https://github.com/MasWag/monaa

MONAA: a Tool for Timed Pattern Matching with Automata-Based Acceleration

Masaki Waga[1], Ichiro Hasuo[1], and Kohei Suenaga[2]

National Institute of Informatics[1] and Kyoto University[2]

10 Apr. 2018, MT-CPS 2018

The authors are supported by ERATO HASUO Metamathematics for Systems Design Project (No. JPMJER1603), JST, Grants-in-Aid No. 15KT0012, JSPS, and JST PRESTO Grant Number JPMJPR15E5, Japan.

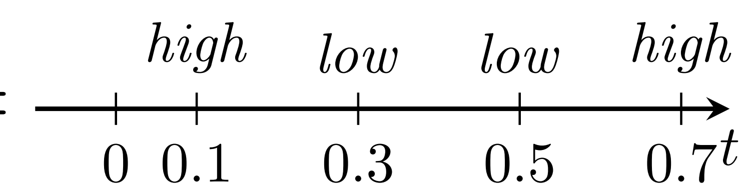
Timed Pattern Matching

[Ulus et al., FORMATS'14]

Input

- **Time-series data** (Logs of a car/ a robot)

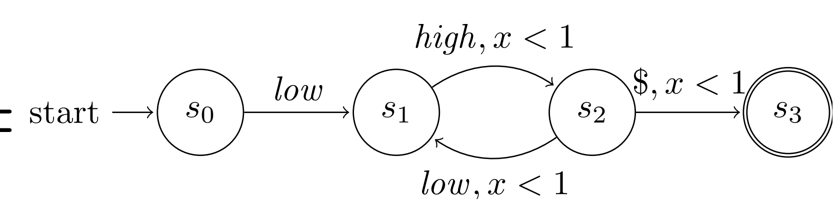
Timed Words
(Sequence of events with timestamps)

$w =$ 

- **Real-time spec.** (Spec. useful for debugging)

  e.g., The ge

Timed Automata
[Alur & Dill, TCS'94]

$\mathcal{A} =$ 

Output

- The intervals where the spec. is satisfied in the data

  e.g., The gea $\mathcal{M}(w, \mathcal{A}) = \{(t, t') \mid w|_{(t,t')} \in L(\mathcal{A})\}$ s

M. Waga (NII)

MONAA Overview

Command Line Interface (MONAA)

- Command line tool for timed pattern matching
- We can inspect a log
- The log is read lazily
  - online monitor
- **Text-based I/O**

C++ API (libmonaa)

- Execute timed pattern matching in a user's code
- Accelerated by **Skipping**
  [Waga et al., FORMATS'17]
- I/O by function/class

Outline

1. Algorithm in MONAA
    - Skipping for timed pattern matching
      [Waga et al., FORMATS'17]
2. Frontend of MONAA
    - Command line interface (CLI) / C++ API
3. Experiments

Skipping for String Matching

Brute-Force Search

Find “STRING”(pattern)
from “STRONG_STRING”(target)

```
STRONG_STRING
-------------
STRING
 STRING
   ⋮
      STRING
```

Match until the **3rd** char.

KMP Search

[Knuth+, SIAM J. Comput. ’77]

```
STRONG_STRING
-------------
STRING
 STRING
  STRING
   STRING
     ⋮
      STRING
```

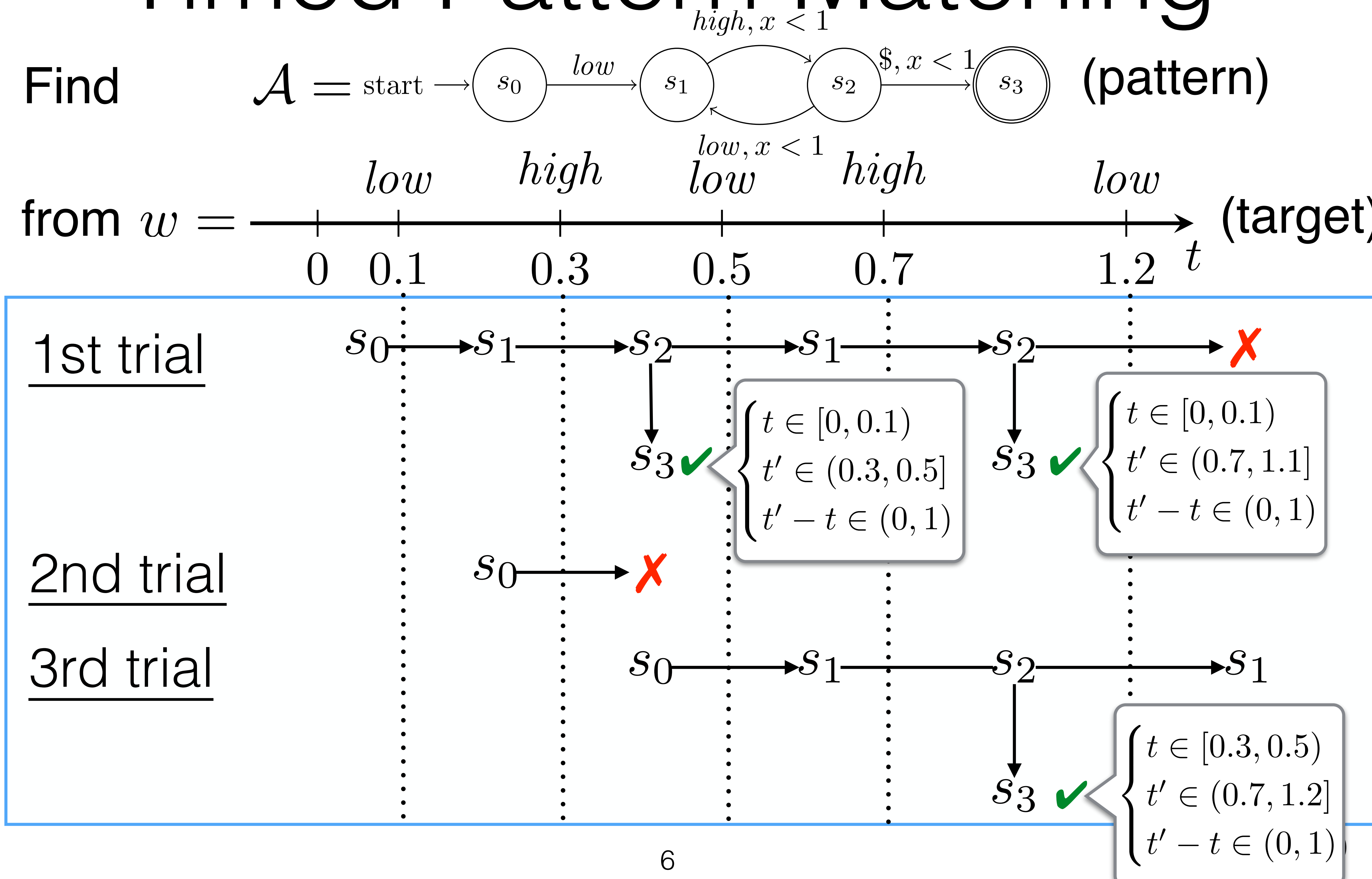
Skip 2 trials!!

Table for **length 3** partial match

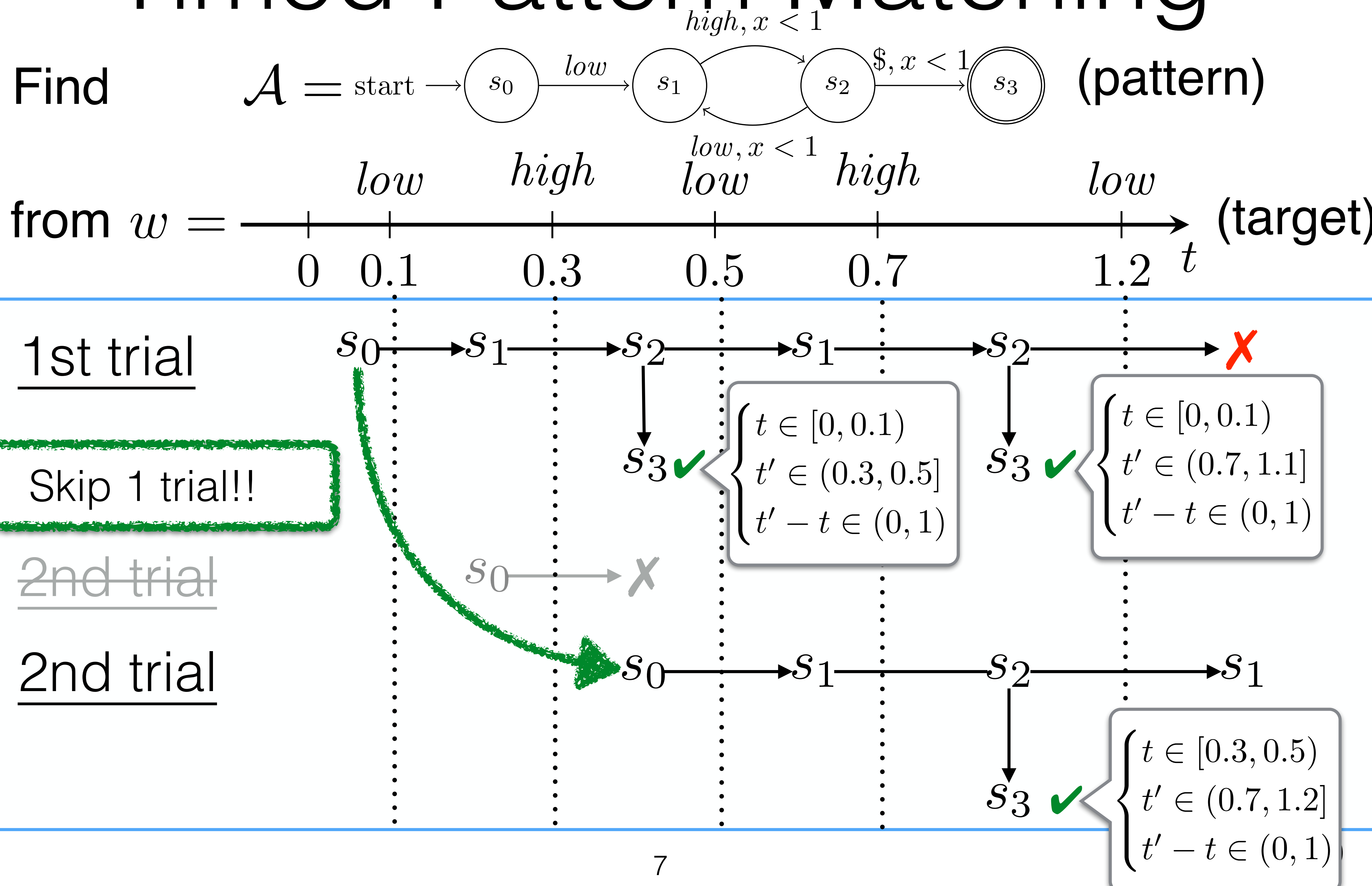
	S	**T**	**R**	I	N	G
✗	*	**S**	**T**	R	I	N
✗	*	*	**S**	T	R	I
✔	*	*	*	S	T	R

M. Waga (NII)

Brute-Force Algorithm for Timed Pattern Matching



KMP-Style Algorithm for Timed Pattern Matching



Problems in Skipping for Timed Pattern Matching

- The length of partial match is unbounded.

String Matching

Table for **length 3** partial match

	S	**T**	**R**	I	N	G
✗	*	**S**	**T**	R	I	N
✗	*	*	**S**	T	R	I
✓	*	*	*	S	T	R

We construct for each length

Timed Pattern Matching

Infinitely Many Tables!!

- Infinitely many timestamps

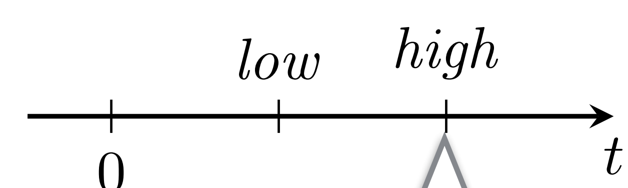
String Matching

Length 3 partial match
in string matching

S T R

Timed Pattern Matching

Length 2 partial match
in timed pattern matching



Infinitely many timestamps

Problems in Skipping for Timed Pattern Matching

- The length of partial match is unbounded.

String Matching

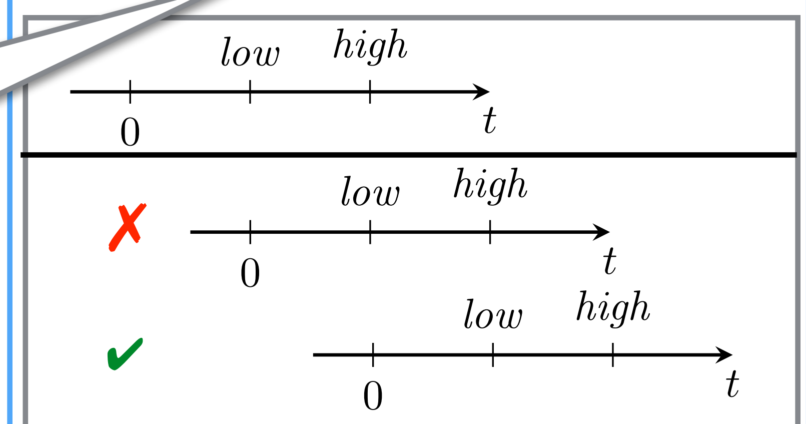
Table for **length 3** partial match

	S	T	R	:	I	N	G
✗	*	S	T	:	R	I	N
✗	*	*	S	:	T	R	I
✓	*	*	*	:	S	T	R

Construct for each **state**

Timed Pattern Matching

Table for **state s_2** partial match



- Infinitely many timestamps

String Matching

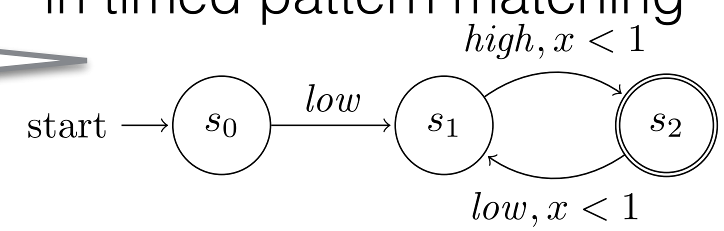
Length 3 partial match
in string matching

S T R

Represent by a timed automaton

Timed Pattern Matching

State s_2 partial match
in timed pattern matching



M. Waga (NII)

Problems in Skipping for Timed Pattern Matching

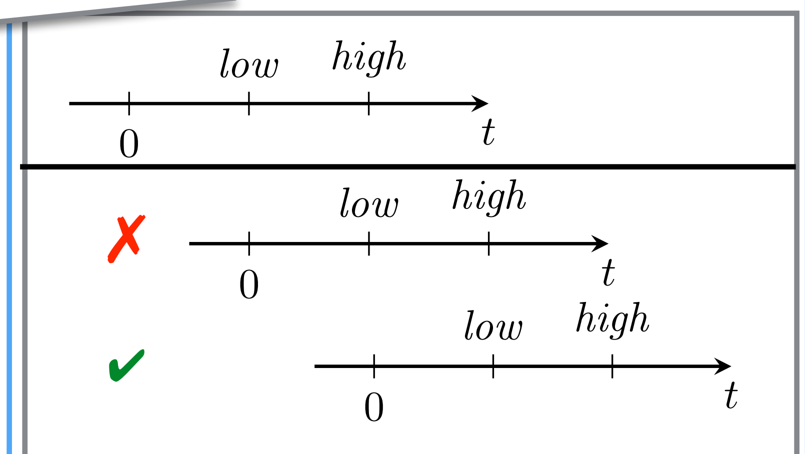
- The length of partial match is unbounded.

Table for **length 3** partial mat

	S	T	R	:	I	N	G
✗	*	S	T	:	R	I	N
✗	*	*	S	:	T	R	I
✓	*	*	*	:	S	T	R

Construct for each **state**

Table for **state s_2** partial match



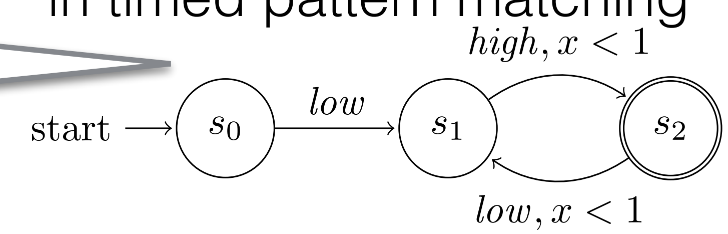
- Infinitely many timestamps

Length 3 partial match
in string matching

S T R

Represent by a timed automaton

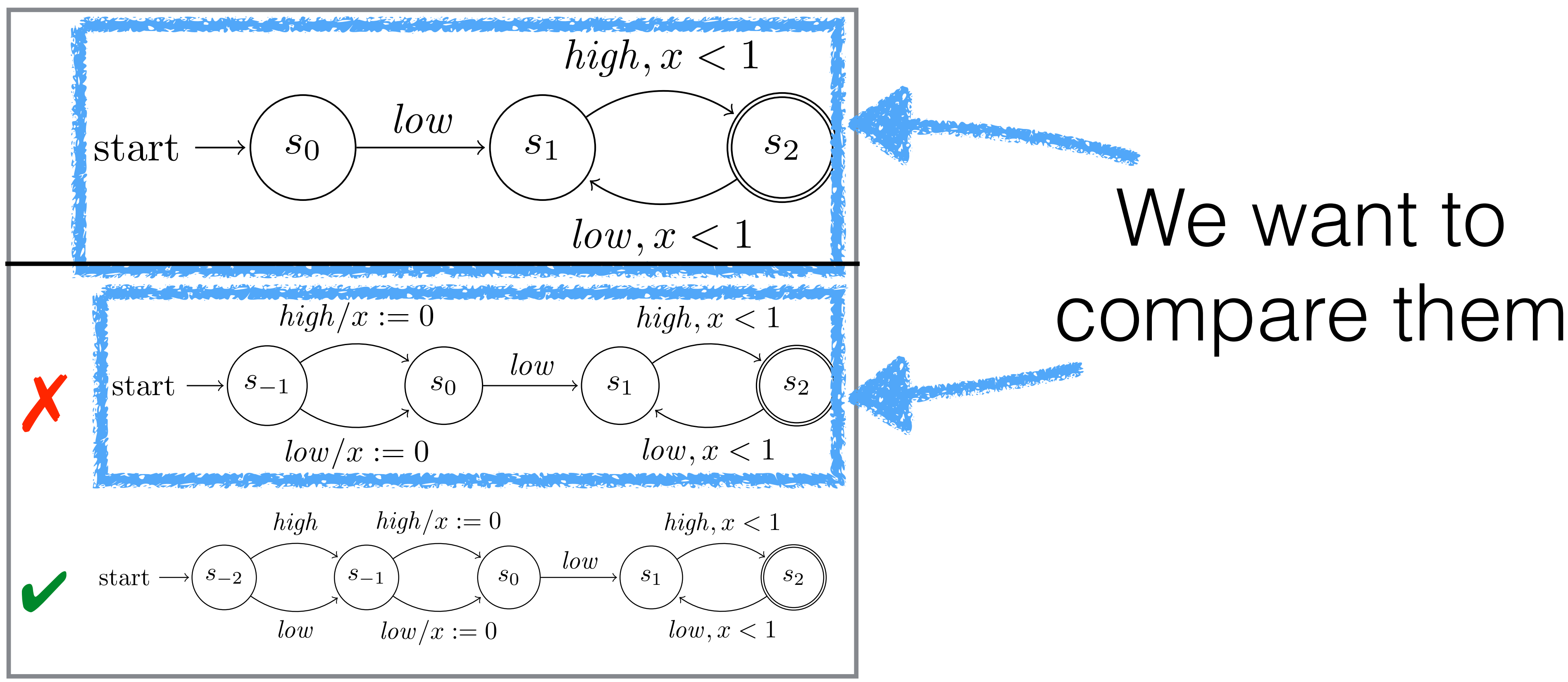
State s_2 partial match
in timed pattern matching



M. Waga (NII)

Skipping for Timed Pattern Matching

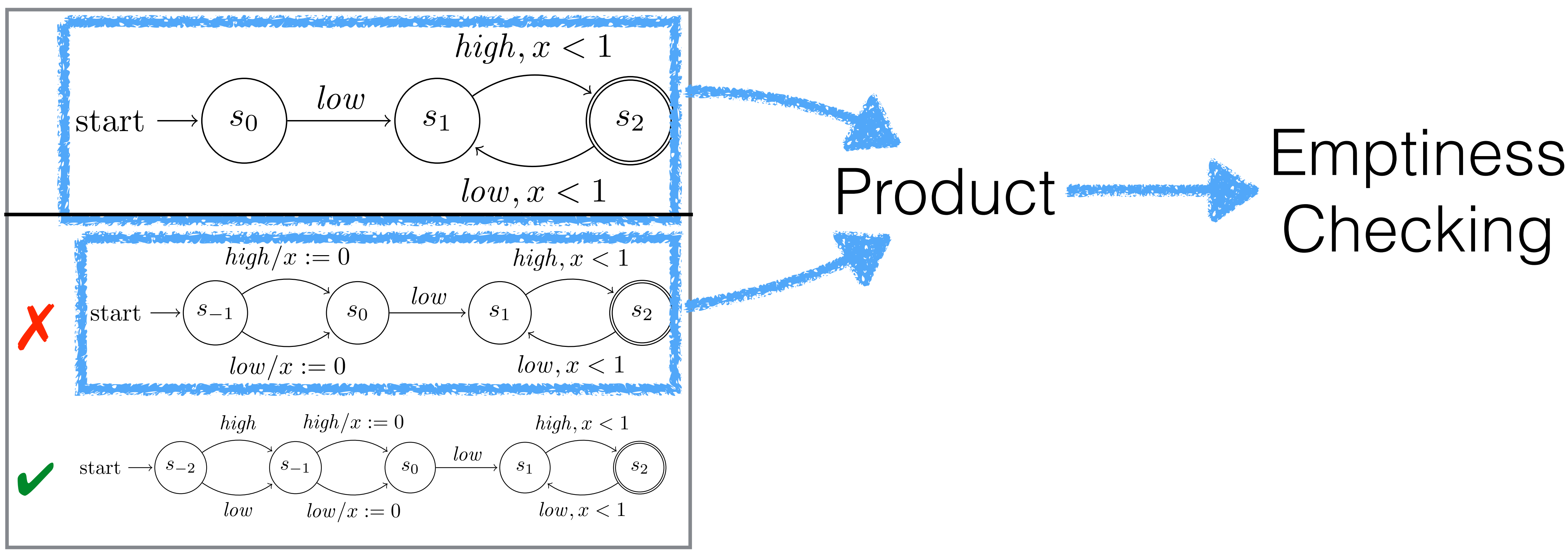
Table for **state s_2** partial match



M. Waga (NII)

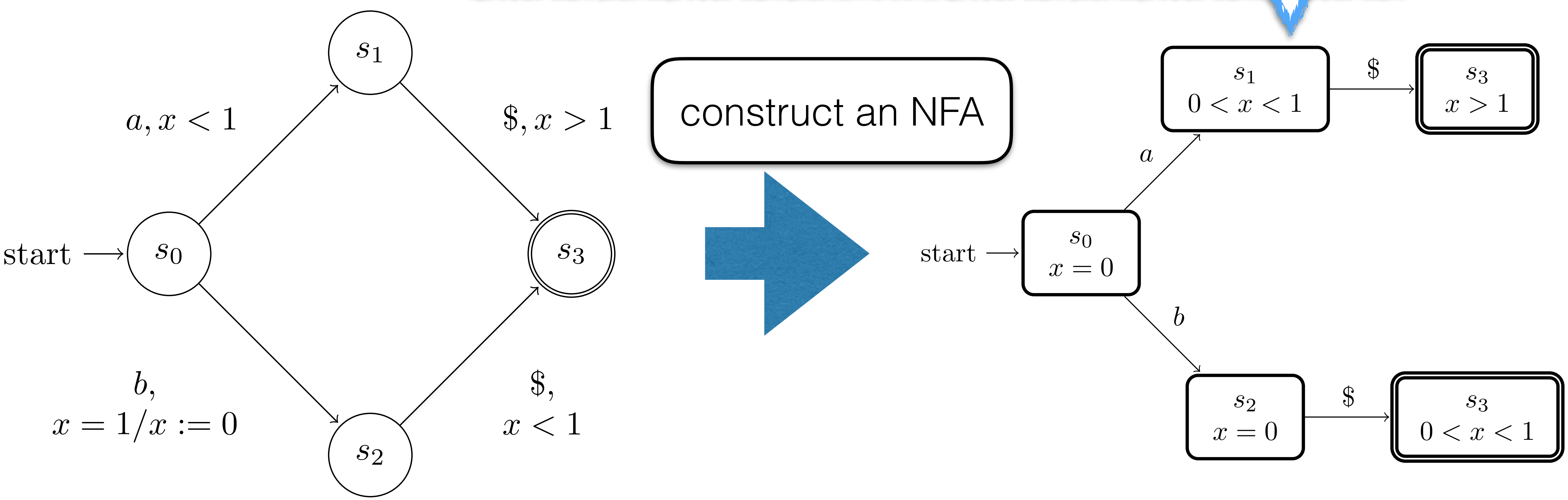
Skipping for Timed Pattern Matching

Table for **state s_2** partial match



Emptiness Checking by Zone Construction

Labelled by a set of "similar" clock valuations represented by a zone



Thm. (soundness and completeness)
Zone automata maintain state reachability.

M. Waga (NII)

Outline

1. Algorithm in MONAA
    - Skipping for timed pattern matching
      [Waga et al., FORMATS'17]
2. Frontend of MONAA
    - Command line interface (CLI) / C++ API
3. Experiments

Input of MONAA (CLI)

```
digraph G {
        1 [init=1][match=0];
        2 [init=0][match=0];
        3 [init=0][match=0];
        4 [init=0][match=0];
        5 [init=0][match=0];
        6 [init=0][match=0];
        7 [init=0][match=1];
        1->2 [label=b][reset="{0}"];
        2->3 [label=a][guard="{x0 < 1}"];
        3->4 [label=a][guard="{x0 < 1}"];
        4->5 [label=a][guard="{x0 < 1}"];
        5->6 [label=a][guard="{x0 < 1}"];
        6->6 [label=a][guard="{x0 < 1}"];
        6->7 [label=a][guard="{x0 > 1}"];
}
```

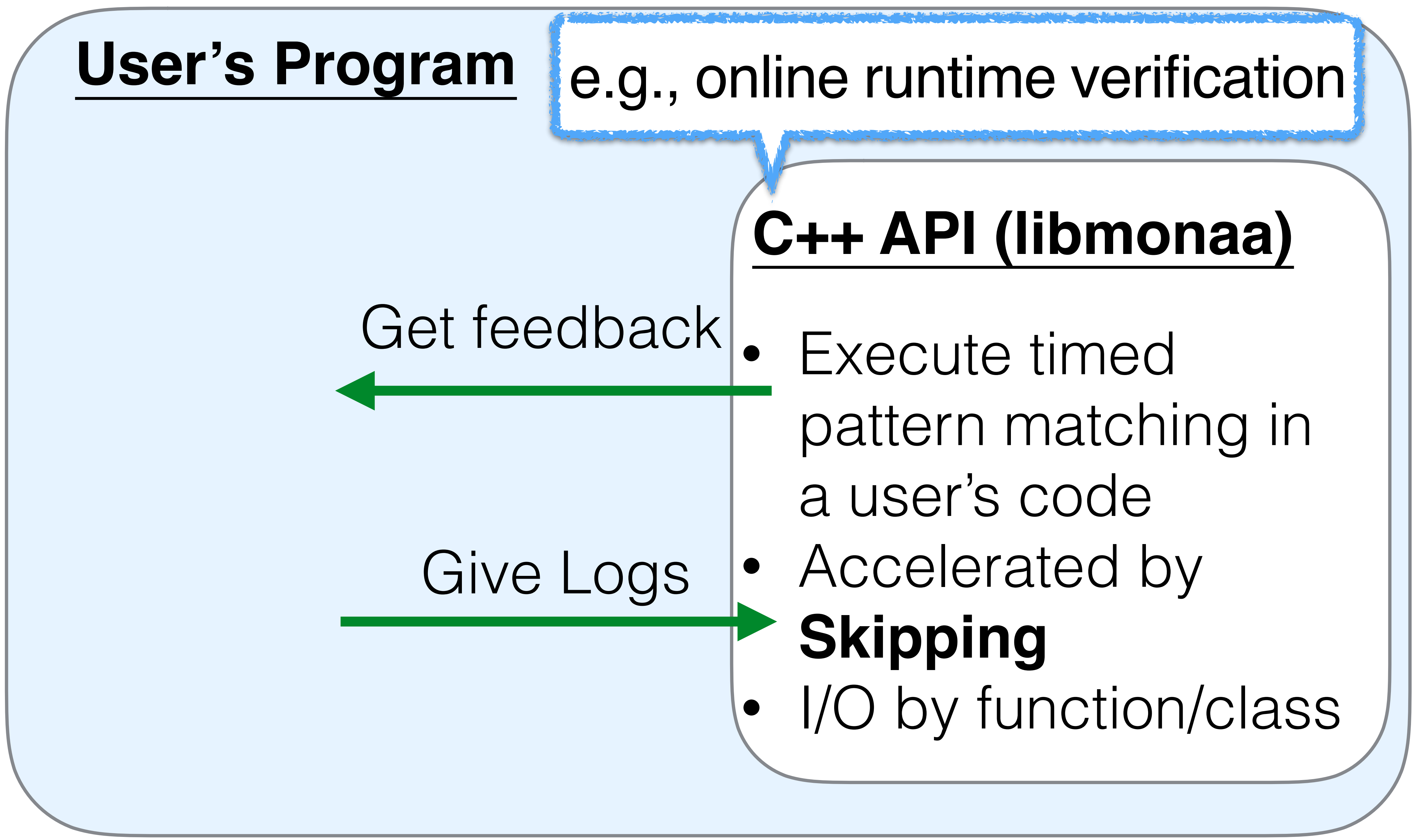
```
a          0.267718
b          0.280545
b          0.293307
b          0.300000
b          0.306016
b          0.318685
b          0.331324
b          0.343941
b          0.356541
b          0.369126
b          0.381700
b          0.394264
b          0.400000
b          0.406823
b          0.411769
b          0.419377
b          0.431929
b          0.444483
b          0.444485
b          0.457039
b          0.457062
b          0.469615
b          0.469677
b          0.482231
```


M. Waga (NII)

Output of MONAA (CLI)

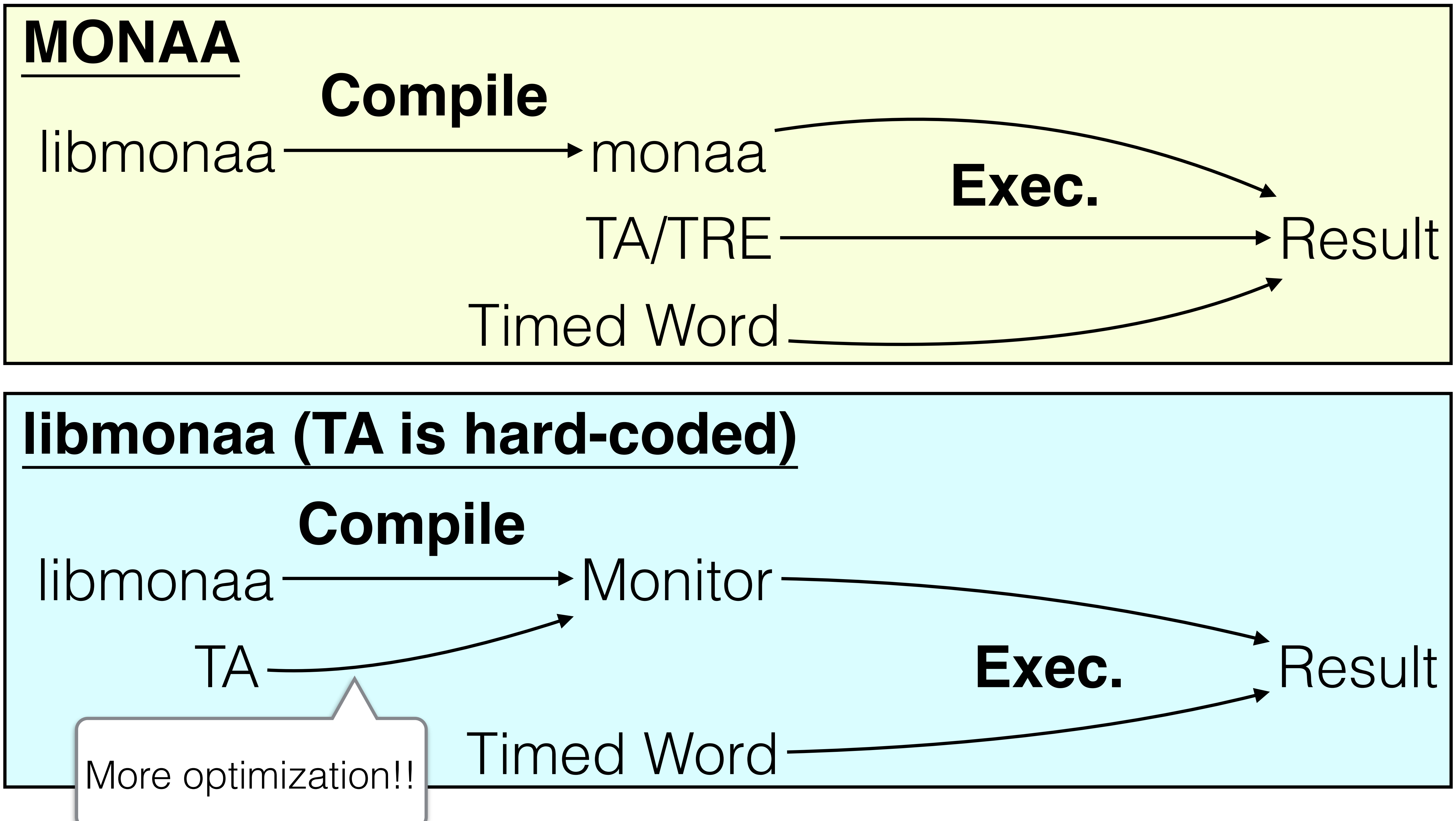
```
Masakis-MacBook-Pro:examples calros$ monaa -f torque.dot  < torque-1000.txt
137.700380       <= t < 137.734850
138.734850        < t' <= 138.744730
  1.000000        < t' - t <=   1.044350
=============================
695.670550       <= t < 695.683090
696.683090        < t' <= 696.684680
  1.000000        < t' - t <=   1.014130
=============================
842.300000       <= t < 842.309420
843.309420        < t' <= 843.315490
  1.000000        < t' - t <=   1.015490
=============================
```

C++ API (libmonaa)



M. Waga (NII)

Better Performance by libmonaa



Outline

1. Algorithm in MONAA
    - Skipping for timed pattern matching
      [Waga et al., FORMATS'17]
2. Frontend of MONAA
    - Command line interface (CLI) / C++ API
3. Experiments

Comparison with Montre

MONAA

- Use timed automata
  - We can also construct a TA from a TRE
- Accelerated by **skipping**
- Both command line and C++ interface

Existing Tool (Montre)

[Ulus, CAV'17]

- Use timed regular expression
- (online) On-the-fly construction of a state machine from TRE
- Only command line interface

Our algorithm should work faster because of skipping

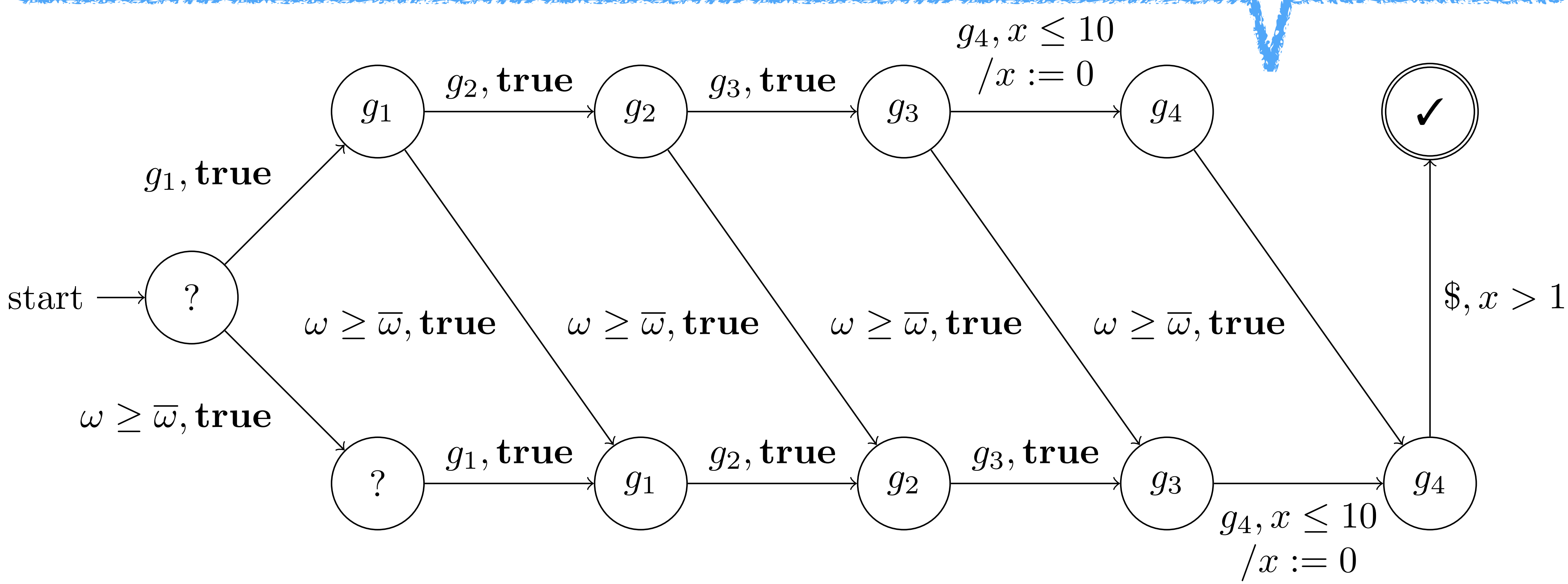
Experiments

- Monitoring of a Simulink model of an automatic transmission in a car.
- Model and spec. are from an automotive benchmark paper [Hoxha et al., ARCH '15]
- Events: gears: g_1, g_2, g_3, g_4
  velocity: $v > \overline{v}$, $v \leq \overline{v}$
  RPM: $\omega > \overline{\omega}$, $\omega \leq \overline{\omega}$

M. Waga (NII)

Experiments

Gear changes from g_1 to g_4 in 10 sec. and RPM changes to high enough, but velocity is still low.



M. Waga (NII)

Result of the Comparison

Table 1. Execution time (sec.)

Length of timed word	`MONAA` (TRE)	`MONAA` (TA)	`libmonaa` (TA is hard coded)	Montre (online)	Montre (offline)
306	7.03	0.80	0.20	0.13	0.03
127,552	7.55	1.27	0.31	37.45	1.56
255,750	8.05	1.73	0.42	75.93	3.13
383,168	8.54	2.21	0.53	115.88	4.69
508,756	9.16	2.69	0.64	153.71	6.21
632,484	9.53	3.14	0.75	189.55	7.75
758,500	10.05	3.60	0.85	216.92	9.33
894,692	10.53	4.06	0.97	260.77	10.88
1,011,426	11.05	4.56	1.07	289.63	12.39

Efficient and online

Blow up!!

Efficient but offline only

M. Waga (NII)

Conclusion

- MONAA can inspect logs with timestamps
  - fast (skipping)
  - simple (text-based I/O)
  - flexible (C++ API)

Future Works

- Theoretical Side:
  - Investigate other techniques for efficient monitoring
- Practical Side:
  - Case study of timed pattern matching

https://github.com/MasWag/monaa

M. Waga (NII)