A Real-Time, Flexible Logging Infrastructure for MonPoly

Bachelor's Thesis

Jonas Degelo

ETH Zürich

February 24, 2023

Agenda

MonPoly

- Runtime Monitor
- Metric First Order Temporal Logic (MFOTL)
- MFOTL temporal operators (left) and syntactic sugar (right)
  - $\bigcirc_I$ ("Next")
  - $\bullet_I$ ("Previous")
  - $\mathcal{U}_I$ ("Until")
  - $\mathcal{S}_I$ ("Since")
  - $\Box_I$ ("Always")
  - $\blacksquare_I$ ("Historically")
  - $\Diamond_I$ ("Eventually")
  - $\blacklozenge_I$ ("Once")

  The interval I restricts the time points relevant to an operator.
- Time Point vs. Time Stamp
  - **Time Point**: Time stamped collection of events (predicates), often referred to by an integer index
  - **Time Stamp**: Specifies a moment in time, e.g. seconds since Unix epoch

Online and Offline Monitoring

- **Online Monitoring** is done while the system is running
- **Offline Monitoring** is done on a log after a system has completed
- We focus on the Online Monitoring problem

Time-Series Databases

- A time-series database is optimized for the insertion and retrieval of temporal data.
- QuestDB
  - Largely compatible with PostgreSQL
  - Adds a `TIMESTAMP` datatype in addition to the usual types `INT`, `STRING`, etc.
- Dedicated time stamp column (analogous to an index column)

Monitor vs. Database Approach

Monitor (MonPoly)

- Good when:
  - Data changes often
  - Policy changes infrequently
- Bad when:
  - Policy changes frequently

Database (QuestDB)

- Good when:
  - Data stays almost constant
  - Varying, frequent queries
- Bad when:
  - Data changes rapidly

Motivation

- Exploit the combination of monitoring and time-series databases
  - Allows us to offer a policy change method
- More accessible interface
  - Web services
  - Distributed systems
- Increase fault tolerance

Signature to Database Schema

schema.sql

```
CREATE TABLE P(x1 INT, x2 STRING,
               tp INT,
               ts TIMESTAMP)
               timestamp(ts);
CREATE TABLE Q(x1 INT, x2 INT,
               tp INT,
               ts TIMESTAMP)
               timestamp(ts);
CREATE TABLE R(x1 INT,x2 STRING,
               x3 STRING,
               tp INT,
               ts TIMESTAMP)
               timestamp(ts);
CREATE TABLE time_points(tp INT,
                         ts TIMESTAMP)
                         timestamp(ts);
```

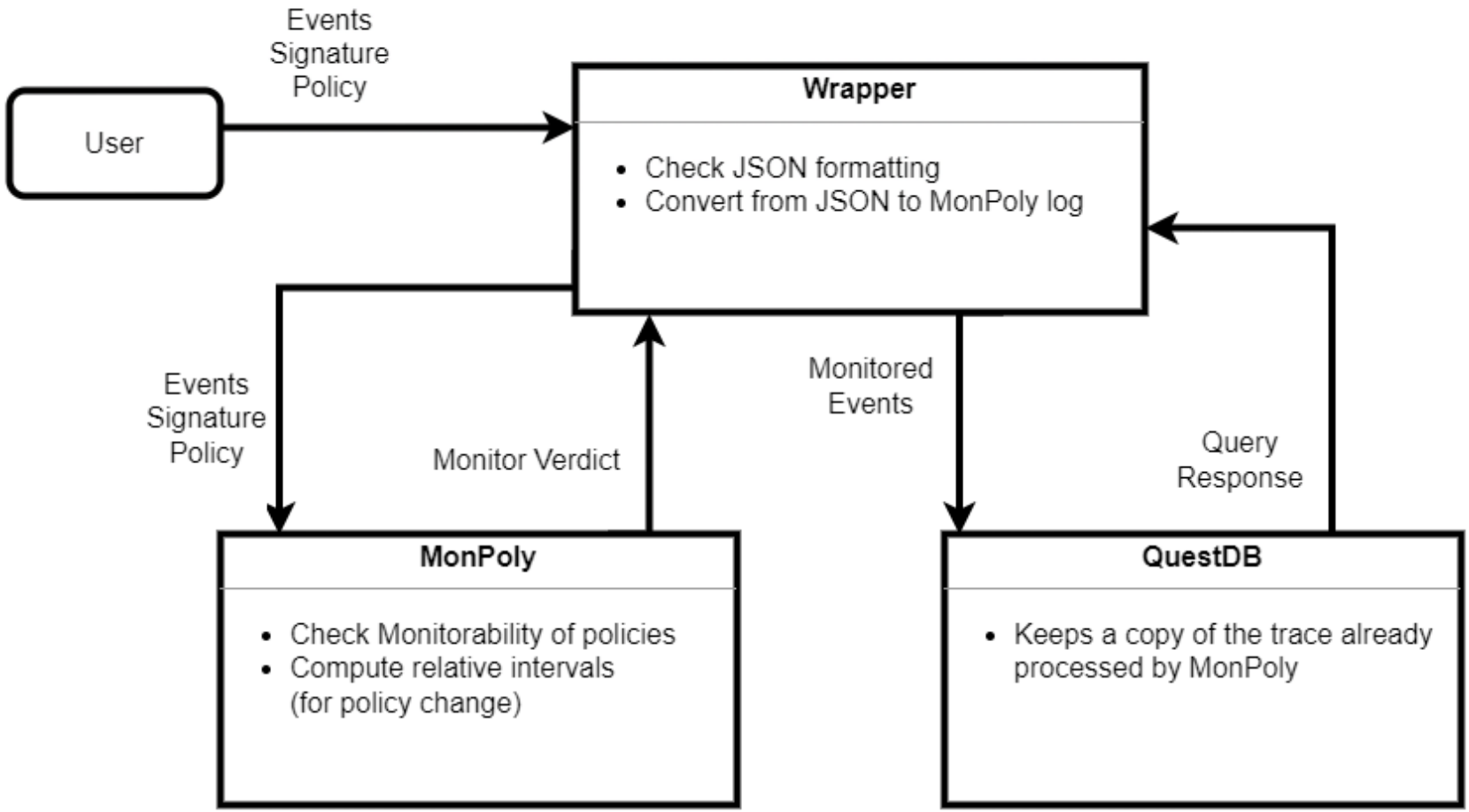
signature.sig

```
P(id:int, action:string)
Q(id1:int, id2:int)
R(int, string, string)
```

The Wrapper - REST API

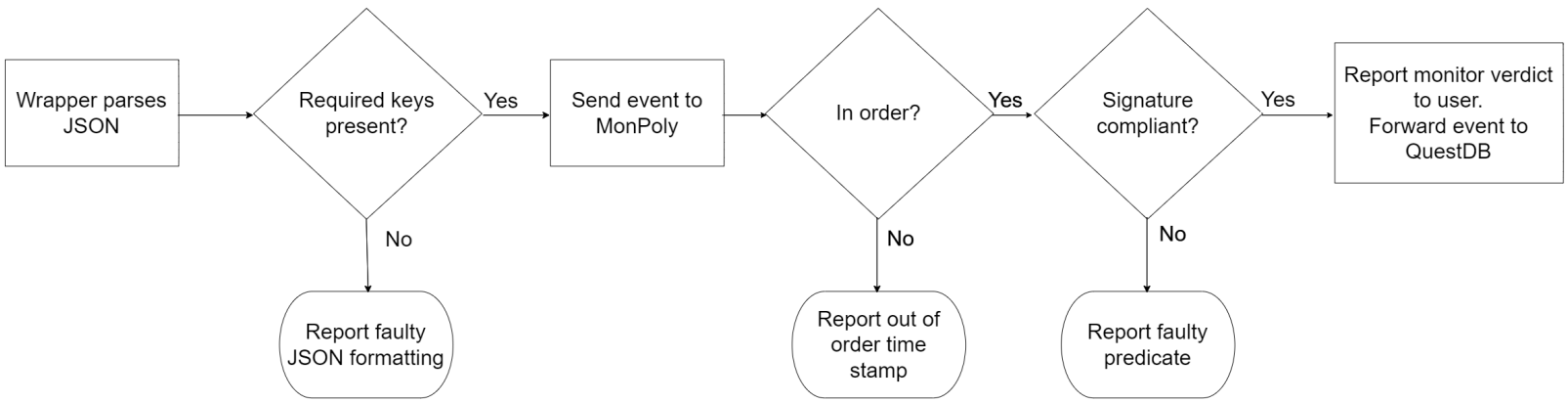
- / (Info-Page)
- /set-policy
- /set-signature
- /start-monitor
- /stop-monitor
- /change-policy
- /get-policy
- /get-signature
- /reset-everything
- /log-events
- /get-events
- /get-most-recent
- /db-set-user
- /db-set-password
- /db-set-host
- /db-set-pgsql-port
- /db-set-influxdb-port
- /db-set-database
- /db-get-user
- /db-get-password
- /db-get-host
- /db-get-pgsql-port
- /db-get-influxdb-port
- /db-get-database

The Wrapper - Architecture



Data flow

- Increased fault tolerance (compared to MonPoly)



Policy Change

- Start a new monitor with the new policy
- **Goal:** Our monitor evaluates the new policy at the current time point just as if it had seen the same trace as the old monitor
- **Naive approach:** Read entire trace again
- **Idea:** Reduce the size of the trace by removing events that do not influence how the new policy gets evaluated

Optimization Example

$$\phi = (P(x, "a") \, \mathcal{S}_{[0,30)} \, Q(x, 2)) \wedge (\neg(R(y, "c", z)) \, \mathcal{S}_{[20,60)} \, P(y, z))$$

- **Relative Interval:** $\mathrm{RI}(\phi) = (-60, 0]$
- **Extended Relative Intervals:** $\mathrm{ERI}(\phi) =$

$$\begin{aligned} P(*, *) &\rightarrow (-60, -20] \\ P(*, "a") &\rightarrow (-30, 0] \\ Q(*, 2) &\rightarrow (-30, 0] \\ R(*, "c", *) &\rightarrow (-60, 0] \end{aligned}$$

- All time points with time stamps not in the interval $\{\text{current time stamp}\} \oplus \mathrm{RI}(\phi)$ do not change the evaluation of ϕ at the current time point.

Interval Operators

Let I and J be two intervals, then

- $I \oplus J = \{i + j \mid i \in I, j \in J\}$
  - $[0, 3] \oplus [-2, 4] = [-2, 7]$
- $I \Cup J$ is the smallest interval that contains all elements that are in at least one of the intervals I and J.
  - $[-4, 1] \Cup [4, 5] = [-4, 5]$

Relative Intervals

Definition

The relative interval of the formula ϕ, $\mathsf{RI}(\phi) \subseteq \mathbb{Z}$ is defined inductively over the formula structure: $\mathsf{RI}(\phi) =$

$$
\begin{cases}
\{0\} & \text{atomic formulas} \\
\mathsf{RI}(\psi) & \neg\psi,\ \exists x.\psi, \forall x.\psi \\
\mathsf{RI}(\psi) \Cup \mathsf{RI}(\chi) & \psi \vee \chi, \psi \wedge \chi \\
(-b, 0] \Cup ((-b, -a] \oplus \mathsf{RI}(\psi)) & \blacksquare_{[a,b)}\psi \\
[0, b) \Cup ([a, b) \oplus \mathsf{RI}(\psi)) & \bigcirc_{[a,b)} \\
(-b, 0] \Cup ((-b, 0] \oplus \mathsf{RI}(\psi)) \Cup ((-b, -a] \oplus \mathsf{RI}(\chi)) & \psi\, \mathcal{S}_{[a,b)} \chi \\
[0, b) \Cup ([0, b) \oplus \mathsf{RI}(\psi)) \Cup ([a, b) \oplus \mathsf{RI}(\chi)) & \psi\, \mathcal{U}_{[a,b)}\, \chi
\end{cases}
$$

Basin et al. "Scalable Offline Monitoring of Temporal Specifications". In: *Formal Methods ind System Design 49 (1 2016)*, pp. 75-108

Extended Relative Intervals

Definition

Let M and N be two masked predicate maps and T a positive interval, then

$$\begin{aligned}
M \mathbin{\ddot{\Cup}} N =& \{p(I) \to (I \Cup J) \mid p(I) \to I \in m \text{ and } p(I) \to J \in n\} \\
&\cup \{p(I) \to I \mid (p(I) \to I \in m \text{ and } p(I) \in \mathrm{k}(M) \setminus \mathrm{k}(N))\} \\
&\cup \{p(I) \to I \mid (p(I) \to I \in n \text{ and } p(I) \in \mathrm{k}(N) \setminus \mathrm{k}(M))\} \\
T \mathbin{\dot{\Cup}} M =& \{p(I) \to (T \Cup I) \mid p(I) \to I \in M\} \\
T \mathbin{\dot{\oplus}} M =& \{p(I) \to (T \oplus I) \mid p(I) \to I \in M\}
\end{aligned}$$

Relative Intervals

Definition

The relative interval of the formula ϕ, $\mathsf{RI}(\phi) \subseteq \mathbb{Z}$ is defined inductively over the formula structure: $\mathsf{RI}(\phi) =$

$$
\begin{cases}
\{0\} & \text{atomic formulas} \\
\mathsf{RI}(\psi) & \neg\psi,\ \exists x.\psi, \forall x.\psi \\
\mathsf{RI}(\psi) \Cup \mathsf{RI}(\chi) & \psi \vee \chi, \psi \wedge \chi \\
(-b, 0] \Cup ((-b, -a] \oplus \mathsf{RI}(\psi)) & \bullet_{[a,b)}\psi \\
[0, b) \Cup ([a, b) \oplus \mathsf{RI}(\psi)) & \bigcirc_{[a,b)} \\
(-b, 0] \Cup ((-b, 0] \oplus \mathsf{RI}(\psi)) \Cup ((-b, -a] \oplus \mathsf{RI}(\chi)) & \psi\, \mathcal{S}_{[a,b)} \chi \\
[0, b) \Cup ([0, b) \oplus \mathsf{RI}(\psi)) \Cup ([a, b) \oplus \mathsf{RI}(\chi)) & \psi\, \mathcal{U}_{[a,b)}\, \chi
\end{cases}
$$

Basin et al. "Scalable Offline Monitoring of Temporal Specifications". In: *Formal Methods ind System Design 49 (1 2016)*, pp. 75-108

Extended Relative Intervals

$$\mathrm{ERI}(\phi) = \begin{cases} \{\} & \text{atomic formulas} \\ & \text{excl. predicates} \\ \{p(m) \to [0,0]\} & \text{predicate } p \\ & \text{with mask } m, \\ \mathrm{ERI}(\psi) & \neg\psi, \exists x.\psi, \forall x.\psi \\ \mathrm{ERI}(\psi) \,\dot{\Cup}\, \mathrm{ERI}(\chi) & \psi \vee \chi, \psi \wedge \chi \\ (-b, 0] \,\dot{\Cup}\, ((-b, -a] \,\dot{\oplus}\, \mathrm{ERI}(\psi)) & \bullet_{[a,b)}\psi \\ [0, b) \,\dot{\Cup}\, ([a, b) \,\dot{\oplus}\, \mathrm{ERI}(\psi)) & \bigcirc_{[a,b)}\psi \\ (-b, 0] \,\dot{\Cup}\, ((-b, 0] \,\dot{\oplus}\, \mathrm{ERI}(\psi)) \,\dot{\Cup}\, ((-b, -a] \,\dot{\oplus}\, \mathrm{ERI}(\chi)) & \psi \,\mathcal{S}_{[a,b)}\, \chi \\ [0, b) \,\dot{\Cup}\, ([0, b) \,\dot{\oplus}\, \mathrm{ERI}(\psi)) \,\dot{\Cup}\, ([a, b) \,\dot{\oplus}\, \mathrm{ERI}(\chi)) & \psi \,\mathcal{U}_{[a,b)}\, \chi \end{cases}$$

We proofed that this definition is a correct approximation such that it extracts all necessary time points from a trace and that the evaluation of ϕ is the same for the original trace and the extraction.

Extended Relative Intervals - SQL Query

$$\phi = (P(x, "a") \, \mathcal{S}_{[0,30)} \, Q(x, 2)) \wedge (\neg(R(y, "c", z)) \, \mathcal{S}_{[20,60)} \, P(y, z))$$

- **Relative Interval:** $\mathrm{RI}(\phi) = (-60, 0]$
- **Extended Relative Intervals:** $\mathrm{ERI}(\phi) =$

$$\begin{aligned} P(*, *) &\rightarrow (-60, -20] \\ P(*, "a") &\rightarrow (-30, 0] \\ Q(*, 2) &\rightarrow (-30, 0] \\ R(*, "c", *) &\rightarrow (-60, 0] \end{aligned}$$

Extended Relative Intervals - SQL Query

```
SELECT * FROM P WHERE
    (time_stamp > <NOW>-60 AND time_stamp <= <NOW>-20)
    OR
    (x2 = "a" AND
    time_stamp > <NOW> -30 AND time_stamp <= <NOW>-0);
SELECT * FROM Q WHERE
     (x2 = 2 AND
    time_stamp > <NOW>-30 AND time_stamp <= <NOW>-0);
SELECT * FROM R WHERE
     (x2 = "c" AND
    time_stamp > <NOW>-60 AND time_stamp <= <NOW>-0);
SELECT * FROM time_points WHERE
    (time_stamp > <NOW>-60 AND time_stamp <= <NOW>-0);
```

Database Result Conversion

QuestDB Response


```
["R":[],
"Q":[[0,8,65536,"Thu, 01 Jan 1970 00:00:01 GMT"]],
"P":[],
"time_points":[[65536,"Thu, 01 Jan 1970 00:00:01 GMT"],
[65537,"Thu, 01 Jan 1970 00:00:02 GMT"]]]
```


wrapper.json


```
["predicates":["name":"Q","occurrences":[[0,8]]],
"timepoint":65536,"timestamp":"1970-01-01 00:00:01",
"predicates":[],"timepoint":65537,"timestamp":"1970-01-01
00:00:02"]
```


monpoly.log

```
@1 Q(0, 8) ;
@2 ;
```

Partial Policy Change in MonPoly (Work in Progress)

- Only for **First Order Logic** operators above temporal operators
- **Named formulas:**
  `NAME[f1, name1] OR NAME[f2 and f3, name2]`
- Added *data types* for `NAME` in MonPoly
- Updated *formula parser* for `NAME` constructs
- Started work on commands for adding and removing parts of formulas.
- **Commands** to add or remove conjuncts or disjuncts:

```
>remove_part <name><
>add_conjunct <name1> <name2> <formula><
>add_disjunct <name1> <name2> <formula><
```

Partial Policy Change in MonPoly (Work in Progress)

$$\phi = (P(x, "a") \, \mathcal{S}_{[0,30)} \, Q(x, 2)) \wedge (\neg(R(y, "c", z)) \, \mathcal{S}_{[20,60)} \, P(y, z))$$

Could become:

```
NAME[(P(x,"a") SINCE[0,30) Q(x,2)), part1]
AND
NAME[(NOT (R(y,"c",z)) SINCE[20,60) P(y,z)), part2]

>add_disjunct part1 part3 'NOT P(2,"b")'<

(NAME[P(x,"a") SINCE[0,30) Q(x,2), part1]
OR
NAME[NOT P(2,"b"), part3])
AND
NAME[NOT (R(y,"c",z)) SINCE[20,60) P(y,z), part2]
```

Partial Policy Change in MonPoly (Work in Progress)

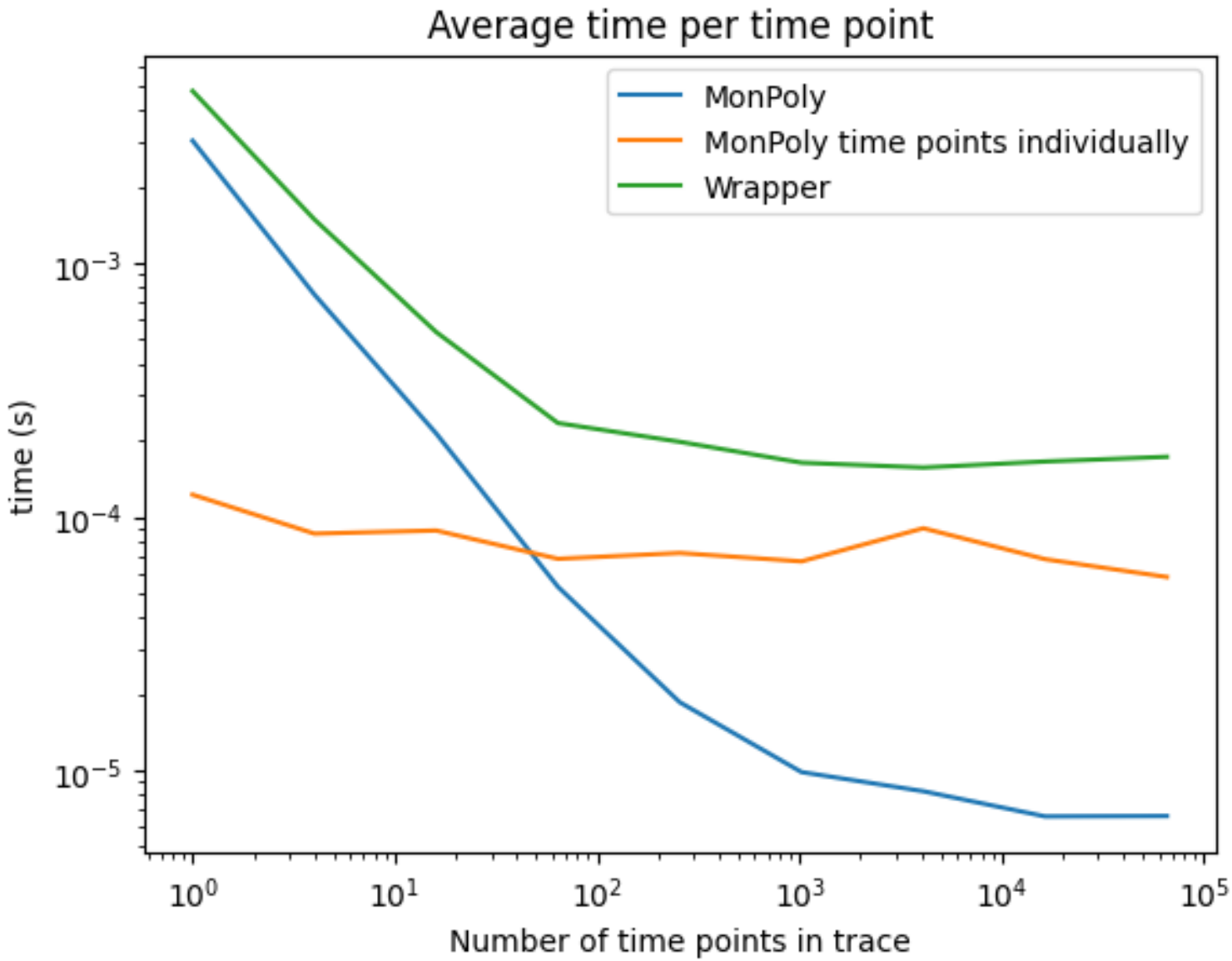
Next up:

- Compute the internal state for formula parts that will be added.
- Combine existing state with the state of the new formula.
- Update state when a formula part gets removed

Overhead of the Wrapper

- Random policy with intervals bounded by $[0, 20)$
- Random trace with 4^i time points, for $i = 0, \ldots, 8$
- Measure the time it takes for MonPoly to monitor these traces
- Measure the time it takes for MonPoly to monitor the traces, when the time points are not in a single file, but sent individually
  - This is a better simulation of online monitoring
- Measure the time it takes for the wrapper to monitor these traces
- This was done **5 times** (with 5 different policies)

Overhead of the Wrapper



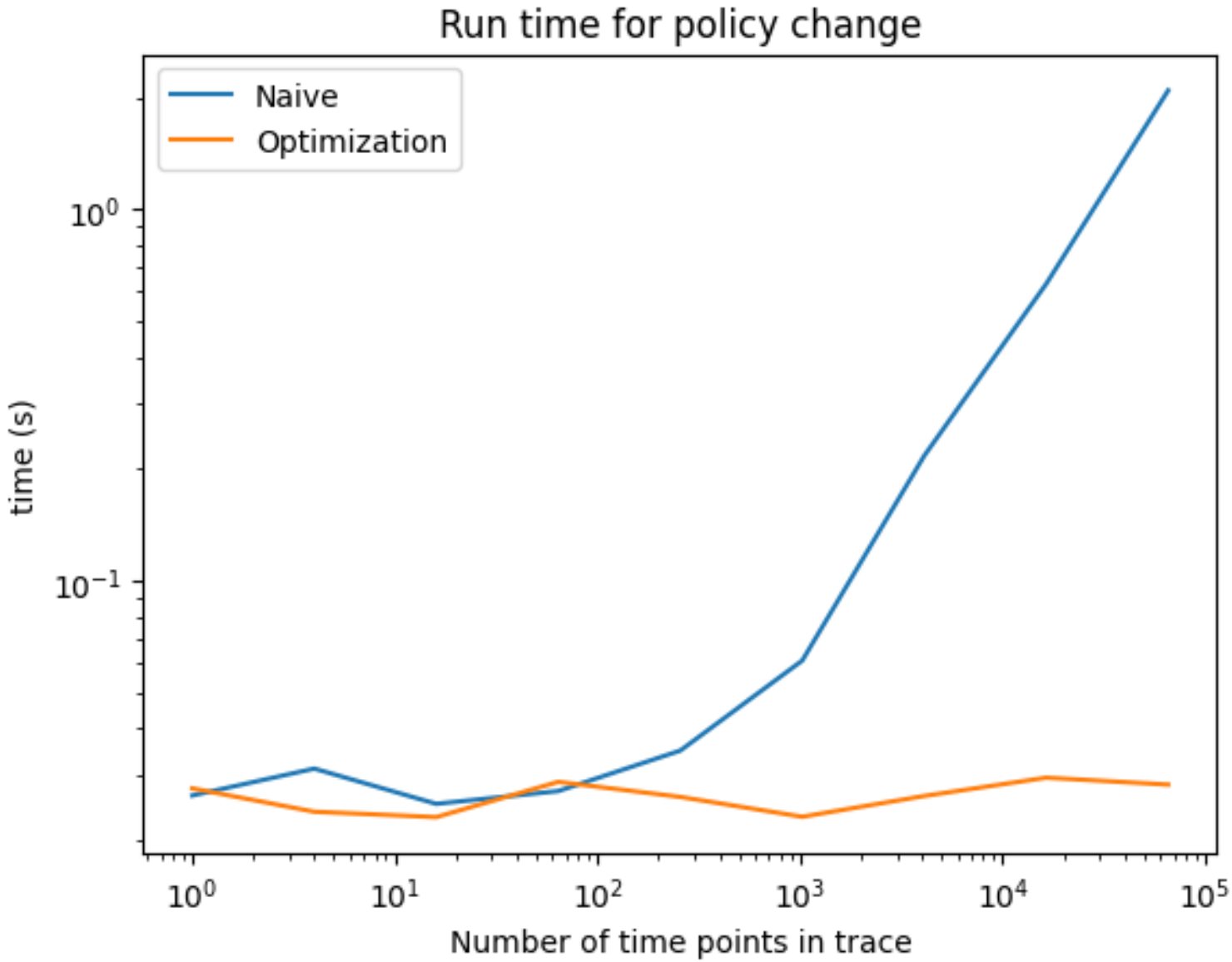
Overhead of the Wrapper - Conclusion

- More detailed profiling is required
- Possible sources of the overhead are:
  - The conversion of our JSON format for logs to the MonPoly format
  - The wrapper sends time points synchronously to MonPoly
  - The wrapper must go over the events once to send them to MonPoly and again later to send them to QuestDB

Policy Change Optimization

- Policy change with 2 randomly generated formulas
- For every pair of formulas the policy change is done with random traces of different length
- For the policies all temporal operators have intervals that are at most $[0, 20)$
- The trace length refers to the number of time points in the trace
- We used traces with lengths 4^i for $i = 0, \ldots, 8$
- **In conclusion:** 9 times policy change for each of the 5 formula pairs

Policy Change Optimization



Outlook

- Reduce overhead of the wrapper
  - More in depth profiling of the wrapper
  - Send time points asynchronously (don't wait for response before sending the next time point)
- Speed up policy change
  - More involved checking for constraints on variables in formulas (such as $x < 20$)
- Continue the partial policy change within MonPoly