Behind the Scene of the Model Checking Contest, Analysis of Results from 2018 to 2023

Nicolas Amat[1], Elvio Amparore[2], Bernard Berthomieu[3], Pierre Bouvier[4,5], Silvano Dal Zilio[3], Francis Hulin-Hubard[6], Peter G. Jensen[7], Loig Jezequel[8], Fabrice Kordon[6,⋆], Shuo Li[9], Emmanuel Paviot-Adet[6,10], Laure Petrucci[11], Jiří Srba[7], Yann Thierry-Mieg[6], and Karsten Wolf[12]

[1] IMDEA Software Institute, Madrid, Spain
[2] Università di Torino, Turin, Italy
[3] LAAS-CNRS, Université de Toulouse, CNRS, Toulouse, France
[4] Univ. Grenoble Alpes, Inria, CNRS, Grenoble INP, LIG, France
[5] Kalray S.A, Montbonnot-Saint-Martin, France
[6] Sorbonne Université, CNRS, LIP6, F-75005 Paris, France
fabrice.kordon@lip6.fr
[7] Aalborg University, Denmark
[8] Université de Nantes, LS2N, UMR CNRS 6004, F-44321 Nantes, France
[9] Tongji University, Shanghai, China
[10] Université Paris-Cité, F-75006 Paris, France
[11] Univ. Sorbonne Paris Nord, CNRS, LIPN, F-93430 Villetaneuse, France
[12] University of Rostock, Germany

Abstract. This paper takes you behind the scenes of the Model Checking Contest (MCC), an annual competition focusing on the behavioral analysis of asynchronous systems using state-space exploration and model checking techniques. The MCC is part of a thriving group of scientific events dedicated to provide a fair evaluation of formal analysis tools, with the goal to push forward the state of the art and provide insights into the evolution of the involved technologies and approaches. We give details on the organization of the competition and on the ways we manage models and formulas. We also take a look at the evolution of the results over the 2018—2023 period, using the wide variety of data we collect each year, and report on the impact the MCC had on the competing tools.

1 Introduction

The Model Checking Contest (MCC) is an annual competition that benchmarks and compares model checking tools. Since its inception in 2011, the MCC has consistently been held alongside the Petri nets conference. However, for the second time, the 2023 edition is a part of the TOOLympics, held under the umbrella of the European Joint Conferences on Theory and Practice of Software (ETAPS).

Scientific competitions like the MCC play a crucial role in the progress of their respective fields. These competitions improve reproducibility and provide a basis for comparing various techniques. They also encourage the development of mature tools,

⋆ Corresponding author

a trend exemplified by the SAT, SMT, and recently, the SV competitions. The MCC is no exception, pushing tool developers to innovate and optimize their tools for better performance.

The MCC focuses on the model checking of concurrent systems. The concurrency in system semantics is an essential aspect of modern computing, given the prevalence of multicore and distributed CPU architectures. Petri nets, with their well-grounded theory and wealth of structural results, offer an ideal formalism for capturing this concurrency and the associated complexity.

In the MCC, model checking seeks to determine if a system satisfies a given property. This could range from safety conditions or invariants to more complex specifications expressed in Computation Tree Logic (CTL) or Linear Temporal Logic (LTL) [14]. The contest also examines global properties such as deadlock detection or liveness, along with metrics on the state space and place markings bounds. These examinations are not purely academic; they reflect practical questions that users might pose about their systems.

Over its decade-long run, the MCC has established itself as a reputable and mature competition. The accumulated data across these years presents a rich source for tracking the evolution and progression of model checking tools and techniques. This paper dives into this data, presenting insights and analyses that shed light on the state of model checking today.

The goal of this paper is to provide an overview on this event as it was operated in 2023 (after more than a decade of experience), to discuss results of the 2023 edition and to present some observations based on the results for the period 2018-2023.

The paper is structured as follows: section 2 sets the definitions of the terms we use and the categories of the MCC; section 3 presents an in depth analysis of the models used as benchmarks and presents how models are collected; section 4 presents the process used to generate formulas that are used as queries in LTL or CTL examinations; section 5 contains subsections for each tool that participated recently (since 2021), with some insight on their strengths and weaknesses provided by the respective tool authors; section 6 presents the result of the 2023 edition of the MCC; section 7 presents a retrospective analysis of the evolution of the contest and its results since 2018 and section 8 concludes the paper with some perspectives for improving future editions of the MCC.

2 Main Definitions within the Model Checking Contest

Let us first define the vocabulary and concepts used in the Model Checking Contest and in this paper.[13]

Tools. The primary aim of the Model Checking Contest is the evaluation of various tools. Developers submit their tools to contend for medals across diverse categories. While tools can be submitted in multiple configurations or variants, they are deemed as part of a single "family". Within this family, only the top-performing variant vies for a

[13] More details can also be found in the 2019 report on the MCC [62].

podium position, i.e., being among the top three. However, scores of other variants and those of "reference tools" (see section 6.1) are shared for informational purposes only.

Models and Model Instances. Every participating tool in a specific MCC edition is evaluated against a consistent benchmark suite. The community updates this benchmark annually, comprising **models** and **model instances**:

- A **model** symbolizes either an academic or industrial problem, modeled using (possibly Colored) Petri nets and provided in the PNML standard format [51].
- Each model may have multiple **model instances** which are variants with differing configurations. These changes typically include scaling the initial marking, changes in the structure and scaling the definition of color domains for colored nets. This lets us assess how tools behave when the model is scaled up in complexity.

Some models are simply Petri nets (abbreviated as PT nets in this paper standing for Place Transitions Nets [71]) with integer arc weights and place markings. Other models are colored nets expressed as Symmetric Petri nets [35].

When models are provided as colored nets, the MCC also provides an equivalent PT net computed using the unfolding technique [61]. Some colored model instances however do not have such an equivalent PT net because the resulting net is too large (see Sect. 3 for details).

Each year, the community contributes with new models that are called "surprise models". Model instances from previous years are kept in the benchmark as "known models".

Examinations. Tools are tasked with processing **examinations** on the **model instances**. An examination encompasses one or more queries that yield either Boolean or numeric outcomes. If a tool is unable to resolve a query or examination, it can respond with "do not compete" or "cannot compute".

For 2023, the following six examinations were:

- **StateSpace** examination, where the tool must generate the state space of the system and provide four metrics on the state space of the Petri net: the number of nodes (states) and edges (transitions) in the marking graph, the bound on the number of tokens in any given place, the bound on the total number of tokens in any given reachable state. This category is the oldest category of the MCC, it was introduced in the very first edition of the MCC in 2011.
- **Global Properties** examination, where the tool must answer by a Boolean whether the model instance satisfies five global properties: can the net deadlock, are some place markings invariant (stable marking), is the net one-safe, is it quasi-live (are all transitions fireable at least once) and is the net live. These global properties are all model specific, and while they could be expressed as a set of reachability or CTL queries (one per place or transition) there exist some more effective strategies that can be applied such as structural reductions. While Reachability Deadlock was introduced in 2013, the four other queries were only introduced in 2020.

- **Upper Bounds** examination, where the tool must provide the maximum number of tokens that can mark a given set of places in any reachable marking. There are 16 such queries per model instance and the queries are generated randomly each year.
- **Reachability** examination, where the tool must provide a Boolean answer to queries that are either an invariant (AGp) or a test for reachability of a given situation (EFp). The predicate p is a Boolean combination of comparisons between markings of places (for Cardinality) or fireability of certain transitions (Fireability). [14] The reachability problem is a core problem in model-checking, and is known to be decidable for Petri nets (despite unbounded places). The Reachability examination was introduced in the very first MCC in 2011. It is an examination that typically has more participants than any other, and is thus a highly contested category.
- **CTL** examination, where the tool must provide a Boolean answer to queries that are expressed in Computation Tree Logic (CTL). Again 16 formulas are cardinality based and 16 are fireability based. The category was introduced in 2013.
- **LTL** examination, where the tool must provide a Boolean answer to queries that are expressed as a Linear-time Temporal Logic (LTL) formula. Again 16 formulas are cardinality based and 16 are fireability based. The category was introduced in 2013.

For the Upper Bounds, Reachability, CTL and LTL examinations a new set of formulas is generated every year, so even if the model is a "known model" the formulas are always unknown to the competitors.

Runs. A "run" means a tool's execution applied to a specific model instance for a designated examination. For instance, computing the 16 Cardinality-based CTL formulas for the model instance named Philosophers-PT-001000 constitutes a run. Every run is allotted a maximum of 1 hour, except for the Global Properties, which are given half that time for each of the five queries. These runs are constrained to utilize up to 4 CPU cores and 16 GB of RAM.

To ensure uniform comparison metrics such as execution time and memory usage, all tools for a given model instance/examination combination are run on the same machine, despite parallel evaluations across multiple computers or clusters.

3 The Models in the Benchmark

The models in the MCC benchmark are enriched every year with new models provided by the community in reply to a "Call for Models", but models from previous years are kept in the benchmark. In this section we provide an analysis of the models used in the 2023 edition of the Model Checking Contest and their evolution since 2018.

3.1 Evolution of Models between 2018 and 2023

The MCC benchmark is composed of both colored nets (COL) and place transition nets (PT). Each model comes with a description that explains its origins and the nature of the problem that it tries to capture.

[14] In a given run of a tool, 16 such queries are submitted. So with 16 Cardinality queries and 16 Fireability queries we have 32 reachability queries per model instance in total.

	Models			Instances				
					PT			
Year	PT	COL	Total	COL	Native	From COL	Total	Total
2018	69	20	89	180	614	153	767	947
2019	71	22	93	193	666	159	825	1 018
2020	80	23	103	213	837	179	1 016	1 229
2021	90	24	114	230	985	196	1 181	1 411
2022	103	24	127	230	1 191	196	1 387	1 617
2023	105	27	132	252	1 208	218	1 426	1 678

Fig. 1: Evolution of the number of models and model instances in the MCC benchmark.

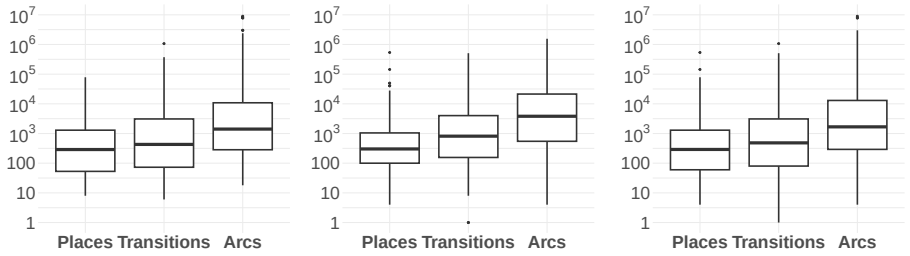
Table 1 shows there were in 2023 a total of 132 models in the benchmark, up from 89 in 2018. From these models, a larger number of instances is produced thanks to their parameters. We reach to a total of 1 678 instances in 2023. While the MCC provides equivalent PT nets for COL specifications, some larger scalable COL instances cannot be unfolded due to the explosion in the size of the resulting PT net. Note that there are some COL model instances without a corresponding PT net (e.g. in 2023, only 218 out of 252 COL instances had an associated "PT from COL" version).

3.2 Analysis of the Benchmark Models in 2023

So far, the 132 models featured by the MCC have been submitted by 60 different authors, from 12 countries. This number of contributors leads to a wide variety of models, whether in terms of their origins or the way they are built. Since the MCC benchmark currently covers a wide scope of models, we provide here a broad classification of the models according to 14 application domains. For each domain, we also give (in brackets) its numbers of colored models, PT models and models designed within the context of industrial projects.

- **Biology and Chemistry** (11 PT – 1 industrial): Angiogenesis, CircadianClock, Diffusion2D, DNAWalker, EGFr, ERK, GPPP, MAPK, MAPKbis, PaceMaker, PhaseVariation, ViralEpidemic.
- **Business Process and Automation** (3 COL, 15 PT – 7 industrial): BugTracking, BusinessProcesses, CryptoMiner, FamilyReunion, FMS, HealthRecord, HospitalTriage, HouseConstruction, IBM (4 models), Kanban, Medical, ProductionCell, ParamProductionCell, RobotManipulation, UtilityControlRoom.
- **Distributed Memory and Related Algorithms** (2 COL, 6 PT – 1 industrial): CANConstruction, CANInsertWithFailure, LeafsetExtension, MultiCrashLeafsetExtension, QuasiCertifProtocol, SatelliteMemory, SharedMemory, StigmergyCommit.
- **Elections or Consensus** (1 COL, 4 PT – 2 industrial): Election2020, HirschbergSinclair, NeoElection, Raft, StigmergyElection.
- **Games** (1 COL, 4 PT): DLCRound, DLCShifumi, NQueens, Solitaire, Sudoku.
- **Hardware** (2 COL, 8 PT – 5 industrial): ARMCacheCoherence, ASLink, DiscoveryGPU, GPUForwardProgress, NoC3x3, Ring, SafeBus, TokenRing, UtahNoC, Vasy2003.

- **Operating Systems or Middleware** (2 COL, 2 PT): PolyORBLF, PolyORBNT, SimpleLoadBalancer, SmallOperatingSystem.
- **IoT, Cloud, Reconfiguration** (5 PT – 3 industrial): CloudDeployment, CloudOpsManagement, CloudReconfiguration, Planning, SmartHome.
- **Mutual Exclusion** (6 COL, 10 PT): Anderson, DatabaseWithMutex, Dekker, DoubleLock, EisenbergMcGuire, FunctionPointer, GlobalResAllocation, LamportFastMutEx, Peterson, Philosophers, PhilosophersDyn, ResAllocation, RwMutex, SwimmingPool, Szymanski, TwoPhaseLocking.
- **Network Protocols** (3 COL, 9 PT): CSRepetitions, Echo, HexagonalGrid, HypercubeGrid, HypertorusGrid, IOTPpurchase, NeighborGrid, PermAdmissibility, SquareGrid, TCPcondis, TriangularGrid, VehicularWifi.
- **Security** (7 PT – 7 industrial): DES, ShieldIIPs, ShieldIIPt, ShieldPPPs, ShieldPPPt, ShieldRVs, ShieldRVt.
- **Synchronisations and Message Passing** (9 PT): ClientsAndServers, DBSingleClientW, DLCflexbar, FlexibleBarrier, MultiwaySync, RingSingleMessageInMbox, SemanticWebServices, ServersAndClients, SieveSingleMsgMbox.
- **Academic and Synthetic Models** (4 COL, 9 PT): DoubleExponent, Eratosthenes, DrinkVendingMachine, JoinFreeModules, Murphy, PGCD, Referendum, RefineWMG, RERS (4 models).
- **Transportation Systems** (3 COL, 6 PT – 3 industrial) AirplaneLD, AutoFlight, AutonomousCar, BART, BridgeAndVehicles, CircularTrains, EnergyBus, Parking, Railroad.



(a) Native PT Models (b) PT from COL Models (c) All PT Models

Fig. 2: Log scale box plots of Places, Transitions, and Arcs for PT, unfolded PT from colored models, and all PT nets (including unfolded). The central line of the box represents the median, while the box itself spans the interquartile range (IQR) from the first to the third quartile, covering the middle 50% of the values. Whiskers stretch out to 1.5 times the IQR. Any values exceeding the whiskers are considered outliers and depicted as individual points.

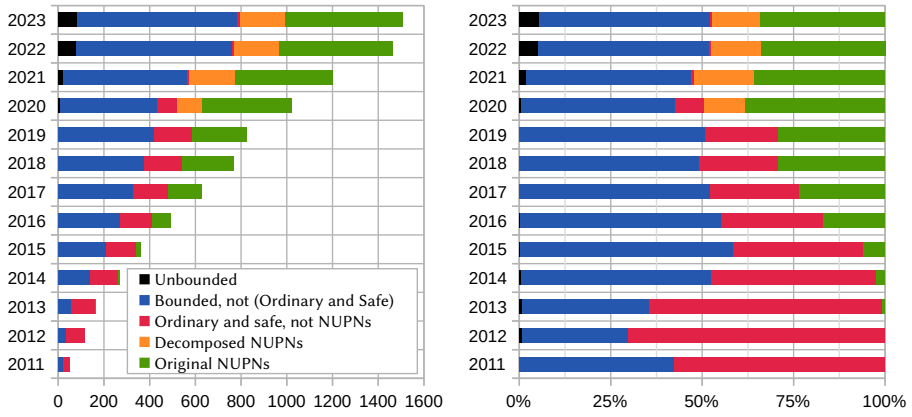
In this section, we delve into the core characteristics of PT model instances in the benchmark, visualized through box plots. Figure 2 showcases the distribution for three principal metrics in the MCC 2023 models: the count of arcs, places, and transitions. Given the significant variance in model sizes, a logarithmic scale was adopted.

For clarity, we have excluded COL model sizes from this visualization, as their pre-unfolding sizes are not directly comparable to PT net dimensions (e.g. whiskers range up to 100 places or transitions at most).

We separate the PT models into "native PT" (Fig. 2a) that were provided in this format, and "PT from COL" (Fig. 2b) that are produced from the COL models up to a certain size. The rightmost plot Fig. 2c merges both of these subcategories. While these plots show that on average models unfolded from COL are larger than native PT nets (in number of arcs), their structure can be expected to be more symmetric, and the largest PT instances in the benchmark are native PT.

The larger PT models in the MCC thus contain up to 10^4 places (with some outliers reaching 10^5 or more), up to 10^5 transitions (with some outliers reaching 10^6), and up to a few million arcs. These plots illustrate the wide variability in the structure size and complexity of PT models.

3.3 A Study of PT Model Instances



(a) Absolute Numbers (b) Normalised Percentages

Fig. 3: Distribution of the types of instances provided for each edition of the competition.

Figure 3 offers a yearly breakdown of all PT instances since the MCC's inception, into five mutually exclusives categories. From 2011 to 2023, the first two categories, presented below, accounted for 42% to 58% of PT instances, stabilizing around 50% after 2014:

– Unbounded nets (Black): These nets have an infinite number of reachable markings because of unbounded places, where tokens can accumulate indefinitely. Though they were exceedingly rare before 2020, they now represent 5.4% of all PT instances.

– Bounded, but non-ordinary or non-safe nets (Blue): These are bounded nets that are not ordinary, meaning arc weights can exceed 1, or that are not one-safe, meaning there is at least one reachable marking having several tokens in the same place[15].

The remaining instances are ordinary and one-safe nets, which means they can be expressed as NUPNs (for "Nested-Unit Petri Nets" [45]), an extension of Petri Nets bringing them modularity and hierarchy through a structure of sequential processes (called units), nested in the form of a tree, representing their parent-child relationships. MCC instances are provided in the PNML file format, which enables this NUPN information to be provided through an additional "tool specific" section[16]. These instances are divided into the following three categories:

– Original NUPNs (Green): These are NUPNs stemming from models written in high-level specification languages, which feature the concept of concurrent processes. These processes have been directly translated into NUPN units. In 2023, they constitute 72% of the NUPN instances.
– Decomposed NUPNs (Yellow): These models had their NUPN structures inferred and appended retrospectively, using the 22 decomposition approaches of nets into networks of automata [31], which internally leverage SAT solvers, SMT solvers, and tools for graph coloring and finding maximum cliques[17]. To determine, for a given Petri Net, which NUPN structures are possible, these decomposition approaches depend on the efficient computation of the concurrent place relation [29]. Since 2021, a striking 95.2% of the ordinary and safe instances, lacking an initial NUPN structure, have been upgraded in this manner.
– Non-NUPNs (Red): These nets are ordinary and safe, but they are not described using NUPNs. As of 2021, only ten instances of this category remain, owing to various reasons such as the presence of only trivial decomposition possibilities[18], or their extensive sizes (at present, no participing tool can handle such instances).

4 Formulas in the Benchmark

4.1 The Need and Purpose for Formula Generation

The MCC faces a dichotomy. Whereas models are handcrafted and reused from one year to the next, formulas are randomly generated and changed for each edition of the competition.

In model checking, formulas define the properties of systems, setting clear criteria on the behaviors and conditions a system must meet or avoid. However, even though the

[15] It should be noted that some instances are not ordinary, yet are safe, because their initial markings are safe, and all their non-ordinary arcs are connected to dead transitions.

[16] See https://mcc.lip6.fr/nupn.php and https://cadp.inria.fr/man/nupn.html for further details about NUPN file formats.

[17] Notice that subproblems invoked by these decompositions have been used to provide benchmark for the Model Counting Competition [49,28], for the SAT Competition [50,30], and for the SMT-Comp [22,27].

[18] Which provide no additional information to their underlying Petri nets, see [45, Prop. 11].

model forms allow for the declaration of behavioral properties, very few models come with any predefined formulas, and none of the models come with formulas covering all the examinations of the MCC. This poses a challenge that was solved by relying on an automated process for generating new formulas, under the supervision of a dedicated "formula board".

There are some benefits to this situation. Although custom formulas might better align with the model's purpose, a static set of such formulas risks tool over-fitting for known properties. This regular update ensures model checking tools face novel challenges, fostering innovation and avoiding tool stagnation. Notably, the MCC requires 32 formulas for each model instance that span across reachability, CTL, and LTL examinations. This led to the production of 90 912 formulas in 2018, and this number increased to 161 088 by 2023.

Next, we describe the process used for the generation of formulas for the Reachability, CTL, and LTL examinations. While the Upper Bounds examination also utilizes formulas, its generation approach is direct, primarily consisting of selecting places from a net without repetition.

Generation of Reachability and CTL formulas using Citili. Reachability and CTL formulas are generated using the tool Citili[19] since 2020. For a given model, Citili proceeds as follows in order to generate one formula.

1. Generate a generic CTL (or Reachability) formula with abstract atoms by randomly selecting operators from a set of allowed operators, up to a given depth.
2. Verify that the formula is acceptable by analyzing its syntax, otherwise generate another one:
   - ensure that it is not in another class of formulas (e.g. it has tree operators for a CTL formula)
   - perform limited checks for triviality of the formula (e.g. tautologies)
3. Instantiate each abstract atom with a concrete one corresponding to the type of formula we want (Cardinality or Fireability).

Building on this, Citili employs the subsequent methodology to generate a challenging formula set of a predefined size (currently set at 16):

1. Generate an initial batch of formulas (currently 32).
2. Set up a rudimentary model checker for each formula:
   - Limit exploration to a set number of states (presently 2 000).
   - Keep formulas that do not produce a result and classify them as challenging.
3. If the stipulated time has not been exhausted and the challenging formula set is below the target size (currently 16), the process returns to step 1.
4. When the time threshold is reached, Citili generates additional formulas to populate the set of challenging formulas. This additional set may include trivial formulas, since they are not subject to any filtering.

[19] Available from https://github.com/mcc-petrinets/citili

Generation of LTL Formulas with the Aid of Spot. The inclusion of a diverse set of LTL formulas is essential to provide a thorough, representative benchmark. In their seminal work, Manna and Pnueli introduced a categorization of LTL formulas, identifying six distinct categories: Reactivity, Recurrence, Persistence, Obligation, Safety, and Guarantee [66]. A balanced representation of these categories ensures that overfitting is avoided and provides a broad coverage of the temporal behaviors in the examination.

Since 2019, we use the following two tools from the SPOT platform [43], in the MCC, to achieve balanced and non-trivial formula generation:

- **ltlrand**: This tool generates random LTL formulas using a predefined set of LTL operators and a designated formula depth. Notably, ltlrand ensures the generated formulas are not trivially reducible, enhancing the complexity of the evaluation process.
- **ltlfilt**: Once the formulas are procured using ltlrand, ltlfilt categorizes each formula based on Manna and Pnueli's classification. This classification step ensures the formulas offer a well-rounded representation across all LTL categories.

Currently, there is no mechanism akin to a "trivial model checker" to evaluate the generated LTL formulas on the first few states of a model instance. However, such an addition can be considered for future improvements, potentially enhancing the quality of formula generation further.

4.2 Analysis of Formulas in 2023

We study in this section a classification of the properties of the MCC into those that can be disproved by exhibiting a counter-example (CEX), and those that must be proved to hold over all reachable configurations or paths (INV). This classification reflects the one for SAT/SMT competitions [50] into SAT (and a model is provided, corresponding to CEX) or UNSAT (which corresponds to INV in our classification).

We study this classification for Global Properties, Reachability and LTL properties. These notions are not applicable to the other categories (CTL, Upper Bounds, State Space). We can only classify a property as being INV or CEX if there is a verdict of at least one tool, so properties no tool could answer are left as Unknown (UNK).

- For "OneSafe" CEX corresponds to existence of a place whose marking can exceed 1,
- for "ReachabilityDeadlock" CEX corresponds to existence of a deadlocked state,
- for "StableMarking" CEX corresponds to existence of a place whose marking never varies,
- for "QuasiLiveness" CEX corresponds to existence of a transition that can never be fired,
- for "Liveness" CEX corresponds to existence of a transition that is not live.
- For "Reachability" properties, an "AG (p)" formula (an invariant that must be true of all states) is a CEX if it is false, and an "EF (p)" formula (a test to see if we can reach a state satisfying p) is a CEX if it is true.
- For "LTL" properties, a false property is a CEX (and we can exhibit a run that does not satisfy the property) and a true property is an INV.

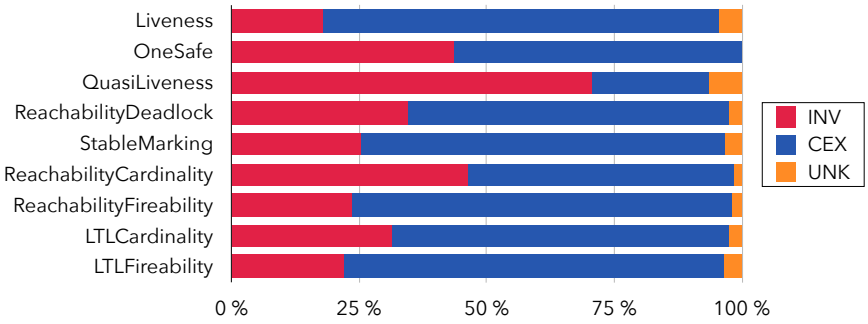


Fig. 4: Distribution of INV vs CEX properties across the different examinations; UNK corresponds to the situation where no tool answered (thus, classification is unknown).

Figure 4 presents the distribution of INV vs CEX for the relevant examinations.

For global properties, the result is purely model-dependent, and in some cases, even instance-dependent. It shows that 56% of the models are one-safe, and that 63% of models contain a deadlock. The models in majority (71%) contain stable places, but most of them (70%) do not contain dead transitions. Only 17% of the models are known to be live, probably due to the abundance of workflow like nets with start and end activities, and models featuring some non repeatable initialization step.

For both Reachability and LTL, there is a clear bias overall for CEX, i.e. properties that can be disproved with a single trace or state. The bias towards CEX is much stronger on Fireability queries (74% of CEX in both reachability and LTL) than on Cardinality queries (52% of CEX for reachability and 65% for LTL). The distribution does evolve from year to year since the properties are generated, but only by a few percent so that qualitatively these observations hold since 2018.

The overall bias towards CEX might skew the results of MCC in favor of tools efficient at bug finding (e.g. directed walks) with respect to strategies that are better at *proving* the system correct (INV), although this is typically a harder problem to solve.

5 Participating Tools

This section presents in more detail the tools that have participated in the MCC since 2021: LoLA, SMPT, TINA.tedd, Enpac, ITS-Tools, TAPAAL, GreatSPN.

5.1 EnPAC

Overview and Evolution EnPAC (Enhanced Petri-net Analyser and Checker) is a model-checking tool for large concurrent systems modeled as PT nets or their colored extension (COL models). It can evaluate arbitrary queries specified in linear temporal logic (LTL). We started at the end of 2018 and developed an initial version to participate in MCC'2019, which initially supported PT nets. From 2019 to 2020, we extended its capability to colored Petri nets. We directly analyze colored nets without unfolding them

to PT nets, which is different from other tools. And we investigated several existing algorithms to improve efficiency, of which the on-the-fly method [46] makes huge progress. From 2020 and 2021, we added encoding strategies (one-safe net encoding, NUPN encoding [45], and P-invariant encoding [85]), and we developed bitwise operations to read and write encoded states, which made encoding strategies more efficient. Additionally, we added heuristic information into Büchi automata so that the counterexample search always follows the shortest path to an acceptable state. From 2021 to 2022, we developed a method where fireable transitions are no longer stored in each state but dynamically generated each time to save memory further.

MCC Impact The biggest benefit of participating in MCC is that it allows us to know what our weaknesses are. For example, from the result of MCC'2020, we found that EnPAC generally computed faster than other tools but consumed much more memory. Therefore, in the following year, we focused on encoding strategies. We also investigated how to read and write an encoded state using only bit operations, which not only improved memory usage but also reduced the time penalty due to encoding. At present, we recognized the advantages of parallelism shown in MCC and are paying more attention to the sequential-to-parallel bottleneck to extend EnPAC to be parallel.

Availability and Contributors

- **Homepage URL:**
  1. https://github.com/Tj-Cong/EnPAC_2021 (for PT nets)
  2. https://github.com/Tj-Cong/EnPAC_CPN_2021 (for colored nets).
- **License Type:** MIT License.
- **Affiliation:** Tongji University, Shanghai, China.
- **Tool Authors:** Zhijun Ding, Cong He, Shuo Li.
- **Relevant Publications and Contributions:** [48]

5.2 GreatSPN

Overview and Evolution GreatSPN is an open source framework for Petri nets modeling and analysis. It has several tools for drawing, computing, and verifying different types of Petri nets, either through a graphical interface or through commands. One of the tools is *starMC* [11], a symbolic model checker that uses Multivalued Decision Diagrams (MDD) to generate the state space and check CTL/LTL/CTL* properties. The MDD data structure employed is developed separately in the Meddly library[20], developed at Iowa State university. GreatSPN supports several Petri net formats, including PT nets, generalized stochastic Petri nets, and colored Petri nets.

[20] https://github.com/asminer/meddly

MCC Impact The *starMC* model checker of GreatSPN has been adapted to compete in all categories of the MCC, albeit it does not have optimized implementations for all of them. The competition for which GreatSPN has the most advanced solution is *StateSpace* generation. The efficient generation of the symbolic reachability graph in GreatSPN is a combination of different factors:

- The representation adopts MDDs to encode the state space;
- The tool uses a specialized *saturation* algorithm with implicit transition relation firings to list all the reachable states;
- An effective heuristic for static variables reordering of the Petri net places in the MDD encoding is used; This heuristic is based on linear algebra properties [9] of the Petri net incidence matrix, and has proven to be highly general and scalable.

GreatSPN also performs model checking of CTL and CTL* properties. LTL expressions can also be checked as CTL* expression. This choice is however suboptimal, since it encounters a significant reduction in the performances.

Availability and Contributors

- **Homepage URL:** https://github.com/greatspn/SOURCES
- **License Type:** GPLv2.
- **Affiliation:** University of Torino, Italy.
- **Tool Authors:** Initially started by Giovanni Chiola, the framework has seen contributions from several developers over its long history. The current main contributors and maintainers are Elvio G. Amparore and Marco Beccuti.
- **Relevant Publications and Contributions:** See [8] for a recent description of the stochastic functionalities of the tool, and [11] for the symbolic model checking algorithms.

5.3 ITS-Tools

Overview and Evolution ITS-Tools [77] is a model-checker using a portfolio of diverse strategies that include symbolic methods based on hierarchical Set Decision Diagrams [81], constraint based reasoning using the SMT solver Z3 [69] to over-approximate the state space, fast pseudo-random walks to under-approximate it, partial order reduction leveraging the tool LTSMin [59,63], as well as advanced structural reduction rules to reduce the size of the system [79].

The engine has specific support for colored nets using skeleton over-approximations [83] where possible as well as symmetry aware unfoldings that can bypass the model size explosion due to large domains. The LTL engine benefits from several strategies unique to ITS-Tools such as length-sensitivity analysis [70]. For Global Properties and Upper Bounds ITS-Tools benefits from a dedicated set of strategies [80].

MCC Impact ITS-Tools currently participates in all categories of the MCC. It has participated since the very first edition in 2011 and almost continuously since. While until 2019 Petri net support was offered through a translation to Instantiable Transition Systems (ITS) [77] that give their name to the tool, dedicated Petri net support was gradually introduced to better compete in the MCC.

Availability and Contributors

- **Homepage URL:** https://github.com/lip6/ITSTools
- **License Type:** GPL, EPL
- **Affiliation:** LIP6, Sorbonne Université, CNRS
- **Tool Authors:** Yann Thierry-Mieg with other contributors
- **Relevant Publications and Contributions:** [77,79,70,80]

5.4 LoLA

Overview and Evolution LoLA (a **Lo**w **L**evel Petri net **A**nalyzer) is a model checker for place/transition Petri nets. Its was first released in 1998. LoLA can analyze the state space using stubborn set [82,74], symmetry [75], coverability graph [73], and sweep-line [36,76] reductions. Many formula classes are evaluated by tailored search routines and specific versions of reduction techniques [65,64]. Elements of Petri net theory are used for speeding up verification [84]. Before actual verification, both net and query are simplified using net reduction and abstraction methods [83], as well as Petri net structure theory. LoLA can run several alternative techniques on a query, organized by a portfolio manager [89].

LoLA started as a reachability checker that was soon extended with a CTL model checker. Later, an LTL model checker was added. Then, specific routines for many classes of CTL formulas followed. Most recently, a reduction engine, the portfolio manager, and a skeleton abstraction method became part of LoLA.

MCC Impact Before the MCC was launched, development of LoLA was mostly driven by our own research. LoLA was the vehicle to generate the "experimental results" table on the second but last page of every paper. LoLA was used in real applications as long as they saw the available techniques fit for their purpose. However, there was no substantial impact of applications to the performance of LoLA.

Through the MCC, we had the opportunity to earn scientific reputation for implementing methods invented by somebody else. This way, the tool grew much more mature and complete. The very competitive nature of the contest pushed us into implementing all the little tricks, and doing all the little optimizations that would have been difficult to publish in a purely theoretical paper. Being quite successful in the MCC, LoLA received a lot of additional attention. The MCC generates very productive exchange of thoughts between tool developers.

Availability and Contributors

- **Homepage URL:** https://theo.informatik.uni-rostock.de/theo-forschung/tools/lola/
- **License Type:** GPL
- **Affiliation:** Universität Rostock, Institut für Informatik, Germany
- **Tool Authors:** Main author is Karsten Wolf. Contributors are mentioned in the source code.
- **Relevant Publications and Contributions:** [87,86,88,85]

5.5 SMPT

Overview and Evolution SMPT (for Satisfiability Modulo Petri Net) is a model checker [4] that participates in the Reachability examination of the MCC since 2021. The tool started as a portfolio of methods to experiment with symbolic, SMT-based model checking techniques, and was designed to be easily extended. Some of its distinctive features are: an adaptation of the Property Directed Reachability (PDR) method for PT nets [5]; and its ability to generate verdict certificates for invariants. It also integrates an approach combining integer linear systems and structural reductions, like in TINA.tedd [16], but adapted to the verification of reachability properties. We give a formal definition of this approach in [2,7], where it is called *polyhedral reduction*, and show how it can be applied to different symbolic methods (BMC, k-induction and PDR), and to more general problems, such as computing the concurrency relation of safe nets [3,6].

At its core, SMPT acts as a front-end to SMT solvers. Since 2022, we added several structural methods, such as invariant checking based on the so-called "state equation method" or with the addition of extra constraints during the verification process, based on results from Petri net theory: structural invariants; traps; invariants originating from the NUPN specification; etc. We also started experimenting with methods based on random walks, which relies on a simulation tool called walk, part of the TINA toolbox [18]. SMPT was again improved in 2023 with the addition of a dedicated method [1] able to transform, in most cases, an initial query (a pair made of a model instance and a reachability formula) into an equivalent query on a reduced, simplified version of the model. It is also the first edition of the MCC where we experimented with formula simplification methods.

MCC Impact Our participation in the MCC had the effect of transforming what was supposed to be a simple prototype, used for experiments, into a standalone verification tool. It helped us better automate the use of our tool, adding options that simplify the use of SMPT by non-experts and that simplify building strategies to use our portfolio of methods more effectively. It also helped increase the interoperability and the reliability of SMPT, leading us to support use cases that we did not originally envision. For instance supporting models with markings that cannot be represented using 32 bits integers. Finally, it motivated us to expand the perimeter of our tool, and to consider the use of methods outside SMT-based approaches, like random walks, which have been used very effectively by competing tools when checking CEX formulas. By being able to quickly identify classes of queries that are solved more efficiently with other methods, we are able to better focus our efforts on what is, we believe, the strong point of SMPT; the verification of "difficult" invariants (that we could roughly define as INV formulas that are not implied by the state equation).

Availability and Contributors

- **Homepage URL:** https://github.com/nicolasAmat/SMPT
- **License Type:** GNU GPL v3.0
- **Affiliation:** LAAS-CNRS

– **Tool Authors:** Nicolas Amat
– **Relevant Publications and Contributions:** [2,4]

5.6 TAPAAL

Overview and Evolution The model checker TAPAAL verifies **T**imed-**A**rc **P**etri nets and is developed at **AAL**borg University in Denmark. The tool was first released in 2008 (see [34,41]), where it provided a graphical interface for modelling and simulating of timed-arc Petri nets as well as a translation to the UPPAAL-style timed automata while using UPPAAL as the backend engine [33,52]. In 2011, TAPAAL released its standalone continuous-time engine [42] for reachability and liveness properties. A discrete-time version of this engine [12,56] was released in 2012. Finally, in 2014 TAPAAL released its own dedicated untimed engine [53] with which it started to participate in the Model Checking Contest.

Initially, TAPAAL participated in the reachability/deadlock category and in 2016 it entered the CTL category with its own on-the-fly CTL model checking algorithm based on dependency graphs [40]. A year later, TAPAAL participated also in the upper-bounds category and finally in 2021 it extended its model checking capabilities to LTL [58]. Nowadays, TAPAAL competes in all categories except state-space size analysis. TAPAAL also supports Petri games [54,26,24] and a new release (in preparation) will allow the user to model and verify timed-arc colored Petri games. All verification engines are supported by a GUI that enables us to model, simulate and verify the different extensions of Petri nets, import and export nets in PNML format, create automatic graphical layout, analyze timed workflow nets [67] and many other features.

MCC Impact TAPAAL's participation in the MCC initiated a significant programming and theoretical research. We have developed a specialised unfolding frontend for our engine in order to deal with colored nets and invented a number of optimization techniques to speedup the unfolding process [19,20]. In order to store the large state space of reachable markings, we designed a novel data structure PTrie [55] that provides a good compromize between the space efficiency while allowing for fast access times. To efficiently deal with large, randomly generated queries, we carry on an extensive query simplification algorithm [23] as a preprocessing step. This step allows us to simplify, using linear programming, verification queries by identifying subformulae that can never/always be satisfied. Perhaps the most beneficial technique is based on structural reductions [25] that are being continuously improved throughout TAPAAL development. Similarly, new partial order reduction techniques, both for the reachability queries as well as LTL [25,58], showed a significant improvement. Finally, the MCC benchmark helped us to develop competitive heuristic search strategies [53,44].

Availability and Contributors

– **Homepage URL:** http://www.tapaal.net
– **License Type:** The GUI is licensed under Open Source Licence 3.0, reduction to timed automata and discrete verification engine is licenced under BSD and

continuous-time and untimed engines are licenced under GPL version 2 and 3, respectively.
- **Affiliation:** Department of Computer Science, Aalborg University, Denmark.
- **Tool Authors:** The development is supervised by Peter G. Jensen, Kenneth Y. Jørgensen and Jiří Srba. A complete list of contributors can be obtained at: https://www.tapaal.net/about/.
- **Relevant Publications and Contributions:** [44,57,20,58,19,25,38,23,55,39]

5.7 TINA.tedd

Overview and Evolution Tedd is a symbolic model-checker part of the TINA toolbox (TIme Petri Net Analyzer) [17], a set of analysis tools supporting various extensions of Petri nets and Time Petri nets developed at LAAS-CNRS since 1981. TINA provides a wide range of tools for state space generation, structural analysis, model checking, or simulation. It only competes in the StateSpace competition at the MCC.

Tedd plays a role similar to other state-exploration tools provided in TINA, called sift and tina, but based on the use of decision diagrams instead of explicit methods. It is officially part of the TINA release since version 3.7.0 and relies on a dedicated implementation of hierarchical Set Decision Diagrams [81] developed together with Alexandre Hamez. While it only provides support for a limited class of reachability properties, such as finding dead states and transitions, it implements new methods based on the combined use of integer linear systems and structural reductions [15,16] for efficiently counting the number of reachable states and the maximal number of tokens in the marking of places. We credit our uninterrupted first place in the StateSpace examination since 2019 to this new equational technique.

The most important change in the last few years is the move from a sequential portfolio (until 2020) to a parallel portfolio, where we combine tedd with the sift and tina tools at the beginning of each run. Explicit methods can sometimes be faster when there are a few markings, and we are not able to find a good variable order with our symbolic approach. They are also useful for detecting unbounded models. In particular, tina is able to identify all the unbounded model instances found in the current benchmark.

MCC Impact Beyond enhancing the interoperability and the reliability of our tools, the MCC had a crucial influence on the design and the enhancement of our state space generation technique based on structural reductions, that was first experimented in the 2018 edition of the contest. Since then, we have used new models in this benchmark to experiment with possible reduction rules. More generally, the set of models provided in the MCC have become an invaluable benchmark for testing our tools and comparing new techniques with existing approaches. The MCC benchmark offers many qualities in this respect, because of its impartiality, its representativeness for a large class of use cases, and its scalability (since many models are parameterized). Our participation to the MCC also motivated us to add several tools to our public release, such as an open-source unfolder for colored models [37], and a new dedicated tool, called reduce, that implements the reduction system presented in [16].

Availability and Contributors

- **Homepage URL:** https://projects.laas.fr/tina/
- **License Type:** TINA is freeware; it is closed source, but binary distributions may be freely installed and used.
- **Affiliation:** LAAS-CNRS
- **Tool Authors:** Bernard Berthomieu with other contributors
- **Relevant Publications and Contributions:** [17]

6 Results of the MCC in 2023

6.1 Reference Tools

For the first time during the 2023 edition, the MCC introduced the concept of "Reference Tools". It means tools that are not part of the main competition, and therefore cannot obtain medals, but which are scored and evaluated on the same benchmark. Reference tools may be submitted without the requirement that the submitter be an author of the tool. To test this new opportunity, Y. Thierry-Mieg developed a driver to retro-fit some tools that have competed in previous editions of the MCC. The Reference Tools submitted in 2023 are Marcie, PNMC, LoLA, Smart and LTSMin. All of them have participated before in the MCC but were not submitted as competitors in 2023.

To ensure the fairest possible comparison, the latest stable release of each tool is paired with a driver[21] that provides an unfolding for colored models using ITS-Tools if the back-end tool does not support them natively. The driver also includes a reduction mode, where the model is first processed by ITS-Tools using the strategy described in [78] that uses an SMT solver, some random or directed walks and structural reductions to produce a simpler Petri net and/or formula. The reference tools benefiting from this preprocessing step are indicated with the "+red" suffix.

6.2 Scoring of Tools in 2023

Participating Tools. All together there were fifteen participating tools. Fourteen coming from submissions and one synthetic tool (see below), made from the winners of the previous competition.

- Submitted tools: GreatSPN, ITS-Tools, SMPT, TAPAAL, tedd.
- Reference tools: Marcie, PNMC, LoLA, Smart, LTSMin.
- Enriched reference tools (including the preprocessing step of [78]): LoLa+red, LTSMin+red, Marcie+red, Smart+red. These tools, since they are enriched are considered to be participants but, since the preprocessing technology they embed is issued from ITS-Tools, they are considered as a variant (i.e. only the best variant can appear in the podium of each examination). In 2020, a previous experience (called ITS-LoLA but reported as being LoLA+red in figure 5) already associated the 2020 version of the preprocessing phase with the 2020 version of LoLA.

[21] Source files are available at: https://github.com/yanntm/MCC-Drivers

The fifteenth tool is the winner of the previous year; it is not a participating tool but a way to see how participants have evolved since the previous edition. This "virtual tool" called 2022-gold changes from examination to examination. It is tedd for StateSpace, ITS-Tools for GlobalProperties and UpperBounds, and TAPAAL for Reachability, CTL and LTL formulas. Of course, these tools are in their setting of the previous edition so that we can measure the progression compared to the winner of the previous edition, as it deals with the new benchmark.

Finally, we introduce BVT (Best Virtual Tool): the results for this tool are computed as the union of the results from all other participants. So if at least one tool answered a given query, so did the BVT.

Basics of score computation. The scoring follows some basic rules:

- Each examination has its own and separated scoring,
- The maximum score of the model having the largest number of instances (41) is approximatively the double of those with only one instance to limit the bias between models in the scoring,
- Since "surprise" models have never been encountered by tools, their scoring is applied a multiplier to give them a bit more importance than "known" models (even if for those, formulas are recomputed every year),
- For each miscalculated result, a penalty of twice the expected score is applied.

The scoring multiplier for the "surprise" models change regularly ; they are detailed in a rule document provided at an early stage of the MCC. The rules for 2023 are presented here : https://mcc.lip6.fr/2023/pdf/rules.pdf.

The scoring of tools. Table 1 summarizes the results for the MCC'2023. It is divided into three parts. The first one depicts scores of competing tools. Then, we report scores from the reference tools. Finally, we show the results of 2022-gold and BVT.

Below each score line, we show the score as a percentage of the one of BVT. This is a way to outline how tools are positioned compared to BVT. We can also see the evolution compared to the winner of 2022 in each examination.

We note from Table 1 that:

- In several cases, enriched reference tools are positioned on the podium: 1^{st} for Reachability formulas and 2^{nd} for CTL formulas.
- The preprocessing step is quite successful by increasing the original score of reference tools. The only decrease observed is for LTSMin (-3%) for StateSpace, which is not an examination for which much improvements are expected for these techniques.
- Winners always increase their score compared to the 2022 ones. Nevertheless, 2022-gold always remain second or third place in the scoring.

Fully detailed results are available on the official web site of the MCC [60]. It details scores but also provide full results tables, as well as execution traces (when relevant), cactus plots to summarize tools' behavior, scatter plots to compare memory and time consumption, etc.

	GreatSPN	smpt	Tapaal	tedd-c	ITS-Tools	LoLa+red	LTSMin+red	Marcie+red	Smart+red	LoLa	LTSMin	Marcie	pnmc	Smart	2022-gold	BVT-2023
StateSPace	14 479	DNC	DNC	16 699	12 794	DNC	8 806	11 825	10 853	DNC	9 073	11 468	10 209	9 535	16 632	17 734
	81,64%	—	—	94,16%	72,14%	—	49,66%	66,68%	61,20%	—	51,16%	64,67%	57,57%	53,77%	93,78%	100%
Global Properties	62 230	DNC	88 642	DNC	108 501	107 384	104 280	105 084	102 776	93 651	9 767	11 018	DNC	22 481	93 632	111 287
	55,92%	—	79,65%	—	97,50%	96,49%	93,70%	94,43%	92,35%	84,15%	8,78%	9,90%	—	20,20%	84,14%	100%
UpperBound	13 390	DNC	19 851	DNC	21 844	20 746	20 974	21 782	21 270	19 407	10 492	11 541	DNC	9 174	19 535	22 263
	60,15%	—	89,17%	—	98,12%	93,19%	94,21%	97,84%	95,54%	87,17%	47,13%	51,84%	—	41,21%	87,75%	100%
Reachability Formulas	21 166	42 813	44 396	DNC	44 340	44 612	43 753	44 115	43 134	39 630	22 168	19 414	DNC	16 620	44 009	45 965
	46,05%	93,14%	96,59%	—	96,46%	97,06%	95,19%	95,97%	93,84%	86,22%	48,23%	42,24%	—	36,16%	95,74%	100%
CTL Formulas	20 228	DNC	34 931	DNC	26 417	32 163	20 925	25 647	DNC	28 021	12 752	17 631	DNC	DNC	34 428	40 899
	49,46%	—	85,41%	—	64,59%	78,64%	51,16%	62,71%	—	68,51%	31,18%	43,11%	—	—	84,18%	100%
LTL Formulas	20 023	DNC	44 019	DNC	44 539	44 139	42 131	DNC	DNC	37 697	CC	DNC	DNC	DNC	44 227	45 663
	43,85%	—	96,40%	—	97,54%	96,66%	92,26%	—	—	82,55%	—	—	—	—	96,86%	100%

legend 1st 2nd 3rd

Table 1: Table depicting the score for the MCC'2023. DNC means "does not compete" (the tool does not participate in the examination) and CC means "cannot compute" (there was a bug in the driver for LTSMin which was discovered too late and no run was able to produce correct results). First ranked tool for an examination is outlined in **bold-red**, second in **bold-green** and third in **bold-blue**.
Please note that ITS-Tools and <tool>+red belong to the same family, thus, only the best one among them is awarded.

6.3 Values Computed by Tools In 2023 and their Evolution

Evaluation of miscalculated values. in 2015, the MCC has introduced the notion of "Tools Confidence". It is a ratio $\frac{Correct}{Computed}$ where $Correct$ is the number of corrected values computed by the tool for the whole examination and $Computed$ the total number of values it has computed.

Since we cannot predict the results of each generated formula, we define a correct values as the one computed by a majority of at least three tools (when several variants of a given tool are submitted, they all count as one complete tool). When only one tool computes a value, it is considered as being true if its confidence is over a threshold (99,3% in 2023). Otherwise, no score is granted and the value is considered as being unknown (it is not considered as a computed one for this tool).

This notion and the associated algorithm, is considered as being quite safe. Even if, in some very rare cases, a false error is detected ; it could not lead to any alteration of the scoring and ranking.

The MCC displays the confidence rate for each examination and also the global one. The "lowest" global confidence for a tool in 2023 is 99,526%[22] ; it means that 574 values were miscalculated out of almost 89000 in all examinations (this tool reaches 100% in several examinations this year).

Note that confidence of participating tools has dramatically been improved since 2015.

[22] See https://mcc.lip6.fr/2023/results.php

Computed Values in 2023. Table 2 shows the number of computed values by tools in 2023. The first line in each category provides absolute numbers while the second line normalizes these values compared to the "Ideal tool" which correctly computes all values for all examinations (its ratio is 100%). As in the previous table, it is separated in three parts. The first one depicts scores of competing tools. Then, we report scores from the reference tools. Finally, we show the results of 2022-gold, BVT and IdealTool.

	GreatSPN	smpt	Tapaal	tedd-c	ITS-Tools	LoLa+red	LTSMin+red	Marcie+red	Smart+red	LoLa	LTSMin	Marcie	pnmc	Smart	2022-gold	BVT-2023	Ideal tool
StateSpace	4 003	DNC	DNC	4 649	2 950	DNC	2 098	2 866	2 586	DNC	2 239	2 714	1 925	2 498	4 619	4 911	**6 712**
	59,64%	—	—	69,26%	43,95%	—	31,26%	42,70%	38,53%	—	33,56%	40,44%	28,68%	37,22%	66,82%	73,17%	**100%**
Global Properties	4 351	DNC	6 985	DNC	7 939	7 946	7 763	7 767	7 787	7 236	728	624	DNC	2 079	7 850	8 110	**8 309**
	51,86%	—	83,25%	—	94,62%	94,71%	92,53%	92,57%	92,81%	86,25%	8,68%	7,44%	—	24,78%	93,56%	96,66%	**100%**
UpperBound	14 976	DNC	23 357	DNC	25 128	24 920	24 893	25 060	24 980	22 644	11 888	10 922	DNC	9 830	24 880	25 483	**26 848**
	55,78%	—	87,00%	—	93,59%	92,82%	92,72%	93,34%	93,04%	84,34%	44,28%	40,68%	—	33,62%	92,67%	94,92%	**100%**
Reachability Formulas	22 951	50 240	50 628	DNC	50 781	52 432	50 252	50 353	49 988	44 041	24 872	18 507	DNC	17 973	50 228	52 679	**53 696**
	42,74%	93,56%	94,29%	—	94,57%	95,78%	93,59%	93,77%	93,09%	82,02%	46,30%	34,47%	—	33,47%	93,54%	98,11%	**100%**
CTL Formulas	21 759	DNC	40 462	DNC	30 509	36 414	24 000	27 184	DNC	30 422	13 587	16 337	DNC	DNC	40 043	46 421	**53 696**
	40,52%	—	75,35%	—	56,82%	67,82%	44,70%	50,63%	—	56,66%	25,30%	30,42%	—	—	74,57%	86,45%	**100%**
LTL Formulas	22 254	DNC	50 021	DNC	50 523	50 086	47 690	DNC	DNC	41 429	CC	DNC	DNC	DNC	50 303	52 012	**53 696**
	41,14%	—	93,16%	—	94,01%	93,28%	88,81%	—	—	77,15%	—	—	—	—	93,68%	96,83%	**100%**

legend 1st 2nd 3rd

Table 2: Table depicting the number of values computed during the MCC'2023. DNC means "does not compete" (the tool does not participate in the examination) and CC means "cannot compute" (there was a bug in the driver for LTSMin which was discovered too late and no run was able to produce correct results). Best tools (considering the number of computed values) are outlined in **bold-red**, second best tools in **bold-green** and third best tools in **bold-blue**.
Please note that ITS-Tools and <tool>+red belong to the same family, thus, only the best one among them is considered.

We can observe that, for GlobalProperties, the tool computing the largest number of values is not the winner outlined in Table 1. There are two reasons: *(i)* the scoring puts some multipliers for "surprise models" which were never previously confronted to tools, and *(ii)*, when a tool miscalculates a value, it has a penalty of twice the expected score for this value. This may change the score when the number of computed values is very close: for GlobalProperties, ITS-Tools (winner) only computes 7 values less than Lola+red, but probably more in surprise models. However, the order of tools in the podium is correlated with the number of values computed; permutations only occurs when tools are very close in terms of performances.

We note from Table 2 that:

- As it could be noted in Table 1, the preprocessing step strongly increases the number of values computed by reference tools (up to 1 245% in the case of GlobalProperties); However, StateSpace marginally benefits from this step and LoLA, which embeds its own strategies is also less concerned.
- Winners always increase their computation capabilities compared to the 2022 ones. Nevertheless, 2022-gold always remains second or third.

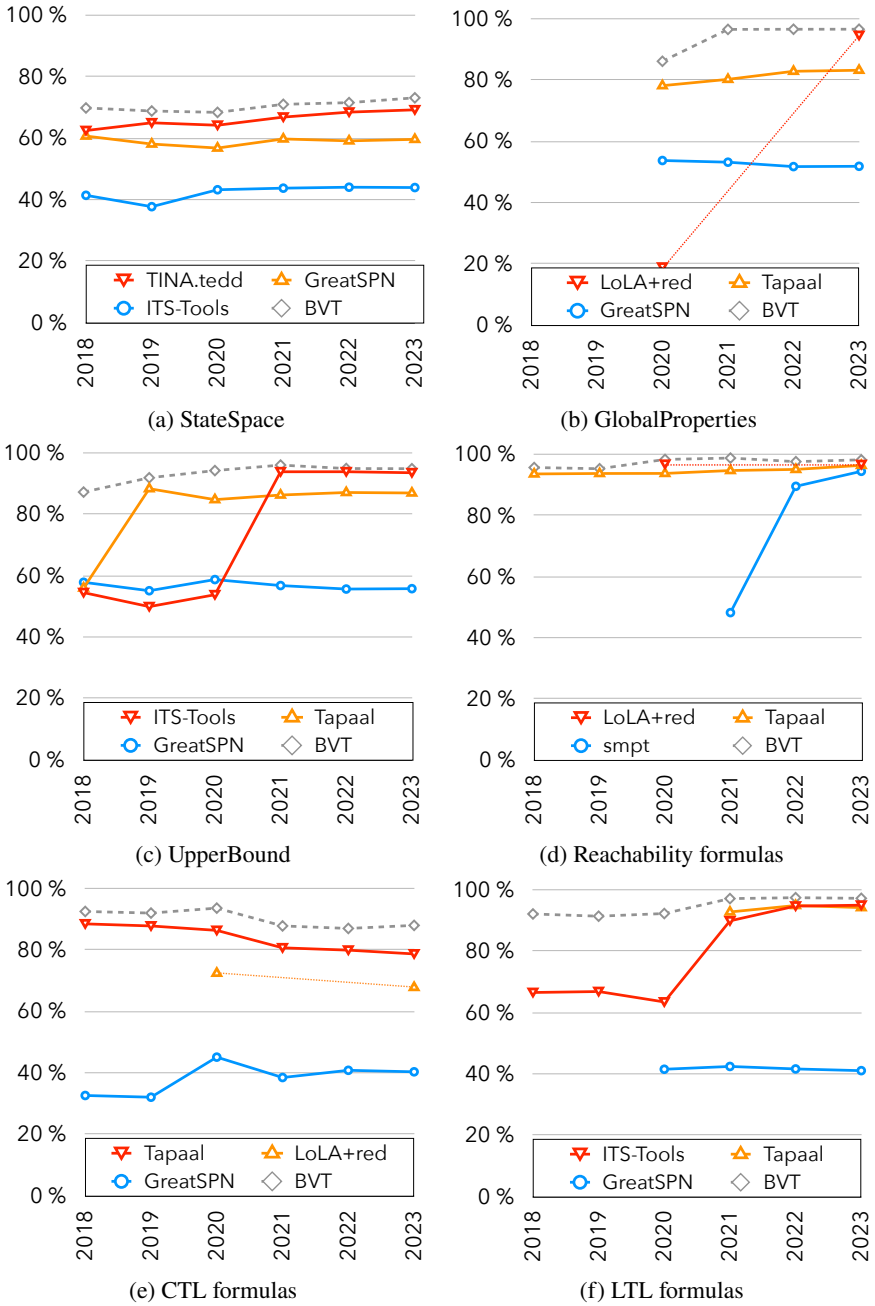
- For some examinations, almost all possible values are computed (*e.g.* 95.78% for reachability formulas, and even over 96% for BVT); if this can be seen as a good improvement of tools, it also outlines that formulas complexity might be improved in the future.
- StateSpace appears to be the most difficult examination since the best tool only computed 69.26% of the values.
- BVT computes between 1.24% (UpperBound) and 14.73% more values than the best tool, showing that, if complementarity between tools exists, it remains quite low at this stage.

6.4 Evolution of the Best Tools of 2023 since 2018 (from the perspective of computed values).

Figure 5, shows, for each examination, how the three tools which compute the highest number of values in 2023 have evolved since 2018. It is a way to check how they evolved and to compare this evolution with BVT. All data is normalized to the "ideal tool" which computed all the values (representing 100% in the charts).

Let us have observations for each examination first:

- **StateSpace** (Fig. 5a): it clearly shows that over the years, the tool computing the highest number of values is very close to BVT but yet far from the "ideal tool". Since the introduction of a dedicated technique for counting states [15], TINA.tedd dominates this examination.
- **GlobalProperties** (Fig. 5b): because the properties used in this examination were changed after 2019, there is no meaningful data before 2020. In 2020, the best tool in 2023 was only capturing deadlocks but an efficient technique elaborated for deadlocks (which led to the preprocessing step associated to reference tools) was extended to other questions in 2021 and pushed forward the tool at the first place, very close to BVT. The winner follows the curve of ITS-Tools; first in the podium since 2021 (see Fig. 7) but computing 7 less values than LoLA+red.
- **UpperBound** (Fig. 5c): in 2021 the preprocessing step was also applied to this examination, thus pushing the third tool at the first place and very close to BVT. In 2018 there were 4 more participating tools and variants; this explains the large difference between the best tool and BVT.
- **Reachability formulas** (Fig. 5d): the first tool (a reference tool with the preprocessing step) holds the first position for its first participation in such a setting (as for a similar setting in 2020); LoLA, the original tool was in the podium in 2018, 2019 and 2020. The third tool's first participation was in 2021 and could capture a large number of values. It then entered in the podium at the third position in 2022 and gets very close to the other podium tools in 2023.
- **CTL formulas** (Fig. 5e): the best tool in this examination, TAPAAL, has consistently dominated the competition since 2018. It is also one of the examination where the difference between the first two tools is the largest, which may be explained by the use of specific reduction techniques. In 2018 and 2019, LoLA was very close to TAPAAL (see Fig. 7) before TAPAAL took a clear advantage.



(a) StateSpace (b) GlobalProperties

(c) UpperBound (d) Reachability formulas

(e) CTL formulas (f) LTL formulas

Fig. 5: Evolution of the performances of the best 2023 tools (considering the number of values computed) between 2018 and 2023. The Ideal Tool (not shown here) has always 100% of the computed values. This figure only features the three tools in the podiums for 2023 and BVT. Tools in the podium are sorted according to their rank in the MCC'2023. The line in red corresponds to the best one according to Table 2, the line in orange to the second best tool and the line in blue to the third one.

– **LTL formulas** (Fig. 5f): TAPAAL, ranked second in 2023, broke records in its first appearance in this examination, in 2021. ITS-Tools also dramatically increased its performances with the preprocessing step; it is even a bit better than TAPAAL since 2022 (a few more values computed). In 2018, for its last participation, LTS-Min was ranked second and LoLA was first in 2019. We should also underline the very good performances of EnPAC between 2020 and 2022, which was not participating in 2023 (see Fig. 7).

A recurring pattern in all the examinations is that at least one tool stays in the podium for the whole 2018–2023 period. And for an examination like StateSpace, the three podium tools have not changed. This seems to indicate that there is a bonus for the most experienced tools, which is understandable. On the other hand, we often observe that new tools rapidly reach a high position; often entering the podium. This may be explained by the fact that new tools often embed some "breakthrough technique". This occurred several times in the 2018–2023 period. For instance with the preprocessing step for GlobalProperties and UpperBound that we just described, or with a new SMT encoding for reachability properties. This also occurred twice for LTL (once with a tool that did not participate in 2023, but twice reached the third place).

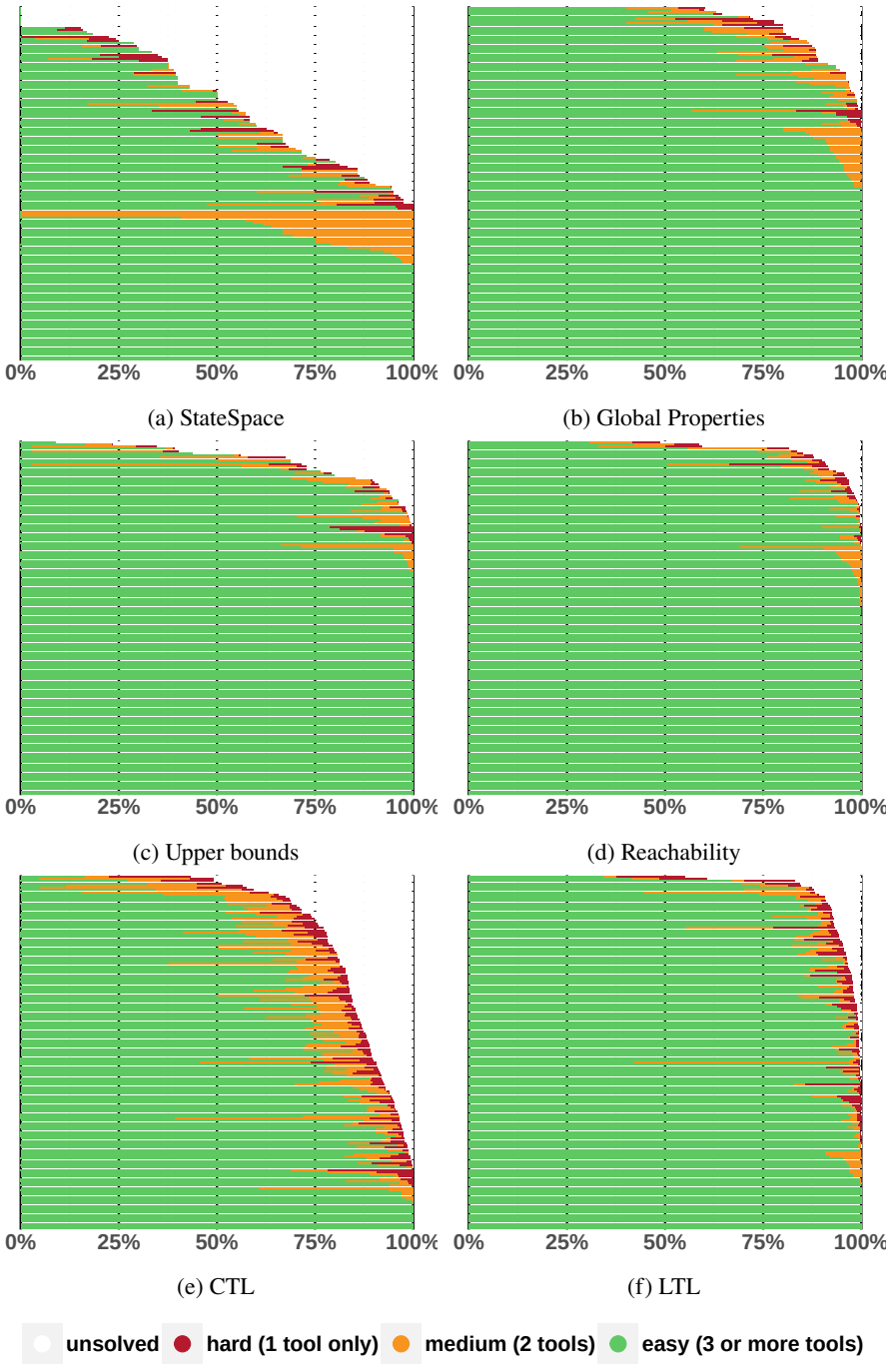
We can observe a general progression of BVT in all cases except for CTL formulas. This must be tempered by the fact that there was an increase of 77.2% of the values to be computed for each examination between 2018 and 2023 (due to the introduction of new models). The decrease of BVT for CTL formulas can also be explained by the evolution in the way formulas have been generated, making them a bit more difficult. This is also true for the increase of LTL formulas where a new generation technique was introduced; it produces fairer formulas (spanning all the classes defined by Pnueli and Manna) but still needs some improvement in the way atomic propositions are selected.

We also observe that the best tool is getting closer to BVT; it means that the best tools are able to compute almost all the values computed by all the tools together, making them the really best solution.

6.5 Observations on Hardness of Examinations

With so many instances and queries, and in many cases some very high success rates, it is hard to see the specificities of each examination. That is why we proposed an innovative visualization that makes it easier to compare the complexity of each examination separately.

The plots in Fig. 6 represent the results of the examination in 2023. Each horizontal line represents a different model (with separate lines for a COL model and its unfolded PT version), so there are 158 lines in each plot. At best a line can reach the right border representing that 100% of queries were solved by the BVT in that examination for that model (all instances of it). Such lines are sorted to be at the bottom of the diagram, they are the "easiest" models. At worst a line can be empty indicating no query could be answered by any tool on any instance of that model (this is the case at the top of the state space plot for instance). Visually, if the box is full, the BVT answered all queries; the surface in white represents unsolved ones.



(a) StateSpace (b) Global Properties

(c) Upper bounds (d) Reachability

(e) CTL (f) LTL

Fig. 6: Model difficulty in 2023 across different examinations.

To refine this view, the colors give an indication of tool complementarity; we count for each query that was answered how many *different* tools were able to provide that answer. Variants of a tool do not count as "different", for instance all the "+red" variants and ITS-Tools count for a single tool family. A query at least three tools answer is considered "easy", if two tools from different origins could compute the answer it is "medium", and when only one tool could compute the result it is marked as "hard". So the darker medium and hard colors correspond to existence of tool complementarity, queries that *some* tools (but not all) can solve. We designate a model as "fully solved" if the BVT could answer all queries on all model instances, and as "easy" if at least three tools could compute all queries.

- **State Space.** This category is the hardest of the MCC, with the BVT (Best Virtual Tool) only reaching 73.16% of the ideal tool score in 2023. It is also the only examination in which for some models no query could be answered. In total 70 models are fully solved (of which 43 are considered "easy"). The hardest models are identified as Echo, Election2020, FamilyReunion, HypercubeGrid, PolyORBNT, RERS. None of the queries could be answered for these models.
- **Global Properties.** This examination is currently one of the "easiest" in the MCC with the BVT reaching in percentage of "Ideal Tool": 97.55% for Reachability Deadlocks, 96.66% of Stable Marking, 100% of One Safe (NB: the only query in the contest where BVT answers for all model instances), 93.62% of Quasi Liveness and 95.47% of Liveness. Quasi-Liveness and Liveness are thus the hardest queries in this examination. In total 112 models are fully solved (of which 77 are considered "easy"). The hardest models are identified as RERS, SharedMemory (COL), DrinkVendingMachine (COL), SafeBus and EisenbergMcGuire. These models are all structurally very large.
- **Upper Bounds.** This examination is relatively easy; in 2023 the BVT reached 94.91% of the ideal score, only slightly lower than the global properties examination. In total 120 models are fully solved (of which 100 are considered "easy"). The hardest models are identified as SemanticWebServices, CANInsertWithFailure, RERS, FamilyReunion, PhilosophersDyn, FunctionPointer, Planning, VehicularWifi. Most of these models are in fact unbounded.
- **Reachability.** This examination is mostly solved by the BVT, which scores up to 98.1% of the ideal score in 2023, the highest score over all examinations. This result is quite impressive given the size of the input models (see Sect. 3) and a fortiori their state space. In total 116 models are fully solved (of which 80 are considered "easy"). The hardest models are identified as RERS, SharedMemory (COL), DoubleExponent, CANInsertWithFailure, FamilyReunion. Only the RERS models are truly hard (60% or less queries treated), 80% of the queries are answered on SharedMemory.
- **CTL.** This is the second hardest examination (after State Space) currently, the BVT scores 86.4% of the ideal score. As the colors on the plot show, there is also a significant amount of tool complementarity, with many queries solved by a single tool. The leader of the examination TAPAAL in 2023 scores 75.3% of the ideal score. In total 27 models are fully solved (of which 12 are considered "easy"). The hardest models are identified as FamilyReunion, RERS, SharedMemory (COL),

Planning, CANInsertWithFailure, Philosophers and HouseConstruction. These are mostly models for which the state space computation was not possible.

– **LTL.** Surprisingly, the LTL examination seems quite easy at the current time, with BVT reaching 96.9% of the ideal score. This is almost the level observed with Reachability. This is in part because, as discussed in Sect. 4.2, the formulas are biased towards counter-examples (70.1% of formulas can be contradicted versus only 26.7% that must be proven). In total 60 models are fully solved (of which 19 are considered "easy"), comparing these values to Reachability we see that *some* of the LTL queries are hard. The hardest models are identified as FamilyReunion, SharedMemory (COL), TokenRing, RERS, Philosophers and DatabaseWithMutex (COL). Apart from RERS, these models are mostly colored models with structurally very large unfoldings.

7 Detailed Results Analysis for 2018-2023

In this section we analyze each category of the MCC to look at the evolution of the results over the period ranging from 2018 to 2023 inclusive.

Figure 7 shows the evolution of tool performance over time as they strive to answer 100% of the queries. In these plots, the "Best Virtual Tool" BVT is computed as the union of all other tool results (hence it is always on top). To have more comparable results from year to year, all the results in this section are normalized to be a percentage of the queries that were answered. Thus 100% in the plots corresponds to answering all queries in the category. This choice deviates from the scoring used in the MCC to decide the podium, where surprise models are worth more points, errors are scored negatively, etc. But it makes the data more easily comparable on a year to year basis and is simpler to interpret.

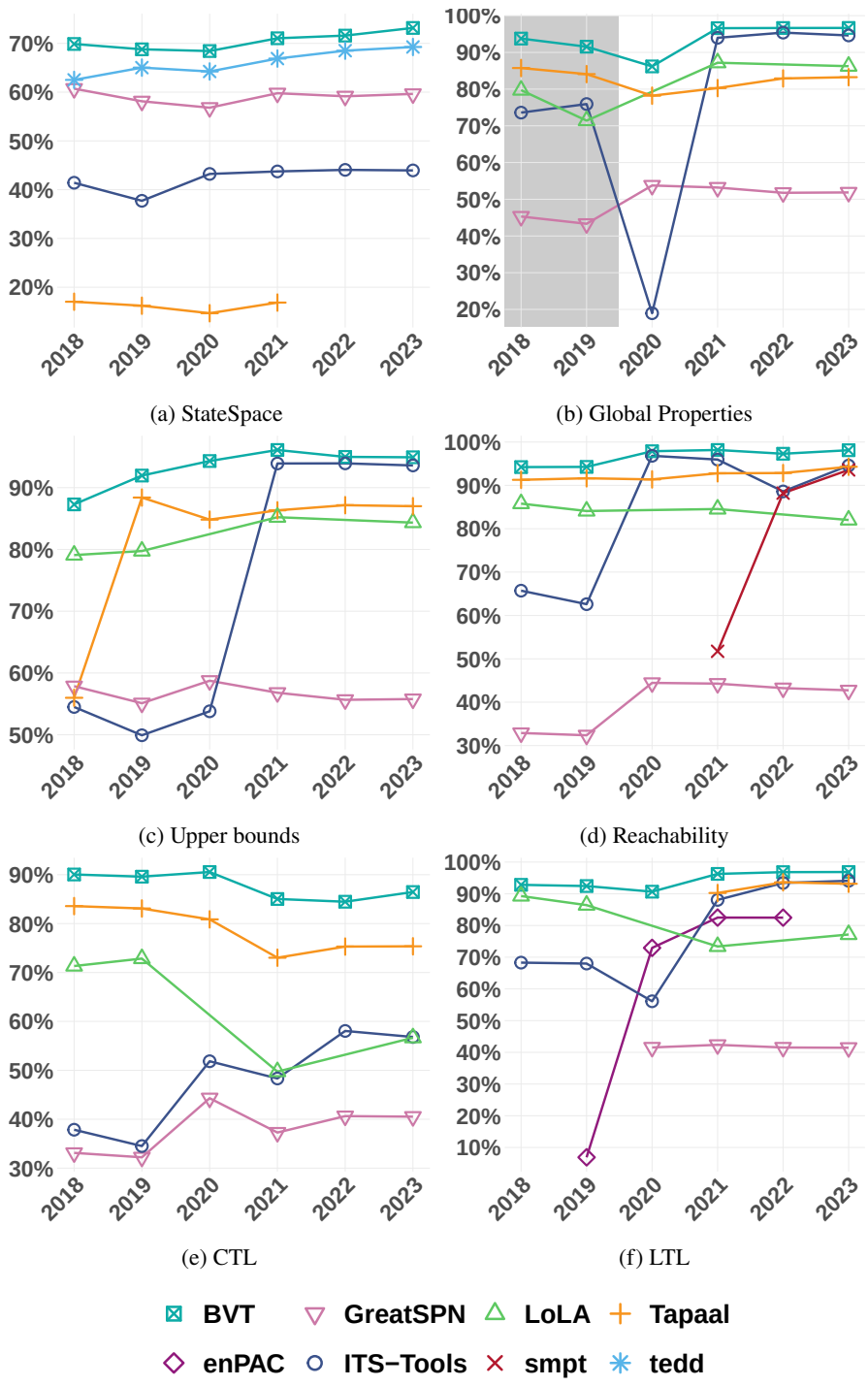
We plot all the tools that have participated as a competitor since 2021 as well as the BVT. The tools are presented in more detail in Sect. 5 by their respective authors. The "+red" combination tools that use ITS-Tools as a preprocessing step are not represented, nor are the reference tools.

7.1 Evolution of State Space Examination

The number of participants in the examination is stable with regular contenders Tedd, GreatSPN, ITS-Tools. All these tools use symbolic decision diagram based strategies to compute the desired metrics; explicit state based methods seem to perform more poorly on this examination, and tools that use these technologies have withdrawn from participating in it over time.

ITS-Tools relies on hierarchical set decision diagrams [81] (SDD) as well as decompositions of the system using NUPN (see Sect. 3.3) and Louvain modularity [21]. GreatSPN uses the Meddly decision diagram library [13] and some advanced heuristics for variable ordering described in [10]. Tedd uses a variant of hierarchical SDD as well as a unique technology to compute these metrics on a structurally reduced net [16].

Tedd is the best tool in the category since 2018 where it overtook GreatSPN, and in 2023 reaches 94.66% of the score of BVT. This indicates the existence of some



(a) StateSpace

(b) Global Properties

(c) Upper bounds

(d) Reachability

(e) CTL

(f) LTL

Fig. 7: Evolution of tool performance across different examinations.

complementarity between the tools, probably related to the fact that static ordering heuristics for the variables in the decision diagram are critical to efficiency.

7.2 Evolution of Global Properties Examination

The global property examination contains five queries with a boolean answer expected: Reachability Deadlock, Stable Marking, One Safe, Quasi Liveness and Liveness. Before 2020, only deadlocks were computed (light gray area). This is visible on the pluri-annual plot (Fig. 7b) where the BVT score dips down in 2020, and ITS-Tools that still only supported Reachability Deadlock that year scoring below 20%.

The main contenders in this examination are ITS-Tools, LoLA, TAPAAL, GreatSPN in that order. The relative rankings of these tools is stable since 2021.

GreatSPN answers by building the state space as a decision diagram then querying it, but ITS-Tools, LoLA and TAPAAL all use strategies dedicated to the problem. LoLA uses specific reductions described in [88], some strategies dedicated to colored nets [83] as well as an advanced portfolio management [89]. TAPAAL uses advanced compression [55], structural reductions [25] and directed search heuristics [44]. ITS-Tools relies on an abstraction refinement based strategy [80] with steps dedicated to the global properties that include some semi-decision procedures, as well as both decision diagram based technology [81] and explicit state model-checking with partial order reductions [63].

Current leader ITS-Tools in the category reaches overall 97.89% of the BVT score, itself very close to the ideal score (above 95% except in Quasi Liveness). However, there is still some complementarity between the participants and room for improvement on the harder queries Quasi Liveness (ITS-Tools is at 96.3% of BVT) and Liveness (ITS-Tools is at 96.75% of BVT).

7.3 Evolution of Upper Bounds Examination

The main contenders in this examination are ITS-Tools, TAPAAL, LoLA, GreatSPN in that order. The relative rankings of these tools is stable since 2021 but the examination was introduced in MCC'2016 and there has been significant improvements in it with the BVT progressing from 87.2% in 2018 to 94.9% of the ideal score in 2023. ITS-Tools currently leads the examination with 98.60% of BVT.

Some notable improvements to tools and algorithms are visible in the plot 7c (left) such as the marked progress of TAPAAL from 2018 to 2019, continuous improvements of Lola from 2018 to 2021, and the break in performance of ITS-Tools between 2020 and 2021 due to introduction of dedicated strategies [80] instead of only relying on decision diagrams.

7.4 Evolution of Reachability Examination

The main contenders in this examination are ITS-Tools, TAPAAL, SMPT, LoLA, GreatSPN in that order. It is notable that in 2023 the top three competing tools (ITS-Tools, TAPAAL and SMPT) are within one percent of each other at roughly 94% of the ideal

score, but the BVT is much higher at 98.1% of ideal indicating that many queries could be solved by only one tool.

There has been significant improvements in this examination since 2018; the BVT score has risen from 94.2% (with leader TAPAAL in 2018 scoring 91.3% of ideal) to 98.1% in 2023 (with the leader a combination of ITS-Tools and LoLA that scored 95.8%).

Most of the current leading tools benefit from a mixture of advanced structural reduction rules to reduce the input nets, some form of linear reasoning on e.g. the state equation and guided or directed search to achieve this performance.

Notable improvements to the performance of tools are also visible on Fig. 7d, for instance the progress of GreatSPN between 2018 and 2019 due to variable ordering heuristics [10] or for ITS-Tools between 2019 and 2020 with a switch from pure decision diagrams to strategies involving SMT and structural reductions [79]. The relatively recent tool SMPT saw a drastic rise from its first submission in 2021 to become a top contender in 2023 thanks to introduction of new strategies [2]. TAPAAL also shows clear improvements from year to year with more queries and models treated, though it started from a high level in 2018 so it is less apparent on the plot. The strategies of ITS-Tools [79] to reduce the reachability problem are very effective in this examination, with all the "+red" combination tools scoring 93% or better even when the naked reference tool scores below 50%.

7.5 Evolution of CTL Examination

The main contenders in this examination are TAPAAL, LoLA, ITS-Tools and GreatSPN in that order. The BVT scores 86.4% of ideal score in 2023, which makes this the second hardest examination after state space. There is also a visible dip in the BVT score since 2021 (down to roughly 85% from roughly 90% up to 2020) that we attribute to the improvements of the formula generator Citili (see Sect. 4.1).

TAPAAL is leading the examination with 75% of queries answered, while LoLA (a reference tool in 2023) and ITS-Tools are very close to each other around 56.7%. The combination tool using ITS-Tools and LoLA performs significantly better than either in isolation at 67.8% of queries treated. GreatSPN solves 40.7% of CTL queries in 2023 but a refined analysis shows 10.6% of these answers are solved only by GreatSPN.

7.6 Evolution of LTL Examination

The main contenders in this examination are ITS-Tools, TAPAAL, EnPAC, LoLA and GreatSPN in that order. The BVT solves 97.3% of all queries a significant progression from 92% in 2018, but this high value could indicate that the formulas are not hard enough.

The leaders ITS-Tools and TAPAAL are within one percent of each other around 95%, a result that is also matched by the combination of ITS-Tools and LoLA. Then EnPAC and LoLA are around 80% of queries solved, though EnPAC did not participate in 2023 and LoLA was only submitted as a reference tool. Using its CTL* verification engine [11], GreatSPN performs similarly to CTL solving 41% of the queries.

The results of competitors in this examination have progressed a lot. TAPAAL only started supporting the LTL examination in 2021, where it immediately took the gold. EnPAC participated from 2019 where it solved 7% of queries to 2022 where it solved 82.6% of queries, an impressive progression. It obtained bronze medals in 2021 and 2022.

8 Conclusion and novelties for the next MCCs

We presented the Model Checking Contest and how it has been operated in its last edition (2023). We have also presented an analysis of the results for 2023, highlighted with some observations deduced from the six editions that occurred between 2018 and 2023. It shows that the MCC is an established regular event which already had a strong impact on our community: prototype tools' efficiency has been improved and a large benchmark covering models, formulas, etc., is now available.

We now sketch some novelties that could be considered to enrich the competition in the upcoming years.

8.1 Standardizing the Formula Format

This is not an issue for the competition itself but an interesting outcome for the whole community. Actually, all the participating tools have implemented libraries to parse the formulas provided in some of the examinations. Since models are provided using an ISO/IEC international standard favoring exchange and interoperability of tools, it could be of interest to provide such a standard for properties.

At this stage the format proposed in the MCC is a sort of "de facto" standard based on an XML representation of formulas. But it is not complete yet. In particular, it is not currently appropriate to be used as an exchange or storage format, but could be extended in that direction (TRUE/FALSE terminal nodes, richer atomic properties, more comparison operators than $\leq$, etc.). Thus, it needs to be refined and discussed to be possibly inserted in a future revision of the Petri net standard (ISO/IEC 15909, parts I to III).

The Model Checking Contest would help to gradually experiment with this format and improve it, so that various libraries are available when the standard is out.

8.2 Execution

At this stage, tools must report, when they provide results, the techniques they used to to compute values. This could lead to an analysis of the evolution of techniques between 2015 and 2019 [62]. However, such analysis is quite complex since the vocabulary designating techniques is not normalized. Thus, manual preprocessing steps must be performed. Moreover, tools do not always differentiate the techniques for a given formula but report those of all the formulas computed in one examination.

So, there is an improvement we could complete in the next years. Then, since we get a large volume of data every year, this could rapidly enable the detection of situations were one technique (or a combination of techniques) is more efficient than other ones.

Such an analysis has been proposed before [32] but the corresponding tool, competing in 2018, was not really successful.

8.3 Models

Model Forms. Each submitted model consists of one or several instances, as well as a model form. These forms, initially filled in by the people submitting the models, contain precious information about the models' origin, the methods and tools used to generate their instances, and their properties. The model board members carefully check the content of these forms, enriching them as much as possible with the results provided by the tools participating in the MCC, as well as other tools (e.g. CÆSAR.BDD[23] and ConcNUPN [29]). This is a difficult and time-consuming task, particularly for models including instances that cannot be handled by state-of-the-art tools, or presenting unexpected specifics (e.g. divergent behaviors between colored and PT instances). It happens quite regularly that models that are several years old need to be enhanced.

Currently, model forms (in tex format) are processed to produce XML files[24] giving, for each model, a total of 24 global properties (each of which can be evaluated as true, false or unknown). Currently, when there are divergences among instances of a model (for example, where some instances are safe, while others are not), the property has to be marked as "unknown" and we have to write manually a statement that is only readable by humans. We would like to improve this format, to be able to describe these properties for each instance, possibly with new useful properties for the community, and possibly beyond ternary evaluations (for example, it is helpful to know the names of (all) dead transitions, or at least their number, rather than merely knowing their existence).

Correction of Existing Models. As far as possible, we avoid altering instances from previous editions of the MCC, the only exception being when they present serious, unnoticed problems. For example, instances containing places or transitions with label names different from their XML id, a nice feature of the PNML format, but leading to incorrect formula evaluation by certain tools not taking this feature into account. Or when some instances are found to be isomorphic (there is a permutation of their places, their transitions and, in the case of NUPNs, their units, such that these two instances coincide): in this case, we eliminate the duplicate.

Model Enhancements. In order to follow the noticeable progression of model-checkers through successive editions and to keep existing models "challenging" for these tools, we can add extra instances to existing models.

Feeding some Examinations with Structural Information. One idea should be to feed the reachability, CTL and LTL examinations with structural information (extracted when computed from GlobalProperties questions). This information can help tools to improve their algorithms.

[23] https://cadp.inria.fr/man/caesar.bdd.html.
[24] https://mcc.lip6.fr/verdict-properties.php.

New Petri Net Formalisms (e.g. Time). So far, we support Petri nets and colored (*i.e.* Symmetric) Petri nets; both are described in the ISO/IEC standard. One suggestion is to extend the MCC to time(d) nets. This is a difficult task since there are only a few tools dealing with time in Petri nets. Moreover, these tools support different timing schemes. The main ones are: Time Petri Nets [68] that associate a firing interval with each transition ; Timed Petri Nets [72] which feature a global clock and tokens carry an availability time ; and Timed-Arc Petri Nets [47] where tokens carry an age and arcs between places and transitions are labelled with time intervals restricting the age of tokens available for transition firing.

The idea is to define the core semantics supported by existing tools and to encode it using the new extensions mechanisms proposed in the Part III of the ISO/IEC standard. Integrating such Petri Nets extensions must involve both the tool developers and the MCC organizers.

References

1. Amat, N.: Octant: The Reachability Formula Projector. A tool to project Petri net reachability properties on reduced nets using polyhedral reduction. (2023), https://github.com/nicolasAmat/Octant
2. Amat, N., Berthomieu, B., Dal Zilio, S.: A Polyhedral Abstraction for Petri Nets and its Application to SMT-Based Model Checking. Fundamenta Informaticae **187**(2-4) (2022). https://doi.org/10.3233/FI-222134, publisher: IOS Press
3. Amat, N., Chauvet, L.: Kong: a tool to squash concurrent places. In: International Conference on Applications and Theory of Petri Nets and Concurrency. pp. 115–126. Springer (2022). https://doi.org/10.1007/978-3-031-06653-5_6
4. Amat, N., Dal Zilio, S.: SMPT: A testbed for reachability methods in generalized Petri nets. In: Chechik, M., Katoen, J.P., Leucker, M. (eds.) Formal Methods: 25th International Symposium, FM 2023. pp. 445–453. Springer (2023). https://doi.org/10.1007/978-3-031-27481-7_25
5. Amat, N., Dal Zilio, S., Hujsa, T.: Property Directed Reachability for Generalized Petri Nets. In: Tools and Algorithms for the Construction and Analysis of Systems (TACAS). LNCS, vol. 13243. Springer (2022). https://doi.org/10.1007/978-3-030-99524-9_28
6. Amat, N., Dal Zilio, S., Le Botlan, D.: Leveraging polyhedral reductions for solving Petri net reachability problems. International Journal on Software Tools for Technology Transfer (Dec 2022). https://doi.org/10.1007/s10009-022-00694-8
7. Amat, N., Dal Zilio, S., Le Botlan, D.: Automated polyhedral abstraction proving. In: International Conference on Applications and Theory of Petri Nets and Concurrency. pp. 324–345. Springer (2023). https://doi.org/10.1007/978-3-031-33620-1_18
8. Amparore, E.G.: Stochastic modelling and evaluation using greatspn. ACM SIGMETRICS Performance Evaluation Review **49**(4), 87–91 (2022)
9. Amparore, E.G., Ciardo, G., Miner, A.S.: The footprint form of a matrix: Definition, properties, and an application. Linear Algebra and its Applications **651**, 209–229 (2022)
10. Amparore, E.G., Donatelli, S., Ciardo, G.: Variable order metrics for decision diagrams in system verification. Int. J. Softw. Tools Technol. Transf. **22**(5), 541–562 (2020)
11. Amparore, E.G., Donatelli, S., Gallà, F.: starMC: an automata based CTL* model checker. PeerJ Computer Science **8**, e823 (2022)
12. Andersen, M., Larsen, H., Srba, J., Sørensen, M., Taankvist, J.: Verification of liveness properties on closed timed-arc Petri nets. In: Proceedings of the 8th Annual Doctoral Workshop

on Mathematical and Engineering Methods in Computer Science (MEMICS'12). LNCS, vol. 7721, pp. 69–81. Springer-Verlag (2013)
13. Babar, J., Miner, A.S.: Meddly: Multi-terminal and edge-valued decision diagram library. In: QEST. pp. 195–196. IEEE Computer Society (2010)
14. Bérard, B., Bidoit, M., Finkel, A., Laroussinie, F., Petit, A., Petrucci, L., Schnoebelen, P., McKenzie, P.: Systems and Software Verification, Model-Checking Techniques and Tools. Springer (2001). https://doi.org/10.1007/978-3-662-04558-9, https://doi.org/10.1007/978-3-662-04558-9
15. Berthomieu, B., Le Botlan, D., Dal Zilio, S.: Petri net Reductions for Counting Markings. In: Model Checking Software (SPIN). LNCS, vol. 10869. Springer (2018). https://doi.org/10.1007/978-3-319-94111-0_4
16. Berthomieu, B., Le Botlan, D., Dal Zilio, S.: Counting Petri net markings from reduction equations. International Journal on Software Tools for Technology Transfer **22** (2019). https://doi.org/10.1007/s10009-019-00519-1, publisher: Springer
17. Berthomieu, B., Ribet, P.O., Vernadat, F.: The tool TINA–construction of abstract state spaces for Petri nets and time Petri nets. International journal of production research **42**(14), 2741–2756 (2004). https://doi.org/10.1080/00207540412331312688
18. Berthomieu, B., Vernadat, F.: Time petri nets analysis with TINA. In: QEST. pp. 123–124. IEEE Computer Society (2006)
19. Bilgram, A., Jensen, P., Pedersen, T., Srba, J., Taankvist, P.: Improvements in unfolding of colored petri nets. In: Proceedings of the 15th International Conference on Reachability Problems (RP'21). LNCS, vol. 13035, pp. 69–84. Springer-Verlag (2021)
20. Bilgram, A., Jensen, P., Pedersen, T., Srba, J., Taankvist, P.: Methods for efficient unfolding of colored petri nets. Fundamenta Informaticae pp. 1–24 (2023), to appear.
21. Blondel, V.D., Guillaume, J., Lambiotte, R., Lefebvre, E.: Fast unfolding of community hierarchies in large networks. CoRR **abs/0803.0476** (2008)
22. Bobot, F., Bromberger, M., Hoenicke, J.: The International SMT Competition Web Page. https://smt-comp.github.io/ (July 2023)
23. Boenneland, F., Dyhr, J., Jensen, P., Johannsen, M., Srba, J.: Simplification of ctl formulae for efficient model checking of petri nets. In: Proceedings of the 39th International Conference on Application and Theory of Petri Nets and Concurrency (Petri Nets'18). LNCS, vol. 10877, pp. 143–163. Springer-Verlag (2018)
24. Boenneland, F., Jensen, P., Larsen, K., Muniz, M., Srba, J.: Stubborn set reduction for timed reachability and safety games. In: Proceedings of the 19th International Conference on Formal Modelling and Analysis of Timed Systems (FORMATS'21). LNCS, vol. 12860, pp. 32–49. Springer-Verlag (2021)
25. Bønneland, F., Dyhr, J., Jensen, P., Johannsen, M., Srba, J.: Stubborn versus structural reductions for petri nets. Journal of Logical and Algebraic Methods in Programming **102**(1), 46–63 (2019)
26. Bønneland, F., Jensen, P., Larsen, K., Muniz, M., Srba, J.: Partial order reduction for reachability games. In: Proceedings of the 30th International Conference on Concurrency Theory (CONCUR'19). LIPICS, vol. 140, pp. 23:1–23:15. Dagstuhl Publishing (2019)
27. Bouvier, P.: The VLSAT-3 Benchmark Suite. Technical Report RT-0516, INRIA, Grenoble, France (Dec 2021), available from https://hal.inria.fr/hal-3468625 and https://arxiv.org/abs/2112.03675
28. Bouvier, P., Garavel, H.: The VLSAT-1 Benchmark Suite. Technical Report RT-0510, INRIA, Grenoble, France (Nov 2020), available from https://hal.inria.fr/hal-03007233 and https://arxiv.org/abs/2011.11049
29. Bouvier, P., Garavel, H.: Efficient Algorithms for Three Reachability Problems in Safe Petri Nets. In: Buchs, D., Carmona, J. (eds.) Proceedings of the 42nd International Conference on

Application and Theory of Petri Nets and Concurrency (PETRI NETS'21), Paris, France. Lecture Notes in Computer Science, vol. 12734, pp. 339–359. Springer (Jun 2021)
30. Bouvier, P., Garavel, H.: The VLSAT-2 Benchmark Suite. Technical Report RT-0514, INRIA, Grenoble, France (Sep 2021), available from https://hal.inria.fr/hal-03337115 and https://arxiv.org/abs/2110.06336
31. Bouvier, P., Garavel, H., de León, H.P.: Automatic Decomposition of Petri Nets into Automata Networks – A Synthetic Account. In: Janicki, R., Sidorova, N., Chatain, T. (eds.) Proceedings of the 41st International Conference on Application and Theory of Petri Nets and Concurrency (PETRI NETS'20), Paris, France. Lecture Notes in Computer Science, vol. 12152, pp. 3–23. Springer (Jun 2020)
32. Buchs, D., Klikovits, S., Linard, A., Mencattini, R., Racordon, D.: A model checker collection for the model checking contest using docker and machine learning. In: Khomenko, V., Roux, O.H. (eds.) Application and Theory of Petri Nets and Concurrency - 39th International Conference, PETRI NETS 2018, Bratislava, Slovakia, June 24-29, 2018, Proceedings. Lecture Notes in Computer Science, vol. 10877, pp. 385–395. Springer (2018)
33. Byg, J., Jørgensen, K., Srba, J.: An efficient translation of timed-arc Petri nets to networks of timed automata. In: Proceedings of the 11th International Conference on Formal Engineering Methods (ICFEM'09). LNCS, vol. 5885, pp. 698–716. Springer-Verlag (2009)
34. Byg, J., Jørgensen, K., Srba, J.: TAPAAL: Editor, simulator and verifier of timed-arc Petri nets. In: Proceedings of the 7th International Symposium on Automated Technology for Verification and Analysis (ATVA'09). LNCS, vol. 5799, pp. 84–89. Springer-Verlag (2009)
35. Chiola, G., Dutheillet, C., Franceschinis, G., Haddad, S.: A symbolic reachability graph for coloured Petri nets. Theoretical Computer Science **176**(1–2), 39–65 (1997)
36. Christensen, S., Kristensen, L.M., Mailund, T.: A sweep-line method for state space exploration. In: TACAS. Lecture Notes in Computer Science, vol. 2031, pp. 450–464. Springer (2001)
37. Dal Zilio, S.: MCC: A Tool for Unfolding Colored Petri Nets in PNML Format. In: Application and Theory of Petri Nets and Concurrency. Springer (2020). https://doi.org/10.1007/978-3-030-51831-8_23
38. Dalsgaard, A., Enevoldsen, S., Fogh, P., Jensen, L., Jensen, P., Jepsen, T., Kaufmann, I., Larsen, K., Nielsen, S., Olesen, M., Pastva, S., Srba, J.: A distributed fixed-point algorithm for extended dependency graphs. Fundamenta Informaticae **161**(4), 351–381 (2018)
39. Dalsgaard, A., Enevoldsen, S., Fogh, P., Jensen, L., Jepsen, T., Kaufmann, I., Larsen, K., Nielsen, S., Olesen, M., Pastva, S., Srba, J.: Extended dependency graphs and efficient distributed fixed-point computation. In: Proceedings of the 38th International Conference on Application and Theory of Petri Nets and Concurrency (Petri Nets'17). LNCS, vol. 10258, pp. 139–158. Springer-Verlag (2017)
40. Dalsgaard, A.E., Enevoldsen, S., Larsen, K.G., Srba, J.: Distributed computation of fixed points on dependency graphs. In: SETTA. Lecture Notes in Computer Science, vol. 9984, pp. 197–212 (2016)
41. David, A., Jacobsen, L., Jacobsen, M., Jørgensen, K., Møller, M., Srba, J.: TAPAAL 2.0: Integrated development environment for timed-arc Petri nets. In: Proceedings of the 18th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'12). LNCS, vol. 7214, pp. 492–497. Springer-Verlag (2012)
42. David, A., Jacobsen, L., Jacobsen, M., Srba, J.: A forward reachability algorithm for bounded timed-arc Petri nets. In: Proceedings of the 7th International Conference on Systems Software Verification (SSV'12). EPTCS, vol. 102, pp. 125–140. Open Publishing Association (2012)
43. Duret-Lutz, A., Renault, E., Colange, M., Renkin, F., Aisse, A.G., Schlehuber-Caissier, P., Medioni, T., Martin, A., Dubois, J., Gillard, C., Lauko, H.: From Spot 2.0 to Spot 2.10: What's new? In: Proceedings of the 34th International Conference on Computer Aided Verification

(CAV'22). Lecture Notes in Computer Science, vol. 13372, pp. 174–187. Springer (Aug 2022)
44. E.G.Henriksen, Khorsid, A., Nielsen, E., Risager, T., Srba, J., Stück, A., Sørensen, S.: Potency-based heuristic search with randomness for explicit model checking. In: Proceedings of the 29th International SPIN Symposium on Model Checking of Software (SPIN'23). LNCS, vol. 13872, pp. 180–187. Springer-Verlag (2023)
45. Garavel, H.: Nested-Unit Petri Nets. Journal of Logical and Algebraic Methods in Programming **104**, 60–85 (Apr 2019)
46. Geldenhuys, J., Valmari, A.: More efficient on-the-fly LTL verification with tarjan's algorithm. Theor. Comput. Sci. **345**(1), 60–82 (2005)
47. Hanisch, H.: Analysis of place/transition nets with timed arcs and its application to batch process control. In: Marsan, M.A. (ed.) Application and Theory of Petri Nets 1993, 14th International Conference, Chicago, Illinois, USA, June 21-25, 1993, Proceedings. Lecture Notes in Computer Science, vol. 691, pp. 282–299. Springer (1993). https://doi.org/10.1007/3-540-56863-8_52, https://doi.org/10.1007/3-540-56863-8_52
48. He, C., Ding, Z.: More efficient on-the-fly verification methods of colored petri nets. Comput. Informatics **40**(1), 195–215 (2021)
49. Hecher, M., Fichte, J.: The Model Counting Competition Web Page. https://mccompetition.org (July 2023)
50. Heule, M., Jarvisalo, M., Suda, M., Iser, M., Balyo, T.: The International SAT Competition Web Page. http://www.satcompetition.org/ (July 2023)
51. Hillah, L., Kordon, F., Lakos, C., Petrucci, L.: Extending PNML Scope: the Prioritised Petri Nets Experience. In: Petri Net and Software Engineering (PNSE 2011). vol. 723, pp. 61–75. CEUR, Newcastle, UK (June 2011)
52. Jacobsen, L., Jacobsen, M., Møller, M., Srba, J.: A framework for relating timed transition systems and preserving TCTL model checking. In: Proceedings of the 7th European Performance Engineering Workshop (EPEW'10). LNCS, vol. 6342, pp. 83–98. Springer-Verlag (2010)
53. Jensen, J., Nielsen, T., Oestergaard, L., Srba, J.: TAPAAL and reachability analysis of P/T nets. LNCS Transactions on Petri Nets and Other Models of Concurrency (ToPNoC) **9930**, 307–318 (2016)
54. Jensen, P., Larsen, K., Srba, J.: Real-time strategy synthesis for timed-arc Petri net games via discretization. In: Proceedings of the 23rd International SPIN Symposium on Model Checking of Software (SPIN'16). LNCS, vol. 9641, pp. 129–146. Springer-Verlag (2016)
55. Jensen, P., Larsen, K., Srba, J.: PTrie: Data structure for compressing and storing sets via prefix sharing. In: Proceedings of the 14th International Colloquium on Theoretical Aspects of Computing (ICTAC'17). LNCS, vol. 10580, pp. 248–265. Springer (2017)
56. Jensen, P., Larsen, K., Srba, J., Sørensen, M., Taankvist, J.: Memory efficient data structures for explicit verification of timed systems. In: Proceedings of the 6th NASA Formal Methods Symposium (NFM'14). LNCS, vol. 8430, pp. 307–312. Springer-Verlag (2014)
57. Jensen, P., Larsen, K., Srba, J., Ulrik, N.: Elimination of detached regions in dependency graph verification. In: Proceedings of the 29th International SPIN Symposium on Model Checking of Software (SPIN'23). LNCS, vol. 13872, pp. 163–179. Springer-Verlag (2023)
58. Jensen, P., Srba, J., Ulrik, N., Virenfeldt, S.: Automata-driven partial order reduction and guided search for ltl model checking. In: Proceedings of the 23rd International Conference on Verification, Model Checking, and Abstract Interpretation (VMCAI'22). LNCS, vol. 13182, pp. 151–173. Springer-Verlag (2022)
59. Kant, G., Laarman, A., Meijer, J., van de Pol, J., Blom, S., van Dijk, T.: Ltsmin: High-performance language-independent model checking. In: TACAS. Lecture Notes in Computer Science, vol. 9035, pp. 692–707. Springer (2015)

60. Kordon, F., Bouvier, P., Garavel, H., Hulin-Hubard, F., Amat., N., Amparore, E., Berthomieu, B., Donatelli, D., Dal Zilio, S., Jensen, P., Jezequel, L., Paviot-Adet, E., Srba, J., Thierry-Mieg, Y.: Complete Results for the 2023 Edition of the Model Checking Contest. https://mcc.lip6.fr/2023/results.php (April 2023)
61. Kordon, F., Linard, A., Paviot-Adet, E.: Optimized Colored Nets Unfolding. In: 26th International Conference on Formal Methods for Networked and Distributed Systems (FORTE'06). LNCS, vol. 4229, pp. 339–355. Springer Verlag, Paris, France (September 2006)
62. Kordon, F., Hillah, L., Hulin-Hubard, F., Jezequel, L., Paviot-Adet, E.: Study of the efficiency of model checking techniques using results of the MCC from 2015 to 2019. Int. J. Softw. Tools Technol. Transf. **23**(6), 931–952 (2021)
63. Laarman, A.: Stubborn transaction reduction. In: NFM. LNCS, vol. 10811, pp. 280–298. Springer (2018)
64. Lehmann, A., Lohmann, N., Wolf, K.: Stubborn sets for simple linear time properties. In: Petri Nets. Lecture Notes in Computer Science, vol. 7347, pp. 228–247. Springer (2012)
65. Liebke, T., Wolf, K.: Taking some burden off an explicit CTL model checker. In: Petri Nets. Lecture Notes in Computer Science, vol. 11522, pp. 321–341. Springer (2019)
66. Manna, Z., Pnueli, A.: A hierarchy of temporal properties. In: Proceedings of the 9th ACM Symposium on Principles of Distributed Computing (PODC'90). pp. 377–410. ACM Press (Aug 1990)
67. Mateo, J., Srba, J., Sørensen, M.: Soundness of timed-arc workflow nets in discrete and continuous-time semantics. Fundamenta Informaticae **140**(1), 89–121 (2015)
68. Merlin, P.M., Farber, D.J.: Recoverability of communication protocols-implications of a theoretical study. IEEE Trans. Commun. **24**(9), 1036–1043 (1976)
69. de Moura, L.M., Bjørner, N.: Z3: an efficient SMT solver. In: TACAS. LNCS, vol. 4963, pp. 337–340. Springer (2008)
70. Paviot-Adet, E., Poitrenaud, D., Renault, E., Thierry-Mieg, Y.: LTL under reductions with weaker conditions than stutter invariance. In: FORTE. Lecture Notes in Computer Science, vol. 13273, pp. 170–187. Springer (2022)
71. Peterson, G.L.: Myths about the mutual exclusion problem. Information Processing Letters **12**(3), 115–116 (June 1981)
72. Ramchandani, C.: Analysis of asynchronous concurrent systems by timed Petri nets. Ph.D. thesis, MIT, USA (1973)
73. Schmidt, K.: Model-checking with coverability graphs. Formal Methods Syst. Des. **15**(3), 239–254 (1999)
74. Schmidt, K.: Stubborn sets for standard properties. In: ICATPN. Lecture Notes in Computer Science, vol. 1639, pp. 46–65. Springer (1999)
75. Schmidt, K.: How to calculate symmetries of petri nets. Acta Informatica **36**(7), 545–590 (2000)
76. Schmidt, K.: Automated generation of a progress measure for the sweep-line method. In: TACAS. Lecture Notes in Computer Science, vol. 2988, pp. 192–204. Springer (2004)
77. Thierry-Mieg, Y.: Symbolic model-checking using ITS-tools. In: TACAS. LNCS, vol. 9035, pp. 231–237. Springer (2015)
78. Thierry-Mieg, Y.: Structural reductions revisited. In: Petri Nets. LNCS, vol. 12152, pp. 303–323. Springer (2020)
79. Thierry-Mieg, Y.: Symbolic and structural model-checking. Fundam. Informaticae **183**(3-4), 319–342 (2021)
80. Thierry-Mieg, Y.: Efficient strategies to compute invariants, bounds and stable places of petri nets. In: PNSE @ Petri Nets. CEUR Workshop Proceedings, vol. TBA, pp. 16–33. CEUR-WS.org (2023)

81. Thierry-Mieg, Y., Poitrenaud, D., Hamez, A., Kordon, F.: Hierarchical set decision diagrams and regular models. In: TACAS. Lecture Notes in Computer Science, vol. 5505, pp. 1–15. Springer (2009)
82. Valmari, A.: Stubborn sets for reduced state space generation. In: Applications and Theory of Petri Nets. Lecture Notes in Computer Science, vol. 483, pp. 491–515. Springer (1989)
83. Wallner, S., Wolf, K.: Skeleton abstraction for universal temporal properties. Fundam. Informaticae **187**(2-4), 245–272 (2022)
84. Wimmel, H., Wolf, K.: Applying CEGAR to the petri net state equation. Log. Methods Comput. Sci. **8**(3) (2012)
85. Wolf, K.: The petri net twist in explicit model checking. Software & Systems Modeling **14**(2), 711–717 (2015)
86. Wolf, K.: Running lola 2.0 in a model checking competition. Trans. Petri Nets Other Model. Concurr. **11**, 274–285 (2016)
87. Wolf, K.: Petri net model checking with lola 2. In: Petri Nets. Lecture Notes in Computer Science, vol. 10877, pp. 351–362. Springer (2018)
88. Wolf, K.: How petri net theory serves petri net model checking: A survey. Trans. Petri Nets Other Model. Concurr. **14**, 36–63 (2019)
89. Wolf, K.: Portfolio management in explicit model checking. Trans. Petri Nets Other Model. Concurr. **16**, 91–111 (2022)