# Acceleration in multi pushdown systems

(TACAS'16)

Prakash Saivasan

Joint work with

Mohamed Faouzi Atig

Narayan Kumar K

# Motivation:

## Verification of concurrent programs with:

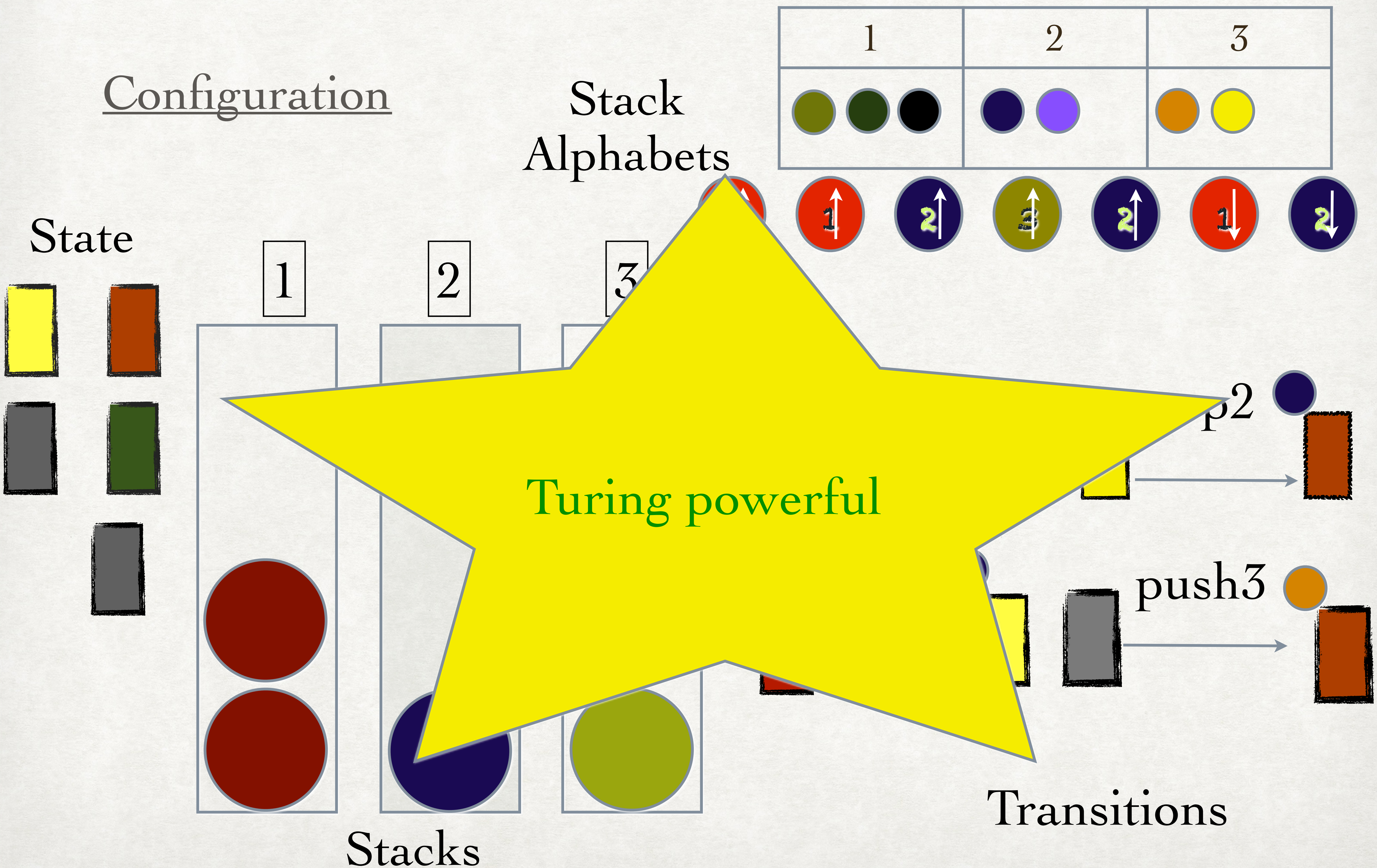
- Programs with multiple threads
- Threads can have recursion
- Finite data domain
- Shared memory

# Formal models

| Programs | Model |
| --- | --- |
| Recursive | Pushdown Systems |
| Concurrent Recursive | Multi-pushdown Systems |

# Multi-pushdown systems

# Multi pushdown Systems



# Existing underapproximations

- Bounded Context
- Bounded Phase
- Ordered MPDS
- Bounded Scope

# Bounded context


Context is a sequence of operations restricted to a stack

*S. Qadeer J. Rehof*

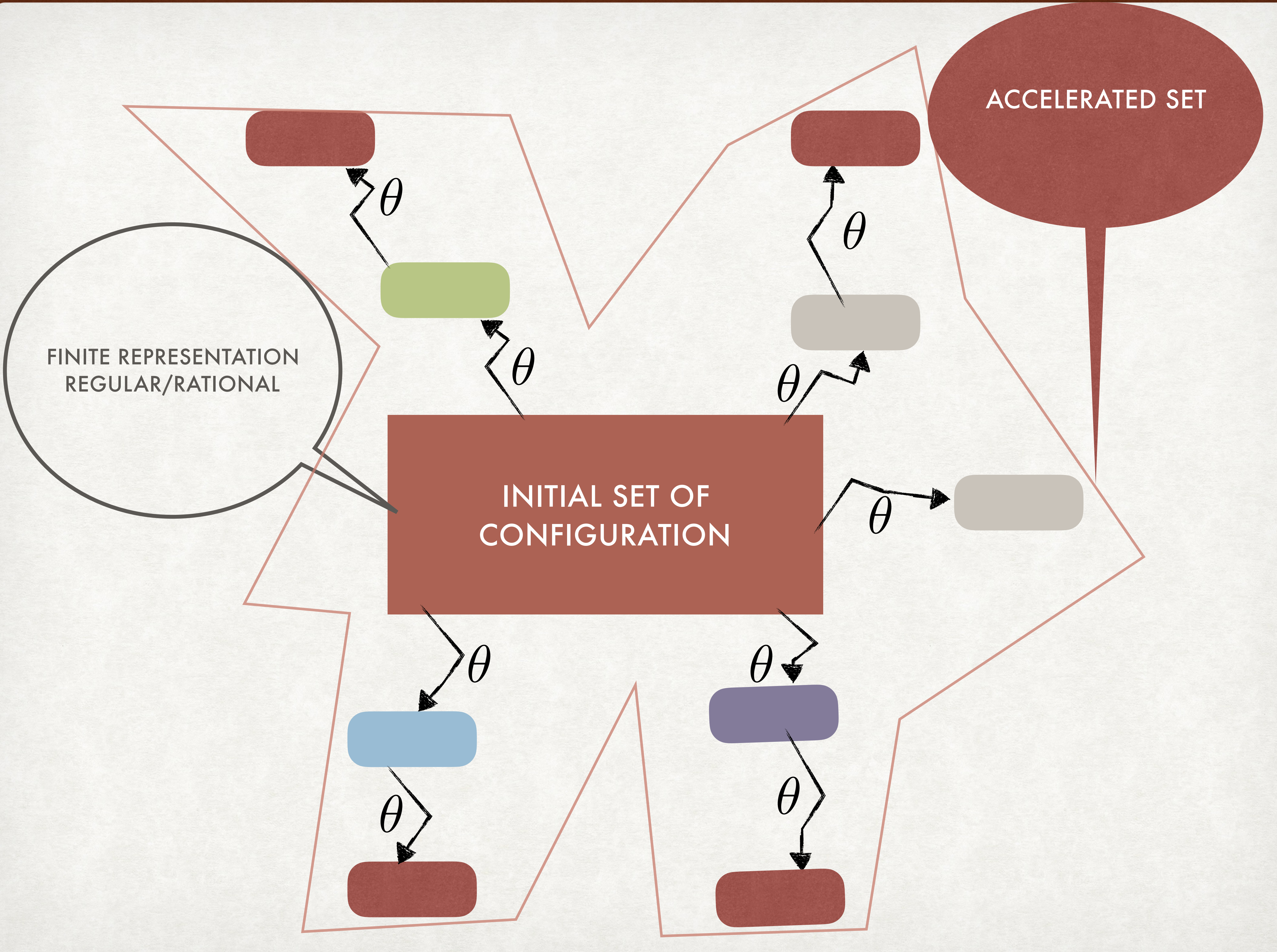
Reachability is NP-Complete

# Acceleration

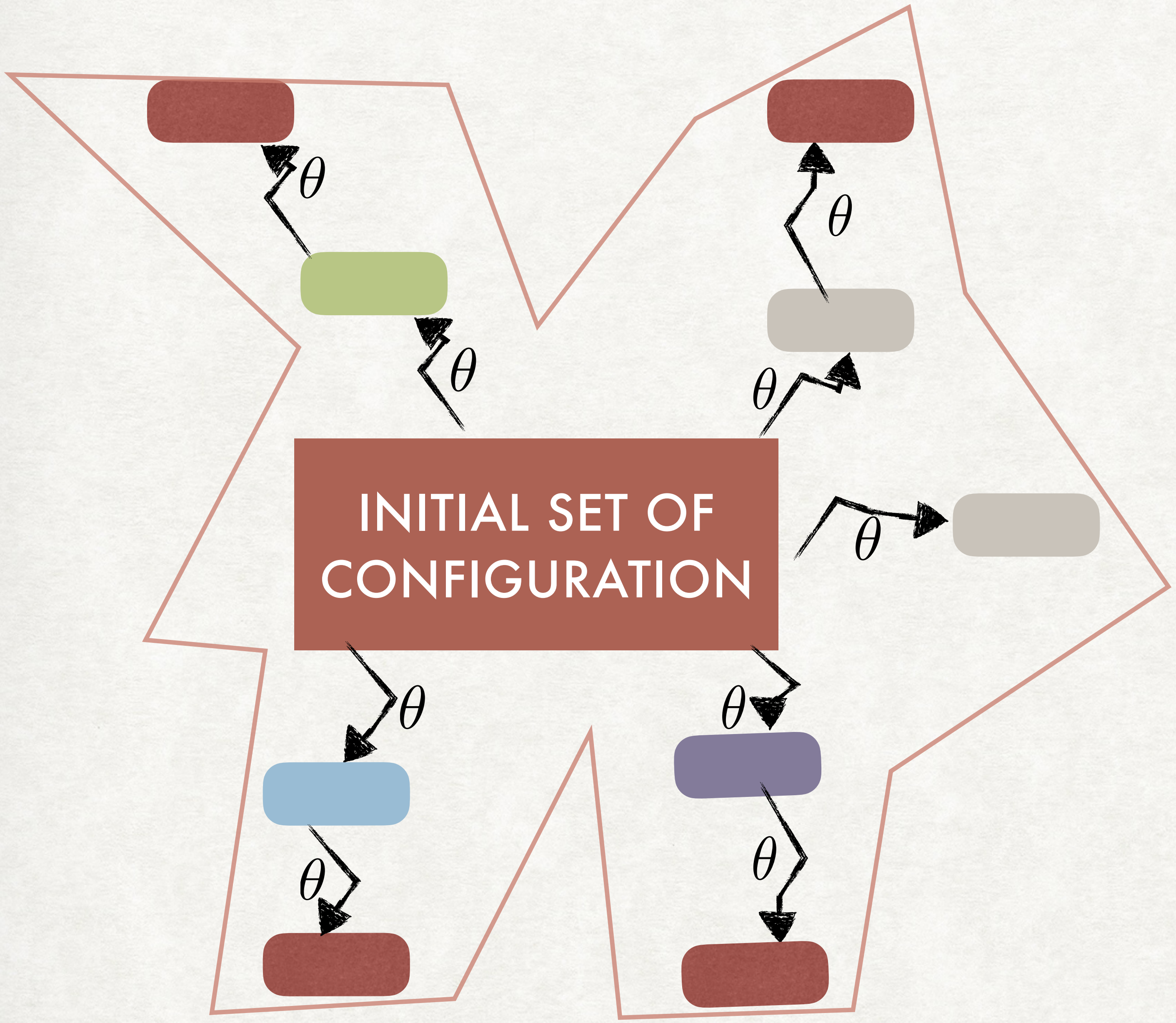
- Multi pushdown system $\mathcal{M}$
- Transitions $\Delta$
- Set of configurations $\mathcal{C}$
- Set of sequences of transitions $\theta$

<u>Acceleration problem is to compute</u>

$$\{c' \mid c \xrightarrow{\sigma} c', c \in \mathcal{C}, \sigma \in \theta^*\}$$


ACCELERATED SET
FINITE REPRESENTATION
REGULAR/RATIONAL
INITIAL SET OF
CONFIGURATION
$\theta$

# Stability




**Stability: Representation of initial configuration and the accelerated set are the same**

# Bounded context analysis as an acceleration problem

- Multi pushdown system $\mathcal{M}$
- Transitions $\Delta$
- Set of configurations $\mathcal{C}$
- Set of sequences of transitions

$$\theta = \bigcup_{i_1,\cdots,i_k \in [1..n]} \Delta_{i_1}^*.\Delta_{i_2}^* \ldots \Delta_{i_k}^*$$

- We are interested in the following set

$$\{c' \mid c \xrightarrow{\sigma} c, c \in \mathcal{C}, \sigma \in \theta\}$$

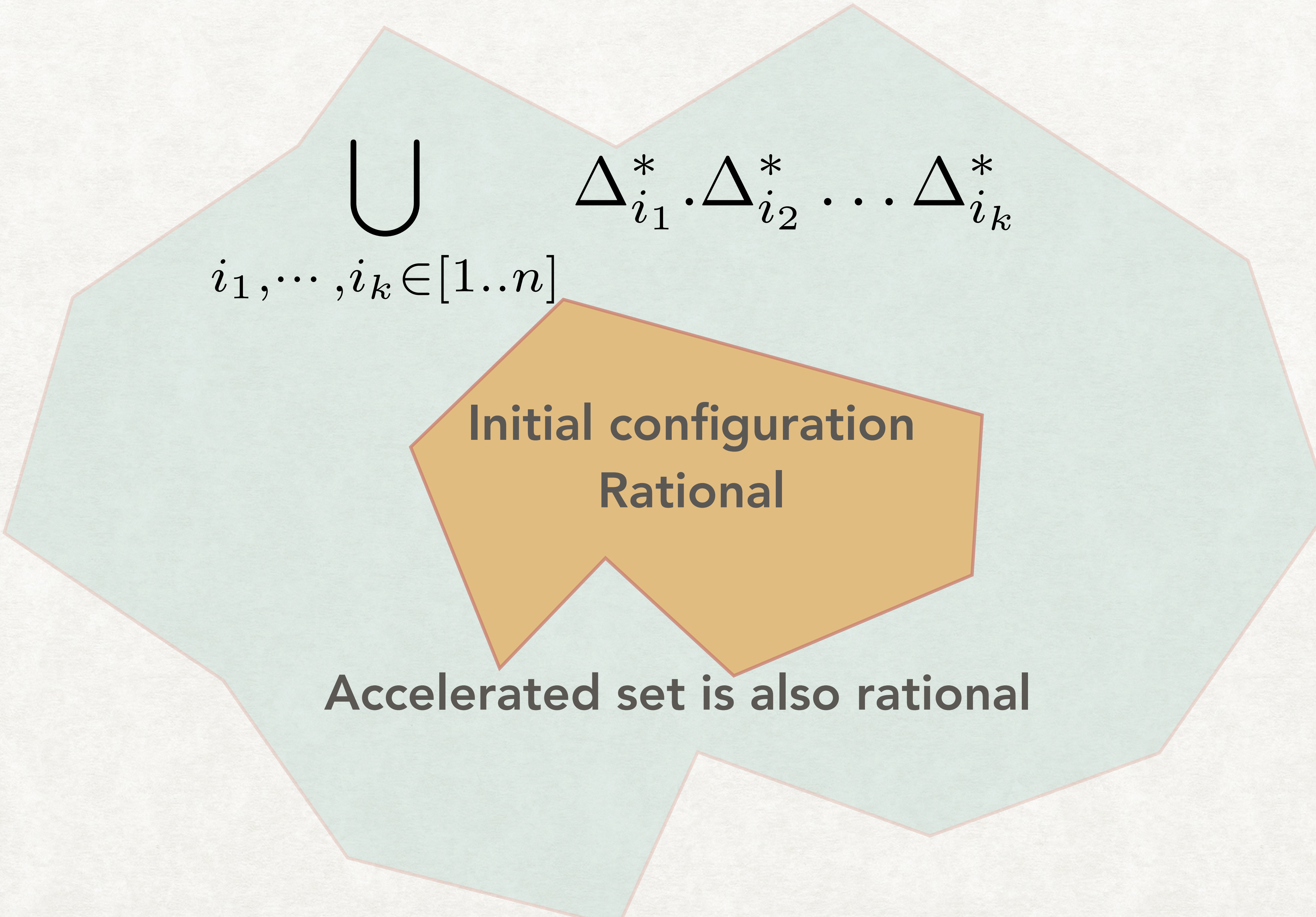
# Bounded context acceleration

$$\bigcup_{i_1,\cdots,i_k\in[1..n]} \Delta_{i_1}^*.\Delta_{i_2}^*\ldots\Delta_{i_k}^*$$

**Initial configuration**
**Regular**

**Accelerated set is also regular**

# Bounded context acceleration

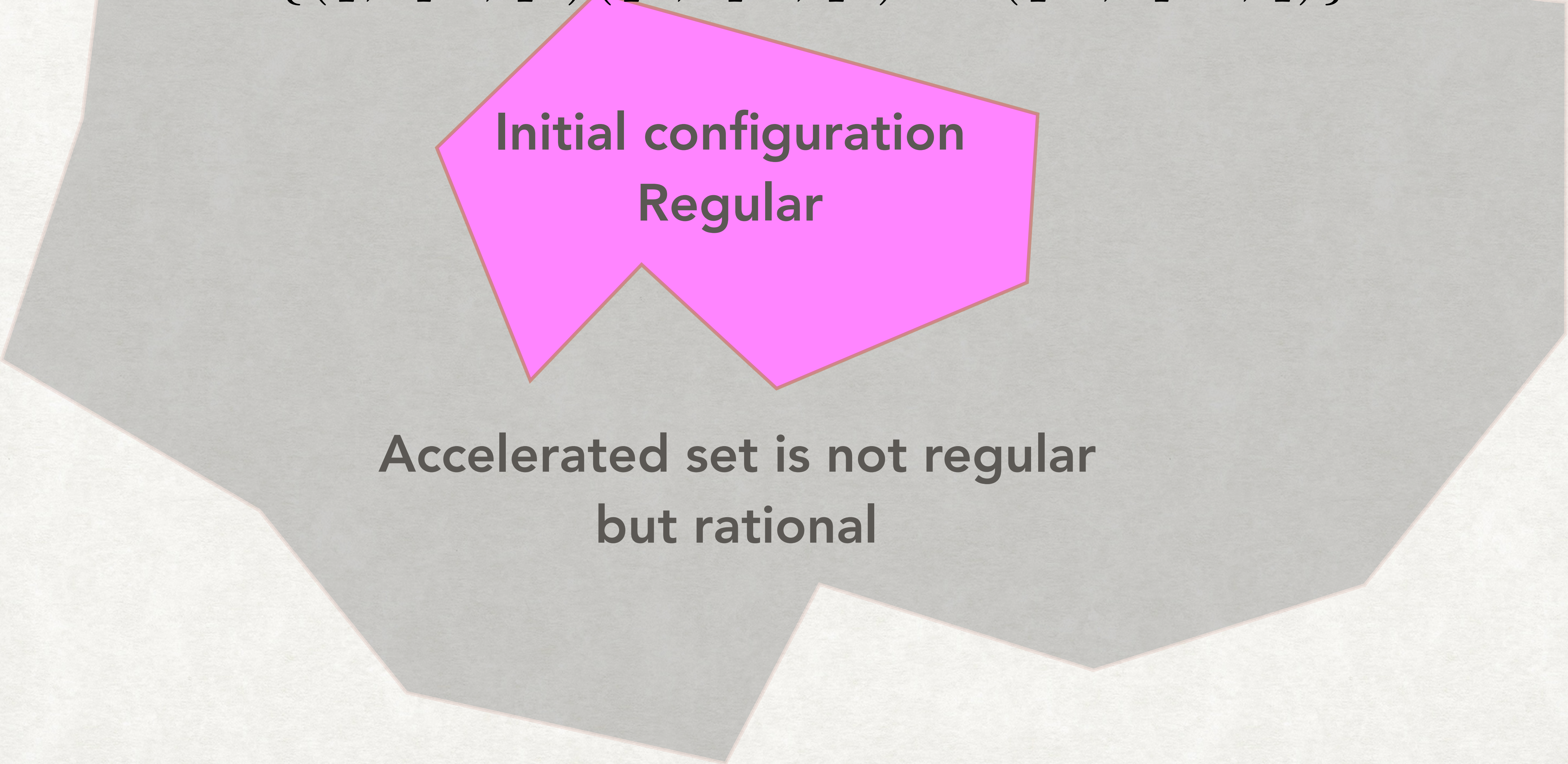


# Accelerating loop

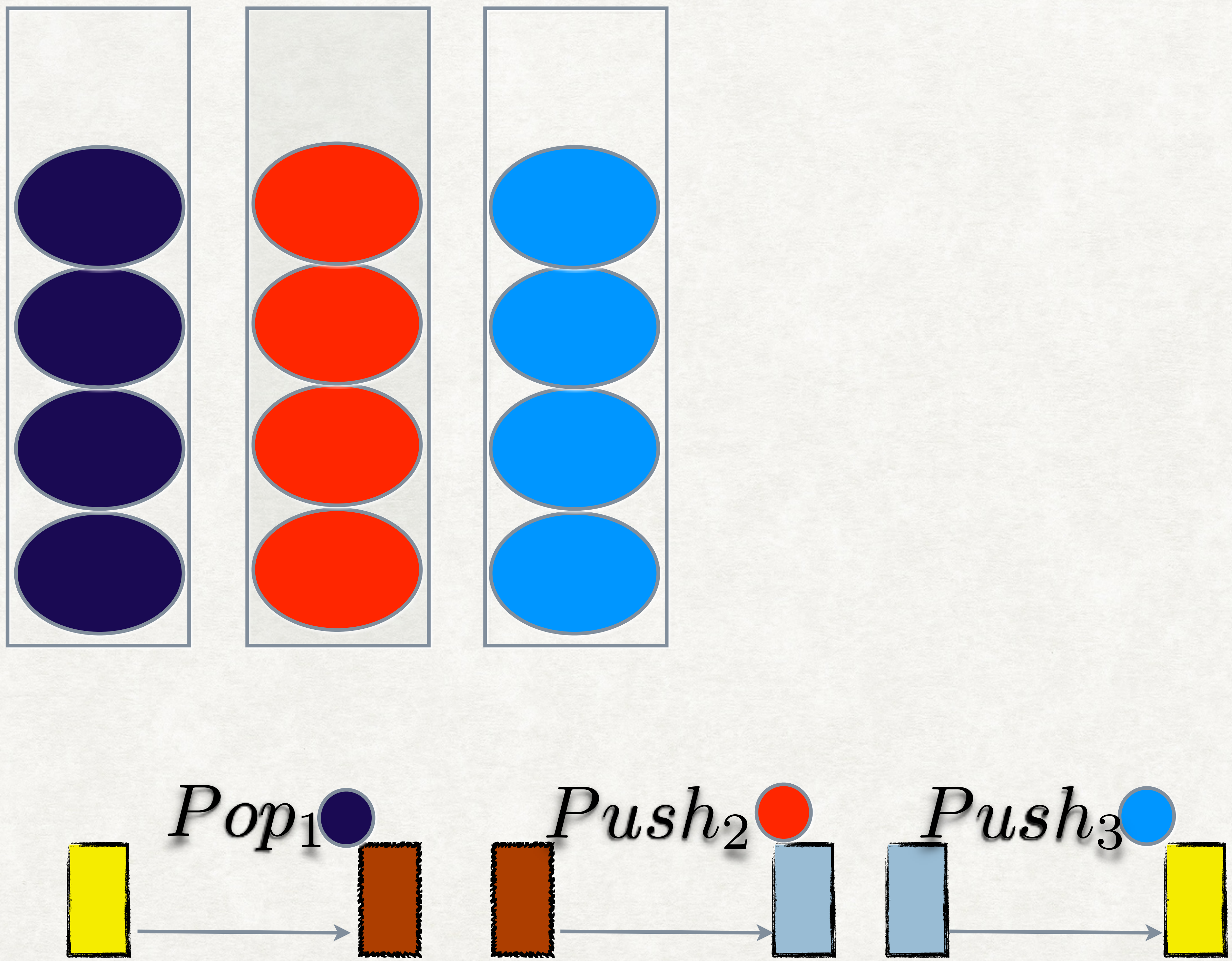
- Multi pushdown system $\mathcal{M}$
- Transitions $\Delta$
- Set of configurations $\mathcal{C}$
- loop $\theta = \{(q, op_1, q_1)(q_1, op_2, q_2) \cdots (q_m, op_m, q)\}$

$$\{c' \mid c \xrightarrow{\sigma} c', c \in \mathcal{C}, \sigma \in \theta^*\}$$

$$\theta = \{(q, op_1, q_1)(q_1, op_2, q_2) \cdots (q_m, op_m, q)\}$$



# Accelerating loop on regular set is not regular



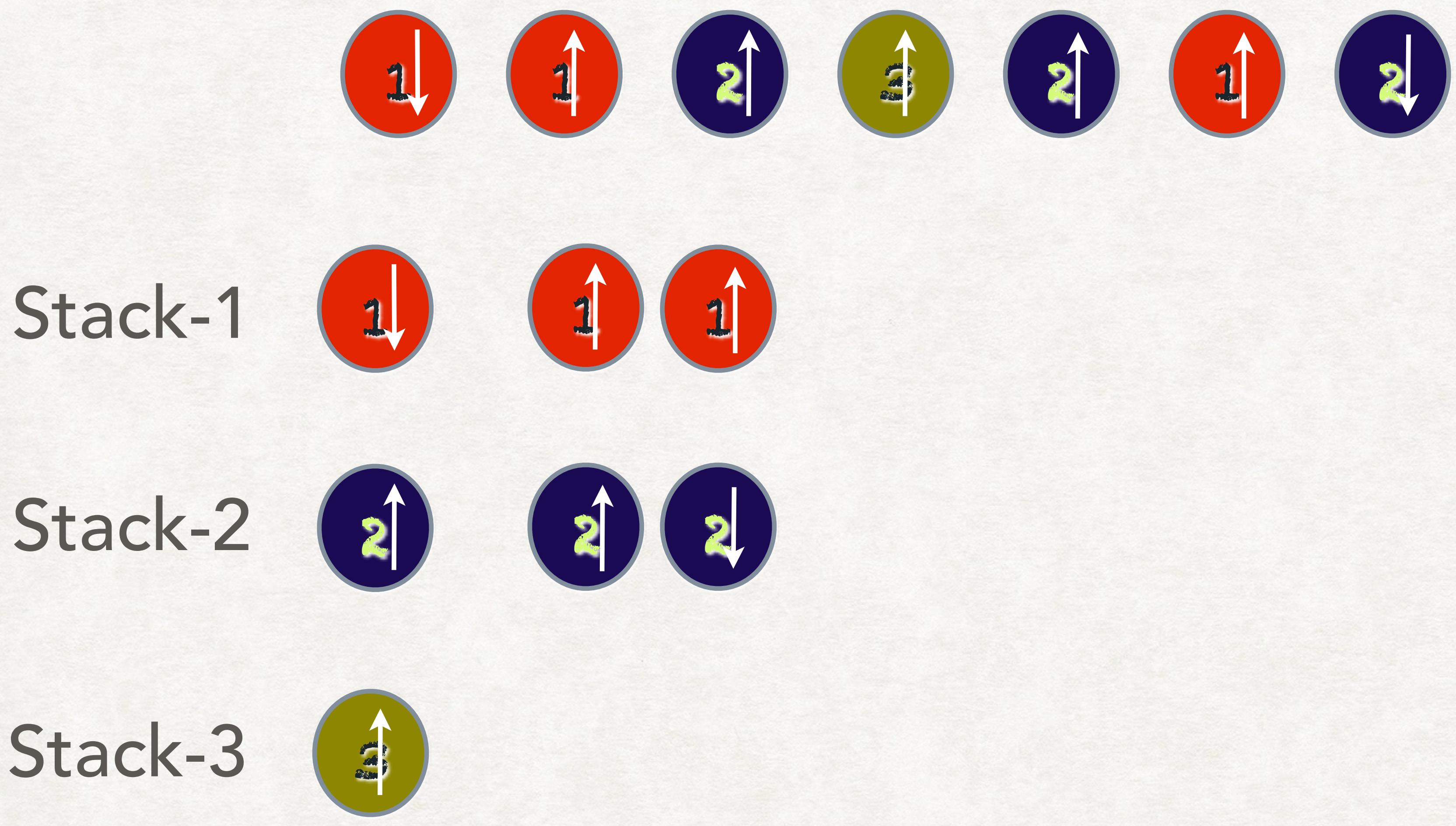
# Accelerating loop on regular set is rational

$\boxed{B_1}$ $\boxed{B_2}$ $\ldots\ldots\ldots$ $\boxed{B_n}$

We will assume that we are given a set of finite state automata one for each stack recognising the regular set of configurations

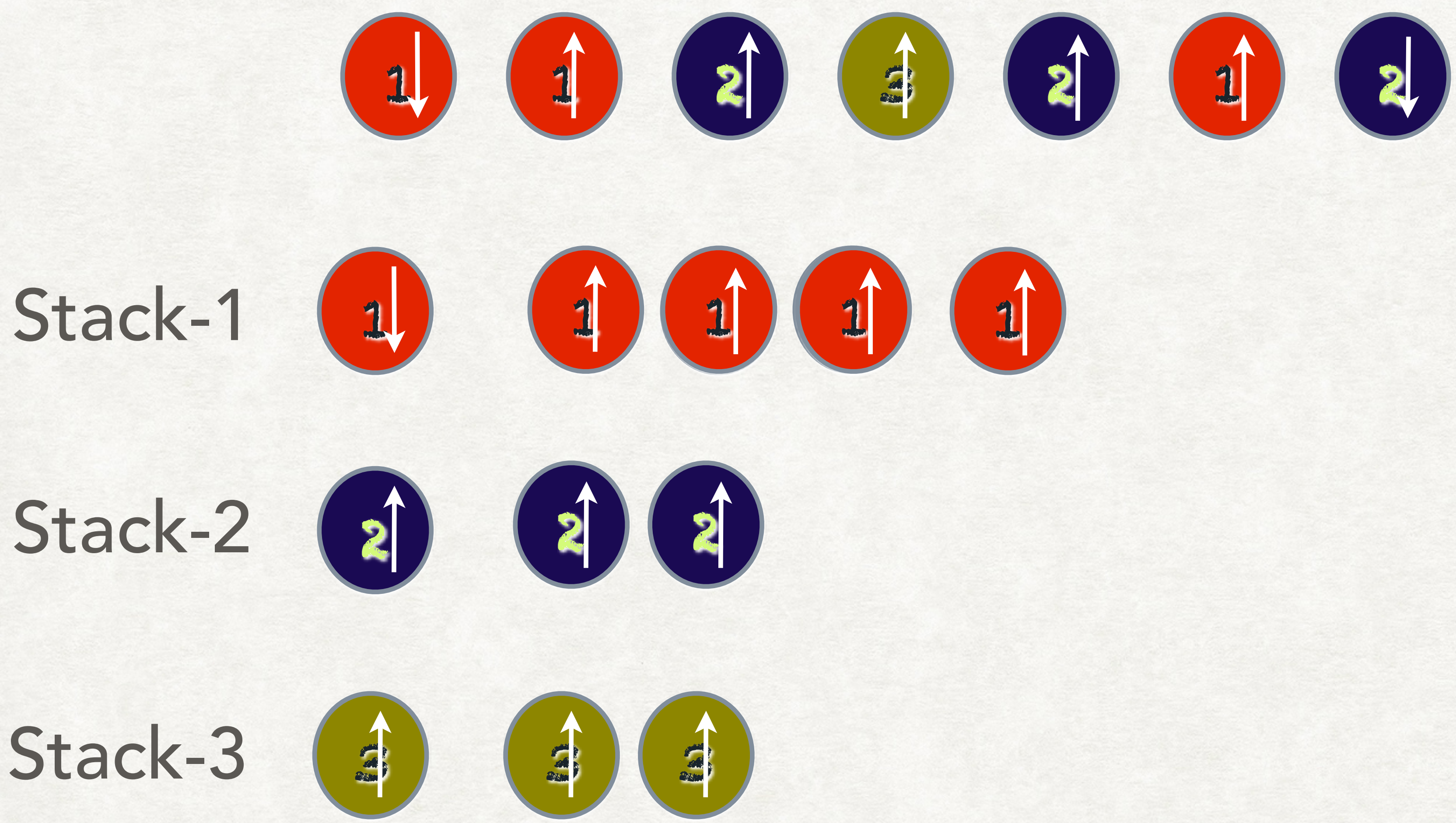
# Accelerating loop on regular set is rational

- We will first examine the effect of a loop on each stack



# Accelerating loop on regular set is rational

- What is the effect of accelerating the loop repeatedly?




| Stack | pop word | push word |
|---|---|---|
| 1 | 1↓ | 1↑ 1↑ |
| 2 | $\epsilon$ | 2↑ |
| 3 | $\epsilon$ | 3↑ |

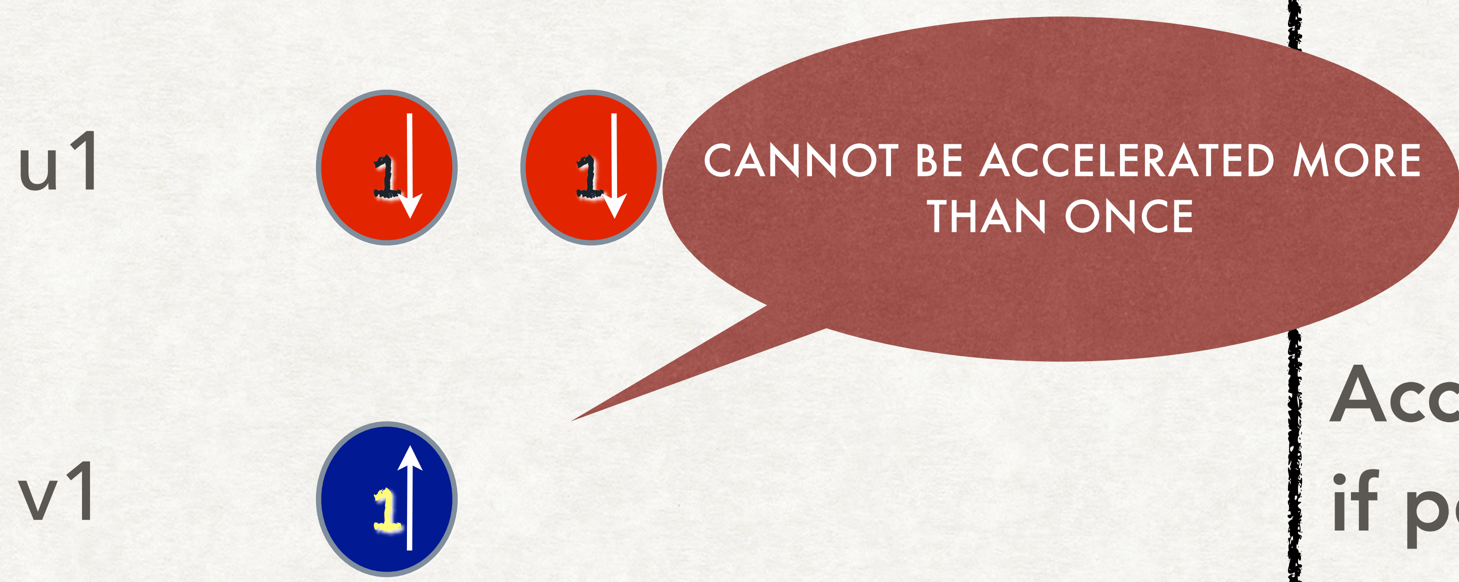
Given a loop, its effect can be summarised as two words for each stack

The first word is what is removed from the stack at the end of execution of the loop

The second word is what is appended to the stack at the end of loop execution

# Accelerating loop on regular set is rational

| Stack | pop | push |
|---|---|---|
| 1 | u1 | v1 |
| 2 | u2 | v2 |
| 3 | u3 | v3 |




Accelerating loop once amounts to removing pop word and adding push word to the stack, what about accelerating multiple times?

Acceleration is possible only if pop word is prefix of push word or push word is prefix of pop word.

# Accelerating loop on regular set is rational

| Stack | pop | push |
|---|---|---|
| 1 | $u_1$ | $v_1$ |
| 2 | $u_2$ | $v_2$ |
| 3 | $u_3$ | $v_3$ |

$$u_i <_{pre} v_i$$

$$v_i = u_i y_i, x_i = \epsilon$$

$$v_i <_{pre} u_i$$

$$u_i = v_i x_i, y_i = \epsilon$$

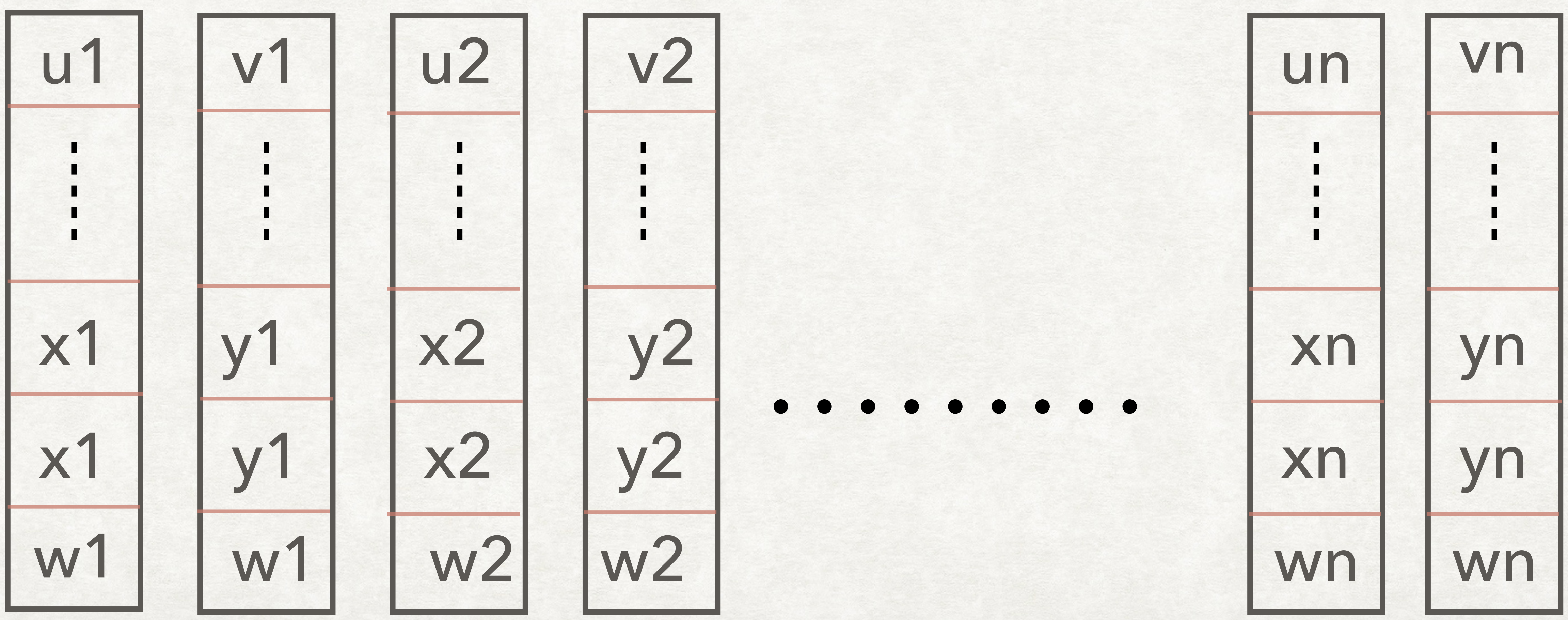
# Accelerating loop j+1 times

| Stack | pop | push |
|---|---|---|
| 1 | $u_1 x_1^j$ | $v_1 y_1^j$ |
| 2 | $u_2 x_2^j$ | $v_2 y_2^j$ |
| 3 | $u_3 x_3^j$ | $v_3 y_3^j$ |

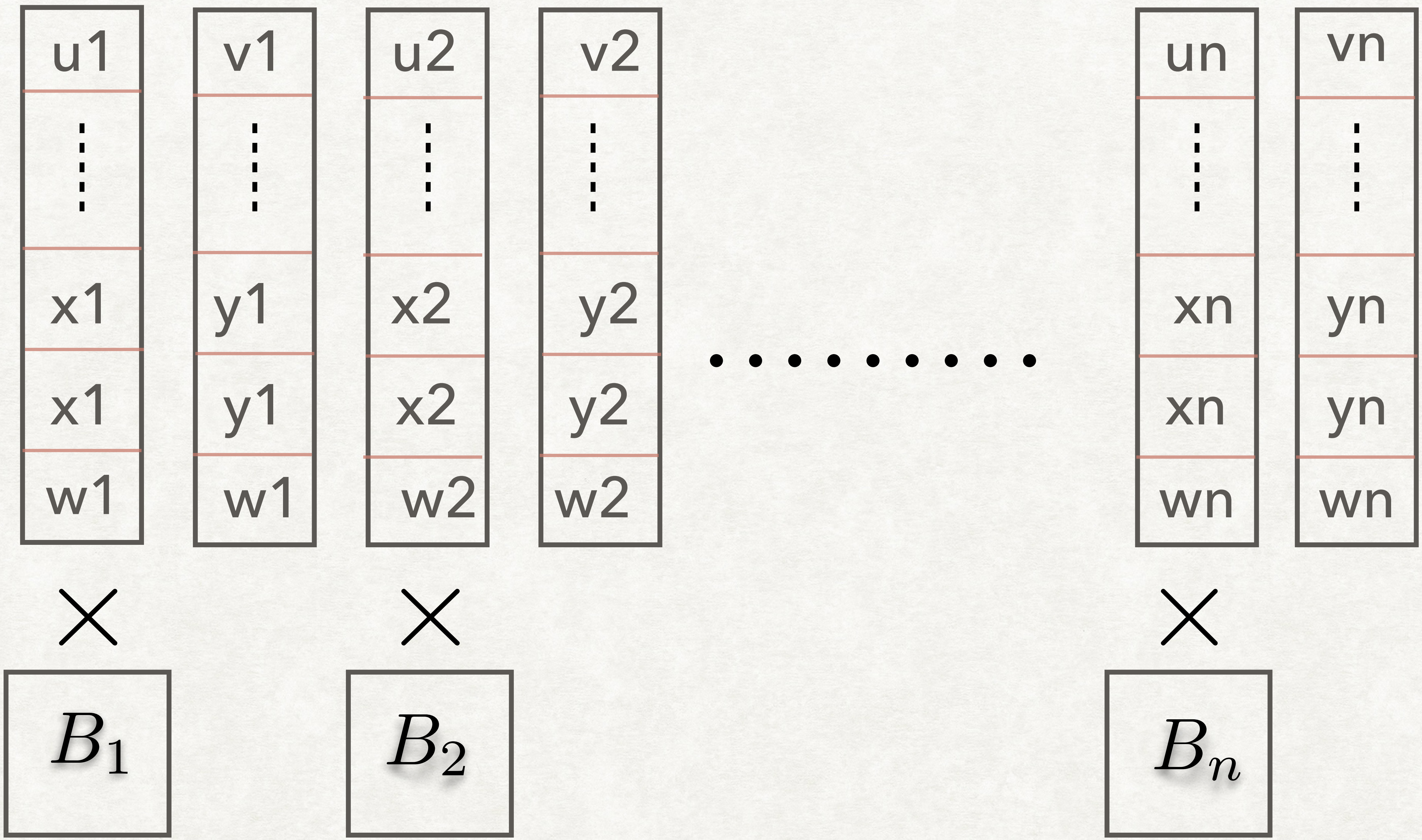
# Accelerating loop on regular set is rational

| Stack | pop | push |
|---|---|---|
| 1 | u1 | v1 |
| 2 | u2 | v2 |
| 3 | u3 | v3 |

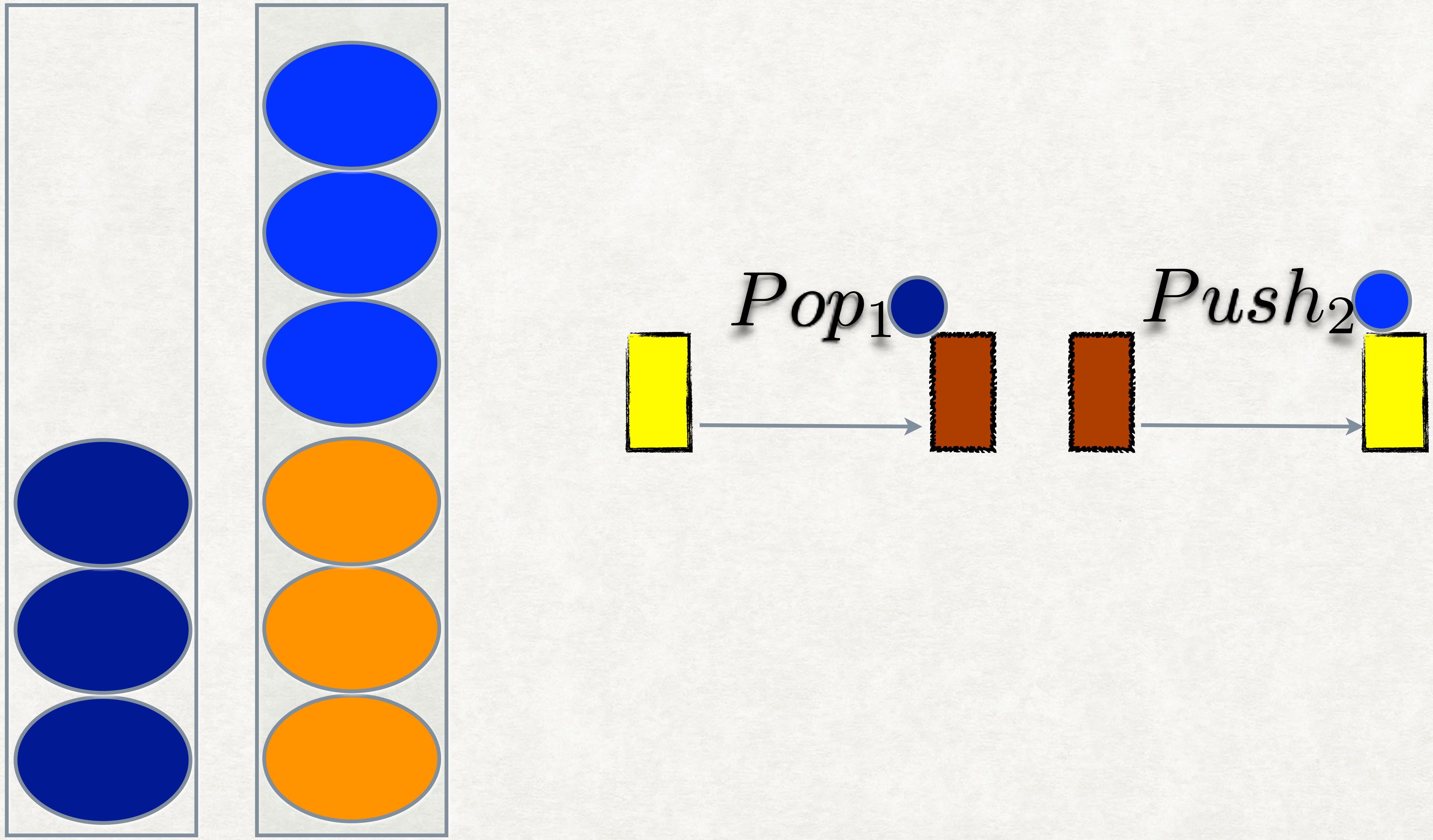
We construct an 2n tape rational automata.



# Accelerating loop on regular set is rational

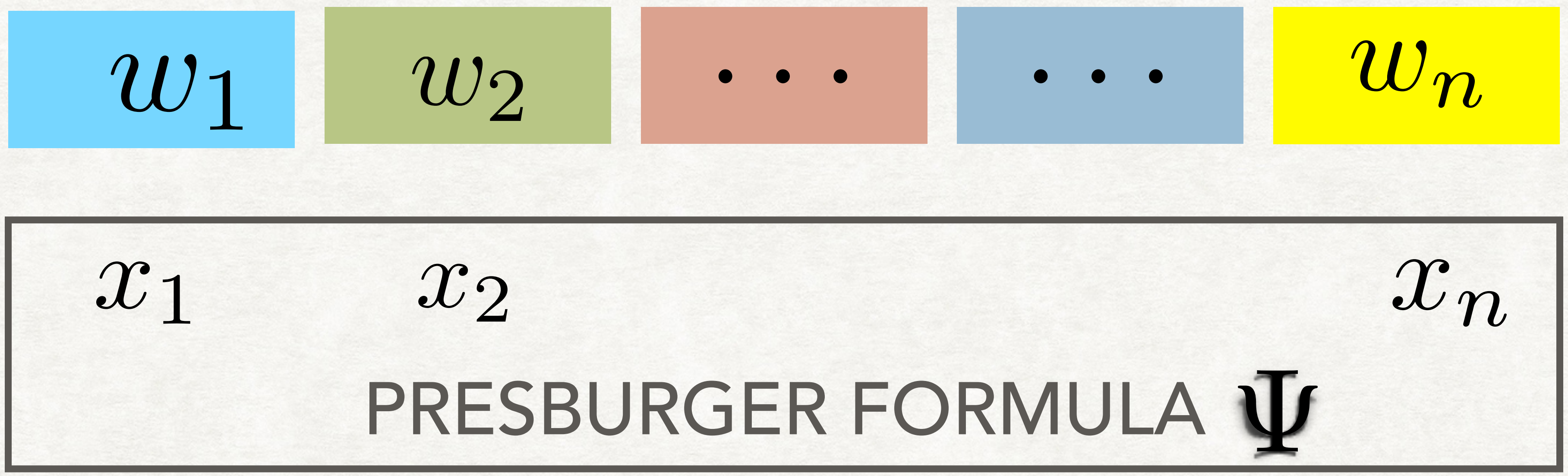


# Accelerating loop on rational set is not rational



# Constrained simple regular expression

# Constrained Simple Regular Expression (1-dim)




CSRE is given by sequence of words and a Presburger formula with one free variable per every word.

# Acceptance in CSRE

$$\underbrace{w_1 \; w_1 \cdots}_{i_1} \; \underbrace{w_2 \; w_2 \cdots}_{i_2} \quad \underbrace{w_n \; w_n \cdots}_{i_n}$$

$$i_1, i_2, \cdots, i_n \models \Psi(x_1, \cdots, x_n)$$

# Constrained Simple Regular Expression (m-dim)

| | | | | |
|---|---|---|---|---|
| $w_1^1$ | $w_2^1$ | $\cdots$ | $\cdots$ | $w_n^1$ |
| $w_1^2$ | $w_2^2$ | $\cdots$ | $\cdots$ | |
| $\vdots$ | $\vdots$ | $\vdots$ | | $\vdots$ |
| $w_1^m$ | $w_2^m$ | $\cdots$ | $\cdots$ | $w_n^m$ |

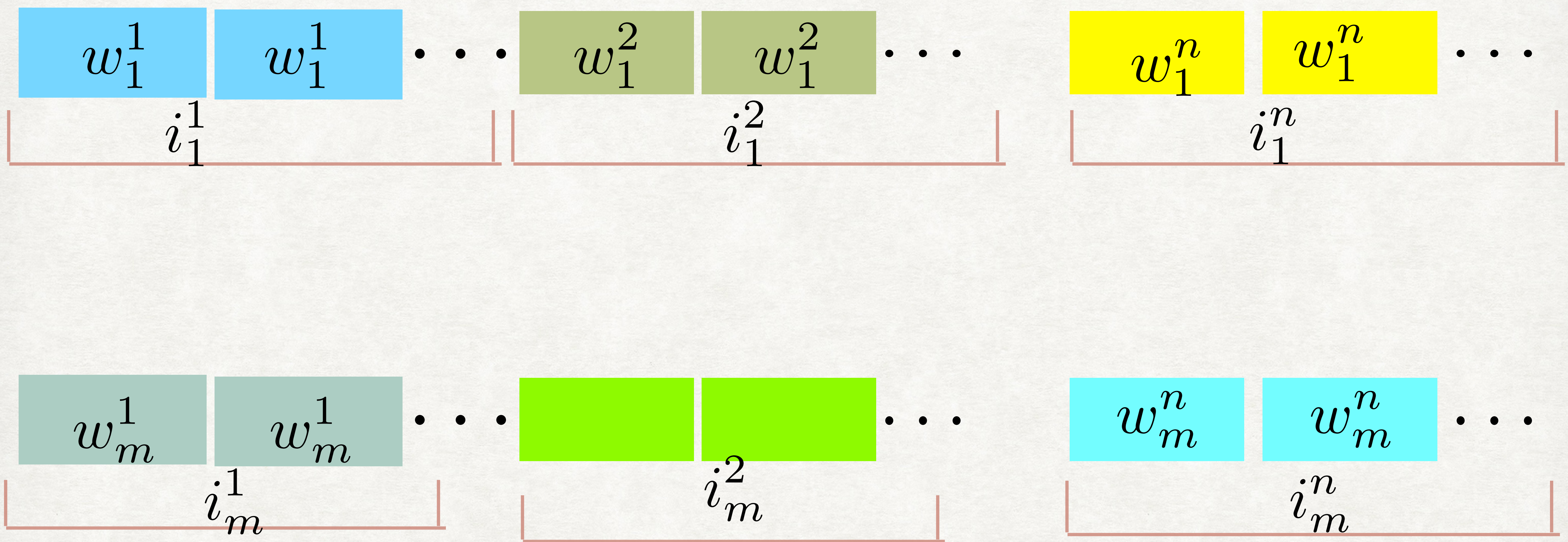
$x_1^1$ $\quad$ $x_1^2$ $\quad$ $\cdots$ $\quad$ $\cdots$ $\quad$ $x_n^1$

$x_2^1$ $\quad$ $x_2^2$

$\vdots$

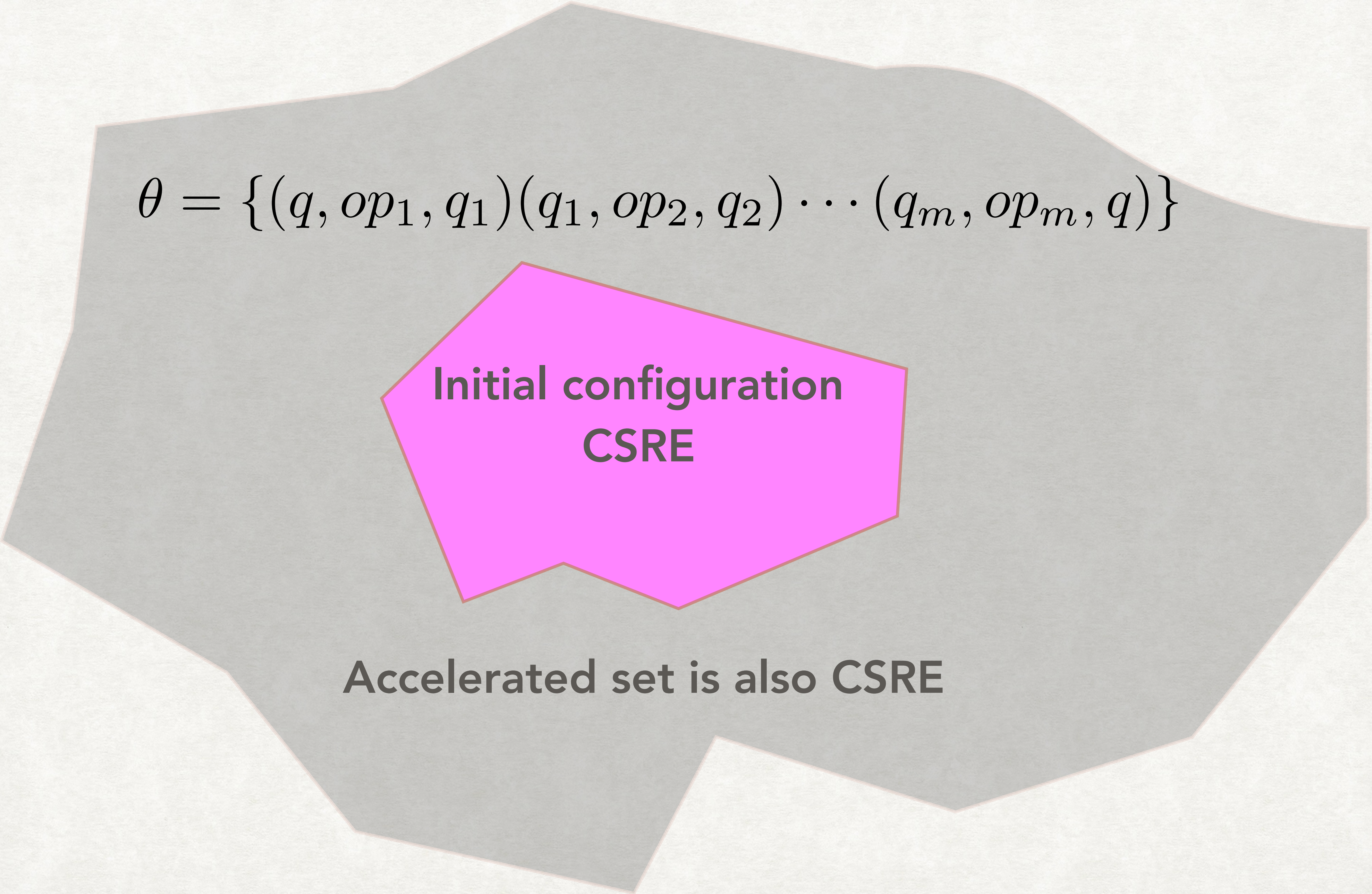
PRESBURGER FORMULA $\Psi$

# Acceptance in m-dim CSRE




$$i_1^1, i_1^2, \cdots, i_1^n, \cdots, i_m^1, \cdots, i_m^n \models \Psi$$

# STABLE



$\theta = \{(q, op_1, q_1)(q_1, op_2, q_2)\cdots(q_m, op_m, q)\}$
Initial configuration
CSRE
Accelerated set is also CSRE

## Some properties of CSRE

- CSRE are closed under intersection, union and concatenation
- Emptiness, membership and inclusion problems are decidable.
- CSRE is closed under left quotient.

# Accelerating loop on CSRE

# Accelerating loop j+1 times

| Stack | pop word | push word |
|---|---|---|
| 1 | $u_1 x_1^j$ | $v_1 y_1^j$ |
| 2 | $u_2 x_2^j$ | $v_2 y_2^j$ |
| 3 | $u_3 x_3^j$ | $v_3 y_3^j$ |

- We first left quotient $u_1, \cdots, u_n$
- We left quotient $x_1^j, \cdots, x_n^j$
- We concatenate $y_1^j, \cdots, y_n^j$
- We concatenate $v_1, \cdots, v_n$
- Presburger formula is used to ensure concatenation and left quotient is done same number of times

# Context switch set

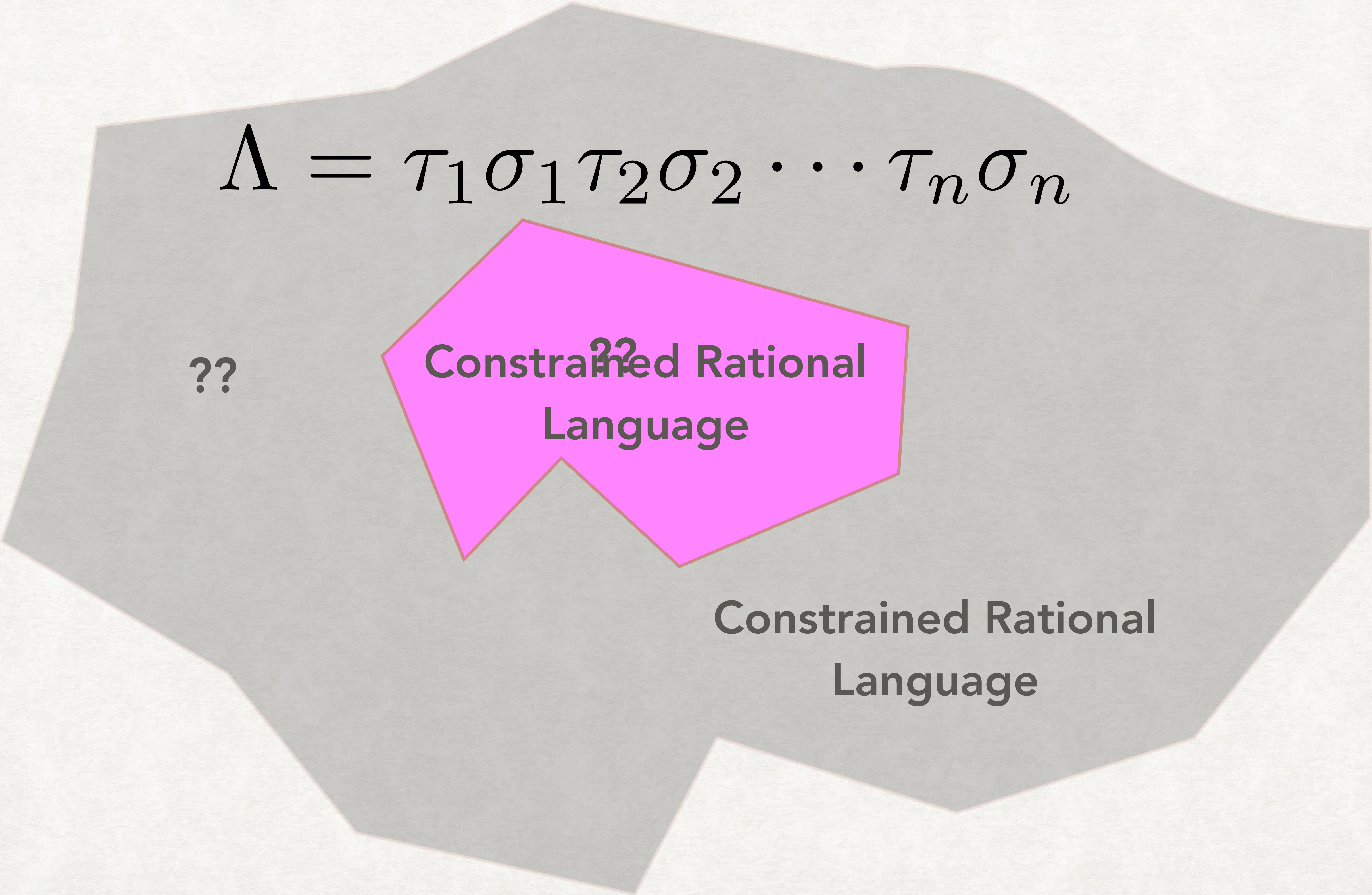
# Context switch set

- Loops are weak, cannot capture bounded context switch
- We will introduce notion of context switch set

$$\Lambda = \tau_1^* \sigma_1 \tau_2^* \sigma_2 \cdots \tau_n^* \sigma_n$$

- $\tau_i$ Subset of transitions operating on a stack
- $\sigma_i$ Single transition

$$\Lambda = \tau_1\sigma_1\tau_2\sigma_2\cdots\tau_n\sigma_n$$



# Constrained Rational Automata

# Constrained Rational Automata (Parikh automata)

**Multi tape automata**

$q_1 \quad q_4 \quad q_2$

$q_3 \qquad q_m$

$\Sigma_1 \quad \Sigma_2 \quad \Sigma_3$

$\Sigma_4 \qquad \Sigma_n$

$\tau_1 \quad \tau_2 \quad \tau_3 \quad \cdots \quad \tau_m$

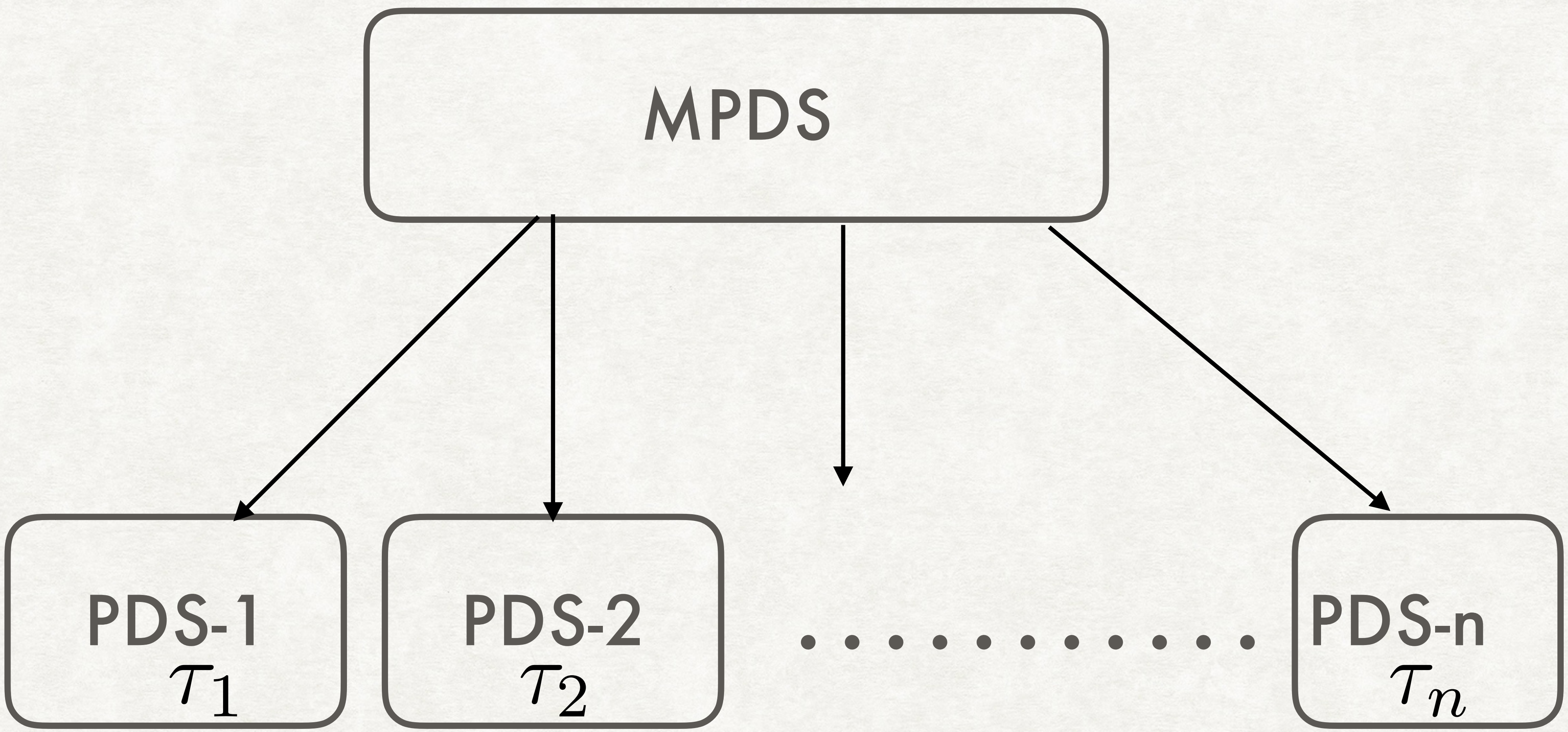
$t_1 \quad t_2 \quad t_3 \qquad t_m$

**Presburger formula** $\Phi$

## Some properties of  Constrained Rational Automata (CRA)

- CRA are closed under concatenation, union but not under intersection
- Emptiness and membership problems are decidable.

- Create a pushdown systems, one for each stack.



- Each of the pushdown system i simulates moves of stack-i
- It further has jump transitions corresponding to $\sigma_1, \cdots, \sigma_n$
- It outputs number of times a jump transition was made.
- The following language for any pushdown is rational

$$\{(v, w, \#^j)|(q, v) \xrightarrow{\#^j} (q, v')\}$$

- We can use Presburger formula to ensure that the number of jumps of each PDS match.

# THANK YOU