The Anytime Automaton

Joshua San Miguel

Natalie Enright Jerger

The Edward S. Rogers Sr. Department
of Electrical & Computer Engineering
UNIVERSITY OF TORONTO

Summary

We propose the **Anytime Automaton**:

- A new computation model for approximate computing.

Summary

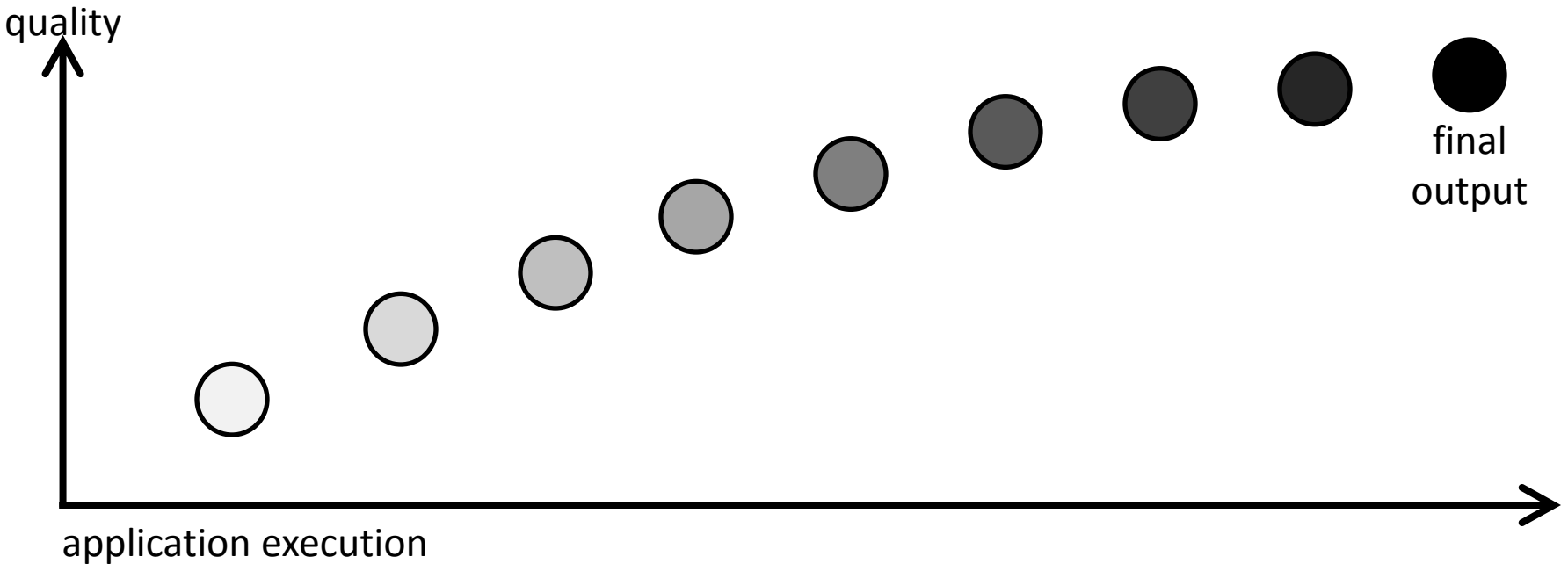
We propose the **Anytime Automaton**:

- A new computation model for approximate computing.

Summary

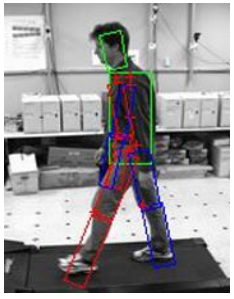
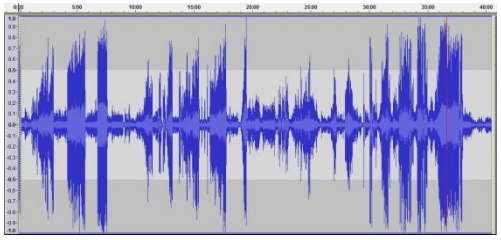
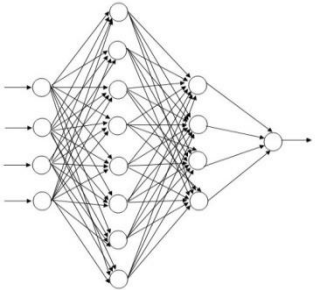
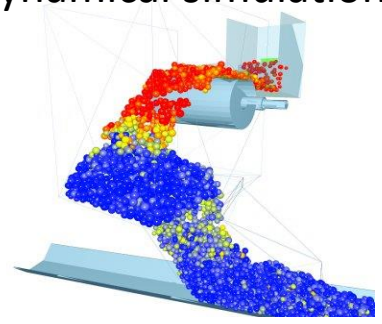
We propose the **Anytime Automaton**:

- A new computation model for approximate computing.



Approximate Computing

Many applications are inherently noisy and imprecise.

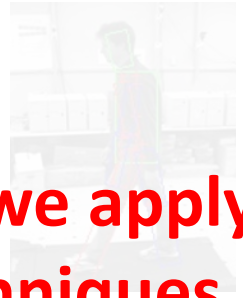
Data mining  http://www.zentut.com/	Computer vision  http://www.cc.gatech.edu/~cnieto6/	Audio and video processing  http://themusicparlour.blogspot.ca/
Gaming  http://www.businessweek.com/	Machine learning  http://www.analyticbridge.com/	Dynamical simulation  http://www.scientific-computing.com/

Approximate Computing

Many applications are inherently noisy and imprecise.

Data mining

Computer vision



Audio and video processing

http://themusicparlour.blogspot.ca/

But how can we apply approximate computing techniques and still ensure acceptability in final output?

Gaming

http://www.businessweek.com/

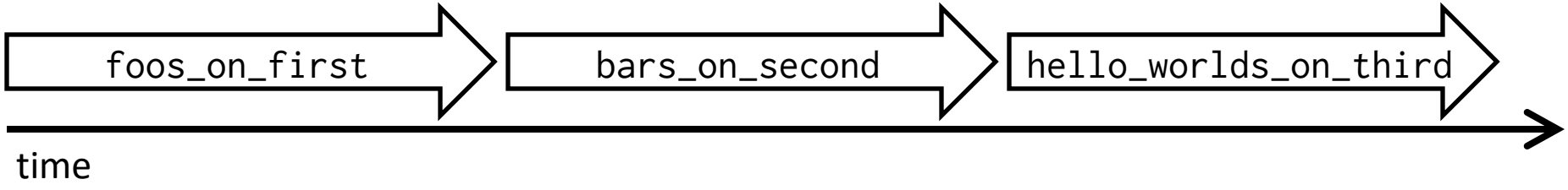
http://www.analyticbridge.com/

namical simulation

http://www.scientific-computing.com/

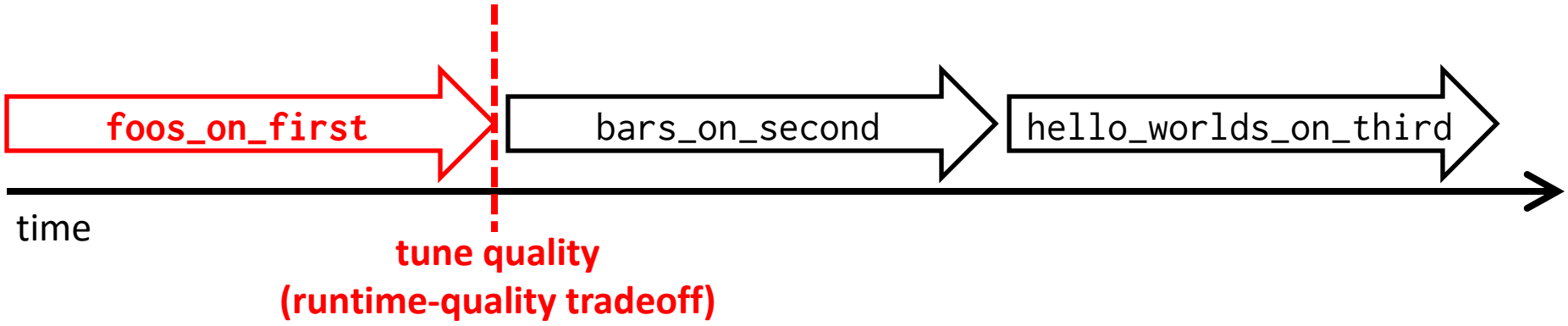
Approximate Computing

```
program ()
{
    foos_on_first();
    bars_on_second();
    hello_worlds_on_third();
}
```



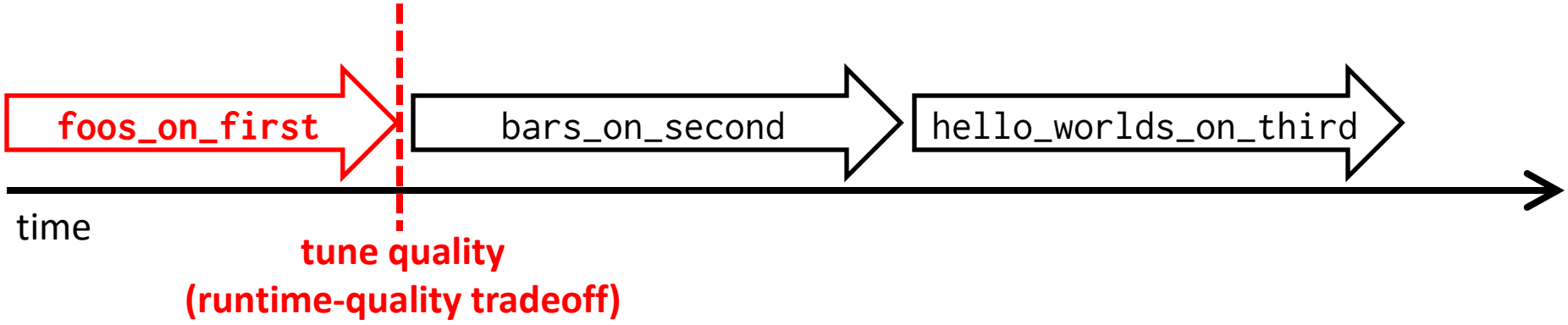
Approximate Computing

```
program ()
{
    approx_foos_on_first();
    bars_on_second();
    hello_worlds_on_third();
}
```



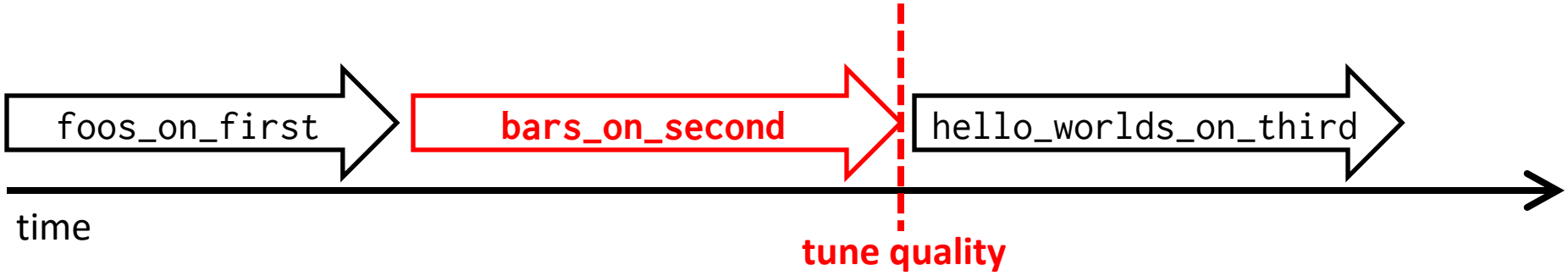
Approximate Computing

```
program ()
{
    approx_foos_on_first();
    bars_on_second();
    hello_worlds_on_third();
}
```



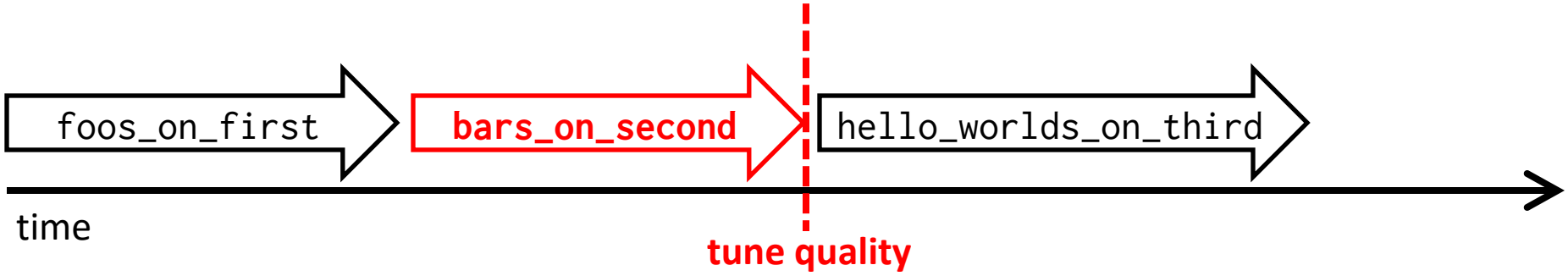
Approximate Computing

```
program ()
{
    approx_foos_on_first();
    approx_bars_on_second();
    hello_worlds_on_third();
}
```



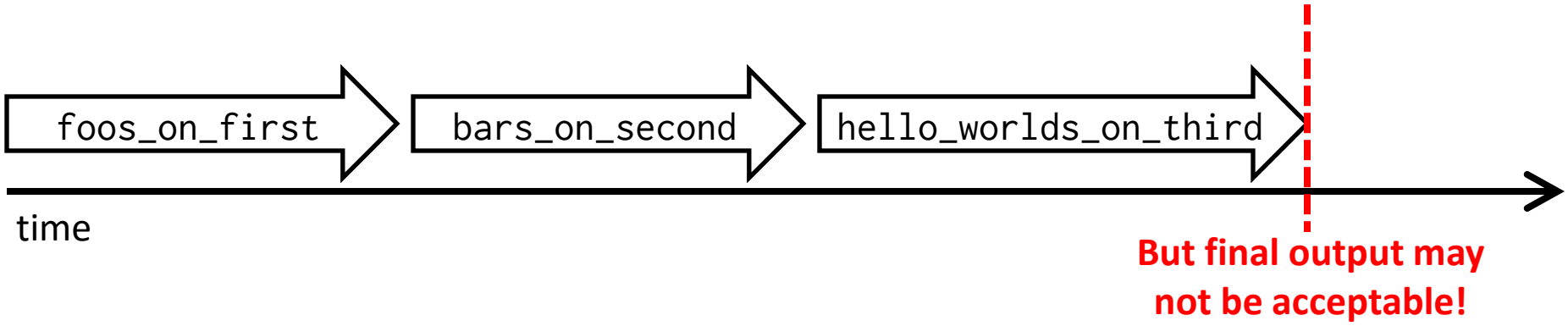
Approximate Computing

```
program ()
{
    approx_foos_on_first();
    approx_bars_on_second();
    hello_worlds_on_third();
}
```



Approximate Computing

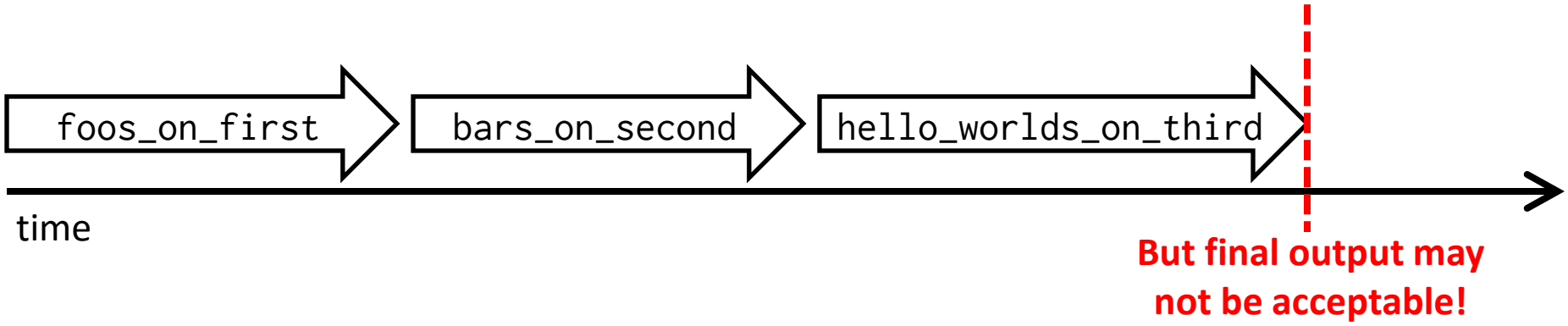
```
program ()
{
    approx_foos_on_first();
    approx_bars_on_second();
    hello_worlds_on_third();
}
```



Approximate Computing

Difficult to ensure acceptability of final output on-the-fly, since quality control limited to local approximations and not their composition.

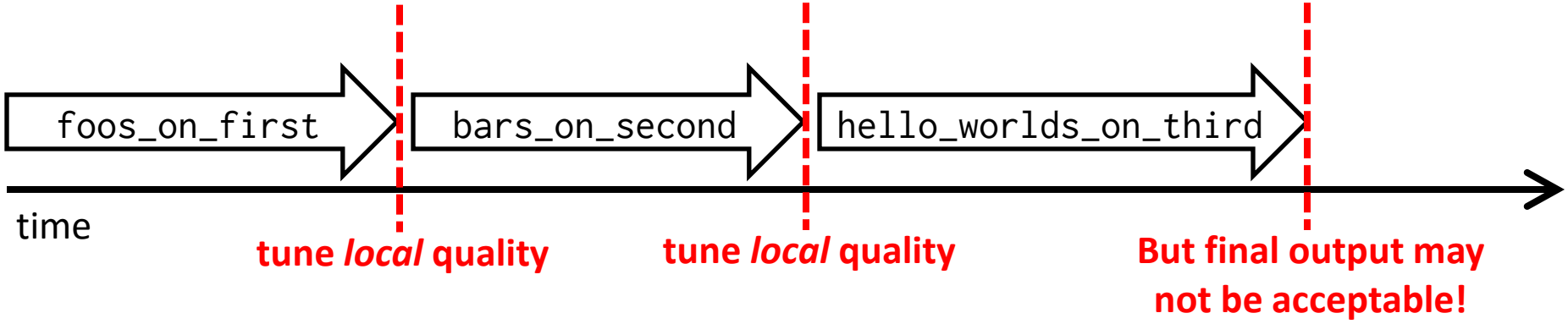
(Challenge #1: Holistic Quality Control)



Approximate Computing

Difficult to ensure acceptability of final output on-the-fly, since quality control limited to local approximations and not their composition.

(Challenge #1: Holistic Quality Control)



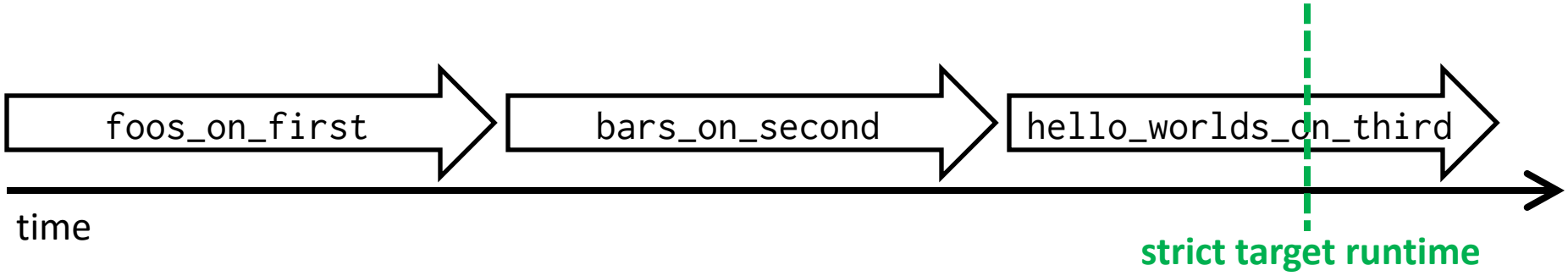
Real-Time Computing

Real-time systems impose strict runtime constraints; loss in output quality more tolerable than not finishing in time.

Streaming multimedia	Automotive systems	Telecommunications
		
http://www.streamingvideoprovider.co.uk/	http://www.beaudaniels-illustration.com/	http://m.exed.hec.edu/

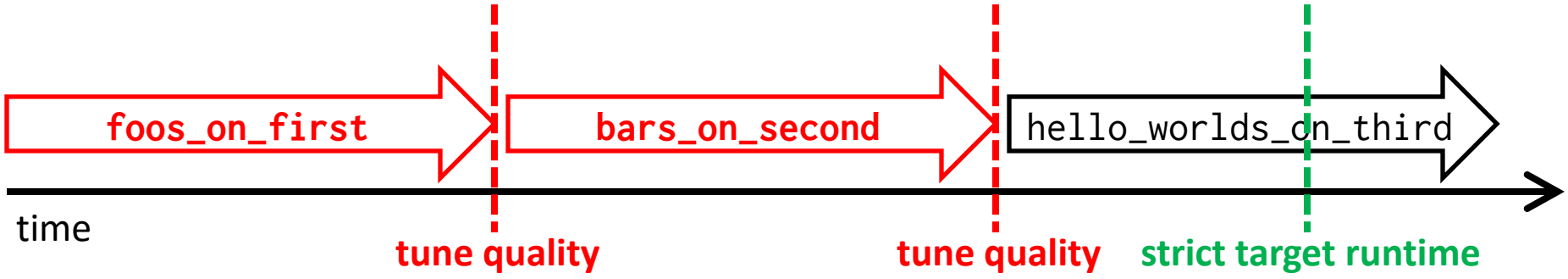
Real-Time Computing

```
program ()
{
    approx_foos_on_first();
    approx_bars_on_second();
    hello_worlds_on_third();
}
```



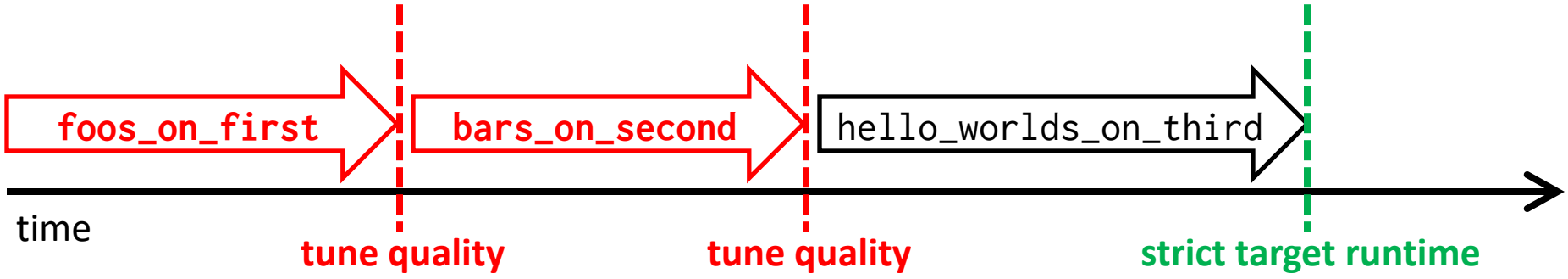
Real-Time Computing

```
program ()
{
    approx_foos_on_first();
    approx_bars_on_second();
    hello_worlds_on_third();
}
```



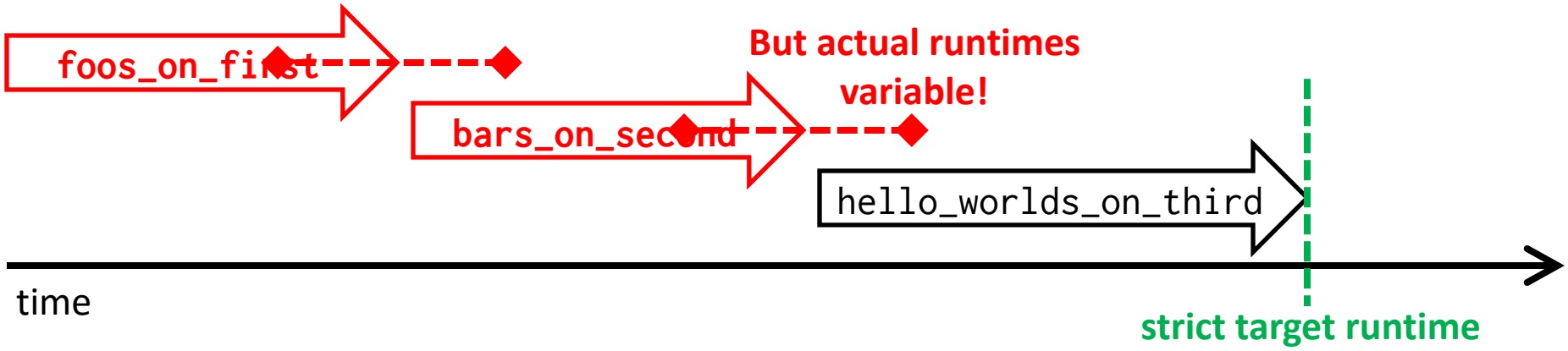
Real-Time Computing

```
program ()
{
    approx_foos_on_first();
    approx_bars_on_second();
    hello_worlds_on_third();
}
```



Real-Time Computing

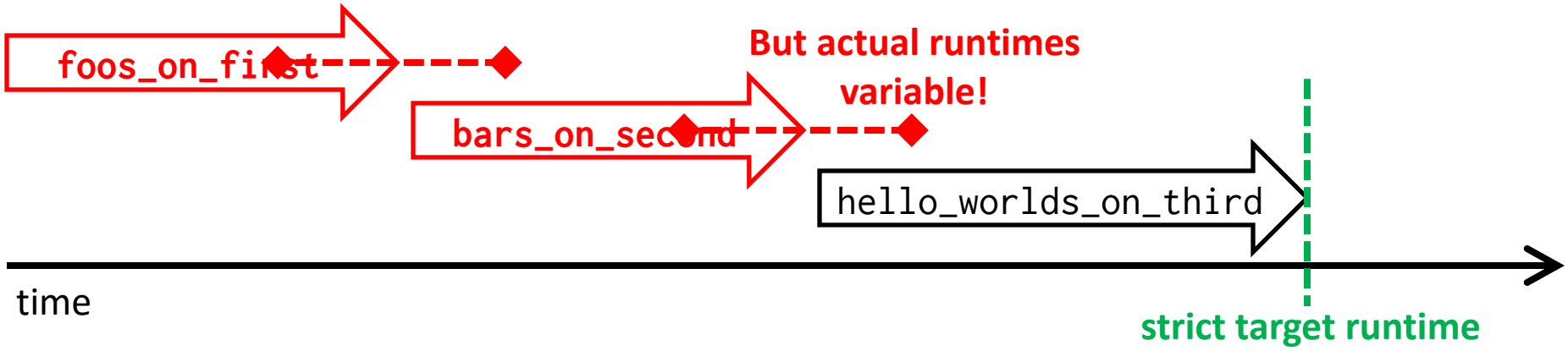
```
program ()
{
    approx_foos_on_first();
    approx_bars_on_second();
    hello_worlds_on_third();
}
```



Real-Time Computing

Difficult to ensure strict real-time constraints are met (i.e., interrupt the application), since runtime-quality tradeoffs vary dynamically.

(Challenge #2: Interruptibility)



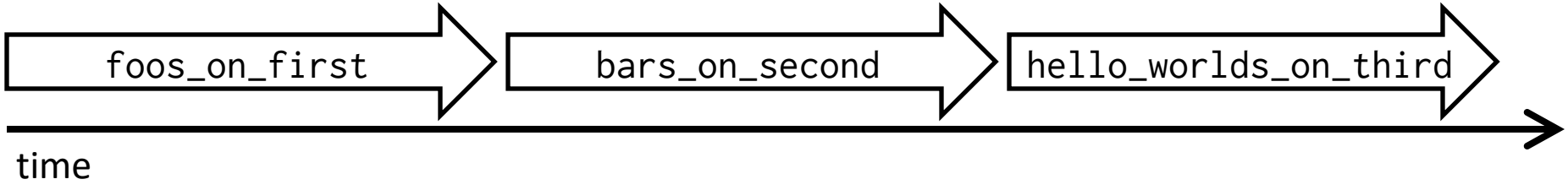
User-Interactive Computing

In user-interactive environments, users dictate quality requirements on-the-fly.

Gaming	Mobile vision	Search/recommendation
		
http://www.businessweek.com/	http://www.pcadvisor.co.uk/	http://www.expressvpn.com/

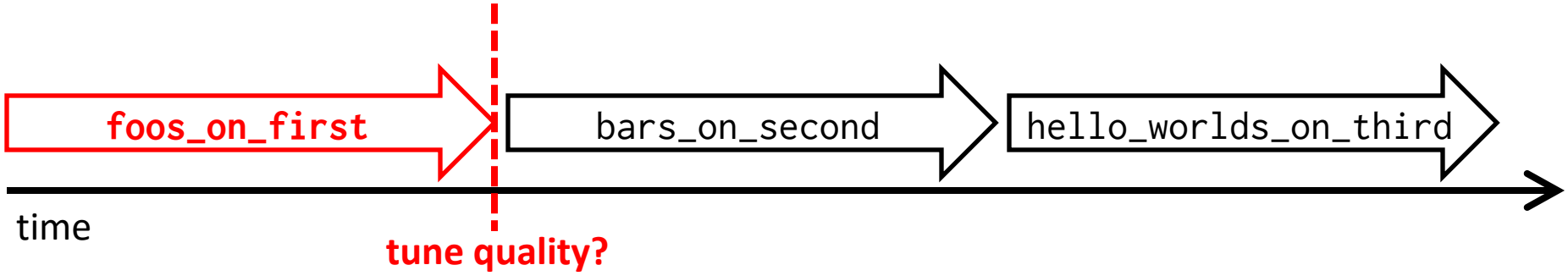
User-Interactive Computing

```
program ()
{
    approx_foos_on_first();
    approx_bars_on_second();
    hello_worlds_on_third();
}
```



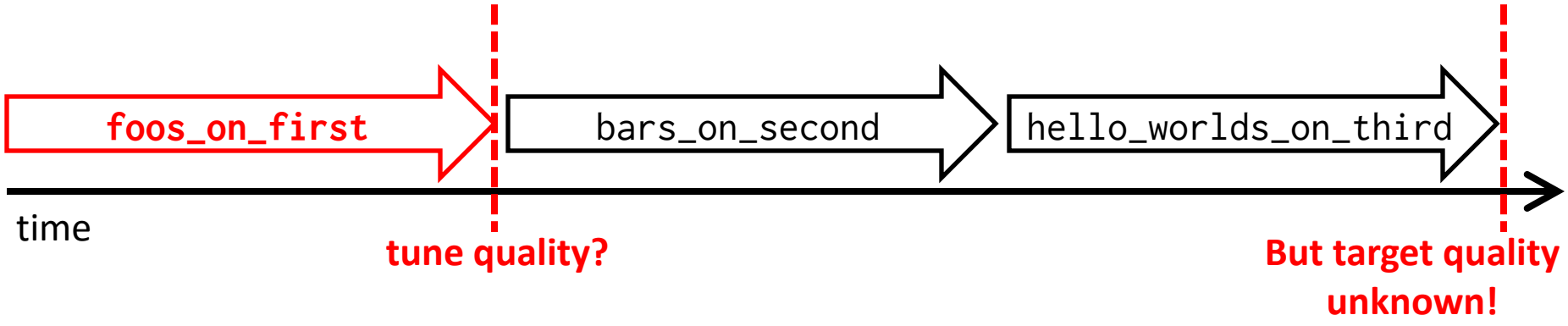
User-Interactive Computing

```
program ()
{
    approx_foos_on_first();
    approx_bars_on_second();
    hello_worlds_on_third();
}
```



User-Interactive Computing

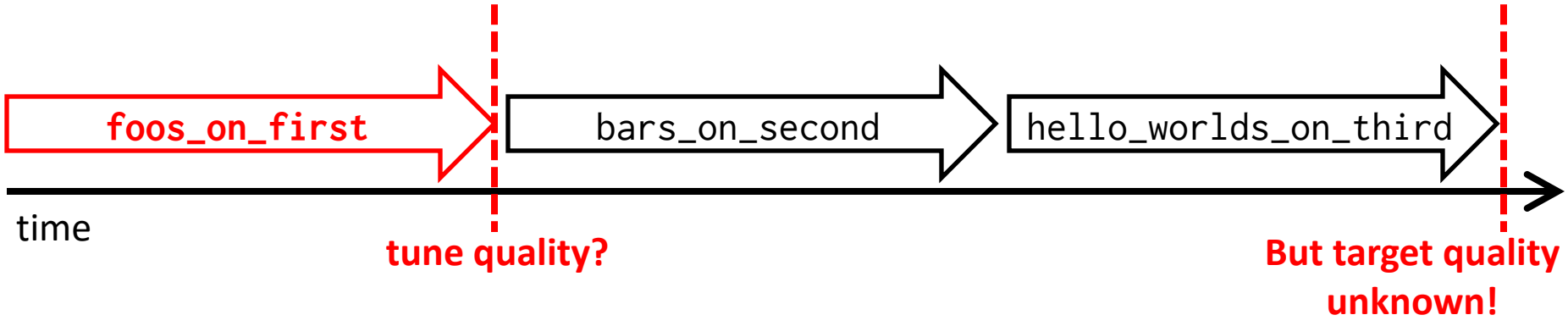
```
program ()
{
    approx_foos_on_first();
    approx_bars_on_second();
    hello_worlds_on_third();
}
```



User-Interactive Computing

Difficult to ensure acceptability for a given user at a given context, since acceptable quality cannot be determined a priori.

(Challenge #3: User Flexibility)

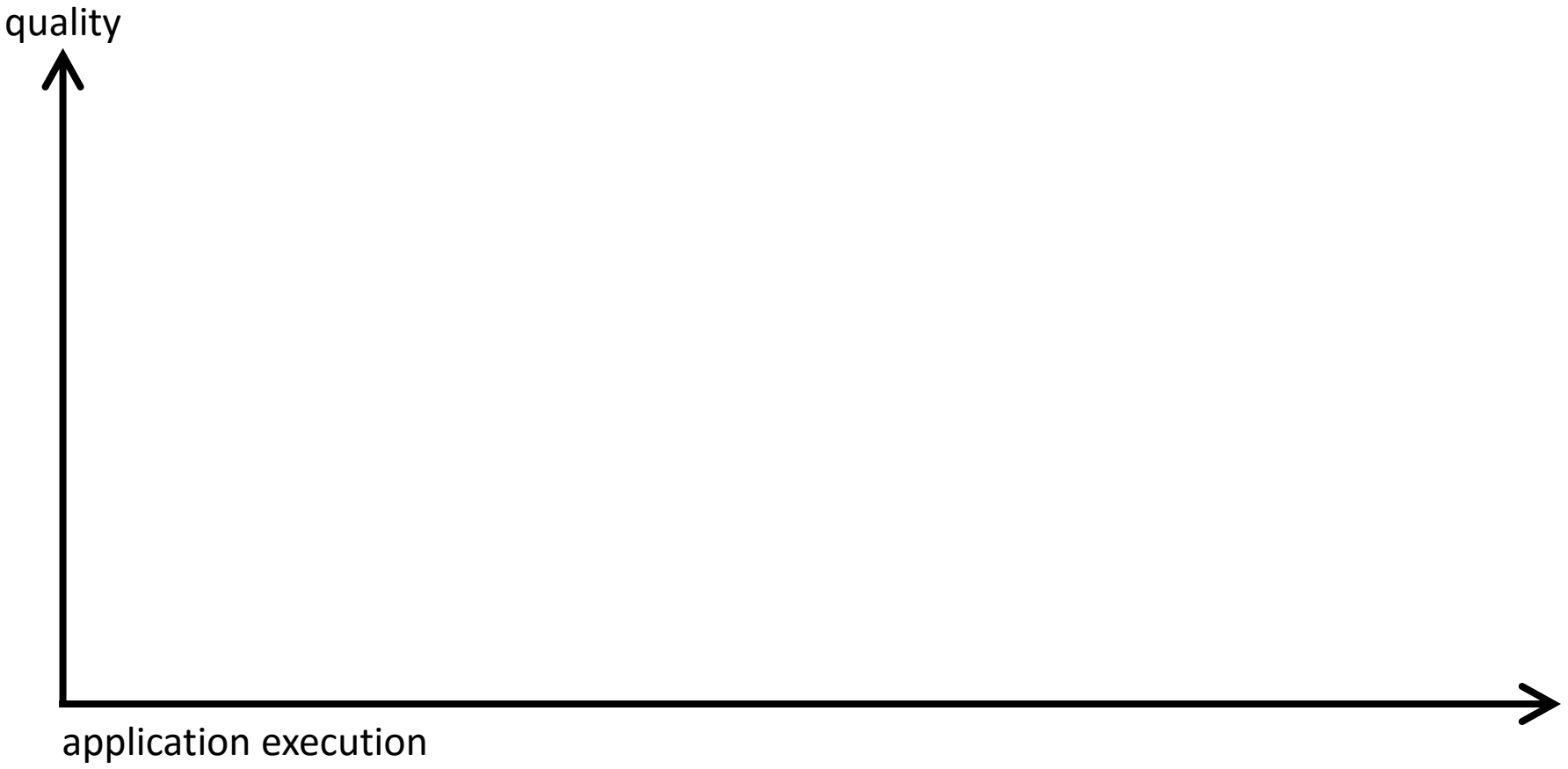


Anytime Automaton

We propose the **Anytime Automaton**:

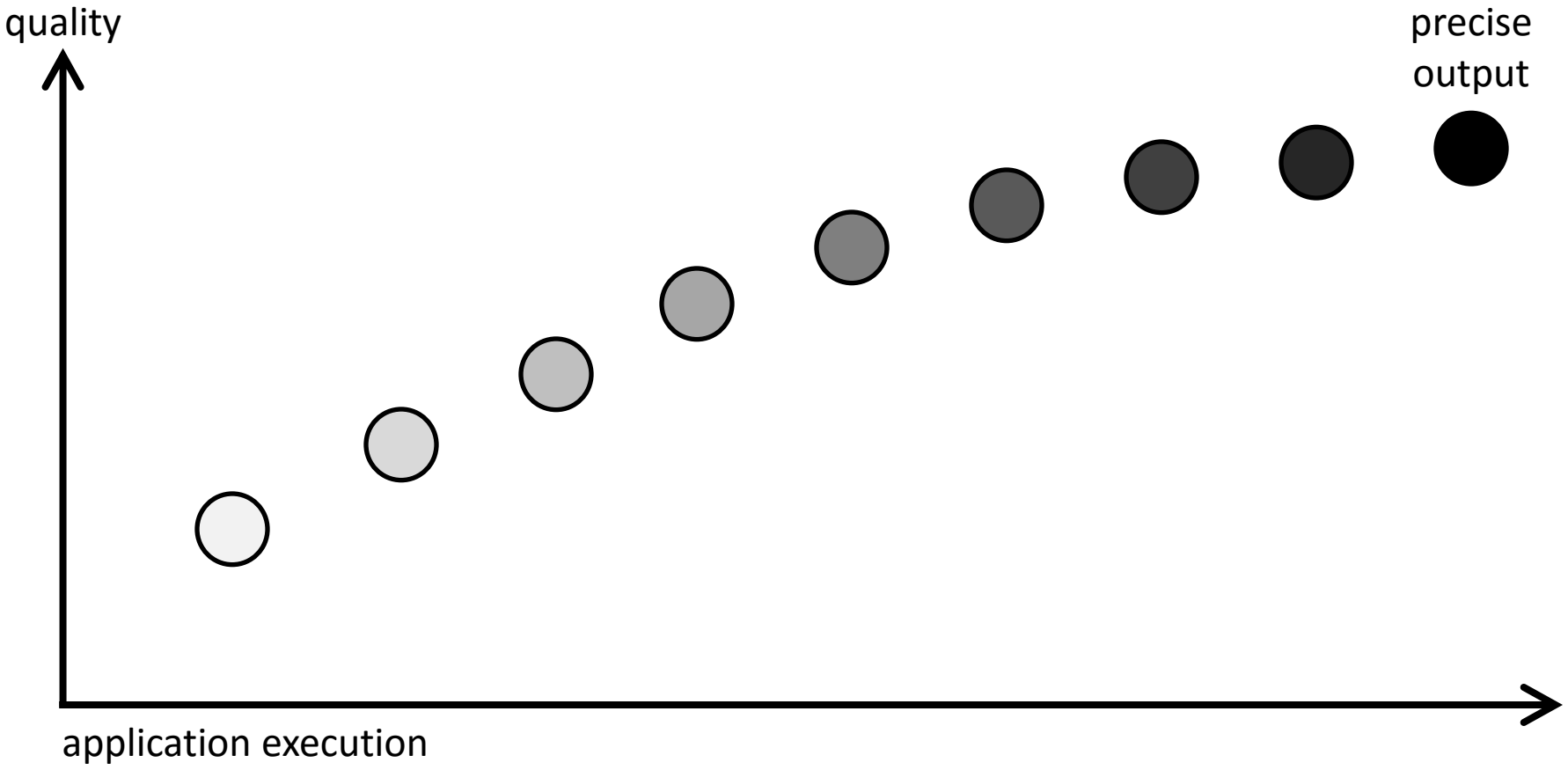
- A new computation model for approximate computing.
- Revisits and generalizes concepts from anytime (or iterative) algorithms, originally studied for real-time decision problems.
- A *recipe* for applying approximate computing techniques such that the final output is available early and improves in quality over time.

Anytime Automaton

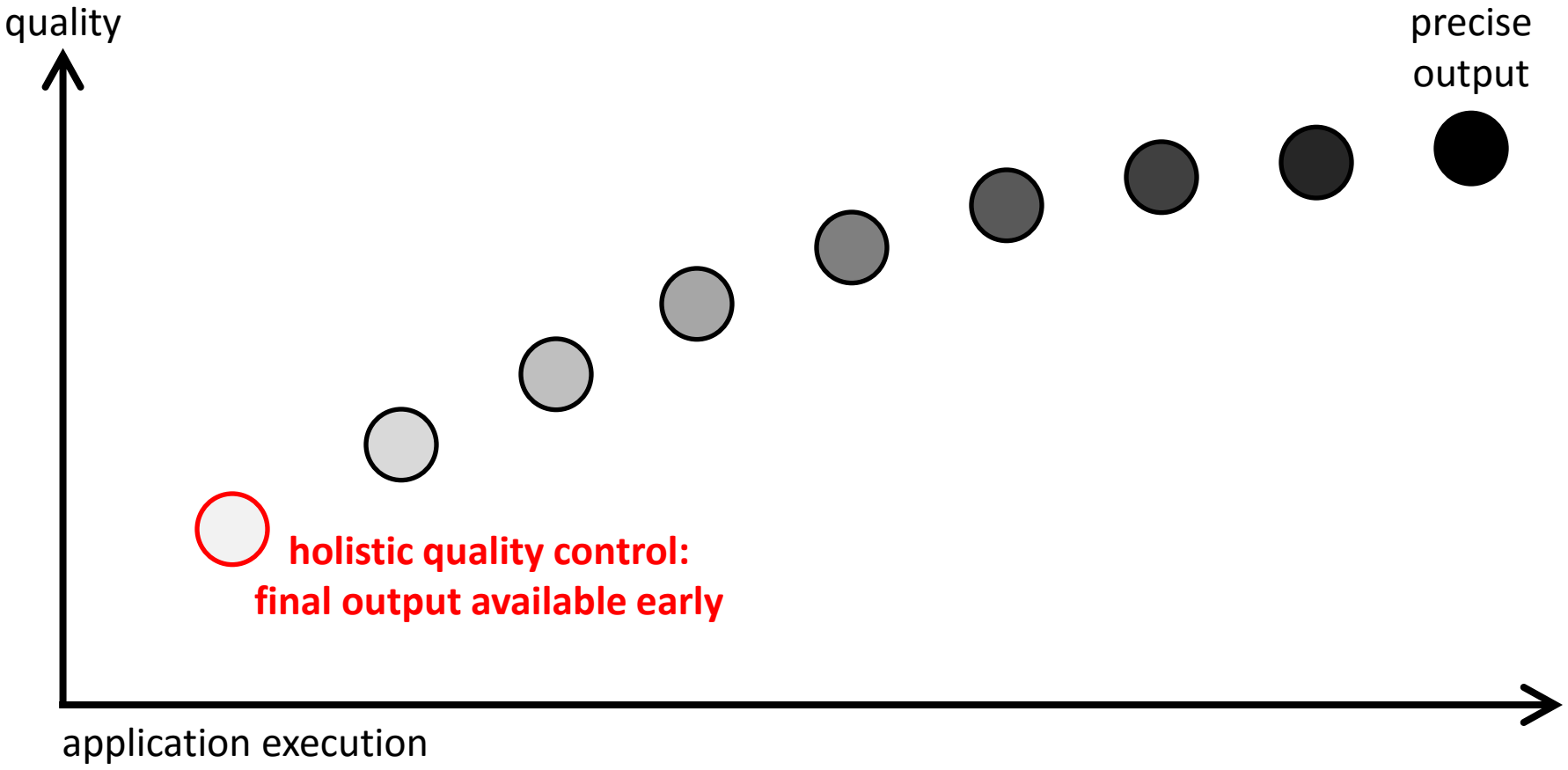


Anytime Automaton

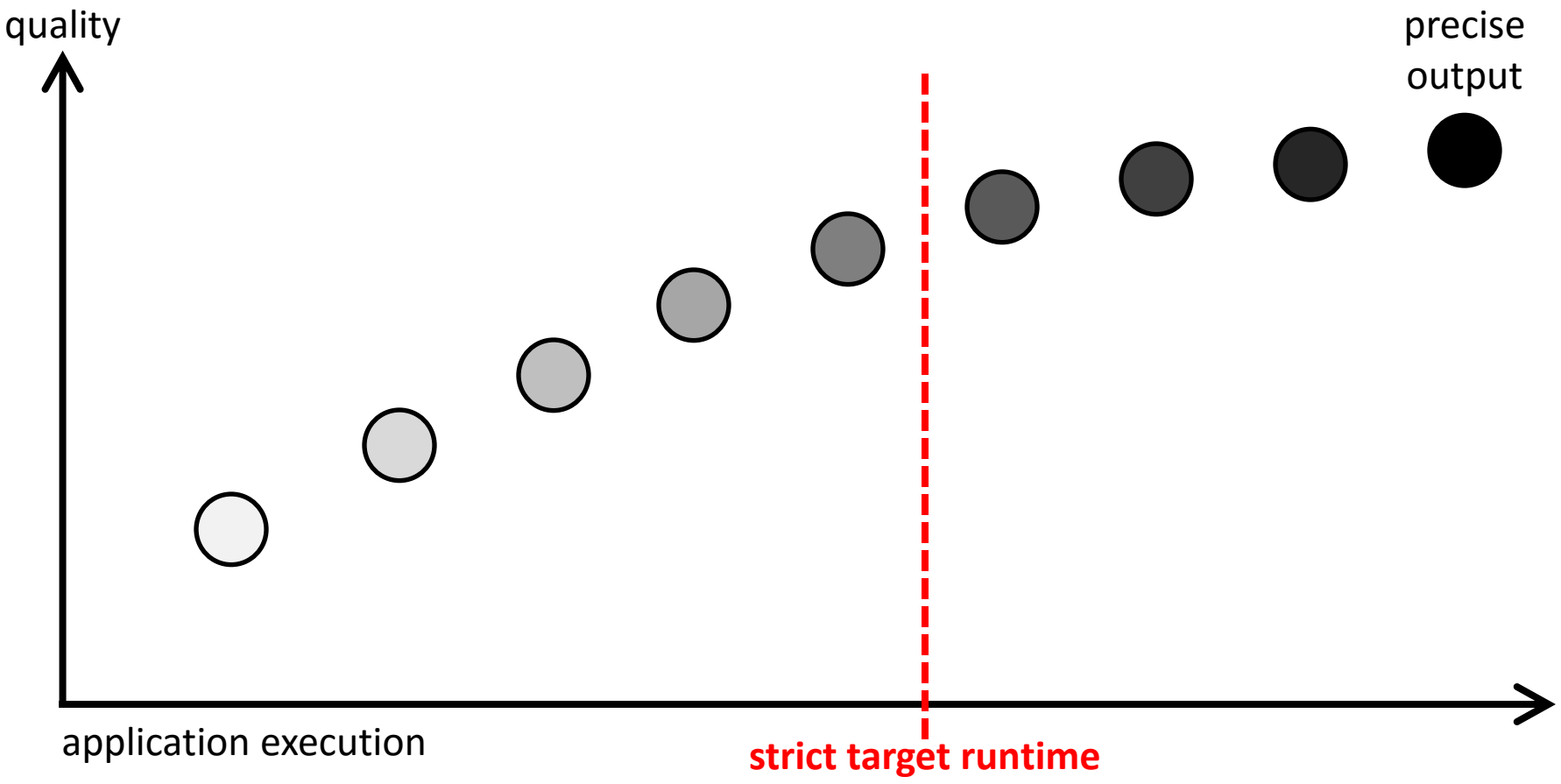
Anytime Automaton



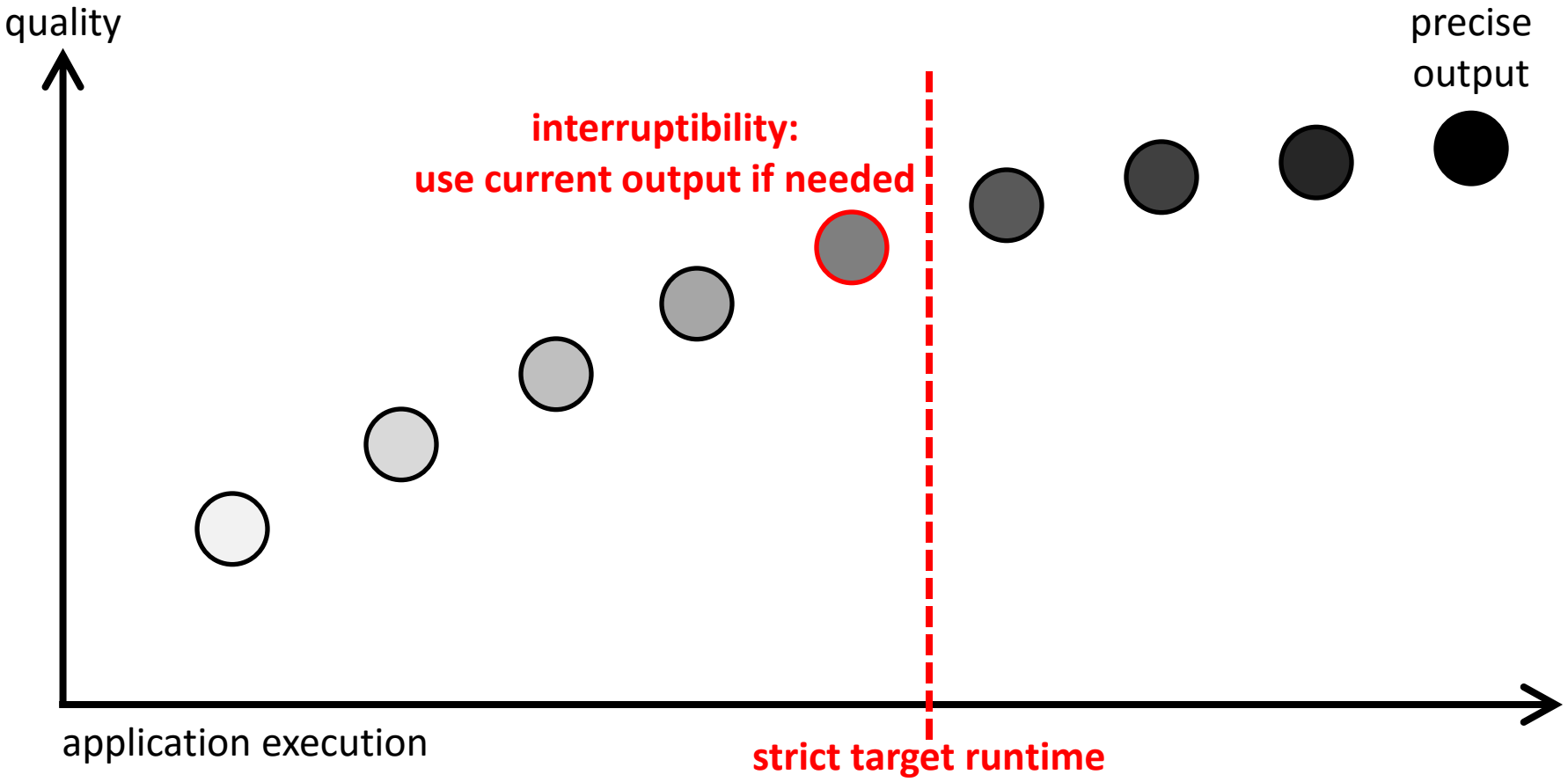
Anytime Automaton



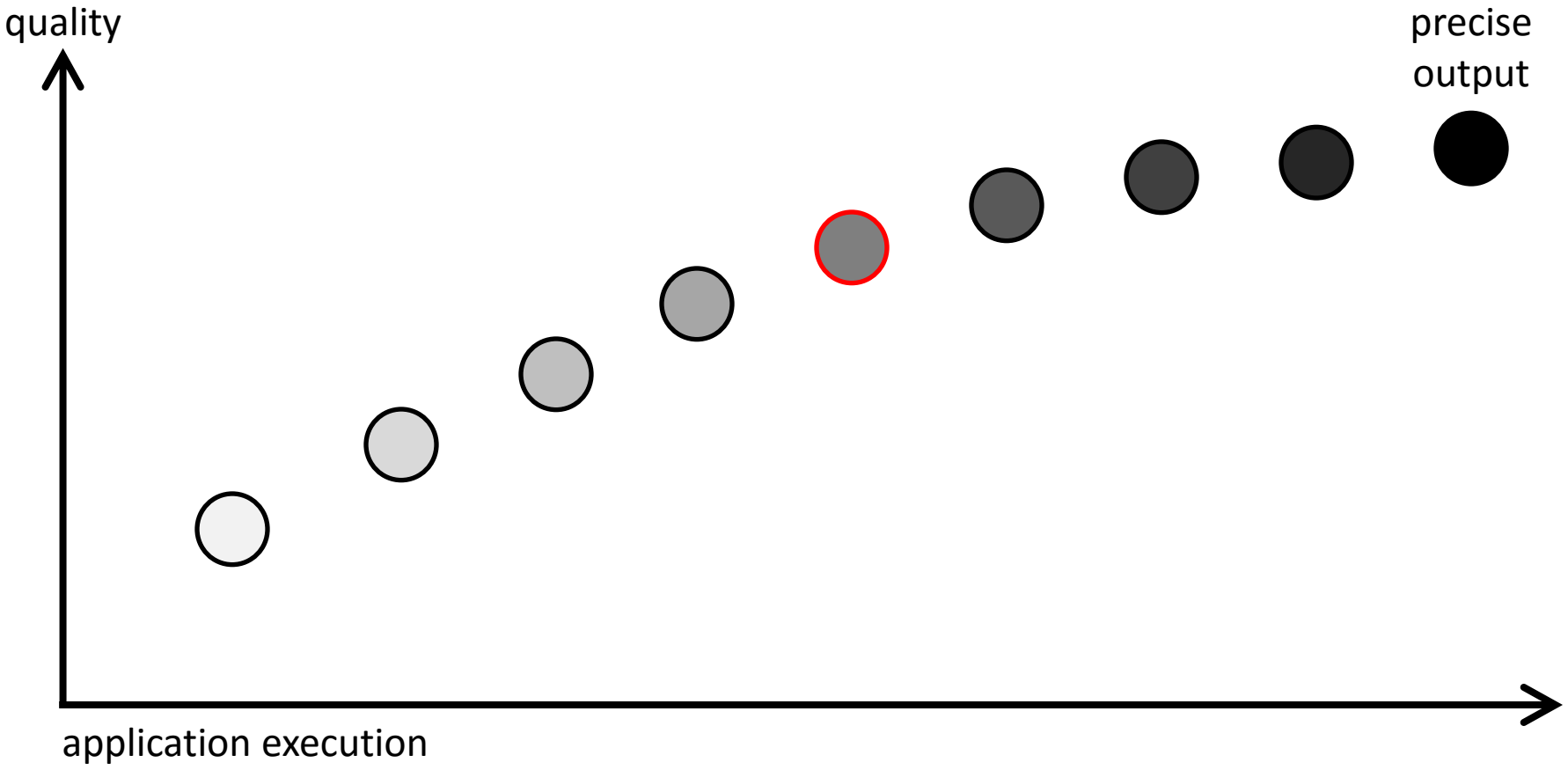
Anytime Automaton



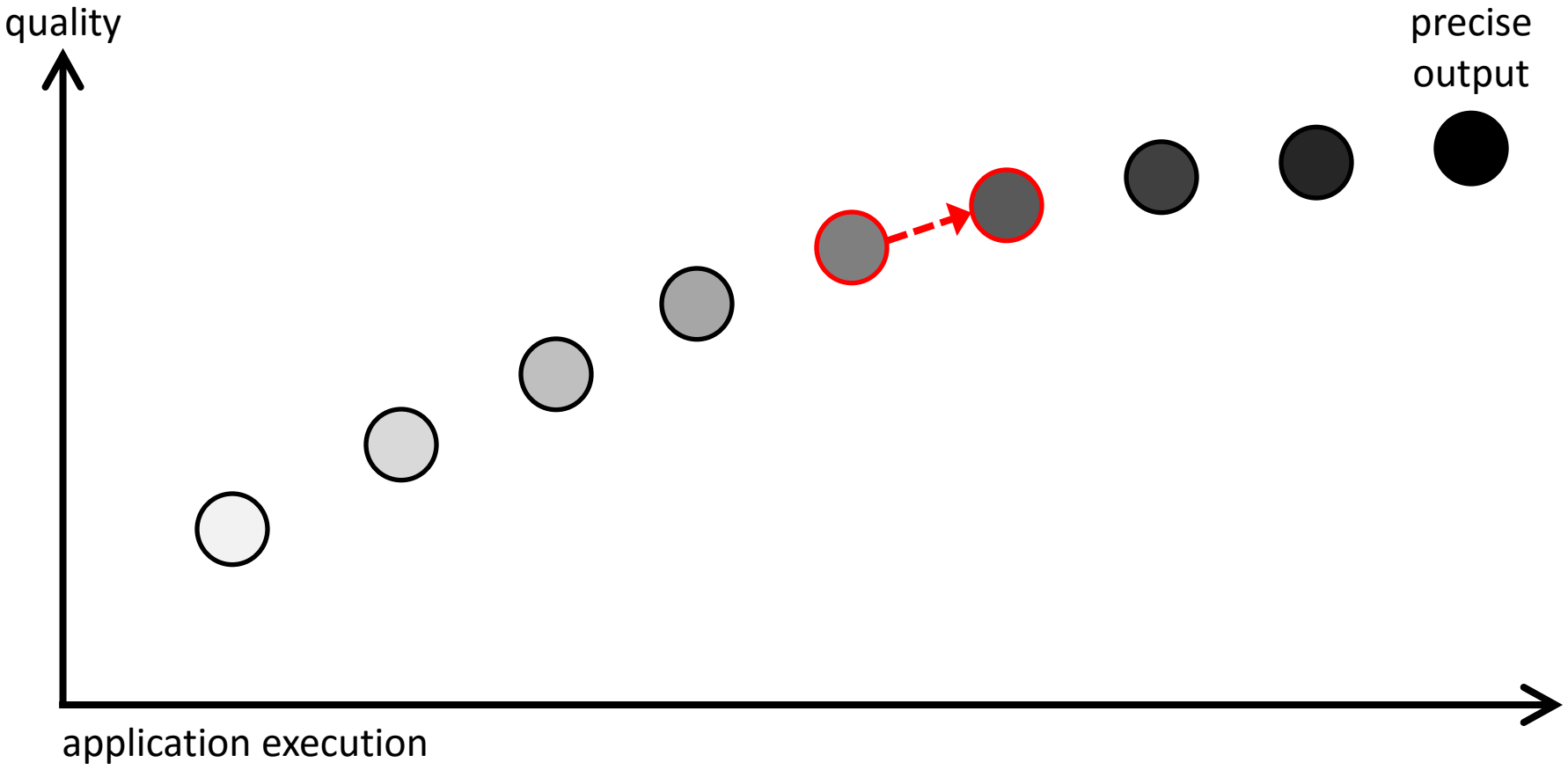
Anytime Automaton



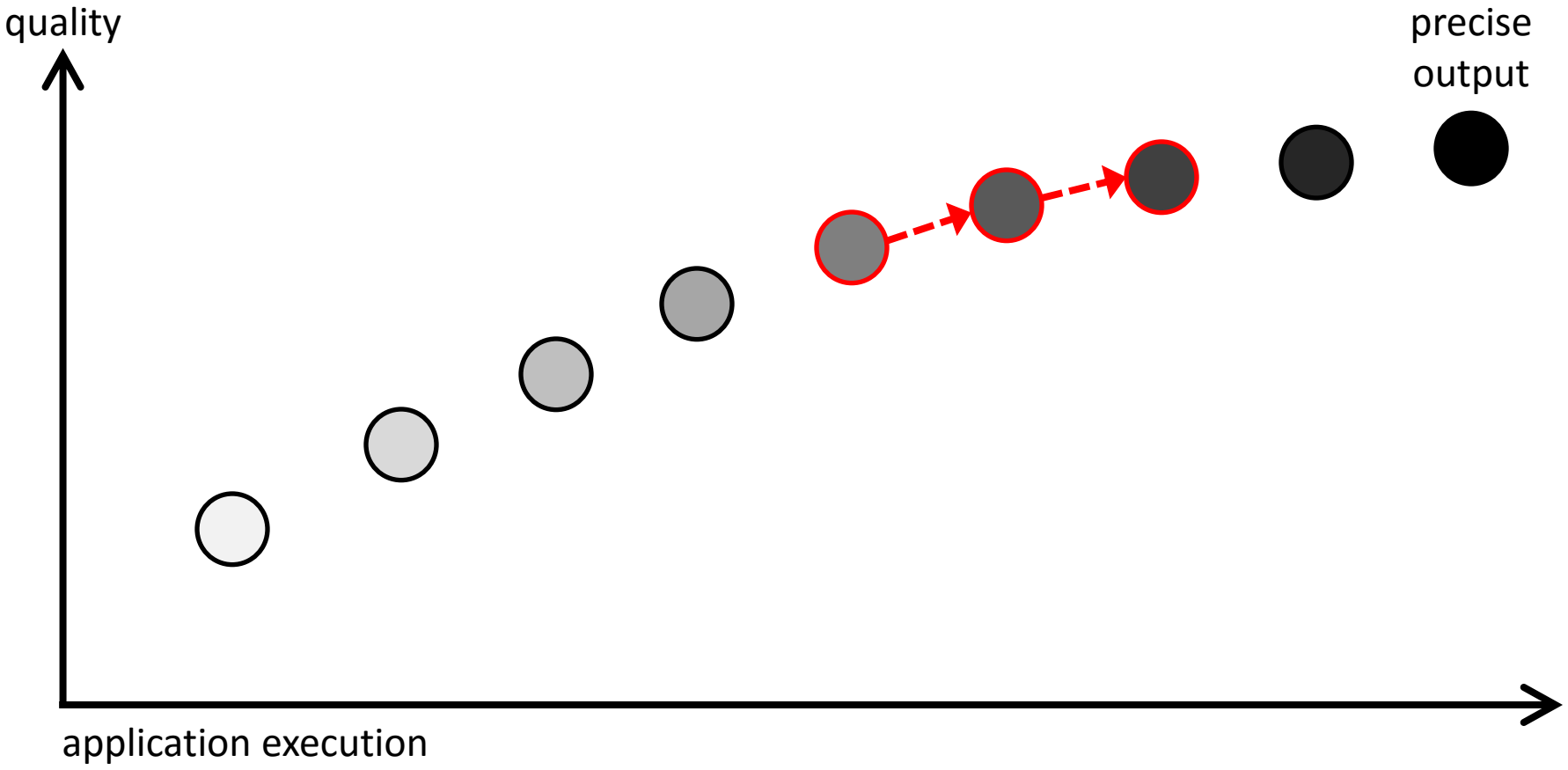
Anytime Automaton



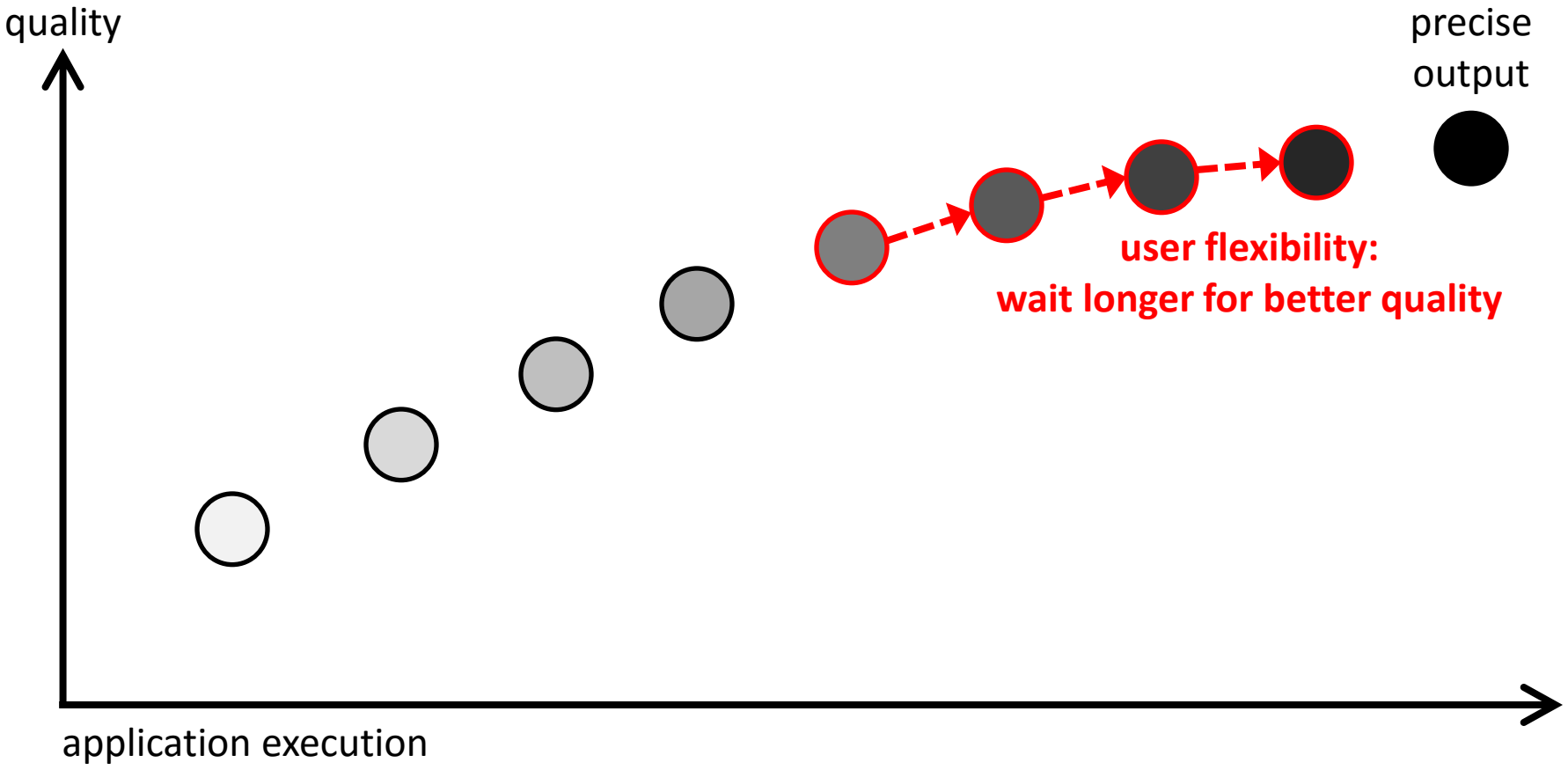
Anytime Automaton



Anytime Automaton



Anytime Automaton



Outline

Anytime Automaton

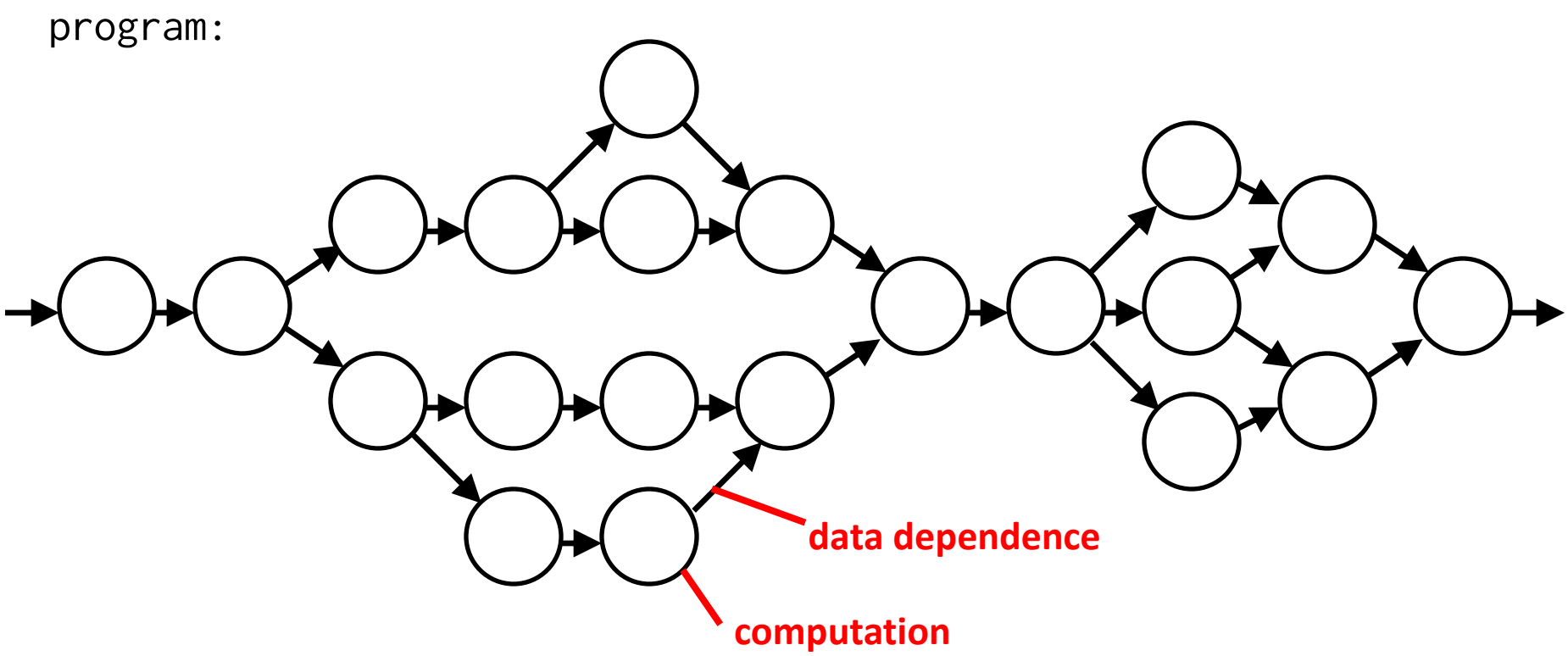
- The Model
- The Approximations

Evaluation

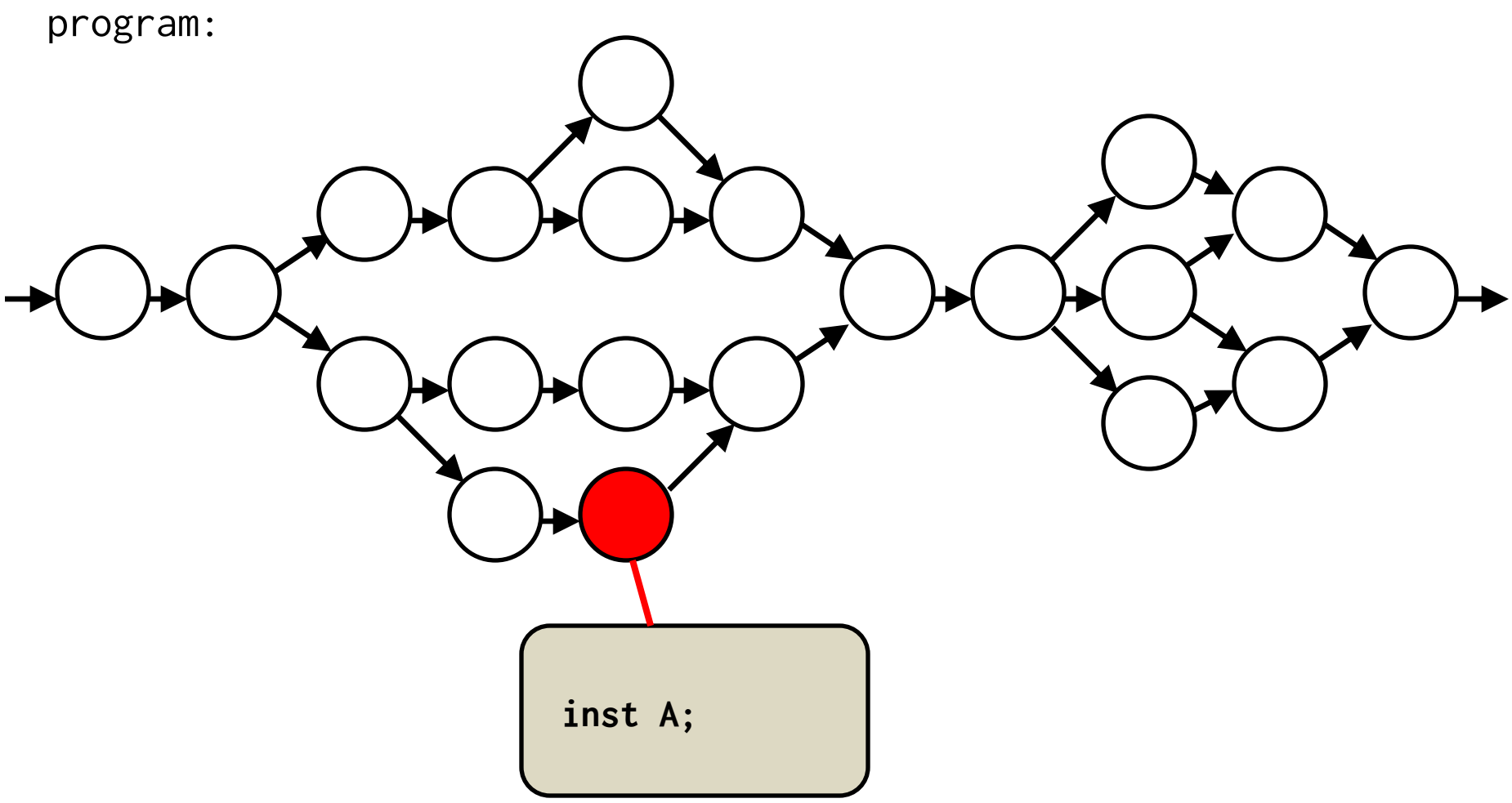
- Methodology
- Experimental Results

Conclusion

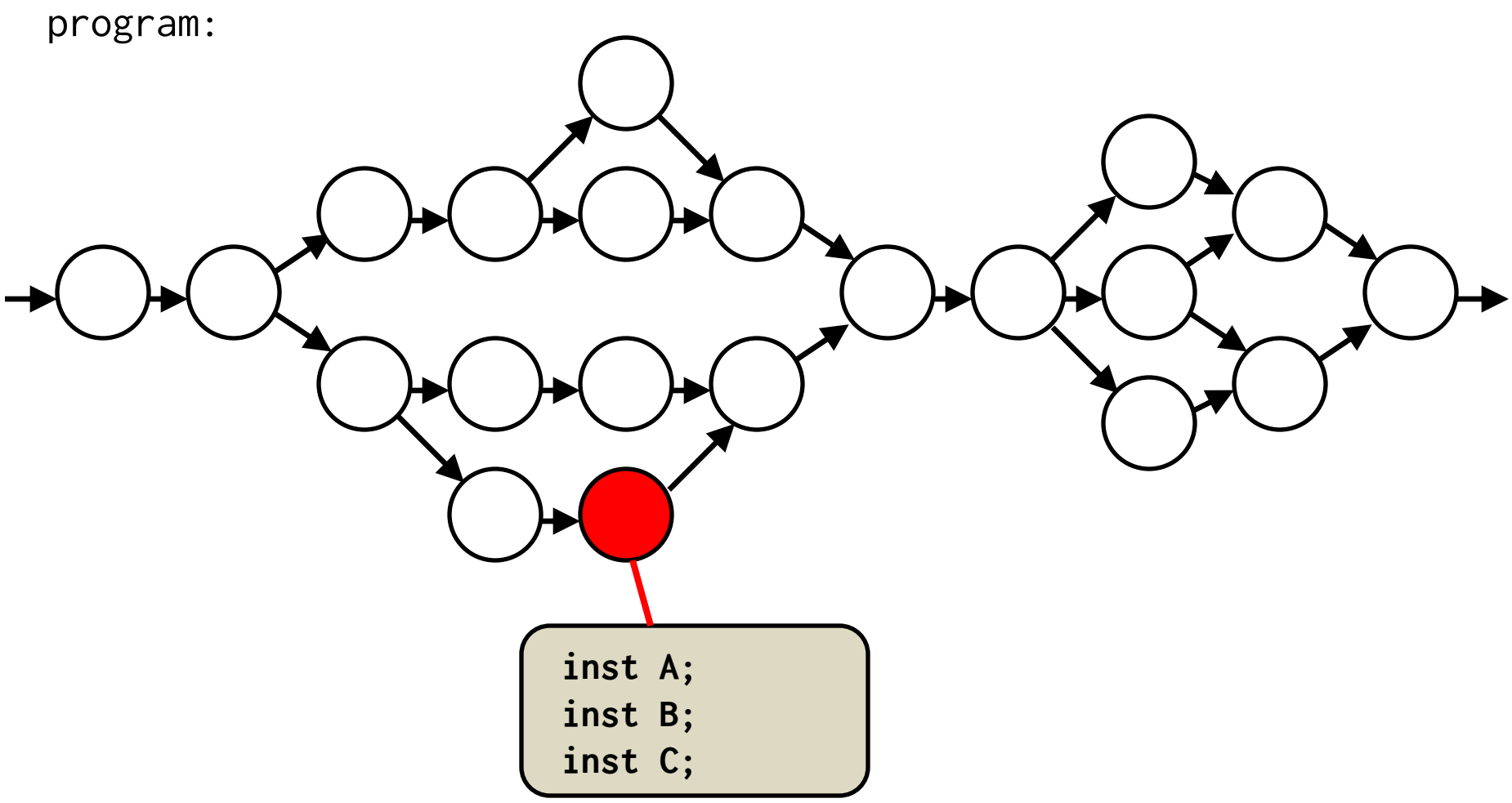
Anytime Automaton



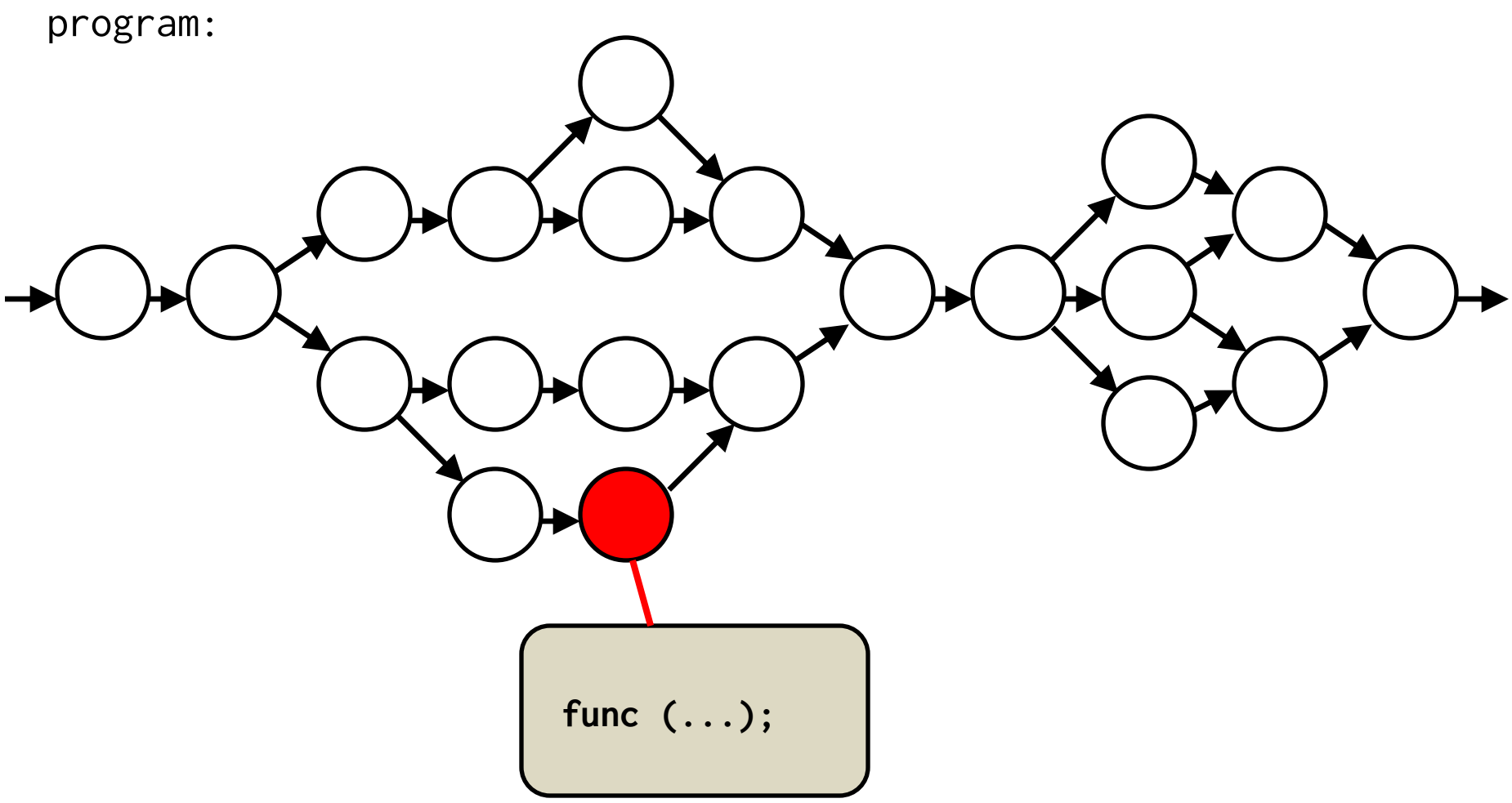
Anytime Automaton



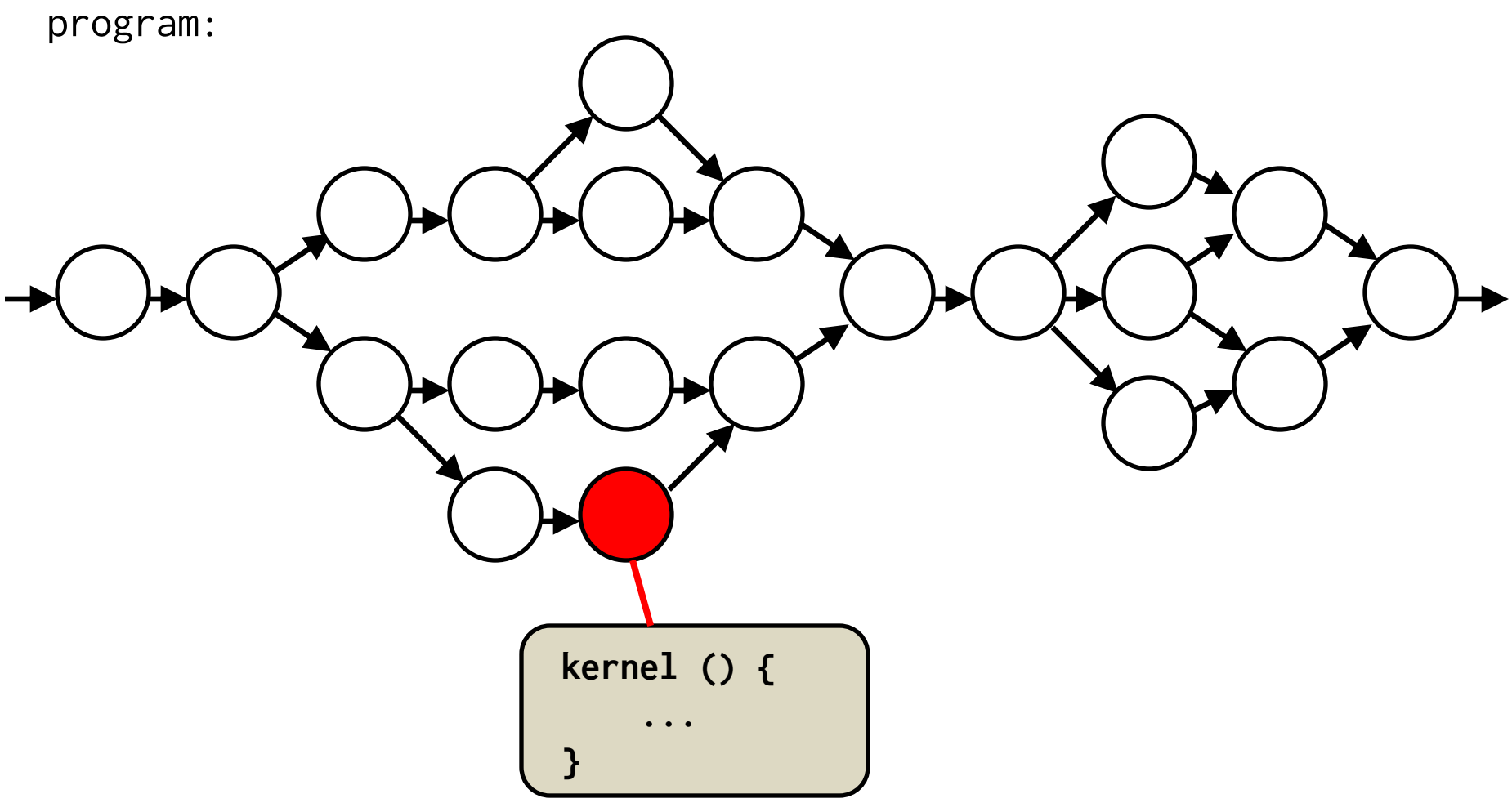
Anytime Automaton



Anytime Automaton

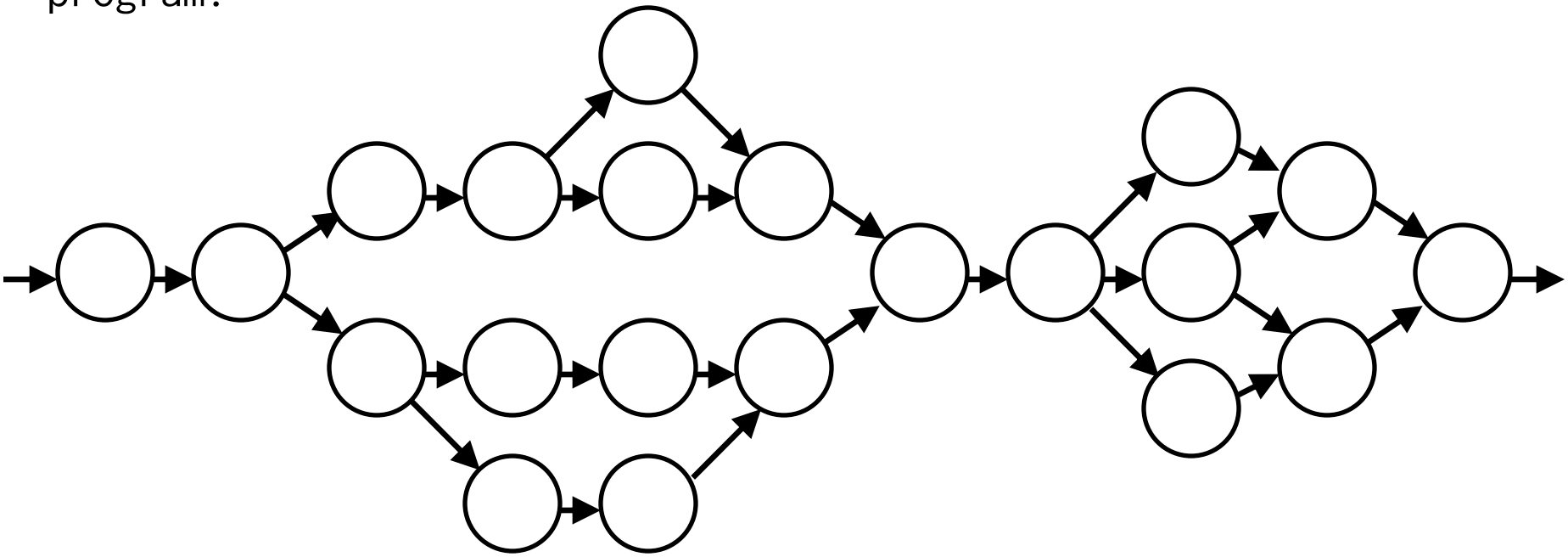


Anytime Automaton

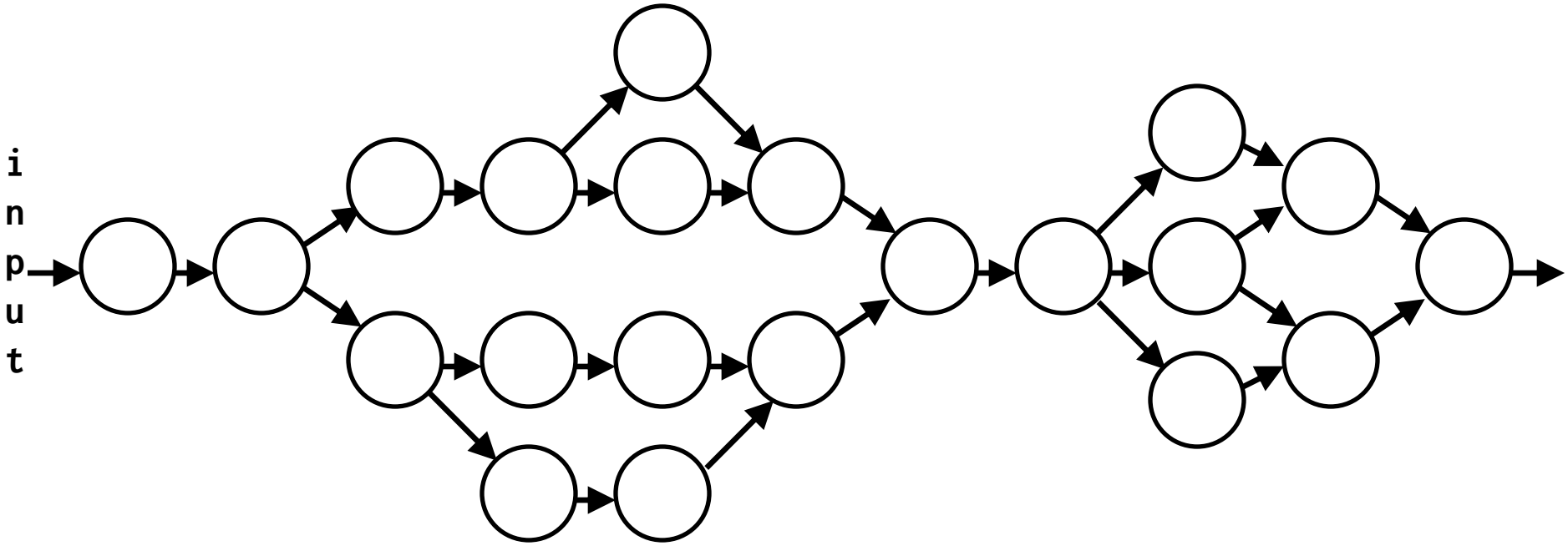


Dataflow Model

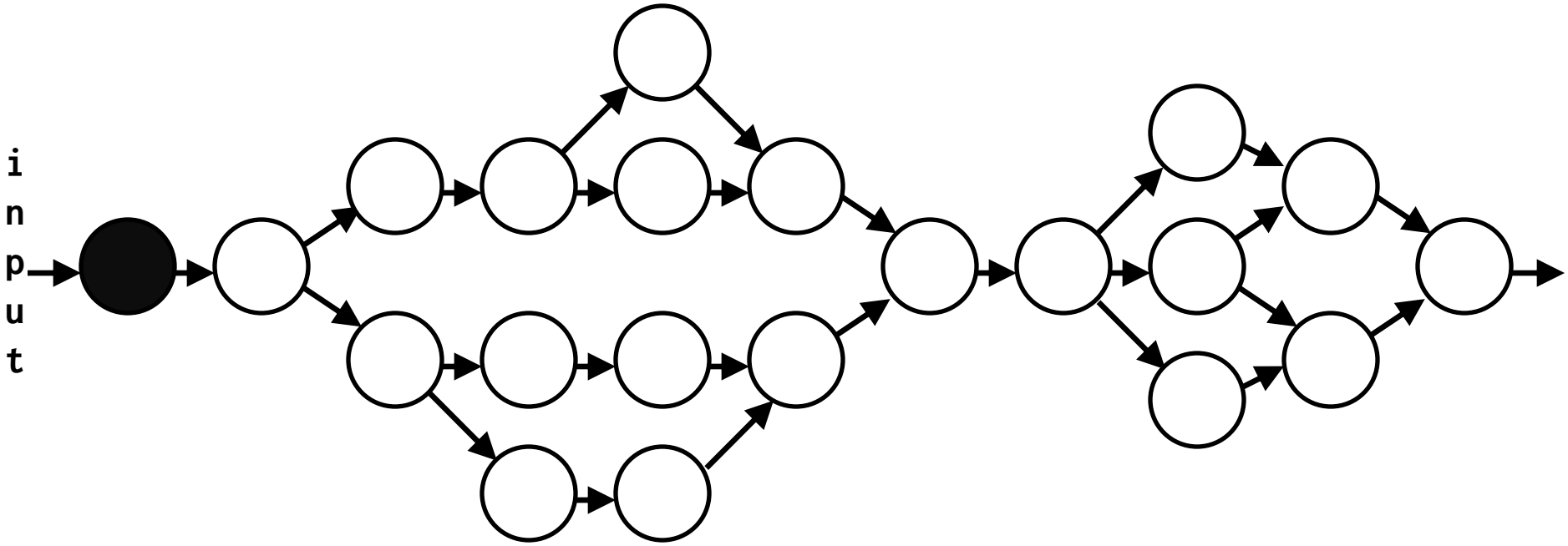
program:



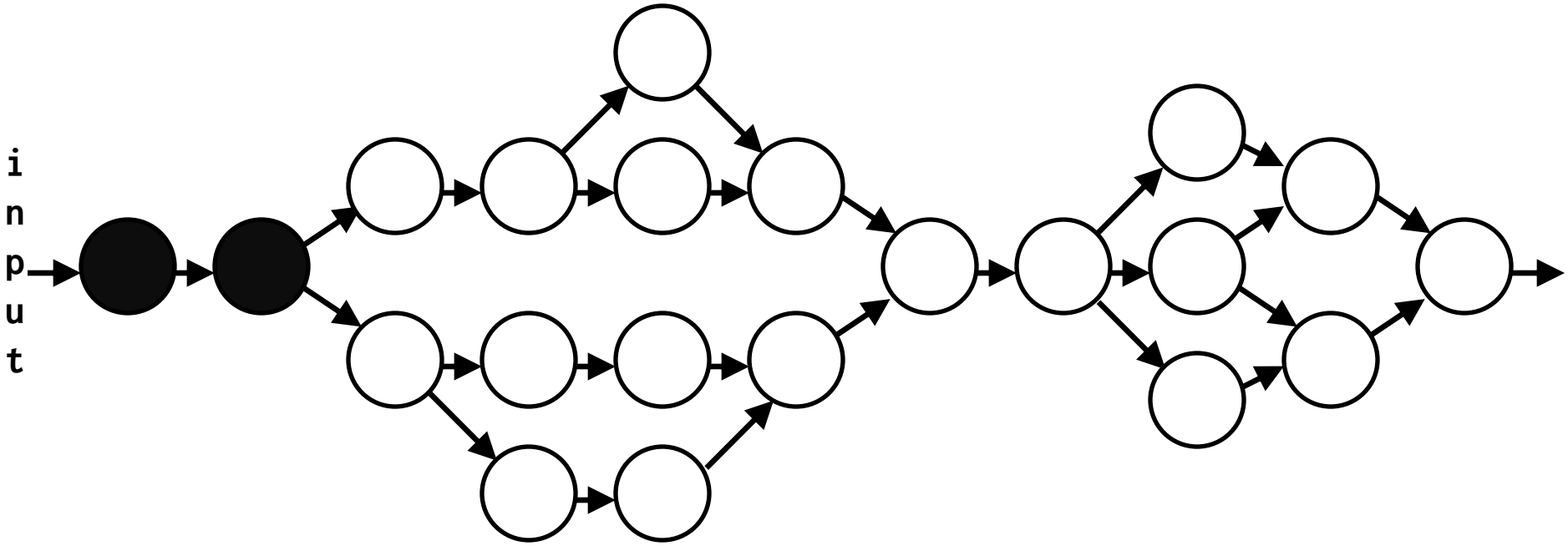
Dataflow Model



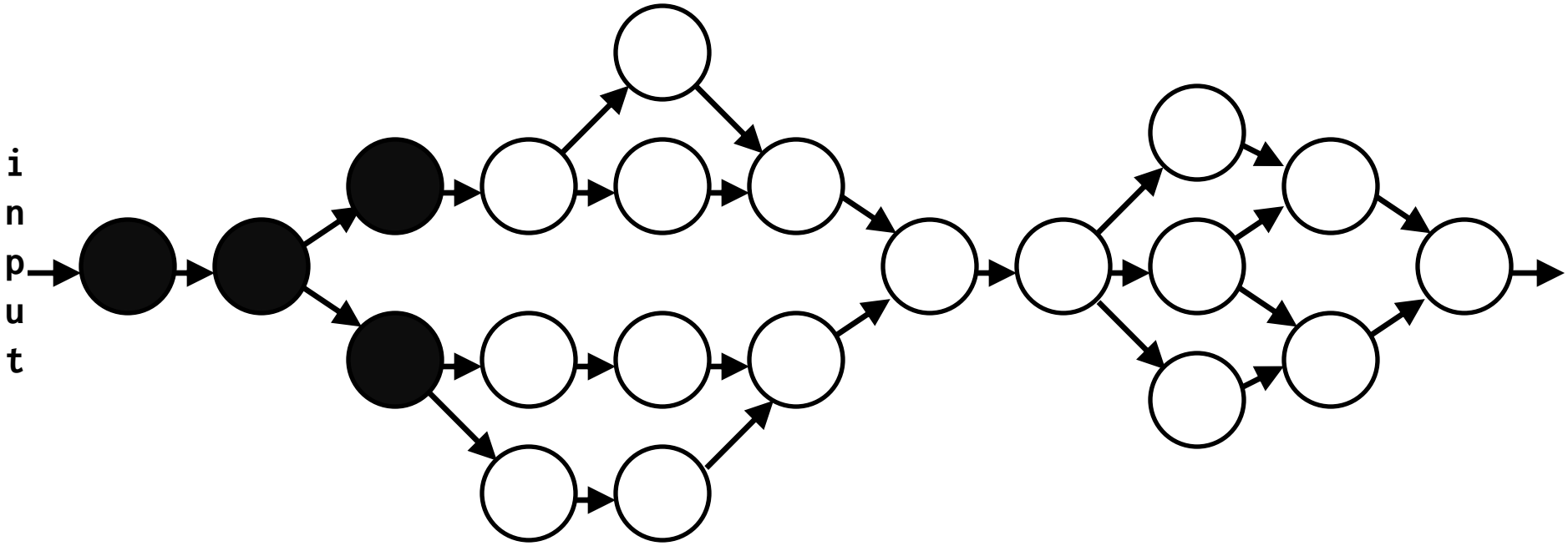
Dataflow Model



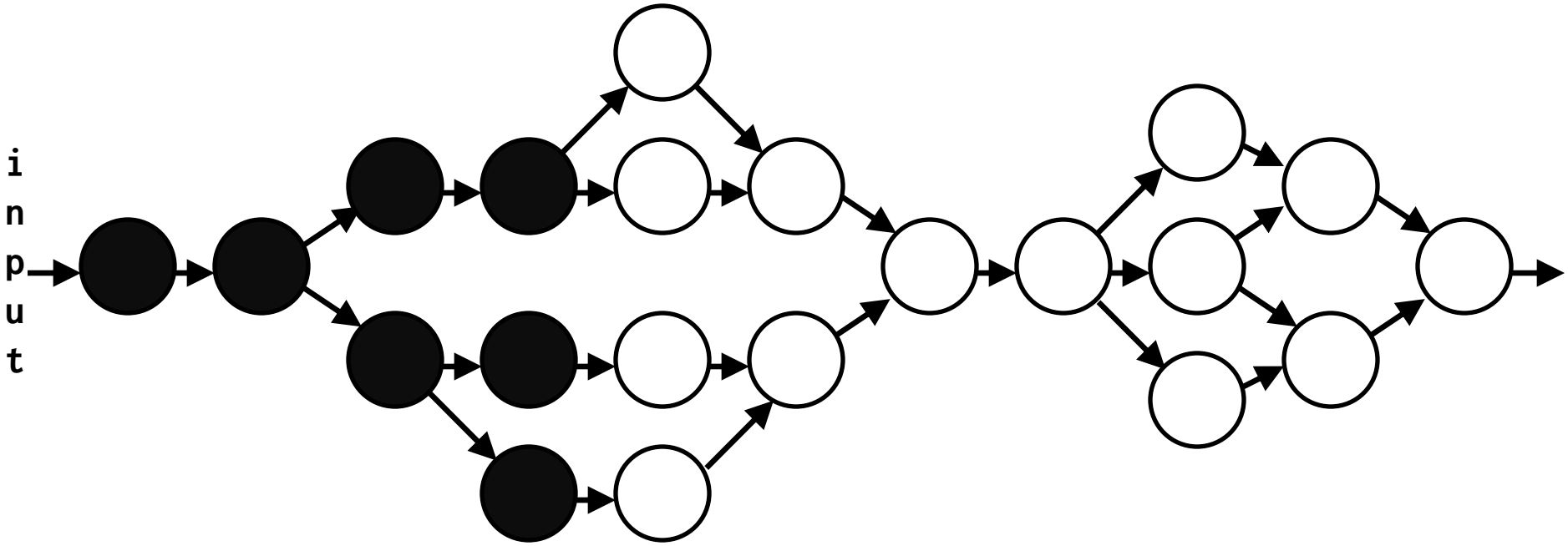
Dataflow Model



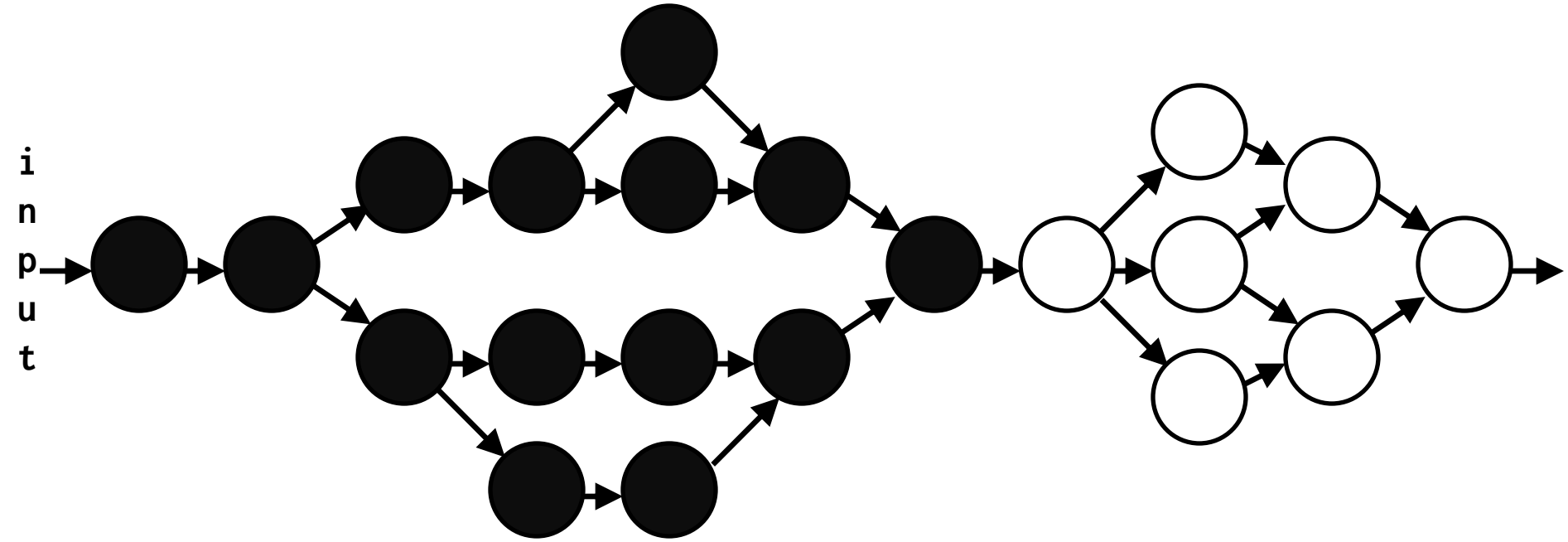
Dataflow Model



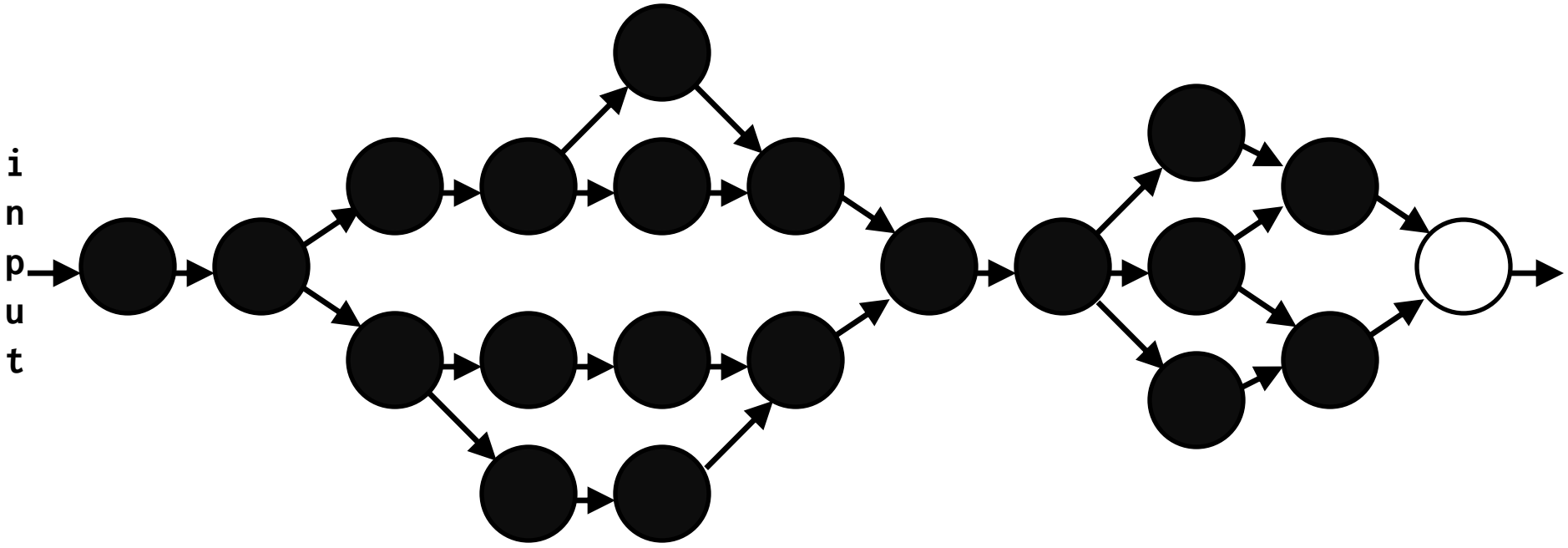
Dataflow Model



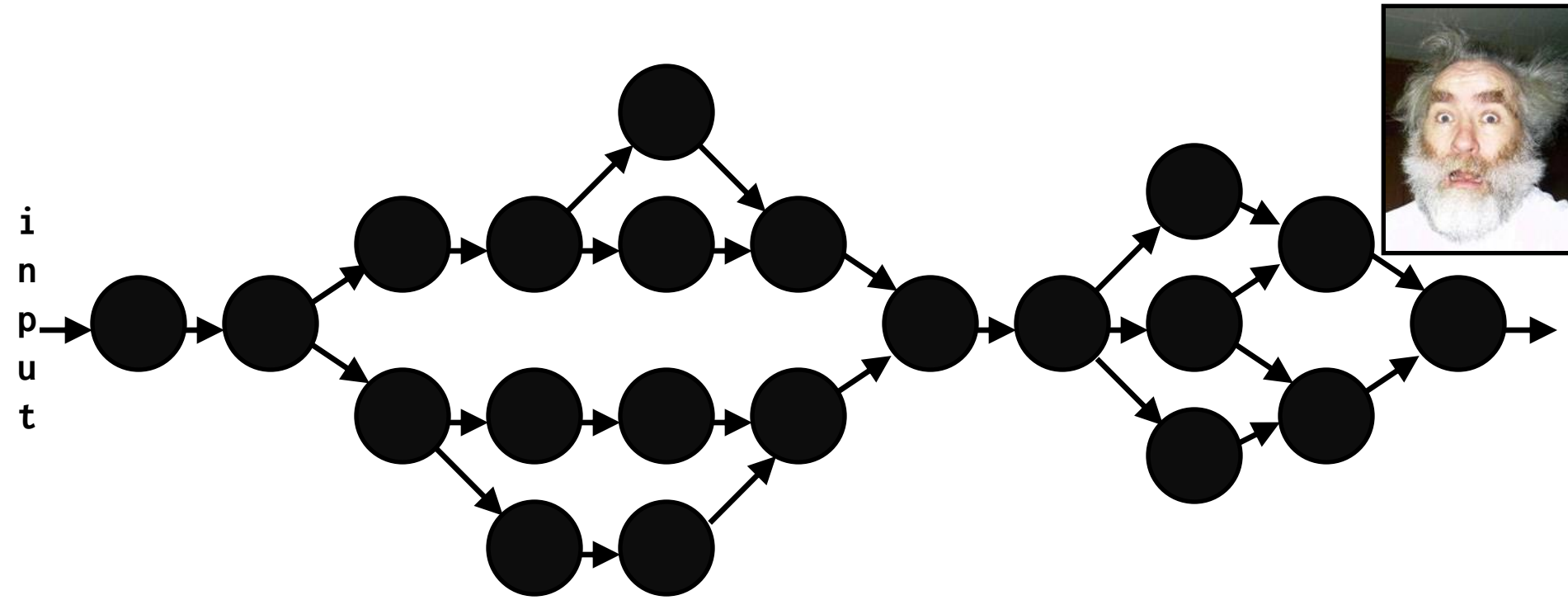
Dataflow Model



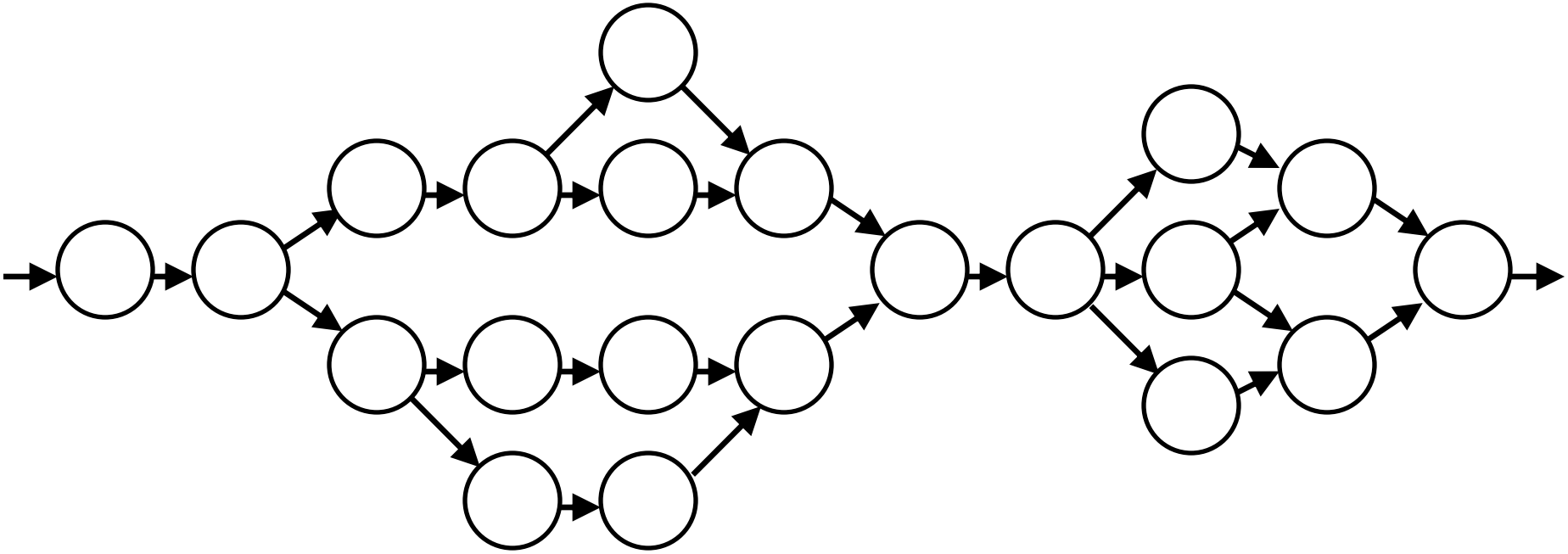
Dataflow Model



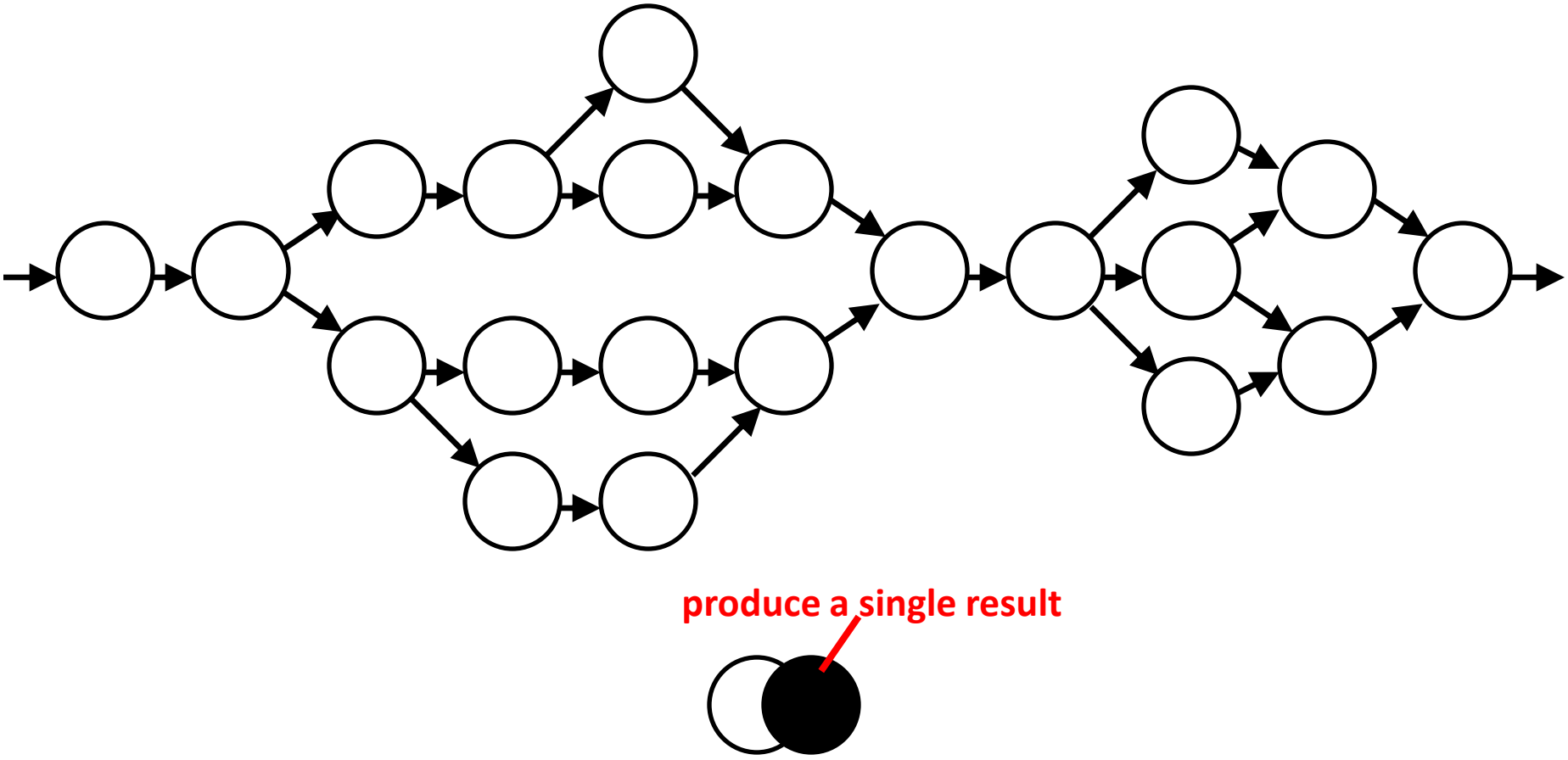
Dataflow Model



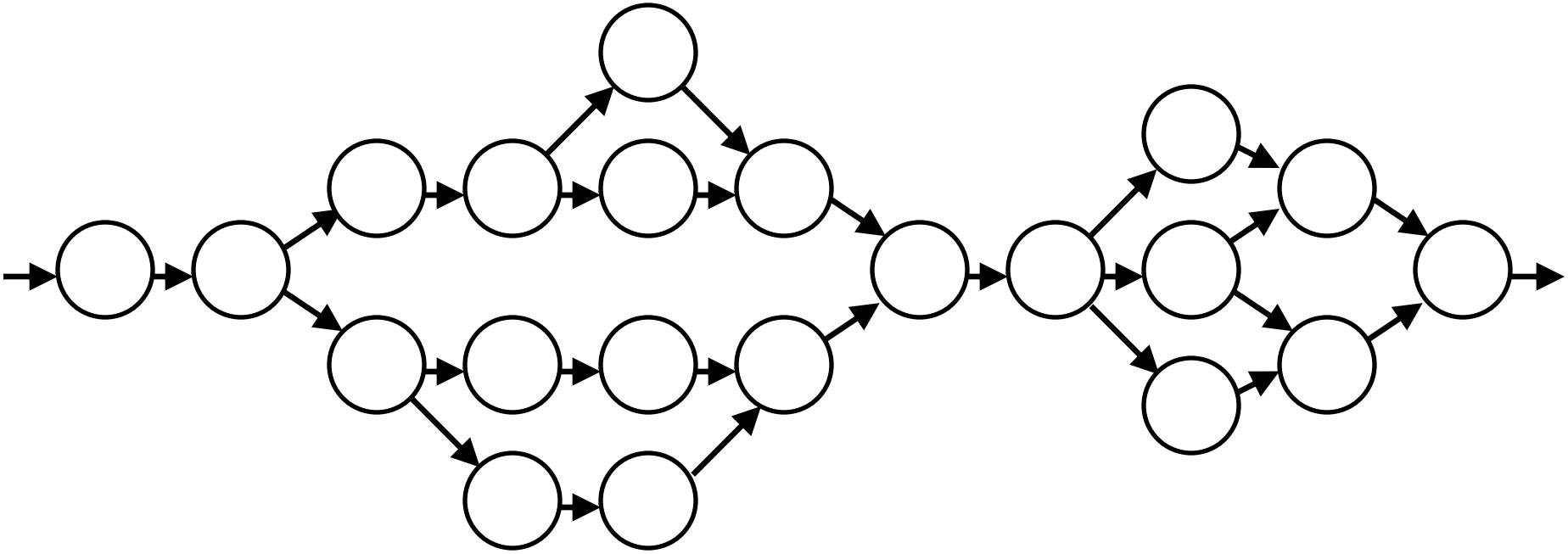
Anytime Automaton



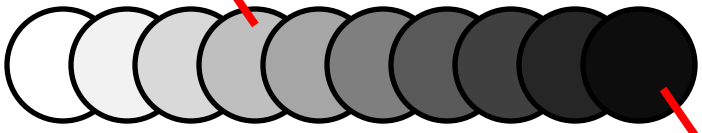
Anytime Automaton



Anytime Automaton



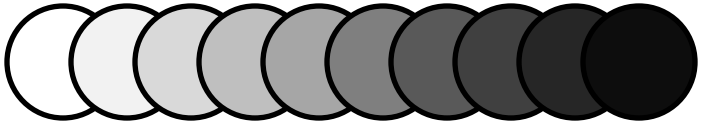
produce multiple approx *versions* of result (i.e., anytime)



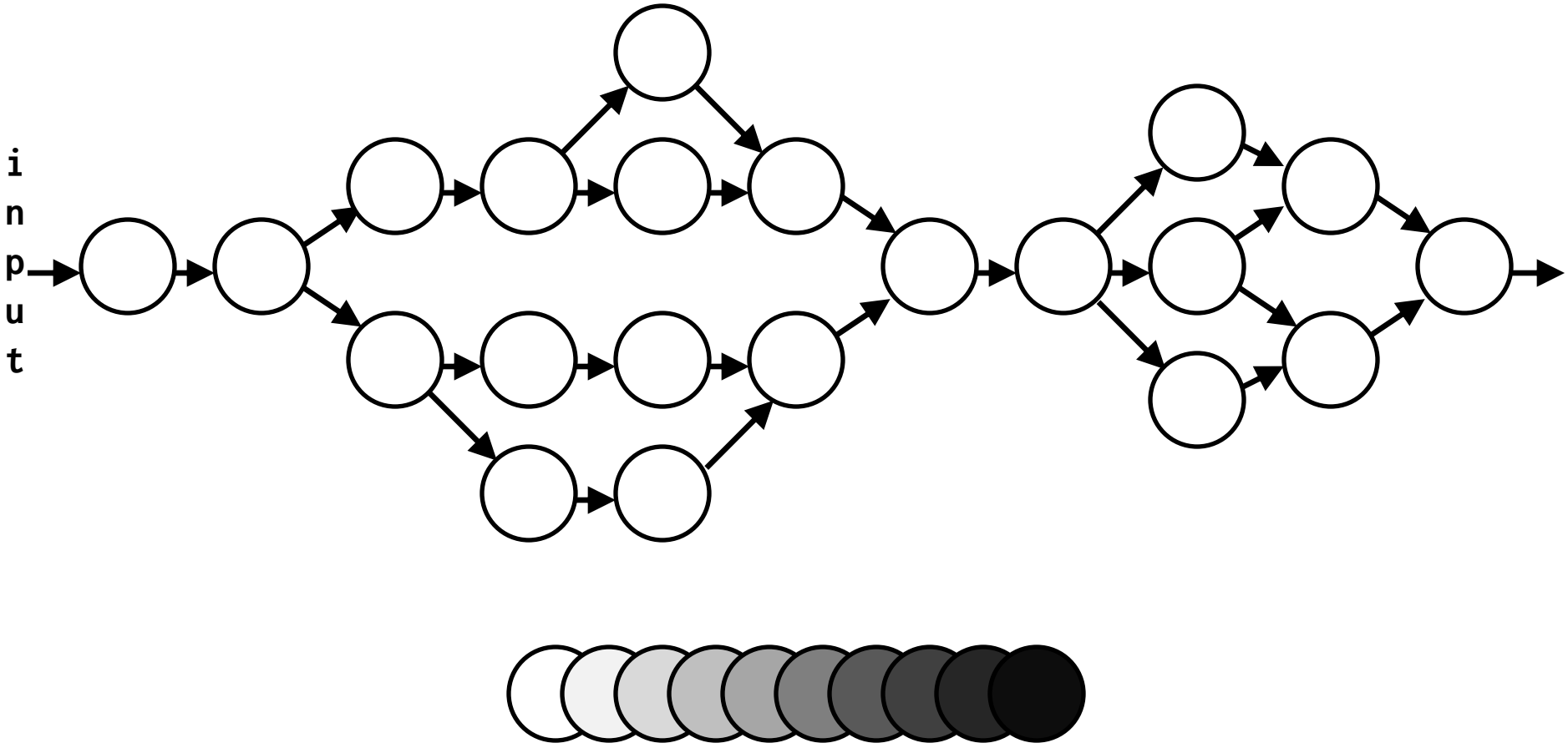
precise version

Anytime Automaton

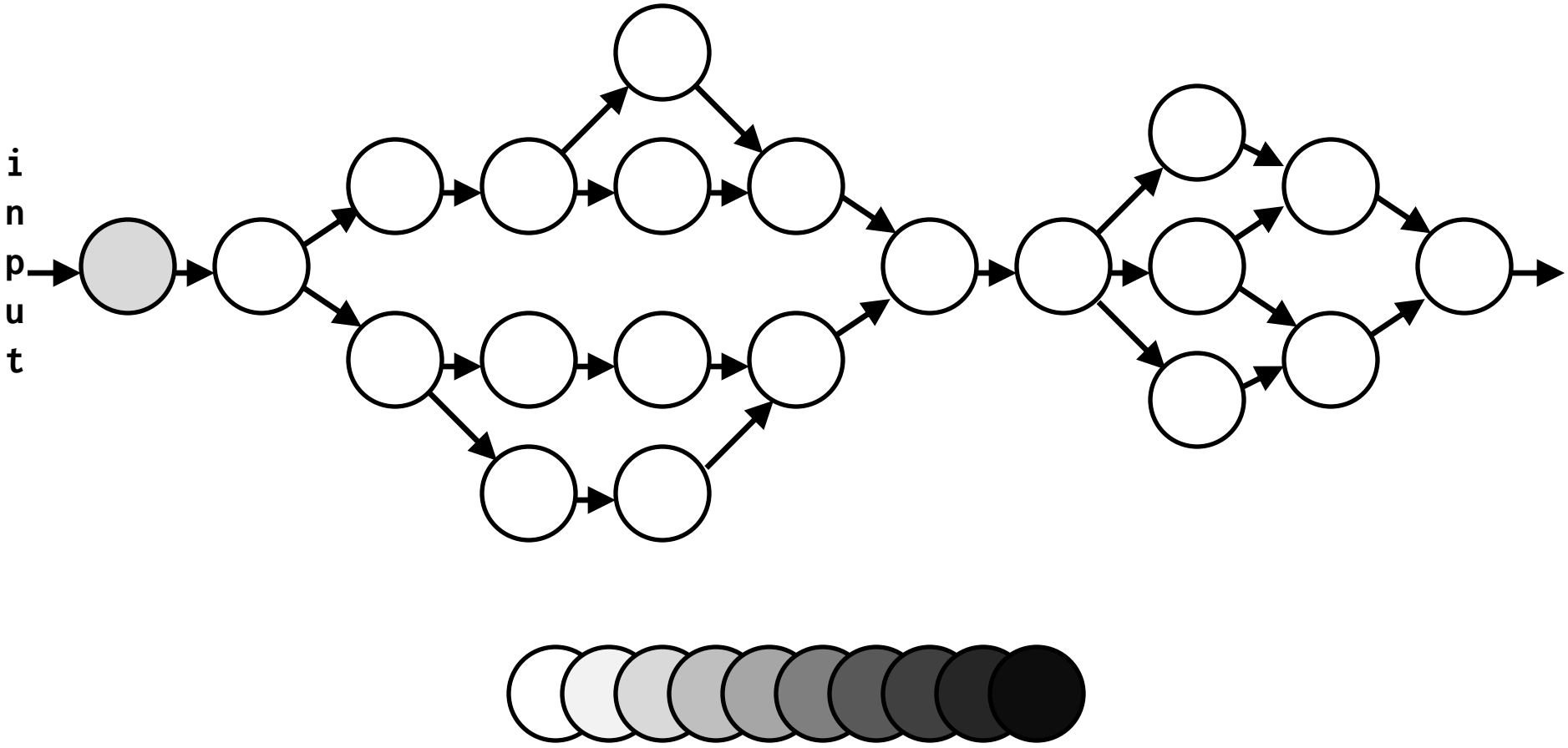
***Data Diffusion* Model**



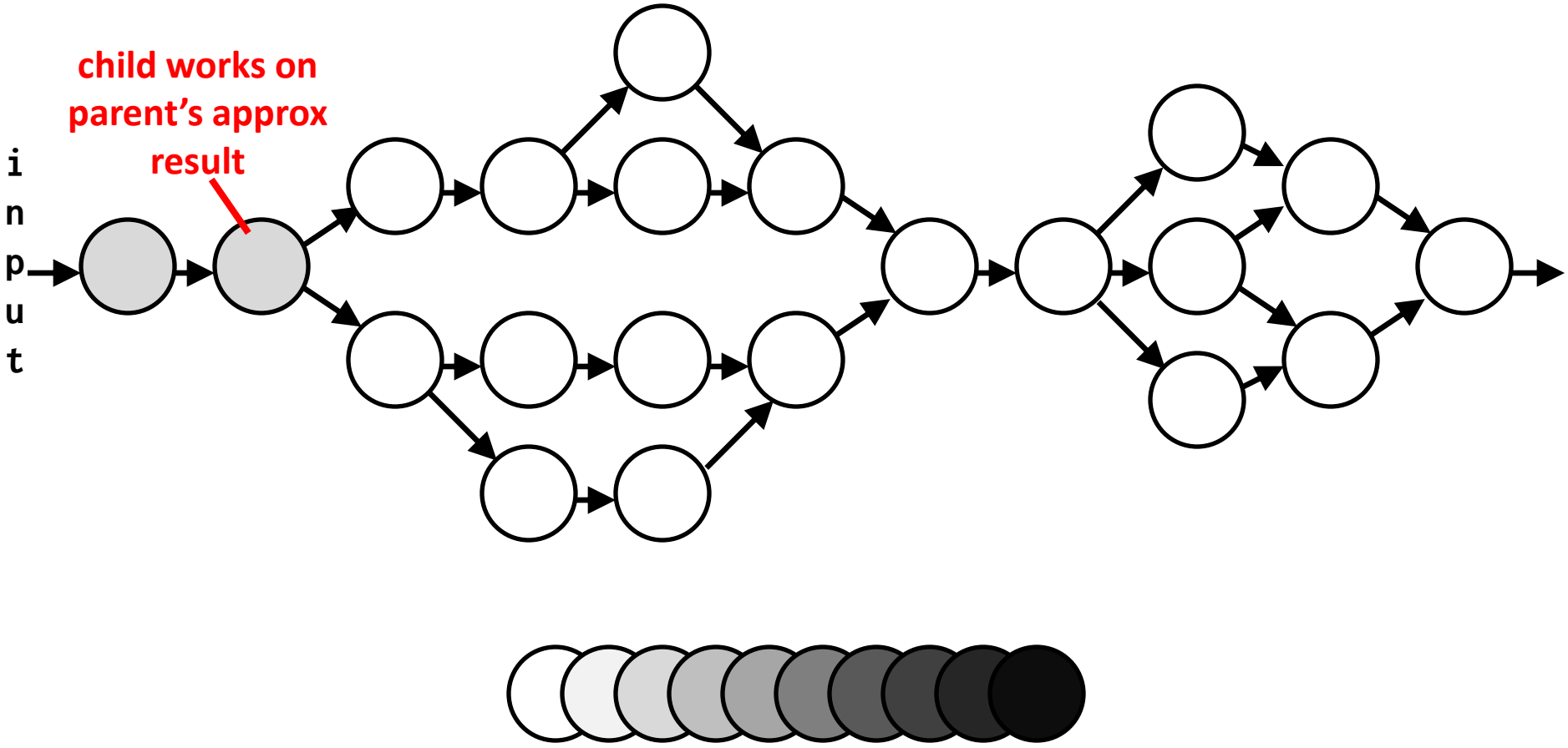
Anytime Automaton



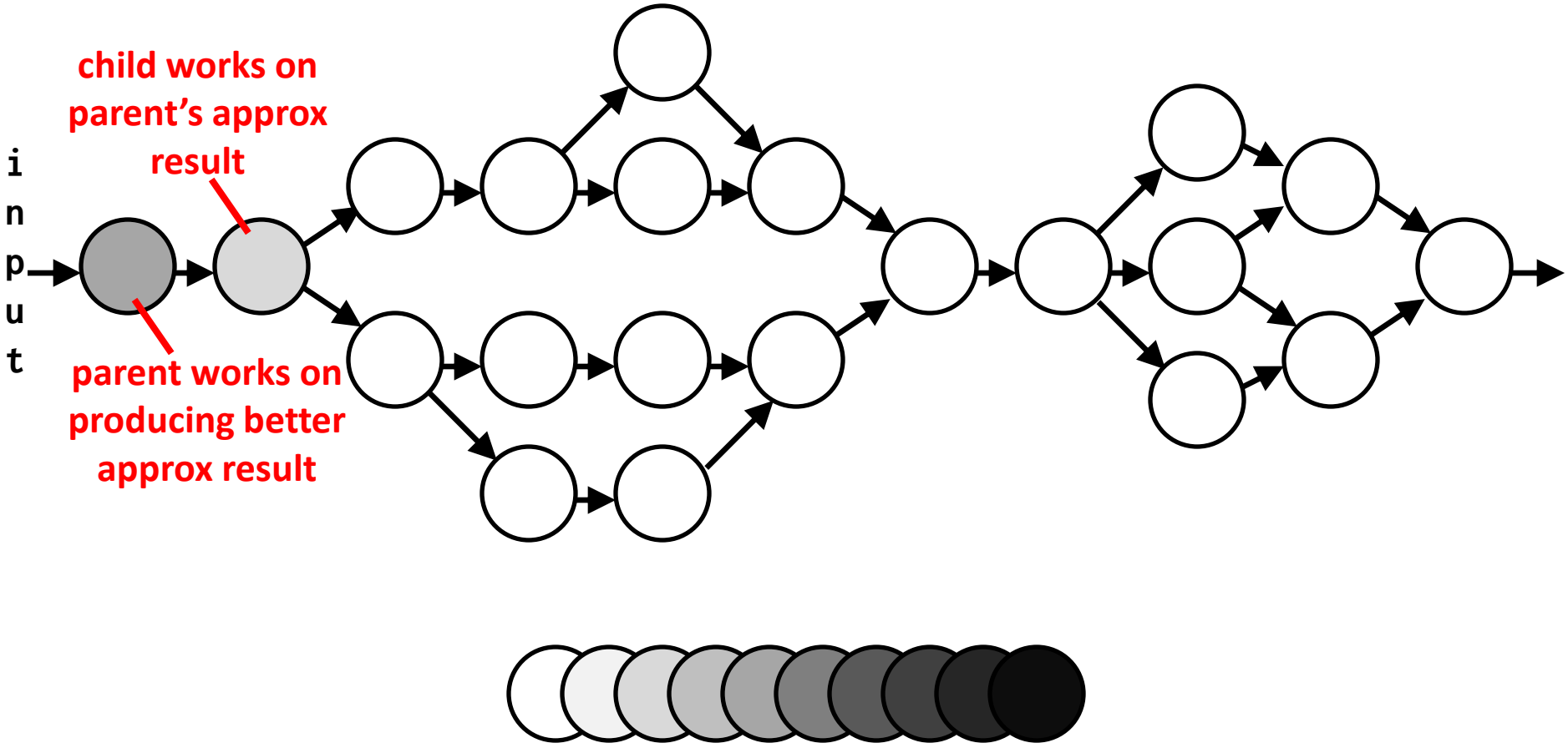
Anytime Automaton



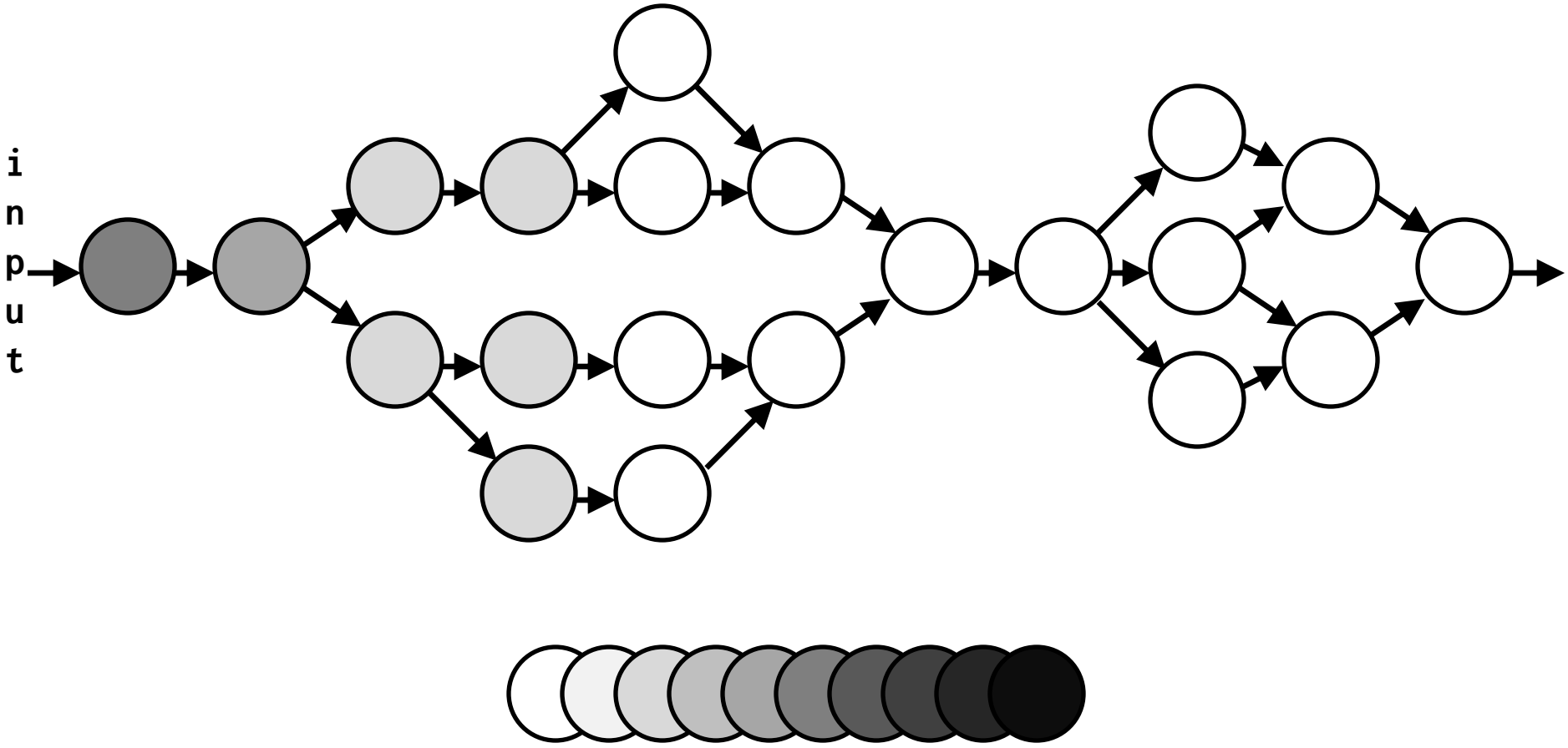
Anytime Automaton



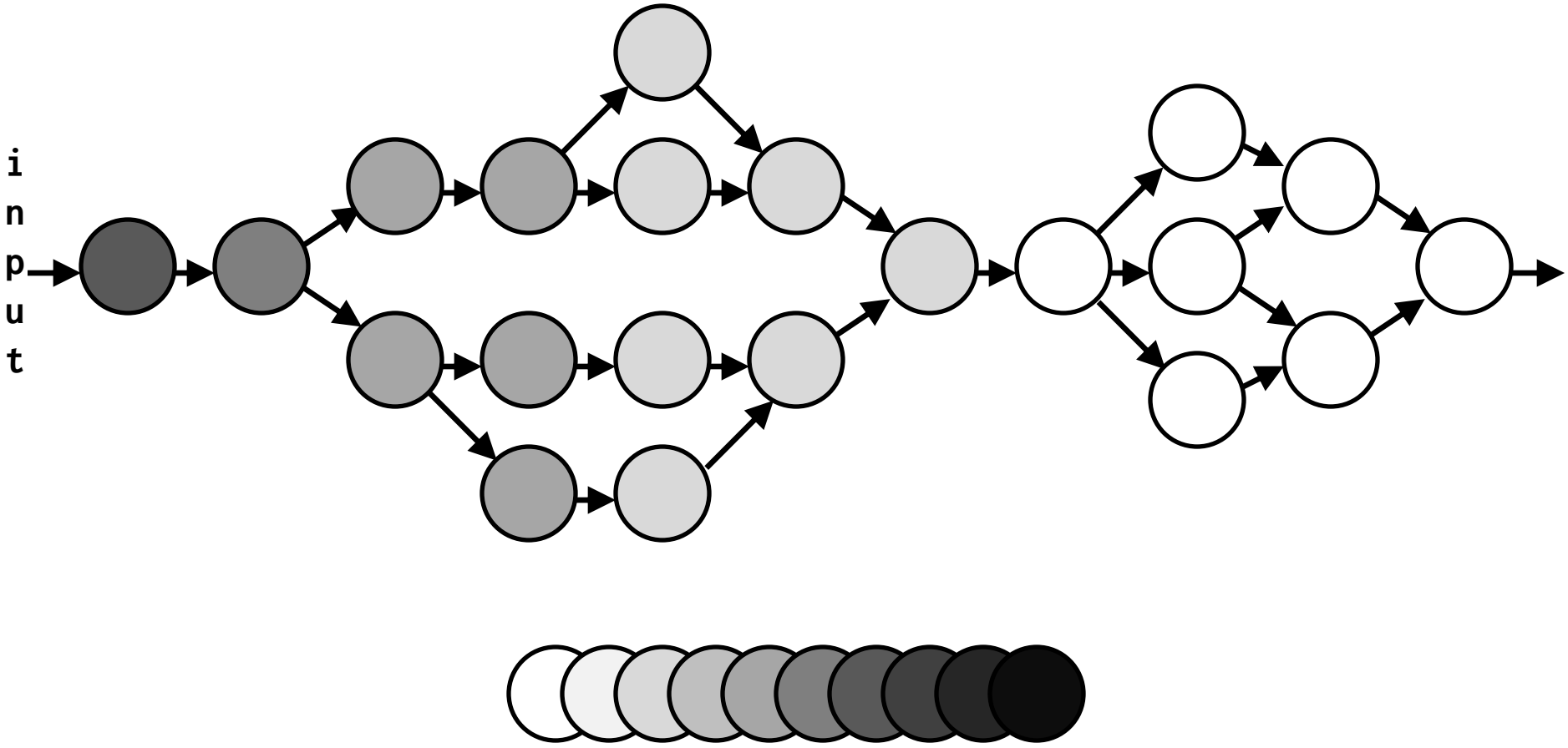
Anytime Automaton



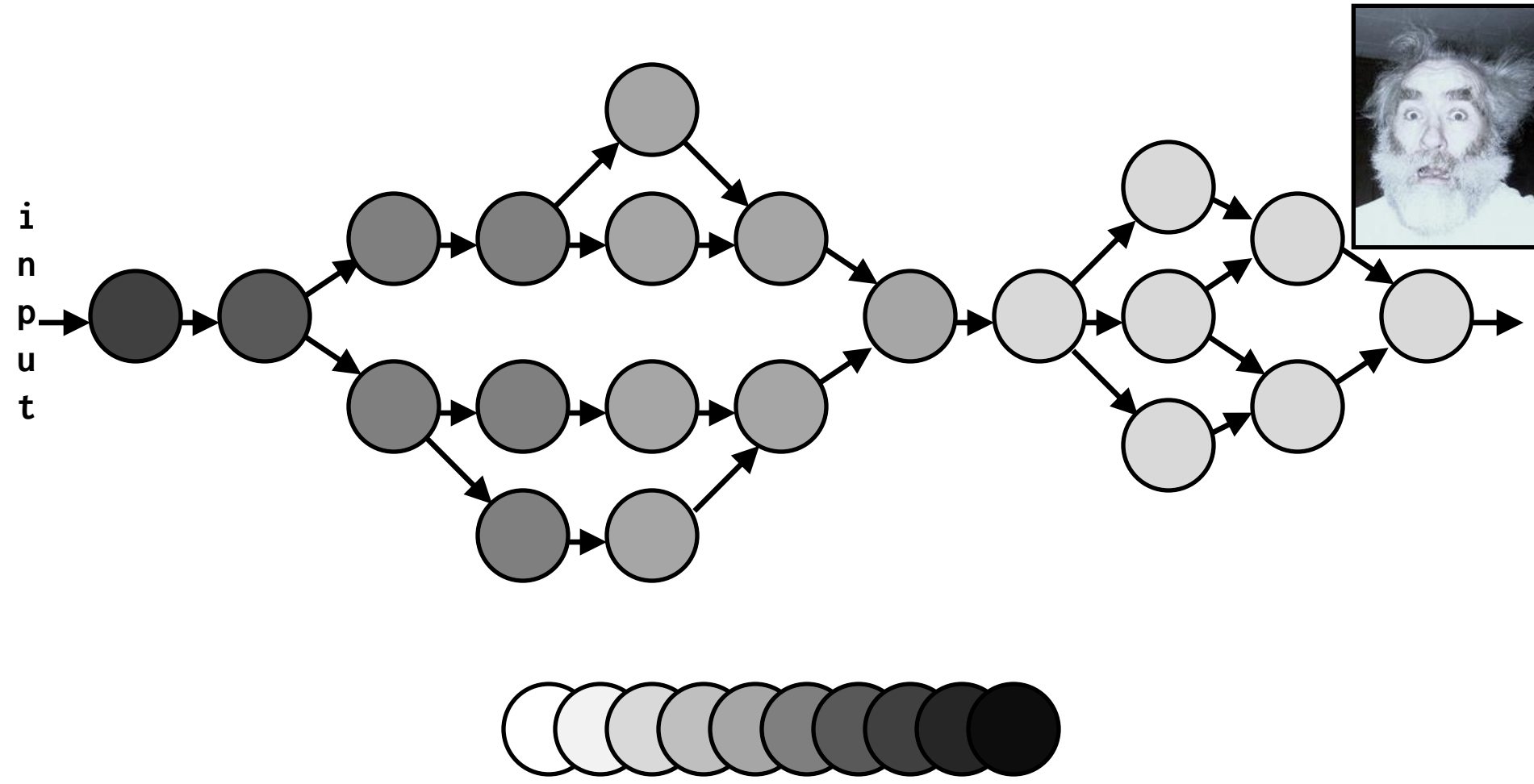
Anytime Automaton



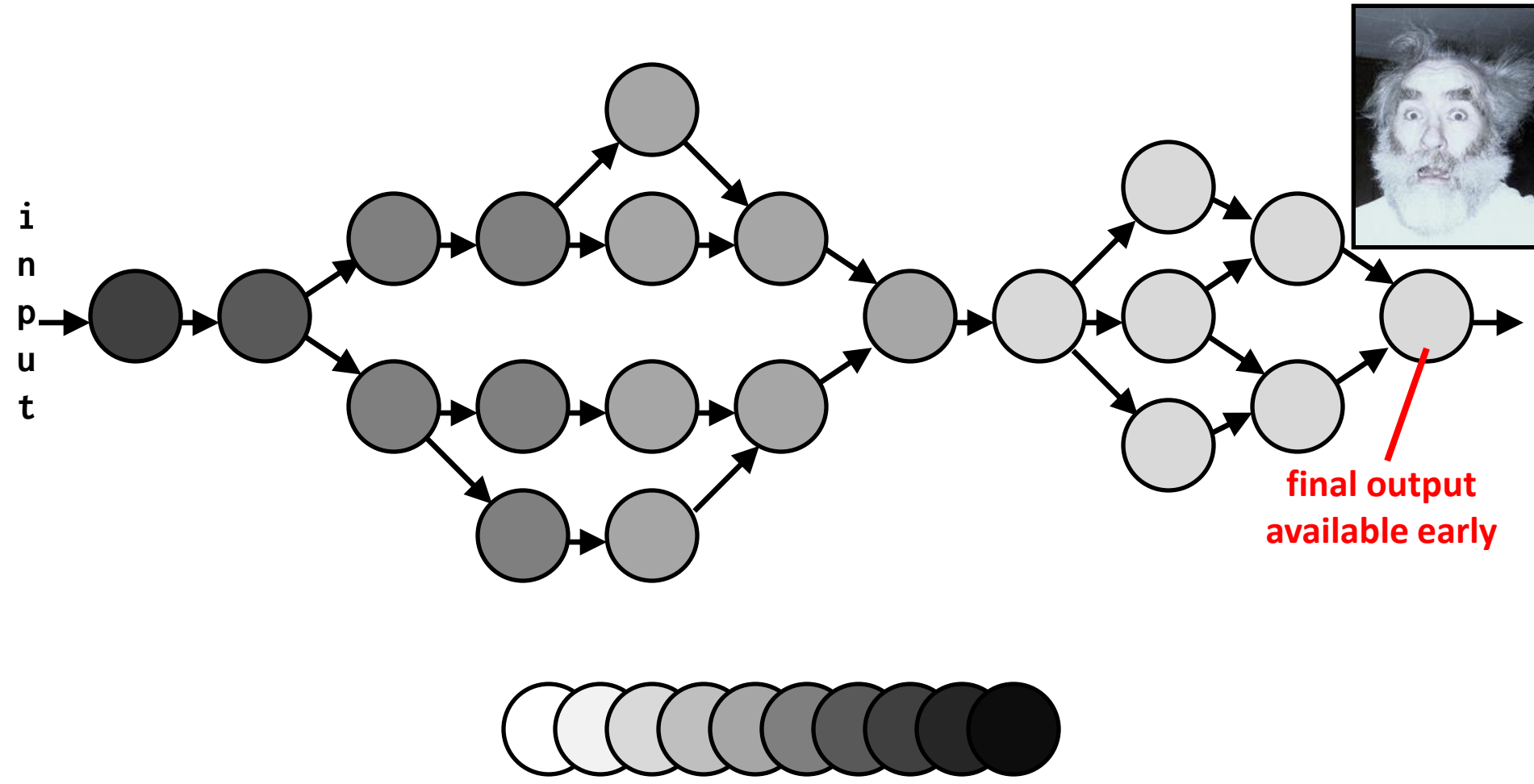
Anytime Automaton



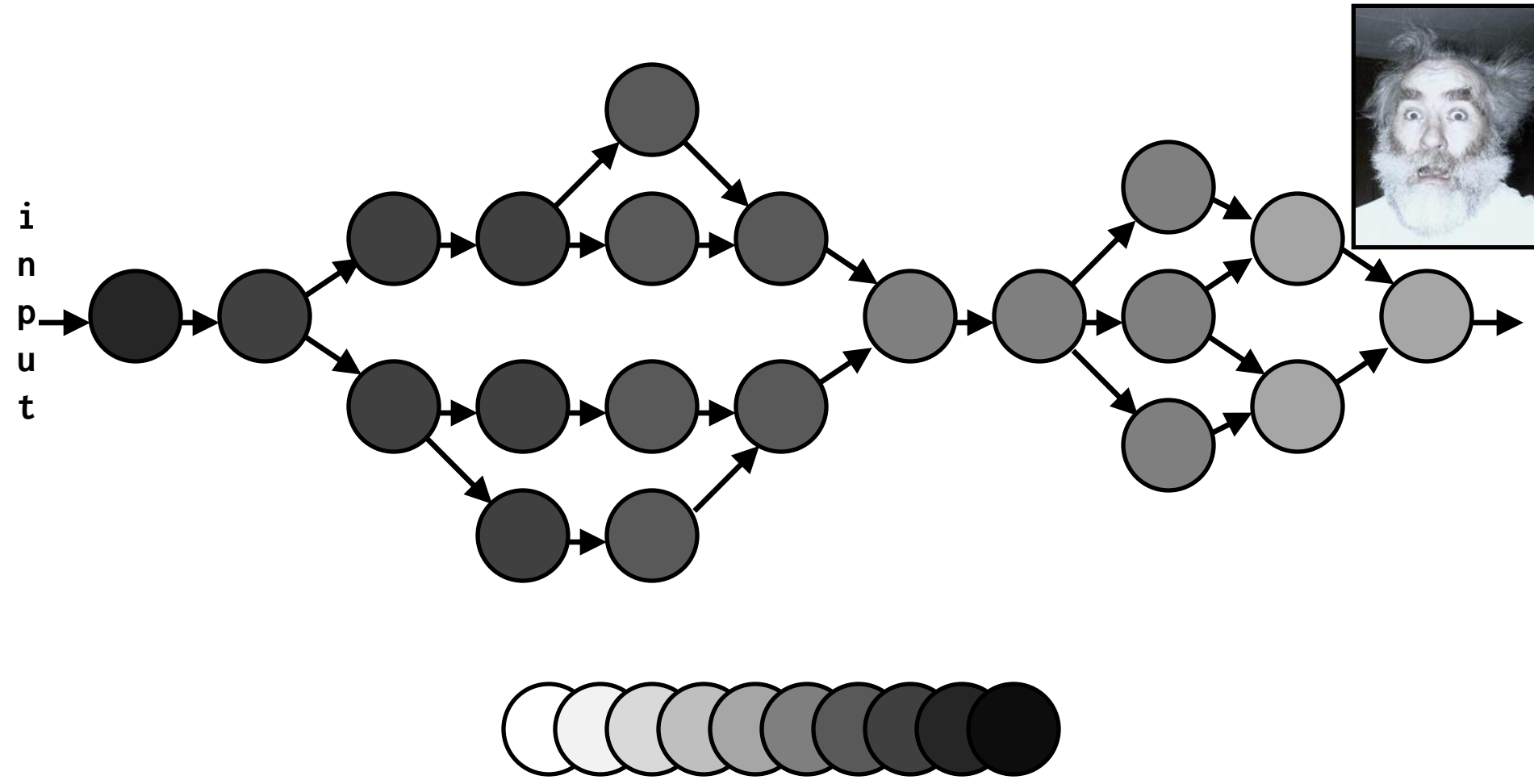
Anytime Automaton



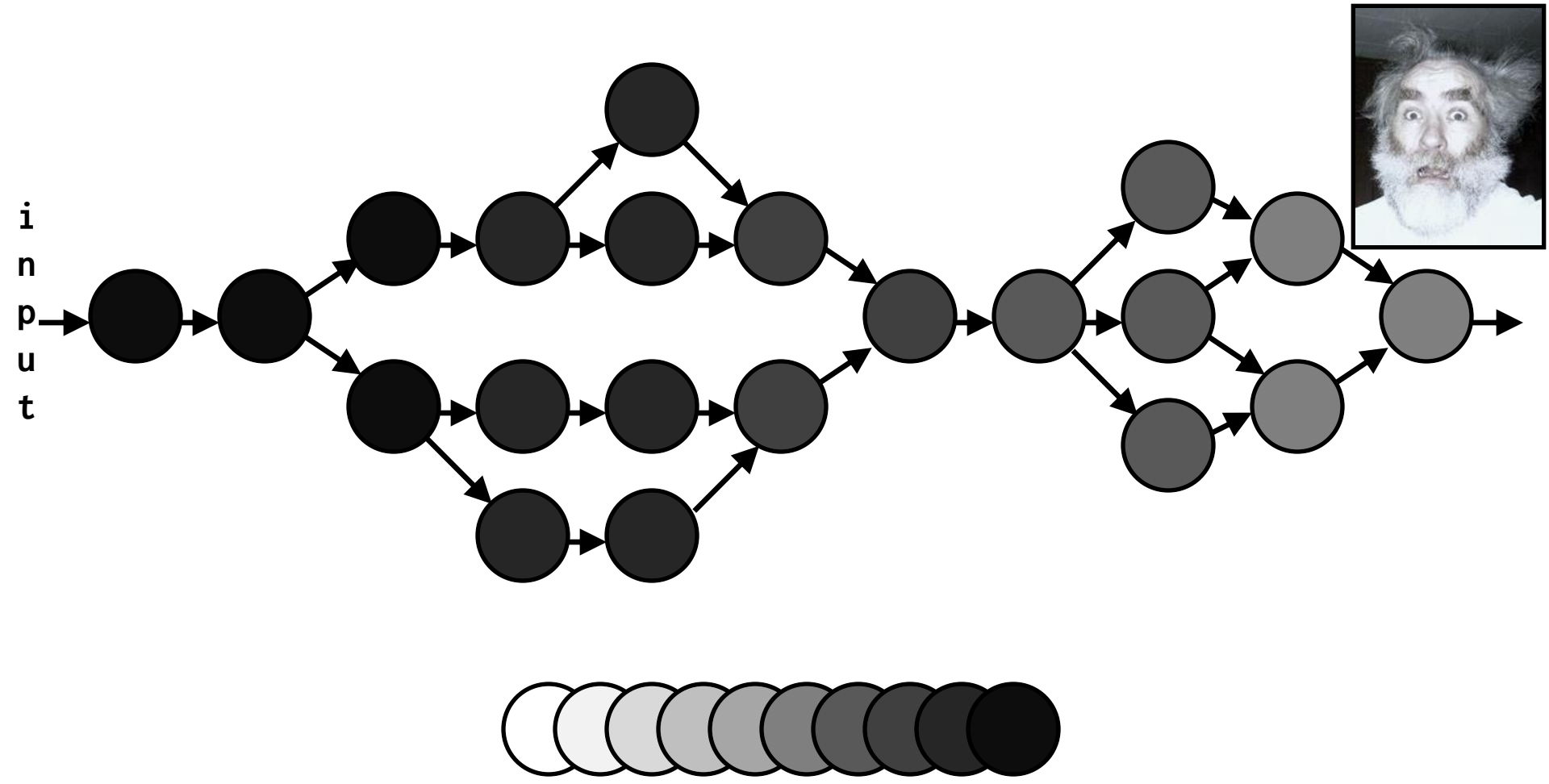
Anytime Automaton



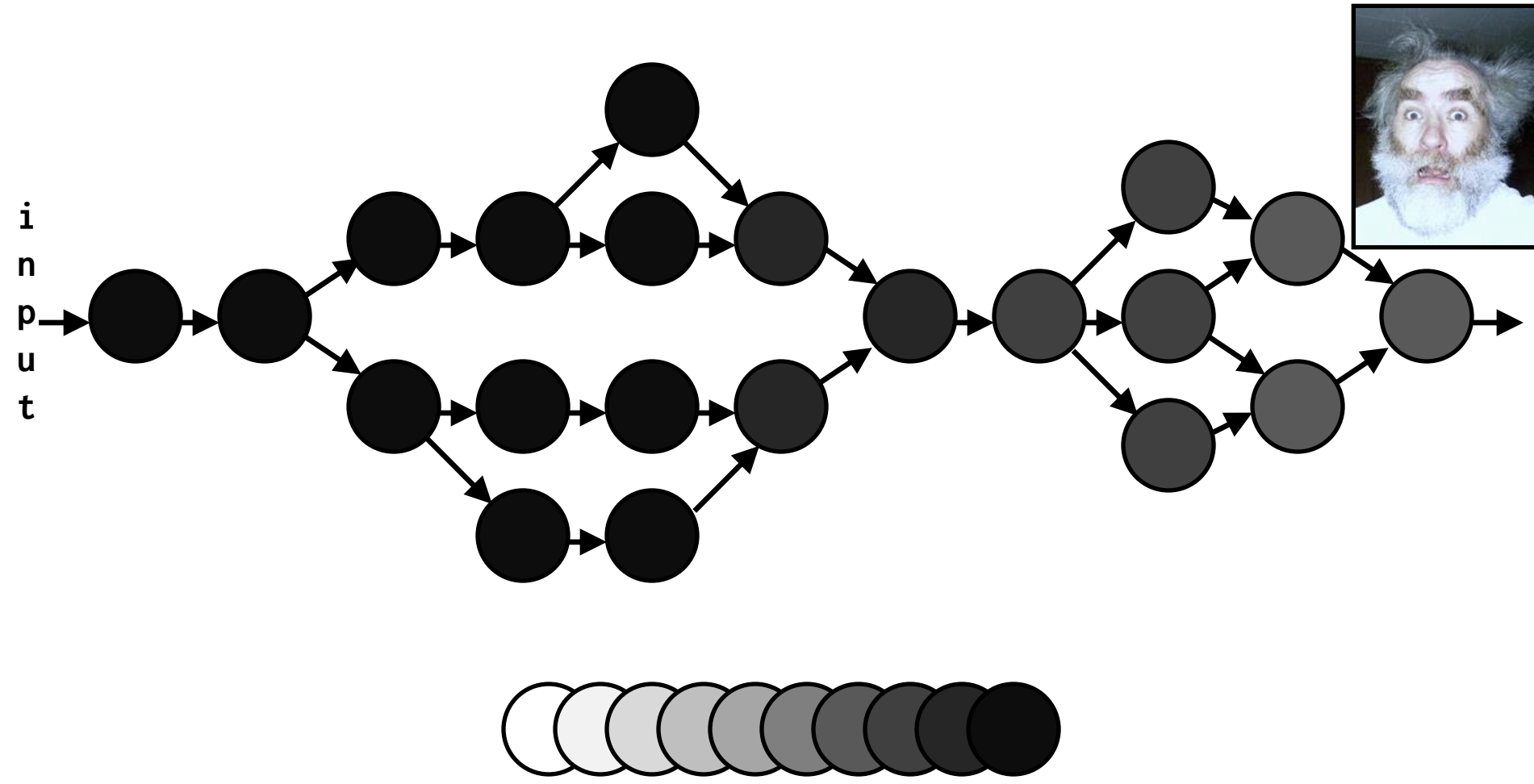
Anytime Automaton



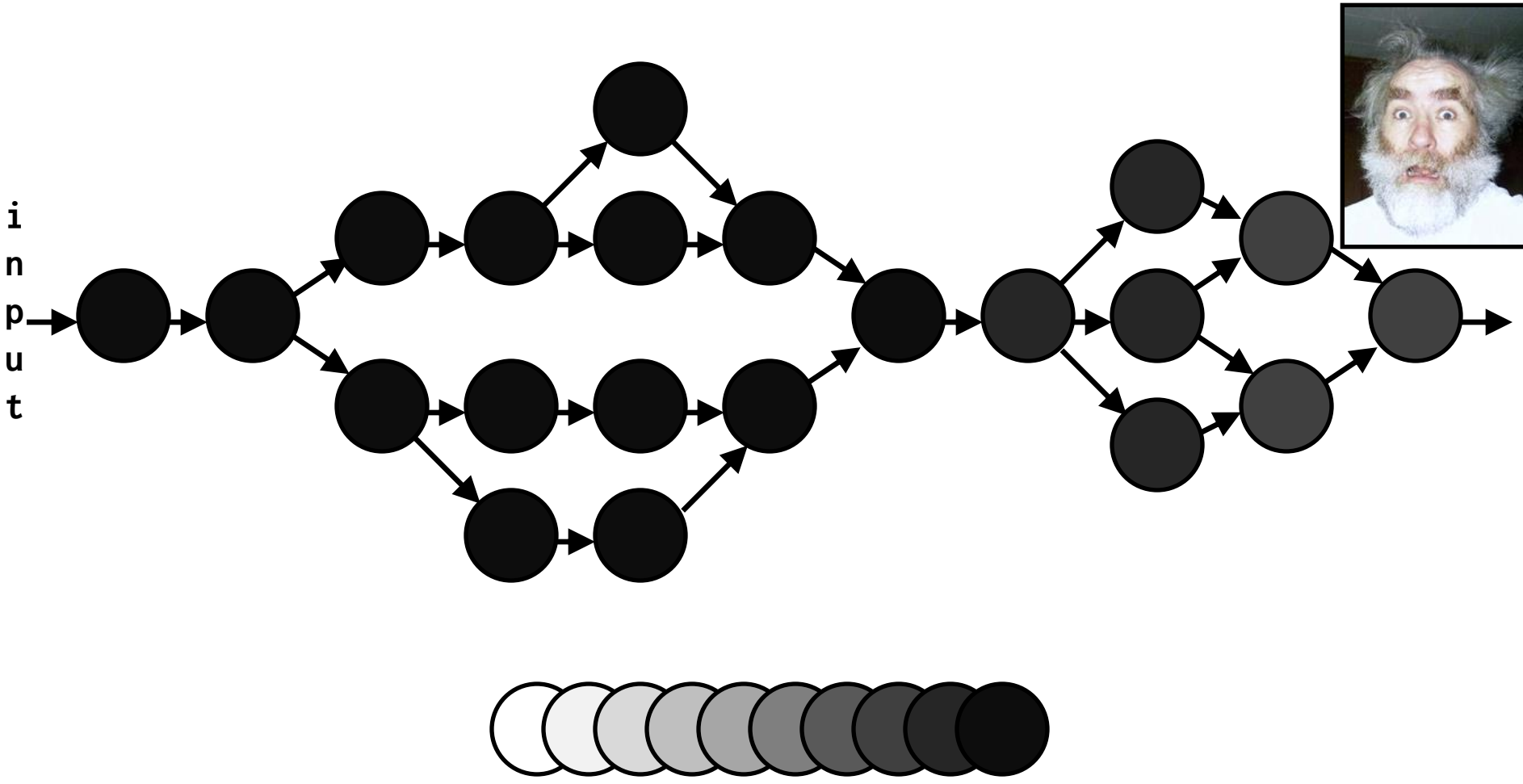
Anytime Automaton



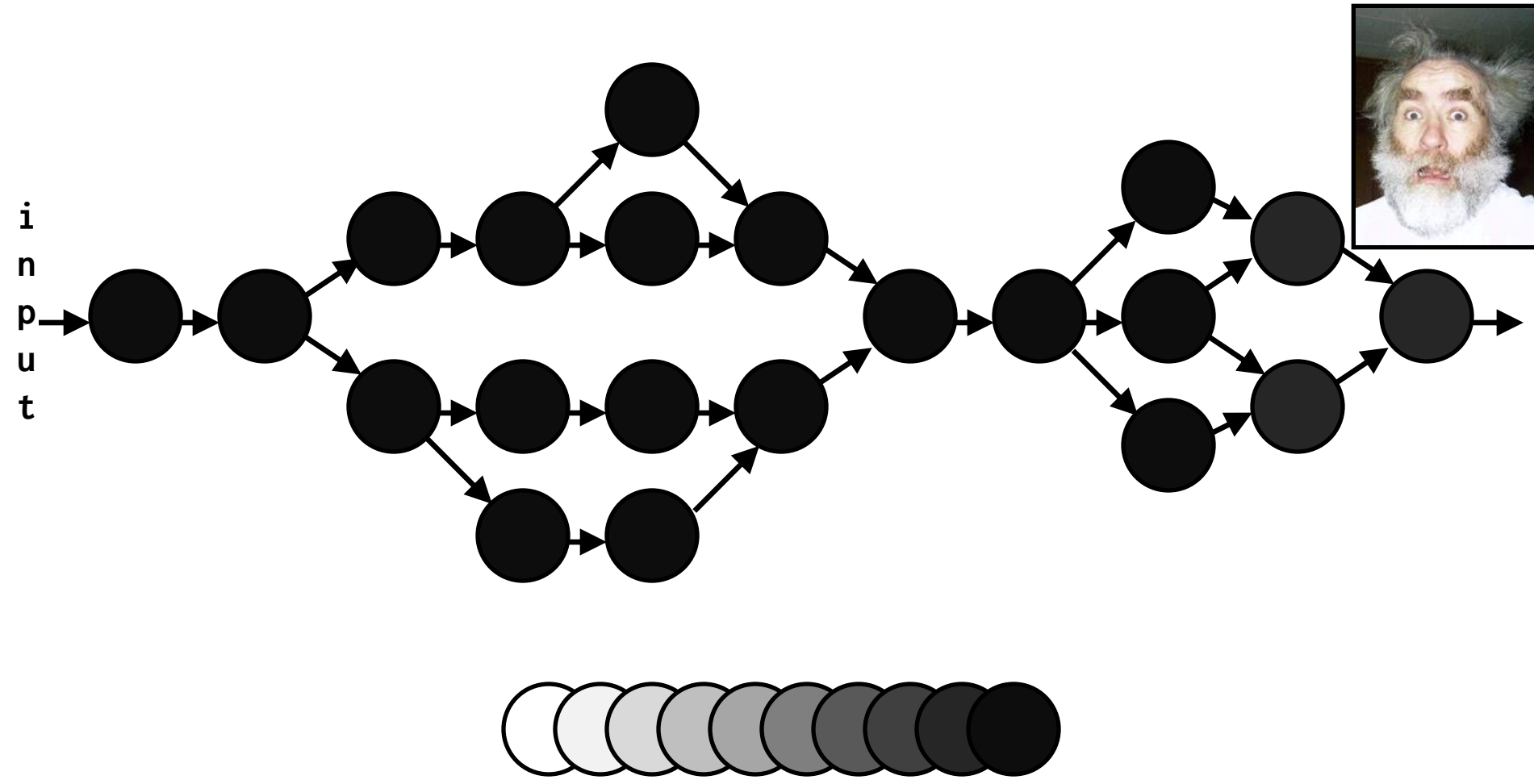
Anytime Automaton



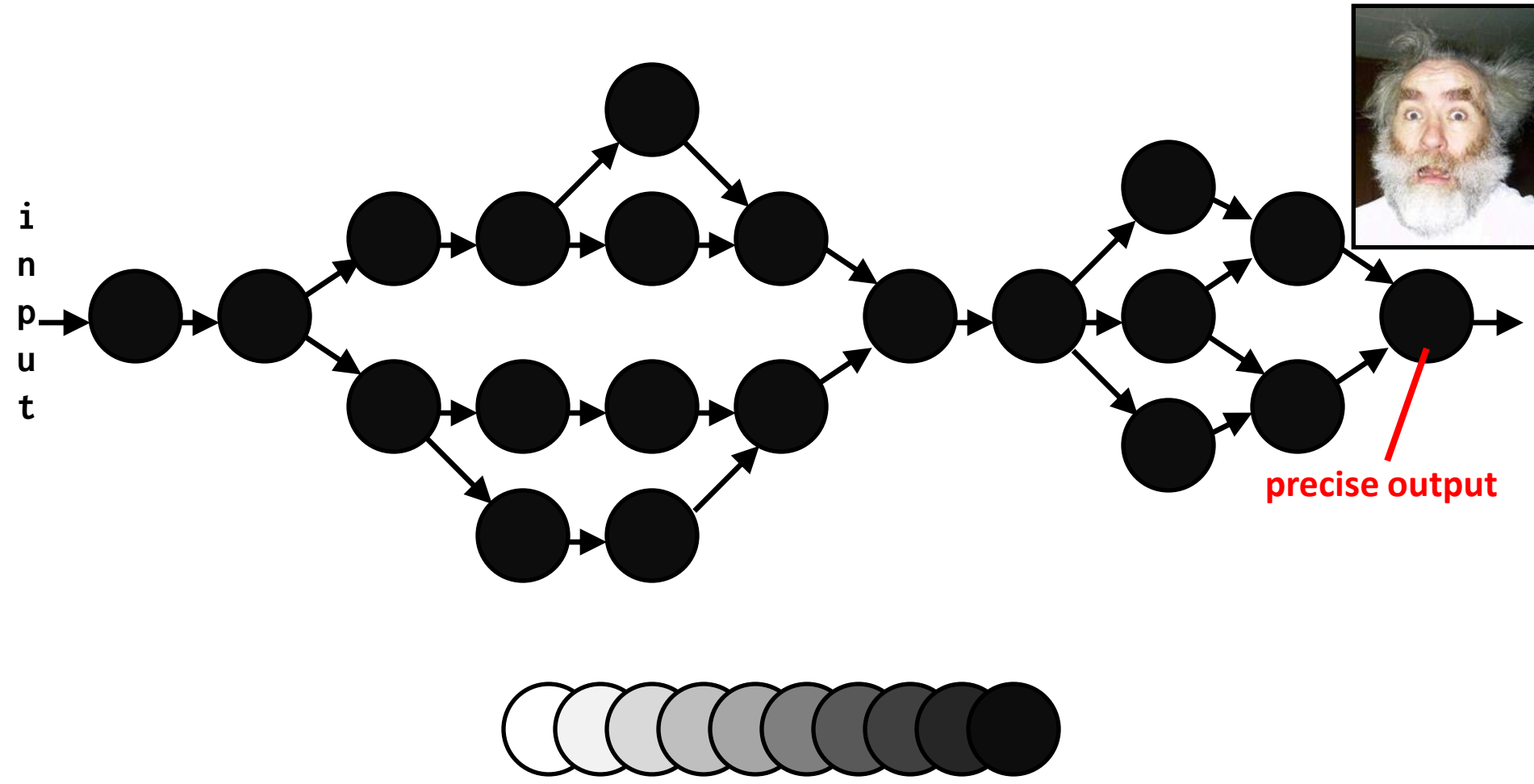
Anytime Automaton



Anytime Automaton



Anytime Automaton

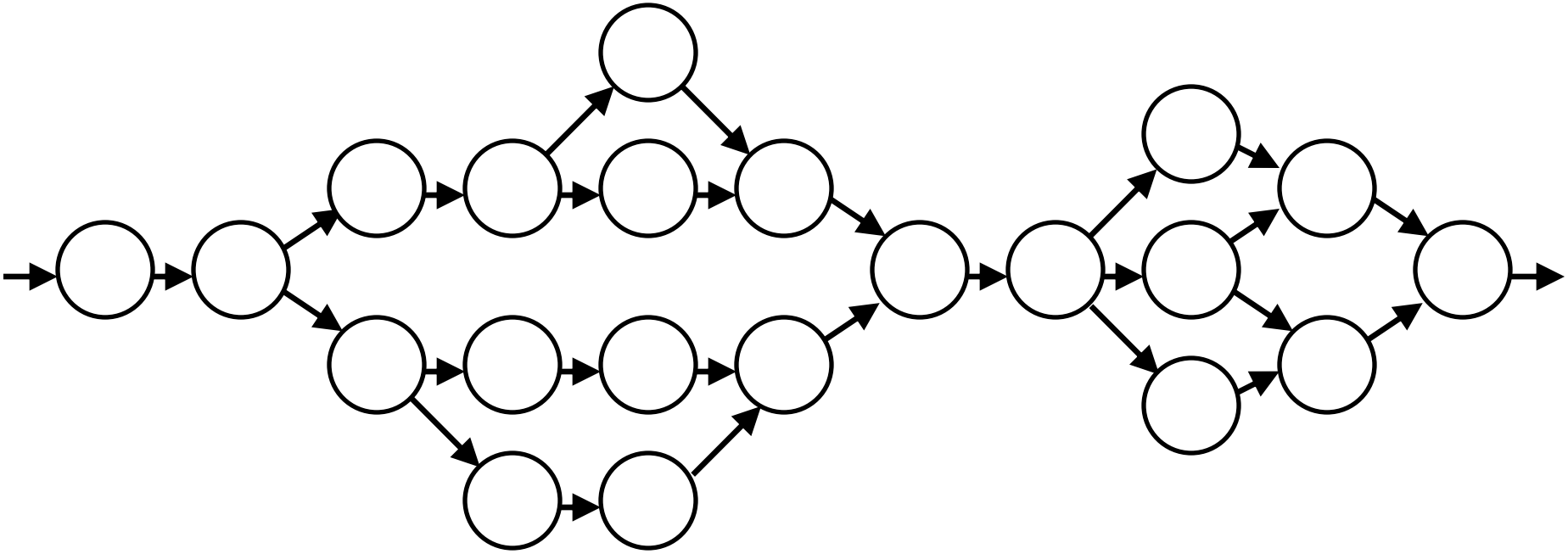


Anytime Automaton – The Model

1. Ensure precise output is always produced eventually.

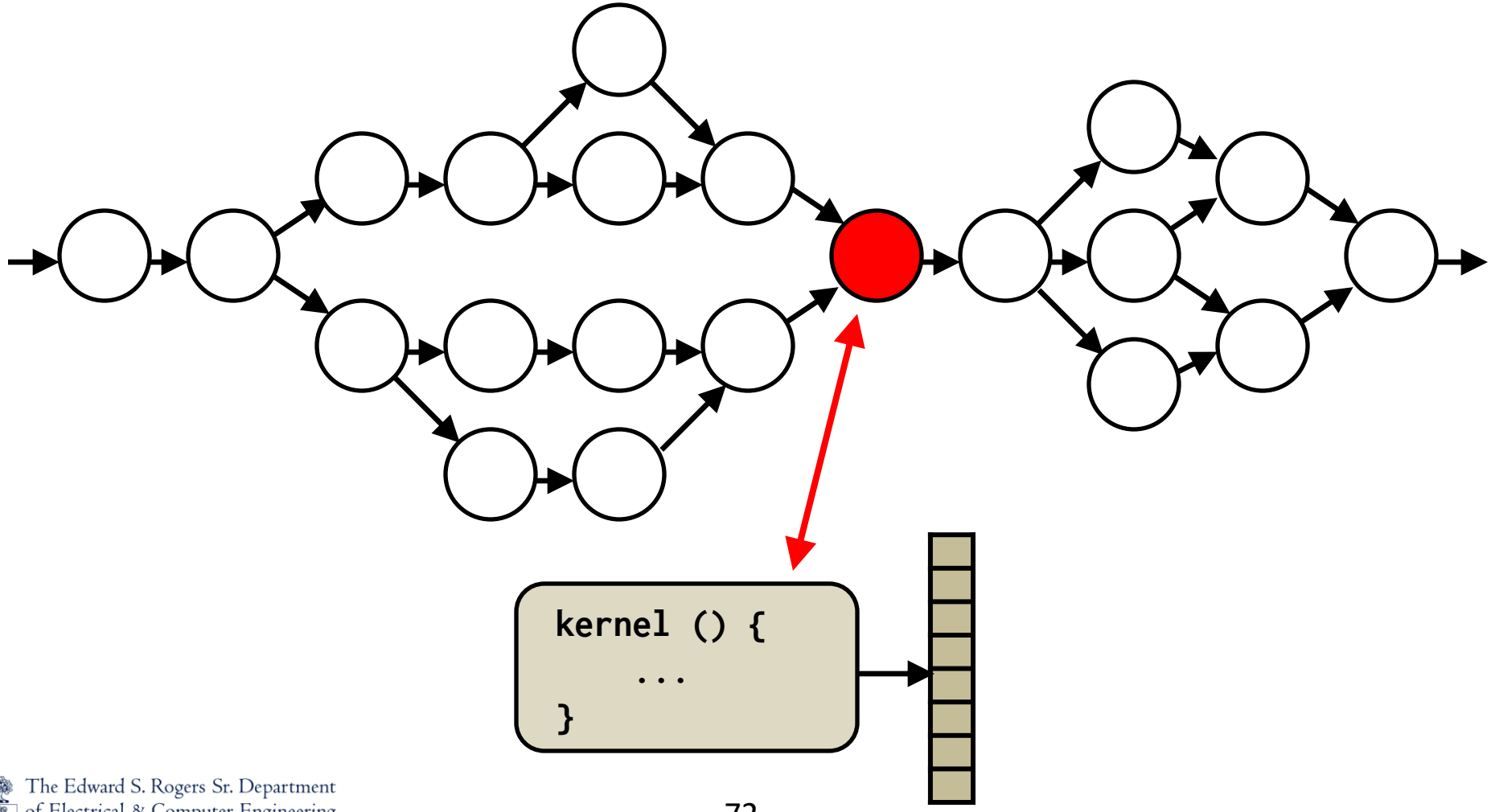
Anytime Automaton – The Model

1. Ensure precise output is always produced eventually.



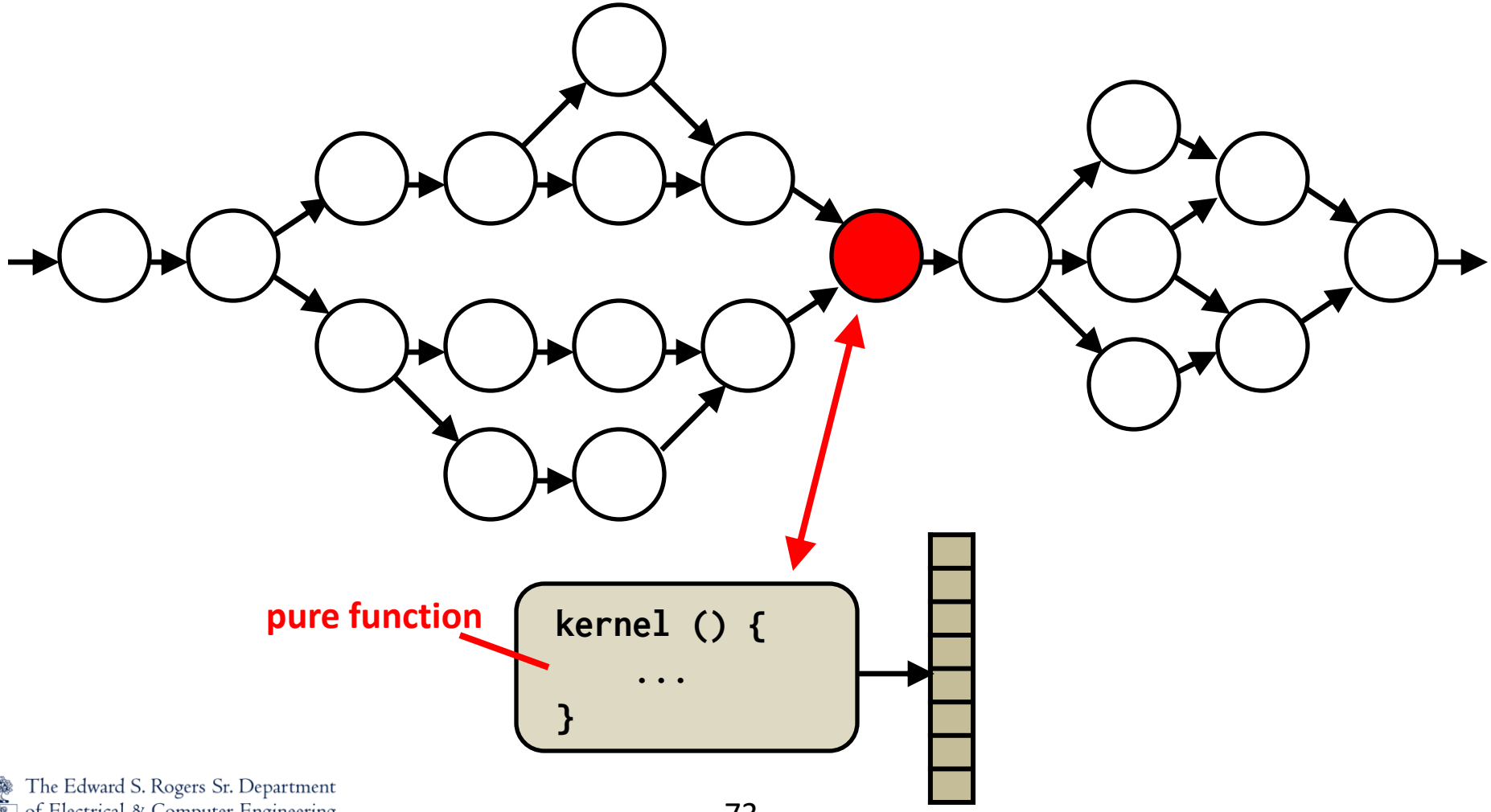
Anytime Automaton – The Model

1. Ensure precise output is always produced eventually.



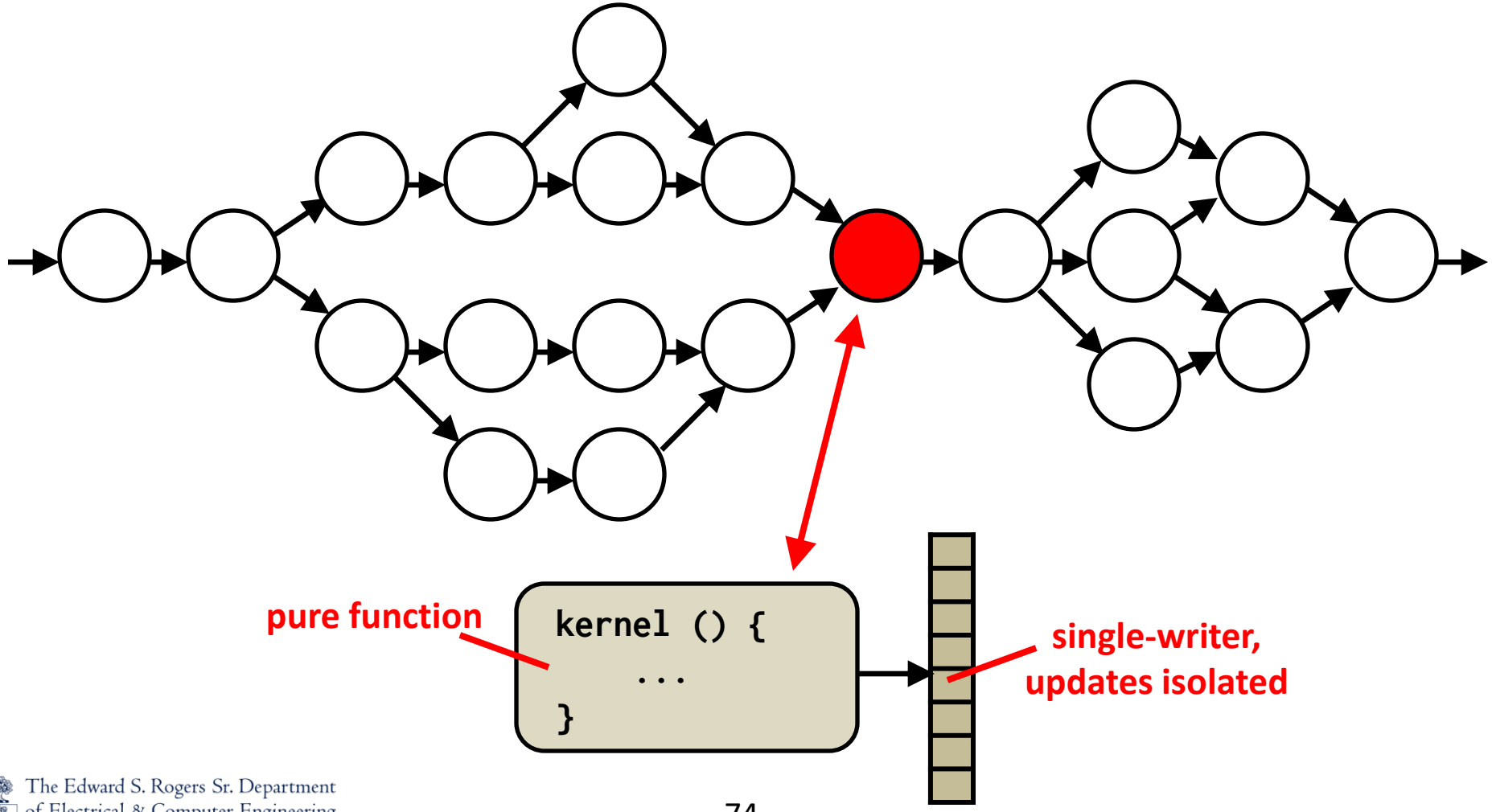
Anytime Automaton – The Model

1. Ensure precise output is always produced eventually.



Anytime Automaton – The Model

1. Ensure precise output is always produced eventually.



Anytime Automaton – The Model

2. Create the effect of improving accuracy over time.

Anytime Automaton – The Model

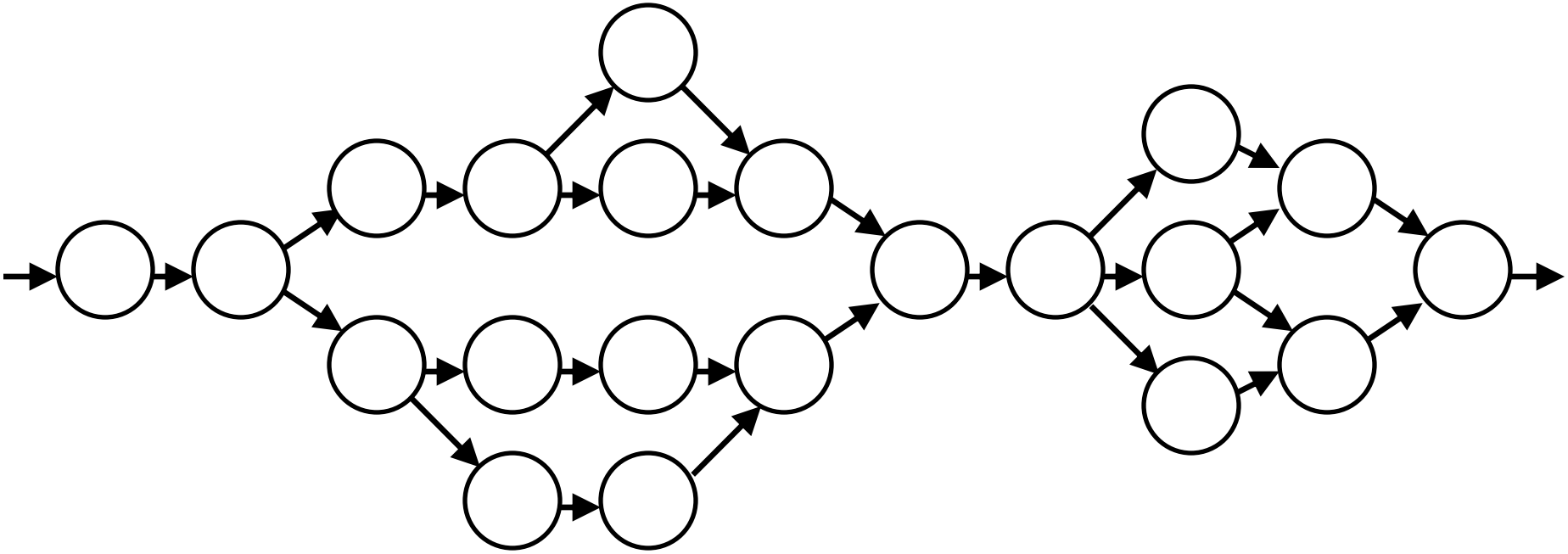
2. Create the effect of improving accuracy over time.

Anytime (or iterative) algorithms have been studied before but are traditionally built into the *coarse-grained* derivation of an application.

Approximate computing techniques have proliferated recently and have been shown to have general *fine-grained* applicability.

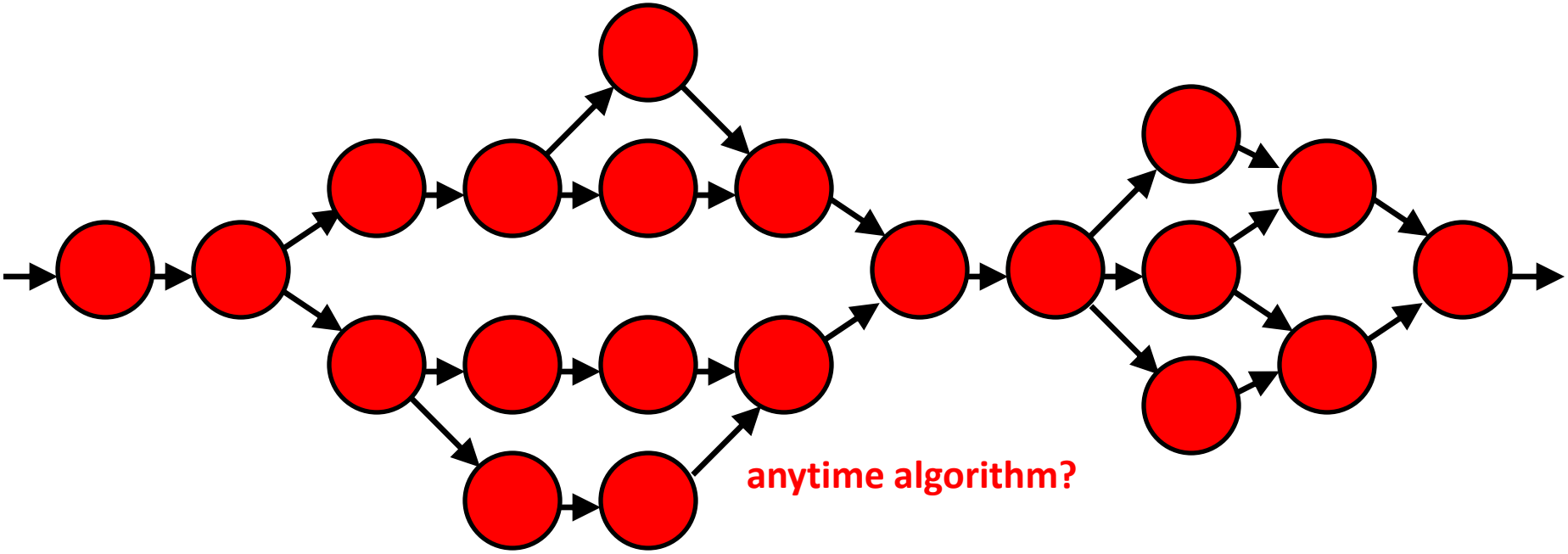
Anytime Automaton – The Model

2. Create the effect of improving accuracy over time.



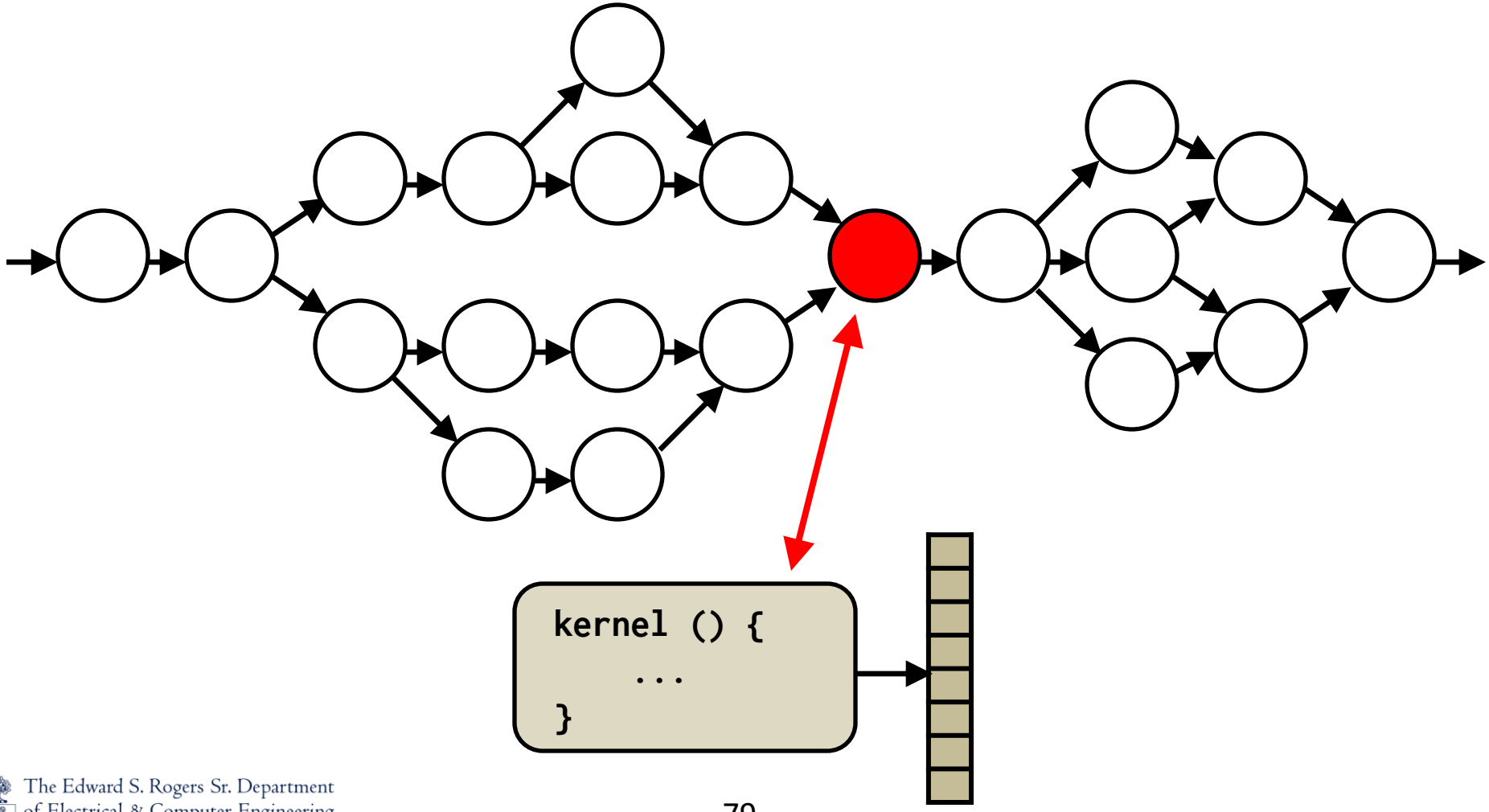
Anytime Automaton – The Model

2. Create the effect of improving accuracy over time.



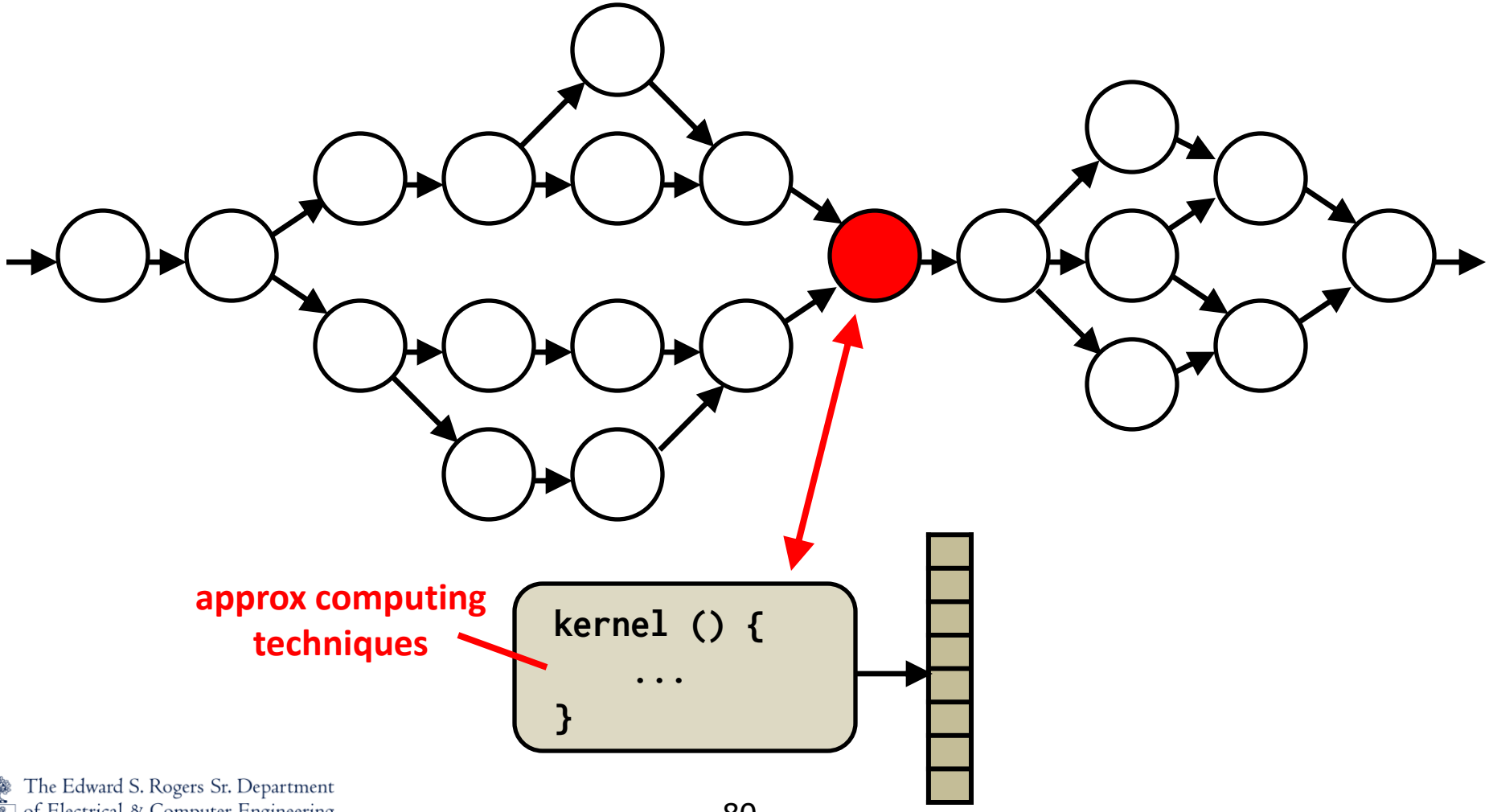
Anytime Automaton – The Model

2. Create the effect of improving accuracy over time.



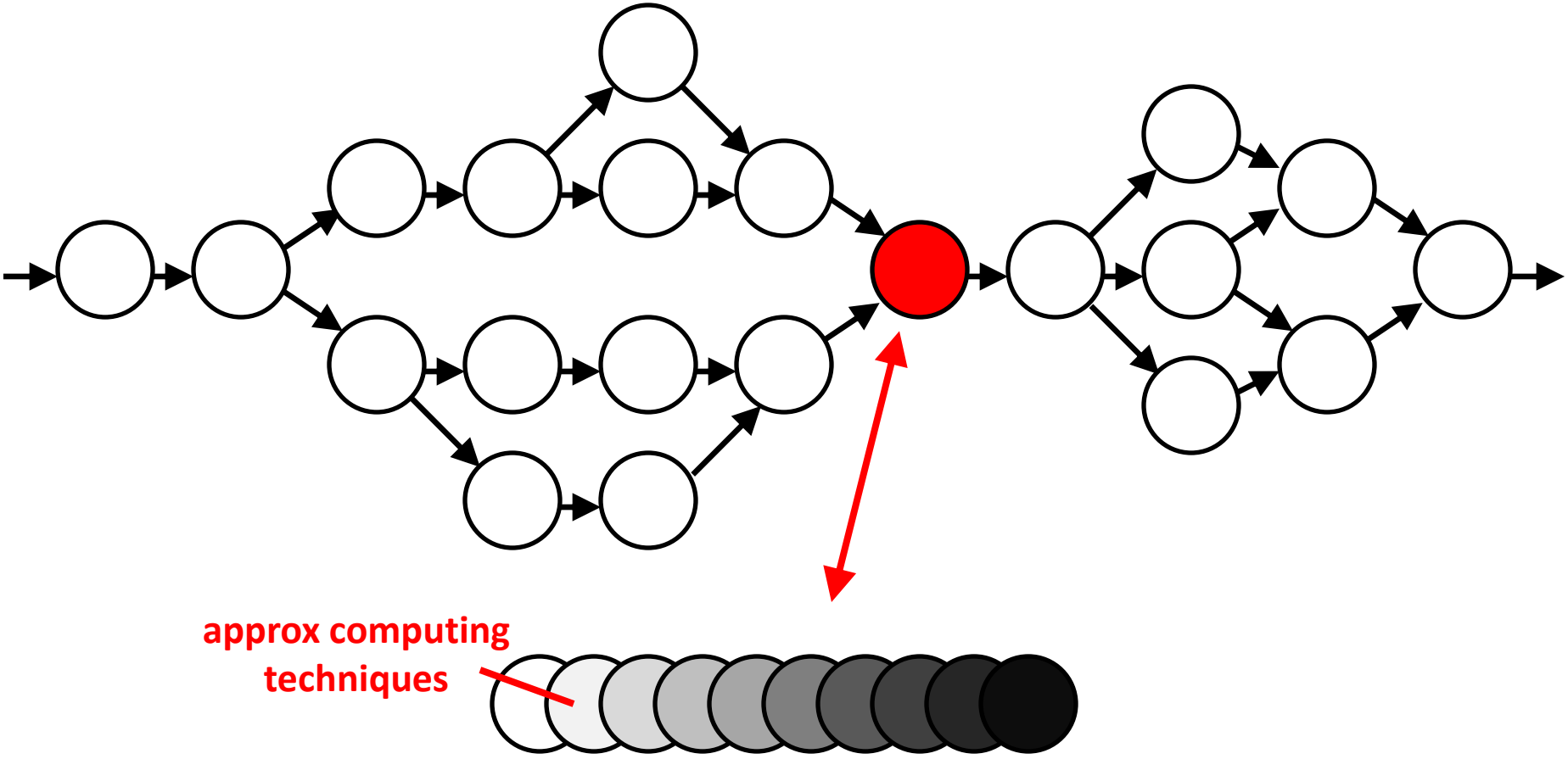
Anytime Automaton – The Model

2. Create the effect of improving accuracy over time.



Anytime Automaton – The Model

2. Create the effect of improving accuracy over time.

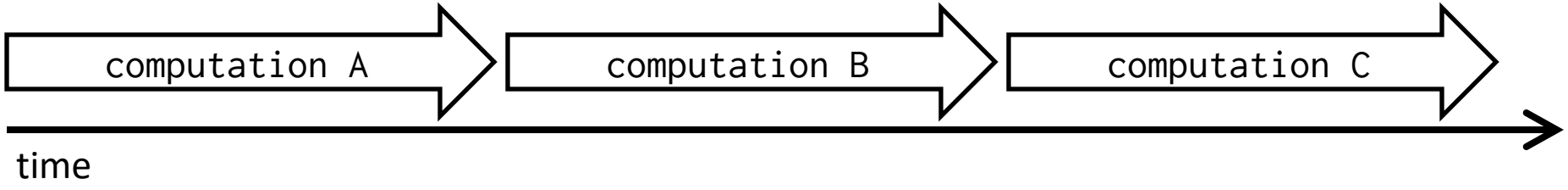


Anytime Automaton – The Model

3. Enable interruptibility via pipelining.

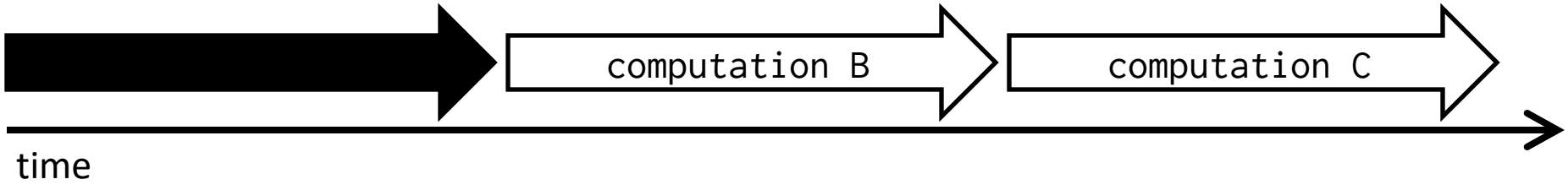
Anytime Automaton – The Model

3. Enable interruptibility via pipelining.



Anytime Automaton – The Model

3. Enable interruptibility via pipelining.

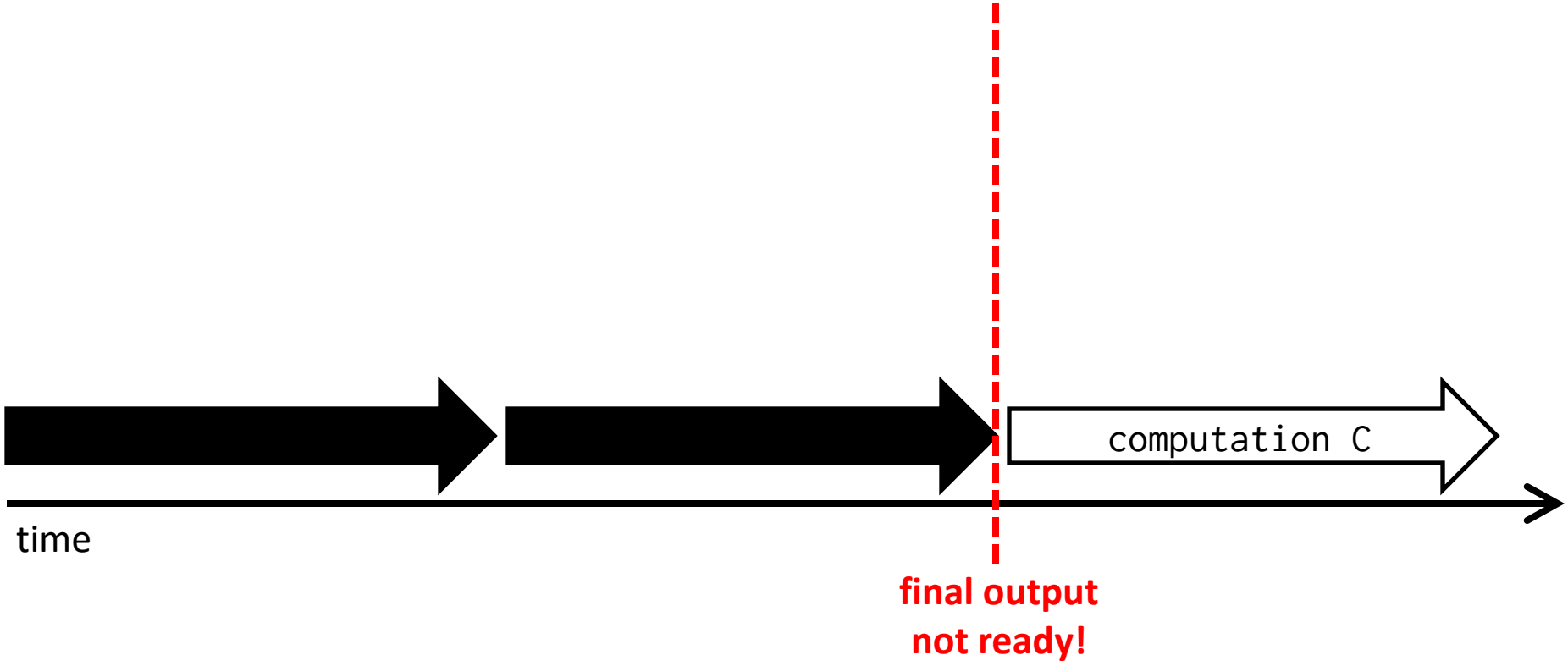


Anytime Automaton – The Model

3. Enable interruptibility via pipelining.

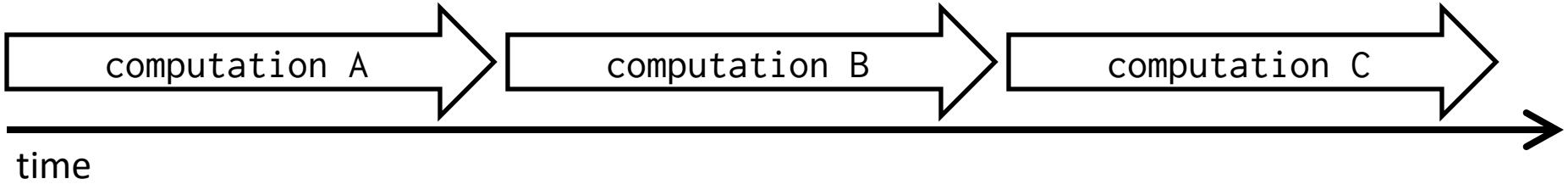
Anytime Automaton – The Model

3. Enable interruptibility via pipelining.



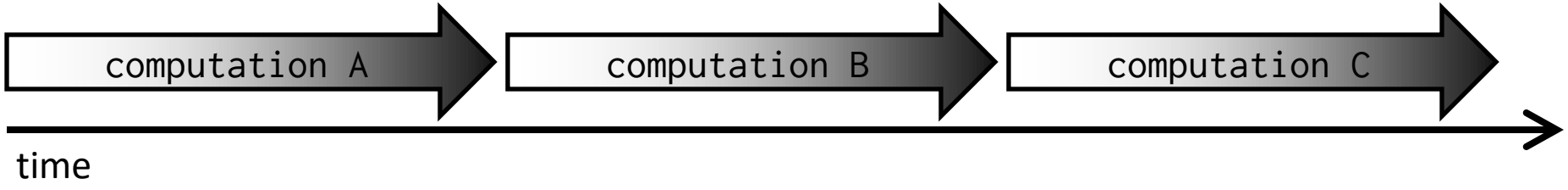
Anytime Automaton – The Model

3. Enable interruptibility via pipelining.



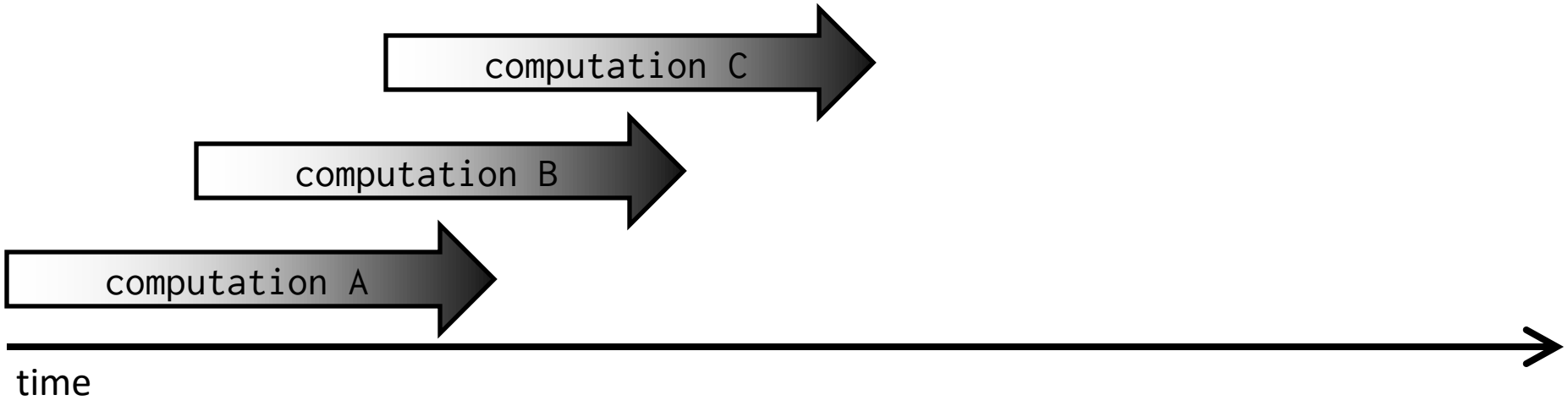
Anytime Automaton – The Model

3. Enable interruptibility via pipelining.



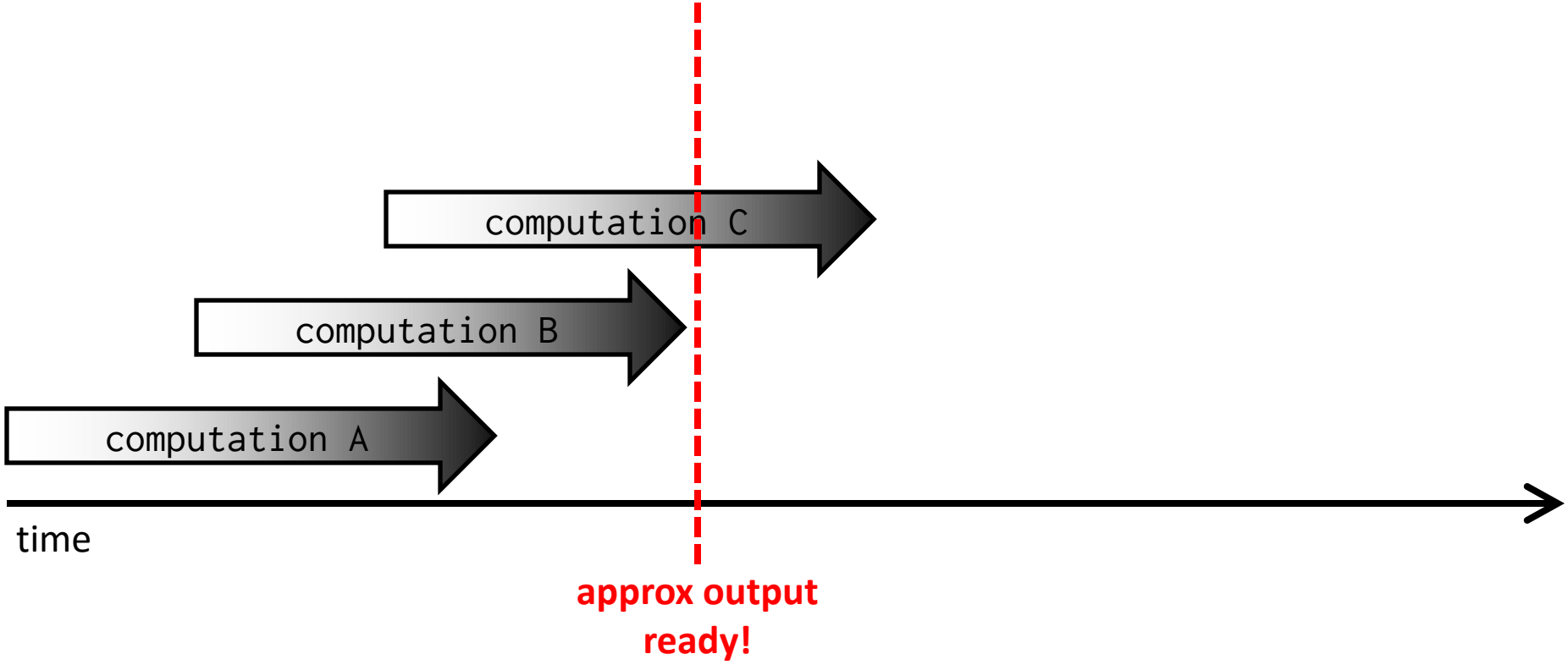
Anytime Automaton – The Model

3. Enable interruptibility via pipelining.



Anytime Automaton – The Model

3. Enable interruptibility via pipelining.

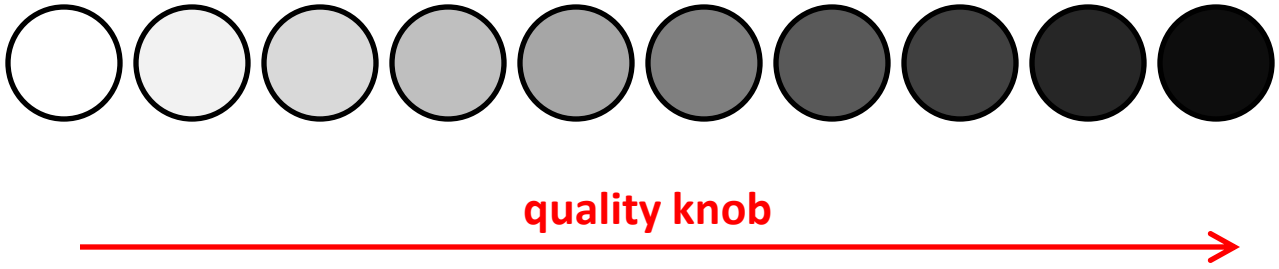


Anytime Automaton – The Approximations

1. General case: apply approximations **iteratively**.

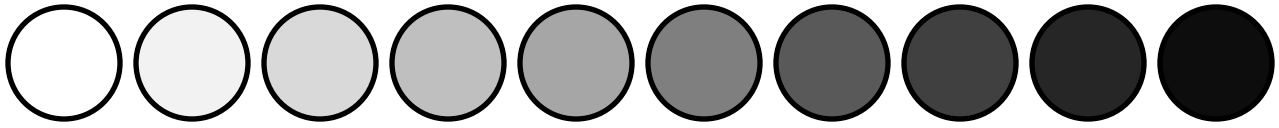
Anytime Automaton – The Approximations

1. General case: apply approximations **iteratively**.



Anytime Automaton – The Approximations

1. General case: apply approximations **iteratively**.



loop perforation — smaller perforation stride →

floating-point precision — more mantissa bits →

SRAM bit upsets — higher supply voltage →

load value approximation — lower approximation degree →

neural acceleration — higher neural network complexity →

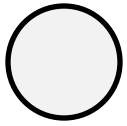
Anytime Automaton – The Approximations

1. General case: apply approximations **iteratively**.
   - ➢ Loop perforation

Anytime Automaton – The Approximations

1. General case: apply approximations **iteratively**.

- Loop perforation

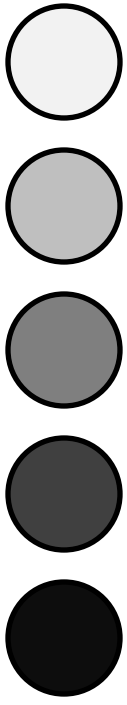


perforation stride 20: `for i = 0, 20, 40, 60, 80, 100, ...., N-1`

Anytime Automaton – The Approximations

1. General case: apply approximations **iteratively**.

- Loop perforation



perforation stride 20: `for i = 0, 20, 40, 60, 80, 100, ...., N-1`

perforation stride 15: `for i = 0, 15, 30, 45, 60, 75, ....., N-1`

perforation stride 10: `for i = 0, 10, 20, 30, 40, 50, ....., N-1`

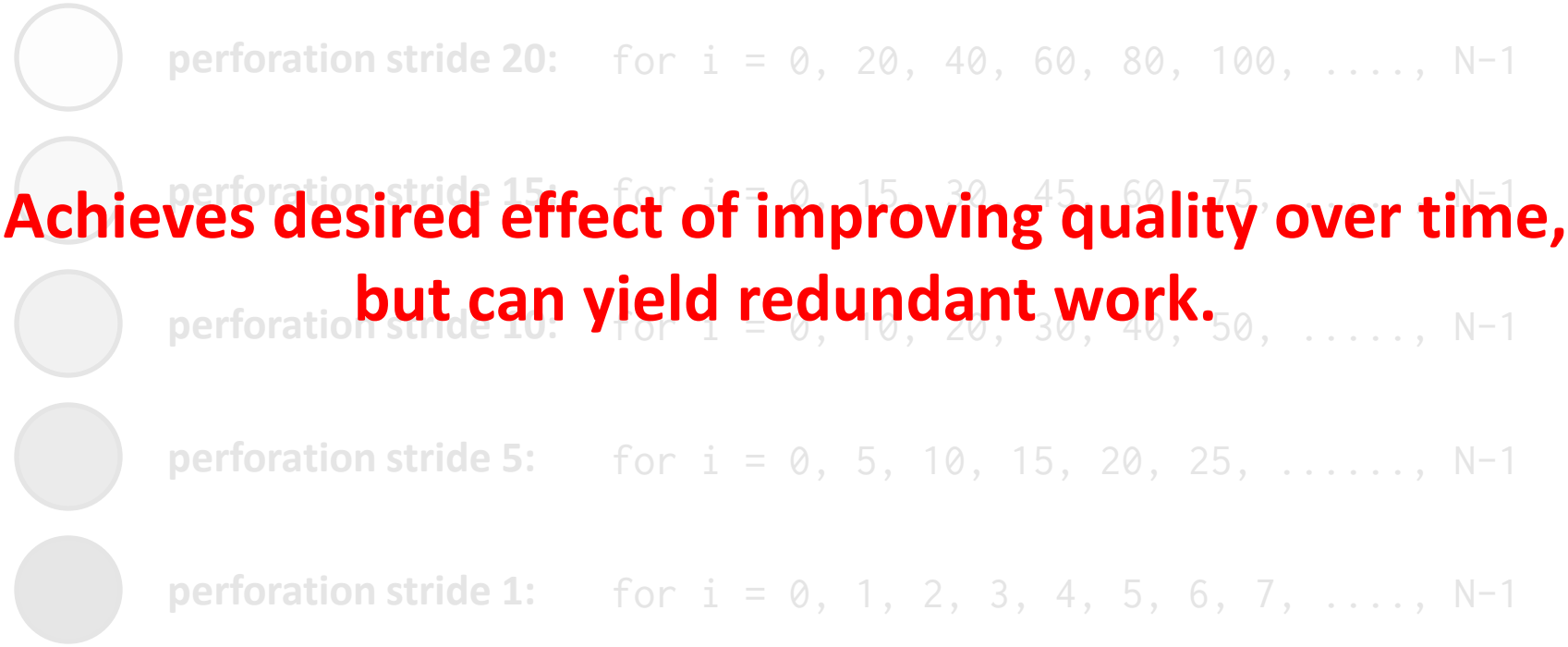
perforation stride 5: `for i = 0, 5, 10, 15, 20, 25, ......, N-1`

perforation stride 1: `for i = 0, 1, 2, 3, 4, 5, 6, 7, ...., N-1`

Anytime Automaton – The Approximations

1. General case: apply approximations **iteratively**.

➢ Loop perforation

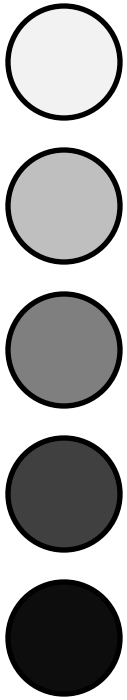


Achieves desired effect of improving quality over time, but can yield redundant work.

Anytime Automaton – The Approximations

1. General case: apply approximations **iteratively**.

- Loop perforation



perforation stride 20: `for i = 0, 20, 40, 60, 80, 100, ...., N-1`

perforation stride 15: `for i = 0, 15, 30, 45, 60, 75, ....., N-1`

perforation stride 10: `for i = 0, 10, 20, 30, 40, 50, ....., N-1`

perforation stride 5: `for i = 0, 5, 10, 15, 20, 25, ......, N-1`

perforation stride 1: `for i = 0, 1, 2, 3, 4, 5, 6, 7, ...., N-1`

Anytime Automaton – The Approximations

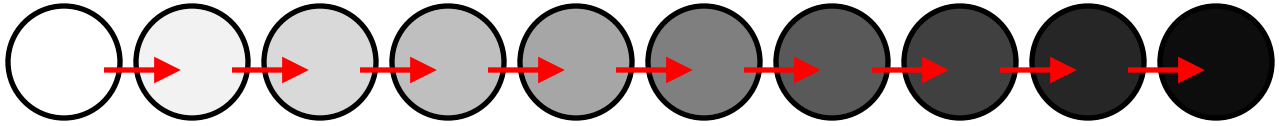
2. Better case: apply **diffusive** approximations.

- Each approximation builds on the previous one.

Anytime Automaton – The Approximations

2. Better case: apply **diffusive** approximations.

- Each approximation builds on the previous one.



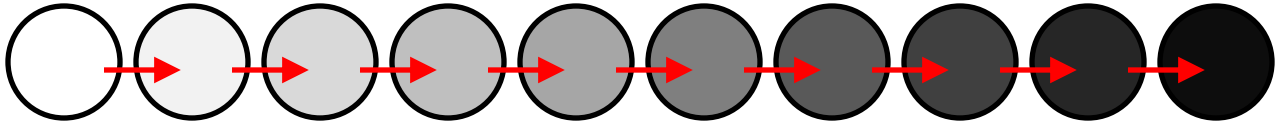
data dependences

(each approximate result contributes usefully to precise result)

Anytime Automaton – The Approximations

2. Better case: apply **diffusive** approximations.

- Each approximation builds on the previous one.



data dependences
(each approximate result contributes usefully to precise result)

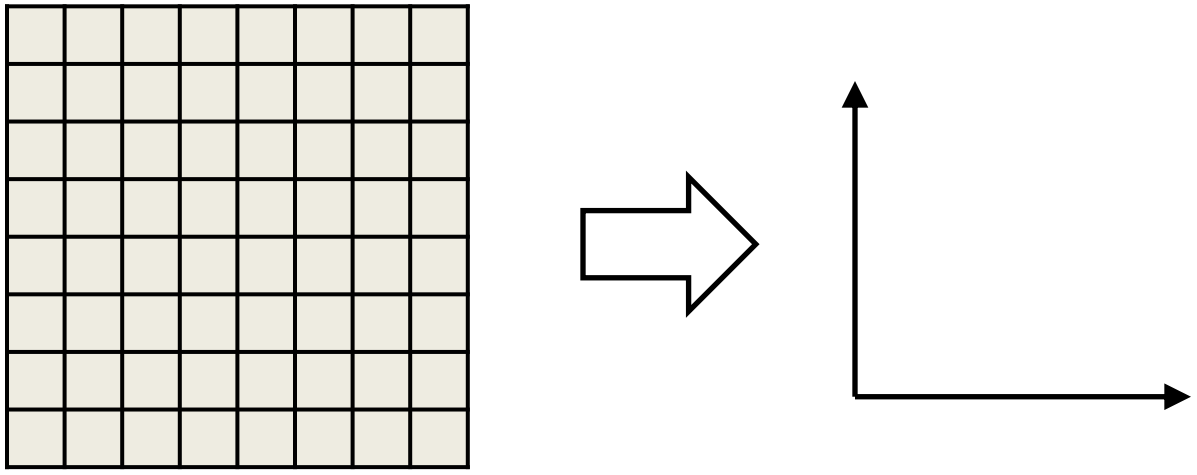
data sampling — more samples →

integer/fixed-point precision — more bits →

Anytime Automaton – The Approximations

2. Better case: apply **diffusive** approximations.

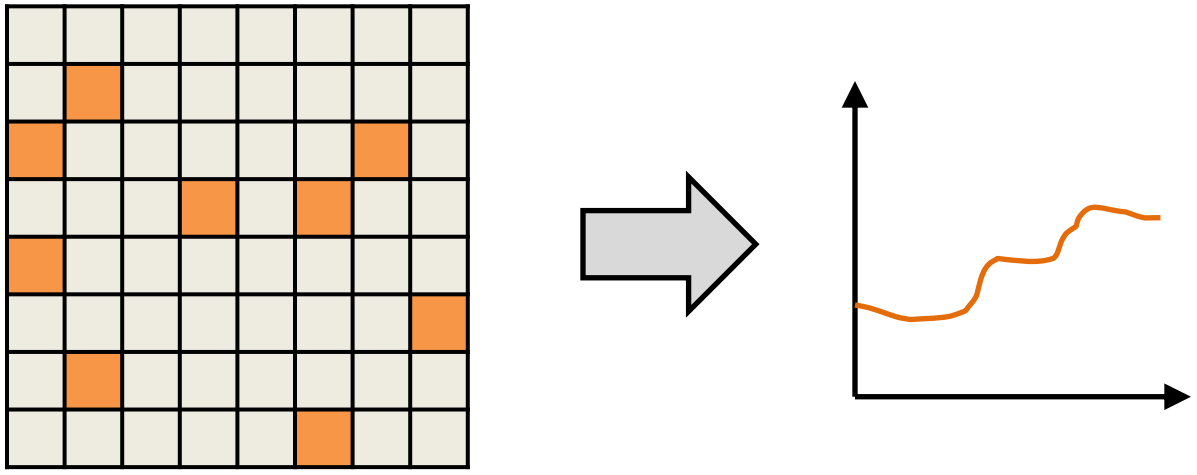
- Input sampling (e.g., generating a distribution)



Anytime Automaton – The Approximations

2. Better case: apply **diffusive** approximations.

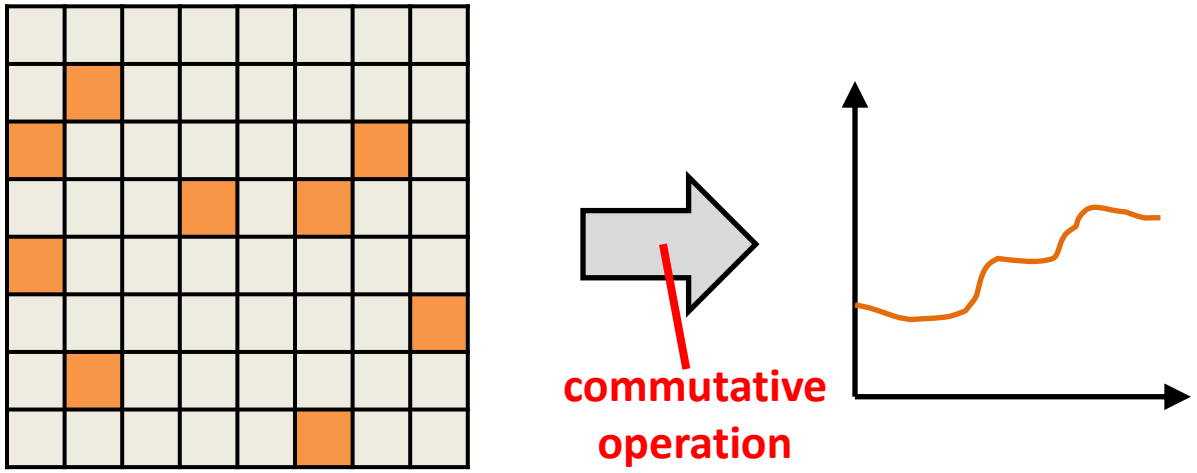
- Input sampling (e.g., generating a distribution)



Anytime Automaton – The Approximations

2. Better case: apply **diffusive** approximations.

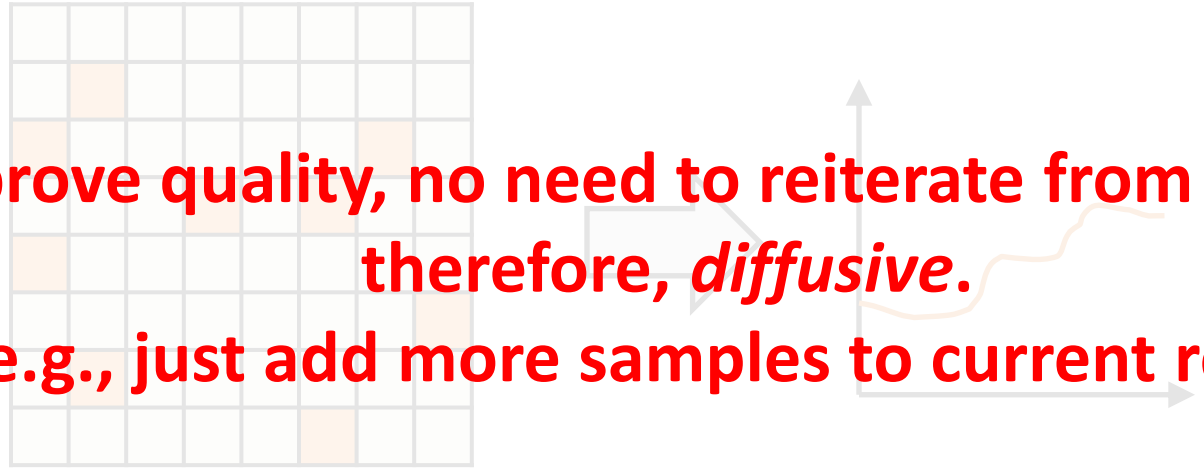
- Input sampling (e.g., generating a distribution)



Anytime Automaton – The Approximations

2. Better case: apply **diffusive** approximations.

- Input sampling (e.g., generating a distribution)

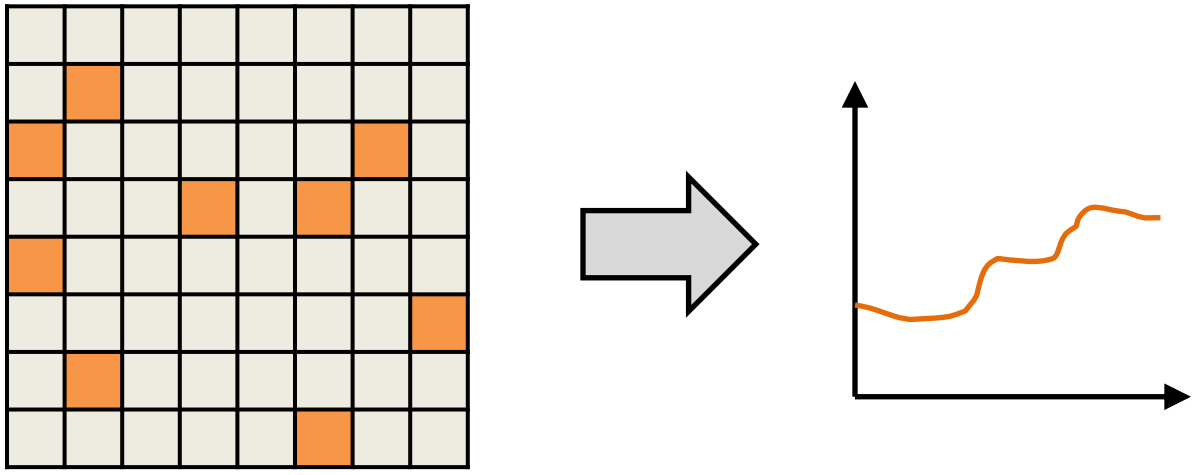


To improve quality, no need to reiterate from beginning; therefore, *diffusive*.
(e.g., just add more samples to current result)

Anytime Automaton – The Approximations

2. Better case: apply **diffusive** approximations.

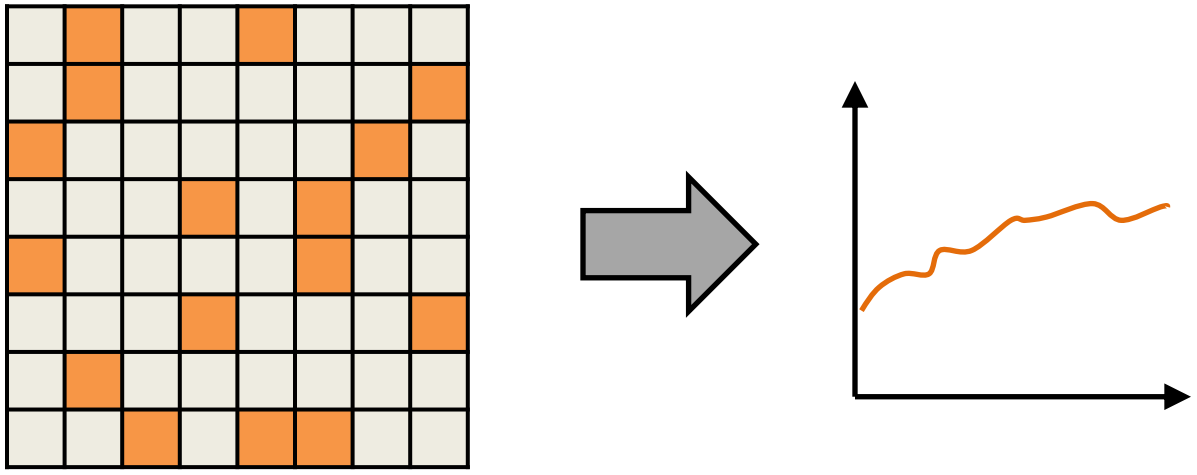
- Input sampling (e.g., generating a distribution)



Anytime Automaton – The Approximations

2. Better case: apply **diffusive** approximations.

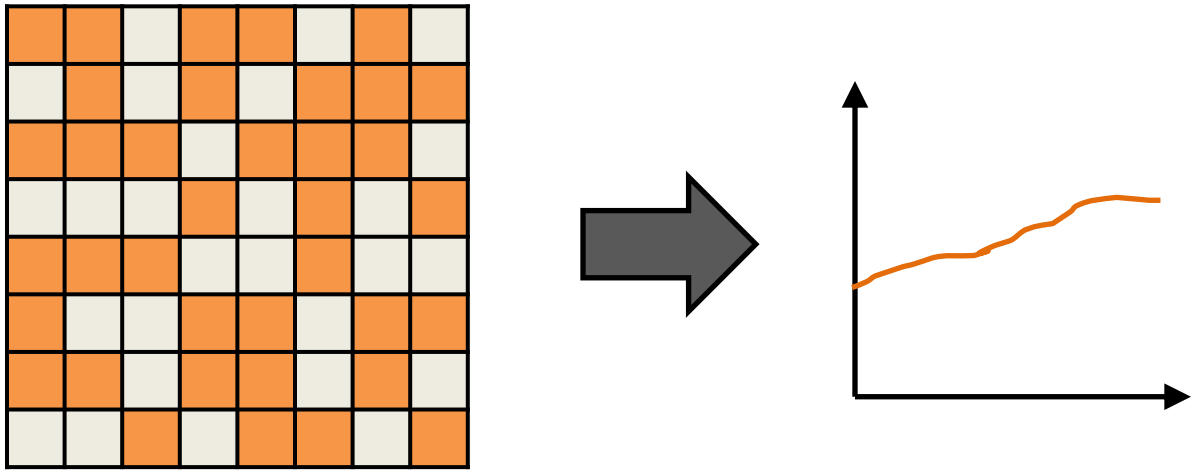
- Input sampling (e.g., generating a distribution)



Anytime Automaton – The Approximations

2. Better case: apply **diffusive** approximations.

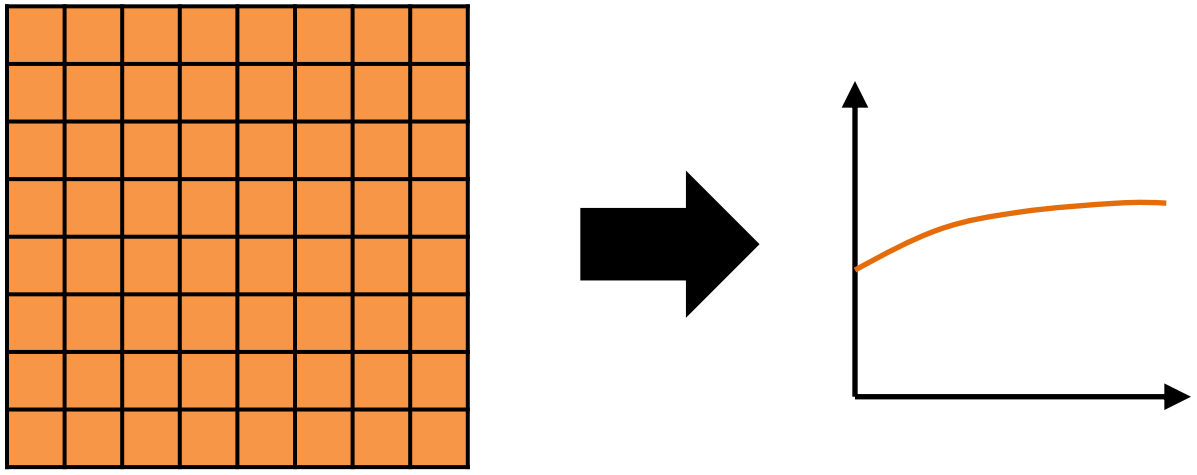
- Input sampling (e.g., generating a distribution)



Anytime Automaton – The Approximations

2. Better case: apply **diffusive** approximations.

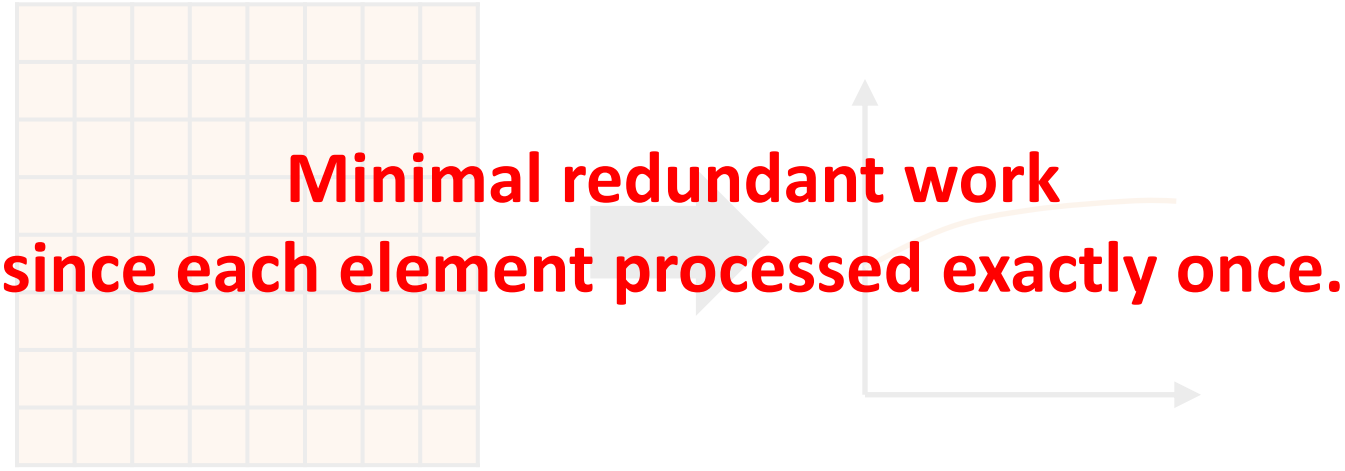
- Input sampling (e.g., generating a distribution)



Anytime Automaton – The Approximations

2. Better case: apply **diffusive** approximations.

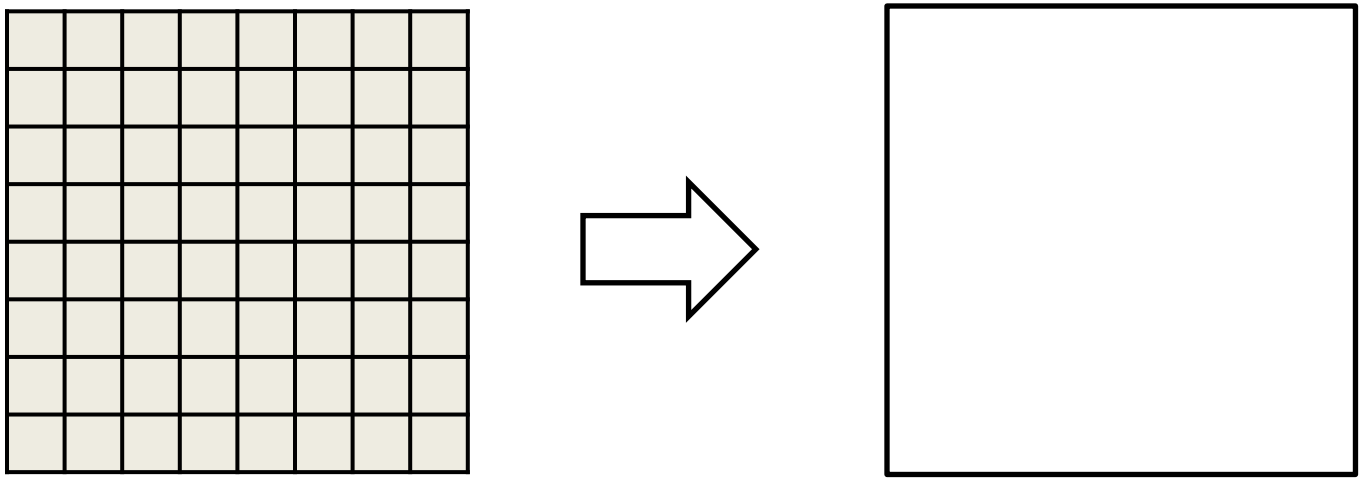
- Input sampling (e.g., generating a distribution)



Anytime Automaton – The Approximations

2. Better case: apply **diffusive** approximations.

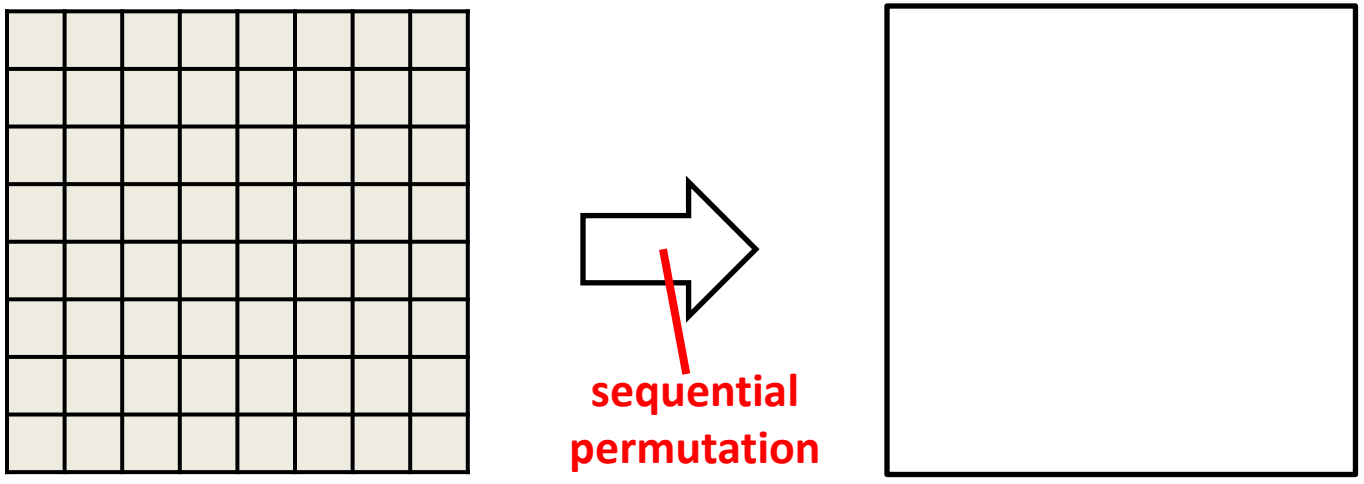
- Output sampling (e.g., generating an image)



Anytime Automaton – The Approximations

2. Better case: apply **diffusive** approximations.

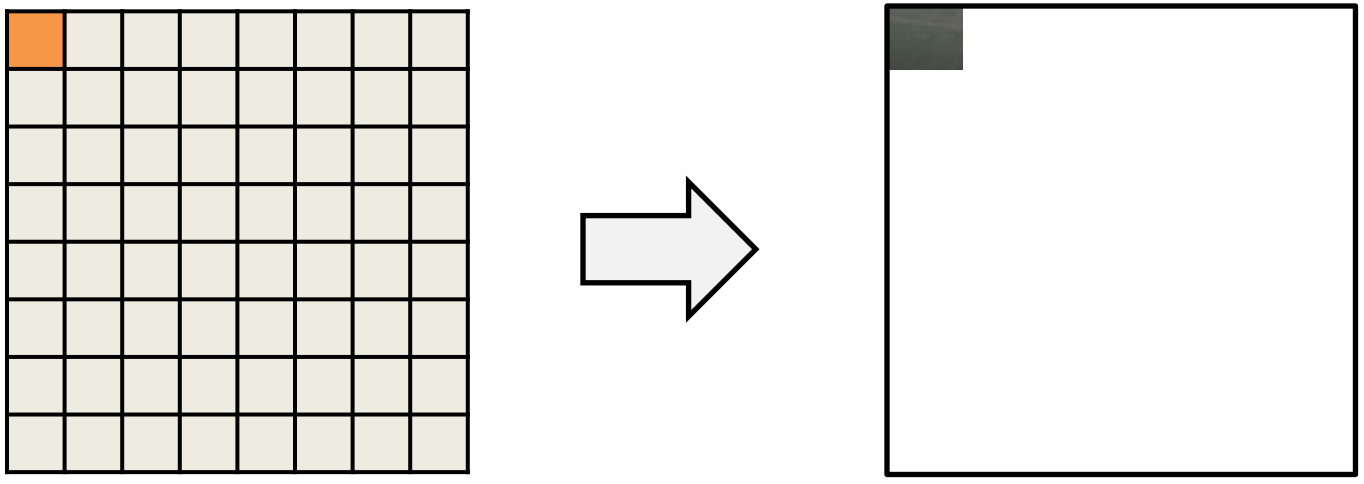
- Output sampling (e.g., generating an image)



Anytime Automaton – The Approximations

2. Better case: apply **diffusive** approximations.

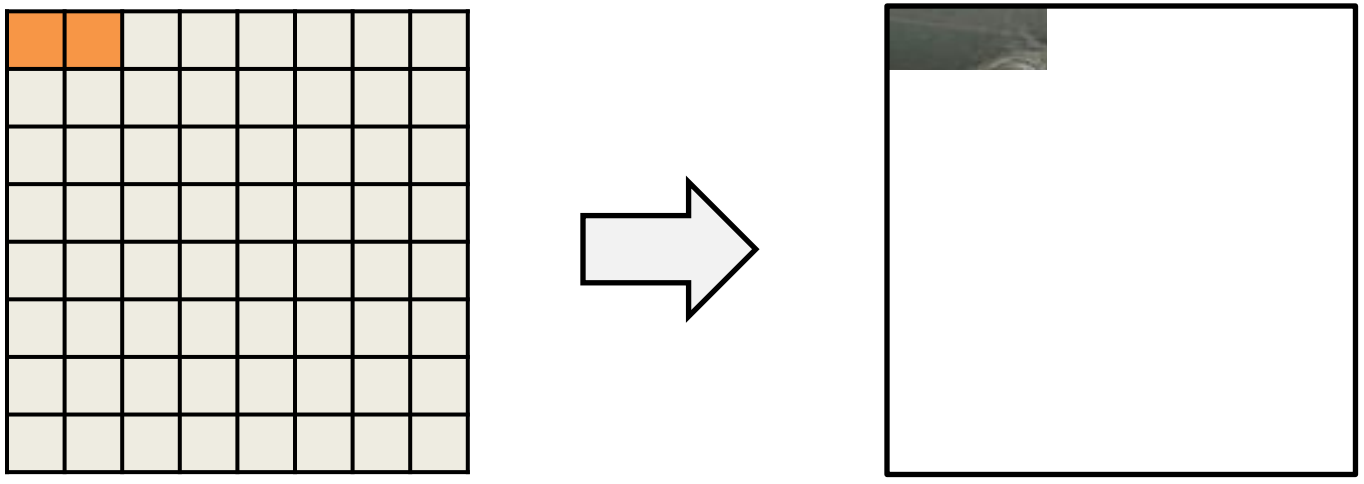
- Output sampling (e.g., generating an image)



Anytime Automaton – The Approximations

2. Better case: apply **diffusive** approximations.

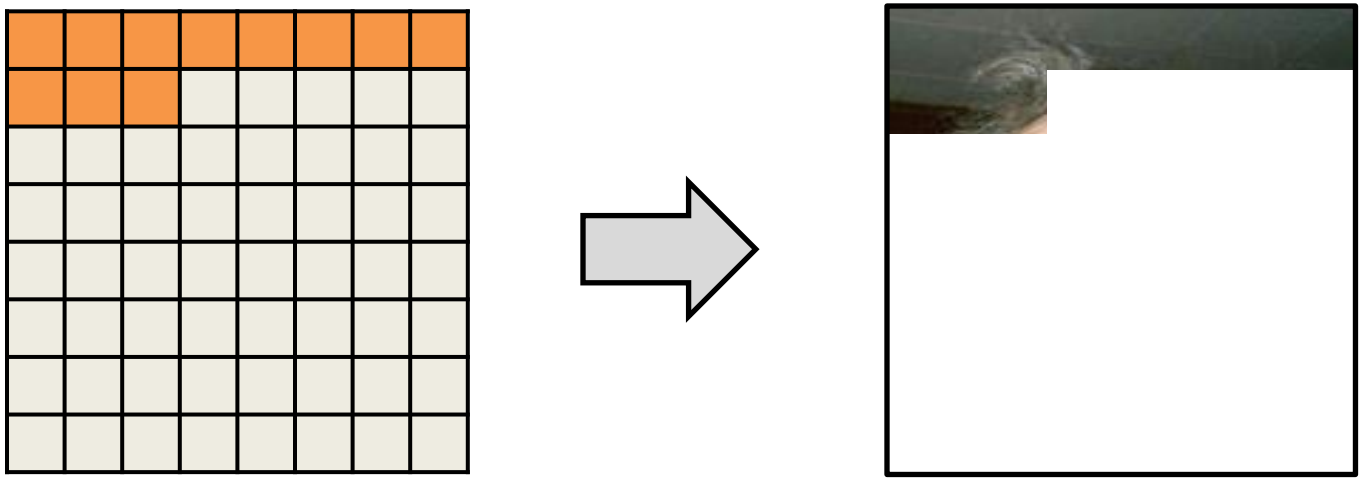
- Output sampling (e.g., generating an image)



Anytime Automaton – The Approximations

2. Better case: apply **diffusive** approximations.

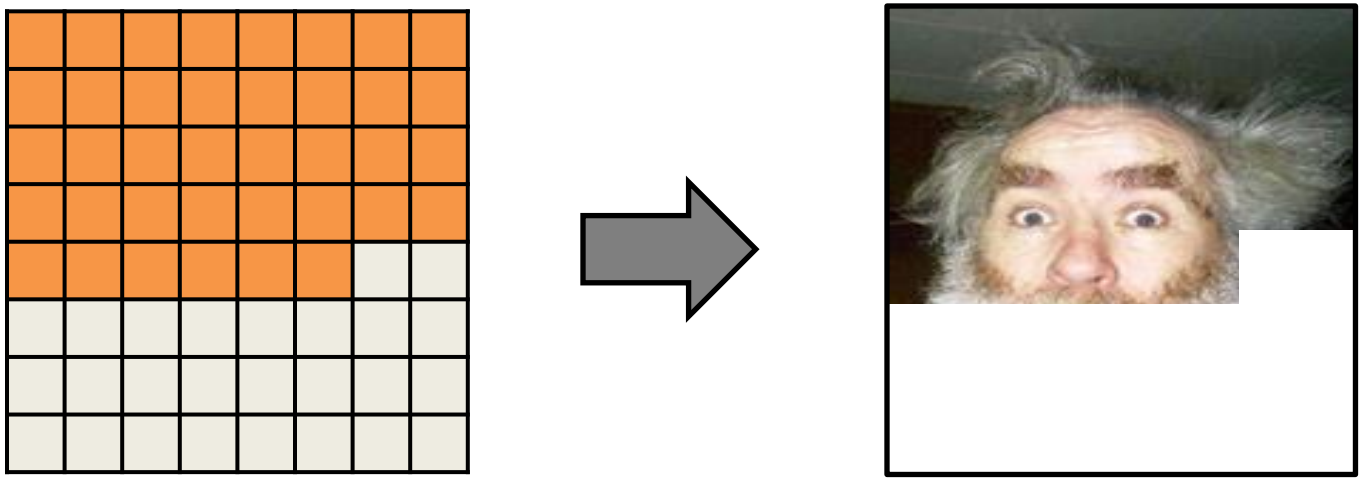
- Output sampling (e.g., generating an image)



Anytime Automaton – The Approximations

2. Better case: apply **diffusive** approximations.

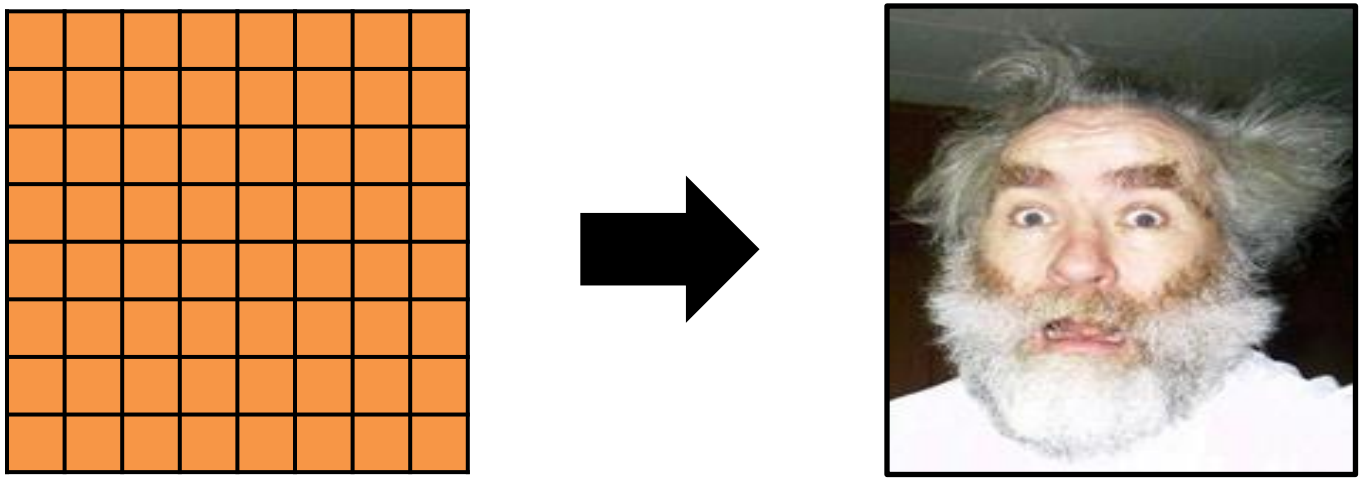
- Output sampling (e.g., generating an image)



Anytime Automaton – The Approximations

2. Better case: apply **diffusive** approximations.

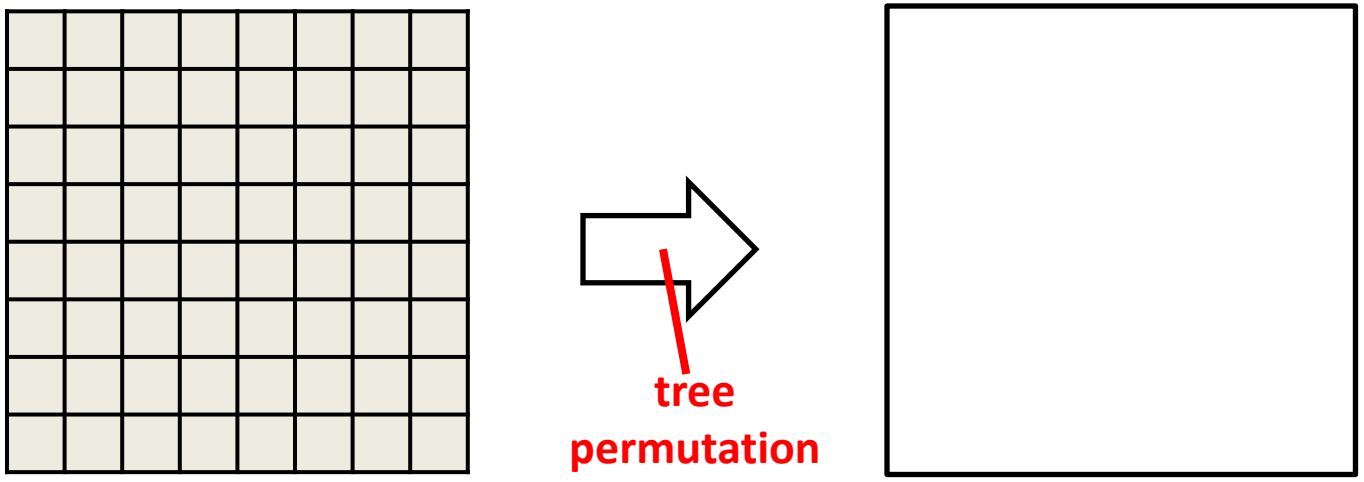
- Output sampling (e.g., generating an image)



Anytime Automaton – The Approximations

2. Better case: apply **diffusive** approximations.

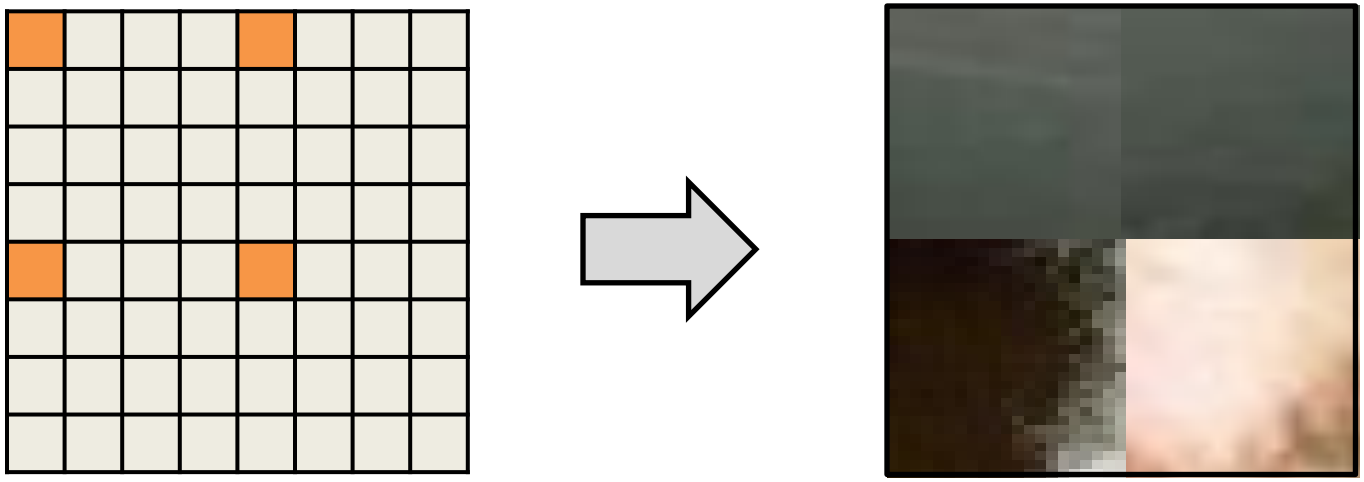
- Output sampling (e.g., generating an image)



Anytime Automaton – The Approximations

2. Better case: apply **diffusive** approximations.

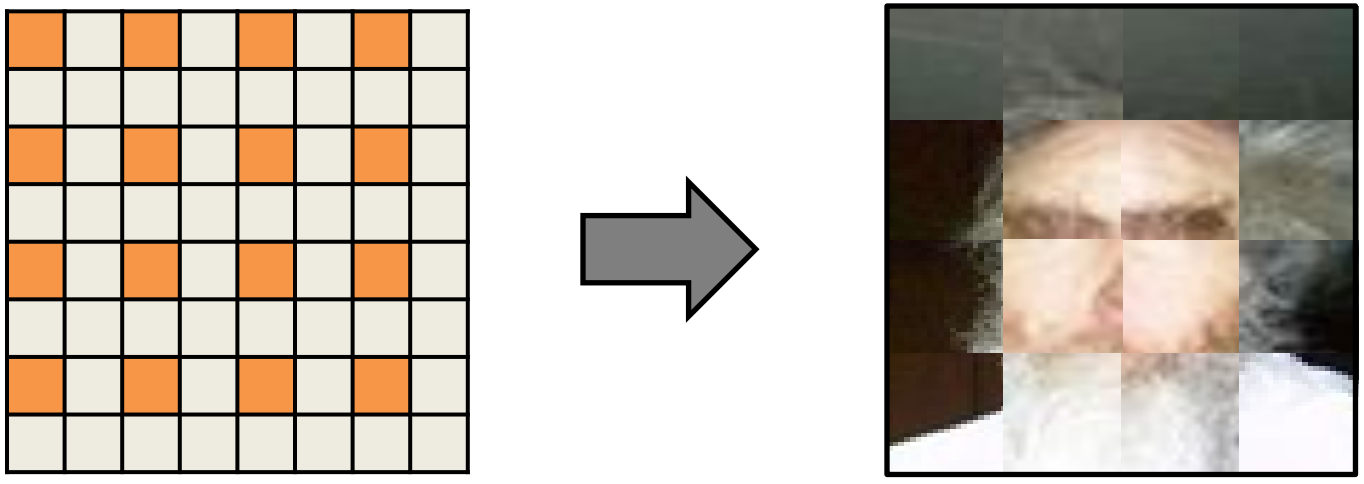
- Output sampling (e.g., generating an image)



Anytime Automaton – The Approximations

2. Better case: apply **diffusive** approximations.

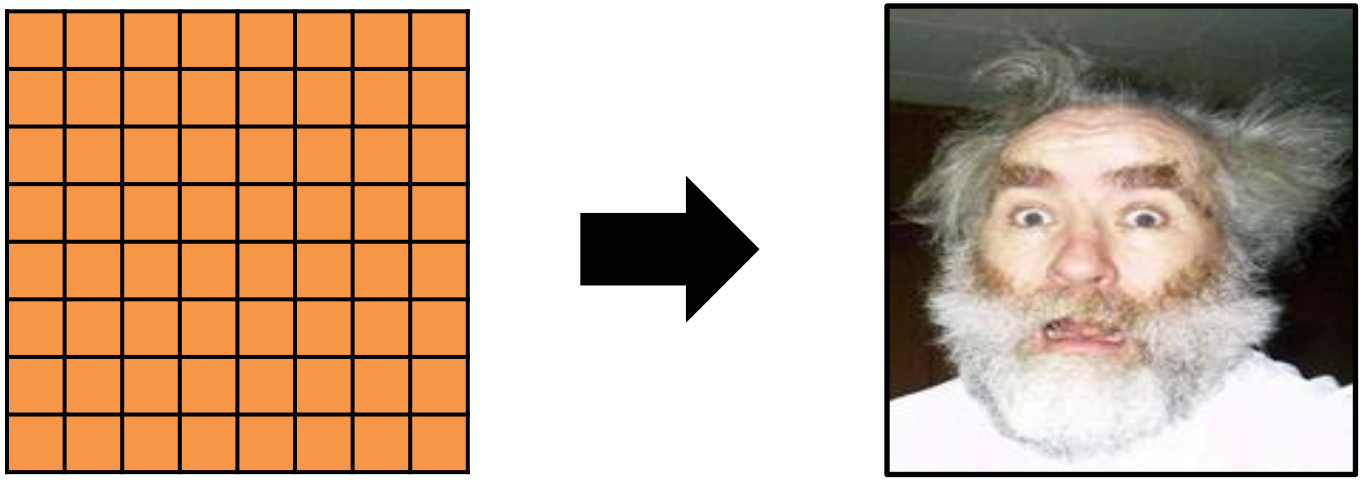
- Output sampling (e.g., generating an image)



Anytime Automaton – The Approximations

2. Better case: apply **diffusive** approximations.

- Output sampling (e.g., generating an image)



Anytime Automaton – The Approximations

2. Better case: apply **diffusive** approximations.

- Integer/fixed-point precision (e.g., dot product)

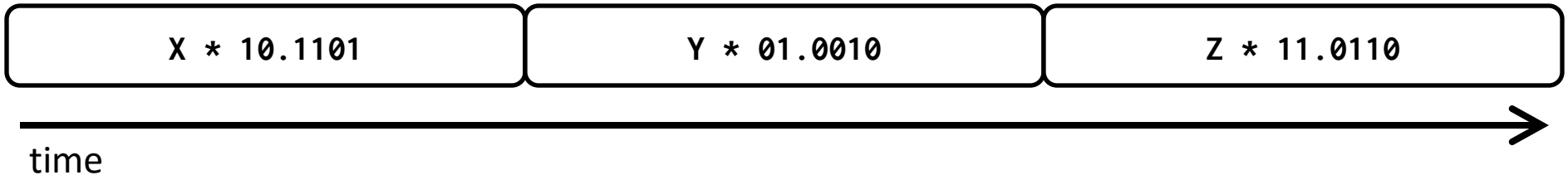
$$[\ x\ \ y\ \ z\ ] \bullet [\ 10.1101\ \ 01.0010\ \ 11.0110\ ]$$

Anytime Automaton – The Approximations

2. Better case: apply **diffusive** approximations.

- Integer/fixed-point precision (e.g., dot product)

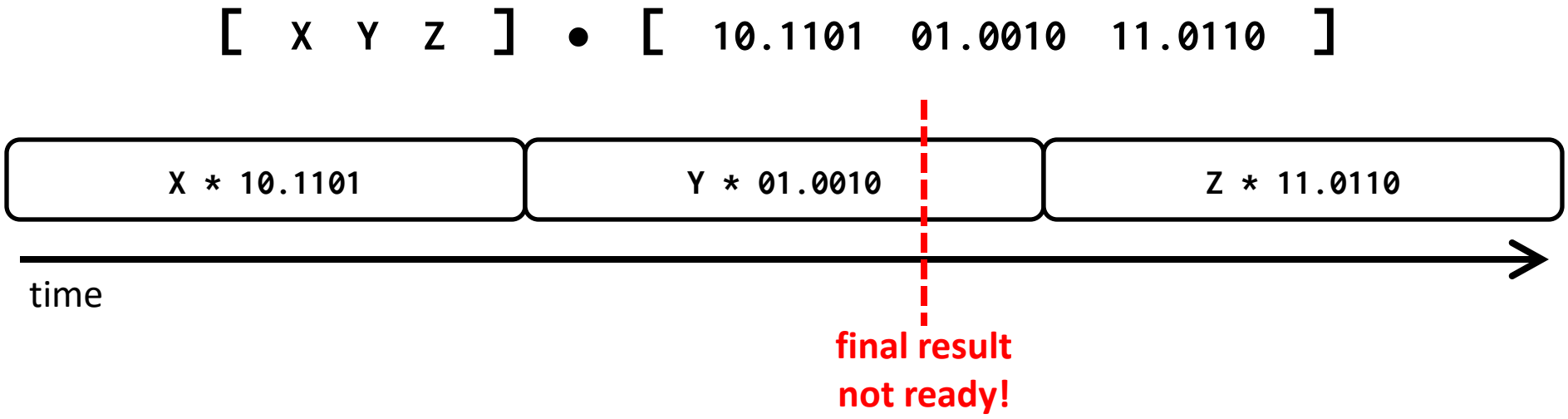
$$[\ X\ \ Y\ \ Z\ ] \bullet [\ 10.1101\ \ 01.0010\ \ 11.0110\ ]$$



Anytime Automaton – The Approximations

2. Better case: apply **diffusive** approximations.

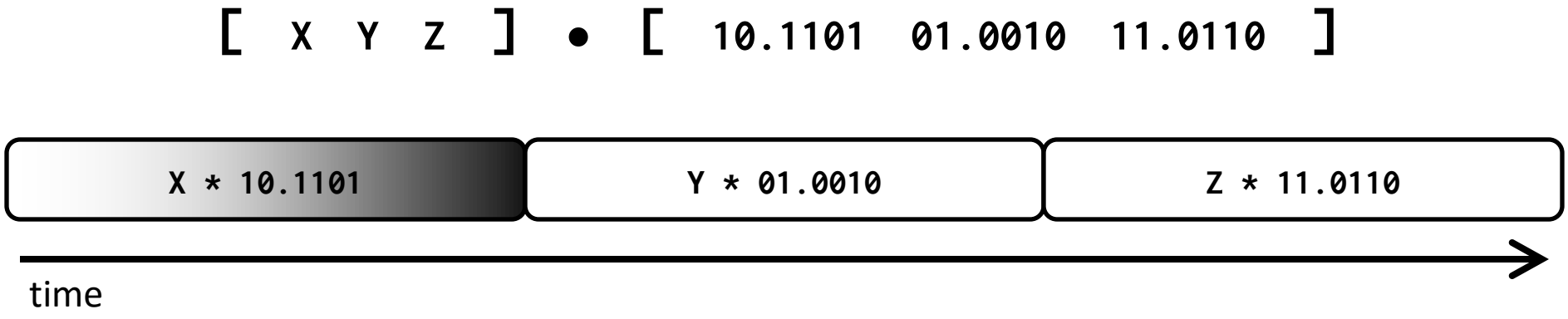
- Integer/fixed-point precision (e.g., dot product)



Anytime Automaton – The Approximations

2. Better case: apply **diffusive** approximations.

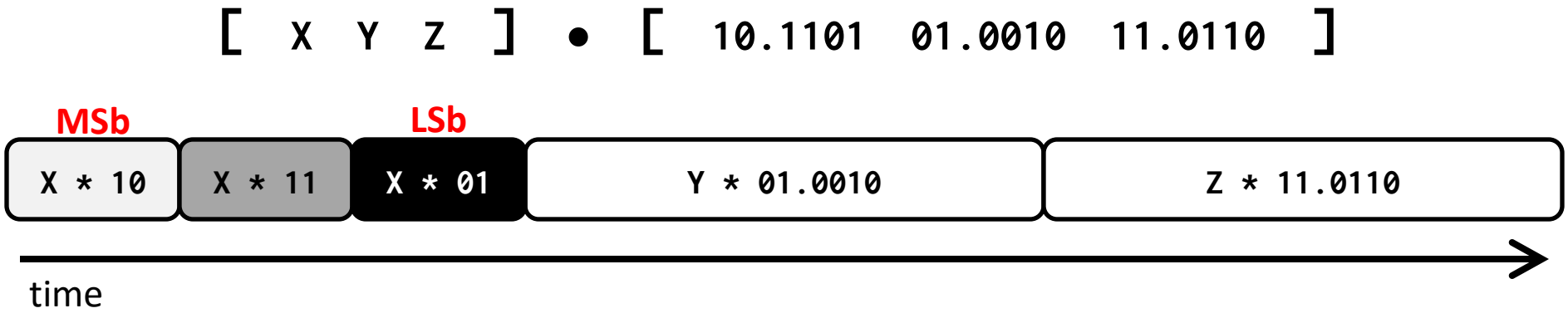
- Integer/fixed-point precision (e.g., dot product)



Anytime Automaton – The Approximations

2. Better case: apply **diffusive** approximations.

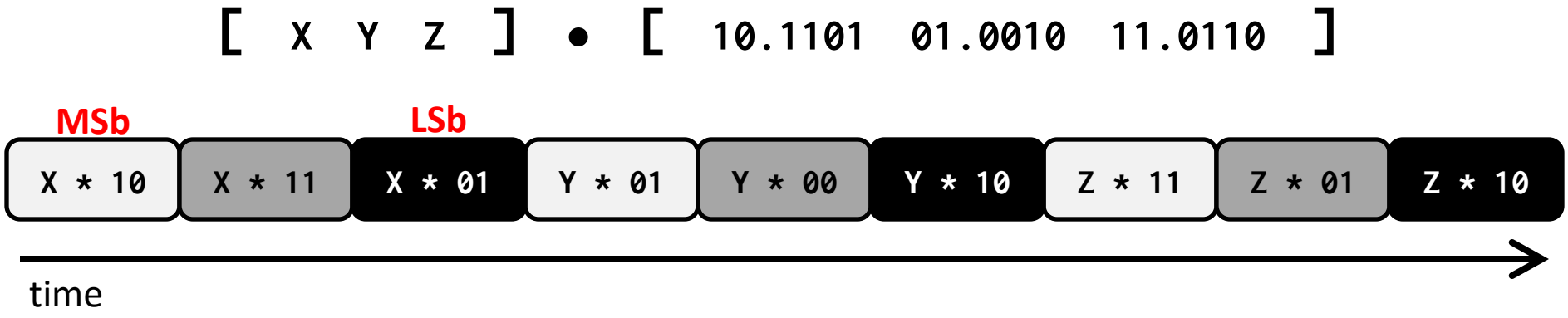
- Integer/fixed-point precision (e.g., dot product)



Anytime Automaton – The Approximations

2. Better case: apply **diffusive** approximations.

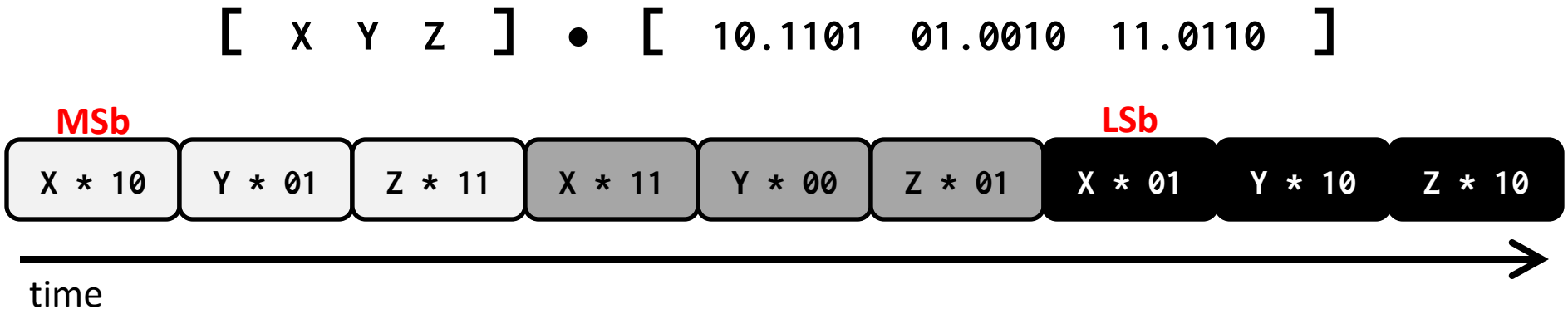
- Integer/fixed-point precision (e.g., dot product)



Anytime Automaton – The Approximations

2. Better case: apply **diffusive** approximations.

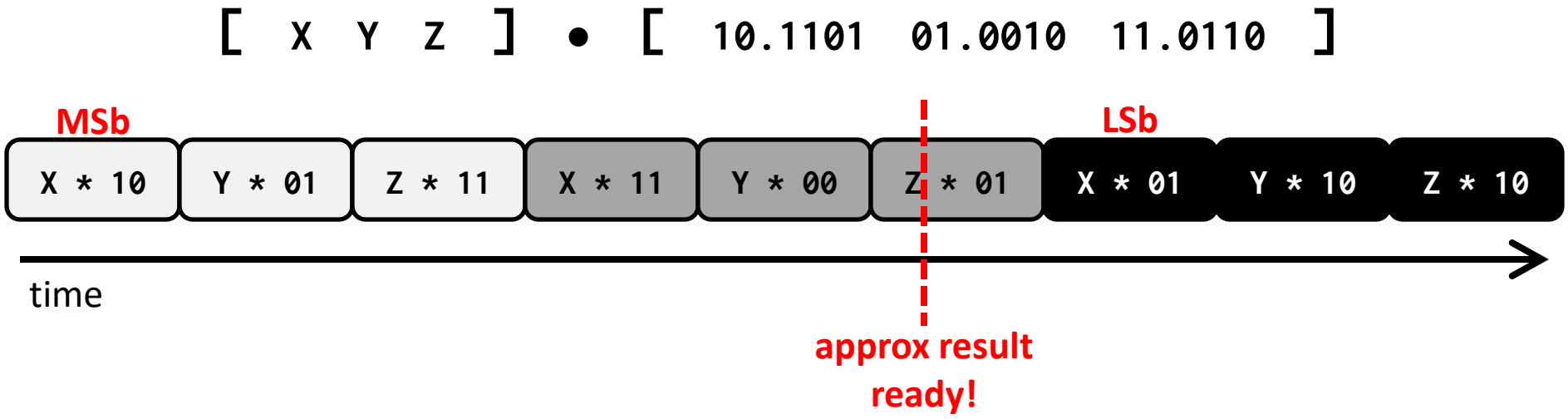
- Integer/fixed-point precision (e.g., dot product)



Anytime Automaton – The Approximations

2. Better case: apply **diffusive** approximations.

- Integer/fixed-point precision (e.g., dot product)



Anytime Automaton

More details in paper:

- Asynchronous/synchronous pipelining
- Data locality with sampling
- Approximate storage techniques
- Thread scheduling

Evaluation – Methodology

Experiments:

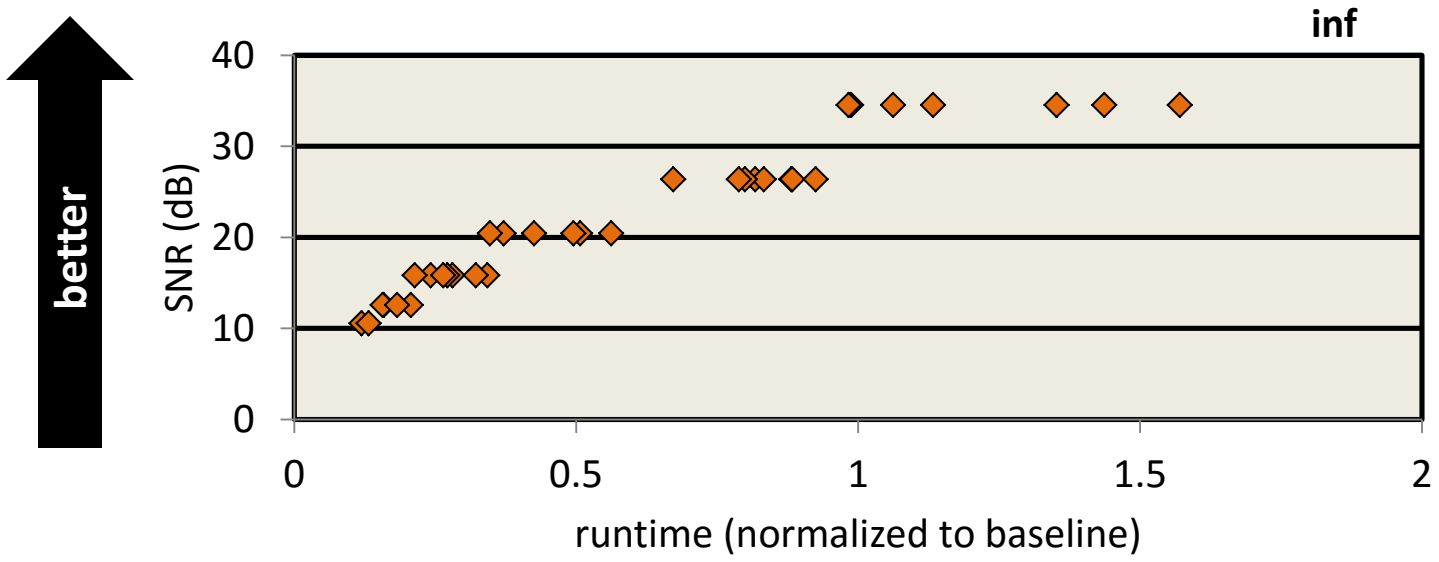
- IBM Power 780 system
  - 4 POWER7+ cores
  - 32 total hardware threads

Applications:

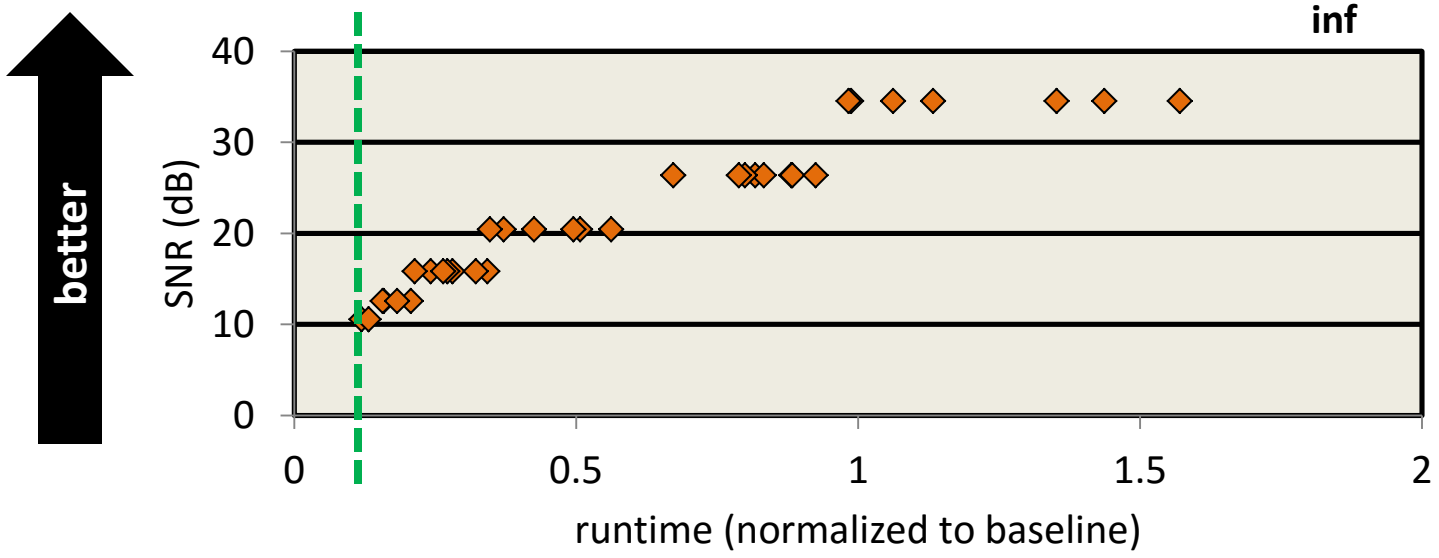
- PERFECT and AxBench suites
  - 2D convolution (output sampling, reduced precision†, SRAM bit upsets†)
  - debayer (output sampling)
  - discrete wavelet transform (loop perforation)
  - histogram equalization (input and output sampling)
  - k-means clustering (output sampling)

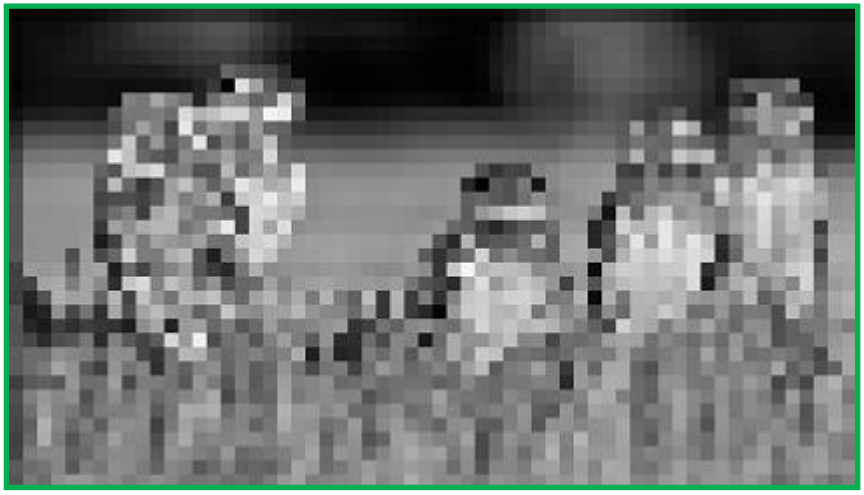
†see paper

Evaluation – 2D Convolution

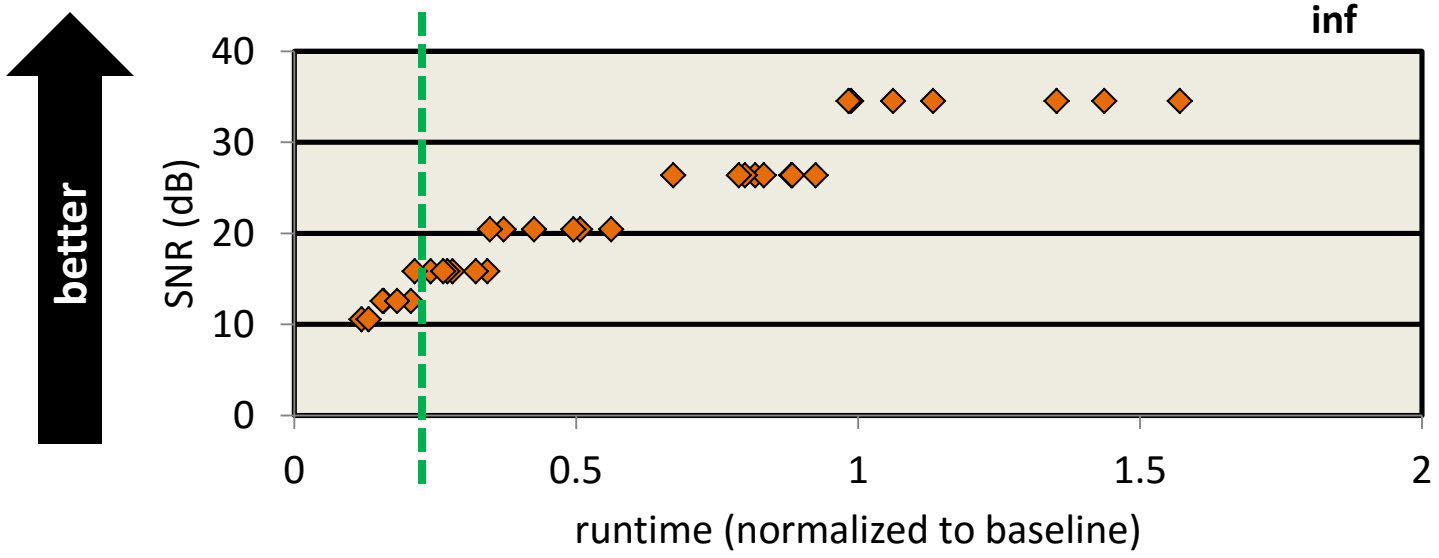


Evaluation – 2D Convolution

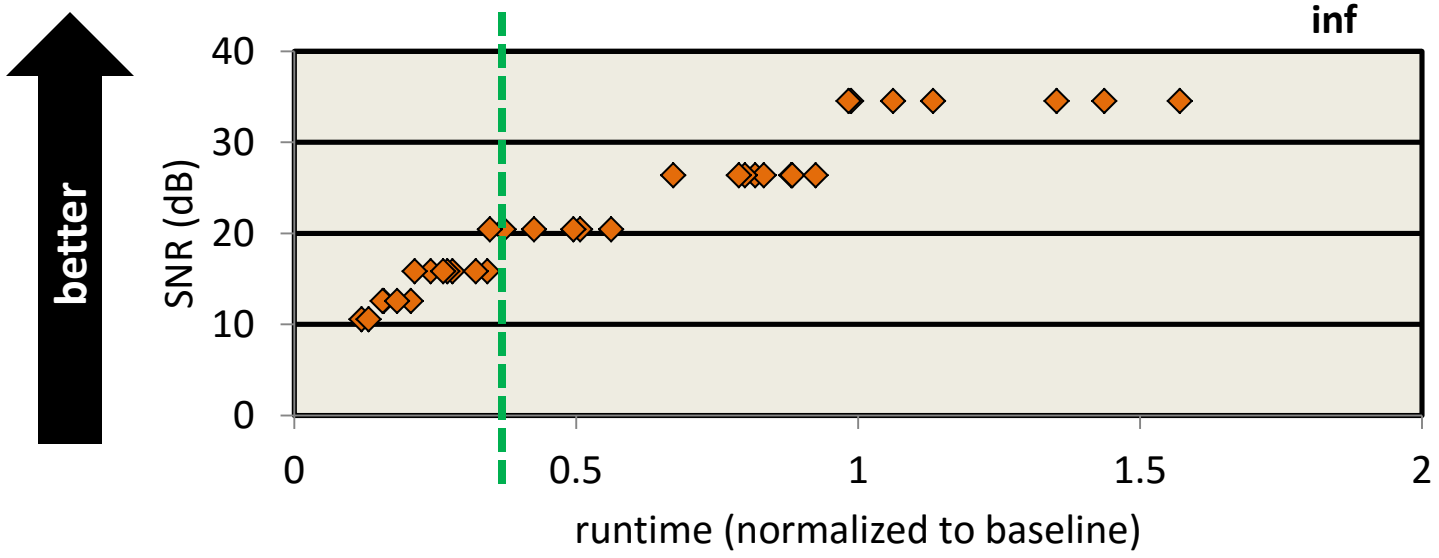




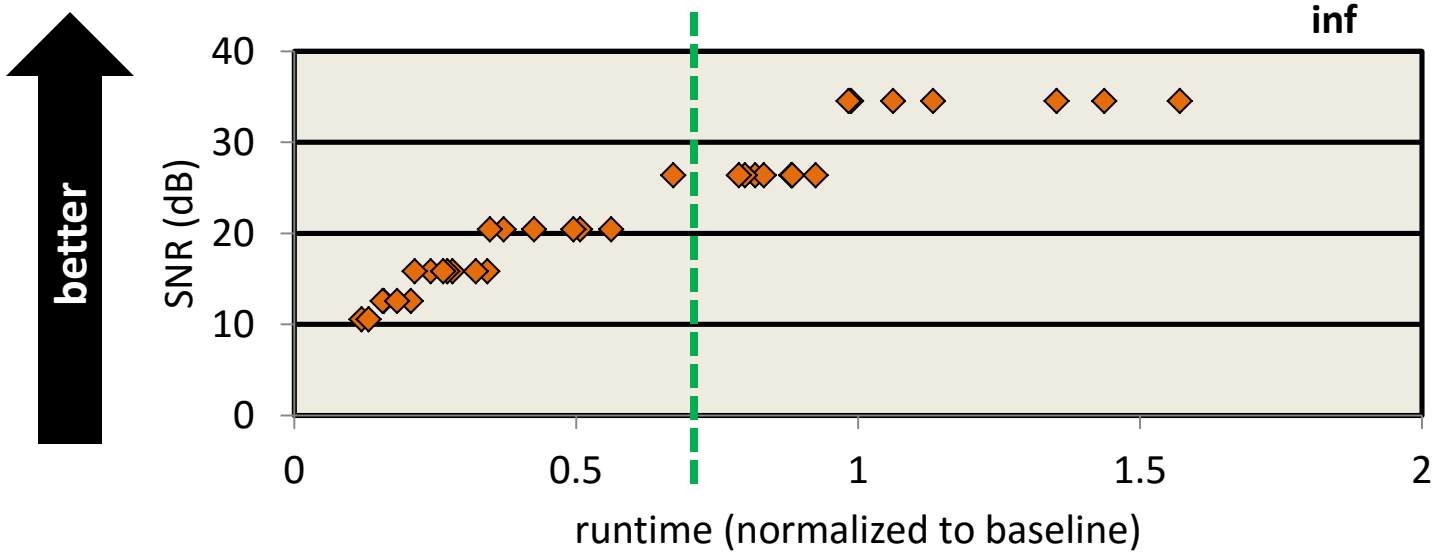
Evaluation – 2D Convolution



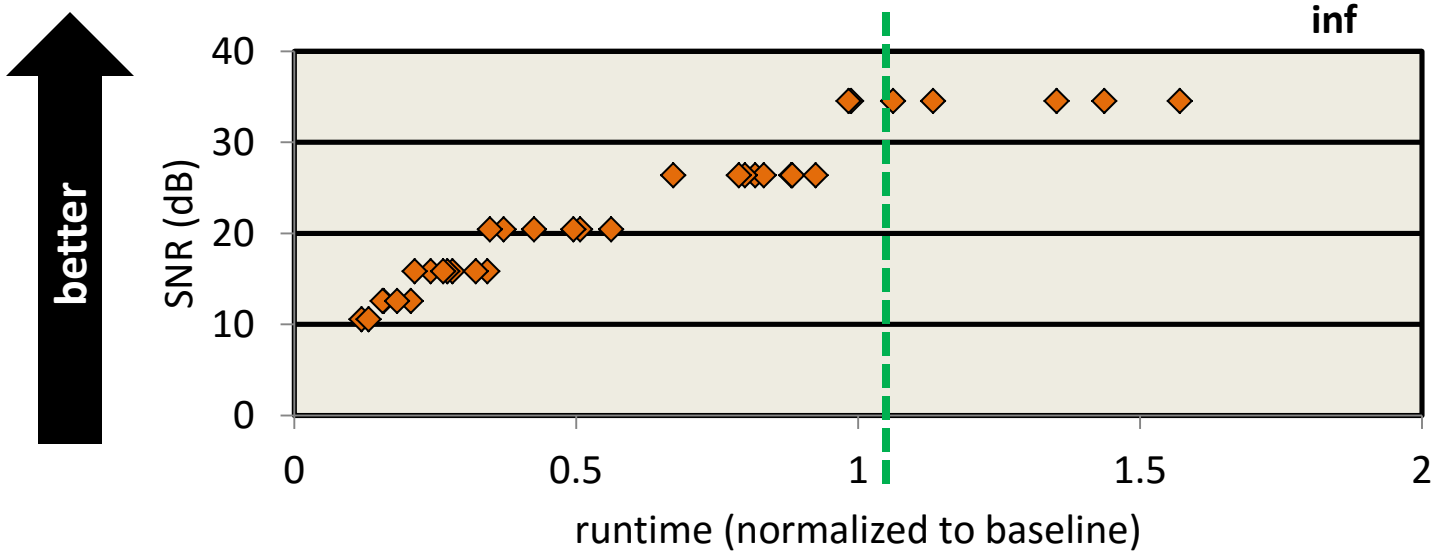
Evaluation – 2D Convolution



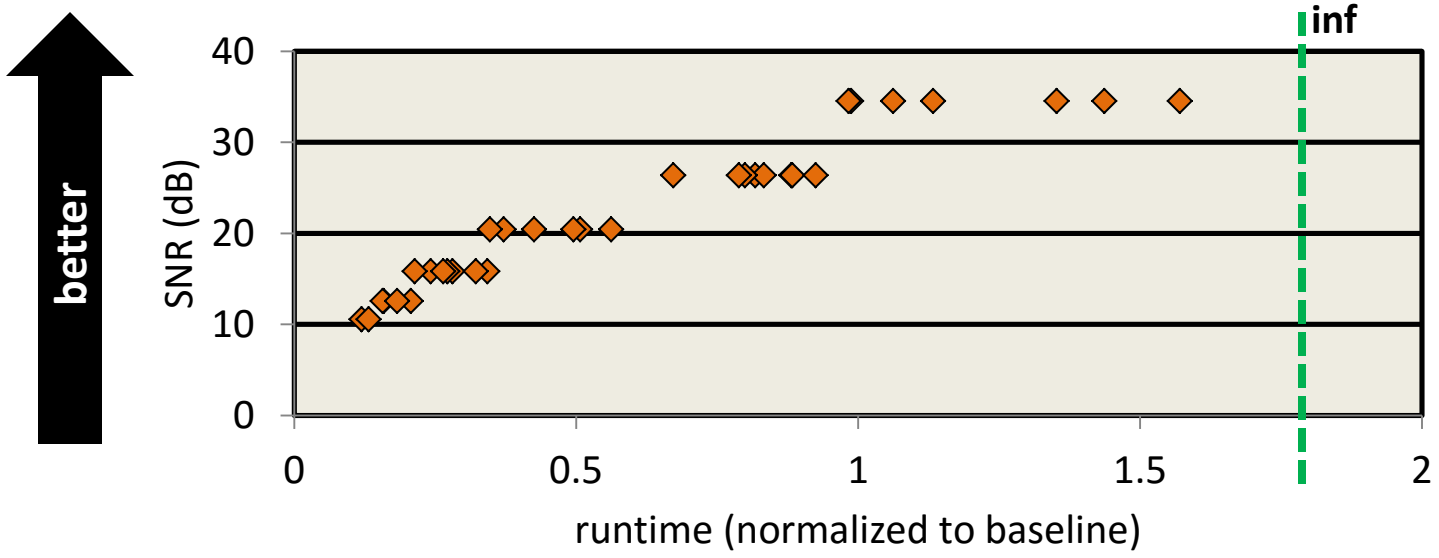
Evaluation – 2D Convolution



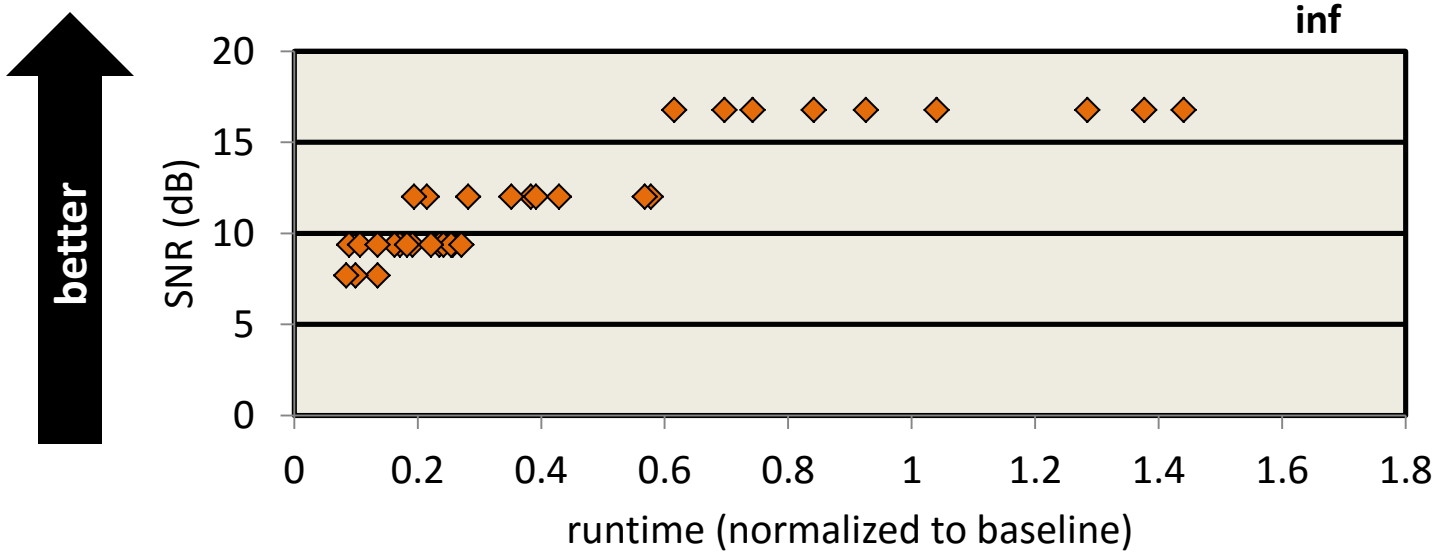
Evaluation – 2D Convolution



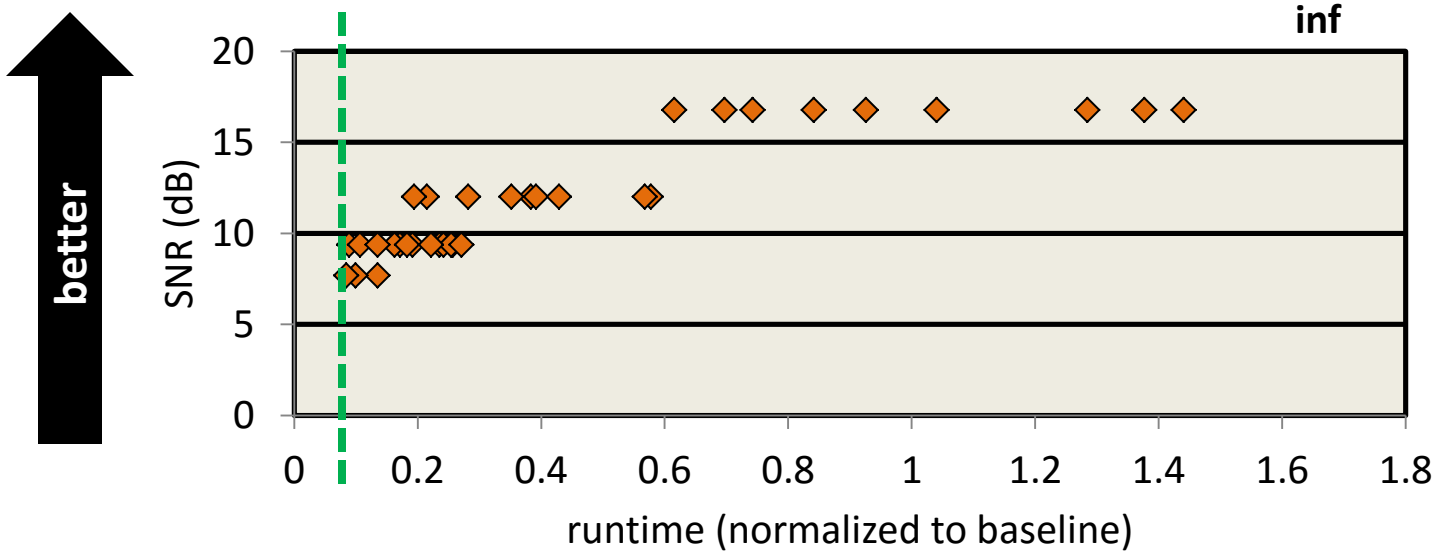
Evaluation – 2D Convolution

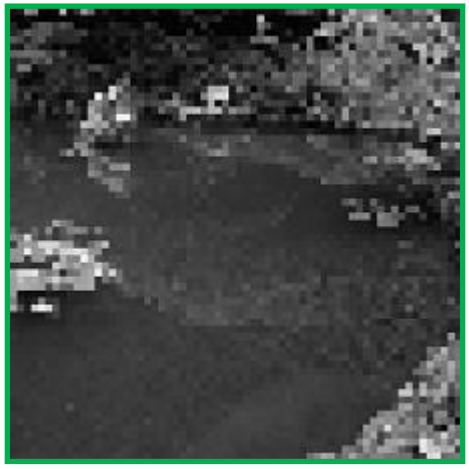


Evaluation – Debayer

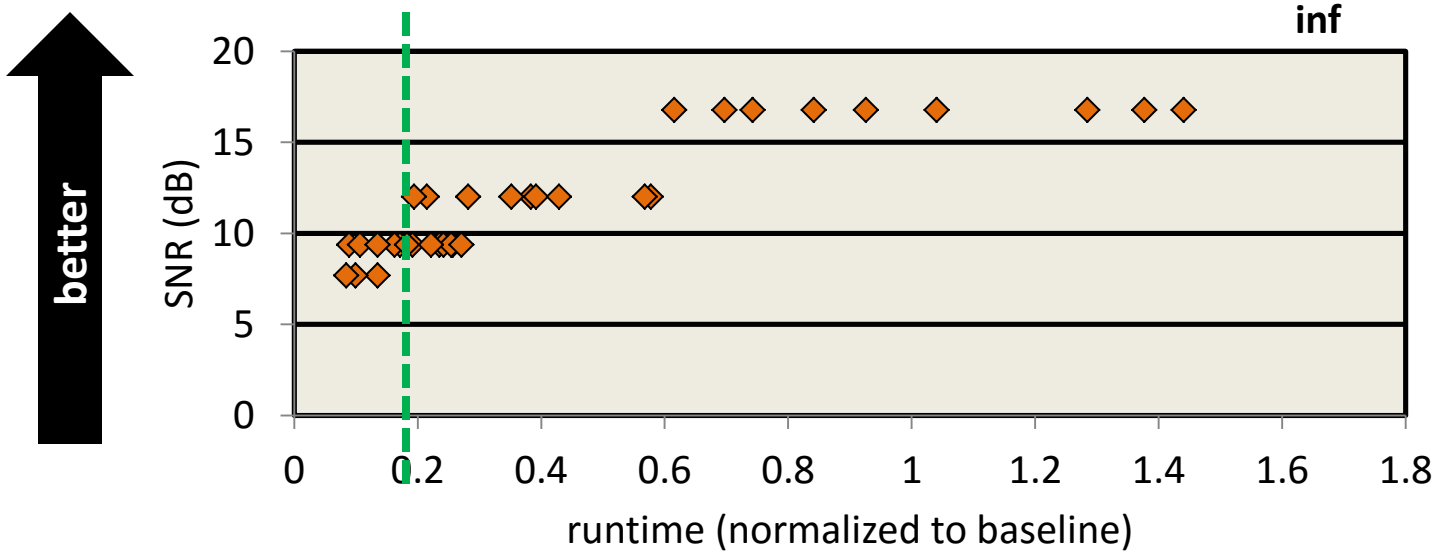


Evaluation – Debayer

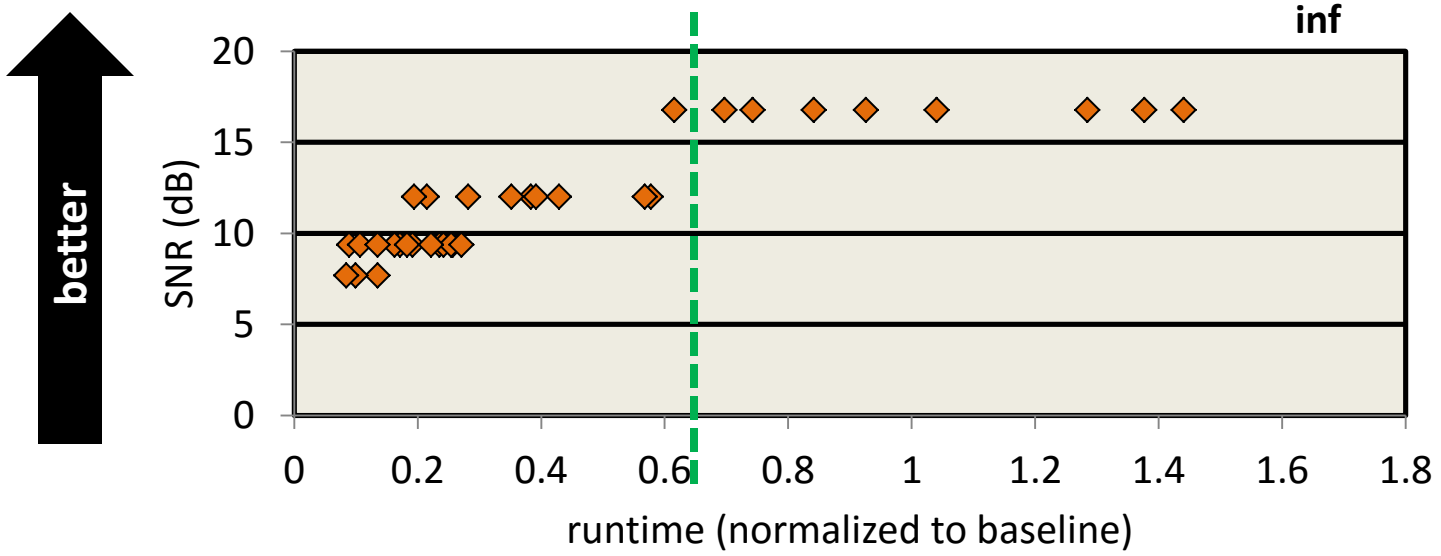




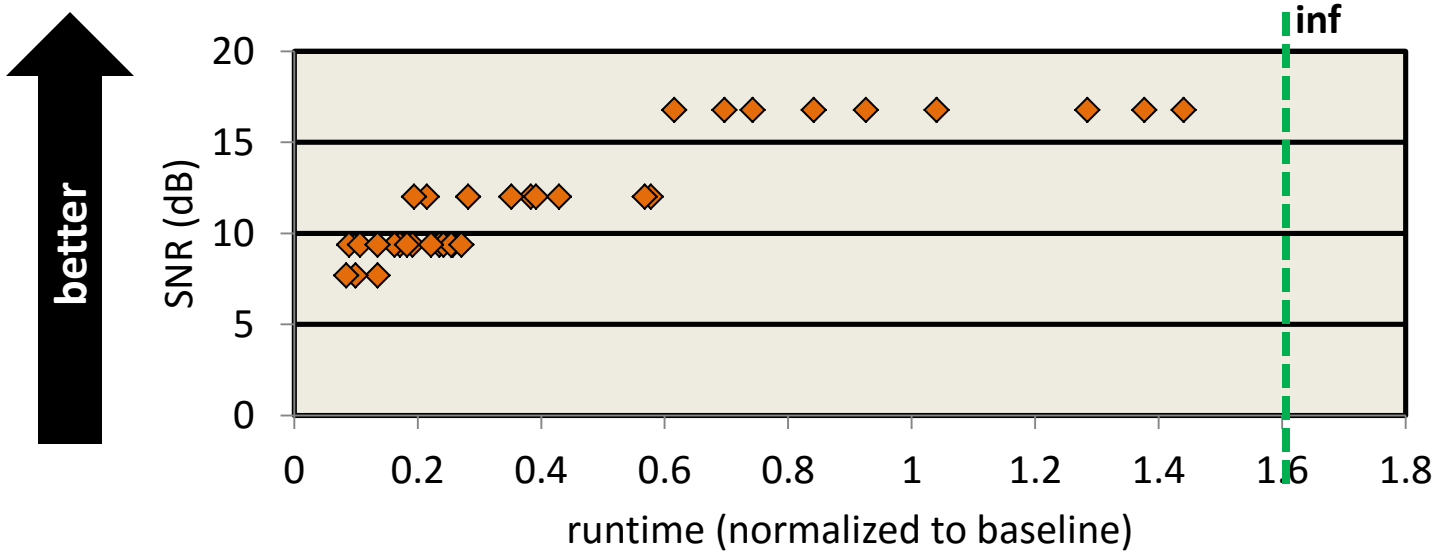
Evaluation – Debayer



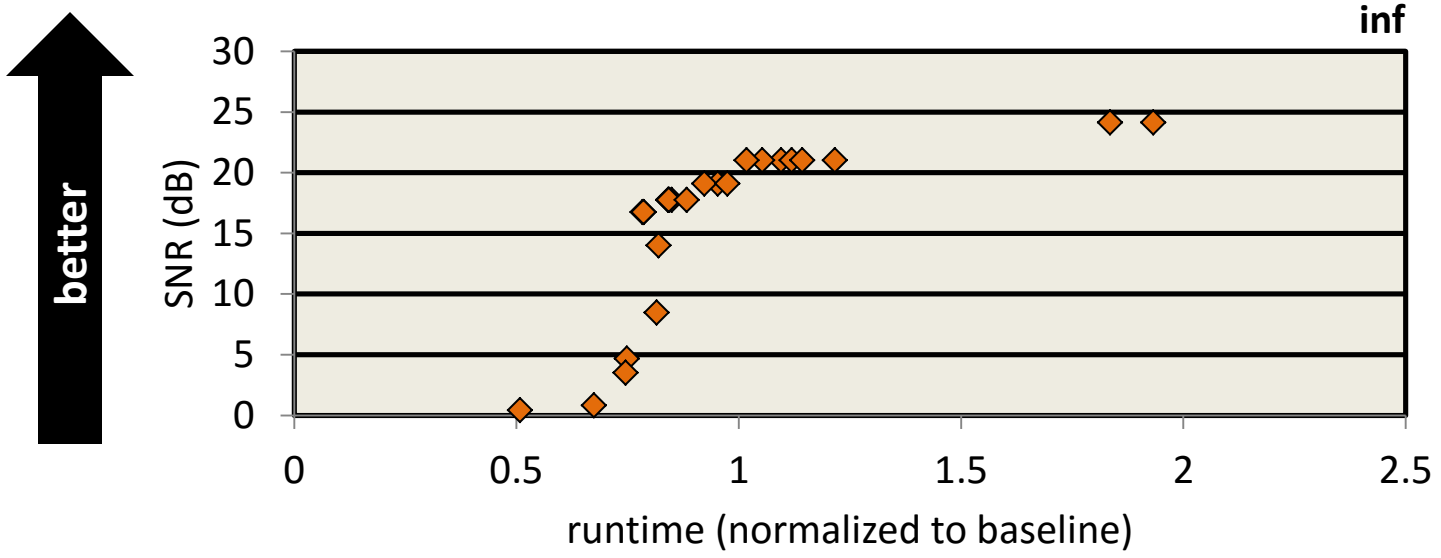
Evaluation – Debayer



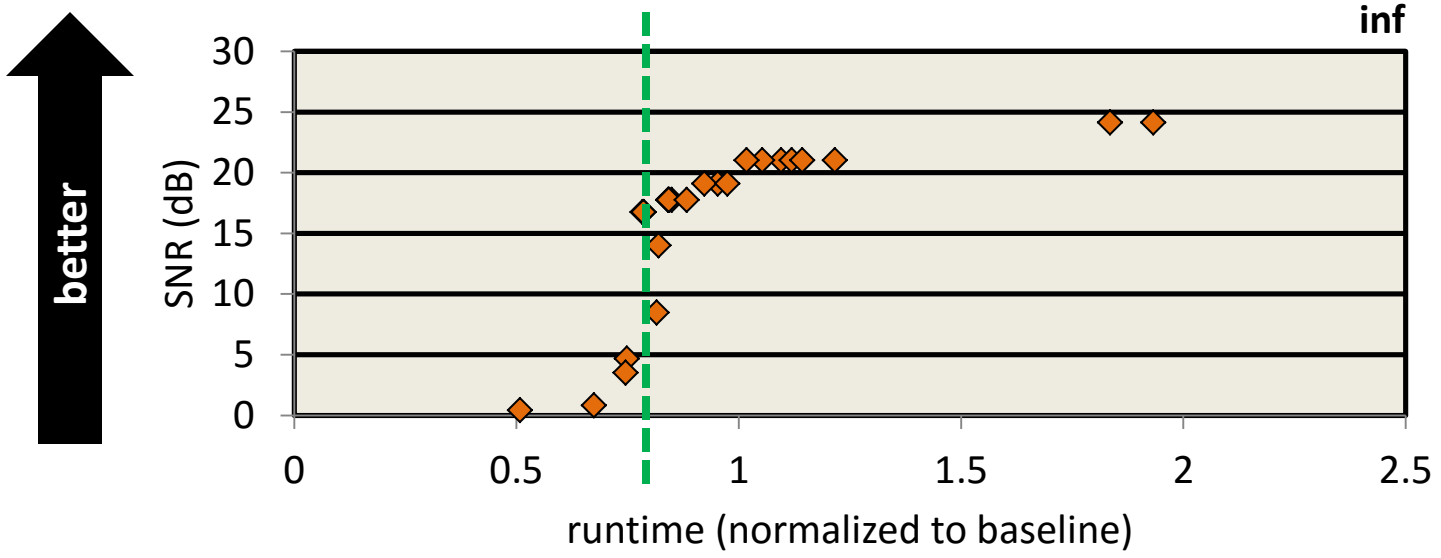
Evaluation – Debayer



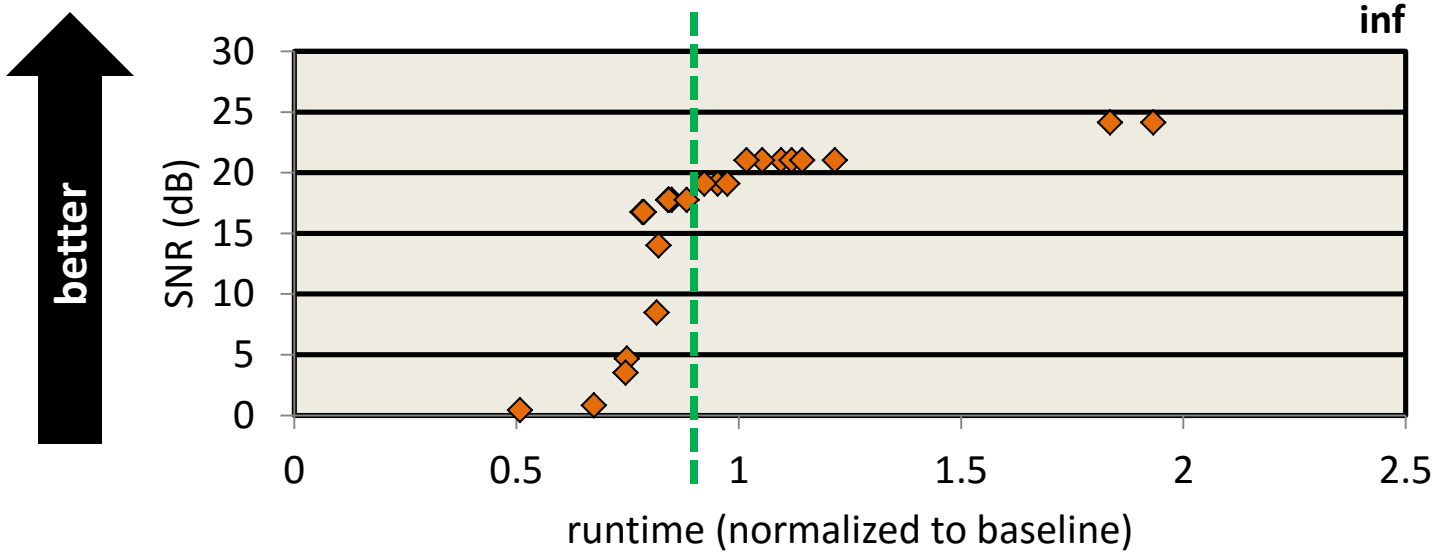
Evaluation – Discrete Wavelet Transform



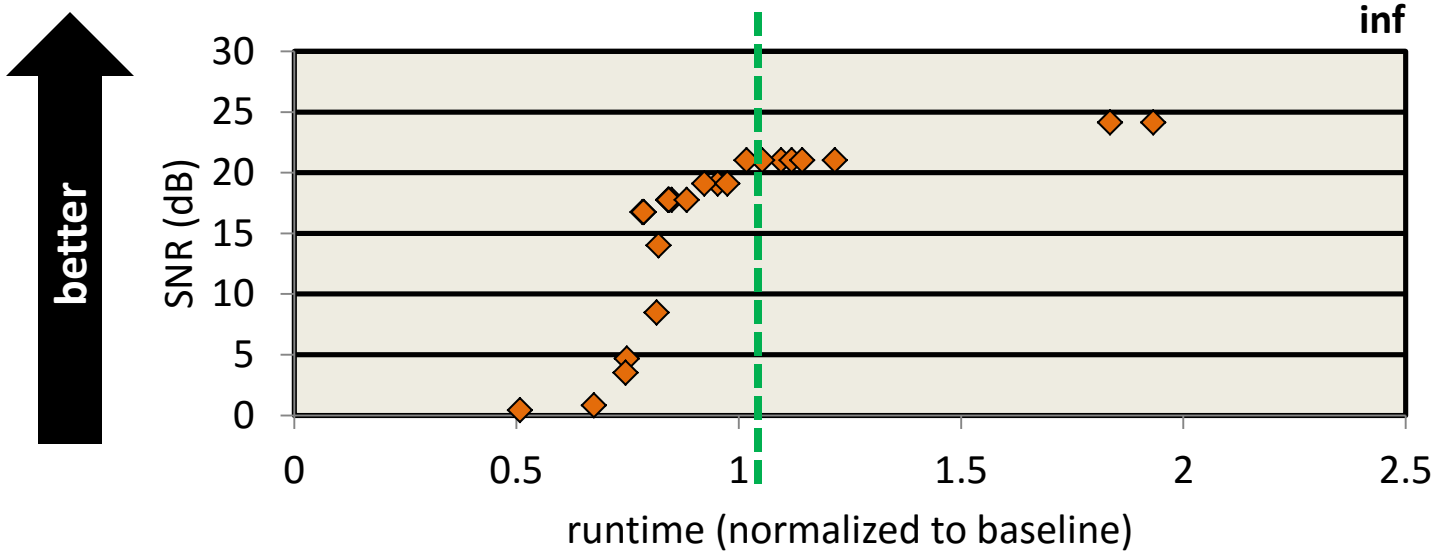
Evaluation – Discrete Wavelet Transform



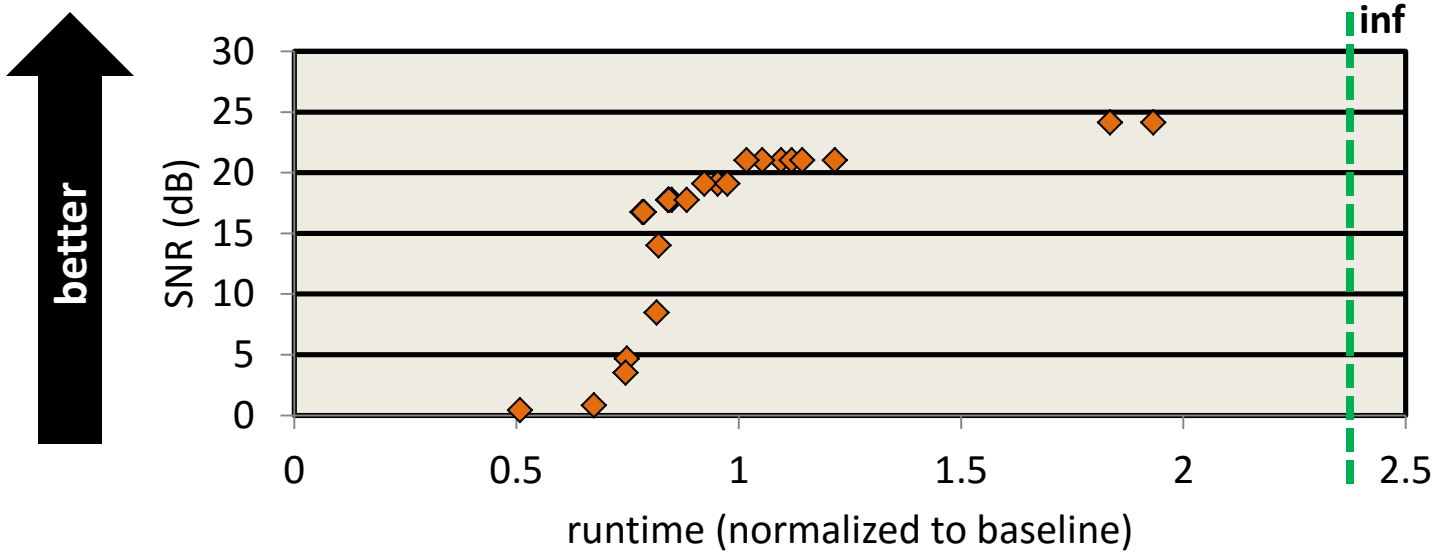
Evaluation – Discrete Wavelet Transform



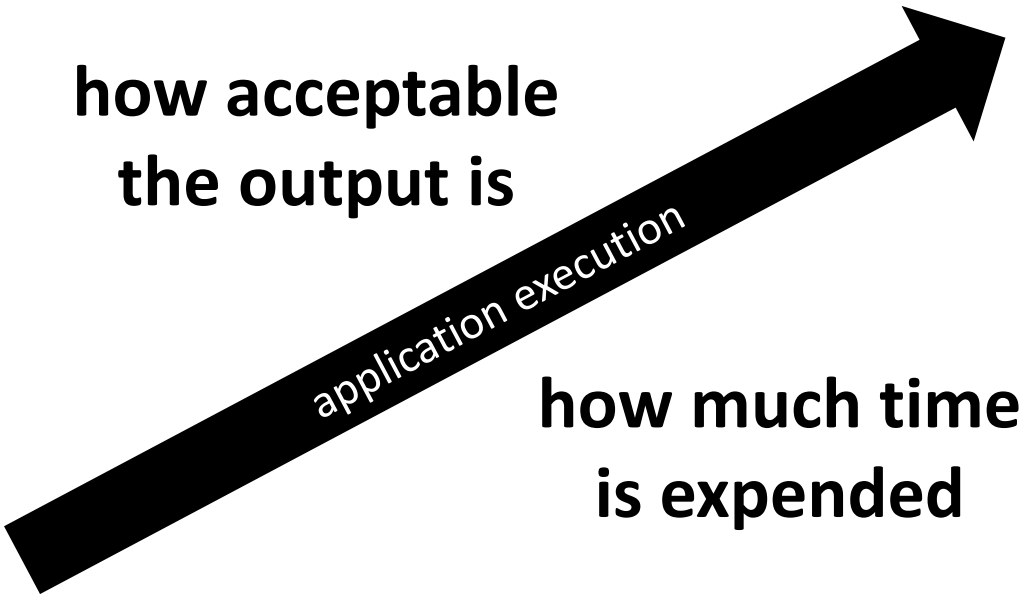
Evaluation – Discrete Wavelet Transform



Evaluation – Discrete Wavelet Transform



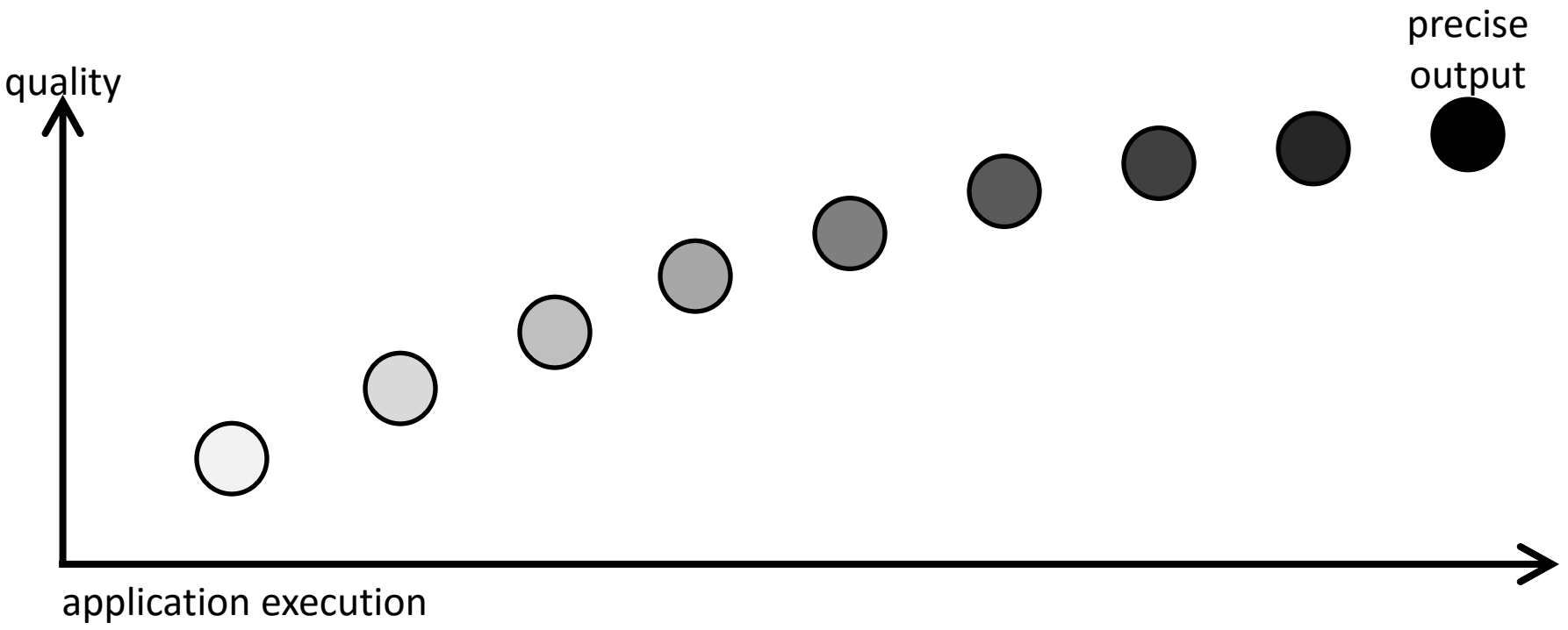
Evaluation – Summary



Conclusion

We propose the **Anytime Automaton**:

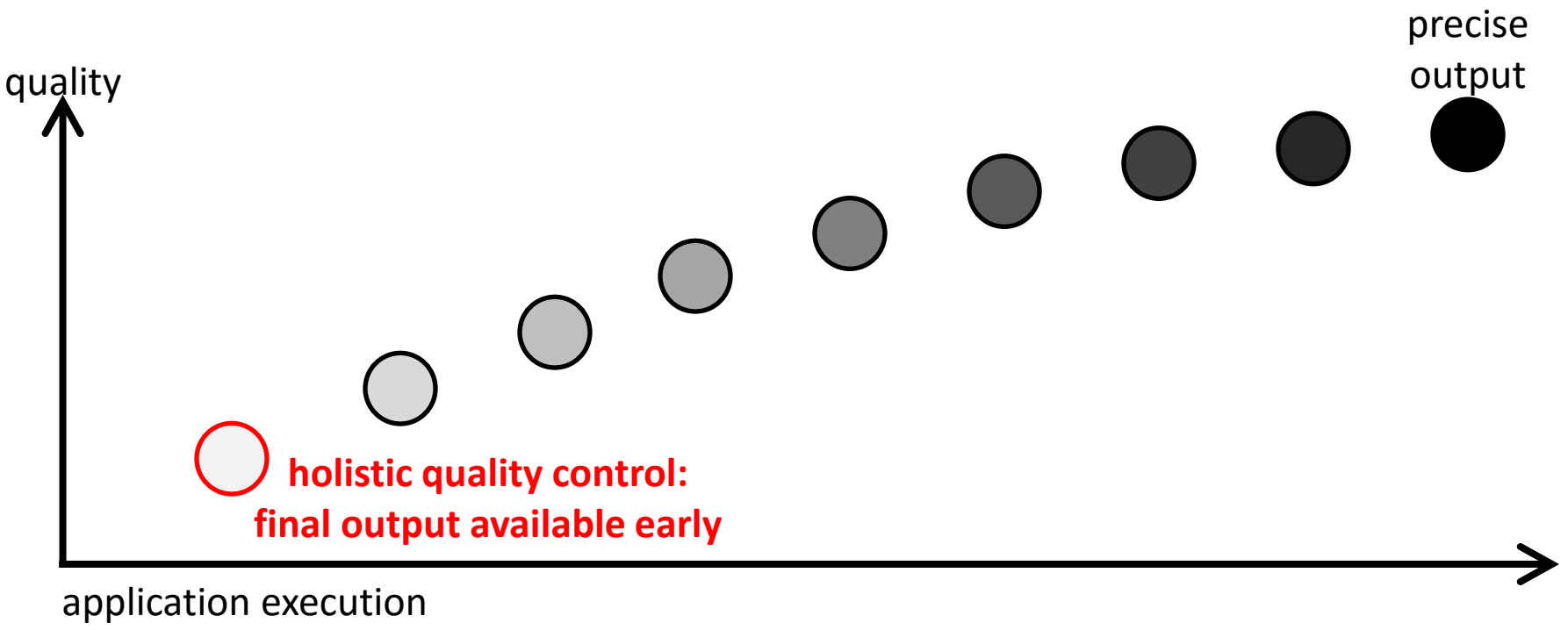
- A new computation model for approximate computing.



Conclusion

We propose the **Anytime Automaton**:

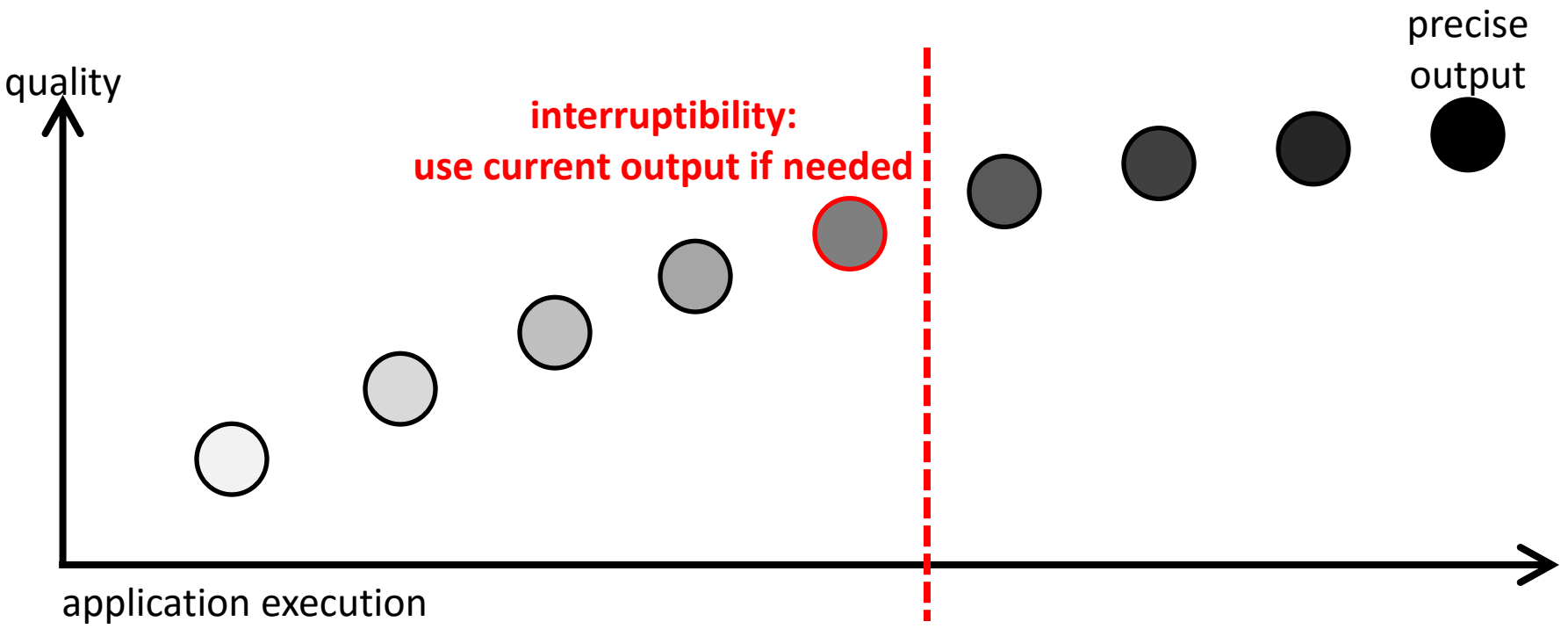
- A new computation model for approximate computing.



Conclusion

We propose the **Anytime Automaton**:

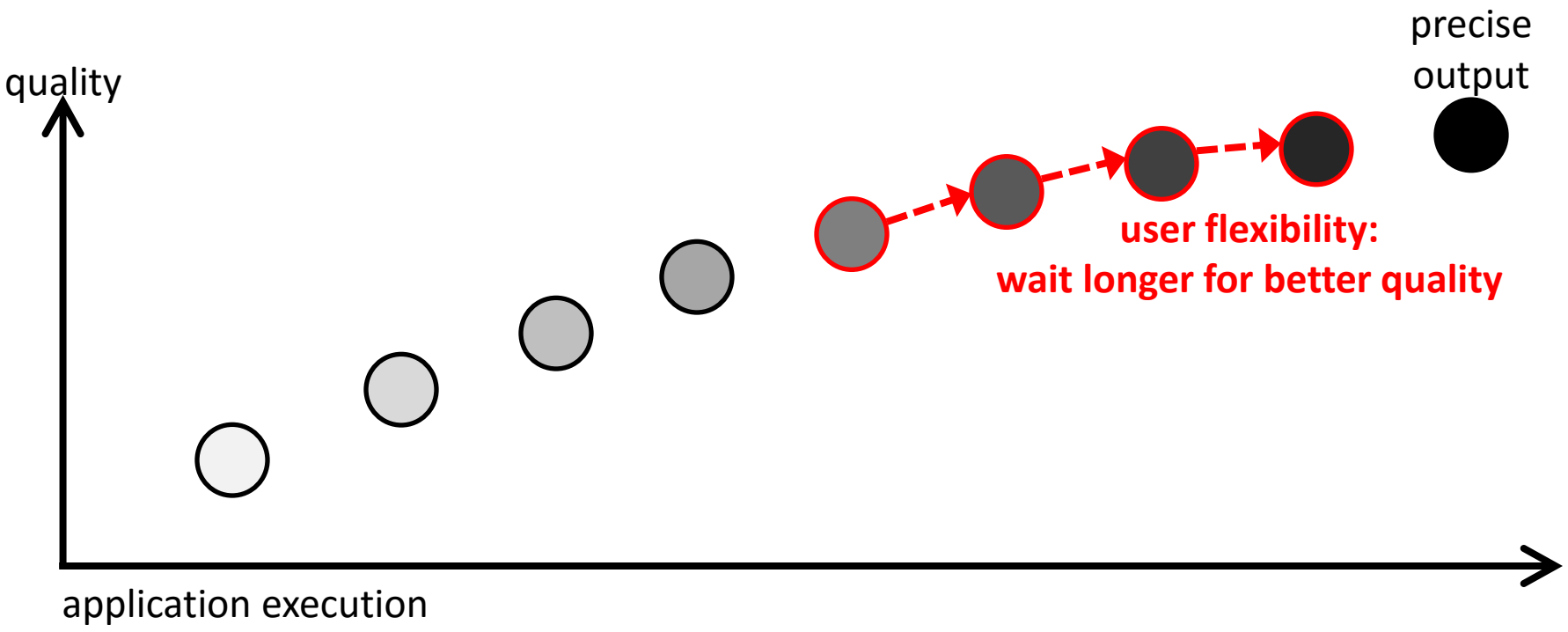
- A new computation model for approximate computing.



Conclusion

We propose the **Anytime Automaton**:

- A new computation model for approximate computing.



Thank you

The Anytime Automaton

Joshua San Miguel
Natalie Enright Jerger

The Edward S. Rogers Sr. Department
of Electrical & Computer Engineering
UNIVERSITY OF TORONTO

Special thanks to IBM collaborators:
Viji Srinivasan, Ravi Nair, Dan Prener