

# Reachability of subclasses of (branching) vector addition systems

**Stefan Göller**  
**LSV, CNRS & ENS Cachan, France**

based joint works with

Michael Blondin (Montréal)

Alain Finkel (Cachan)

Christoph Haase (Cachan)

Pierre McKenzie (Montréal)

Ranko Lazic (Warwick)

Patrick Totzke (Warwick)

**PART 1:** Vector addition systems

**PART 2:** Branching vector addition systems

**PART 1: Vector addition systems**

**PART 2: Branching vector addition systems**

# Vector addition systems

Some background

Vector addition systems / Presburger reachability

2VASS reachability: Our main result

# Overview

## **Some background**

Vector addition systems / Presburger reachability

2VASS reachability: Our main result















# Mathematical modeling

```
decl s := 0;  
0: goto 1;  
1: s := 1;  
2: s := 0;  
3: assert (!s);
```

# Mathematical modeling

```
decl s := 0;  
0: goto 1;  
1: s := 1;  
2: s := 0;  
3: assert (!s);
```

Many threads execute  
this Boolean program



# Mathematical modeling

```
decl s := 0;  
0: goto 1;  
1: s := 1;  
2: s := 0;  
3: assert (!s);
```

Many threads execute  
this Boolean program



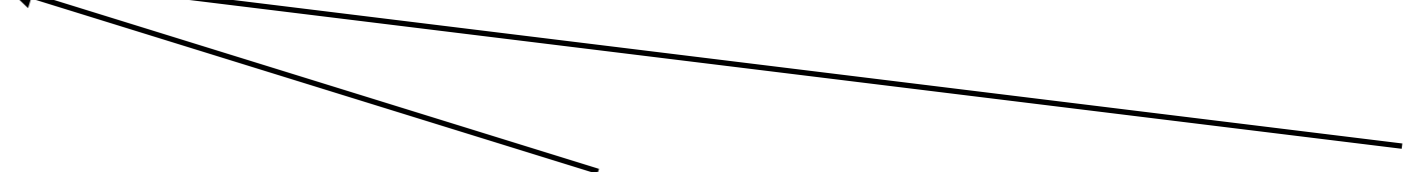
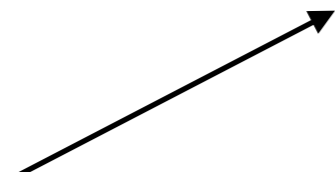
$(1, 0, 0, 0)$

#threads with PC=0

#threads with PC=1

#threads with PC=2

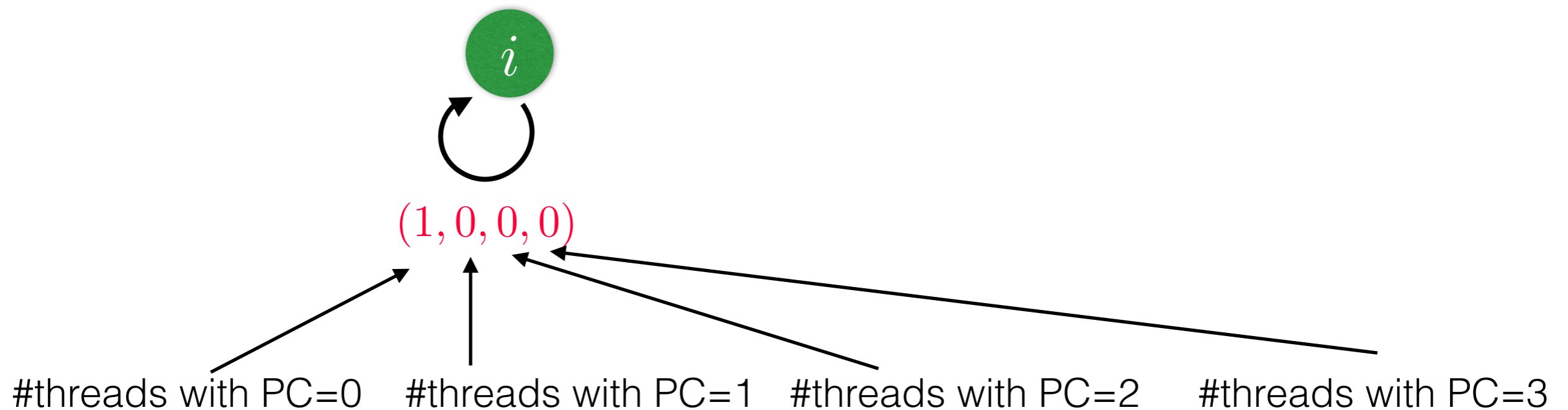
#threads with PC=3



# Mathematical modeling

```
decl s := 0;  
0: goto 1;  
1: s := 1;  
2: s := 0;  
3: assert (!s);
```

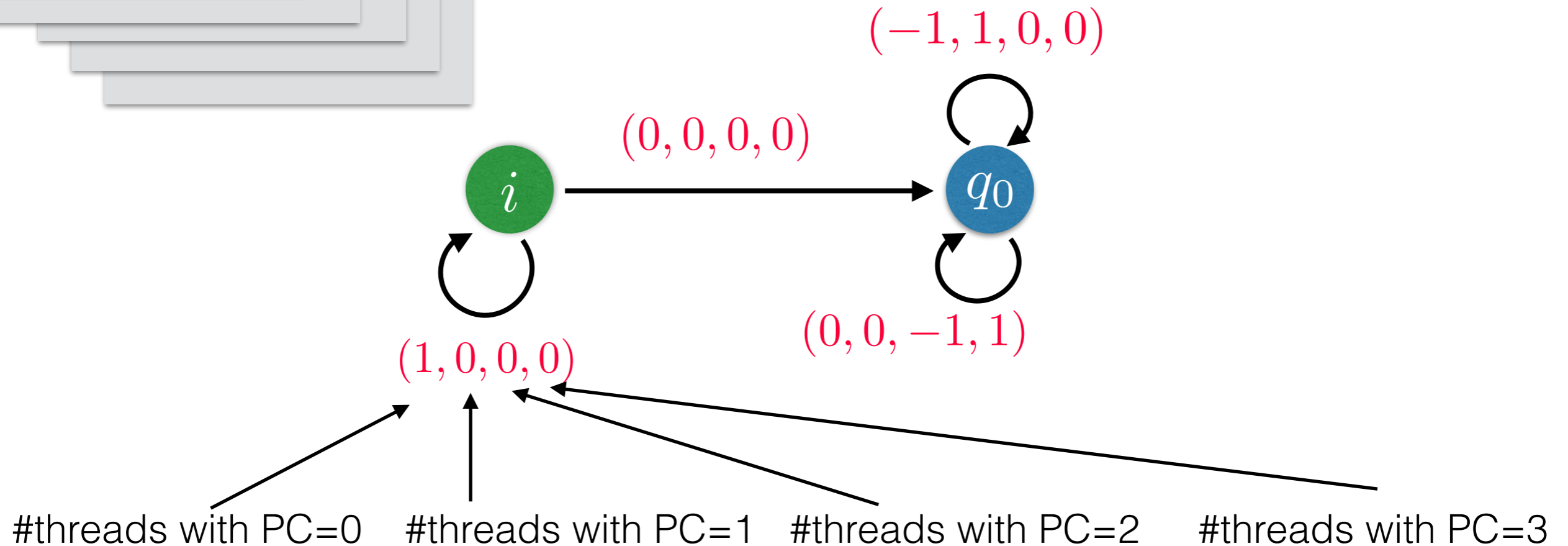
Many threads execute  
this Boolean program



# Mathematical modeling

```
decl s := 0;  
0: goto 1;  
1: s := 1;  
2: s := 0;  
3: assert (!s);
```

Many threads execute this Boolean program

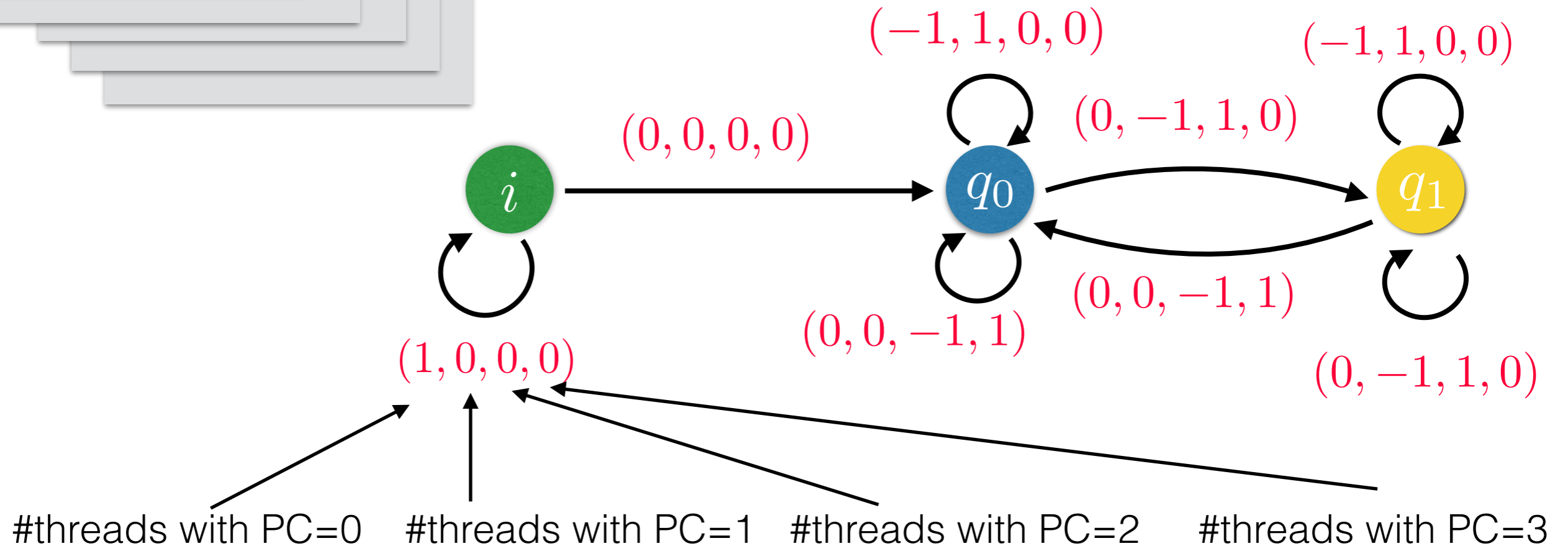




# Mathematical modeling

```
decl s := 0;  
0: goto 1;  
1: s := 1;  
2: s := 0;  
3: assert (!s);
```

Many threads execute this Boolean program

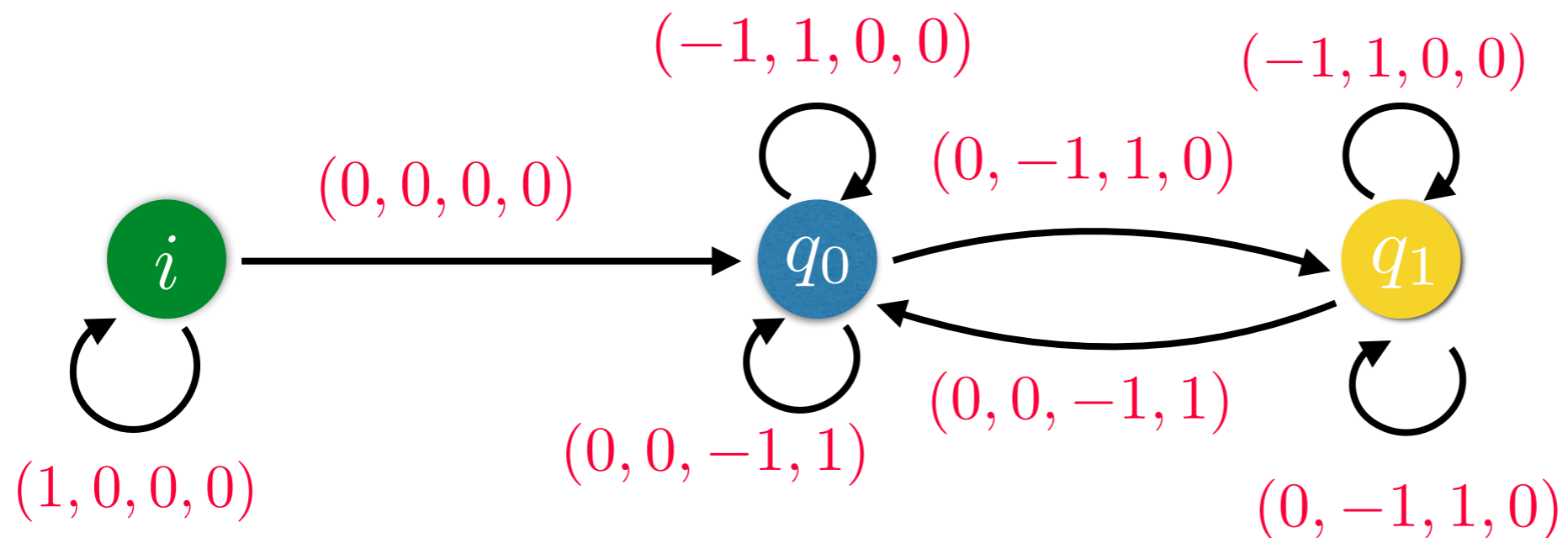


# Mathematical modeling

```
decl s := 0;  
0: goto 1;  
1: s := 1;  
2: s := 0;  
3: assert (!s);
```

An execution:

$(0, 0, 0, 0)$

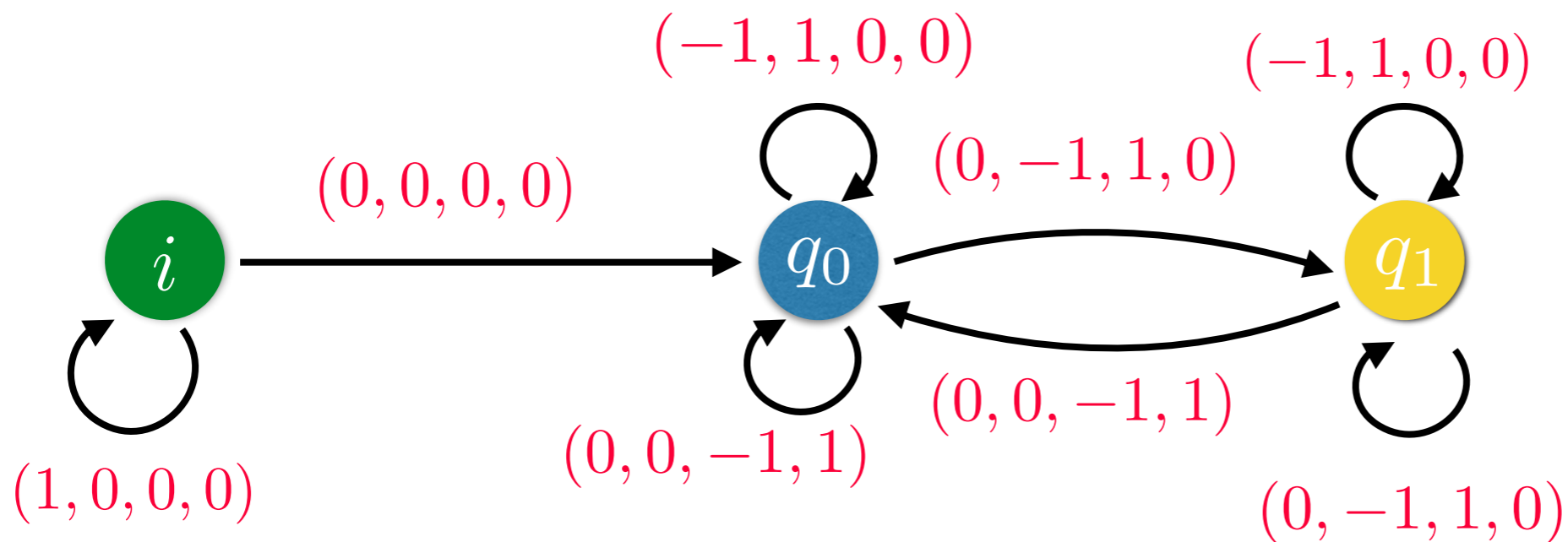


# Mathematical modeling

```
decl s := 0;  
0: goto 1;  
1: s := 1;  
2: s := 0;  
3: assert (!s);
```

An execution:

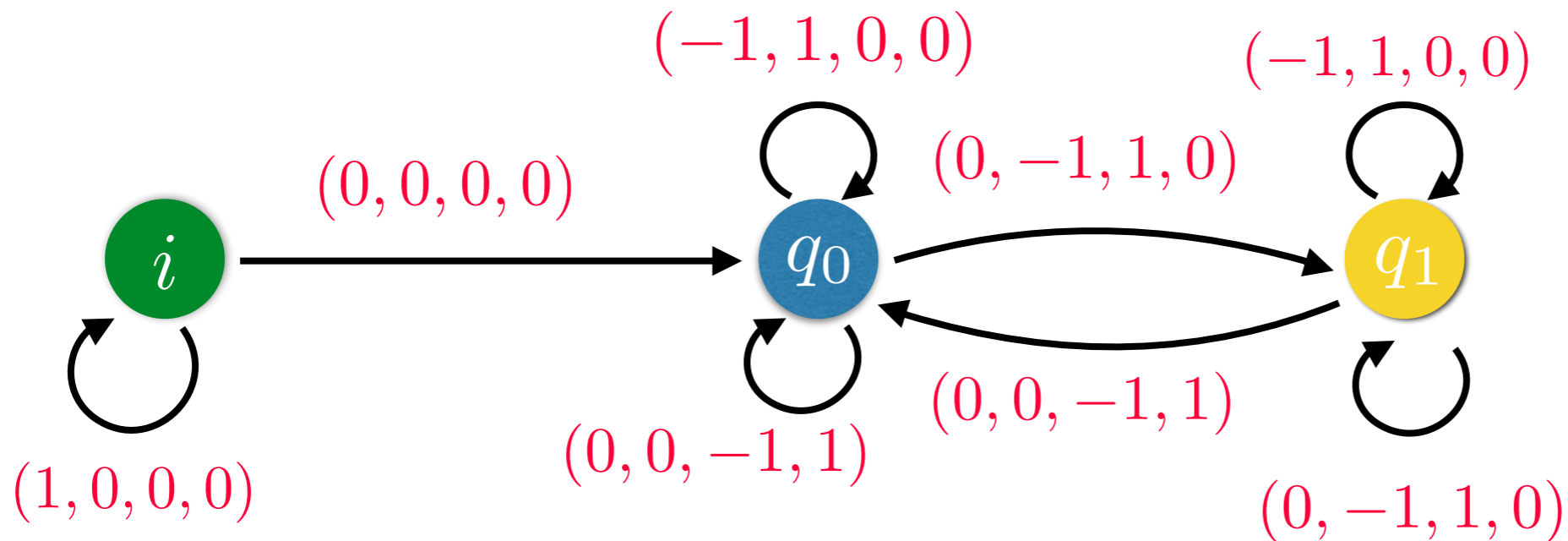
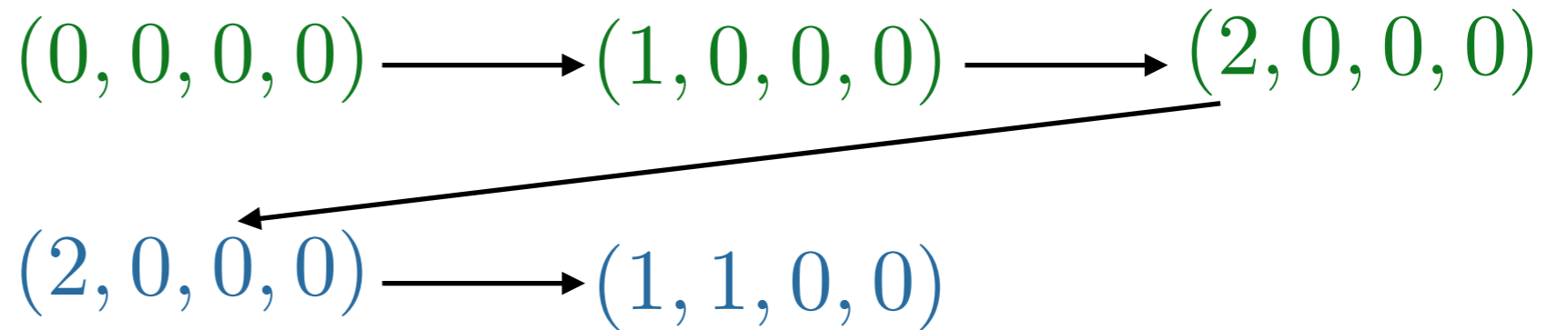
$(0, 0, 0, 0) \longrightarrow (1, 0, 0, 0) \longrightarrow (2, 0, 0, 0)$



# Mathematical modeling

```
decl s := 0;  
0: goto 1;  
1: s := 1;  
2: s := 0;  
3: assert (!s);
```

An execution:

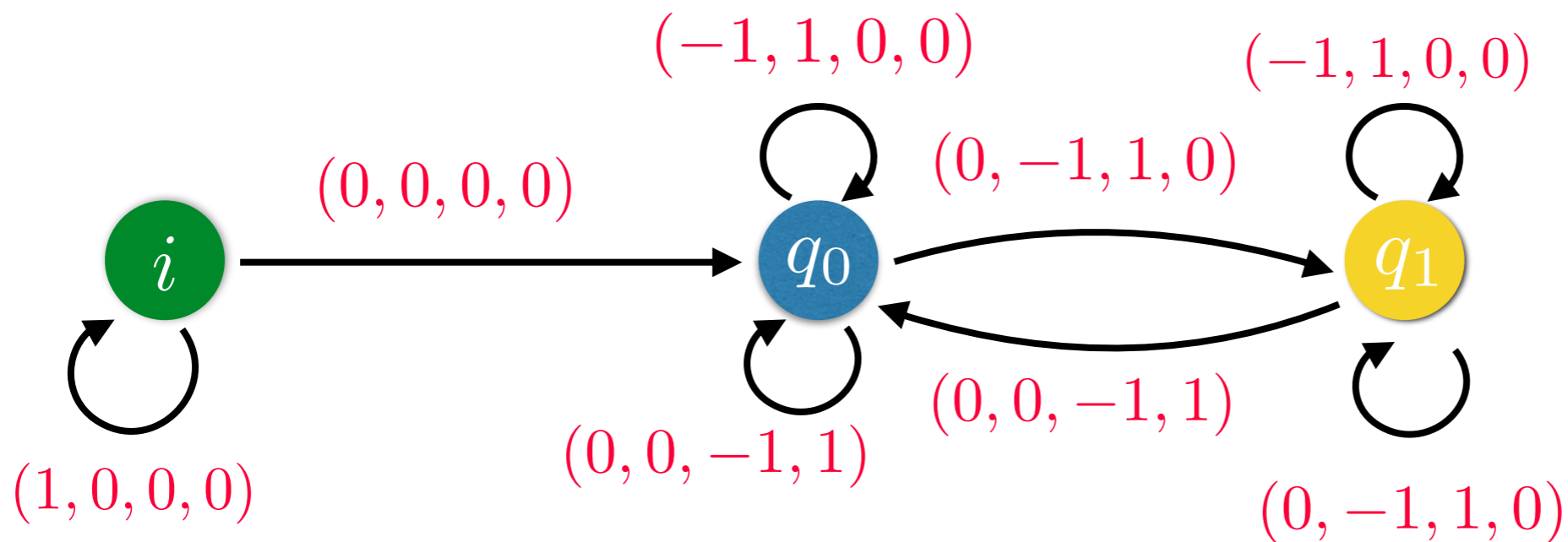
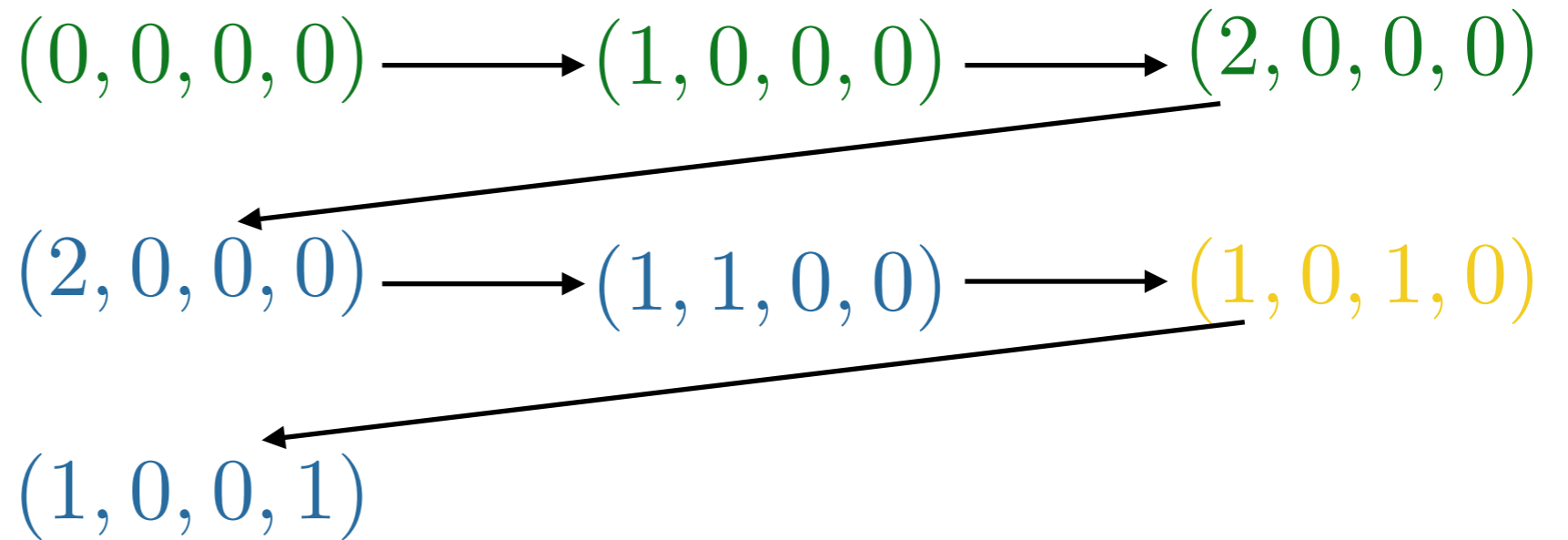


# Mathematical modeling

```

decl s := 0;
0: goto 1;
1: s := 1;
2: s := 0;
3: assert (!s);
    
```

An execution:

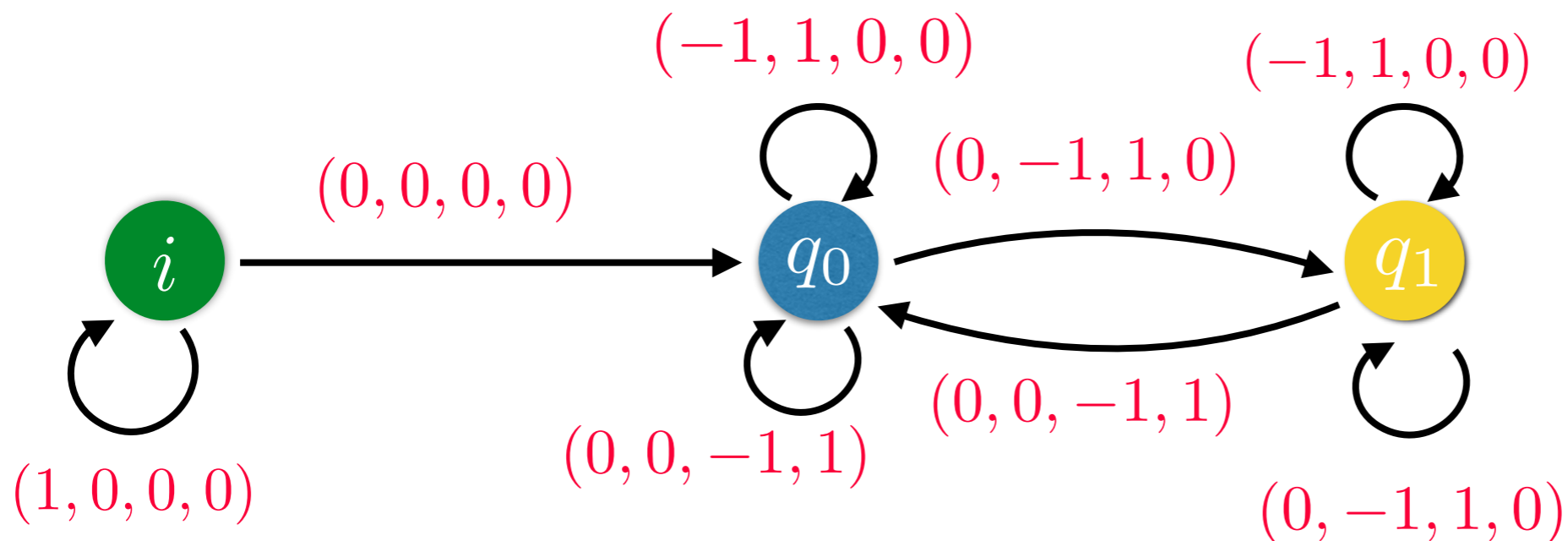
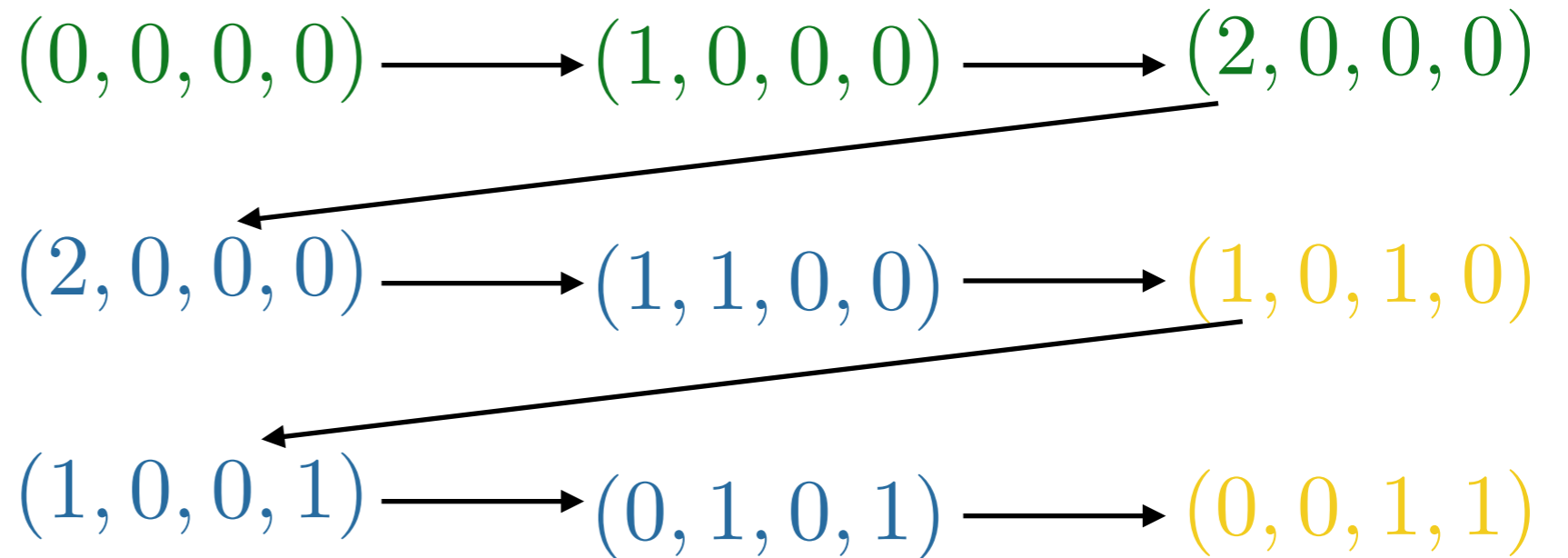


# Mathematical modeling

```

decl s := 0;
0: goto 1;
1: s := 1;
2: s := 0;
3: assert (!s);
    
```

An execution:

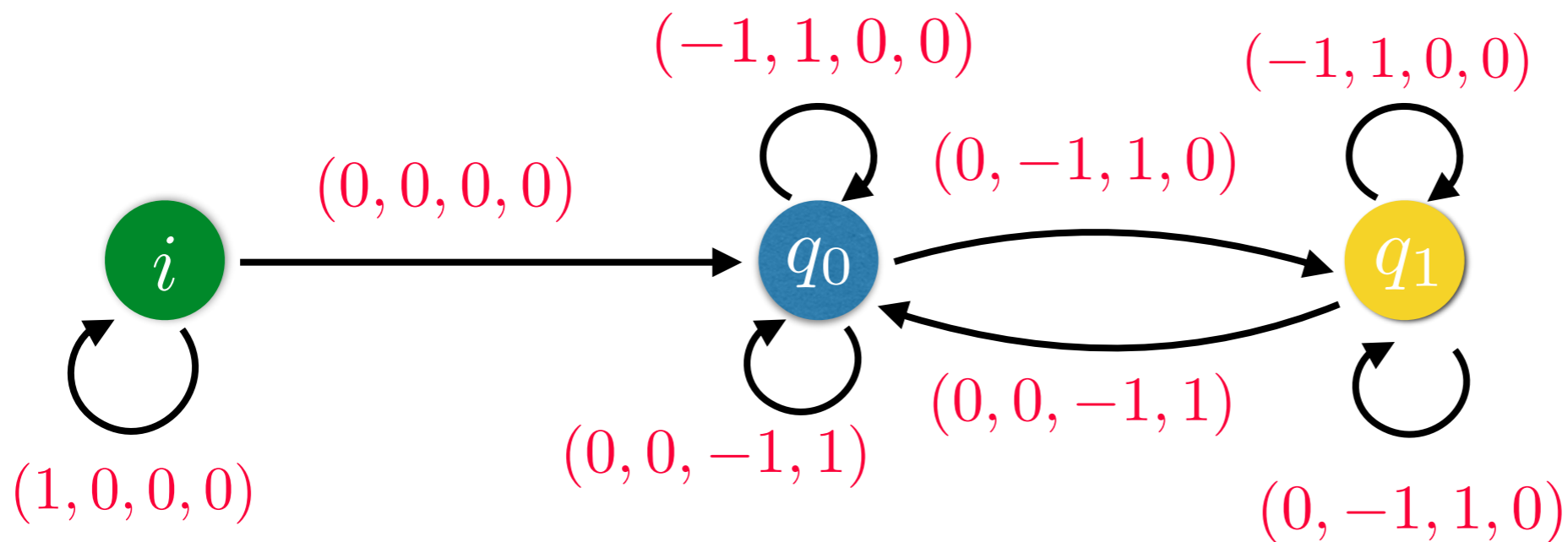
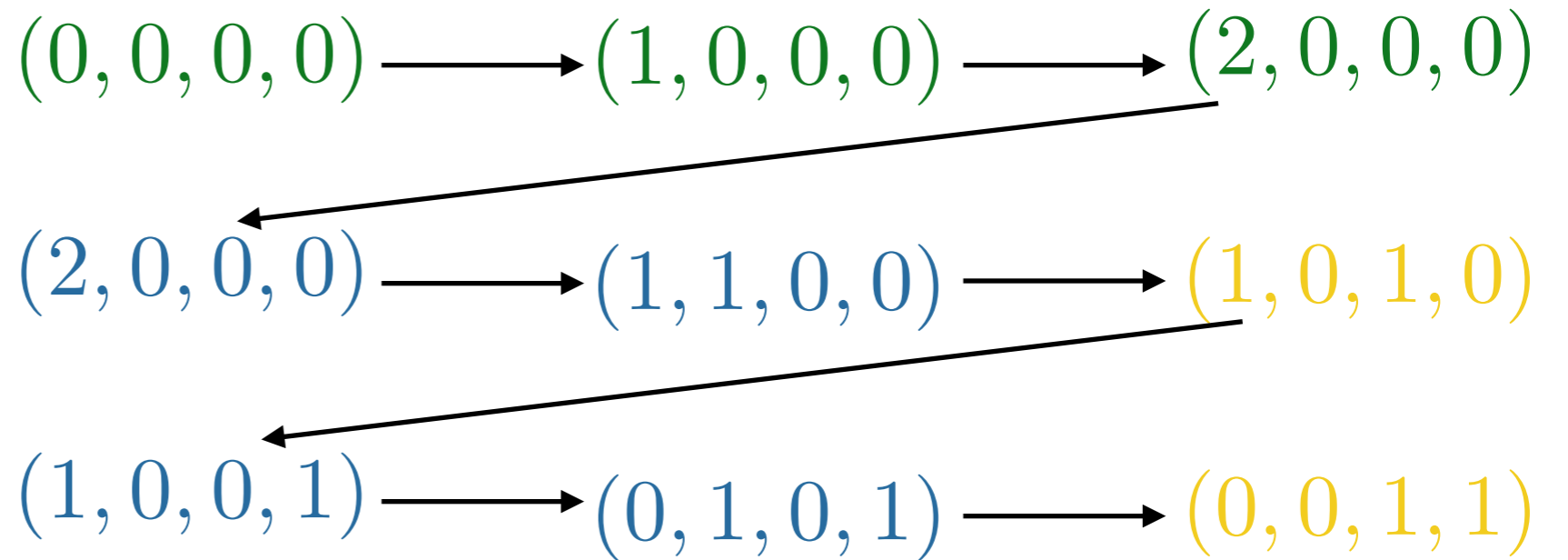


# Mathematical modeling

```

decl s := 0;
0: goto 1;
1: s := 1;
2: s := 0;
3: assert (!s);
    
```

An execution:



# Overview

Some background

Vector addition systems / Presburger reachability

2VASS reachability: Our main result



# Overview

Some background

**Vector addition systems / Presburger reachability**

2VASS reachability: Our main result

# Vector addition systems with states

A **vector addition system with states (d-VASS)** is a finite automaton that consists of

- a finite set of **control states**  $Q$  and

# Vector addition systems with states

A **vector addition system with states (d-VASS)** is a finite automaton that consists of

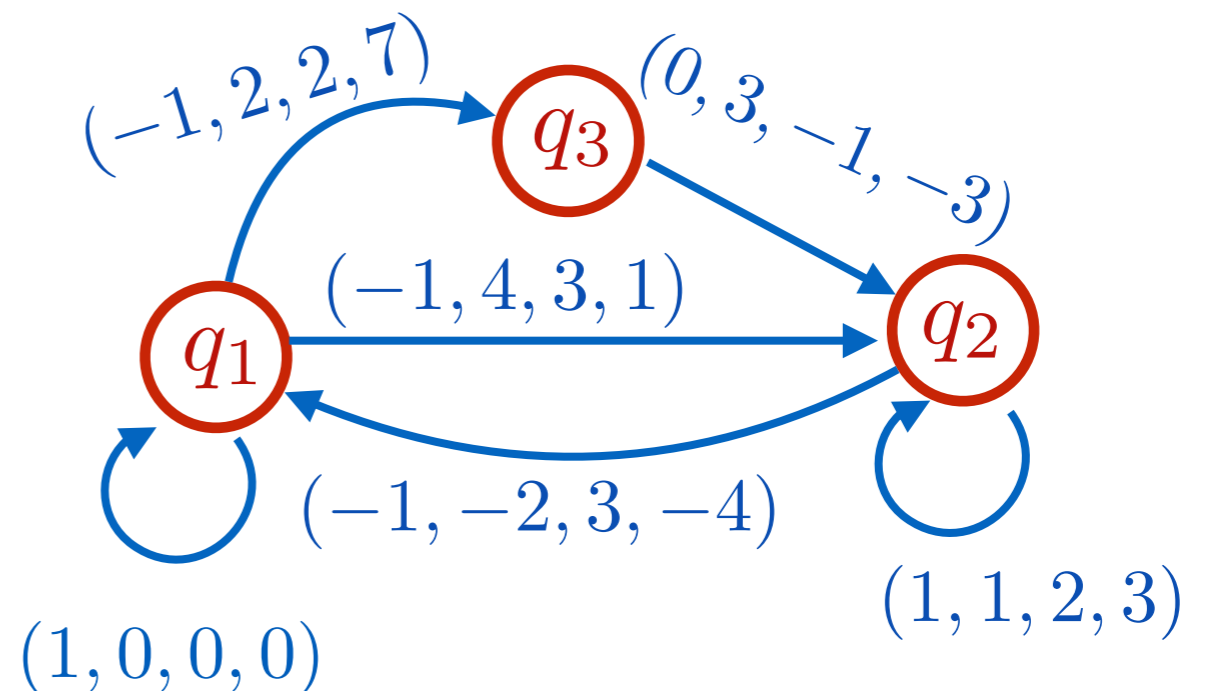
- a finite set of **control states**  $Q$  and
- a set of **transitions**  $T \subseteq Q \times \mathbb{Z}^d \times Q$  .

# Vector addition systems with states

A **vector addition system with states (d-VASS)** is a finite automaton that consists of

- a finite set of **control states**  $Q$  and
- a set of **transitions**  $T \subseteq Q \times \mathbb{Z}^d \times Q$ .

Example of a 4-VASS:



# Underlying infinite system

Each d-VASS  $V = (Q, T)$  defines the infinite system/graph

with vertices  $q(\vec{u}) \in Q \times \mathbb{N}^d$

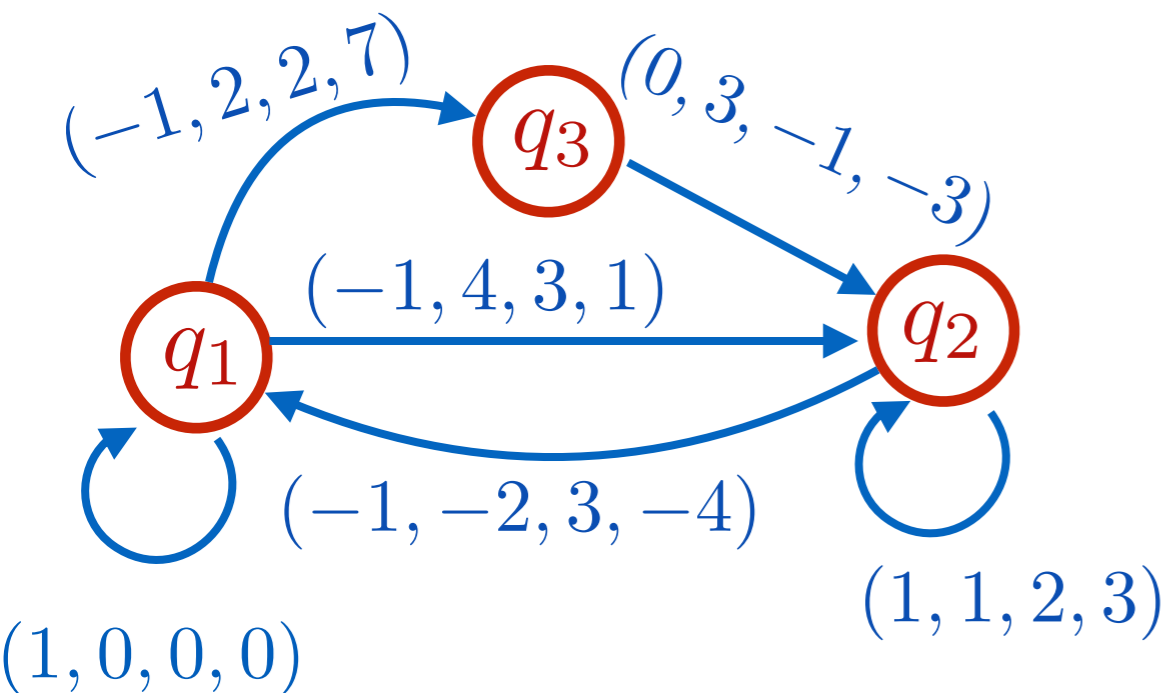
transitions/edges  $p(\vec{u}) \rightarrow q(\vec{v})$  if  $(p, \vec{v} - \vec{u}, q) \in T$

# Underlying infinite system

Each d-VASS  $V = (Q, T)$  defines the infinite system/graph

with **vertices/configurations**  $q(\vec{u}) \in Q \times \mathbb{N}^d$

**transitions/edges**  $p(\vec{u}) \rightarrow q(\vec{v})$  if  $(p, \vec{v} - \vec{u}, q) \in T$



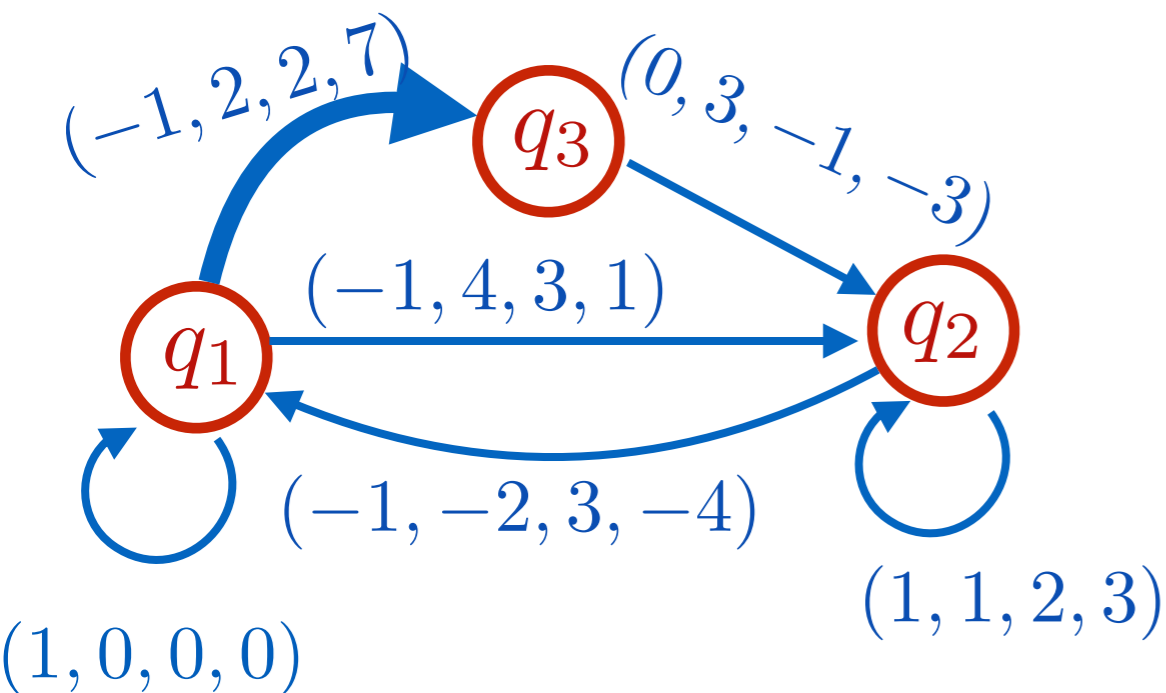
A 4-VASS

# Underlying infinite system

Each d-VASS  $V = (Q, T)$  defines the infinite system/graph

with **vertices/configurations**  $q(\vec{u}) \in Q \times \mathbb{N}^d$

**transitions/edges**  $p(\vec{u}) \rightarrow q(\vec{v})$  if  $(p, \vec{v} - \vec{u}, q) \in T$



A 4-VASS

$$q_1(1, 0, 0, 0) \rightarrow q_3(0, 2, 2, 7)$$

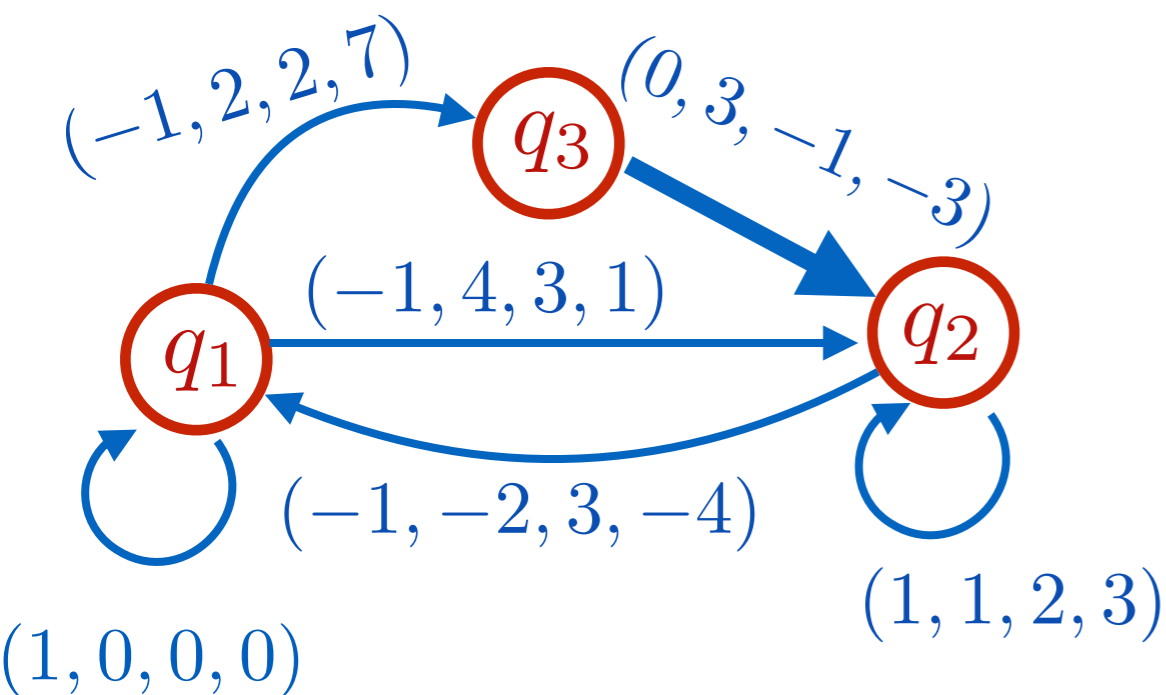
Underlying infinite system

# Underlying infinite system

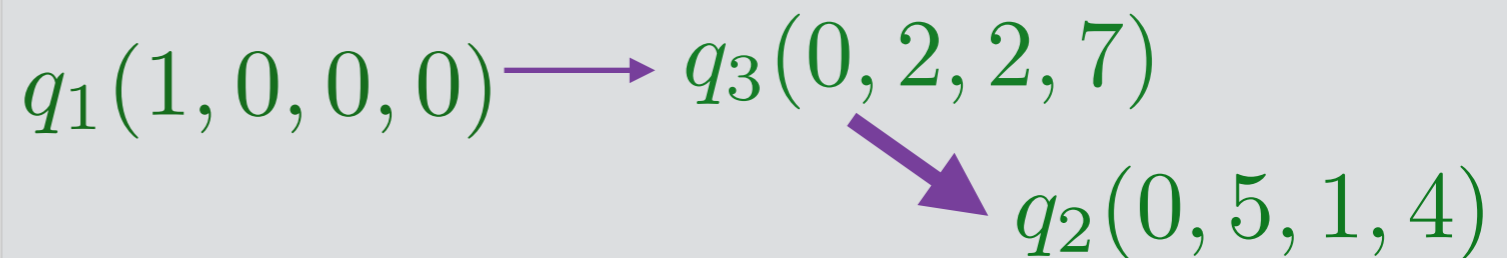
Each d-VASS  $V = (Q, T)$  defines the infinite system/graph

with **vertices/configurations**  $q(\vec{u}) \in Q \times \mathbb{N}^d$

**transitions/edges**  $p(\vec{u}) \rightarrow q(\vec{v})$  if  $(p, \vec{v} - \vec{u}, q) \in T$



A 4-VASS



Underlying infinite system

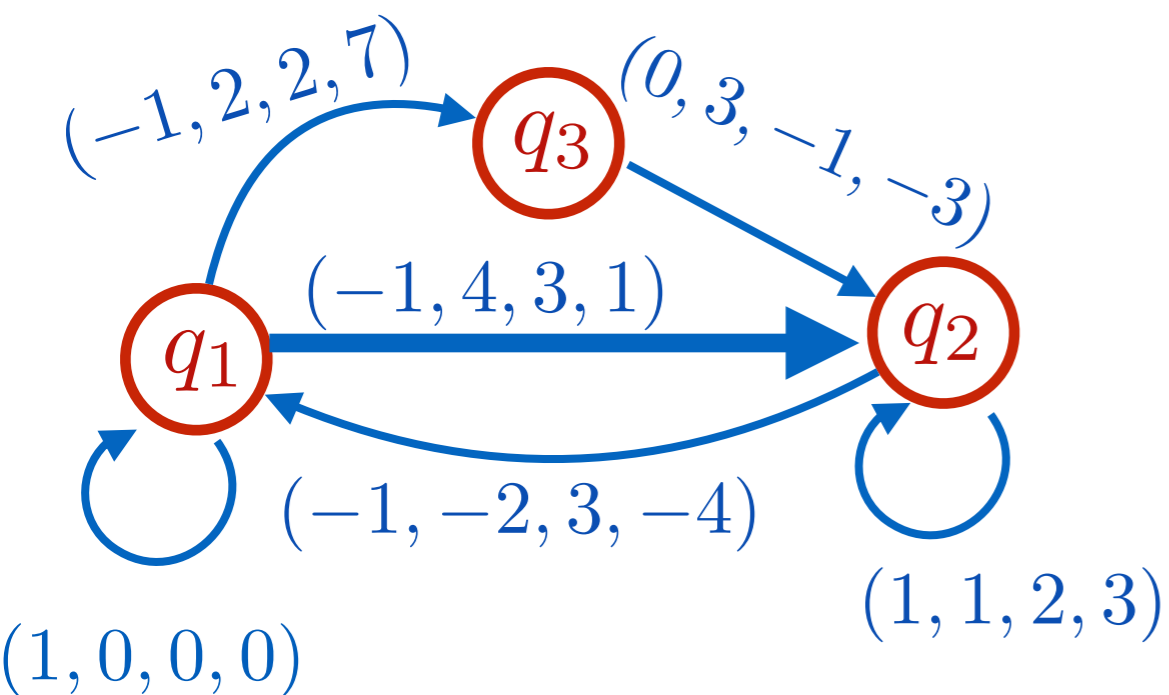


# Underlying infinite system

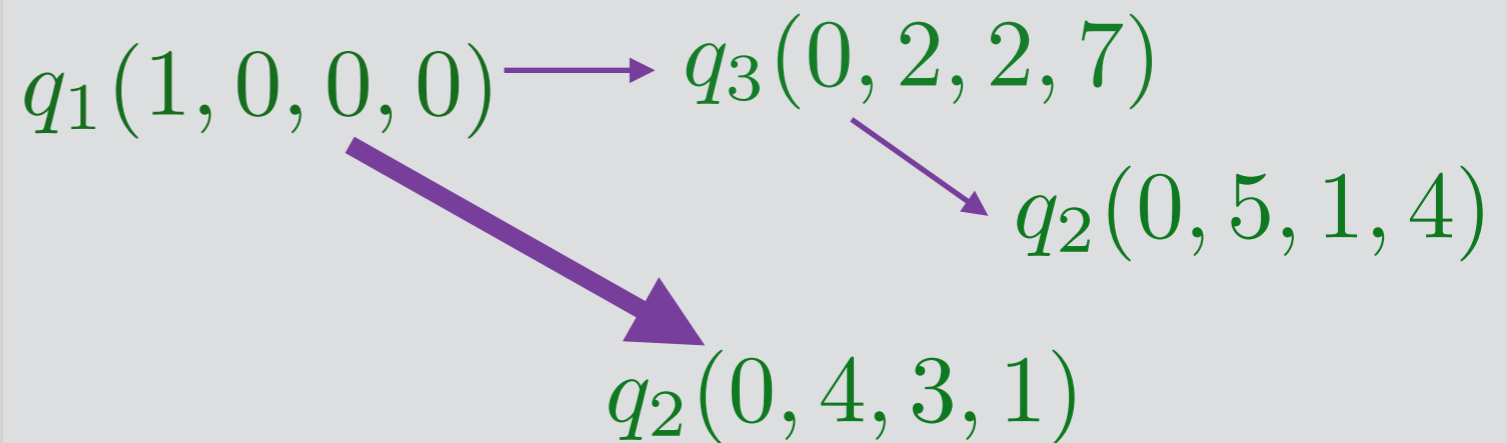
Each d-VASS  $V = (Q, T)$  defines the infinite system/graph

with **vertices/configurations**  $q(\vec{u}) \in Q \times \mathbb{N}^d$

**transitions/edges**  $p(\vec{u}) \rightarrow q(\vec{v})$  if  $(p, \vec{v} - \vec{u}, q) \in T$



A 4-VASS



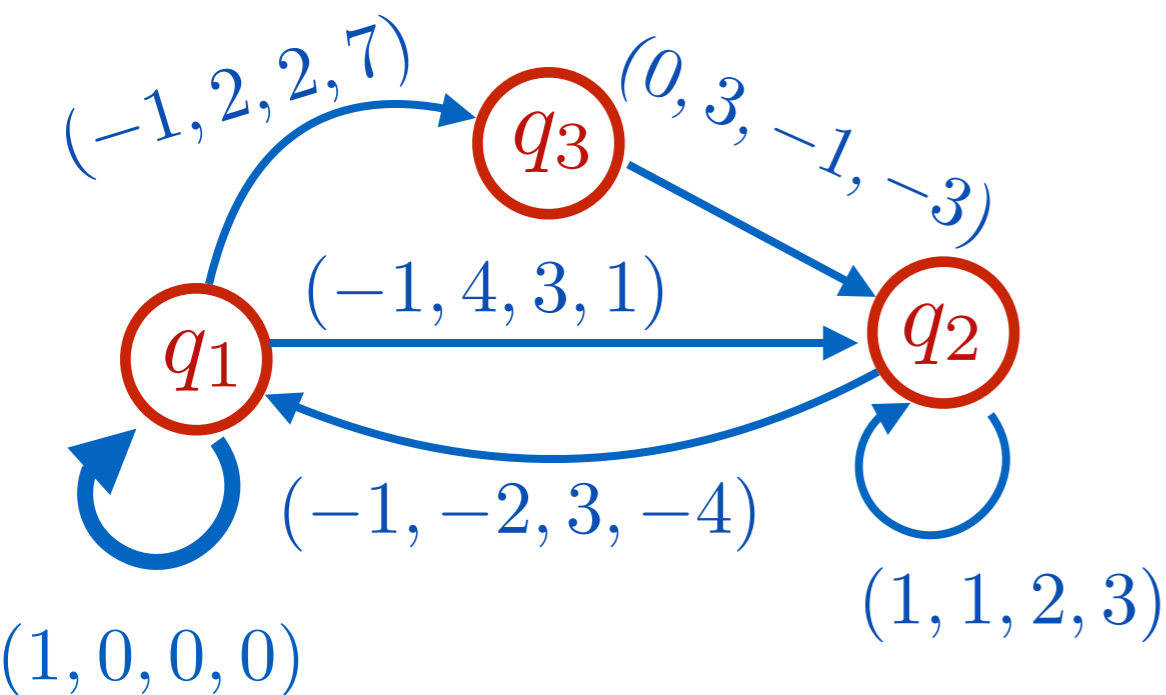
Underlying infinite system

# Underlying infinite system

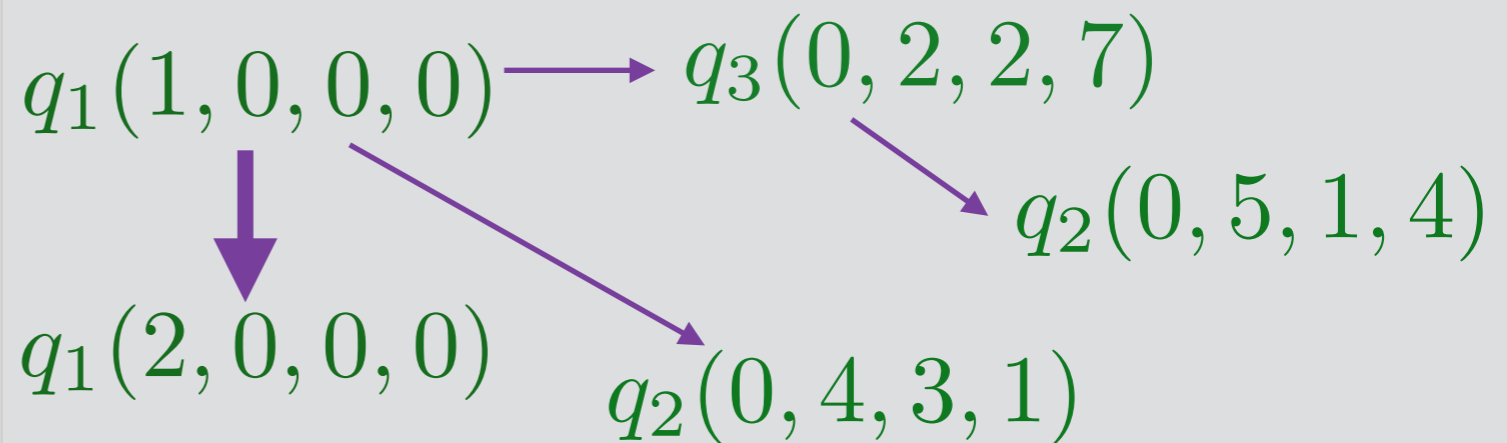
Each d-VASS  $V = (Q, T)$  defines the infinite system/graph

with **vertices/configurations**  $q(\vec{u}) \in Q \times \mathbb{N}^d$

**transitions/edges**  $p(\vec{u}) \rightarrow q(\vec{v})$  if  $(p, \vec{v} - \vec{u}, q) \in T$



A 4-VASS



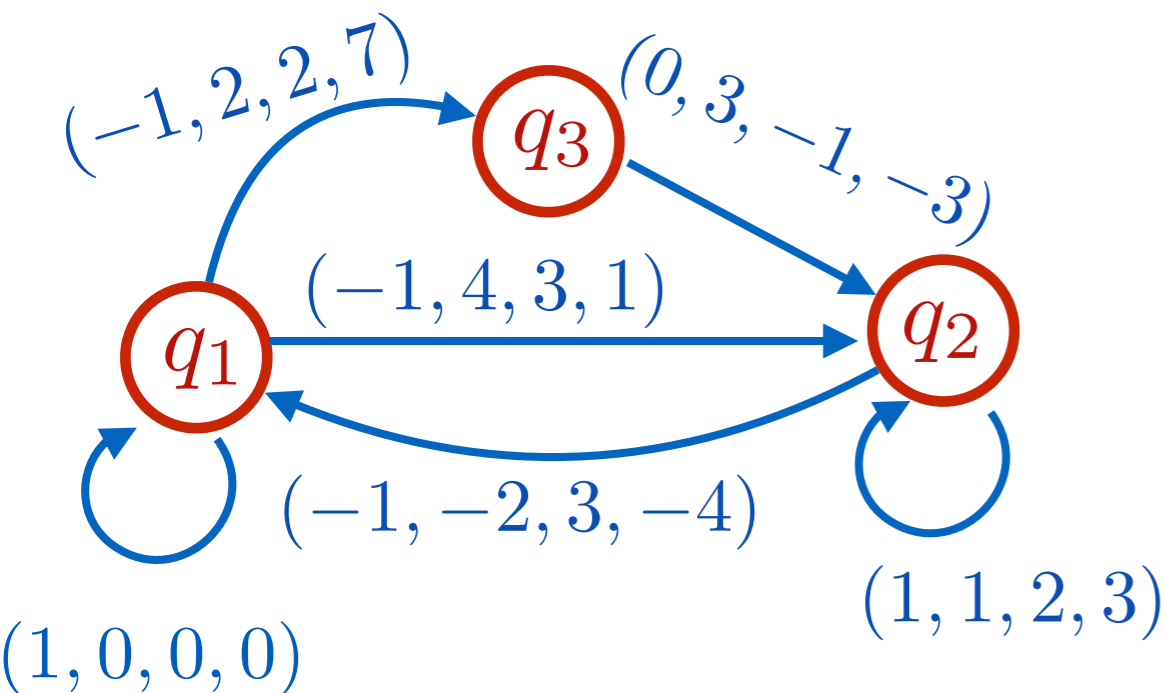
Underlying infinite system

# Underlying infinite system

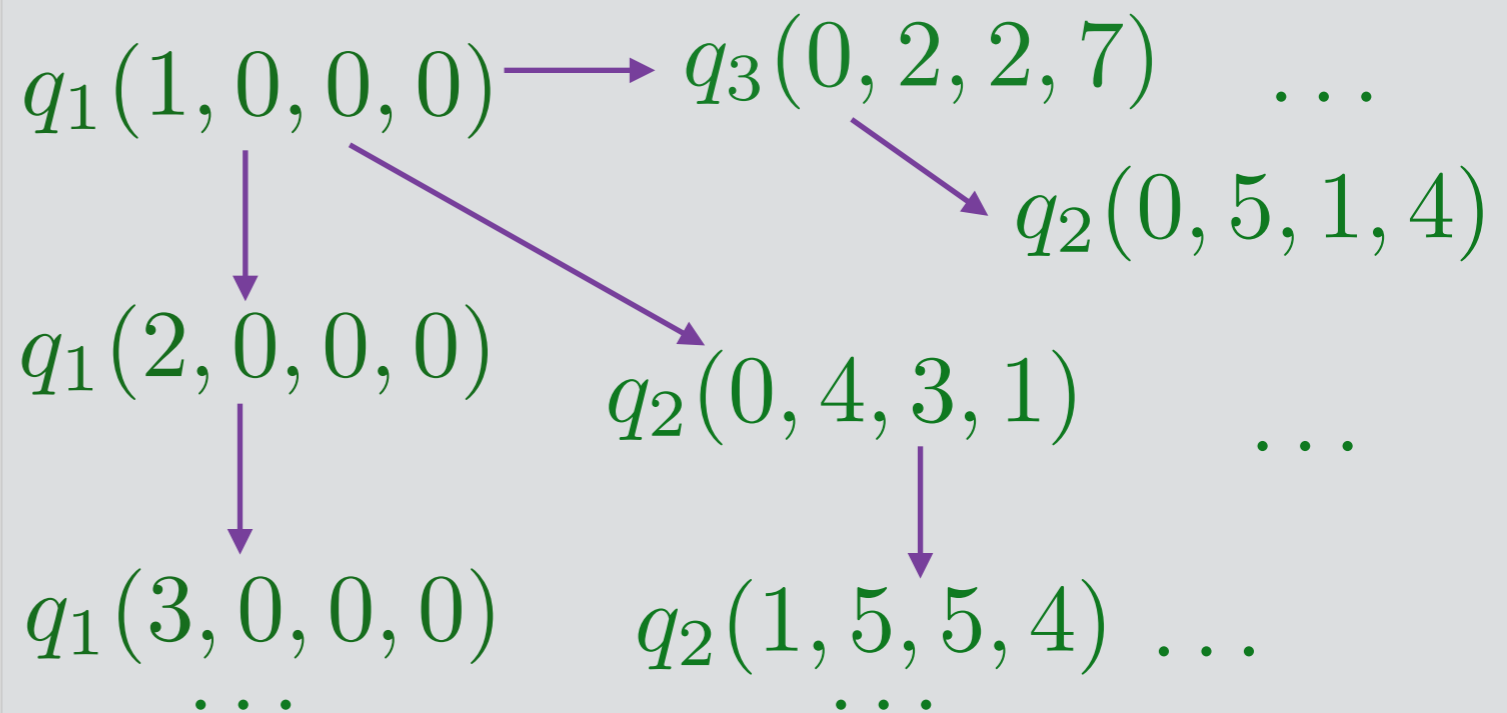
Each d-VASS  $V = (Q, T)$  defines the infinite system/graph

with **vertices/configurations**  $q(\vec{u}) \in Q \times \mathbb{N}^d$

**transitions/edges**  $p(\vec{u}) \rightarrow q(\vec{v})$  if  $(p, \vec{v} - \vec{u}, q) \in T$

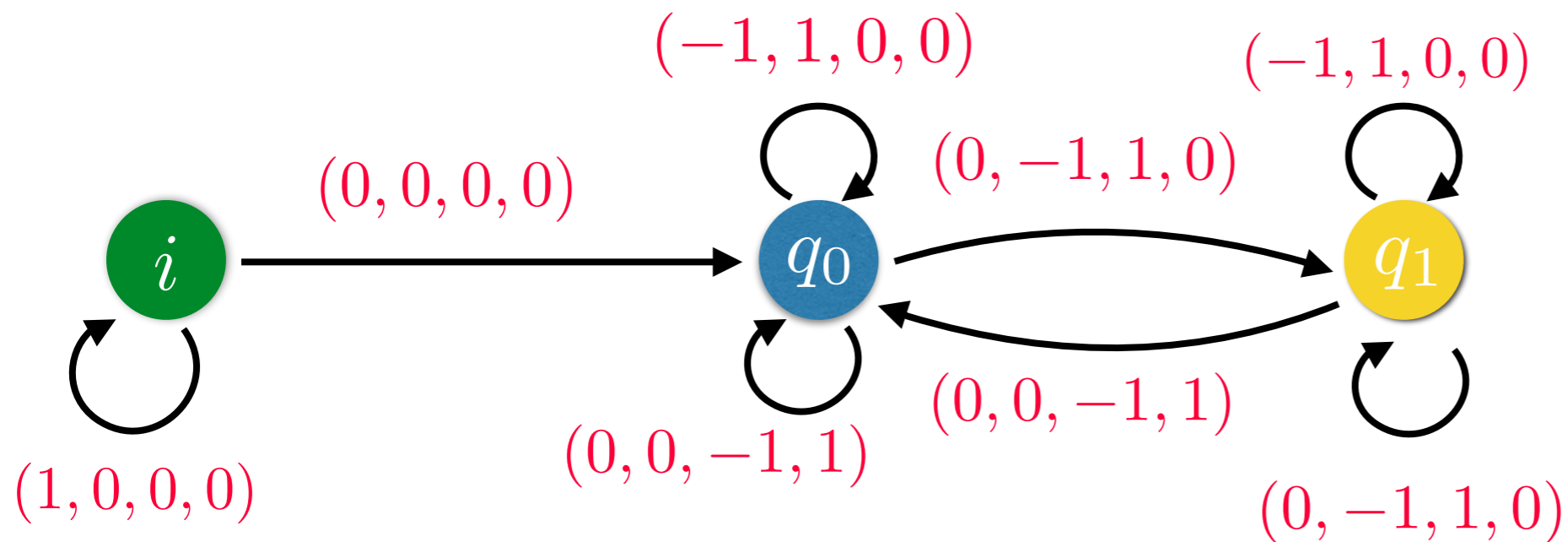
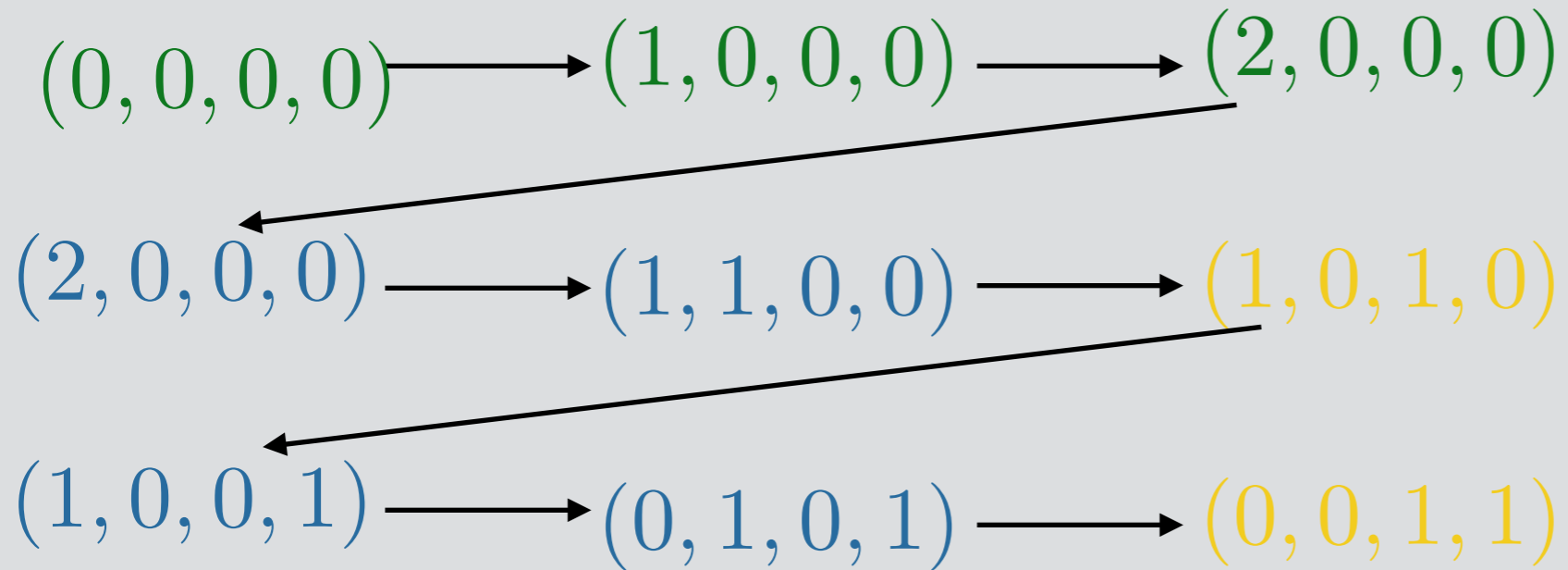


A 4-VASS



Underlying infinite system

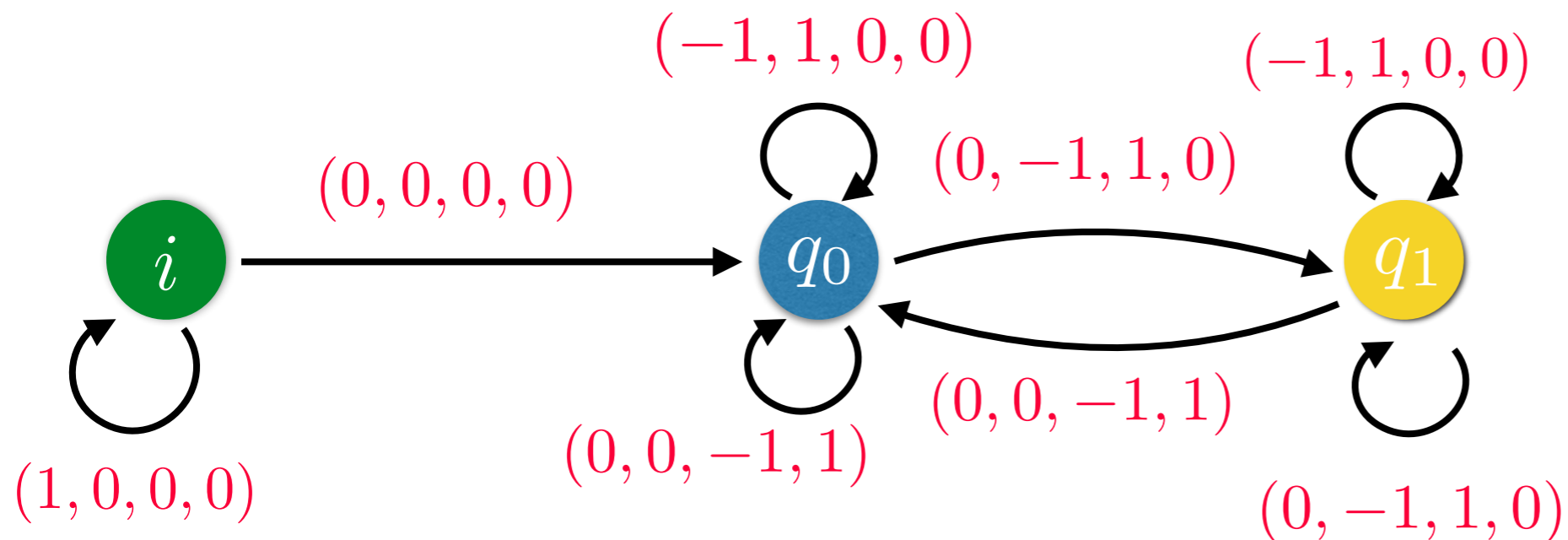
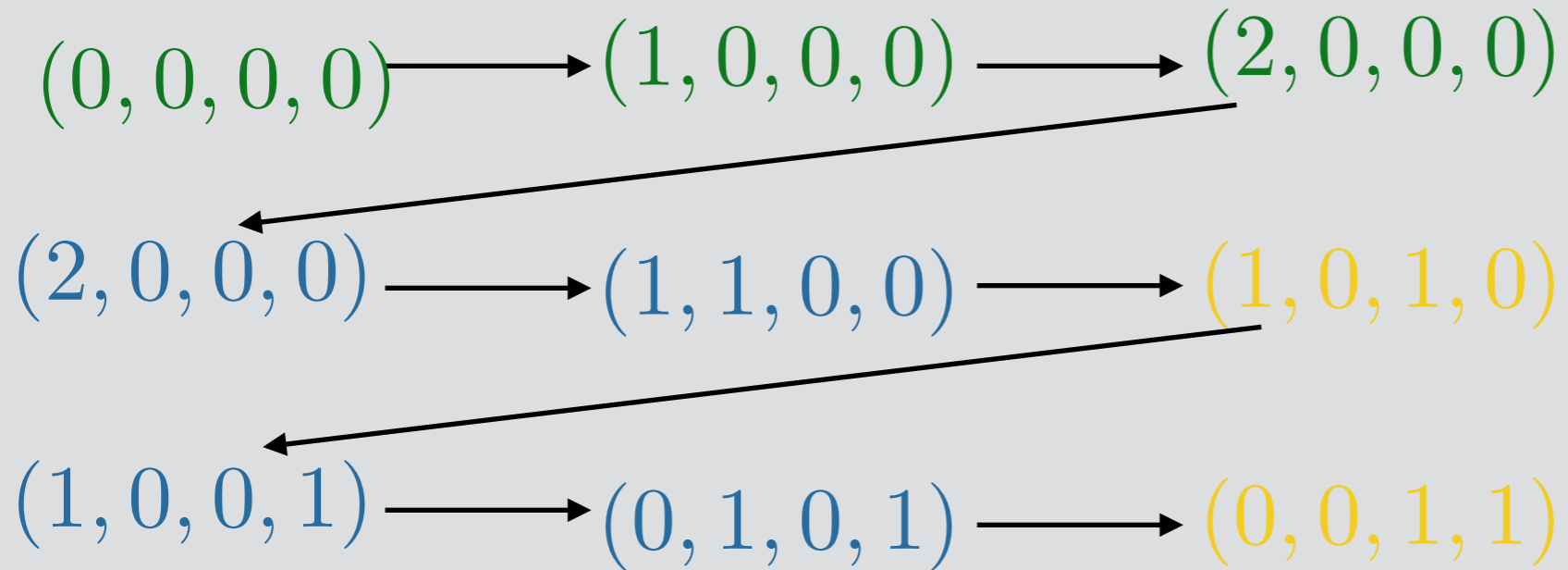
# Decision problems



# Decision problems

## Reachability:

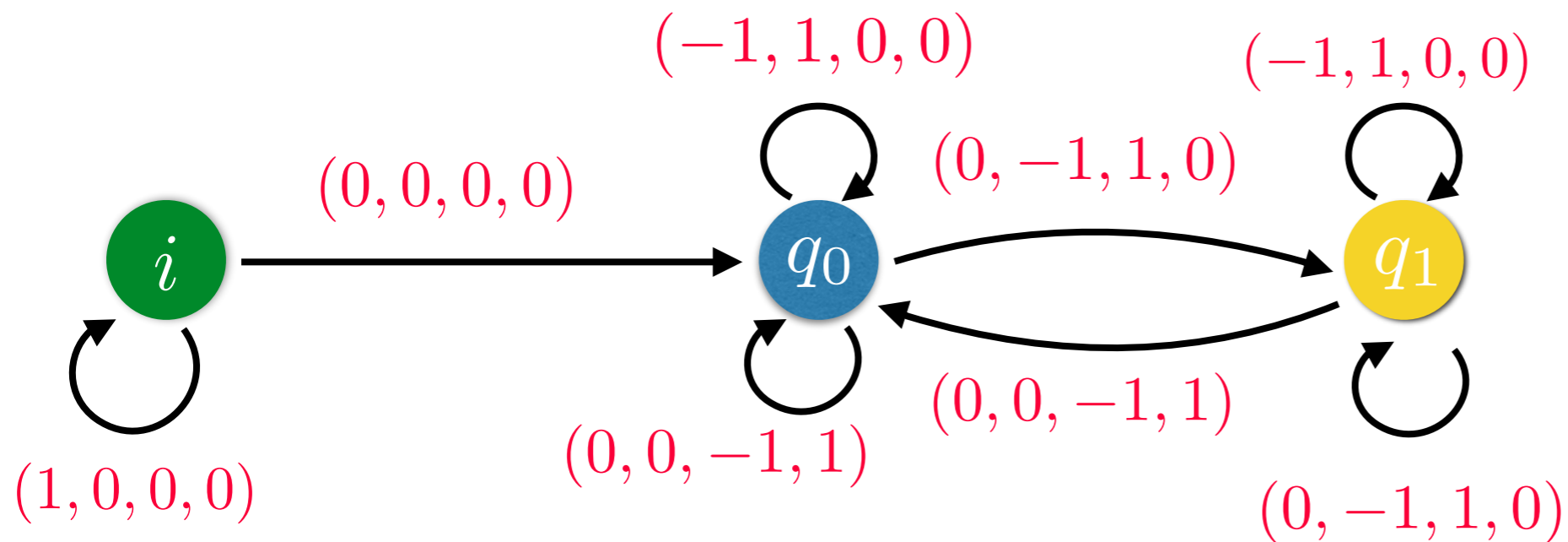
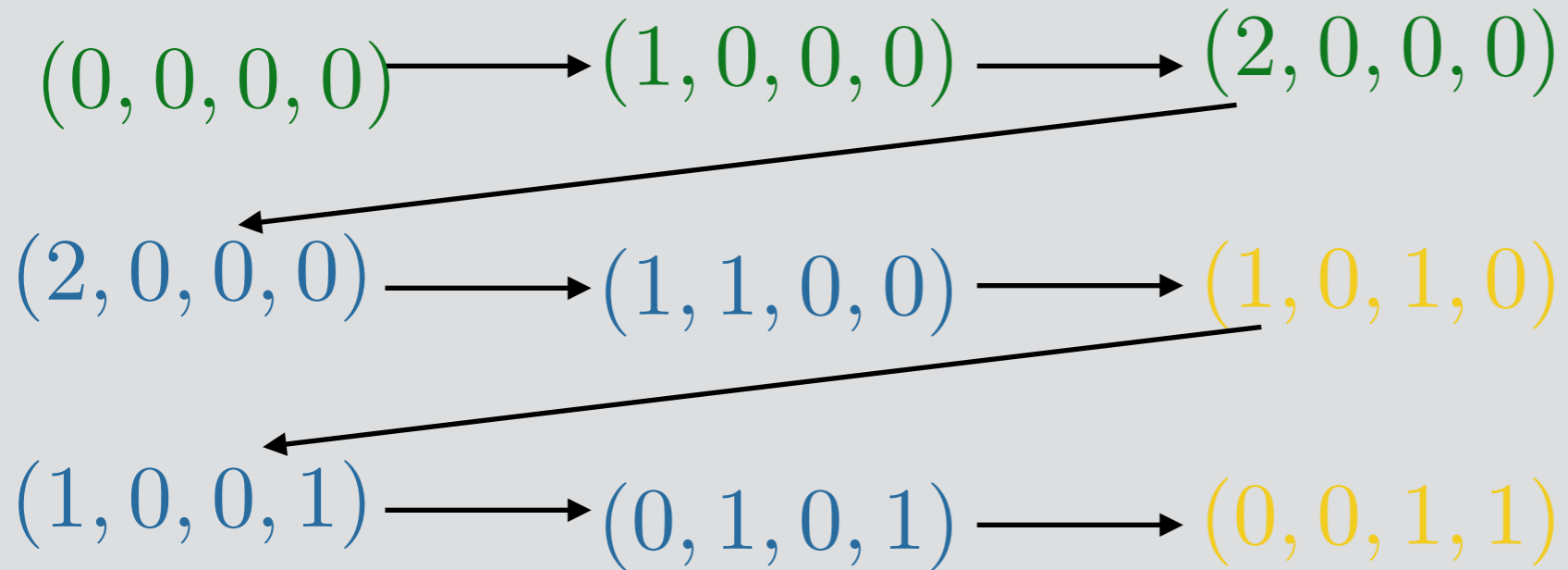
Can  $(1, 2, 3, 4)$   
reach  $(7, 8, 3, 4)$  ?



# Decision problems

**Reachability:** ✗

Can  $(1, 2, 3, 4)$   
reach  $(7, 8, 3, 4)$  ?



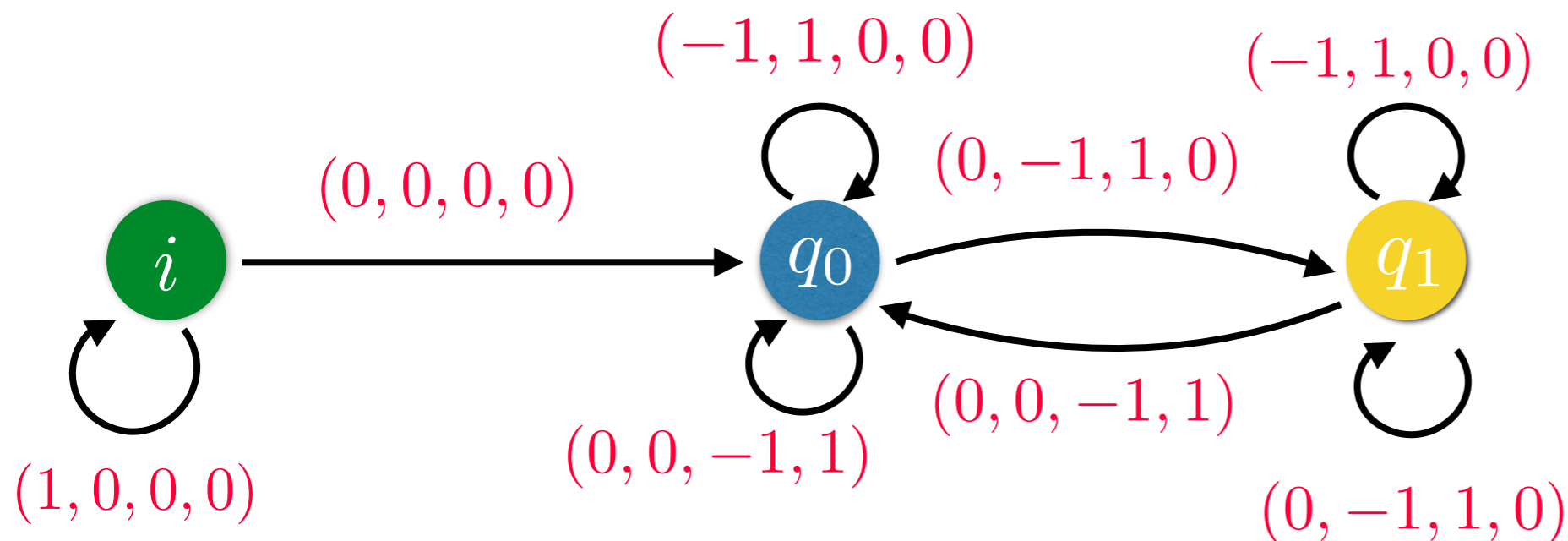
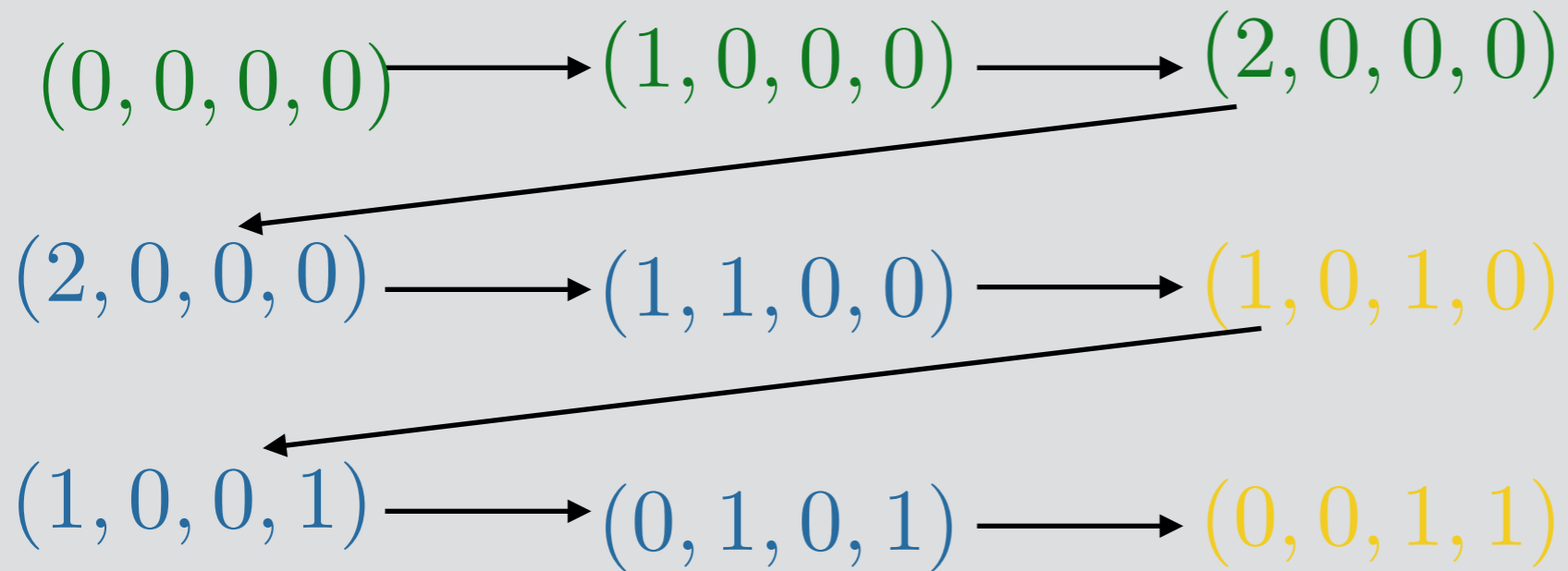
# Decision problems

**Reachability:** ✗

Can  $(1, 2, 3, 4)$   
reach  $(7, 8, 3, 4)$  ?

**Coverability:**

Can  $(0, 0, 0, 0)$   
reach  $(u, v, x, y)$   
with  $x, y \geq 1$  ?



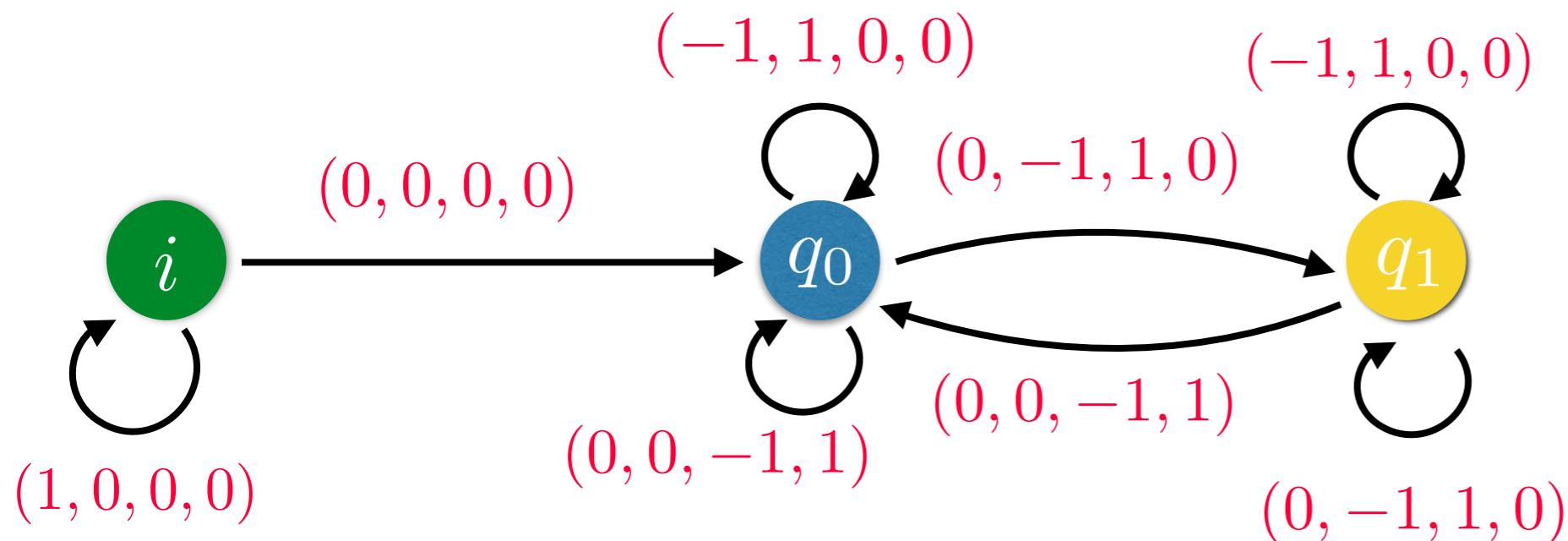
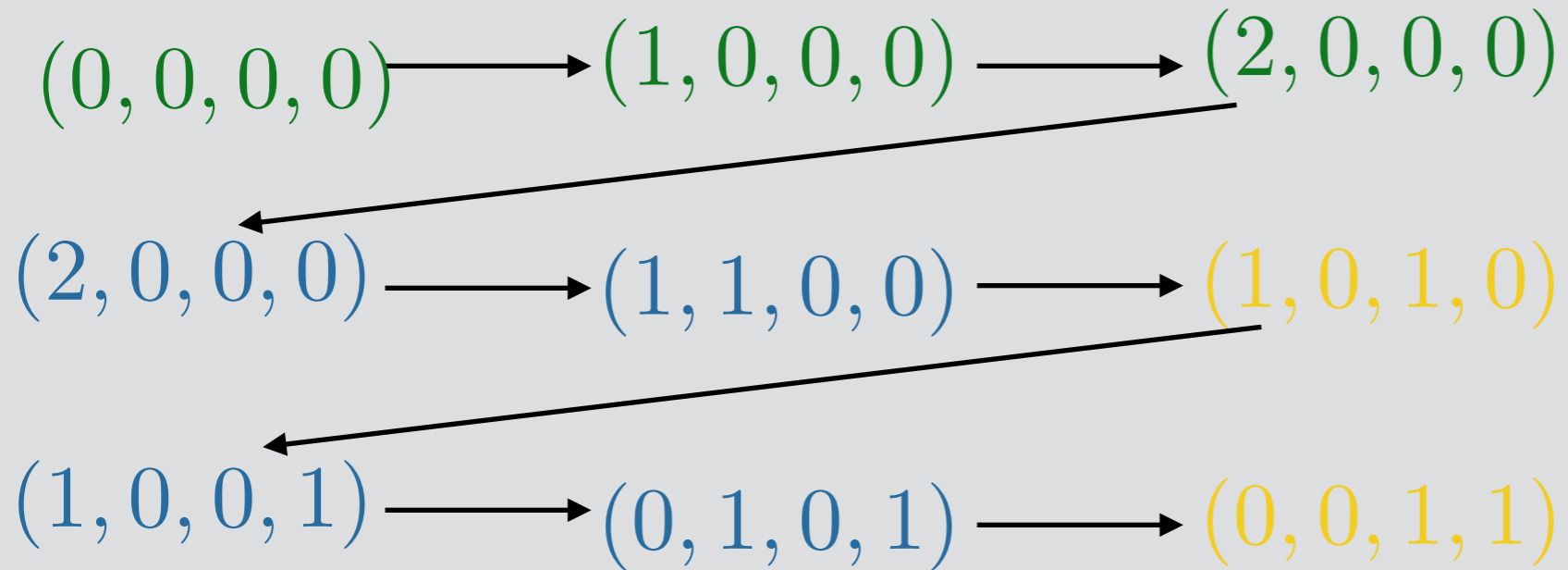
# Decision problems

**Reachability:** ❌

Can  $(1, 2, 3, 4)$   
reach  $(7, 8, 3, 4)$  ?

**Coverability:** ✅

Can  $(0, 0, 0, 0)$   
reach  $(u, v, x, y)$   
with  $x, y \geq 1$  ?





# Decision problems

**Reachability:** ❌

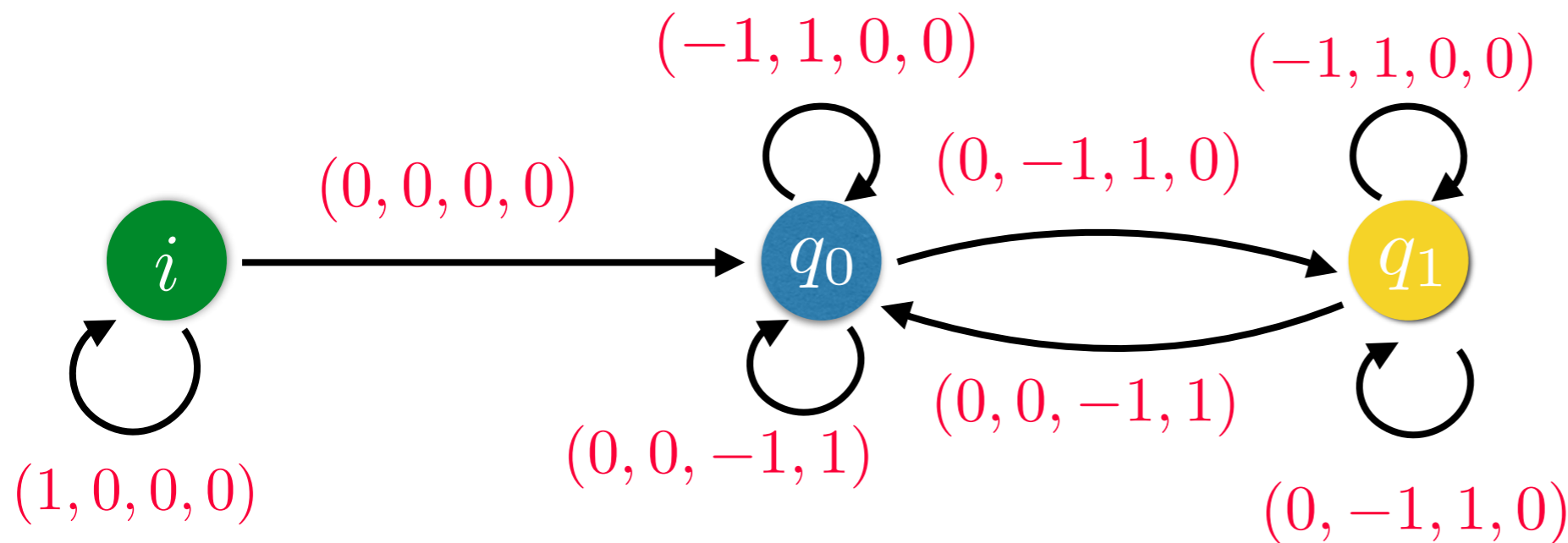
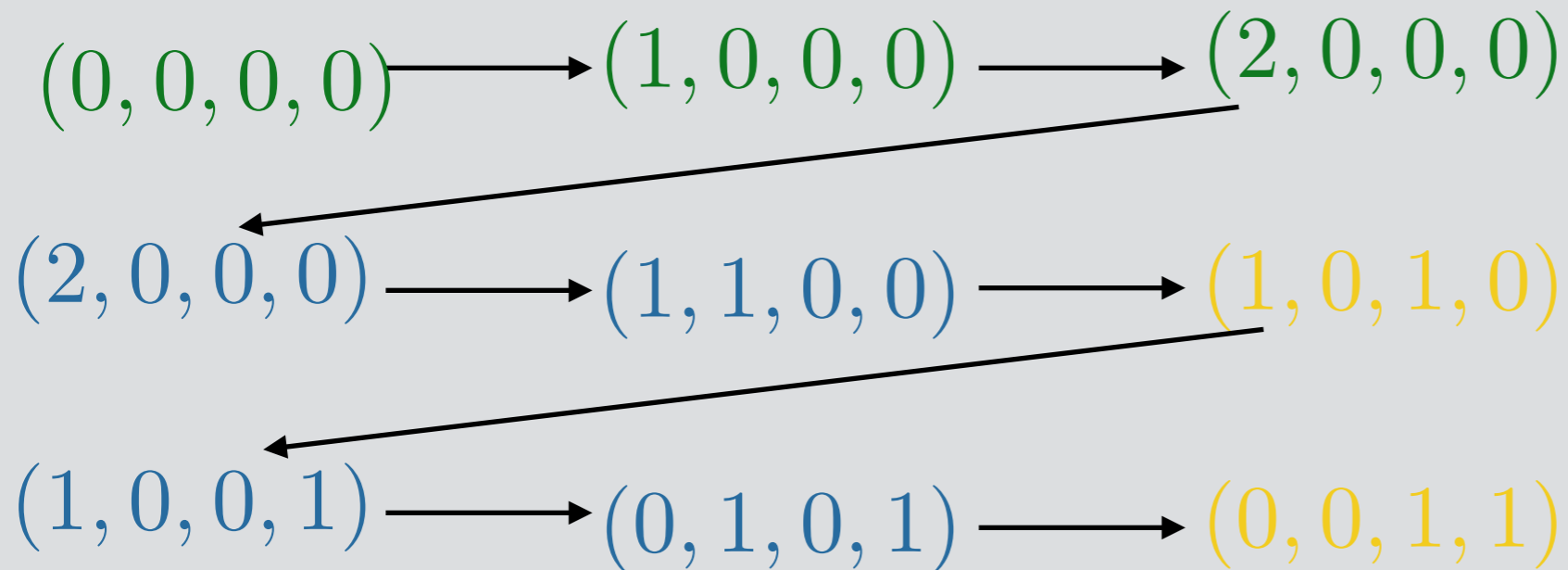
Can  $(1, 2, 3, 4)$   
reach  $(7, 8, 3, 4)$  ?

**Coverability:** ✅

Can  $(0, 0, 0, 0)$   
reach  $(u, v, x, y)$   
with  $x, y \geq 1$  ?

**Boundedness:**

Can  $(0, 0, 0, 0)$   
reach only finitely  
many config.?



# Decision problems

**Reachability:** ❌

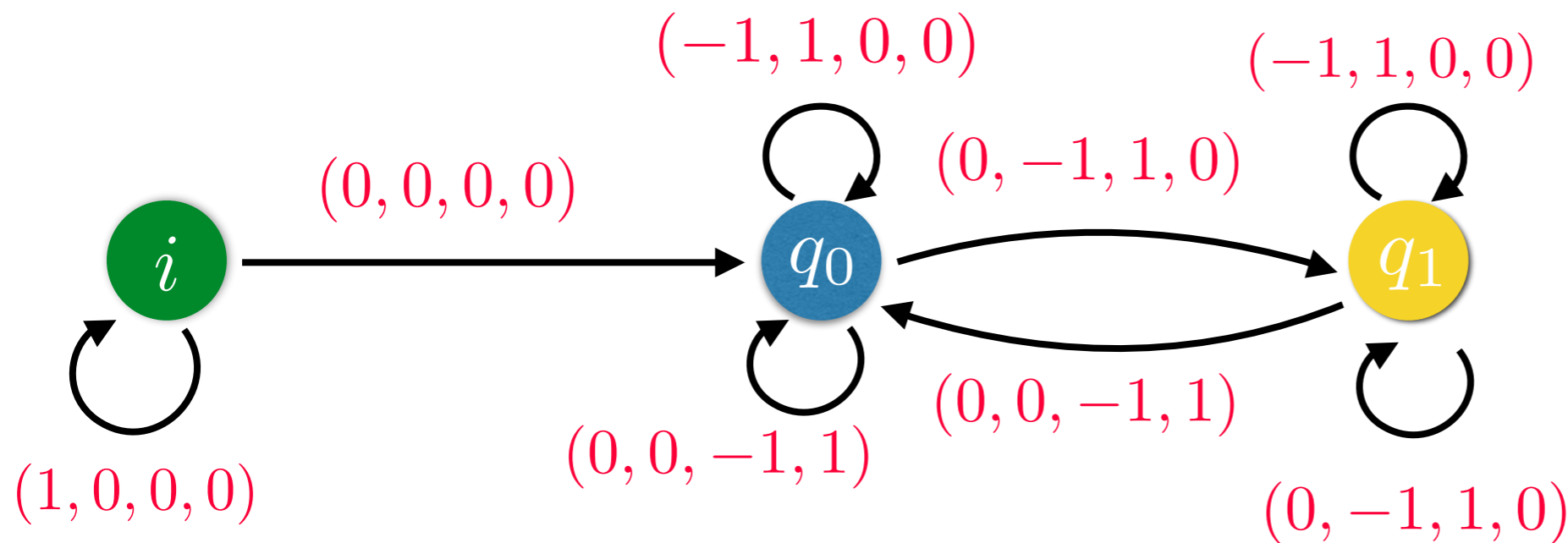
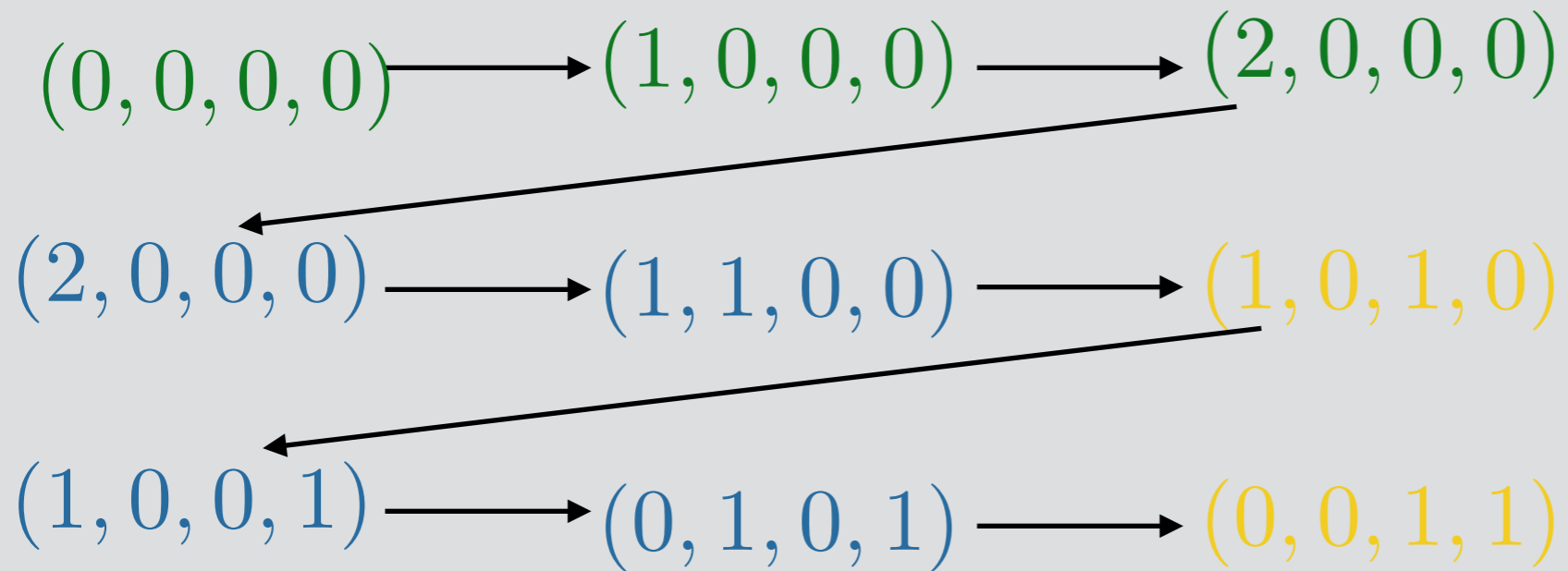
Can  $(1, 2, 3, 4)$   
reach  $(7, 8, 3, 4)$  ?

**Coverability:** ✅

Can  $(0, 0, 0, 0)$   
reach  $(u, v, x, y)$   
with  $x, y \geq 1$  ?

**Boundedness:** ❌

Can  $(0, 0, 0, 0)$   
reach only finitely  
many config.?



# Vector addition systems in computer science

## Verification:

- Concurrent and recursive programs
- Heap-manipulating programs



C.A. Petri

# Vector addition systems in computer science

## Verification:

- Concurrent and recursive programs
- Heap-manipulating programs

## Modeling:

- Workflow modeling
- Business processes



C.A. Petri

# Vector addition systems in computer science

## Verification:

- Concurrent and recursive programs
- Heap-manipulating programs

## Modeling:

- Workflow modeling
- Business processes

## Mathematical and computational logic:

- Hilbert's 10th problem
- Data logics



C.A. Petri

# The reachability problem for VASS

undecidable

**VASS with zero tests**  
Minsky 1961

# The reachability problem for VASS

undecidable

decidable

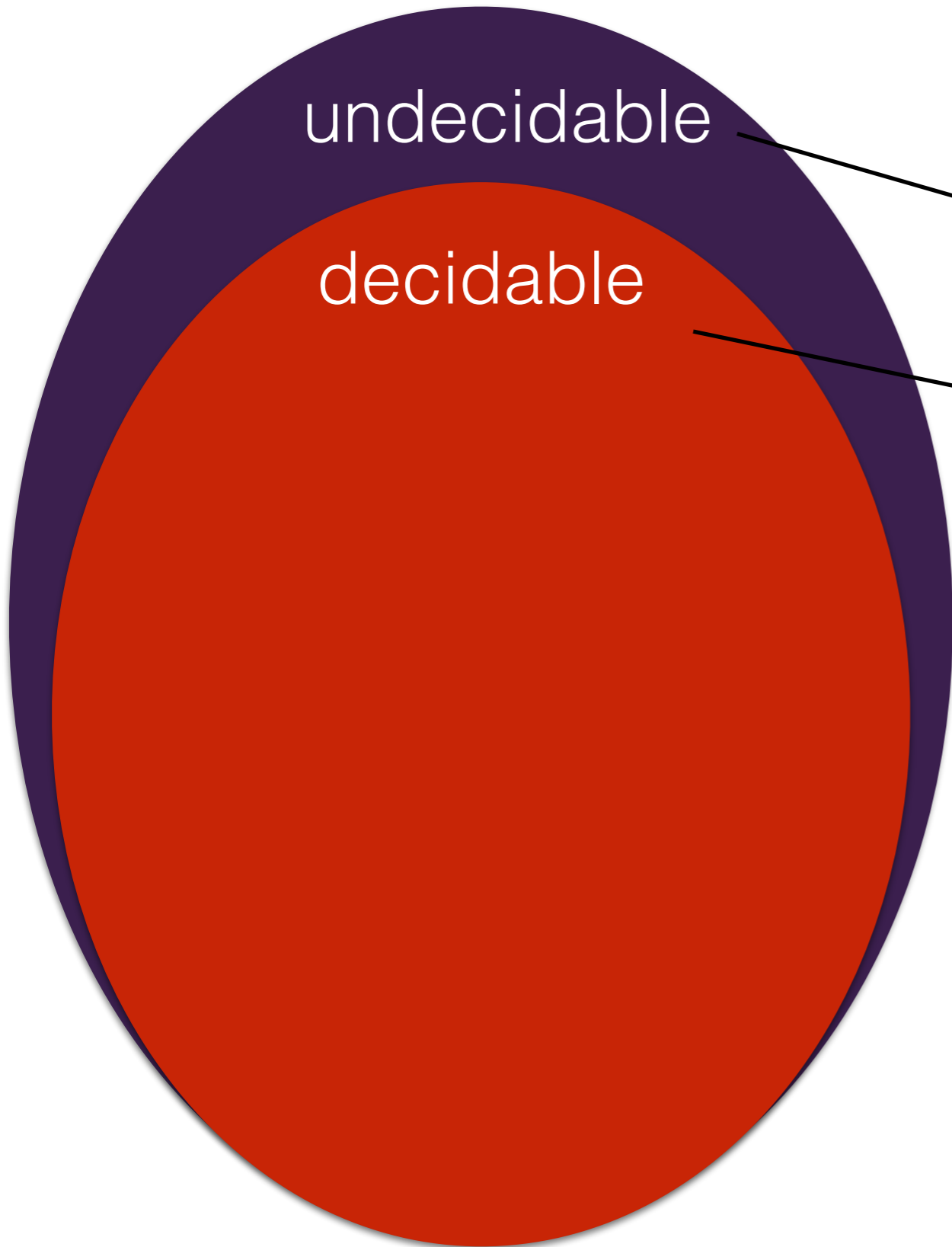
**VASS with zero tests**

Minsky 1961

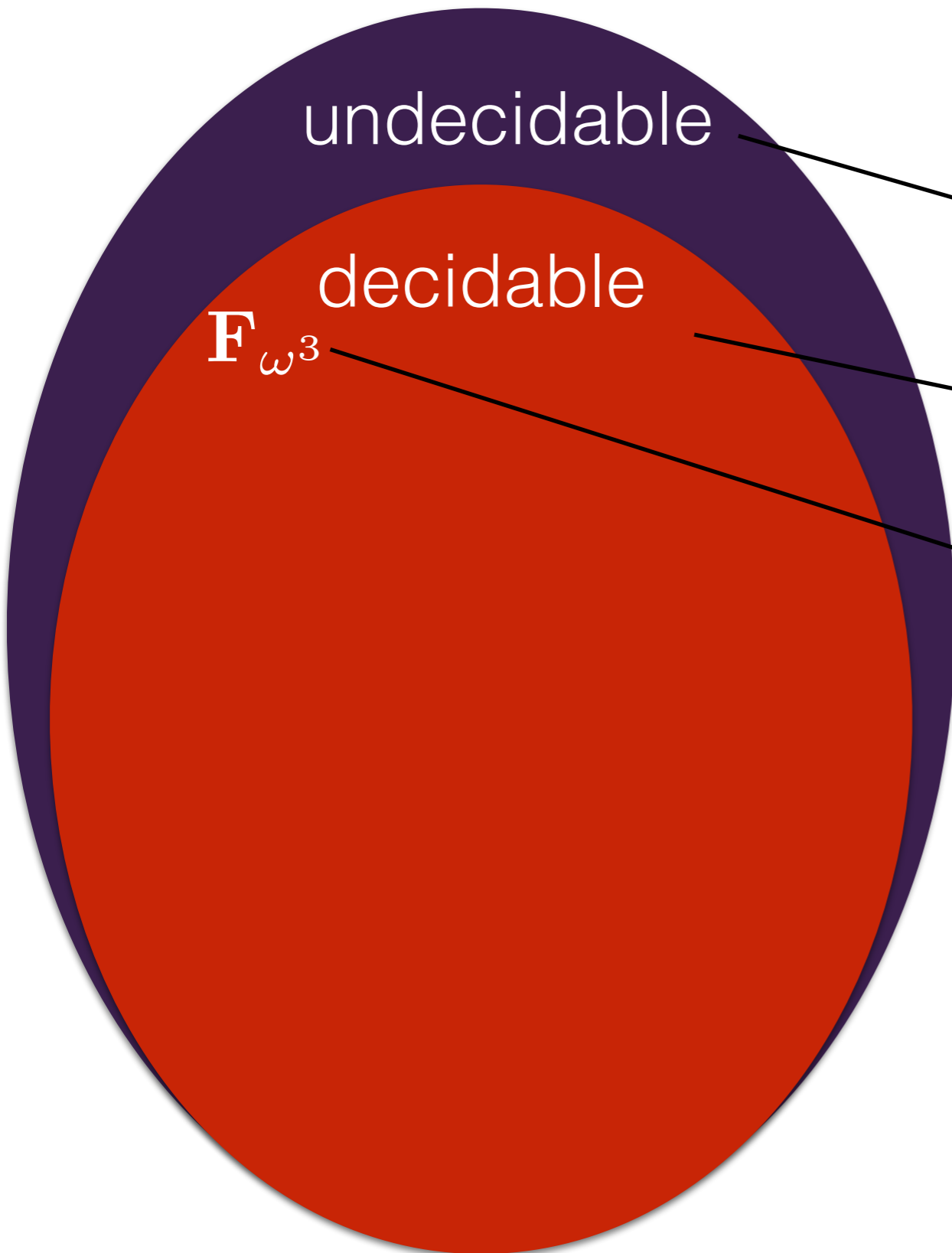
**VASS**

Mayr 1981, Kosaraju 1982

Leroux 2009-2014



# The reachability problem for VASS



**VASS with zero tests**

Minsky 1961

**VASS**

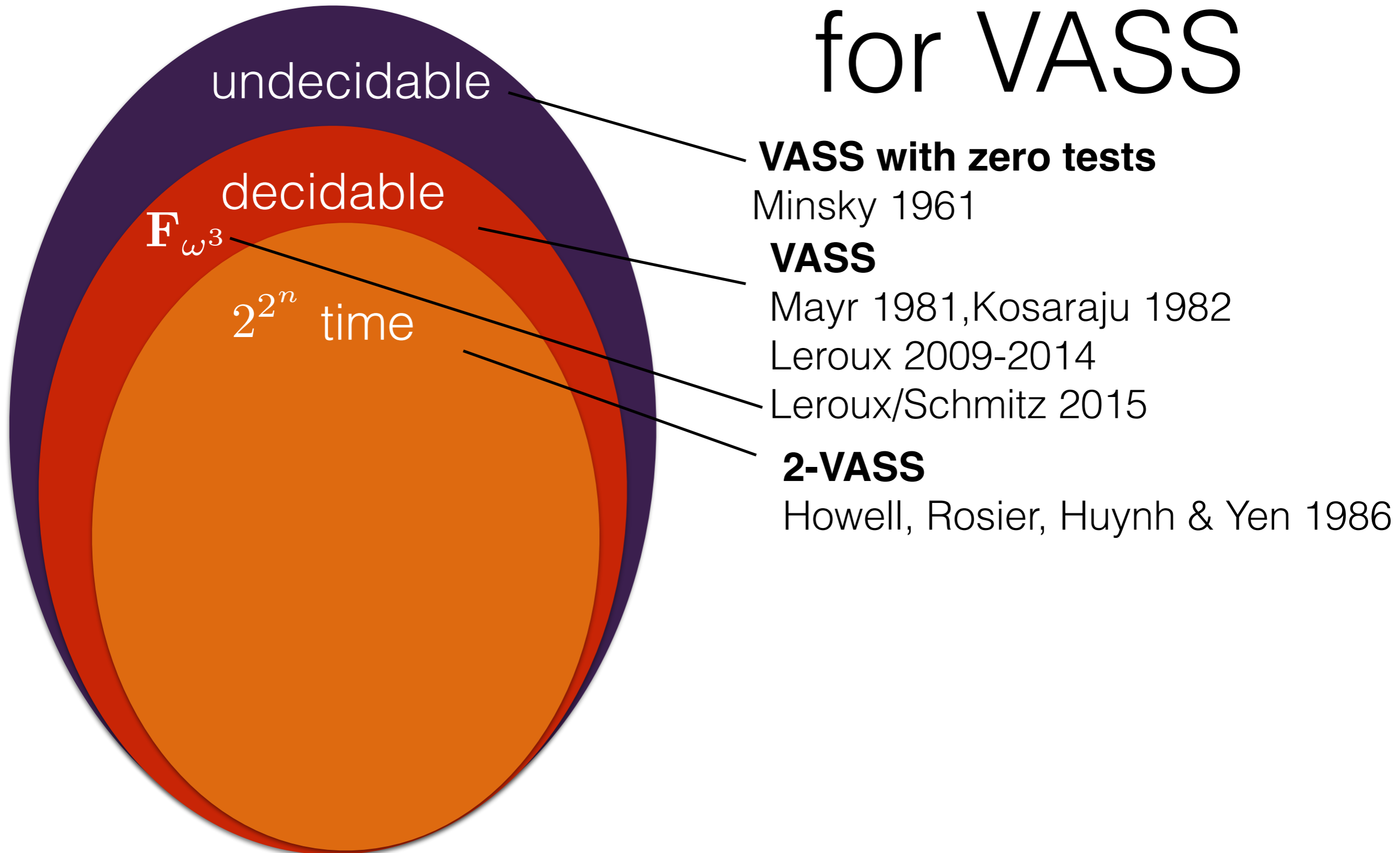
Mayr 1981, Kosaraju 1982

Leroux 2009-2014

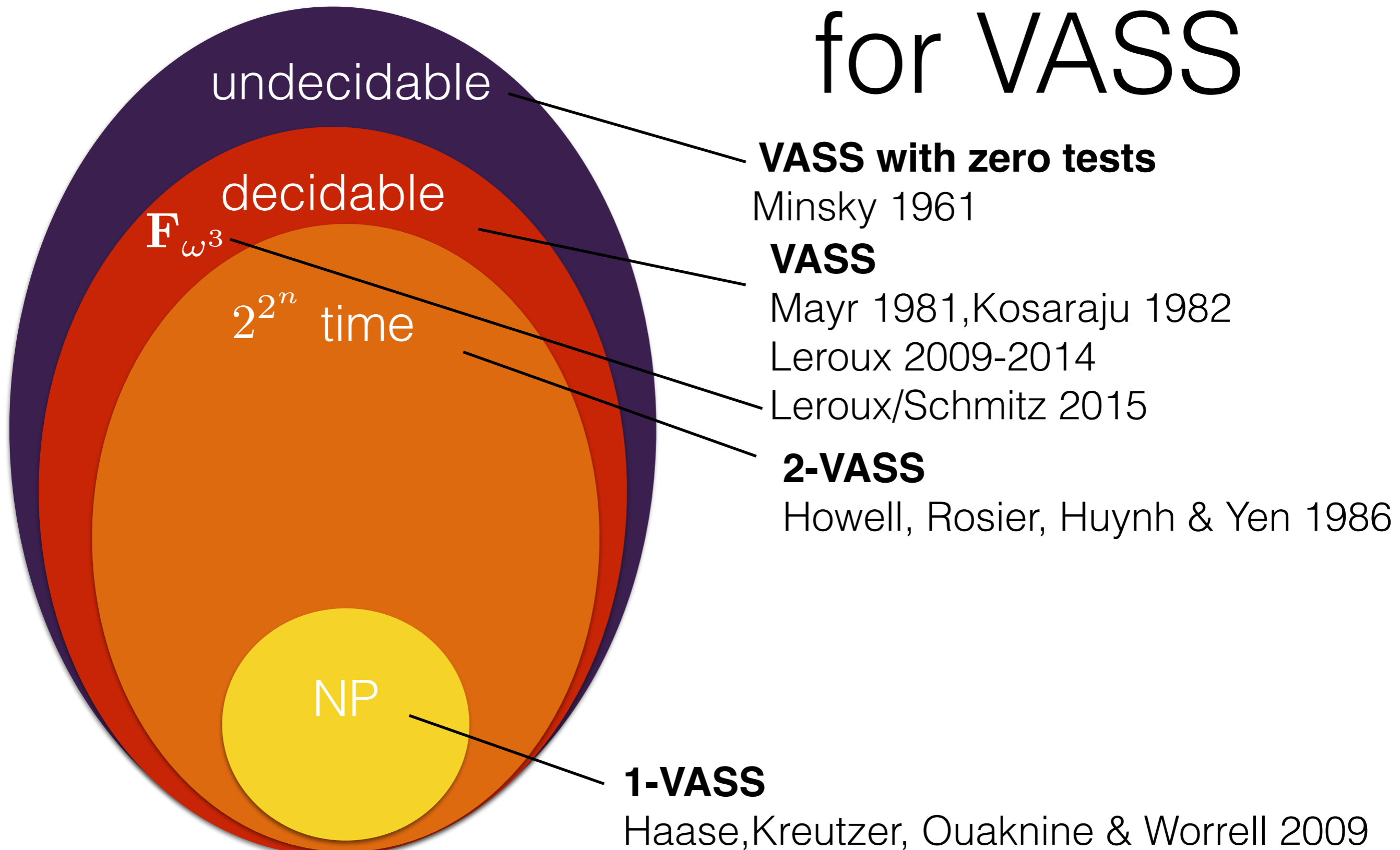
Leroux/Schmitz 2015



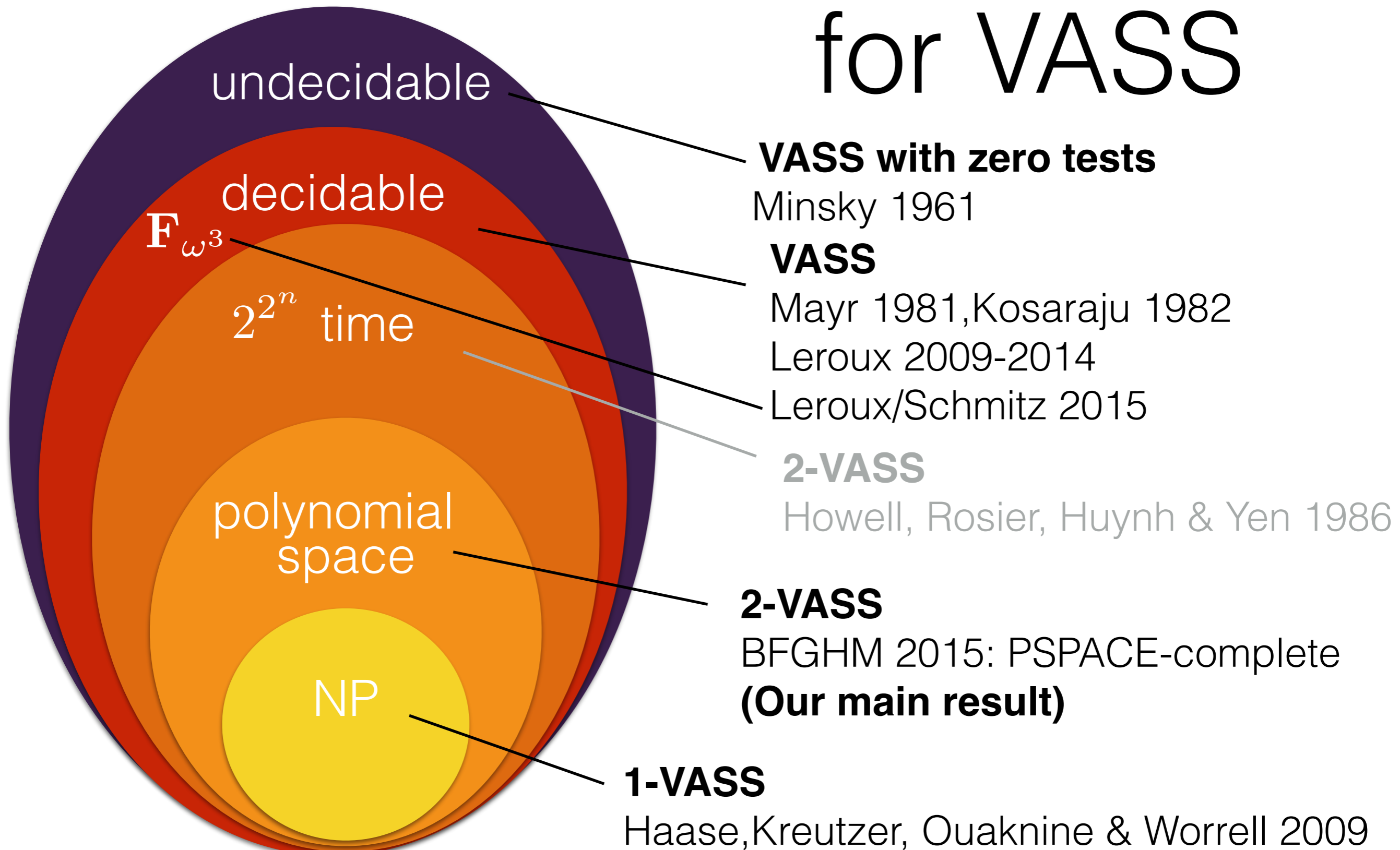
# The reachability problem for VASS



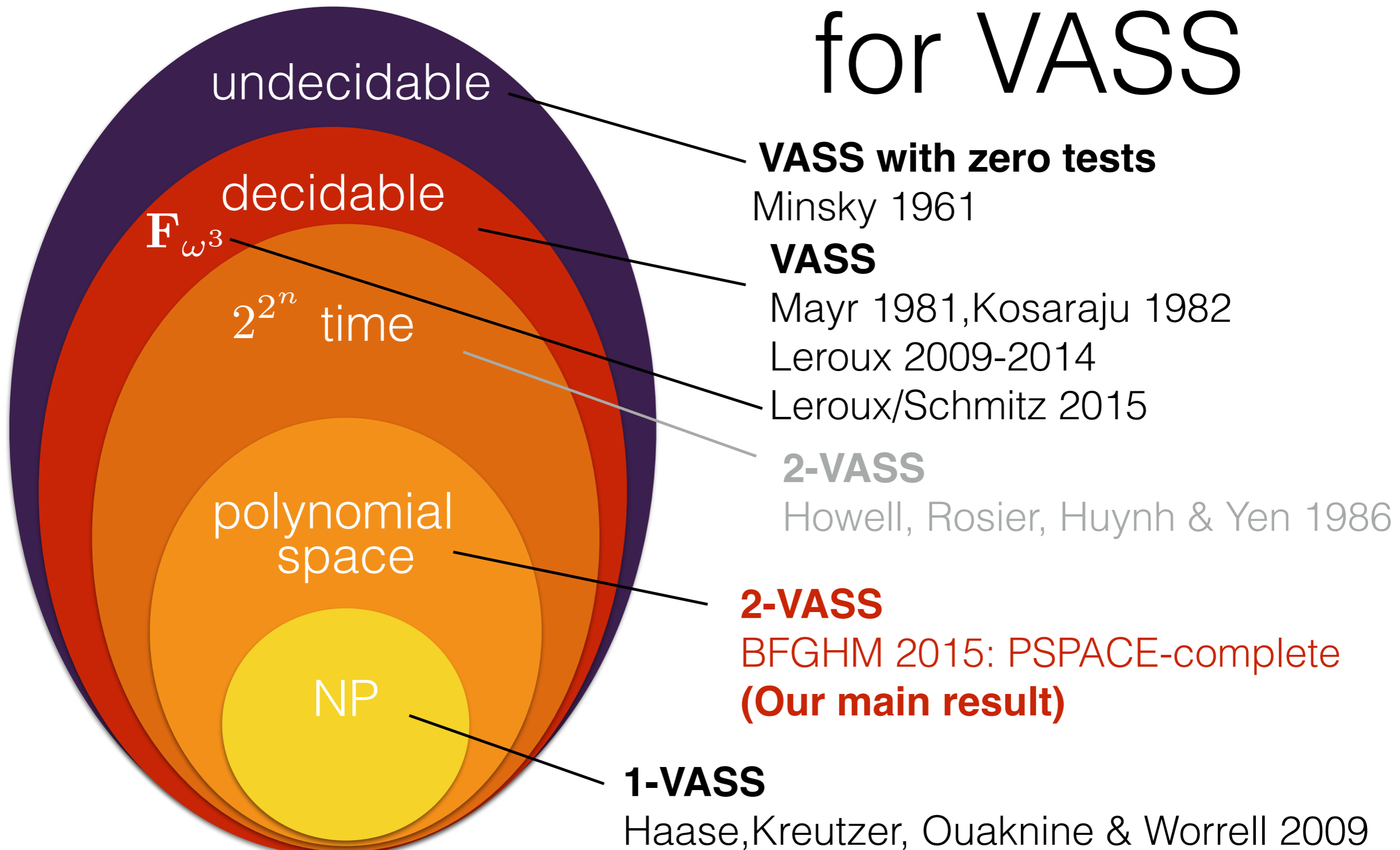
# The reachability problem for VASS



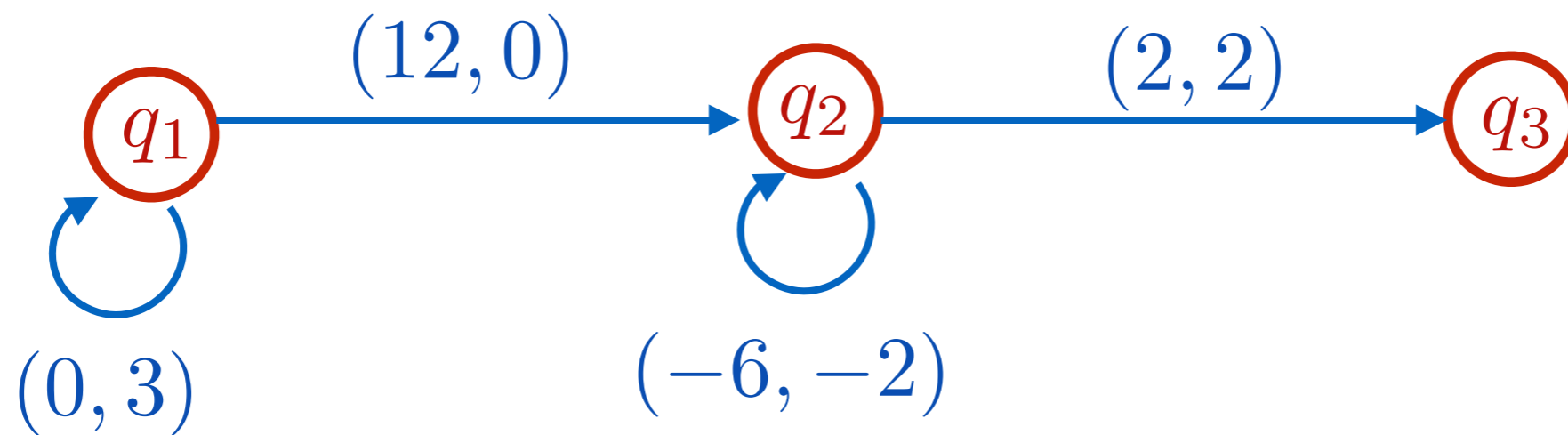
# The reachability problem for VASS



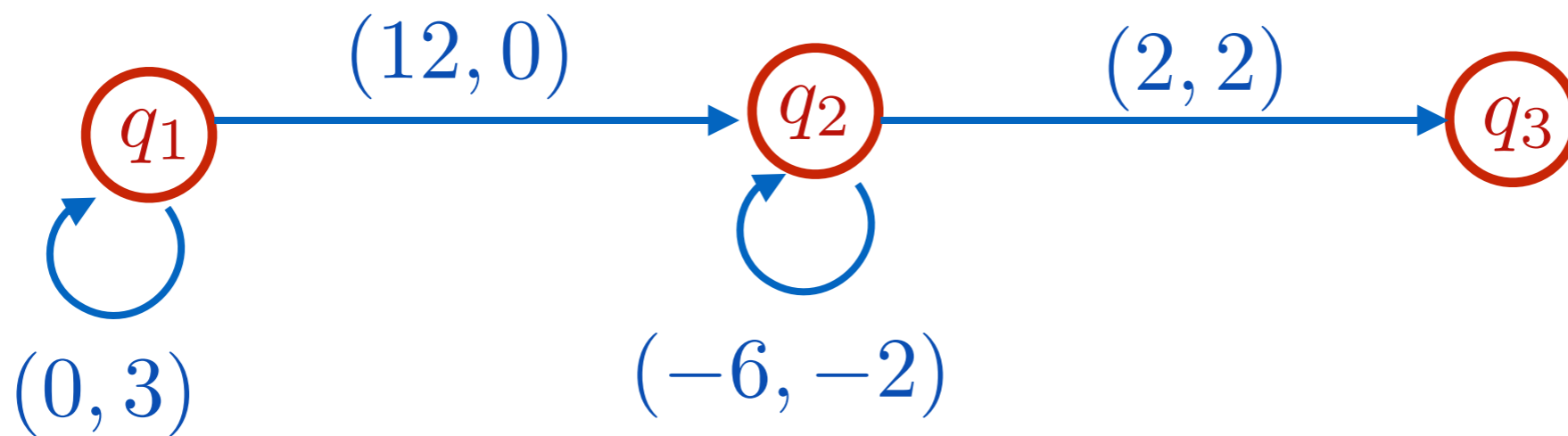
# The reachability problem for VASS



# Vector addition systems and Presburger Arithmetic



# Vector addition systems and Presburger Arithmetic

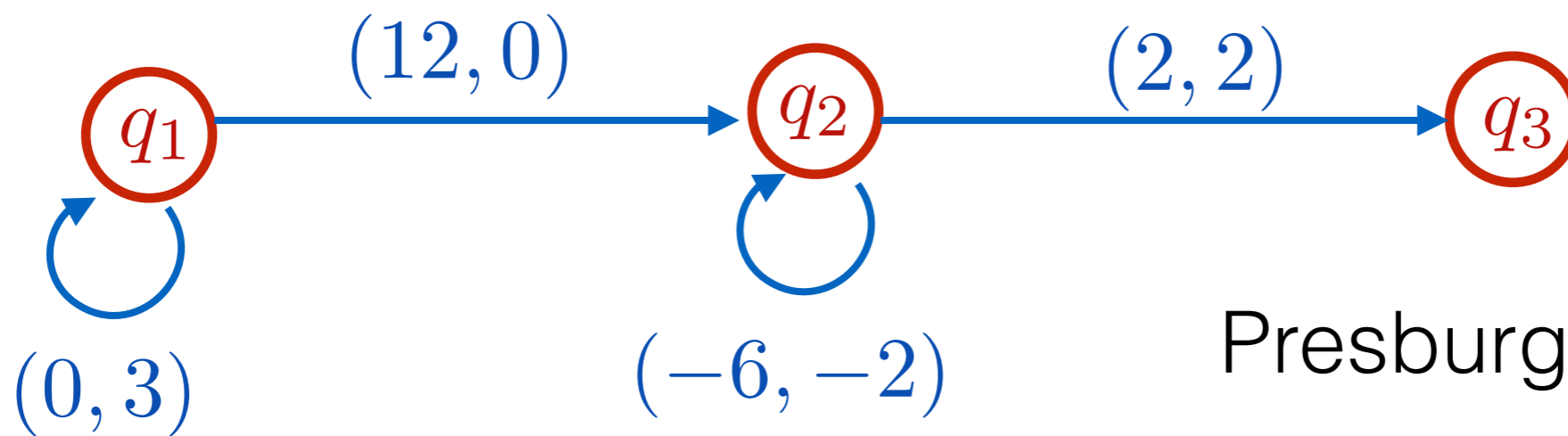


$q_1(u, v)$  can reach  $q_3(x, y)$

if, and only if,

$$(u, v, x, y) \models \exists i, j : (x - 2 = u + 12 - 6j) \wedge \\ (y - 2 = v + 3 \cdot i - 2j)$$

# Vector addition systems and Presburger Arithmetic



Presburger Arithmetic  
 $= \text{Th}(\mathbb{N}, <, +)$

$q_1(u, v)$  can reach  $q_3(x, y)$

if, and only if,

$$(u, v, x, y) \models \exists i, j : (x - 2 = u + 12 - 6j) \wedge (y - 2 = v + 3 \cdot i - 2j)$$

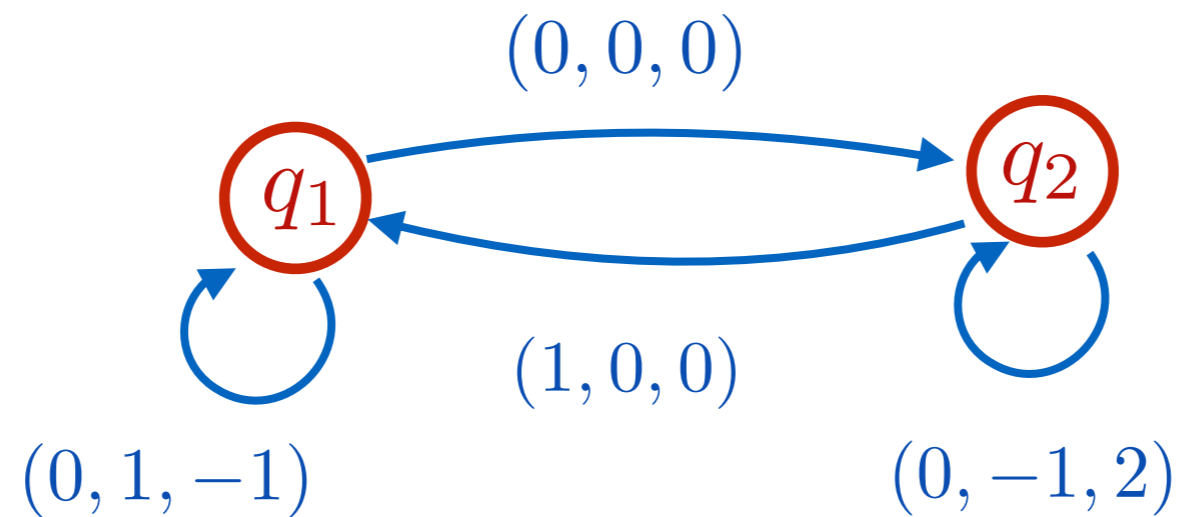


M. Presburger

# Presburger reachability

**Example.** (without proof)

The reachability relation of the following 3-VASS is not definable in Presburger Arithmetic.

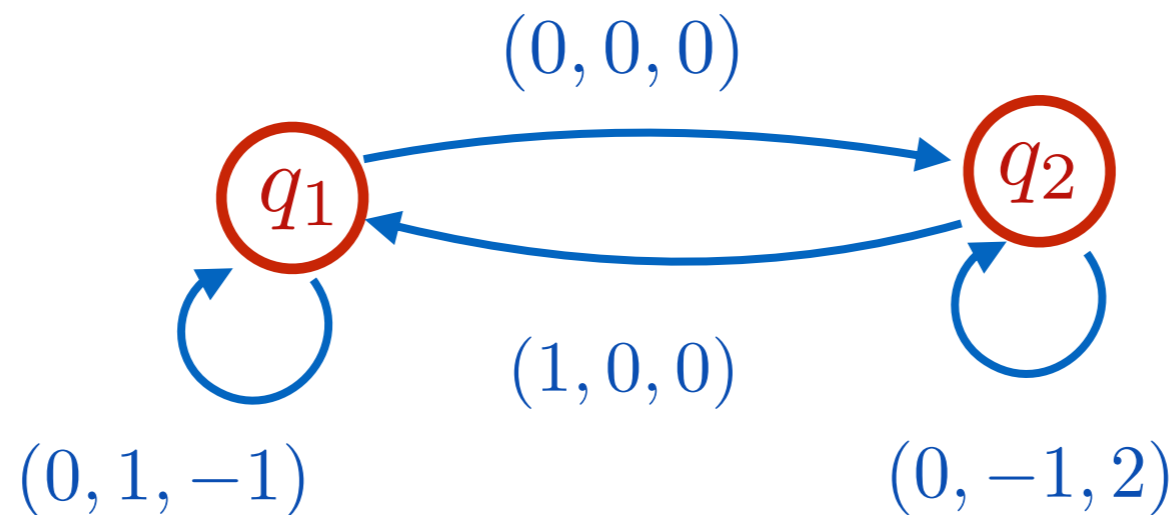




# Presburger reachability

**Example.** (without proof)

The reachability relation of the following 3-VASS is not definable in Presburger Arithmetic.



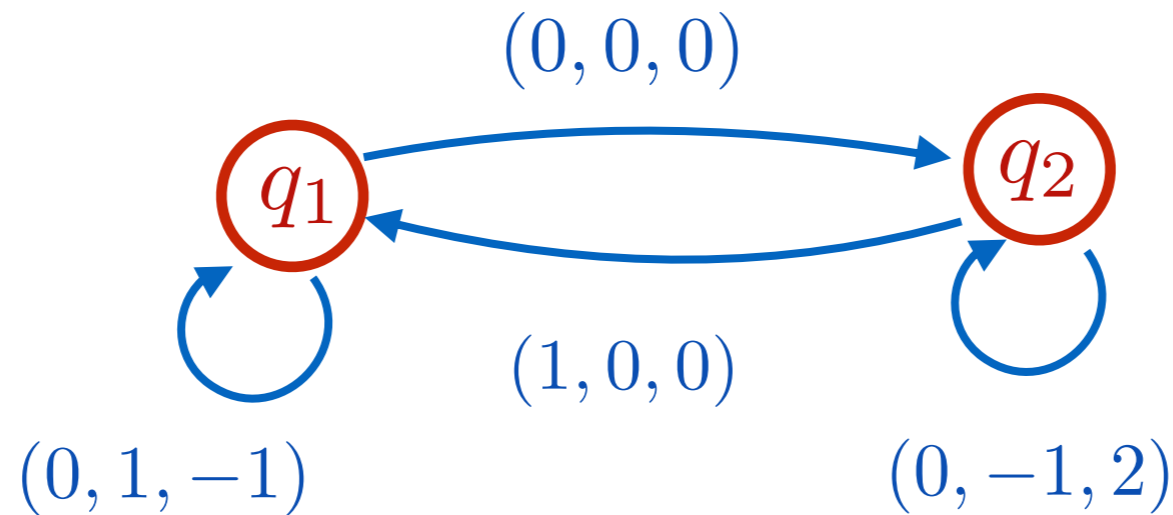
**Proposition.**

The reachability relation of a VASS  $V = (Q, T)$  is Presburger definable if and only if it can be described by a **bounded language** over  $T$ .

# Presburger reachability

**Example.** (without proof)

The reachability relation of the following 3-VASS is not definable in Presburger Arithmetic.



**Proposition.**

The reachability relation of a VASS  $V = (Q, T)$  is Presburger definable if and only if it can be described by a **bounded language** over  $T$ .

$\equiv$  **finite unions of**  $\alpha_0 \beta_1^* \alpha_1 \cdots \beta_k^* \alpha_k$

# Overview

Some background

Vector addition systems / Presburger reachability

2VASS reachability: Our main result

# Overview

Some background

Vector addition systems / Presburger reachability

**2VASS reachability: Our main result**

# 2-VASS history

**Theorem.** (Hopcroft & Pansiot 1979)

Given a 2-VASS and an initial configuration one can effectively a Presburger Arithmetic presentation of the reachable configurations.

# 2-VASS history

**Theorem.** (Hopcroft & Pansiot 1979)

Given a 2-VASS and an initial configuration one can effectively a Presburger Arithmetic presentation of the reachable configurations.

**Theorem.** (Howell, Rosier, Huynh & Yen 1986)

Reachability in 2-VASS can be decided in doubly-exponential time.

# 2-VASS history

**Theorem.** (Hopcroft & Pansiot 1979)

Given a 2-VASS and an initial configuration one can effectively a Presburger Arithmetic presentation of the reachable configurations.

**Theorem.** (Howell, Rosier, Huynh & Yen 1986)

Reachability in 2-VASS can be decided in doubly-exponential time.

**Theorem.** (Leroux & Sutre 2004)

The reachability relation of 2-VASS is effectively Presburger definable.

# 2-VASS history

**Theorem.** (Hopcroft & Pansiot 1979)

Given a 2-VASS and an initial configuration one can effectively a Presburger Arithmetic presentation of the reachable configurations.

**Theorem.** (Howell, Rosier, Huynh & Yen 1986)

Reachability in 2-VASS can be decided in doubly-exponential time.

**Theorem.** (Leroux & Sutre 2004)

The reachability relation of 2-VASS is effectively Presburger definable.

**Theorem.** (Blondin, Finkel, G, Haase & McKenzie 2015)

- The reachability relation of 2-VASS is computable in exponential space.
- The reachability problem for 2-VASS is PSPACE-complete.



# Our main result

The following is computable in exponential space:

**Given:** 2-VASS  $V = (Q, T)$  .

**Output:** Finite unions of regular expressions

$$\bigcup_i \alpha_0 \beta_1^* \alpha_1 \cdots \beta_k^* \alpha_k$$

such that

$$p(u_1, u_2) \longrightarrow^* q(v_1, v_2)$$

# Our main result

The following is computable in exponential space:

**Given:** 2-VASS  $V = (Q, T)$  .

**Output:** Finite unions of regular expressions

$$\bigcup_i \alpha_0 \beta_1^* \alpha_1 \cdots \beta_k^* \alpha_k$$

such that

$$p(u_1, u_2) \longrightarrow^* q(v_1, v_2)$$

if, and only, if

$$\exists e_1, \dots, e_k \in \mathbb{N} : p(u_1, u_2) \xrightarrow{\alpha_0 \beta_1^{e_1} \alpha_1 \cdots \beta_k^{e_k} \alpha_k} q(v_1, v_2)$$

# Our main result

The following is computable in exponential space:

**Given:** 2-VASS  $V = (Q, T)$  .

**Output:** Finite unions of regular expressions

$$\bigcup_i \alpha_0 \beta_1^* \alpha_1 \cdots \beta_k^* \alpha_k$$

Length at most  $(|Q| + |T| + \|T\|)^{O(1)}$

such that

$$p(u_1, u_2) \longrightarrow^* q(v_1, v_2)$$

if, and only, if

$$\exists e_1, \dots, e_k \in \mathbb{N} : p(u_1, u_2) \xrightarrow{\alpha_0 \beta_1^{e_1} \alpha_1 \cdots \beta_k^{e_k} \alpha_k} q(v_1, v_2)$$

# Our main result

The following is computable in exponential space:

**Given:** 2-VASS  $V = (Q, T)$  .

**Output:** Finite unions of regular expressions

$$\bigcup_i \alpha_0 \beta_1^* \alpha_1 \cdots \beta_k^* \alpha_k$$

Length at most  
 $(|Q| + |T| + \|T\|)^{O(1)}$

such that

$$p(u_1, u_2) \longrightarrow^* q(v_1, v_2)$$

if, and only, if

$$\exists e_1, \dots, e_k \in \mathbb{N} : p(u_1, u_2) \xrightarrow{\alpha_0 \beta_1^{e_1} \alpha_1 \cdots \beta_k^{e_k} \alpha_k} q(v_1, v_2)$$

$k \leq O(|Q|^2)$

# Our main result

The following is computable in exponential space:

**Given:** 2-VASS  $V = (Q, T)$  .

**Output:** Finite unions of regular expressions

$$\bigcup_i \alpha_0 \beta_1^* \alpha_1 \cdots \beta_k^* \alpha_k$$

Length at most  
 $(|Q| + |T| + \|T\|)^{O(1)}$

$k \leq O(|Q|^2)$

such that

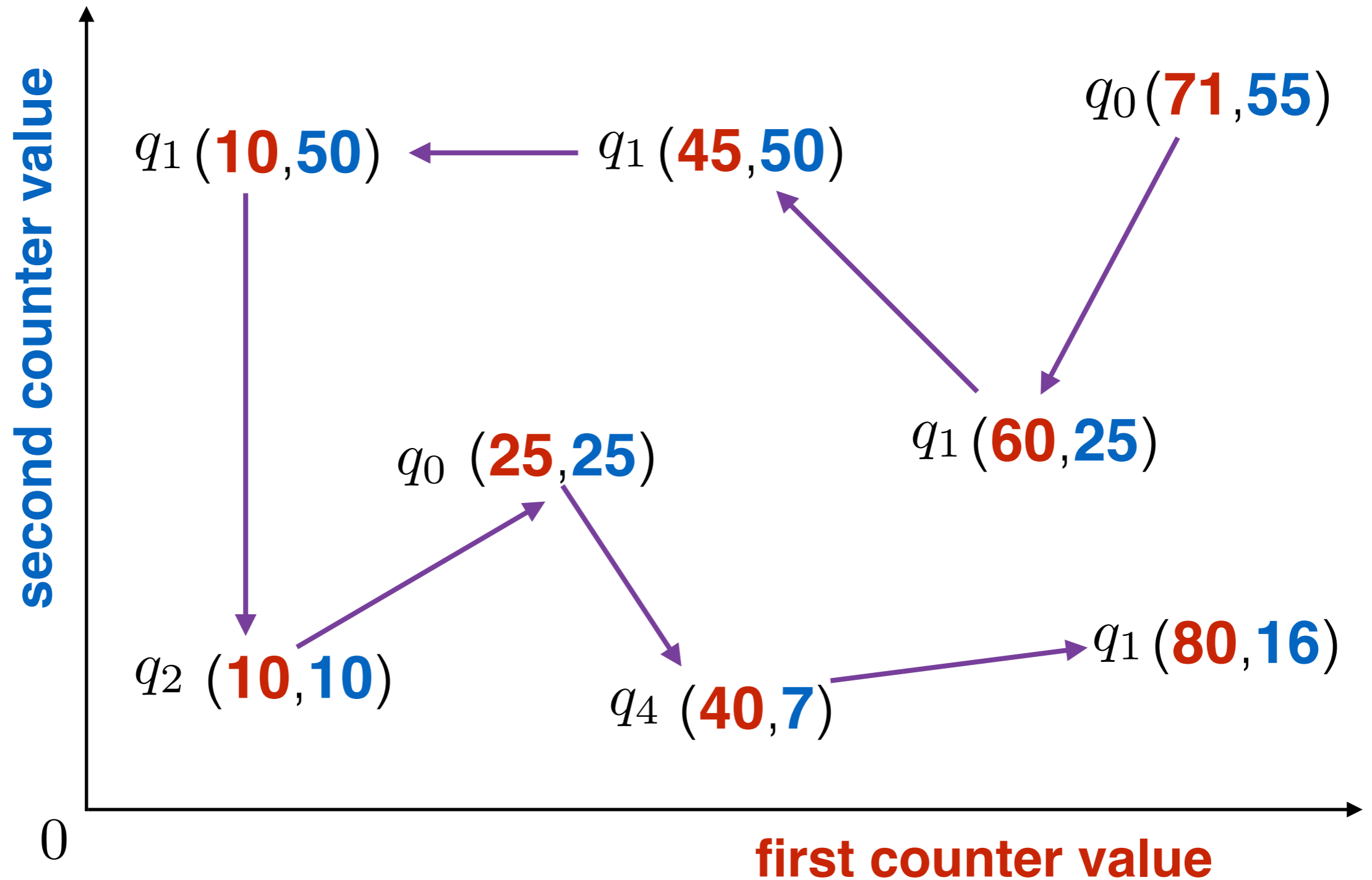
$$p(u_1, u_2) \longrightarrow^* q(v_1, v_2)$$

if, and only, if

$$\exists e_1, \dots, e_k \in \mathbb{N} : p(u_1, u_2) \xrightarrow{\alpha_0 \beta_1^{e_1} \alpha_1 \cdots \beta_k^{e_k} \alpha_k} q(v_1, v_2)$$

“Reachability is witnessed by a small bounded language”

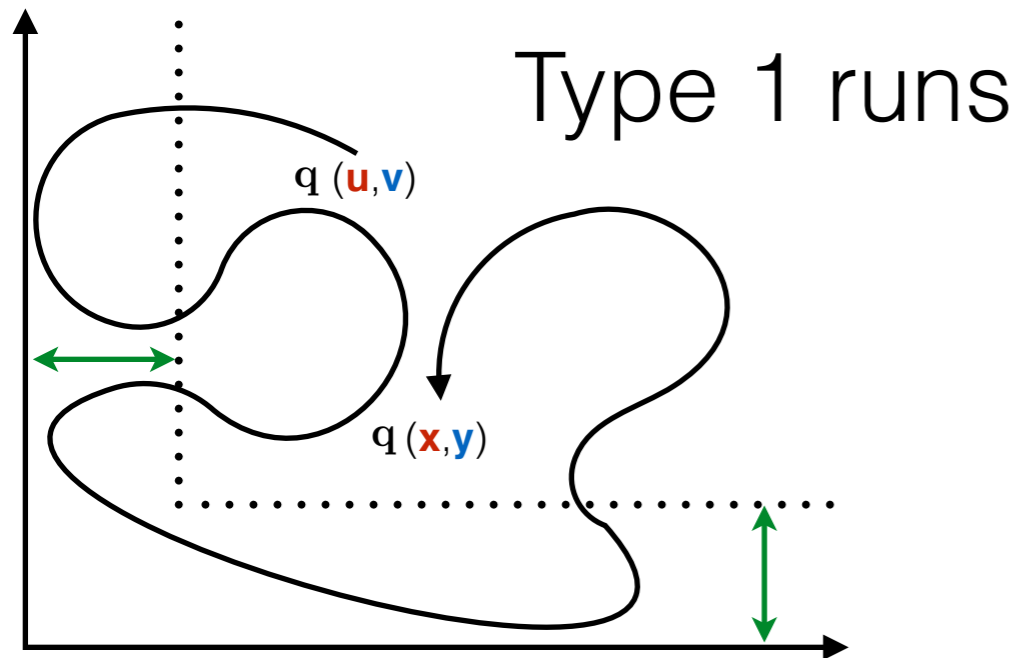
# 2-D projection of runs



# Proof strategy

following Leroux&Sutre 2004

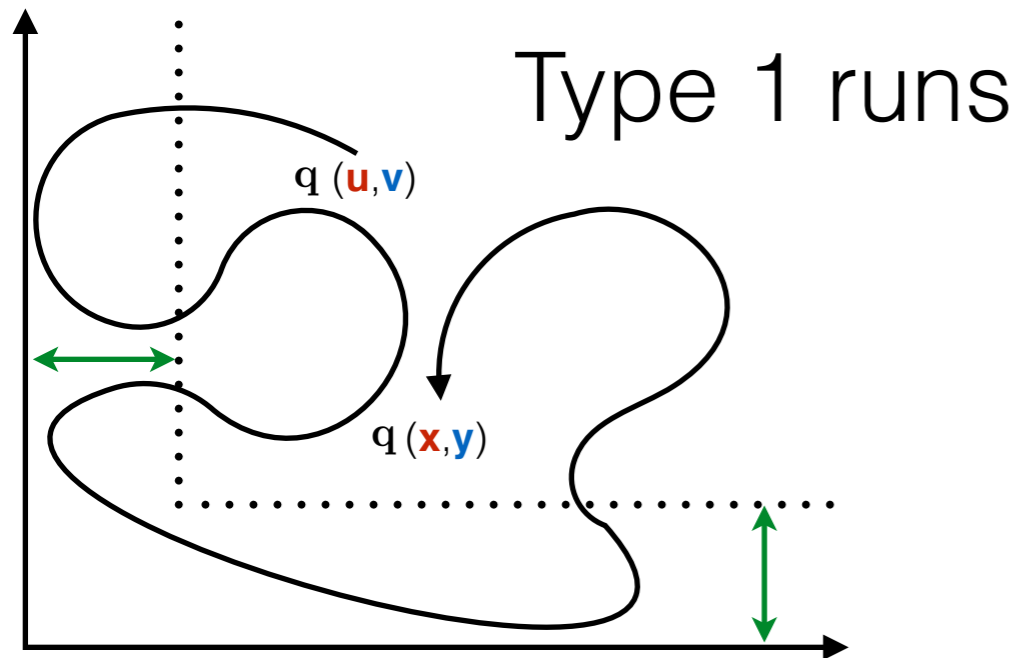
Starting and ending **sufficiently large**  
in **same** control state  $q$



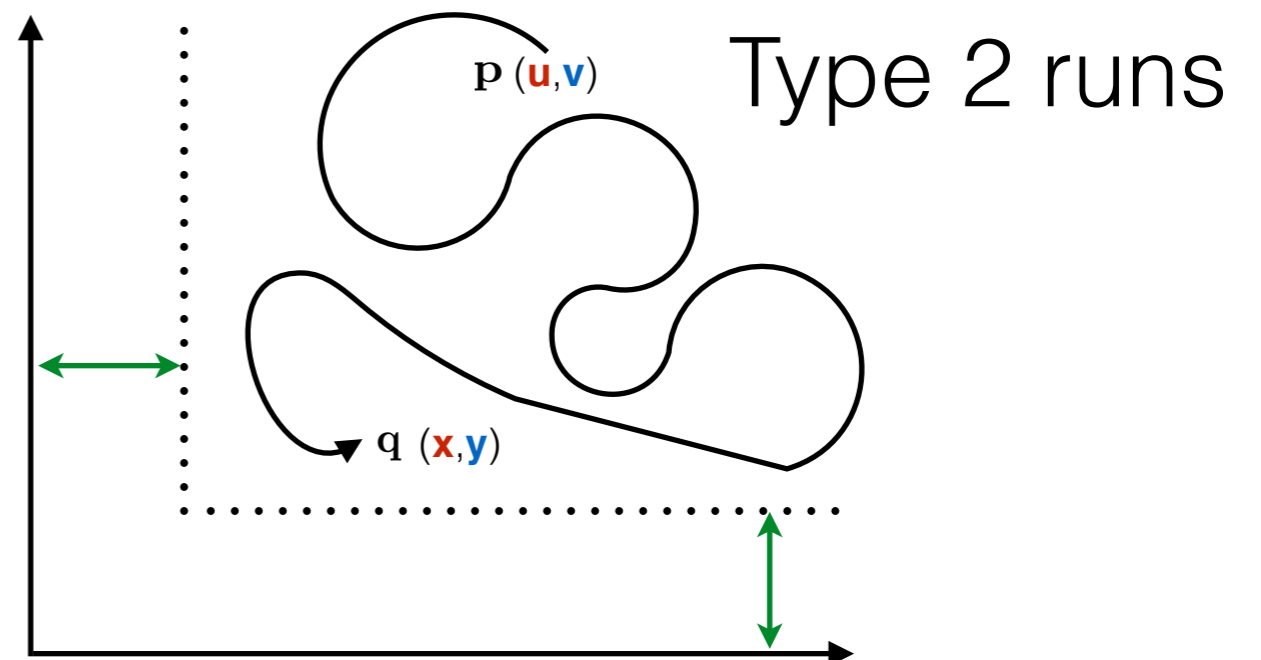
# Proof strategy

following Leroux&Sutre 2004

Starting and ending **sufficiently large**  
in **same** control state  $q$



Staying **sufficiently large all the time**

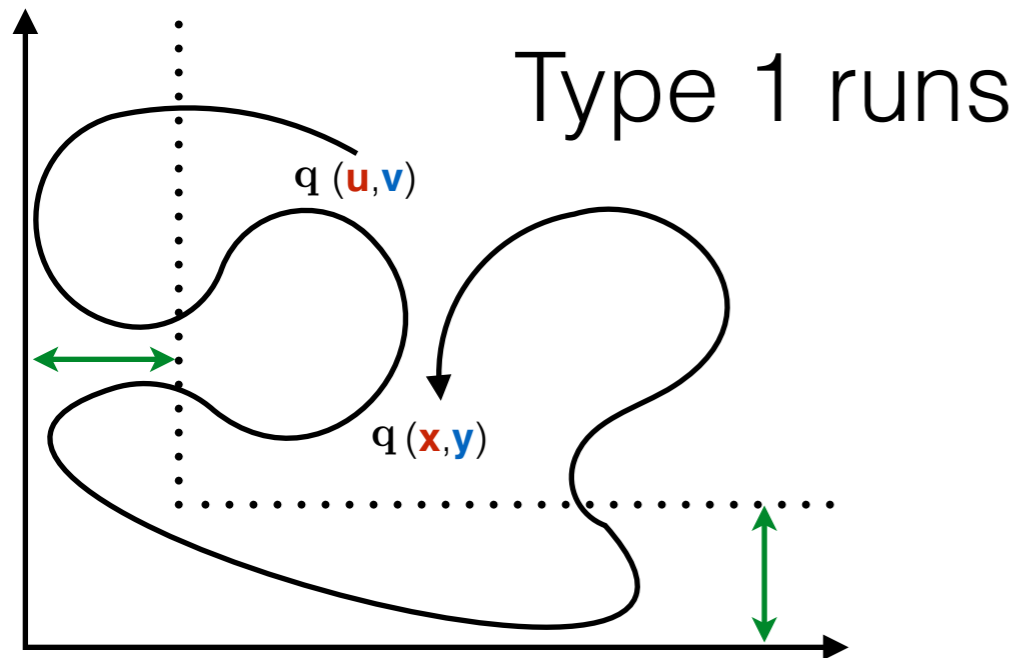




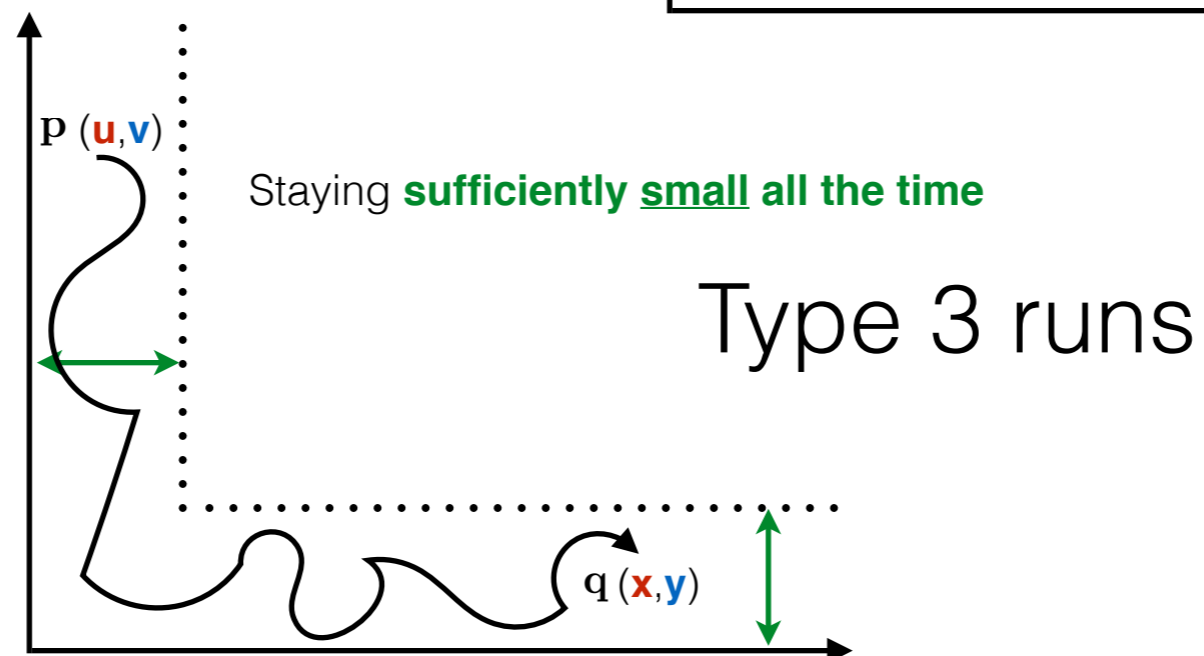
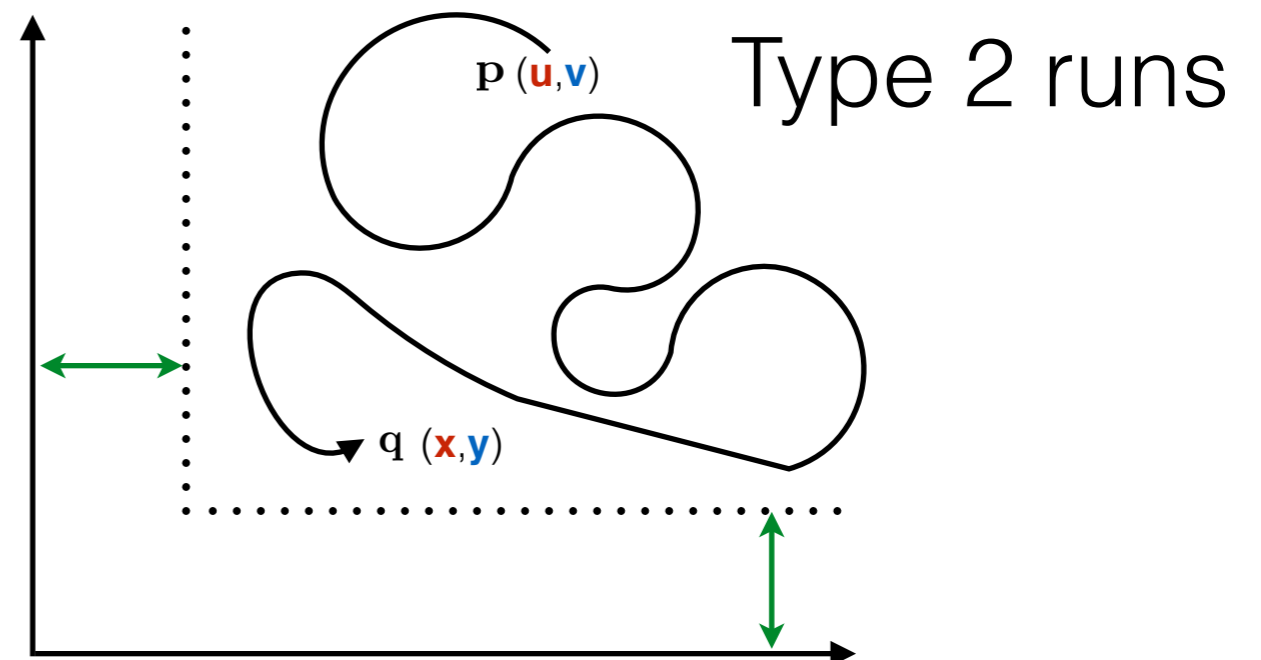
# Proof strategy

following Leroux&Sutre 2004

Starting and ending **sufficiently large**  
in **same** control state  $q$



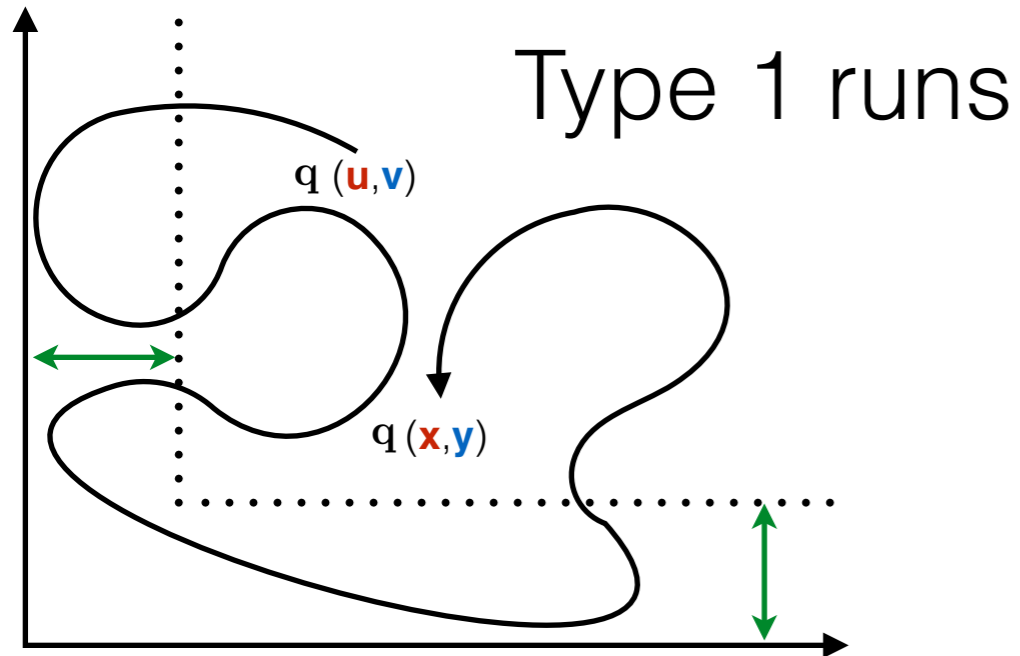
Staying **sufficiently large all the time**



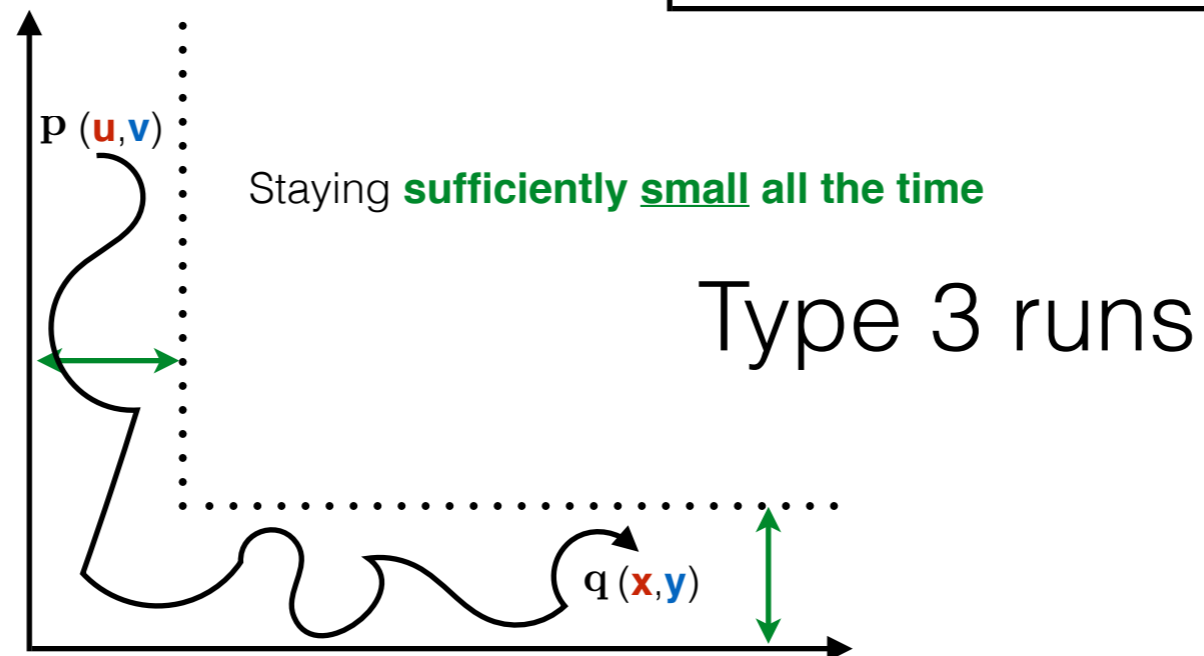
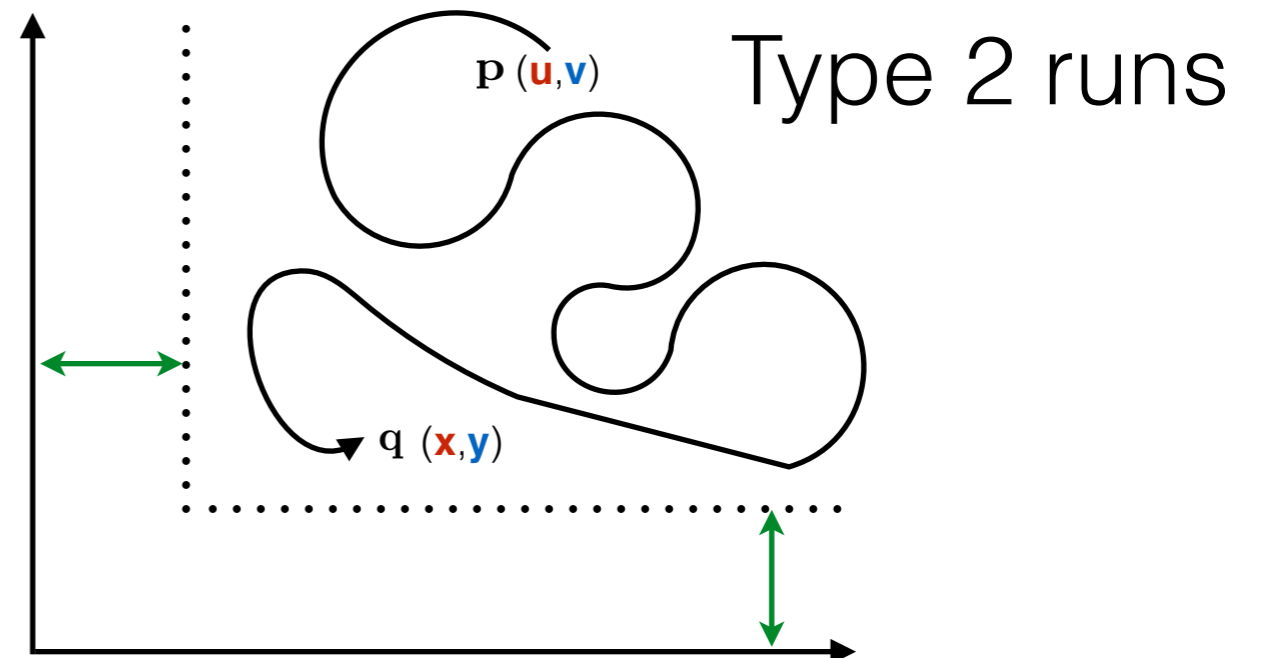
# Proof strategy

following Leroux&Sutre 2004

Starting and ending **sufficiently large**  
in **same** control state  $q$



Staying **sufficiently large all the time**



``Type 1,2,3 runs can be captured by a small bounded language``

# Our main result

The following is computable in exponential space:

**Given:** 2-VASS  $V = (Q, T)$  .

**Output:** Finite unions of regular expressions

$$\bigcup_i \alpha_0 \beta_1^* \alpha_1 \cdots \beta_k^* \alpha_k$$

Length at most  
 $(|Q| + |T| + \|T\|)^{O(1)}$

$k \leq O(|Q|^2)$

such that

$$p(u_1, u_2) \longrightarrow^* q(v_1, v_2)$$

if, and only, if

$$\exists e_1, \dots, e_k \in \mathbb{N} : p(u_1, u_2) \xrightarrow{\alpha_0 \beta_1^{e_1} \alpha_1 \cdots \beta_k^{e_k} \alpha_k} q(v_1, v_2)$$

“Reachability is witnessed by a small bounded language”

# Our main result

The following is computable in exponential space:

**Given:** 2-VASS  $V = (Q, T)$  .

**Output:** Finite unions of regular expressions

$$\bigcup_i \alpha_0 \beta_1^* \alpha_1 \cdots \beta_k^* \alpha_k$$

How to get PSPACE  
for reachability?

Length at most  
 $(|Q| + |T| + \|T\|)^{O(1)}$

$k \leq O(|Q|^2)$

such that

$$p(u_1, u_2) \longrightarrow^* q(v_1, v_2)$$

if, and only, if

$$\exists e_1, \dots, e_k \in \mathbb{N} : p(u_1, u_2) \xrightarrow{\alpha_0 \beta_1^{e_1} \alpha_1 \cdots \beta_k^{e_k} \alpha_k} q(v_1, v_2)$$

“Reachability is witnessed by a small bounded language”

# Our main result

The following is computable in exponential space:

**Given:** 2-VASS  $V = (Q, T)$  .

**Output:** Finite unions of regular expressions

$$\bigcup_i \alpha_0 \beta_1^* \alpha_1 \cdots \beta_k^* \alpha_k$$

How to get PSPACE for reachability?

Length at most  $(|Q| + |T| + \|T\|)^{O(1)}$

$k \leq O(|Q|^2)$

such that

$$p(u_1, u_2) \longrightarrow^* q(v_1, v_2)$$

if, and only, if

$$\exists e_1, \dots, e_k \in \mathbb{N} : p(u_1, u_2) \xrightarrow{\alpha_0 \beta_1^{e_1} \alpha_1 \cdots \beta_k^{e_k} \alpha_k} q(v_1, v_2)$$

bounded by  $(|Q| + |T| + \|T\|)^{O(1)}$

“Reachability is witnessed by a small bounded language”

# Outlook & Future Work



- Unary 2-VASS reachability: NL ..... NP

# Outlook & Future Work



- Unary 2-VASS reachability: **NL-complete**

**Englert, Lazic, Totzke 2016**

# Outlook & Future Work



- Unary 2-VASS reachability: **NL-complete**

**Englert, Lazic, Totzke 2016**

- Deciding semi-linearity of the reachability relation of VASS



# Outlook & Future Work



- Unary 2-VASS reachability: **NL-complete**

**Englert, Lazic, Totzke 2016**

- Deciding semi-linearity of the reachability relation of VASS
- 3-VASS

# Outlook & Future Work



- Unary 2-VASS reachability: **NL-complete**

**Englert, Lazic, Totzke 2016**

- Deciding semi-linearity of the reachability relation of VASS
- 3-VASS
- Model checking reachability logics

**PART 1:** Vector addition systems

**PART 2:** Branching vector addition systems

**PART 1:** Vector addition systems

**PART 2: Branching vector addition systems**

# Branching vector addition systems with states

A **branching vector addition system with states (d-BVASS)** is a finite automaton that consists of

- a finite set of **states**  $Q$  and

# Branching vector addition systems with states

A **branching vector addition system with states (d-BVASS)** is a finite automaton that consists of

- a finite set of **states**  $Q$  and
- a set of **transitions**  $T \subseteq Q \times \mathbb{Z}^d \times Q \cup Q^3$

# Branching vector addition systems with states

A **branching vector addition system with states (d-BVASS)**

is a finite automaton that consists of

- a finite set of **states**  $Q$  and

- a set of **transitions**  $T \subseteq \overbrace{Q \times \mathbb{Z}^d \times Q}^{\text{unary rules}} \cup Q^3$

# Branching vector addition systems with states

A **branching vector addition system with states (d-BVASS)**

is a finite automaton that consists of

- a finite set of **states**  $Q$  and

- a set of **transitions**  $T \subseteq Q \times \mathbb{Z}^d \times Q \cup Q^3$

**unary rules**

**binary/branching rules**



# Branching vector addition systems with states

A **branching vector addition system with states (d-BVASS)**

is a finite automaton that consists of

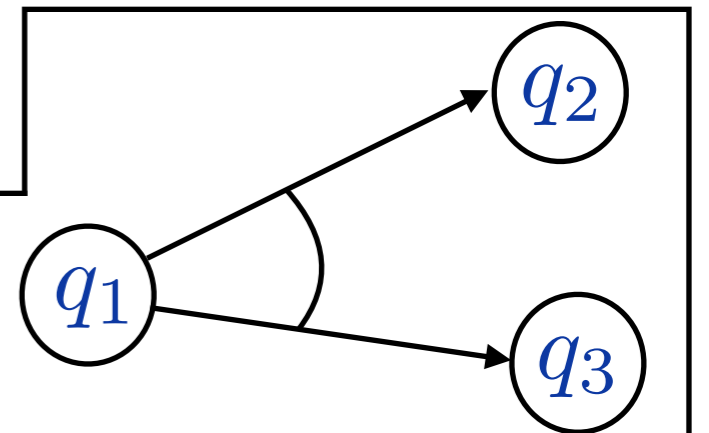
- a finite set of **states**  $Q$  and

- a set of **transitions**  $T \subseteq Q \times \mathbb{Z}^d \times Q \cup Q^3$

**unary rules**

**binary/branching rules**

branching rule  $(q_1, q_2, q_3) \in T$  displayed as



# Branching vector addition systems with states

A **branching vector addition system with states (d-BVASS)**

is a finite automaton that consists of

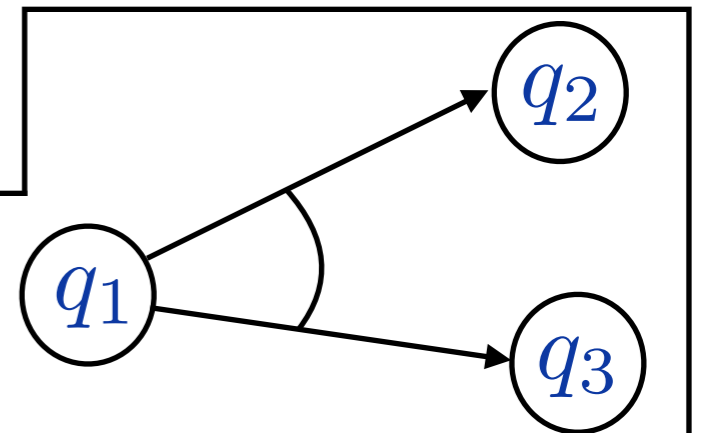
- a finite set of **states**  $Q$  and

- a set of **transitions**  $T \subseteq Q \times \mathbb{Z}^d \times Q \cup Q^3$

**unary rules**

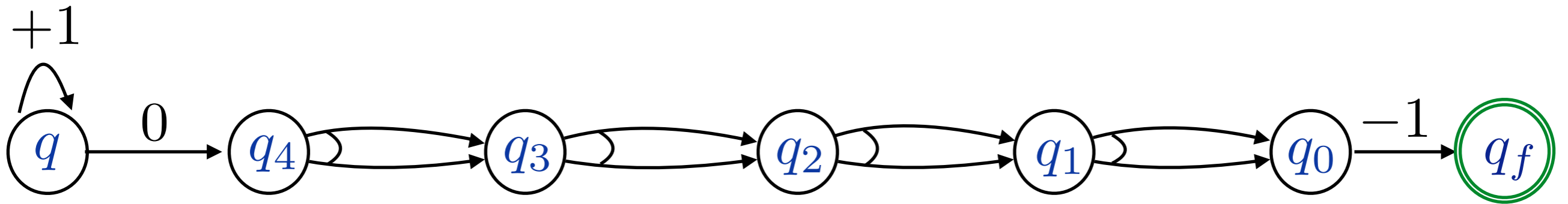
**binary/branching rules**

branching rule  $(q_1, q_2, q_3) \in T$  displayed as

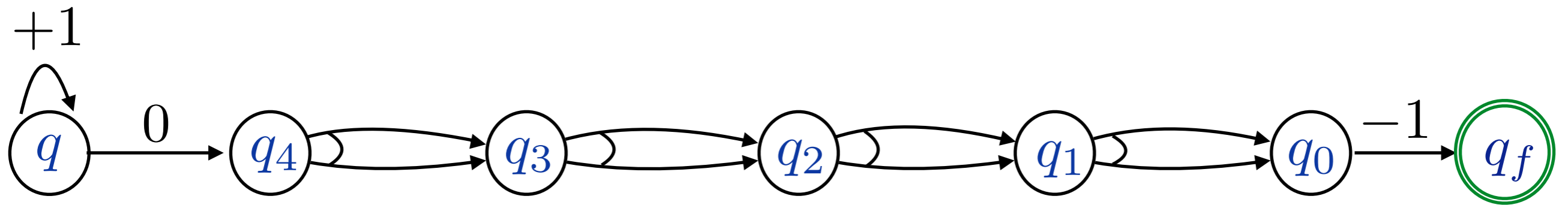


- a set of **final states**  $F \subseteq Q$

# Example for a 1-BVASS



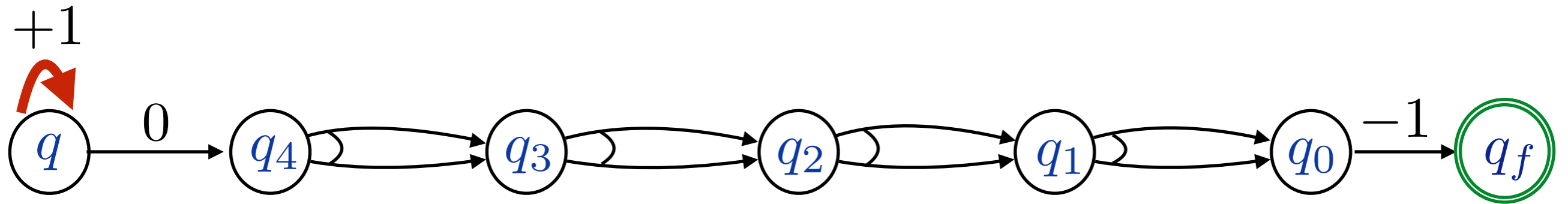
# Example for a 1-BVASS



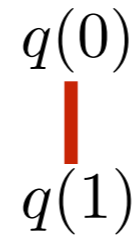
Reachability for a configuration is witnessed by a tree

$q(0)$

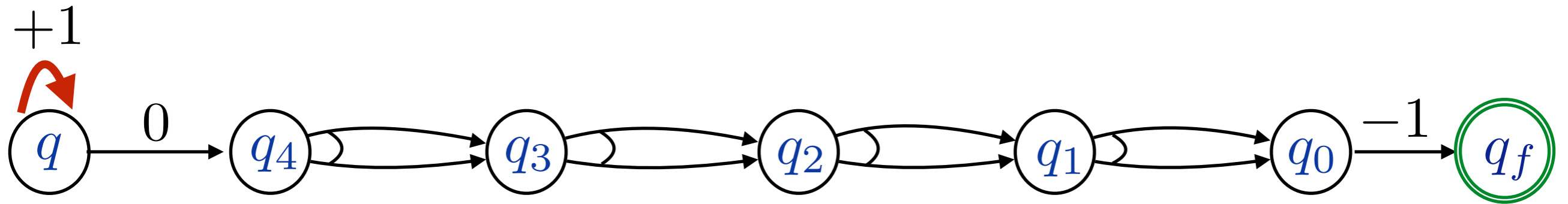
# Example for a 1-BVASS



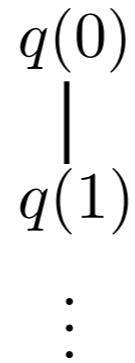
Reachability for a configuration is witnessed by a tree



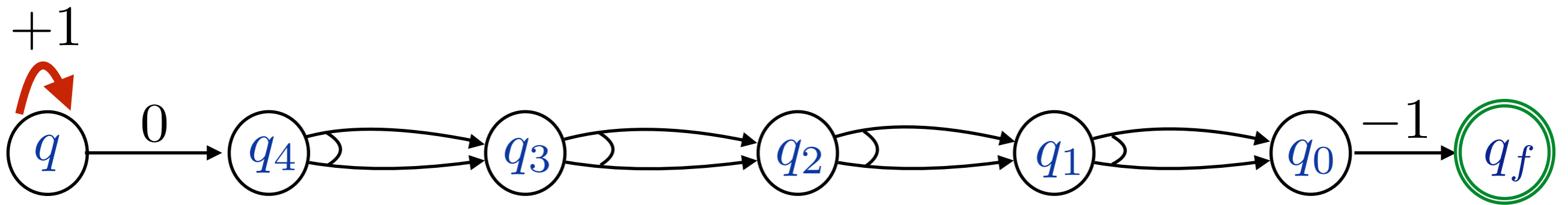
# Example for a 1-BVASS



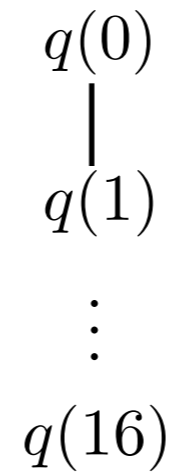
Reachability for a configuration is witnessed by a tree



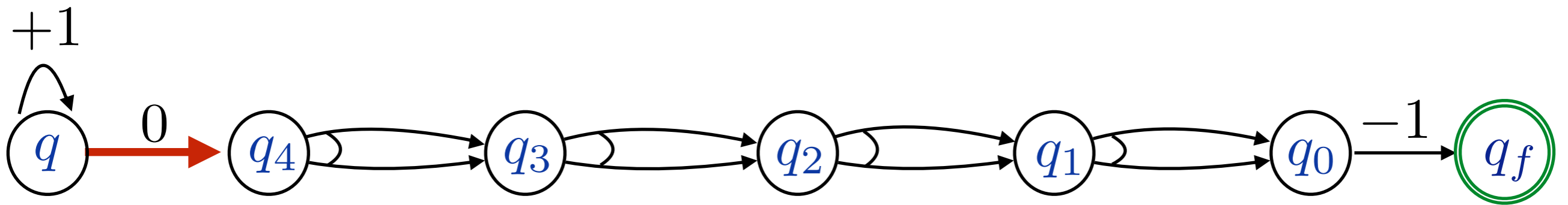
# Example for a 1-BVASS



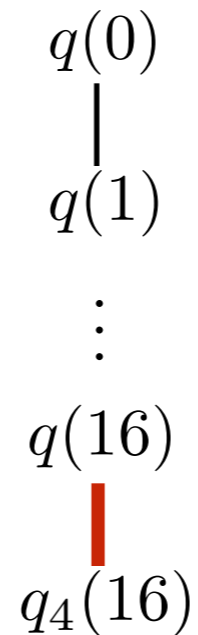
Reachability for a configuration is witnessed by a tree



# Example for a 1-BVASS

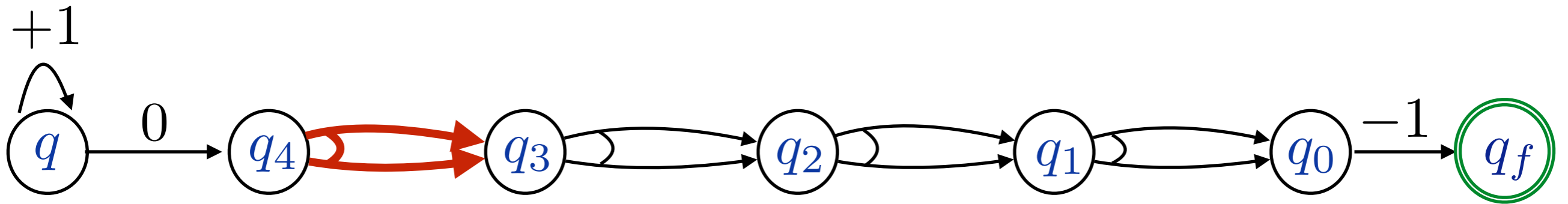


Reachability for a configuration is witnessed by a tree

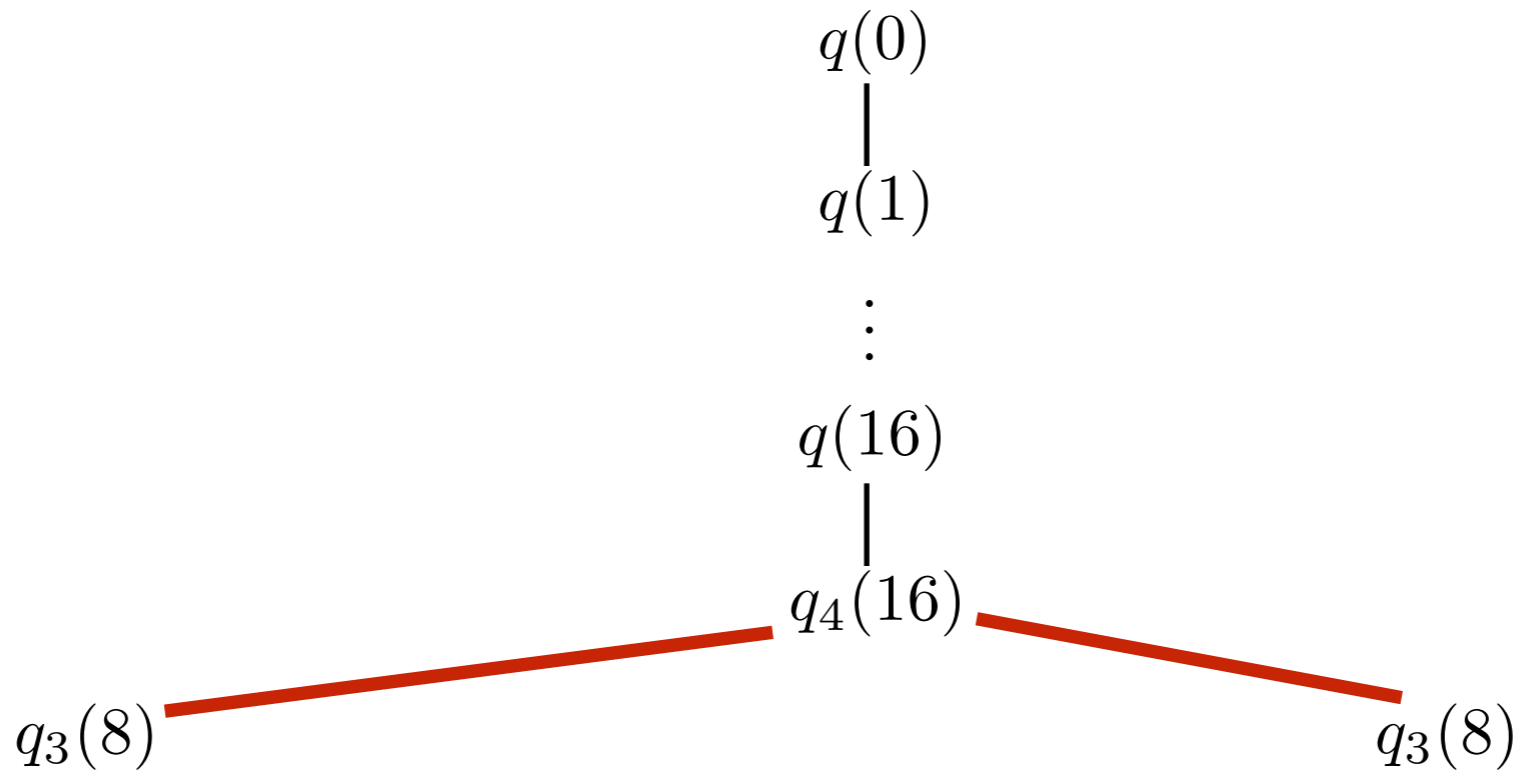




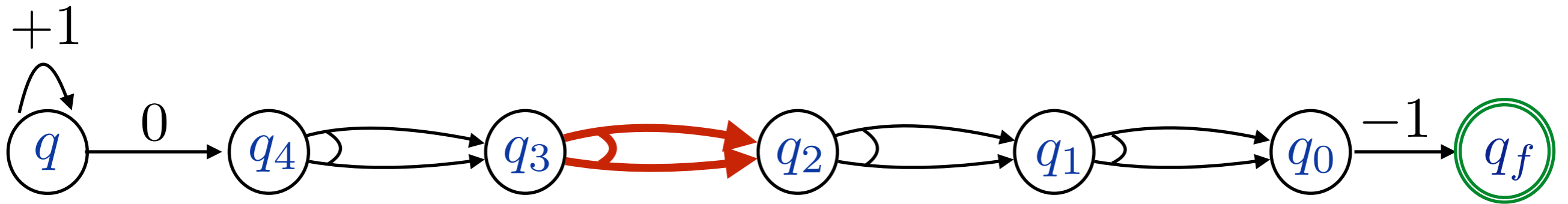
# Example for a 1-BVASS



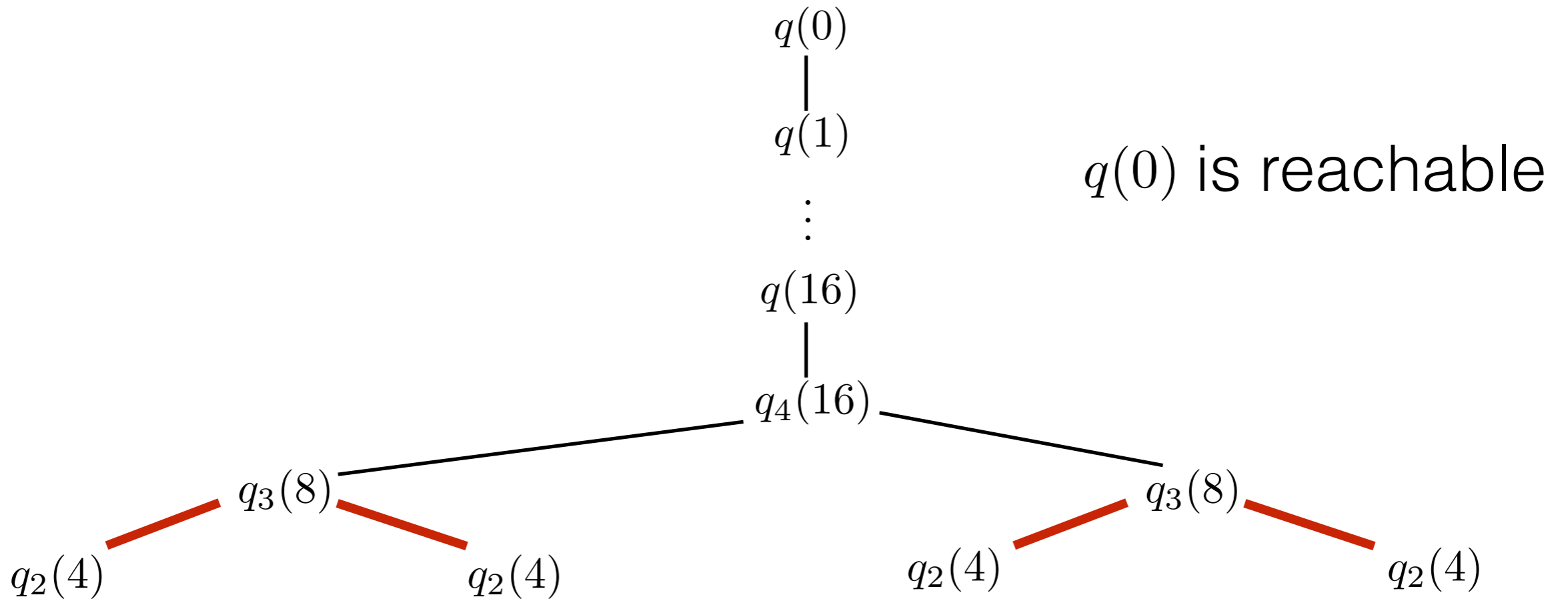
Reachability for a configuration is witnessed by a tree



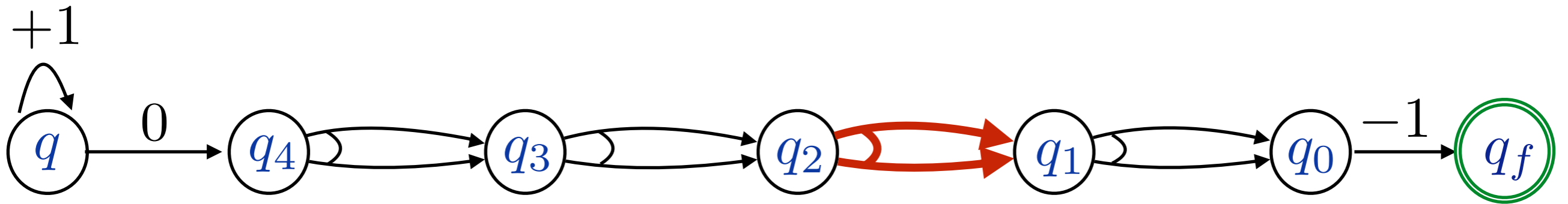
# Example for a 1-BVASS



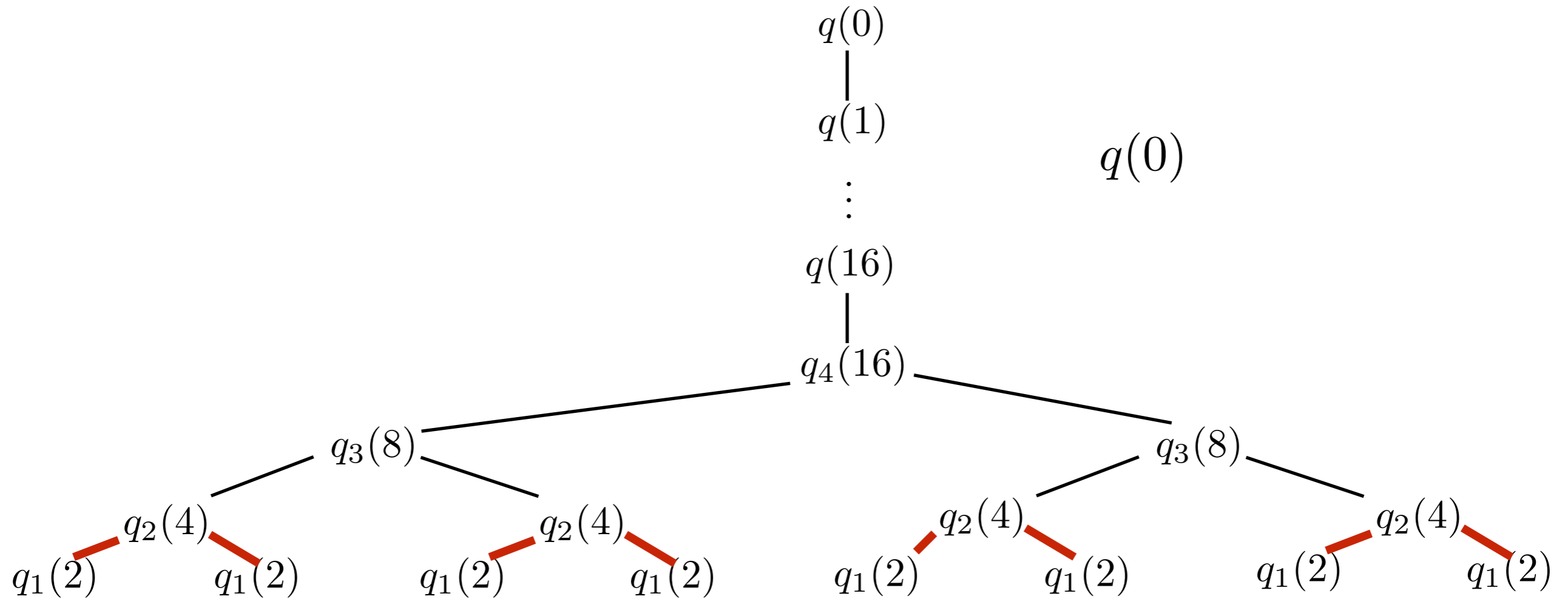
Reachability for a configuration is witnessed by a tree



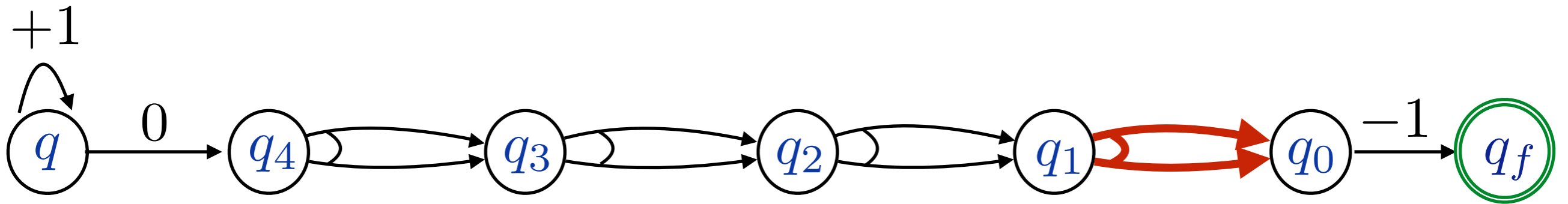
# Example for a 1-BVASS



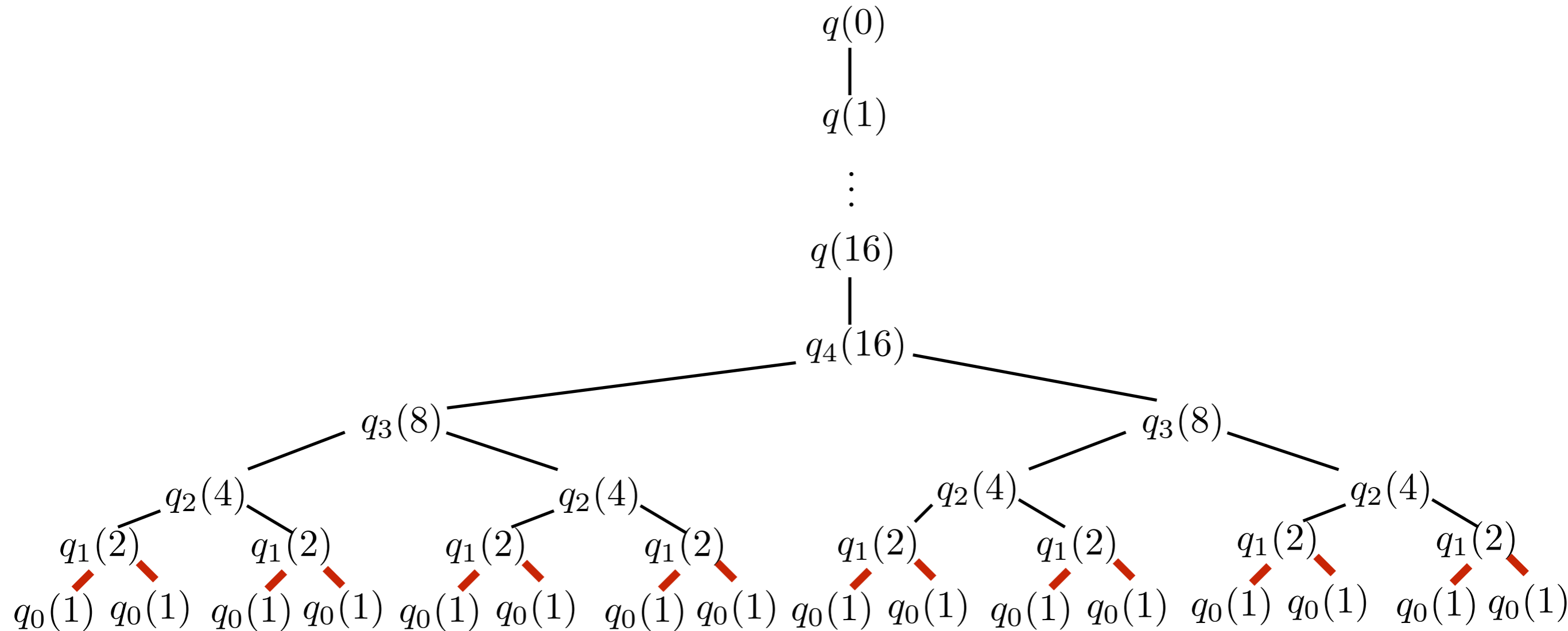
Reachability for a configuration is witnessed by a tree



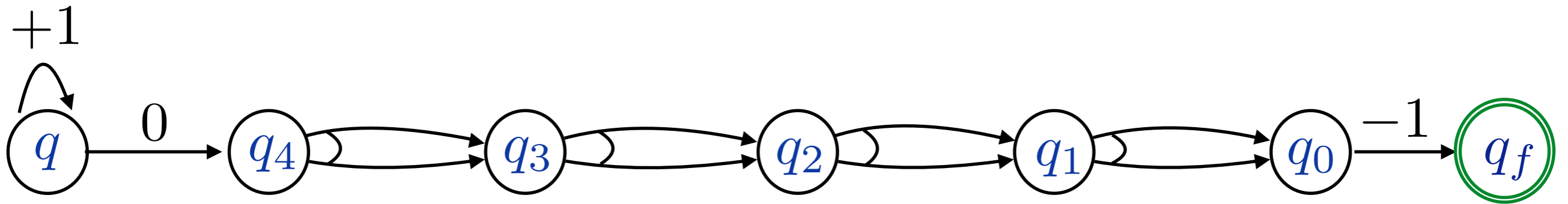
# Example for a 1-BVASS



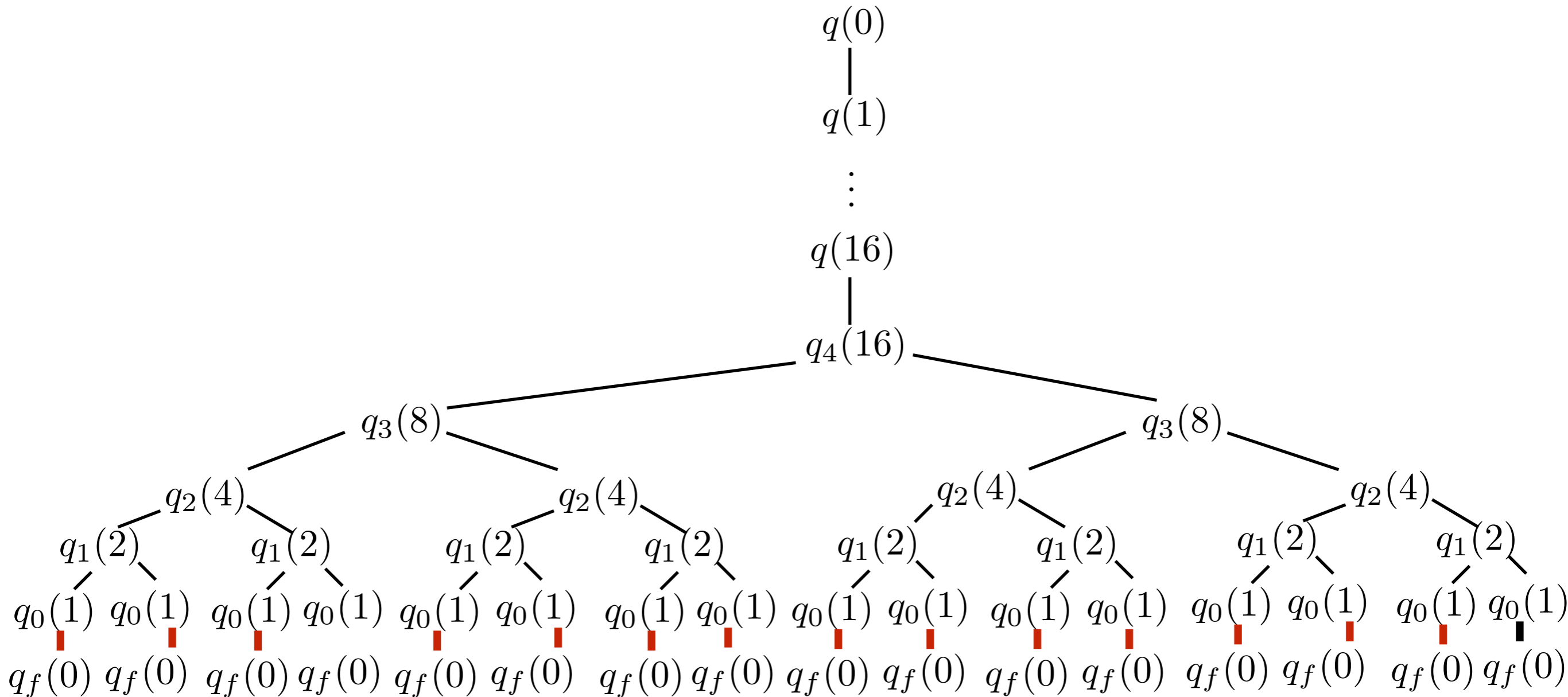
Reachability for a configuration is witnessed by a tree



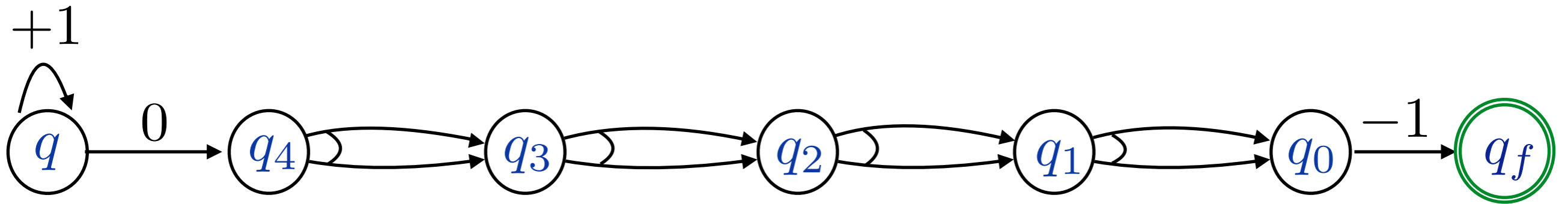
# Example for a 1-BVASS



Reachability for a configuration is witnessed by a tree

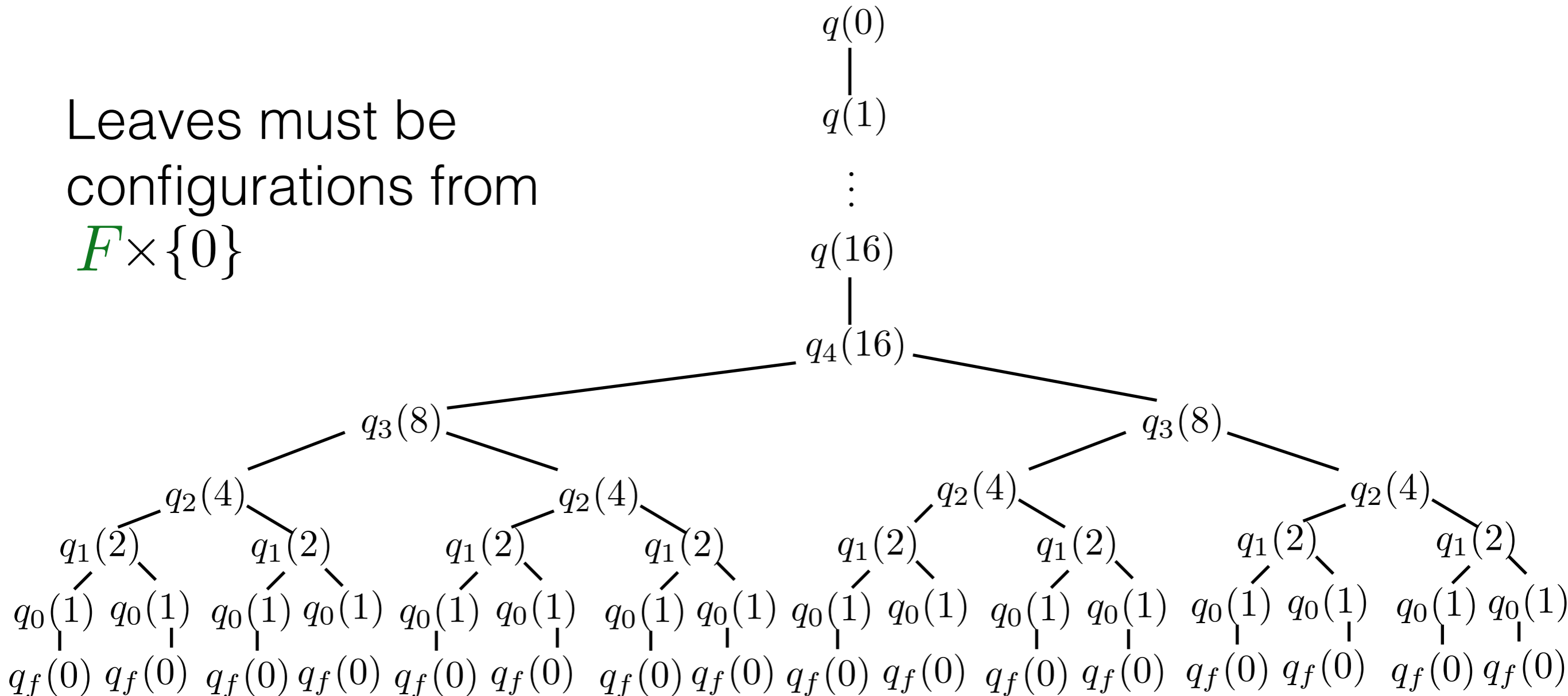


# Example for a 1-BVASS

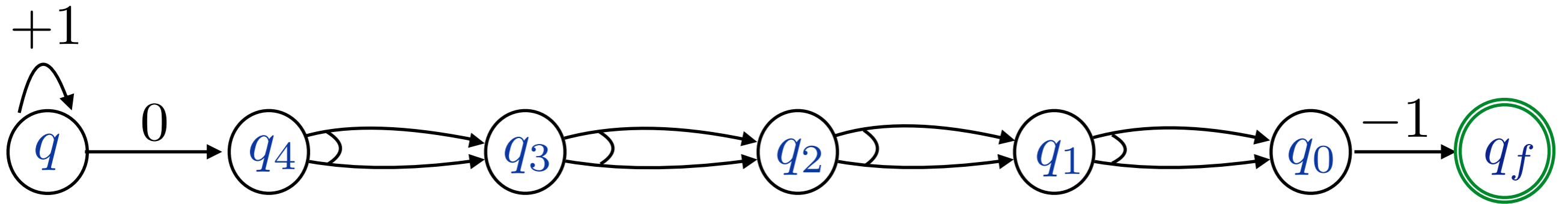


Reachability for a configuration is witnessed by a tree

Leaves must be configurations from  $F \times \{0\}$



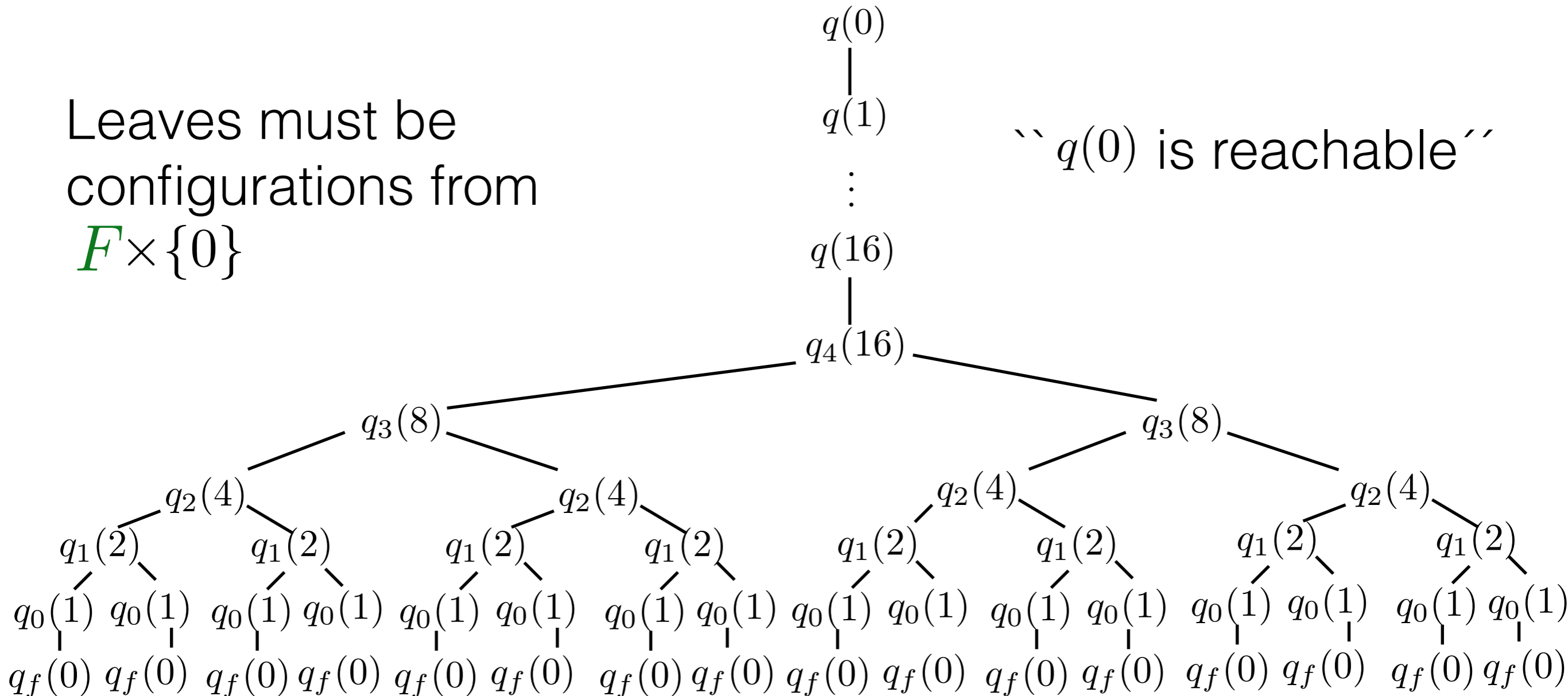
# Example for a 1-BVASS



Reachability for a configuration is witnessed by a tree

Leaves must be configurations from  $F \times \{0\}$

“ $q(0)$  is reachable”



# Some context

- BVASS were introduced in the context of computational linguistics  
(**Rambow '94**)



# Some context

- BVASS were introduced in the context of computational linguistics  
(**Rambow '94**)
- Reachability in BVASS is equivalent to provability in multiplicative exponential linear logic.  
(**de Groote, Guillaume, Salvati '04**)

# Some context

- BVASS were introduced in the context of computational linguistics  
(**Rambow '94**)
- Reachability in BVASS is equivalent to provability in multiplicative exponential linear logic.  
(**de Groote, Guillaume, Salvati '04**)
- Reachability of an extension of BVASS is equivalent to  $FO^2(<, +1, \sim)$  on data trees.  
(**Jacquemard, Segoufin, Dimono, '16**)

# Reachability in BVASS

- Coverability in BVASS is
    - 2EXPTIME-complete (bottom-up variant)  
**(Demri, Jurdziński, Lachish, Lazic '13)**
- INPUT
- conf.  $q(\vec{n})$
- leaves  $F \times \{0\}$
- root  $\geq q(\vec{n})$

# Reachability in BVASS

- |   | INPUT   |
|---|---|
| • Coverability in BVASS is  | conf. $q(\vec{n})$                                |
| - 2EXPTIME-complete (bottom-up variant)<br><b>(Demri, Jurdziński, Lachish, Lazic '13)</b> | leaves $F \times \{0\}$<br>root $\geq q(\vec{n})$ |
| - TOWER-complete (top-down variant)<br><b>(Schmitz, Lazic '14)</b>                        | leaves $F \times \mathbb{N}$<br>root $q(\vec{n})$ |

# Reachability in BVASS

- Coverability in BVASS is
  - 2EXPTIME-complete (bottom-up variant)  
**(Demri, Jurdziński, Lachish, Lazic '13)**  
INPUT  
conf.  $q(\vec{n})$   
leaves  $F \times \{0\}$   
root  $\geq q(\vec{n})$
  - TOWER-complete (top-down variant)  
**(Schmitz, Lazic '14)**  
leaves  $F \times \mathbb{N}$   
root  $q(\vec{n})$
- Reachability in BVASS  
is TOWER-hard **(Schmitz, Lazic '14)**

# Reachability in BVASS

- Coverability in BVASS is
  - 2EXPTIME-complete (bottom-up variant)  
(**Demri, Jurdziński, Lachish, Lazic '13**)
  - TOWER-complete (top-down variant)  
(**Schmitz, Lazic '14**)
- Reachability in BVASS
  - is TOWER-hard (**Schmitz, Lazic '14**)
  - not known to be decidable**

INPUT

conf.  $q(\vec{n})$

leaves  $F \times \{0\}$

root  $\geq q(\vec{n})$

leaves  $F \times \mathbb{N}$

root  $q(\vec{n})$



# Exponentiality in 1-BVASS reachability

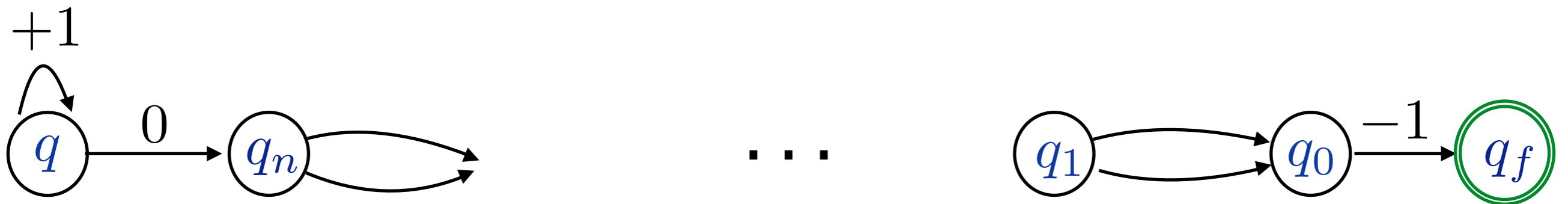
In the 1-BVASS



there is precisely one reachability tree for  $q(0)$  .

# Exponentiality in 1-BVASS reachability

In the 1-BVASS



there is precisely one reachability tree for  $q(0)$  .

It has  $2^{n+2}$  nodes and

$2^n + n + 3$  different configurations.



# Exponentiality in 1-BVASS reachability

In the 1-BVASS



there is precisely one reachability tree for  $q(0)$  .

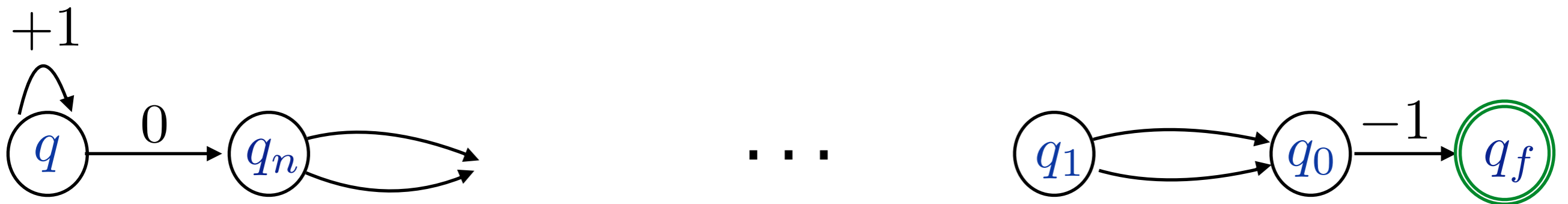
It has  $2^{n+2}$  nodes and

$2^n + n + 3$  different configurations.

**Proposition.** The following problem is NP-hard:

# Exponentiality in 1-BVASS reachability

In the 1-BVASS



there is precisely one reachability tree for  $q(0)$  .

It has  $2^{n+2}$  nodes and

$2^n + n + 3$  different configurations.

**Proposition.** The following problem is NP-hard: in binary  
↓

INPUT: 1-BVASS with unary updates and configuration  $q(n)$

QUESTION: Is  $q(n)$  reachable?

# Exponentiality in 1-BVASS reachability

In the 1-BVASS



there is precisely one reachability tree for  $q(0)$ .

It has  $2^{n+2}$  nodes and

$2^n + n + 3$  different configurations.

**Proposition.** The following problem is NP-hard: in binary  
↓

INPUT: 1-BVASS with unary updates and configuration  $q(n)$

QUESTION: Is  $q(n)$  reachable?

From **SUBSET SUM**

# Reachability in 1-BVASS

**Theorem.** (G, Haase, Lazic, Totzke 2016)

Reachability in 1-BVASS (updates in unary) is PTIME-complete.

# Reachability in 1-BVASS

**Theorem.** (G, Haase, Lazic, Totzke 2016)

Reachability in 1-BVASS (updates in unary) is PTIME-complete.

**Lower bound:** Obvious reduction from Circuit Value

# Reachability in 1-BVASS

**Theorem.** (G, Haase, Lazic, Totzke 2016)

Reachability in 1-BVASS (updates in unary) is PTIME-complete.

**Lower bound:** Obvious reduction from Circuit Value

**Upper bound strategy:**

1) **Residue reachability** for unary 1-BVASS is in PTIME.

INPUT:

1-BVASS  $(Q, T, F)$ , configuration  $q(n)$ ,  $d \geq 1$

QUESTION:

$\exists m \geq n$  s.t.  $q(m)$  is reachable and  $m \equiv n \pmod{d}$  ?

# Reachability in 1-BVASS

**Theorem.** (G, Haase, Lazic, Totzke 2016)

Reachability in 1-BVASS (updates in unary) is PTIME-complete.

**Lower bound:** Obvious reduction from Circuit Value

**Upper bound strategy:**

1) **Residue reachability** for unary 1-BVASS is in PTIME.

INPUT:

1-BVASS  $(Q, T, F)$ , configuration  $q(n)$ ,  $d \geq 1$

QUESTION:

$\exists m \geq n$  s.t.  $q(m)$  is reachable and  $m \equiv n \pmod{d}$  ?

2) **Reachability** 1-BVASS  $\leq^P$  **Residue reachability** 1-BVASS.

# Bounded reachability is in PTIME

**Proposition.** The following problem is in PTIME:

INPUT:

1-BVASS  $(Q, T, F)$ , configuration  $q(n)$ ,  $b \in \mathbb{N}$

all numbers in unary.

QUESTION:

Is there a reachability tree for  $q(n)$  in which  
all configurations have counter value  $\leq b$  ?



# Bounded reachability is in PTIME

**Proposition.** The following problem is in PTIME:

INPUT:

1-BVASS  $(Q, T, F)$ , configuration  $q(n)$ ,  $b \in \mathbb{N}$

all numbers in unary.

QUESTION:

Is there a reachability tree for  $q(n)$  in which  
all configurations have counter value  $\leq b$  ?

**Proof.**

Simple saturation by Dynamic Programming.

# Residue reachability (1/4)

INPUT:

1-BVASS  $(Q, T, F)$ , configuration  $q(n)$ ,  $d \geq 1$

QUESTION:

$\exists m \geq n$  s.t.  $q(m)$  is reachable and  $m \equiv n \pmod{d}$  ?

# Residue reachability (1/4)

INPUT:

1-BVASS  $(Q, T, F)$ , configuration  $q(n)$ ,  $d \geq 1$

QUESTION:

$\exists m \geq n$  s.t.  $q(m)$  is reachable and  $m \equiv n \pmod{d}$  ?

**Corollary.**

The set  $S \stackrel{def}{=} \{p(m) \mid \exists \underbrace{(n + |Q| \cdot d)}_b \text{-bounded reach. tree for } p(m)\}$

is computable in PTIME.

# Residue reachability (1/4)

INPUT:

1-BVASS  $(Q, T, F)$ , configuration  $q(n)$ ,  $d \geq 1$

QUESTION:

$\exists m \geq n$  s.t.  $q(m)$  is reachable and  $m \equiv n \pmod{d}$  ?

**Corollary.**

The set  $S \stackrel{def}{=} \{p(m) \mid \exists \underbrace{(n + |Q| \cdot d)}_b \text{-bounded reach. tree for } p(m)\}$   
is computable in PTIME.

What about reachability trees that involve counter values  $> b$  ?



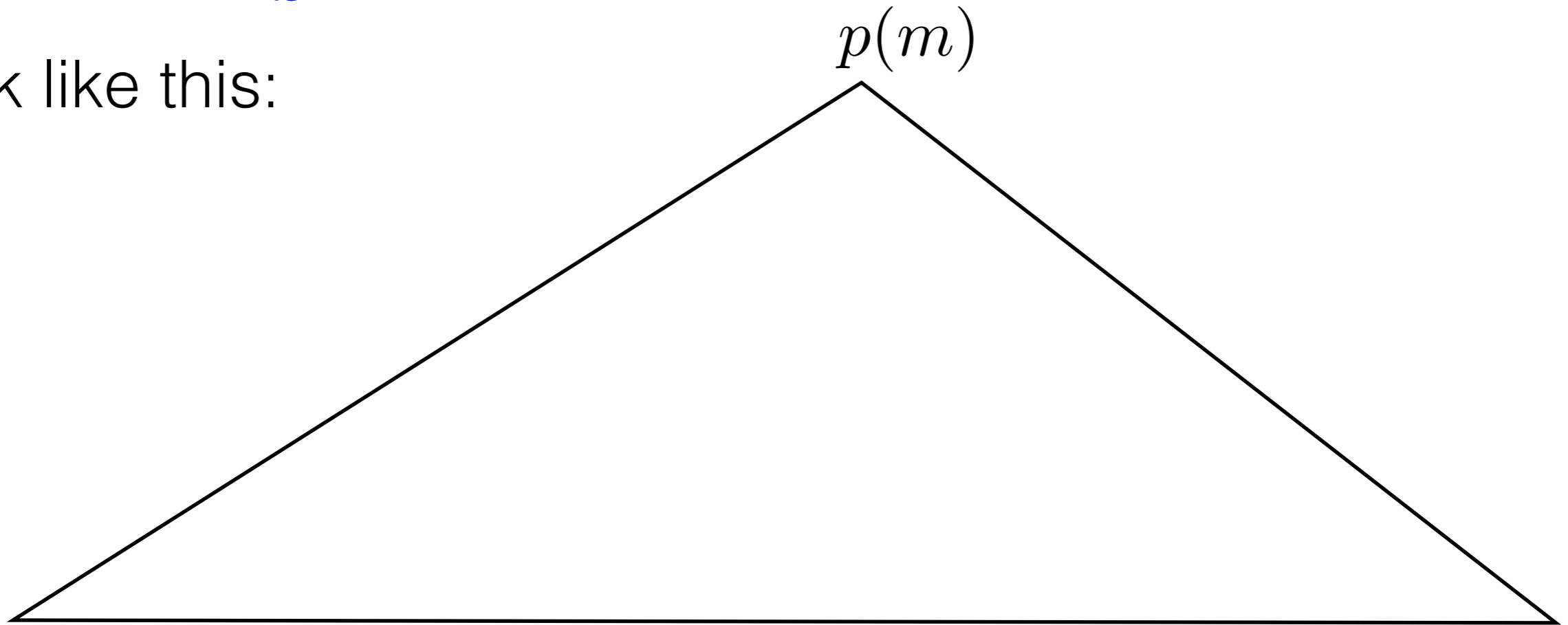
# Residue reachability (2/4)

What about reachability trees that involve counter values  $> \mathbf{b}$ ?

# Residue reachability (2/4)

What about reachability trees that involve counter values  $> \mathbf{b}$ ?

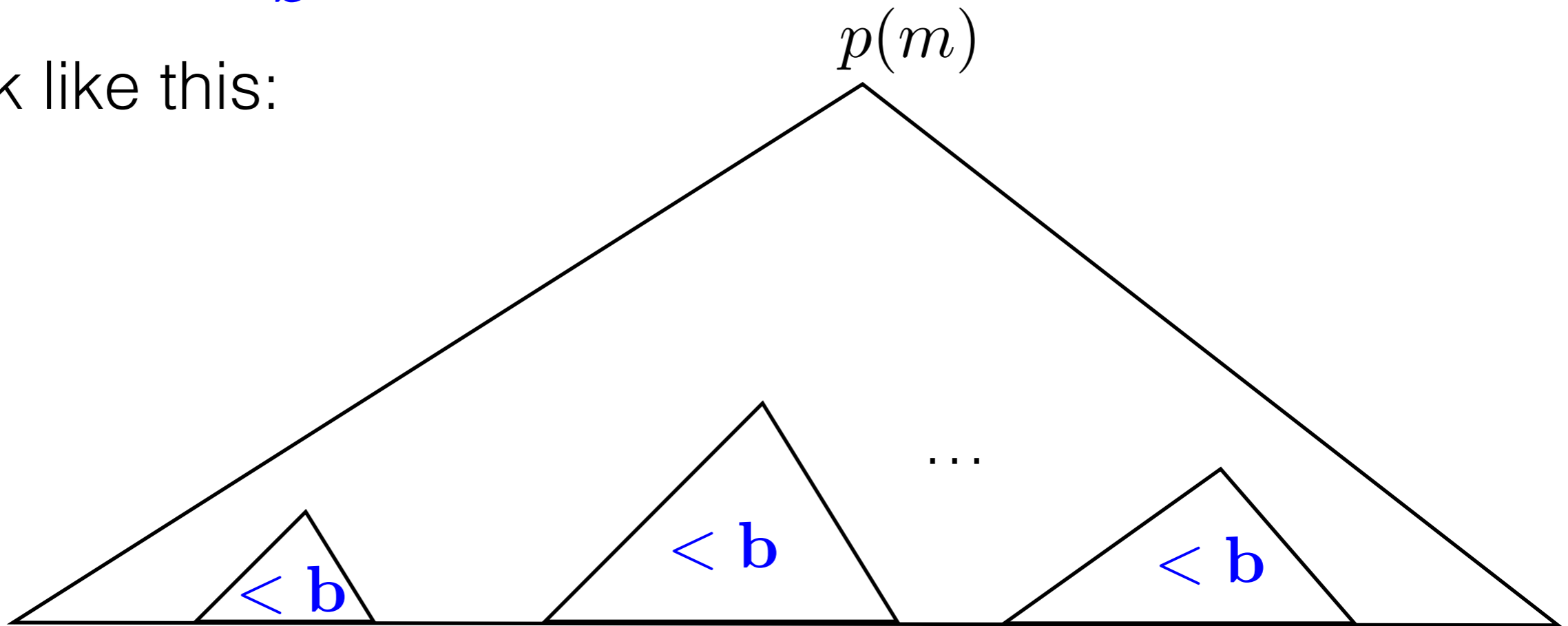
They look like this:



# Residue reachability (2/4)

What about reachability trees that involve counter values  $> \mathbf{b}$ ?

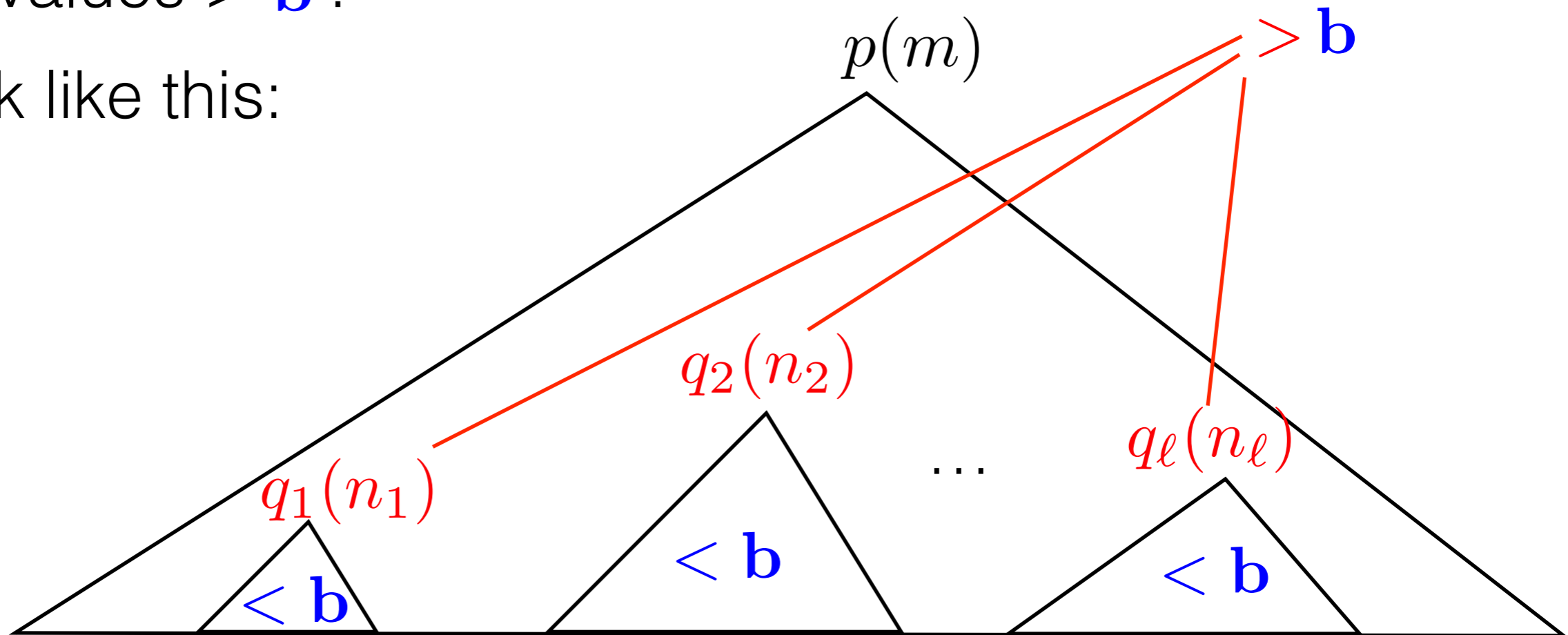
They look like this:



# Residue reachability (2/4)

What about reachability trees that involve counter values  $> \mathbf{b}$ ?

They look like this:

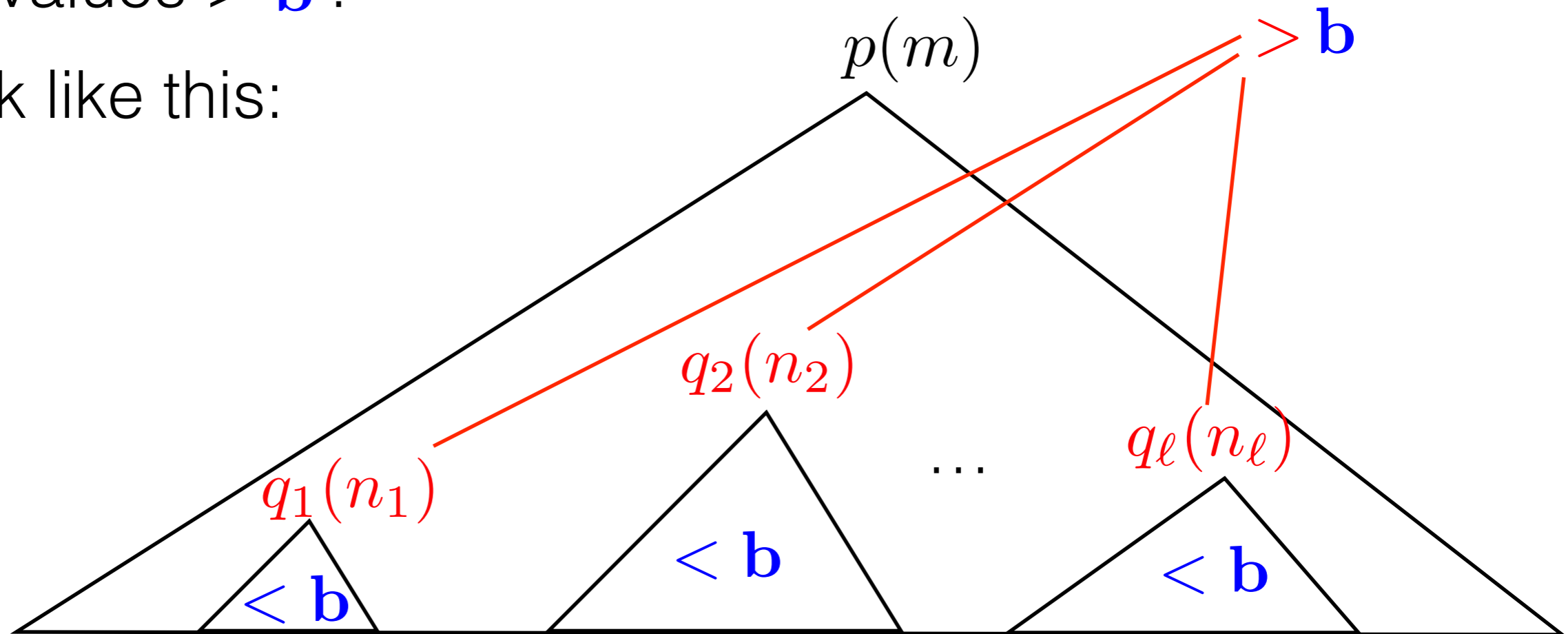




# Residue reachability (2/4)

What about reachability trees that involve counter values  $> \mathbf{b}$ ?

They look like this:



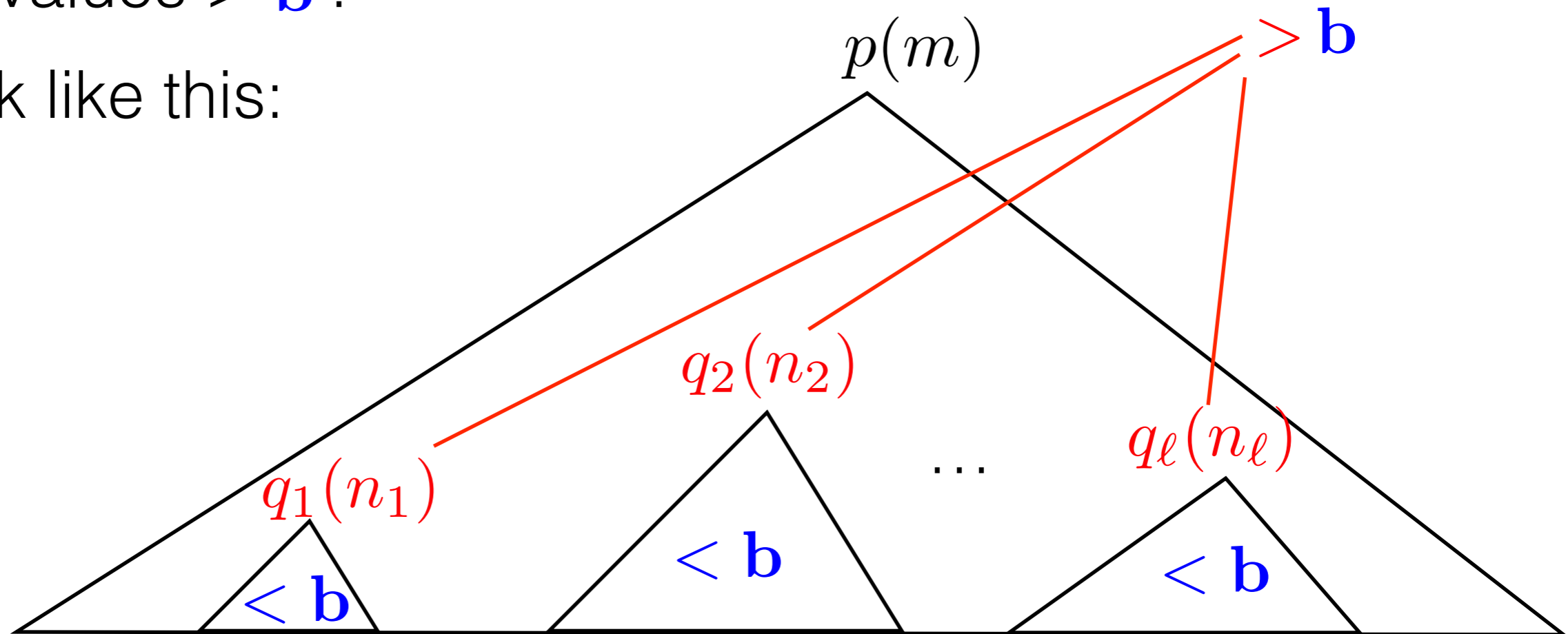
Let us collect these residue classes:

$$R_0 \stackrel{def}{=} \left\{ q(n \bmod d) \mid \begin{array}{c} q(n) \\ \triangle \\ < \mathbf{b} \end{array} \right\} \subseteq Q \times \mathbb{Z}_d$$

# Residue reachability (2/4)

What about reachability trees that involve counter values  $> \mathbf{b}$ ?

They look like this:



Let us collect these residue classes:

$$R_0 \stackrel{def}{=} \left\{ q(n \bmod d) \mid \begin{array}{c} q(n) \\ \triangle \\ < \mathbf{b} \end{array} \right\} \subseteq Q \times \mathbb{Z}_d$$

**Computable in PTIME!**

# Residue reachability (3/4)

Define  $R_i \subseteq Q \times \mathbb{Z}_d$  for each  $i \geq 1$

# Residue reachability (3/4)

Define  $R_i \subseteq Q \times \mathbb{Z}_d$  for each  $i \geq 1$

$$R_{i+1} \stackrel{def}{=} R_i$$

# Residue reachability (3/4)

Define  $R_i \subseteq Q \times \mathbb{Z}_d$  for each  $i \geq 1$

$$R_{i+1} \stackrel{def}{=} R_i \cup T(R_i)$$

Residue classes obtainable  
from  $R_i$  by applying a **unary rule**.

# Residue reachability (3/4)

Define  $R_i \subseteq Q \times \mathbb{Z}_d$  for each  $i \geq 1$

$$R_{i+1} \stackrel{def}{=} R_i \cup T(R_i) \cup T(R_i, S/\mathbb{Z}_d)$$

Residue classes obtainable from  $R_i$  by applying a **unary rule**.

Residue classes obtainable by applying **binary rule**:  
Left child from  $R_i$   
Right child from  $S/\mathbb{Z}_d$

# Residue reachability (3/4)

Define  $R_i \subseteq Q \times \mathbb{Z}_d$  for each  $i \geq 1$

$$R_{i+1} \stackrel{def}{=} R_i \cup T(R_i) \cup T(R_i, S/\mathbb{Z}_d)$$

$$\cup T(S/\mathbb{Z}_d, R_i)$$

Residue classes obtainable from  $R_i$  by applying a **unary rule**.

Residue classes obtainable by applying **binary rule**:  
Left child from  $R_i$   
Right child from  $S/\mathbb{Z}_d$

**binary rule**  
Left child from  $S/\mathbb{Z}_d$   
Right child  $R_i$

# Residue reachability (3/4)

Define  $R_i \subseteq Q \times \mathbb{Z}_d$  for each  $i \geq 1$

$$R_{i+1} \stackrel{def}{=} R_i \cup T(R_i) \cup T(R_i, S/\mathbb{Z}_d)$$

$$\cup T(S/\mathbb{Z}_d, R_i)$$

Residue classes obtainable from  $R_i$  by applying a **unary rule**.

Residue classes obtainable by applying **binary rule**:  
Left child from  $R_i$   
Right child from  $S/\mathbb{Z}_d$

**binary rule**  
Left child from  $S/\mathbb{Z}_d$   
Right child  $R_i$

## Observation.

The fixed point  $R \stackrel{def}{=} \bigcup \{R_i \mid i \geq 0\}$  is computable in PTIME.



# Residue reachability (4/4)

## **Lemma.**

For all  $q(r) \in Q \times \mathbb{Z}_d$  we have

- $\exists m \geq n : m \equiv r \pmod{d}$  and  $q(m)$  is reachable

**if, and only, if**

# Residue reachability (4/4)

## Lemma.

For all  $q(r) \in Q \times \mathbb{Z}_d$  we have

- $\exists m \geq n : m \equiv r \pmod{d}$  and  $q(m)$  is reachable

**if, and only, if**

- $q(r) \in R$  (deals with proof trees having at least one counter value  $> \mathbf{b}$ )

or

# Residue reachability (4/4)

## Lemma.

For all  $q(r) \in Q \times \mathbb{Z}_d$  we have

- $\exists m \geq n : m \equiv r \pmod{d}$  and  $q(m)$  is reachable

**if, and only, if**

- $q(r) \in R$  (deals with proof trees having at least one counter value  $> \mathbf{b}$ )

or

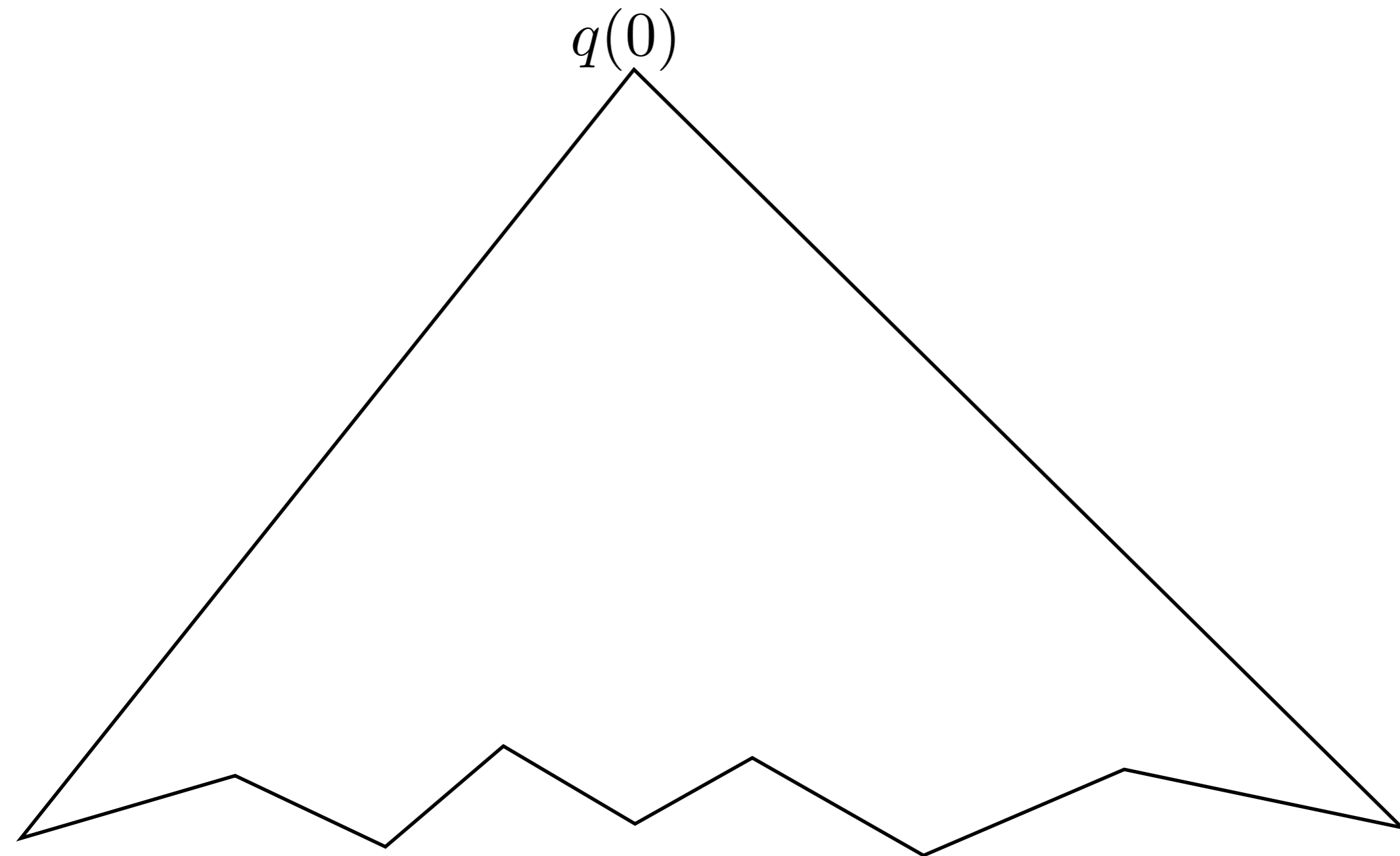
$\exists m \in [n, n + \overbrace{|Q| \cdot d}^{\mathbf{b}}], m \equiv n \pmod{d}$  and  $q(m) \in S$

(deals with proof trees having all counter values  $< \mathbf{b}$ )

1-BVASS:

Reachability  $\leq \frac{P}{T}$  residue reachability

Assume a minimal reachability tree for  $q(0)$

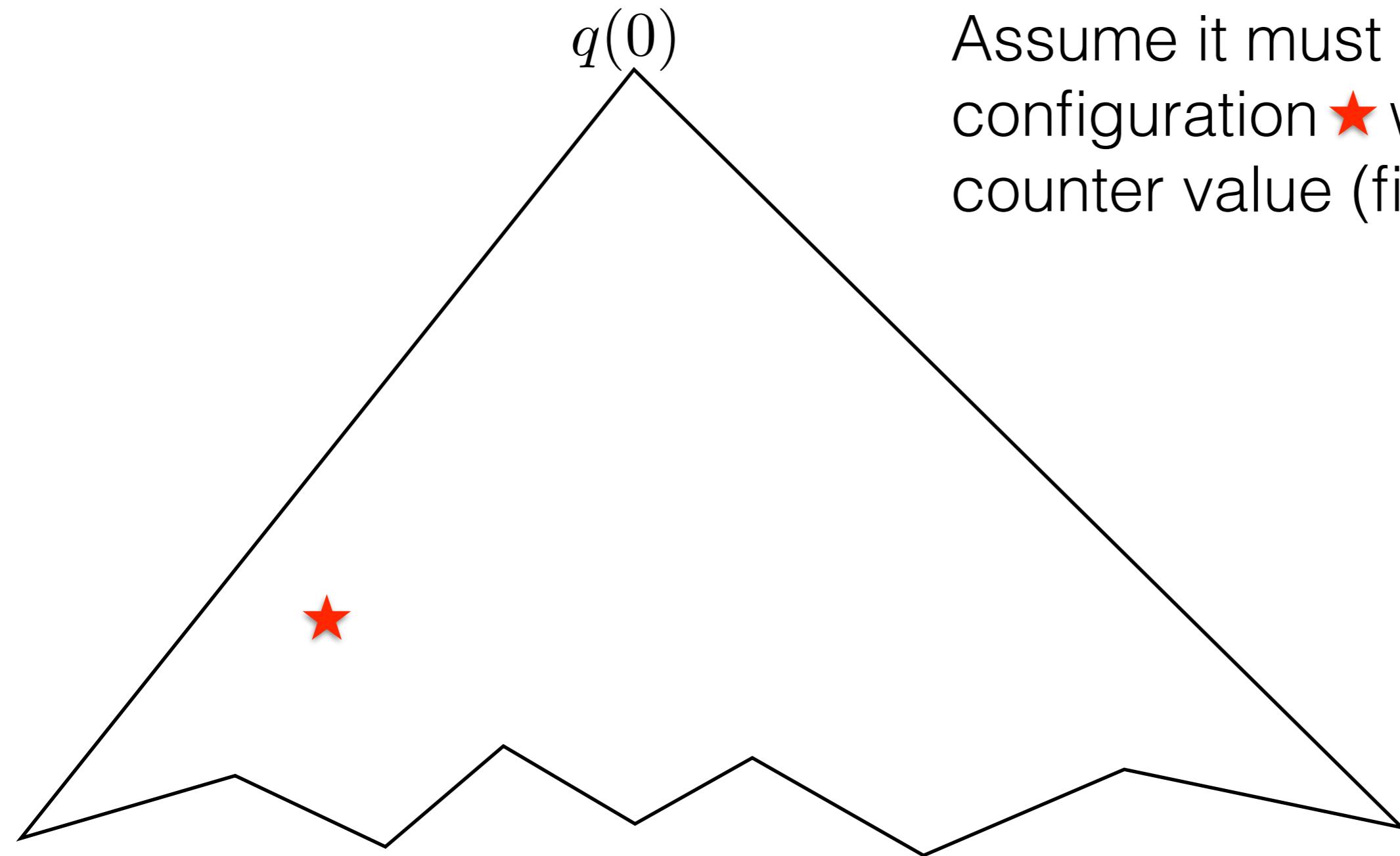


# 1-BVASS:

Reachability  $\leq \frac{P}{T}$  residue reachability

Assume a minimal reachability tree for  $q(0)$


Assume it must involve a configuration ★ with a `big` counter value (first time)



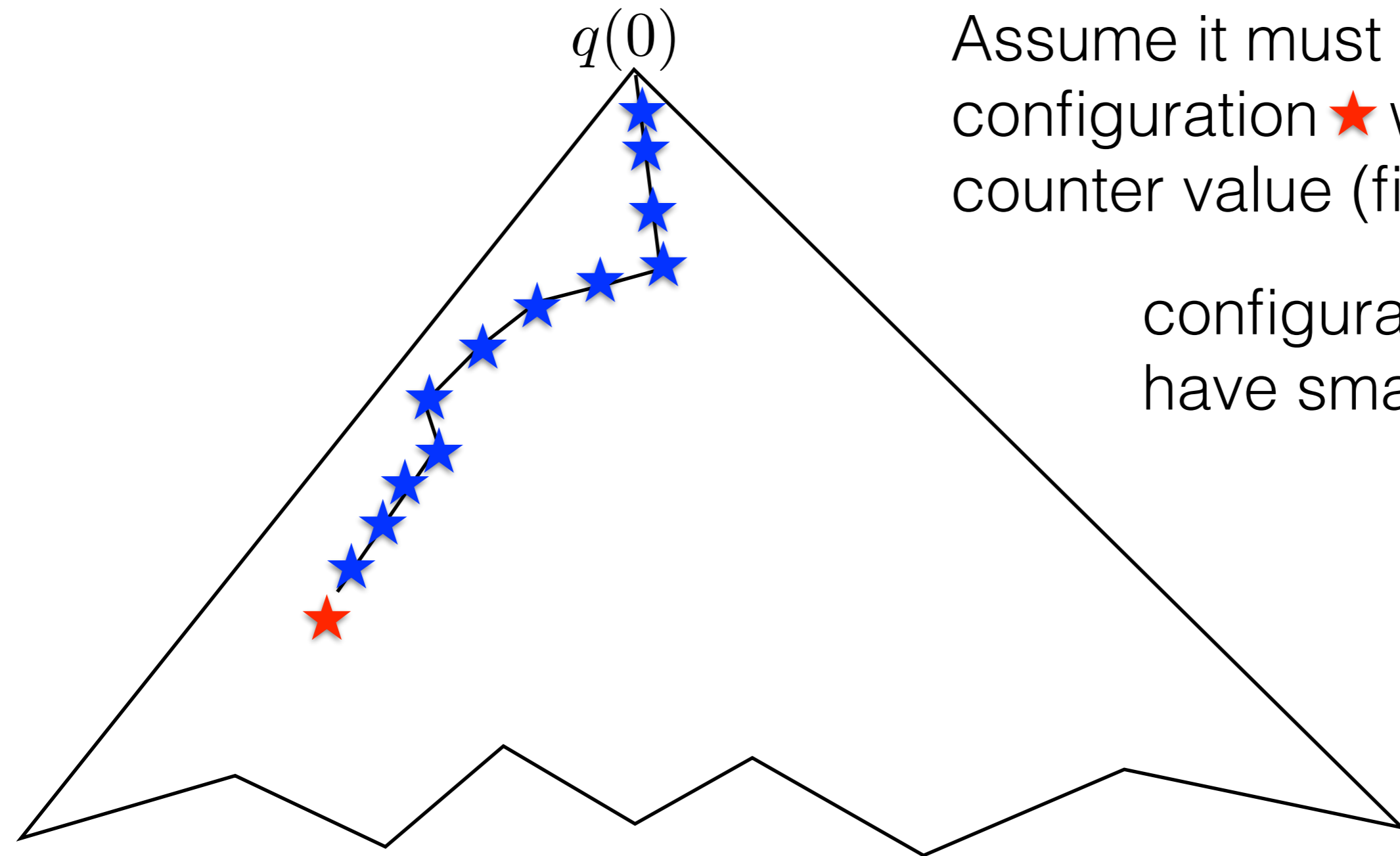
# 1-BVASS:

Reachability  $\leq \frac{P}{T}$  residue reachability

Assume a minimal reachability tree for  $q(0)$

Assume it must involve a configuration  with a 'big' counter value (first time)


configurations  have small values



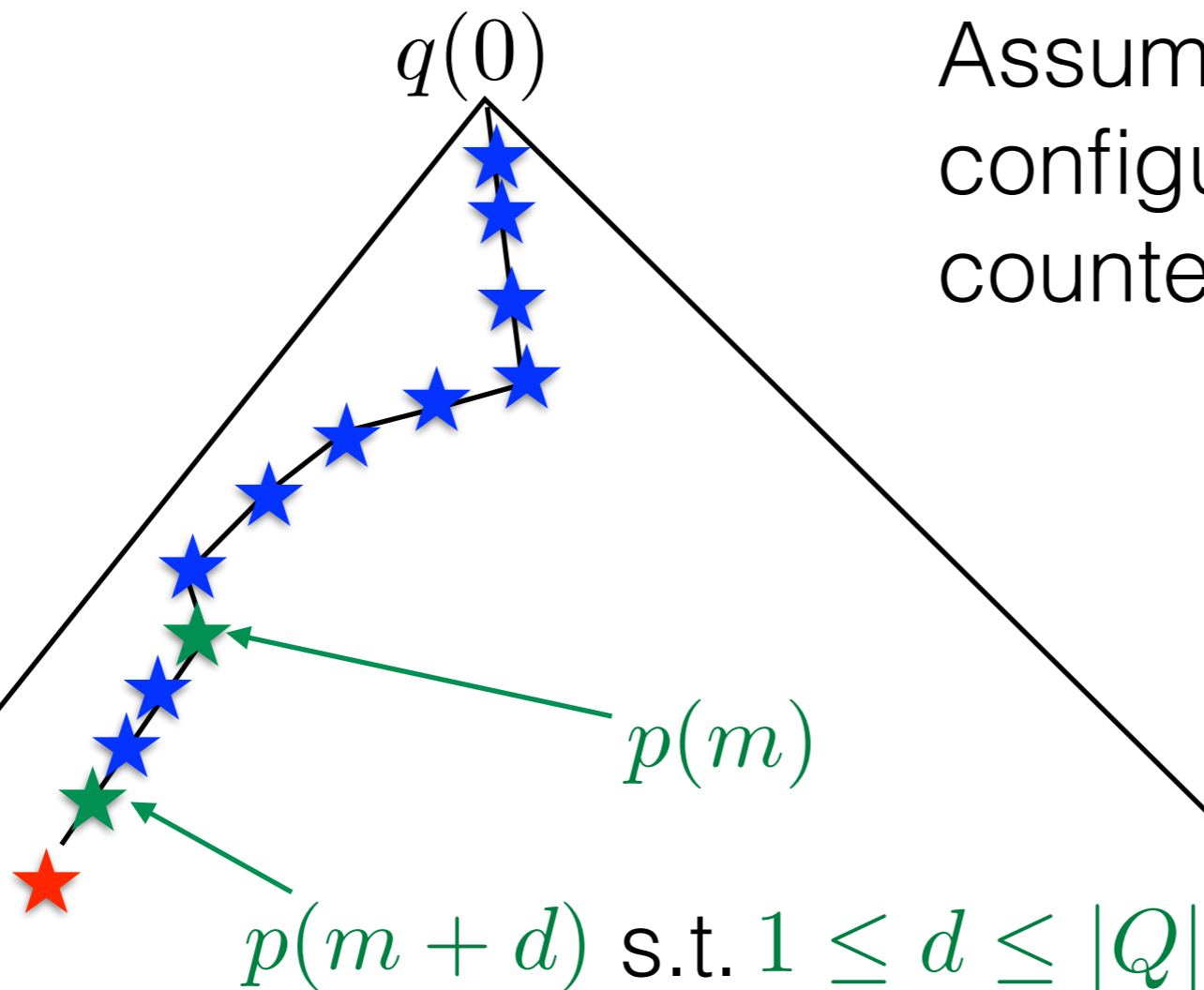
# 1-BVASS:

Reachability  $\leq \frac{P}{T}$  residue reachability

Assume a minimal reachability tree for  $q(0)$

Assume it must involve a configuration  with a 'big' counter value (first time)

configurations  have small values



# 1-BVASS:

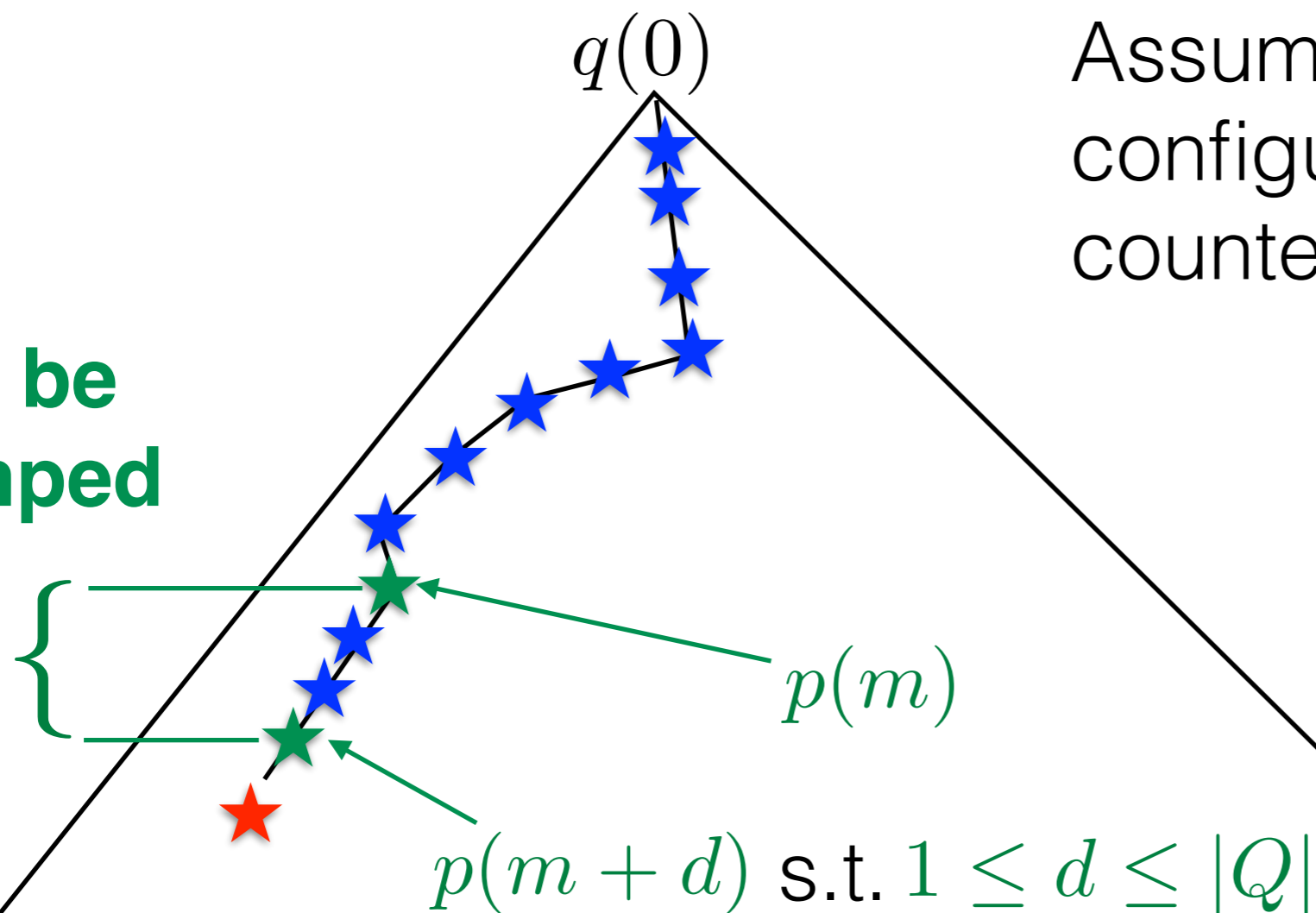
Reachability  $\leq \frac{P}{T}$  residue reachability

Assume a minimal reachability tree for  $q(0)$

Assume it must involve a configuration  $\star$  with a 'big' counter value (first time)

configurations  $\star$  have small values

Can be pumped






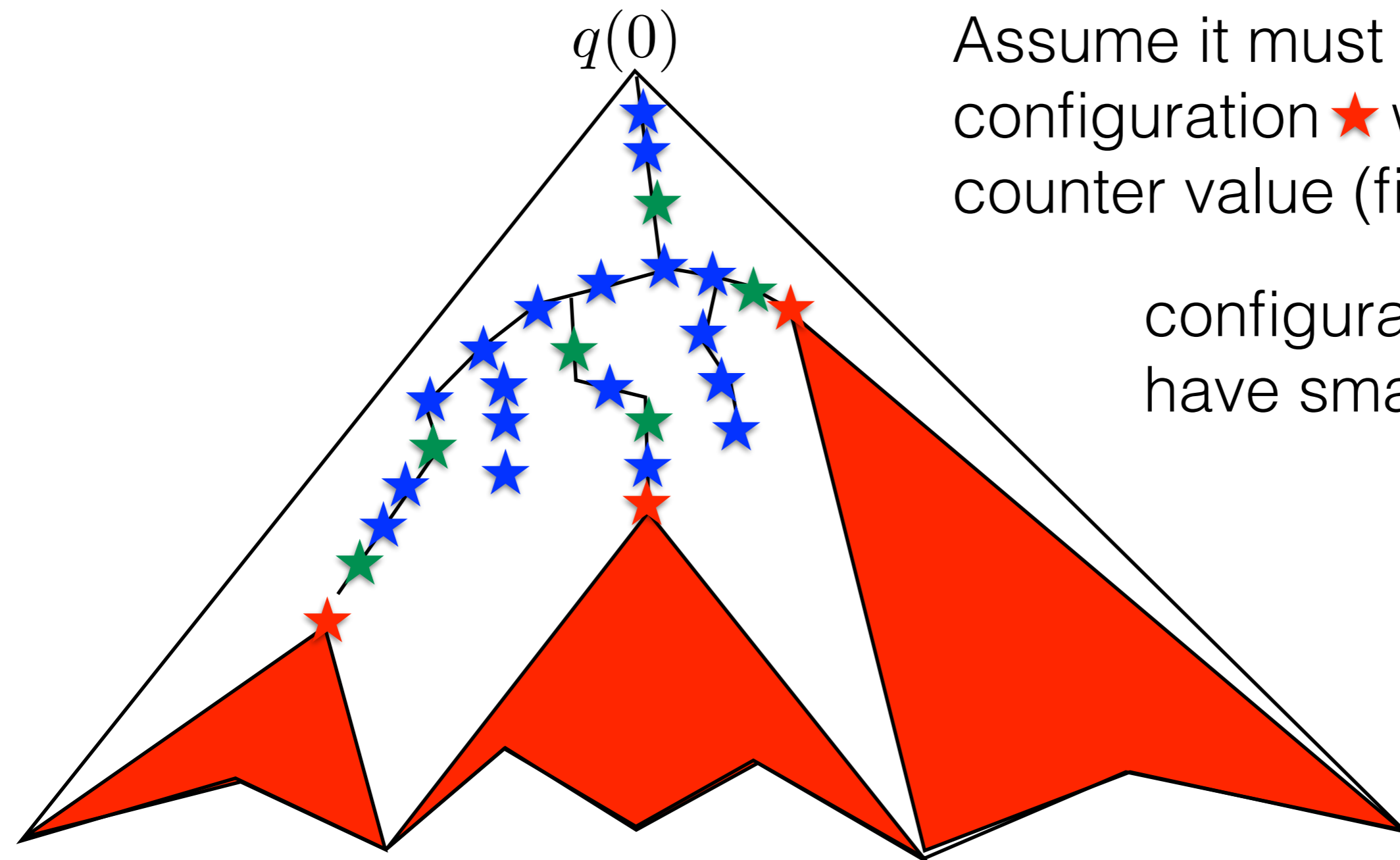
# 1-BVASS:

Reachability  $\leq \frac{P}{T}$  residue reachability

Assume a minimal reachability tree for  $q(0)$

Assume it must involve a configuration  with a 'big' counter value (first time)

configurations   
have small values




# 1-BVASS:

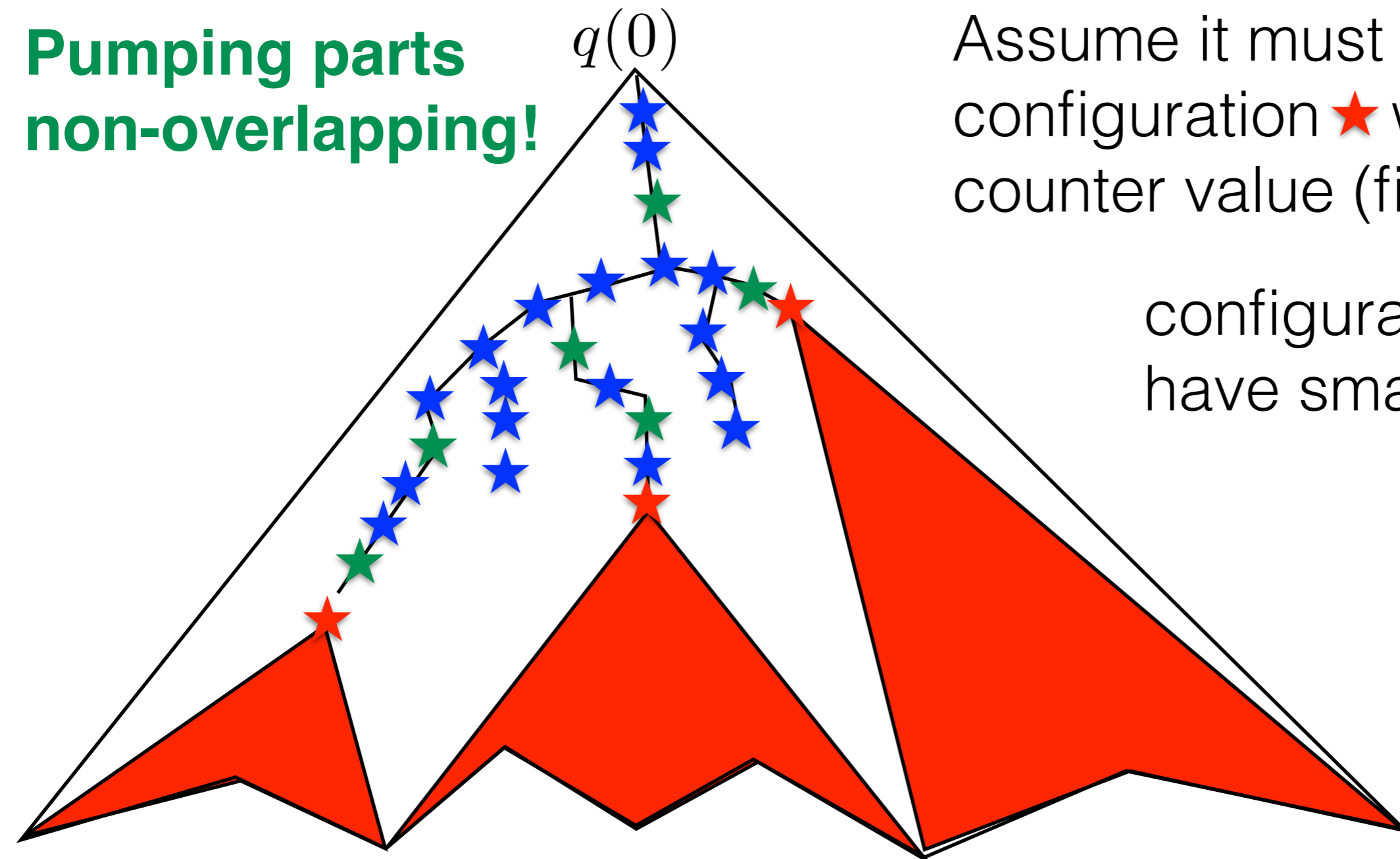
Reachability  $\leq \frac{P}{T}$  residue reachability

Assume a minimal reachability tree for  $q(0)$

**Pumping parts  
non-overlapping!**

Assume it must involve a configuration  with a 'big' counter value (first time)

configurations   
have small values




# 1-BVASS:

Reachability  $\leq \frac{P}{T}$  residue reachability

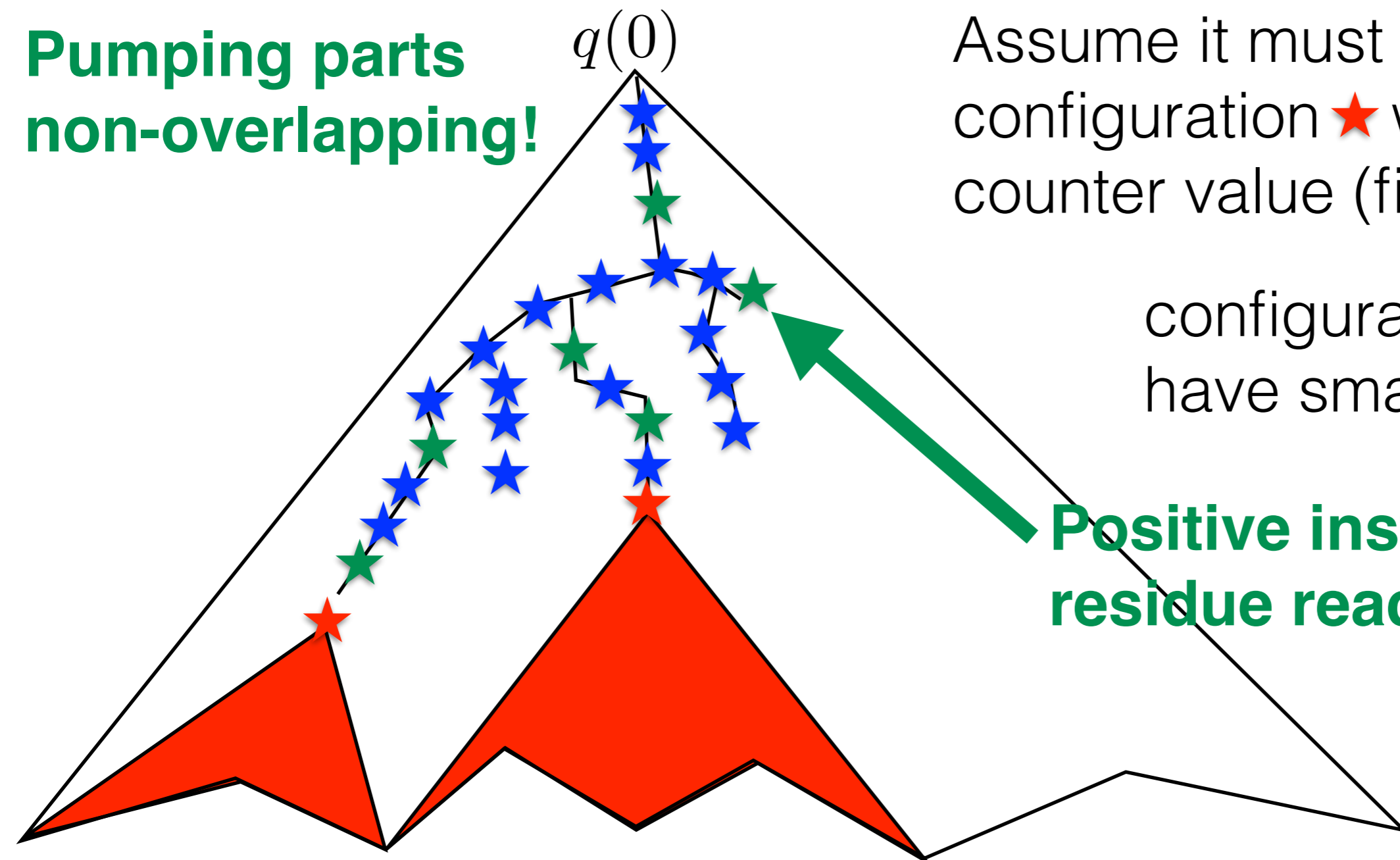
Assume a minimal reachability tree for  $q(0)$

**Pumping parts  
non-overlapping!**

Assume it must involve a configuration  with a 'big' counter value (first time)

configurations  have small values

**Positive instance of  
residue reachability!**




# 1-BVASS:

Reachability  $\leq \frac{P}{T}$  residue reachability

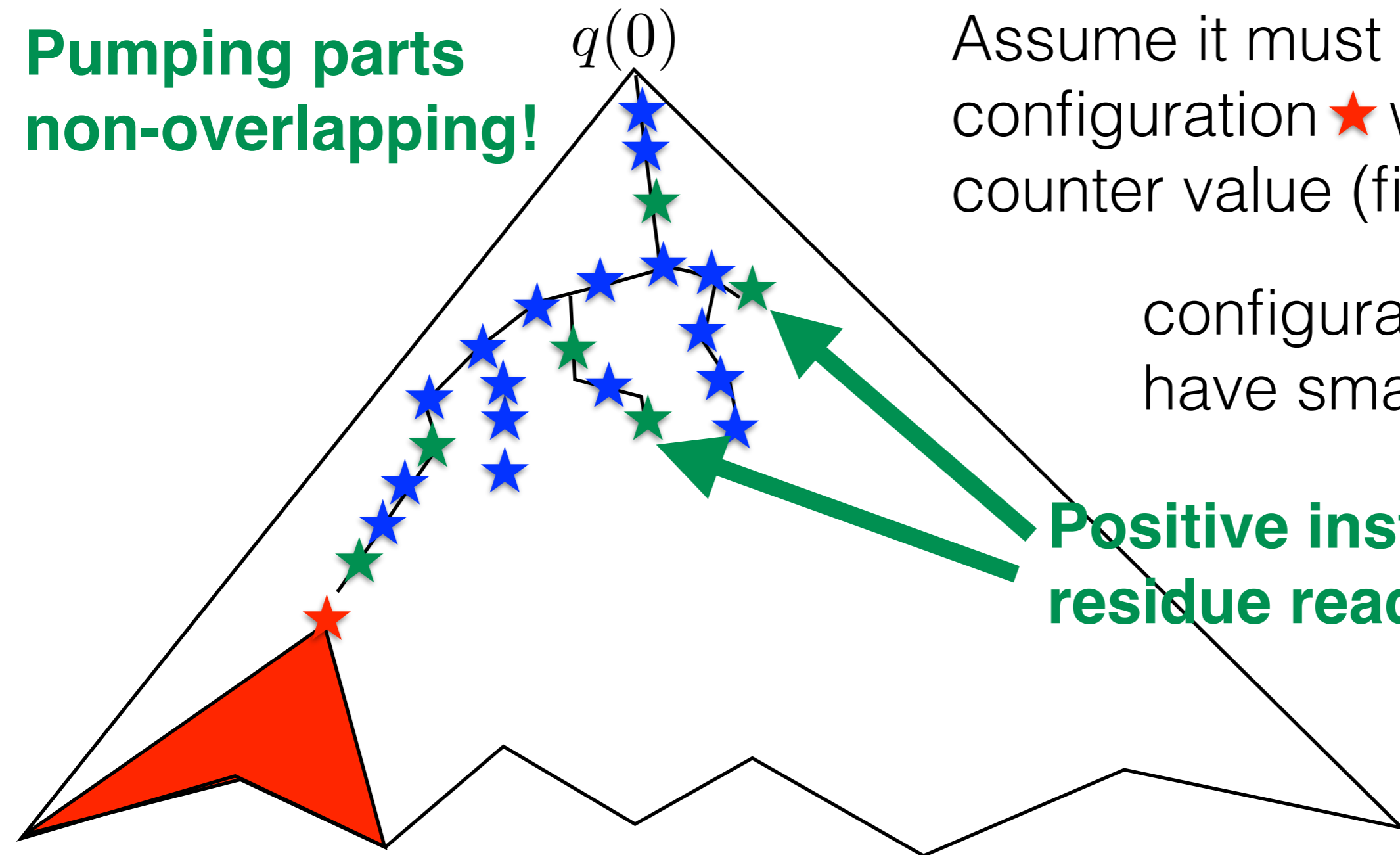
Assume a minimal reachability tree for  $q(0)$

**Pumping parts  
non-overlapping!**

Assume it must involve a configuration  with a 'big' counter value (first time)

configurations   
have small values

**Positive instance of  
residue reachability!**



# 1-BVASS:

Reachability  $\leq \frac{P}{T}$  residue reachability

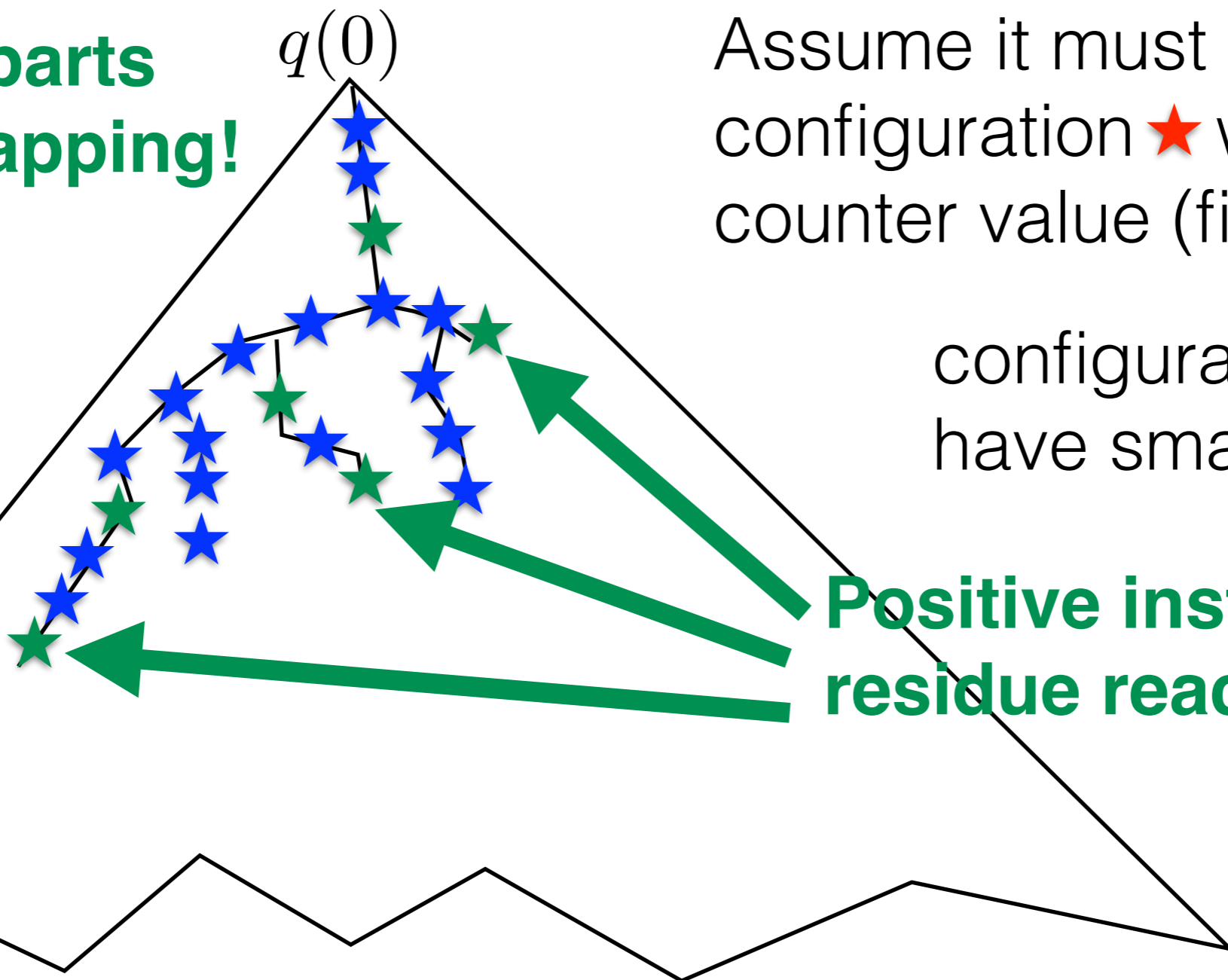
Assume a minimal reachability tree for  $q(0)$

**Pumping parts  
non-overlapping!**

Assume it must involve a configuration **★** with a 'big' counter value (first time)

configurations **★**  
have small values

**Positive instance of  
residue reachability!**

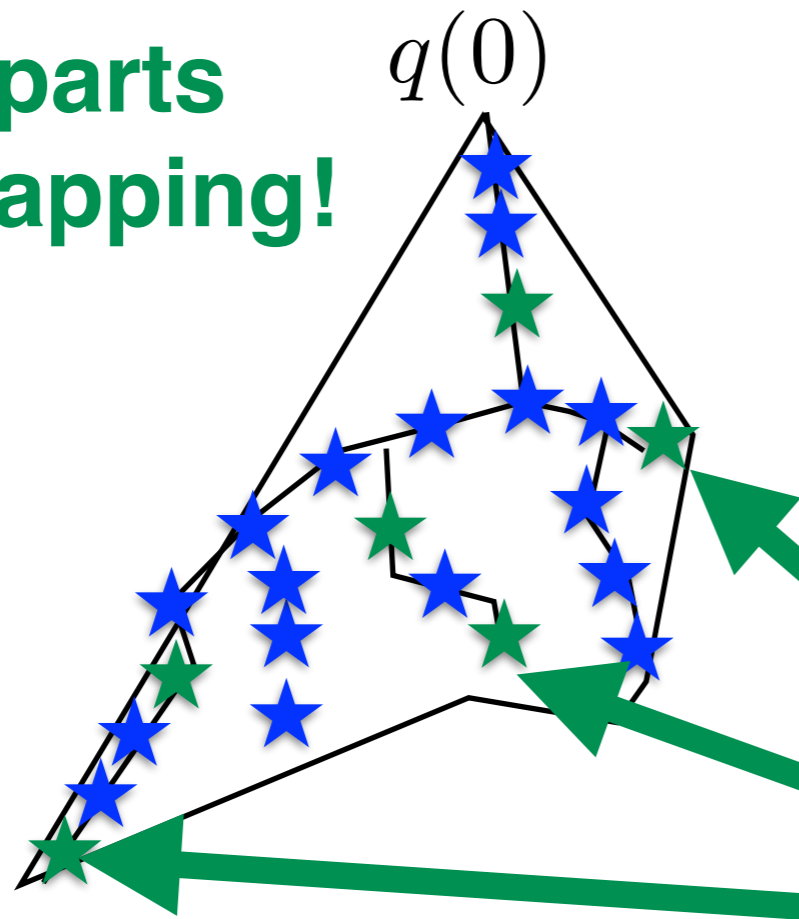


# 1-BVASS:

Reachability  $\leq \frac{P}{T}$  residue reachability

Assume a minimal reachability tree for  $q(0)$

**Pumping parts  
non-overlapping!**



Assume it must involve a configuration ★ with a `big` counter value (first time)

configurations ★  
have small values

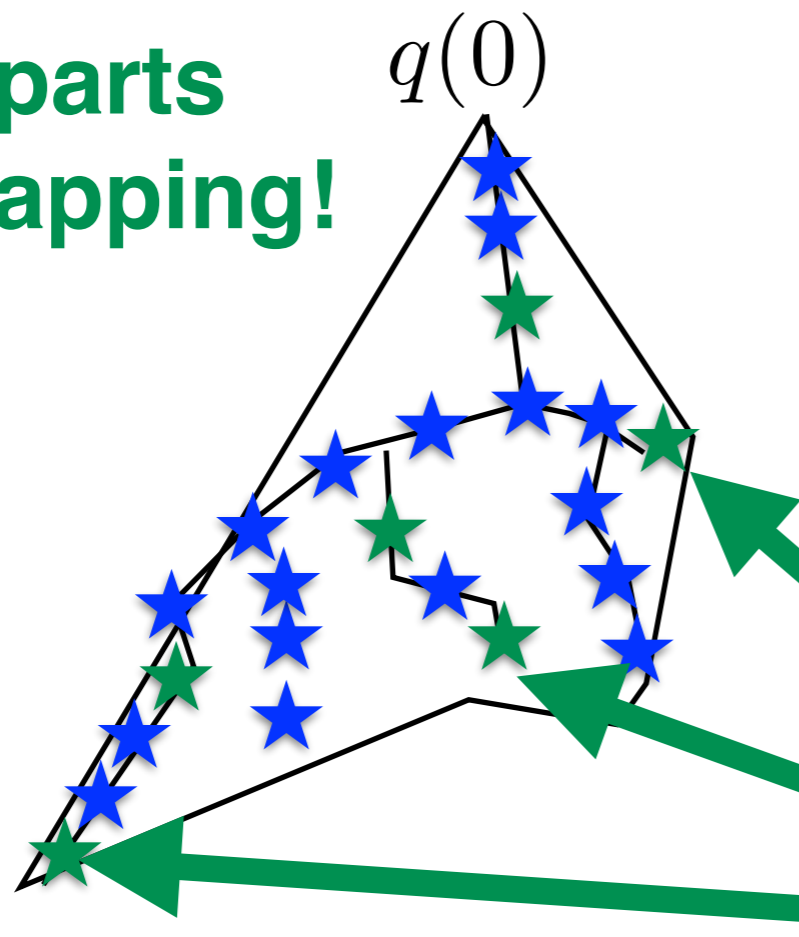
**Positive instance of  
residue reachability!**

# 1-BVASS:

Reachability  $\leq \frac{P}{T}$  residue reachability

Assume a minimal reachability tree for  $q(0)$

**Pumping parts  
non-overlapping!**



Assume it must involve a configuration ★ with a 'big' counter value (first time)

configurations ★  
have small values

**Positive instance of  
residue reachability!**

Alternating logspace machine guesses tree and instances to suitably guessed instance of **residue reachability** oracle.

# Outlook & Future Work



- Reachability for BVASS



# Outlook & Future Work



- Reachability for BVASS
- Binary 1-BVASS: PTIME...EXPTIME

# Outlook & Future Work



- Reachability for BVASS
- Binary 1-BVASS: PTIME...EXPTIME
- Is reachability for 2-BVASS decidable?

# Outlook & Future Work

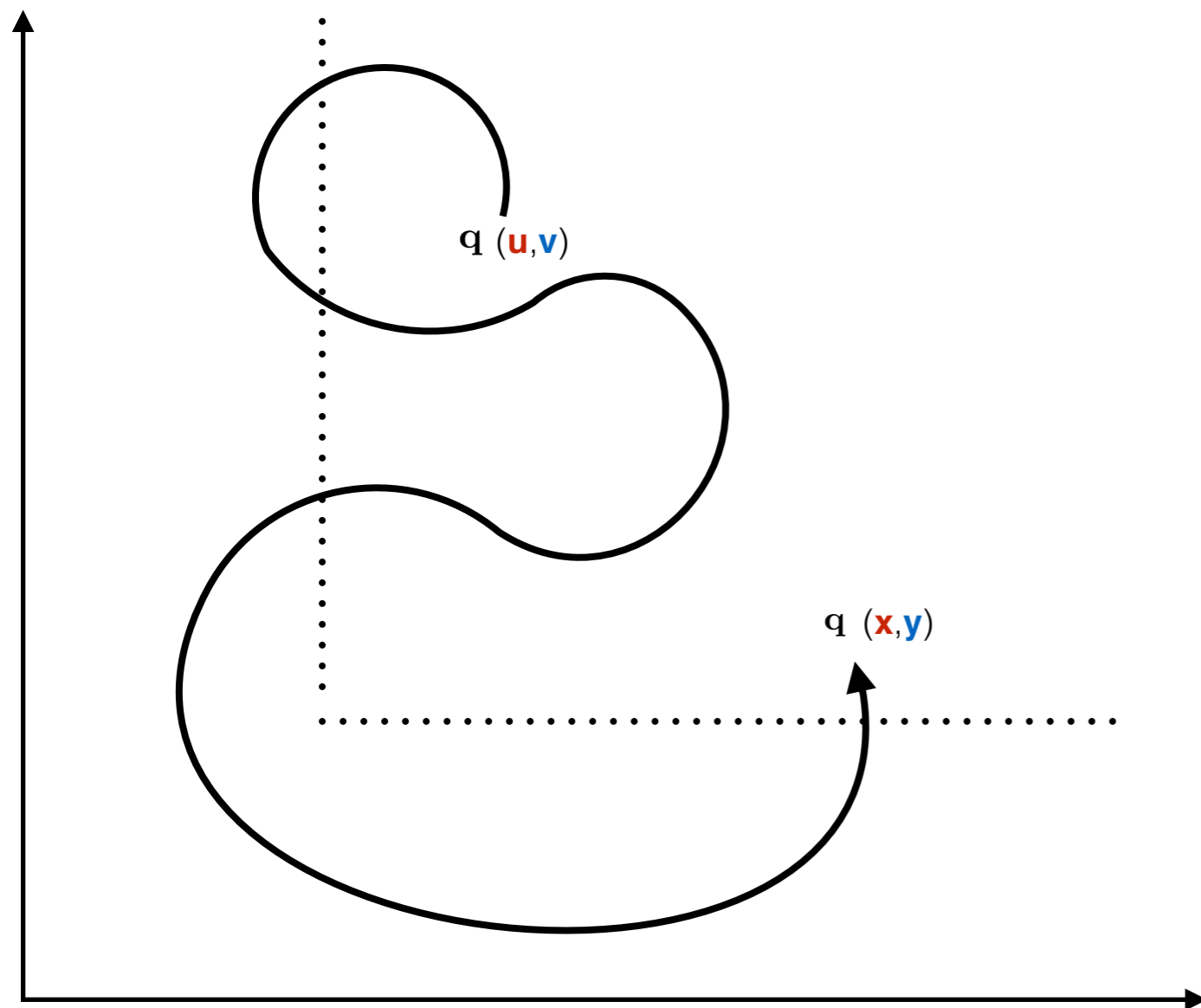


- Reachability for BVASS
- Binary 1-BVASS: PTIME...EXPTIME
- Is reachability for 2-BVASS decidable?
- Is reachability for 2-BVASS semilinear?

Thank you for your attention!

# Type 1 runs

Starting and ending **sufficiently large** in **same** control state  $q$

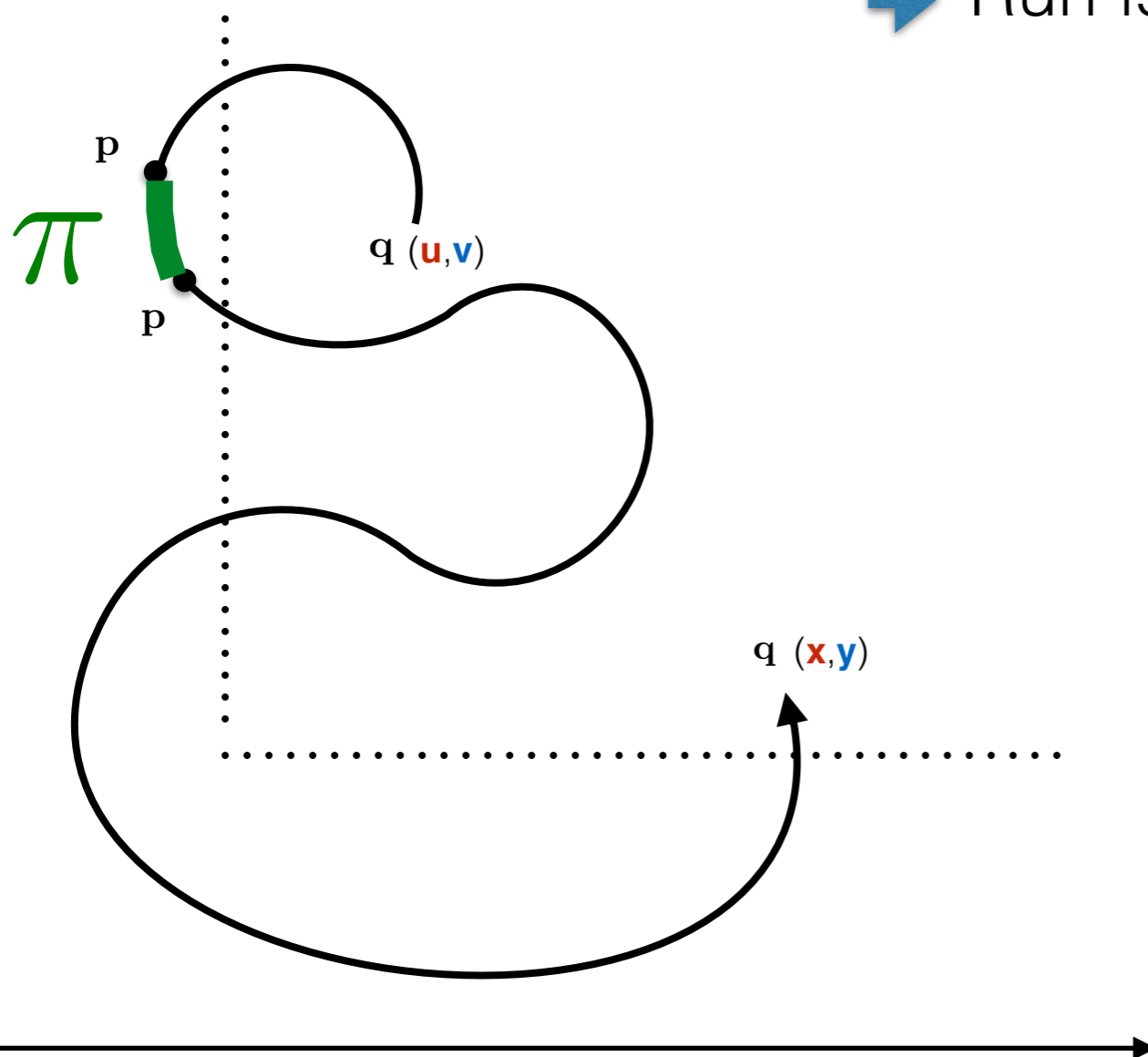


# Type 1 runs

Starting and ending **sufficiently large** in **same** control state  $q$

Idea:

➔ Run is a decomposition of small cycles

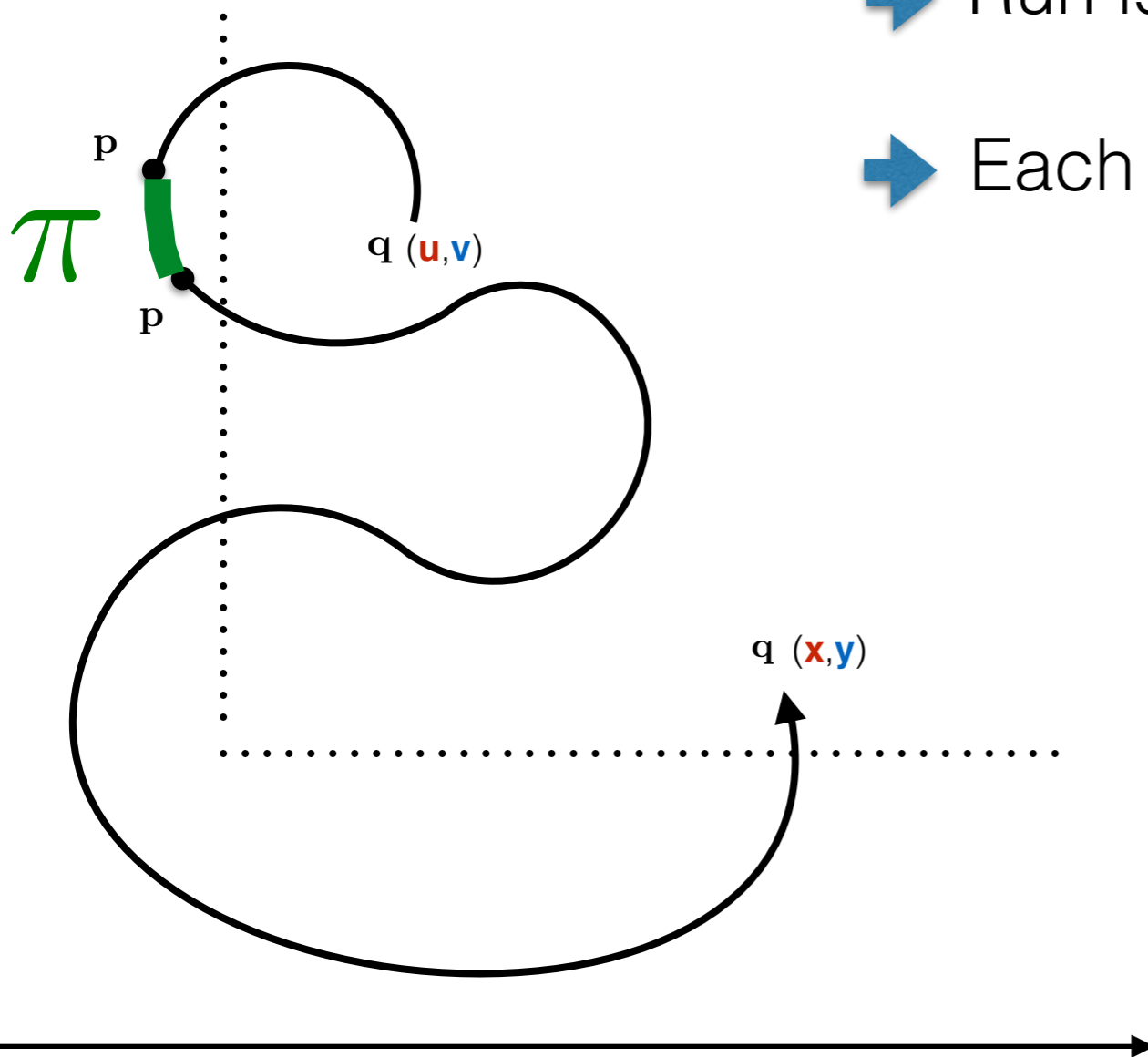


# Type 1 runs

Starting and ending **sufficiently large** in **same** control state  $q$

Idea:

- ➔ Run is a decomposition of small cycles
- ➔ Each cycle  $\pi$  has an **effect**  $\delta(\pi) \in \mathbb{Z}^2$



# Type 1 runs

Starting and ending **sufficiently large** in **same** control state  $q$

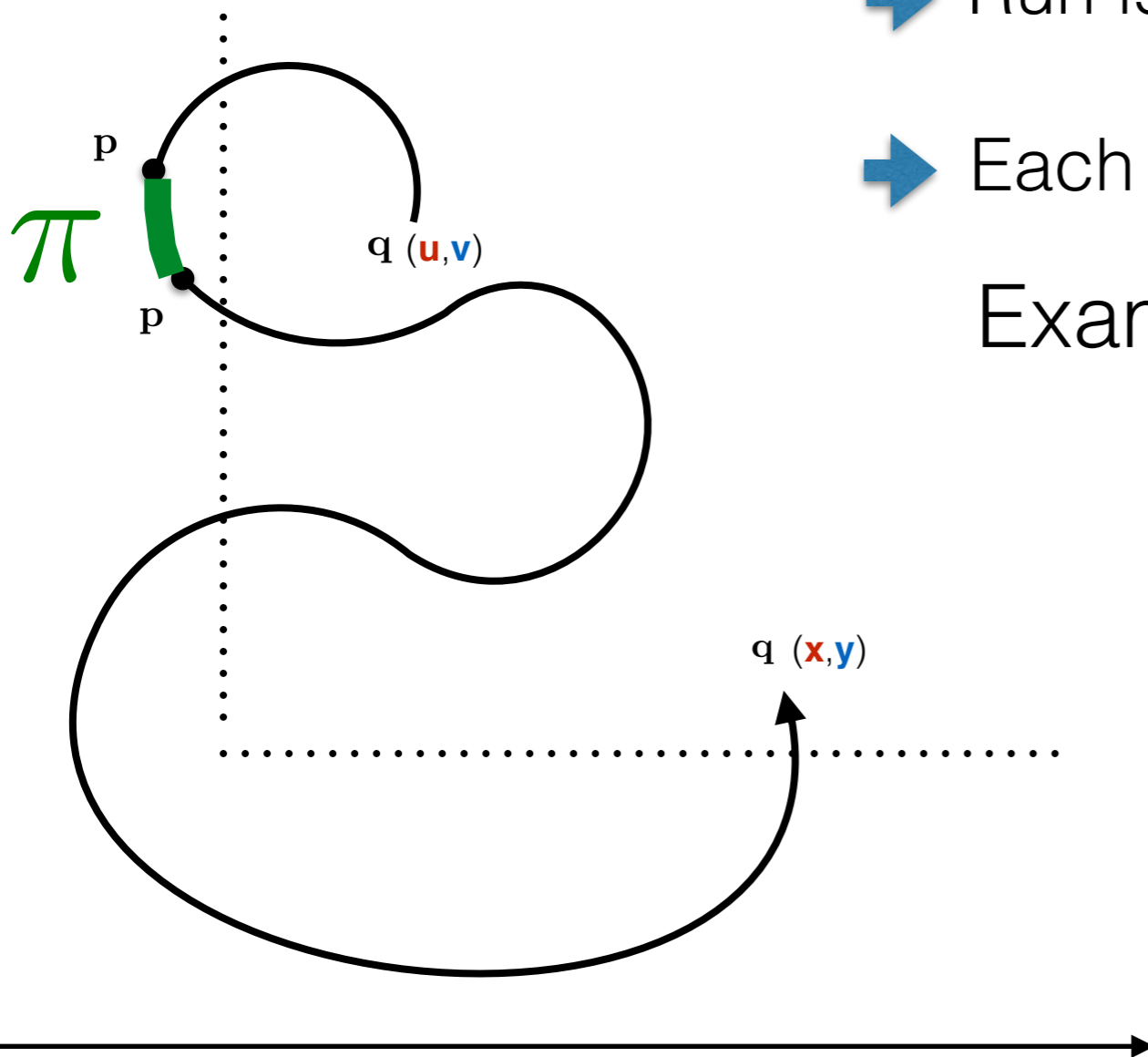
Idea:

- ➔ Run is a decomposition of small cycles
- ➔ Each cycle  $\pi$  has an **effect**  $\delta(\pi) \in \mathbb{Z}^2$

Example:

$$\delta(\pi) \in \mathbb{N} \times \mathbb{N}^-$$

“points in the right direction”





# Type 1 runs

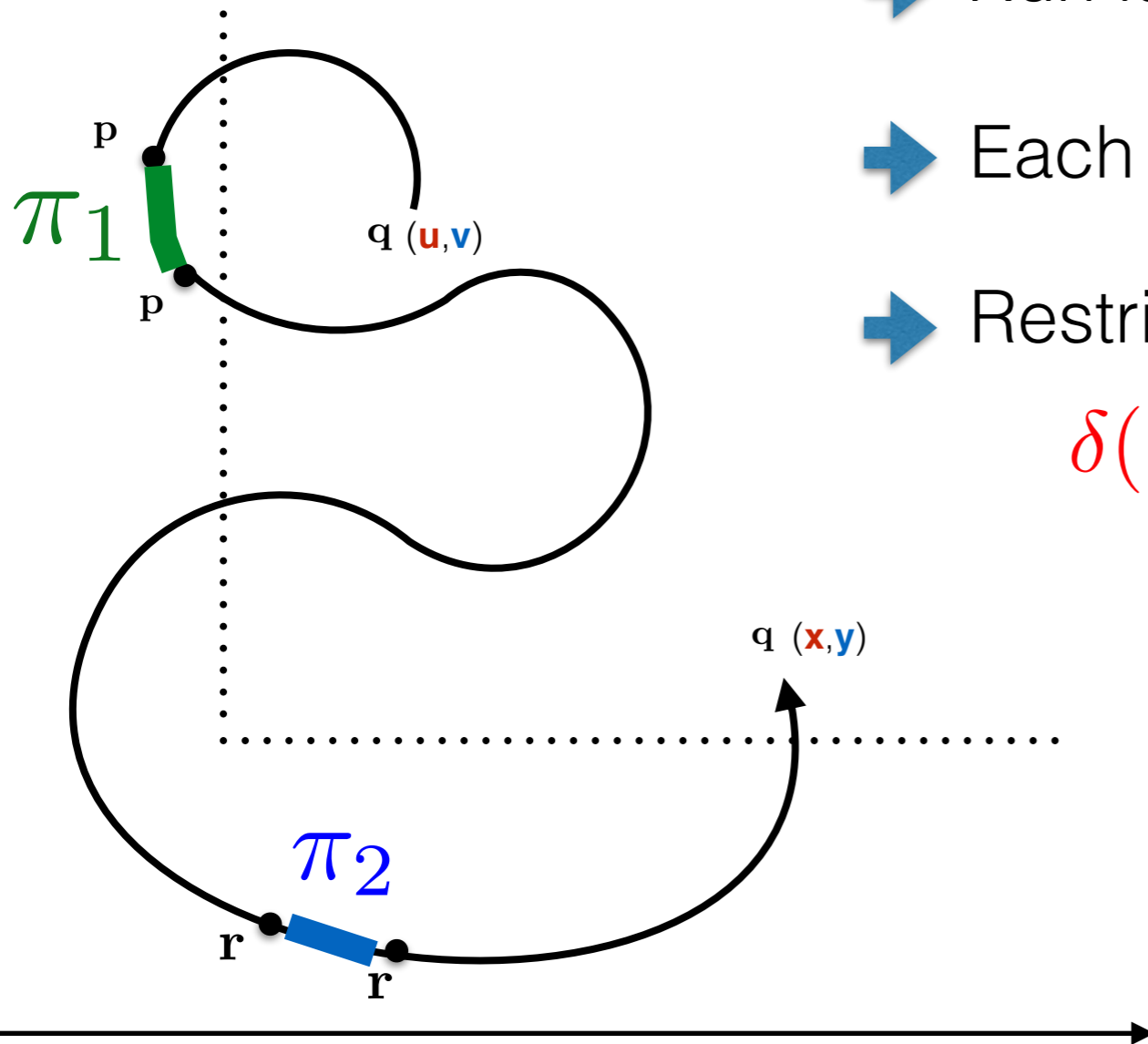
Starting and ending **sufficiently large** in **same** control state  $q$

Idea:

- ➔ Run is a decomposition of small cycles
- ➔ Each cycle  $\pi$  has an **effect**  $\delta(\pi) \in \mathbb{Z}^2$
- ➔ Restrict to **two** cycles  $\pi_1$  and  $\pi_2$

$$\delta(\pi_1), \delta(\pi_2) \in \mathbb{N} \times \mathbb{N}^-$$

“that point in the right direction”



# Type 1 runs

Starting and ending **sufficiently large** in **same** control state  $q$

Idea:

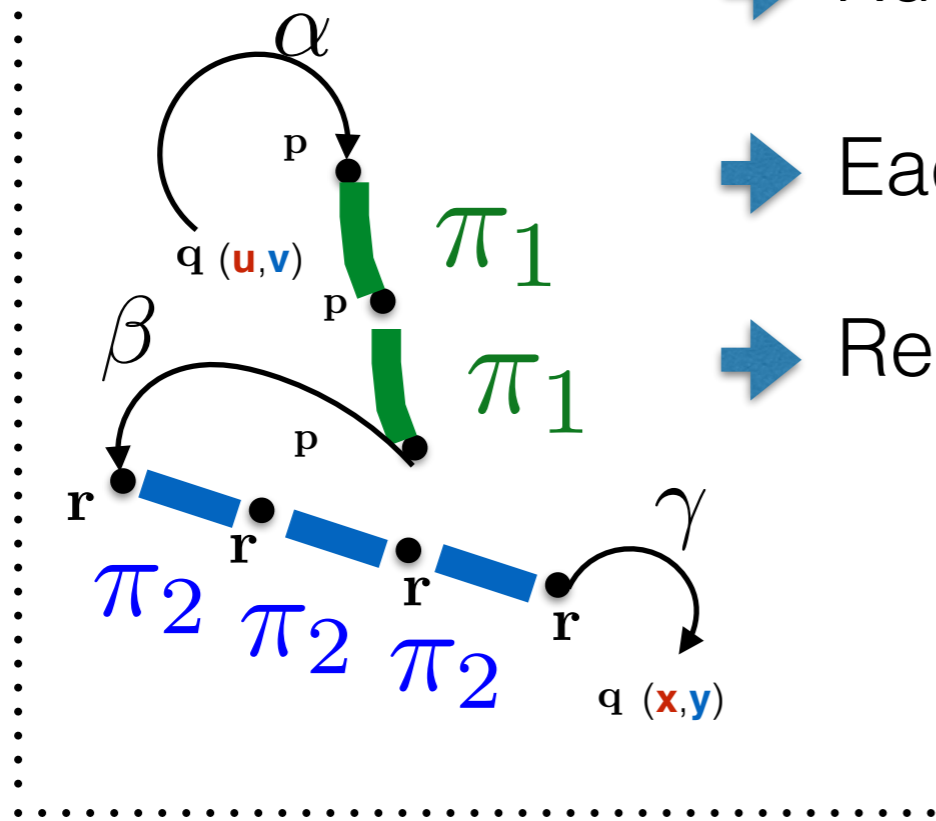
➔ Run is a decomposition of small cycles

➔ Each cycle  $\pi$  has an **effect**  $\delta(\pi) \in \mathbb{Z}^2$

➔ Restrict to **two** cycles  $\pi_1$  and  $\pi_2$

$$\delta(\pi_1), \delta(\pi_2) \in \mathbb{N} \times \mathbb{N}^-$$

“that point in the right direction”



# Type 1 runs

Starting and ending **sufficiently large** in **same** control state  $q$

Idea:

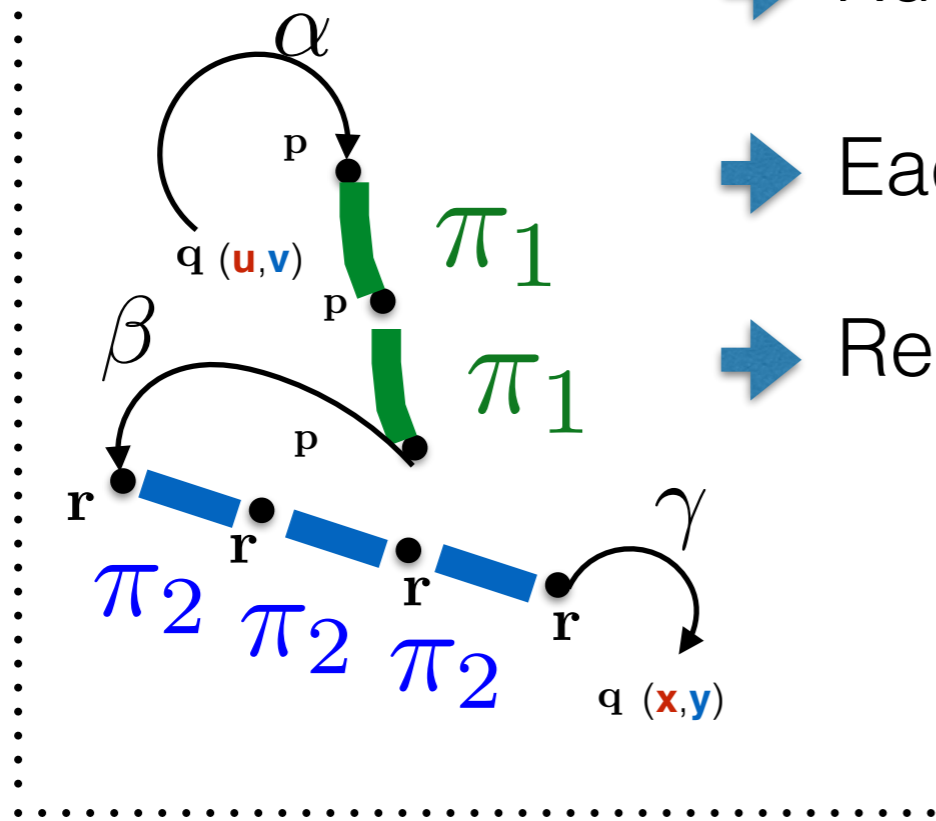
➔ Run is a decomposition of small cycles

➔ Each cycle  $\pi$  has an **effect**  $\delta(\pi) \in \mathbb{Z}^2$

➔ Restrict to **two** cycles  $\pi_1$  and  $\pi_2$

$$\delta(\pi_1), \delta(\pi_2) \in \mathbb{N} \times \mathbb{N}^-$$

“that point in the right direction”

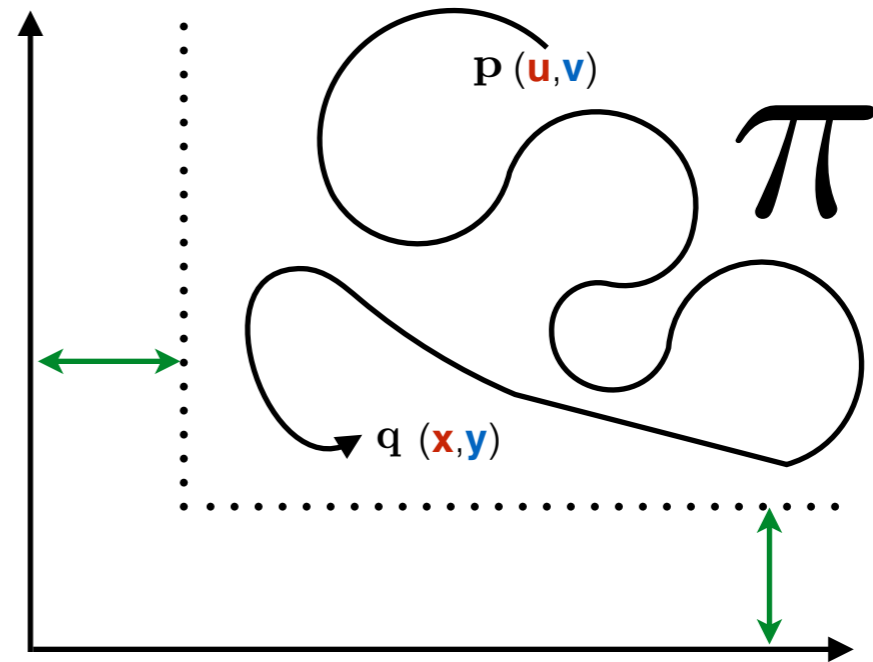


➔ Use bounded language small

$$\alpha \pi_1^* \beta \pi_2^* \gamma$$

# Type 2 runs

Staying **sufficiently large** all the time



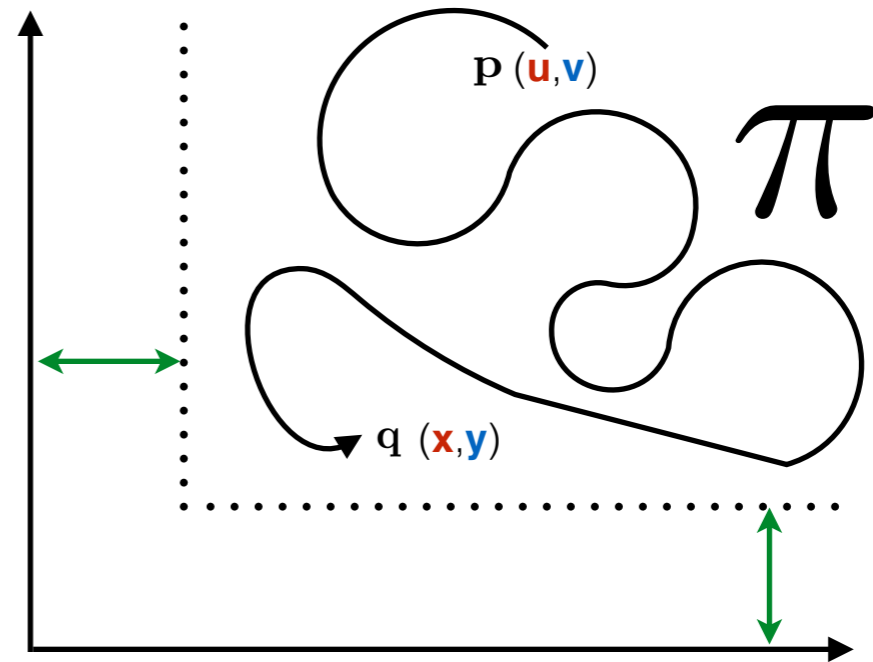
# Type 2 runs

Staying **sufficiently large** all the time

Idea:

➔ Each run  $\pi$  can be factorized as

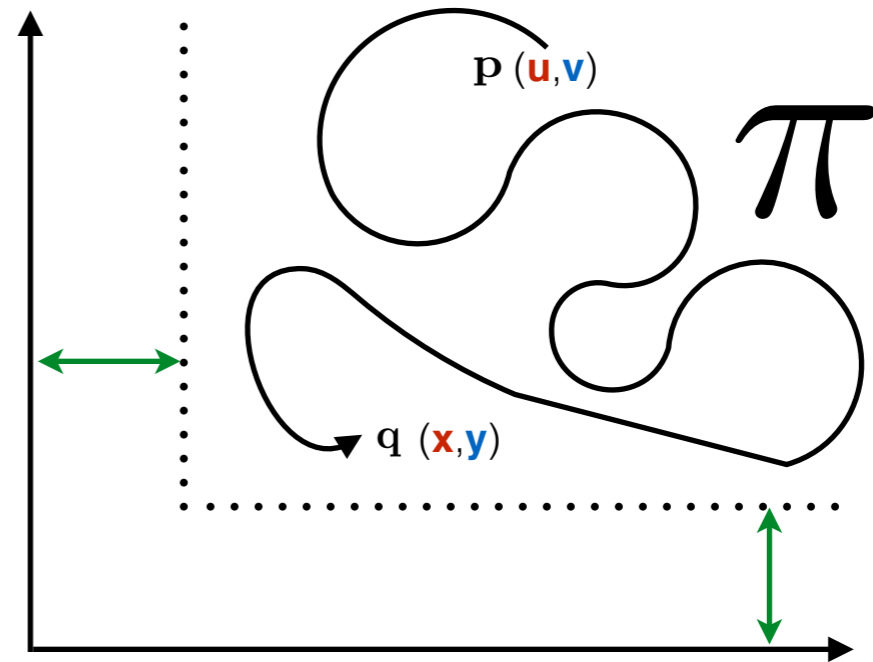
$$\pi = \alpha_0 \beta_1 \alpha_1 \cdots \beta_\ell \alpha_\ell$$



# Type 2 runs

Staying **sufficiently large** all the time

Idea:

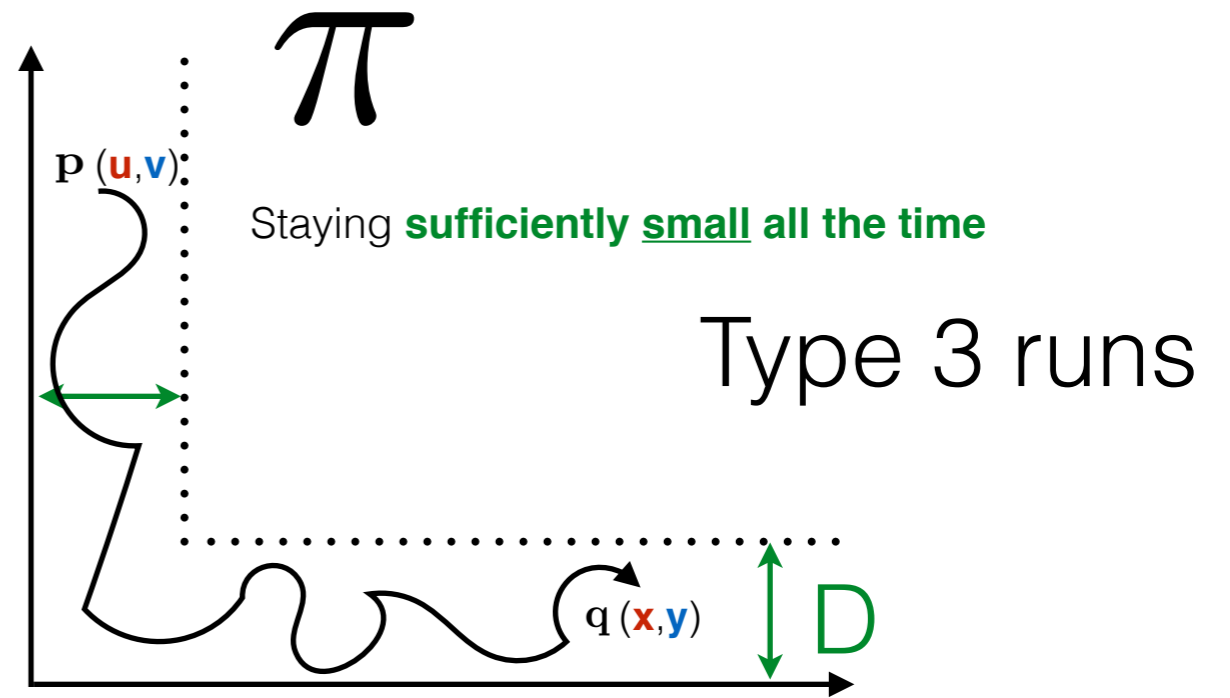


➔ Each run  $\pi$  can be factorized as

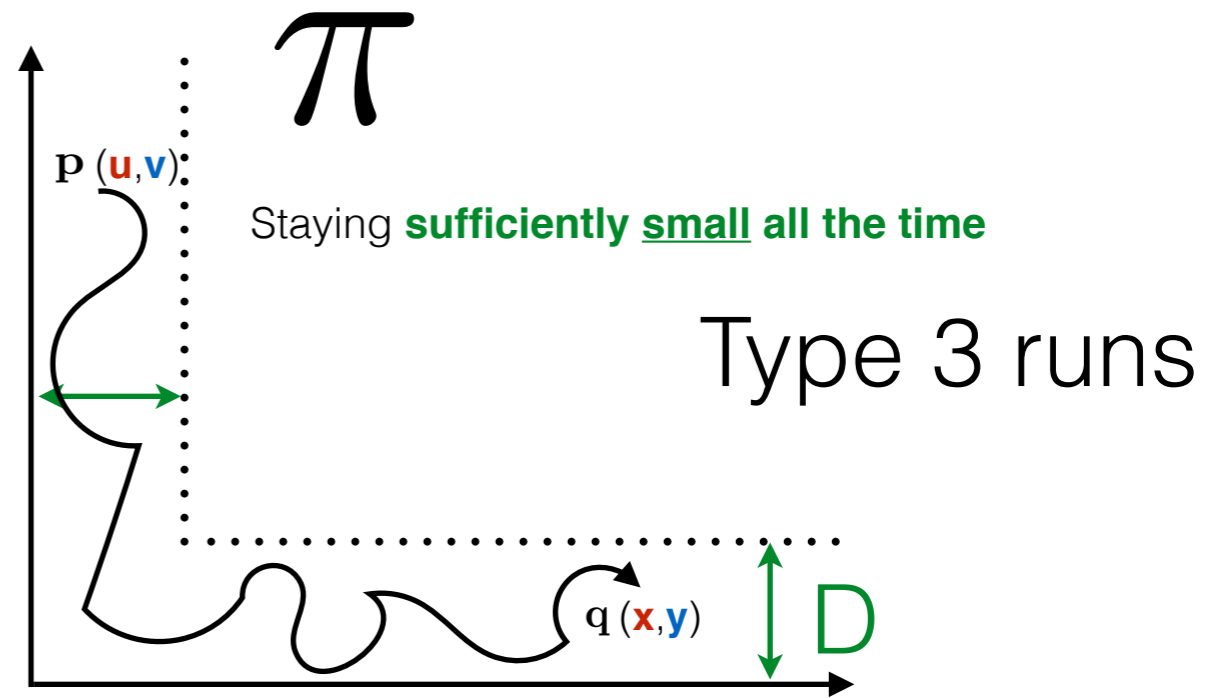
$$\pi = \alpha_0 \beta_1 \alpha_1 \cdots \beta_\ell \alpha_\ell \quad \ell \leq |Q|$$

cycles starting and ending sufficiently large (i.e. Type 1)

# Type 3 runs



# Type 3 runs

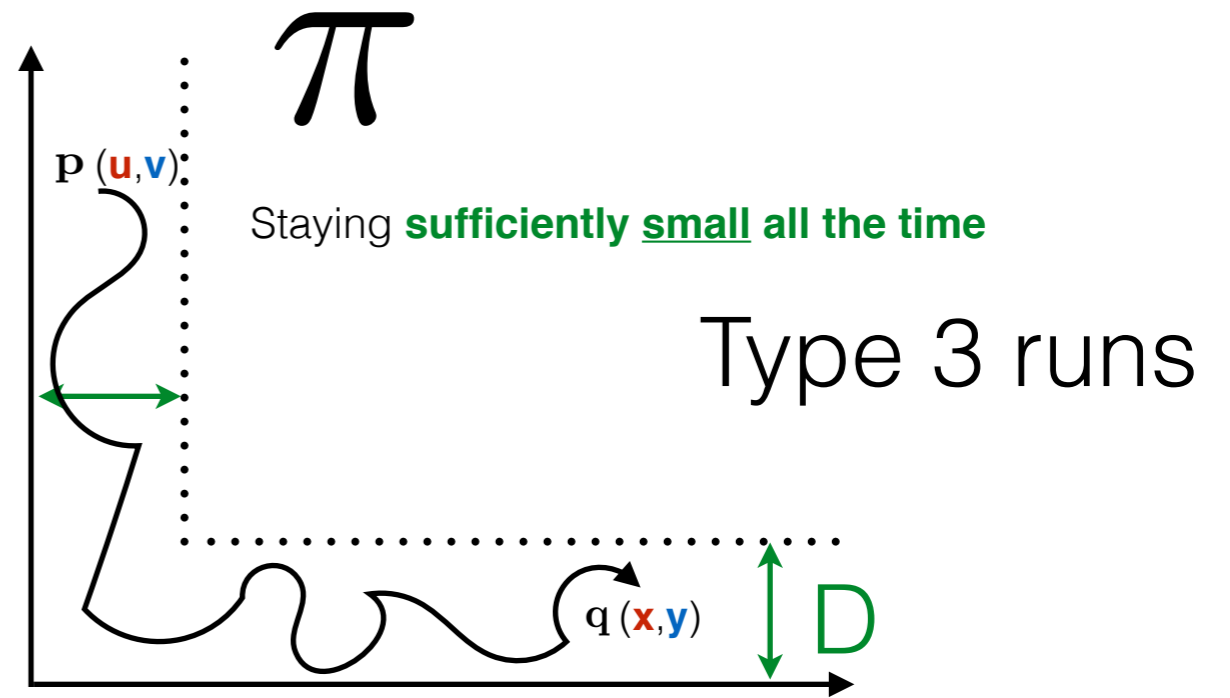


Idea:

➔ Treat each run  $\pi$  as a run in a big 1-VASS (with  $|Q| \cdot D$  states)



# Type 3 runs



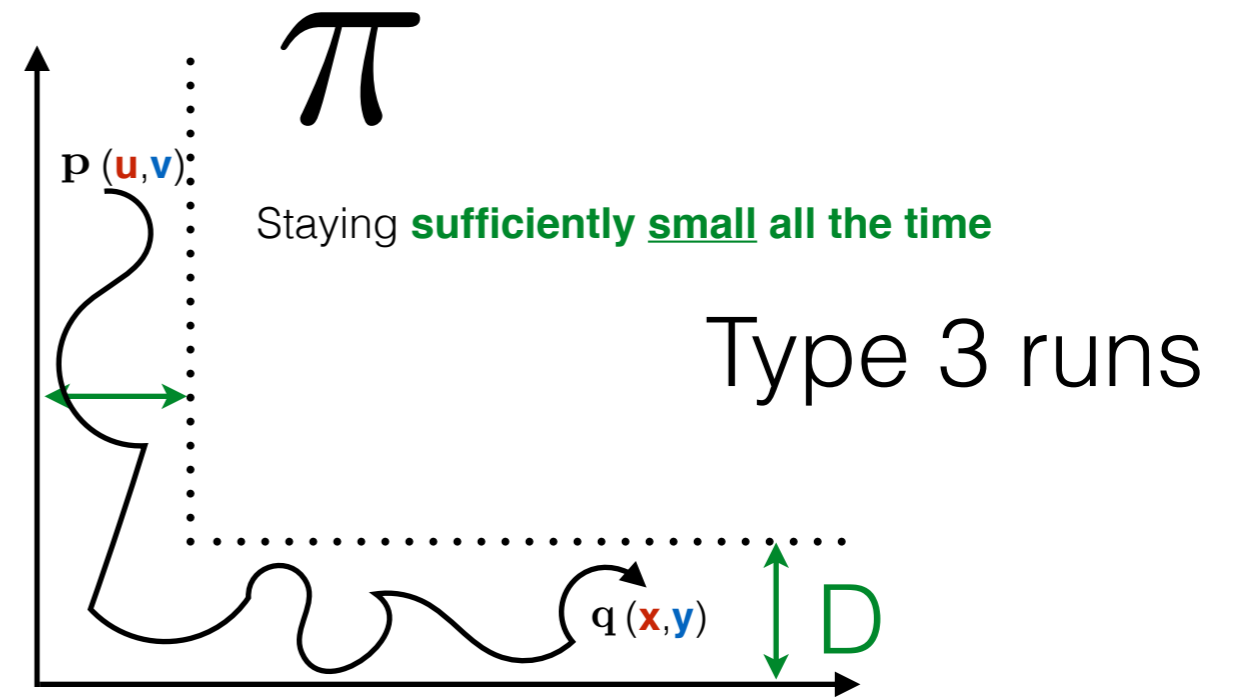
Idea:

➔ Treat each run  $\pi$  as a run in a big 1-VASS (with  $|Q| \cdot D$  states)

➔ Run  $\pi$  can be factorized as

$$\pi = \alpha \pi_1^* \beta \pi_2^* \gamma$$

# Type 3 runs



Idea:

➔ Treat each run  $\pi$  as a run in a big 1-VASS (with  $|Q| \cdot D$  states)

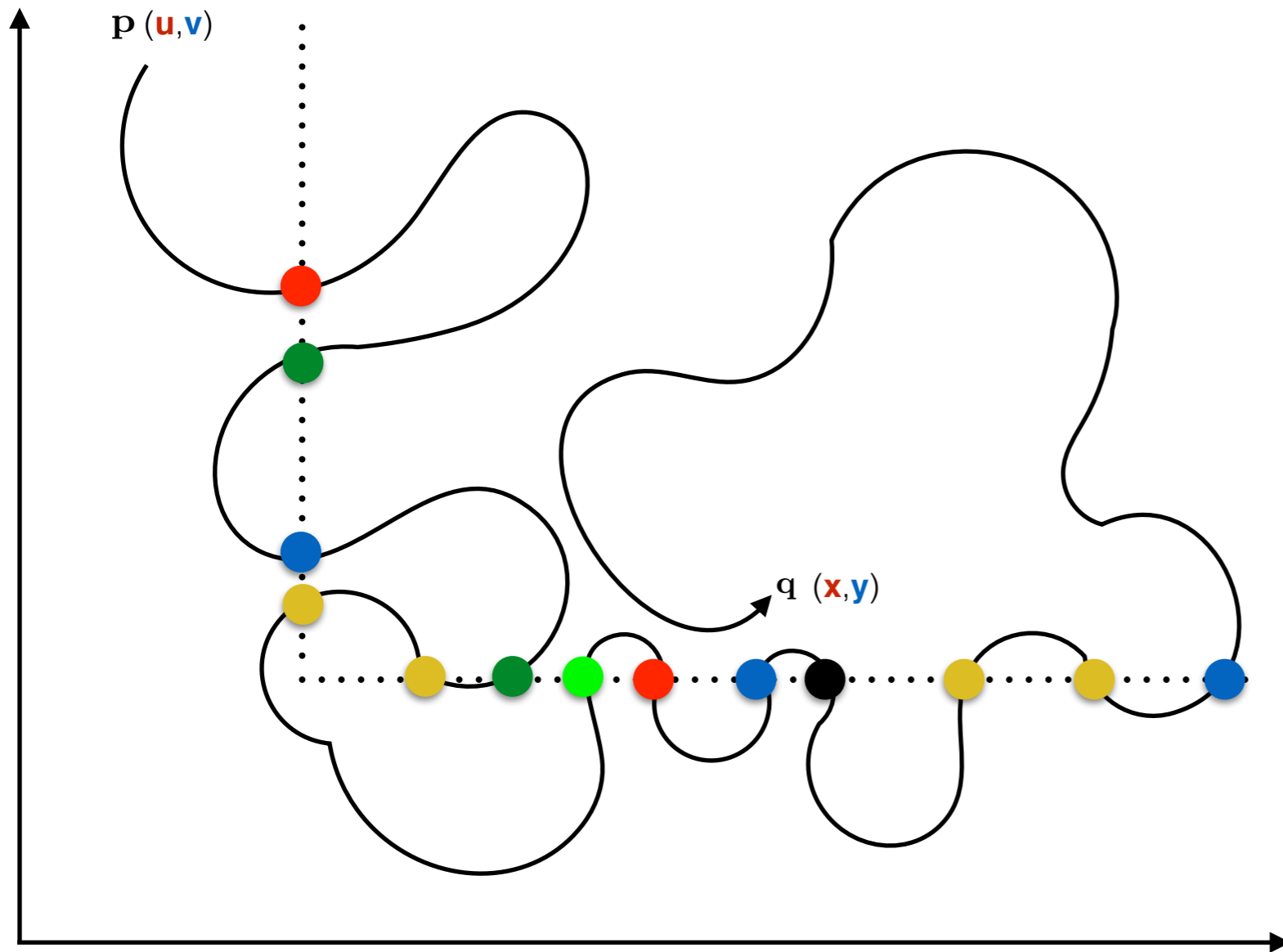
➔ Run  $\pi$  can be factorized as

$$\pi = \alpha \pi_1^* \beta \pi_2^* \gamma$$

2 possible cases:

- $\alpha$  **down\***  $\beta$  **right\***  $\gamma$
- $\alpha$  **left\***  $\beta$  **up\***  $\gamma$

# Factorizing arbitrary runs

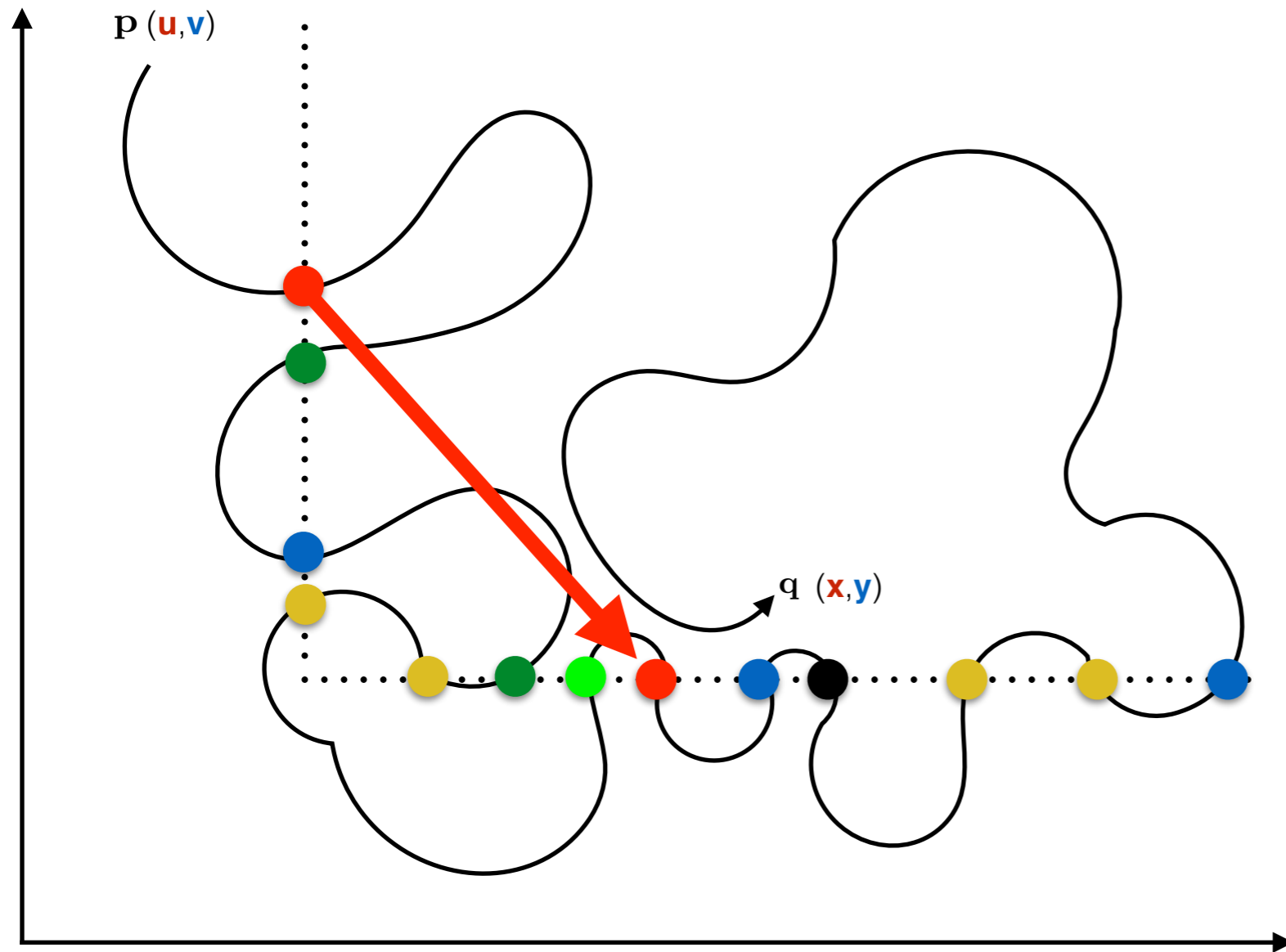


Idea:

Color each border position with its control state



# Factorizing arbitrary runs



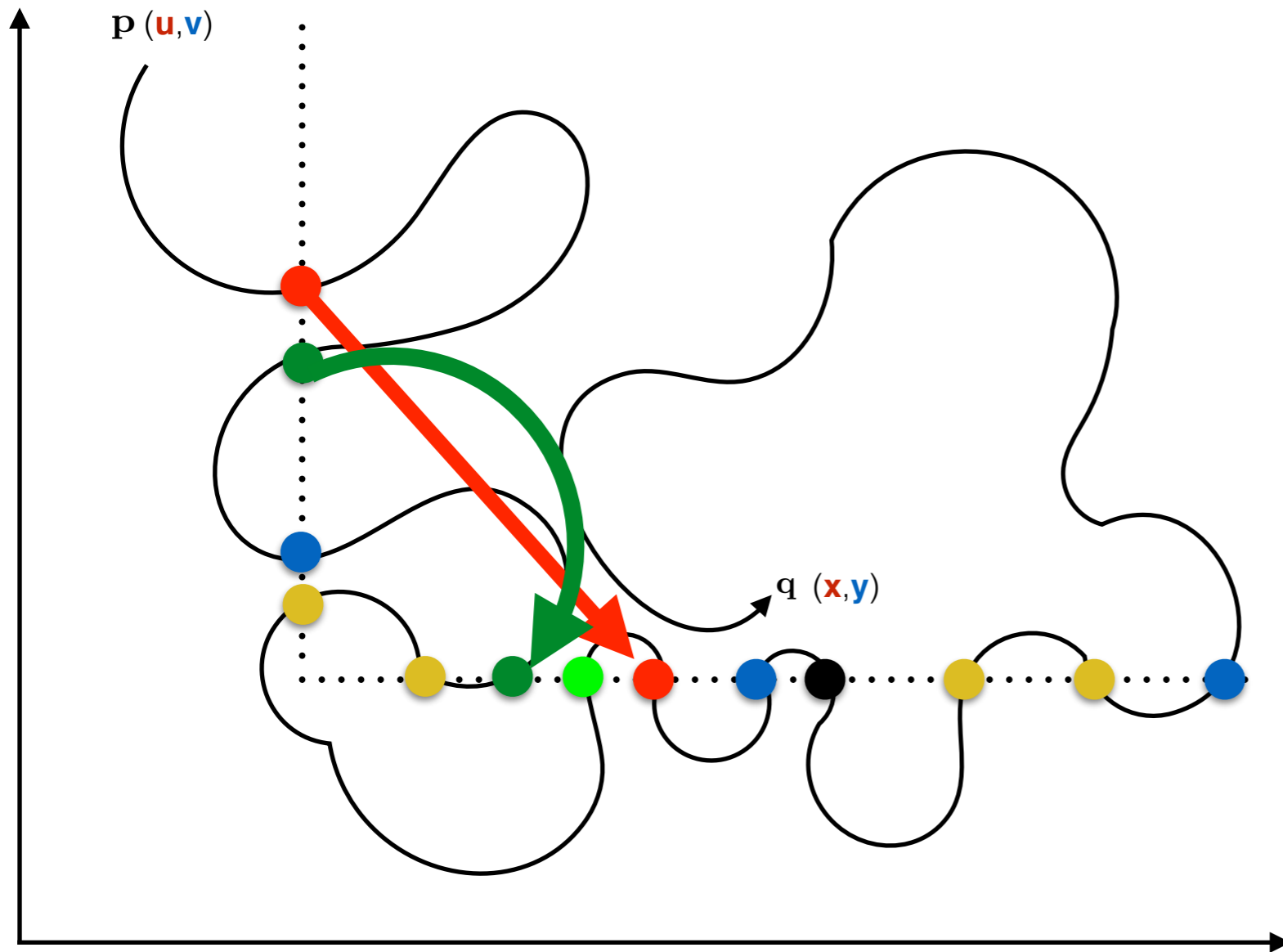
**Type 1 run!**

Idea:

Color each border position with its control state



# Factorizing arbitrary runs



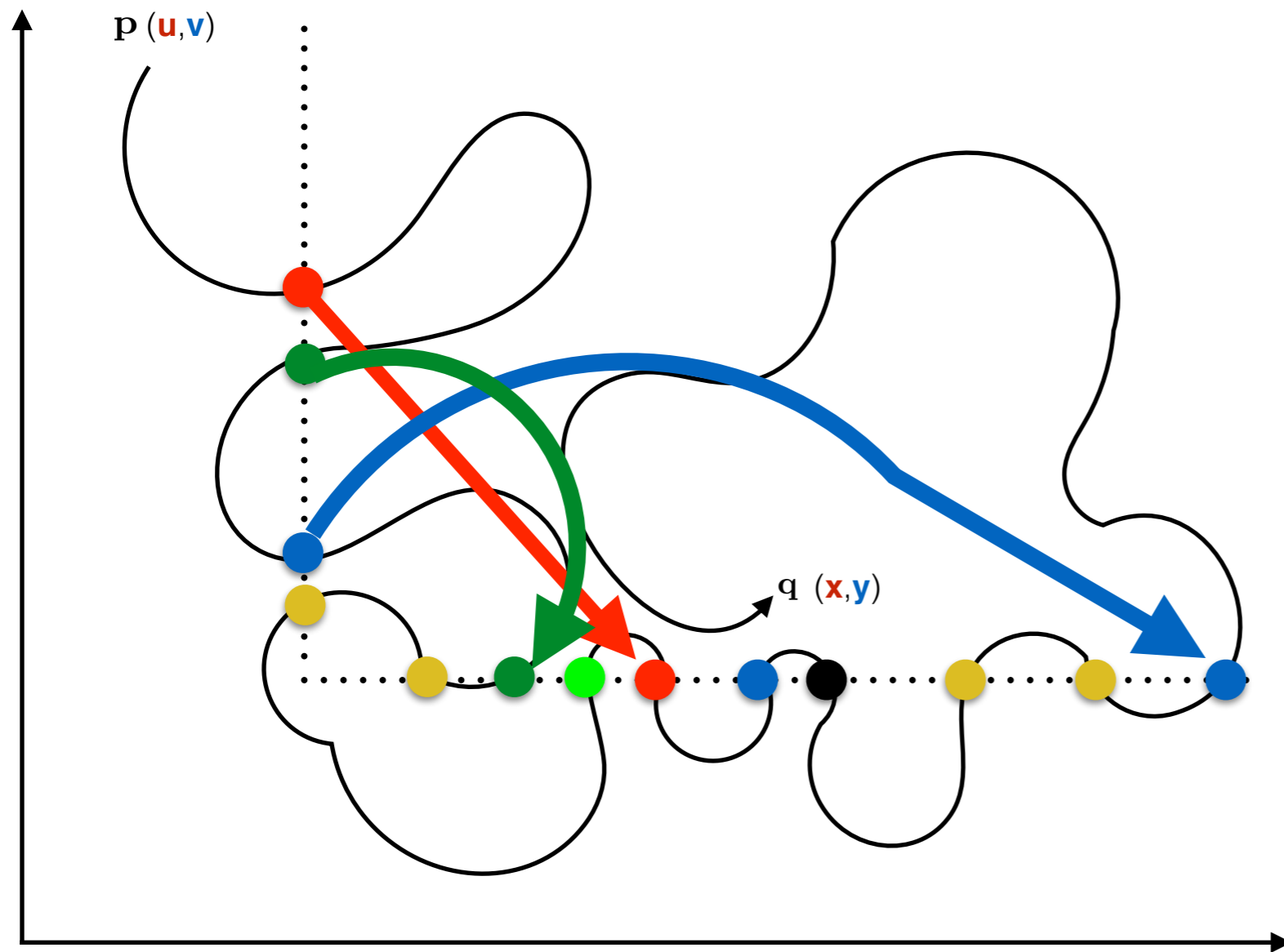
Type 1 run!

Idea:

Color each border position with its control state



# Factorizing arbitrary runs



**Type 1 run!**

**Type 1 run!**

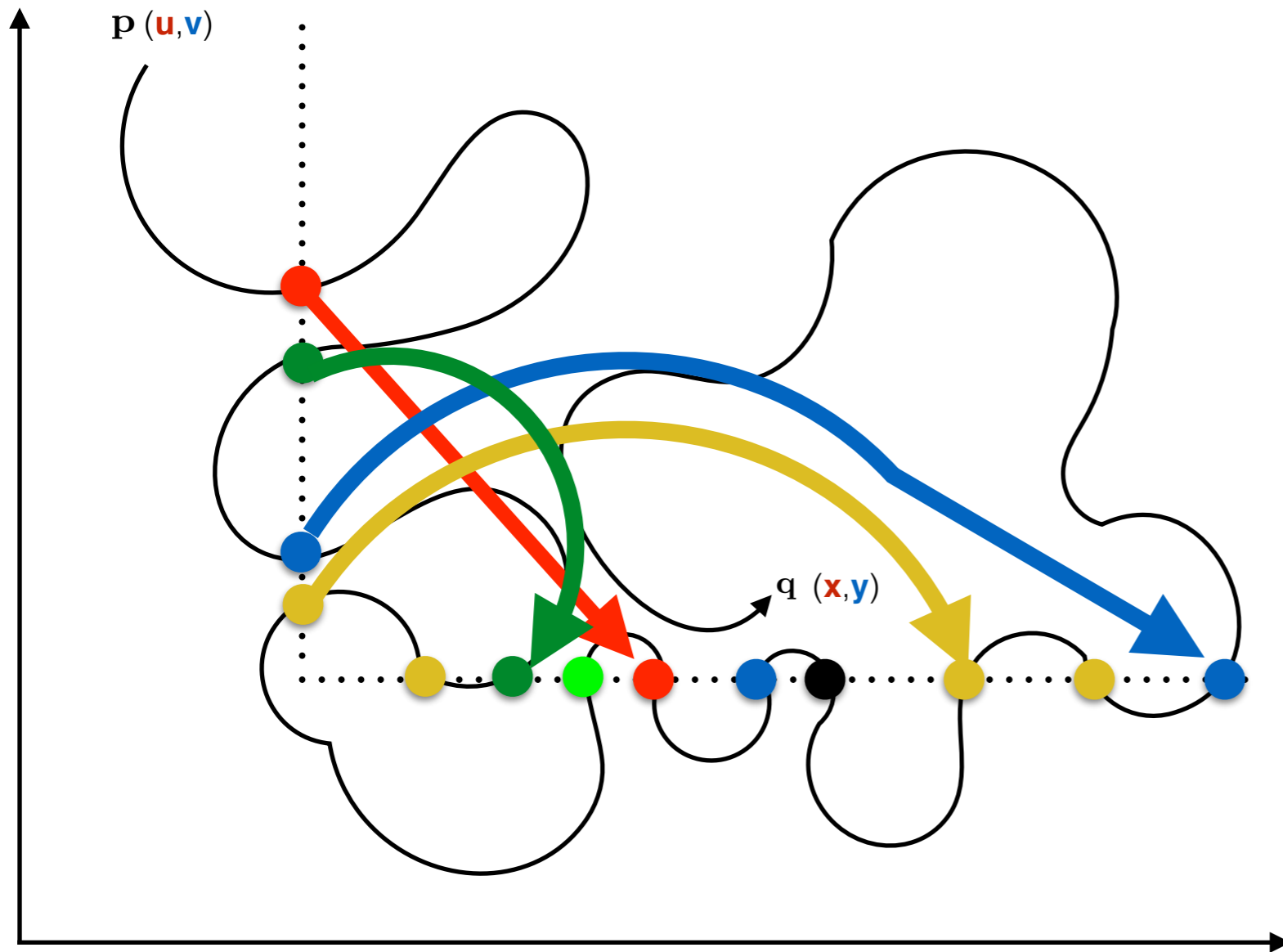
**Type 1 run!**

Idea:

Color each border position with its control state



# Factorizing arbitrary runs



**Type 1 run!**

**Type 1 run!**

**Type 1 run!**

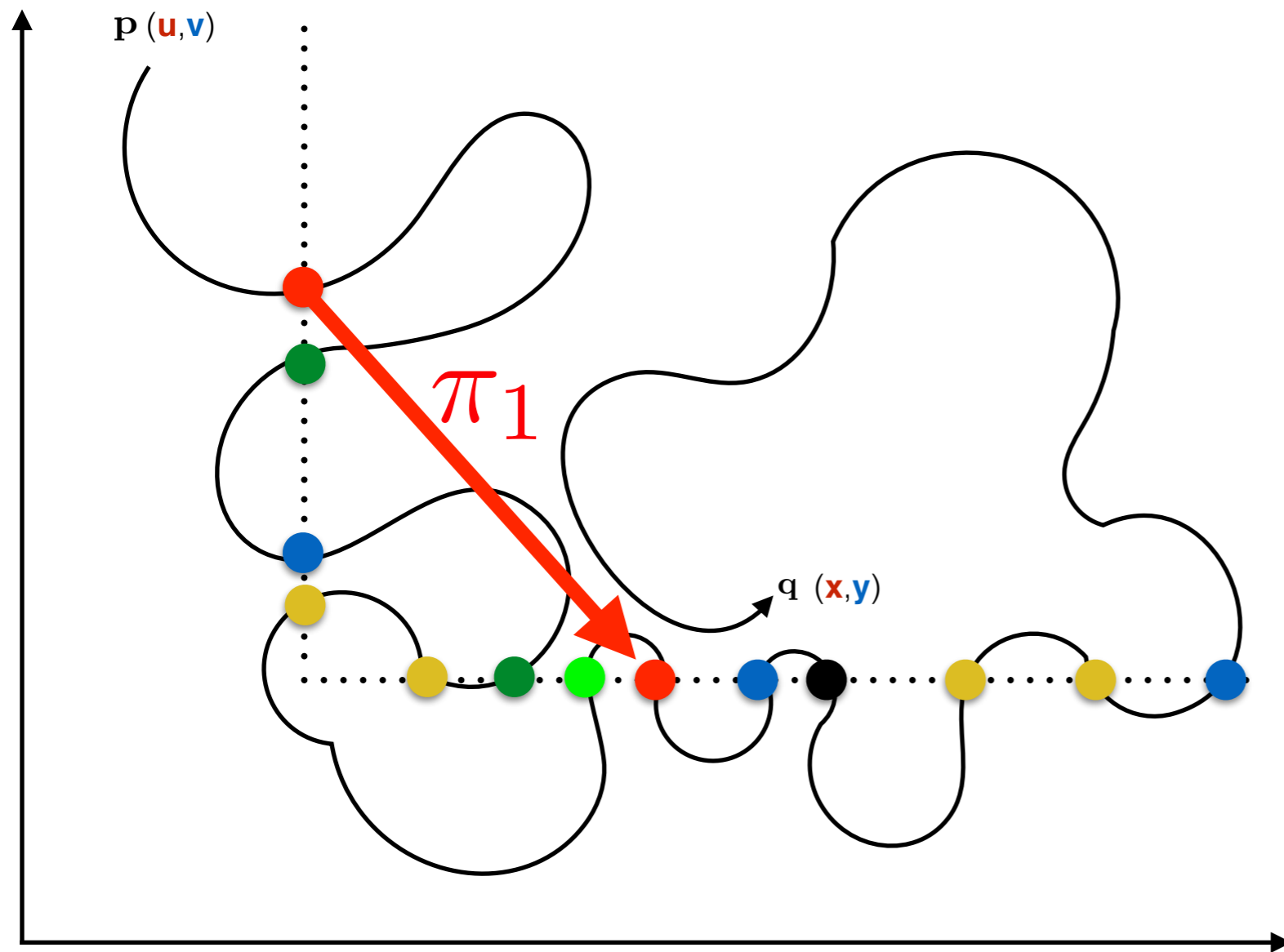
**Type 1 run!**

Idea:

Color each border position with its control state



# Factorizing arbitrary runs



$\pi_1$  Type 1 run!

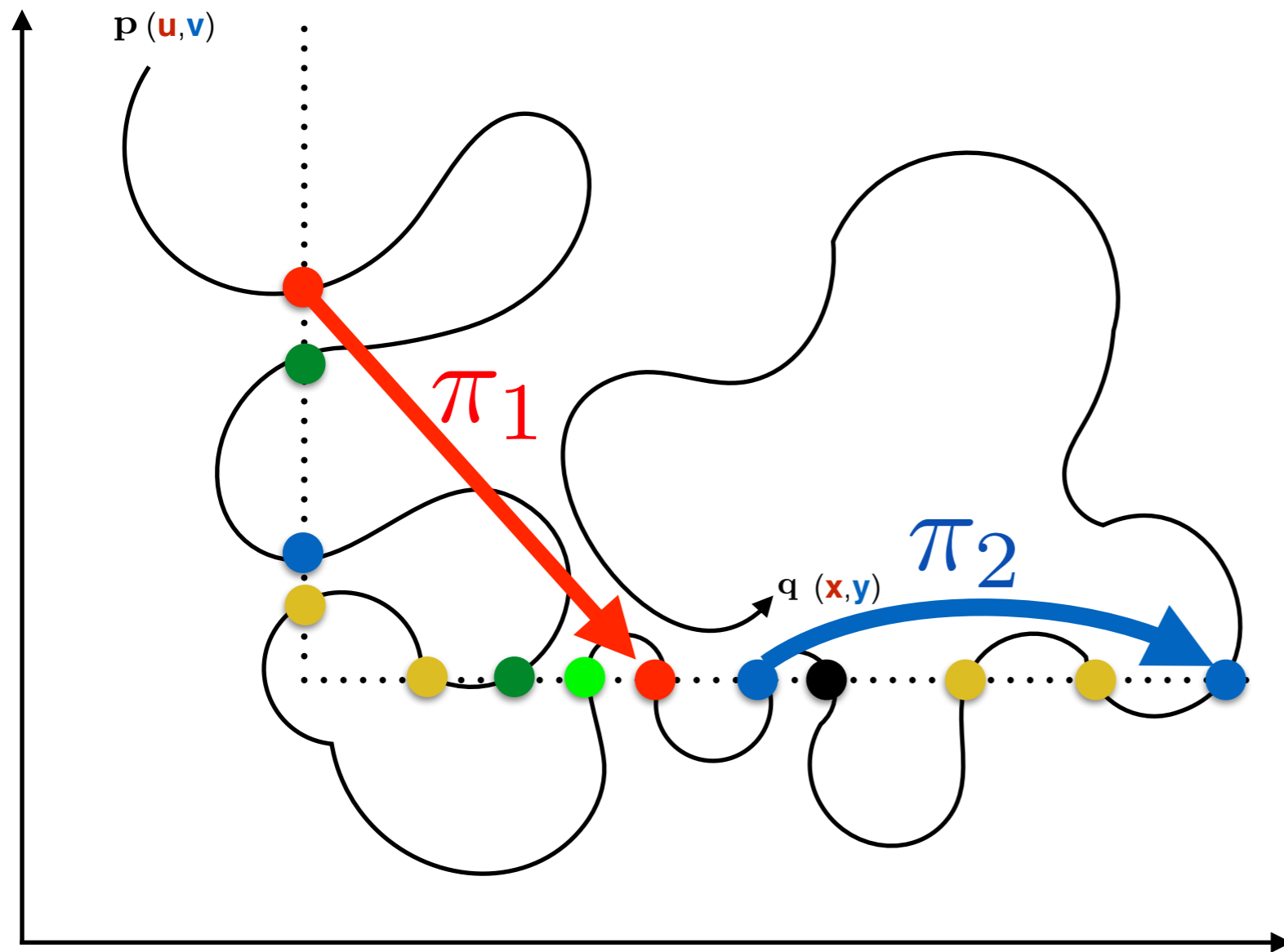
Idea:

Color each border position with its control state





# Factorizing arbitrary runs



$\pi_1$  **Type 1 run!**

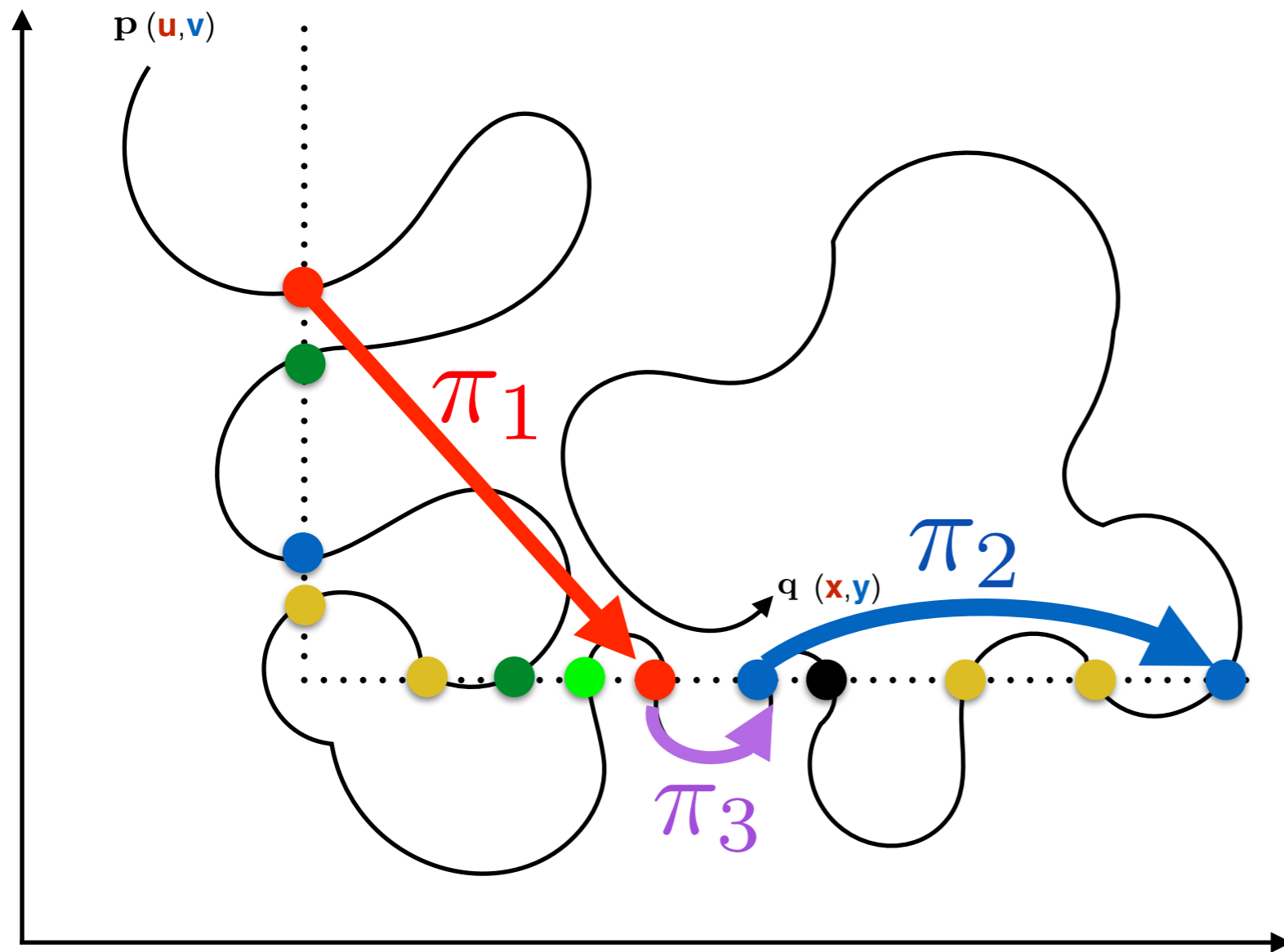
$\pi_2$  **Type 1 run!**

Idea:

Color each border position with its control state



# Factorizing arbitrary runs



$\pi_1$  **Type 1 run!**

$\pi_2$  **Type 1 run!**

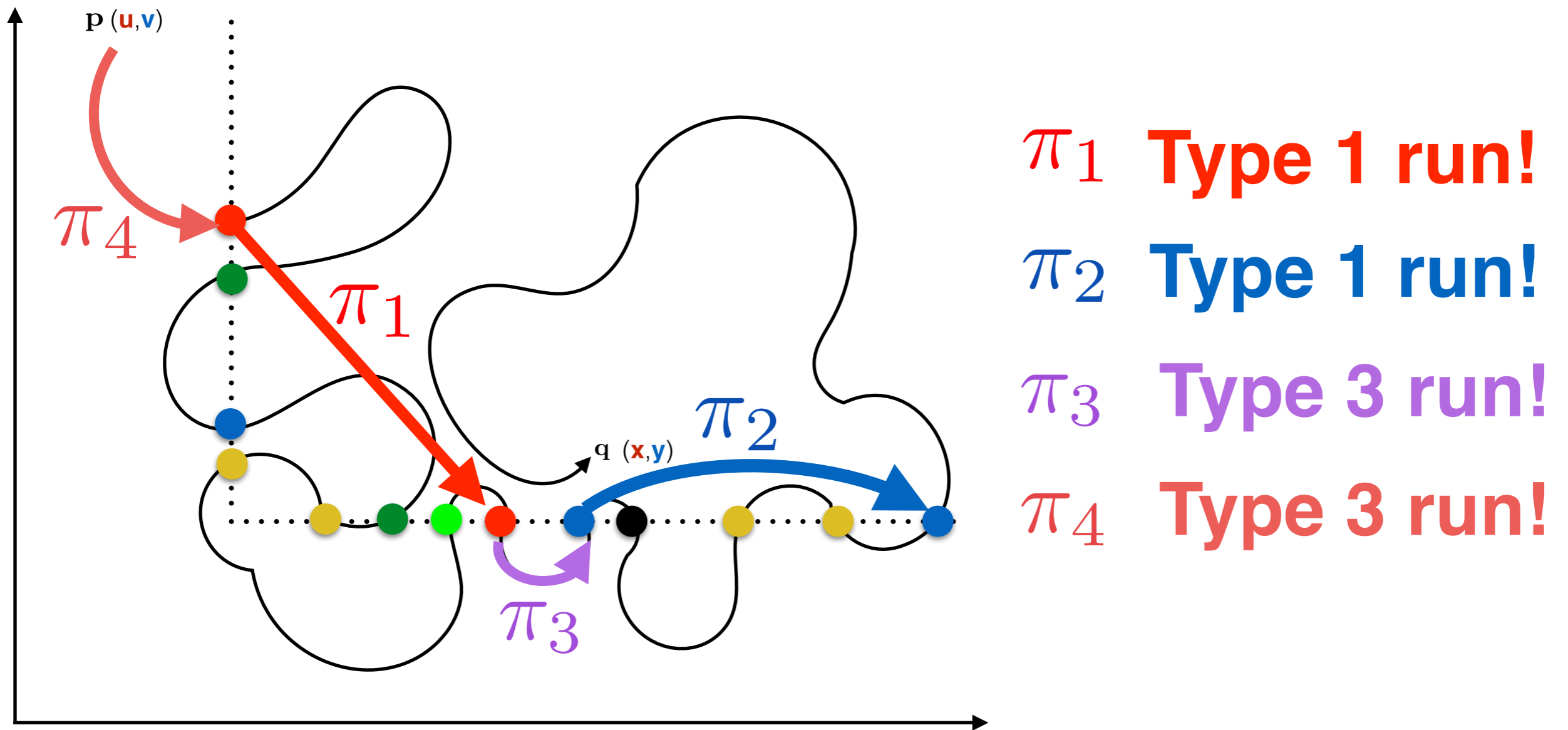
$\pi_3$  **Type 3 run!**

Idea:

Color each border position with its control state



# Factorizing arbitrary runs

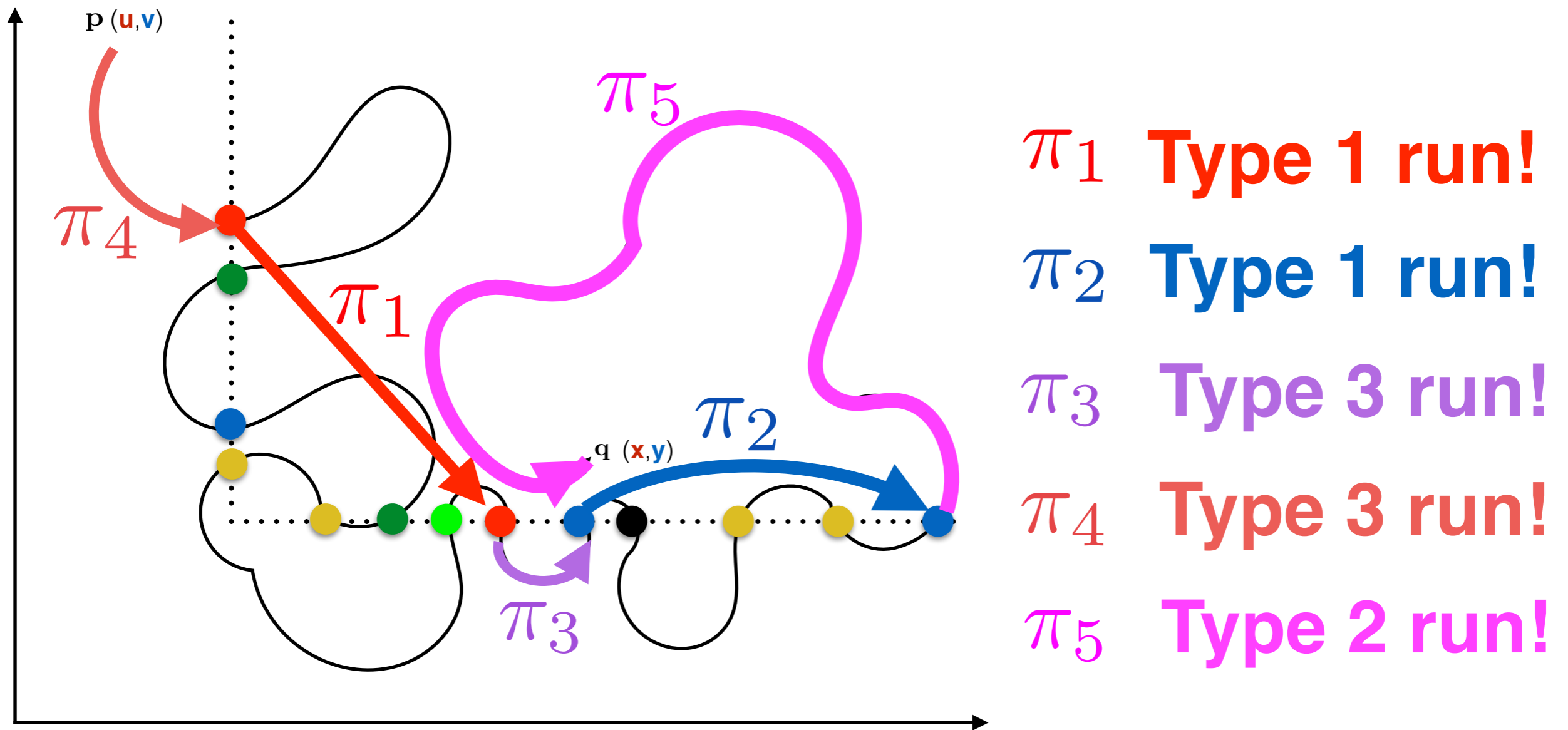


Idea:

Color each border position with its control state



# Factorizing arbitrary runs

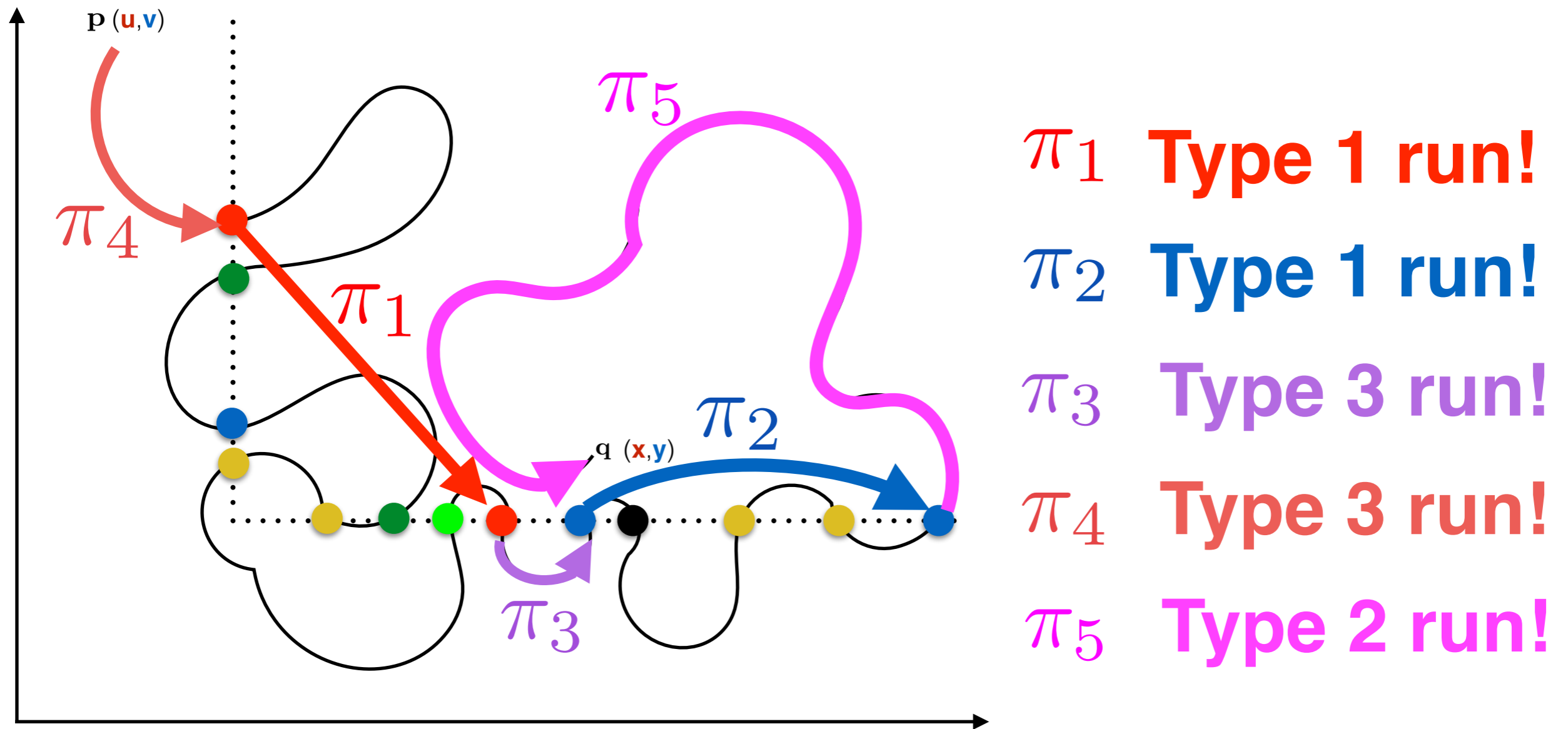


Idea:

Color each border position with its control state



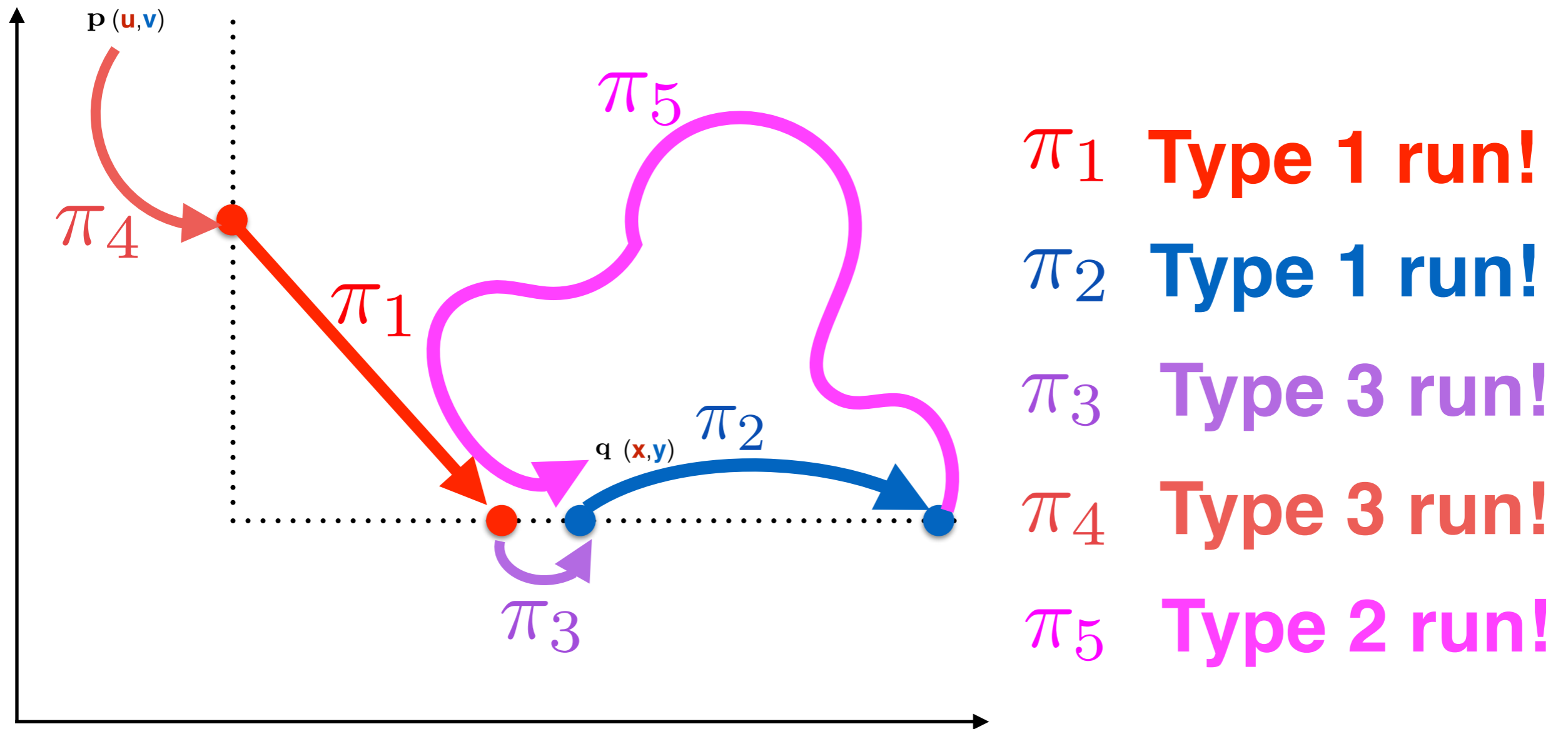
# Factorizing arbitrary runs



Small factorization:

$\pi_4 \pi_1 \pi_3 \pi_2 \pi_5$

# Factorizing arbitrary runs



Small factorization:

$\pi_4$   $\pi_1$   $\pi_3$   $\pi_2$   $\pi_5$

# Our main result

The following is computable in exponential space:

**Given:** 2-VASS  $V = (Q, T)$  .

**Output:** Finite unions of regular expressions

How to get PSPACE for reachability?

$$\bigcup_i \alpha_0 \beta_1^* \alpha_1 \cdots \beta_k^* \alpha_k$$

Length at most  $(|Q| + |T| + \|T\|)^{O(1)}$

$k \leq O(|Q|^2)$

such that

$$p(u_1, u_2) \longrightarrow^* q(v_1, v_2)$$

if, and only, if

$$\exists e_1, \dots, e_k \in \mathbb{N} : p(u_1, u_2) \xrightarrow{\alpha_0 \beta_1^{e_1} \alpha_1 \cdots \beta_k^{e_k} \alpha_k} q(v_1, v_2)$$

bounded by  $(|Q| + |T| + \|T\|)^{O(1)}$

“Reachability is witnessed by a small bounded language”

# Our main result

The following is computable in exponential space:

**Given:** 2-VASS  $V = (Q, T)$  .

**Output:** Finite unions of regular expressions

$$\bigcup_i \alpha_0 \beta_1^* \alpha_1 \cdots \beta_k^* \alpha_k$$

How to get PSPACE for reachability?

Length at most

$$(|Q| + |T| + \|T\|)^{O(1)}$$

$$k \leq O(|Q|^2)$$

such that

$$p(u_1, u_2) \longrightarrow^* q(v_1, v_2)$$

if, and only, if

$$\exists e_1, \dots, e_k \in \mathbb{N} : p(u_1, u_2) \xrightarrow{\alpha_0 \beta_1^{e_1} \alpha_1 \cdots \beta_k^{e_k} \alpha_k} q(v_1, v_2)$$

bounded by  $(|Q| + |T| + \|T\|)^{O(1)}$

Why?

“Reachability is witnessed by a small bounded language”



# Reachability is in PSPACE

$$\bigcup_i \alpha_0 \beta_1^* \alpha_1 \cdots \beta_k^* \alpha_k$$

Length at most  $(|Q| + |T| + \|T\|)^{O(1)}$

$k \leq O(|Q|^2)$

# Reachability is in PSPACE

$$\bigcup_i \alpha_0 \beta_1^* \alpha_1 \cdots \beta_k^* \alpha_k$$

Length at most  $(|Q| + |T| + \|T\|)^{O(1)}$

$k \leq O(|Q|^2)$

1) For each  $\alpha_0 \beta_1^* \alpha_1 \cdots \beta_k^* \alpha_k$  construct a matrix  $A \in \mathbb{Z}^{n \times k}$  and a vector  $\vec{b} \in \mathbb{Z}^n$  such that **solutions**  $(e_1, \dots, e_k)$  to the system of linear Diophantine inequalities  $Ax \geq \vec{b}$  **correspond to real runs**  $\alpha_0 \beta^{e_1} \alpha_1 \cdots \beta_k^{e_k} \alpha_k$  in the 2-VASS.

# Reachability is in PSPACE

$$\bigcup_i \alpha_0 \beta_1^* \alpha_1 \cdots \beta_k^* \alpha_k$$

Length at most  $(|Q| + |T| + \|T\|)^{O(1)}$

$k \leq O(|Q|^2)$

1) For each  $\alpha_0 \beta_1^* \alpha_1 \cdots \beta_k^* \alpha_k$  construct a matrix  $A \in \mathbb{Z}^{n \times k}$  and a vector  $\vec{b} \in \mathbb{Z}^n$  such that **solutions**  $(e_1, \dots, e_k)$  to the system of linear Diophantine inequalities  $Ax \geq \vec{b}$  **correspond to real runs**  $\alpha_0 \beta^{e_1} \alpha_1 \cdots \beta_k^{e_k} \alpha_k$  in the 2-VASS.

2) Use results from integer linear programming that solutions to  $Ax \geq \vec{b}$  are bounded by  $2^{k^{O(1)}} \cdot O(\|A\| + \|\vec{b}\|)$ .

**(Shrijver 1998)**