Modelling the Confidence of Timing Analysis for Time Randomised Caches

Pedro Benedicte[‡,†], Leonidas Kosmidis[‡,†], Eduardo Quiñones[†], Jaume Abella[†], Francisco J. Cazorla[†,*]
[‡]Universitat Politècnica de Catalunya (UPC), Spain
[†]Barcelona Supercomputing Center (BSC-CNS), Spain
[*]Spanish National Research Council (IIIA-CSIC), Spain

***Abstract*—Timing is a key non-functional property in embedded real-time systems (ERTS). ERTS increasingly require higher levels of performance that can only be sensibly provided by deploying high-performance hardware, which however complicates timing analysis. Measurement-Based Probabilistic Timing Analysis (MBPTA) aims at analysing the timing behaviour of ERTS deploying complex hardware features such as caches. A key parameter for MBPTA to provide reliable results is the number of runs to perform to ensure *probabilistic representativeness* of the execution time measurements taken at analysis time with respect to execution times that can occur during system operation. In this paper, focusing on the cache – acknowledged as one of the most complex resources to time analyse – we address the problem of determining whether the number of observations taken at analysis, as part of the normal MBPTA application process, captures the cache events significantly impacting execution time and Worst-Case Execution Time (WCET). If this is not the case, our techniques provide the user with the number of extra runs to perform to guarantee that those cache events are captured ensuring confidence on provided WCET estimates.**

I. Introduction

The advent of unmanned vehicles and criticality-related on-board features makes that Embedded Real-Time Systems (ERTS) increasingly deal with highly sophisticated – and complex – value-added software functionalities. These, in turn, require higher levels of computing power to be timely executed. High-performance hardware is the natural way to respond to these performance needs, but it is well-known that it challenges timing analysis techniques which make pessimistic, yet reliable, assumptions on resource latencies, resulting in longer (degraded) Worst-Case Execution Time (WCET) estimates [2].

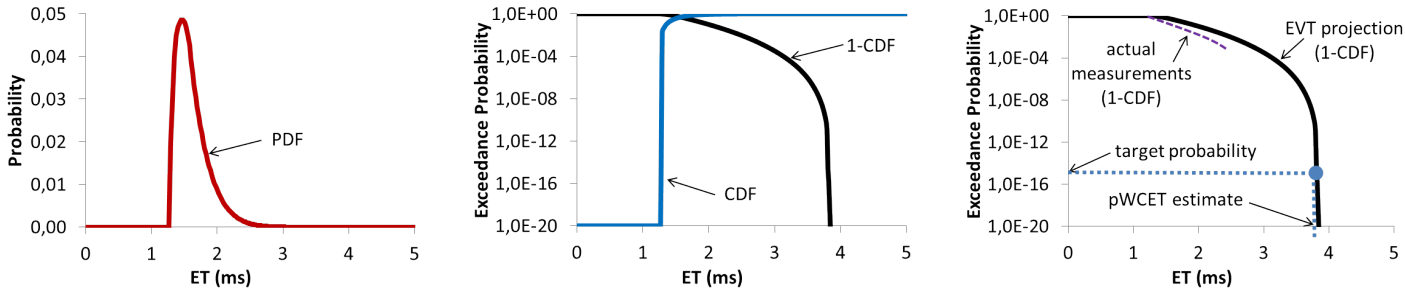
Measurement-Based Probabilistic Timing Analysis (MBPTA) [5] deals with complex hardware while staying close to industrial timing analysis practice. MBPTA, whose potential viability in industrial setups has been positively assessed [27], [28], provides a probabilistic WCET (pWCET) estimate: a distribution expressing the *residual risk* [12], [24] (in the form of a probability) with which one instance of the program is proven not to exceed a given execution time bound. That probability is made low enough to be in line with the standards in the application domain that dictate the degree of rigor required in system assurance.

MBPTA deploys Extreme Value Theory [6], [18] (EVT) on a set of execution time observations captured in the analysis tests. From those observations, whose number is maintained in the range of thousands to keep the analysis cost affordable, EVT estimates bounds on the timing behaviour of tasks during operation for much smaller probabilities, e.g. 10^{-15} per run. The use of EVT in MBPTA is challenged by the fact that the execution time observations used for the prediction are collected at analysis time, while the pWCET estimate must provide a reliable upper-bound during operation. This requires dealing with representativeness [4] such that *evidence is provided on the fact that analysis time observations capture the impact of those events that can arise during operation and significantly impact execution time and so, pWCET*. These are called *events of interest* (*eoi*). Hence, *for a correct application of MBPTA it is critically important to capture in the analysis-time measurements those events that can increase execution time meaningfully for a reliable application of MBPTA [1].*

Caches are one of the resources whose timing behaviour is hard to analyse [7], [8], [10], [19], [21], [22]. For the set of processor architectures considered so far in MBPTA [16] — which resembles that of the LEON3 [25] processor, caches have been shown to be the only resource that challenges the reliability of MBPTA [1]. In particular, a cache *eoi* is triggered when a number of program objects (i.e. code or data) larger than the associativity (W) are mapped to the same set [1]. This event may cause an abrupt increase in the number of misses with the corresponding increase in execution time [1], [23] with respect to the case where at most W addresses are mapped to the same set. If such an *eoi* can happen during operation with non-negligible (relevant) probability, *at least one* eoi *needs to be observed in the runs performed at analysis time, not to compromise MBPTA reliability*. Relevant probability depends on the domain specific safety standard and the application safety integrity level such that the residual risk cannot be deemed as sufficiently low, e.g. above 10^{-9} per hour of operation for DAL-A applications in avionics [24].

Illustrative example: let us assume a 4-way 32-set cache and a program comprising 8 single-line objects. In general if 5 or more of these objects are mapped to the same set, they will evict each other in cache causing an increase in the miss rate and execution time. This is in contrast to the case when only 4 or less objects are mapped to the same set, since they fit in that set. In a time randomised cache in which addresses (objects) are – in each run – randomly mapped to sets, the probability that 5 or more objects are mapped in the same cache set is $4.9 \cdot 10^{-5}$. Hence there is a low probability that in 1,000 experiments (a typical number of runs with MBPTA) this cache event of interest is captured. In particular this probability is 0.048. This is a problem for MBPTA since, without observing this event, MBPTA cannot predict its impact and how it can interact with other timing events. Hence, for a correct application of MBPTA we must ensure that we make enough analysis runs so that this event is captured at least in one of the runs.

Contribution. For set associative time-randomised caches (*TRc*), which have been already prototyped into FPGAs [11],



(a) Probability distribution function (PDF) (b) CDF and 1-CDF (logarithmic scale) (c) Example of pWCET curve

Fig. 1. Synthetic program's PDF, CDF, 1-CDF and pWCET curve. Probabilities of interest for MBPTA.

hence having a high Technology Readiness Level (TRL), this paper proposes a technique to compute in an exact manner the probability of the cache event of interest, called P_{eoi}^{TRc}. This is a fundamental step to gain confidence on pWCET estimates obtained with MBPTA. Given the set of objects to be allocated and their size – which is known at design time – our technique determines whether with the number of runs R carried out at analysis time, the cache event of interest will be captured with a sufficiently high probability. If this is not the case, our technique reports back the increased number of runs to carry out R' so that confidence is regained on the fact that the *eoi* will be observed. With focus on programs with homogeneously accessed objects, we make the following contributions:

1) We present an exact formulation of P_{eoi}^{TRc} in contrast to previous approximation formulas [1]. We identify the reasons for the inaccuracies of the previous technique proposed in [1] and we qualitatively and quantitatively compare it with our approach.
2) We present a methodology, which building on previous formulas, determines whether more runs are required to ensure that all relevant events of interest are captured in the analysis-time measurements.
3) We provide a solid evaluation based on synthetic benchmarks for sensitivity analysis and a real avionics application [27].

The rest of the paper is organised as follows. Section II states the problem addressed in this paper and introduces some basic concepts. Section III presents our approach to model the confidence of timing analysis for *TRc*. Section IV presents some experimental results. Section V describes some related work. Finally, Section VI summarises our main conclusions.

II. Problem Statement and Basic Concepts

MBPTA provides a pWCET distribution function that describes the residual risk in the form of an exceedance probability (e.g., 10^{-15} per run) at which evidence cannot be retrieved on whether one instance of a program cannot exceed the corresponding execution time bound. For instance, Figure 1(a) represents for several runs of a synthetic program on a MBPTA-compliant platform [4], the probability distribution function (PDF) and Figure 1(b) – in logarithmic scale – the cumulative distribution function (CDF) and the complementary CDF (1-CDF). When collecting a sample with R observations (execution time measurements), one could estimate the pWCET at an exceedance probability of $1/R$ at most. Since much smaller probabilities (so lower residual risk) are needed in the context of safety-relevant systems, EVT is used to estimate the function that describes the rightmost tail of the execution time distribution. For our example, Figure 1(c) shows the result of collecting $R = 1,000$ measurements and applying EVT to estimate the pWCET distribution. The dashed line corresponds to the 1-CDF for the 1,000 measurements collected in the test runs performed *at analysis time*, whereas the solid line corresponds to the pWCET distribution estimates with EVT *that must hold during operation*.

The pWCET estimates obtained with MBPTA stay valid under the *execution conditions* considered at analysis time once the system is in operation. Those execution conditions include all events that may impact the execution time of the program under analysis (e.g., memory layout, arbitration in shared resources). However, the conditions experienced at analysis time differ from those during operation simply because the latter may be unknown. In particular MBPTA imposes several requirements beyond those of EVT [4], [5] which, as used in MBPTA [5], requires a data sample of a random variable so that each execution time observation is independent and identically distributed. Additionally, MBPTA defines *representativeness* as the requirement in which the impact of any *relevant event* affecting execution time is properly upper-bounded at analysis time, where a relevant event corresponds to any event occurring with a probability above a cutoff threshold (e.g. 10^{-9} per hour of operation). We relate such threshold to the assurance/integrity level of the task and the probability of hardware random failures allowed under such assurance/integrity level as dictated by the corresponding functional safety standards in the domain. In particular, in the context of MBPTA the cutoff threshold upper bounds the residual risk under which evidence of reliable operation is not had as detailed later in Section II-B. While those events occurring with overly low probability become irrelevant for pWCET estimation purposes, events occurring with higher probability need to be accounted for, and this requires that their effect is captured in the measurements taken at analysis time. This occurs because, while EVT predicts the combined impact and the probability of observed events, EVT cannot predict in general those events that are never observed and whose impact in execution time is larger than that of the observed ones [1].

MBPTA representativeness (Figure 2) on the events of interest (*eoi*) relates to two specific probabilities.

The *exceedance probability* (P_{exc}) defines the lowest relevant probability for events occurring during operation. Events with smaller probability than P_{exc} are considered not relevant. P_{exc} is a function of the safety standards in the application domain and the criticality (integrity) level of the program. For instance, for commercial airborne systems at the highest assur-

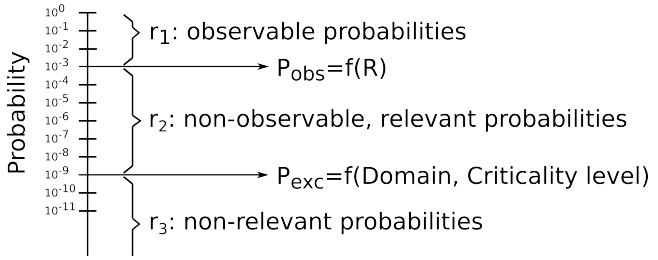


Fig. 2. Range of probabilities of interest for MBPTA.

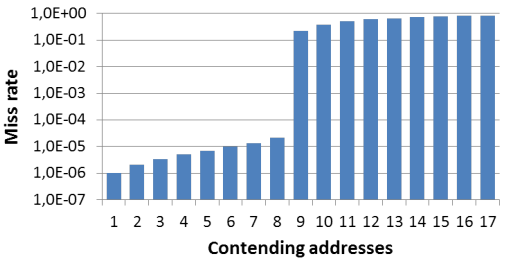


Fig. 3. Miss rates (in log scale) for different number of addresses accessed in a round-robin fashion competing for a 8-way cache set.

ance/integrity level (DAL-A), the maximum allowed random hardware failure rate in a system component is 10^{-9} per hour of operation [24]. Thus, we use the same threshold to upper bound the residual risk in the software verification process.

The *observable probability* (P_{obs}) determines the lowest probability of occurrence of an event such that the probability of not observing it in the execution time measurements collected at analysis time is below a cutoff probability, e.g. $P_{coff} = 10^{-9}$. P_{obs} is a function of the probability of occurrence per run of the event, P_{eoi}, and the number of runs R (observations) collected by MBPTA at analysis time:

$$P_{obs} = 1 - (1 - P_{eoi})^R \quad (1)$$

For instance, for a cutoff probability of $P_{coff} = 10^{-9}$ and $R = 1,000$ runs, events with $P_{eoi} \geq 0.021$ will not be observed with a probability below P_{coff}, that is, $10^{-9} \geq (1 - 0.021)^{1000}$. It also follows that the higher the number of runs, the lower the P_{eoi} that can be captured. Similar to P_{exc}, P_{coff} is a function of the applicable safety standard and criticality level. P_{exc} and P_{obs} define (Figure 2) three probability ranges:

1) *r1* is the probability interval for which a probabilistic argument can be provided on the fact that events with a probability in this range are captured in R runs, i.e. the probability of not observing them is irrelevant.
2) *r2* corresponds to the probability interval for which events may not be observed, yet they are considered relevant for the correctness (i.e. non-optimism) of the pWCET estimate.
3) *r3* corresponds to the probability interval below the exceedance threshold. Events occurring during operation with such a low (or smaller) probability are regarded as irrelevant in relation with the corresponding safety standard and criticality of the function.

This paper aims to determine P_{eoi}, taking different actions depending on its value: If $P_{eoi} \in r1$ or $P_{eoi} \in r3$ MBPTA is deemed as reliable as described above. However, when $P_{eoi} \in r2$ it is required to determine the increase in the number of runs (Δ_R) to carry out. Increasing the number of runs to $R' = R + \Delta_R$ increases P_{obs} to P'_{obs} such that events with lower probability can be observed, making that $P'_{eoi} \in r1$. Hence, our proposed approach is vital to maintain confidence on MBPTA-provided pWCET estimates.

A. Cache-related representativeness challenges

The Heart of Gold (HoG) approach [1] is the first attempt to address representativeness issues of cache related events for hardware time-randomised caches (*TRc*). HoG, whose representativeness findings for *TRc* were also identified in [20], [23], addresses the scenario in which the execution times, obtained from a MBPTA-compliant architecture deploying *TRc* [14] cause MBPTA to yield optimistic pWCET estimates. Authors in [1] note that the number of addresses competing for a set is the critical parameter affecting execution time noticeably. If competing addresses fit in the cache set – so there are up to W addresses where W is the cache associativity – then they will end up fitting in the cache set after some random evictions. Conversely, if there are more than W cache line addresses competing for the cache set space, then they do not fit and evictions will occur often, if all those cache lines are accessed often and in an interleaved fashion. This scenario where more than W cache line addresses compete for the space in a cache set is therefore the *cache event of interest*. We illustrate this scenario by performing an experiment where we access a number of addresses, between 1 and 17, in a loop iterating 1,000,000 times and accessing a cache set with $W = 8$, as seen in Figure 3. Miss rates are very low when the number of addresses does not exceed the space in the set. However, the miss rate increases abruptly (4 orders of magnitude) when 9 or more contending addresses are accessed.

Overall, for the sanity of the MBPTA results, it is crucial to determine whether execution times resulting from $W + 1$ addresses competing for the same cache set can occur with a sufficiently high probability to be relevant and, in that case, make sure they are included in the observations, which defines our event of interest for the cache. In HoG authors provided an approximate formula to derive P_{eoi} on *TRc*. However, as we show later, its inaccuracy can be significant in some cases, so we provide means to compute the exact value of P_{eoi}.

Let U be the number of addresses accessed by the program under analysis. In this paper we assume that the impact on execution time of mapping any arbitrary group of K addresses to the same set – with $K \in [W + 1, U]$ – is similar. This is the case, for instance, for the instruction addresses for many programs that may access them homogeneously inside a main control loop. In this case our method would require identifying those relevant addresses. In other cases where addresses are accessed heterogeneously (e.g., data accesses for control applications) a different solution would be required. Such a solution will likely require analysing program's access patterns. This is part of our current work.

B. Relating exceedance probabilities and safety standards

Functional safety standards such as DO178B/C [24] in avionics and ISO26262 [12] in automotive relate assurance (integrity) levels with failure rates, either absolute or per hour

of operation. However, software verification and testing is not explicitly related to those failure rates.

In any software verification process in the context of certification there is a qualitative step to collect "enough" evidence about software not failing during operation, where standards describe appropriate means to collect sufficient evidence for the different assurance (integrity) levels. In the context of MBPTA, pWCET estimates come along with an exceedance threshold. Such threshold upper bounds the risk of one instance of the task to overrun its assigned budget, i.e. suffering a timing violation (failure). The purpose of the exceedance threshold is not truly upper-bounding software failure rates which, in principle, are not allowed, but upper-bounding the residual risk of the software verification process.

For instance, with deterministic caches the placement of objects in memory determines which cache set each object is assigned to (e.g. based on modulo placement) resulting in a given cache layout. Conventional measurement-based practice on deterministic caches relies on the user ability to reduce the risk of not evaluating memory placements leading to bad cache layouts that produce high execution times, which can occur during operation. Such (residual) risk is only assessed qualitatively given that the user, despite making many tests, does not have a way to determine whether the space of potential memory mappings (and the corresponding cache layouts) is truly covered. This occurs because for complex software it is hard to force a particular placement in a test run.

In the context of MBPTA and time randomised caches, the space of potential cache layouts and their impact is randomly explored: in each run, a random cache layout (mapping of objects to sets) is explored. In this way, the risk brought by unexplored placements is no longer to be controlled by the end user but it is transferred to the confidence had on the pWCET estimate obtained based on a given number of runs. As we present in this paper, it is possible to assess the probability of a particular mapping not to be observed.

Thus, while end users need to argue qualitatively on the non-existence of unobserved placements when using non-probabilistic measurement-based methods, MBPTA allows to argue *quantitatively* on the fact that evidence shows that pWCET estimates are not exceeded with extremely high probabilities, and the residual risk (a.k.a. exceedance threshold), which can be made arbitrarily low, indicates that beyond that probability (e.g., 10^{-15} per program run) evidence is not had and so there is some residual risk of failure.

Interestingly, the certification process is the same as for conventional (non-probabilistic) practice, but replacing user's ability and unquantified residual risk by a systematic and sound approach and a quantitatively upper-bounded residual risk.

III. Timing analysis of TRc

TRc [14], which have been prototyped into LEON3 [25] designs in FPGA [11], combine the address being accessed with a random number (RII) to compute the (random) set where the address is placed. RII is generated by a pseudo-random number generator [3] that provides sequences with long periods to prevent any correlations among random events. RII holds constant during the program execution so that an address is placed in the same set during the whole execution, but it is randomly changed across executions so that the particular set where an address is placed is also random and independent from the placement for the other addresses across executions. As a result, with *TRc* the probability of assigning a given object to any set is independent of how the previous objects were allocated. Further, each object actually has the same probability to be assigned to a given set, see Eq. 2.

TABLE I. Basic notation.

$\mathcal{O}$	Sequence of objects to allocate
R	Number of runs carried out by MBPTA at analysis time
S, W	Number of sets and ways (respectively) in cache
a_i	Cache allocation scenario i
a_i^{max}	Maximum allocation of any set in a_i

$$Pset_i^{TRc} = \frac{1}{S} \tag{2}$$

Next, we analyse the current approach [1] to *approximate* P_{eoi}^{TRc} showing why it does not provide the exact value. Then we introduce an approach based on multinomial coefficient to define an exact formula to P_{eoi}^{TRc}. To that end we use the notation in Table I and also build on the following definitions.

It is noted that *TRc* assign different addresses to different sets regardless of whether those addresses belong to the same (software) object. As a result, the allocation of a multi-line object of size l lines is equivalent to the allocation of l objects of size 1 line. That is, the allocation of n objects $i \in [0, n-1]$ with sizes $\{l_i\}$ is equivalent to the allocation of $\sum_{i=0}^{n-1}\{l_i\}$ objects of size 1.

Definition 1 (Allocation Scenario): An allocation scenario defines how allocated objects are mapped to sets. We denote allocation scenarios as a_i, with $a_i = (a_i^1, a_i^2, ..., a_i^S)$. a_i^j is the number of objects allocated to set s_j under a_i.

Definition 2 (Cardinality of an allocation scenario): The cardinality of an allocation scenario $|a_i|$ is given by the number of objects allocated under it.

Definition 3 (Maximum of an allocation scenario): The maximum of an allocation scenario a_i^{max}, is given by the maximum number of objects allocated to any set in that allocation scenario.

Definition 4 (Cache event of interest): The cache event of interest is defined by those scenarios, called scenarios of interest, that have a set where the number of allocated objects is higher than W, i.e. $a_i | a_i^{max} > W$.

For instance, for a 3-set 2-way cache (S=3, W=2) and a sequence with 3 single-line objects ($|\mathcal{O}| = 3$), the allocation scenarios are $\mathcal{A} = \{(0,0,3), (0,1,2), (0,2,1), (0,3,0), (1,0,2), (1,1,1), (1,2,0), (2,0,1), (2,1,0), (3,0,0)\}$. From those, the scenarios of interest are $(3,0,0), (0,3,0)$ and $(0,0,3)$.

Definition 5 (Probability of the event of interest): The probability of the event of interest (P_{eoi}) is given by the addition of the probabilities for the allocation of scenarios of interest.

A. Weak compositions based approach

Determining whether at least $W+1$ objects out of the $|\mathcal{O}|$ under consideration are mapped into the same cache set

requires deriving all potential mappings of the $|\mathcal{O}|$ objects into the S cache sets and the fraction of those mappings in which at least one cache set has $W+1$ objects allocated. As shown in [1] these values can be approximated by means of *weak compositions* theory. A weak composition of an integer n is a way of writing n as the sum of a sequence of non-negative integers [9]. We are interested in all the weak compositions of $|\mathcal{O}|$ made of exactly S parts where at least one part is higher than W and the total number of weak compositions of $|\mathcal{O}|$ is exactly S sets. When no limit is put on the values of the parts we have $WComp(|\mathcal{O}|, S, -)$. If we impose that no part can have more than W objects we have $WComp(|\mathcal{O}|, S, \leq W)$. The probability of the mappings of all objects such that one part is greater than W can be approximated as:

$$\widehat{P^{TRc}_{eoi-wc}}(|\mathcal{O}|, S, W) = 1 - \frac{WComp(|\mathcal{O}|, S, \leq W)}{WComp(|\mathcal{O}|, S, -)} \quad (3)$$

The problem of this approach is that it considers that all potential allocation scenarios have the same probability. However, in reality two scenarios can have different probabilities. For instance, Figure 4 shows all possible allocation scenarios (10) resulting from allocating 3 objects in a 3-set cache. Edges represent how scenarios are allocated while nodes show each allocation scenario. The nodes in the same level have the same number of allocated objects. In this example, the event of interest for a 2-way cache is computed with weak compositions as $\widehat{P^{TRc}_{eoi-wc}}(3,3,2) = 1 - 7/10 = 3/10$, while in reality it is $3/27$, as we present later in this section.

B. Multinomial coefficient based approach

In explaining our proposed approach based on multinomial coefficient we further make the following definition.

Definition 6 (Path of allocation scenarios leading to a_i): Given an allocation scenario a_i, each of the successions of allocations in the probability tree leading to a_i is called *path to* a_i. Each path leading to a_i is represented as $pth(a_i) \in PTH(a_i)$, where $PTH(a_i)$ is the set of all paths leading to a_i.

For instance, in the example in Figure 4 the path leading to $a = (2,0,0)$ is $PTH(2,0,0) = \{(0,0,0),(1,0,0),(2,0,0)\}$

With *TRc*, all the paths reaching any of the allocation scenarios with the same cardinality have the same probability. This occurs because for every new allocated object the probability of being mapped to any set does not vary with the allocation of previous objects. After all objects in $\mathcal{O}$ have been allocated (represented by the leaves in Figure 4), the cardinality of any of the potential resulting allocation scenarios a_i equals to the number of allocated objects, i.e. $|a_i| = |\mathcal{O}|$, and the probability of reaching any of them through one path is:

$$P^{TRc}_{pth(|a_i|)} = \left(\frac{1}{S}\right)^{|a_i|} \quad \forall a_i \quad (4)$$

It is worth noting that not all allocation scenarios with the same cardinality are reached the same number of times, i.e. the number of paths leading to each scenario varies. For instance, in the example in Figure 4, from all those scenarios with cardinality three, $PTH(a_i)$ for $a_i = (3,0,0)$ comprises 1 path, while $PTH(a_j)$ for $a_j = (1,1,1)$ comprises 6 paths.

We approach the problem of computing the number of times a specific allocation scenario can be reached – which determines the number of paths leading to it – using the combinatorial interpretation of the multinomial coefficient [26]. The multinomial coefficient defines the possible combinations of distributing n elements over k containers, each containing exactly $y = (y^1, y^2, ..., y^k)$ elements. By replacing n by $|\mathcal{O}|$, k by S and y^j by a_i^j, the multinomial coefficient provides the number of times a_i can be reached, i.e. the number of paths from the original empty allocation scenario leading to a_i, which we call Np_{ai}. Np_{ai} is computed as follows:

$$Np_{ai} = \begin{pmatrix} n \\ a_i \end{pmatrix} = \frac{n!}{a_i^1! a_i^2! \cdots a_i^S!} \quad (5)$$

For instance, in the example in Figure 4 (3-sets cache and 3 objects) the number of times that the possible combinations give the outcome $a_i = (1,1,1)$ is given by $Np_{a_i} = \binom{3}{1,1,1} = 6$.

We derive the probability of a specific allocation scenario by multiplying the number of paths leading to it by the probability of a single path:

$$P^{TRc}_{a_i} = Np_{ai} \cdot P^{TRc}_{pth(|a_i|)} = \begin{pmatrix} |a_i| \\ a_i \end{pmatrix} \cdot \left(\frac{1}{S}\right)^{|a_i|} \quad (6)$$

In the previous example, $P^{TRc}_{a_i}$ for $a_i = (1,1,1)$ is $6 \cdot \left(\frac{1}{3}\right)^3 = \frac{6}{27}$.

Overall, P^{TRc}_{eoi} is computed by adding $P^{TRc}_{a_i}$ for those allocation scenarios of interest $a_i | max(a_i) > W$.

$$P^{TRc}_{eoi}(|\mathcal{O}|, S, W) = \sum_{\forall a_i | max(a_i) > W} P^{TRc}_{a_i} \quad (7)$$

For instance in the example in Figure 4 (3-sets cache and 3 objects) and assuming that the cache has $W = 2$ ways, the allocation scenarios where the event of interest occurs are: $a_i = (3,0,0)$, $a_j = (0,3,0)$ and $a_k = (0,0,3)$. Adding the probabilities of each one of this scenarios gives the total probability $P^{TRc}_{eoi} = \frac{1}{27} + \frac{1}{27} + \frac{1}{27} = \frac{3}{27}$.

The advantage of the *multinomial coefficient* approach is that it can exactly compute P^{TRc}_{eoi} while the *weak compositions* approach only approximates it. Yet, its computational cost is non-negligible. In particular, while the probability of a specific scenario of interest is fast to compute, the enumeration of all the scenarios of interest may take long (still less than half an hour per program in our experiments). Reducing the timing needs of our approach is part of our future work. As explained in following sections, the number of runs R required to gain sufficient confidence on MBPTA depends on P^{TRc}_{eoi}. Thus, failing to compute P^{TRc}_{eoi} exactly may lead to not collecting enough runs and hence, deriving optimistic pWCET estimates.

C. Increasing the number of runs

Given a sequence of objects $\mathcal{O}$ and a number of runs carried out at analysis time R, the probability of the event of interest P^{TRc}_{eoi}, as computed in Equation 7 might be below P_{obs} and above P_{exc}, shown in Figure 1(a) as range *r2*, which challenges MBPTA reliability. With R runs done by default

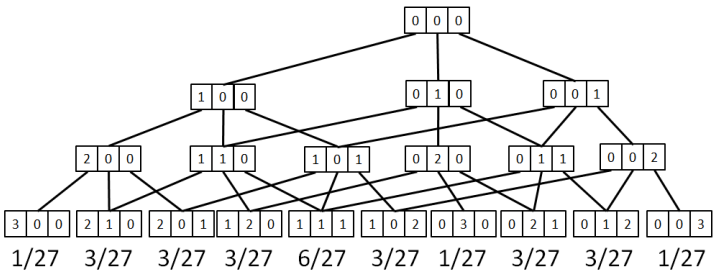


Fig. 4. Probability tree when allocating 3 objects in a 3-set cache with *TRc*. Leaf nodes represent the different allocation scenarios with cardinality 3.

by MBPTA, the lowest probability of an event such that the probability of not observing it in the R runs is below a given cutoff probability P_{coff}, is given by: $(1 - P_{eoi}^{TRc})^R \leq P_{coff}$. If we work out R we obtain $R \leq log_{(1-Peoi^{TRc})} P_{coff}$ hence $R \leq \frac{log(P_{coff})}{log(1-P_{eoi}^{TRc})}$. The minimum number of runs R' is:

$$R' = \frac{log(P_{coff})}{log(1 - P_{eoi}^{TRc})} \tag{8}$$

With R' it can be guaranteed that for the specified level of confidence P_{coff} the event of interest, whose probability is P_{eoi}^{TRc}, will be observed at analysis time.

IV. Experimental Results

In this section we assess the different approaches proposed to compute P_{eoi}^{TRc} using several object sequences. We carry out our experiments focusing on two cache setups, one small and one big, which we respectively call *scache* and *bcache*. The *scache* setup considers a 2KB size, 4-way, 8-set cache; while the *bcache* setup considers a 8KB, 4-way, 32-set cache. The *scache*, despite having a small size, has been considered to allow the comparison with a brute force approach (*probability tree generation*) as described later.

We consider randomly-generated sequences of objects of two types regarding their size. Small objects whose size is in the range $[8B, 16B, 32B, 64B, 128B]$ and big objects whose size ranges $[256B, 1KB, 2KB, 8KB]$. By mixing objects of these two types we generate 3 types of object sequences. The *smallSeq* comprises 100% of small objects, the *balSeq* comprises 75% of small objects and 25% big objects and the *bigSeq* comprises 50% of each group of objects. All object sequences have a size of $|\mathcal{O}| = 100$.

A. Accuracy Results

This section compares the results obtained for P_{eoi-mc}^{TRc} with our approach against weak compositions [1], i.e. $\widehat{P_{eoi-wc}^{TRc}}$. As a reference we use a method that is exact to the desired level of precision, but that has high memory and execution time requirements, which grow exponentially with the number of objects to allocate and the number of sets in cache. For this reason, in this section, we use the *scache* setup. This method, called *probability tree generation*, builds the probability tree as presented in Figure 4. Before the allocation of any object, a number of possible allocation scenarios exists (when the first object is to be allocated this number is one, with no object allocated). Each of those scenarios is expanded by allocating the new object in every possible set. The probability of each new generated scenario is the probability of the scenario from which it expands times 1/S (see Eq. 2). After the expansion, repeated scenarios are joined by adding their probabilities.

Figure 5(a) shows the result of the comparison. In particular we show how the different P_{eoi}^{TRc} estimates increase as more objects are allocated. First, we observe that the higher the number of allocated objects the higher the probability of the *eoi*, that is, the higher the probability that more than W addresses are mapped to the same set. Interestingly in this example for sequences smaller than 5 objects the probability of the cache *eoi* is below 10^{-9} that we use as the reference exceedance probability, P_{exc}. When 10 objects are allocated the P_{eoi}^{TRc} is above P_{obs} (this is further discussed in the next section). Note that we show P_{obs} for $R = 300$ and $R = 1,000$, two typical number of runs used by MBPTA.

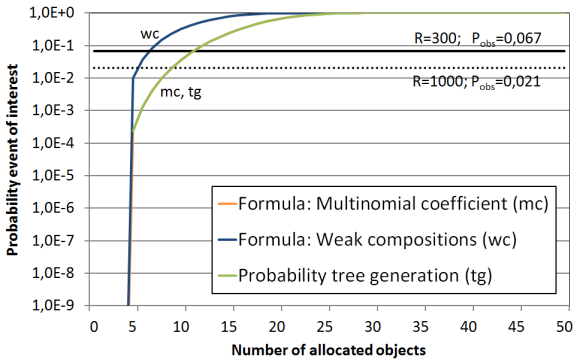
Our multinomial coefficient method provides exactly the same result (P_{eoi-mc}^{TRc}) as the probability tree generation (tg) method, while we observe deviations in the weak compositions (*wc*) method presented in [1]. With *wc* the probability of the *eoi* is higher than the exact one obtained with *mc* for a range of object counts. This can lead to pessimistic/optimistic results in terms of the number of runs to carry out:

1) $P_{eoi-mc}^{TRc} \in r3$ and $\widehat{P_{eoi-wc}^{TRc}} \in r3$: In this case both approaches, *mc* and *wc*, conclude that the probability of the *eoi* is irrelevant.
2) $P_{eoi-mc}^{TRc} \in r3$ and $\widehat{P_{eoi-wc}^{TRc}} \in r2$: In reality the probability of the *eoi* is irrelevant and with *wc* the user may be required to increase the number of runs.
3) $P_{eoi-mc}^{TRc} \in r2$ and $\widehat{P_{eoi-wc}^{TRc}} \in r2$: The user is asked to carry out, as a result of applying *wc*, fewer experiments than required to ensure that the cache *eoi* is captured, since $\widehat{P_{eoi-wc}^{TRc}} > P_{eoi-mc}^{TRc}$.
4) $P_{eoi-mc}^{TRc} \in r2$ and $\widehat{P_{eoi-wc}^{TRc}} \in r1$: As in the previous case.
5) $P_{eoi-mc}^{TRc} \in r1$ and $\widehat{P_{eoi-wc}^{TRc}} \in r1$: Both approaches, *mc* and *wc*, indicate that the probability of the *eoi* is captured with the runs carried out by the user.

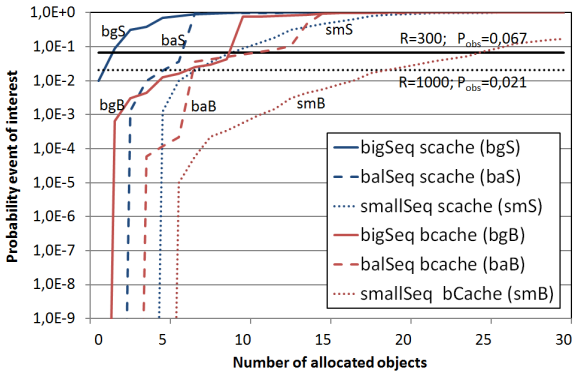
Under cases 3) and 4) MBPTA may lead to optimistic pWCET estimates if it is applied with the weak compositions approach since the user is requested to perform fewer runs than needed due to an overestimated P_{eoi}^{TRc}.

B. Results with object Sequences

In Figure 5(b) we use the different object mixes previously described and show P_{eoi}^{TRc} (obtained with our multinomial coefficient method) in each case as we increase the number of objects allocated. For the *scache*, with balSeq (baS), when allocating 1 to 3 objects the P_{eoi}^{TRc} is in range $r3$, not shown



(a) P_{eoi}^{TRc} vs $\widehat{P_{eoi-wc}^{TRc}}$ [1].



(b) Results for different sequences.

Fig. 5. Experimental results for synthetic object sequences.

in the figure because it is below $P_{exc} = 10^{-9}$. If the number of objects allocated is within 4 and 6, P_{eoi}^{TRc} is in the range $r2$, which can lead to a relevant event being missed in the analysis runs. When more than 7 objects are allocated, P_{eoi}^{TRc} is in $r1$ so it is observable. Similar trends are observed for smS, bgS, smB, baB, bgB, with the rule of thumb that the higher the allocated object count, the higher the cache occupancy, and hence the higher P_{eoi}^{TRc} is and the faster it converges to $r1$.

Number of runs. In Figure 5(b) we observe that with *scache*, for both smallSeq (smS) and balSeq (baS), when few objects are allocated (between 3 and 6) P_{eoi}^{TRc} is in $r2$. In this scenario, in order to ensure that P_{eoi}^{TRc} lies in $r1$, we use Equation 8 to increase R. For instance, if we allocate 6 objects in the smS we have that $P_{eoi}^{TRc} = 0.009833$ and $R' = 2,097$ runs would be needed to make sure that the event of interest is captured (i.e. lies in $r1$). Similarly, for baS, if we allocate 5 objects, $P_{eoi}^{TRc} = 0.019943$ and $R' = 1,028$ runs would be needed to guarantee that the P_{eoi}^{TRc} is located in $r1$.

C. Avionics case study

Next we apply our techniques to an industrial-size case-study [27] comprising around 5,000 functions ranging from few bytes to 300KB. The total size of those functions is 4.7MB if they are enforced to be aligned with cache line boundaries assuming a cache line size of 32B. In this experiment the focus is on code randomisation, i.e. on the instruction cache. With instruction *TRc* the cache set assigned to instructions is randomised across runs by hardware means. Figure 6 shows the probability of the *eoi* for the allocation of these objects in both

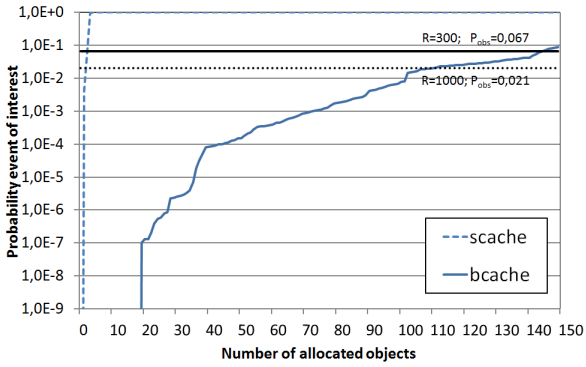


Fig. 6. Experimental results for the avionics case study.

the *scache* and the *bcache* setups. For the *scache* we can see that with as low as 4 objects allocated, the probability of the *eoi* is above P_{obs}. For the *bcache* setup, we need around 150 objects to reach P_{obs}. Hence, for this particular application, the number of runs carried out $R = 1,000$ with MBPTA standard process was enough to ensure representativeness of the instruction cache events of interest.

It is also worth noting that this real case requires more object allocations than the synthetic object mixes because the average function size is smaller than the smaller objects considered in our synthetic sequences. However, even with objects this small, the number of objects that must be allocated so that the event of interest is shown in the analysis runs is largely below the total number of objects in the application.

V. Related Work

Time-Randomised caches have been object of intense study in the last years. Their level of maturity has raised until they have been already prototyped in FPGA [11].

Several studies already compare the performance of measurement-based timing analysis on top of TRc and static timing analysis [2], [29] on top of time-deterministic caches. In terms of WCET, results show that MBPTA provides competitive results with respect to static timing analysis [16]. MBPTA also presents the advantage of being a measurement-based approach that can be faster adapted to new processors (systems) [29]. In terms of average performance, TRc have slight worse behaviour than deterministic caches, 12% on average [16]. Although this is not the focus of this paper it is worth mentioning several works on static probabilistic timing analysis (SPTA) for TRc [13], [14]. In general, SPTA is in a much more immature state than MBPTA that has been evaluated with avionics and automotive case studies [15], [27], [28]. Further, an important difference with respect to SPTA is that, while SPTA requires deriving or upper-bounding the hit/miss probability of every cache access, MBPTA only requires that the probability exists.

A set of techniques in the literature shows how MBPTA handles control flow dependences and data dependences [16]. For control flow dependences, the techniques in [17], [31] show how MBPTA can be adapted to provide a pWCET estimate that upper bounds the execution time of all the execution paths of the program, even when the user-provided input vectors only exercise a subset of the paths.

Although EVT can be used with time-deterministic architectures [30], there is no guarantee that EVT captures

the representative events that can occur at operation. This is possible with MBPTA [4] and is the object of this work.

Several academic works have identified the issue of representativeness of the event of interest [1], [20], [23]. Although some authors indicated that representativeness can be a risk [23], solutions have been provided later to mitigate the risk [1], [20] for *TRc*. For instance, HoG [1] – explained in Section II – is the first approach to solve this representativeness issue in *TRc*. However, HoG relies on an approximate method to obtain P_{eoi}^{TRc} and so the number of runs needed, thus not removing the risk completely. Our approach solves this issue by providing an exact method to obtain P_{eoi}^{TRc} and so the minimum number of runs to use MBPTA.

VI. Conclusions

The use of Extreme Value Theory in MBPTA is challenged by the fact that the execution time observations used for the prediction are those obtained at analysis time, while the predicted pWCET estimate must provide a reliable upper bound during operation. Therefore, evidence is required proving that execution time observations obtained at analysis time capture the impact of relevant events affecting execution time and that can arise during operation. Given the objects to be allocated for an application, we have proposed an exact method to compute the probability of cache related events of interest for time-randomised caches for homogeneously accessed objects. Our method identifies whether the probability of those events is high enough to be relevant and low enough so that high confidence on observing the event cannot be had. In that case confidence on MBPTA is regained by increasing the number of runs as indicated by our method.

Acknowledgments

The research leading to these results has received funding from the European Community's Seventh Framework Programme [FP7/2007-2013] under the PROXIMA Project (www.proxima-project.eu), grant agreement no 611085. This work was also supported by the Spanish Ministry of Science and Innovation under grant TIN2015-65316-P, the HiPEAC Network of Excellence. Jaume Abella was partially supported by the Ministry of Economy and Competitiveness under Ramon y Cajal postdoctoral fellowship (RYC-2013-14717).

References

[1] J. Abella et al. Heart of gold: Making the improbable happen to increase confidence in mbpta. In *Real-Time Systems (ECRTS), 2014 26th Euromicro Conference on*, pages 255–265, July 2014.

[2] J. Abella et al. Wcet analysis methods: Pitfalls and challenges on their trustworthiness. In *Industrial Embedded Systems (SIES), 2015 10th IEEE International Symposium on*, pages 1–10, June 2015.

[3] I. Agirre et al. IEC-61508 SIL 3 Compliant Pseudo-Random Number Generators for Probabilistic Timing Analysis. In *Digital System Design (DSD), 2015 Euromicro Conference on*, pages 677 – 684. IEEE, 2015.

[4] F.J. Cazorla et al. Upper-bounding program execution time with extreme value theory. In *13th International Workshop on Worst-Case Execution Time Analysis, WCET 2013, Paris, France*, pages 64–76, 2013.

[5] L. Cucu-Grosjean et al. Measurement-based probabilistic timing analysis for multi-path programs. In *Real-Time Systems (ECRTS), 2012 24th Euromicro Conference on*, pages 91–101, July 2012.

[6] W. Feller. *An introduction to Probability Theory and Its Applications*. 1996.

[7] C. Ferdinand and R. Wilhelm. Fast and Efficient Cache Behavior Prediction for Real-Time Systems. *Real-Time System*, XVII:131–181, 1999.

[8] C. Ferdinand et al. *Embedded Software: First International Workshop, EMSOFT 2001 Tahoe City, CA, USA, Proceedings*, chapter Reliable and Precise WCET Determination for a Real-Life Processor, pages 469–485. 2001.

[9] P. Flajolet and R. Sedgewick. *Analytic Combinatorics*. Cambridge University Press, 2009.

[10] D. Hardy and I. Puaut. WCET analysis of multi-level non-inclusive set-associative instruction caches. In *Proceedings of the 2008 Real-Time Systems Symposium*, RTSS '08, pages 456–466, 2008.

[11] C. Hernandez et al. Towards making a LEON3 multicore compatible with probabilistic timing analysis. In *20th Data Systems In Aerospace Conference (DASIA)*, 2015.

[12] International Organization for Standardization. *ISO/DIS 26262. Road Vehicles – Functional Safety*, 2009.

[13] L. Kosmidis, J. Abella, E. Quinones, and F. J. Cazorla. Multi-level unified caches for probabilistically analysable real-time systems. In *34th IEEE Real-Time Systems Symposium (RTSS)*, 2013.

[14] L. Kosmidis, J. Abella, E. Quinones, and F.J. Cazorla. A cache design for probabilistically analysable real-time systems. In *Design, Automation Test in Europe Conference Exhibition (DATE), 2013*, pages 513–518, March 2013.

[15] L. Kosmidis et al. Containing timing-related certification cost in automotive systems deploying complex hardware. In *DAC*, 2014.

[16] L. Kosmidis et al. Probabilistic timing analysis and its impact on processor architecture. In *DSD*, 2014.

[17] L. Kosmidis et al. PUB: Path upper-bounding for measurement-based probabilistic timing analysis. In *ECRTS*, 2014.

[18] S. Kotz et al. *Extreme value distributions: theory and applications*. World Scientific, 2000.

[19] Benjamin Lesage, Damien Hardy, and Isabelle Puaut. WCET analysis of multi-level set-associative data caches. In *9th Intl. Workshop on Worst-Case Execution Time (WCET) Analysis, Dagstuhl*, 2009.

[20] E. Mezzetti et al. Randomized caches can be pretty useful to hard real-time systems. *Leibniz Transactions on Embedded Systems (LITES)*, 2(1), 2015.

[21] Frank Mueller. Timing analysis for instruction caches. *Real-Time Syst.*, 18(2/3):217–247, May 2000.

[22] Frank Mueller, David B. Whalley, and Marion Harmon. Predicting instruction cache behavior. In *In ACM SIGPLAN Workshop on Language, Compiler, and Tool Support for Real-Time Systems*, 1993.

[23] J. Reineke. Randomized caches considered harmful in hard real-time systems. *LITES*, 1(1), 2014.

[24] RTCA and EUROCAE. *DO-178B / ED-12B, Software Considerations in Airborne Systems and Equipment Certification*, 1992.

[25] http://www.gaisler.com/cms/index.php?option=com_content&task=view&id=13&Itemid=53. *Leon3 Processor*. Areroflex Gaisler.

[26] C. G. Wagner. *Basic Combinatronics*. CreateSpace Independent Publishing Platform, 2014.

[27] F. Wartel et al. Measurement-based probabilistic timing analysis: Lessons from an integrated-modular avionics case study. In *Industrial Embedded Systems (SIES), 2013 8th IEEE International Symposium on*, pages 241–248, June 2013.

[28] F. Wartel et al. Timing analysis of an avionics case study on complex hardware/software platforms. In *Proceedings of the 2015 Design, Automation & Test in Europe Conference & Exhibition*, DATE '15, pages 397–402, San Jose, CA, USA, 2015. EDA Consortium.

[29] R. Wilhelm et al. The worst-case execution time problem: overview of methods and survey of tools. *ACM TECS*, 7(3):1–53, 2008.

[30] Lu Yue, Iain Bate, Thomas Nolte, and Liliana Cucu-Grosjean. A new way about using statistical analysis of worst-case execution times. September 2011.

[31] M. Ziccardi, E. Mezzetti, T. Vardanega, J. Abella, and F. J. Cazorla. EPC: Extended Path Coverage for measurement-based probabilistic timing analysis. In *RTSS*, 2015.