Parametric Timed Pattern Matching

Masaki Waga[1], Étienne André[2], Ichiro Hasuo[3]

Kyoto University[1], Université de Lorraine[2],
National Institute of Informatics[3]

SES 2022 6th Sep. 2022

This work is partially supported by JST ERATO HASUO Metamathematics for Systems Design Project (No.JPMJER1603), by JST ACT-X Grant No. JPMJAX200U, by JSPS Grants-in-Aid No. 15KT0012 & 18J22498, by JST CEREST Grant No. JPMJCR2012, by the ANR national research program PACS (ANR-14-CE28-0002) and by theANR-NRF French-Singaporean research program ProMiS (ANR-19-CE25-0015)

形式仕様を使った
実行時モニタリング

Parametric Timed Pattern Matching

Masaki Waga[1], Étienne André[2], Ichiro Hasuo[3]

Kyoto University[1], Université de Lorraine[2],
National Institute of Informatics[3]

SES 2022 6th Sep. 2022

This work is partially supported by JST ERATO HASUO Metamathematics for Systems Design Project (No.JPMJER1603), by JST ACT-X Grant No. JPMJAX200U, by JSPS Grants-in-Aid No. 15KT0012 & 18J22498, by JST CEREST Grant No. JPMJCR2012, by the ANR national research program PACS (ANR-14-CE28-0002) and by theANR-NRF French-Singaporean research program ProMiS (ANR-19-CE25-0015)

形式仕様を使った
実行時モニタリング

パラメタ??

Parametric Timed Pattern Matching

Masaki Waga[1], Étienne André[2], Ichiro Hasuo[3]

Kyoto University[1], Université de Lorraine[2],
National Institute of Informatics[3]

SES 2022 6th Sep. 2022

This work is partially supported by JST ERATO HASUO Metamathematics for Systems Design Project (No.JPMJER1603), by JST ACT-X Grant No. JPMJAX200U, by JSPS Grants-in-Aid No. 15KT0012 & 18J22498, by JST CEREST Grant No. JPMJCR2012, by the ANR national research program PACS (ANR-14-CE28-0002) and by theANR-NRF French-Singaporean research program ProMiS (ANR-19-CE25-0015)

実行時モニタリング

デプロイ後: 経年劣化・異常事態の検出など

開発中: テストオラクルとして

モニタリング手法の比較

「目grep」

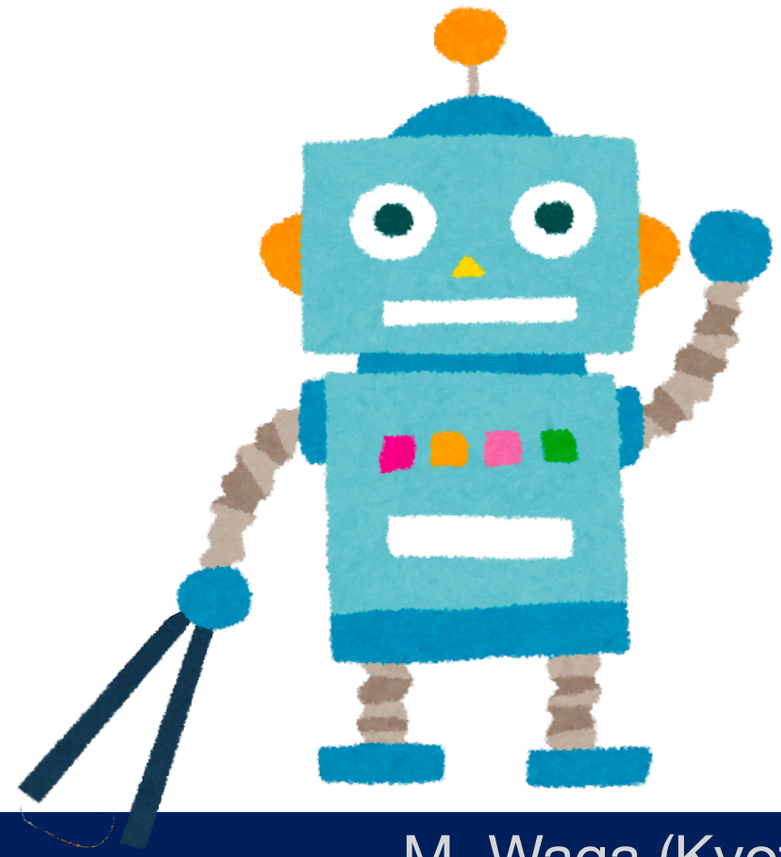
	形式仕様による モニタリング	人手による モニタリング
スケーラビリティ	高	低
属人性	低	高
再利用性	高	低

モニタリング手法の比較

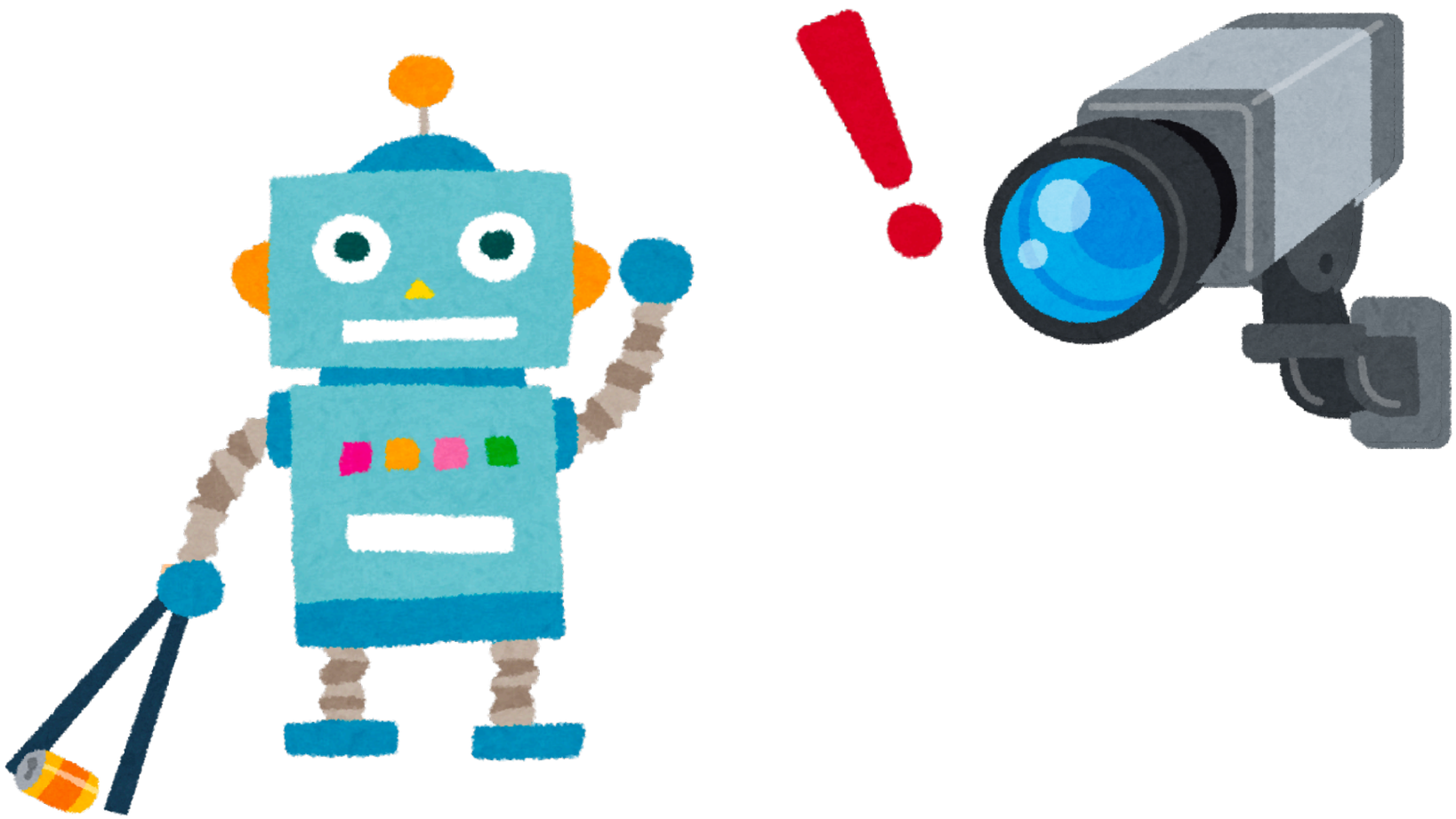
「目grep」

	形式仕様による モニタリング	人手による モニタリング
スケーラビリティ	高	低
属人性	低	高
再利用性	高	低

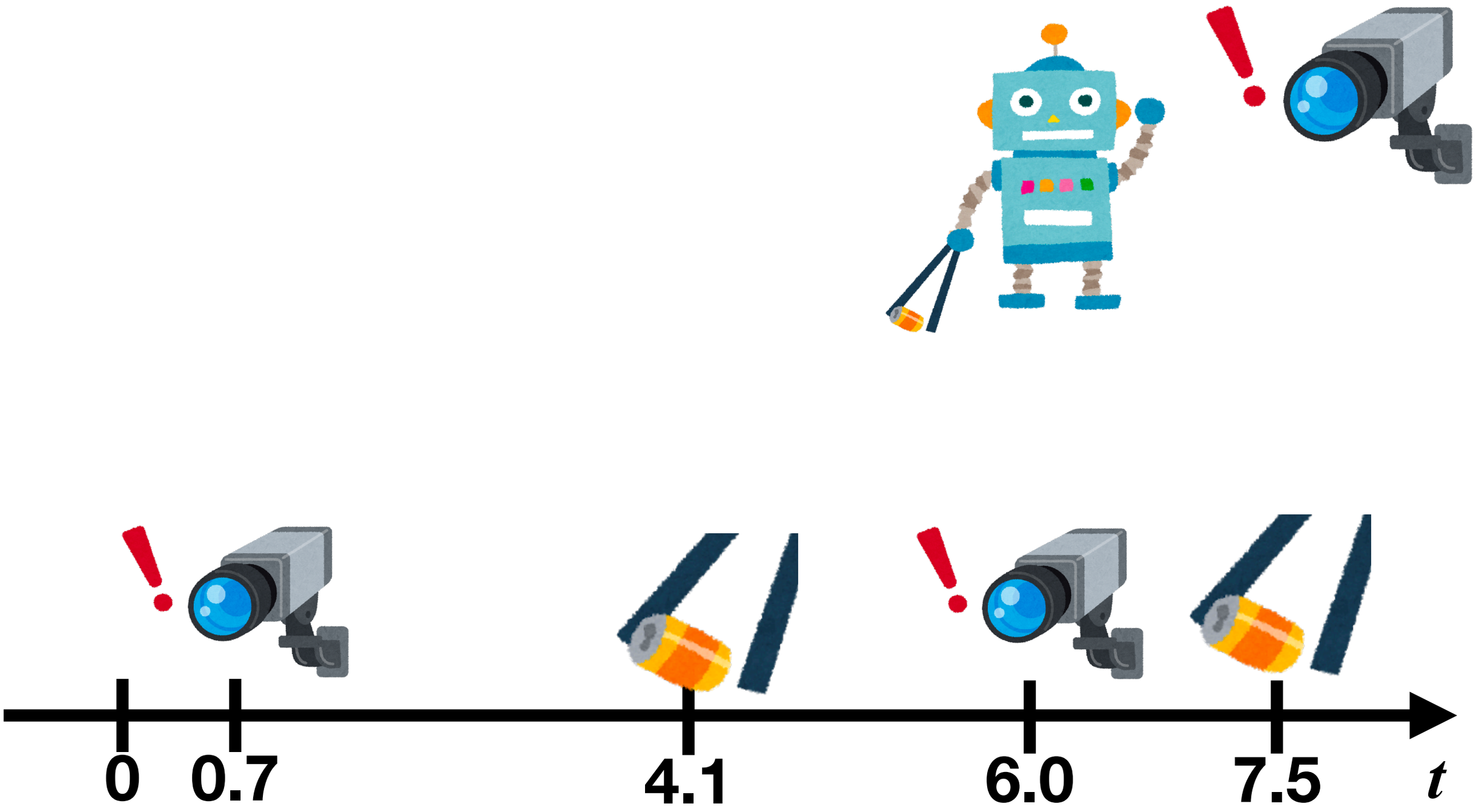
例: 清掃ロボットのモニタリング



例: 清掃ロボットのモニタリング



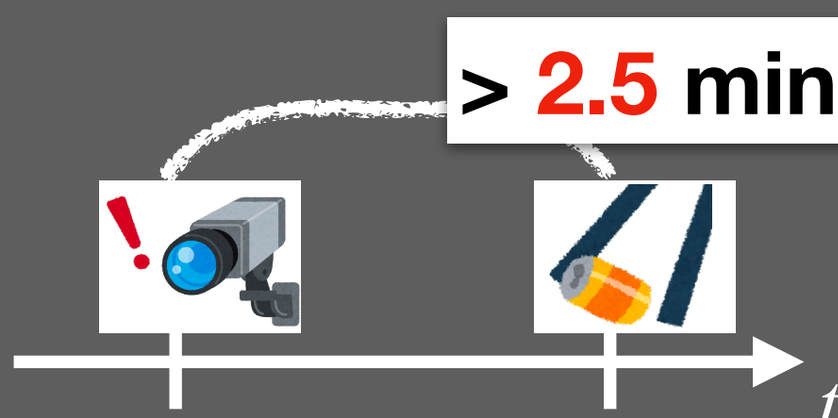
例: 清掃ロボットのモニタリング

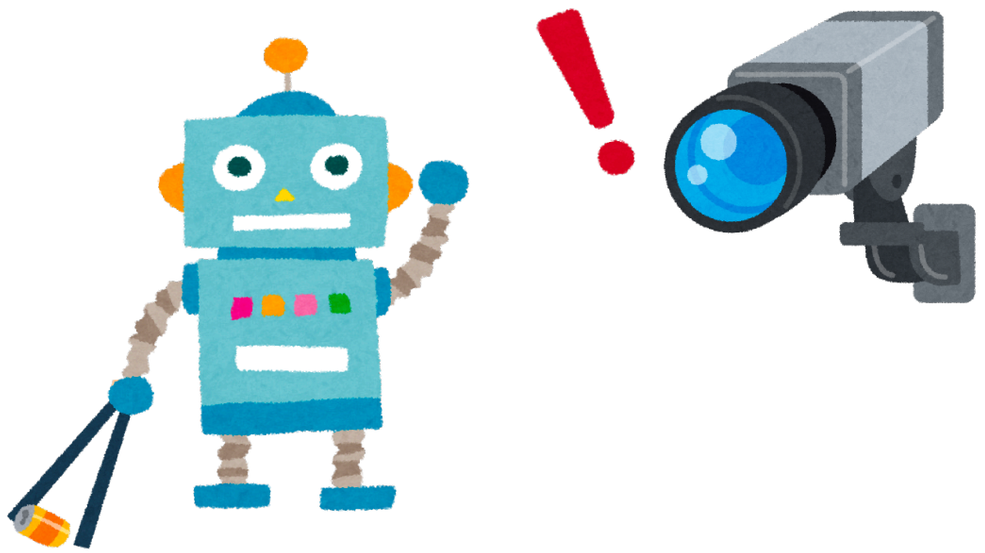


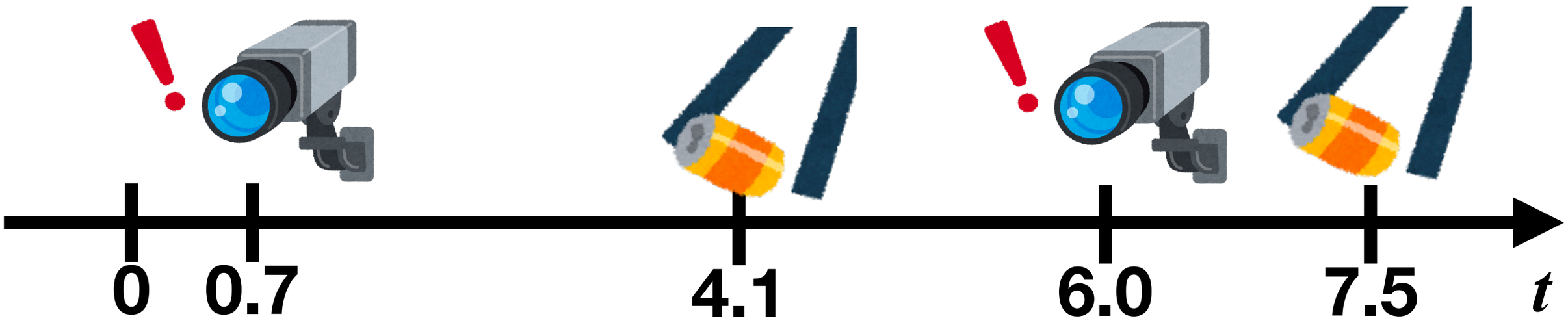
例: 清掃ロボットのモニタリング

Spec.

No



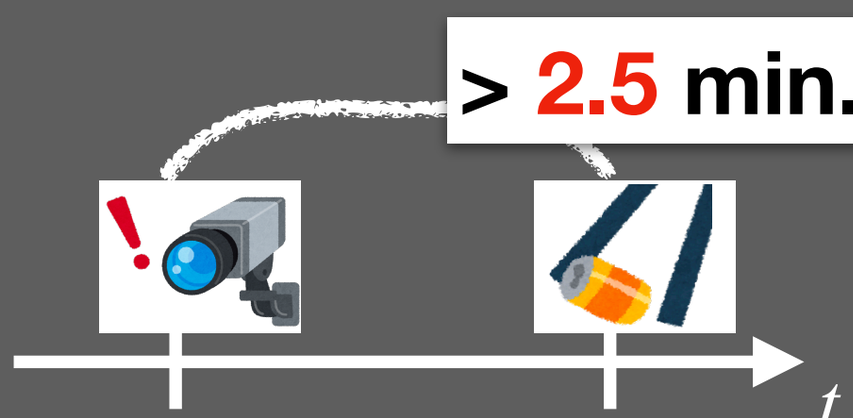


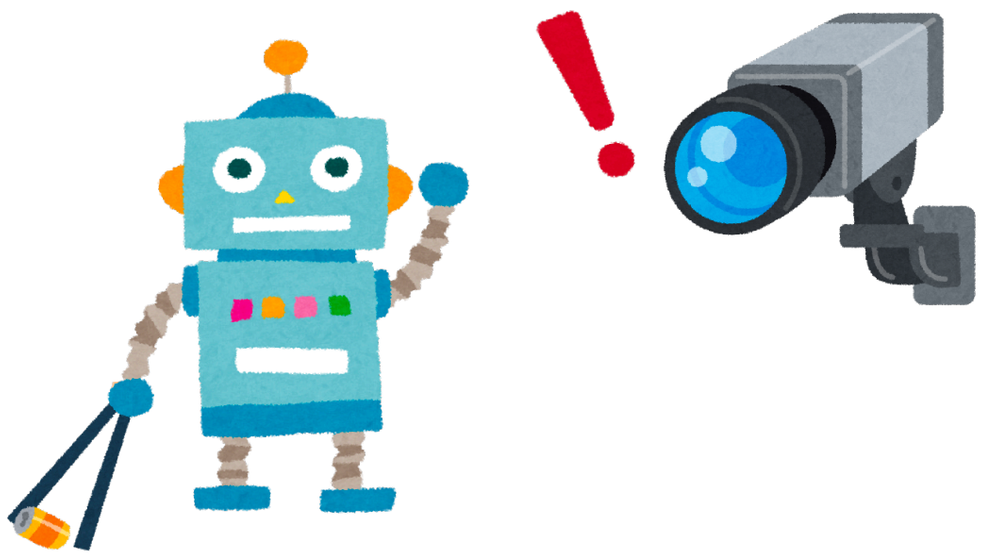


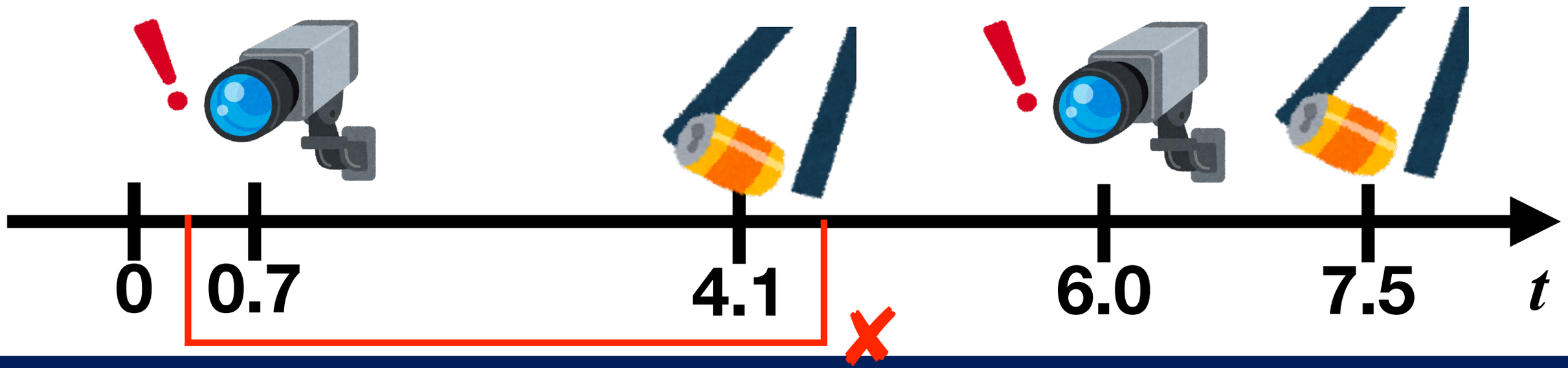
例: 清掃ロボットのモニタリング

Spec.

No



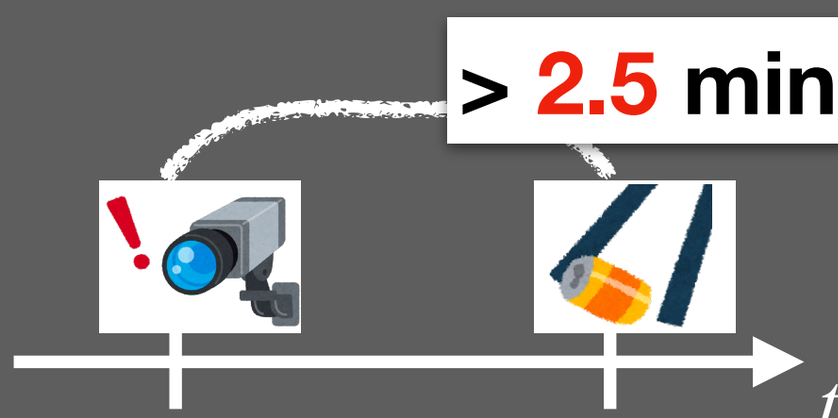


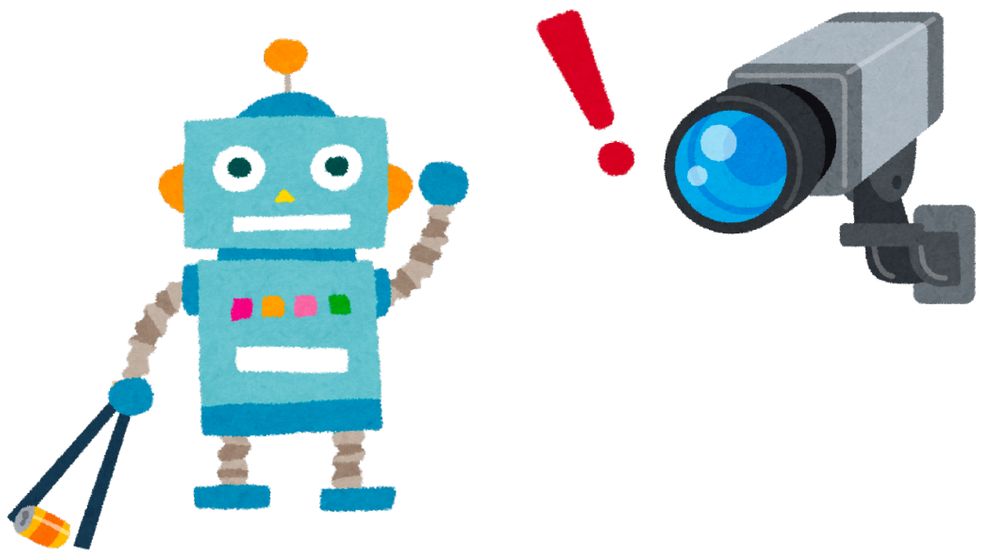


例: 清掃ロボットのモニタリング

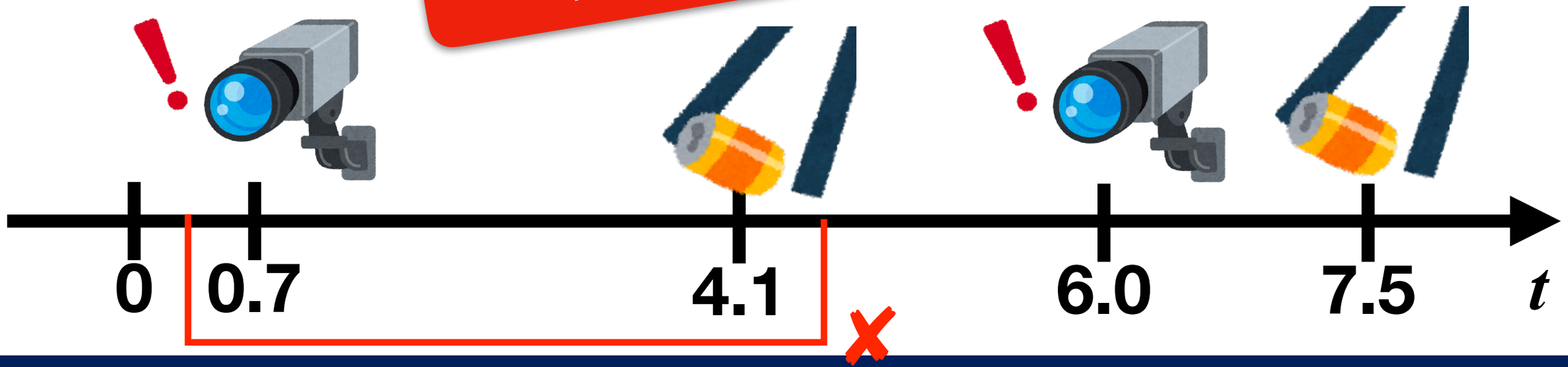
Spec.

No





閾値を決めるのは大変!!



パラメタを用いた仕様記述

[Our contribution]

Concrete Spec.

> 2.5 min.

No

閾値の決定が困難

パラメタを用いて仕様を記述

→ パラメタを生成

Parametric Spec.

> p min.

No

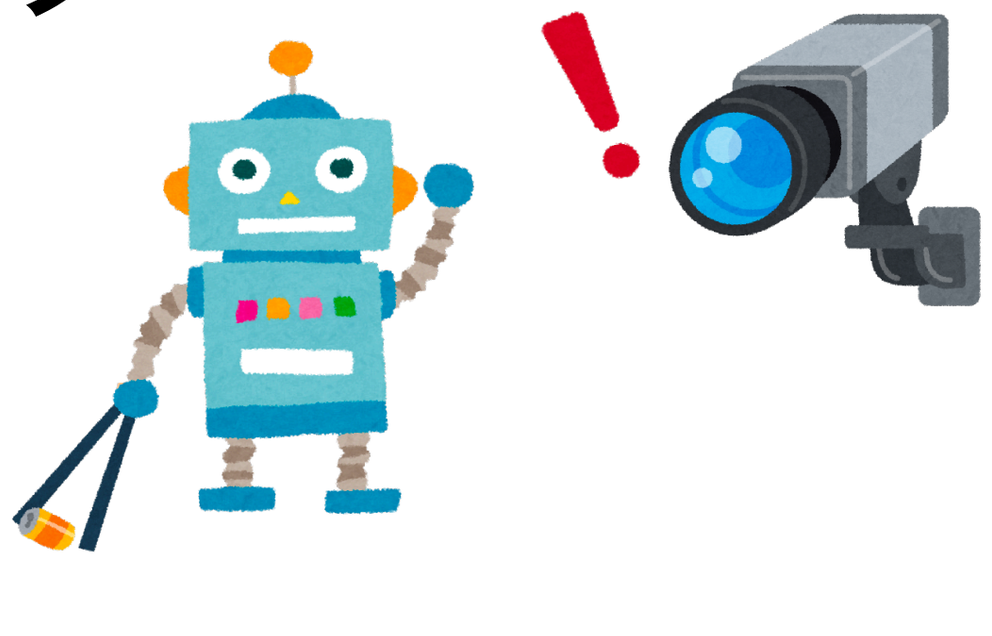
閾値を生成
量的な結果

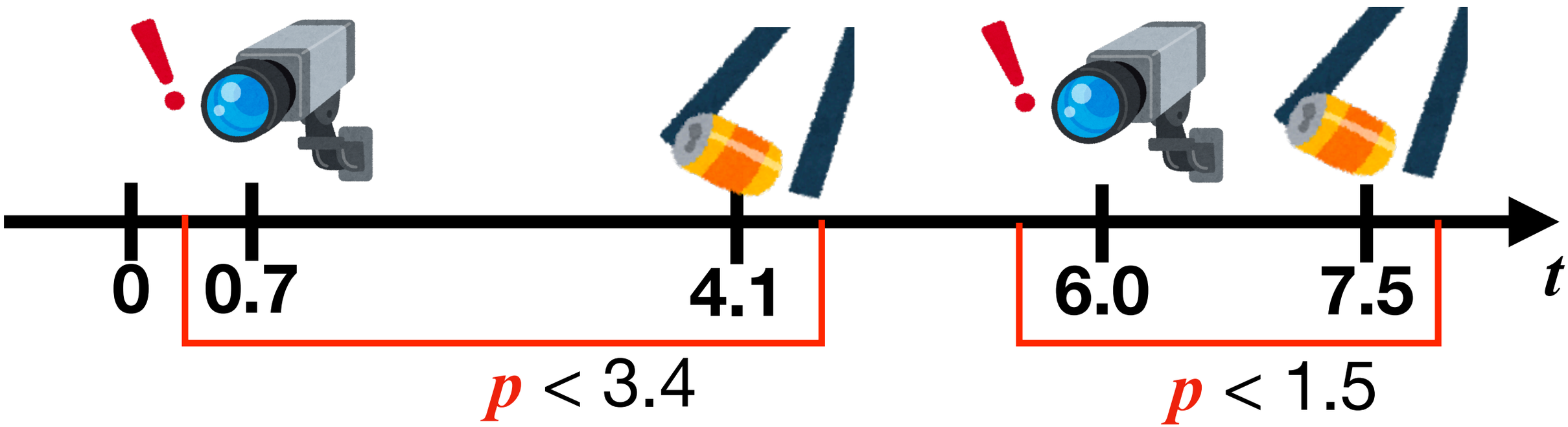
パラメタ付き時間パターンマッチング

[Our Contribution]

Parametric Spec.

No





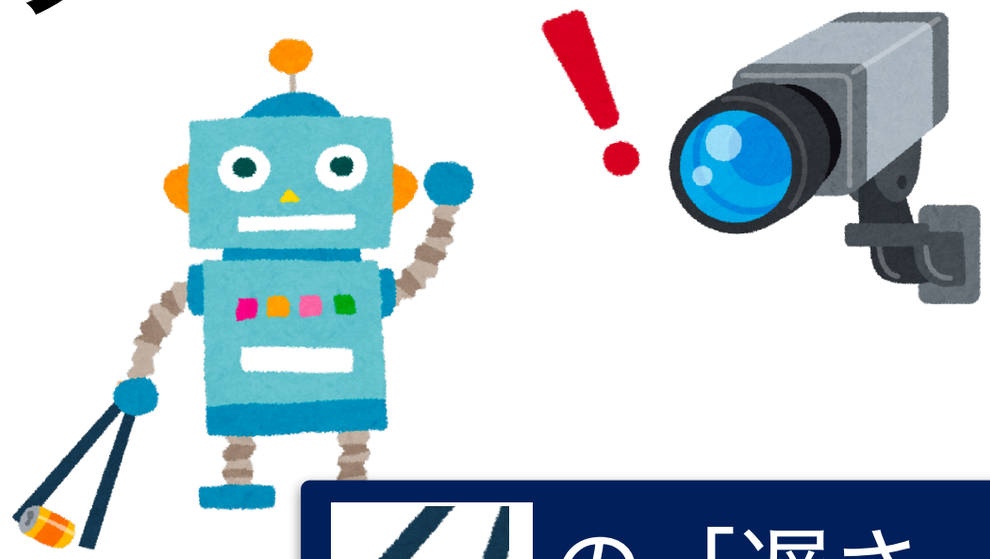
パラメタ付き時間パターンマッチング

Parametric Spec.

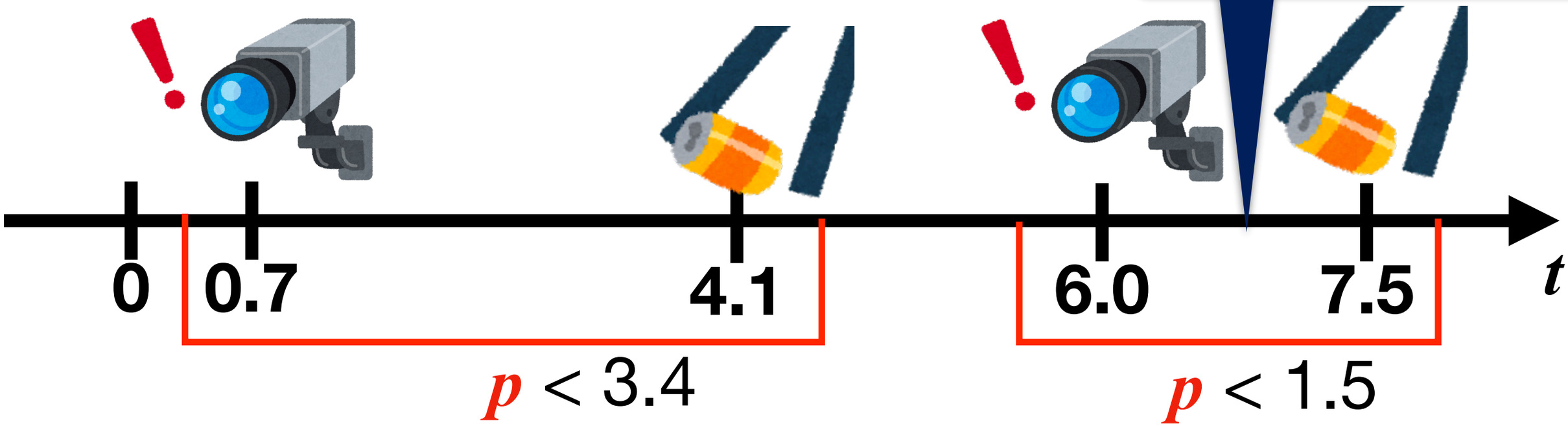
> p min.

No

[Our Contribution]



の「遅さ」



貢献

- パラメタ付き時間パターンマッチング問題を定式化
  - パラメタを用いた仕様記述 → パラメタを生成
  - パターンにマッチする領域とパラメタを全列挙

- 二種類のアルゴリズムを提案
  - 手法1: パラメタ付き時間オートマトンのモデル検査に帰着
  - 手法2: 正規表現マッチングに類似、より直接的

- 実験的に効率性を評価 → 30 us / event 程度

Outline

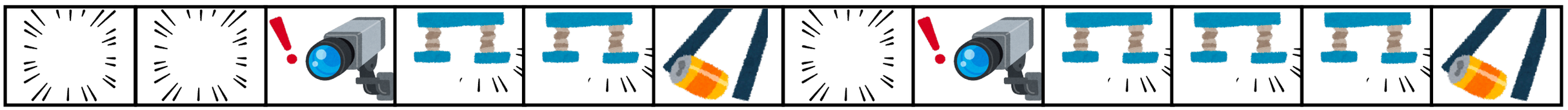
- パラメタ付き時間パターンマッチング問題を定式化
  - パラメタを用いた仕様記述 → パラメタを生成
  - パターンにマッチする領域とパラメタを全列挙
- 二種類のアルゴリズムを提案
  - 手法1: パラメタ付き時間オートマトンのモデル検査に帰着
  - 手法2: 正規表現マッチングに類似、より直接的
- 実験的に効率性を評価 → 30 us / event 程度

正規言語マッチング

Pattern

0回以上の繰り返し

Log

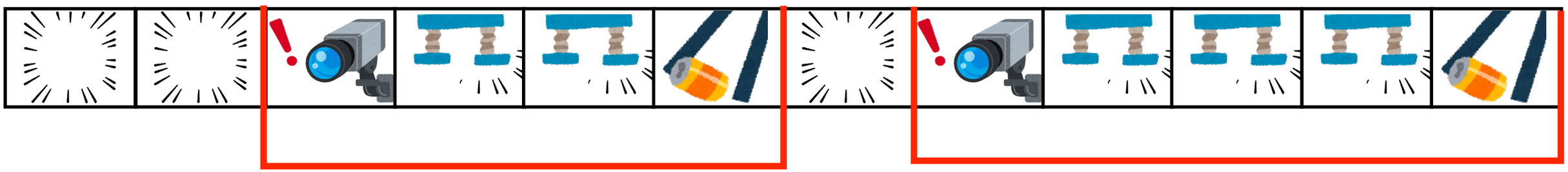


正規言語マッチング

Pattern

0回以上の繰り返し

Log

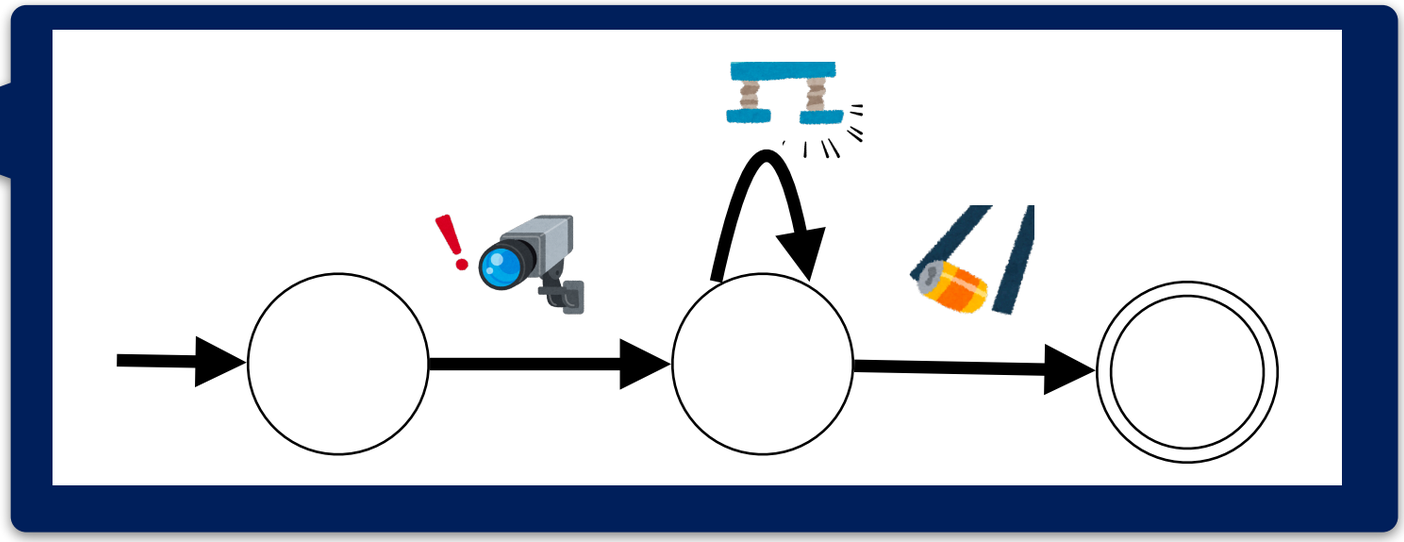


範囲の集合: {(2, 5), (7, 11)}

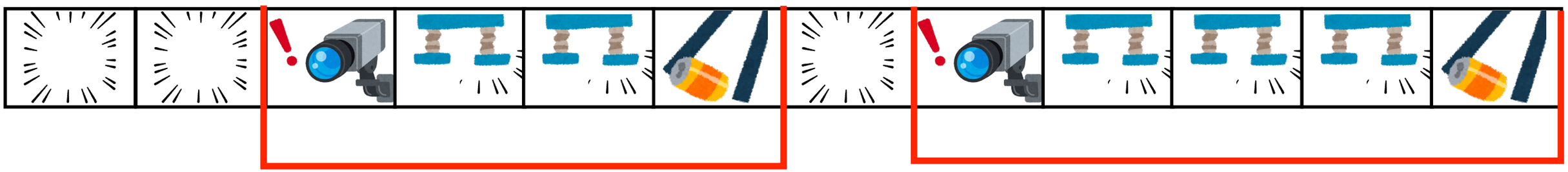
正規言語マッチング

Pattern

0回以上の繰り返し



Log

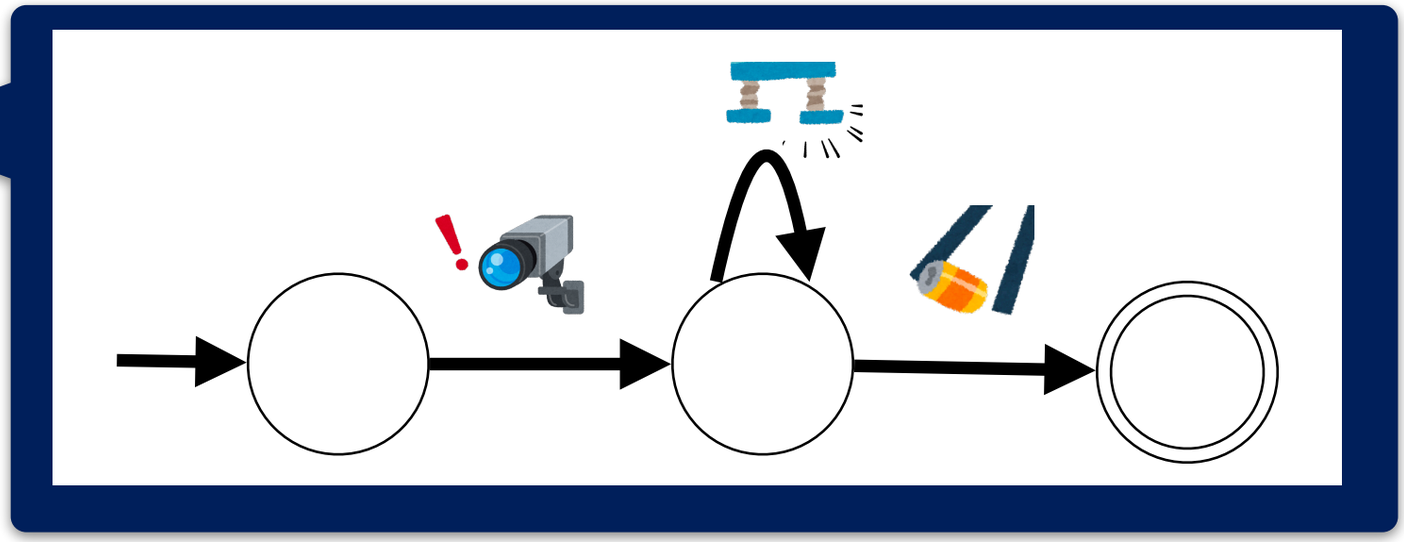


範囲の集合: {(2, 5), (7, 11)}

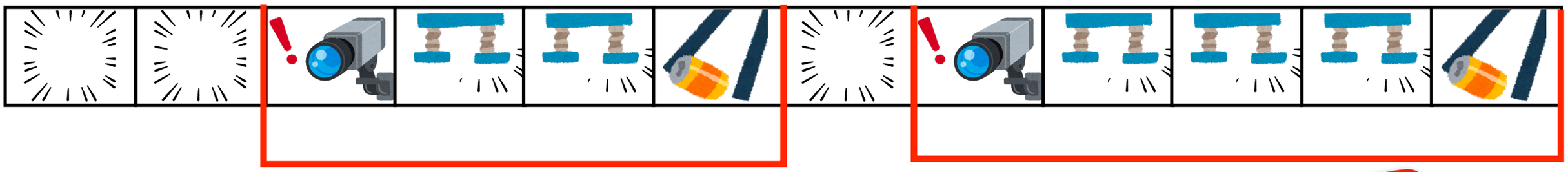
正規言語マッチング

Pattern

0回以上の繰り返し



Log



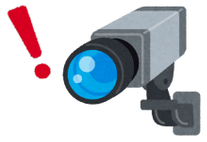
範囲の集合: {(2, 5), (7, 11)}

時間の扱いが
離散的!!

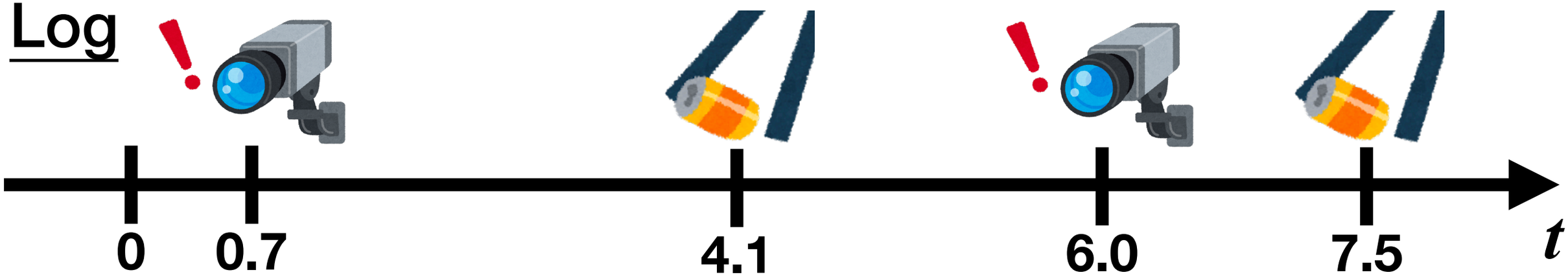
時間パターンマッチング

[Ulus+, FORMATS'14] [Waga+, FORMATS'16]

Pattern

 と  の間隔が 2.5min. 以上

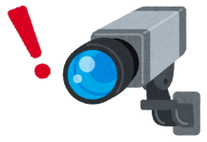
Log



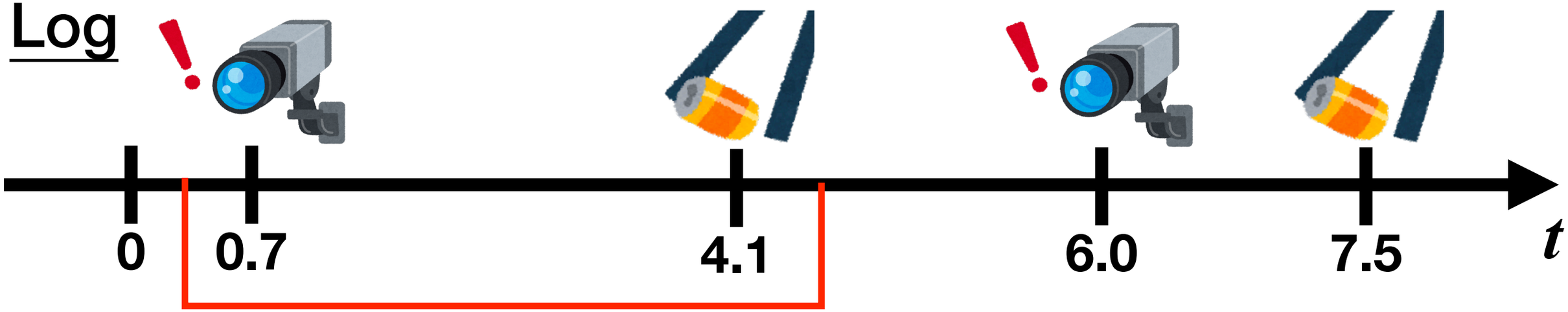
時間パターンマッチング

[Ulus+, FORMATS'14] [Waga+, FORMATS'16]

Pattern

 と  の間隔が 2.5min. 以上

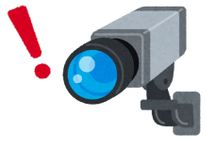
Log



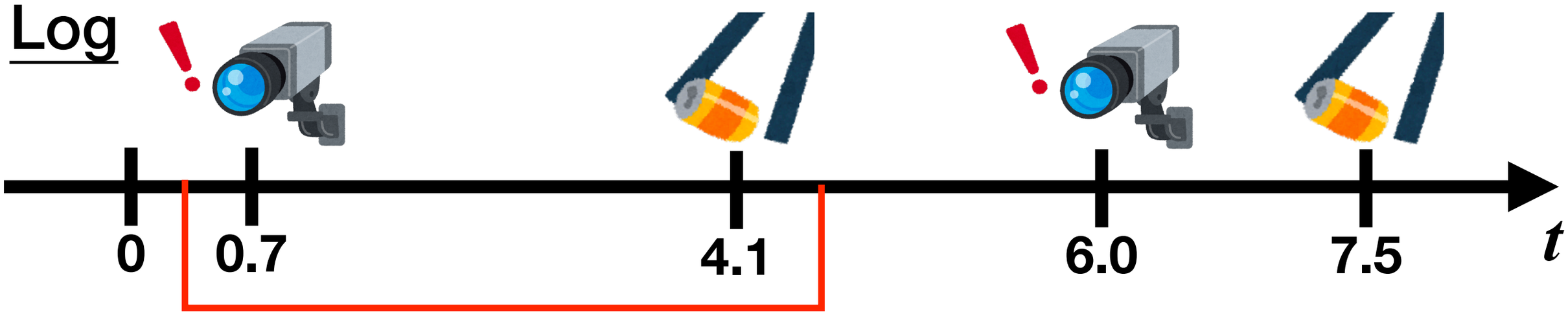
時間パターンマッチング

[Ulus+, FORMATS'14] [Waga+, FORMATS'16]

Pattern

 と  の間隔が 2.5min. 以上

Log

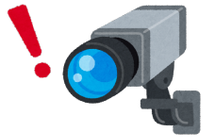


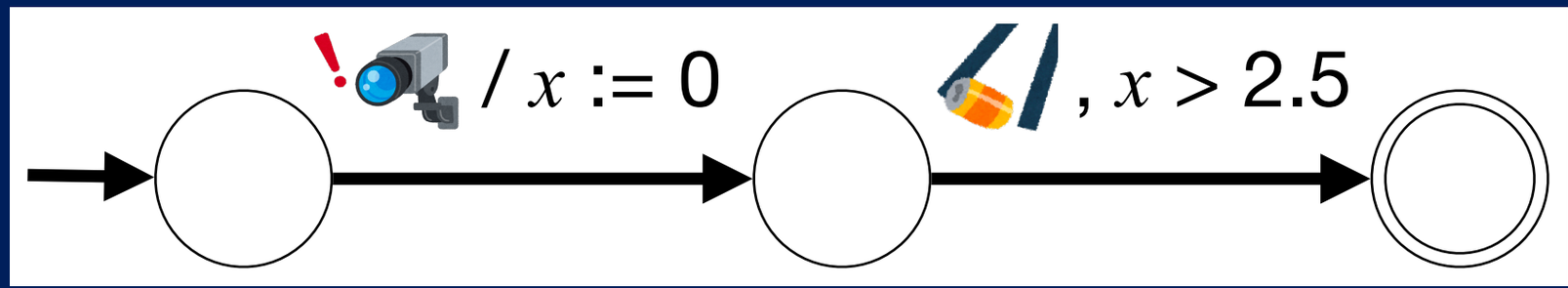
区間の集合 $\{(t, t') \mid 0 \leq t < 0.7, 4.1 < t' \leq 6.0\}$

時間パターンマッチング

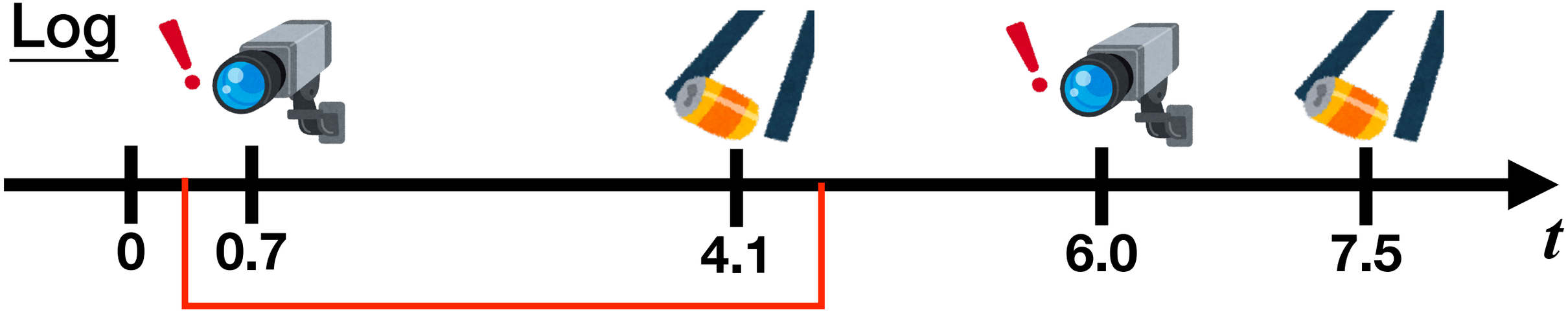
[Ulus+, FORMATS'14] [Waga+, FORMATS'16]

Pattern

 と  の間隔が 2.5min. 以上



Log

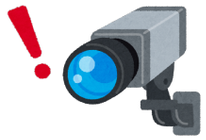


区間の集合 $\{(t, t') \mid 0 \leq t < 0.7, 4.1 < t' \leq 6.0\}$

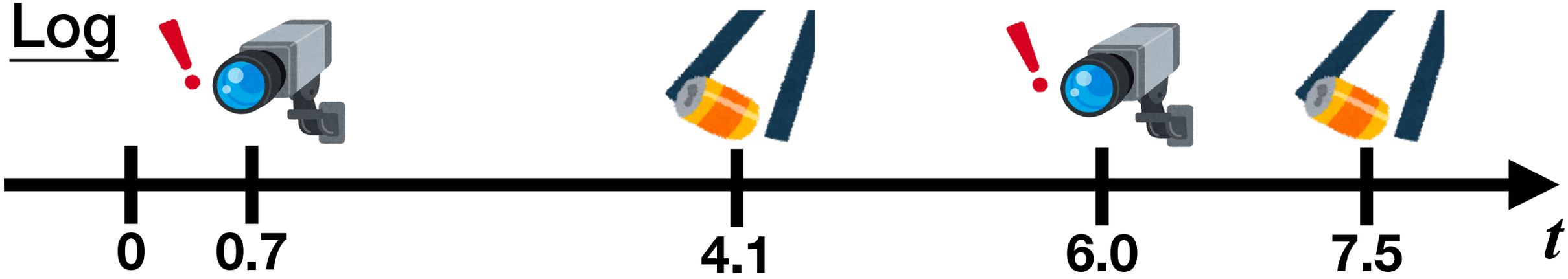
パラメタ付き時間パターンマッチング

[Our Contribution]

Pattern

 と  の間隔が p min. 以上

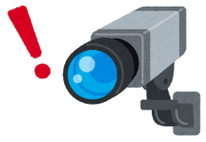
Log



パラメタ付き時間パターンマッチング

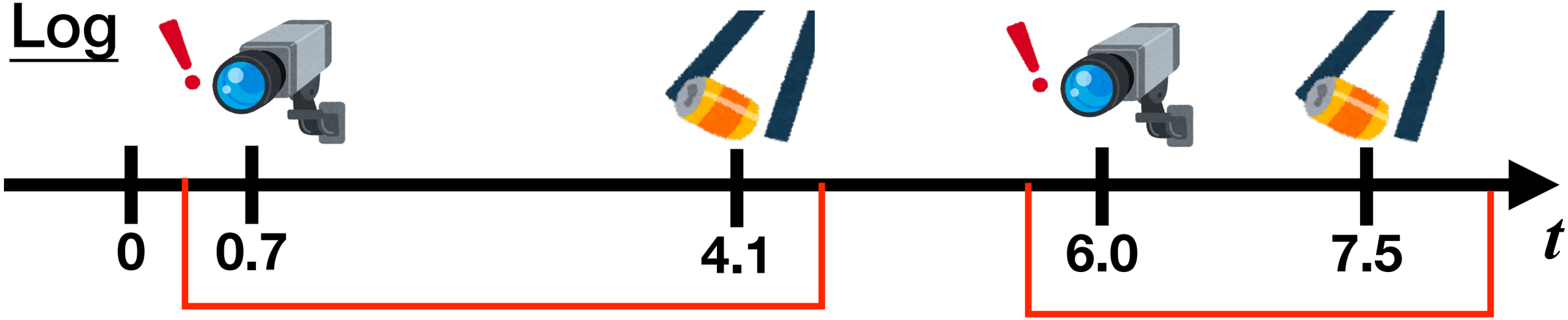
[Our Contribution]

Pattern


と

の間隔が p min. 以上

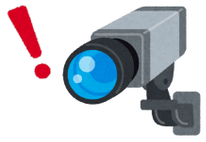
Log



パラメタ付き時間パターンマッチング

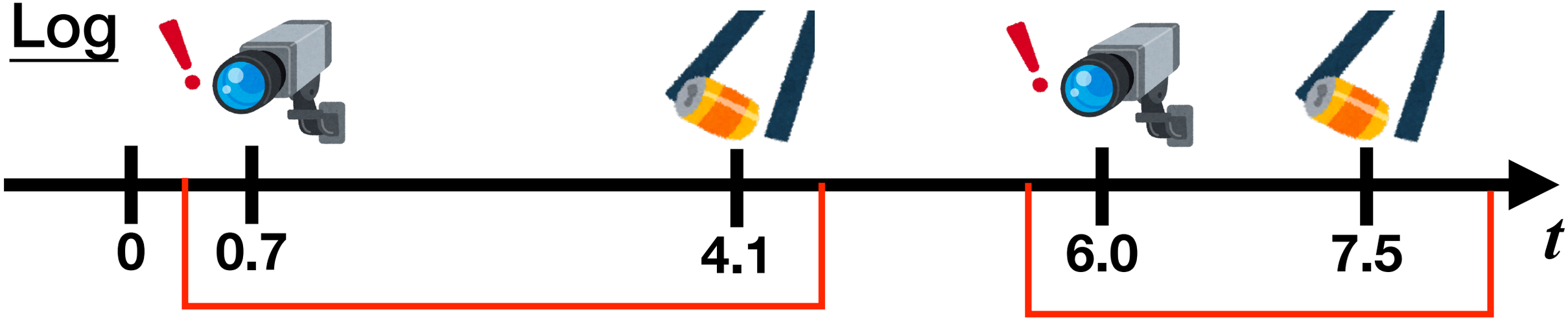
[Our Contribution]

Pattern


と

の間隔が p min. 以上

Log



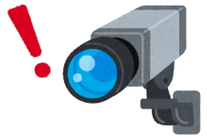
$\{(t, t', p) \mid 0 \leq t < 0.7, 4.1 < t' \leq 6.0, p > 3.4\}$

区間とパラメタの集合　$\{(t, t', p) \mid 4.1 \leq t < 6.0, 7.5 < t', p > 1.5\}$

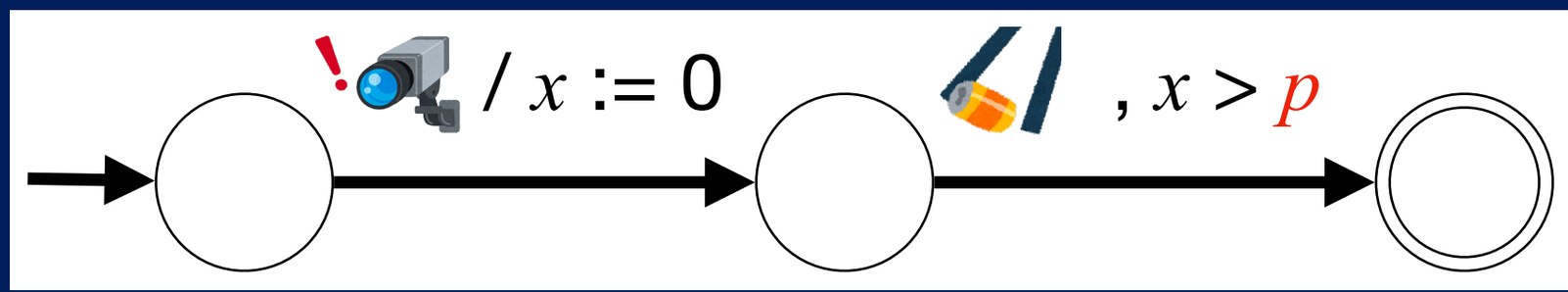
パラメタ付き時間パターンマッチング

[Our Contribution]

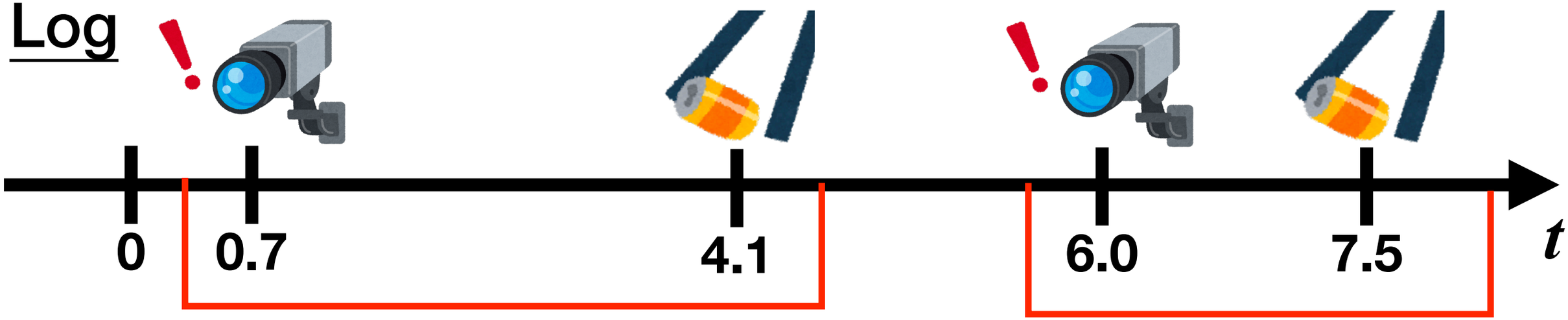
Pattern


と

の間隔が p min. 以上



Log



$\{(t, t', p) \mid 0 \leq t < 0.7, 4.1 < t' \leq 6.0, p > 3.4\}$

区間とパラメタの集合　　$\{(t, t', p) \mid 4.1 \leq t < 6.0, 7.5 < t', p > 1.5\}$

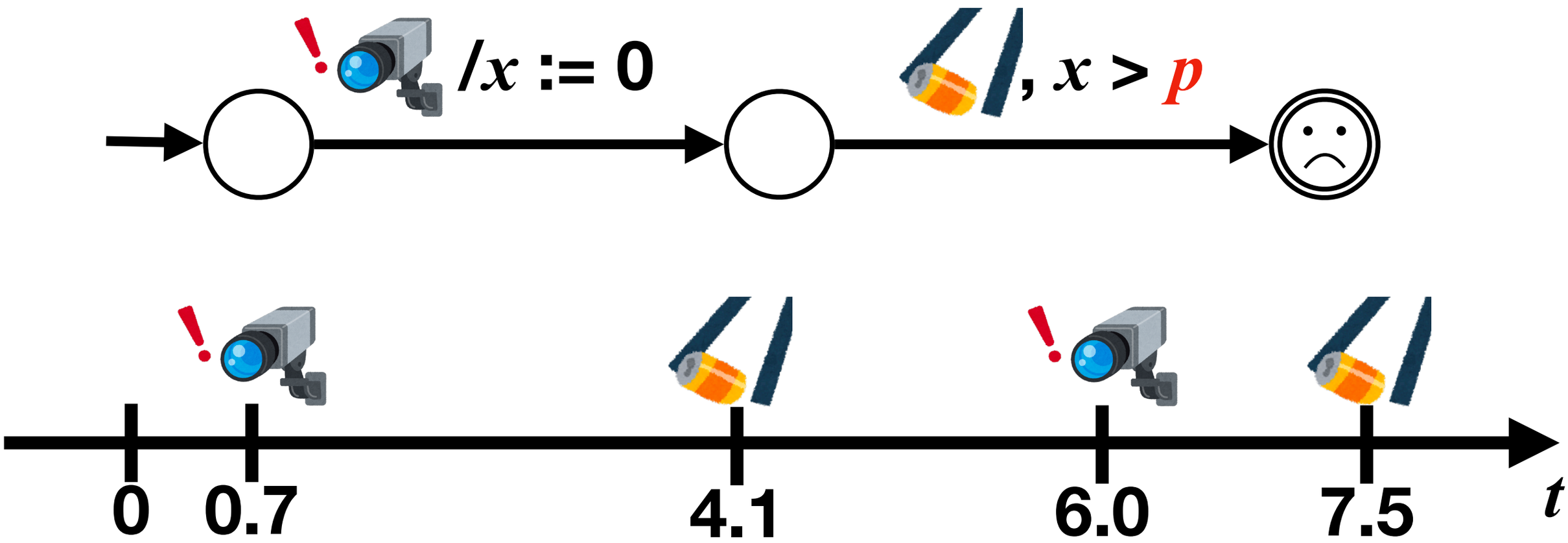
Outline

- パラメタ付き時間パターンマッチング問題を定式化
  - パラメタを用いた仕様記述 → パラメタを生成
  - パターンにマッチする領域とパラメタを全列挙

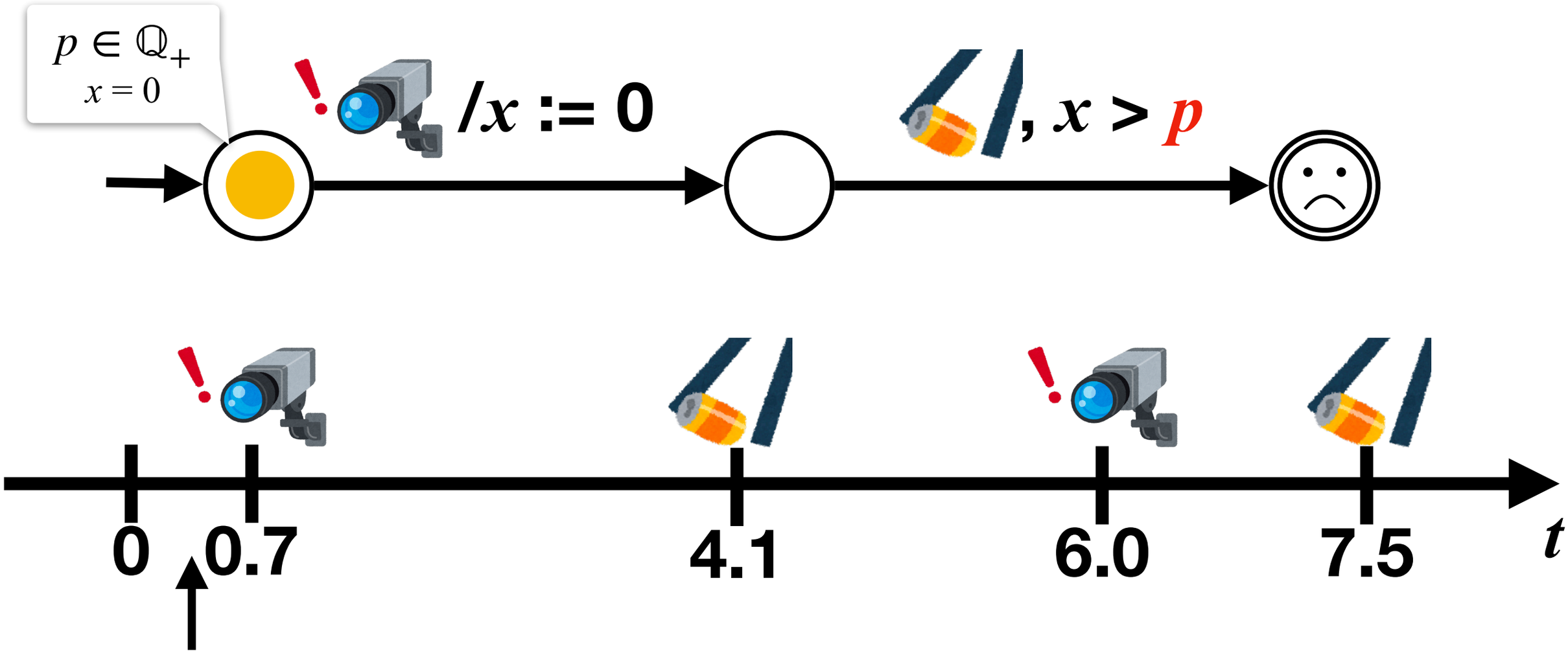
- 二種類のアルゴリズムを提案
  - 手法1: パラメタ付き時間オートマトンのモデル検査に帰着
  - 手法2: 正規表現マッチングに類似、より直接的

- 実験的に効率性を評価 → 30 us / event 程度

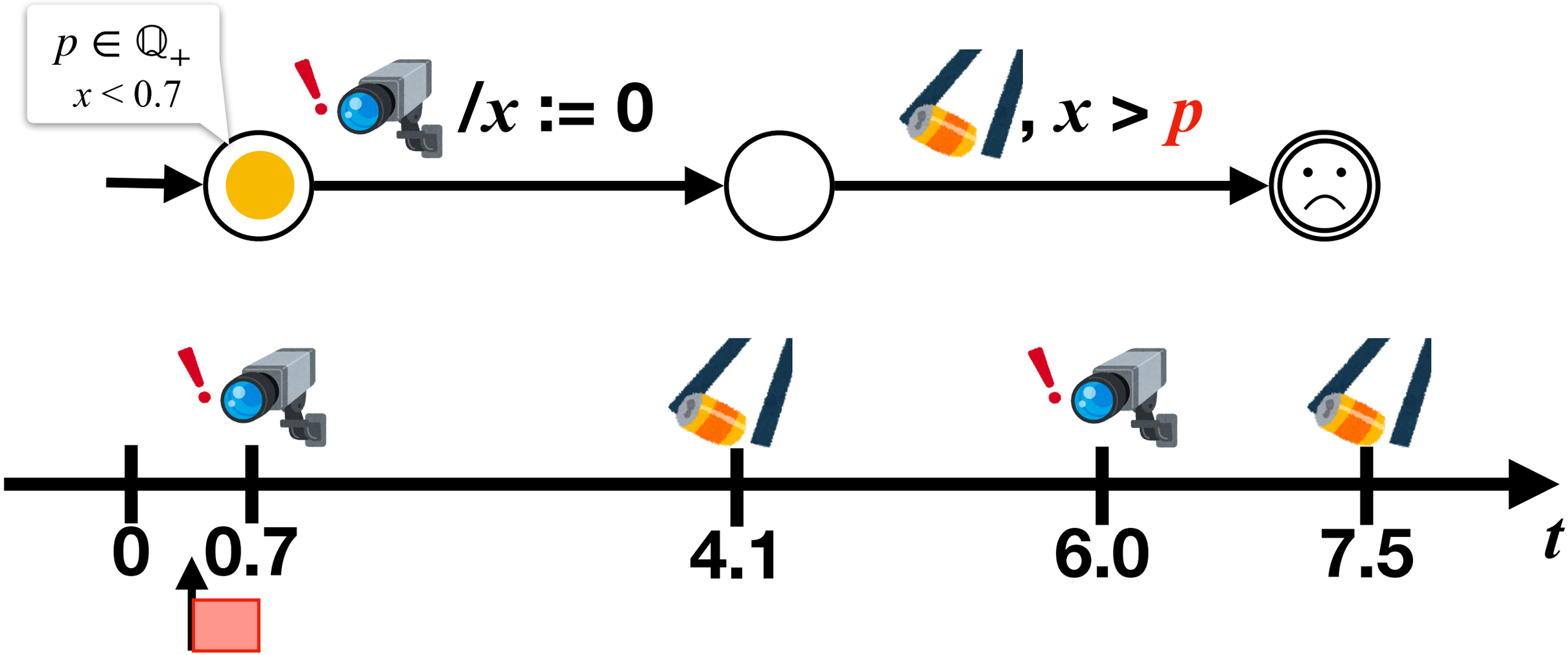
Symbolicな走査による
パラメタ付き時間パターンマッチング



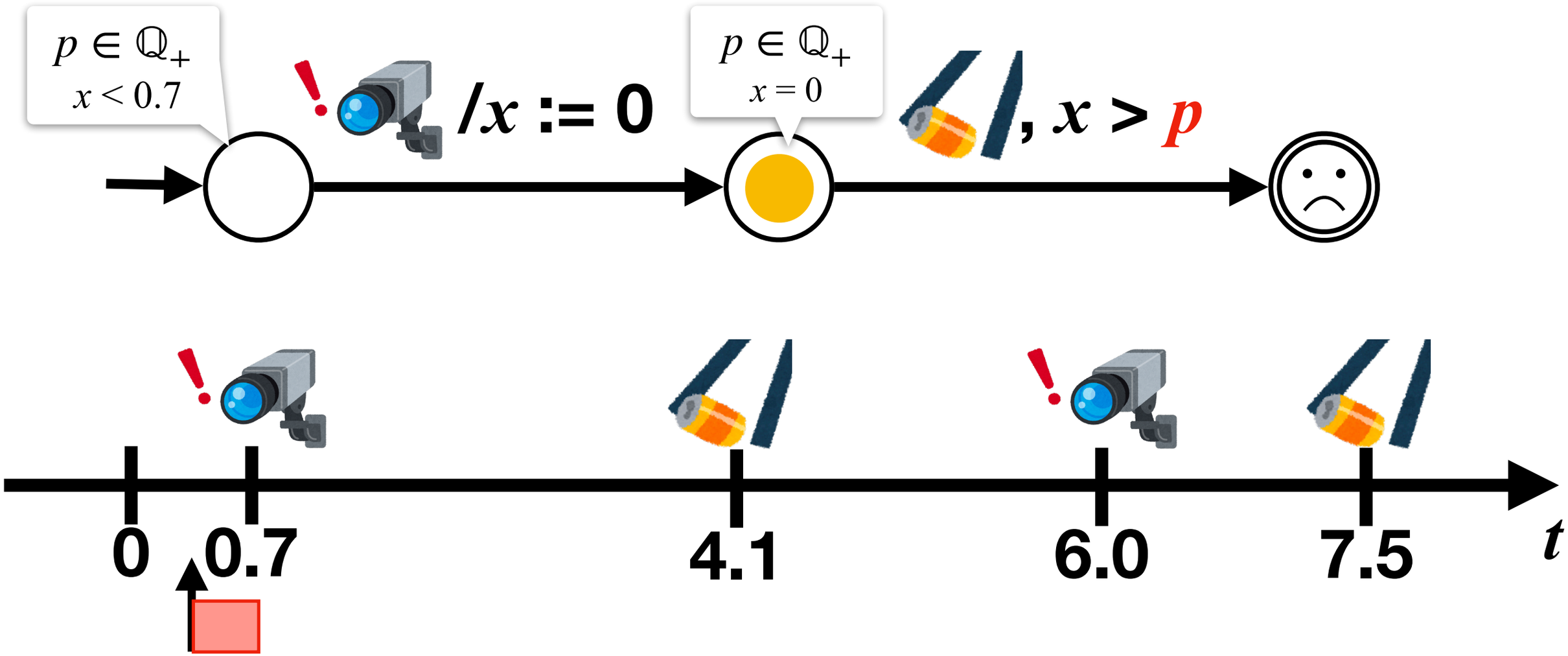
Symbolicな走査による
パラメタ付き時間パターンマッチング



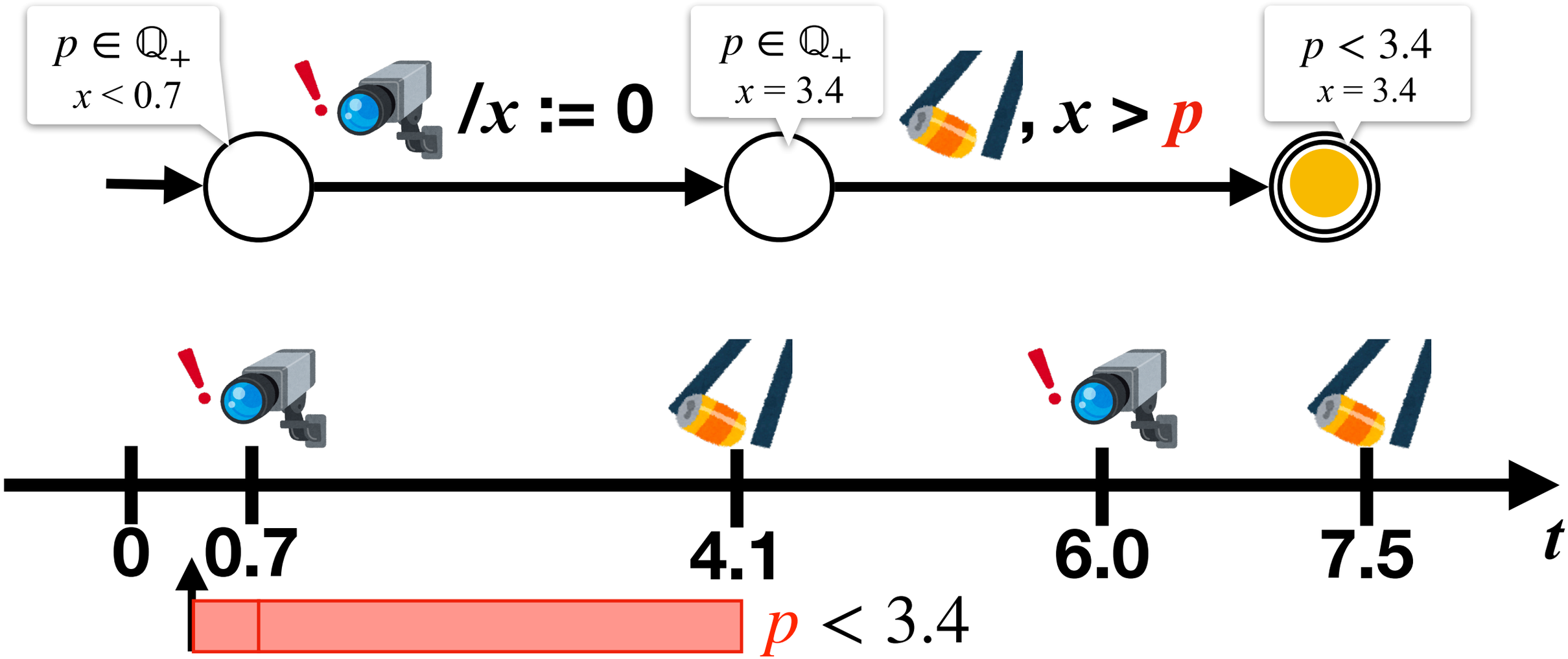
Symbolicな走査による
パラメタ付き時間パターンマッチング



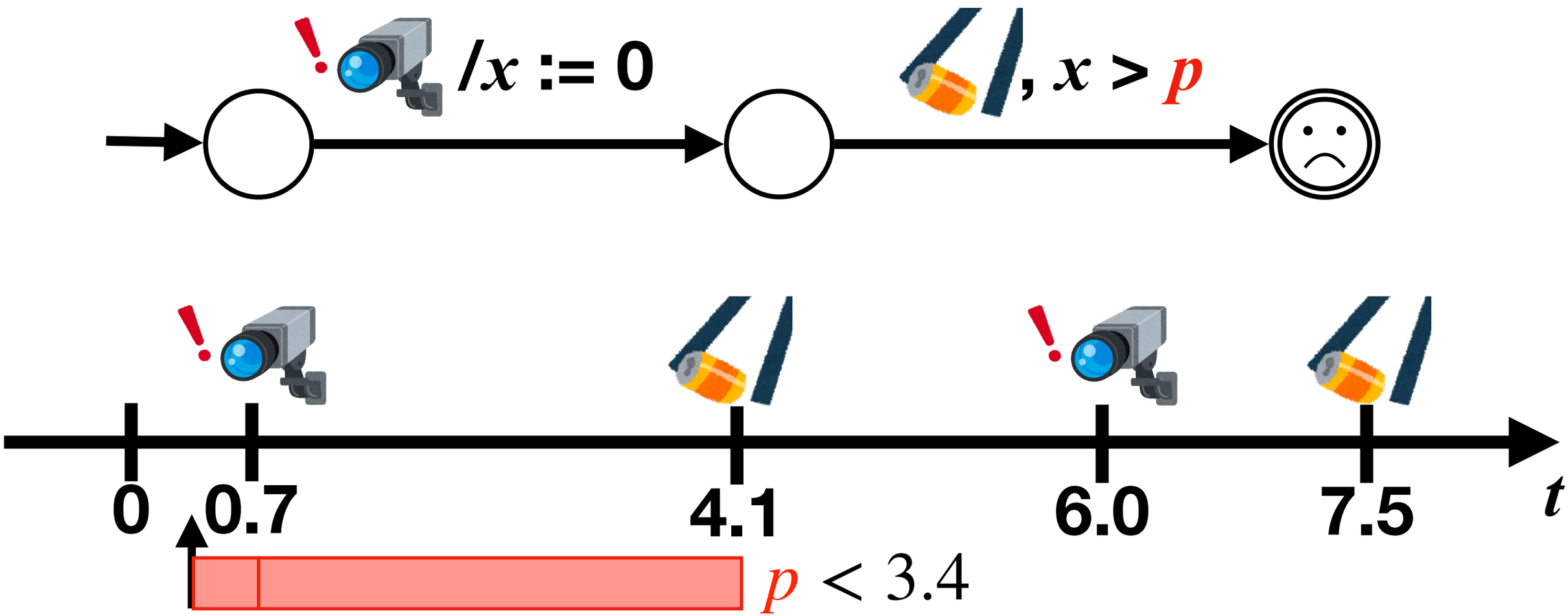
Symbolicな走査による
パラメタ付き時間パターンマッチング



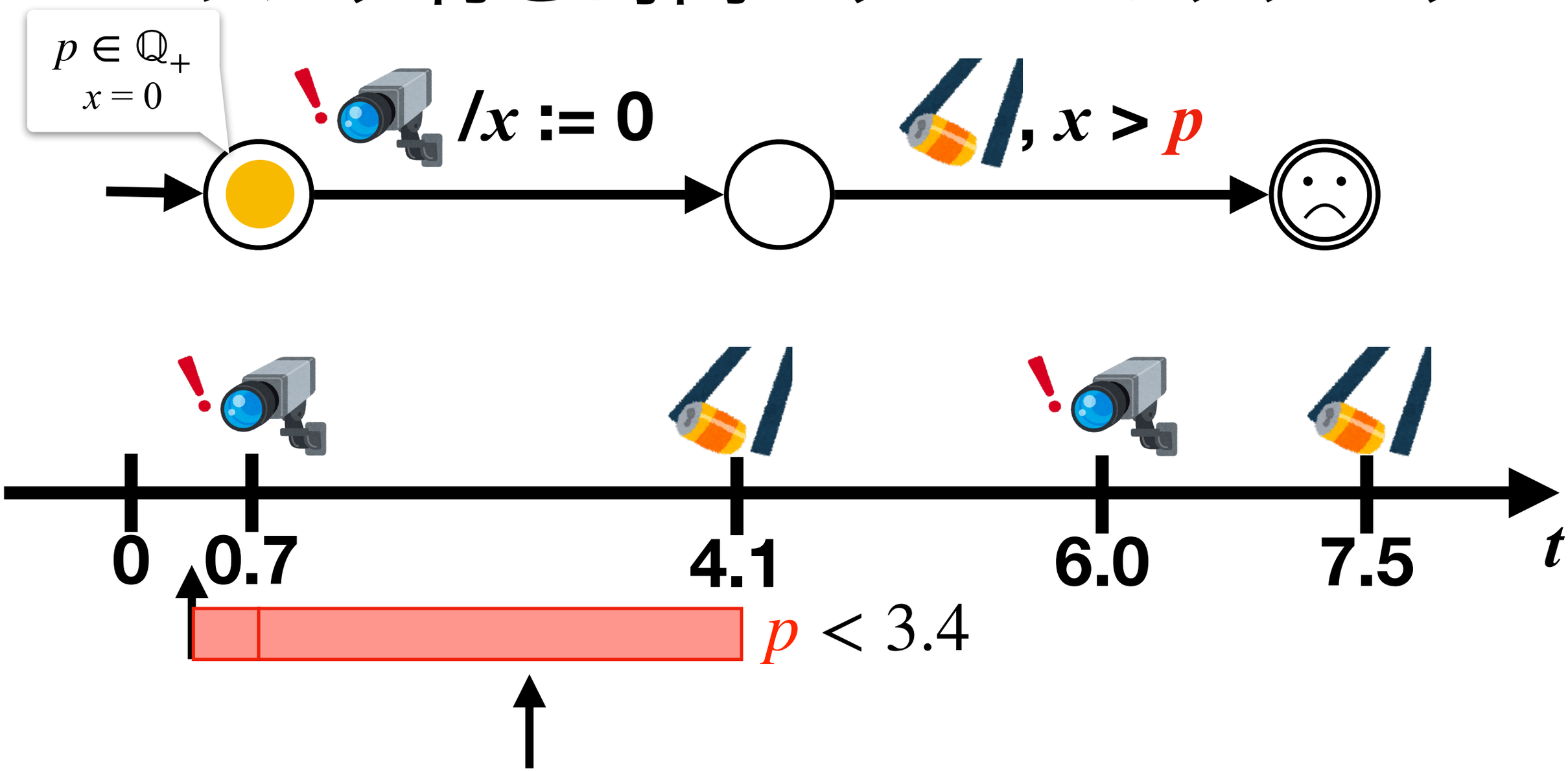
Symbolicな走査による
パラメタ付き時間パターンマッチング



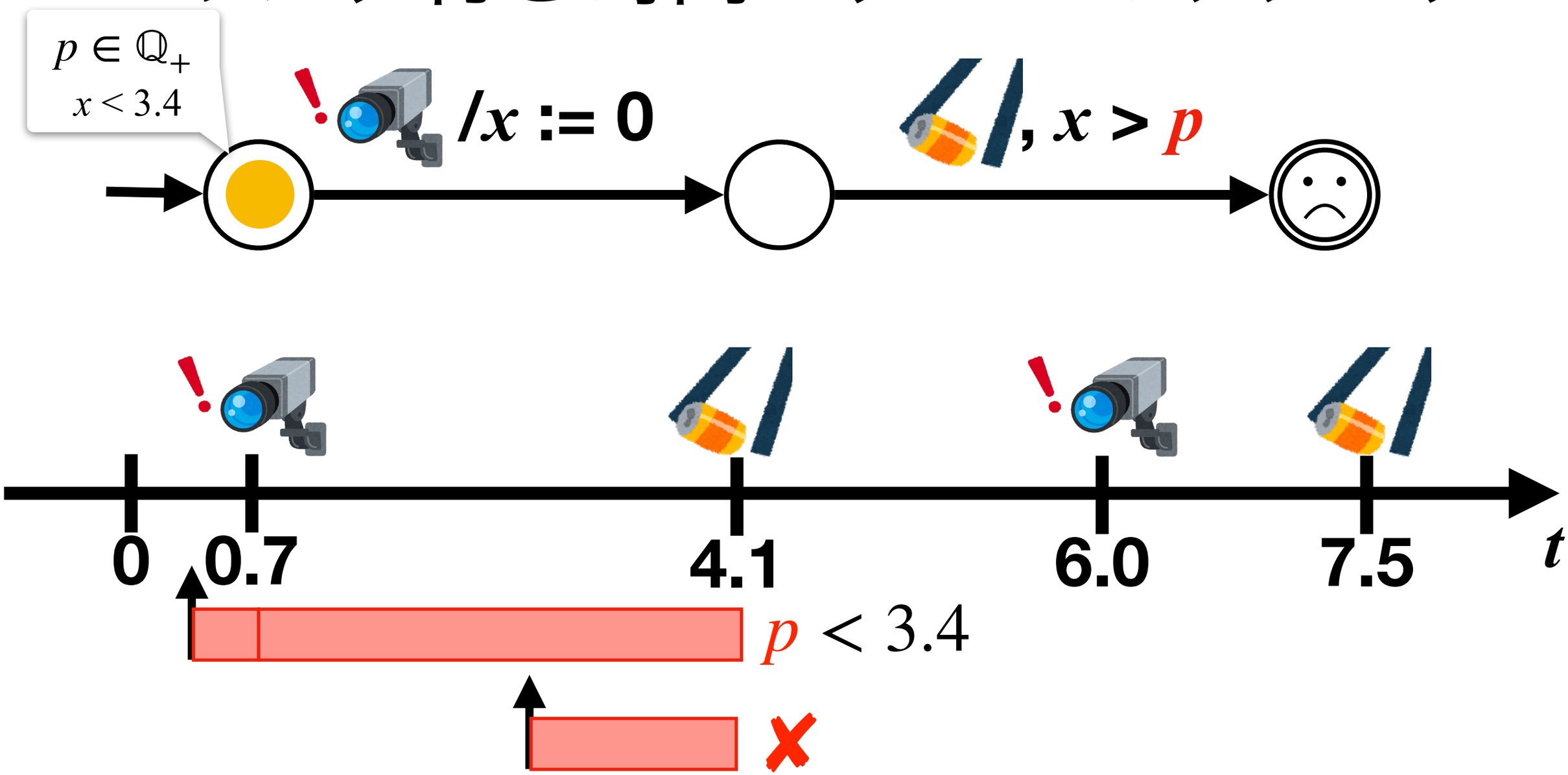
Symbolicな走査による
パラメタ付き時間パターンマッチング



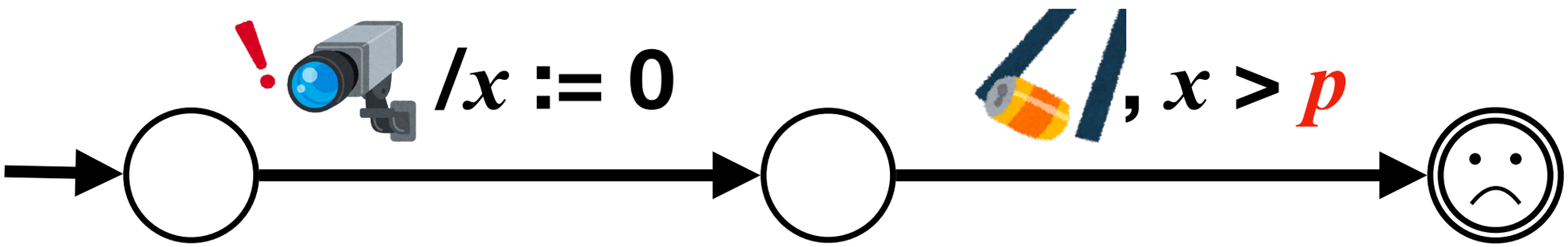
Symbolicな走査による
パラメタ付き時間パターンマッチング

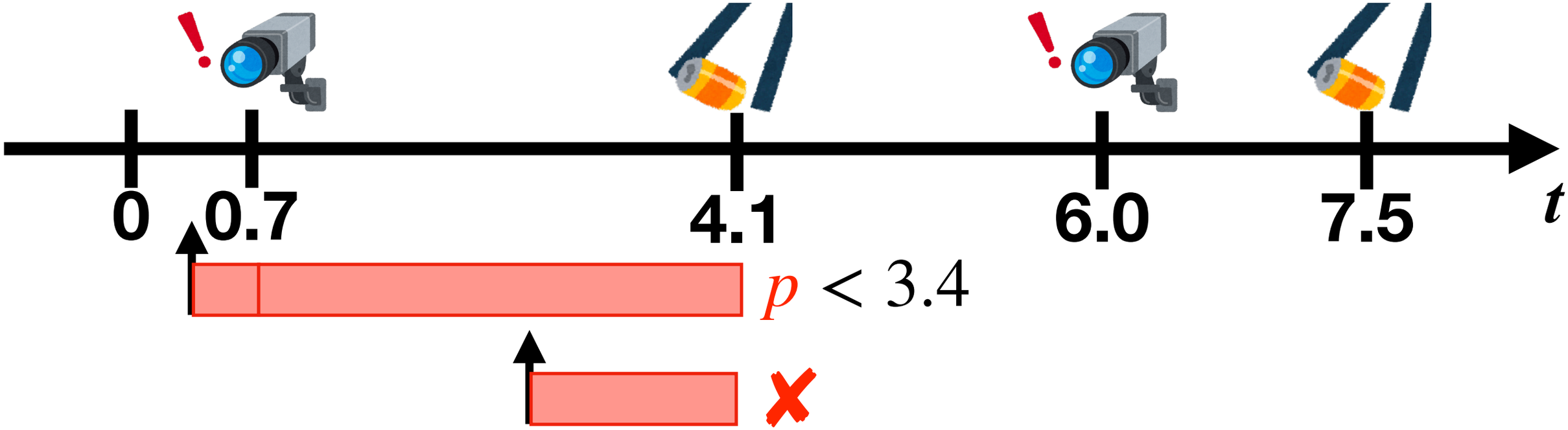


Symbolicな走査による
パラメタ付き時間パターンマッチング

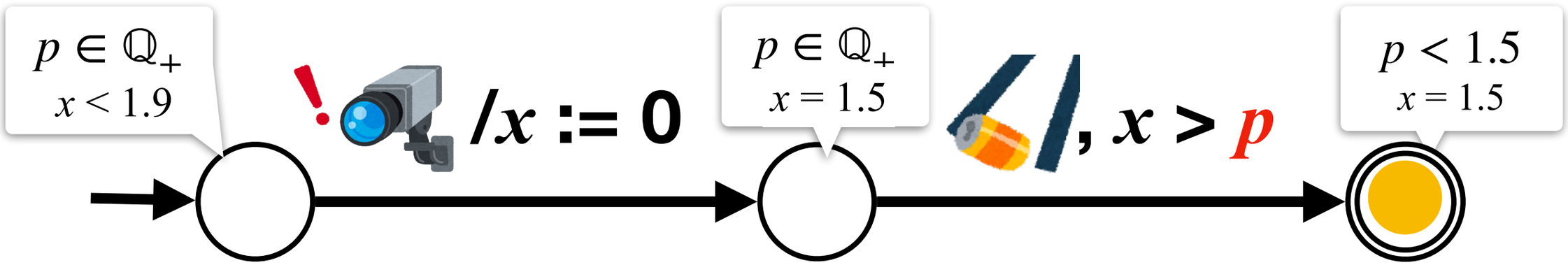


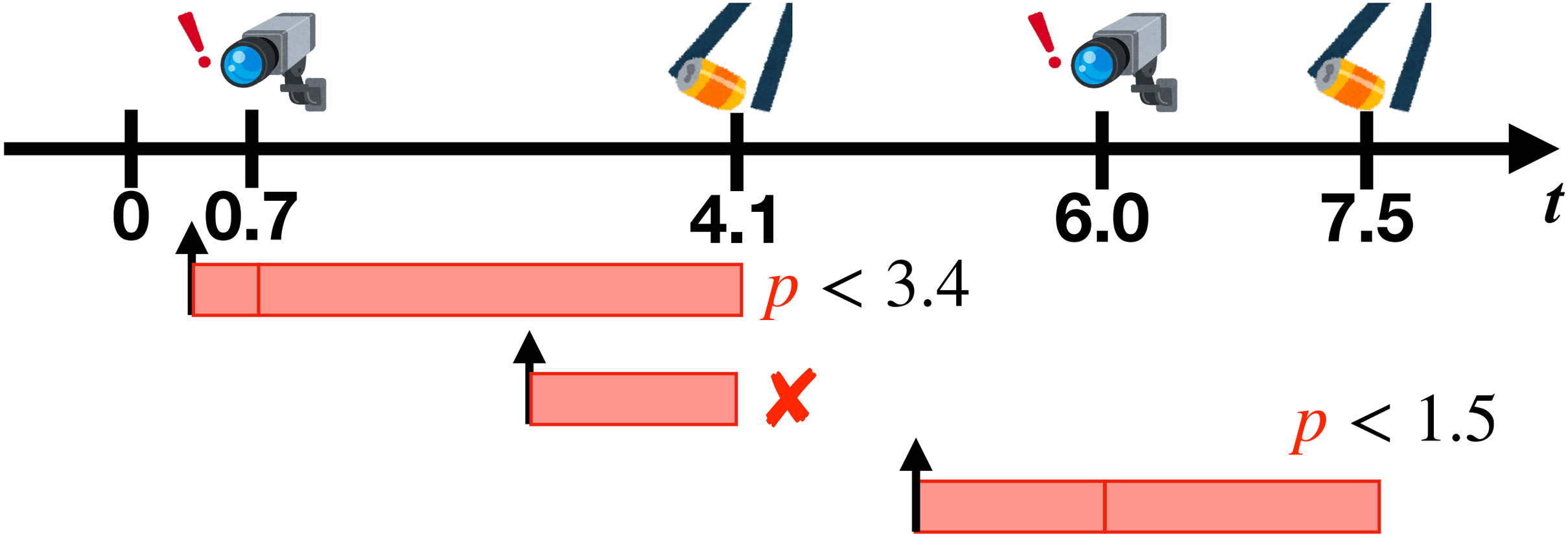
Symbolicな走査による
パラメタ付き時間パターンマッチング





Symbolicな走査による
パラメタ付き時間パターンマッチング

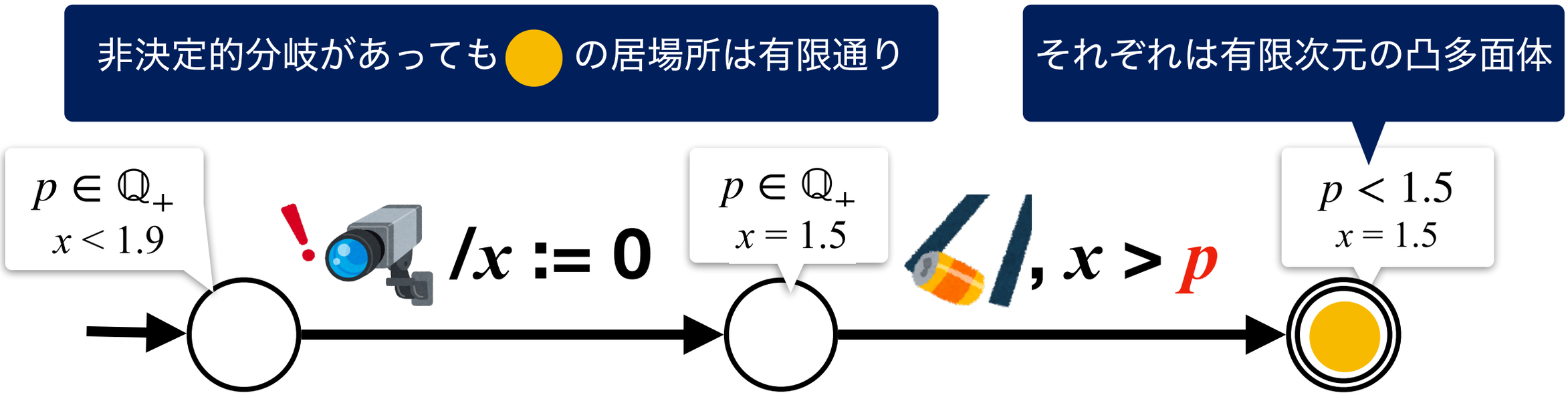




停止性・有限性

定理

パラメタ付き時間パターンマッチングは前述のアルゴリズムで有限時間で解くことができ、かつ解の集合は凸多面体の有限和で表現可能



Outline

- パラメタ付き時間パターンマッチング問題を定式化
  - パラメタを用いた仕様記述 → パラメタを生成
  - パターンにマッチする領域とパラメタを全列挙
- 二種類のアルゴリズムを提案
  - 手法1: パラメタ付き時間オートマトンのモデル検査に帰着
  - 手法2: 正規表現マッチングに類似、より直接的
- 実験的に効率性を評価 → 30 us / event 程度

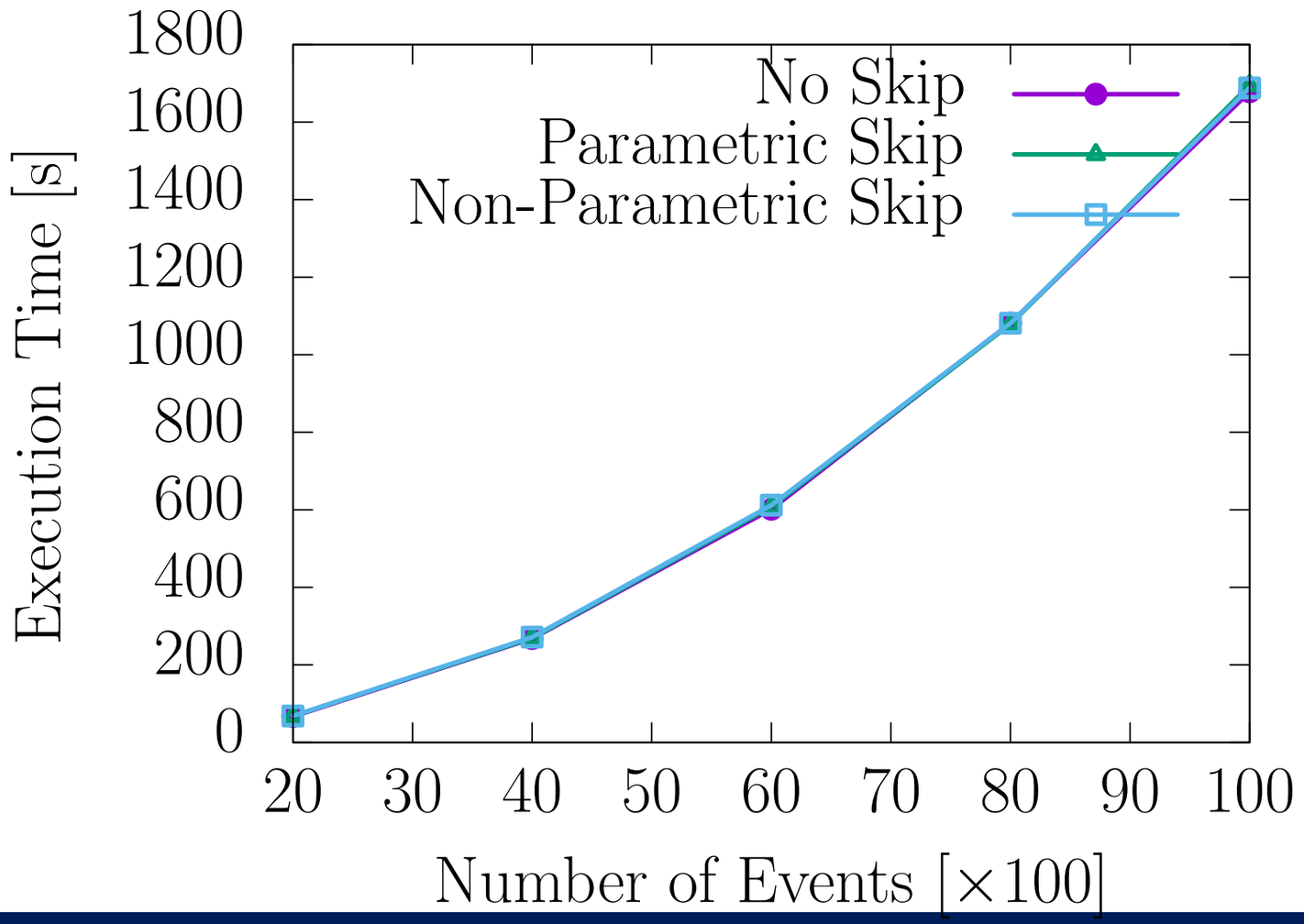
実験設定

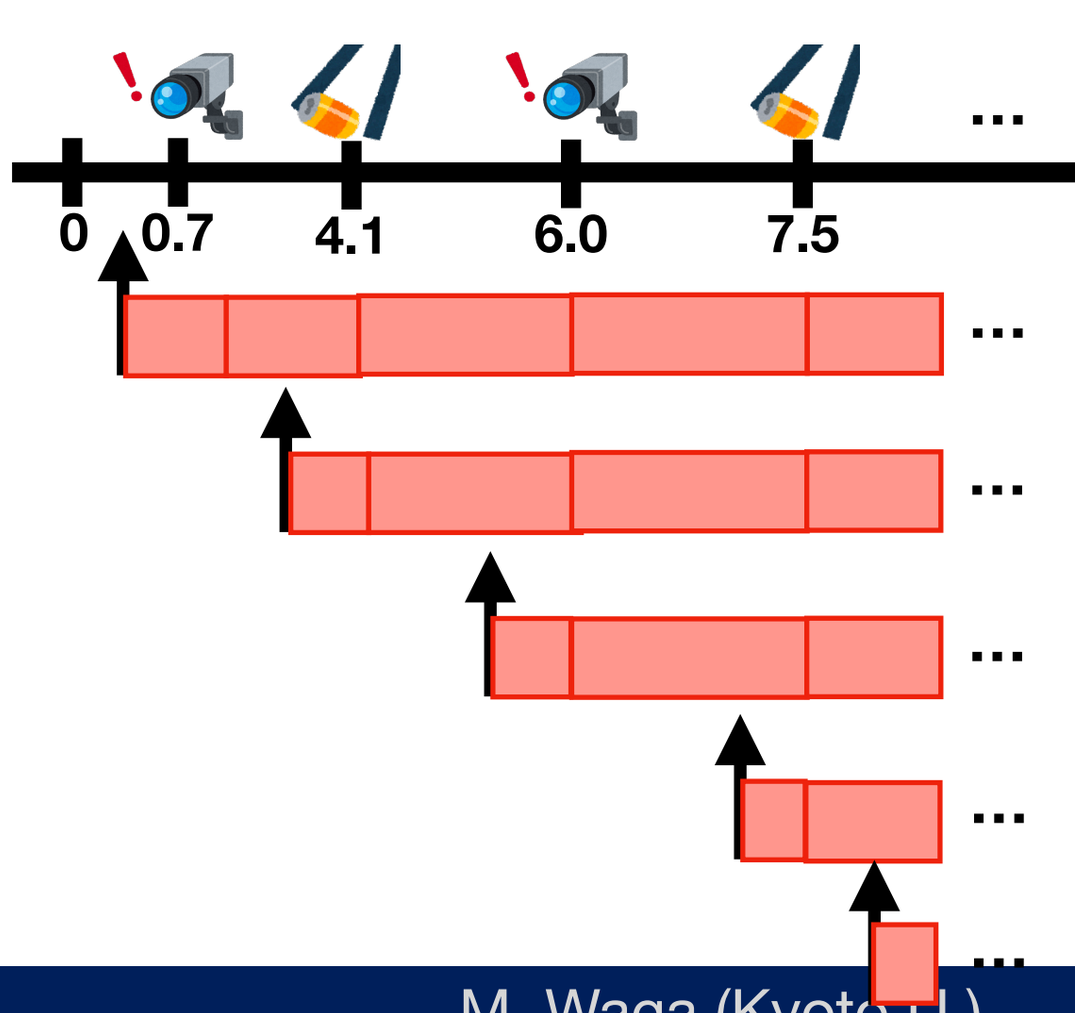
- Amazon EC2 c4.large instance
  - 2.9 GHz Intel Xeon E5-2666 v3, 2 vCPUs, 3.75 GiB RAM
- Ubuntu 18.04 LTS (64 bit)
- GCC 7.3.0 with optimization flag -O3
- マッチング長がboundedな場合とunboundedな場合でscalabilityを評価

Result 1: マッチング長が Unbounded だとScaleしない

マッチング長がunbounded

→ 組み合わせ爆発が発生しうる

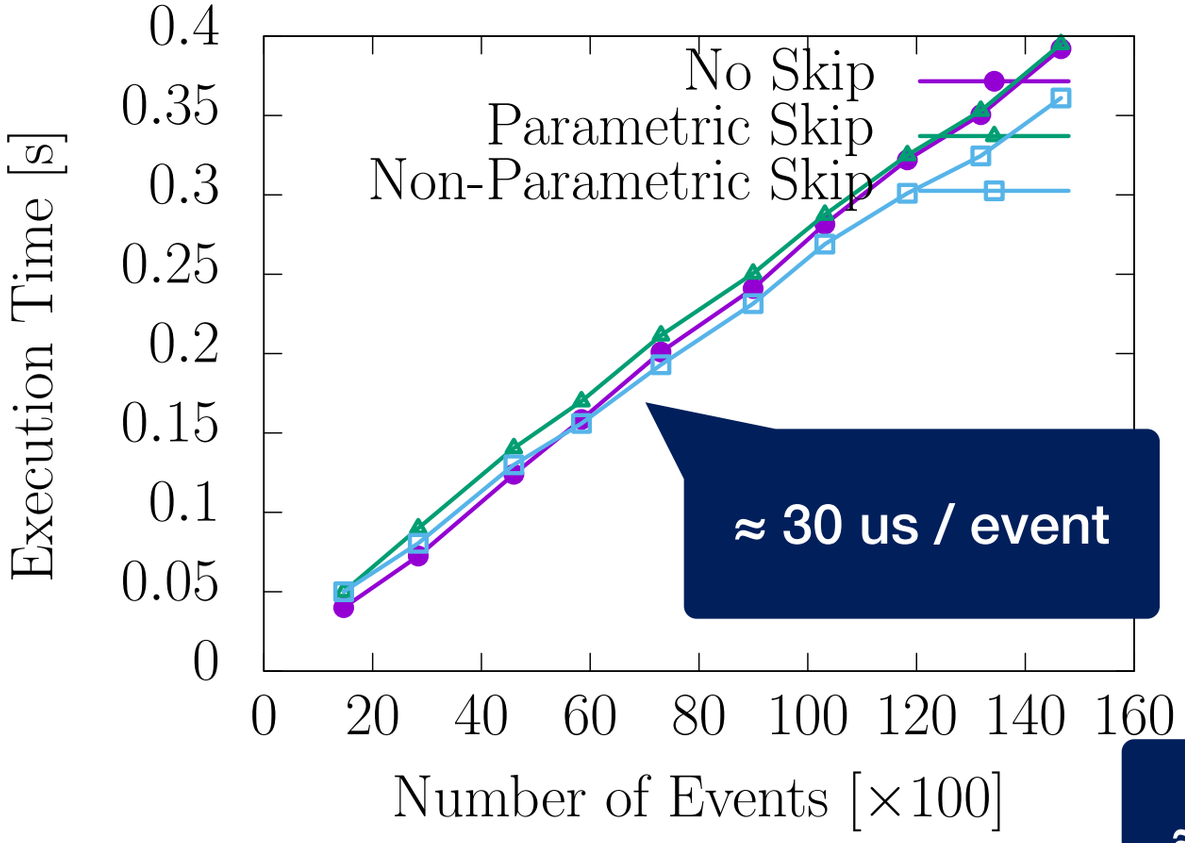




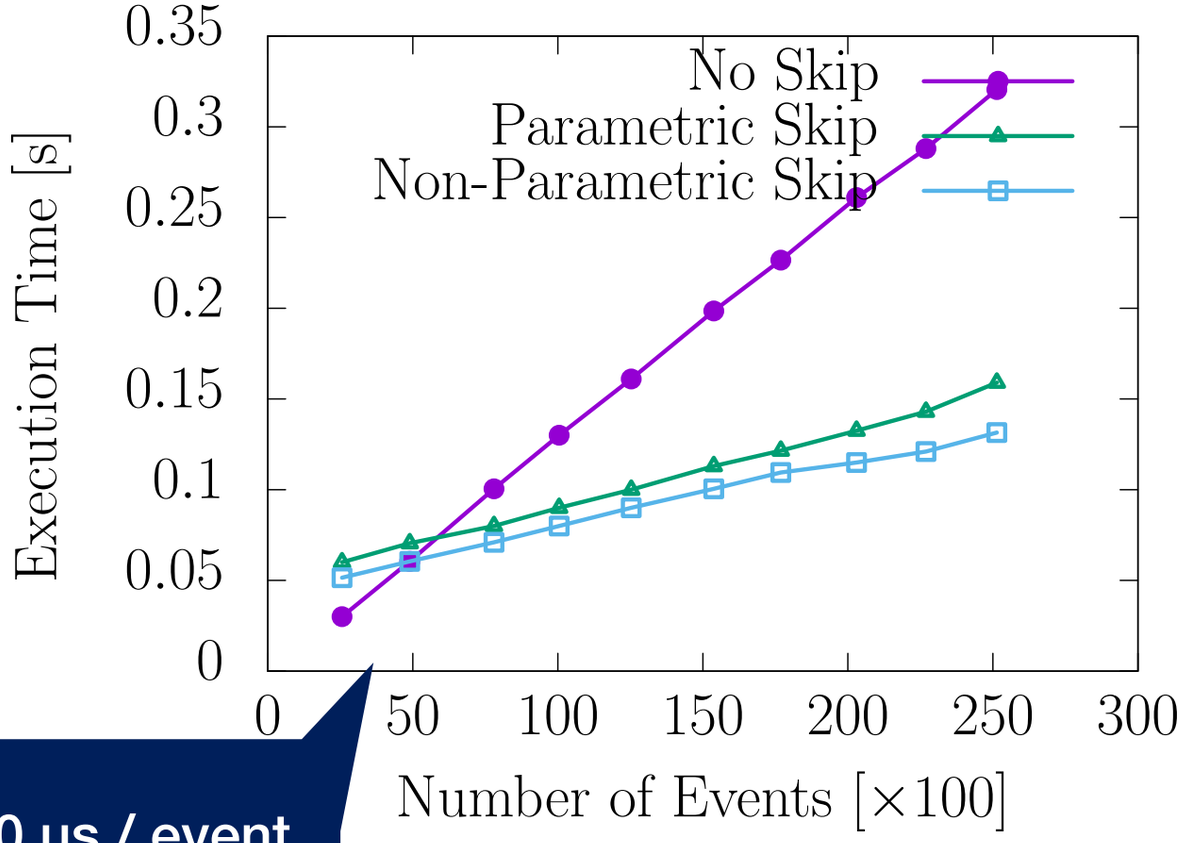
Result 2: マッチング長が Bounded ならScaleする

マッチング長がBounded

→ 入力長に対して線形時間で動作



≈ 30 us / event



≈ 10 us / event

まとめ...の前にツールの紹介

ParamMONAA:
Parametric Monitor with Automata-Based Acceleration

実験的なコマンドラインツール

GitHubからダウンロード可

https://github.com/MasWag/ParamMONAA/

ParamMONAA Demo

Log (length: 25,137)

```
E 2926.36
C 2926.6
B 2926.68
F 2926.84
E 2926.88
C 2927.08
B 2927.16
F 2927.32
E 2927.36
C 2927.56
B 2927.6
F 2927.84
E 2927.88
C 2928.04
B 2928.08
F 2928.32
E 2928.36
C 2928.56
B 2928.6
F 2928.8
E 2928.84
C 2929.04
B 2929.12
F 2929.28
E 2929.32
C 2929.56
B 2929.6
```

build — ~/C/m/build — bash — 69×16

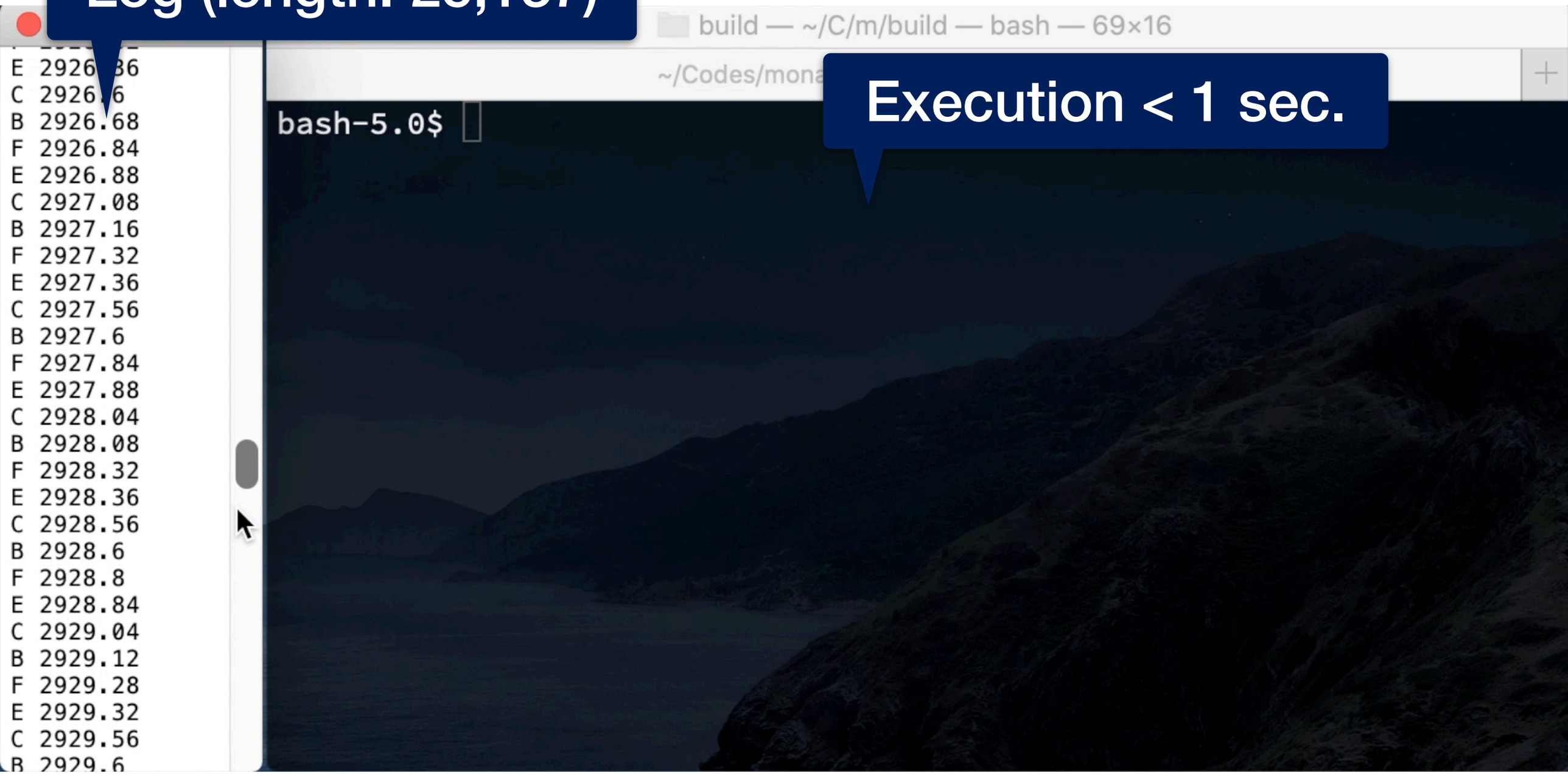
~/Codes/monaa/build — ~/C/m/build — bash

```
bash-5.0$
```

ParamMONAA Demo

Log (length: 25,137)

Execution < 1 sec.



ParamMONAA Demo

Log (length: 25,137)

Pattern in file

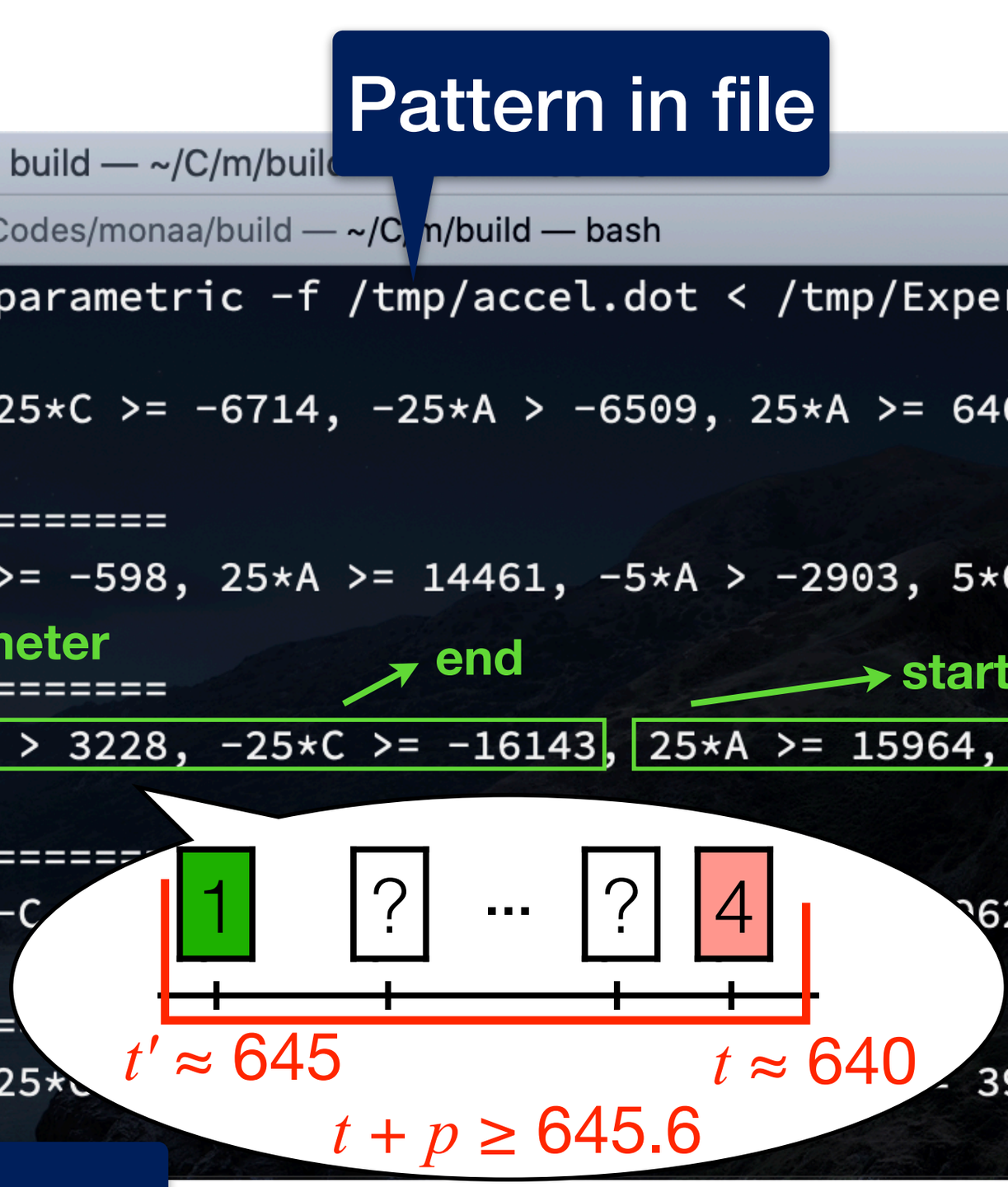
```
C 9993.6
E 9993.64
F 9994.08
E 9994.12
D 9994.24
C 9994.28
F 9994.56
D 9994.76
C 9994.8
F 9995.04
D 9995.24
C 9995.28
F 9995.56
D 9995.72
C 9995.76
F 9996.08
D 9996.2
C 9997.04
D 9997.16
C 9997.52
D 9997.68
C 9998
D 9998.16
C 9998.28
B 9998.4
A 10000
```

```
bash-5.0$ ./pmonaa -s parametric -f /tmp/accel.dot < /tmp/Experiment3
_AT-10000.tsv
25*A + 25*B >= 6682, -25*C >= -6714, -25*A > -6509, 25*A >= 6464, 25*
C > 6682
==============================
5*A + 5*B >= 2937, -C >= -598, 25*A >= 14461, -5*A > -2903, 5*C > 293
7
==============================
5*A + 5*B >= 3228, 5*C > 3228, -25*C >= -16143, 25*A >= 15964, -25*A
> -16009
==============================
25*A + 25*B >= 24166, -C
 > 24166
==========
25*A + 25*B >= 39622, 25*
5*A > -7902
```

parameter

end

start



Result (Intervals + parameter valuations)

まとめ

- パラメタ付き時間パターンマッチング問題を定式化
  - パターンにマッチする領域とパラメタを全列挙

- Symbolicな走査によるアルゴリズム
  - 有限通りの分岐 + 多面体によるsymbolicな処理
  - Skippingによる最適化

- 実験的に効率性を評価 → 30 us / event 程度

Appendix

Parametric Timed Pattern Matching

Input

- **Time-series data**
  - System ***log*** (event + timestamp)
  - e.g.,

- ***Parametric*** **Real-time spec.**
  - ***Spec.*** to be monitored
  - e.g., Duration between a pair of two withdrawals is ≤ p min.

Output

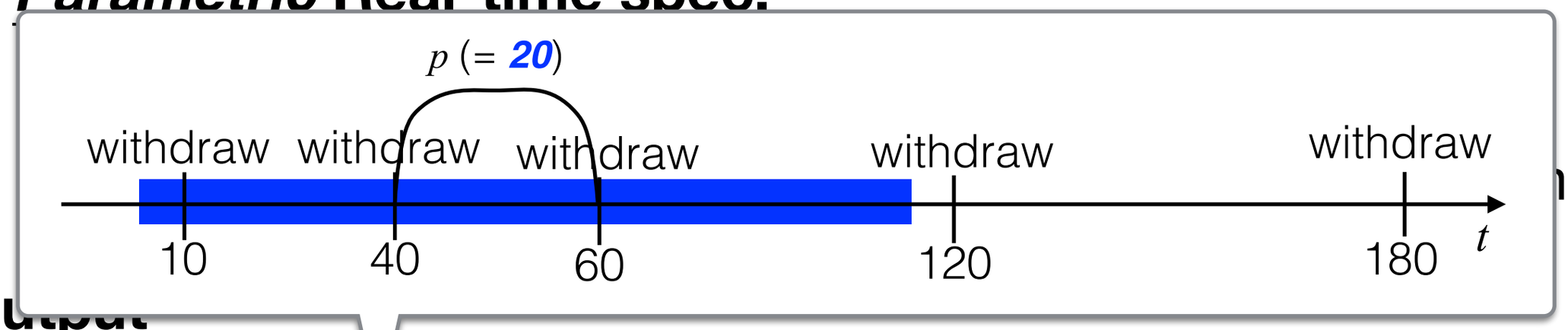
- **All** the intervals + **param. val.**, s.t. the ***log*** satisfies the ***spec.***
  - e.g., (8min-100min, $p = 20$), (50min-190min, $p = 60$), …
  - **Infinitely** many
    → **Symbolic** representation by **convex polyhedra**

Parametric Timed Pattern Matching

Input

- **Time-series data**
  - System ***log*** (event + timestamp)
  - e.g.,

  - ***Parametric* Real-time spec.**



Output

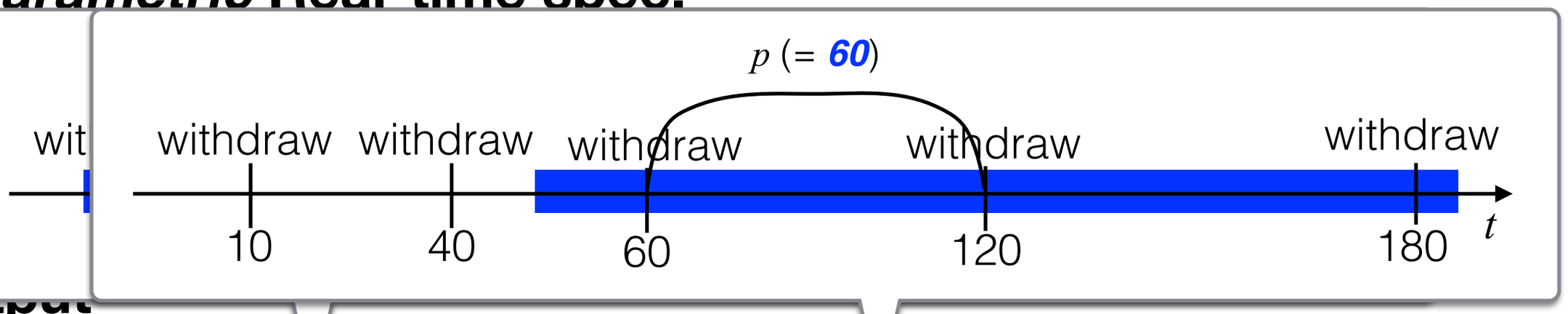
- **All** the intervals + **param. val.**, s.t. the ***log*** satisfies the ***spec.***
  - e.g., (8min-100min, $p = 20$), (50min-190min, $p = 60$), …
  - **Infinitely** many
    → **Symbolic** representation by **convex polyhedra**

Parametric Timed Pattern Matching

Input

- **Time-series data**
  - System ***log*** (event + timestamp)
  - e.g.,

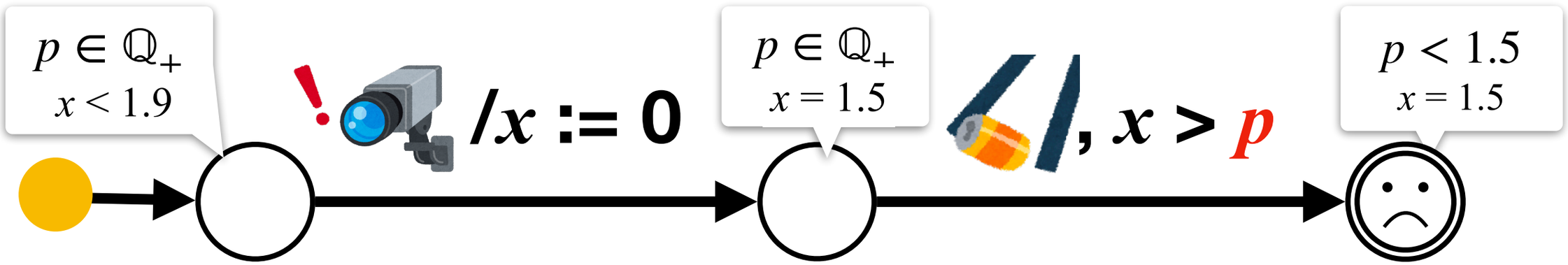
  - ***Parametric* Real-time spec.**

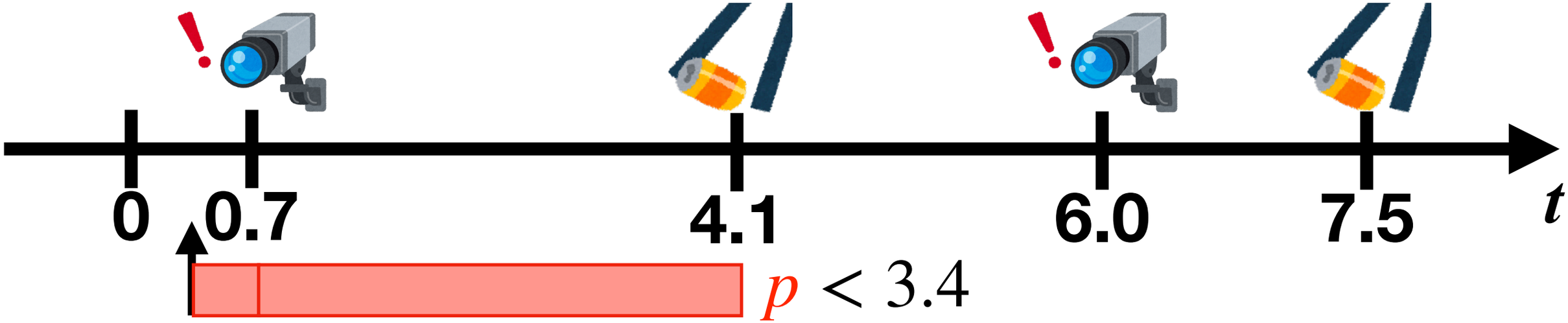


Output

- **All** the intervals + **param. val.**, s.t. the ***log*** satisfies the ***spec.***
  - e.g., (8min-100min, $p = 20$), (50min-190min, $p = 60$), …
  - **Infinitely** many
    → **Symbolic** representation by **convex polyhedra**

Symbolicな走査による
パラメタ付き時間パターンマッチング





パターンは
から始まらない
不要では?

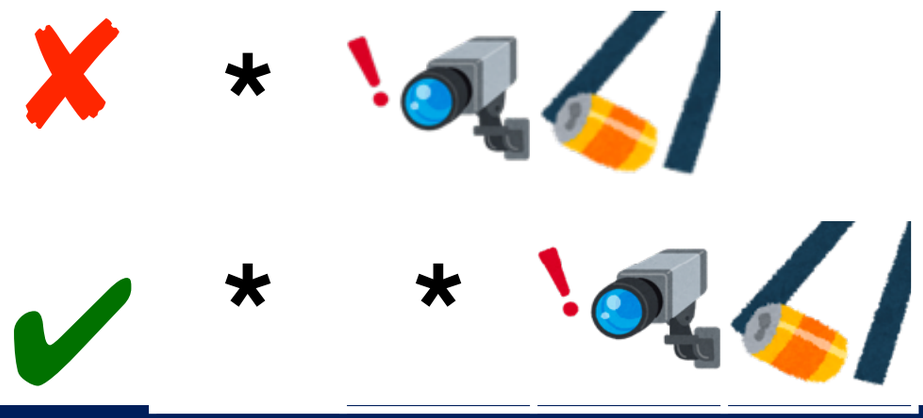
“Skipping”による最適化

効率的なマッチングのover approx.

→ 不要なら飛ばす

文字列マッチング (KMP)

[Knuth+, SIAM J. Comput. ’77]



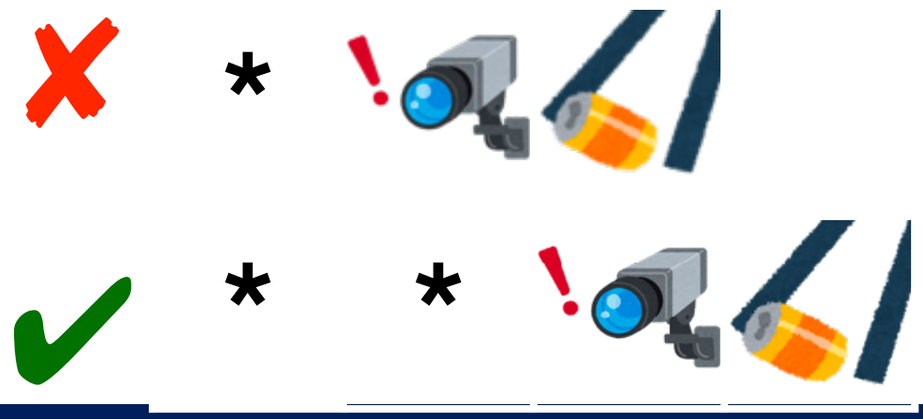
“Skipping”による最適化

効率的なマッチングのover approx.

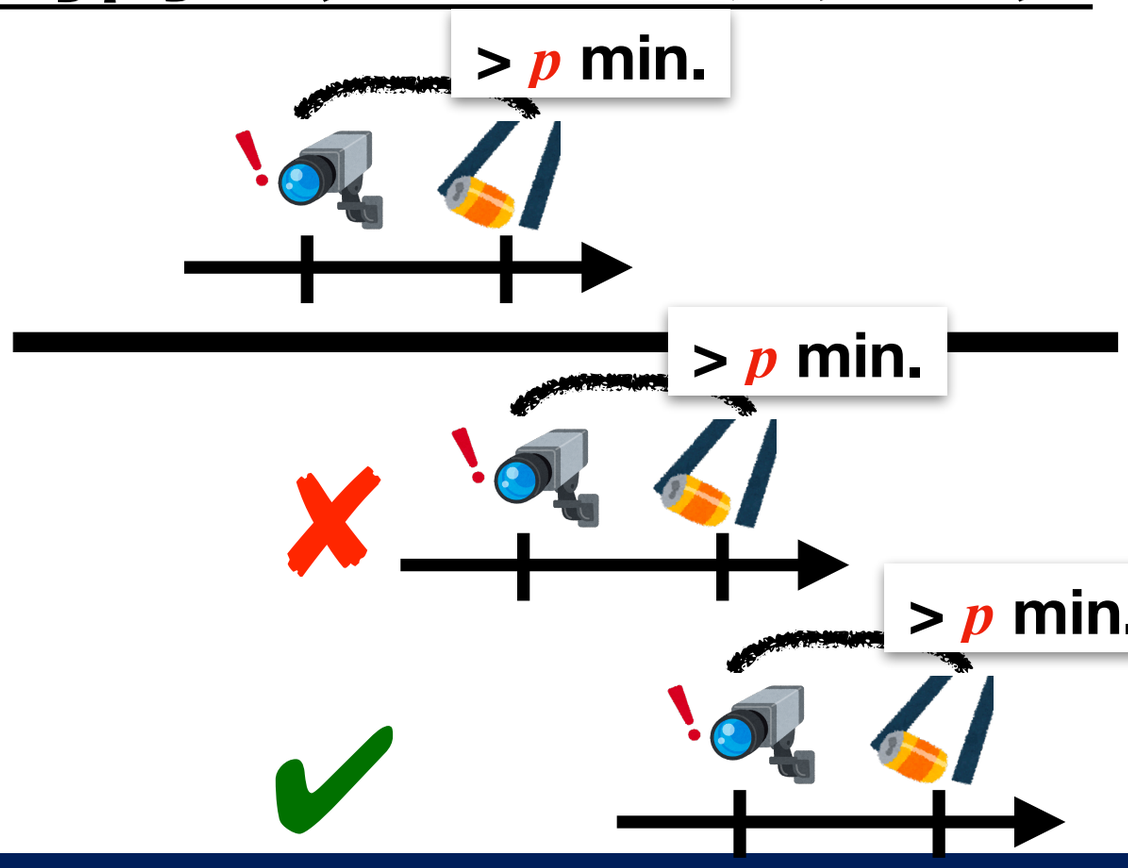
→ 不要なら飛ばす

文字列マッチング (KMP)

[Knuth+, SIAM J. Comput. ’77]



パラメタ付き 時間パターンマッチング



Detail of Skipping

Idea of Skipping from String Matching

- For each length $n \in \mathbb{N}$, check if

Over-approx. of the read timed word $\cap$ n-shift + accepted timed words $= \emptyset$

  before the matching trials.
- Empty $\Rightarrow$ no need to try n-shift
- Over-approx. using location $l \in L$ (finite)
  - In PTA, we also use param. val. v: $\mathbb{P} \to \mathbb{R}$
  - Infinite but represented by convex polyhedra

Skipping from Non-Parametric to Parametric

Skipping for TA [Waga+, FORMATS'17]

For each $l \in L$, $n \in \mathbb{N}$, if $\mathcal{L}(\mathcal{A}_l) \cdot \mathcal{T}(\Sigma) \cap \mathcal{T}^n(\Sigma) \cdot \mathcal{L}(\mathcal{A}) = \emptyset$

- $\mathcal{L}(\mathcal{A}_l) \cdot \mathcal{T}(\Sigma)$: Over-approx. of the read timed word
- $\mathcal{T}^n(\Sigma) \cdot \mathcal{L}(\mathcal{A})$: n-shift + accepted timed words

---

(Parametric) Skipping for PTA

For each $l \in L$, $v\colon \mathbb{P} \to \mathbb{R}$ $n \in \mathbb{N}$, if there is $v'\colon \mathbb{P} \to \mathbb{R}$ s.t.

$$\mathcal{L}(v(\mathcal{A}_l)) \cdot \mathcal{T}(\Sigma) \cap \mathcal{T}^n(\Sigma) \cdot \mathcal{L}(v'(\mathcal{A})) = \emptyset$$

- $\mathcal{L}(v(\mathcal{A}_l)) \cdot \mathcal{T}(\Sigma)$: Over-approx. of the read timed word for the **given** param. val. v
- $\mathcal{T}^n(\Sigma) \cdot \mathcal{L}(v'(\mathcal{A}))$: n-shift + accepted timed words for some param. val. v'

Skipping from Non-Parametric to Parametric

Skipping for TA [Waga+, FORMATS'17]

For each $l \in L$, $n \in \mathbb{N}$, if $\mathcal{L}(\mathcal{A}_l) \cdot \mathcal{T}(\Sigma) \cap \mathcal{T}^n(\Sigma) \cdot \mathcal{L}(\mathcal{A}) = \emptyset$

- $\mathcal{L}(\mathcal{A}_l) \cdot \mathcal{T}(\Sigma)$: Over-approx. of the read timed word
- $\mathcal{T}^n(\Sigma) \cdot \mathcal{L}(\mathcal{A})$: n-shift + accepted timed words

---

(Parametric) Skipping for PTA

For each $l \in L$, $v\colon \mathbb{P} \to \mathbb{R}$ $n \in \mathbb{N}$, if there is $v'\colon \mathbb{P} \to \mathbb{R}$ s.t.

$\mathcal{L}(v(\mathcal{A}_l)) \cdot \mathcal{T}(\Sigma) \cap \mathcal{T}^n(\Sigma) \cdot \mathcal{L}(v'(\mathcal{A})) = \emptyset$

- Runtime Overhead!!
- $\mathcal{L}(v(\mathcal{A}_l)) \cdot \mathcal{T}(\Sigma)$: Over-approx. of the read timed word for the **given** param. val. v
- $\mathcal{T}^n(\Sigma) \cdot \mathcal{L}(v'(\mathcal{A}))$: n-shift + accepted timed words for some param. val. v'

Skipping with Less Overhead

(Parametric) Skipping for PTA

For each $l \in L$, v: $\mathbb{P} \to \mathbb{R}$ $n \in \mathbb{N}$, if there is v': $\mathbb{P} \to \mathbb{R}$ s.t.

$$\mathcal{L}(v(\mathcal{A}_l)) \cdot \mathcal{T}(\Sigma) \cap \mathcal{T}^n(\Sigma) \cdot \mathcal{L}(v'(\mathcal{A})) = \emptyset$$

More overhead
better approx.

Over-approx. of the read timed word for the **given** param. val. v

n-shift + accepted timed words for some param. val. v'

Trade off!!

(Non-Parametric) Skipping for PTA

For each $l \in L$, $n \in \mathbb{N}$, if there is v, v': $\mathbb{P} \to \mathbb{R}$ s.t.

$$\mathcal{L}(v(\mathcal{A}_l)) \cdot \mathcal{T}(\Sigma) \cap \mathcal{T}^n(\Sigma) \cdot \mathcal{L}(v'(\mathcal{A})) = \emptyset$$

Less overhead
worse approx.

Over-approx. of the read timed word for **some** param. val. v

n-shift + accepted timed words for some param. val. v'

Other Experiments

Environment of Experiment

- Amazon EC2 c4.large instance
  - 2.9 GHz Intel Xeon E5-2666 v3, 2 vCPUs, 3.75 GiB RAM
- Ubuntu 18.04 LTS (64 bit)
- GCC 7.3.0 with optimization flag -O3
- Used 4 benchmarks
  - **Automotive**: Accel, Gear
  - **Toy**: Blowup, OnlyTiming

Performance for extreme inputs

Comparison with IMITATOR

Table 2: Execution time for GEAR [s]

$\|w\|$	No Skip	Non-Param. Skip	Param. Skip	IMITATOR
1467	0.04	0.05	0.05	1.781
2837	0.0725	0.0805	0.09	3.319
4595	0.124	0.13	0.1405	5.512
5839	0.1585	0.156	0.17	7.132
7301	0.201	0.193	0.2115	8.909
9995	0.241	0.2315	0.2505	10.768
11315	0.2815	0.269	0.2875	12.778
12831	0.322	0.301	0.325	14.724
13183	[illegible]	0.3245	0.353	16.453
14657	0.392	0.361	0.395	18.319

60x faster

Table 3: Execution time for ACCEL [s]

$\|w\|$	No Skip	Non-Param. Skip	Param. Skip	IMITATOR
2559	0.03	0.0515	0.06	2.332
4894	0.0605	0.0605	0.0705	4.663
7799	0.1005	0.071	0.08	7.532
10045	0.13	0.08	0.09	9.731
12531	0.161	0.09	0.1	12.503
15375	0.1985	0.1005	0.113	15.583
17688	0.2265	0.1095	0.1215	17.754
20299	0.261	0.115	0.1325	21.[illegible]
22691	0.288	0.121	0.[illegible]	23.044
25137	0.3205	0.1315	0.159	25.815

200x faster

Table 4: Execution time for BLOWUP [s]

$\|w\|$	No Skip	Non-Param. Skip	Param. Skip	IMITATOR
2000	66.75	68.0125	67.9735	OutOfMemory
4000	267.795	271.642	269.084	OutOfMemory
6000	601.335	611.782	607.58	OutOfMemory
8000	1081.42	1081.25	1079	OutOfMemory
10000	1678.15	1688.22	1694.53	OutOfMemory

out of memory!!

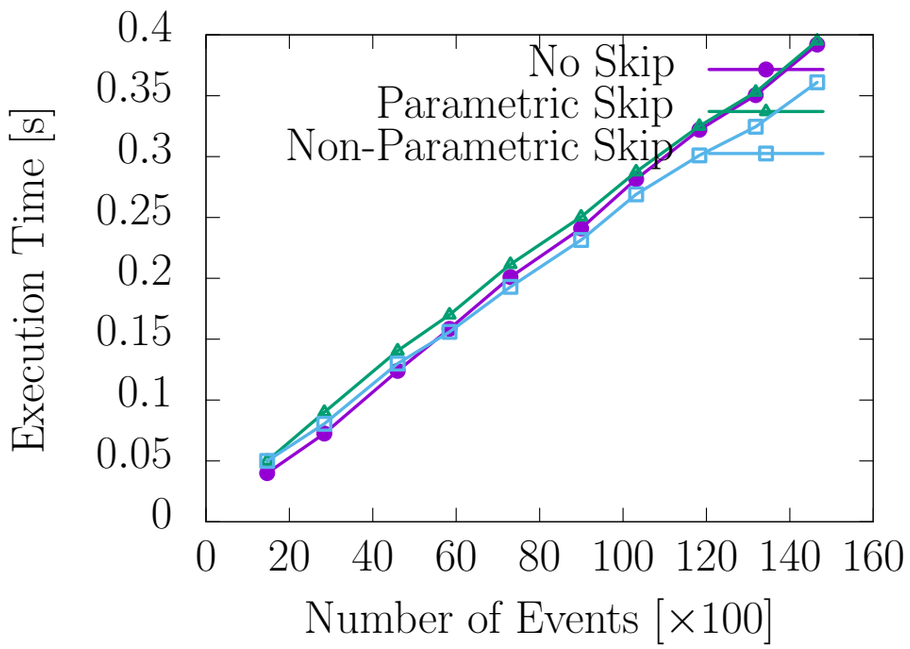
Table 5: Execution time for ONLYTIMING [s]

$\|w\|$	No Skip	Non-Param. Skip	Param. Skip	IMITATOR
1000	0.0995	0.1305	0.11	1.690
2000	0.191	0.23	0.191	3.518
3000	0.2905	0.3265	0.273	5.499
4000	0.3905	0.426	0.3525	7.396
5000	0.488	0.5225	0.4325	9.123
6000	0.588	0.6235	0.517	11.005

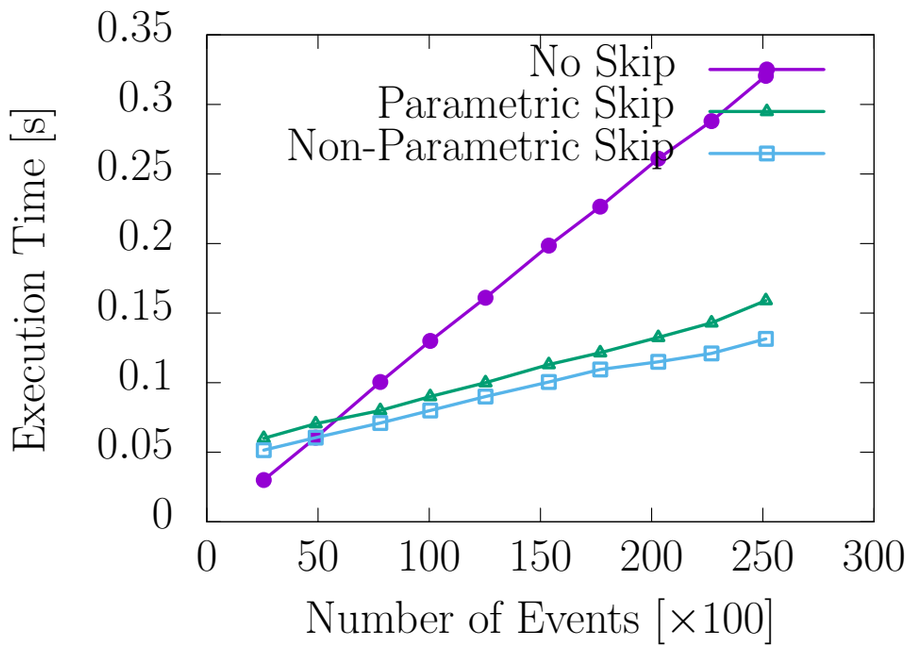
20x faster

Comparison among ours (Accel & Gear)

Accel



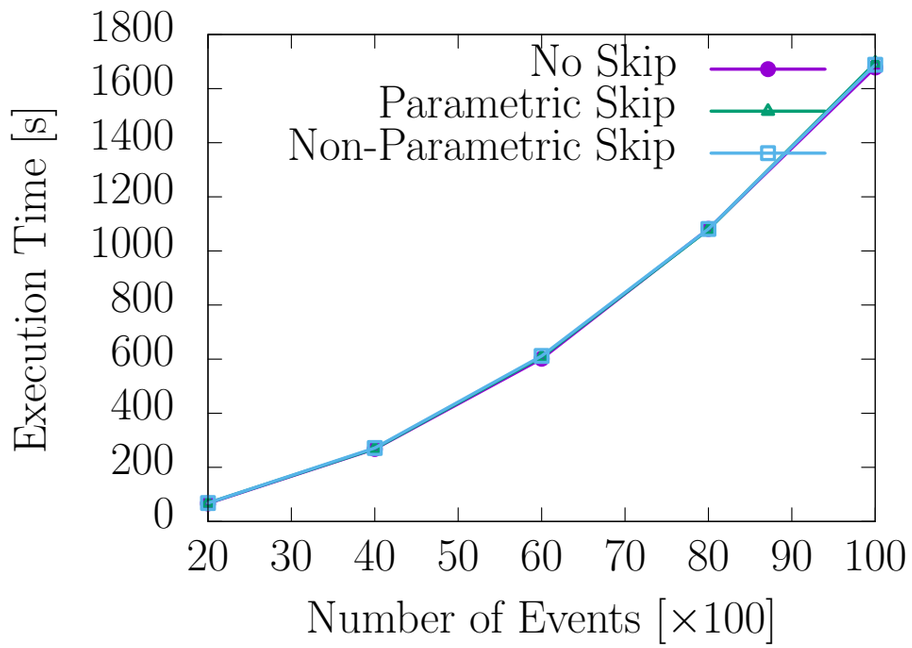
Gear



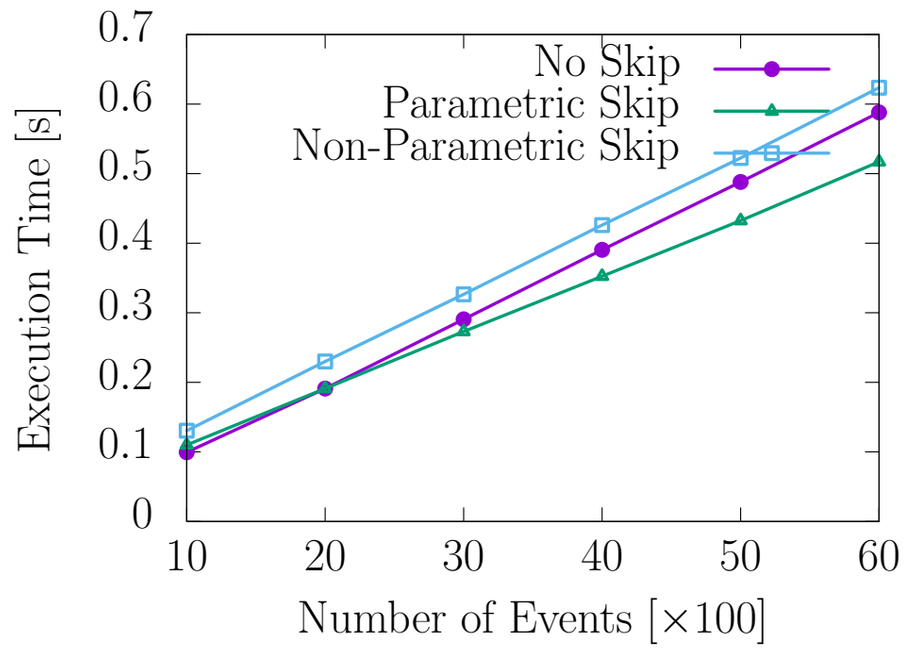
- No Skip has the steepest slope ⇒ worst scalability
- Parametric Skip is slower than Non-Parametric Skip due to the overhead

Comparison among ours (Blowup & OnlyTiming)

Blowup



OnlyTiming



- Blowup: Skipping does not help much
  - Exponential blowup vs. constant speed up by skipping
- OnlyTiming: No Skip has the steepest slope ⇒ worst scalability

  Parametric Skip is the fastest due to the better over-approx.