Exclusion of Non-Divergence in Probabilistic Timed Automata Model Checking

Simon Vreman
s.vreman@student.utwente.nl
University of Twente
Enschede, The Netherlands

ABSTRACT

Model checking is a method to check the correctness of a model of a system. It is important that a model representing a real-world system is verified in a meaningful way, only considering behaviour the real system can exhibit. In probabilistic timed automata, time divergence should always occur, an example of this not happening is infinitely many actions being taken in a finite amount of time, which does not correspond to any real-world system. Digital clocks are often used for model checking but a check for time divergence is required for their result to be meaningful. We show how existing algorithms can be adapted to check for this using a digital clocks approach for model checking. After this, we show the performance impact of this new implementation in terms of runtime. Finally, we show the result of this check on existing benchmark models.

KEYWORDS

Probabilistic timed automata, digital clocks, time divergence, probabilistic model checking

1 INTRODUCTION

Model checking is the automatic verification of a model of a system, a way to check the correctness of a modelled system. This can be a model of a real-world system, automotive control systems or communication protocols for example, whose verification is of interest and where the results should be highly reliable. Discrepancies between the model, the automatic verification, and the real-world system can affect the verification results [6]. One such discrepancy is the behaviour of allowing infinitely many actions to be taken in a finite amount of time, an example of non-divergence. This behaviour is not possible in the real world, but can be modelled in *probabilistic timed automata* (PTA) [18]. Such behaviour is not meaningful and should be disregarded during model checking [19, 20].

An example of such a communication protocol that can be modelled using PTA is Wireless LAN [11]. Then we can check for properties such as the probability of a station's backoff counter reaching some value, a so called reachability property. When checking these properties, the minimum probability of reaching this state is expected to be larger than 0 for certain configurations. But when the model is not time divergent, we can consider a strategy that does allow time to proceed beyond a certain point, never reaching this state. Then our minimum probability is 0, which is not a meaningful result because time will always progress in the real world.

When modelling using PTA, it is often useful to abstract them to Markov decision processes (MDP). For this, both a dense time model and an integral model, digital clocks [13], can be used. Digital clocks are a reduction that can be applied to a large class of systems and can provide a meaningful improvement in performance [17] compared to *forward reachability* [19] and *backwards reachability* [18]. It is one of the approaches implemented in the Modest Toolset [10], which this research aims to extend. The toolset does not, however, check for divergence when using digital clocks for model checking.

An algorithm for checking for probabilistic divergence in PTA has been proposed by Sproston [21]. We show this algorithm can be adapted to the digital clocks approach for model checking. To achieve this, we design and implement a prototype into the Modest Toolset's model checker *mcsta* [7]. We test it with PTA models from the *Quantitative Verification Benchmark Set* (QVBS) [12] and our own developed examples to verify the correctness of the implementation. The prototype implementation will be discussed in section 6.

We also analyse the performance impact of this new implementation in terms of runtime. We use the same benchmark models from the QVBS to determine the effect of executing the model checker with and without the check for non-divergence. Furthermore, we also show the result of the non-divergence check on these models to see which of these exhibit this behaviour. The performance analysis and result of the check will also be discussed in section 6.

In the next section, relevant topics that are used throughout this paper are summarized. Then in section 3 we will discuss the contributions and research questions. After that, in section 4, we discuss related work. In section 5, we will discuss the methodology used for solving our questions. We will continue with the results in section 6 and finally, in section 7, we will discuss conclusions from this research and future work.

2 BACKGROUND

This section will provide some background information on the concepts used in this research. Throughout this paper we use $\mathbb{N}$ to denote the set of natural number and AP to denote a set of atomic propositions. The set of discrete probability distributions over a set S is denoted by $\mu(S)$. Each $p \in \mu(S)$ is thus a function $p : S \rightarrow [0, 1]$ such that $\sum_{s \in S} p(s) = 1$.

2.1 Markov decision processes

A *Markov decision process* (MDP) is a tuple $(S, \rightarrow, lab)$ [17, 18] where S is the set of states, $\rightarrow \subseteq S \times \mu(S)$ is the transition relation, and lab is the labelling function $lab : S \rightarrow 2^{AP}$. The state space is traversed from current state s by a nondeterministic selection of $(s, p) \in \rightarrow$, then a probabilistic choice made using p, with the probability of transitioning to $s' \in S$ being $p(s')$.

41th Twente Student Conference on IT, July 5, 2024, Enschede, The Netherlands

2.2 Probabilistic timed automata

We define *probabilistic timed automata* (PTA) here using *digital clocks* [17], with our time domain being $\mathbb{N}$, the natural numbers. Given a set of clocks $\mathcal{X}$ which take values from this time domain, we have the set $CC(\mathcal{X})$ of clock constraints of the form $x \sim c$ where $x \in \mathcal{X}$, $c \in \mathbb{N}$ and $\sim \in \{\leq, =, \geq\}$. Then we define a PTA as a tuple $(L, l_0, \mathcal{X}, inv, prob, \mathcal{L})$ [18, 21] where:

- L is a finite set of locations
- $l_0 \in L$ is the initial location
- $\mathcal{X}$ is a finite set of clocks
- inv is a function $inv : L \to CC(\mathcal{X})$ that assigns an invariant to each location
- $prob$ is a finite set $prob \subseteq L \times CC(\mathcal{X}) \times \mu(2^{\mathcal{X}} \times L)$ of probabilistic edges such that for each $l \in L$ there exists a *probabilistic edge* $(l, g, p) \in prob$ where g is a clock constraint, and p is a probability distribution
- $\mathcal{L}$ is a labelling function $\mathcal{L} : L \to 2^{AP}$

In this automaton, time can advance in a location as long as the invariant holds, and a probabilistic edge can be taken so long as its clock constraint, in the context of a probabilistic edge called a *guard*, is satisfied by the clocks.

2.3 Reachability properties

The main measure for PTA we are interested in is *probabilistic reachability*, where we make a distinction between the minimal and maximal probability of reaching, from the initial state l_0, a set of target states $T \subseteq L$. To reason about this, we use the notion of an *adversary*, which is a resolution of all nondeterministic choices in the model. The minimum reachability is then the probability of reaching T under the least favourable adversary, and the maximum reachability is the probability of reaching T under the most favourable adversary. [17]

2.4 Maximal end components

We start by defining *sub-MDPs*, which, given an MDP $(S, \to, lab)$, is a pair (C, D) where $C \subseteq S$ and $D \subseteq \to$. We also define *edge relations*: there is an edge (s_C, s_S) from $s_C \in C$ to $s_S \in S$ if it is possible to go from s_C to s_S in one step with positive probability. Then we define the edge relation ρ, where $\rho_{(C,D)}$ is the set of tuples (s_C, s_S) for which there exists $t \in D(s_C)$ such that $p(s_S) > 0$, where p is the set of discrete probability distributions over S such that $(s_C, p) \in D$. A sub-MDP (C, D) is also an *end component* (EC) if:

- For every successor s' of $s \in C$, using transition $t \in D$, $s' \in C$
- The graph $(C, \rho_{(C,D)})$ is strongly connected

Intuitively, this means that an adversary can choose to stay in the end component indefinitely by selecting the right transitions.

The state-action set of a sub-MDP (C, D) is denoted by $sa(C, D) = \{(s, t) \mid s \in C \wedge t \in D(s)\}$. An end component (C, D) is *maximal* in an MDP $(S, \to, lab)$ if there is no other end component (C', D') such that $sa(C, D) \subset sa(C', D') \subseteq (C, D)$. These so-called *maximal end components* (MECs) are crucial for our approach to check for divergence, which we will discuss further in section 5.

The *maximal end component decomposition* (MEC decomposition) of an MDP consists of all the MECs of the MDP, and all the states that do not belong to any MEC. This partitions C, with no state belonging to more than one MEC, because when for ECs (C, D) and (C', D'), some $C \cap C' \neq \{\}$, then their union is also an EC. Following from the definition of MECs, this means that any state in some MEC, must not be part of any other MEC. [1, 5]

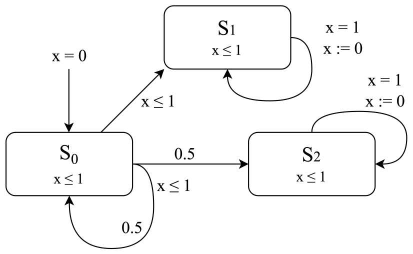

Figure 1: Reachability under strict and probabilistic divergence. [21]

2.5 Probabilistic and strict divergence

In the context of PTA, often a notion of probabilistic time divergence is used, that is, time diverges with probability 1 for all strategies of the model. This is often sufficient for model checking purposes. However, a stricter notion of time divergence can also be used, as was proposed by Sproston [21]. This notion is called strict divergence, and requires that all behaviours of an adversary should be time divergent, not just probability 1 of the behaviours.

The relevance of this distinction can be seen in Figure 1. There exist adversaries that continuously select the probabilistic transition from s_0, preventing time from proceeding beyond a certain point. Following the notion of probabilistic divergence, the maximum probability of reaching s_2 is 1, as the adversary will always eventually result in reaching s_2. However, there is no adversary under strict divergence that will always eventually reach s_2, thus the maximum probability will always be less than 1.

In this research, we focus on checking for probabilistic divergence. When referring to time divergence from now on, it can be assumed we are referring to probabilistic divergence.

3 PROBLEM STATEMENT

PTA can be model checked using the Modest Toolset [4], but this toolset only checks for divergence when using the backwards reachability algorithm [19]. This algorithm is quite slow in practise, which is why the digital clocks [17] approach is used by default for checking PTA in the Modest Toolset. However, this algorithm does not check for divergence. Identifying non-divergence has been shown to be possible [21] and this research shows how this can be adapted to a digital clocks approach in the Modest Toolset. Subsequently, we will also show the performance impact of adding this check by analysing existing benchmark models.

- **Goal 1**: Identify PTA that exhibit non-divergence behaviour and produce incorrect results in the Modest Toolset.
- **Goal 2**: Include the identification of non-divergence in PTA using an adaptation of the algorithm by Sproston [21], tailored to digital clocks, in the statistical model checker of the Modest Toolset [4].
- **Goal 3**: Analyze the performance impact in terms of runtime of checking for divergence in the Modest Toolset using this new implementation.

3.1 Research questions

The following research questions are used as the basis of our research:

- **RQ 1**: How can existing algorithms be adapted and optimized for detecting non-divergence in probabilistic timed automata during model checking using digital clocks?
- **RQ 2**: What is the performance impact in terms of runtime of including the check from **Goal 2** in the Modest Toolset?

4 RELATED WORK

In this section we will discuss work related to time divergence in model checking using digital clocks.

In their 2002 paper Kwiatkowska et al [18] propose the forward reachability algorithm for the verification op PTA. This is based on a symbolic forwards exploration, and can be implemented efficiently with certain data structures such as difference-bound matrices [15]. However, for PTA specifically, it only yields an upper bound on the maximum reachability probabilities.

Another approach for model checking PTA is the backwards reachability algorithm [19], which involves state-space exploration from the target states to the initial states. Unlike the forward reachability approach, this algorithm does yield exact minimum and maximum reachability probabilities. However, the operations are expensive which limits its capabilities.

The stochastic games approach by Kwiatkowska et al in their 2009 paper [15] for the analysis of PTA guarantees time divergent behaviour. However, stochastic games are not implemented in Modest. They also show that, compared to other approaches of verifying PTA including digital clocks, this approach has superior performance and scalability.

The PRISM model checker [16] is a tool for model checking PTA, among other purposes, which has considerable overlap with parts of the Modest Toolset including the *mcsta* tool. In PRISM digital clocks can also be used for model checking, but the default is the stochastic games approach. As shown in the paper above, on stochastic games, it guarantees time divergent behaviour.

Another tool for model checking PTA is the UPPAAL tool [2], which is also able to compute reachability properties. A continuous time model is used, not the digital clocks approach. A tool to convert a Modest *timed automata* model to a UPPAAL model, *mctau* [3], is available.

5 METHODOLOGIES

To solve our first research question we determined how probabilistic non-divergence behaviour can be identified during model checking using digital clocks, then we created a prototype of this in the Modest Toolset. We then analysed the performance impact of this implementation to answer our second research question. Finally, we reviewed existing benchmark models for PTA and executed our prototype on these to determine the presence of probabilistic divergence.

5.1 Theoretical background

The first step in our check is to transform the PTA into an MDP, which is a common approach for model checking PTA. Further analysis can then be done on this resulting model. During this transformation, the concept of time is lost, which we do need in our check. The passing of time in a PTA is represented in the resulting MDP $(S, \rightarrow, lab)$ by a transition $(s, p) \in\rightarrow$, for which $p(s') = 1$ for some state $s \in S$. This is no different from how a probabilistic transition with a single target state in the original PTA is represented in the MDP. Thus we have to make a modification here to keep track of this information, before continuing.

As discussed in section 2.4, for any EC an adversary can be given that will always select a transition $t \in\rightarrow$ for which the target state $s' \in S$ is in the same EC with probability 1 while taking each state-transition pair infinitely often [1]. Because the MEC decomposition partitions S, and there is no possibility to reach the same MEC again with probability 1 after transitioning out of it, we know that if a strategy exists that is divergent, it has to eventually be contained to a MEC in which time diverges. This is a key insight, based on the work by Sproston [21] and Alfaro [1], that we will use to check for non-divergence. We can compute the MEC decomposition in multiple ways, for example the algorithm by Chatterjee et al [5].

The final step in our check is to put these pieces together. Using the transformed MDP, the knowledge of which transition $t \in\rightarrow$ corresponds with the passing of time and the MEC decomposition, we have all the information needed to identify this behaviour. For each MEC (C, D), we identify if there is any transition $t \in D$ that is a transition corresponding with the passing of time. Because when there is such a transition, it will be taken infinitely by an adversary that is eventually confined to the MEC, meaning that time must divergence [1]. Then the result of our check is simply the presence of a MEC that does not have such a transition, and thus is not divergent. If all MECs are divergent, the original PTA is divergent.

5.2 Constructing a prototype

To construct a prototype of this algorithm, we used the theoretical background and combined this with the existing functionality in the Modest Toolset. For the conversion of PTA to MDP, we used the existing functionality in the *mcsta*, with only a minor modification needed to keep track of which transitions in the resulting MDP correspond with the passing of time in the original PTA. We did not implement the computation of the MEC decomposition in our prototype, as the present calculation for MECs in Modest was deemed sufficient for our purposes.

Then we modelled examples of PTA that exhibit this behaviour, and their correct counterparts. A very basic example of a PTA that includes this behaviour, and does in some cases cause *mcsta* to provide an incorrect[1] probability, is shown in Figure 2. We highlighted the rightmost transition t_1 using a dashed line, as the addition of this transition makes the PTA probabilistically non-divergent. Namely, we can formalize a strategy that upon reaching s_1 will always transition to s_2, and then immediately back to s_1. The minimum probability of ever reaching s_3 is thus 0, if we consider this as a valid strategy. This type of behaviour is what we want to exclude from our model checking as it does not represent real-world behaviour.

Shown in Figure 3 is a model of this behaviour written in the Modest language [8]. Here we use a variable *state* which corresponds with the state numbers in Figure 2. Again we highlight the added transition t_1 that causes the model to be probabilistically non-divergent. THis is done by removing the transition which updates *state* to 3, and replacing it with a deterministic choice between updating *state* to 3 or 1. The strategy we reasoned about earlier would then be to always update *state* to 1, and thus never reach s_3.

These examples are very simplistic and not too interesting in themselves, so we also gathered models from the QVBS [12] to

[1]It can be considered that correct models already exclude this behaviour, and that the output is not in fact incorrect but rather expected, given the incorrect input.

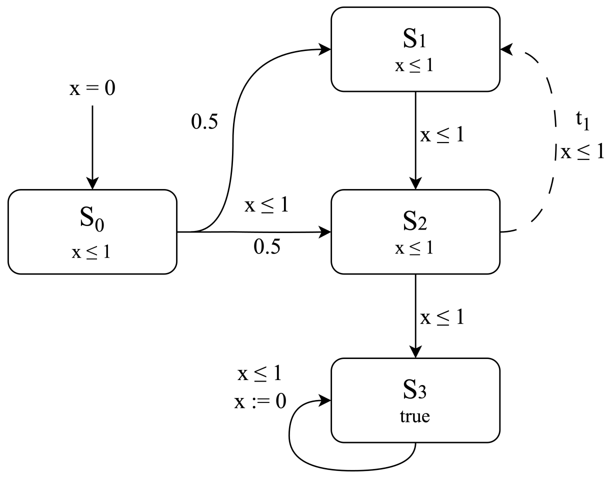

Figure 2: An example of probabilistic (non)-divergence.

```
  int state = 0;
  clock c;

  property P_Reached_s3 = Pmin(<> (state == 3));

  do {
    alt {
      :: when(state == 0) constrain (c <= 1) palt {
        :50: {= state = 1 =}
        :50: {= state = 2 =}
      }
      :: when(state == 1) constrain (c <= 1)
        {= state = 2 =}
      :: when(state == 2) constrain (c <= 1)
-       {= state = 3 =}
+       alt {
+         :: {= state = 3 =}
+         :: {= state = 1 =}
+       }
      :: when(state == 3) {= c = 0 =}
    }
  }
```

Figure 3: A Modest model of probabilistic (non)-divergence.

complement our own examples. The set contains diverse probabilistic models that can be used to benchmark the performance of tools for model checking.

Finally we implemented a prototype of this algorithm, this implementation will be discussed in section 5. Using the examples we built and collected we confirmed that the prototype works as expected.

5.3 Analysing the performance impact

For the performance analysis, we used benchmark models from the QVBS [12]. The selection we used, the parameters provided and the resulting amount of states in the MDP are shown in Table 1. The parameters are interesting for reproduction mostly, whereas the resulting state count is useful to get a grasp of how complex these models are relative to each other, with especially the *firewire-pta* and *wlan-large* models resulting in large MDPs.

Table 1: Model parameters used for the analysis

Model	Parameters	States
brp-pta	N = 16, MAX = 2, TD = 1, TIME_BOUND = 64	3959
firewire-pta	delay = 30, T = 2500	4432272
firewire_abst-pta	delay = 30, T = 5000	6020
wlan-large	K = 2	3283371
zeroconf-pta	T = 150	498

Our selection consists of PTA models supported by *mcsta* only[2], as this is the tool the prototype was built for.

We came to these parameters by checking the documentation of the QVBS and selecting of the examples, where available. We don't consider the selection of the exact parameters to be of importance for this research, as we are interested in the relative performance change of the tool in terms of runtime. For every model, we initiate property checking for every property in the model. As a consequence, the results are influenced by the number of properties in the model. We consider this to not be a problem, as the property definition itself also has an impact on the runtime, as well as the model. We then compiled the Modest Toolset with our prototype implementation, and a version without this check, being completely identical otherwise.

First, for each model, we prepared two benchmarking instances, one using the version with our prototype, and one using the unmodified version. We wrote a benchmarking script to repeat each benchmarking instance 25 times, and time the duration of each run. We did not complete property checking entirely, but aborted each run after executing the non-divergence check, or in the case of the unmodified version, at the point where the check would be executed. This is done because there is no influence on the runtime after the check is done, thus the absolute runtime difference would be unchanged, and aborting execution allows us to also measure the impact on larger models.

After this, we also prepared two other benchmarking instances for both the brp-pta and zeroconf-pta model, this time not aborting property checking at any point but fully completing execution to determine the relative performance difference. We repeated these instances 50 times each.

We also analysed the relationship between the number of states in the MDP and the runtime of the non-divergence check. For this we used the brp-pta model [9] and varied the input parameters to control the state count. The properties being checked were kept identical, and the same benchmarking tool was used again to obtain the difference in execution time. As we again are looking for the absolute difference, execution is aborted after the check is done. In this case, we used five iterations for each parameter set. We kept the maximum number of retransmissions per frame MAX and TIME_BOUND parameters constant at 2 and 64 respectively, and varied the number of frames per file N and the transmission delay TD.

6 RESULTS

In this section we will show the results of our research, starting with the prototype implementation and then a performance analysis of our prototype to learn about the effect on runtime.

[2]We also include the wlan-large model, which is a stochastic timed automaton, but is converted to a PTA by *mcsta* before we execute our check [7].

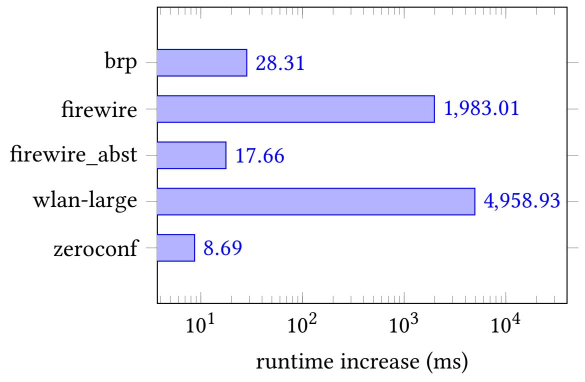

Figure 4: Performance change of the prototype implementation

We will also show the result of our non-divergence check on the benchmark models, to see which of these exhibit this behaviour.

6.1 Prototype

Based on the work outlined in section 5, we constructed a prototype implementation of the algorithm to check for non-divergence. The pseudocode for this algorithm is shown in Algorithm 1. There are some simplifications related to optimisation that are not shown in the pseudocode, for presentation purposes, namely early breaks when further checking of a state, transition or branch is not necessary. When we refer to a state s, transition t or branch b in the pseudocode, we refer to $s \in S$, $t = (s, p) \in \longrightarrow$ for some s and $b \in \{s' \in S \mid p(s') > 0\}$ for some t respectively.

Here we check for branches to determine if a transition results in a target state within the same MEC with probability 1. We also check for transitions that do not correspond with the passing of time, this is done to differentiate between MECs, and states that are not part of any MEC. After all, these are included in our MEC decomposition, but cannot result in non-divergence. This differentiation relies on the fact that a state that does not have any transition leading back to, a state in the same MEC, or more correctly itself, cannot be part of a MEC, as any adversary would leave upon taking any transition.

The prototype was implemented in the Modest Toolset's model checker *mcsta*, and reuses the existing functionality for the conversion of PTA to MDP as well as the computation MEC decomposition. Determining which MECs are time-divergent is done by checking bottom-up, starting with the set of states S and checking the transitions origination from it to find which correspond with the passing of time and thus make the MEC non-divergent.

We tested this prototype on the examples we built and collected, including those shown in Figures 1 and 2, which we discussed in subsections 2.5 and 5.2 respectively. We also tested it on the benchmark models, adding additional transitions to create non-divergent counterparts for those that were divergent. The results were as expected, with the prototype working correctly.

6.2 Performance

We were able to gather data for various models from QVBS, the results being the mean over 25 repetition for each instance, corresponding to a specific model and version of *mcsta*. Each model was benchmarked with and without the check for non-divergence subsequently.

Algorithm 1 Checking of MECs for divergence

```
$S \leftarrow$ set of states
$T_s \leftarrow$ set of transitions of s
$B_t \leftarrow$ set of branches of t
$M \leftarrow$ set of MECs
$D \leftarrow \{\}$                                  ▷ Set of divergent MECs
for all $s \in S$ do
    $hasTickTransition \leftarrow false$
    $hasNonTickTransition \leftarrow false$
    for all $t \in T_s$ do
        $hasExternalBranch \leftarrow false$
        for all $b \in B_t$ do
            if IsExternalBranch($b$) then
                $hasExternalBranch \leftarrow true$
            end if
        end for
        if hasExternalBranch then
            continue
        end if
        if IsTickTransition($t$) then
            $hasTickTransition \leftarrow true$
        else
            $hasNonTickTransition \leftarrow true$
        end if
    end for
    if hasTickTransition or not hasNonTickTransition then
        $D \leftarrow D \cup \{$FindMEC($s$)$\}$
    end if
end for
return $|D| = |M|$                   ▷ If equal, all MECs are divergent
```

The absolute performance results of the prototype implementation for each model is shown in Figure 4. Shown are the mean differences in runtime for each model. It should be noted that these results are also affected by the amount of properties checked, as the non-divergence check is performed once for each property. We kept this consistent between the two versions of the tool within the same model. We see significant differences in performance between the models, note the logarithmic scale. This is expected for our implementation, as the time complexity is dependent on the size of the MDP as shown by Sproston [21]. Intuitively, this also makes sense from the implemented algorithm we presented in the previous subsection, which is based on the states, transitions and branches of the state space.

The relation between the number of states resulting from the MDP conversion and the increase in runtime in shown in Figure 5. Again we use logarithmic scales, to align with the measurements we gathered. A promising observation here is that the time complexity of both version appears similar, with both going through state-space exploration. Thus the time complexity does not appear to explode when adding the non-divergence check. The difference in execution time is also shown, here we see similar results. The data was gathered for the brp-pta model, as discussed in the methodology section. Interesting to note here is that the amount of states, transitions and branches in the MDP grow proportionally to each other. Thus we see the same relationship regardless, in this case we show the number of states. The properties that were used for each benchmarking instance, the resulting number of states and associated runtime are shown

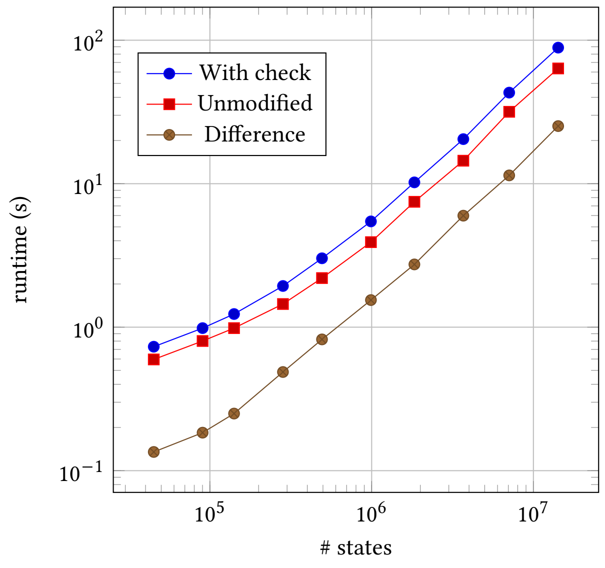

Figure 5: Runtime of check by number of states

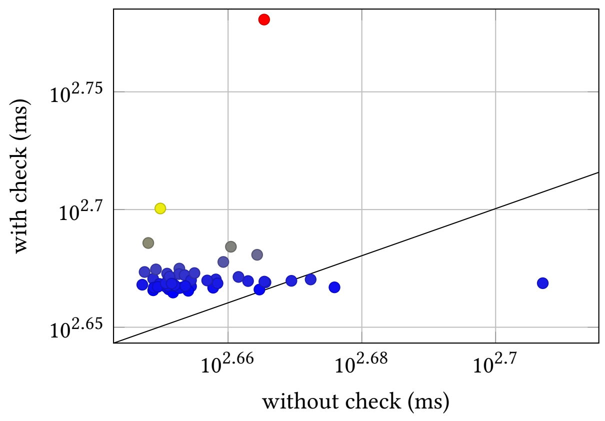

Figure 6: Total runtime comparison for brp-pta

Table 3, included in appendix A. Here we can also see the exact differences between the instances in seconds.

Finally, we also compared the total runtime of the brp-pta and zeroconf-pta to see what the relative performance difference is across the entire model checking process. The results are shown in Figure 6 and Figure 7, where each point corresponds to a set of benchmarking runs, with the result of the instance without the check on the x-axis and the result of the instance with the check on the y-axis. We see a clear deviation from the black line $y = x$, with most points showing a higher runtime, $y > x$, for the instance with the check, as should be expected. There are also multiple significant outliers, for which we do not know the exact cause. The benchmarking instances should behave deterministically, thus we expect the source of these outliers to be related to computer on which we benchmarked. It is likely that the benchmarking tool simply got less CPU time from the operation system in these cases, thus they are not particularly interesting.

Overall, we see a 3.86% mean increase in total runtime for the brp-pta model, and a 2.46% mean increase for the zeroconf-pta model. This looks promising, with the performance impact being relatively low.

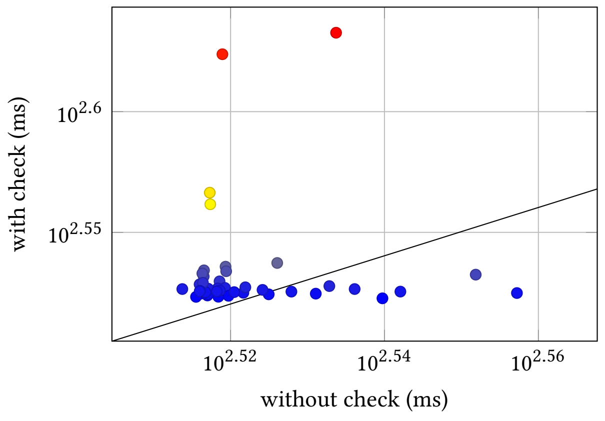

Figure 7: Total runtime comparison for zeroconf-pta

Table 2: Result of the non-divergence check on benchmark models

Model	Probabilistically divergent
brp-pta	yes
firewire-pta	yes
firewire_abst-pta	yes
wlan-large	yes
zeroconf-pta	no

6.3 Non-divergence in benchmark models

We ran the prototype implementation on the benchmark models from the QVBS to see what the result of the non-divergence check would be. The results are shown in Table 2. We consider the model to be probabilistically divergent if for all properties defined, our check considers the model divergent[3]. As can be seen, only one of the models, zeroconf-pta [14], does not pass our check for probabilistic divergence. The result means that for all properties defined in the model, we do not find probabilistic non-divergence.

In the case of zeroconf-pta, we find that the check fails for both the deadline and incorrect properties, with no property passing the check. Manual observation of both cases reveals that the MDP resulting from this model indeed has multiple single-state MECs that have a non-tick self loop, thus not satisfying probabilistic time divergence.

7 CONCLUSION

In this research we have shown that it is possible to check for probabilistic time divergence during model checking of PTA using digital clocks. We have provided a prototype implementation of this in the Modest Toolset, and analysed the performance impact of this implementation. We have also shown which existing models from the QVBS exhibit this behaviour.

To answer our first research question, we adapted the algorithm by Sproston to the digital clocks approach used in the Modest Toolset. We then implemented this as a prototype in the *mcsta* tool. We tested this implementation on our own examples

[3]If one defines a property that does not require model checking of certain parts of the model in which non-divergence can occur, our check concludes that the model is probabilistically divergent for that specific property, though one might define another property which is then not divergent

and models from the QVBS to verify the correctness of the prototype. This approach can be used to identify non-divergence in PTA when model checking using digital clocks.

For our second research question, we analysed the performance impact of our prototype implementation. We used benchmark models from the QVBS to determine the increase in execution time of the model checker. We have shown that although significant, the increase in runtime is manageable. Furthermore, we have compared the runtime of the brp-pta model with varying parameters to show the impact of the size of the state space on execution time. Here we observed that the inclusion of our check does not appear to have significantly altered the time complexity, with our check yielding results even on large models. We have also shown that for the brp-pta and zeroconf-pta benchmark models, the increase in runtime is only a few percent.

7.1 Future work

In future work, one could consider the implementation of the strict divergence check as well. This would require a more complex algorithm, which we were not able to implement in the scope of this research. As we discussed in section 2, the difference between probabilistic and strict divergence is useful, and such a check could be beneficial.

Another interesting avenue for future research is automatically excluding non-divergent behaviour from the model. One approach could be to alter the properties of the model to correct for this behaviour. We would then calculate the probability of a property, excluding the non-divergent behaviour.

REFERENCES

[1] Luca Alfaro. 1997. *Formal Verification of Probabilistic Systems.* Technical Report. Stanford, CA, USA.

[2] Gerd Behrmann, Alexandre David, Kim Larsen, John Håkansson, Paul Pettersson, Wang yi, and Martijn Hendriks. 2006. Uppaal 4.0. *Third International Conference on the Quantitative Evaluation of Systems, QEST 2006*, 125–126. https://doi.org/10.1109/QEST.2006.59

[3] Jonathan Bogdoll, Alexandre David, Arnd Hartmanns, and Holger Hermanns. 2012. mctau: Bridging the Gap between Modest and UPPAAL. In *Model Checking Software - 19th International Workshop, SPIN 2012, Oxford, UK, July 23-24, 2012. Proceedings (Lecture Notes in Computer Science, Vol. 7385)*, Alastair F. Donaldson and David Parker (Eds.). Springer, 227–233. https://doi.org/10.1007/978-3-642-31759-0_16

[4] Carlos E. Budde, Pedro R. D'Argenio, Arnd Hartmanns, and Sean Sedwards. 2018. A Statistical Model Checker for Nondeterminism and Rare Events. In *Tools and Algorithms for the Construction and Analysis of Systems - 24th International Conference, TACAS 2018, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2018, Thessaloniki, Greece, April 14-20, 2018, Proceedings, Part II (Lecture Notes in Computer Science, Vol. 10806)*, Dirk Beyer and Marieke Huisman (Eds.). Springer, 340–358. https://doi.org/10.1007/978-3-319-89963-3_20

[5] Krishnendu Chatterjee and Monika Henzinger. 2011. Faster and Dynamic Algorithms For Maximal End-Component Decomposition And Related Graph Problems In Probabilistic Verification. *Proceedings of the Annual ACM-SIAM Symposium on Discrete Algorithms*, 1318–1336. https://doi.org/10.1137/1.9781611973082.101

[6] Edmund Clarke, Orna Grumberg, and Doron Peled. 2001. *Model Checking.*

[7] Ernst Moritz Hahn, Arnd Hartmanns, and Holger Hermanns. 2014. Reachability and Reward Checking for Stochastic Timed Automata. *Electron. Commun. Eur. Assoc. Softw. Sci. Technol.* 70 (2014). https://doi.org/10.14279/TUJ.ECEASST.70.968

[8] Ernst Moritz Hahn, Arnd Hartmanns, Holger Hermanns, and Joost-Pieter Katoen. 2013. A compositional modelling and analysis framework for stochastic hybrid systems. *Formal Methods Syst. Des.* 43, 2 (2013), 191–232. https://doi.org/10.1007/S10703-012-0167-Z

[9] Arnd Hartmanns and Holger Hermanns. 2009. A Modest Approach to Checking Probabilistic Timed Automata. In *2009 Sixth International Conference on the Quantitative Evaluation of Systems.* 187–196. https://doi.org/10.1109/QEST.2009.41

[10] Arnd Hartmanns and Holger Hermanns. 2014. The Modest Toolset: An Integrated Environment for Quantitative Modelling and Verification. In *Tools and Algorithms for the Construction and Analysis of Systems - 20th International Conference, TACAS 2014, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2014, Grenoble, France, April 5-13, 2014. Proceedings (Lecture Notes in Computer Science, Vol. 8413)*, Erika Ábrahám and Klaus Havelund (Eds.). Springer, 593–598. https://doi.org/10.1007/978-3-642-54862-8_51

[11] Arnd Hartmanns and Holger Hermanns. 2015. Explicit Model Checking of Very Large MDP Using Partitioning and Secondary Storage. In *Automated Technology for Verification and Analysis*, Bernd Finkbeiner, Geguang Pu, and Lijun Zhang (Eds.). Springer International Publishing, Cham, 131–147.

[12] Arnd Hartmanns, Michaela Klauck, David Parker, Tim Quatmann, and Enno Ruijters. 2019. The Quantitative Verification Benchmark Set. In *Tools and Algorithms for the Construction and Analysis of Systems*, Tomáš Vojnar and Lijun Zhang (Eds.). Springer International Publishing, Cham, 344–350.

[13] Thomas A. Henzinger, Zohar Manna, and Amir Pnueli. 1992. What good are digital clocks?. In *Automata, Languages and Programming*, W. Kuich (Ed.). Springer Berlin Heidelberg, Berlin, Heidelberg, 545–558.

[14] Marta Kwiatkowska, Gethin Norman, and David Parker. 2012. The PRISM Benchmark Suite. In *2012 Ninth International Conference on Quantitative Evaluation of Systems.* 203–204. https://doi.org/10.1109/QEST.2012.14

[15] Marta Kwiatkowska, Gethin Norman, and David Parker. 2009. Stochastic Games for Verification of Probabilistic Timed Automata. In *Formal Modeling and Analysis of Timed Systems*, Joël Ouaknine and Frits W. Vaandrager (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 212–227.

[16] Marta Kwiatkowska, Gethin Norman, and David Parker. 2011. PRISM 4.0: Verification of Probabilistic Real-Time Systems. In *Computer Aided Verification*, Ganesh Gopalakrishnan and Shaz Qadeer (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 585–591.

[17] Marta Kwiatkowska, Gethin Norman, David Parker, and Jeremy Sproston. 2006. Performance analysis of probabilistic timed automata using digital clocks. *Formal Methods in System Design* 29, 1 (01 Jul 2006), 33–78. https://doi.org/10.1007/s10703-006-0005-2

[18] Marta Kwiatkowska, Gethin Norman, Roberto Segala, and Jeremy Sproston. 2002. Automatic verification of real-time systems with discrete probability distributions. *Theoretical Computer Science* 282, 1 (2002), 101–150. https://doi.org/10.1016/S0304-3975(01)00046-9 Real-Time and Probabilistic Systems.

[19] Marta Kwiatkowska, Gethin Norman, Jeremy Sproston, and Fuzhi Wang. 2007. Symbolic model checking for probabilistic timed automata. *Information and Computation* 205, 7 (2007), 1027–1077. https://doi.org/10.1016/j.ic.2007.01.004

[20] Gethin Norman, David Parker, and Jeremy Sproston. 2013. Model checking for probabilistic timed automata. *Formal Methods in System Design* 43, 2 (01 Oct 2013), 164–190. https://doi.org/10.1007/s10703-012-0177-x

[21] Jeremy Sproston. 2009. Strict Divergence for Probabilistic Timed Automata. In *CONCUR 2009 - Concurrency Theory*, Mario Bravetti and Gianluigi Zavattaro (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 620–636.

A PARAMETERS AND RESULTS STATE-RUNTIME BENCHMARK

We include the parameters and results used for the runtime by state count benchmark on the brp-pta in Table 3.

Table 3: Difference in runtime depending on number of states

Parameters	States	With check	Mean (s)	Increase (s)
N = 32, TD = 4	44967	no	0.731	0.135
		yes	0.596	
N = 64, TD = 4	90119	no	0.985	0.184
		yes	0.801	
N = 32, TD = 8	141037	no	1.235	0.250
		yes	0.985	
N = 64, TD = 8	282893	no	1.937	0.487
		yes	1.450	
N = 32, TD = 16	493161	no	3.021	0.822
		yes	2.199	
N = 64, TD = 16	989705	no	5.460	1.545
		yes	3.915	
N = 32, TD = 32	1837345	no	10.216	2.738
		yes	7.479	
N = 64, TD = 32	3688385	no	20.446	5.986
		yes	14.461	
N = 32, TD = 64	7085457	no	43.125	11.426
		yes	31.699	
N = 64, TD = 64	14225969	no	88.765	25.203
		yes	63.562	