# Hyper Pattern Matching

**Masaki Waga**[1,2] and Étienne André[3]

Kyoto University[1], National Institute of Informatics[2], and Nantes Université[3]

RV 2025, 19th Sep. 2025

A kind of monitoring problem

# Hyper Pattern Matching

**Masaki Waga**[1,2] and Étienne André[3]

Kyoto University[1], National Institute of Informatics[2], and Nantes Université[3]

RV 2025, 19th Sep. 2025

Hyperproperties

A kind of monitoring problem

# Hyper Pattern Matching

**Masaki Waga**[1,2] and Étienne André[3]

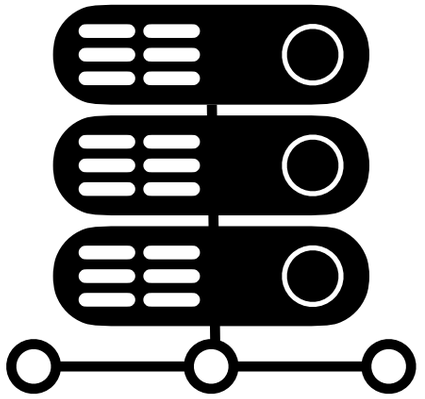
Kyoto University[1], National Institute of Informatics[2], and Nantes Université[3]

RV 2025, 19th Sep. 2025

# Monitoring

Detect violation of formal spec. in a trace

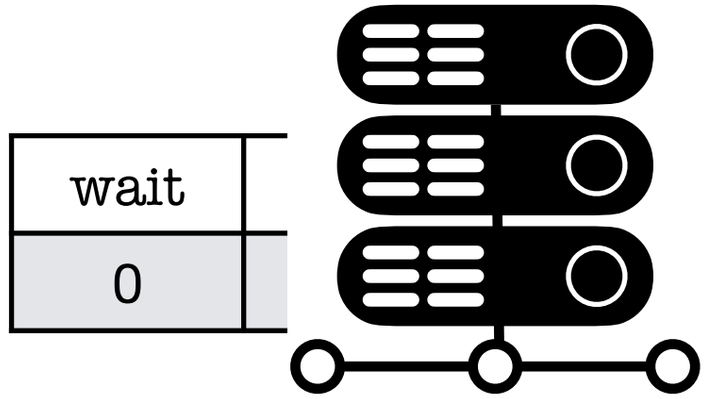
**Spec:** `inc`, `dec`, `wait`, …
implement a shared counter



# Monitoring

Detect violation of formal spec. in a trace

**Spec:** `inc`, `dec`, `wait`, …
implement a shared counter

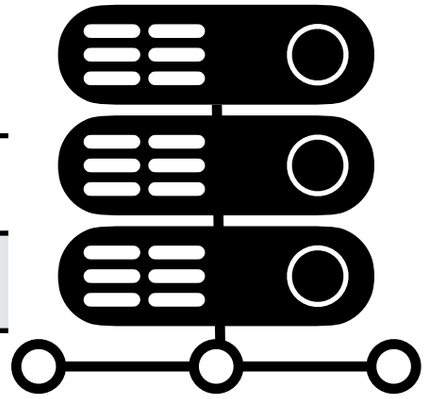


# Monitoring

Detect violation of formal spec. in a trace

**Spec:** `inc`, `dec`, `wait`, …
implement a shared counter

| wait | inc | |
|---|---|---|
| 0 | 1 | |

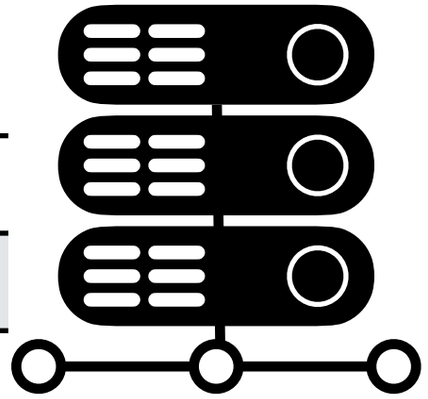


# Monitoring

Detect violation of formal spec. in a trace

**Spec:** `inc`, `dec`, `wait`, …
implement a shared counter

| wait | inc | inc | |
|---|---|---|---|
| 0 | 1 | 2 | |

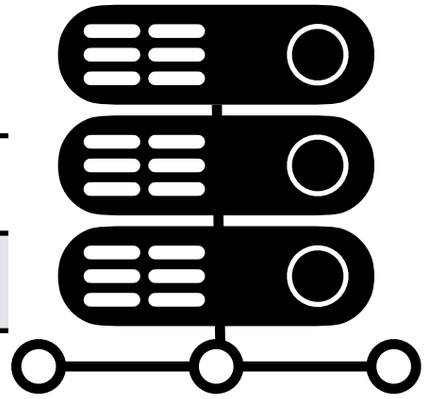


# Monitoring

Detect violation of formal spec. in a trace

**Spec:** `inc`, `dec`, `wait`, …
implement a shared counter

| wait | inc | inc | dec | |
|---|---|---|---|---|
| 0 | 1 | 2 | 1 | |

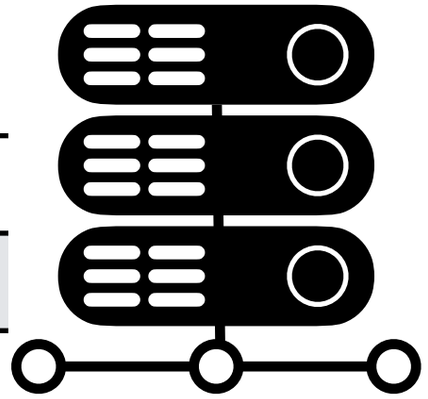


# Monitoring

Detect violation of formal spec. in a trace

**Spec:** `inc`, `dec`, `wait`, …
implement a shared counter

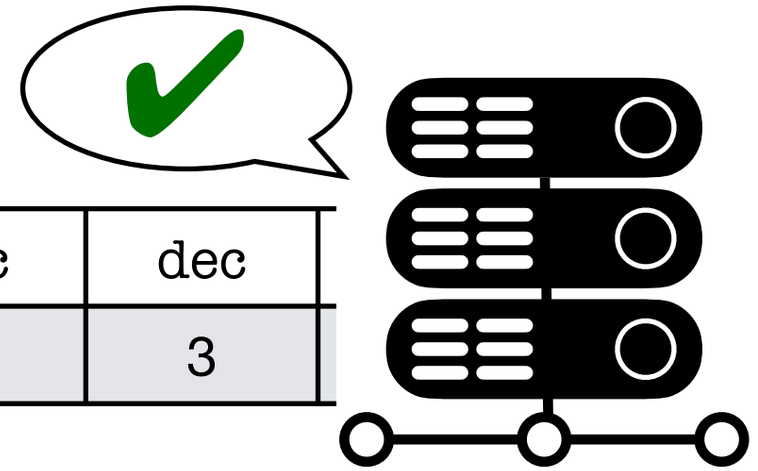
| wait | inc | inc | dec | wait | |
|---|---|---|---|---|---|
| 0 | 1 | 2 | 1 | 2 | |



# Monitoring

Detect violation of formal spec. in a trace

**Spec:** `inc`, `dec`, `wait`, …
implement a shared counter

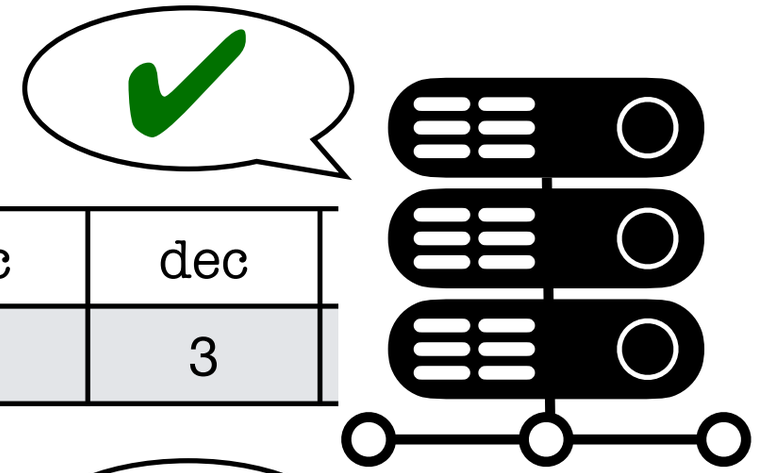


| wait | inc | inc | dec | wait | wait | inc | dec |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 1 | 2 | 3 | 4 | 3 |

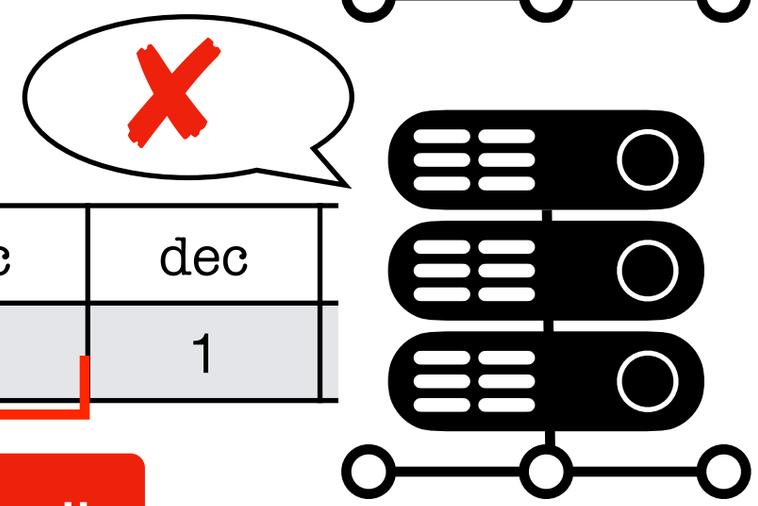
# Monitoring

Detect violation of formal spec. in a trace

**Spec:** `inc`, `dec`, `wait`, …
implement a shared counter



| wait | inc | inc | dec | wait | wait | inc | dec |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 1 | 2 | 3 | 4 | 3 |



| wait | inc | inc | dec | wait | wait | inc | dec |
|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 1 | 2 | 3 | 2 | 1 |

**Erroneous!!**

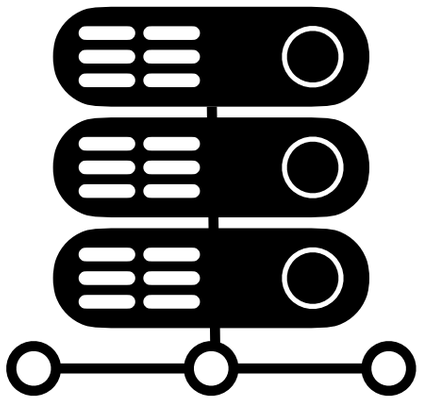
# Pattern Matching for Monitoring

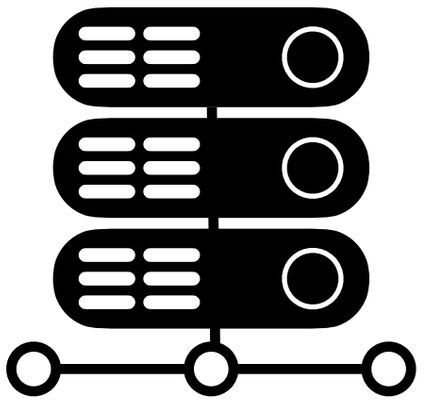
Return the intervals witnessing the violations

**Spec:** `inc`, `dec`, `wait`, … implement a shared counter

## Conventional Monitoring (w/ Boolean verdict)

## Pattern Matching





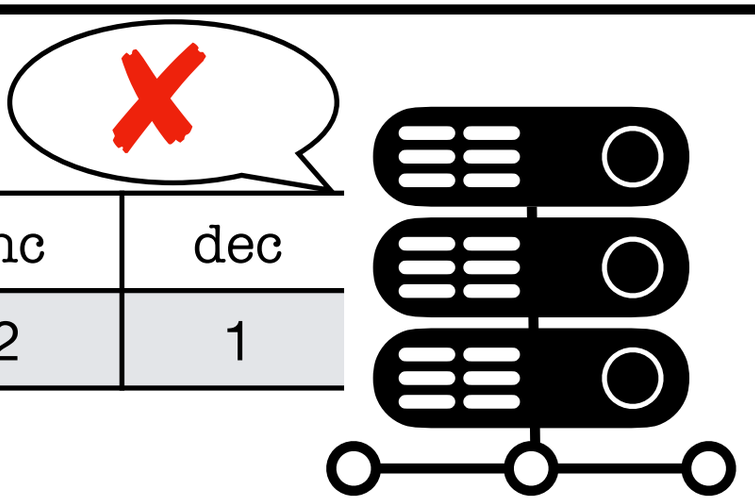
# Pattern Matching for Monitoring

Return the intervals witnessing the violations

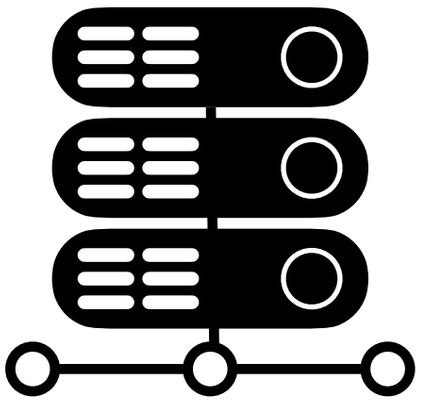
**Spec:** `inc`, `dec`, `wait`, … implement a shared counter

## Conventional Monitoring (w/ Boolean verdict)

| dec | wait | inc | inc | dec | wait | wait | inc | dec |
|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 2 | 1 | 2 | 3 | 2 | 1 |



## Pattern Matching

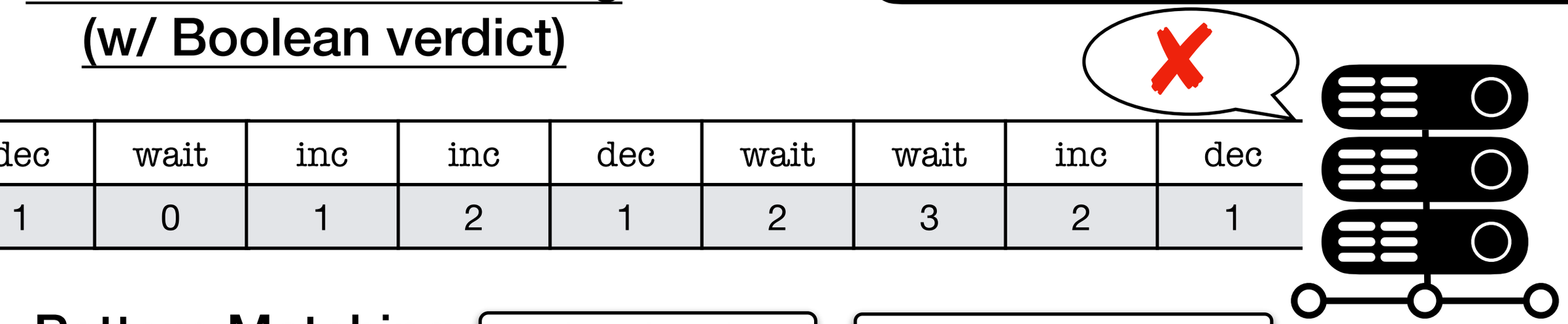


# Pattern Matching for Monitoring

Return the intervals witnessing the violations

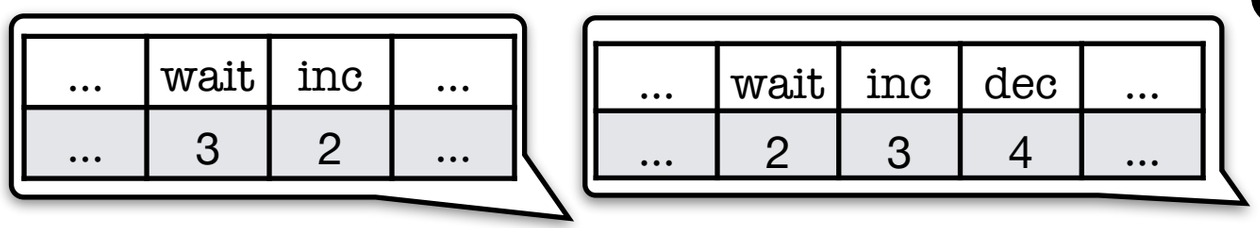
**Spec:** `inc`, `dec`, `wait`, … implement a shared counter

## Conventional Monitoring (w/ Boolean verdict)



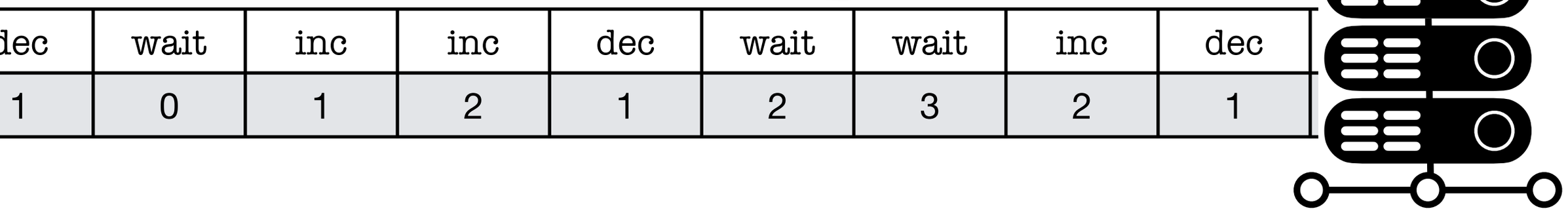
| dec | wait | inc | inc | dec | wait | wait | inc | dec |
|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 2 | 1 | 2 | 3 | 2 | 1 |

## Pattern Matching



| … | wait | inc | … |
|---|---|---|---|
| … | 3 | 2 | … |

| … | wait | inc | dec | … |
|---|---|---|---|---|
| … | 2 | 3 | 4 | … |



| dec | wait | inc | inc | dec | wait | wait | inc | dec |
|---|---|---|---|---|---|---|---|---|
| 1 | 0 | 1 | 2 | 1 | 2 | 3 | 2 | 1 |

# Hyperproperties

[Clarkson & Schneider, J. Comput. Secur. 2010]

Properties addressing multiple independent traces

**Trace Property**

inc, dec, wait, … implement a shared counter

| ... | wait | inc | inc | wait | dec | wait | inc | dec | wait | ... |
|---|---|---|---|---|---|---|---|---|---|---|
| ... | 0 | 1 | 2 | 3 | 2 | 2 | 3 | 4 | 3 | ... |

**Hyperproperty**

Updates in wait are deterministic < 10 steps after sync

| ... | inc | dec | wait | wait | sync | inc | dec | wait | inc | ... |
|---|---|---|---|---|---|---|---|---|---|---|
| ... | 2 | 1 | 2 | 2 | 2 | 3 | 2 | 3 | 4 | ... |

# Hyperproperties

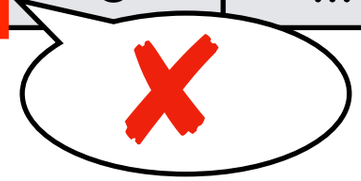
[Clarkson & Schneider, J. Comput. Secur. 2010]

Properties addressing multiple independent traces

**Trace Property**

inc, dec, wait, … implement a shared counter

| … | wait | inc | inc | wait | dec | wait | inc | dec | wait | … |
|---|---|---|---|---|---|---|---|---|---|---|
| … | 0 | 1 | 2 | 3 | 2 | 2 | 3 | 4 | 3 | … |



**Hyperproperty**

Updates in wait are deterministic < 10 steps after sync

| … | inc | dec | wait | wait | sync | inc | dec | wait | inc | … |
|---|---|---|---|---|---|---|---|---|---|---|
| … | 2 | 1 | 2 | 2 | 2 | 3 | 2 | 3 | 4 | … |

# Hyperproperties

[Clarkson & Schneider, J. Comput. Secur. 2010]

Properties addressing multiple independent traces

**Trace Property**

inc, dec, wait, … implement a shared counter

| … | wait | inc | inc | wait | dec | wait | inc | dec | wait | … |
|---|---|---|---|---|---|---|---|---|---|---|
| … | 0 | 1 | 2 | 3 | 2 | 2 | 3 | 4 | 3 | … |

**Hyperproperty**

Updates in wait are deterministic < 10 steps after sync

| … | inc | dec | wait | wait | sync | inc | dec | wait | inc | … |
|---|---|---|---|---|---|---|---|---|---|---|
| … | 2 | 1 | 2 | 2 | 2 | 3 | 2 | 3 | 4 | … |

# Hyperproperties

[Clarkson & Schneider, J. Comput. Secur. 2010]

Properties addressing multiple independent traces

**Trace Property**

inc, dec, wait, … implement a shared counter

| … | wait | inc | inc | wait | dec | wait | inc | dec | wait | … |
|---|---|---|---|---|---|---|---|---|---|---|
| … | 0 | 1 | 2 | 3 | 2 | 2 | 3 | 4 | 3 | … |

**Hyperproperty**

Updates in wait are deterministic < 10 steps after sync

| … | inc | dec | wait | wait | sync | inc | dec | wait | inc | … |
|---|---|---|---|---|---|---|---|---|---|---|
| … | 2 | 1 | 2 | 2 | 2 | 3 | 2 | 3 | 4 | … |

| … | wait | inc | inc | dec | sync | inc | dec | wait | inc | … |
|---|---|---|---|---|---|---|---|---|---|---|
| … | 1 | 2 | 3 | 2 | 2 | 3 | 2 | 1 | 2 | … |

# Hyperproperties

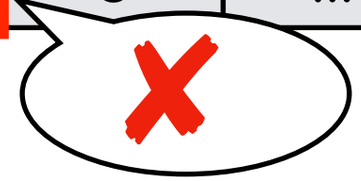
[Clarkson & Schneider, J. Comput. Secur. 2010]

Properties addressing multiple independent traces

**Trace Property**

inc, dec, wait, … implement a shared counter

| … | wait | inc | inc | wait | dec | wait | inc | dec | wait | … |
|---|---|---|---|---|---|---|---|---|---|---|
| … | 0 | 1 | 2 | 3 | 2 | 2 | 3 | 4 | 3 | … |

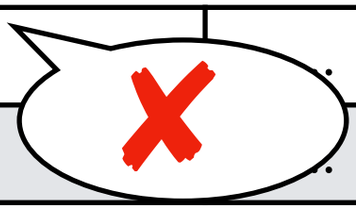


**Hyperproperty**

Updates in wait are deterministic < 10 steps after sync

| … | inc | dec | wait | wait | sync | inc | dec | wait | inc | … |
|---|---|---|---|---|---|---|---|---|---|---|
| … | 2 | 1 | 2 | 2 | 2 | 3 | 2 | 3 | 4 | … |

| … | wait | inc | inc | dec | sync | inc | dec | wait | … |
|---|---|---|---|---|---|---|---|---|---|
| … | 1 | 2 | 3 | 2 | 2 | 3 | 2 | 1 | … |



# Hyper Pattern Matching

[Contribution]

## Inputs

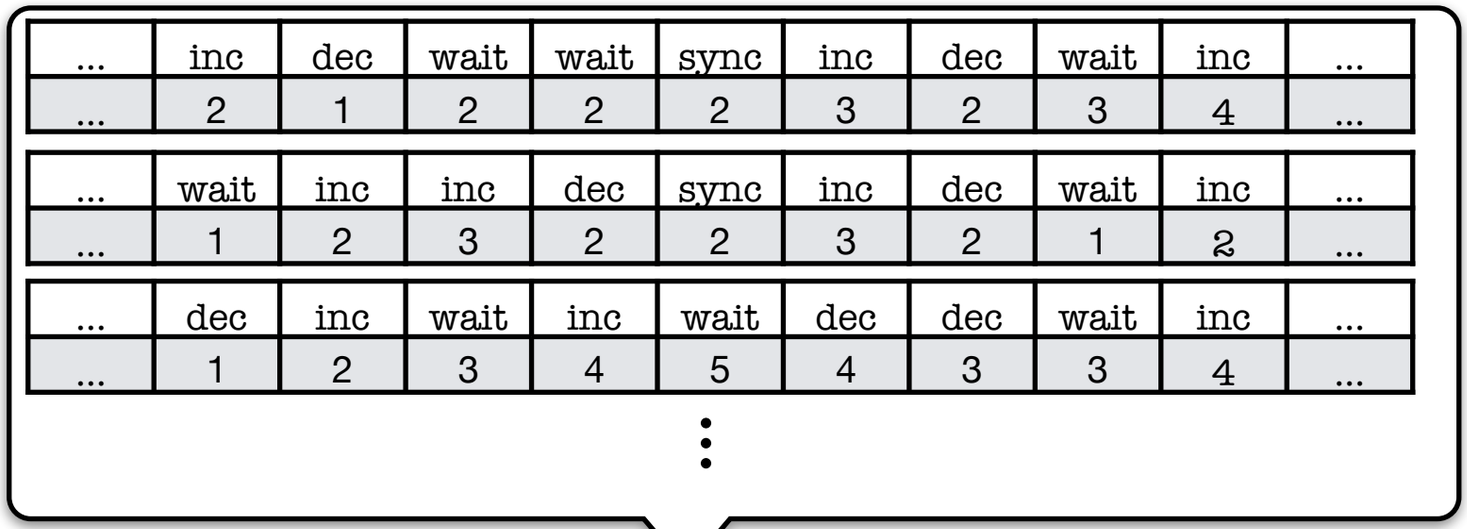
- Finite set $\mathscr{W} = \{w_1, w_2, \ldots, w_n\}$ of words
- A pattern $\mathscr{A}$ representing a hyperproperty

## Output

- Set of tuples of subwords in $\mathscr{W}$ matching $\mathscr{A}$

# Hyper Pattern Matching

[Contribution]




## Inputs

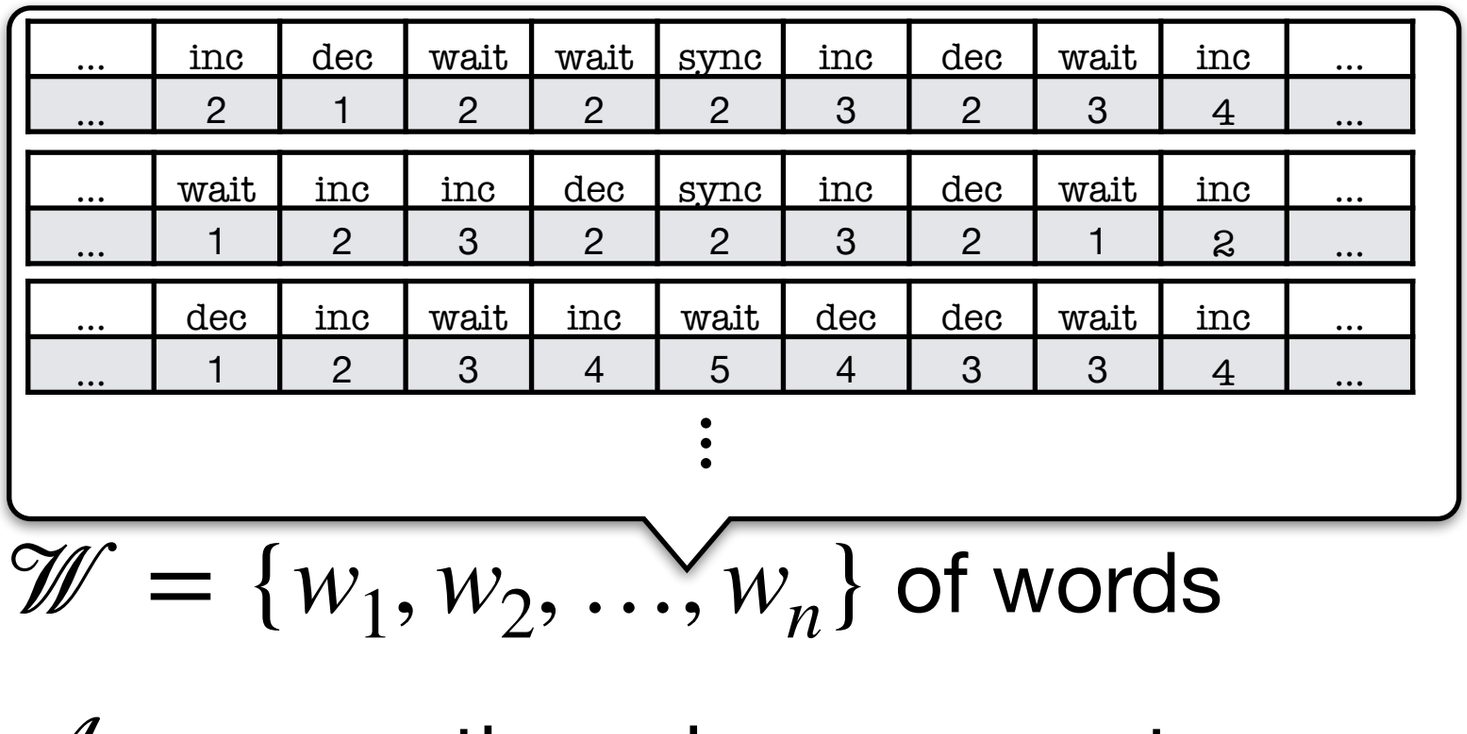
- Finite set $\mathscr{W} = \{w_1, w_2, \ldots, w_n\}$ of words
- A pattern $\mathscr{A}$ representing a hyperproperty

## Output

- Set of tuples of subwords in $\mathscr{W}$ matching $\mathscr{A}$

# Hyper Pattern Matching

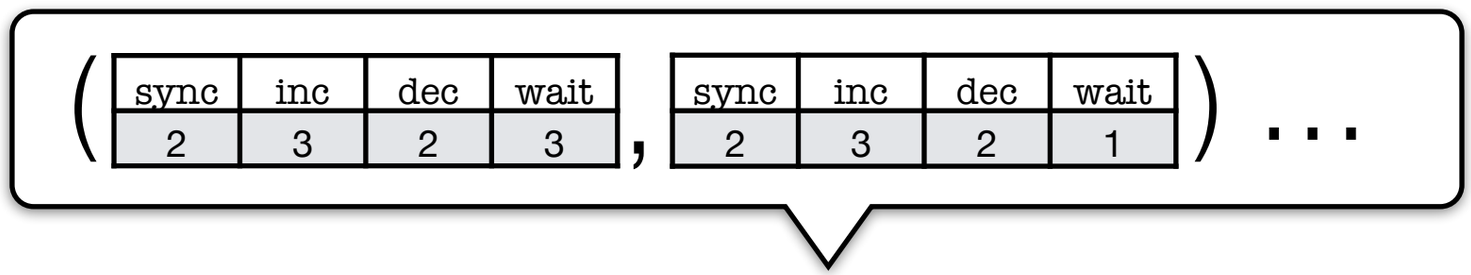
[Contribution]




## Inputs

- Finite set $\mathscr{W} = \{w_1, w_2, \ldots, w_n\}$ of words
- A pattern $\mathscr{A}$ representing a hyperproperty

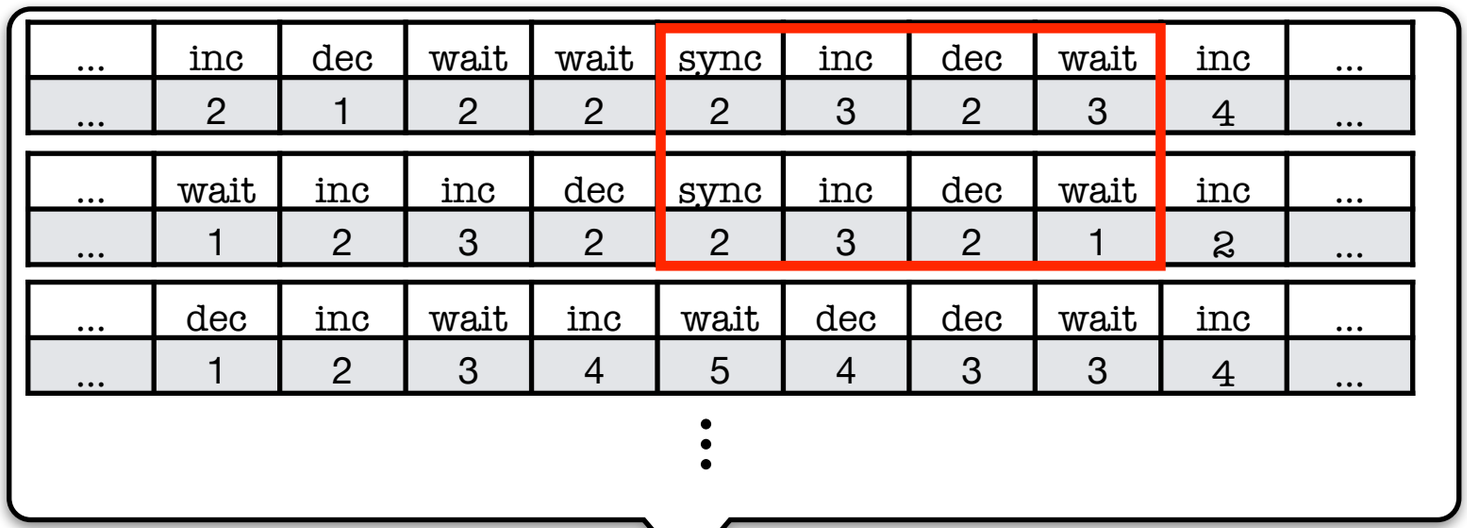
## Output




- Set of tuples of subwords in $\mathscr{W}$ matching $\mathscr{A}$

# Hyper Pattern Matching

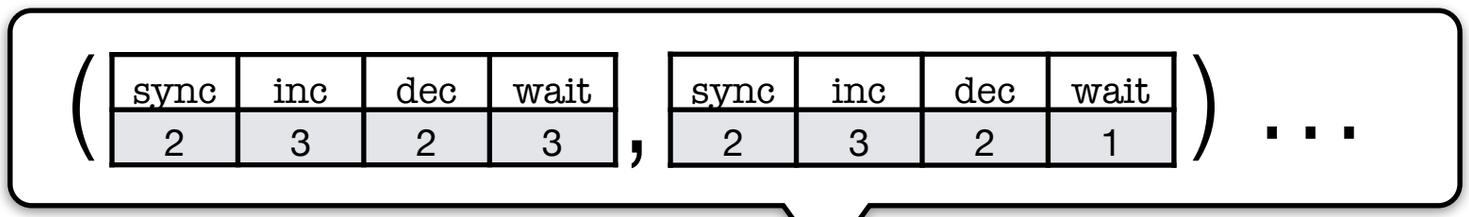
[Contribution]




## Inputs

- Finite set $\mathscr{W} = \{w_1, w_2, \ldots, w_n\}$ of words
- A pattern $\mathscr{A}$ representing a hyperproperty

## Output




- Set of tuples of subwords in $\mathscr{W}$ matching $\mathscr{A}$

# Contributions

Automata with multiple “tracks”

- Introduce hyper pattern matching with asynchronous automata (aka multi-tape automata)

Incrementally handles streams of words

- Propose incremental algorithms for hyper pattern matching
  - Naive algorithm + heuristics for better efficiency
- Works for reasonable size of data
  - e.g. words of length 1500 around 10 sec

# Related Work

e.g. stuttering reduction

| | Spec. | Results | Asynchronous hyperperoperties? |
|---|---|---|---|
| **[Ours]** | Asynchronous Automata | Tuples of intervals | **Yes** |
| **[Finkbeiner+, 2021]** | HyperLTL | Tuple of traces | **No** |
| **[Aceto+, 2024]** | Hyper-recHML | Boolean verdict | **No** |
| **[Chalupa+, 2024]** | Extended Hypernode logic | Boolean verdict | **Yes** |
| **[Bird, 1977]** | Two-dimensional array | Tuples of indices | **No** |

# Outline

- The hyper pattern matching problem
- A naive algorithm
- Heuristics by skipping unnecessary matching trials
- Experiments

# Asynchronous Automata

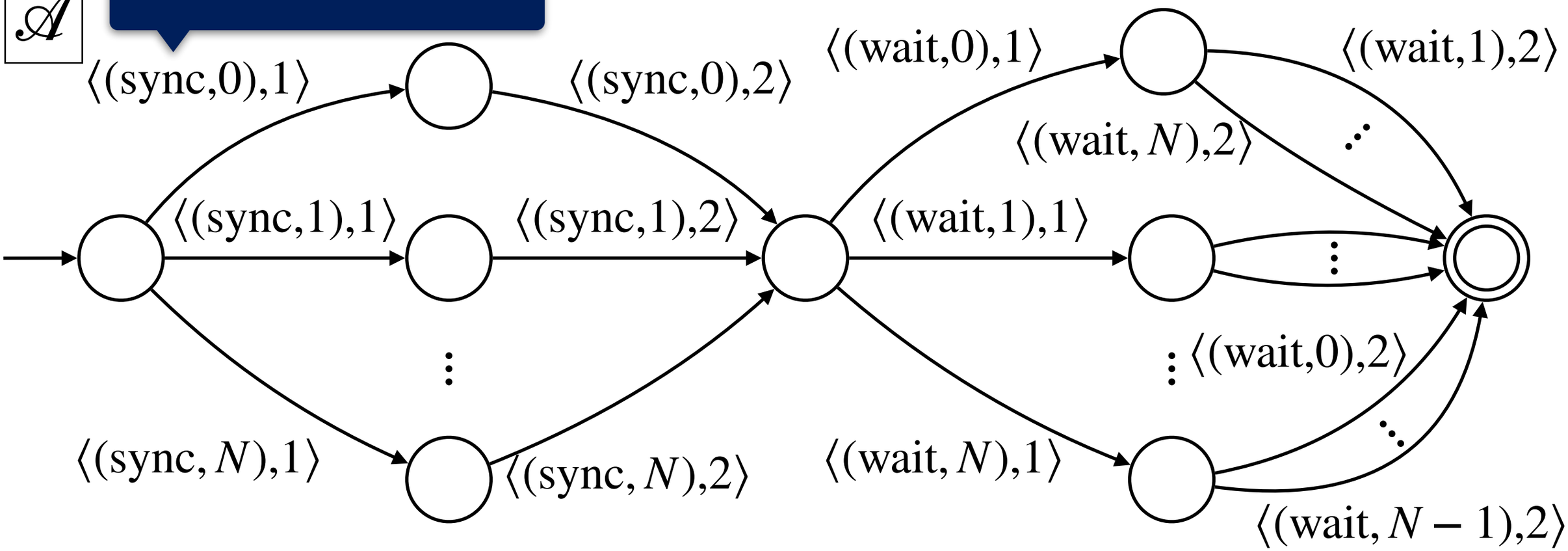
[Rabin & Scott, 1959; Gutsfeld+, 2021]

NFA over indexed alphabet to accept tuples of words

$\mathscr{A}$

(sync,0) on track 1

**Pattern** Update in `wait` just after `sync` deviates




$$\left(\begin{array}{|c|c|}\hline \texttt{sync} & \texttt{wait} \\ \hline 0 & 0 \\ \hline \end{array}, \begin{array}{|c|c|}\hline \texttt{sync} & \texttt{wait} \\ \hline 0 & 1 \\ \hline \end{array}\right) \in L(\mathscr{A})$$

# Asynchronous Automata

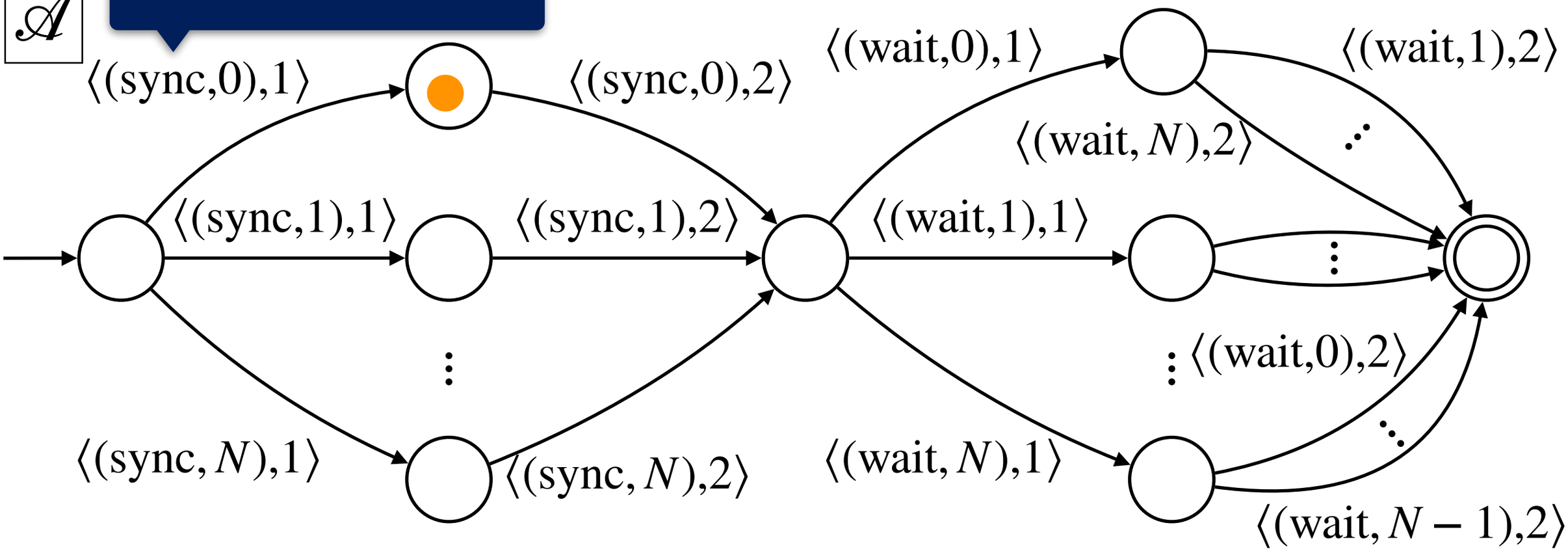
[Rabin & Scott, 1959; Gutsfeld+, 2021]

NFA over indexed alphabet to accept tuples of words

$\mathscr{A}$

(sync,0) on track 1

**Pattern** Update in wait just after sync deviates




$$\left(\begin{array}{|c|c|}\hline \text{sync} & \text{wait} \\ \hline 0 & 0 \\ \hline \end{array}\;,\;\begin{array}{|c|c|}\hline \text{sync} & \text{wait} \\ \hline 0 & 1 \\ \hline \end{array}\right) \in L(\mathscr{A})$$

# Asynchronous Automata

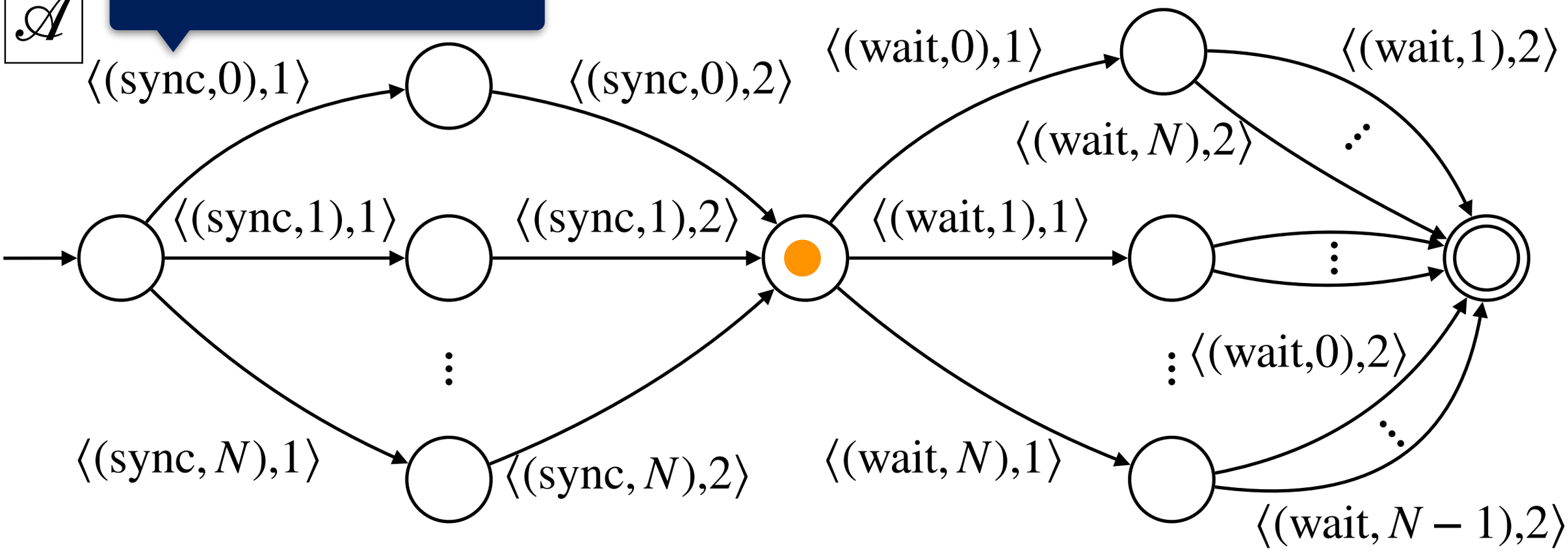
[Rabin & Scott, 1959; Gutsfeld+, 2021]

NFA over indexed alphabet to accept tuples of words

$\mathscr{A}$

(sync,0) on track 1

**Pattern** Update in wait just after sync deviates




$$\left(\begin{array}{|c|c|}\hline \text{sync} & \text{wait} \\ \hline 0 & 0 \\ \hline \end{array}\;,\;\begin{array}{|c|c|}\hline \text{sync} & \text{wait} \\ \hline 0 & 1 \\ \hline \end{array}\right) \in L(\mathscr{A})$$

# Asynchronous Automata

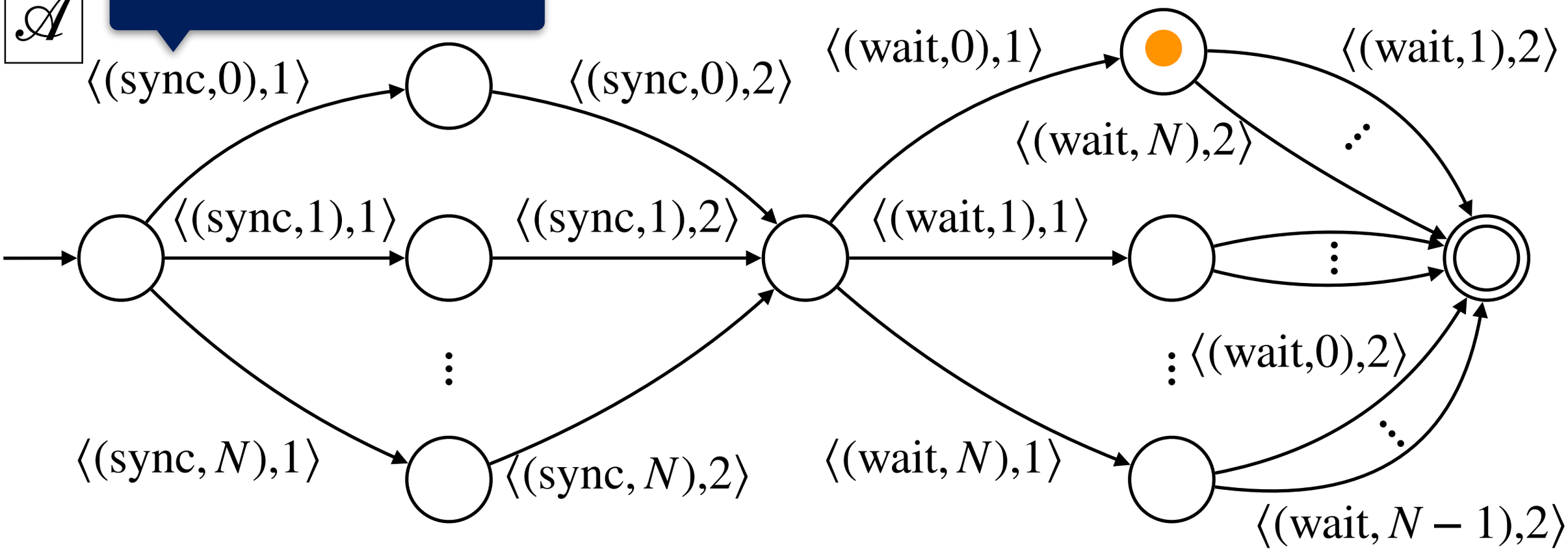
[Rabin & Scott, 1959; Gutsfeld+, 2021]

NFA over indexed alphabet to accept tuples of words

$\mathscr{A}$

(sync,0) on track 1

**Pattern** Update in wait just after sync deviates




( | sync | wait | | sync | wait | ) $\in L(\mathscr{A})$

| | |
|---|---|
| 0 | 0 |

| | |
|---|---|
| 0 | 1 |

# Asynchronous Automata

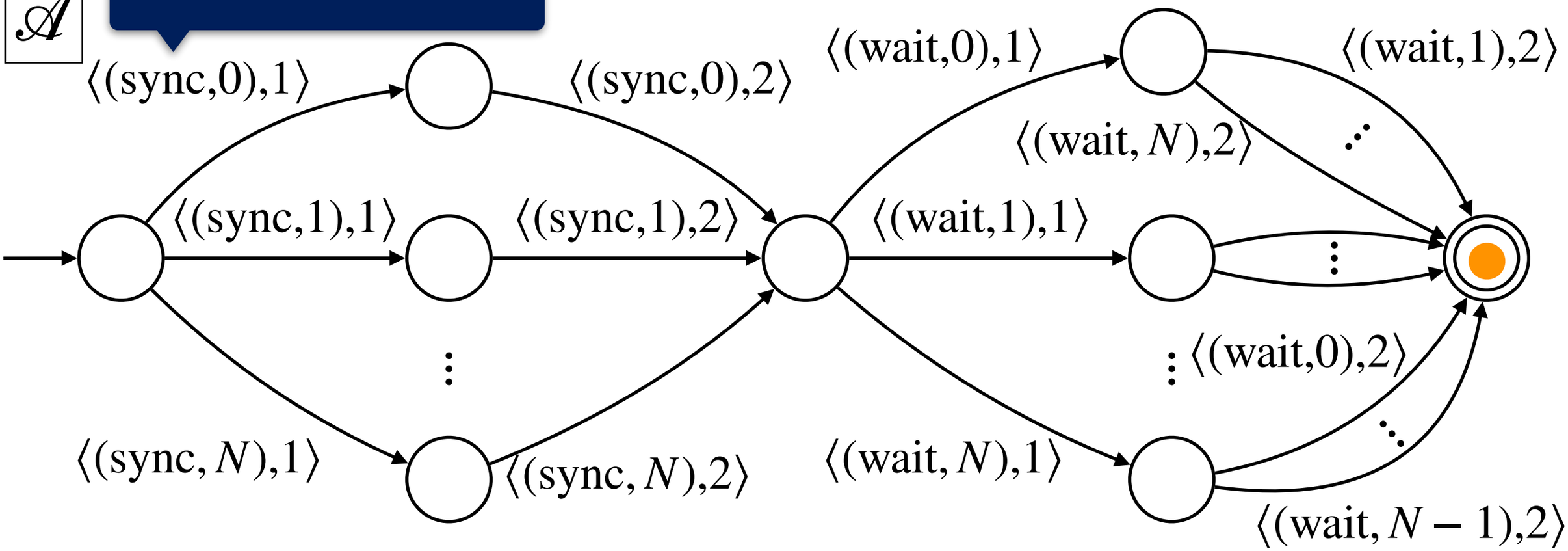
[Rabin & Scott, 1959; Gutsfeld+, 2021]

NFA over indexed alphabet to accept tuples of words

$\mathscr{A}$

(sync,0) on track 1

**Pattern** Update in wait just after sync deviates




$$\left(\begin{array}{|c|c|}\hline \text{sync} & \text{wait} \\ \hline 0 & 0 \\ \hline \end{array}, \begin{array}{|c|c|}\hline \text{sync} & \text{wait} \\ \hline 0 & 1 \\ \hline \end{array}\right) \in L(\mathscr{A})$$

# Hyper Pattern Matching

[Contribution]

## Inputs

- Finite set $\mathscr{W} = \{w_1, w_2, \ldots, w_n\}$ of words
- An asynchronous automaton $\mathscr{A}$ with $k$-tracks

## Output

Interval on $w^1$

Member of $\mathscr{W}$

- Set of $k$-tuples $\langle (i^1, j^1, w^1), \ldots, (i^k, j^k, w^k) \rangle$ s.t. $\langle w^1|_{[i^1,j^1]}, \ldots w^k|_{[i^k,j^k]} \rangle \in L(\mathscr{A})$

Restriction of $w^1$ on $[i^1, j^1]$

# Example of Hyper Pattern Matching

**Pattern** Update in `wait` just after `sync` deviates

| sync | wait | inc | dec | wait | sync | wait | dec | dec | sync | wait |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 1 | 1 | 1 | 2 | 1 | 0 | 0 | 0 |

| sync | inc | dec | wait | sync | wait | Inc | inc | dec | inc | dec |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 2 | 1 | 2 | 1 |

# Example of Hyper Pattern Matching

**Pattern** Update in `wait` just after `sync` deviates

| sync | wait | inc | dec | wait | sync | wait | dec | dec | sync | wait |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 1 | 1 | 1 | 2 | 1 | 0 | 0 | 0 |

| sync | inc | dec | wait | sync | wait | Inc | inc | dec | inc | dec |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 2 | 1 | 2 | 1 |

**Matching!**

# Example of Hyper Pattern Matching

**Pattern** Update in wait just after sync deviates

Matching!

| sync | wait | inc | dec | wait | sync | wait | dec | dec | sync | wait |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 1 | 1 | 1 | 2 | 1 | 0 | 0 | 0 |

| sync | inc | dec | wait | sync | wait | Inc | inc | dec | inc | dec |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 2 | 1 | 2 | 1 |

Matching!

# Outline

- The hyper pattern matching problem
- A naive algorithm
- Heuristics by skipping unnecessary matching trials
- Experiments

# Naive Algorithm for Hyper Pattern Matching

1. Pick the starting positions

2. Feed letters to the asynchronous automaton $\mathscr{A}$

   - Report matching if we reach an accepting state

3. Move the starting position after matching trial
   → Go back to Step 2

# Naive Algorithm for Hyper Pattern Matching

## 1. Pick the starting position

**Pattern** Update in `wait` just after `sync` deviates (w/ 2 tracks)

$w_1 =$

| sync | wait | inc | dec | wait | sync | wait | dec | dec | sync | wait |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 1 | 1 | 1 | 2 | 1 | 0 | 0 | 0 |

$w_2 =$

| sync | inc | dec | wait | sync | wait | inc | inc | dec | inc | dec |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 2 | 1 | 2 | 1 |

# Naive Algorithm for Hyper Pattern Matching

## 1. Pick the starting position

**Pattern** Update in `wait` just after `sync` deviates (w/ 2 tracks)

$$(w_1, w_1), (w_1, w_2), (w_2, w_1), (w_2, w_2)$$

$w_1 =$

| sync | wait | inc | dec | wait | sync | wait | dec | dec | sync | wait |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 1 | 1 | 1 | 2 | 1 | 0 | 0 | 0 |

$w_2 =$

| sync | inc | dec | wait | sync | wait | inc | inc | dec | inc | dec |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 2 | 1 | 2 | 1 |

# Naive Algorithm for Hyper Pattern Matching

## 1. Pick the starting position

**Pattern** Update in `wait` just after `sync` deviates (w/ 2 tracks)

$$(w_1, w_1), (w_1, w_2), (w_2, w_1), (w_2, w_2)$$

Pick this in this example

$w_1 =$

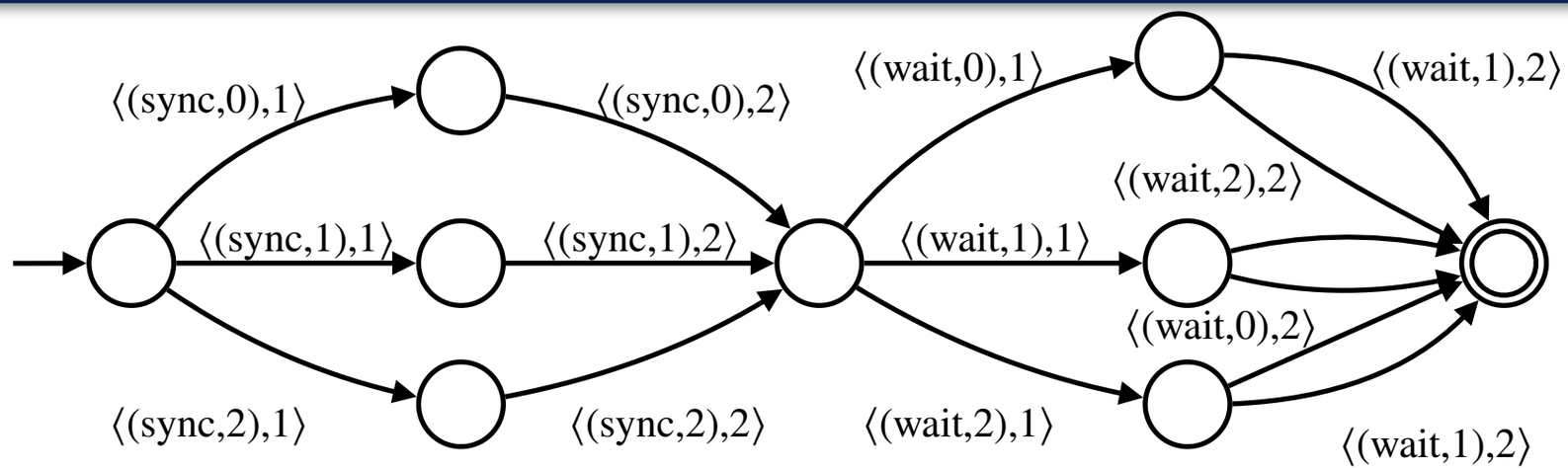
| sync | wait | inc | dec | wait | sync | wait | dec | dec | sync | wait |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 1 | 1 | 1 | 2 | 1 | 0 | 0 | 0 |

$w_2 =$

| sync | inc | dec | wait | sync | wait | inc | inc | dec | inc | dec |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 2 | 1 | 2 | 1 |

# Naive Algorithm for Hyper Pattern Matching

## 2. Feed letters to $\mathscr{A}$




$i^1$ $j^1$

$w_1 =$

| sync | wait | inc | dec | wait | sync | wait | dec | dec | sync | wait |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 1 | 1 | 1 | 2 | 1 | 0 | 0 | 0 |

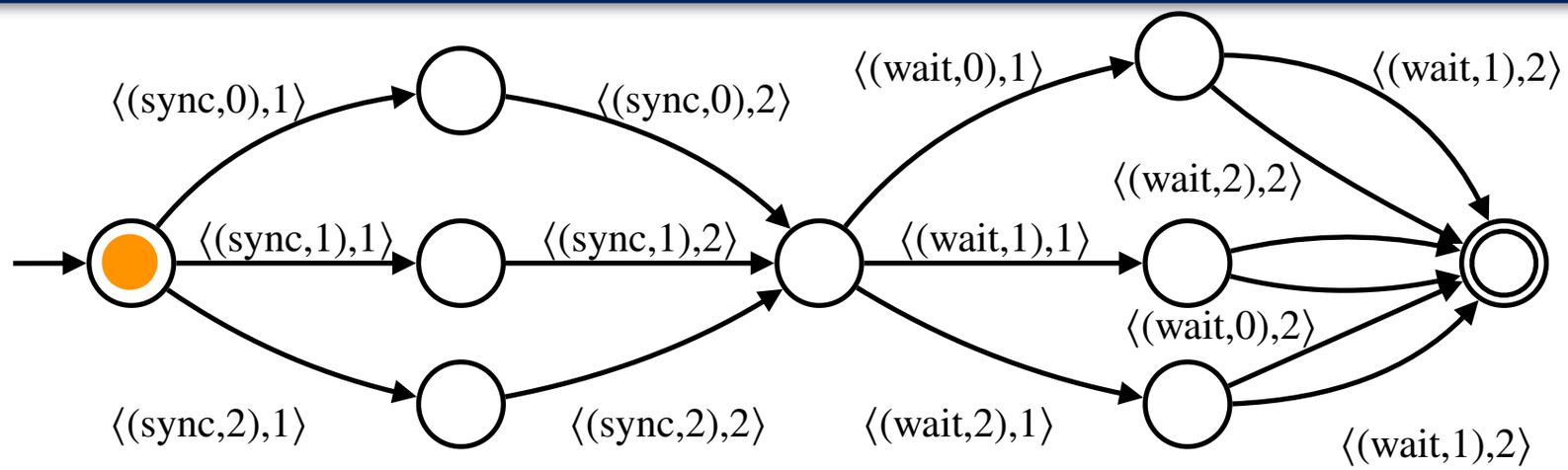
$w_2 =$

| sync | inc | dec | wait | sync | wait | inc | inc | dec | inc | dec |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 2 | 1 | 2 | 1 |

$i^2$ $j^2$

# Naive Algorithm for Hyper Pattern Matching

## 2. Feed letters to $\mathscr{A}$




$i^1$ $j^1$

$w_1 =$

| sync | wait | inc | dec | wait | sync | wait | dec | dec | sync | wait |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 1 | 1 | 1 | 2 | 1 | 0 | 0 | 0 |

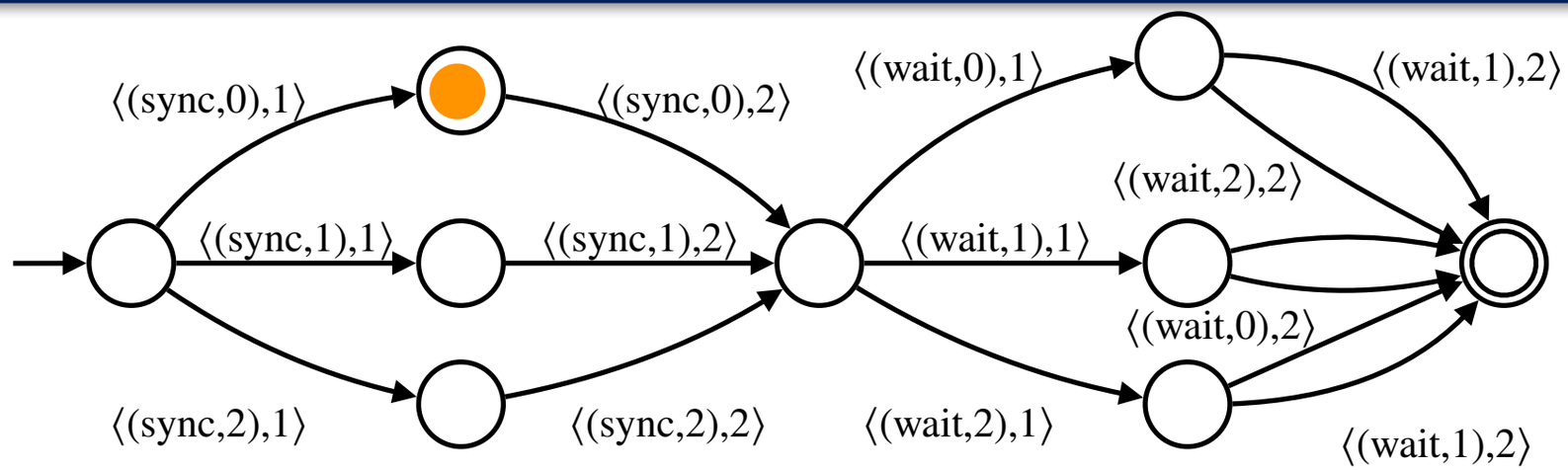
$w_2 =$

| sync | inc | dec | wait | sync | wait | inc | inc | dec | inc | dec |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 2 | 1 | 2 | 1 |

$i^2$ $j^2$

# Naive Algorithm for Hyper Pattern Matching

## 2. Feed letters to $\mathscr{A}$




$i^1$ $j^1$

$w_1 =$

| sync | wait | inc | dec | wait | sync | wait | dec | dec | sync | wait |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 1 | 1 | 1 | 2 | 1 | 0 | 0 | 0 |

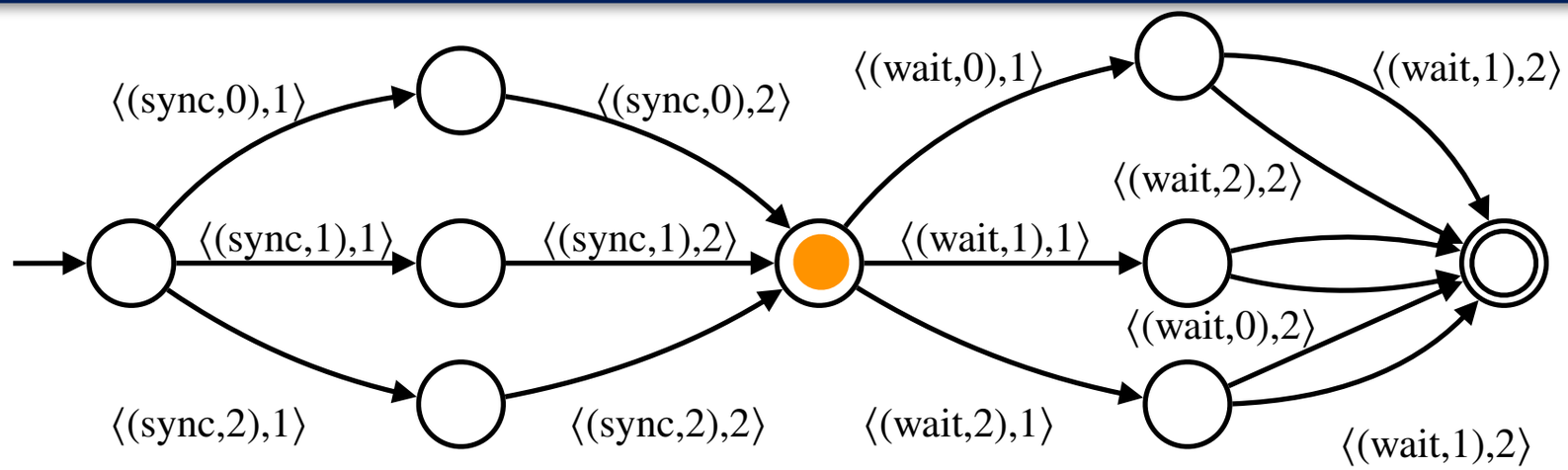
$w_2 =$

| sync | inc | dec | wait | sync | wait | inc | inc | dec | inc | dec |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 2 | 1 | 2 | 1 |

$i^2$ $j^2$

# Naive Algorithm for Hyper Pattern Matching

## 2. Feed letters to $\mathscr{A}$




$i^1$ $j^1$

$w_1 =$

| sync | wait | inc | dec | wait | sync | wait | dec | dec | sync | wait |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 1 | 1 | 1 | 2 | 1 | 0 | 0 | 0 |

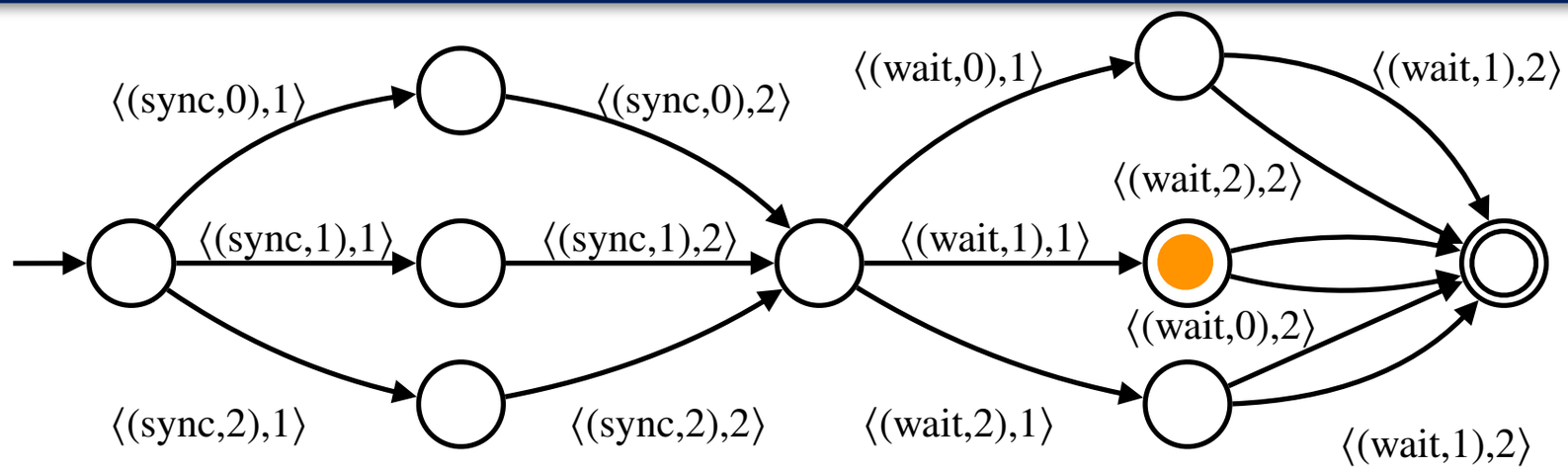
$w_2 =$

| sync | inc | dec | wait | sync | wait | inc | inc | dec | inc | dec |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 2 | 1 | 2 | 1 |

$i^2$ $j^2$

# Naive Algorithm for Hyper Pattern Matching

## 2. Feed letters to $\mathscr{A}$




$i^1$ $j^1$

$w_1 =$

| sync | wait | inc | dec | wait | sync | wait | dec | dec | sync | wait |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 1 | 1 | 1 | 2 | 1 | 0 | 0 | 0 |

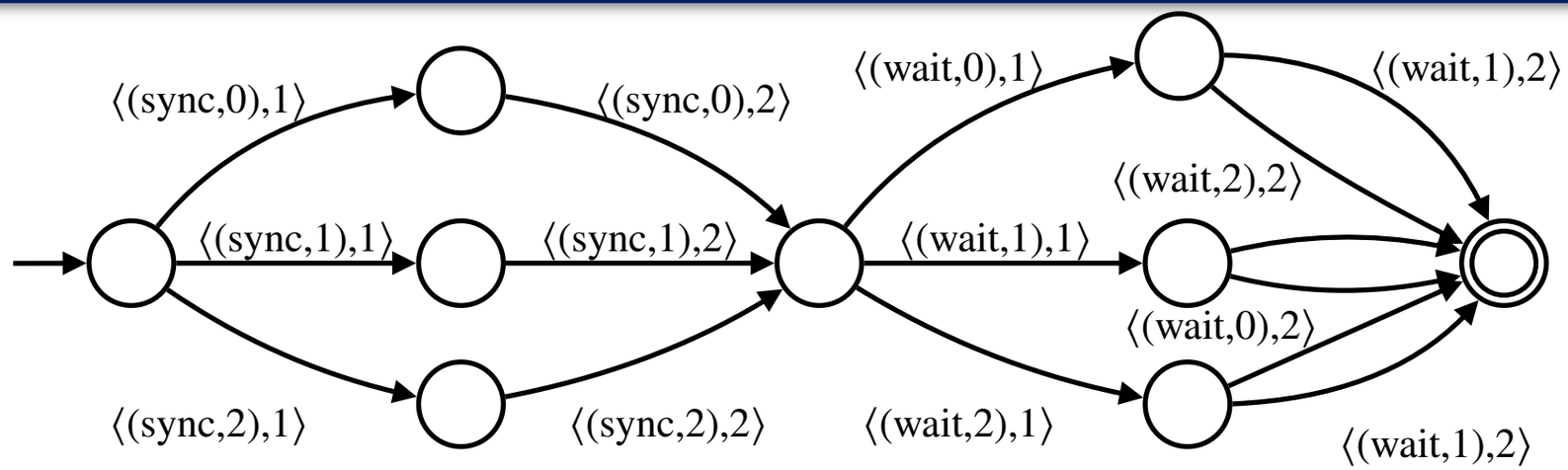
$w_2 =$

| sync | inc | dec | wait | sync | wait | inc | inc | dec | inc | dec |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 2 | 1 | 2 | 1 |

$i^2$ $j^2$

# Naive Algorithm for Hyper Pattern Matching

## 2. Feed letters to $\mathscr{A}$




$i^1$ $j^1$

$w_1 =$

| sync | wait | inc | dec | wait | sync | wait | dec | dec | sync | wait |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 1 | 1 | 1 | 2 | 1 | 0 | 0 | 0 |

$w_2 =$

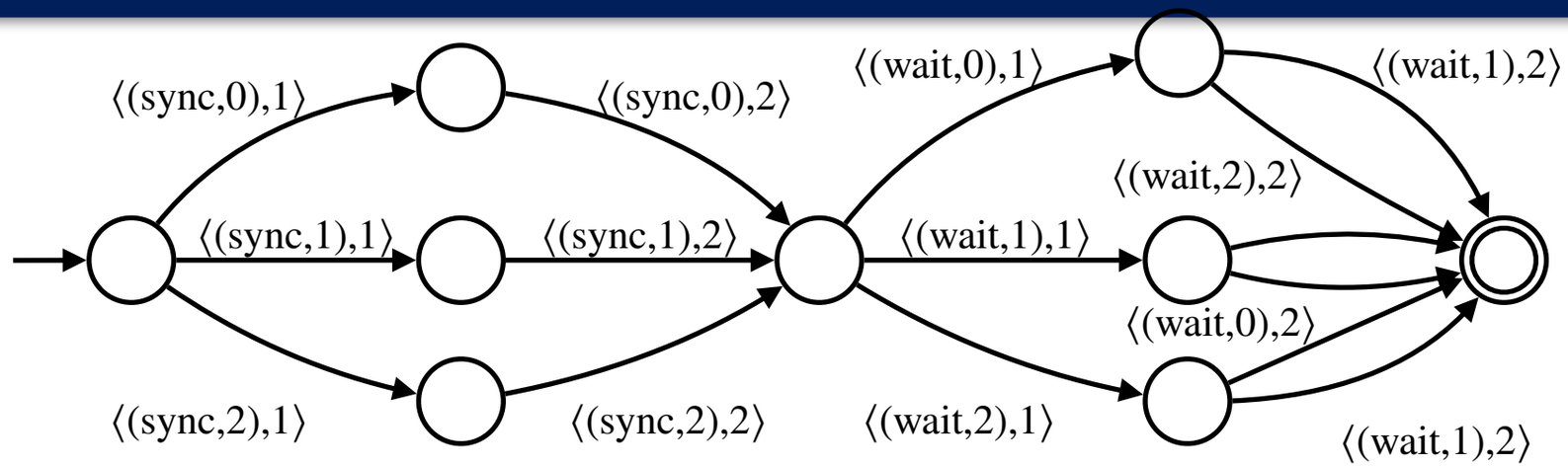
| sync | inc | dec | wait | sync | wait | inc | inc | dec | inc | dec |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 2 | 1 | 2 | 1 |

$i^2$ $j^2$

No Successor!

# Naive Algorithm for Hyper Pattern Matching

## 3. Move the starting position after matching trial




$i^1$ $i^1$

$w_1 =$

| sync | wait | | dec | wait | sync | wait | dec | dec | sync | wait |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | | | 1 | 1 | 1 | 2 | 1 | 0 | 0 | 0 |

We try both with BFS

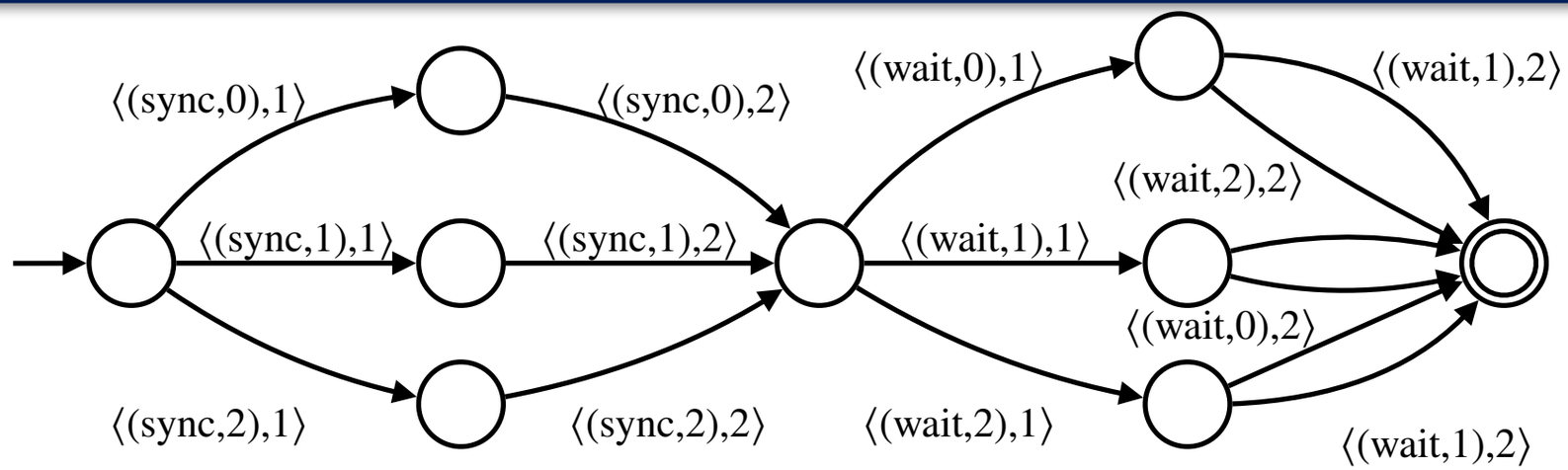
$w_2 =$

| sync | inc | dec | wait | sync | wait | inc | inc | dec | inc | dec |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 2 | 1 | 2 | 1 |

$i^2$ $i^2$

# Naive Algorithm for Hyper Pattern Matching

## 2. Feed letters to $\mathscr{A}$




$i^1$ $j^1$

$w_1 =$

| sync | wait | inc | dec | wait | sync | wait | dec | dec | sync | wait |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 1 | 1 | 1 | 2 | 1 | 0 | 0 | 0 |

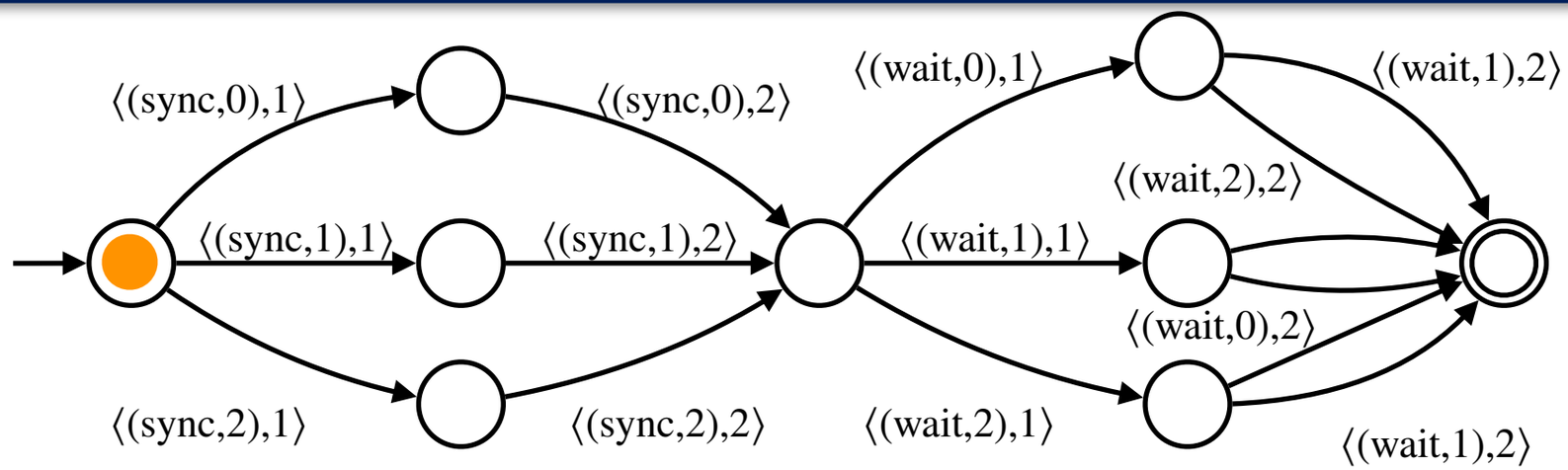
$w_2 =$

| sync | inc | dec | wait | sync | wait | inc | inc | dec | inc | dec |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 2 | 1 | 2 | 1 |

$i^2$ $j^2$

# Naive Algorithm for Hyper Pattern Matching

## 2. Feed letters to $\mathscr{A}$




$i^1$ $j^1$

$w_1 =$

| sync | wait | inc | dec | wait | sync | wait | dec | dec | sync | wait |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 1 | 1 | 1 | 2 | 1 | 0 | 0 | 0 |

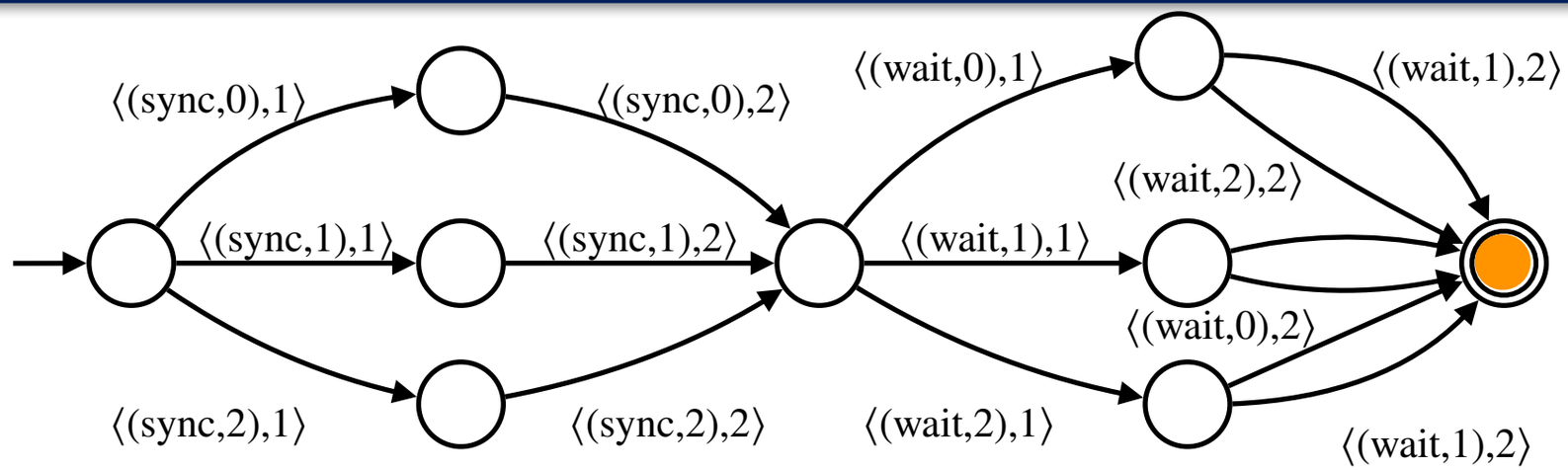
$w_2 =$

| sync | inc | dec | wait | sync | wait | inc | inc | dec | inc | dec |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 2 | 1 | 2 | 1 |

$i^2$ $j^2$

# Naive Algorithm for Hyper Pattern Matching

## 2. Feed letters to $\mathscr{A}$




$i^1$ $j^1$

$w_1 =$

| sync | wait | inc | dec | wait | sync | wait | dec | dec | sync | wait |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 1 | 1 | 1 | 2 | 1 | 0 | 0 | 0 |

$w_2 =$

| sync | inc | dec | wait | sync | wait | inc | inc | dec | inc | dec |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 2 | 1 | 2 | 1 |

$i^2$ $j^2$

Accepted!
→ reported

# Complexity Analysis

**Theorem**

The naive algorithm has time complexity
$\mathcal{O}(|\mathscr{W}|^k \times \max_{w \in \mathscr{W}} |w|^{k+1} \times |S|^2)$
where $k$: number of tracks $|S|$: number of states of $\mathscr{A}$

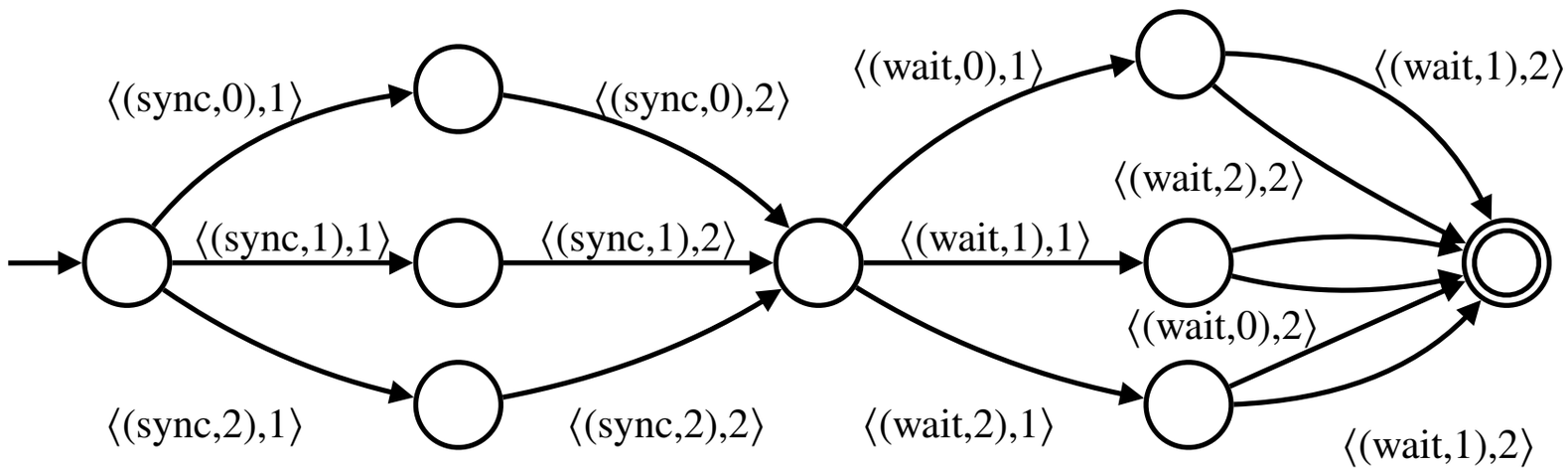
**Theorem**

Deciding the nonemptiness of the result of hyper pattern matching is NP-complete

# Outline

- The hyper pattern matching problem
- A naive algorithm
- Heuristics by skipping unnecessary matching trials
- Experiments

# Q. Can we reduce the number of matching trials?




$i^1$ $i^1$

$w_1 =$

| sync | wait | inc | dec | wait | sync | wait | dec | dec | sync | wait |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 1 | 1 | 1 | 2 | 1 | 0 | 0 | 0 |

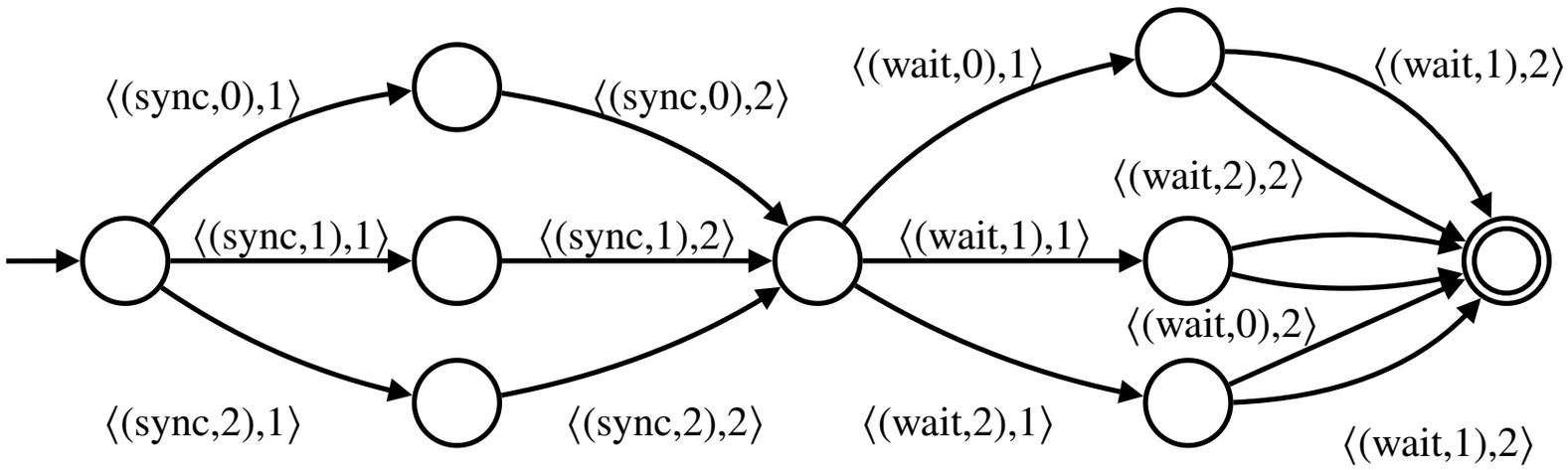
$w_2 =$

| sync | inc | dec | wait | sync | wait | inc | inc | dec | inc | dec |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 2 | 1 | 2 | 1 |

$i^2$ $i^2$

# Q. Can we reduce the number of matching trials?

Can we move by >1?




$i^1$

| | sync | wait | inc | dec | wait | sync | wait | dec | dec | sync | wait |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $w_1 =$ | 0 | 1 | 2 | 1 | 1 | 1 | 2 | 1 | 0 | 0 | 0 |

| | sync | inc | dec | wait | sync | wait | inc | inc | dec | inc | dec |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $w_2 =$ | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 2 | 1 | 2 | 1 |

$i^2$

# Skipping Matching Trials

Idea: Pruning w/ fast precomputation

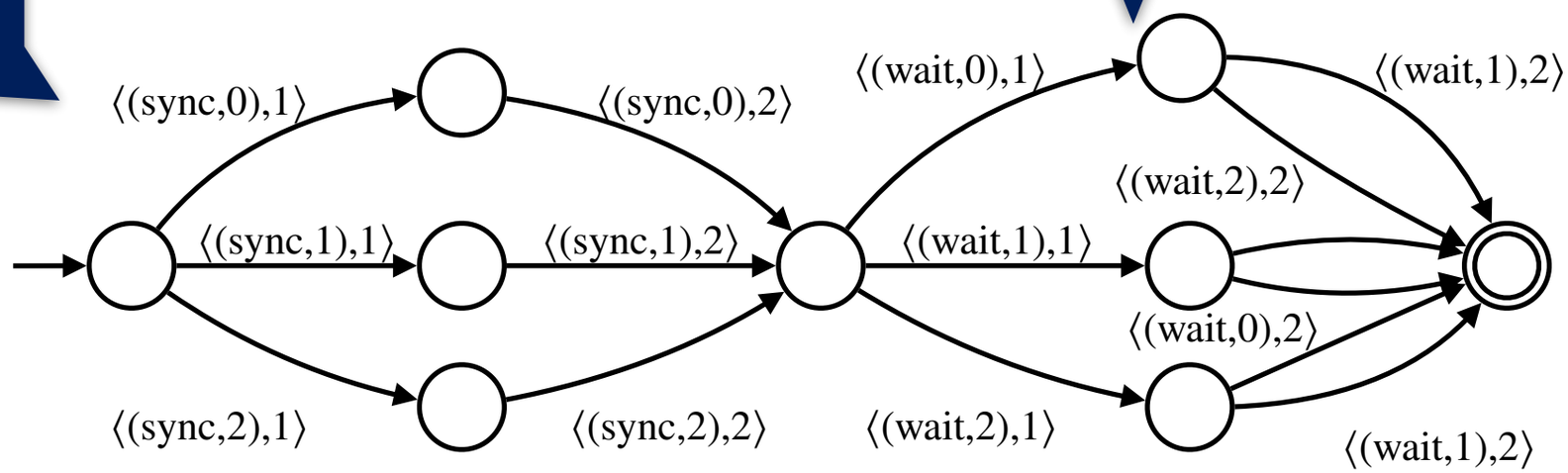
- Unnecessary matching trials can be skipped based on
  - look ahead of some letters
  - states reached during latest matching trials
  - track-wise pattern matching of projected patterns
- Similar to some string matching algorithms
  - e.g. KMP, BM, Quick Search, …

Explain this!

# Skipping based on lookahead

Track-wise matching
length = 2

last letter: (wait, $n$)




$i^1$

| | sync | wait | inc | dec | wait | sync | wait | dec | dec | sync | wait |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $w_1 =$ | 0 | 1 | 2 | 1 | 1 | 1 | 2 | 1 | 0 | 0 | 0 |

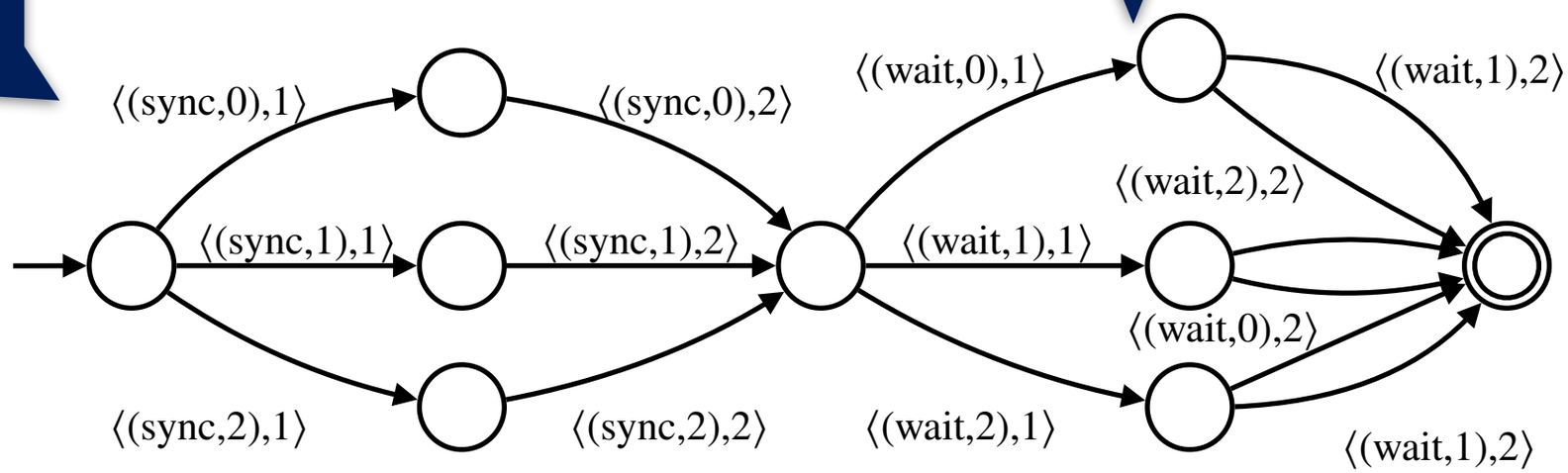
| | sync | inc | dec | wait | sync | wait | inc | inc | dec | inc | dec |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $w_2 =$ | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 2 | 1 | 2 | 1 |

$i^2$

# Skipping based on lookahead

Track-wise matching
length = 2

last letter: $(\mathrm{wait}, n)$




2nd letter is not $(\mathrm{wait}, n)$
→ No Matching!

$i^1$

| $w_1 =$ | sync | wait | inc | dec | wait | sync | wait | dec | dec | sync | wait |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 1 | 1 | 1 | 2 | 1 | 0 | 0 | 0 |

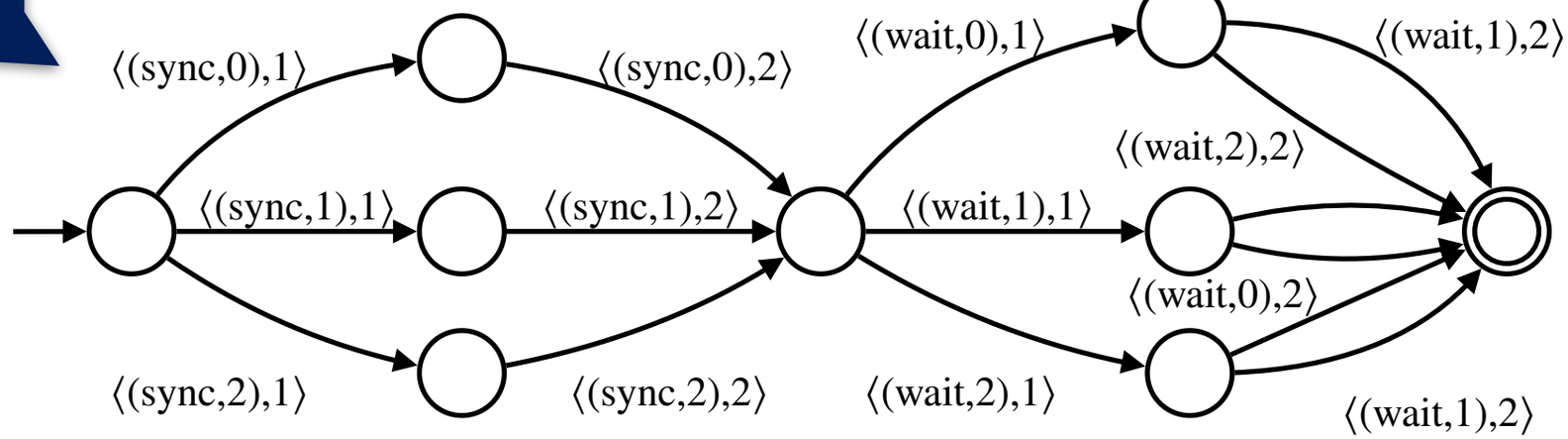
| $w_2 =$ | sync | inc | dec | wait | sync | wait | inc | inc | dec | inc | dec |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 2 | 1 | 2 | 1 |

$i^2$

# Skipping based on lookahead

Track-wise matching length = 2

last letter: $(\mathrm{wait}, n)$




2nd letter is not $(\mathrm{wait}, n)$
→ No Matching!

$(\mathrm{dec},0)$ cannot appear in matching
→ Skip to after this

$i^1$

| $w_1 =$ | sync | wait | inc | dec | wait | sync | wait | dec | dec | sync | wait |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | | | | | | 0 | 0 | 0 |

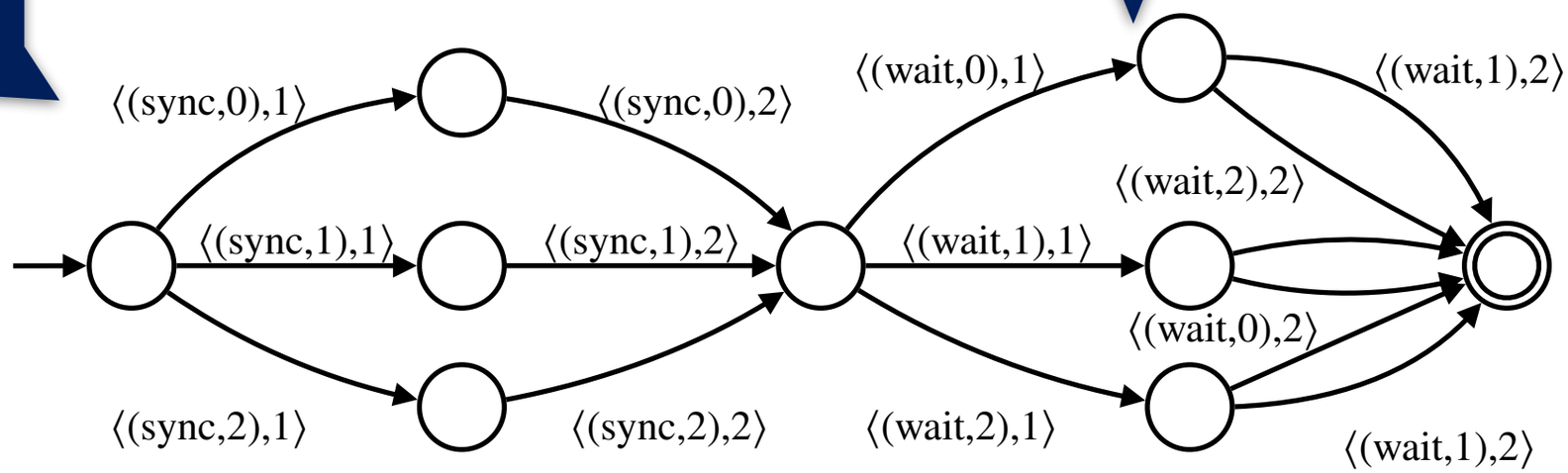
| $w_2 =$ | sync | inc | dec | wait | sync | wait | inc | inc | dec | inc | dec |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 2 | 1 | 2 | 1 |

$i^2$

# Skipping based on lookahead

Track-wise matching
length = 2

last letter: $(\mathrm{wait}, n)$




2nd letter is not $(\mathrm{wait}, n)$
→ No Matching!

$(\mathrm{dec},0)$ cannot appear in matching
→ Skip to after this

$i^1$

$w_1 =$

| sync | wait | inc | dec | wait | sync | wait | dec | dec | sync | wait |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | | | | | | 0 | 0 | 0 |

$w_2 =$

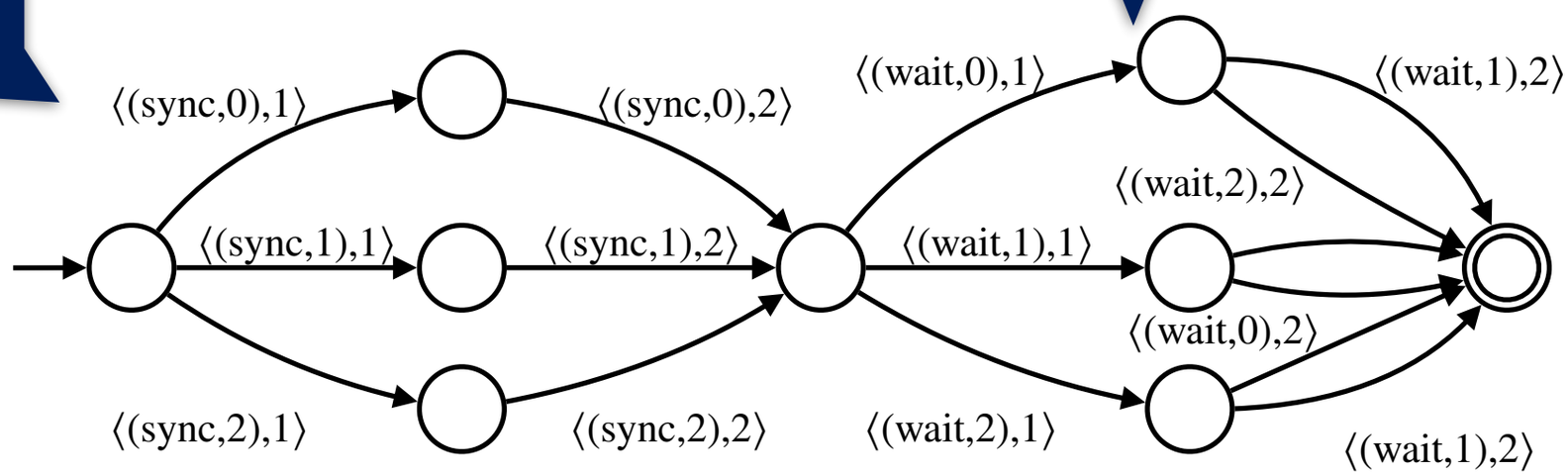
| sync | inc | dec | wait | sync | wait | inc | inc | dec | inc | dec |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 2 | 1 | 2 | 1 |

$i^2$

# Skipping based on lookahead

Track-wise matching
length = 2

last letter: (wait, $n$)




$i^1$

| | sync | wait | inc | dec | wait | sync | wait | dec | dec | sync | wait |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $w_1 =$ | 0 | 1 | 2 | 1 | 1 | 1 | 2 | 1 | 0 | 0 | 0 |

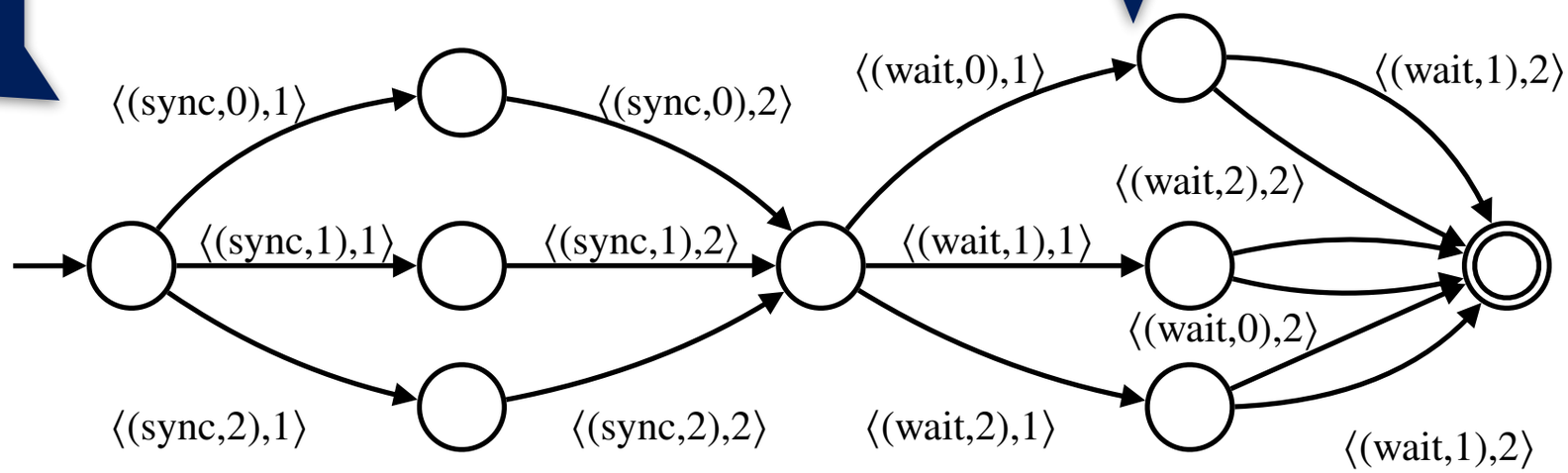
| | sync | inc | dec | wait | sync | wait | inc | inc | dec | inc | dec |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $w_2 =$ | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 2 | 1 | 2 | 1 |

$i^2$

# Skipping based on lookahead

Track-wise matching
length = 2

last letter: $(\mathrm{wait}, n)$




2nd letter is not $(\mathrm{wait}, n)$
→ No Matching!

$i^1$

| $w_1 =$ | sync | wait | inc | dec | wait | sync | wait | dec | dec | sync | wait |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 1 | 1 | 1 | 2 | 1 | 0 | 0 | 0 |

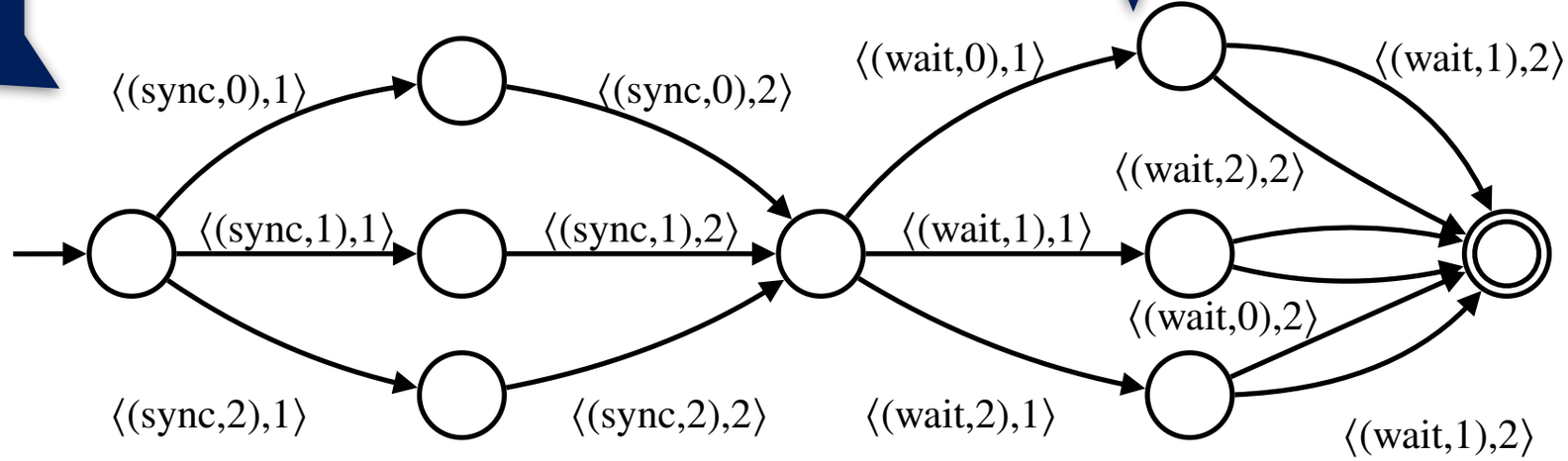
| $w_2 =$ | sync | inc | dec | wait | sync | wait | inc | inc | dec | inc | dec |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 2 | 1 | 2 | 1 |

$i^2$

# Skipping based on lookahead

Track-wise matching
length = 2

last letter: $(\mathrm{wait}, n)$




$i^1$

$w_1 =$

| sync | wait | inc | dec | wait | sync | wait | dec | dec | sync | wait |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 1 | 1 | 1 | | | | | |

$w_2 =$

| sync | inc | dec | wait | sync | wait | inc | inc | dec | inc | dec |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 2 | 1 | 2 | 1 |

$i^2$

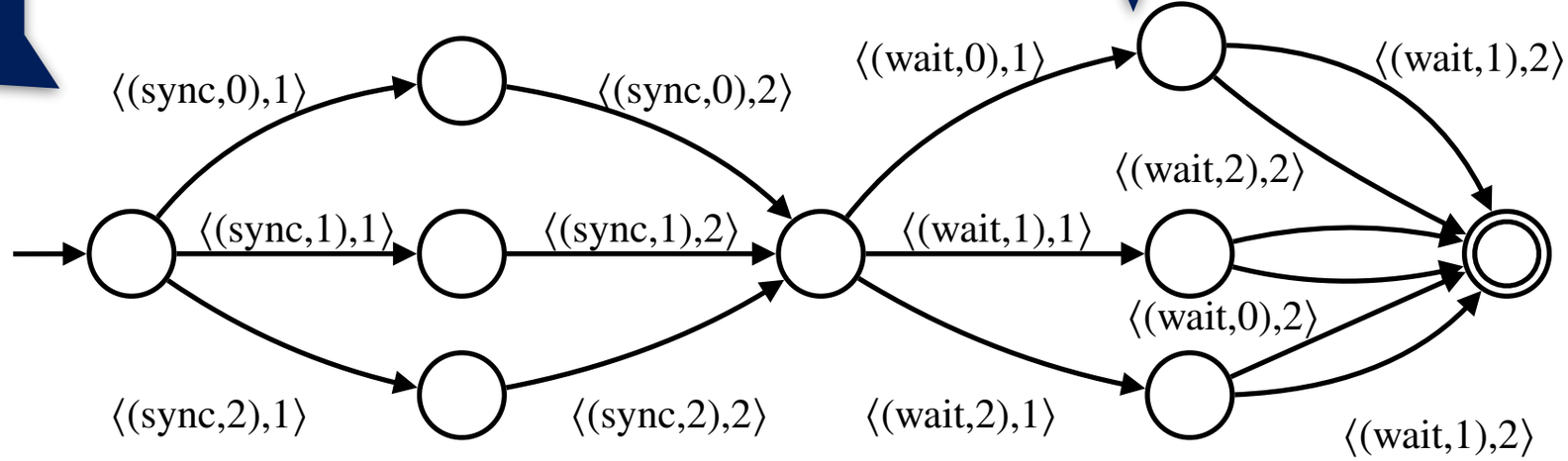
2nd letter is not $(\mathrm{wait}, n)$
→ No Matching!

$(\mathrm{wait},0)$ can be 2nd word in matching
→ Skip by 1

# Skipping based on lookahead

Track-wise matching
length = 2

last letter: $(\mathrm{wait}, n)$




$i^1$

| | sync | wait | inc | dec | wait | sync | wait | dec | dec | sync | wait |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $w_1 =$ | 0 | 1 | 2 | 1 | 1 | 1 | | | | | |

| | sync | inc | dec | wait | sync | wait | inc | inc | dec | inc | dec |
|---|---|---|---|---|---|---|---|---|---|---|---|
| $w_2 =$ | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 2 | 1 | 2 | 1 |

$i^2$

2nd letter is not $(\mathrm{wait}, n)$
→ No Matching!

$(\mathrm{wait},0)$ can be 2nd word in matching
→ Skip by 1

# Quick Search-Style Skipping

[Contribution]

## Skipping based on lookahead

For the shortest matching length $l^k$ for the $k$-th track,
look ahead $(i^k + l^k - 1)$-th letter

- start matching if it can be the $l^k$-th letter of $w \in L(\mathscr{A})$ projected to the $k$-th track
- otherwise look ahead $(i^k + l^k)$-th letter and shift $i^k$ by the minimum $j$ s.t. it can be the $(l^k + 1 - j)$-th letter of $w \in L(\mathscr{A})$ projected to $k$-th track

# Quick Search-Style Skipping

[Contribution]

## Skipping based on lookahead

For the shortest matching length $l^k$ for the $k$-th track,
look ahead $(i^k + l^k - 1)$-th letter

- start matching if it can be the $l^k$-th letter of $w \in L(\mathscr{A})$ projected to the $k$-th track
- otherwise look ahead $(i^k + l^k)$-th letter and shift $i^k$ by the minimum $j$ s.t. it can be the $(l^k + 1 - j)$-th letter of $w \in L(\mathscr{A})$ projected to $k$-th track

Independent of the monitored words
→ Precompute beforehand

# Generalize skip over tracks

We “invalidate” useless starting pos. dimension-wise
→ The skipping effect also exponentially “blows up”

$i^1$

$w_1 =$

| sync | wait | inc | dec | wait | sync | wait | dec | dec | sync | wait |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 1 | 1 | 1 | 2 | 1 | 0 | 0 | 0 |

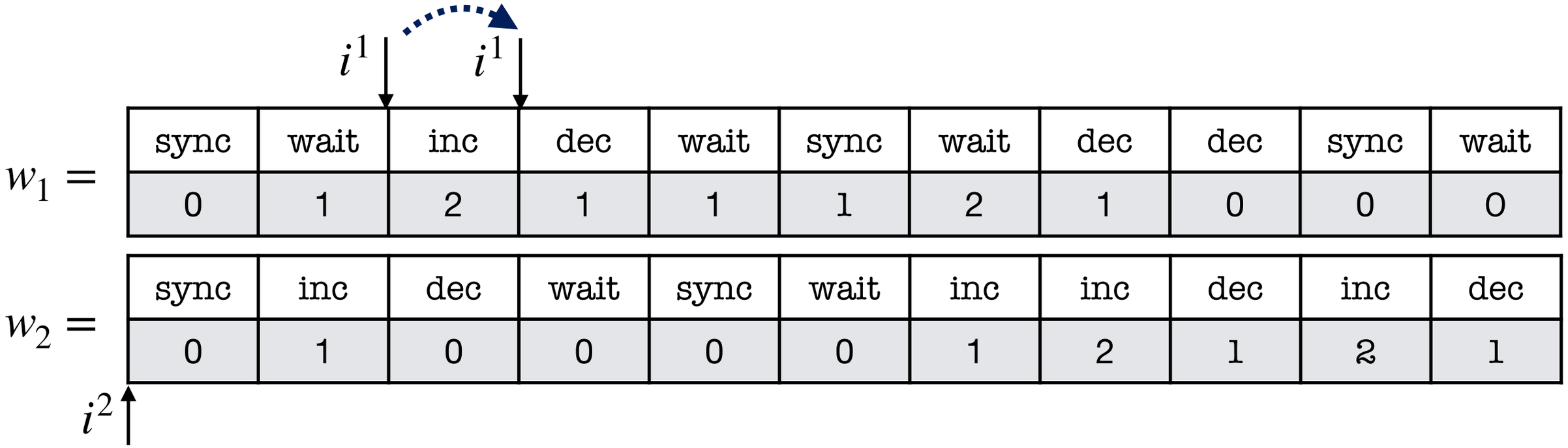
$w_2 =$

| sync | inc | dec | wait | sync | wait | inc | inc | dec | inc | dec |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 2 | 1 | 2 | 1 |

$i^2$

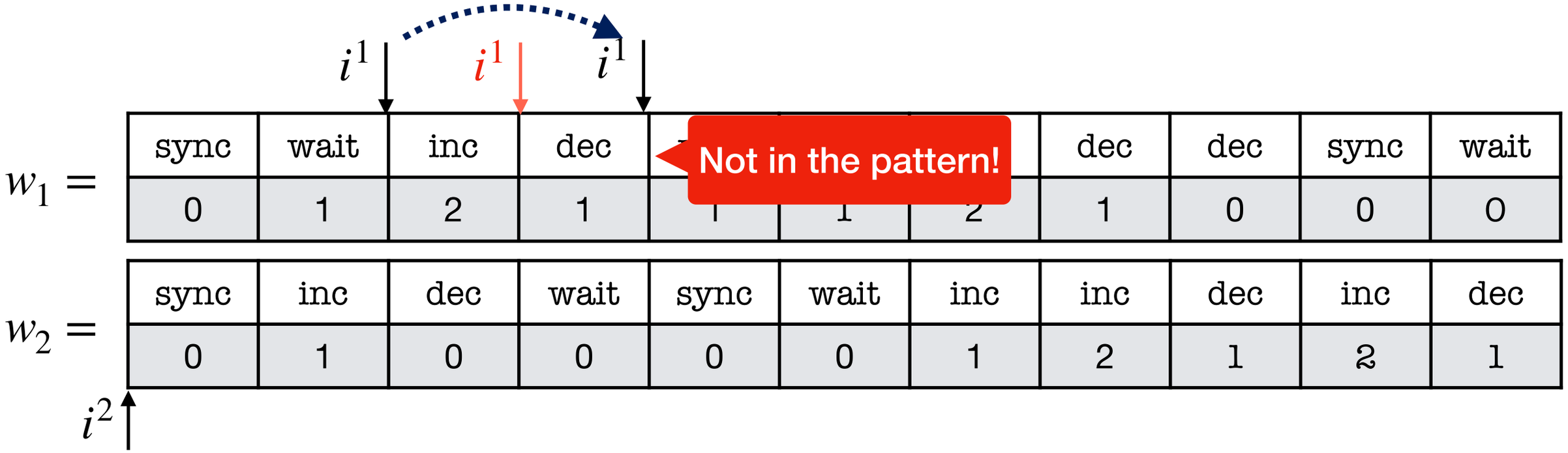
# Generalize skip over tracks

We “invalidate” useless starting pos. dimension-wise
→ The skipping effect also exponentially “blows up”



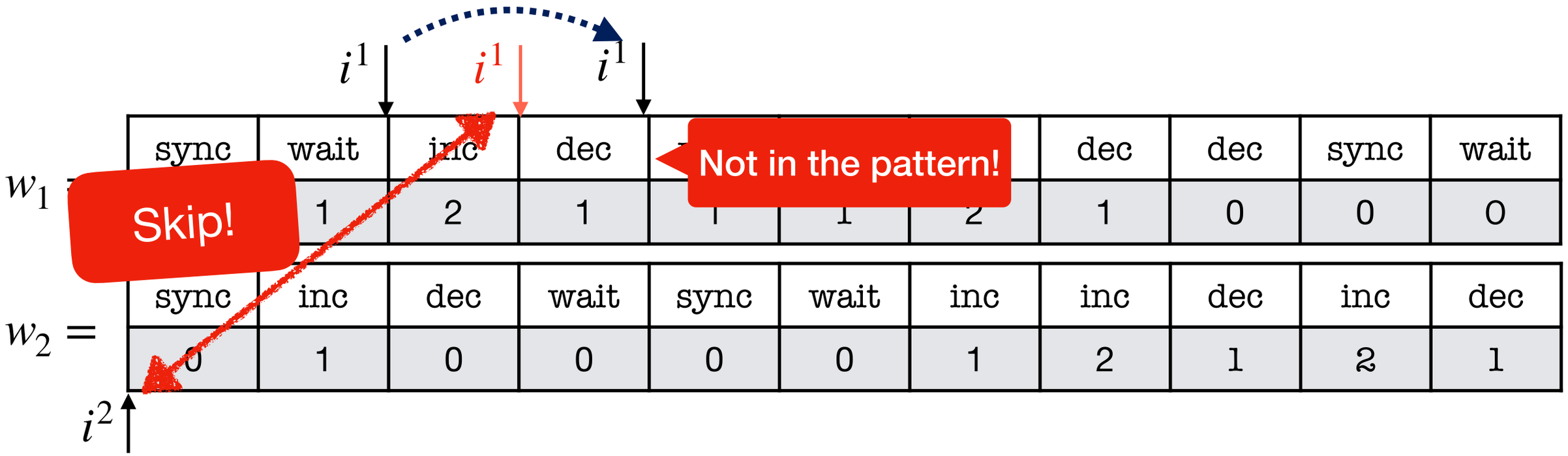
# Generalize skip over tracks

We “invalidate” useless starting pos. dimension-wise
→ The skipping effect also exponentially “blows up”



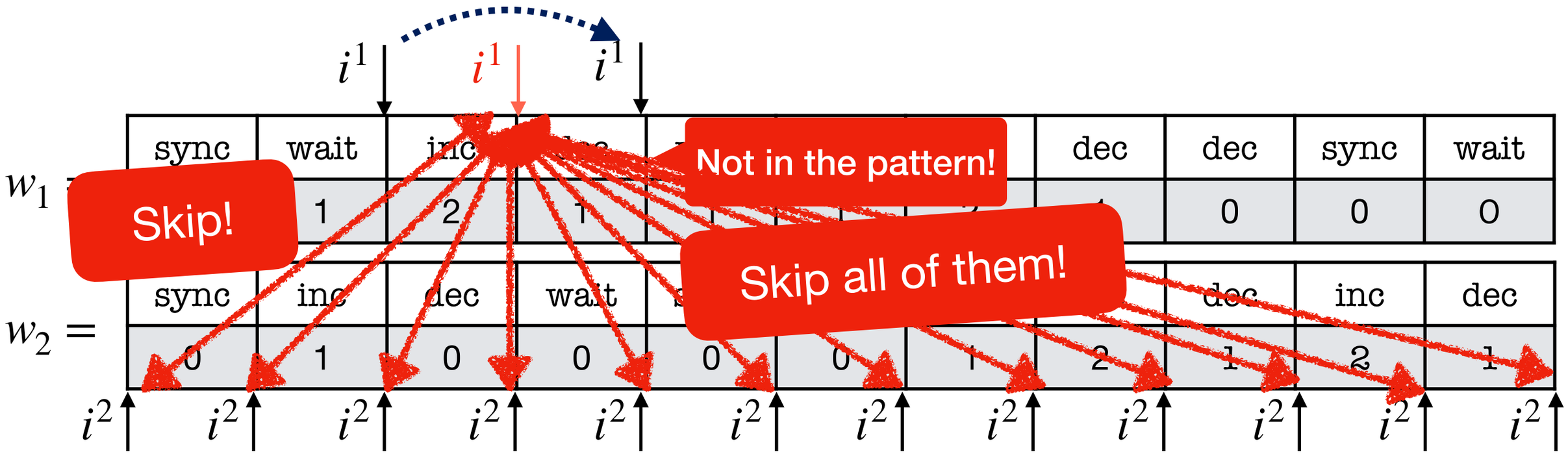
# Generalize skip over tracks

We “invalidate” useless starting pos. dimension-wise
→ The skipping effect also exponentially “blows up”



# Generalize skip over tracks

We “invalidate” useless starting pos. dimension-wise
→ The skipping effect also exponentially “blows up”



# Outline

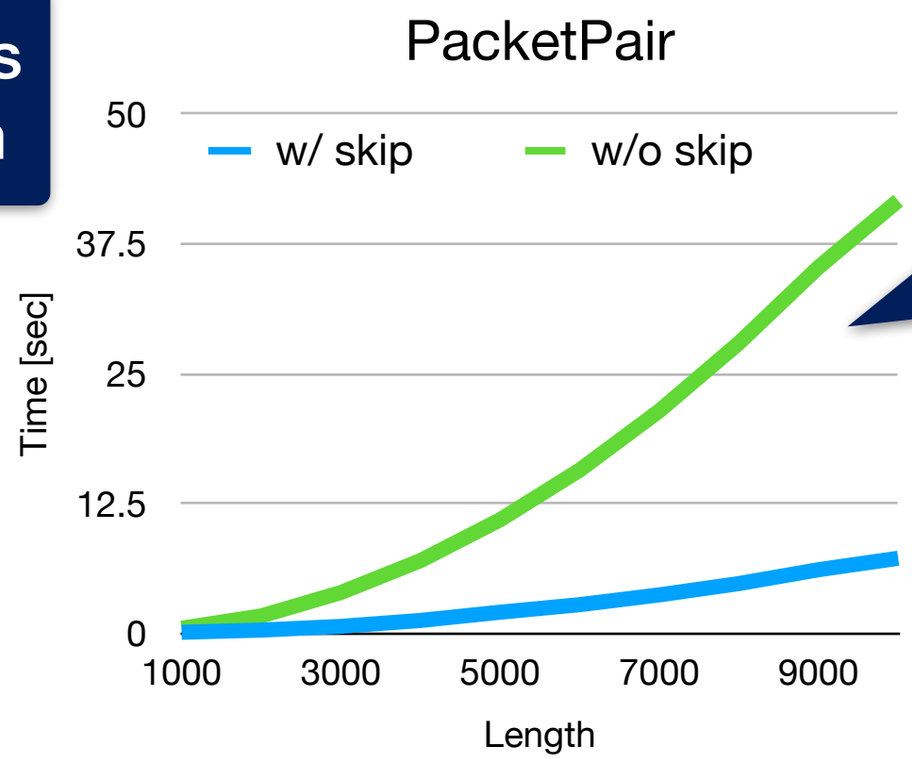
- The hyper pattern matching problem
- A naive algorithm
- Heuristics by skipping unnecessary matching trials
- Experiments

# Implementation/Experiments

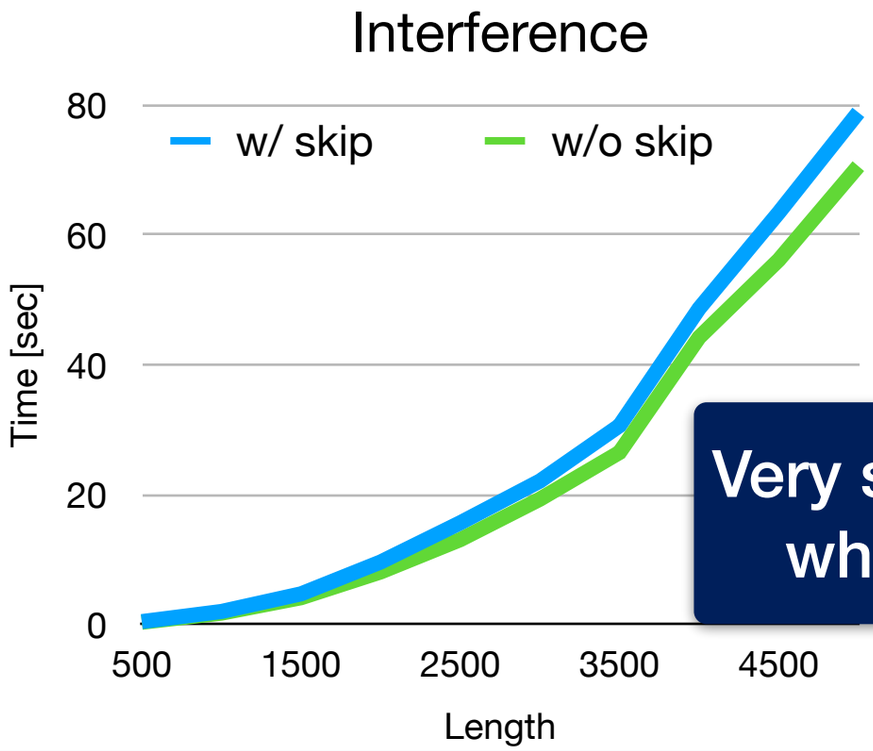
- Implemented the proposed algorithms in Rust
  - HypPAu: Published under GPLv3
- Evaluated the scalability wrt the length of words and the number of tracks
- Used our original benchmarks
- Intel Xeon w5-3435X 4.5 GHz and 63 GiB RAM w/ Ubuntu 24.04.2 LTS.

# Scalability wrt. words' length

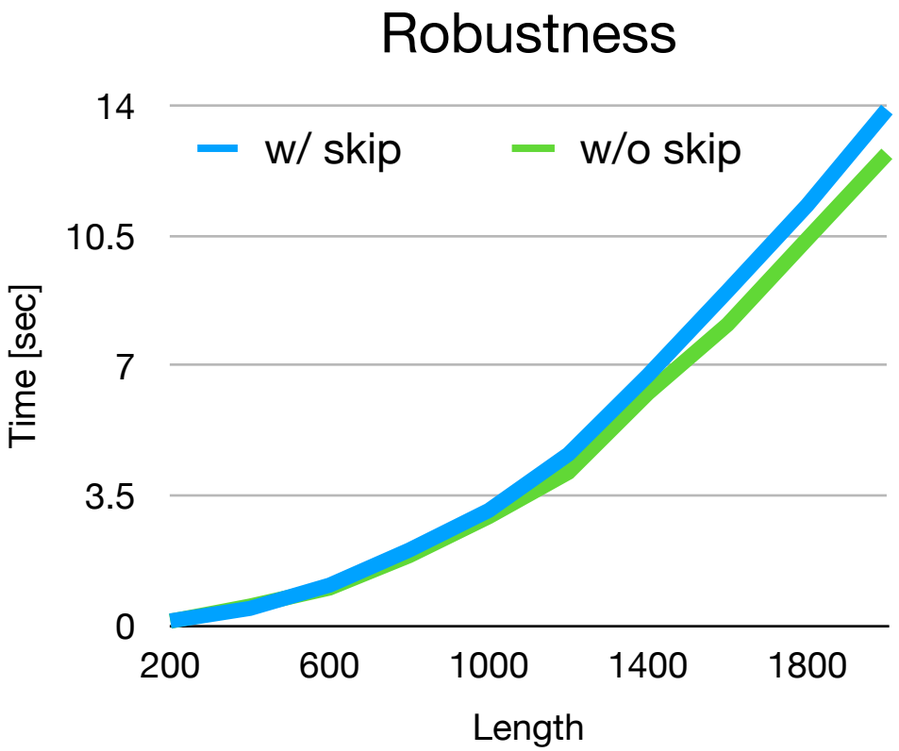
Benchmarks w/ 2 tracks
→ Quadratic wrt length




Much more efficient
when we can skip




Very small overhead even
when no skip occurs



# Scalability wrt. # of tracks

Word's length = 200

Works at most 4 tracks

| | 2 | 3 | 4 |
|---|---|---|---|
| **w/ skip** | 0.012 sec. | 0.949 sec. | 95.479 sec. |
| **w/o skip** | 0.026 sec. | 4.651 sec. | **Timeout (> 1800 sec.)** |

Speed up by skipping is more evident w/ many tracks

# Conclusions

Automata with multiple “tracks”

- Introduce hyper pattern matching with asynchronous automata (aka multi-tape automata)

Incrementally handles streams of words

- Propose incremental algorithms for hyper pattern matching
  - Naive algorithm + heuristics for better efficiency
- Works for reasonable size of data
  - e.g. words of length 1500 around 10 sec

# Future Directions

- Generalization with more general automata
  - Register automata
  - Timed automata
  - Something quantitative
- More efficient algorithms
  - Approximation of the match set
  - Indexing of the given word
- Extensions for NLP systems, e.g., via “soft” comparison of words w/ semantics similarity [Deguchi+, ICLR’25]

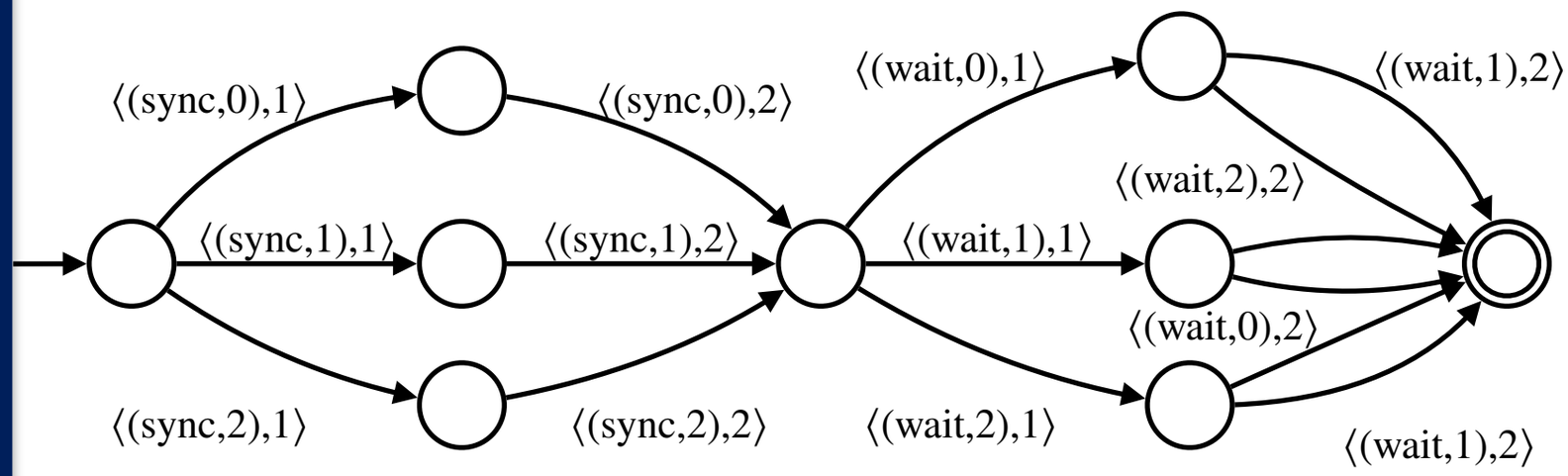
# Appendix

# Skipping based on lookahead

**Observation**

For each track,
- matching length = 2
- last letter is (wait, $n$)

→ matching starts from $i$ only if $(i + 1)$-th letter is (wait, $n$)




$i^1$

| $w_1 =$ | sync | wait | inc | dec | wait | sync | wait | dec | dec | sync | wait |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 1 | 1 | 1 | 2 | 1 | 0 | 0 | 0 |

| $w_2 =$ | sync | inc | dec | wait | sync | wait | inc | inc | dec | inc | dec |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 2 | 1 | 2 | 1 |

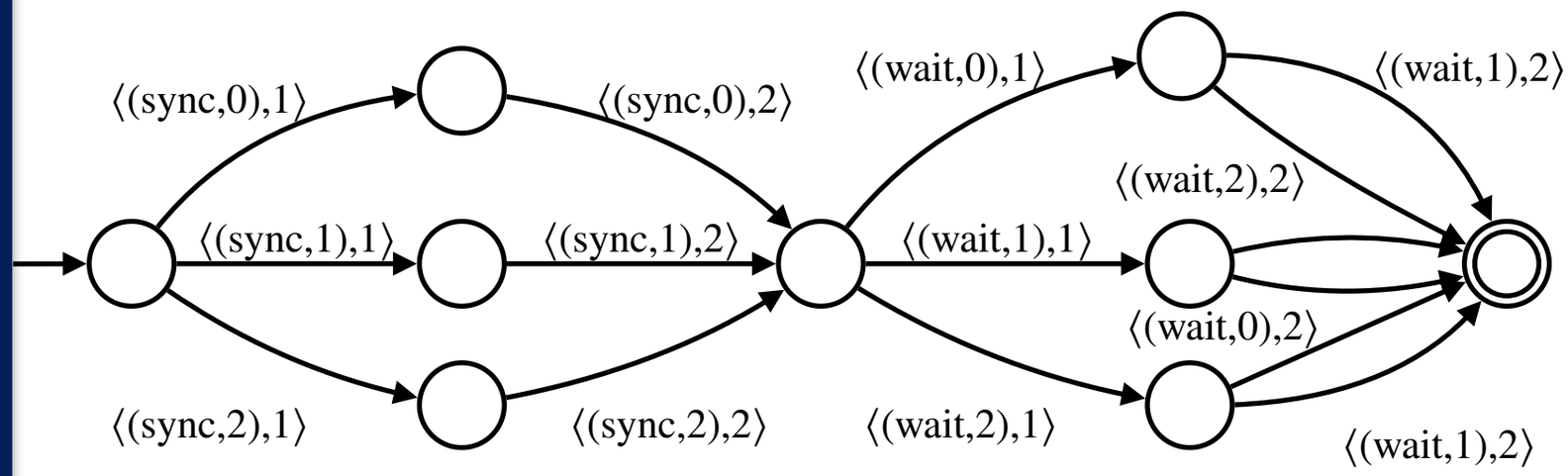
$i^2$

# Skipping based on lookahead

**Observation**

For each track,
- matching length = 2
- last letter is (wait, $n$)

→ matching starts from $i$ only if $(i + 1)$-th letter is (wait, $n$)




$i^1$ — No Matching!

$w_1 =$

| sync | wait | inc | dec | wait | sync | wait | dec | dec | sync | wait |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 1 | 1 | 1 | 2 | 1 | 0 | 0 | 0 |

$w_2 =$

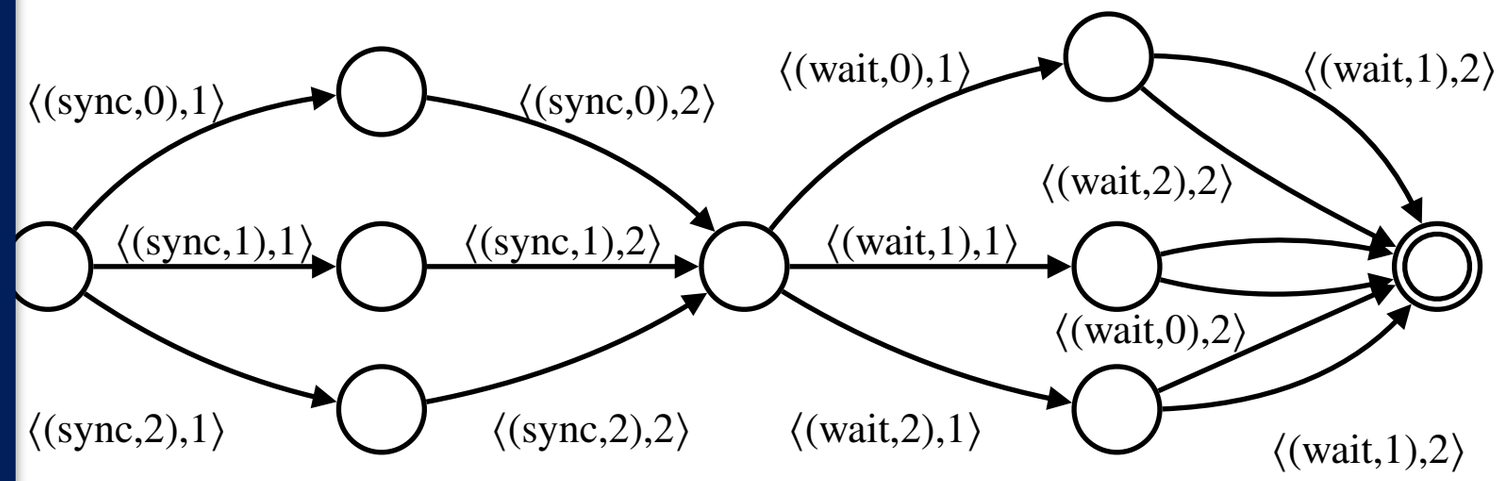
| sync | inc | dec | wait | sync | wait | inc | inc | dec | inc | dec |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 2 | 1 | 2 | 1 |

$i^2$

# Skipping based on lookahead

**Rule 1**

Look ahead $(i^k + 1)$-st letter
- start matching if it is (wait, $n$)
- otherwise look ahead $(i^k + 2)$-nd letter and shift $i^k$ by
  - 1 if it is (wait, $n$)
  - 2 if it is (sync, $n$)
  - 3 otherwise




$i^1$

| $w_1 =$ | sync | wait | inc | dec | wait | sync | wait | dec | dec | sync | wait |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 1 | 1 | 1 | 2 | 1 | 0 | 0 | 0 |

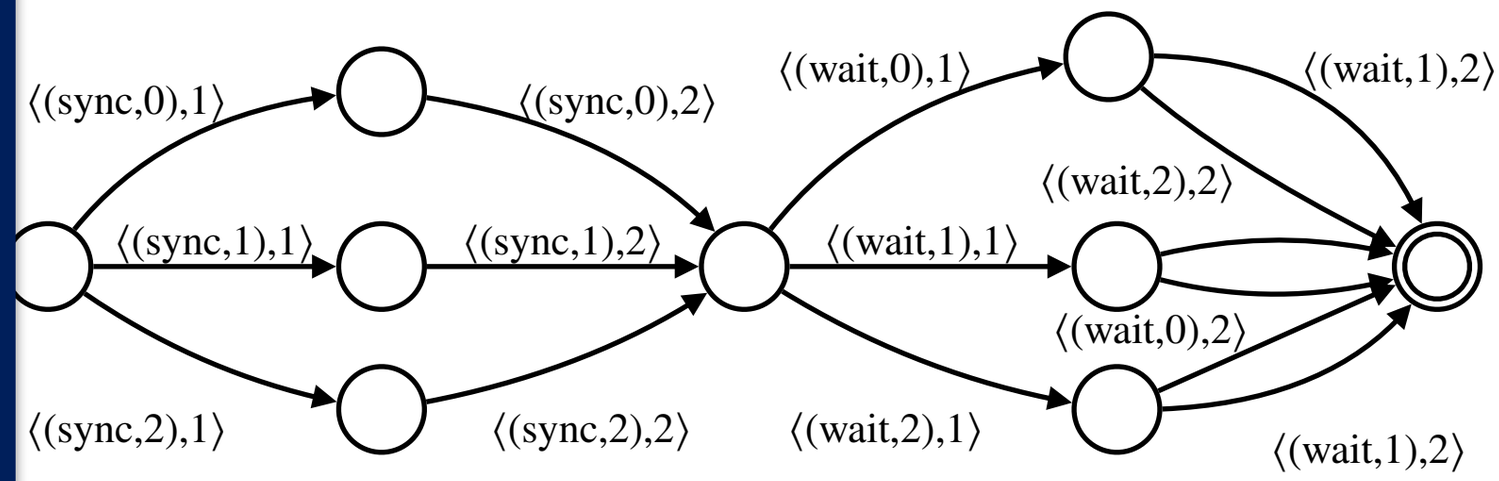
| $w_2 =$ | sync | inc | dec | wait | sync | wait | inc | inc | dec | inc | dec |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 2 | 1 | 2 | 1 |

$i^2$

# Skipping based on lookahead

**Rule 1**

Look ahead $(i^k + 1)$-st letter
- start matching if it is (wait, $n$)
- otherwise look ahead $(i^k + 2)$-nd letter and shift $i^k$ by
  - 1 if it is (wait, $n$)
  - 2 if it is (sync, $n$)
  - 3 otherwise




$i^1$

| $w_1 =$ | sync | wait | inc | dec | wait | sync | wait | dec | dec | sync | wait |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 1 | 1 | 1 | 2 | 1 | 0 | 0 | 0 |

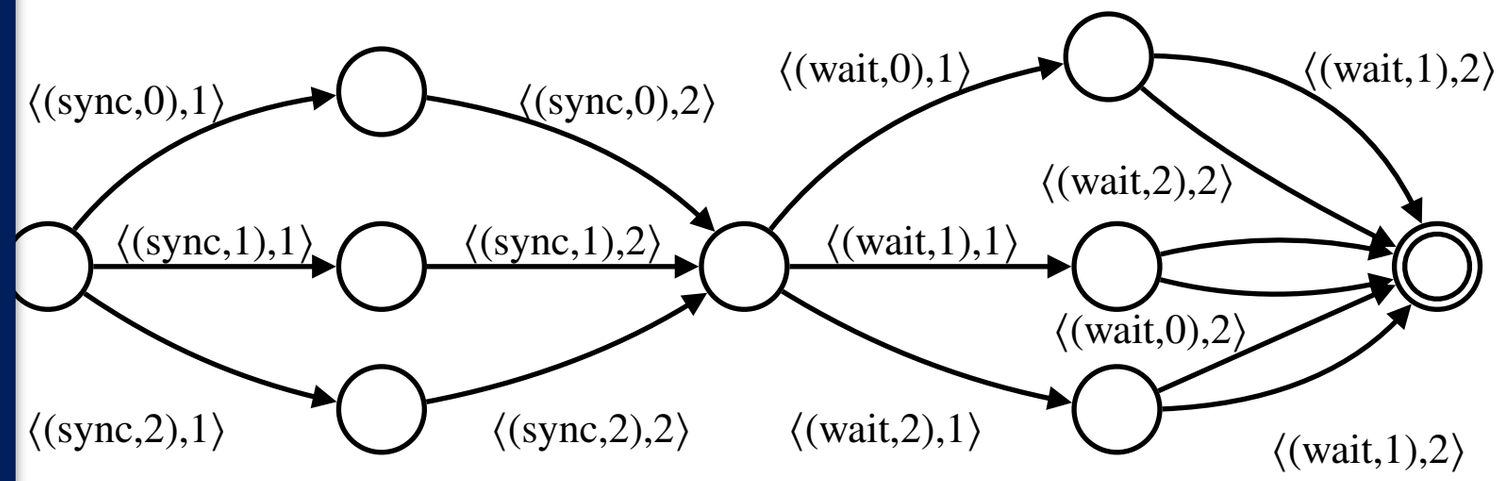
| $w_2 =$ | sync | inc | dec | wait | sync | wait | inc | inc | dec | inc | dec |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 2 | 1 | 2 | 1 |

$i^2$

# Skipping based on lookahead

**Rule 1**

Look ahead $(i^k + 1)$-st letter
- start matching if it is (wait, $n$)
- otherwise look ahead $(i^k + 2)$-nd letter and shift $i^k$ by
  - 1 if it is (wait, $n$)
  - 2 if it is (sync, $n$)
  - 3 otherwise




$i^1$

| $w_1 =$ | sync | wait | inc | dec | wait | sync | wait | dec | dec | sync | wait |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 1 | 1 | 1 | 2 | 1 | 0 | 0 | 0 |

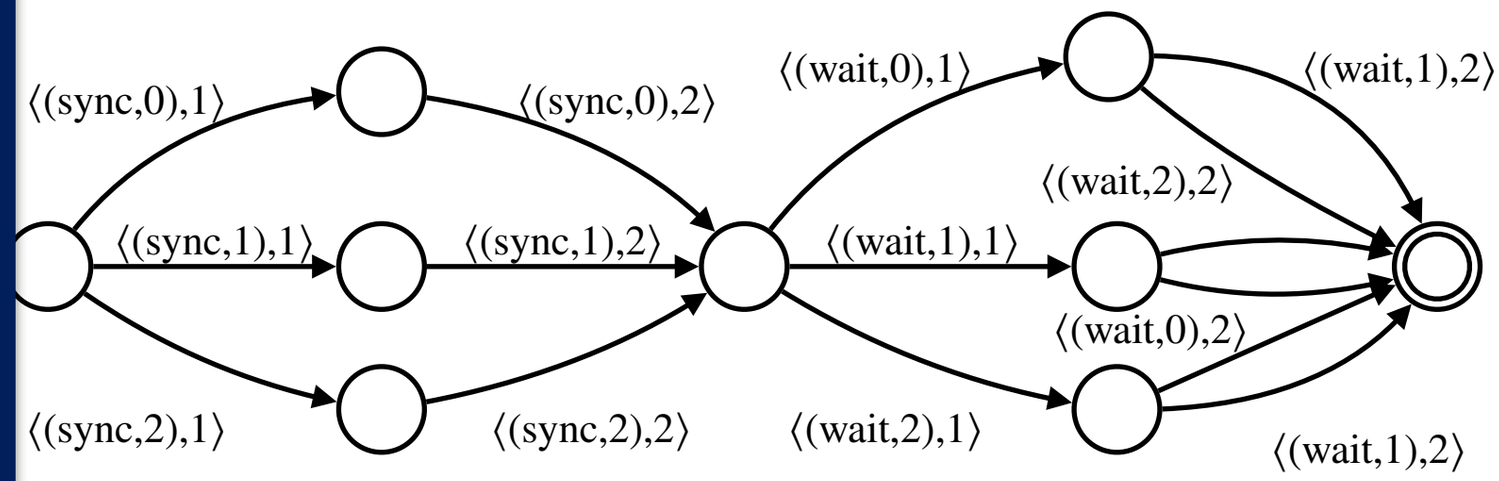
| $w_2 =$ | sync | inc | dec | wait | sync | wait | inc | inc | dec | inc | dec |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 2 | 1 | 2 | 1 |

$i^2$

# Skipping based on lookahead

**Rule 1**

Look ahead $(i^k + 1)$-st letter
- start matching if it is (wait, $n$)
- otherwise look ahead $(i^k + 2)$-nd letter and shift $i^k$ by
  - 1 if it is (wait, $n$)
  - 2 if it is (sync, $n$)
  - 3 otherwise




$i^1$

$w_1 =$

| sync | wait | inc | dec | wait | sync | wait | dec | dec | sync | wait |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 1 | 1 | 1 | 2 | 1 | 0 | 0 | 0 |

$w_2 =$

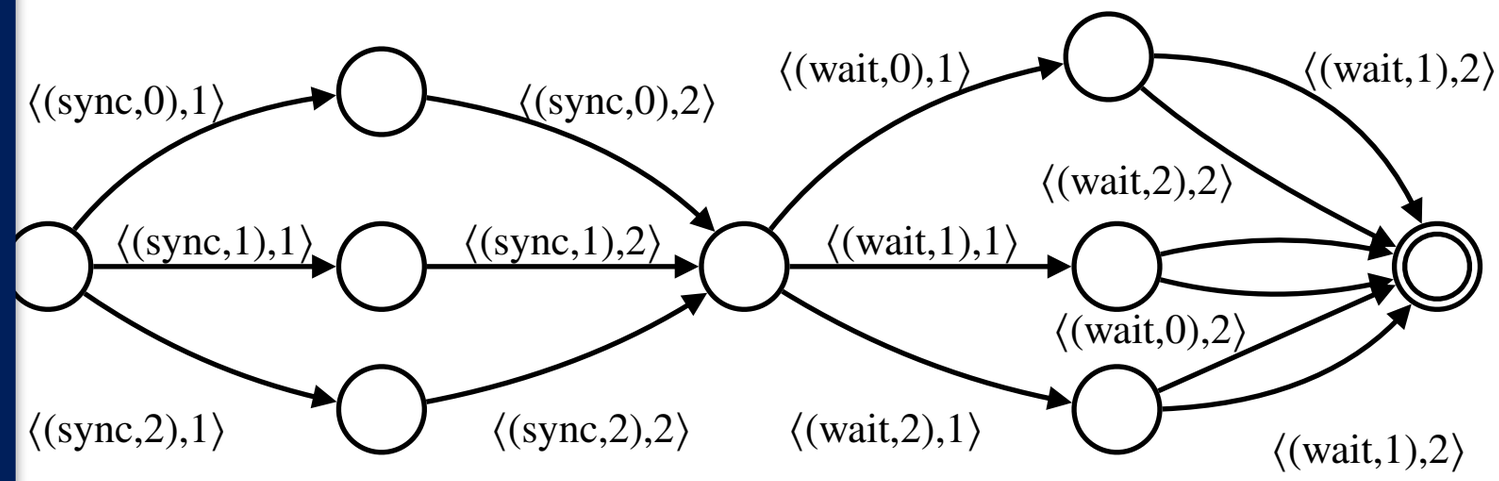
| sync | inc | dec | wait | sync | wait | inc | inc | dec | inc | dec |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 2 | 1 | 2 | 1 |

$i^2$

# Skipping based on lookahead

**Rule 1**

Look ahead $(i^k + 1)$-st letter
- start matching if it is (wait, $n$)
- otherwise look ahead $(i^k + 2)$-nd letter and shift $i^k$ by
  - 1 if it is (wait, $n$)
  - 2 if it is (sync, $n$)
  - 3 otherwise




Start Matching

$i^1$

$w_1 =$

| sync | wait | inc | dec | wait | sync | wait | dec | dec | sync | wait |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 1 | 1 | 1 | 2 | 1 | 0 | 0 | 0 |

$w_2 =$

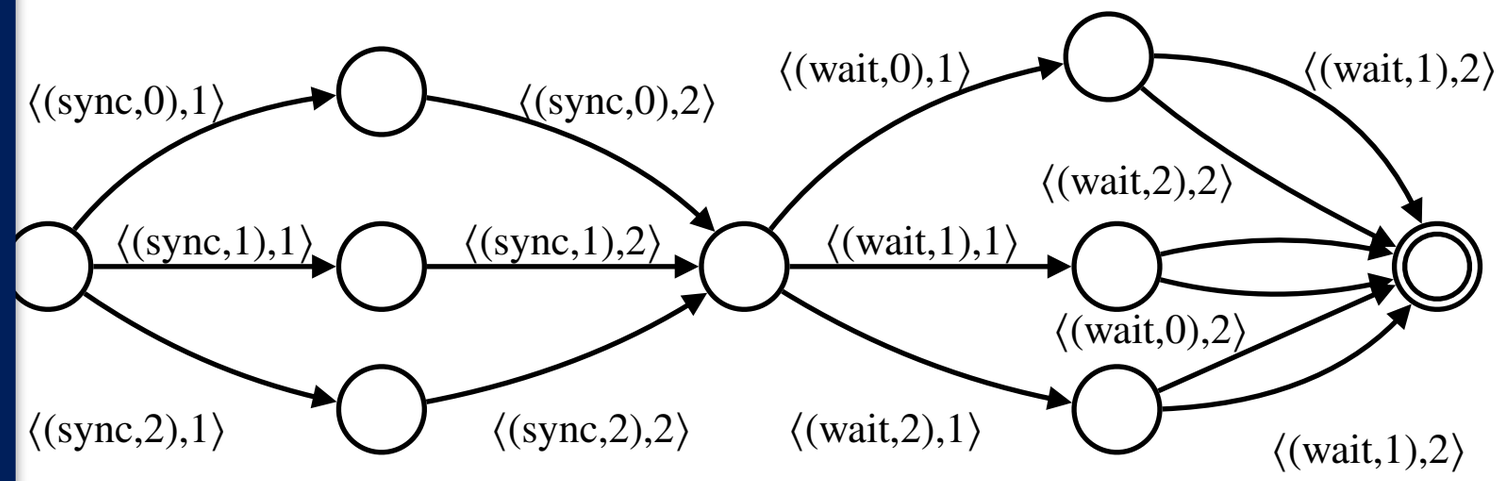
| sync | inc | dec | wait | sync | wait | inc | inc | dec | inc | dec |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 2 | 1 | 2 | 1 |

$i^2$

# Skipping based on lookahead

**Rule 1**

Look ahead $(i^k + 1)$-st letter
- start matching if it is (wait, $n$)
- otherwise look ahead $(i^k + 2)$-nd letter and shift $i^k$ by
  - 1 if it is (wait, $n$)
  - 2 if it is (sync, $n$)
  - 3 otherwise




Start Matching

$i^1$

| $w_1 =$ | sync | wait | inc | dec | wait | sync | wait | dec | dec | sync | wait |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 1 | 1 | 1 | 2 | 1 | 0 | 0 | 0 |

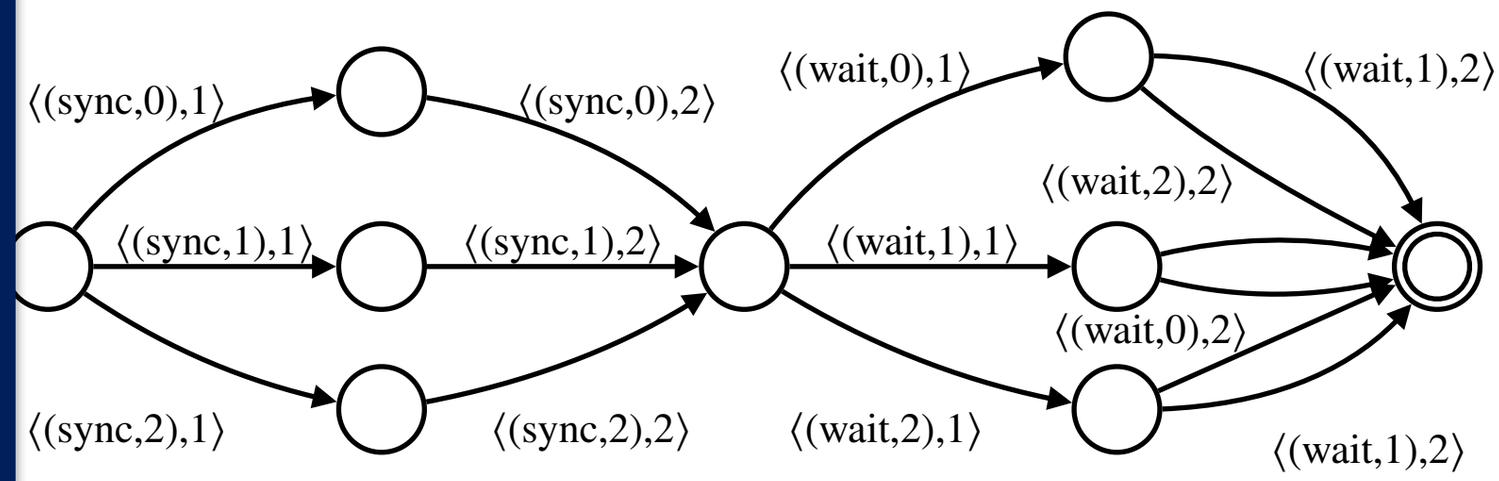
| $w_2 =$ | sync | inc | dec | wait | sync | wait | inc | inc | dec | inc | dec |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 2 | 1 | 2 | 1 |

$i^2$

# Skipping based on lookahead

**Rule 1**

Look ahead $(i^k + 1)$-st letter
- start matching if it is (wait, $n$)
- otherwise look ahead $(i^k + 2)$-nd letter and shift $i^k$ by
  - 1 if it is (wait, $n$)
  - 2 if it is (sync, $n$)
  - 3 otherwise




Start Matching

$i^1$

| $w_1 =$ | sync | wait | inc | dec | wait | sync | wait | dec | dec | sync | wait |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 2 | 1 | 1 | 1 | 2 | 1 | 0 | 0 | 0 |

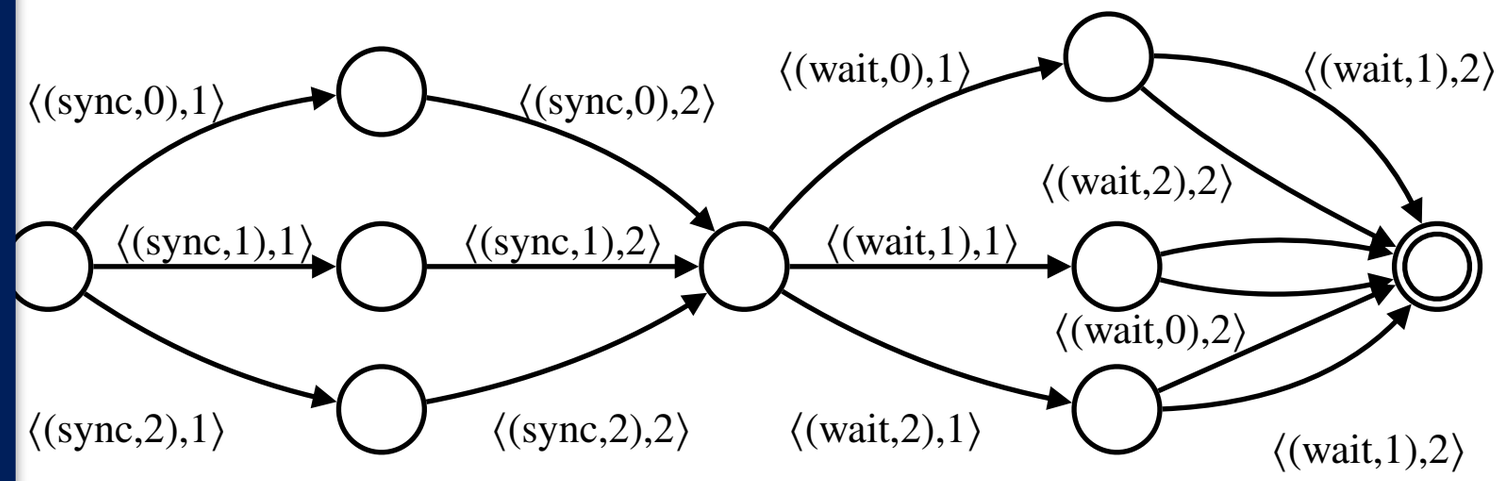
| $w_2 =$ | sync | inc | dec | wait | sync | wait | inc | inc | dec | inc | dec |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | 0 | 1 | 0 | 0 | 0 | 0 | 1 | 2 | 1 | 2 | 1 |

$i^2$

# Skipping based on lookahead

**Rule 1**

Look ahead $(i^k + 1)$-st letter
- start matching if it is (wait, $n$)
- otherwise look ahead $(i^k + 2)$-nd letter and shift $i^k$ by
  - 1 if it is (wait, $n$)
  - 2 if it is (sync, $n$)
  - 3 otherwise




Start Matching

$i^1$

$w_1 =$

| sync | wait | inc | dec | wait | sync | wait | dec | dec | sync | wait |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 1 | 1 | 1 | 2 | 1 | 0 | 0 | 0 |

$w_2 =$

| sync | inc | dec | wait | sync | wait | inc | inc | dec | inc | dec |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 2 | 1 | 2 | 1 |

$i^2$

Start Matching

# Quick Search-Style Skipping

[Contribution]

## Rule 1 (Generalized)

For the shortest matching length $l^k$ for the $k$-th track,
look ahead $(i^k + l^k - 1)$-th letter

- start matching if it can be the $l^k$-th letter of $w \in L(\mathscr{A})$ projected to the $k$-th track
- otherwise look ahead $(i^k + l^k)$-th letter and shift $i^k$ by the minimum $j$ s.t. it can be the $(l^k + 1 - j)$-th letter of $w \in L(\mathscr{A})$ projected to $k$-th track

# Quick Search-Style Skipping

[Contribution]

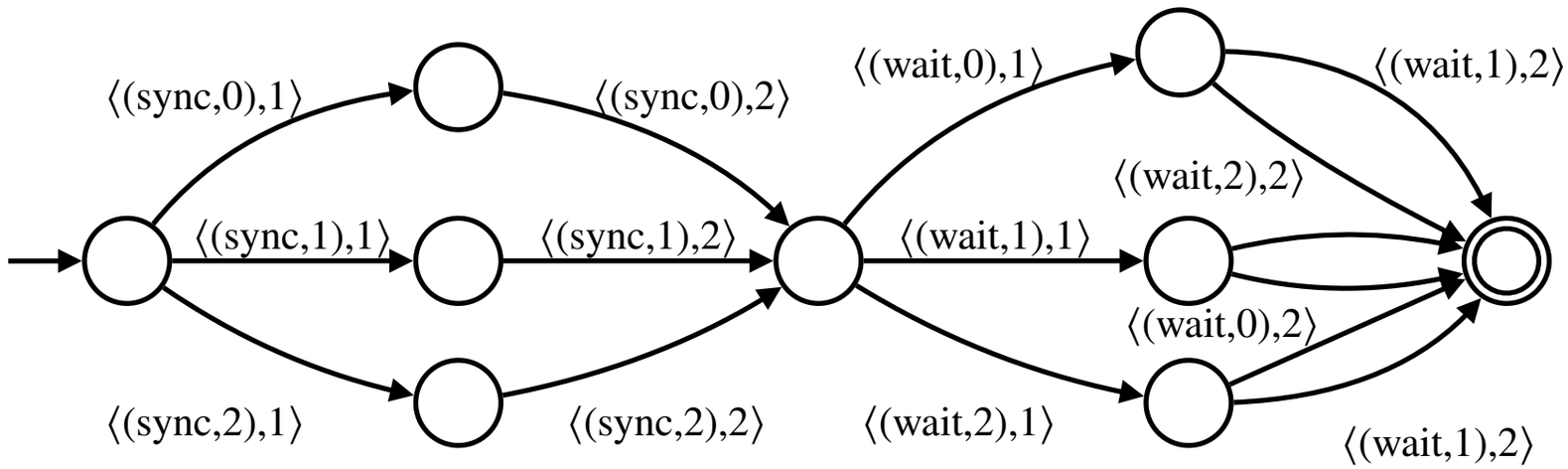
## Rule 1 (Generalized)

For the shortest matching length $l^k$ for the $k$-th track,
look ahead $(i^k + l^k - 1)$-th letter

- start matching if it can be the $l^k$-th letter of $w \in L(\mathscr{A})$ projected to the $k$-th track
- otherwise look ahead $(i^k + l^k)$-th letter and shift $i^k$ by the minimum $j$ s.t. it can be the $(l^k + 1 - j)$-th letter of $w \in L(\mathscr{A})$ projected to $k$-th track

**Independent of the monitored words → Precompute beforehand**

# Skip matching trials based on reached states in the latest trial




$i^1$

$w_1 =$

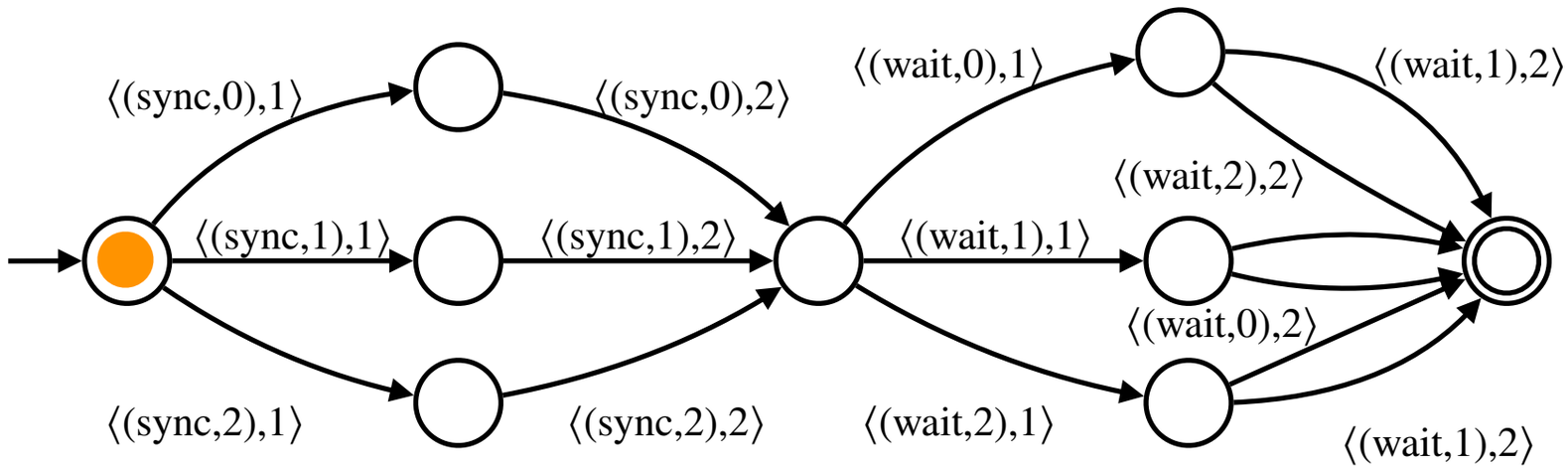
| sync | wait | inc | dec | wait | sync | wait | dec | dec | sync | wait |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 1 | 1 | 1 | 2 | 1 | 0 | 0 | 0 |

$w_2 =$

| sync | inc | dec | wait | sync | wait | inc | inc | dec | inc | dec |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 2 | 1 | 2 | 1 |

$i^2$

# Skip matching trials based on reached states in the latest trial




$i^1$

$w_1 =$

| sync | wait | inc | dec | wait | sync | wait | dec | dec | sync | wait |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 1 | 1 | 1 | 2 | 1 | 0 | 0 | 0 |

$w_2 =$

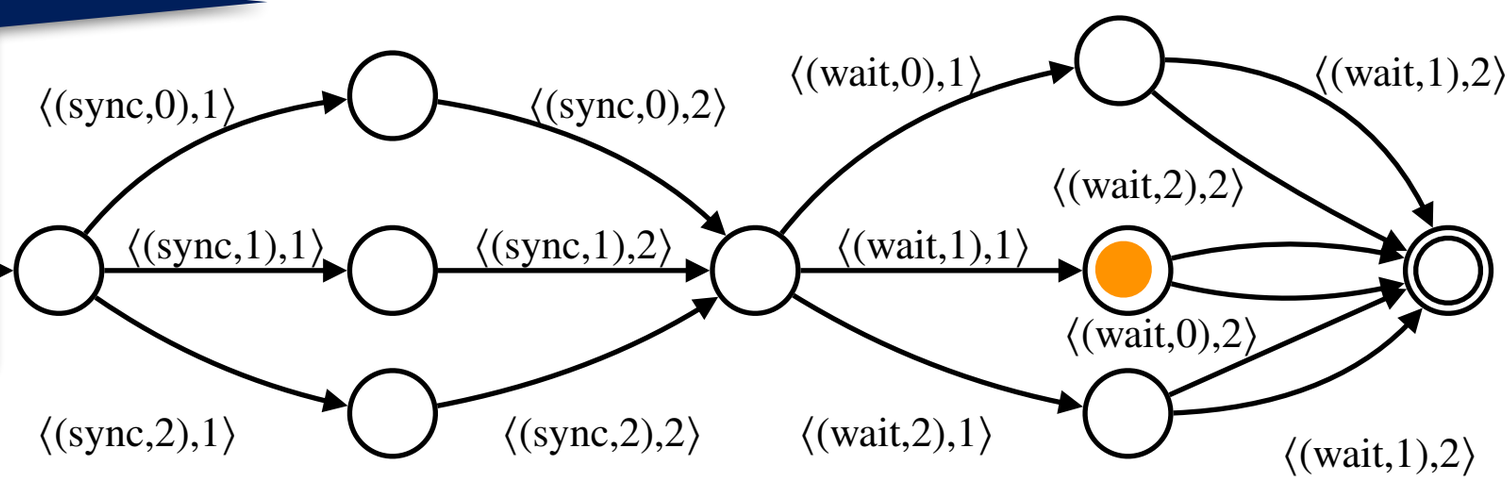
| sync | inc | dec | wait | sync | wait | inc | inc | dec | inc | dec |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 2 | 1 | 2 | 1 |

$i^2$

# Skip matching trials based on reached states in the latest trial

**Observation 2**

On track 1, the suffix from $i^1$ is
(sync,0)(wait,1)⋯
(sync,1)(wait,1)⋯
(sync,2)(wait,1)⋯




$i^1$

$w_1 =$

| sync | wait | inc | dec | wait | sync | wait | dec | dec | sync | wait |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 1 | 1 | 1 | 2 | 1 | 0 | 0 | 0 |

$w_2 =$

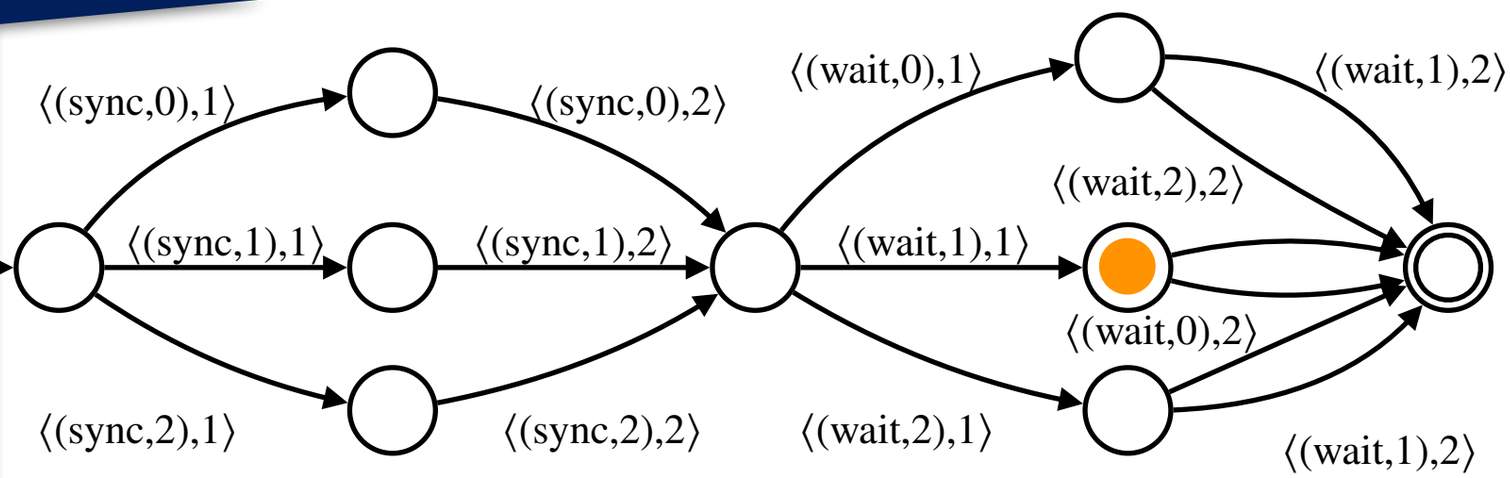
| sync | inc | dec | wait | sync | wait | inc | inc | dec | inc | dec |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 0 | 0 | 0 | 0 | 1 | 2 | 1 | 2 | 1 |

$i^2$

# Skip matching trials based on reached states in the latest trial

## Observation 2

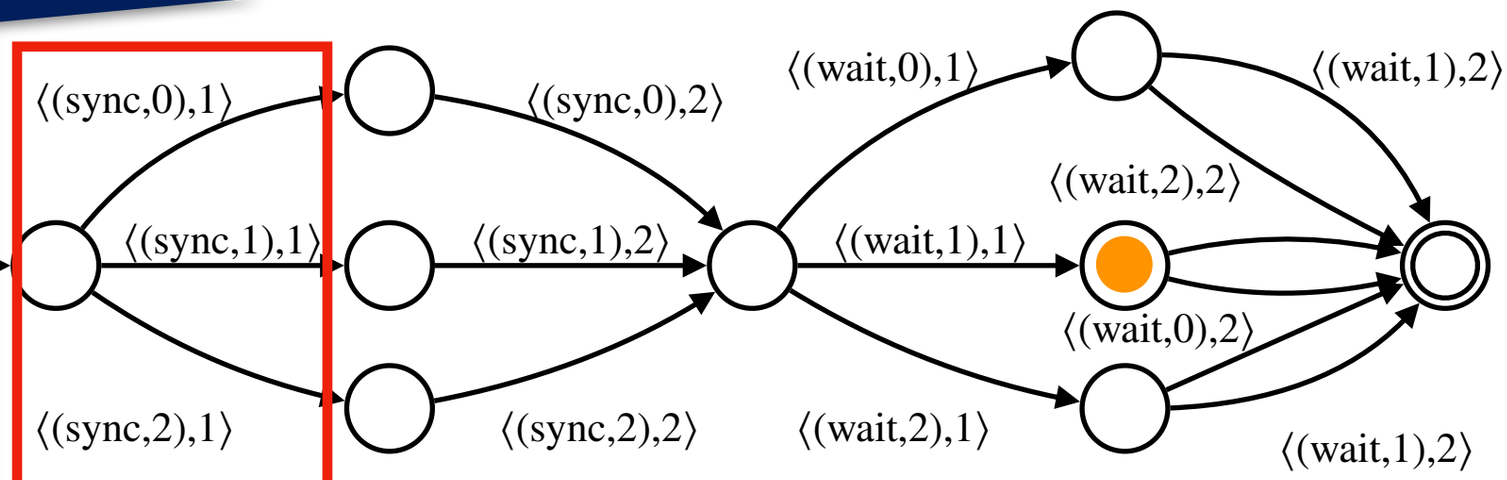
On track 1, the suffix from $i^1$ is
$(\mathrm{sync},0)(\mathrm{wait},1)\cdots$
$(\mathrm{sync},1)(\mathrm{wait},1)\cdots$
$(\mathrm{sync},2)(\mathrm{wait},1)\cdots$
→ the suffix from $i^1 + 1$ is
$(\mathrm{wait},1)\cdots$
→ no matching from $i^1 + 1$



# Skip matching trials based on reached states in the latest trial

## Observation 2

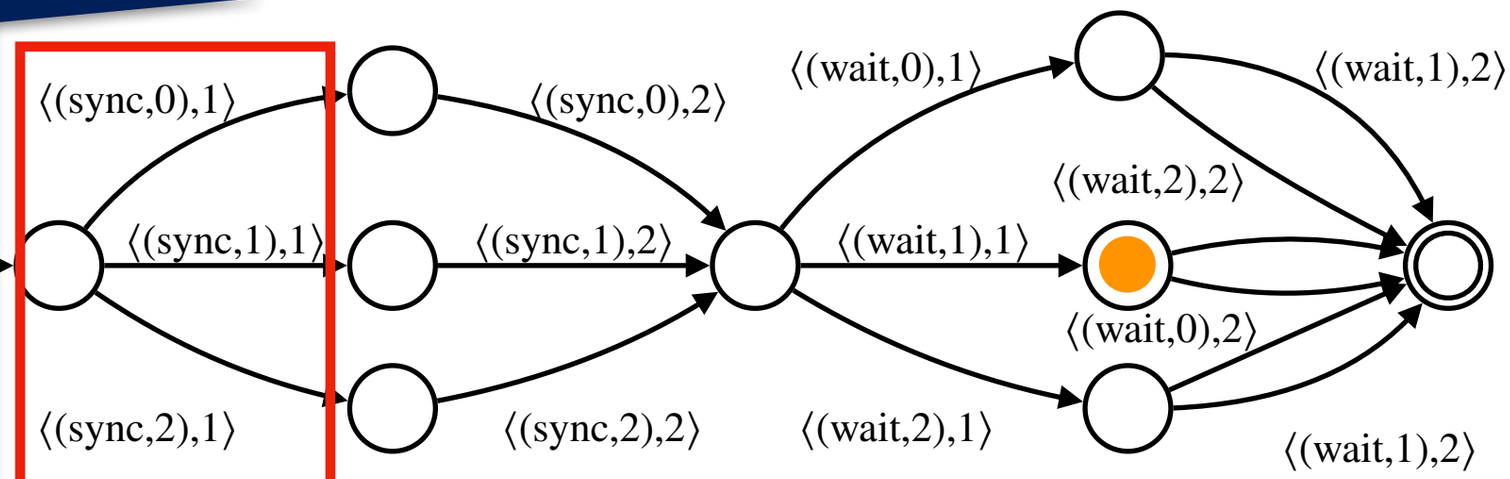
On track 1, the suffix from $i^1$ is
$(\mathrm{sync},0)(\mathrm{wait},1)\cdots$
$(\mathrm{sync},1)(\mathrm{wait},1)\cdots$
$(\mathrm{sync},2)(\mathrm{wait},1)\cdots$
→ the suffix from $i^1 + 1$ is
$(\mathrm{wait},1)\cdots$
→ no matching from $i^1 + 1$



# Skip matching trials based on reached states in the latest trial

**Observation 2**

On track 1, the suffix from $i^1$ is
(sync,0)(wait,1)⋯
(sync,1)(wait,1)⋯
(sync,2)(wait,1)⋯
→ the suffix from $i^1$ + 1 is
(wait,1)⋯
→ no matching from $i^1$ + 1




$i^1$

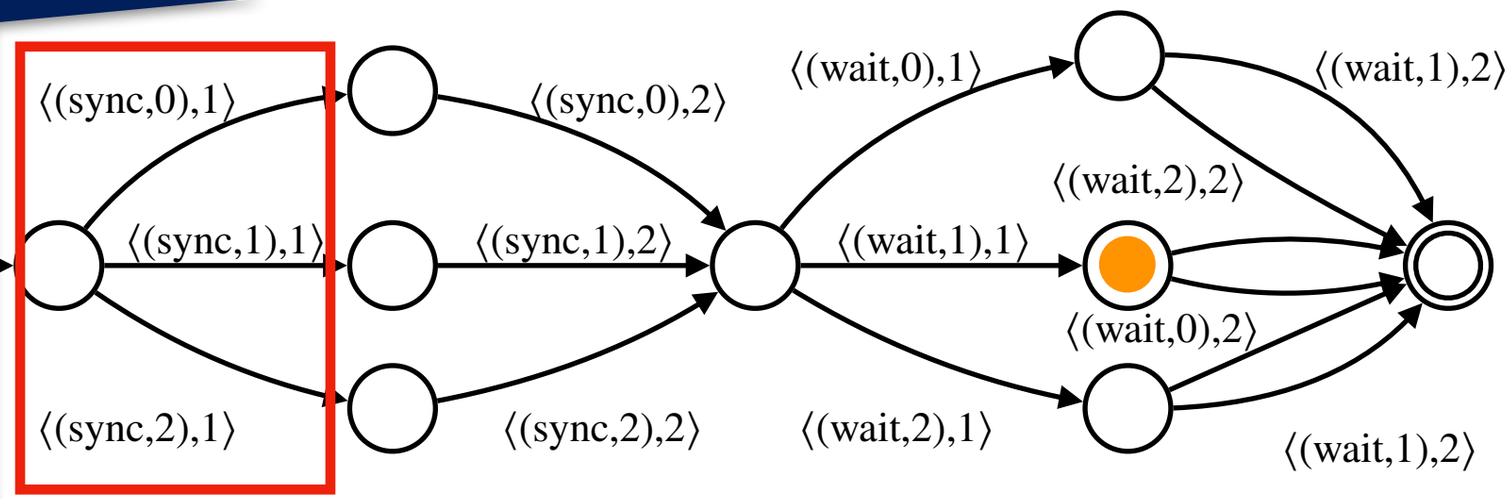
$w_1 =$

| sync | wait | inc | dec | wait | sync | wait | dec | dec | sync | wait |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 1 | 1 | 1 | 2 | 1 | 0 | 0 | 0 |

# Skip matching trials based on reached states in the latest trial

**Observation 2**

On track 1, the suffix from $i^1$ is
(sync,0)(wait,1)⋯
(sync,1)(wait,1)⋯
(sync,2)(wait,1)⋯
→ the suffix from $i^1$ + 1 is
(wait,1)⋯
→ no matching from $i^1$ + 1




$i^1$ ✗

$w_1 =$

| sync | wait | inc | dec | wait | sync | wait | dec | dec | sync | wait |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1 | 2 | 1 | 1 | 1 | 2 | 1 | 0 | 0 | 0 |

# KMP-Style Skipping

[Contribution]

Independent of the monitored words → Precompute beforehand

**Rule 2**

If we reach state $s$ in the latest matching trial, we can move the $k$-th track by the minimum $n > 0$ s.t.

$$\pi(L(\mathscr{A}_s), k) \cdot \Sigma^* \cap \Sigma^n \cdot \pi(L(\mathscr{A}), k) \neq \emptyset$$

Overapproximation of the suffixes from $i^k$

Pattern after shifted by $n$ (projected to $k$)

**Notations**

- $\Sigma$: Alphabet
- $\mathscr{A}_s$: $\mathscr{A}$ with accepting state $s$
- $\pi(L, k)$: Projection of $L$ to the $k$-th track

# FJS-style Algorithm for Hyper Pattern Matching

1. Pick the starting positions

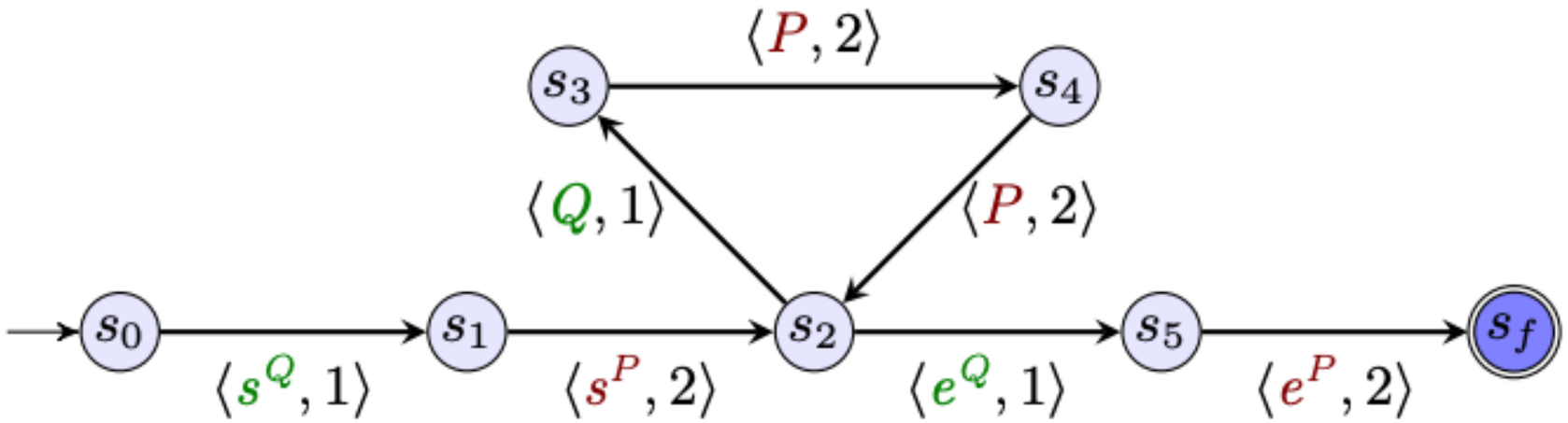
2. Apply Rule 1 to skip positions with lookahead

3. Feed letters to the asynchronous automaton $\mathscr{A}$

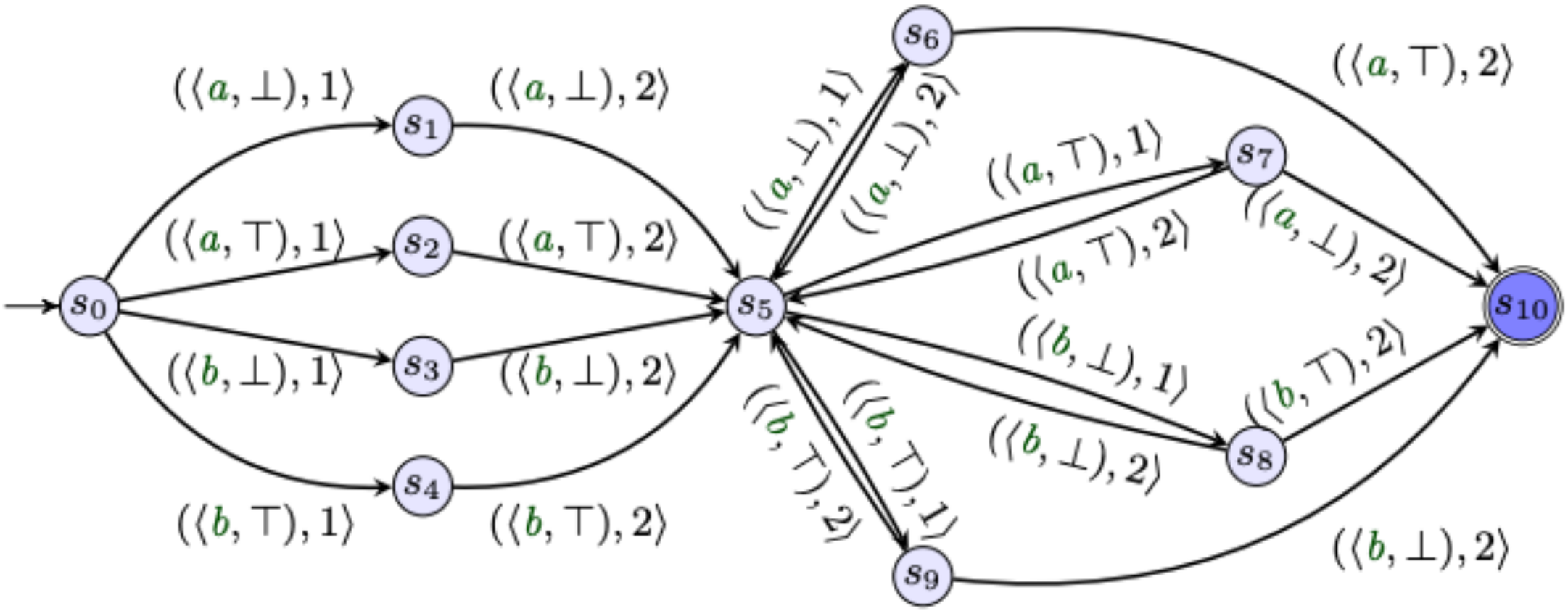
   - Report matching if we reach an accepting state

4. Apply Rule 2 to move the starting position after the matching trial
   → Go back to Step 2

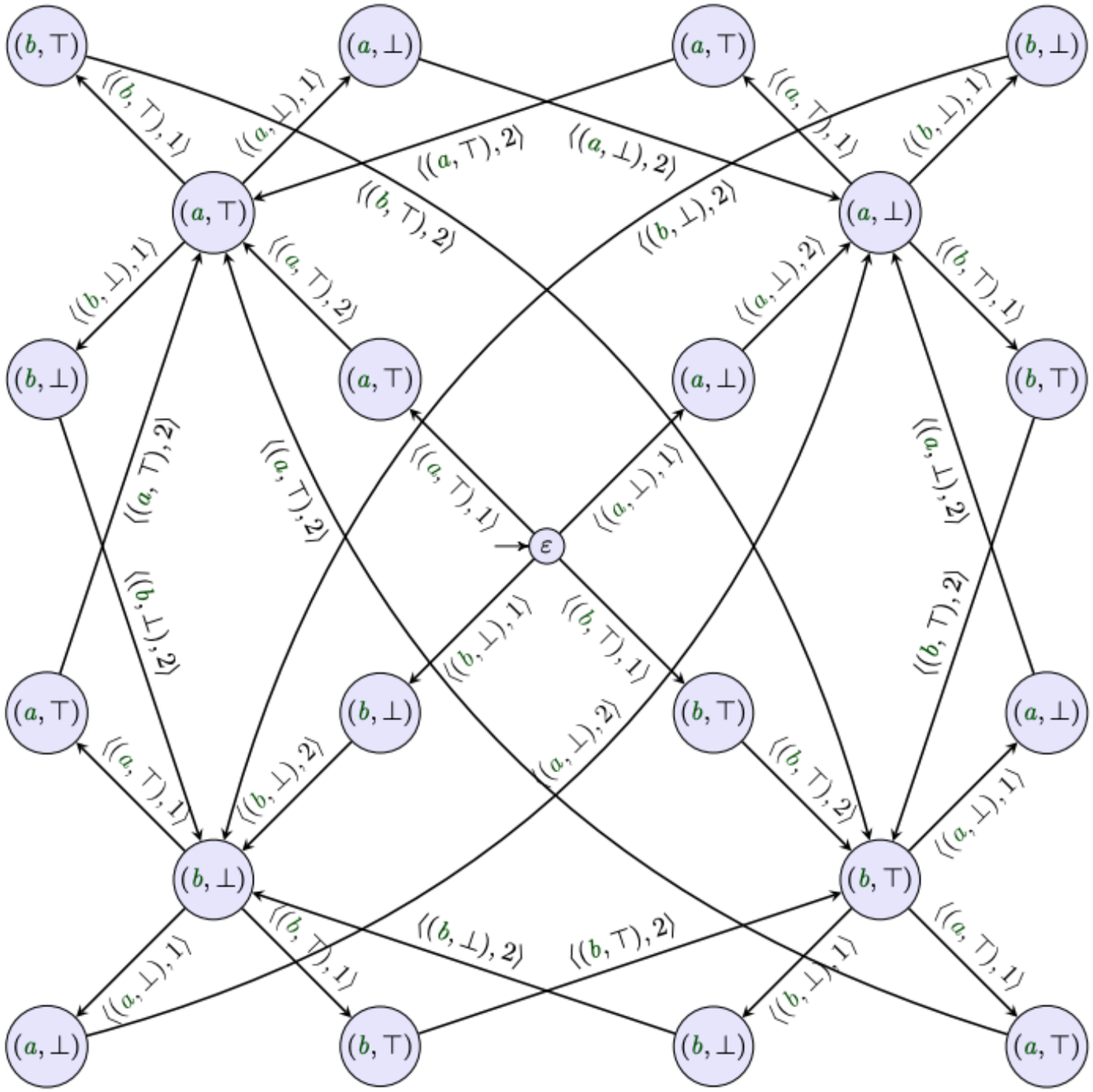
# PacketPair


$\langle P, 2\rangle$
$s_3$
$s_4$
$\langle Q, 1\rangle$
$\langle P, 2\rangle$
$s_0$
$s_1$
$s_2$
$s_5$
$s_f$
$\langle s^Q, 1\rangle$
$\langle s^P, 2\rangle$
$\langle e^Q, 1\rangle$
$\langle e^P, 2\rangle$

# Interference



# Robustness


(b, ⊤)
(a, ⊥)
(a, ⊤)
(b, ⊥)
(a, ⊤)
(a, ⊥)
(b, ⊥)
(a, ⊤)
(b, ⊥)
(a, ⊤)
(b, ⊤)
(a, ⊥)
(b, ⊤)
(a, ⊥)
(b, ⊥)
(a, ⊤)
(b, ⊤)
(b, ⊥)
(a, ⊥)
(b, ⊤)
ε
⟨(b, ⊤), 1⟩
⟨(a, ⊥), 1⟩
⟨(a, ⊤), 2⟩
⟨(a, ⊥), 2⟩
⟨(b, ⊤), 2⟩
⟨(b, ⊥), 2⟩
⟨(a, ⊤), 1⟩
⟨(b, ⊥), 1⟩
⟨(b, ⊥), 1⟩
⟨(a, ⊤), 2⟩
⟨(a, ⊥), 2⟩
⟨(b, ⊤), 1⟩
⟨(a, ⊤), 2⟩
⟨(b, ⊤), 2⟩
⟨(a, ⊤), 1⟩
⟨(b, ⊥), 2⟩
⟨(a, ⊥), 1⟩
⟨(b, ⊤), 1⟩
⟨(b, ⊥), 2⟩
⟨(b, ⊤), 2⟩
⟨(a, ⊥), 1⟩
⟨(b, ⊥), 1⟩
⟨(a, ⊤), 1⟩
⟨(a, ⊥), 1⟩
⟨(b, ⊤), 2⟩
⟨(a, ⊤), 1⟩
⟨(b, ⊤), 1⟩
⟨(a, ⊥), 2⟩
⟨(b, ⊥), 2⟩
⟨(b, ⊤), 2⟩
⟨(a, ⊥), 1⟩
⟨(a, ⊤), 1⟩