A Denotational Model for Component-Based Risk Analysis

Gyrd Brændeland[1,2,⋆], Atle Refsdal[2], and Ketil Stølen[1,2]

[1] Department of Informatics, University of Oslo, Norway
[2] SINTEF, Norway

Abstract. Risk analysis is an important tool for developers to establish the appropriate protection level of a system. Unfortunately, the shifting environment of components and component-based systems is not adequately addressed by traditional risk analysis methods. This paper addresses this problem from a theoretical perspective by proposing a denotational model for component-based risk analysis. In order to model the probabilistic aspect of risk, we represent the behaviour of a component by a probability distribution over communication histories. The overall goal is to provide a theoretical foundation facilitating an improved understanding of risk in relation to components and component-based system development.

1 Introduction

The flexibility offered by component-based development facilitates rapid development and deployment, but causes challenges for security and safety as upgraded sub-components may interact with a system in unforeseen ways. The difficulties faced by Toyota in explaining what caused the problem with the sticky accelerators [1] illustrate this problem. Due to their lack of modularity conventional risk analysis methods are poorly suited to address these challenges. A modular understanding of risks is a prerequisite for robust component-based development and for maintaining the trustworthiness of component-based systems.

There are many forms and variations of risk analysis, depending on the application domain, such as finance, reliability and safety, or security. Within reliability/safety and security, which are the most relevant for component-based development, risk analysis is concerned with protecting assets. This is the type of risk analysis we focus upon in this paper, referred to as defensive risk analysis. The purpose of defensive risk analysis is to gather sufficient knowledge about vulnerabilities, threats, consequences and probabilities, in order to establish the appropriate protection level for assets. It is important that the level of protection matches the value of the assets to be protected. A certain level of risk may be acceptable if the risk is considered to be too costly or technically impossible to rule out entirely. Hence, a risk is part of the behaviour of a system that is implicitly allowed but not necessarily intended. Based on this observation we

⋆ Contact author: email: gyb@sintef.uio.no

© Springer-Verlag Berlin Heidelberg 2012

have defined a component model that integrates the explicit representation of risks as part of the component behaviour and provides rules for composing component risks. We also explain how the notion of hiding can be understood in this component model. We define a hiding operator that allows partial hiding of internal interactions, to ensure that interactions affecting the component risk level are not hidden. We are not aware of other approaches where the concept of risk is integrated in a formal component semantics.

An advantage of representing risks as part of the component behaviour, is that the risk level of a composite component, as well as its behaviour, is obtained by composing the representations of its sub-components. That is, the composition of risks corresponds to ordinary component composition. The component model provides a foundation for component-based risk analysis, by conveying how risks manifests themselves in an underlying component implementation. By component-based risk analysis we mean that risks are identified, analysed and documented at the component level, and that risk analysis results are composable.

1.1 Outline of paper

The objective of Section 2 is to give an informal understanding of component-based risk analysis. Risk is the probability that an event affects an asset with a given consequence. In order to model component risks, we explain the concept of asset, asset value and consequence in a component setting. In order to represent the *behavioural* aspects of risk, such as the probability of unwanted incidents, we make use of an asynchronous communication paradigm. The selection of this paradigm is motivated as part of the informal explanation of component-based risk analysis. We also explain the notions of observable and unobservable behaviour in a component model with assets. The informal understanding introduced in Section 2 is thereafter formalised in a semantic model that defines:

– The denotational representation of interfaces as probabilistic processes (Section 3).
– The denotational representation of interface risks including the means to represent risk probabilities (Section 4). Interface risks are incorporated as a part of the interface behaviour.
– The denotational representation of a component as a collection of interfaces or sub-components, some of which may interact with each other (Section 5). We obtain the behaviour of a component from the probabilistic processes of its constituent interfaces or sub-components in a basic mathematical way.
– The denotational representation of component risks (Section 6).
– The denotational representation of hiding (Section 7).

We place our work in relation to ongoing research within related areas in Section 8. Finally we summarise our findings and discuss possibilities for future work in Section 9. Formal proofs of all the results presented in this paper is available in a technical report [2][1].

[1] `http://heim.ifi.uio.no/ ketils/kst/Reports/2011-02.UIO-IFI-363.A-Denotational-Model-For-Component-Based-Risk-Analysis.pdf`

2 An Informal Explanation of Component-Based Risk Analysis

In order to provide a foundation for component-based risk analysis, we first explain informally how concepts from risk analysis can be understood at the component level. Concepts to consider in defensive risk analysis [31,14] include: A *stakeholder* refers to a person or organisation who is affected by a decision or activity. An *asset* is something to which a stakeholder directly assigns value and, hence, for which the stakeholder requires protection. An *incident* is an event that reduces the value of one or more assets. A *consequence* is the reduction in value caused by an incident to an asset. A *vulnerability* is a weakness which can be exploited by one or more threats to harm an asset. A *threat* is a potential cause of an incident. *Probability* is a measure of the chance of occurrence of an event, expressed as a number between 0 and 1. A *risk* is the combination of the probability of an incident and its consequence with regard to a given asset. There may be a range of possible outcomes associated with an incident. This implies that an incident may have consequences for several assets. Hence, an incident may be part of several risks.

2.1 Component-Based Risk Analysis

We explain the concepts of component-based risk analysis and how they are related to each other through a conceptual model, captured by a UML class diagram [22] in Figure 1. The associations between the elements have cardinalities specifying the number of instances of one element that can be related to one instance of the other. The hollow diamond symbolises aggregation and the filled composition. Elements connected with an aggregation can also be part of other aggregations, while composite elements only exist within the specified composition.

An *interface* is a contract describing both the provided operations and the services required to provide the specified operations. To ensure modularity of our component model we represent a stakeholder by the component interface, and identify assets on behalf of component interfaces. Each interface has a set of assets. A vulnerability may be understood as a property (or lack thereof) of

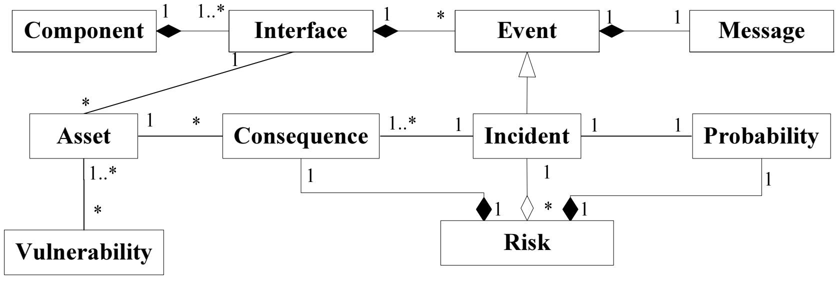

Fig. 1. Conceptual model of component-based risk analysis

an interface that makes it prone to a certain attack. It may therefore be argued that the vulnerability concept should be associated to the interface concept. However, from a risk perspective a vulnerability is relevant to the extent that it can be exploited to harm a specific asset, and we have therefore chosen to associate it with the asset concept. The concept of a threat is not part of the conceptual model, because a threat is something that belongs to the environment of a component. We cannot expect to have knowledge about the environment of the component as that may change depending on the where it is deployed.

A *component* is a collection of interfaces some of which may interact with each other. Interfaces interact by the transmission and consumption of messages. We refer to the transmission and consumption of messages as *events*. An event that harms an asset is an incident with regard to that asset.

2.2 Behaviour and Probability

A probabilistic understanding of component behaviour is required in order to measure risk. We adopt an asynchronous communication model. This does not prevent us from representing systems with synchronous communication. It is well known that synchronous communication can be simulated in an asynchronous communication model and the other way around [13].

An interface interacts with an environment whose behaviour it cannot control. From the point of view of the interface the choices made by the environment are non-deterministic. In order to resolve the external non-determinism caused by the environment we use queues that serve as schedulers. Incoming messages to an interface are stored in a queue and are consumed by the interface in the order they are received. The idea is that, for a given sequence of incoming messages to an interface, we know the probability with which the interface produces a certain behaviour. For simplicity we assume that an interface does not send messages to itself.

A component is a collection of interfaces some of which may interact. For a component consisting of two or more interfaces, a queue history not only resolves the external non-determinism, but also all internal non-determinism with regard to the interactions of its sub-components. The behaviour of a component is the set of probability distributions given all possible queue histories of the component.

Figure 2 shows two different ways in which two interfaces n_1 and n_2 with queues q_1 and q_2, and sets of assets a_1 and a_2, can be combined into a component. We may think of the arrows as directed channels.

– In Figure 2 (1) there is no direct communication between the interfaces of the component, that is, the queue of each interface only contains messages from external interfaces.
– In Figure 2 (2) the interface n_1 transmits to n_2 which again transmits to the environment. Moreover, only n_1 consumes messages from the environment.

Initially, the queue of each interface is empty; its set of assets is fixed throughout an execution. When initiated, an interface chooses probabilistically between a number of different actions. An action consists of transmitting an arbitrary

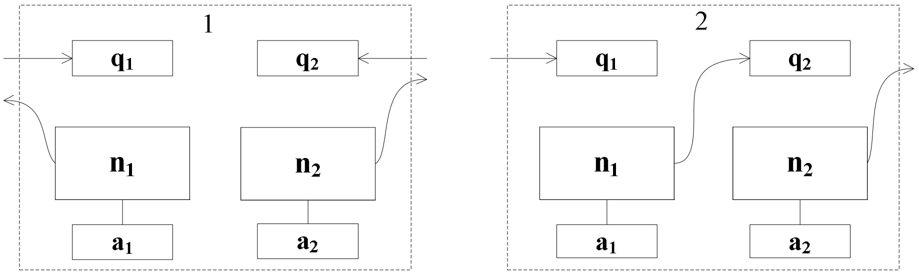

Fig. 2. Two interface compositions

number of messages in some order. The number of transmission messages may be finite, including zero which corresponds to the behaviour of skip, or infinite. The storing of a transmitted message in a queue is instantaneous: a transmitted message is placed in the queue of the recipient, without time delay. There will always be some delay between the transmission of a message and the consumption of that message. After transmitting messages the interface may choose to quit or to check its queue for messages. Messages are consumed in the order they arrive. If the queue is empty, an attempt to consume blocks the interface from any further action until a new message arrives. The consumption of a message gives rise to a new probabilistic choice. Thereafter, the interface may choose to quit without checking the queue again, and so on.

A probabilistic choice over actions never involves more than one interface. This can always be ensured by decomposing probabilistic choices until they have the granularity required. Suppose we have three interfaces; die, player1 and player2 involved in a game of Monopoly. The state of the game is decided by the position of the players' pieces on the board. The transition from one state to another is decided by a probabilistic choice "Throw die and move piece", involving both the die and one of the players. We may however, split this choice into two separate choices: "Throw die" and "Move piece". By applying this simple strategy for all probabilistic choices we ensure that a probabilistic choice is a local event of an interface.

The probability distribution over a set of actions, resulting from a probabilistic choice, may change over time during an execution. Hence, our probabilistic model is more general than for example a Markov process [32,21], where the probability of a future state given the present is conditionally independent of the past. This level of generality is needed to be able to capture all types of probabilistic behaviour relevant in a risk analysis setting, including human behaviour.

The behaviour of a component is completely determined by the behaviour of its constituent interfaces. We obtain the behaviour of a component by starting all the interfaces simultaneously, in their initial state.

2.3 Observable Component Behaviour

In most component-based approaches there is a clear separation between external and purely internal interaction. External interaction is the interaction

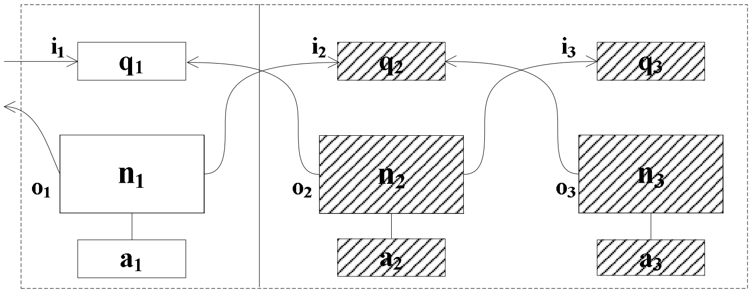

Fig. 3. Hiding of unobservable behaviour

between the component and its environment; while purely internal interaction is the interaction within the components, in our case, the interaction between the interfaces of which the component consists. Contrary to the external, purely internal interaction is hidden when the component is viewed as a black-box.

When we bring in the notion of risk, this distinction between what should be externally and only internally visible is no longer clear cut. After all, if we blindly hide all internal interaction we are in danger of hiding (without treating) risks of relevance for assets belonging to externally observable interfaces. Hence, purely internal interaction should be externally visible if it may affect assets belonging to externally visible interfaces. Consider for example the component pictured in Figure 3. In a conventional component-oriented approach, the channels i_2, i_3, o_2 and o_3 would not be externally observable from a black-box point of view. From a risk analysis perspective it seems more natural to restrict the black-box perspective to the right hand side of the vertical line. The assets belonging to the interface n_1 are externally observable since the environment interacts with n_1. The assets belonging to the interfaces n_2 and n_3 are on the other hand hidden since n_2 and n_3 are purely internal interfaces. Hence, the channels i_3 and o_3 are also hidden since they can only impact the assets belonging to n_1 indirectly via i_2 and o_2. The channels i_2 and o_2 are however only partly hidden since the transmission events of i_2 and the consumption events of o_2 may include incidents having an impact on the assets belonging to n_1.

3 Denotational Representation of Interface Behaviour

In this section we explain the formal representation of interface behaviour in our denotational semantics. We represent interface behaviour by sequences of events that fulfil certain well-formedness constraints. Sequences fulfilling these constraints are called traces. We represent probabilistic interface behaviour as probability distributions over sets of traces.

3.1 Sets

We use standard set notation, such as *union* $A \cup B$, *intersection* $A \cap B$, set difference $A \setminus B$, *cardinality* $\#A$ and *element of* $e \in A$ in the definitions of our basic concepts and operators. We write $\{e_1, e_2, e_3, \ldots, e_n\}$ to denote the set consisting of n elements $e_1, e_2, e_3, \ldots, e_n$. Sometimes we also use $[i..n]$ to denote a totally ordered set of numbers between i and n. We introduce the special symbol $\mathbb{N}$ to denote the set of natural numbers and $\mathbb{N}_+$ to denote the set of strictly positive natural numbers.

3.2 Events

There are two kinds of events: transmission events tagged by ! and consumption events tagged by ?. $\mathcal{K}$ denotes the set of kinds $\{!, ?\}$. An event is a pair of a kind and a message. A message is a quadruple $\langle s, tr, co, q\rangle$ consisting of a signal s, a transmitter tr, a consumer co and a time-stamp q, which is a rational number. The consumer in the message of a transmission event coincides with the addressee, that is, the party intended to eventually consume the message. The *active* party in an event is the one performing the action denoted by its kind. That is, the transmitter of the message is the active party of a transmission event and the consumer of the message is the active party of a consumption event.

We let $\mathcal{S}$ denote the set of all signals, $\mathcal{P}$ denote the set of all parties (consumers and transmitters), $\mathcal{Q}$ denote the set of all time-stamps, $\mathcal{M}$ denote the set of all messages and $\mathcal{E}$ denote the set of all events. Formally we have that:

$$\mathcal{E} \overset{\mathsf{def}}{=} \mathcal{K} \times \mathcal{M}$$
$$\mathcal{M} \overset{\mathsf{def}}{=} \mathcal{S} \times \mathcal{P} \times \mathcal{P} \times \mathcal{Q}$$

We define the functions

$$k.\_ \in \mathcal{E} \to \mathcal{K} \quad tr.\_, co.\_ \in \mathcal{E} \to \mathcal{P} \quad q.\_ \in \mathcal{E} \to \mathcal{Q}$$

to yield the kind, transmitter, consumer and time-stamp of an event. For any party $p \in \mathcal{P}$, we use $\mathcal{E}_p$ to denote the set of all events in which p is the active part. Formally

(1) $$\mathcal{E}_p \overset{\mathsf{def}}{=} \{e \in \mathcal{E} \mid (k.e =! \wedge tr.e = p) \vee (k.e =? \wedge co.e = p)\}$$

For a given party p, we assume that the number of signals assigned to p is a most countable. That is, the number of signals occurring in messages consumed by or transmitted to p is at most countable.

We use $\mathcal{E}_p^{\downarrow}$ to denote the set of transmission events with p as consumer. Formally

$$\mathcal{E}_p^{\downarrow} \overset{\mathsf{def}}{=} \{e \in \mathcal{E} \mid k.e =! \wedge co.e = p\}$$

3.3 Sequences

For any set of elements A, we let A^ω, A^∞, A^* and A^n denote the set of all sequences, the set of all infinite sequences, the set of all finite sequences, and the set of all sequences of length n over A. We use $\langle\rangle$ to denote the empty sequence of length zero and $\langle 1,2,3,4\rangle$ to denote the sequence of the numbers from 1 to 4. A sequence over a set of elements A can be viewed as a function mapping positive natural numbers to elements in the set A. We define the functions

$$(2) \qquad \#\_ \in A^{\omega} \rightarrow \mathbb{N} \cup \{\infty\} \quad \_\sqsubseteq\_ \in A^{\omega} \times A^{\omega} \rightarrow \mathbb{B}\mathrm{ool}$$

to yield the length, the nth element of a sequence and the prefix ordering on sequences[2]. Hence, $\#s$ yields the number of elements in s, $s[n]$ yields s's nth element if $n \leq \#s$, and $s_1 \sqsubseteq s_2$ evaluates to true if s_1 is an initial segment of s_2 or if $s_1 = s_2$.

For any $0 \leq i \leq \#s$ we define $s|_i$ to denote the prefix of s of length i. Formally:

$$(3) \qquad \_|\_ \in A^{\omega} \times \mathbb{N} \rightarrow A^{\omega}$$

$$s|_i \overset{\mathsf{def}}{=} \begin{cases} s' \text{ if } 0 \leq i \leq \#s, \text{ where } \#s' = i \wedge s' \sqsubseteq s \\ s \text{ if } i > \#s \end{cases}$$

Due to the functional interpretation of sequences, we may talk about the *range* of a sequence:

$$(4) \qquad \mathsf{rng}.\_ \in A^{\omega} \rightarrow \mathbb{P}(A)$$

For example if $s \in A^{\infty}$, we have that:

$$\mathsf{rng}.s = \{s[n] \mid n \in \mathbb{N}_+\}$$

We define an operator for obtaining the sets of events of a set of sequences, in terms of their ranges:

$$(5) \qquad ev.\_ \in \mathbb{P}(A^{\omega}) \rightarrow \mathbb{P}(A)$$

$$ev.S \overset{\mathsf{def}}{=} \bigcup_{s \in S} \mathsf{rng}.s$$

We also define an operator for concatenating two sequences:

$$(6) \qquad \_\frown\_ \in A^{\omega} \times A^{\omega} \rightarrow A^{\omega}$$

$$s_1 \frown s_2[n] \overset{\mathsf{def}}{=} \begin{cases} s_1[n] \text{ if } 1 \leq n \leq \#s_1 \\ s_2[n - \#s_1] \text{ if } \#s_1 < n \leq \#s_1 + \#s_2 \end{cases}$$

Concatenating two sequences implies gluing them together. Hence $s_1 \frown s_2$ denotes a sequence of length $\#s_1 + \#s_2$ that equals s_1 if s_1 is infinite and is prefixed by s_1 and suffixed by s_2, otherwise.

[2] The operator $\times$ binds stronger than $\rightarrow$ and we therefore omit the parentheses around the argument types in the signature definitions.

The filtering function $\circledS$ is used to filter away elements. By $B \circledS s$ we denote the sequence obtained from the sequence s by removing all elements in s that are not in the set of elements B. For example, we have that

$$\{1,3\} \circledS \langle 1,1,2,1,3,2\rangle = \langle 1,1,1,3\rangle$$

We define the filtering operator formally as follows:

$$\text{(7)} \qquad \_\circledS\_ \in \mathbb{P}(A) \times A^{\omega} \to A^{\omega}$$

$$B \circledS \langle\rangle \overset{\mathsf{def}}{=} \langle\rangle$$

$$B \circledS (\langle e\rangle \frown s) \overset{\mathsf{def}}{=} \begin{cases} \langle e\rangle \frown B \circledS s & \text{if } e \in B \\ B \circledS s & \text{if } e \notin B \end{cases}$$

For an infinite sequence s we need the additional constraint:

$$(B \cap \mathsf{rng}.s) = \emptyset \Rightarrow B \circledS s = \langle\rangle$$

We overload $\circledS$ to filtering elements from sets of sequences as follows:

$$\_\circledS\_ \in \mathbb{P}(A) \times \mathbb{P}(A^{\omega}) \to \mathbb{P}(A^{\omega})$$

$$B \circledS S \overset{\mathsf{def}}{=} \{B \circledS s \mid s \in S\}$$

We also need a projection operator $\Pi_i.s$ that returns the ith element of an n-tuple s understood as a sequence of length n. We define the projection operator formally as:

$$\Pi_{\_}.\_ \in \{1 \ldots n\} \times A^n \to A$$

$$\_[\_] \in A^{\omega} \times \mathbb{N}_+ \to A$$

The projection operator is overloaded to sets of index values as follows.

$$\Pi_{\_}.\_ \in \mathbb{P}(\{1 \ldots n\}) \setminus \emptyset \times A^n \to \bigcup_{1 \le k \le n} A^k$$

$$\Pi_I.s \overset{\mathsf{def}}{=} s'$$

$$\text{where } \forall j \in I : \Pi_j.s = \Pi_{\#\{i \in I \mid i \le j\}}.s' \wedge \#s' = \#I$$

For example we have that:

$$\Pi_{\{1,2\}}.\langle p,q,r\rangle = \langle p,q\rangle$$

For a sequence of tuples s, $\Pi_I.s$ denotes the sequence of k-tuples obtained from s, by projecting each element in s with respect to the index values in I. For example we have that

$$\Pi_{\{1,2\}}.\langle\langle a,r,p\rangle,\langle b,r,p\rangle\rangle = \langle \Pi_{\{1,2\}}.\langle a,r,p\rangle\rangle \frown \langle \Pi_{\{1,2\}}.\langle b,r,p\rangle\rangle = \langle\langle a,r\rangle,\langle b,r\rangle\rangle$$

We define the projection operator on a sequence of n-tuples formally as follows:

$$\Pi_{\_}.\_ \in \mathbb{P}(\{1 \ldots n\}) \setminus \emptyset \times (A^n)^{\omega} \rightarrow \bigcup_{1 \leq k \leq n} (A^k)^{\omega}$$

$$\Pi_I.s \overset{\mathsf{def}}{=} s'$$

where

$$\forall j \in \{1 \ldots \#s\} : \Pi_I.s[j] = s'[j] \wedge \#s = \#s'$$

If we want to restrict the view of a sequence of events to only the signals of the events, we may apply the projection operator twice, as follows:

$$\Pi_1.(\Pi_2.\langle !\langle a, r, p, 3\rangle, !\langle b, r, p, 5\rangle\rangle) = \langle\langle a\rangle, \langle b\rangle\rangle$$

Restricting a sequence of events, that is, pairs of kinds and messages, to the second elements of the events yields a sequence of messages. Applying the projection operator a second time with the subscript 1 yields a sequence of signals.

3.4 Traces

A trace t is a sequence of events that fulfils certain well-formedness constraints reflecting the behaviour of the informal model presented in Section 2. We use traces to represent communication histories of components and their interfaces. Hence, the transmitters and consumers in a trace are interfaces. We first formulate two constraints on the timing of events in a trace. The first makes sure that events are ordered by time while the second is needed to avoid Zeno-behaviour. Formally:

$$\forall i, j \in [1..\#t] : i < j \Rightarrow q.t[i] < q.t[j] \tag{8}$$

$$\#t = \infty \Rightarrow \forall k \in \mathcal{Q} : \exists i \in \mathbb{N} : q.t[i] > k \tag{9}$$

For simplicity, we require that two events in a trace never have the same time-stamp. We impose this requirement by assigning each interface a set of time-stamps disjoint from the set of time-stamps assigned to every other interface. Every event of an interface is assigned a unique time-stamp from the set of time-stamps assigned to the interface in question.

The first constraint makes sure that events are totally ordered according to when they take place. The second constraint states that time in an infinite trace always eventually progress beyond any fixed point in time. This implies that time never halts and Zeno-behaviour is therefore not possible. To lift the assumption that two events never happen at the same time, we could replace the current notion of a trace as a sequence of events, to a notion of a trace as a sequence of sets of events where the messages in each set have the same time-stamp.

We also impose a constraint on the ordering of transmission and consumption events in a trace t. According to the operational model a message can be transmitted without being consumed, but it cannot be consumed without having been transmitted. Furthermore, the consumption of messages transmitted to the same party must happen in the same order as transmission. However, since a trace may include consumption events with external transmitters, we can constrain only the consumption of a message from a party which is itself active in the trace. That is, the ordering requirements on t only apply to the communication between the internal parties. This motivates the following formalisation of the ordering constraint:

```
(10) let $N = \{n \in \mathcal{P} \mid \mathsf{rng}.t \cap \mathcal{E}_n \neq \emptyset\}$
     in $\forall n, m \in N :$
         let $i = (\{?\} \times (\mathcal{S} \times n \times m \times \mathcal{Q})) Ⓢ t$
             $o = (\{!\} \times (\mathcal{S} \times n \times m \times \mathcal{Q})) Ⓢ t$
         in $\Pi_{\{1,2,3\}}.(\Pi_{\{2\}}.i) \sqsubseteq \Pi_{\{1,2,3\}}.(\Pi_{\{2\}}.o) \wedge \forall j \in \{1..\#i\} : q.o[j] < q.i[j]$
```

The first conjunct of constraint (10) requires that the sequence of consumed messages sent from an internal party n to another internal party m, is a prefix of the sequence of transmitted messages from n to m, when disregarding time. We abstract away the timing of events in a trace by applying the projection operator twice. Thus, we ensure that messages communicated between internal parties are consumed in the order they are transmitted. The second conjunct of constraint (10) ensures that for any single message, transmission happens before consumption when both the transmitter and consumer are internal. We let $\mathcal{H}$ denote the set of all traces t that are well-formed with regard to constraints (8), (9) and (10).

3.5 Probabilistic Processes

As explained in Section 2.2, we understand the behaviour of an interface as a probabilistic process. The basic mathematical object for representing probabilistic processes is a *probability space* [11,30]. A probability space is a triple $(\Omega, \mathcal{F}, f)$, where Ω is a sample space, that is, a non-empty set of possible outcomes, $\mathcal{F}$ is a non-empty set of subsets of Ω, and f is a function from $\mathcal{F}$ to $[0,1]$ that assigns a probability to each element in $\mathcal{F}$.

The set $\mathcal{F}$, and the function f have to fulfil the following constraints: The set $\mathcal{F}$ must be a σ-field over Ω, that is, $\mathcal{F}$ must be not be empty, it must contain Ω and be closed under complement[3] and countable union. The function f must be a *probability measure* on $\mathcal{F}$, that is, a function from $\mathcal{F}$ to $[0,1]$ such that $f(\emptyset) = 0$, $f(\Omega) = 1$, and for every sequence ω of disjoint sets in $\mathcal{F}$, the following holds: $f(\bigcup_{i=1}^{\#\omega} \omega[i]) = \sum_{i=1}^{\#\omega} f(\omega[i])$ [10]. The last property is referred to as countably additive, or σ-additive.

We represent a probabilistic execution H by a probability space with the set of traces of H as its sample space. If the set of possible traces in an execution is infinite, the probability of a single trace may be zero. To obtain the probability

[3] Note that this is the relative complement with respect to Ω, that is if $A \in \mathcal{F}$, then $\Omega \setminus A \in \mathcal{F}$.

that a certain sequence of events occurs up to a particular point in time, we can look at the probability of the set of all *extensions* of that sequence in a given trace set. Thus, instead of talking of the probability of a single trace, we are concerned with the probability of a set of traces with common prefix, called a *cone*. By $c(t, D)$ we denote the set of all continuations of t in D. For example we have that

$$\begin{aligned} c(\langle a\rangle, \{\langle a,a,b,b\rangle, \langle a,a,c,c\rangle\}) &= \{\langle a,a,b,b\rangle, \langle a,a,c,c\rangle\} \\ c(\langle a,a,b\rangle, \{\langle a,a,b,b\rangle, \langle a,a,c,c\rangle\}) &= \{\langle a,a,b,b\rangle\} \\ c(\langle b\rangle, \{\langle a,a,b,b\rangle, \langle a,a,c,c\rangle\}) &= \emptyset \end{aligned}$$

We define the cone of a finite trace t in a trace set D formally as:

Definition 1 (Cone). *Let D be a set of traces. The cone of a finite trace t, with regard to D, is the set of all traces in D with t as a prefix:*

$$\begin{aligned} c\,\_\,\_ &\in \mathcal{H} \times \mathbb{P}(\mathcal{H}) \rightarrow \mathbb{P}(\mathcal{H}) \\ c(t, D) &\overset{\mathsf{def}}{=} \{t' \in D \mid t \sqsubseteq t'\} \end{aligned}$$

We define the *cone set* with regard to a set of traces as:

Definition 2 (Cone set). *The cone set of a set of traces D consists of the cones with regard to D of each finite trace that is a prefix of a trace in D:*

$$\begin{aligned} C\_ &\in \mathbb{P}(\mathcal{H}) \rightarrow \mathbb{P}(\mathbb{P}(\mathcal{H})) \\ C(D) &\overset{\mathsf{def}}{=} \{c(t, D) \mid \#t \in \mathbb{N} \wedge \exists t' \in D : t \sqsubseteq t'\} \end{aligned}$$

We understand each trace in the trace set representing a probabilistic process H as a complete history of H. We therefore want to be able to distinguish the state where an execution stops after a given sequence and the state where an execution may continue with different alternatives after the sequence. We say that a finite trace t is complete with regard to a set of traces D if $t \in D$. Let D be a set of set of traces. We define the *complete extension* of the cone set of D as follows:

Definition 3 (Complete extended cone set). *The complete extended cone set of a set of traces D is the union of the cone set of D and the set of singleton sets containing the finite traces in D:*

$$\begin{aligned} C_E\,\_ &\in \mathbb{P}(\mathcal{H}) \rightarrow \mathbb{P}(\mathbb{P}(\mathcal{H})) \\ C_E(D) &\overset{\mathsf{def}}{=} C(D) \cup \{\{t\} \subseteq D \mid \#t \in \mathbb{N}\} \end{aligned}$$

We define a probabilistic execution H formally as:

Definition 4 (Probabilistic execution). *A probabilistic execution H is a probability space:*

$$\mathbb{P}(\mathcal{H}) \times \mathbb{P}(\mathbb{P}(\mathcal{H})) \times (\mathbb{P}(\mathcal{H}) \rightarrow [0, 1])$$

whose elements we refer to as D_H, $\mathcal{F}_H$ and f_H where D_H is the set of traces of H, $\mathcal{F}_H$ is the σ-field generated by $C_E(D_H)$, that is the intersection of all σ-fields including $C_E(D_H)$, called the cone-σ-field of D_H, and f_H is a probability measure on $\mathcal{F}_H$.

If D_H is countable then $\mathbb{P}(D_H)$ (the power set of D_H) is the largest σ-field that can be generated from D_H and it is common to define $\mathcal{F}_H$ as $\mathbb{P}(D_H)$. If D_H is uncountable, then, assuming the continuum hypothesis, which states that there is no set whose cardinality is strictly between that of the integers and that of the real numbers, the cardinality of D_H equals the cardinality of the real numbers, and hence of $[0, 1]$. This implies that there are subsets of $\mathbb{P}(D_H)$ which are not measurable, and $\mathcal{F}_H$ is therefore usually a proper subset of $\mathbb{P}(D_H)$ [8]. A simple example of a process with uncountable sample space, is the process that throws a fair coin an infinite number of times [23,9]. Each execution of this process can be represented by an infinite sequence of zeroes and ones, where 0 represents "head" and 1 represents "tail". The set of infinite sequences of zeroes and ones is uncountable, which can be shown by a diagonalisation argument [5].

3.6 Probabilistic Interface Execution

We define the set of traces of an interface n as any well-formed trace consisting solely of events where n is the active party. Formally:

$$\mathcal{H}_n \stackrel{\mathsf{def}}{=} \mathcal{H} \cap {\mathcal{E}_n}^{\omega}$$

We define the behavioural representation of an interface n as a function of its queue history. A queue history of an interface n is a well-formed trace consisting solely of transmission events with n as consumer. That a queue history is well formed implies that the events in the queue history are totally ordered by time. We let $\mathcal{B}_n$ denote the set of queue histories of an interface n. Formally:

$$\mathcal{B}_n \stackrel{\mathsf{def}}{=} \mathcal{H} \cap \mathcal{E}_n^{\downarrow\,\omega}$$

A queue history serves as a scheduler for an interface, thereby uniquely determining its behaviour [27,6]. Hence, a queue history gives rise to a probabilistic execution of an interface. That is, the probabilistic behaviour of an interface n is represented by a function of complete queue histories for n. A *complete queue history* for an interface n records the messages transmitted to n for the whole execution of n, as opposed to a *partial queue history* that records the messages transmitted to n until some (finite) point in time. We define a probabilistic interface execution formally as:

Definition 5 (Probabilistic interface execution). *A probabilistic execution of an interface n is a function that for every complete queue history of n returns a probabilistic execution:*

$$I_{n}\_ \in \mathcal{B}_n \rightarrow \mathbb{P}(\mathcal{H}_n) \times \mathbb{P}(\mathbb{P}(\mathcal{H}_n)) \times (\mathbb{P}(\mathcal{H}_n) \rightarrow [0,1])^4$$

Hence, $I_n(\alpha)$ denotes the probabilistic execution of n given the complete queue history α. We let $D_n(\alpha), \mathcal{F}_n(\alpha)$ and $f_n(\alpha)$ denote the projections on the three elements of the probabilistic execution of n given queue history α. I.e. $I_n(\alpha) = (D_n(\alpha), \mathcal{F}_n(\alpha), f_N(\alpha))$.

In Section 2 we described how an interface may choose to do nothing. In the denotational trace semantics we represent doing nothing by the empty trace. Hence, given an interface n and a complete queue history α, $D_n(\alpha)$ may consist of only the empty trace, but it may never be empty.

The queue history of an interface represents the input to it from other interfaces. In Section 2.2 we described informally our assumptions about how interfaces interact through queues. In particular, we emphasised that an interface can only consume messages already in its queue, and the same message can be consumed only once. We also assumed that an interface does not send messages to itself. Hence, we require that any $t \in D_n(\alpha)$ fulfils the following constraints:

$$\begin{aligned}
&(11)\quad \mathsf{let}\ i = (\{?\} \times \mathcal{M}) \circledS t \\
&\qquad\qquad \mathsf{in}\ \Pi_{\{1,2\}}.(\Pi_{\{2\}}.i) \sqsubseteq \Pi_{\{1,2\}}.(\Pi_{\{2\}}.\alpha) \wedge \forall j \in \{1..\#i\} : q.\alpha[j] < q.i[j] \\
&(12)\quad \forall j \in [1..\#t] : k.t[j] \neq co.t[j]
\end{aligned}$$

The first conjunct of constraint (11) states that the sequence of consumed messages in t is a prefix of the messages in α, when disregarding time. Thus, we ensure that n only consumes messages it has received in its queue and that they are consumed in the order they arrived. The second conjunct of constraint (11) ensures that messages are only consumed from the queue after they have arrived and with a non-zero delay. Constraint (12) ensures that an interface does not send messages to itself.

A complete queue history of an interface uniquely determines its behaviour. However, we are only interested in capturing time causal behaviour in the sense that the behaviour of an interface at a given point in time should depend only on its input up to and including that point in time and be independent of the content of its queue at any later point.

In order to formalise this constraint, we first define an operator for truncating a trace at a certain point in time. By $t{\downarrow_k}$ we denote the timed truncation of t, that is, the prefix of t including all events in t with a time-stamp lower than or equal to k. For example we have that:

$$\begin{aligned}
\langle ?\langle c,q,r,1\rangle, !\langle a,r,p,3\rangle, !\langle b,r,p,5\rangle\rangle{\downarrow_4} &= \langle ?\langle c,q,r,1\rangle, !\langle a,r,p,3\rangle\rangle \\
\langle ?\langle c,q,r,1\rangle, !\langle a,r,p,3\rangle, !\langle b,r,p,5\rangle\rangle{\downarrow_8} &= \langle ?\langle c,q,r,1\rangle, !\langle a,r,p,3\rangle, !\langle b,r,p,5\rangle\rangle \\
\langle ?\langle c,q,r,\tfrac{1}{2}\rangle, !\langle a,r,p,\tfrac{3}{2}\rangle, !\langle b,r,p,\tfrac{5}{2}\rangle\rangle{\downarrow_{\frac{3}{2}}} &= \langle ?\langle c,q,r,\tfrac{1}{2}\rangle, !\langle a,r,p,\tfrac{3}{2}\rangle\rangle
\end{aligned}$$

[4] Note that the type of I_n ensures that for any $\alpha \in \mathcal{B}_n : \mathsf{rng}.\alpha \cap ev.D_n(\alpha) = \emptyset$.

The function $\downarrow$ is defined formally as follows:

(13)
$$
\begin{aligned}
&\_\downarrow\_ \in \mathcal{H} \times \mathcal{Q} \rightarrow \mathcal{H}\\
&t\downarrow_k \overset{\mathsf{def}}{=} \begin{cases} \langle\rangle \text{ if } t = \langle\rangle \vee q.t[1] > k \\ r \text{ otherwise where } r \sqsubseteq t \wedge q.r[\#r] \leq k \\ \qquad\qquad\qquad\qquad\quad \wedge \ (\#r < \#t \Rightarrow q.t[\#r+1] > k) \end{cases}
\end{aligned}
$$

We overload the timed truncation operator to sets of traces as follows:

$$
\begin{aligned}
&\_\downarrow\_ \in \mathbb{P}(\mathcal{H}) \times \mathcal{Q} \rightarrow \mathbb{P}(\mathcal{H})\\
&S\downarrow_k \overset{\mathsf{def}}{=} \{t\downarrow_k \mid t \in S\}
\end{aligned}
$$

We may then formalise the time causality as follows:

$$
\begin{aligned}
\forall \alpha, \beta \in \mathcal{B}_n : \forall q \in \mathcal{Q} : \alpha\downarrow_q = \beta\downarrow_q \Rightarrow (&D_n(\alpha)\downarrow_q = D_n(\beta)\downarrow_q) \wedge \\
&((\forall t_1 \in D_n(\alpha) : \forall t_2 \in D_n(\beta)) : t_1\downarrow_q = t_2\downarrow_q) \Rightarrow \\
&\qquad (f_n(\alpha)(c(t_1\downarrow_q, D_n(\alpha))) = f_n(\beta)(c(t_2\downarrow_q, D_n(\beta))))
\end{aligned}
$$

The first conjunct states that for all queue histories α, β of an interface n, and for all points in time q, if α and β are equal until time q, then the trace sets $D_n(\alpha)$ and $D_n(\beta)$ are also equal until time q. The second conjunct states that if α and β are equal until time q, and we have two traces in $D_n(\alpha)$ and $D_n(\beta)$ that are equal until time q, then the likelihoods of the cones of the two traces truncated at time q in their respective trace sets are equal. Thus, the constraint ensures that the behaviour of an interface at a given point in time depends on its queue history up to and including that point in time, and is independent of the content of its queue history at any later point.

4 Denotational Representation of an Interface with a Notion of Risk

Having introduced the underlying semantic model, the next step is to extend it with concepts from risk analysis according to the conceptual model in Figure 1. As already explained, the purpose of extending the semantic model with risk analysis concepts is to represent risks as an integrated part of interface and component behaviour.

4.1 Assets

An *asset* is a physical or conceptual entity which is of value for a stakeholder, that is, for an interface (see Section 2.1) and which the stakeholder wants to protect. We let $\mathcal{A}$ denote the set of all assets and $\mathcal{A}_n$ denote the set of assets of interface n. Note that $\mathcal{A}_n$ may be empty. We require:

(14)
$$\forall n, n' \in \mathcal{P} : n \neq n' \Rightarrow \mathcal{A}_n \cap \mathcal{A}_{n'} = \emptyset$$

Hence, assets are not shared between interfaces.

4.2 Incidents and Consequences

As explained in Section 2.1 an *incident* is an event that reduces the value of one or more assets. This is a general notion of incident, and of course, an asset may be harmed in different ways, depending on the type of asset. Some examples are reception of corrupted data, transmission of classified data to an unauthorised user, or slow response to a request. We provide a formal model for representing events that harm assets. For a discussion of how to obtain further risk analysis results for components, such as the cause of an unwanted incident, its consequence and probability we refer to [3].

In order to represent incidents formally we need a way to measure harm inflicted upon an asset by an event. We represent the *consequence* of an incident by a positive integer indicating its level of seriousness with regard to the asset in question. For example, if the reception of corrupted data is considered to be more serious for a given asset than the transmission of classified data to an unauthorised user, the former has a greater consequence than the latter with regard to this asset. We introduce a function

$$cv_{n\,\_\_} \in \mathcal{E}_n \times \mathcal{A}_n \to \mathbb{N} \tag{15}$$

that for an event e and asset a of an interface n, yields the consequence of e to a if e is an incident, and 0 otherwise. Hence, an event with consequence larger than zero for a given asset is an incident with regard to that asset. Note that the same event may be an incident with respect to more than one asset; moreover, an event that is not an incident with respect to one asset, may be an incident with respect to another.

4.3 Incident Probability

The *probability* that an incident e occurs during an execution corresponds to the probability of the set of traces in which e occurs. Since the events in each trace are totally ordered by time, and all events include a time-stamp, each event in a trace is unique. This means that a given incident occurs only once in each trace.

We can express the set describing the occurrence of an incident e, in a probabilistic execution H, as $occ(e, D_H)$ where the function occ is formally defined as:

$$\begin{aligned} (16) \qquad & occ_{\_\_} \in \mathcal{E} \times \mathbb{P}(\mathcal{H}) \to \mathbb{P}(\mathcal{H}) \\ & occ(e, D) \overset{\mathsf{def}}{=} \{t \in D \,|\, e \in \mathsf{rng}.t\} \end{aligned}$$

($\mathsf{rng}.t$ yields the range of the trace t, i.e., the set of events occurring in t). The set $occ(e, D_H)$ corresponds to the union of all cones $c(t, D_H)$ where e occurs in t (see Section 3.5). Any union of cones can be described as a disjoint set of cones [26]. As described in Section 3, we assume that an interface is assigned at most a countable number of signals and we assume that time-stamps are rational numbers. Hence, it follows that an interface has a countable number of events.

Since the set of finite sequences formed from a countable set is countable [18], the union of cones where e occurs in t is countable. Since by definition, the cone-σ-field of an execution H, is closed under countable union, the occurrence of an incident can be represented as a countable union of disjoint cones, that is, it is an element in the cone-σ-field of H and thereby has a measure.

4.4 Risk Function

The *risk function* of an interface n takes a consequence, a probability and an asset as arguments and yields a risk value represented by a positive integer. Formally:

$$rf_n \_\_ \in \mathbb{N} \times [0,1] \times \mathcal{A}_n \rightarrow \mathbb{N} \tag{17}$$

The risk value associated with an incident e in an execution H, with regard to an asset a, depends on the probability of e in H and its consequence value. We require that

$$rf_n(c, p, a) = 0 \Leftrightarrow c = 0 \vee p = 0$$

Hence, only incidents have a positive risk value, and any incident has a positive risk value.

4.5 Interface with a Notion of Risk

Putting everything together we end up with the following representation of an interface:

Definition 6 (Semantics of an interface). *An interface n is represented by a quadruple*

$$(I_n, \mathcal{A}_n, cv_n, rf_n)$$

consisting of its probabilistic interface execution, assets, consequence function and risk function as explained above.

Given such a quadruple we have the necessary means to calculate the risks associated with an interface for a given queue history. A *risk* is a pair of an incident and its risk value. Hence, for the queue history $\alpha \in \mathcal{B}_n$ and asset $a \in \mathcal{A}_n$ the associated risks are

$$\{rv \mid rv = rf_n(cv(e,a), f_n(occ(e, D_n(\alpha))), a) \wedge rv > 0 \wedge e \in \mathcal{E}_n\}$$

5 Denotational Representation of Component Behaviour

A component is a collection of interfaces, some of which may interact. We may view a single interface as a basic component. A composite component is a component containing at least two interfaces (or basic components). In this section we lift the notion of probabilistic execution from interfaces to components. Furthermore, we explain how we obtain the behaviour of a component from the behaviours of its sub-components. In this section we do not consider the issue of hiding; this is the topic of Section 7.

In Section 5.1 we introduce the notion of conditional probability measure, conditional probabilistic execution and probabilistic component execution. In Section 5.2 we characterise how to obtain the trace set of a composite component from the trace sets of its sub-components. The cone-σ-field of a probabilistic component execution is generated straightforwardly from that. In Section 5.3 we explain how to define the conditional probability measure for the cone-σ-field of a composite component from the conditional probability measures of its sub-components. Finally, in Section 5.4, we define a probabilistic component execution of a composite component in terms of the probabilistic component executions of its sub-components. We sketch the proof strategies for the lemmas and theorems in this section and refer to Brændeland et al. [2] for the full proofs.

5.1 Probabilistic Component Execution

The behaviour of a component is completely determined by the set of interfaces it consists of. We identify a component by the set of names of its interfaces. Hence, the behaviour of the component $\{n\}$ consisting of only one interface n, is identical to the behaviour of the interface n. For any set of interfaces N we define:

(18) $$\mathcal{E}_N \stackrel{\mathsf{def}}{=} \bigcup_{n \in N} \mathcal{E}_n$$

(19) $$\mathcal{E}_N^{\downarrow} \stackrel{\mathsf{def}}{=} \bigcup_{n \in N} \mathcal{E}_n^{\downarrow}$$

(20) $$\mathcal{H}_N \stackrel{\mathsf{def}}{=} \mathcal{H} \cap {\mathcal{E}_N}^{\omega}$$

(21) $$\mathcal{B}_N \stackrel{\mathsf{def}}{=} \mathcal{H} \cap {\mathcal{E}_N^{\downarrow}}^{\omega}$$

Just as for interfaces, we define the behavioural representation of a component N as a function of its queue history. For a single interface a queue history α resolves the external nondeterminism caused by the environment. Since we assume that an interface does not send messages to itself there is no internal non-determinism to resolve. The function representing an interface returns a probabilistic execution which is a probability space. Given an interface n it follows from the definition of a probabilistic execution, that for any queue history $\alpha \in \mathcal{B}_n$, we have $f_n(\alpha)(D_n(\alpha)) = 1$.

For a component N consisting of two or more sub-components, a queue history α must resolve both external and internal non-determinism. For a given queue history α the behaviour of N, is obtained from the behaviours of the sub-components of N that are possible with regard to α. That is, all internal choices concerning interactions between the sub-components of N are fixed by α. This means that the probability of the set of traces of N given a queue history α may be lower than 1, violating the requirement of a probability measure. In order to formally represent the behaviour of a component we therefore introduce the notion of a *conditional probability measure.*

Definition 7 (Conditional probability measure). *Let D be a non-empty set and $\mathcal{F}$ be a σ-field over D. A conditional probability measure f on $\mathcal{F}$ is a function that assigns a value in $[0,1]$ to each element of $\mathcal{F}$ such that; either $f(A) = 0$ for all A in $\mathcal{F}$, or there exists a constant $c \in \langle 0,1]$[5] such that the function f' defined by $f'(A) = f(A)/c$ is a probability measure on $\mathcal{F}$.*

We define a conditional probabilistic execution H formally as:

Definition 8 (Conditional probabilistic execution). *A conditional probabilistic execution H is a measure space [11]:*

$$\mathbb{P}(\mathcal{H}) \times \mathbb{P}(\mathbb{P}(\mathcal{H})) \times (\mathbb{P}(\mathcal{H}) \to [0,1])$$

whose elements we refer to as D_H, $\mathcal{F}_H$ and f_H where D_H is the set of traces of H, $\mathcal{F}_H$ is the cone-σ-field of D_H, and f_H is a conditional probability measure on $\mathcal{F}_H$.

We define a probabilistic component execution formally as:

Definition 9 (Probabilistic component execution). *A probabilistic execution of a component N is a function I_N that for every complete queue history of N returns a conditional probabilistic execution:*

$$I_{N\,\_} \in \mathcal{B}_N \to \mathbb{P}(\mathcal{H}_N) \times \mathbb{P}(\mathbb{P}(\mathcal{H}_N)) \times (\mathbb{P}(\mathcal{H}_N) \to [0,1])$$

Hence, $I_N(\alpha)$ denotes the probabilistic execution of N given the complete queue history α. We let $D_N(\alpha), \mathcal{F}_N(\alpha)$ and $f_N(\alpha)$ denote the canonical projections of the probabilistic component execution on its elements.

5.2 Trace Sets of a Composite Component

For a given queue history α, the combined trace sets $D_{N_1}(\mathcal{E}_{N_1}^{\downarrow} \circledS \alpha)$ and $D_{N_2}(\mathcal{E}_{N_2}^{\downarrow} \circledS \alpha)$ such that all the transmission events from N_1 to N_2 are in α and the other way around, constitute the legal set of traces of the composition of N_1 and N_2. Given two probabilistic component executions I_{N_1} and I_{N_2} such that $N_1 \cap N_2 = \emptyset$, for each $\alpha \in \mathcal{B}_{N_1 \cup N_2}$ we define their composite trace set formally as:

$$\begin{aligned}
(22)\quad & D_{N_1} \otimes D_{N_2\,\_} \in \mathcal{B}_{N_1 \cup N_2} \to \mathbb{P}(\mathcal{H}_{N_1 \cup N_2}) \\
& D_{N_1} \otimes D_{N_2}(\alpha) \overset{\mathsf{def}}{=} \\
& \quad \{t \in \mathcal{H}_{N_1 \cup N_2} | \mathcal{E}_{N_1} \circledS t \in D_{N_1}(\mathcal{E}_{N_1}^{\downarrow} \circledS \alpha) \wedge \mathcal{E}_{N_2} \circledS t \in D_{N_2}(\mathcal{E}_{N_2}^{\downarrow} \circledS \alpha) \wedge \\
& \quad\ (\{!\} \times \mathcal{S} \times N_2 \times N_1 \times \mathcal{Q}) \circledS t \sqsubseteq (\{!\} \times \mathcal{S} \times N_2 \times N_1 \times \mathcal{Q}) \circledS \alpha \wedge \\
& \quad\ (\{!\} \times \mathcal{S} \times N_1 \times N_2 \times \mathcal{Q}) \circledS t \sqsubseteq (\{!\} \times \mathcal{S} \times N_1 \times N_2 \times \mathcal{Q}) \circledS \alpha\}
\end{aligned}$$

[5] We use $\langle a, b\rangle$ to denote the open interval $\{x \mid a < x < b\}$.

The definition ensures that the messages from N_2 consumed by N_1 are in the queue history of N_1 and vice versa. The operator $\otimes$ is obviously commutative and also associative since the sets of interfaces of each component are disjoint.

For each $\alpha \in \mathcal{B}_{N_1 \cup N_2}$ the cone-σ-field is generated as before. Hence, we define the cone-σ-field of a composite component as follows:

(23) $$\mathcal{F}_{N_1} \otimes \mathcal{F}_{N_2}(\alpha) \overset{\mathsf{def}}{=} \sigma(C_E(D_{N_1} \otimes D_{N_2}(\alpha)))$$

where $\sigma(D)$ denotes the σ-field generated by the set D. We refer to $C_E(D_{N_1} \otimes D_{N_2}(\alpha))$ as the *composite extended cone set* of $N_1 \cup N_2$.

5.3 Conditional Probability Measure of a Composite Component

Consider two components C and O such that $C \cap O = \emptyset$. As described in Section 2, it is possible to decompose a probabilistic choice over actions in such a way that it never involves more than one interface. We may therefore assume that for a given queue history $\alpha \in \mathcal{B}_{C \cup O}$ the behaviour represented by $D_C(\mathcal{E}_C^{\downarrow} \circledS\, \alpha)$ is independent of the behaviour represented by $D_O(\mathcal{E}_O^{\downarrow} \circledS\, \alpha)$. Given this assumption the probability of a certain behaviour of the composed component equals the product of the probabilities of the corresponding behaviours of C and O, by the law of statistical independence. As explained in Section 3.5, to obtain the probability that a certain sequence of events t occurs up to a particular point in time in a set of traces D, we can look at the cone of t in D. For a given cone $c \in C_E(D_C \otimes D_O(\alpha))$ we obtain the corresponding behaviours of C and O by filtering c on the events of C and O, respectively.

The above observation with regard to cones does not necessarily hold for all elements of $\mathcal{F}_C \otimes \mathcal{F}_O(\alpha)$. The following simple example illustrates that the probability of an element in $\mathcal{F}_C \otimes \mathcal{F}_O(\alpha)$, which is not a cone, is not necessarily the product of the corresponding elements in $\mathcal{F}_C(\mathcal{E}_C^{\downarrow} \circledS\, \alpha)$ and $\mathcal{F}_O(\mathcal{E}_O^{\downarrow} \circledS\, \alpha)$. Assume that the component C tosses a fair coin and that the component O tosses an Othello piece (a disk with a light and a dark face). We assign the singleton time-stamp set $\{1\}$ to C and the singleton time-stamp set $\{2\}$ to O. Hence, the traces of each may only contain one event. For the purpose of readability we represent in the following the events by their signals. The assigned time-stamps ensure that the coin toss represented by the events $\{h, t\}$ comes before the Othello piece toss. We have:

$$\begin{aligned}
D_C(\langle\rangle) &= \{\langle h\rangle, \langle t\rangle\} \\
\mathcal{F}_C(\langle\rangle) &= \{\emptyset, \{\langle h\rangle\}, \{\langle t\rangle\}, \{\langle h\rangle, \langle t\rangle\}\} \\
f_C(\langle\rangle)(\{\langle h\rangle\}) &= 0.5 \\
f_C(\langle\rangle)(\{\langle t\rangle\}) &= 0.5
\end{aligned}$$

and

$$\begin{aligned}
D_O(\langle\rangle) &= \{\langle b\rangle, \langle w\rangle\} \\
\mathcal{F}_O(\langle\rangle) &= \{\emptyset, \{\langle b\rangle\}, \{\langle w\rangle\}, \{\langle b\rangle, \langle w\rangle\}\} \\
f_O(\langle\rangle)(\{\langle b\rangle\}) &= 0.5 \\
f_O(\langle\rangle)(\{\langle w\rangle\}) &= 0.5
\end{aligned}$$

Let $D_{CO} = D_C \otimes D_O$. The components interacts only with the environment, not with each other. We have:

$$D_{CO}(\langle\rangle) = \{\langle h, b\rangle, \langle h, w\rangle, \langle t, b\rangle, \langle t, w\rangle\}$$

We assume that each element in the sample space (trace set) of the composite component has the same probability. Since the sample space is finite, the probabilities are given by discrete uniform distribution, that is each trace in $D_{CO}(\langle\rangle)$ has a probability of 0.25. Since the traces are mutually exclusive, it follows by the laws of probability that the probability of $\{\langle h, b\rangle\} \cup \{\langle t, w\rangle\}$ is the sum of the probabilities of $\{\langle h, b\rangle\}$ and $\{\langle t, w\rangle\}$, that is 0.5. But this is not the same as $f_C(\langle\rangle)(\{\langle h\rangle, \langle t\rangle\}) \cdot f_O(\langle\rangle)(\{\langle b\rangle, \langle w\rangle\})$[6], which is 1.

Since there is no internal communication between C and O, there is no internal non-determinism to be resolved. If we replace the component O with the component R, which simply consumes whatever C transmits, a complete queue history of the composite component reflects only one possible interaction between C and R. Let $D_{CR} = D_C \otimes D_R$. To make visible the compatibility between the trace set and the queue history we include the whole events in the trace sets of the composite component. We have:

$$\begin{aligned} D_{CR}(\langle !\langle h, C, R, 1\rangle\rangle) =& \{\langle !\langle h, C, R, 1\rangle, ?\langle h, C, R, 2\rangle\rangle\} \\ D_{CR}(\langle !\langle t, C, R, 1\rangle\rangle) =& \{\langle !\langle t, C, R, 1\rangle, ?\langle t, C, R, 2\rangle\rangle\} \end{aligned}$$

For a given queue history α, the set $\mathcal{E}_C \circledS D_{CR}(\alpha)$ is a subset of the trace set $D_C(\mathcal{E}_C^{\downarrow} \circledS \alpha)$ that is possible with regard to α (that $\mathcal{E}_C \circledS D_{CR}(\alpha)$ is a subset of $D_C(\mathcal{E}_C^{\downarrow} \circledS \alpha)$ is shown in [2]). We call the set of traces of C that are possible with regard to a given queue history α and component R for $CT_{C-R}(\alpha)$, which is short for *conditional traces*.

Given two components N_1 and N_2 and a complete queue history $\alpha \in \mathcal{B}_{N_1 \cup N_2}$, we define the set of conditional traces of N_1 with regard to α and N_2 formally as:

$$(24) \qquad CT_{N_1-N_2}(\alpha) \overset{\mathsf{def}}{=} \Big\{t \in D_{N_1}(\mathcal{E}_{N_1}^{\downarrow} \circledS \alpha) \mid (\{!\} \times \mathcal{S} \times N_1 \times N_2 \times \mathcal{Q}) \circledS t \sqsubseteq \\ (\{!\} \times \mathcal{S} \times N_1 \times N_2 \times \mathcal{Q}) \circledS \alpha\Big\}$$

Lemma 1. *Let I_{N_1} and I_{N_2} be two probabilistic component executions such that $N_1 \cap N_2 = \emptyset$ and let α be a queue history in $\mathcal{B}_{N_1 \cup N_2}$. Then*

$$CT_{N_1-N_2}(\alpha) \in \mathcal{F}_{N_1}(\mathcal{E}_{N_1}^{\downarrow} \circledS \alpha) \wedge CT_{N_2-N_1}(\alpha) \in \mathcal{F}_{N_2}(\mathcal{E}_{N_2}^{\downarrow} \circledS \alpha)$$

Proof sketch: The set $CT_{N_1-N2}(\alpha)$ includes all traces in $D_{N_1}(\mathcal{E}_{N_1}^{\downarrow} \circledS \alpha)$ that are compatible with α, i.e., traces that are prefixes of α when filtered on the transmission events from N_1 to N_2. The key is to show that this set can be constructed as an element in $\mathcal{F}_{N_1}(\mathcal{E}_{N_1}^{\downarrow} \circledS \alpha)$. If α is infinite, this set corresponds

[6] We use $\cdot$ to denote normal multiplication.

to (1) the union of all finite traces in $D_{N_1}(\mathcal{E}^{\downarrow}_{N_1} \circledS \alpha)$ that are compatible with α and (2) the set obtained by constructing countable unions of cones of traces that are compatible with finite prefixes of $\alpha|_i$ for all $i \in \mathbb{N}$ (where $\alpha|_i$ denotes the prefix of α of length i) and then construct the countable intersection of all such countable unions of cones. If α is finite the proof is simpler, and we do not got into the details here. The same procedure may be followed to show that $CT_{N_2-N_1}(\alpha) \in \mathcal{F}_{N_2}(\mathcal{E}^{\downarrow}_{N_2} \circledS \alpha)$.

As illustrated by the example above, we cannot obtain a measure on a composite cone-σ-field in the same manner as for a composite extended cone set. In order to define a conditional probability measure on a composite cone-σ-field, we first define a measure on the composite extended cone set it is generated from. We then show that this measure can be uniquely extended to a conditional probability measure on the generated cone-σ-field. Given two probabilistic component executions I_{N_1} and I_{N_2} such that $N_1 \cap N_2 = \emptyset$, for each $\alpha \in \mathcal{B}_{N_1 \cup N_2}$ we define a measure $\mu_{N_1} \otimes \mu_{N_2}(\alpha)$ on $C_E(D_{N_1} \otimes D_{N_2}(\alpha))$ formally as follows:

$$(25) \qquad \begin{aligned} &\mu_{N_1} \otimes \mu_{N_2} \_ \in \mathcal{B}_{N_1 \cup N_2} \rightarrow (C_E(D_{N_1} \otimes D_{N_2}(\alpha)) \rightarrow [0,1]) \\ &\mu_{N_1} \otimes \mu_{N_2}(\alpha)(c) \overset{\mathsf{def}}{=} f_{N_1}(\mathcal{E}^{\downarrow}_{N_1} \circledS \alpha)(\mathcal{E}_{N_1} \circledS c) \cdot f_{N_2}(\mathcal{E}^{\downarrow}_{N_2} \circledS \alpha)(\mathcal{E}_{N_2} \circledS c) \end{aligned}$$

Theorem 1. *The function $\mu_{N_1} \otimes \mu_{N_2}(\alpha)$ is well defined.*

Proof sketch: For any $c \in C_E(D_{N_1} \otimes D_{N_2}(\alpha))$ we must show that $(\mathcal{E}_{N_1} \circledS c) \in \mathcal{F}_{N_1}(\mathcal{E}^{\downarrow}_{N_1} \circledS \alpha)$ and $(\mathcal{E}_{N_2} \circledS c) \in \mathcal{F}_{N_2}(\mathcal{E}^{\downarrow}_{N_2} \circledS \alpha)$. If c is a singleton (containing exactly one trace) the proof follows from the fact that (1): if $(D, \mathcal{F}, f)$ is a conditional probabilistic execution and t is a trace in D, then $\{t\} \in \mathcal{F}$ [23], and (2): that we can show $\mathcal{E}_{N_1} \circledS t \in D_{N_1}(\mathcal{E}^{\downarrow}_{N_1} \circledS \alpha) \wedge \mathcal{E}_{N_2} \circledS t \in D_{N_2}(\mathcal{E}^{\downarrow}_{N_2} \circledS \alpha)$ from Definition 3 and definition (22).

If c is a cone $c(t, D_{N_1} \otimes D_{N_2}(\alpha))$ in $C(D_{N_1} \otimes D_{N_2}(\alpha))$, we show that $CT_{N_1-N_2}(\alpha)$, intersected with $c(\mathcal{E}_{N_1} \circledS t, D_{N_1}(\mathcal{E}^{\downarrow}_{N_1} \circledS \alpha))$ and the traces in $D_{N_1}(\mathcal{E}^{\downarrow}_{N_1} \circledS \alpha)$ that are compatible with t with regard to the timing of events, is an element of $\mathcal{F}_{N_1}(\mathcal{E}^{\downarrow}_{N_1} \circledS \alpha)$ that equals $(\mathcal{E}_{N_1} \circledS c)$. We follow the same procedure to show that $(\mathcal{E}_{N_2} \circledS c) \in \mathcal{F}_{N_2}(\mathcal{E}^{\downarrow}_{N_2} \circledS \alpha)$.

Lemma 2. *Let I_{N_1} and I_{N_2} be two probabilistic component executions such that $N_1 \cap N_2 = \emptyset$ and let $\mu_{N_1} \otimes \mu_{N_2}$ be a measure on the extended cones set of $D_{N_1} \otimes D_{N_2}$ as defined by (25). Then, for all complete queue histories $\alpha \in \mathcal{B}_{N_1 \cup N_2}$*

1. $\mu_{N_1} \otimes \mu_{N_2}(\alpha)(\emptyset) = 0$
2. $\mu_{N_1} \otimes \mu_{N_2}(\alpha)$ *is σ-additive*
3. $\mu_{N_1} \otimes \mu_{N_2}(\alpha)(D_{N_1} \otimes D_{N_2}(\alpha)) \leq 1$

Proof sketch: We sketch the proof strategy for point 2 of Lemma 2. The proofs of point 1 and 3 are simpler, and we do not go into the details here. Assume ϕ is a sequence of disjoint sets in $C_E(D_{N_1} \otimes D_{N_2}(\alpha))$. We construct a sequence ψ of length $\#\phi$ such that $\forall i \in [1..\#\phi] : \psi[i] = \{(\mathcal{E}_{N_1} \circledS t, \mathcal{E}_{N_2} \circledS t) \mid t \in$

$\phi[i]\}$ and show that $\bigcup_{i=1}^{\#\psi} \psi[i] = \mathcal{E}_{N_1} Ⓢ \bigcup_{i=1}^{\#\phi} \phi[i] \times \mathcal{E}_{N_2} Ⓢ \bigcup_{i=1}^{\#\phi} \phi[i]$. It follows by Theorem 1 that $(\mathcal{E}_{N_1} Ⓢ \bigcup_{i=1}^{\#\phi} \phi[i]) \times (\mathcal{E}_{N_2} Ⓢ \bigcup_{i=1}^{\#\phi} \phi[i])$ is a measurable rectangle [11] in $\mathcal{F}_{N_1}(\mathcal{E}^{\downarrow}_{N_1} Ⓢ \alpha) \times \mathcal{F}_{N_2}(\mathcal{E}^{\downarrow}_{N_2} Ⓢ \alpha)$. From the above, and the product measure theorem [11] it can be shown that $f_{N_1}(\mathcal{E}^{\downarrow}_{N_1} Ⓢ \alpha)(\mathcal{E}_{N_1} Ⓢ \bigcup_{i=1}^{\#\phi} \phi[i]) \cdot$
$f_{N_2}(\mathcal{E}^{\downarrow}_{N_2} Ⓢ \alpha)(\mathcal{E}_{N_2} Ⓢ \bigcup_{i=1}^{\#\phi} \phi[i]) = \sum_{i=1}^{\#\phi} f_{N_1}(\mathcal{E}^{\downarrow}_{N_1} Ⓢ \alpha)(\mathcal{E}_{N_1} Ⓢ \phi[i]) \cdot$
$f_{N_2}(\mathcal{E}^{\downarrow}_{N_2} Ⓢ \alpha)(\mathcal{E}_{N_2} Ⓢ \phi[i])$.

Theorem 2. *There exists a unique extension of* $\mu_{N_1} \otimes \mu_{N_2}(\alpha)$ *to the cone-*σ*-field* $\mathcal{F}_{N_1} \otimes \mathcal{F}_{N_2}(\alpha)$*.*

Proof sketch: We extend $C_E(D_{N_1} \otimes D_{N_2}(\alpha))$ in a stepwise manner to a set obtained by first adding all complements of the elements in $C_E(D_{N_1} \otimes D_{N_2}(\alpha))$, then adding the finite intersections of the new elements and finally adding finite unions of disjoint elements. For each step we extend $\mu_{N_1} \otimes \mu_{N_2}(\alpha)$ and show that the extension is σ-additive. We end up with a finite measure on the field generated by $C_E(D_{N_1} \otimes D_{N_2}(\alpha))$. By the extension theorem [11] it follows that this measure can be uniquely extended to a measure on $\mathcal{F}_{N_1} \otimes \mathcal{F}_{N_2}(\alpha)$.

Corollary 1. *Let* $f_{N_1} \otimes f_{N_2}(\alpha)$ *be the unique extension of* $\mu_{N_1} \otimes \mu_{N_2}(\alpha)$ *to the cone-*σ*-field* $\mathcal{F}_{N_1} \otimes \mathcal{F}_{N_2}(\alpha)$*. Then* $f_{N_1} \otimes f_{N_2}(\alpha)$ *is a conditional probability measure on* $\mathcal{F}_{N_1} \otimes \mathcal{F}_{N_2}(\alpha)$*.*

Proof sketch: We first show that $\forall \alpha \in \mathcal{B}_{N_1 \cup N_2} : f_{N_1} \otimes f_{N_2}(\alpha)(D_{N_1} \otimes D_{N_2}(\alpha)) \leq 1$. When $f_{N_1} \otimes f_{N_2}(\alpha)$ is a measure on $\mathcal{F}_{N_1} \otimes \mathcal{F}_{N_2}(\alpha)$ such that $f_{N_1} \otimes f_{N_2}(\alpha)(D_{N_1} \otimes D_{N_2}(\alpha)) \leq 1$ we can show that $f_{N_1} \otimes f_{N_2}(\alpha)$ is a conditional probability measure on $\mathcal{F}_{N_1} \otimes \mathcal{F}_{N_2}(\alpha)$.

5.4 Composition of Probabilistic Component Executions

We may now lift the $\otimes$-operator to probabilistic component executions. Let I_{N_1} and I_{N_2} be probabilistic component executions such that $N_1 \cap N_2 = \emptyset$. For any $\alpha \in \mathcal{B}_{N_1 \cup N_2}$ we define:

$$(26) \qquad I_{N_1} \otimes I_{N_2}(\alpha) \stackrel{\mathsf{def}}{=} (D_{N_1} \otimes D_{N_2}(\alpha), \mathcal{F}_{N_1} \otimes \mathcal{F}_{N_2}(\alpha), f_{N_1} \otimes f_{N_2}(\alpha))$$

where $f_{N_1} \otimes f_{N_2}(\alpha)$ is defined to be the unique extension of $\mu_{N_1} \otimes \mu_{N_2}(\alpha)$ to $\mathcal{F}_{N_1} \otimes \mathcal{F}_{N_2}(\alpha)$.

Theorem 3. $I_{N_1} \otimes I_{N_2}$ *is a probabilistic component execution of* $N_1 \cup N_2$*.*

Proof sketch: This can be shown from definitions (22) and (23) and Corollary 1.

6 Denotational Representation of a Component with a Notion of Risk

For any disjoint set of interfaces N we define:

$$A_N \overset{\mathsf{def}}{=} \bigcup_{n \in N} A_n$$

$$cv_N \overset{\mathsf{def}}{=} \bigcup_{n \in N} cv_n$$

$$rf_N \overset{\mathsf{def}}{=} \bigcup_{n \in N} rf_n$$

The reason why we can take the union of functions with disjoint domains is that we understand a function as a set of *maplets*. A maplet is a pair of two elements corresponding to the argument and the result of a function. For example the following set of three maplets

$$\{(e_1 \mapsto f(e_1)), (e_2 \mapsto f(e_2)), (e_2 \mapsto f(e_2))\}$$

characterises the function $f \in \{e_1, e_2, e_3\} \to S$ uniquely. The arrow $\mapsto$ indicates that the function yields the element to the right when applied to the element to the left [4].

We define the semantic representation of a component analogous to that of an interface, except that we now have a set of interfaces N, instead of a single interface n:

Definition 10 (Semantics of a component). *A component is represented by a quadruple*

$$(I_N, A_N, cv_N, rf_N)$$

consisting of its probabilistic component execution, its assets, consequence function and risk function, as explained above.

We define composition of components formally as:

Definition 11 (Composition of components). *Given two components N_1 and N_2 such that $N_1 \cap N_2 = \emptyset$. We define their composition $N_1 \otimes N_2$ by*

$$(I_{N_1} \otimes I_{N_2}, A_{N_1} \cup A_{N_2}, cv_{N_1} \cup cv_{N_2}, rf_{N_1} \cup rf_{N_2})$$

7 Hiding

In this section we explain how to formally represent hiding in a denotational semantics with risk. As explained in Section 2.3 we must take care not to hide incidents that affect assets belonging to externally observable interfaces, when we hide internal interactions. An interface is externally observable if it interacts with interfaces in the environment. We define operators for hiding assets and interface names from a component name and from the semantic representation of a component. The operators are defined in such a way that partial hiding of internal interaction is allowed. Thus internal events that affect assets belonging to externally observable interfaces may remain observable after hiding. Note that hiding of assets and interface names is optional. The operators defined below simply makes it possible to hide e.g. all assets belonging to a certain interface n, as well as all events in an execution where n is the active party. We sketch the proof strategies for the lemmas and theorems in this section and refer to Brændeland et al. [2] for the full proofs.

Until now we have identified a component by the set of names of its interfaces. This has been possible because an interface is uniquely determined by its name, and the operator for composition is both associative and commutative. Hence, until now it has not mattered in which order the interfaces and resulting components have been composed. When we in the following introduce two hiding operators this becomes however an issue. For example, consider a component identified by $N \overset{\mathsf{def}}{=} \{c_1, c_2, c_3\}$. Then we need to distinguish the component $\delta c_2 : N$, obtained from N by hiding interface c_2, from the component $\{c_1, c_3\}$. To do that we build the hiding information into the name of a component obtained with the use of hiding operators. A component name is from now one either:

(a) a set of interface names,
(b) of the form $\delta n : N$ where N is a component name and n is an interface name,
(c) of the form $\sigma a : N$ where N is a component name and a is an asset, or
(d) of the form $N_1 +\!\!+ N_2$ where N_1 and N_2 are component names and at least one of N_1 or N_2 contains a hiding operator.

Since we now allow hiding operators in component names we need to take this into consideration when combining them. We define a new operator for combining two component names N_1 and N_2 as follows:

$$(27)\quad N_1 \uplus N_2 \overset{\mathsf{def}}{=} \begin{cases} N_1 \cup N_2 & \text{if neither } N_1 \text{ nor } N_2 \text{ contain hiding operators} \\ N_1 +\!\!+ N_2 & \text{otherwise} \end{cases}$$

By $\mathsf{in}(N)$ we denote the set of all hidden and not hidden interface names occurring in the component name N. We generalise definitions (18) to (21) to component names with hidden assets and interface names as follows:

$$(28)\quad \mathcal{E}_{\sigma a\,:\,N} \overset{\mathsf{def}}{=} \mathcal{E}_N \qquad \mathcal{E}_{\delta n\,:\,N} \overset{\mathsf{def}}{=} \mathcal{E}_{\mathsf{in}(N)\setminus\{n\}}$$

$$(29)\quad \mathcal{E}^{\downarrow}_{\sigma a\,:\,N} \overset{\mathsf{def}}{=} \mathcal{E}^{\downarrow}_N \qquad \mathcal{E}^{\downarrow}_{\delta n\,:\,N} \overset{\mathsf{def}}{=} \mathcal{E}^{\downarrow}_{\mathsf{in}(N)\setminus\{n\}}$$

$$(30)\quad \mathcal{H}_{\sigma a\,:\,N} \overset{\mathsf{def}}{=} \mathcal{H} \cap \mathcal{E}_{\sigma a\,:\,N}{}^{\omega} \qquad \mathcal{H}_{\delta n\,:\,N} \overset{\mathsf{def}}{=} \mathcal{H} \cap \mathcal{E}_{\delta n\,:\,N}{}^{\omega}$$

$$(31)\quad \mathcal{B}_{\sigma a\,:\,N} \overset{\mathsf{def}}{=} \mathcal{B}_N \qquad \mathcal{B}_{\delta n\,:\,N} \overset{\mathsf{def}}{=} ((\mathcal{E}_{\overline{\mathsf{in}(N)}} \setminus \mathcal{E}^{\downarrow}_n) \cup \mathcal{E}_{\mathsf{in}(N)}) \circledS \mathcal{B}_N$$

Definition 12 (Hiding of interface in a probabilistic component execution). *Given an interface name n and a probabilistic component execution I_N we define:*

$$\begin{aligned}
\delta n : I_N(\alpha) &\overset{\mathsf{def}}{=} (D_{\delta n\,:\,N}(\alpha), \mathcal{F}_{\delta n\,:\,N}(\alpha), f_{\delta n\,:\,N}(\alpha))\\
\textit{where}\quad D_{\delta n\,:\,N}(\alpha) &\overset{\mathsf{def}}{=} \{\mathcal{E}_{\delta n\,:\,N} \circledS t \mid t \in D_N(\delta n : \alpha)\}\\
\mathcal{F}_{\delta n\,:\,N}(\alpha) &\overset{\mathsf{def}}{=} \sigma(C_E(D_{\delta n\,:\,N}(\alpha)))\ \textit{i.e., the cone-}\sigma\textit{-field of } D_{\delta n\,:\,N}(\alpha)\\
f_{\delta n\,:\,N}(\alpha)(c) &\overset{\mathsf{def}}{=} f_N(\delta n : \alpha)\big(\{t \in D_N(\delta n : \alpha) \mid \mathcal{E}_{\delta n\,:\,N} \circledS t \in c\}\big)\\
\delta n : \alpha &\overset{\mathsf{def}}{=} \big((\mathcal{E}_{\overline{\mathsf{in}(N)}} \setminus \mathcal{E}_n^{\downarrow}) \cup \mathcal{E}_{\mathsf{in}(N)}\big) \circledS \alpha
\end{aligned}$$

When hiding an interface name n from a queue history α, as defined in the last line of Definition 12, we filter away the external input to n but keep all internal transmissions, including those sent to n. This is because we still need the information about the internal interactions involving the hidden interface to compute the probability of interactions it is involved in, after the interface is hidden from the outside.

Lemma 3. *If I_N is a probabilistic component execution and n is an interface name, then $\delta n : I_N$ is a probabilistic component execution.*

Proof sketch: We must show that: (1) $D_{\delta n\,:\,N}(\alpha)$ is a set of well-formed traces; (2) $\mathcal{F}_{\delta n:N}(\alpha)$ is the cone-σ-field of $D_{\delta n\,:\,N}(\alpha)$; and (3) $f_{\delta n\,:\,N}(\alpha)$ is a conditional probability measure on $\mathcal{F}_{\delta n\,:\,N}(\alpha)$. (1) If a trace is well-formed it remains well-formed after filtering away events with the hiding operator, since hiding interface names in a trace does not affect the ordering of events. The proof of (2) follows straightforwardly from Definition 12.

In order to show (3), we first show that $f_{\delta n\,:\,N}(\alpha)$ is a measure on $\mathcal{F}_{\delta n\,:\,N}(\alpha)$. In order to show this, we first show that the function $f_{\delta n\,:\,N}$ is well defined. I.e., for any $c \in \mathcal{F}_{\delta n\,:\,N}(\alpha)$ we show that $\big\{t \in D_N(\delta n : \alpha) \mid \mathcal{E}_{\delta n\,:\,N} \circledS t \in c\big\} \in \mathcal{F}_N(\delta n : \alpha)$. We then show that $f_N(\delta n : \alpha)(\emptyset) = 0$ and that $f_N(\delta n : \alpha)$ is σ-additive. Secondly, we show that $f_{\delta n\,:\,N}(\alpha)(D_{\delta n\,:\,N}(\alpha)) \leq 1$. When $f_{\delta n\,:\,N}(\alpha)$ is a measure on $\mathcal{F}_{\delta n\,:\,N}(\alpha)$ such that $f_{\delta n\,:\,N}(\alpha)(D_{\delta n\,:\,N}(\alpha)) \leq 1$ we can show that $f_{\delta n\,:\,N}(\alpha)$ is a conditional probability measure on $\mathcal{F}_{\delta n\,:\,N}(\alpha)$.

Definition 13 (Hiding of component asset). *Given an asset a and a component (I_N, A_N, cv_N, rf_N) we define:*

$$\begin{aligned}
\sigma a :(I_N, A_N, cv_N, rf_N) &\overset{\mathsf{def}}{=} (I_N, \sigma a : A_N, \sigma a : cv_N, \sigma a : rf_N)\\
\textit{where}\quad \sigma a : A_N &\overset{\mathsf{def}}{=} A_N \setminus \{a\}\\
\sigma a : cv_N &\overset{\mathsf{def}}{=} cv_N \setminus \{(e, a) \mapsto c \mid e \in \mathcal{E} \wedge c \in \mathbb{N}\}\\
\sigma a : rf_N &\overset{\mathsf{def}}{=} rf_N \setminus \{(c, p, a) \mapsto r \mid c, r \in \mathbb{N} \wedge p \in [0, 1]\}
\end{aligned}$$

As explained in Section 6 we see a function as a set of maplets. Hence, the consequence and risk function of a component with asset a hidden is the set-difference between the original functions and the set of maplets that has a as one of the parameters of its first element.

Theorem 4. *If N is a component and a is an asset, then $\sigma a : N$ is a component.*

Proof sketch: This can be shown from Definition 13 and Definition 10.

We generalise the operators for hiding interface names and assets to the hiding of sets of interface names and sets of assets in the obvious manner.

Definition 14 (Hiding of component interface). *Given an interface name n and a component (I_N, A_N, cv_N, rf_N) we define:*

$$\delta n :(I_N, A_N, cv_N, rf_N) \stackrel{\mathsf{def}}{=} (\delta n : I_N, \sigma A_n : A_N, \sigma A_n : cv_N, \sigma A_n : rf_N)$$

Theorem 5. *If N is a component and n is an interface name, then $\delta n : N$ is a component.*

Proof sketch: This can be show from Lemma 3 and Theorem 4.

Since, as we have shown above, components are closed under hiding of assets and interface names, the operators for composition of components, defined in Section 5, are not affected by the introduction of hiding operators. We impose the restriction that two components can only be composed by $\otimes$ if their sets of interface names are disjoint, independent of whether they are hidden or not.

8 Related Work

There are a number of proposals to integrate security requirements into the requirements specification, such as SecureUML and UMLsec. SecureUML [20] is a method for modelling access control policies and their integration into model-driven software development. SecureUML is based on role-based access control and specifies security requirements for well-behaved applications in predictable environments. UMLsec [15] is an extension to UML that enables the modelling of security-related features such as confidentiality and access control. Neither of these two approaches have particular focus on component-oriented specification. Khan and Han [17] characterise security properties of composite systems, based on a security characterisation framework for basic components [16]. They define a *compositional security contract* CsC for two components, which is based on the compatibility between their required and ensured security properties. This approach has been designed to capture security properties, while our focus is on integrating risks into the semantic representation of components.

Our idea to use queue histories to resolve the external nondeterminism of probabilistic components is inspired by the use of schedulers, also known as adversaries, which is a common way to resolve external nondeterminism in reactive systems [7,27,6]. Segala and Lynch [27,26] use a randomised scheduler to model input from an external environment and resolve the nondeterminism of a probabilistic I/O automaton. They define a probability space for each probabilistic execution of an automaton, given a scheduler. Seidel uses a similar approach in an extension of CSP with probabilities [28], where a process is represented by a conditional probability measure that, given a trace produced by the environment, returns a probability distribution over traces. Sere and Troubitsyna [29] handle external nondeterminism by treating the assignment of probabilities of alternative choices as a refinement step. Alfaro et al. [6] present a probabilistic model for variable-based systems with trace semantics similar to that of Segala and Lynch. Unlike the model of Segala and Lynch, theirs allows multiple schedulers to resolve the nondeterminism of each component. This is done to achieve deep compositionality, where the semantics of a composite system can be obtained from the semantics of its constituents.

Our decision to use a cone-based probability space to represent probabilistic systems is inspired by Segala [26] and Refsdal et al. [24,23]. Segala uses probability spaces whose σ-fields are cone-σ-fields to represent fully probabilistic automata, that is, automata with probabilistic choice but without non-determinism. In pSTAIRS [23] the ideas of Segala is applied to the trace-based semantics of STAIRS [12,25]. A probabilistic system is represented as a probability space where the σ-field is generated from a set of cones of traces describing component interactions. In pSTAIRS all choices are global. The different types of choices may only be specified for closed systems, and there is no nondeterminism stemming from external input. Since we wish to represent the behaviour of a component independently of its environment we cannot use global choice operators of the type used in pSTAIRS.

9 Conclusion

We have presented a component model that integrates component risks as part of the component behaviour. The component model is meant to serve as a formal basis for component-based risk analysis. To ensure modularity of our component model we represent a stakeholder by the component interface, and identify assets on behalf of component interfaces. Thus we avoid referring to concepts that are external to a component in the component model

In order to model the probabilistic aspect of risk, we represent the behaviour of a component by a probability distribution over traces. We use queue histories to resolve both internal and external non-determinism. The semantics of a component is the set of probability spaces given all possible queue histories of the component. We define composition in a fully compositional manner: The semantics of a composite component is completely determined by the semantics of its constituents. Since we integrate the notion of risk into component behaviour, we obtain the risks of a composite component by composing the behavioural representations of its sub-components.

The component model provides a foundation for component-based risk analysis, by conveying how risks manifests themselves in an underlying component implementation. By component-based risk analysis we mean that risks are identified, analysed and documented at the component level, and that risk analysis results are composable. Our semantic model is not tied to any specific syntax or specification technique. At this point we have no compliance operator to check whether a given component implementation complies with a component specification. In order to be able to check that a component implementation fulfils a requirement to protection specification we would like to define a compliance relation between specifications in STAIRS, or another suitable specification language, and components represented in our semantic model.

We believe that a method for component-based risk analysis will facilitate the integration of risk analysis into component-based development, and thereby make it easier to predict the effects on component risks caused by upgrading or substituting sub-parts.

Acknowledgements. The research presented in this paper has been partly funded by the Research Council of Norway through the research projects COMA 160317 (Component-oriented model-based security analysis) and DIGIT 180052/S10 (Digital interoperability with trust), and partly by the EU 7th Research Framework Programme through the Network of Excellence on Engineering Secure Future Internet Software Services and Systems (NESSoS).

We would like to thank Bjarte M. Østvold for creating lpchk:, which is a proof analyser for proofs written in Lamport's style [19], which is used in the full proofs of all the results presented in this paper, to be found in the technical report [2]. The proof analyser lpchk checks consistency of step labelling and performs parentheses matching, and also for proof reading and useful comments.

References

1. Ahrens, F.: Why it's so hard for Toyota to find out what's wrong. The Washington Post (March 2010)
2. Brændeland, G., Refsdal, A., Stølen, K.: A denotational model for component-based risk analysis. Technical Report 363, University of Oslo, Department of Informatics (2011)
3. Brændeland, G., Stølen, K.: Using model-driven risk analysis in component-based development. In: Dependability and Computer Engineering: Concepts for Software-Intensive Systems. IGI Global (2011)
4. Broy, M., Stølen, K.: Specification and development of interactive systems – Focus on streams, interfaces and refinement. Monographs in computer science. Springer (2001)
5. Courant, R., Robbins, H.: What Is Mathematics? An Elementary Approach to Ideas and Methods. Oxford University Press (1996)
6. de Alfaro, L., Henzinger, T.A., Jhala, R.: Compositional Methods for Probabilistic Systems. In: Larsen, K.G., Nielsen, M. (eds.) CONCUR 2001. LNCS, vol. 2154, pp. 351–365. Springer, Heidelberg (2001)
7. Derman, C.: Finite state Markovian decision process. Mathematics in science and engineering, vol. 67. Academic Press (1970)
8. Dudley, R.M.: Real analysis and probability. Cambridge studies in advanced mathematics, Cambridge (2002)
9. Probability theory. Encyclopædia Britannica Online (2009)
10. Folland, G.B.: Real Analysis: Modern Techniques and Their Applications. Pure and Applied Mathematics, 2nd edn. John Wiley and Sons Ltd., USA (1999)
11. Halmos, P.R.: Measure Theory. Springer (1950)
12. Haugen, Ø., Husa, K.E., Runde, R.K., Stølen, K.: STAIRS towards formal design with sequence diagrams. Software and System Modeling 4(4), 355–357 (2005)

13. He, J., Josephs, M., Hoare, C.A.R.: A theory of synchrony and asynchrony. In: IFIP WG 2.2/2.3 Working Conference on Programming Concepts and Methods, pp. 459–478. North Holland (1990)
14. ISO. Risk management – Vocabulary, ISO Guide 73:2009 (2009)
15. Jürjens, J. (ed.): Secure systems development with UML. Springer (2005)
16. Khan, K.M., Han, J.: Composing security-aware software. IEEE Software 19(1), 34–41 (2002)
17. Khan, K.M., Han, J.: Deriving systems level security properties of component based composite systems. In: Australian Software Engineering Conference, pp. 334–343 (2005)
18. Komjáth, P., Totik, V.: Problems and theorems in classical set theory. Problem books in mathematics. Springer (2006)
19. Lamport, L.: How to write a proof. American Mathematical Monthly 102(7), 600–608 (1993)
20. Lodderstedt, T., Basin, D., Doser, J.: SecureUML: A UML-Based Modeling Language for Model-Driven Security. In: Jézéquel, J.-M., Hussmann, H., Cook, S. (eds.) UML 2002. LNCS, vol. 2460, pp. 426–441. Springer, Heidelberg (2002)
21. Meyn, S.: Control Techniques for Complex Networks. Cambridge University Press (2007)
22. OMG. Unified Modeling Language^{TM} (OMG UML), Superstructure, Version 2.3 (2010)
23. Refsdal, A.: Specifying Computer Systems with Probabilistic Sequence Diagrams. PhD thesis, Faculty of Mathematics and Natural Sciences, University of Oslo (2008)
24. Refsdal, A., Runde, R.K., Stølen, K.: Underspecification, Inherent Nondeterminism and Probability in Sequence Diagrams. In: Gorrieri, R., Wehrheim, H. (eds.) FMOODS 2006. LNCS, vol. 4037, pp. 138–155. Springer, Heidelberg (2006)
25. Runde, R.K., Haugen, Ø., Stølen, K.: The Pragmatics of STAIRS. In: de Boer, F.S., Bonsangue, M.M., Graf, S., de Roever, W.-P. (eds.) FMCO 2005. LNCS, vol. 4111, pp. 88–114. Springer, Heidelberg (2006)
26. Segala, R.: Modeling and Verification of Randomized Distributed Real-Time Systems. PhD thesis, Laboratory for Computer Science, Massachusetts Institute of Technology (1995)
27. Segala, R., Lynch, N.A.: Probabilistic simulations for probabilistic processes. Nordic Journal of Computing 2(2), 250–273 (1995)
28. Seidel, K.: Probabilistic communicationg processes. Theoretical Computer Science 152(2), 219–249 (1995)
29. Sere, K., Troubitsyna, E.: Probabilities in action system. In: Proceedings of the 8th Nordic Workshop on Programming Theory (1996)
30. Skorokhod, A.V.: Basic principles and application of probability theory. Springer (2005)
31. Standards Australia, Standards New Zealand. Australian/New Zealand Standard. Risk Management, AS/NZS 4360:2004 (2004)
32. Weisstein, E.W.: CRC Concise Encyclopedia of Mathematics, 2nd edn. Chapman & Hall/CRC (2002)