UNIVERSITY OF WATERLOO

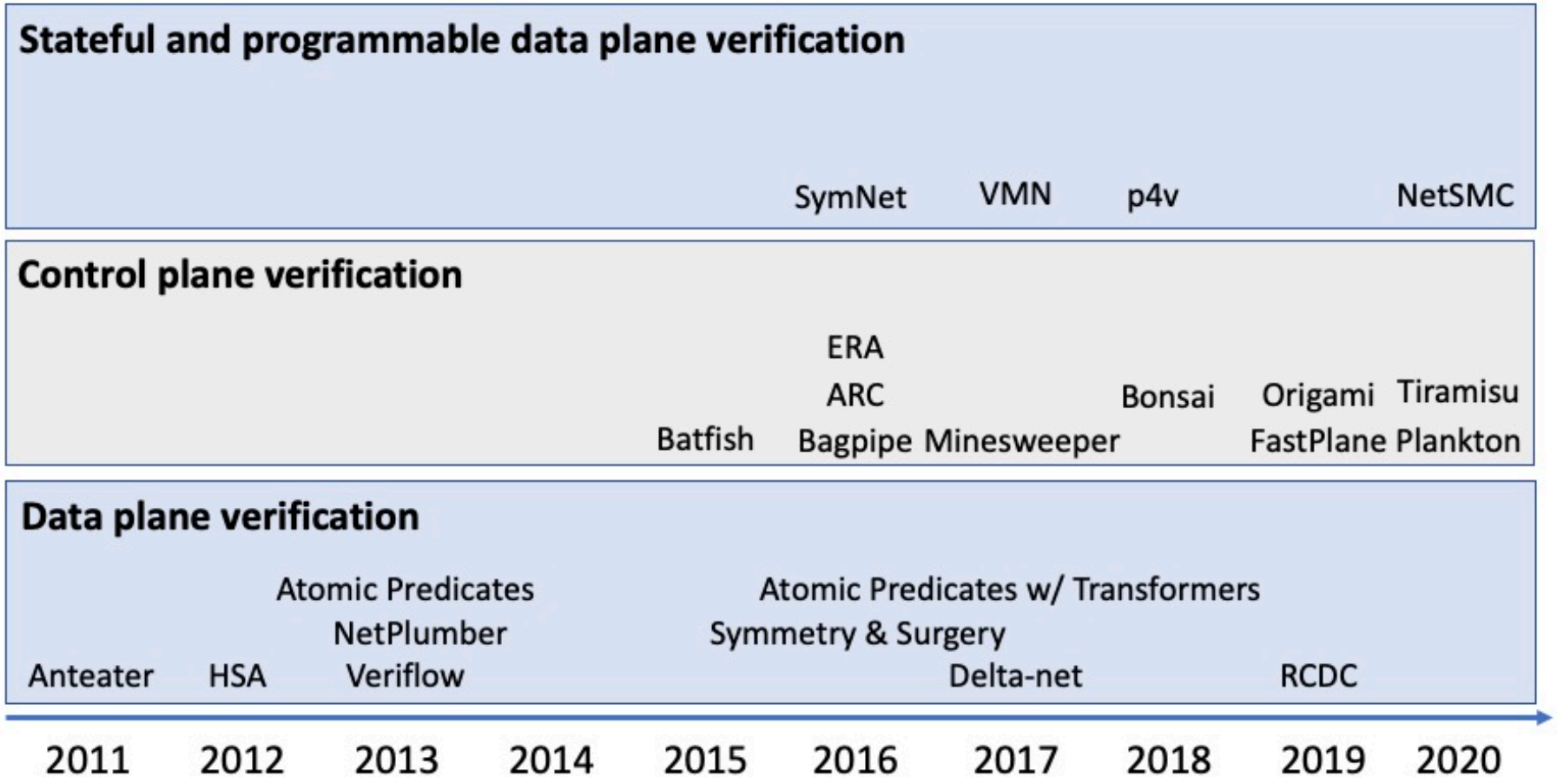
Formal Methods for Network Performance Analysis

Mina Tahmasbi Arashloo
University of Waterloo

Ryan Beckett
Microsoft

Rachit Agarwal
Cornell University

The shift towards automated formal analysis



"Capturing the state of research on network verification"
Ryan Beckett and Ratul Mahajan, netverify.fun

The shift towards automated formal analysis

1 Create a mathematical model of the network

$\forall t \ \ (dstip(t) = A \wedge at(s_1, t)) \rightarrow at(s_2, t+1)$

$\forall t \ \ dstip(t) = B \wedge \forall s \ \ \neg at(s, t+1)$

…

The shift towards automated formal analysis

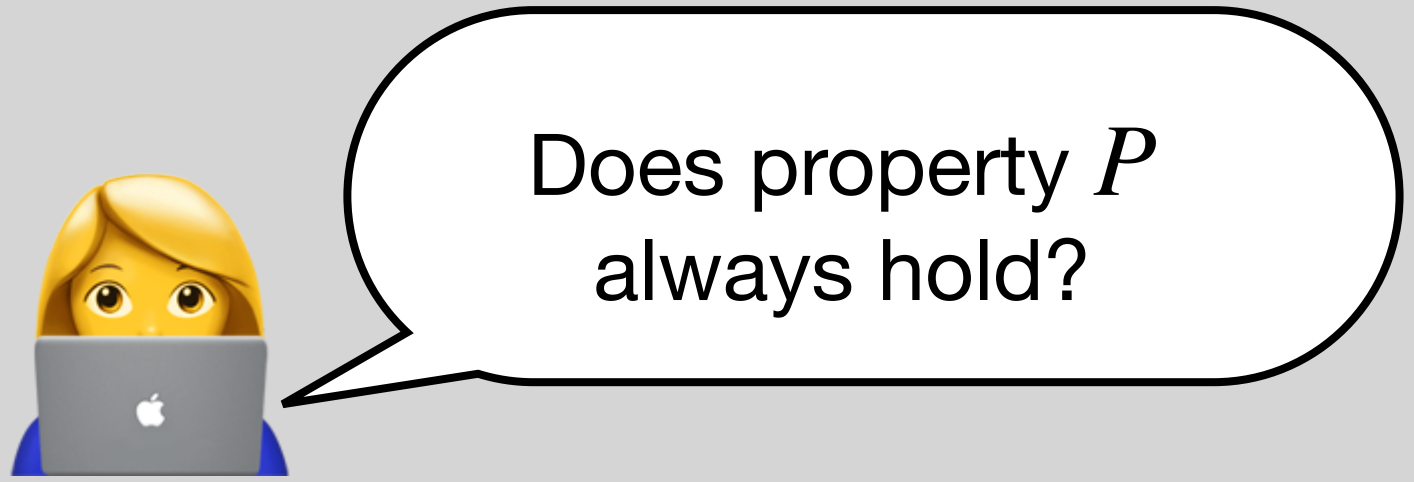
1 Create a mathematical model of the network

$$\forall t \;\; (dstip(t) = A \land at(s_1, t)) \rightarrow at(s_2, t+1)$$

$$\forall t \;\; dstip(t) = B \land \forall s \;\; \neg at(s, t+1)$$

...

2 Specify desired property



The shift towards automated formal analysis

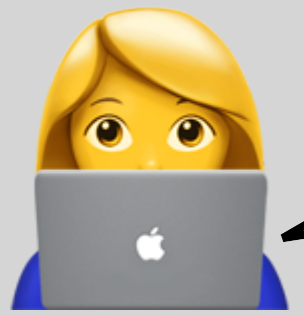
1 Create a mathematical model of the network

$$\forall t \ \ (dstip(t) = A \land at(s_1, t)) \rightarrow at(s_2, t+1)$$

$$\forall t \ \ dstip(t) = B \land \forall s \ \ \neg at(s, t+1)$$

...

2 Specify desired property



Does property P always hold?

3 Automatically analyze the entire input space.

- Model checking
- Symbolic execution
- ...

The shift towards automated formal analysis

e.g., packets entering the network

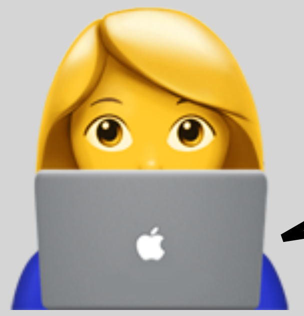
1 Create a mathematical model of the network

$\forall t \ (dstip(t) = A \wedge at(s_1, t)) \rightarrow at(s_2, t+1)$

$\forall t \ dstip(t) = B \wedge \forall s \ \neg at(s, t+1)$

...

2 Specify desired property



Does property P always hold?

3 Automatically analyze the entire input space.

- Model checking
- Symbolic execution
- ...

The shift towards automated formal analysis

e.g., packets entering the network

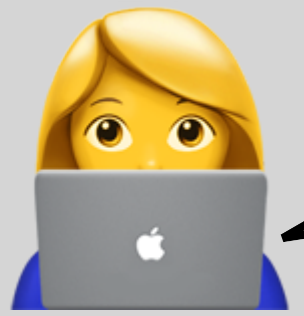
1 Create a mathematical model of the network

$$\forall t \; (dstip(t) = A \land at(s_1, t)) \rightarrow at(s_2, t+1)$$

$$\forall t \; dstip(t) = B \land \forall s \; \neg at(s, t+1)$$

...

2 Specify desired property



Does property P always hold?

3 Automatically analyze the entire input space.

- Model checking
- Symbolic execution
- …

4 Prove or disprove the property

✔ Property P always holds

✘ An example input for which P does not hold

Existing work focuses on functional correctness

1 Create a mathematical model of the network

$$\forall t \ (dstip(t) = A \wedge at(s_1, t)) \rightarrow at(s_2, t+1)$$

$$\forall t \ dstip(t) = B \wedge \forall s \ \neg at(s, t+1)$$

...

2 Specify desired property

Does property P always hold?

Lots of progress on analyzing functional correctness

Existing work focuses on functional correctness

1 Create a mathematical model of the network

$\forall t \ \ (dstip(t) = A \wedge at(s_1, t)) \rightarrow at(s_2, t+1)$

$\forall t \ \ dstip(t) = B \wedge \forall s \ \ \neg at(s, t+1)$

...

2 Specify desired property

Does property P always hold?

Lots of progress on analyzing functional correctness

- Is A reachable from B?
- Are there cyclic zone dependencies in DNS configurations?
- Is VLAN X traffic isolated from VLAN Y?
- ...

Existing work focuses on functional correctness

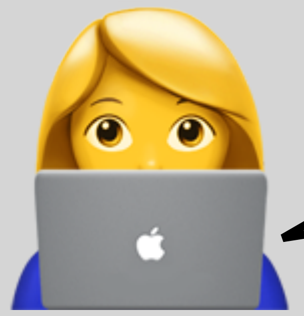
1 Create a mathematical model of the network

$\forall t \ (dstip(t) = A \wedge at(s_1, t)) \rightarrow at(s_2, t+1)$

$\forall t \ dstip(t) = B \wedge \forall s \ \neg at(s, t+1)$

...

2 Specify desired property



Does property P always hold?

Lots of progress on analyzing functional correctness

- Is A reachable from B?
- Are there cyclic zone dependencies in DNS configurations?
- Is VLAN X traffic isolated from VLAN Y?
- ...

But, what about performance?

Existing work focuses on functional correctness

1 Create a mathematical model of the network

$\forall t \ (dstip(t) = A \land at(s_1, t)) \rightarrow at(s_2, t+1)$

$\forall t \ dstip(t) = B \land \forall s \ \neg at(s, t+1)$

...

2 Specify desired property

Does property P always hold?

Lots of progress on analyzing functional correctness

- Is A reachable from B?
- Are there cyclic zone dependencies in DNS configurations?
- Is VLAN X traffic isolated from VLAN Y?
- ...

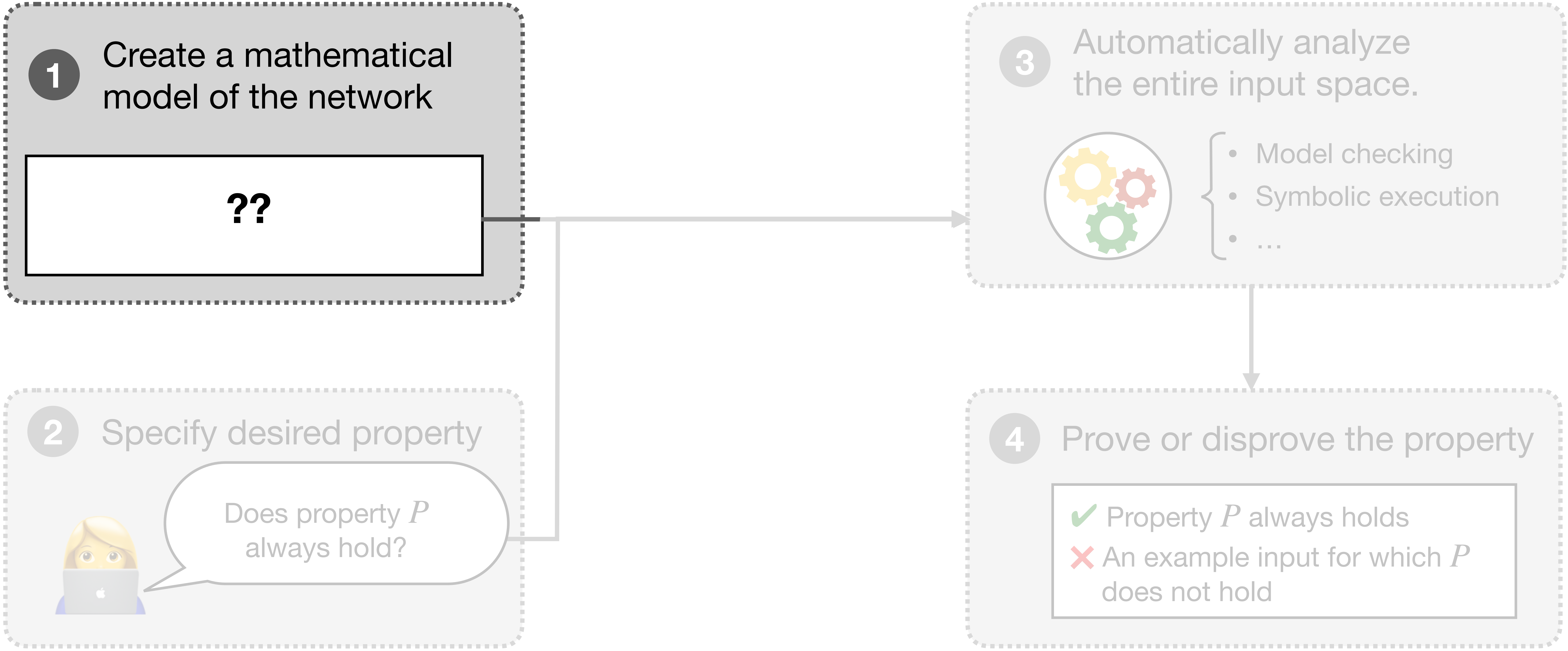
But, what about performance?

- Can flow A's throughput drop below R?
- Can packets in traffic class B experience latency > L?
- Can flow X get a much larger share of the bandwidth than Y?
- ...

This work:

Using formal methods to analyze performance properties

Finding the “right” model



Finding the “right” model

1 Create a mathematical model of the network

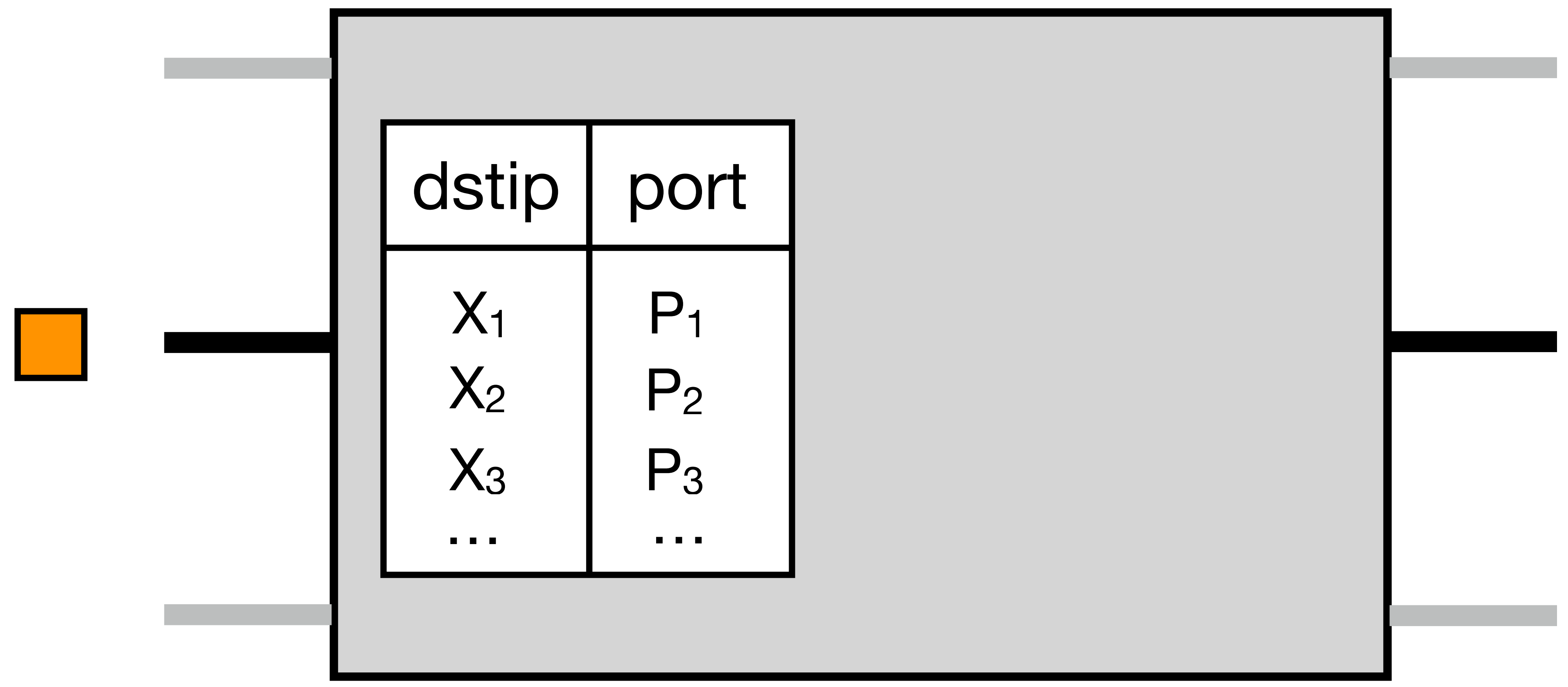
??

2 Specify desired property

Does property P always hold?

Extensively explored for packet forwarding.

A Switch



Finding the “right” model

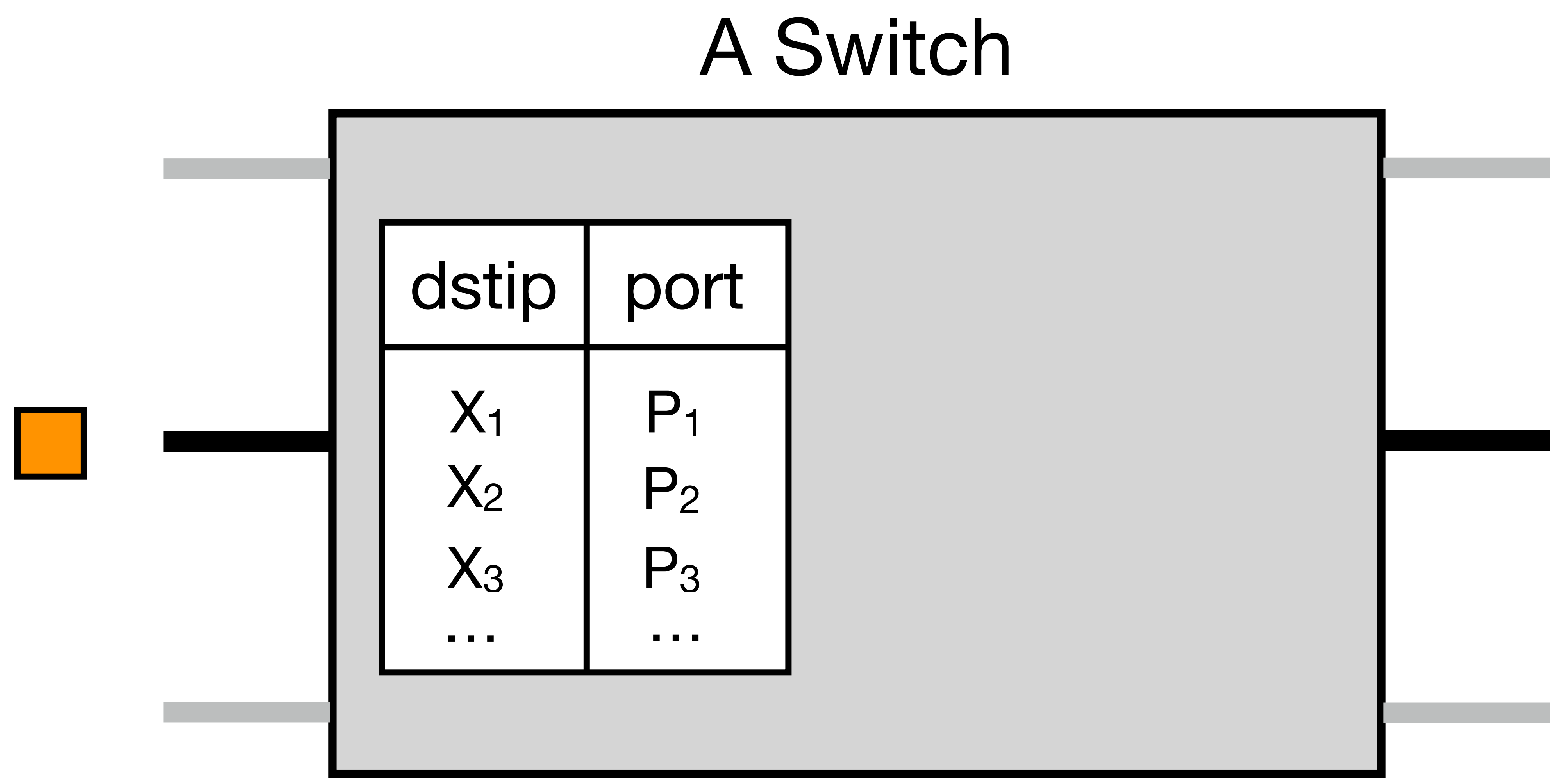
1 Create a mathematical model of the network

??

2 Specify desired property

Does property P always hold?

For performance analysis, we need more than just forwarding



Finding the “right” model

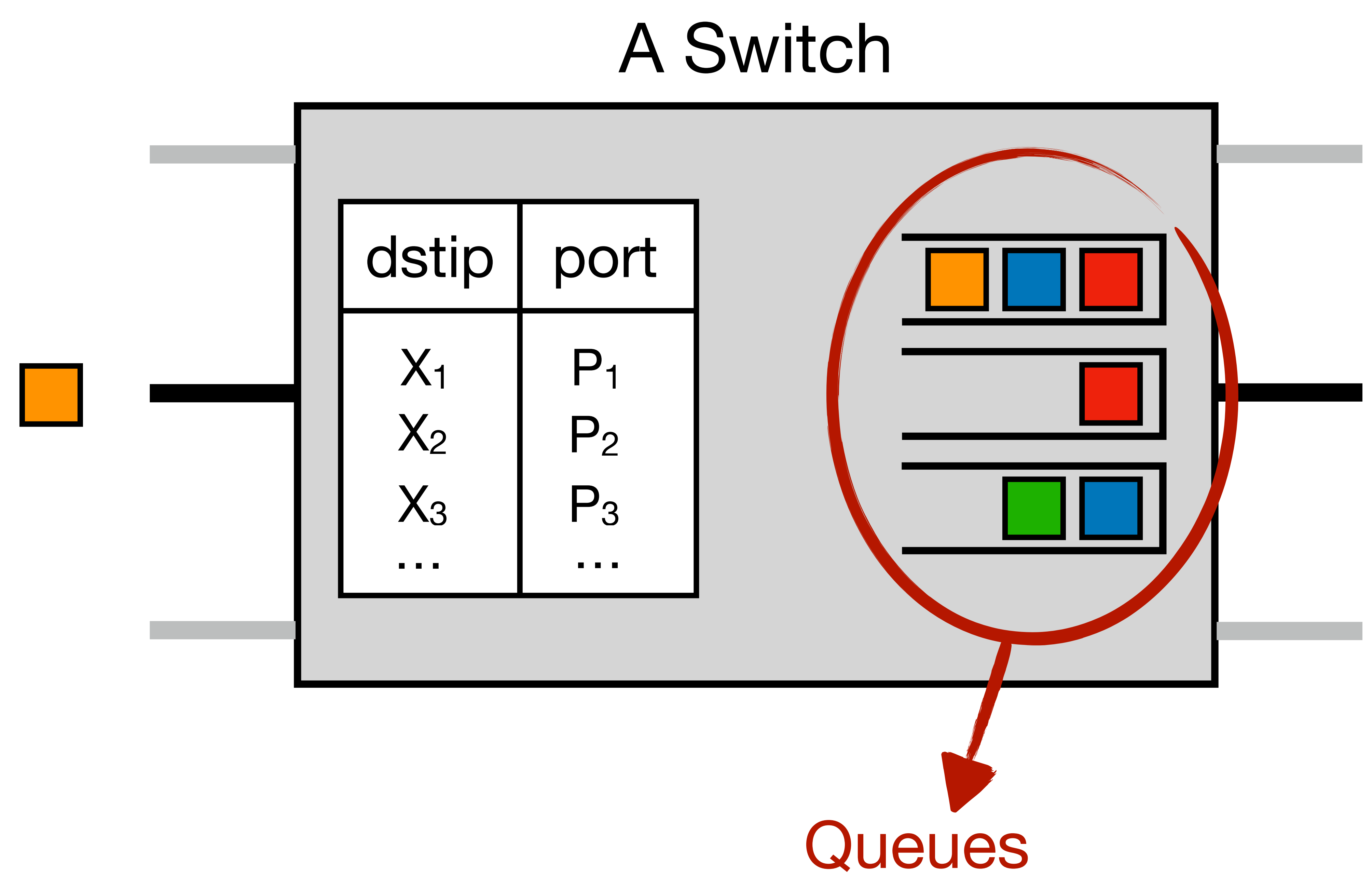
1 Create a mathematical model of the network

??

2 Specify desired property

Does property P always hold?

For performance analysis, we need more than just forwarding



Finding the “right” model

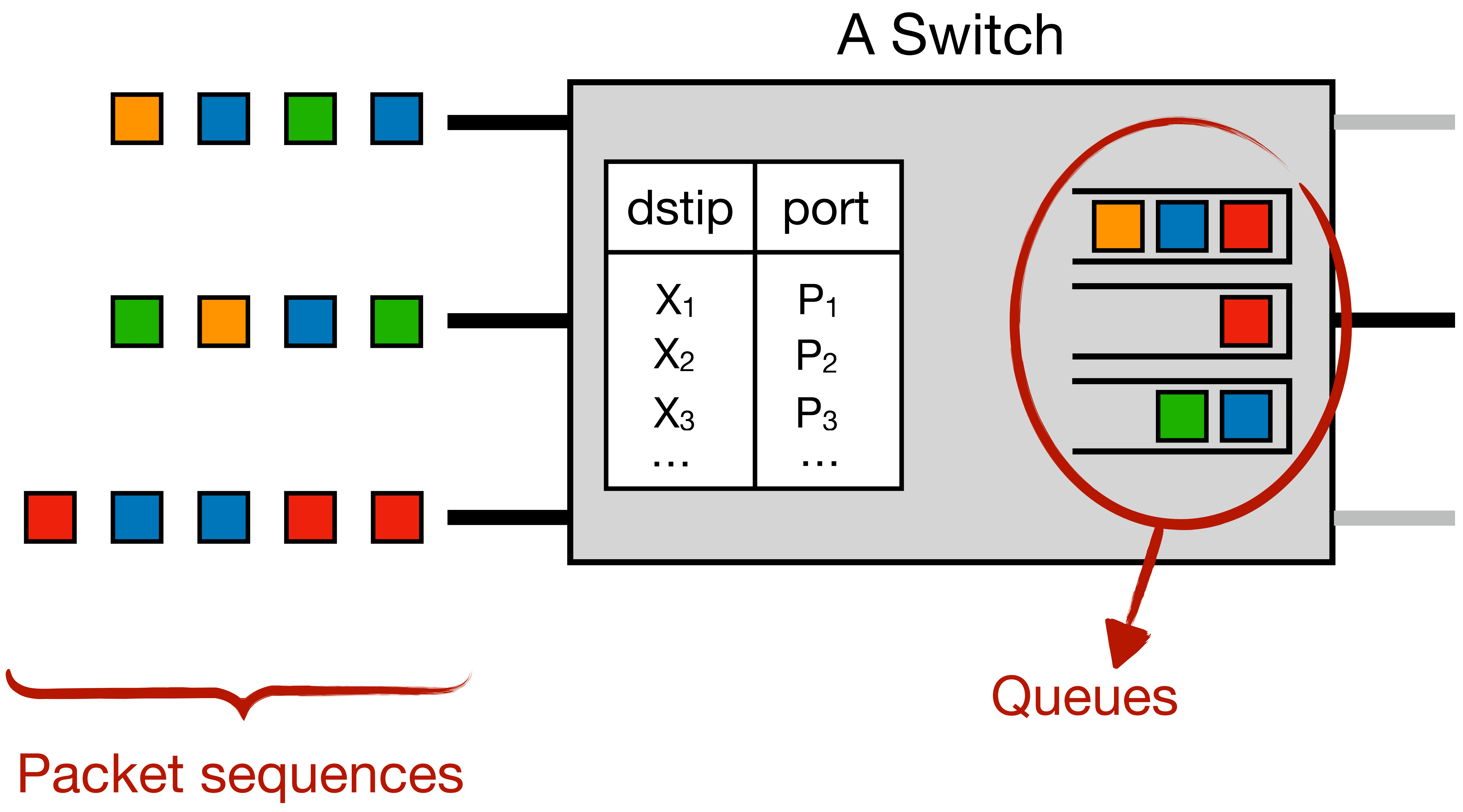
1 Create a mathematical model of the network

??

2 Specify desired property

Does property P always hold?

For performance analysis, we need more than just forwarding



Finding the "right" model

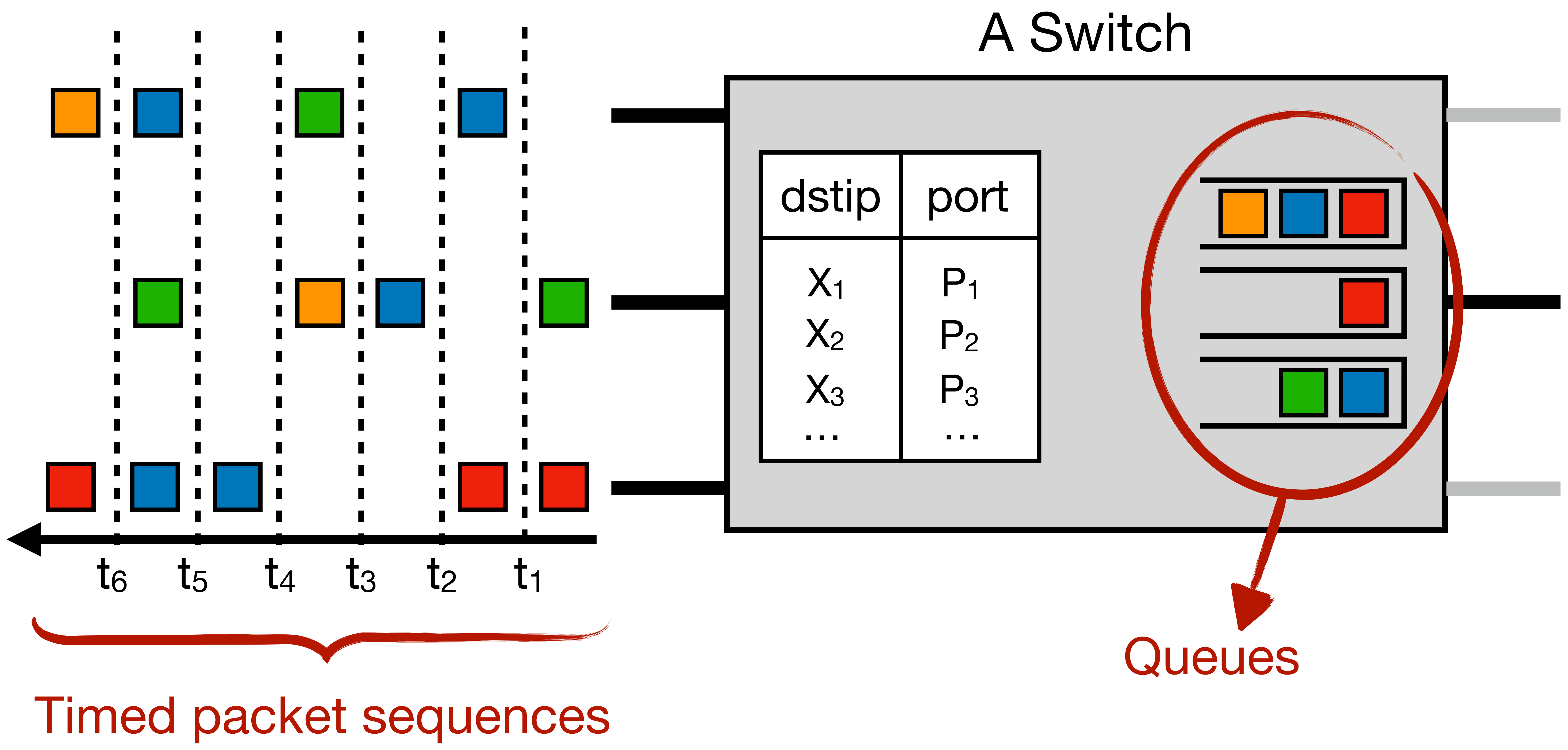
1 Create a mathematical model of the network

??

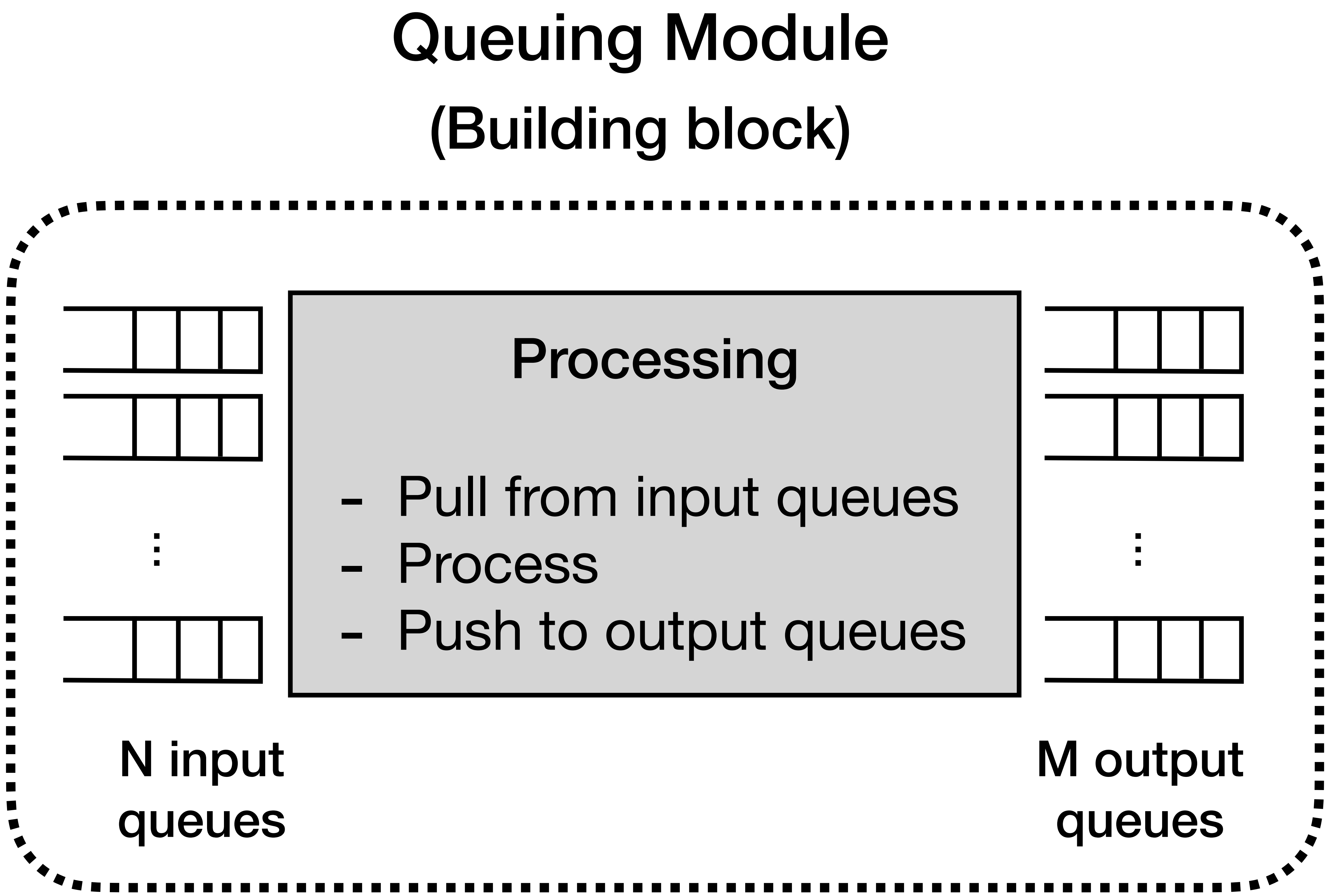
2 Specify desired property

Does property P always hold?

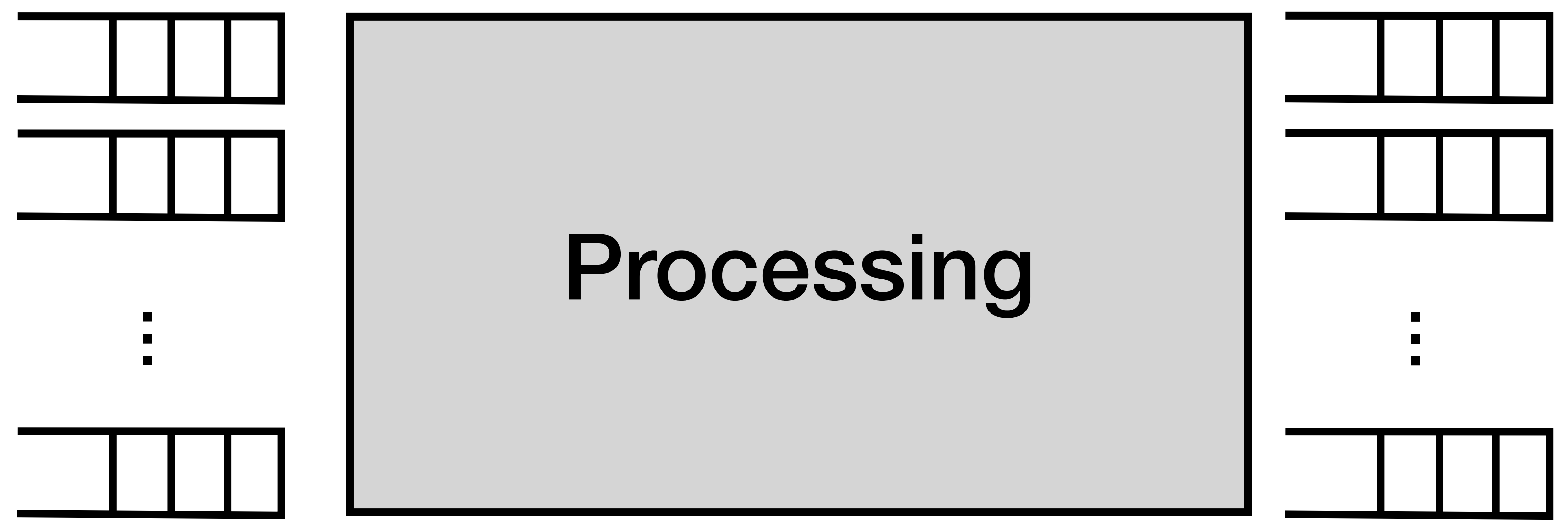
For performance analysis, we need more than just forwarding



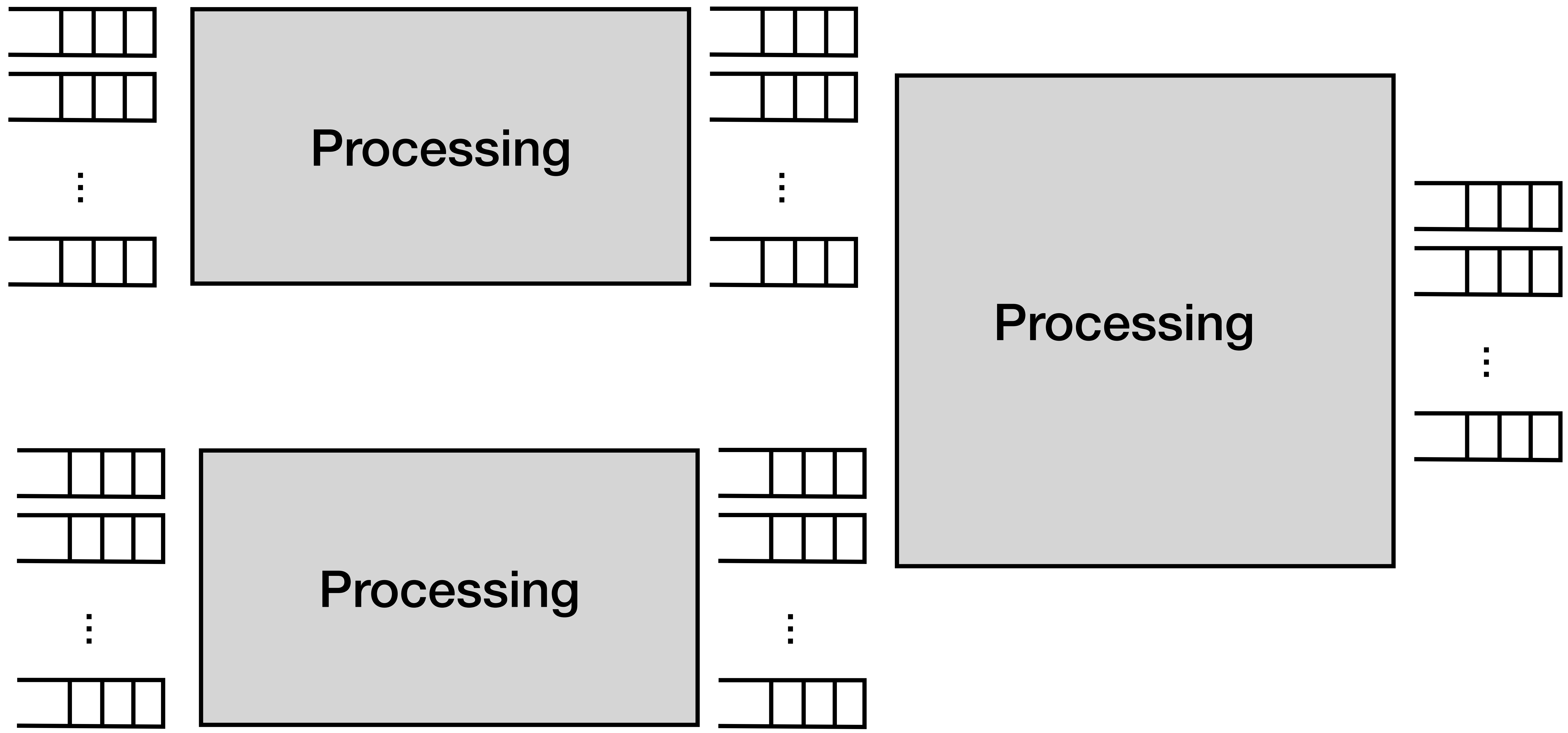
Our model: Composition of “queuing modules”



Our model: Composition of “queuing modules”



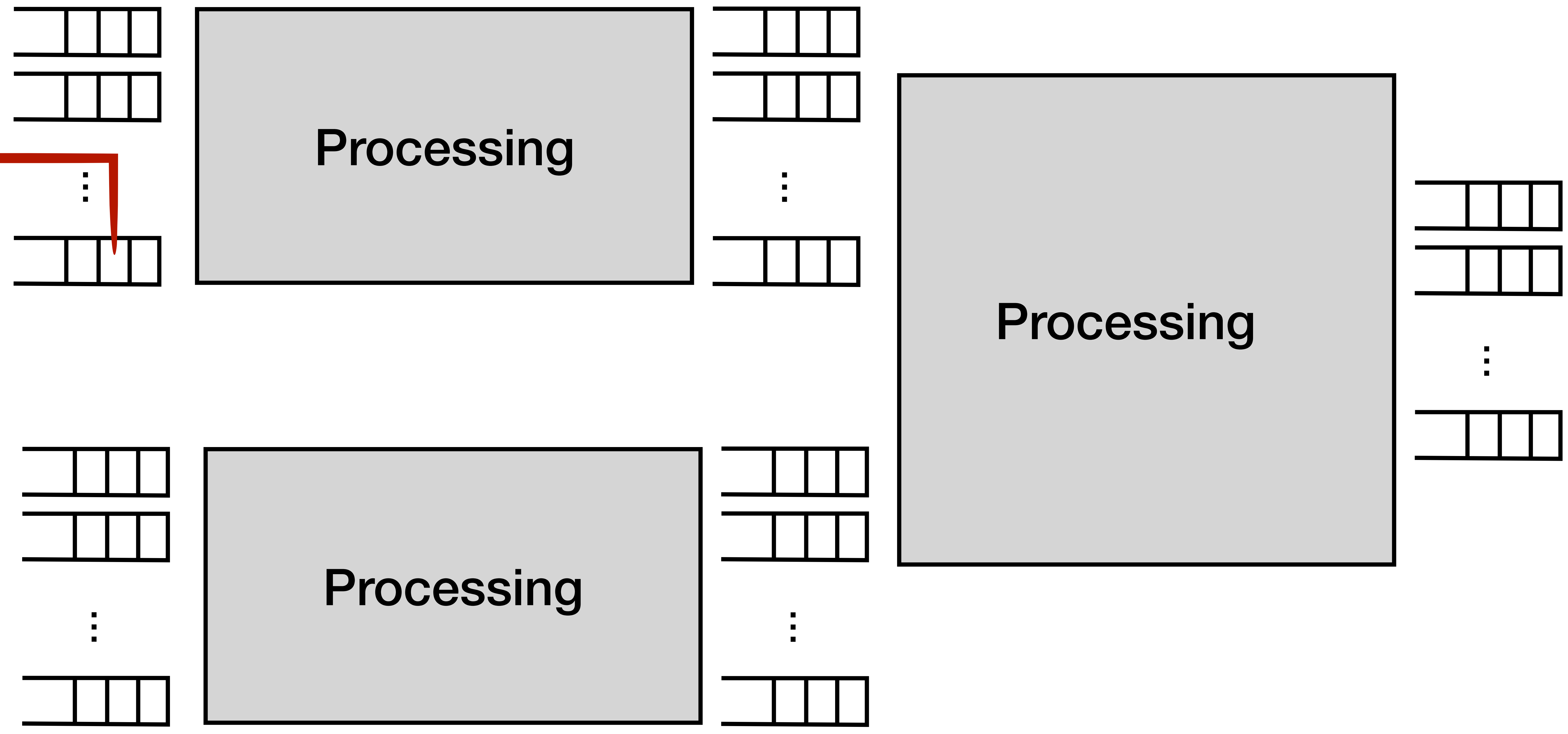
Our model: Composition of “queuing modules”



Our model: Composition of “queuing modules”

Queues are modeled explicitly:

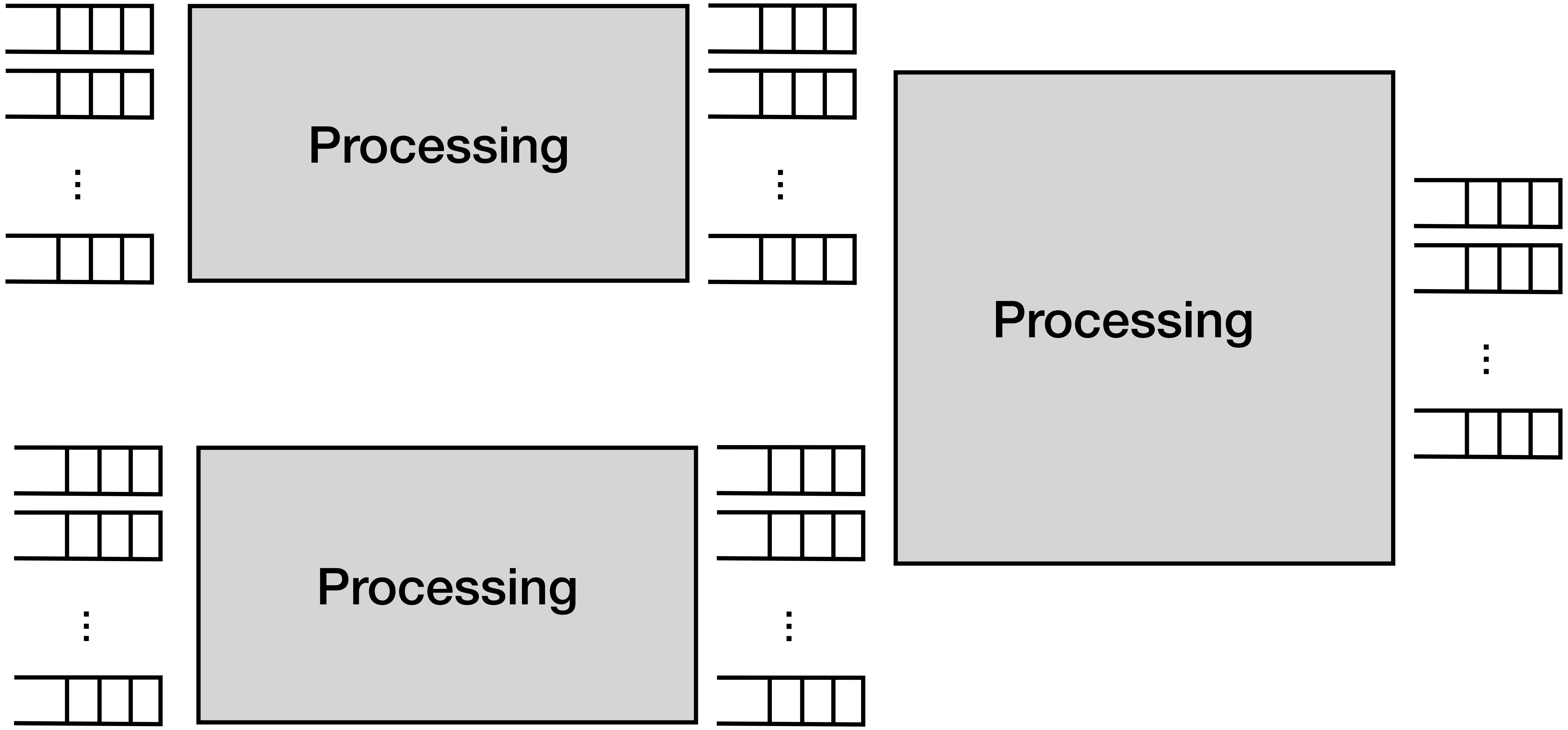
$q.elem[i][t]$ → i-th packet in the queue at time t



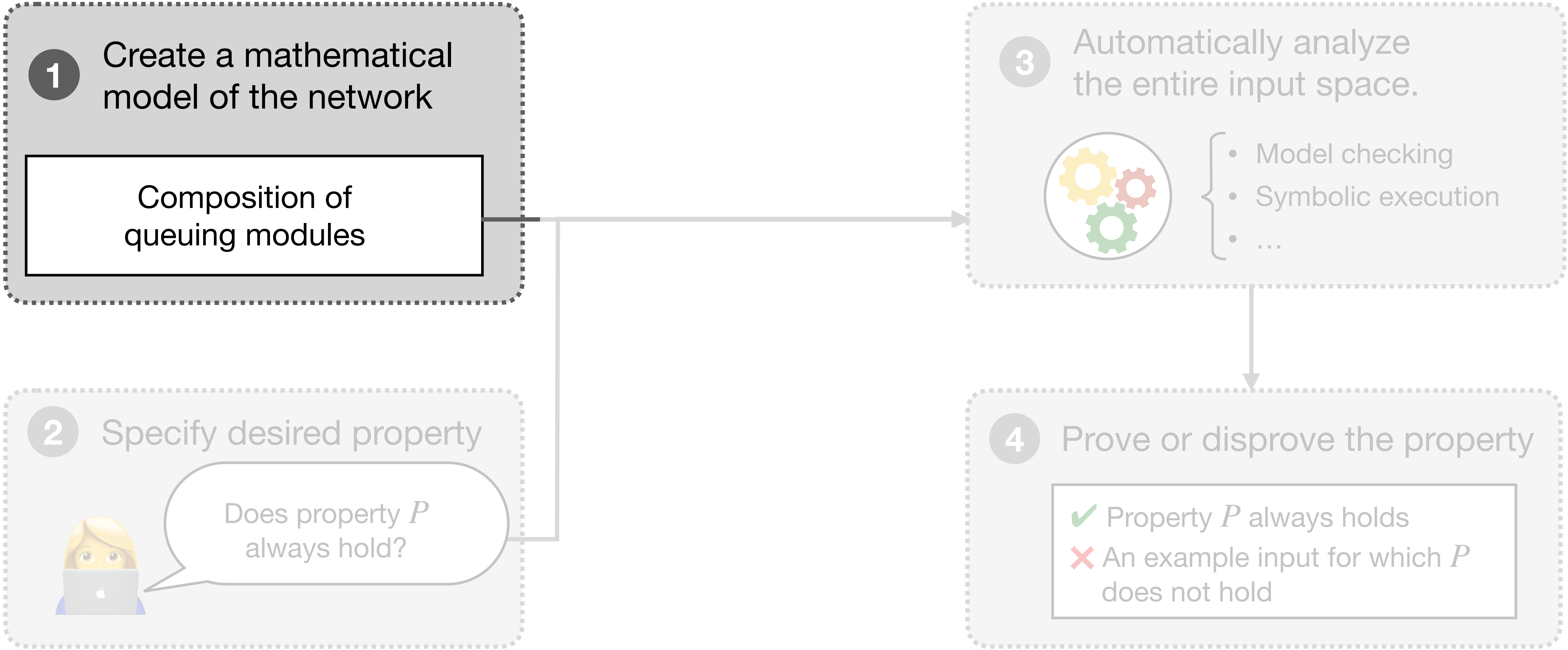
Our model: Composition of “queuing modules”

How do we make it tractable to analyze?

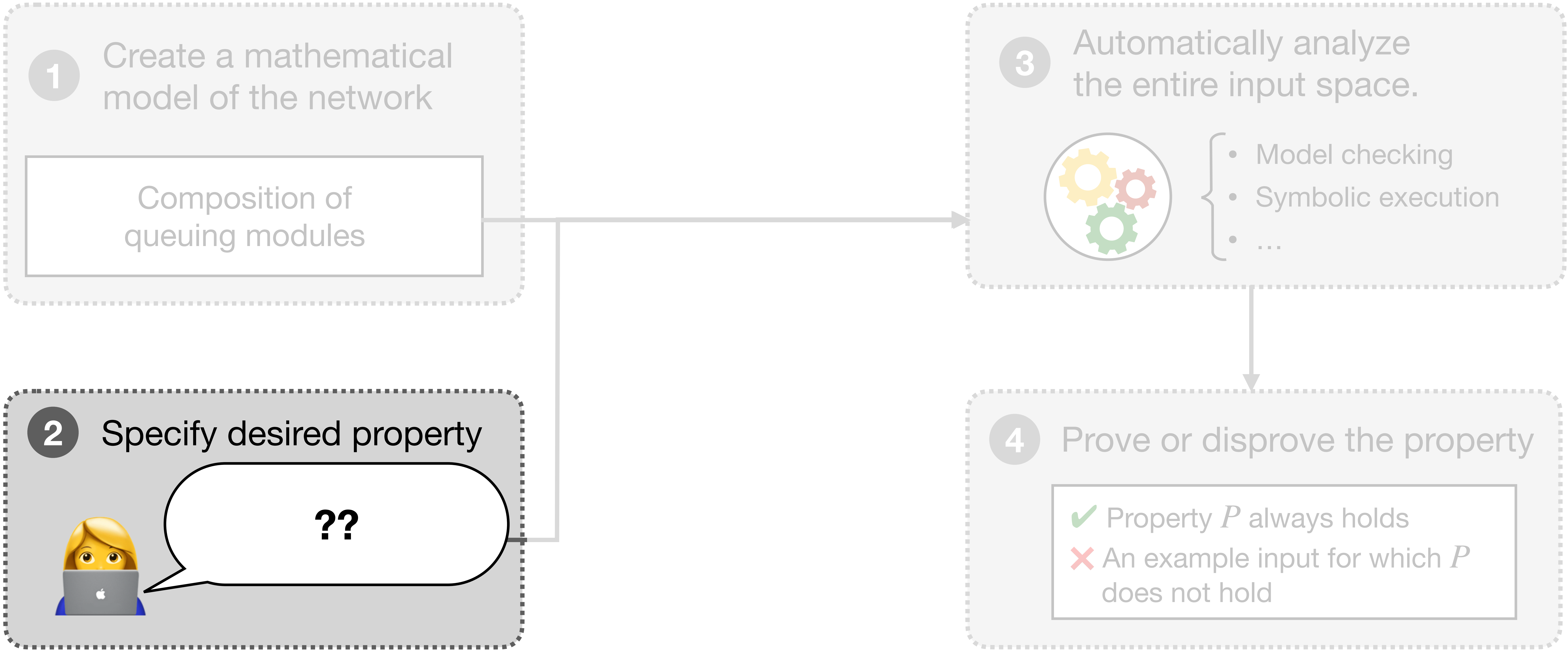
- Abstract time over dequeues
- Bounded time analysis
- Efficient queue encoding
- Optimizing compositions



Finding the “right” model



Specifying performance properties of interest

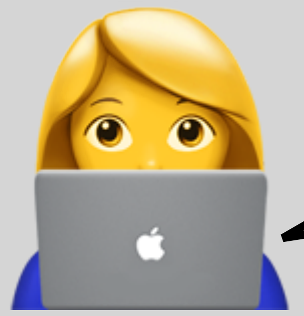


Specifying performance properties of interest

1 Create a mathematical model of the network

Composition of queuing modules

2 Specify desired property



??

Automatically analyze

- **Pre- or user-defined metrics over queues**
  - Queue size: $queue\_size(q, t)$
  - Number of enqueued packets: $total\_packets(q, t)$
  - Arrival inter-packet gap: $inter\_packet\_gap(q, t)$
  - <insert your metric of interest>

✔ Property P always holds

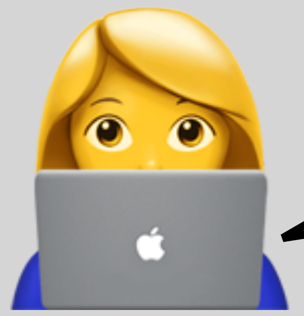
✘ An example input for which P does not hold

Specifying performance properties of interest

1 Create a mathematical model of the network

Composition of queuing modules

2 Specify desired property



??

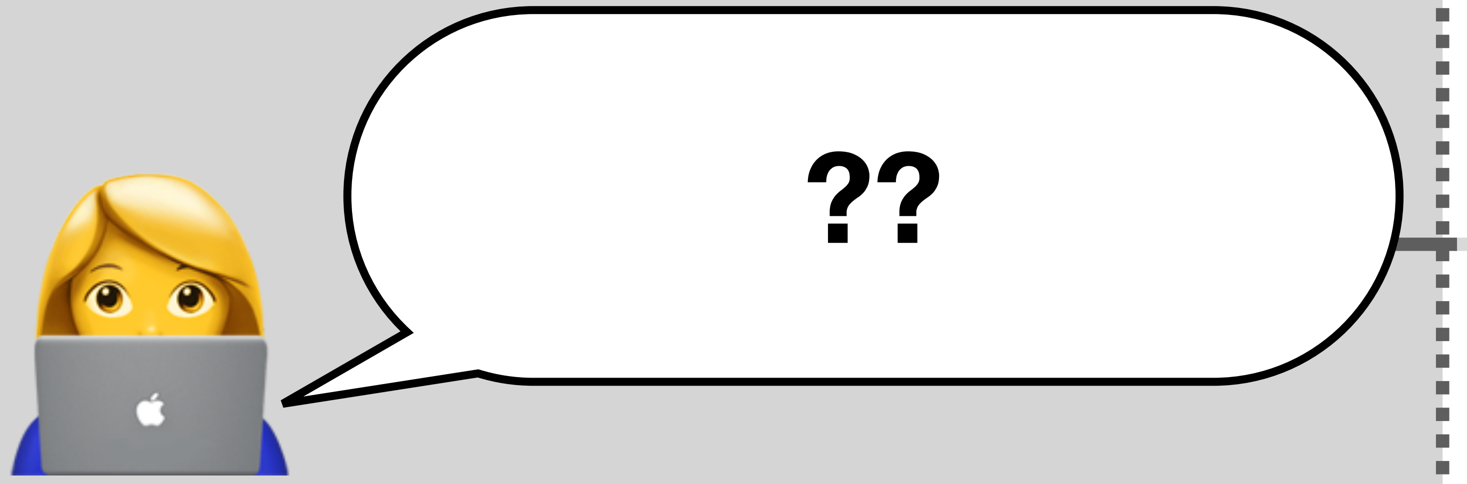
- Properties compare metrics to certain values

Specifying performance properties of interest

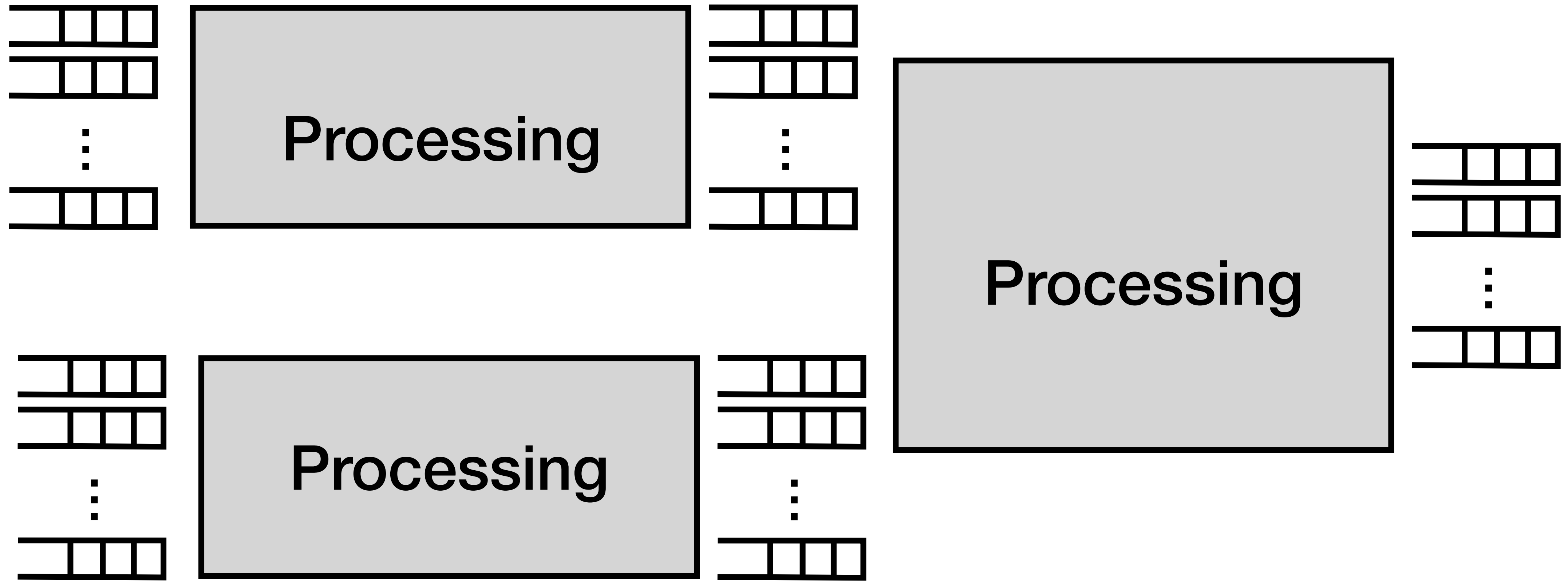
1 Create a mathematical model of the network

Composition of queuing modules

2 Specify desired property



- Properties compare metrics to certain values

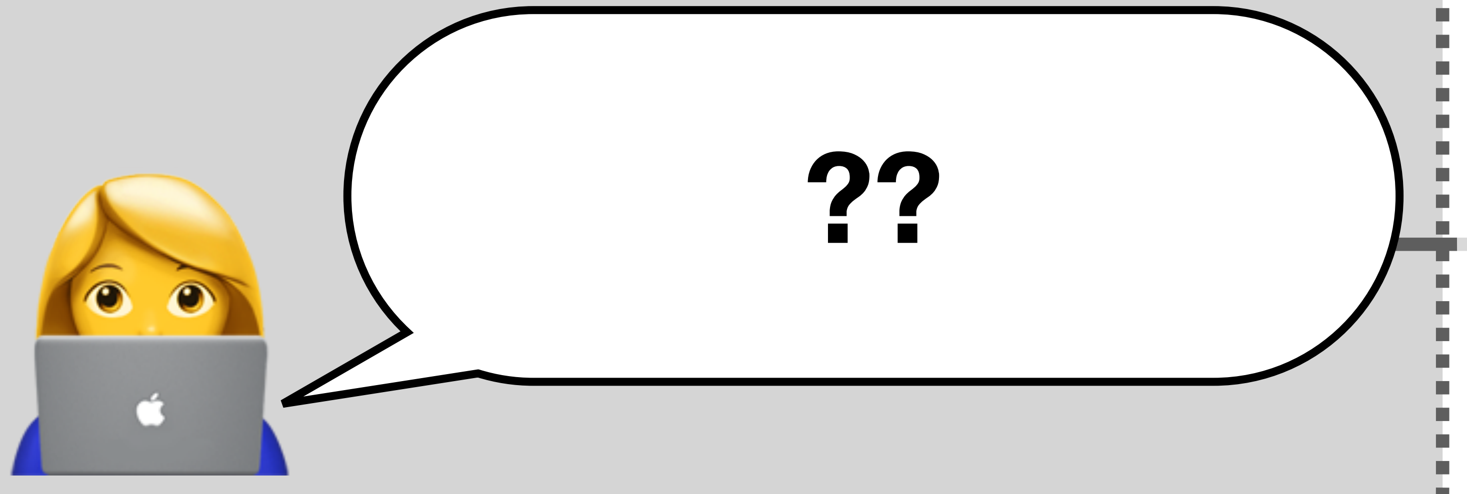


Specifying performance properties of interest

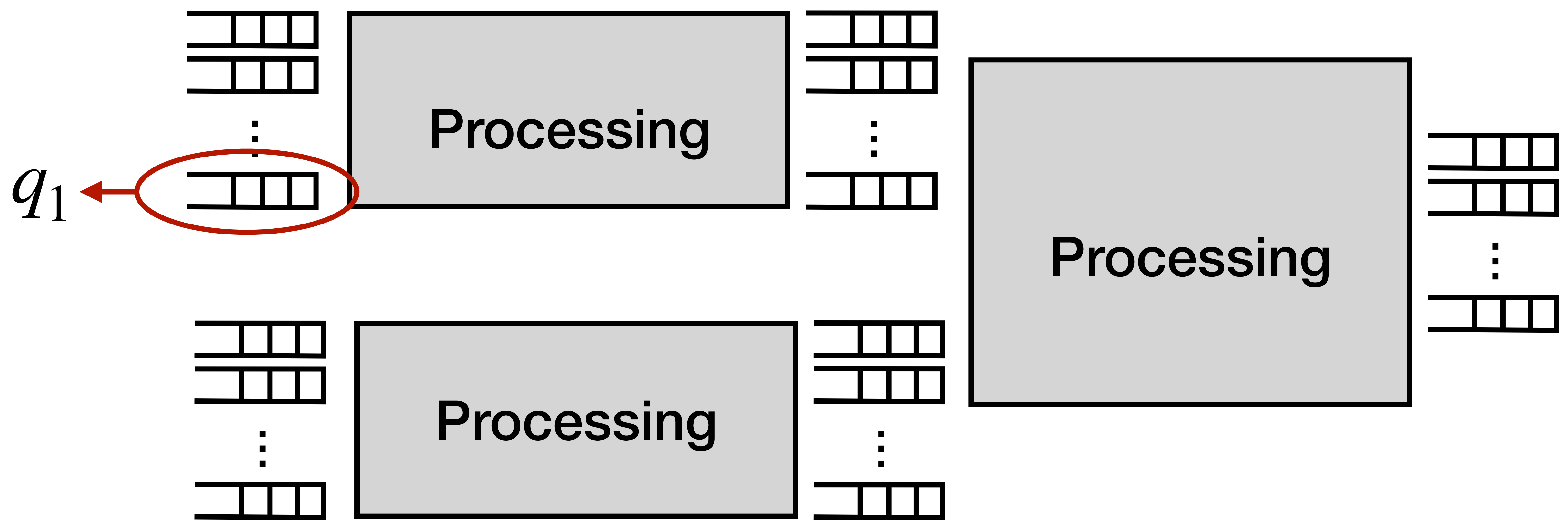
1 Create a mathematical model of the network

Composition of queuing modules

2 Specify desired property



- **Properties compare metrics to certain values**
  - $inter\_packet\_gap(q_1, t_1) \geq 10$

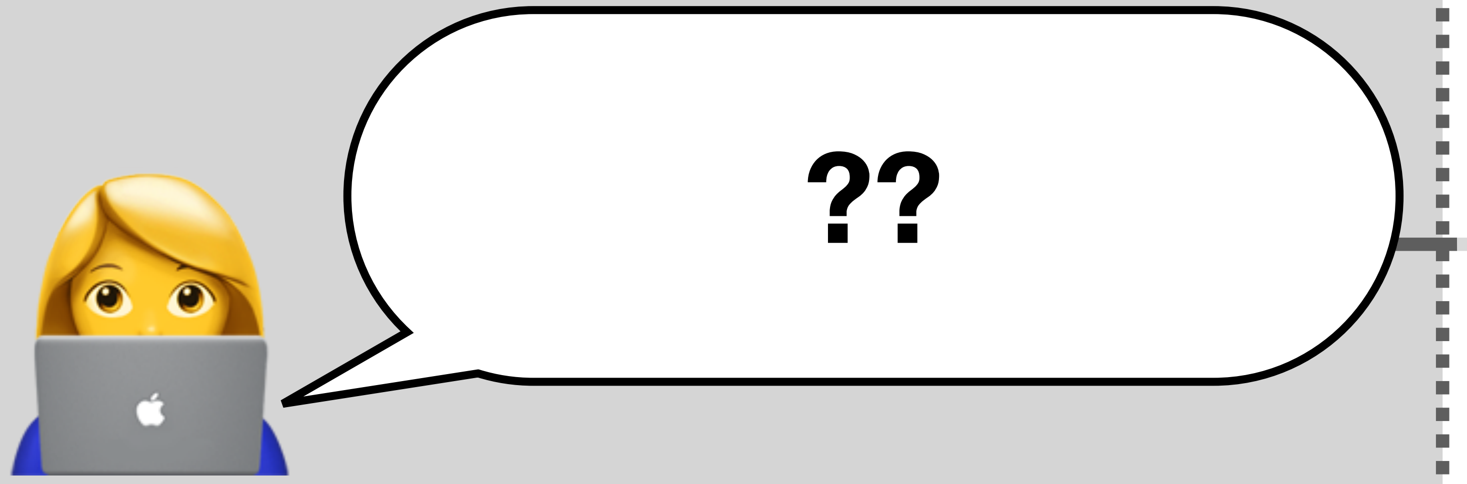


Specifying performance properties of interest

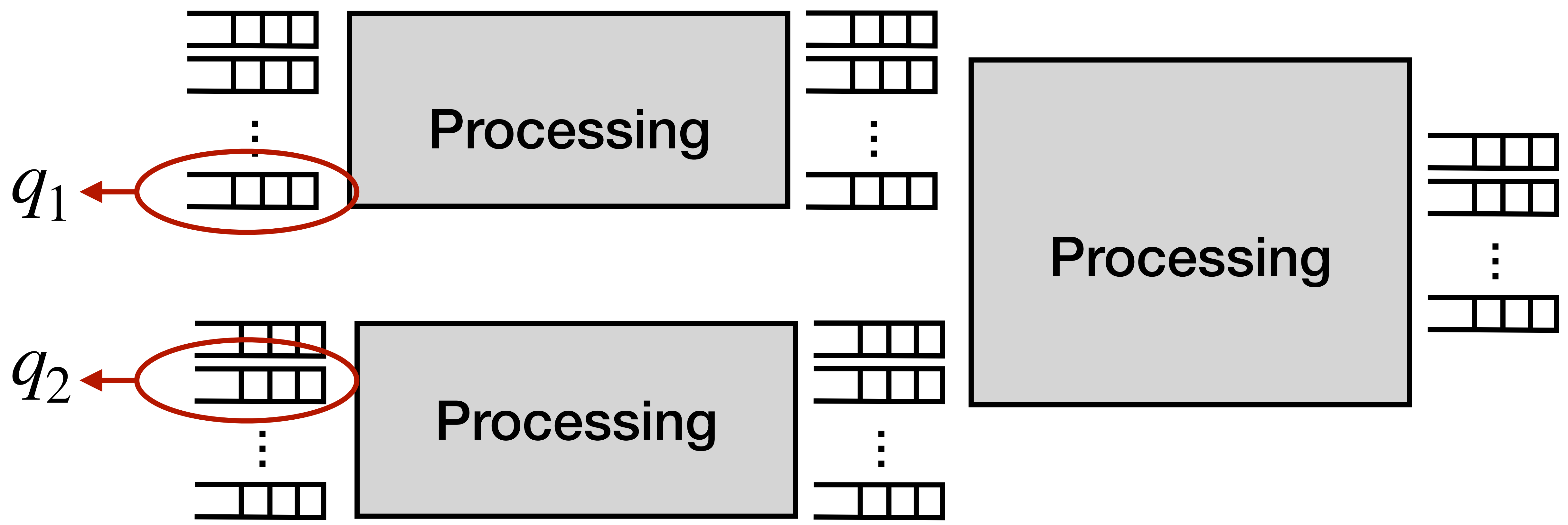
1 Create a mathematical model of the network

Composition of queuing modules

2 Specify desired property



- **Properties compare metrics to certain values**
  - $inter\_packet\_gap(q_1, t_1) \geq 10$
  - $queue\_size(q_1, t_5) \leq queue\_size(q_2, t_6)$



Specifying performance properties of interest

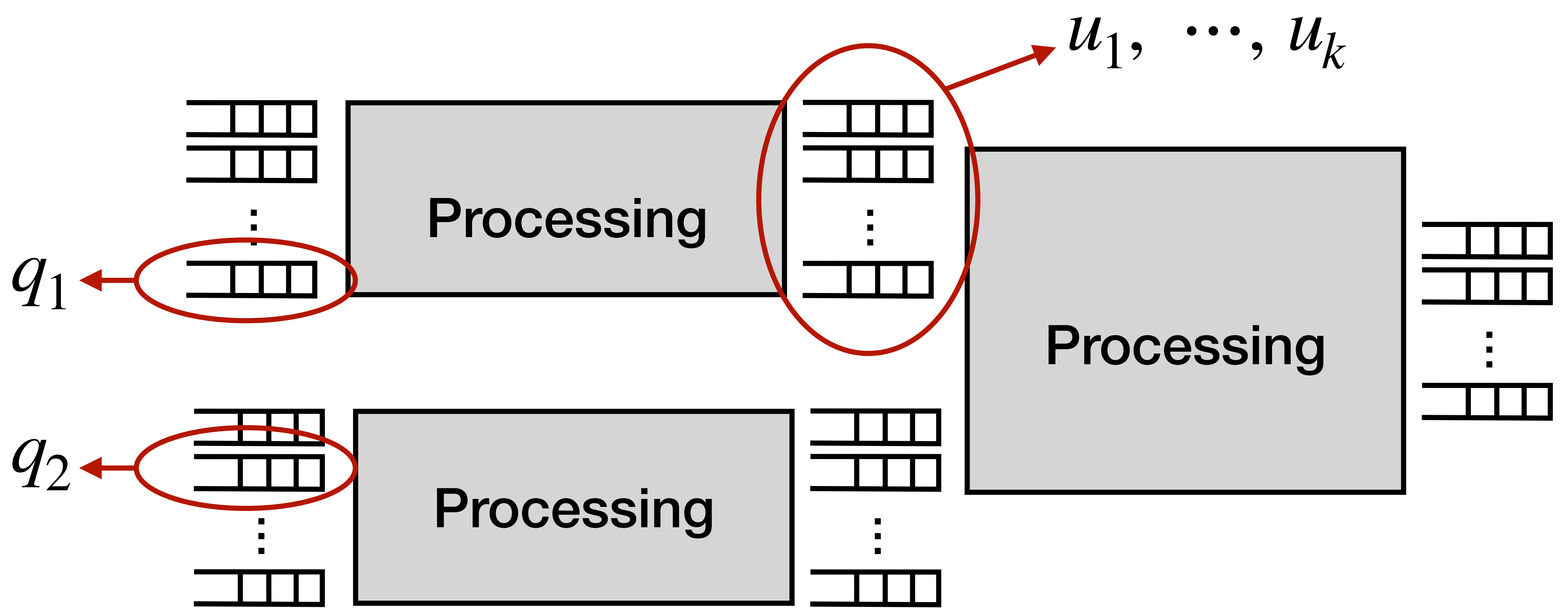
1 Create a mathematical model of the network

Composition of queuing modules

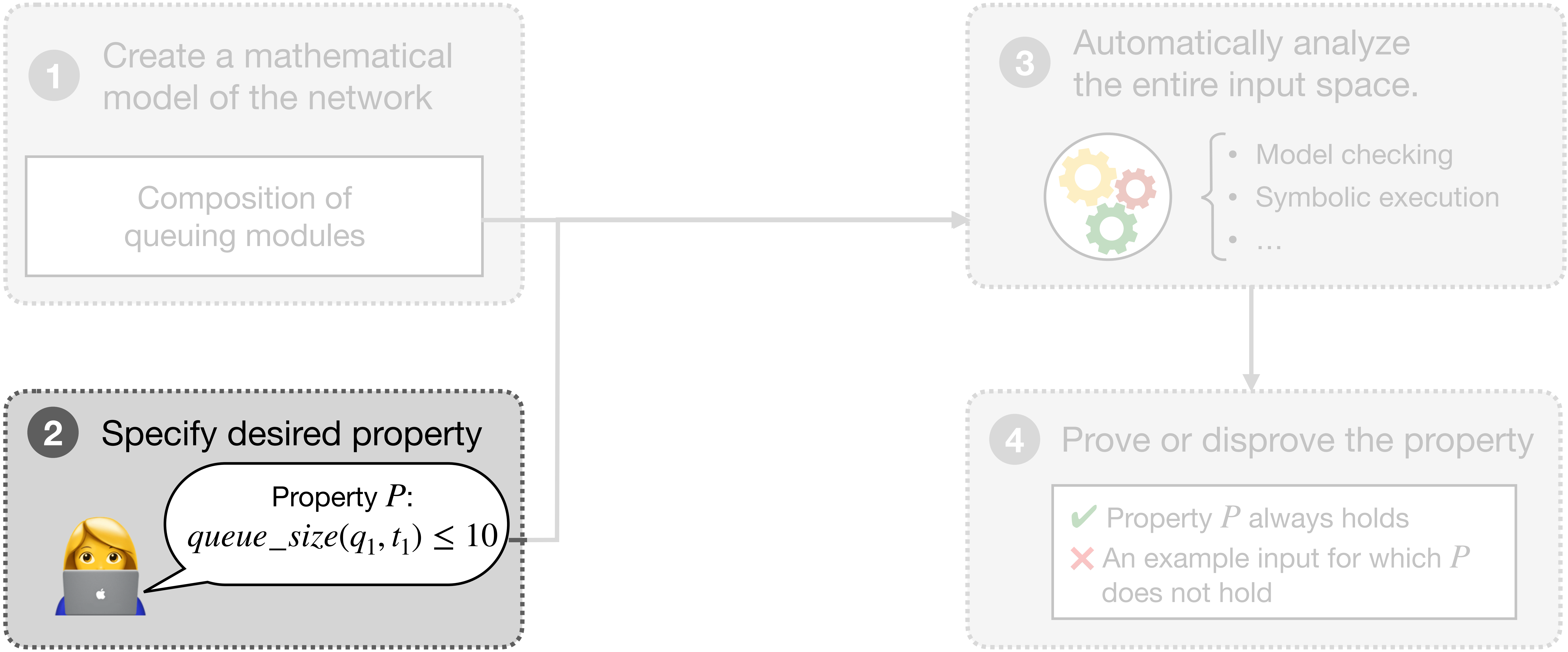
2 Specify desired property

??

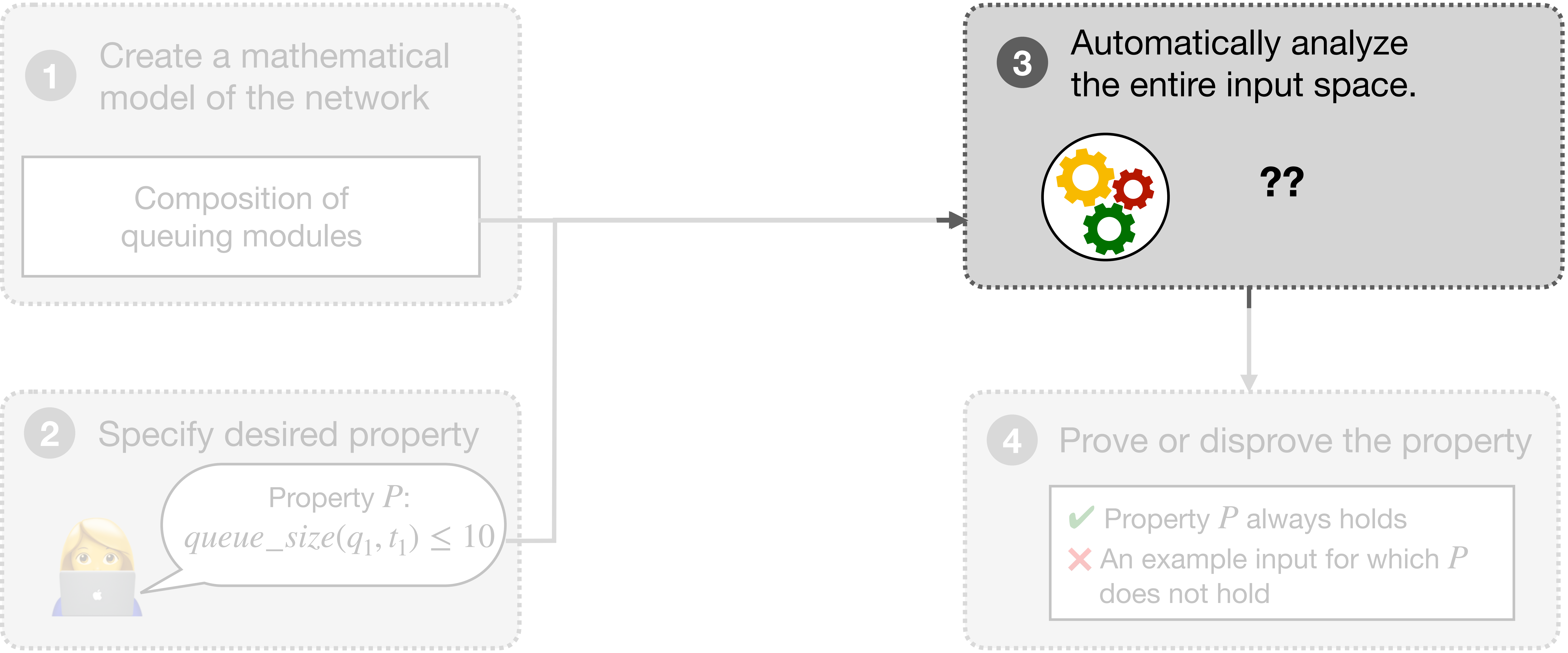
- **Properties compare metrics to certain values**
  - $inter\_packet\_gap(q_1, t_1) \geq 10$
  - $queue\_size(q_1, t_5) \leq queue\_size(q_2, t_6)$
  - $\Sigma_{q \in \{u_1, \cdots, u_k\}} \; total\_packets(q, t_{10}) \geq 20$



Specifying performance properties of interest



Analyzing $model \wedge \neg property$



Analyzing $model \land \neg property$

- $model \land \neg property$ is a quantifier-free SMT formula with integer arithmetic
- We use Z3 to analyze its satisfiability.

3 Automatically analyze the entire input space.

??

2 Specify desired property

Property P:
$queue\_size(q_1, t_1) \leq 10$

4 Prove or disprove the property

✔ Property P always holds
✕ An example input for which P does not hold

Analyzing $model \wedge \neg property$

1 Create a mathematical model of the network

Composition of queuing modules

2 Specify desired property

Property P:
$queue\_size(q_1, t_1) \leq 10$

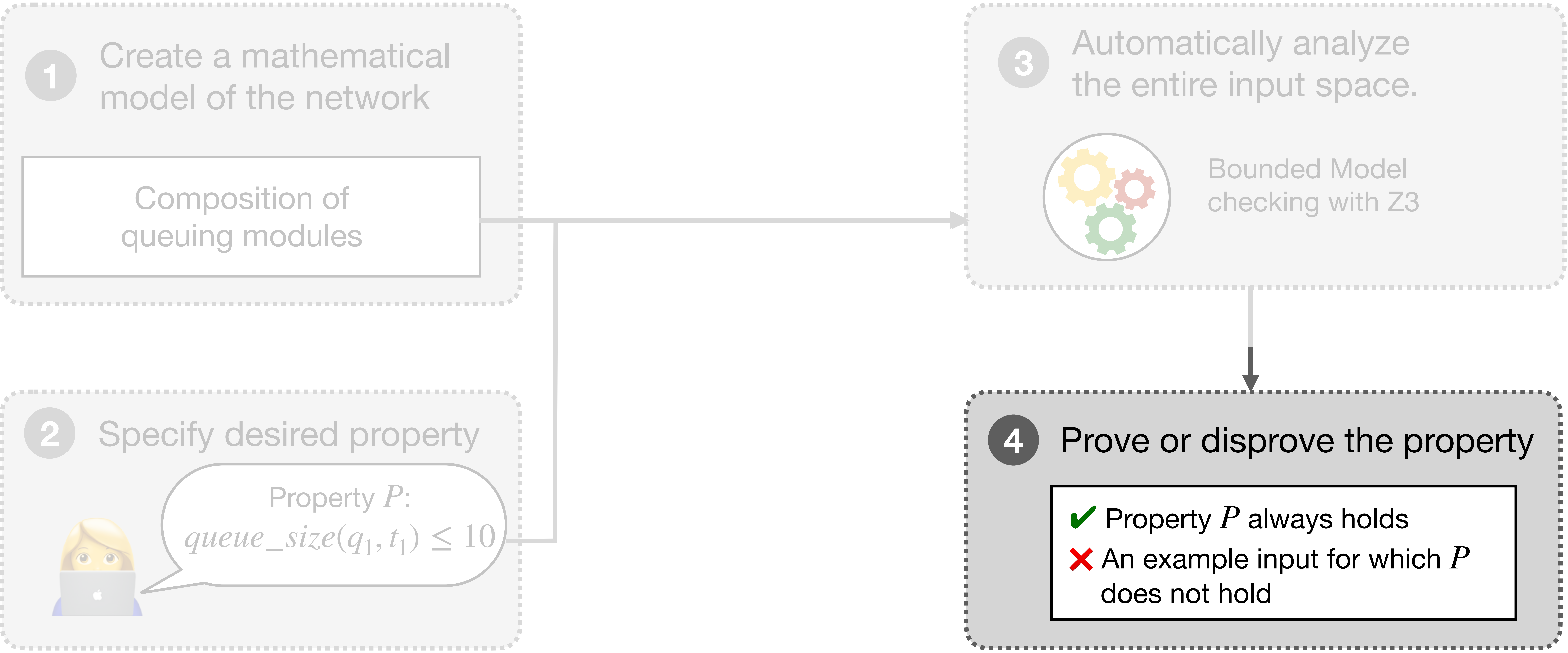
3 Automatically analyze the entire input space.

Bounded Model checking with Z3

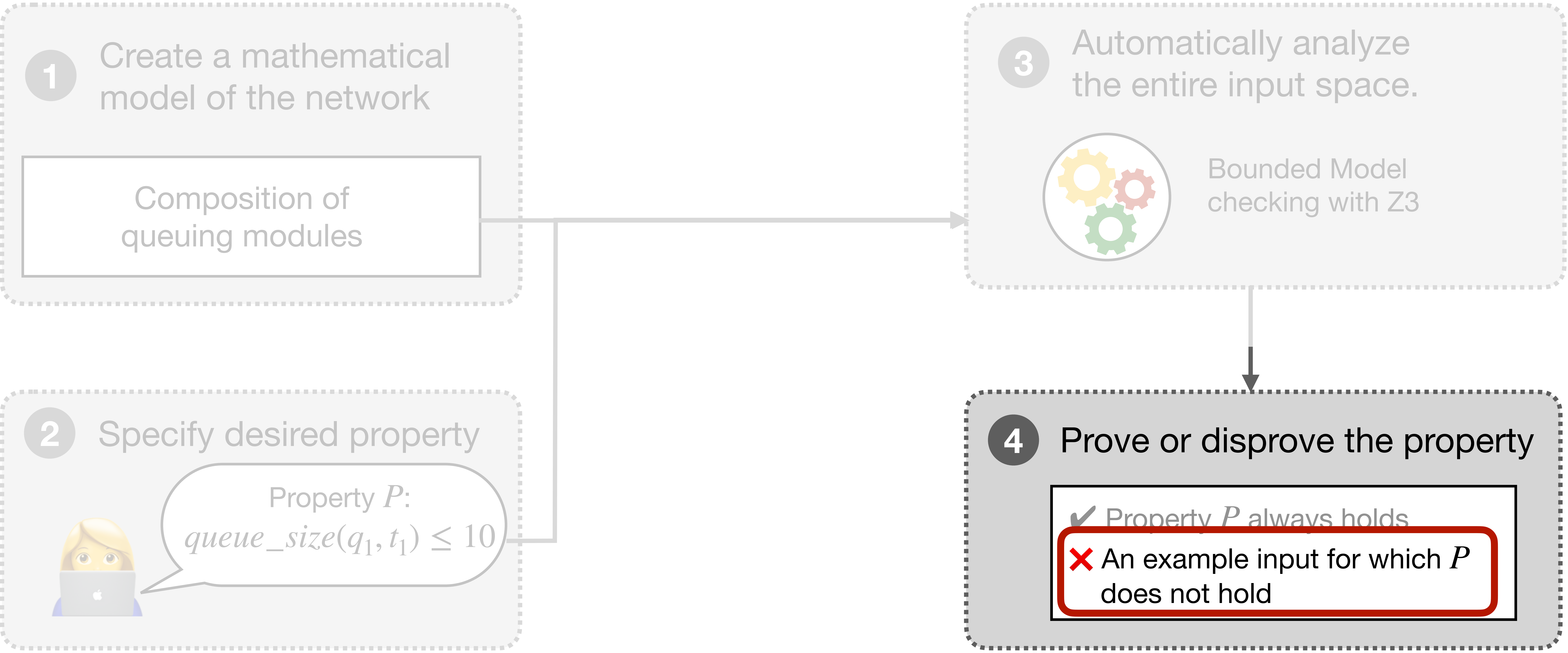
4 Prove or disprove the property

✔ Property P always holds
✘ An example input for which P does not hold

When the property doesn't hold…

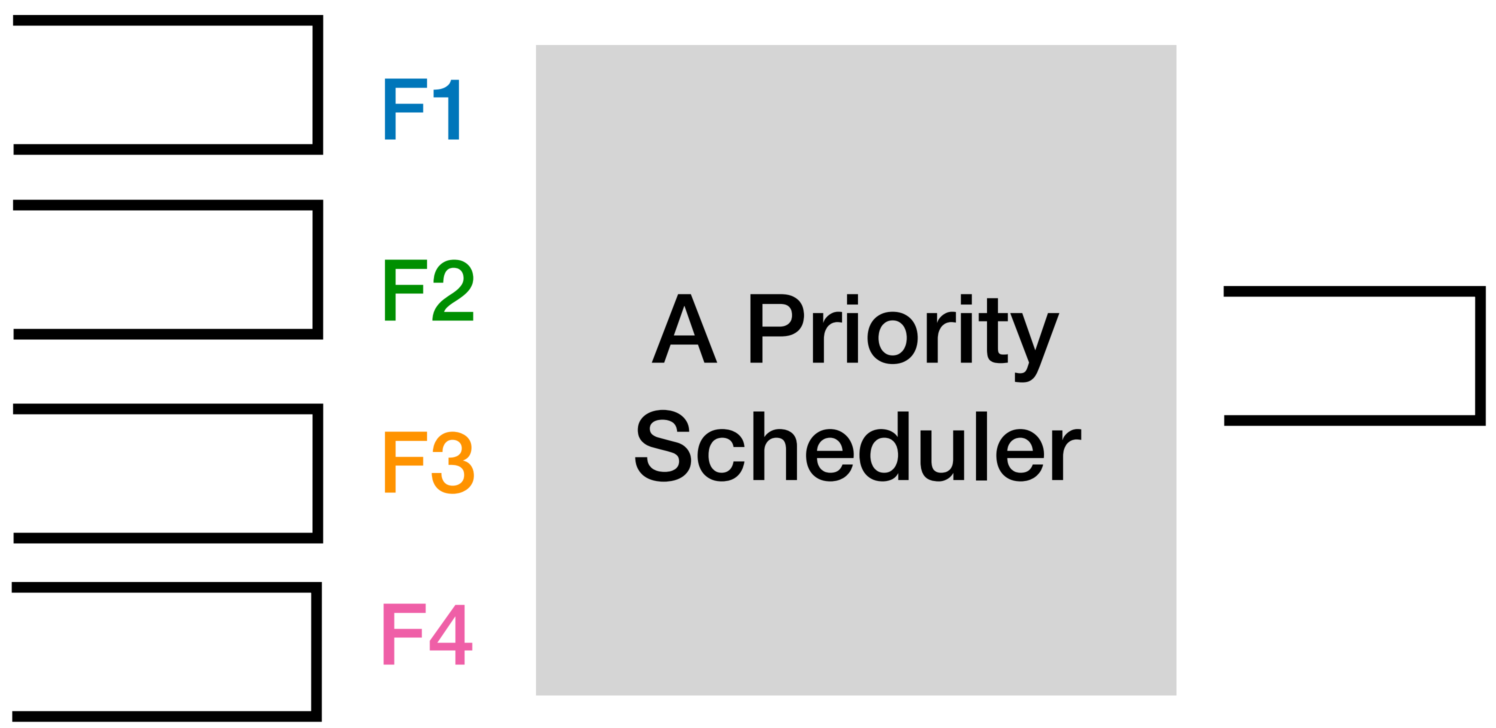


When the property doesn't hold…



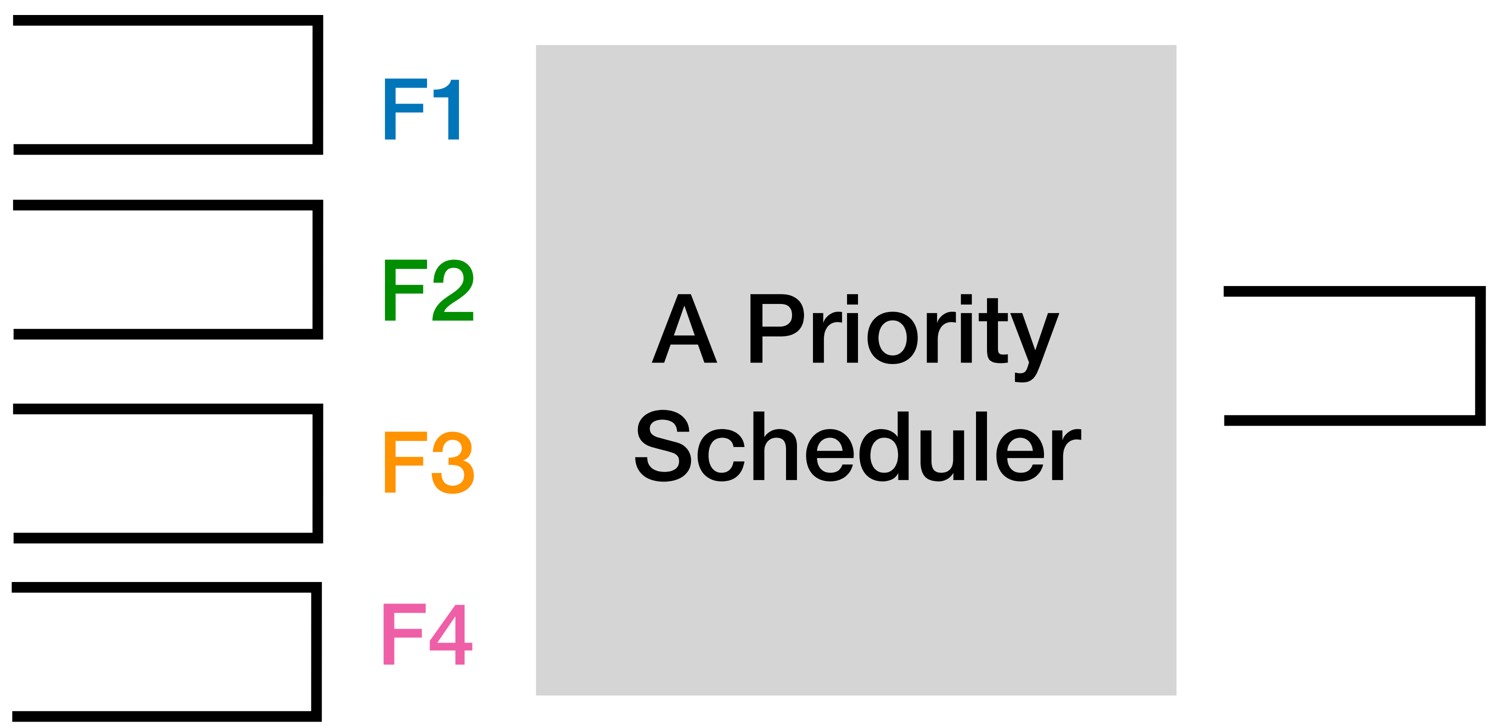
A single trace is not an informative output

A single trace is not an informative output



A single trace is not an informative output

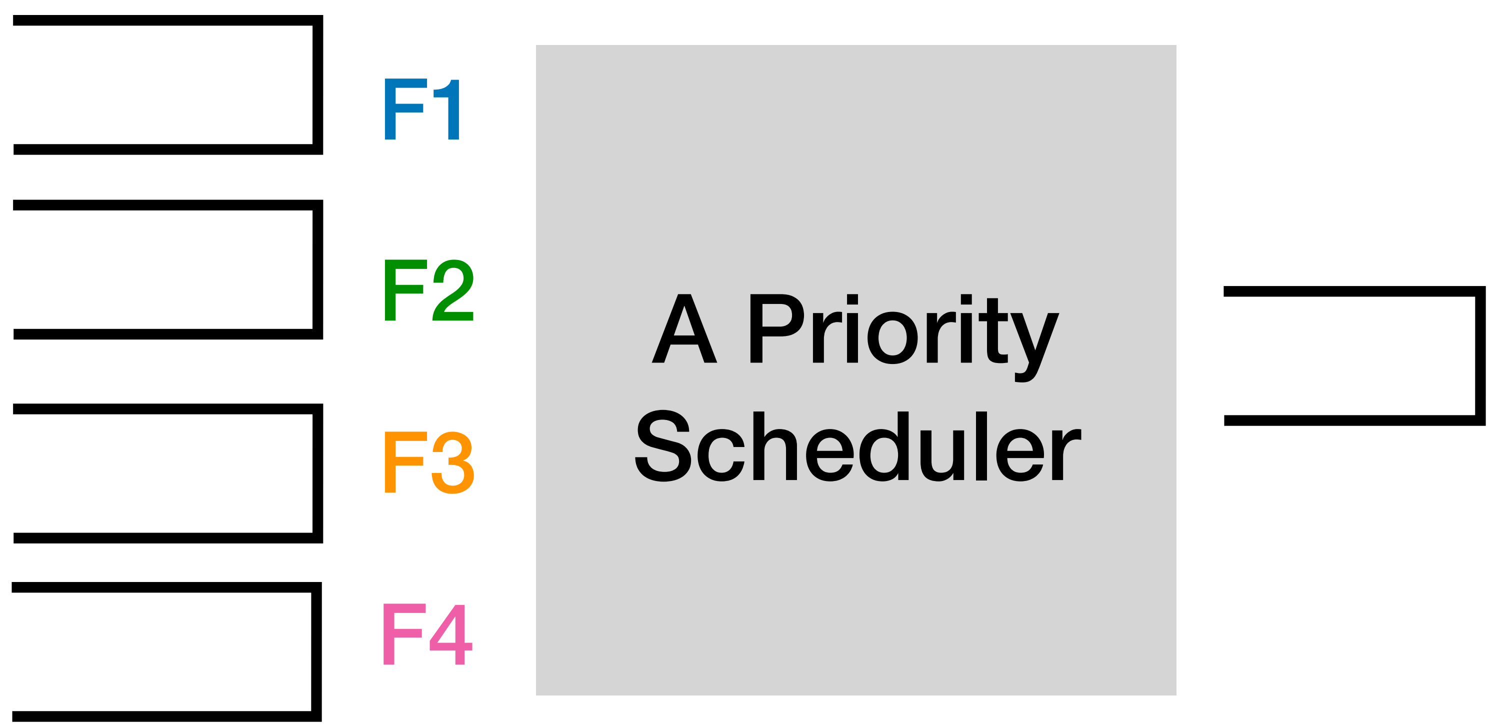
Property: F3 should not be blocked for dequeue (get starved) for X consecutive time steps.



A single trace is not an informative output

Property: F3 should not be blocked for dequeue (get starved) for X consecutive time steps.

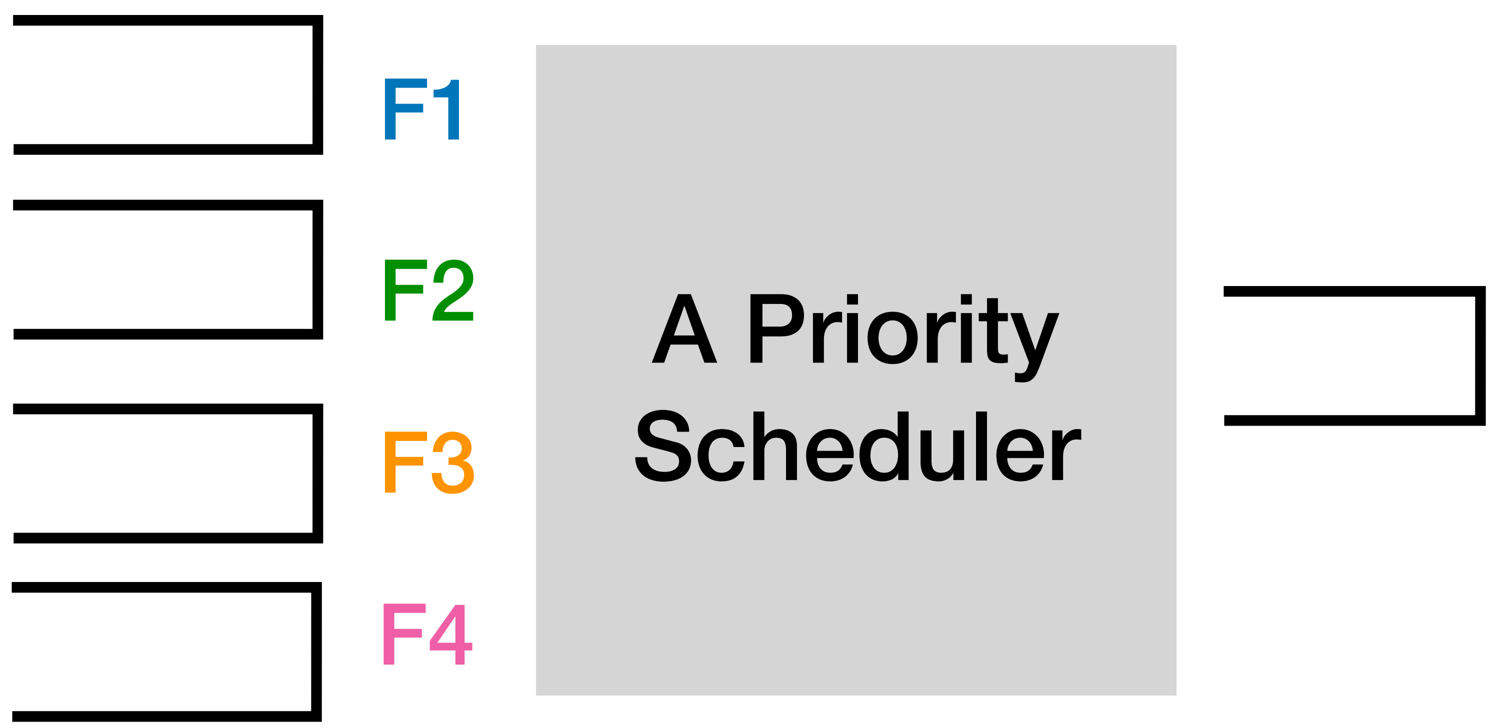
Output: Does not hold.



A single trace is not an informative output

Property: F3 should not be blocked for dequeue (get starved) for X consecutive time steps.

Output: Does not hold.
e.g., for this particular input:

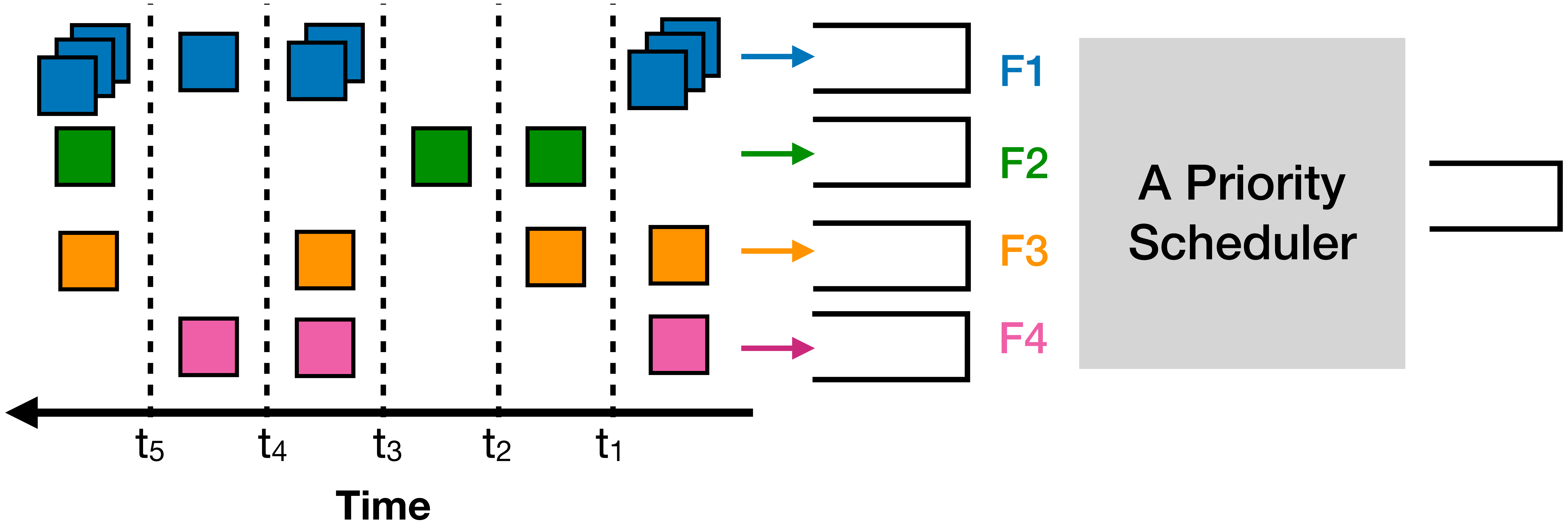


A single trace is not an informative output

- Timed packet sequences
  - Needed in the model
  - Not necessarily useful in the output

d not be blocked for dequeue (get starved)
nsecutive time steps.

Outpu : Does not hold.
e.g., for this particular input:

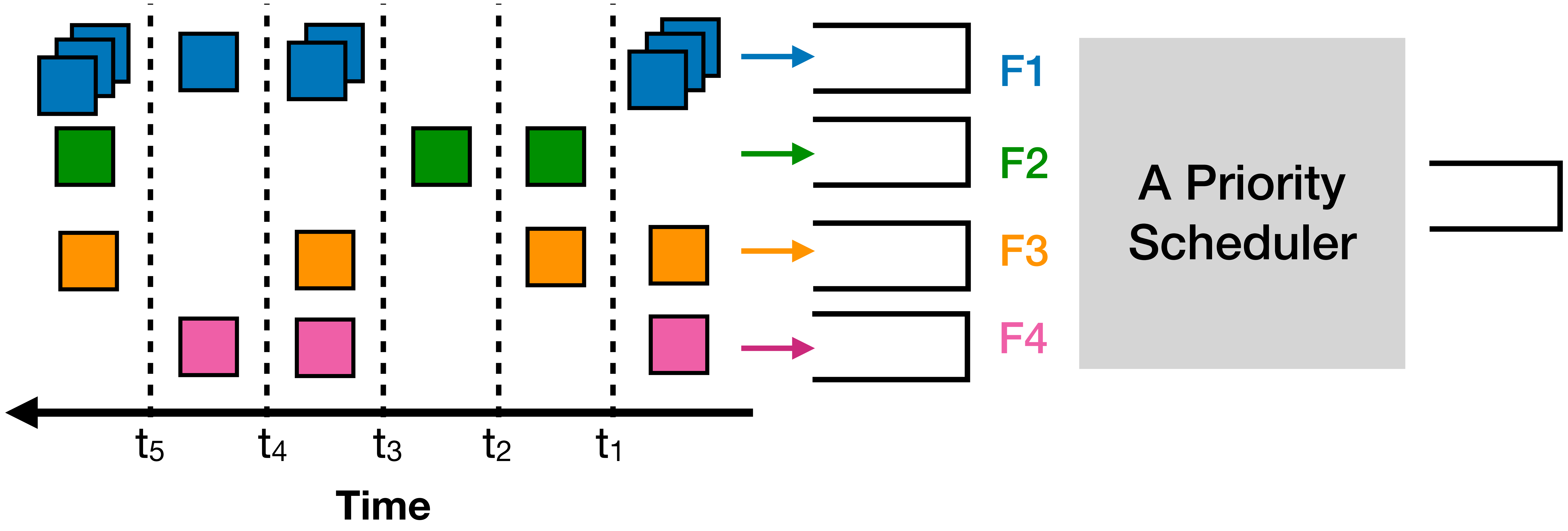


A single trace is not an informative output

- Not all details matter with respect to the property

ue (get starved)

e.g., for this particular input:

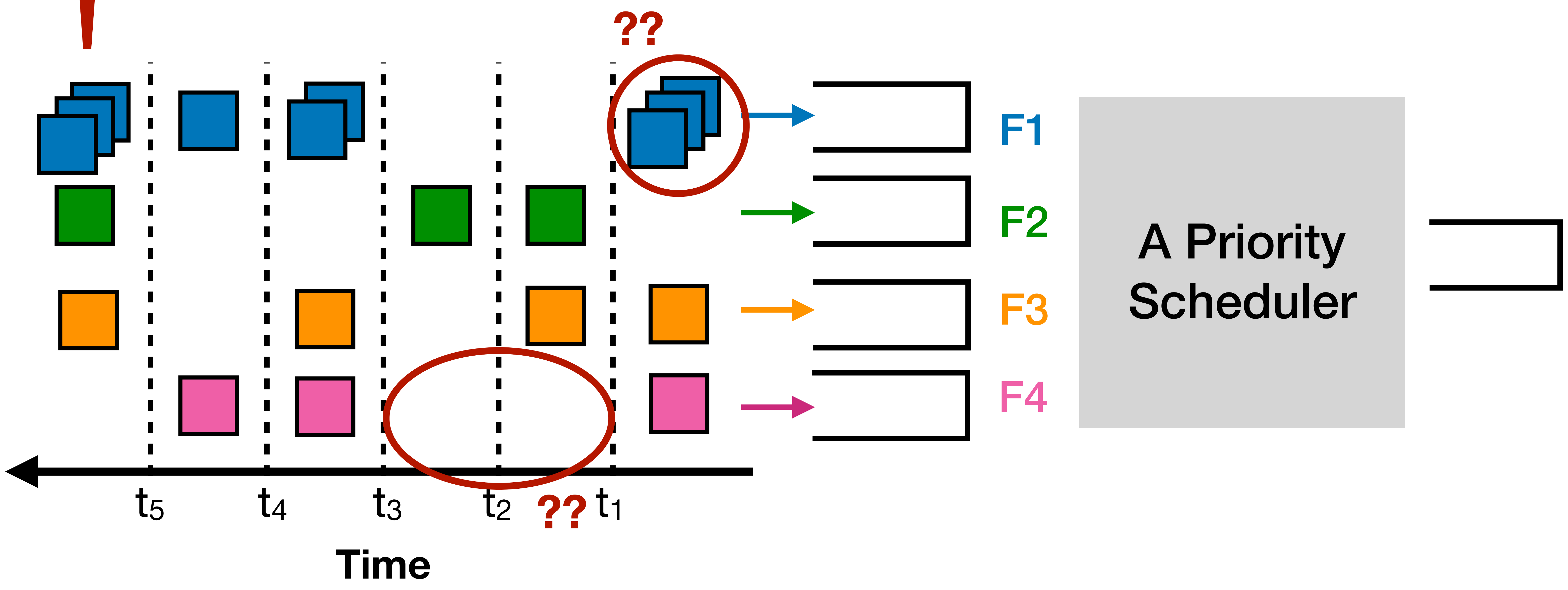


A single trace is not an informative output

- Not all details matter with respect to the property

ue (get starved)

e.g., for this particular input:

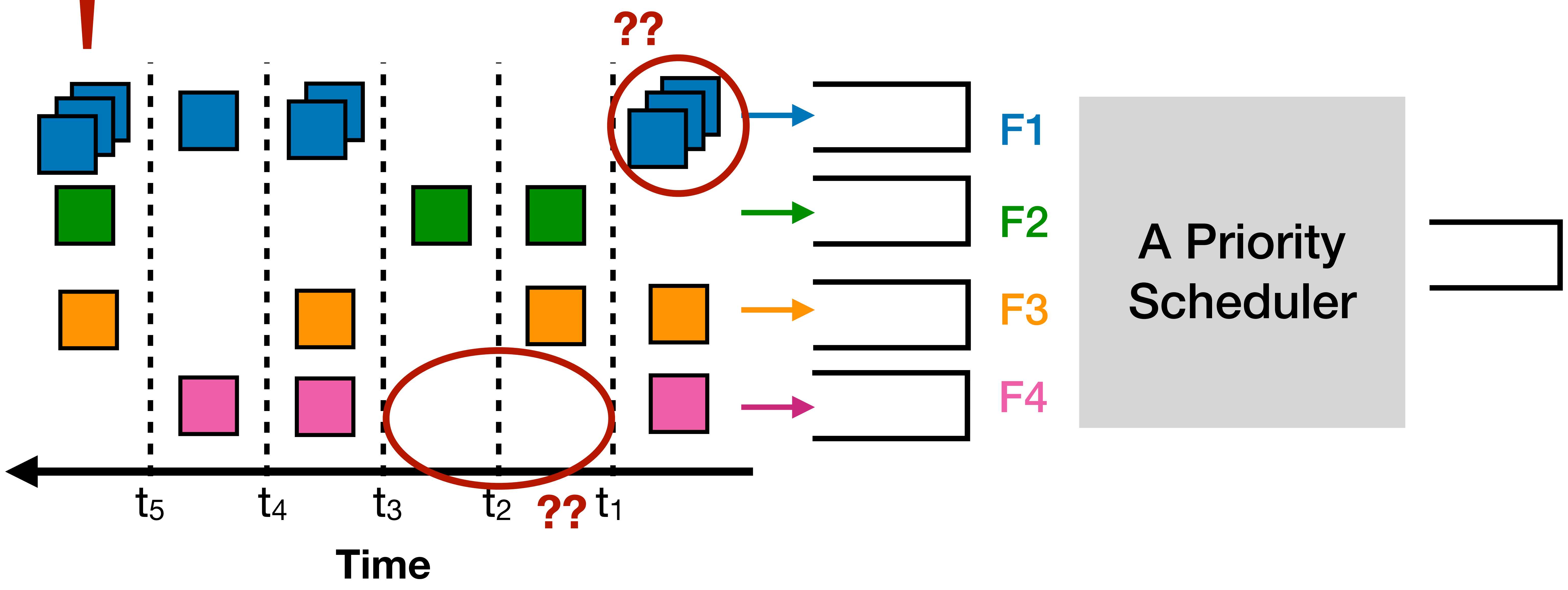


A single trace is not an informative output

- Not all details matter with respect to the property
- Unclear if it points to an “important” problem
  - Note the contrast to functional correctness properties

ue (get starved)

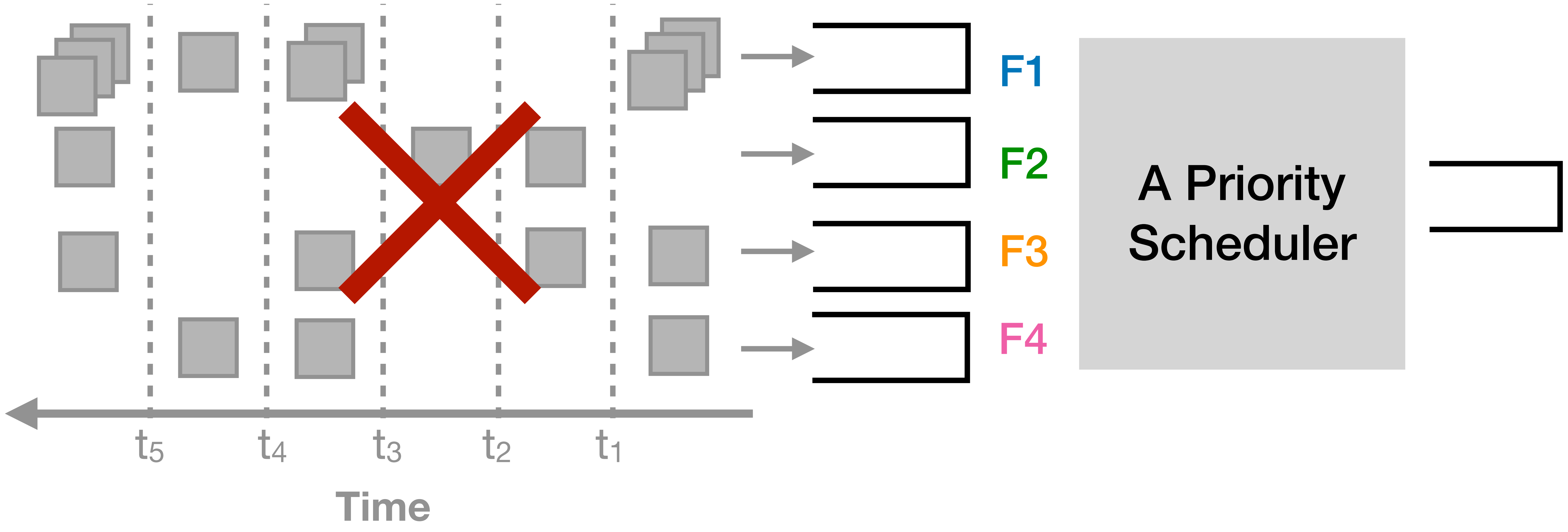
e.g., for this particular input:



A single trace is not an informative output

Property: F3 should not be blocked for dequeue (get starved) for X consecutive time steps.

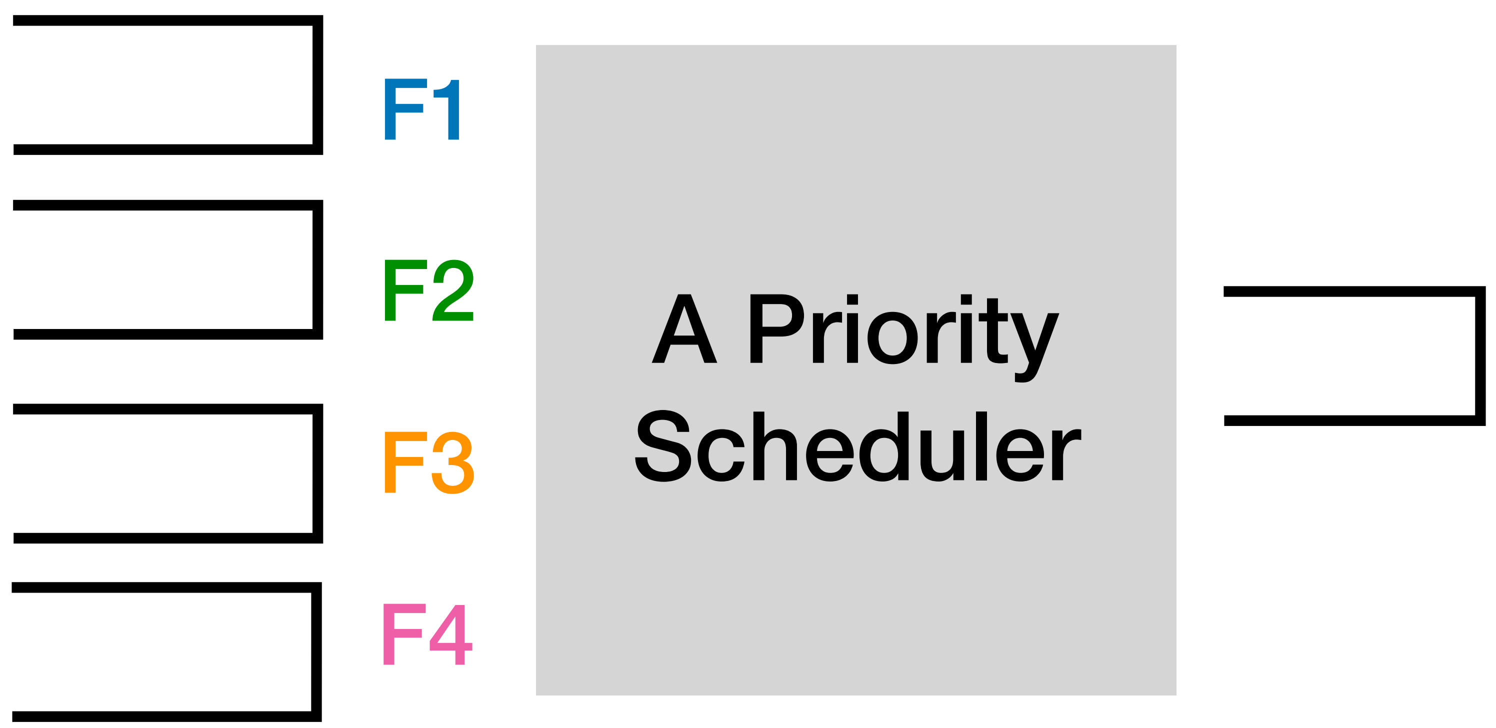
Output: Does not hold.
~~e.g., for this particular input:~~



Alternative? Conditions on the input

Property: F3 should not be blocked for dequeue (get starved) for X consecutive time steps.

Output: Does not hold.
~~e.g., for this particular input:~~
e.g., for these set of conditions on the input:



Alternative? Conditions on the input

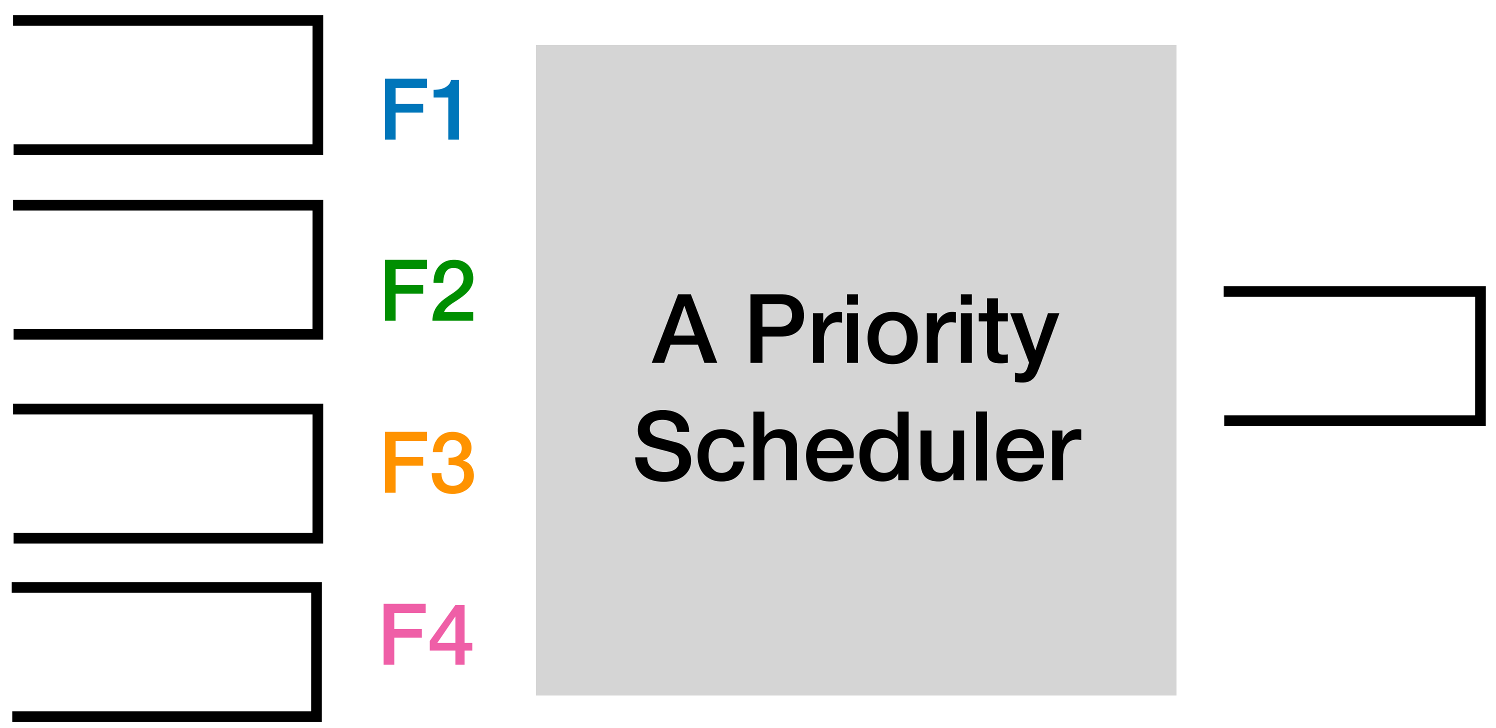
Property: F3 should not be blocked for dequeue (get starved) for X consecutive time steps.

Output: Does not hold.

~~e.g., for this particular input:~~

e.g., for these set of conditions on the input:

- F1 or F2 have packets for X consecutive time steps
- F3 has at least a packet



Alternative? Conditions on the input

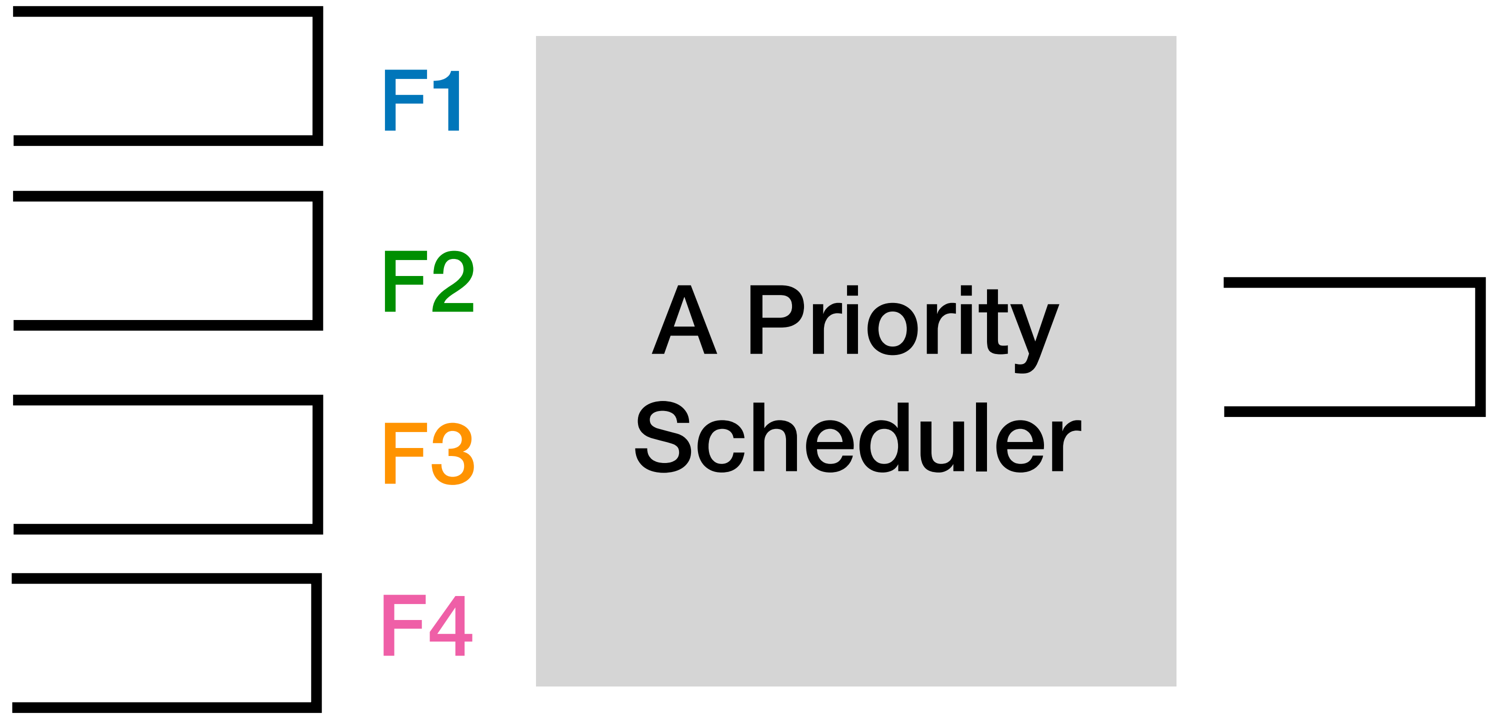
- **Workload:** Conjunction of constraints on the input

ueue (get starved)

on the input:

Workload

- $F1$ or $F2$ have packets for X consecutive time steps
- $F3$ has at least a packet



Alternative? Conditions on the input

- **Workload:** Conjunction of constraints on the input

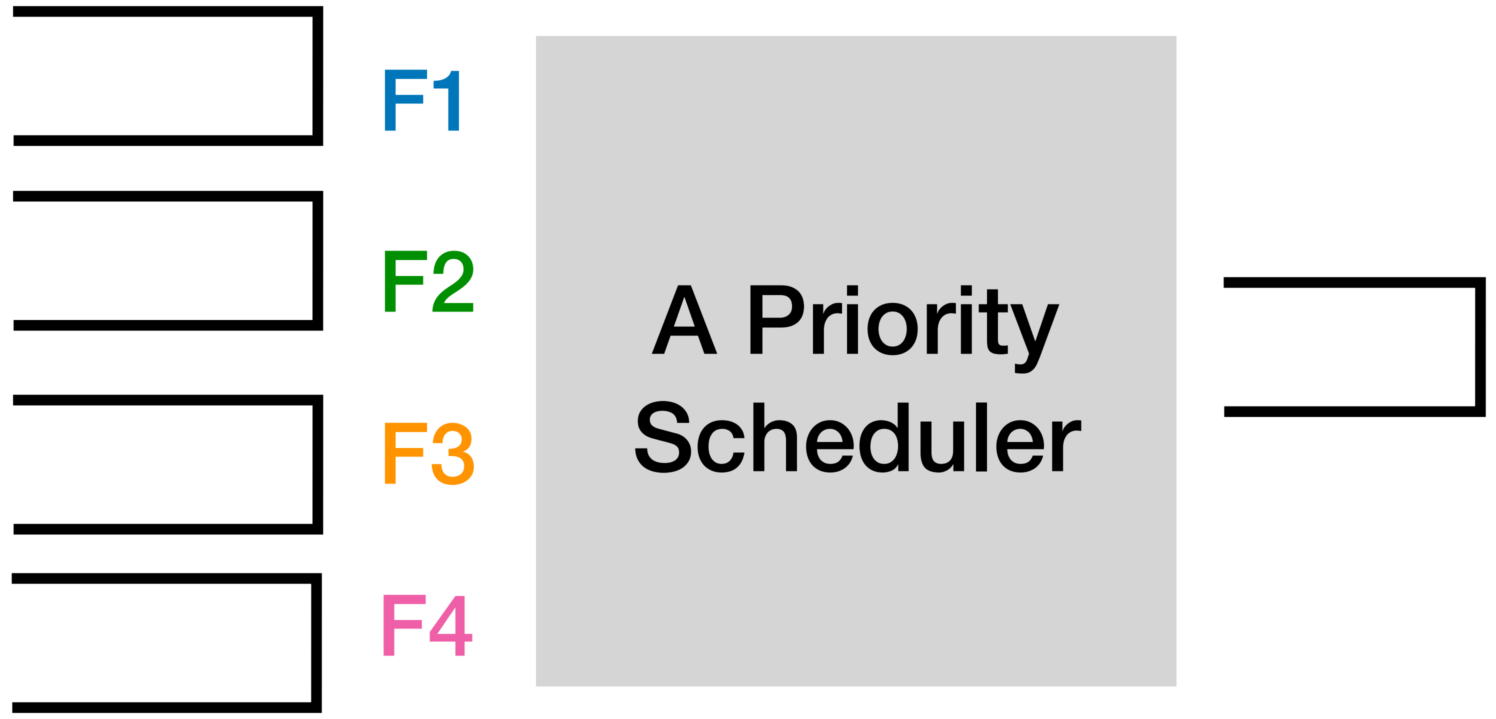
ueue (get starved)

$$\forall t \in [1,X]\ \Sigma_{q\in\{F_1,F_2\}}\ total\_packets(q,t) \geq t$$
$$\wedge\ \forall t \in [1,X]\ \ total\_packets(F_3,t) \geq 1$$

on the input:

Workload

- F1 or F2 have packets for X consecutive time steps
- F3 has at least a packet



Alternative? Conditions on the input

- **Workload:** Conjunction of constraints on the input
- (Concisely) represents a set of traces
  - More informative
  - Indicative of a more prominent problem.

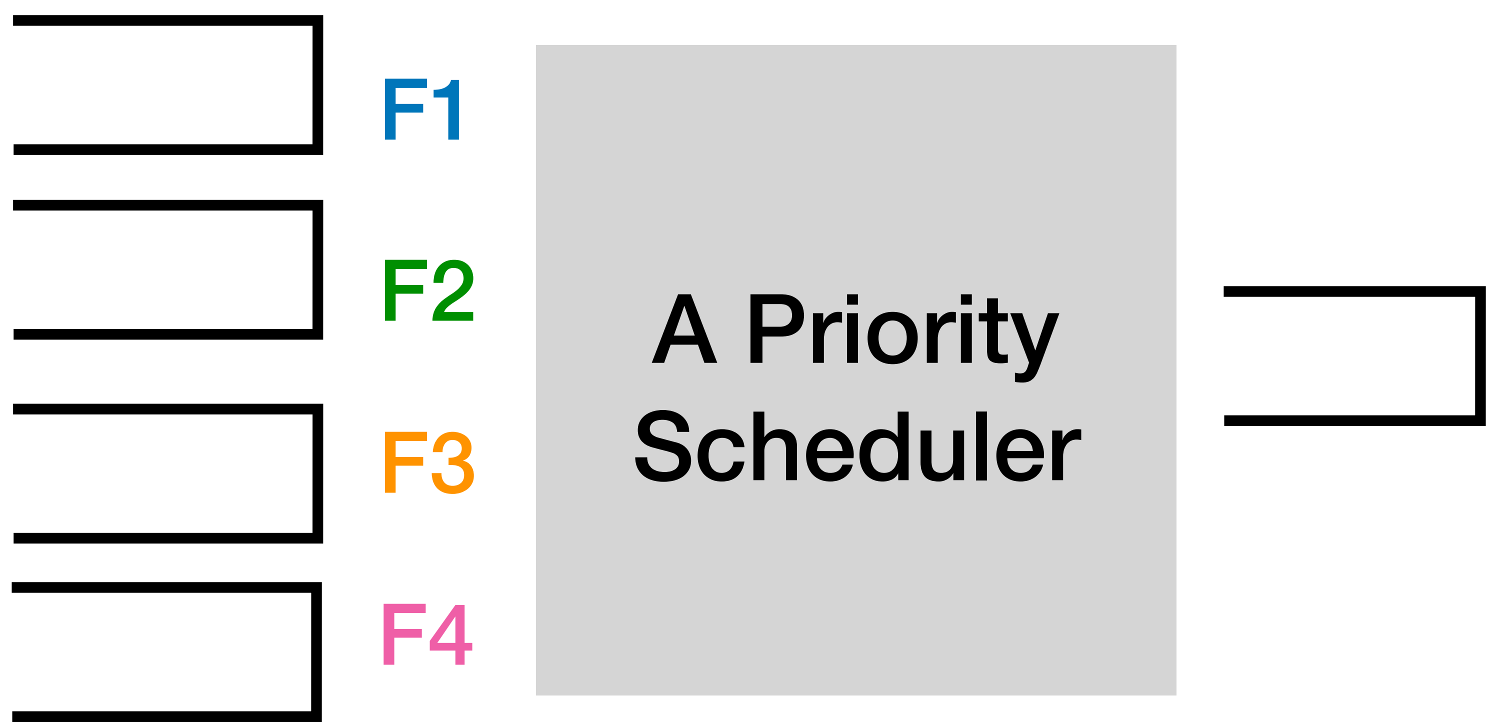
ueue (get starved)

$$\forall t \in [1, X]\ \Sigma_{q \in \{F_1, F_2\}}\ total\_packets(q, t) \geq t$$
$$\land\ \forall t \in [1, X]\ total\_packets(F_3, t) \geq 1$$

on the input:

Workload

- F1 or F2 have packets for X consecutive time steps
- F3 has at least a packet



Synthesizing workloads

1 Create a mathematical model of the network

Composition of queuing modules

2 Specify desired property

Property P:
$queue\_size(q_1, t_1) \leq 10$

3 Automatically analyze the entire input space.

Bounded Model checking with Z3

4 Prove or disprove the property

✔ Property P always holds

✘ An example input for which P does not hold

Synthesizing workloads

1 Create a mathematical model of the network

Composition of queuing modules

2 Specify desired property

Property P:
$queue\_size(q_1, t_1) \leq 10$

3 Automatically analyze the entire input space.

Bounded Model checking with Z3

4 Prove or disprove the property

✔ Property P always holds

✘ **A workload wl such that P does not hold for any trace in wl**

Synthesizing workloads

1 Create a math model of the r

Composition of queuing modules

Syntax-Guided Synthesis

2 Specify desired property

Property P:
$queue\_size(q_1, t_1) \leq 10$

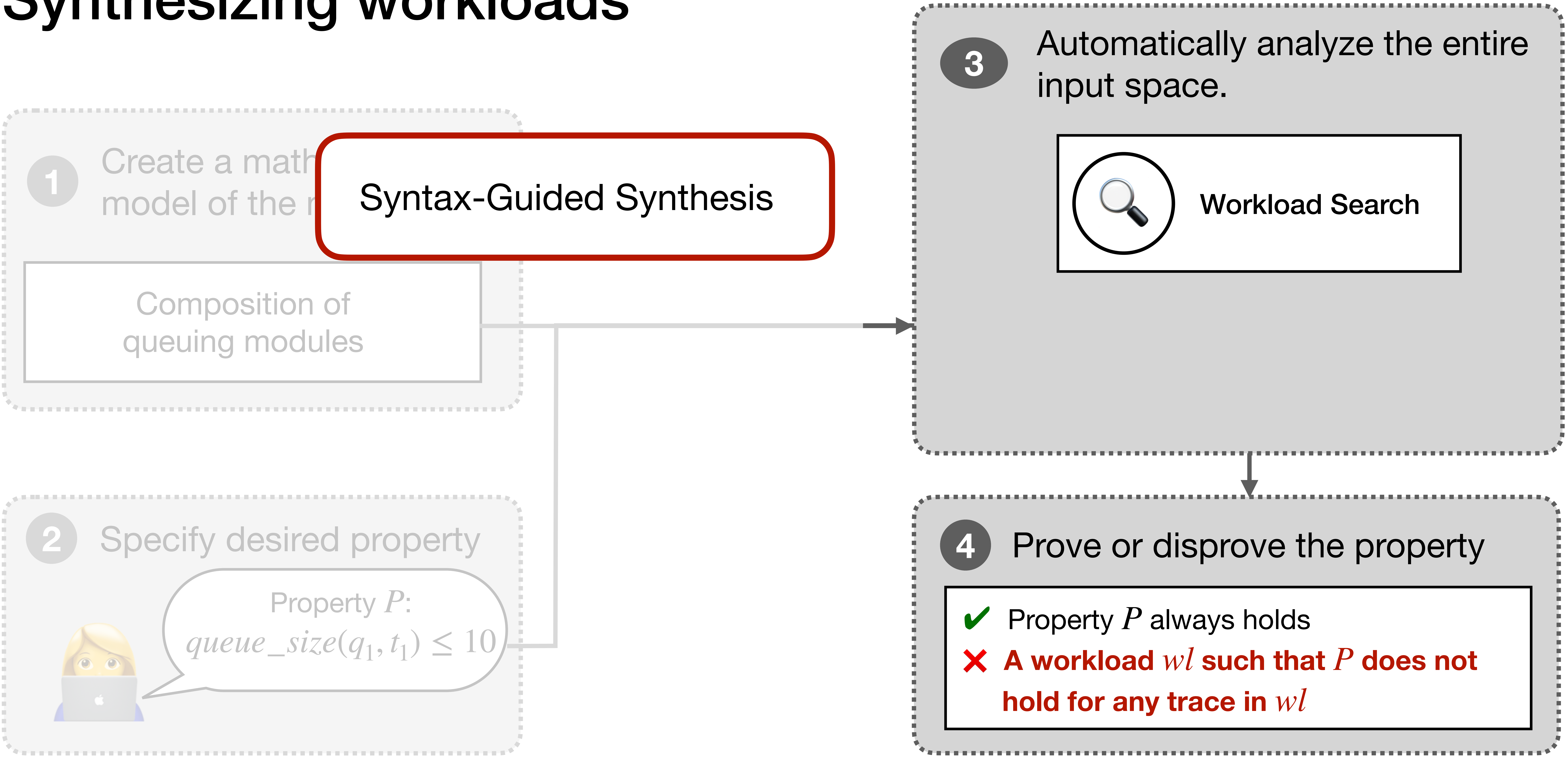
3 Automatically analyze the entire input space.

4 Prove or disprove the property

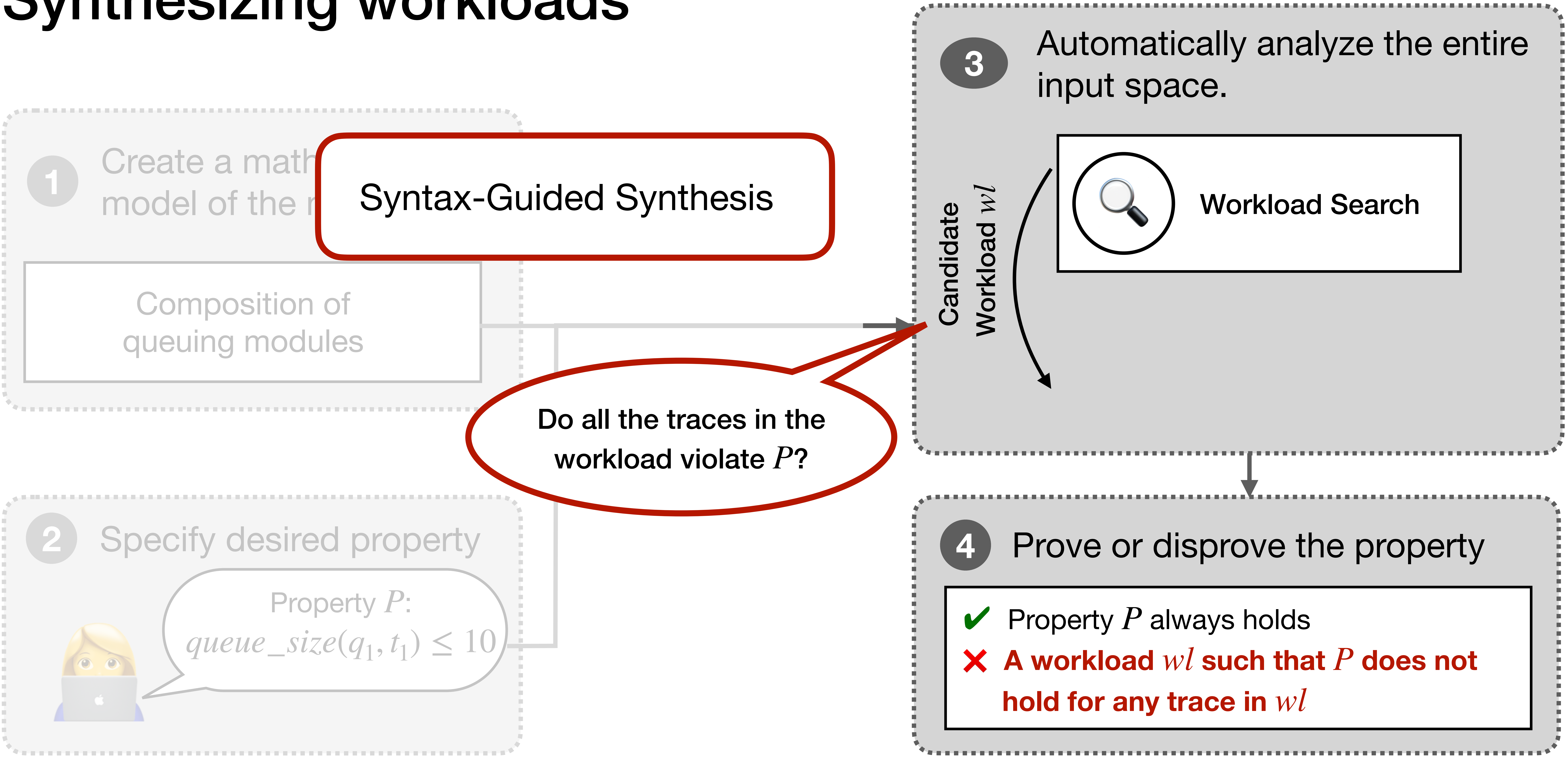
✔ Property P always holds

✘ **A workload wl such that P does not hold for any trace in wl**

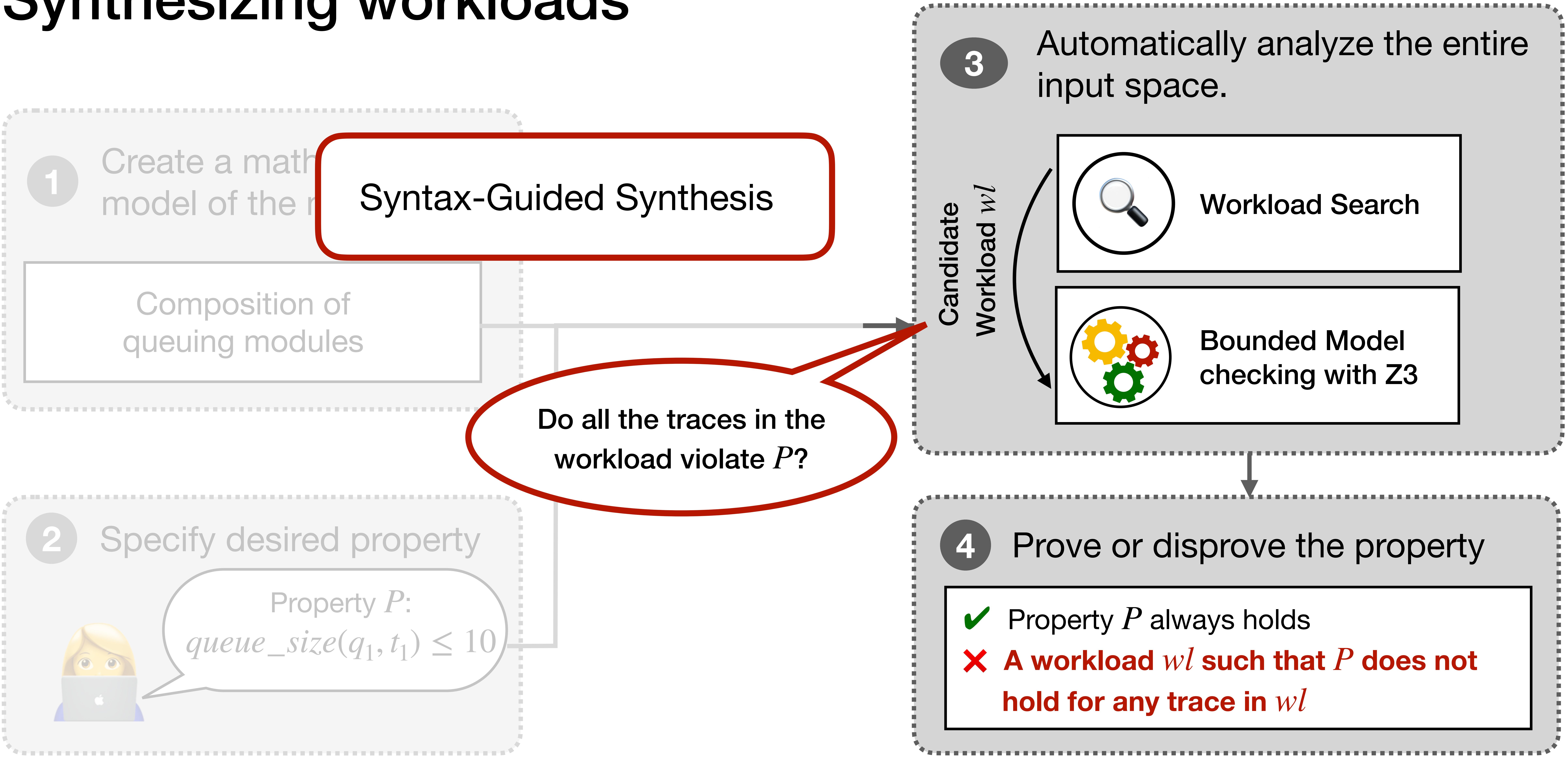
Synthesizing workloads



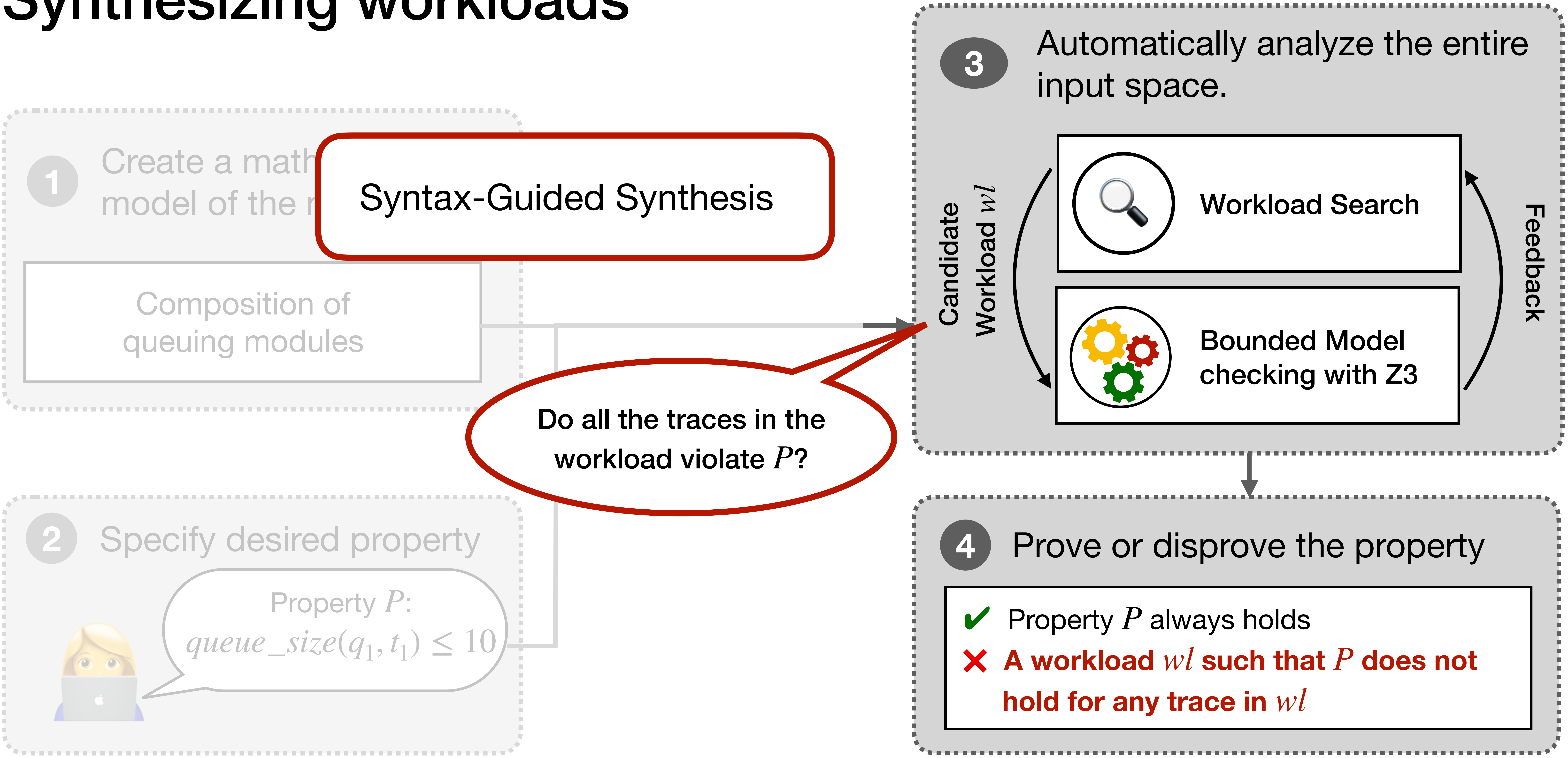
Synthesizing workloads



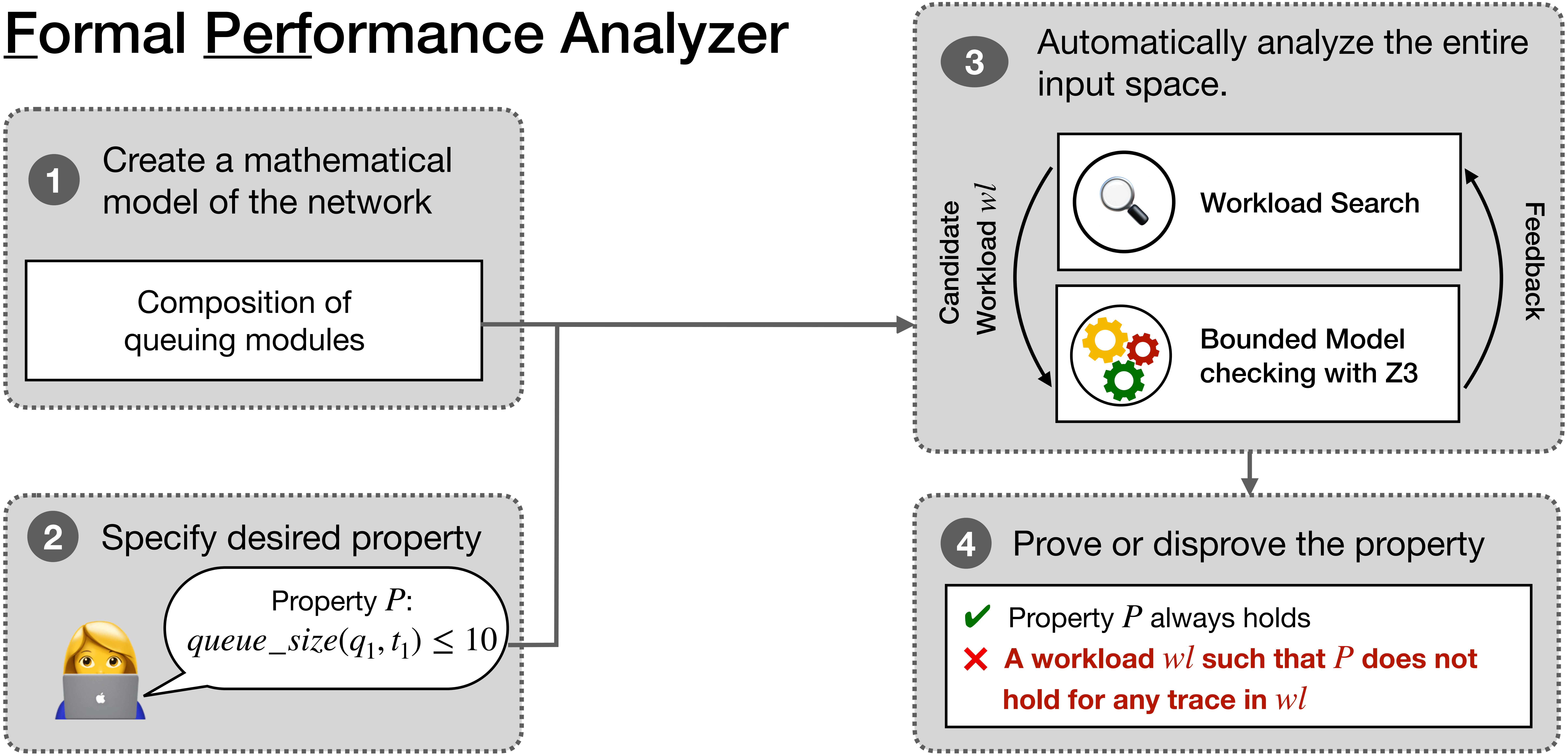
Synthesizing workloads



Synthesizing workloads



FPerf:
Formal Performance Analyzer



See the paper for

- Details of the search algorithm
  - Randomized search
  - Guided by a cost function over workloads
- Generating example traces for the search cost function
- Optimizations for the search and verification process
- Constraining the input search space to the user’s interest
- …

Case study - Packet scheduling

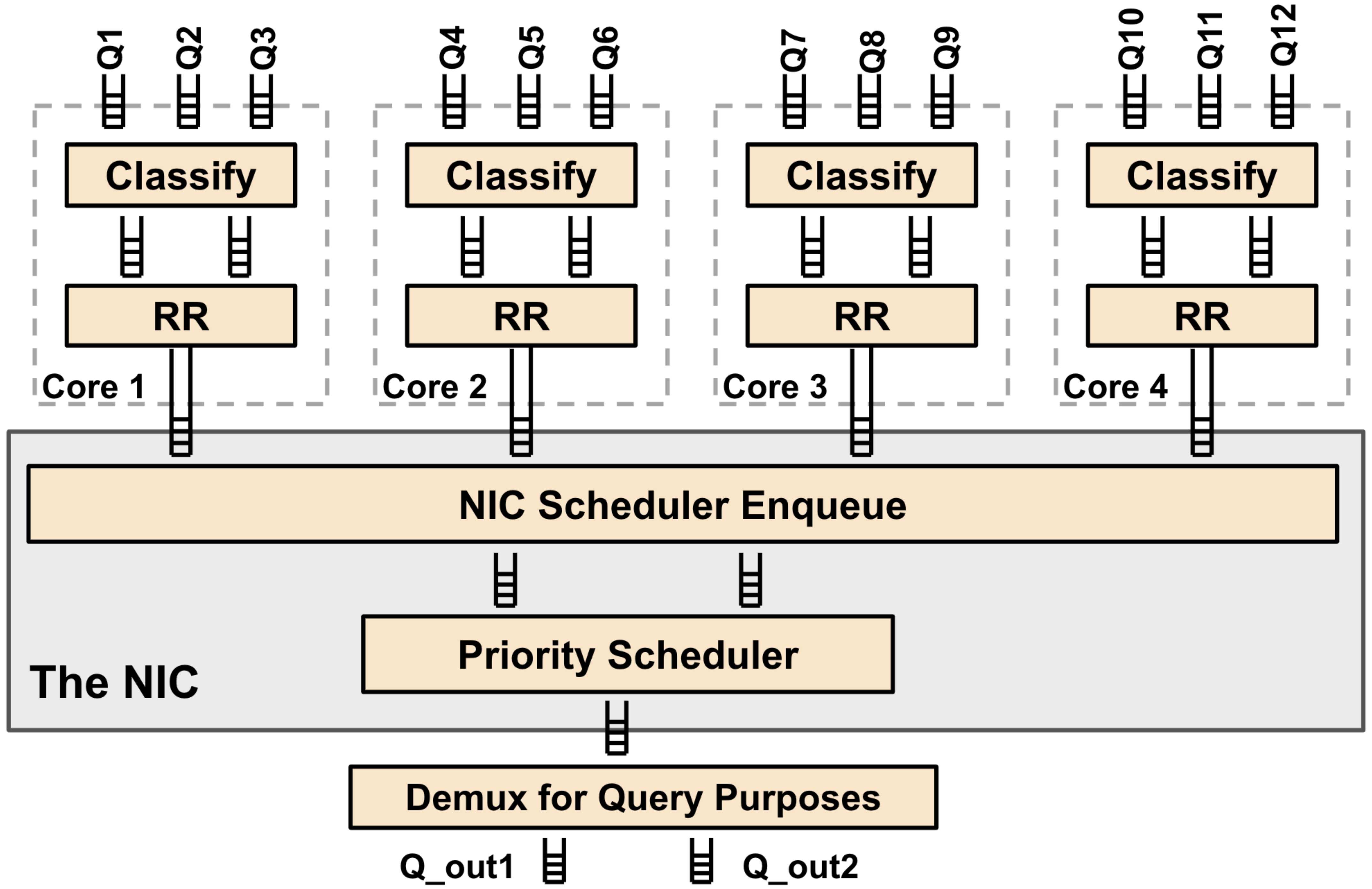
	Stand-alone packet schedulers		
	Priority	**Round-Robin**	**FQ in FQ-CoDel**
Property	Starvation	Fairness	Fairness

Case study - Packet scheduling

	$\overbrace{\textbf{Priority} \quad \textbf{Round-Robin} \quad \textbf{FQ in FQ-CoDel}}^{\text{Stand-alone packet schedulers}}$			
	Priority	**Round-Robin**	**FQ in FQ-CoDel**	**Composition**
Property	Starvation	Fairness	Fairness	Starvation + Fairness

Case study - Packet scheduling

- 11 queuing modules
- Host + NIC scheduling
- Inspired from Loom (NSDI’19)



Composition

Starvation + Fairness

Case study - Packet scheduling

	Priority	**Round-Robin**	**FQ in FQ-CoDel**	**Composition**
Property	Starvation	Fairness	Fairness	Starvation + Fairness

Case study - Packet scheduling

10s of thousands variables
and constraints

		Priority	Round-Robin	FQ in FQ-CoDel	Composition
Property		Starvation	Fairness	Fairness	Starvation + Fairness
Model Size	**# variables**	1.5K	2.6K	4.5K	17.9K
	# constraints	7K	13K	21K	94K

Case study - Packet scheduling

Search time is reasonable
Example generation is a bottleneck

		Priority	Round-Robin	FQ in FQ-CoDel	Composition
Property		Starvation	Fairness	Fairness	Starvation + Fairness
Model Size	# variables	1.5K	2.6K	4.5K	17.9K
	# constraints	7K	13K	21K	94K
The Search Algorithm	# rounds	65	268	769	361
	time (sec.)	3	59	223	461

Case study - Packet scheduling

Workload verification (and model analysis) is efficient!

		Priority	Round-Robin	FQ in FQ-CoDel	Composition
Property		Starvation	Fairness	Fairness	Starvation + Fairness
Model Size	**# variables**	1.5K	2.6K	4.5K	17.9K
	# constraints	7K	13K	21K	94K
The Search Algorithm	**# rounds**	65	268	769	361
	time (sec.)	3	59	223	461
Verifying Candidate Workloads (avg) (sec.)		0.03	0.04	0.10	0.81

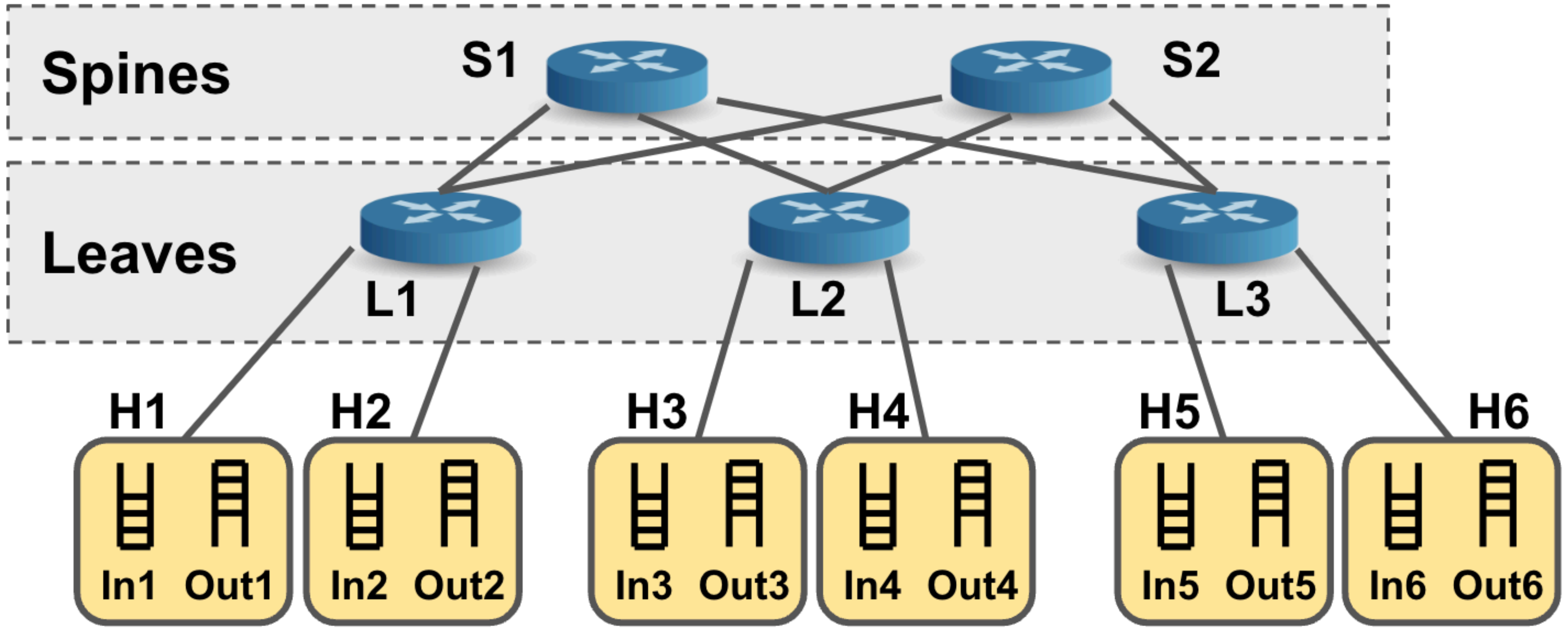
Case study - Packet scheduling

It is possible to synthesize workloads in a few minutes

		Priority	Round-Robin	FQ in FQ-CoDel	Composition
Property		Starvation	Fairness	Fairness	Starvation + Fairness
Model Size	# variables	1.5K	2.6K	4.5K	17.9K
	# constraints	7K	13K	21K	94K
The Search Algorithm	# rounds	65	268	769	361
	time (sec.)	3	59	223	461
Verifying Candidate Workloads (avg) (sec.)		0.03	0.04	0.10	0.81
Total Time (min.)		0.2	6.2	9.6	18.5

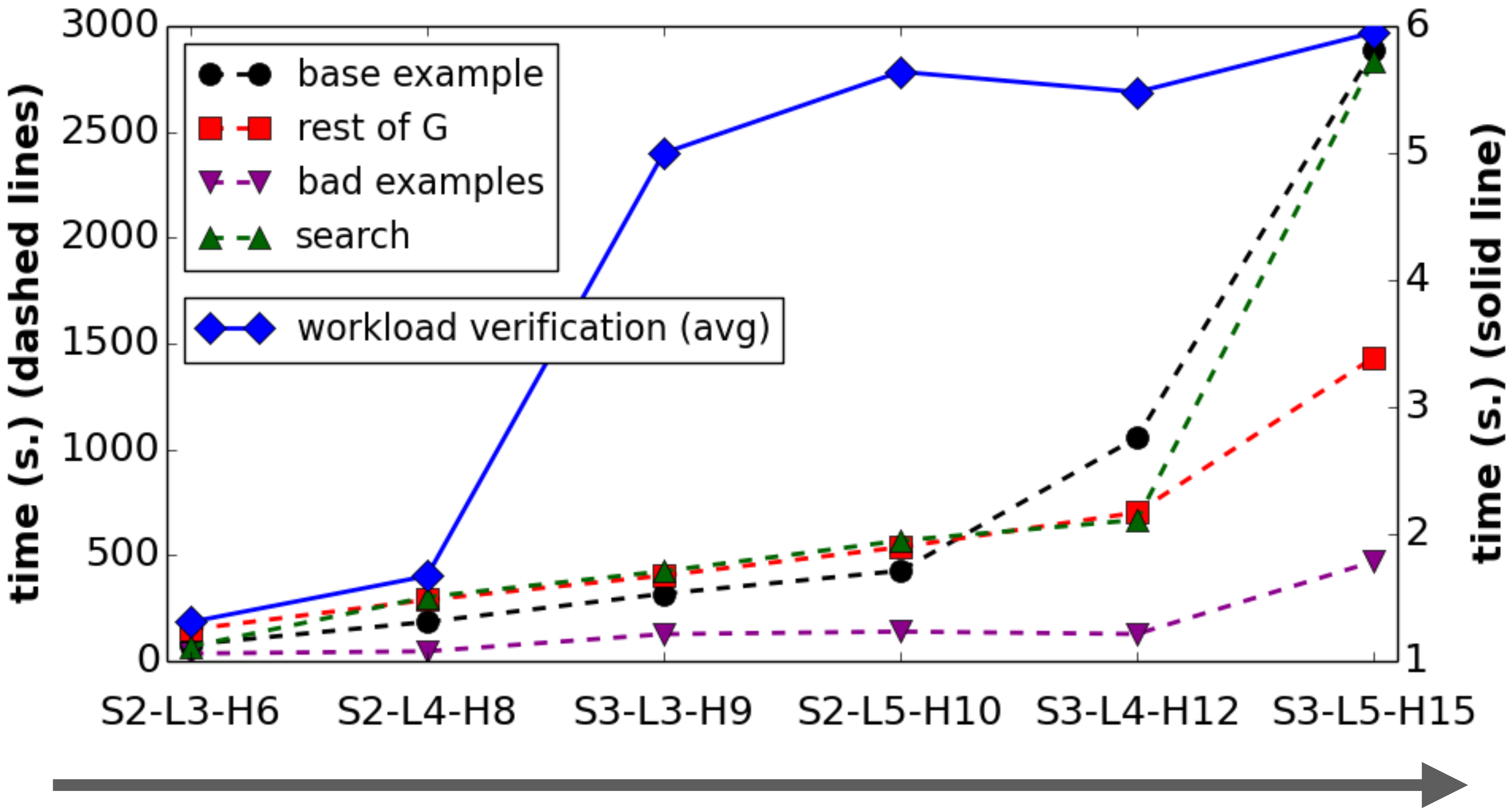
Case study - A (small) leaf-spine network

- Modeled with ~23 queuing modules with 66 queues
    - ~twice larger than the packet composition case study
- Asked about properties related to throughput and latency
- Observed similar trends



Case study - A (small) leaf-spine network

- The trend is (unsurprisingly) exponential
- Modular analysis will be crucial for scale



Larger network sizes
(Sx-Ly-Hz: x Spines, y Leaves, z Hosts)

Concluding remarks

- **Our goal:** Exploring the transition from reasoning about functional correctness to performance properties

- **Our findings:** Intriguing implications on modeling and analysis techniques.
  - e.g., workloads as opposed to individual counter examples

- We are excited about the possibilities ahead!
  - FPerf's code is available on GitHub: https://github.com/minmit/fperf
  - And we are actively looking for more use cases to improve