# SyMon – Symbolic Monitor for Parametric Spec.

Masaki Waga ・ Kyoto University & National Institute of Informatics

**Q. How can we detect patterns with unknown thresholds, IDs, etc.?**

**Our Approach** Use parametric patterns and synthesize param. val. wrt. the log

## Example 1: Frequent Requests

**Pattern:** From a user with (parametric) ID `current_sender`, > 10 requests occur within 3 seconds

```
#!/usr/bin/env symon -dnf
var {
    current_sender: string;
    count: number;
}

signature arrival {
    sender: string;
}

one_or_more {
    arrival(sender)
};
# Nondeterministically start counting for current_sender
arrival( sender | sender == current_sender | count := 1 );
within (<= 3) {
    one_or_more {
        one_of {
            arrival( sender | sender != current_sender && count <= 10 )
        } or {
            arrival( sender | sender == current_sender && count < 10 | count := count + 1 )
        }
    };
    arrival( sender | sender == current_sender && count = 10 | count := count + 1 )
}
```

String parameter for the id of the current sender

Ignore irrelevant message arrivals

Nondeterministically
- start counting
- choose current_sender

Finish when the 11th arrival is observed

**Monitored Log**

```
arrival   alice@example.com   0.0
arrival   bob@example.org     0.3
arrival   alice@example.com   0.5
arrival   carol@example.net   0.7
arrival   alice@example.com   0.8
arrival   alice@example.com   1.0
arrival   bob@example.org     1.2
arrival   alice@example.com   1.5
arrival   alice@example.com   1.7
arrival   alice@example.com   1.9
arrival   carol@example.net   2.0
arrival   alice@example.com   2.2
arrival   alice@example.com   2.4
arrival   bob@example.org     2.5
arrival   alice@example.com   2.7
arrival   carol@example.net   2.9
arrival   alice@example.com   3.1
arrival   bob@example.org     3.25
arrival   alice@example.com   3.3
arrival   bob@example.org     3.4
```

Nondeterministically ignore irrelevant arrival actions (internally by nondeterministic branching + BFS after a translation to an automaton)

11th arrival by alice@example.com since time 0.5

**Monitoring Result**

```
@3.300000.  (time-point 18) current_sender: alice@example.com  count: 10
```

## Example 2: Periodic Withdrawal

**Pattern:** Withdrawal greater than unknown value `threshold` occurs every `period`±1 time units, where `period` is an unknown constant

```
#!/usr/bin/env symon -pnf
var {
    threshold: number;
    period: param;
}

signature withdraw {
    amount: number;
}

zero_or_more {
    withdraw( amount | amount < threshold )
};
withdraw( amount | amount >= threshold );
one_or_more {
      # We constrain the time elapse between two relevant withdraw events with a closed interval.
      within [period - 1, period + 1] {
          zero_or_more {
              withdraw( amount | amount < threshold )
          };
          withdraw( amount | amount >= threshold )
      }
};
zero_or_more {
    withdraw( amount | amount < threshold )
}
```

Parametric threshold and the period

Ignore irrelevant withdrawals

Relevant withdrawals occur periodically

**Monitored Log**

```
withdraw   1000   0.0
withdraw   200    1.0
withdraw   1200   10.0
withdraw   1300   20.5
withdraw   300    22.0
withdraw   1100   31.2
```

**Monitoring Result**

period ∈ [9.7, 11] for threshold ∈ (300, 1000]

```
...

@31.2.  (time-point 5)  Num: threshold > 300, -threshold >= -1000  Clock:
10*last_interval = 107, -period >= -11, 10*period >= 97

@31.2.  (time-point 5)  Num: threshold > 1000, -threshold >= -1100   Clock:
10*last_interval = 107, -2*period >= -23, 10*period >= 97

@31.2.  (time-point 5)  Num: threshold > 1100, -threshold >= -1200   Clock:
5*last_interval = 106, -2*period >= -23, 2*period >= 19
```

period ∈ [9.5, 11.5] for threshold ∈ (1100, 1200]

period ∈ [9.7, 11.5] for threshold ∈ (1000, 1100]

# Installation and Usage of SyMon

**Homebrew (for instance on macOS)**

```
$ brew install --HEAD maswag/scientific/symon
$ symon -pnf [foo.symon] < [bar.log]
```

**Docker**

```
$ docker pull maswag/symon:new_syntax
$ docker run -i -v $PWD:/tmp maswag/
symon:new_syntax -pnf /tmp/[foo.symon] < [bar.log]
```

GPL Free Software — Free as in Freedom

# References


- [Paper on the algorithm] Waga, Masaki, Étienne André, and Ichiro Hasuo. "Symbolic monitoring against specifications parametric in time and data." *International Conference on Computer Aided Verification*. Cham: Springer International Publishing, 2019.


- [GitHub] https://github.com/MasWag/SyMon
- [Docker Hub] https://hub.docker.com/r/maswag/symon

Note: We plan to further improve the syntax. Examples in this poster may not work in future versions.

# Future directions

- Improve spec. language to make it easier to express common patterns
  - e.g., filtering irrelevant actions based on identifier
- Make the specification language more intuitive
- Improve efficiency through better handling of variables
- Support hyperproperties, e.g., for monitoring robustness and fairness

This work is partially supported by JST ACT-X Grant No. JPMJAX200U, JST PRESTO Grant No. JPMJPR22CA, JST ERATO HASUO Metamathematics for Systems Design Project (No. JPMJER1603), JST CREST Grant Number JPMJCR2012, and JSPS KAKENHI Grant Number 15KT0012, 18J22498, 19H04084, and 22K17873.