

# Certifying Positive Almost Sure Termination of Probabilistic Pushdown Automata

Tobias Winkler, Joost-Pieter Katoen



Highlights 2024

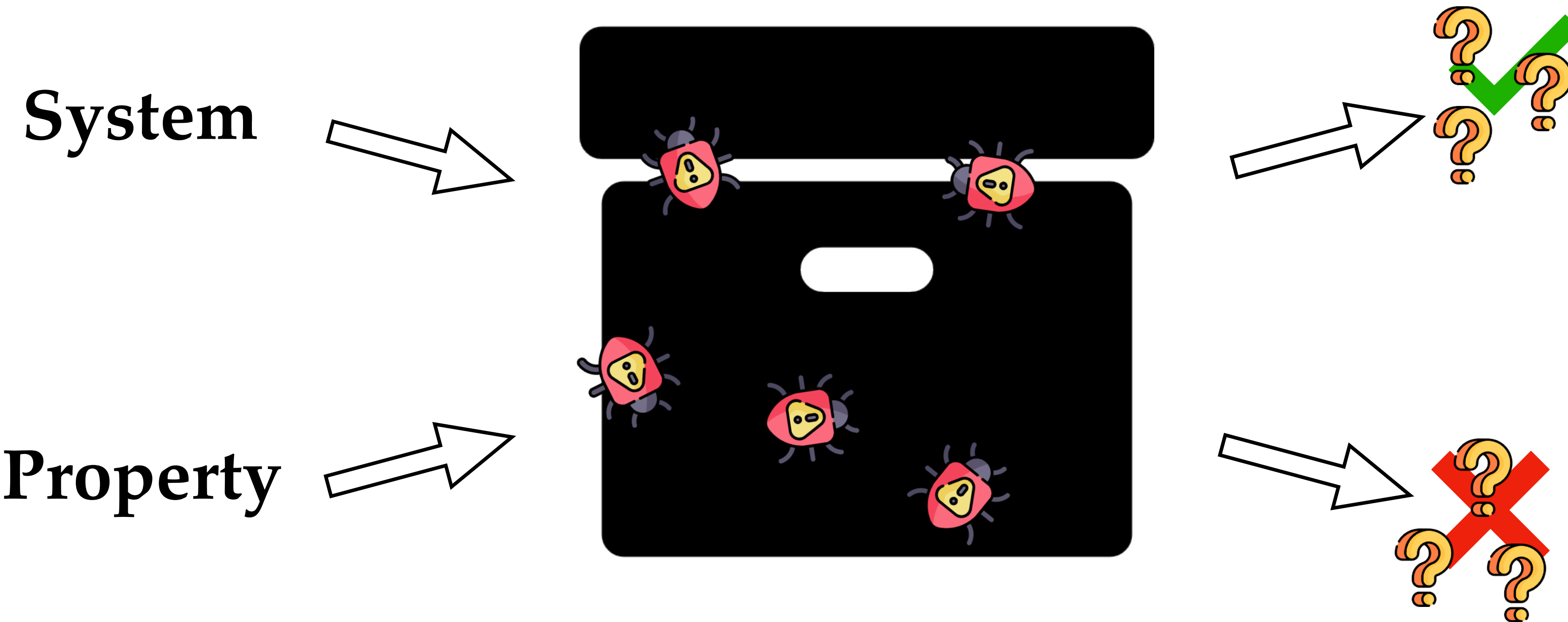
# Certificates in Verification



# Certificates in Verification



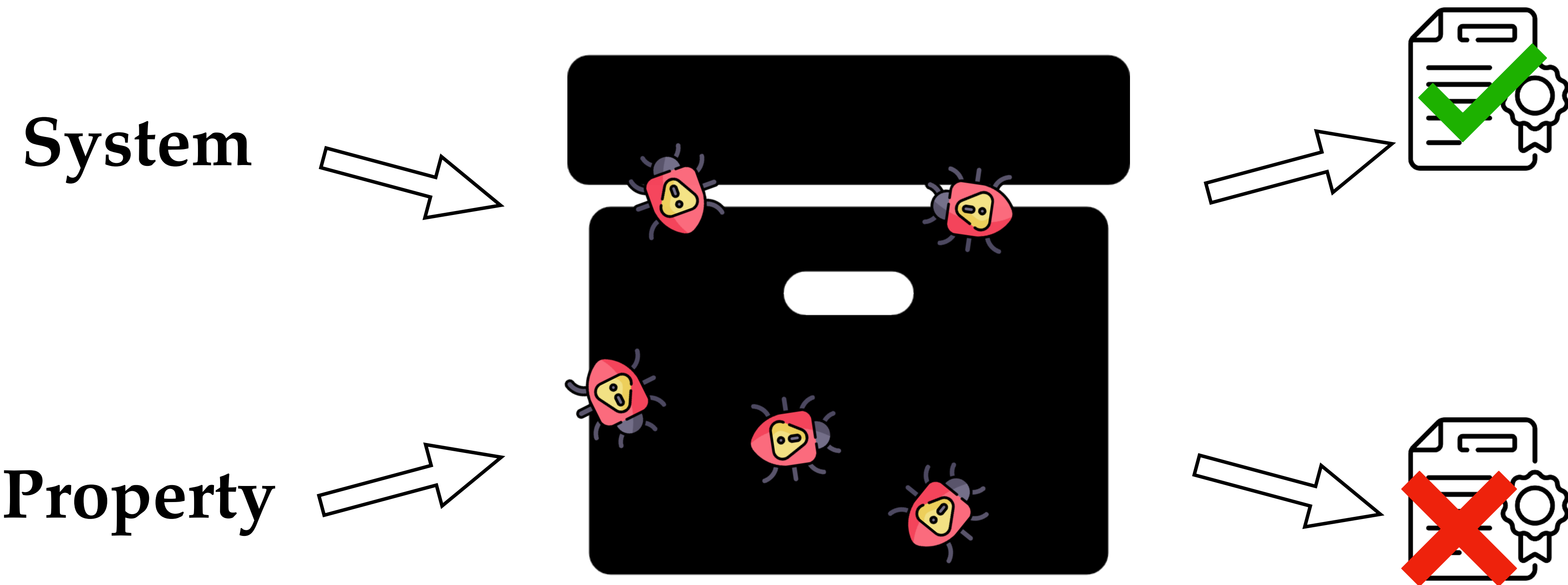
# Certificates in Verification



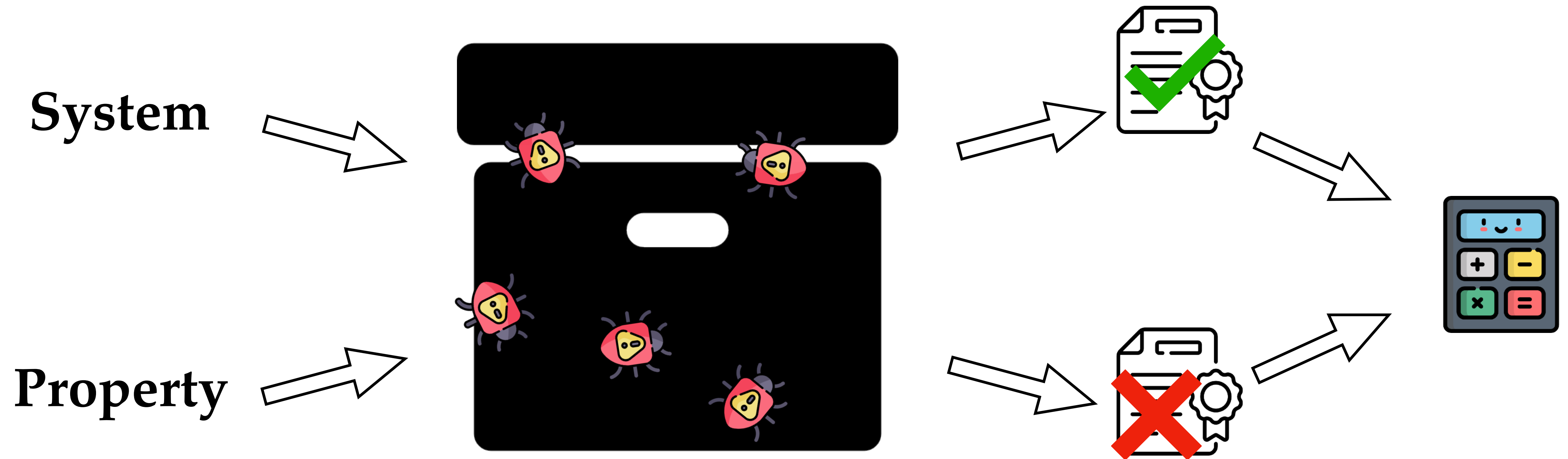
# Certificates in Verification



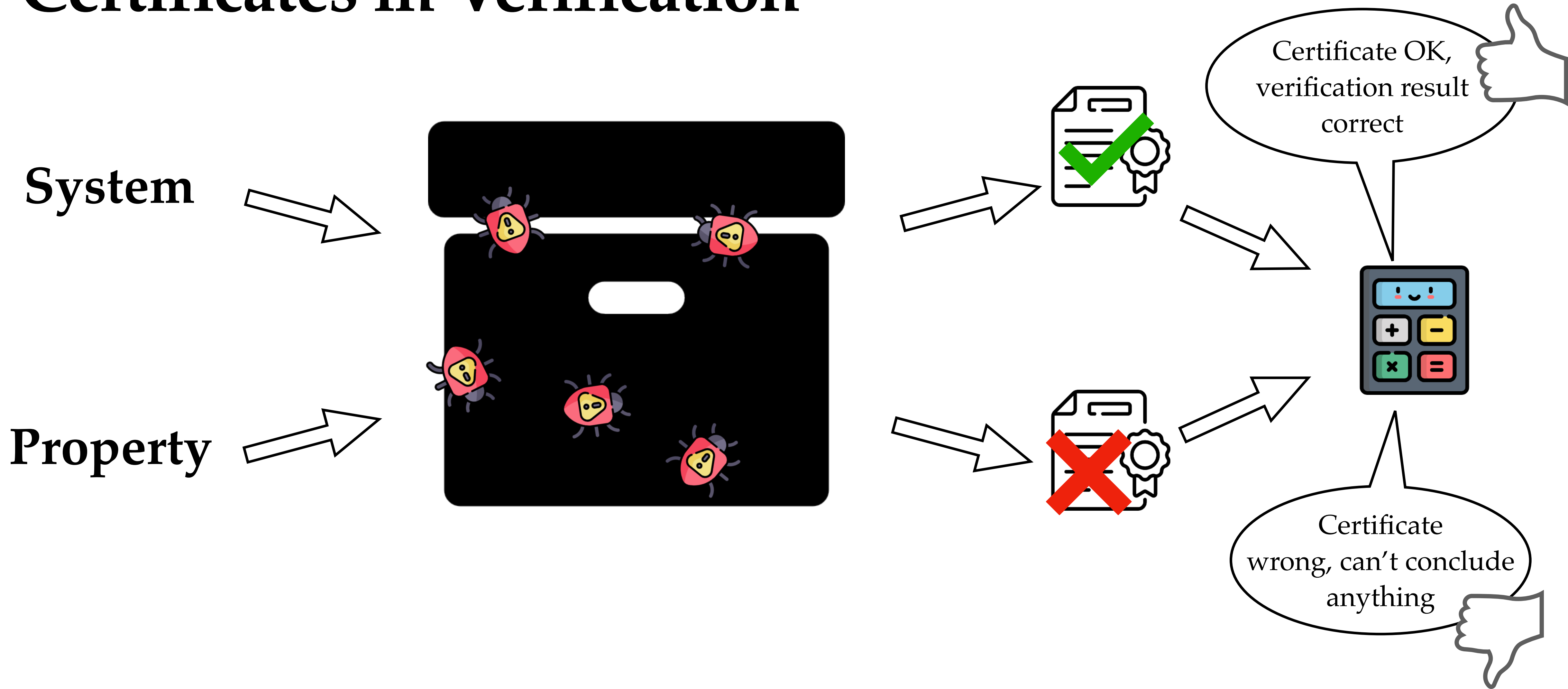
# Certificates in Verification



# Certificates in Verification

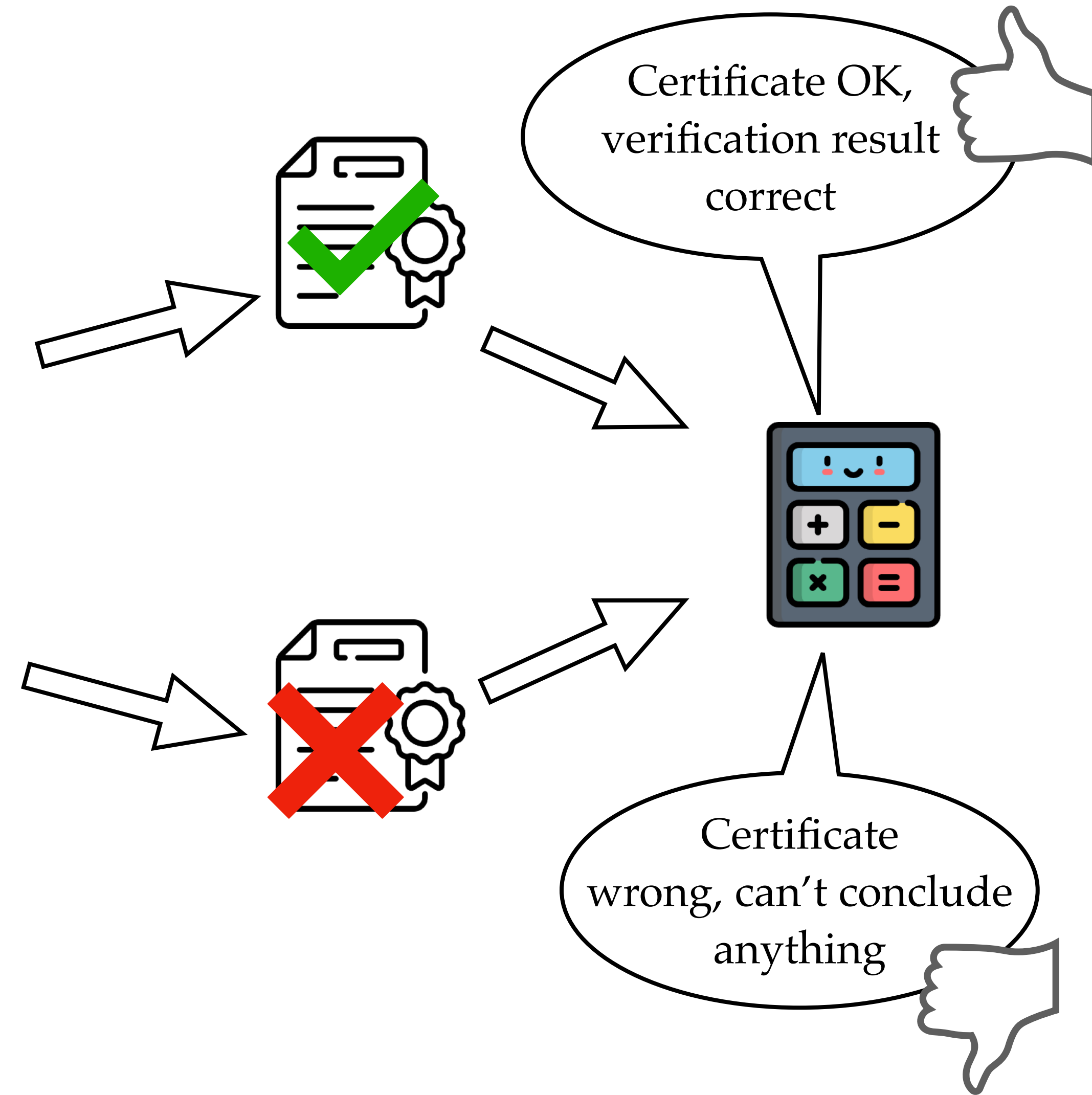
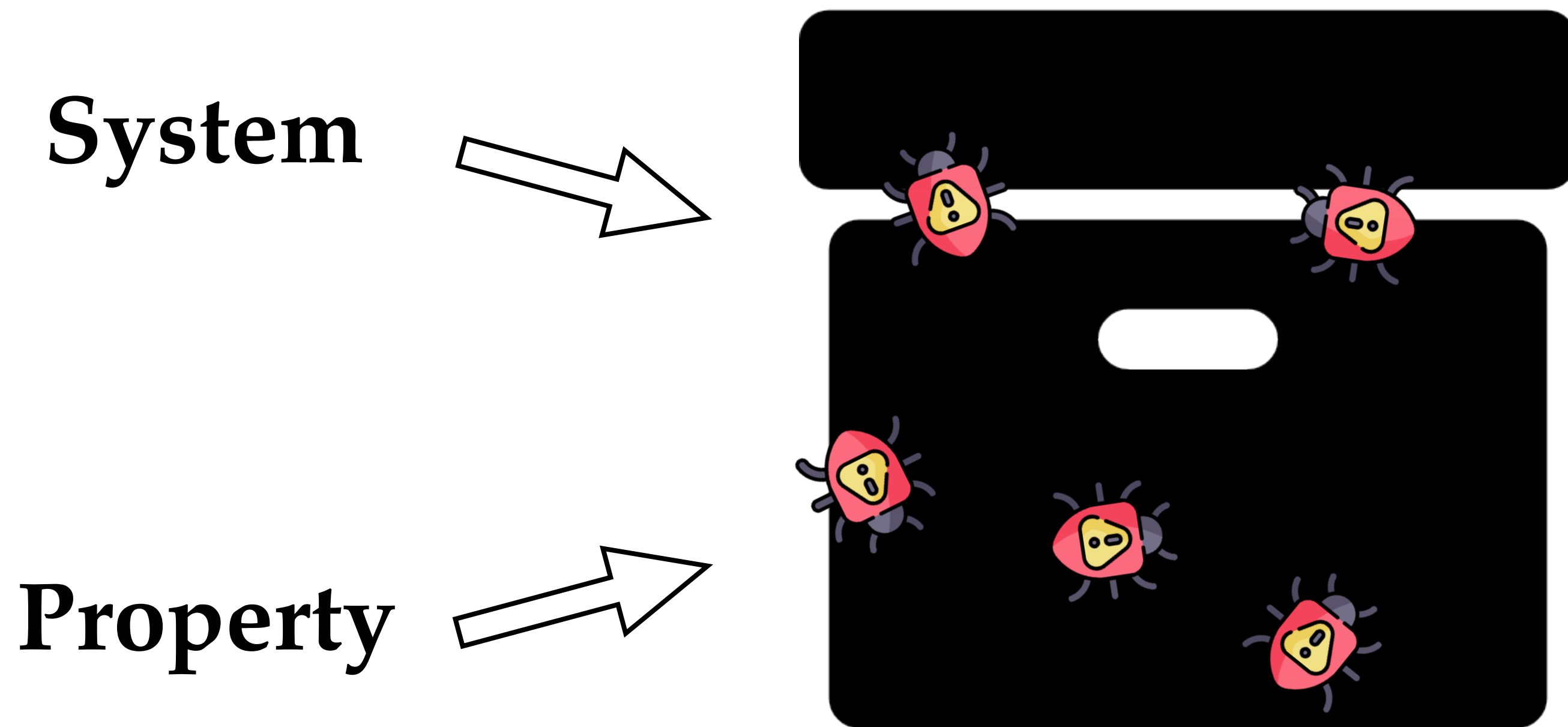


# Certificates in Verification



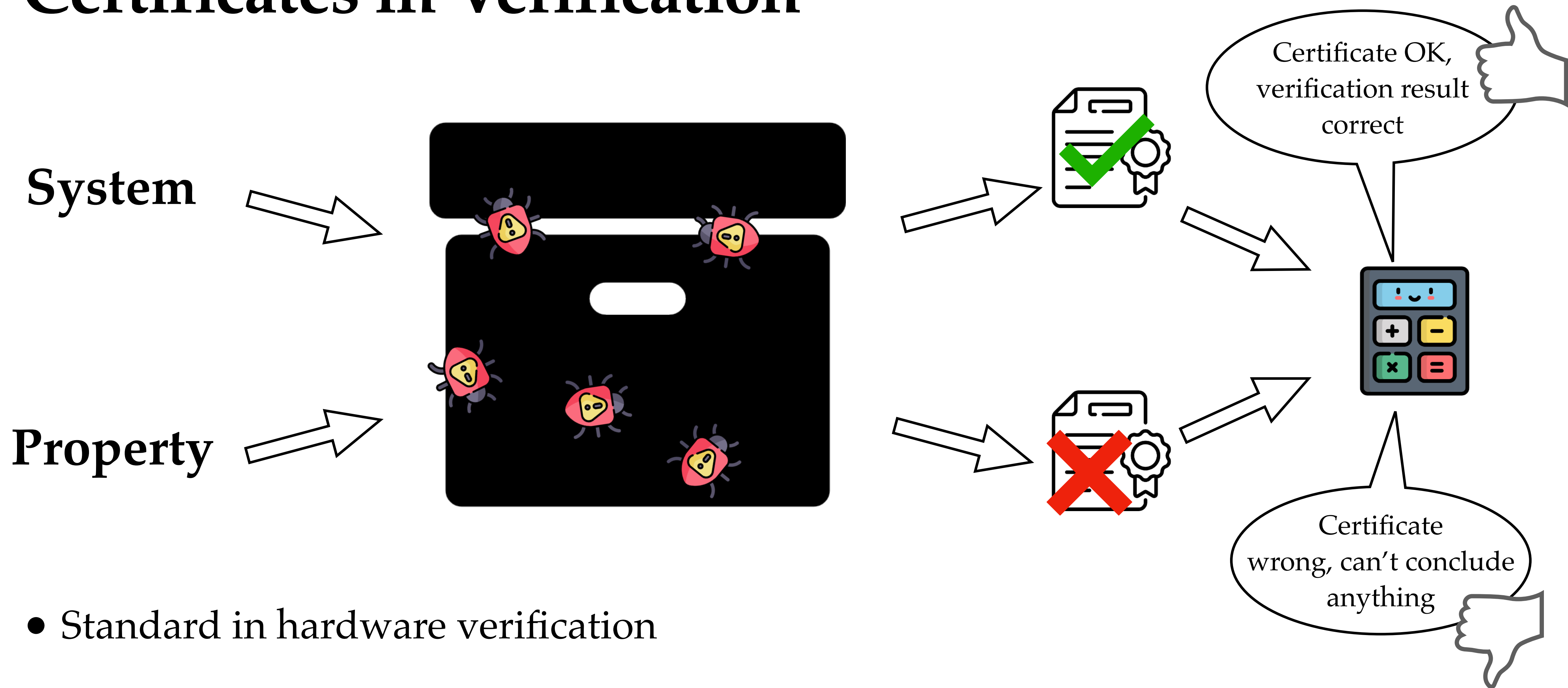


# Certificates in Verification



- Standard in hardware verification

# Certificates in Verification



- Standard in hardware verification
- Our goal: Certified verification of **probabilistic** systems

# Probabilistic Pushdown Automata

[Esparza, Kucera, Mayr, LICS 2004 + 2005]

# Probabilistic Pushdown Automata

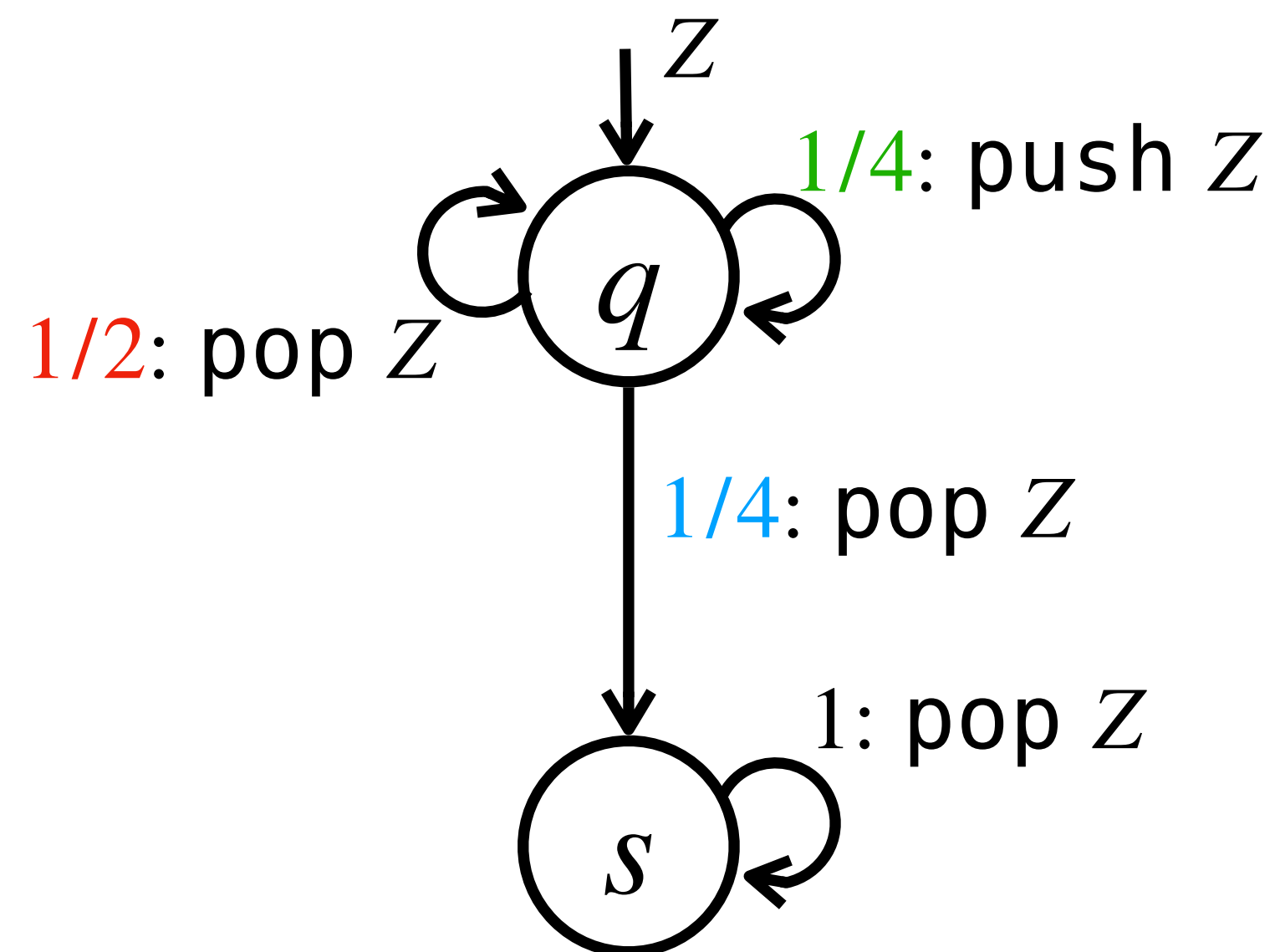
[Esparza, Kucera, Mayr, LICS 2004 + 2005]

Like standard PDA, but probabilities instead of input letters:

# Probabilistic Pushdown Automata

[Esparza, Kucera, Mayr, LICS 2004 + 2005]

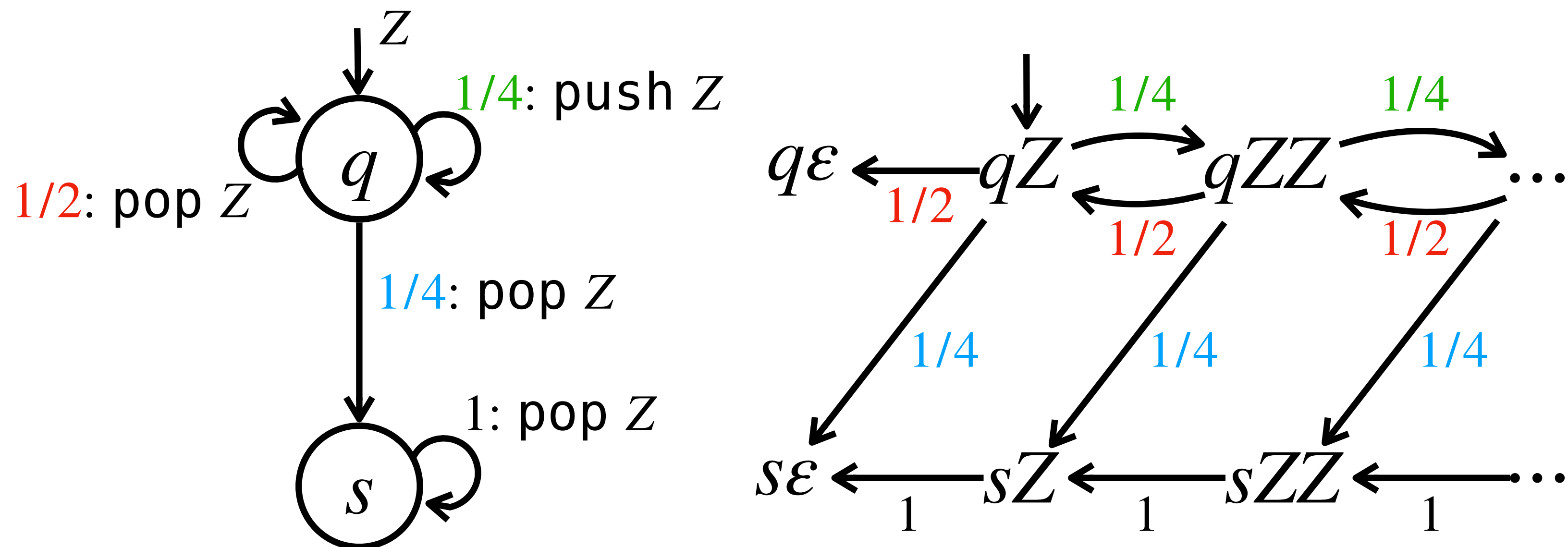
Like standard PDA, but probabilities instead of input letters:



# Probabilistic Pushdown Automata

[Esparza, Kucera, Mayr, LICS 2004 + 2005]

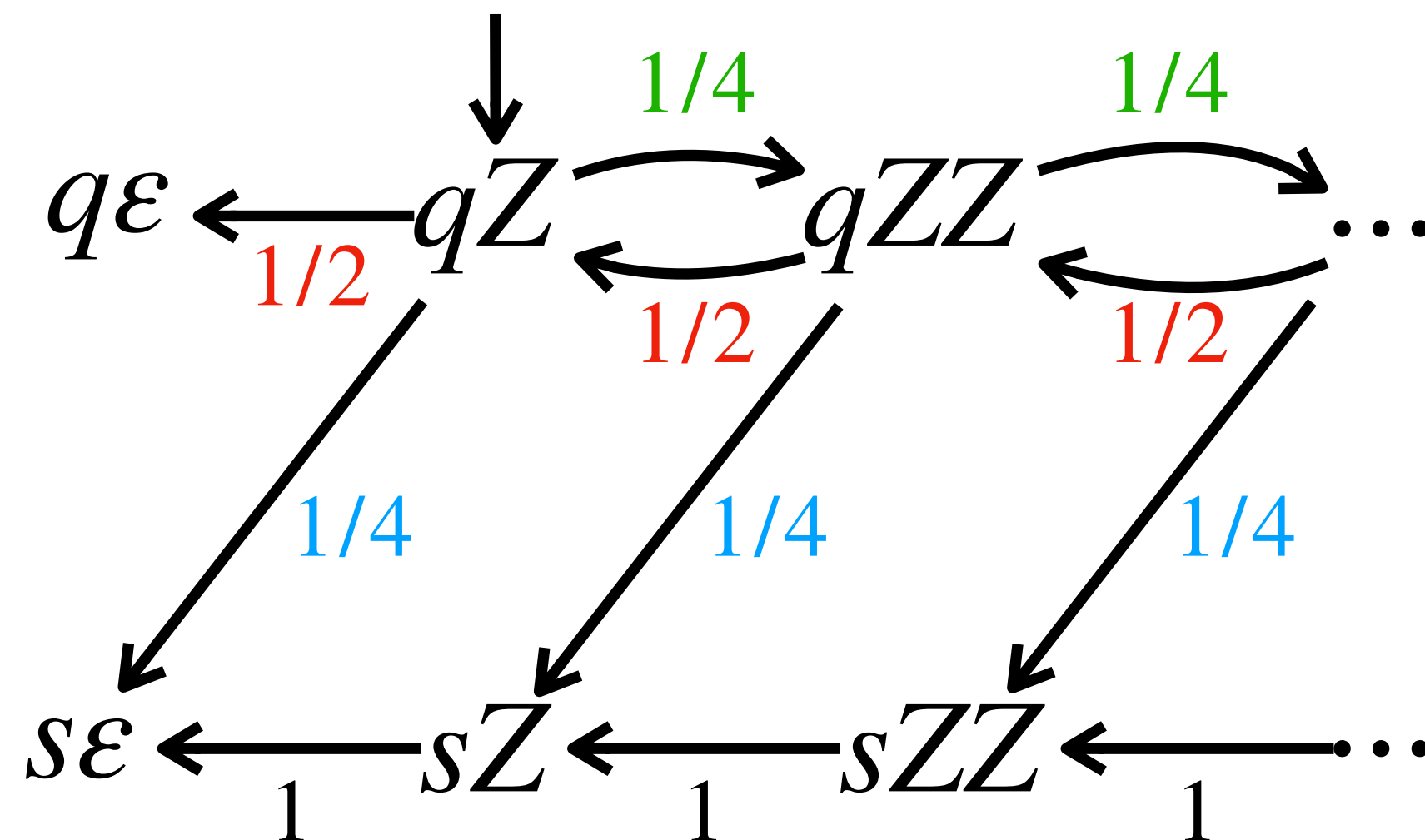
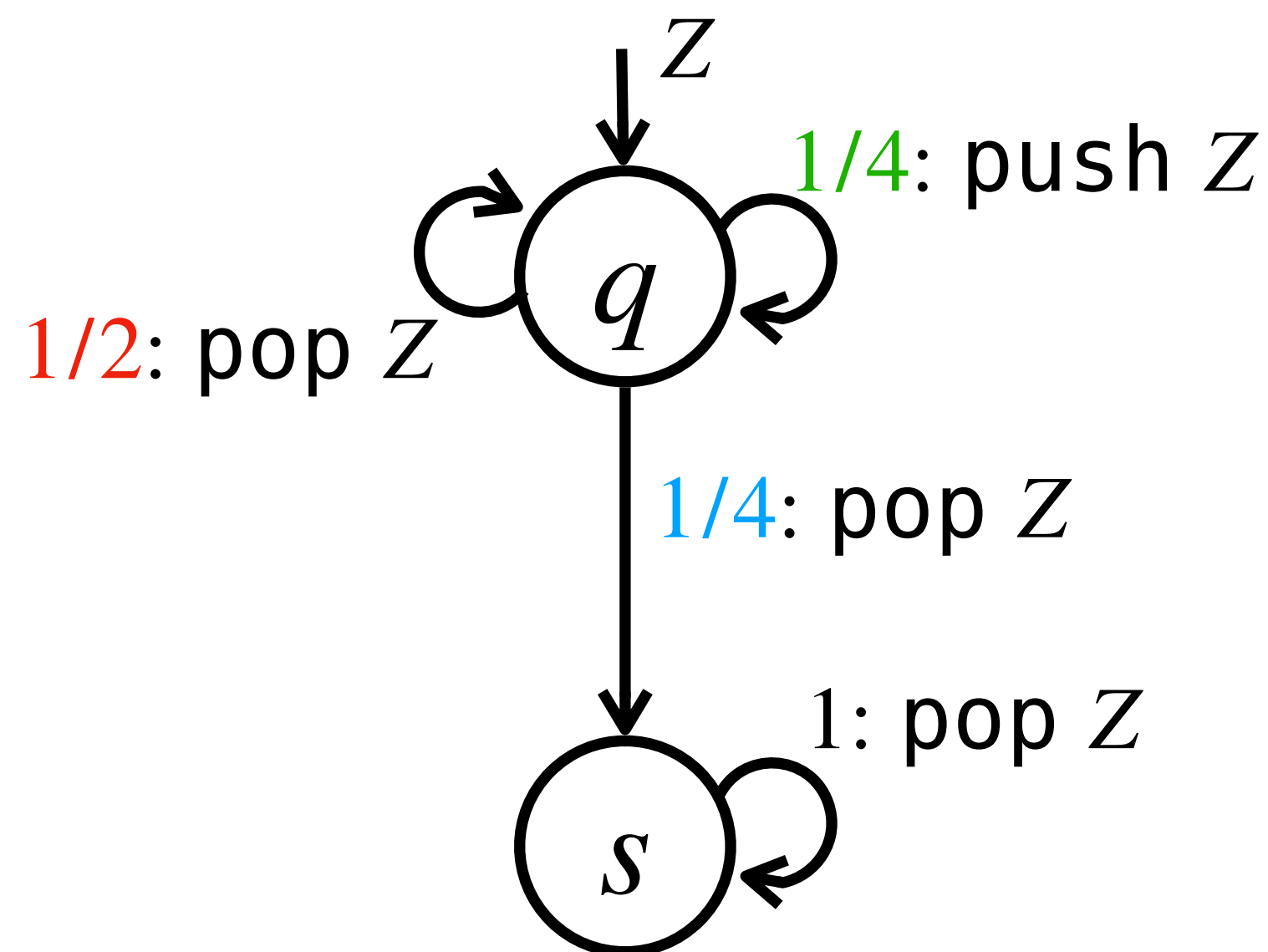
Like standard PDA, but probabilities instead of input letters:



# Probabilistic Pushdown Automata

[Esparza, Kucera, Mayr, LICS 2004 + 2005]

Like standard PDA, but probabilities instead of input letters:



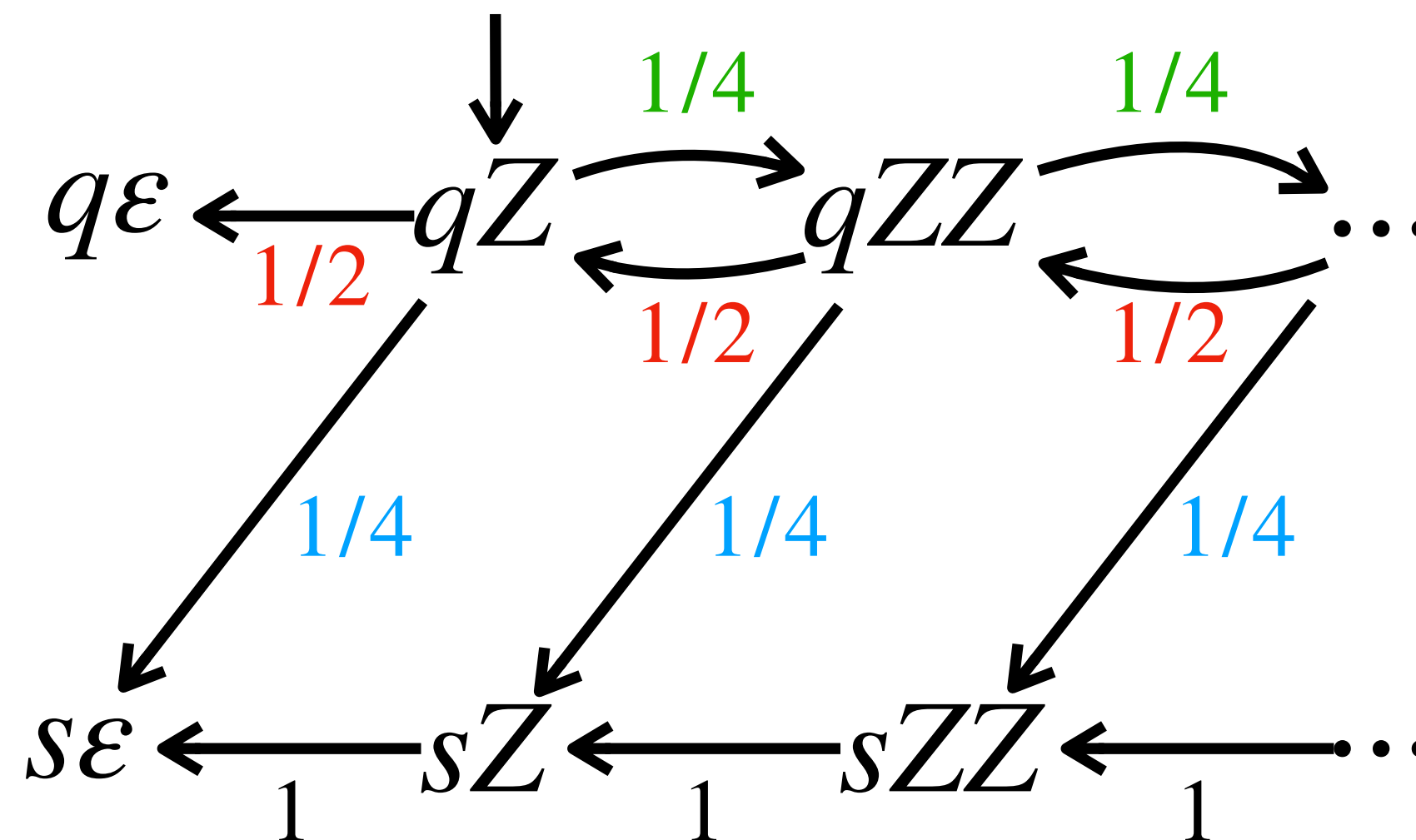
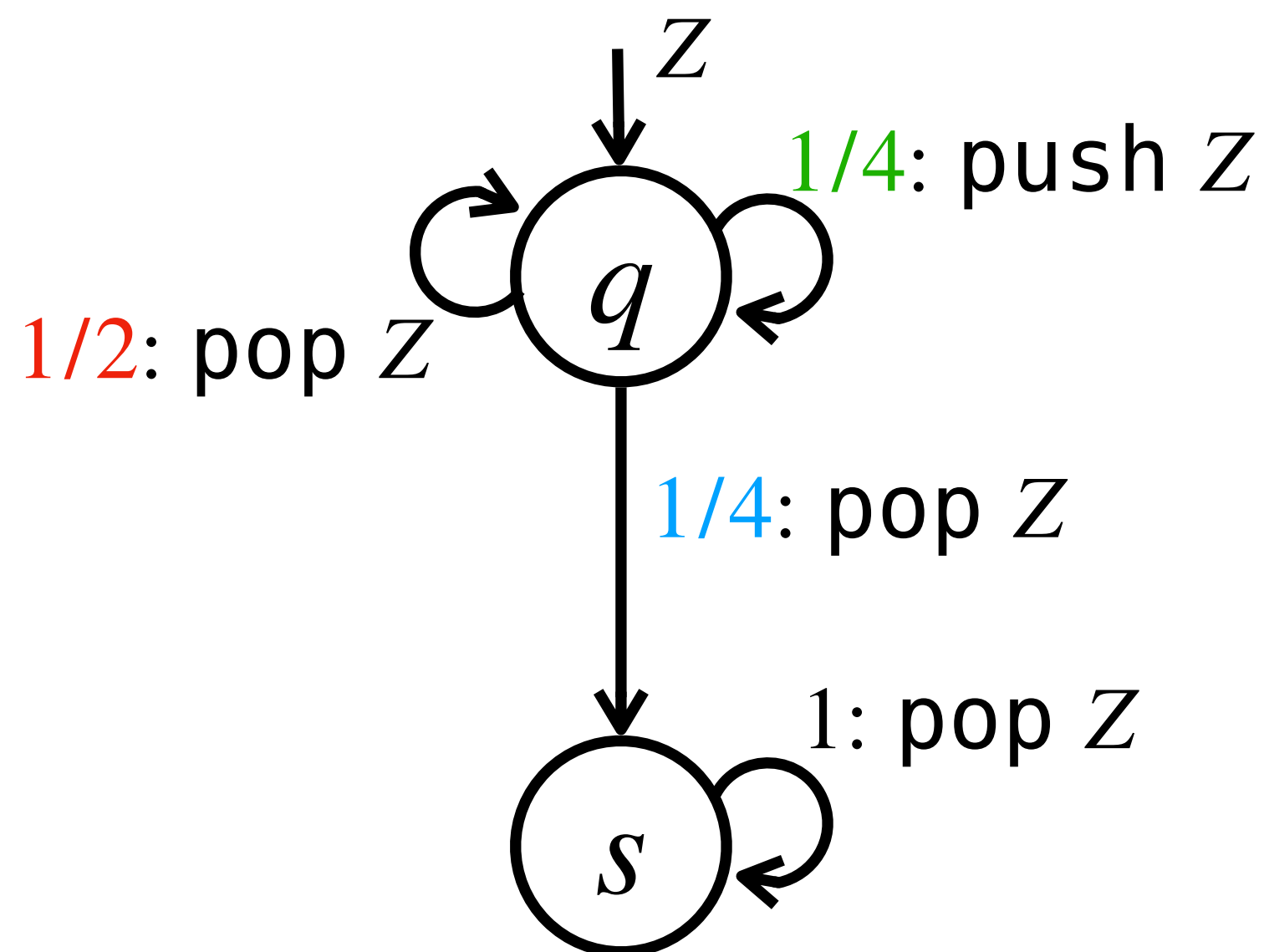
```
bool f(bool b) {
  if(b) {
    choice {
      0.25: return f(f(b));
      0.50: return true;
      0.25: return false;
    }
  } else return false;
}
```

# Probabilistic Pushdown Automata

[Esparza, Kucera, Mayr, LICS 2004 + 2005]

Like standard PDA, but probabilities instead of input letters:

$f(\text{true})$  returns **true** with probability  $2 - \sqrt{2}$



```

bool f(bool b) {
  if(b) {
    choice {
      0.25: return f(f(b));
      0.50: return true;
      0.25: return false;
    }
  } else return false;
}

```

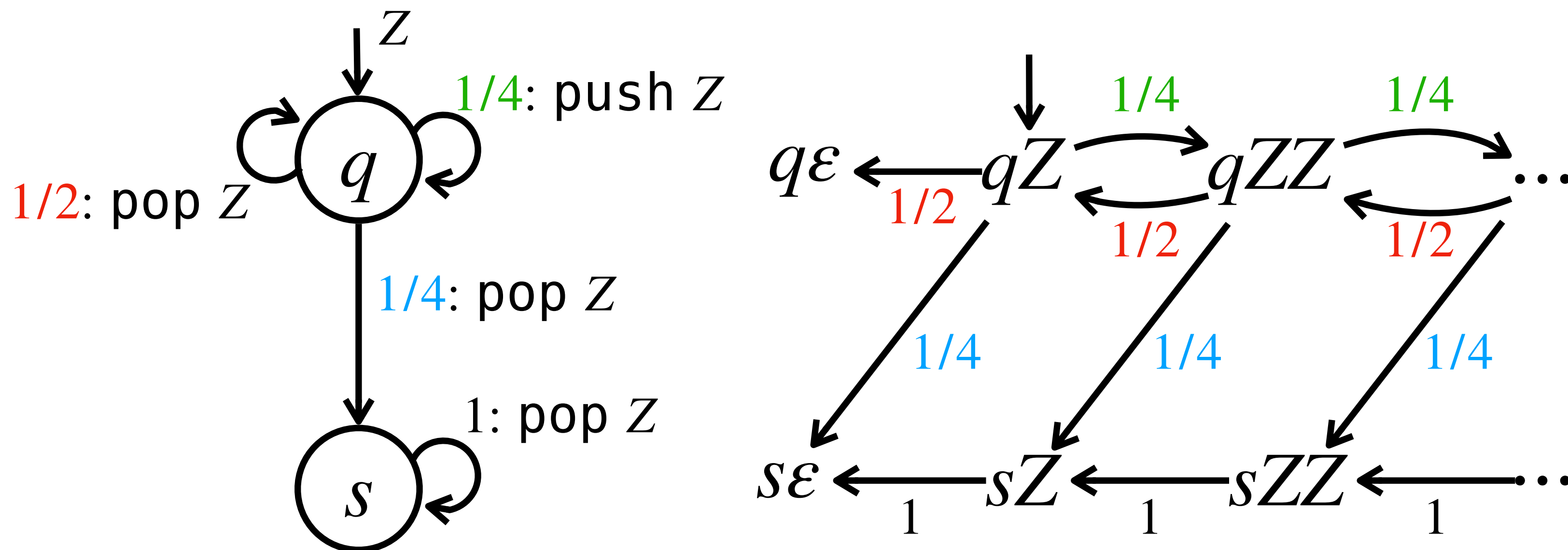


# Probabilistic Pushdown Automata

[Esparza, Kucera, Mayr, LICS 2004 + 2005]

Like standard PDA, but probabilities instead of input letters:

$f(\text{true})$  returns **true** with probability  $2 - \sqrt{2}$



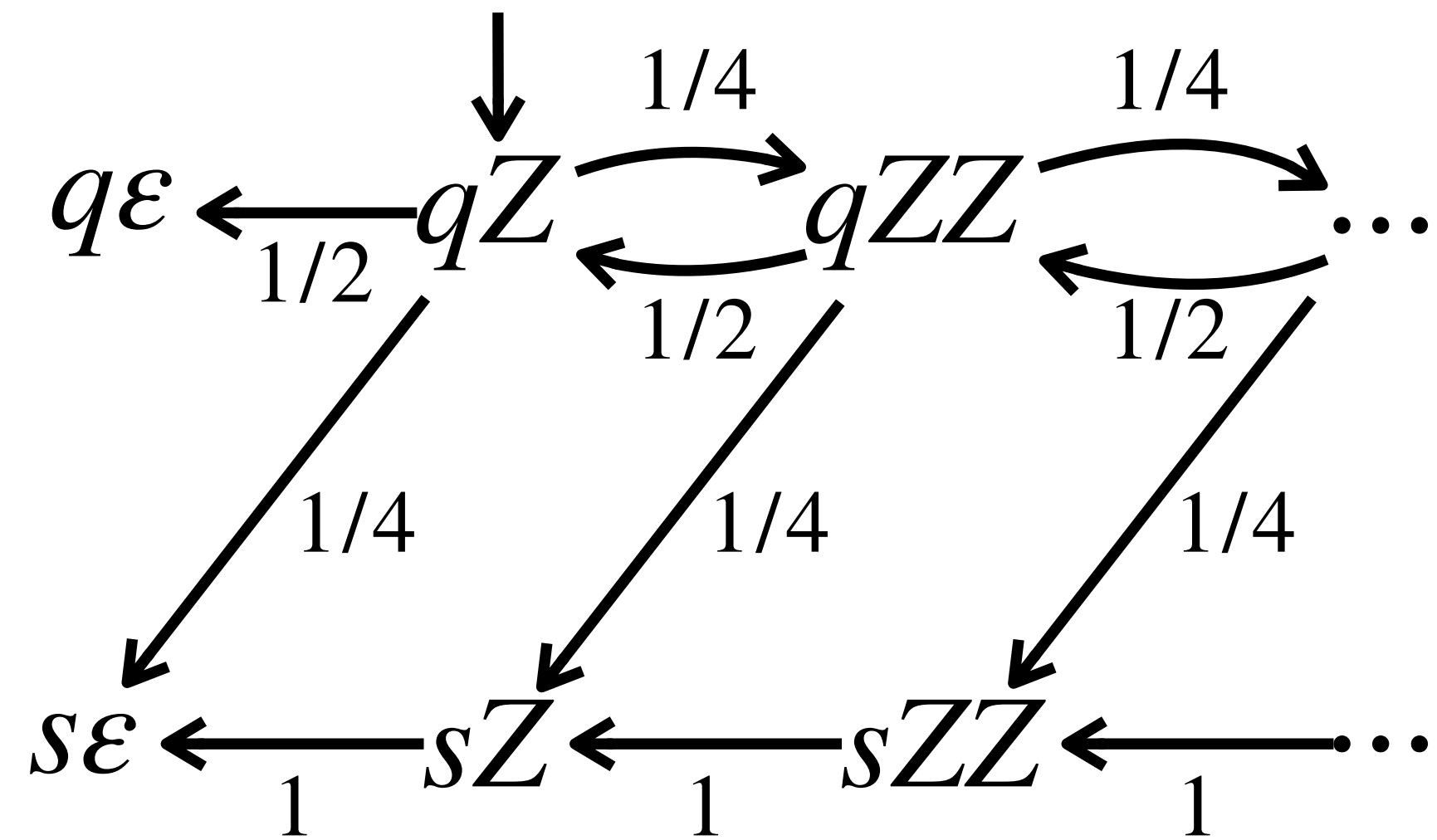
```
bool f(bool b) {
  if(b) {
    choice {
      0.25: return f(f(b));
      0.50: return true;
      0.25: return false;
    }
  } else return false;
}
```

WANTED



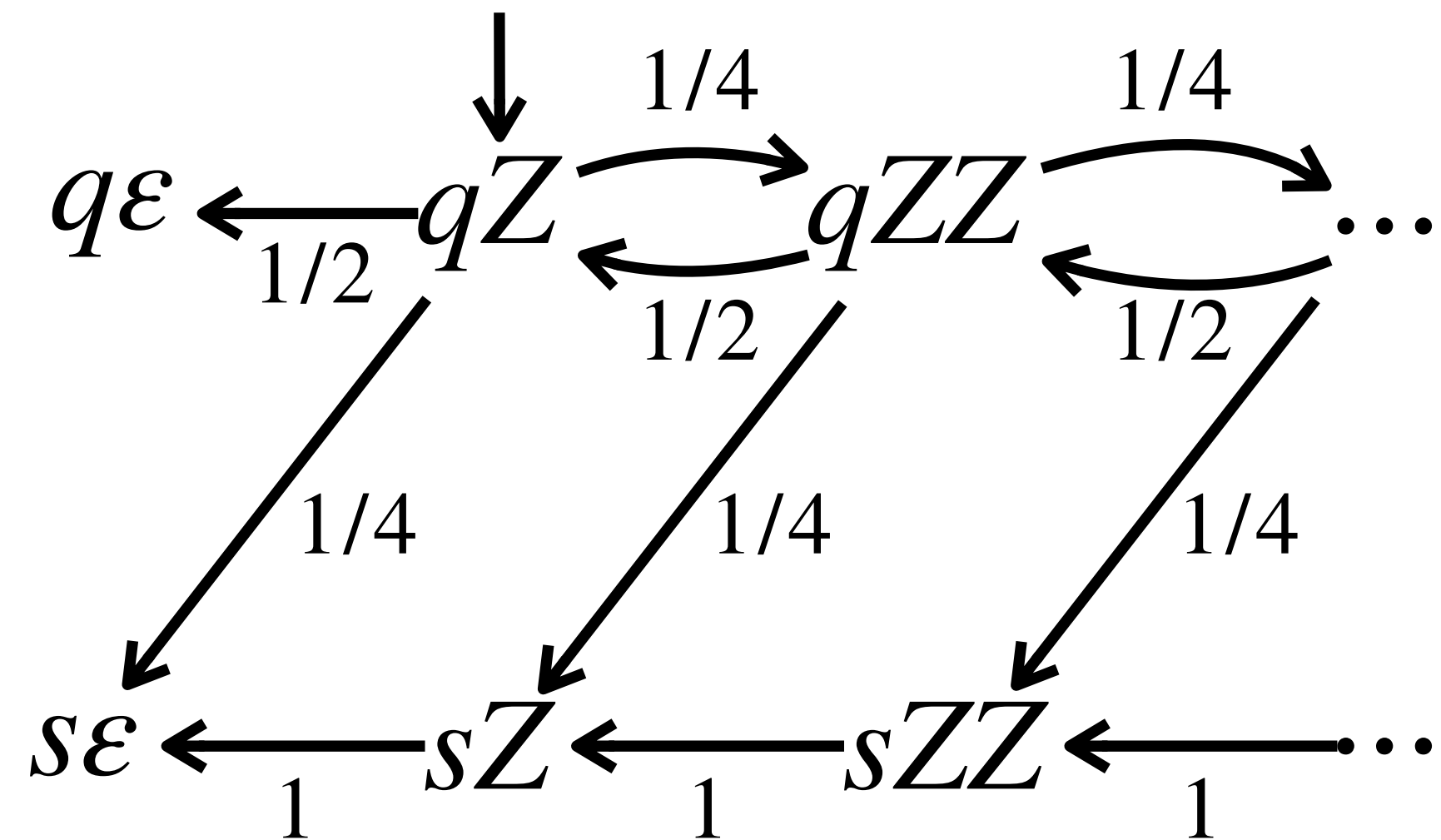
Certificates for **positive almost sure termination (PAST)**  
 = empty stack is reached after *finitely* many steps *on average*

# Certifying Empty Stack Reachability



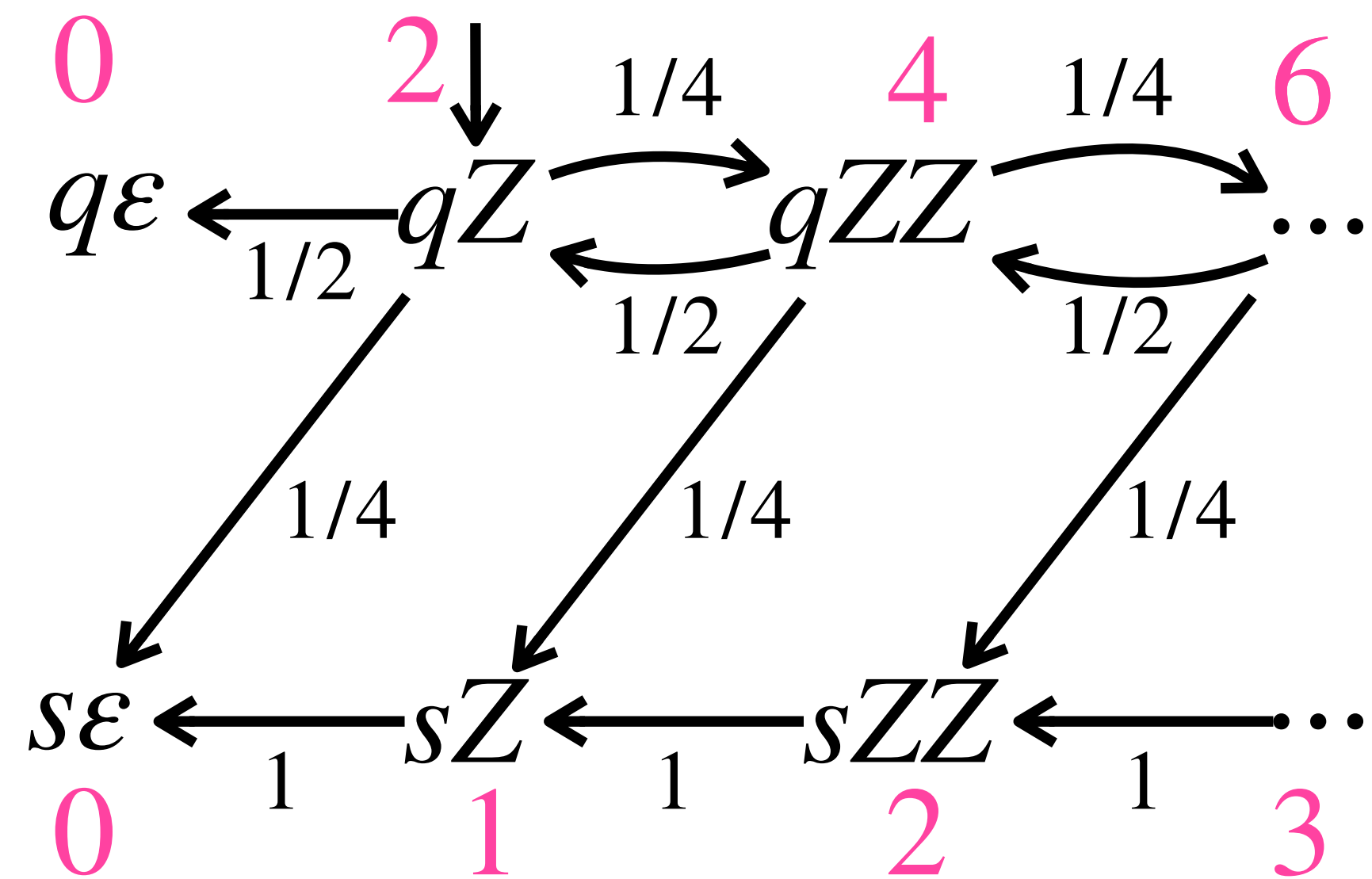
# Certifying Empty Stack Reachability

Assign a **rank** to each configuration:



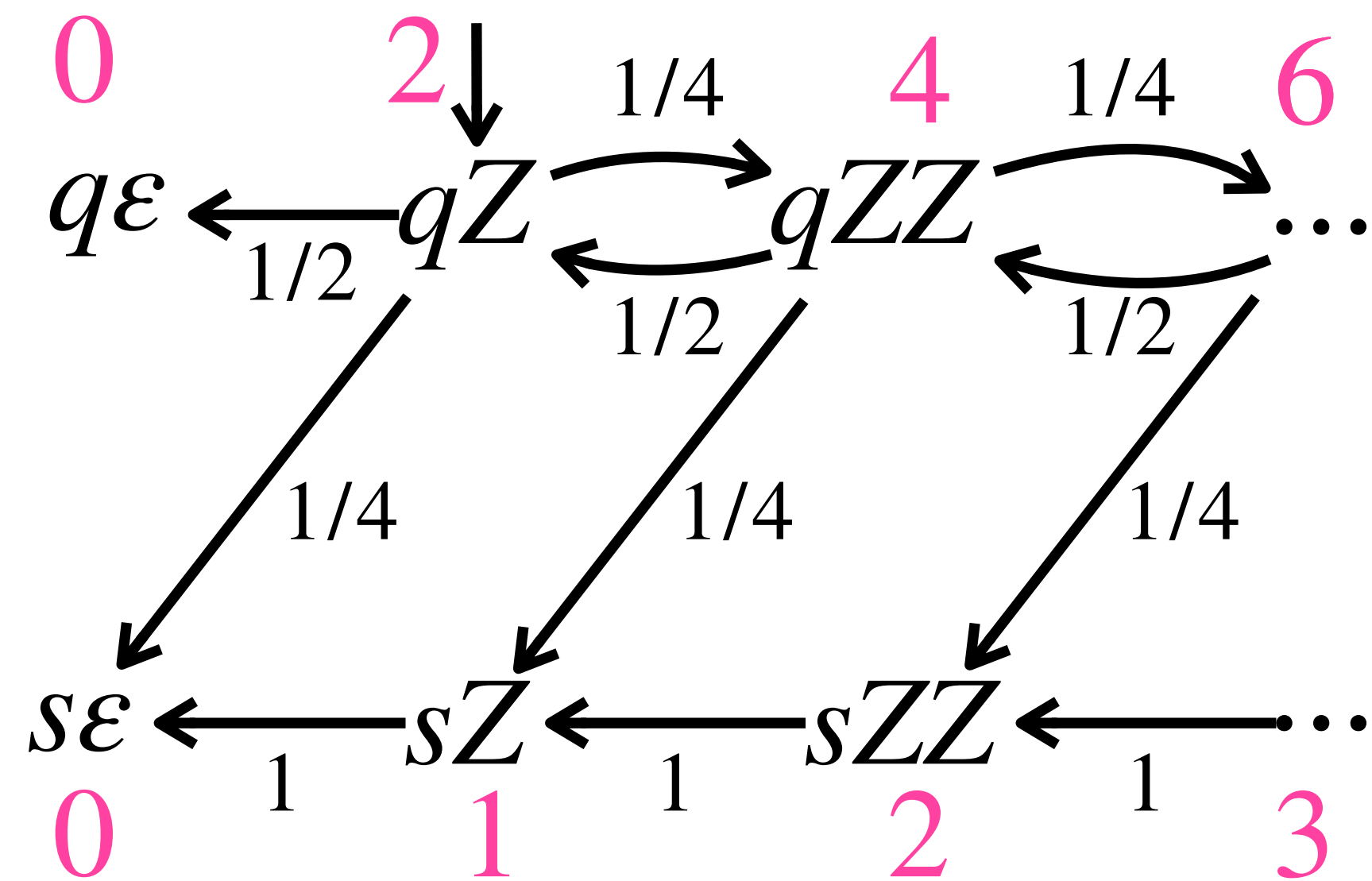
# Certifying Empty Stack Reachability

Assign a **rank** to each configuration:



# Certifying Empty Stack Reachability

Assign a **rank** to each configuration:

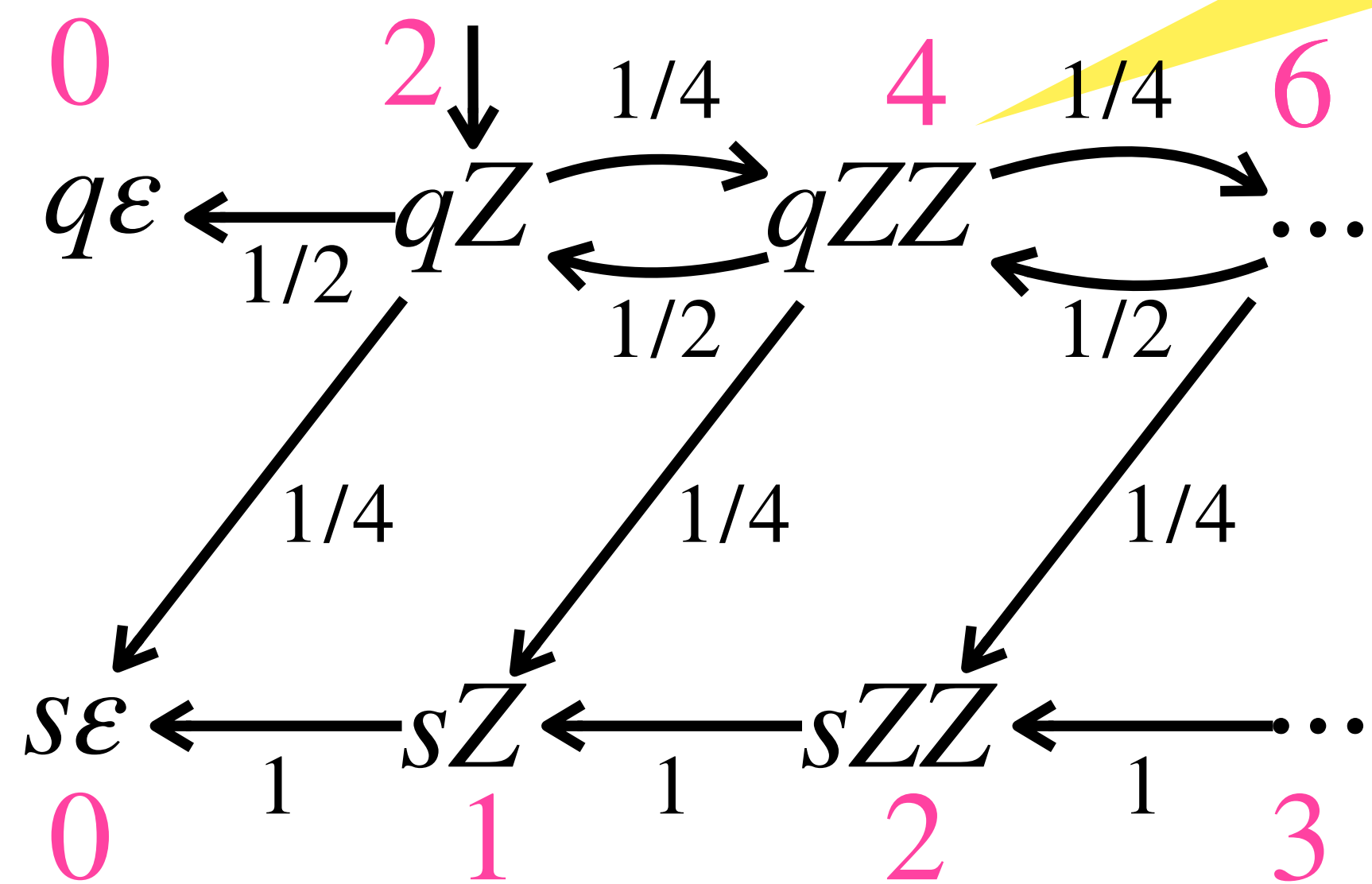


**Ranks** decrease by  $\geq 1$  on average in one step  $\implies$  PAST

# Certifying Empty Stack Reachability

Assign a **rank** to each configuration:

$$\frac{1}{2} \cdot 2 + \frac{1}{4} \cdot 6 + \frac{1}{4} \cdot 1 = 2.75 < 4$$



**Ranks** decrease by  $\geq 1$  on average in one step  $\implies$  PAST

# Finite Representation of $\infty$ -many Ranks?

# Finite Representation of $\infty$ -many Ranks?

We only need

- **ranks** for each configuration  $qZ$  quadratically many
- the **probabilities**  $\mathbb{P}(qZ \xrightarrow{*} s\mathcal{E})$  cubically many

to infer ranks for **all** ( $\infty$ -many) configurations.



# Finite Representation of $\infty$ -many Ranks?

We only need

- **ranks** for each configuration  $qZ$  quadratically many
- the **probabilities**  $\mathbb{P}(qZ \xrightarrow{*} s\varepsilon)$  cubically many

to infer ranks for **all** ( $\infty$ -many) configurations.

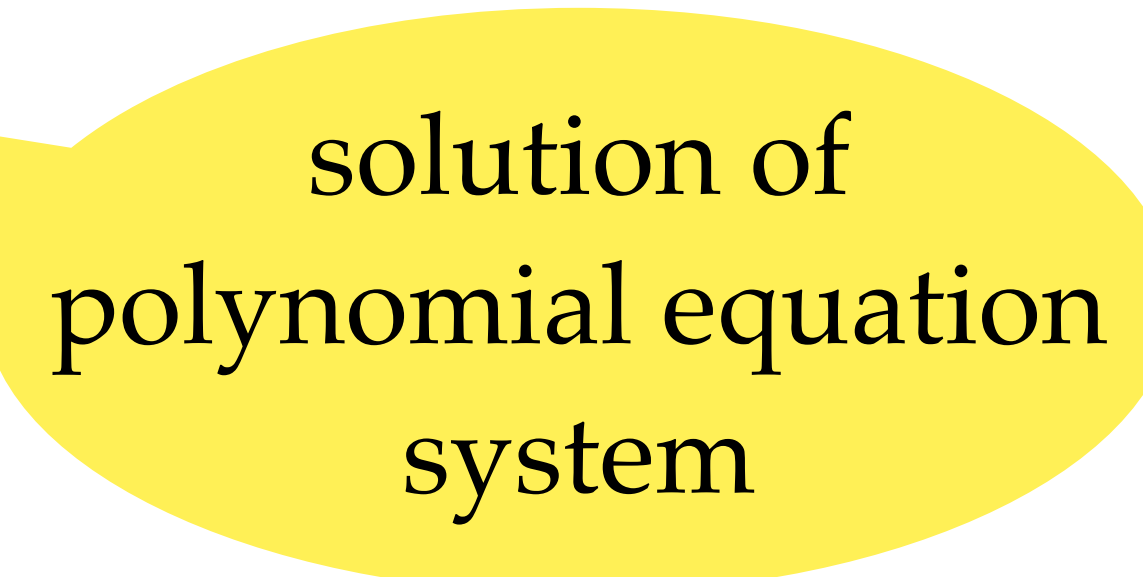
This is due to the specific **structure** of the configuration graph generated by a PDA.

# Finite Representation of $\infty$ -many Ranks?

We only need

- **ranks** for each configuration  $qZ$  quadratically many
- the **probabilities**  $\mathbb{P}(qZ \xrightarrow{*} s\varepsilon)$  cubically many

to infer ranks for **all** ( $\infty$ -many) configurations.



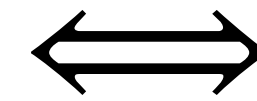
solution of  
polynomial equation  
system

This is due to the specific **structure** of the configuration graph generated by a PDA.

# Theorem: Certificates for PAST

[W & Katoen, LICS 2023]

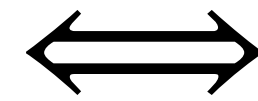
The probabilistic PDA  $\Delta$  is **PAST**



# Theorem: Certificates for PAST

[W & Katoen, LICS 2023]

The probabilistic PDA  $\Delta$  is **PAST**



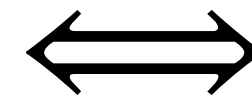
there exist non-negative ranks  $\vec{r}$  and probabilities  $\vec{p}$  satisfying a system of polynomial inequalities (that can be “read off” from  $\Delta$  in polynomial time).

# Theorem: Certificates for PAST

[W & Katoen, LICS 2023]

The probabilistic PDA  $\Delta$  is **PAST**

rational



there exist non-negative ranks  $\vec{r}$  and probabilities  $\vec{p}$  satisfying a system of polynomial inequalities (that can be “read off” from  $\Delta$  in polynomial time).

# Theorem: Certificates for PAST

[W & Katoen, LICS 2023]

The probabilistic PDA  $\Delta$  is PAST

↔  
rational

there exist non-negative ranks  $\vec{r}$  and probabilities  $\vec{p}$  satisfying a system of polynomial inequalities (that can be “read off” from  $\Delta$  in polynomial time).

**Certifying Positive Almost Sure Termination of Probabilistic Pushdown Automata**  
Tobias Winkler — based on a LICS '23 paper with Joost-Pieter Katoen

**Probabilistic Pushdown Automata (PPDA) [1]**

- Like standard PDA, but with probabilities on transitions.
- Natural model for procedural probabilistic programs with recursion, stochastic grammars, probabilistic counter systems, and others.
- Various notions of termination (all decidable in PSPACE):
  - traditional termination: all runs reach empty stack
  - almost sure termination (AST): reach empty stack with probability 1
  - positive AST (PAST): reach empty stack in finite expected time

**Certificates**

- “Simple” proof of a property, trivially (machine-)checkable
- Relevant in formal methods as software tools may have bugs
- Certifying algorithms as an alternative to formally verified model checkers

**Example: Certifying PAST of a Recursive Probabilistic Program**

Program: Bernoulli( $\sqrt{2}-1$ )-sampler

```
bool f(bool b) {
  if (b) {
    pchoice {
      1/4: return f(f(true));
      1/4: return false;
      1/2: return true;
    }
  } else return false;
}
bool main() { return f(true); }
```

PPDA model, annotated with certificate:

Interpretation of the annotations:

- $\odot$  means  $\mathbb{E}[\#Steps(qZ \rightsquigarrow ?o)] \leq r$
- $\oplus$  means  $\mathbb{P}[qZ \rightsquigarrow re] \leq p$

Verifying the certificate: This proves PAST!

$$2 \leq 1 + \frac{1}{2} \cdot \left(2 + \frac{3}{2} - 2 + \frac{1}{2}\right) \quad \frac{3}{2} \leq \frac{1}{2} \cdot \left(\frac{3}{2}\right)^2 - \frac{1}{2} \quad \frac{1}{2} \leq \frac{1}{2} \cdot \left(\frac{3}{2} + \frac{1}{2}\right) + \frac{1}{2}$$

**Theoretical Results [3]**

- Our certificates are sound and complete w.r.t.  $\mathbb{Q}$ . This means that the numbers  $r$  and  $p$  can always be chosen in the rationals.
- Proof idea: PAST implies that the Jacobi matrix of the polynomial system characterizing the probabilities  $p$  as its least fixed point (lfp) is invertible around the lfp—intuitively, it follows that there is “wiggle room” in which we can fit a sufficiently precise rational upper bound.
- In general, exponentially many bits of precision are necessary.

**Outlook**

- Searching for certificates [2] leads to novel, practically feasible algorithms to prove PAST.
- Certificates for AST?
- Exact complexity of (P)AST?

**References**

- [1] J. Espartero et al. “Model Checking Probabilistic Pushdown Automata”. In: LICS 2004.
- [2] T. Winkler and J.-P. Katoen. “Certificates for PPDA via Optimistic Value Iteration”. In: TACAS (2), 2023.
- [3] T. Winkler and J.-P. Katoen. “On Certificates, Expected Runtimes, and Termination in PPDA”. In: LICS 2023.

erc RWTH AACHEN UNIVERSITY 2 RWTH AACHEN UNIVERSITY

Check poster for concrete example



Link to paper