MONITORING DECENTRALIZED SPECIFICATIONS

Antoine El-Hokayem   Yliès Falcone

December 7, 2017

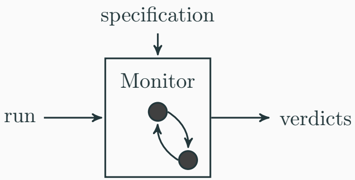
Univ. Grenoble Alpes, Inria, CNRS (Grenoble, France)

GDR GPL - Journée MTV2
Paris, France

(Decentralized) Monitoring

MONITORING (AKA RUNTIME VERIFICATION) ↪ OVERVIEW

- Lightweight verification technique
- Checks whether a run of a program conforms to a specification (As opposed to model checking which verifies all runs)
- Monitors are synthesized and integrated to observe the system
- Monitors determine a verdict: $\mathbb{B}_3 = \{\top, \bot, ?\}$
  - $\top$ (true): run complies with specification
  - $\bot$ (false): run does not comply with specification
  - ?: verdict cannot be determined (yet)



MONITORING $\hookrightarrow$ SYSTEM ABSTRACTION

1. Components ($\mathcal{C}$)
2. Atomic propositions (AP)
3. Observations/Events ($AP \to \mathbb{B}_2$, possibly partial)
4. Trace: a sequence of events for each component (partial function)

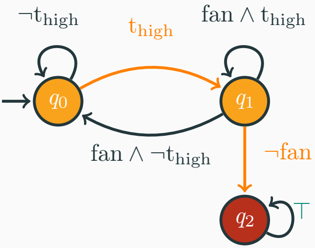
Example

1. $\{c_0, c_1\}$ (Temp sensor + Fan)
2. $\{\mathrm{t}_{\mathrm{low}}, \mathrm{t}_{\mathrm{med}}, \mathrm{t}_{\mathrm{high}}, \mathrm{t}_{\mathrm{crit}}, \mathrm{fan}\}$ (e.g., $\mathrm{t}_{\mathrm{crit}}$ "temperature critical")
3. $\{\langle \mathrm{t}_{\mathrm{low}}, \top\rangle, \langle \mathrm{fan}, \bot\rangle\}$ — "temperature is low and fan is not on"
4. $\begin{bmatrix} 0 \mapsto c_0 & \mapsto \{\langle \mathrm{t}_{\mathrm{low}}, \top\rangle, \langle \mathrm{t}_{\mathrm{med}}, \bot\rangle, \dots\} & 0 \mapsto c_1 & \mapsto \{\langle \mathrm{fan}, \bot\rangle\} \\ 1 \mapsto c_0 & \mapsto \{\langle \mathrm{t}_{\mathrm{med}}, \top\rangle, \dots\} & 1 \mapsto c_1 & \mapsto \{\langle \mathrm{fan}, \bot\rangle\} \\ 2 \mapsto c_0 & \mapsto \{\langle \mathrm{t}_{\mathrm{high}}, \top\rangle, \dots\} & 2 \mapsto c_1 & \mapsto \{\langle \mathrm{fan}, \top\rangle\} \end{bmatrix}$

$$\{\langle \mathrm{t}_{\mathrm{low}}, \top\rangle, \langle \mathrm{fan}, \bot\rangle, \dots\} \cdot \{\langle \mathrm{t}_{\mathrm{med}}, \top\rangle, \langle \mathrm{fan}, \bot\rangle, \dots\} \cdot \{\langle \mathrm{t}_{\mathrm{high}}, \top\rangle, \langle \mathrm{fan}, \top\rangle, \dots\}$$

MONITORING USING AUTOMATA ↪ EXAMPLE

"Fan must always be turned on when temperature is high"



$$\boldsymbol{G}(\mathrm{t_{high}} \implies \boldsymbol{X}\mathrm{fan})$$

1. At $t = 1$, from q_0:

   1.1 Observe

$\mathrm{t_{high}}$	$\top$
fan	$\bot$

   1.2 Eval

$\neg\mathrm{t_{high}}$	$\bot$
$\mathrm{t_{high}}$	$\top$

2. At $t = 2$, from q_1:

   2.1 Observe

$\mathrm{t_{high}}$	$\top$
fan	$\bot$

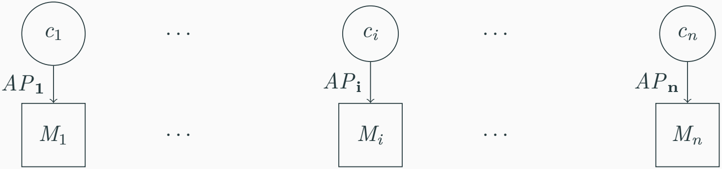
   2.2 Eval

$\mathrm{fan} \wedge \neg\mathrm{t_{high}}$	$\bot$
$\mathrm{fan} \wedge \mathrm{t_{high}}$	$\bot$
$\neg\mathrm{fan}$	$\top$

Monitoring this property requires a central observation point!

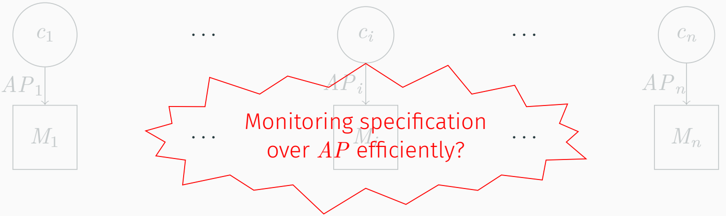
DECENTRALIZED MONITORING ↪ PROBLEM STATEMENT

- General setting
  - $\mathcal{C} = \{c_0, \ldots, c_n\}$: components
  - $AP = AP_0 \cup \ldots \cup AP_n$: atomic propositions, partitioned by $\mathcal{C}$
  - no central observation point
  - but monitors attached to components
- Issues in decentralized monitoring
  - partial views of AP – unknown global state
  - partial execution of the automaton (evaluation)
  - communication between monitors



DECENTRALIZED MONITORING ↪ PROBLEM STATEMENT

- General setting
- Issues in decentralized monitoring
  - partial views of AP – unknown global state
  - partial execution of the automaton (evaluation)
  - communication between monitors
- Existing approaches:
  - based on LTL rewriting — unpredictability of monitor performance
  - all monitors check the same specification — inefficiency



GOALS

Define a methodology of design and evaluation of decentralized monitoring

1. Aim for predictable behavior
   - Move from LTL → Automata
   - Common ground to compare existing (and future) strategies
2. Separate monitor synthesis from monitoring strategies
   - Centralized specification → Decentralized specification
   - Monitorability of a decentralized specification
   - Define a general decentralized monitoring algorithm

★ Extend tooling support for the design methodology

★ Ensure reproducibility

Monitoring with EHEs

Execution History Encoding ↪ Information as Atoms

★ Encode the execution as a datastructure that
  - supports flexibility when receiving partial information
  - is insensitive to the reception order of information
  - has predictable size and operations
- Atomic propositions → Atoms
  - Allow algorithms to add data to observations ($\text{enc} : AP \rightarrow Atoms$)
  - Ordering information (timestamp, round number, vector clock etc.)
- Monitors store Atoms in their Memory
- Monitors need to evaluate $Expr_{Atoms}$
  - rewrite using Memory
  - simplify using Boolean logics (much easier than simplification for LTL)

$$Expr_{Atoms} \times Mem \rightarrow \mathbb{B}_3$$

$$\text{eval}(expr, \mathcal{M}) = \text{simplify}(\text{rw}(expr, \mathcal{M}))$$

$$\text{eval}(\langle 1, \text{t}_{\text{high}} \rangle \wedge \langle 2, \text{fan} \rangle, [\langle 1, \text{t}_{\text{high}} \rangle \mapsto \bot]) = \bot \wedge \langle 2, \text{fan} \rangle = \bot$$

EXECUTION HISTORY ENCODING ↪ AUTOMATA EXECUTION

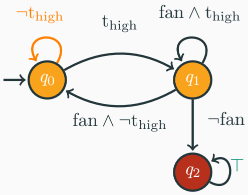
- EHE is a partial function:

$$\mathcal{I} : \mathbb{N} \times Q_{\mathcal{A}} \rightarrow \mathit{Expr}_{\mathit{Atoms}}$$
$$\mathcal{I}(t, q) = \mathit{expr}$$

  - For a given timestamp t
  - The automaton is in state q iff
  - $\mathrm{eval}(\mathit{expr}, \mathcal{M}) = \top$

$$\mathcal{I}(2, q_0) = [\neg\langle 1, \mathrm{t_{high}}\rangle \wedge \neg\langle 2, \mathrm{t_{high}}\rangle] \vee [\langle 1, \mathrm{t_{high}}\rangle \wedge (\langle 2, \mathrm{fan}\rangle \wedge \neg\langle 2, \mathrm{t_{high}}\rangle)]$$

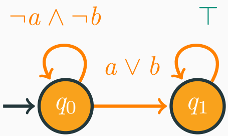
$$\mathrm{eval}(\mathcal{I}(2, q_0), [\langle 1, \mathrm{t_{high}}\rangle \mapsto \bot]) = \mathrm{eval}(\neg\langle 2, \mathrm{t_{high}}\rangle, \ldots) = ?$$



- EHE is constructed recursively and lazily (as needed and on-the-fly) using $\mathcal{A}$

EXECUTION HISTORY ENCODING $\hookrightarrow$ CONSTRUCTION

$$\mathcal{I}^2 = \text{mov}([0 \mapsto q_0 \mapsto \top], 0, 2)$$



t	q	expr
0	q_0	$\top$
1	q_0	$\top \wedge \quad \neg\langle 1, a\rangle \wedge \neg\langle 1, b\rangle$
1	q_1	$\langle 1, a\rangle \vee \langle 1, b\rangle$
2	q_0	$(\neg\langle 1, a\rangle \wedge \neg\langle 1, b\rangle) \wedge (\neg\langle 2, a\rangle \wedge \neg\langle 2, b\rangle)$
2	q_1	$[(\neg\langle 1, a\rangle \wedge \neg\langle 1, b\rangle) \wedge (\langle 2, a\rangle \vee \langle 2, b\rangle)] \vee [(\langle 1, a\rangle \vee \langle 1, b\rangle) \wedge \top]$

$\vdots$

EXECUTION HISTORY ENCODING $\hookrightarrow$ PROPERTIES

1. Soundness (provided that observations can be totally ordered)
    - For the same trace, EHE and $\mathcal{A}$ report the same state
    - → They find the same verdict
2. Strong Eventual Consistency (SEC)
    - We can merge EHEs by disjoining ($\vee$) each entry $\langle t, q \rangle$
    - $\vee$ is commutative, associative and idempotent
    - → EHE is a state-based replicated data-type (CvRDT)
    - → Monitors that exchange their EHE find the same verdict
    - → Can monitor centralized specification shared with multiple monitors
3. Predictable size
    - The EHE encodes all potential and past states, as needed
    - The more we keep track of potential states, the bigger the size
    - → We can assess algorithms by how they manipulate the EHE

EXECUTION HISTORY ENCODING ↪ ANALYSIS

- Information Delay (δ)

  Timestamps needed to expand before determining a state

  Potential states to keep track of

- Size of expression grows with each move beyond t

- Size of EHE:

$$|\mathcal{I}^{\delta}| = \mathcal{O}(\delta|Q|\sum_{1}^{\delta} LP)$$
$$= \mathcal{O}(\delta^{2}|Q|LP)$$

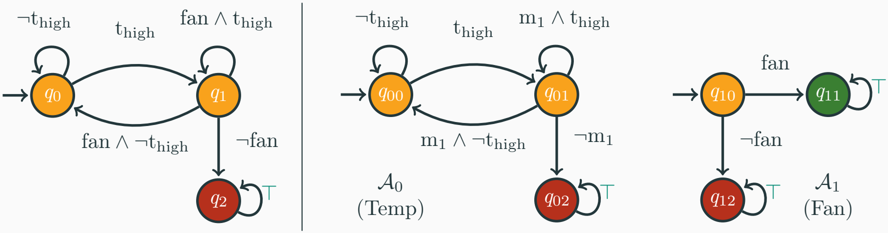
$$t \mapsto q \mapsto \top$$

$$\delta \left\{ \begin{array}{lll}
t+1 \mapsto & \left.\begin{array}{ll} q_0 & \mapsto e_{10} \\ q_1 & \mapsto e_{11} \\ & \vdots \\ q_{|Q|-1} & \mapsto e_{1(|Q|-1)} \end{array}\right\} & |Q| \\
t+2 \mapsto & \left.\begin{array}{ll} q_0 & \mapsto e_{20} \\ & \vdots \\ q_{|Q|-1} & \mapsto e_{2(|Q|-1)} \end{array}\right\} & |Q| \\
\vdots & & \\
t+\delta \mapsto & \left.\begin{array}{ll} q_0 & \mapsto e_{\delta 0} \\ q_1 & \mapsto e_{\delta 1} \\ & \vdots \\ q_{|Q|-1} & \mapsto e_{\delta(|Q|-1)} \end{array}\right\} & |Q|
\end{array}\right.$$

Monitoring Decentralized Specifications

DECENTRALIZED SPECIFICATION

- Each monitor is associated with a tuple $\langle \mathcal{A}, c \rangle$
  - $\mathcal{A}$ is its specification automaton
  - c is the component the monitor is attached to
- The transition labels of an automaton $\mathcal{A}$ are restricted to:
  - Atomic propositions local to the attached component
  - References to other monitors



DECENTRALIZED SPECIFICATION ↪ SEMANTICS & MONITORABILITY

- For an automaton $\mathcal{A}_k$, to evaluate a label m_j at t with a trace tr
  - Run tr starting with t on $\mathcal{A}_j$ starting from q_{j_0}
  - Consider the verdict of the run to be the observation m_j at t

(!) If $\mathcal{A}_j$ never reaches a final verdict we will not be able to monitor $\mathcal{A}_k$

(?) Monitorability: "From any state in $\mathcal{A}_k$, we can reach a final verdict"

- $\mathrm{monitorable}(\mathcal{A}_k)$ iff $\forall q \in Q_{\mathcal{A}_k}, \exists q_f \in Q_{\mathcal{A}_k}, \exists e_f \in \mathrm{paths}(q, q_f)$, s.t.
  1. Path can be taken: e_f is satisfiable;
  2. Path leads to a verdict: $\mathrm{ver}_k(q_f) \in \{\bot, \top\}$;
  3. All its dependencies are monitorable:

     $\forall \mathrm{m}_j \in \mathrm{dep}(e_f)$: $\mathrm{monitorable}(\mathcal{A}_j)$.

- Expressions that determine paths between states (n = path length)
  - $\mathrm{paths}(q_s, q_e) = \left\{ \mathtt{expr} \;\middle|\; \begin{array}{l} \exists n \in \mathbb{N} : \mathcal{I}^n(n, q_e) = \mathtt{expr} \\ \wedge\, \mathcal{I}^n = \mathrm{mov}([0 \mapsto q_s \mapsto \top], 0, n) \end{array} \right\}$

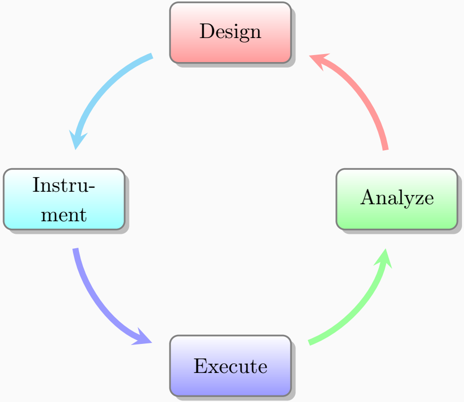
GENERALIZED MONITORING ALGORITHM ↪ OVERVIEW

1. Setup (Deploy)
   - 1.1 Analyze and convert the specification as necessary
   - 1.2 Create monitors, and assign them a specification
     - (!) The monitor handles encoding of AP and Memory
   - 1.3 Attach monitors to components

2. Monitoring
   - 2.1 Wait to receive observations from attached component
   - 2.2 Receive messages (EHE) from monitors
   - 2.3 Process observations and messages (update the local EHE)
   - 2.4 Communicate with other monitors

The THEMIS Approach

THEMIS ↪ Overview



Step	Description
Design	Design a monitoring algorithm
Instru-ment	Create or re-use metrics. Metrics are automatically instrumented using AspectJ
Execute	Use `THEMIS` tools to execute one or more monitoring run(s)
Analyze	Measures are stored in a database for postmortem analysis

Setup

```java
Map<Integer, ? extends Monitor> setup() {
  config.getSpec().put("root",
      Convert.makeAutomataSpec(
        config.getSpec().get("root")));
  Map<Integer, Monitor> mons = new HashMap<Integer, Monitor>();
  Integer i = 0;
  for(Component comp : config.getComponents()) {
    MonMigrate mon = new MonMigrate(i);
    attachMonitor(comp, mon);
    mons.put(i, mon);
    i++;
  }
  return mons;
}
```

Monitor

```java
void monitor(int t, Memory<Atom> observations)
throws ReportVerdict, ExceptionStopMonitoring {
    m.merge(observations);
    if(receive()) isMonitoring = true;
    if(isMonitoring) {
      if(!observations.isEmpty())
        ehe.tick();
      boolean b = ehe.update(m, -1);
      if(b) {
        VerdictTimed v = ehe.scanVerdict();
        if(v.isFinal())
          throw new ReportVerdict(v.getVerdict(), t);
        ehe.dropResolved();
      }
      int next  = getNext();
      if(next != getID()) {
        Representation toSend = ehe.sliceLive();
        send(next, new RepresentationPacket(toSend));
        isMonitoring = false;
      }
    }
  }
```

EXAMPLES ↪ METRICS

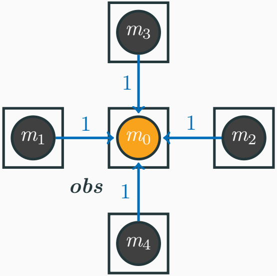
```
void setupRun(MonitoringAlgorithm alg) {
 addMeasure(new Measure("msg_num","Msgs",0L,Measures.addLong));
}
after(Integer to, Message m) : Commons.sendMessage(to, m) {
 update("msg_num" , 1L);
}
```

```
SELECT alg, comps, avg(msg_num), avg(msg_data), count(*)
FROM bench WHERE alg in ('Migration', 'MigrationRR')
GROUP BY alg, comps
```

	alg	comps	avg(msg_num)	avg(msg_data)	count(*)
1	Migration	3	2.04226336011177	267.8458714635	572600
2	Migration	4	2.16402472527473	668.129401098901	364000
3	Migration	5	3.33806822465134	3954.09705050886	530600
4	MigrationRR	3	32.7222301781348	482.572275585051	572600
5	MigrationRR	4	31.8533351648352	932.708425824176	364000
6	MigrationRR	5	19.2345269506219	4361.30746324915	530600

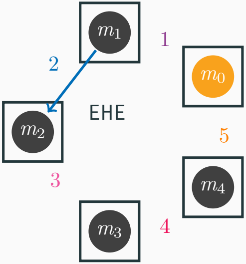
EXISTING ALGORITHMS

STUDYING EXISTING ALGORITHMS ↪ EXPECTED BEHAVIOR



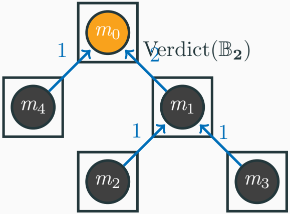
Orchestration

- δ is constant
- #Msgs is linear in components
- |Msg| constant: observations per component



Migration

- δ is linear in components
- #Msgs is constant
- |Msg| is size of EHE: $\mathcal{O}(\delta^2)$, quadratic in components



Choreography

- δ is linear in network depth (algorithm)
- #Msgs is linear in network edges
- |Msg| is constant

#Msgs and |Msg| are predicted on a per round basis

Conclusions

SUMMARY AND FUTURE WORK

⋆ Decentralized Monitoring of (De)Centralized Specifications

1. Aim for predictable behavior → Automata + EHE data structure
2. Separate synthesis from monitoring: decentralized specifications
3. Methodology + tool support for designing, measuring, comparing and extending decentralized RV algorithms
4. Adapted and compared existing algorithms

⋆ Future Work

1. Centralised specification → equivalent decentralized specifications
    - Optimize existing methods
    - Take into account topology of the monitored system
2. Extend THEMIS
    - New metrics
    - Support a fully-asynchronous monitoring approach
    - Better visualization of (the behavior of) algorithms
3. Runtime enforcement of centralized and decentralized specifications

- 11th Euromicro Conference on Real-Time Systems (ECRTS 1999), 9-11 June 1999, York, England, UK, Proceedings. IEEE Computer Society (1999)
- Twelfth ACM/IEEE International Conference on Formal Methods and Models for Codesign, MEMOCODE 2014, Lausanne, Switzerland, October 19-21, 2014. IEEE (2014)
- Ábrahám, E., Palamidessi, C. (eds.): Formal Techniques for Distributed Objects, Components, and Systems - 34th IFIP WG 6.1 International Conference, FORTE 2014, Held as Part of the 9th International Federated Conference on Distributed Computing Techniques, DisCoTec 2014, Berlin, Germany, June 3-5, 2014. Proceedings, Lecture Notes in Computer Science, vol. 8461. Springer (2014)
- Aceto, L., Damgård, I., Goldberg, L.A., Halldórsson, M.M., Ingólfsdóttir, A., Walukiewicz, I. (eds.): Automata, Languages and Programming, 35th International Colloquium, ICALP 2008, Reykjavik, Iceland, July 7-11, 2008, Proceedings, Part I: Tack A: Algorithms, Automata, Complexity, and Games, Lecture Notes in Computer Science, vol. 5125. Springer (2008)

Barringer, H., Falcone, Y., Finkbeiner, B., Havelund, K., Lee, I., Pace, G.J., Rosu, G., Sokolsky, O., Tillmann, N. (eds.): Runtime Verification - First International Conference, RV 2010, St. Julians, Malta, November 1-4, 2010. Proceedings, Lecture Notes in Computer Science, vol. 6418. Springer (2010)

Bartocci, E.: Sampling-based decentralized monitoring for networked embedded systems. In: Bortolussi et al. [12], pp. 85–99, `http://dx.doi.org/10.4204/EPTCS.124.9`

Bartocci, E., Falcone, Y., Bonakdarpour, B., Colombo, C., Decker, N., Havelund, K., Joshi, Y., Klaedtke, F., Milewicz, R., Reger, G., Rosu, G., Signoles, J., Thoma, D., Zalinescu, E., Zhang, Y.: First international competition on runtime verification: rules, benchmarks, tools, and final results of crv 2014. International Journal on Software Tools for Technology Transfer pp. 1–40 (2017), `http://dx.doi.org/10.1007/s10009-017-0454-5`

- Basin, D.A., Klaedtke, F., Zalinescu, E.: Failure-aware runtime verification of distributed systems. In: Harsha and Ramalingam [29], pp. 590–603
- Bauer, A., Leucker, M., Schallhart, C.: Runtime verification for LTL and TLTL. ACM Trans. Softw. Eng. Methodol. 20(4), 14 (2011)
- Bauer, A.K., Falcone, Y.: Decentralised LTL monitoring. In: Giannakopoulou and Méry [28], pp. 85–100
- Bonakdarpour, B., Fraigniaud, P., Rajsbaum, S., Travers, C.: Challenges in fault-tolerant distributed runtime verification. In: Margaria and Steffen [35], pp. 363–370
- Bortolussi, L., Bujorianu, M.L., Pola, G. (eds.): Proceedings Third International Workshop on Hybrid Autonomous Systems, HAS 2013, Rome, Italy, 17th March 2013, EPTCS, vol. 124 (2013), `http://dx.doi.org/10.4204/EPTCS.124`

- Broy, M., a. Peled, D., Kalus, G. (eds.): engineering dependable software systems, NATO science for peace and security series, d: information and communication security, vol. 34. ios press (2013)
- Buchfuhrer, D., Umans, C.: The complexity of boolean formula minimization. In: Aceto et al. [4], pp. 24–35
- Colombo, C., Falcone, Y.: Organising LTL monitors over distributed systems with a global clock. Formal Methods in System Design 49(1-2), 109–158 (2016)
- Cotard, S., Faucou, S., Béchennec, J., Queudet, A., Trinquet, Y.: A data flow monitoring service based on runtime verification for AUTOSAR. In: Min et al. [36], pp. 1508–1515
- Défago, X., Petit, F., Villain, V. (eds.): Stabilization, Safety, and Security of Distributed Systems - 13th International Symposium, SSS 2011, Grenoble, France, October 10-12, 2011. Proceedings, Lecture Notes in Computer Science, vol. 6976. Springer (2011)

Diekert, V., Leucker, M.: Topology, monitorable properties and runtime verification. Theoretical Computer Science 537, 29 – 41 (2014), theoretical Aspects of Computing (ICTAC 2011)

Diekert, V., Muscholl, A.: On distributed monitoring of asynchronous systems. In: Ong and de Queiroz [38], pp. 70–84, http://dx.doi.org/10.1007/978-3-642-32621-9_5

Duret-Lutz, A.: Manipulating LTL formulas using Spot 1.0. In: Proceedings of the 11th International Symposium on Automated Technology for Verification and Analysis (ATVA'13). Lecture Notes in Computer Science, vol. 8172, pp. 442–445. Springer, Hanoi, Vietnam (Oct 2013)

El-Hokayem, A., Falcone, Y.: Themis: A tool for decentralized monitoring algorithms. In: Proceedings of 26th ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA'17-DEMOS), Santa Barbara, CA, USA, July 2017 (2017)

- El-Hokayem, A., Falcone, Y.: Themis website (2017), `https://gitlab.inria.fr/monitoring/themis`
- Falcone, Y.: You should better enforce than verify. In: Barringer et al. [5], pp. 89–105
- Falcone, Y., Cornebize, T., Fernandez, J.: Efficient and generalized decentralized monitoring of regular languages. In: Ábrahám and Palamidessi [3], pp. 66–83
- Falcone, Y., Fernandez, J., Mounier, L.: What can you verify and enforce at runtime? STTT 14(3), 349–382 (2012)
- Falcone, Y., Havelund, K., Reger, G.: A tutorial on runtime verification. In: Broy et al. [13], pp. 141–175
- Finkelstein, A., Estublier, J., Rosenblum, D.S. (eds.): 26th International Conference on Software Engineering (ICSE 2004), 23-28 May 2004, Edinburgh, United Kingdom. IEEE Computer Society (2004)

- Giannakopoulou, D., Méry, D. (eds.): FM 2012: Formal Methods - 18th International Symposium, Paris, France, August 27-31, 2012. Proceedings, Lecture Notes in Computer Science, vol. 7436. Springer (2012)
- Harsha, P., Ramalingam, G. (eds.): 35th IARCS Annual Conference on Foundation of Software Technology and Theoretical Computer Science, FSTTCS 2015, December 16-18, 2015, Bangalore, India, LIPIcs, vol. 45. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik (2015)
- Kiczales, G., Hilsdale, E., Hugunin, J., Kersten, M., Palm, J., Griswold, W.G.: An overview of aspectj. In: Knudsen [32], pp. 327–353
- Kim, M., Viswanathan, M., Ben-Abdallah, H., Kannan, S., Lee, I., Sokolsky, O.: Formally specified monitoring of temporal properties. In: 11th Euromicro Conference on Real-Time Systems (ECRTS 1999), 9-11 June 1999, York, England, UK, Proceedings [1], pp. 114–122
- Knudsen, J.L. (ed.): ECOOP 2001 - Object-Oriented Programming, 15th European Conference, Budapest, Hungary, June 18-22, 2001, Proceedings, Lecture Notes in Computer Science, vol. 2072. Springer (2001)

- Leucker, M., Schallhart, C.: A brief account of runtime verification. J. Log. Algebr. Program. 78(5), 293–303 (2009)
- Leucker, M., Schmitz, M., à Tellinghusen, D.: Runtime verification for interconnected medical devices. In: Margaria and Steffen [35], pp. 380–387
- Margaria, T., Steffen, B. (eds.): Leveraging Applications of Formal Methods, Verification and Validation: Discussion, Dissemination, Applications - 7th International Symposium, ISoLA 2016, Imperial, Corfu, Greece, October 10-14, 2016, Proceedings, Part II, Lecture Notes in Computer Science, vol. 9953 (2016)
- Min, G., Hu, J., Liu, L.C., Yang, L.T., Seelam, S., Lefèvre, L. (eds.): 14th IEEE International Conference on High Performance Computing and Communication & 9th IEEE International Conference on Embedded Software and Systems, HPCC-ICESS 2012, Liverpool, United Kingdom, June 25-27, 2012. IEEE Computer Society (2012)
- Misra, J., Nipkow, T., Sekerinski, E. (eds.): FM 2006: Formal Methods, 14th International Symposium on Formal Methods, Hamilton, Canada,

August 21-27, 2006, Proceedings, Lecture Notes in Computer Science, vol. 4085. Springer (2006)

- Ong, C.L., de Queiroz, R.J.G.B. (eds.): Logic, Language, Information and Computation - 19th International Workshop, WoLLIC 2012, Buenos Aires, Argentina, September 3-6, 2012. Proceedings, Lecture Notes in Computer Science, vol. 7456. Springer (2012)
- Pnueli, A., Zaks, A.: PSL model checking and run-time verification via testers. In: Misra et al. [37], pp. 573–586
- Rosu, G., Havelund, K.: Rewriting-based techniques for runtime verification. Autom. Softw. Eng. 12(2), 151–197 (2005)
- Scheffel, T., Schmitz, M.: Three-valued asynchronous distributed runtime verification. In: Twelfth ACM/IEEE International Conference on Formal Methods and Models for Codesign, MEMOCODE 2014, Lausanne, Switzerland, October 19-21, 2014 [2], pp. 52–61

- Sen, K., Vardhan, A., Agha, G., Rosu, G.: Efficient decentralized monitoring of safety in distributed systems. In: Finkelstein et al. [27], pp. 418–427
- Shapiro, M., Preguiça, N.M., Baquero, C., Zawirski, M.: Conflict-free replicated data types. In: Défago et al. [17], pp. 386–400
- Thati, P., Rou, G.: Monitoring algorithms for metric temporal logic specifications. Electronic Notes in Theoretical Computer Science 113, 145 – 162 (2005), `http://www.sciencedirect.com/science/article/pii/S1571066104052570`
- Valiant, L.G.: A bridging model for parallel computation. Commun. ACM 33(8), 103–111 (Aug 1990)
- Wuu, G.T.J., Bernstein, A.J.: Efficient solutions to the replicated log and dictionary problems. Operating Systems Review 20(1), 57–66 (1986)