Execution-Time Profiles

Stefan M. Petters

National ICT Australia *
Sydney NSW 2052
Australia
smp@nicta.com.au

Abstract

Probabilistic methods of analysis have gained increased popularity in the real-time research community. This covers schedulability analysis as well as WCET analysis. While the concept of execution-time profiles as unit of computation in theses analysis is not new, there is so far a lack of a reference definition and summary of the interpretation of, representation of and operations performed on execution-time profiles. This technical report aims to collate and summarise these in a single document.

1 Introduction

Establishing the worst-case system response time for all system actions is an integral part of the real-time analysis of a system. We define such a system action as the operations performed by the system in response to a triggering event, e.g. a time event or an event in the system environment and its response time is described by the time interval between the triggering event and the system response.

Various methods of doing this response-time analysis (e.g. [1]) have been proposed and in most cases they require the worst-case execution time (WCET) of each part of all actions, for example, task or system call in isolation. The response-time analysis takes interference and interdependence into account to result in a worst-case response time for each system action.

*National ICT Australia is funded by the Australian Government's Department of Communications, Information Technology, and the Arts and the Australian Research Council through Backing Australia's Ability and the ICT Research Centre of Excellence programs.

For both the response-time analysis as well as for the WCET analysis probabilistic approaches have been proposed which describe the execution time of code or the response time of an action as a distribution with different probabilities rather than a single worst-case description. While the terminology used in this paper is mainly derived from measurement-based WCET analysis as a starting point, the discussion is limited neither to WCET analysis (e.g. [2]) nor to measurement-based analysis (e.g. [3]). Furthermore it is equally applicable to tree or path based methods.

Most of the content of this paper is not fundamentally new, but has been implicitly or explicitly used or introduced in other papers [4–6]. The aim of this paper is to collate those findings and discussions for future reference.

2 Execution-Time Profile Interpretation

Execution-time profiles (ETPs) have two different meanings depending on whether they represent units of software directly measured or obtained by static analysis or represent compound units whose ETP is a result of the combination of two or more other ETPs.

2.1 Measurement Unit Level

In this context an ETP describes the probability distribution of different execution times of the code measured. This is usually achieved by taking histograms of execution time observed as depicted in Figure 1 and dividing those by the overall number of measurements observed of the respective unit, effectively norming the

distribution to 1. The result as shown in Figure 2 is distinguished from Figure 1 only in scale and interpretation.

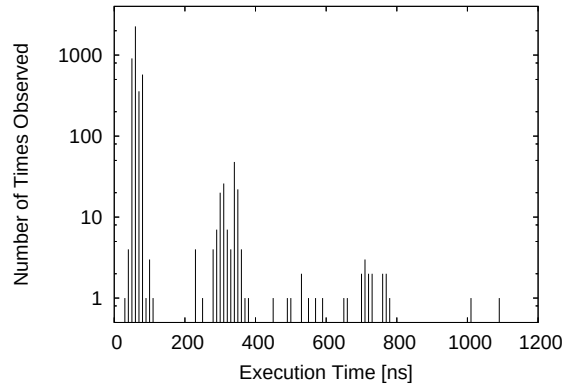

Figure 1: Execution Time Histogram

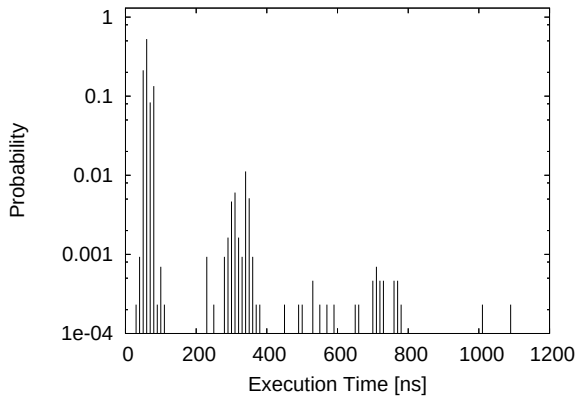

Figure 2: Execution Time Profile from Example in Figure 1

Usually measurements are taken with discrete values down to the granularity of processor core cycles. The discrete nature of the probabilities implies that these profiles are probability mass functions.

Initial work [6] used basic blocks represented by the nodes in a control-flow graph as measurement unit while later work [7] moved the weight of computation from the basic blocks to the transitions between basic blocks represented as the edges in a control-flow graph. The discussion of ETPs in this paper applies to both concepts. From now on we will use the term *code* with an identifier $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$, ...to reference either a block of code if applied to an approach like [6] or a transitions between two nodes of the control-flow graph [7].

2.2 Compound Level

While the notion of what ETPs represent at the measurement-unit level is fairly straightforward, it is less clear what they actually mean and represent for compound blocks, which are constructed by combining two or more measured or compound profiles. For this it is helpful to reflect on what the extreme value of such an ETP represents. The extreme value of an ETP at compund level is only an upper bound on the WCET, while the real value for the WCET is generally unknown. Just as for the WCETs, the probabilities of a compound block need to be interpreted as an upper bound on the *real* probability distribution of the code represented by the compound block.

This interpretation is based on two assumptios:

1. The measurements captured the WCET of the code at the measurement unit level. Within NICTA there is currently an effort which aims to provide proof of having observed the WCET on the measurement level [8].

2. All operations performed to obtain this ETP need to be conservative, meaning, no operation may make optimistic assumptions, which could potentially lead to a computed ETP to be lesser in any point than the *real* ETP. The comparison operators of *lesser than*, *greater than* etc. will be explored in greater detail in Section 4.1.

3 Representation of ETPs

The discussions in this section use the terminology and context of Section 2.1 for clarity of representation, but the defined operations are equally applicable to any ETPs describing a compound block of code.

Let $\mathbf{S}$ be the set of different execution times $ET_{\mathcal{A}}$ observed for the measured code $\mathcal{A}$ the probability mass function an ETP ($c_{\mathcal{A}}(t)$) is consistently with the example in Figure 2 defined as:

$$c_{\mathcal{A}}(t) = \begin{cases} P(ET_{\mathcal{A}} = t) & : \quad t \in \mathbf{S} \\ 0 & : \quad t \in \mathbb{R} \setminus \mathbf{S} \end{cases} \tag{1}$$

In order to be able to define comparison operators as required by Section 2.2, the cumulative distribution function provides a slightly more adequate representation.

$$C_{\mathcal{A}}^{cum}(t) = \sum_{s=0}^{t} c_{\mathcal{A}}(s) \quad (2)$$

Informally $C_{\mathcal{A}}^{cum}(t)$ describes the probability of an execution time of $\mathcal{A}$ being shorter than or equal to t. Trivially, the inverse function to obtain the probability mass function from the cumulative probability function is:

$$c_{\mathcal{A}}(t) = C_{\mathcal{A}}^{cum}(t) - C_{\mathcal{A}}^{cum}(t-1) \quad (3)$$

Graphically this representation of our example distribution introduced in Figure 1 is depicted in Figure 3. While the cumulative representation is reasonable for defining some operators, it is not suited for graphical analysis by inspection, as the most relevant portion of the ETP at the right hand tail is usually subject to small probabilities and thus barely intelligible as can be seen for execution time larger than 400ns in Figure 3.

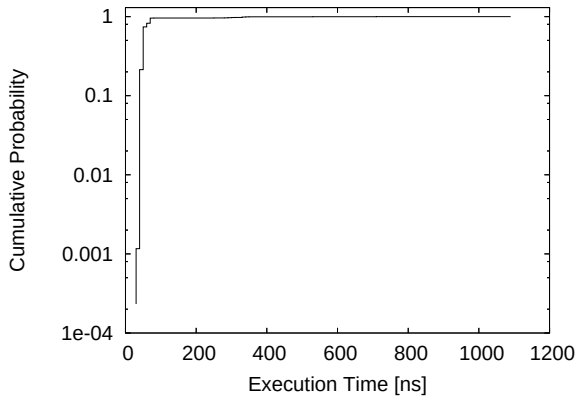

Figure 3: Cumulative Representation of ETP in Figure 2

The complementary cumulative probability (CCP) mass function $C_{\mathcal{A}}(t)$ combines the ease of mathematical expression with the good visibility of the most relevant part of the ETP which can be observed in the sample provided as Figure 4 enhanced by using a logarithmic scale on the y-axis. $C_{\mathcal{A}}(t)$ provides the probability of $\mathcal{A}$ exposing a execution time larger than t. The CCP representation has also the advantage of making the comparison and choice operators more intuitive which will become obvious in Sections 4.1 and 4.2.

$$C_{\mathcal{A}}(t) = 1 - C_{\mathcal{A}}^{cum}(t) \quad (4)$$

The direct conversion between CCP representation and probability mass function representation is defined as follows:

$$C_{\mathcal{A}}(t) = 1 - \sum_{s=0}^{t} c_{\mathcal{A}}(s) \quad (5)$$

or simpler as:

$$C_{\mathcal{A}}(t) = \sum_{s=t+1}^{\infty} c_{\mathcal{A}}(s) \quad (6)$$

The reverse transformation is defined as:

$$c_{\mathcal{A}}(t) = C_{\mathcal{A}}(t-1) - C_{\mathcal{A}}(t) \quad (7)$$

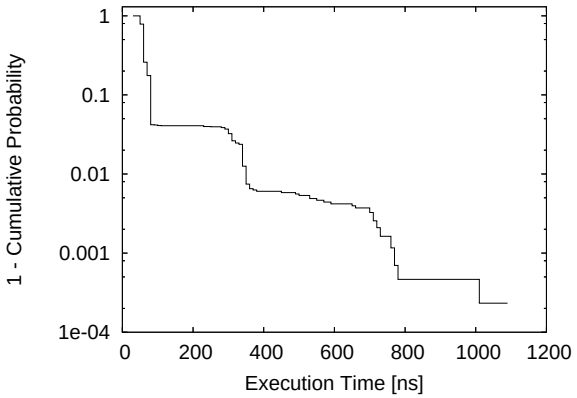

Figure 4: Complementary Cumulative Representation of ETP in Figure 2

It needs to be stressed that the three representations $C_{\mathcal{A}}^{cum}(t)$, $C_{\mathcal{A}}(t)$, and $c_{\mathcal{A}}(t)$ describe all the same unit $\mathcal{A}$ and have simple one-to-one and only one relationship in all their points. In the rest of the paper we make a slight distinction between $C_{\mathcal{A}}(t)$ which denotes the probability of time t being exceeded for code $\mathcal{A}$ and $C_{\mathcal{A}}$ which denotes the whole ETP.

4 Operators

This section aims to provide the mathematical foundation for operations performed on ETPs. Similar to the last section we are referring in this context to ETPs at the measurement unit as well as at the compound level. Figures 5 and 6 contain sample ETPs which will be used in the following discussion.

4.1 Comparisons

Before going into detail of the discussion about comparison operators it has to be stressed that the concepts presented here are not of standard nomenclature in probability theory, but are very specific to the domain of real-time analysis.

The trivial comparison operator is equality of two ETPs $\mathcal{A}$ and $\mathcal{B}$. The two ETPs are identical i.e.:

$$C_{\mathcal{A}} = C_{\mathcal{B}} \iff \forall t : C_{\mathcal{A}}(t) = C_{\mathcal{B}}(t) \tag{8}$$

The *greater or equal* and *lesser or equal* operators need some more detailed discussion. As a first step we need to informally define what these operators imply. If the ETP of block $\mathcal{A}$ is *greater or equal* to that of block $\mathcal{B}$ that means that for any given execution time t, the probability of an execution time of $\mathcal{A}$ being greater than t is larger or equal than the probability for $\mathcal{B}$.

$$C_{\mathcal{A}} \geq C_{\mathcal{B}} \iff \forall t : C_{\mathcal{A}}(t) \geq C_{\mathcal{B}}(t) \tag{9}$$

The same logic applies when looking to the *lesser or equal* operator.

$$C_{\mathcal{A}} \leq C_{\mathcal{B}} \iff \forall t : C_{\mathcal{A}}(t) \leq C_{\mathcal{B}}(t) \tag{10}$$

Since these operators are defined on CCP mass functions a strict *less than* or strict *greater than* would require exceptions for values $C_{\mathcal{A}} = 0$ and $C_{\mathcal{A}} = 1$. However, for the purpose of real-time analysis these strict comparison operators are not needed. The *Gaussian* and *Biased* ETPs in Figure 6 are less or equal compered to the *Supremal* ETP. However, such a ordering is not possible between the *Gaussian* and *Biased* ETPs i.e. both *Gaussian* ETP is in some points larger than the *biased* and vice versa.

4.2 Choice

Whenever choices need to be expressed the *max* and *min* operators are used.

$$C_{\mathcal{C}} = \max\{C_{\mathcal{A}}, C_{\mathcal{B}}\} \tag{11}$$

if and only if:

$$\forall t : C_{\mathcal{C}}(t) = \max\{C_{\mathcal{A}}(t), C_{\mathcal{B}}(t)\} \tag{12}$$

$$C_{\mathcal{C}} = \min\{C_{\mathcal{A}}, C_{\mathcal{B}}\} \tag{13}$$

if and only if:

$$\forall t : C_{\mathcal{C}}(t) = \min\{C_{\mathcal{A}}(t), C_{\mathcal{B}}(t)\} \tag{14}$$

These two operators are associative and commutative. The act of defining the $\min$ operator may come as a bit of a surprise, but simplifies some comparisons when deciding which of more than two alternatives is the worst case.

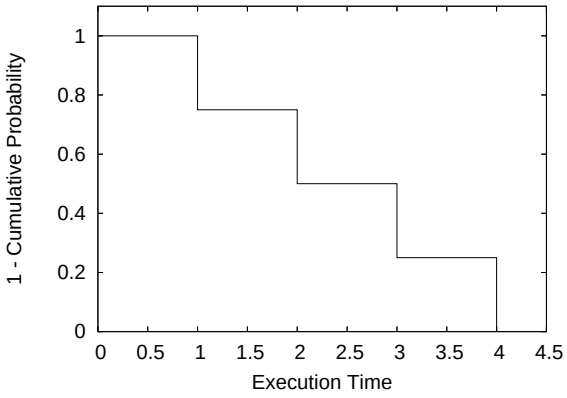

Figure 5: A Simple Sample ETP

4.3 Convolutions

Convolutions are used to express the combination of random variables in an additive way, where the resulting value is the value produced by the first random variable plus the value produced by the second random variable. While there exist many different convolutions representing different assumptions regarding the dependencies between the random variable, we will concentrate on the discussion of only a few, which we perceive as relevant in the context of real-time analysis. Figure 5 contains a very simple sample ETP which will be used to illustrate the effects of the different types of convolutions. Three central features of convolutions are associativity, commutativity, and distributivity with respect to the $\max$ and $\min$ operators are important as they imply that order of performing the convolutions is irrelevant and we can move convolutions into alternatives. This is highly desirable in a tree based approaches which validates the use of a tree and associated schema and allows tightening results by delaying decisions with regards to the choice operators. However, please note the exceptions with the known dependencies below.

Gaussian Convolution The most common convolution is the Gaussian convolution. However, the its use is only permissible if the two random variables combined are independent of each other. The Gaussian convolution is defined using the probability mass representation as:

$$c_{\mathcal{C}}(t) = \sum_{s} c_{\mathcal{A}}(s) * c_{\mathcal{B}}(t-s) \tag{15}$$

Figure 6 shows the Gaussian convolution of two identical ETPs depicted in Figure 5.

For obvious reasons the independence assumption for ETPs is not generally the case for subsequent code segments as, for example, instruction caches establish a certain dependence between subsequent code which is often reflected in dependecies between the ETPs of these code blocks. It can be argued that there is a certain level of independence between basic blocks which are far apart from each other. The assumption of independence may be used in some soft real-time systems, to obtain a more "realistic" distribution as such is given here, but for hard real-time systems this would not not good enough.

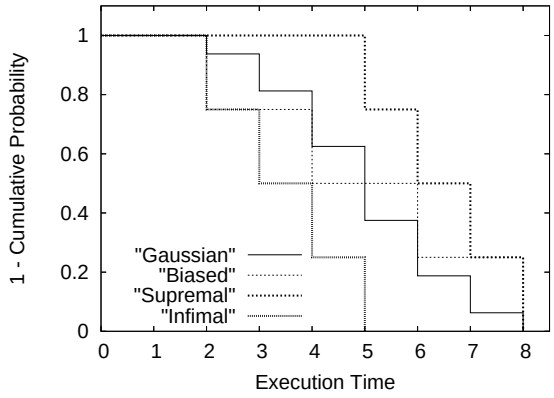

Figure 6: Different Convolutions

"Biased" Convolution A second option is to assume perfect positive correlation between two random variables branded by Bernat et al. [6] as *biased convolution*. However, this convolution leads to considerable overestimation in the probabilities near the WCET while lacking the ability to account for arbitrary dependencies. This is particularly evident in the center of the resulting distribution where it is easy to create a dependency structure between the two ETPs to produce an underestimation as the result of the convolution compared to the *real* value. Hence, the convolution is only provided for the sake of completeness, as we doubt it has any practical relevance. The biased convolution can be described by:

$$\forall t_1 \exists t_2 : C_{\mathcal{A}}(t_1) = C_{\mathcal{B}}(t_2) \rightarrow C_{\mathcal{C}}(t_1 + t_2) = C_{\mathcal{A}}(t_1) \tag{16}$$

Supremal Convolution The third option is the supremal convolution [9]. This options is the most pessimistic of the three convolutions presented here, but captures any possible dependency in a worst case manner. It is the dual operation to the infimal operation, which covers any possible dependency for a best case scenario. Any real dependency is bound by these two functions.

$$C_{\mathcal{C}}(t_r) = \sup_{t_1+t_2=t_r} \{\max\left(C_{\mathcal{A}}(t_1), C_{\mathcal{B}}(t_2)\right)\} \tag{17}$$

The infimal convolution is the dual of the supremal convolution and might be used for best-case analysis. It is defined as:

$$C_{\mathcal{C}}(t_r) = \inf_{t_1+t_2=t_r} \{\min\left(C_{\mathcal{A}}(t_1), C_{\mathcal{B}}(t_2)\right)\} \tag{18}$$

Figure 6 shows how the supremal and infimal convolutions on the sample ETP in Figure 5 envelope the profiles.

Known Dependencies So far we have either made simplistic assumptions about dependencies, or, in case of the supremal convolution no assumptions at all, but rather used a bounding function. However, when the dependency structure is known or partially known [8] we can use copulas to express this dependency structure. A copula is a mapping [9]:

$$[0,1] \times [0,1] \rightarrow [0,1] \tag{19}$$

Discounting all but the boundary cases of $C_{\mathcal{A}}(t) = 1$ and $C_{\mathcal{A}}(t) = 0$, $C_{\mathcal{A}}(t)$ is a one-to-one mapping $t \rightarrow [0,1]$ for a block $\mathcal{A}$ we can use a copula to relate the execution time of two ETPs $\mathcal{A}$ and $\mathcal{B}$. The supremal and infimal convolutions may easily be derived and proven using copulas [9]. Note that the operation using copulas is not commutative and only distributive with respect to

the $\max$ and $\min$ operator. Developing the exact formalism to introduce known dependencies is subject to future work.

A slightly different challenge is the expression of partial known dependencies. Schaefer et al. [8], for example, aim to establish the dependencies of first order effects between measurement-unit level blocks. However, second order effects are not covered and thus have to be subject to treatment with appropriate measures within the bounds set by the known dependencies. In this particular model, sections of the ETPs are associated with each other. This can be shown in an example where the interval $[t_1, t_2]$ of $C_{\mathcal{A}}$ is yielding an execution time in the interval $[t_3, t_4]$ of $C_{\mathcal{B}}$. We can perform supremal convolutions of the corresponding parts of the ETPs and superpose the results in the probability mass function representation.

However, the question arises with which probabilities in block $\mathcal{C}$ the result would be associated with. Motivation for this is the case in which the parts of the two ETPs associated with each other contain not necessarily the same probabilistic weight, i.e. $C_{\mathcal{A}}(t_2) - C_{\mathcal{A}}(t_1) \neq C_{\mathcal{B}}(t_4) - C_{\mathcal{B}}(t_3)$. This can be explained by, for example, $\mathcal{B}$ being executed in a path without executing $\mathcal{A}$. If a clear path dependency exists i.e., whenever $\mathcal{A}$ is executed $\mathcal{B}$ is also executed, the weight of the relevant section of $\mathcal{A}$ (i.e. $C_{\mathcal{A}}(t_2) - C_{\mathcal{A}}(t_1)$ in the example used above) can be used to weight the result of the convolution in the superposition.

In case the resulting partial profiles are non overlapping we can still reason about the weights for the superposition to bias the results towards the worst case, but the issue when there is neither the statement of path dependency nor non-overlapping partial profiles for superposition is so far unresolved and needs to be addressed.

5 Conclusion

Within this paper we have discussed execution-time profiles, their interpretation, representations and some operations which may be performed on them. The presented methods are not only applicable for WCET analysis but also for response-time analysis in which case the profile would be called response-time profile.

Open questions for future work are mainly in the area of dealing with known dependencies between execution-time profiles using copulas and dealing with partial dependencies in a effective, but nevertheless conservative manner.

References

[1] K. Tindell, "Holistic schedulability analysis for distributed hard real-time systems," technical report YCS187 (1993), University of York, Department of Computer Science, York, YO10 5DD, United Kingdom, 1993.

[2] A. Burns, G. Bernat, and I. Broster, "A probabilistic framework for schedulability analysis," in *Proceedings of the Third International Embedded Software Conference*, (Philadelphia, Pennsylvania, USA), pp. 1–15, Oct. 13-15 2003.

[3] L. David and I. Puaut, "Static determination of probabilistic execution times," in *Proceedings of the 16th Euromicro Conference on Real-Time Systems*, (Catania, Italy), July 1-3 2004.

[4] A. Colin and S. M. Petters, "Experimental evaluation of code properties for WCET analysis," in *Proceedings of the 24th IEEE International Real-Time Systems Symposium*, (Cancun, Mexico), Dec. 3–5 2003.

[5] G. Bernat, A. Colin, and S. M. Petters, "pWCET: a tool for probabilistic worst case execution time analysis of real–time systems," technical report YCS353 (2003), University of York, Department of Computer Science, York, YO10 5DD, United Kingdom, Apr. 2003.

[6] G. Bernat, A. Colin, and S. M. Petters, "WCET analysis of probabilistic hard real–time systems," in *Proceedings of the 24th IEEE Real-Time Systems Symposium*, (Austin, Texas, USA), pp. 279–288, Dec. 3–5 2002.

[7] S. M. Petters, A. Betts, and G. Bernat, "A new timing schema for WCET analysis," in *Proceedings of the 4th Workshop on Worst-Case Execution-Time Analysis*, (Catania, Italy), June 30 2004. Satellite Workshop of the 16th Euromicro Conference on Real-Time Systems.

[8] S. Schaefer, B. Scholz, S. M. Petters, and G. Heiser, "Static analysis support for measurement-based WCET analysis," in *12th IEEE International Conference on Embedded and Real-Time Computing Systems and Applications, Work-in-Progress Session*, (Sydney, Australia), Aug. 2006.

[9] R. C. Williamson, *Probabilistic Arithmetic*. PhD thesis, Department of Electrical Engineering, University of Queensland, Brisbane, Australia, Aug. 1989.