KATHOLIEKE
UNIVERSITEIT
LEUVEN

DEPARTEMENT TOEGEPASTE
ECONOMISCHE WETENSCHAPPEN

RESEARCH REPORT 0311

MODEL VALIDATION: AN INTEGRATED, FUZZY SET AND SYSTEM THEORETIC APPROACH

by
J. MARTENS

D/2003/2376/11

Model Validation: An Integrated, Fuzzy Set and System Theoretic Approach

Martens J.
Faculty of Economics and Applied Economics
Katholieke Universiteit Leuven
Naamsestraat 69
3000 Leuven
Belgium

March 2003

Introduction

There are many reasons thinkable why we eventually decide to build a model, and use it to study, explain or analyse a part of reality. Possibly, we are driven by plain *curiosity* to better understand how a part of reality works. In effect, this is the typical motive that we find in *physics*. A part of physical science intends to discover the rules of motion, the equations that govern the force of gravity, the mathematics that capture the principle of electromagnetic inference, the formal aspects of light and light breaking, and many other natural phenomena. For what purpose? Although debatable, the typical reason for the physical scientist to *model* comes from the desire for knowledge, the desire to be able to explain how a natural phenomenon works - all this for the primary goal of *comprehension*. The study of a part of reality may be motivated on other grounds than curiosity. Not the desire for knowledge *an sich*, but the desire to *improve* a current *unsatisfactory* situation, forms the core motivation for embarking on a modelling study. Stated otherwise, the *performance* of a real system is in need of improvement, and the goal of modelling is to evaluate alternative strategies that are proposed to realise this improvement. It is clear that this ground for modelling assumes that we have the capability to control a real system or to intervene to some extent, as we intend to bring the real system in *alignment* with the best of all alternatives for improvement of the current situation. A feeling of dissatisfaction with the current performance of a real system, and the awareness of potential improvements, constitute the typical mainspring for modelling that we find in *economics*, *management* and *operations research* - see e.g. [17, 26, 22]. It forms *the* underlying motive that we assume in this research report, to construct, to analyse and above all to *validate* models.

Since we conceive modelling as an activity that aims at evaluating alternative strategies for upgrading the current performance of a real system, we like to think of the entire modelling process as a *two-step* activity. In a first step, a model of the current situation is built. We refer to this model as a *primary model*. In a second step, the primary model is *modified* to analyse particular suggestions for improvement. We refer to a modified primary model as a *secondary model*. Secondary models answer typical *what-if* questions that allow us to measure the effect on the performance of certain changes in the primary model. They are competing plans of action to resolve the problem situation, and to remove the feeling of dissatisfaction with the current performance of the real system of interest. We think of the entire process of building a primary model, and modifying it into a number of secondary models as a process that aims to advance the *best* strategy for improvement - the *best* secondary model -, that is believed to yield the *highest* return on investment upon implementation, and that is expected to provide the *greatest* level of satisfaction for all model stakeholders.

As an immediate follow-up of interpreting the modelling process as a two-step activity, it goes without saying that we are eager to offer some level of *assurance* that the strategy for improvement that we finally retain, will indeed upon implementation bring about the betterment that we expected from it. What factors determine whether or not the level of expected improvement that we attached to a secondary model will indeed be *realised* when the real system is modified according to the strategy that is contained by the model? This question in fact probes for a measure of the *predictive adequacy* of a secondary model. This measure ought to give an indication of the *discrepancy* between the performance gain that will effectively be *realised* by implementing the strategy for improvement that is incorporated in the model, and the performance gain that is *estimated* from experiments with the secondary model. It needs little argument to recognise that such a measure is extremely hard to establish, as it requires us to impeccably foresee the performance gain that will be achieved in reality. However, since a secondary model is a modified primary model, it is only natural that the predictive adequacy of our *secondary model* will depend to some extent on the *accordance* of our *primary model* with the current situation in the real system of interest. In the assumption that we have the capability to gather data or observations from the real system of interest, then we can attempt to contrast real system observations with similar such data or observations that we obtain through experiments with a primary model. In this research report, we identify such data or observations with *behaviour*. In case data are obtained from a real system, we speak of *real system behaviour*. Similarly, in case data are gathered from a primary model, we speak of *primary model behaviour*. In that view, a primary model is as much in accordance with the current situation of a real system as the behaviour of the primary model strikes with that of its real counterpart. If the primary model on the basis of which we develop secondary models, is biased or contains fundamental flaws - in particular, if it exhibits behaviour that does *not* comply with behaviour obtained from the real system -, then to what extent can we justify the choice of a particular secondary model when this model is a modification of an inaccurate primary model? For, if we remain ignorant of the adequacy of our primary model, then what confidence can we grant to the claim that the performance gain that we attached to the best secondary model, echoes the *true* performance gain that we will eventually achieve in reality upon implementation of the strategy that is contained in this secondary model? Therefore, where a measure of predictive adequacy for a secondary model is too much to hope for, we like to attach to a secondary model a degree of *confidence* that reflects the accordance of the behaviour of the primary model on which the secondary model is based, with behaviour of the real system of interest.

In this research report, we define the process of *model validation* as an activity that labels a *primary model* with a degree of *accuracy*, reflecting the similarity of its behaviour with behaviour of the real system of interest. Our research goal is then to formally develop a method, that integrates aspects of *systems theory*, *stochastic processes* and *fuzzy set theory*, and that can be used by a model builder to evaluate the *degree* of accuracy of the *behaviour* of his *primary model* in view of similar such *real system behaviour.* In order to realise this goal, we define three research questions. A *first* research question involves the development of a general *framework* for our validation method. We develop such a framework in section 1. There, we discuss a modelling hierarchy, and expose in an *informal* way the nature of a primary model, its position in the hierarchy, and the notion of primary model behaviour. Also, we pin down the concept of a real system, and real system behaviour. We position *validation* in the framework and the modelling hierarchy, and distinguish it from *verification.* A *second* research question involves the formal development of the operands of our validation method. We will address this matter in sections 2 and 3. There, we develop a *structured discrete event system* and a *stochastic system* construct, identify a primary model with a stochastic system, and define primary model behaviour as a derivative of a stochastic system that involves *statistics* of *stochastic processes* that are specified by the stochastic system. We will postulate that real system behaviour stems from an unknown, underlying primary model, and thus can be seen as a derivative of an unknown stochastic system. Finally, we present in section 4 our *fuzzy set* theoretic based validation technique. The validation technique that we develop employs a *neuro-fuzzy* learning algorithm and a *resemblance relation* concept in fuzzy set theory, and allows to derive a measure of *similarity* between primary model behaviour and real system behaviour.

1 A framework for validation

In this section, we develop a framework for our validation method. We postulate the meaning of concepts like real systems, primary models, behaviour, and others. First, in subsection 1.1, we introduce real systems and real system behaviour. Then, we present a modelling hierarchy in subsection 1.2. The hierarchy lays down the difference between conceptual models, formal models and simulation models. Further, we introduce a number of additional postulates on the idea of an environment and a behaviour orientation, in order to come to a primary model and primary model behaviour in subsection 1.3. Also, we outline the structure of a typical validation problem for which our validation method may be suitable. Finally, in subsection 1.4, we integrate the concepts that are postulated in this section in the modelling hierarchy, and we situate where processes like *verification* and *validation* come in.

1.1 Real systems

In almost every debate on modelling and validation - see e.g. [16, 31, 3, 14] -, the process of *validation* is described as an activity that confronts a *model* with a *real system.* In light of the notion of a primary model that we introduced in the introduction, we may want to define validation as a process that requires two arguments: a *primary model* on the one hand, and the *real system* for which we developed the primary model on the other hand. Although this viewpoint on validation feels intuitively the right way to go - that is, validation should *ideally* contrast a model with *the* real system that the model aims to capture - it is too much to expect, for it implies that validation involves two constructs of which we can in fact *pinpoint* only one. Indeed, as we will see in sections 2 and 3, we follow a system theoretic and stochastic process approach to formally lay down what we mean by a primary model or a stochastic system. The stochastic system construct that we develop is an unambiguous, mathematical construct. The fact that we can pinpoint a stochastic system is in sheer contrast with the fact that we are unable to identify *the* (!) real system for which we developed the stochastic system. Naturally, we can always use our natural language to describe the problem situation that led to the development of the stochastic system, and consider the model as an attempt to deal with the problem situation. In case multiple model builders are confronted with *one and the same* problem situation, should we then not expect that their descriptions in a natural language of what their primary models actually stand for, will *differ* from one another? But then, should we conceive their models as different models of *one and the same* real system, or should we conceive their models as different models of *different* real systems? The problem that we face is that there is *no* unambiguous way to take segments of reality, to label them, and then to claim that a primary model is a model for one particular such labelled segment. Any attempt to pinpoint *the* real system of interest is therefore in vain. To illustrate this point, assume that we have built a primary model in response to an undesirable low departure punctuality of flights in an airline network. Assume that the primary model implements the *rotation* of aircraft at the stations in the network, and models activities like aircraft unloading, cleaning, refueling, boarding, etc. For that purpose, assume that it contains model logic for ground crew, air traffic management, flight operations and other components that play a role in the rotation of aircraft. In case we are asked to explain what our model stands for, then we may want to communicate in a natural language that our model mimics the rotation process of aircraft in an airline network, in an attempt to study the reasons for delays at departure, and incorporates therefore model logic for ground crew, air traffic management, flight operations, air traffic control, flight connections, etc. But then, haven't we just cut out a part of reality? Haven't we pinpointed *the* real system for which we developed our primary model? The answer is in the *negative.* Instead, we have revealed our *perception* of a real system - a *conceptual model* -, and have described this perception in a natural language. We come back to the notion of a conceptual model in subsection 1.2. Here, it is only natural to ask for the *nature* of the real system that we have perceived. In the following postulate, we lay down the nature of real systems,

Postulate 1 (Real system)
A *real system* is a family of building blocks of matter and matter interaction in the universe.

▷

According to postulate 1, a real system is nothing but a set of elementary constituents of the universe. In that respect, real systems are studied by the (quantum) physicist in a laboratory, and do not constitute the direct object of interest of modelling in light of this research report. Despite being unable to put the finger on *the* real system of interest in a modelling study, we do not want to rule out the real system concept from our argument, if only it constitutes the ultimate *source of data* [31] in regard to which we like to express the validity of a primary model. We identify this data with a *relation* between the Cartesian product of the range of some variables of interest and the Cartesian product of the range of other variables of interest. We call the former variables *independent* variables, and use the term *dependent* variables for the latter variables. In that respect, we postulate that for every observed relation between the Cartesian product of the range of independent variables of interest and the Cartesian product of the range of dependent variables of interest, there must be a real system - some family of building blocks of matter and matter interaction - that is responsible for, or that *induces* this relation. We call such a relation *real system behaviour.* In case of the aforementioned airline example, an independent variable of interest possibly measures the *average delay at arrival* of incoming aircraft at the central station in the network, or the *temporal correlation among successive such arrival delays.* A dependent variable of interest possibly measures the *average delay at departure* of outgoing aircraft at the central station, or the *relative frequency* of departure delays that are caused by *endogenous* delay reasons - delays which can be attributed to the station itself. We postulate the notion of *real system behaviour* generally as follows,

Postulate 2 (Real system behaviour)
Real system behaviour is a relation between the Cartesian product of the range of independent variables and the Cartesian product of the range of dependent variables of interest, induced by a real system.

▷

According to postulate 2, we must have identified a number of independent and dependent variables before we can speak of real system behaviour. Having identified independent and dependent variables, *every* relation between the Cartesian product of the range of the independent variables and the Cartesian product of the range of the dependent variables that is *induced* by a real system, satisfies to be called *real system behaviour.* Clearly, a complete theory of reality is required if we want to determine *which* particles in the universe influence the independent and the dependent variables that we have chosen, and eventually determine the relation that we have retained as real system behaviour. The underlying real system that is responsible for the values that we observed of independent and dependent variables, is in fact of no importance. Stated otherwise, all that matters to us is the *trace* that a real system leaves in the universe that is set up by the range of independent and dependent variables of interest. This trace, being a *behaviour relation*, forms one of the operands of our validation approach.

1.2 A modelling hierarchy

In many general discussions on modelling - see e.g. [2, 15, 22, 31] -, we encounter a certain *hierarchy* among models. In the following, we intend to compose from the literature a suitable modelling hierarchy. Then, once we have postulated the concept of a primary model, we position in subsections 1.3 and 1.4 the primary model concept in the hierarchy, and develop a *structure* for the typical problem situation in which our validation approach may be applied to assess the adequacy of a primary model. The modelling hierarchy that we present in the following, consists of a *conceptual* layer, a *formal* layer, and an *implementation* layer.

We postulate that a model at the conceptual layer of the hierarchy coincides with a *cognitive image*, comprising components, objects, (causal) relations between components, resources, entities, attributes of resources, parameters, events, assumptions, hypotheses, etc. that are *believed* to make up a necessary pool of ingredients to explain real system behaviour of interest. Since a conceptual model is developed to resolve a particular problem situation, it is influenced by the goals that are put forward in a modelling study. Further, every conceptual model is part of a *world view*, which forms an idiosyncratic, problem dependent and cognitive representation of the real world - see e.g. [5] on the concept of world views. A conceptual model expresses a certain *angle of attack* [22] to deal with a problem situation. For a *common* problem situation, we can expect every model builder to have its own particular world view. Hence, there will likely be a multitude of *different* conceptual models, conceived by these model builders when they attempt to resolve the problem situation. We refer to conceptual models as *perceived* *systems*, as they reside in the minds of the modellers [20]. They are in a way *perceptions* of real systems. We postulate a conceptual model as follows,

Postulate 3 (Conceptual model)
A *conceptual model* is an idiosyncratic and cognitive image in a world view of a model builder, that comprises objects, components, interactions, resources, entities, parameters, events, relationships, attributes, assumptions, hypotheses, etc. that are believed to be relevant to explain real system behaviour.

▷

A conceptual model embodies a *theory* to account for how independent variables affect dependent variables - albeit in an implicit way. In light of our validation method, we desire that a conceptual model also accounts for *how* independent variables themselves are established. To give a short example, assume that we are interested in modelling a simple coin striking process. In particular, assume that our interest goes to the effect of the *rate of arrival* of coin planchets (unfinished coins) on the *utilisation* of the striking machine. In that particular case, we desire that a conceptual model of the entire striking process not only tries to give an account for how *arrival rate* influences *striking machine utilisation*, but also that it carries a rationale for how the arrival rate itself is established. Possibly, the conceptual model embraces a theory to explain the complete production of the coin planchets themselves in order to determine this arrival rate. We whish to agree here that the part of a conceptual model that explains the behaviour of *independent* variables is *autonomous* in nature, in that we do not look out for other variables that can be seen as independent variables with respect to the former independent variables. Thus, very simply, when we speak of a conceptual model, we have in mind a model that provides *both* an explanation for the effect that independent variables have on dependent variables, *and* a theory for how independent variables themselves are established - all this in an implicit and informal way. We refer to the part that formulates a theory of how independent variables affect dependent variables as the *main* part of a conceptual model. We refer to the part that explains independent variables as the *input* part of a conceptual model. In case of the simple coin striking process, the input part of a conceptual model may specify that coin planchets arrive according to a so-called *Poisson arrival process* with an unspecified arrival rate. The main part of the conceptual may then conceive the striking process itself as a *single-queue-single-server* system with unlimited queue capacity, and an unspecified service rate.

Under the conceptual modelling level, we encounter a *formal* modelling level. At this level, a conceptual model is *calibrated* and made *explicit* with the help of a formal specification language. The fact that we let a conceptual model include both a main part and an input part, re-enters in the following postulate of a formal model,

Postulate 4 (Formal model)
A *formal model* is a calibrated, structured image of the main and/or the input part of a conceptual model, written in a particular formal specification language.

▷

With the term *structured* in postulate 4, we want to emphasise that a formal model is a *regular* image, that explicitly contains a certain system *structure* that remained only indefinite in a perceived system. Stated otherwise, a formal model is a *white-box* derivative of a perceived system, a completely spelled out and formal theory to explain real system behaviour. We like to think of the main and the input part of a conceptual model as *uncalibrated* theories. In that respect, we conceive a formal model as a formal *calibration* [3] of the main and/or the input part of a conceptual model. By that, we mean that a formal model lays down particular *values* for key *parameters* in the conceptual model. In the aforementioned coin striking example, such parameters may include the service rate of the striking machine, the queuing discipline of the buffer out of which the striking machine samples coins, or possibly the rate at which coin planchets are cut out of a strip of metal and advanced to the striking process.

Finally, at the implementation layer, we position the *simulation model.* A simulation model is nothing more than an implementation on a computer of a formal model, for ease of experimentation and analysis. Although we could in theory perform experiments *manually* with a formal model, this is in practice not a viable option. Instead, we want a computer program for *automated*, fast experimentation. We postulate a simulation model here as follows,

Postulate 5 (Simulation model)
A *simulation model* is a program, that implements a formal model in a particular computer programming language.

▷

1.3 Primary models

In view of the modelling hierarchy that we presented in subsection 1.2 and in light of our validation method, we like to think of the outcome of a modelling process in terms of a number of *formal models.* Some of these models are formal calibrations of the *main* part of a conceptual model, while others are formal calibrations of the *input* part. For any part of a conceptual model, we like to agree that the formal models for that part differ from one another *only* in the values that are given to some key parameters. The fact that we let a single conceptual model bring about a multiple of formal models raises the issue *how* we should *integrate* all these models in the process of validation. To be more precise, given some real system behaviour of interest, should we validate each *main* model individually in light of the observed behaviour, without considering *input* models? Should we validate main and input formal models separately, and then integrate the obtained measures of validity for every combination of main and input model that we can make? Or should we measure the accuracy of all main and input formal models at once? In our method for validation, input model validation is *intertwined* with main model validation. For, we like to assemble alternative input models in an *environment.* In view of the structured discrete event system formalism that we develop in this report, we think of an environment as a family of structured discrete event systems, each one of which generates input for a main model - again, a structured discrete event system - in an *autonomous* way. The autonomous nature of models in an environment simply means that these models do not get their input from other models. This is in contrast with a main model, input for which we always generate through a particular autonomous model in an environment. We postulate the notion of an environment here as follows,

Postulate 6 (Environment)
An *environment* is a family of autonomous, calibrated formal models of the input part of a conceptual model.

▷

As a short illustration of an environment, let's reconsider the simple coin striking process that we introduced earlier. Assume that we are dissatisfied with the average number of coins that are produced per hour in the real production process. To improve the system throughput, we want to study the effect of upgrading the striking machine. Within this problem situation and modelling goal, we require a model of the striking process itself, *and* a model of the process that is responsible for the generation of coin planchets. In case we have decided *conceptually* to model coin planchet arrivals by a so-called *Poisson arrival process*, then an environment will - in light of the specification language that we employ - comprise autonomous structured discrete event systems that produce coin planchets. Each system in the environment models a Poisson arrival process at a *different* arrival rate. If we then pin down a *single* formal model of the *main* part of the conceptual model of the striking process, then we can study the effect that *workload* (arrival rate) has on *throughput* for the main formal model-environment pair that we have identified.

In light of the short example that we presented above, we state that a *primary model* embraces - among other issues - an environment of systems that model the *generation* of input on the one hand, and a single main formal model that models the *processing* of input on the other hand. In a way, we specify in the following postulate the nature of the *modelling part* of a primary model,

Postulate 7 (Primary model)
A *primary model* is a construct that contains - among other issues - an environment and a calibrated main formal model to process input from models in the environment.

▷

Comparing postulate 7 with the definition of a stochastic system that we develop in section 3, the postulate is somewhat incomplete in that it does not explicitly mention the fact that we like a primary model to define also the *experiments* that we are allowed to perform with the models that the primary model contains, as well as the *stochastic processes* that we intend to retain as outcome of these experiments. We conceive these experiments and the stochastic processes as *add-ons* to a main formal model and an environment, in order to achieve a *primary model.* Running a little ahead of things, one part of a primary model pins down a *main* structured discrete event system and an environment of *input* structured discrete event systems - this part is emphasised in postulate 7 -, while another part specifies the experiments or *replications* that we are allowed to perform, and lays down a number of *stochastic processes.* As we will point out, we use *statistics* of these stochastic processes to constitute *primary model behaviour.* We will capture the precise connection between statistics of stochastic processes that are induced by experiments with formal models of a primary model, and behaviour of the primary model itself, by the concept of a *behaviour orientation*, that we postulate as follows,

Postulate 8 (Behaviour orientation)
A *behaviour orientation* is a window on statistics of stochastic processes, that are induced through experiments with the main formal model and models in the environment of a primary model.

$\triangleright$

Thus, very simply, we think of a behaviour orientation as a kind of *filter* that specifies, given a family of stochastic processes, how statistics of these stochastic processes yield observations on independent and dependent variables. Since other behaviour orientations may yield other behaviour, we like to speak of behaviour of a primary model *through* a behaviour orientation. In that view, we postulate primary model behaviour as follows,

Postulate 9 (Primary model behaviour)
Primary model behaviour is a relation between the Cartesian product of the range of independent variables and the Cartesian product of the range of dependent variables of interest, induced by a primary model through a behaviour orientation.

$\triangleright$

Figure 1 puts some of the concepts that we introduced in perspective. On top of the figure, we placed a conceptual model $\mathbb{C}_i$ that we have constructed in response to a particular problem situation.[1] On the lowest level in the figure, we placed behaviour of a real system $\mathbb{R}$. Uncertainty with respect to some key parameters in the *main* part of $\mathbb{C}_i$, makes that we whish to retain at the formal level a number of *different* main formal models. In the figure, we used an index set $J \triangleq \{1, 2, \ldots, n\}$ to index the different calibrations of the main part of the conceptual model. The formal model that we obtain for a calibration $j \in J$ is denoted in the figure by $\mathbb{F}_{i,j}$. Assuming in the aforementioned coin striking example that the rate at which coins are struck on the striking machine is the only key parameter in the main part of our conceptual model,

[1]If we let I be an index set to index different model builders, then we can associate with every index in I a scheme like that of figure 1.

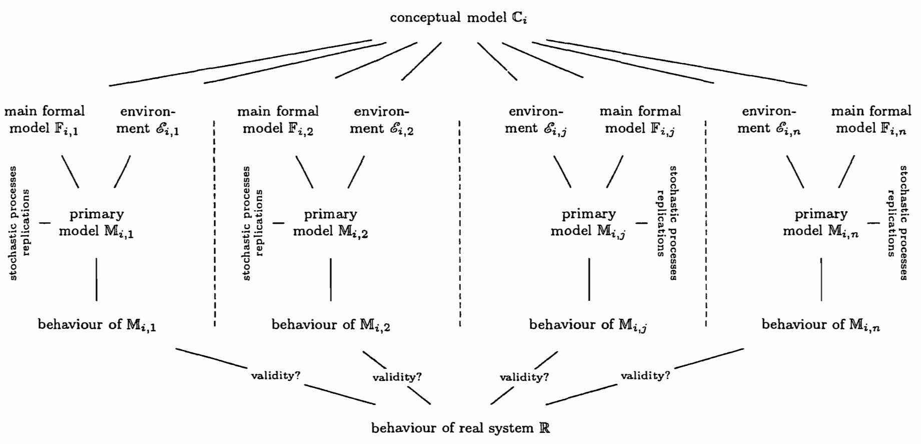

conceptual model $\mathbb{C}_i$
main formal model $\mathbb{F}_{i,1}$
environment $\mathscr{E}_{i,1}$
main formal model $\mathbb{F}_{i,2}$
environment $\mathscr{E}_{i,2}$
environment $\mathscr{E}_{i,j}$
main formal model $\mathbb{F}_{i,j}$
environment $\mathscr{E}_{i,n}$
main formal model $\mathbb{F}_{i,n}$
stochastic processes replications
primary model $\mathbb{M}_{i,1}$
primary model $\mathbb{M}_{i,2}$
primary model $\mathbb{M}_{i,j}$
primary model $\mathbb{M}_{i,n}$
behaviour of $\mathbb{M}_{i,1}$
behaviour of $\mathbb{M}_{i,2}$
behaviour of $\mathbb{M}_{i,j}$
behaviour of $\mathbb{M}_{i,n}$
validity?
behaviour of real system $\mathbb{R}$

Figure 1: A framework for validation

then every main formal model $\mathbb{F}_{i,j}$, $\forall j \in J$ installs a particular *striking rate* of the striking machine. As we pointed out earlier, we are likely to be affected by uncertainty with respect to some key parameters in the *input* part of $\mathbb{C}_i$ as well. At the formal level, this translates into a family of input formal models, or an *environment.* In the *usual* case, we want to group these input formal models in a *single* environment. However, there are some cases conceivable in which it is preferable to let the environments differ - albeit only slightly - from one another.[2] In view of the coin striking case that we develop and that will frequently re-appear in this report, one can safely set all environments equal to one another. The environments in figure 1 are denoted by $\mathscr{E}_{i,j}$, $j \in J$. Taking back the coin striking example, in case we have decided e.g. on a *minimum* and a *maximum* arrival rate of coin planchets to the striking process, then an environment might comprise an input formal model for every rate in between this minimum and maximum arrival rate.

As we indicated earlier, a primary model comprises an environment, a main formal model, replications and stochastic processes. Primary models in the figure are denoted by $\mathbb{M}_{i,j}$, $j \in J$. Given these primary models and a suitable behaviour orientation, we are now faced with the dilemma of picking the *best* primary model in view of the real system behaviour that we have available. In effect, as we intend to use a primary model to develop *secondary models*, which are then in turn used to evaluate the benefit of competing strategies of improvement, we like to discriminate *superior* from *inferior* primary models. When is a primary model superior to another primary model? Here, we state that the adequacy or *validity* of a primary model depends on the *similarity* of the behaviour that it induces through the behaviour orientation,

[2]The cases that we have in mind in which we may prefer to let environments differ from one another, lie beyond the scope of this research report.

with regard to the observed real system behaviour. The very assessment of this similarity is precisely where we like our validation method to come in. We expect thus from our technique that it assists the model builder in determining which primary model he should eventually retain to develop secondary models.

1.4 Verification, validation and the modelling hierarchy

In figure 2,[3] we positioned the notion of a primary model and primary model behaviour in the modelling hierarchy. Also, we situated where activities as *verification* and *validation* take place in the overall picture. The solid arrows in the figure represent typical *modelling* actions. These include *perceiving* and *experimenting* with a real system, *specifying* and *calibrating* a conceptual model in a number of formal models, and *implementing* these models on a computer. The dashed arrows indicate where *verification* and *validation* come in. In short, verification is a process that operates *within* the modelling hierarchy, whereas validation is a process that extends *beyond* the hierarchy, and always involves a real system - real system behaviour in particular.

The large rectangle in the upper right of the figure is meant to represent *reality.* We postulate reality as a source of data, that can be structured as a family of *real systems* with labels. Recall that a real system was postulated as a family of building blocks of matter and matter interaction. We argued that our interest is not in real systems *an sich*, but in the *trace* that real systems leave in a universe, set up by the range of measurable independent and dependent variables of interest. We referred to such a trace as *real system behaviour*, and argued that we like to think of modelling as an activity that attempts to provide an explanation for such behaviour.

We postulated a conceptual model as an idiosyncratic, cognitive image of *all that*, that is believed to be relevant to explain real system behaviour. A conceptual model is part of the world view of a model builder, and depends on the modelling goals that have been put forward. We pointed out that a conceptual model is inflicted with a problem of *calibration.* Some key parameters have to be decided upon, and we are unsure what decision is the *best* decision to make. Therefore, at the formal level of modelling, a conceptual model yields a number of *primary models.* Each such model comprises a certain calibration of the *main* part of the conceptual model - a *main formal model* -, and a number of calibrations of the *input* part of the conceptual model - organised in an *environment.* We mentioned that, in order to obtain a primary model, we must define the experiments that we allow ourselves to perform with a main formal model and with the models in an environment, as well as a number of stochastic

[3]In the figure, *PFM* stands for *primary formal model*, while *PSM* stands for *primary simulation model.* Also, we displayed on the left side of the figure, the languages to express conceptual models, primary models and simulation models. The simulation language that we use to run experiments, is called SIMAN. Good expositions on SIMAN are widely available in the literature - see e.g. [12, 25].

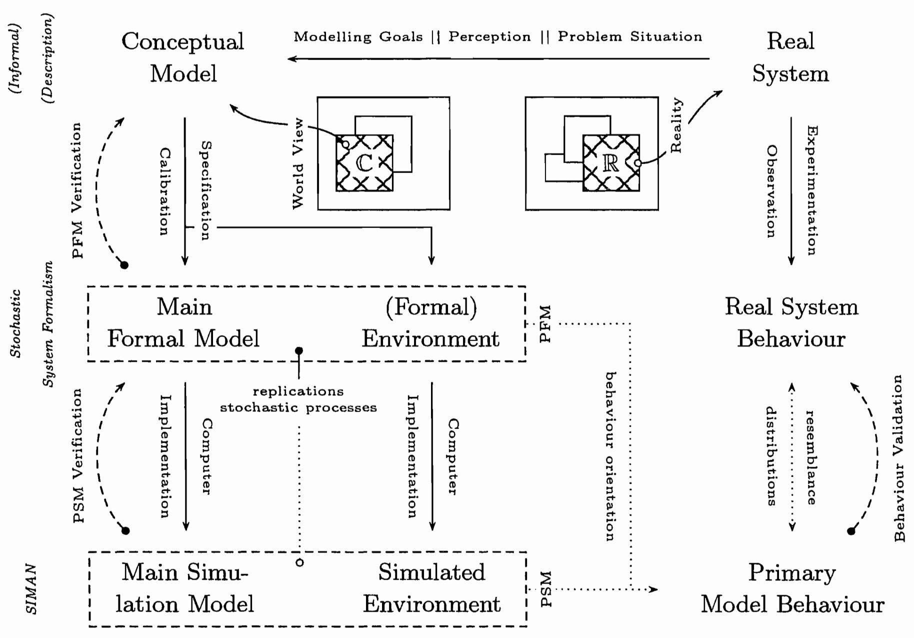


Figure 2: Verification, validation and the modelling hierarchy

processes. Once a primary model is fully established, we can distillate behaviour from the model through a *behaviour orientation.*

In the figure, we placed primary formal model and primary simulation model *verification* on the one hand, and behaviour *validation* on the other hand. Verification is a process that does not leave the modelling hierarchy. It questions the adequacy of a model in view of a model at a higher level in the hierarchy. We can thus address the adequacy of a primary simulation model in relation to the primary (formal) model that it claims to implement. This kind of verification investigates whether we correctly *simulated* the formal model specifications of a primary model on a computer. We can also address the adequacy of a primary (formal) model in relation to the conceptual model that it claims to calibrate. This kind of verification is concerned with the fact whether or not we correctly formalised the components, component interactions, assumptions, hypotheses, etc. that are contained in a conceptual model. In contrast to verification, validation *does* leave the modelling hierarchy. We define validation as an activity that compares real system behaviour with primary model behaviour. The technique that we develop in this report offers a way to determine the *degree* of similarity between primary model and real system behaviour.

2 Structured discrete event systems

In this section, we develop a *structured discrete event system* construct that we like to use to specify formal models. The structured discrete event system construct is an extension of the classic discrete event system construct, and is inspired on the concept of so-called *structured machines* and *discrete event networks* [31]. Other approaches to model discrete event-like systems can be found in [4]. We decided to develop a structured discrete event system formalism, as we felt that there was some room for improvement of the original discrete event system formalism of [31]. In addition, the original discrete event system formalism exhibits some shortcomings that prevented us from employing the formalism right-away in light of our validation method.

Although we start our presentation here with an introduction to general systems theory in subsection 2.1, we assume that the reader is already familiar with most of the system theoretic concepts that we present. Good expositions on general systems theory include [19, 31, 30, 32, 28]. In subsection 2.2, we pin down the notion of abstract objects and abstract systems, which are general system theoretic constructs that we need to define the operands of our validation method. Then, we introduce in subsections 2.3 through 2.6 the necessary discrete event system terminology, and expose our structured discrete event system construct. We postpone an example of our system construct until subsection 3.4.

2.1 System theory fundamentals

In order to build a formal model - irrespective of whether we are dealing with the main or the input part of a conceptual model -, we identify a number of *descriptive variables*. Some descriptive variables are *input* variables, representing *facts*, that are determined *externally*, possibly by an experimenter. Some of the non-input variables are needed to derive *primary model behaviour*. We call those non-input variables *output* variables. Where input and output variables represent *causes* and *effects* respectively, *state* variables form a kind of *bridge* in between the causes and the effects. In a way, they constitute the *memory* of a system. In the following definition, we use the concept of input and output variables to lay down the notion of an *input* and *output set*.

Definition 1 (Input and output set)
For x_i, $i \in I$ input variables, y_j, $j \in J$ output variables, indexed by finite index sets I and J, X_i and Y_j the range of the respective variables x_i and y_j for all $i \in I$ and $j \in J$, the Cartesian product $X \triangleq \prod_{i \in I} X_i$ and the Cartesian product $Y \triangleq \prod_{j \in J} Y_j$ are called an *input* and *output set* respectively.

□

In order to be able to speak of *behaviour* of descriptive variables, we require the notion of *time* and of a *time set*. We formally define a time set as follows,

Definition 2 (Time set)
An arbitrary set T is called a *time set* iff. T is equivalent to the positive real line $[0, +\infty[$.

$\square$

Members of a time set are called *time points.* In the remainder of this report, we will always assume that a time set T equals the positive real line $[0, +\infty[$. Let now x_i, $i \in I$ be descriptive variables, indexed by a finite index set I. Denote the range of every variable x_i, $\forall i \in I$ by X_i. We call a function from a time set T into $\prod_{i \in I} X_i$ a *time function.* Formally,

Definition 3 (Time function)
For T a time set, x_i, $i \in I$ descriptive variables, indexed by a finite index set I, and X_i the range of x_i for every $i \in I$, a function from T into $\prod_{i \in I} X_i$ is called a *time function.*

$\square$

Depending on whether the variables that are indexed by the index set I in definition 3 are input, state, or output variables, we speak of an *input* time function, a *state* time function, or an *output* time function. We use time functions to describe the behaviour of descriptive variables for an *unlimited* period of time. We call the universe of all possible time functions a *time function set.* Formally,

Definition 4 (Time function set)
For T a time set, x_i, $i \in I$ descriptive variables, indexed by a finite index set I, and X_i the range of x_i for every $i \in I$, the family of all time functions from T into $X \triangleq \prod_{i \in I} X_i$, denoted by $\mathscr{F}(T, X)$, is called a *time function set.*

$\square$

We call the restriction of a time function from a time set T into a set X to an *half-open* interval $]t_0, t_1]$ of T such that $t_1 > t_0$, a *trajectory* or *segment* over $]t_0, t_1]$. We use trajectories or segments to describe the behaviour of descriptive variables over a *limited* period of time. Formally,

Definition 5 (Trajectory/Segment)
For T a time set, x_i, $i \in I$ descriptive variables, indexed by a finite index set I, X_i the range of x_i for every $i \in I$, f a time function from T into $\prod_{i \in I} X_i$, and $T' \triangleq]t_0, t_1]$ an half-open interval of T such that $t_1 > t_0$, the restriction of f to T' is called a *trajectory* or *segment over T'.*

$\square$

As we did for time functions, we speak of *input*, *state* and *output* trajectories, depending on whether we are dealing with input, state or output variables. Where a time function set embodies all possible behaviour of some descriptive variables over an *unlimited* period of time, a *trajectory* or *segment set* embodies all possible behaviour over *any limited* period of time. Formally,

Definition 6 (Trajectory/Segment set)
For T a time set, x_i, $i \in I$ descriptive variables, indexed by a finite index set I, X_i the range of x_i for every $i \in I$, and $X \triangleq \prod_{i \in I} X_i$, the family of all restrictions of time functions from T into X to half-open intervals of T, denoted by $\mathscr{S}(T,X)$, is called a *trajectory* or *segment set.*

□

In general systems theory, the family of all segments of an input segment set that are *allowed* to be applied to a model, is called an *allowable input segment set.* Members of an allowable input segment set are called *allowable input segments.* Let Ω be a subset of an input segment set $\mathscr{S}(T,X)$. Then Ω is called an allowable input segment set if and only if it satisfies a number of desirable properties. A first property concerns the fact that, when an input segment $\omega \in \mathscr{S}(T,X)$ over an interval $]t_0,t_1]$ is allowable - and hence belongs to Ω -, then we like that the restriction of ω to every half-open interval $]t_0,t]$ such that $t_0 < t < t_1$, is also an allowable input segment. This requirement follows readily from the reasonable assumption that we can at any time *interrupt* an experiment. We call the restriction of a segment ω over an interval $]t_0,t_1]$ to an interval $]t_0,t]$ the *left segmentation of ω to t.* In the following definition, we define left segmentation to a time point for a segment in an *arbitrary* segment set.

Definition 7 (Left segmentation)
For $\mathscr{S}(T,X)$ a segment set, $s \in \mathscr{S}(T,X)$ a segment over an interval $]t_0,t_1]$, and t a time point in T such that $t_0 < t < t_1$, the *left segmentation of s to t*, denoted by $s_{t]}$, is a segment over $]t_0,t]$, defined in every $t' \in]t_0,t]$ by

$$s_{t]}(t') \triangleq s(t') \tag{2.1}$$

□

If we can *interrupt* an experiment, then it is only reasonable to assume that we can *continue* an interrupted experiment. In particular, if ω is an allowable segment over $]t_0,t_1]$, and if t is a time point for which holds that $t_0 < t < t_1$, then we like that the restriction of ω to $]t,t_1]$ is also an allowable input segment. We call this restriction the *right segmentation of ω to t*,

Definition 8 (Right segmentation)
For $\mathscr{S}(T,X)$ a segment set, $s \in \mathscr{S}(T,X)$ a segment over an interval $]t_0,t_1]$, and t a time point in T such that $t_0 < t < t_1$, the *right segmentation of s to t*, denoted by $s_{>t}$, is a segment over $]t,t_1]$, defined in every $t' \in]t,t_1]$ by

$$s_{>t}(t') \triangleq s(t') \tag{2.2}$$

□

With the help of definition 7 and 8, we can now concisely formulate a first requirement for a subset of an input segment set to be an allowable input segment set, in that it must be *closed* under left and right segmentation of segments. Any left or right segmentation of a segment in the set, must yield another segment in the set.

A second property involves the notion of *time invariance* of a system. Stated informally here, a system is called time invariant if its reaction to an input segment is independent of the interval over which the input segment is defined. In that respect, we desire that the *translation* of an allowable input segment to *any* time point in the time set, yields another allowable input segment. We define the translation of a segment to a time point as follows,

Definition 9 (Translation)
For $\mathscr{S}(T,X)$ a segment set, $s \in \mathscr{S}(T,X)$ a segment over an interval $]t_0,t_1]$, and t a time point in T, the *translation of s to t*, denoted by $s_{\rightarrow t}$, is a segment over $]t, t+(t_1-t_0)]$, defined in every $t' \in \,]t, t+(t_1-t_0)]$ by

$$s_{\rightarrow t}(t') \triangleq s(t_0 + (t'-t)) \tag{2.3}$$

□

A translation of a segment thus comes down to a *shift* of its domain over some amount of time units. With definition 9 in mind, we state a second requirement for a subset of an input segment set to be an allowable input segment set, in that it must be *closed* under translation of segments. This implies that we can make abstraction of the precise interval over which a segment is defined.

A final desirable property requires the concept of *right addition* of segments. In particular, when both ω and ω' are allowable input segments, and when ω' is *contiguous* to ω - that is, when the begin point of the domain of ω' coincides with the end point of the domain of ω -, then it is only reasonable to expect that the new segment formed by 'continuing' ω with ω' must also be an allowable input segment. We write the continuation of ω with ω' as $\omega\omega'$ and call $\omega\omega'$ the *right addition* of ω' to ω. Formally,

Definition 10 (Right addition)
For $\mathscr{S}(T,X)$ a segment set, and $s, s' \in \mathscr{S}(T,X)$ segments over $]t_0,t_1]$ and $]t_1,t_2]$ respectively, the *right addition of s' to s*, denoted by ss', is a segment over $]t_0,t_2]$, defined in every $t \in \,]t_0,t_2]$ by

$$ss'(t) \triangleq \begin{cases} s(t) & t \in \,]t_0,t_1] \\ s'(t) & t \in \,]t_1,t_2] \end{cases} \tag{2.4}$$

□

For a subset of an input segment set to be an allowable input segment set, we require that it is *closed* under right addition of segments. Summarising the above desirable properties, yields the following definition of an allowable input segment set,

Definition 11 (Allowable input segment set)
For $\mathscr{S}(T,X)$ an input segment set, a subset Ω of $\mathscr{S}(T,X)$ is called an *allowable input segment set* iff. it is closed under left and right segmentation, translation and right addition of segments.

□

2.2 Abstract objects and abstract systems

The *abstract object* construct that we cover in the following, is a *primitive* construct in general systems theory, that we like to use to model *all* that can *possibly* be known from a real system - see e.g. [30].[4] We will define the notion of an *abstract object trace* as all that we *do* know of a real system.

For Ω an allowable input segment set, and $\mathscr{S}(T,Y)$ an output segment set, we denote a generic input/output segment pair by (ω,θ), with $\omega \in \Omega$ and $\theta \in \mathscr{S}(T,Y)$. We whish to agree here once and for all that $\mathrm{dom}(\omega) = \mathrm{dom}(\theta)$ for any input/output segment pair (ω,θ). Imagine now in light of the aforementioned coin striking process, that we let an experimenter determine when coin planchets are added to the buffer of the process over some appropriate interval of time. Assume that, over this interval, we simply record the number of coins in the buffer as well as the status (busy/idle) of the striking machine. In other words, the experimenter applies some allowable input segment ω - scheduling arrivals -,[5] while we observe the returned output segment θ - measuring queue length and machine status. What factors will determine the output segment that we observe? Clearly, the output segment depends on the input segment that the experimenter applies. However, we expect that it will also depend on the initial *conditions* in the striking system - as there are the initial number of coins in the buffer, and the fact whether the striking machine is initially idle or busy. Naturally, the output segment that we observe also depends on the very way that the real coin striking process works. What matters here, is that for the interval of experimentation, there is a particular *universe* of input/output segment pairs that characterises the family of all *possible* experimental outcomes that can be obtained by letting the experimenter choose an arbitrary allowable input segment, and by applying this segment to the process, initialised in some arbitrary initial 'state'. We call such a universe an *input/output segment relation.* Formally,

Definition 12 (Input/output segment relation)
For T a time set, X and Y an input and an output set, $\Omega \subseteq \mathscr{S}(T,X)$ an allowable input segment set, $]t_0,t_1] \subset T$ such that $t_1 > t_0$, and $A_{]t_0,t_1]} \triangleq \{\omega \mid \mathrm{dom}(\omega) =]t_0,t_1]\}_{\omega\in\Omega}$ and $B_{]t_0,t_1]} \triangleq \{\theta \mid \mathrm{dom}(\theta) =]t_0,t_1]\}_{\theta\in\mathscr{S}(T,Y)}$, a relation between $A_{]t_0,t_1]}$ and $B_{]t_0,t_1]}$, denoted by $R_{]t_0,t_1]}$, is called an *input/output* or *I/O segment relation.*

$\square$

Given that an input/output segment pair (ω,θ) belongs to an input/output segment relation $R_{]t_0,t_1]}$, it is only natural to ask whether we should expect the pair $(\omega_{\to t},\theta_{\to t})$ with translations of ω and θ to belong to the input/output segment relation $R_{]t,t+(t_1-t_0)]}$, and this for all $t \in T$. In the following, we let this issue be answered in the *positive.* In that respect, we define the notion of a *time invariant* family of I/O segment relations as follows,

[4]In that respect, real system behaviour must be derivable from an abstract object.

[5]We thus assume that the arrival trajectory of coin planchets, composed by the experimenter, is *allowable.* In particular, we require here that it consists of *countably* many arrivals.

Definition 13 (Time invariant family of I/O segment relations)
For T a time set, X and Y an input and an output set, $\Omega \subseteq \mathscr{S}(T,X)$ an allowable input segment set, $\mathcal{I} \triangleq \{]t_0,t_1] \mid t_1 > t_0\}_{t_0,t_1 \in T}$, and $R_{]t_0,t_1]} \subset \Omega \times \mathscr{S}(T,Y)$ an input/output segment relation for every $]t_0,t_1] \in \mathcal{I}$, the family $\{R_{]t_0,t_1]}\}_{]t_0,t_1] \in \mathcal{I}}$ is called a *time invariant family of I/O segment relations* iff. it holds for all $]t_0,t_1] \in \mathcal{I}$ that

$$(\omega,\theta) \in R_{]t_0,t_1]} \Rightarrow \forall t \in T : (\omega_{\to t}, \theta_{\to t}) \in R_{]t,t+(t_1-t_0)]} \tag{2.5}$$

□

The fact that an allowable input segment set is closed under left and right segmentation, raises the issue whether, in case an input/output segment pair (ω,θ) belongs to an input/output segment relation $R_{]t_0,t_1]}$, we should expect that the input/output segment pair $(\omega_{t]}, \theta_{t]})$ and the input/output segment pair $(\omega_{>t}, \theta_{>t})$ with left and right segmentations of ω and θ belong to the input/output segment relations $R_{]t_0,t]}$ and $R_{]t,t_1]}$ respectively, and this all $t \in]t_0,t_1[$. Again, we let this issue be answered in the *positive*. For, we can split up our experiment, and consider the experimental outcomes in the first and the subsequent second experiment as I/O segment pairs in (different) I/O segment relations.

Since an allowable input segment set is closed under right addition, it is only natural to ask whether we should expect that for every input/output segment pair (ω,θ) in an input/output segment relation $R_{]t_0,t_1]}$, there is for every $t_2 \in T$ such that $t_2 > t_1$ at least one input/output segment pair (ω',θ') in the input/output segment relation $R_{]t_1,t_2]}$, for which holds that the input/output segment pair $(\omega\omega', \theta\theta')$ with the right additions of ω' to ω and θ' to θ is an input/output segment pair in the input/output segment relation $R_{]t_0,t_2]}$. In other words, if we obtained (ω,θ) in an experiment, where $\mathrm{dom}(\omega) \triangleq]t_0,t_1]$, should we then expect that it is possible to find for every $t_2 \in T$ such that $t_2 > t_1$ at least one pair (ω',θ') in $R_{]t_1,t_2]}$ such that $(\omega\omega',\theta\theta') \in R_{]t_0,t_2]}$? Again, we let this question be answered in the *positive*. In that respect, we define a *consistent* family of I/O segment relations as follows,

Definition 14 (Consistent family of I/O segment relations)
For T a time set, X and Y an input and an output set, $\Omega \subseteq \mathscr{S}(T,X)$ an allowable input segment set, $\mathcal{I} \triangleq \{]t_0,t_1] \mid t_1 > t_0\}_{t_0,t_1 \in T}$, and $R_{]t_0,t_1]} \subset \Omega \times \mathscr{S}(T,Y)$ an input/output segment relation for every $]t_0,t_1] \in \mathcal{I}$, the family $\{R_{]t_0,t_1]}\}_{]t_0,t_1] \in \mathcal{I}}$ is called a *consistent family of I/O segment relations* iff. it holds for all $]t_0,t_1] \in \mathcal{I}$ that

$$(\omega,\theta) \in R_{]t_0,t_1]} \Rightarrow \begin{cases} \forall t \in]t_0,t_1[: (\omega_{t]}, \theta_{t]}) \in R_{]t_0,t]} \text{ and } (\omega_{>t}, \theta_{>t}) \in R_{]t,t_1]} \\ \forall t_2 \in T \text{ s.t. } t_2 > t_1 : \exists (\omega',\theta') \in R_{]t_1,t_2]} : (\omega\omega', \theta\theta') \in R_{]t_0,t_2]} \end{cases} \tag{2.6}$$

□

With the help of definitions 13 and 14, we are now ready to give a concise definition of an *abstract object*. The definition that we provide here is similar in nature, though somewhat stricter than a definition given by [30].[6]

[6]In particular, the definition requires that the family of I/O segment relations that makes up an abstract

Definition 15 (Abstract object)
For T a time set, X and Y an input and an output set, $\Omega \subseteq \mathscr{S}(T,X)$ an allowable input segment set, $\mathcal{I} \triangleq \{]t_0, t_1] \mid t_1 > t_0\}_{t_0,t_1 \in T}$, and $R_{]t_0,t_1]} \subset \Omega \times \mathscr{S}(T,Y)$ an input/output segment relation for every $]t_0,t_1] \in \mathcal{I}$, the family $\{R_{]t_0,t_1]}\}_{]t_0,t_1] \in \mathcal{I}}$ is called an *abstract object* or *AO* iff. it is a time invariant and consistent family of I/O segment relations.

□

An abstract object characterises *all* that can *possibly* be known from a real system. It embodies a *black-box* view on a real system. That is, it tells us for every item that can go into the box - for every allowable input segment -, which items can come out of the box - the output segments to which the input segment is related by an I/O segment relation. We conclude our discussion of abstract objects by introducing the notion of an *abstract object trace* as follows,

Definition 16 (Abstract object trace)
A family of subsets of the I/O segment relations of an abstract object $\mathcal{A}$ is called an *abstract object trace* or a *trace of* $\mathcal{A}$.

□

In general, for $\mathcal{A}$ an abstract object, we denote a trace of $\mathcal{A}$ by $\mathcal{A}_{|.}$. We use the concept of an abstract object trace to formally express what we *do* know of a real system. A limited time frame, and experimental constraints imply that we cannot expect to have a fully dressed abstract object of a real system at our disposal. Instead, all that we can hope for is a number of observed input/output segment pairs. We organise these pairs in an abstract object trace. With some of the input/output segment pairs of an abstract object *trace*, we intend to perform particular *time based* arithmetic, in order to derive values for all independent and dependent variables of interest, and to construct real system behaviour.

An intuitive interpretation of state variables proceeds by assuming that we have an abstract object available. In effect, state variables can be seen as those variables that are required, once the object is known, to *label* every I/O segment pair of the object that carries a common allowable input segment ω, with a *unique* combination of values for the variables - unique within the family of I/O segment pairs that have ω at their first coordinate -, and this for every allowable input segment ω. Such a label is referred to as a *state* of the abstract object. Much similar to the definition of an input and an output set, we define a *state set* as follows,

Definition 17 (State set)
For σ_k, $k \in K$ state variables, indexed by a finite index set K, and Σ_k the range of state variable σ_k for all $k \in K$, the Cartesian product $\Sigma \triangleq \prod_{k \in K} \Sigma_k$ is called a *state set.*

□

Members of a state set are called *states.* Just like with input and output variables, the state behaviour through time is captured by a *trajectory.* Once we have pinned down a number

object must be a *time invariant* family.

of state variables, some states may not be very meaningful to describe the conditions in an abstract system. We call those states *irregular* states. In short, an irregular state is a state 1) that we will never want to choose as a starting state in an experiment with an abstract system, *and* 2) that can never be *reached* from a non-irregular state. Where the former is under our direct control, the latter requires us to see the notion of *state transitions.* What matters here is that an abstract system is always in a non-irregular state, *either* because we have put it in such a state, *or* because the system has reached this state from another non-irregular state when it processed an input segment. In the following, we refer to non-irregular states as *initial states*, and we assemble all initial states in an *initial state set.* Practically, given a particular state set, we will derive an *initial* state set by imposing a family of *constraints* on the state variables. Therefore, we define an initial state set formally as follows,

Definition 18 (Initial state set)
For Σ a state set, and $\mathcal{C}$ a family of constraints on the state variables that make up Σ, an *initial state set*, denoted by Σ_0, is a subset of Σ that holds every state satisfying all of the constraints in $\mathcal{C}$.

□

We can rearrange the information that is contained in an abstract object, by defining a function for every initial state of the object, that assigns to every allowable input segment *the* output segment that the object will return when it is in this particular state. Such a function is termed an *input/output segment function.* In view of our definition of an abstract object, we define an input/output segment function as follows,

Definition 19 (Input/output segment function)
For T a time set, X and Y an input and an output set, $\Omega \subseteq \mathscr{S}(T,X)$ an allowable input segment set, $\mathcal{A}$ an abstract object with an I/O segment relation $R_{]t_0,t_1]} \subset \Omega \times \mathscr{S}(T,Y)$ for every $]t_0,t_1] \subset T$ s.t. $t_1 > t_0$, and σ an initial state from an initial state set Σ_0 for $\mathcal{A}$, a function from Ω into $\mathscr{S}(T,Y)$, denoted by ι_σ, is called an *input/output* or *I/O segment function in* σ iff. it holds for all ω in Ω that

$$\iota_\sigma(\omega) = \theta \Rightarrow \begin{cases} \omega[R_{\mathrm{dom}(\omega)}]\theta \\ \iota_\sigma(\omega_{t]}) = \theta_{t]},\ \forall t \in \mathrm{dom}(\omega) \setminus \{\sup(\mathrm{dom}(\omega))\} \\ \iota_\sigma(\omega_{\rightarrow t}) = \theta_{\rightarrow t},\ \forall t \in T \end{cases} \tag{2.7}$$

□

The conditions in (2.7) comply with the intuitive notion of state, and are in agreement with the fact that an abstract object is a consistent, time invariant family of I/O segment relations. Another way to represent the information in an abstract object $\mathcal{A} \triangleq \{R_{]t_0,t_1]}\}_{]t_0,t_1] \subset T \text{ s.t. } t_1 > t_0}$ with allowable input segment set Ω, is then to give an input/output segment function for every state in an appropriate initial state set Σ_0 for $\mathcal{A}$, such that for all ω in Ω and σ in Σ_0, it holds that

$$\omega[R_{\text{dom}(\omega)}]\theta \Leftrightarrow \exists\sigma \in \Sigma_0 : \iota_\sigma(\omega) = \theta \tag{2.8}$$

The *destination* state that is reached from a particular initial state under an applied allowable input segment, is specified by a so-called *state transition function.* For Σ_0 an initial state set, and Ω an allowable input segment set, a state transition function is a function from $\Sigma_0 \times \Omega$ into Σ_0 that satisfies certain *consistency* conditions. A discussion of these conditions extends beyond the scope of this research report - see e.g. [30].

The connection between states and outputs is captured by a so-called *output function.* For Σ_0 an initial state set, and Y an output set, an output function, denoted by λ, is a function from Σ_0 into Y that assigns to every state in Σ_0 an output from the output set Y.

The following abstract system construct, which eventually comes down to the *system* construct of [31], integrates the concepts that were introduced before.

Definition 20 (Abstract system)
For T a time set, X and Y an input and an output set, $\Omega \subseteq \mathscr{S}(T, X)$ an allowable input segment set, Σ_0 an initial state set, δ a state transition function from $\Sigma_0 \times \Omega$ into Σ_0, and λ an output function from Σ_0 into Y, an *abstract system* or *AS* is a 7-tuple, defined by

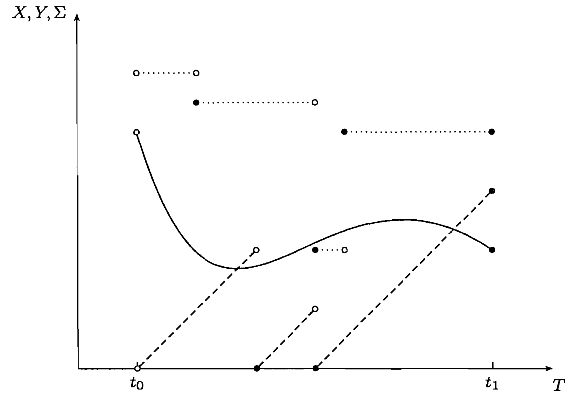
$$\mathcal{S} \triangleq \langle T, X, \Omega, Y, \Sigma_0, \delta, \lambda \rangle \tag{2.9}$$

□

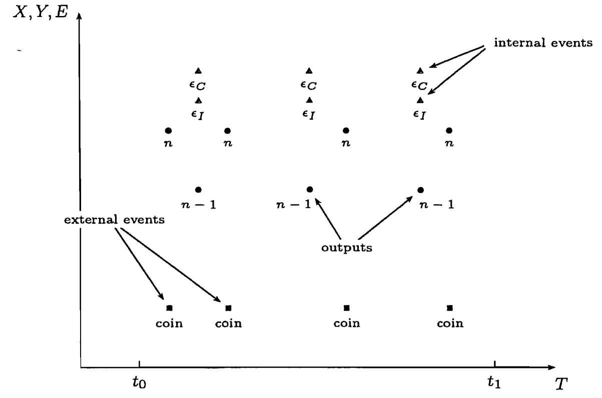
The specification of a *state transition function,* that should determine the destination state under *any* allowable input segment, prevents us from building abstract systems for problems of a realistic complexity. The *structured discrete event system* formalism that we develop in the following is precisely aimed to be able to construct formal models for the kind of systems and problem situations that we like to think of in light of our validation method. The abstract system construct will re-appear as we develop the stochastic system construct in section 3.

2.3 Events, timed sequences and event scheduling

Intuitively, a (structured) discrete event system is a system theoretic construct, in which *events* take place at *discrete* points in time. The term *structured* emphasises that a certain structure of components and component interactions is contained in the system specification. Unlike an abstract system, events are the driving force behind the advance of time in a structured discrete event system. We classify events in two categories: events that are predicted by the system specification and drive the system from one state to another, or *internal events,* and events that happen outside the system (and are hence unpredictable) but affect the state of the system, or *external events.* In view of the aforementioned coin striking process, a typical internal event could be the striking machine, taking an unfinished coin planchet from the buffer,



(a) Segments in an AS



(b) Timed sequences in an sDEVS

Figure 3: Segments and timed sequences

and commencing the striking process. A typical external event could be a new, unprocessed coin planchet being added to the buffer in front of the striking machine.

We call a sequence of time-event pairs, whereby every pair specifies an event and a time point of occurrence, a *timed event sequence* if and only if the time point of occurrence of the event in any time-event pair of the sequence is greater than or equals the time point of occurrence of events in previous time-event pairs in the sequence. The following definition of a *timed* *sequence* is somewhat more general, in that it does not explicitly require that the members of the sequence must be events,

> **Definition 21 (Timed sequence)**
> For T a time set, X an arbitrary non-empty set, and I a countable index set, a function from I into $T \times X$, denoted by $\dot{s}$, is called a *timed sequence* iff. for all $i_1, i_2 \in I$ s.t. $i_1 < i_2$ it holds that $\triangledown_T(\dot{s}(i_1)) \leq \triangledown_T(\dot{s}(i_2))$.
>
> □

Figure 3 highlights the difference between segments and timed sequences. Figure 3(a) displays three potential segments over an interval $]t_0, t_1]$. We used different line styles in the figure to differentiate the trajectories. The segments could be input, state or output segments of an abstract system. Figure 3(b) shows three timed sequences - albeit in a somewhat implicit way. We used different dot styles to differentiate the timed sequences. If we associate e.g. the squared dots in figure 3(b) with coin arrivals, then the timed sequence $\dot{\chi}$ that assigns to every index i in $\{1, 2, 3, 4\}$ a *pair* with a time point in $]t_0, t_1]$ at which a coin arrives, and the keyword *coin*, is a typical *timed external event sequence* for the coin striking process. Figure 3(b) contains another example of a timed sequence. Interpret therefore the lower and upper triangular shaped dots in the figure as $\epsilon_I \triangleq$ *striking process initiated* and $\epsilon_C \triangleq$ *striking process completed* respectively. According to the figure, the striking process is initiated on the interval $]t_0, t_1]$ at the same time that it is completed, indicating that the striking machine takes a new coin planchet from the buffer, everytime it finishes the current coin in process. The timed sequence $\dot{\epsilon}$ that assigns to every index i in $\{1, 2, \ldots, 6\}$ a pair with a time point in $]t_0, t_1]$ at which the striking process is completed or re-initiated, and the event ϵ_I or ϵ_C, is a typical example of a *timed internal event sequence.* Finally, figure 3(b) contains a third example of a timed sequence. Assume that the round shaped dots in the figure represent the number of coins in the buffer of the striking process. If we associate buffer content with system output, then the round shaped dots in the figure constitute a *timed output sequence* $\dot{v}$ that assigns to every index i in $\{1, 2, \ldots, 7\}$ a pair with a time point in $]t_0, t_1]$ at which the system dumps its output, and a natural number indicating the number of coins in the buffer at that point in time.

Given an initial state set Σ_0, a family of internal events E, and an extended time set $\bar{T}$,[7] we capture the time-event advance mechanism by a function τ from Σ_0 into $\bar{T} \times E$. For every initial state, this function specifies which internal event will happen next, and how many time units will have to elapse before this event occurs. We call such a function a *time-event advance function.* Formally,[8]

[7]By *extended time set*, we mean the time set T, united with the singleton-set that holds the supremum of T. In case $T \triangleq [0, +\infty[$, we thus have that $\bar{T} \triangleq [0, +\infty]$.

[8]The definition that we provide of a time-event advance function runs parallel to that of the *time-advance function* in the classic discrete event system formalism of [31].

Definition 22 (Time-event advance function)
For $\bar{T}$ an extended time set, E an internal event set, and Σ_0 an initial state set, a *time-event advance function*, denoted by τ, is a function from Σ_0 into $\bar{T} \times E$.

□

In discrete event systems, it is not uncommon that *multiple* internal events are *anticipated* to occur *simultaneously* in the future from the viewpoint of a particular initial state. In that respect, instead of directly specifying the value of a time-event advance function at an initial state, we whish to set up a list of *anticipated* time-event pairs. Then, we use the amount of time units that need to elapse before the event in the anticipated time-event pairs take place, together with a partial order relation that we install on the internal event set, to determine the next internal event, and the amount of time units that need to elapse before it will occur. First, let's introduce the notion of an *internal event scheduler* in the following definition,

Definition 23 (Internal event scheduler)
For $\bar{T}$ an extended time set, E an internal event set, and Σ_0 an initial state set, an *internal event scheduler* is a relation, denoted by Γ, between Σ_0 and $\bar{T} \times E$.

□

Let Γ be an internal event scheduler. For an arbitrary initial state $\sigma \in \Sigma_0$, *one or more* of the time-event pairs to which σ is related by Γ, will have at their first coordinate, an amount of time units that is the *smallest* amount of time units among all time-event pairs to which σ is related by Γ. Such time-event pairs carry *imminent* events. Formally, we define an imminent event as follows,

Definition 24 (Imminent event)
For Γ an internal event scheduler between Σ_0 and $\bar{T} \times E$, and σ an initial state of Σ_0, an event $\epsilon \in E$ is called an *imminent event in σ* iff.

$$\epsilon \text{ is imminent in } \sigma \Leftrightarrow \begin{cases} \exists s \in \bar{T} : \sigma[\Gamma](s,\epsilon) \\ \neg\exists[\epsilon' \in E,\ s' \in \bar{T} \text{ s.t. } s' < s] : \sigma[\Gamma](s',\epsilon') \end{cases} \tag{2.10}$$

□

We derive a time-event advance function τ from an internal event scheduler Γ, and a partial order relation $\preceq$ on the internal event set, as follows

$$\tau(\sigma) \triangleq (s,\epsilon) \Leftrightarrow \begin{cases} \sigma[\Gamma](s,\epsilon) \text{ and } \epsilon \text{ is imminent in } \sigma \\ \neg\exists[\epsilon' \in E \text{ s.t. } \epsilon' \prec \epsilon] : \sigma[\Gamma](s,\epsilon') \end{cases} \tag{2.11}$$

2.4 Components and local transitions

The state of an abstract system is described by a number of state variables in a state variable set. In a structured discrete event system, we partition a state variable set, attach a linguistic label to every member of the partition, and refer to such a label as a *component.* A component points to a set of state variables that we conceive as meaningfully belonging together. Typical examples of components that we think of are *queue* and *striking machine* in the context of the simple coin striking system. We call the state variables to which a component points *component state variables.* A typical state variable of a queue might indicate the number of items that it contains, while a striking machine could be described by the remaining time that it needs to finish the current coin in process. The Cartesian product of the range of state variables of a component constitutes a *component state set.* Some states in a component state set may carry incompatible values for the state variables. Therefore, we retain all meaningful states of a component in a *component initial state set.* In the following, we use the symbol C to denote a set of components, and we employ the notation Σ_0^c to denote the initial state set of component $c \in C$.

A logical follow-up of thinking of a system in terms of components and events, is that components are perceived to *influence* one another through events. To be more precise, for every component, we identify events that *significantly* affect the state of the component, and the *scheduling* of which depends on the state of other components. By *significant*, we mean that the state change of the component as a result of the event, is *not* limited to a simple reduction of the value of a countdown clock time variable - like a variable that measures the remaining striking time.[9] By an event, the *scheduling* of which depends on the state of other components, we mean that state variables of the latter are indispensable to anticipate the event, and/or to know how many time units need to elapse before the event is expected to take place. In a way, this kind of influence of a component on another component through an event embodies a notion of *causality.* In effect, the state of a component (the influencer) determines - possibly among other components - the scheduling of an event, which upon occurrence significantly affects the state of another component (the influencee). It is as if a state change of the latter, in response to the event, is *caused* by a state change of the former.[10] For this kind of influence that a component has on another component, we call the influencing component an *active influencer.*

Sometimes, to calculate the next state of a component under an event, we need the state of *another* component that does *not* influence the former in the sense that we explained above. The *random number generator component* that we encounter later in this report is a typical

[9]A *countdown clock time variable* is a state variable that carries the interpretation of measuring a *remaining* 'processing' time of a component. Usually - that is, at almost every time point that an event occurs -, a countdown clock time variable is simply reduced with the time that has elapsed since the event took place. Such a straight-forward reduction is a *trivial* or *insignificant* state change of a component.

[10]The idea is that a state change of the influencer suddenly results in the scheduling of an event, which upon occurrence affects the state of the influencee in a non-trivial or significant way.

example of such an influencing component that is not an active influencer. As we will point out, we let every state variable of a random number generator component hold a so-called *(pseudo) random number* - a positive real number on the unit interval. We use random numbers to *draw* samples of a population of e.g. different striking times. This allows us to model an apparently *random* striking time. In that case, a component *striking machine* requires the state of a component *random number generator* at the very moment that the striking process is initiated, and that a striking time must be sampled. In that way, the *next* state of the striking machine component depends on the *current* state of the random number generator component. Therefore, we say that the random number generator component is a *passive influencer* of the striking machine component.

To model active and passive influencers, we define a *relation* in a component set. A component c is related to a component $c' \neq c$ if and only if c is an active or a passive influencer of c'. We call such a relation a *causality-dependency relation.* We employ a causality-dependency relation to model the effect that *internal* events have on the state of a component with the help of a *local transition function.* A local transition function is defined for every component, and maps a triple with the current state of the component, the current state of all components that are related to the component by the causality-dependency relation, and a time-event pair, on the state that the component will travel to upon occurrence of the event in the time-event pair. Formally, we define a *local transition function* as follows,[11]

Definition 25 (Local transition function)
For $\bar{T}$ an extended time set, E an internal event set, Σ_0 an initial state set, τ a time-event advance function from Σ_0 into $\bar{T} \times E$, $C \triangleq \{c_i\}_{i \in I}$ a component set, indexed by a finite index set I, $\Sigma_0^{c_i}$, $i \in I$ initial state sets of the components in C such that $\Sigma_0 \subseteq \prod_{i \in I} \Sigma_0^{c_i}$, A a causality-dependency relation in C, and c a component in C, a function from a subset of $\Sigma_0^c \times (\prod_{j \in I \text{ s.t. } c_j [A] c} \Sigma_0^{c_j}) \times (\bar{T} \times E)$ into Σ_0^c, denoted by δ_ϕ^c, that is defined in a point $(\sigma^c, \sigma^{[A]c}, (s, \epsilon))$, with $\sigma^{[A]c} \triangleq (\sigma^{c_{j_1}}, \sigma^{c_{j_2}}, \ldots, \sigma^{c_{j_n}})$ if and only if there exists a state $\sigma \in \Sigma_0$ such that $\sigma^c = \nabla_{\Sigma_0^c}(\sigma)$, $\sigma^{c_j} = \nabla_{\Sigma_0^{c_j}}(\sigma)$ for every $j \triangleq j_1, j_2, \ldots, j_n$ and $\tau(\sigma) = (s, \epsilon)$, is called a *local transition function in c.*

$\square$

The idea of local transition functions comes down to the following. Everything as in definition 25, let σ be an initial state of the initial state set Σ_0 of a structured discrete event system. The state σ pins down an initial state for every component c_i, $\forall i \in I$. The next internal event that will take place, and the amount of time units that will elapse before it occurs, are given by the image of σ under the time-event advance function τ. Denote this time-event pair by $\tau(\sigma) \triangleq (s, \epsilon)$. In order to calculate the next state of the system when ϵ takes place, we can proceed in two ways. *Either* we immediately define the *global* state that the system reaches from the current state, once the event ϵ is processed after s time units, *or* we calculate the next

[11]In the definition, we adopt the convention that $\Sigma_0 \triangleq \Sigma_0^{c_1} \times \Sigma_0^{c_2} \triangleq A \times B \times C$ in case $C \triangleq \{c_1, c_2\}$ and $\Sigma_0^{c_1} \triangleq A \times B$ and $\Sigma_0^{c_2} \triangleq C$. Thus, in case (a, b) is a state of c_1 and c is a state of c_2, then we denote an initial state of Σ_0 by (a, b, c) and *not* by $((a, b), c)$.

local states of the system first under the time-event pair - we calculate the states of all components individually -, and then assemble all these local states in a global state. By specifying local transition functions, we follow the latter approach. Clearly, in order to compute the next (local) state of a component, we must know its current state, the state of all its influencers and the time-event pair that takes place. Therefore, a typical member of the domain of the local transition function of a component c is of the form $(\sigma^c, \sigma^{[A]c}, (s, \epsilon))$, where σ^c is the current state of the component, $\sigma^{[A]c}$ is the current state of all the influencers of the component, and (s, ϵ) is a time-event pair that can take place when the system is in one of the (global) states that specifies the state of c and its influencers to be σ^c and $\sigma^{[A]c}$.

2.5 Total states and external state transitions

In a (structured) discrete event system, a change of state occurs only when an internal or an external event takes place. Since nothing notable happens in between a pair of subsequent events, the state in between two state transitions remains fixed. In order to be able to correctly model the effect that an *external* event has on the current state of a structured discrete event system, we must however be aware of the amount of time that the system has resided in its current state. Therefore, a *system time variable* e is used in the classic discrete event system formalism, that keeps track of the time that the system has been in its current state. With the help of a system time variable, the concept of a *total initial state set* can be defined as follows,

Definition 26 (Total initial state set)
For $\bar{T}$ an extended time set, Σ_0 an initial state set, E an internal event set, and τ a time-event advance function from Σ_0 into $\bar{T} \times E$, a *total initial state set*, denoted by $\bar{\Sigma}_0$, is defined by

$$\bar{\Sigma}_0 \triangleq \{(\sigma, e) \mid e < \triangledown_{\bar{T}}(\tau(\sigma))\}_{\sigma \in \Sigma_0,\, e \in \bar{T}} \tag{2.12}$$

□

Members of a total initial state set are called *total initial states.* In the following, we use the notation $\bar{\sigma}$ to denote an arbitrary total initial state. Thus, very simply, in case a structured discrete event system is said to be in a total initial state $\bar{\sigma} \triangleq (\sigma, e)$ at a certain time point t, then it has been in state σ at time point t for e time units. Using the concept of a total initial state set, we define a *transitory state set* as follows,

Definition 27 (Transitory state set)
For $\bar{T}$ an extended time set, Σ_0 an initial state set, and $\bar{\Sigma}_0 \subset \Sigma_0 \times \bar{T}$ a total initial state set, a *transitory state set*, denoted by $\Sigma_{\rightharpoondown}$, is defined by

$$\Sigma_{\rightharpoondown} \triangleq \{\sigma \mid \neg\exists e \in \bar{T} : (\sigma, e) \in \bar{\Sigma}_0\}_{\sigma \in \Sigma_0} \tag{2.13}$$

□

Members of a transitory state set are called *transitory states.* Combining (2.12) and (2.13), it follows that a transitory state must be a state in which the next internal event is bounded to occur *immediately.* A (structured) discrete event system will hence only *pass* through its transitory states. Once all events at a time point have been processed, it always settles in a *non-transitory* state.

Using the concepts of total initial states and transitory states, we now define a pair of final discrete event system concepts, i.e. an *external state transition function* and a *discrete event output function.* For $\bar{T}$ an extended time set, Σ_0 an initial state set, $\bar{\Sigma}_0 \subset \Sigma_0 \times \bar{T}$ a total initial state set, and X an external event set, a function from $\bar{\Sigma}_0 \times X$ into Σ_0, denoted by δ_X, is called an *external state transition function.* An external state transition function specifies, for every pair with a total initial state of a (structured) discrete event system and an external event, the next state of the system. By convention, the system time variable e is set at 0 when an external and/or an internal event takes place. For $\bar{T}$ an extended time set, Σ_0 an initial state set, $\bar{\Sigma}_0 \subset \Sigma_0 \times \bar{T}$ a total initial state set, and Y an output set, a function from $\Sigma_0 \setminus \Sigma_\rightarrow$ into Y, denoted by λ, is called a *discrete event output function.* A discrete event output function is the discrete event like version of the output function of an abstract system. Here, we explicitly state that output is only defined for those states that are non-transitory states.

2.6 The structured discrete event system construct

In the following definition, we formally lay down the *structured discrete event system* construct that we intend to employ to build models for the problem situations that we think of in light of our validation method.

Definition 28 (Structured discrete event system)
For $\bar{T}$ an extended time set, X an external event set, $(E, \preceq)$ a partially ordered internal event set, Y an output set, $C \triangleq \{c_i\}_{i\in I}$ a component set, indexed by a finite index set I, $\Sigma_0^{c_i}$, $i \in I$ initial state sets for components in C, $\Sigma_0 \subseteq \prod_{i\in I} \Sigma_0^{c_i}$ an initial state set, Γ an internal event scheduler between Σ_0 and $\bar{T} \times E$, A a causality-dependency relation in C, δ_ϕ^c, $\forall c \in C$ a local transition function in c, δ_X an external state transition function from $\bar{\Sigma}_0 \times X$ into Σ_0, and λ a discrete event output function from $\Sigma_0 \setminus \Sigma_\rightarrow$ into Y, a *structured discrete event system* or *sDEVS* is a 12-tuple, defined by

$$\mathcal{Q} \triangleq \langle \bar{T}, X, (E, \preceq), Y, C, \{\Sigma_0^c\}_{c\in C}, \Sigma_0, A, \{\delta_\phi^c\}_{c\in C}, \delta_X, \Gamma, \lambda\rangle \tag{2.14}$$

□

For $\mathcal{Q}$ the structured discrete event system of definition 28, we call *(global) internal state transition function* of $\mathcal{Q}$, denoted by δ_ϕ, the function from a subset of $\Sigma_0 \times (\bar{T} \times E)$ into Σ_0, that is defined in $(\sigma, \tau(\sigma))$ for every $\sigma \in \Sigma_0$, by

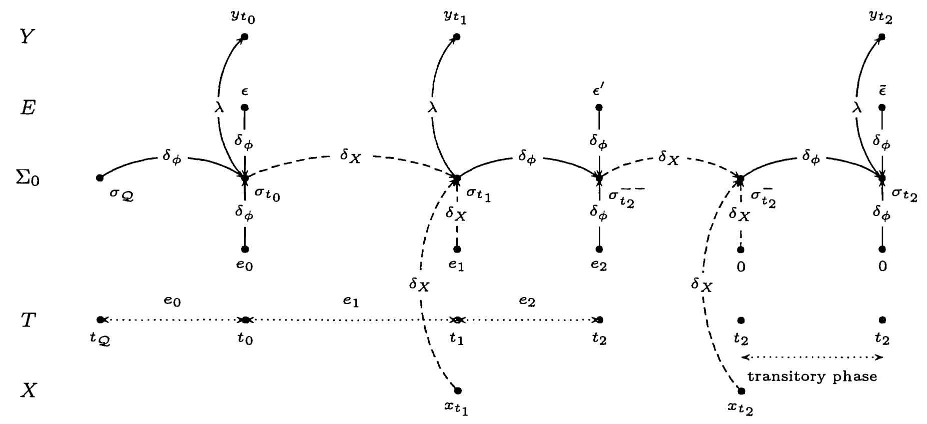


Figure 4: State transitions in a structured discrete event system

$$\delta_\phi(\sigma,\tau(\sigma)) \triangleq \underbrace{(\delta_\phi^{c_{i_1}}(\sigma^{c_{i_1}},\tau(\sigma)), \delta_\phi^{c_{i_2}}(\sigma^{c_{i_2}},\tau(\sigma)),\dots,\delta_\phi^{c_{i_n}}(\sigma^{c_{i_n}},\tau(\sigma)))}_{\text{organised in a } \sum_{i\in I}\dim(\Sigma_0^{c_i}) \text{ tuple}} \qquad (2.15)$$

with $\sigma^{c_i} \triangleq \nabla_{\Sigma_0^{c_i}}(\sigma)$ for every $i \in I$, and with $I \triangleq \{1,2,\dots,n\}$. As we pointed out in (2.15), we organise the RHS of (2.15) such that the image of $(\sigma,\tau(\sigma))$ under δ_ϕ forms an initial state in Σ_0. Thus, very simply, a (global) internal state transition of a structured discrete event system is modelled by triggering *every* local transition function of the system individually, and then organising the outcome of these triggering operations in an initial state of the structured discrete event system.

In figure 4, we put the essential items of a structured discrete event system Q in perspective. We illustrate in the figure how a structured discrete event system travels from an initial state σ_Q to a (destination) state σ_{t_2}, under a timed external event sequence $\dot{\chi} \triangleq (t_1, x_{t_1}) \sim (t_2, x_{t_2})$. At time point t_Q, the system is assumed to be in a *total* initial state $\bar{\sigma}_{t_Q} \triangleq (\sigma_Q, 0)$. We now use the time-event advance function τ, that is induced by the internal event scheduler Γ and the partial order relation $\preceq$ on E, to derive the next internal event, and the remaining time until this event will occur. The time-event pair with the first imminent event that needs to be processed equals $\tau(\sigma_Q) \triangleq (e_0, \epsilon)$. In the figure, the event ϵ is scheduled to occur at time point t_0. Since no external events happen from time point t_Q until and including time point t_0, the state of the system remains σ_Q at every time point in the interval $[t_Q, t_0[$. The system time variable e steadily increases however. The *total* state of the discrete event system changes thus *continuously* on the interval $[t_Q, t_0[$. At time point t_0, the internal state transition function δ_ϕ of (2.15) is used to derive the next state of the system. The next state equals $\sigma_{t_0} \triangleq \delta_\phi(\sigma_Q, (e_0, \epsilon))$.

What is now *the* state of the system at time point t_0? We define this state to be the state that the system eventually reaches, after having processed *all* events, no matter whether they are internal or external events. Since there is no external event at time point t_0, and in the assumption that σ_{t_0} is a non-transitory state, *the* state of $\mathcal{Q}$ at t_0 equals thus σ_{t_0}.

We let the structured discrete event system $\mathcal{Q}$ produce output at time point t_0, in response to the fact that a state transition has occurred. In contrast with the classic discrete event system formalism, we whish to agree here once and for all to apply the *discrete event output function* λ of $\mathcal{Q}$ every time that an event - *no matter* whether it is an internal or an external event - takes place, that differs from the *no-internal-event* and the *no-external-event.* As we will indicate in the example specifications that we develop later in this report, we use the *no-internal-event* to let time advance in a structured discrete event system in case no significant internal event is expected to take place in the future. Sometimes, we apply the *no-external-event* to a structured discrete event system as the result of the fact that the *no-output* of another system is offered as an external event to the former system. To be more precise, there may not always be a meaningful output to be reported when a state transition occurs in a system that produces external events for another system - which makes that we will define the output in that case to be the *no-output.* Assuming that the event at time point t_0 is not the no-internal-event, the system produces an output $y_{t_0} \triangleq \lambda(\sigma_{t_0})$.

After the state transition at time point t_0, the system time variable e is reset at 0. The *total* state of $\mathcal{Q}$ at time point t_0 hence equals $(\sigma_{t_0}, 0)$. Now, we employ again the time-event advance function τ to derive the next internal event, and the time until this event will occur. Assume that $\tau(\sigma_{t_0}) \triangleq (e_1 + e_2, \epsilon')$. As indicated by the figure, an external event x_{t_1} occurs *before* the internal event ϵ' takes place. In case x_{t_1} is the no-external-event, we leave the total state of the system untouched. In case x_{t_1} differs from the no-external-event, we use the *total* state (σ_{t_0}, e_1) to look up the state that is assigned to the pair $((\sigma_{t_0}, e_1), x_{t_1})$ by the external state transition function δ_X. The system hence travels to $\sigma_{t_1} \triangleq \delta_X((\sigma_{t_0}, e_1), x_{t_1})$ at time point t_1. We reset the system time variable e at 0. What happens once the system has settled in σ_{t_1}? Then, we re-employ the time-event advance map τ to figure out when the next internal event will take place. The time-event pair with the next internal event equals $\tau(\sigma_{t_1})$. In the figure, we assumed that the internal event ϵ' (that was expected to be the first internal event when the system had moved to state σ_{t_0}), is still the next internal event that will take place, as seen from state σ_{t_1}. Thus, $\tau(\sigma_{t_1}) \triangleq (e_2, \epsilon')$. Since all events at time point t_1 have been processed, we use the discrete event output function λ to derive the output at t_1 - in the assumption here that x_{t_1} differs from the no-external-event. Since *the* state of the structured discrete event system at t_1 equals σ_{t_1}, the output of the system at this point is given by $y_{t_1} \triangleq \lambda(\sigma_{t_1})$.

The situation in figure 4 at time point t_2 is a little more complicated. As we illustrated in the figure, *both* an internal *and* an external event are bound to happen at time point t_2. How should we tie-break these simultaneous events? We whish to agree here once and for all that we first handle the internal event, after which we deal with the external event. Possibly, once this external event is processed, we may have to apply the internal state transition function again to deal with an internal event, that is scheduled to happen immediately as a follow-up of the external state transition. Let's clarify this point somewhat further. At time point t_2, we first apply the internal state transition function δ_ϕ to derive the next state $\sigma_{t_2}^{--} \triangleq \delta_\phi(\sigma_{t_1}, (e_2, \epsilon'))$. Then, we use the time-event advance function τ to determine when the next internal event takes place. In the figure, we assumed that the next internal event is anticipated to take place at some point *beyond* t_2. Thus, the state $\sigma_{t_2}^{--}$ is the state to which the system travels to, after having processed all *internal* events so far. Since there is an external event x_{t_2} at time point t_2, we now apply the external state transition function δ_X to the pair $((\sigma_{t_2}^{--}, 0), x_{t_2})$. The system hence immediately moves to $\sigma_{t_2}^{-} \triangleq \delta_X((\sigma_{t_2}^{--}, 0), x_{t_2})$. But then, since the system does not reside in $\sigma_{t_2}^{--}$ for a strictly positive amount of time, isn't $\sigma_{t_2}^{--}$ a *transitory* state? In view of definition 27, the answer is *no*. In effect, if we did not apply the external event x_{t_2}, then the system would have remained in the state $\sigma_{t_2}^{--}$ for some strictly positive amount of time units, since we assumed that $\nabla_{\bar{T}}(\tau(\sigma_{t_2}^{--})) > 0$. Therefore, $\sigma_{t_2}^{--}$ is a non-transitory state. In state $\sigma_{t_2}^{-}$, we apply the time-event advance function to derive the imminent time-event pair with the next internal event. According to the figure, this pair equals $\tau(\sigma_{t_2}^{-}) \triangleq (0, \bar{\epsilon})$. Once the system is in state $\sigma_{t_2}^{-}$, the next internal event thus takes place *immediately*. For that reason, $\sigma_{t_2}^{-}$ *is* a transitory state. As usual, we apply the internal state transition function to derive the next state $\sigma_{t_2} \triangleq \delta_\phi(\sigma_{t_2}^{-}, (0, \bar{\epsilon}))$. In the figure, we assumed that the next internal event from the viewpoint of state σ_{t_2}, is expected to occur at a time point *beyond* t_2. In that respect, once the system enters the state σ_{t_2}, all events have been handled, and the discrete event output function λ is employed to derive the system's output at time point t_2. Since *the* state of the structured discrete event system at time point t_2 equals σ_{t_2}, the output of the system equals $y_{t_2} \triangleq \lambda(\sigma_{t_2})$ at time point t_2. Concluding our discussion of state transitions in a structured discrete event system, observe from the figure that the system, that was started in the total initial state $(\sigma_Q, 0)$, has produced the timed output sequence $\dot{\upsilon} \triangleq (t_0, y_{t_0}) \sim (t_1, y_{t_1}) \sim (t_2, y_{t_2})$, in response to our timed external event sequence $\dot{\chi} \triangleq (t_1, x_{t_1}) \sim (t_2, x_{t_2})$.

3 Stochastic systems

In this section, we lay down the *operands* of our validation method, and state the *conditions* that must be fulfilled for the method to be applicable. Therefore, we develop a *stochastic system* construct. Roughly said, a stochastic system embraces a family of structured discrete event systems and a family of stochastic processes. In subsection 3.1, we introduce some basic stochastic process terminology. We assume that the reader is familiar with basic *measure theoretic* notions of random variables and stochastic processes. For an extensive treatment of stochastic processes, we refer to [23, 24, 10, 7]. For a profound coverage of measure theoretic aspects of probability theory, we refer to [8, 27, 11]. Then, we introduce in subsection 3.2 the notion of random numbers and random number generators, and explain how random numbers and a random number generator can be included in the structured discrete event system formalism that we proposed in section 2. The inclusion of random numbers in the structured discrete event system formalism is a formal development of the idea to model apparent randomness in a deterministic way in a system specification, as it was originally proposed by [31]. Further, we use the concept of random numbers to formally define the notion of *replications* in subsection 3.3, and integrate replications with the structured discrete event system construct in subsection 3.4. We also develop a pair of examples of structured discrete event systems in subsection 3.4. We define statistics of interest of stochastic processes in subsection 3.5. Finally, we conclude this section by defining a stochastic system construct in subsection 3.6 and by laying down the concept of *behaviour* of a stochastic system.

3.1 Stochastic processes

For Ω an arbitrary non-empty set, a *σ-field on Ω* is defined as a class of subsets of Ω, that contains Ω, and that is closed under *complementation* and *countable* union. Formally,

Definition 29 (σ-field)
For Ω an arbitrary non-empty set, a class $\mathcal{F}$ of subsets of Ω is called a *σ-field (on Ω)* iff. 1) $\Omega \in \mathcal{F}$, 2) $A \in \mathcal{F} \Rightarrow A^c \in \mathcal{F}$ for all $A \in \mathcal{F}$, and 3) $A_i \in \mathcal{F},\ i \in I \Rightarrow \bigcup_{i\in I} A_i \in \mathcal{F}$ for every countable index set I.

□

For Ω an arbitrary non-empty set, members of a σ-field on Ω are called *measurable sets in* Ω. For $\mathcal{F}$ a σ-field on Ω, the pair $(\Omega, \mathcal{F})$ is called a *measurable space.* Measurable spaces and *probability measures* are concepts that we require to formally define the notion of a random variable. In the following definition, we formally pin down the notion of a probability measure.

Definition 30 (Probability measure)
For $(\Omega, \mathcal{F})$ a measurable space, a function P from $\mathcal{F}$ into $[0,1]$ is called a *probability measure on* $\mathcal{F}$ if and only if 1) $P(\Omega) = 1$, 2) $P(A) = \alpha \Rightarrow P(A^c) = 1 - \alpha$ for all A in $\mathcal{F}$, and 3) $\sum_{i \in I} P(A_i) = P(\bigcup_{i \in I} A_i)$ for every countable index set I and mutually disjunct measurable sets A_i, $i \in I$ in Ω.

□

Keeping the notation of definition 30, the triple $(\Omega, \mathcal{F}, P)$ is called a *probability space.* The set Ω is referred to as the *sample set* of the probability space, while measurable sets in Ω are called *events.* An experiment on a probability space involves taking a sample of Ω. Every event that contains the sample of Ω that is obtained in the experiment, is said *to have occurred.* A *random variable* can now formally be defined as follows,

Definition 31 (Random variable)
For $(\Omega, \mathcal{F})$ and $(X, \mathcal{S})$ measurable spaces, and P a probability measure on $\mathcal{F}$, an $\mathcal{F} - \mathcal{S}$ measurable function $\mathbf{x}$ from Ω into X is called a *random variable on* $(\Omega, \mathcal{F}, P)$.

□

The function $\mathbf{x}$ in definition 31 is measurable if and only if $\mathbf{x}^{-1}(A) \in \mathcal{F}$ for all $A \in \mathcal{S}$. A *finite* number of random variables, that are defined on a *common* probability space, is said to be a number of *jointly distributed* random variables. Formally,

Definition 32 (Jointly distributed random variables)
For $\mathbf{x}_i$, $i \in I$ random variables, indexed by a finite index set I, the variables $\mathbf{x}_i$, $i \in I$ are called *jointly distributed random variables* iff. they are defined on a common probability space.

□

The *joint probability law* of jointly distributed random variables is defined as follows,

Definition 33 (Joint probability law)
For $(\Omega, \mathcal{F}, P)$ a probability space, $(X_i, \mathcal{S}_i)$, $i \in I$ measurable spaces, indexed by a finite index set $I \triangleq \{i_1, i_2, \ldots, i_n\}$, and $\mathbf{x}_i$ an $\mathcal{F} - \mathcal{S}_i$ measurable function from Ω into X_i for every $i \in I$, the *joint probability law* of $\mathbf{x}_{i_1}, \mathbf{x}_{i_2}, \ldots, \mathbf{x}_{i_n}$, denoted by $P_{\mathbf{x}_{i_1}, \mathbf{x}_{i_2}, \ldots, \mathbf{x}_{i_n}}$, is a probability measure on $\otimes_{i \in I} \mathcal{S}_i$, defined in every measurable rectangle $A \triangleq \prod_{i \in I} A_i$ of $\{\mathcal{S}_i\}_{i \in I}$ by

$$P_{\mathbf{x}_{i_1}, \mathbf{x}_{i_2}, \ldots, \mathbf{x}_{i_n}}(A) \triangleq P(\bigcap_{i \in I} \mathbf{x}_i^{-1}(A_i)) \tag{3.16}$$

□

The notation $\otimes_{i \in I} \mathcal{S}_i$ in the above definition, stands for the so-called *product σ-field* of $\{\mathcal{S}_i\}_{i \in I}$. Notice that the joint probability law of the jointly distributed random variables $\mathbf{x}_{i_1}, \mathbf{x}_{i_2}, \ldots, \mathbf{x}_{i_n}$, indexed by the index set $I \triangleq \{i_1, i_2, \ldots, i_n\}$, is in general *not* the same as the joint probability law of the jointly distributed random variables $\mathbf{x}_{\amalg(i_1)}, \mathbf{x}_{\amalg(i_2)}, \ldots, \mathbf{x}_{\amalg(i_n)}$, indexed by the index set $I' \triangleq \{\amalg(i_1), \amalg(i_2), \ldots, \amalg(i_n)\}$, where $\amalg(i_1), \amalg(i_2), \ldots, \amalg(i_n)$ is any permutation of $i_1, i_2, \ldots, i_n$.

Independent random variables form a special kind of jointly distributed random variables. The notion of independent random variables will be of use when we come to random numbers in an attempt to develop the *extended* structured discrete event system construct in subsection 3.4. Formally,

Definition 34 (Independent random variables)
For $(\Omega, \mathcal{F}, P)$ a probability space, $(X_i, \mathcal{S}_i)$, $i \in I$ measurable spaces, indexed by a finite index set $I \triangleq \{i_1, i_2, \ldots, i_n\}$, $\mathbf{x}_i$ an $\mathcal{F} - \mathcal{S}_i$ measurable function from Ω into X_i for every $i \in I$, the random variables $\mathbf{x}_{i_1}, \mathbf{x}_{i_2}, \ldots, \mathbf{x}_{i_n}$ are called *(mutually) independent random variables* iff. for every measurable rectangle $A \triangleq \prod_{i \in I} A_i$ of $\{\mathcal{S}_i\}_{i \in I}$, it holds that

$$P_{\mathbf{x}_{i_1}, \mathbf{x}_{i_2}, \ldots, \mathbf{x}_{i_n}}(A) = P_{\mathbf{x}_{i_1}}(A_{i_1}) P_{\mathbf{x}_{i_2}}(A_{i_2}) \ldots P_{\mathbf{x}_{i_n}}(A_{i_n}) \tag{3.17}$$

□

Thus, keeping the notation of definition 34, in case $I \triangleq \{i_1, i_2\}$ and in case the random variables $\mathbf{x}_{i_1}$ and $\mathbf{x}_{i_2}$ are independent, then the value of their joint probability law at a measurable rectangle $A_{i_1} \times A_{i_2}$ of $\{\mathcal{S}_{i_1}, \mathcal{S}_{i_2}\}$ must coincide with the product of the value of the probability law of $\mathbf{x}_{i_1}$ at A_{i_1} with the value of the probability law of $\mathbf{x}_{i_2}$ at A_{i_2}, and this for all measurable sets A_{i_1} and A_{i_2} of $\mathcal{S}_{i_1}$ and $\mathcal{S}_{i_2}$ respectively.

In the above, we have introduced the notion of probability spaces, random variables, and jointly distributed random variables. We now define a *stochastic process* as follows,

Definition 35 (Stochastic process)
For $(\Omega, \mathcal{F}, P)$ a probability space, $(X_i, \mathcal{S}_i)$, $i \in I$ measurable spaces, indexed by an arbitrary index set I, and $\mathbf{x}_i$ an $\mathcal{F} - \mathcal{S}_i$ measurable function from Ω into X_i for every $i \in I$, the $\mathcal{F} - \otimes_{i \in I} \mathcal{S}_i$ measurable function ϖ from Ω into $\prod_{i \in I} X_i$, defined in every $\omega \in \Omega$ by $\nabla_{X_i} \circ \varpi(\omega) \triangleq \mathbf{x}_i(\omega)$ for every $i \in I$, is called a *stochastic process on* $(\Omega, \mathcal{F}, P)$.

□

In the definition, we use the notation ∇_{X_i} to denote the i^{th} projection function on $\prod_{i \in I} X_i$. Members of the *range* of a stochastic process are called *realisations* of the stochastic process. A stochastic process can also be defined as an indexed family of random variables that are defined on a common probability space. In case the index set is countable, we speak of a *discrete* stochastic process. In case the index set is uncountable, we speak of a *continuous* stochastic process.

We illustrate the concept of a stochastic process in the following example. In the example, we take back the problem situation of the simple coin striking process. The stochastic processes that we define in example 3.1 are processes that we typically think of when it comes to *validate* a stochastic system.

Example 3.1 (Probability space, random variable and stochastic process)
Let $\mathcal{S} \triangleq \langle T, X, \Omega, Y, \Sigma_0, \delta, \lambda\rangle$ be an abstract system, generated by a structured discrete event system $\mathcal{Q}$ that models the coin striking process.[12] Assume that input segments in the allowable input segment set of $\mathcal{S}$ are *pointwise* trajectories with coin arrivals. Assume that output segments of $\mathcal{S}$ are *pointwise* trajectories that indicate the number of coins in the buffer in front of the striking machine. Let now $t > 0$ be a time point in T, and $m \in \mathbb{N}$. Define

$$\Omega_{t,m} \triangleq \{\omega \mid \mathrm{dom}(\omega) \triangleq]0,t'] \text{ s.t } t' \geq t \text{ and } |\{t'' \mid \omega(t'') \neq \chi_\phi\}_{t'' \in]0,t']}| \geq m\}_{\omega \in \Omega} \tag{3.18}$$

Thus, very simply, for a particular choice of t and m, $\Omega_{t,m}$ assembles all allowable input trajectories over intervals that contain the interval $]0,t]$, and that schedule at least m coin arrivals. Assume now that, on the input side, we are interested in *inter-arrival time*. In that respect, define an index set $I_m \triangleq \{1, 2, \ldots, m\}$, and a function $\mathbf{x}_{t,m,i}$, $\forall i \in I$ from $\Omega_{t,m}$ into $\mathbb{R}^+$ in $\omega \in \Omega_{t,m}$ by

$$\mathbf{x}_{t,m,i}(\omega) \triangleq \inf_{t' \in \mathrm{dom}(\omega)}\{t' \mid c(\omega,t') \geq i\} - \inf_{t' \in \mathrm{dom}(\omega)}\{t' \mid c(\omega,t') \geq i-1\} \tag{3.19}$$

where c is a simple function from $\Omega_{t,m} \times T$ into $\mathbb{N}$ that maps a pair (ω, t') on the number of coin arrivals that are scheduled by ω to occur *before* or *at* time point t'. Every function $\mathbf{x}_{t,m,i}$, $\forall i \in I$ maps every input segment ω of $\Omega_{t,m}$ on the time in between the $(i-1)^{\text{th}}$ and the i^{th} coin arrival that is scheduled by ω.

Let now σ be an arbitrary initial state of $\mathcal{S}$. Keeping the time point t and the minimum number of coin arrivals m, define

$$\Theta_{\sigma,t,m} \triangleq \{\theta \mid \exists \omega \in \Omega_{t,m} : \theta = \iota_\sigma(\omega)\}_{\theta \in \mathscr{S}(T,Y)} \tag{3.20}$$

Thus, very simply, $\Theta_{\sigma,t,m}$ contains all output trajectories over intervals that contain the interval $]0,t]$, each of which is the image of some input segment in $\Omega_{t,m}$ under the input/output segment function ι_σ of $\mathcal{S}$. In other words, $\Theta_{\sigma,t,m}$ equals the *range* of the restriction of ι_σ to $\Omega_{t,m}$. Assume that we are interested on the output side in *queue length*. That being the case, let $J_t \triangleq [0,t]$ be an index set, and define a function $\mathbf{y}_{\sigma,t,m,j}$, $\forall j \in J_t$ from $\Theta_{\sigma,t,m}$ into $\mathbb{R}^+$ in $\theta \in \Theta_{\sigma,t,m}$ by[13]

$$\mathbf{y}_{\sigma,t,m,j}(\theta) \triangleq \begin{cases} \theta[\sup_{t' \in]0,j]}\{t' \mid \theta(t') \neq v_\phi\}] & \exists t' \in]0,j] \text{ s.t. } \theta(t') \neq v_\phi \\ \nabla_{\mathrm{range}(q)}(\sigma) & \textit{otherwise} \end{cases} \tag{3.21}$$

Every function $\mathbf{y}_{\sigma,t,m,j}$, $\forall j \in J_t$ maps an output segment θ of $\Theta_{\sigma,t,m}$ on the most recent observation

[12]The specific connection between structured discrete event systems and abstract systems is beyond the scope of this report. The induction of an *abstract system* by a *structured* discrete event system runs parallel to the induction of an abstract system by a *classic* discrete event system. For more information on the connection between *classic* discrete event systems and abstract systems, we refer to [31].

[13]In case $\theta_{j]}$ attains at every time point $t' \in]0,j]$ the no-output v_ϕ, then (3.21) evaluates in θ to the number of coins that were originally in the queue as specified by the initial state σ. In that respect, we made here the implicit assumption that one of the state variables of the abstract system $\mathcal{S}$ - that is denoted by q - describes the number of coins in the buffer in front of the striking machine.

$$\begin{tikzcd}
\Omega_{t,m} \arrow[r, "\iota_{\sigma|\Omega_{t,m}}"] \arrow[d, "\mathbf{x}_{t,m,i}"', "\text{all } i \in I_m"] & \Theta_{\sigma,t,m} \arrow[d, "\text{all } j \in J_t"', "\mathbf{y}_{\sigma,t,m,j}"] \\
\mathbf{R}^+ & \mathbf{R}^+
\end{tikzcd}$$

(a) Measurable functions

$$\begin{tikzcd}
P \arrow[r, dashed, "\iota_{\sigma|\Omega_{t,m}}"] \arrow[d, dashed, "\mathbf{x}_{t,m,i}"', "\text{all } i \in I_m"] \arrow[dr, dashed] & P_{\iota_{\sigma|\Omega_{t,m}}} \arrow[d, dashed, "\text{all } j \in J_t"', "\mathbf{y}_{\sigma,t,m,j}"] \\
P_{\mathbf{x}_{t,m,i}},\ i \in I_m & P_{\mathbf{y}_{\sigma,t,m,j}},\ j \in J_t
\end{tikzcd}$$

(b) Probability laws

Figure 5: Measurable functions and probability laws

on the queue length - if there is any such observation -,[14] that lies on the interval $]0, j]$.

For the functions $\mathbf{y}_{\sigma,t,m,j},\ j \in J_t$ to be *measurable*, we impose on $\Theta_{\sigma,t,m}$ the minimal σ-field over the class of all preimages of measurable sets of the Borel σ-field on $\mathbf{R}^+$ under $\mathbf{y}_{\sigma,t,m,j},\ j \in J_t$.[15] In other words, we impose on $\Theta_{\sigma,t,m}$ the minimal σ-field over

$$\mathcal{A}_{\Theta_{\sigma,t,m}} \triangleq \bigcup_{j \in J_t} \{\mathbf{y}^{-1}_{\sigma,t,m,j}(A) \mid A \in \mathbb{B}(\mathbf{R}^+)\} \tag{3.22}$$

Having installed $\sigma(\mathcal{A}_{\Theta_{\sigma,t,m}})$ on $\Theta_{\sigma,t,m}$, the functions $\mathbf{y}_{\sigma,t,m,j},\ j \in J_t$ become functions that are defined on a common measurable space. In a similar way, we impose on $\Omega_{t,m}$ a σ-field such that all the functions $\mathbf{x}_{t,m,i},\ i \in I_m$ *and* the restriction of the input/output segment function ι_σ to $\Omega_{t,m}$ are measurable. Thus, we impose on $\Omega_{t,m}$ the minimal σ-field over

$$\mathcal{A}_{\Omega_{t,m}} \triangleq \bigcup_{i \in I_m} \{\mathbf{x}^{-1}_{t,m,i}(A) \mid A \in \mathbb{B}(\mathbf{R}^+)\} \cup \{\iota^{-1}_{\sigma|\Omega_{t,m}}(A) \mid A \in \sigma(\mathcal{A}_{\Theta_{\sigma,t,m}})\} \tag{3.23}$$

Figure 5(a) puts the sets and the functions that were defined above in perspective. In figure 5(a), we placed the sets $\Omega_{t,m}$ and $\Theta_{\sigma,t,m}$, and positioned the functions $\iota_{\sigma|\Omega_{t,m}}$, $\mathbf{x}_{t,m,i},\ i \in I_m$ and $\mathbf{y}_{\sigma,t,m,j},\ j \in J_t$. In figure 5(b), we illustrated how a probability *measure* P on $\sigma(\mathcal{A}_{\Omega_{t,m}})$ is turned into a probability *law* $P_{\mathbf{x}_{t,m,i}}$ for every $i \in I_m$, a probability law $P_{\iota_{\sigma|\Omega_{t,m}}}$, and a probability law $P_{\mathbf{y}_{\sigma,t,m,j}}$ for every $j \in J_t$. Carefully observe that the functions $\mathbf{x}_{t,m,i},\ i \in I_m$ are *random variables* defined on a common probability space $(\Omega_{t,m}, \sigma(\mathcal{A}_{\Omega_{t,m}}), P)$, as are the functions $\mathbf{y}_{\sigma,t,m,j} \circ \iota_{\sigma|\Omega_{t,m}},\ j \in J_t$. In view of the definition of a stochastic process, they define two stochastic processes on $(\Omega_{t,m}, \sigma(\mathcal{A}_{\Omega_{t,m}}), P)$.

$\triangleleft$

[14]If there is no such observation, then θ is mapped on the initial queue length, as specified by the initial state.
[15]The minimal σ-field over a class of sets is the intersection of *all* σ-fields that contain the sets in the class. The Borel σ-field on $\mathbf{R}^+$ contains all intervals, singleton-sets and countable unions and intersections of intervals in $\mathbf{R}^+$.

3.2 Random number generators

We generate realisations of a random variable with the help of a *random number generator*. We define a random number generator as follows,

Definition 36 (Random number generator)
For R a subset of $[0,1]$, a function Z on R is called a *random number generator* or *RNG* with coverage R and period p iff. it is a 1-1 function onto R, and it holds for every $r \in R$ that $Z(r) \neq r$ and $\underbrace{Z \circ Z \circ \ldots \circ Z}_{p \text{ times}}(r) = r$.

$\square$

In the following, for Z a random number generator, we denote $Z \circ Z$ as Z^2, $Z \circ Z \circ Z$ as Z^3, etc. Members of the coverage of a random number generator are called *(pseudo)-random numbers.* Notice that it follows readily from the definition that a random number generator exhibits *cyclical* behaviour. In effect, for Z a random number generator with period p, the definition implies that we can use Z to generate a sequence of at most p *unique* random numbers. The $(p+1)^{\text{th}}$ number that we generate, will equal the first number that we started from. We call *any* sequence of random numbers generated by a random number generator, a *stream* of the generator. The first random number in a stream is called the *seed* of the stream.

From definition 36, it follows that we can construct a multiple of random number generators that all have one and the same coverage R and period p. It is then only natural to ask whether all those generators are equally *valuable.* The answer is in the *negative.* For, the *quality* of a random number generator depends on the extent that it induces a *uniform i.i.d process.*[16] To give an example, assume that the quality of a random number generator Z, the specification of which is unknown to us, needs to be established. For that purpose, we are allowed to perform an experiment, whereby we take an arbitrary random number r from the coverage of Z, designate any *finite* number $i_1, i_2, \ldots, i_n$ of coordinates, and ask a neutral supervisor who has knowledge of Z to return the $i_1^{\text{th}}, i_2^{\text{th}}, \ldots, i_n^{\text{th}}$ random number in the (infinite) stream with seed r that is generated by Z. If we are allowed to repeat this experiment as many times as we like, picking each time a seed and keeping the *same* coordinates, then we can construct a probability measure on $\otimes_{j \in J} \mathbb{B}(\mathbf{R}_j)$, where $\mathbf{R}_j$ denotes a copy of $\mathbf{R}$ for every $j \in J$ and where $J \triangleq \{i_1, i_2, \ldots, i_n\}$.[17] This measure assigns to every measurable rectangle of $\{\mathbb{B}(\mathbf{R}_j)\}_{j \in J}$ the relative frequency of n-tuples of random numbers, found in all our experiments, that belong to the rectangle. Now, assume that we are allowed to redo the former experiments for *all* possible finitely many coordinates that we can choose. Then, it is not difficult to see intuitively that the probability measures that we will hence obtain constitute a family of *consistent* measures. Therefore, by the well-known Kolmogorov's extension theorem, this family induces a probability measure P on $\otimes_{i \in I} \mathbb{B}(\mathbf{R}_i)$, where $I \triangleq \{1, 2, \ldots\}$. Hence, we can

[16]A stochastic process with *independent* and *identically* distributed random variables.
[17]The notation $\mathbb{B}(\mathbf{R})$ stands for the Borel σ-field on $\mathbf{R}$.

consider every $\otimes_{i\in I}\mathbb{B}(\mathbf{R}_i) - \mathbb{B}(\mathbf{R}_i)$ measurable function $\mathbf{x}_i$, $\forall i \in I$, that assigns to every infinite stream of random numbers in $\prod_{i\in I} \mathbf{R}_i$ the random number at the i^{th} coordinate in the stream, as a random variable on $(\prod_{i\in I} \mathbf{R}_i, \otimes_{i\in I}\mathbb{B}(\mathbf{R}_i), P)$. The variables $\{\mathbf{x}_i\}_{i\in I}$ constitute a stochastic process on $(\prod_{i\in I} \mathbf{R}_i, \otimes_{i\in I}\mathbb{B}(\mathbf{R}_i), P)$. Now, we like to call the random number generator Z an *(ε, p)-ideal random number generator* if and only if the random variables $\mathbf{x}_j$ and $\mathbf{x}_k$ are independent for all $j, k \in I$ s.t. $0 < |j-k| \leq p-1$, *and* the joint probability law of any finite number of random variables $\mathbf{x}_{i_1}, \mathbf{x}_{i_2}, \dots, \mathbf{x}_{i_n}$ deviates (in absolute terms) at *any* measurable rectangle of $\{\mathbb{B}(\mathbf{R}_j)\}_{j\in J}$ at most ε from the joint probability law of n independent (continuous) *uniformly* distributed random variables that have the unit interval as their range, at the same measurable rectangle. We refer to $1-\varepsilon$ as the *density* of the generator.

3.3 Replications

In the above, we introduced the notion of random numbers and a random number generator. At the formal level of modelling, we like to refine the structured discrete event system concept of section 2 such that it allows to model apparent *randomness* in the behaviour of one or more descriptive variables of interest. For that purpose, we include a *random number generator component* in the component set of a structured discrete event system. For Z a random number generator with coverage R and period p, we call a component G in the component set of a structured discrete event system $\mathcal{Q}$, that has influencers $c_1, c_2, \dots, c_n$, a *random number generator component (that implements Z)* iff. 1) the state set of G is defined by $\Sigma^G \triangleq R^s$, where s denotes the different *purposes* for which random numbers are used in the model, *and* 2) for every initial state σ_0 of $\mathcal{Q}$, with $\sigma_0^G \triangleq \nabla_{\Sigma^G}(\sigma_0)$ and $\sigma^{[A]G} \triangleq (\nabla_{\Sigma^{c_1}}(\sigma_0), \nabla_{\Sigma^{c_2}}(\sigma_0), \dots, \nabla_{\Sigma^{c_n}}(\sigma_0))$, it holds that

$$\delta_\varphi^G(\sigma_0^G, \sigma^{[A]G}, \tau(\sigma_0)) \triangleq (\nabla_{R_1}(\sigma_0^G), \dots, \underbrace{Z(\nabla_{R_j}(\sigma_0^G))}_{\text{for at most one } j \in \{1,2,\dots,s\}}, \dots, \nabla_{R_s}(\sigma_0^G)) \tag{3.24}$$

with τ the time-event advance function of $\mathcal{Q}$, and R_i a copy of R for every $i \in \{1, 2, \dots, s\}$. We thus whish to agree here that, for *every* state variable of $\mathcal{Q}$ for which we like to model apparent randomness in behaviour, we reserve one particular state variable of the random number generator component. To give an example, if we decide in a more elaborate coin striking case to model apparent randomness for the total processing time on a number of different machines, then we will reserve a separate generator component state variable for every machine that we whish to sample varying processing times for. Every time that a processing time needs to be determined for a machine, we consume the random number that is held by the corresponding state variable of the random number generator component. The generator component then undergoes a state transition as specified by (3.24), where j points to the coordinate that holds the consumed random number.

In the following, we refer to a structured discrete event system that contains a random number generator component as a *random structured discrete event system* or an *rsDEVS*. A point that we whish to emphasise is the following. Assume that we are presented with an abstract system $\mathcal{S} \triangleq \langle T, X, \Omega, Y, \Sigma_0, \delta, \lambda \rangle$ that has been generated by a random structured discrete event system $\mathcal{Q} \triangleq \langle \bar{T}, X, (E, \preceq), Y, C, \{\Sigma_0^c\}_{c \in C}, \Sigma_0, A, \{\delta_\phi^c\}_{c \in C}, \delta_X, \Gamma, \lambda \rangle$. Let $\mathcal{S}$ be initialised in some *partial* initial state. By that, we mean that the value of every state variable of $\mathcal{S}$ has been specified, with the exception of the state variables of the random number generator component of $\mathcal{Q}$. Then, if we whish to perform an experiment with $\mathcal{S}$, we must first 'complete' the partial initial state to obtain an initial state of $\mathcal{S}$, after which we can apply an allowable input segment to this initial state. It is now only natural to ask, moving from the partial initial state to the initial state, whether there are some combinations of values for the random number state variables that are *inferior* to other combinations. The answer is in the *positive*. For, assume that we must determine initial values for two random number state variables. If we choose these initial values (or seeds) *unwisely*, in the sense that after a relatively low number of state transitions of the random number generator component, one of the state variables of this component attains the seed that we used for the other state variable, then we will induce an unwanted (short-term) temporal correlation between the descriptive variables for which we use random numbers from the generator component. This is not desirable. In that respect, to properly define the experiments that we are *allowed* to carry out with $\mathcal{S}$, we whish to introduce here some additional terminology. We define a *partial initial state set* of $\mathcal{S}$ by

Definition 37 (Partial initial state set)
For $\mathcal{Q}$ a random structured discrete event system with component set $C \triangleq \{c_i\}_{i \in I}$, where $I \triangleq \{1, 2, \ldots, n\}$, random number generator component $G \triangleq c_n$ and component initial state sets $\Sigma_0^{c_i}$, $i \in I$, and $s(\mathcal{Q})$ the abstract system that is generated by $\mathcal{Q}$ with initial state set Σ_0', the *partial initial state set* of $s(\mathcal{Q})$, denoted by Π_0, is defined by

$$\Pi_0 \triangleq \{\underbrace{[(\nabla_{\Sigma_0^{c_1}}(\sigma), \nabla_{\Sigma_0^{c_2}}(\sigma), \ldots, \nabla_{\Sigma_0^{c_{n-1}}}(\sigma)), e]}_{\text{organised in a } \sum_{i=1}^{n-1} \dim(\Sigma_0^{c_i}) + 1 \text{ tuple}} \mid \bar{\sigma} \triangleq (\sigma, e) \in \Sigma_0'\} \tag{3.25}$$

$\square$

A partial initial state of an abstract system $\mathcal{S} \triangleq s(\mathcal{Q})$ that is generated by a random structured discrete event system $\mathcal{Q}$, pins down a value for every state variable of $\mathcal{S}$, apart from those state variables that are random number component generator state variables of $\mathcal{Q}$. For π a partial initial state of $\mathcal{S}$, we call a *pair* that carries at its first coordinate an ordered tuple with a *seed* for every random number state variable of $\mathcal{Q}$, and at its second coordinate an allowable input segment for $\mathcal{S}$, a *replication with $\mathcal{S}$ in π*. Formally,[18]

[18]The way that we define a replication differs from the more pragmatic description that is often given in the simulation literature, where a replication is described as a *run* with a simulation model - see e.g. [16].

Definition 38 (Replication)
For $\mathcal{Q}$ a random structured discrete event system, r an initial state of the random number generator component of $\mathcal{Q}$, ω an input segment in the allowable input segment set of $s(\mathcal{Q})$, and π a partial initial state of $s(\mathcal{Q})$, the pair (r, ω) is called a *replication with* $s(\mathcal{Q})$ *in* π.

□

We call *length* of a replication the length of the domain of the input segment that the replication contains. To *perform* a replication with an abstract system in a partial initial state reads as 1) to *construct* an initial state out of the partial initial state and the random number stream seeds of the replication, 2) to *apply* the allowable input segment of the replication in the resulting initial state, and 3) to *use* the input/output segment function of the abstract system to determine the output segment. We refer to the input/output segment pair that is obtained as the *outcome* of the replication.

3.4 The extended structured discrete event system construct

In the following, we organise replications that we allow ourselves to perform with an abstract system in *replication sets.* We define a replication set for every partial initial state of the abstract system. We denote a replication set, associated with a partial initial state π, by Δ_π. With the structured discrete event system formalism of section 2, and with the notion of replication sets in mind, we now define an *extended structured discrete event system* as follows,

Definition 39 (Extended structured discrete event system)
For $\mathcal{Q}$ a random structured discrete event system, Π_0 the partial initial state set of $s(\mathcal{Q})$, and Δ_π, $\pi \in \Pi_0$ replication sets for all partial initial states of $s(\mathcal{Q})$, an *extended structured discrete event system* is an ordered pair, defined by

$$\mathcal{Q}_e \triangleq \langle \mathcal{Q}, \{\Delta_\pi\}_{\pi \in \Pi_0} \rangle \tag{3.26}$$

□

An extended structured discrete event system $\mathcal{Q}_e$ embodies a random structured discrete event system $\mathcal{Q}$ on the one hand, and a family $\{\Delta_\pi\}_{\pi \in \Pi_0}$ of replication sets on the other hand. The importance of replications, that in fact constrain the experiments that we are *allowed* to perform with the abstract system that is generated by $\mathcal{Q}$, will come to its full right when we address the behaviour of a stochastic system that embodies $\mathcal{Q}$.

In the following, we create a pair of prolonged examples to illustrate our structured discrete event system construct. First, we develop a random structured discrete event system that models the striking process of coins in the aforementioned simple coin striking example. Then, we create a random structured discrete event system that models the *generation* of coin planchets, which are offered to the former system. After the examples, we comment on how we intend to create a family of replication sets for both random structured discrete event systems.

Coin striking process In table 1, we summarised the specification of our random structured discrete event system for the simple coin striking process. The upper part of the table defines an extended time set $\bar{T}$, an external event set X and output set Y, as well as a partially ordered interval event set E. We use the keyword *coin* to indicate the arrival of a coin to the system. The symbol χ_ϕ stands for the *no-external-event.* The internal events ϵ_I, ϵ_C and ϵ_ϕ stand for *striking process initiated*, *striking process completed* and the *no-internal-event* respectively. The symbol v_ϕ represent the no-output. In case the events ϵ_C and ϵ_I are anticipated to occur simultaneously, then we will always handle ϵ_C first.

The center part of the table defines a component set C, initial state sets Σ_0^c, $c \in C$ of the components in C, and a causality-dependency relation A. The components that we identified are Q *(queue)*, S *(striking machine)* and G *(random number generator)*. We describe Q by a state variable q that indicates the number of coin planchets in the queue. We describe S by a state variable s that denotes the *remaining* striking time of the coin currently in process. We describe G by a single random number state variable r, that we use to sample (total) striking times for S. The initial state set of Q and G equal respectively the set of natural numbers $\mathbb{N}$ and the coverage R of the random number generator Z that is implemented by G. We let Z be the prime modulo linear congruential generator, defined in $r \in R$ by[19]

$$Z(r) \triangleq [a(mr) \bmod m]/m \tag{3.27}$$

where $a \triangleq 7^5$ and $m \triangleq 2^{31} - 1$. Further, we let the initial state set of S be the closed interval $[0, \max_{r \in R}\{F_\mu^{-1}(r)\}]$, where F_μ^{-1} is the inverse function of the so-called cumulative *exponential* distribution function F_μ, that is defined in $x \in \mathbb{R}$ by

$$F_\mu(x) \triangleq \begin{cases} 1 - e^{-\mu x} & x \geq 0 \\ 0 & \textit{otherwise} \end{cases} \tag{3.28}$$

where the parameter μ in (3.28) is some fixed, strictly positive real number such that $\frac{1}{\mu}$ has the interpretation of the average *total striking time.* The idea behind the initial state set of S is that the largest *total* striking time that we can possibly realise, equals $\max_{r \in R}\{F_\mu^{-1}(r)\}$, while the *remaining* striking time can be any point in the interval $[0, \max_{r \in R}\{F_\mu^{-1}(r)\}]$, since we can apply a coin arrival at any time we want.

The final row of the center part of table 1 defines a causality-dependency relation A in the component set C. Recall from our discussion in section 2 that, for a component c to be related by A to another component c', 1) the component c must be a *scheduling component* for an internal event, that upon occurrence alters the state of c' in a *significant* way - thus ruling out state changes that come down to a reduction with the time since the last state

[19]For more information on this kind of generators, we refer to [16].

	rsDEVS specification
$\bar{T}$	$[0, +\infty]$
X	$\{coin, \chi_\phi\}$
Y	$\mathbb{N} \cup \{\upsilon_\phi\}$
E	$\{\epsilon_I, \epsilon_C, \epsilon_\phi\} \quad \epsilon_C \prec \epsilon_I$
C	$\{Q, S, G\}$
Σ_0^Q	$\mathbb{N}$
Σ_0^S	$[0, \max_{r \in R}\{F_\mu^{-1}(r)\}]$
Σ_0^G	R
Σ_0	$\{\sigma \mid \nabla_{\Sigma_0^S}(\sigma) \leq Z^{(p-1)}(\nabla_{\Sigma_0^G}(\sigma))\}_{\sigma \in \Sigma_0^Q \times \Sigma_0^S \times \Sigma_0^G}$
A	$Q[A]S \quad Q[A]G \quad S[A]Q \quad S[A]G \quad G[A]S$
δ_ϕ^Q	$(q, s, (u, \epsilon_I)) \mapsto q - 1$
δ_ϕ^S	$(s, (q, r), (u, \epsilon_I)) \mapsto F_\mu^{-1}(r) \quad (s, (q, r), (u, \epsilon_C)) \mapsto 0$
δ_ϕ^G	$(r, (q, s), (u, \epsilon_I)) \mapsto Z(r)$
δ_X	$[((q, s, r), e), x] \mapsto \begin{cases} (q+1, \max\{(s-e), 0\}, r) & x = coin \\ (q, s, r) & otherwise \end{cases}$
Γ	$q > 0 \Rightarrow (s, \epsilon_I) \quad s > 0 \Rightarrow (s, \epsilon_C) \quad q = s = 0 \Rightarrow (+\infty, \epsilon_\phi)$
λ	$(q, s, r) \mapsto q$

Table 1: Simple coin striking rsDEVS

transition -, *or* 2) the state of component c must be known in order to determine the next state of component c' under an internal event. This policy that we proposed to determine the *influencers* for each component yields the following result for this example. Looking ahead at the specification of the internal event scheduler Γ in the lower part of table 1, both Q and S are scheduling components for ϵ_I.[20] When this event occurs, we intend to reduce the queue length, to generate a total striking time, and - as we consume a random number to accomplish the former action - to advance the value of the random number state variable such that its new value points to the random number that *follows* the current random number (according to the random number generator Z). In that respect, since Q and S are scheduling components of ϵ_I, and since this event upon occurrence significantly affects the components Q, S and G, we retain Q as an *active* influencer of S and G, and S as an *active* influencer of Q and G. Applying a similar reasoning for the events ϵ_C and ϵ_ϕ, we find that S is the only scheduling component for ϵ_C, while both Q and S are scheduling components for ϵ_ϕ.[21] As we intend to simply reduce the value of the state variable of S with the time since the last state transition when ϵ_C occurs, and

[20]For, the state of Q is required to determine *whether or not* ϵ_I will occur in the future, while the state of S is required to determine the *amount of time* that needs to elapse until ϵ_I will occur.

[21]In effect, we need the state of S to determine *whether or not* ϵ_C will occur in the future, and how many *time units* need to elapse before ϵ_C will occur. Further, the states of both Q and S are required to determine *whether or not* the no-internal-event ϵ_ϕ will occur.

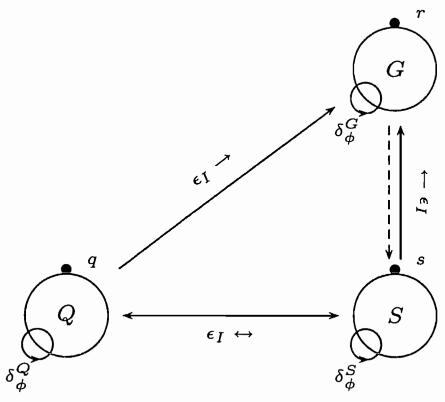


Figure 6: Simple coin striking rsDEVS component interaction scheme

as we intend to leave all state variables unchanged under ϵ_ϕ, none of these events serves as an event through which one component actively influences another component. Finally, looking ahead at the specification of the local transition functions in the lower part of table 1, we see that component G forms a *passive* influencer of S, as its state is required to generate a total striking time.

In the lower part of table 1, we summarised the specification of the local transition functions δ_ϕ^c, $c \in C$, and specified an external state transition function δ_X, an internal event scheduler Γ, and a discrete event output function λ. With respect to the local transition functions, we displayed for each function only those time-event pairs that result in a state *change* of the corresponding component. When the current state of the structured discrete event system equals $\sigma \triangleq (q, s, r)$, and when the time-event pair with the next internal event from the viewpoint of this state equals $\tau(\sigma) \triangleq (u, \epsilon_I)$, then the next state of Q, S and G will be $q - 1$, $F_\mu^{-1}(r)$ and $Z(r)$ respectively - assuming no external event takes place before the interval event ϵ_I is processed. In case $\tau(\sigma) \triangleq (u, \epsilon_C)$, then the state of S will change to 0 when the event ϵ_C is processed - again, assuming the absence of external events. The specification of the external state transition function δ_X indicates that we add a coin planchet to the queue whenever the external event *coin* is applied. Also, we reduce the remaining striking time by the time since the last state transition, with the help of the *system time variable e*. Under the no-external-event, the state of the system remains unaltered (as does its total state). The specification of the internal event scheduler Γ should be clear from our coverage of the causality-dependency relation A. Finally, the discrete event output function λ maps every initial, non-transitory state on the number of coin planchets in the buffer.

In figure 6, we constructed a component interaction scheme to visualise the structure of the random structured discrete event system specification of table 1. Components are indicated by large white circles, while their state variables are depicted by small solid circles attached to the components. Solid lines in the scheme represent *active* influences, while dashed lines stand for *passive* influences. Since Q and S actively influence one another through the event ϵ_I, and since these components both actively influence G through the same event, we placed a solid arrowed line from Q to S and vice versa, and from both Q and S to G, accompanied each time by the label ϵ_I. The passive influence of G on S is indicated by a dashed arrowed line from G to S.

Coin planchet generation process In the table in figure 7, we summarised our structured discrete event system specification for the process that generates coin planchets to the system that we developed above. The first block of the table defines an extended time set $\bar{T}$, an external event set X, an output set Y and an internal event set E. Carefully observe that the external event set is a singleton-set, containing the no-external-event χ_ϕ. The output set contains the outputs *coin* and v_ϕ. We reserve the output *coin* in case a coin planchet is ready to be advanced to the coin striking process. As usual, v_ϕ stands for the no-output. Apart from the no-internal-event ϵ_ϕ, we defined an internal event ϵ_{cA}, which stands for *coin arrival*, and which we let take place every time a coin planchet is advanced to the coin striking process.

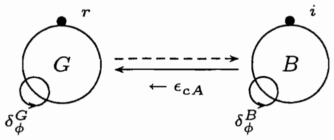
The second block in the table of figure 7 defines a component set C, component initial state sets Σ_0^c, $c \in C$, an initial state set Σ_0 for the structured discrete event system, and a causality-dependency relation A. We identify two components: a *blanks* component B, representing the production of blanks (coin planchets) - albeit at an highly aggregated level -, and a *random number generator* component G, random numbers of which we intend to use to generate the time *in between* consecutively produced blanks. In that respect, we define a single state variable i for B, that represents the *remaining inter-arrival time.* We define a single state variable r for G, that represents the *random number* that we use to generate the next (total) inter-arrival time. We employ the prime modulo linear congruential generator Z of (3.27), with parameters $a \triangleq 7^5$ and $m \triangleq 2^{31} - 1$ to model the state transitions of the random number generator component. The (initial) state set of G hence equals the coverage R of this random number generator.

Each time the *coin arrival* event ϵ_{cA} takes place - each time a coin planchet is advanced to the striking process -, we use the current state of the random number generator component to generate a realisation of a random variable with cumulative distribution function $F_{\underline{\lambda}}$, with $F_{\underline{\lambda}}$ defined in $x \in \mathbf{R}$ by

$$F_{\underline{\lambda}}(x) \triangleq \begin{cases} 1 - e^{-\underline{\lambda} x} & x \geq 0 \\ 0 & otherwise \end{cases} \tag{3.29}$$

	rsDEVS specification
$\bar{T}$	$[0,+\infty]$
X	$\{\chi_\phi\}$
Y	$\{coin, \upsilon_\phi\}$
E	$\{\epsilon_{cA}, \epsilon_\phi\}$
C	$\{B,G\}$
Σ_0^B	$\{F_{\underline{\lambda}}^{-1}(r)\}_{r\in R}$
Σ_0^G	R
Σ_0	$\Sigma_0^B \times \Sigma_0^G$
A	$B[A]G \quad G[A]B$
δ_ϕ^B	$(i,r,(u,\epsilon_{cA})) \mapsto F_{\underline{\lambda}}^{-1}(r)$
δ_ϕ^G	$(r,i,(u,\epsilon_{cA})) \mapsto Z(r)$
δ_X	$[((i,r),e),\chi_\phi] \mapsto (i,r)$
Γ	(i,ϵ_{cA})
λ	$(i,r) \mapsto coin$

(a) System specification



(b) Component interaction scheme

Figure 7: System specification and component interaction scheme of simple coin planchet production process

where the parameter $\underline{\lambda}$ is some predetermined strictly positive real number such that $\frac{1}{\underline{\lambda}}$ carries the interpretation of the average *total inter-arrival time.* Thus, very simply, total inter-arrival times are realisations of an exponentially distributed random variable with cumulative distribution function $F_{\underline{\lambda}}$.

Since ϵ_{cA} and ϵ_ϕ are the only internal events, a time-event pair with an internal event is either of the form (u, ϵ_{cA}), or equals $(+\infty, \epsilon_\phi)$. The state of the components B and G remains as usual unaltered under the latter time-event pair. In case the ϵ_{cA} event takes place, we require a random number to determine the next state of B. Hence, the random number generator component is a (passive) influencer of B, and thus $G[A]B$. Looking ahead at the internal event scheduler Γ, component B is a *scheduling component* for ϵ_{cA}, as we anticipate ϵ_{cA} to take place after i time units, from *any* state. Since an occurrence of ϵ_{cA} alters the state of G, component B is an (active) influencer of G, and hence $B[A]G$.

The specification of the local transition functions δ_ϕ^B and δ_ϕ^G in the final block of the table in figure 7, should be clear from our presentation so far. Each time the event ϵ_{cA} takes place, we consume the current random number r and generate a (total) inter-arrival time $F_{\underline{\lambda}}^{-1}(r)$. Also, we change the state of the random number generator into $Z(r)$. With regard to the external state transition function δ_X, notice how the state of the system remains unaffected by the no-external-event χ_ϕ. Finally, for $\sigma \triangleq (i, r)$ *any* initial state, we let the only anticipated time-event pair by the internal event scheduler Γ equal the time-event pair (i, ϵ_{cA}).

As we have seen in our discussion of output of a structured discrete event system, every time an external or internal event takes place, that differs from the no-external-event, the discrete event output function λ of the system specification is applied to generate an output. Since ϵ_{cA} forms here the only significant event in the system, we let the discrete event output function λ map *every* non-transitory initial state of the system on the output *coin*. A timed output sequence of the system will thus be a timed sequence with coin arrivals, ready to be applied as a timed external event sequence for the system that we developed for the striking process itself.

Replication sets In the examples above, we developed two (generic) structured discrete event systems. Let's denote these system specifications by $\mathcal{Q}_\mu$ and $\mathcal{Q}_{\underline{\lambda}}$ respectively, and denote their induced abstract systems by $\mathcal{S}_\mu \triangleq s(\mathcal{Q}_\mu)$ and $\mathcal{S}_{\underline{\lambda}} \triangleq s(\mathcal{Q}_{\underline{\lambda}})$. Notice that for every choice of μ and $\underline{\lambda}$, we obtain a pair of fully specified and calibrated structured discrete event systems. In the following, we assume that μ and $\underline{\lambda}$ are fixed.

In order to build *extended* structured discrete event systems from $\mathcal{Q}_\mu$ and $\mathcal{Q}_{\underline{\lambda}}$, we have to define a replication set for every partial initial state of the abstract system $\mathcal{S}_\mu$ and for every partial initial state of the abstract system $\mathcal{S}_{\underline{\lambda}}$. Denoting the initial state sets of of $\mathcal{S}_\mu$ and $\mathcal{S}_{\underline{\lambda}}$ by Σ_0 and Σ_0' respectively, the partial initial state sets of $\mathcal{S}_\mu$ and $\mathcal{S}_{\underline{\lambda}}$, denoted by Π_0 and Π_0', are defined by

$$\Pi_0 \triangleq \{(q, s, e) \mid \bar{\sigma} \triangleq ((q, s, r), e)\}_{\bar{\sigma} \in \Sigma_0} \qquad \Pi_0' \triangleq \{(i, e) \mid \bar{\sigma} \triangleq ((i, r), e)\}_{\bar{\sigma} \in \Sigma_0'} \tag{3.30}$$

Let now π' be an arbitrary partial initial state of Π_0'. Assume that we whish to retrieve from the outcome of *every* replication that we perform with $\mathcal{S}_{\underline{\lambda}}$ in π', at least m observations on inter-arrival time, for some $m \in \mathbb{N}$. Then, we define for every random number r in the coverage R of the generator that is used by $\mathcal{Q}_{\underline{\lambda}}$, *one* input segment Λ of an appropriate duration,[22] such that $\mathcal{S}_{\underline{\lambda}}$ returns an output segment with at least m coin planchet arrivals in response to Λ, when it is initialised in the state, made up by the partial initial state π' and the random number r. Thus, very simply, our replication set $\Delta_{\pi'}'$ for $\mathcal{S}_{\underline{\lambda}}$ in π' contains a total of p replications, where p is the period of the random number generator that is used. The outcome of every replication allows us to compute m coin inter-arrival times. Repeating the above for every partial initial state of $\mathcal{S}_{\underline{\lambda}}$, yields a family of replication sets. The system $\mathcal{Q}_{\underline{\lambda}}$, together with the replication sets $\Delta_{\pi'}'$, $\pi' \in \Pi_0'$, define an *extended* structured discrete event system $\mathcal{Q}_{\underline{\lambda},e}$.

[22]We use the notation Λ here, since input segments to $\mathcal{S}_{\underline{\lambda}}$ must be so-called *null-segments* because the external event set of $\mathcal{Q}_{\underline{\lambda}}$ carries only the no-external-event χ_ϕ.

Let now π be a partial initial state of $\mathcal{S}_\mu$. We whish to define for every partial initial state π' of $\mathcal{S}_{\underline{\lambda}}$ and for every replication (r', Λ) in $\Delta'_{\pi'}$, *one and only one* replication with $\mathcal{S}_\mu$ in π. We let this replication carry the output segment ω that is produced by $\mathcal{S}_{\underline{\lambda}}$ in π' under the replication (r', Λ). Assume that we whish to observe the queue length over some predetermined interval $]0, t]$ in the outcome of every replication with $\mathcal{S}_\mu$ in π. Then, all that we have to do is take care that the input segments that we defined in replications with $\mathcal{S}_{\underline{\lambda}}$ are defined over an interval that contains $]0, t]$. Further, we choose an appropriate random number r of the coverage R *such that* during the replication (r, ω) with $\mathcal{S}_\mu$ in π, *no* random numbers are employed that were used during the replication (r', Λ) with $\mathcal{S}_{\underline{\lambda}}$ in π'. Is re-usage of random numbers a source of great concern? The answer is *yes*, for it most likely induces an unwanted temporal correlation between *future* striking times and *past* inter-arrival times. Keeping the partial initial states π' and π, we now repeat the above for every replication that we defined in $\Delta'_{\pi'}$. This yields a total of p replications with $\mathcal{S}_\mu$ in π, *conditional* on the partial initial state π' of $\mathcal{S}_{\underline{\lambda}}$. Looking ahead at the notation that we introduce in subsection 3.6, we assemble these replications in a *conditional replication set* $\Delta_\pi \triangleleft \langle \mathcal{Q}_{\underline{\lambda}}, \pi' \rangle$. Next, we repeat all of the above for every partial initial state of $\mathcal{S}_{\underline{\lambda}}$, keeping the partial initial state π of $\mathcal{S}_\mu$. Reviewing the partial initial state set of $\mathcal{S}_{\underline{\lambda}}$ that we defined in (3.30), this yields *uncountably* many replications with $\mathcal{S}_\mu$ in π. In that respect, given the partial initial state π of $\mathcal{S}_\mu$, we have defined so far a (candidate) replication set $\bigcup_{\pi' \in \Pi'_0} \Delta_\pi \triangleleft \langle \mathcal{Q}_{\underline{\lambda}}, \pi' \rangle$ for $\mathcal{S}_\mu$ in π.

Although we could in theory stop right here, and define a family of replication sets for $\mathcal{S}_\mu$ applying the approach that we introduced above for every partial initial state of $\mathcal{S}_\mu$, we like to repeat our above procedure for a *number* of system specifications $\mathcal{S}_{\underline{\lambda}_k}$, $k \in K$, where K is an index set that indexes different *arrival rates*. We like to think of deciding on the arrival rate of coin planchets as a typical *calibration* problem that we face when we have to develop a formal model of the input part of a conceptual model for the entire coin striking process. To be more precise, in view of postulate 6 on an *environment*, we like to define an environment here as a family of random structured discrete event systems $\mathcal{Q}_{\underline{\lambda}_k}$, $k \in K$ for some index set K, where every model in the environment generates coin planchets at a *different* rate. In that respect, the replication set that we eventually whish to define for every partial initial state π of Π_0 can be written as $\Delta_\pi \triangleq \bigcup_{k \in K} \bigcup_{\pi' \in \Pi'_{0,k}} \Delta_\pi \triangleleft \langle \mathcal{Q}_{\underline{\lambda}_k}, \pi' \rangle$, where $\Pi'_{0,k}$ denotes the partial initial state set of $s(\mathcal{Q}_{\underline{\lambda}_k})$. The structured discrete event system $\mathcal{Q}_\mu$, together with the replication sets Δ_π, $\pi \in \Pi_0$, define an *extended* structured discrete event system $\mathcal{Q}_{\mu,e}$.

3.5 Statistics of stochastic processes

The *stochastic system* construct that we define in subsection 3.6, is a structure that contains a number of extended structured discrete event system specifications, representing a main formal model, an environment of input formal models, and an experimental set-up for every such model. In addition, a stochastic system specifies the stochastic processes that retrieve information of interest from allowable replications with the abstract systems, that are induced by the structured discrete event system specifications. Eventually, certain *statistics* of these processes will constitute stochastic system *behaviour*. In the following, we formally lay down these statistics. We start with the *expected value* and *variance* of a random variable, as well as the *correlation* between two random variables. We then use these to define the *mean value function*, the *variance function* and the *correlation kernel* of a stochastic process.

In the following, all random variables are assumed to be *positive* real valued. If we speak of a random variable on a probability space, then we assume that the variable is a measurable function with respect to the σ-field of the probability space and the Borel σ-field on $\mathbf{R}$. For x a random variable on a probability space $(\Omega, \mathcal{F}, P)$, we define the *expected value of* x formally as follows,[23]

Definition 40 (Expected value)
For $(\Omega, \mathcal{F}, P)$ a probability space, and x a random variable on $(\Omega, \mathcal{F}, P)$ into $\mathbf{R}^+$, the *expected value* of x, denoted by $E[\mathrm{x}]$, is defined by

$$E[\mathrm{x}] \triangleq \int_\Omega \mathrm{x} dP \tag{3.31}$$

□

Roughly said, the expected value of a random variable has the interpretation of a sum of *realisations* of the random variable, where each realisation is weighted by the *probability* that it shows up. The *variance* of a random variable is then in a way a sum of weighted *spreads* of realisations around the expected value of the random variable. Formally,

Definition 41 (Variance)
For $(\Omega, \mathcal{F}, P)$ a probability space, and x a random variable on $(\Omega, \mathcal{F}, P)$ into $\mathbf{R}^+$, the *variance* of x, denoted by $\sigma^2[\mathrm{x}]$, is defined by

$$\sigma^2[\mathrm{x}] \triangleq \int_\Omega (\mathrm{x} - E[\mathrm{x}])^2 dP \tag{3.32}$$

□

The *covariance* (and the *correlation*) between two random variables that are defined on a common probability space, gives an indication of the extent to which realisations of one random variable deviate in the same direction from its expected value as realisations of the

[23]The integral that appears in (3.31) is called the *Lebesgue integral* of x with respect to P.

other variable do from the expected value of that random variable. In the following definition, we formally lay down the notion of covariance and correlation,

Definition 42 (Covariance and correlation)
For $(\Omega, \mathcal{F}, P)$ a probability space, and x and x' random variables on $(\Omega, \mathcal{F}, P)$ into $\mathbf{R}^+$, the *covariance* and *correlation* between x and x', denoted by $\sigma[\mathrm{x}, \mathrm{x}']$ and $\rho[\mathrm{x}, \mathrm{x}']$ respectively, are defined by

$$\sigma[\mathrm{x}, \mathrm{x}'] \triangleq \int_\Omega \mathrm{x}\mathrm{x}' dP - E[\mathrm{x}]E[\mathrm{x}'] \qquad \rho[\mathrm{x}, \mathrm{x}'] \triangleq \frac{\sigma[\mathrm{x}, \mathrm{x}']}{\sqrt{\sigma^2[\mathrm{x}]\sigma^2[\mathrm{x}']}} \tag{3.33}$$

□

For a stochastic process that consists of positive real valued random variables, the *mean value function* of the process is defined through the notion of expected value of the random variables that the process comprises. Formally,

Definition 43 (Mean value function)
For $(\Omega, \mathcal{F}, P)$ a probability space, and $\varpi \triangleq \{\mathrm{x}_i\}_{i \in I}$ a stochastic process on $(\Omega, \mathcal{F}, P)$ with positive real valued realisations, the *mean value function* of ϖ, denoted by m_ϖ, is a function from I into $\mathbf{R}^+$, defined in every $i \in I$ by

$$m_\varpi(i) \triangleq E[\mathrm{x}_i] \tag{3.34}$$

□

In a similar way, we define the *variance function* of a stochastic process as follows,

Definition 44 (Variance function)
For $(\Omega, \mathcal{F}, P)$ a probability space, and $\varpi \triangleq \{\mathrm{x}_i\}_{i \in I}$ a stochastic process on $(\Omega, \mathcal{F}, P)$ with positive real valued realisations, the *variance function* of ϖ, denoted by σ^2_ϖ, is a function from I into $\mathbf{R}^+$, defined in every $i \in I$ by

$$\sigma^2_\varpi(i) \triangleq \sigma^2[\mathrm{x}_i] \tag{3.35}$$

□

The definition of the *covariance kernel* and the *correlation kernel* of a stochastic process is similar in nature to the definition of covariance and correlation between two random variables,

Definition 45 (Covariance and correlation kernel)
For $(\Omega, \mathcal{F}, P)$ a probability space, and $\varpi \triangleq \{\mathrm{x}_i\}_{i \in I}$ a stochastic process on $(\Omega, \mathcal{F}, P)$ with positive real valued realisations, the *covariance kernel* and the *correlation kernel* of ϖ, denoted by K_ϖ and C_ϖ respectively, are functions from $I \times I$ into $\mathbf{R}$ and into $[-1, 1]$ respectively, defined in every $(i, j) \in I \times I$ by

$$K_\varpi(i, j) \triangleq \sigma[\mathrm{x}_i, \mathrm{x}_j] \qquad C_\varpi(i, j) \triangleq \rho[\mathrm{x}_i, \mathrm{x}_j] \tag{3.36}$$

□

For some stochastic processes, the mean value function and variance function show *convergent* behaviour. Also, the value of the correlation kernel tends to depend only on the absolute difference of its arguments. Such processes are called *asymptotically stationary.* Formally,

Definition 46 (Asymptotically stationary stochastic process)
For $(\Omega, \mathcal{F}, P)$ a probability space, and $\varpi \triangleq \{\mathbf{x}_i\}_{i \in I}$ a stochastic process on $(\Omega, \mathcal{F}, P)$, realisations of which are functions on an index set I into $\mathbf{R}^+$, the stochastic process ϖ is called *asymptotically stationary* iff. there exists constants μ and σ^2 in $\mathbf{R}^+$ and a function ρ from I to $[-1, 1]$ such that for all $\varepsilon \in I$ it holds that

$$\lim_{i \to +\infty} m_{\varpi}(i) \triangleq \mu \qquad \lim_{i \to +\infty} \sigma^2_{\varpi}(i) \triangleq \sigma^2 \qquad \lim_{i \to +\infty} C_{\varpi}(i, i \pm \varepsilon) \triangleq \rho(\varepsilon) \tag{3.37}$$

□

We call the constants μ and σ^2 of (3.37) the *mean* and *variance* of ϖ. We refer to the function ρ of (3.37) as the *auto-correlation function* of ϖ.

Some asymptotically stationary stochastic processes have the property that their mean, variance and auto-correlation function can be estimated by *any* realisation of the process. Such processes are called *ergodic.* Formally,

Definition 47 (Ergodic stochastic process)
For $(\Omega, \mathcal{F}, P)$ a probability space, and $\varpi \triangleq \{\mathbf{x}_i\}_{i \in I}$ an asymptotically stationary stochastic process on $(\Omega, \mathcal{F}, P)$ with mean μ, variance σ^2 and auto-correlation function ρ, the stochastic process ϖ is called *ergodic* iff. for every $\omega \in \Omega$ and $\varepsilon \in I$, it holds that

$$\lim_{n \to +\infty} \frac{1}{n} \sum_{i=1}^{n} \mathbf{x}_i(\omega) \triangleq \mu \qquad \lim_{t' \to +\infty} \frac{1}{t'} \int_0^{t'} \varpi(\omega)(t) dt \triangleq \mu \tag{3.38}$$

$$\lim_{n \to +\infty} \frac{1}{n} \sum_{i=1}^{n} (\mathbf{x}_i(\omega) - \mu)^2 \triangleq \sigma^2 \qquad \lim_{t' \to +\infty} \frac{1}{t'} \int_0^{t'} (\varpi(\omega)(t) - \mu)^2 dt \triangleq \sigma^2 \tag{3.39}$$

$$\lim_{n \to +\infty} \frac{1}{n - \varepsilon} \sum_{i=1}^{n-\varepsilon} (\mathbf{x}_i(\omega) - \mu)(\mathbf{x}_{i+\varepsilon}(\omega) - \mu) \triangleq \rho(\varepsilon) \tag{3.40}$$

$$\lim_{t' \to +\infty} \frac{1}{t' - \varepsilon} \int_0^{t'-\varepsilon} (\varpi(\omega)(t) - \mu)(\varpi(\omega)(t + \varepsilon) - \mu) dt \triangleq \rho(\varepsilon) \tag{3.41}$$

□

The LHS of equations (3.38) and (3.39), and equation (3.40) hold in case ϖ is a *discrete* stochastic process, realisations of which are defined on an index set $I \triangleq \{1, 2, \ldots\}$. The RHS of equations (3.38) and (3.39), and equation (3.41) hold in case ϖ is a *continuous* stochastic process, realisations of which are defined on an index set $I \triangleq [0, +\infty[$.

3.6 The stochastic system construct

For $\mathcal{Q}$ and $\mathcal{Q}'$ random structured discrete event systems, we call the pair $\langle\mathcal{Q}', \mathcal{Q}\rangle$ an *autonomous coupled (structured discrete event) system* if and only if the external event set of $\mathcal{Q}'$ equals $\{\chi_\phi\}$, and the output set of $\mathcal{Q}'$ coincides with the external event set of $\mathcal{Q}$. Thus, very simply, an autonomous coupled system contains one structured discrete event system, operating autonomously, that generates external events for another system. For $\langle\mathcal{Q}', \mathcal{Q}\rangle$ an autonomous coupled system, we define a *compatibility relation for* $\langle\mathcal{Q}', \mathcal{Q}\rangle$, denoted by F, as a relation between the partial initial state sets Π_0' and Π_0 of $s(\mathcal{Q}')$ and $s(\mathcal{Q})$. We say that $\mathcal{Q}$ is *coupled* to $\mathcal{Q}'$ *through* F, and also that $\langle\mathcal{Q}', \mathcal{Q}\rangle$ is an autonomous F-coupled system. We think of a compatibility relation as part of an experimental set-up. Given a partial initial state π' of $s(\mathcal{Q}')$, the compatibility relation for $\langle\mathcal{Q}', \mathcal{Q}\rangle$ tells us in which partial initial state we are *allowed* to place $s(\mathcal{Q})$ in, when we want to use the output segment in the outcome of a replication with $s(\mathcal{Q}')$ in π' as the input segment in some replication with $s(\mathcal{Q})$. In the context of the coin striking example in this research report, the reader can safely define a compatibility relation by $F \triangleq \Pi_0' \times \Pi_0$ for every autonomous coupled system $\langle\mathcal{Q}_{\underline{\lambda}}, \mathcal{Q}_\mu\rangle$, where $\mathcal{Q}_{\underline{\lambda}}$ and $\mathcal{Q}_\mu$ are the systems that we developed above, choosing particular values for the parameters $\underline{\lambda}$ and μ, and where Π_0' and Π_0 are the partial initial state sets of $s(\mathcal{Q}_{\underline{\lambda}})$ and $s(\mathcal{Q}_\mu)$. The case that we have in mind in which an arbitrary partial initial state of an *input* abstract system is *not* compatible to every partial initial state of a *main* abstract system, has to do with the airline network that we briefly introduced in section 1, and for which we developed a simulation model in [1]. Stated somewhat informally, we intend in that case to let a (small) main structured discrete event system of the central station or the hub in the network, be a *segment* of a (big) structured discrete event system of the entire airline network. Then, once we pinned down a partial initial state of the abstract system that is generated by the system of the network, it is only reasonable that we choose a partial initial state of the abstract system that is generated by the system of the hub that 'complies' with the first partial initial state.

In the former, we introduced the idea of replications with an abstract system in a partial initial state. We now define a *replication-realisation* function in the following definition,

Definition 48 (Replication-realisation function)
For $\mathcal{Q}_e \triangleq \langle\mathcal{Q}, \{\Delta_\pi\}_{\pi\in\Pi_0}\rangle$ an extended structured discrete event system, with the abstract system generated by $\mathcal{Q}$ denoted by $s(\mathcal{Q}) \triangleq \langle T, X, \Omega, Y, \Sigma_0, \delta, \lambda\rangle$, and π a partial initial state in Π_0, the *replication-realisation function of* $s(\mathcal{Q})$ *in* π, denoted by φ_π, is a function from Δ_π into $\mathscr{S}(T, Y)$, defined in every $(r, \omega) \in \Delta_\pi$, with $\mathrm{dom}(\omega) \triangleq\,]t_0, t_1]$, by

$$\varphi_\pi(r,\omega)(t) \triangleq \begin{cases} \lambda \circ \delta((\pi, r), \omega_{t]}) & t_0 < t < t_1 \\ \lambda \circ \delta((\pi, r), \omega) & t = t_1 \end{cases} \tag{3.42}$$

□

In case $\langle \mathcal{Q}', \mathcal{Q} \rangle$ is an autonomous F-coupled system, then we define a replication-realisation function for *every* partial initial state of $s(\mathcal{Q}')$, and a replication-realisation function for *every* partial initial state of $s(\mathcal{Q})$. However, with regard to $s(\mathcal{Q})$, we intend to define also a *conditional replication-realisation function* for *every* partial initial state of $s(\mathcal{Q})$, given $\mathcal{Q}'$ and any partial initial state π' of $s(\mathcal{Q}')$ such that π' is compatible to π through F.[24] For π and π' partial initial states of $s(\mathcal{Q})$ and $s(\mathcal{Q}')$ such that $\pi'[F]\pi$, we denote the conditional replication-realisation function of $s(\mathcal{Q})$, conditional on $\mathcal{Q}'$ and π', by $\varphi_\pi \lhd \langle \mathcal{Q}', \pi' \rangle$. The idea of a conditional replication-realisation function is the following. Given that we generate input segments by performing replications with the abstract system that is induced by $\mathcal{Q}'$, set up in a partial initial state π', we perform *only finitely many* of all the replications that we can perform with $s(\mathcal{Q})$ in π. Recall that *all* the replications that we are allowed to perform with $s(\mathcal{Q})$ in π are gathered in the replication set Δ_π. To identify the replications that we allow ourselves to perform with $s(\mathcal{Q})$ in π, once we have selected $\mathcal{Q}'$ and π', we introduce the notion of a *replication coupling* - that we define for all compatible partial initial states π' of $s(\mathcal{Q}')$ and π of $s(\mathcal{Q})$. For π' and π partial initial states, we specify such a replication coupling as a 1-1 function from the replication set $\Delta'_{\pi'}$ of $s(\mathcal{Q}')$ onto a finite subset of the replication set Δ_π of $s(\mathcal{Q})$, and formally define it as follows,

Definition 49 (Replication coupling)
For $\mathcal{Q}_e \triangleq \langle \mathcal{Q}, \{\Delta_\pi\}_{\pi \in \Pi_0} \rangle$ and $\mathcal{Q}'_e \triangleq \langle \mathcal{Q}', \{\Delta'_{\pi'}\}_{\pi' \in \Pi'_0} \rangle$ extended structured discrete event systems such that $\langle \mathcal{Q}', \mathcal{Q} \rangle$ is an autonomous coupled system, F a compatibility relation for $\langle \mathcal{Q}', \mathcal{Q} \rangle$, and π' and π compatible partial initial states of Π'_0 and Π_0 respectively, a *replication coupling for $\mathcal{Q}'_e$ and $\mathcal{Q}_e$ in π' and π*, denoted by $\xi_\pi \lhd \langle \mathcal{Q}', \pi' \rangle$, is a 1-1 function from $\Delta'_{\pi'}$ onto a finite subset of Δ_π such that

$$(r', \Lambda) \overset{\xi_\pi \lhd \langle \mathcal{Q}', \pi' \rangle}{\longmapsto} (r, \omega) \Rightarrow \varphi'_{\pi'}(r', \Lambda) = \omega \tag{3.43}$$

□

Everything as in definition 49, in case the replication coupling $\xi_\pi \lhd \langle \mathcal{Q}', \pi' \rangle$ maps a replication (r', Λ) in $\Delta'_{\pi'}$ on the replication (r, ω) in Δ_π, then ω must equal the output segment in the outcome of the replication (r', Λ) with $s(\mathcal{Q}')$ in π', as given by the replication-realisation function $\varphi'_{\pi'}$ of $s(\mathcal{Q}')$ in π'. We call the range of the coupling $\xi_\pi \lhd \langle \mathcal{Q}', \pi' \rangle$ a *conditional replication set*, and denote it by $\Delta_\pi \lhd \langle \mathcal{Q}', \pi' \rangle$. In the following definition, we use the idea of a replication coupling to formally pin down the notion of a *conditional replication realisation function*,

[24]For $\mathscr{E} \triangleq \{\mathcal{Q}'_k\}_{k \in K}$ an environment of systems for $\mathcal{Q}$, and $\langle \mathcal{Q}'_k, \mathcal{Q} \rangle$ an autonomous F_k-coupled system for every $k \in K$, we thus intend to define a conditional replication-realisation function for every $k \in K$, and compatible partial initial states of $\Pi'_{0,k}$ and Π_0, where $\Pi'_{0,k}$ is the partial initial state set of $s(\mathcal{Q}'_k)$ for all $k \in K$, and where Π_0 is the partial initial state set of $s(\mathcal{Q})$.

Definition 50 (Conditional replication-realisation function)
For $\mathcal{Q}_e \triangleq \langle \mathcal{Q}, \{\Delta_\pi\}_{\pi \in \Pi_0} \rangle$ and $\mathcal{Q}'_e \triangleq \langle \mathcal{Q}', \{\Delta'_{\pi'}\}_{\pi' \in \Pi'_0} \rangle$ extended structured discrete event systems such that $\langle \mathcal{Q}', \mathcal{Q} \rangle$ is an autonomous coupled system, with the abstract system generated by $\mathcal{Q}$ denoted by $s(\mathcal{Q}) \triangleq \langle T, X, \Omega, Y, \Sigma_0, \delta, \lambda \rangle$, F a compatibility relation for $\langle \mathcal{Q}', \mathcal{Q} \rangle$, π' and π compatible partial initial states of Π'_0 and Π_0 respectively, and $\xi_\pi \triangleleft \langle \mathcal{Q}', \pi' \rangle$ a replication coupling for $\mathcal{Q}'_e$ and $\mathcal{Q}_e$ in π' and π, the *conditional replication-realisation function of* $s(\mathcal{Q})$ *in* π, *conditional on* $\mathcal{Q}'$ *and* π', denoted by $\varphi_\pi \triangleleft \langle \mathcal{Q}', \pi' \rangle$, is a function from the range $\Delta_\pi \triangleleft \langle \mathcal{Q}', \pi' \rangle$ of $\xi_\pi \triangleleft \langle \mathcal{Q}', \pi' \rangle$ into $\mathscr{S}(T, Y)$, defined in every $(r, \omega) \in \Delta_\pi \triangleleft \langle \mathcal{Q}', \pi' \rangle$ by

$$\varphi_\pi \triangleleft \langle \mathcal{Q}', \pi' \rangle \triangleq [\varphi_\pi]_{|\Delta_\pi \triangleleft \langle \mathcal{Q}', \pi' \rangle} \tag{3.44}$$

□

We intend to define a number of *information functions* to distillate *information* from output segments, that is of interest with regard to *primary model behaviour.* We speak of an *input* information function, in case the function is defined on the range of a replication-realisation function of an abstract system, generated by a structured discrete event system in an *environment.* We speak of an *output* information function, when the function is defined on the range of a *conditional* replication-realisation function of an abstract system, that is generated by a *main* structured discrete event system. In any case, we formally define an information function as a function into the real line that attains only positive real values. We use the concept of information functions to define a *stochastic information process* as follows,

Definition 51 (Stochastic input and output information process)
For $\mathcal{Q}_e \triangleq \langle \mathcal{Q}, \{\Delta_\pi\}_{\pi \in \Pi_0} \rangle$ and $\mathcal{Q}'_e \triangleq \langle \mathcal{Q}', \{\Delta'_{\pi'}\}_{\pi' \in \Pi'_0} \rangle$ extended structured discrete event systems with finite replication sets $\Delta'_{\pi'}$, $\pi' \in \Pi'_0$ such that $\langle \mathcal{Q}', \mathcal{Q} \rangle$ is an autonomous coupled system, F a compatibility relation for $\langle \mathcal{Q}', \mathcal{Q} \rangle$, π' and π compatible partial initial states of Π'_0 and Π_0 respectively, $\{\mathbf{x}_{\pi',i}\}_{i \in I}$ and $\{\mathbf{y}_{\pi,j} \triangleleft \langle \mathcal{Q}', \pi' \rangle\}_{j \in J}$ indexed families of information functions on the range of $\varphi'_{\pi'}$ and $\varphi_\pi \triangleleft \langle \mathcal{Q}', \pi' \rangle$ respectively into $\mathbb{R}^+$, and $P_U^{\Delta'_{\pi'}}$ and $P_U^{\Delta_\pi \triangleleft \langle \mathcal{Q}', \pi' \rangle}$ probability measures on $2^{\Delta'_{\pi'}}$ and $2^{\Delta_\pi \triangleleft \langle \mathcal{Q}', \pi' \rangle}$ respectively, defined by $P_U^{\Delta'_{\pi'}}(\{(r', \Lambda)\}) \triangleq \frac{1}{|\Delta'_{\pi'}|}$ for all $\{(r', \Lambda)\} \in 2^{\Delta'_{\pi'}}$, and $P_U^{\Delta_\pi \triangleleft \langle \mathcal{Q}', \pi' \rangle}(\{(r, \omega)\}) \triangleq \frac{1}{|\Delta_\pi \triangleleft \langle \mathcal{Q}', \pi' \rangle|}$ for all $\{(r, \omega)\} \in 2^{\Delta_\pi \triangleleft \langle \mathcal{Q}', \pi' \rangle}$, the stochastic processes $\{\mathbf{x}_{\pi',i} \circ \varphi'_{\pi'}\}_{i \in I}$ and $\{\mathbf{y}_{\pi,j} \triangleleft \langle \mathcal{Q}', \pi' \rangle \circ \varphi_\pi \triangleleft \langle \mathcal{Q}', \pi' \rangle\}_{j \in J}$ are called a *stochastic input* and a *stochastic output information process* respectively.

□

With the definitions of (conditional) replication-realisation functions, replication couplings and stochastic input and output information processes, we are now ready to give a formal definition of a *stochastic system*,[25]

[25]In equation (3.48), the asterisks stand for

$$*_1 \triangleq \pi' \in \Pi'_{0,k},\ \pi \in \Pi_0 \text{ s.t. } \pi'[F_k]\pi,\ k \in K \tag{3.45}$$

$$*_2 \triangleq \pi' \in \Pi'_{0,k},\ k \in K,\ i \in I \tag{3.46}$$

$$*_3 \triangleq \pi' \in \Pi'_{0,k},\ \pi \in \Pi_0 \text{ s.t. } \pi'[F_k]\pi,\ k \in K,\ j \in J \tag{3.47}$$

Definition 52 (Stochastic system)
For $\mathcal{Q}_e \triangleq \langle \mathcal{Q}, \{\Delta_\pi\}_{\pi \in \Pi_0} \rangle$ and $\mathcal{Q}'_{e,k} \triangleq \langle \mathcal{Q}'_k, \{\Delta'_{\pi',k}\}_{\pi' \in \Pi'_{0,k}} \rangle$, $k \in K$ extended structured discrete event systems with finite replication sets $\Delta'_{\pi',k}$, $\pi' \in \Pi'_{0,k}$, $k \in K$, indexed by an arbitrary index set K, such that for every $k \in K$ the pair $\langle \mathcal{Q}'_k, \mathcal{Q} \rangle$ is an autonomous F_k-coupled system, $\xi_\pi \lhd \langle \mathcal{Q}'_k, \pi' \rangle$ a replication coupling for $\mathcal{Q}'_{k,e}$ and $\mathcal{Q}_e$ in π' and π for every $\pi' \in \Pi'_{0,k}$, $\pi \in \Pi_0$ s.t. $\pi'[F_k]\pi$ and $k \in K$, $\varpi_{\pi',k,i}$, $i \in I$ stochastic input information processes on $(\Delta'_{\pi',k}, 2^{\Delta'_{\pi',k}}, P_U^{\Delta'_{\pi',k}})$ for every $\pi' \in \Pi'_{0,k}$, $k \in K$, indexed by a finite index set I, and $\varsigma_{\pi,j} \lhd \langle \mathcal{Q}'_k, \pi' \rangle$, $j \in J$ stochastic output information processes on $(\Delta_\pi \lhd \langle \mathcal{Q}'_k, \pi' \rangle, 2^{\Delta_\pi \lhd \langle \mathcal{Q}'_k, \pi' \rangle}, P_U^{\Delta_\pi \lhd \langle \mathcal{Q}'_k, \pi' \rangle})$ for every $\pi' \in \Pi'_{0,k}$, $\pi \in \Pi_0$ s.t. $\pi'[F_k]\pi$ and $k \in K$, indexed by a finite index set J, a *stochastic system* or *SS* is a 6-tuple, defined by

$$\mathcal{K} \triangleq \langle \mathcal{Q}_e, \{\mathcal{Q}'_{k,e}\}_{k \in K}, \{F_k\}_{k \in K}, \{\xi_\pi \lhd \langle \mathcal{Q}'_k, \pi' \rangle\}_{*_1}, \{\varpi_{\pi',k,i}\}_{*_2}, \{\varsigma_{\pi,j} \lhd \langle \mathcal{Q}'_k, \pi' \rangle\}_{*_3} \rangle \tag{3.48}$$

□

Some of the components of a stochastic system $\mathcal{K}$ specify a number of formal models and stipulate the experiments that we allow ourselves to perform with these models. Other components of the stochastic system lay down what kind of information we whish to retrieve from these experiments.

It follows readily from definition 52 that a stochastic system induces a *relation* between a set, members of which are tuples of *statistics* of stochastic *input* information processes, and a set, members of which are tuples of *statistics* of stochastic *output* information processes. To be more precise, for every index $k \in K$, partial initial state $\pi' \in \Pi'_{0,k}$ and compatible partial initial state $\pi \in \Pi_0$, we relate a tuple with the mean value function, the variance function and the correlation kernel of every of the stochastic input information processes $\varpi_{\pi',k,i}$, $i \in I$ to a tuple with similar such statistics of every of the stochastic output information processes $\varsigma_{\pi,j} \lhd \langle \mathcal{Q}'_k, \pi' \rangle$, $j \in J$. Clearly, such a relation contains an *overload* of information, and forms a direct practical problem if we want to retain the relation as *behaviour* of a stochastic system and further *approximate* this behaviour with the fuzzy neural network that we discuss in section 4. Therefore, we introduce in the following a number of *conditions* that must be fulfilled by a stochastic system for it be validatable with our fuzzy set theory based validation approach.

A *first* condition that a stochastic system must satisfy for it to be validatable, concerns the fact that we require that the mean value function and the variance function of every stochastic information process of the system shows *convergent* behaviour. In addition, we require that the value of the correlation kernel of every process tends to depend only on the absolute difference between its arguments. We thus want to replace the statistics *mean value function*, *variance function* and *correlation kernel* of every stochastic information process of a stochastic system, by the statistics *mean*, *variance* and *auto-correlation function* of an *asymptotically stationary* stochastic process. A *second* condition comes down to the fact that we whish to be able to make abstraction of partial initial states. For every stochastic information process, we desire that the convergence of its mean value function, variance function and correlation kernel is *independent* of the partial initial state(s) that we choose. Stated somewhat informally, we desire that the mean, variance and auto-correlation function of every stochastic information process is independent of the partial initial states that identify the process.[26] In a *third* and final condition, we require - again stated somewhat informally here -, that we are able to compute estimates of the mean, variance and auto-correlation function of a stochastic information process from a single (very long) realisation of that process. A complete formal development of these conditions goes beyond the research goals of this report. We refer in the following to a stochastic system that satisfies the above conditions, as an *approximately stationary, steady-state and ergodic* stochastic system.

Recall that we postulated in section 1 behaviour of a *primary model* as a relation between the Cartesian product of the range of *independent* variables of interest, and the Cartesian product of the range of *dependent* variables of interest. As we identify a primary model with a stochastic system, we formally pin down in the following definition the notion of an *independent variable* and a *dependent variable* of interest of a primary model,[27]

Definition 53 (Independent and dependent variable)
For $\mathcal{K} \triangleq \langle \mathcal{Q}_e, \{\mathcal{Q}'_{k,e}\}_{k\in K}, \{F_k\}_{k\in K}, \{\xi_\pi \triangleleft \langle \mathcal{Q}'_k, \pi' \rangle\}_{*_1}, \{\varpi_{\pi',k,i}\}_{*_2}, \{\varsigma_{\pi,j} \triangleleft \langle \mathcal{Q}'_k, \pi' \rangle\}_{*_3} \rangle$ an approximately stationary, steady-state and ergodic stochastic system with mean, variance and auto-correlation function $\mu^x_{k,i}$, $\sigma^{2^x}_{k,i}$ and $\rho^x_{k,i}$ attributed to the input information process $\varpi_{\pi',k,i}$ for every $\pi' \in \Pi'_{0,k}$, $i \in I \triangleq \{i_1, i_2, \ldots, i_m\}$ and $k \in K$, and with mean, variance and auto-correlation function $\mu^y_{k,j}$, $\sigma^{2^y}_{k,j}$ and $\rho^y_{k,j}$ attributed to the output information process $\varsigma_{\pi,j} \triangleleft \langle \mathcal{Q}'_k, \pi' \rangle$ for every $\pi' \in \Pi'_{0,k}$, $\pi \in \Pi_0$ s.t. $\pi'[F_k]\pi$, $j \in J \triangleq \{j_1, j_2, \ldots, j_n\}$ and $k \in K$, an *independent variable for* $\mathcal{K}$ and a *dependent variable for* $\mathcal{K}$ are functions from $\bigcup_{k\in K}\{((\mu^x_{k,i_1}, \sigma^{2^x}_{k,i_1}, \rho^x_{k,i_1}), (\mu^x_{k,i_2}, \sigma^{2^x}_{k,i_2}, \rho^x_{k,i_2}), \ldots, (\mu^x_{k,i_m}, \sigma^{2^x}_{k,i_m}, \rho^x_{k,i_m}))\}$ into $\mathbb{R}$ and from $\bigcup_{k\in K}\{((\mu^y_{k,j_1}, \sigma^{2^y}_{k,j_1}, \rho^y_{k,j_1}), (\mu^y_{k,j_2}, \sigma^{2^y}_{k,j_2}, \rho^y_{k,j_2}), \ldots, (\mu^y_{k,j_n}, \sigma^{2^y}_{k,j_n}, \rho^y_{k,j_n}))\}$ into $\mathbb{R}$ respectively.

$\square$

In the context of the simple coin striking example, assuming that I is a singleton-set, we may want to define an independent variable on $A \triangleq \{(\mu^x_k, \sigma^{2^x}_k, \rho^x_k)\}_{k\in K}$ that for every $k \in K$ simply projects the triple $(\mu^x_k, \sigma^{2^x}_k, \rho^x_k)$ of A on the mean inter-arrival time μ^x_k. Likewise, assuming that J is a singleton-set, we may want to define a dependent variable on $B \triangleq \{(\mu^y_k, \sigma^{2^y}_k, \rho^y_k)\}_{k\in K}$ that for every $k \in K$ maps the triple $(\mu^y_k, \sigma^{2^y}_k, \rho^y_k)$ of B on a value that indicates the short-term temporal correlation between queue length observations, computed from *filtering* out the correlations of ρ^y_k at the higher lags, and *aggregating* the remaining correlations at the lower lags. We now use the formal definition of an independent and a dependent variable to define a *behaviour orientation* for a primary model as follows,

[26]This statement is an *informal* statement, since we cannot speak - in the strict sense - of the mean, variance and auto-correlation function of a stochastic information process. All that we can do, is *attribute* such statistics (of an hypothetical asymptotically stationary stochastic process) to a stochastic information process. The major reason for this is that realisations of a stochastic information process are always *finite* in length.

[27]See equations (3.45), (3.46) and (3.47) for a definition of the asterisk subscripts.

Definition 54 (Behaviour orientation)
For $\mathscr{X}$ and $\mathscr{Y}$ finite indexed sets of independent and dependent variables for an approximately stationary, steady-state and ergodic stochastic system $\mathcal{K}$, the pair $\nabla \triangleq (\mathscr{X}, \mathscr{Y})$ is called a *behaviour orientation for* $\mathcal{K}$.

□

Assuming that the single independent and dependent variable that we defined above for the simple coin striking process, are the only variables of interest that we want to define, then we obtain a *relation* from a stochastic system $\mathcal{K}$ of the coin striking process *through* the *orientation* that is set up by the variables. By that, we simply mean that, for every index $k \in K$, we relate the mean inter-arrival time μ_k^x to the filtered auto-correlation function $f(\rho_k^y)$, where f is some predetermined filter that blocks correlations at higher lags, and integrates correlations at lower lags in a single real value. We call this relation the *behaviour relation of* $\mathcal{K}$ *through the orientation* $(\mathscr{X}, \mathscr{Y})$, where $\mathscr{X}$ and $\mathscr{Y}$ are singleton sets with the independent and dependent variable of interest respectively. We formally pin down the notion of behaviour of a stochastic system through a behaviour orientation in the following definition,

Definition 55 (Behaviour of a stochastic system)
For $\mathcal{K}$ an approximately stationary, steady-state and ergodic stochastic system, and $\nabla \triangleq (\mathscr{X}, \mathscr{Y})$ a behaviour orientation for $\mathcal{K}$, the *behaviour of* $\mathcal{K}$ *through* ∇, denoted by $\mathscr{B}(\mathcal{K})_{|\nabla}$, is a relation between $\mathbf{R}^{|\mathscr{X}|}$ and $\mathbf{R}^{|\mathscr{Y}|}$, derived by applying the independent and dependent variables in $\mathscr{X}$ and $\mathscr{Y}$ at every tuple with means, variances and auto-correlation functions that were attributed to stochastic input and output information processes of $\mathcal{K}$.

□

We have now come full circle. In postulate 9, we postulated primary model behaviour as a relation between the Cartesian product of the range of independent variables of interest, and the Cartesian product of the range of dependent variables of interest. Throughout sections 2 and 3, we developed a stochastic system construct, and formulated a triple of conditions that a stochastic system must fulfill for it be validatable. We now see, once we have set up a behaviour orientation for a validatable stochastic system $\mathcal{K}$, that the system induces a behaviour relation through the behaviour orientation. As we identify stochastic systems with primary models, we have thus pinned down one of the arguments of our validation approach, i.e. primary model behaviour - recall figure 2. The opposing argument, i.e. real system behaviour, can be identified in a completely similar way. For, if $\mathcal{A}_{|.}$ stands for a family of relations with gathered input/output segments pairs from a real system, then we postulate that there is a stochastic system $\mathcal{K}'$ that induces a family of input/output segment relations that contain all of the input/output segment pairs that we assembled in $\mathcal{A}_{|.}$.[28] In the *hypothetical* case that we *do* have the system $\mathcal{K}'$ available, then we can set up an orientation ∇' for $\mathcal{K}'$, that is *compatible* to the orientation that we set up for $\mathcal{K}$. By *compatible*, we mean that independent and dependent

[28]The postulate that we formulate here is in a way a stochastic system variant of the notion of a *base model* - an hypothetical and ideal model - in systems theory [31].

variables of both orientations measure the same performance measures of interest. We whish to agree here to identify real system behaviour with the behaviour of $\mathcal{K}'$ through the orientation ∇'.

As a final remark in light of our agreement on the nature of primary model and real system behaviour, notice that *neither* primary model behaviour *nor* real system behaviour will be known *exactly* in practice. For, in order to pinpoint the former, we must have asymptotically stationary and ergodic stochastic processes available. All that we *can* do to *estimate* statistics of these processes is to use the *approximate* such processes of $\mathcal{K}$ themselves. Similarly, in order to pinpoint the latter, we must have arbitrarily long segments in input/output segment pairs of $\mathcal{A}_{|.}$ in order to compute limits of averages of stochastic sums and integrals, and we must have all input/output segment pairs that are induced by $\mathcal{K}'$. All that we *can* do to *estimate* these limits is to use input/output segment pairs of $\mathcal{A}_{|.}$, with segments that are as long as possible. In summary, all that we can hope for is to have an *estimate* of real system behaviour and an *estimate* of primary model behaviour. Such estimates constitute the practical operands of our fuzzy set theory based validation method, that we expose in the following section.

4 A resemblance approach to behaviour validation

In this section, we present our fuzzy set theoretic validation approach to validate stochastic systems. In that respect, we assume that we have constructed an approximately stationary, steady-state and ergodic stochastic system $\mathcal{K}$ on the one hand, and that we have gathered a family of input/output segment relations $\mathcal{A}_{|\cdot}$ - that we refer to as an *abstract object trace* in the following - on the other hand, that we postulate to come from an (unknown) approximately stationary, steady-state and ergodic stochastic system $\mathcal{K}'$. Further, we assume that we have obtained *estimates* $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla}$ and $\hat{\mathscr{B}}(\mathcal{K}')_{|\nabla'}$ - through simulation and time-based arithmetic - of the behaviour of $\mathcal{K}$ and $\mathcal{K}'$ through compatible orientations ∇ and ∇' of interest. The problem is then in a way to establish a measure of *similarity* between these behaviour relation estimates. In that respect, the *more* similar the behaviour relation estimates to one another, the *more* valid we call $\mathcal{K}$ in view of $\mathcal{A}_{|\cdot}$. Our validation method employs a *resemblance relation* concept in fuzzy set theory, that was first introduced by [6], and a neuro-fuzzy learning algorithm NEFPROX, developed by [21].

In subsection 4.1, we state some of the fundamentals of fuzzy sets and approximate reasoning. The list of references on basic aspects of fuzzy set theory and its applications is huge - see e.g. [13, 9, 33, 29] for an introduction to fuzzy set theory and approximate reasoning. Then, we briefly explain in subsection 4.2 the fuzzy learning algorithm that we employ to induce a fuzzy rule base from a behaviour relation estimate, and we introduce the concept of a behaviour relation estimate approximation. We have chosen the NEFPROX algorithm, based on the satisfactory performance that we found in previous experiments that we carried out for a classification-variant of the algorithm [18]. In subsection 4.3, we state the concept of resemblance relations, and we develop a notion of *resemblance random variables* in subsection 4.4. We discuss how we intend to use the cumulative distribution of resemblance random variables in subsection 4.5 to construct fuzzy sets of propositions on the validity of a stochastic system in view of an abstract object trace.

4.1 Fuzzy set and approximate reasoning fundamentals

In *classic* set theory, a set A is defined as a collection of items, called *members* or *elements* of A. In case A is a subset of a universe X, then the set A can be completely identified by its *indicator function* 1_A, that runs from X into $\mathbf{R}$, and assigns to every x in X the value 1 in case x is a member of A, and the value 0 otherwise. Clearly, the range of 1_A equals $\{0, 1\}$. In *fuzzy* set theory, the range of 1_A is in a way extended to the unit interval $[0, 1]$. Basically, fuzzy set theory allows us to define a *grade* of membership to a set.

Let μ_A be a function from X into $\mathbf{R}$ with range $[0, 1]$. In case $\mu_A(x) \approxeq 1$ for $x \in X$, then we say that x *strongly* belongs to A. Similarly, if $\mu_A(x) \approxeq 0$ for $x \in X$, then we say that x *weakly* belongs to A. The set of all pairs $(x, \mu_A(x))$, $x \in X$ is called a *fuzzy set*, and μ_A is called the *membership function* of the fuzzy set. Formally,

Definition 56 (Fuzzy set)
For X an arbitrary non-empty set, and μ_A a function from X into $\mathbb{R}$ with range $[0,1]$, the set $A \triangleq \{(x, \mu_A(x))\}_{x \in X}$ is called a *fuzzy set on X with membership function μ_A*.

□

We also say that A is a *fuzzy subset* of X. Notice that a fuzzy set is fully characterised by its membership function. To alleviate the notation, we denote $\mu_A(x)$ by $A(x)$ for all $x \in X$. Sometimes, we use the term *crisp set* to emphasise that a set is a non-fuzzy set.

An *α-cut*, for $\alpha > 0$, of a fuzzy set A on X, denoted by A^α, is the set of all members of X that have a grade of membership of at least α to A, or $A^\alpha \triangleq \{x \mid A(x) \geq \alpha\}_{x \in X}$. The *core* of a fuzzy set is defined by $\mathrm{core}(A) \triangleq A^1$, while its *support*, denoted by $\mathrm{supp}(A)$, is defined as the set of all members of X that have a strictly positive degree of membership to A.

In classic set theory, the set operators *intersection* and *union* are unanimously defined. In fuzzy set theory, a collection of alternative implementations of these operators is available. The intersection and union of fuzzy sets are defined with the help of so-called *triangular norms* and *triangular conorms*. A triangular norm and a triangular conorm are formally defined as follows,

Definition 57 (Triangular norm and co-norm)
A pair of functions $\top$ and $\bot$ from $[0,1]^2$ into $[0,1]$ are called a *triangular norm* or *t-norm*, and a *triangular conorm* or *t-conorm* respectively, iff. for every $x, y, z \in [0,1]$, it holds that

$$\top(x,1) = x \qquad \bot(x,0) = x \tag{4.49}$$
$$\top(x,y) = \top(y,x) \qquad \bot(x,y) = \bot(y,x) \tag{4.50}$$
$$\top(x,\top(y,z)) = \top(\top(x,y),z) \qquad \bot(x,\bot(y,z)) = \bot(\bot(x,y),z) \tag{4.51}$$
$$x \leq y \Rightarrow \top(x,z) \leq \top(y,z) \qquad x \leq y \Rightarrow \bot(x,z) \leq \bot(y,z) \tag{4.52}$$

□

Although, strictly speaking, a t-norm and a t-conorm require a pair of arguments to be applied, they can easily be extended to *more* than two arguments. In that respect, for A_i, $i \in I$ fuzzy subsets of a universe X, indexed by a finite index set $I \triangleq \{1, 2, \ldots, n\}$, we use the notation $\top_{i \in I}\{A_i(x)\}$ for every $x \in X$ to denote

$$\top_{i \in I}\{A_i(x)\} \triangleq \top[A_1(x), (\top(A_2(x), \ldots, A_n(x)))] \tag{4.53}$$

A similar agreement in notation applies for $\bot_{i \in I}\{A_i(x)\}$ for every $x \in X$. T-norms and t-conorms are used in fuzzy set theory to model the logical connectives *and* and *or* respectively.

With the help of the notion of a t-norm and a t-conorm, we define the intersection and union of two fuzzy sets as follows,

Definition 58 (Intersection and union of fuzzy sets)
For X an arbitrary non-empty set, A and B fuzzy subsets of X, and $\top$ and $\perp$ a t-norm and a t-conorm respectively, the *intersection* of A and B, denoted by $A \wedge B$, and the *union* of A and B, denoted by $A \vee B$, are defined by

$$A \wedge B \triangleq \{(x, \top[A(x), B(x)])\}_{x \in X} \qquad A \vee B \triangleq \{(x, \perp[A(x), B(x)])\}_{x \in X} \tag{4.54}$$

□

In the following, we use Zadeh's t-norm and t-conorm to model the intersection and union of two fuzzy sets. In that respect, one can replace in (4.54), the t-norm $\top$ and the t-conorm $\perp$ by the *min* and *max* operator respectively.

In classic set theory, a relation R between two sets X and Y is defined as a set of pairs of the form (x, y) with $x \in X$ and $y \in Y$, for which holds that the statement x *is related to* y is true. Clearly, R is a subset of $X \times Y$. A *fuzzy relation* R between two crisp sets X and Y is a fuzzy subset of $X \times Y$. The degree of membership of a pair (x, y) to R, with $x \in X$ and $y \in Y$, indicates the extent to which the statement x *is related to* y is true.

In the usual case, we interpret a fuzzy relation as a *fuzzy rule* or a *fuzzy implication*. The degree of membership of a point (x, y) to the fuzzy rule gives an indication of the extent to which the point complies with the rule. Stated otherwise, it gives an indication of the truth value of the statement x *implies* y. Although we can construct a fuzzy rule *directly* by specifying a fuzzy relation, the problem is usually to *derive* from a fuzzy set A on a universe X and a fuzzy set B on a universe Y, a fuzzy subset of $X \times Y$ that adequately models the implication *if* x *is* A *then* y *is* B.[29] For (a, b) a point of $X \times Y$, how should we define the membership value of the implication *if* x *is* A *then* y *is* B at the pair (a, b)? This question comes down to which fuzzy set we should install on $X \times Y$ to model the rule *if* x *is* A *then* y *is* B, given the fuzzy sets A and B. It brings us at the theory of *approximate reasoning*.

In the theory of approximate reasoning, one operation that involves fuzzy sets, deserves special attention, i.e. the *composition* of a fuzzy set with a fuzzy relation. We define the composition of a fuzzy set with a fuzzy relation with the help of the notion of the *projection* of a fuzzy relation on a (crisp) set, and the *extension* of a fuzzy set into the Cartesian product of two (crisp) sets. Formally,

[29]The degree of truth of the statement x *is* A equals the membership value of x to A. Similarly, the truth value of y *is* B equals the membership value of y to B.

Definition 59 (Projection of fuzzy relation and extension of fuzzy set)
For X and Y arbitrary non-empty sets, A a fuzzy set on X, and R a fuzzy relation between X and Y, the *projection of R on X*, denoted by $\text{proj}_X(R)$, and the *extension of A into $X \times Y$*, denoted by $\text{ext}_Y(A)$, are defined by

$$\text{proj}_X(R) \triangleq \{(x,\alpha) \mid \alpha = \sup_{y\in Y}\{R(x,y)\}\}_{x\in X} \qquad \text{ext}_Y(A) \triangleq \{((x,y),A(x)) \mid (x,y) \in X \times Y\} \tag{4.55}$$

□

Using definition (59), we define the composition of a fuzzy set A on X with a fuzzy relation R between X and Y as follows,

Definition 60 (Composition of fuzzy set with fuzzy relation)
For X and Y arbitrary non-empty sets, A a fuzzy set on X, R a fuzzy relation between X and Y, and $\top$ a t-norm, the *composition of A with R*, denoted by $A \cdot R$, is a fuzzy set on Y, defined by

$$A \cdot R \triangleq \text{proj}_Y(R \wedge \text{ext}_Y(A)) \tag{4.56}$$

□

Everything as in definition 60, the composition of A with R yields a fuzzy set on Y. The composition of (4.56) is also called the *sup-$\top$* composition of A with R. Carefully observe that the result of the composition depends on the t-norm $\top$ that we choose to make the intersection between the fuzzy sets R and $\text{ext}_Y(A)$. As the NEFPROX algorithm that we are about to discuss, makes use of Zadeh's t-norm, the t-norm used to compute (4.56) can be replaced by the min-operator, and (4.56) can be called the *sup-min* composition of A with R.

We stated earlier that, for A and B fuzzy sets on the respective universes X and Y, one of the problems in approximate reasoning is to define a fuzzy relation between X and Y using the membership functions of A and B, that models the fuzzy implication *if x is A then y is B*, denoted by $A \to B$. Such a fuzzy implication is a fuzzy subset of $X \times Y$, and there are different ways that this fuzzy subset can be constructed from the membership functions of A and B. In light of the NEFPROX algorithm that we employ in our validation method, one type of implication is of specific interest to us. This implication is called the *Mamdani implication*, and is formally defined as follows,

Definition 61 (Mamdani implication)
For X and Y arbitrary non-empty sets, A and B fuzzy sets on X and Y respectively, and $\top \triangleq \min$ Zadeh's t-norm, a fuzzy relation $A \to B$ between X and Y is called a *Mamdani implication* iff. it holds for all $x \in X$ and $y \in Y$ that

$$[A \to B](x,y) \triangleq \top(A(x),B(y)) \tag{4.57}$$

□

With the notion of a Mamdani implication in mind, we now state the *generalised modus ponens* rule of inference, which forms the backbone of the theory of approximate reasoning. For X and Y arbitrary non-empty sets, and (a, b) a point of $X \times Y$, the LHS of (4.58) contains an example of the *classic modus ponens* rule of inference,

$$\begin{array}{lll} \textit{premise} & \text{if } x \text{ is } a \text{ then } y \text{ is } b \\ \textit{fact} & x \text{ is } a \\ \hline \textit{conseq.} & \hfill y \text{ is } b \end{array} \qquad \begin{array}{lll} \textit{premise} & \text{if } x \text{ is } A \text{ then } y \text{ is } B \\ \textit{fact} & x \text{ is } A' \\ \hline \textit{conseq.} & \hfill y \text{ is } B' \end{array} \tag{4.58}$$

Thus, very simply, for e.g. $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla}$ a behaviour relation estimate in light of the coin striking process in the former sections, with ∇ a behaviour orientation that defines an independent variable *average inter-arrival time* and a dependent variable *average queue length*, and (a, b) a point of the behaviour relation estimate, the premise *if x is a then y is b* reads as *if average inter-arrival time equals a, then average queue length equals b.*

The RHS of (4.58) contains an example of the *generalised modus ponens* rule of inference. In the example, A and A' are fuzzy sets on X, while B and B' are fuzzy sets on Y. In the context of the behaviour relation estimate $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla}$, let's interpret A as a fuzzy set modelling *small* average inter-arrival time, and B as a fuzzy set modelling *large* average queue length. Further, assume that the premise $A \rightarrow B$ of the RHS of (4.58) is defined as a Mamdani implication. The premise thus states that small average inter-arrival time brings about large average queue length. Let now A' be a fuzzy set modelling *very small* average inter-arrival time. Then, applying the generalised modus ponens rule of inference in (4.58), we can induce from the premise $A \rightarrow B$ (small average inter-arrival time yields large average queue length), and the fact A' (*very small* average inter-arrival time), a consequence B'. Intuitively, we expect B' to be a fuzzy set that we can interpret as *very large* average queue length. The membership function of B' is in Zadeh's framework of approximate reasoning defined by

$$B' \triangleq A' \cdot [A \rightarrow B] \tag{4.59}$$

Thus, the composition of a fuzzy set with a fuzzy relation is used to induce a consequence from a fuzzy implication and a fact. In the following, we now present the general working method of the NEFPROX algorithm. We indicate how the algorithm induces a family of fuzzy rules from a behaviour relation estimate. We also point out how the resulting fuzzy rule base can be used to *approximate* the relation.

4.2 The NEFPROX algorithm and behaviour approximations

The term NEFPROX[30] refers to an algorithm that tries to approximate a *function* with a neural network that employs fuzzy sets as weights on connections between neurons of different layers in the network. We intend to apply this algorithm to approximate a *behaviour relation estimate* of a stochastic system. Although the fact that our object of interest is strictly speaking a *relation*, as opposed to a *function*, this point should not be a point of great concern. For, we do *not* expect a behaviour relation to relate one and the same point of (integrated) statistics of stochastic *input* information processes to *different* points of (integrated) statistics of stochastic *output* information processes. Moreover, in the unlikely event that one and the same point is related to multiple points by a behaviour relation, then we are confident that this fact will not significantly 'confuse' the NEFPROX algorithm if only for the fact that the algorithm creates a *fuzzy* rule base, which is precisely aimed at dealing with imprecise and/or noisy information.

A NEFPROX network that is induced from data, consists of three layers: an *input* layer, a *hidden* layer, and an *output* layer. We identify the data from which we like to induce a network, with a finite *estimate* of the behaviour of a stochastic system $\mathcal{K}$ through some behaviour orientation $(\mathscr{X}, \mathscr{Y})$. In a preliminary phase, we create a neuron on the input layer of the network for every *independent* variable in $\mathscr{X}$. Likewise, we create a neuron on the output layer of the network for every *dependent* variable in $\mathscr{Y}$. The algorithm then installs a number of fuzzy subsets of the real line for every independent and dependent variable. For each variable, the fuzzy sets are defined such that their combined support contains all points in between the lowest observation and the highest observation on the variable. Each of the fuzzy sets is associated with a linguistic term, such as *small*, *medium*, *large*, etc. The main task of the NEFPROX *learning* algorithm will then be to create a number of neurons on the hidden layer, to link up these neurons with neurons on the input and the output layer, and to attach to every link a linguistic term, implemented by one of the predetermined fuzzy sets. The main task of the NEFPROX *training* algorithm will then be to modify the membership functions that characterise the fuzzy sets on the neuron to neuron connections, in an attempt to reach a rule base that adequately approximates (in our case) a behaviour relation estimate of a stochastic system. The number of fuzzy sets for each variable, and the connection between the fuzzy sets and the linguistic terms remains fixed during the training algorithm.

Once the network structure and the initial fuzzy sets are defined, the *learning algorithm* of NEFPROX is invoked. In the context of an estimated behaviour relation $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla}$, where $\mathcal{K}$ is a validatable stochastic system, and $\nabla \triangleq (\mathscr{X}, \mathscr{Y})$ a behaviour orientation for $\mathcal{K}$, this algorithm can be summarised as follows. Let $((x_1, x_2, \ldots, x_m), (y_1, y_2, \ldots, y_n))$ be a point of $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla}$.[31] For every $i \triangleq 1, 2, \ldots, m$, the fuzzy set with respect to which x_i has the *highest* degree of membership of all fuzzy sets that are associated with the independent variable that

[30]NEFPROX stands for *neuro-fuzzy function approximation*.

[31]Here, $x_1, x_2, \ldots, x_m$ are values of independent variables, while $y_1, y_2, \ldots, y_n$ are values of dependent variables.

carries index i, is identified. Applying this for every independent variable yields a total of m fuzzy sets, that make up the *antecedent* part of a potentially new rule neuron. In effect, the antecedent part of every rule currently in the rule base is inspected, and a new rule neuron is created on the hidden layer in case no rule neuron exists that has the former identified fuzzy sets as its weights on its links with input neurons.[32] The new rule neuron is then connected to all neurons on the output layer. Further, for every $j \triangleq 1, 2, \ldots, n$, the fuzzy set with respect to which y_j has the *highest* degree of membership of all fuzzy sets that are associated with the dependent variable that carries index j, is identified. In case the membership degree of y_j to this fuzzy set is unsatisfactory low, then a new fuzzy set is created for output neuron j. Applying this for all output neurons, yields a total of n fuzzy sets, that make up the *conclusion* part of the new rule neuron. The above steps are now repeated for all points of the estimated behaviour relation $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla}$.

After an initial (and possibly further optimised) fuzzy rule base has been established in the manner as we outlined above, the *training* algorithm of NEFPROX is invoked. In our case, the training algorithm processes *several times* all points of the estimated behaviour relation $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla}$. Each time all behaviour points are processed, an *epoch* is said to have been completed. For $((x'_1, x'_2, \ldots, x'_m), (y'_1, y'_2, \ldots, y'_n))$ a point of $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla}$, processed in some epoch, the input $(x'_1, x'_2, \ldots, x'_m)$ is *propagated* through the network, in order to achieve an output or target estimate $(\tilde{y}_1, \tilde{y}_2, \ldots, \tilde{y}_n)$. This goes as follows. Denote the set of neurons on the input, hidden and output layer by $U_{\text{in}} \triangleq \{u^1_{\text{in}}, u^2_{\text{in}}, \ldots, u^m_{\text{in}}\}$, $U_{\text{hdd}} \triangleq \{u^1_{\text{hdd}}, u^2_{\text{hdd}}, \ldots, u^q_{\text{hdd}}\}$ and $U_{\text{out}} \triangleq \{u^1_{\text{out}}, u^2_{\text{out}}, \ldots, u^n_{\text{out}}\}$ respectively, where $I \triangleq \{1, 2, \ldots, m\}$, $K \triangleq \{1, 2, \ldots, q\}$ and $J \triangleq \{1, 2, \ldots, n\}$. Further, let $A_{i,k}$, $\forall i \in I$ and $k \in K$, be the fuzzy set that is installed on the link between input neuron i and rule neuron k. Also, let $B_{k,j}$, $\forall k \in K$ and $j \in J$ be the fuzzy set that is installed on the link between rule neuron k and output neuron j. If we focus on an output neuron with index j, then a rule node with index k can be written as a fuzzy premise, denoted by

$$R_{k,j} \triangleq \text{if } x_1 \text{ is } A_{1,k} \text{ and } x_2 \text{ is } A_{2,k} \text{ and } \ldots \text{ and } x_m \text{ is } A_{m,k} \text{ then } y_j \text{ is } B_{k,j} \tag{4.60}$$

The input $x \triangleq (x'_1, x'_2, \ldots, x'_m)$ can be written as a fuzzy subset $\mathcal{F}(x) \triangleq \{(x, 1)\}$ of $\mathbf{R}^m$. In that respect, the generalised modus ponens rule of inference can be applied to compose $\mathcal{F}(x)$ with the fuzzy relation $R_{k,j}$ of (4.60), and this for every $j \in J$. Given the input x, denote the composition by $Y_{k,j}(x) \triangleq \mathcal{F}(x) \cdot R_{k,j}$ for every $j \in J$. The membership value of an arbitrary point y of $\mathbf{R}$ to every of the fuzzy sets $Y_{k,j}(x)$, $j \in J$ is computed as follows. First, the membership value of x'_i to $A_{i,k}$ is determined for every $i \in I$. Then, Zadeh's t-norm is used to compute $\alpha_k(x) \triangleq \top_{i \in I}\{A_{i,k}(x'_i)\}$. This yields an indication of the degree of *matching*

[32] In case such a rule neuron already exists, then the algorithm will take the next point of the behaviour relation estimate.

of the input x with the antecedent part of the rule of (4.60). Next, the membership value of y to $B_{k,j}$ is determined for every $j \in J$. Since NEFPROX makes use of the Mamdani implication operator, the membership value of the point $((x'_1, x'_2, \ldots, x'_m), y)$ to $R_{k,j}$ equals $\gamma_{k,j}(x,y) \triangleq \min(\alpha_k(x), B_{k,j}(y))$ for every $j \in J$. Since $\text{core}(\mathcal{F}(x)) = \{x\}$, the membership value of y to $Y_{k,j}(x)$ equals $\gamma_{k,j}(x,y)$ for every $j \in J$. Repeating the above for every point y of $\mathbb{R}$ completely defines the fuzzy sets $Y_{k,j}(x),\ j \in J$. Further, repeating this for all rule nodes on the hidden layer yields a number of fuzzy sets $Y_{k,j}(x),\ k \in K,\ j \in J$. The *activation* of output neuron $u^j_{\text{out}},\ \forall j \in J$ is now defined as the *union* of the fuzzy sets $Y_{k,j}(x),\ k \in K$, using Zadeh's t-conorm, or

$$\text{activation of } u^j_{\text{out}} \text{ under input } x \triangleq C_j(x) \triangleq \bigvee_{k \in K} Y_{k,j}(x) \tag{4.61}$$

The *output* of output neuron u^j_{out} is then computed from *defuzzifying* the activation of (4.61). A defuzzifier is generally defined as a procedure that converts a fuzzy set on a universe X into a *representative* element of X. The NEFPROX algorithm offers two defuzzification strategies, i.e. *center of gravity* and *middle of maxima.* Denoting $a \triangleq \inf[\text{supp}(C_j(x))]$ and $b \triangleq \sup[\text{supp}(C_j(x))]$, and M as the set of all points that have the highest membership degree to $C_j(x)$, the center of gravity and middle of maxima defuzzifiers can be written as

$$\text{cog}[C_j(x)] \triangleq \frac{\int_a^b y C_j(x)(y) dy}{\int_a^b C_j(x)(y) dy} \qquad \text{mom}[C_j(x)] \triangleq \frac{\int_M y dy}{\int_M dy} \tag{4.62}$$

Using (4.61) and applying one of the defuzzifiers of (4.62) for every output neuron $j \in J$, NEFPROX creates a target estimate $(\tilde{y}_1, \tilde{y}_2, \ldots, \tilde{y}_n)$ in response to the input $(x'_1, x'_2, \ldots, x'_m)$. The difference between the target *estimate* $(\tilde{y}_1, \tilde{y}_2, \ldots, \tilde{y}_n)$ and the *true* target $(y'_1, y'_2, \ldots, y'_n)$ is now propagated *backwards* through the network to alter the fuzzy sets $A_{i,k},\ i \in I,\ k \in K$ and $B_{k,j},\ k \in K,\ j \in J$. The modifications that are made implement a fuzzy variant of the backpropagation algorithm that is used to train multilayer Perceptron networks, and is known as *fuzzy backpropagation.* Eventually, propagation of the input $(x'_1, x'_2, \ldots, x'_m)$ possibly leads to a modified rule base. The learning algorithm now continues with the next pattern of $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla}$, and the steps that we outlined above are repeated to compute an output estimate. Once all patterns are processed, an epoch is completed, and the algorithm starts propagating the input of the first pattern again, until a stopping criterion is met.

A first problem that we encounter when we want to induce from a behaviour relation estimate a fuzzy rule base with the NEFPROX algorithm, is how we should decide on a suitable *parameter set-up* for the algorithm. In effect, we can define a different *number* of fuzzy sets for each independent and dependent variable. Also, we can choose the *shape* of the membership functions of the fuzzy sets to be *triangular*, *trapezoidal* or *normal*. Furthermore, we can select one of the *defuzzification* methods of (4.62). Besides these parameters, a number of other parameters must be decided upon, as there are a *learning rate*, a *maximum* number of rules that are allowed in the rule base, and whether or not the membership functions of the fuzzy sets that are defined for a variable are allowed to *pass* one another. By carrying out a number of preliminary tests with the algorithm, and based on conclusions that we drew in [18] from tests with NEFCLASS - an algorithm much similar to NEFPROX that can be used for *classification* -, we decided to retain the following parameters as being significant: *number* of fuzzy sets per independent and dependent variable, *shape* of the membership function of the fuzzy sets, and the *defuzzification* method. In that respect, for a network with m input and n output neurons, we define an *(m,n)-NEFPROX parameter-set up* as follows,

Definition 62 ((m,n)-NEFPROX parameter set-up)
For a NEFPROX network with m input and n output neurons, indexed by the respective index sets I and J, c^i_{in}, $i \in I$ and c^j_{out}, $j \in J$ numbers of fuzzy sets, S^i_{in}, $i \in I$ and S^j_{out}, $j \in J$ shapes of membership functions of fuzzy sets, and defuzz a defuzzifier, an *(m,n)-NEFPROX parameter set-up* is a 5-tuple, defined by

$$\mathcal{P} \triangleq \langle \{c^i_{\text{in}}\}_{i\in I}, \{c^j_{\text{out}}\}_{j\in J}, \{S^i_{\text{in}}\}_{i\in I}, \{S^j_{\text{out}}\}_{j\in J}, \text{defuzz} \rangle \tag{4.63}$$

□

Thus, very simply, an (m,n)-NEFPROX parameter set-up $\mathcal{P}$ defines a number of triangular, trapezoidal or normal shaped fuzzy sets for each input and output neuron. In addition, the parameter set-up specifies whether we use the center of gravity or the middle of maxima defuzzification method. For all other parameters that are *not* pinned down by the parameter set-up, we use the *default* values for these parameters, that are provided in a standard initialisation file for the algorithm. With the help of a NEFPROX parameter set-up, we can now formally define what we mean by a *NEFPROX approximation* of a behaviour relation estimate of a stochastic system,

Definition 63 (NEFPROX approximation of behaviour relation estimate)
For $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla} \subseteq \mathbf{R}^m \times \mathbf{R}^n$ a finite estimate of the behaviour of an approximately stationary, steady-state and ergodic stochastic system $\mathcal{K}$ through a behaviour orientation ∇, and $\mathcal{P}$ an (m,n)-parameter set-up for NEFPROX, the fuzzy rule base that is induced by applying the NEFPROX learning and training algorithm on $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla}$ under the parameter set-up, is called a *$\mathcal{P}$-approximation of $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla}$*.

□

A NEFPROX approximation of a behaviour relation estimate $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla}$ is thus the fuzzy rule base that is generated by applying the NEFPROX algorithm that we discussed in paragraph 4.2 on $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla}$. In the following, we denote a $\mathcal{P}$-NEFPROX approximation of $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla}$ by $\mathbb{A}_{\mathcal{P}}(\hat{\mathscr{B}}(\mathcal{K})_{|\nabla})$. Sometimes, we also employ the notation $\mathbb{A}$, $\mathbb{B}$, etc. to denote NEFPROX approximations.

4.3 Pseudo-metrics and resemblance relations

We like to use the concept of *resemblance relations* in fuzzy set theory in order to arrive at a notion of gradual validity. Resemblance relations were first introduced by [6] to overcome certain paradoxes that may arise in approximate reasoning. A resemblance relation is defined with the help of a *pseudo-metric*,

Definition 64 (Pseudo-metric)
For Ω an arbitrary non-empty set, a function d from $\Omega \times \Omega$ into $\mathbf{R}^+$ is called a *pseudo-metric on* Ω iff. it holds for all ω, ω' and ω'' in Ω that

$$d(\omega,\omega) = 0 \tag{4.64}$$
$$d(\omega,\omega') = d(\omega',\omega) \tag{4.65}$$
$$d(\omega,\omega') + d(\omega',\omega'') \geq d(\omega,\omega'') \tag{4.66}$$

□

For Ω a non-empty set, and d a pseudo-metric on Ω, the pair (Ω, d) is called a *pseudo-metric space.* A *resemblance relation* can now formally be defined as follows,

Definition 65 ((g,d)-resemblance relation)
For Ω a non-empty set, (X,d) a pseudo-metric space, and g a function from Ω into X, a fuzzy relation R on Ω is called a *(g,d)-resemblance relation* iff. it holds for all ω, ω', ω'' and $\tilde{\omega}$ in Ω that

$$R(\omega,\omega) = 1 \tag{4.67}$$
$$R(\omega,\omega') = R(\omega',\omega) \tag{4.68}$$
$$d(g(\omega),g(\omega')) \leq d(g(\omega''),g(\tilde{\omega})) \Rightarrow R(\omega,\omega') \geq R(\omega'',\tilde{\omega}) \tag{4.69}$$

□

Carefully notice that in order to be able to speak of a resemblance relation, we must have a non-empty set Ω, a metric space (X,d) and a function g from Ω into X. Equation (4.67) is the logical requirement that the resemblance between any point of Ω and itself has to equal 1. According to (4.68), the resemblance between a point ω and a point ω', must be identical to the resemblance between ω' and ω. Finally, equation (4.69) states that, if the distance between $g(\omega)$ and $g(\omega')$ is *not* larger than the distance between $g(\omega'')$ and $g(\tilde{\omega})$ for some $\omega, \omega', \omega''$ and $\tilde{\omega}$ in Ω, then the resemblance between ω and ω' should *not* be lower than the resemblance between ω'' and $\tilde{\omega}$.

It is clear from definition 65 that, given the sets Ω and X, and the function g from Ω into X, whether or not a fuzzy relation on Ω will be a resemblance relation, depends on the pseudo-metric that we define on X. In [6], the resemblance relation concept is illustrated by a number of examples for different pseudo-metrics. Based on these examples, we like to retain the so-called *infinity pseudo-metric* in light of the concept of gradual validity that we develop. We define an infinity pseudo-metric formally as follows,

Definition 66 (Infinity pseudo-metric)
For X_i, $i \in I$ arbitrary non-empty sets, indexed by a finite index set I, a pseudo-metric on $X \triangleq \prod_{i \in I} X_i$ is called the *infinity pseudo-metric on X*, denoted by d_∞, iff. it holds for all $x \triangleq (x_1, x_2, \ldots, x_{|I|})$ and $y \triangleq (y_1, y_2, \ldots, y_{|I|})$ in X that

$$d_\infty(x, y) \triangleq \max_{i \in I}\{|x_i - y_i|\} \tag{4.70}$$

□

Everything as in definition 66, the infinity pseudo-metric d_∞ on X defines the distance between the points x and y to be the maximum over all $i \in I$ of the absolute difference between x_i and y_i. Stated somewhat informally, the infinity pseudo-metric on X is a *conservative* pseudo-metric, as a single coordinate with index i for which holds that $|x_i - y_i|$ is large, is sufficient to render the distance between x and y large.

For Ω an arbitrary non-empty set, $X \triangleq [0,1]^k$ and $A_1, A_2, \ldots, A_k$ fuzzy subsets of Ω for some $k \in \mathbb{N}_0$, an example of a resemblance relation is given in [6] by choosing the infinity pseudo-metric on X and by defining a function from Ω into X that assigns to every point $\omega \in \Omega$ a tuple with membership values $A_1(\omega), A_2(\omega), \ldots, A_k(\omega)$. This example is of particular interest in our case. For, in case $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla} \subseteq \mathbb{R}^m \times \mathbb{R}^n$ is a *finite* behaviour relation estimate of an approximately stationary, steady-state and ergodic stochastic system, then we like to identify Ω with $\mathbb{R}^m \times \mathbb{R}^n$, and the fuzzy sets $A_1, A_2, \ldots, A_k$ with the fuzzy rules in a NEFPROX approximation of the behaviour relation estimate. In that respect, for $\mathbb{A}$ a NEFPROX approximation of a behaviour relation estimate, we introduce the notion of an $\mathbb{A}$-*resemblance relation* in the following definition,

Definition 67 ($\mathbb{A}$-resemblance relation)
For $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla} \subseteq \mathbb{R}^m \times \mathbb{R}^n$ a behaviour relation estimate of an approximately stationary, steady-state and ergodic stochastic system $\mathcal{K}$ through the orientation ∇, $\mathcal{P}$ an (m,n)-parameter set-up for NEFPROX, $\mathbb{A} \triangleq \{A_1, A_2, \ldots, A_k\}$ the $\mathcal{P}$-NEFPROX approximation of $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla}$, d_∞ the infinity pseudo-metric on $[0,1]^k$, and g a function from $\mathbb{R}^m \times \mathbb{R}^n$ into $[0,1]^k$, defined in every $x \in \mathbb{R}^m \times \mathbb{R}^n$ by $g(x) \triangleq (A_1(x), A_2(x), \ldots, A_k(x))$, a fuzzy relation $R_{\mathbb{A}}$ on $\mathbb{R}^m \times \mathbb{R}^n$ is called an $\mathbb{A}$-*resemblance relation* iff. it holds for all x and y in $\mathbb{R}^m \times \mathbb{R}^n$ that

$$R_{\mathbb{A}}(x, y) \triangleq 1 - d_\infty(g(x), g(y)) \tag{4.71}$$

□

Thus, everything as in definition 67, the $\mathbb{A}$-resemblance relation $R_{\mathbb{A}}$ defines the resemblance between two points x and y of $\mathbb{R}^m \times \mathbb{R}^n$ using the membership values of x and y to A_1, of x and y to A_2, etc. The resemblance between x and y will be *small* if there is *at least one* rule in the NEFPROX approximation for which holds that the membership value of x to the rule deviates significantly from the membership value of y to the same rule. The resemblance between x and y will be *large* if and only if for *every* rule in the NEFPROX approximation, x and y have similar membership degrees to the rule.

To better grasp the notion of an $\mathbb{A}$-resemblance relation, let everything be defined as in definition 67, and let $\hat{\mathscr{B}}(\mathcal{K}')_{|\nabla'} \subseteq \mathbf{R}^m \times \mathbf{R}^n$ be a behaviour relation estimate of another stochastic system $\mathcal{K}'$ through a behaviour orientation ∇' that is compatible to ∇. Let x be a point of the behaviour relation estimate $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla}$, and let y be a point of the behaviour relation estimate $\hat{\mathscr{B}}(\mathcal{K}')_{|\nabla'}$. If the resemblance between x and y is *large*, then we can consider y as a point of $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla}$ to some degree, since we find through the NEFPROX approximation of $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla}$ that both x and y are consistent with the knowledge in the approximation. Although y may not be an element of $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla}$, we can consider the statement *y is an element of $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla}$* to be *somewhat* true since there is a point x of $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla}$ for which holds that y highly resembles x.

4.4 Resemblance random variables

In classic systems theory [31], an abstract system is called a *valid subsystem* of another abstract system if and only if every input/output segment pair that the former system produces can also be produced by the latter system. If the reverse statement holds, in that every input/output segment pair of the latter system can be produced by the former, then we call the former system a *valid supersystem* of the latter system. Clearly, if a system is a valid subsystem of another system, then the other system is a valid supersystem of the first system, and vice versa. In classic systems theory, an abstract system is said to be *valid* with respect to another abstract system if and only if it is both a valid subsystem and a valid supersystem of the other system. Valid systems thus bring about the same family of input/output segment pairs in classic systems theory. Now, in light of the stochastic system construct that we developed in this research report, and in light of the notion of stochastic system behaviour, we formulate a fuzzy and stochastic variant of the classic subsystem and supersystem validity conditions.

For $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla}$ and $\hat{\mathscr{B}}(\mathcal{K}')_{|\nabla'}$ behaviour relation estimates of approximately stationary, steady-state and ergodic stochastic systems $\mathcal{K}$ and $\mathcal{K}'$ through compatible behaviour orientations ∇ and ∇', and $\mathbb{A}$ and $\mathbb{B}$ NEFPROX approximations of $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla}$ and $\hat{\mathscr{B}}(\mathcal{K}')_{|\nabla'}$ respectively, we use the $\mathbb{A}$-resemblance relation $R_{\mathbb{A}}$ and the $\mathbb{B}$-resemblance relation $R_{\mathbb{B}}$ to measure the extent to which we can consider $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla}$ a *subset* of $\hat{\mathscr{B}}(\mathcal{K}')_{|\nabla'}$. If we can find for almost every element of $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla}$, an element of $\hat{\mathscr{B}}(\mathcal{K}')_{|\nabla'}$ that highly resembles the former, then we consider the statement $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla} \subseteq \hat{\mathscr{B}}(\mathcal{K}')_{|\nabla'}$ to be *very true* since most members of $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla}$ have at least one highly resembling member in $\hat{\mathscr{B}}(\mathcal{K}')_{|\nabla'}$. On the other hand, if we cannot find for the majority of elements in $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla}$, an element of $\hat{\mathscr{B}}(\mathcal{K}')_{|\nabla'}$ that significantly resembles the former, then we conclude that the statement $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla} \subseteq \hat{\mathscr{B}}(\mathcal{K}')_{|\nabla'}$ is only *little true* since most members of

$\hat{\mathscr{B}}(\mathcal{K})_{|\nabla}$ do not resemble any of the members of $\hat{\mathscr{B}}(\mathcal{K}')_{|\nabla'}$. In order then to formally express the extent to which $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla}$ is a subset of $\hat{\mathscr{B}}(\mathcal{K}')_{|\nabla'}$, we define in the following definition the notion of an $(\mathbb{A}, \mathbb{B})$-*resemblance random variable*,

Definition 68 ($(\mathbb{A}, \mathbb{B})$-resemblance random variable)
For $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla} \subseteq \mathbb{R}^m \times \mathbb{R}^n$ and $\hat{\mathscr{B}}(\mathcal{K}')_{|\nabla'} \subseteq \mathbb{R}^m \times \mathbb{R}^n$ finite estimates of the behaviour of approximately stationary, steady-state and ergodic stochastic systems $\mathcal{K}$ and $\mathcal{K}'$ through the respective compatible behaviour orientations ∇ and ∇', $\mathcal{P}$ an (m, n)-parameter set-up for NEFPROX, and $R_{\mathbb{A}}$ and $R_{\mathbb{B}}$ the $\mathbb{A} \triangleq \mathbb{A}_{\mathcal{P}}(\hat{\mathscr{B}}(\mathcal{K})_{|\nabla})$-resemblance relation and $\mathbb{B} \triangleq \mathbb{A}_{\mathcal{P}}(\hat{\mathscr{B}}(\mathcal{K}')_{|\nabla'})$-resemblance relation respectively, an $(\mathbb{A}, \mathbb{B})$-*resemblance random variable* is a random variable on $(\hat{\mathscr{B}}(\mathcal{K})_{|\nabla}, 2^{\hat{\mathscr{B}}(\mathcal{K})_{|\nabla}}, P)$, where P is defined in every $A \in 2^{\hat{\mathscr{B}}(\mathcal{K})_{|\nabla}}$ by $P(A) \triangleq \frac{|A|}{|\hat{\mathscr{B}}(\mathcal{K})_{|\nabla}|}$, denoted by $\mathbf{r}_{(\mathbb{A},\mathbb{B})}$ and defined in every $x \in \hat{\mathscr{B}}(\mathcal{K})_{|\nabla}$ by

$$\mathbf{r}_{(\mathbb{A},\mathbb{B})}(x) \triangleq \max_{y \in \hat{\mathscr{B}}(\mathcal{K}')_{|\nabla'}} [\min\{R_{\mathbb{A}}(x, y), R_{\mathbb{B}}(x, y)\}] \tag{4.72}$$

□

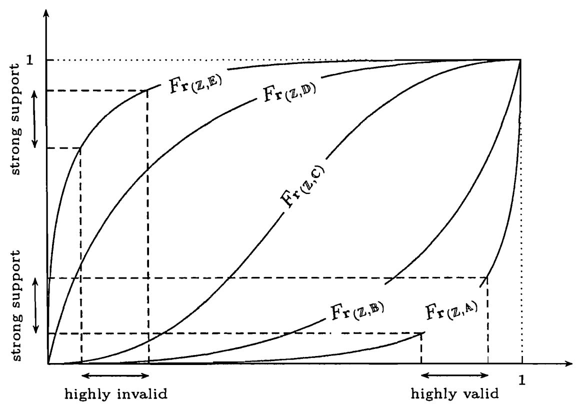
Everything as in definition 68, notice that the sample set of the $(\mathbb{A}, \mathbb{B})$-resemblance random variable $\mathbf{r}_{(\mathbb{A},\mathbb{B})}$ equals the behaviour relation estimate $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla}$. For every behaviour point x of $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla}$, we look up the behaviour point y of $\hat{\mathscr{B}}(\mathcal{K}')_{|\nabla'}$ for which holds that it has overall the *highest* resemblance with respect to x of all behaviour points of $\hat{\mathscr{B}}(\mathcal{K}')_{|\nabla'}$, as measured by *both* resemblance relations $R_{\mathbb{A}}$ and $R_{\mathbb{B}}$. Thus, for every point x of $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla}$, we use the NEFPROX approximations $\mathbb{A}$ and $\mathbb{B}$ of $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla}$ and $\hat{\mathscr{B}}(\mathcal{K}')_{|\nabla'}$ respectively, to identify the point y of $\hat{\mathscr{B}}(\mathcal{K}')_{|\nabla'}$ for which holds that x and y have the *most similar* membership values to all fuzzy rules of $\mathbb{A}$ *and* to all fuzzy rules of $\mathbb{B}$ of all behaviour points of $\hat{\mathscr{B}}(\mathcal{K}')_{|\nabla'}$. If we find for almost every point x of $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla}$ a point y of $\hat{\mathscr{B}}(\mathcal{K}')_{|\nabla'}$ such that $\min\{R_{\mathbb{A}}(x, y), R_{\mathbb{B}}(x, y)\}$ is relatively high, then we consider the proposition $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla} \subseteq \hat{\mathscr{B}}(\mathcal{K}')_{|\nabla'}$ to be *very true*, since in light of the knowledge that is contained in the NEFPROX approximations, almost every behaviour point of $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla}$ has *at least one* behaviour point of $\hat{\mathscr{B}}(\mathcal{K}')_{|\nabla'}$ for which holds that both points comply with the knowledge that is stored in the NEFPROX approximations $\mathbb{A}$ *and* $\mathbb{B}$.

In figure 8(a), we depicted by way of example possible *smoothed* cumulative distributions of different resemblance random variables. What do the smoothed distributions in the figure stand for? In order to address this question, assume that we are confronted with the following validation scenario. Let $\mathcal{A}_{|.}$ be an abstract object trace, representing all that we have available of a real coin striking process. Thus, $\mathcal{A}_{|.}$ is nothing but a family of relations that hold a limited number of input/output segment pairs, where every input/output segment pair contains an input segment with coin arrivals, and an output segment with queue length observations. Let now $\mathcal{K}$ be the stochastic system with main structured discrete event system $\mathcal{Q}$ and environment $\mathscr{E}$, that we postulate to generate an abstract object trace, denoted by $a(\mathcal{Q})_{|\mathscr{E}}$, that embodies all the input/output segment pairs of $\mathcal{A}_{|.}$. Assume now that as a result of our modelling efforts, we developed the stochastic systems $\mathcal{K}_A$ through $\mathcal{K}_F$. Recalling our framework for validation in section 1 and recalling figure 1, the stochastic systems $\mathcal{K}_A$ through $\mathcal{K}_F$ are thus competing primary models that we developed for the coin striking process. The validation dilemma that we face here is that we are uncertain which of the models $\mathcal{K}_A$ through $\mathcal{K}_F$ is the *best* model in light of the abstract object trace $\mathcal{A}_{|.}$. Assume now that we created a behaviour orientation for every stochastic system, containing an independent variable *average inter-arrival time*, and a dependent variable *average queue length*. Then, we can build a number of simulation models, and obtain through experiments with these models - through *simulation* - *estimates* of the behaviour of the stochastic systems $\mathcal{K}_A$, $\mathcal{K}_B$, etc. through their accompanying behaviour orientations. Let's denote these behaviour relation estimates by $\hat{\mathscr{B}}(\mathcal{K}_A)_{|\nabla_A}$, $\hat{\mathscr{B}}(\mathcal{K}_B)_{|\nabla_B}$, etc. We denote the behaviour relation estimate that we obtain by performing time based arithmetic with the input/output segment pairs of $\mathcal{A}_{|.}$ by $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla}$. Next, we create a suitable $(1,1)$-parameter set-up $\mathcal{P}$ for NEFPROX, and generate the $\mathcal{P}$-NEFPROX approximations of $\hat{\mathscr{B}}(\mathcal{K}_A)_{|\nabla_A}$, $\hat{\mathscr{B}}(\mathcal{K}_B)_{|\nabla_B}$, etc. Denote these approximations by $\mathbb{A}$, $\mathbb{B}$, etc. and denote the $\mathcal{P}$-NEFPROX approximation of $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla}$ by $\mathbb{Z}$. Within the validation scenario that we outlined here, the smoothed cumulative distributions that we displayed in figure 8(a) are cumulative distributions that we may possibly find for the resemblance random variables $\mathbf{r}_{(\mathbb{Z},\mathbb{A})}$, $\mathbf{r}_{(\mathbb{Z},\mathbb{B})}$, etc.

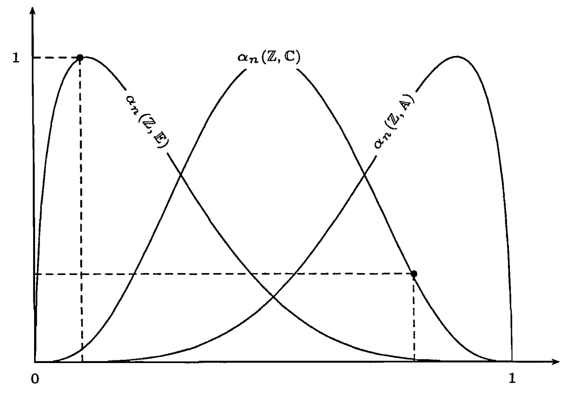
According to figure 8(a), for most points in the behaviour relation estimate $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla}$, we do *not* find a point in the behaviour relation estimate $\hat{\mathscr{B}}(\mathcal{K}_E)_{|\nabla_E}$ that highly resembles the former point through *both* the $\mathbb{Z}$-resemblance relation $R_{\mathbb{Z}}$ *and* the $\mathbb{E}$-resemblance relation $R_{\mathbb{E}}$. Hence, we state that the proposition $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla}$ *is a subset of* $\hat{\mathscr{B}}(\mathcal{K}_E)_{|\nabla_E}$ is only *little true*. In contrast, for most points in the behaviour relation estimate $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla}$, we *do* find at least one point in the behaviour relation estimate $\hat{\mathscr{B}}(\mathcal{K}_A)_{|\nabla_A}$ that highly resembles the former point through *both* the $\mathbb{Z}$-resemblance relation $R_{\mathbb{Z}}$ *and* the $\mathbb{A}$-resemblance relation $R_{\mathbb{A}}$. Therefore, we state that the proposition $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla}$ *is a subset of* $\hat{\mathscr{B}}(\mathcal{K}_A)_{|\nabla_A}$ is *highly true*. The problem that we now face is how we should *convert* the information that is captured by the cumulative distribution of a resemblance random variable, in an overall verdict on the extent that a behaviour relation estimate of a stochastic system can be considered a subset of a behaviour relation estimate of another stochastic system. We deal with this problem in the following and final subsection, in which we develop the concept of a *fuzzy set* of propositions on validity.

4.5 Fuzzy sets of system validity propositions

For $F_{\mathbf{r}_{(\mathbb{A},\mathbb{B})}}$ the cumulative distribution of a resemblance random variable, with $\mathbb{A}$ and $\mathbb{B}$ NEFPROX approximations of behaviour relation estimates $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla}$ and $\hat{\mathscr{B}}(\mathcal{K}')_{|\nabla'}$, we convert the cumulative distribution into a *fuzzy set* of *propositions* on the *degree* to which the behaviour relation estimate $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla}$ can be considered a subset of the behaviour relation estimate $\hat{\mathscr{B}}(\mathcal{K}')_{|\nabla'}$. We interpret the membership value of a proposition to the fuzzy set as the degree of *truth* of the proposition. We derive such a fuzzy set of propositions from the cumulative distribution as follows. First, we divide the unit interval $[0,1]$ in n equally sized intervals $[a_1 \triangleq 0, b_1[$, $[a_2, b_2[$,



(a) Cumulative distributions



(b) Fuzzy sets of subsystem validity propositions

Figure 8: Resemblance random variables and fuzzy sets of subsystem validity propositions

etc. until and including $[a_n, b_n \triangleq 1]$, for some $n \in \mathbb{N}_0$. Then, we compute for every interval the *relative* number of realisations of the random variable $\mathbf{r}_{(\mathbb{A},\mathbb{B})}$ that fall within the interval. Denote the relative number of realisations in the i^{th} interval by c_i. Further, we look up the maximum relative number of realisations that fall within any interval, and divide all relative numbers by this maximum. Thus, in case $c_j \triangleq \max_{i=1,2,\ldots,n}\{c_i\}$, then we compute the ratios $\frac{c_1}{c_j}$, $\frac{c_2}{c_j}$, etc.

Finally, we create a fuzzy subset of the unit interval by defining a membership function from $\mathbb{R}$ into $[0,1]$ that assigns to every point $x \in [0,1]$ the membership value $\frac{c_i}{c_j}$, where i identifies the interval that contains x, and that assigns to every point $x \notin [0,1]$ the membership value 0. This approach of creating a *fuzzy set* on the unit interval resembles to some extent the approach of constructing an *histogram* from empirical data. In a way, we treat the random variable $\mathbf{r}_{(\mathbb{A},\mathbb{B})}$ as if it were *continuous*, and compute a kind of *scaled histogram.* We call the created fuzzy set a *fuzzy set of stochastic subsystem validity propositions of resolution n.* Formally,

Definition 69 (Fuzzy set of stochastic subsystem validity propositions)
For $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla} \subseteq \mathbb{R}^m \times \mathbb{R}^n$ and $\hat{\mathscr{B}}(\mathcal{K}')_{|\nabla'} \subseteq \mathbb{R}^m \times \mathbb{R}^n$ finite estimates of the behaviour of approximately stationary, steady-state and ergodic stochastic systems $\mathcal{K}$ and $\mathcal{K}'$ through the respective compatible behaviour orientations ∇ and ∇', $\mathcal{P}$ an (m,n)-parameter set-up for NEFPROX, and $\mathbb{A} \triangleq \mathbb{A}_{\mathcal{P}}(\hat{\mathscr{B}}(\mathcal{K})_{|\nabla})$ and $\mathbb{B} \triangleq \mathbb{A}_{\mathcal{P}}(\hat{\mathscr{B}}(\mathcal{K}')_{|\nabla'})$ NEFPROX approximations of $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla}$ and $\hat{\mathscr{B}}(\mathcal{K}')_{|\nabla'}$ respectively, a *fuzzy set of stochastic subsystem validity propositions of $\mathcal{K}$ with respect to $\mathcal{K}'$ of resolution n* is a fuzzy subset of $[0,1]$, denoted by $\alpha_n(\mathbb{A},\mathbb{B})$, the membership function of which is defined in every $x \in \mathbb{R}$ by, where $c_{\max} \triangleq \max_{j=1,2,\dots,n}\{P_{\mathbf{r}_{(\mathbb{A},\mathbb{B})}}([\frac{j-1}{n},\frac{j}{n}[), P_{\mathbf{r}_{(\mathbb{A},\mathbb{B})}}(\{1\})\}$,

$$\alpha_n(\mathbb{A},\mathbb{B})(x) \triangleq \begin{cases} 0 & x < 0 \\ P_{\mathbf{r}_{(\mathbb{A},\mathbb{B})}}([\frac{i-1}{n},\frac{i}{n}[)/c_{\max} & \frac{i-1}{n} \leq x < \frac{i}{n} \\ P_{\mathbf{r}_{(\mathbb{A},\mathbb{B})}}(\{1\})/c_{\max} & x = 1 \\ 0 & x > 1 \end{cases} \tag{4.73}$$

□

Returning to figure 8, we displayed by way of example in figure 8(b) a number of possible, *smoothed* fuzzy sets of stochastic subsystem validity propositions of a certain resolution n. In fact, the fuzzy sets that we displayed in figure 8(b) are meant to represent some of the converted cumulative distributions of figure 8(a). To give an example, consider the stochastic system $\mathcal{K}_A$. The curve of figure 8(a) that represents the smoothed cumulative distribution of the resemblance random variable $\mathbf{r}_{(\mathbb{Z},\mathbb{A})}$ is transformed by (4.73) in the fuzzy set $\alpha_n(\mathbb{Z},\mathbb{A})$ in figure 8(b). How should we now interpret this fuzzy set? First, we associate with every point $x \in \{\frac{i}{n}\}_{i\in I}$, where $I \triangleq \{0,1,2,\dots,n\}$, a *linguistic term* of the form *L-similar*, where L represents a linguistic modifier. Examples of modifiers are *very*, *more or less*, *somewhat* and *not very*. We whish to agree here to make this association in a *monotonic* way. For $x \triangleq 1$ and $x \triangleq 0$, we think of the linguistic terms *completely similar* and *not at all similar (or completely dissimilar)* respectively. As we gradually decrease x from 1 to 0, we like to associate with x linguistic terms as there are *very similar*, *somewhat similar*, *little similar* (or *somewhat dissimilar*), etc. Now, in case we have associated with e.g. $x \triangleq \frac{n-1}{n}$ the linguistic term *very similar*, then we introduce the notation $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla}[\sim]_x\hat{\mathscr{B}}(\mathcal{K}_A)_{|\nabla_A}$ to denote the statement *every element of $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla}$ has a very similar element in $\hat{\mathscr{B}}(\mathcal{K}_A)_{|\nabla_A}$.* In case we have associated with e.g. $x \triangleq \frac{1}{n}$ the linguistic term *very dissimilar*, then we read $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla}[\sim]_x\hat{\mathscr{B}}(\mathcal{K}_A)_{|\nabla_A}$ as *every element of $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla}$ has a very dissimilar element in $\hat{\mathscr{B}}(\mathcal{K}_A)_{|\nabla_A}$.* We define the *truth value* of the proposition $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla}[\sim]_x\hat{\mathscr{B}}(\mathcal{K}_A)_{|\nabla_A}$ for some $x \in \{\frac{i}{n}\}_{i\in I}$ to be the membership value of x to the fuzzy set $\alpha_n(\mathbb{Z},\mathbb{A})$. As figure 8(b) suggests, the membership value of $x \triangleq \frac{n-1}{n}$ to $\alpha_n(\mathbb{Z},\mathbb{A})$ is high. Therefore, the proposition $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla}[\sim]_{\frac{n-1}{n}}\hat{\mathscr{B}}(\mathcal{K}_A)_{|\nabla_A}$ is very much true. Stated otherwise, the following proposition very much holds, *every element of $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla}$ has a very similar element in $\hat{\mathscr{B}}(\mathcal{K}_A)_{|\nabla_A}$ - as seen through the NEFPROX approximations $\mathbb{Z}$ and $\mathbb{A}$.*

For x relatively large, a high membership value of x to $\alpha_n(\mathbb{Z},\mathbb{A})$ indicates that there is *strong support* to call $\mathcal{K}$ a *very valid* stochastic subsystem of $\mathcal{K}_A$. The support is strong since the membership value is large. The statement includes the linguistic term *very valid* as we have chosen x to be relatively large. Notice from figure 8(b) that for the stochastic systems $\mathcal{K}$ and $\mathcal{K}_E$, the proposition $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla}[\sim]_x\hat{\mathscr{B}}(\mathcal{K}_E)_{|\nabla_E}$ is *very false* for x relatively large. Hence, there is *little or no support* - since the membership value of x to $\alpha_n(\mathbb{Z},\mathbb{E})$ is *low* - to call $\mathcal{K}$ a *very valid* stochastic subsystem of $\mathcal{K}_E$ - where we employ the term *very valid* since x is relatively large. Notice from the figure that, as we decrease x, the membership value of x to the fuzzy set $\alpha_n(\mathbb{Z},\mathbb{E})$ increases. Therefore, the proposition $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla}[\sim]_x\hat{\mathscr{B}}(\mathcal{K}_E)_{|\nabla_E}$ becomes *more and more* true as we lower x. For $x \triangleq \frac{1}{n}$, it follows from figure 8(b) that the statement *every element of $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla}$ has a very dissimilar element in $\hat{\mathscr{B}}(\mathcal{K}_E)_{|\nabla_E}$* is very true. In other words, there is strong support to call $\mathcal{K}$ a highly invalid stochastic subsystem of $\mathcal{K}_E$.

In the above, we developed a fuzzy set theoretic approach to attach a label of *truth* to propositions on the extent that a behaviour relation estimate of a stochastic system is believed to be a subset of a behaviour relation estimate of another stochastic system. For $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla}$ and $\hat{\mathscr{B}}(\mathcal{K}')_{|\nabla'}$ behaviour relation estimates, it is now only natural to ask whether the statement $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla}[\sim]_x\hat{\mathscr{B}}(\mathcal{K}')_{|\nabla'}$ is *equally* true than the statement $\hat{\mathscr{B}}(\mathcal{K}')_{|\nabla'}[\sim]_x\hat{\mathscr{B}}(\mathcal{K})_{|\nabla}$, and this for every $x \in \{\frac{i}{n}\}_{i\in I}$. The answer is in the *negative*. For, in order to evaluate the former statement, we use the $(\mathbb{A},\mathbb{B})$-resemblance random variable $\mathbf{r}_{(\mathbb{A},\mathbb{B})}$, where $\mathbb{A}$ is a NEFPROX approximation of $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla}$ and $\mathbb{B}$ is a NEFPROX approximation of $\hat{\mathscr{B}}(\mathcal{K}')_{|\nabla'}$. However, in order to evaluate the latter statement, we use the $(\mathbb{B},\mathbb{A})$-resemblance random variable $\mathbf{r}_{(\mathbb{B},\mathbb{A})}$. Although the NEFPROX approximations that are used by the random variables $\mathbf{r}_{(\mathbb{A},\mathbb{B})}$ and $\mathbf{r}_{(\mathbb{B},\mathbb{A})}$ are the same, the random variables themselves will in general not have identical distributions. In that respect, we define in the following definition the concept of a *fuzzy set of stochastic supersystem validity propositions*,

Definition 70 (Fuzzy set of stochastic supersystem validity propositions)
For $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla} \subseteq \mathbf{R}^m \times \mathbf{R}^n$ and $\hat{\mathscr{B}}(\mathcal{K}')_{|\nabla'} \subseteq \mathbf{R}^m \times \mathbf{R}^n$ finite estimates of the behaviour of approximately stationary, steady-state and ergodic stochastic systems $\mathcal{K}$ and $\mathcal{K}'$ through the respective compatible behaviour orientations ∇ and ∇', $\mathcal{P}$ an (m,n)-parameter set-up for NEFPROX, and $\mathbb{A} \triangleq \mathbb{A}_{\mathcal{P}}(\hat{\mathscr{B}}(\mathcal{K})_{|\nabla})$ and $\mathbb{B} \triangleq \mathbb{A}_{\mathcal{P}}(\hat{\mathscr{B}}(\mathcal{K}')_{|\nabla'})$ NEFPROX approximations of $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla}$ and $\hat{\mathscr{B}}(\mathcal{K}')_{|\nabla'}$ respectively, a *fuzzy set of stochastic supersystem validity propositions of $\mathcal{K}$ with respect to $\mathcal{K}'$ of resolution n* is a fuzzy subset of $[0,1]$, denoted by $\beta_n(\mathbb{A},\mathbb{B})$, the membership function of which is defined in every $x \in \mathbf{R}$ by

$$\beta_n(\mathbb{A},\mathbb{B})(x) \triangleq \alpha_n(\mathbb{B},\mathbb{A})(x) \tag{4.74}$$

□

Thus, as we determine the *subsystem* validity of a stochastic system $\mathcal{K}$ with respect to a stochastic system $\mathcal{K}'$, then we look up for every point in the behaviour relation estimate of $\mathcal{K}$ the most similar point in the behaviour relation estimate of $\mathcal{K}'$, using the NEFPROX approximations of both behaviour relation estimates. However, as we determine the *supersystem* validity of $\mathcal{K}$ with respect to $\mathcal{K}'$, then we look up for every point in the behaviour relation estimate of $\mathcal{K}'$ the most similar point in the behaviour relation estimate of $\mathcal{K}$, using the same NEFPROX approximations. Using definitions 69 and 70, we now define a *fuzzy set of stochastic system validity propositions* as follows,

Definition 71 (Fuzzy set of stochastic system validity propositions)
For $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla} \subseteq \mathbf{R}^m \times \mathbf{R}^n$ and $\hat{\mathscr{B}}(\mathcal{K}')_{|\nabla'} \subseteq \mathbf{R}^m \times \mathbf{R}^n$ finite estimates of the behaviour of approximately stationary, steady-state and ergodic stochastic systems $\mathcal{K}$ and $\mathcal{K}'$ through the respective compatible behaviour orientations ∇ and ∇', $\mathcal{P}$ an (m,n)-parameter set-up for NEFPROX, and $\mathbb{A} \triangleq \mathbb{A}_{\mathcal{P}}(\hat{\mathscr{B}}(\mathcal{K})_{|\nabla})$ and $\mathbb{B} \triangleq \mathbb{A}_{\mathcal{P}}(\hat{\mathscr{B}}(\mathcal{K}')_{|\nabla'})$ NEFPROX approximations of $\hat{\mathscr{B}}(\mathcal{K})_{|\nabla}$ and $\hat{\mathscr{B}}(\mathcal{K}')_{|\nabla'}$ respectively, a *fuzzy set of stochastic system validity propositions of $\mathcal{K}$ with respect to $\mathcal{K}'$ of resolution n* is a fuzzy subset of $[0,1]$, denoted by $\alpha\beta_n(\mathbb{A},\mathbb{B})$, and defined by

$$\alpha\beta_n(\mathbb{A},\mathbb{B}) \triangleq \alpha_n(\mathbb{A},\mathbb{B}) \wedge \beta_n(\mathbb{A},\mathbb{B}) \tag{4.75}$$

□

We thus define the validity of a stochastic system $\mathcal{K}$ in view of another stochastic system $\mathcal{K}'$ through a fuzzy set, members of which are *propositions* on the extent that we consider the behaviour relation estimate that we have obtained of $\mathcal{K}$, *equal* to the behaviour relation estimate that we have obtained of $\mathcal{K}'$, labelled by a *degree of truth.*

Conclusion

In this research report, we have developed the core parts of an integrated, fuzzy set and system theoretic approach to validate stochastic systems. First, in section 1, we have set up a framework for our validation method and defined in an informal way the *operands* of our validation technique, i.e. primary model and real system behaviour. We discussed a modelling hierarchy, and positioned the arguments of our validation method in the hierarchy. We also differentiated *validation* from *verification*. Then, in section 2, we covered some basic aspects of general systems theory, and we presented a structured discrete event system construct. We then employed this construct in section 3 to develop a stochastic system construct, and emphasised that we like to identify stochastic system behaviour with a relation between sets that hold tuples of (integrated) statistics of different stochastic processes. We concluded section 3 by defining the conditions that a stochastic system must fulfil for it to be validatable with our approach. Finally, in section 4, we presented the core aspects of our fuzzy set theoretic validation method. We indicated how we employ a resemblance relation concept in fuzzy set theory, and a neuro-fuzzy function approximation algorithm, to derive the truth value of statements on the validity of a stochastic system.

We wrote a bundle of C-programs to compute estimates of statistics of stochastic processes, cumulative distributions of resemblance random variables, fuzzy sets of stochastic system validity propositions, etc. Also, we modified the source code of the NEFPROX algorithm to obtain rule hits of patterns, the input of which is propagated through a fuzzy neural network. We used these programs to experiment with our validation technique on an integrated example that involves the coin striking process. We intend to publish the results of these experiments in a future publication. Also, we like to further refine our technique, test it on a number of different validation scenarios and compare it with other validation techniques.

References

[1] J. Adem and J. Martens. Analysing and improving network punctuality at a belgian airline company using simulation. Technical Report 0140, Katholieke Universiteit Leuven, 2001.

[2] O. Balci. *Handbook of Simulation (edited by Banks J.)*, chapter 10: Verification, Validation and Testing, pages 335–393. Wiley, 1998.

[3] J. Banks, J.S. Carson, B.L. Nelson, and D.M. Mical. *Discrete Event System Simulation.* Prentice Hall, 2001.

[4] C.G. Cassandras and S. Lafortune. *Inroduction to Discrete Event Systems.* Kluwer Academic Publishers, 1999.

[5] P. Checkland. *Systems Thinking, Systems Practice.* Wiley, 1981.

[6] M. DeCock and E.E. Kerre. On (un)suitable fuzzy relations to model approximate equality. *Fuzzy Sets and Systems*, 133(2):181–192, 2003.

[7] J.L. Doob. *Stochastic Processes.* Wiley, 1990.

[8] J.L Doob. *Measure Theory.* Springer-Verlag, 1994.

[9] R. Fullér. *Neural Fuzzy Systems.* Institute for Advanced Management Systems Research, Abo, 1995.

[10] R.M. Gray. *Probability, Random Processes and Ergodic Properties - www-eu.stanford.edu/ gray*, 2001.

[11] R.M. Gray and L.D. Davisson. *Introduction to Statistical Signal Processing - www-eu.stanford.edu/ gray*, 2002.

[12] W.D. Kelton, R.P. Sadowski, and D.A. Sadowski. *Simulation with ARENA.* McGraw-Hill, 1998.

[13] E.E. Kerre. *Introduction to the Basic Principles of Fuzzy Set Theory and Some of its Applications.* Communication and Cognition, 1993.

[14] J.P.C. Kleijnen and W. Van Groenendaal. *Simulation, A Statistical Perspective.* Wiley, 1992.

[15] M. Landry, J.-L. Malouin, and M. Oral. Model validation in operations research. *European Journal of Operations Research*, 14:207–220, 1983.

[16] A.M. Law and W.D. Kelton. *Simulation Modeling and Analysis.* McGraw-Hill, 2000.

[17] J.-L. Malouin and M. Landry. The mirage of universal methods in system design. *Journal of Applied System Analysis*, 10:47–62, 1983.

[18] J. Martens, G. Weets, J. Vanthienen, and C. Mues. Improving a neuro-fuzzy classifier using exploratory factor analysis. *International Journal of Intelligent System*, 15(8):785–800, 2000.

[19] M. D. Mesarovic and T. Yasuhiko. *General Systems Theory.* Academic Press, 1975.

[20] R.E. Nance. The conical methodology and the evolution of simulation model development. *Annals of Operations Research*, 53:1–46, 1994.

[21] D. Nauck, F. Klawonn, and R. Kruse. *Neuro-Fuzzy Systems.* Wiley, 1997.

[22] M. Oral and O. Kettani. The facets of the modeling and validation process in operations research. *European Journal of Operations Research*, 66(2):216–234, 1993.

[23] A. Papoulis. *Probability, Random Variables and Stochastic Processes.* McGraw-Hill, 1991.

[24] E. Parzen. *Stochastic Processes.* Holden-Day, 1967.

[25] C.D. Pegden, R.E. Shannon, and R.P. Sadowski. *Introduction to Simulation Using SIMAN.* McGraw-Hill, 1995.

[26] G.F. Smith. Towards a heuristic theory of problem structuring. *Management Science*, 34(12):1489–1506, 1988.

[27] N. Vaillant. *Probability Tutorials - www.probability.net.*

[28] A.W. Wymore. *A Mathematical Theory of Systems Engineering.* Wiley, 1967.

[29] L.A Zadeh. A theory of approximate reasoning. *Machine Intelligence*, 9:149–194, 1979.

[30] L.A. Zadeh and C.A. Desoer. *Linear System Theory.* McGraw-Hill, 1963.

[31] B.P. Zeigler. *Theory of Modeling and Simulation.* Academic Press, 1976.

[32] B.P. Zeigler, H. Praehofer, and T.G. Kim. *Theory of Modeling and Simulation: Integrating Discrete Event and Continuous Complex Dynamic Systems.* Academic Press, 2000.

[33] J.H. Zimmerman. *Fuzzy Set Theory and its Applications.* Kluwer Academic, 1991.