



Deposited via The University of Sheffield.

White Rose Research Online URL for this paper:

<https://eprints.whiterose.ac.uk/id/eprint/196630/>

Version: Accepted Version

Article:

Walkinshaw, N. and Hierons, R. (2023) Modelling second-order uncertainty in state machines. *IEEE Transactions on Software Engineering*, 49 (5). pp. 3261-3276. ISSN: 0098-5589

<https://doi.org/10.1109/TSE.2023.3250835>

© 2023 The Authors. Except as otherwise noted, this author-accepted version of a journal article published in *IEEE Transactions on Software Engineering* is made available via the University of Sheffield Research Publications and Copyright Policy under the terms of the Creative Commons Attribution 4.0 International License (CC-BY 4.0), which permits unrestricted use, distribution and reproduction in any medium, provided the original work is properly cited. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>

Reuse

This article is distributed under the terms of the Creative Commons Attribution (CC BY) licence. This licence allows you to distribute, remix, tweak, and build upon the work, even commercially, as long as you credit the authors for the original work. More information and the full terms of the licence here: <https://creativecommons.org/licenses/>

Takedown

If you consider content in White Rose Research Online to be in breach of UK law, please notify us by emailing eprints@whiterose.ac.uk including the URL of the record and the reason for the withdrawal request.

Modelling Second-Order Uncertainty in State Machines

Neil Walkinshaw, Robert M. Hierons

Abstract—Modelling the behaviour of state-based systems can be challenging, especially when the modeller is not entirely certain about its intended interactions with the user or the environment. Currently, it is possible to associate a stated level of uncertainty with a given event by attaching probabilities to transitions (producing ‘Probabilistic State Machines’). This captures the ‘First-order uncertainty’ - the (un-)certainty that a given event will occur. However, this does not permit the modeller to capture their own uncertainty (or lack thereof) about that stated probability - also known as ‘Second-order uncertainty’. In this paper we introduce a generalisation of probabilistic finite state machines that makes it possible to incorporate this important additional dimension of uncertainty. For this we adopt a formalism for reasoning about uncertainty called Subjective Logic. We present an algorithm to create these enhanced state machines automatically from a conventional state machine and a set of observed sequences. We show how this approach can be used for reverse-engineering predictive state machines from traces.

Index Terms—State machine, Second order uncertainty, Subjective logic, Inference, Test prioritization



1 INTRODUCTION

State machines are commonly used to model systems where the behaviour can be characterised in terms of sequences of steps. These systems tend to encompass network protocols, GUIs, and the behaviour of reactive systems such as cyber-physical systems. The presence of a state machine model allows the application of powerful algorithms to a range of verification and testing approaches [1]. Numerous languages and notations have been developed for a variety of domains. Popular tools such as Simulink StateFlow¹, CDP Studio², and Eggplant³ are based on state machines, and are used to model systems ranging from low-level control systems to high-level business processes.

In their recent review of uncertainty representation in software models [2], Troya *et al.* emphasise the importance of being able to capture uncertainty: “*Uncertainty is an inherent property of any system that operates in a real environment or that interacts with physical elements or with humans.*”. They note that there is a need for software engineers to be able to express uncertainty within a model in a suitable way. They also note the need to be able to explicitly analyse uncertainty within these models, so that it can be handled appropriately.

In state machines uncertainty might arise if the system being modelled uses probabilistic algorithms or the system’s environment is stochastic or poorly understood. Such systems are conventionally represented as probabilistic state machines [3], where transitions are labelled with probabilities. This information can be leveraged to, for example, verify probabilistic properties [4], or to tailor test sets [5].

If we have a state with two outgoing transitions, and the modeller knows that the choice of transition comes down

to a random process that amounts to a fair coin-flip (c.f. the IEEE1394 FireWire root contention protocol [6]), it is clear that each transition has a probability of 0.5. This sort of uncertainty is referred to as ‘first-order uncertainty’ [7].

It can, however, easily be the case that such probabilities are subject to uncertainty. The modeller might not have prior knowledge or empirical observations upon which to gauge the probabilities of events in a given state (epistemic uncertainty [8]). Even if they have plenty of observations, these might be subject to random effects that make it difficult to definitively pin-down a specific uncertainty value (aleatory uncertainty [8]). In either case, there is uncertainty surrounding the probability in question. This is generally referred to as ‘second-order uncertainty’ [7].

There is a qualitative difference between probabilities that are well-founded (based on, for example, 15,000 observations) and probabilities that are speculative and uncertain (a guess that each transition has a likelihood of 0.5, based on no prior knowledge). However, Probabilistic FSMs (PFSMs) do not capture such second-order (un-)certainties. As a result, if we are using a PFSM, we can only approximate the trustworthiness of a prediction if we understand how it was generated (obtaining associated execution samples if these were used to estimate the probabilities). This could at best be used to produce a generic assessment of trustworthiness of the PFSM, but there is no apparent means of deriving levels of trustworthiness for specific paths through the model.

Subjective Logic [9] is a relatively recent framework that was developed to reason about such uncertainties. It provides a formalism that captures a probability and associated uncertainty as a ‘subjective opinion’. It also provides a variety of operators that enable us to combine subjective opinions (conjunction, disjunction, fusion, etc.), whilst also computing the associated uncertainty.

In this paper we use Subjective Logic to reason about the uncertainty pertaining to probabilities of events occurring at different states in a state machine. We show how Subjective

• Department of Computer Science at The University of Sheffield, Sheffield, UK. E-mail: {n.walkinshaw,r.hierons}@sheffield.ac.uk

1. <https://uk.mathworks.com/products/stateflow.html>

2. <https://cdpstudio.com/>

3. <https://www.eggplantsoftware.com/>

Logic can be used to reason about the cumulative uncertainty of sequences of events in a model. As a result, any path through a model corresponding to a sequence of events can be explicitly associated with a corresponding probability and a level of ‘trustworthiness’ in this probability.

The paper is motivated by a specific scenario: We have an FSM model of a system and a set of traces corresponding to system execution sequences (e.g. execution logs). This situation arises in most settings where Machine Learning is used to infer a model from traces (examples from adaptive GUI / Android app testing are cited here [10], [11], but there are similar approaches for other areas). The ability to associate trustworthiness with predicted likelihoods of sequences would enable us to leverage this additional information, to avoid acting on untrustworthy predictions, or even to correct predictions. As an example of a potential application area, in their work on Android testing, Choi *et al.* [10] trace paths across inferred FSMs to identify test sequences that are as long as possible (to avoid frequent expensive restarts). In this setting, one might leverage the second-order uncertainty surrounding the feasibility of a sequence to avoid attempting impossible paths.

To enable this we propose *Subjective Opinion State Machines (SOSMs)*. We show how these can be automatically derived from a state machine and an accompanying set of traces. In an SOSM, for state q , there is a subjective opinion that captures probability, coupled with a degree of second-order uncertainty, of a given event occurring in state q .

The contributions of this paper are as follows:

- We introduce SOSMs, which generalise PFSMs by replacing probabilities by subjective opinion, enabling the ‘belief’ corresponding to a transition to be accompanied by an explicit level of uncertainty. (*Section 3*)
- We produce an algorithm that generates an SOSM from a traditional (non-probabilistic) state machine and a set of traces. This makes our approach applicable to a broad range of Software Engineering settings.
- We show how Subjective Logic multiplication can be used to reason about the likelihood (and corresponding uncertainty) of paths through SOSMs. (*Section 3.3*)
- We show how SOSMs can be used to improve predictions from reverse-engineered models (*Section 4*).
- We present an empirical study that evaluates the approach on 47 published state machine models.

2 BACKGROUND

We start with definitions of state machines and probabilistic state machines and then describe Subjective Logic.

2.1 State Machines and Probabilistic State Machines

Definition 2.1. A Finite State Machine (FSM) is defined by a tuple (Q, q_0, A, δ, Q_F) . Q is the finite set of states, $q_0 \in Q$ is the initial state, A is the finite alphabet. $\delta : Q \times A \rightarrow Q$ is the state transition function, and $Q_F \subseteq Q$ represents the set of accepting states.

An FSM processes a sequence of elements from A ; we use A^* to denote the set of such sequences. Given FSM F

and string $x \in A^*$, x is *accepted* if there exists a corresponding path $(q_0, x_0, q_1, x_1, \dots, x_n, q_n)$, such that $q_n \in Q_F$. If there is no such path, or $q_n \notin Q_F$ then x is ‘rejected’.

Observe that the FSMs used here are deterministic: for each state and event, there is at most one possible next state. However, a non-deterministic FSM can always be mapped to an equivalent deterministic FSM.

Definition 2.2. A Probabilistic Finite State Machine (PFSM)⁴ is defined by a tuple $(Q, q_0, A, \delta, Q_F, P)$. Q, q_0, A, δ and Q_F are defined as in Definition 2.1. The additional element P is a transition probability function $Q \times A \rightarrow [0, 1]$. For all $q \in Q$, $\sum_{a \in A} P(q, a) = 1$.

Probabilities model how likely it is that a particular event a occurs in state q . We can define a function P_T that gives the probability of a transition occurring from a given state.

$$P_T(q, a, q') = \begin{cases} P(q, a) & \text{if } \delta(q, a) = q'; \\ 0 & \text{otherwise.} \end{cases}$$

Given PFSM PF and string $x \in A^*$, one can derive probability $p(x)$ by tracing the corresponding path $(q_0, x_0, q_1, x_1, \dots, x_n, q_n)$. If no such path exists, or $q_n \notin Q_F$, then $p(x) = 0$. Otherwise, $p(x) = \prod_{i=1}^n P_T(q_{i-1}, x_i, q_i)$.

2.2 First- and Second-Order Probabilities

There have been many attempts to offer different taxonomies of uncertainty, several of which are discussed by Troya *et al.* in their survey of uncertainty in software models [2]. Perhaps the most common way of classifying uncertainty is to divide it into ‘epistemic’ or ‘aleatory’ uncertainty [8]. Epistemic uncertainty refers to the situation where uncertainty arises because of a fundamental lack of information or knowledge about the phenomenon in question. Aleatory uncertainty [8] on the other hand refers to the intrinsic variability or randomness of a phenomenon.

The uncertainty intrinsic to a probabilistic statement (e.g. “*I believe that there is a 60% chance my local team will win the match today.*”) is captured as a ‘first-order’ probability (this is irrespective of whether the uncertainty is aleatory or epistemic [9]). However, our interpretation of that statement might change if we learn that it is made by somebody who has never seen the team in question play, as opposed to someone who has coached the team for many years. This degree of trust (or lack thereof) in a first-order probability is referred to as a ‘second-order probability’ [9].

This notion, that first-order probabilities can be subject to uncertainty is well established. The question of how to reason about probabilities that are subject to different levels of uncertainty is a longstanding one. In Boole’s 1854 “An Investigation of the Laws of Thought” [12], he criticises the assumption that it is possible to “... assign the probabilities with perfect rigour ...” even when the problem “... admits only of an indefinite solution”.

4. This is a restricted version of the PFAs of Vidal *et al.* [3], where there can be multiple initial states. In such models, each initial state is given a probability and each final state is again mapped to a probability.

2.3 Subjective Logic

There have been numerous efforts over the last 60 years to capture second-order probabilities and to reason about them. Notable examples include Dempster Shafer Theory [13], Evidential Reasoning [14], the Imprecise Dirichlet Model [15], and Fuzzy Logic [16]. In recent years Subjective Logic has emerged as a particularly flexible and general basis upon which to reason about probabilities and their respective uncertainties. It subsumes Dempster-Shafer theory, and incorporates a range of operators that make it a generalisation of traditional probabilistic logic [9].

Subjective Logic is based on the premise that it is possible to associate traditional propositions with a belief that the proposition is true and a level of uncertainty, representing the epistemic uncertainty in the assessment of the proposition. These ‘subjective opinions’ can be reasoned about and combined with a variety of operators. Below we introduce the fragments of Subjective Logic used in this paper (a more complete reference is available in Jøsang’s book [9]).

2.3.1 Subjective Opinions

Subjective opinions express beliefs about the truth of propositions. The key difference between subjective opinions and conventional probabilistic statements is the ability to associate them with an explicit level of second-order uncertainty. With reference to the statement of confidence in the performance of a local team (Section 2.2), a subjective opinion can also capture a level of second-order uncertainty associated with that statement: *I believe that there is a 60% chance my local team will win the match today, but I only give my assessment a 10% chance of being accurate [perhaps because I have no experience of watching them or their opposition play]*’.

There are three types of subjective opinion [9]: Binomial opinions, multinomial opinions, and hypernomial opinions. Binomial opinions capture a belief (and associated uncertainty) regarding a single proposition that can be true or false. Multinomial opinions generalise this: the proposition in question can take on one of several possible values (not just true or false). Hypernomial opinions further generalise this to enable the expression of a belief that one of a set of values is true, without the need to identify a particular value which is believed to be true. In this paper we use the first two types: binomial and multinomial opinions. Throughout this paper we base our notation on Jøsang’s notation [9].

Definition 2.3. A *binomial subjective opinion* is an opinion over a binary domain $\mathbb{X} = \{x, \bar{x}\}$, where x is a random variable in \mathbb{X} . A binomial opinion about the truth of x is the ordered tuple: $\omega_x = (b_x, d_x, u_x, a_x)$, where:

- b_x is a belief mass in support of x being true.
- d_x is a belief mass in support of x being false (i.e. \bar{x} being true).
- u_x is a scalar uncertainty mass representing the ‘vacuity of evidence’.
- a_x is a ‘base rate’ - a prior probability of x without any evidence.

For an opinion to be valid, the following ‘additivity requirement’ must hold: $b_x + d_x + u_x = 1$.

The ‘base rate’ is akin to the ‘prior probability distribution’ in a Bayesian setting. To use our earlier example, it may

be common knowledge that our local team has traditionally won 90% of its games, so that the ‘a-priori’ probability of a win can be taken as 0.9 ($a_x = 0.9$). However, we might also know that for this match one of our key players is injured, which changes our assessment of the chances of a win, and introduces a high degree of uncertainty. We might only give a 15% chance that whatever we estimate is accurate ($u_x = 0.85$), but still marginally favour a win with a probability of 60%, which results in the remaining belief mass of 0.15 ($1-u_x$) being split $b_x = 0.09$, $d_x = 0.06$.

It is worth highlighting two notable opinions. First, $u_x = 1$ is a ‘vacuous opinion’ - nothing is known about x (there is zero belief mass). Second, $b_x = 1$ or $d_x = 1$: x is dogmatically true or false respectively (and so $u_x = 0$).

Definition 2.4. The ‘projected probability’ [9] for x is defined as: $P(x) = b_x + a_x u_x$. This represents the overall belief in x , factoring in the prior probability and the uncertainty.

In *multinomial* opinions, a belief is a distribution (across propositions) with an uncertainty in the whole distribution:

Definition 2.5. For a multinomial subjective opinion, let \mathbb{X} be a domain, where $|\mathbb{X}| > 2$. Let X be a random variable in \mathbb{X} . A multinomial opinion over the random variable X is the ordered triplet: $\omega_X = (b_X, u_X, a_X)$, where:

- b_X is a belief mass distribution over \mathbb{X} .
- u_X is a scalar uncertainty mass representing the ‘vacuity of evidence’.
- a_X is a prior probability distribution over \mathbb{X} .

For an opinion to be valid, the following ‘additivity requirement’ must hold: $u_X + \sum_{x \in \mathbb{X}} b_X(x) = 1$.

A vacuous opinion can be expressed by using $u_X = 1$ and $\forall x \in \mathbb{X} : b(x) = 0$. A traditional probability distribution occurs when $u_X = 0$.

Sometimes it can be helpful to ‘extract’ a belief about a single proposition from a multinomial opinion. This process is referred to as ‘coarsening’ [9], and works as follows.

Definition 2.6. Given a multinomial opinion $\omega_X = (b_X, u_X, a_X)$ where X is some random variable in \mathbb{X} , it is possible to *coarsen* it to a binomial opinion $\omega_x = (b_x, d_x, u_x, a_x)$ for some $x \in \mathbb{X}$ such that:

- b_x is the singleton belief of x in b_X .
- $d_x = 1 - (u_X + b_x)$
- $u_x = u_X$
- a_x is the singleton apriori belief of x in a_X .

A binomial opinion can be visualised as an equilateral barycentric triangle [17]. Here, each of the vertices in the triangle represents a maximum value for belief, disbelief and uncertainty respectively, with the minimum value sitting on the opposite axis. For example, the maximum value for Uncertainty is at the top vertex of the triangle, and the lowest value is on the mid-point of the Disbelief-Belief axis of the triangle. All possible binomial opinions sit within this triangle (this is ensured by the inherent constraint that all values sum to 1). Examples of barycentric triangles for two subjective logic opinions are shown in Figure 1.

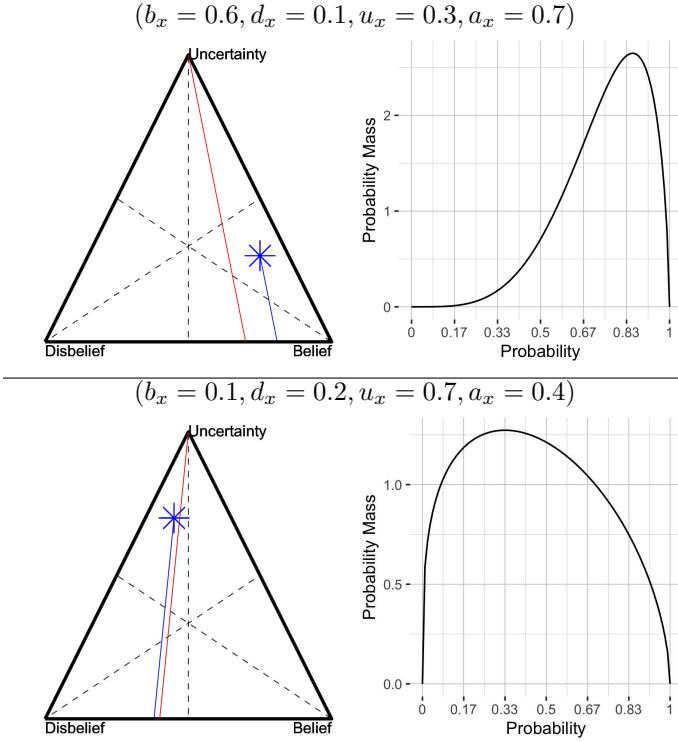


Fig. 1. Barycentric triangles and corresponding beta-distributions for two binomial subjective opinions. An opinion is captured by a coordinate within the triangle, where the distance from each edge represents the degree of uncertainty, belief and disbelief. The red line represents the a-priori probability, and the blue line represents the projected probability.

2.3.2 Subjective Opinions as Probability Distributions

When it comes to reasoning about uncertain, it is commonplace to visualise and understand them in terms of probability distributions. In the context of Software Engineering, for example, Duan *et al.* have shown how to represent safety properties as beta-distributions [18]. One limitation of distributions is that there is no established basis upon which to analytically combine them or reason about them.

One useful attribute of Subjective Logic is the capability to map opinions to probability distributions [9]. The components of a binomial opinion can be mapped to the parameters that are used to express beta-distributions, and the components of multinomial opinions can be mapped to the parameters that are used to express Dirichlet distributions. We will be largely concerned with binomial opinions, and so focus on using beta-distributions.

Definition 2.7. The *beta-distribution* refers to a family of distributions that are a continuous version of the binomial distribution, which make them appropriate for modelling probabilities [18]. Its probability density function is defined by two ‘shape’ parameters α and β , and is:

$$f(x; \alpha, \beta) = \frac{1}{B(\alpha, \beta)} x^{\alpha-1} (1-x)^{\beta-1}, 0 < x < 1$$

where $\alpha > 0, \beta > 0$ and $B(\alpha, \beta)$ is the beta function.

As α increases (and β is held constant), the mode of the distribution shifts to the right (towards 1). As β increases (and α is held constant), the mode shifts to the left (towards

0). When α and β are equal we get a symmetric distribution representing an approximate Gaussian shape. When $\alpha = \beta = 1$ we get a uniform distribution. When modelling a belief, a relative increase in α can be interpreted as an increase in supportive evidence, whereas a relative increase in β can be interpreted as an increase in evidence to the contrary. A distribution with a well-defined peak indicates that the probability is highly concentrated around a point, whereas a flatter distribution indicates higher uncertainty.

Jøsang has shown that there is a bijective mapping between subjective opinions and beta-distributions [17]. It is therefore possible to interpret a subjective opinion ‘coordinate’ as a continuous distribution, where ‘density’ represents probability. This mapping (with respect to some x) is achieved by reformulating the definition of α and β , so that these are explicitly linked to evidence that either supports or contradicts a belief:

Definition 2.8. Jøsang considers the following parameters r_x, s_x, a_x , and W [9]. r_x represents the number of observations in support of x , s_x represents the number of observations that contradict x . a_x is the a-priori belief in x as defined above. Finally W is the weighting to be given to a_x . The α and β parameters are defined by: $\alpha = r_x + a_x W$ and $\beta = s_x + (1 - a_x) W$.

From this, the bijective relation derived by Jøsang [9] is shown⁵ in Definition 2.9. The non-informative prior weight W is set to $W = 2$, because of the requirement that a vacuous opinion is mapped to a uniform beta-distribution in the case of a default base rate $a_x = \frac{1}{2}$ [9].

Definition 2.9. The bijective mapping between a binomial opinion (b_x, d_x, u_x, a_x) and a beta-distribution as expressed by the parameters r_x, s_x, a_x and W (see Definition 2.8) is as follows.

Binomial subjective opinion	beta distribution
$b_x = \frac{r_x}{W+r_x+s_x}$	$r_x = \frac{b_x W}{u_x}$
$d_x = \frac{s_x}{W+r_x+s_x}$	$s_x = \frac{d_x W}{u_x}$
$u_x = \frac{W}{W+r_x+s_x}$	$1 = b_x + d_x + u_x$

This relationship between subjective opinions and the beta-distribution has been explored within Software Engineering. In work on reasoning about uncertainty in safety cases, Duan *et al.* [18] consider the setting where different safety requirements are subject to varying degrees of evidence. They show how the beta-distribution can capture the relationship between evidence and uncertainty.

In Figure 1, the distributions corresponding to the two sample opinions are shown on the right. This illustrates how a higher level of belief will end up with a higher peak in the distribution towards a probability of 1. A higher level of uncertainty will result in a ‘flatter’ distribution.

2.3.3 Multiplying Subjective Opinions

Subjective Logic is associated with operators that combine subjective opinions. All of the traditional probabilistic logic operators have corresponding operators in Subjective Logic, making it possible to factor second-order uncertainty into

5. For space reasons we restrict ourselves to the case where $u_x \neq 0$. The other case can be found in [9].

traditional probabilistic reasoning techniques. Subjective Logic also has additional ‘fusion’ operators, making it possible to combine subjective opinions more flexibly [9]. We use the binomial multiplication operator [9].

Definition 2.10. Given binomial opinions $\omega_x = (b_x, d_x, u_x, a_x)$ and $\omega_y = (b_y, d_y, u_y, a_y)$, the binomial opinion $\omega_{x \wedge y}$ on the conjunction (multiplication) ($x \wedge y$) is:

$$\omega_{x \wedge y} = \begin{cases} b_{x \wedge y} = b_x b_y + \frac{(1-a_x)a_y b_x u_y + a_x(1-a_y)u_x b_y}{1-a_x a_y}, \\ d_{x \wedge y} = d_x + d_y - d_x d_y, \\ u_{x \wedge y} = u_x u_y + \frac{(1-a_y)b_x u_y + (1-a_x)u_x b_y}{1-a_x a_y}, \\ a_{x \wedge y} = a_x a_y \end{cases}$$

The operator involves the multiplication of the respective belief values. However, this product is then modulated according to the respective uncertainties and a-priori probabilities of the two beliefs. A more elaborate discussion of this operator is available elsewhere⁶ [9]. One useful note pertaining to the validity of the operator is the property, observed by Jøsang, that the projected probability of a subjective opinion (Definition 2.4) of the product computed from this operator is always equal to the expected probability of the product of the equivalent beta-distributions [9].

3 SUBJECTIVE OPINION STATE MACHINES

Probabilities in a PFSM can be associated with a considerable amount of second-order uncertainty. For example, where the state machine is created by a human modeller, they might be less sure about the probabilities governing some aspects of system behaviour than others. In a setting where the state machine is inferred by some inference algorithm [19], the probabilities labelling transitions would be entirely dependent on the quality and amount of training data, which will vary from one transition to another.

In both cases, it would be helpful to convey the level of second-order (un-)certainty alongside the probability for a transition, or a path through the machine. If an inferred state machine was used in an operational (e.g. automotive [20]) context, it would be helpful to distinguish between predictions with a low degree of uncertainty, and those that are entirely speculative (and thus entirely uncertain). Conventional PFSMs do not offer such a distinction.

In this section we introduce Subjective Opinion State Machines (SOSMs), where transitions can be associated with explicit levels of uncertainty. We also introduce a technique whereby traditional FSMs that are accompanied by a set of traces can be used to automatically derive SOSMs. This means that our approach can be used as a post-processing step for any technique that seeks to reason about a state machine with respect to a set of observations, such as testing techniques or state machine inference techniques.

For the sake of illustration, we use a toy example of a simple editor, where the basic FSM is shown in Figure 2. A goal in this scenario could, for example, be to probabilistically model a user’s interaction with the editor in a similar vein to the work on PFSM inference for GUIs [21], [22].

6. A visual demo of the multiplication operator, along with others, is available here: <https://folk.universitetetioslo.no/josang/sl/Op.html>

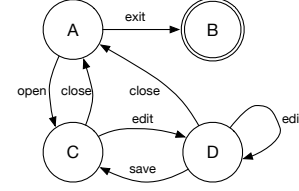


Fig. 2. Toy example of an editor, represented as a traditional FSM.

TABLE 1
Example of a multinomial subjective opinion for state D.

	open	close	edit	save	exit
a_A^D	0	0.33	0.33	0.33	0
b_A^D	0	0.1	0.6	0.3	0
u_A^D	0.2				

3.1 Definition of Subjective Opinion Machines

A Subjective Opinion State Machine (SOSM) combines an FSM structure with subjective opinions. Each state is linked to a multinomial subjective opinion, which captures the belief that a given element of the alphabet can be consumed at that state. Being a subjective-opinion, this includes an explicit level of second-order uncertainty.

Definition 3.1. An SOSM is defined as a tuple $(Q, q_0, A, \delta, Q_F, \Omega, S)$. Elements Q, q_0, A, δ and Q_F are defined as in an FSM (Section 2.1). Ω is a set of multinomial subjective opinions where $\mathbb{X} = A$ (Definition 2.5). $S : Q \rightarrow \Omega$ is a function from states to subjective opinions over elements in A and is defined for all $q \in Q$, where q has at least one outgoing transition. The subjective opinion for state $q \in Q$ is denoted ω^q . By default, we assume a uniform distribution of prior probabilities over the elements in A that label transitions from q .

As with PFSMs, we are interested in the likelihood of an event in A occurring when in a state s . However, we also want to capture the associated second-order uncertainty.

As an example, consider state D in our editor example (Figure 2). We know that some events in A (exit and open) are impossible from state D, so their a-priori probabilities and their belief values are 0. Let us suppose that we have no prior reason to believe that one of the remaining possible events would be more likely than the other in state D. This means that the events save, close, and edit have an a-priori possibility of $\frac{1}{3}$ each. During operation of the program, however, we observe that when in state D the edit event occurs with 60% probability, the save event with a 30% probability, and the close event with a 10% probability. However, we have only observed state D a small number of times and would only rate our certainty in these probabilities at 20%. The corresponding subjective opinion is shown in Table 1.

3.2 Derivation of SOSMs from FSMs and Traces

Given an FSM M , there is a well-established method for taking a set of observations of sequences accepted by M and deriving probabilities for each transition [23] (Chapter 17.1). This can be achieved by counting the number of times each

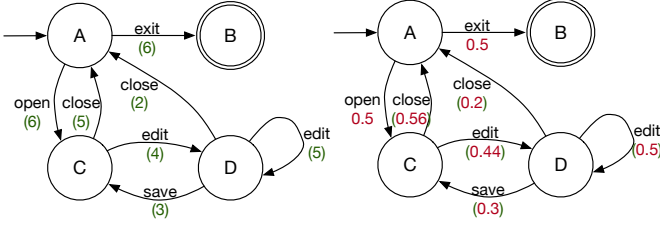


Fig. 3. Calculating traditional transition probabilities from frequencies.

transition is traversed and dividing a transition's frequency by the sum of frequencies across all transitions from its source state. The process is illustrated in Figure 3 in which, for a state q and event a , the machine on the left gives the number of times that the traces included a occurring in q ; the machine on the right is the result of normalising these for each state so that they form probabilities (ie the sum of the values, over transitions leaving q , is 1).

As an example, let us assume that we have observed the following sequences for the FSM in Figure 2.

open, edit, edit, edit, edit, save, close, exit
 open, close, exit
 open, edit, edit, close, exit
 exit
 open, edit, edit, save, close, exit
 open, edit, save, close, open, close, exit

Figure 3 gives frequencies and probabilities. These are based on a small sample and so there is significant *second-order* uncertainty: an additional sequence could lead to very different probabilities. In addition, usage can vary significantly between users, leading to *aleatory* uncertainty [8].

Our approach of deriving SOSMs from traces is similar to the approach for PFSMs described above. However, instead of using trace data to derive a simple probability (which cannot capture second-order uncertainty), we use the same data to produce a subjective opinion for each state.

The approach described here was devised with a specific application in mind: The derivation of SOSMs to make predictions of software behaviour. Thus, certain design choices (particularly around the derivation of a-priori distributions for subjective opinions and the calculation of uncertainties) might not be appropriate for other applications.

We start by providing an overview of the algorithm, following this by elaborating on (1) the parameter α used to calculate the uncertainty for each subjective opinion, and (2) the a-priori distributions for each subjective opinion.

The algorithm: The process is captured in Algorithm 1. We start in lines 3-11 by creating, for each state, a distribution of frequencies spanning the alphabet of the machine (lines 4-6) and an a-priori distribution that attributes an equal probability to all outgoing transitions (lines 7-10). In lines 11-13 we calculate the uncertainty by dividing the sum of transition frequencies by parameter α and subtracting from 1 (line 11); if the resultant value is negative then we assign zero to the uncertainty (line 13). We then calculate the belief distribution from the frequency values, normalising with respect to uncertainty (lines 14-16). The update only happens if there is at least one observation ($total \neq 0$). The

Algorithm 1: Computing the SOSM from traces.

Input: An FSM $F = (Q, q_0, A, \delta, Q_F)$, a multiset $Traces$ of strings such that $Traces \subseteq A^*$, and a certainty threshold α .

Output: An SOSM $O = (Q, q_0, A, \delta, Q_F, \Omega, S)$

```

1 begin
  /* Iterate through each state. */
2   for  $q \in Q$  do
3      $total \leftarrow 0$ ;
  /* Initialise frequency and apriori maps */
4     for  $a \in A$  do
5        $Freq(a) \leftarrow 0$ ;
6        $Apriori(a) \leftarrow 0$ ;
7     for  $(q, a, q') \in \delta$  do
  /* A uniform a-priori distribution across
  all outgoing transitions. */
8        $Apriori(a) \leftarrow \frac{1}{|\delta(q)|}$ ;
  /* Record frequencies with which transitions
  are traversed by traces. */
9        $Freq(a) \leftarrow Count(q, a, F, Traces)$ ;
10       $total \leftarrow total + Freq(a)$ ;
  /* calculate the uncertainty */
11       $uncertainty_q \leftarrow 1 - \frac{total}{\alpha}$ ;
12      if  $uncertainty_q < 0$  then
13         $uncertainty_q \leftarrow 0$ ;
14      for  $a \in A, total > 0$  do
  /* Calculate Belief value by dividing the
  frequency of  $a$  by the total if  $total > 0$ .
  */
15         $Belief_q(a) \leftarrow \frac{Freq(a)}{total}$ ;
  /* Factor in the uncertainty. */
16         $Belief_q(a) \leftarrow Belief_q(a) \times (1 - uncertainty_q)$ ;
17       $\omega_q \leftarrow (Belief_q, uncertainty_q, Apriori_q)$ ;
18       $\Omega \leftarrow \Omega \cup \{\omega_q\}$ ;
19       $S(q) \leftarrow \omega_q$ ;
20   return  $O$ ;
```

belief distribution, uncertainty value and a-priori distribution form a subjective opinion for the state (lines 17-19).

The α parameter to calculate uncertainty: The traditional approach to deriving probabilities for FSMs from observations involves calculating the relative frequencies (i.e. the distribution) of elements in A from each state. If we rely entirely on the resulting probability distribution over A , it is impossible to distinguish whether the distribution is the result of 10 data-points or 1000. If it is the result of 1000 data-points we would consider it to be much more reliable (i.e. have a much lower level of second-order uncertainty).

We incorporate a user-defined parameter α that represents the number of times a state must be traversed by a set of traces for the probabilities to be deemed to be certain. The value of α determines the balance between the probability mass that is associated with belief or disbelief, and the probability mass that is associated with uncertainty.

The choice of α depends to an extent on an understanding of the underlying model and on an intuition of what a sufficient number of observations should be for a given state. In practice, this may vary depending on usage context (e.g. the amount of data that is realistically available) and on the domain (e.g. with a particularly high threshold for safety-critical systems). For the models we refer to in our evaluation, our preliminary study on the selection of α (described in Appendix B) also suggests that it is safer to use higher values. Whereas selecting values of α that are too low can hide differences in terms of uncertainty between sequences, selecting high values of α (perhaps higher than they need to be) does not affect the relative balance between belief and disbelief and ensures that different levels of un-

TABLE 2

Example distributions for traces and corresponding frequencies used in Figure 3, with $\alpha = 20$.

State	Uncertainty	Distribution	open	edit	save	close	exit
A	0.4	Freq	6	0	0	0	6
		Belief	0.3	0	0	0	0.3
		Apriori	0.2	0.2	0.2	0.2	0.2
C	0.55	Freq	0	4	0	5	0
		Belief	0	0.2	0	0.25	0
		Apriori	0.2	0.2	0.2	0.2	0.2
D	0.5	Freq	0	5	3	2	0
		Belief	0	.25	0.15	0.1	0
		Apriori	0.2	0.2	0.2	0.2	0.2

certainty can be established for a larger range of sequences.

Distributions of a-priori probabilities: The distributions of a-priori probabilities for each state enable us to express any prior knowledge about the relative likelihoods of events occurring at a state. As with any Bayesian approach [24], these could be specified up-front (e.g. if our domain knowledge of the program suggests that ‘save’ operations occur much less frequently than ‘edit’ operations). As a default, we presume that there is no such prior knowledge and so we use a ‘uniform’ prior distribution across elements in A that are possible from that state. Any element in A that is not possible from a given state is given an a-priori probability of 0. For any n remaining outgoing elements, the respective a-priori probabilities are set to $\frac{1}{n}$.

Example: Table 2 contains the subjective opinions derived from the traces for the accepting states in the example (Figure 3). To take state A as an example, there are 12 observations of outgoing transitions. The a-priori belief is uniformly distributed; 0.2 per event. Uncertainty (line 11, Algorithm 1) is $1 - \frac{12}{20}$ (for $\alpha = 20$), which is 0.4. The remaining belief mass (0.6) is used to define the belief distribution. For A, ‘open’ and ‘exit’ are executed the same number of times, so we split this belief mass evenly between them (0.3 each), having a belief of 0 for other events. Note that, although we attribute a belief of 0 to *edit*, *save*, and *close*, this does *not* mean that they cannot happen. The uncertainty mass accounts for the possibility that they could occur - that we might have been mistaken in our attribution of beliefs. This is not captured in conventional PFSMs.

3.3 The Subjective Opinion of a Sequence of Events

Given an SOSM, Subjective Logic operators can be used to reason about the likelihood and uncertainty associated with events in different states. As with PFSMs, we can use multiplication to reason about the likelihood of sequences of events. Here we provide a relatively brief and formalised description of how these operators can be applied to SOSMs.

Given a PFSM, one can follow the path corresponding to a sequence and multiply together the probabilities that label the transitions. In an SOSM, it is possible to achieve the equivalent by multiplying together the multinomial opinions on the states of a path. However, if the alphabet has size greater than one then the multiplication of multinomial opinions is exponential in terms of the sequence length [9].

Fortunately, there is a work-around. When we encounter a state, we are only concerned with the probability of a *single* outgoing transition corresponding to the current symbol in

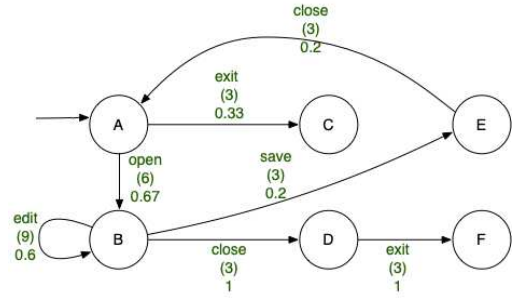


Fig. 4. Inferred state machine with probabilities and frequencies.

the sequence. As such, we can always coarsen a multinomial opinion to the Binomial opinion for the given event (Definition 2.6). Multiplying binomial opinions (Definition 2.10) is computationally much more efficient.

Given string $x_0, x_1, \dots, x_n \in A^*$ and SOSM SO , we trace the path $(q_0, x_0, q_1, \dots, x_n, q_{n+1})$. For a transition (q, x, q') , we obtain (multinomial) subjective opinion $S(q)$ for q , and then apply the binomial extraction procedure with respect to x (Definition 2.6). This leads to a sequence of binomial opinions $\omega_{x_0}^{q_0}, \omega_{x_1}^{q_1}, \dots, \omega_{x_n}^{q_n}$. The result can be obtained by multiplying these together with the binomial multiplication operator (Definition 2.10) as $\omega_{x_0}^{q_0} \times \omega_{x_1}^{q_1} \times \dots \times \omega_{x_n}^{q_n}$.

4 UNCERTAINTY IN INFERRED STATE MACHINES

State machine inference [23] has become an active area of research within Software Engineering, especially within the context of mining software or hardware specifications [20], [21], [22], [25], [26]. State machines are inferred from *samples* of observed sequences of events. Although the task of inferring a state machine is NP-hard [27], [28], numerous approaches have been developed that are capable of inferring approximately correct models [29] in polynomial time.

Before we describe our use of SOSMs with FSM inference, it is worth setting this into the context of existing similar approaches for inferring PFSMs. The task of PFSM inference can be split into two families [23], [30]: (1) augmenting a given FSM with probabilities to enable probabilistic predictions (also referred to as ‘probability estimation’ [23]), or (2) inferring the PFSM structure by taking into account the distributions of sequences in the trace data.

We focus on probability estimation (Section 3.2): to take an inferred FSM and its training set and estimate the probabilities (and in our case uncertainties) for different paths through the model. We use the approach described in Section 3.2 to convert an inferred FSM into an SOSM. We show how subjective opinions can help one interpret uncertain predictions and correcting incorrect predictions.

4.1 Motivation for the use of SOSMs

Probabilities in PFSMs are first-order. As discussed in Section 2.2, they are ‘point-values’ that capture the likelihood of a transition, but do not convey (second-order) uncertainty. In other words, if a state has two outgoing transitions, each of which has a probability of 0.5, we cannot tell from these numbers alone whether the probabilities are the guaranteed

equivalent of a fair coin-flip, or whether they are equal because there is no information to suggest otherwise.

Let us consider the model in Figure 4 – this was inferred by the EDSM state merging algorithm⁷ [31] from the traces used for Figure 3. Superficially, the probabilities look sensible. However, there are some striking features. For example, from state E (entered via a `save` event) there is a 100% probability that the next event will be `close`. This is consistent with the observations: only three of the six traces involve state E, and none of the traces feature the scenario where a file is saved and subsequently edited.

In practice we know that this does not reflect the actual system (see Figure 2): it is possible to carry on editing the document after saving but the sample of sequences used missed this scenario. There is a high degree of epistemic uncertainty [2]; having only observed the state three times, we lack sufficient evidence to conclude that there are no missing transitions from that state. As a result, this value of 100% is subject to a high degree of second-order uncertainty and should be treated with a substantial degree of caution.

This is where subjective opinions can play an important role. In the rest of this section we show how SOSMs can be computed from the same trace data. These make it possible to explicitly factor-in and reason about underlying uncertainties when referring to a prediction made.

4.2 Computing an SOSM for an Inferred State Machine

Uncertainty in subjective opinions conveys the extent to which probabilities for a given state should be trusted. The approach of deriving an SOSM from an FSM and a set of traces (Algorithm 1) can be easily used as a post-processing technique for any model (not just an inferred model). The underlying inference algorithm does not matter (it could be a ‘passive’ state merging algorithm [26] or an ‘active’ algorithm in the Angluin L^* style [32]).

The only requisite is that we have the FSM and the traces used. Commonly these include a set of ‘positive’ examples accepted by the inferred FSM (T^+) and (optionally) a set of ‘negative’ examples rejected by the inferred FSM (T^-). For the sake of simplicity we restrict ourselves to the scenario where traces are *prefix-closed*. In other words, no subsequences of any sequence in T^+ should be rejected, and for sequences in T^- only the final element of any sequence should cause the sequence to be rejected.

Once an FSM has been inferred, to obtain an SOSM, we apply the process described in Algorithm 1. The set *Traces* used is the union of T^+ and the set T^- , with T^- stripped to the accepting prefixes. The input *FSM* used in the algorithm corresponds to the FSM inferred from *Traces*.

Once the SOSM has been derived, it is possible to compute the second-order uncertainty associated with a path (Section 3.3). To illustrate this, consider the sequence: `open, edit, save, close`. Using the probabilities in Figure 4 (and treating the machine as a PFSM), this sequence would yield probability $0.67 \times 0.6 \times 0.2 \times 1 = 0.0804$, or 8.04%. This conveys nothing about the ‘trustworthiness’ of the probabilities.

7. We prevented over-generalisation by only commencing a merge and subsequent determinisation merges if a pair of states had 5 overlapping elements in their outgoing traces.

TABLE 3
Multinomial subjective opinions associated with each state in the inferred machine (see Figure 4), for $\alpha = 20$.

State	Multinomial Opinion					
	Uncertainty	open	edit	save	close	exit
A	0.55	0.3	0	0	0	0.15
B	0.25	0	0.45	0.15	0.15	0
D	0.85	0	0	0	0	0.15
E	0.85	0	0	0	0.15	0

TABLE 4
Binomial opinions corresponding to transitions, extracted from multinomial opinions in Table 3.

Transition	Binomial Opinion			
	Belief	Disbelief	Uncertainty	A-priori
$A \xrightarrow{\text{open}} B$	0.3	0.15	0.55	0.2
$B \xrightarrow{\text{edit}} B$	0.45	0.3	0.25	0.2
$B \xrightarrow{\text{save}} E$	0.15	0.6	0.25	0.2
$E \xrightarrow{\text{close}} A$	0.15	0	0.85	0.2
Product	0.01	0.76	0.23	< 0.01

Suppose we choose $\alpha = 20$ for Algorithm 1; we assume that 20 observations of a state is sufficient. The resulting subjective opinions are shown in Table 3 (recall from Definition 3.1 that we do not define subjective opinions for states without outgoing transitions, so C and F are omitted).

We obtain binomial opinions by *coarsening* the multinomial opinions. The resultant opinions are shown in Table 4. Whereas the probability $E \xrightarrow{\text{close}} A$ was 1 in the PFSM, our binomial opinion differs. Only a small amount of belief mass (0.15) is attributed to `close`. The bulk of the belief mass (0.85) is attributed to uncertainty; there may be other events that are possible, but haven’t occurred in our sample.

Multiplying binomial opinions (Definition 2.10) gives the opinion shown in Table 4 (bottom row), and visualised in Figure 5. The PFSM probability (8.04%) is close to the peak of the beta-distribution. However, uncertainty is also captured and is visualised in the beta-distribution (right of Figure 5). This shows that the probability could be significantly higher - the tail only levels off at 0 when $p > 0.5$.

4.3 Correcting Predictions

Inferred FSMs can be error-prone [26]: they can be too general (accept too many sequences) or too specific (reject

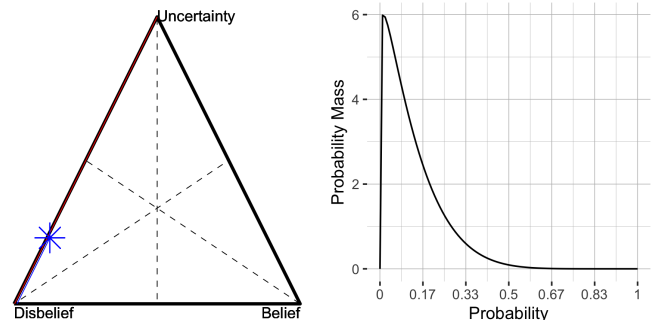


Fig. 5. Barycentric triangle and beta-distribution for opinion computed for path in Table 4.

Algorithm 2: Training a classifier from subj. opinions.

Input: An SOSM $SOSM = (Q, q_0, A, \delta, Q_F, \Omega, S)$, two sets of sequences T^+ and T^- , respectively the sets of positive and negative sequences that were used to infer the underlying FSM.

Output: A binary classifier CL

```

1 begin
2    $TS \leftarrow \emptyset$ ;
   //  $TS$  is a list of tuples, storing the subjective
   // opinion and accept / reject decision.
3    $index \leftarrow 0$ ;
   // Iterate through the set of positive sequences.
4   for  $t \in T^+$  do
5      $BinOpinions \leftarrow extractOpinions(t, SOSM)$ ;
6      $opinion \leftarrow multiply(BinOpinions)$ ;
7      $TS_i \leftarrow (opinion.belief, opinion.uncertainty, ACCEPT)$ ;
8      $i \leftarrow i + 1$ ;
   // Iterate through the set of negative sequences.
9   for  $t \in T^-$  do
10     $BinOpinions \leftarrow extractOpinions(t, SOSM)$ ;
11     $opinion \leftarrow multiply(BinOpinions)$ ;
12     $TS_i \leftarrow (opinion.belief, opinion.uncertainty, REJECT)$ ;
13     $i \leftarrow i + 1$ ;
14   $CL \leftarrow train(TS)$ ;
15  return  $CL$ ;

```

too many sequences). Once an FSM has been converted to an SOSM, however, a prediction of whether a sequence should be accepted is given a (second-order) uncertainty. If a sequence is classified as ‘Accept’, but has high uncertainty, there is a greater chance that it is actually a ‘Reject’.

The relationship between the subjective opinions calculated for sequences and the (in-)correctness of the corresponding prediction depends on the training data. In this subsection we show how it is possible, for an SOSM and associated sequence data, to infer a classifier. This makes it possible to improve the classification (accept/reject decision) accuracy for future sequences that we have not yet encountered and to correct any incorrect classifications that would have been made by the underlying FSM.

The training set for this classifier is ‘recycled’ from the set of sequences used to derive the SOSM. Instead of labelling sequences as ‘accept’ or ‘reject’, we label the corresponding subjective opinions (i.e. belief and uncertainty). The classifier trained on this set of decisions can then be used to provide an accept/reject decision on the basis of the underlying subjective opinion for sequences, even if they are not part of the original set of traces.

The training process is shown in Algorithm 2. First, each positive trace in T^+ is traced in the SOSM (line 5), the subjective opinion is calculated (line 6), and the resulting belief is linked to an ‘ACCEPT’ (line 7). This is repeated for the traces in T^- , with the resulting opinions linked to ‘REJECT’ (lines 9-13). The resulting set is used to train a classifier (line 14). The choice of classifier is flexible. In our evaluation we opt for a Random Forest classifier [33].

5 EVALUATION

The use of SOSMs for inferred automata is based on the rationale that the additional information in subjective opinions should be useful for interpreting their predictions. Subjective opinions should convey the trustworthiness of a prediction. Here we explore whether this is the case:

Research Question 1: Does the level of uncertainty associated with a prediction tend to be higher when a smaller number of traces has been used to derive the SOSM?

When working with inferred models on software engineering tasks, the second-order uncertainty pertaining to their predictions should reflect the amount of evidence from which the underlying model has been derived.

Research Question 2: To what extent, and in what respect, are the computed subjective opinions able to discriminate between correct and incorrect predictions?

If, as we expect, subjective opinions are able to discriminate between correct and incorrect predictions, this gives rise to the question of whether this ability is useful.

Research Question 3: Does the use of a classifier (as detailed in Algorithm 2) lead to more accurate predictions?

5.1 Methodology

5.1.1 RQ1 - Does the level of uncertainty increase as the number of traces used to derive SOSMs decreases

5.1.1.1 Subjects: For this RQ we applied the approach to log data recorded from a production system. For this we use the Android trace from the LogHub data set [34]. This consists of 1,555,005 OS-level messages.

An Android log comprises a list of events. Each event is associated with a process identifier, a thread identifier within that process, and the name of a function, along with other logging data. For this RQ we inferred an SOSM corresponding to each process, where the sequence of events for each thread within the process is treated as a trace.

Splitting the LogHub trace in this way led to 643 sets of traces (i.e. 643 processes) containing a total of 10,917 traces (sequences of events belonging to different threads). The number of traces per process forms a long-tailed distribution. The largest set had 2001 traces (followed by a process that had 1021 traces and another that had 527), and 187 processes had only one trace. We omitted single-trace state machines from this experiment, as explained below.

5.1.1.2 Analysis: For each Android process we inferred a state machine from the underlying trace set. For this we used the Evidence Driven State Merging (EDSM) algorithm with the Blue-Fringe search window [31], a baseline that is commonly used to evaluate passive state machine inference approaches [26]. In state merging algorithms it is possible to set a “minimal state match” score threshold, where a pair of states are only merged if a given number of transitions in their outgoing state transition structures overlap. We chose a threshold of 10, because values below 10 would tend to ‘collapse’ into trivial single-state machine.

We used the Leave One Out k -folds cross validation approach. One trace was kept aside, and the remainder was used to infer the machine. The trace left-out was used to compute a walk over the inferred machine, to produce the final uncertainty value for that walk. This process was repeated for all traces in the set. Since this process requires at least two traces in a set, we omitted models for which there was only a single trace (187 of the 643 processes).

5.1.2 RQ2 - Are the computed subjective opinions able to discriminate between correct and incorrect predictions?

5.1.2.1 Subjects: To answer this question (and RQ3), we required a set of reference models. We identified two sets of state machines that have been used in the state machine literature (mainly related to testing and verification). We

TABLE 5

Subject models. Models from the same source are grouped together, then ordered alphabetically.

Model	States	Transitions	Alphabet	Source
ActiveMQ	18	689	40	[37]
bbara	11	140	14	[35]
bbsse	17	448	28	[35]
bbtas	7	42	7	[35]
beecount	8	63	9	[35]
coffeemachine	7	48	8	[38]
cse	17	864	54	[35]
dk14	8	189	27	[35]
dk15	5	64	16	[35]
dk16	28	513	19	[35]
dk17	9	112	14	[35]
dk27	8	42	6	[35]
dk512	16	120	8	[35]
donfile	25	96	4	[35]
ex1	21	1580	79	[35]
ex4	15	240	16	[35]
ex6	9	216	27	[35]
keyb	20	1672	88	[35]
libressl	8	196	27	[39]
lion	5	36	9	[35]
lion9	10	72	8	[35]
OpenSSH	32	2480	80	[40]
planet	49	4560	95	[35]
planet1	49	4560	95	[35]
pma	25	1752	73	[35]
QUIC	8	70	10	[41]
s1	21	1860	93	[35]
s1488	49	10272	214	[35]
s1494	49	10128	211	[35]
s1a	21	1360	68	[35]
s27	7	132	22	[35]
s298	219	8066	37	[35]
s386	14	585	45	[35]
s510	48	2162	46	[35]
s8	6	20	4	[35]
s820	26	3575	143	[35]
s832	26	3750	150	[35]
sand	33	4576	143	[35]
shiftreg	9	32	4	[35]
sse	17	448	28	[35]
styr	31	2520	84	[35]
tav	5	88	22	[35]
tbk	33	2177	68	[35]
TCP_Client	16	450	30	[42]
TCP_Server	58	1710	30	[42]
tma	21	880	44	[35]
train11	12	88	8	[35]
train4	5	32	8	[35]

used the ACM/SIGDA benchmarks [35], a set of FSMs used in workshops between 1989 and 1993. We also used some machines from a collection curated by Neider *et al.* [36].

We used Mealy machines that were not inferred, but represented a genuine ‘ground truth’. We excluded those where we could not generate suitable traces. In total, 11 models were left out. These are listed in Appendix A. The final set of 47 models is listed in Table 5.

5.1.2.2 Generating traces: For each model we created a set of positive traces (sequences ending in a final state) and a set of negative traces (the final element was not accepted). For both sets, the maximum length was the depth of the machine (the length of the longest shortest-path) + 5. Traces were generated as random walks. We generated 200 random walks that were accepted, and 200 that were rejected. We omitted walks that were prefixes of existing sequences, ensuring that each sequence added information.

The ACM/SIGDA benchmark machines were fully-specified (every state had an outgoing transition for every element in the alphabet). In such machines, elements in the alphabet for which there should be no response by the machine tend to be modelled by silent loops. For these machines, we removed the silent self-loops, so that the corresponding sequences involving them would be rejected.

5.1.2.3 Model Inference and Accuracy Evaluation:

Since the objective was to investigate whether subjective opinions can provide information about (in-)accuracies, we used sets of traces that were *not* likely to be sufficient to infer an accurate model. We used k -folds cross validation for $k = 10$ to partition the set of traces into ten different batches of training and testing samples. To ensure that inferred models were inaccurate, we reduced the proportion of negative training sequences to 10% of what they should have been⁸, but used the full set of negative traces when testing the model (evaluating its accuracy) as part of the cross-validation process. For the inference step we used the same EDSM algorithm as was used for RQ1.

To measure model-accuracy, we used the following [43]:

- **Sensitivity**, also known as ‘Recall’, measures the proportion of sequences that were correctly accepted against the set of sequences that *should have been* accepted (including the set of False Negatives FN):

$$\text{Sensitivity} = \frac{|TP|}{|TP \cup FN|}.$$
- **Specificity** measures the ability of the model to correctly reject sequences by measuring the proportion of true negatives (TN) against the total set of sequences that *should have been* rejected (including false positives):

$$\text{Specificity} = \frac{|TN|}{|TN \cup FP|}.$$
- **BCR** (Balanced Classification Rate) is the harmonic mean of Sensitivity and Specificity: $BCR = \frac{2(\text{Sensitivity} * \text{Specificity})}{\text{Sensitivity} + \text{Specificity}}$.

5.1.2.4 Method: To choose a suitable value of α , for use in Algorithm 1, we carried out a small preliminary study (see Appendix B). This indicated that, for our model inference setting, $\alpha = 2000$ is appropriate. We will return to a more detailed study of how to calibrate α in future work.

We used the training set partitions (from the k -folds cross validation) to calculate the differences in uncertainty and belief with respect to sequences that are correctly or incorrectly classified. To measure this difference we used the Vargha-Delaney \hat{A}_{12} effect size [44]. This is a non-parametric measure, where the size is between 0 and 1 and represents the probability that two sets of classifications are equivalent.

The \hat{A}_{12} score related sequences correctly classified against sequences incorrectly classified. We calculated the score in terms of Belief and Uncertainty. For example, if we are measuring uncertainty, and we get $\hat{A}_{12} = 0.7$, this can be interpreted as ‘70% of correctly classified sequences have a higher uncertainty score than incorrectly classified sequences’. To be discriminative, effect size for either belief or uncertainty should be higher or lower than 0.5 (indicating no effect), with the utility increasing as the distance from 0.5 increases.

We divided the set of sequences into sequences classified by the inferred FSM as “reject” and sequences classified as

8. We found that 10% was the lowest limit for which none of the target models led to a trivial state machines with only one state.

“accept”. Bearing in mind that the level of belief specifically refers to the proposition that a sequence should be accepted, we would expect correctly accepted sequences to have a high level of belief, and correctly rejected sequences to have a low level of belief. For sequences that are accepted but should be rejected, there should be a lower level of belief than for sequences that are correctly accepted (i.e. $\hat{A}_{12} > 0.5$). Sequences that are incorrectly rejected should have a higher belief value than sequences that are correctly rejected. It was not clear what to expect of uncertainty.

We plotted the \hat{A}_{12} values on two charts: one for belief and one for uncertainty. For each model (x -axis), we plotted the \hat{A}_{12} for sequences that had been accepted and rejected. For subjective opinions to effectively discriminate between correct and incorrect predictions, we would expect \hat{A}_{12} to be ‘medium’ or ‘high’ for belief, uncertainty or both. We adopted the thresholds by Vargha and Delaney [44] for a “medium” effect size: $\hat{A}_{12} > 0.71$ or $\hat{A}_{12} < 0.29$ (0.5 represents a negligible effect size). To visualise the results we subtracted 0.5 from the \hat{A}_{12} value.

5.1.3 RQ3 - Does the use of a classifier lead to more accurate predictions?

The setup for RQ3 was identical to that of RQ2. However, we applied Algorithm 2 to produce a classifier from the SOSM and the original training set. We again used k -folds cross validation, with stratified sub-sampling to ensure that we properly distributed ‘ACCEPT’ and ‘REJECT’ sequences. We used a Random Forest classifier [33] to relate subjective opinions to classifications. For every evaluation phase in k -folds cross validation, we retained the classifications made by the inferred model against the training set. We compared the prediction made by our classifier against the ground truth and the prediction made by the original inferred FSM, computing the BCR for each case.

5.2 Results and Discussion

The software used to infer state machines, along with the target state machines used for RQs 2 and 3 is available online⁹.

5.2.1 RQ1: Relating uncertainty to amount of evidence

Figure 6 shows an apparent relationship between the number of traces used to infer a model, and the uncertainty computed for sequences. When fewer than 20 traces were involved in the inference, the uncertainty tends to be high. For all models that involved over 20 traces, the uncertainty is below 0.5 (mostly below 0.25). The Spearman-rank correlation coefficient is -0.68 (a ‘strong’ correlation).

RQ1: Uncertainty values tend to be higher for state machines inferred from low numbers of traces.

5.2.2 RQ2: Discriminating between correct and incorrect predictions

The \hat{A}_{12} results are shown in Figure 7. We separate the statistics for the accept- and reject-sequences because they

9. https://figshare.shef.ac.uk/articles/dataset/Modelling_Uncertainty_in_State_Based_Systems/14287040

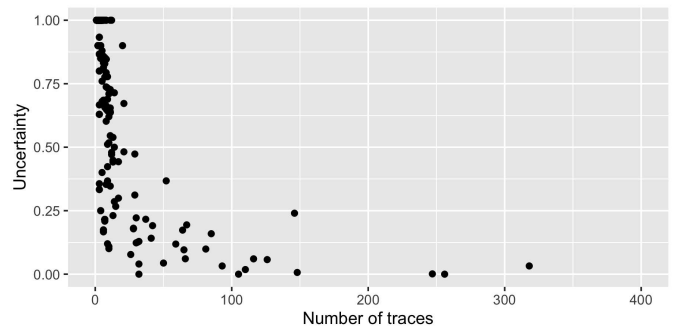


Fig. 6. Relationship between the number of traces used to infer a SOSM, and the uncertainty computed for sequences across it (restricted to models with 400 traces or fewer).

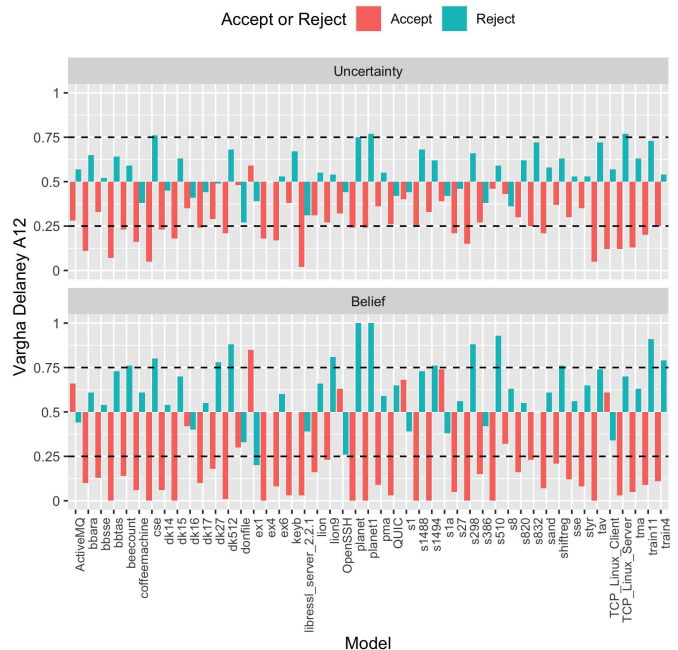


Fig. 7. \hat{A}_{12} effect sizes for belief and uncertainty (accepting and rejecting sequences). Dashed lines show thresholds for a medium effect size.

are different (for reasons which we discuss below). As a reminder of how to interpret them, consider the “Accept” and “Reject” bars for ActiveMQ in the ‘Uncertainty’ chart. The dashed horizontal bars indicate the threshold for ‘medium’ effect size in either direction. The red bar for ActiveMQ can be read as “The majority of sequences correctly classified as “accept” have a lower level of uncertainty than sequences that were incorrectly classified as “accept”. Only 28% ($\hat{A}_{12} = 0.28$) had a higher level of uncertainty”. The blue bar can be interpreted as “57% ($\hat{A}_{12} = 0.57$) of sequences correctly classified as ‘reject’ have a higher level of uncertainty than sequences incorrectly classified as ‘reject’”. In both cases the effect sizes would count as “low” [44] – they are not either above the 0.75 or below the 0.25 threshold to count as “medium”.

In 98% (46/47) of the subject systems, accepted sequences have an \hat{A}_{12} value of ≤ 0.5 for uncertainty (i.e., in the majority of cases, sequences that are correctly classified as accept have a lower level of uncertainty than sequences that are incorrectly classified). In 53% (25/47)

$\hat{A}_{12} \leq 0.25$. Thus, for sequences accepted by the inferred model, uncertainty tends to be a good indicator for whether the ‘accept’ classification is correct (incorrect classifications are associated with higher uncertainty).

For sequences rejected by the inferred model, uncertainty is a less reliable indicator. There is no obvious trend - \hat{A}_{12} values are > 0.5 in 68% (32/47) of models; they are ≤ 0.5 in 34% (16/47) of cases. The magnitude of the effect is only ‘medium’ (i.e. ≤ 0.25 or ≥ 0.75) in three instances.

Looking at levels of belief, in 87% (41/47) of cases, accept sequences have an \hat{A}_{12} value of ≤ 0.5 , and in 81% of cases (38/47) this is ≤ 0.25 . In other words, sequences that are correctly classified as accepts have a lower level of belief than incorrectly classified sequences. In summary, accept sequences that are incorrectly classified will tend to be associated with a higher level of belief as well as a higher level of uncertainty than correctly classified sequences.

For rejecting sequences, the inverse tends to true (i.e. incorrectly classified sequences tend to have a lower level of belief than correctly classified sequences). In 74% (35/47) of subjects, $\hat{A}_{12} > 0.5$ and in 30% of cases (14/47) $\hat{A}_{12} > 0.75$.

RQ2: Subjective opinions are capable of discriminating between correct and incorrect predictions.

Incorrect predictions of accept sequences tend to be associated with a higher level of uncertainty, as well as a higher level of belief. There is a less apparent effect for sequences that should be rejected.

Although the uncertainty results are intuitive (correctly accepted sequences have lower uncertainty than incorrectly accepted ones), the results for belief are somewhat counterintuitive. Correctly accepted sequences have lower belief values than incorrectly accepted ones (and correctly rejected sequences have higher belief values than incorrectly rejected ones). This can be explained by the fact that the EDSM algorithm (used to infer the models) always merges together states for which there is the most evidence in the underlying trace-sets. In the resulting state machine, we can always expect an especially high level of belief for any accepted sequence, and an especially low level of belief for any incorrect sequence. When the algorithm is inaccurate, and incorrectly merges together states, this will be reflected in levels of belief that are too high for sequences that should be rejected, and too low for sequences that should be accepted.

Recall that we deliberately used a setting where the the inferred models are inaccurate (see Section 5.1.2.3). Although the belief values can be misleading (reflecting inference mistakes made by the EDSM algorithm), second-order uncertainty values are more intuitive; correctly classified ‘accept’ sequences tend to have a lower level of uncertainty than incorrectly classified ones. This gives rise to the question of whether the differentiation between correctly and incorrectly classified sequences can be exploited (RQ3).

5.2.3 RQ3: Correcting incorrect predictions

Here we face the question of whether we can use the additional information available from subjective opinions to predict when a classification is incorrect, and correct the prediction? The results are shown in Table 6; the left half

TABLE 6

The Sensitivity, Specificity and BCR scores for the FSM, and equivalent scores produced with the Subjective-Opinion trained classifier (RQ2).

Model	Uncorrected			Corrected		
	Sens.	Spec.	BCR	Sens.	Spec.	BCR
ActiveMQ	0.94	0.54	0.68	0.79	0.81	0.80
bbara	0.93	0.69	0.79	0.94	0.97	0.95
bbsse	0.98	0.65	0.78	0.96	0.94	0.95
bbtas	0.92	0.65	0.76	0.96	0.99	0.98
beecount	0.99	0.66	0.79	0.98	0.96	0.97
coffeemachine	1.00	0.82	0.90	1.00	0.99	0.99
cse	0.97	0.72	0.83	0.97	0.99	0.98
dk14	0.95	0.60	0.73	0.94	0.95	0.95
dk15	0.96	0.70	0.81	0.97	1.00	0.98
dk16	0.88	0.21	0.34	0.69	0.63	0.66
dk17	0.97	0.55	0.70	0.94	0.91	0.93
dk27	1.00	0.61	0.76	0.98	0.95	0.97
dk512	0.98	0.58	0.73	0.97	0.97	0.97
donfile	0.99	0.53	0.69	0.92	0.76	0.83
ex1	0.92	0.60	0.72	0.82	0.81	0.81
ex4	1.00	0.70	0.83	1.00	0.99	0.99
ex6	0.98	0.67	0.80	0.97	0.93	0.95
keyb	0.96	0.78	0.86	0.96	0.98	0.97
libressl	0.98	0.82	0.89	0.98	0.99	0.99
lion	1.00	0.82	0.90	0.99	0.97	0.98
lion9	0.99	0.70	0.82	0.97	0.91	0.94
OpenSSH	0.96	0.76	0.85	0.92	0.91	0.92
planet	0.96	0.51	0.66	1.00	1.00	1.00
planet1	0.97	0.50	0.66	1.00	1.00	1.00
pma	0.95	0.57	0.72	0.92	0.90	0.91
QUIC	0.98	0.67	0.80	0.98	0.98	0.98
s1	0.90	0.48	0.63	0.73	0.74	0.73
s1488	0.96	0.82	0.88	0.98	0.99	0.98
s1494	0.95	0.82	0.88	0.98	1.00	0.99
s1a	0.90	0.48	0.63	0.74	0.76	0.75
s27	0.98	0.77	0.87	0.98	0.97	0.97
s298	0.81	0.47	0.59	0.94	0.97	0.95
s386	0.98	0.71	0.82	0.97	0.95	0.96
s510	1.00	0.63	0.77	1.00	1.00	1.00
s8	0.99	0.70	0.82	0.97	0.94	0.96
s820	0.97	0.85	0.90	0.96	0.97	0.96
s832	0.93	0.82	0.87	0.93	0.96	0.95
sand	0.91	0.69	0.78	0.90	0.90	0.90
shiftreg	0.98	0.45	0.61	0.97	0.90	0.93
sse	0.98	0.65	0.78	0.96	0.94	0.95
styr	0.94	0.65	0.77	0.93	0.93	0.93
tav	0.84	0.18	0.29	0.96	0.98	0.97
TCP_Client	0.95	0.70	0.81	0.91	0.94	0.93
TCP_Server	0.95	0.69	0.80	0.96	0.98	0.97
tma	0.98	0.56	0.71	0.97	0.94	0.95
train11	1.00	0.57	0.73	0.99	0.98	0.99
train4	1.00	0.66	0.79	0.99	0.97	0.98
Mean	0.96	0.64	0.76	0.93	0.94	0.94

of the table contains the ‘uncorrected’ results, and the right half contains the corrected counterparts.

The uncorrected scores show the prediction accuracy of using just the inferred FSM. These show, as expected, that there has been a significant degree of over-generalisation in FSM inference. Accepting too many sequences leads to a high Sensitivity (a mean of 0.96), but a low Specificity (mean of 0.64), with a mean BCR of 0.76.

The uncorrected mean BCR is high because Sensitivity tends to be in the high 90s. Specificity scores can be very low. There are 16 Specificity scores < 0.6 . For dk16 there is Specificity of 0.21. For planet and planet1 there is a Specificity of 0.51 and 0.5 respectively; both have a BCR of 0.66. For tav there is a Specificity of 0.18 and a BCR of 0.29.

With the use of the classifier trained on the subjective opinions, there is a marked improvement. Mean Sensitivity

slightly decreases by 0.03 to 0.93, but mean Specificity increases by a very large margin from 0.64 to 0.94, leading to an increase in mean BCR from 0.76 to 0.94. There is no trade-off; BCR increases for every model, and often by a significant margin. This extends to the cases mentioned above for which the initial inference produced especially inaccurate results. For example for *tav* the original BCR score is 0.29 (arising from a poor Specificity score of 0.18). The corrected model produces a BCR of 0.97, with a Specificity of 0.98.

RQ3: Subjective opinions can be used to correct incorrect predictions from inferred state machines.

The extent to which the additional information offered through subjective opinions can improve predictions is worth highlighting. The mean improvement of the BCR score is 18% (minimum of 6% and maximum of 67%). In the case of *planet* and *planet1*, BCR improves from 0.66 to 1.

Increases in accuracy result from improved Specificity. The inferred state machines tended to over-generalised. However, for the ‘corrected’ versions, a larger proportion of sequences that had been falsely accepted (false-positives) were instead correctly rejected (true-negatives). In a software-engineering context where, for example, the inferred models are used to identify candidate test cases [10], [11], this would lead to a more focussed set of candidates, avoiding candidates that are infeasible in practice.

5.3 Threats to Validity

Internal Validity: For all RQs we used the EDSM Blue-Fringe algorithm [31] and this choice may have affected the results. In future work we will investigate the use of different learning algorithms to infer SOSMs.

External Validity: For RQ1 we based our findings on state machines inferred from logs for Android processes, derived from a single system [34]. There is the possibility that the same findings might not hold for other classes of traces, and exploring this will be the subject of future work.

The state machines for RQs 2 and 3 were drawn from two collections [35], [36], raising the question of whether they are more generally representative. This threat was mitigated by the results being similar for the two collections. In addition, the models are highly diverse (see Table 5). Some are small and simple (e.g. *lion* has 5 states, 36 transitions and an alphabet of 9), but others are much larger (e.g. *s298* with 219 states, 8066 transitions and an alphabet of 46).

Sensitivity, Specificity, and BCR scores were calculated with respect to 10-folds Cross-Validation, mitigating against overfitting or selection bias. However, the results have to be interpreted as being valid with respect to traces sampled according to the probabilities attached to the transitions in the models. Although there are measures that assess the structural accuracy of inferred models (as opposed to language accuracy) [43], these will need to be enhanced to accommodate probabilities, and fit into future work.

6 RELATED WORK

6.1 Second-Order Uncertainty in Software Engineering

Several recent developments support reasoning about uncertainty in Software Engineering [2], particularly uncer-

tainty arising from noisy, partial and skewed distributions in empirical studies. The rigidity of “traditional” statistical analyses has often led to findings that have been difficult to justify and explain. The rise of Bayesian analysis, recently illustrated by *Furia et al.* [24] and *Dorn et al.* [45] offers a more robust and explainable basis for managing this uncertainty. Several research efforts have specifically sought to focus on representing second-order uncertainty.

Recent work by Walkinshaw and Shepperd [46] showed how outcomes from empirical studies could be encoded as binomial subjective opinions, and how Subjective Logic fusion operators could be used to combine results from multiple experiments, whilst providing an explicit measure of uncertainty for the grouped experiments. Although we did not use fusion operators, their ability to systematically ‘fuse’ together opinions is something that could be useful in the context of reasoning about or manipulating SOSMs.

Software safety assessment has also seen efforts to reason about epistemic uncertainty. *Duan et al.* used Subjective Logic to reason about uncertainty in safety-cases, and emphasised the value of using the beta-distribution (Section 2.3.2) for capturing this uncertainty. *Nair et al.* [47] applied Evidential Reasoning [14] to the same area. Evidential Reasoning is also founded on Dempster-Shafer theory, but does not feature the variety of operators in Subjective Logic.

6.2 Uncertainty in Sequential Models

There is an extensive history of research into the combination of finite automata and probability theory. Although this paper used PFSMs [3] as a baseline, there exist other potential representations of probabilistic sequential systems. Hidden Markov Models (HMMs) and Fuzzy Finite Automata are two notable, widely used examples.

HMMs can be seen as a form of PFSM [48], where symbols label states and every state has a probability of being the initial state. Given the equivalence with PFSMs [48], HMMs also only model ‘first-order’ probabilities and do not explicitly model uncertainty associated with a probability.

Fuzzy Automata map a transitions t to a value representing the degree to which t is in the automaton [49]. Initial work used values in $[0, 1]$ but, more generally, one can use values from a complete lattice [50]. The value for a path is the greatest lower bound of the values of the transitions; for $[0, 1]$, this is the minimum. With multiple paths, the value is the least upper bound of the values for the individual paths [49]; for $[0, 1]$, the maximum, over the paths.

One might use values to represent uncertainty in probabilities and we discuss two options. First, one might use a value in $[0, 1]$. Consider a path t_1t_2 , where t_1 and t_2 are transitions. If t_1 is assigned a and t_2 is assigned b then t_1t_2 has value $\min\{a, b\}$. Now, if a and b are identical and strictly between 0 and 1 then t_1t_2 also has uncertainty a . However, we would expect the path to have greater uncertainty than the transitions. We obtain the same problem if we instead give a transition a value $[a, b]$, $0 \leq a \leq b \leq 1$, and we use the subset relation as the partial order. SOSMs thus appear to be more suitable for the work described in this paper.

6.3 Inference of Probabilistic Sequential Models

There have been several efforts to reverse-engineer probabilistic models from software (and hardware) systems [51].

Although most concerned PFSM inference, some involved HMMs and other probabilistic sequential models.

Efforts to infer PFSMs date back to work by Rivest and Shapiro [52], who inferred robot controllers. In 1998 Cook and Wolf experimented with different inference approaches to infer sequential models, one being a Markov Model¹⁰ [53]. More recent examples include PFSM inference in areas such as Android GUI testing [22], specification mining [25], web apps [21] and car drivers [20].

There have been recent efforts to infer HMMs. Nguyen *et al.*'s DroidAssist tool [54] infers what are in effect HMMs representing sequences of API usages in mobile apps. Emam and Miller's approach [25] produces PFSMs, but is underpinned by HMM inference. HMMs have been extensively used to detect malware from the structure of executable fragments of bytecode [55]. Recently, HMMs featured in eye-tracking studies of software developers [56].

6.4 Testing Uncertain Sequential Systems

The SUT might incorporate stochastic or non-deterministic behaviours. Elbaum and Rosenblum [57] present compelling examples, such as services that depend on GPS location services and so on estimated locations of a device. They suggest HMMs as a useful basis for testing such systems.

There is also epistemic uncertainty; having observed a set of executions (a test set), how certain can we be that the SUT is 'correct'? Weyuker showed that the answer to this question can never (in the general case) be certain [58]. There is always the risk that there exists some input that is not part of the test set that might expose new behaviours.

This has given rise to 'learning-based testing' [59]. Much of the attention has been on the combination of software testing with state machine inference; a good overview is provided by Aichernig *et al.* [60]. Although uncertainty has been exploited for non-sequential systems [61], the models inferred have tended to be FSMs [60].

Where probabilistic models *are* available, authors have explored testing from these. Much of this work concerns probabilistic labelled transition system. In order to decide which test cases to use, one can find tests that kill mutants [5]. Give test case t , one can test the SUT with t multiple times and use statistical techniques to check that the observed frequencies are consistent with the specification [5].

7 CONCLUSIONS AND FUTURE WORK

We have introduced SOSMs, a generalisation of PFSMs, where states are labelled with multinomial subjective opinions. These not only capture the relative likelihoods of transitions, but also associate them with a level of uncertainty. Thanks to Subjective Logic [9], it is possible to compute the combined likelihood and uncertainty associated with sequences of events. This means that predictions can be associated with a level of 'trustworthiness'. We also provided an algorithm that can be used to automatically generate an SOSM from an FSM and a set of traces.

In our evaluation, we used SOSMs to infer predictive classifiers of behaviour, showing that uncertainty captured

within an inferred SOSM can be a strong indicator of how (in-)accurate a prediction will be. We also showed that the subjective opinions associated with classifications can be used to 'correct' the predictions of the underlying FSM, to the point that the predictions in our experiments were improved from a mean BCR score of 0.76 to a score of 0.94.

We believe that there are several exciting opportunities to apply SOSMs within software testing. There is a well established line of work in testing from probabilistic models, and a (currently) separate line of research into the relationship between testing and uncertainty. We believe that the SOSM could form the basis for combining these two strands. Finally, we have only used a relatively small subset of Subjective Logic; there are other elements that could be used to refine and strengthen our approach, such as the use of hyper-opinions [9], which would enable us to model uncertainty at individual states in more accurate terms.

Acknowledgements: We thank the reviewers for their extensive, constructive feedback. Both authors are supported by the EPSRC CITCOM grant (EP/T030526/1). For the purpose of open access, the author has applied a Creative Commons Attribution (CC BY) licence to any Author Accepted Manuscript version arising.

REFERENCES

- [1] D. Lee and M. Yannakakis, "Principles and methods of testing finite state machines—a survey," *Proceedings of the IEEE*, vol. 84, no. 8, pp. 1090–1123, 1996.
- [2] J. Troya, N. Moreno, M. F. Bertoa, and A. Vallecillo, "Uncertainty representation in software models: a survey," *Software and Systems Modeling*, vol. 20, no. 4, pp. 1183–1213, 2021.
- [3] E. Vidal, F. Thollard, C. De La Higuera, F. Casacuberta, and R. C. Carrasco, "Probabilistic finite-state machines—part I," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 7, pp. 1013–1025, 2005.
- [4] M. Kwiatkowska, G. Norman, and D. Parker, "PRISM 4.0: Verification of probabilistic real-time systems," in *International conference on computer aided verification*. Springer, 2011, pp. 585–591.
- [5] R. M. Hierons and M. G. Merayo, "Mutation testing from probabilistic and stochastic finite state machines," *Journal of Systems and Software*, vol. 82, no. 11, pp. 1804–1818, 2009.
- [6] M. Kwiatkowska, G. Norman, and J. Sproston, "Probabilistic model checking of deadline properties in the IEEE 1394 firewire root contention protocol," *Formal Aspects of Computing*, vol. 14, no. 3, pp. 295–318, 2003.
- [7] P. Gärdenfors and N.-E. Sahlin, "Unreliable probabilities, risk taking, and decision making," *Synthese*, vol. 53, pp. 361–386, 1982.
- [8] A. Der Kiureghian and O. Ditlevsen, "Aleatory or epistemic? Does it matter?" *Structural Safety*, vol. 31, no. 2, pp. 105–112, 2009.
- [9] A. Josang, *Subjective logic*. Springer, 2016.
- [10] W. Choi, G. Necula, and K. Sen, "Guided gui testing of android apps with minimal restart and approximate learning," *Acm Sigplan Notices*, vol. 48, no. 10, pp. 623–640, 2013.
- [11] N. Walkinshaw, "Improving automated gui testing by learning to avoid infeasible tests," in *2020 IEEE International Conference On Artificial Intelligence Testing (AITest)*. IEEE, 2020, pp. 107–114.
- [12] G. Boole, *An investigation of the laws of thought: on which are founded the mathematical theories of logic and probabilities*. Walton and Maberly, 1854, vol. 2.
- [13] G. Shafer, *A mathematical theory of evidence*. Princeton university press, 1976, vol. 42.
- [14] J.-B. Yang and M. G. Singh, "An evidential reasoning approach for multiple-attribute decision making with uncertainty," *IEEE Transactions on systems, Man, and Cybernetics*, vol. 24, no. 1, pp. 1–18, 1994.
- [15] P. Walley, "Inferences from multinomial data: learning about a bag of marbles," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, no. 1, pp. 3–34, 1996.
- [16] L. A. Zadeh, "Fuzzy sets," *Information and control*, vol. 8, no. 3, pp. 338–353, 1965.

10. The non-hidden version of the Markov Model is equivalent to a PFSM; probabilities are expressed on transitions as opposed to states.

- [17] A. Jøsang, "A logic for uncertain probabilities," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 9, no. 03, pp. 279–311, 2001.
- [18] L. Duan, S. Rayadurgam, M. Heimdahl, O. Sokolsky, and I. Lee, "Representation of confidence in assurance cases using the beta distribution," in *2016 IEEE 17th International Symposium on High Assurance Systems Engineering (HASE)*. IEEE, 2016, pp. 86–93.
- [19] R. C. Carrasco and J. Oncina, "Learning stochastic regular grammars by means of a state merging method," in *International Colloquium on Grammatical Inference*. Springer, 1994, pp. 139–152.
- [20] S. Verwer, M. De Weerd, and C. Witteveen, "Learning driving behavior by timed syntactic pattern recognition," in *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.
- [21] J. Borges and M. Levene, "Data mining of user navigation patterns," in *International workshop on web usage analysis and user profiling*. Springer, 1999, pp. 92–112.
- [22] T. Su, G. Meng, Y. Chen, K. Wu, W. Yang, Y. Yao, G. Pu, Y. Liu, and Z. Su, "Guided, stochastic model-based GUI testing of android apps," in *Proceedings of the 2017 11th Joint Meeting on Foundations of Software Engineering*, 2017, pp. 245–256.
- [23] C. De la Higuera, *Grammatical inference: learning automata and grammars*. Cambridge University Press, 2010.
- [24] C. A. Furia, R. Feldt, and R. Torkar, "Bayesian data analysis in empirical software engineering research," *IEEE Transactions on Software Engineering*, 2019.
- [25] S. S. Emam and J. Miller, "Inferring extended probabilistic finite-state automaton models from software executions," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 27, no. 1, pp. 1–39, 2018.
- [26] N. Walkinshaw, B. Lambeau, C. Damas, K. Bogdanov, and P. Dupont, "Stamina: a competition to encourage the development and assessment of software model inference techniques," *Empirical software engineering*, vol. 18, no. 4, pp. 791–824, 2013.
- [27] D. Angluin, "On the complexity of minimum inference of regular sets," *Information and control*, vol. 39, no. 3, pp. 337–350, 1978.
- [28] E. M. Gold, "Complexity of automaton identification from given data," *Information and control*, vol. 37, no. 3, pp. 302–320, 1978.
- [29] L. G. Valiant, "A theory of the learnable," *Communications of the ACM*, vol. 27, no. 11, pp. 1134–1142, 1984.
- [30] E. Vidal, F. Thollard, C. De La Higuera, F. Casacuberta, and R. C. Carrasco, "Probabilistic finite-state machines-part ii," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 27, no. 7, pp. 1026–1039, 2005.
- [31] K. J. Lang, B. A. Pearlmutter, and R. A. Price, "Results of the abbadingo one DFA learning competition and a new evidence-driven state merging algorithm," in *International Colloquium on Grammatical Inference*. Springer, 1998, pp. 1–12.
- [32] D. Angluin, "Learning regular sets from queries and counterexamples," *Information and Computation*, vol. 75, pp. 87–106, 1987.
- [33] T. K. Ho, "The random subspace method for constructing decision forests," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 20, no. 8, pp. 832–844, 1998.
- [34] S. He, J. Zhu, P. He, and M. R. Lyu, "Loghub: A large collection of system log datasets towards automated log analytics," 2020. [Online]. Available: <https://arxiv.org/abs/2008.06448>
- [35] F. Brglez, "ACM/SIGMOD benchmark dataset," 1996. [Online]. Available: <https://people.engr.ncsu.edu/brglez/CBL/benchmarks/Benchmarks-upto-1996.html>
- [36] D. Neider, R. Smetsers, F. Vaandrager, and H. Kuppens, "Benchmarks for automata learning and conformance testing," in *Models, Mindsets, Meta: The What, the How, and the Why Not?* Springer, 2019, pp. 390–416.
- [37] M. Tappler, B. K. Aichernig, and R. Bloem, "Model-based testing IoT communication via active automata learning," in *2017 IEEE International conference on software testing, verification and validation (ICST)*. IEEE, 2017, pp. 276–287.
- [38] B. Steffen, F. Howar, and M. Merten, "Introduction to active automata learning from a practical perspective," in *International School on Formal Methods for the Design of Computer, Communication and Software Systems*. Springer, 2011, pp. 256–296.
- [39] J. de Ruiter, "A tale of the openssl state machine: A large-scale black-box analysis," in *Nordic Conference on Secure IT Systems*. Springer, 2016, pp. 169–184.
- [40] P. Fiterău-Broștean, T. Lenaerts, E. Poll, J. de Ruiter, F. Vaandrager, and P. Verleg, "Model learning and model checking of SSH implementations," in *Proceedings of the 24th ACM SIGSOFT International SPIN Symposium on Model Checking of Software*, 2017, pp. 142–151.
- [41] A. Rasool, "Quic patrol: Protocol analysis at the transport layer," 2018, thesis.
- [42] P. Fiterău-Broștean, R. Janssen, and F. Vaandrager, "Combining model learning and model checking to analyze TCP implementations," in *International Conference on Computer Aided Verification*. Springer, 2016, pp. 454–471.
- [43] N. Walkinshaw and K. Bogdanov, "Automated comparison of state-based software models in terms of their language and structure," *ACM Transactions on Software Engineering and Methodology (TOSEM)*, vol. 22, no. 2, pp. 1–37, 2013.
- [44] A. Vargha and H. D. Delaney, "A critique and improvement of the CL common language effect size statistics of McGraw and Wong," *Journal of Educational and Behavioral Statistics*, vol. 25, no. 2, pp. 101–132, 2000.
- [45] J. Dorn, S. Apel, and N. Siegmund, "Mastering uncertainty in performance estimations of configurable software systems," in *2020 35th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2020, pp. 684–696.
- [46] N. Walkinshaw and M. Shepperd, "Reasoning about uncertainty in empirical results," in *Proceedings of the Evaluation and Assessment in Software Engineering*, 2020, pp. 140–149.
- [47] S. Nair, N. Walkinshaw, T. Kelly, and J. L. de la Vara, "An evidential reasoning approach for assessing confidence in safety evidence," in *2015 IEEE 26th International Symposium on Software Reliability Engineering (ISSRE)*. IEEE, 2015, pp. 541–552.
- [48] P. Dupont, F. Denis, and Y. Esposito, "Links between probabilistic automata and hidden markov models: probability distributions, learning models and induction algorithms," *Pattern recognition*, vol. 38, no. 9, pp. 1349–1371, 2005.
- [49] W. G. Wee and K.-S. Fu, "A formulation of fuzzy automata and its application as a model of learning systems," *IEEE Transactions on Systems Science and Cybernetics*, vol. 5, no. 3, pp. 215–223, 1969.
- [50] Z. Li, P. Li, and Y. Li, "The relationships among several types of fuzzy automata," *Inf. Sci.*, vol. 176, no. 15, pp. 2208–2226, 2006. [Online]. Available: <https://doi.org/10.1016/j.ins.2005.05.001>
- [51] S. Verwer, R. Eyraud, and C. de la Higuera, "PAutomA: a probabilistic automata and hidden markov models learning competition," *Mach. Learn.*, vol. 96, no. 1–2, pp. 129–154, 2014. [Online]. Available: <https://doi.org/10.1007/s10994-013-5409-9>
- [52] R. L. Rivest and R. E. Schapire, "Inference of finite automata using homing sequences," *Information and Computation*, vol. 103, no. 2, pp. 299–347, 1993.
- [53] J. E. Cook and A. L. Wolf, "Discovering models of software processes from event-based data," *ACM Transactions on Software Engineering and Methodology*, vol. 7, no. 3, pp. 215–249, 1998.
- [54] T. T. Nguyen, H. V. Pham, P. M. Vu, and T. T. Nguyen, "Recommending API usages for mobile apps with hidden markov model," in *2015 30th IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 2015, pp. 795–800.
- [55] S. Attaluri, S. McGhee, and M. Stamp, "Profile hidden markov models and metamorphic virus detection," *Journal in computer virology*, vol. 5, no. 2, pp. 151–169, 2009.
- [56] Z. Sharafi, B. Sharif, Y.-G. Guéhéneuc, A. Begel, R. Bednarik, and M. Crosby, "A practical guide on conducting eye tracking studies in software engineering," *Empirical Software Engineering*, vol. 25, no. 5, pp. 3128–3174, 2020.
- [57] S. Elbaum and D. S. Rosenblum, "Known unknowns: Testing in the presence of uncertainty," in *Proceedings of the 22Nd ACM SIGSOFT International Symposium on Foundations of Software Engineering*, 2014, pp. 833–836.
- [58] E. J. Weyuker, "On testing non-testable programs," *The Computer Journal*, vol. 25, no. 4, pp. 465–470, 1982.
- [59] —, "Assessing test data adequacy through program inference," *ACM Transactions on Programming Languages and Systems (TOPLAS)*, vol. 5, no. 4, pp. 641–655, 1983.
- [60] B. K. Aichernig, W. Mostowski, M. R. Mousavi, M. Tappler, and M. Taromirad, "Model learning and model-based testing," in *Machine Learning for Dynamic Software Analysis: Potentials and Limits*. Springer, 2018, pp. 74–100.
- [61] N. Walkinshaw and G. Fraser, "Uncertainty-driven black-box test data generation," in *2017 IEEE International Conference on Software Testing, Verification and Validation (ICST)*. IEEE, 2017, pp. 253–263.
- [62] W. H. Kruskal and W. A. Wallis, "Use of ranks in one-criterion variance analysis," *Journal of the American statistical Association*, vol. 47, no. 260, pp. 583–621, 1952.

APPENDIX A

MODELS THAT WERE EXCLUDED FROM OUR STUDIES

For completeness-sake, we include a list of models for which our trace generation algorithm was unable to produce a diverse set of sequences that matched our requirements in terms of size and diversity. This would either be because a model was too small to enable a sufficiently diverse set of traces (of the length-limits we stipulated). Alternatively, it was because their transition structure made it difficult for random walks to converge on sufficiently diverse accepting or rejecting walks. All of the models in question are in the ACM/SIGDA benchmark set [35]: ex2, ex3, ex5, ex7, kirkman, mark1, mc, modulo12, opus, s208, s420, and scf.

APPENDIX B

PRELIMINARY STUDY ON EFFECT OF α

We computed \hat{A}_{12} values for α : 250, 500, 1000, and 2000. For each, we created two box-plots (the \hat{A}_{12} values for belief, the other for uncertainty). The boxplots are shown in Figure 8, and capture the distribution of \hat{A}_{12} values for accepting and rejecting sequences, computed for all models and random seeds. If there is a statistically significant difference between distributions for different α , we can conclude that the choice of α is significant. We tested for significance using the non-parametric Kruskal-Wallis test [62], which estimates the likelihood that samples from different groups are from the same distribution. We adopt the convention of setting significance to 0.05.

The p -values are shown in Table 7. The table shows that there are no statistically significant differences between \hat{A}_{12} values for any values of α for rejecting sequences. There are also no differences for accepting values for α values below 2000. However results for $\alpha = 2000$ are significantly different from lower values for belief and uncertainty.

The choice of $\alpha = 2000$ leads to \hat{A}_{12} that are particularly distinctive (high for accepting sequences and low for rejecting sequences), for both Belief and Uncertainty. The differences to other choices of α are statistically significant (as shown in Table 7).

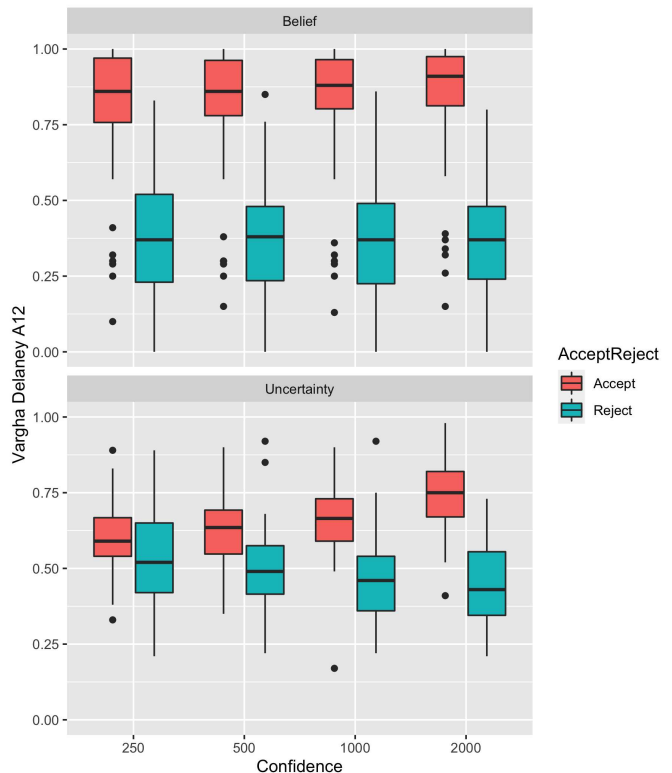


Fig. 8. Distributions of \hat{A}_{12} scores for four values of α . 0.5 indicates no differentiation between correctly and incorrectly classified sequences.

TABLE 7

p -values for Pairwise Wilcoxon Tests, comparing distributions of belief and uncertainty \hat{A}_{12} values for accepted and rejected sequences.

Accepting							
Belief \hat{A}_{12}				Uncertainty \hat{A}_{12}			
	250	500	1000		250	500	1000
500	1.00	-	-	500	1.00	-	-
1000	1.00	1.00	-	1000	0.03	0.53	-
2000	0.81	1.00	1.00	2000	< 0.01	< 0.01	< 0.01

Rejecting							
Belief \hat{A}_{12}				Uncertainty \hat{A}_{12}			
	250	500	1000		250	500	1000
500	1.00	-	-	500	1.00	-	-
1000	1.00	1.00	-	1000	0.18	1.00	-
2000	1.00	1.00	1.00	2000	0.03	0.22	1.00