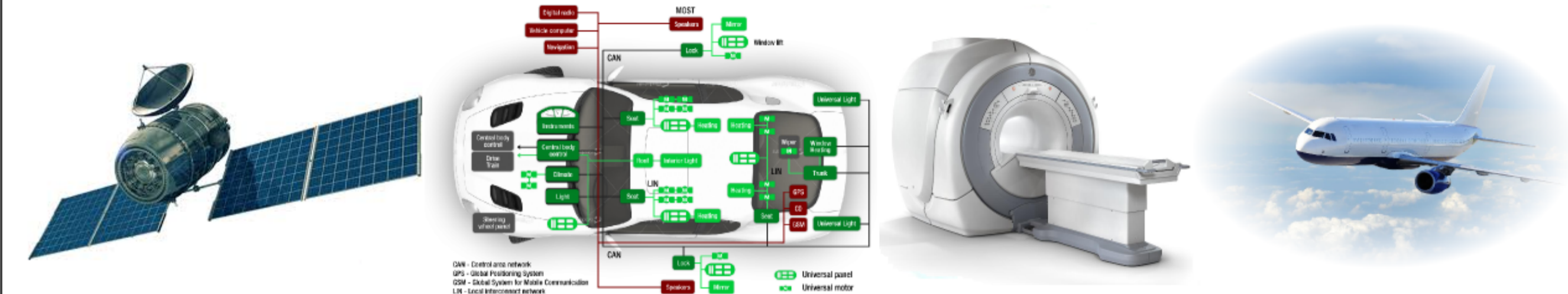


1. Context

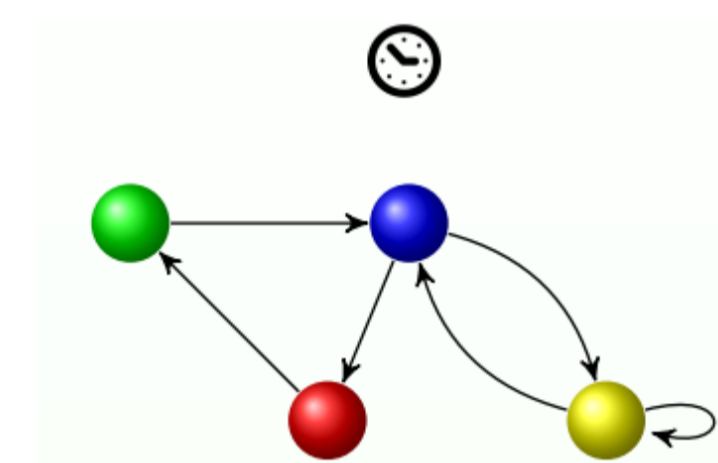
Real-time systems are difficult to test and their failure leads to dramatic consequences



Model checking is an automatic verification technique to verify the correctness of the system model w.r.t. a property:

- **Verification** procedure: exhaustive search of the state space of the model

(State: \circ ; Transition: \rightarrow)



A model of the system



• is unreachable

A property to be satisfied

- **Checking question:** Does the model of the system satisfy the property?

Yes

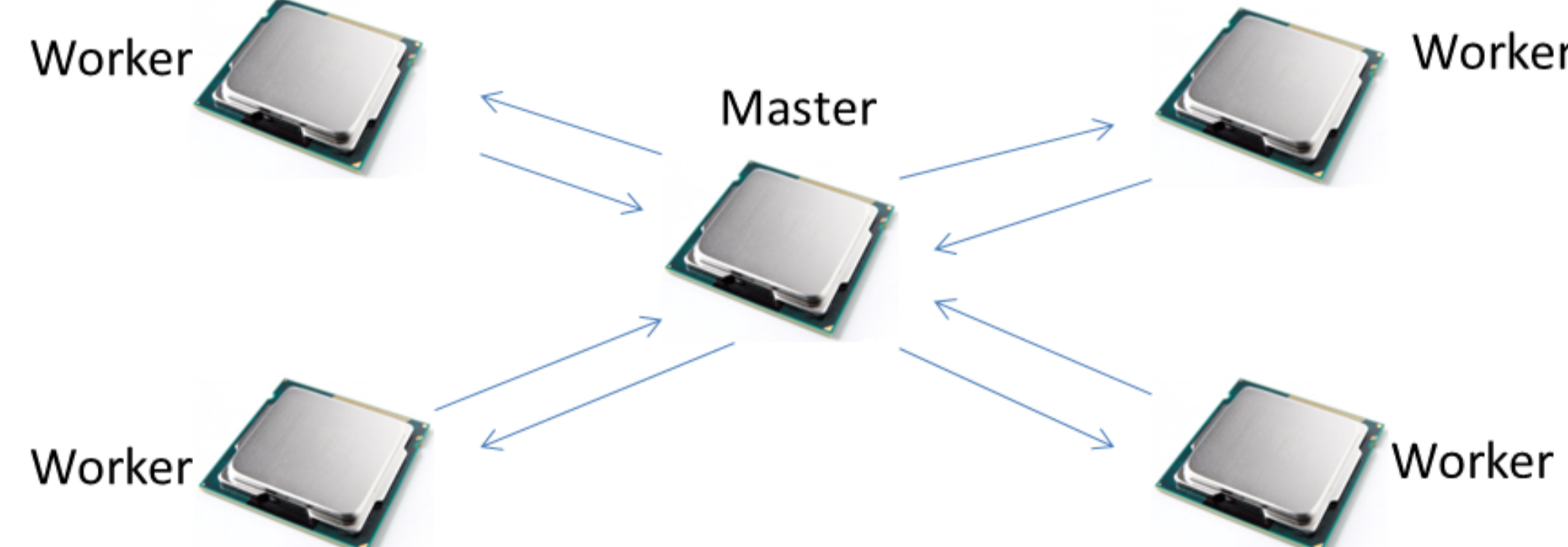
No



Counterexample

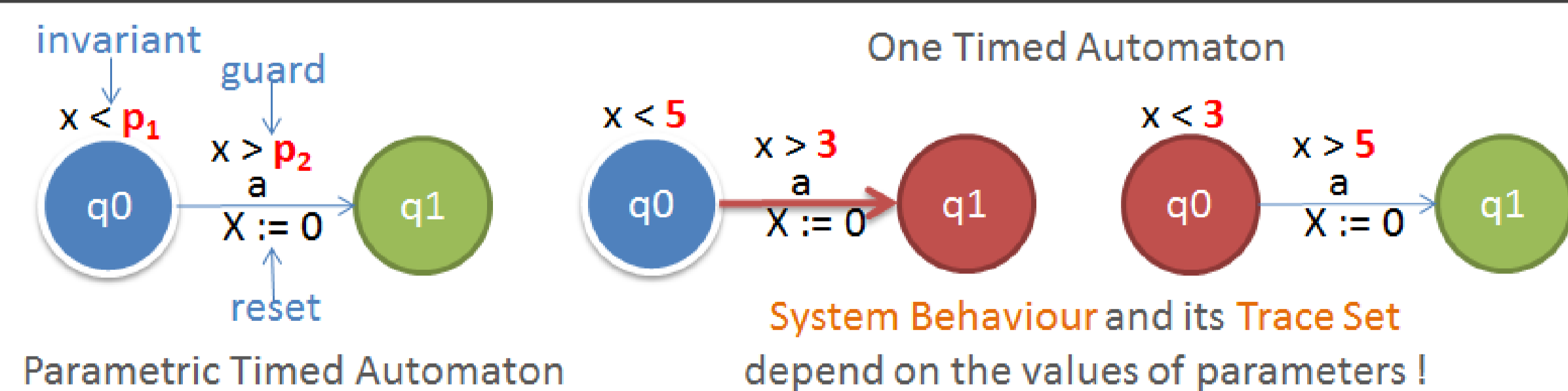
2. Goal

- Verify real-time systems, modelled by parametric timed automata. Take advantage of high-performance distributed computing for faster verification



→ Design algorithms distributed on a cluster to perform faster
(Note: Most algorithms use a Master-Worker scheme)

3. System model: Parametric timed automata



- A formalism to model and verify concurrent real-time systems [AHV93].
- 1 Parametric timed automaton (parameter) \leftrightarrow n. Timed automata (concrete value)

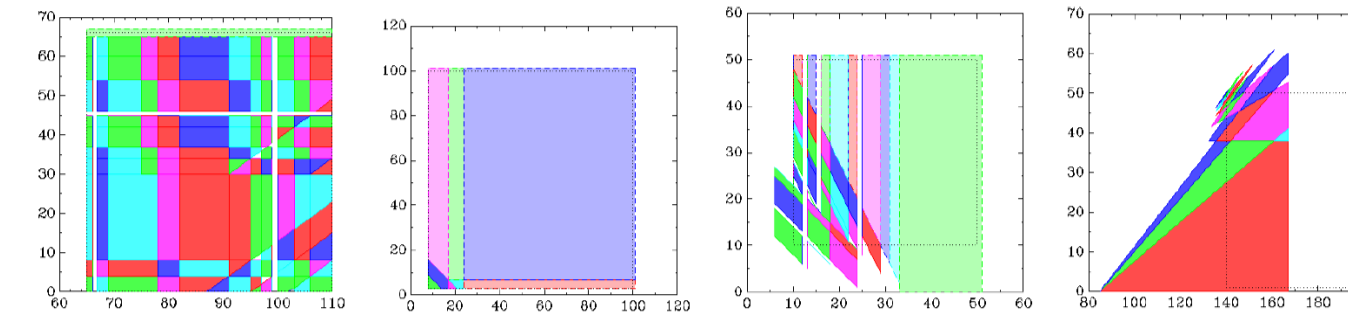
x: Clock

p: Parameters allow to represent unknown values (e.g. a transmission delay or a timeout)
Trace set:

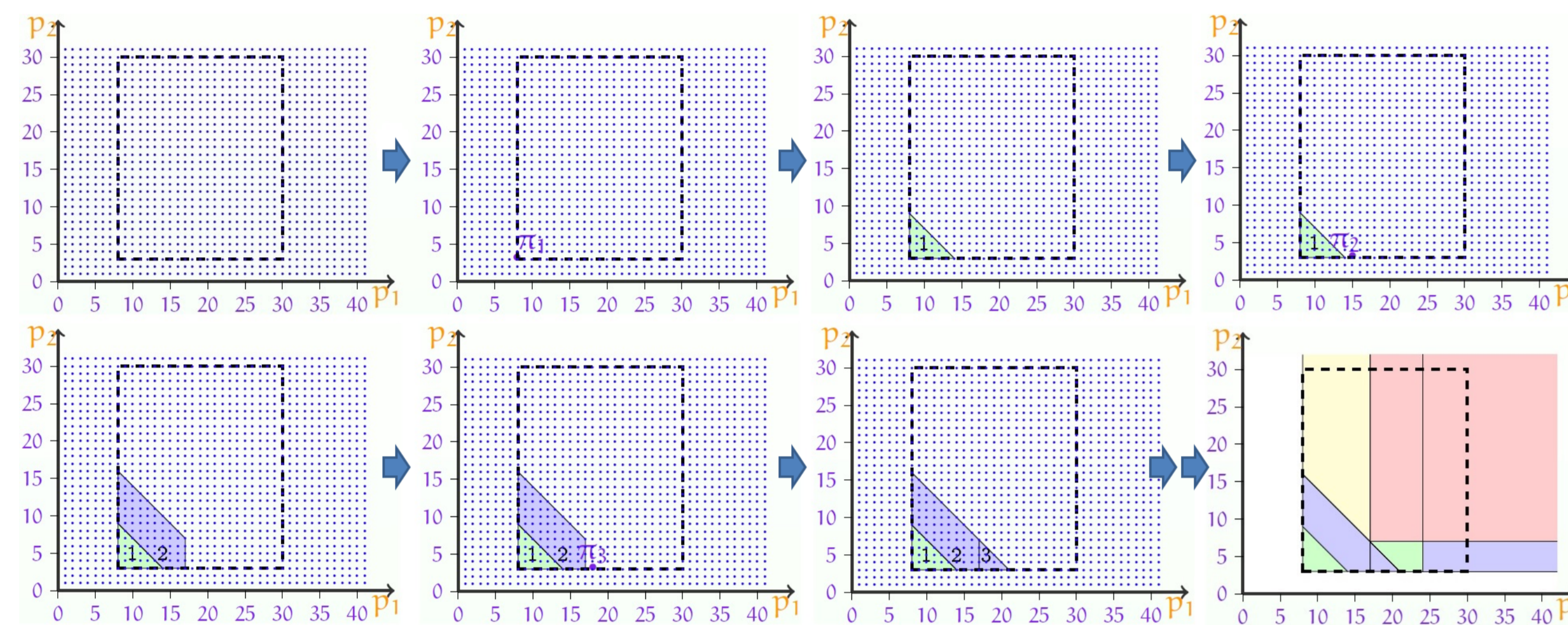
- Set of all sequences of (untimed) actions could be explored in the model
- This is also a specific behavior of the system or a subpart of cartography in the next part

4. Checking algorithm: Behavioural cartography

- Exhibit all subparts of the parameter space (system behaviours) (i.e. dense sets of parameter values of the parametric timed automata) [AF10]



- Easily check a certain value or a certain trace set for a certain behaviour

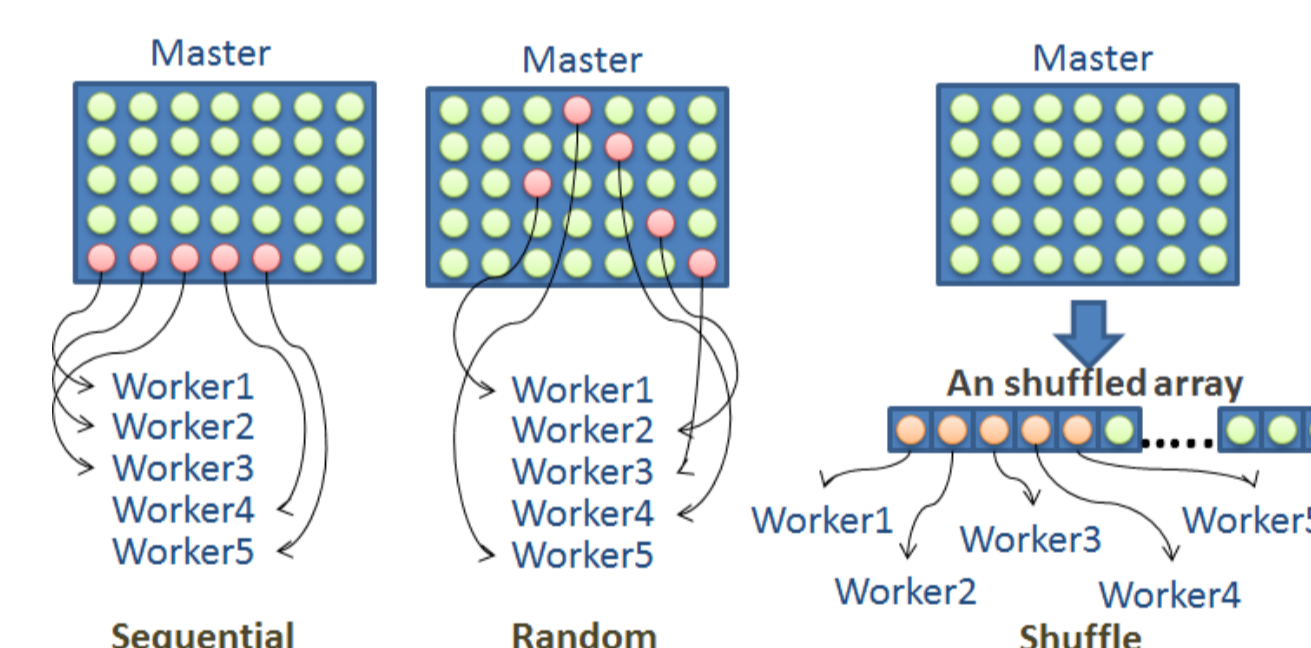


Method: Enumerate integer points and generate a tile (use the Inverse Method [ACEF09]). All points in a same tile have the same possible behaviours

5. Distribution: High performance distributed algorithms Checking multi values at the same time

Solution 1: Point-based distribution

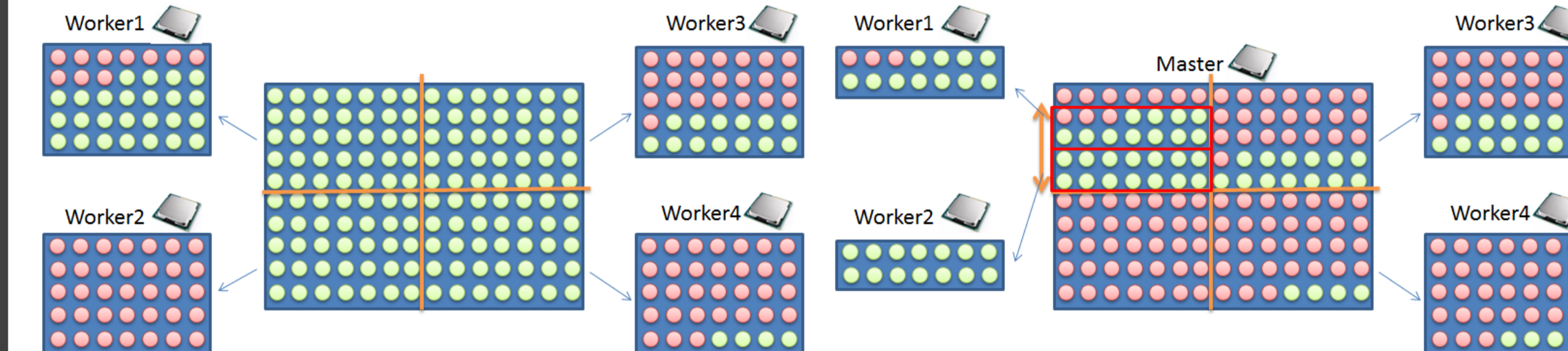
Master sends all the individual points to the Workers [ACE14, ALN15]. 3 algorithms:



1. **Sequential:** Each point is sent to a worker sequentially
2. **Random:** Points are selected randomly, then switch to Sequential
3. **Shuffle:** Similar to Sequential, but the master must statically compute the array of all points, then shuffle all points, then store them back in the array

Solution 2: Region-based distribution

Each process is in charge of a set of points (subdomain) [ALN15]



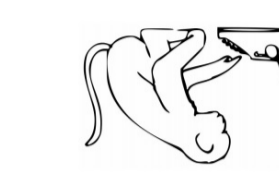
Static:

- One of the processes splits the domain, then sends to other processes and gathers the results of all processes
- **Drawback:** No load balancing although the workload is irregular

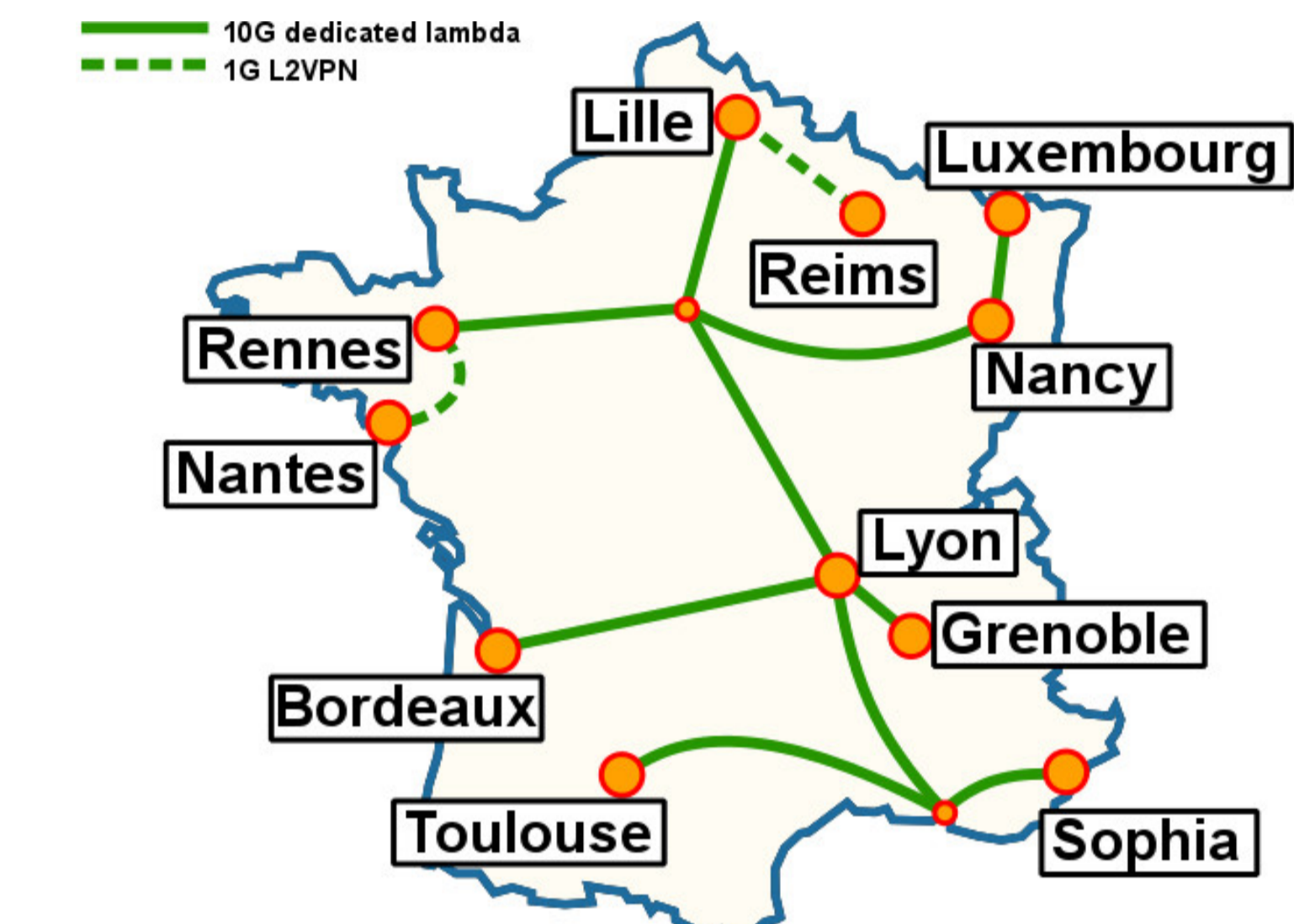
Dynamic:

- A Master is solely responsible for gathering tiles and splitting domain/subdomains
- The master monitors the progress of all workers: it can balance workload (by splitting) between workers

6. Implementation



- **IMITATOR** [André, Fribourg, Kühne, Soulat, 2012] : Parameter synthesis tool for real-time systems
- **OCaml:** All algorithms implemented in the OCaml language
- **MPI:** Using the OcamlMPI library bindings on top of Open MPI for message-passing between processes
- **Grid'5000:** homogeneous cluster featuring various technologies. Experiments conducted on 2 real clusters: **Pastel** (Toulouse, FR) 140 4-core nodes and **Griffon** (Nancy, FR) 92 8-core nodes



Experiments

The case studies are a flip-flop circuit, a root contention protocol, some tasks scheduling problems and a networked automation system:

Case study	Flip-flop4	RCP	Sched3-2	Sched3B-2	Sched3B-3	Sched5	SiMoP
Execution time(s)							
Static	33.0	2108.0	4.0	26.6	181.0	213.0	21.4
Seq	2059.0	653.0	4.6	11.0	810.0	219.0	36.1
Random	652.0	635.0	3.6	8.4	524.0	148.0	23.6
Shuffle	670.0	624.0	3.1	7.6	243.0	140.0	18.7
Subdomain	48.0	1286.0	7.2	15.8	217.0	273.0	32.4
Subdomain + H	24.0	622.0	4.0	11.0	81.0	199.0	23.2
Hybrid	24.0	624.0	3.1	7.6	81.0	140.0	18.7

Hybrid: switch between Subdomain + H (≥ 100.000 points) and Shuffle (< 100.000 points)

7. Conclusion

- Proposed a new efficient distributed algorithm + Heuristic for Behavioural Cartography
- Implemented the new algorithms in IMITATOR

Future works

- Design an fully distributed scheme for BC (No Master!)
- Try BC in GPU's or CPU+GPU's environments
- Formally prove the deadlock-freeness of our master-worker communication scheme

References

- [ACE14] Étienne André, Camille Coti, and Sami Evangelista. Distributed behavioral cartography of timed automata. In Jack Dongarra, Yutaka Ishikawa, and Hori Atsushi, editors, *21st European MPI Users' Group Meeting (EuroMPI/ASIA'14)*, pages 109–114. ACM, September 2014.
- [ACEF09] Étienne André, Thomas Chatain, Emmanuelle Encrenaz, and Laurent Fribourg. An inverse method for parametric timed automata. *IJFCS*, 20(5):819–836, 2009.
- [AF10] Étienne André and Laurent Fribourg. Behavioral cartography of timed automata. In *RP*, volume 6227 of *Lecture Notes in Computer Science*, pages 76–90. Springer, 2010.
- [AFKS12] Étienne André, Laurent Fribourg, Ulrich Kühne, and Romain Soulat. IMITATOR 2.5: A tool for analyzing robustness in scheduling problems. In *FM*, volume 7436 of *Lecture Notes in Computer Science*, pages 33–36. Springer, 2012.
- [AHV93] Rajeev Alur, Thomas A. Henzinger, and Moshe Y. Vardi. Parametric real-time reasoning. In *STOC*, pages 592–601. ACM, 1993.
- [ALN15] Étienne André, Camille Lipari, Coti, and Hoang Gia Nguyen. Enhanced distributed behavioral cartography of parametric timed automata. In Michael Butler and Sylvain Conchon, editors, *Proceedings of the 7th ICFEM'15, Springer LNCS 9407, November 2015*. Springer, 2015.
- [ALNS15] Étienne André, Giuseppe Lipari, Hoang Gia Nguyen, and Youcheng Sun. Reachability preservation based parameter synthesis for timed automata. In Klaus Havelund, Gerard Holzmann, and Rajeev Joshi, editors, *Proceedings of the 7th NASA Formal Methods Symposium (NFM'15)*, volume 9058 of *Lecture Notes in Computer Science*, pages 50–65. Springer, April 2015.