Institut für Informatik der
Technischen Universität München

A Probabilistic Theory of Interactive Systems

Philipp R. Neubeck

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Die Dissertation wurde am 20.06.2012 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 18.10.2012 angenommen.

Abstract

The aim of this dissertation is to provide foundations for the specification and development of interactive systems in consideration of probabilistic effects. We develop a theory of probabilistic and interactive systems based on probability theory and the FOCUS theory for interactive systems. The notions of composition, specification and properties thereof like realizability are adapted to the probabilistic case. We discuss the interaction of nondeterministic and probabilistic systems as well as provide the framework for their integration. The thesis finishes with the formalization of various description techniques and of activities during the development of software systems using probabilistic models.

Contents

Chapter 1

Introduction

The complexity of software and embedded systems is ever growing. Traditional development approaches struggle to yield a working and failure-free end product. Current methodologies address the complexity with model-based development and component architectures. The formal and local specification of each component prevents failures in the whole system. In order to allow for formal descriptions, a formal theory has to provide the foundation of the methodology. It provides the semantics of the components and their compositions. One advanced and well-established theory of this kind is FOCUS [Bro10].

FOCUS and many other established theories mainly deal with deterministic and nondeterministic behavior, but especially embedded systems have several sources of probabilistic behavior, like physical transmission media, the random behavior of the user or randomized algorithms. Certain non-functional requirements like availability are actually probabilistic properties. That these theories do not consider probabilistic behavior is a shortcoming which we will address in this thesis. Our aim is to provide a fundamental and formal theory for the development of systems containing probabilistic elements. We follow the basic principles of FOCUS: systems are structured as hierarchical networks of interacting components, the component's interface abstracts from the component's internal state, et cetera.

1.1 Probabilistic Systems

A common understanding of probability is required for our later discussion. Therefore we introduce the concept of an experiment and on top of that we give different interpretations of probability.

To systematically observe a system, we bring the system into an initial state, activate the system's execution and finally observe some result (or outcome) of the system. These steps form an **experiment**. We can repeat the same experiment several times and note down each result. It is important to establish the very same environment, which may also provide some input for the system, and system state for each experiment. An **event** is then a set of results and occurs if the system produces one of these results.

Purely Probabilistic Systems At first, we assume that we were indeed able to cleanly reset the whole system state and identically repeat all influence on the system during each experiment. In the following discussion, we consider the sequence of experiment outcomes at hand.

The only variation in the outcomes can be caused by some probabilistic/random elements. That means, there is a degree of uncertainty as to which result will be produced in an experiment. This degree of uncertainty is exactly what the notion of probability tries to measure. Several definitions of probability exist, each trying to capture a certain insight into probability [Háj10]. We will now describe two informal definitions of probability that try to relate real world observations with the logical idea of probability. Both definitions have in common that they try to define a measure of the certainty or likelihood of results.

The *classical interpretation*, which was especially pushed by Laplace, requires results for which we have equally balanced evidence of their occurrence. Such evidence may be given by some symmetry of the results. We can then say, that these results are equally possible. The probability of an arbitrary event is then determined by the fraction of the number of results the event consists of. The typical example for this interpretation is the fair coin (or die) tossing. In the latter case, we can equally argue for each side of the die to occur and we obtain the set $\Omega = \{1, 2, 3, 4, 5, 6\}$ of six equally possible outcomes. Each of these outcomes has a probability of $^1/_6$. The event "occurrence of an even number when the die is rolled" consists of three elements and therefore has a probability of $^3/_6 = {^1/_2}$. This definition is problematic to apply in the real world because of phrases like "balanced evidence" and "equally possible".

The *frequency interpretation* considers the relative frequency of an outcome, i.e. the ratio of occurrences of this outcome to the total number of experiments. We assume now, that this relative frequency converges against a fixed value. This means, that for a very large number of experiments the relative frequency is nearly constant. This value defines the probability of this outcome. In mathematics this definition is easily realizable using the concept of "limit". In practice, we can always only observe a finite sequence and therefore can only guess against which value the sequence converges. For instance, with a sequence of coin tosses beginning with $(H, T, T, H, T, H, T, T, H, \ldots)$, we obtain the following relative frequencies of head

$$(1, {^1/_2}, {^1/_3}, {^2/_4}, {^2/_5}, {^3/_6}, {^3/_7}, {^3/_8}, {^4/_9}, \ldots) \quad .$$

In contrast to these definitions, which are informal but related to practical observations, mathematicians and statisticians use a formal *axiomatic definition of probability.* These axioms describe in particular what constitutes a probability space and therefor a (probabilistic) measure of events and they allow the rigorous logical deduction of theorems. But, this axiomatic definition does not explain what the logical notion of probability means in the real world. Hence, it cannot replace more philosophical definitions like the ones mentioned above. In Chapter 3.2, we will introduce such a formal and axiomatic definition that allows us to deduce statements about probabilistic systems.

Based on the notion of probability, we can point out the special case of probability one. If one outcome has a probability of one, then for any rerun of the experiment the system will produce this outcome. We call such systems **deterministic**.

The problem remains, how to determine the probability distribution of the system's results or, if we a priori assume certain probabilities, how to verify this hypothesis. Therefore statisticians provide a comprehensive set of statistical tests [SSS08]. One kind of such tests is the statistical hypothesis testing. There we usually assume a hypothesis (also called *null-hypothesis*), then we decide on the test and an appropriate decision rule (usually based on a level of significance), run the experiments and finally accept or reject the hypothesis according to the decision rule. Now the four combinations of "the hypothesis is true" or "the hypothesis is false" and "the hypothesis is accepted" or "the hypothesis is rejected" can occur, two of which are erroneous. Usually the decision rule is chosen such that the probability of the error "the hypothesis is true but rejected" is below some threshold (e.g., 1%). This ensures that we can be rather confident when rejecting the hypothesis.

Other statistical methods allow us to approximate some parameters of a probabilistic system. Assume for example that the observed system produces numerical results. Then the law of large numbers, a well-known theorem of statistics, states that it is very likely that the average value of a large number of observed results is near the expected value[1]. For an increasing number of results the observed average value tends towards the expected value with probability one. From this fact we can derive that we can approximate the expected value with the observed average value. Similarly it is possible to approximate the probability of each result by its observed relative frequency.

[1]The expected value describes the "long-run average value" and for discrete results is given by the weighted-sum of the possible results where the weight is given by each probability. E.g., when tossing a die we can interpret each side of the die numerically. So we have the possible results $\{1, 2, \ldots, 6\}$ each of probability $1/6$. Then the expected value is $\frac{1}{6} + \frac{2}{6} + \ldots + \frac{6}{6} = 3.5$.

Not purely Prob. Systems Until now we assumed that we were able to bring the system into an unbiased state before each experiment. We will now consider the opposite case. This divergence may lead to various non-probabilistic behaviors.

Because we did not reset the systems state correctly, the system may carry over some information from experiment to experiment. The system's output depends on the system's state and can differ in each experiment. Or similarly, we may not be able to provide the system with the same environment for each experiment.

In contrast to a purely probabilistic system, such systems can produce regular patterns. These patterns can be produced deterministically, i.e., each result depends uniquely on the results of the previous experiments, or can additionally depend on some probabilistic elements. If the system is very complex, then it may be difficult to distinguish such a system from a purely probabilistic system, imagine for example pseudo-random number generators imitating random behavior.

Another category of systems is known for its very unstable behavior. They are called *chaotic*, because even the smallest influence on these systems has a large effect on its output. A well-known example for such systems is the double pendulum. In the short term these systems can behave very well-ordered and predictable. However, it is physically impossible to repeatedly reset such systems into the very same state and to seal them off completely from their environment. Even the faintest influence increases to an immense change in the long term. As we cannot observe or control the former, the latter seems arbitrary.

Criticism of Probability One may object, that given an arbitrary probabilistic system one has just to look so closely at the system's operation (or gather enough knowledge about its operation, and use a better simulation of the system) such that the operation is again understood and non-probabilistic and deterministic. The throw of a six-sided die for instance seems to be a probabilistic system resulting in six different results of equal probability $1/6$ but if we look at the detailed physics of a die then the random behavior vanishes and we could predict the results (with a lot of computer power). Similarly, we could (theoretically) analyze the exact causes for errors of a transmission medium or try to predict the exact way a ball will take in the game roulette. We know that this may overwhelm current computers but at least for dice throwing and roulette we know all of the relevant physics – and also know that the smallest divergence in the initial conditions or computation errors lead to widely diverging results, i.e., the systems yield deterministic but chaotic behavior. In modern physics however, quantum mechanics and radioactive decay are supposed to be truly probabilistic. To date, we have no deterministic explanation of these events.

Summary We can summarize, that probability is a mathematical concept to describe unpredictable experiments which follow certain statistical patterns and we use probabilistic models as a simplification of complex systems (abstraction of a die, as described above).

According to the above classification of systems, we can also deduce criteria for when a system is not probabilistic or, more correctly, a probabilistic modeling is not appropriate.

In any case, if we model a system probabilistically, we have to argue about its appropriateness. From the frequentism point of view, we gather statistical patterns by running experiments and then derive predictions for a particular system. So we have to justify that this particular system and its environment are similar to the set-up of the experiment. Depending on how much information we have about the system our prediction will vary significantly. If we follow the classical interpretation according to Laplace, we have to argue about symmetries and independencies which never occur in this idealized form in practice. Thinking of the die tossing, we may assume a certain symmetry of the die, however, the starting position and the direction and the velocity of the throw may break this symmetry.

1.2 Systems Engineering

Separation of Concerns A major effort during engineering is the understanding of a system – no matter if the system is to be developed or parts of it exist already. Therefore, different aspects are usually described separately using different kinds of models. Both this separation of concerns as well as the specialization of each model increase the understanding of the system. A model should also be accompanied by rules or, more general, a theory such that meaningful properties can be derived or transformations can be applied. According to this principle, a probabilistic model enables us to comprehend statistical aspects of a system.

Stochastic Nature of Certain Requirements A key activity of engineering is balancing opposing requirements [Wie03]. Instead of aiming for a fault-free system, which would be too expensive and time-consuming to develop, we have to find an economical compromise between development costs and dependability. Other tradeoffs to be dealt with are usually between dependability and performance or maintainability.

This explains the necessity to consider dependability during system development. In turn, a common way to model availability and reliability as part of dependability is stochastically. For details see, e.g., [Hos60], [Bir10] or [ALRL04]. The latter argues:

"The extent to which a system possesses the attributes of dependability and security should be considered in a relative, probabilistic sense, and not in an absolute, deterministic sense: Due to the unavoidable presence or occurrence of faults, systems are never totally available, reliable, safe, or secure."

Although dependability is the most popular example for a requirement that can be modeled stochastically, this is also the case for other kinds of requirements. For example, performance metrics of queuing networks are traditionally described stochastically [Hap08].

Dealing with Existing Stochastic Systems Intensive interaction with its environment is characteristic for embedded systems. We understand as part of the environment both physical hardware but also other (software) systems that the system under design interacts with. Requirements constrain the system under the assumption of certain properties of its environment. This has two implications for the system engineering: we have both to express properties of the environment, i.e., of existing systems, but also have to respect these assumptions during the development.

In particular for environments of embedded systems, properties are in many cases only available as statistical information – maybe it is the only provided information or other descriptions are just not practical.

Consider for example, the development of a car radio. In order to obtain a high dependability, we can concentrate on the dependability of the frequently used and critical functions. The "weighted sum" of the dependability of each function has to be sufficient.

Example In the next simple example, we can address every concept mentioned in the previous paragraphs. Consider the development of a transmission protocol on top of an unreliable electrical transmission line. For simplicity assume that we have to send one datum of a fixed size with a certain maximal duration. On the one hand, a short transmission duration to send the whole datum is preferred. On the other hand, this is counteracted by the errors during transmission. Thus, we have both a constraint on the system (the successful transmission) but also an assumption about the environment (the unreliability of the transmission line).

At this point, we have to make clear how to describe the unreliability of the transmission line. Usually, but not necessarily required, we abstract from the complex electromagnetic effects of the transmission line to a simple probabilistic model. Say, this model explains bit-errors to occur independently and according to a non-zero single-error probability. In that case, it is impossible to guarantee a successful transmission within a bounded duration because with a non-zero probability every attempt to send a bit will fail. This probability will be lower the more often we attempt and the longer the transmission duration. Thus, we have to trade-off the transmission duration against the transmission reliability.

Utilizing Randomness in the Implementation It is also common to deliberately utilize probabilistic elements in certain systems. Most common are randomized algorithms, which integrate random number generators to improve the overall (time or space) complexity or to be robust against bad inputs. In the case of distributed systems, random generators are one way to break the symmetry in distributed algorithms like wireless protocols [Lyn96].

For instance in cryptographic applications, we need to generate random keys. Some cryptographic systems, like RSA, utilize prime numbers for the key generation. Therefore a randomly generated number is tested to be prime. Some very efficient and common primality testing algorithms are randomized ones. These algorithms' result is not guaranteed to be correct but the error probability can be reduced arbitrarily (but not completely eliminated). It is possible to reduce this probability in n iterations to a value as low as $1/4^n$ [Blu01]. A larger class of such algorithms is called Monte Carlo algorithms and in many cases these algorithms are more efficient than their non-probabilistic counterparts.

That means that we construct probabilistic systems on purpose. Already, modern computer systems heavily depend on the existence of good random number generators to run cryptographic applications or probabilistic algorithms which have to be as unpredictable as possible. Pseudo random number generators are deterministic algorithms which mimic (emulate) probabilistic behavior. Such generators are widespread but hardware random number generators produce better (i.e., less predictable and more uniformly distributed) results. Such hardware depends on physical processes like thermal noise or quantum effects.

Implications for the System Development Large systems have to be developed in a structured way. Hierarchies of components are a typical structuring concept. If we introduce probabilistic components into our system, then we have to understand the effect of their composition, also with non-probabilistic components. How can the random behavior propagate through the system? Can the combination of several probabilistic components in a system add up to some unforeseen behavior?

Finally, a systems engineer will be interested in the placement of probabilistic modeling within a development methodology. We have to know the phases at which probabilistic elements can be incorporated. What are the differences and consequences of probabilistic modeling compared to the deterministic and nondeterministic non-probabilistic variants?

We conclude this introduction with a short summary of points of interest:

- A system has to hide random sources of error and provide its functionality to the user with a practically high likelihood. In many cases, it is impractical or even theoretically impossible to completely eliminate the probability of error. We need the means to check the remaining randomness exposed by the system.
- Absolute specification conformance is not always required or possible, but conformance with high probability. This can reduce costs and lead to more efficient systems.
- Systems can incorporate random elements on purpose, again with the possible benefit of simplification and efficiency.
- To build large systems with probabilistic elements, we have to understand the probabilistic behavior of a single component but also of a network of components.
- We are interested in the effect of incorporating probabilistic models into the development process.

1.3 Outline

The subsequent Chapter 2 gives an introduction to existing works dealing with probabilistic systems that are relevant to the systems engineering. We take a look at some existing modeling theories, logics and verification techniques. In Chapter 3, we provide the basic definitions used throughout this thesis. This includes a brief extract of probability theory.

The contribution of this thesis is presented starting with Chapter 4. This chapter presents probabilistic components as the counterpart of deterministic non-probabilistic components of Focus. Chapter 5 provides different description techniques, which are also used for the examples in this thesis. Chapter 6 extends probabilistic components about nondeterminism. Based on this introduced system model, we workout the concept of system refinement and explain the relation to development steps in Chapter 7.

Finally, Chapter 8 summarizes the thesis and gives an outlook on future research directions.

Chapter 2

Related Work

In this chapter, we will review different topics relevant for the development of systems with probabilistic behavior: probabilistic models, logics and verification. For the interested reader, this should serve as a starting point for further studies.

2.1 Probabilistic Models and Specifications

Several approaches to represent and specify probabilistic systems exist. However, only a fraction focuses on compositionality. To our best knowledge, all of these approaches assume statistical independence between components during composition which clearly differs from the proposed model theory presented in this thesis. Furthermore opposed to these other works, we propose a framework not primary for a particular analysis or specification technique but as a common foundation for a catalog of different system engineering activities and nomenclature.

Besides the approaches presented in the following, extensions to the process algebras CCS [HJ90, GcJS90, YL92] and CSP [MMSS96] exist, of which [RS11] considers refinement of probabilistic processes with testing semantics.

2.1.1 Interactive Markov Chains

In [Her02], H. Hermanns addresses the compositional specification and analysis of Markov chains. He introduces *interactive Markov chains (IMCs)* as a combination of *interactive processes* and *continuous-time Markov chains*. Interactive processes are represented as transition systems labeled with *external* and *internal actions*. A continuous-time Markov chain is a Markov chain with discrete state space and continuous time range. The only memory-less continuous distributions are the class of exponential distributions. Therefore, continuous-time Markov chains are represented as transition systems labeled with transition rates, where a *rate* determines a unique exponential distribution. IMCs allow transitions of both kinds, each either labeled with an action or a transition rate.

Based on the definition of IMCs, H. Hermanns defines the hiding of actions, also called abstraction, and the composition of IMCs. The composed IMCs are synchronized according to their actions and the time delays determined by the transition rates. Thereby, actions are executed instantaneously.

[Her02] covers also two notions of *similarity* between IMCs, namely strong and weak bisimilarity. These are shown to be congruencies with respect to the composition of IMCs. Algorithms to decide bisimilarity and to minimize IMCs are provided. Due to interleaved minimization steps, the provided methodology allows the analysis of systems with several million states, exemplarily performed for a telephone system.

2.1.2 Compositional Methods for Probabilistic Systems

In [dAHJ01], a deterministic system's behavior is represented as a probability distribution on finite traces, a so called *bundle*. The semantics of a nondeterministic system is a set of bundles.

Given a description of a nondeterministic system, a *scheduler* resolves each nondeterministic choice and produces a set of purely probabilistic bundles. A *deterministic* scheduler resolves every choice with a deterministic behavior, whereas a *randomized* scheduler replaces the choice by a probabilistic distribution. Using randomized schedulers, nondeterminism allows for unspecified probability.

A system's state is purely variable-based. In each step, several variable assignments occur synchronously and may depend on the previous values. Similar to *reactive modules*, the variables are partitioned into *private*, *output* and *input* variables. Private variables are only visible and modifiable by the owning module. Schedulers have to respect this partition, because the assignment to external variables, i.e., variables owned by other modules, must be statistically independent of the assignment to private variables.

This required independence leads to the notion of *atoms*. An atom is a part of a component's syntactic description. Each atom consists of a set of controlled variables and a set of read variables. A composite system obtains the union of the subsystems' sets of atoms. Thus in a composite system, several atoms describe a partitioning of the variables into independent parts. For each atom a scheduler resolves the nondeterministic choice of its controlled variables depending only on the read variables, and an additional scheduler resolves the choices of the environment, i.e., the values of all remaining external variables.

The semantics of composition is given by the intersection of the operands' bundle sets. Similarly, refinement corresponds to set inclusion. This choice of composition and refinement allows for the *assume-guarantee* rule.

2.1.3 Compositional Design Methodology with Constraint Markov Chains

Compared to Markov Chains, Interval Markov Chains [CDL$^+$09] allow constraining transitions not only with a fixed probability but with intervals of probabilities. Simple examples show that these are neither closed under logical conjunction nor under parallel composition. Let x be the probability for a transition, then an interval constraint $x \in [a, b]$ is equivalent to the inequalities $a \leq x \wedge x \leq b$. The paper [CDL$^+$09] shows that the generalization to linear constraints suffices for closure under conjunction. Constraint Markov Chains (CMCs) allow polynomial constraints which are required and sufficient for closure under parallel composition.

Besides composition, a refinement relation is introduced. For the subset of deterministic CMCs, this refinement coincides with the implementation set inclusion, i.e., a deterministic CMC A specificies a subset of the systems specified by the deterministic CMC B, if and only if A is a refinement of B.

2.2 Probabilistic Logics

We distinguish qualitative and quantitative system properties [Kwi07]. The former are properties that refer only to the extreme probabilities zero and one – does an event occur almost surely or almost never? Quantitative properties are not bound to this restriction and refer to arbitrary probabilities and occur in two variants: as a query asking for a certain probability (or value) and as a predicate with a fixed boundary. Verification of qualitative properties depends in certain cases only on the topology of the underlying transition system [HSP83] whereas quantitative properties depend on the concrete transition probabilities.

For example, we may ask for the probability to reach an error states or ask if this probability is greater than 0.9. In both cases we speak of a qualitative property. The property that an error state is reached with probability one, i.e., almost surely, is a qualitative one.

In a probabilistic system model, every set of execution paths has a certain probability. From a predicate Q over states, we can derive the set of paths reaching a satisfying state and we can ask its probability, which is the reachability probability of Q. Every LTL formula defines also a set of paths; if this set is measurable we can ask for its probability. Similarly, we obtain probabilistic properties from CTL formulas.

In the literature, we find several definitions of probabilistic time logics [SL94, BK08] like PCTL, PCTL*, WPCTL and PCRTL (PCTL with rewards), which allow the compact definition of properties of dynamic systems.

PCTL The logic PCTL, for example, is defined similar to CTL, but instead of universal and existential path quantification, a probabilistic operator $\mathbf{P}_J(\varphi)$ is defined where φ is a path formula and J is an interval of $[0,1]$. The meaning of the formula $\mathbf{P}_J(\varphi)$ in a state s is: the probability of the set of paths starting in s and satisfying φ lies within the interval J.

For instance in the context of a communication protocol, the formula [BK08]

$$\mathbf{P}_{=1}(\Diamond\, \mathit{delivered}) \wedge \mathbf{P}_{=1}(\Box\, (\mathit{tryToSend} \rightarrow \mathbf{P}_{\geq 0.99}(\Diamond^{\leq 3} \mathit{delivered})))$$

specifies that almost surely a message will be delivered and every attempt to send a message succeeds within 3 steps with probability at least 0.99.

2.3 Verification of Probabilistic Systems

For the verification of software systems two approaches exist: we can apply a programming logic and prove the correctness of the system by logical deduction. Alternatively, we consider a computational model of the system and systematically verify the correctness by regarding every possible computation. The latter approach usually incorporates engineering techniques to automate the verification process. Such automated tools are called *Model Checker*.

Much effort was invested to adapt both approaches to probabilistic systems. Today we have both probabilistic programming logics and probabilistic model checker.

In this section, we will at first look into a probabilistic extension of Dijkstra's *guarded-command language* and programming logic of *weakest precondition*. Afterwards we review the probabilistic model checker PRISM and give a rough idea of the properties it is able to verify. A promising optimization technique for the model checking of large systems is the counterexample-guided abstraction refinement, the discussion of which will conclude this section.

2.3.1 A Programming Logic for *pGCL*

Dijkstra's guarded command language (*GCL*) is a programming language for the description of imperative sequential programs. A peculiarity is its containment of "demonic" nondeterminism to express the abstraction from a decision between alternative program branches. The nondeterministic choice is said to be demonic because it may be resolved in a way that harms the program's correctness.

The probabilistic extension *pGCL* of *GCL* introduces additionally a probabilistic choice [MM04]. This choice follows the laws of probability theory and could, for example, be decided by a coin flip. With this extension randomized algorithms can be developed.

Based on this extension, it was possible also to adapt the programming logic of *weakest precondition.* The programming logic for *pGCL* follows the principle of the *greatest pre-expectations.* An **expectation** is a generalization of predicates: instead of the two truth-values *true* and *false* it assigns each valuation a probability. A pre-expectation determines then, for each starting state, the probability that a certain program execution yields a state that fulfills the post-expectation. The greatest pre-expectation is the maximum over the probabilities of all possible executions.

According to this principle, the predicate transformer *wp* is extended to an expectation transformer. Just to give a concrete idea of such transformations, we highlight its main difference, namely the definition for nondeterministic and probabilistic choice (P and Q are program fragments, A is an expectation i.e., a distribution over variable assignments):

$$\begin{aligned} wp(\mathsf{P} \sqcap \mathsf{Q}).A &:= \min\{wp.\mathsf{P}.A, wp.\mathsf{Q}.A\} \\ wp(\mathsf{P}\ {}_p\!\oplus \mathsf{Q}).A &:= p \cdot wp.\mathsf{P}.A + (1-p) \cdot wp.\mathsf{Q}.A \quad . \end{aligned}$$

The first definition shows that only the minimum of the greatest pre-expectations of each branch is propagated through a nondeterministic choice. The second definition shows that the probabilistic choice behaves like the p-weighted average of its arguments.

With this foundation proofs of randomized algorithms can be developed by annotating the program code with the probabilistic analogue of Hoare triples.

Similar to the logical implication an order between expectations can be defined [MM05] which induces a refinement relation between probabilistic *pGCL* programs. Using refinement and the other way round abstraction, proofs can be developed in a modular fashion. It is interesting to note that the theory presented in [MM05] states that deterministic choice is a refinement of probabilistic choice, which in turn is a refinement of nondeterministic choice.

2.3.2 A Probabilistic Model Checker: PRISM

PRISM [KNP09, KNP12] is a model checker for three types of probabilistic models: discrete-time Markov chains, continuous-time Markov chains and Markov decision processes (Markov chains with nondeterminism). Additionally, extensions of these models with costs and rewards are also supported.

PRISM's property specification language (supporting temporal logics like PCTL and LTL) supports the following central operators:

- The **P** operator refers to the probability of an event occurring.
- With the **S** operator one can determine steady-state probabilities of a model.

- The **R** operator calculates expected values of rewards.

These operators can be combined with the well-known LTL operators. Let *operational* and *fail* denote the sets of operational respectively failure states of a system. The specification language then allows asking for quantitative properties like

- $\mathbf{P}_{=?}[\mathbf{F}^{[0,10]}\mathit{operational}]$ – the probability that the system is operational at least once within the first 10 time steps.
- $\mathbf{S}_{=?}[\mathit{operational}]$ – the long-run availability of the system, i.e., the steady-state probability that the system is operational.
- $\mathbf{R}_{\{\text{"}time\text{"}\}=?}[\mathbf{F}\mathit{fail}]$ – the mean-time-to-failure of the system, i.e., the expected time until the first failure occurs.
- What is the worst-case probability of the protocol terminating in error, over all possible initial configurations?
- What is the expected size of the message queue after 30 minutes?
- What is the worst-case expected time taken for the algorithm to terminate?

The specification language is accompanied by a textual modeling language based on the *Reactive Modules* formalism. Modules consist of local variables and guarded commands which can be labeled with actions for synchronization.

PRISM analyzes large models efficiently by using optimization techniques like symbolic calculation (based on Binary Decision Diagrams, BDDs).

2.3.3 Probabilistic CEGAR

Counterexample-guided abstraction refinement (CEGAR) is a technique for model checking of large systems. [HWZ08] considers the application of predicate abstraction to probabilistic systems. Predicate abstraction partitions the possibly infinite state space into a finite number of regions. As long as the property to check can neither be proved nor refuted, the abstraction is refined with additional predicates. These predicates are obtained from abstract counterexamples. So far, this approach is restricted to probabilistic reachability (cf. 2.3.2) and can determine if the probability to reach a set of bad states exceeds a given threshold.

The abstract automaton is a quotient automaton which preserves a fragment of PCTL. That means, if a property holds for the abstract automata, we can conclude that the property holds also for the original automata. The converse, however, is not true in general. We have to check if there exists a concretization of the abstract counterexample. But an abstract counterexample can be spurious, i.e., there exists no concrete counterpart. Then the abstract model has to be refined.

In classical CEGAR with Boolean logic, an abstract counterexample is a single finite path to some bad state. In the probabilistic setting, a set of abstract paths reaching a bad state is searched and the probability of this set must exceed the given threshold. This is the idea of the strongest evidence.

The input language is a guarded command language similar to the one of Prism (cf. Section 2.3.2) but supports infinite data domains.

Chapter 3

Foundations

3.1 Basic Definitions

Numbers We use $\mathbb{N} = \{1, 2, \ldots\}$ to denote the natural numbers excluding zero. A 0 in the index includes the zero, as does a ∞ mean to include the limit symbol ∞, which has the property:

$$\forall n \in \mathbb{N}_0 \colon n < \infty \land n + \infty = \infty \quad .$$

Functions Let f be a function of type $X \to Y$. The domain X and range Y is given by $\operatorname{dom} f$ and $\operatorname{rng} f$, respectively.

To avoid clusters of parentheses, we occasionally write $f.x$ instead of $f(x)$ for function application. We choose left associativity for this operator such that $f.x.y = (f.x).y$.

For finite $X = \{x_1, \ldots, x_n\}$, we may define f by

$$f = (x_1 \mapsto f.x_1, \ldots, x_n \mapsto f.x_n) \quad .$$

Note that the notation $(\,)$ for the unique function with empty domain $X = \emptyset$ and range Y coincides with the notation for the empty tuple.

Valuations Assume a set of variables (or variable identifiers) V and a typing function $t \colon V \to \mathcal{P}(\mathcal{U})$, where $\mathcal{U}$ is the universal set of values. A **valuation** of V is a mapping $\nu \colon V \to \mathcal{U}$ such that the values are in the range of each variable:

$$\forall v \in V \colon \nu(v) \in t(v) \quad .$$

Given a typing function t, we refer to the set of all valuations of the variables $\operatorname{dom} t$ by $\bar{t}$. If the typing is clear from the context, we may omit the typing function and instead write $\overline{V}$ for the set of valuations. Usually, we will assume a global typing function called **type**. Note the notation $\bar{\emptyset}$ for the set of valuations over the empty set of variables, which is the singleton containing only $()$, the unique function of type $\emptyset \to \mathcal{U}$.

The **restriction** of a valuation $\nu \in \overline{V}$ to a subset $V' \subseteq V$ is the valuation $\nu|_{V'} \in \overline{V'}$ defined by:

$$\forall v \in V' \colon \nu|_{V'}(v) = \nu(v) \quad .$$

Valuations $\nu \in \overline{V}$ and $\mu \in \overline{W}$ with not necessarily disjoint domains can be combined to the **union** $\nu \uplus \mu \in \overline{V \cup W}$ if their values on common variables coincide, that means $\nu|_{V \cap W} = \mu|_{V \cap W}$. This union is defined by

$$(\mu \uplus \nu).v = \begin{cases} \nu.x & \text{if } x \in V \\ \mu.x & \text{if } x \in W \end{cases} \quad .$$

Sequences We call a function $s \colon \{0, \ldots, n-1\} \to A$ a **sequence** (or stream or word) of length n over the alphabet A. A sequence s may be written as

$$\langle s.0, s.1, \ldots, s.(n-1) \rangle \quad .$$

An **infinite sequence** over alphabet A is a function $\mathbb{N}_0 \to A$. We write A^n, A^*, A^∞ and A^ω for the set of sequences of length n, finite sequences, infinite sequences and sequences of arbitrary length, respectively. Note the notation $\langle\,\rangle$ of the unique sequence of zero length. The length of a sequence s is obtained by $|s|$.

A finite sequence a can be **concatenated** with any other sequence b written as $a \cdot b$:

$$\begin{aligned} &\forall n \in \mathbb{N}_0, m \in \mathbb{N}_{0,\infty} \colon \\ &\quad a \in A^n \wedge b \in B^m \implies \\ &\quad (a \cdot b) \in A^{n+m} \wedge \forall i \leq n \colon (a \cdot b).i = a.i \wedge \forall i \leq m \colon (a \cdot b).(n+i) = b.i \quad . \end{aligned}$$

In the case $a \cdot b = c$, we call a a **prefix** of c and write $a \sqsubseteq c$ or $c \sqsupseteq a$. We call b a **postfix** of c. Let s be a sequence over A of length greater or equal to $n \in \mathbb{N}_0$ then $s{\downarrow_n}$ denotes the **prefix of length** n of s and $s{\uparrow^n}$ denotes the **postfix** beginning at the $(n+1)$-th element. Thus we have:

$$s{\downarrow_n} \in A^n \wedge s{\downarrow_n} \cdot s{\uparrow^n} = s \quad .$$

We apply the Cartesian product $a \in A, b \in B \mapsto (a, b) \in A \times B$ for arbitrary sets A and B also to sequences by element-wise application. Given the sequences $\varphi \in A^n$ and $\sigma \in B^n$ for some $n \in \mathbb{N}_{0,\infty}$, we denote with $\varphi \times \sigma$ the sequence of $(A \times B)^n$ such that $(\varphi \times \sigma).k = (\varphi.k, \sigma.k)$ for all $k \in [0, n]$. This function is a bijection from elements of $A^\infty \times B^\infty$ to $(A \times B)^\infty$.

Similarly, the union and restriction of valuations can be generalized to sequences of valuations by element-wise application. That means, we define for two sequences of valuations $\varphi \in \overline{V}^n$ and $\sigma \in \overline{W}^n$ with $n \in \mathbb{N}_{0,\infty}$ and all subsets $V' \subseteq V$:

$$\forall t \leq n\colon (\varphi \uplus \sigma).t = \varphi.t \uplus \sigma.t$$

as well as

$$\forall t \leq n\colon (\varphi|_{V'}).t = \varphi.t|_{V'} \quad .$$

Formulas and Predicates By **formula** we mean some evaluable term. The type of the arguments (i.e., free variables) and the value resulting from evaluation to be of some specified type. A special case are **predicates** which are Boolean valued formulas. Usually we will express these in propositional or predicate calculus. Other calculus may be used instead, too. We call non-Boolean valued formulas also **expressions**.

Formulas may be syntactically combined or modified to obtain new formulas.

Applying the underlying semantics[1], we can evaluate a formula by supplying values to its free variables. In that case, we simply interpret the formula as a function. Let f be a X-valued formula with free variables V, then we identify with $[\![f]\!]\colon \overline{V} \to X$ the function which maps a valuation $v \in \overline{V}$ to the evaluation of the formula f given the values v, which is a value in X.

3.2 Extract of Probability Theory

This section presents the definitions of probability theory relevant for this work. The definitions and statements are taken from [KSK76] and [GDG+01] while the proofs are omitted.

3.2.1 Measure Theory

Let Ω be an arbitrary set and $\mathcal{F}$ a family of subsets of Ω. We call $\mathcal{F}$ a **field** of sets if

1. $\emptyset \in \mathcal{F}$,
2. if A is a set in $\mathcal{F}$ then its complement $\overline{A}$ is in $\mathcal{F}$, and
3. if A and B are in $\mathcal{F}$ then their union $A \cup B$ is also in $\mathcal{F}$.

[1]We assume a given interpretation of the used predicates in the case of predicate logic.

We say that $\mathcal{F}$ is a **σ-field** or **σ-algebra**, if it is additionally closed under countable unions, that means for sets $A_1, A_2, \ldots \in \mathcal{F}$ it is $\bigcup_{i=1}^{\infty} A_i$. We usually denote σ-fields with the symbol $\mathcal{B}$.

A function μ from a field $\mathcal{F}$ of subsets of Ω to the real extended numbers $\mathbb{R} \cup \{-\infty, +\infty\}$ is called a **set function** and it is called a **measure** if

1. $\mu(A) \geq 0$ for all $A \in \mathcal{F}$ (Non-negativity),
2. $\mu(\bigcup_{i=1}^{\infty} A_i) = \sum_{i=1}^{\infty} \mu(A_i)$ for any pairwise disjoint sets $A_1, A_2, \ldots \in \mathcal{F}$, i.e., $A_i \cap A_j = \emptyset$ whenever $i \neq j$ (Countable Additivity).

The triple $(\Omega, \mathcal{B}, \mu)$ with a measure μ over the σ-field $\mathcal{B}$ is called a **measure space**. If the measure is additionally normalized, i.e., $\mu(\Omega) = 1$, we call it a **probability measure** and the triple is called **probability space**.

Let p be a predicate over the sample space, i.e., a function $\Omega \to \mathbb{B}$. Then the **probability** of p is defined as the measure of its truth set, i.e., $\mu(p) = \mu(\{\omega \in \Omega \mid p\})$. The probability is only defined if the truth set is measurable.

Given events $F, G \in \mathcal{B}$ with $\mu(F) > 0$ of a probability space $(\Omega, \mathcal{B}, \mu)$, we define the **conditional probability of G given F** by

$$\mu(G \mid F) = \frac{\mu(G \cap F)}{\mu(F)} \quad .$$

For a fixed F, the function $G \mapsto \mu(G \mid F)$ is itself a probability measure on a certain probability space over F.

For any set $\mathcal{G}$ of subsets of Ω there exists a smallest σ-field containing $\mathcal{G}$: There exists a containing σ-field, namely the power set $\mathcal{P}(\Omega)$, and the intersection of all σ-fields containing $\mathcal{G}$ is also a σ-field containing $\mathcal{G}$. This σ-field is the smallest one as it is contained in all others. We denote this σ-field by $\sigma(\mathcal{G})$.

Let $\mathcal{F}$ be a field (not necessarily a σ-field) over a space Ω and μ a probability measure. The *Carathéodory Extension Theorem* states that there exists a unique measure λ on $\sigma(\mathcal{F})$ that agrees with μ on $\mathcal{F}$, that is $\lambda(F) = \mu(F)$ for all $F \in \mathcal{F}$. This measure λ is given by:

$$\lambda(F) = \inf\Big\{\mu(\bigcup_i F_i) \,\Big|\, \bigcup_i F_i \supseteq F \wedge \forall i \colon F_i \in \mathcal{F}\Big\} \quad .$$

A field $\mathcal{F}$ has the **countable extension property** if it has a countable number of elements and if every non-negative, normalized and finitely additive set function μ on $\mathcal{F}$ is also countably additive on $\mathcal{F}$. Finite additivity of a set function μ on a field $\mathcal{F}$ means

$$\text{if } A_1, A_2, \ldots, A_n \in \mathcal{F} \text{ are disjoint, then } \mu(\bigcup_{i=1}^{n} A_i) = \sum_{i=1}^{n} \mu(A_i).$$

An **atom** of a field $\mathcal{F}$ is an element $F \in \mathcal{F}$ such that no other member of $\mathcal{F}$ except the empty set and the set itself are subsets of F.

A sequence of finite fields $\mathcal{F}_n, n \in \mathbb{N}$, is said to be a **basis** for a field $\mathcal{F}$, if the following properties hold:

1. $\mathcal{F}_n \subseteq \mathcal{F}_{n+1}$ for all $n \in \mathbb{N}$
2. $\mathcal{F} = \bigcup_{n=i}^{\infty} \mathcal{F}_n$
3. If G_n is a sequence of elements of $\mathcal{F}$ with G_n being an atom of $\mathcal{F}_n$ and $G_{n+1} \subseteq G_n$ for all $n \in \mathbb{N}$, then
$$\bigcap_{n=1}^{\infty} G_n \neq \emptyset \quad .$$

If such a basis exists, the field $\mathcal{F}$ is called **standard**. A σ-field $\mathcal{B}$ or the measurable space $(\Omega, \mathcal{B})$ is called **standard** if it is generated by a standard field $\mathcal{F}$, i.e., $\mathcal{B} = \sigma(\mathcal{F})$.

One such standard measurable space is $(\Omega, \mathcal{P}(\Omega))$, where Ω is countable and $\mathcal{P}(\Omega)$ is the power set of Ω, i.e., the set of all subsets of Ω. Furthermore the product space of a countable family of standard spaces is again standard.

From [GDG+01], we also cite the fundamental

Theorem 1 *A field has the countable extension property if and only if it is standard.* □

Sequence Spaces We wish to have a measure on subsets of Σ^∞, where Σ is some countable alphabet. That means, we consider the uncountable sample space $\Omega = \Sigma^\infty$. In [GDG+01], the product space $(\Sigma^\infty, \sigma(\mathcal{P}(\Sigma)^\infty))$ is defined, which is exactly what we seek. We will briefly outline the construction of this product space and show that *rectangles* can be replaced by *cylinders*.

The product Σ^n of the sample space Σ is the set of all sequences of length n, $n \in \mathbb{N} \cup \{\infty\}$. The **product field** $\mathcal{F}^n$, for $n \in \mathbb{N} \cup \{\infty\}$, of some field $\mathcal{F}$ over Ω however is defined to be the field generated by all finite dimensional **rectangles** R, which are the sets of the form:

$$R = \{\omega \in \Omega^n \mid \forall j \in J \colon \omega.j \in F_j\} \tag{3.1}$$

where $J \subseteq [1, n]$ is a finite set of coordinates and $F_j \in \mathcal{F}$ for all $j \in J$. Let $RECT(\mathcal{F}, n)$ be the set of all such rectangles, then we have formally $\mathcal{F}^n = field(RECT(\mathcal{F}, n))$.

Instead of considering rectangles, we can also confine to the class of cylinders. We say a rectangle is a k-th order **cylinder** if in its representation (3.1) above, it is $J = \{1, 2, \dots, k\}$. The set of all k-th order cylinders in $RECT(\mathcal{F}, n)$ is denoted by $CYL(\mathcal{F}, k, n)$.

It follows that

$$\begin{aligned} \forall n \in \mathbb{N} \colon RECT(\mathcal{F}, n) &= CYL(\mathcal{F}, n, n) \\ \wedge RECT(\mathcal{F}, \infty) &= \bigcup_{k \in \mathbb{N}} CYL(\mathcal{F}, k, \infty) \quad . \end{aligned}$$

Since every cylinder is a rectangle, the one inclusion in both equations obvious. In the case $n \in \mathbb{N}$, let R be some rectangle in $RECT(\mathcal{F}, n)$. There exist $J \subseteq [1, n]$ and $F_j \in \mathcal{F}$ fulfilling Equation (3.1). Then $J' = [1, n]$ and $F'_j = F_j$ for $j \in J$ and $F'_j = \Omega \in \mathcal{F}$ for $j \in [1, n] \setminus J$ fulfill Equation (3.1) as well and we see, that $R \in CYL(\mathcal{F}, n, n)$.

Let now be $n = \infty$ and R, J and F_j, $j \in J$ as in Equation (3.1). Then the equation holds also for $J' = \{1, 2, \dots, \max J\} \supseteq J$, $F'_j = F_j$ for $j \in J$ and the additional events $F'_j = \Omega$, for $j \in J' \setminus J$. This shows $R \in CYL(\mathcal{F}, \max J, \infty)$.

From the *Kolmogorov Extension Theorem* we derive the following corollary. In its formulation we use preimages of the projection $\Pi_{l \to k} \colon X^l \to X^k$, for $k < l \leq \infty$:

$$\forall x \in X^l \colon \Pi_{l \to k}(x) = x{\downarrow_k} \quad .$$

Corollary 1 *Given a standard measurable space $(\Omega, \mathcal{B})$ and a family of probability measures μ_n on the product spaces $(\Omega^n, \sigma(\mathcal{B}^n))$, $n \in \mathbb{N}$, that is consistent, i.e., for all $k < l$ and for all $F \in \mathcal{B}^k$,*

$$\mu_k(F) = \mu_l(\Pi^{-1}_{l \to k}(F)) \quad ,$$

then there exists a unique probability measure μ on the product space $(\Omega^\infty, \sigma(\mathcal{B}^\infty))$ that agrees with the family of probability measures, i.e.,

$$\mu_k(F) = \mu(\Pi^{-1}_{\infty \to k}(F)) \text{ for all } k \in \mathbb{N} \text{ and } F \in \mathcal{B}^n. \qquad \square$$

The probability measure μ is the unique measure defined in the *Extension Theorem of Carathéodory.*

3.2.2 Random Variables

A **random variable** or **measurable function** is a mapping $f \colon \Omega \to A$ from a measurable space $(\Omega, \mathcal{B})$ to another measurable space $(A, \mathcal{B}_A)$ such that every preimage is measurable:

$$\forall F \in \mathcal{B}_A \colon f^{-1}(F) = \{\omega \mid f(\omega) \in F\} \in \mathcal{B} \quad .$$

Given a probability measure μ on $(\Omega, \mathcal{B})$, the random variable f induces a probability measure μ_f on $(A, \mathcal{B}_A)$:

$$\forall F \in \mathcal{B}_A \colon \mu_f(F) = \mu(f^{-1}(F)) \quad .$$

For example, we can choose for A any countable set and $\mathcal{B}_A = \mathcal{P}(A)$, all subsets of A. Usually, random variables are used with the real numbers and the **Borel field** on the real numbers, a certain σ-field which we will not introduce here.

It is handy to lift operators over Ω to random variables over Ω. Thus, we may write $f + g$ instead of explicitly defining a random variable $h(\omega) = f(\omega) + g(\omega)$.

As usual, we also shortcut the notation for measures of random variables. For example, given the measure μ and some $a \in A$, we write $\mu(f = a)$ instead of $\mu(\{\omega \mid f(\omega) = a\})$. This generalizes to arbitrary statements over several random variables. One must however be careful that the resulting truth set is still measurable.

3.2.3 Stochastic Processes

Let $(\Omega, \mathcal{B}, \mu)$ be a probability space. A **stochastic process** or **random process** is a sequence of A-valued random variables $\langle x_1, x_2, \ldots \rangle$. A is called the **state space**.

Markov Chains

A stochastic process $\langle x_1, x_2, \ldots \rangle$ is called a **Markov process** if it has the **Markov property** (also called **memoryless**):

$$\mu(x_{n+1} = c_{n+1} \mid x_1 = c_1 \wedge \ldots \wedge x_n = c_n) = \mu(x_{n+1} = c_{n+1} \mid x_n = c_n)$$

for any n and for any $c_1, \ldots c_{n+1}$ such that

$$\mu(x_1 = c_1 \wedge \ldots \wedge x_n = c_n) > 0 \quad .$$

It is called a **Markov chain** if it is additionally **time-homogeneous**:

$$\mu(x_{n+1} = c_{n+1} \mid x_n = c_n) = \mu(x_{m+1} = c_{n+1} \mid x_m = c_n)$$

for any $n, m \in \mathbb{N}$ such that $\mu(x_n = c_n) > 0$ and $\mu(x_m = c_m) > 0$.

Note that we can translate any stochastic process into a Markov chain by encoding the history up to time n into the state of time n.

Chapter 4

Probabilistic Components

This section introduces probabilistic components and composition of such. We will consider the generalization of Moore automata to the probabilistic case by annotating transitions with probabilities. Similar to the non-probabilistic case in Focus, we can abstract from the state of probabilistic automata and obtain a description for probabilistic behavior. This description captures fully specified and probabilistic behavior. Adding the notion of channels yields probabilistic components and allows the composition of components.

4.1 Probabilistic I/O Automata

Recall how non-probabilistic (strongly causal or Moore) I/O automata were defined. Such an automaton N consists of a set of states S, a start state s_0, a transition function $\Delta\colon S \times I \to S$ and an output function $\omega\colon S \to O$. At some time of N's execution, say it is currently in state $s \in S$, the current output is given by $\omega(s)$ and the environment provides the next input $i \in I$. Then N passes over to state $\delta(s, i)$. As N is discrete, we can safely assume that S, I and O are countable.

Instead of passing from one state to the next in a deterministic way, we wish to randomly or probabilistically choose the next state, say by throwing a dice. Consider, a probabilistic I/O automaton M is currently in state $s \in S$ and the input symbol $i \in I$ is provided. Independent of the input but depending on the current state, we randomly choose an output symbol.

This can be modeled with a function $\omega\colon S \to \mathbf{dist}(O)$, where $\mathbf{dist}(X)$ is the set of distributions over a countable set X and a **distribution**[1] over X is a function $f\colon X \to [0,1]$ from X to the real numbers between 0 and 1 (both inclusive) with

[1]From Section 3.2.1 we know, that every distribution induces a probability measure μ over

the property that

$$\sum_{x\in X} f(x) = 1 \quad .$$

The value $\omega(s)(o)$ describes the probability that the automaton M outputs o given that it is currently in state s. The next state may depend on the current state, input and output, thus we obtain the probabilistic transition function $\Delta : S \times I \times O \to \mathbf{dist}(S)$, which says: if the automaton is in state s, has just output the symbol o, and reads the input i, then it will switch to state s' with probability $\Delta(s,i,o)(s')$.

Note that we require the independence of the output from the input, as we interpret both the n-th input as well as the n-th output symbol to occur at time t. This means, this definition enforces the minimal delay of one time step between the input and the automaton's reaction and is analogous to the restriction of non-probabilistic Moore automata.

Just as any joint distribution $P(X,Y)$ over two random variables X and Y can be decomposed into a marginal distribution $P(X)$ and a conditional distribution $P(X \mid Y)$ and vice versa, we can translate the two functions Δ and ω (ω plays the role of $P(X)$ and Δ the role of $P(X \mid Y)$) into a single function $\Delta\colon S \times I \to \mathbf{dist}(S \times O)$ according to:

$$\Delta(s,i)(s',o) = \omega(s)(o) \cdot \Delta(s,i,o)(s') \quad .$$

If this Δ fulfills

$$\sum_{s'} \Delta(s,i)(s',o) = \sum_{s'} \Delta(s,j)(s',o) \tag{4.1}$$

for all $i,j \in I$, s and o, i.e., the output is independent of the input, then we can translate it back by

$$\omega(s)(o) = \sum_{s'} \Delta(s,i)(s',o) \quad \text{for any } i \in I$$

$$\Delta(s,i,o)(s') = \begin{cases} d(s') & \text{if } \omega(s,o) = 0 \\ \frac{\Delta(s,i)(s',o)}{\omega(s)(o)} & \text{otherwise} \end{cases} \quad .$$

the measurable space $(X, \mathcal{P}(X))$ by

$$\forall F \subseteq X\colon \mu(F) = \sum_{x\in F} f(x).$$

and vice versa, by $f(x) = \mu(\{x\})$. Thus, distributions in this form are just another handier notation for probability measures of this particular space.

where $d \in \mathbf{dist}(S)$ is some arbitrary distribution. If we drop the independence restriction (4.1), we obtain probabilistic **Mealy** instead of Moore automata.

So far, we did not mention the probabilistic equivalent to the initial state s_0. We do not allow the first output to depend on the first input, but still we would like to randomly produce output in the first time step. Therefore, we replace the initial state s_0 by an initial state distribution $\delta \in \mathbf{dist}(S)$. Again, this δ combined with ω is equivalent to an initial state and output distribution $\delta \in \mathbf{dist}(S \times O)$. Note that time dependent output behavior can be obtained with any of the mentioned variants by storing the time within the state. This holds, in particular, for the first output.

We summarize, that a **probabilistic I/O automaton** consists of a state set S, input and output alphabets I and O, an initial state distribution $\delta \in \mathbf{dist}(S)$, a transition function $\Delta\colon S \times I \times O \to \mathbf{dist}(S)$ and an output function $\omega\colon S \to \mathbf{dist}(O)$. This preliminary definition will be refined further near the end of this section.

Input Sequences Feeding a sequence $i \in I^n$ to the non-probabilistic automaton N results in a sequence of states $\langle s_0, s_1, \ldots, s_n\rangle \in S^{n+1}$ and an output sequence $\langle o_0, o_1, \ldots, o_n\rangle \in O^{n+1}$. This is easily generalized to the infinite case $i \in I^\infty$.

When reading the input sequence $i \in I^n$, the probabilistic automaton M however may go through various sequences of states and output various sequences each with a different probability. These probabilities form a measure over $(S \times O)^{n+1}$, which describes more generally the probability of any subset of $(S \times O)^{n+1}$. But how can this be generalized to the infinite case? This is exactly where the concept of a random process and the *Kolmogorov Extension Theorem* (cf. Section 3.2.1) come into play. This theorem states basically that given consistent probabilities for finite sequences, there exists a unique probability measure over the infinite sequences consistent to these probabilities. We will now formally establish the theorem's application.

For the next few paragraphs, assume a fixed infinite input sequence $i \in I^\infty$. How are the probabilities of sequences $\varphi = \langle (s_0, o_0), (s_1, o_1), \ldots, (s_n, o_n)\rangle \in (S \times O)^{n+1}, n \in \mathbb{N}_0$, determined? Denote such probabilities by $p(\varphi)$. The initial state/output distribution δ determines the probability of sequences of length 1, i.e., for the case $n = 0$, it is clear that we have $p(\langle(s_0, o_0)\rangle) = \delta(s_0, o_0) = \delta(s_0)\cdot\omega(s_0)(o_0)$.

Let $n > 1$, given the probability for the sequence $\varphi{\downarrow_{n-1}}$, the probabilistic transition function defines the conditional probability that the next output is (s_n, o_n). We obtain $p(\varphi) = p(\varphi{\downarrow_{n-1}}) \cdot \Delta(s_{n-1}, i_{n-1})(s_n, o_n)$ and unfolding the recursion yields:

$$p(\langle(s_0,o_0),(s_1,o_1),\ldots,(s_n,o_n)\rangle) = \delta(s_0,o_0)\cdot\prod_{k=0}^{n-1}\Delta(s_k,i_k)(s_{k+1},o_{k+1}) \quad .$$

Ad Infinitum An alternative point of view is to consider $(S\times O)$-valued random variables $x_k, k\in\mathbb{N}_0$, where $x_k=(s_k,o_k)$ if the automaton is in state s_k at time k and outputs the symbol o_k. Such an infinite sequence $\langle x_0,x_1,\ldots\rangle$ of random variables is a random process and we can apply the insights from Section 3.2. As the underlying measurable space for each variable x_k, we choose the space $((S\times O),\mathcal{P}(S\times O))$ as both S and O are countable. For this random process to be fully defined, we have to construct a measure μ_n for every $n\in\mathbb{N}_0$ over subsets of sequences of length n, i.e., for the product space $((S\times O)^n,\sigma(\mathcal{P}(S\times O)^n))$. These measures correspond directly to the probabilities $p(\varphi)$ we have just defined! We obtain $\mu_n(\{\varphi\})=p(\varphi)$ for all $\varphi\in(S\times O)^n, n>0$. This suffices for the definition of μ_n as the singleton sets generate all other sets by countable union. In the extreme case $n=0$, we have only the empty sequence $\langle\ \rangle$ and the trivial measure $\mu_0(\{\langle\ \rangle\})=1$.

Now, we apply the *Kolmogorov Extension Theorem* (cf. Section 3.2.1). The probability measures μ_n are consistent (for all $n>0$):

$$\begin{aligned}
&\mu_{n+1}(\Pi^{-1}_{n+1\to n}(\{\varphi\})) = \mu_{n+1}(\{\varphi\cdot(s',o') \mid s'\in S\wedge o'\in O\})\\
&\quad= \sum_{s'\in S,o'\in O}\mu_{n+1}(\{\varphi\cdot(s',o')\})\\
&\quad= \sum_{s'\in S,o'\in O}p(\varphi)\cdot\Delta(s_{n-1},i_{n-1})(s',o')\\
&\quad= p(\varphi)=\mu_n(\{\varphi\})
\end{aligned}$$

as in the case $n=0$:

$$\begin{aligned}
&\mu_1(\Pi^{-1}_{1\to 0}(\{\langle\ \rangle\}))\\
&\quad= \sum_{s'\in S,o'\in O}\mu_1(\{\langle(s',o')\rangle\})\\
&\quad= \sum_{s'\in S,o'\in O}\delta(s',o') = 1 = \mu_0(\{\langle\ \rangle\}) \quad .
\end{aligned}$$

Then the theorem states, that a unique probability measure μ on the measurable space $((S\times O)^\infty,\sigma(\mathcal{P}(S\times O)^\infty))$ exists such that, for all $\varphi\in(S\times O)^n$ and $n\in\mathbb{N}_0$,

$$\mu_n(\{\varphi\}) = \mu(\{\rho\in(S\times O)^\infty \mid \varphi\sqsubseteq\rho\}) \quad .$$

This probability measure μ fully describes the behavior of the probabilistic I/O automaton M for the given input i and we can ask for the probability of subsets of $(S \times O)^\infty$, as long as they are measurable, i.e., they are elements of the σ-field $\sigma(\mathcal{P}(S \times O)^\infty)$.

We call sets of the form $\{\rho \in (S \times O)^\infty \mid \varphi \sqsubseteq \rho\}$ **basic cylinders**, denoted by $\mathcal{C}(\varphi)$. The underlying σ-field $\sigma(\mathcal{P}(S \times O)^\infty)$ contains these basic cylinders and is closed under countable unions, countable intersections and the complement operation. In particular, it contains the singleton sets $\{\rho\}$ for $\rho \in (S \times O)^\infty$.

To make the dependence of the measure μ on the input i explicit, we write in the following instead $P_i = \mu$. The induced random process $\langle x_0, x_1, \ldots\rangle$ is actually a Markov process (see Section 3.2.3). The Markov property holds, i.e., the process is memory-less, because we have

$$\begin{aligned}
&P_i(x_{n+1} = (s_{n+1}, o_{n+1}) \mid x_n = (s_n, o_n)) \\
&\quad= \frac{P_i(x_{n+1} = (s_{n+1}, o_{n+1}) \wedge x_n = (s_n, o_n))}{P_i(x_n = (s_n, o_n))} \\
&\quad= \frac{\sum_{\varphi \in (S,O)^n} P_i(x_{n+1} = (s_{n+1}, o_{n+1}) \wedge x_n = (s_n, o_n) \wedge \langle x_0, \ldots, x_{n-1}\rangle = \varphi)}{\sum_{\varphi \in (S,O)^n} P_i(x_n = (s_n, o_n) \wedge \langle x_0, \ldots, x_{n-1}\rangle = \varphi))} \\
&\quad= \frac{\sum_{\varphi \in (S,O)^n} \Delta(s_n, i.n)(o_{n+1}, s_{n+1}) \cdot P_i(x_n = (s_n, o_n) \wedge \langle x_0, \ldots, x_{n-1}\rangle = \varphi)}{\sum_{\varphi \in (S,O)^n} P_i(x_n = (s_n, o_n) \wedge \langle x_0, \ldots, x_{n-1}\rangle = \varphi))} \\
&\quad= \Delta(s_n, i.n)(s_{n+1}, o_{n+1}) \\
&\quad= P_i(x_{n+1} = (s_{n+1}, o_{n+1}) \mid x_n = (s_n, o_n) \wedge \ldots \wedge x_0 = (s_0, o_0)) \quad .
\end{aligned}$$

This property means that the probability of the next state and output only depends on the last state and not on how this state was reached. In general, the value $\Delta(s_n, i.n)(s_{n+1}, o_{n+1})$ may depend on n because of the input $i.n$ and the process would not be time-homogeneous.

Output Behavior Like in the non-probabilistic case, we follow the idea that only the output of an automaton is observable from the outside. The state is internal and not visible. To obtain the probabilities of the output sequences, we project the tuples (s, o) to the output using the projection function[2] $q\colon (S \times O)^\infty \to O^\infty$ with $q(s \times o) = o$ for any $(s \times o) \in (S \times O)^\infty$. Since q is a random variable[3], it induces a probability measure Q_i on the measurable space $(O^\infty, \sigma(\mathcal{P}(O)^\infty))$ by $Q_i(X) = P_i(q^{-1}(X))$ for all $X \in \sigma(\mathcal{P}(O)^\infty)$. This concludes the definition of the

[2] $s \times o$ denotes the sequence of $(S \times O)^\infty$ with $(s \times o).n = (s.n, o.n)$ for all $n \in \mathbb{N}$. With this function, we can translate between elements of $(S \times O)^\infty$ and $S^\infty \times O^\infty$.

[3] In order to see that q is measurable function (i.e., it is a random variable) into the measurable space $(O^\infty, \sigma(\mathcal{P}(O)^\infty))$ we have to prove that every preimage of a basic cylinder is measurable. This is enough, because the basic cylinders generate the whole measurable space and the "good

output behavior of a probabilistic I/O automaton as the family $Q_i, i \in I^\infty$, of probability measures.

We call two automata **behaviorally equivalent** if they have the same output behavior. Finding small behavioral equivalent automata corresponds to *state lumping* of a Markov chain as described in [RS89].

Note that both the probabilities $P_i(\mathcal{C}(\varphi))$ and $Q_i(\mathcal{C}(\rho))$ for sequences $\varphi \in (S \times O)^{n+1}$ and $\rho \in O^{n+1}$ depend only on the first n elements of the input sequence i. That means, for all input sequences $i, i' \in I^\infty$ we have

$$i{\downarrow_n} = i'{\downarrow_n} \implies P_i(\mathcal{C}(\varphi)) = P_{i'}(\mathcal{C}(\varphi)) \quad , \tag{4.2}$$

and accordingly for Q_i and $Q_{i'}$.

Non-Probabilistic Output Functions Before advancing to other topics, we simplify the previous definition of probabilistic I/O automata without reducing their expressiveness.

Every prob. I/O automaton M can be translated into a behaviorally equivalent I/O automaton K which has a non-probabilistic output function ω_K, i.e., for all states s there exists an output o such that $\omega_K(s)(o) = 1$, for all other outputs the probability is zero.

The new I/O automaton K consists of the state space $S_K = S_M \times O$ and the following functions:

$$\begin{aligned}
\delta_K((s,o)) &= \delta(s) \cdot \omega_M(s)(o) \\
\omega_K((s,o),o') &= \begin{cases} 1 & \text{if } o = o' \\ 0 & \text{otherwise} \end{cases} \\
\Delta_K((s,o),i,o)((s',o')) &= \Delta_M(s,i,o)(s') \cdot \omega_M(s')(o') \quad .
\end{aligned} \tag{4.3}$$

For all $s \in S^n, o \in O^n$, we have $P_{K,i}(\mathcal{C}((s \times o) \times o')) = 0$ for $o' \neq o$ by induction over n. The base case follows from

$$P_{K,i}(\mathcal{C}(\langle((s_0,o_0),o_0')\rangle)) = \delta_K((s_0,o_0)) \cdot \omega_K((s_0,o_0))(o_0')$$

sets" principle is applicable [GDG$^+$01]. Indeed we have for all $o' \in O^n, n \in \mathbb{N}_0$:

$$\begin{aligned}
q^{-1}(\{o \mid o' \sqsubseteq o\}) &= \{\varphi \mid \exists s' \in S^n \colon (s' \times o') \sqsubseteq \varphi\} \\
&= \bigcup_{s' \in S^n} \{\varphi \mid (s' \times o') \sqsubseteq \varphi\} \in \sigma(\mathcal{P}(S \times O)^\infty) \quad .
\end{aligned}$$

and Equation (4.3). The inductive step is

$$\begin{aligned}
o \neq o' &\implies o_n \neq o'_n \;\vee\; o{\downarrow_n} \neq o'{\downarrow_n} \\
&\overset{IH}{\implies} \omega_K((s_n, o_n))(o'_n) = 0 \;\vee\; P_{K,i}(\mathcal{C}((s \times o) \times o'{\downarrow_n})) = 0 \\
&\implies P_{K,i}(\mathcal{C}((s \times o) \times o')) \\
&\quad = \Delta_M((s_{n-1}, o_{n-1}), i_{n-1}, o'_{n-1})((s_n, o_n)) \cdot \omega_K((s_n, o_n))(o'_n) \\
&\qquad \cdot P_{K,i}(\mathcal{C}((s \times o) \times o'{\downarrow_n})) \\
&\quad = 0 \quad .
\end{aligned}$$

Similarly, it is easy to show that $P_{K,i}(\mathcal{C}((s \times o) \times o)) = P_{M,i}(\mathcal{C}(s \times o))$ and since for all $o' \in O^n, n \in \mathbb{N}_0$,

$$\begin{aligned}
Q_{K,i}(\mathcal{C}(o')) &= P_{K,i}(\bigcup_{(s \times o) \in (S \times O)^n} \mathcal{C}((s \times o) \times o')) \\
&= P_{K,i}(\bigcup_{s \in S^n} \mathcal{C}((s \times o') \times o')) \\
&= P_{M,i}(\bigcup_{s \in S^n} \mathcal{C}(s \times o')) \\
&= Q_{M,i}(\mathcal{C}(o')) \quad ,
\end{aligned}$$

it follows that the output behaviors of M and K are identical.

Thus, I/O automata of the form as K are sufficient and they can be represented with more specific functions. Namely, as the output function is non-probabilistic, it can be replaced by a function $\omega_K \colon S \to O$. The values $\Delta_K((s, o), i, o')$ of the transition function are irrelevant if $o' \neq o$, thus it can be represented by a function $\Delta_K \colon S \times I \to \mathbf{dist}(S)$.

We conclude these considerations with the final definition of a **probabilistic I/O automaton** as a tuple $(S, I, O, \delta, \Delta, \omega)$ where

- S is a countable set of states,
- I and O are countable input and output alphabets, respectively,
- $\delta \colon \mathbf{dist}(S)$ is the initial state distribution,
- $\Delta \colon S \times I \to \mathbf{dist}(S)$ is the probabilistic transition function,
- $\omega \colon S \to O$ is the deterministic output function.

The probabilities occur only in the state transitions and are independent of the output, thus it makes sense to consider the probability of state sequences. For all

$s = \langle s_0, s_1, \ldots, s_n \rangle \in S^n, n \in \mathbb{N}$, we have

$$P_i(\mathcal{C}(s)) = \delta(s_0) \cdot \prod_{k=0}^{n-1} \Delta(s_k, i_k)(s_{k+1}) \quad .$$

By element-wise application of the output function ω mapping state sequences S^n to output sequences O^n, $n \in \mathbb{N}_{0,\infty}$, we can characterize the output behavior by (for all $\rho \in O^n, n \in \mathbb{N}_0$)

$$\begin{aligned} Q_i(\mathcal{C}(\rho)) &= P_i(\omega^{-1}(\mathcal{C}(\rho))) \\ &= P_i(\{\varphi \mid \rho \sqsubseteq \omega(\varphi)\}) \\ &= \sum_{\substack{\varphi \in S^n \\ \omega(\varphi)=\rho}} P_i(\mathcal{C}(\varphi)) \quad , \end{aligned}$$

thus summing the probabilities of all state sequences producing ρ.

In [Buk95], the author considers several different types of *stochastic automata*, equivalences of automata and translations between different types of automata. Actually, the introduced probabilistic I/O automata are similar to *Markovian stochastic Moore-automata*, however, with the difference that the output of a Moore-automaton depends only on the *successor* state whereas we require that the output depends only on the *current* state in order to obtain a strongly causal behavior. Given two stochastic automata A and B and fixed initial state distributions δ_A and δ_B (in [Buk95], called *state vectors*), the equivalence of A and B with respect to δ_A and δ_B, which compares the probability for every input/output combination, is identical to our notion of behavioral equivalence. Therefore, we refer the interested reader to [Buk95] to obtain further insights into probabilistic automata.

Notation We finish this section by summarizing and unifying the notation introduced so far.

Automata are usually named with the capital Latin letters A, B, K, M and N. For the induced probability measure of an automaton M and input i, we write $P_{M,i}$. Thus, the probability for a set of state sequences $X \subseteq S^\infty$ is given by $P_{M,i}(X)$. The induced probability measure of output sequences is called $Q_{M,i}$.

Alternatively, we reuse the automaton name, say M, to denote the family of S^∞-valued random variables corresponding to the measure $P_{M,i}$. Thus, we have for example $\mathbf{P}[M(i) = s] = P_{M,i}(\{s\})$ for $s \in S^\infty$ and $\mathbf{P}[M(i) \sqsupseteq s] = P_{M,i}(\mathcal{C}(s))$ for $s \in S^*$. Recall that formally a X-valued random variable is a function from a probability space to X. In this case, the random variable $M(i)$ is defined over the probability space $(S^\infty, \sigma(\mathcal{P}(S)^\infty), P_{M,i})$, namely as the identity function over S^∞.

The useful consequence of the notation with random variables is that we can easily apply operations to random variables and obtain new variables. In the case of $M(i)$, we can apply some function $h\colon S^\infty \to X$ and obtain the variable $h(M(i))\colon S^\infty \to X$ with

$$\mathbf{P}[h(M(i)) = x] = P_{M,i}(\{\varphi \in S^\infty \mid h(\varphi) = x\}) \quad .$$

Because of the strong causality stated in Equation (4.2), the probability of a finite output sequence depends only on a finite part of the input sequence. Therefore, it is legal to write $\mathbf{P}[M(i) = s]$ for all $n \in \mathbb{N}_0, i \in I^n$ and $s \in S^{n+1}$, i.e., in this case we understand $M(i)$ as a S^{n+1}-valued random variable.

As we consider usually several probabilistic automata, where each automaton brings its own probability space, we have to be careful not to mix random variables without common probability space. Only random variables with a common underlying probability space may be used within a single expression inside $\mathbf{P}[\cdot]$. In particular, we are not allowed to use several random variables for different inputs or of different automata within the same expression because there is no common probability space. However according to Equation (4.2), it is legal to refer to random variables $M(i_k), i_k \in I^*$, when the i_k are prefixes of a common infinite sequence i (i.e., for all k, $i_k \sqsubseteq i$).

4.2 Probabilistic Behavior

From strongly causal Focus components, we already know that we can describe an interactive non-probabilistic behavior as a function $I^\infty \to O^\infty$ mapping infinite input sequences to infinite output sequences (or sets of output streams in case of nondeterministic components). We saw in the previous section, that, in the case of probabilistic components, we do not obtain a single output sequence but a probability measure describing the output behavior. This measure assigns probabilities to sets of infinite output sequences. It is important that the probability is assigned to sets and is not restricted to probabilities of single sequences. As an example, imagine the tossing of a coin. Each side occurs with probability $1/2$ and every infinite sequence of heads and tails occurs with probability zero. All cylinders $\mathcal{C}(\varphi), \varphi \in O^n$ however have non-zero probability $(1/2)^n$.

We want to formally define what a probabilistic behavior with input I and output O is. Independent of the concrete behavior, we define which output events are assigned probabilities; events are measurable sets, i.e., we define the set of the measurable sets of infinite output sequences. In Section 3.2, we defined for this purpose the σ-field $\sigma(\mathcal{P}(O)^\infty)$, that is the smallest σ-field containing all cylinders $\mathcal{C}(\varphi), \varphi \in O^*$. As this σ-field depends only on the alphabet O, we denote it by $\mathcal{B}_O$.

Upon this σ-field, a specific behavior provides for each input a probability measure, which is a function $\mathcal{B}_O \to [0,1]$ fulfilling the properties of a probability measure (see again Section 3.2). We give to the set of all these probability measures the name O^P, to reflect the similarity to the notation O^∞.

Finally, a **(deterministic) probabilistic behavior** is a function $f \colon I^\infty \to O^P$ that maps any infinite input sequence i to a measure over infinite output sequences. As in the non-probabilistic case, this fully captures the observable output behavior of a probabilistic component, thereby abstracting from its inner state. We call it deterministic, as it allows only one possible output distribution. In Chapter 6, we will also consider nondeterministic probabilistic behavior functions, where the system can "choose" from a set of possible output distributions.

As with probabilistic automata, we can reuse the identifier $f(i)$ to denote the corresponding random variable. For example we have, for all $\varphi \in O^*$,

$$\mathbf{P}[f(i) \sqsupseteq \varphi] = f(i)(\mathcal{C}(\rho)) \quad .$$

We apply operations $h \colon O^\infty \to X$ to the variable $f(i)$ creating new variables $h(f(i))$ with an induced distribution.

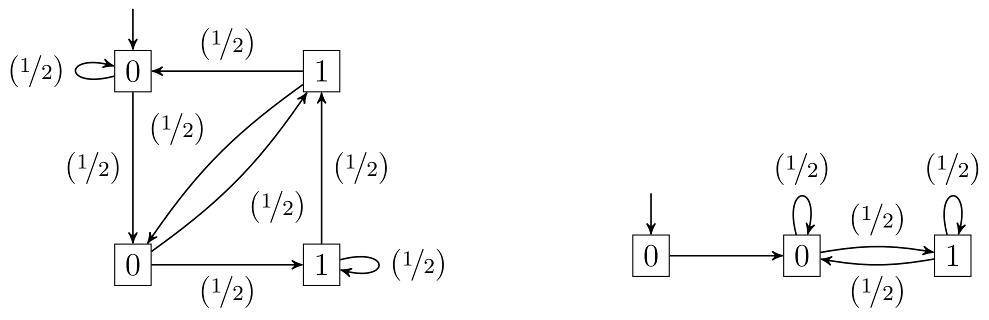
Behavior Abstraction In the previous section, we presented already the probabilistic behavior of a probabilistic I/O automaton M. It is given by the probability measures $Q_{M,i}$. We call the probabilistic behavior $f_M \colon I^\infty \to O^P$ with $f_M(i) = Q_{M,i}$ the **behavior abstraction** of the I/O automaton M.

The central property of the output behavior of I/O automata was stated in Equation (4.2), which we repeat here (for all $n \in \mathbb{N}_0, i, i' \in I^n$ and $\varphi \in O^{n+1}$):

$$i{\downarrow}_n = i'{\downarrow}_n \implies \mathbf{P}[f_M(i) = \varphi] = \mathbf{P}[f_M(i') = \varphi] \quad .$$

It states, that the output of an automaton only depends on past inputs and not on the present or future inputs. This property is called the **strong causality property**. Accordingly, we call every probabilistic behavior which fulfills the causality property **strongly causal**. Obviously, not every behavior is strongly causal, however the probabilistic abstraction of every I/O automaton is strongly causal.

Example 4.1 The two automata A and B shown in Figure 4.1 have the same behavior, that means $f_A = f_B$. The nodes in the graph correspond to the automaton's states. Each node's label defines its output. In this example, the states are not given any names. The transitions define the transition relation, where each transition's label defines its probability but probabilities equal to one are omitted. The initial distribution is given by source-less arrows. $\square$



(a) Automaton A (b) Automaton B

Figure 4.1: Two behaviorally equivalent prob. I/O automata.

We will now show an important theorem which shows that probabilistic I/O automata have the same expressiveness as strongly causal prob. behaviors. An analogous theorem exists for non-probabilistic behavior functions [Bro07]:

> Every deterministic and strongly causal (non-probabilistic) behavior function is the (interface) abstraction of some total and deterministic Moore automaton.

Theorem 2 *Every strongly causal probabilistic behavior is the abstraction of some probabilistic I/O automaton.* □

Proof Let f be some strongly causal, probabilistic behavior with input and output alphabets I and O. We will construct an I/O automaton $M = (S, I, O, \delta, \Delta, \omega)$ with behavior abstraction $f_M = f$. We choose the state space $S = (I \times O)^* \times O$, such that each state encodes the input and output history as well as the current output symbol. However, only states $(\tau \times \rho, o) \in S$ with $\mathbf{P}[f(\tau) = \rho \cdot \langle o\rangle] > 0$ are reachable.

The output function ω simply returns the current output symbol of each state:

$$\omega((\sigma, o)) = o \quad .$$

The initial state distribution is given by the distribution $f(\langle\,\rangle)$:

$$\delta((\langle\,\rangle, o)) = \mathbf{P}[f(\langle\,\rangle) = \langle o\rangle] \quad .$$

The transition function is determined by the conditional probabilities of f producing an output symbol given a certain output history (note that we consider only reachable states $(\tau \times \rho, o)$):

$$\begin{aligned}&\Delta((\tau \times \rho, o), i)((\tau \times \rho \cdot \langle (i, o)\rangle, o')) \\ &\quad = \mathbf{P}[f(\tau \cdot \langle i\rangle) = \rho \cdot \langle o, o'\rangle \mid f(\tau) = \rho \cdot \langle o\rangle] \quad .\end{aligned}$$

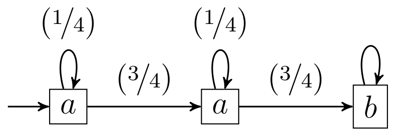


Figure 4.2: A prob. I/O automaton with infinite state reconstruction.

Given these defintions, we can characterize the induced probability space. For every state sequence $s \in S^{n+1}$ that is consistent, i.e., it exists some input sequence $i \in I^\infty$ and output sequence $\rho \in O^\infty$ such that $s.k = (i{\downarrow_k} \times \rho{\downarrow_k}, \rho.k)$ for all $k \in \{0, \ldots, n\}$, it holds:

$$\begin{aligned}
\mathbf{P}[M(i) \sqsupseteq s] &= \delta(s.0) \cdot \prod_{k=0}^{n-1} \Delta(s.k, i.k)(s.k+1) \\
&= \mathbf{P}[f(\langle\,\rangle) = \langle \rho.0 \rangle] \cdot \prod_{k=0}^{n-1} \frac{\mathbf{P}[f(i{\downarrow_k} \cdot \langle i.k \rangle) = \rho{\downarrow_k} \cdot \langle \rho.k, \rho.k+1 \rangle]}{\mathbf{P}[f(i{\downarrow_k}) = \rho{\downarrow_k} \cdot \langle \rho.k \rangle]} \\
&= \mathbf{P}[f(\langle\,\rangle) = \langle \rho.0 \rangle] \cdot \prod_{k=0}^{n-1} \frac{\mathbf{P}[f(i{\downarrow_{k+1}}) = \rho{\downarrow_{k+2}}]}{\mathbf{P}[f(i{\downarrow_k}) = \rho{\downarrow_{k+1}}]} \\
&= \mathbf{P}[f(i{\downarrow_n}) = \rho{\downarrow_{n+1}}] \quad .
\end{aligned}$$

Finally, as there is only a single state sequence producing a certain output sequence $\rho \in O^{n+1}$, we obtain:

$$\begin{aligned}
\mathbf{P}[f_M(i) \sqsupseteq \rho] &= \mathbf{P}[\omega(M(i)) \sqsupseteq \rho] \\
&= \mathbf{P}[M(i) \sqsupseteq (i \times \rho{\downarrow_n}, \rho.n)] \\
&= \mathbf{P}[f(i{\downarrow_n}) = \rho] \quad .
\end{aligned}$$

■

The theorem shows that the definition of probabilistic I/O automata exhausts the possibilities of strongly causal probabilistic behaviors.

In the proof, we constructed a probabilistic automaton for a given behavior. In the proof for non-probabilistic behavior, the constructed automaton is also minimal. Here however, the resulting I/O automaton reaches an infinite part of the state space even if a finite automaton exists. This problem persists even if we construct the automaton analogously to the minimal construction in the non-probabilistic case. We show this in the following example.

Example 4.2 We inspect the probabilistic behavior abstraction f of the I/O automaton shown in Figure 4.2. The automaton has only one possible (or constant) input[4], so in the following we can omit the input argument to f. Analogously to

[4]This corresponds to the idea of *no input*.

the construction of a minimal automaton in the non-probabilistic case, we consider the *remaining* behavior in a certain state. We want to show that although there exists a finite automaton for the behavior f, there exist infinitely many remaining behaviors.

At first, we give a closed representation of the behavior f for output sequences $\langle a\rangle^{k+1}$:

$$\forall k \geq 0 : f(\langle a\rangle^k \cdot \langle a\rangle) = 4^{-k} + k \cdot 4^{1-k} \cdot \frac{3}{4} = (1+3k)4^{-k} \quad .$$

To prove our claim, it suffices to consider the conditional probability that f produces another output a given it has already produced $\langle a\rangle^{k+1}$:

$$\begin{aligned}\mathbf{P}[f \sqsupseteq \langle a\rangle^{k+2} \mid f \sqsupseteq \langle a\rangle^{k+1}] &= \frac{(1+3(k+1))4^{-k-1}}{(1+3k)4^{-k}} \\ &= \frac{1}{4} \cdot \frac{4+3k}{1+3k} = \frac{1}{4}\left(\frac{3}{1+3k} + 1\right) \quad .\end{aligned}$$

These probabilities are all different. Thus, we reach infinitely many different states, in the sense that the states have different output distributions. That means, the reachable state space would be infinite if we followed the same construction as in the non-probabilistic case. □

4.3 Composition

In order to compose several behavior functions or automata, we introduce the notion of channels. An output channel may be connected to an input channel such that the owning components communicate over this shared channel. This communication is unidirectional with exactly one sender and one or more receivers. For bidirectional communication, a second channel has to be used.

In the same way the syntactic interface of non-probabilistic components is defined [Bro00], we say that a strongly causal prob. behavior with input and output channels I and O has the **syntactic interface** $I \triangleright O$. We represent channels using variables, thus the input and output alphabets are the valuations of the variable sets I and O. Such behaviors, which can be defined as the behavior abstraction of an I/O automaton, are called **probabilistic deterministic components** and we introduce the modified notation $I \overset{P}{\triangleright} O$ to refer to probabilistic components only. Note, that we always require components to be strongly causal.

Recall that the type of variables and accordingly of channels is determined by the function **type** (see Section 3.1). We will assume the existence of a global function **type**, such that a variable never occurs with two different types during composition.

In this section, we define the operators needed to construct composite components. Components can be connected and can communicate over shared channels. For locality reasons, we may hide communication channels and remove them from the composition's interface.

We will introduce two special cases of composition, namely *parallel composition* and *feedback*, which suffice to define the general composition. The former is essential to allow parallel independent execution of several components, the latter provides the essence of communication: The dependence of input on previous output. In any case, we forbid several components to write to the same channel. We will consider two systems with the syntactic interfaces $I_1 \overset{P}{\rhd} O_1$ and $I_2 \overset{P}{\rhd} O_2$ such that every channel has at most one source:

$$O_1 \cap O_2 = \emptyset \quad .$$

After introducing composition, we explain how to realize channel hiding.

All of these operators will be defined in terms of probabilistic I/O automata. This allows us to easily comprehend the definitions as they immediately describe their operationalization. From each definition we derive the behavior abstraction of the resulting automaton, which leads us to the operators' semantics in terms of probabilistic behavior functions.

By defining the operators for probabilistic behaviors in this way, we guarantee the operators to fulfill the *congruence* property. This means, for each operator and any I/O automaton, we can either at first abstract and apply the operator to the resulting behaviors or we apply the operator to the automata and abstract the composite automaton afterwards. In both cases the resulting behavior will be identical. Or put in another way, behavior abstraction forms an homomorphism from prob. I/O automata to prob. behavior with respect to the composition and hiding operators.

4.3.1 Parallel Composition

We consider two components executed in parallel without communication. They may share some input channels but cannot feed their output to the other component:

$$O_1 \cap I_2 = O_2 \cap I_1 = \emptyset \quad .$$

Independent of the underlying components' semantics, we can declare the syntactic interface $I \overset{P}{\rhd} O$ of the composition with $O = O_1 \cup O_2$ and $I = I_1 \cup I_2$.

We write $M_1 \varobar M_2$ to denote the parallel composition of two automata or behavior functions.

Operational Semantics We begin with the composition of two automata $M_1 = (S_1, \bar{I}_1, \overline{O}_1, \delta_1, \Delta_1, \omega_1)$ and $M_2 = (S_2, \bar{I}_2, \overline{O}_2, \delta_2, \Delta_2, \omega_2)$. They are executed in parallel and cannot communicate in any way. In stochastics, this is called statistical independence and means that the probability of the conjunct of two events equals the product of the probabilities of each event. This is reflected in the following definition. The composite automaton is given by the tuple $(S, \bar{I}, \overline{O}, \delta, \Delta, \omega)$ with

$$\begin{aligned} S &= S_1 \times S_2 \\ \delta(s_1, s_2) &= \delta_1(s_1) \cdot \delta_2(s_2) \\ \Delta((s_1, s_2), i)(s_1', s_2') &= \Delta_1(s_1, i|_{I_1})(s_1') \cdot \Delta_2(s_2, i|_{I_2})(s_2') \\ \omega(s_1, s_2) &= \omega_1(s_1) \uplus \omega_2(s_2) \quad . \end{aligned}$$

Recall that the binary operator $\uplus$ merges two valuations.

Behavior Abstraction Next, we consider how the parallel composition is reflected in its behavior.

Let $i \in \bar{I}^\infty$ be some input sequence and $s = \langle (s_0^1, s_0^2), (s_1^1, s_1^2), \ldots \rangle \in S^n = (S_1 \times S_2)^n$ some finite state sequence. Using the superscript index l, we denote the l-th component of a tuple or the sequence of all l-th components of a sequence of tuples. It follows

$$\begin{aligned} \mathbf{P}[M(i) \sqsupseteq s] &= \delta(s_0) \cdot \prod_{k=0}^{n-1} \Delta(s_k, i_k)(s_{k+1}) \\ &= \prod_{l \in \{1,2\}} \delta_l(s_0^l) \cdot \prod_{k=0}^{n-1} \Delta_l(s_k^l, i_k|_{I_l})(s_{k+1}^l) \\ &= \prod_{l \in \{1,2\}} \mathbf{P}[M_l(i|_{I_l}) \sqsupseteq s^l] \quad . \end{aligned}$$

This equation incorporates three different probability spaces. But at least in this case, we can retrieve the random variables $M_1(i|_{I_1})$ and $M_2(i|_{I_2})$ from the variable $M(i)$:

$$\forall n \in \mathbb{N}_0, s^1 \in S_1^n \colon \mathbf{P}[\exists s^2 \in S_2^n \colon M(i) \sqsupseteq s^1 \times s^2] = \mathbf{P}[M_1(i|_{I_l}) \sqsupseteq s^1]$$

and analogously for the variable $M_2(i|_{I_2})$. Thus, we can interpret the variables $M_1(\cdot)$ and $M_2(\cdot)$ also within the probability space of the composition and there they are indeed statistically independent.

Instead of first defining $M(i)$ and afterwards deriving $M_1(\cdot)$ and $M_2(\cdot)$, we could alternatively define the variables $M_1(\cdot)$ and $M_2(\cdot)$ as independent variables and derive $M(i)$ as their conjunction:

$$M(i) = s^1 \times s^2 \iff M_1(i|_{I_1}) = s^1 \wedge M_2(i|_{I_2}) = s^2 \quad ,$$

which again reflects the similarity to the non-probabilistic case.

Finally, we can characterize the behavior abstraction f of the composition. Let f_1 and f_2 be the behavior abstractions of M_1 and M_2, respectively. Then for all output sequences $\rho \in \overline{O}^* = \overline{O_1 \cup O_2}^*$, holds

$$\begin{aligned}
\mathbf{P}[f(i) \sqsupseteq \rho] &= \mathbf{P}[\omega(M(i)) \sqsupseteq \rho] \\
&= \mathbf{P}[\bigwedge_{l\in\{1,2\}} \omega(M_l(i|_{I_l})) \sqsupseteq \rho|_{O_l}] \\
&= \prod_{l\in\{1,2\}} \mathbf{P}[\omega(M_l(i|_{I_l})) \sqsupseteq \rho|_{O_l}] \\
&= \prod_{l\in\{1,2\}} \mathbf{P}[f_l(i|_{I_l})) \sqsupseteq \rho|_{O_l}] \quad .
\end{aligned}$$

This equation provides the definition of the **parallel composition of probabilistic behaviors** and we reuse the operator $\oplus$ with behavior functions.

From the above equations the following important property follows.

Theorem 3 (Associative and commutative) *Parallel composition is both associative and commutative.* □

This means that the parallel composition of several probabilistic behaviors $\mathcal{F} = \{f_1, \ldots, f_n\}$ can be performed in arbitrary order. The notation $\oplus\mathcal{F}$ is therefore a reasonable abbreviation for $f_1 \oplus f_2 \oplus \ldots \oplus f_n$.

4.3.2 Feedback

We consider now a component M with the syntactic interface $I \overset{P}{\triangleright} O$. We feedback the output channels which also occur syntactically as input channels, thus creating a loop. This poses no problems as components are strongly causal and delay the output by at least one time step. We feedback all channels C that occur both as input and output of the component, i.e., $C = I \cap O$. The resulting syntactic interface is $(I \setminus O) \overset{P}{\triangleright} O$ (it is $I \setminus C = I \setminus O$). The resulting component is denoted by $M^{\circlearrowleft}$.

Operational Semantics Let M be a probabilistic I/O automaton given as the usual tuple. The feedback of M is the automaton $M^{\circlearrowleft}$ defined by the tuple $(S, \overline{I \setminus O}, \overline{O}, \delta, \Delta^{\circlearrowleft}, \omega)$ with

$$\begin{aligned}
&\Delta^{\circlearrowleft} \colon S \times \overline{I \setminus O} \to \mathbf{dist}(S) \\
&\Delta^{\circlearrowleft}(s, i') = \Delta(s, i' \uplus \omega(s)|_I) \quad .
\end{aligned}$$

This exactly reflects the informal description that the output of the current state (i.e., $\omega(s)$) determines (part of) the input for the next transition.

Behavior Abstraction Before we derive the behavior abstraction of the feedback automaton, we review its induced probability space. Let $M(i) \in \overline{O}^{n+1}$ and $M^{\circlearrowleft}(j) \in \overline{O}^{n+1}$ be the respective random variables for finite input sequences $i \in \overline{I}^n, j \in \overline{I \setminus O}^n, n \in \mathbb{N}_0$. Then we have for all state sequences $s \in S^{n+1}$:

$$\begin{aligned}
\mathbf{P}[M^{\circlearrowleft}(j) = s] &= \delta(s_0) \cdot \prod_{k=0}^{n-1} \Delta^{\circlearrowleft}(s_k, j_k)(s_{k+1}) \\
&= \delta(s_0) \cdot \prod_{k=0}^{n-1} \Delta(s_k, j_k \uplus \omega(s_k)|_I)(s_{k+1}) \\
&= \mathbf{P}[M(j \uplus \omega(s){\downarrow_n}|_I) = s] \quad .
\end{aligned}$$

Thereby, the union $\uplus$ of two sequences of valuations is meant to be applied elementwise. Again, we have to realize that this equation incorporates two different probability spaces and this time there is no reasonable way to embed both random variables ($M(i)$ and $M^{\circlearrowleft}(j)$) within a common space.

Now, we are ready to characterize the behavior abstraction $f^{\circlearrowleft}$ of $M^{\circlearrowleft}$ with regard to the behavior abstraction f of M. For all input sequences $j \in \overline{I \setminus O}^n$ and output sequences $\rho \in \overline{O}^{n+1}$, it is

$$\begin{aligned}
\mathbf{P}[f^{\circlearrowleft}(j) = \rho] &= \mathbf{P}[\omega(M^{\circlearrowleft}(j)) = \rho] \\
&= \mathbf{P}[\omega(M(j \uplus \rho{\downarrow_n}|_I)) = \rho] \\
&= \mathbf{P}[f(j \uplus \rho{\downarrow_n}|_I) = \rho] \quad .
\end{aligned}$$

This equation defines the **feedback operation on a probabilistic behavior** f.

Example 4.3 We consider the simplest example, a strongly causal identity function over the Booleans $\mathbb{B}$, as it already provides an interesting insight into the effect of the feedback operation. In order to obtain a strongly causal function, we have to provide an initial state distribution which in turn determines the initial output distribution. For this example, it is not important which concrete distribution we choose, as long as it is non-trivial.

We decided on the uniform distribution which leads us to the following definition of the probabilistic I/O automaton $M_{\mathsf{Id}[x,y]} = (\mathbb{B}, \{x\}, \{y\}, \delta, \Delta_x, \omega_y)$ with the syntactic interface $\{x\} \overset{P}{\rhd} \{y\}$ and $\mathbf{type}(x) = \mathbf{type}(y) = \mathbb{B}$. The initial distribution and transition function are given by (for all $s, b \in \mathbb{B}$):

$$\begin{aligned}
\delta(s) &= {}^1\!/\!_2 & \omega_y(s) &= (y \mapsto s) \\
\Delta_x(s, (x \mapsto b)) &= (b \mapsto 1) \quad .
\end{aligned}$$

The left transition diagram of Figure 4.3 visualizes $M_{\mathsf{Id}[x,y]}$. Thereby, a node label $c!a$ describes the sending of a message a and a transition label $c?a$ describes the

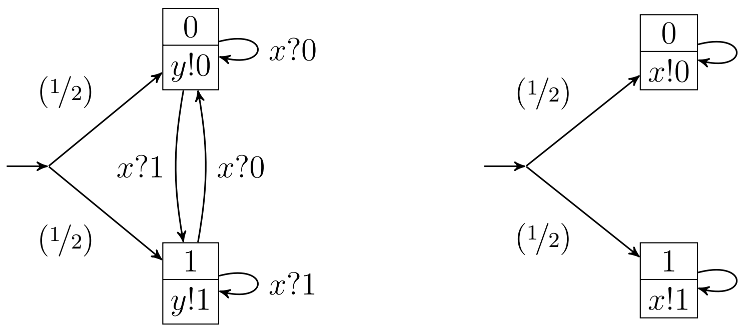


Figure 4.3: Visualization of automaton $M_{\mathsf{Id}[x,y]}$, a probabilistic identity function with initial uniform distribution, and its feedback $M^{\circlearrowleft}_{\mathsf{Id}[x,x]}$. Annotations (1) for probabilities of value one are omitted.

receiving of a message a, both on channel c. Probabilities are given in parentheses beneath each transition. As before, probabilities 1 are not annotated. The initial state distribution is annotated on arrows without source node.

The application of the feedback operator to $M_{\mathsf{Id}[x,x]}$ results in $M^{\circlearrowleft}_{\mathsf{Id}[x,x]} : \emptyset \overset{P}{\rhd} \{x\}$ as visualized on the right of Figure 4.3. Thereby, nothing changes except the transition function which is now characterized by (for all $s \in \mathbb{B}$)

$$\Delta^{\circlearrowleft}_x(s, (\,)) = \Delta_x(s, \omega_x(s)) = \Delta_x(s, (x \mapsto s)) = (s \mapsto 1) \quad .$$

The behavior abstraction resulting from this automaton is a simple uniform distribution over two sequences (omitting the channel identifier x):

$$\mathsf{Id}[x,x]^{\circlearrowleft}((\,)^\infty) = (0^\infty \mapsto 1/2, 1^\infty \mapsto 1/2) \quad .$$

However, if we provide any of these sequences to the original behavior function we obtain some additional outputs, e.g.,

$$\mathsf{Id}[x,x](0^\infty) = (0^\infty \mapsto 1/2, 10^\infty \mapsto 1/2) \quad .$$

This difference emphasizes, that the feedback operation indeed operates on a single instance of an automaton: the output of this instance is provided as input to the very same instance. Thereby, the automaton is not restarted for the feedback of each message but continues at its current state. We would obtain another behavior than determined by the feedback operator, if we would iteratively run the automaton for n transitions producing $n+1$ outputs and provide these $n+1$ outputs in the next iteration in which the automaton runs for $n+1$ transitions producing $n+2$ outputs and so on. In Example 4.6, we elaborate on this observation and compare it to unfold rules that are valid in the non-probabilistic case. □

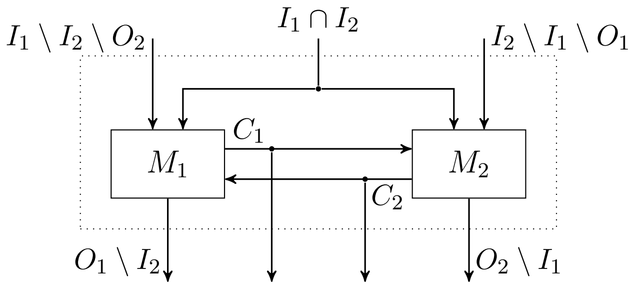


Figure 4.4: General composition of two systems without self-feedback, that means $I_1 \cap O_1 = \emptyset = I_2 \cap O_2$.

4.3.3 General Composition

In general, two arbitrary components with disjoint output channels ($O_1 \cap O_2 = \emptyset$) may be composed. Figure 4.4 illustrates the case where additionally $I_1 \cap O_1 = \emptyset$ as well as $I_2 \cap O_2 = \emptyset$ holds. The depicted components share some input channels $I_1 \cap I_2$ and use the channels $C_1 = I_2 \cap O_1$ and $C_2 = I_1 \cap O_2$ for communication.

The **general composition** $M_1 \otimes M_2$ of two probabilistic I/O automata, and analogously for two probabilistic behaviors, is the combination of parallel composition and the feedback operation:

$$M_1 \otimes M_2 = (M_1 \oplus M_2)^{\circlearrowleft}$$

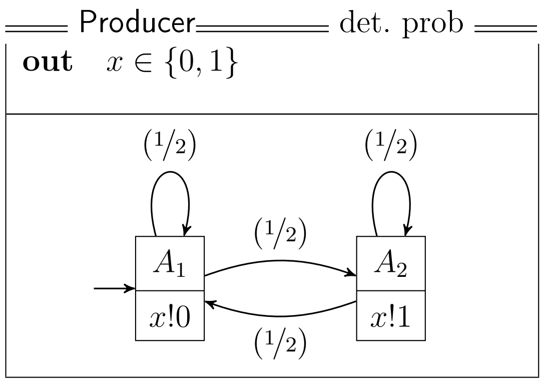
with the syntactic interface $I \overset{P}{\triangleright} O$ where $O = O_1 \cup O_2$ and $I = (I_1 \cup I_2) \setminus O$.

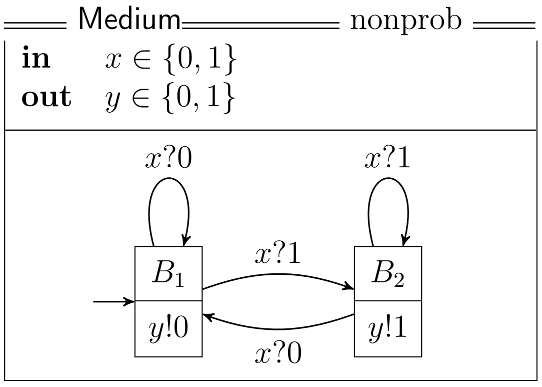
The parallel composition enables parallel operation of both automata whereas the feedback operation allows the components to exchange messages over shared channels.

Operational Semantics This composition defines an automaton $M_1 \otimes M_2 = (S, \overline{I}, \overline{O}, \delta, \Delta, \omega)$ with

$$\begin{aligned}
O &= O_1 \cup O_2 \\
I &= (I_1 \cup I_2) \setminus O \\
S &= S_1 \times S_2 \\
\delta(s_1, s_2) &= \delta_1(s_1) \cdot \delta_2(s_2) \\
\omega(s_1, s_2) &= \omega_1(s_1) \uplus \omega_2(s_2) \\
\Delta((s_1, s_2), i)(s_1', s_2') &= \Delta(s_1, i \uplus \omega(s_1, s_2)|_{I_1})(s_1') \cdot \Delta(s_2, i \uplus \omega(s_1, s_2)|_{I_2})(s_2') \quad .
\end{aligned}$$

Example 4.4 We illustrate the definition of composition with a small example. We define two components Producer and Medium by the means of automata – they are shown in separate frames labeled accordingly.





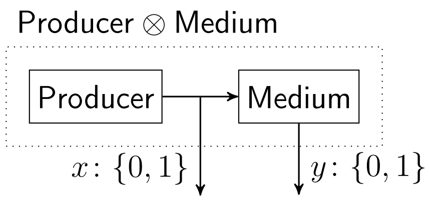


Figure 4.5: Component network of the composition.

The Producer has no input and outputs a message 0 or 1 with equal probability on channel x after the first time step. The Producer's output is connected to the Medium, which forwards all received messages with a delay of 1 to its output channel y. Formally, we have Producer $= (S_P, \bar{I}_P, \overline{O}_P, \delta_P, \Delta_P, \omega_P)$ and Medium $= (S_M, \bar{I}_M, \overline{O}_M, \delta_M, \Delta_M, \omega_M)$ with

$$\begin{aligned} S_P &= \{A_1, A_2\} & S_M &= \{B_1, B_2\} \\ \delta_P(A_1) &= 1 & \delta_M(B_1) &= 1 \\ \mathbf{type}(x) = \mathbf{type}(y) &= \{0,1\} & O_M &= \{y\} \\ I_P &= \emptyset & O_P = I_M &= \{x\} \end{aligned}$$

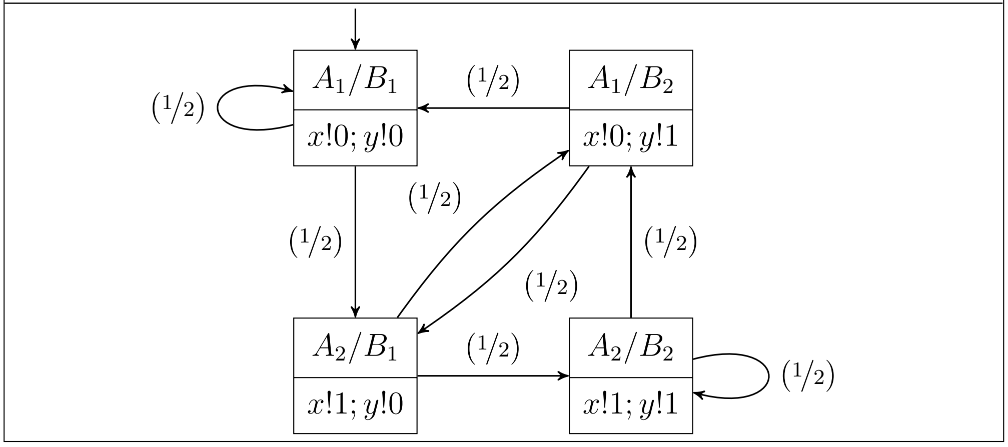
and for every $i \in \{1,2\}$ it is

$$\begin{aligned} \omega_P(A_1) &= (x \mapsto 0) & \omega_M(B_1) &= (y \mapsto 0) \\ \omega_P(A_2) &= (x \mapsto 1) & \omega_M(B_2) &= (y \mapsto 1) \\ \Delta_P(A_1, ())(A_i) &= 1/2 & \Delta_M(B_i, (x \mapsto 0))(B_1) &= 1 \\ & & \Delta_M(B_i, (x \mapsto 1))(B_2) &= 1 \quad . \end{aligned}$$

Figure 4.5 illustrates the composition with syntactic interface $\emptyset \overset{P}{\triangleright} \{x, y\}$, which means that the composition has no input channels and the shared channel x is not hidden from but forwarded to the outside. The behavior of the composition is given explicitly in the specification frame below.

Producer $\otimes$ Medium det. prob

out $x, y \in \{0,1\}$



For example, the probability that the composite automaton transits from (A_1, B_1) to (A_2, B_1) can be calculated by:

$$\begin{aligned}
&\Delta((A_1, B_1), ())(A_2, B_1)\\
&\quad = \Delta_P(A_1, ())(A_2) \cdot \Delta_M(B_1, (x \mapsto 0))(B_1)\\
&\quad = \frac{1}{2} \cdot 1 \quad .
\end{aligned}$$

□

Behavior Abstraction We denote with f_1 and f_2 the behavior abstraction of M_1 and M_2, respectively. By combining our previous results, we obtain

$$\begin{aligned}
\mathbf{P}[(f_1 \otimes f_2)(i) = \rho] &= \mathbf{P}[(f_1 \obar f_2)(i \uplus \rho{\downarrow_n}|_I) = \rho]\\
&= \prod_{l \in \{1,2\}} \mathbf{P}[f_l(i \uplus \rho{\downarrow_n}|_{I_l}) = \rho|_{O_l}] \quad .
\end{aligned}$$

As expected, the right side of the equation depends only on the behavior abstraction of the two I/O automata and the equation provides the characterization of general composition in terms of probabilistic behaviors.

4.3.4 Projection

In order to restrict the behavior of a component on a selection of output channels, we introduce the **projection operator** †. This operator is also essential to realize *information hiding* regarding channels: communication channels can be made internal by projection on the public channels. Then, other components cannot access the internal channels from the outside anymore.

Operational Semantics Let M be an automaton as before and $O' \subseteq O$ a subset of channels we want to project onto. The resulting syntactic interface is $I \overset{P}{\rhd} O'$. In order to hide some channels H, we have to select $O' = O \setminus H$. Projecting the automaton M onto the channels O' yields the automaton $M \dagger O' = (S, \bar{I}, \overline{O'}, \delta, \Delta, \omega')$ with

$$\omega'(s) = \omega(s)|_{O'} \quad .$$

Behavior Abstraction The probability space over the state space S is not changed by this operation, that means the underlying sequence space and the probability measures induced by M and $M \dagger O'$ are the same. We only have to review the behavior abstraction $f \dagger O'$ of the new automaton and compare it to the behavior abstraction f of M. We have

$$\begin{aligned}(f \dagger O')(i) &= \omega'((M \dagger O')(i)) \\ &= \omega'(M(i)) \\ &= \omega(M(i))|_{O'} \\ &= f(i)|_{O'} \quad .\end{aligned}$$

As before, the output functions ω and ω' are applied element-wise to state sequences. This equation (of random variables) is legal as $M \dagger O'$ is embedded within the probability space of the original automaton and so these are dependent variables.

Note that since we talk about random variables over sequences of valuations, we mean the projection in $f(i)|_{O'}$ to be applied element-wise to the output sequence yielded by $f(i)$. We could alternatively characterize the projection operation $f \dagger O'$ directly in terms of f, but that is less readable:

$$(f \dagger O')(i)(\mathcal{C}(\rho)) = f(i)\Big(\bigcup_{\varphi|_{O'}=\rho} \mathcal{C}(\varphi) \Big) \quad .$$

Example 4.5 We illustrate the idea of this new operator by hiding the channel x from the composition ($\mathsf{Producer} \otimes \mathsf{Medium}$), say because x was meant for internal communication only. Hiding can be easily applied to I/O automata by removing the output of the hidden channels. The resulting automaton is the same as shown in Example 4.1. There, we saw already the minimal equivalent automaton with three states. For convenience, we repeat this automaton (see frame ($\mathsf{Producer} \otimes \mathsf{Medium}) \dagger \{y\}$) and annotate which states were merged.

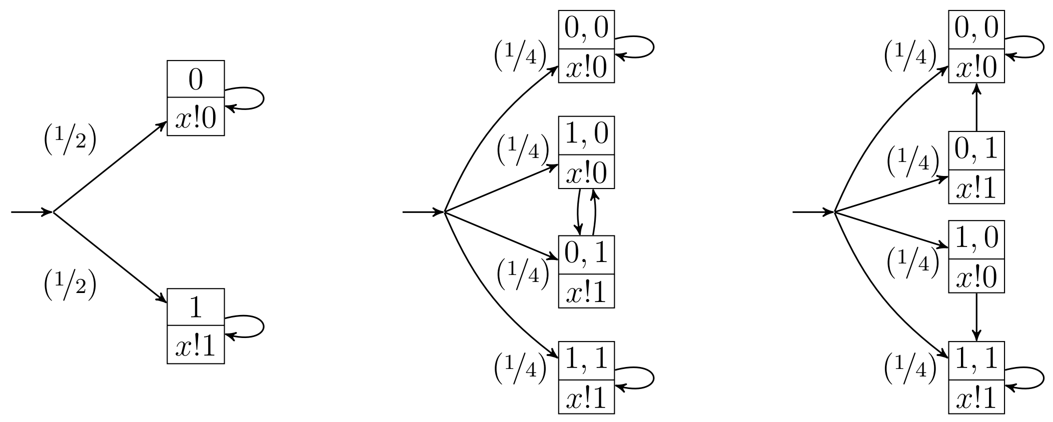
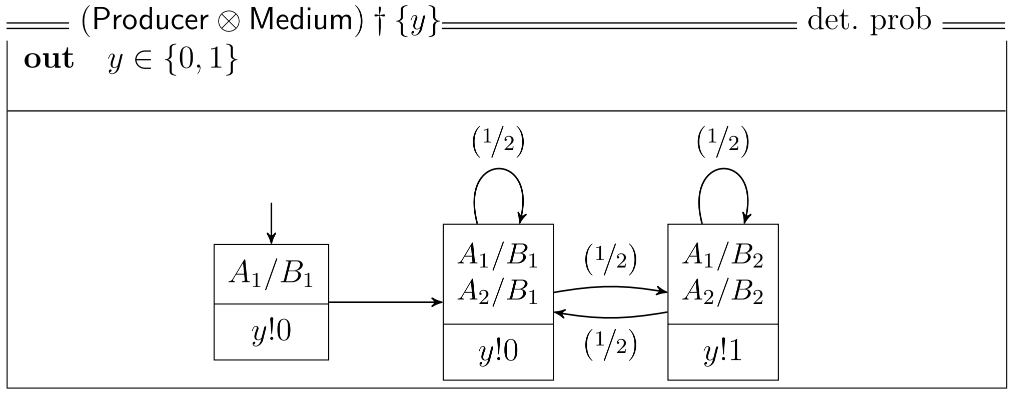


Figure 4.6: Automaton $M^{\circlearrowleft}_{\mathsf{Id}[x,x]}$ on the left and its two unfoldings: Eq. (4.4) in the middle and (4.5) on the right. Comparison shows that unfolding does not preserve probabilistic behavior. Probabilities of value one are not annotated.



□

Example 4.6 Let us contrast the observations of Example 4.3 to the non-probabilistic feedback operation of FOCUS. In [Bro97], M. Broy elaborates on different unfolding rules of feedback operations. Applying these rules to our example analogously, we would expect the two unfoldings

$$\mathsf{Id}[x,x]^{\circlearrowleft} = (\mathsf{Id}[x,y] \otimes \mathsf{Id}[y,x]) \dagger \{x\} \tag{4.4}$$

$$\mathsf{Id}[x,x]^{\circlearrowleft} = (\mathsf{Id}[y,y]^{\circlearrowleft} \otimes \mathsf{Id}[y,x]) \dagger \{x\} \quad . \tag{4.5}$$

However, both unfoldings are not valid in the probabilistic case. By evaluating the right-hand sides of the equations, we obtain the behaviors shown in the middle and right of Figure 4.6. Obviously, these are not behaviorally equivalent to $\mathsf{Id}[x,x]^{\circlearrowleft}$ which is shown on the left in Figure 4.6. □

4.4 Special Case: Non-Probabilistic Behavior

Every non-probabilistic automaton and behavior has a counterpart in the probabilistic formalism. To proof this, we will, at first, explain the translation of the FOCUS components (deterministic automaton and behavior) into their probabilistic counterpart. Then, we show that this translation is a homomorphism with respect to composition.

4.4.1 Automata

For the following definition, we use the indicator function

$$\begin{aligned}&[\cdot]\colon \mathbb{B} \to [0,1]\\ &q \in \mathbb{B} \mapsto [q] = \begin{cases}1 & \text{if } q \text{ is true}\\ 0 & \text{otherwise}\end{cases} \quad .\end{aligned}$$

The **probabilistic translation** of the deterministic non-probabilistic I/O automaton $M = (S, I, O, \sigma_0, \Delta, \omega)$ is given by $M^P = (S, I, O, \delta, \Delta^P, \omega)$, where

$$\begin{aligned}\delta(s) &= [s = \sigma_0]\\ \Delta^P(s,i)(o) &= [\Delta(s,i) = o] \quad .\end{aligned}$$

4.4.2 Behavior

Analogously to the probabilistic behavior abstraction, we have the non-probabilistic abstraction $f_M\colon I^\infty \to O^\infty$ of some non-probabilistic, deterministic automaton M. This function is defined as the output produced from the unique state sequence that M traverses for a given input sequence i. If $run_M(i)$ denotes this state sequence and $\omega\colon S \to O$ denotes the output function, then we have $f_M(i) = \omega(run_M(i))$. The probabilistic behavior of M^P is very simple. For all finite state sequences $s = \langle s_0, \dots, s_n\rangle$ and infinite input sequences i, it follows that

$$\mathbf{P}[M^P(i) \sqsupseteq s] = \delta(s_0) \cdot \prod_{k=0}^{n-1} \Delta^P(s_k, i_k)(s_{k+1})$$

equals either zero or one because the product of only zeros and ones is again either zero or one. For the latter case, we have the equivalences:

$$\begin{aligned}&\mathbf{P}[M^P(i) \sqsupseteq s] = 1\\ \iff\ & s_0 = \sigma_0 \ \wedge\ \forall k \in \{0, \dots, n-1\}\colon \Delta(s_k, i_k) = s_{k+1}\\ \iff\ & run_M(i) = s \qquad \text{i.e., } s \text{ is the run of } M \text{ on } i.\end{aligned}$$

Applying these facts to the definition of the behavior abstraction f_{M^P} of M^P yields:

$$\begin{aligned}\mathbf{P}[f_{M^P}(i) \sqsupseteq \rho] &= \mathbf{P}[\omega(M^P(i)) \sqsupseteq \rho] \\ &= [\omega(run_M(i)) = \rho] \quad ,\end{aligned}$$

which equals one if, and only if, $\rho = \omega(run_M(i)) = f_M(i)$ and otherwise equals zero. We can now characterize the probabilistic behavior f_{M^P} compactly:

$$\forall i \in I^\infty, \rho \in O^*: \quad \mathbf{P}[f_{M^P}(i) \sqsupseteq \rho] = [f_M(i) \sqsupseteq \rho] \quad .$$

We reuse this characterization as a definition and obtain the **probabilistic translation** f^P of a deterministic behavior function f:

$$\mathbf{P}[f^P(i) \sqsupseteq \rho] = [f(i) \sqsupseteq \rho] \quad .$$

This immediately leads to the commutability of behavior abstraction and probabilistic translation:

Theorem 4 $f_{M^P} = {f_M}^P$. □

The probabilistic translation allows us to reuse descriptions of non-probabilistic components from Focus. To make this usage explicit, we introduce a special specification frame **nonprob** (cf. Section 5.1). The specification Medium of Example 4.4 can be understood as such a translation. In the subsequent section, we will see further applications of this translation.

Theorem 5 *Given some deterministic and non-probabilistic behaviors f and g, we have*

$$\begin{aligned}(f^P \oplus g^P) &= (f \oplus g)^P \\ (f^P)^{\circlearrowleft} &= (f^{\circlearrowleft})^P \\ (f^P \otimes g^P) &= (f \otimes g)^P \quad .\end{aligned}$$

This shows, that the probabilistic translation $\cdot^P$ is a homomorphism from deterministic Focus *behavior functions to (deterministic) probabilistic behavior functions with respect to the three operators parallel composition $\oplus$, feedback $\cdot^{\circlearrowleft}$ and general composition $\otimes$.* □

Proof The proofs are straightforward. We have, for all $i \in I^\infty$ and $o \in O^*$ with $I = I_f \cup I_g$, $O = O_f \cup O_g$ and $i_f = i|_{I_f}$, $o_f = o|_{O_f}$ etc.,

$$\begin{aligned}\mathbf{P}[o \sqsubseteq (f^P \oplus g^P)(i)] &= \mathbf{P}[o_f \sqsubseteq f^P(i_f)] \cdot \mathbf{P}[o_g \sqsubseteq g^P(i_g)] \\ &= [o_f \sqsubseteq f(i_f) \ \wedge \ o_g \sqsubseteq g(i_g)] \\ &= [o \sqsubseteq (f \oplus g)(i)] \\ &= \mathbf{P}[o \sqsubseteq (f \oplus g)^P(i)] \quad .\end{aligned}$$

This proves the first equation. The second equation follows similarly. We have with $I'_f = I_f \setminus O_f$ and $i'_f \in I'^{\infty}_f$

$$\begin{aligned}
\mathbf{P}[o_f \sqsubseteq (f^P)^{\circlearrowleft}(i'_f)] &= \mathbf{P}[o_f \sqsubseteq f^P(i'_f \uplus o_f|_{I_f})] \\
&= [o_f \sqsubseteq f(i'_f \uplus o_f|_{I_f})] \\
&= [o_f \sqsubseteq f^{\circlearrowleft}(i'_f)] \\
&= \mathbf{P}[o_f \sqsubseteq (f^{\circlearrowleft})^P(i'_f)] \quad .
\end{aligned}$$

By definition of the general composition it follows as well

$$(f^P \otimes g^P) = (f^P \oplus g^P)^{\circlearrowleft} = ((f \oplus g)^P)^{\circlearrowleft} = ((f \oplus g)^{\circlearrowleft})^P = (f \otimes g)^P \quad . \qquad \blacksquare$$

4.5 Example: Alternating Bit Protocol

In this section, we discuss a probabilistic derivative of the Alternating Bit Protocol (ABP).

The Sender and Receiver are connected by two instances of LossyMedium, all of which are specified in the following frames. All but the LossyMedium component are non-probabilistic, they are specified using Focus and translated into their probabilistic correspondent as indicated by the **nonprob** frame annotation. In contrast to previously specified media, the LossyMedium forwards the current message only with probability p and drops it otherwise. All channels in this example allow the communication of a stub message (usually called *nothing, null* or *nil*) indicating that there is nothing interesting to be sent. We will use the symbol $\square$ to identify this kind of message and introduce the abbreviation $T^{-} = T \cup \{\square\}$ for any type T.

Additionally to the Sender, Medium and Receiver components, we introduce a Producer component which provides our communication system with input. In reaction to each `ok` message received on channel `y`, this Producer has to output the next message to transmit. For this study, we choose the simple Producer that outputs a constant stream of `'a'` messages. Thus, we can instantiate all components with type parameter T set to the singleton set $\{\mathtt{a}\}$.

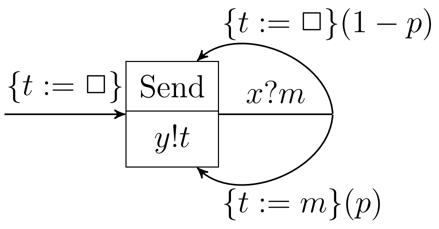
Figure 4.7 shows how the components are integrated to form the overall communication system. The specification of LossyMedium refers to the channels x and y, which we have to rename to fit the channels of the Sender and Receiver. We use the notation $C[^x/_y]$ to denote the component that we obtain by renaming channel x as channel y in the specification of component C. The probability p that a medium forwards a message successfully is set to $^3/_4$ for this example.

From the composition we hide all output channels but channel `z`. This allows us to easily verify the correct transmission of messages output by Producer at `x` by

LossyMedium[**type** $T, p \in [0,1]$] det. prob

in $x \in T^-$
out $y \in T^-$

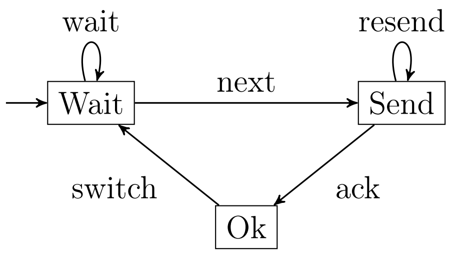
local $t \in T^-$



Sender[**type** T] nonprob.

in $x \in T^-, c_4 \in \mathbb{B}^-$
out $y \in \{\texttt{ok}\}^-, c_1 \in (T \times \mathbb{B})^-$

local $bs \in \mathbb{B}, d \in T^-$
univ $m \in T$
state $s \in \{\text{Wait}, \text{Send}, \text{Ok}\}$



init $bs.0 = \text{false} \wedge d.0 = \Box$

s	y	c_1
Send	$\Box$	(d, bs)
Ok	ok	$\Box$
Wait	$\Box$	$\Box$

	x	c_4	d'	bs'
wait	$\Box$	–	$\Box$	bs
next	m	–	m	bs
resend	–	$\Box \vee \neg bs$	d	bs
ack	–	bs	$\Box$	bs
switch	–	–	$\Box$	$\neg bs$

Receiver[**type** T] nonprob.

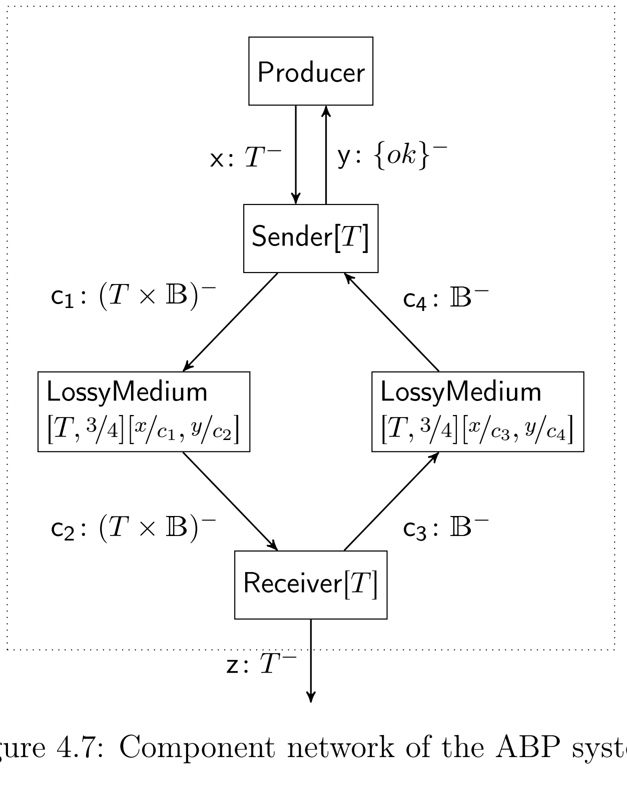
in $c_2 \in \mathbb{B}^-$
out $z \in T^-, c_3 \in \mathbb{B}^-$

local $br \in \mathbb{B}$
univ $m \in T$

init $br.0 = \text{false}$

c_2	br'	z'
(m, br)	$\neg br$	m
$(-, \neg br) \vee \square$	br	$\square$

c_3
$\neg br$



Producer
x: T^-
y: $\{ok\}^-$
Sender[T]
c$_1$: $(T \times \mathbb{B})^-$
c$_4$: $\mathbb{B}^-$
LossyMedium $[T, 3/4][x/c_1, y/c_2]$
LossyMedium $[T, 3/4][x/c_3, y/c_4]$
c$_2$: $(T \times \mathbb{B})^-$
c$_3$: $\mathbb{B}^-$
Receiver[T]
z: T^-

Figure 4.7: Component network of the ABP system.

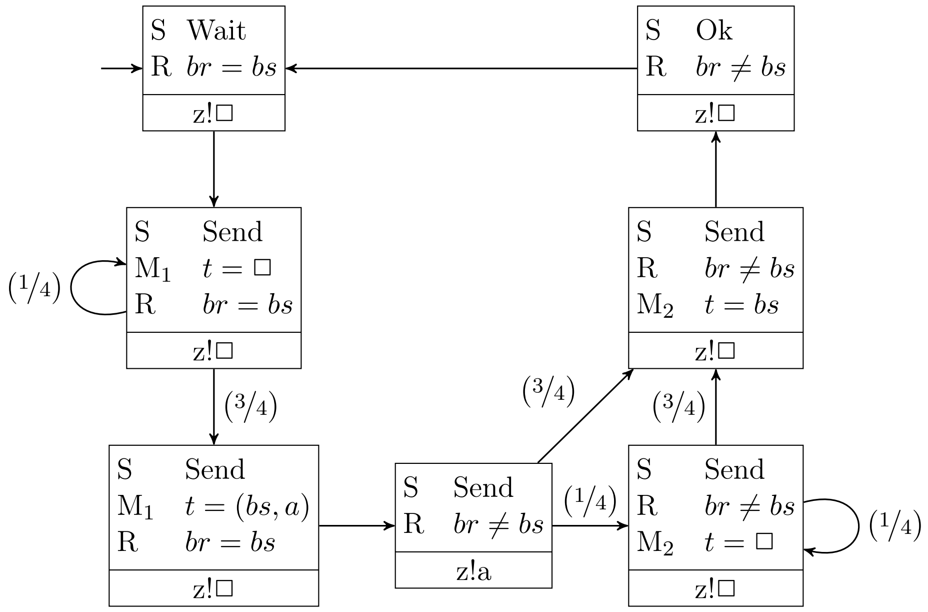


Figure 4.8: Reduced automaton of the composed ABP system.

analyzing the output behavior at z. Figure 4.8 illustrates the automaton of the composition of the five components.

Without optimization, the resulting automaton would have 36 reachable states. Therefore, we show a reduced but behaviorally equivalent automaton which combines several states of equivalent behavior into a single state. The diagram reflects this reduction as the label of each node indicates the original components' states. We use the abbreviations S, M_1, R and M_2 respectively for Sender, Medium from Sender, Receiver, and Medium from Receiver. The reduced automaton has only 7 instead of 36 states. For example, the left node in the second row combines all states where the Sender is in state Send, the first Medium stores $t = \square$ (i.e., it does not forward the previously received message) and the parity bits of Sender and Receiver are equal. This indicates that the output behavior of the composition is independent of the current value of the parity bit as long as the local bits of the Sender and Receiver are consistent.

We will now discuss some properties of the composition's automaton. The automaton consists of a single class, meaning that from every state any other state can be reached. This class is finite and *regular* (or *aperiodic*) as it contains a self-loop (a state with a transition to itself). Thus, the automaton (or the underlying Markov chain) is *ergodic*[5].

[5]A set of states in a Markov Chain is ergodic if all its states are ergodic. A state is ergodic if it is aperiodic and positive recurrent. A state has period k if starting in the state a return occurs

From this categorization, we can derive several other properties:

- Every state is visited infinitely often almost surely. This is an important observation as one of the states (middle column with output "z!a") constitutes the forwarding of the original message `'a'` on channel `z`. Thus, every time this state is reached the protocol successfully transmits a message. As this state is reached infinitely often with probability one, this system fulfills the property of a correct transmission protocol with probability one. Formally, this property can be described by:

$$\mathbf{P}[\text{count}(\{\mathtt{a}\}, \mathtt{z}) = \infty] = 1$$

  where `z` refers to the random variable for channel `z` and $\text{count}(X, \varphi)$ is the number of times an element of X occurs in the sequence φ (cf. specification of lossy medium in [BS01]).

- After sufficient time has elapsed the process can be in any state at any time, i.e., there is no periodic behavior.

- The probability to find the process in a specific state converges, i.e., for every state s the limit

$$\pi(s) = \lim_{n\to\infty} \mathbf{P}[M.n = s]$$

  exists, where M denotes the random variable describing the composite automaton's state sequence. This limit distribution is independent of the initial distribution.

  We can calculate the actual distribution either by solving a linear equation system or by relying on the probabilistic model checker Prism. In the context of the latter, this distribution is called *steady-state probabilities.* By enumerating the states as shown in Figure 4.8 in counterclockwise order beginning with the initial state, we obtain the following values for π:

s	0	1	2	3	4	5	6
$\pi(s)$	0.15	0.2	0.15	0.15	0.05	0.15	0.15

  The state distribution for the first 100 time steps is shown in Figure 4.9. At first, the distribution seems periodic but it converges clearly to the limits $\pi(s)$.

only in multiples of k steps. It is aperiodic if its period is 1, thus returns can occur at irregular times. A state is positive recurrent if the expected return time is finite. [KSK76]

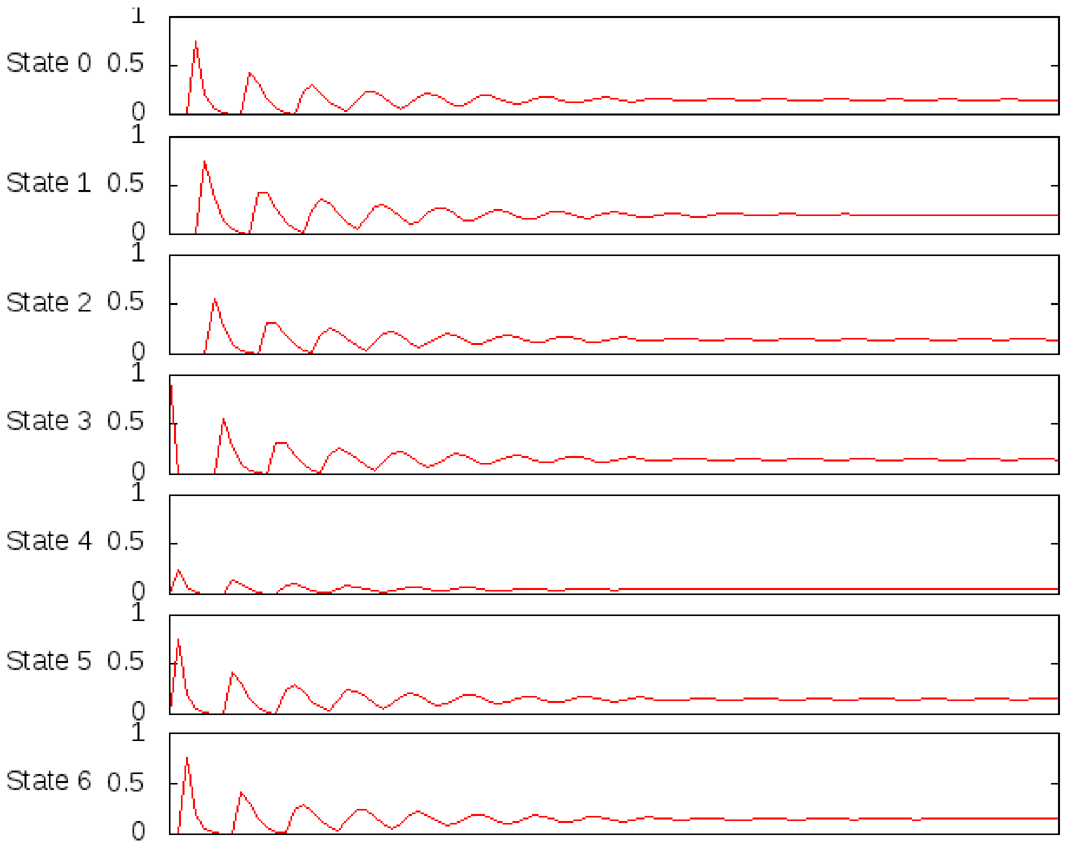


Figure 4.9: State distribution for the first 100 steps.

Next, we determine the expected number of steps between two consecutive message outputs on channel z. Again, PRISM provides the required facility to calculate this value. State 3 is the only state where the automaton outputs a symbol on z. So we have to measure the expected time needed from State 3 to State 3, i.e., one round trip. This way, we would get the result zero steps from PRISM. Instead, we have to measure the steps from State 3 to State 2 and add one to the result. In PRISM, we introduce a "time" reward and set State 3 as the initial state. Then, the property specification

$$R_{\{"time"\}=?}[F(s=2)]+1$$

describes the expected round trip time and resolves to $6\frac{2}{3}$ and corresponds to the reciprocal of the steady-state probability 0.15 of State 3.

Chapter 5

Description Techniques for Probabilistic Specifications

In this chapter, we present description techniques for probabilistic specifications. Such techniques are needed to actually write down specifications. The most basic description technique is the mathematical notation based on predicate logic, which is used throughout this thesis to define the foundations of our modeling theory. Although, by using higher-order logic we are in principle able to specify anything, this general form can be rather cumbersome for the specification of certain systems. According to the principle “the right tool for the right job”, we present additional notations and thereby increase the class of specifications that we can write down in a comprehensive and compact form.

The FOCUS theory, on which this work is based, already encompasses several description techniques for non-probabilistic specifications. The contribution of this chapter is to extend some of these notations for the probabilistic case. We remain compatible to FOCUS in the way that we allow reusing FOCUS specifications and that the extended notations are kept as similar as possible to the original ones.

The description techniques that we are going to introduce form a syntactic framework with sound semantics which can be used for unambiguous specification documents.

5.1 Introduction

This section provides an overview and the common principles of the description techniques to be introduced in the remaining chapter. The different techniques are listed in Table 5.1 where we distinguish elementary and composite specifications. The latter describe components by means of decomposition, whereas the former describe the components directly. That means, in an architecture of com-

	Tabular	Graphical	Logical
Elementary	Logical Tables	Transition Diagrams	Predicate and Temporal Logics
Composite	n/a	Component and Dependency Diagrams	Textual Composition

Table 5.1: Classification of the presented description techniques. We distinguish between composed and elementary specifications and different visual styles.

ponents the inner components are described by composite specifications and the leaves of the hierarchy are described by elementary specifications. In particular, a composite specification allows the usage of different description techniques for the specification of its sub-components.

Description techniques can also be classified according to their visual **style**. According to the Focus theory, we distinguish the *graphical*, *tabular* and *logical* style. We will provide a probabilistic extension to at least one description technique of each style.

Graphical Style A graph is a set of vertices connected by edges.

For an elementary specification, we will visualize I/O automata graphically. Then vertices represent states and edges represent transitions between states. Such graphs are called *transition diagrams.*

For composite specifications, we connect components with channels. Such graphs are called *component networks.*

Tabular Style Tables are useful to layout relations. For elementary component specifications, each column can be associated to a certain channel and every row defines one entry or rule of the input/output-relation of the component's behavior. Such tables are known as *logical tables* from the Focus theory. We will extend this kind of tables to incorporate probabilities.

Logical Style We can textually specify components using formulas written in a certain logic. Using predicate logic and the definitions given throughout this thesis, we can also specify concrete components.

For elementary specifications, we have to refer to the component's behavior function. Several examples for such specifications were given in the preceding chapters. For composite specifications, we can rely on the composition operators introduced in Sections 4.3, 6.3 and 6.4.

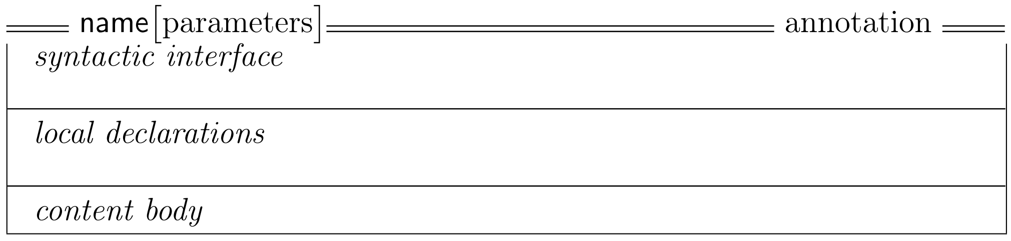


Figure 5.1: The general structure of a specification frame. Frames visually distinguish the scope of a specification.

We will not provide any further description techniques of the logical style. However, several probabilistic temporal logics exist, which could be combined with our modeling theory.

In the Focus theory, specification **frames** are used to visualize the scope of a specification. Each frame is dedicated to one description technique and the specification of a single component. The general pattern of a frame is given in Figure 5.1. We distinguish the following parts of a frame:

- *name* provides the identifier of the specified component. If the specification is parametric, *name* is followed by the formal parameters in square brackets.
- The definition of the *syntactic interface* is an integral part of elementary specifications but may be omitted for composite specifications if it is clear from the composition itself.

  The declarations of input and output channels are introduced with the keywords **in** and **out**, respectively. Each channel is declared by its name and the type of messages communicated over this channel. For example, the declaration

  $$\begin{gathered}\textbf{in } i_1 \in I_1, \ldots, i_k \in I_k \\ \textbf{out } o_1 \in O_1, \ldots, o_n \in O_n\end{gathered}$$

  denotes the syntactic interface $\{i_1, \ldots, i_k\} \rhd \{o_1, \ldots o_n\}$ with the types:

  $$\begin{gathered}\textbf{type}(i_1) = I_1, \ldots, \textbf{type}(i_k) = I_k \\ \text{and } \textbf{type}(o_1) = O_1, \ldots, \textbf{type}(o_n) = O_n \quad .\end{gathered}$$

- The next part of the frame, *local declarations*, is optional and used to declare variables which are not part of the component's interface.

  The syntax is the same as for the declaration of the syntactic interface. However, we use other keywords depending on the description technique.

For example, **local** is used for local state variables and **univ** for universally qualified variables in tabular specifications.

- The *content body* defines the behavior using one of the available description techniques.

- We use the *annotation* field for two purposes.

  An annotation may be given to point out a property of the component that is specified. For example, we will highlight that a specified component is deterministic and probabilistic using the annotation **det. prob** or that it is nondeterministic and probabilistic using **nondet. prob**.

  An annotation can further constraint or influence the specification. For example, we may mark a component network with the *independent* annotation to require statistically independent sub-components' (cf. Section 5.4 Composite Specifications).

In order to reuse any description technique for non-probabilistic components from the FOCUS theory, we introduce two particular annotations that determine the interpretation of such non-probabilistic descriptions within our probabilistic theory:

(a.s.) With this annotation, a nondeterministic non-probabilistic specification is interpreted as a nondeterministic probabilistic one according to the probabilistic translation introduced in Chapter 6 (there denoted as F^P).

The meaning of this translation is: the non-probabilistic specification has to be fulfilled almost surely (a.s.), i.e., with probability one.

(nonprob) With this annotation, a deterministic non-probabilistic specification is interpreted as a deterministic probabilistic one. The resulting behavior contains only trivial output distributions with the probabilities one and zero. This is a special case of the previous translation and the simpler definition of the probabilistic translation from Section 4.4 is applicable.

5.2 Tabular Specifications

A tabular probabilistic specification follows the pattern shown in Figure 5.2. We distinguish **init tables** and **logical tables**. The latter are an extension of the logical tables of the FOCUS theory to additionally capture probabilities. These tables determine values of variables depending on the values of other observable variables, usually of the previous time step. In order to obtain a strongly causal behavior, the initial output distribution has to be defined independent of the input name[parameters] det. prob

in	$i_1 \in I_1, \ldots$
out	$o_1 \in O_1, \ldots$
local	$l_1 \in L_1, \ldots$
univ	$u_1 \in U_1, \ldots$

init

Outputs			*Probability*
S_1	$\ldots$	S_l	Prob
$\vdots$			
$s_{1,k}$	$\ldots$	$v_{l,k}$	p_k
$\vdots$			

$\ldots$ *further init tables* $\ldots$

Precondition	*Inputs*			*Outputs*			*Probability*
Cond	V_1	$\ldots$	V_n	W_1	$\ldots$	W_m	Prob
$\vdots$							
c_k	$v_{1,k}$	$\ldots$	$v_{n,k}$	$w_{1,k}$	$\ldots$	$w_{m,k}$	p_k
$\vdots$							

$\ldots$ *further logical tables* $\ldots$

Figure 5.2: The general pattern of tabular probabilistic specifications with *init tables* and *logical tables*.

values. Therefore if necessary, we use **init tables** to specify the initial distribution of output channels and local variables.

For any given point in time, the k-th row in a logical table as shown in Figure 5.2 means: If the current input values match the input pattern (given by the values $v_{1,k} \dots v_{n,k}$ and the precondition c_k) on the left hand side of the double line, then this row *matches* and the values $(w_{1,k} \dots w_{m,k})$ on the right hand side are assigned to the outputs with the specified probability (p_k). Thereby, the columns V_1 to V_n are a subset of the input channels and local variables. The columns W_1 to W_m on the right hand side are a subset of the output channels and local variables. Additionally, we may refer to variables of the subsequent time step by using primed variable identifiers (e.g., W' instead of W). At any time, the sum over the probabilities of all matching rows of a table has to be equal to one and at least one row of each table must be matching. Otherwise, the specification is incorrect.

In general, a specification contains several tables. Given a selection of matching rows, one for each table, the overall probability that the behavior proceeds according to this selection is given by the product of the row probabilities. That means the rows are selected statistically independent.

Additionally to input, output and local variables, universal variables can be declared. If we refer to such a variable within a row, this variable will be universally quantified over this row. Using universal variables, we can formulate equalities of different variables and reference variables from within the precondition.

5.2.1 Syntax

In the following, we will define the syntax of a tabular probabilistic specification. The syntax is based on the frame template as explained in the introduction of this chapter and extended as shown in Figure 5.2.

Variables The syntactic interface is declared as stated in the introduction, Section 5.1. The local declaration part, however, is extended by the keywords **local** and **univ**. These introduce **local state variables** and **universally quantified variables**, respectively:

$$\begin{array}{l} \textbf{local}\ l_1 \in L_1, \dots, l_k \in L_k \\ \textbf{univ}\ u_1 \in U_1, \dots, u_n \in U_n \quad . \end{array}$$

The declared typing has to be respected within the whole specification.

We define the set of all local variables $L = \{l_1, \dots, l_k\}$ and the set of all universal variables $U = \{u_1, \dots, u_n\}$. Given the syntactic interface and local declaration,

S_1	$\ldots$	S_l	Prob	
$\vdots$				
$s_{1,k}$	$\ldots$	$s_{l,k}$	p_k	$\leftarrow$ k-th row
$\vdots$				

Figure 5.3: The general form of an *init table*. The column headers $S_1, \ldots, S_l$ refer to controlled variables.

we define also the sets of **readable** (R) and **controlled** (C) variables:

$$\begin{aligned} R &= I \cup O \cup L \\ C &= O \cup L \end{aligned}$$

where I and O contain all identifiers of the input and output channels, respectively.

In Chapter 4, we characterized a probabilistic behavior as a mapping from infinite input sequences to a distribution of output sequences. For a specific infinite run of a system, we can assign to each input, local and output variable an infinite sequence of values – ignoring probabilities for a moment.

The rows of a table represent rules that define this temporal behavior of a component. The rules are meant to be applied for each time step. Therefore, each variable referenced by a table represents not a sequence of values but the value of the variable at the current point in time.

In order to allow rules to determine values of the next time step, we introduce primed variables. A **primed variable** v' denotes the value of the variable v at the next time step. Formally, v' is a new variable identifier. For example, given the output channel speed, within a table we refer to the current value of this output channel by speed and to the value at the next time step by speed′. We use the same notation to obtain sets and valuations of primed variables:

$$\begin{aligned} V' &= \{v' \mid v \in V\} \\ (v_1 \mapsto x_1, \ldots, v_k \mapsto x_k)' &= (v_1' \mapsto x_1, \ldots, v_k' \mapsto x_k) \quad . \end{aligned}$$

Init Tables These tables determine the initial distribution of the controlled variables, i.e., both local and output variables. The general form is shown in Figure 5.3. Thereby, the column headers $S_1, \ldots, S_l$ refer to controlled variables. Each variable may occur at most once in a table:

$$\begin{aligned} &\mathit{VInit} = \{S_1, \ldots, S_l\} \subseteq C \\ &\wedge \forall i, j \in \{1, \ldots, l\} \colon S_i \neq S_j \quad . \end{aligned}$$

Cond	V_1	$\dots$	V_n	W_1	$\dots$	W_m	Prob	
$\vdots$								
c_k	$v_{1,k}$	$\dots$	$v_{n,k}$	$w_{1,k}$	$\dots$	$w_{m,k}$	p_k	$\leftarrow$ k-th row
$\vdots$								

Figure 5.4: The general form of a *logical table*. The column headers $V_1, \dots, V_n$ refer to readable (input and local) variables, the headers $W_1, \dots, W_m$ refer to controllable (output and local) variables.

The entries $s_{i,k}$ are expressions that evaluate to values according to the type of S_i and are not allowed to have any free variables. The values p_k denote probabilities and have to be real numbers from the interval $[0, 1]$.

Let $\mathcal{I}$ be the set of all init tables. We require their column headers to be disjoint:

$$\forall \tau, \lambda \in \mathcal{I} \colon \; \mathit{VInit}_\tau \cap \mathit{VInit}_\lambda = \emptyset \quad .$$

Logical Tables Figure 5.4 shows the general form of a logical table. Each row consists of a condition c_k, the input patterns $v_{1,k}, \dots, v_{n,k}$, the output values $w_{1,k}, \dots, w_{m,k}$ and a probability p_k. The entries on the left hand side of the vertical double line form the **precondition**. Accordingly we call the set of identifiers in the header $\mathit{VPre} = \{V_1, \dots, V_n\}$. Respectively, the right hand side of the vertical double line forms the **postcondition** and we call the set of identifiers $\mathit{VPost} = \{W_1, \dots, W_n\}$. Note that each variable may occur at most once in each condition:

$$\begin{aligned} &\forall i, j \in \{1, \dots, n\} \colon V_i \neq V_j \\ \wedge\; &\forall i, j \in \{1, \dots, m\} \colon W_i \neq W_j \quad . \end{aligned}$$

We require that the precondition variables are readable whereas postcondition variables are controlled and can be primed or unprimed. In particular, they have to be disjoined. Thus, we have

$$\begin{aligned} &\mathit{VPre} \subseteq R \\ \wedge\; &\mathit{VPost} \subseteq C \cup C' \\ \wedge\; &\mathit{VPre} \cap \mathit{VPost} = \emptyset \quad . \end{aligned}$$

The cells $v_{i,k}$ and $w_{j,k}$ have to be expressions whose result type corresponds to the variables specified in their column header. Note that priming yields a variable

i	q	o'	Prob
$\vdots$		$\vdots$	

$$\Rightarrow$$

$$\begin{tikzcd}
i@0 \arrow[d] \arrow[ddr] & i@1 \arrow[d] \arrow[ddr] & i@2 \arrow[d] \arrow[ddr] & \cdots \\
q@0 & q@1 & q@2 & \cdots \\
o@0 & o@1 & o@2 & \ldots
\end{tikzcd}$$

Figure 5.5: An example to illustrate the meaning of column headers. The column headers of the logical table on the left induce the dependencies between time-indexed variables as shown on the right.

of the same type as the original variable. The condition c_k is a predicate and thus must evaluate to a Boolean value.

The free variables of all expressions are restricted to the set $\mathit{VPre} \cup U$, where U was defined as the set of declared universal variables.

The probability p_k must be a real number in the unit interval $[0,1] \subset \mathbb{R}$.

Let $\mathcal{T}$ be the set of all logical tables. We require their output variables to be disjoint:

$$\forall \tau, \lambda \in \mathcal{T}\colon\ \mathit{VPost}_\tau \cap \mathit{VPost}_\lambda = \emptyset \quad .$$

5.2.2 Semantics

Our tabular notation is similar to Bayesian networks [KF09]. These use tables for the definition of conditional probabilities over random variables. There are a few modifications, however: we allow universal variables, a logical precondition and constraining several output variables within a single table.

We start with studying the dependencies between variables declared by the headers of the tables. Then, we derive from initial tables the initial distributions and from logical tables the conditional probability distributions. Combining these results, we obtain the probabilistic behavior. Finally, we give a few syntactic extensions and important properties.

Variable Dependencies Each logical table introduces dependencies between variables according to its header and these dependencies are repeated at each time step. For example, a table with the input variable i and the output variables q and o' introduces the dependencies shown in Figure 5.5. For example, the value of variable o at time step 1 depends on the value of i at time step 0. These dependencies between the time-indexed variables correspond to the structure of a dynamic Bayesian network.

For convenience, we introduce auxiliary variables: for each point in time $t \in \mathbb{N}$, we map the variable $v \in R$ to the variable $v@t$ with $\mathbf{type}(v@t) = \mathbf{type}(v)$. In the case of a primed variable, the result is shifted about one time step, i.e., $v'@t = v@(t+1)$.

Next, we map these auxiliary variables onto the original sequences. Let V be a set of unprimed variables, $v \in V$ a variable and $x \in \overline{V}^\infty$ a sequence of valuations (see Section 3.1). The notation $x(v@t)$ or $x.v@t$ refers to the t-th value of the variable v in the sequence x, i.e., $x(v@t) = x(t)(v)$. In particular, for primed variables, we have $x(v'@t) = x(v@(t+1)) = x(t+1)(v)$.

We extend this notation also to sets of variables and valuations: given a valuation $r \in \overline{V}$ for a set V of primed and unprimed variables, we apply the time-indexing element-wise, i.e., $V@t = \{v@t \mid v \in V\}$, $r@t \in \overline{V@t}$ and $r@t(v@t) = r(v)$ for every $v \in V$. In particular, for primed variables, we have $r@t(v@(t+1)) = r(v')$.

We can now capture the dependencies induced by a table as the smallest relation $\mathit{Deps} \subseteq C@\mathbb{N} \times \mathcal{P}(C@\mathbb{N})$ such that the following property holds:

> For every point in time $t \in \mathbb{N}$, every logical table $\tau \in \mathcal{T}$ and every controlled variable in the precondition $x \in \mathit{VPre}_\tau \cap C$, the set of variables $\mathit{VPost}_\tau@t$ depends on $x@t$, i.e., $(x@t, \mathit{VPost}_\tau@t) \in \mathit{Deps}$.

In the sense of dynamic Bayesian networks, the variables $v@t$ correspond to the nodes of the network. However, compared to the relation between variables used in Bayesian networks, we relate variables to sets of variables in order to realize the simultaneous assignment to variables. However, one can easily translate this representation into an equivalent but larger Bayesian network by introducing auxiliary variables for tuples of variables. Figure 5.6 illustrates a possible part of the relation *Deps* and its translation to a Bayesian network.

The specification is only well-formed, if the dependency relation is acyclic and if every controlled variable occurs as the output of a table. We will discuss these properties in detail after giving the specification's semantics.

Initial Distribution Each init table determines the initial distribution (i.e., at time step zero) of some controlled variables *VInit*. We can denote this distribution schematically by $\mathbf{P}[\,\mathit{VInit}@0]$. The probability that the values are assigned according to the k-th row equals p_k (see the schema in Figure 5.3). Formally, we call this distribution *PrInit* and define it as a distribution over valuations of the variables *VInit*:

$$\mathit{PrInit} \in \mathbf{dist}(\overline{\mathit{VInit}})$$

$$\mathit{PrInit}(x) = \begin{cases} p_k & \text{if for row } k \text{ it holds } \forall i \in [1,l]\colon x.S_i = s_{i,k} \\ 0 & \text{otherwise.} \end{cases}$$

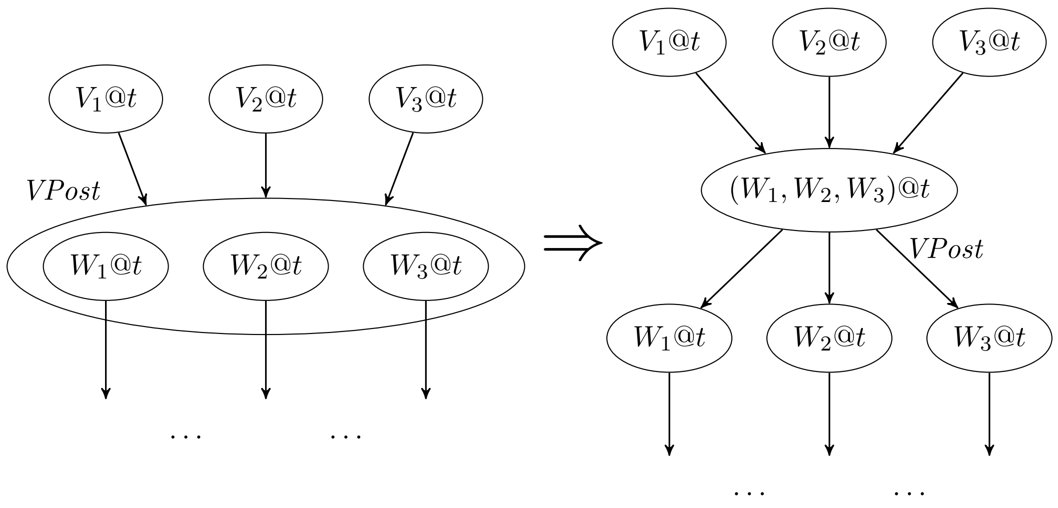


Figure 5.6: For example, assume the excerpt of the relation *Deps* shown on the left. It relates variables (at a certain point in time) with sets of variables. In Bayesian networks, however, variables are related to variables. In order to translate the *Deps* relation into a Bayesian network, we have to introduce an auxiliary variable $(W_1, W_2, W_3)@t$ with $\mathbf{type}((W_1, W_2, W_3)) = \mathbf{type}(W_1) \times \mathbf{type}(W_2) \times \mathbf{type}(W_3)$ and obtain the result on the right.

Note that for this definition to be sound, we require that the rows have disjoint value combinations, i.e., at most one row is matching a given valuation x, and that the sum of the probabilities equals one, i.e., $\sum_k p_k = 1$ where k ranges over all row indices.

Matching Rows Given the k-th row of a table, we interpret it as a set of constraints: we have the condition c_k and each cell in a variable column results in one equation $V_i = v_{i,k}$ or $W_i = w_{i,k}$. Each expression ($v_{i,k}$ and $w_{i,k}$) can refer to the variables *VPre* and U and, overall, the equations range over the variables *VPre*, U and *VPost*. We refer to the conjunction of these constraints as the predicate $match_k \colon \overline{VPre \cup U} \times \overline{VPost} \to \mathbb{B}$ where

$$\begin{aligned} match_k(x,y) \iff [\![c_k]\!](x) \;&\wedge\; \forall i \in [1,n] \colon x.V_i = [\![v_{i,k}]\!](x) \\ &\wedge\; \forall i \in [1,m] \colon y.W_i = [\![w_{i,k}]\!](x) \quad . \end{aligned}$$

Note that for some formula e (here e equals c_k, $v_{i,k}$ or $w_{i,k}$), we denote with $[\![e]\!](x)$ the evaluation of the expression e according to the valuation x (cf. Section 3.1).

Given valuations *pre* and *post* of the variables *VPre* and *VPost*, respectively. The conditional probability of *post* given *pre* is determined by all matching rows k, i.e., all rows for which a valuation u of the universal variables exists such that $match_k(pre \uplus u, post)$ becomes true.

The question arises what it means if more than one row matches *pre* and *post*. We could either forbid the usage of more than one matching row or aggregate the probabilities p_k of all matching rows. The latter is the more general interpretation, which we will stick to in the following. Only a practical evaluation of this description technique can demonstrate the implications of this expressivity, which, however, is not part of this thesis.

Furthermore, we have to decide how to handle the matching of a row according to more than one valuation of the universal variables. Either these are counted separately, as one matching or as an error. We decide for the first variant as it fits the idea that universal variables can be used as a kind of macro-expansion. A row containing universal variables can syntactically be replaced by a set of rows where the universal variables are replaced by all possible value combinations. For example, a row containing the universal variable $u \in \{0,1,2\}$ is then interpreted three times, once for each possible value of u (see also Example 5.2). The semantics could be easily adapted to the other variants (count several matchings only once or as an error) if practical studies indicate any complications.

In summary, we determine the *cumulated probability cPr* assigned by the table to a particular input/output combination *pre* and *post* as the sum over the probabilities of all matching rows (let *Rows* be the set of the table's row indices):

$$\begin{aligned} & \mathit{cPr}\colon \overline{\mathit{VPre}} \times \overline{\mathit{VPost}} \to \mathbb{R} \\ & \mathit{cPr}(\mathit{pre}, \mathit{post}) = \sum_{(k,u)\in \mathit{matches}} p_k \\ \text{where} \quad & \mathit{matches} = \{(k,u) \in \mathit{Rows} \times \overline{U} \mid \mathit{match}_k(\mathit{pre} \uplus u, \mathit{post})\} \quad . \end{aligned}$$

Conditional Probabilities We discussed already that each table induces dependencies between variables over time according to the table's header. Furthermore, we have determined the conditional probability that a table assigns to a particular input/output valuation. Next, we make clear how to combine these two insights.

Given a sequence of input valuations $i \in \bar{I}^\infty$ and a point in time $t \in \mathbb{N}$, each logical table τ provides a conditional probability distribution $\mathbf{P}[\mathit{VPost}@t \mid (C \cap \mathit{VPre})@t]$ (for conditional probability see Section 3.2) for time step t. This distribution determines the conditional probability that certain values are assigned to the output variables *VPost* given the values of the controlled variables in the precondition *VPre* (note that $\mathit{VPre} \subseteq I \cup C$ and $C \cap \mathit{VPre} = \mathit{VPre} \setminus I$). Let $f\colon I \overset{P}{\rhd} C$ be a probabilistic behavior function. The conditional distribution is defined over the $\overline{C}^\infty$-valued random variable $f(i)$:

$$\begin{aligned} & \forall \mathit{pre} \in \overline{C \cap \mathit{VPre}}, \mathit{post} \in \overline{\mathit{VPost}}\colon \\ & \quad \mathbf{P}\big[f(i) \overset{\mathit{VPost}@t}{=} \mathit{post}@t \;\big|\; f(i).t \overset{C\cap \mathit{VPre}}{=} \mathit{pre}\big] = \mathit{cPr}(\mathit{pre} \uplus i.t, \mathit{post}) \quad . \end{aligned}$$

The equality $v \overset{X}{=} w$ for valuations v and w and the set of variables X holds if, and only if,

$$\forall x \in X\colon v.x = w.x \quad .$$

Furthermore, as $f(i)$ is a sequence of valuations, we apply our previous definition that $f(i).(x@t) = f(i).t.x$.

Semantics of the Specification The semantics of the overall tabular specification is then obtained, similar to the joint distribution determined by a Bayesian network, from the product of the initial distribution and the conditional probability distributions. We will not explain how a Bayesian network induces the factorization of a joint distribution but refer to [KF09]. The product is of the form

$$\mathbf{P}[C^\infty] = \prod_{\tau\in\mathcal{I}} \mathbf{P}[\mathit{VInit}_\tau@0] \cdot \prod_{t\in\mathbb{N},\tau\in\mathcal{T}} \mathbf{P}[\mathit{VPost}_\tau@t \mid (C \cap \mathit{VPre}_\tau)@t] \quad .$$

However, as we talk about infinites sequences and have an infinite number of factors this product does not define much: for all cases but a few exceptions, this product equals zero. So, adapted to finite sequences, the correct product is, for all $k \in \mathbb{N}_0$:

$$\begin{aligned}\mathbf{P}[C^{k+1}] = \prod_{\tau \in \mathcal{I}} \mathbf{P}[\mathit{VInit}_\tau @0] \cdot \prod_{t \in [0,k-1], \tau \in \mathcal{T}} \mathbf{P}[\mathit{VPost}_\tau @t \mid (C \cap \mathit{VPre}_\tau)@t] \\ \cdot \prod_{\tau \in \mathcal{T}} \mathbf{P}[(C \cap \mathit{VPost}_\tau)@k \mid (C \cap \mathit{VPre}_\tau)@k] \quad . \qquad (5.1)\end{aligned}$$

The modified last factor in this product is necessary, in order to exclude variables of time $k+1$, which could occur by priming: $C'@k = C@k+1$.

Finally, we obtain a probabilistic behavior function $f\colon \bar{I}^\infty \to \bar{C}^P$ defined, for all $i \in \bar{I}^\infty$ and $\varphi \in \bar{C}^{k+1}, k \in \mathbb{N}_0$, by:

$$\begin{aligned}\mathbf{P}[f(i) \sqsupseteq \varphi] = \prod_{\tau \in \mathcal{I}} \mathit{PrInit}_\tau(\varphi.0|_{\mathit{VInit}_\tau}) \\ \cdot \prod_{t \in [0,k-1], \tau \in \mathcal{T}} \mathit{cPr}_\tau(\varphi.t \uplus i.t|_{\mathit{VPre}_\tau}, \varphi.t \uplus \varphi(t+1)'|_{\mathit{VPost}_\tau}) \\ \cdot \prod_{\tau \in \mathcal{T}} \sum_{d \in \bar{C}} \mathit{cPr}_\tau(\varphi.k \uplus i.k|_{\mathit{VPre}_\tau}, \varphi.k \uplus d'|_{\mathit{VPost}_\tau}) \quad .\end{aligned}$$

The specification of the component with syntactic interface $I \triangleright O$ is derived from f by projection, dropping the local variables L.

Example 5.1 The first example is the trivial specification of the producer from the ABP example:

Producer' ══════ det. prob

out $x \in \{a, b\}$

x	Prob
a	1/2
b	1/2

The semantic is given as a behavior function $f\colon \bar{\emptyset}^\infty \to \overline{\{x\}}^P$ with $\mathbf{type}(x) = \{a, b\}$ and the property

$$\mathbf{P}[f(()^\infty) = \varphi] = 1/2^k$$

for all $\varphi \in \overline{\{x\}}^k, k \in \mathbb{N}$. As we see, in this definition there is no need for an init table as we did not use any primed variables. □

Example 5.2 The following specification Medium1 shows the usage of a universal variable m in the logical table to propagate the input value of channel x to the output (first row) and to match an arbitrary input (both rows). Note that, for a set T, we defined the set $T^- = T \cup \{\Box\}$ containing additionally the empty symbol.

Medium1$\big[$**type** $T, q \in]0,1[\,\big]$ det. prob

in	$x \in T^-$
out	$y \in T^-$
univ	$m \in T^-$

init

y	Prob
$\Box$	1

x	y'	Prob
m	m	q
m	$\Box$	$1-q$

Medium1 specifies unambiguously a behavior function $f\colon \overline{\{x\}}^\infty \to \overline{\{y\}}^P$ with **type**$(x) =$ **type**$(y) = T^-$.

The headers of both tables (y and $x \to y'$) determine that the probabilities of the behavior are given as the following product. For all input sequences $i \in \overline{\{x\}}^\infty$ and output sequences $\varphi \in \overline{\{y\}}^{k+1}, k \in \mathbb{N}_0$, we have

$$\mathbf{P}[f(i) \sqsupseteq \varphi] = \mathbf{P}[f(i).0 = \varphi.0] \cdot \prod_{t \in [0,k]} \mathbf{P}[f(i).(t+1) = \varphi.(t+1)] \quad .$$

The factors of this product are dictated by the entries of the tables. The init table determines the initial distribution

$$\mathbf{P}[f(i).0.y = \Box] = 1$$

and the logical table translates to, for all $t \in \mathbb{N}_0$,

$$\begin{aligned} i.t.x = \Box \Rightarrow \ & \mathbf{P}[f(i).t+1.y = \Box] = q + (1-q) = 1 \\ \wedge\ i.t.x \neq \Box \Rightarrow \ & (\ \mathbf{P}[f(i).t+1.y = \Box] = 1-q \\ & \wedge\ \mathbf{P}[f(i).t+1.y = i.t] = q\ ) \quad . \end{aligned}$$

□

Syntactic Extensions Often, we want to refer to a readable variable in the postcondition without constraining it in the precondition. In other cases, we may have several rows that may not all depend on the same readable variables. To make such specifications more concise, the **wildcard** entry "–" can be used. Every occurrence of the wildcard is translated to a new universal variable.

An input column containing only wildcards may be omitted. Note that this column still exists implicitly and is relevant for the definition of the table's semantics.

Example 5.3 In the second row of the tabular specification of Medium1, the universal variable m occurs only within a single entry. Hence, we can replace it by the wildcard as shown in the specification of Medium2. We can even completely drop the universal variable m by directly naming the input variable x in the y' column. Then the x column contains only wildcards and can be omitted, as shown in Medium3. The semantics of all three specifications Medium1-3 are identical. $\square$

Medium2[...] det. prob

in $x \in T^-$
out $y \in T^-$

univ $m \in T^-$

init

y	Prob
$\square$	1

x	y'	Prob
m	m	q
$-$	$\square$	$1-q$

Medium3[...] det. prob

in $x \in T^-$
out $y \in T^-$

init

y	Prob
$\square$	1

y'	Prob
x	q
$\square$	$1-q$

5.2.3 Properties

Theorem 6 (Soundness, Completeness)
A tabular specification defines a probabilistic behavior (soundness) and this behavior is fully specified (completeness) if the following requirements hold:

R1) Each controlled variable $c \in C$ either occurs once primed on the right hand side of exactly one logical table and in exactly one init table or it occurs once unprimed on the right hand side of exactly one logical table and nowhere else[1]*:*

$$\begin{aligned}&\big(\exists^{=1}\tau \in \mathcal{T}\colon c' \in VPost_\tau \wedge \neg\exists\tau \in \mathcal{T}\colon c \in VPost_\tau\\ &\quad\wedge \exists^{=1}\tau \in \mathcal{I}\colon c \in VInit_\tau\big)\\ \vee\ &\big(\exists^{=1}\tau \in \mathcal{T}\colon c \in VPost_\tau \wedge \neg\exists\tau \in \mathcal{T}\colon c' \in VPost_\tau\\ &\quad\wedge \neg\exists\tau \in \mathcal{I}\colon c \in VInit_\tau\big) \quad .\end{aligned}$$

R2) The dependency relation Deps is acyclic. Note that cycles can only occur because of mutually dependent logical tables with regard to unprimed variables

[1]We use the notation $\exists^{=1}$ for "exists exactly one".

(e.g., tables with headers $x \to y$, $y \to z$ and $z \to x$):

$$\begin{aligned}&\neg\exists v_0, \ldots, v_{k+1} \in C, \tau_0, \ldots, \tau_k \in \mathcal{T}:\\ &\quad v_0 = v_{k+1} \wedge\\ &\quad \forall l \in [0,k]\colon v_l \in \mathit{VPre}_{\tau_l} \wedge v_{l+1} \in \mathit{VPost}_{\tau_l} \quad .\end{aligned}$$

R3) For every logical table $\tau \in \mathcal{T}$ and every valuation of pre $\in \overline{\mathit{VPre}_\tau}$, the function post $\in \overline{\mathit{VPost}_\tau} \mapsto cPr_\tau(pre, post)$ must define a probability distribution:

$$\sum_{post \in \overline{\mathit{VPost}_\tau}} cPr_\tau(pre, post) = 1 \quad .$$

R4) The same must hold for each init table $\tau \in \mathcal{I}$:

$$\sum_{v \in \overline{\mathit{VInit}_\tau}} PrInit_\tau(v) = \sum_k p_{\tau,k} = 1 \quad . \qquad \Box$$

Proof We have to show that the relation *Deps* indeed induces a Bayesian network that determines a probabilistic measure over sequences of the controlled variables, i.e., $\overline{C}^\infty$.

Requirements *(R3, R4)* ensure that the tables indeed define well-formed distributions.

Requirement *(R1)* ensures that every controlled variable $c \in C$ occurs in the Bayesian network for every time $t \geq 0$:

In case of usage of the primed variable c', the according logical table implicates the conditional probability distribution $\mathbf{P}[\mathit{VPost}@t \mid (C \cap \mathit{VPre})@t]$ with $c@(t+1) \in \mathit{VPost}@t$ for every $t \geq 0$ (note that $C \cap \mathit{VPre}$ may be empty) and the init table determines the initial distribution $\mathbf{P}[\mathit{VInit}@0]$ with $c@0 \in \mathit{VInit}@0$.

In case of usage of the unprimed variable c in a logical table, this table alone determines conditional distributions $\mathbf{P}[\mathit{VPost}@t \mid (C \cap \mathit{VPre})@t]$ with $c@t \in \mathit{VPost}@t$ for every $t \geq 0$.

Both cases together show that every variable is determined for every time step by a probability distribution. Because of Requirement *(R2)* and the fact that only unprimed variables occur on the left hand side of logical tables, the conditional distributions are not mutual dependent and we know from Bayesian networks that the product of these distributions up to time k indeed defines a joint distribution over $\overline{C}^k$. This in turn determines a probability measure over $\overline{C}^\infty$. ■

Theorem 7 (Strong Causality) *If, additionally, none of the output channels $o \in O$ instantaneously depends on any input $i \in I$, i.e.,*

$$\begin{aligned}&\neg\exists v_0, \ldots, v_{k+1} \in R, \tau_0, \ldots, \tau_k \in \mathcal{T}\colon\\ &\quad v_0 \in I \wedge v_{k+1} \in O \wedge\\ &\quad \forall l \in [0,k]\colon v_l \in \mathit{VPre}_{\tau_l} \wedge v_{l+1} \in \mathit{VPost}_{\tau_l}\end{aligned}$$

then the specified behavior is strongly causal. □

Proof The probability distribution for the outputs O up to some time step k cannot depend on any input of time step k because of this restriction. During the projection of the distribution $\mathbf{P}[C^k]$ onto the output channels any dependency on inputs of time step k is removed. More concretely, at most the last factor in the defining Equation (5.1) refers to the input at time k (it may be $i@k \in \mathit{VPre}_\tau@k$ for some $i \in I$). Because of the above restriction, this factor does not influence the probability of any output up to time k. Thus, summing over the local variables L, as it is done during the projection on the outputs, removes this factor and the resulting behavior function has to be strongly causal. ■

5.3 Transition Diagrams

In this section, we present a graphical notation for the definition of probabilistic behavior. We saw already many examples of deterministic transition diagrams throughout the preceding chapters. We will formalize this notation and additionally show how to graphically specify probabilistic behavior also in the nondeterministic case. We adapt the state transition diagrams for non-probabilistic systems from the Focus theory to incorporate probability annotations. In these extended diagrams, transitions are labeled with pre- and post-conditions, input and output messages as well as ranges of probabilities.

In Chapter 4, we saw already that a variety of representations of probabilistic I/O automata exists, which are all of equal expressiveness. Accordingly, we could introduce a variety of graphical notations each of which is particularly suited for certain applications. In the following, we will focus on one variant that allows nondeterminism at three places:

- Instead of dictating exact probabilities, inequalities can be used.
- We allow for several different probability distributions over successor states.
- We allow for several initial transitions to determine a set of initial distributions.

In the next section, we provide the syntax of transition diagrams. With such diagrams we want to define nondeterministic probabilistic behavior as defined in Chapter 6. So far however, we did not introduce any notion of nondeterministic automata. Thus, as a necessary step towards the semantics of transition diagrams, we will define nondeterministic probabilistic I/O automata. Finally, we interpret transition diagrams using this notion of I/O automata and illustrate the given definitions with an example.

5.3.1 Syntax

In the following, we will define the syntax of a transition diagram specification. The syntax is based on the frame template as explained in the introduction of this chapter.

Besides the syntactic interface, which is declared as stated in the introduction (Section 5.1), **local state variables** can be declared using the keyword **local**:

$$\textbf{local}\ l_1 \in L_1, \ldots, l_k \in L_k \quad .$$

The declared typing has to be respected within the whole specification. We define the set of all local variables $L = \{l_1, \ldots, l_k\}$ which we will later extend about a state variable that is assigned the current control state.

The content body contains a **transition diagram** which consists of **control states** and **transitions** between these states, and one or more **initial transitions**. For an example, see Figure 5.8. In our notation, the control states are visualized as rectangular nodes and the transitions between states, as usual, as lines with arrow-heads connecting the nodes.

A control state in turn consists of a unique name and an output statement as visualized on the left in Figure 5.7. The statement is a list of assignments: to each output channel $o_i \in O$ the value $v_i \in \textbf{type}(o_i)$ is assigned.

In Focus transition diagrams, a transition label has the form $\{P\}in/out\{Q\}$ where P is the **pre-condition**, Q is the **post-condition** and *in* and *out* are variable bindings to input and output channels, respectively. In our case, the output assignment *out* is annotated at each control state and not at the transitions to ensure strong causality as is typical for Moore automata. The **input binding** *in* has the form

$$in = (i_1?w_1, \ldots, i_k?w_k)$$

where the $i_j \in I$ are input channels and each w_j is either a variable or an expression that evaluates to a value. Such an expression may not refer to any other variables. If w_j is a variable, it can be constrained in the pre-condition P and be referenced in the post-condition. The **post-condition** has to be an assignment to local variables of the form

$$Q = (l_1 := e_1, \ldots, l_k := e_k)$$

where $l_1, \ldots, l_k \in L$ are local variables. The expressions $e_1, \ldots, e_k$ may refer to the local variables as well as the variables w_l declared in the input binding *in*. A reference to a local variable within an expression e_j is interpreted as a reference to this variable's value in the source state. In contrast, the resulting value assigned to l_j determines the value of l_j in the destination state.

The pre-condition P may leave the inputs completely unconstrained (if $P =$ *true*), and the post-condition may change none of the local variables (if Q equals the empty assignment $(\,)$). We will usually leave out such trivial conditions from a transition diagram.

Instead of annotating a single postcondition, however, we have to extend the notation to capture a distribution over post-conditions. As shown in the middle of Figure 5.7, a **transition** consists of a precondition P, an input pattern *in* and a set of quadruples $(Q_l, p_l, q_l, c'_l), l \in [1, n]$ for some $n \geq 1$. The Q_l are assignments to local variables as in the non-probabilistic case. Each of the pairs (p_l, q_l) with $p_l, q_l \in [0, 1] \subset \mathbb{R}$ and $p_l < q_l$ defines a closed interval $[p_l, q_l]$ for the probability that the according successor state is chosen. All these intervals together constraint the probability distribution over the successor states. If only a single probability p_l is given, we assume the singleton interval $[p_l, p_l]$. If additionally p_l equals one, i.e., $(p_l, q_l) = (1, 1)$, we usually drop the probability annotation. In this case, the transition has only a single branch and we draw a single arrow from a state c to state c' with a label of the form $\{P\}in\{Q\}$.

Compared to the just defined transitions, each **initial transition**, as shown in the right of Figure 5.7, also declares a set of quadruples $(Q_l, p_l, q_l, c'_l), l \in [1, n]$. However, it omits the precondition P and the input binding *in*.

5.3.2 Nondeterministic Probabilistic I/O Automata

Similar to the deterministic probabilistic I/O automaton (cf. Chapter 4), a **nondeterministic probabilistic I/O automaton** is a tuple $M = (S, I, O, \delta, \Delta, \omega)$ where S is the state space, I the input alphabet, O the output alphabet, δ a set of initial distributions, Δ a nondeterministic transition function and ω an output function:

$$\begin{aligned}
&\delta \in \mathcal{P}(\mathbf{dist}(S)) \\
&\Delta : S \times I \to \mathcal{P}(\mathbf{dist}(S)) \\
&\omega : S \to O \quad .
\end{aligned}$$

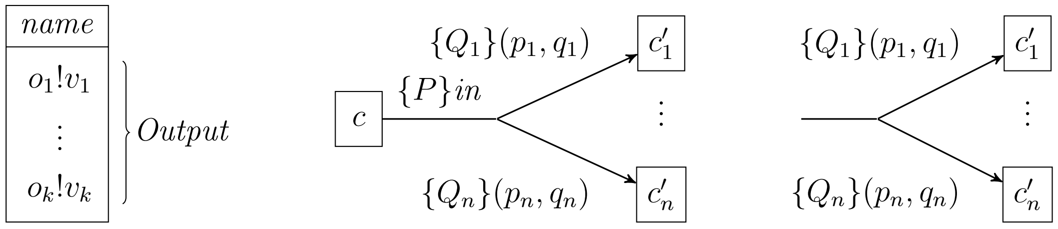


Figure 5.7: The elements of a state transition diagram: a control state with output assignments (left), a transition (middle) and an initial transition (right). P is a precondition, *in* is an input pattern, the Q_j are postconditions, the (p_j, q_j) are intervals of probabilities. The many branches of a transition together specify a set of distributions over the successor states.

As we can see, the difference to the deterministic variant is that instead of providing particular distributions initially and at each transition, we specify a set of possible distributions.

The selection of a particular distribution and thereby resolving the nondeterminism is the task of the adversary. An **adversary** A (also called *scheduler* and *policy*, cf. [dAHJ01, SL94]) of the nondeterministic probabilistic I/O automaton M is a function

$$A\colon (S \times I)^* \to \mathbf{dist}(S)$$

where, for all $p \in (S \times I)^+$,

$$\begin{aligned} & A(\langle\,\rangle) \in \delta \\ \wedge\ & A(p) \in \Delta(\text{last}(p)) \quad . \end{aligned}$$

Each of the (possibly infinitely many) adversaries for M induces a deterministic and probabilistic I/O automaton $M_A = (S_A, I, O, \delta_A, \Delta_A, \omega_A)$ where each state additionally encodes the sequence of past inputs and states, i.e., $S_A = (S \times I)^* \times S$. The functions of M_A are defined by (for all $(p,s), (p',s') \in S_A, i \in I$):

$$\begin{aligned} \delta_A((p,s)) &= \begin{cases} A(\epsilon)(s) & \text{if } p = \langle\,\rangle \\ 0 & \text{otherwise} \end{cases} \\ \Delta_A((p,s),i)((p',s')) &= \begin{cases} A(p')(s') & \text{if } p' = p \cdot \langle (s,i) \rangle \\ 0 & \text{otherwise} \end{cases} \\ \omega_A((p,s)) &= \omega(s) \quad . \end{aligned}$$

The deterministic probabilistic I/O automaton M_A in turn defines a strongly causal, deterministic and probabilistic behavior function $f_{M_A}\colon I \to O^P$, which abstracts from the states of the automaton (see Section 4.2).

The semantics of the nondeterministic probabilistic I/O automaton M is given as a nondeterministic behavior function $F_M \colon I \to \mathcal{P}(O^P)$. For each input, we allow any behavior according to one of the induced deterministic automata as well as any countable linear combination of these behaviors (cf. Convex Closure $\Gamma(\cdot)$ defined in Section 6.2):

$$F_M(i) = \Gamma(\{f_{M_A}(i) \mid A \text{ is an adversary of } M\}) \quad .$$

Instead of instantiating the nondeterministic I/O automaton with *arbitrary* adversaries, we could also restrict the abilities of the adversaries and obtain other varieties of semantics. We can restrict, for example, the temporal extent of the adversary's memory. In this case, given a limit $k \in \mathbb{N}_0$, we require that the adversary's decision depends only on the input and the states of the last k time steps, i.e.,

$$\forall n \in \mathbb{N}_0, p_1, p_2 \in (S \times I)^n, q \in (S \times I)^k \colon A(p_1 \cdot q) = A(p_2 \cdot q)$$

We call such an adversary **memory-k-bounded**. For $k = 0$, we call it **memoryless**. Additionally, we can require **time-independence** which means: the adversary's decision may depend on the k-bounded history but does not know how long the system (as an absolute measure) is running. Formally, this property requires (this time, p_1 and p_2 may have different lengths):

$$\forall p_1, p_2 \in (S \times I)^*, q \in (S \times I)^k \colon A(p_1 \cdot q) = A(p_2 \cdot q) \quad .$$

Independent of these temporal restrictions, we can limit the variables or channels that the adversary can depend on. For example, we can restrict the access to the state (i.e., the adversary's result is independent of the input):

$$\forall n \in \mathbb{N}, i_1, i_2 \in I^n, s \in S^n \colon A(i_1 \times s) = A(i_2 \times s) \quad .$$

5.3.3 Semantics

A transition diagram as part of a specification with input and output alphabets I and O defines a nondeterministic, probabilistic automaton $M = (S, \overline{I}, \overline{O}, \delta, \Delta, \omega)$ which in turn defines the nondeterministic behavior function $F_M \colon I \overset{P}{\rhd\!\!\gg} O$. The control states together with the explicitly declared local variables L form the state space of this I/O automaton. We introduce an additional local variable *state* typed with the set of names of all control states and define the state space S to be the set of all possible valuations of the local variables $L' = L \cup \{\mathit{state}\}$, i.e., $S = \overline{L'}$.

Output Function The output statement $out = (o_1!v_1, \dots, o_k!v_k)$ (cf. Figure 5.7) of a control state of name c determines the output function ω for all states $s \in S$ with $s.state = c$:

$$\forall s \in S\colon s.state = c \implies \omega(s) = (o_1 \mapsto v_1, \dots, o_k \mapsto v_k) \quad .$$

Transition Function The postcondition Q of a transition modifies the values of the local variables if this transition is traversed. Given a state $s \in S$ and a postcondition $Q = (l_1 := e_1, \dots, l_k := e_k)$, we denote with $s/Q \in S$ the valuation that results from s if the assignments of Q are applied. Formally,

$$(s/Q).x = \begin{cases} e_j & \text{if } x = l_j \text{ for some } j \in [1,k] \\ s.x & \text{otherwise} \quad . \end{cases}$$

Assume $s \in S$ to be the current state. Let be $i \in \bar{I}$ some input and τ some transition consisting of a source control state c, a precondition P, an input pattern in and a set of branches $(Q_l, p_l, q_l, c'_l), l \in [1,n]$ (cf. middle of Figure 5.7). Every such τ determines a set of distributions over the possible successor states $D_\tau(s,i) \subseteq$ **dist**(S). If the transition τ starts in the current control state, i.e., $c = s.state$, and if the values of the local variables as described by s and the input i match the input pattern in as well as the precondition P, then the set $D_\tau(s,i)$ contains exactly the distributions that fulfill the assignments of the postconditions Q_l with probabilities that match the intervals $[p_l, q_l]$. Otherwise, the set $D_\tau(s,i)$ is empty. Formally, we have $d \in D_\tau(s,i)$ if and only if

$$\begin{aligned} & s.state = c \wedge in(i) \ \wedge \ P \ \wedge \\ & \qquad \forall s' \in S\colon \big(\exists l \in [1,n]\colon s' = s/Q_l/(state := c'_l) \ \wedge \ d(s') \in [p_l, q_l]\big) \\ & \qquad\qquad \vee \ \big(\forall l \in [1,n]\colon s' \neq s/Q_l/(state := c'_l) \ \wedge \ d(s') = 0\big) \quad . \end{aligned} \tag{5.2}$$

Given the above definitions, the transition function Δ maps each state s and input i to the union of all $D_\tau(s,i)$, i.e.,

$$\Delta(s,i) = \bigcup_\tau D_\tau(s,i)$$

where τ ranges over all transitions. Note that we did not explicitly declare and existentially quantify over the variables bound by the precondition in in Formula (5.2). The assignments Q_l may refer to variables bound by the input pattern in.

Initial Distribution The initial distribution δ is obtained similarly to the transition function $\Delta \subseteq$ **dist**(S). We define the set of distributions D_τ for each initial

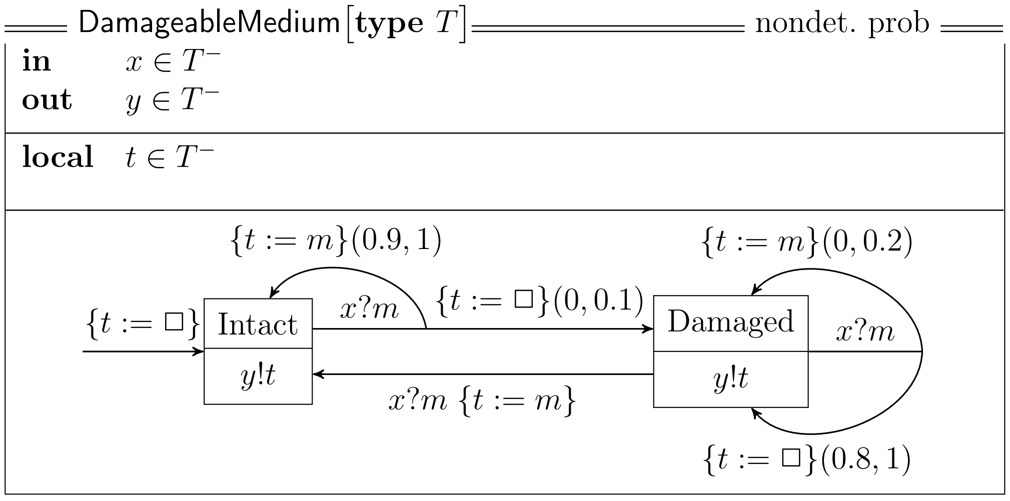


Figure 5.8: Specification of a transmission medium that can be intact or damaged. Depending on this state it forwards messages with a high and low probability, respectively.

transition (cf. right of Figure 5.7) similar to the Formula (5.2). We have $d \in D_\tau$ if and only if

$$\begin{aligned}\exists s \in S, \forall s' \in S \colon & \big(\exists l \in [1, n] \colon s' = s/Q_l/(\mathit{state} := c'_l) \ \wedge\ d(s') \in [p_l, q_l]\big) \\ & \vee \big(\forall l \in [1, n] \colon s' \neq s/Q_l/(\mathit{state} := c'_l) \ \wedge\ d(s') = 0\big)\end{aligned}$$

and δ is defined as the union of all D_τ, where τ ranges over all initial transitions.

Example 5.4 We model a transmission medium that can be either intact or damaged (see Figure 5.8). As long as it stays in the intact state, it correctly propagates messages. However, it may be damaged with a low probability (at most 0.1) and from then on the medium corrupts the messages with a high probability (at least 0.8). At any time the damaged medium might be repaired. As we do not know *when* the medium will be repaired, we employ a nondeterministic model.

This model integrates two degrees of nondeterminism. Firstly, we specify ranges of probabilities. This is useful if we do not have exact probabilities available and have to rely on lower and upper bounds. For example, in this specification we require that the intact medium forwards a message and stays intact with a probability of at least 0.9. Accordingly it gets damaged and drops a message with a probability of at most 0.1. The two alternative transitions leaving the state Damaged constitute the second kind of nondeterminism in this specification. These transitions lead to different combinations of successor states and post-conditions: either the medium is repaired and, accordingly, it forwards the incoming message or the medium stays damaged and drops the incoming message with a high probability (at least 0.8). A realization of this medium could be repaired arbitrarily. For example, it could be repaired according to a certain probability or always after a certain time-span, i.e., non-probabilistically and deterministically. □

5.4 Composite Specifications

As in Focus the composition of probabilistic components can be visualized graphically as *component networks*. We extend this notation, which emphasizes the flow of data between components over channels, with the notion of unspecified dependencies. This extension takes account for the probabilistic dependencies between components as discussed in Chapter 6.

The composition operators we have introduced are parallel composition $\varobar$, feedback $^{\circlearrowleft}$, output projection $\dagger$ and renaming of channels $[\cdot/.]$ (see Table A.3). Using these operators, we can combine existing components to create new components. This can be done textually using equations. For a more visual and intuitive representation, however, the Focus theory relies on component networks. These, in turn, are similar to data flow diagrams described in [DK82, LM87].

If we assume statistically independent components (cf. Section 6.3.2), the probabilistic counterparts of the non-probabilistic operators have the same properties (in particular, associativity and commutativity) and we can reuse the visualizations of Focus without modification.

In the presence of statistical dependencies between components, we need an additional means to describe these dependencies. To describe component dependencies visually, we introduce dependency diagrams in Section 5.4.2.

5.4.1 Component Networks

Components are visualized as rectangles tagged with the component's names. Typed channels connect these rectangles and are annotated with the channel's name and type in the form Name: Type. An arrow connects a pair of components if and only if the source and sink components have respectively an output and input channel of the same name. In that case, the types have to be identical. This means that connections visualize exactly the common channels of the composed components. This channel type restriction refers only to the syntactic interface of the sub-components. There is no other restriction for the composition. In particular, both elementary and composite specifications can be composed in a component network.

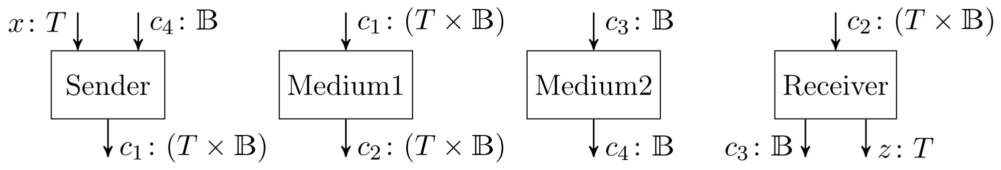
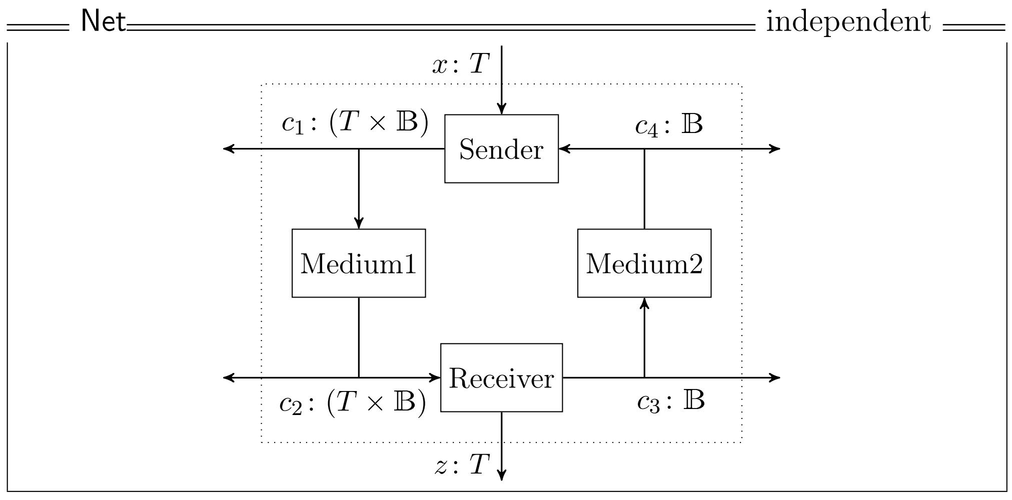


Figure 5.9: Visualization of the syntactic interfaces of some example components.

Given the components shown in Figure 5.9, their composition

$$\begin{aligned}\mathsf{Net} &= \bigotimes^{\perp}\{\mathsf{Sender}, \mathsf{Receiver}, \mathsf{Medium1}, \mathsf{Medium2}\}\\ &= \mathsf{Sender} \otimes^{\perp} \mathsf{Receiver} \otimes^{\perp} \mathsf{Medium1} \otimes^{\perp} \mathsf{Medium2}\end{aligned}$$

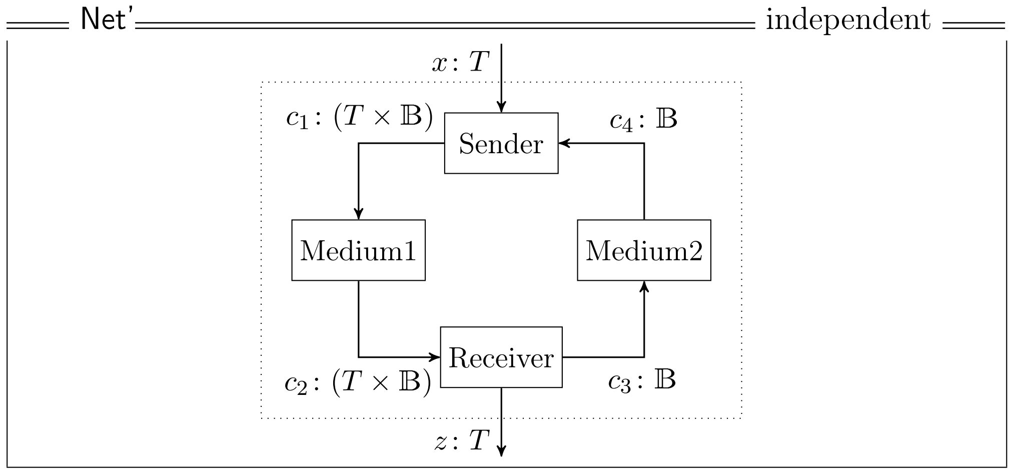
can be visualized as shown in the following specification frame:



Additionally to interconnecting components, we have to decide which channels of the composition are internal and which are exposed to the environment as part of the syntactic interface. Formally, we project the composition to a subset of its output channels using the projection operator †. Note that this operator only projects output channels and leaves the input channels untouched. In the case of our example, we would like to hide the channels c_1 to c_4 and define the component

$$\mathsf{Net}' = \mathsf{Net} \dagger \{z\}$$

which has the syntactic interface $\mathsf{Net}'\colon \{x\} \rhd \{z\}$. In the component network, we visualize the projection by forwarding only the projected channels to the composition border:



In both specifications (Net and Net'), we added the annotation **independent** to the frame. This means, that we assume the sub-components to behave statistically independent and messages can only be exchanged over the explicitly modeled channels. This annotation is reflected in the application of the $\otimes^{\perp}$ operator.

In the following section, we will see how to model component dependencies.

5.4.2 Dependency Diagrams

As described in Section 6.3.2, probabilistic components can depend on each other in a way like random variables can be statistically dependent. A typical description technique for dependencies between random variables is the *Bayesian network*. We adopt this technique and introduce *dependency diagrams* as a means to model potential dependencies and at the same time to prohibit all other dependencies. The idea is that a potential dependency of component B on component A can be understood as a directed and untyped channel that allows the flow of arbitrary information from A to B without delay. Note that in Bayesian networks a random variable only depends on the values of its parent random variables according to the variables' type [KF09]. Dependency diagrams, however, do not restrict the kind of information that may be exchanged by dependent components, i.e., arbitrary information can flow from A to B and it immediately follows that this kind of component dependency is *transitive*.

It is important to realize that we are not modeling *required* dependencies but we are modeling *potential* dependencies. That means, we allow a dependency, which will later, during the following developing, either be resolved by an explicit dependency (e.g., using a typed channel) or be prohibited. Every dependency that is not allowed is prohibited.

A **dependency diagram** consists, like a FOCUS component network, of **rectangles** representing components and **arrows** representing dependencies (or un-

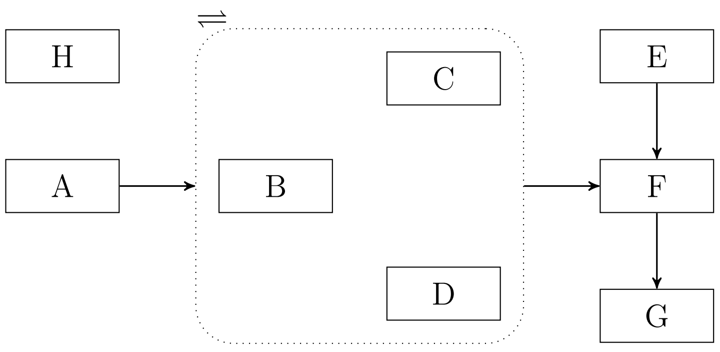


Figure 5.10: The graphical elements of a dependency diagram. Arrows indicate the allowed flow of information. The frame marked by $\rightleftharpoons$ and surrounding the components B, C and D is a shortcut to declare their mutual dependency. In this example, we have: F may depend on E, every component except H can depend on A and so on.

typed channels). Mutual dependencies form equivalence classes, where every component depends on every other component of this class. An explicit notation of such mutual dependencies using arrows is cumbersome. Therefore, we declare equivalence classes by a surrounding **rounded and dashed rectangle** annotated with the symbol $\rightleftharpoons$. In the following, we assume that every equivalence class is declared explicitly so that the arrows form an acyclic graph over components and equivalence classes.

Figure 5.10 illustrates these graphical elements. The components B, C and D form an equivalence class. The two arrows connecting single components declare that F depends on E and G depends on F. The arrow leaving the equivalence class denotes that F depends on B, C and D. The arrow entering the class denotes that B, C and D depend on A. Any dependency between H and the other components is prohibited.

A dependency diagram defines an acyclic relation $\mathcal{R}$ on the equivalence classes of the components. Thereby, every component is at least member of the singleton class containing only itself. Given the set of all components $\mathcal{C}$, we have $\mathcal{R} \subseteq \mathcal{P}(\mathcal{C}) \times \mathcal{P}(\mathcal{C})$. As explained before, we understand the modeled dependencies to work transitively. Thus, we are in fact interested in the transitive closure $\mathcal{R}^*$ of $\mathcal{R}$.

We can interpret the relation $\mathcal{R}^*$ like we do in the case of Bayesian networks. We define the parent, ancestor and descendant sets for each class in the obvious way and derive a set of conditional independencies L. A conditional independency, say $(A \perp B \mid C)$, means A is independent of B given C. According to Bayesian networks, a class $X \subseteq \mathcal{C}$ behaves independent of all its non-descendants given the behavior of X's parents. Formally, L is the smallest set of conditional independencies such that for every class X that occurs in $\mathcal{R}^*$:

$$(X \perp \textit{non-descendants}\,(X) \mid \textit{parents}(X)) \in L$$

where the non-descendant and parent sets refer to $\mathcal{R}^*$. Instead of defining the set L by means of the transitive closure $\mathcal{R}^*$, we can also characterize it directly using $\mathcal{R}$ and incorporating the transitivity. We obtain

$$(X \perp \textit{non-descendants}\,(X) \mid \textit{ancestor}(X)) \in L$$

where the non-descendant and ancestor sets refer to $\mathcal{R}$.

For the example in Figure 5.10, we obtain:

$$\begin{aligned} L = \{&(H \perp A, B, C, D, E, F, G), \\ &(A \perp H), \\ &(B, C, D \perp A, H \mid A), \\ &(E \perp A, B, C, D, H), \\ &(F \perp A, B, C, D, E, H \mid A, B, C, D, E), \\ &(G \perp A, B, C, D, E, F, H \mid A, B, C, D, E, F)\} \end{aligned}$$

which can be simplified to the equivalent[2] set

$$\begin{aligned} L' = \{&(H \perp A, B, C, D, E, F, G), \\ &(E \perp A, B, C, D)\} \quad . \end{aligned}$$

5.4.3 Combining both Diagrams

In general, a composite specification consists of both a component network and a dependency diagram. The former determines the sub-components $\mathcal{C}$ and the channels O visible to the outside. The latter defines a set of dependencies L between the declared components. The semantics of this specification is then given as the nondeterministic behavior function

$$F = \bigotimes^{L} \mathcal{C} \dagger O \quad .$$

[2]Two sets of independencies K and L over a set of variables (or components) V are equivalent if and only if:

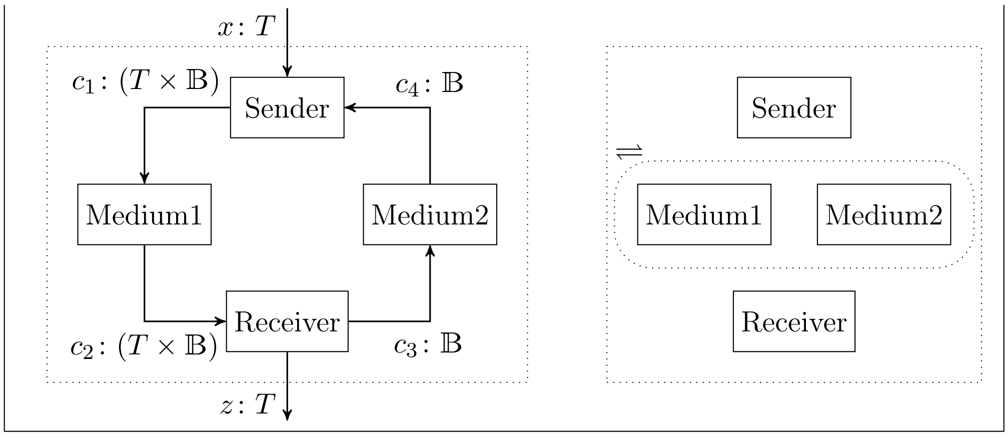
For any distribution P over V, it holds that P fulfills K if, and only if, P fulfills L.

See Section 6.3.2 for details.

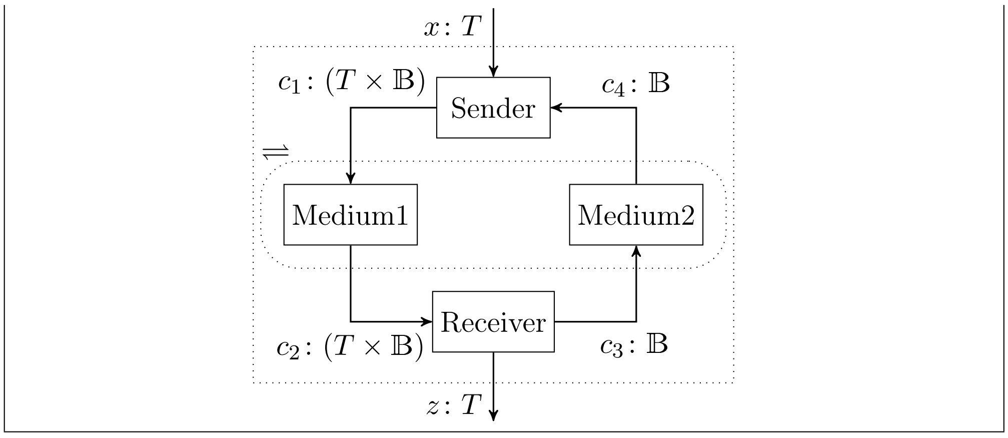
Example 5.5 Consider the composite specification Net′ from our previous example. There, we enforced the independence of all sub-components. In contrast, the following frame Net″ specifies that the communication media Medium1 and Medium2 may depend on each other in a not further specified way. For example, think of unreliable media that may drop messages and we do not want to dictate a concrete realization like the usage of a single bidirectional physical medium or of two separate ones. However, depending on the realization, errors may occur statistically dependent in both directions. The media may also be realized in a way that errors occur statistically independent. Therefore, we model a *potential* dependency between Medium1 and Medium2, which can be refined in later development steps. At the same time, Net″ requires that Sender and Receiver behave not only mutually independent but also independent of the Medium1 and Medium2.

The first specification includes both diagrams separately. However, we can also integrate both diagrams into one, as shown in the second, equivalent specification. □

Net″



Net″



Chapter 6

Nondeterminism

In Chapter 4 we introduced the framework to model deterministic probabilistic components. Thereby, as every theory, it realizes a certain abstraction of real systems. In this case, we abstract from implementation details, the hardware, the electronic circuits and so forth of real systems. Instead, we characterize systems with respect to the logical entities communicated over communication channels. If we fixate this level of abstraction, we understand these deterministic probabilistic components as fully specified. Each such component represents exactly one observable behavior, we cannot add further restrictions. Thus, there is no remaining degree of freedom. Nonetheless, a deterministic component can be realized by many real-world systems, in the same way, it can be realized by many probabilistic automata.

If we have agreed on the description for deterministic components, a *specification* is characterized by the set of deterministic components that fulfill it. The other way round, every set of deterministic components makes up a specification. Thus, a specification can comprise arbitrary properties about components. As a special case, we can say that *specifying* a deterministic probabilistic component means to describe a set containing only a single component.

Nondeterminism is another special kind of specification. Where a deterministic specification fixes every aspect such that there is only a single component left, a nondeterministic specification leaves certain degrees of freedom: For each input, it describes a set of possible output distributions. From these possibilities, a component may choose arbitrarily.

We will now sketch the idea of nondeterminism defined later in this chapter.

6.1 Introduction

Like in the deterministic case, a nondeterministic specification includes a syntactic interface, i.e., it declares a set of typed input and output channels. Using these channels, the output-behavior of the component is constrained for each input. For example, we could prohibit certain output-input combinations or require certain output probabilities. In general, these constraints are fulfilled by several behaviors. A realization of this nondeterministic specification has to fulfill these constraints but apart from these it may behave arbitrarily.

The Focus theory abstracts from probabilities and models systems either as deterministic or nondeterministic systems. There, a nondeterministic component specifies *which* outputs are allowed to occur for each input. Thereby, every component has to be input-enabled, i.e., it has to react to every input. It does not matter *how* this output is calculated but we ensure that no other output than the ones specified can be observed. In particular, in Focus we subsume probabilistic behavior: A certain output occurs if, and only if, the probability that this output occurs is greater than zero. We see this informal description directly reflected in the signature of a nondeterministic behavior function F:

$$I^\infty \to \mathcal{P}(O^\infty) \quad .$$

For every input i, the set $F(i)$ describes the possible outputs.

A real system that implements or realizes a nondeterministic specification can *choose* or *decide* between alternative behaviors. This process may be arbitrarily complex. Most important, it may depend on further, not even modeled, information. We have to expect that the system has additionally access to any of the following sources of information:

- The system may incorporate internal memory of any size.
- The system may depend on any other component. Thereby, it is not restricted to the channels that are modeled explicitly if the nondeterministic specification defines only a lower bound of channels. As long as it is not explicitly prohibited in the specification, its realizations may incorporate many additional channels to any other component, usually called **hidden channels**.
- We have to assume that the system is integrated in an unknown environment and it may interact with this environment in ways that are not explicitly modeled.
- The initial state of a system may not be unique, e.g., it may carry over information from a previous execution.

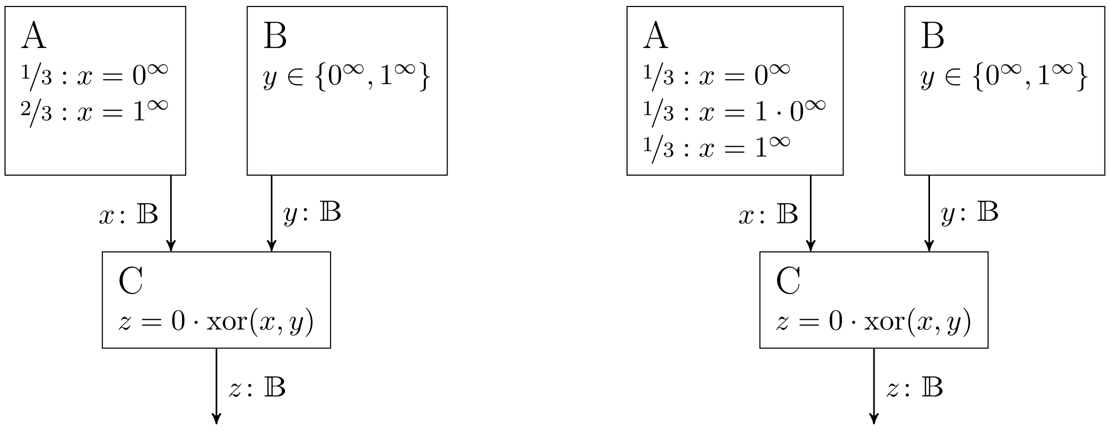


Figure 6.1: Composition of a probabilistic and a nondeterministic component.

- Finally, the computation itself may be probabilistic or appear randomly because of a large available memory.

In this chapter, we consider a more detailed kind of specification than the non-probabilistic and nondeterministic specification of FOCUS, namely specifications that constrain not only the allowed outputs but also the allowed probabilistic distribution of outputs. The need for such a kind of specification can be motivated with the question: What happens when a nondeterministic component is connected to a deterministic probabilistic one? Or in more detail: Given a realization a of a nondeterministic specification A and a realization b of a probabilistic specification B. What properties hold for the composition of a and b? Can we model the composition of A and B in a meaningful way, such that it describes at least some of these properties? To be more concrete, consider the following example.

Example 6.1 We consider three components A, B and C and their composition, as sketched in the left part of Figure 6.1. The component A probabilistically produces either of two values: with probability 1/3 repeatedly the value 0 otherwise repeatedly value 1. This value is forwarded by C to channel z but is changed depending on the output of B on channel y. In case of $y = 0^\infty$, the output of A is left untouched and, in case of $y = 1^\infty$, the output is flipped. We see, that component A is fully specified with a single valid output distribution, whereas component B is nondeterministic and restricted to the output of these two output sequences. The merging component C is nonprobabilistic and deterministic. After hiding the channels x and y, the composition has no remaining input channels and only the output channel z.

How can a single run of this system look like? With respect to the probabilistic elements in this system, we also want to ask for the behavior over several executions. Consider the first step of an execution: A will probabilistically choose between 0 and 1, but B can rely on any source of information, maybe on other components or on the environment using hidden channels. So, we arbitrarily say that B chooses 0. The choices in the first step determine the behavior for the remaining steps of the execution. Every time we restart the system, A chooses again probabilistically but B may produce again 0 or arbitrarily switch to the output 1. Thus, as B randomly flips the result of A, we cannot make any statement about the probability of a certain output on channel z. For example, every 1^∞ could be changed to 0^∞ and z would always take on the value 0^∞. Accordingly, we could achieve the opposite, that z always outputs 1^∞.

We conclude, that the channel z can obtain the following sequences, but we cannot fixate any bounds on the probability of occurrence:

$$\begin{aligned}(A \otimes B \otimes C).z &= \{0 \cdot \mathrm{xor}(0^\infty, 0^\infty), 0 \cdot \mathrm{xor}(0^\infty, 1^\infty),\\ &\qquad 0 \cdot \mathrm{xor}(1^\infty, 0^\infty), 0 \cdot \mathrm{xor}(1^\infty, 1^\infty)\}\\ &= \{0 \cdot 0^\infty, 0 \cdot 1^\infty\} \quad .\end{aligned}$$

In contrast, if we extend the example as shown in the right part of Figure 6.1, such that B cannot erase the output of A completely, we can indeed state something about the possible distribution of output values. Again, component B can swap the outputs 0^∞ and 1^∞ of A arbitrarily. But, no matter what B produces, the output $1 \cdot 0^\infty$ or $0 \cdot 1^\infty$ has to occur with probability $1/3$. We can state, that in every run of the composition of any realizations of A, B and C the property

$$\mathbf{P}[z = 0 \cdot 1^\infty \ \vee \ z = 1 \cdot 0^\infty] = 1/3 \ \wedge \ \mathbf{P}[z = 0^\infty \ \vee \ z = 1^\infty] = 2/3 \quad . \tag{6.1}$$

holds.

This property is tight, that means for every distribution fulfilling this equation, we can construct a realization of B such that we observe this distribution. In this construction B has to statistically depend on the distribution of A.

In this chapter, we will propose a formalism to capture such a composition. In this formalism, a nondeterministic specification is represented by a function from the input sequences $\bar{I}^\infty$ to a set of distributions over output sequences. Then the composition of the specifications A, B and C will look like (note that $I = \emptyset$ and $O = \{z\}$):

$$\begin{aligned}&(A \otimes B \otimes C) \colon \bar{\emptyset}^\infty \to \mathcal{P}(\overline{O}^P)\\ &(A \otimes B \otimes C)((\,)^\infty) = \{\mu \mid \mu \text{ is a measure over } \overline{O}^\infty\\ &\qquad\qquad\qquad\qquad\qquad \text{and fulfills equation (6.1)}\} \quad .\end{aligned}$$

□

$$\begin{tikzcd}
& \boxed{\text{det. Focus}} \arrow[dl] \arrow[dr] & \\
\boxed{\text{nondet. Focus}} \arrow[dr] & & \boxed{\text{det. prob. Focus}} \arrow[dl] \\
& \boxed{\text{nondet. prob. Focus}} &
\end{tikzcd}$$

Figure 6.2: Homomorphisms between flavors of FOCUS.

Conservative Extension

We want to extend both the FOCUS theory about probabilistic behavior and the deterministic probabilistic theory introduced in Chapter 4 about nondeterministic behavior. The latter was our motivating goal and the former ensures a meaningful notion of probabilistic components backed by probabilistic automata. Thus, we have to fulfill the diagram of Figure 6.2 where an arrow means that a homomorphism from one system notion to the other with respect to the general composition ($\otimes$) exists.

We will take over the concept of describing the behavior of a component per input history. This is reflected in the signatures for the different kinds of behavior functions, summarized in the following table:

Kind of behavior	**Signature of the behavior functions**	**Set Notation**
deterministic non-probabilistic	$\bar{I}^\infty \to \overline{O}^\infty$	$I \rhd O$
nondeterministic non-probabilistic	$\bar{I}^\infty \to \mathcal{P}(\overline{O}^\infty)$	$I \rhd\!\!\!\rhd O$
deterministic probabilistic	$\bar{I}^\infty \to \overline{O}^P$	$I \overset{P}{\rhd}$
nondeterministic probabilistic	$\bar{I}^\infty \to \mathcal{P}(\overline{O}^P)$	$I \overset{P}{\rhd\!\!\!\rhd} O$

Thereby, $\overline{O}^P$ designates the set of all probabilistic measures over infinite output sequences.

Testing Semantics

The meaning of a probabilistic and nondeterministic behavior function $F\colon I^\infty \to \mathcal{P}(O^P)$ can be explained by providing a decision method for the conformance of a real system. Assume that we review some real system for which we want to test or verify that it fulfills the specification F.

Superficially, we expect that a realization of F reacts to an input i according to one of the distributions in $F(i)$. "To react according to some distribution" would intuitively mean, that the system has to incorporate some kind of random generator and derive from this a behavior according to this distribution. Beyond that, we allow the system to behave according to a different distribution in each execution. This captures, more or less our intuition about the interpretation of F, but if we go into details of this concept, we meet a problem. On the one hand, probabilities and distributions can only be observed by repetition. Therefore, we have to consider several executions. On the other hand, we allow the system to behave according to several distributions. This makes it difficult, to verify or test a system for conformance or even indicate some conflict. We will propose now an approach to tackle with this problem.

Since the output behavior is specified per input, we will, for a start, test the system per input i. As we did in the previous example, we consider properties of the form $(\mathbf{P}[a]\,\mathrm{rel}\,p)$ for some event $a \subseteq O^\infty$, a relation $\mathrm{rel} \in \{=, \leq, \geq\}$ and a probability $p \in [0,1]$. Let $Q(i)$ be the set of all such properties that hold for every distribution in $F(i)$. Note that this is just one possible choice of a category of properties. Others might be useful, as well. Our condition is then, that a real system passes the test for input i if, and only if, it fulfills all properties in $Q(i)$. Instead of testing all properties, it is sufficient to test any set of properties $Q' \subseteq Q(i)$ such that the system fulfills Q' iff it fulfills $Q(i)$. A realization of F would have to pass the test for each possible input. In the previous example, we would test for the properties

$$Q' = \left\{ \begin{array}{l} \mathbf{P}[z = 0 \cdot 1^\infty \ \vee \ z = 1 \cdot 0^\infty] = 1/3, \\ \mathbf{P}[z = 0^\infty \ \vee \ z = 1^\infty] = 2/3 \end{array} \right\} \quad .$$

In order to verify conditions over distributions for a real system, we have to execute the system repeatedly. Thus, we abort the execution at some time but ensure that we test the system "long enough". For example, we could test the system at the n-th execution for n steps. Each time we provide the input i. Here, we assume it possible to abort the execution or restart the system. During all executions, we maintain a histogram of how often each output sequence occurred. Using this histogram, we can apply several statistical tests known from statistics [SSS08]. The standard test is to approximate probabilities of events by the relative frequencies. In this way, we could check every property $(\mathbf{P}[a]\,\mathrm{rel}\,p)$ using the relative frequency of the event a tolerating a small deviation about ϵ. Other tests incorporate also the order of the observed outputs.

In this form, the test is rather tolerant. For example, consider a system which remembers the first input symbol of the previous execution and behaves as specified if the first input of this execution is the same as remembered and behaves against the specification otherwise. Against our intuition, the system would pass our test.

A more rigorous test would consider various successions of different input sequences. It could also test the behavior for randomness and not only for the convergence of relative frequencies by applying stricter statistical tests. We leave this as an open challenge.

6.2 Behavior

For the definition of nondeterministic and probabilistic behaviors with input type I and output type O, we consider functions F with the signature

$$F\colon I^\infty \to \mathcal{P}(O^P) \quad .$$

This reflects the possibility that a nondeterministic system can decide on different outputs for each run. Still it has to follow certain probabilistic constraints. For such a function F to be a behavior, we require a few properties.

Causality Let $\mu{\downarrow_n}$ denote the X^n-valued random variable obtained by applying the prefix operator to the random variable $\mu \in X^P$. Furthermore, we lift this operation for application to sets of random variables. The function F is **strongly causal** if (for all $i, j \in I^\infty, n \in \mathbb{N}_0$):

$$i{\downarrow_n} = j{\downarrow_n} \implies F(i){\downarrow_{n+1}} = F(j){\downarrow_{n+1}}$$

where the conclusion is equivalent to $\{\mu{\downarrow_{n+1}} \mid \mu \in F(i)\} = \{\mu{\downarrow_{n+1}} \mid \mu \in F(j)\}$.

Convex closure The linear combination d of countably many distributions $d_k \in \mathbf{dist}(X), k \in \mathbb{N}$ is commonly defined by

$$d(x) = \sum_k w_k \cdot d_k(x)$$

for some weights $w_k \in [0,1] \subset \mathbb{R}$ with $\sum_k w_k = 1$.

We extend this notion of combination to probability measures over infinite sequences. Thereby, we have to consider that the probabilities for all single infinite sequences can be zero and thus have to fallback to basic cylinders of sequences. A probability measure μ is a **countable linear combination** over a set of probability measures $Y \subseteq X^P$ if, for each $n \in \mathbb{N}$, countably many measures $\mu_k \in Y$ and weights $w_k \in [0,1] \subset \mathbb{R}, k \in \mathbb{N}$, exist such that (for all $\varphi \in X^n$)

$$\mu(\mathcal{C}(\varphi)) = \sum_k w_k \cdot \mu_k(\mathcal{C}(\varphi)) \;\wedge\; \sum_k w_k = 1 \quad .$$

We call the set of all countable linear combinations of Y the **convex closure** of Y and write $\Gamma(Y)$. We call the set Y to be **convex** if $Y = \Gamma(Y)$ and a nondeterministic behavior F to be **convex** if $F(i)$ is convex for all $i \in I$.

Realizations Not every implementation of a nondeterministic behavior can be modeled within our theory. The main reason for this is the missing notion of a system restart in our theory. An implementation of a nondeterministic behavior may behave differently after each system restart. This variation may be probabilistic but could also be deterministic. For example, consider a system that alternately shows two different behaviors.

Being aware of this shortcoming, we want at least to consider all possible implementations or realizations of a nondeterministic behavior within our theory. The set of **realizations** $[\![F]\!]$ of a nondeterministic behavior function F is defined by

$$\begin{aligned} [\![F]\!] &\subseteq (I^\infty \to O^P) \\ f \in [\![F]\!] &\iff f \text{ is strongly causal} \\ &\qquad \wedge\; \forall i \in I^\infty \colon f(i) \in \Gamma(F(i)) \quad . \end{aligned} \tag{6.2}$$

As described in the introduction of this chapter, the idea of nondeterministic descriptions is to provide alternative outputs and allow arbitrary selection from these alternatives. In the probabilistic context, we also want to allow this selection procedure itself to be probabilistic. This explains why we incorporated the convex closure in the definition of realizations. By selecting the behaviors $f(i)$ from the convex closure of $F(i)$, we effectively allow a realization to probabilistically choose between alternative behaviors.

Realizability We call a nondeterministic behavior F **realizable** if there is some realization of F, i.e., $[\![F]\!] \neq \emptyset$, and **fully realizable** if additionally

$$\forall i \in I^\infty \colon F(i) \subseteq \{f(i) \mid f \in [\![F]\!]\} \quad .$$

That means, every output behavior in $F(i)$ occurs as the output of some realization of F given the input i.

Behavior The function F is a **nondeterministic (and probabilistic) behavior function** if it is strongly causal and convex.

Note that a behavior function F is characterized by its realizations and that these realizations depend only on the convex closure of F, see Equation (6.2). Thus, the required convexity does not restrict the possible behaviors but makes this representation canonical in the case that F is fully realizable. We have, for all nondeterministic, probabilistic and fully realizable behavior functions $F, F' \colon I^\infty \to \mathcal{P}(O^P)$,

$$[\![F]\!] = [\![F']\!] \iff F = F' \quad .$$

However, not every behavior function is fully realizable or even realizable:

Theorem 8 *There are nondeterministic behavior functions which are not realizable.* □

Proof Consider the behavior $F \colon X^\infty \to \mathcal{P}(X^P)$ with $X = \{0, 1\}$ defined by

$$\forall i \in X^\infty \colon \mu \in F(i) \iff \exists n \in \mathbb{N} \colon \mu(i{\downarrow_n}) = 0 \quad .$$

Assume we are given a realization $f \in [\![F]\!]$. We construct an input j that leads to a contradiction. To start with, we choose $j.0$ such that $f(\langle\ \rangle)(j{\downarrow_1}) > 0$. Then, given $j{\downarrow_n}$ for some $n \in \mathbb{N}$, we define $j.n$ such that $f(j{\downarrow_n})(j{\downarrow_{n+1}}) > 0$.

This input sequence j has the property that, for all $n \in \mathbb{N}_0$,

$$f(j)(j{\downarrow_{n+1}}) = f(j{\downarrow_n})(j{\downarrow_{n+1}}) > 0 \quad .$$

This contradicts our assumption that $f \in [\![F]\!]$ and concludes this proof. ■

Translation of Deterministic, Probabilistic Behavior The natural translation of a strongly causal behavior $f \colon I^\infty \to O^P$ to a nondeterministic behavior is given by the function $f^N \colon I^\infty \to \mathcal{P}(O^P)$ with

$$\forall i \colon f^N(i) = \{f(i)\} \quad .$$

Translation of Nondeterministic, Non-Probabilistic Behavior We described the translation of deterministic behavior functions from Focus into the probabilistic context in Section 4.4. Similarly, we define the **probabilistic translation** F^P of a nondeterministic and non-probabilistic behavior function $F \colon I^\infty \to \mathcal{P}(O^\infty)$ by

$$\forall i \in I^\infty \colon \mu \in F^P(i) \iff \mu(F(i)) = 1 \quad .$$

This is furthermore equivalent to $\mathbf{P}[\mu \in F(i)] = 1$. If we understand F as a system property depending on the provided input i, this definition reads as

$$\mu \in F^P(i) \iff \text{almost surely } \mu \text{ fulfills the property } F(i) \quad .$$

Translation of Deterministic and Non-Probabilistic Behavior In Focus, a deterministic behavior $f\colon I^\infty \to O^\infty$ is translated to a nondeterministic behavior as $f^N\colon I^\infty \to \mathcal{P}(O^\infty)$ with $f^N(i) = \{f(i)\}$. With this definition we obtain two alternative but equivalent translations (i.e., the translations commute), namely $(f^N)^P$ and $(f^P)^N$. We have, for all $i \in I^\infty$,

$$\begin{aligned} \mu \in & (f^N)^P(i) \iff \mu(\{f(i)\}) = 1 \\ & \iff \mu = f^P(i) \\ & \iff \mu \in (f^P)^N(i) \quad . \end{aligned}$$

Therefore, we write for this translation just f^{NP}.

Note that we do not require channels for these definitions, instead I and O are arbitrary countable alphabets.

In the subsequent sections, we provide the definitions for the composition and the feedback operators and show the relation to both their deterministic and their non-probabilistic counterparts.

6.3 Parallel Composition

Nondeterministic, probabilistic behavior functions with channel valuations as input and output sets are called nondeterministic, probabilistic components. Their syntactic interface is given by the channel sets. The syntactic interface of a behavior function $F\colon \overline{I}^\infty \to \mathcal{P}(\overline{O}^P)$ with channel sets I and O is denoted by $I \overset{P}{\rhd} O$. In this section, we consider the parallel composition of such components.

6.3.1 Dependent Composition

The parallel composition of nondeterministic behaviors is based on the concept of a joint distribution. Given two random variables x and y over values $x_1, x_2, \ldots$ and $y_1, y_2, \ldots$ and distributions $\mathbf{P}[x]$ and $\mathbf{P}[y]$, a **joint distribution** is a distribution $\mathbf{P}[x, y]$ over both variables. We call each distribution over a single variable **marginal distribution**. The notion of joint distribution is easily extended to more than two random variables. For non-trivial distributions, i.e., which do not assign the probabilities zero and one, there exist many joint distributions.

For example, consider two random variables x and y, both $\{a, b\}$-valued, then we can visualize a joint distribution d over the marginal distributions of x and y in a table of the form:

	$\mathbf{P}[x = a]$	$\mathbf{P}[x = b]$
$\mathbf{P}[y = a]$	$\mathbf{P}[x = a \wedge y = a]$	$\mathbf{P}[x = b \wedge y = a]$
$\mathbf{P}[y = b]$	$\mathbf{P}[x = a \wedge y = b]$	$\mathbf{P}[x = b \wedge y = b]$

The random variables are **statistically independent** if we have

$$\mathbf{P}[x = x_k \ \wedge \ y = y_l] = \mathbf{P}[x = x_k] \cdot \mathbf{P}[y = y_l]$$

for all k and l. In general, this does not hold and the variables are **statistically dependent**. We will see several examples in the context of probabilistic components in this chapter.

We define now the **parallel composition** of m behaviors A_k ($k \in [1, m]$) of syntactic interfaces $I_k \overset{P}{\rhd} O_k$ with disjoint output channels, i.e., $\bigcap_k O_k = \emptyset$. As in the deterministic case, the composition has the syntactic interface $I \overset{P}{\rhd} O$ where I and O are the unions of the input and output channels, i.e., $I = \bigcup_k I_k$ and $O = \bigcup_k O_k$. We denote the composition by $\prod_k A_k$ and define it by:

$$\forall i \in \bar{I}^{\infty} \colon \mu \in \Big(\prod\nolimits_k A_k\Big)(i) \iff \forall k \colon \mu|_{O_k} \in A_k(i|_{I_k}) \quad .$$

Analogously to joint distributions as describe before, we think of μ as a joint distribution of the projections $\mu|_{O_k}$. While each projection $\mu|_{O_k}$ conforms to the behavior A_k (the set of possible marginal distributions), there can be arbitrary dependencies in their combination, as is the case with joint distributions. Thus in general, μ is not uniquely defined, even if the $A_k(i|_{I_k})$ are singletons. In the case $m = 2$, we reuse the notation $A_1 \obar A_2$.

Example 6.2 Consider the unreliable transportation medium Medium1 from Example 5.2, whose specification we repeat here.

Medium1[**type** $T, q \in]0,1[$] det. prob

in $i \in T^-$
out $o \in T^-$

univ $m \in T^-$

init

o	Prob
$\square$	1

i	o'	Prob
m	m	q
m	$\square$	$1 - q$

Let q be some common single-failure probability and T some data type. In order to apply the just defined parallel composition, we rename the channels of Medium1 such that we have to components with disjoint channels:

$$\begin{aligned} \mathsf{M}_1 &= \mathsf{Medium1}[T, q][{}^{i}/{}_{i_1}, {}^{o}/{}_{o_1}] \\ \mathsf{M}_2 &= \mathsf{Medium1}[T, q][{}^{i}/{}_{i_2}, {}^{o}/{}_{o_2}] \quad . \end{aligned}$$

We will compare the composition of M_1 and M_2 with the composition of their nondeterministic translations $\mathsf{N}_1 = \mathsf{M}_1{}^N$ and $\mathsf{N}_2 = \mathsf{M}_2{}^N$.

The parallel composition $\mathsf{M}_1 \oplus \mathsf{M}_2$ of the deterministic and probabilistic components was introduced in Section 4 and is behaviorally equivalent to $\mathsf{IndMedia}$ (see the specification below), formally:

$$\mathsf{M}_1 \oplus \mathsf{M}_2 = \mathsf{IndMedia}[T, q] \quad .$$

$\mathsf{IndMedia}\big[\textbf{type } T, q \in]0,1[\ \big]$ det. prob

in $i_1, i_2 \in T^-$
out $o_1, o_2 \in T^-$

init

o_1	o_2	Prob
$\square$	$\square$	1

o'_1	o'_2	Prob
i_1	i_2	$(1-q)^2$
i_1	$\square$	$q(1-q)$
$\square$	i_2	$q(1-q)$
$\square$	$\square$	q^2

We see, that the probability that both media drop messages at the same point in time, lets call it r, is equal to q^2.

It is interesting to compare this with the composition $\mathsf{N}_1 \oplus \mathsf{N}_2$, which allows arbitrary dependencies between the components. Therefore, we consider two extreme realizations of this parallel composition, namely $\mathsf{DepMedia1}, \mathsf{DepMedia2} \in [\![\mathsf{N}_1 \oplus \mathsf{N}_2]\!]$ as defined in the following specification frames. In $\mathsf{DepMedia1}$ the two media always drop and transport messages synchronously, i.e., $r = q$, whereas the second system $\mathsf{DepMedia2}$ minimizes the probability r. For a single-failure probability of $q \leq 1/2$, $\mathsf{DepMedia2}$ reduces the probability r to zero and to $2q-1$ otherwise.

We conclude that in realizations of the nondeterministic system $\mathsf{N}_1 \oplus \mathsf{N}_2$ the probability r ranges from $\max\{0, 2q-1\}$ up to q. As expected, this interval also contains the probability $r = q^2$ of the independent case:

$$2q - 1 \leq q^2 \iff 0 \leq q^2 - 2q + 1 = (q-1)^2 \quad .$$

$\mathsf{DepMedia1}\big[\textbf{type } T, q \in]0,1[\ \big]$ det. prob

in $i_1, i_2 \in T^-$
out $o_1, o_2 \in T^-$

init

o_1	o_2	Prob
$\square$	$\square$	1

o'_1	o'_2	Prob
i_1	i_2	$1-q$
$\square$	$\square$	q

DepMedia2[**type** $T, q \in]0,1[$] — det. prob

in $i_1, i_2 \in T^-$

out $o_1, o_2 \in T^-$

let $r = \max\{0, 2q - 1\}$

init

o_1	o_2	Prob
$\Box$	$\Box$	1

o'_1	o'_2	Prob
i_1	i_2	$1 - 2q + r$
i_1	$\Box$	$q - r$
$\Box$	i_2	$q - r$
$\Box$	$\Box$	r

In order to determine the possible outputs of the dependent composition ignoring the concrete probabilities, we can consider a non-probabilistic abstraction of Medium1 and apply the composition of Focus. Note that q was chosen from the open interval $]0, 1[$. Thus abstracting from probabilities, we have to expect that a medium drops or forwards the incoming message at any time. We are only interested in the behavior at some specific point in time, such that we can ignore any liveness properties and can consider the following specification for a single medium:

NDMedium[**type** T] — (a.s.)

in $i \in T^-$

out $o \in T^-$

$o.0 = \Box \ \wedge \ \forall t \colon o.(t + 1) \in \{\Box, i.t\}$

The composition of two instances of NDMedium is essentially the conjunction of the specifications. Thus, the media may arbitrarily drop or forward messages. In other words, using this abstraction, we have to assume the probability r to range between 0 and 1 (both inclusive).

This example shows, how the degree of abstraction influences the accuracy of the analysis. We summarize the results in the following table:

Abstraction Level	**Range of probability** r
prob., determ. composition with independencies	$\{q^2\}$
nondeterm., prob. composition	$[\max\{0, 2q - 1\}, q]$
nondeterm., non-prob. composition	$[0, 1]$

It is important to realize, that in both cases ($\mathsf{N}_1 \oplus \mathsf{N}_2$ and the composition of NDMedium) the probability r is not fixed over time. By nondeterminism, a different probability may be chosen at each time step and system-run.

6.3.2 Adding Independence

Independence of Outputs Given a measure $\mu\colon \overline{O}^P$ and three sets of output variables $X, Y, Z \subseteq O$, we say that X *and* Y *are independent given* Z in μ, written $\mu \models (X \perp Y \mid Z)$, if:

$$\begin{aligned}
&\forall n \in \mathbb{N}, x \in \overline{X}^n, y \in \overline{Y}^n, z \in \overline{Z}^n\colon\\
&\quad \mathbf{P}[z \sqsubseteq \mu|_Z] > 0 \implies\\
&\quad \mathbf{P}\big[x \uplus y \sqsubseteq \mu|_{X\cup Y} \bigm| z \sqsubseteq \mu|_Z\big] = \mathbf{P}\big[x \sqsubseteq \mu|_X \bigm| z \sqsubseteq \mu|_Z\big] \cdot \mathbf{P}\big[y \sqsubseteq \mu|_Y \bigm| z \sqsubseteq \mu|_Z\big] \quad .
\end{aligned}$$

The expression $X \perp Y \mid Z$ is a conditional independency. For a set L of such independencies, we say μ fulfills L and write $\mu \models L$ if:

$$\forall l \in L\colon \mu \models l \quad .$$

Instead of $X \perp Y \mid \emptyset$ we may also write $X \perp Y$.

Restricted Composition with Independence We can now introduce the composition of nondeterministic components $A_k, k \in \{1, \dots, n\}$, restricted by a set of independencies L, denoted by $\prod_k^L A_k$. We define:

$$\begin{aligned}
\forall i \in \bar{I}^\infty\colon \mu \in (\textstyle\prod_k^L A_k)(i)&\\
\iff \mu \in (\textstyle\prod_k A_k)(i) \wedge \mu \models L& \quad .
\end{aligned}$$

Instead of explicitly referring to the components' output channels, we sometimes express independencies in terms of components and implicitly mean their outputs. For example, in the context of $\prod_k^L A_k$ with $A_k\colon I_k \overset{P}{\rhd\!\!>} O_k$, we write $L = \{A_1 \perp A_2 \mid A_3\}$ instead of $L = \{O_1 \perp O_2 \mid O_3\}$.

For the restricted composition of two components we write instead $A_1 \oplus^L A_2$ and further abbreviate the independent composition of two components $A_1 \oplus^{\{A_1 \perp A_2\}} A_2$ with $A_1 \oplus^\perp A_2$.

6.3.3 Properties of Parallel Composition

Relation between Deterministic and Nondeterministic Components It follows immediately that the independent composition of two components subsumes the composition of each pair of deterministic realizations:

Corollary 2 *For all composable nondeterministic, probabilistic components A and B it holds*

$$[\![A \oplus^\perp B]\!] \supseteq \{a \oplus b \mid a \in [\![A]\!] \wedge b \in [\![B]\!]\} \quad . \qquad \square$$

In general, the equality does not hold. Given a realization of $A \oplus B$, its output O_A can depend on the input I_B and respectively output O_B can depend on the input I_A. This cannot be prevented given the definition of nondeterministic behavior functions and their realizations. The presented parallel composition is already the most specific one that is possible (i.e., the sets $(A \oplus B)(i)$ are the smallest possible). For the independent case, this is shown with the following theorem:

Theorem 9 *Given fully realizable components A and B. Then, for all inputs $i \in \overline{I_A \cup I_B}^{\infty}$, it holds:*

$$(A \oplus^{\perp} B)(i) = \{(a \oplus b)(i) \mid a \in [\![A]\!] \wedge b \in [\![B]\!]\} \quad . \qquad \square$$

Proof Given $\mu \in (A \oplus^{\perp} B)(i)$, we have by definition of parallel composition

$$\begin{aligned} &\mu|_{O_A} \in A(i|_{I_A}) \wedge \mu|_{O_B} \in B(i|_{I_B}) \\ \wedge\ &\forall o \in \overline{O_A \cup O_B}^{*} \colon \mathbf{P}[\mu \sqsupseteq o] = \mathbf{P}[\mu|_{O_A} \sqsupseteq o|_{O_A}] \cdot \mathbf{P}[\mu|_{O_B} \sqsupseteq o|_{O_B}] \quad . \end{aligned}$$

Because of the full realizability of A and B there exist behavior functions $a \in [\![A]\!]$ and $b \in [\![B]\!]$ such that

$$\mu|_{O_A} = a(i|_{I_A}) \ \wedge \ \mu|_{O_B} = b(i|_{I_B}) \quad .$$

It follows $\mu = (a \oplus b)(i)$ and the set inclusion $(A \oplus^{\perp} B)(i) \subseteq \{(a \oplus b)(i) \mid a \in [\![A]\!] \wedge b \in [\![B]\!]\}$ is proven.

To prove the other direction, let be given some realizations $a \in [\![A]\!]$ and $b \in [\![B]\!]$. For the measure $\mu = (a \oplus b)(i)$ it holds (for all $o \in \overline{O_A \cup O_B}^{*}$) :

$$\mu(o) = a(i|_{I_A})(o|_{O_A}) \cdot b(i|_{I_B})(o|_{O_B}) \quad .$$

It follows

$$\begin{aligned} &\mu|_{O_A} = a(i|_{I_A}) \in A(i|_{I_A}) \\ \wedge\ &\mu|_{O_B} = b(i|_{I_B}) \in B(i|_{I_B}) \\ \wedge\ &\mu \models \{O_A \perp O_B\} \quad . \end{aligned}$$

Thus, it is $\mu \in (A \oplus^{\perp} B)(i)$ and the proof is complete. ■

Theorem 10 *For composable and strongly causal deterministic, probabilistic behavior functions a and b it holds:*

$$a^N \oplus^{\perp} b^N = (a \oplus b)^N \quad . \qquad \square$$

Proof For all input sequences $i \in \overline{(I_a \cup I_b)}^\infty$ it holds:

$$\begin{aligned}
&\mu \in (a^N \obar^\perp b^N)(i) \\
&\qquad \iff \mu|_{O_a} = a(i|_{I_a}) \;\wedge\; \mu|_{O_b} = b(i|_{I_b}) \\
&\qquad\quad \wedge\, \forall o \in \overline{O_A \cup O_B}^* \colon \mathbf{P}[\mu \sqsupseteq o] = \mathbf{P}[\mu|_{O_a} \sqsupseteq o|_{O_a}] \cdot \mathbf{P}[\mu|_{O_b} \sqsupseteq o|_{O_b}] \\
&\qquad \iff \mu = (a \obar b)(i) \quad .
\end{aligned}$$

■

Non-Probabilistic Abstraction The next theorem shows that the parallel composition provided by Focus is indeed a non-probabilistic abstraction of the just defined parallel composition. The probabilistic translation of Focus behavior functions is a homomorphism with respect to the parallel composition $\obar$.

Theorem 11 *For nondeterministic, non-probabilistic behavior functions F_1 and F_2, it is*

$$F_1^P \obar F_2^P = (F_1 \obar F_2)^P \quad .$$

□

Proof Using

$$\begin{aligned}
(F_1 \obar F_2)(i) &= \{o_1 \uplus o_2 \mid o_1 \in F_1(i|_{I_1}) \wedge o_2 \in F_2(i|_{I_2})\} \\
&= F_1(i|_{I_1}) \uplus F_2(i|_{I_2})
\end{aligned}$$

we have the equivalences

$$\begin{aligned}
\mu \in (F_1 \obar F_2)^P(i) &\iff \mu((F_1 \obar F_2)(i)) = 1 \\
&\iff \mu(F_1(i|_{I_1}) \uplus F_2(i|_{I_2})) = 1 \\
&\iff \mu|_{O_1}(F_1(i|_{I_1})) = 1 \wedge \mu|_{O_2}(F_2(i|_{I_2})) = 1 \\
&\iff \mu|_{O_1} \in F_1^P(i|_{I_1}) \wedge \mu|_{O_2} \in F_2^P(i|_{I_2}) \\
&\iff \mu \in (F_1^P \obar F_2^P)(i) \quad .
\end{aligned}$$

■

6.4 Feedback

Now, we want to define the feedback of nondeterministic components, i.e., for a nondeterministic behavior A the feedback $A^{\circlearrowleft}$. Connecting output channels to identical input channels leads to the syntactic interface $A^{\circlearrowleft} \colon (I \setminus O) \overset{P}{\rhd} O$ for some $A \colon I \overset{P}{\rhd} O$.

We have already defined the feedback operation for the deterministic case. As nondeterministic components are an abstraction of deterministic ones, we expect also that this holds for the feedback operation. That means, given a realization $f \in [\![A]\!]$ of a nondeterministic component A, we obtain the component $f^{\circlearrowleft}$ by application of the deterministic feedback operation. If we translate the operation to the nondeterministic theory, we expect that

$$f^{\circlearrowleft} \in [\![A^{\circlearrowleft}]\!]$$

or equivalently

$$\forall i\colon f^{\circlearrowleft}(i) \in A^{\circlearrowleft}(i)$$

holds. This leads to the following lower-bound on the definition of the feedback operator in the nondeterministic case:

$$A^{\circlearrowleft}(i) \supseteq \{f^{\circlearrowleft}(i) \mid f \in [\![A]\!]\} \quad . \tag{6.3}$$

The question is, if we should define the operator to equal this lower-bound or if we should include some additional behavior.

Thinking of an operationalization of the feedback operation, the idea is, that a system implementing the nondeterministic component produces some output distribution. Then, we would feedback each of these possible outputs in order to obtain the behavior of the implementation for the next time step. The important point here is, that whatever state the system had at this point of time, it has to continue from there. A system cannot undo the past during a feedback operation and furthermore it has to behave strongly causal. These are exactly the constraints we captured in the definition of realizations. Thus, by looking at the feedback of nondeterministic components in a step-wise manner and in order to obtain the most concrete abstraction as possible, we claim that the feedback operation should indeed equal the lower-bound (6.3).

We define the **feedback of a probabilistic and nondeterministic behavior** A to be the minimal nondeterministic behavior containing the deterministic feedback of each of A's realizations:

$$A^{\circlearrowleft}(i) = \{f^{\circlearrowleft}(i) \mid f \in [\![A]\!]\} \quad .$$

Obviously we have the relation

$$[\![A^{\circlearrowleft}]\!] \supseteq \{f^{\circlearrowleft} \mid f \in [\![A]\!]\} \quad .$$

But we can show, that the realizations on the left contain more components than the set one the right. This means, that the representation of feedback systems as behavior functions is not complete and indeed an abstraction. In other words, feedback systems have certain constraints about their behavior on different inputs and these constraints not captured by the representation as behavior functions. Because of these lost constraints, realizations are allowed that correspond to no feedback system. To understand the background of this explanation, consider the following theorem and its proof:

Theorem 12 *It exists a fully realizable, nondeterministic and probabilistic behavior function A such that*

$$[\![A^{\circlearrowleft}]\!] \neq \{f^{\circlearrowleft} \mid f \in [\![A]\!]\} \quad . \qquad \square$$

Proof In this proof, we consider a certain system A and a suitable realization f of $A^{\circlearrowleft}$.

The component A has two alternative behaviors. Furthermore, it has two input channels and one output channel that will be fed back to one of its inputs. The alternatives diverge at the second time step in a way that will not be visible after applying the feedback operator. Beginning with the third time step, the difference is also visible after the feedback.

Figure 6.3 depicts, from top to bottom, the nondeterministic probabilistic behavior A and the feedback $A^{\circlearrowleft}$. There, we see that the two alternatives are not distinguishable before time step three, therefore the third graph shows a simplification of $A^{\circlearrowleft}$. The simplification also relies on the fact, that a nondeterministic behavior is defined per input and cannot restrict the possible combinations of behavior regarding several inputs. The fourth graph shows a realization g of $A^{\circlearrowleft}$. For this realization g, there is no realization f of A such that g is the feedback of f, i.e., $g = f^{\circlearrowleft}$. The full-realizability of A is obvious.

Formally, we define the behavior $A\colon \{x, y\} \overset{P}{\rhd\!\!\!>} \{x\}$ with $\mathbf{type}(x) = \{0, 1\}$ and $\mathbf{type}(y) = \{a, b\}$ by

$$A(i) = \begin{cases} \Gamma(\{d_1, d_2\}) & \text{if } i.0 = (x \mapsto 0, y \mapsto a) \\ \{d_1\} & \text{otherwise} \end{cases}$$

where (dropping the channel annotation)

$$d_1 = \begin{pmatrix} \langle 0\rangle \cdot \langle 0\rangle^\infty \mapsto 1/2 \\ \langle 1\rangle \cdot \langle 0\rangle^\infty \mapsto 1/2 \end{pmatrix} \text{ and } d_2 = \begin{pmatrix} \langle 0, 0, 1\rangle \cdot \langle 0\rangle^\infty \mapsto 1/2 \\ \langle 1, 1, 1\rangle \cdot \langle 0\rangle^\infty \mapsto 1/2 \end{pmatrix} \quad .$$

Next, we consider the realizations of A. For every realization $f \in [\![A]\!]$ and every i, there exists some $q(i) \in [0, 1]$ such that

$$f(i) = \begin{cases} \big(q(i) \cdot d_1 + (1 - q(i)) \cdot d_2\big)(o) & \text{if } i.0 = (x \mapsto 0, y \mapsto a) \\ d_1(o) & \text{otherwise.} \end{cases}$$

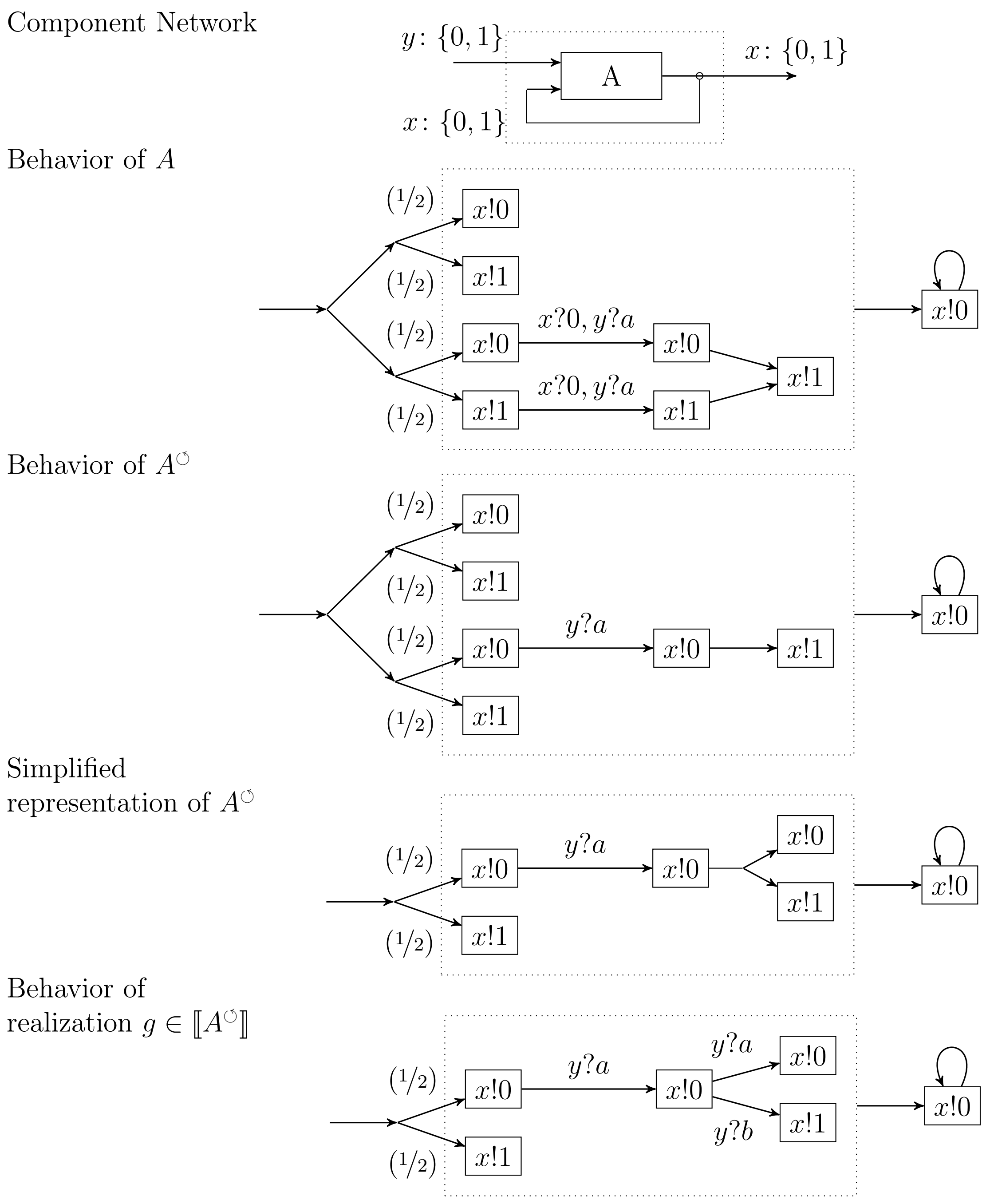

Component Network
$y\colon \{0,1\}$
$x\colon \{0,1\}$
A
$x\colon \{0,1\}$
Behavior of A
(1/2)
$x!0$
$x!1$
$x?0, y?a$
$x!1$
Behavior of $A^{\circlearrowleft}$
$y?a$
Simplified representation of $A^{\circlearrowleft}$
Behavior of realization $g \in [\![A^{\circlearrowleft}]\!]$
$y?b$

Figure 6.3: Component A, the feedback $A^{\circlearrowleft}$ and the realization g of $A^{\circlearrowleft}$.

The distribution in the first case can be written more concisely as (with an appropriate choice of $p \in [0,1]$)

$$d_{3,p} = \begin{pmatrix} \langle 0\rangle \cdot \langle 0\rangle^\infty & \mapsto p/2 \\ \langle 0,0,1\rangle \cdot \langle 0\rangle^\infty & \mapsto (1-p)/2 \\ \langle 1\rangle \cdot \langle 0\rangle^\infty & \mapsto p/2 \\ \langle 1,1,1\rangle \cdot \langle 0\rangle^\infty & \mapsto (1-p)/2 \end{pmatrix} .$$

Thereby, the probabilities $q(i)$ cannot be chosen arbitrarily but must respect the causality constraint. Consider any input i with $i.0 = (x \mapsto 0, y \mapsto a)$, then the causality requires that all initial behaviors $f(i){\downarrow_2}$ must be identical. With

$$f(i){\downarrow_2} = d_{3,q(i)}{\downarrow_2} = \begin{pmatrix} \langle 0,0\rangle \mapsto 1/2 \\ \langle 1,0\rangle \mapsto q(i)/2 \\ \langle 1,1\rangle \mapsto (1-q(i))/2 \end{pmatrix}$$

we conclude, that all $q(i)$ are identical and can be identified with a single $q \in [0,1]$.

Then we obtain for the feedback of f:

$$f^{\circlearrowleft}(j)(o) = f(j \uplus o)(o) = \begin{cases} d_{3,q}(o) & \text{if } o.0.x = 0 \wedge j.0.y = a \\ d_1(o) & \text{otherwise} \end{cases}$$

which can be simplified to

$$f^{\circlearrowleft}(j) = \begin{cases} d_{4,q} & \text{if } j.0.y = a \\ d_1 & \text{otherwise} \end{cases} \tag{6.4}$$

where (again dropping the channel annotation)

$$d_{4,p} = \begin{pmatrix} \langle 0\rangle \cdot \langle 0\rangle^\infty & \mapsto p/2 \\ \langle 0,0,1\rangle \cdot \langle 0\rangle^\infty & \mapsto (1-p)/2 \\ \langle 1\rangle \cdot \langle 0\rangle^\infty & \mapsto 1/2 \end{pmatrix} .$$

This result corresponds to the second transition diagram in Figure 6.3 which emphasizes that the nondeterministic decision is independent of the input.

Now we can express each $A^{\circlearrowleft}(j)$ as the convex closure of finitely many distributions:

$$\begin{aligned} A^{\circlearrowleft}(j) &= \{f^{\circlearrowleft}(j) \mid f \in [\![A]\!]\} \\ &= \begin{cases} \Gamma(\{d_{4,0}, d_{4,1}\}) & \text{if } j.0.y = a \\ \{d_1\} & \text{otherwise} \end{cases} \end{aligned}$$

which agrees with the third transition diagram.

Finally, we choose the following realization $g \in [\![A^{\circlearrowleft}]\!]$, which is also shown at the bottom of Figure 6.3. Let, for all j,

$$g(\langle a,a\rangle \cdot j) = d_{4,1} = d_1 \qquad\qquad g(\langle a,b\rangle \cdot j) = d_{4,0}$$
$$g(\langle b\rangle \cdot j) = d_1 \quad .$$

If we compare this to the characterization of the feedbacks $f^{\circlearrowleft}$ in Equation (6.4), we find that $f^{\circlearrowleft}(j) \neq g(j)$ for $j = \langle a,a,\ldots\rangle$ or $j = \langle a,b,\ldots\rangle$. This shows that $g \neq f^{\circlearrowleft}$ for all $f \in [\![A]\!]$ and $[\![A^{\circlearrowleft}]\!] \neq \{f^{\circlearrowleft} \mid f \in [\![A]\!]\}$ as claimed. ■

The Dummy Component D_J We introduce the dummy component D_J with the syntactic interface $J \overset{P}{\rhd\!\!>} \emptyset$. The set $\bar{\emptyset}^\infty$ contains only the infinite sequence of empty valuations $(\,)^\infty = \langle (\,), (\,), \ldots\rangle$. Accordingly, there is only a single distribution in $\bar{\emptyset}^P = \{((\,)^\infty \mapsto 1)\}$. We define D_J to be non-empty for all inputs, i.e., $D_J(\varphi) = \bar{\emptyset}^P$ for all $\varphi \in \bar{J}^\infty$. By composition, this component can be used for extending the set of input channels of an other component.

We show the following important property, which we will use in the subsequent examples.

Theorem 13 *Let* $A\colon I \overset{P}{\rhd\!\!>} O$ *and* $J \cap O \subseteq I \cap O$ *then*

$$(A \oplus D_J)^{\circlearrowleft} = A^{\circlearrowleft} \oplus D_J \quad . \qquad \square$$

PROOF We have for all $i' \in I \setminus O$ and $j' \in J \setminus O$ with $i'|_{I\cap J\setminus O} = j'|_{I\cap J\setminus O}$:

$$\begin{aligned}
&\mu \in (A \oplus D_J)^{\circlearrowleft}(i' \uplus j')\\
\iff &\exists f\colon I \cup J \overset{P}{\rhd} O\colon \mu = f^{\circlearrowleft}(i' \uplus j') \ \wedge\ f \in [\![A \oplus D_J]\!]\\
\iff &\exists f\colon I \cup J \overset{P}{\rhd} O\colon \forall o\colon \mu(o) = f(i' \uplus j' \uplus o)(o) \ \wedge\ \forall i,j\colon f(i \uplus j) \in A(i)\\
\iff &\exists f\colon I \overset{P}{\rhd} O\colon \forall o\colon \mu(o) = f(i' \uplus o)(o) \ \wedge\ \forall i\colon f(i) \in A(i)\\
\iff &\exists f\colon I \overset{P}{\rhd} O\colon \mu = f^{\circlearrowleft}(i') \ \wedge\ f \in [\![A]\!]\\
\iff &\mu \in A^{\circlearrowleft}(i') \ \wedge\ \mu|_{\emptyset} \in D_J(j')\\
\iff &\mu \in (A^{\circlearrowleft} \oplus D_J)(i', j') \quad .
\end{aligned}$$
■

Example 6.3 In the previous Theorem 12, the constructed component A had two input channels. We will give now a second example system, which highlights that this second channel is only needed to allow the combination of behaviors during the composition.

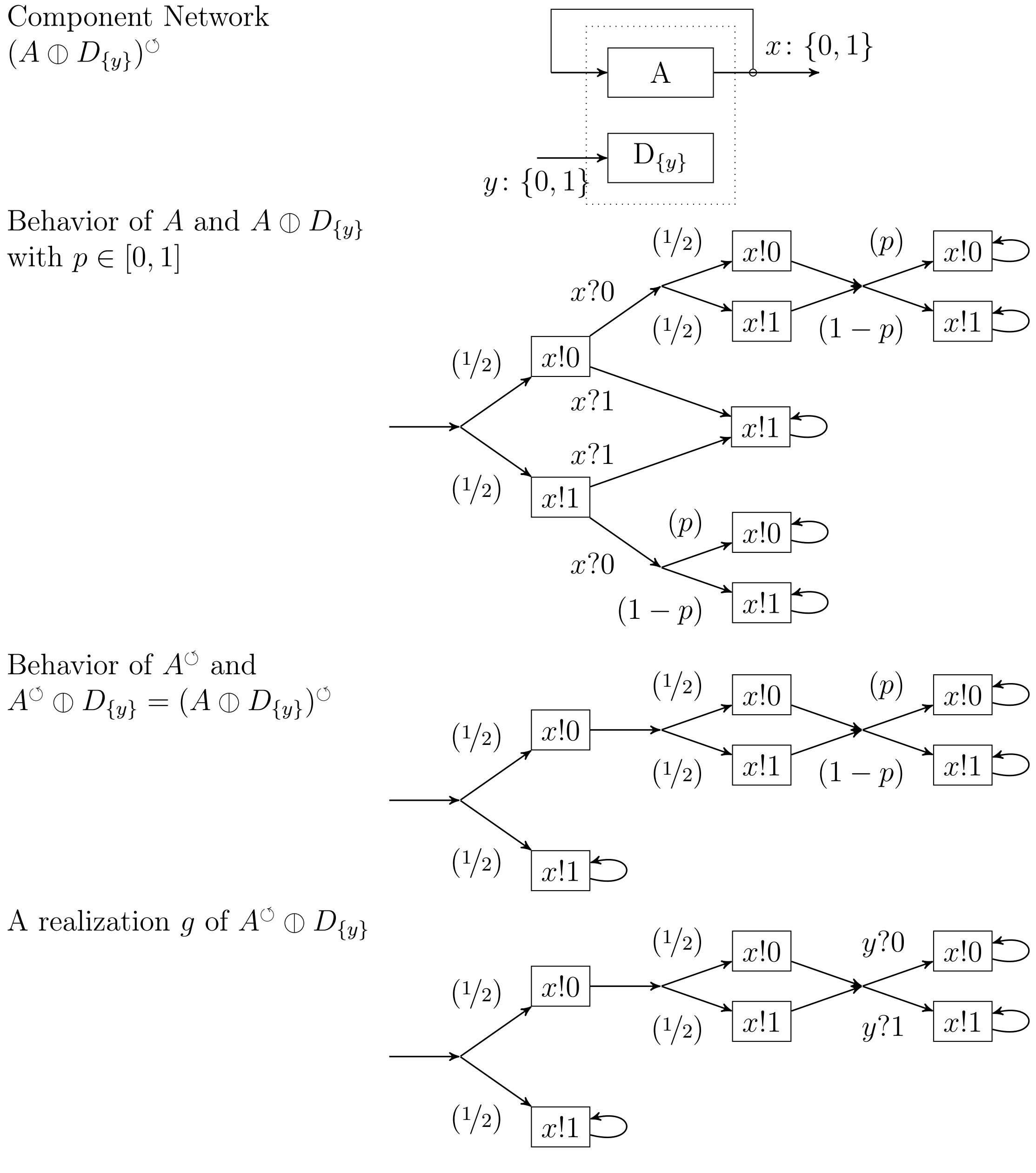

Component Network
$(A \obar D_{\{y\}})^{\circlearrowleft}$
A
$x\colon \{0,1\}$
$\mathrm{D}_{\{y\}}$
$y\colon \{0,1\}$
Behavior of A and $A \obar D_{\{y\}}$
with $p \in [0,1]$
$(1/2)$
$x!0$
$x?0$
$x?1$
$x!1$
(p)
$(1-p)$
Behavior of $A^{\circlearrowleft}$ and
$A^{\circlearrowleft} \obar D_{\{y\}} = (A \obar D_{\{y\}})^{\circlearrowleft}$
A realization g of $A^{\circlearrowleft} \obar D_{\{y\}}$
$y?0$
$y?1$

Figure 6.4: Component A, the feedback $A^{\circlearrowleft}$ and the realization g of $A^{\circlearrowleft} \obar D_{\{y\}}$.

The system is illustrated in Figure 6.4. The central component A has the single channel x both as input and output. The behavior is shown in the first transition diagram, where p can assume any value in the range $[0, 1] \subset \mathbb{R}$.

As the second component, we use the dummy component $D_{\{y\}}$ with the syntactic interface $\{y\} \overset{P}{\rhd} \emptyset$. For the sake of completeness, let $\mathbf{type}(y) = \{0, 1\}$. This component is used to extend the set of input channels such that, after applying the feedback operation, there are input channels that allow the combination of various distributions. The composition $A \obar D_{\{y\}}$ of both components will play the role of the component A in the previous example.

The behavior of the composition is also visualized by the first transition diagram. As we can see, for every input sequence $x \uplus y$, we obtain a range of possible output distributions $(A \obar D_{\{y\}})(x \uplus y)$, one for each value of $p \in [0, 1]$. The distributions are constructed in such a way that the probability in the lower branch at depth 2 is linked to the probability of the upper branch at depth 3. This linkage ensures that a realization of $A \obar D_{\{y\}}$ which shows some output distribution up to time step 2 has to repeat this probability on the other branch at time step 3, no matter what is the input at time 2. Thus, this linkage prevents arbitrary mixing of the two possible behaviors.

The second transition diagram shows the behavior after applying the feedback operation. Note the equality $(A \obar D_{\{y\}})^{\circlearrowleft} = A^{\circlearrowleft} \obar D_{\{y\}}$ shown in the preceding theorem. The linkage disappears or becomes irrelevant, as the lower branch is not observable anymore. This allows combinations of the distributions that were not possible previously. Consider, for example, the realization g of $A^{\circlearrowleft} \obar D_{\{y\}}$ visualized by the last transition diagram. If the second input on channel y equals 0, the component g chooses $p = 1$ and $p = 0$ otherwise. Formally, we have

$$g(\langle y\colon 0, 0\rangle) = \begin{pmatrix} \langle 0,0,0\rangle \mapsto 1/4 \\ \langle 0,1,0\rangle \mapsto 1/4 \\ \langle 1,1,1\rangle \mapsto 1/2 \end{pmatrix} \quad \text{and} \quad g(\langle y\colon 0, 1\rangle) = \begin{pmatrix} \langle 0,0,1\rangle \mapsto 1/4 \\ \langle 0,1,1\rangle \mapsto 1/4 \\ \langle 1,1,1\rangle \mapsto 1/2 \end{pmatrix} \quad .$$

We want to show, that there exists no $f \in [\![A \obar D_{\{y\}}]\!]$, such that $f^{\circlearrowleft} = g$. For this equality to hold, it is necessary that

$$f\left(\left\langle \begin{array}{c} x\colon 0,0 \\ y\colon 0,0 \end{array}\right\rangle\right)(\langle 0,0,0\rangle) = g(\langle y\colon 0,0\rangle)(\langle 0,0,0\rangle) = 1/4 \tag{6.5}$$

and

$$f\left(\left\langle \begin{array}{c} x\colon 0,0 \\ y\colon 0,1 \end{array}\right\rangle\right)(\langle 0,0,0\rangle) = g(\langle y\colon 0,1\rangle)(\langle 0,0,0\rangle) = 0 \quad . \tag{6.6}$$

Thus, let any realization $f \in [\![A \oplus D_{\{y\}}]\!]$ be given. By the definition of realization, we have

$$\forall \varphi \in \overline{\{y\}}^2 \colon f(\langle x\colon 0,0\rangle \uplus \varphi) \in A(\langle x\colon 0,0\rangle) = \left\{ \begin{pmatrix} \langle 0,0,0\rangle \mapsto {}^{p}\!/_{4} \\ \vdots \\ \langle 1,0,0\rangle \mapsto {}^{p}\!/_{2} \\ \langle 1,0,1\rangle \mapsto 0 \end{pmatrix} \;\middle|\; p \in [0,1] \right\}.$$

By strong causality, it follows

$$\begin{aligned} &\forall \varphi \in \overline{\{y\}}^2\ \exists p \in [0,1] \colon \\ &\qquad f(\langle x\colon 0\rangle \uplus \varphi{\downarrow_1})(\langle 1,0\rangle) = f(\langle x\colon 0,0\rangle \uplus \varphi)(\langle 1,0,0\rangle) = {}^{p}\!/_{2} \\ &\qquad \wedge\ f(\langle x\colon 0,0\rangle \uplus \varphi)(\langle 0,0,0\rangle) = {}^{p}\!/_{4} \quad . \end{aligned}$$

By instantiating these relations once with $\varphi = \langle y\colon 0,0\rangle$ and once with $\varphi = \langle y\colon 0,1\rangle$, we finally have

$$\begin{aligned} &\forall \varphi \in \overline{\{y\}}^2, p \in [0,1] \colon \\ &\qquad f\left(\left\langle \begin{matrix} x\colon 0,0 \\ y\colon 0,0 \end{matrix} \right\rangle\right)(\langle 0,0,0\rangle) = {}^{p}\!/_{4} \\ &\iff f\left(\left\langle \begin{matrix} x\colon 0 \\ y\colon 0 \end{matrix} \right\rangle\right)(\langle 1,0\rangle) = {}^{p}\!/_{2} \\ &\iff f\left(\left\langle \begin{matrix} x\colon 0,0 \\ y\colon 0,1 \end{matrix} \right\rangle\right)(\langle 0,0,0\rangle) = {}^{p}\!/_{4} \quad . \end{aligned}$$

This proves our claim that the Equations (6.5) and (6.6) cannot hold simultaneously. Thus there is no such realization f that $f^{\circlearrowleft} = g$. □

Nondeterministic Abstraction Let $f\colon I \overset{P}{\rhd} O$ be some deterministic and probabilistic component. We apply the feedback operation to the nondeterministic translation f^N and obtain the following simplifications:

$$\begin{aligned} g \in [\![f^N]\!] &\iff \forall i\colon g(i) \in f^N(i) = \{f(i)\} \\ &\iff g = f \end{aligned}$$

and it follows

$$(f^N)^{\circlearrowleft}(i) = \{g^{\circlearrowleft}(i) \mid g \in [\![f^N]\!]\} = \{f^{\circlearrowleft}(i)\} = (f^{\circlearrowleft})^N(i) \quad .$$

This proves the equality $(f^N)^{\circlearrowleft} = (f^{\circlearrowleft})^N$ and we obtain the following theorem.

Theorem 14 *Given a strongly causal, nondeterministic, probabilistic behavior function $f : I \rhd\!\!> O$, it holds*

$$(f^N)^{\circlearrowleft} = (f^{\circlearrowleft})^N \quad .$$

□

Non-Probabilistic Abstraction The next theorem shows that the feedback operation in Focus is a non-probabilistic abstraction of the just defined feedback operation on probabilistic behavior.

Theorem 15 *Given a non-probabilistic and nondeterministic behavior function $F : I \rhd\!\!> O$, it holds*

$$(F^P)^{\circlearrowleft} = (F^{\circlearrowleft})^P \quad . \qquad \square$$

Proof Let $J = I \setminus O$ and $j \in \overline{J}^\infty$ be some fixed input. We first prove that the left hand side is included in the right hand side.

"$\subseteq$": Let $n \in \mathbb{N}_0$ and $o \in \overline{O}^{n+1}$. With $\mu \in (F^P)^{\circlearrowleft}(j)$, we have $\mu(o) = f(j \uplus o{\downarrow_n})(o)$ for some $f \in [\![F^P]\!]$ and we derive the following implications:

$$\begin{aligned} & \mu(o) = f(j \uplus o{\downarrow_n})(o) > 0 \\ \Longrightarrow\ & o \in F(j \uplus o{\downarrow_n}) \\ \Longrightarrow\ & o \in F^{\circlearrowleft}(j{\downarrow_n}) \quad . \end{aligned}$$

Since this holds for all $o \in \overline{O}^{n+1}$ and $n \in \mathbb{N}_0$, it follows that $\mu(F^{\circlearrowleft}(j)) = 1$ or equivalently $\mu \in (F^{\circlearrowleft})^P(j)$.

"$\supseteq$": For the proof of the other inclusion, we need the **length of the longest common prefix** $s = \mathrm{gcl}(a, b)$ of two sequences $a, b \in X^\omega$ defined by

$$a{\downarrow_s} = b{\downarrow_s} \ \wedge\ \big(a(s+1) \neq b(s+1)\ \vee\ s = \max\{|a|, |b|\}\big) \quad .$$

The **valid continuation** of some output $x \in F(i){\downarrow_n}$ is defined by some function $x, i \mapsto \mathrm{vc}(x, i) \in \overline{O}^\infty$ with the properties

$$\begin{aligned} & \mathrm{vc}(x, i) \in F(i) \\ \wedge\ & x \sqsubseteq \mathrm{vc}(x, i) \\ \wedge\ & i{\downarrow_k} = i'{\downarrow_k} \implies \mathrm{vc}(x, i){\downarrow_{k+1}} = \mathrm{vc}(x, i'){\downarrow_{k+1}} \quad . \end{aligned}$$

Such a function can be constructed inductively because of the strong causality of F. Choose some arbitrary $\langle o_1 \rangle \in F(\langle\ \rangle)$ and set $\mathrm{vc}(x, \langle\ \rangle) = \langle o_1 \rangle$. Then for any $k \in \mathbb{N}_0$, $i \in \overline{I}^k$ and $i_{k+1} \in \overline{I}$, we assume $\mathrm{vc}(x, i) \in F(i)$ which implies that some o_{k+2} exists such that $\mathrm{vc}(x, i) \cdot \langle o_{k+2} \rangle \in F(i \cdot \langle i_{k+1} \rangle)$. We set $\mathrm{vc}(x, i \cdot \langle i_{k+1} \rangle) = \mathrm{vc}(x, i) \cdot \langle o_{k+2} \rangle$.

Given some $\mu \in (F^{\circlearrowleft})^P(j)$, we construct a behavior function $f \in [\![F^P]\!]$ such that $\mu = f^{\circlearrowleft}(j)$. We define, for all $n \in \mathbb{N}_0$, $q \in \overline{O}^n$ and $o \in \overline{O}^{n+1}$,

$$f(j \uplus q{\downarrow_n})(o) = \begin{cases} \mu(o{\downarrow_{s+1}}) & \text{if } s = \mathrm{gcl}(q, o)\ \wedge\ \mu(o{\downarrow_{s+1}}) > 0 \\ & \quad \wedge\ o = \mathrm{vc}(o{\downarrow_{s+1}}, j \uplus q){\downarrow_{n+1}} \\ 0 & \text{otherwise} \quad . \end{cases}$$

Note, since $\mu \in (F^{\circlearrowleft})^P(j)$, we have the implication

$$\mu(o) > 0 \implies o \in F(j \uplus o)$$

and in this case the valid continuation is defined. The equation defines indeed a function since the term $\mathrm{vc}(o{\downarrow_{s+1}}, j \uplus q){\downarrow_{n+1}}$ depends only on the first n elements of $j \uplus q$.

We show now that f is a well-defined behavior function. Therefore, we have to prove that

$$\sum_{a \in \bar{O}} f(\langle\,\rangle)(\langle a\rangle) = 1 \tag{6.7}$$

as well as

$$\sum_{a \in \bar{O}} f(j \uplus q{\downarrow_{n+1}})(o \cdot \langle a\rangle) = f(j \uplus q{\downarrow_n})(o) \quad . \tag{6.8}$$

The first Equation (6.7) follows directly from the definition. With

$$n = s = \mathrm{gcl}\big(\langle\,\rangle, \langle a\rangle\big) = 0$$

it follows

$$f(\langle\,\rangle)(\langle a\rangle) = \mu(\langle a\rangle) \quad .$$

For the second equation we consider three cases:

1. If $s = \mathrm{gcl}(q, o) = n + 1$, we have $q = o$ and:

$$\begin{aligned} f(j \uplus o{\downarrow_{n+1}})(o \cdot \langle a\rangle) &= \begin{cases} \mu(o \cdot \langle a\rangle) & \text{if } \mu(o \cdot \langle a\rangle) > 0 \\ & \ \wedge\ o \cdot \langle a\rangle = \mathrm{vc}(o \cdot \langle a\rangle, j \uplus o){\downarrow_{n+2}} \\ 0 & \text{otherwise} \end{cases} \\ &= \mu(o \cdot \langle a\rangle) \end{aligned}$$

   as the valid continuation of length $n + 2$ is $o \cdot \langle a\rangle$ itself. With the same argument, we further obtain:

$$f(j \uplus o{\downarrow_n})(o) = \mu(o)$$

   and the equality (6.8) follows. In particular, this case also proves $\mu = f^{\circlearrowleft}(j)$.

2. If $s = \mathrm{gcl}(q, o) \leq n$ the terms simplify to:

$$f(j \uplus q{\downarrow}_{n+1})(o \cdot \langle a \rangle) = \begin{cases} \mu(o{\downarrow}_{s+1}) & \text{if } \mu(o{\downarrow}_{s+1}) > 0 \\ & \ \wedge\ o \cdot \langle a \rangle = \mathrm{vc}(o{\downarrow}_{s+1}, j \uplus q){\downarrow}_{n+2} \\ 0 & \text{otherwise} \end{cases}$$

and

$$f(j \uplus q{\downarrow}_{n})(o) = \begin{cases} \mu(o{\downarrow}_{s+1}) & \text{if } \mu(o{\downarrow}_{s+1}) > 0 \\ & \ \wedge\ o = \mathrm{vc}(o{\downarrow}_{s+1}, j \uplus q){\downarrow}_{n+1} \\ 0 & \text{otherwise} \quad . \end{cases}$$

In the case that $o \neq \mathrm{vc}(o{\downarrow}_{s+1}, j \uplus q){\downarrow}_{n+1}$ it is also $o \cdot \langle a \rangle \neq \mathrm{vc}(o{\downarrow}_{s+1}, j \uplus q){\downarrow}_{n+2}$ and both sides of (6.8) equate to zero. Otherwise, both $f(j \uplus q{\downarrow}_n)(o)$ and the summand for $a = \mathrm{vc}(o{\downarrow}_{s+1}, j \uplus q).n + 1$ equal $\mu(o{\downarrow}_{s+1})$ and all other summands are zero. In both cases the proposition follows.

Thus, f is a well-defined behavior function.

Next we show $f \in [\![F^P]\!]$. This is the case if $f(i)(F(i)) = 1$ for all inputs $i \in \bar{I}^\infty$. Let $i = j \uplus q \in \bar{I}^\infty$ be some fixed input. Then we characterize f by the equation (for all $o \in \overline{O}^\infty$):

$$f(j \uplus q)(o) = \begin{cases} \mu(q{\downarrow}_k \cdot \langle a \rangle) & \text{if } a \neq q.k \ \wedge\ o = \mathrm{vc}(q{\downarrow}_k \cdot \langle a \rangle, j \uplus q) \\ & \text{for some } k \in \mathbb{N}_0, a \in \overline{O} \\ \mu(q) & \text{if } o = q \\ 0 & \text{otherwise.} \end{cases}$$

We know that $\mathrm{vc}(\cdot, j \uplus q) \in F(j \uplus q)$ and $q \in F(j \uplus q)$ if $\mu(q) > 0$. Furthermore, $\{q{\downarrow}_k \cdot \langle a \rangle \cdot \overline{O}^\infty \mid k \in \mathbb{N}_0 \wedge a \neq q.k\} \cup \{\{q\}\}$ is a partition of $\overline{O}^\infty$. We can then calculate:

$$\begin{aligned} f(j \uplus q)(F(j \uplus q)) &\geq f(j \uplus q)(q) + \sum_{k \in \mathbb{N}_0, a \neq q.k} f(j \uplus q)(\mathrm{vc}(q{\downarrow}_k \cdot \langle a \rangle, j \uplus q)) \\ &= \mu(q) + \sum_{k \in \mathbb{N}_0, a \neq q.k} \mu(q{\downarrow}_k \cdot \langle a \rangle) = 1 \quad . \end{aligned}$$

This shows $\mu \in (F^P)^{\circlearrowleft}(j)$. ■

6.5 General Composition

As in the deterministic case, we define the **general composition** of nondeterministic behaviors by $A \otimes B = (A \oplus B)^{\circlearrowleft}$.

Theorem 16 *Given some nondeterministic, non-probabilistic behaviors A and B, we have*

$$\begin{gathered}(A^P \obar B^P) = (A \obar B)^P \\ (A^P)^{\circlearrowleft} = (A^{\circlearrowleft})^P \\ (A^P \otimes B^P) = (A \otimes B)^P \quad .\end{gathered}$$

This shows, that the probabilistic translation $\cdot^P$ is a homomorphism from nondeterministic Focus *behavior functions to nondeterministic probabilistic behavior functions with respect to the three operators parallel composition $\obar$, feedback $\cdot^{\circlearrowleft}$ and general composition $\otimes$.* □

Proof The first two equalities were proven in the previous sections. By definition of the general composition it follows immediately that

$$(A^P \otimes B^P) = (A^P \obar B^P)^{\circlearrowleft} = ((A \obar B)^P)^{\circlearrowleft} = ((A \obar B)^{\circlearrowleft})^P = (A \otimes B)^P \quad . \quad \blacksquare$$

Theorem 17 *The nondeterministic translation $f \mapsto f^N$ is a homomorphism from deterministic, probabilistic components $(I \overset{P}{\rhd} O)$ to nondeterministic, probabilistic components $(I \overset{P}{\rhd\!\!\rhd} O)$ with respect to the pairs of operators $(\cdot^{\circlearrowleft}, \cdot^{\circlearrowleft}), (\obar, \obar^{\perp})$ and $(\otimes, \otimes^{\perp})$. In other words, given some deterministic, probabilistic behaviors f and g, we have the equations:*

$$\begin{gathered}f^N \obar^{\perp} g^N = (f \obar g)^N \\ (f^N)^{\circlearrowleft}(i') = (f^{\circlearrowleft})^N \\ (f^N \otimes^{\perp} g^N) = (f \otimes g)^N \quad .\end{gathered}$$

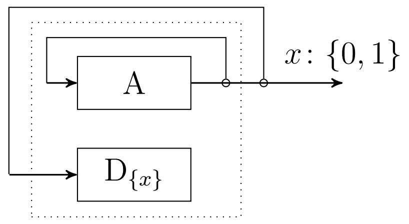
This shows, that the deterministic operators coincide with the nondeterministic operators assuming statistical independence between the components. □

Example 6.4 Using the same system A from Example 6.3, we can show that the general composition is not associative. From Theorem 13 about dummy components and the obvious equalities $A \obar D_{\{x\}} = A$ and $D_{\{x\}} \obar D_{\{x\}} = D_{\{x\}}$, we obtain the implications:

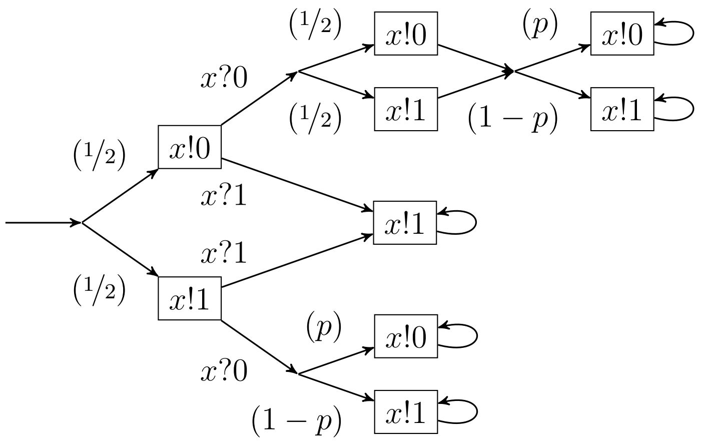
$$\begin{aligned}&(A^{\circlearrowleft} \obar D_{\{x\}})^{\circlearrowleft} \neq A^{\circlearrowleft} \\ \Longrightarrow\ &((A \obar D_{\{x\}})^{\circlearrowleft} \obar D_{\{x\}})^{\circlearrowleft} \neq (A \obar (D_{\{x\}} \obar D_{\{x\}})^{\circlearrowleft})^{\circlearrowleft} \\ \Longrightarrow\ &((A \otimes D_{\{x\}}) \otimes D_{\{x\}}) \neq (A \otimes (D_{\{x\}} \otimes D_{\{x\}})) \quad .\end{aligned}$$

Therefore, it is sufficient to prove the first inequality.

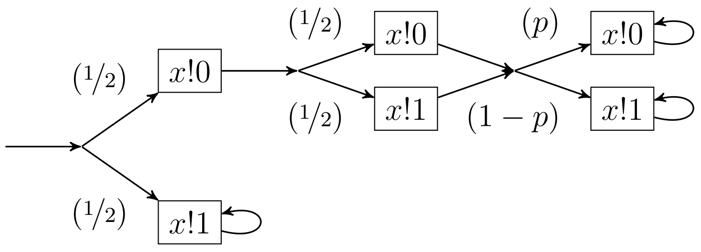
Component Network
$(A^{\circlearrowleft} \oplus D_{\{x\}})^{\circlearrowleft}$



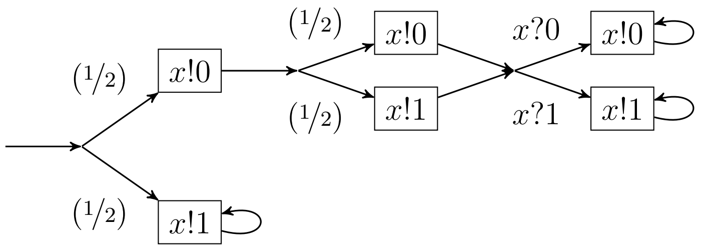
Behavior of A with $p \in [0,1]$



Behavior of $A^{\circlearrowleft}$



A realization g of $A^{\circlearrowleft} \oplus D_{\{x\}}$



Behavior of $g^{\circlearrowleft}$

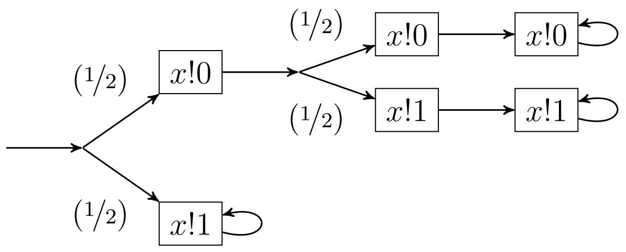


Figure 6.5: Illustrations for the example proving the general composition to be not associative.

In Figure 6.5, we show a realization $g \in [\![A^{\circlearrowleft} \obar D_{\{x\}}]\!]$, such that $g^{\circlearrowleft}(()^{\infty}) \notin A^{\circlearrowleft}(()^{\infty})$ although $g^{\circlearrowleft}(()^{\infty}) \in (A^{\circlearrowleft} \obar D_{\{x\}})^{\circlearrowleft}(()^{\infty})$. Note that the infinite sequence of empty valuations $()^{\infty}$ is the only input value for components without any input channels like $A^{\circlearrowleft}$. If we compare the visualizations of both $g^{\circlearrowleft}$ and $A^{\circlearrowleft}$, we easily see that there is no value for p, such that the two behaviors match. □

We conclude this result with the following theorem.

Theorem 18 *The general composition $\otimes$ of nondeterministic, probabilistic behavior functions is not associative.* □

Using the same example and argumentation, we can analogously conclude

Theorem 19 *The general independent composition $\otimes^{\perp}$ of nondeterministic, probabilistic behavior functions is not associative.* □

Example 6.5 Assume two persons Bob and Lisa. Bob has a coin he tosses. Bob announces the result if Lisa asks for it and is silent otherwise. We will model this interaction once assuming the actions of Bob and Lisa to be independent. That means that any other flow of information and dependence than the one explained is prohibited. For comparison, we will also model this interaction a second time, this time allowing any arbitrary dependence, for example that Lisa knows the result of the coin tossing before deciding to ask for it. In both cases, we ask for the probability that Bob announces "head" when Lisa asked for the result.

We model Lisa as a nondeterministic behavior and Bob as a deterministic and probabilistic behavior. Note that we are only interested in the interaction within the first two time steps. Therefore, we will not analyze the behavior for later time steps. We use as channel types the sets $\texttt{YN} = \{\texttt{y}, \texttt{n}\}$ ("yes", "no") and $\texttt{HT} = \{\texttt{h}, \texttt{t}, \Box\}$ ("head", "tail", "nothing").

Bob — det. prob

in $c \in \texttt{YN}$
out $o \in \{\texttt{h}, \texttt{t}, \Box\}$

init

o	Prob
$\Box$	1

c	o'	Prob
y	h	1/2
y	t	1/2
n	$\Box$	1

Lisa — (a.s.)

out $c \in \texttt{YN}$

The parallel composition with dependencies Lisa $\obar$ Bob yields the set of all joint distributions over the random variables $c \in \texttt{YN}^{\infty}$ and $o(u) \in \texttt{HT}^{\infty}$, for all $u \in \texttt{YN}^{\infty}$, with the following restrictions (only for the first two time steps and omitting the angle brackets $\langle\rangle$ of sequences):

$$\mathbf{P}[o(\langle\rangle).0 = \Box] = 1 \qquad \mathbf{P}[o(\texttt{n}).1 = \Box] = 1 \tag{6.9}$$

$$\mathbf{P}[o(\texttt{y}).1 = \texttt{h}] = 1/2 \qquad \mathbf{P}[o(\texttt{y}).1 = \texttt{t}] = 1/2 \quad . \tag{6.10}$$

Applying the feedback operation means to introduce random variables $\tilde{c} \in \mathtt{YN}^\infty$ and $\tilde{o} \in \mathtt{HT}^\infty$ such that (for all $u \in \mathtt{YN}^\infty, v \in \mathtt{HT}^\infty$)

$$\mathbf{P}[\tilde{c} = u \wedge \tilde{o} = v] = \mathbf{P}[c = u \wedge o(u) = v]$$

and to project each joint distribution of the parallel composition onto these new variables.

The probability we asked for corresponds to $\mathbf{P}[\tilde{o}.1 = \mathtt{h} | \tilde{c}.0 = \mathtt{y}]$ and can take on any value between 0 and 1 (both inclusive). The following two distributions $\mathbf{P}_1$ (Lisa says "yes" if Bob throws "Head" and "no" otherwise) and $\mathbf{P}_2$ (Lisa says "yes" if Bob throws "Tail" and "no" otherwise) explain these extreme probabilities. All other cases can be explained by their combination:

$$\begin{array}{ll}
\mathbf{P}_1[c.0 = \mathtt{y} \wedge o(\mathtt{y}).1 = \mathtt{h}] = 1/2 & \mathbf{P}_2[c.0 = \mathtt{y} \wedge o(\mathtt{y}).1 = \mathtt{h}] = 0 \\
\mathbf{P}_1[c.0 = \mathtt{y} \wedge o(\mathtt{y}).1 = \mathtt{t}] = 0 & \mathbf{P}_2[c.0 = \mathtt{y} \wedge o(\mathtt{y}).1 = \mathtt{t}] = 1/2 \\
\mathbf{P}_1[c.0 = \mathtt{n} \wedge o(\mathtt{y}).1 = \mathtt{h}] = 0 & \mathbf{P}_2[c.0 = \mathtt{n} \wedge o(\mathtt{y}).1 = \mathtt{h}] = 1/2 \\
\mathbf{P}_1[c.0 = \mathtt{n} \wedge o(\mathtt{y}).1 = \mathtt{t}] = 1/2 & \mathbf{P}_2[c.0 = \mathtt{n} \wedge o(\mathtt{y}).1 = \mathtt{t}] = 0 \quad .
\end{array}$$

For the independent composition $\mathsf{Lisa} \oplus^\perp \mathsf{Bob}$, we have to add to the equations (6.9)-(6.10) the additional independence restriction

$$\mathbf{P}[c.0 = u \wedge o(\mathtt{y}).1 = v] = \mathbf{P}[c.0 = u] \cdot \mathbf{P}[o(\mathtt{y}).1 = v] \tag{6.11}$$

$$\mathbf{P}[c.0 = u \wedge o(\mathtt{n}).1 = v] = \mathbf{P}[c.0 = u] \cdot \mathbf{P}[o(\mathtt{n}).1 = v] \tag{6.12}$$

of which the second (6.12) follows already from (6.9). Again, we introduce variables $\tilde{c}$ and $\tilde{o}$ to handle the feedback. It follows that

$$\begin{aligned}
\mathbf{P}[\tilde{o}.1 = \mathtt{h} | \tilde{c}.0 = \mathtt{y}] &= \mathbf{P}[\tilde{c}.0 = \mathtt{y} \wedge \tilde{o}.1 = \mathtt{h}] / \mathbf{P}[\tilde{c}.0 = \mathtt{y}] \\
&= \mathbf{P}[c.0 = \mathtt{y} \wedge o(\mathtt{y}).1 = \mathtt{h}] / \mathbf{P}[c.0 = \mathtt{y}] \\
&= \mathbf{P}[o(\mathtt{y}).1 = \mathtt{h}] = 1/2 \quad .
\end{aligned}$$

□

Chapter 7

Refinement

So far we have considered the description of systems, in particular probabilistic ones, in a formal way. This description captures both the syntactic interface of systems (i.e., which data can the system send to and receive from its environment) and the interface behavior of systems (i.e., which interactions of the system with its environment are allowed).

The next step is to relate this system description to the process of system development. Instead of describing one particular system, we have to consider transitions between systems as they appear during the development. Typical activities in the development process are the requirements engineering and the design phase. The former iteratively extends the system specification until a satisfactory specification of the intended system is reached. The latter works towards an operational implementation of the system specification and therefore applies hierarchical decomposition and implementation of sub-components using a behavior description like transition diagrams.

One idea of formal methods is to formally grasp such development steps [BS01, Bro09]. By proving properties about development steps – which is only possible if we have formalized them in the first place – we can justify the usage of these steps and rules about their usage during the development.

The FOCUS theory formalizes the development process of non-probabilistic systems [Bro10, Bro97]. The foundation of this formalization is the notion of *refinement*, which explains when one specification complies with another specification. Upon this notion of *refinement* different development steps are explained like enhancing requirements, decomposition, implementation or change of level of abstraction, just to mention a few examples.

In this chapter, we will extend the notion of refinement to probabilistic systems and elaborate important refinement rules. Based on the results of the FOCUS theory, we will present the formalization of a selection of activities during the development of probabilistic systems.

7.1 Probabilistic Refinement Relations

In Focus, a nondeterministic behavior function $F' \colon I \rhd\!\!\!\gg O$ is a **refinement** of the function $F \colon I \rhd\!\!\!\gg O$ if it holds (for all $i \in \bar{I}^\infty$)

$$F'(i) \subseteq F(i) \quad .$$

In that case, we notate $F \rightsquigarrow F'$. Every behavior allowed by F' complies also with the function F. This is also emphasized by the fact that every realization of the refinement F' is also a realization of F, i.e.,

$$(F \rightsquigarrow F') \implies [\![F']\!] \subseteq [\![F]\!] \quad .$$

This definition can analogously be applied to the probabilistic case. We say, a probabilistic behavior function $H \colon I \overset{P}{\rhd\!\!\!\gg} O$ is **refined by** the function $H' \colon I \overset{P}{\rhd\!\!\!\gg} O$ if it holds (for all $i \in \bar{I}^\infty$)

$$H'(i) \subseteq H(i) \quad .$$

In this case, we notate also $H \rightsquigarrow H'$.

Based on this notion of refinement, which is also called **property refinement**, the Focus theory defines more concrete kinds of refinement. As a special case of the property refinement, we talk about **glass box refinement** if a function is refined either by a composition or a function that is defined using an operational description technique. The third kind is the **interaction refinement** which allows changes to the representation of the communication and thereby changes the level of abstraction.

The glass box refinement is a property refinement with an additional syntactic requirement about the refining function F'. This syntactic requirement directly carries over to the probabilistic case. We call a refinement $F \rightsquigarrow F'$ a **glass box refinement** if either

- F' is defined by composition, i.e., $F' = F_1 \circ^L F_2$ for $\circ \in \{\otimes, \obar\}$ and some independence constraints L,
- or F' is defined by an executable description, i.e., $F' = F_A$ for some nondeterministic probabilistic automaton A.

Similar to how the interaction refinement allows changes of the level of abstraction, we introduce an additional kind of refinement that allows the transition from non-probabilistic models to probabilistic ones. Given a non-probabilistic behavior function $F \colon I \rhd\!\!\!\gg O$ and a probabilistic behavior function H, we say that $H \colon I \overset{P}{\rhd\!\!\!\gg} O$

is an **almost surely refinement** of F if it fulfills the specification of F almost surely. Formally, we define

$$F \overset{a.s.}{\rightsquigarrow} H \iff F^P \rightsquigarrow H \quad .$$

In that case, we also say F is refined by H almost surely. This definition is based on the translation of non-probabilistic to probabilistic systems as it was introduced in Chapter 6. By substituting its definition we obtain:

$$F \overset{a.s.}{\rightsquigarrow} H \iff (\forall i\colon \mu \in H(i) \implies \mu(F(i)) = 1) \quad .$$

7.2 Properties

In this section, we present several important properties of the refinement relation of nondeterministic, probabilistic behavior functions. We start with basic properties like transitivity and monotony with respect to our composition operators. Afterwards, we prove useful refinement relations like merging dependent components into a single component.

Theorem 20 (Transitivity of refinement) *Given nondeterministic probabilistic behavior functions H_1, H_2 and H_3, from*

$$H_1 \rightsquigarrow H_2 \wedge H_2 \rightsquigarrow H_3$$

follows

$$H_1 \rightsquigarrow H_3 \quad .$$

□

Proof The definition of refinement implies

$$\forall i\colon H_2(i) \subseteq H_1(i) \wedge H_3(i) \subseteq H_2(i) \quad .$$

From the transitivity of the subset relation, we have

$$\forall i\colon H_3(i) \subseteq H_1(i) \quad .$$

By definition this means, that H_3 refines H_1 and completes the proof. ■

Theorem 21 (Monotony of Feedback with respect to refinement)

$$H \rightsquigarrow H' \implies H^{\circlearrowleft} \rightsquigarrow H'^{\circlearrowleft} \quad .$$

□

PROOF From the definition of refinement, we have

$$\forall i\colon H'(i) \subseteq H(i) \quad . \tag{7.1}$$

Let $h \in [\![H']\!]$ be a realization of H', i.e.,

$$\forall i\colon h(i) \in H'(i) \quad .$$

With (7.1) it follows

$$\forall i\colon h(i) \in H(i)$$

and h is also a realization of H. As this holds for every realization h of H', we have $[\![H]\!] \subseteq [\![H']\!]$.

The feedback of a nondeterministic behavior function F is defined as (for all inputs i)

$$F^{\circlearrowleft}(i) = \{f^{\circlearrowleft}(i) \mid f \in [\![F]\!]\} \quad ,$$

which is monotonic in the set of realizations $[\![F]\!]$. Applied to H and H' this proves the claim of the theorem. ∎

Theorem 22 (Monotony of Parallel Composition wrt. refinement)

$$A \rightsquigarrow A' \wedge B \rightsquigarrow B' \implies (A \oplus^L B) \rightsquigarrow (A' \oplus^L B') \quad .$$ □

PROOF Let i be an arbitrary input sequence. Given a measure $\mu \in (A' \oplus^L B')(i)$, it follows

$$\mu|_{O_A} \in A'(i|_{I_A}) \wedge \mu|_{O_B} \in B'(i|_{I_B}) \wedge \mu \models L \quad .$$

Because of the refinement relation to A respectively B, we obtain

$$\mu|_{O_A} \in A(i|_{I_A}) \wedge \mu|_{O_B} \in B(i|_{I_B}) \wedge \mu \models L \quad ,$$

which is exactly the definition of $\mu \in (A \oplus^L B)(i)$ and completes the proof. ∎

Theorem 23 (Monotony of General Composition wrt. refinement)

$$A \rightsquigarrow A' \wedge B \rightsquigarrow B' \implies (A \otimes^L B) \rightsquigarrow (A' \otimes^L B') \quad .$$ □

PROOF This theorem follows immediately by applying the transitivity of the refinement relation and the two previous theorems to the definition of general composition. ∎

These theorems can easily be generalized to the composition of more than two components. However, we will omit the proof here.

Thanks to the monotony property of refinement, a composite component can be refined by refining its subcomponents. Thus, we have a formal means of evolving architectures. In the following, we present a collection of useful refinement relations which are usually applied during a system development. At the end of this section, you will find a summary of these relations.

Theorem 24 (Refinement by adding independence constraints)

$$L \subseteq L' \implies \prod^{L} \mathcal{C} \rightsquigarrow \prod^{L'} \mathcal{C} \quad .$$ □

PROOF Let be i an arbitrary input sequence. Given a measure $\mu \in (\prod^{L'} \mathcal{C})(i)$, it follows

$$\mu \in \prod \mathcal{C} \ \wedge \ \mu \models L' \quad .$$

Because of $L \subseteq L'$, the measure μ fulfills also $\mu \models L$. This explains $\mu \in (\prod^{L} \mathcal{C})(i)$ and completes the proof. ■

The following refinement relations allow replacing two components by a single component. Therefore, we introduce a substitution operation on sets of independencies. Namely, for variable sets $S, T \subseteq V$, we denote with $(X \perp Y \mid Z)[{}^{S}/{}_{T}]$ the constraint obtained by replacing all occurrences of U by V, i.e.,

$$(X \perp Y \mid Z)[{}^{S}/{}_{T}] = (X' \perp Y' \mid Z')$$

with

$$X' = \begin{cases} X \setminus S \cup T & \text{if } S \subseteq X \\ X & \text{otherwise.} \end{cases}$$

and analogous definitions of Y' and Z'.

Theorem 25 (Refinement by merging independent components) *Let $\mathcal{C} = \{C_1, C_2, \ldots\}$ be a finite set of at least two components and $\mathcal{C}' = \mathcal{C} \cup \{C_{12}\} \setminus \{C_1, C_2\}$ with $C_{12} = C_1 \oplus^{\perp} C_2$. Furthermore, let L be a set of independencies over $\mathcal{C}$ and L' a set of independencies over $\mathcal{C}'$ such that it holds*

$$L \to (C_1 \perp C_2)$$

as well as (for all independency constraints x)

$$\begin{aligned}(x \in L \wedge x \neq C_1 \perp C_2) &\implies x = x[\{C_1\}/\{C_1,C_2\}] = x[\{C_2\}/\{C_1,C_2\}] \\ \wedge\ (x \in L \wedge x \neq C_1 \perp C_2) &\iff x[\{C_1,C_2\}/\{C_{12}\}] \in L' \quad . \end{aligned} \tag{7.2}$$

Then merging the two components C_1 and C_2 into the single component C_{12} is a refinement step, i.e.,

$$\prod\nolimits^{L} \mathcal{C} \rightsquigarrow \prod\nolimits^{L'} \mathcal{C}' \quad . \qquad \Box$$

Proof Let an arbitrary input i and measure $\mu \in \left(\prod^{L'} \mathcal{C}'\right)(i)$ be fixed. The definition of the restricted parallel composition says

$$\begin{aligned}\forall k \geq 3 \colon \mu|_{O_k} &\in C_k(i|_{I_k}) \\ \wedge\ \mu|_{O_{12}} &\in C_{12}(i|_{I_{12}}) \quad . \end{aligned}$$

Together with the fact that $C_{12} = C_1 \oplus^{\perp} C_2$, this implies:

$$\begin{aligned}\mu|_{O_{12}}|_{O_1} = \mu|_{O_1} &\in C_1(i|_{I_1}) \\ \wedge\ \mu|_{O_{12}}|_{O_2} = \mu|_{O_2} &\in C_2(i|_{I_2}) \quad . \end{aligned}$$

Thus, it is also $\mu \in \left(\prod \mathcal{C}\right)(i)$. What remains to be shown, is that the constraints L are fulfilled by μ. Therefore, we expand the independence constraints over components to constraints over the components' outputs. Because of $O_{12} = O_1 \cup O_2$, the equivalence (7.2) implies:

$$(x \in L \wedge x \neq O_1 \perp O_2) \iff x \in L' \quad . \tag{7.3}$$

Given some arbitrary $x \in L$, we consider two cases. The first case is $x = (O_1 \perp O_2)$. Because of $\mu|_{O_{12}} \in (C_1 \oplus^{\perp} C_2)(i|_{I_{12}})$, we have $\mu|_{O_{12}} \models (O_1 \perp O_2) = x$ and accordingly $\mu \models x$.

In the other case, i.e., $x \neq (O_1 \perp O_2)$, it is $x \in L'$ and from the equivalence (7.3) and from $\mu \in \left(\prod^{L'} \mathcal{C}'\right)(i)$ we know that $\mu \models x$.

Thus, it holds $\mu \models L$ and finally $x \in \prod^{L} \mathcal{C}(i)$. ■

Example 7.1 This example is based on the communication example introduced in Section 5.4. This time, we require that the communication media are independent and we allow only dependencies between the Sender and the Receiver. This allows refining the composed system by merging the media components into a single composite media component. Note that we have to apply an independent composition when merging the two media. The transition is visualized in Figure 7.1.

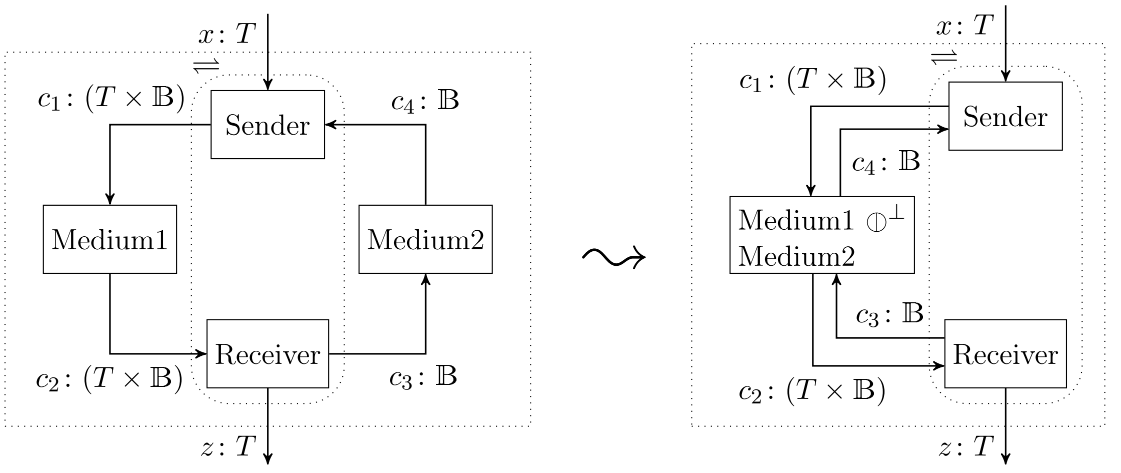


Figure 7.1: Merging the two independent media (left) into a single composite component (right).

Formally, we can justify this refinement according to the previous theorem and the independencies modeled by the diagrams in Figure 7.1 (L for the left and L' for the right diagram):

$$\begin{aligned}
L &= \{(\mathsf{Sender}, \mathsf{Receiver} \perp \mathsf{Medium1}, \mathsf{Medium2}), (\mathsf{Medium1} \perp \mathsf{Medium2})\}\\
L' &= \{(\mathsf{Sender}, \mathsf{Receiver} \perp \mathsf{Medium12})\}\\
&\quad \text{where } \mathsf{Medium12} = \mathsf{Medium1} \oplus^{\perp} \mathsf{Medium2} \quad .
\end{aligned}$$ □

Theorem 26 (Refinement by merging dependent components)
Let $\mathcal{C} = \{C_1, C_2, \ldots\}$ be a finite set of at least two components and $\mathcal{C}' = \mathcal{C} \cup \{C_{12}\} \setminus \{C_1, C_2\}$ with $C_{12} = C_1 \oplus C_2$. Furthermore, let L be a set of independencies over $\mathcal{C}$ and L' a set of independencies over $\mathcal{C}'$ such that it holds

$$\begin{aligned}
x \in L &\implies x = x[\{C_1\}/\{C_1,C_2\}] = x[\{C_2\}/\{C_1,C_2\}]\\
\wedge\, x \in L &\iff x[\{C_1,C_2\}/\{C_{12}\}] \in L' \quad .
\end{aligned} \tag{7.4}$$

Then merging the two components C_1 and C_2 into the single component C_{12} is a refinement step with regard to the general composition of $\mathcal{C}$, i.e.,

$$\prod\nolimits^{L} \mathcal{C} \rightsquigarrow \prod\nolimits^{L'} \mathcal{C}' \quad .$$ □

PROOF Let an arbitrary input i and measure $\mu \in \left(\prod^{L'} \mathcal{C}'\right)(i)$ be fixed. As in the previous proof, it follows that $\mu \in \left(\prod \mathcal{C}\right)(i)$. What remains to be shown, is that the constraints L are fulfilled by μ.

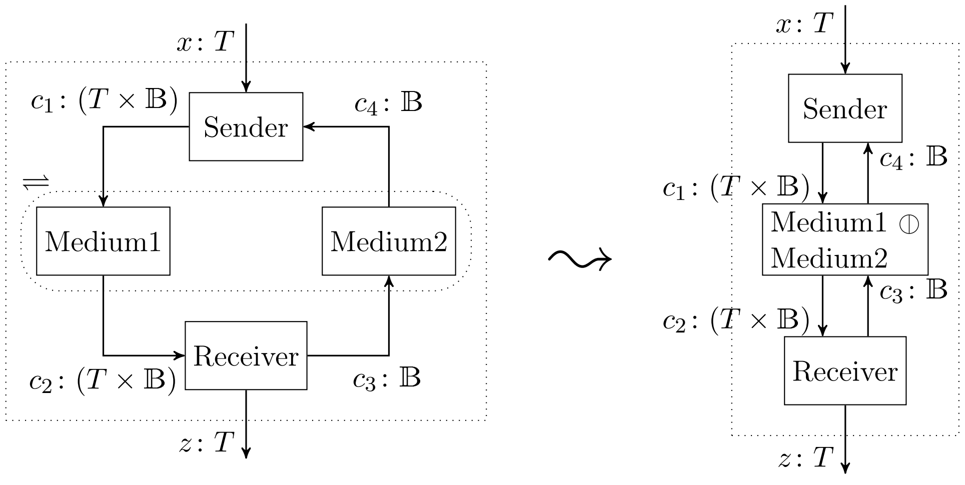


Figure 7.2: Merging the two dependent media (left) into a single composite component (right).

This becomes obvious, when we expand the independence constraints over components to constraints over the components' outputs. Because of $O_{12} = O_1 \cup O_2$, the equivalence (7.4) does then mean $L = L'$. Thus, it holds $\mu \models L$ and finally $x \in (\prod^L \mathcal{C})(i)$. ■

Example 7.2 In this example, we consider dependent communication media. We can merge these two components, according to the previous theorem, into a single composite component. The refinement is visualized in Figure 7.2.

We can justify this refinement relation by looking at the independence constraints of both models (L for the left and L' for the right diagram):

$$\begin{aligned} L &= \{(\mathsf{Sender}, \mathsf{Receiver} \perp \mathsf{Medium1}, \mathsf{Medium2}), (\mathsf{Sender} \perp \mathsf{Receiver})\} \\ L' &= \{(\mathsf{Sender}, \mathsf{Receiver} \perp \mathsf{Medium12}), (\mathsf{Sender} \perp \mathsf{Receiver})\} \\ &\quad \text{where } \mathsf{Medium12} = \mathsf{Medium1} \oplus \mathsf{Medium2} \quad . \end{aligned}$$

Replacing Dependencies by Typed Channels Given two dependent components, we saw already that these can be merged into a single component. Alternatively, we can make the dependency explicit by introducing a new typed channel and restricting the dependency to this channel, i.e., enforcing independency beyond this channel communication.

Let C_1 and C_2 be nondeterministic, probabilistic behavior functions. Consider the refinement relation shown in Figure 7.3. We will decide on components C_1' and C_2' with an additional interconnecting channel d such that the refinement relation

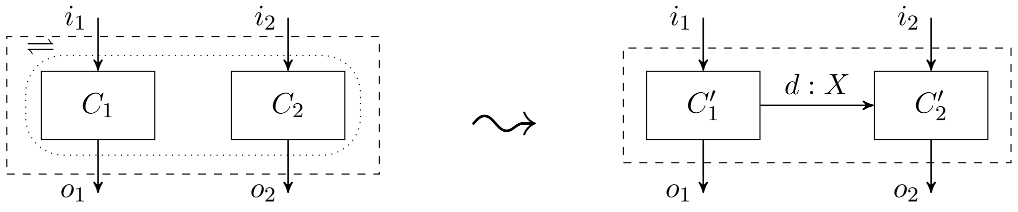


Figure 7.3: Replacing dependency (left) by an explicit channel (right).

holds, i.e.,

$$(C_1 \obar C_2) \rightsquigarrow (C'_1 \otimes^{\perp} C'_2) \dagger (O_1 \cup O_2) \quad .$$

The following theorem states a possible definition of C'_1 and C'_2 under which this refinement relation is guaranteed.

Theorem 27 *Let $C_1 \colon I_1 \overset{P}{\rhd\!\!\gg} O_1$ and $C_2 \colon I_2 \overset{P}{\rhd\!\!\gg} O_2$ be nondeterministic, probabilistic behavior functions such that $(I_1 \cup O_1) \cap (I_2 \cup O_2) = \emptyset$ and $I_1 \cap O_1 = \emptyset = I_2 \cap O_2$. Furthermore, let d be some unused channel identifier of arbitrary type and let D equal $\{d\}$. We define $C'_1 \colon I_1 \overset{P}{\rhd\!\!\gg} (O_1 \cup D)$ and $C'_2 \colon (I_2 \cup D) \overset{P}{\rhd\!\!\gg} O_2$ to be the components such that*

$$\mu \in C'_1(i_1) \iff \forall n \in \mathbb{N}_0, \sigma \in \overline{D}^n \; \exists \nu \in C_1(i_1) \; \forall o_1 \in \overline{O_1}^n \colon$$
$$\mathbf{P}[\mu \sqsupseteq o_1 \uplus \sigma] = \mathbf{P}[\mu|_D \sqsupseteq \sigma] \cdot \mathbf{P}[\nu|_{O_1} \sqsupseteq o_1]$$

and

$$C'_2(i_2 \uplus \sigma) = C_2(i_2)$$

holds. Given these definitions, the refinement relation

$$(C_1 \obar C_2) \rightsquigarrow (C'_1 \otimes^{\perp} C'_2) \dagger (O_1 \cup O_2)$$

holds, which is visualized in Figure 7.3. □

Proof Let $i_1 \in \overline{I_1}^{\infty}$, $i_2 \in \overline{I_2}^{\infty}$ and a measure $\mu \in (C'_1 \otimes^{\perp} C'_2)(i_1 \uplus i_2)$ be given. The definition of the feedback operation states (with $f \colon I_1 \cup I_2 \cup D \overset{P}{\rhd} O_1 \cup O_2 \cup D$):

$$\mu \in (C'_1 \obar^{\perp} C'_2)^{\circlearrowleft}(i_1 \uplus i_2)$$
$$\iff \exists f \colon f \in [\![C'_1 \obar^{\perp} C'_2]\!] \;\wedge\; \mu = f^{\circlearrowleft}(i_1 \uplus i_2) \quad .$$

Given such a function f it follows:

$$\forall i'_1, i'_2, d \colon f(i'_1 i'_2 d)|_{O_1 \cup D} \in C'_1(i'_1) \;\wedge\; f(i'_1 i'_2 d)|_{O_2} \in C'_2(i'_2 d)$$
$$\wedge\; f \models O_1 \cup D \perp O_2$$
$$\wedge\; \forall o_1, o_2, d \colon \mu(o_1 o_2 d) = f(i_1 i_2 d)(o_1 o_2 d) \quad .$$

Using the definition of C_1', C_2' and of independence, we obtain:

$$\begin{aligned}\forall i_1', i_2', d, d' \colon \exists \nu \in C_1(i_1')\ \forall o_1 \colon\ & f(i_1' i_2' d)(o_1 d') = f(i_1' i_2' d)(d') \cdot \nu(o_1) \\ \wedge\ & f(i_1' i_2' d)|_{O_2} \in C_2(i_2') \\ \wedge\ \forall o_1, o_2, d \colon\ & \mu(o_1 o_2 d) = f(i_1 i_2 d)(o_1 d) \cdot f(i_1 i_2 d)(o_2) \quad .\end{aligned}$$

The measures ν may depend on d and d'. Applying the first conjunct to i_1, i_2 and $d = d'$, we obtain measures $\nu_d \in C_1(i_1)$ such that it holds:

$$\forall o_1, o_2, d \colon \mu(o_1 o_2 d) = f(i_1 i_2 d)(d) \cdot \nu_d(o_1) \cdot f(i_1 i_2 d)(o_2) \quad .$$

Then for the projection $\mu|_{O_1}$ it follows:

$$\mu(o_1) = \sum_{o_2, d} \mu(o_1 o_2 d) = \sum_{d} f(i_1 i_2 d)(d) \cdot \nu_d(o_1) \cdot \sum_{o_2} f(i_1 i_2 d)(o_2) \quad .$$

The last summation in this product equals one. The factors $f(i_1 i_2 d)(d)$ sum up to one. Thus, $\mu(o_1)$ is a linear combination of the measures $\nu_d \in C_1(i_1)$. That means $\mu|_{O_1} \in C_1(i_1)$, which is convex by the definition of nondeterministic behavior.

Similarly, we obtain for $\mu|_{O_2}$:

$$\mu(o_2) = \sum_{o_1, d} \mu(o_1 o_2 d) = \sum_{d} f(i_1 i_2 d)(d) \cdot \sum_{o_1} \nu_d(o_1) \cdot f(i_1 i_2 d)(o_2) \quad .$$

This time, the second factor $\sum_{o_1} \nu_d(o_1)$ equals one and with $f(i_1 i_2 d)|_{O_2} \in C_2(i_2)$, we have $\mu|_{O_2} \in C_2(i_2)$.

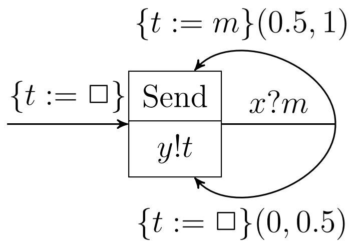
It follows $\mu|_{O_1 \cup O_2} \in (C_1 \oplus C_2)(i_1 \uplus i_2)$, which completes the proof. ■

Example 7.3 In this example, we will apply the results of the previous theorems. We continue with the lossy communication media example and show how to introduce an interconnecting channel within a refinement steps. As a starting point, consider the nondeterministic lossy medium in the following specification:

LossyMedium [**channel** x, **channel** y] nondet. prob

in $x \in T^-$
out $y \in T^-$

local $t \in T^-$



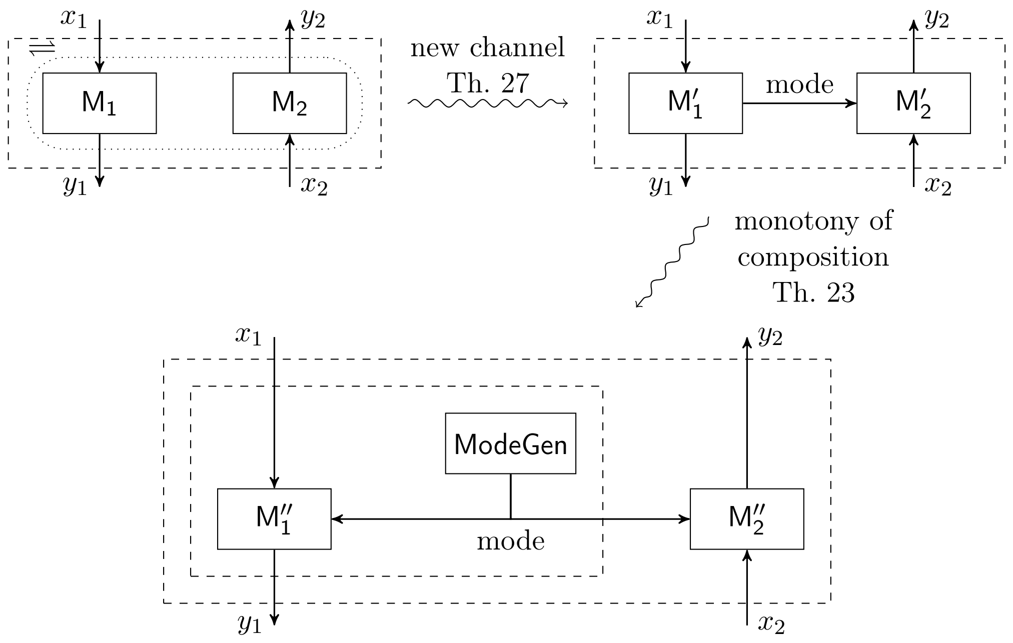


Figure 7.4: Replacing dependency (top left) by an explicit channel (right) and then decomposing the component M''_1 (bottom).

It does not enforce a concrete distribution to drop or forward a message but specifies a range of possible distributions. We compose two such media allowing unspecified dependencies between both instances. The composition is visualized on the left of Figure 7.4 with $\mathsf{M}_1 = \mathsf{LossyMedium}[x_1, y_1]$ and $\mathsf{M}_2 = \mathsf{LossyMedium}[x_2, y_2]$. Our goal is, to make a specific coupling of the two media explicit using a typed channel and more concrete specifications of both. Say, we pick three distributions out of the allowed range and call these according to their forwarding probability:

Mode	**Forwarding Probability**	**Dropping Probability**
`hi`	1	0
`med`	0.9	0.1
`lo`	0.5	0.5

We would like to have both media to behave according to these three distributions. Thus, we can imagine these to be *modes* of behavior. In order to synchronize the media, we introduce the channel mode of type $\mathtt{Mode} = \{\mathtt{hi}, \mathtt{med}, \mathtt{lo}\}$. We can do so by applying Theorem 27 and obtain the refinement shown in the upper part of Figure 7.4. The theorem states that we obtain two derived components M'_1 and

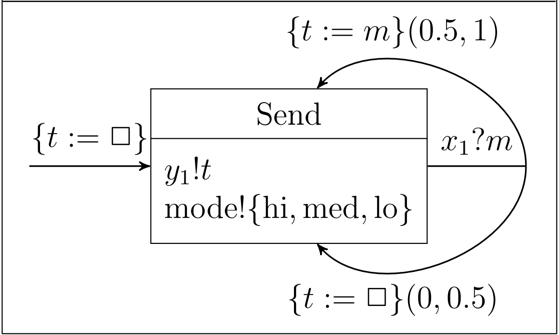
M'_2. These can be visualized as shown in the specification frames shown at the end of this example.

From these two components, we can continue with an additional refinement step, as shown with the arrow directed downward in Figure 7.4. This step makes use of the monotony with respect to the general composition and refines both M'_1 by $\mathsf{M}''[x_1, y_1] \otimes^{\perp}$ ModeGen and M'_2 by $\mathsf{M}''[x_2, y_2]$. Thereby, we specify the mode transitions using the dedicated component ModeGen. Both media behave then according to this communicated mode. The specifications of the components M'' and ModeGen are given in the following. □

M'_1 nondet. prob

in $x \in T^-$
out $y \in T^-$, mode ∈ `Mode`

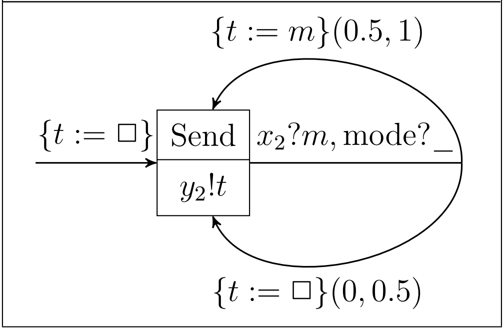
local $t \in T^-$



M'_2 nondet. prob

in $x \in T^-$, mode ∈ `Mode`
out $y \in T^-$

local $t \in T^-$



ModeGen det. prob

out mode ∈ `Mode`

init

mode	Prob
`med`	1

mode	mode'	Prob
`hi`	`hi`	1/2
	`med`	1/2
`med`	`hi`	1/3
	`med`	1/3
	`lo`	1/3
`lo`	`med`	1/2
	`lo`	1/2

M'' [**channel** x, y] det. prob

in $x \in T^-$, mode ∈ `Mode`
out $y \in T^-$

local $t \in T^-$

init

y	Prob
$\square$	1

mode	y'	Prob
`hi`	x	1
`med`	x	0.9
	$\square$	0.1
`lo`	x	1/2
	$\square$	1/2

Summary of Proven Refinement Relations We conclude this section, by summarizing all of the proven refinement relations in Table 7.1.

Th.	**Precondition**	
21	$H \rightsquigarrow H'$	$H^{\circlearrowleft} \rightsquigarrow H'^{\circlearrowleft}$
22	$A \rightsquigarrow A' \wedge B \rightsquigarrow B'$	$A \obar^{L} B \rightsquigarrow A' \obar^{L} B'$
23	$A \rightsquigarrow A' \wedge B \rightsquigarrow B'$	$A \otimes^{L} B \rightsquigarrow A' \otimes^{L} B'$
24	$L \subseteq L'$	$\prod^{L} \mathcal{C} \rightsquigarrow \prod^{L'} \mathcal{C}$
25	$C_{12} = C_1 \obar^{\perp} C_2$ C_1 and C_2 occur always as pair in L. L' equals L with all occurrences of C_1, C_2 replaced by C_{12}.	$\prod^{L \uplus \{C_1 \perp C_2\}} \mathcal{C} \uplus \{C_1, C_2\} \rightsquigarrow \prod^{L'} \mathcal{C} \uplus \{C_{12}\}$
26	$C_{12} = C_1 \obar C_2$ C_1 and C_2 occur always as pair in L. L' equals L with all occurrences of C_1, C_2 replaced by C_{12}.	$\prod^{L} \mathcal{C} \uplus \{C_1, C_2\} \rightsquigarrow \prod^{L'} \mathcal{C} \uplus \{C_{12}\}$
27	C_1, C_2 having disjoint channels. d an unused channel. $C_1'\colon I_1 \overset{P}{\vartriangleright\!\!>} (O_1 \cup \{d\}), C_2'\colon (I_2 \cup \{d\}) \overset{P}{\vartriangleright\!\!>} O_2$	$C_1 \obar C_2 \rightsquigarrow C_1' \otimes^{\perp} C_2' \dagger (O_A \cup O_B)$

Table 7.1: Summary of the proven refinement relations, given $C_1\colon I_1 \overset{P}{\vartriangleright\!\!>} O_1$ and $C_2\colon I_2 \overset{P}{\vartriangleright\!\!>} O_2$ with $O_1 \cap O_2 = \emptyset$. The binary operator $\uplus$ denotes the disjoint union.

7.3 Development of Probabilistic Systems

The refinement of probabilistic behavior functions has similar properties as the non-probabilistic refinement in Focus. But how does this relate to the activities in the development of a probabilistic system? To see this, we will at first recall the different development activities explained in the Focus theory. Afterwards, we highlight the additions in the probabilistic case.

Development of Non-Probabilistic Systems The development process can be roughly divided into the requirements engineering and the design phase accompanied by quality ensuring activities. In a strict top-down development these two phases would be performed sequentially in this order. The Focus theory formalizes the steps we may apply repeatedly to come from an initial requirement to the final system-model [Bro10].

During the requirements engineering the main activity is to successively add properties to the specification. That means starting with an empty specification S_0 which allows just any behavior, we obtain for each added property a refinement of the previous specification $S_i \rightsquigarrow S_{i+1}$. At some point, we assume the specification, say S_k, to be complete and go over to designing the implementation.

During the design phase, we have to implement one of the many system behaviors that conform to the final specification S_k. To accomplish this, we apply glass box refinements which follow the "divide and conquer" principle: Either divide the system into subsystems and implement each of these or directly provide an implementation of the system, say in the form of an automaton. If we do this repeatedly for each subsystem, we obtain a hierarchy of components, the so called logical architecture. This hierarchy is a tree of behavior functions where the composition of sibling functions is a refinement of the parent function. The functions towards the root of the hierarchy will be nondeterministic.

Instead of extending the logical architecture, we may also intersperse **design decisions** which restrict the set of allowed behaviors. That means, we refine one of the nondeterministic functions in the logical architecture F by a more restrictive function F'. Such a step does not correspond to a glass-box refinement but is, like a requirements enhancement, solely a property refinement. It excludes behaviors that fulfill the requirements for the sake of reaching a final implementation.

Extensions in the Probabilistic Case The development steps in the probabilistic case can be explained analogously. However, we have now the additional notion of probability which leads to additional choices during the development.

During the requirements engineering, we have exactly the same notion of property refinement which allows the step-wise enhancement of the specification about properties. Additionally, the presented translation of non-probabilistic specifications into a probabilistic ones allows us to combine both worlds and separate concerns. We can, for example, start by developing a non-probabilistic specification, say S_k, translate this into a probabilistic one, i.e., $P_0 = (S_k)^P$, and then continue to add probabilistic requirements using the probabilistic analog of refinement, i.e., $P_i \rightsquigarrow P_{i+1}$. Finally, we obtain a probabilistic specification P_l which is used as a starting point for the design phase. This method allows separating the non-probabilistic properties from probabilistic properties as long as possible.

During the design phase, we can again apply glass-box and property refinements. As the probabilistic composition operators are parameterized with independency-constraints, in a decomposition step, we have additionally to decide on the allowed dependencies between the components. The dependencies can then be further restricted during a property refinement. This restriction can be performed without changing the structure of the hierarchy. One application of this refinement step was shown in Example 7.3, where we replaced an implicit dependency by an explicit communication channel.

Summary of Development Steps In the following table, we summarize the steps during the development of a non-probabilistic system and annotate the extensions for the probabilistic case at the corresponding places:

Activity	**Formal Representation**	**Probabilistic Extension**
Enhance Requirements	Property Refinement $S \rightsquigarrow S'$	Transition from nonprob. to prob. spec.: $S' = S^P$
Decomposition	Glass-box Refinement $C \rightsquigarrow (C_1 \otimes C_2)$	Decide on implicit dependencies: $C_1 \otimes^L C_2$ for some set of independencies L
Implementation	Glass-box Refinement $C \rightsquigarrow F_A$ for some automaton A	Apply prob. description techniques e.g., prob. I/O automata
Clarify Dependencies	Property Refinement $(C_1 \otimes C_2) \rightsquigarrow (C_1' \otimes C_2')$ with adapted channels	Enhance set of independencies, e.g., $(C_1 \obar C_2) \rightsquigarrow (C_1' \otimes^L C_2')$

Chapter 8

Conclusion

Inspired by the modeling theory Focus for non-probabilistic interactive systems, this thesis establishes the basis of a modeling theory for probabilistic interactive systems. Together with other ongoing research about continuous systems, we hope to create a comprehensive modeling theory for interactive systems. This would be very beneficial for the model-based development of embedded systems.

In order to obtain a clear understanding of probabilistic systems, this thesis started with the mathematical foundation known from probability theory. Using this knowledge, we developed the basic notion of deterministic probabilistic systems. These systems have shown to be rather intuitive to understand and it makes sense to compare every further aspect of the modeling theory against this basic notion.

We continued to examine nondeterministic probabilistic systems in very detail. Borrowing from Focus, we represented such systems using set-valued behavior functions, i.e., each input is mapped to a set of alternative output distributions. Compared to other approaches [GcJS90, Her02, CDL$^+$09], we considered also the composition of systems without assuming statistical independence of the subsystems. Dropping this assumption has wide implications.

We have shown that in the presence of nondeterminism the resulting composition does not have all properties we would intuitively expect. The representation as behavior functions is an abstraction and leads to a non-associative composition operator.

Our notion of nondeterminism induces a refinement relation between probabilistic systems which allows the step-wise development of such systems. Thereby, every development step can be verified so that the resulting implementation conforms to the overall specification. As far as possible, we maintained the compatibility to the Focus theory. One effect of this effort is that non-probabilistic systems can be refined by probabilistic systems, preserving properties *almost surely*, i.e., with probability one. This justifies the specification of systems using non-probabilistic properties during the early development and enhancing the probabilistic details later on. Other refinement steps allow decomposing systems or restricting dependencies between components.

Every modeling approach also needs appropriate representations of these models that support the specification and implementation. Therefore, we presented a variety of description techniques for probabilistic systems. Again, we were able to extend existing notions of the non-probabilistic case, e.g., logical tables from Focus can be extended with an additional column to assign probabilities.

8.1 Evaluation

In retrospect, we want to review the applicability and scalability of the presented modeling theory for the development of interactive systems.

Component Hierarchies The Focus theory emphasizes the usage of component hierarchies for structuring systems. We leverage this concept in our modeling theory also for probabilistic systems. In the following, we consider a few common principles of component-based development that continue to be true, even in the presence of probabilistic behavior.

The interplay of several comprehensible components can realize complex behavior. This principle is well-known as separation of concerns or *divide-and-conquer*. Depending on the actual functionality to be realized, components can be operated in sequence, parallel or with mutual communication. The explicit decomposition of computations into parallel components is also an important step towards efficient *parallel computing*.

Components can be connected to several other components to reuse the results of their computation. In this way, we can reduce redundancy of computation at runtime. This can lead to a cheaper and more energy-efficient end product.

In the sense of a component library, we can instantiate common, maybe parameterized, component templates repeatedly in a system. Thus, we tackle *reuseability* and the efficiency of the system development.

Notion of Interface In Chapters 4 and 6, we extended the notion of a component interface to capture probabilistic behavior. The specification of interfaces is central to the development of large systems. Interfaces with a definite semantic support the communication between development teams similarly to contracts. Misunderstandings are avoided and not only a textual but an unambiguous documentation exists on which both teams can rely.

In the case of probabilistic systems, the presented notion of probabilistic behavior captures both the allowed logical behavior of a component, but also enables the exact definition of the probability of such behavior. Both fully determined probability distributions (i.e., every realization has to show this particular distribution) and ranges of distributions can be specified in an interface.

Today in the automotive domain, embedded systems are typically developed by suppliers and integrated by the carmaker. In this scenario, the carmaker could develop a probabilistic interface capturing both the logical functionality of the controller to be developed as well as the required degree of availability. This interface description is then handed over to the supplier to develop an implementation of this interface. In [JN12], we elaborate on this approach and develop a formalization of availability based on the thesis in hand.

Separation of Probabilistic Subsystems The presented component-based development approach supports the composition of non-probabilistic with probabilistic subsystems. Thus, non-probabilistic components described with methods of the Focus theory can be integrated and the scalability of the Focus theory is inherited by our probabilistic extension. As an example, consider the alternating bit protocol from Section 4.5. There, we reused description techniques of Focus to model the non-probabilistic compononents Sender and Receiver and combined them with the probabilistic LossyMedium.

This compositionality allows us also to separate a complex probabilistic behavior into a smaller probabilistic kernel and a non-probabilistic part. In certain cases, the non-probabilistic part will more compactly represent the functionality of the system. Some mathematical results in this direction are presented in [Buk95].

In [JN12], we apply this concept in the context of availability modeling: The non-probabilistic part represents the functionality of the system ignoring its availability. The probabilistic part is added by using generic templates for different kinds of availablity. In this way, we obtain a very modular system description. For example, we can derive from a specification M of a *reliable* medium the specification M' of a *faulty* medium by augmenting the availability details:

$$M' = M \triangle (\mathsf{FE}_{\mathsf{fo}}, \mathsf{SAv}(\mathtt{err}) > 0.9) \quad .$$

This expression declares that messages may be dropped as long as the steady-state probability of successfully forwarding a message is greater than 0.9.

Explicit Modeling of Dependencies That subsystems behave statistically independent is a very strong assumption and changes the analysis results tremendously. If a modeling theory only supports composition of systems under this assumption, it is tempting to apply this composition even if the assumption is unreasonable.

The presented modeling theory allows explicit modeling of statistical dependencies according to the principle:

> As long as systems are not explicitly declared to be independent, we assume that the systems possibly depend on each other.

Thus an engineer can post-pone such a declaration if he is uncertain or ignorant of its correctness until he can give good reasons for it.

Instead of accepting wrong assumptions during modeling, we want to maintain a correct system model during development. Therefore, we accept partial models with nondeterministic behavior. This reduces misunderstandings and increases the confidence in the model and derived analysis results.

In Example 6.2, we get a glimpse of the resulting system development where we incrementally restrict a nondeterministic design towards a deterministic and probabilistic implementation. This transition is accompanied by increasingly accurate analysis results.

8.2 Achievements

Based on the FOCUS modeling theory, we developed a notion of a probabilistic system and carried over and analyzed concepts from the non-probabilistic case, namely: strong causality, realizability, full realizability, nondeterminism, composition and refinement.

To the best of our knowledge, previous works about the modeling of probabilistic systems assumed statistical independence of subsystems. In contrast to these approaches, we also integrated the modeling of statistical dependencies between components. This allows us to get a more distinguished understanding of nondeterminism and opens new possibilities for the step-wise development of systems. As we have shown both concepts, nondeterminism and statistical dependency, are highly interrelated.

We provided translations and refinement relations of non-probabilistic to probabilistic systems. Thus, within the development of a large system, subsystems of both kinds can be integrated. We have rigorously proven derived refinement relations which justify certain development steps, e.g., merging dependent or independent components or replacing a dependency by a typed channel.

Existing description techniques of non-probabilistic systems can not only be reused but we also provided extensions to directly define probabilistic behavior. In particular, we introduced a new tabular notation as an alternative to transition diagrams and probabilistic command languages.

8.3 Outlook

The presented modeling theory opens a wide field of interesting topics. Extensions are imaginable in many directions: description techniques, system notion, analysis techniques and formalization of requirements. We will give directions of future research, which we think are most valuable.

System Notion In this thesis, we limited ourselves to strongly-causal behavior functions. In certain cases however, **weakly-causal** behavior functions are more appropriate as they have no input-output delay. Weakly-causal systems are problematic if they are composed in a cycle. Therefore, their composition could be restricted to non-cyclic topologies.

Having the notion of weakly-causal systems, we could consider **interaction refinements** of probabilistic systems. This kind of refinement defined in Focus considers changes of the representation of messages, i.e., changes of the type of channels. The translation of messages should however introduce no additional delay and requires weakly-causal systems.

For specifications, an alternative representation using sets of behavior functions may be more appropriate. It would be interesting to research how such specifications relate to nondeterministic behavior functions. Is it maybe necessary to model specifications as sets of nondeterministic functions?

Assumption/guarantee specifications can be realized by the combination of two probabilistic nondeterministic behavior functions. One function describes the expected environment, the other specifies the guaranteed behavior of the systems. Which rules to composition and decomposition apply here?

Analysis Current methods to efficiently analyze hierarchical systems are based on the independence assumption, e.g., the model checker Prism is based on this assumption. It is unclear if nondeterministic systems as presented in this thesis can be analyzed efficiently as well if arbitrary dependencies of subcomponents are allowed.

Requirements Probabilistic modeling theories allow the formalization of reliability requirements such that reliability should be considered rather a functional than a non-functional requirement. How does this integration of reliability influence the development process?

8.4 Acknowledgments

I would like to express my gratitude to supervisor Professor Manfred Broy. Without his generosity and his support, this dissertation would not have been possible. We had many interesting discussions and he indicated directions to work into when needed. I am very grateful to Professor Claudia Klüppelberg who accepted my second supervision so willingly.

At all times, my colleagues were a very welcomed distraction and provided me with entertaining coffee breaks. In particular, I would like to thank Maximilian Irlbeck and Alarico Campetelli, we shared many interesting ideas and a lot of fun in our regular discussions. I thank Maximilian Junker, Christian Leuxner and Maximilian Irlbeck for their helpful proof reading of this dissertation.

In the end, I want to give my hearty thanks to my family. My parents and my wife have always cared about my life and work. They patiently encouraged me and gave valuable advices everytime I met difficulties.

Bibliography

[ALRL04] Algirdas Avizienis, Jean-Claude Laprie, Brian Randell, and Carl Landwehr, *Basic Concepts and Taxonomy of Dependable and Secure Computing*, IEEE Transactions on Dependable and Secure Computing **1** (2004), no. 1, 11–33.

[Bir10] A. Birolini, *Reliability Engineering: Theory and Practice*, Springer, 2010.

[BK08] Christel Baier and Joost-Pieter Katoen, *Principles of Model Checking*, The MIT Press, 2008.

[Blu01] Norbert Blum, *Theoretische Informatik: Eine anwendungsorientierte Einführung*, second ed., Oldenbourg, 2001.

[Bro97] Manfred Broy, *Compositional refinement of interactive systems*, Journal of the ACM **44** (1997), 850–891.

[Bro00] ———, *A Logical Basis for Component-Based Systems Engineering*, Calculational System Design, NATO Science Series, IOS Press, 2000.

[Bro07] ———, *Interaction and Realizability*, Proceedings of the 33rd Conference on Current Trends in Theory and Practice of Computer Science (SOFSEM), Lecture Notes in Computer Science, vol. 4362, 2007, pp. 29–50.

[Bro09] ———, *Seamless Model Driven Systems Engineering Based on Formal Models*, Proceedings of the 11th International Conference on Formal Engineering Methods (ICFEM), Lecture Notes in Computer Science, vol. 5885, Springer, 2009, pp. 1–19.

[Bro10] ———, *A Logical Basis for Component-Oriented Software and Systems Engineering*, The Computer Journal **53** (2010), no. 10, 1758–1782.

[BS01] Manfred Broy and Ketil Stølen, *Specification and Development of Interactive Systems: Focus on Streams, Interfaces, and Refinement*, Springer, 2001.

[Buk95] Rais G. Bukharaev, *Theorie der stochastischen Automaten (Translation from Russian to German by Michael Schenke)*, Teubner Stuttgart, 1995.

[CDL$^+$09] Benoit Caillaud, Benoît Delahaye, Kim G. Larsen, Axel Legay, Mikkel L. Pedersen, and Andrzej Wasowski, *Compositional Design Methodology with Constraint Markov Chains*, Research Report RR-6993, INRIA, 2009.

[dAHJ01] Luca de Alfaro, Thomas A. Henzinger, and Ranjit Jhala, *Compositional Methods for Probabilistic Systems*, Proceedings of the 12th International Conference on Concurrency Theory (CONCUR), Lecture Notes in Computer Science, vol. 2154, Springer, 2001, pp. 351–365.

[DK82] A.L. Davis and R.M. Keller, *Data Flow Program Graphs*, Computer **15** (1982), no. 2, 26–41.

[GcJS90] Alessandro Giacalone, Chi chang Jou, and Scott A. Smolka, *Algebraic Reasoning for Probabilistic Concurrent Systems*, Proceedings of the IFIP TC2 Working Conference on Programming Concepts and Methods, North-Holland Publishing, 1990, pp. 443–458.

[GDG$^+$01] Robert M. Gray, Elizabeth Dubois, Jordan Gray, R. Adm, Augustine Heard Gray, and Sara Jean Dubois, *Probability, Random Processes, and Ergodic Properties*, Springer, 2001.

[Háj10] Alan Hájek, *Interpretations of Probability*, The Stanford Encyclopedia of Philosophy, Stanford University, 2010.

[Hap08] Jens Happe, *Analytical Performance Metrics*, Dependability Metrics, Lecture Notes in Computer Science, vol. 4909, Springer, 2008, pp. 207–218.

[Her02] Holger Hermanns, *Interactive Markov Chains: The Quest for Quantified Quality*, Lecture Notes in Computer Science, vol. 2428, Springer, 2002.

[HJ90] H. Hansson and B. Jonsson, *A Calculus for Communicating Systems with Time and Probabilities*, Proceedings of the 11th Real-Time Systems Symposium, IEEE Computer Society Press, December 1990, pp. 278–287.

[Hos60] John E. Hosford, *Measures of Dependability*, Operations Research **8** (1960), no. 1, 53–64.

[HSP83] Sergiu Hart, Micha Sharir, and Amir Pnueli, *Termination of Probabilistic Concurrent Program*, ACM Transactions on Programming Languages and Systems (TOPLAS) **5** (1983), 356–380.

[HWZ08] Holger Hermanns, Björn Wachter, and Lijun Zhang, *Probabilistic CEGAR*, Proceedings of the 20th International Conference on Computer Aided Verification, Springer, 2008, pp. 162–175.

[JN12] Maximilian Junker and Philipp Neubeck, *A Rigorous Approach to Availability Modeling*, Proceedings of the 4th International Workshop on Modelling in Software Engineering, 2012.

[KF09] D. Koller and N. Friedman, *Probabilistic Graphical Models: Principles and Techniques*, The MIT Press, 2009.

[KNP09] M. Kwiatkowska, G. Norman, and D. Parker, PRISM*: Probabilistic Model Checking for Performance and Reliability Analysis*, ACM SIGMETRICS Performance Evaluation Review **36** (2009), no. 4, 40–45.

[KNP12] ———, PRISM *website*, `www.prismmodelchecker.org`, May 2012.

[KSK76] J.G. Kemeny, J.L. Snell, and A.W. Knapp, *Denumerable Markov Chains*, Springer, 1976.

[Kwi07] M. Kwiatkowska, *Quantitative Verification: Models, Techniques and Tools*, Proceedings of the 6th joint meeting of the European Software Engineering Conference and the ACM SIGSOFT Symposium on the Foundations of Software Engineering (ESEC/FSE), ACM, 2007, pp. 449–458.

[LM87] E.A. Lee and D.G. Messerschmitt, *Synchronous Data Flow*, Proceedings of the IEEE **75** (1987), no. 9, 1235–1245.

[Lyn96] Nancy A. Lynch, *Distributed Algorithms*, Morgan Kaufmann Publishers, 1996.

[MM04] Annabelle McIver and Carroll Morgan, *Developing and Reasoning About Probabilistic Programs in pGCL*, Refinement Techniques in Software Engineering, First Pernambuco Summer School on Software Engineering (PSSE), Lecture Notes in Computer Science, vol. 3167, Springer, 2004, pp. 123–155.

[MM05] Annabelle McIver and Carroll Morgan, *Abstraction and Refinement in Probabilistic Systems*, ACM SIGMETRICS Performance Evaluation Review **32** (2005), 41–47.

[MMSS96] Carroll Morgan, Annabelle McIver, Karen Seidel, and J.W. Sanders, *Refinement-oriented probability for CSP*, Formal Aspects of Computing **8** (1996), no. 6, 617–647.

[RS89] Gerardo Rubino and Bruno Sericola, *On Weak Lumpability in Markov Chains*, Journal of Applied Probability **26** (1989), no. 3, 446–457.

[RS11] Steve Reeves and David Streader, *Refinement for Probabilistic Systems with Nondeterminism*, Proceedings 15th International Refinement Workshop, EPTCS, vol. 55, 2011, pp. 84–100.

[SL94] Roberto Segala and Nancy A. Lynch, *Probabilistic Simulations for Probabilistic Processes*, Proceedings of the 5th International Conference on Concurrency Theory (CONCUR), Lecture Notes in Computer Science, vol. 836, Springer, 1994, pp. 481–496.

[SSS08] M.R. Spiegel, J.J. Schiller, and R.A. Srinivasan, *Schaum's Outline of Probability and Statistics*, Schaum's Outline Series, 2008.

[Wie03] K.E. Wiegers, *Software Requirements*, second ed., Pro-Best Practices, Microsoft Press, 2003.

[YL92] Wang Yi and Kim G. Larsen, *Testing Probabilistic and Nondeterministic Processes*, Proceedings of the IFIP TC6/WG6.1 12th International Symposium on Protocol Specification, Testing and Verification, North-Holland Publishing, 1992, pp. 47–61.

Appendix A

Notation

Identifiers	**Type**	**Description**
$\mathcal{U}$		universal set of all values of channels
I, O		input and output alphabet in the context of I/O automaton and behaviors
I, O		input and output channels (i.e., sets of identifiers) in the context of components
A, B, K, M, N		probabilistic I/O automaton
F, G, H	$I^\infty \to O^P$	probabilistic behavior
φ, σ, ρ	X^ω	sequences
μ, ν	X^P	probability measures

Table A.1: Summary of common identifiers.

Identifier / Application	**Superset**	**Description**
$\mathcal{P}(X)$	$= \{Y \mid Y \subseteq X\}$	power set (i.e., set of subsets) of a set X
$\overline{C}$	$\subseteq C \to \mathcal{U}$	set of valuations for a channel set C
X^n	$= [0, n-1] \to X$	set of sequences over X of length n
X^*	$= \bigcup_n X^n$	set of finite sequences over X
X^∞	$= \mathbb{N}_0 \to X$	set of infinite sequences over X
X^ω	$= X^\infty \cup X^*$	set of all sequences over X
$\mathcal{C}(\varphi)$	$\subseteq X^\infty$	set of all infinite words with prefix φ, called cylinder set
$\mathbf{dist}(X)$	$\subset X \to [0,1]$	set of distributions over at most countable X
X^P	$\subset \mathcal{B} \to [0,1]$	set of probability measures over infinite X-sequences; $\mathcal{B} \subseteq \mathcal{P}(X^\infty)$ the set of measurable sets
$I \rhd O$	$\subseteq \bar{I}^\infty \to \overline{O}^\infty$	set of strongly causal, deterministic, non-probabilistic behavior functions with input channels I and output channels O
$I \rhd\!\!\rhd O$	$\subseteq \bar{I}^\infty \to \mathcal{P}(\overline{O}^\infty)$	set of strongly causal, nondeterministic, non-probabilistic behavior functions with input channels I and output channels O
$I \overset{P}{\rhd} O$	$\subseteq \bar{I}^\infty \to \overline{O}^P$	set of strongly causal probabilistic behavior functions with input channels I and output channels O
$I \overset{P}{\rhd\!\!\rhd} O$	$\subseteq \bar{I}^\infty \to \mathcal{P}(\overline{O}^P)$	set of strongly causal, nondeterministic, probabilistic behavior functions with input channels I and output channels O

Table A.2: Frequently used sets.

Application	Range	Description
$\langle\rangle, \langle\varphi_0, \ldots, \varphi_{n-1}\rangle, \langle\varphi_0, \varphi_1, \ldots\rangle$	X^0, X^n, X^∞	constructor of empty, finite and infinite sequences, resp.
$\left\langle\begin{matrix} x\colon x_0, x_1, \ldots \\ y\colon y_0, y_1, \ldots \end{matrix}\right\rangle$	$\overline{\{x, y\}}^\infty$	short notation for sequences over valuations
$\varphi.k$	X	k-th element of the X-sequence φ with $0 \leq k < \lvert\varphi\rvert$
$\varphi \cdot \sigma$	$X^{\lvert\varphi\rvert+\lvert\sigma\rvert}$	concatenation of sequences φ and σ
$\varphi \sqsubseteq \sigma$	$\mathbb{B}$	true if φ is a prefix of σ
$\varphi \times \sigma$	$(X \times Y)^{\lvert\varphi\rvert}$	element-wise Cartesian product of sequences $\varphi \in X^n$ and $\sigma \in Y^n$, $n \geq 0$
$\varphi{\downarrow_n}$	X^n	prefix of length n of a sequence of length $\geq n$
$v\vert_Y$	$\overline{Y}$	projection of a valuation $v \in \overline{X}$ to a subset $Y \subseteq X$
$\varphi\vert_Y$	$\overline{Y}^{\lvert\varphi\rvert}$	element-wise projection of a sequence of valuations $\varphi \in \overline{X}^\omega$ to a subset $Y \subseteq X$ (usually channel sets Y, X)
$v \uplus w$	$\overline{X \cup Y}$	union of valuations $v \in \overline{X}, w \in \overline{Y}$ with identical values for common identifiers
$(x_0 \mapsto {}^1\!/\!_4, x_1 \mapsto {}^1\!/\!_2, \ldots)$	$\mathbf{dist}(X)$	short notation for functions, here used for a distribution; $x_0, x_1, \ldots \in X$
$\mathbf{P}[X \mid Y]$		conditional probability of X given Y
$A \oplus B$		parallel composition of automata or behaviors A and B
$A^\circlearrowleft$		feedback of automaton or behavior A
$A \otimes B$		general composition of automata or behaviors A and B
$A \dagger O'$		output projection of automaton or behavior A to a subset of output channels O'
f_M	$I^\infty \to O^P$	behavior abstraction of I/O automaton M
F^P		probabilistic translation of a non-probabilistic behavior
$\llbracket A \rrbracket$		realizations of behavior A
$A[{}^x\!/\!_y]$		component obtained by renaming channel x as channel y in the specification of A

Table A.3: Frequently used functions.