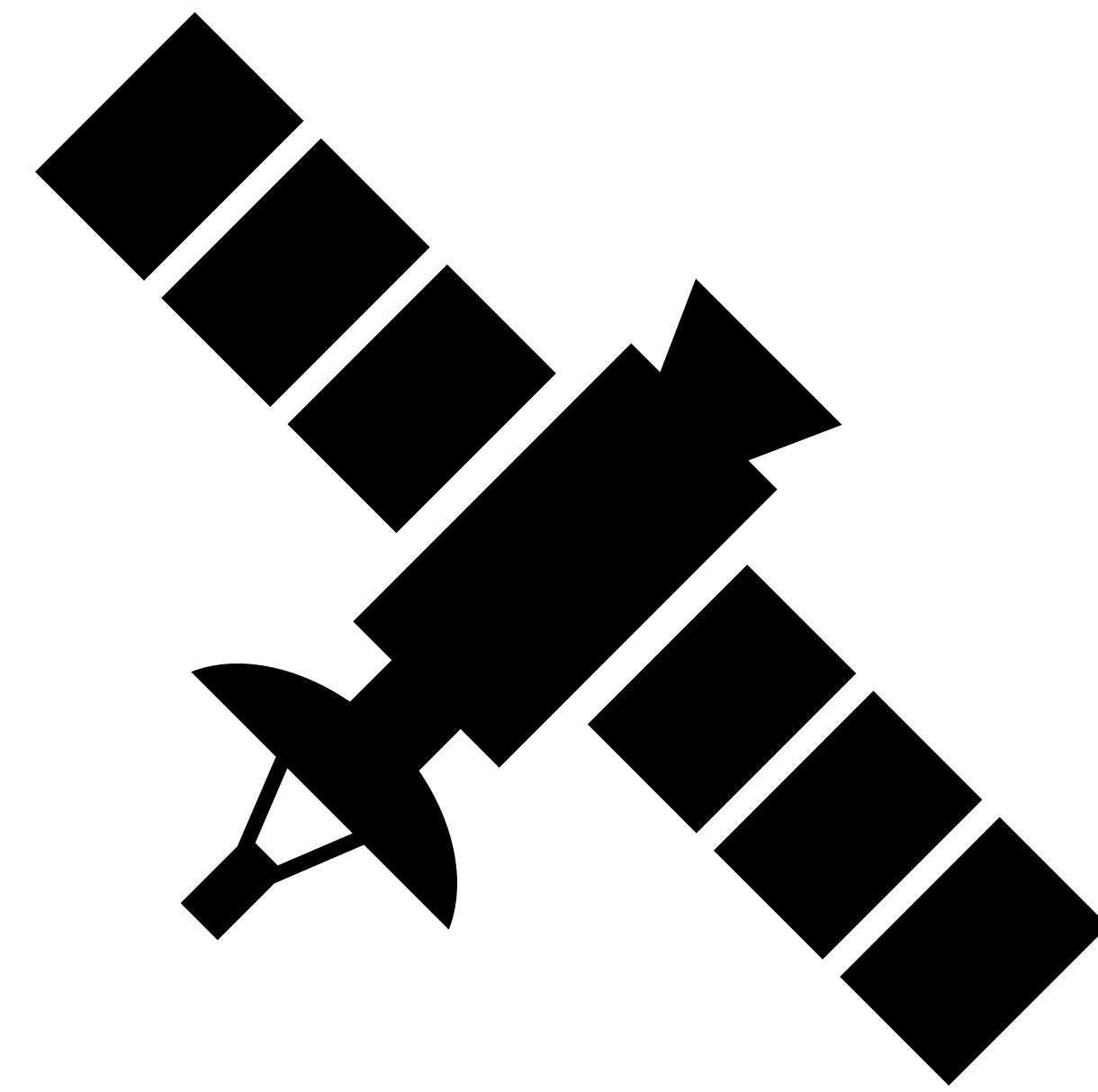
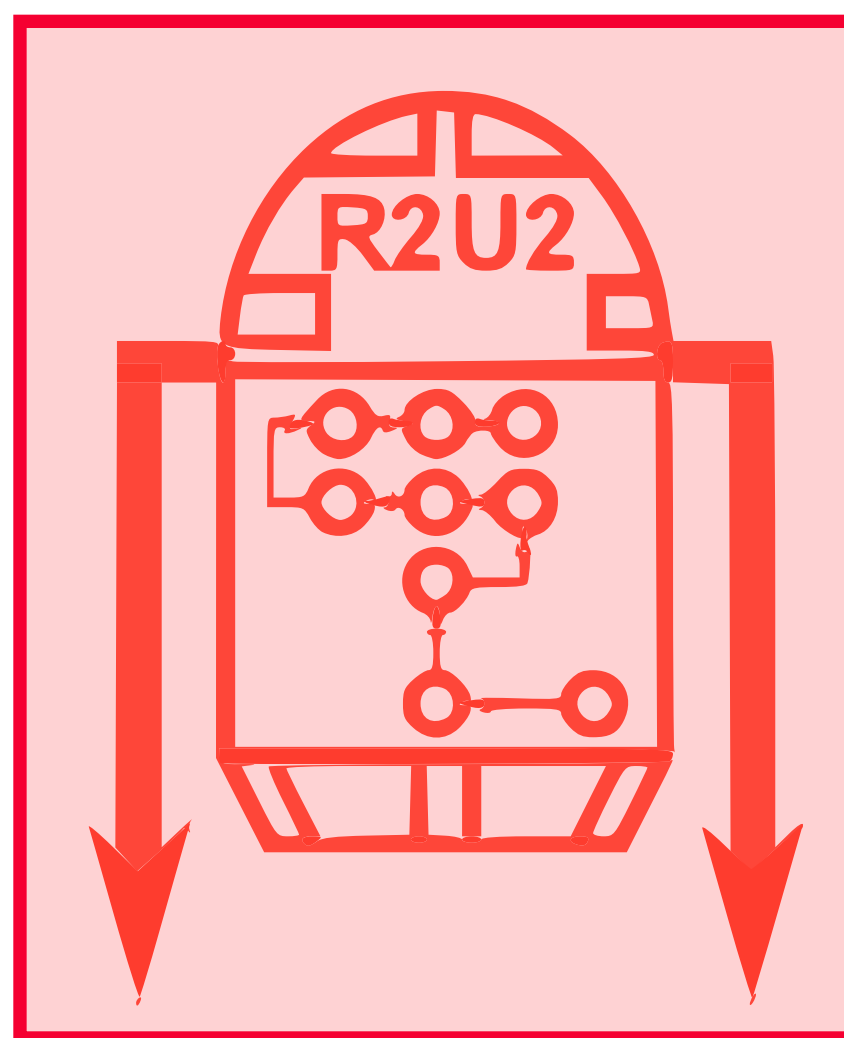


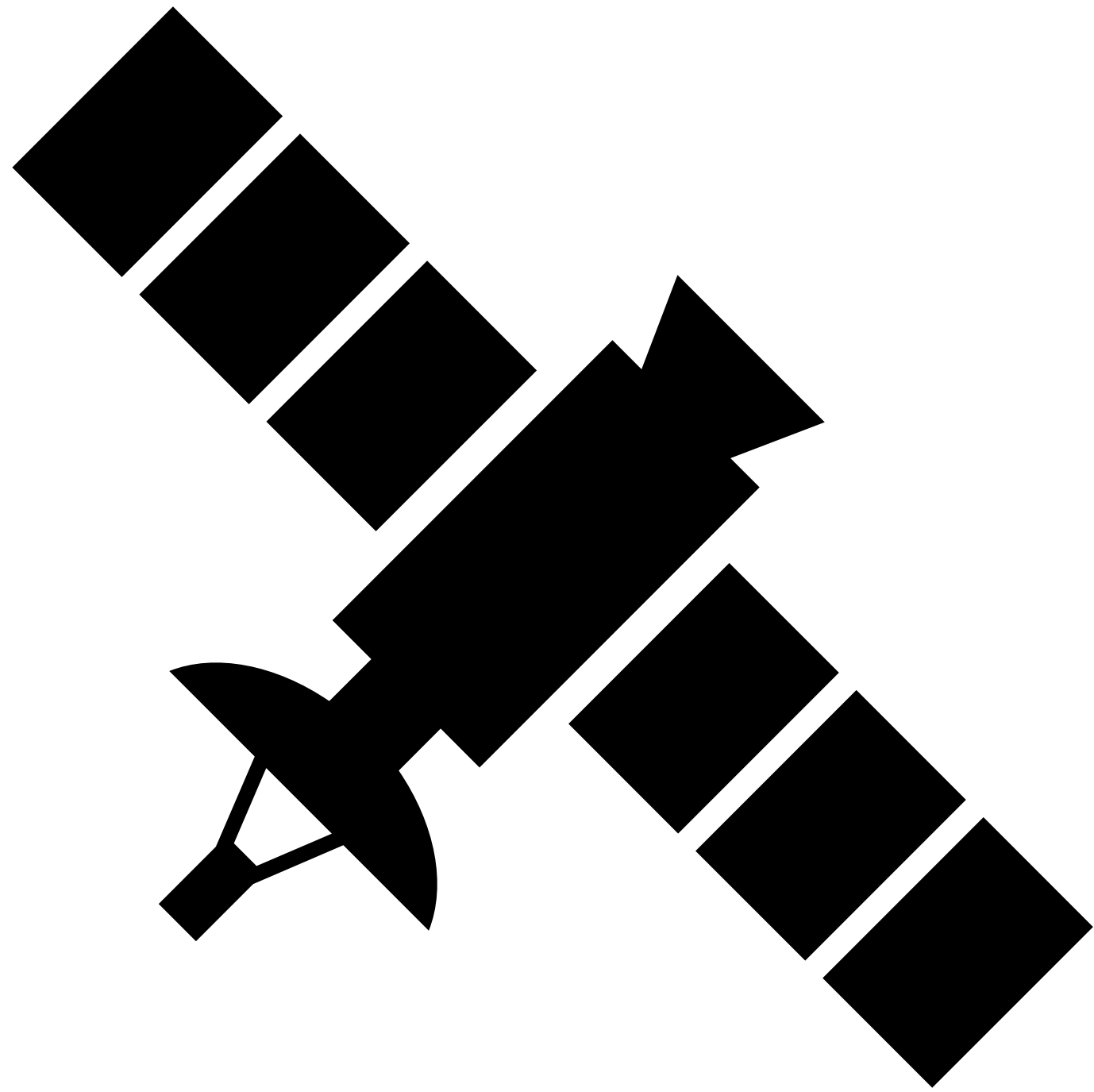
Shrinking Mission-time LTL Monitors with Equality Saturation

Christopher Johannsen and
Kristin Yvonne Rozier

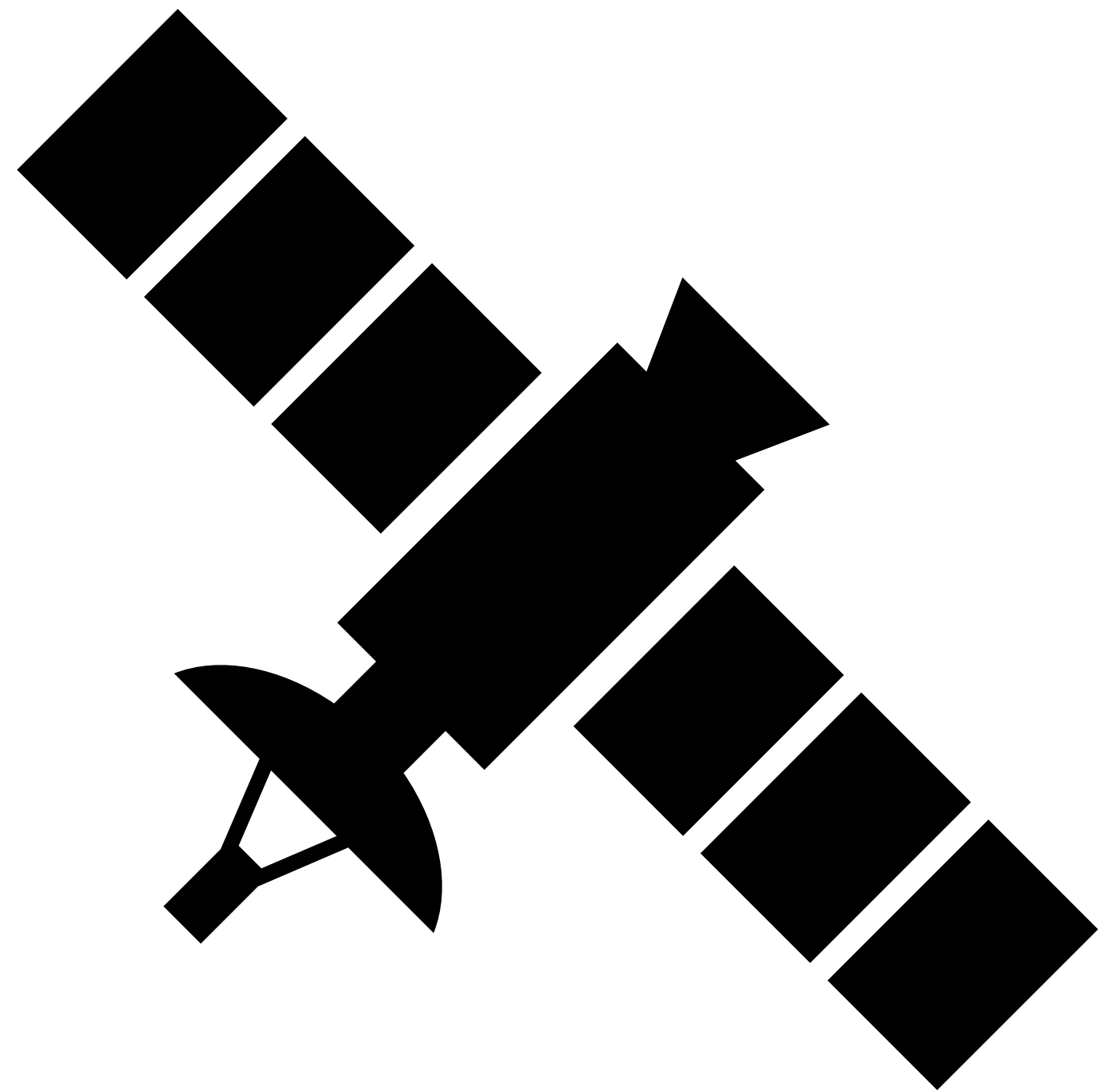
IOWA STATE UNIVERSITY[®]
OF SCIENCE AND TECHNOLOGY



**Small satellite failures are common and difficult to trace,
but runtime verification (RV) can help.**



Small satellite failures are common and difficult to trace, but runtime verification (RV) can help.



Controls

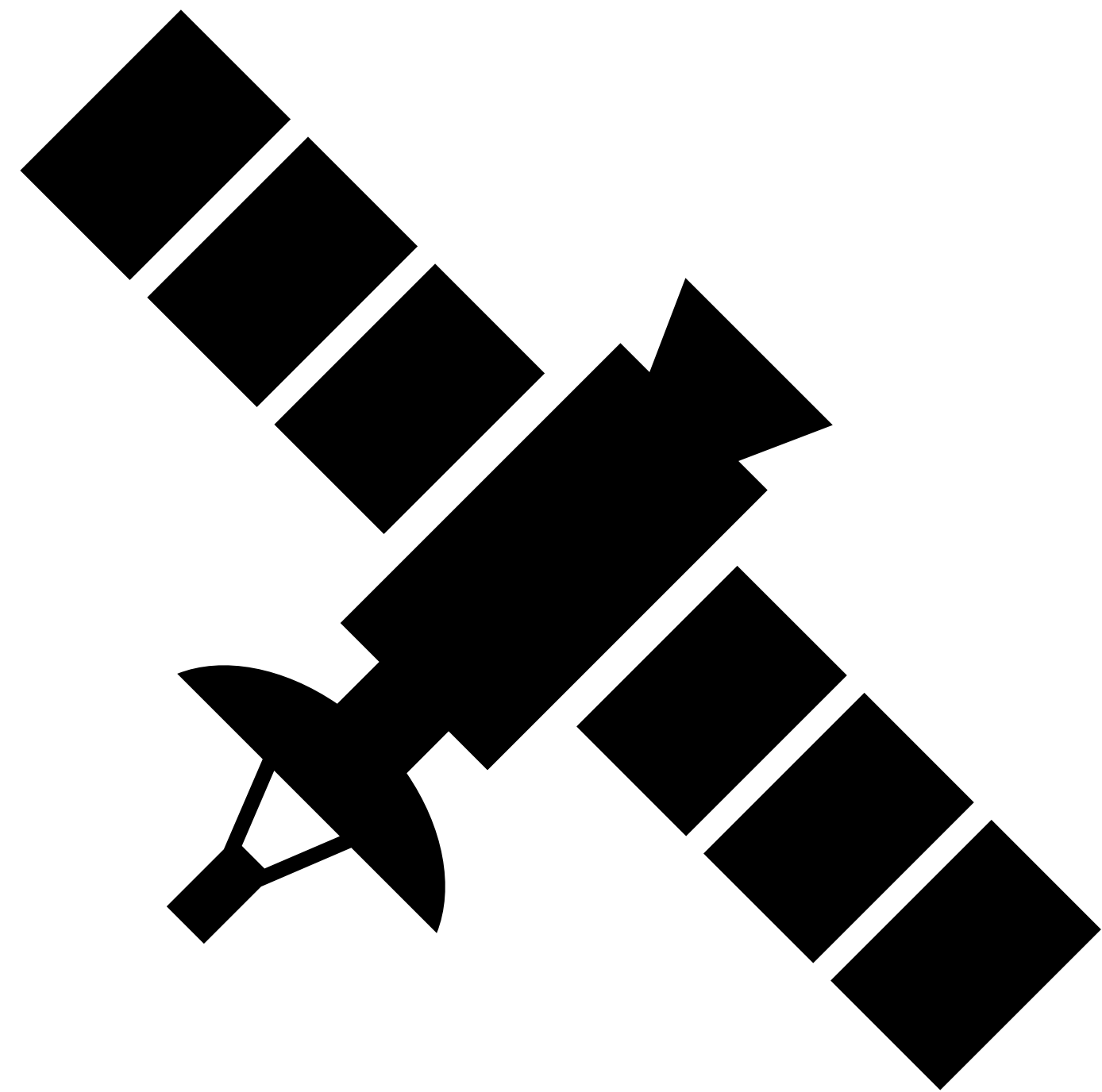
Power Systems

Comms

Scientific

...

Small satellite failures are common and difficult to trace, but runtime verification (RV) can help.



Controls

Power Systems

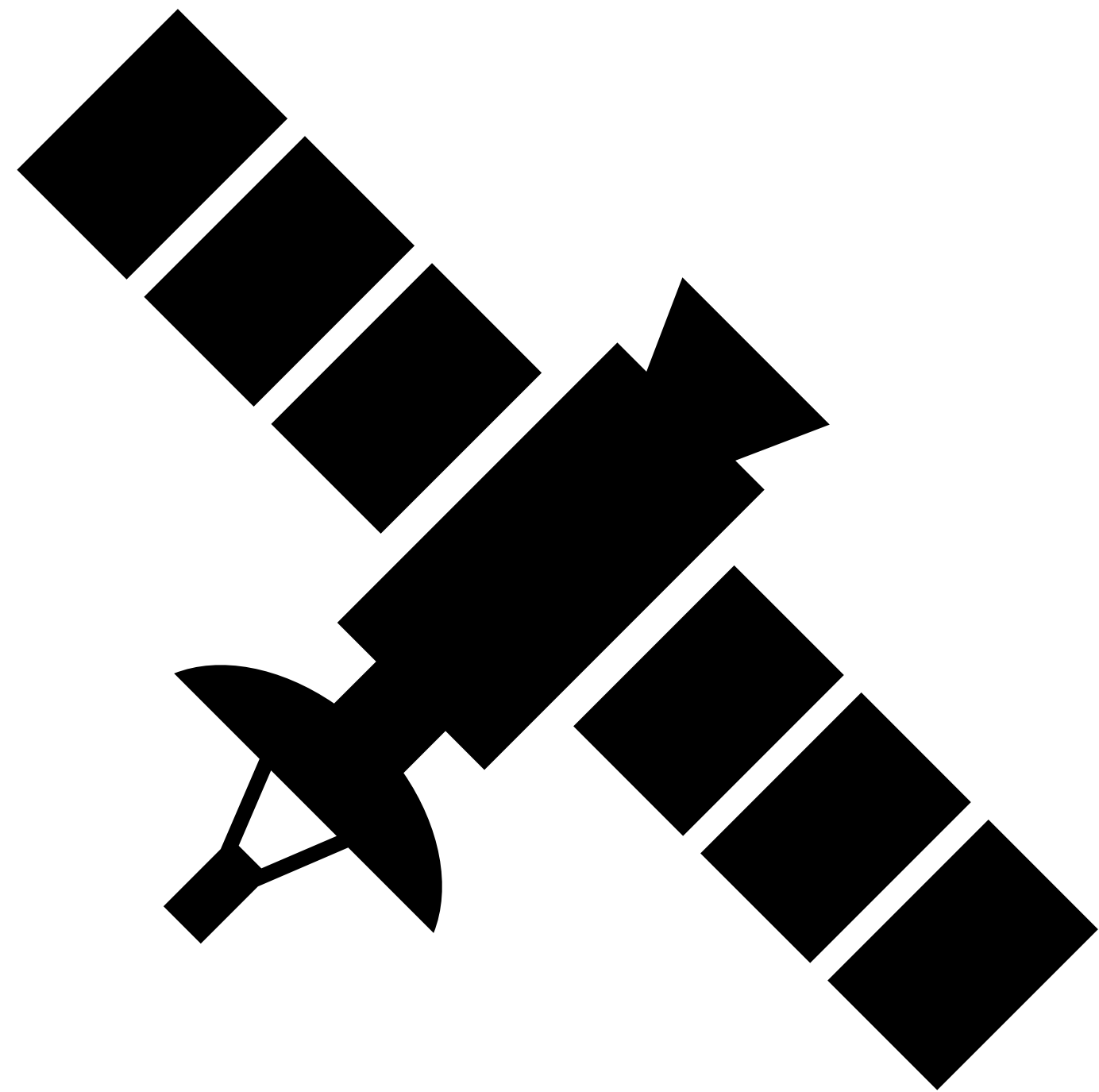
Comms

Scientific

...

Small satellites face a ~14% failure rate
(as of 2019)*

Small satellite failures are common and difficult to trace, but runtime verification (RV) can help.



Controls

Power Systems

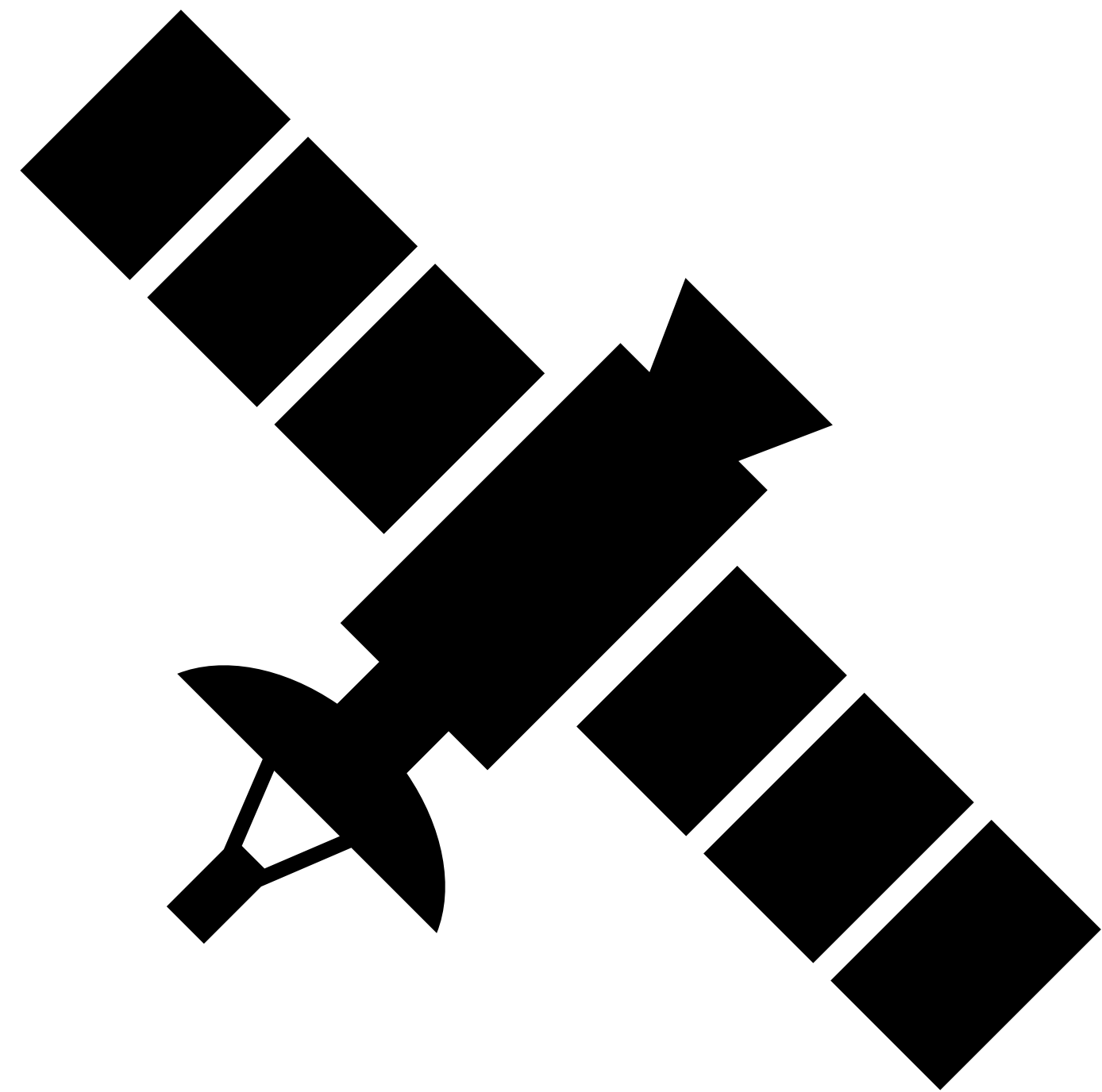
Comms

Scientific

...

“If a command is received, it shall be acknowledged either in the near window (within 5 cycles) or after a 2-cycle bus delay (from 2 to 7 cycles from now)”

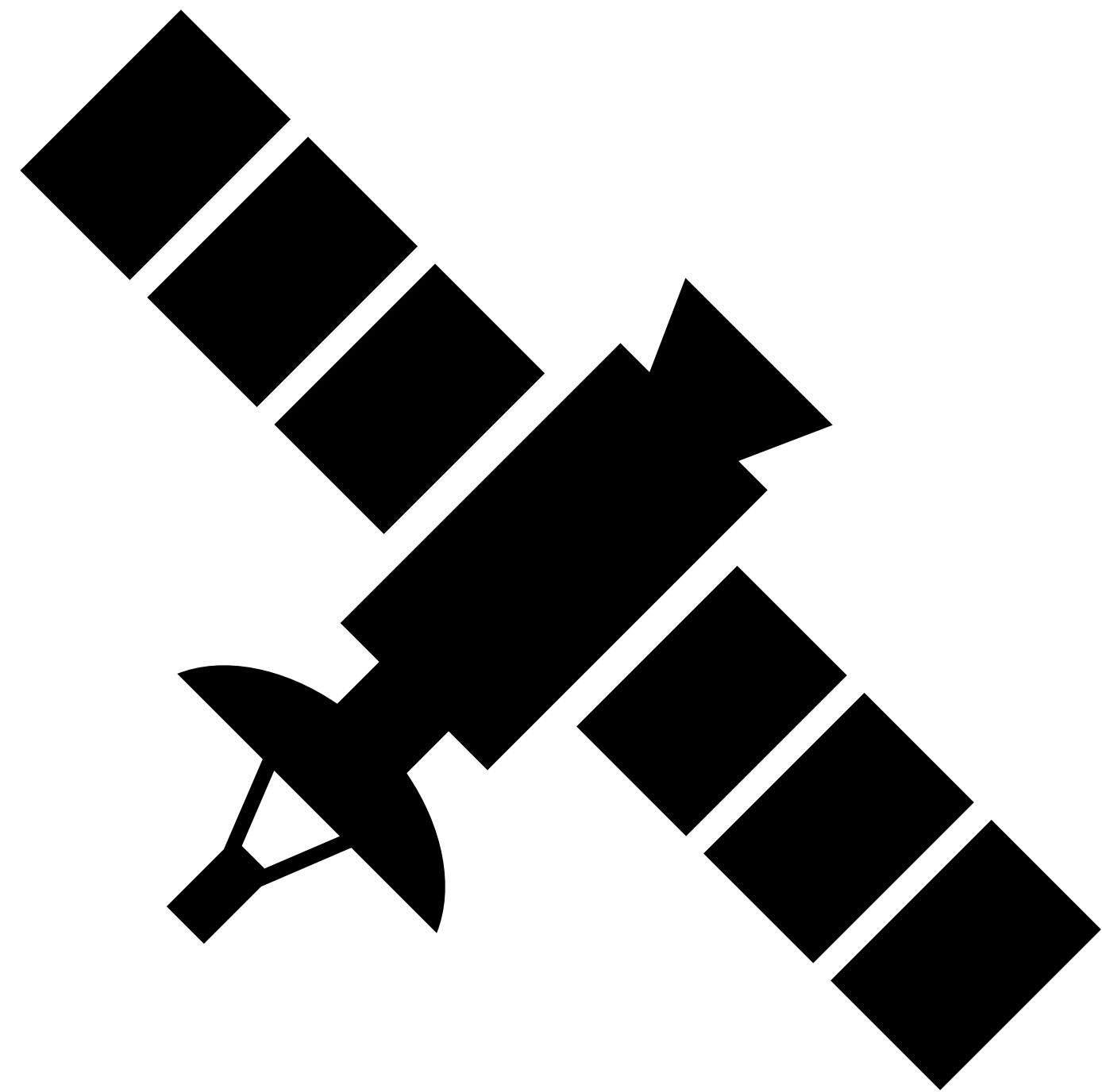
Small satellite failures are common and difficult to trace, but runtime verification (RV) can help.



Controls Power Systems
FAULT Comms Scientific
...

“If a command is received, it shall be acknowledged either in the near window (within 5 cycles) or after a 2-cycle bus delay (from 2 to 7 cycles from now)”

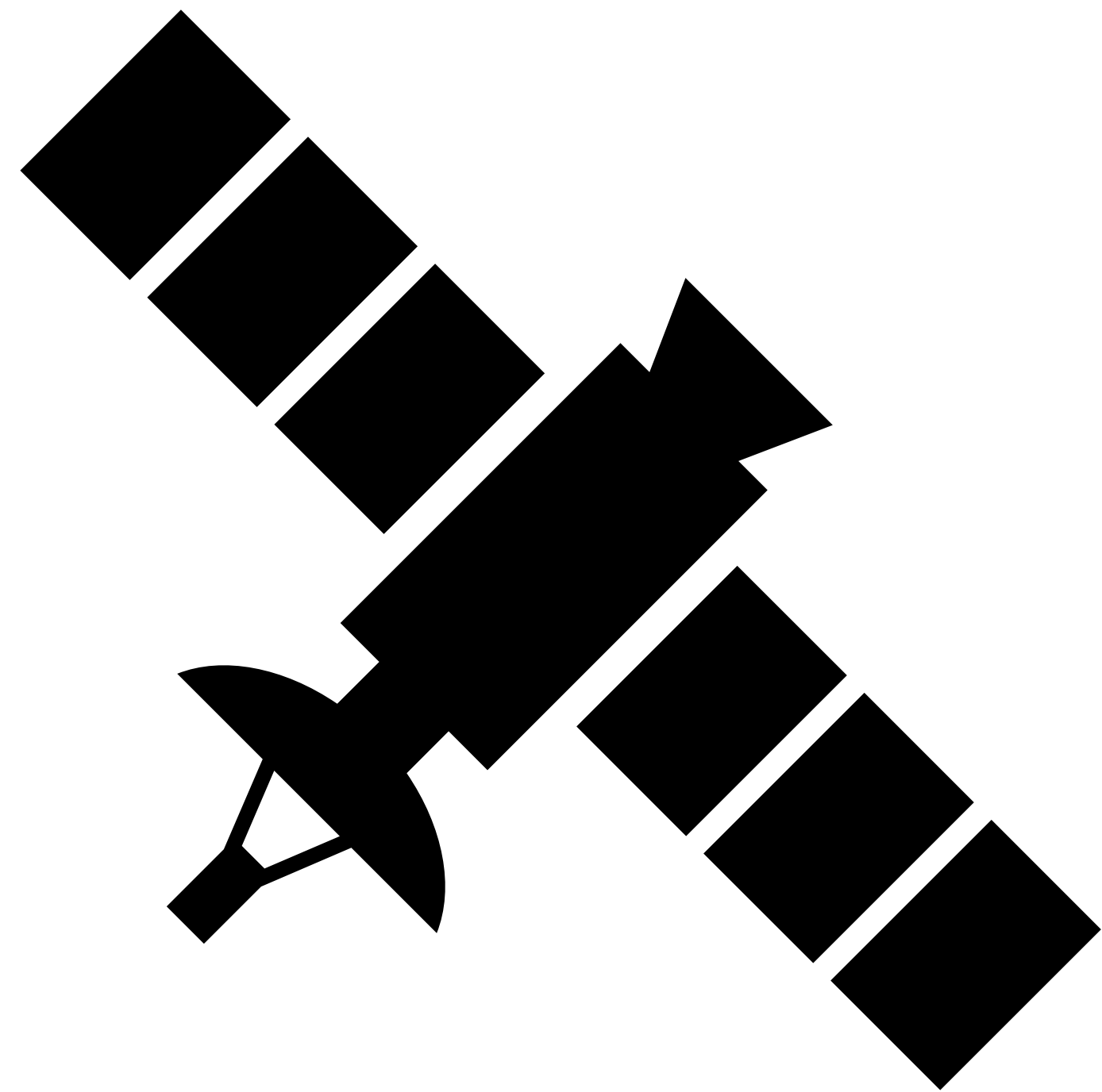
Small satellite failures are common and difficult to trace, but runtime verification (RV) can help.



FAULT	Controls	Power Systems	
FAULT	Comms	Scientific	FAULT
...			

“If a command is received, it shall be acknowledged either in the near window (within 5 cycles) or after a 2-cycle bus delay (from 2 to 7 cycles from now)”

Small satellite failures are common and difficult to trace, but runtime verification (RV) can help.

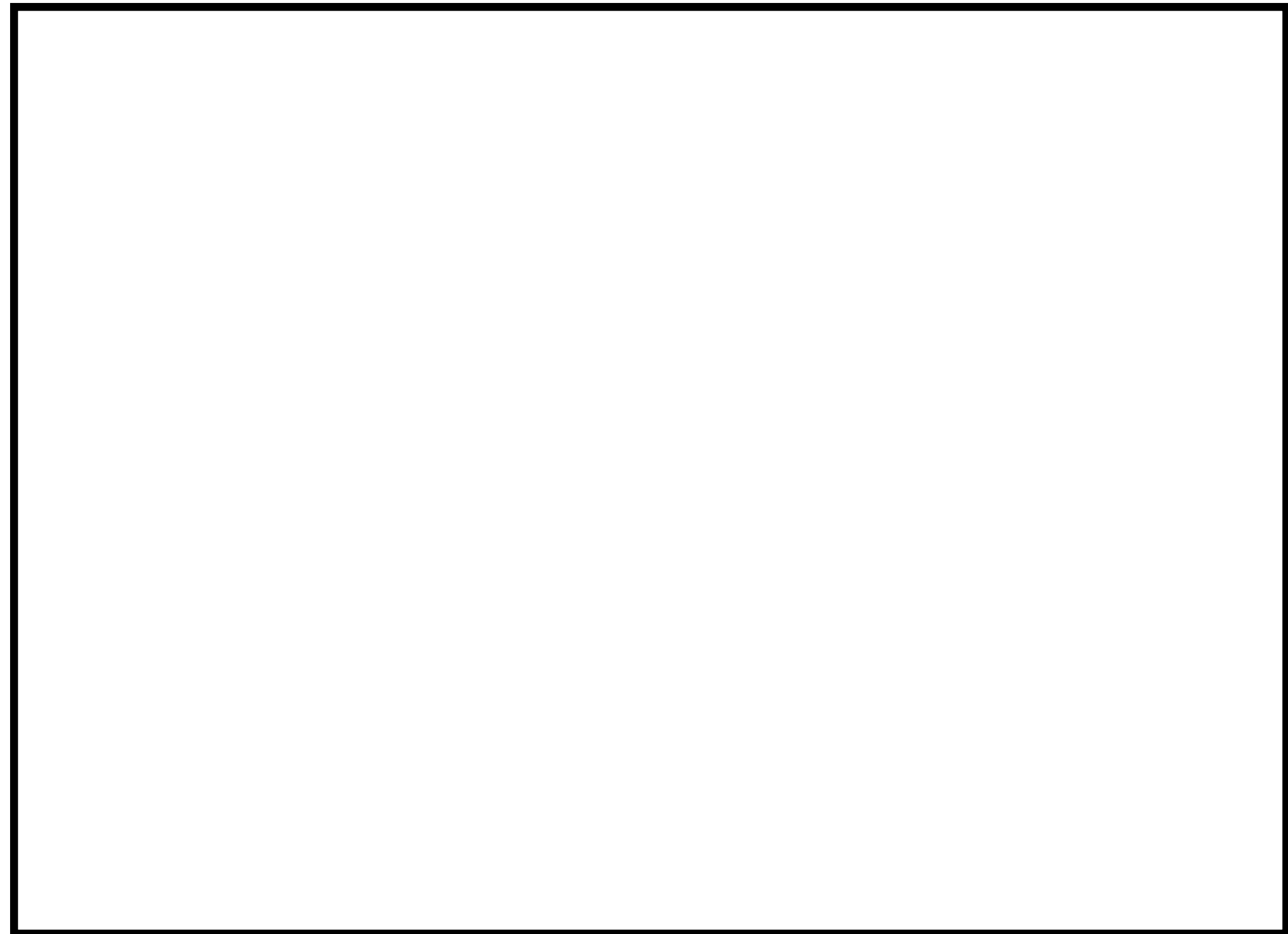
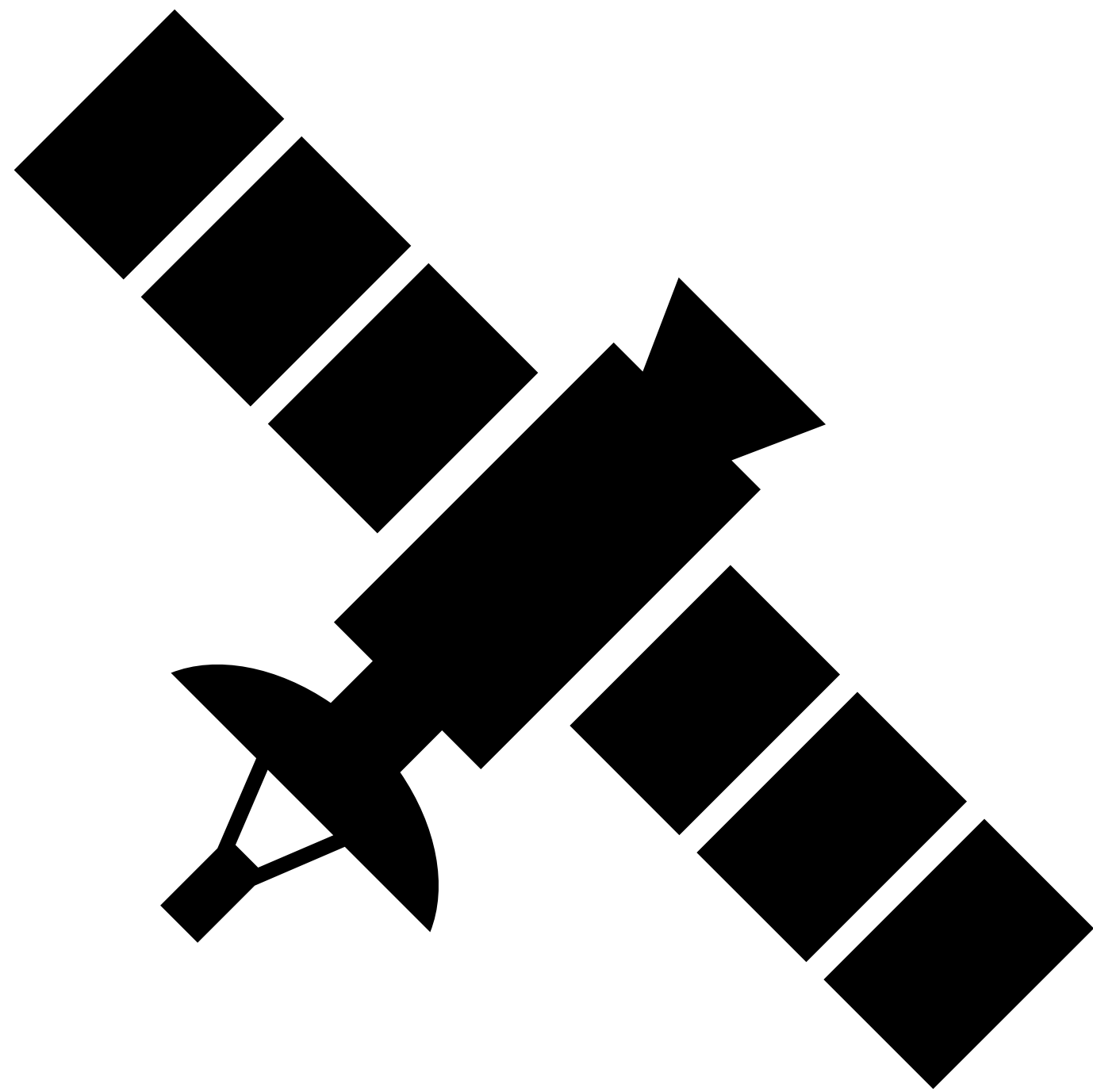


FAULT Controls Power Systems
FAULT Comms Scientific FAULT
...
Runtime Verification

“If a command is received, it shall be acknowledged either in the near window (within 5 cycles) or after a 2-cycle bus delay (from 2 to 7 cycles from now)”

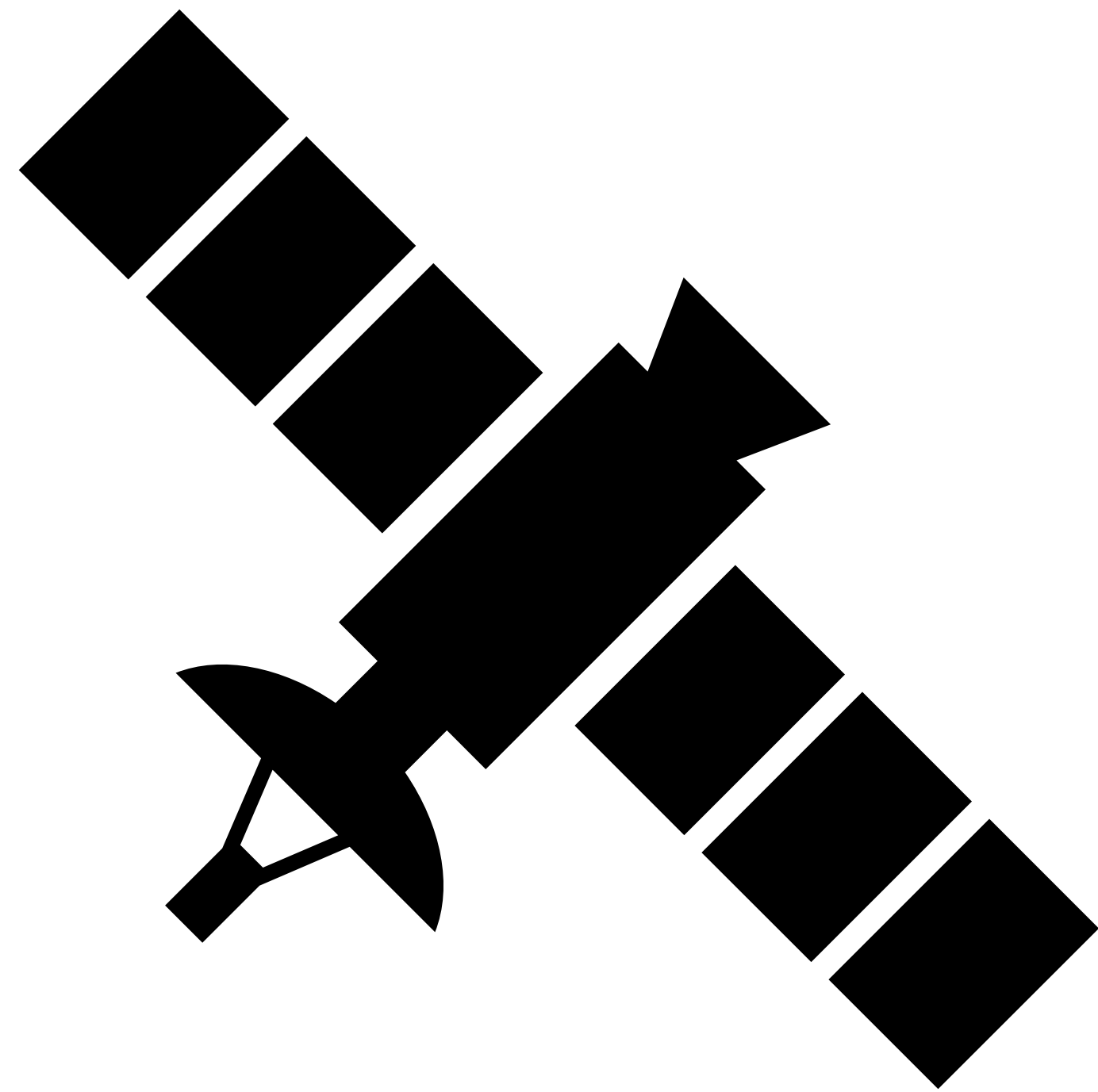
Real-time embedded systems have strict resource constraints, leaving little room for RV.

Computational Resources



Real-time embedded systems have strict resource constraints, leaving little room for RV.

Computational Resources

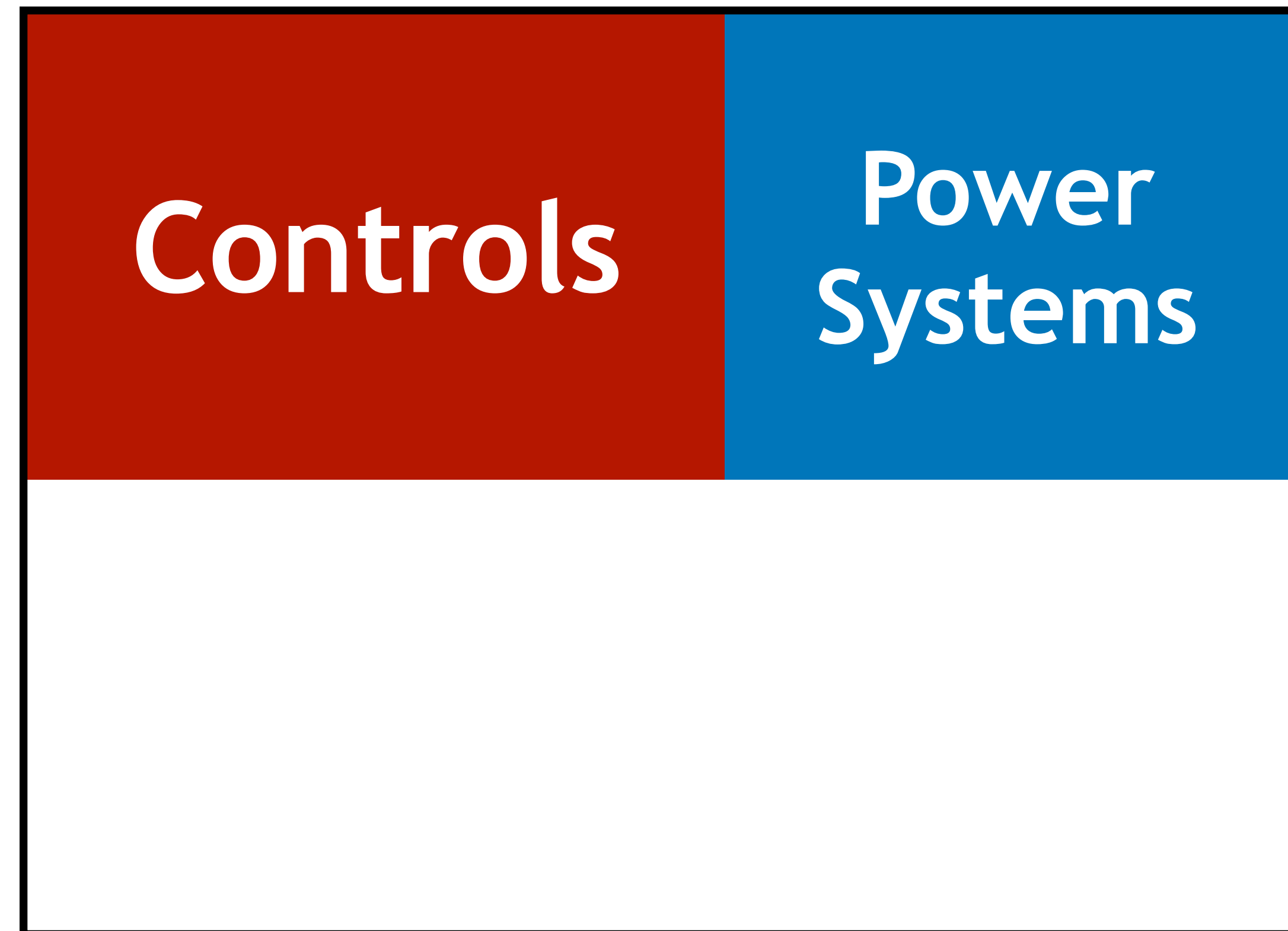
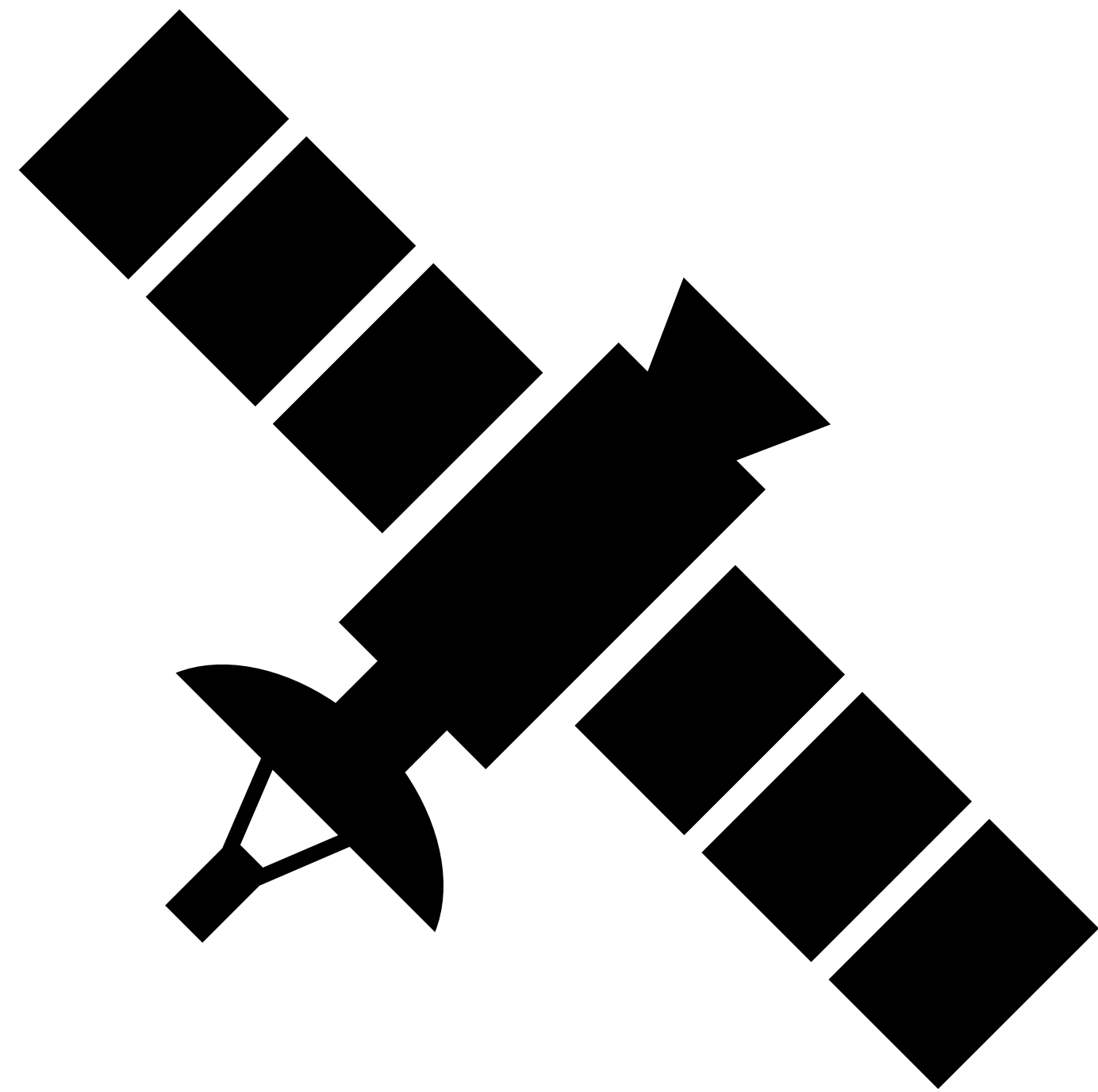


Controls

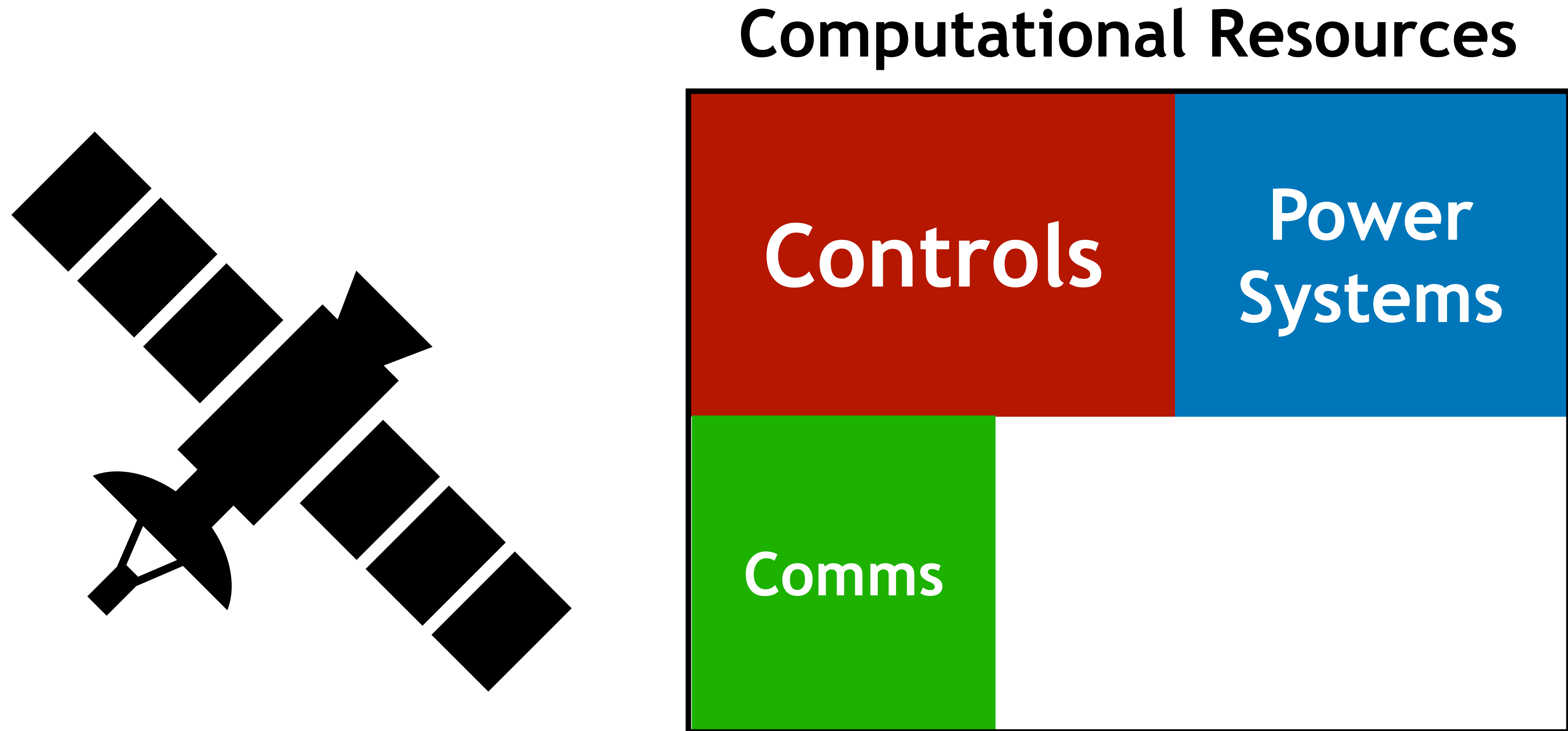
A diagram illustrating the allocation of computational resources. It consists of a large white rectangle with a black border. Inside the top-left corner of this rectangle is a smaller, solid red rectangle. The word "Controls" is written in white, bold, sans-serif font, centered within the red rectangle. The remaining space within the large white rectangle is empty, representing the limited resources available for other tasks like RV.

Real-time embedded systems have strict resource constraints, leaving little room for RV.

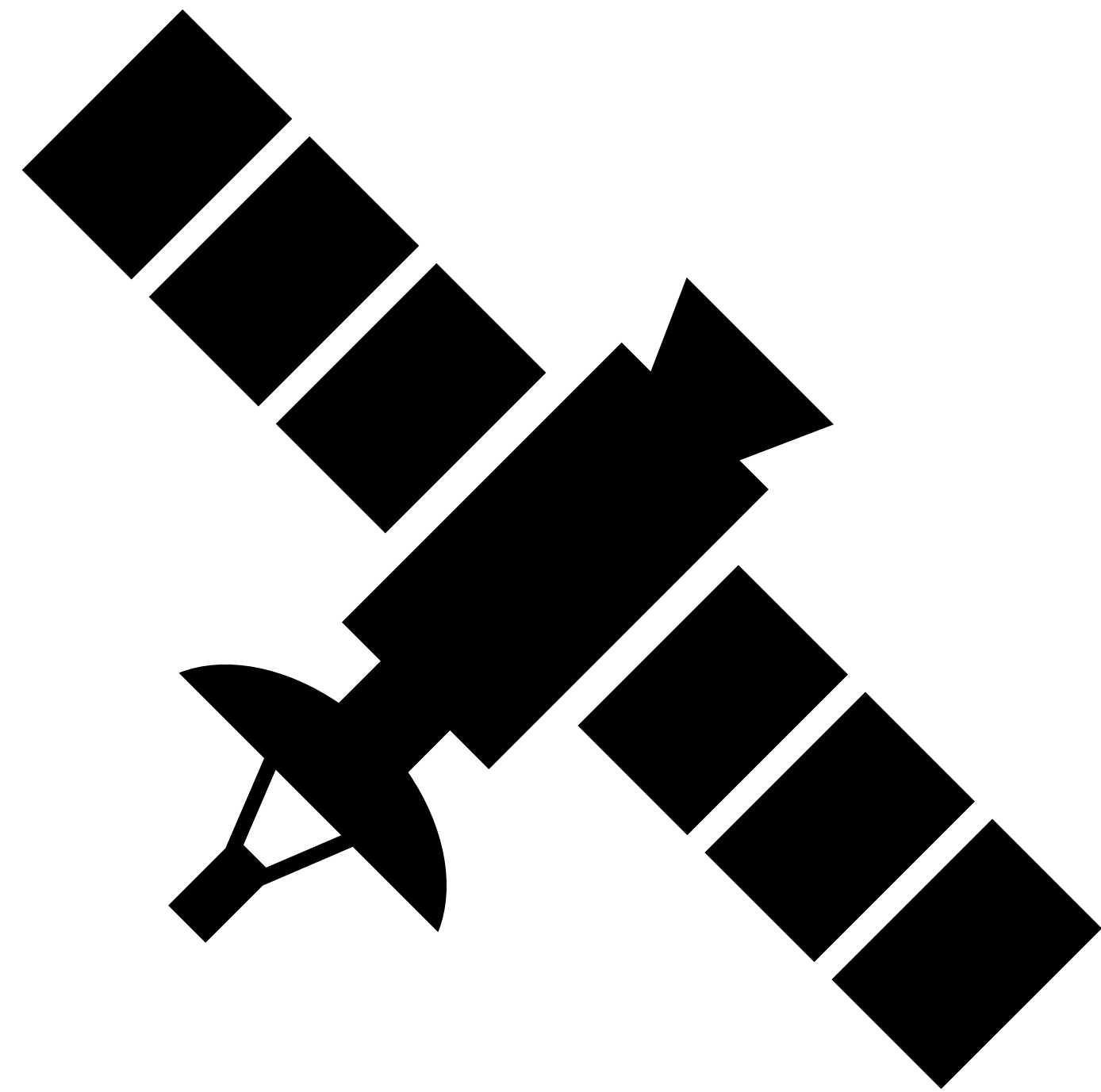
Computational Resources



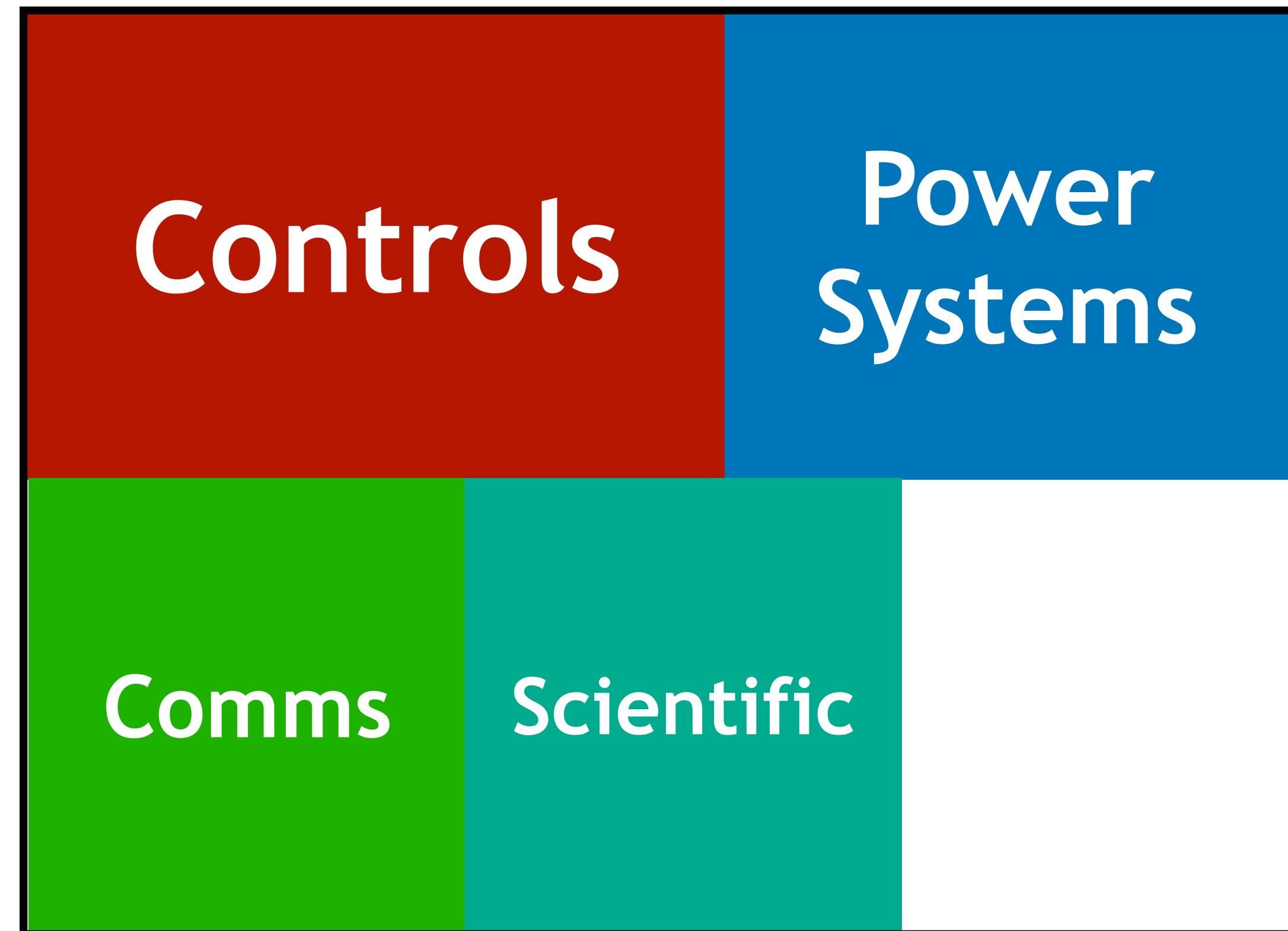
Real-time embedded systems have strict resource constraints, leaving little room for RV.



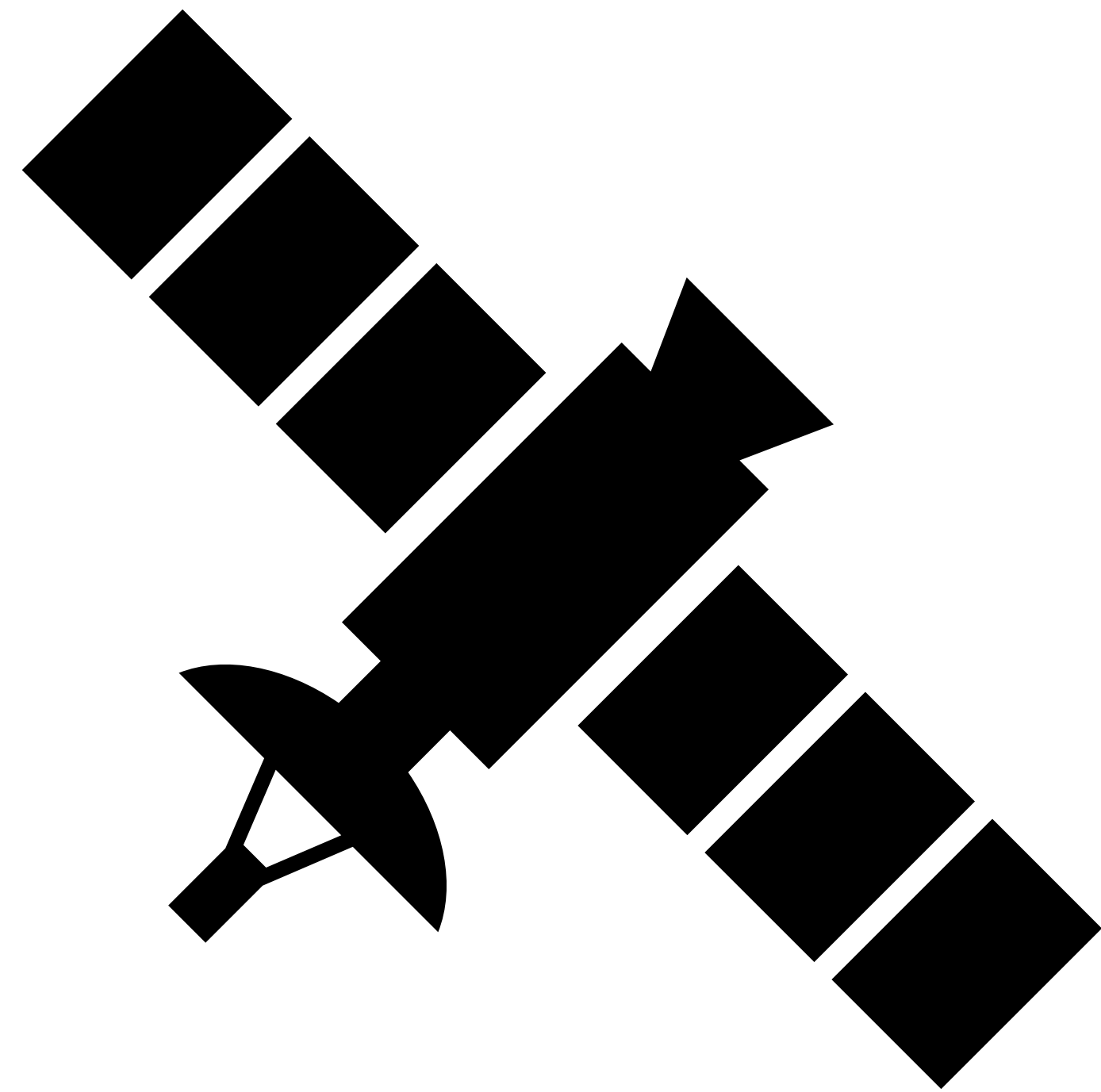
Real-time embedded systems have strict resource constraints, leaving little room for RV.



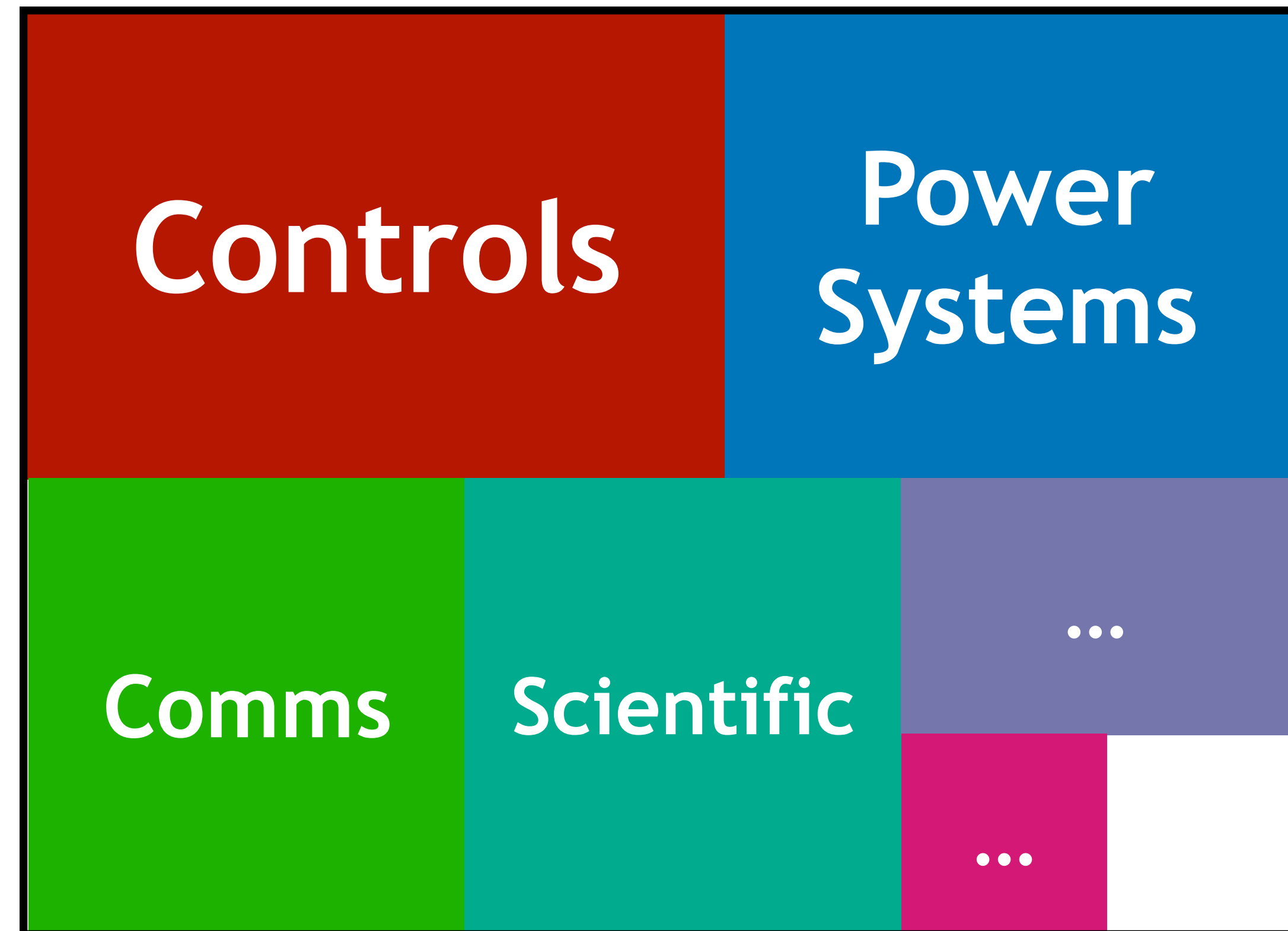
Computational Resources



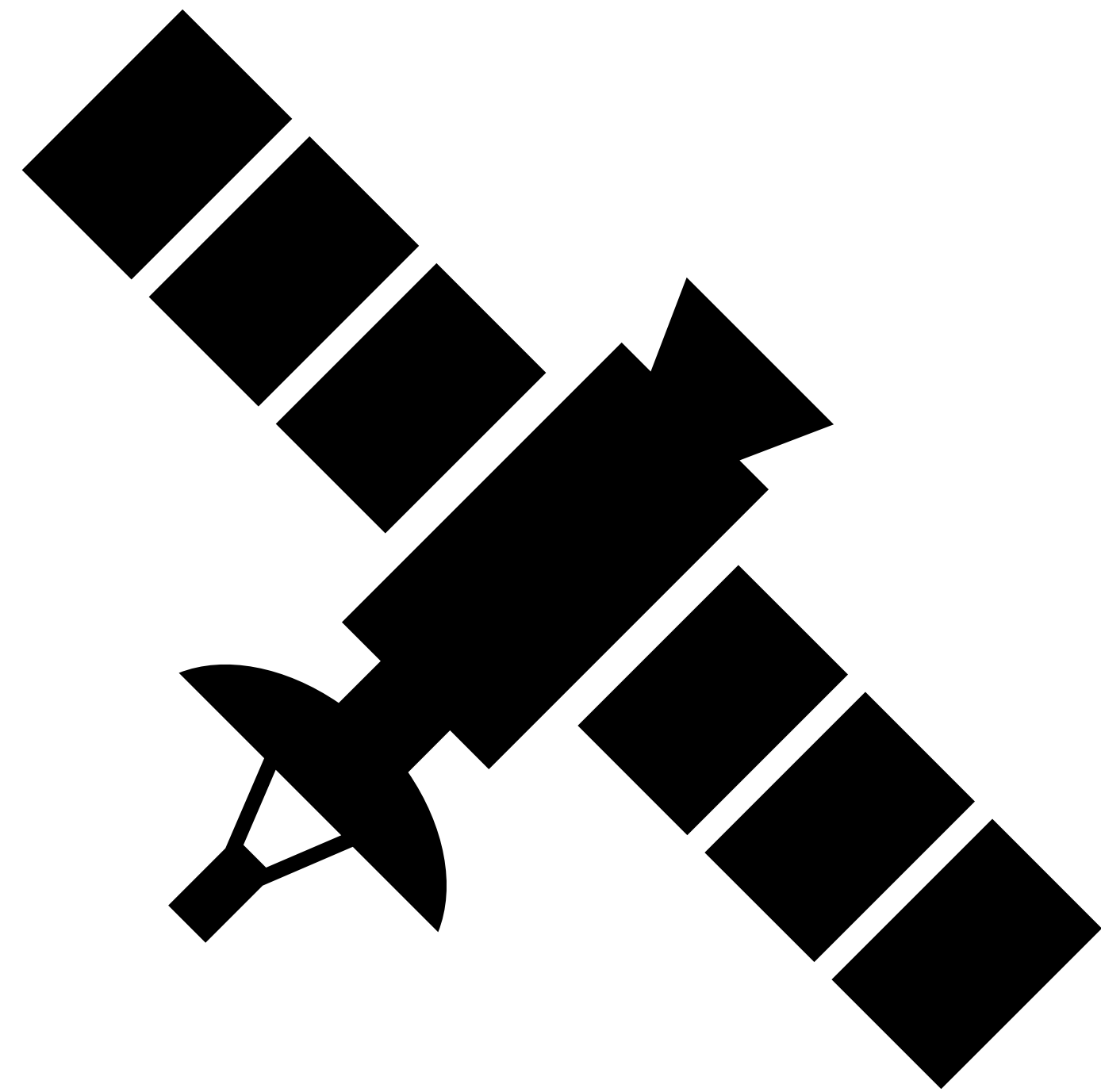
Real-time embedded systems have strict resource constraints, leaving little room for RV.



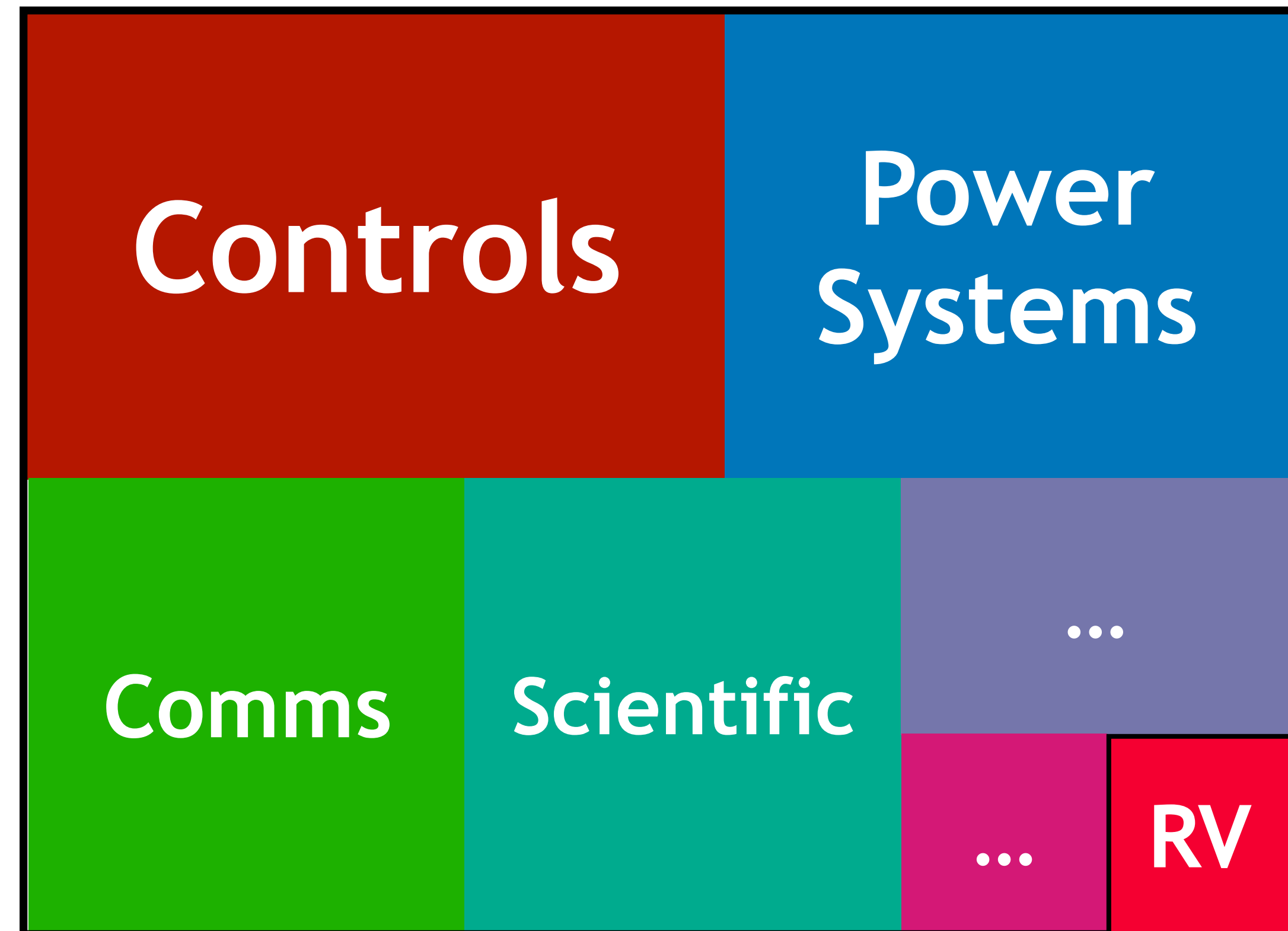
Computational Resources



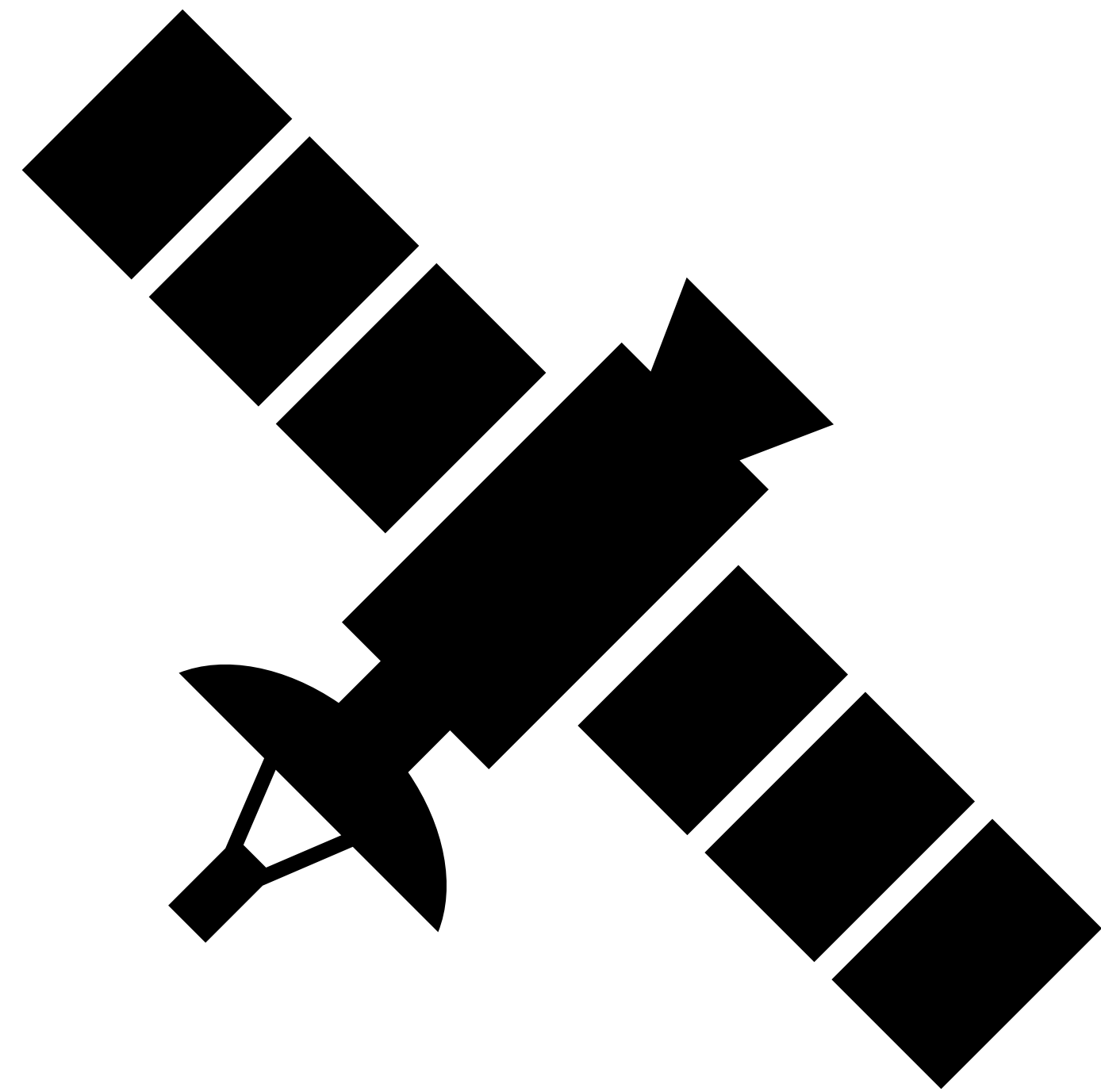
Real-time embedded systems have strict resource constraints, leaving little room for RV.



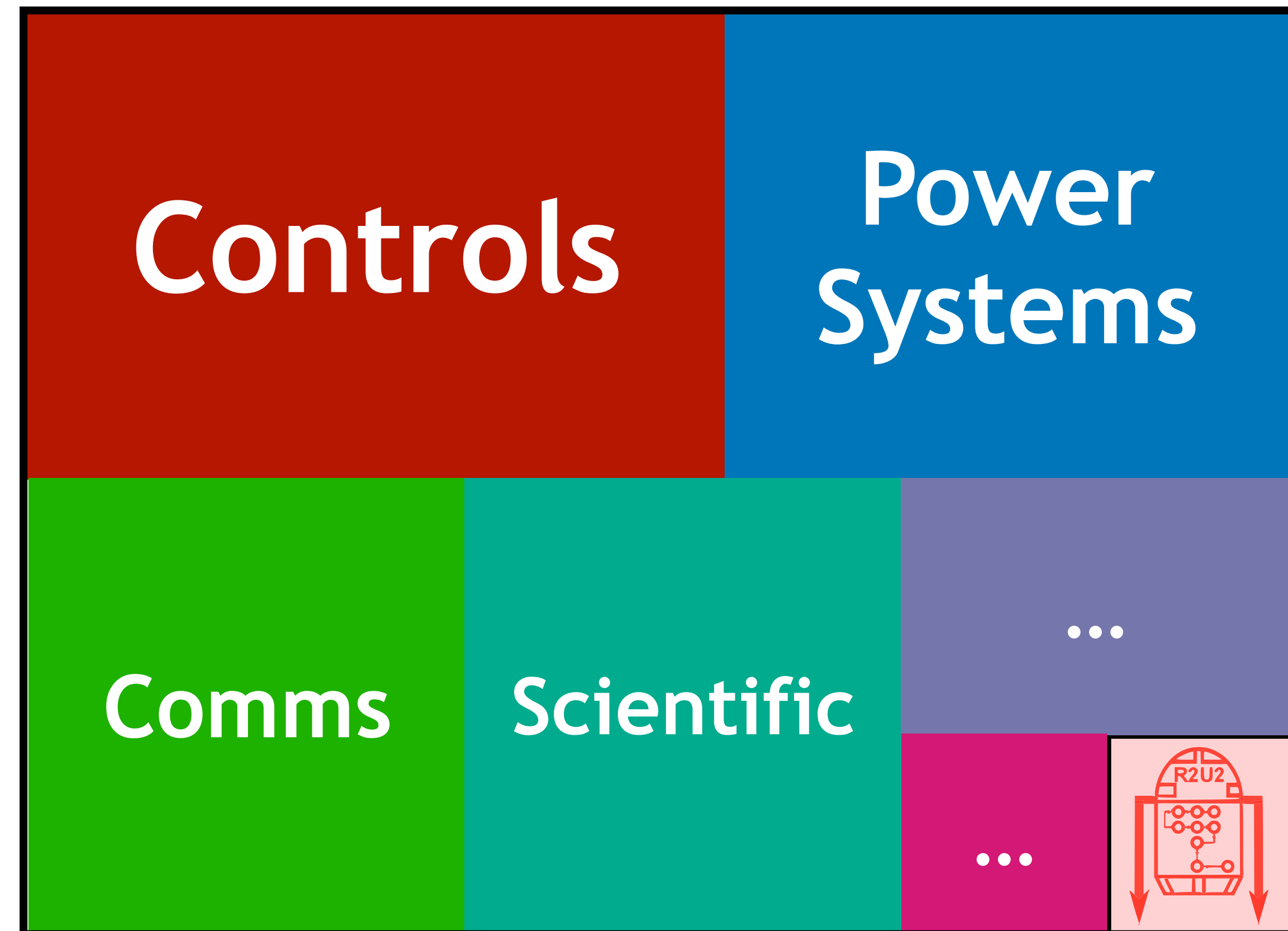
Computational Resources



Real-time embedded systems have strict resource constraints, leaving little room for RV.



Computational Resources



Real-time embedded systems have strict resource constraints leaving little room for RV.

Reinbacher, Thomas, Kristin Yvonne Rozier, and Johann Schumann. "**Temporal-logic based runtime observer pairs for system health management of real-time systems.**" *International Conference on Tools and Algorithms for the Construction and Analysis of Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, **2014**.

Johannsen, Chris, et al. "**R2u2 version 3.0: Re-imagining a toolchain for specification, resource estimation, and optimized observer generation for runtime verification in hardware and software.**" *International Conference on Computer Aided Verification*. Cham: Springer Nature Switzerland, **2023**.

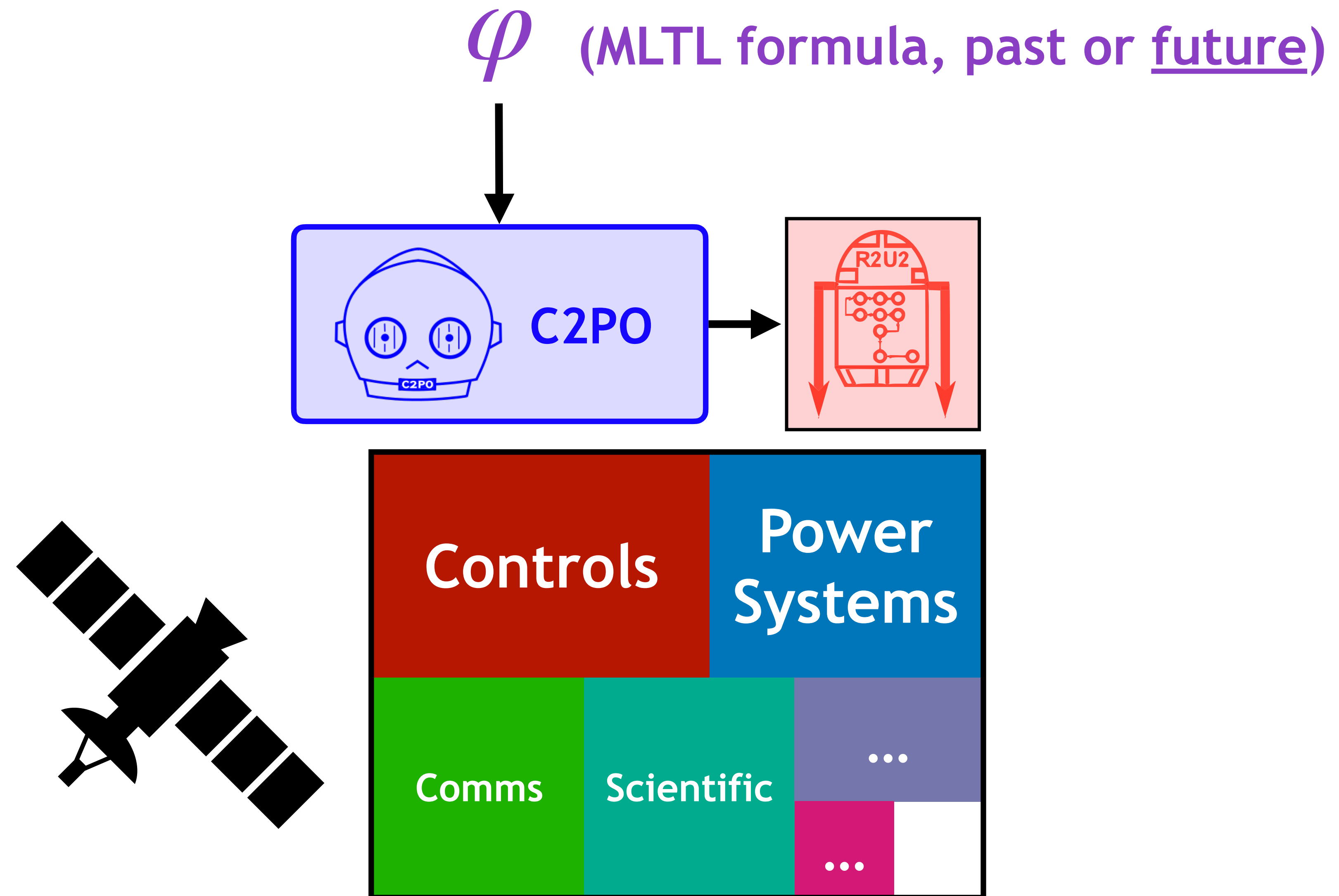
Aurandt, Alexis, Phillip H. Jones, and Kristin Yvonne Rozier. "**Towards a safe, verified runtime monitor for embedded systems: R2U2 in embedded rust.**" *NASA Formal Methods Symposium*. Cham: Springer Nature Switzerland, **2025**.

And more!

<https://github.com/R2U2/r2u2>



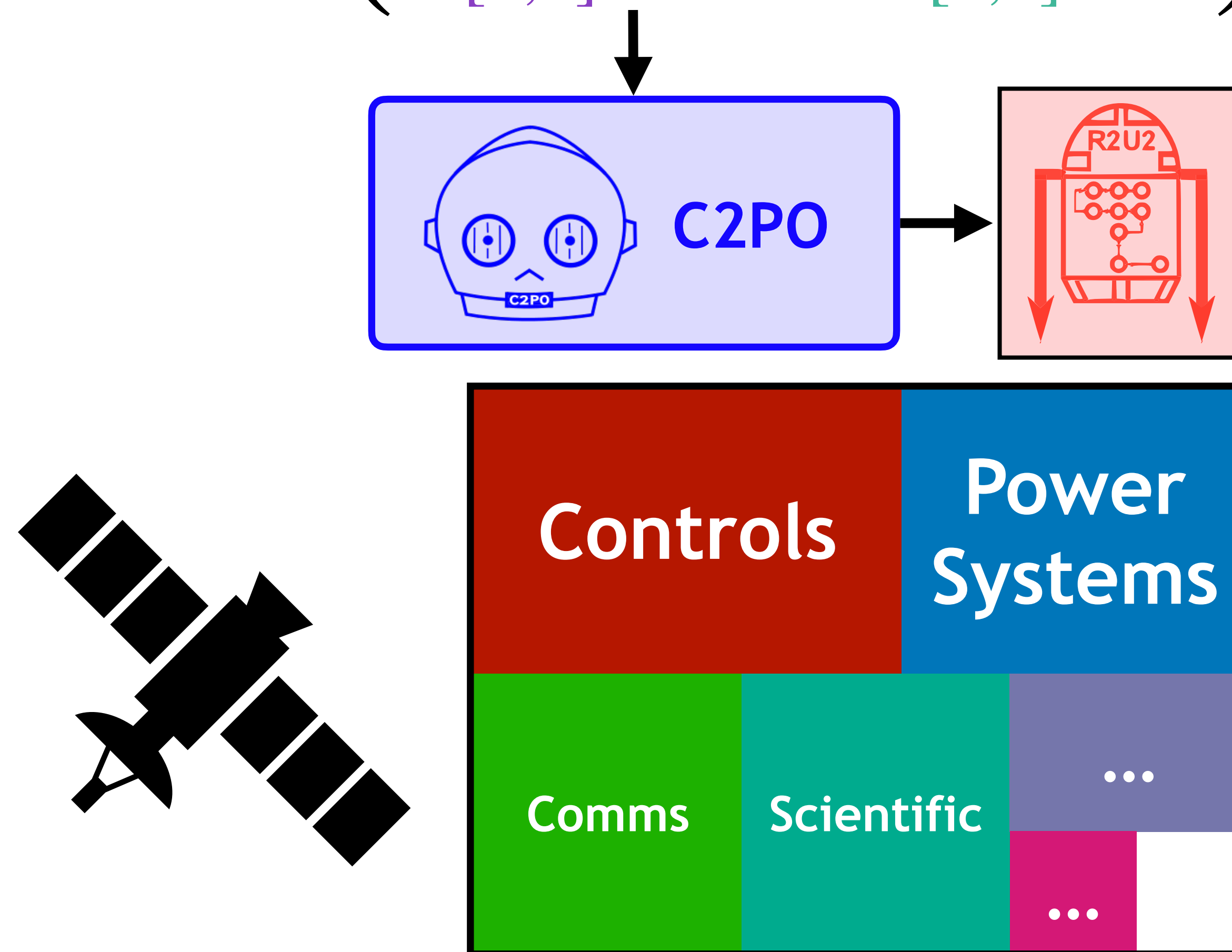
C2PO encodes Mission-time LTL formulas into R2U2 monitors.



C2PO encodes Mission-time LTL formulas into R2U2 monitors.

“If a command is **received**, it shall be **acknowledged** either in the near window (**within 5 cycles**) or after a 2-cycle bus delay (**from 2 to 7 cycles from now**)”

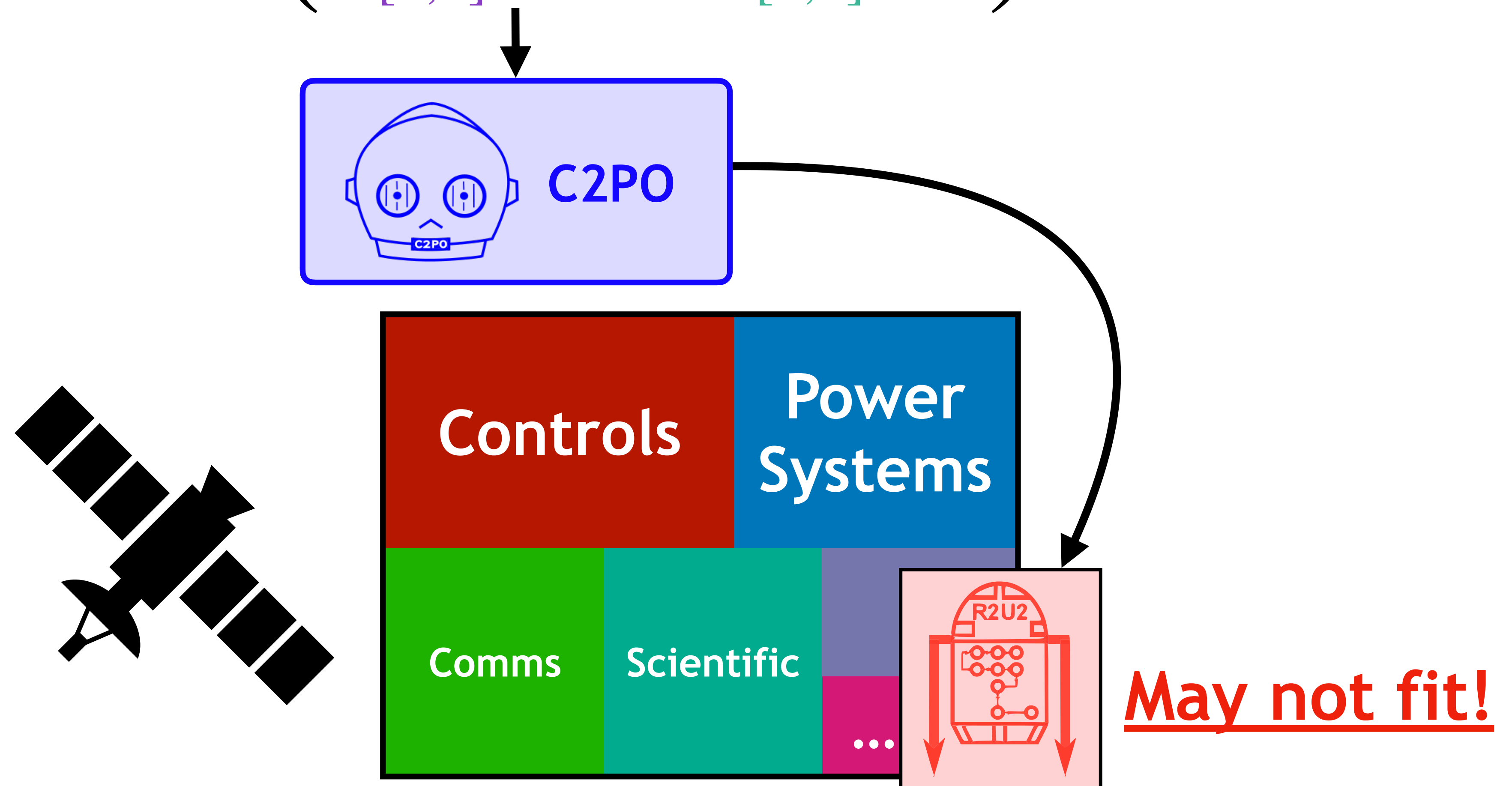
$$\text{RCVD} \rightarrow \left(\diamond_{[0,5]} \text{ACK} \vee \diamond_{[2,7]} \text{ACK} \right)$$



C2PO encodes Mission-time LTL formulas into R2U2 monitors.

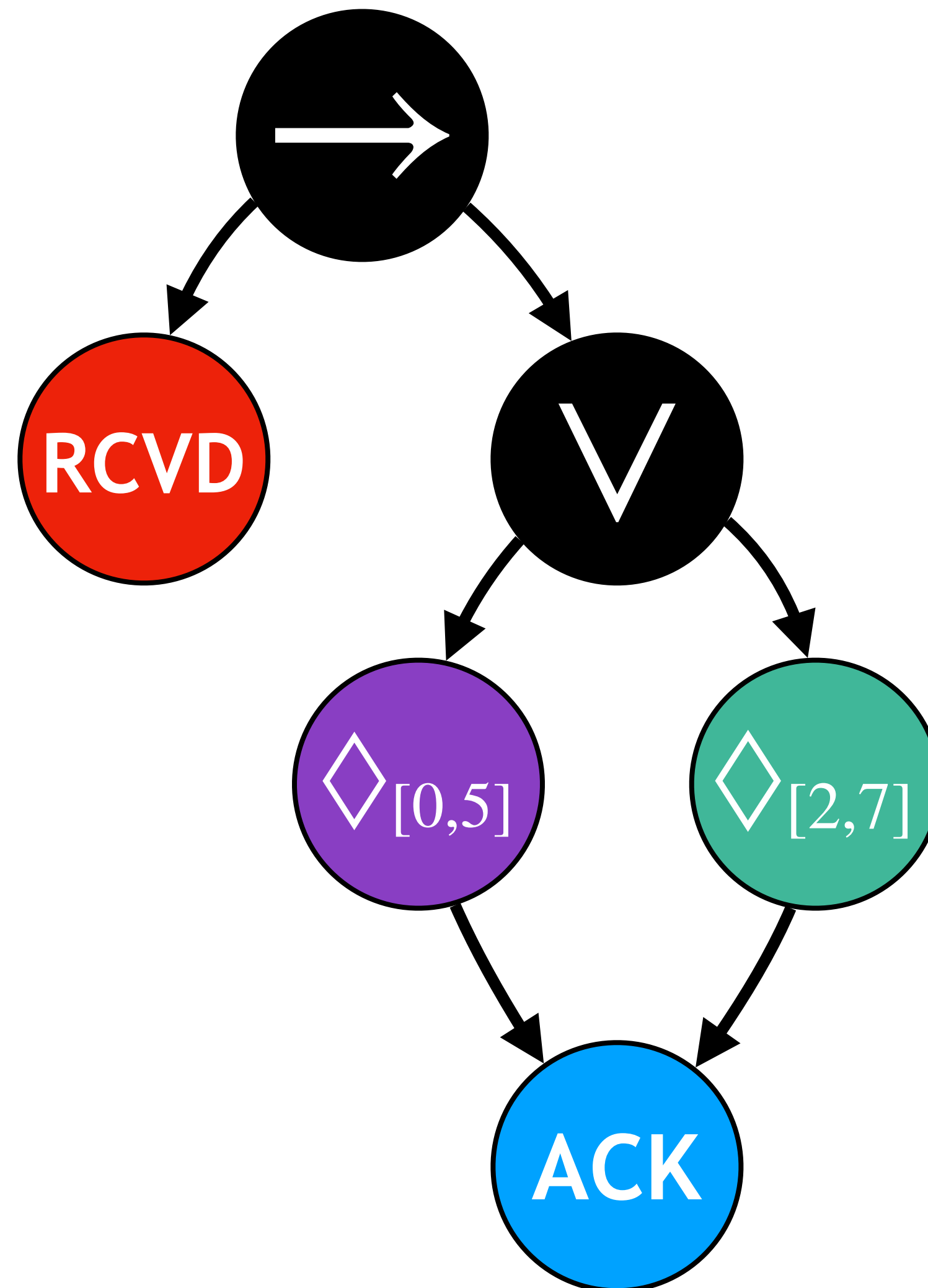
“If a command is **received**, it shall be **acknowledged** either in the near window (**within 5 cycles**) or after a 2-cycle bus delay (**from 2 to 7 cycles from now**)”

$$\text{RCVD} \rightarrow \left(\diamond_{[0,5]} \text{ACK} \vee \diamond_{[2,7]} \text{ACK} \right)$$



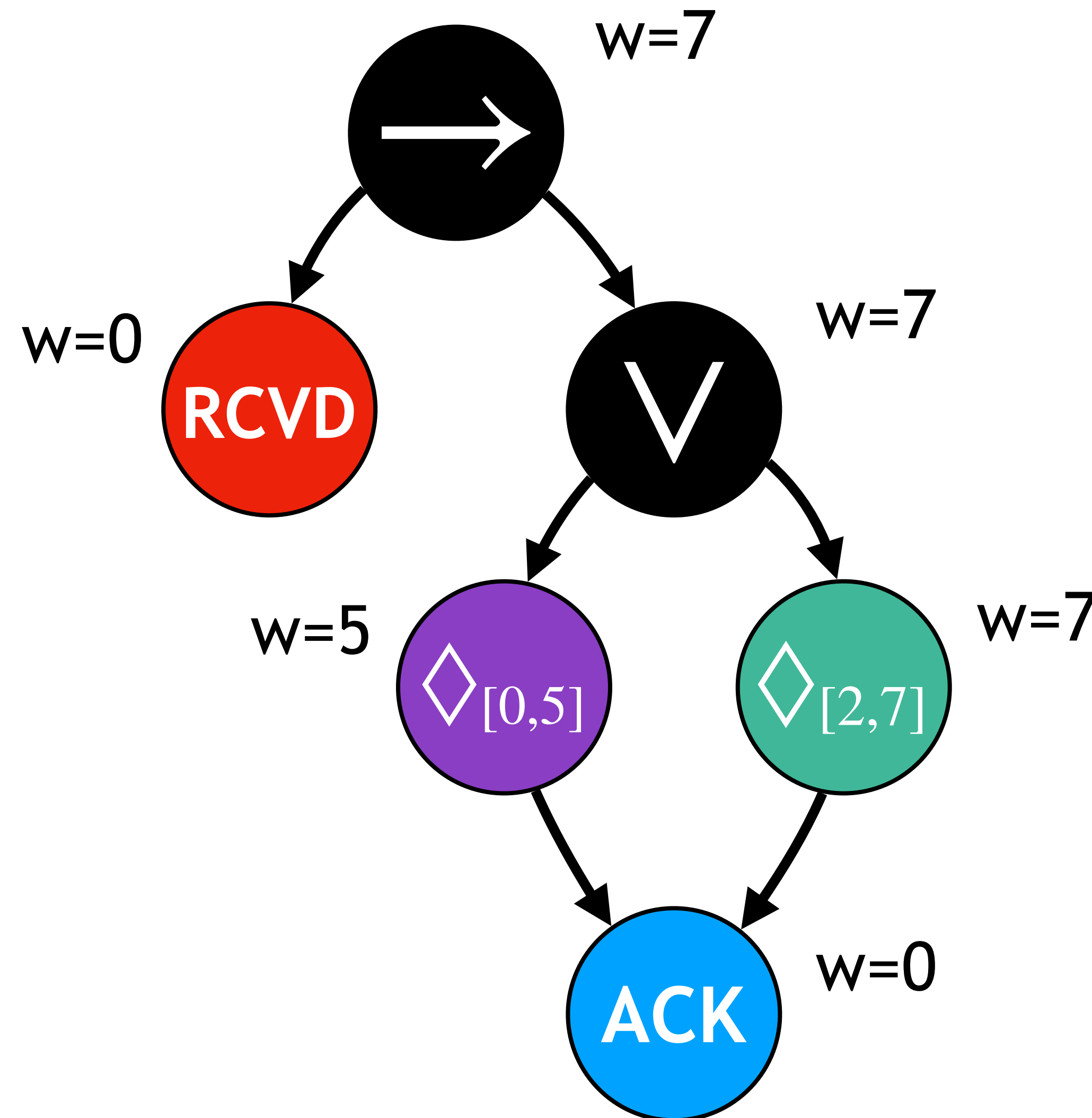
R2U2 monitors require memory for each node in a formula's DAG

Memory per node $\simeq$ (worst-case delay (w) of sibling — best-case delay (b)) + 1
(in # of machine integers)



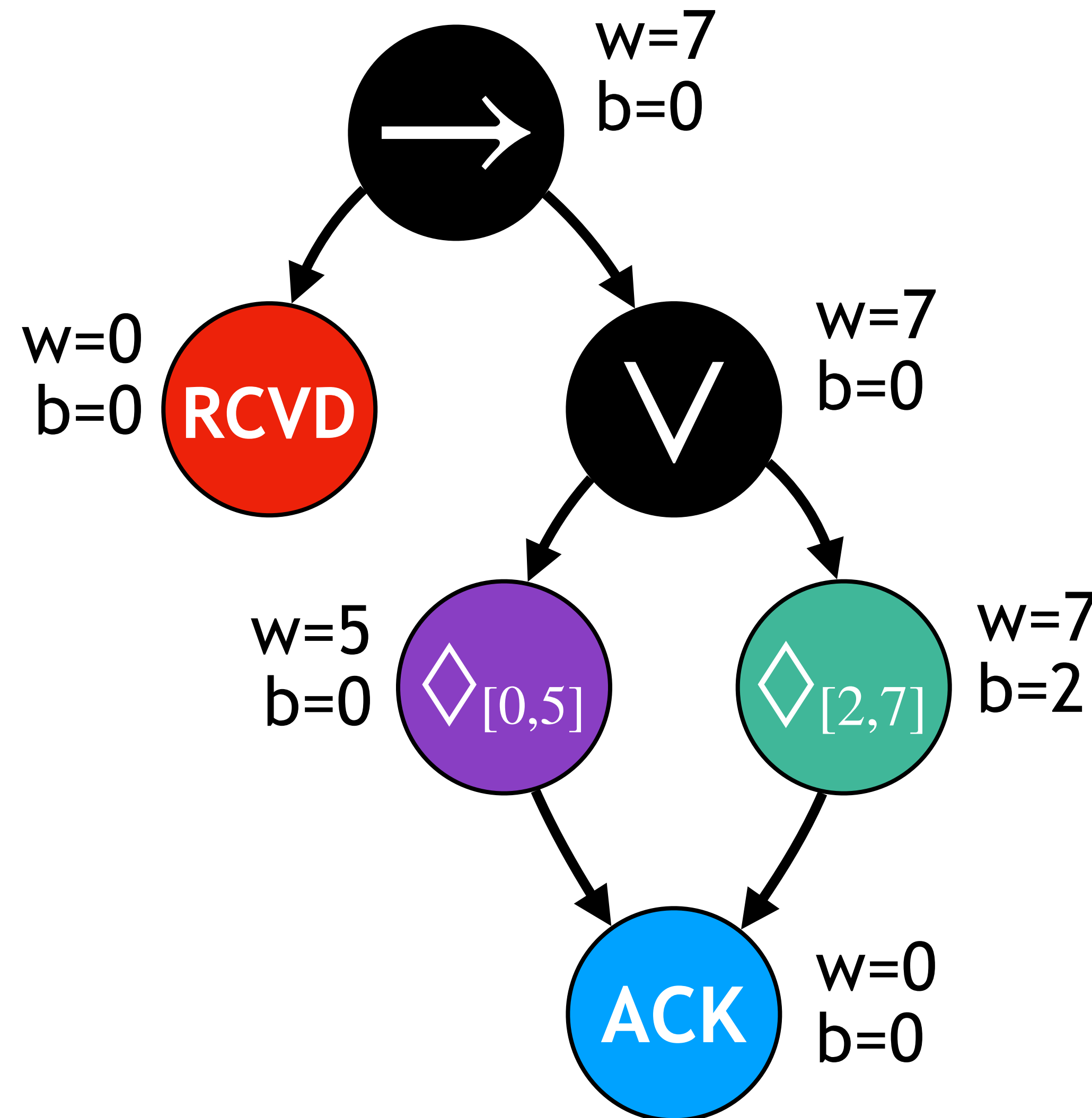
R2U2 monitors require memory for each node in a formula's DAG

Memory per node $\simeq$ (worst-case delay (w) of sibling — best-case delay (b)) + 1
(in # of machine integers)



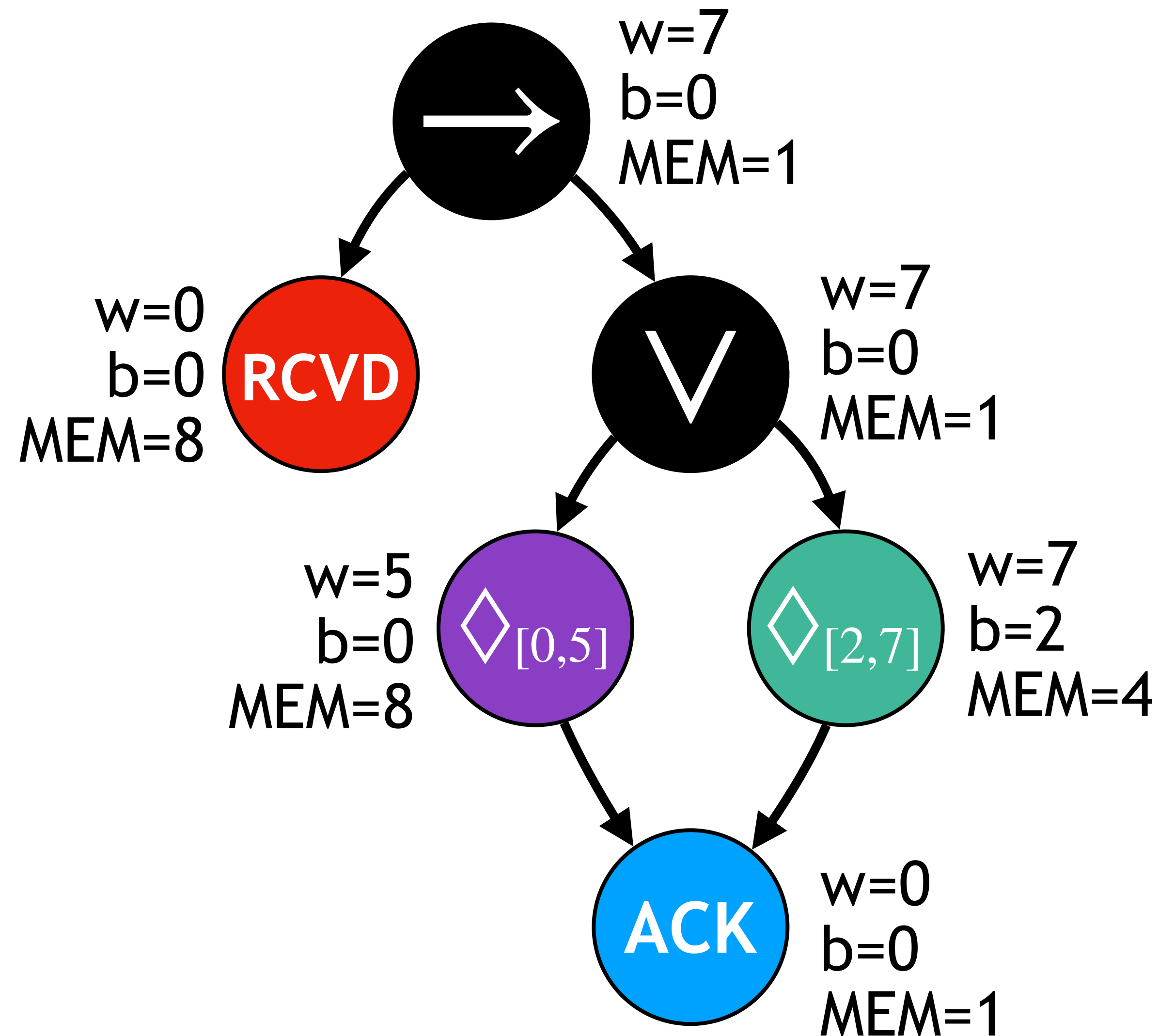
R2U2 monitors require memory for each node in a formula's DAG

Memory per node $\simeq$ (worst-case delay (w) of sibling — best-case delay (b)) + 1
(in # of machine integers)

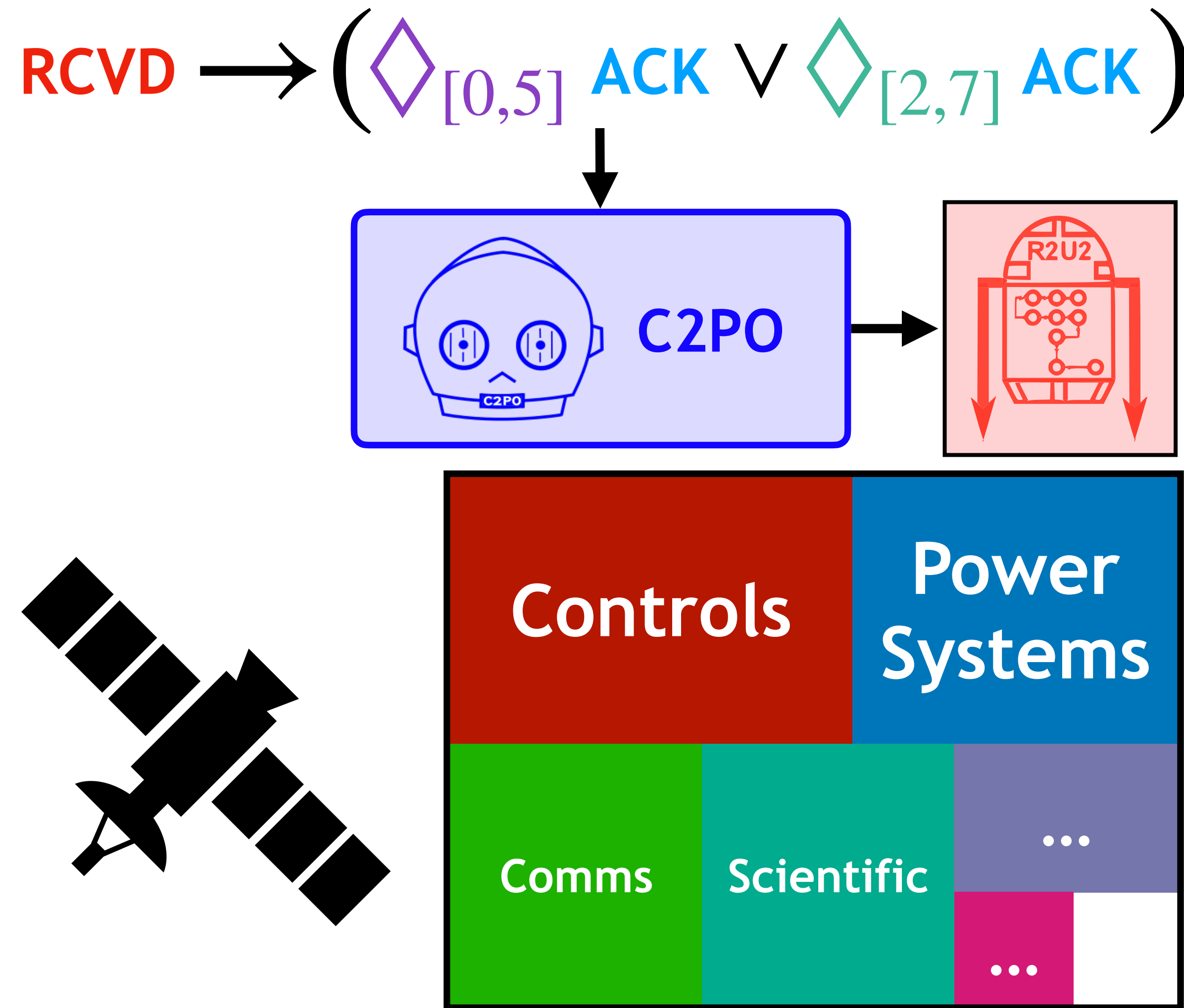


R2U2 monitors require memory for each node in a formula's DAG

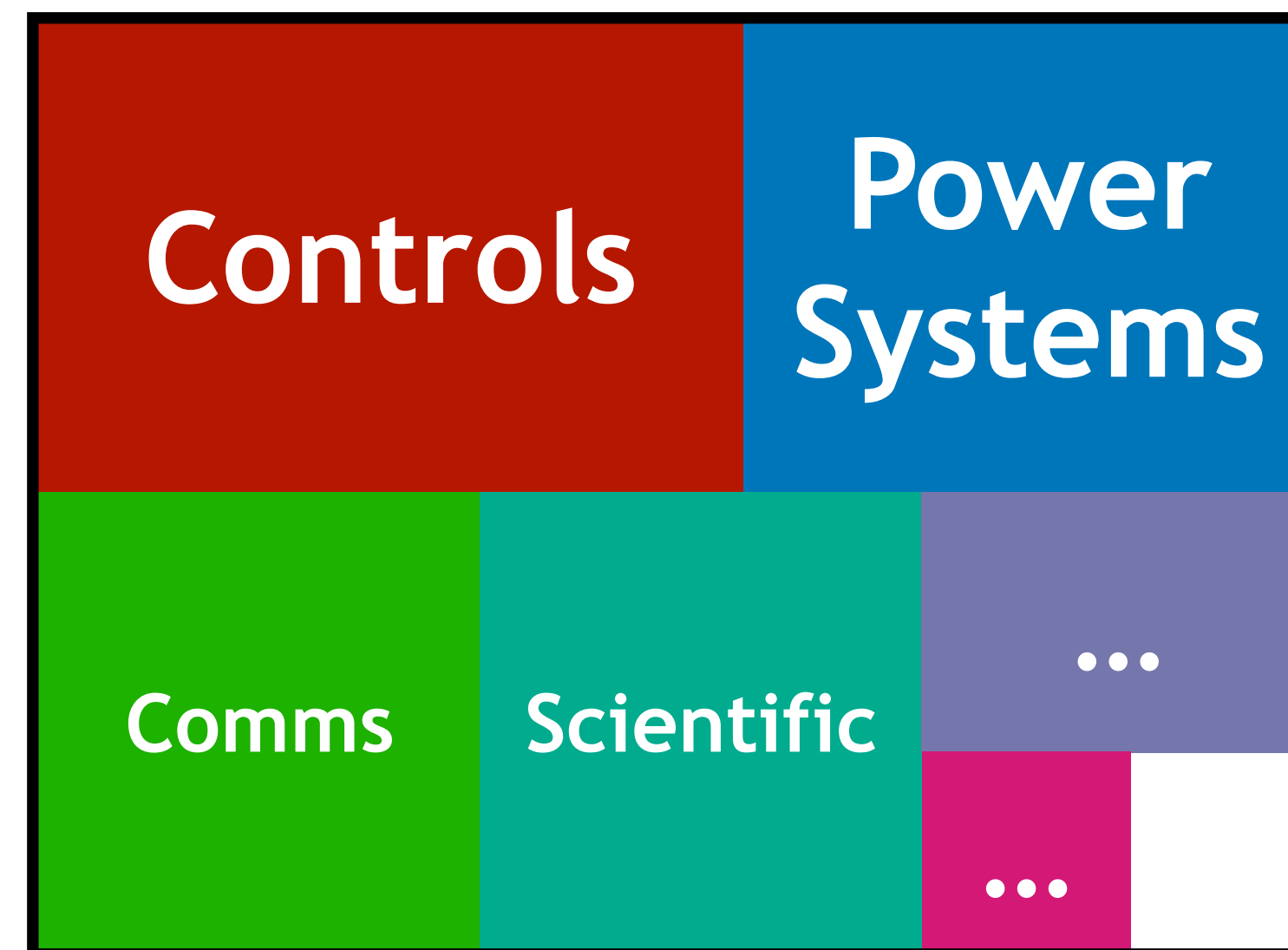
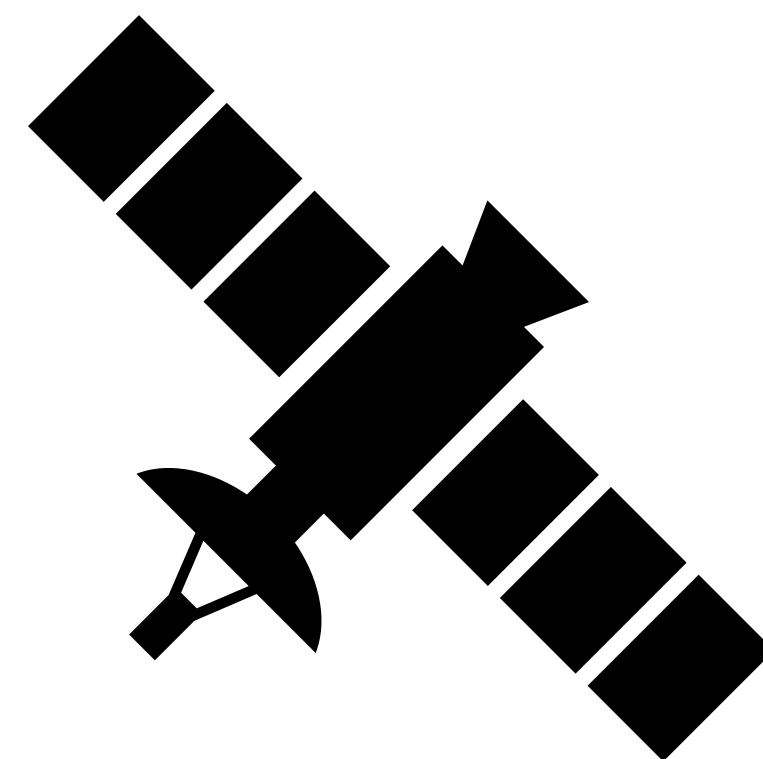
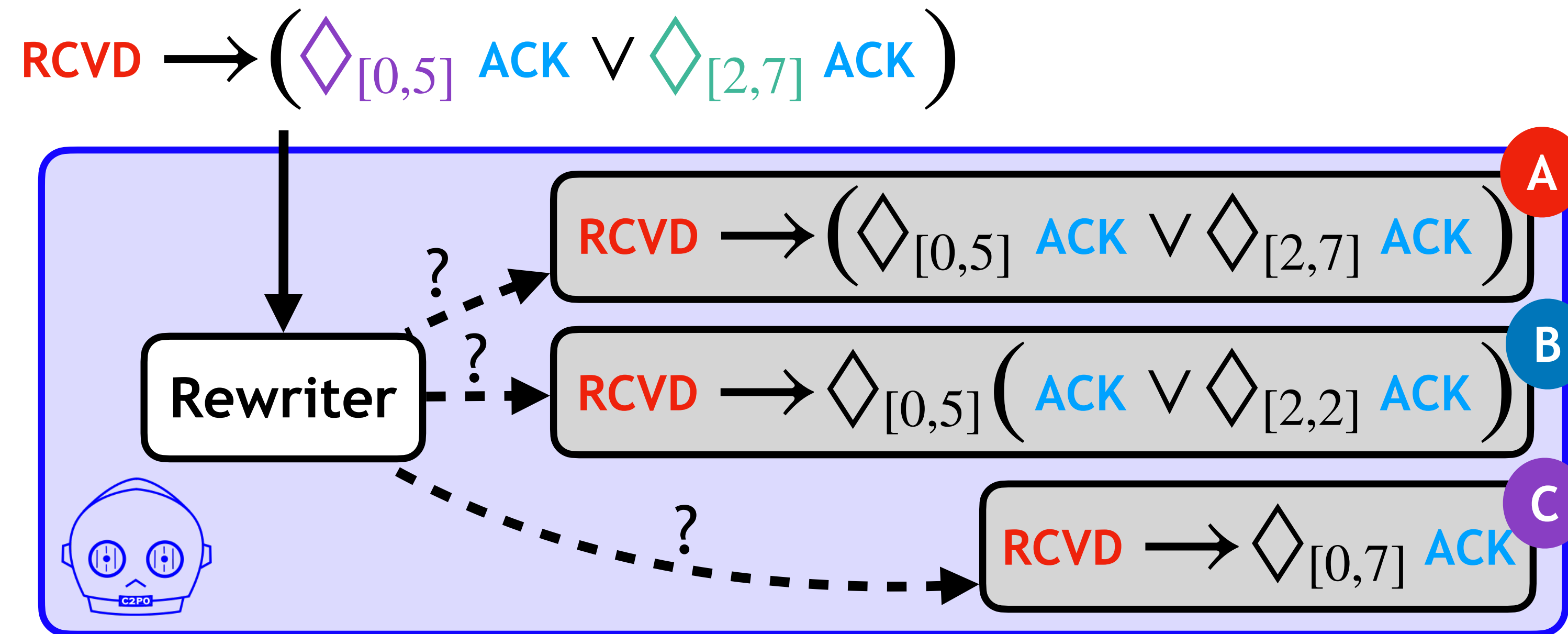
Memory per node $\simeq$ (worst-case delay (w) of sibling — best-case delay (b)) + 1
(in # of machine integers)



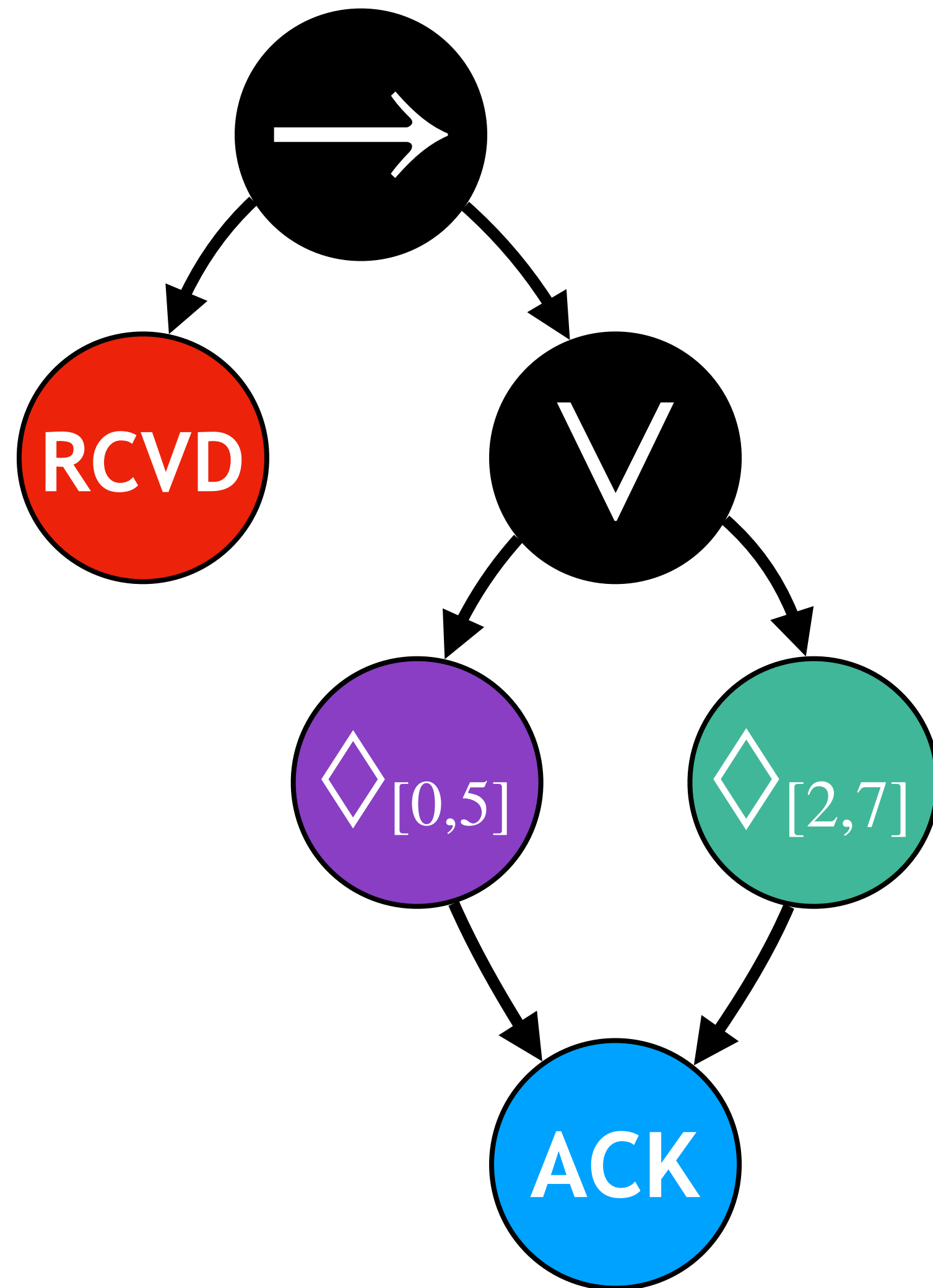
Rewriting can produce smaller equivalent monitors, but it's difficult to choose which rewrites to apply in general.



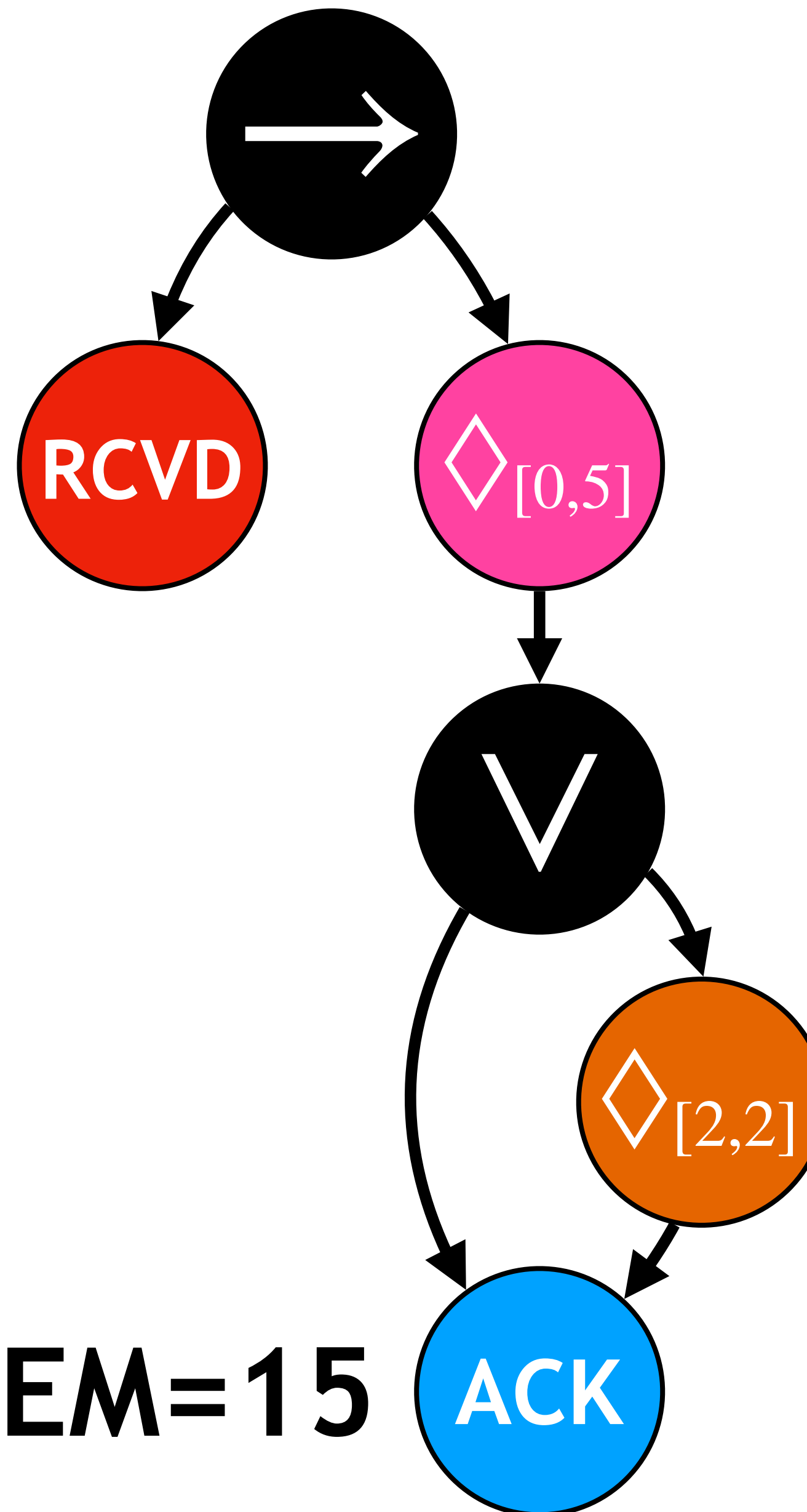
Rewriting can produce smaller equivalent monitors, but it's difficult to choose which rewrites to apply in general.



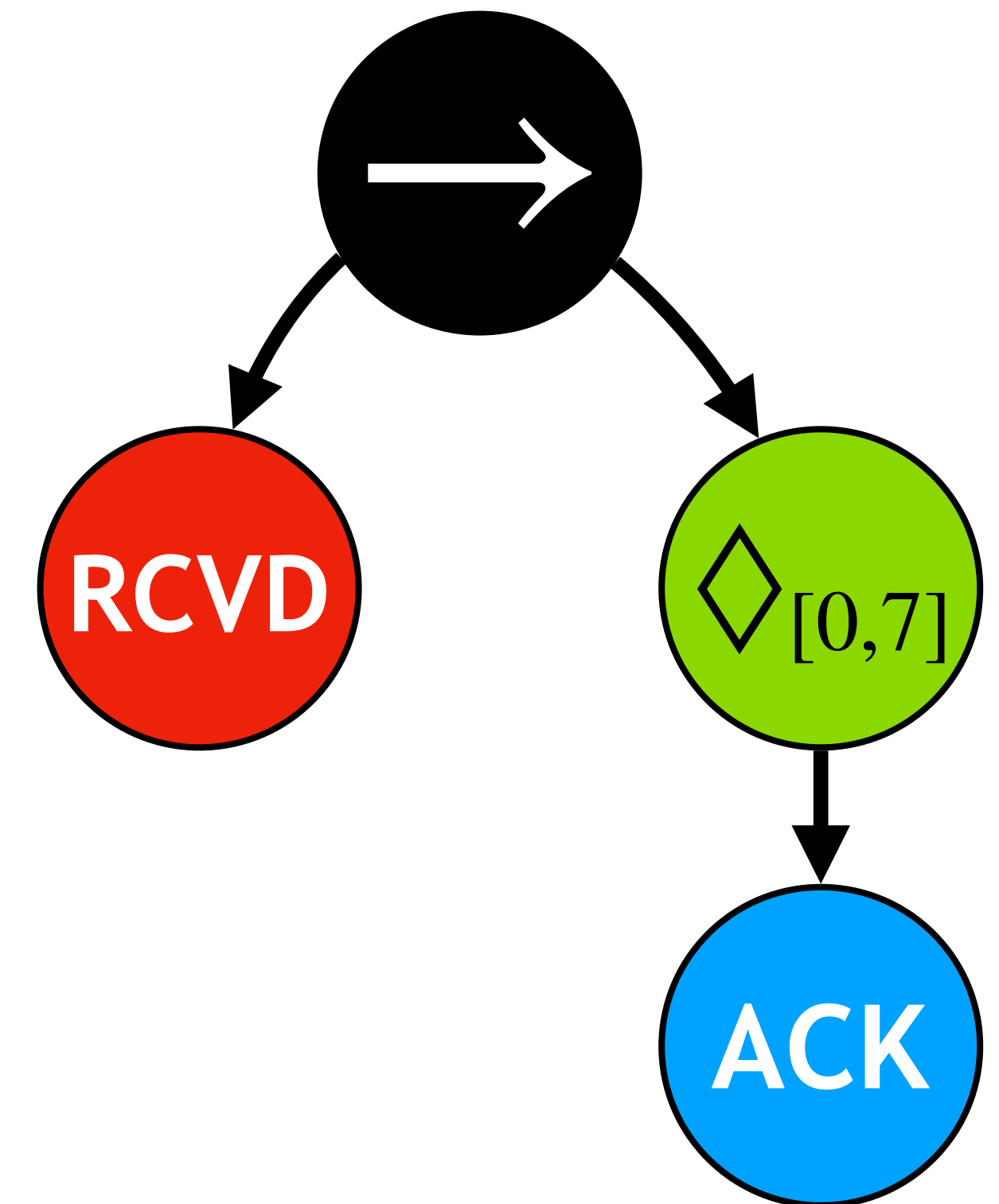
Equivalent formulas can have different R2U2 memory requirements.



A MEM=23

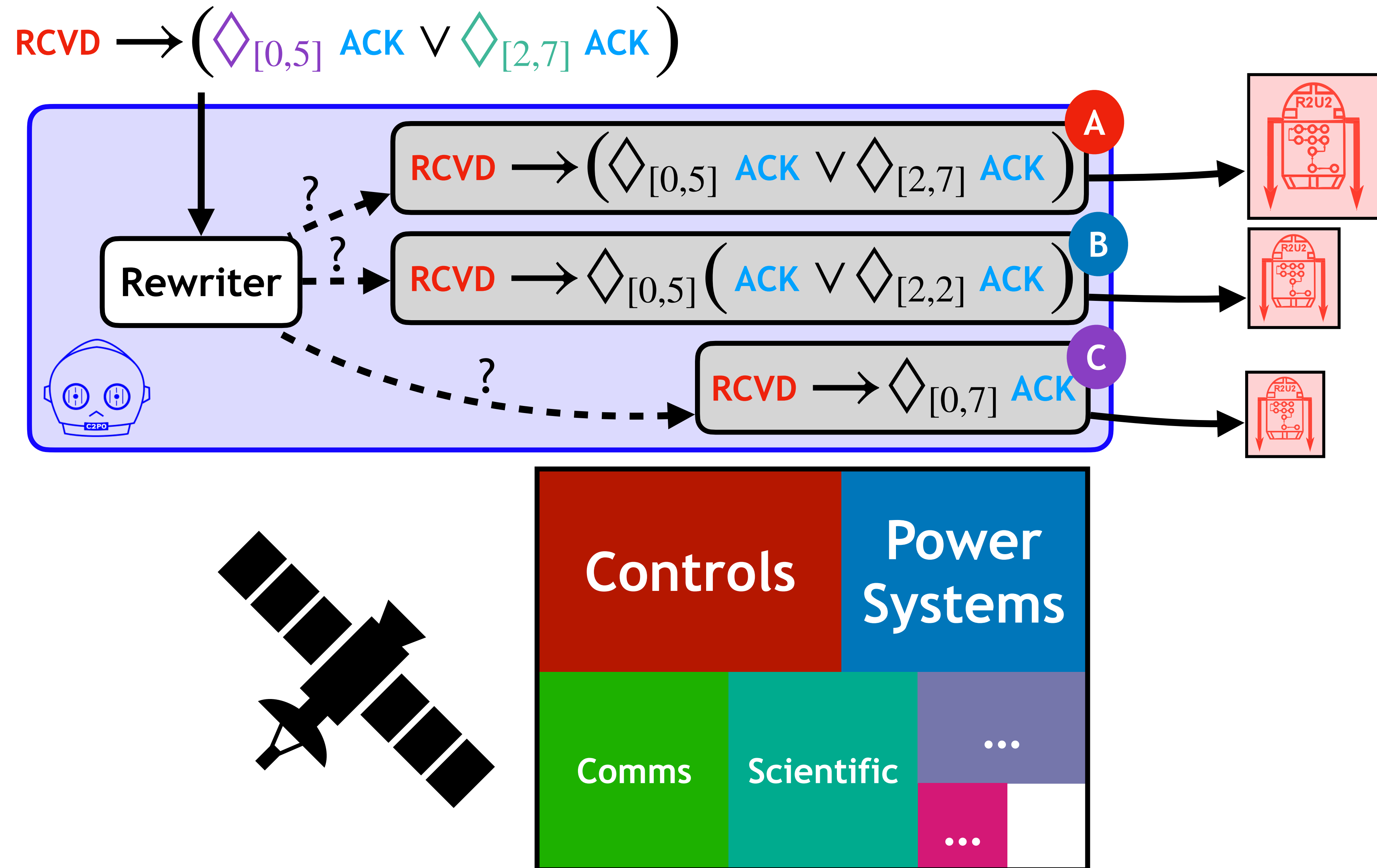


B MEM=15

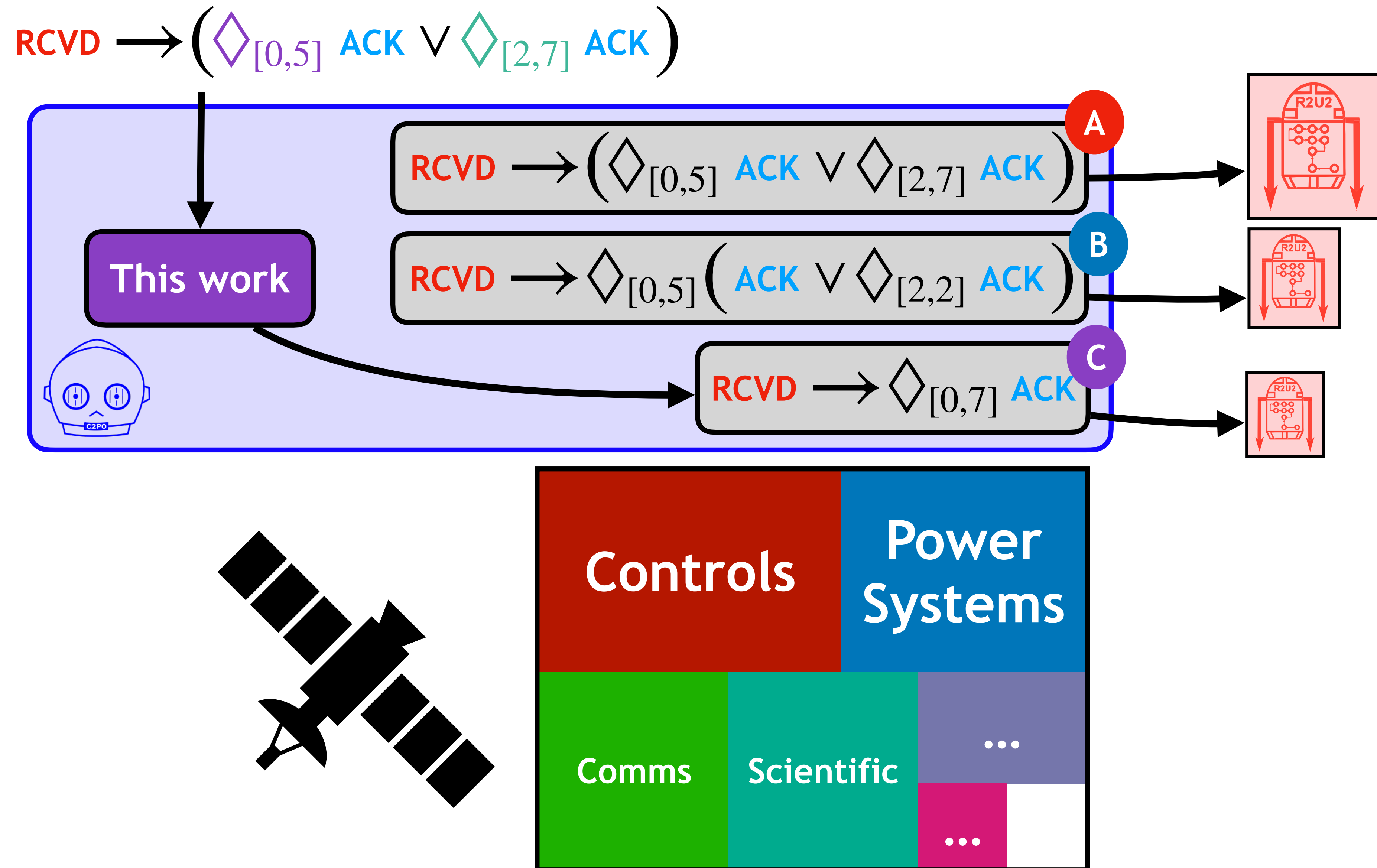


C MEM=11

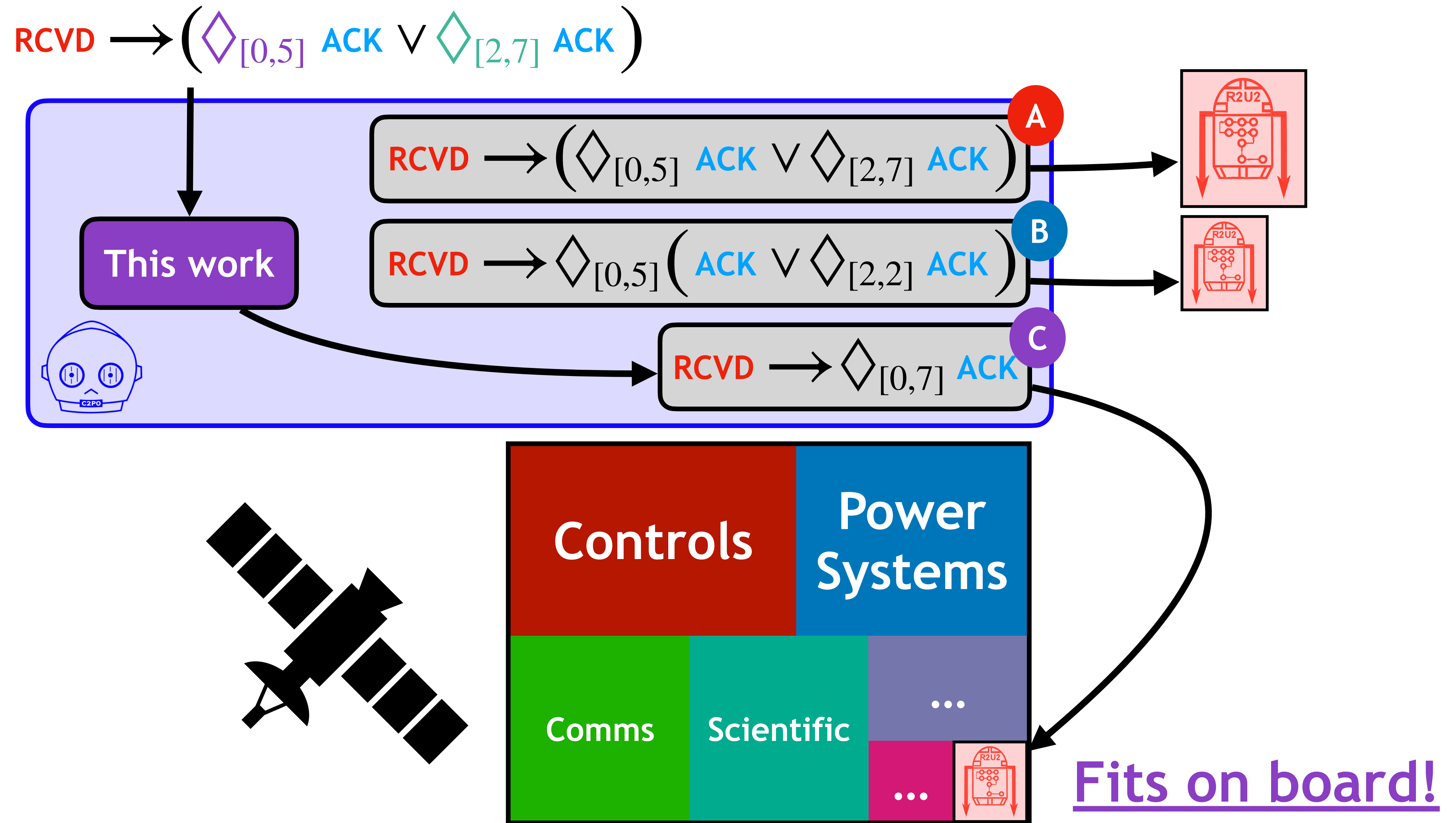
Our work adapts equality saturation to automatically find smaller equivalent R2U2 monitors.



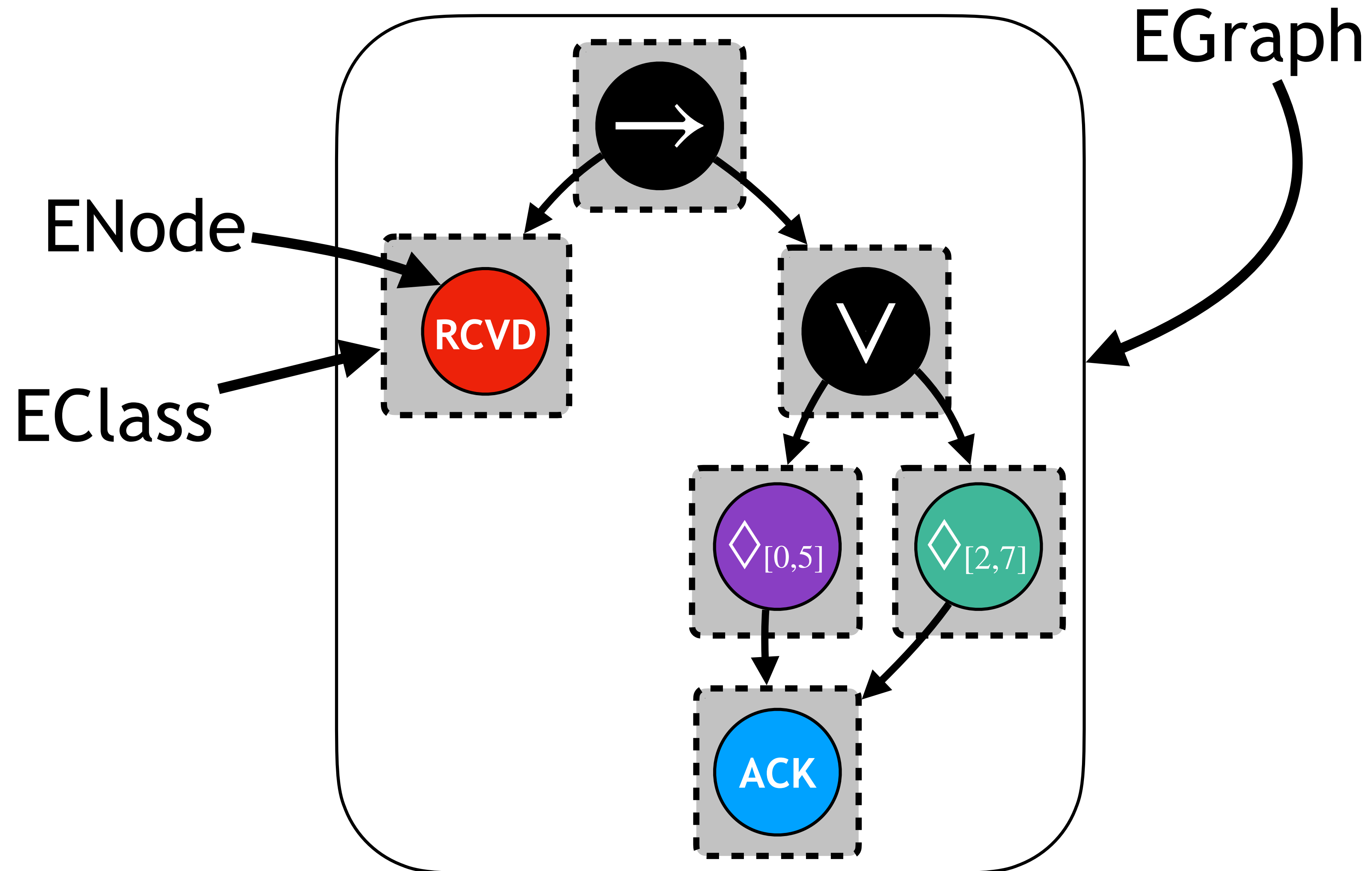
Our work adapts equality saturation to automatically find smaller equivalent R2U2 monitors.



Our work adapts equality saturation to automatically find smaller equivalent R2U2 monitors.

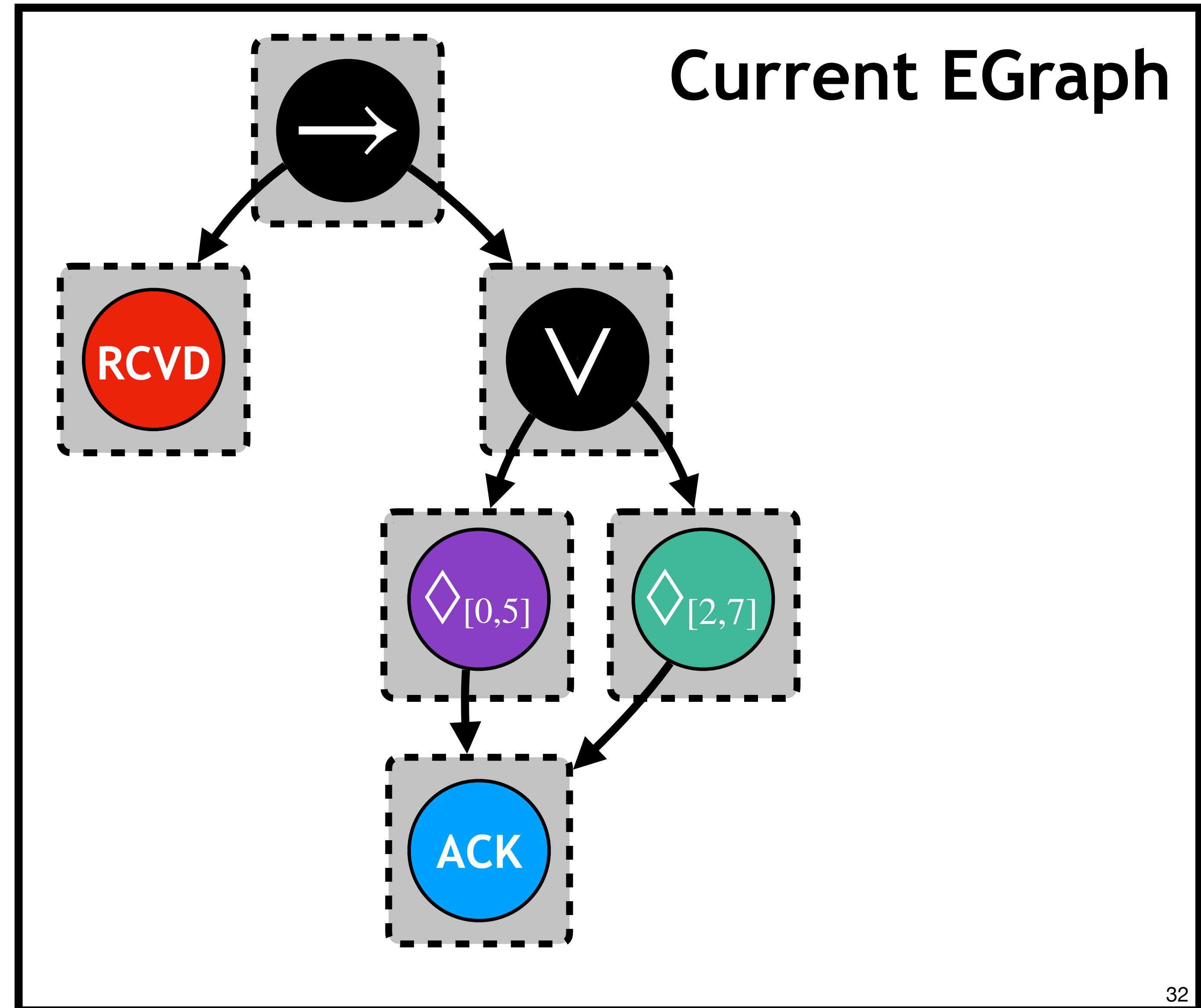


Equality Saturation applies rewrites to an EGraph until no more can be applied.



Equality Saturation applies rewrites to an EGraph until no more can be applied.

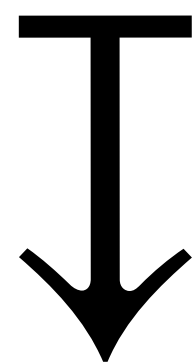
Applying rewrite:



Equality Saturation applies rewrites to an EGraph until no more can be applied.

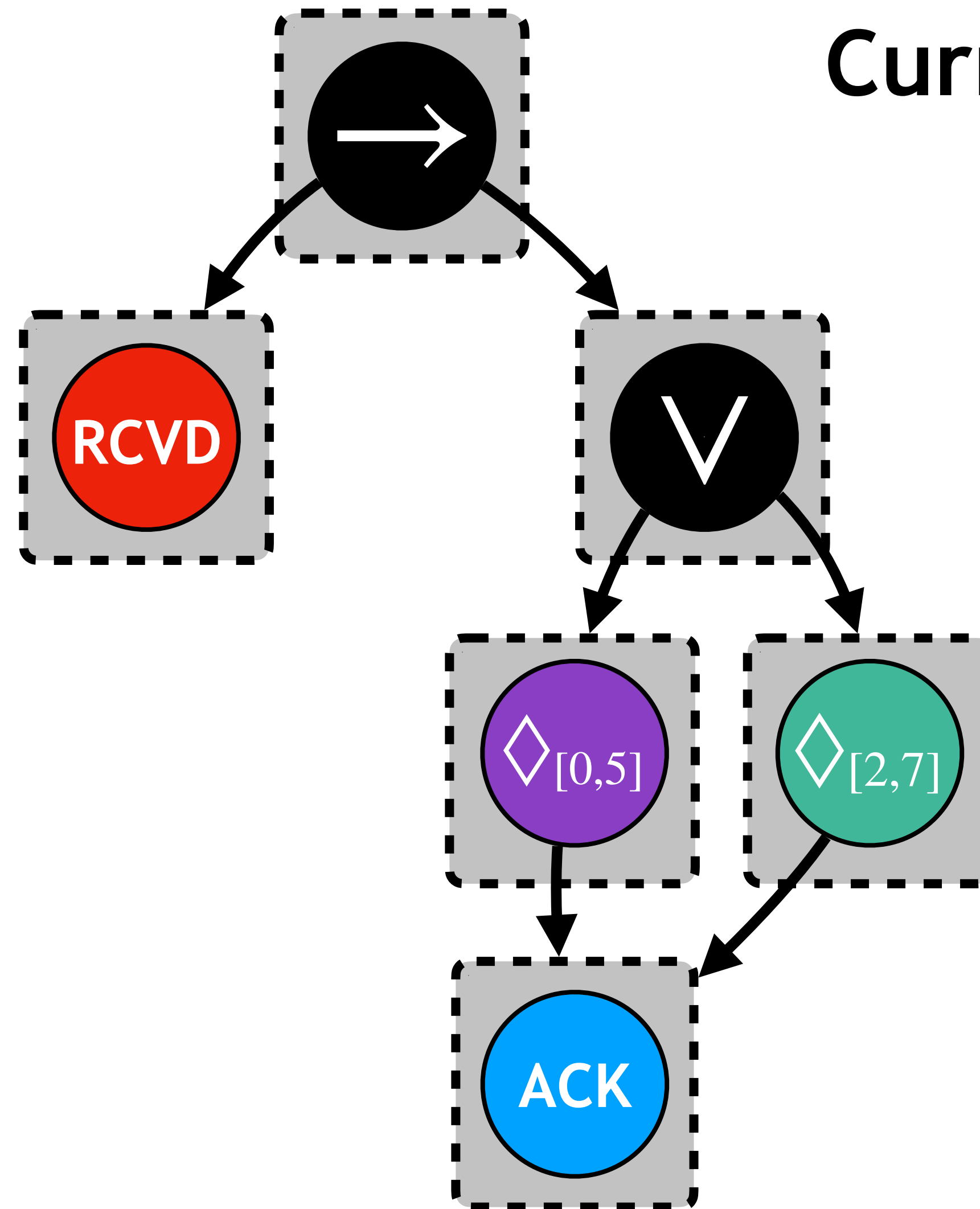
Applying rewrite:

$\diamond_{[0,5]} \text{ ACK} \vee \diamond_{[2,7]} \text{ ACK}$



$\diamond_{[0,5]} (\diamond_{[0,0]} \text{ ACK} \vee \diamond_{[2,2]} \text{ ACK})$

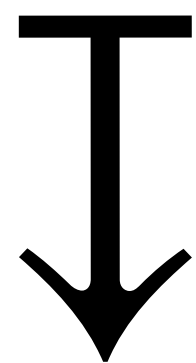
Current EGraph



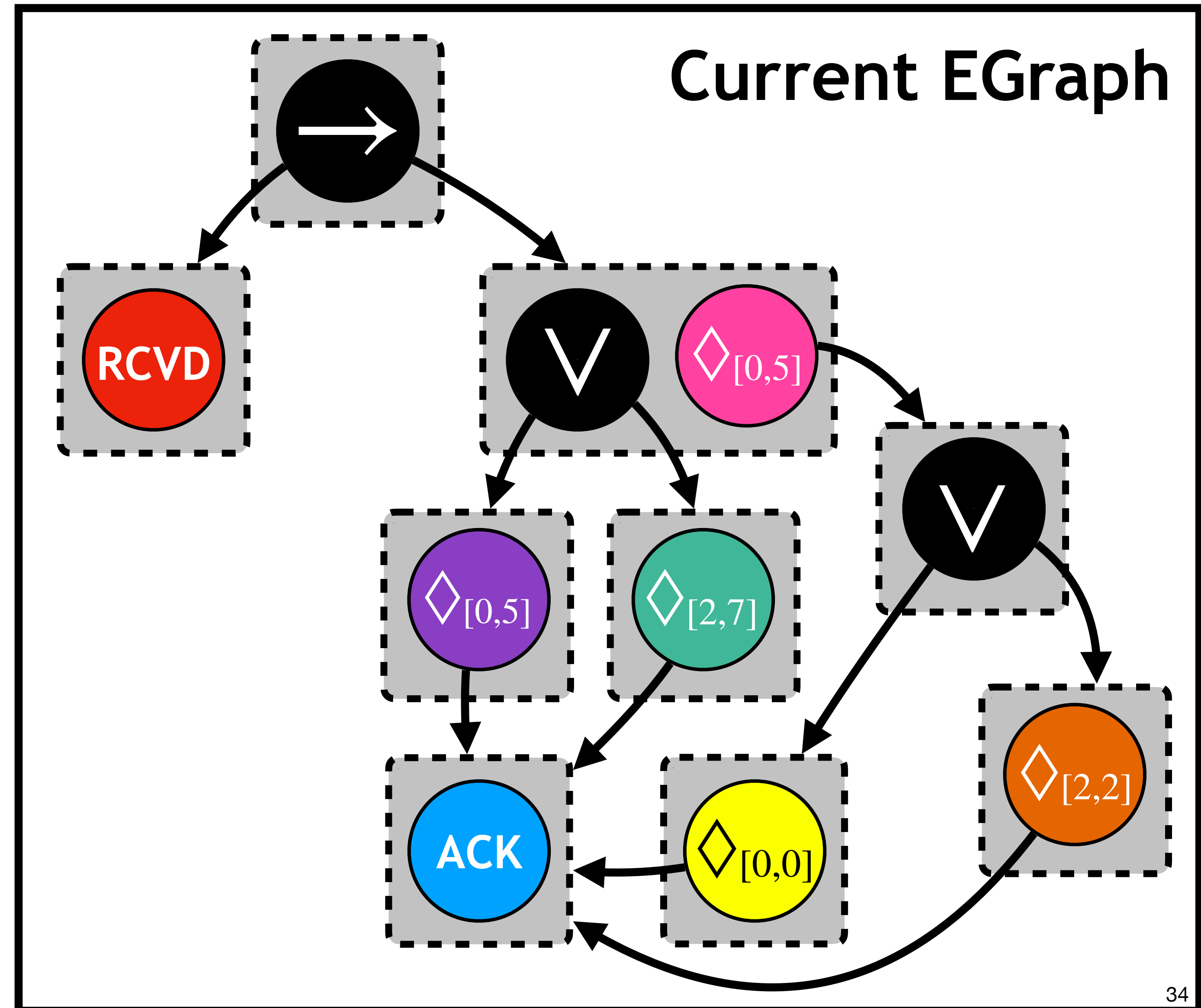
Equality Saturation applies rewrites to an EGraph until no more can be applied.

Applying rewrite:

$\diamond_{[0,5]} \text{ ACK} \vee \diamond_{[2,7]} \text{ ACK}$



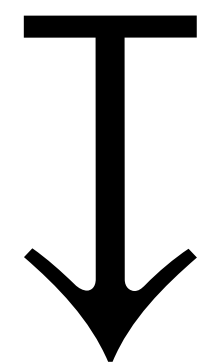
$\diamond_{[0,5]} (\diamond_{[0,0]} \text{ ACK} \vee \diamond_{[2,2]} \text{ ACK})$



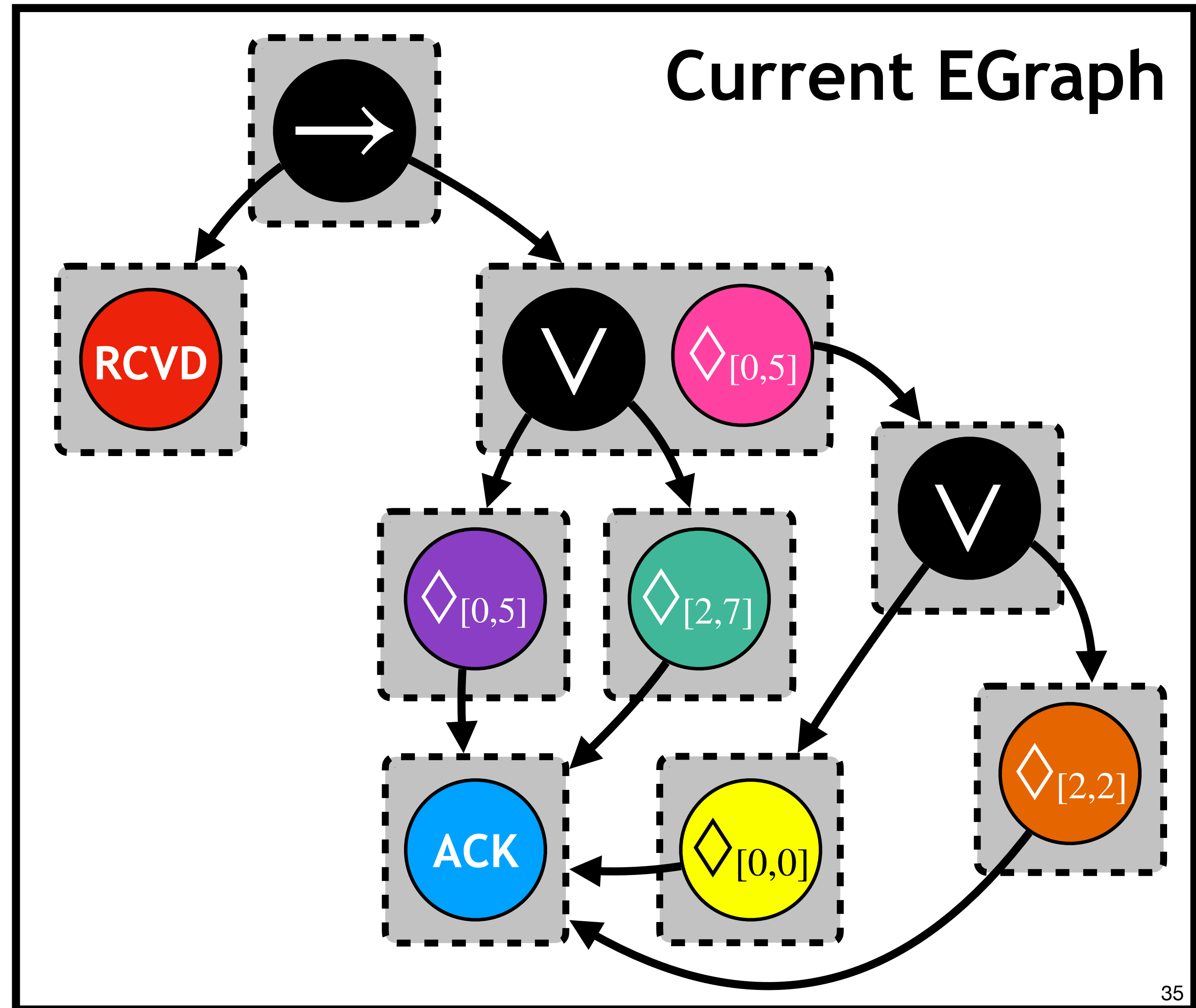
Equality Saturation applies rewrites to an EGraph until no more can be applied.

Applying rewrite:

$\diamond_{[0,0]}$ ACK



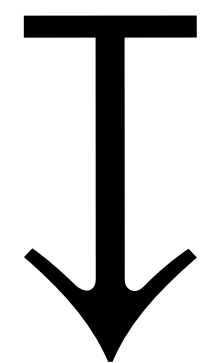
ACK



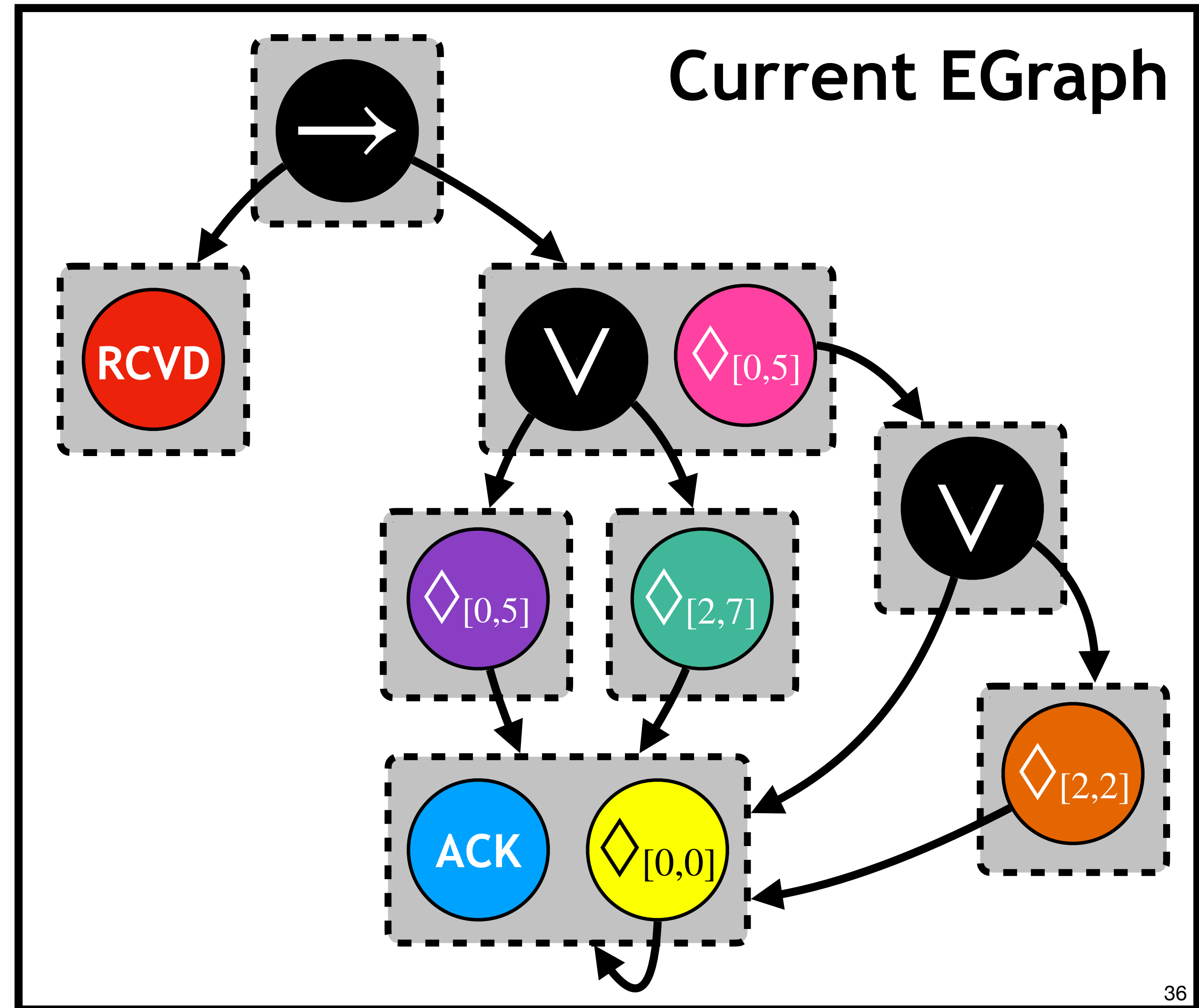
Equality Saturation applies rewrites to an EGraph until no more can be applied.

Applying rewrite:

$\diamond_{[0,0]}$ ACK



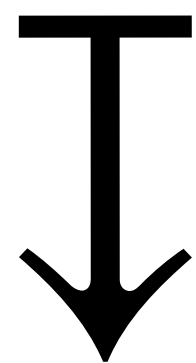
ACK



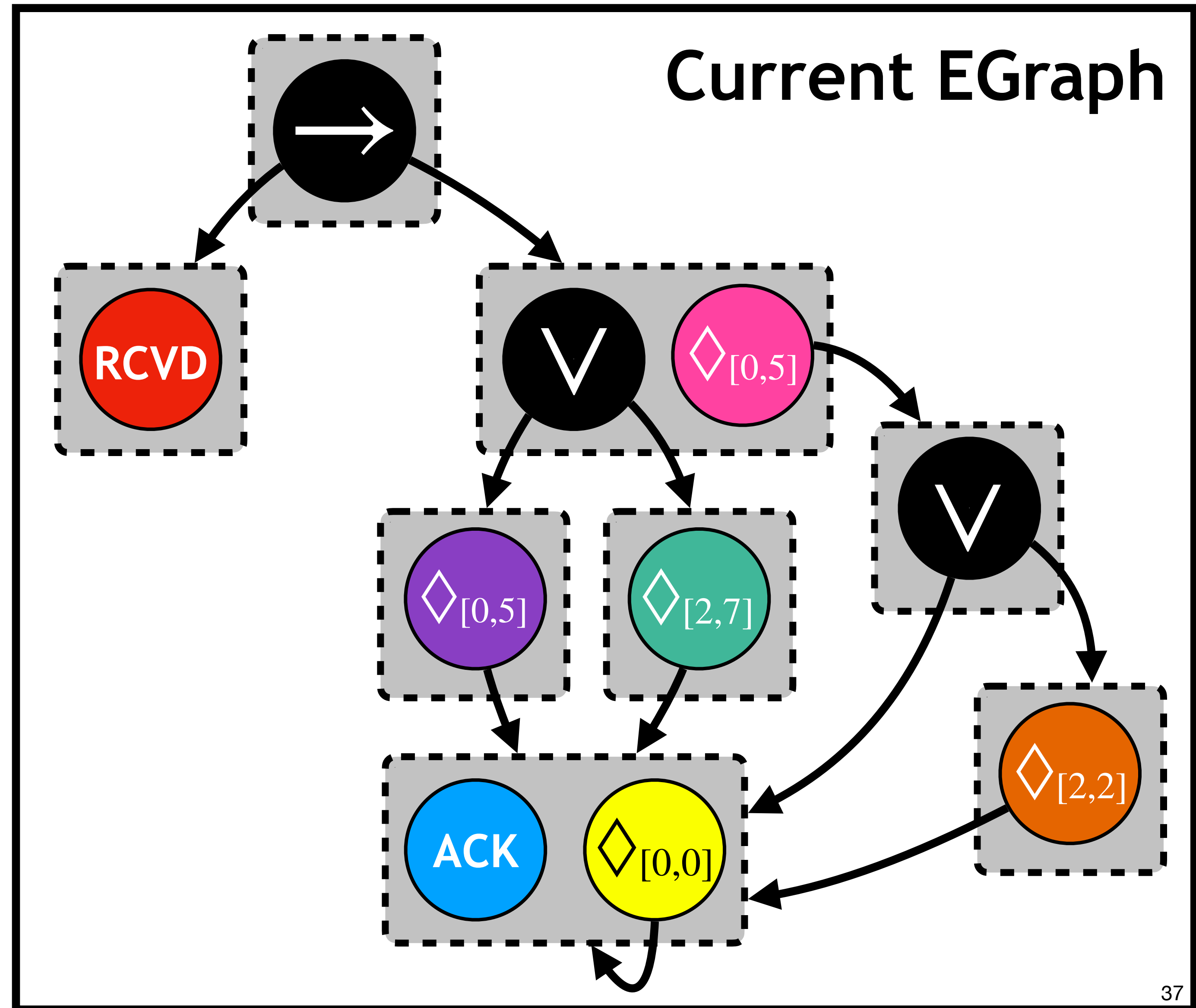
Equality Saturation applies rewrites to an EGraph until no more can be applied.

Applying rewrite:

$\diamond_{[0,5]}$ ACK $\vee$ $\diamond_{[2,7]}$ ACK



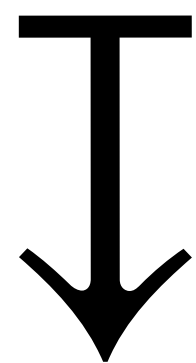
$\diamond_{[0,7]}$ ACK



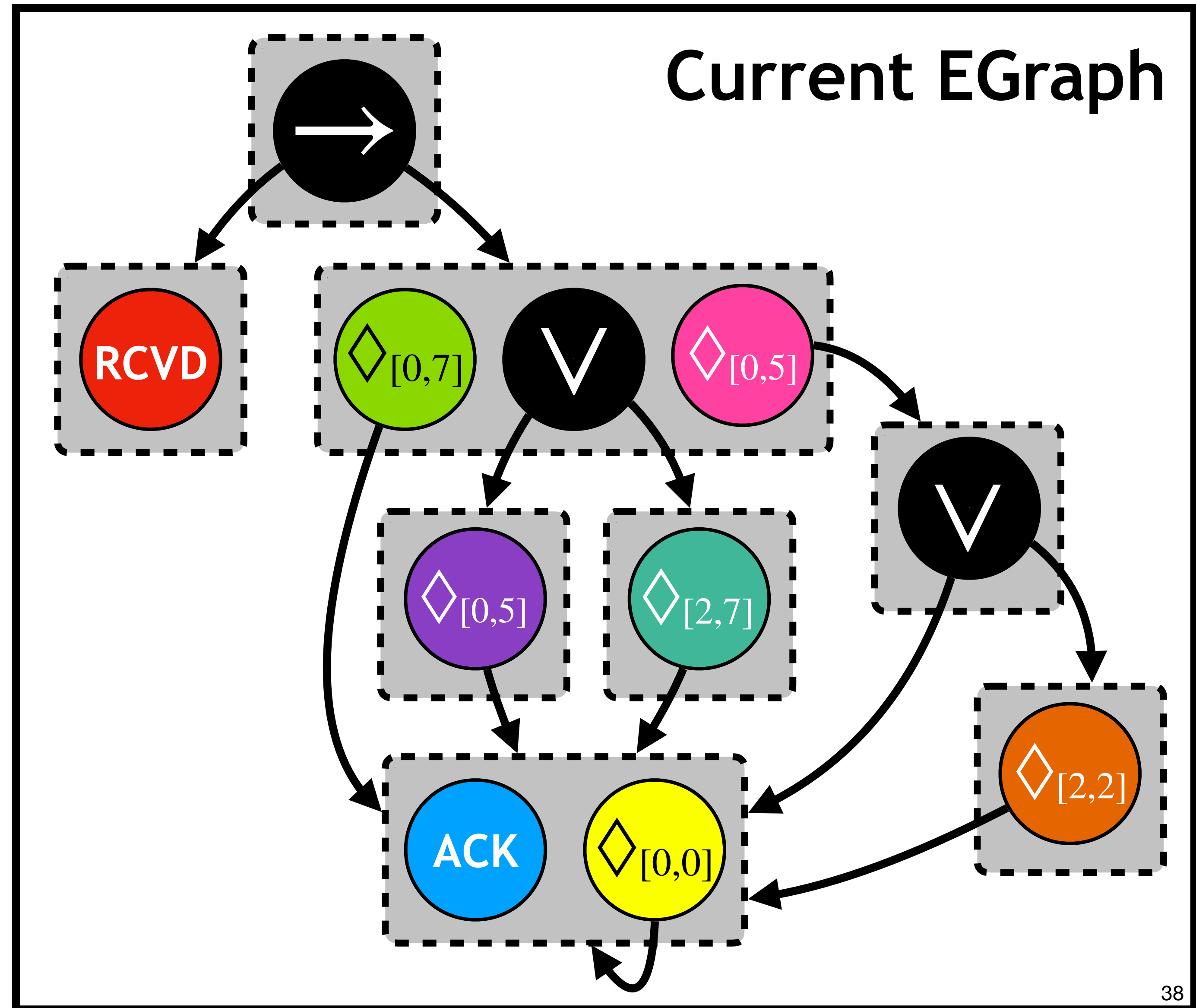
Equality Saturation applies rewrites to an EGraph until no more can be applied.

Applying rewrite:

$\diamond_{[0,5]}$ ACK $\vee$ $\diamond_{[2,7]}$ ACK



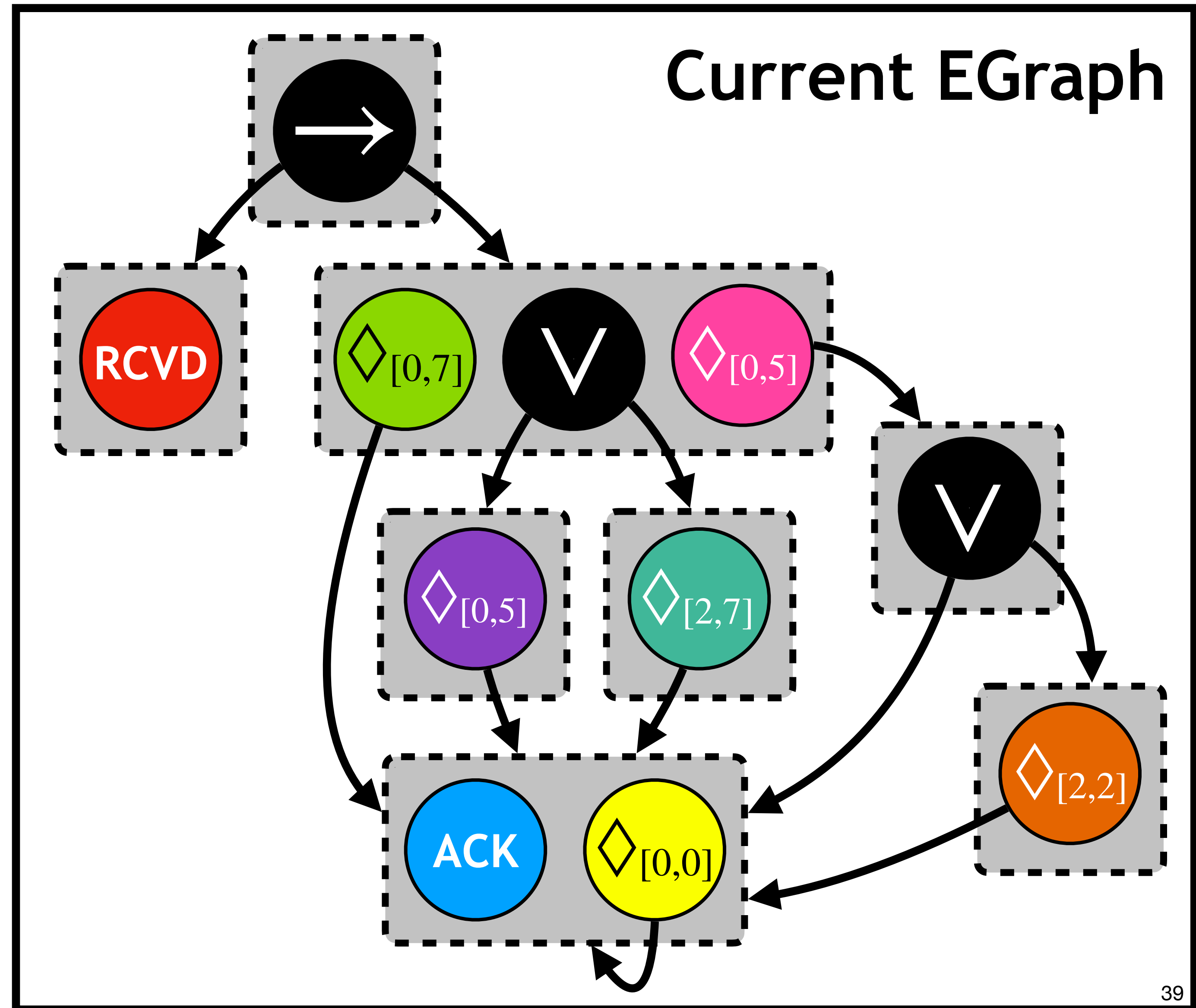
$\diamond_{[0,7]}$ ACK



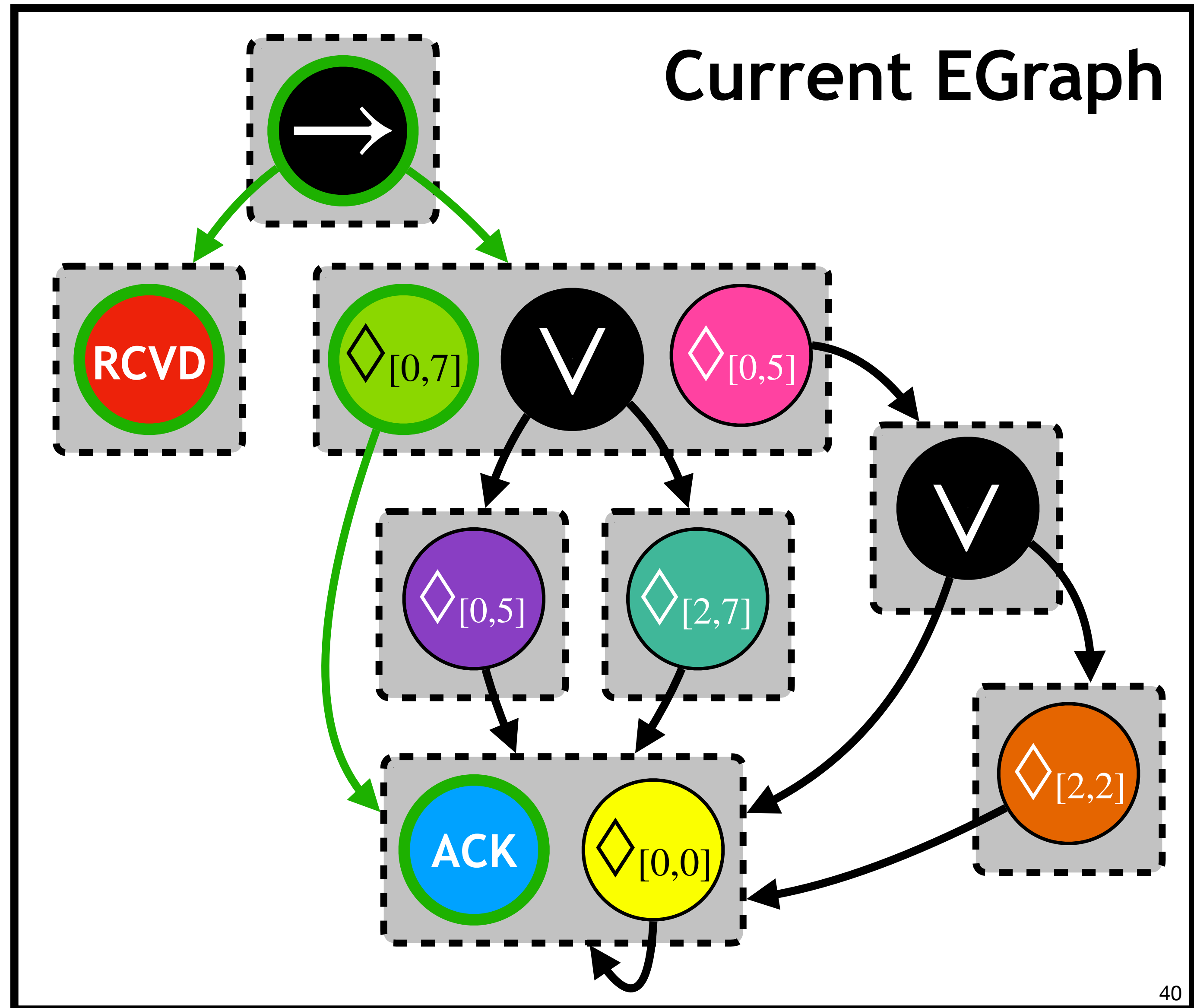
Equality Saturation applies rewrites to an EGraph until no more can be applied.

Applying rewrite:

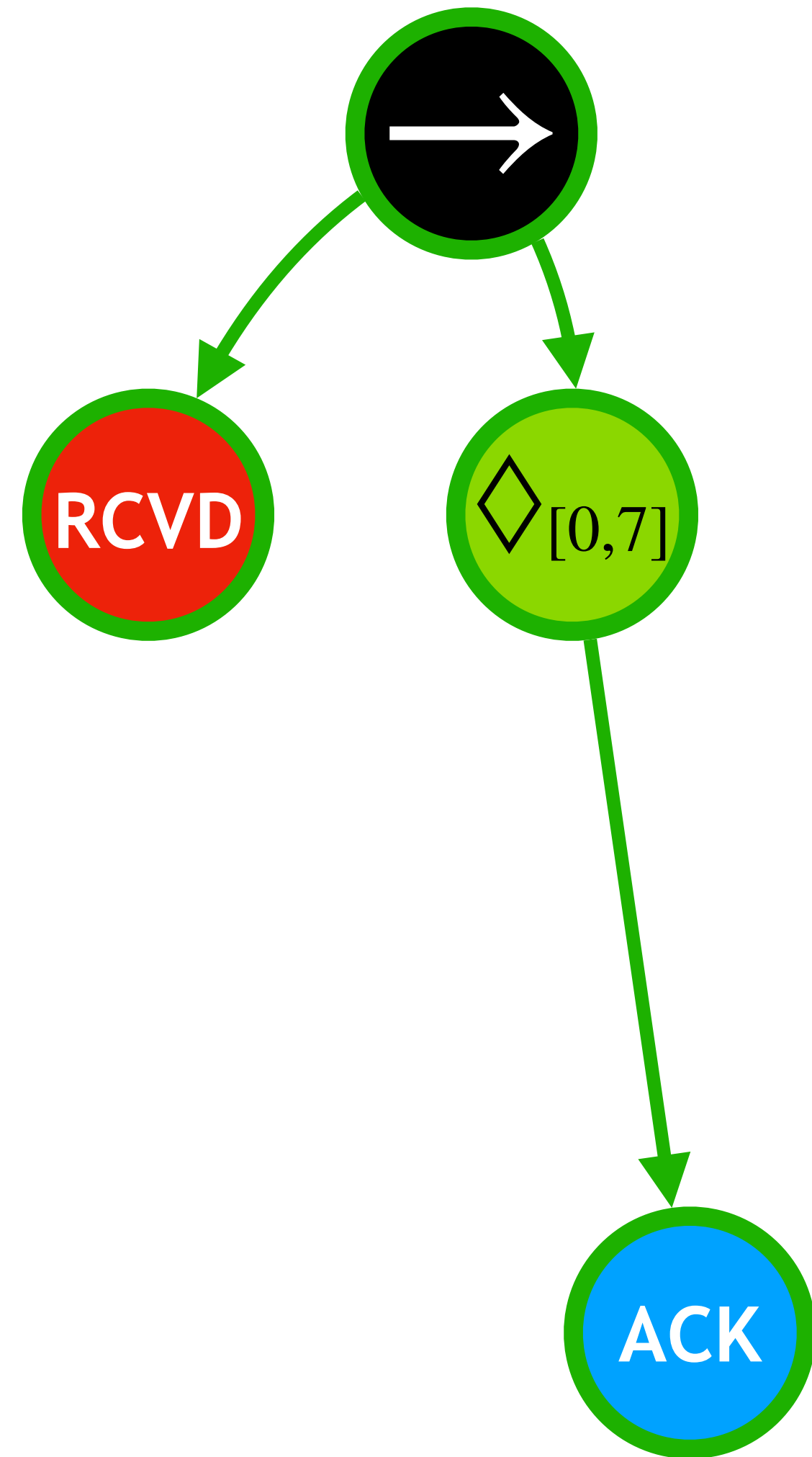
No more, EGraph is saturated!



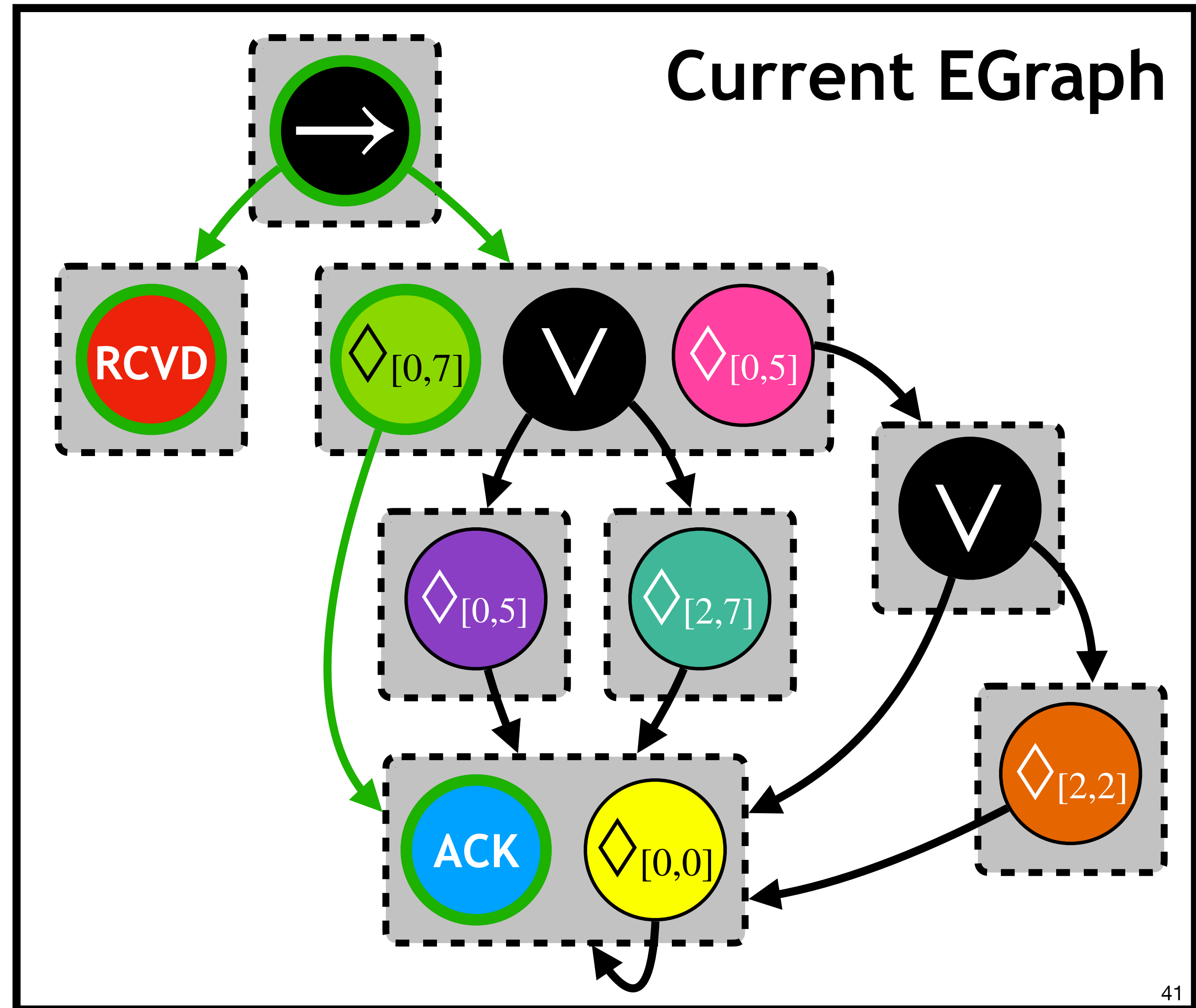
We extract a formula from the EGraph using two methods: an ILP encoding and a greedy algorithm.



We extract a formula from the EGraph using two methods: an ILP encoding and a greedy algorithm.



Extract



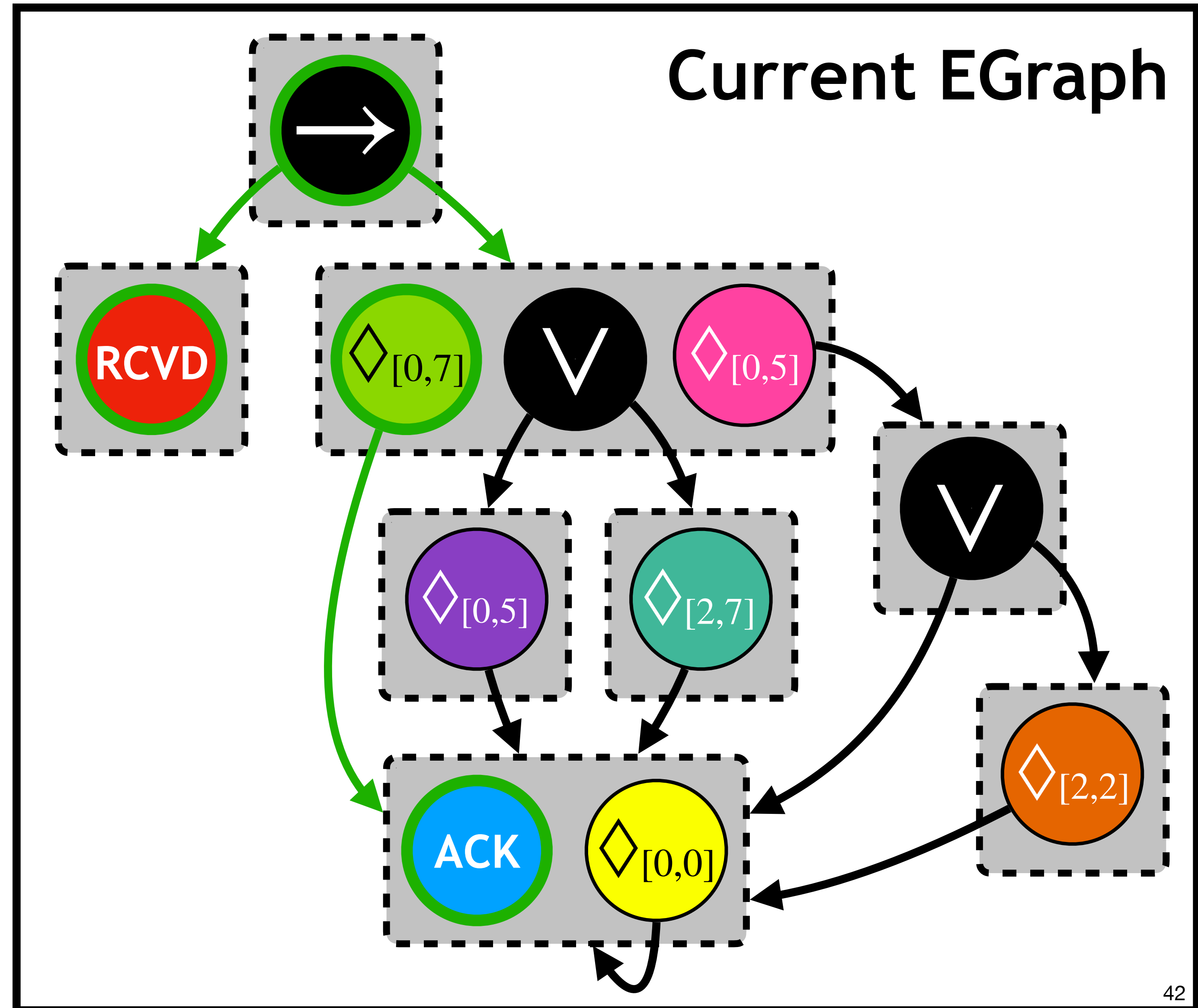
We extract a formula from the EGraph using two methods: an ILP encoding and a greedy algorithm.

(A) Integer Linear Programming (ILP)

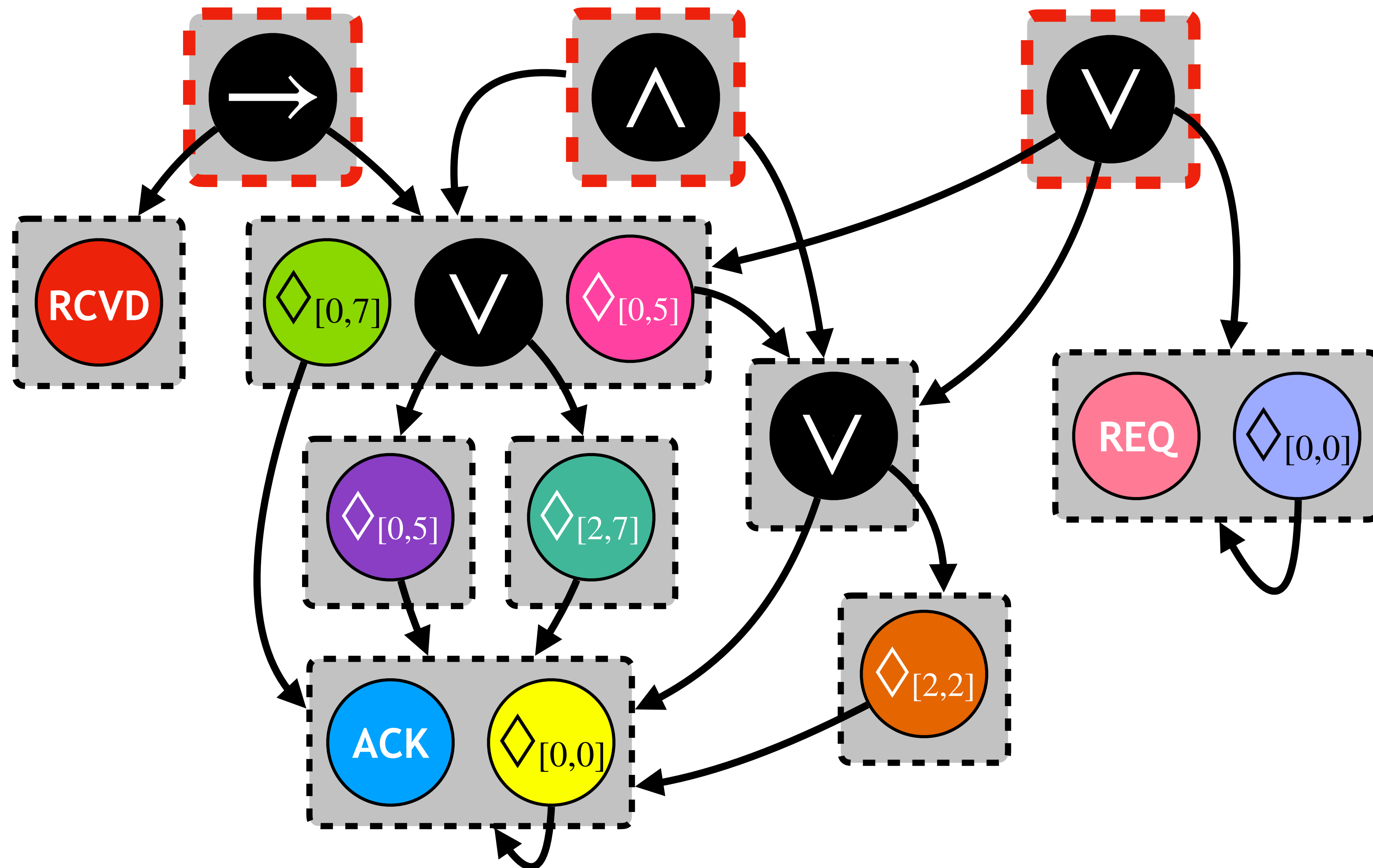
1. Declare Boolean (0/1) variable for each **enode** and **eclass**
2. Add constraints to ensure **valid** formula
3. Minimize **cost** of the output formula

(B) Greedy

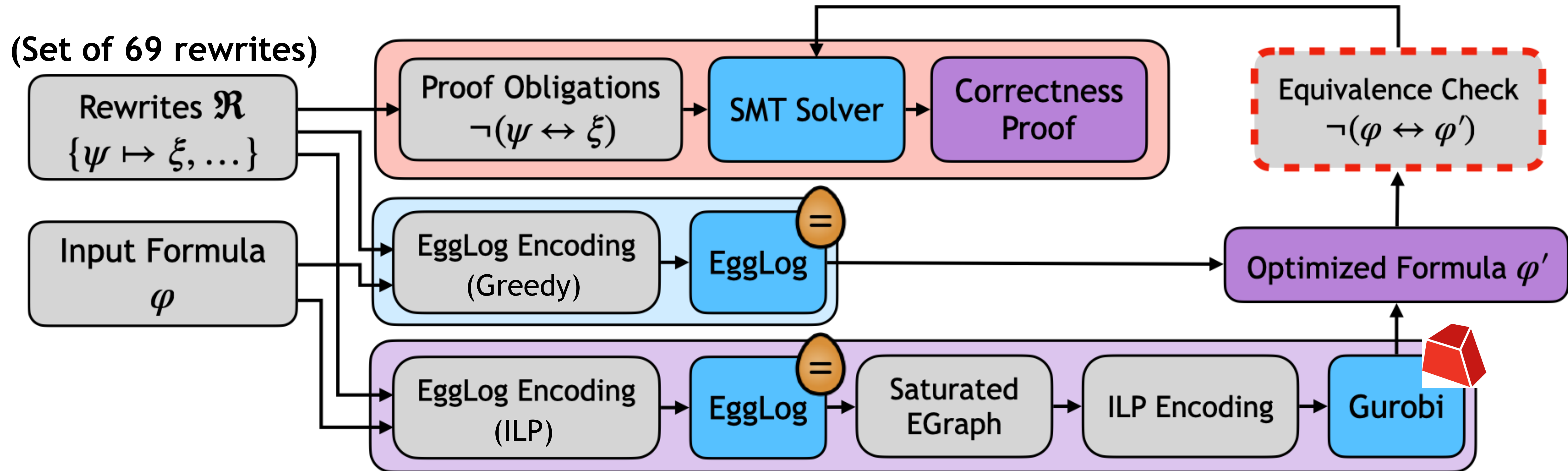
1. Approximate MEM for each ENode
2. Take lowest cost ENode from each EClass, starting from the root



We can perform equality saturation over **multiple formulas** to take advantage of R2U2's ability to node share.



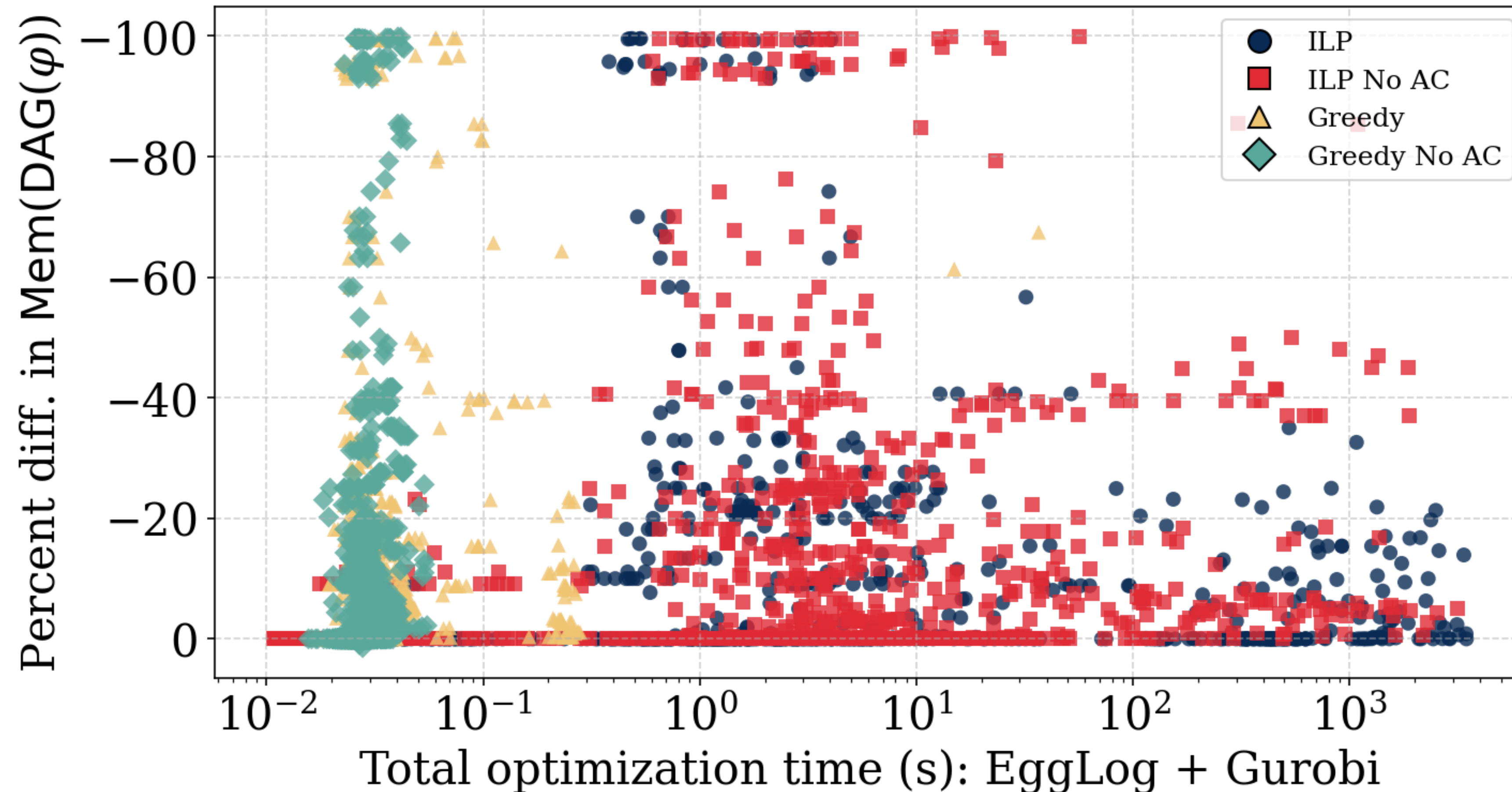
Our toolchain applies **verified rewrites** to automatically produce R2U2-optimized MTL formulas using **EggLog** (for equality saturation) and **Gurobi** (for ILP solving).



Experimental Setup

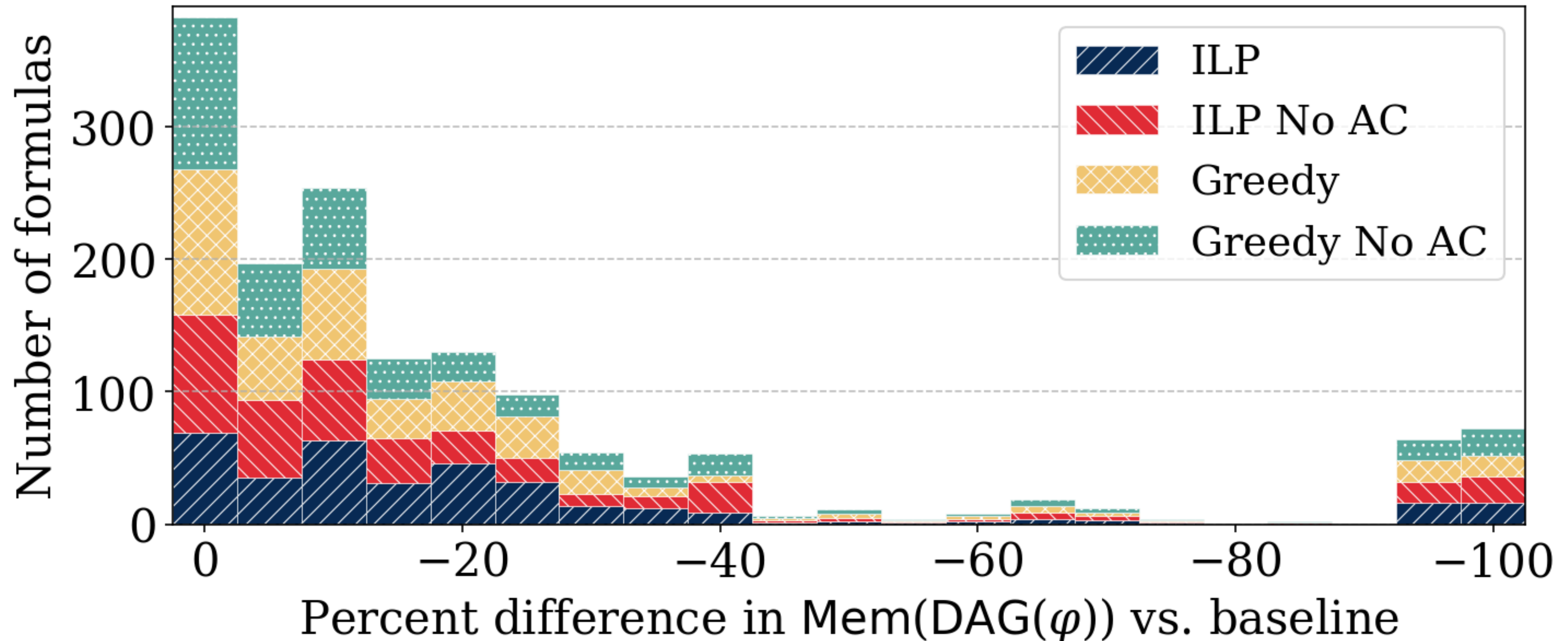
- Benchmarks: 1,172 real-world formulas in total
 - Boeing WBS, NASA ATC, Fluxgate, UTM, MAVLink, MTL patterns
- 1 hour time limit and 64GB memory limit each for EggLog and Gurobi
- Ran 4 toolchain configurations for each formula (AC = associate+commutative):
 1. ILP extraction, AC rewrites enabled (ILP)
 2. ILP extraction, AC rewrites disabled (ILP No AC)
 3. Greedy extraction, AC rewrites enabled (Greedy AC)
 4. Greedy extraction, AC rewrites disabled (Greedy No AC)
- All graphs omit unchanged formulas: ILP (815), ILP No AC (592), Greedy (540), Greedy No AC (538)..

Greedy extraction is significantly faster than and performs similar to **ILP extraction**.



- **AC rewrites** increase cost with little benefit.
- 16 formulas saw an increase in MEM with **Greedy** extraction.

Most formulas see a modest decrease in MEM, others see a dramatic decrease.

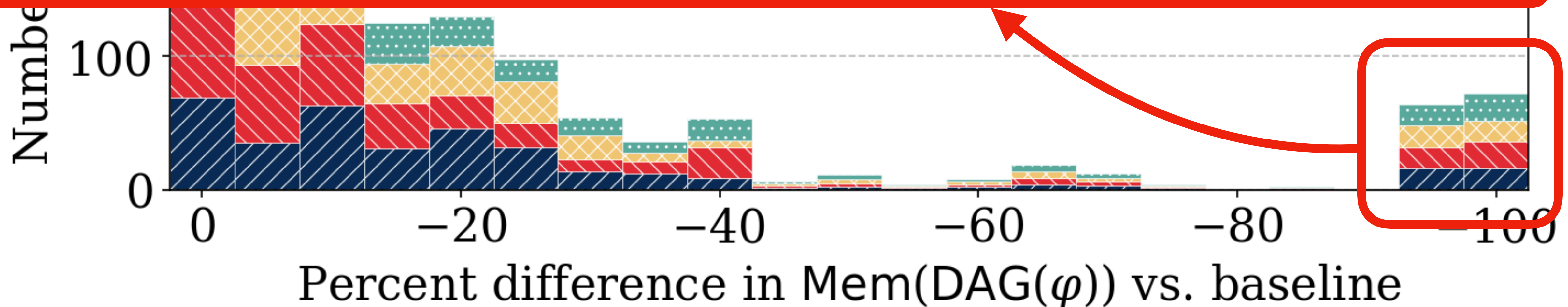


wbs_arch3_inst_ctrl_sys_expected_behavior_as_cmd_pair_2_6_norm_guarantee
(Boeing WBS benchmark):

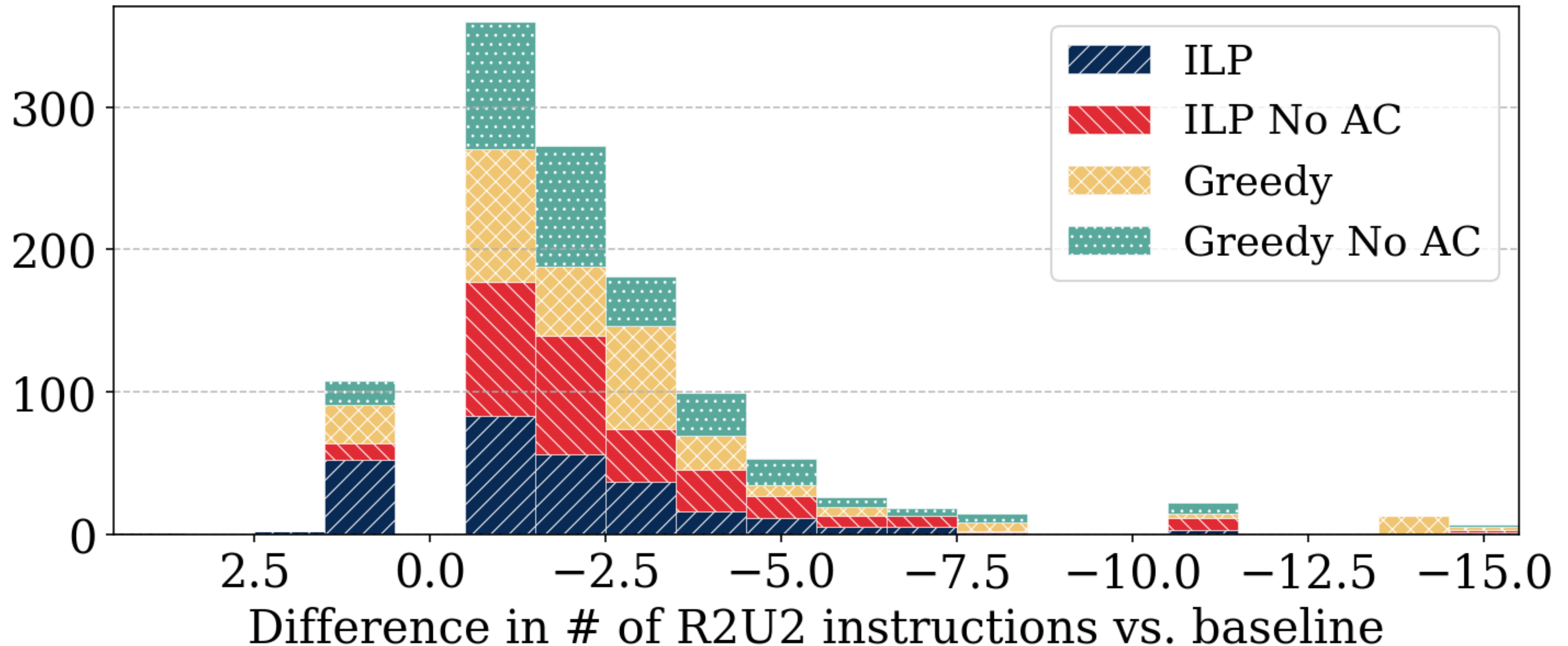
```
(  
  (G[0,1000] ((! (__expr104 && __expr0) && (__expr11 || __expr13))) || __expr126)) &&  
  (G[0,1000] ((! __expr126) || ((__expr104 && __expr0) && (__expr11 || __expr13))))  
)
```

→

```
G[0,1000] (  
  (__expr126 → (__expr104 && (__expr13 || __expr11) && __expr0)) &&  
  ((__expr104 && (__expr13 || __expr11) && __expr0) → __expr126)  
)
```



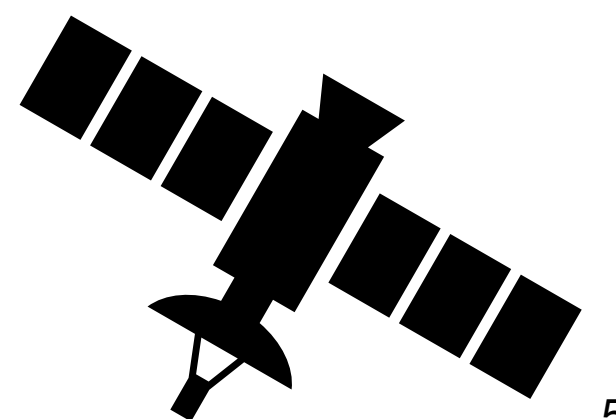
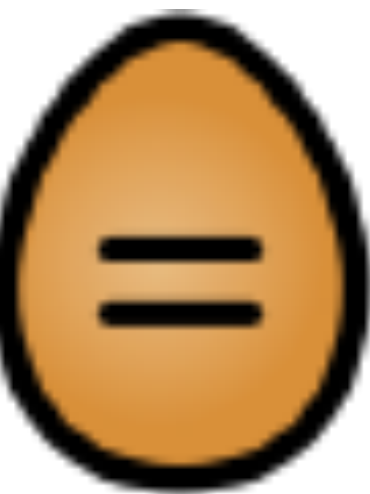
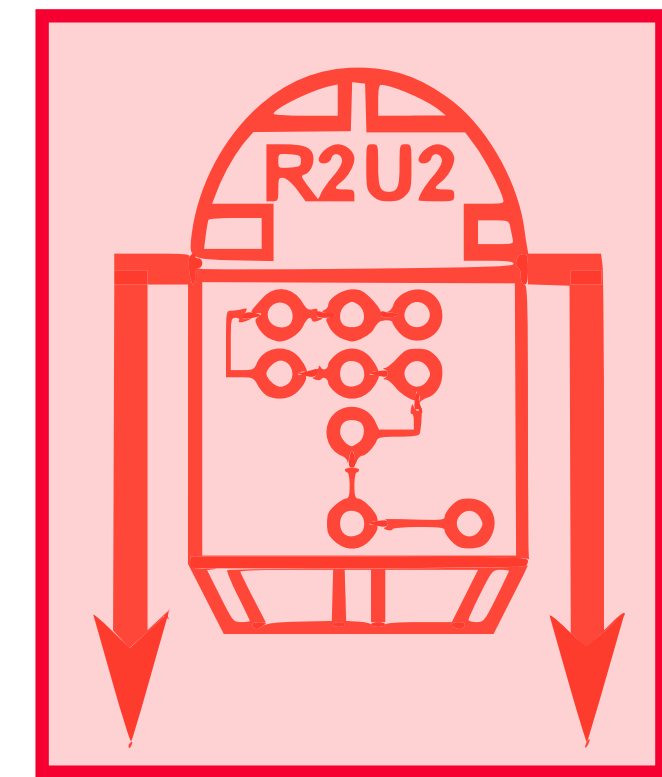
Toolchain reduces the number of R2U2 instructions for most formulas, increases for some.





Equality saturation shrinks **R2U2 monitors** for resource-constrained embedded systems.

- Curated a set of 69 MLTL rewrites.
- Encoded MLTL and rewrites with **EggLog** as off-the-shelf equality saturation engine.
- Defined two methods for extracting formulas from saturated EGraph (**ILP** and **Greedy**).
- Implemented toolchain with verified rewrites to automatically optimize MLTL formulas for R2U2 monitor configurations.
- Demonstrated the method **effectively reduces memory and WCET for R2U2 monitor configurations** over benchmark set of 1,172 formulas.



- Looking to adapt the method to other logics (LTL, LTLf, STL, MLTL Multi-type, Metric Dynamic Logic).
- May be useful for minimizing automata representation of formulas.

We verify that each rewrite is correct by reducing to MTL satisfiability checking.

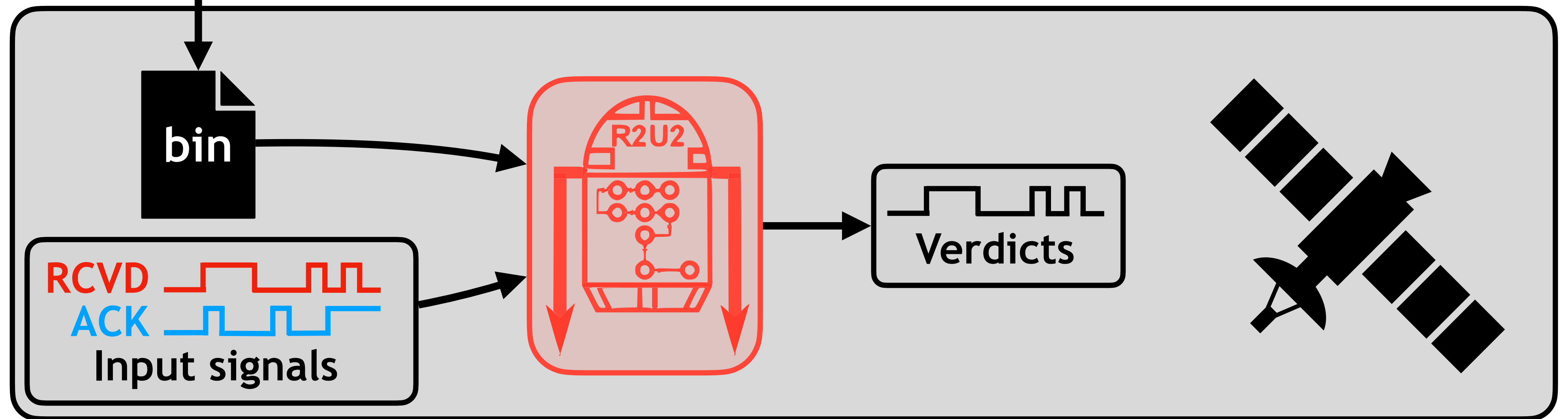
Example: $\Box_{[a,b]} \Box_{[c,d]} \varphi \mapsto \Box_{[a+c,b+d]} \varphi$

Formulas are equivalent iff following formula is unsatisfiable:

$$\exists a, b, c, d. \neg \left(\Box_{[a,b]} \Box_{[c,d]} \varphi \leftrightarrow \Box_{[a+c,b+d]} \varphi \right)$$

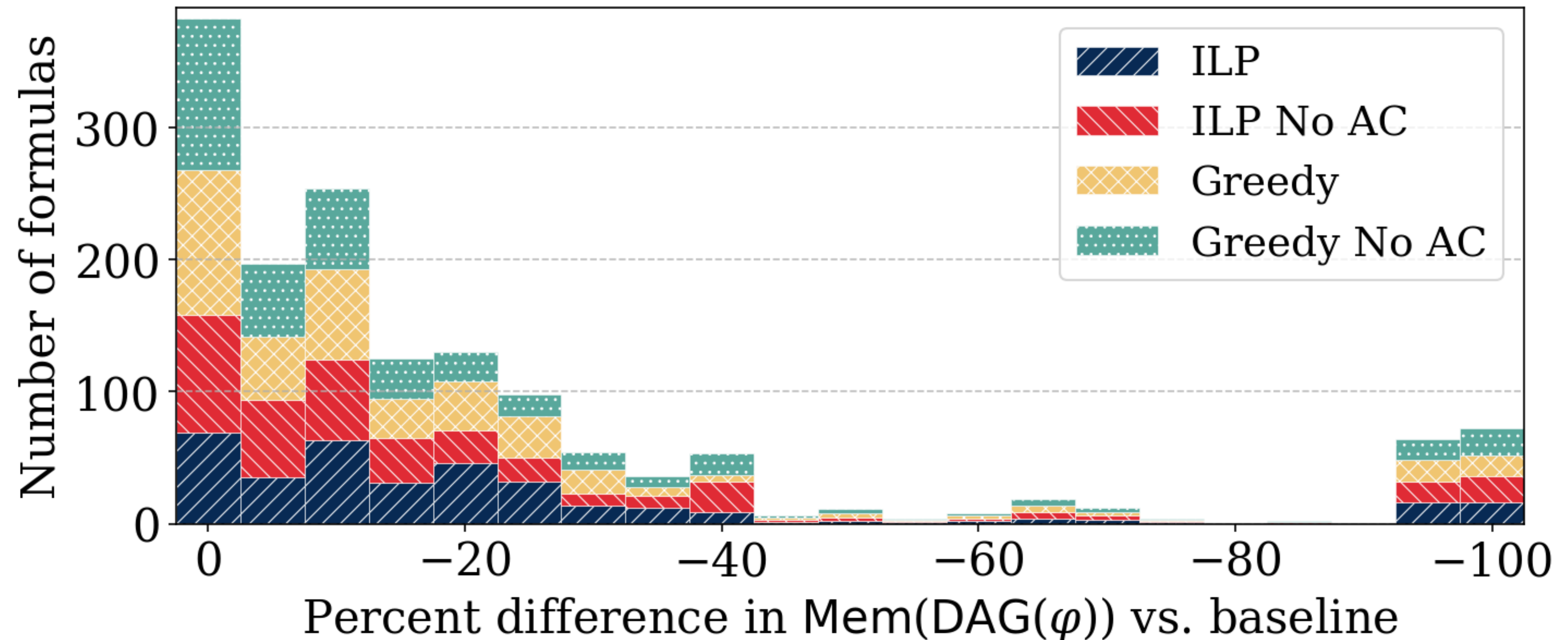
- Encode using SMT:
 - For each symbolic interval variable, declare an uninterpreted integer.
 - For each interval $[a, b]$, assert $a \leq b$, $a \geq 0$, and $b \geq 0$.
 - Check satisfiability of $\Box_{[a,b]} \Box_{[c,d]} \varphi \leftrightarrow \Box_{[a+c,b+d]} \varphi$ using SMT solver.

RCVD $\rightarrow \left(\diamond_{[0,5]} \text{ACK} \vee \diamond_{[2,7]} \text{ACK} \right)$



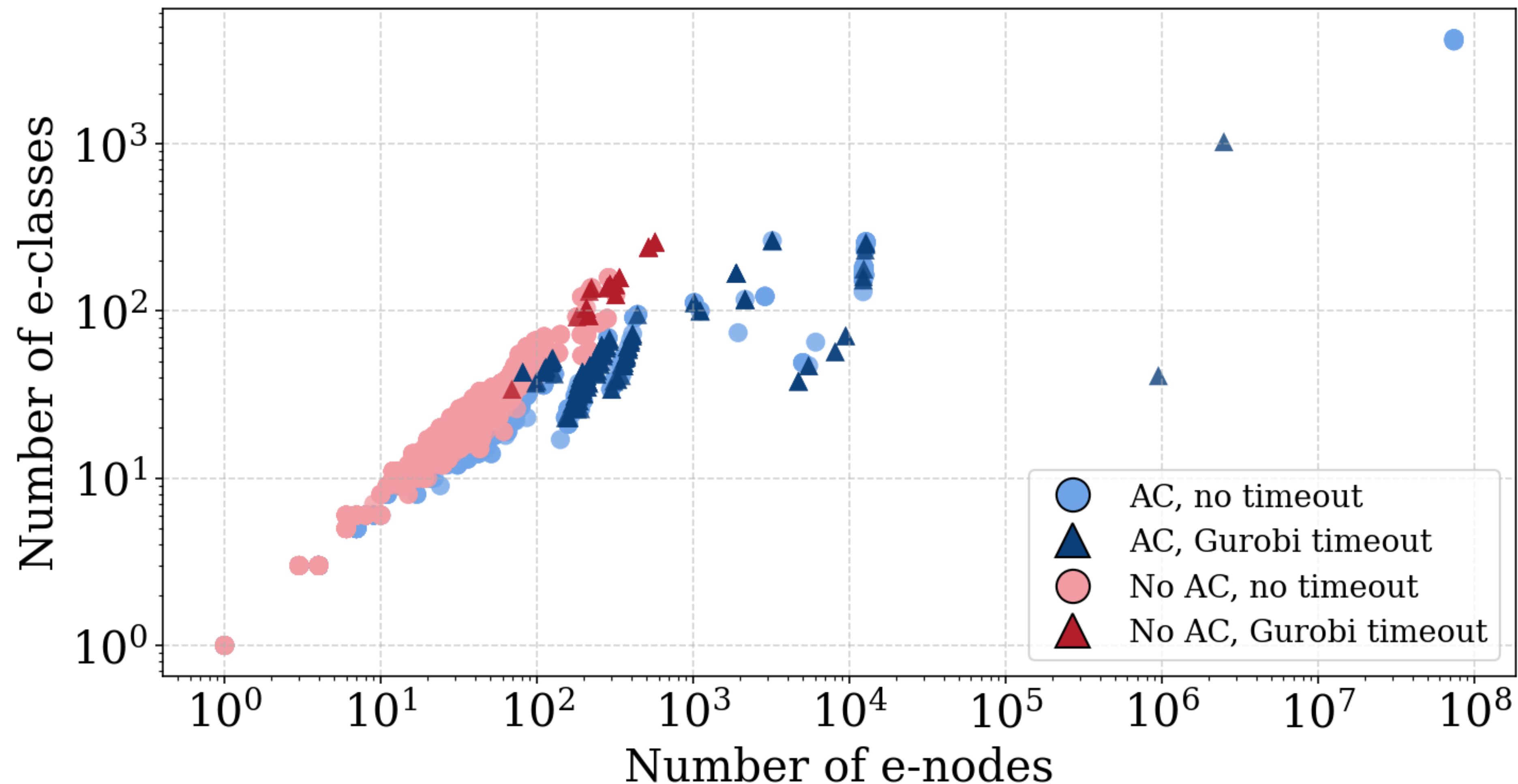
Most formulas see a modest decrease in MEM, others see a dramatic decrease.

Avg % reduction by method: **ILP (6.92%)**, **ILP No AC (11.37%)**, **Greedy (9.49%)**, **Greedy No AC (10.14%)**, Virtual best (13.72%).



EGraph size significantly impacts scalability for ILP extraction.

71% of EGraphs had **less than 100 ENodes and 100 EClasses**, of which only **5%** saw a time/memout.



AC rewrites meaningfully increase EggLog memouts and Gurobi timeouts.

Method	Egglog Time	Gurobi Time	Par-2 Score
	avg (# MO)	avg (# TO)	avg
I-AC	3.109s (75)	170.528s (233)	2402.898s
I-No-AC	0.013s (0)	61.772s (47)	348.044s
G-AC	0.089s (85)	N/A	522.267s
G-No-AC	0.029s (0)	N/A	0.029s