Proceedings Paper:
Burns, Alan and Baruah, S (2023) Multi-Model Specifications and their application to Classification Systems. In: 31st International Conference on Real-Time Networks and Systems, proceedings. 31st International Conference on Real-Time Networks and Systems, 06-08 Jun 2023 ACM, pp. 155-165.

https://doi.org/10.1145/3575757.3575760

Multi-Model Specifications and their application to Classification Systems

Alan Burns
The University of York
UK
alan.burns@york.ac.uk

Sanjoy Baruah
Washington University
USA
baruah@wustl.edu

ABSTRACT

Many safety-critical systems are required to have their correctness validated prior to deployment. Such validation is typically performed using *models* of the run-time behaviour that the system is expected to exhibit and experience during run-time. However, these systems may be subject to different requirements under different circumstances; also, there may be multiple stakeholders involved, each with a somewhat different perspective on correctness. We examine the use of a multi-model framework based on assumptions (Pre and Rely conditions) and obligations (Post and Guarantee conditions) to represent the workload and resource related needs of complex AI system components such as DNN classifiers. We identify three kinds of multi-models that are of particular interest: Independent, Integrated and Hierarchical. All the individual models comprising an *independent* multi-model must remain valid at all times during run-time; at least one of the models comprising an *integrated* multi-model must always be valid. With *hierarchical* multi-models all models are initially valid but the component's behaviour may gracefully degrade through a series of models with successively weaker assumptions and commitments (we show that Mixed-Criticality Systems, widely studied in the real-time computing community, are particularly well-suited for representation via hierarchical multi-models). We explain how this modelling framework is intended to be used, and present algorithms for determining the worst-case timing behaviour of systems that are specified using multi-models.

ACM Reference Format:
Alan Burns and Sanjoy Baruah. 2023. Multi-Model Specifications and their application to Classification Systems. In *The 31st International Conference on Real-Time Networks and Systems (RTNS 2023), June 7–8, 2023, Dortmund, Germany.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3575757.3575760

1 INTRODUCTION

The safety properties of many safety-critical systems must be verified before they may be deployed out in the field. Since such verification occurs prior to run-time, it is typically performed upon carefully-constructed *models* of the run-time behaviour that the system is expected to exhibit. Such models are designed to emphasize the salient features of interest from the perspective of verification.

The verification of timing correctness properties (e.g., that deadlines are met) is usually done by the application of results from *real-time scheduling theory*. The models used in real-time scheduling theory make assumptions regarding the form of the workload that will need to be accomodated and the characteristics of the platform upon which such executions will occur. The validity of the verification depends upon the actual workload and platform being compliant with these model assumptions. For instance, the widely used Liu & Layland task model [20] assumes that the real-time workload comprises an a priori known number of recurrent processes that are called *tasks*, each of which generates pieces of work ("*jobs*") a specified minimum duration (called the task *period*) apart, with each job needing to execute for no more than a specified duration of time (called the *worst-case execution time* or simply *WCET*); for such a workload executing upon a single fully preemptive processor, results in [20] a guarantee that any workload for which the sum of the ratios of the WCET-to-period parameters of all the tasks does not exceed $\ln 2$ (≈ 0.69) is scheduled by the *Rate-Monotonic* scheduling algorithm such that each job completes execution prior to the arrival of the next job of the same task. However, this guarantee need not hold if any of the assumptions are violated – if either the workload or the processing platform is not compliant with the model, or if the WCET-to-period ratios sum to more than the specified bound.

In this paper we model such workload and resource-usage specifications as a contract between *assumptions* ($\mathcal{A}$) and *obligations* ($\mathcal{O}$) (or *commitments*) [10, 17, 18, 23]: if the system behaves according to the assumptions then the obligations (including meeting deadlines) shall be delivered[1].

At runtime a system that has been verified according to the appropriate schedulability test may depend upon the validity of the assumptions regarding the characterisation of the work that must be performed and the resources required for this work. And if these assumptions hold then a verified implementation guarantees to meet its obligations. (Note that the system does not need to check during run-time that its assumptions are being met, although a more resilient/robust implementation may choose to do so.)

And if the assumptions do turn out to be invalid at some time during operation then the system is allowed to undertake *any* action, including shut-down (although again a more resilient or robust implementation may make an effort towards meeting its commitments at least partially, invalid assumptions notwithstanding).

In 2007, Vestal [29] proposed a generalization to the Liu & Layland task model [20], the distinctive feature of which is that the WCET parameter of each task is no longer a single value. Instead,

[1] Assumptions are often described [8] as a combination of Pre-conditions ($\mathcal{P}$) and Rely conditions ($\mathcal{R}$), while Obligations are a combination of Postconditions ($\mathcal{Q}$) and Guarantee conditions ($\mathcal{G}$).

RTNS 2023, June 7–8, 2023, Dortmund, Germany

ACM ISBN 978-1-4503-9983-8/23/06.
https://doi.org/10.1145/3575757.3575760

each task is characterized by multiple WCET parameter values representing different estimates, that may be trusted to different levels of assurance, of the actual (unknown) maximum duration for which each job of the task may actually execute. Each task is assigned a "*criticality"* level, informally denoting its importance to some stakeholder in the system. The correctness criterion is that all tasks at or above a particular criticality level commit to meet their deadlines assuming that the actual execution durations of all jobs do not exceed the WCET estimates made at the level of assurance corresponding to that criticality level. MCS's have been very widely studied in the real-time scheduling literature (see, e.g., [6] for a survey); we will see, in Section 2.1, that this Vestal model for MCS's is essentially what we are terming here a hierarchical multi-model.

For relatively simple components a single model, such as the Liu & Layland characterization [20] of each task by a single period parameter and a single WCET estimate, is adequate. In general, however, it is the case that the work that each task in a component has to undertake may vary according to ambient operating conditions (for example, the number of planes in a radar image, the number of faces in a recognition system, or the number of cars in a traffic control system), and as a consequence the expectations upon the system –the obligations that can reasonably be expected from it– may vary. It may also be the case that different stakeholders have somewhat different expectations of the system. We will show how both these cases may be modelled by specifying *multiple* assumption-obligation pairs for a single component. It is not always the case that the worst-case load on the system is when these parameters are at their maximum. What may maximise the load on one task may reduce the load on other tasks; these relations must be taken into account if overly pessimistic scheduling analysis is to be avoided.

The first contribution of this paper is therefore an extension of the properties of a mixed-criticality system to a more general notion of a multi-model specification. And rather than linking assumptions only to execution times (the resources needed), in this paper we allow them to also incorporate assumptions about the number of relevant entities in the input space (the work that has to be done). We believe that this framework is widely applicable to a range of systems, in particularly those that incorporate AI algorithms and other forms of Learning-Enabled components [21] such as classifiers.

The second contribution is to consider how the worst-case execution time of software components that are based on deep learning and related AI technologies can be computed. Such components are increasingly being deployed for classification problems in complex autonomous resource-constrained cyber-physical systems. Many of these systems are employed (or are being considered for employment) in safety-critical applications and require accurate predictions to be delivered in real time using limited computing resources (this is sometimes called "edge AI" where the efficient execution of machine intelligence algorithms on embedded edge devices is required [9, 31]).

A number of schemes have been produced that aim to determine the worst-case path through a sequence (or cascade) of classifiers. For example Razavi et al. [24] note "*Deep learning (DL) inference has become an essential building block in modern intelligent applications. Due to the high computational intensity of DL it is crucial to scale DL inference serving systems in response to fluctuating workloads to achieve resource efficiency.*" They provide a heuristic to reduce the typical execution time of an object recognition system that is made up of a set of different classifier (including face recognition, optical character recognition, and natural language understanding). In this paper we demonstrate that a relative straightforward approach (compared with more general forms of WCET analysis) based on Dynamic Programming can be used to derive worst-case execution times for systems of classifiers whose temporal behaviours are bounded by workload assumptions.

Having derived this modelling and analysis technique for classification systems we use it to illustrate the multi-model framework. The remainder of the paper is therefore organised as follows. In the next section we introduce the notion of a multi-model and define three different forms: independent, integrated and hierarchical. In Section 3 we then define a single-model specification scheme based on Assumptions and Obligation for a SIMO-based classification system, and illustrate how timing analysis can be performed upon systems that are specified in this manner. Section 4 then describes a Multi-Model classification system, building upon the modelling framework for a single classification system from Section 3. Conclusions are drawn, and directions for future work suggested, in Section 5.

In this initial paper on Multi-Models and their application to classification systems we will keep the discussion informal and focus more upon communicating insight and intuition rather than formally defining our approach and providing rigorous correctness proofs. In this spirit we introduce the salient aspects of our proposed approach via a number of examples.

2 MULTI-MODEL SYSTEMS

Here we consider systems having more than one model to specify their expected runtime behaviour. Such multi-models[2] are particularly relevant if (i) there are *different modes of operation* that give rise to different models; or (ii) there are *different stakeholders* that define different assumptions and obligations for the system.

We noted in the introduction that Mixed-Criticality Systems (MCS's) are a specific example of the Multi-Model approach. We therefore start with a review of MCS. Although the use of contracts (mappings from assumptions to obligations) are used extensively in component engineering, they have not been widely applied to the temporal properties of real-time systems. Notable exceptions are works by Benveniste et al. [3] and Stoimenov et al. [28].

2.1 Related Work: Mixed-Criticality Systems

Mixed-criticality systems (MCS), widely studied in the real-time scheduling literature, provide an illustrative example of the use of multi-models for representing complex components. As stated in Section 1, each task in the task model proposed by Vestal [29] is characterized by multiple WCET parameter values representing different estimates, that may be trusted to different levels of assurance, of the actual (unknown) maximum duration for which each job of the task may actually execute. Each task is also assigned a *criticality* level, which is, informally speaking, an indicator of the importance

[2]The term 'multi-model' is used in a number of different contexts, in particular with regard to multi-model databases; there are also similar notions such as compositional analysis – here we use the term to simply express that a single system is being specified using more than one workload/resource model.

of that task to overall system correctness. As stated in Section 1, the Vestal [29] notion of correct system behavior is this: *assuming* that the actual execution durations of all jobs of all tasks do not exceed the WCET estimates made at the level of assurance corresponding to a particular criticality level, the system *commits* to meet their deadlines (i.e., complete execution prior to the arrival of the next job of the same task) of all tasks with criticality level at or above that criticality level.

From an analysis standpoint the important property of the Vestal model is not the use of criticality but the fact that the task-set under inspection has *more than one model* [4]. Vestal suggests that different stakeholders would want to assign different values to one of the parameters (the WCET) characterising each task: in effect there is not one but a collection of models that are being applied to the task-set, each modelling the system from a somewhat different perspective. Since the 2007 publication of Vestal's paper [29] there have been over 500 papers produced that have extended and utilised this notion of MCS [6, 7]. However, there have also been a number of papers that have criticised the Vestal approach [13–16, 22]; much of this criticism is based on different views as to the meaning of "criticality." But we point out that the rich body of results that have appeared under the umbrella of MCS do not require or assign any particular meaning to the term "criticality;" what they utilise and exploit is the idea that there is more than one interpretation of the temporal properties (i.e. parameters) of the tasks under consideration. Testament to the usefulness of this multi-model extension is the volume of applicable results that have been generated in under 15 years.

Burns et al. have illustrated [5, 8, 19] how the run-time behaviour of a simple MCS may be specified by using Rely Conditions (Assumptions) and Guarantee Conditions (Obligations). In the Mixed-Criticality framework there is a "degraded" mode with weaker Rely and Guarantee conditions into which the system may transition. In this degraded mode only the higher-criticality jobs are guaranteed to meet their deadlines. *This is therefore an example of a hierarchical multi-model*. In the following section we will argue that this is one of three possible kinds of multi-model.

2.2 Types of Multi-Model

It is sometimes convenient to interpret assumption-obligation specifications in terms of *mappings*. Under such an interpretation, the assumptions specify the set of all behaviors of the environment for which the system is expected to behave correctly; the obligations specify the corresponding correct system behaviors. Then *correct system execution maps each assumed behavior of the environment to some correct system behavior* – see the top diagram of Figure 1. The middle diagram in this Figure depicts a MCS with a hierarchical relationship between the assumptions and obligations. The bottom diagram generalises this relationship; there are overlapping sets of assumptions leading to overlapping obligations. In both of these situations, correct behaviour of the system requires at least one of the set of assumptions to remain true.

As stated above, our objective is to develop efficient algorithms that satisfy multiple models – multiple assumption-obligation specifications. We consider such a multi-model framework to be very general, and applicable to modelling a variety of different situations,

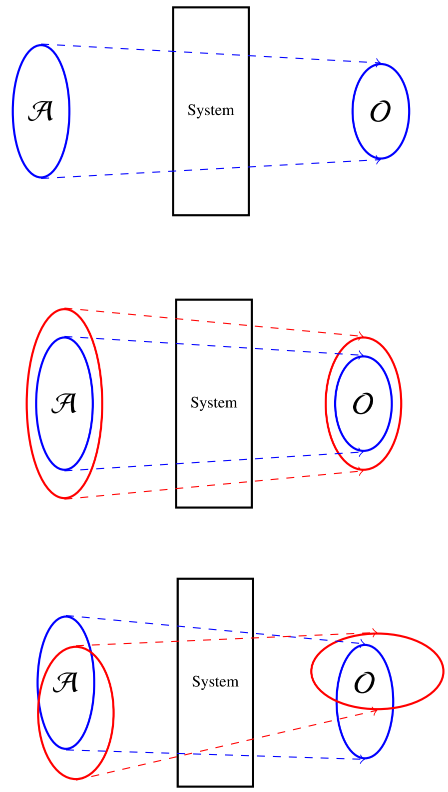

Figure 1: The top diagram depicts system execution as a mapping from a set $\mathcal{A}$ of assumed behaviors of its environment to a set $\mathcal{O}$ of system behaviors that fulfils its obligations. The middle diagram depicts a *mixed-criticality* system in which the sets of assumptions and obligations satisfy a subset/ superset relationship. And the bottom diagram depicts the execution of multi-model systems with overlapping integrated assumptions and obligations.

with the different models accorded different interpretations. For example:

Different Environmental Conditions ('Modes'). A system that is intended to operate in several different environmental conditions may be expected to behave very differently under these different conditions. For such systems, the expected behaviors under the different environmental conditions may be represented as different models. (For instance, the expected number and type of objects in an image may vary significant depending upon the time of day.)

Different Stakeholders. It may sometimes be the case that rather than developing individual bespoke systems for several different stakeholders, it is more efficient to develop a single system that is capable of meeting all their needs.

Generalising from the representation of MCS's as multi-models we identify three forms of relationship between the individual models within a Multi-Model framework:

(1) independent multi-models,

(2) integrated multi-models, and
(3) hierarchical multi-models.

We shall look at each of these relationships assuming, for ease of presentation, that there are just two individual models, a and b, in each case. Recall that each model (e.g., a) is defined by a set of assumptions ($\mathcal{A}^a$).

Independent Multi-Models. For independent multi-models, an implementation must assume that both sets of assumptions remain true at all times. It follows that all obligations are met. The multi-model is violated if, for example, $\mathcal{A}^a$ or $\mathcal{A}^b$ fails at run-time. It follows that there is just one mode of behaviour: (i) $\mathcal{A}^a \wedge \mathcal{A}^b$.

Integrated multi-models. Here an implementation may assume that one, or both, sets of assumptions hold. It follows that one set of assumptions may fail as long as the other set remain valid. Hence there are three modes of behaviour that are determined by these assumptions: (i) $\mathcal{A}^a \wedge \mathcal{A}^b$, (ii) $\neg\mathcal{A}^a \wedge \mathcal{A}^b$ and (iii) $\mathcal{A}^a \wedge \neg\mathcal{A}^b$.

Hierarchical Multi-Models. This is a special case of integrated multi-models that additionally satisfy a hierarchical relationship (mixed criticality systems are examples). Where a system degrades, from a model with $\mathcal{A}^a$ to one with $\mathcal{A}^b$ satisfying $\mathcal{A}^a \Rightarrow \mathcal{A}^b$ (and $O^a \Rightarrow O^b$) the assumptions and obligations are said to be *weakened*. Consequently one of the modes of behaviour ($\mathcal{A}^a \wedge \neg\mathcal{A}^b$) cannot arise, and hence we just have: (i) $\mathcal{A}^a \wedge \mathcal{A}^b$ and (ii) $\neg\mathcal{A}^a \wedge \mathcal{A}^b$. Indeed as $\mathcal{A}^a \Rightarrow \mathcal{A}^b$, (i) can be written simply as $\mathcal{A}^a$.

Figure 2 illustrates the constraints associated with these three model types.

Note that independent multi-models, in which all assumptions must always be satisfied, are really just a partitioning of the system's behavior and hence do not add to the expressive power of the modelling approach. It is the integrated and hierarchical multi-models that are novel constructs. Note also that it is possible for an integrated multi-model to include hierarchical elements, and this is discussed further in Section 4.4.

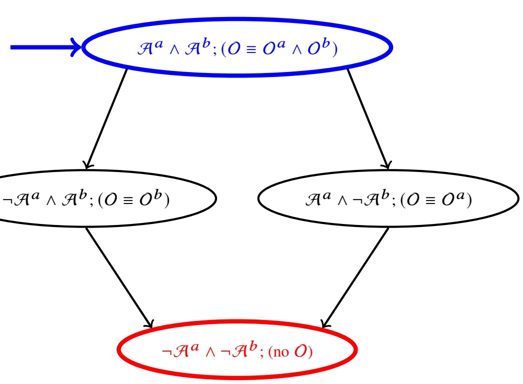

Figure 2: "Mode" changes in Multi-Models. Independent: no guarantees upon any transition out of the initial (blue) mode. Integrated: guarantees as shown. (Hierarchical: $A^a \Rightarrow A^b$, and hence the right-most path is impossible.)

The use of integrated multi-models will be illustrated in Section 4 by applying it to models of a typical classification system. The single model version of which is introduced in the next section.

3 A SINGLE-MODEL CLASSIFICATION SYSTEM

In this section we present a single-model specification scheme based upon Assumptions and Obligations for a classification system, and illustrate how timing analysis can be performed upon systems so specified. We will use the example of Single Input, Multiple Output (SIMO) classifiers to provide an application context for the purposes of illustrating our ideas. In systems such as Faster R-CNN [27], SIMO classifiers break down a complex image into a number of 'boxes' (RoIs – Regions of Interest) and then the content of each RoI is classified. (Note, the approach described in this paper is also applicable to YOLO (You Only Look Once) classifiers [25, 26].) To make things concrete, in Sec. 3.1 we introduce a toy example of the use of such a specification.

For software components such as classifiers it is necessary to define a workload and resource-usage model that will allow the worst-case input sequence to be derived and the worst-case execution time for this sequence to be computed. We assume that a single execution of the classifier involves analysing a sequence of RoIs that are required to be placed into one of a finite set of classes. There must be a bound on the number of RoIs and there may also be bounds on the number of entries in each class. In addition, there may be further (arbitrary) constraints over the mix of classes in the input sequence.

The proposed workload model uses Assumptions to capture the above constraints. We allow the cost (required execution time on the available computing resource) for each RoI to be class specific. Moreover, we allow these costs to be sensitive to knowledge that the classifier may have obtained from the input sequence that it has already processed. For example, if the applicable Assumptions imply that there can be at most one RoI of class C_x in any sequence of RoI's, then once such a RoI has been identified in a sequence the classifier may be able to reduce its execution time by simplifying the processing of subsequent RoI's – *the Assumptions can be relied upon*.

Below (Sec. 3.1) we first illustrate this modelling approach via a simple contrived example. We then show (Sec. 3.2) how the maximum execution duration for our example can be derived for a sequence of RoI's satisfying a given set of assumptions; this maximum execution duration immediately yields an obligation (*guarantee*) on whether the processing of the sequence of RoI's can meet a specified deadline or a predefined bound on the total execution time. In Section 4 this single model approach will be generalised so that a classifier can be subject to the requirements of more than one model.

3.1 An Example Classifier - CADIS

Our illustrative toy example[3] concerns a *CADIS* (for *Cat And Dog Identification System*), a software component that is tasked with identifying the breeds of all the cats and dogs that appear in an input

[3]This toy example is very loosely based on an *Identify Friend or Foe* (IFF) application system that uses DNN-based image processing to distinguish between friendly and hostile aircraft, and may further classify each kind.

image – see Fig. 3. Given such an image, an **Initial** component first breaks it down into a number of "*boxes*" (RoIs – Regions of Interest), each of which contains an image of interest (i.e., an image of either a cat or a dog) – we assume this takes an execution duration of one time unit per identified RoI. Each RoI is then passed on to a Cat Breed Classifier (**CBC**).

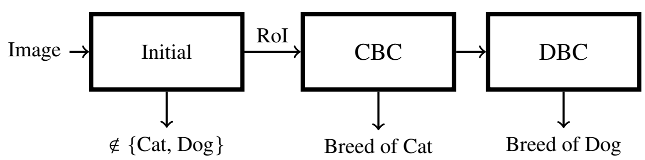

Figure 3: CADIS – A *Cat And Dog Identification System*

(1) The CBC first determines whether the image contained in this RoI is of a cat – we assume this operation takes at most two time units. If the answer is "no" (hence it must be a "dog") then this RoI is immediately passed on to the Dog Breed Classifier (**DBC**). If however the answer is "yes," the CBC processes the RoI further (taking up to an additional six time units to do so) to identify the actual breed of the cat.
(2) However if it is known (because it follows from the current state of the system and its assumptions) that the RoI passed on to the CBC cannot possibly contain the image of a dog, it follows that it must contain the image of a cat. In this event, the CBC can skip the first step and immediately begin processing the RoI to identify the cat breed (with at most six time units of processing).
(3) In a similar vein, if it is known that the RoI passed on to the CBC cannot possibly contain the image of a cat, the CBC immediately passes this RoI through to the DBC.

The DBC processes any RoI passed on to it to identify the breed of the dog in the RoI; we assume that such processing takes up to five time units.

To summarise the execution durations (or worst-case execution times – WCET's) of the three classifiers in Figure 3:

- Initial: The WCET is 1 on any RoI determined to contain an image of interest.
- CBC: If a RoI passed to it is known to contain a cat image, then its WCET is 6. If it is known to contain a dog image, then its WCET is 0 (since it can directly pass this RoI through to the DBC). If it is a priori unknown whether it contains a dog or a cat image, then its WCET is 8 (2+6) if it contains a cat image and 2 (2+0) if it contains a dog image.
- DBC: Any RoI passed on to it must contain a dog image; processing such a RoI has WCET of 5.

In the remainder of this section, we will seek to determine the tightest guarantees that can be made on the maximum duration taken to complete the processing of an unknown sequence of RoI's. We emphasize that the above description of both the functional behavior and the WCET numbers of the three components – Initial, CBC, and DBC – *comprise a part of the assume conditions*: they are part of the assumptions upon which our analysis may rely.

In the absence of any further assumptions, it is evident that the "worst" sequence (the one that requires the maximum duration to process) is one in which each RoI contains the image of a cat: for such a sequence, each RoI would experience a WCET of 1 in the Initial classifier, and $2 + 6 = \mathbf{8}$ in the CBC, for a total bound of $9N$ for N RoIs. So if N is bounded to be, for example, no greater than 4:

$$\mathcal{A} \stackrel{\text{def}}{=} N \leq 4$$

(i.e. no input image will contain more than 4 RoI's) then the maximum duration cannot exceed $(9 \times 4) = \mathbf{36}$.

In this Assumption, which can be looked upon as a predicate that holds true throughout the execution of the classifiers, N denotes the number of RoIs that have been passed from Initial to CBC. The Assumption predicate $\mathcal{A}$ is assumed to be true whenever CBC or DBC undertakes an action (i.e. executes an operation). So N can be thought of as a state variable that counts the number of RoIs seen thus far. Similar state variables, N_c and N_d, may denote the number of cat images and dog images that have been forwarded from Initial. Bounds on these values may also form part of the specification. The role of the Assumption predicate is to bound the work that the classifiers may be required to do. The simplest way of doing this is to bound N as the above example illustrates.

For a more interesting example, let us suppose that our assumptions additionally asserts that there will be at most two dogs and at most two cats in the sequence:

$$\mathcal{A} \stackrel{\text{def}}{=} N \leq 4 \wedge N_c \leq 2 \wedge N_d \leq 2$$

Each RoI will have a WCET=1 in the Initial classifier (for a total WCET of 4 for this classifier); let us compute the execution duration upon the other two classifiers for particular sequences.

- If the four animal images were to appear in the order ⟨D, D, C, C⟩ within the sequence, each would have a WCET of 1 in Initial. The first two would each have a WCET of 2 in the CBC followed by 5 in the DBC; hence, each would have a WCET of $1 + 2 + 5 = \mathbf{8}$. However, it will subsequently follow (from $\mathcal{A}$) that there are no more dog images in the sequence, and hence the remaining animal images do not need to be pre-processed in the CBC; each would consequently only experience a WCET of $(1 + 6)$. Summing over all four RoI's we have a duration equal to $8 + 8 + 7 + 7 = \mathbf{30}$.
- If, however, they were to appear in the order ⟨C, D, C, D⟩ within the sequence, it may be verified that only the last pet-image RoI (the last "D") would skip the first step in the CBC, for a total duration bound of $(1 + 2 + 6) + (1 + 2 + 5) + (1 + 2 + 6) + (1 + 5)$, or $\mathbf{32}$.

It may be verified by exhaustive enumeration (in Sec. 3.2 below, we obtain a more efficient means of doing so) over all possible orderings of the two dogs and the two cats in the RoI sequence that ⟨C, D, C, D⟩ represents the worst case and that $\mathbf{32}$ is consequently the duration bound under the assumption that there are at most two cat images and at most two dog images in the sequence of 4 RoI's.

Another, less intuitive, example is where the Assumption predicate asserts that there may be a maximum of 2 dogs and **3** cats in our 4-RoI sequence:

$$\mathcal{A} \stackrel{\text{def}}{=} N \leq 4 \wedge N_c \leq 3 \wedge N_d \leq 2$$

The above cases all still apply but there are additional sequences where there are 3 cats and 1 dog. For example, ⟨C, C, D, C⟩ gives 9+9+8+9 (=35). The same result occurs wherever the single dog appears in the first three RoIs.

The specification of the classifier is completed by asserting that the Obligation on the classifier, expressed as a Postcondition, Q, is that all pets have their species (type) and breed identified:

$$Q \stackrel{\text{def}}{=} \forall_i \bullet \mathit{Species}(\mathit{RoI}_i) \wedge \mathit{Breed}(\mathit{RoI}_i)$$

The index i is bounded by $\mathcal{A}$ (in effect $i \leq 4$ in the example). The predicates *Species* and *Breed* simply return TRUE when that attribute has been identified. This Postcondition is required to be true when the classifier completes. The other aspect of the component's obligations is that the execution time (e) of the classification system (Initial, CBC and DBC) is bounded to a known acceptable value, V. This is best expressed as a Guarantee condition ($\mathcal{G}$) [8]:

$$\mathcal{G} \stackrel{\text{def}}{=} e \leq V$$

In the above example if V is equal or greater than 35 then this obligation can be satisfied.

3.2 Determining the Maximum Execution Duration

We now generalize from the examples above, and devise a general procedure for determining the maximum duration needed to process an image, given an assumption asserting that there are at most $N^{\max}$ RoI's in the input image of pets (cats or dogs), of which at most $N_c^{\max}$ will be of cats and $N_d^{\max}$ of dogs. We will show below that we can *guarantee* to process this entire sequence of RoI's in an interval of duration not exceeding the value $F(N^{\max}, N_c^{\max}, N_d^{\max})$ obtained by solving the recurrence defined in Fig 4 for $F(N, N_c, N_d)$. This recurrence may be understood as follows:

(1) If N equals zero or if N_c and N_d both equal zero, then there can be no RoI of a pet; hence no RoI will be passed on from the Initial classifier to CBC (and subsequently to DBC). This is the base case. The cost of processing zero pets is of course 0.

(2) Else, if (N_d == 0) the CBC may assume that each RoI passed on to it must be of a cat, and hence skip the pre-processing and immediately move on to identifying the cat's breed, at a WCET of 6. Furthermore, it is evident that at most $\min(N, N_c)$ RoI's will be passed on from the Initial classifier to CBC.

(3) Analogously to the above case, if (N_c == 0) the CBC may assume that each RoI passed on to it *cannot* be of a cat and must hence be of a dog. It therefore immediately passes it on to the DBC, which will process it with a WCET of 5.

(4) It remains to consider when both $N_c \geq 1$ and $N_d \geq 1$. Observe that the maximum time required to process the entire sequence is the larger of the maximum processing time if (i) the first RoI in the sequence is of a cat, or (ii) it is of a dog:

(i) In the former case, the CBC would take a total of up to 8 time units to process the first pet-containing RoI, (since the pre-processing WCET on the CBC is 2, followed by a further WCET of 6 for the actual breed identification), after which the remainder of the sequence has at most $(N-1)$ pet-containing RoI's of which at most $N_c - 1$ are of cats and at most N_d of dogs.

(ii) In the latter case, the CBC would pre-process the RoI (WCET of 2) and pass it on to the DBC (WCET of 5 for identifying the dog-breed), after which the remainder of the sequence has at most $(N-1)$ pet-containing RoI's of which at most N_c are of cats and at most $N_d - 1$ of dogs.

A Dynamic Program. The recurrence in Figure 4 clearly demonstrates that the problem of computing $F(N, N_c, N_d)$ possesses the *optimal substructure* property (see, e.g., [11, p. 379]), and is hence amenable to solution as a Dynamic Program [2]. Notice that the recursive calls made in computing $F(N, N_c, N_d)$ are to $F(N-1, N_c-1, N_d)$ and $F(N-1, N_c, N_d-1)$ – in both cases, two of the three arguments are strictly smaller integers. Hence computing the values $F(x, y, z)$ in order and storing them in a table:

```
for x = 1 to N^max
    for y = 1 to min(x, N_c^max)
        for z = 1 to min(x, N_d^max)
            Compute and store F(x, y, z)
            // Using previously computed-and-stored F values
```

clearly has running time no worse that $O(N^{\max} \times N_c^{\max} \times N_d^{\max})$, implying an asymptotic complexity no worse than $\boldsymbol{O((N^{\max})^3)}$, for computing $f(N^{\max}, N_c^{\max}, N_d^{\max})$.

This straightforward derivation of a dynamic program contrasts with more complex optimal solutions such as model checking, controller synthesis, or two-player strategies. Moreover, the use of simple assumption predicates contrasts favourable with more comprehensive specification approaches such as guarded command languages, state diagrams etc. Nevertheless, the expressive power of the approach does seem to be sufficient to allow a wide range of constraints to be managed without recall to the use of these methods or heuristic (non-optimal) solutions.

3.3 A Bottom-up Implementation

Although it may seem more natural to solve the dynamic program obtained in Section 3.2 above in a top-down manner, here we apply a bottom-up approach since that more easily generalises to the multi-model case we will discuss in Section 4. Accordingly, let us first reformulate the recurrence to facilitate bottom-up implementation: let $Fc(T, TC, TD)$ denote the maximum cost of processing an image with T pets (RoIs), TC cats and TD dogs. It is readily seen that the bottom-up recurrence is

$$Fc(T, TC, TD) = \max\Big(Cc + Fc\big(T+1, TC+1, TD\big), \\ Dc + Fc\big(T+1, TC, TD+1\big)\Big)$$

where Cc is the cost of processing an extra cat (i.e. $TC+1$), and Dc is the processing cost of a further dog (i.e. $TD+1$). The iteration stops when $Fc(T+1, TC+1, TD)$ and $Fc(T+1, TC, TD+1)$ are both invalid; i.e. not sanctioned by the model. If both are valid then the maximum must be taken, with the cat costing Cc (8 in our running example) and the dog Dc ((2+5)=7). If only the cat possibility is valid then

$$Fc(T, TC, TD) = Cck + Fc(T+1, TC+1, TD)$$

$$F(N, N_c, N_d) = \begin{cases} 0, & \text{if } (N == 0) \text{ or } ((N_c == 0) \wedge (N_d == 0)) \\ 6 \times \min(N, N_c), & \text{if } (N_d == 0) \\ 5 \times \min(N, N_d), & \text{if } (N_c == 0) \\ \max \begin{pmatrix} 8 + F(N-1, N_c-1, N_d) \\ 7 + F(N-1, N_c, N_d-1) \end{pmatrix} & \text{otherwise} \end{cases}$$

Figure 4: Computing the worst-case cost of processing N RoI's, under the assumption that there are $\leq N_c$ cat images and $\leq N_d$ dog images.

```
type SoFar is array(0..MaxN, 0..MaxN) of integer
      with Default_Component_Value => -1
   S : Sofar

function Fc(TC, TD : integer) return integer is
   X,Y : integer := 0
   VD, VC : boolean
begin
   if S(TC,TD) > -1 then return S(TC,TD); end if
   VD := Valid(TC, TD+1)
   VC := Valid(TC+1, TD)
   if VD and VC then
     X := Cc + Fc(TC, TD+1)
     Y := Dc + Fc(TC+1, TD)
     X := max(X,Y)
     S(TC,TD) := X
     return X
   end if
   if VC then return Cck + Fc(TC+1, TD); end if
   if VD then return Dck + Fc(TC, TD+1); end if
   return 0
end Fc
```

Figure 5: Bottom-up implementation of the recurrence: Ada pseudo-code.

where Cck is the cost of a cat when the type of the input is known (so 6 in this example). And if only a dog is possible then

$$Fc(T, TC, TD) = Dck + Fc(T+1, TC, TD+1)$$

with $Dck = 5$.

In the above description, three parameter (T, TC and TD) are employed to illustrate the recurrence property. However, on inspective, it is clear that T (the number of pets) is always equal to $TC + TD$ (number of cats plus the number of dogs). Hence the implementation drops the T parameter.

An outline of the pseudo (Ada) code for the algorithm is given in Figure 5. The function returns one of four values: (i) the maximum of the two allowed paths, or else (ii) the value of taking a cat when only a cat is valid, or else (iii) the value of taking a dog when only a dog is valid, or else (iv) the value 0 as neither a cat nor a dog can be taken.

The array `S` holds previously computed values – that can be used to reduce the computational load. A simple two dimensional array is used in the pseudo code, with all elements in this array being initialised to -1.

Since the recurrence is bottom up, the initial call of the function is:

```
Cost := Fc(0, 0)
```

The call terminates and returns when a recursive call is made that has no valid successor (and hence returns 0).

The code implementing the function `Valid` is written according to the assumptions, and is therefore model-specific. For example, if there is a maximum of 6 pets, 3 cats and 4 dogs then the `Valid` function is simply:

```
function Valid(TC, TD : integer) is
begin
     return TC+TD <= 6 and TC <= 3 and TD <= 4
end
```

This gives a result of 51 which is delivered by the sequence ⟨C, C, D, D, D, C⟩.

The algorithm was coded in Ada and when executed on a normal laptop returns 'instantaneously' from relatively large models such as $T = 400$, $TC = 200$, and $TD = 250$; i.e., 400 RoIs, $\leq$ 200 cats and $\leq$ 250 dogs. For this particular example, the computed worst-case execution time for the classifier is 3400, and happens when a sequence of 199 cats is followed by 200 dogs and then a final cat. In this example the function Fc was called 128,976 times with the S array providing the (previously computed) answer on 48,326 occasions.

3.4 Extending the model – arbitrary constraints

The above example shows a model defined by the costs of each operation and a function that checks for a valid operation. The costs reflect assumptions made about the RoI. Typically, if something about the type of the RoI is known then the cost of the operation can be reduced. In the simple example above if the RoI is known to not be a cat then it may be passed directly to the dog classifier and its execution time reflects the fact that the input is definitely a dog. We re-emphesize that the assumptions are, in effect, *axioms* – they are true if the system behaves correctly, while if the system does not behave correctly then nothing need be guaranteed.

In addition to constraints concerning the number of RoIs and the maximum number of each type of RoI, it is possible to add further constraints that can help reduce the solution space for the algorithm. So, for example, if it is known (i.e. it is a valid assumption) that there are always more dogs than cats then `Valid` can reflect this:

```
function Valid(TC, TD : integer) is
begin
   return TC+TD <= N and
      TD + min(N-T, Nd-TD) > TC
end
```

The function returns true if `TC+TD` is not too large and if the number of dogs so far identified (`TD`) plus the minimum that could still be in the input image is greater than the number of cats so far identified (`TC`). If this is true then there is a possible future that will satisfy the constraint and hence this is a Valid step.

This example demonstrates the expressive power of the modelling technique being proposed. A wide range of constraints can be utilised. Some, for example incorporating cache effects that reduce the execution time of repeating steps (e.g. a cat after a cat), may require modifications to the recurrence formulation so that the history of identified RoI processed so far is available at each step; but this is not a fundamental change to the scheme and is easily incorporated.

Depending upon the kinds of assumptions that it is permitted to specify for a given application, determining satisfiability of assumptions may turn out to be considerably more complex than was the case in the earlier examples. Indeed, one could envision assumptions that are of arbitrary computational complexity to check – e.g., if one of our CADIS stakeholders were to specify an assumption that *the number of cats is the index, in some given standard encoding, of a Turing Machine that halts on all inputs*, then determining satisfiability of this assumption requires the solving of the Halting Problem and is thus undecidable. Although this example is admittedly very contrived and rather extreme, one could envision more plausible assumptions that similarly encode, say, some NP-complete problem. If checking the satisfiability of assumptions is computationally non-trivial, then efficiency considerations must take the computational complexity of doing so into account; it may be computationally more efficient to simply assume that some or all of the assumptions hold and thereby take on the responsibility of satisfying more obligations than may be strictly necessary.

As part of future work we plan to give further consideration to the properties of the constraints that are amenable to inclusion in the proposed modelling framework. In this paper we now focus on extending this classification example to illustrate Multi-Model specifications.

4 USE OF THE CADIS EXAMPLE TO ILLUSTRATE MULTI-MODEL SPECIFICATIONS AND ANALYSIS

We now extend the CADIS example to illustrate the use of a Multi-Model for classification. Suppose that the nature of the environment in which the classifier is to be deployed gives rise to two types of input image. As cats and dogs do not naturally share the same space, the image will either contain mainly dogs or mainly cats, but not significant numbers of both. Each of the two image types will have different assumptions. Alternatively, the CADIS may be used simultaneously by two stakeholders, one that is interested in determining the breeds of all the dogs in an image and the other, in determining the breeds of all the cats in the (same) image. Each stakeholder may again make different assumptions.

As before let N be a counter of the number of RoI's, N_c the number of these RoI's containing images of cats, and N_d the number of those containing images of dogs. The assumptions bound all of these counters. The image type that is predominantly populated with dogs is defined by the model, *DM*. A second model, *CM*, captures the properties of images that contain mostly cats.

Let the assumption predicate for the *DM* model be given by:

$$\mathcal{A}^{DM} \stackrel{\text{def}}{=} N \leq 8 \land N_c \leq 1 \land N_d \leq 7$$

and for *CM*:

$$\mathcal{A}^{CM} \stackrel{\text{def}}{=} N \leq 7 \land N_c \leq 6 \land N_d \leq 1$$

Both have the same Postcondition:

$$Q^{DM}, Q^{CM} \stackrel{\text{def}}{=} \forall_i \bullet Species(RoI_i) \land Breed(RoI_i)$$

and Guarantee condition:

$$\mathcal{G}^{DM}, \mathcal{G}^{CM} \stackrel{\text{def}}{=} e \leq V$$

Hence model *DM* allows up to 8 Pets with a maximum of 1 Cat and 7 Dogs; whereas *CM* allows up to 7 Pets, with a maximum of 6 Cats and 1 Dog. If both scenarios are to be catered for by a single model S then the assumption predicate must incorporate both extremes:

$$\mathcal{A}^{S} \stackrel{\text{def}}{=} N \leq 8 \land N_c \leq 6 \land N_d \leq 7$$

The algorithm of Section 3.3 reveals that the worst-case execution duration of just *DM* is 63, just *CM* is 60 and of S is 70.

However it is clear that the single model S covers combinations that are not possible; for example there cannot be 4 Dogs and 3 Cats in the same image. An integrated Multi-Model of *DM* and *CM* will more accurately specify how the classifier can behave, for example:

(1) The first RoI received from Initial will be pre-processed in CBC (WCET = 2) to determine whether it is of a cat or a dog. If the former, its breed is determined at an additional WCET of 6; if the latter, it is passed on to DBC which determines the dog-breed at an additional WCET of 5.
Suppose the outcome here were "cat" – from the perspective of *DM*, its assumption predicate implies that all following RoI's are of dogs. (Analogously if the outcome were "dog" the *CM* model will determine, based on its assumption predicate, that all following RoI's are of cats.)
(2) Our system seeks to satisfy the integration of both requirements. Hence regardless of the outcome above, neither assumption is invalidated and consequently the second RoI of interest must also be pre-processed.
Let us suppose that the outcome for this RoI is the opposite of the outcome for the first (i.e., the first two RoI's are either ⟨Cat, Dog⟩ or ⟨Dog, Cat⟩). The reader may verify that the maximum duration required in Initial, CBC and DBC for processing these two RoI's is $2 + 8 + 7 = \mathbf{17}$.
(3) The third RoI must also be preprocessed. Note that this preprocessing *necessarily invalidates one of the two assumptions* – if the outcome is "dog" then the assumption of the *CM* model no longer holds (analogously if the outcome is "cat" then the assumption predicate for the *DM* model is no longer valid).
Let us separately consider the possibilities when the preprocessing (WCET=2) reveals that this third RoI is of a) a dog or of b) a cat.
   a) If this turns out to be a dog image then the assumption of the *CM* model is not valid and henceforth our system need only seek to satisfy the requirement of the *DM* model. It may therefore assume that every subsequent RoI is of a dog, and consequently no pre-processing in CBC is needed; rather, the RoI is immediately passed through to the DBC which identifies the dog breed at a WCET cost of 5. Since there may be at most six such RoI's (including the current –third– one), the total processing duration does not exceed $6 + 6 \times 5 = \mathbf{36}$.

b) If, on the other hand, the third RoI turns out to be of a cat then the assumption defining the *DM* model is invalidated; henceforth our system need only seek to satisfy the requirement of the *CM* model. It will therefore assume that every subsequent RoI is of a cat, and consequently no pre-processing in CBC is needed; rather, the RoI is immediately processed to identify the cat breed (at a WCET 6). Since there may be at most 5 such RoI's (including the current one), the total processing duration does not exceed $5 + 5 \times 6 = \mathbf{35}$.

Summarising the discussion above, (i) worst-case duration for processing the first two RoI's is 17; (ii) pre-processing the third RoI takes a maximum duration of 2; and (iii) processing the remaining RoI's takes a maximum duration of either 36 (if of a dog) or 35 (if of a cat). Hence, the worst-case duration for a system to satisfy the requirements of this sequence is

$$17 + 2 + \max(36, 35) = \mathbf{55}$$

However, this sequence of images which has the property of satisfying both models for as long as possible is not the worst-case. Consider the sequence $\langle D, D, D, D, D, D, C, D\rangle$. After two RoIs the assumption of the *CM* model is broken and hence only the *DM* model applies, but because the allowed single cat does not appear until almost the end the preprocessing of all but the last RoI is required. This means that the worst case is

$$8 + (6 \times 7) + 8 + 5 = \mathbf{63}$$

We continue with the issue of using the Multi-Model to estimate the worst-case cost ($cost(MM)$) of the classification. As it is necessary to ensure that either (or both) of the assumptions remains true, the Multi-Model caters for each of the single models and hence:

$$cost(MM) \geq \max(cost(DM), cost(CM))$$

With this example the computed cost is as low as possible as $cost(MM) = 63$. This compares favourable with $cost(S) = 70$.

4.1 Necessary Properties for Integrated Multi-Models

To integrate *DM* and *CM* to form an effective single Multi-Model there are some necessary prerequisites:

- The two model assumptions are not inherently contradictory: it is possible for both to be true.
- If both assumptions are true then the obligations are complementary.

In the example

$$\mathcal{A}^{DM} \wedge \mathcal{A}^{CM} = N \leq 7 \wedge N_c \leq 1 \wedge N_d \leq 1$$

but as $N \leq N_c + N_d$ then

$$\mathcal{A}^{DM} \wedge \mathcal{A}^{CM} = N \leq 2 \wedge N_c \leq 1 \wedge N_d \leq 1$$

Hence a maximum of two pets, one cat and one dog; both of which will have their breeds identified.

A system that adheres to the integration of models *DM* and *CM* may experience various Modes of behaviour:

- Mode 1: $\mathcal{A}^{DM}$ and $\mathcal{A}^{CM}$ are true. Both sets of obligations are delivered
- Mode 2a: $\mathcal{A}^{DM}$ remains true, $\mathcal{A}^{CM}$ is false. Only obligations of *DM* are satisfied.
- Mode 2b: $\mathcal{A}^{DM}$ is false, $\mathcal{A}^{CM}$ remains true. Only obligations of *CM* are satisfied.
- Mode 3: $\mathcal{A}^{DM}$ and $\mathcal{A}^{CM}$ are false. No obligations are satisfied.

In Fig. 2, the top-most mode corresponds to Mode 1, the two modes depicted one layer down represent Modes 2a and 2b, and the mode depicted at the bottom represents Mode 3. A system that enters Mode 3 (from either 2a or 2b) has failed. A transition from Mode 2a to 2b, or vice versa, cannot be taken. Modes 1, 2a and 2b are all valid and legal.

We note, as illustrated earlier, that the worst-case cost does not necessarily occur when the system stays in Mode 1 for the longest time.

A final example illustrates that the estimate of the Multi-Model can lie between that of the combined model and the individual models. Let

$$\mathcal{A}^{DM} \overset{\text{def}}{=} N \leq 3 \wedge N_c \leq 0 \wedge N_d \leq 3$$

and for *CM*:

$$\mathcal{A}^{CM} \overset{\text{def}}{=} N \leq 3 \wedge N_c \leq 3 \wedge N_d \leq 0$$

then the combined single model is :

$$\mathcal{A}^{S} \overset{\text{def}}{=} N \leq 3 \wedge N_c \leq 3 \wedge N_d \leq 3$$

These give rise to the following computations: the cost of DM is 18, CM is 21 and S is 27. However the Multi-Model results in a cost of 23, which is higher than either of the individual models but lower that the combined single model.

4.2 How to compute the cost of the worst-case load

To compute the worst-case duration any input adhering to a Multi-Model specification requires only a trivial change to the algorithm given earlier. For the single model case a `Valid` function was required that checked that the next step in the recurrence was allowed (was sanctioned by the model). For the Multi-Model case this is simply extended:

```
function Valid(TC, TD : integer) is
begin
     return Valid_CM(TC, TD) or Valid_DM(TC, TD)
end
```

where `Valid_CM` and `Valid_DM` are the checks for each specific model.

When applied to the earlier example this dynamic program does return with the worst-case estimate of 63.

We note, for completeness, that for independent Multi-Models (where both models must be true at all times) then the following code is appropriate.

```
function Valid(TC, TD : integer) is
begin
     return Valid_CM(TC, TD) and Valid_DM(TC, TD)
end
```

Both models must sanction the step.

4.3 Discussion – Extending the Scope of the Approach

The CADIS example discussed above has the property that the temporal parameters of the models (Cc Dc, Cck, Dck) as illustrated in Figure 5 are constant; they are not a function of the model that is being applied (e.g. not a function of which of the single models is valid when the parameter is employed). But this constraint is not necessarily always true.

If we return to the example given in Section 4 then the worst-case sequence of RoIs was obtained from the DM model: $\langle D, D, D, D, D, D, C, D\rangle$. One interpretation of the DM model is that it applies to stakeholders that are only interested in determining the breeds of all the dogs in any input image. By the time a RoI is processed that has the sole cat the CM model has become invalidated. Hence only DM applies. Arguably the DM stakeholder is not interested in the breed of the solitary cat. And hence the cost associated with the cat should be only 2 not $2 + 6$. Giving an overall cost of **57** (not **63**).

To illustrate how this can be taken into account consider the parameter Cc which is the cost of determining the breed of an identified cat. In the examples discussed so far it has the constant value of 6. To make its value model-specific requires a simple modification to the code outlined in Figure 5, i.e. to include:

```
if Valid_CM then Cc := 6 else Cc := 0
```

Similar changes are needed to the other WCET parameters.

4.4 Integrated and Hierarchical Multi-Models

It was noted earlier that with a pure hierarchical model the assumptions are weakened as the system moves from one mode of operation to another, degraded, mode. This means, with two models with predicates `Valid1` and `Valid2`, then if `Valid1` is true then so is `Valid2`. The normal mode of operation is governed by the first model, the degraded mode by the second. In the degraded mode less will be achieved — i.e., the obligations are reduced. And it follows that the resources required will also be reduced.

So in the CADIS example rather than the classifier failing if there are more than N^{max} RoIs in the input image, we could define a degraded mode in which the type of the Pet within the RoI, but not the breed, is computed. So in the normal mode we had the assumptions and obligations as before:

$$\begin{aligned} \mathcal{A} &\stackrel{\text{def}}{=} N \le 4 \\ Q &\stackrel{\text{def}}{=} \forall_i \bullet Species(RoI_i) \land Breed(RoI_i) \end{aligned}$$

but in degraded mode (X):

$$\begin{aligned} \mathcal{A}^X &\stackrel{\text{def}}{=} N \le 10 \\ Q^X &\stackrel{\text{def}}{=} \forall_{i \in 1..4} \bullet Species(RoI_i) \land Breed(RoI_i) \ \land \\ & \qquad \forall_{j>4} \bullet Species(RoI_j) \end{aligned}$$

So if the number of RoIs is bounded by the initial aassumption then all Pets will have their type and breed identified. But if there is a 5th RoI then rather than fail, the system degrades to a mode in which only the species of the RoI is identified. To make this commitment it is still necessary to bound the load on the system. And if the number of RoIs now raises above 10 then even the degraded mode will fail.

Note in this simple example the two models have the appropriate hierarchical relationship as $\mathcal{A} \Rightarrow \mathcal{A}^X$.

It is of course acceptable to combine Integrated and Hierarchical Multi-Models. So again with the CADIS use case if $\mathcal{A}^{DM}$ and $\mathcal{A}^{CM}$ both fail then there could be a degraded model similar to the one given above that delivers only a partial classification.

5 CONCLUSIONS AND FUTURE WORK

We have proposed a framework for modelling and evaluating the worst-case execution times of complex software components such as classifiers. We have used a combination of assumptions and obligations to define a workload model and a resource (CPU time) requirements model. The assumptions are used to constrain potential paths through the software and hence deliver effective estimates of overall end-to-end timing behaviour. These estimates are obtained by utilising a bottom-up recurrence algorithm that only considers steps that are compliant with the defined assumptions. These assumptions are also used to identify input elements and sequences that are easier to process and hence lead to a reduction in the worst-case execution time.

Although single models are potentially useful, a strong motivation for the modelling approach adopted is to facilitate the combination of models into, what has been termed here, Multi-Models. The extensive literature on Mixed-Criticality systems has revealed a large number of applications where one model is used to describe the required behaviour in a "normal" mode of operation, and another the acceptable reduced behaviour in a "degraded" mode. These Multi-Model descriptions are mostly hierarchical – the degraded behaviour is a restricted form of the normal behaviour. In this paper we have generalised this relationship to also include independent and integrated Multi-Models. The integrated Multi-Model seems to be particularly effective at describing and analysing complex systems with multiple stakeholders or modes of operation.

In this first paper on these execution time Models and Multi-Models we used an artificial simple example to motivate and illustrate the main ideas. Readers will hopefully be able to appreciate that functionally similar applications (such as real-time classifiers and other AI inspired autonomous components) within future Cyber Physical Systems are likely to become increasingly common. For example, a road-side monitoring unit could take periodic photographs and be tasked with (a) estimating the real-time volume of traffic, (b) classifying the traffic into cars, vans, lorries, bikes, motor bikes etc, (c) estimate the total number of drivers/passengers for various combinations of these vehicle classes, taking into account the fact that a single photograph cannot simultaneously have a maximum number of each vehicle class, (d) identify the number of self-driving cars, (e) identify the number of cyclists not wearing helmets, etc. A combination of these requirements could be expected to lead to realistic independent, integrated and hierarchical Multi-Models.

There are a number of extensions that follow naturally from the work presented in this paper:

- For classifiers that have multiple components, such as IDKs [1, 12, 30], the order in which components are arranged can have a significant influence on the worst-case execution time of the classification. In future work we will use the framework developed to investigate this optimisation.

- In future work we will also give further consideration to the properties of the constraints that are amenable to inclusion in the proposed modelling framework.
- A required extension to the framework is to consider multiple concurrent components, their deadlines and system scheduling; for Mixed-Criticality Systems this has been addressed [8] within an assumptions/obligations formulation. In future work we will integrate this approach with the more general Multi-Model notion present in this paper.
- In the models presented in this paper the only failures considered are those caused by the input sequence failing to comply with the defined assumptions. It is also possible to introduce classification failures; e.g. a dog being wrongly identified as being a cat, and hence its breed not being ascertained unless it passes through both the CBC and DBC components. With such failures the Assumptions must be extended to include a Fault Model that bounds the number of such mis-classification. This addition will be described in detail in an extended version of this paper.

REFERENCES

[1] S.K. Baruah, A. Burns, and Y. Wu. 2021. Optimal Synthesis of IDK-Cascades. In *Proc. 29th International Conference on Real Time Networks and Systems, RTNS*. ACM.

[2] R. Bellman. 1957. *Dynamic Programming* (1 ed.). Princeton University Press, Princeton, NJ, USA.

[3] A. Benveniste, B. Caillaud, D. Nickovic, R. Passerone, J-B. Raclet, P. Reinkemeier, A. Sangiovanni-Vincentelli, W. Damm, T. Henzinger, and K.G. Larsen. 2018. Contracts for System Design. *Foundations and Trends in Electronic Design Automation* 12 (2018), 124–400.

[4] A. Burns. 2019. Multi-Model Systems – an MCS by any other name. In *Proc. 7th Int. RTSS Workshop On Mixed Criticality Systems (WMC)*. 5–8.

[5] A. Burns, S. Baruah, C.B. Jones, and I. Bate. 2019. Reasoning about the Relationship Between the Scheduler and Mixed-Criticality Jobs. In *Proc. 7th Int. RTSS Workshop On Mixed Criticality Systems (WMC)*. 17–22.

[6] A. Burns and R.I. Davis. 2017. A Survey of Research into Mixed Criticality Systems. *ACM Computer Surveys* 50, 6 (2017), 1–37.

[7] A. Burns and R.I. Davis. 2022. Mixed Criticality Systems: A Review (13th Edition). https://www-users.cs.york.ac.uk/burns/review.pdf and White Rose Repository: https://eprints.whiterose.ac.uk/183619/. (2022).

[8] A. Burns and C.B. Jones. 2022. An Approach to Formally Specifying the Behaviour of Mixed-Criticality Systems. In *Proc. 34th Euromicro Conference on Real-Time Systems (ECRTS) (Leibniz International Proceedings in Informatics (LIPIcs))*, Martina Maggio (Ed.), Vol. 231. Schloss Dagstuhl – Leibniz-Zentrum für Informatik, Dagstuhl, Germany, 14:1–14:23.

[9] J. Chen and X. Ran. 2019. Deep Learning With Edge Computing: A Review. *Proc. IEEE* 107, 8 (2019), 1655–1674.

[10] Y.F. Chen, E.M. Clarke, A. Farzan, M.H. Tsai, Y.K. Tsay, and B-Y Wang. 2010. Automated Assume-Guarantee Reasoning through Implicit Learning. In *Computer Aided Verification*, Tayssir Touili, Byron Cook, and Paul Jackson (Eds.). Springer Berlin Heidelberg, Berlin, Heidelberg, 511–526.

[11] T. H. Cormen, C. E. Leiserson, R. L. Rivest, and C. Stein. 2009. *Introduction to Algorithms.* (third ed.). MIT Press.

[12] R. Davis, S. Baruah, A. Burns, and Y. Wu. 2022. Optimally ordering IDK classifiers subject to deadlines. *Real-Time Systems* online (2022).

[13] R. Ernst and M. Di Natale. 2016. Mixed Criticality Systems?A History of Misconceptions? *IEEE Design & Test* 33, 5 (2016), 65–74.

[14] A. Esper, G. Neilissen, V. Neils, and E. Tovar. 2015. How Realistic is the mixed-criticality real-time system model. In *23rd International Conference on Real-Time Networks and Systems (RTNS 2015)*. 139–148.

[15] A. Esper, G. Nelissen, V. Nélis, and E. Tovar. 2018. An industrial view on the common academic understanding of mixed-criticality systems. *Real-Time Systems* 54, 3 (2018), 745–795.

[16] P. Graydon and I. Bate. 2013. Safety Assurance Driven Problem Formulation for Mixed-Criticality Scheduling. In *Proc. WMC, RTSS*. 19–24.

[17] T.A. Henzinger, S. Qadeer, and S.K. Rajamani. 1998. You assume, we guarantee: methodology and case studies. In *International Conference on Computer Aided Verification*. Springer Berlin Heidelberg, 440–451.

[18] C.B. Jones. 1981. *Development Methods for Computer Programs including a Notion of Interference*. Ph.D. Dissertation. Oxford University. Printed as: Programming Research Group, Technical Monograph 25.

[19] C.B. Jones and A. Burns. 2020. A Rely-Guarantee Specification of Mixed-Criticality Scheduling. arXiv. (2020). 2012.01493.

[20] C. Liu and J. Layland. 1973. Scheduling Algorithms for Multiprogramming in a Hard Real-Time Environment. *J. ACM* 20, 1 (1973), 46–61.

[21] S. Neema. 2019. Assurance for Autonomous Systems is Hard. https://www.darpa.mil/attachments/AssuredAutonomyProposersDay_ProgramBrief.pdf. (2019). Last Accessed: 2022-21-01.

[22] M. Paulitsch, O.M. Duarte, H. Karray, K. Mueller, D. Muench, and J. Nowotsch. 2015. Mixed-Criticality Embedded Systems–A Balance Ensuring Partitioning and Performance. In *Proc. Euromicro Conference on Digital System Design (DSD)*. IEEE, 453–461.

[23] D. Powell. 1992. Failure Mode Assumptions and Assumption Coverage. In *Proc. 22nd Int. Symp. on Fault-Tolerant Computing (FTCS-22)*. IEEE Computer Society Press, 386–95.

[24] K. Razavi, M. Luthra, B. Koldehofe, Max M. Muhlhauser, and L. Wang. 2022. FA2: Fast, Accurate Autoscaling for Serving Deep Learning Inference with SLA Guarantees. In *Proc. IEEE Real-Time and Embedded Technology and Applications Symposium (RTAS)*.

[25] J. Redmon, S. Divvala, R. Girshick, and A. Farhadi. 2016. You Only Look Once: Unified, Real-Time Object Detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

[26] J. Redmon and A. Farhadi. 2018. YOLOv3 Incremental Improvement. *CoRR* abs/1804.02767 (2018).

[27] S. Ren, K. He, R. Girshick, and J. Sun. 2016. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. (2016). arXiv, cs.CV, 1506.01497.

[28] N. Stoimenov, S. Chakraborty, and L. Thiele. 2012. *Interface-Based Design of Real-Time Systems*. Springer Berlin Heidelberg, Berlin, Heidelberg, 83–101.

[29] S. Vestal. 2007. Preemptive Scheduling of Multi-criticality Systems with Varying Degrees of Execution Time Assurance. In *Proc. Real-Time Systems Symposium (RTSS)*. 239–243.

[30] X. Wang, Y. Luo, D. Crankshaw, A. Tumanov, F. Yu, and J.E. Gonzalez. 2018. IDK Cascades: Fast Deep Learning by Learning not to Overthink. (2018). arXiv, cs.CV, arXiv, 1706.00885.

[31] S. Yao, S. Hu, Y. Zhao, A. Zhang, and T. Abdelzaher. 2017. DeepSense: A Unified Deep Learning Framework for Time-Series Mobile Sensing Data Processing. In *Proc. of the 26th International Conference on World Wide Web*. 351–360.