Stochastic Automata for Fault Tolerant Concurrent Systems

Raúl Enrique Monti

Argentina
2018

Advisor: Pedro Ruben D'Argenio
Co-advisor: Holger Hermanns

A thesis presented for the degree of
Doctor of Philosophy

Universidad
Nacional
de Córdoba

Facultad
de Matemática,
Astronomía, Física
y Computación

Abstract

Automata modeling gives a rigorous mathematical structure to the invaluable analysis of highly dependable systems. Many kinds of automata have been defined, ranging from the simple transition systems to the complexity of hybrid automata, all the way through probabilistic automata, Petri nets and many others. The main purpose of automata is to give a mathematical representation of a real system for the purpose of its analysis. Being able to represent probabilistic election is a desired modeling feature since most real life models include some kind of probabilistic behavior. Much of this behavior happens on the continuous domain, when modeling for physical measures such as time, temperature, energy, etc. Complex systems that present such behaviour are usually impossible to analyze by Model Checking techniques given the state space explosion. Other techniques, such as simulation (also known as Statistical Model Checking), become an alternative, given that they avoid the construction of the full state space. No real simulation can be taken over a non-deterministic model as it is. Non-determinism has to be resolved.

In this thesis we introduce a new class of automata named Input/Output Stochastic Automata (IOSA), a restriction of Stochastic Automata with input and output transitions. We define a first version of IOSA and we give it semantic on Non-deterministic Labeled Markov Process (NLMP). We define a parallel composition operator and a notion of bisimulation. We show that parallel composition is a congruence with respect to bisimulation. Finally we demonstrate that closed IOSA models (fully generative models, i.e. without input actions) are fully deterministic and thus amenable to discrete event simulation.

A second version of IOSA introduces urgent actions to the model in order to enhance compositional modeling, a most desired characteristic in any modeling formalism. This extension introduces non-determinism even into the closed model. We are able to tell apart spurious non-determinism produced by confluent actions. We first show that confluent models are weakly deterministic in the sense that, regardless the resolution of the non-determinism, the stochastic behaviour is the same. In addition, we provide sufficient conditions to ensure that a network of interacting IOSAs, constructed with possibly non-confluent components, is con-

fluent, without the need to analyse the larger composed IOSA. In doing so, we address the complications of defining a particular form of weak transition on a continuous setting that is normally elusive.

Finally we use IOSA with urgency to define a formal semantics for Repairable Fault Trees (RFT), a prominent technique for analyzing industrial models. From what we know, this is the first work on semantics for Dynamic Fault Trees with complex repairs that allows for general distributions of failure and repairs. Furthermore our RFT models are shown to be deterministic in the absence of spare gates, and even in the presence of a subset of combinations of spare gates and spare basic elements, making them amenable to discrete event simulation. An example of rare event simulation on Repairable Fault Trees using the rare event simulation tool FIG (developed in the Dependable Systems group at FAMAF) closes this work.

Acknowledgements

I want to acknowledge ...

Contents

Chapter 1

Introduction

1.1 Motivation

Reactive embedded systems are present in most of the activities in our daily life. From relatively simple tasks like controlling the lights in our streets, watering the grass of our gardens, washing our clothes, or making us coffee, to much more complex activities like flying planes, managing our satellites, enabling complex medical analysis by the use of sophisticated machinery, routing our Internet connection to the rest of the world, securing our cars and buildings, controlling our power plants, etc. Embedded software is at the core of much important systems to which we trust our security, health and money. Hence, there is an important need to ensure their correct functioning.

The use of robots in industry has already meant a revolution in how things are produced. The so called "Internet of things" presents us a future where every day objects such as lamps, thermometers, hair dryers, and other home appliances are all connected to the Internet. It is not difficult to find this connected every day objects already in the market. This was not the case some time ago when most of this activities were performed by simpler electronic circuits, which were of not much analysis interest given their simplicity, or were not performed at all. Nowadays, an important part of the market for microprocessors is filled by microcontrollers that are the programmable core of embedded systems. A typical mid-range automobile has about 40 microcontrollers, while luxury cars can have up to 150. Most of the home electronics carry one or more microcontrollers in them. This microcontrollers are usually simpler than most known processors found in

our cellphones or computers, since they usually have a specific activity to perform in contrast to the general utility of a PC.

One of the main challenges of analyzing embeded software is the continued interaction of this systems with the physical world. Classical computers were aimed to interact with a keyboard and a monitor as input and output objects. This is not the case of most of embeded systems which are instead connected to sensors through which they perceive input data, and they respond to the physical world by means of different and heterogeneous actuators. Furthermore embeded systems are far more complex and analysis demanding than those electronic circuits that used to perform their activity in a more naive way. Some special characteristics of this systems will help us to introduce the needs, and later the means, to analyze their behaviour:

Embedded systems are not meant to be upgraded nor tunned after deployment. It is not the case that we can upgrade the software at our microwave oven, or that we can take our fridge and make it do something new from what it came with. It is usually difficult to stop a nuclear power plant and upgrade the sensors and microcontrollers in charge of controlling its correct functioning, or to get to change the behavior of microcontrollers at a satellite. Embedded systems need to be correct and sound from the beginning, and remain unchanged until the end of their functioning.

Embedded systems are in constant and never ending interaction with their environment. They land on what Pnueli describes as *reactive systems* [78]. During their interaction with the physical world, real-time constraints, and concurrency become an important issue to tackle. The usual representation of this kind of systems behaviour is that of a never ending loop, where the system listens to inputs from the concurrent behaving physical environment and reacts by responding accordingly and listening again. This contrasts with other kind of systems that halt immediately after they have transformed an input data into a result, known as transformational systems. This is also one of the reasons why time measuring is important and availability is required, since the system has to keep up with the speed and requests of a most heterogeneous concurrent physical world environment.

Embedded systems are highly dependable. There is no doubt that the risk that involves the bad behavior of this systems is very high. Imagine

the landing sensors of an airplane failing, or the heat sensors of a power plant failing. Imagine the loses on money if the microcontrollers for a line of new cars are defective and all sold cars have to be called back to be fixed. This systems require to be checked not only for correctnesses, but also for availability and effectiveness. It is not the case that an ATM will make you wait half an hour to give you the money you asked for. It should not be the case that a microcontroller on a satellite consumes half of the available battery power. Thus, not only qualitative aspects regarding correctness are required but also quantitative measures regarding effectiveness and availability are desired to be met.

1.2 Needs and Means for Formal Analysis

We have left clear the necessity of a rigorous and efficient analysis of embedded software. It can be deduced of course that the sooner we detect the problem the more we reduce the costs of solving it. The design phase of a software product is then of high interest for verification. High requirements are put on performance and dependability, which places the necessity to find the correct and effective methods to analyze this kind of software, taking not only into account its nature but also its concurrent for-ever communicating environment. Many formalisms have been proposed for this purpose, ranging over stochastic process algebras [66, 40, 65], variations of Petri Nets [83, 82], or appropriate extensions of automata such as timed automata [1, 63], probabilistic automata [84] and hybrid automata [62]. Automata theory has provided us with an efficient and effective way of formally verifying dependability and correctness of software design. Formal analysis has mostly focused on qualitative analysis, that is, determining whether a correctness property is satisfied or not by an abstract model of the system. However, the analysis of quantitative properties like reliability, throughput, mean time to failure, etc. is required for the system to provide an efficient and dependable service. For example, we do not only want to know if a system is error free but also if it can respond in an adequate time lapse while consuming a limited amount of energy.

Time is an important requirement when modeling the behaviour of physical systems. It is of interest to accurately model perhaps the rate of failure of a train component in order to analyze the mean time to failure of the train itself. Dependability usually demands to meet probability goals which are

in much cases quantified in a continuous space responding to the physical nature of the systems environment.

Model Checking [33, 79, 6] has been for a long time an important rigorous technique for exhaustively verifying the correct behaviour of a model. Nevertheless, the increasing complexity of models of interest exposes Model Checking limitations. In general, it is not possible to analyze stochastic systems with general continuous distributions by using Model Checking. Simulation [74, 106, 32], also known as statistical model checking, offers an alternative by giving tight probability bounds without the need to construct nor visit the whole state space, avoiding the state space explosion problem. Regardless which of this techniques we decide to use, a common pattern has to be followed: a model has to be obtained as a formal specification of the design, plus a formal specification of the desired properties of the model should be written down using some kind of logic. The model has to be analyzed against this properties using one of the mentioned methods. The expressiveness of the modeling language is critical for the analysis of the desired properties.

While formal methods give a rigorous mathematical foundation in order to justify the results of the analysis, their language and handling suffer many times of a steep curve of learning, which keeps it away from being spreadly used in industry. On the other hand, simpler design languages known as lightweight languages, such as SDL (System Description Language), the UML (Unified Modeling Language), or FT (Fault Trees), offer a far easier and intuitive way of representing and analyzing this designs, although their usual lack of formalization or underspecification is a big drawback for formal analysis.

In order to keep up with nowadays needs for systems dependability analysis, it becomes crucial to join this two techniques by giving this lightweight languages the firm mathematical foundations of model checking and simulation.

1.3 Contribution

In this thesis we contribute on the modeling and verification of general distributed stochastic systems. Historically, modeling for Simulation and Model Checking has been centered on Markovian models. Algorithms have been studied and optimized for this kind of models where, in the continuous case, time and other measures are governed by the exponential distribution. However, it is more often that we find that the physical world does not respond

to this kind of distribution. Stochastic Automata (SA) [40, 43] presents a framework for compositionally modeling general distributed stochastic systems, dropping the Markovian restriction. By extending SA, we propose a new formal modeling language for describing systems where events are governed by general continuous probability distributions. The language is called *Input/Output Stochastic Automata* (IOSA) and its semantic is build on the firm mathematical bases of Nondeterministic Labeled Markov Processes (NLMP)[46, 102]. Its purpose is to serve as modeling language for the analysis of systems using Simulation techniques. For this reason, it is important to ensure that the model under analysis is deterministic (only deterministic systems are subject to real simulation) in a first version, while sufficient conditions for weak determinism are given for its second version.

IOSA has been design as a compositional description language. This attends to several concerns such as avoiding the state explosion problems which greatly reduce the efficiency and even the possibility to reach the desired analysis. Moreover, compositionality allows re-usability of components and models, by decoupling the system behaviour into independent components. Furthermore, it helps on the whole engineering process by offering a more intuitive and clear understanding of the system while modeling it as arbitrarily simple independent components with precise communication points. This contrasts with huge monolithic models, where the understanding of the control flow becomes difficult and errors in the design turn to be very common.

Finally we make an approach on bridging the gap with industrial engineering on performance analysis and verification, by defining a rigorous semantic for Dynamic Fault Trees with Repairs (RFT) in terms of our language IOSA. We prove that models written in this RFT formalism are weak deterministic in the sense that the only non-determinism present in them is spurious and thus no matter how we decide to solve it, it will not affect the final result of the properties to analyze. This determinism permits us to simulate over the RFT IOSA models. Nowadays, systems are required to have a high degree of resilience and dependability. Assessing properties that fail with extremely small probability in complex models can be computationally very demanding. Rare Event Simulation is an effective alternative to the more naive Monte Carlo Simulation when carrying out this kind of demanding analysis. We analyze a case study using the Rare Event Simulation tool FIG [31, 28, 25], which allows to effectively model and analyze fault tolerant systems using the IOSA formalism.

1.4 Related work

Many formal modeling languages for general distributed stochastic systems have already been devised. In fact, most of this work is an extension from Stochastic Automata (SA) [40, 42, 43], which were motivated as an alternative to probabilistic automata in order to be capable of symbolically representing the infinite nature of stochastic behaviour systems. Inspired by generalized semi-Markov process and Timed Automata, SA introduce random variable called clocks which determine at what time an event will occur by sampling from a general continuous probability distribution. In [42] the model is presented as a compositional framework with an open and a closed behaviour, and different types of bisimulation are suggested. SA are non-deterministic, being this a drawback for simulation, since the non-determinism has to be resolved either by patching the model or by the use of schedulers defined by an expert in the model. In any case this is an error prone activity, and thus there is a need to think and model systems in a fully probabilistic way from its construction. IOSA is an attempt to do so. Other works in this direction are Bravetti's thesis on the specification and analysis of stochastic timed systems via process algebras [21] and its work with D'Argenio [22] where the stochastic behavior is generalized to arbitrary continuous distributions. We also take several ideas from the work [103] for the definition and analysis of the compositional asynchronous behavior of IOSA.

Contemporary to SA we find Interactive Markov Chains (IMC) [64] and its input/output version Input/Output Interactive Markov Chains (I/O-IMC) [36], from which we inspired our work on confluence and weak determinism for the urgent version of IOSAİ/O-IMC are also oriented to compositional modeling, as an efficient way of avoiding the state explosion problem and as a clean engineering practice. The main difference with our work is that I/O-IMC only allows Markovian distributions on its transitions, i.e. only exponential distributions in continuous spaces (although it suggests to use phase type distributions to model other kinds of distributions), in contrast to the native general continuous distribution support that IOSA enables.

For defining bisimulation on IOSA we take ideas from [45, 46, 10, 48]. More precisely we use one of the bisimulation on NLMP defined in [45]. The parallel composition operator on IOSA is defined based on ideas from [52, 45, 46, 76]. In fact it very closely resembles the parallel composition operator defined for Stochastic Automata in [40], but takes the generative-

reactive approach, as well as the input enableness concept, from [76].

There exist many attempts to effectively give a formal semantic to one of the many variants of Fault Trees [51, 69, 12, 86, 9, 17], and many others. The main difference between our work at chapter 5 and these other works is that they either not consider the repair model, or they restrict to Markovian models, or they do not ensure deterministic modeling. One of the most comprehensive works is [18]. It presents the formalization of a complete Fault Tree framework including a Repair model, it does it in a compositional aggregational manner in order to reduce the size of the final Markov Chain, but relying on I/O-IMC as its semantic and thus concerning only on Markovian models. Similarly, another model that comprises a repairable version of fault trees is [12].Again, it restricts to Markovian Models by using Petri Nets as its semantic, although being successful in reducing the state space of the model in comparison to a pure Markov Chain approach. A less comprehensive but similar work is [17], where Dynamic Fault Trees are formalized by means of I/O-IMC with the restrictions we have already mentioned. Some other recent works make an effective attempt to formalize Fault Trees. This is the case of [70] which presents a unification of several known Fault Tree formalisms in literature, which can be obtained by adequately selecting priorities and nondeterminism treatment. A compositional semantic is given on Generalized Stochastic Petri Nets, but it does not treat the repairable model, neither the general continuous probability: only exponential distribution is allowed. In contrast to this works, we formalized Fault Trees and defined a clear semantic for them all the way to its repairable fault trees extension, which completely changes the logic of the models and introduces several new sources of nondeterminism and underspecification. It is important to remark that in none of the aforementioned works it has been ensured deterministic models. Instead, we show that our semantics ensures weak-deterministic RFT models, in the sense that the only non-determinism present in the model is spurious and does not affect the stochastic behavior and hence the assessment of the top event probability.

Furthermore we present a compositional semantic for the FT models, and we symbolically execute simulations on the IOSA level avoiding the state space explosion. Good surveys on fault trees and their formal specification can be found at [92]. Finally we have not treated maintenance models as [90, 91] do. Maintenance models are an interesting approach to fault tolerance treatment reason why we are living this topic as a future work.

1.5 Thesis layout

Besides the introduction and final discusions, this thesis is organized as follows:

Chapter 2 presents the general background on automata and measure theory that are the foundations to most of the central theoretical results of the thesis. More precisely: Section 2.1 presents some principles of measure theory and probabilities, along with some important definitions on particular σ-algebras and probability measures. Section 2.2 Presents *Non Deterministic Labeled Markov Process*, which is used as the concrete semantics of IOSA and IOSA$_u$. We also recall the notion of bisimulation on this structure.

Chapter 3 introduces a first version of Input/Output Stochastic Automata. Sections 3.1 and 3.2 recall previous works and concepts on modeling and analysis of stochastic concurrent systems, such as *concurrency*, *compositionallity*, and modeling of general distributed time events in concurrent models. These concepts motivate the work on the new automata IOSA. IOSA is defined in Section 3.3. The semantics of IOSA in terms of NLMP in section Section 3.4. In Section 3.5 we define parallel composition and bisimulation for IOSA. We also show that bisimulation is a congruence for the parallel composition. In Section 3.6 we prove that closed IOSA are deterministic and thus amenable for discrete event simulation.

Chapter 4. In this chapter we extend IOSA with urgent actions. Section 4.1 defines IOSA$_u$. Section 4.2 defines the semantics of IOSA$_u$ in terms of NLMP. Section 4.3 defines parallel composition on IOSA$_u$. Closed IOSA$_u$ are non-deterministic in the general case. Confluence on IOSA$_u$ is defined in Section 4.4. In Section 4.5, we define the concept of weak determinism and show that every closed confluent IOSA$_u$ is weak deterministic. Finally, in Section 4.6 we find sufficient conditions to ensure that a net of possible non-confluent components is actually confluent. We also present a polynomial time algorithm to check for these conditions.

Chapter 5. In this chapter we give semantics to Repairable Fault Trees (RFT) in terms of IOSA$_u$. The chapter begins with a discussion on the state of the art in Fault Tree analysis. It also motivates the need for allowing

arbitrary continuous distributions along with complex repairs in Fault Tree. Section 5.3 presents a symbolic language to describe IOSA_u models. In Section 5.4 we give a formal definition of RFT. We use the IOSA_u symbolic language to provide a formal semantics for RFT in Section 5.5. In Section 5.6, we show that any RFT model induces a wakly deterministic IOSA_u. We extend the RFT semantics with spare systems in Section 5.7. In Section 5.8, we present a small case study which we analyze using the FIG simulator.

Chapter 2

Preliminaries

2.1 Probability and Measure theory

In this section we briefly present the main concepts on probability and measure theory, which we will use to sustain the development of our most important results in the thesis. Measure theory allows us to analyze the behavior of our automatons which evolve in a continuous space and take probabilistic decisions on which path to follow. It is a mathematical sound foundation over which we are able to define and justify our results. Most of the content for this section is taken from [3].

2.1.1 σ-algebras (or σ-fields)

Measure theory deals with the problem of quantifying the possible outcomes of an experiment. Given the set of possible outcomes S, a measure μ is a set function on the subsets of S, i.e. $\mu : S^2 \to \mathbb{R}^+$. Not every subset can be measured, and we will call events to those subsets which can. A σ-algebra gives a structure to the sets of events:

Definition 2.1. Let Σ be a collection of subsets of a set S. Then Σ is called a σ-algebra if $S \in \Sigma$ and Σ is closed under complementation and countable union (thus under countable intersection too). This is:

- $S \in \Sigma$.
- if $Q \in \Sigma$, then $Q^c \in \Sigma$.

- if $Q_1, Q_2, ... \in \Sigma$, then $\bigcup_{i=1}^{\infty} Q_i \in \Sigma$.

The intuition over this definition is that if a set Q is measurable then for any experiment outcome $\omega \in S$ we should be able to know if ω is in Q. If so, then we can tell if ω is in Q^c. Even more, the answer to "is ω in S"? is always True and thus S is an event. Finally if we can answer if ω is in Q_i for $i \geq 1$, then we can answer if ω is in $\bigcup_i Q_i$, and the same for their intersection.

Let Q be a non empty proper subset of S. Then it can be easily proved that $\{\varnothing, S, Q, Q^c\}$ is the smallest σ-algebra containing Q. For a set $\mathcal{G} \subseteq 2^S$ of subsets of S, we write $\sigma(\mathcal{G})$ to denote the minimal σ-algebra containing $\mathcal{G}$. We call each element in $\mathcal{G}$ a generator, we call $\mathcal{G}$ the generator set, and $\sigma(\mathcal{G})$ the generated σ-algebra.

A very useful σ-algebra is the Borel σ-algebra which is generated by the set of all open sets in a topology. Particularly, the Borel σ-algebra on the real line is $\mathscr{B}(\mathbb{R}) = \sigma(\{(a, b) | a, b \in \mathbb{R} \text{ and } a < b\})$. Similarly, $\mathscr{B}([0, 1])$ is the Borel σ-algebra on the interval $[0, 1]$ generated by the open sets in the interval $[0, 1]$. An equivalent construction is obtained by using the set of all right semiclosed intervals or the set of all left semiclosed intervals as generators.

We will call every $Q \in \Sigma$ a measurable set, and we will usually attach the σ-algebra to their corresponding base sets and we will call the tuple (S, Σ) a measurable space, being $(S, \{\varnothing, S\})$ the smaller one and $(S, \mathcal{P}(S))$ the biggest.

Let (L, Λ) and (S, Σ) be measurable spaces. A *measurable rectangle* is a set $Q \times B$ with $Q \in \Lambda$ and $B \in \Sigma$. The *product σ-algebra* on $L \times S$ is the smallest σ-algebra containing all measurable rectangles, and is denoted by $\Lambda \otimes \Sigma$. The *coproduct σ-algebra* $\Lambda \oplus \Sigma$ of L and S is defined in the disjoint union $L \uplus S$ and it is generated by the set $\Lambda \cup \Sigma$. If a collection of measurable spaces (S_i, Σ_i) for $i = 1...n$ coincide in their σ-algebra to be a same Σ, then we denote their product σ-algebra with Σ^n.

2.1.2 Probability measure

A measure on a σ-algebra Σ is a non-negative, extended real-valued function μ on Σ such that it is σ-additive, i.e.

$$\mu(\bigcup_{i \in \mathbb{N}} Q_i) = \sum_{i \in \mathbb{N}} \mu(Q_i)$$

for all countable family of pairwise disjoint measurable sets $\{Q_i \mid i \in \mathbb{N}\} \subseteq \Sigma$. If furthermore $\mu(S) = 1$ then μ is called a *probability measure*, in which case $\mu : \Sigma \to [0, 1]$. We will often make use of the Dirac probability measure concentrated on a point $a \in S$, defined for every $Q \in \Sigma$ as:

$$\delta_a(Q) = \begin{cases} 1 & \text{if } a \in Q \\ 0 & \text{if } a \notin Q \end{cases}$$

The following construction will be useful to define and analyze measures on product sigma algebras. Given measures μ and μ' on (L, Λ) and (S, Σ) respectively, the product measure $\mu \times \mu'$ on the product space $(L \times S, \Lambda \otimes \Sigma)$ is defined as the unique measure such that $(\mu \times \mu')(Q \times B) = \mu(Q) \cdot \mu'(B)$ for all $Q \in \Lambda$ and $B \in \Sigma$. Furthermore, any measure μ on (L, Λ) can be naturally extended into a measure $\hat{\mu}$ in the coproduct space $(L \uplus S, \Lambda \oplus \Sigma)$ by taking $\hat{\mu}(Q) = \mu(Q \backslash S)$, and similarly for measures on (S, Σ).

Let $\Delta(S)$ denote the set of all probability measures over the measurable space (S, Σ). We let μ, μ', μ_1,... range over $\Delta(S)$. It will be convenient to endow $\Delta(S)$ with a σ-algebra. For this we will use a standard construction from Giry [56], where $\Delta(\Sigma)$ is defined as the σ-algebra generated by the sets of probability measures $\Delta^{\geq p}(Q) \doteq \{\mu \mid \mu(Q) \geq p\}$, with $Q \in \Sigma$ and $p \in [0, 1]$. We let ξ and ζ range over $\Delta(\Sigma)$. It is worth to point out that, if the underlying σ-algebra is generated by a denumerable π-system, each singleton $\{\mu\}$ is measurable in the Giry σ-algebra. Furthermore it is guaranteed that $\Delta(\Sigma)$ separates points, i.e. if there are two measures $\mu \neq \mu'$ then there is a generator that distinguishes them, or equivalently there exists $Q \in \Delta(\Sigma)$ such that $\mu \in Q$ and $\mu' \notin Q$.

2.1.3 Measurable functions and Lebesgue integrals

Let (S_1, Σ_1) and (S_2, Σ_2) be two measurable spaces. A function $f : S_1 \to S_2$ is said to be *measurable* if for all $Q_2 \in \Sigma_2$, $f^{-1}(Q_2) \in \Sigma_1$, i.e., its inverse image maps measurable sets to measurable sets. It is standard to write $f : (S_1, \Sigma_1) \to (S_2, \Sigma_2)$ to denote that the function f is measurable. Notice that functions $f : S \to S'$ preserve complements and arbitrary (in particular

countable) unions,

$$f^{-1}(X^c) = (f^{-1}(X))^c$$

$$f^{-1}(\bigcup_i X_i) = \bigcup_i f^{-1}(X_i)$$

Thus to prove that a function is measurable from (S_1, Σ_1) to (S_2, Σ_2), it is sufficient to prove that its inverse maps every set from $\mathcal{G}$ into measurable set in Σ_1, where $\mathcal{G}$ is a generator set for Σ_2.

It will often come handy to build probability measures from other probability measures. Given that we are working on continuous grounds we will make use of integrals. Moreover, given its generality, we will make use of the Lebesgue integral instead of the Riemann integral.

Definition 2.2 (Lebesgue integral of simple functions)**.** Let $h : (S, \Sigma) \to (\mathbb{R}^+, \mathscr{B}(\mathbb{R}^+))$ be a simple function, say $h = \sum_{i=1}^n x_i A_i$, for A_i disjoints sets in Σ. Then we define the Lebesgue integral of h with respect to μ:

$$\int_S h \ \delta\mu = \sum_{i=1}^n x_i \mu(A_i)$$

Since every measurable function can be considered as the limit of increasing simple functions, the integral of measurable functions can be defined as follows.

Definition 2.3 (Lebesgue integral of probability measures)**.** Let h be a non-negative Borel measurable function, i.e. $h : (S, \Sigma) \to (\mathbb{R}^+, \mathscr{B}(\mathbb{R}^+))$, we define the Lebesgue integral of h with respect to μ as follows

$$\int_S h \ \delta\mu = sup \left\{ \int_S s \ \delta\mu : s \ simple, 0 \leq s \leq h \right\}$$

Proposition 2.1. Let (S, Σ, μ) be a measure space and $h : (S, \Sigma) \to (\mathbb{R}^+, \mathscr{B}(\mathbb{R}^+))$ be a measurable function. Then $\nu = \int_S h \ \delta\mu$ is a measure.

2.2 Non-deterministic Labeled Markov Process

Labeled Markov Processes [10, 47, 48] are a probabilistic class of labeled transition systems. They distinguish from other approaches by being founded on

the solid grounds of measure theory and Markov Process theory. LMP are motivated by the needs of correctly modeling physical systems (also known as hybrid systems) which evolve in a continuous state-space, by involving continuous parameters such as distance, time, temperature, pressure. Such systems usually combine this continuous space with discrete and commonly but not always finite sets of actions. The resulting models are tested against actions taken by the environment. In this sense we say that they are reactive systems, and are meant to work concurrently. By not modeling the environment, LMP work on external non-determinism, this is, the behavior of the environment is not known (i.e. non-deterministic) to the model. Bisimulation on LMP extends that from Larsen and Skou [72], defined for the simpler case of discrete space systems. Ideas from Joyal, Nielsen and Winskel [68], allow to adapt it to the continuous space.

Definition 2.4 (Labeled Markov Processes)**.** A labeled Markov process is a triple $(S, \Sigma, \{\mathcal{T}_a \mid a \in L\})$ where Σ is a σ-algebra on the set of states S , and for each label $a \in L$, the transition probability function $\mathcal{T}_a : S \to \Delta(S) \cup \{\bar{\varnothing}\}$ is a measurable function. Here, we let $0 : \Sigma \to [0, 1]$ be the null measure such that $\bar{\varnothing}(Q) = 0$ for all $Q \in \Sigma$.

For each state $s \in S$, $\mathcal{T}_a(s)(Q) \in [0, 1]$ represents the conditional probability of reaching any state in $Q \in \Sigma$ given that we are in state s and we make a transition label a. The null function notation, and this LMP definition, is taken from [29] which slightly differs from the original definition from [47]. Here, in order to model cases where certain transitions are refused to be taken from certain states, we use the null function which gives probability zero to every equivalence set in Σ, i.e. $\bar{\varnothing}(Q) = 0$ for all $Q \in \Sigma$. In [47] transition function are allowed to be a sub-probability function, instead.

Non-deterministic Labeled Markov Process arises as an extension of Labeled Markov Process introducing internal non-determinism [45] . They can also be seen as a continuous extension of Probabilistic Automata [93] to give them a measure theoretic sound basis to continuous state space. This sound base on measure theory is a key distinction from other approaches to the topic. We find two distinguishable characteristics in NLMP: transition function $\mathcal{T}_a$ maps states into measurable sets of probability measures rather than arbitrary sets. This allows to resolve non-determinism by the introduction of schedulers later on. Allowing arbitrary sets could make the system suffer from nonmeasurability in presence of certain non measurable continuous sets. A class of structured NLMPs [29] also endows labels with a σ-algebra.

A second characteristic allows, as in LMP, to have well defined modal operators of a probabilistic Hennessy-Milner logic. This is achieved by restricting $\mathcal{T}_a$ to measurable functions for each $a \in L$.

Definition 2.5. A *non-deterministic labeled Markov process* (NLMP for short) is a structure $(\mathbf{S}, \Sigma, \{\mathcal{T}_a \mid a \in \mathcal{L}\})$ where Σ is a σ-algebra on the set of states $\mathbf{S}$, and for each label $a \in \mathcal{L}$ we have $\mathcal{T}_a : \mathbf{S} \to \Delta(\Sigma)$ is measurable from Σ to the hit σ-algebra $H(\Delta(\Sigma))$.

The measurability requirement of $\mathcal{T}_a$ requires the definition of a σ-algebra over its codomain, the Giry σ-algebra $\Delta(\Sigma)$. Furthermore, in order to be able to map into measurable sets of measures over S, $\Delta(\Sigma)$, is endowed with a σ-algebra called the hit σ-algebra $H(\Delta(\Sigma))$ [46].

Definition 2.6 (Hit σ-algebra)**.** Let (S, Σ) be a measurable space, then $H(\Delta(\Sigma))$ is the minimal σ-algebra containing all sets

$$H_\xi = \{\zeta \in \Delta(\Sigma) | \zeta \cap \xi \neq \varnothing\}$$

for the measurable sets $\xi \in \Delta(\Sigma)$.

For each label $a \in L$ the corresponding nondeterministic transition function T_a must be measurable from the σ-algebra of states to the hit σ-algebra of measures, i.e. $T_a : (S, \Sigma) \to (\Delta(\Sigma), H(\Delta(\Sigma)))$. To prove so, it is sufficient to check the generators of the hit σ-algebra $H(\Delta(\Sigma))$, that is, it suffices to show that for each $\xi \in \Delta(\Sigma), T_a^{-1}(H_\xi) \in \Sigma$. Notice that $T_a^{-1}(H_\xi) = \{s \in S | T(s) \cap \xi \neq \varnothing\}$ is the set of all states s such that, through label a, the transition function "hits" the set of measures in ξ.

In this work, we use NLMP to define the underlying semantics of Input/Output Stochastic Automata.

Bisimulation has become the main way to analyze behavioral equivalence of transition systems in concurrency. It stands on the principle that two agents should only be distinguish from each other if they can be observed to behave differently by an interacting external third agent [81].

In [45] three varieties of bisimulation for NLMP are presented. The so called *traditional bisimulation* is based on Milner's original definition and is constructed by lifting the state bisimulation to the probabilities space, and checking that the set of outgoing probability measures from equivalent states matches modulo this lifting. We resemble this definition when defining

bisimulation on IOSA. In fact, this definition is an adaptation of *probabilistic bisimulation*, introduced by Larsen and Skou [72] in a discrete setting and adapted to a continuous setting like NLMP in [48, 45]. The idea behind the bisimulation equivalence is that from two equivalent states, an a-transition should lead with equal probability to any measurable aggregate of equivalence classes (properly speaking, to any measurable set that results from an arbitrary union of equivalence classes).

Given a relation $R \subseteq \mathbf{S}\times\mathbf{S}$, a set $Q \subseteq \mathbf{S}$ is *R-closed* if $R(Q) \subseteq Q$. If R is symmetric, Q is R-closed iff for all $s, t \in S$ such that $s\ R\ t$, $s \in Q \Leftrightarrow t \in Q$. Using this definition, a symmetric relation R can be lifted to an equivalence relation in $\Delta(\mathbf{S})$ as follows: $\mu\ R\ \mu'$ iff for every R-closed $Q \in \Sigma$, $\mu(Q) = \mu'(Q)$.

Definition 2.7 (Traditional bisimulation)**.** A relation $R \subseteq \mathbf{S}\times\mathbf{S}$ is a *bisimulation* on the NLMP $\mathcal{P} = (\mathbf{S}, \Sigma, \{\mathcal{T}_a \mid a \in \mathcal{L}\})$ if it is symmetric and for all $a \in \mathcal{L}$, $s\ R\ t$ implies that for all $\mu \in \mathcal{T}_a(s)$, there is $\mu' \in \mathcal{T}_a(t)$ s.t. $\mu\ R\ \mu'$. We say that $s, t \in S$ are *bisimilar*, denoted by $s \sim t$, if there is a bisimulation R such that $s\ R\ t$.

We know that $\sim$ is an equivalence relation [45].

A second notion of bisimulation is defined just as for LMP in [38], and is referred to as *event bisimulation*. The third bisimulation presented is original to the paper and is based on the also original hit σ-algebras (see [45]). It will be referred to as *state bisimulation* and it asserts that if $s\ R\ t$ then both $\mathcal{T}_a(s)$ and $\mathcal{T}_a(t)$ hit the same measurable sets of measures.

Definition 2.8 (State bisimulation)**.** A relation $R \subseteq S \times S$ is a state bisimulation if it is symmetric and for all $a \in L$, we have sRt implies

$$\forall \xi \in \Delta(\Sigma(R)) : \mathcal{T}_a(s) \cap \xi \neq \varnothing \Leftrightarrow \mathcal{T}_a(t) \cap \xi \neq \varnothing,$$

where $\Sigma(R)$ is the sub-σ-algebra of Σ containing all R-closed Σ-measurable sets, i.e. all $Q \in \Sigma$ such that $R(Q) \subseteq Q$.

It has been shown that provided R is symmetric, R is a state bisimulation if and only if $\Sigma(R)$ is an event bisimulation. Furthermore, it has been proved that if the NLMP is image denumerable, then R is a traditional bisimulation if and only if it is a state bisimulation. This is precisely the case for IOSAs, since its semantic is given by an image denumerable NLMP, thus the three bisimulation concepts coincide.

Chapter 3

Input/Output Stochastic Automata

Many formalisms have been proposed for modeling systems in order to analyze their reliability and performance. The field of stochastic process is one of the most prominent in this topic. Modeling is no easy task, since not only a good representation of the real system is intended, but also an efficient representation is desired in order to avoid the state space explosion phenomena that arises with the ever increasing size of the systems. This phenomena is specially problematic in Model Checking where an exhaustive state space exploration is the base of the technique. Though the full state space is not needed to be inspected by simulation techniques, the state explosion is still a problem in this kind of analysis, since it represents a drawback on the efficiency of algorithms and increases the time to obtain sufficiently accurate statistic results.

In addition to considering efficiency matters, one should also take a look at quality matters when modeling a system for performance and dependability analysis. We will always want a model as true as possible to the real system. Two main aspects of the modeling language are determining in this sense: it should be complex enough to capture all those interesting characteristics of the real system relevant to the intended analysis, and furthermore it should allow to deliver maintainable models with a clearly understandable and sound meaning, in order to avoid as much as possible the introduction of errors during modeling.

That being said, there is no question that modeling can greatly benefit from compositional approaches. This approaches facilitate systematic design

and the interchange of components, enable compositional analysis and help for the compact representation of state spaces, along with other ways of attacking the state explosion problem such as aggregation and minimization techniques [5, 17]. Compositional modeling allows the designer to focus on the modeling of the rather discernible operational behaviour of the components and the evident synchronization among them (compare to the difficulty of figuring out the whole behaviour in a monolithic model). The jump from a monolithic kind of modeling to a compositional one is not direct, and it requires to build formal mechanisms of interaction between components, and analyze equivalence in behaviour, by means of for example bisimulation.

There is a clear need to consider general distributions when modeling several time aspects of a concurrent system as could be activity durations or time stamps. In fact it is nearly mandatory if the system is intended to be analyzed on performance. This is though to the certainty that much of the functional correctness of systems depend heavily in this real-time quantities, and that moreover, correctly expressing this time aspects allows to analyze and estimate performance issues. Although traditionally the memoryless negative exponential distribution has offered a quite useful framework for modeling and analyzing systems on continuous domains, and has yield analytically tractable models (i.e., CTMCs), it is not able to truly model many phenomena. Phenomena such as timeouts in communication protocols, hard deadlines in real-time systems, human response times, the variability of the delay of sound and video frames (so-called jitter) in modern multi-media communication systems, censorial information in embedded systems, or natural phenomena, are typically described by non-memoryless distributions such as uniform, log-normal, or Weibull distributions.

Some works have been addressed in this direction, which introduce a compositional kind of modeling on generally distributed stochastic systems [22, 14], but they introduce non-determinism, which is undesirable if we want to simulate over a model. No real simulation can be considered in a non-deterministic model alone. Certain artificial mechanisms as the introduction of schedulers or the intervention of a human decision needs to be implemented in order to solve the non-determinism. Furthermore, care should be taken in doing so in an unbiased way, which is not a simple task. This has two main drawbacks: the simulation technique is no more a stand alone push button technique, and artificial decisions, which are not considered to be part of the real system to model, are introduced into the analysis. In general, it is not possible to analyze generally distributed stochastic processes, let alone if they

are also non-deterministic. However, deterministic stochastic processes can be simulated using discrete event simulation. Though there are approaches to simulate MDP either by recognizing spurious non-determinism [13, 36] or by sampling schedulers [39], it is not clear how these techniques scale to continuous settings.

In this chapter we introduce an extension of Stochastic Automata named Input/Output Stochastic Automata. Stochastic Automata [37, 42, 40] were introduced as a model to symbolically represent probabilistic transition systems which for continuous probability spaces are infinite by nature. This is, for instance, the case when we want to model systems with stochastic time behavior, i.e. models in which the occurrence time of events may respond to any continuous random variable. Stochastic automata (SA) provide a way to represent their behavior in a finite fashion. SA are inspired on Generalized semi-Markov Processes (GSMP) and timed automata, and are equipped with compositionality and a probabilistic version of bisimulation, inherited from its semantic built on probabilistic transition systems.

In a first attempt to produce a model that accomplishes both being able to represent general distributed time events, and being suitable for simulation, i.e. being fully deterministic, we introduced in this chapter the Input/Output Stochastic Automata model. As the name suggests, we bring here an input/output variant of Stochastic Automata that, once the model is closed –i.e., all synchronizations are resolved–, the resulting automaton does not contain non-deterministic choices. This turns it amenable to simulation in the general case and to much more efficient analysis if restricted to Markov models. We start the chapter by presenting a theoretical introduction to IOSA, for which we provide a concrete semantic in terms of non-deterministic labeled Markov Process (NLMP). We then prove that bisimulation is a congruence for parallel composition both in NLMP and IOSA, show that parallel composition commutes in the symbolic and concrete level, and provide a proof that a closed IOSA is indeed deterministic.

3.1 Clocks

Stochastic automata [40, 42] use clock variables to control and observe the passage of time. Since in the context of IOSA the time at which events occur is random, clocks are in fact random variables. Whenever a clock is set, it takes a random value whose probability depends on the distribution function

associated to the clock. As time evolves, clocks count down synchronously, i.e., all do so at the same rate. When a clock reaches the value zero, "the clock expires" and this may enable some events. In fact, in our context the expiration of a clock will enable a transition to be taken. In this sense, the evolution of the system will be determined by the continuously setting and expiring of clocks. At each step we will check for the lowest valued clock and the transitions it may enable. A key factor in our work will be to ensure that the expiration of a clock does not enable more than one transition, which otherwise would induce a non-deterministic situation. If x is a random clock variable in $\mathcal{C}$, then μ_x will denote its probability distribution. The notation

$$s \xrightarrow{C,a,C'} s'$$

will represent a transition from a state s to a state s' producing the action a that will be taken as soon as all clocks in the set C have expired, and will instantly set all clocks in C', each one with a value sampled from their corresponding probability distributions. Since clocks are random variables, they will probably be assigned with a different value each, which will represent a key factor when analyzing determinacy in IOSA models. We will usually want to impose an ordering on the clocks of the model, for sake of simplicity. For a set C of clocks we will denote $\vec{C}$ to this ordering, and given the set of all valuations $V : C- > \mathbb{R}^n$, $\vec{v}$ will denote a valuation for each clock in $\vec{C}$.

3.2 Open vs Closed model

Starting from the notion of stochastic automata, we restrict this framework to obtain IOSA. Based on Probabilistic Input/Output Automata [103] we split actions into inputs and outputs and let them behave in a reactive and generative manner respectively. This is, input actions stay passive and their occurrence depend only in their interaction with output actions. In contrast, the occurrence of output actions depends on the expiration of clocks and are taken as soon as they become enabled (see [96] for the concepts of reactive and generative transitions). The synchronization of an input and output action results in an output action which is again available to synchronization with other input actions. Thus, an output is broadcast to all components able to listen to it through input actions. This is not the case in SA, where synchronization between components is achieved by agreement and no distinction

between actions types is done. Agreement allows to block components by not synchronizing with them. In contrast, our models are input enabled, and thus they do not model the blocking of transitions by means of synchronization. In our opinion this achieves higher decoupling of components, and a much more clear treatment of determinism and compositionality.

We could also think that inputs are externally controlled actions and outputs are locally controlled actions. Precisely because of this, the occurrence time of output actions is controlled by a random variable, while inputs are passive and hence their occurrence time can only depend on their interaction with outputs. In this sense we will call open models to those reactive models where input actions are still present, and somehow information is missing in order to determine the complete behaviour of that model. On the other hand we will call closed models to those generative models where no input action is present, and thus its behaviour is completely determine by the model itself. A set of restrictions, which we will explain later, ensures that, almost surely, no two outputs actions are enabled at the same time. This is intended to ensure determinism on closed models in order to enable discrete event simulation on IOSA models. An open model will become a closed model once all its input actions have been synchronized through parallel composition with other models, and turned into output (generative) actions.

When analyzing if a model is deterministic, we will focus on closed models, since open models are intrinsically nondeterministic given that their behaviour depends on the decisions made by the not yet synchronized and potentially nondeterministic environment. Nevertheless, since our analysis of determinism is done over the components of the model, we will find many open models in the way.

Example 3.2.1 (Open vs Closed model for an egg selling business)**.** In Figure 3.1 we find a model of an egg selling business. In the open model (Figure 3.1b), we find that we know the rate of production of the eggs along we the rate for packing the eggs. Input transitions are still present in the model since we do not have information about the shipping rates. This will disallow us to make a complete analysis of the real system, since it will depend on the time rates at which the shipping is done. These timings are not known a priori, and, on what concerns to us, introduce a non-deterministic choices on the occurrence time. The closed model (Figure 3.1d) completes the model by synchronizing the open model with the missing shipping model (Figure 3.1c). In this way input transitions are turned into output transitions

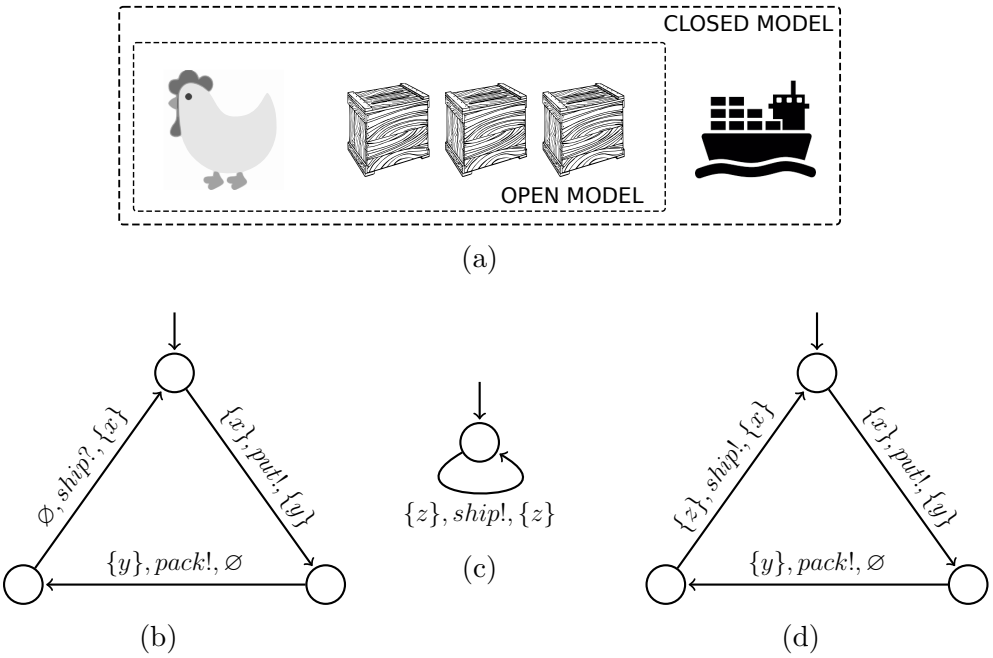


Figure 3.1: Open vs closed model. (a) Graphical representation of an egg selling business. (b) IOSA of the open model in (a). (c) IOSA of a shipping model. (d) IOSA of the closed model after composing (b) and (c).

which rates are given by the shipping model. The open model gives us a hint on one of the benefits of compositionality, which is reusability. In fact many models can be obtained by reusing the open model 3.1b and composing it with different shipping models as 3.1c. A comparative analysis can be carried out between this different settings, without the need of remodeling the whole system each time, which would be the case on a monolithic style of modeling.

3.3 Input/Output Stochastic Automata

We present a first approach to Input/Output Stochastic Automata. The formal definition of IOSA consists of a structure and a set of restrictions which will eventually ensure that closed models are deterministic.

Definition 3.1. An input/output stochastic automaton (IOSA for short) is

a structure $(\mathcal{S}, \mathcal{A}, \mathcal{C}, \rightarrow, \mathcal{C}_0, s_0)$, where $\mathcal{S}$ is a (denumerable) set of states, $\mathcal{A}$ is a (denumerable) set of labels partitioned into disjoint sets of input labels $\mathcal{A}^{\mathsf{i}}$, and output labels $\mathcal{A}^{\circ}$, $\mathcal{C}$ is a (finite) set of clocks such that each $x \in \mathcal{C}$ has associated a continuous probability measure μ_x on $\mathbb{R}$ (hence $\mu_x(d) = 0$ for any $d \in \mathbb{R}$) also satisfying that $\mu_x(\mathbb{R}_{>0}) = 1$, $\rightarrow \subseteq \mathcal{S} \times \mathcal{C} \times \mathcal{A} \times \mathcal{C} \times S$ is a transition function, $\mathcal{C}_0$ is the set of clocks that are initialized in the initial state, and $s_0 \in \mathcal{S}$ is the initial state. In addition an IOSA should satisfy the following constraints:

(a) If $s \xrightarrow{C,a,C'} s'$ and $a \in \mathcal{A}^{\mathsf{i}}$, then $C = \varnothing$.

(b) If $s \xrightarrow{C,a,C'} s'$ and $a \in \mathcal{A}^{\circ}$, then C is a singleton set.

(c) If $s \xrightarrow{\{x\},a_1,C_1} s_1$ and $s \xrightarrow{\{x\},a_2,C_2} s_2$ then $a_1 = a_2$, $C_1 = C_2$ and $s_1 = s_2$.

(d) If $s \xrightarrow{\{x\},a,C} s'$ then, for every transition $t \xrightarrow{C_1,b,C_2} s$, either $x \in C_2$, or $x \notin C_1$ and there exists a transition $t \xrightarrow{\{x\},c,C_3} t'$.

(e) If $s_0 \xrightarrow{\{x\},a,C} s$ then $x \in C_0$.

(f) For every $a \in \mathcal{A}^{\mathsf{i}}$ and state s, there exists a transition $s \xrightarrow{\varnothing,a,C} s'$.

(g) For every $a \in \mathcal{A}^{\mathsf{i}}$, if $s \xrightarrow{\varnothing,a,C_1} s_1$ and $s \xrightarrow{\varnothing,a,C_2} s_2$, then $C_1 = C_2$ and $s_1 = s_2$.

The occurrence of an action is controlled by the expiration of clocks. Thus, whenever $s \xrightarrow{\{x\},a,C} s'$ and the system is in state s, output action a will occur once the value of clock x reaches 0. At this point, the system moves to state s' setting the values of every clock $y \in C$ to a value sampled according to the distribution μ_y. For input transitions $s \xrightarrow{\varnothing,a,C} s'$, the behaviour is similar, only that its occurrence can potentially occur at any time which will become definite once the action interacts with an output.

Since we intend IOSA to be deterministic, a few restrictions have been made to the model, such that no two transitions are enabled at the same time. Restriction (a) states that every input is reactive and hence their occurrence is controlled by the environment. Hence no clock controls its occurrence. Restriction (b) states that each output is generative (or locally controlled)

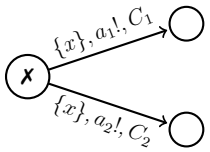


Figure 3.2: Restriction (c)

so it has associated a clock which determines its occurrence time. We also limit the set to exactly one clock, to have a clean definition.

Restriction (c) forbids that a single clock enables two different transitions, otherwise two output actions would become enable simultaneously and the choice on which one to take would become non-deterministic (See Figure 3.2 where ✗ denotes a forbidden state). Besides, notice that if clocks are used

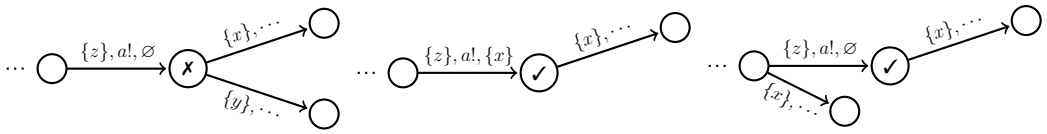


Figure 3.3: Restriction (d)

when they have already expired they would immediately enable the respective output transition. This may lead to simultaneously enabled outputs if the system arrives to a states with two expired clocks enabling two different transitions. Restrictions (d) and (e) ensure that a clock would never be used when it has already expired. Particularly (d) states that an enabling clock x at state s should either be set on arrival ($x \in C_2$) or it has not been used immediately before ($x \notin C_1$) but should be also enabling on the immediately preceding state (See Figures 3.4 and 3.3 where ✗ denotes a forbidden state and ✓ allowed states). Since clocks are set by sampling from a continuous random variable, the probability that the values of two different clocks are equal is 0. This last fact, together with restrictions (c), (d) and (e), guarantees that almost never two different output transitions are enabled at the same time.

Restrictions (f) and (g) are usual restrictions on I/O-like automata: (f) ensures that outputs are not blocked in a composition. This has a twofold effect by ensuring that the original behaviour of the automatons is preserved, and on the other hand by avoiding non-deterministic situations as the one depicted in Figure 3.5. There, the gray state at the leftmost component lacks

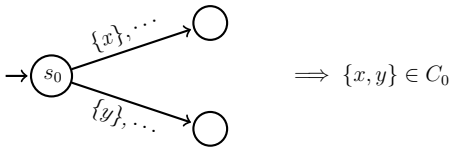


Figure 3.4: Restriction (e)

of input enableness. This introduces non-determinism after synchronization during parallel composition (see Definition 3.4 for the rules of composition), finally violating restriction (d) at the state marked with an ✗. Restriction

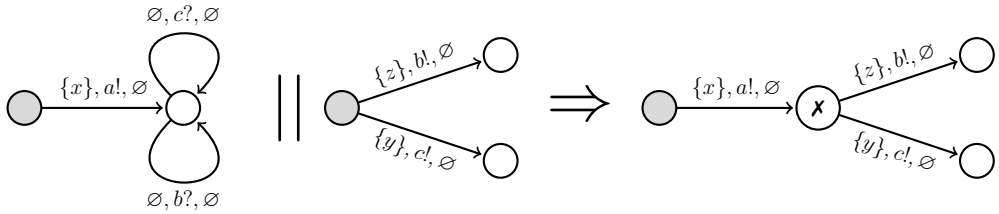


Figure 3.5: Restriction (f)

(g), also helps to ensure that determinism is preserved after composition, since otherwise situations as the one depicted in Figure 3.6 could arise. This is due to composition, where the synchronization between input and output actions generates output actions as we will see in Definition 3.4.

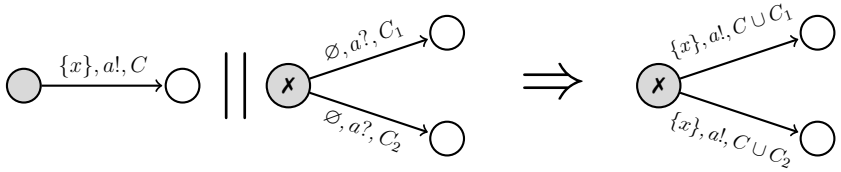


Figure 3.6: Restriction (g)

3.4 Semantics

The semantics of IOSA is defined in terms of NLMP(see section 2.2), a generalization of probabilistic transition systems with continuous domain. The

formal semantics of an IOSA is defined by a NLMP with two classes of transitions: one that encodes the discrete steps and contains all the probabilistic information introduced by the sampling of clocks, and other describing the time steps, that only records the passage of time synchronously decreasing the value of all clocks. In order to simplify the definition, we assume that the set of clocks has a particular order and their current values follow the same order in a vector. This is, $\vec{C}$ will denote this order on the set of clocks C of cardinality n, and a valuation $\vec{v}$ from the set of valuations $V : \mathcal{C} \to \mathbb{R}^n$, will denote the values for each clock at any moment.

Definition 3.2. Given an IOSA $\mathcal{I} = (\mathcal{S}, \mathcal{A}, \mathcal{C}, \to, \mathcal{C}_0, s_0)$ with $\mathcal{C} = \{x_1, \ldots, x_N\}$, its semantics is defined by the NLMP $\mathcal{P}(\mathcal{I}) = (\mathbf{S}, \mathscr{B}(\mathbf{S}), \{\mathcal{T}_a \mid a \in \mathcal{L}\})$ where

- $\mathbf{S} = (\mathcal{S} \cup \{\mathsf{init}\}) \times \mathbb{R}^N$, $\mathcal{L} = \mathcal{A} \cup \mathbb{R}_{>0} \cup \{\mathsf{init}\}$, with $\mathsf{init} \notin \mathcal{S} \cup \mathcal{A} \cup \mathbb{R}_{>0}$
- $\mathcal{T}_{\mathsf{init}}(\mathsf{init}, \vec{v}) = \{\delta_{s_0} \times \prod_{i=1}^{N} \mu_{x_i}\}$
- $\mathcal{T}_a(s, \vec{v}) = \{\mu_{\vec{v},C',s'} \mid s \xrightarrow{C,a,C'} s', \bigwedge_{x_i \in C} \vec{v}(i) \leq 0\}$,

  for all $a \in \mathcal{A}$, where $\mu_{\vec{v},C',s'} = \delta_{s'} \times \prod_{i=1}^{N} \overline{\mu}_{x_i}$ with

$$\overline{\mu}_{x_i} = \begin{cases} \mu_{x_i} & \text{if} x_i \in C' \\ \delta_{\vec{v}(i)} & \text{otherwise} \end{cases}$$

- $\mathcal{T}_d(s, \vec{v}) =$
  $\{\delta^{-d}_{(s,\vec{v})} \mid 0 < d \leq \min\{\vec{v}(i) \mid \exists a \in \mathcal{A}^{\circ}, C' \subseteq \mathcal{C}, s' \in S : s \xrightarrow{\{x_i\},a,C'} s'\}\}$

  for all $d \in \mathbb{R}_{\geq 0}$, where $\delta^{-d}_{(s,\vec{v})} = \delta_s \times \prod_{i=1}^{N} \delta_{\vec{v}(i)-d}$.

The state space is the product space of the states of the IOSA with all possible clock valuations. A distinguished initial state init is added to encode the random initialization of all clocks (it would be sufficient to initialize clocks in $\mathcal{C}_0$ but we decided for this simplification). Such encoding is done by transition $\mathcal{T}_{\mathsf{init}}$. The state space is structured in the usual Borel σ-algebra. The discrete step is encoded by $\mathcal{T}_a$, with $a \in \mathcal{A}$. Notice that, at state $(s, \vec{v})$, the transition $s \xrightarrow{C,a,C'} s'$ will only take place if $\bigwedge_{x_i \in C} \vec{v}(i) \leq 0$, that is, if the current values of all clocks in C are not positive. For the particular case of the input actions this will always be true. The next actual state

would be determined randomly as follows: the symbolic state will be s' (this corresponds to $\delta_{s'}$ in $\mu_{\vec{v},C',s'} = \delta_{s'} \times \prod_{i=1}^{N} \overline{\mu}_{x_i}$), any clock not in C' preserves the current value (hence $\overline{\mu}_{x_i} = \delta_{\vec{v}(i)}$ if $x_i \notin C'$), and any clock in C' is set randomly according to its respective associated distribution (hence $\overline{\mu}_{x_i} = \mu_{x_i}$ if $x_i \in C'$). The time step is encoded by $\mathcal{T}_d(s, \vec{v})$ with $d \in \mathbb{R}_{\geq 0}$. It can only take place at d units of time if there is no output transition enabled at the current state within the next d time units (this is verified by condition $0 < d \leq \min\{\vec{v}(i) \mid \exists a \in \mathcal{A}^{\circ}, C' \subseteq \mathcal{C}, s' \in S : s \xrightarrow{\{x_i\},a,C'} s'\}$). That is, the maximum possible time jump equals the minimum time required to elapse until an output transition becomes enabled in the current state. In this case, the system remains in the same symbolic state (this corresponds to δ_s in $\delta^{-d}_{(s,\vec{v})} = \delta_s \times \prod_{i=1}^{N} \delta_{\vec{v}(i)-d}$), and all clock values are decreased by d units of times (represented by $\delta_{\vec{v}(i)-d}$ in the same formula). Figure 3.7 shows a drawing representing the two types of transitions defined at IOSA semantics: the transition corresponding to a discrete step on the left side, and the transition corresponding to the time jump at the right side.

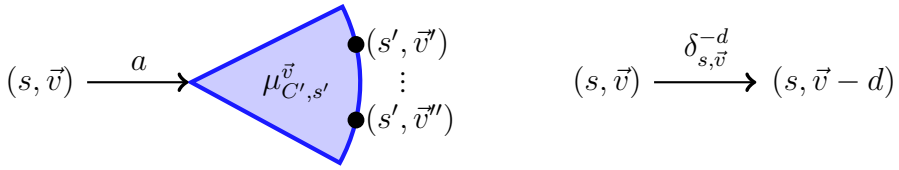


Figure 3.7: Transitions in the NLMP semantic of IOSA.

We still need to show that $\mathcal{P}(\mathcal{I})$ is indeed a NLMP. For this we have to prove that $\mathcal{T}_a$ maps into measurable sets in $\Delta(\mathscr{B}(\mathbf{S}))$ (Lemma 3.1), and that $\mathcal{T}_a$ is a measurable function for every $a \in \mathcal{L}$ (Lemma 3.2).

Lemma 3.1. $\mathcal{T}_a(s, \vec{v}) \in \Delta(\mathscr{B}(\mathbf{S}))$ for all $a \in \mathcal{L}$ and $(s, \vec{v}) \in \mathbf{S}$.

Proof. The proof makes use of Lemma 3.1 in [45], from which we know that for all $\mu \in \Delta(\mathbf{S})$, $\{\mu\} \in \Delta(\mathscr{B}(\mathbf{S}))$ (since $\mathscr{B}(\mathbf{S})$ is generated by a discrete π-system).

Notice that for any $\vec{v} \in \mathbb{R}^N$, $\mathcal{T}_{\mathsf{init}}(\mathsf{init}, \vec{v})$ is a singleton set and hence measurable. Similarly, notice that for every $d \in \mathbb{R}_{>0}$, $s \in \mathcal{S}$, and $\vec{v} \in \mathbb{R}^N$,

$\mathcal{T}_d(s,\vec{v})$ is either a singleton set or the empty set, and hence measurable. Finally, since there is only a denumerable number of transitions in an IOSA, for every $a \in \mathcal{A}$, $s \in \mathcal{S}$, and $\vec{v} \in \mathbb{R}^N$, $\mathcal{T}_a(s,\vec{v})$ is a denumerable union of singleton sets, and hence also measurable. □

The following lemma uses the *hit σ-algebra* introduced in the preliminaries (Section 2.6).

Lemma 3.2. For all $a \in \mathcal{L}$, $\mathcal{T}_a$ is measurable from $\mathscr{B}(\mathbf{S})$ to $H(\Delta(\mathscr{B}(\mathbf{S})))$.

Proof. We need to show that for every $a \in \mathcal{L}$ and every $\xi \in \Delta(\mathscr{B}(\mathbf{S}))$, $\mathcal{T}_a^{-1}(H(\xi)) = \{(s,\vec{v}) \mid \mathcal{T}_a(s,\vec{v}) \cap \xi \neq \varnothing\}$ is measurable.

We divide the proof in three cases depending on the nature of the label on the transition function. First, notice that $\mathcal{T}_{\mathsf{init}}^{-1}(H(\xi)) = \{\mathsf{init}\} \times \mathbb{R}^N$ if $\delta_{s_0} \times \prod_{i=1}^N \mu_{x_i} \in \xi$ and $\mathcal{T}_{\mathsf{init}}^{-1}(H(\xi)) = \varnothing$ otherwise, and both sets are measurable.

We analyze now the case of $a \in \mathcal{A}$, for which we can calculate

$$\begin{aligned}\mathcal{T}_a^{-1}(H(\xi)) &= \{(s,\vec{v}) \mid \{\mu_{\vec{v},C',s'} \mid s \xrightarrow{C,a,C'} s', \textstyle\bigwedge_{x_i \in C} \vec{v}(i) \leq 0\} \cap \xi \neq \varnothing\} \\ &= \textstyle\bigcup_{s \xrightarrow{C,a,C'} s'} \{(s,\vec{v}) \mid \bigwedge_{x_i \in C} \vec{v}(i) \leq 0\} \cap \{(s,\vec{v}) \mid \mu_{\vec{v},C',s'} \in \xi\}\end{aligned}$$

Since the union is denumerable, it is sufficient to prove that the two intersecting sets are measurable. First, notice that $\{(s,\vec{v}) \mid \bigwedge_{x_i \in C} \vec{v}(i) \leq 0\} = \{s\} \times \prod_{i=1}^N V_i$ where $V_i = (-\infty, 0]$ if $x_i \in C$ and $V_i = \mathbb{R}$ otherwise. Hence, it is measurable.

For the second case, define $f_{C',s'} : \mathbb{R}^N \to \Delta(\mathbf{S})$ by $f_{C',s'}(\vec{v}) = \mu_{\vec{v},C',s'}$. Then $\{(s,\vec{v}) \mid \mu_{\vec{v},C',s'} \in \xi\} = \{(s,\vec{v}) \mid f_{C',s'}(\vec{v}) \in \xi\} = \{s\} \times f_{C',s'}^{-1}(\xi)$. So, it only remains to prove that $f_{C',s'}$ is a measurable function. Using [99, Lemma 3.6], we only have to prove that $f_{C',s'}^{-1}(\Delta^{\geq q}(A \times \prod_{i=1}^N V_i))$ with $A \subseteq \mathbf{S}$ and $V_i \in \mathscr{B}(\mathbb{R})$, $1 \leq i \leq N$, is measurable, for which we can calculate

$$\begin{aligned}f_{C',s'}^{-1}(\Delta^{\geq q}(A \times \textstyle\prod_{i=1}^N V_i)) &= \{\vec{v} \mid \mu_{\vec{v},C',s'}(A \times \textstyle\prod_{i=1}^N V_i) \geq q\} \\ &= \{\vec{v} \mid s' \in A, (\textstyle\prod_{x_i \in C'} \mu_{x_i})(\prod_{x_i \in C'} V_i) \geq q, \forall x_i \notin C' : \vec{v}(i) \in V_i\}\end{aligned}$$

Then, if $s' \in A$ and $(\prod_{x_i \in C'})(\prod_{x_i \in C'} V_i) \geq q$, $f_{C',s'}^{-1}(\Delta^{\geq q}(A \times \prod_{i=1}^N V_i)) = \prod_{i=1}^N \overline{V_i}$ with $\overline{V_i} = \mathbb{R}$ if $x_i \in C'$, $\overline{V_i} = V_i$ if $x_i \notin C'$, or $f_{C',s'}^{-1}(\Delta^{\geq q}(A \times \prod_{i=1}^N V_i)) = \varnothing$ otherwise, and in both cases the sets are measurable.

For the case of $d \in \mathbb{R}$, notice that

$$\mathcal{T}_d^{-1}(H(\xi)) = \{(s,\vec{v}) \mid \delta^{-d}_{(s,\vec{v})} \in \xi\} \cap \{(s,\vec{v}) \mid 0 < d \leq \min\{\vec{v}(i) \mid \exists a \in \mathcal{A}^\circ, C' \subseteq \mathcal{C}, s' \in S : s \xrightarrow{\{x_i\},a,C'} s'\}$$

The second set is equal to $\mathbf{S} \times \prod_{i=1}^{N} V_i$ where $V_i = [d,\infty)$ if $s \xrightarrow{\{x_i\},a,C'} s'$, and $V_i = \mathbb{R}$ otherwise. Hence it is measurable. For the first set, define $f_d : \mathbf{S} \to \Delta(\mathbf{S})$ by $f_d(s,\vec{v}) = \delta^{-d}_{(s,\vec{v})}$. Then $\{(s,\vec{v}) \mid \delta^{-d}_{(s,\vec{v})} \in \xi\} = f_d^{-1}(\xi)$ and hence it suffices to show that f_d is measurable. So, we have to prove that $f_d^{-1}(\Delta^{\geq q}(Q))$ is measurable for any $Q \in \mathscr{B}(\mathbf{S})$. But $f_d^{-1}(\Delta^{\geq q}(Q)) = \{(s,\vec{v}) \mid \delta^{-d}_{(s,\vec{v})}(Q) \geq q\}$, and hence $f_d^{-1}(\Delta^{\geq q}(Q)) = \{(s,\vec{v}) \mid (s,\vec{v}-d) \in Q\}$ if $q > 0$ or $f_d^{-1}(\Delta^{\geq q}(Q)) = \mathcal{S}$ if $q = 0$, and in both cases the sets are measurable. □

3.5 Composition and bisimulation as a congruence

Parallel composition is a fundamental tool for the modular construction of large models. Parallel composition in IOSA models concurrency through the interleaving of independent actions, but synchronizes equally named actions just as it happens in CSP [67] and LOTOS [15].

Bisimulation is a notion of equivalence between agents. It is based on the idea that we only want to distinguish between agents if their observable behavior can be distinguished by a third agent [81].

In the case of IOSA, composition preserves bisimulation relation. This is, bisimulation is a congruence with respect to the composition operator. This allows us to replace components in our models by other components which behave equally, hence without altering the behavior of the composed model.

In this section we define parallel composition of IOSA and show that IOSA is closed for parallel composition. We also show that bisimulation is a congruence for the parallel composition and we achieve it through defining parallel composition on NLMP. Since we intend outputs to be autonomous (or locally controlled), we do not allow synchronization between outputs. Besides, we need to avoid name clashes on the clock, so that the intended behaviour of each component is preserved and moreover, to ensure that the resulting composed automata is indeed an IOSA. Thus we require to compose only *compatible* IOSAs.

Definition 3.3. Two IOSAs $\mathcal{I}_1$ and $\mathcal{I}_2$ are said to be compatible if they do not share output actions nor clocks, i.e. $\mathcal{A}_1^O \cap \mathcal{A}_2^O = \varnothing$ and $\mathcal{C}_1 \cap \mathcal{C}_2 = \varnothing$.

Definition 3.4. Given two compatible IOSAs $\mathcal{I}_1$ and $\mathcal{I}_2$, the parallel composition $\mathcal{I}_1||\mathcal{I}_2$ is a new IOSA $(\mathcal{S}_1 \times \mathcal{S}_2, \mathcal{A}, \mathcal{C}, \rightarrow, \mathcal{C}_0, s_0^1||s_0^2)$ where

(i) $\mathcal{A}^{\mathsf{o}} = \mathcal{A}_1^O \cup \mathcal{A}_2^O$

(ii) $\mathcal{A}^{\mathsf{i}} = (\mathcal{A}_1^I \cup \mathcal{A}_2^I) \setminus \mathcal{A}^{\mathsf{o}}$

(iii) $\mathcal{C} = \mathcal{C}_1 \cup \mathcal{C}_2$

(iv) $\mathcal{C}_0 = \mathcal{C}_0^1 \cup \mathcal{C}_0^2$

and $\rightarrow$ is the smallest relation defined by rules in Table 3.1 where we write $s||t$ instead of (s,t).

Table 3.1: Parallel composition on IOSA

$$\frac{s_1 \xrightarrow{C,a,C'}_1 s_1'}{s_1||s_2 \xrightarrow{C,a,C'} s_1'||s_2} \; a \in \mathcal{A}_1 \backslash \mathcal{A}_2 \tag{R1}$$

$$\frac{s_2 \xrightarrow{C,a,C'}_2 s_2'}{s_1||s_2 \xrightarrow{C,a,C'} s_1||s_2'} \; a \in \mathcal{A}_2 \backslash \mathcal{A}_1 \tag{R2}$$

$$\frac{s_1 \xrightarrow{C_1,a,C_1'}_1 s_1' \quad s_2 \xrightarrow{C_2,a,C_2'}_2 s_2'}{s_1||s_2 \xrightarrow{C_1 \cup C_2, a, C_1' \cup C_2'} s_1'||s_2'} \; a \in \mathcal{A}_1 \cap \mathcal{A}_2 \tag{R3}$$

Rules R1 and R2 from Table 3.1 refer to the interleaving of transitions. This will be the case for all transitions for which the action does not belong to the intersection of $\mathcal{A}_1$ and $\mathcal{A}_2$. On the other hand, rule R3 defines synchronization for all actions in the intersection of the alphabets. Since both automata are compatible, their output actions are disjoint, from which we deduce that action a must be an input action in at least one of $\mathcal{I}_1$ or $\mathcal{I}_2$. In

case it is in both, the resulting transition will be an input transition. Otherwise it would be an output transition (notice item (ii) in definition 3.4). Given IOSA restrictions (a) and (b), the resulting transition will be well formed.

The previous definition is only structural. We need to show that the seven restrictions that define IOSAs also hold in the composed automata.

Theorem 3.1. Let $\mathcal{I}_1$ and $\mathcal{I}_2$ be two compatible IOSAs. Then $\mathcal{I}_1||\mathcal{I}_2$ is indeed an IOSA.

Proof. The proof of restrictions (a), (b), (f), (e), and (g) follow by straightforward inspection on the rules, considering that $\mathcal{I}_1$ and $\mathcal{I}_2$ also satisfy the respective restriction, and doing some case analysis. Since $\mathcal{I}_1$ and $\mathcal{I}_2$ are compatible, restriction (c) also follows by inspecting the rules, taking into account, in addition, that $\mathcal{I}_1$ and $\mathcal{I}_2$ also satisfy restriction (g).

So, we only focus on (d). Suppose $s_1||s_2 \xrightarrow{\{x\},a,C} s_1'||s_2'$. We analyze the case in which $a \in \mathcal{A}_1$ and $x \in \mathcal{C}_1$. The other is symmetric. Moreover, we only consider the case in which $a \in \mathcal{A}_1 \cap \mathcal{A}_2$ since the case $a \in \mathcal{A}_1 \setminus \mathcal{A}_2$ follows similarly.

In this case, we have that $s_1 \xrightarrow{\{x\},a,C_1}_1 s_1'$, $s_2 \xrightarrow{\varnothing,a,C_2}_2 s_2'$, and $C = C_1 \cup C_2$. Let $t_1||t_2 \xrightarrow{C',b,C''} s_1||s_2$. We distinguish three cases:

(i) Suppose $b \in \mathcal{A}_1 \setminus \mathcal{A}_2$. Then $t_1 \xrightarrow{C',b,C''} s_1$ and $t_2 = s_2$. Because $\mathcal{I}_1$ satisfies (d), then either $x \in C''$, or $x \notin C'$ and there exist $t_1 \xrightarrow{\{x\},c,C_3}_1 t_1'$. Hence $x \in C''$, or $x \notin C'$ and there exist t_2' and C_3' such that $t_1||t_2 \xrightarrow{\{x\},c,C_3'} t_1'||t_2'$ (which may occur either by rule (R1) or (R3) if $c \in \mathcal{A}_1 \cap \mathcal{A}_2$).

(ii) If $b \in \mathcal{A}_2 \setminus \mathcal{A}_1$, then $t_2 \xrightarrow{C',b,C''}_2 s_2$ and $t_1 = s_1$. Notice that $C', C'' \subseteq \mathcal{C}_2$ and hence $x \notin C'$ and $x \notin C''$. Moreover, since $\mathcal{I}_2$ is input enabled (restriction (f)), $t_2 \xrightarrow{\varnothing,a,C_3}_2 t_2'$ for some C_3 and t_2'. Then, by rule (R3), $s_1||t_2 \xrightarrow{\{x\},a,C_1 \cup C_3} s_1'||t_2'$ which proves this case.

(iii) If $b \in \mathcal{A}_1 \cap \mathcal{A}_2$, then, by rule (R3), $t_1 \xrightarrow{C_1',b,C_1''}_1 s_1$, $t_2 \xrightarrow{C_2',b,C_2''}_2 s_2$, $C' = C_1' \cup C_2'$ and $C'' = C_1'' \cup C_2''$. Because $\mathcal{I}_1$ satisfies (d), then either $x \in C_1''$, or $x \notin C_1'$ and there exist $t_1 \xrightarrow{\{x\},c,C_3}_1 t_1'$. If $x \in C_1''$, then $x \in C'$ partially proving this case. If instead $x \notin C_1'$ and there exist $t_1 \xrightarrow{\{x\},c,C_3}_1 t_1'$, then $x \notin C''$ (since $x \notin C_2''$ by compatibility), and there

exist t_2' and C_3' such that $t_1||t_2 \xrightarrow{\{x\},c,C_3'} t_1'||t_2'$ (which may occur either by rule (R1) or (R3) if $c \in \mathcal{A}_1 \cap \mathcal{A}_2$), finally proving this case.

□

To prove that bisimulation is a congruence on IOSAs, we first define a parallel composition on NLMPs, prove congruence in this setting, and then show that the semantics of the parallel composition of two IOSAs is isomorphic to the parallel composition of the semantics of each IOSA. From this, it follows that bisimulation is also a congruence for the parallel composition of IOSAs. An important consideration is that NLMPs are not closed for parallel composition [54] in general, although they are in our settings, i.e. when they correspond to the semantic of IOSAs. So we need to require that the parallel composition of NLMPs is also a NLMP as a hypothesis of the congruence theorem on NLMP (Theorem 3.2).

Definition 3.5. Let $\mathcal{P}_i = (\mathbf{S}_i, \Sigma_i, \{\mathcal{T}_a^i \mid a \in \mathcal{L}_i\})$, $i \in \{1,2\}$, be two NLMPs. We define the parallel composition by $\mathcal{P}_1||\mathcal{P}_2 = (\mathbf{S}_1 \times \mathbf{S}_2, \Sigma_1 \otimes \Sigma_2, \{\mathcal{T}_a \mid a \in \mathcal{L}_1 \cup \mathcal{L}_2\})$ where, writing $s_1||s_2$ instead of (s_1, s_2),

(i) $\mathcal{T}_a(s_1||s_2) = \{\mu_1 \times \delta_{s_2} \mid \mu_1 \in \mathcal{T}_a^1(s_1)\}$, if $a \in \mathcal{L}_1 \setminus \mathcal{L}_2$,

(ii) $\mathcal{T}_a(s_1||s_2) = \{\delta_{s_1} \times \mu_2 \mid \mu_2 \in \mathcal{T}_a^2(s_2)\}$, if $a \in \mathcal{L}_2 \setminus \mathcal{L}_1$, and

(iii) $\mathcal{T}_a(s_1||s_2) = \{\mu_1 \times \mu_2 \mid \mu_1 \in \mathcal{T}_a^1(s_1), \mu_2 \in \mathcal{T}_a^2(s_2)\}$, if $a \in \mathcal{L}_1 \cap \mathcal{L}_2$.

The next theorem states that $\sim$ is a congruence for parallel composition whenever the resulting composition is indeed a NLMP. For the definition of $\sim$ we refer the reader to Preliminaries **??**.

Theorem 3.2. Let $\mathcal{P}_i = (\mathbf{S}_i, \Sigma_i, \{\mathcal{T}_a^i \mid a \in \mathcal{L}_i\})$ $i \in \{1,2\}$, be two NLMPs. If $\mathcal{P}_1||\mathcal{P}_2$ is a NLMP, then for all $s_1, s_1' \in \mathbf{S}_1$ and $s_2 \in \mathbf{S}_2$, if $s_1 \sim s_1'$, then $s_1||s_2 \sim s_1'||s_2$ and $s_2||s_1 \sim s_2||s_1'$.

Proof. We only prove that $s_1||s_2 \sim s_1'||s_2$. The other case is symmetric. Let $R \subseteq \mathbf{S}_1 \times \mathbf{S}_1$ be a bisimulation relation. Define $R' \subseteq (\mathbf{S}_1 \times \mathbf{S}_2) \times (\mathbf{S}_1 \times \mathbf{S}_2)$ by $R' = \{(s_1||s_2, s_1'||s_2) \mid (s_1, s_1') \in R, s_2 \in \mathbf{S}_2\}$. We prove that R' is a bisimulation by doing case analysis on the definition of the transition relation in the parallel composition.

Suppose in general that $s_1||s_2 \; R' \; s_1'||s_2$, and consider the case in which $\mathcal{T}_a(s_1||s_2)$ results from (i) in Definition 3.5. Let $\mu_1 \times \delta_{s_2} \in \mathcal{T}_a(s_1||s_2)$ with

$\mu_1 \in \mathcal{T}_a^1(s_1)$. Since $s_1\ R\ s_1'$, there exists $\mu_1' \in \mathcal{T}_a^1(s_1')$ such that $\mu_1\ R\ \mu_1'$. Let $Q \in \Sigma_1 \otimes \Sigma_2$ be R'-closed and define $Q|_{s_2} = \{s_1 \mid s_1||s_2 \in Q\}$. $Q|_{s_2}$ is measurable in Σ_1 [3], and can be easily proven to be R-closed. Now we can calculate:

$$\begin{aligned}(\mu_1 \times \delta_{s_2})(Q) &= (\mu_1 \times \delta_{s_2})(Q|_{s_2} \times \{s_2\}) = \mu_1(Q|_{s_2}) \\ &\overset{(*)}{=} \mu_1'(Q|_{s_2}) = (\mu_1' \times \delta_{s_2})(Q|_{s_2} \times \{s_2\}) = (\mu_1' \times \delta_{s_2})(Q)\end{aligned}$$

where equality $(*)$ follows from $\mu_1\ R\ \mu_1'$, and hence $(\mu_1 \times \delta_{s_2})\ R'\ (\mu_1' \times \delta_{s_2})$.

Case (ii) in Definition 3.5 follows with a similar analysis, so we focus on case (iii). Let $\mu_1 \times \mu_2 \in \mathcal{T}_a(s_1||s_2)$ with $\mu_1 \in \mathcal{T}_a^1(s_1)$. Since $s_1\ R\ s_1'$, there exists $\mu_1' \in \mathcal{T}_a^1(s_1')$ such that $\mu_1\ R\ \mu_1'$. Let $Q \in \Sigma_1 \otimes \Sigma_2$ be R'-closed. Using Fubini's theorem [3], we calculate:

$$\begin{aligned}(\mu_1 \times \mu_2)(Q) &= \int_{\mathbf{S}_2}\int_{\mathbf{S}_1} 1_Q(x,y)\ d\mu_1(x)\ d\mu_2(y) \\ &= \int_{\mathbf{S}_2}\int_{\mathbf{S}_1} 1_{Q|_y}(x)\ d\mu_1(x)\ d\mu_2(y) \\ &= \int_{\mathbf{S}_2} \mu_1(Q|_y)\ d\mu_2(y) \\ &\overset{(*)}{=} \int_{\mathbf{S}_2} \mu_1'(Q|_y)\ d\mu_2(y) \\ &= (\mu_1' \times \mu_2)(Q)\end{aligned}$$

where 1_Q is the usual characteristic function, and $(*)$ follows from $\mu_1\ R\ \mu_1'$. Therefore $(\mu_1 \times \mu_2)\ R'\ (\mu_1' \times \mu_2)$. □

Next, we prove that the semantic interpretation of IOSAs and parallel composition commutes, that is, that the NLMP resulting from interpreting a parallel composition of two IOSAs is isomorphic to the parallel composition of the two NLMPs interpreting each of the IOSAs.

Theorem 3.3. Given two IOSAs $\mathcal{I}_1$ and $\mathcal{I}_2$, there is an isomorphism between (the reachable parts of) $\mathcal{P}(\mathcal{I}_1||\mathcal{I}_2)$ and $\mathcal{P}(\mathcal{I}_1)||\mathcal{P}(\mathcal{I}_2)$.

Proof. Let N and M be the number of clocks in $\mathcal{I}_1$ and $\mathcal{I}_2$, respectively. Let $\mathbf{S} = ((\mathcal{S}_1 \times \mathcal{S}_2) \cup \{\mathsf{init}\}) \times \mathbb{R}^{N+M}$ and $\mathbf{S}' = ((\mathcal{S}_1 \times \mathbb{R}^N) \times (\mathcal{S}_2 \times \mathbb{R}^M)) \cup (((\{\mathsf{init}\} \times \mathbb{R}^N) \times (\{\mathsf{init}\} \times \mathbb{R}^M))$ be the states of $\mathcal{P}(\mathcal{I}_1||\mathcal{I}_2)$ and $\mathcal{P}(\mathcal{I}_1)||\mathcal{P}(\mathcal{I}_2)$,

respectively [1]. The isomorphism is given by function $f : \mathbf{S} \to \mathbf{S}'$ defined by $f(\mathsf{init}, \vec{v}_1\vec{v}_2) = (\mathsf{init}, \vec{v}_1)||(\mathsf{init}, \vec{v}_2)$, and $f((s_1||s_2), \vec{v}_1\vec{v}_2) = (s_1, \vec{v}_1)||(s_2, \vec{v}_2)$ for all $s_1 \in \mathcal{S}_1$, $s_2 \in \mathcal{S}_2$, and vectors $\vec{v}_1$ and $\vec{v}_2$ which represent valuations on the sets of clocks $\mathcal{C}_1$ and $\mathcal{C}_2$ respectively. f is clearly bijective, and it can be proved straightforwardly that both f and f^{-1} are measurable (i.e. f is *bimeasurable*). From this, it follows that the measurable spaces $(\mathbf{S}, \mathscr{B}(\mathbf{S}))$ and $(\mathbf{S}', \mathscr{B}(\mathbf{S}'))$ are isomorphic.

Following [50], f induces a map $\Delta f : \Delta(\mathbf{S}) \to \Delta(\mathbf{S}')$ defined by $\Delta f(\mu) = \mu \circ f^{-1}$. It is not difficult to prove that Δf is bijective and bimeasurable. Hence, $(\Delta(\mathbf{S}), \Delta(\mathscr{B}(\mathbf{S})))$ and $(\Delta(\mathbf{S}'), \Delta(\mathscr{B}(\mathbf{S}')))$ are isomorphic.

We can lift f a second time to obtain an isomorphism on hit σ-algebras. Define [2] $Hf : \Delta(\mathscr{B}(\mathbf{S}')) \to \Delta(\mathscr{B}(\mathbf{S}))$ by $Hf = (\Delta f)^{-1}$. Again Hf can be proven to be bijective and bimeasurable and hence, $(\Delta(\mathscr{B}(\mathbf{S})), H(\Delta(\mathscr{B}(\mathbf{S}))))$ and $(\Delta(\mathscr{B}(\mathbf{S}')), H(\Delta(\mathscr{B}(\mathbf{S}'))))$ are isomorphic.

Now, it is not difficult to see that for all $a \in \mathcal{L}$, $\mathcal{T}_a(r) = Hf(\mathcal{T}'_a(f(r)))$ for all $r \in \mathbf{S}$ where $\mathcal{T}_a$ and $\mathcal{T}'_a$ are the transition functions on $\mathcal{P}(\mathcal{I}_1||\mathcal{I}_2)$ and $\mathcal{P}(\mathcal{I}_1)||\mathcal{P}(\mathcal{I}_2)$, respectively. This proves that both NLMPs are isomorphic. □

Given two NLMPs $\mathcal{P}_1$ and $\mathcal{P}_2$ with the same set of labels, the definition of bisimulation can be extended to states in the different NLMPs by constructing the NLMP induced by the coproduct σ-algebra. The NLMP $\mathcal{P}_1 \oplus \mathcal{P}_2$ is defined by the structure $(\mathbf{S}_1 \uplus \mathbf{S}_2, \Sigma_1 \oplus \Sigma_2, \{\mathcal{T}_a \mid a \in \mathcal{L}\})$ where, for all $s \in \mathbf{S}_1 \uplus \mathbf{S}_2$ and $a \in \mathcal{L}$, $\mathcal{T}_a(s) = \mathcal{T}^1_a(s)$ if $s \in \mathbf{S}_1$ and $\mathcal{T}_a(s) = \mathcal{T}^2_a(s)$ if $s \in \mathbf{S}_2$. Thus, if s_1 and s_2 are states of $\mathcal{P}_1$ and $\mathcal{P}_2$ respectively, $s_1 \sim s_2$ whenever they are bisimilar in $\mathcal{P}_1 \oplus \mathcal{P}_2$.

By [50, Prop. 3.6], the next corollary follows immediately from Theorem 3.3.

Corollary 3.1. For any $\vec{v}_1$ and $\vec{v}_2$ representing valuations of clocks in $\mathcal{I}_1$ and $\mathcal{I}_2$, resp., $(\mathsf{init}, \vec{v}_1\vec{v}_2) \sim (\mathsf{init}, \vec{v}_1)||(\mathsf{init}, \vec{v}_2)$ and $((s_1||s_2), \vec{v}_1\vec{v}_2) \sim (s_1, \vec{v}_1)||(s_2, \vec{v}_2)$.

[1] Strictly speaking, $\mathcal{P}(\mathcal{I}_1)||\mathcal{P}(\mathcal{I}_2)$ should also contain states of the form $(s, \vec{v}_1)||(\mathsf{init}, \vec{v}_2)$ and $(\mathsf{init}, \vec{v}_1)||(s, \vec{v}_2)$ with $s \neq \mathsf{init}$. Nonetheless, these states are not reachable. Thus, we do not consider them since otherwise the result would not be strictly an isomorphism and it would only add irrelevant technical problems to the proof.

[2] Note that the domain and image of Hf appear apparently inverted. This is necessary in [50] since they only deal with morphisms, and we are following their definitions. In our case, we could have also defined a direct map from $\Delta(\mathscr{B}(\mathbf{S}))$ to $\Delta(\mathscr{B}(\mathbf{S}'))$ since Δf is bimeasurable, namely $H(f^{-1}) = (\Delta(f^{-1}))^{-1}$.

We say that two IOSAs $\mathcal{I}_1$ and $\mathcal{I}_2$ are bisimilar, notation $\mathcal{I}_1 \sim \mathcal{I}_2$ whenever $(\mathsf{init}, \vec{v}_1) \sim (\mathsf{init}, \vec{v}_2)$ for any vectors $\vec{v}_1$ and $\vec{v}_2$ representing the valuations of clocks in $\mathcal{I}_1$ and $\mathcal{I}_2$, respectively.

Then, the fact that bisimulation equivalence is a congruence on IOSAs follows from Theorem 3.2 and Corollary 3.1 and it is stated in the following theorem.

Theorem 3.4. Let $\mathcal{I}_1$ and $\mathcal{I}_2$ be two IOSAs such that $\mathcal{I}_1 \sim \mathcal{I}_2$. Then, for any IOSA $\mathcal{I}_3$, $\mathcal{I}_1||\mathcal{I}_3 \sim \mathcal{I}_2||\mathcal{I}_3$ and $\mathcal{I}_3||\mathcal{I}_1 \sim \mathcal{I}_3||\mathcal{I}_2$.

3.6 Determinism

A *closed* IOSA is an IOSA in which all synchronizations, if ever existed in the model, have been resolved through parallel composition. Therefore, it has no input actions (i.e. $\mathcal{A}^{\mathsf{i}} = \varnothing$), and hence represents a fully generative model. It is just then reasonable to talk about determinism of the model.

In this section we show that a closed IOSA is deterministic. Deterministic IOSA are amenable for discrete event simulation or, in case all its clocks are exponentially distributed random variables, also amenable for analysis as a continuous time Markov chain. We will say that an IOSA is deterministic if almost surely at most one discrete transition is enabled at every time point. To avoid referring explicitly to time, we say instead that an IOSA is deterministic if almost never reaches a state in which two different discrete transitions are enabled.

Definition 3.6. An IOSA $\mathcal{I}$ is *deterministic* whenever in $\mathcal{P}(\mathcal{I}) = (\mathbf{S}, \mathscr{B}(\mathbf{S}), \{\mathcal{T}_a \mid a \in \mathcal{L}\})$, a state $(s, \vec{v}) \in \mathbf{S}$ such that $\bigcup_{a \in \mathcal{A} \cup \{\mathsf{init}\}} \mathcal{T}_a(s, \vec{v})$ contains at least two different probability measures, is almost never reached from any $(\mathsf{init}, \vec{v}') \in \mathbf{S}$.

By "almost never" we mean that the measure of the set of all paths leading to a state $(s, \vec{v}) \in \mathbf{S}$ such that $\bigcup_{a \in \mathcal{A} \cup \{\mathsf{init}\}} \mathcal{T}_a(s, \vec{v})$ contains at least two elements is 0. A strictly formal definition of this requires a series of definitions related to schedulers and measures on paths in NLMPs which is not crucial for the developing of the result. (For a formal definition of scheduler and probability measures on paths in NLMPs see [102, Chap. 7].)

Of course, to ensure determinism, timed transitions have to be taken into account too. Thus, the previous definition only makes sense if $\mathcal{P}(\mathcal{I})$ satisfies *time additivity*, *time determinism*, and *maximal progress* [105]. Particularly,

by maximal progress we understand that time cannot progress if an output transition is enabled.

Theorem 3.5. For an IOSA $\mathcal{I}$, its semantics $\mathcal{P}(\mathcal{I}) = (\mathbf{S}, \mathscr{B}(\mathbf{S}), \{\mathcal{T}_a \mid a \in \mathcal{L}\})$ satisfies, for all $(s, \vec{v}) \in \mathbf{S}$, $a \in \mathcal{A}^\circ$ and $d, d' \in \mathbb{R}_{>0}$,

maximal progress: $\mathcal{T}_a(s, \vec{v}) \neq \varnothing \;\Rightarrow\; \mathcal{T}_d(s, \vec{v}) = \varnothing$

time determinism: $\mu, \mu' \in \mathcal{T}_d(s, \vec{v}) \;\Rightarrow\; \mu = \mu'$, and

time additivity: $\delta^{-d}_{(s,\vec{v})} \in \mathcal{T}_d(s, \vec{v}) \wedge \delta^{-d'}_{(s,\vec{v}-d)} \in \mathcal{T}_{d'}(s, \vec{v} - d) \;\Leftrightarrow\; \delta^{-(d+d')}_{(s,\vec{v})} \in \mathcal{T}_{d+d'}(s, \vec{v})$.

Proof. Notice that if $\mathcal{T}_a(s, \vec{v}) \neq \varnothing$, with $a \in \mathcal{A}^\circ$, then there exists a transition $s \xrightarrow{\{x_j\}, a, C'} s'$ such that $\vec{v}(j) \leq 0$. Suppose by contradiction that $\mathcal{T}_d(s, \vec{v}) \neq \varnothing$, then $0 < d \leq \min\{\vec{v}(i) \mid \exists a \in \mathcal{A}^\circ, C' \subseteq \mathcal{C}, s' \in S : s \xrightarrow{\{x_i\}, a, C'} s'\} \leq \vec{v}(j) \leq 0$, which is a contradiction.

Time determinism is immediate by Definition 3.2 since either $\mathcal{T}_d(s, \vec{v}) = \{\delta^{-d}_{(s,\vec{v})}\}$ or $\mathcal{T}_d(s, \vec{v}) = \varnothing$.

For time additivity, let $\hat{d} = \min\{\vec{v}(i) \mid \exists a \in \mathcal{A}^\circ, C \subseteq \mathcal{C}, s' \in \mathcal{S} : s \xrightarrow{\{x_i\}, a, C} s'\}$. Suppose $\delta^{-d}_{(s,\vec{v})} \in \mathcal{T}_d(s, \vec{v})$ and $\delta^{-d'}_{(s,\vec{v}-d)} \in \mathcal{T}_{d'}(s, \vec{v} - d)$. By Definition 3.2, $0 < d \leq \hat{d}$ and $0 < d' \leq \hat{d} - d$, i.e. $0 < d + d' \leq \hat{d}$. Thus $\delta^{-(d+d')}_{(s,\vec{v})} \in \mathcal{T}_{d+d'}(s, \vec{v})$. Suppose now that $\delta^{-(d+d')}_{(s,\vec{v})} \in \mathcal{T}_{d+d'}(s, \vec{v})$. Then $0 < d + d' \leq \hat{d}$ and thus $0 < d \leq \hat{d}$ and $0 < d' \leq \hat{d} - d$, which implies that $\delta^{-d}_{(s,\vec{v})} \in \mathcal{T}_d(s, \vec{v})$ and $\delta^{-d'}_{(s,\vec{v}-d)} \in \mathcal{T}_{d'}(s, \vec{v} - d)$. □

We now present the main theorem of this section.

Theorem 3.6. Every closed IOSA is deterministic.

The rest of the section is devoted to proving this theorem. From now on, we work with the closed IOSA $\mathcal{I} = (\mathcal{S}, \mathcal{C}, \mathcal{A}, \rightarrow, s_0, \mathcal{C}_0)$, with $|\mathcal{C}| = N$, and its semantics $\mathcal{P}(\mathcal{I}) = (\mathbf{S}, \mathscr{B}(\mathbf{S}), \{\mathcal{T}_a \mid a \in \mathcal{L}\})$. We recall that IOSAs only admit sampling clock values from continuous random variables, which is essential for the validity of Theorem 3.6.

For every state $s \in \mathcal{S}$, let $\mathit{active}(s) = \{x \mid s \xrightarrow{\{x\}, a, C} s'\}$ be the set of active clocks at state s. By Definition 3.1(d) it follows that $\mathit{active}(s') \subseteq (\mathit{active}(s) \backslash \{x\}) \cup C$ whenever $s \xrightarrow{\{x\}, a, C} s'$.

The idea of the proof of Theorem 3.6 is to show that the property that all active clocks have non-negative values and they are different from each other is almost surely an invariant of $\mathcal{I}$, and that at most one transition is enabled in every state satisfying such invariant. Formally, the invariant is the set

$$\begin{aligned}\mathsf{Inv} = \{(s,\vec{v}) \mid s \in \mathcal{S}, \vec{v}(i) \neq \vec{v}(j), \text{ and } \vec{v}(i) \geq 0 \\ \text{for all } x_i, x_j \in \mathit{active}(s) \text{ with } i \neq j\} \cup (\{\mathsf{init}\}\times\mathbb{R}^N)\end{aligned} \tag{3.1}$$

therefore, its complement set is

$$\begin{aligned}\mathsf{Inv}^c = \{(s,\vec{w}) \mid s \in \mathcal{S}, \vec{w}(i) = \vec{w}(j) \text{ for some } x_i, x_j \in \mathit{active}(s) \text{ with } i \neq j\} \\ \cup\ \{(s,\vec{w}) \mid s \in \mathcal{S}, \vec{w}(i) < 0 \text{ for some } x_i \in \mathit{active}(s)\}\end{aligned} \tag{3.2}$$

The next lemma states that Inv^c is almost never reached in one step from a state satisfying the invariant.

Lemma 3.3. For all $(s,\vec{v}) \in \mathsf{Inv}$, $a \in \mathcal{L}$, and $\mu \in \mathcal{T}_a(s,\vec{v})$, $\mu(\mathsf{Inv}^c) = 0$.

Proof. We proceed analyzing by cases, according a is init, in $\mathcal{A}$, or in $\mathbb{R}_{>0}$.

For $a = \mathsf{init}$, we only consider states of the form $(\mathsf{init},\vec{v})$ since $\mathcal{T}_{\mathsf{init}}(s,\vec{v}) \neq \varnothing$ iff $s = \mathsf{init}$. So, let $\mu \in \mathcal{T}_{\mathsf{init}}(\mathsf{init},\vec{v})$. Then $\mu = \delta_{s_0} \times \prod_{i=1}^{N} \mu_{x_i}$. Since each μ_{x_i} is a continuous probability measure (hence the likelihood that two clocks are set to the same value is 0) and $\mu_{x_i}(\mathbb{R}_{>0}) = 1$, then $\mu(\mathsf{Inv}^c) = 0$.

For $a \in \mathcal{A}$, take $\mu \in \mathcal{T}_a(s,\vec{v})$ with $(s,\vec{v}) \in \mathsf{Inv}$. Notice that $s \in \mathcal{S}$. By Definition 3.2 and because $\mathcal{I}$ is closed, there exists $s \xrightarrow{\{x\},a,C} s'$ with $\vec{v}(i) \leq 0$ and $\mu = \mu_{\vec{v},C,s'} = \delta_{s'} \times \prod_{i\in I} \mu_{x_i} \times \prod_{j\in J} \delta_{\vec{v}(j)}$ where $I = \{i \mid x_i \in C\}$ and $J = \{j \mid x_j \notin C\}$.

For each $x_i, x_j \in \mathit{active}(s')$ define $\mathsf{Inv}^c_{ij} = \{(s'',\vec{w}) \mid s'' \in \mathcal{S}, \vec{w}(i) = \vec{w}(j)\}$ whenever $i \neq j$, and $\mathsf{Inv}^c_i = \{(s'',\vec{w}) \mid s'' \in \mathcal{S}, \vec{w}(i) < 0\}$. Notice that $\mathsf{Inv}^c = \bigcup \mathsf{Inv}^c_{ij} \cup \bigcup \mathsf{Inv}^c_i$ and, since the unions are finite, $\mu(\mathsf{Inv}^c) = 0$ iff $\mu(\mathsf{Inv}^c_{ij}) = 0$ and $\mu(\mathsf{Inv}^c_i) = 0$, for every i, j. In the following, we show this last statement.

Let $x_i \in \mathit{active}(s')$. Then $x_i \in (\mathit{active}(s)\backslash\{x\}) \cup C$. If $x_i \in C$, then $\mu(\mathsf{Inv}^c_i) = 0$ because $\mu_i(\mathbb{R}_{\geq 0}) = 1$. If instead $x_i \in \mathit{active}(s)\backslash\{x\}$, then $\mu(\mathsf{Inv}^c_i) = 0$ because $\delta_{\vec{v}(i)}(\mathbb{R}_{\geq 0}) = 1$, since $(s,\vec{v}) \in \mathsf{Inv}$ and hence $\vec{v}(i) \geq 0$.

Let $x_i, x_j \in \mathit{active}(s')$ with $i \neq j$. Then $x_i, x_j \in (\mathit{active}(s)\backslash\{x\}) \cup C$. If $x_i \in C$ then μ_i is a continuous probability measure and hence $\mu(\mathsf{Inv}^c_{ij}) = 0$. Similarly if $x_j \in C$. If instead $x_i, x_j \in \mathit{active}(s)\backslash\{x\}$, then $\delta_{\vec{v}(i)} \neq \delta_{\vec{v}(j)}$ because $(s,\vec{v}) \in \mathsf{Inv}$ and hence $\vec{v}(i) \neq \vec{v}(j)$. Therefore $\mu(\mathsf{Inv}^c_{ij}) = 0$. This proves that $\mu(\mathsf{Inv}^c) = 0$ for this case.

Finally, take $d \in \mathbb{R}_{>0}$ and suppose that $\mathcal{T}_d(s,\vec{v}) = \{\delta^{-d}_{(s,\vec{v})}\}$ with $(s,\vec{v}) \in \mathsf{Inv}$. Notice that $s \in \mathcal{S}$. By Definition 3.2, $0 < d \leq \min\{\vec{v}(k) \mid s \xrightarrow{\{x_k\},a,C'} s', a{\in}\mathcal{A}^{\circ}\}$ and $\delta^{-d}_{(s,\vec{v})} = \delta_s \times \prod_{i=1}^{N} \delta_{\vec{v}(i)-d}$. We take sets Inv^c_{ij} and Inv^c_i as before and follow a similar reasoning. For $x_i \in \mathit{active}(s)$, $\vec{v}(i){-}d \geq \min\{\vec{v}(k) \mid s \xrightarrow{\{x_k\},a,C'} s', a{\in}\mathcal{A}^{\circ}\}{-}d \geq 0$ and hence $\delta_{\vec{v}(i)-d}(\mathbb{R}_{\geq 0}) = 1$. Therefore $\mu(\mathsf{Inv}^c_i) = 0$. For $x_i, x_j \in \mathit{active}(s)$ with $i{\neq}j$, $\delta_{\vec{v}(i)-d} \neq \delta_{\vec{v}(j)-d}$ because $(s,\vec{v}) \in \mathsf{Inv}$ and hence $\vec{v}(i) \neq \vec{v}(j)$. So $\mu(\mathsf{Inv}^c_{ij}) = 0$. This proves that $\mu(\mathsf{Inv}^c) = 0$ for this case, and hence the lemma. □

From Lemma 3.3 we have the following corollary.

Corollary 3.2. The set Inv^c is almost never reachable in $\mathcal{P}(\mathcal{I})$.

The proof of the corollary requires, again, the definitions related to schedulers and measures on paths in NLMPs. We omit it here since the proof eventually boils down to directly applying Lemma 3.3 and seeing that the measure of all paths leading to a state in Inv^c is 0 for all possible schedulers.

The next lemma states that any state in the invariant Inv has at most one discrete transition enabled.

Lemma 3.4. For all $(s,\vec{v}) \in \mathsf{Inv}$, the set $\mathit{enabled}(s,\vec{v}) = \bigcup_{a \in \mathcal{A} \cup \{\mathsf{init}\}} \mathcal{T}_a(s,\vec{v})$ is either a singleton set or the empty set.

Proof. By Def 3.2, $\mathit{enabled}(\mathsf{init},\vec{v}) = \mathcal{T}_{\mathsf{init}}(s,\vec{v}) = \{\delta_{s_0} \times \prod_{i=1}^{N} \mu_{x_i}\}$, which proves this case. So, let $(s,\vec{v}) \in \mathsf{Inv}$ with $s \in \mathcal{S}$ and suppose that $\mathit{enabled}(s,\vec{v}) \neq \varnothing$. By Def 3.2, there is at least one transition $s \xrightarrow{\{x_i\},a,C} s'$ such that $\vec{v}(i) \leq 0$. Because, $(s,\vec{v}) \in \mathsf{Inv}$ and $x_i \in \mathit{active}(s)$, then $\vec{v}(i) = 0$ and for all $x_j \in \mathit{active}(s)$ with $i \neq j$, $\vec{v}(j) > 0$. Condition (c) in Definition 3.1 ensures that there is no other transition $s \xrightarrow{\{x_i\},b,C'} s''$ and, as a consequence, $\mathit{enabled}(s,\vec{v})$ is a singleton set. □

Finally, the proof of Theorem 3.6 is a direct consequence of Corollary 3.2 and Lemma 3.4.

Proof of Theorem 3.6. Let $\mathsf{En}_{\geq 2} = \{(s,\vec{v}) \in \mathbf{S} \mid \ |\mathit{enabled}(s,\vec{v})| \geq 2\}$. By Corollary 3.2, $\mathsf{En}_{\geq 2} \subseteq \mathsf{Inv}^c$. Therefore, by Lemma 3.4, $\mathsf{En}_{\geq 2}$ is almost never reachable. □

3.7 Conclusion

In this chapter we have defined Input/Output Stochastic Automata. IOSA enjoy the following important characteristics:

- IOSA is compositional, enabling reuse of components, significantly reducing the state space explosion problem, and easing the engineering of modeling systems.
- IOSA allows to model events that occur according to general continuous probability measures and not only memoryless.
- Closed IOSA models are warrantied to be fully deterministic, that is, all choices are resolved probabilistically.

These characteristics enable to use IOSA as an efficient and reality-fair model for discrete event simulation. It has been used as the input specification language in the first version of FIG rare event simulation tool (see Section 5.8.1).

All the formalities needed to support the language have been treated in this chapter: its semantic has been given in terms of NLMPs. A bisimulation relation has been defined (on NLMPs and then lifted to IOSAs) as well as a parallel composition operator. Furthermore bisimulation on IOSAs has been shown to be a congruence with respect to the parallel composition. Finally we have proved that closed IOSA models are fully deterministic and hence amenable to discrete event simulation without any further intervention of schedulers nor human expertise to solve non-determinism.

Chapter 4

IOSA with Urgency

In the previous chapter we presented IOSA, a modeling language for general distributed stochastic timed systems. From its features we highlighted (i) the possibility to define general distributed occurrence of events, allowing for a better representation of the real systems to be model in comparison with other formalisms which only allow memoryless distributed time stamps; (ii) that a closed IOSA model is deterministic and thus amanable to discrete event simulation; (iii) that compositionality on IOSA attacks the exponential growth of the state space, resulting in a more efficient analysis framework, and providing cleaner and reusable models.

This chapter and, more specifically, the introduction of urgent transitions in IOSA find motivation on the fact that compositionality in IOSA is greatly limited by synchronization being taken only upon the expiration of a clock (see parallelization rule R3). This constraints our capabilities for modularization and makes a limited use of the benefits of compositionality.

In fact, by experimenting with the rare event simulation tool FIG [28, 25, 41], we have often experienced difficulties on compositional modeling and this finally leads us to model a big monolithic component instead of the more natural compositional model that would arise more naturally in other languages such as Modest [14, 60, 61].

To illustrate this problem, in Figure 4.1 we find a graphical model of a full two bits adder. Like other electronic devices it is described by gates, which produce output signals as a logical combination of inputs. An intuitive and reusable way of modeling such a device should take into account that all gates of a given type work the same. Thus, a good design should, in principle, model each kind of gate as a separate component, and combine their inputs

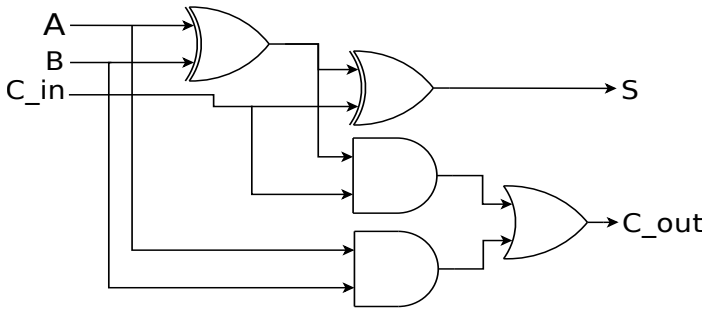


Figure 4.1: Tow-bit full adder. The adder sums bit inputs A and B, taking into account the carry in bit C_{in}. The result is signaled at S and if there is any carry out, then it is placed at C_{out}.

and outputs in order to synchronize with each other and produce the desired result. A usual abstraction when analyzing this kind of systems, considers that gates work instantaneously, and communication between them is also assumed to be instantaneous. Hence, a compositional approach as the one described becomes impossible to obtain with the IOSA model introduced in the previous chapter, since synchronization between gate components is obliged to introduce clocks. Thus, we are forced to produce a monolithic model in IOSA.

In this chapter we introduce *Input/Output Stochastic Automata with Urgency* (IOSA$_u$), an extension of IOSA with urgent actions. The occurrence of output urgent transitions, i.e. those labeled by output urgent actions, are not govern by the expiration of a clock. Instead, an urgent transition is taken immediately, as soon as a state where it is enabled is reached. Urgent outputs are still considered generative and urgent inputs reactive, but neither of them have clock constraints in their guards.

Whit the introduction of this new kind of transitions, we improve one of the most useful characteristics of IOSA, namely its compositionality, but on the other hand we have a drawback on another of its interesting features: this extension introduces non-determinism even in the closed models. Nevertheless, we will find out that this non-determinism is produced between urgent actions and that, usually, it is introduced by the interleaving of *confluent* urgent actions. This non-determinism results to be spurious in the sense that it does not change the stochastic results of the behavior of the system.

Based on [36], we define a notion of weak determinism for confluent

IOSA$_u$s, proving that these automata maintain their stochastic behavior regardless the resolution of the non-determinism introduced by urgent actions. Also based on the work of Crouzen [36], we provide sufficient conditions to ensure that a network of interacting IOSA$_u$s is confluent, and thus weak deterministic, without the need to obtain the composed model, hence avoiding the usual state explosion problem. Such conditions can be verified in polynomial time.

4.1 Input/Output Stochastic Automata with urgency (IOSA$_u$)

The formal definition of IOSA$_u$ is similar to that of IOSA. Just as before, conditions are given with the purpose of ensuring that the defined structure is deterministic in the absence of urgent actions. In the case of IOSA$_u$, not only input actions but also urgent actions lack of an enabling clock. This corresponds to the fact that they will trigger as soon as the state is reached.

Definition 4.1. An *input/output stochastic automaton with urgency* is a structure $(\mathcal{S}, \mathcal{A}, \mathcal{C}, \rightarrow, C_0, s_0)$, where

- $\mathcal{S}$ is a (denumerable) set of states,
- $\mathcal{A}$ is a (denumerable) set of labels partitioned into disjoint sets of *input* labels $\mathcal{A}^{\mathsf{i}}$ and *output* labels $\mathcal{A}^{\mathsf{o}}$, from which a subset $\mathcal{A}^{\mathsf{u}} \subseteq \mathcal{A}$ is marked as *urgent*,
- $\mathcal{C}$ is a (finite) set of clocks such that each $x \in \mathcal{C}$ has an associated continuous probability measure μ_x on $\mathbb{R}$ s.t. $\mu_x(\mathbb{R}_{>0}) = 1$,
- $\rightarrow \subseteq \mathcal{S} \times \mathcal{C} \times \mathcal{A} \times \mathcal{C} \times S$ is a transition function,
- C_0 is the set of clocks that are initialized in the initial state, and
- $s_0 \in \mathcal{S}$ is the initial state.

In addition an IOSA$_u$ should satisfy the following constraints:

(a) If $s \xrightarrow{C,a,C'} s'$ and $a \in \mathcal{A}^{\mathsf{i}} \cup \mathcal{A}^{\mathsf{u}}$, then $C = \varnothing$.

(b) If $s \xrightarrow{C,a,C'} s'$ and $a \in \mathcal{A}^{\mathsf{o}} \setminus \mathcal{A}^{\mathsf{u}}$, then C is a singleton set.

(c) If $s \xrightarrow{\{x\},a_1,C_1} s_1$ and $s \xrightarrow{\{x\},a_2,C_2} s_2$ then $a_1 = a_2$, $C_1 = C_2$ and $s_1 = s_2$.

(d) For every $a \in \mathcal{A}^{\mathsf{i}}$ and state s, there exists a transition $s \xrightarrow{\varnothing,a,C} s'$.

(e) For every $a \in \mathcal{A}^{\mathsf{i}}$, if $s \xrightarrow{\varnothing,a,C'_1} s_1$ and $s \xrightarrow{\varnothing,a,C'_2} s_2$, $C'_1 = C'_2$ and $s_1 = s_2$.

(f) There exists a function $active : \mathcal{S} \to 2^{\mathcal{C}}$ such that:

(i) $active(s_0) \subseteq C_0$,

(ii) $enabling(s) \subseteq active(s)$,

(iii) if s is stable, $active(s) = enabling(s)$, and

(iv) if $t \xrightarrow{C,a,C'} s$ then $active(s) \subseteq (active(t) \setminus C) \cup C'$.

where $enabling(s) = \{y \mid s \xrightarrow{\{y\},\_,\_} \_\}$, and s is *stable* if there is no $a \in \mathcal{A}^{\mathsf{u}} \cap \mathcal{A}^{\mathsf{o}}$ such that $s \xrightarrow{\varnothing,a,\_} \_$. ($\_$ indicates the existential quantification of a parameter.)

The occurrence of an output transition is controlled by the expiration of clocks. If $a \in \mathcal{A}^{\mathsf{o}}$, $s \xrightarrow{C,a,C'} s'$ indicates that there is a transition from state s to state s' that can be taken only when all clocks in C have expired and, when taken, it triggers action a and sets all clocks in C' to a value sampled from their associated probability distribution. Notice that if $C = \varnothing$ (which means $a \in \mathcal{A}^{\mathsf{o}} \cap \mathcal{A}^{\mathsf{u}}$) $s \xrightarrow{C,a,C'} s'$ is immediately triggered. Instead, if $a \in \mathcal{A}^{\mathsf{i}}$, $s \xrightarrow{\varnothing,a,C'} s'$ is only intended to take place if an external output synchronizes with it, which, in terms of an open system semantic, means that it may take place at any possible time.

Restrictions (a) to (f) ensure that any *closed* IOSA$_u$ without urgent actions is deterministic, just as in the original IOSA. An IOSA$_u$ is closed if all its synchronizations have been resolved, that is, the IOSA$_u$ resulting from a composition does not have input actions ($\mathcal{A}^{\mathsf{i}} = \varnothing$). Since this restrictions are very similar to those for IOSA (3.1), we just go through the differences. Restriction (a) is two-folded: on the one hand, it specifies that output urgent actions must occur as soon as the enabling state is reached, on the other hand, as input actions are reactive and their time occurrence can only depend on the interaction with an output, no clock can control their enabling.

Finally, (f) ensures that clocks enabling some output transition have not expired before, that is, they have not been used before by another output

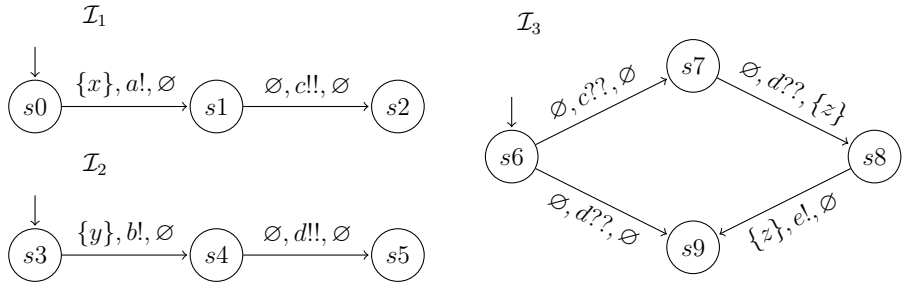


Figure 4.2: Examples of IOSA$_u$s.

transition (without being reset in between) nor inadvertedly reached zero. Function *active* collects all clocks that are required to be active (i.e. that have been set but not yet expired) at each state. Notice that enabling clocks are required to be active (conditions (f)(ii) and (f)(iii)). Also note that clocks that are active in a state are allowed to remain active in a successor state as long as the clock has not been used, and clocks that has just been set may become active in the successor state (condition (f)(iv)). Furthermore, this condition ensures that all the initially enabled clocks are included in the initial clocks set. Note that this condition replaces conditions (d) and (e) from definition 3.1. In this case, we need to take into account clocks that are reset by a series of urgent transitions eventually arriving to the stable state.

Figure 4.2 depicts three simple examples of IOSA$_u$s. Although IOSA$_u$s are input enabled, we have omitted self loops of input enabling transitions for the sake of readability. In the figure, we represent output actions suffixed by '!' and by '!!' when they are urgent, and input actions suffixed by '?' and by '??' when they are urgent.

4.2 Semantics of IOSA$_u$

Just as with IOSA, the semantic of IOSA$_u$ is defined in terms of non-deterministic labeled Markov processes (NLMP) which extends LMP with *internal* non-determinism (see Section 2.2).

The formal semantic of an IOSA$_u$ is defined by a NLMP with two classes of transitions: one that encodes the discrete steps and contains all the probabilistic information introduced by the sampling of clocks, and another describing the time steps that only records the passage of time synchronously

decreasing the value of all clocks. Again, for simplicity, we assume that the set of clocks has a particular order and their current values follow the same order in a vector.

Definition 4.2. Given an IOSA$_u$ $\mathcal{I} = (\mathcal{S}, \mathcal{A}, \mathcal{C}, \rightarrow, C_0, s_0)$ with $\mathcal{C} = \{x_1, \ldots, x_N\}$, its semantic is defined by the NLMP $\mathcal{P}(\mathcal{I}) = (\mathbf{S}, \mathscr{B}(\mathbf{S}), \{\mathcal{T}_a \mid a \in \mathcal{L}\})$ where

- $\mathbf{S} = (\mathcal{S} \cup \{\mathsf{init}\}) \times \mathbb{R}^N$, $\mathcal{L} = \mathcal{A} \cup \mathbb{R}_{>0} \cup \{\mathsf{init}\}$, with $\mathsf{init} \notin \mathcal{S} \cup \mathcal{A} \cup \mathbb{R}_{>0}$
- $\mathcal{T}_{\mathsf{init}}(\mathsf{init}, \vec{v}) = \{\delta_{s_0} \times \prod_{i=1}^{N} \mu_{x_i}\}$,
- $\mathcal{T}_a(s, \vec{v}) = \{\mu^{\vec{v}}_{C',s'} \mid s \xrightarrow{C,a,C'} s', \bigwedge_{x_i \in C} \vec{v}(i) \leq 0\}$, for all $a \in \mathcal{A}$, where $\mu^{\vec{v}}_{C',s'} = \delta_{s'} \times \prod_{i=1}^{N} \overline{\mu}_{x_i}$ with $\overline{\mu}_{x_i} = \mu_{x_i}$ if $x_i \in C'$ and $\overline{\mu}_{x_i} = \delta_{\vec{v}(i)}$ otherwise, and
- $\mathcal{T}_d(s, \vec{v}) = \{\delta_s \times \prod_{i=1}^{N} \delta_{\vec{v}(i)-d}\}$ if there is no urgent $b \in \mathcal{A}^{\mathsf{o}} \cap \mathcal{A}^{\mathsf{u}}$ for which $s \xrightarrow{\_,b,\_} \_$ and $0 < d \leq \min\{\vec{v}(i) \mid \exists a \in \mathcal{A}^{\mathsf{o}}, C' \subseteq \mathcal{C}, s' \in S : s \xrightarrow{\{x_i\},a,C'} s'\}$, and $\mathcal{T}_d(s, \vec{v}) = \varnothing$ otherwise, for all $d \in \mathbb{R}_{\geq 0}$.

The state space is the product space of the states of the IOSA$_u$ with all possible clock valuations. A distinguished initial state init is added to encode the random initialization of all clocks (it would be sufficient to initialize clocks in C_0 but we decided for this simplification). Such encoding is done by transition $\mathcal{T}_{\mathsf{init}}$. The state space is structured with the usual Borel σ-algebra. The discrete step is encoded by $\mathcal{T}_a$, with $a \in \mathcal{A}$. Notice that, at state $(s, \vec{v})$, the transition $s \xrightarrow{C,a,C'} s'$ will only take place if $\bigwedge_{x_i \in C} \vec{v}(i) \leq 0$, that is, if the current values of all clocks in C are not positive. For the particular case of the input or urgent actions this will always be true. The next actual state would be determined randomly as follows: the symbolic state will be s' (this corresponds to $\delta_{s'}$ in $\mu^{\vec{v}}_{C',s'} = \delta_{s'} \times \prod_{i=1}^{N} \overline{\mu}_{x_i}$), any clock not in C' preserves the current value (hence $\overline{\mu}_{x_i} = \delta_{\vec{v}(i)}$ if $x_i \notin C'$), and any clock in C' is set randomly according to its respective associated distribution (hence $\overline{\mu}_{x_i} = \mu_{x_i}$ if $x_i \in C'$). The time step is encoded by $\mathcal{T}_d(s, \vec{v})$ with $d \in \mathbb{R}_{\geq 0}$. It can only take place at d units of time if there is no output transition enabled at the current state within the next d time units (this is verified by condition $0 < d \leq \min\{\vec{v}(i) \mid \exists a \in \mathcal{A}^{\mathsf{o}}, C' \subseteq \mathcal{C}, s' \in S : s \xrightarrow{\{x_i\},a,C'} s'\}$). In this case, the system remains in the same symbolic state (this corresponds to δ_s in $\delta^{-d}_{(s,\vec{v})} = \delta_s \times \prod_{i=1}^{N} \delta_{\vec{v}(i)-d}$), and all clock values are decreased by d units of

time (represented by $\delta_{\vec{v}(i)-d}$ in the same formula). Note the difference from the timed transitions semantic of pure IOSA. This is due to the maximal progress assumption, which forces to take urgent transition as soon as they get enabled. We encode this by not allowing to make time transitions in presence of urgent actions, i.e. we check that there is no urgent $b \in \mathcal{A}^{\circ} \cap \mathcal{A}^{\mathsf{u}}$ for which $s \xrightarrow{\_,b,\_} \_$ (in which case $\mathcal{T}_d(s, \vec{v}) = \varnothing$.) Instead, notice the *patient* nature of a state $(s, \vec{v})$ that has no output enabled. That is, $\mathcal{T}_d(s, \vec{v}) = \{\delta_s \times \prod_{i=1}^{N} \delta_{\vec{v}(i)-d}\}$ for all $d > 0$ whenever there is no output action $b \in \mathcal{A}^{\circ}$ such that $s \xrightarrow{\_,b,\_} \_$.

In a similar way to Section 3.4, it is possible to show that $\mathcal{P}(\mathcal{I})$ is indeed a NLMP, i.e. that $\mathcal{T}_a$ maps into measurable sets in $\Delta(\mathscr{B}(\mathbf{S}))$, and that $\mathcal{T}_a$ is a measurable function for every $a \in \mathcal{L}$. The proof follows exactly as for lemmas 3.1 and 3.2

4.3 Parallel Composition

In this section we define parallel composition of IOSA$_u$. In doing so, we need to avoid name clashes on the clocks, so that we ensure that the intended behavior of each component is preserved. Furthermore, we do not allow synchronization between outputs to ensure they are autonomous and locally controlled. Finally, we should only allow synchronizing IOSA$_u$s that agree on urgent actions in order to ensure their immediate occurrence. More precisely we should only allow synchronization of compatible IOSA$_u$s as defined next:

Definition 4.3. Two IOSA$_u$s $\mathcal{I}_1$ and $\mathcal{I}_2$ are *compatible* if they do not share output actions nor clocks, i.e. $\mathcal{A}_1^O \cap \mathcal{A}_2^O = \varnothing$ and $\mathcal{C}_1 \cap \mathcal{C}_2 = \varnothing$ and, moreover, they agree on urgent actions, i.e. $\mathcal{A}_1 \cap \mathcal{A}_2^{\mathsf{u}} = \mathcal{A}_2 \cap \mathcal{A}_1^{\mathsf{u}}$.

Definition 4.4. Given two compatible IOSA$_u$s $\mathcal{I}_1$ and $\mathcal{I}_2$, the parallel composition $\mathcal{I}_1||\mathcal{I}_2$ is a new IOSA$_u$ $(\mathcal{S}_1 \times \mathcal{S}_2, \mathcal{A}, \mathcal{C}, \rightarrow, C_0, s_0^1||s_0^2)$ where

(i) $\mathcal{A}^{\circ} = \mathcal{A}_1^{\circ} \cup \mathcal{A}_2^{\circ}$

(ii) $\mathcal{A}^{\mathsf{i}} = (\mathcal{A}_1^{\mathsf{i}} \cup \mathcal{A}_2^{\mathsf{i}}) \setminus \mathcal{A}^{\circ}$

(iii) $\mathcal{A}^{\mathsf{u}} = \mathcal{A}_1^{\mathsf{u}} \cup \mathcal{A}_2^{\mathsf{u}}$

(iv) $\mathcal{C} = \mathcal{C}_1 \cup \mathcal{C}_2$

(v) $C_0 = C_0^1 \cup C_0^2$

and $\rightarrow$ is defined by the same rules in Table 3.1 where we write $s||t$ instead of (s, t).

Def 4.4 does not ensures *a priori* that the resulting structure satisfies conditions (a)–(f) in Definition 4.1. This is only guaranteed by the following proposition.

Proposition 4.1. Let $\mathcal{I}_1$ and $\mathcal{I}_2$ be two compatible IOSA$_u$s. Then $\mathcal{I}_1||\mathcal{I}_2$ is indeed an IOSA$_u$.

Proof. The proof of restrictions (a), (b), (d), and (e) follow by straightforward inspection of the rules, considering that $\mathcal{I}_1$ and $\mathcal{I}_2$ also satisfy the respective restriction, and doing some case analysis. Since $\mathcal{I}_1$ and $\mathcal{I}_2$ are compatible, restriction (c) also follows by inspecting the rules taking into account, in addition, that $\mathcal{I}_1$ and $\mathcal{I}_2$ satisfy restriction (e).

To prove (f) we need to take into account that

$$\mathit{enabling}(s_1) \cap \mathit{enabling}(s_2) = \varnothing$$

which is guaranteed by compatibility, and that

$$\mathit{enabling}(s_1||s_2) = \mathit{enabling}(s_1) \cup \mathit{enabling}(s_2)$$

which is guaranteed by input enabling.

We take $\mathit{active}(s_1||s_2) = \mathit{active}_1(s_1) \cup \mathit{active}_2(s_2)$ and prove that it satisfies conditions (i)–(iv) in (f).

(i) $\mathit{active}(s_0^1||s_0^2) = \mathit{active}_1(s_0^1) \cup \mathit{active}_2(s_0^2) \subseteq C_0^1 \cup C_0^2 = C_0$.

(ii) $\mathit{enabling}(s_1||s_2) = \mathit{enabling}(s_1) \cup \mathit{enabling}(s_2) \subseteq \mathit{active}_1(s_1) \cup \mathit{active}_2(s_2) = \mathit{active}(s_1||s_2)$.

(iii) Let $s_1||s_2$ be stable, then s_1 and s_2 are stable as well (guaranteed by input enabledness). Then $\mathit{active}(s_1||s_2) = \mathit{active}_1(s_1) \cup \mathit{active}_2(s_2) = \mathit{enabling}(s_1) \cup \mathit{enabling}(s_2) = \mathit{enabling}(s_1||s_2)$.

(iv) Let $t_1||t_2 \xrightarrow{C,a,C'} s_1||s_2$. We prove by cases according to the rules in Table 3.1

(R1) Let $a \in \mathcal{A}_1 \setminus \mathcal{A}_2$. Then $t_1 \xrightarrow{C,a,C'} s_1$ and $s_2 = t_2$, and we can calculate: $active(s_1||s_2) = active_1(s_1) \cup active_2(s_2) = active_1(s_1) \cup active_2(t_2) \subseteq (active_1(t_1) \setminus C) \cup C' \cup active_2(t_2) = ((active_1(t_1) \cup active_2(t_2)) \setminus C) \cup C' = (active(t_1||t_2) \setminus C) \cup C'$. In particular, the last but one equality follows by compatibility.

(R2) Similar to the previous case if $a \in \mathcal{A}_2 \setminus \mathcal{A}_1$.

(R3) Let $a \in \mathcal{A}_1 \cup \mathcal{A}_2$. Then $t_1 \xrightarrow{C_1,a,C_1'} s_1$ and $t_2 \xrightarrow{C_2,a,C_2'} s_2$, with $C = C_1 \cup C_2$ and $C' = C_1' \cup C_2'$, and we can calculate: $active(s_1||s_2) = active_1(s_1) \cup active_2(s_2) \subseteq ((active_1(t_1) \setminus C_1) \cup C_1') \cup ((active_2(t_2) \setminus C_2) \cup C_2') = ((active_1(t_1) \cup active_2(t_2)) \setminus C_1 \cup C_2) \cup C_1' \cup C_2' = (active(t_1||t_2) \setminus C) \cup C'$. The last but one equality follow by compatibility.

□

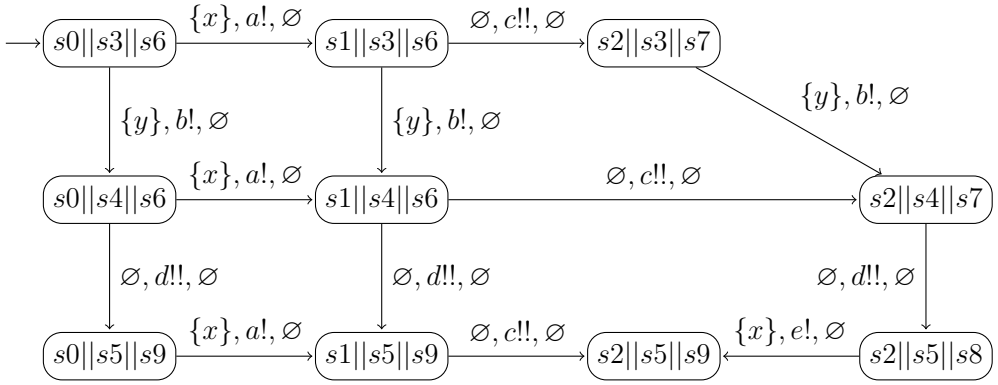


Figure 4.3: Composition $\mathcal{I}_1||\mathcal{I}_2||\mathcal{I}_3$.

Figure 4.3 shows the result of composing components $\mathcal{I}_1$, $\mathcal{I}_2$, and $\mathcal{I}_3$ from example 4.2.

It can be shown that the bisimulation equivalence is a congruence for parallel composition of IOSA$_u$. In fact, we have already shown this for IOSA without urgency at Corollary 3.1, and since the characteristics of urgency do not play any role in the proof of Corollary 3.1 the result immediately extends to this new setting. We formalize this in the following theorem.

Theorem 4.1. Let $\sim$ denote the bisimimulation equivalence relation on NLMPs from definition 2.7 properly lifted to IOSA$_u$, and let $\mathcal{I}_1$, $\mathcal{I}'_1$, $\mathcal{I}_2$, $\mathcal{I}'_2$ be IOSA$_u$s such that $\mathcal{I}_1 \sim \mathcal{I}'_1$ and $\mathcal{I}_2 \sim \mathcal{I}'_2$.Then, $\mathcal{I}_1||\mathcal{I}_2 \sim \mathcal{I}'_1||\mathcal{I}'_2$.

4.4 Confluence

In this section we present a notion of *confluence*, first introduced by Robert Milner in his book Communication and Concurrency [81]. As a notion of behaviour equivalence, confluence will help us to eliminate of certain kinds of non-determinism introduced by urgent actions. We say that such cases of non-determinism are spurious, as they represent situations where any possible decision on which path to follow do not change the stochastic behaviour of the automaton. We call *weakly deterministic* the class of IOSA$_u$s which only present such kind of non-determinism.

As exemplified by $\mathcal{I}_3$ in Figure 4.2, IOSA$_u$s with urgency can be nondeterministic (a priori no indication is given on which action to take at state $s6$). Furthermore, even closed IOSA$_u$s may be non-deterministic. Consider for example the composition of $\mathcal{I}_3$ with $\mathcal{I}_4$ in Figure 4.4 resulting int the composed IOSA$_u$ $\mathcal{I}_5$ at the right of Figure 4.4, Notice that $\mathcal{I}_5$ is closed but nevertheless non-deterministic.

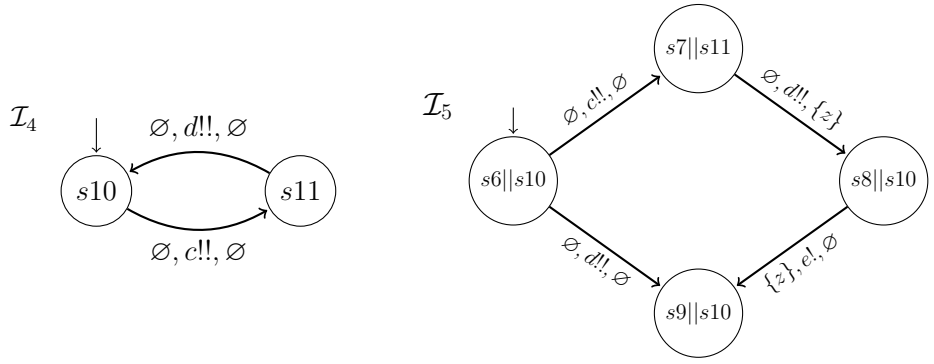


Figure 4.4: $\mathcal{I}_5$ is the result of composition $\mathcal{I}_3||\mathcal{I}_4$.

Using the concept of confluence [81, Chapter 11.3], we proceed to identify an important set of IOSA$_u$s that are weakly-deterministic. In our case we will eventually consider that the urgent actions in a closed IOSA$_u$ are silent, since they do not delay the behaviour of the model. Furthermore, confluent urgent

actions can be interchanged while not modifying the stochastic behaviour of the model. Thus, we will focus on the study of confluence over our urgent actions only.

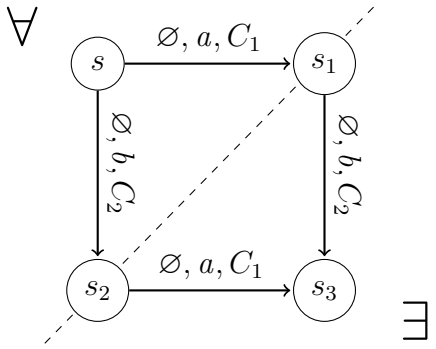


Figure 4.5: Confluence in IOSA.

More precisely, an IOSA$_u$ $\mathcal{I}$ is confluent with respect to urgent actions a and b in $\mathcal{A}^{\mathsf{u}}$ if for every state s in $\mathcal{S}$ we can complete the diagram from Figure 4.5. Note that we are asking to converge in a single state, which is stronger than Milner's strong confluence, where convergence takes place on bisimilar but potentially distinct states. Formally:

Definition 4.5. An IOSA$_u$ $\mathcal{I}$ is confluent with respect to actions $a, b \in \mathcal{A}^{\mathsf{u}}$, if for every state $s \in \mathcal{S}$ it satisfies that:

$$s \xrightarrow{\varnothing,a,C_1} s_1 \wedge s \xrightarrow{\varnothing,b,C_2} s_2 \implies \exists s_3 \in \mathcal{S} \cdot s_1 \xrightarrow{\varnothing,b,C_2} s_3 \wedge s_2 \xrightarrow{\varnothing,a,C_1} s_3$$

$\mathcal{I}$ is *confluent* if it is confluent with respect to every pair of urgent actions.

Confluence is preserved by parallel composition as demonstrated in the following proposition.

Proposition 4.2. If both $\mathcal{I}_1$ and $\mathcal{I}_2$ are confluent w.r.t. urgent actions a and b, then so is $\mathcal{I}_1||\mathcal{I}_2$. Therefore, if $\mathcal{I}_1$ and $\mathcal{I}_2$ are confluent, $\mathcal{I}_1||\mathcal{I}_2$ is also confluent.

Proof. Let $s_1||s_2$ in $\mathcal{S}_{\mathcal{I}_1||\mathcal{I}_2}$, such that $s_1||s_2 \xrightarrow{\varnothing,a,C'} s_1'||s_2'$ and $s_1||s_2 \xrightarrow{\varnothing,b,C''} s_1''||s_2''$ with $a, b \in \mathcal{A}_{\mathcal{I}_1||\mathcal{I}_2}$. We proceed by case analysis on each possible combinations of the rules in Table 3.1 that originates the transitions. We prove the case in which $s_1||s_2 \xrightarrow{\varnothing,a,C'} s_1'||s_2'$ is produced by rule (R1), hence

$a \in \mathcal{A}_{\mathcal{I}_1} \setminus \mathcal{A}_{\mathcal{I}_2}$. The rest proceeds in a similar way. Then $s_2' = s_2$ and $s_1 \xrightarrow{\varnothing,a,C'} s_1'$. We have then three sub-cases given the nature of b:

- If $b \in \mathcal{A}_{\mathcal{I}_1} \setminus \mathcal{A}_{\mathcal{I}_2}$, rule (R1) applies and hence $s_2'' = s_2$ and $s_1 \xrightarrow{\varnothing,b,C'} s_1''$. Since $\mathcal{I}_1$ is confluent, there exists s_1''' such that $s_1' \xrightarrow{\varnothing,a,C'} s_1'''$ and $s_1'' \xrightarrow{\varnothing,b,C''} s_1'''$. Using (R1) in both cases, $s_1'||s_2 \xrightarrow{\varnothing,a,C'} s_1'''||s_2$ and $s_1''||s_2 \xrightarrow{\varnothing,b,C''} s_1'''||s_2$, which proves this case.

- If $b \in \mathcal{A}_{\mathcal{I}_2} \setminus \mathcal{A}_{\mathcal{I}_1}$, (R2) applies and hence $s_1 = s_1''$ and $s_2 \xrightarrow{\varnothing,b,C''} s_2''$. By (R1), $s_1||s_2'' \xrightarrow{\varnothing,a,C'} s_1'||s_2''$, and by (R2), $s_1'||s_2 \xrightarrow{\varnothing,b,C''} s_1'||s_2''$ which proves this case.

- If $b \in \mathcal{A}_{\mathcal{I}_1} \cap \mathcal{A}_{\mathcal{I}_2}$, (R3) applies. Hence there are C_1'' and C_2'' such that $C'' = C_1'' \cup C_2''$, $s_1 \xrightarrow{\varnothing,b,C_1''} s_1''$ and $s_2 \xrightarrow{\varnothing,b,C_2''} s_2''$. Furthermore, since $\mathcal{I}_1$ is confluent, there exists s_1''' such that $s_1' \xrightarrow{\varnothing,b,C_1''} s_1'''$ and $s_1'' \xrightarrow{\varnothing,a,C'} s_1'''$. Then, by (R3), $s_1'||s_2 \xrightarrow{\varnothing,b,C''} s_1'''||s_2''$, and by (R1), $s_1''||s_2'' \xrightarrow{\varnothing,a,C'} s_1'''||s_2''$, which concludes the proof. □

□

Proposition 4.2 implies that no matter how big and complex our system is, if it can be decomposed into confluent components then the model is confluent. Furthermore, composing the model with other confluent components delivers a new confluent model.

Nevertheless this is not the case of non-confluence. In fact, it could happen that several non-confluent components are composed resulting in a confluent IOSA$_u$. Later, following [36], we will provide sufficient conditions on possibly non-confluent components to ensure that the composed IOSA$_u$ is nevertheless confluent.

By looking at the IOSA$_u$ of Fig. 4.6 one can notice that the non-determinism introduced by confluent urgent output actions is spurious in the sense that it does not change the stochastic behaviour of the model. (Here $\tau \in \mathcal{A}^{\mathsf{u}} \cap \mathcal{A}^{\circ}$.) Indeed, since time does not progress, it is the same to sample first clock x and then clock y passing through state s_1, or first y and then x passing through s_2, or even sampling both clocks simultaneously through a transition $s_1 \xrightarrow{\varnothing,\tau,\{x,y\}} s_3$. In any of this cases, the stochastic resolution of the execution

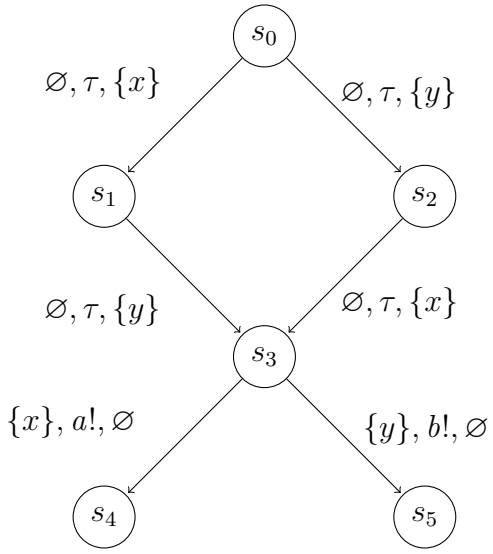


Figure 4.6: Confluence is weakly deterministic

of a or b in the stable state s_3 is the same. This could be generalized to any number of confluent transitions.

In this sense, it will be convenient to use term rewriting techniques to collect all clocks that are active in the convergent stable state and have been activated through a path of urgent actions. The resulting *reduction system* will be then used to prove weak determinism on confluent IOSA$_u$s (Section 4.5). Therefore, we recall some basic notions of rewriting systems.

An *abstract reduction system* [4] is a pair $(\mathcal{E}, \rightarrowtail)$, where the reduction $\rightarrowtail$ is a binary relation over the set $\mathcal{E}$, i.e. $\rightarrowtail \subseteq \mathcal{E} \times \mathcal{E}$. We write $a \rightarrowtail b$ for $(a, b) \in \rightarrowtail$. We also write $a \overset{*}{\rightarrowtail} b$ to denote that there is a path $a_0 \rightarrowtail a_1 \ldots \rightarrowtail a_n$ with $n \geq 0$, $a_0 = a$ and $a_n = b$. An element $a \in \mathcal{E}$ is in *normal form* if there is no b such that $a \rightarrowtail b$. We say that b is a normal form of a if $a \overset{*}{\rightarrowtail} b$ and b is in normal form. A reduction system $(\mathcal{E}, \rightarrowtail)$ is *confluent* if for all $a, b, c \in \mathcal{E}$ $a \overset{*}{\leftarrowtail} c \overset{*}{\rightarrowtail} b$ implies $a \overset{*}{\rightarrowtail} d \overset{*}{\leftarrowtail} b$ for some $d \in \mathcal{E}$. This notion of confluence is implied by the following statement: for all $a, b, c \in \mathcal{E}$, $a \leftarrowtail c \rightarrowtail b$ implies that either $a \rightarrowtail d \leftarrowtail b$ for some $d \in \mathcal{E}$, or $a = b$. A reduction system is *normalizing* if every element has a normal form, and it is *terminating* if there is no infinite chain $a_0 \rightarrowtail a_1 \rightarrowtail \cdots$. A terminating reduction system is also normalizing. In a confluent reduction system every element has at most one normal form. If in addition it is also normalizing,

then such normal form always exists and is unique.

We now define the abstract reduction system introduced by the urgent transitions of an IOSA$_u$. The idea is to introduce a reduction in the reduction system each time we find an urgent transition in the IOSA$_u$ model, and let stable states be normal form. In the reduction we accumulate all reseted clocks and the number of steps taken to get there.

Definition 4.6. Given an IOSA$_u$ $\mathcal{I} = (\mathcal{S}, \mathcal{A}, \mathcal{C}, \rightarrow_{\mathcal{I}}, C_0, s_0)$, define the abstract reduction system $\mathcal{U}_I$ as $(\mathcal{S} \times \mathcal{P}(\mathcal{C}) \times \mathbb{N}_0, \rightarrowtail)$ where $(s, C, n) \rightarrowtail (s', C \cup C', n+1)$ if and only if there exists $a \in \mathcal{A}^{\mathsf{u}}$ such that $s \xrightarrow{\varnothing, a, C'} s'$.

An IOSA$_u$ is *non-Zeno* if there is no loop of urgent actions. This concept is actually related to Zenoness in timed automata, where infinite action take place in a finite amount of time (see e.g. [6].) The following result can be straightforwardly proven.

Proposition 4.3. Let the IOSA$_u$ $\mathcal{I}$ be closed and confluent.Then $\mathcal{U}_{\mathcal{I}}$ is confluent, and hence every element has at most one normal form. Moreover, an element (s, C, n) is in normal form iff s is stable in $\mathcal{I}$. If in addition $\mathcal{I}$ is non-Zeno, $\mathcal{U}_{\mathcal{I}}$ is also terminating and hence every element has a unique normal form.

The following is the interesting cut of the abstract reduction system for the IOSA$_u$ of Figure 4.6.

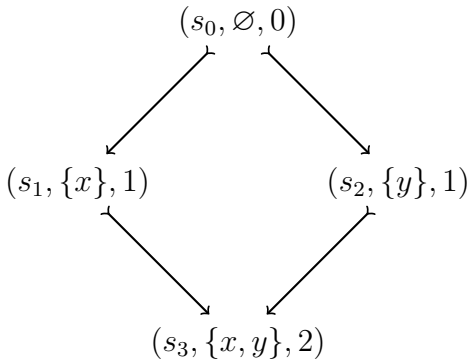


Figure 4.7: Abstract reduction system of Fig. 4.6

The abstract reduction system of an IOSA$_u$ will become useful in the following section, where we prove that closed confluent IOSA$_u$s are weakly deterministic.

4.5 Weak Determinism

We call an IOSA$_u$ closed when all its synchronizations have been resolved through composition and no input actions remain, i.e. $\mathcal{A}^{\mathsf{i}} = \varnothing$ (see Section 3.2). In this section we show that closed confluent IOSA$_u$s are deterministic. Notice that in the more general case, closed IOSA$_u$s may not be deterministic as exemplified by the IOSA$_u$ $\mathcal{I}_3$ in Figure 4.2.

A deterministic IOSA$_u$ is amenable for discrete event simulation or, in case all its clocks are exponentially distributed random variables, also amenable for analysis as a continuous time Markov chain. Actually, we show that closed confluent IOSA$_u$s behave deterministically in the sense that the stochastic behaviour of the model is the same, regardless the way in which nondeterminism is resolved. Thus, we say that an IOSA$_u$ is *weakly deterministic* if

(i) almost surely at most one discrete non-urgent transition is enabled at every time point,

(ii) the choice over enabled urgent transitions does not affect the non urgent-behavior of the model, and

(iii) no non-urgent output and urgent output are enabled simultaneously.

To avoid referring explicitly to time in (i), we say instead that an IOSA$_u$ is weakly deterministic if it almost never reaches a state in which two different non-urgent discrete transitions are enabled. Moreover, to ensure (ii), we define the following weak transition, where the notation $\mathrm{st}(s)$ (read "s is stable") indicate that the state s has no urgent transitions enabled.

Definition 4.7. For a non stable state s, and $v \in \mathbb{R}^N$, we define $(s, \vec{v}) \stackrel{C}{\Longrightarrow}_n \mu$ inductively by the following rules:

$$\frac{s \xrightarrow{\varnothing,a,C} s' \qquad \mathrm{st}(s')}{(s,\vec{v}) \stackrel{C}{\Longrightarrow}_1 \mu^{\vec{v}}_{C,s'}} \tag{T1}$$

$$\frac{s \xrightarrow{\varnothing,a,C'} s' \qquad \forall \vec{v}' \in \mathbb{R}^N : \exists C'', \mu' : (s', \vec{v}') \stackrel{C''}{\Longrightarrow}_n \mu'}{(s,\vec{v}) \stackrel{C' \cup C''}{\Longrightarrow}_{n+1} \hat{\mu}} \tag{T2}$$

where $a \in \mathcal{A}^{\mathsf{u}}$, $\mu^{\vec{v}}_{C,s}$ is defined as in Definition 3.2, and $\hat{\mu} = \int_{\mathcal{S}\times\mathbb{R}^N} f_n^{C''} d\mu^{\vec{v}}_{C',s'}$, with $f_n^{C''}(t,\vec{w}) = \nu$, if $(t,\vec{w}) \stackrel{C''}{\Longrightarrow}_n \nu$, and $f_n^{C''}(t,\vec{w}) = \bar{\varnothing}$ otherwise.

We define the *weak transition* $(s,\vec{v}) \Rightarrow \mu$ if $(s,\vec{v}) \stackrel{C}{\Longrightarrow}_n \mu$ for some $n \geq 1$ and $C \subseteq \mathcal{C}$.

We find here an inductive definition. In this definition, function $f_n^{C''}$ acts as an accumulator for the measures until step $n-1$. A Lebesgue integral defines the final measure, given that $f_n^{C''}$ is measurable. As given above, there is no guarantee that $\stackrel{C}{\Longrightarrow}_n$ is well defined. In particular, there is no guarantee that $f_n^{C''}$ is a well defined measurable function. We postpone this to Lemma 4.1 below.

With this definition, we can introduce the concept of weak determinism:

Definition 4.8. A closed IOSA $\mathcal{I}$ is *weakly deterministic* if $\Rightarrow$ is well defined in $\mathcal{I}$ and, in $P(\mathcal{I})$, any state $(s,v) \in \mathbf{S}$ that satisfies one of the following conditions is almost never reached from any $(\mathsf{init}, v_0) \in \mathbf{S}$:

(a) s is stable and $\cup_{a\in\mathcal{A}\cup\{\mathsf{init}\}}\mathcal{T}_a(s,v)$ contains at least two different probability measures,

(b) s is not stable, $(s,v) \Rightarrow \mu$, $(s,v) \Rightarrow \mu'$ and $\mu \neq \mu'$, or

(c) s is not stable and $(s,v) \xrightarrow{a} \mu$ for some $a \in \mathcal{A}^{\mathsf{o}} \setminus \mathcal{A}^{\mathsf{u}}$.

By "almost never" we mean that the measure of the set of all paths leading to any measurable set in $\mathscr{B}(\mathbf{S})$ containing only states satisfying (a), (b), or (c) is zero. Thus, Definition 4.8 states that, in a weakly deterministic IOSA$_u$, a situation in which a non urgent output action is enabled with another output action, being it urgent (case (c)) or non urgent (case (a)), or in which sequences of urgent transitions lead to different stable situations (case (b)), is almost never reached.

For the previous definition to make sense we need that $\mathcal{P}(\mathcal{I})$ satisfies *time additivity*, *time determinism*, and *maximal progress* [105]. This is stated in the following theorem whose proof is very similar to that of Theorem 3.5.

Theorem 4.2. Let $\mathcal{I}$ be an IOSA$_u$. Its semantics $\mathcal{P}(\mathcal{I})$ satisfies, for all $(s,\vec{v}) \in \mathbf{S}$, $a \in \mathcal{A}^{\mathsf{o}}$ and $d, d' \in \mathbb{R}_{>0}$, the following three items:

(i) $\mathcal{T}_a(s,\vec{v}) \neq \varnothing \;\Rightarrow\; \mathcal{T}_d(s,\vec{v}) = \varnothing$ (maximal progress),

(ii) $\mu, \mu' \in \mathcal{T}_d(s, \vec{v}) \;\Rightarrow\; \mu = \mu'$ (time determinism), and

(iii) $\delta^{-d}_{(s,\vec{v})} \in \mathcal{T}_d(s, \vec{v}) \wedge \delta^{-d'}_{(s,\vec{v}-d)} \in \mathcal{T}_{d'}(s, \vec{v} - d) \;\Leftrightarrow\; \delta^{-(d+d')}_{(s,\vec{v})} \in \mathcal{T}_{d+d'}(s, \vec{v})$ (time additivity).

In the next lemma, we prove that, under the hypothesis that the IOSA is closed and confluent, $\overset{C}{\Longrightarrow}_n$ is well defined. Simultaneously, we prove that $\overset{C}{\Longrightarrow}_n$ is deterministic.

Lemma 4.1. Let $\mathcal{I}$ be a closed and confluent IOSA. Then, for all $n \geq 1$, the following holds:

1. If $(s, \vec{v}) \overset{C}{\Longrightarrow}_n \mu$ then there is a stable state s' such that

   (i) $\mu = \mu^{\vec{v}}_{C,s'}$,

   (ii) $(s, C', m) \overset{*}{\rightarrowtail} (s', C' \cup C, m+n)$ for all $C' \subseteq \mathcal{C}$ and $m \geq 0$, and

   (iii) if $(s, \vec{v}') \overset{C'}{\Longrightarrow}_n \mu'$ then $C' = C$ and moreover, if $\vec{v}' = \vec{v}$, also $\mu' = \mu$; and

2. f^C_n is a measurable function.

Proof. We proceed by induction on n proving first item 1 and using it to prove 2.

So, suppose $n = 1$ and $(s, \vec{v}) \overset{C}{\Longrightarrow}_1 \mu$. By rule (T1) in Definition 4.7, there exists s' stable such that $s \xrightarrow{\varnothing, a, C} s'$ for some $a \in \mathcal{A}^{\mathsf{u}}$ with $\mu = \mu^{\vec{v}}_{C,s'}$, which proves (i). From here and Definition 4.6, $(s, C', m) \overset{*}{\rightarrowtail} (s', C' \cup C, m+1)$, proving (ii). To prove (iii), suppose $(s, \vec{v}') \overset{C'}{\Longrightarrow}_1 \mu'$. By (i) and (ii) applied to this other transition, there exists a stable s'' such that $\mu' = \mu^{\vec{v}'}_{C',s''}$ and $(s, \varnothing, 0) \overset{*}{\rightarrowtail} (s'', C', 1)$. But also $(s, \varnothing, 0) \overset{*}{\rightarrowtail} (s', C, 1)$ as proven before. Since s' and s'' are stable, then, by Prop. 4.3, both $(s', C, 1)$ and $(s'', C', 1)$ are in normal form which must also be unique. Then $s' = s''$ and $C' = C''$. Moreover, if $\vec{v}' = \vec{v}$ then $\mu' = \mu^{\vec{v}'}_{C',s''} = \mu^{\vec{v}}_{C,s'} = \mu$.

To prove item 2 for $n = 1$, notice first that, by (iii), f^C_1 is indeed a function. By (i), $f^C_1(t, \vec{w}) = \mu^{\vec{w}}_{C,t'}$ whenever $(t, \vec{w}) \overset{C}{\Longrightarrow}_1 \mu^{\vec{w}}_{C,t'}$ for some t' stable which is granted to exist, and $f^C_1(t, \vec{w}) = \bar{\varnothing}$ otherwise. To show that f^C_1

is measurable, by [98, Lemma 3.6], it suffices to prove that $(f_1^C)^{-1}(\Delta^q(A \times \prod_{i=1}^N V_i))$ is measurable for all $A \subseteq \mathcal{S}$ and $V_i \in \mathscr{B}(\mathbb{R})$. Notice that

$$
\begin{aligned}
&(f_1^C)^{-1}(\Delta^q(A \times \textstyle\prod_{i=1}^N V_i)) = \\
&= \{(t, \vec{w}) \mid \exists t' : (t, \vec{w}) \overset{C}{\Rightarrow}_1 \mu^{\vec{w}}_{C,t'} \wedge \mu^{\vec{w}}_{C,t'}(A \times \textstyle\prod_{i=1}^N V_i) \geq q\} \\
&= \{(t, \vec{w}) \mid \exists t' \in A : (t, \vec{w}) \overset{C}{\Rightarrow}_1 \mu^{\vec{w}}_{C,t'} \wedge \textstyle\prod\limits_{x_i \in C} \mu_{x_i}(\prod\limits_{x_i \in C} V_i) \geq q \wedge \forall x_i \notin C : \vec{w}(i) \in V_i\}
\end{aligned}
$$

$$
= \bigcup\nolimits_{t' \in A}^{t \in \mathcal{S}} \underbrace{\{(t, \vec{w}) \mid (t, \vec{w}) \overset{C}{\Rightarrow}_1 \mu^{\vec{w}}_{C,t'} \wedge \textstyle\prod\limits_{x_i \in C} \mu_{x_i}(\prod\limits_{x_i \in C} V_i) \geq q \wedge \forall x_i \notin C : \vec{w}(i) \in V_i\}}_{=X_t}
$$

Notice that, if $\prod_{x_i \in C} \mu_{x_i}(\prod_{x_i \in C} V_i) \geq q$, then $X_t = \{t\} \times \prod_{i=1}^N \overline{V}_i$, with $\overline{V}_i = \mathbb{R}$ if $x_i \in C$ and $\overline{V}_i = V_i$ if $x_i \notin C$, and $X_t = \varnothing$ otherwise. In both cases X_t is measurable. Since $\mathcal{S}$ is finite, the union is also finite and hence f_1^C es measurable, which proves the base case.

For the inductive case, let $n \geq 1$ and suppose $(s, \vec{v}) \overset{C}{\Rightarrow}_{n+1} \mu$. By (T2), there are C' and C'' such that $C = C' \cup C''$, $s \xrightarrow{\varnothing, a, C'} s'$, $\forall \vec{v'} \in \mathbb{R}^{\mathbb{N}} : (s', \vec{v'}) \overset{C''}{\Longrightarrow}_n \mu'$, and $\mu = \int_{\mathcal{S} \times \mathbb{R}^N} f_n^{C''} d\mu^{\vec{v}}_{C',s'}$. By induction, C'' is unique (by 1.(iii)), $(s', \vec{v'}) \overset{C''}{\Longrightarrow}_n \mu^{\vec{v'}}_{C'',s''}$ for all $\vec{v'}$ and unique stable state s'' (by 1.(i) and 1.(ii)), and $f_n^{C''}$ is measurable (by 2). Thus $\int_{\mathcal{S} \times \mathbb{R}^N} f_n^{C''} d\mu^{\vec{v}}_{C',s'}$ is well defined. Moreover, notice that $f_n^{C''}(s', \vec{v'}) = \mu^{\vec{v'}}_{C'',s''}$ for all $\vec{v'}$.

We focus on 1.(i) and show that $\mu = \mu^{\vec{v}}_{C' \cup C'',s''}$. First, notice that $\mu = \int_{\{s'\} \times \mathbb{R}^N} f_n^{C''} d\mu^{\vec{v}}_{C',s'} + \int_{(\mathcal{S} \setminus \{s'\}) \times \mathbb{R}^N} f_n^{C''} d\mu^{\vec{v}}_{C',s'}$ and since $\mu^{\vec{v}}_{C',s'} = \delta_{s'} \times \prod_{i=1}^N \overline{\mu}^{v_i}_{x_i}$ with $\overline{\mu}^{v_i}_{x_i} = \mu_{x_i}$ if $x_i \in C'$ and $\overline{\mu}^{v_i}_{x_i} = \delta_{v_i}$ otherwise (we write v_i for $\vec{v}(i)$), then the second summand is the null function $\bar{\varnothing}$. Now, for $A \subseteq \mathcal{S}$ and $Q_i \in \mathbb{R}$, $1 \leq i \leq N$, we calculate

$$
\begin{aligned}
&\mu(A \times Q_1 \times \cdots \times Q_N) = \\
&= \int_{\{s'\} \times \mathbb{R}^N} f_n^{C''}(t, \vec{w})(A \times Q_1 \times \cdots \times Q_N)\ d\mu^{\vec{v}}_{C',s'}(t, \vec{w}) \\
&= \int_{\mathbb{R}^N} f_n^{C''}(s', \vec{w})(A \times Q_1 \times \cdots \times Q_N)\ d(\textstyle\prod_{i=1}^N \overline{\mu}^{v_i}_{x_i})(\vec{w}) \\
&= \int_{\mathbb{R}^N} \mu^{\vec{w}}_{C'',s''}(A \times Q_1 \times \cdots \times Q_N)\ d(\textstyle\prod_{i=1}^N \overline{\mu}^{v_i}_{x_i})(\vec{w}) = (\dagger)
\end{aligned}
$$

By definition, $\mu^{\vec{w}}_{C'',s''} = \delta_{s''} \times \prod_{i=1}^{N} \overline{\mu}^{w_i}_{x_i}$ with $\overline{\mu}^{w_i}_{x_i} = \mu_{x_i}$ if $x_i \in C''$ and $\overline{\mu}^{w_i}_{x_i} = \delta_{v_i}$ otherwise. Then (in the following we omit the domain $\mathbb{R}$ of each integral), using Fubini's theorem, we have:

$$(\dagger) = \int\cdots\int \delta_{s''}(A) \cdot \overline{\mu}^{w_1}_{x_1}(Q_1)\cdots\overline{\mu}^{w_N}_{x_N}(Q_N)\; d\overline{\mu}^{v_1}_{x_1}(w_1)...d\overline{\mu}^{v_N}_{x_N}(w_N)$$
$$= \delta_{s''}(A)\int\cdots\int \overline{\mu}^{w_2}_{x_2}(Q_2)\cdots\overline{\mu}^{w_N}_{x_N}(Q_N)\Big(\underbrace{\int \overline{\mu}^{w_1}_{x_1}(Q_1)\, d\overline{\mu}^{v_1}_{x_1}(w_1)}_{(*)}\Big) d\overline{\mu}^{v_2}_{x_2}(w_2)...d\overline{\mu}^{v_N}_{x_N}(w_N)$$

We focus on $(*)$. Three cases may arise. If $x_1 \in C''$, then

$$(*) = \int \mu_{x_1}(Q_1)\, d\overline{\mu}^{v_1}_{x_1}(w_1) = \mu_{x_1}(Q_1)\int d\overline{\mu}^{v_1}_{x_1}(w_1) = \mu_{x_1}(Q_1)$$

since $\int d\overline{\mu}^{v_1}_{x_1}(w_1) = 1$. If $x_1 \in C' \setminus C''$,

$$(*) = \int \delta_{w_1}(Q_1)\, d\mu_{x_1}(w_1) = \int \chi_{Q_1}(w_1)\, d\mu_{x_1}(w_1) = \mu_{x_1}(Q_1)$$

where χ_{Q_1} is the usual characteristic function. Finally, if $x_1 \notin C \cup C''$,

$$(*) = \int \delta_{w_1}(Q_1)\, d\delta_{v_1}(w_1) = \int \chi_{Q_1}(w_1)\, d\delta_{v_1}(w_1) = \delta_{v_1}(Q_1).$$

Therefore $(*) = \overline{\mu}_{x_1}(Q_1)$ with $\overline{\mu}_{x_1} = \mu_{x_1}$ if $x_1 \in C' \cup C''$ and $\overline{\mu}_{x_1} = \delta_{v_1}$ otherwise. Then, proceeding in the same manner for all the indices, we continue,

$$= \delta_{s''}(A)\overline{\mu}_{x_1}(Q_1)\int\cdots\int \overline{\mu}^{w_2}_{x_2}(Q_2)\cdots\overline{\mu}^{w_N}_{x_N}(Q_N)\, d\overline{\mu}^{v_2}_{x_2}(w_2)...d\overline{\mu}^{v_N}_{x_N}(w_N)$$
$$= \delta_{s''}(A)\cdot\overline{\mu}_{x_1}(Q_1)\cdots\overline{\mu}_{x_N}(Q_N) = (\delta_{s''} \times \textstyle\prod_{i=1}^{N}\overline{\mu}_{x_i})(A \times Q_1 \times \cdots \times Q_N)$$
$$= \mu^{\vec{v}}_{C\cup C'',s''}(A \times Q_1 \times \cdots \times Q_N)$$

which proves 1.(i).

To prove 1.(ii), by Definition 4.6, $(s, C^*, m) \rightarrowtail (s', C^*\cup C', m+1)$ since $s \xrightarrow{\varnothing,a,C'} s'$. By induction, $(s', \vec{v}') \overset{C''}{\Longrightarrow}_n \mu'$ implies $(s', C^*\cup C', m+1) \overset{*}{\rightarrowtail} (s'', C^*\cup C'\cup C'', m+1+n)$. Thus $(s, C^*, m) \overset{*}{\rightarrowtail} (s'', C^*\cup C'\cup C'', m+1+n)$, which proves 1.(ii).

The proofs of 1.(iii) and 2 follows like for the base case.

□

The next corollary follows by item 1 of Lemma 4.1. It states that $\Rightarrow$ is deterministic.

Corollary 4.1. Let $\mathcal{I}$ be a closed and confluent IOSA$_u$. Then, for all $(s, \vec{v})$, if $(s, \vec{v}) \Rightarrow \mu_1$ and $(s, \vec{v}) \Rightarrow \mu_2$, $\mu_1 = \mu_2$.

This corollary already shows that closed and confluent IOSA$_u$s satisfy part (b) of Definition 4.8. In general, we can state:

Theorem 4.3. Every closed confluent IOSA is weakly deterministic.

The rest of the section is devoted to proving this theorem. As we have already showed that closed confluent IOSA$_u$s satisfy (b) of the definition of weak determinism (Definition 4.8), we now focus on points (a) and (c). From now on, we work with the closed confluent IOSA $\mathcal{I} = (\mathcal{S}, \mathcal{C}, \mathcal{A}, \rightarrow, s_0, C_0)$, with $|\mathcal{C}| = N$, and its semantics $\mathcal{P}(\mathcal{I}) = (\mathbf{S}, \mathscr{B}(\mathbf{S}), \{\mathcal{T}_a \mid a \in \mathcal{L}\})$.

The idea of the proof of Theorem 4.3 is to show that the property that all active clocks have non-negative values and they are different from each other is almost surely an invariant of $\mathcal{I}$, and that at most one non-urgent transition is enabled in every state satisfying such invariant. Furthermore, we want to show that, for unstable states, active clocks have strictly positive values, which implies that non-urgent transitions are never enabled in these states. Formally, the invariant is the set

$$\begin{aligned}\mathsf{Inv} = \quad & \{(s, \vec{v}) \mid \mathrm{st}(s) \text{ and } \forall x_i, x_j \in \mathit{active}(s) : i \neq j \Rightarrow \vec{v}(i) \neq \vec{v}(j) \wedge \vec{v}(i) \geq 0\} \\ & \cup \{(s, \vec{v}) \mid \neg\mathrm{st}(s) \text{ and } \forall x_i, x_j \in \mathit{active}(s) : i \neq j \Rightarrow \vec{v}(i) \neq \vec{v}(j)\vec{v}(i) > 0\} \\ & \cup (\{\mathsf{init}\} \times \mathbb{R}^N) \end{aligned} \tag{4.1}$$

with *active* as in Definition 4.1. Note that its complement is:

$$\begin{aligned}\mathsf{Inv}^c = \quad & \{(s, \vec{v}) \mid \exists x_i, x_j \in \mathit{active}(s) : i \neq j \wedge \vec{v}(i) = \vec{v}(j)\} \\ & \cup \{(s, \vec{v}) \mid \mathrm{st}(s) \text{ and } \exists x_i \in \mathit{active}(s) : \vec{v}(i) < 0\} \\ & \cup \{(s, \vec{v}) \mid \neg\mathrm{st}(s) \text{ and } \exists x_i \in \mathit{active}(s) : \vec{v}(i) \leq 0\} \end{aligned} \tag{4.2}$$

It is not difficult to show that Inv^c is measurable and, in consequence, so is Inv. The following lemma states that Inv^c is almost never reached in one step from a state satisfying the invariant.

Lemma 4.2. If $(s, \vec{v}) \in \mathsf{Inv}$, $a \in \mathcal{L}$, and $\mu \in \mathcal{T}_a(s, \vec{v})$, then $\mu(\mathsf{Inv}^c) = 0$.

Proof. We proceed analyzing by cases according a is init, in $\mathcal{A}$, or in $\mathbb{R}_{>0}$.

If a is init, we only consider cases where $s = \mathsf{init}$, since $\mathcal{T}_{\mathsf{init}}(s, v) = \varnothing$ otherwise. If $\mu \in \mathcal{T}_{\mathsf{init}}(\mathsf{init}, v)$, then $\mu = \delta_{s_0} \times \prod_{i=1}^{N} \mu_{x_i}$. Since each μ_{x_i} is a continuous probability measure, the likelihood of two clocks being set to the same value is 0 and $\mu_{x_i}(\mathbb{R}_{>0}) = 1$. Then $\mu(\mathsf{Inv}^c) = 0$. This proves the first case.

For the other cases we introduce the following notation. For each $x_i, x_j \in \mathit{active}(s')$, define $\mathsf{Inv}^c_{ij} = \{(s'', \vec{w}) \mid \vec{w}(i) = \vec{w}(j)\}$ whenever $i \neq j$, $\mathsf{Inv}^c_{i,\mathrm{st}} = \{(s'', \vec{w}) \mid \mathrm{st}(s''), \vec{w}(i) < 0\}$, and $\mathsf{Inv}^c_{i,\mathrm{nst}} = \{(s'', \vec{w}) \mid \neg\mathrm{st}(s''), \vec{w}(i) \leq 0\}$. It is not difficult to prove that each of this type of sets is measurable. Notice that $\mathsf{Inv}^c = \bigcup \mathsf{Inv}^c_{ij} \cup \bigcup \mathsf{Inv}^c_{i,\mathrm{st}} \cup \bigcup \mathsf{Inv}^c_{i,\mathrm{nst}}$ and, since the unions are finite, $\mu(\mathsf{Inv}^c) = 0$ if and only if $\mu(\mathsf{Inv}^c_{ij}) = 0$, $\mu(\mathsf{Inv}^c_{i,\mathrm{st}}) = 0$, and $\mu(\mathsf{Inv}^c_{i,\mathrm{nst}}) = 0$, for every i, j. Thus, for the remaining two cases we focus on proving these last three equalities.

Let $a \in \mathcal{A}$, $\mu \in \mathcal{T}_a(s, \vec{v})$ and $(s, \vec{v}) \in \mathsf{Inv}$. Then $s \neq \mathsf{init}$ and hence, by Definition 3.2, there exists $s \xrightarrow{C,a,C'} s'$ such that $\bigwedge_{x_i \in C} \vec{v}(i) \leq 0$, and $\mu = \delta_{s'} \times \prod_{i=1}^{N} \overline{\mu}_{x_i}$ with $\overline{\mu}_{x_i} = \mu_{x_i}$ if $x_i \in C$, $\overline{\mu}_{x_i} = \delta_{\vec{v}(i)}$ otherwise.

Let $x_i \in \mathit{active}(s')$, then $x_i \in (\mathit{active}(s) \setminus C) \cup C'$. If $x_i \in C'$, then $\mu_{x_i}(\mathbb{R}_{>0}) = 1$ and hence $\mu(\mathsf{Inv}^c_{i,\mathrm{st}}) = \mu(\mathsf{Inv}^c_{i,\mathrm{nst}}) = 0$. If $x_i \in (\mathit{active}(s) \setminus C) \setminus C'$ we consider two subcases: either $C = \varnothing$ or $C = \{x_j\}$. In the first case, $a \in \mathcal{A}^{\mathsf{u}}$ and therefore s is not stable. Then $\vec{v}(i) > 0$ (since $(s, \vec{v}) \in \mathsf{Inv}$) and hence $\delta_{\vec{v}(i)}(\mathbb{R}_{>0}) = 1$, which implies $\mu(\mathsf{Inv}^c_{i,\mathrm{st}}) = \mu(\mathsf{Inv}^c_{i,\mathrm{nst}}) = 0$. If instead $C = \{x_j\}$, $i \neq j$ and, by Definition 3.2, $\vec{v}(j) = 0$. Since s is stable and $(s, \vec{v}) \in \mathsf{Inv}$, then $\vec{v}(i) \geq 0$ and $\vec{v}(i) \neq \vec{v}(j)$, hence $\vec{v}(i) > 0$ and, as before, $\mu(\mathsf{Inv}^c_{i,\mathrm{st}}) = \mu(\mathsf{Inv}^c_{i,\mathrm{nst}}) = 0$.

Suppose now $x_i, x_j \in \mathit{active}(s')$ with $i \neq j$, then $x_i, x_j \in (\mathit{active}(s) \setminus C) \cup C'$. If $x_i \in C$ then μ_{x_i} is a continuous probability measure and hence $\mu(\mathsf{Inv}^c_{ij}) = 0$. Similarly if $x_j \in C$. If instead $x_i, x_j \in \mathit{active}(s) \setminus C$, then $\vec{v}(i) \neq \vec{v}(j)$ because $(s, \vec{v}) \in \mathsf{Inv}$ and hence $\delta_{\vec{v}(i)} \neq \delta_{\vec{v}(j)}$. Therefore $\mu(\mathsf{Inv}^c_{ij}) = 0$. This proves that $\mu(\mathsf{Inv}^c) = 0$ for this case.

Finally, take $d \in \mathbb{R}_{>0}$ and suppose that $\mathcal{T}_d(s, \vec{v}) = \{\mu\}$ with $(s, \vec{v}) \in \mathsf{Inv}$. By Definition 3.2, s needs to be stable, $0 < d \leq \min\{\vec{v}(k) \mid s \xrightarrow{\{x_k\},a,C'} s', a \in \mathcal{A}^{\mathsf{o}}\}$, and $\mu = \delta_s \times \prod_{i=1}^{N} \delta_{\vec{v}(i)-d}$. Since s is stable, $\mu(\mathsf{Inv}^c_{i,\mathrm{nst}}) = 0$. For $x_i \in \mathit{active}(s)$, $\vec{v}(i) - d \geq \min\{\vec{v}(k) \mid s \xrightarrow{\{x_k\},a,C'} s', a \in \mathcal{A}^{O}\} - d \geq 0$, since $\mathit{active}(s) = \mathit{enabling}(s)$ (s is stable). Hence $\delta_{\vec{v}(i)-d}(\mathbb{R}_{\geq 0}) = 1$. Therefore $\mu(\mathsf{Inv}^c_{i,\mathrm{st}}) = 0$. For $x_i, x_j \in \mathit{active}(s)$ with $i \neq j$, $\vec{v}(i) \neq \vec{v}(j)$ because $(s, \vec{v}) \in$

Inv. Hence $\delta_{\vec{v}(i)-d} \neq \delta_{\vec{v}(j)-d}$. So $\mu(\mathsf{Inv}^c_{ij}) = 0$. This proves that $\mu(\mathsf{Inv}^c) = 0$ for this case, and therefore the lemma.

□

From this lemma we have the following corollary

Corollary 4.2. The set Inv^c is almost never reached in $\mathcal{P}(\mathcal{I})$.

The proof of the corollary requires the definitions related to schedulers and measures on paths in NLMPs (see [102, Chap. 7] for a formal definition of scheduler and probability measures on paths in NLMPs.) We omit the proof of the corollary since it eventually boils down to an inductive application of Lemma 4.2.

The next lemma states that any stable state in the invariant Inv has at most one discrete transition enabled. Its proof is the same as that of Lemma 3.4.

Lemma 4.3. For all $(s, \vec{v}) \in \mathsf{Inv}$ with s stable or $s = \mathsf{init}$, the set $\bigcup_{a \in \mathcal{A} \cup \{\mathsf{init}\}} \mathcal{T}_a(s, \vec{v})$ is either a singleton set or the empty set.

Thus we have only left item (c) of Definition 4.8 in order to prove Theorem 4.3. The next lemma states that any unstable state in the invariant Inv can only produce urgent actions.

Lemma 4.4. For every state $(s, \vec{v}) \in \mathsf{Inv}$, if $\neg\mathrm{st}(s)$ and $(s, \vec{v}) \xrightarrow{a} \mu$, then $a \in \mathcal{A}^{\mathsf{u}}$.

Proof. First, recall $\mathcal{I}$ is closed, hence $\mathcal{A}^{\mathsf{i}} = \varnothing$. If $(s, \vec{v}) \in \mathsf{Inv}$ and $\neg\mathrm{st}(s)$ then $\vec{v}_i > 0$ for all $x_i \in \mathit{enabling}(s) \subseteq \mathit{active}(s)$. Therefore, by Definition 3.2, $\mathcal{T}_a(s, \vec{v}) = \varnothing$ if $a \in \mathcal{A}^{\mathsf{o}} \setminus \mathcal{A}^{\mathsf{u}}$. Furthermore, for any $d \in \mathbb{R}_{>0}$, $\mathcal{T}_d(s, \vec{v}) = \varnothing$ since s is not stable and hence $s \xrightarrow{\_, b, \_} \_$ for some $b \in \mathcal{A}^{\mathsf{o}} \cup \mathcal{A}^{\mathsf{u}}$.

□

Finally, Theorem 4.3 is a consequence of Lemma 4.3, Lemma 4.4, Corollary 4.2, and Corollary 4.1.

Proof of Theorem 4.3. We have to show that every measurable set $B \in \mathscr{B}(\mathbf{S})$ of states satisfying conditions (a), (b), or (c) in Definition 4.8 is almost never reached in $\mathcal{P}(\mathcal{I})$. Let $B_{\mathrm{st}} = B \cap ((\{s \mid \mathrm{st}(s)\} \cup \{\mathsf{init}\}) \times \mathbb{R}^N)$ and $B_{\neg\mathrm{st}} = B \cap (\{s \mid \neg\mathrm{st}(s)\} \times \mathbb{R}^N)$. Then $B = B_{\mathrm{st}} \cup B_{\neg\mathrm{st}}$, and B_{st} and $B_{\neg\mathrm{st}}$ are

measurable. Hence B is almost never reached if and only if B_{st} and $B_{\neg\mathrm{st}}$ are almost never reached.

Let $\mathsf{En}_{\geq 2} = \{(s, \vec{v}) \in \mathbf{S} \mid (\mathrm{st}(s) \vee s = \mathsf{init}) \wedge |\bigcup_{a \in \mathcal{A} \cup \{\mathsf{init}\}} \mathcal{T}_a(s, \vec{v})| \geq 2\}$. By Lemma 4.3, $\mathsf{En}_{\geq 2} \subseteq \mathsf{Inv}^c$, and by (a) in Definition 4.8, $B_{\mathrm{st}} \subseteq \mathsf{En}_{\geq 2}$. Then, by Corollary 4.2, B_{st} is almost never reached. In addition, Corollary 4.1, ensures that no $(s, \vec{v}) \in B_{\neg\mathrm{st}}$ satisfies (b). Therefore every $(s, \vec{v}) \in B_{\neg\mathrm{st}}$ satisfies (c). Hence, by Lemma 4.4 $B_{\neg\mathrm{st}} \subseteq \mathsf{Inv}^c$. Then, by Corollary 4.2, $B_{\neg\mathrm{st}}$ is almost never reached, which proves the theorem. □

4.6 Sufficient conditions for weak determinism

So far, we insistently pointed out that we were interested in building models in a compositional fashion, in a way that they result amenable to real simulation, i.e fully stochastic models. Assuming we have reached a closed model by composing only confluent components, we can apply Theorem 4.3 to ensure that the composition is weakly deterministic, since parallel composition preserves confluence Proposition 4.2. Nevertheless, having a non-confluent component does not imply that the composed model is non-confluent as well. Thus it would be interesting to find conditions under which potentially non-confluent components can be composed into a confluent model.

Fig. 4.3 shows an example in which the composed IOSA is weakly deterministic despite that some of its components are not confluent. The potential non-determinism introduced by state $s_2||s_4||s_6$ is never reached since urgent actions at states $s_0||s_4||s_6$ and $s_1||s_3||s_6$ prevent the execution of a non urgent action leading to such state. We say that state $s_2||s_4||s_6$ is not *potentially reachable*. The concept of potentially reachable can be defined as follows.

Definition 4.9. Given an IOSA$_u$ $\mathcal{I}$, a state s is *potentially reachable* if there is a path $s_0 \xrightarrow{\_,a_0,\_} s_1 \ldots, s_{n-1} \xrightarrow{\_,a_{n-1},\_} s_n = s$ from the initial state, with $n \geq 0$, such that for all $0 \leq i < n$, if $s_i \xrightarrow{\_,b,\_} \_$ for some $b \in \mathcal{A}^{\mathsf{u}} \cap \mathcal{A}^{\circ}$ then $a_i \in \mathcal{A}^{\mathsf{u}}$. In such case we call the path *plausible*.

Notice that none of the paths leading to $s_2||s_4||s_6$ in Fig. 4.3 are plausible. Also, notice that an IOSA$_u$ is bisimilar to the same IOSA$_u$ when its set of states is restricted to only potentially reachable states.

Proposition 4.4. Let $\mathcal{I}$ be a close IOSA$_u$ with set of states $\mathcal{S}$ and let $\overline{\mathcal{I}}$ be the same IOSA$_u$ as $\mathcal{I}$ restricted to the set of states $\overline{\mathcal{S}} = \{s \in \mathcal{S} \mid$ is potentially reachable in $\mathcal{I}\}$. Then $\mathcal{I} \sim \overline{\mathcal{I}}$.

It should be clear that both semantics are bisimilar through the identity relation since a $s \xrightarrow{\{x\},a,C} s'$ with s unstable does not introduce any concrete transition. (Recall the IOSA$_u$ is closed so there is no input action on $\mathcal{I}$.)

For a state in a composed IOSA$_u$ to be potentially reachable, necessarily each of the component states has to be potentially reachable in its respective component IOSA$_u$.

Lemma 4.5. If a state $s_1||\cdots||s_n$ is potentially reachable in $\mathcal{I}_1||\cdots||\mathcal{I}_n$ then s_i is potentially reachable in $\mathcal{I}_i$ for all $i = 1, \ldots, N$.

Proof. We only prove it for $\mathcal{I}_1||\mathcal{I}_2$. The generalization to any n follows easily. We prove it by induction on the length of the plausible path σ that leads to $s_1||s_2$. If $|\sigma| = 0$ then $\sigma = s_1^0||s_2^0$, where each s_i^0 is initial in each $\mathcal{I}_i$ and hence potentially reachable. For the inductive case let $\sigma = \sigma' \cdot (s_1'||s_2') \xrightarrow{C,a,C'} (s_1||s_2)$. W.l.o.g. and by contradiction, suppose s_1 is not potentially reachable in $\mathcal{I}_1$. Necessarily, $s_1 \neq s_1'$ since s_1' is potentially reachable by induction ($|\sigma| = |\sigma'| + 1$). Thus $s_1'||s_2' \xrightarrow{C,a,C'} s_1||s_2$ is the result of applying (R1) or (R3). The rest of the proof follows similarly for both cases. So suppose (R3) was applied. Then $s_1' \xrightarrow{C_1,a,C_1'} s_1$ for some $C_1 \subseteq C$ and $C_1' \subseteq C'$. Since s_1 is not potentially reachable but s_1' is, then $a \in \mathcal{A} \setminus \mathcal{A}^{\mathsf{u}}$ and there is a $b \in \mathcal{A}^{\mathsf{u}} \cap \mathcal{A}^{\circ}$ such that $s_1' \xrightarrow{\_,b,\_} \_$. Then $s_1'||s_2' \xrightarrow{\_,b,\_} \_$, either by (R1) or by (R3) (being $\mathcal{I}_2$ input enabled) yielding σ not plausible and hence a contradiction. □

In this section, and following ideas introduced in [36], we build on a theory that allows us to ensure that a closed composed IOSA$_u$ is confluent (and thus weakly deterministic) in a compositional manner, even when its components may not be confluent. Theorem 4.5 provides such sufficient conditions to guarantee that the composed IOSA$_u$ is confluent. Because of Proposition 4.2, it suffices to check whether two urgent actions that are not confluent in a single component are potentially reached. Since potential reachability depends on the composition, the idea is to overapproximate by inspecting the components in order to avoid the state space explosion. The rest of the section builds on concepts that are essential to construct such overapproximation.

We identify the sets of urgent actions that may be enabled at the same time. To do so, we look at the events that may trigger this action to become enabled. There are three types of such events, occurring into the enabling of spontaneously enabled actions, initially enabled actions, and triggered actions. We also formally define enabled sets and show that the proposed events are the only ones capable of enabling this urgent actions.

Let us define the set $\mathrm{uen}(s) = \{a \in \mathcal{A}^{\mathsf{u}} \mid s \xrightarrow{\_,a,\_} \_\}$ as the set of urgent actions enabled in a state s. We say that a set B of output urgent actions is spontaneously enabled by an action b if it becomes enabled just after a non-urgent transition labeled by b.

Definition 4.10. A set $B \subseteq \mathcal{A}^{\mathsf{u}} \cap \mathcal{A}^{\mathsf{o}}$ is *spontaneously enabled* by $b \in \mathcal{A} \setminus \mathcal{A}^{\mathsf{u}}$ in $\mathcal{I}$, if either $B = \varnothing$ or there are potentially reachable states s and s' such that s is stable, $s \xrightarrow{\_,b,\_} s'$, and $B \subseteq \mathrm{uen}(s')$. B is *maximal* if for any B' spontaneously enabled by b in $\mathcal{I}$ such that $B \subseteq B'$, $B = B'$.

A set that is spontaneously enabled in a composed IOSA_u, can be constructed as the union of spontaneously enabled sets in each of the components as stated by the following proposition. Therefore, spontaneously enabled sets in a composed IOSA_u can be overapproximated by unions of spontaneously enabled sets of its components.

Proposition 4.5. Let B be spontaneously enabled by a in $\mathcal{I}_1||\ldots||\mathcal{I}_n$. Then, there are $B_1, \ldots, B_n$ such that each B_i is spontaneously enabled by a in $\mathcal{I}_i$, and $B = \bigcup_{i=1}^{n} B_i$. If in addition B is maximal, there are $B_1, \ldots, B_n$ such that each B_i is a maximal spontaneously enabled by a in $\mathcal{I}_i$, and $B \subseteq \bigcup_{i=1}^{n} B_i$.

Proof. We only prove it for $\mathcal{I}_1||\mathcal{I}_2$. The generalization to any n follows easily. Let $\bar{B}_i = B \cap \mathcal{A}_i$ for $i = 1, 2$ and note that $B = \bar{B}_1 \cup \bar{B}_2$. We show that $\bar{B}_1$ is spontaneously enabled by a in $\mathcal{I}_1$. The case of $\bar{B}_2$ follows similarly. Since B is spontaneously enabled by a in $\mathcal{I}_1||\mathcal{I}_2$, there exist potentially reachable states $s_1||s_2$ and $s_1'||s_2'$, such that $s_1||s_2$ is stable, $s_1||s_2 \xrightarrow{\_,a,\_} s_1'||s_2'$, and $B \subseteq \mathrm{uen}(s_1'||s_2')$. First notice that $\bar{B}_1 \subseteq \mathrm{uen}(s_1)$. Also, suppose $\bar{B}_1 \neq \varnothing$, otherwise $\bar{B}_1$ is spontaneously enabled by a trivially. Consider first the case that $a \in \mathcal{A}_2 \setminus \mathcal{A}_1$. By (R2), $s_1 = s_1'$, but, since there is some $b \in \bar{B}_1$, $s_1 \xrightarrow{\_,b,\_} \_$ and hence $s_1||s_2 \xrightarrow{\_,b,\_} \_$ rendering $s_1||s_2$ unstable, which is a contradiction. So $a \in \mathcal{A}_1$ and $s_1 \xrightarrow{\_,a,\_} s_1'$. By Lemma 4.5, s_1 and s_1' are potentially reachable and, necessarily, s_1 is stable (otherwise $s_1||s_2$ has to be unstable as shown

before). Therefore $\bar{B}_1$ is spontaneously enabled by a in $\mathcal{I}_1$. The second part of the proposition is immediate from the first part. □

Proposition 4.5 allows us to overapproximate the set of spontaneously enabling actions in the composed IOSA$_u$:

$$\{B \mid B \text{ spontaneously enabled by } a \text{ in } \mathcal{I}_1||\ldots||\mathcal{I}_n\} \subseteq \left\{\bigcup_{i=1}^{n} B_i \mid \forall 1 \leq i \leq n \cdot B_i \text{ spontaneously enabled by } a \text{ in } \mathcal{I}_i\right\} \tag{4.3}$$

Spontaneously enabled sets refer to sets of urgent output actions that are enable after some steps of execution. Urgent output actions can also be enabled at the initial state.

Definition 4.11. A set $B \subseteq \mathcal{A}^{\mathsf{u}} \cap \mathcal{A}^{\circ}$ is *initial* in an IOSA$_u$ $\mathcal{I}$ if $B \subseteq \text{uen}(s_0)$, with s_0 being the initial state of $\mathcal{I}$. B is *maximal* if $B = \text{uen}(s_0) \cap \mathcal{A}^{\circ}$.

An initial set of a composed IOSA$_u$ can be constructed as the union of initial sets of its components. In particular the maximal initial set is the union of all the maximal sets of its components. The proof follows directly from the definition of parallel composition taking into consideration that IOSA$_u$s are input enabled.

Proposition 4.6. Let B be initial in $\mathcal{I} = (\mathcal{I}_1||\ldots||\mathcal{I}_n)$. Then, there are $B_1, \ldots, B_2$, with B_i initial of $\mathcal{I}_i$, $1 \leq i \leq n$ and $B = \bigcup_{i=1}^{n} B_i$. Moreover, $\text{uen}(s_0) \cap \mathcal{A}^{\circ}_{\mathcal{I}} = \bigcup_{i=1}^{n} \text{uen}(s_i^0) \cap \mathcal{A}^{\circ}_i$.

Proposition 4.6 allows us to identify the sets of initially enabled actions in the composed IOSA$_u$ by considering the set of initially enabled actions in its components:

$$\{B \mid B \text{ initial in } \mathcal{I}_1||\ldots||\mathcal{I}_n\} = \left\{\bigcup_{i=1}^{n} B_i \mid \forall 1 \leq i \leq n \cdot B_i \text{ initial in } \mathcal{I}_i\right\} \tag{4.4}$$

The same can be deduced for the maximal set of initial actions:

$$\{B \mid B \text{ maximally initial in } \mathcal{I}_1||\ldots||\mathcal{I}_n\} = \left\{\bigcup_{i=1}^{n} B_i \mid \forall 1 \leq i \leq n \cdot B_i \text{ maximally initial in } \mathcal{I}_i\right\}. \tag{4.5}$$

We say that an urgent action triggers an urgent output action if the first one enables the occurrence of the second one which was not enabled before. With this, we identify not only urgent output actions that are enabled after a non-urgent transition (spontaneous sets) but also those that become enabled after an urgent transition.

Definition 4.12. Let $a \in \mathcal{A}^{\mathsf{u}}$ and $b \in \mathcal{A}^{\mathsf{u}} \cap \mathcal{A}^{\circ}$. *$a$ triggers b* in an IOSA$_u$ $\mathcal{I}$ if there are potentially reachable states s_1, s_2, and s_3 such that $s_1 \xrightarrow{-,a,-} s_2 \xrightarrow{-,b,-} s_3$ and, if $a \neq b$, $b \notin \mathrm{uen}(s_1)$.

Notice that, for the particular case in which $a = b$, $b \notin \mathrm{uen}(s_1)$ is not required, since it would be a contradiction. The following proposition states that if one action triggers another one in a composed IOSA$_u$, then the same triggering occurs in a particular component.

Proposition 4.7. Let $a \in \mathcal{A}^{\mathsf{u}}$ and $b \in \mathcal{A}^{\mathsf{u}} \cap \mathcal{A}^{\circ}$ such that a triggers b in $\mathcal{I}_1||\ldots||\mathcal{I}_n$.Then there is a component $\mathcal{I}_i$ such that $b \in \mathcal{A}_i^{\circ}$ and a triggers b in $\mathcal{I}_i$.

Proof. We only prove it for $\mathcal{I}_1||\mathcal{I}_2$. The generalization to any n follows easily. Because $b \in \mathcal{A}^{\mathsf{u}} \cap \mathcal{A}^{\circ}$ necessarily $b \in \mathcal{A}_1^{\circ}$ or $b \in \mathcal{A}_2^{\circ}$. W.l.o.g. suppose $b \in \mathcal{A}_1^{\circ}$. Since a triggers b in $\mathcal{I}_1||\mathcal{I}_1$, $s_1||s_2 \xrightarrow{-,a,-} s_1'||s_2' \xrightarrow{-,b,-} s_1''||s_2''$ with $s_1||s_2$, $s_1'||s_2'$, and $s_1''||s_2''$ being potentially reachable.

Suppose first that $a \neq b$. Then $b \notin \mathrm{uen}(s_1||s_2)$. Recall that, by Lemma 4.5, s_1, s_1', and s_1'' are potentially reachable in $\mathcal{I}_1$. Since $b \in \mathcal{A}_1^{\circ}$, $s_1' \xrightarrow{-,b,-} s_1''$. Suppose $a \in \mathcal{A}_2 \setminus \mathcal{A}_1$. Then, necessarily, $s_1 = s_1'$ which gives $b \in \mathrm{uen}(s_1) \cap \mathcal{A}^{\circ} \subseteq \mathrm{uen}(s_1||s_2)$, yielding a contradiction. Thus, necessarily $a \in \mathcal{A}_1^{\mathsf{u}}$ and hence $s_1 \xrightarrow{-,a,-} s_1'$, by the definition of parallel composition. It remains to show that $b \notin \mathrm{uen}(s_1)$, but this is immediate since $\mathrm{uen}(s_1) \cap \mathcal{A}^{\circ} \subseteq \mathrm{uen}(s_1||s_2)$ and $b \notin \mathrm{uen}(s_1||s_2)$. Thus a triggers b in $\mathcal{I}_1$ in this case. If instead $a = b$, by the definition of parallel composition we immediately have that $s_1 \xrightarrow{-,b,-} s_1' \xrightarrow{-,b,-} s_1''$, proving thus the proposition. □

Proposition 4.7 tells us that the triggering relation of a composed IOSA$_u$ can be overapproximated by the union of the triggering relations of its components. Thus we define:

Definition 4.13. The *approximate triggering relation* of $\mathcal{I}_1||\ldots||\mathcal{I}_n$ is defined by $\rightsquigarrow = \bigcup_{i=1}^{n}\{(a,b) \mid a \text{ triggers } b \text{ in } \mathcal{I}_i\}$. Its reflexive transitive closure $\rightsquigarrow^*$ is called *approximate indirect triggering relation.*

We have gone through the three ways of enabling urgent output actions. These are by spontaneous sets, by initial sets, by triggered sets. We will now formally define the enabled sets and prove that these are indeed the only three ways of generating them. The next definition characterizes all sets of urgent outputs actions that are simultaneously enable in any potentially reachable state of a given IOSA_u.

Definition 4.14. A set $B \subseteq \mathcal{A}^{\mathsf{u}} \cap \mathcal{A}^{\circ}$ is an *enabled set* in an IOSA_u $\mathcal{I}$ if there is a potentially reachable state s such that $B \subseteq \text{uen}(s)$.If $a \in B$, we say that a is *enabled* in s. Let $\mathsf{ES}_{\mathcal{I}}$ be the set of all enable sets in $\mathcal{I}$.

If an urgent output action is enabled in a potentially reachable state of IOSA_u, then it is is either initial, spontaneously enabled, or triggered by some action, as proved by the following Theorem.

Theorem 4.4. Let $b \in \mathcal{A}^{\mathsf{u}} \cap \mathcal{A}^{\circ}$ be enabled in some potentially reachable state of the IOSA_u $\mathcal{I}$. Then there is a set B with $b \in B$ that is either initial, or spontaneously enabled by some action $a \in \mathcal{A} \setminus \mathcal{A}^{\mathsf{u}}$, or b is triggered by some action $a \in \mathcal{A}^{\mathsf{u}}$.

Proof. Let s be potentially reachable in $\mathcal{I}$ such that $b \in \text{uen}(s) \cap \mathcal{A}^{\circ}$. We prove the theorem for b by induction on the plausible path σ leading to s. If $|\sigma| = 0$, then $\sigma = s$ and s is the initial state. Then the set $\text{uen}(s) \cap \mathcal{A}^{\circ}$ is initial and we are done in this case. If $|\sigma| > 0$, then $\sigma = \sigma' \cdot (s' \xrightarrow{\neg,a,\neg} s)$ for some s', a, and plausible σ'. If $a \in \mathcal{A} \setminus \mathcal{A}^{\mathsf{u}}$ then s' is stable (since σ is plausible) and thus $\text{uen}(s) \cap \mathcal{A}^{\circ}$ is spontaneously enabled by a. If instead $a \in \mathcal{A}^{\mathsf{u}}$, we get two possibilities. If $b \notin \text{uen}(s')$, then b is triggered by a. If $b \in \text{uen}(s')$, the conditions are satisfied by induction since $|\sigma'| = |\sigma| - 1$. □

We now present a way to construct the enabled sets of a closed IOSA_u by analyzing its components. In fact we overapproximate this sets by looking at its components instead of looking at the composed model, thus avoiding the state explosion problem. For this we make use of Definition 4.13, along with overapproximation studied in Propositions 4.5 and 4.6. The next definition is auxiliary to prove the main theorem of this section. It constructs a graph from a closed and composed IOSA_u whose vertices are sets of urgent output actions. It has the property that, if there is a path from one vertex to another, all actions in the second vertex are approximately indirectly triggered by actions in the first vertex (Lemma 4.7). This allows to show that any set

of simultaneously enabled urgent output actions is approximately indirectly triggered by initial actions or spontaneously enabled set (Lemma 4.8).

Definition 4.15. Let $\mathcal{I} = (\mathcal{I}_1 || \dots || \mathcal{I}_n)$ be a closed IOSA. The *enabled graph* of $\mathcal{I}$ is defined by the labelled graph $\mathsf{EG}_{\mathcal{I}} = (V, E)$, where $V \subseteq 2^{\mathcal{A}^\circ \cap \mathcal{A}^{\mathsf{u}}}$ and $E \subseteq V \times (\mathcal{A}^{\mathsf{u}} \cap \mathcal{A}^\circ) \times V$, with $V = \bigcup_{k \geq 0} V_k$ and $E = \bigcup_{k \geq 0} E_k$, and, for all $k \in \mathbb{N}$, V_k and E_k are inductively defined by

$$
\begin{aligned}
V_0 = {} & \textstyle\bigcup_{a \in \mathcal{A}} \{ \bigcup_{i=1}^{n} B_i \mid \forall 1 \leq i \leq n : \\
& \qquad\qquad B_i \textit{ is spontaneously enabled by } a \textit{ and maximal in } \mathcal{I}_i \} \\
& \cup \{ \textstyle\bigcup_{i=1}^{n} \mathrm{uen}(s_i^0) \cap \mathcal{A}_i^\circ \mid \forall 1 \leq i \leq n : s_i^0 \textit{ is the initial state in } \mathcal{I}_i \} \\
E_k = {} & \{ (v, a, (v \backslash \{a\}) \cup \{b \mid a \rightsquigarrow b\}) \mid v \in V_k, a \in v \} \\
V_{k+1} = {} & \{ v' \mid v \in V_i, (v, v') \in E_k, v' \notin \textstyle\bigcup_{j=0}^{k} V_j \}
\end{aligned}
$$

Notice that V_0 contains the maximal initial set of $\mathcal{I}$ and an overapproximation of all its maximal spontaneously enabled sets. Notice also that, by construction, there is a path from any vertex in V to some vertex in V_0.

The set closure of V in $\mathsf{EG}_{\mathcal{I}}$, defined by

$$\overline{\mathsf{ES}}_{\mathcal{I}} = \{ B \mid B \subseteq v, v \in V \}$$

turns out to be an overapproximation of the actual set $\mathsf{ES}_{\mathcal{I}}$ of all enabled sets in $\mathcal{I}$, as proved by the following lemma.

Lemma 4.6. For any closed IOSA $\mathcal{I} = (\mathcal{I}_1 || \cdots || \mathcal{I}_n)$, $\mathsf{ES}_{\mathcal{I}} \subseteq \overline{\mathsf{ES}}_{\mathcal{I}}$.

Proof. Let $B \in \mathsf{ES}_{\mathcal{I}}$. We proceed by induction on the length of the plausible path σ that leads to the state s such that $B \subseteq \mathrm{uen}(s)$. If $|\sigma| = 0$ then s is the initial state and thus B is initial in $\mathcal{I}$. Thus, by Definition 4.11, Prop. 4.6, and Definition 4.15, $B \subseteq (\mathrm{uen}(s_0) \cap \mathcal{A}_{\mathcal{I}}^\circ) = (\bigcup_{i=1}^{n} \mathrm{uen}(s_i^0) \cap \mathcal{A}_i^\circ) \in V_0 \subseteq \overline{\mathsf{ES}}_{\mathcal{I}}$. As a consequence $B \in \overline{\mathsf{ES}}_{\mathcal{I}}$.

If $|\sigma| > 0$ then $\sigma = \sigma' \cdot (s' \xrightarrow{\_,a,\_} s)$, for some s', a, and plausible σ'. If $a \in \mathcal{A} \setminus \mathcal{A}^{\mathsf{u}}$ then s' is stable (since σ is plausible) and thus B is spontaneously enabled by a. By Prop. 4.5, there are $B_1, \dots, B_n$ such that each B_i is spontaneously enabled by a and maximal in $\mathcal{I}_i$, and $B \subseteq \bigcup_{i=1}^{n} B_i$. Since $\bigcup_{i=1}^{n} B_i \in V_0 \subseteq \overline{\mathsf{ES}}_{\mathcal{I}}$, then $B \in \overline{\mathsf{ES}}_{\mathcal{I}}$. If instead $a \in \mathcal{A}^{\mathsf{u}}$, let $B' = \{a\} \cup (B \cap \mathrm{uen}(s'))$. Notice that $B' \subseteq \mathrm{uen}(s') \cap \mathcal{A}^\circ$. Since s' is the last state on σ' and $|\sigma'| = |\sigma| - 1$, $B' \in \overline{\mathsf{ES}}_{\mathcal{I}}$ by induction. Hence, there is a vertex $v' \in V$ in $\mathsf{EG}_{\mathcal{I}}$ such that $B' \subseteq v$ and, by Def 4.15, $v' \in V_k$ for some $k \geq 0$.

Let $v = (v' \backslash \{a\}) \cup \{b \mid a \rightsquigarrow b\}$, then $(v', a, v) \in E_k$ and hence $v \in V_{k+1}$. We show that $B \subseteq v$. Let $b \in B$. If $b = a$, then $a \in \mathrm{uen}(s) \cap \mathcal{A}^{\mathsf{o}}$ and hence a triggers a in $\mathcal{I}$. By Prop. 4.7, $a \rightsquigarrow a$ which implies $a \in v$. Suppose, instead, that $b \neq a$. If $b \in \mathrm{uen}(s')$, then $b \in B' \backslash \{a\} \subseteq v' \backslash \{a\} \subseteq v$. If $b \notin \mathrm{uen}(s')$, then a triggers b in $\mathcal{I}$, and by Prop. 4.7, $a \rightsquigarrow b$ which implies $b \in v$. This proves $B \subseteq v \in \overline{\mathsf{ES}}_{\mathcal{I}}$ and hence $B \in \overline{\mathsf{ES}}_{\mathcal{I}}$. □

The next lemma states that if there is path from a vertex of $\mathsf{EG}_{\mathcal{I}}$ to another vertex, every action in the second vertex is approximately indirectly triggered by some action in the first vertex.

Lemma 4.7. Let $\mathcal{I}$ be a closed IOSA, let $v, v' \in V$ be vertices of $\mathsf{EG}_{\mathcal{I}}$ and let ρ be a path following E from v to v'. Then for every $b \in v'$ there is an action $a \in v$ such that $a \rightsquigarrow^* b$.

Proof. We proceed by induction in the length of ρ. If $|\rho| = 0$ then $v = v'$ and the lemma holds since $\rightsquigarrow^*$ is reflexive. If $|\rho| > 0$, there is a path ρ', $v'' \in V$, and $c \in \mathcal{A}^{\mathsf{u}} \cap \mathcal{A}^{\mathsf{o}}$ such that $\rho = \rho' \cdot (v'', c, v')$. By induction, for every action $d \in v''$ there is some $a \in v$ such that $a \rightsquigarrow^* d$. Because of the definition of E in Definition 4.15, either $b \in v''$ or $c \rightsquigarrow b$ and $c \in v''$. The first case follows by induction. In the second case, also by induction, $a \rightsquigarrow^* c$ for some $a \in v$ and hence $a \rightsquigarrow^* b$. □

Since the initial vertex of the enabled graph of an IOSA_u consists in initial sets and spontaneously enabled sets, and since by construction each other vertex in the graph has a path to the initial vertex, the last lemma allows us to deduce the following one. The next lemma states that every enabled set B in a composed IOSA_u is either approximately triggered by a set of initial actions of the components of the IOSA_u or by a subset of the union of spontaneously enabled sets in each component where such sets are spontaneously enabled by the same event.

Lemma 4.8. Let $\mathcal{I} = (\mathcal{I}_1 || \ldots || \mathcal{I}_n)$ be a closed IOSA_u and let $\{b_1, \ldots, b_m\} \subseteq \mathcal{A}^{\mathsf{u}} \cap \mathcal{A}^{\mathsf{o}}$ be enabled in $\mathcal{I}$. Then, there are (not necessarily different) $a_1, \ldots, a_m$ such that $a_j \rightsquigarrow^* b_j$, for all $1 \leq j \leq m$, and either (i) $\{a_1, \ldots, a_m\} \subseteq \bigcup_{i=1}^{n} \mathrm{uen}(s_i^0) \cap \mathcal{A}_i^{\mathsf{o}}$, or (ii) there exists $e \in \mathcal{A}$ and (possibly empty) sets $B_1, \ldots, B_n$ spontaneously enabled by e in $\mathcal{I}_1, \ldots, \mathcal{I}_n$ respectively, such that $\{a_1, \ldots, a_m\} \subseteq \bigcup_{i=1}^{n} B_i$.

Proof. Because of Lemma 4.6 there is a vertex v of $\mathsf{EG}_{\mathcal{I}}$ such that $\{b_1, \ldots, b_n\} \subseteq v$. Because of the inductive construction of E and V, there is a path from some $v' \in V_0$ to v in $\mathsf{EG}_{\mathcal{I}}$. From Lemma 4.7, for each $1 \leq j \leq m$, there is an $a_j \in v'$ such that $a_j \rightsquigarrow^* b_j$. Because $v' \in V_0$, then either $v' = \bigcup_{i=1}^{n} \mathrm{uen}(s_i^0) \cap \mathcal{A}_i^{\circ}$ or there is some $e \in \mathcal{A}$ such that $v' = \bigcup_{i=1}^{n} B_i$ with B_i spontaneously enabled by e in $\mathcal{I}_i$ □

Proposition 4.2 and Lemma 4.8 give us the ingredients for the main theorem of this section which provides sufficient conditions to guarantee that a closed composed IOSA$_u$ is confluent or, as stated in the theorem, necessary conditions for the IOSA to be non-confluent.

Theorem 4.5. Let $\mathcal{I} = (\mathcal{I}_1 || \cdots || \mathcal{I}_n)$ be a closed IOSA. If $\mathcal{I}$ potentially reaches a non confluent state then there are actions $a, b \in \mathcal{A}^{\mathsf{u}} \cap \mathcal{A}^{\circ}$ such that some $\mathcal{I}_i$ is not confluent w.r.t. a and b, and there are c and d such that $c \rightsquigarrow^* a$, $d \rightsquigarrow^* b$, and, either

(i) c and d are initial actions in any component, or

(ii) there is $e \in \mathcal{A}$ and (possibly empty) sets $B_1, \ldots, B_n$ spontaneously enabled by e in $\mathcal{I}_1, \ldots, \mathcal{I}_n$ respectively, such that $c, d \in \bigcup_{i=1}^{n} B_i$.

Proof. Suppose $\mathcal{I}$ potentially reaches a non confluent state s. Then there are necessarily $a, b \in \mathrm{uen}(s)$ that shows it and hence $\mathcal{I}$ is not confluent w.r.t. a and b. By Prop. 4.2, there is necessarily a component $\mathcal{I}_i$ that is not confluent w.r.t. a and b. Since $\{a, b\}$ is an enabled set in $\mathcal{I}$, the rest of the theorem follows by Lemma 4.8. □

Because of Prop. 4.4 and Theorem 4.3, if all potentially reachable states in a closed IOSA $\mathcal{I}$ are confluent, then $\mathcal{I}$ is weak deterministic. Thus, if no pair of actions satisfying conditions in Theorem 4.5 are found in $\mathcal{I}$, then $\mathcal{I}$ is weak deterministic.

Notice that the IOSA $\mathcal{I} = \mathcal{I}_1 || \mathcal{I}_2 || \mathcal{I}_3$ of Fig 4.3 (see also Fig. 4.2) is an example that does not meet the conditions of Theorem 4.5, and hence detected as confluent. The only potential non-confluent actions are c and d which are not confluent in state s_6 of $\mathcal{I}_3$. The approximate indirect triggering relation can be calculated to $\rightsquigarrow^* = \{(c, c), (d, d)\}$. Also, $\{c\}$ is spontaneously enabled by a in $\mathcal{I}_1$ and $\{d\}$ is spontaneously enabled by b in $\mathcal{I}_2$. Since both sets are spontaneously enabled by *different* actions and c and d are not initial,

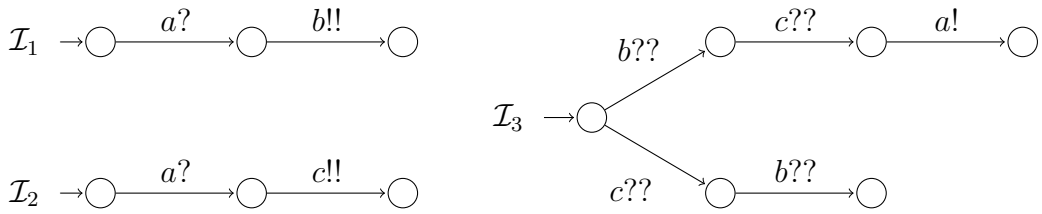


Figure 4.8: $\mathcal{I}_1||\mathcal{I}_2||\mathcal{I}_3$ meets conditions in Theorem 4.5

the set $\{c, d\}$ does not appear in V_0 of $\mathsf{EG}_{\mathcal{I}}$ which would be required to meet the conditions of the theorem.

On the other hand, since conditions in Theorem 4.5 are not sufficient, also confluent IOSAs may satisfy them. Consider the IOSAs in Fig. 4.8. $\mathcal{I}_1||\mathcal{I}_2||\mathcal{I}_3$ is a closed IOSA with a single state and no outgoing transition. Hence, it is confluent. However, $\mathcal{I}_3$ is not confluent w.r.t. b and c, $\rightsquigarrow^* = \{(b,b),(c,c)\}$, $B_1 = \{b\}$ is spontaneously enabled by a in $\mathcal{I}_1$, and $B_2 = \{c\}$ is spontaneously enabled by a in $\mathcal{I}_2$. Hence $b, c \in \bigcup_{i=1}^{n} B_i$, thus meeting the conditions of Theorem 4.5.

Avoiding the state space explosion has its benefits. Indeed,there is an algorithm to check for conditions of Theorem 4.5 in polynomial time and space. Algorithm 1 is a revision of the one for I/O-IMC at [36, Chapter 8.6.1]. It verifies for a given closed model $\mathcal{I} = \mathcal{I}_1||\ldots||\mathcal{I}_n$ the algorithm verifies if it satisfies conditions of Theorem 4.5. It does it in polynomial time and space. For each component it computes the sets of all initial actions $A^{(init)}$, of size $\mathcal{O}(|\mathcal{A}|)$, and the set of nonconfluent actions which has size $\mathcal{O}(|\mathcal{A}|^2)$. Computing all the spontaneous sets will take exponential size to keep. Since we are only interested in knowing if two actions are in a same spontaneous set, we instead compute the relation

$$R^{sp} = \{(a,b) \mid \text{a and b are in a same spontaneous set}\}$$

Notice that R^{sp} has size $\mathcal{O}(|\mathcal{A}|^2)$.

Roughly speaking, to calculate R^{sp} we should compute n depth first search for each pair of actions $(e,a) \in \mathcal{A} \setminus \mathcal{A}^{\mathsf{u}} \times \mathcal{A}^{\mathsf{u}}$, in order to mark each action b which shares with a a same spontaneous set enabled by e. This will take $\mathcal{O}(|\mathcal{A}|^2 \cdot \sum_{i=1}^{n} |S_i|^2)$ time. Computing initial actions, triggering relation, and nonconfluent pairs of actions can be done by applying depth first search to each component which requires $\mathcal{O}(\sum_{i=1}^{n} |\mathcal{S}_i|^2)$ time. The approximate trig-

gering relation for $\mathcal{I}$ is the union of the triggering relations of its components and again has $\mathcal{O}(|\mathcal{A}|^2)$ size. Computing the reflexive, transitive closure of this relation has time complexity $\mathcal{O}(|\mathcal{A}|^3)$ [35].

After building this sets it remains to look for nonconfluent actions a and b, in a set of size $\mathcal{O}(|\mathcal{A}|^2)$. Then we shold look for a pair of actions c and d, that are either initial or contained in a spontaneous set, is also $\mathcal{O}(|\mathcal{A}|^2)$, that approximately indirectly trigger a and b. This gives a time complexity of $\mathcal{O}(|\mathcal{A}|^4)$ for checking conditions in loop 6-13 of the algorithm. Then the overall time complexity is

$$\mathcal{O}(|A|^2 \cdot \sum_{i=1}^{n} |S_i|^2 + |A|^4),$$

while the space complexity is

$$\mathcal{O}(\sum_{i=1}^{n} |S_i|^2 + |A|^2).$$

4.7 Conclusions

In this chapter we have introduced a new version of Input/Output Stochastic Automata, which extends the former IOSAs with the possibility of modeling instantaneous transitions. We called this transitions *urgent*. In contrast to non-urgent transitions, they lack of an enabling clock, and thus their occurrence happens immediately as soon as the enabling state is reached. This results in a much more flexible framework for modelling compositionality in comparison with original IOSAs where many times we experienced the impossibility to modularize a model and instead need to construct a big monolithic model, making a very poor use of compositionality.

We showed how the introduction of urgency turned our models to be non-deterministic even in their closed behavior. Nevertheless, we pointed out that many times this non-determinism is introduced by confluent actions. This turns out to be spurious non-determinism in the sense that it does not change the stochastic behavior of the model. We then defined a notion of weak determinism based on the confluence between urgent actions, and proved that confluent models – those where all urgent actions are confluent

Algorithm 1 Verifies whether a closed IOSA$_u$ $\mathcal{I} = (\mathcal{I}_1 || \ldots || \mathcal{I}_n)$ satisfies the conditions of Theorem 4.5. If the algorithm returns "True" then C may be nondeterministic, otherwise $\mathcal{I}$ is deterministic.

```
1: Compute R^sp.
2: for 1 ≤ i ≤ n do
3:     Compute A_i^init, and the triggering relation for I_i.
4:     Compute all pairs of nonconfluent actions for I_i.
5: Compute approximate triggering relation for C.
6: Compute reflexive, transitive closure of approximate triggering relation.
7: for all nonconfluent pairs of actions a, b do
8:     for all initial actions c that approximately indirectly trigger a do
9:         for all initial actions d do
10:            if d indirectly triggers b then
11:                return True
12:    for all spontaneous actions c that approximately indirectly trigger a
   do
13:        for all actions d in the same spontaneous set as c do
14:            if d indirectly triggers b then
15:                return True
16: return False
```

– are weak deterministic. We showed that if the components are confluent then the composition is confluent.

Based on [36], we obtained sufficient conditions to ensure that a net of possibly non-confluent IOSA$_u$ components is nevertheless confluent. Furthermore, we proposed a methodology that uncovers the triggering conditions of urgent actions, which is able to determine if a net of IOSA$_u$s meets this confluence conditions. Finally we develop an algorithm that follows this methodology in order to determine if the conditions are met. The algorithm runs in polynomial time, by working directly over the components. Notice that if the composition is confluent then we are sure that it is weakly deterministic. Also notice that the algorithm can return *false negatives*, given that the conditions are sufficient but no necessary.

The IOSA$_u$ formalism delivers a framework to model and analyze stochastic systems with general distributions by means of rare event simulation. Its compositional nature, allows to model industrial sized systems in a comfortable and robust manner by concentrating in the clear behaviour of each

component and the intuitive synchronization between them. This contrasts against monolithic approaches where the size and complexity of the models quickly turn the engineering job an error-prone activity. Models with general continuous stochastic behaviour can not be treated by model checking in the general case. Discrete event simulation is the main alternative for analyzing this kind of models. We proved that IOSA_u is weakly deterministic and thus amenable to discrete event simulation.

Chapter 5

Repairable Fault Trees

Fault Tree Analysis (FTA) is a prominent technique for analyzing reliability and safety properties on industrial systems. The appealing characteristics of FTA are based on its apparently easy-to-understand graphical notation. It was first introduced by Bell Laboratories in 1962, as a mean to analyze a ballistic system for the USA army. From then it has been used in many fields specially where risk analysis is very important such as in aviation and space (NASA, SPACE-X, Boeing,etc), automotive industry, pharmaceutical, and other high-hazard industries [49, 73, 97]. It is also a promising technique for the assessment of profit and availability issues in train industry (with maintenance fault trees) [90], for security assessment in cybersecurity (with attack trees), and other. Fault Tree analysis is a top-down technique, where, starting from a top-event (a fault which is of interest to analyze in the system), we go down connecting the possible causes of this fault by means of logical gates until we reach to basic faults (those that we can not, or do not want to decompose anymore) known as basic events (BE). These basic events have a known rate of failure usually described by a probability distribution. The calculation of the probability of failure of the top event depends on the causal relation existing between the top fault and the basic events, which is given by the gates of the tree.

From its origins, Fault Trees have been extended in many different ways, and used for many different purpose. Informally we can say that the original fault trees, called *Standard or Static Fault Tress* (SFT) [59], are DAGs whose leafs are called Basic Events (BE), although in this work we will conveniently call them *Basic Elements*. BEs usually represent the failure of an atomic system component. Each leaf is equipped with a failure rate, that

indicates the frequency at which the component breaks, usually described by an exponential distribution (memoryless). The rest of the nodes on the DAG are called gates, and they model how basic events failures combine to induce more complex system failures. The standard gates AND, OR, and Voting gates represent simple logical combinators. Other varieties of FTs allow for further modeling capabilities by for instance introducing new gates, or a more complex behaviour into the BEs.

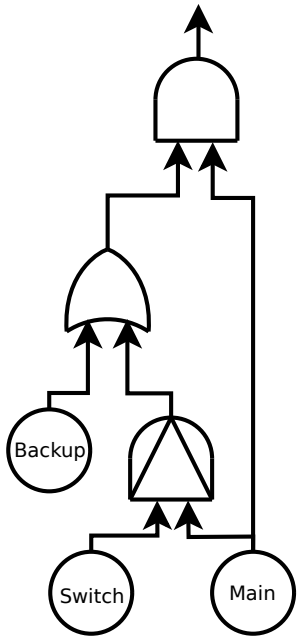


Figure 5.1: Energy backup.

One of the most common extensions are *Dynamic Fault Trees* (DFT) [51, 69], which introduce Priority AND gates to model order in faults occurrence, Spare gates for managing spare parts for broken components, and Functional Dependence gates that model functional dependency between BEs. A typical use example for these new gates can be found in Figure 5.1, where we model an electricity backup system composed from a switch that senses the main line power and turns on the power backup, and a main power supply. We are interested in measuring the likelihood of running out of power (top event). This fault will occur if the switch breaks and then the main power line stops serving electricity, or if both the main and the backup break. In the first case,

the time conditioning between the switch and the main supply is established by the PAND gate. Notice that in a real world example, these BEs would usually be decomposed into more complex subtrees, that would serve with the means of profoundly analyzing the causes of the top event, and hence giving hints on where to look for an improvement on the reliability of the modeled system.

Repairable Fault Trees (RFT) [85, 8] increase FTs expressiveness by introducing the possibility to represent repair events in addition to fail events. Several ways for introducing this modeling capability have been proposed. We will focus on the Repair Box (RBOX) model [12, 85]. A RBOX gate models a repair unit in charge of repairing a certain set of basic elements when they fail, following certain repair policy. Different repair policies such as first come first serve, priority service, random or non-deterministic choice on repair boxes allow users to analyze the impact of taking these decisions in the real system. The introduction of these boxes greatly changes the dynamic of the tree, since the failure probability calculation is not a bottom-up calculation anymore. Moreover, it allows to study the availability in a given model. In this enhanced model, we do not only signal failures but also repairs.

Both qualitative and quantitative analysis can be carried out on Fault Trees. Traditionally, for SFTs, qualitative analysis has been carried out by discovering the minimal cut sets (MCS) [95]. Moreover, many of the existent analysis techniques on SFTs are based on this sets. A cut set is a set of basic events that causes the occurrence of the top event of the tree. A cut set is minimal if it has no proper subset which is also a cut set. In a SFT, if the top event occurs it is because all the BEs in one of its MCS failed. The discovery for instances of a minimal cut set for which the failing probabilities of its BEs is high, might indicate the need to strengthen that part of the system. Although it looks as a simple analysis, it is usually the case that certain MCS could escape to a simple visual inspection of the model, reason why several techniques have been proposed to carry out a more precise discovery of this sets, such as boolean manipulation or BDD techniques [95].

The timing constraints introduced by dynamic gates make MCS not very useful for DFTs. Take for example a model of a single PAND gate with two input BE_1 and BE_2 as in Figure 5.2. If BE_1 and BE_2 fail in that order, then we may think of them as a MCS, although it is not the case that every time they fail the system fails. MCS do not cover all the behavior of the dynamic

gates and are not sufficient to analyze the failing constraints introduced by DFTs. An alternative to MCS are Cut Sequence Sets [75]. Cut sequences work by separating the combinatorial and the timing aspects of dynamic gates in order to extract the MCSs and impose an order inside them afterwards. [69] discusses some important issues related to cut sequences, which may discourage in using them in the general case of DFTs.

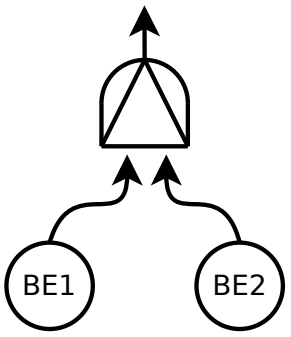


Figure 5.2: Pand gate.

No MCS nor cut sequences are sufficiently interesting in the case of RFT since they do not take into account the repairing process. This is, the probability of failure of a cut set can not be taken straightforward without a state space analysis given the complex inter-dependent repair mechanisms. Thus there is not much significance on obtaining the cut sets. Moreover they are for the same reason not as relevant on quantitative analysis as they are for SFTs.

On the other hand, quantitative analysis may be significant interest in the analysis of RFTs. Typical quantitative analysis measures of interest are *reliability* and *availability*. The reliability of a model is defined as the probability that the system it represents do not fail for a certain amount of time, while the availability of a system is defined as the probability that the system is up and running. In the case of reliability, it is usually the case that we would like to calculate the opposite, i.e. the long run probability of failure of a Fault Tree and we spect it to be as small as possible. While reliability is usually an interesting measure to take on a SFT (which has a single run), availability is usually the interesting measure to take on a RFT where the system may get repaired and thus a long run probability will have a stronger meaning.

The most efficient analysis technique on non-stochastic SFTs consists in

building a Binary Decision Diagram (BDD) representing the same formula as the FT and then solving the required dependability study by using several optimized algorithms. For the case of DFT, new analysis techniques were introduced in order to capture the temporal requirements, such as cut sequences, translation to Markov models [51, 51, 17], Sequence Binary Decision Diagrams [55, 87, 104], algebraic approaches [80, 2], simulation, and combination and optimizations of these methods [11, 58]. The cyclic behaviour introduced by the inter-dependency fault-repair in RFTs disallows most of these techniques, and a state based approach such as discrete event simulation or Model Checking has to be considered.

We find two possible approaches to analyze RFTs. A first approach would be translating the model to a Markov model, maybe applying as much optimizations during the modeling and analysis in order to relieve the state explosion problem. This is the approach followed by many works such as [9, 11, 12]. Two main drawbacks can be pointed out on this approach. The first one is that no matter which existing optimization methods are used, there is no guarantee that there will be a significant state space reduction in general models. This is a specially difficult situation in big and complex industrial size systems analysis involving repair. A second drawback is the restriction to exponentially distributed events, which does not properly capture the correct behavior of the events where timing is governed by other continuous distributions. This is the case for example of phenomena such as timeouts in communication protocols, hard deadlines in real-time systems, human response times or the variability of the delay of sound and video frames (so-called jitter) in modern multi-media communication systems, which are typically described by non-memoryless distributions such as uniform, log-normal, or Weibull distributions [44]. A second approach to RFT analysis would be recurring to simulation, which does not need to construct the full state space of the model, and does not impose *per se* the restriction to any particular kind of probabilistic distributions. The main problem when confronting simulation is the great amount of computation needed to reach a sufficiently accurate result if dealing with very low possible events. This is a most relevant issue when analyzing highly dependable or fault tolerant systems, where the failure probability is very small and plane Monte Carlo simulation becomes infeasible. To face this problem one can make use of Rare Event Simulation techniques such as Importance Splitting or Importance Sampling [101, 26, 27, 88].

In this section we present a formal definition of Repairable Fault Trees, along with its semantic given in terms of Input/Output Stochastic Automata (IOSA). We also analyze determinism on the RFT models, in order to obtain suitable models for discrete event simulation. Many works address the problem of defining a rigorous syntax and semantic to FT, DFT, and RFT, some of them being [34, 12, 18, 8]. They usually differ for instance in the types and meaning of gates, expressiveness power, how spare elements are claimed and how repair races are resolved. Presence of non-deterministic situations is also a main discording issue. Further topics have been addressed in literature, such as compositional aggregation and modular modeling [17]. Good surveys on state of the art on FTs and tools can be found at [92], [69] and [7].

Although broadly used, Fault Trees still have several limitations. In particular, we focus in the following ones:

(a) The expressiveness power of RFTs comes along with a vague formal definition of the behavior of its components. Many works address or at least warn about this issue. It is then a main interest to completely define the behavior of each component, in a way that no ambiguity could harm the modeling and analysis of a model, while attempting, at the same time, to be as permissive as possible, in order to allow modeling as many real systems characteristics as possible. It is in our interest to pay special attention to those underspecification that could lead to non-deterministic behavior which would render the models not amenable for discrete event simulation.

(b) As software and embedded systems get larger and larger, numerical and analytic techniques become unfeasible given the enormous size of the state space to build and explore. Other techniques, with simulation being the most popular, present an alternative that, although resulting on an approximate answer, present a tunable and adequate confidence on the given result. Simulation can not be carried on non-deterministic systems. It is then a principal matter to ensure that the models are deterministic, i.e that at each step of the computation we can determine a unique probabilistic next step. In the case of RFTs, non-determinism is a very spread problem, given the under-specification on the behavior of its components. This under-specification is many times on purpose of allowing liberty to the modeler. Other times, it is an effective mechanism

to analyze different possible solutions to an unknown fact in the modeled system.

(c) The exponential distribution does not capture many of the real life systems probabilistic behavior that FTs intend to model and analyze. Unfortunately most of nowadays DFT analysis tools only allow for exponential distributions, thus restricting only to Markovian models. Moreover, the few cases where they do allow non-Markovian distributions restricts the model to a small range of gates, and do not offer support for the repair model [7]. Then, there is a need to confer a more general semantic to RFT models that allows for arbitrary continuous distributions to be adopted as failure and repair rates.

In Section 5.5 we formally define the syntax for RFTs following ideas from [16]. Moreover, in order to define the compositional deterministic semantic using IOSA_u we discuss different concerns about determinism on RFTs. Our main contribution in this chapter consists in allowing for general probability distributions for failure and repair rates. With this we cover points (a) and (b) from our concerns. This capability is given by our modeling language IOSA_u which at the same time turns to generate weakly deterministic RFTs models, focusing also on point (c). Furthermore, we are able to simulate these deterministic models using the FIG rare event simulation tool, greatly increasing efficiency when analyzing highly dependable systems [31, 28, 89, 101], and if the model is compatible, also in other tools via Jani [30]. Recently [91] covered the matter of using rare event simulation to analyze fault trees, although they restrict to exponential and Erlang distributions. At the moment we are not aware of any other approach applying rare event simulation specifically to fault trees.

5.1 Repairable Fault Trees

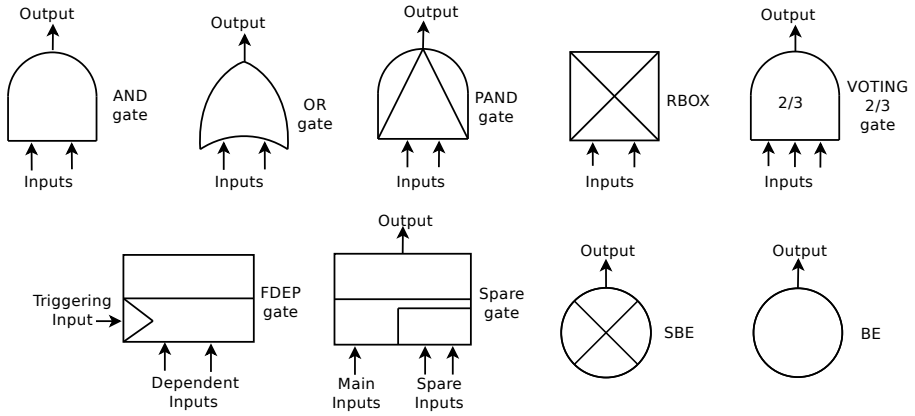


Figure 5.3: RFT elements

In Figure 5.3 we present the graphical representation of each possible gate in a RFTs. Each element has a set of inputs where to connect its subtrees, and an output to propagate the failure, repair and other signals. The propagation of a failure and its subsequent repair starts at the leafs of the fault tree. Only basic elements can be a leaf in a fault tree.

Basic Elements (BE) are the leaves of RFTs. They model atomic units of failure and also the possibility of repair. They are graphically represented by a circle, and accompanied by a failure time distribution, as well as a repair time distribution. They usually describe a basic component of the modeled system, for which times to failure and repair are known. BEs in Repairable Fault Trees will signal a failure event when they fail, as in original fault trees, and in addition, will signal a repair event when they get repaired. When a BE fails, or gets repaired, it instantaneously propagates the event to the gates to which it is connected. We can think that the state of a gate changes based on the signals it receives from its inputs. Usually a fail signal may change the gate state to failing and a repair signal may change it to a working state. At the same time, gates may output a signal when their state changes. Again it will be usually the case that they will output a fail signal when their state changes from working to failing, and a repair signal when it changes the other way round. Other signals may also be produced, as it can be in the case of repair boxes, which may output a "start repairing" signal to any of the BEs

they are in charge of repairing. The intuition about the behavior of each gate is as follows.

An *AND gate* fails whenever all its inputs are failing, and gets repaired when at least one of its inputs gets so. AND gates correspond to a logical conjunction with respect to fail signals.

An *OR gate* fails whenever at least one of its inputs fail and stop failing when all of its inputs get fixed. OR gates correspond to a logical disjunction with respect to fail signals.

A *VOT gate* fails whenever at least k of its n inputs fail and stop failing if, in a state where exactly k of its inputs are failing, one of them gets repaired. This type of gate can be replaced by an appropriate combination of AND and OR gates.

Note that the three gates we presented so far react only based on changes in the combination of their inputs. These are called *static gates* and are already present in Static Fault Trees. In contrast, the following gates are called *dynamic gates* and react to the signals of their inputs taking into account other aspects like timings and dependence.

A *PAND gate* fails whenever all its inputs fail and they do so from left to right, imposing an ordering condition into the failure occurrence. It gets repaired whenever its last input gets repaired. Note this repairing condition. Not much has been written about the repair of a PAND gate. We decided for this possibility based on most used cases like the one on Figure 5.1, where fixing the first input does not represent a fixing of the represented system, while fixing the second input does. If we agree that an n inputs PAND gate can be modeled with a sequence of $n-1$ 2-input PAND gates connected in cascade, then our point is still consistent and reasonable for larger PAND gates. Another possibility we did not follow, but still does make some sense, is to wait until all inputs of the PAND gate get fixed. Finally, we did not find any reasonable meaning on the remaining case, i.e. fixing the PAND gate as soon as the first input (or other than the last one) gets fixed. That is why our choice on the PAND gate repairing behavior.

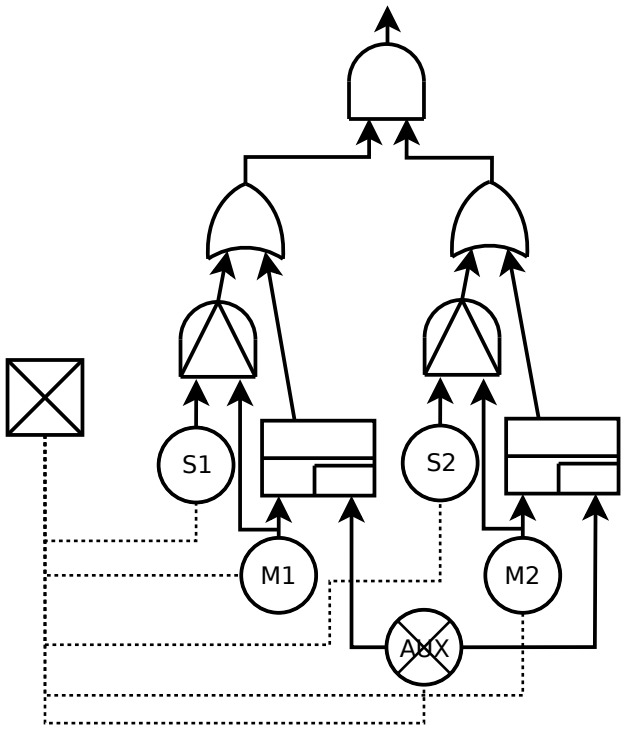
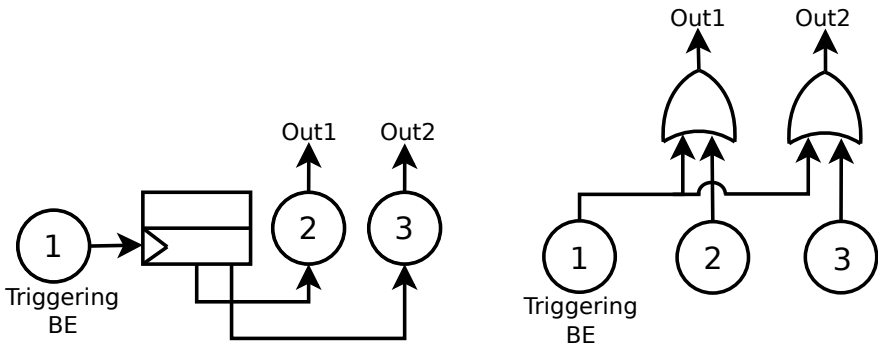


Figure 5.4: A fault tolerant cooling system.

A *Functional dependency gate (FDEP)* has $n+1$ inputs. The fail signal of one of its inputs (the triggering one) makes all the other inputs inaccessible to the rest of the system. Note that the dependent inputs do not fail, and they will be accessible again as soon as the triggering component gets repaired (note the difference with [16, 91] where dependent BEs do fail). This gate is a syntactic sugar for an OR gates system as depicted in Figure 5.5. See Section 5.2 for further discussion on these gates.

A *Spare basic element (SBE)* is a special case of BEs which can be enabled and disabled, and can be used as spare parts for other BEs by means of Spare gates. One SBE can be shared by several Spare gates, and different sharing policies are introduced for this purpose. It is common to distinguish three types of SBEs, depending on whether they do not fail when disabled, named *cold SBE*, or they fail but with a lower rate than when enabled, called *warm SBE*, or they fail with the same frequency as they do when they are enabled, called *hot SBE*.



(a) (b)

Figure 5.5: A FDEP gate system (a) and an equivalent construction (b).

A *Spare gate (SG)* allows to backup a main basic element with several spare basic elements in case the main one fails. Each spare gate has a main input and n spare parts inputs. A main input can only be a BE. The spare inputs can only be SBEs. As soon as the main input fails, the SG selects the next available spare element to replace it. The SG will fail whenever it does not obtain a replacement, and will inform repair whenever the main input gets repaired or a spare input is obtained. If an in-use replacement fails the SG will look for a new one. If the main input is repaired, the SG will free its acquired spare input, in case there is one. An example for the use of spare gates and SBEs can be found in Figure 5.4. It shows a RFT model of a two water pumps (*M1* and *M2*) system with a spare backup pump (AUX) in case any of them fail. If both subsystems fail then the whole system does (note the AND gate at the top). Both subsystems share a spare pump which is connected to their corresponding spare gates. When a main pump breaks, the corresponding spare gate asks for the Auxiliary pump. If the spare pump is not broken and available, then the spare gate enables it in order to start working. Further description of the model and its behavior can be found in Section 5.8.2.

A *Repair Boxs (RBOX)* is the unit in charge of managing the repair of failed BEs and SBEs. It has n inputs, corresponding to each of the basic elements it should repair, and a dummy output. A RBOX policy determines in which order the failed BEs (or SBEs) will be repaired. The time that takes repairing each input is defined at the at the BE (or SBE) itself. We think that this is a better approach than defining a single repair time at the repair box,

since the nature of the inputs vary, and may influence its repair complexity. Notice that simply changing policies on the Repair Boxes already opens the possibility to compare different scenarios that much certainly would influence on the reliability of the modeled system.

5.2 Discussion on design.

In this section, we would like to discuss some alternatives to what we have decided on the design of the elements of a Repairable Fault Tree. We intend to clarify and justify our choices on the semantics of the different gates.

It is noticeable that on Standard Fault Trees no reasoning about the repair of a model component was consider. In effect, only the propagation and logical combination of failures were considered. In fact, the basic elements in SFT are the failures of the system components which then propagate and combined, building more complex failures until reaching the top event. In this context, the leaves of the trees represent the whole information we need about these atomic components, i.e. the failure rates. With the introduction of repair units (RBs) and the possibility of repairing broken components of the system, the whole logic changes. There is first the need to introduce a new kind of event, that is the repair event. Furthermore, now leaves need to represent more information in order to completely model an atomic component. The former meaning of a basic element does not cover these repair events, and for their simplicity are named *Basic Events*. A leaf now has information about both kind of events for a component, its failure event and its repair event, which are comprised by the name *Basic Element* instead.

The change of dynamic also attains the different gates of the tree. In some cases like the AND and OR gates, a clear and simple definition of the behaviour under the repair of components is achieved without much difficulty, and without introducing non-determinism which is of or higher interest. This is not the case with the PAND gate for example, as we have seen above, where many questions arise and situations involving non-determinism force a deeper analysis.

Another gate that deserves our attention is the functional dependence gate (FDEP). In former works such as [51] this gate was defined as to disable the access to several components as a consequence of the failure of a triggering component. Note that there was no explicit mention on the failing of the dependent components, the only consequence was not being able to access

them anymore. In the context of a Standard Fault Tree, the failing of the dependent components had the same effect that would have not being able to access them, since there was no notion of repairing after the failure. In our context, on the other hand, these two different situations have a sever impact on the meaning of the gate. We decided to keep the meaning from [51] in contrast to other works like [34, 91] where the failing of the triggering basic element causes the failure of the dependent basic elements. This is, dependent basic elements will not fail as a consequence of the failing of the triggering basic element, but will become inaccessible instead. Choosing an apposite approach would require analyzing the non-determinism caused by the simultaneous failures. Our modeling desition ensures that this gate does not introduce any non-determinism. The work [17] mentions that it is not difficult to generalize the FDEP gate in order to accept other gates or subtrees as dependent inputs. This could also be done in our framework but we have decided not to work on it yet.

The spare gate and spare basic element deserve some discussion as well. While some works allow to extend the possibilities of these constructions in order to allow a general subtree to work as spare part of a failing subtree [18], this would with no doubt require restricting the allowed spare subtrees in order to avoid non-determinism. We leave this possibility for upcoming work.

All in all, many extensions and further support for gates can be analyzed and introduced to our RFT. Nevertheless, as a first step, we intend to give here a tight but robust framework that can be ensured to be deterministic, and leave those possibilities for upcoming work.

5.3 IOSA symbolic language

We present a symbolic language to describe IOSA_u models, which will ease the modelling by avoiding the extensive description of the discrete state space and numerous transitions. Since our framework (IOSA_u) is compositional, this is reflected also in the language, where each component is modeled separately by what we call a *module*. A module is composed of a set of variables, whose valuation represent the actual state of the component, a set of clocks corresponding to the enabling clocks for non urgent transitions, and a set of transitions which symbolically describe the possible jumps between states (changes of valuations and reseting of clocks). Figure 5.6 models a basic element as an example. Variables can be of integer (with finite range) or

boolean type. As we will see later, also arrays can be defined as variables.

```
 1 module BE
 2   fc, rc : clock;
 3   signal : [0..2] init 0;
 4   broken : [0..2] init 0;
 5
 6   [fl!] broken=0 @ fc -> (signal'=1) & (broken'=1);
 7   [r??] broken=1 -> (broken'=2) & (rc'=γ);
 8   [up!] broken=2 @ rc -> (signal'=2) &
 9                          (broken'=0) & (fc'=μ);
10
11   [f!!] signal=1 -> (signal'=0);
12   [u!!] signal=2 -> (signal'=0);
13 endmodule
```

Figure 5.6: Basic Element IOSA symbolic model.

An initial value for each variable is determined after the keyword `init`. Clocks distributions are defined at the transitions where clocks are reset. A transition is described by the name of the action which takes place (or no name if not needed), a condition restricting the origin states, an enabling clock (only for the case of output non-urgent transitions), a condition describing the target states, and the set of clocks to be reset. A fast overview of Figure 5.6 will help to further understand our symbolic language: Two clocks, `fc` and `rc`, are defined at line 2. these clocks will be used as enabling clocks for transitions at lines 6 and 8, and reset on transitions at lines 7 and 8. Lines 3 and 4 define variables `signal` and `broken`, both of integer type ranging between 0 and 2, and initialized with value 0. Line 6 defines a set of output non-urgent transitions, which produce the output action `fl`. More precisely, this line defines the set of non-urgent transitions $s \xrightarrow{\{fc\},fl!,\varnothing} s'$, where s meets the condition `broken=0`, and s' is the result of changing the values in state s of variables `signal` and `broken` to `1`. The `@` symbol precedes the enabling clock for the transition while the $\rightarrow$ symbol distinguishes between conditions for the origin state and the target state. The conditions on the target state are expressed as assignments to the next state values of the variables, indicated with an apostrophe. Line 7 defines an urgent input

transition with label `r`. The double question marks after the name indicate that it describes urgent input transitions. Urgent output transition are indicated with double exclamation marks (`!!`), non urgent input transitions with a single question mark, and non-urgent output transitions with a single exclamation mark or no mark if no label is provided. At the end of the line we find the reset of the clock `rc` to a value sampled with a probability distribution γ. This line then defines transitions $s \xrightarrow{\varnothing, r??, \{rc\}} s'$, where s meets with condition `broken=1`, s' is identical to s except for variable `broken` which has value `2`, and clock `rc` has distribution γ. At line 10, an urgent output transition is defined, indicating the failure of this component. We will usually use these urgent transitions to synchronize and communicate with other modules. As a final note on the matter, we should point out that a model as the one described at Figure 5.6 is not legally a IOSA since it is not input enabled. In fact it is missing a transition:

```
[r??] broken != 1 -> ;
```

Nonetheless, for the sake of simplicity, we will avoid writing down this kind of "self loop" transitions, although we will assume they exist in the model.

The full specification of the IOSA symbolic modeling language can be found in Appendix A. An IOSA model is described by a set of modules, each one describing a concurrent component of the system to model. The body of a module can be clearly divided into three parts: the variables declarations, the clocks declarations, and the transition specification. Arrays are declared along with variables, with the additional requirement of defining the range of the array between brackets. Transition guards are boolean formulas describing the origin states for the symbolic transition. In this case the & symbol stands for the propositional conjunction operator while | stands for the propositional disjunction operator, and ! for the negation. Assignments (or Postconditions) on the other side, describe the changes of the values. Each assignment is enclosed by parenthesis, and the variable's name is followed by an apostrophe to indicate that corresponds to the value of the variable in the reached state after taking the transition. An ampersand ('&') separates each assignment. Notice the similarity with PRISM [71] syntax for describing transitions. Along with the assignment of values to future variables, we find the reset of clocks. A clock is assigned a probability distribution ($clock' = \gamma$) to indicate that it will be set to a value from that probability distribution immediately before reaching the new state. For a clock c, its probability distribution should be always the same, thus it should

coincide in every line where it is set. Each line that defines a variable or array, describes a transitions or declares a clock, ends in semicolon and can be placed wherever inside the scope of a module.

5.4 A formal definition of RFT

In this section we present a formal definition of the RFT along with its semantic given in terms of IOSA_u. We extend the DFT formalization given by [17] with new features as repair boxes as well as some other modifications such as conditions in order to ensure determinism. Each element of a RFT is characterized by a tuple consisting of its type, its arity, i.e. number of inputs, and possibly other parameters such as probability distributions for the fail and repair events on BE.

Definition 5.1. Let n, m and k belong to $\mathbb{N}^+$, and let μ, ν and γ be continuous probability distributions. We define the set $\mathcal{E}$ of elements of a RFT to be composed of the following tuples:

- $(\mathsf{be}, 0, \mu, \gamma)$ and $(\mathsf{sbe}, 0, \mu, \nu, \gamma)$, which represents basic and spare basic element, with no inputs, with an active failure distribution μ, a dormant failure distribution ν, and a repair distribution γ.
- (and, n), (or, n) and (pand, n), which represent AND, OR and PAND gates with n inputs, respectively,
- (vot, n, k), which represent a k from n inputs voting gate,
- (fdep, n), which represents a functional dependency gate, with 1 trigger input and $n-1$ dependent ones. By convention the first input is the triggering one,
- (sg, n), which represents a spare gate with one main input and $n-1$ spare inputs. By convention the first input is the main one.
- (rbox, n), which represents a RBOX element for n BEs (or SBEs).

A RFT is a direct acyclic graph, for which every vertex v is labeled with an element $l(v) \in \mathcal{E}$. An edge from v to w means that the output of v is connected to an input of w. Since the order of the inputs is relevant, we give them in terms of a list $i(w)$ instead of a set. Similarly, $si(v)$ will list all the

spare gates to which a spare basic element v is connected as an input. Let $\mathrm{t}(v)$ indicate the type of v. That is, $\mathrm{t}(v)$ is the first projection of $l(v)$. Let $\#(v)$ indicate the number of inputs of v, that is, it is the second projection $l(v)$.

Definition 5.2. A repair fault tree is a four-tuple $T = (V, i, si, l\ )$, where V is a set of vertices, $l\colon V \to \mathcal{E}$ is a function labeling each vertex with an RFT element, $i\colon V \to V^*$ is a function assigning $\#(v)$ inputs to each element v in V, and $si\colon V \to V^*$ which indicate which spare gates manage each SBE. The set of edges $\mathrm{E} = \{(v, w) \in V^2 \mid \exists j \cdot v = (i(w))[j]\}$ is the set of pairs (v, w) such that v is an input of w. If such an edge exists we will say that v is connected to w and w to v.

For a RFT T to be *well formed*, the following conditions should be met:

(i) The tuple (V, E) is a direct acyclic graph (DAG).

(ii) T has a unique top element, i.e. a unique element whose non dummy output is not connected to another gate. This is, there is a unique vertex $v \in V$ such that for all $w \in V$, $(v, w) \notin E$ and $\mathrm{t}(v) \notin \{\mathsf{fdep}, \mathsf{rbox}\}$.

(iii) An output can not be more than once the input of a same gate. That is, for all $1 \leq j, k \leq |i(w)|$ with $i(w)[j] = i(w)[k]$, we have $j = k$.

(iv) Since FDEP and RBOX outputs are dummy, if $(v, w) \in E$ then $\mathrm{t}(v) \notin \{\mathsf{fdep}, \mathsf{rbox}\}$.

(v) The inputs of a RBOX can only be basic elements. That is if $(v, w) \in E$ and $\mathrm{t}(w) = \mathsf{rbox}$ then either $\mathrm{t}(v) = \mathsf{be}$ or $\mathrm{t}(v) = \mathsf{sbe}$.

(vi) Each (spare) basic element can be connected to at most one RBOX. This is if $(v, w) \in E$ and $(v, w') \in E$ and $\mathrm{t}(w) = \mathrm{t}(w') = \mathsf{rbox}$, then $w = w'$.

(vii) The spare inputs of a spare gate can only be SBEs, while its main input can only be a BE. I.e., if $(v, w) \in E$ and $\mathrm{t}(w) = \mathsf{sg}$ then $\mathrm{t}(i(v)[0]) = \mathsf{be}$ and for $j > 0, \mathrm{t}(i(v)[j]) = \mathsf{sbe}$. Furthermore, a SBE can only be connected to a spare gate or a RBOX, i.e., if $(v, w) \in E$ and $\mathrm{t}(v) = \mathsf{sbe}$ then $\mathrm{t}(w) \in \{\mathsf{sg}, \mathsf{rbox}\}$.

(viii) A spare basic element is an input of a spare gate, if and only if that spare gate is spare input of the spare basic element, i.e. for v and v'

such that $l(v') = (\mathsf{sbe}, 0, \mu, \nu, \gamma)$ and $l(v) = (\mathsf{sg}, n)$, $(v', v) \in E$ if and only if there exists j such that $v = si(v')[j]$.

(ix) A basic element can be connected to at most one spare gate, i.e. if $(v, w) \in E$ and $(v, w') \in E$ with $\mathrm{t}(w) = \mathrm{t}(w') = \mathsf{sg}$ and $\mathrm{t}(v) = \mathsf{be}$ then $w = w'$.

(x) If a basic element is connected to a spare gate then it can not be connected to a FDEP gate, i.e. if $(v, w) \in E$ and $\mathrm{t}(v) = \mathsf{be}$ and $\mathrm{t}(w') = \mathsf{sg}$, then there is no $(v, w') \in E$ such that $\mathrm{t}(w') = \mathsf{fdep}$.

Many of the conditions that we have imposed on Repairable Fault Trees follow two main reasons: the RFT should not have loops and it should be deterministic. Conditions (i), (ii) and (iii) are usual conditions for FTs, since they ensure that the structure corresponds to a tree. Condition (iv) ensures that dummy outputs stay free. Conditions (v) and (vi) make sure that RBOX work correctly. In particular (vi) avoids non-deterministic situations where, after a basic element fails, more than one RBOX is available to fix it.

Condition (vii) ensures that SBEs act only as a spare. This is, they are connected as spare parts of a spare gate and they can be repaired as other basic elements. Furthermore, it ensures that spare gates only manage BEs and SBEs, since managing other gates would require a larger analysis on determinism that is not part of this work.

The following items determine, for a given SBE, which spare gates are able to request it. Correctly defining the function si would be essential for this, hence condition (viii) is there to ensures so. Condition (ix) controls that the main element of a spare gate is only managed by this same spare gate. Finally condition (x) intends to avoid non-deterministic situations such as the ones shown in Figure 5.7. These non-deterministic situations are a result of the race conditions to acquire a replacement between main events that become inaccessible at the same time. Note that since FDEP can be replaced by an OR gates system (as we showed in 5.5), connecting main basic elements to the FDEP would not meet condition (vii) either.

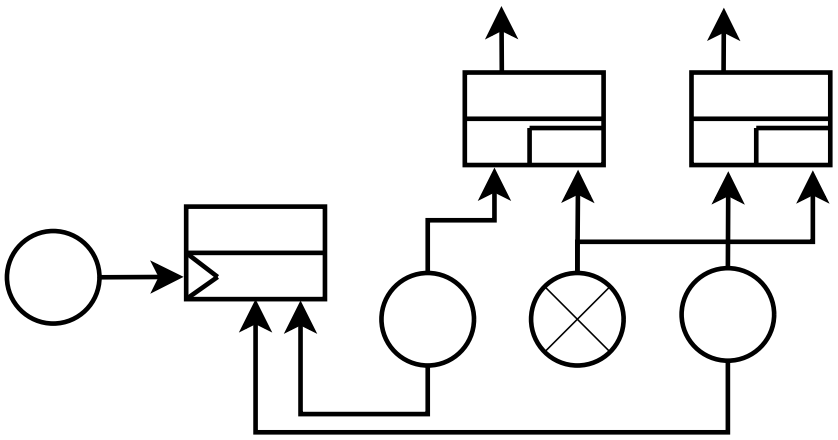


Figure 5.7: Race condition between spare gates by simultaneous failing of main components given an FDEP connection.

For the rest of this work we will only consider well formed RFTs.

5.5 Semantics of RFT

We present now a parametric semantic for each element in $\mathcal{E}$. This will be used later to define the semantic for each vertex of RFT, and the consequent semantic of the full model as a parallel composition of that of its components. We only give the semantic for BEs, AND gates, OR gates, PAND gates, and RBOX. Remember that FDEP gates are a syntax sugar, and can be replaced using OR gates. Similarly voting gates can be modeled by a series of AND gates and an OR gate, although a simpler model is presented here. Spare gates and SBEs will be presented later at Section 5.7.

In the design of the IOSA modules we should take into account the communication between each element of an RFT and its children and parents. For instance a basic element has to communicate its failure and repair to those gates for which it is an input. Similarly, a RBOX has to communicate to its inputs a "start repairing" signal. In order to do so, the semantic of each element will be given by a function, which takes actions as parameters. We are going to call these actions signals, as they will represent the failing, repairing, and other signals from RFT elements. In this sense, for an input index i of an element $e \in \mathcal{E}$ we will name f_i to the failing signal from that

input, u_i to the up (repaired) signal, and r to the signal that repair boxes send to their basic elements to start the repairing process. Accordingly, we will name output signals with f for failure, u for repair, and the r for "start repairing".

Basic Element

For a *BE* element $e \in \mathcal{E}$, its semantics is a function $[\![(\mathsf{be}, 0, \mu, \gamma)]\!] : \mathcal{A}^5 \rightarrow$ IOSA, where $[\![(\mathsf{be}, 0, \mu, \gamma)]\!](fl, up, f, u, r)$ results in the IOSA of Figure 5.8. The state of a basic element is defined by the fail clock `fc`, the repair clock

```
 1 module BE
 2   fc, rc : clock;
 3   signal : [0..2] init 0;
 4   broken : [0..2] init 0;
 5
 6   [ fl!] broken=0 @ fc -> (signal'=1) & (broken'=1);
 7   [ r??] broken=1 -> (broken'=2) & (rc'=γ);
 8   [ up!] broken=2 @ rc -> (signal'=2) &
 9                           (broken'=0) & (fc'=μ);
10
11   [ f!!] signal=1 -> (signal'=0);
12   [ u!!] signal=2 -> (signal'=0);
13 endmodule
```

Figure 5.8: Basic Element IOSA symbolic model.

`rc`, a variable `signal` that indicates when to signal the failure or repair, and variable `broken` to distinguish between broken and normal states. A basic element fails when clock `fc` expires (line 6) and immediately informs it with the urgent signal `f!!` at line 11. As soon as the repair begins by the corresponding connected repair box (line 7), clock `rc` is set. When it expires, the component becomes repaired. Hence, `fc` is set again at line 8, and the repair is signaled with urgent action `u!!` at line 11. At the starting state of an IOSA module all its clocks are set randomly according to their associated distributions. Thus, `rc` is set at the initial state and could eventually expire without having been set by a repair transition. This is why we have to

distinguish between cases when the BE is being repaired (`broken=2`) from when it is not.

AND gate

For an *AND gate* element with two inputs, its semantics is a function $[\![(\mathsf{and}, 2)]\!] : \mathcal{A}^6 \to \text{IOSA}$, where $[\![(\mathsf{and}, 2)]\!](f, u, f_1, u_1, f_2, u_2)$ results in the following IOSA:

```
 1 module AND
 2  singalf: bool init false;
 3  signalu: bool init false;
 4  count: [0..2] init 0;
 5
 6  [ f1??] count=1 -> (count'=2) & (signalf'=true);
 7  [ f1??] count=0 -> (count'=1);
 8  [ f1??] count=2 -> ;
 9  [ f2??] count=1 -> (count'=2) & (signalf'=true);
10  [ f2??] count=0 -> (count'=1);
11  [ f2??] count=2 -> ;
12
13  [ u1??] count=2 -> (count'=1) & (signalu'=true);
14  [ u1??] count=1 -> (count'=0);
15  [ u1??] count=0 -> ;
16  [ u2??] count=2 -> (count'=1) & (signalu'=true);
17  [ u2??] count=1 -> (count'=0);
18  [ u2??] count=0 -> ;
19
20  [ f!!] signalf & count=2 -> (signalf'=false);
21  [ u!!] signalu & count!=2 -> (signalu'=false);
22 endmodule
```

At lines 6 to 11, the AND gate gets informed of the failure of either of its inputs. When so, we distinguish between the case where the other input has already failed (`count=1`) and the case where it has not (`count=0`). In the first case we have to output the failure of this gate, for which we set the `signalf` variable in order to enable transition at line 20. Furthermore in both cases we increase the value of `count` so that we take note of the failure of an input. A similar reasoning is done for the case of the repairing of an input at lines 13 to 18. In this case we have to set the module to signal a repair when

an input gets repaired at a state where both inputs where failing (lines 13 and 16), by enabling transition at line 21. In further gates modeling we will omit writing down self loops originated by IOSA's input enableness, such as lines $8, 11, 15$ and 18 of the AND gate model. Nevertheless, we remark that it is necessary to take them into account when analyzing confluence on the modules at section 5.6. While we have described an AND gate with only two inputs, it is easy to imagine a generalization to a larger number of inputs. Furthermore, it is easy to model an N inputs AND gate by composing several AND gates in cascade or pyramid topologies.

OR gate

For an OR gate element $e \in \mathcal{E}$ with two inputs, its semantic is a function $[\![(\mathsf{or}, 2)]\!] : \mathcal{A}^6 \to \text{IOSA}$, where $[\![(\mathsf{or}, 2)]\!](f, u, f_1, u_1, f_2, u_2)$ results in the following IOSA:

```
 1 module OR
 2   signalf: bool init false;
 3   signalu: bool init false;
 4   count: [0..2] init 0;
 5
 6   [ f1??] count=0 -> (count'=1) & (signalf'=true);
 7   [ f1??] count=1 -> (count'=2);
 8   [ f2??] count=0 -> (count'=1) & (signalf'=true);
 9   [ f2??] count=1 -> (count'=2);
10
11   [ u1??] count=2 -> (count'=1);
12   [ u1??] count=1 -> (count'=0) & (signalu'=true);
13   [ u2??] count=2 -> (count'=1);
14   [ u2??] count=1 -> (count'=0) & (signalu'=true);
15
16   [ f!!] signalf & count!=0 -> (signalf'=false);
17   [ u!!] signalu & count=0 -> (signalu'=false);
18 endmodule
```

Notice that the OR gate model is very similar to the AND gate. We take as a premise for these models that an input will not break two times in a row without being repaired in the middle, neither it will be repaired if it has not failed. Just as for AND gates, OR gates with more than 2 inputs can be

easily modeled or replaced by a combination of 2-input OR gates.

VOT gate

The following IOSA model corresponds to a 2 from 3 voting gate. A generalization to other values of N and K can be easily obtained. Although an alternative modeling of this gates can be obtained by a combination of OR and AND gates, one may want to reduce the complexity of the system modeling by using the interpretation presented here, which also happens to be weakly deterministic.

For a 2 from 3 Voting gate element $e \in \mathcal{E}$, its semantic is a function $[\![(\mathsf{vot}, 3, 2)]\!] : \mathcal{A}^8 \to \mathrm{IOSA}_u$, where $[\![(\mathsf{vot}, 3, 2)]\!](f, u, f_0, u_0, f_1, u_1, f_2, u_2)$ results in the following IOSA_u:

```
 1 module VOTING
 2   count: [0..3] init 0;
 3   inform: bool init false;
 4
 5   [ f0??] -> (count'=count+1) & (inform'=(count+1=2));
 6   [ f1??] -> (count'=count+1) & (inform'=(count+1=2));
 7   [ f2??] -> (count'=count+1) & (inform'=(count+1=2));
 8
 9   [ u0??] -> (count'=count-1) & (inform'=(count=2));
10   [ u1??] -> (count'=count-1) & (inform'=(count=2));
11   [ u2??] -> (count'=count-1) & (inform'=(count=2));
12
13   [ f!!] inform & count >= 2 -> (inform'=false);
14   [ u!!] inform & count < 2 -> (inform'=false);
15 endmodule
```

Voting gates are modeled using a counter which counts how many inputs have failed. This is done by listening to the corresponding fail signals at lines 5 to 7, and repair signals at lines 9 to 11. In these same lines we take into account if we have just reached the K value (2 in our example) or if we have just gone down this value, which are the circumstances under which to inform the failure and repair respectively, which is finally done at lines 13 and 14.

PAND gate

The semantics of a *Priority AND gate* with 2 inputs is defined by $[\![(\mathsf{pand}, 2)]\!] : \mathcal{A}^6 \rightarrow$ IOSA, where $[\![(\mathsf{pand}, 2)]\!](f, u, f_0, u_0, f_1, u_1)$ results in the following IOSA$_u$:

```
 1 module PAND
 2
 3  f1: bool init false;
 4  f2: bool init false;
 5  st: [0..4] init 0; // 0:up, 1:inform fail, 2:failed,
 6                     // 3:inform up, 4:unbreakable
 7
 8  [_?] st=0 & f1 & !f0 -> (st'=4);
 9
10  [ f0??] st=0 & !f0 & !f1-> (f0'=true);
11  [ f0??] st=0 & !f0 & f1 -> (st'=1) & (f0'=true);
12  [ f0??] st!=0 & !f0 -> (f0'=true);
13  [ f0??] f0 -> ;
14
15  [ f1??] st=0 & !f0 & !f1 -> (f1'=true);
16  [ f1??] st=0 & f0 & !f1 -> (st'=1) & (f1'=true);
17  [ f1??] st=3 & !f1 -> (st'=2) & (f1'=true);
18  [ f1??] (st==1|st==2|st=4) & !f1 -> (f1'=true);
19  [ f1??] f1 -> ;
20
21  [ u0??] st!=1 & f0 -> (f0'=false);
22  [ u0??] st=1 & f0 -> (st'=0) & (f0'=false);
23  [ u0??] !f0 -> ;
24
25  [ u1??] (st=0|st=3) & f1 -> (f1'=false);
26  [ u1??] (st=1|st=4) & f1 -> (st'=0) & (f1'=false);
27  [ u1??] st=2 & f1 -> (st'=3) & (f1'=false);
28
29  [ f!!] st=1 -> (st'=2);
30  [ u!!] st=3 -> (st'=0);
31
32 endmodule
```

Priority AND gates fail only when all their inputs fail and they do so from left to right. This allows to condition the failure of a system not only to the failure of subsystems but also to the ordering in time in which they occur. Notice that an n inputs PAND gate can be modeled by a system of $n-1$ two-input PAND gates connected in a cascade topology. Literature is not always clear or even disagrees on what should be the behavior of the PAND gate in case both inputs fail at the same time [77, 34, 69]. This situation arises in some constructions with AND and OR gates, or when the inputs of a PAND gate are connected to a same FDEP (see Figure 5.9), since there is no established order in the failing of the dependent BEs. It should even be questioned if this represents a non-deterministic behavior. Some works do not allow these situations and discard them during early syntactic checks [91]. Some others find that the nondeterminism is important to analyze real scenarios where the behavior is in fact unknown [17]. Other works decided that the PAND gate will not break unless its inputs break strictly from left to right [16, 12]. Some works allow PAND gates to break when both inputs fail at the same time [34, 20, 19]. In our case we agree with this last option and decide to model our gate to be able to identify if time has passed between the occurrence of the failures, and act consequently. In the particular case where no time passes between the failure of the inputs, we consider that the order in which the dependent BEs fail does not really matter and thus the non-determinism is spurious.

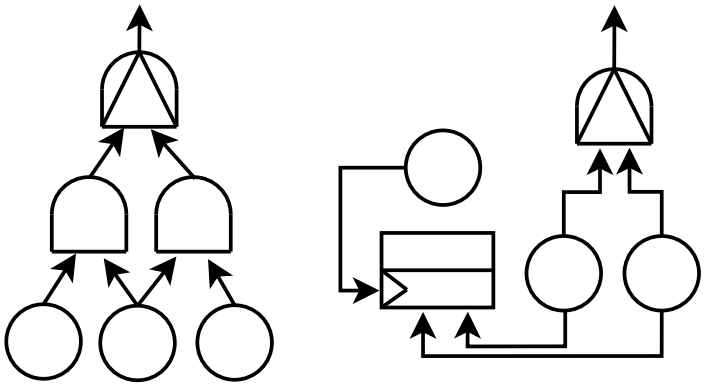


Figure 5.9: Spurious non-determinism.

To identify if time has passed between the occurrence of the input failures, our model has to listen to the output actions which indicate that a clock

has expired. This is done by the special input action at line 8, which will synchronize with all urgent outputs, no matter which action they trigger. Notice that there is only one scenario that we want to rule out, which is when the second input fails and then time passes without the first input failing too. This is in fact the case described by line 8 guard. Furthermore this transition takes us to the 'failsafe' state (`st=4`), from which we can only go back by fixing the last input. In consequence, the failure of our gate occurs either if both inputs fail at the same time or if the first input fails, then time passed, and then the second input fails. A variable `st` allows us to distinguish between states where the PAND gate is working (value 0), from when it needs to signal a failure (value 1), or it is failed and there is no need to signal anymore (value 2), or if it needs to signal that it has been repaired (value 3), or finally it is in a failsafe state, as a consequence of the inverse order of the occurrence of the failures (value 4).

Repair Box with priority policy

The semantics of a *n inputs repair box with priority policy*, is a function $[\![(\mathsf{rbox}, n)]\!] : \mathcal{A}^{3*n} \rightarrow$ IOSA, where $[\![(\mathsf{rbox}, n)]\!](\mathit{fl}_0, \mathit{up}_0, r_0, ..., \mathit{fl}_{n-1}, \mathit{up}_{n-1}, r_{n-1})$ is the following IOSA:

```
 1 module RBOX
 2  broken[n]: bool init false;
 3  busy: bool init false;
 4
 5  [ fl_0?] -> (broken[0]'=true);
 6  ...
 7  [ fl_{n-1}?] -> (broken[n-1]'=true);
 8
 9  [ r_0!!] !busy & broken[0] -> (busy'=true);
10  ...
11  [ r_{n-1}!!] !busy & broken[n-1] & !broken[n-2]
12               & ... & !broken[0] -> (busy'=true);
13
14  [ up_0?] -> (broken[0]'=false) & (busy'=false);
15  ...
16  [ up_{n-1}?] -> (broken[n-1]'=false) & (busy'=false);
17 endmodule
```

The priority repair box uses an array to keep track of failed inputs (`broken[n]`), updating it when it receives their fail signals (lines 5 to 7) and up signals (lines 13 to 15). At the same time, when not busy, it goes on sending repair signals to broken inputs (lines 9 to 12). Note that instead of listening to the urgent output signals of the input BEs, it listens for the non-urgent actions of the transitions that trigger the failure or repair. This is done with the only purpose of facilitating the confluence analysis over this module.

Repair Box with first come first serve policy

The semantics of a *repair box with first come first serve policy* with n inputs, is a function $[\![(\mathsf{rbox}, n)]\!] : \mathcal{A}^{3*n} \rightarrow$ IOSA, where $[\![(\mathsf{rbox}, n)]\!](\mathit{fl}_0, \mathit{up}_0, r_0, ..., \mathit{fl}_{n-1}, \mathit{up}_{n-1}, r_{n-1})$ results in the following IOSA:

```
 1 module RBOX % with first come first serve policy
 2   queue[n]: [0..n] init 0;
 3   busy: bool init false;
 4   r: [0..n] init n;
 5   dummy: [0..0] init 0;
 6
 7   [ fl_0?] -> (dummy'=broken(queue,0));
 8   ...
 9   [ fl_{n-1}?] -> (dummy'=broken(queue,n-1));
10
11   [!!] some(queue,0) & r=n -> (r'=maxfrom(queue,0));
12
13   [ r_0!!] !busy & r=0 -> (busy'=true) & (queue[0]'=0);
14   ...
15   [ r_{n-1}!!] !busy & r=n-1 -> (busy'=true) & (queue[n-1]'=0);
16
17   [ up_0?] -> (queue[0]'=false) & (busy'=false) & (r' = n);
18   ...
19   [ up_{n-1}?] -> (queue[n-1]'=false) & (busy'=false) & (r' = n);
20 endmodule
```

The model for a repair box with first come first serve policy uses an array to mark down each broken input. Notice that each position in the queue corresponds to each input. A value 0 on an index i means that the input i has not failed, while a grater value on that position indicates for "how long"

has it been broken. Repair boxes use some syntactic elements present in FIG simulation tool [25]. These elements do not introduce a new semantic behavior and are there only to reduce the complexity and obfuscation that would represent modeling this using only the grammar presented in Appendix A. Examples of this are the function `broken` at line 7, which given an array, in this case `queue`, and an index, in this case `0`, it increases by one the value at that index and every other value grater than 0 in the array. In this way we can check the order in which the inputs failed by comparing the values at the corresponding index. The grater the value the sooner they broke. The syntactic function `some`, on the other hand, returns a boolean value indicating if there is some value different to zero in the array. In this case we use it to check if there is any failed input. If there is at least one, then `maxfrom` function will return the index of the highest value in `queue`, which corresponds to the input who broke first in between all the broken ones. The `dummy` variable is used only to fullfil with the syntactic restrictions of IOSAs assignments. For a quick determinism analysis we point out that all `broken`, `fstexclude`, and `maxfrom` are deterministic. Furthermore all pairs of urgent transitions in the model are confluent given that their guards are mutually exclusive due to the value of variable `r`.

A general model for RBOX.

The following model can be used to model many different repair boxes with arbitrary policies:

```
1  module RBOX % general policy
2    queue[n]: [0..n] init 0;
3    busy: bool init false;
4    r: [0..n] init n;
5
6    [ fl_0?] -> (dummy'=broken(queue,0));
7    ...
8    [ fl_{n-1}?] -> (dummy'=broken(queue,n-1));
9
10   [!!] some(queue) & r=n -> (r'=policy(queue));
11
12   [ r_0!!] !busy & r=0 -> (busy'=true) & (queue[0]'=0);
13   ...
14   [ r_{n-1}!!] !busy & r=n-1 -> (busy'=true) & (queue[n-1]'=0);
```

```
15
16    [ up_0?] -> (queue[0]'=false) & (busy'=false) & (r' = n);
17    ...
18    [ up_{n-1}?] -> (queue[n-1]'=false) & (busy'=false) & (r' = n);
19  endmodule
```

Notice that in this general RBOX model we take note not only the failed inputs but also on their order of failure. This is done by using the `broken` function already explained, along with a compatible array (`queue`). A simpler model can be obtained if we are not interested on the order in which they failed. Defining how `policy` selects the next input to repair from the broken inputs marked at `queue` will produce the desired RBOX. In our case we are only interested of course on deterministic policies, otherwise this function would render the model non-deterministic.

Semantic of RFT

The semantic of a RFT is that of the parallel composition of the semantic of its components, being conveniently synchronized.

Definition 5.3. Given a RFT $T = (V, i, si, l)$ we define the semantic of T as $[\![T]\!] = ||_{v \in V} [\![v]\!]$ where $[\![v]\!]$ is defined by:

$$[\![v]\!] = \begin{cases} [\![l(v)]\!](\mathtt{fl}_v, \mathtt{up}_v, \mathtt{f}_v, \mathtt{u}_v, \mathtt{r}_v) & \text{if } l(v) = (\mathsf{be}, 0, \mu, \gamma) \\ [\![l(v)]\!](\mathtt{f}_v, \mathtt{u}_v, \mathtt{f}_{i(v)[0]}, \mathtt{u}_{i(v)[0]}, ..., \mathtt{f}_{i(v)[n-1]}, \mathtt{u}_{i(v)[n-1]}) & \\ & \text{if } l(v) \in \{(\mathsf{and}, n), (\mathsf{or}, n)\} \\ [\![l(v)]\!](\mathtt{f}_v, \mathtt{u}_v, \mathtt{f}_{i(v)[0]}, \mathtt{u}_{i(v)[0]}, \mathtt{f}_{i(v)[1]}, \mathtt{u}_{i(v)[1]}) & \text{if } l(v) = (\mathsf{pand}, 2) \\ [\![l(v)]\!](\mathtt{fl}_{i(v)[0]}, \mathtt{up}_{i(v)[0]}, \mathtt{r}_{i(v)[0]}, ..., \mathtt{fl}_{i(v)[n-1]}, \mathtt{up}_{i(v)[n-1]}, \mathtt{r}_{i(v)[n-1]}) & \\ & \text{if } l(v) = (\mathsf{rbox}, n) \end{cases}$$

Notice that the function i is used to conveniently synchronize to each input. In fact, $i(v)$ is an ordered list of the inputs of v. Suppose as an example that v is an AND gate, and that the BE w is the first input of v. Then the name assigned to the fail signal of w will be f_w, since $l(w) = (\mathsf{be}, 0, \mu, \gamma)$ (see Definition 5.3). Conveniently, the name of the first failing input signal of v will be named f_w, since $lv = (\mathsf{and}, n)$ and $i(v) = [w, ...]$ and thus $i(v)[0] = w$. In Section 5.7, we extend the semantics to spare gates and SBEs.

5.6 RFTs are deterministic

In this section we show that RFTs composed only by BEs, AND gates, OR gates, PAND gates, and RBOX, are weakly deterministic. We make use of the results on IOSA given in Section 4.5. Since voting and FDEP gates can be constructed using OR and AND gates, we are also proving that they can be modeled deterministically. We will work under the premise that we can build the reachable states space for each component in reasonable time and space. This is not a strange assumption since compositionality allows us to keep components small enough. In the following propositions we enumerate the sets of initially and spontaneously enabled actions in a RFT. Afterwards we analyze the possible cases of non confluent actions in a RFT, and furthermore we describe their approximate indirect triggering relation. With all these ingredients we can finally prove that RFTs are weakly deterministic by applying Theorem 4.5.

Proposition 5.1. Let T be a RFT. $[\![T]\!]$ has no initially enabled actions. Moreover, the only spontaneous sets of actions are singletons of the form $\{f_v\}$ and $\{u_v\}$, for $\mathrm{t}(v) = \mathsf{be}$, which are spontaneously enabled by fl_v and up_v, respectively.

Proof. As a consequence of Proposition 4.6, the initially enabled actions of $[\![T]\!]$ are contained in the union of the sets of initially enabled actions of its components $[\![v]\!]$, $v \in V$, and the spontaneously enabled actions of $[\![T]\!]$ are contained in the union of the spontaneously enabled sets of $[\![v]\!]$. It is direct to see that, for any element $e \in \mathcal{E}$, none of the urgent outputs are enabled at the initial state of $[\![e]\!]$, since their guards are initially false. Furthermore, the only non-urgent output transition in our models are at lines 6 and 8 of the BE (Figure 5.6). Let $v \in V$ be such that $\mathrm{t}(v) = \mathsf{be}$. Then, after taking transition at line 6 the only urgent output enabled is $\mathtt{f}_v$ (on the instance $[\![v]\!]$), while after taking transition at line 8 the only one is $\mathtt{u}_v$, and thus these are the only possible spontaneous enabled actions. □

Proposition 5.2. Let T be a RFT. The only possible pairs of non-confluent actions in $[\![T]\!]$ are:

- $\{(\mathtt{f}_v, \mathtt{u}_{v'}) \mid v, v' \in i(w), t(w) \in \{\mathsf{and}, \mathsf{or}, \mathsf{pand}\}\}$, and
- $\{(\mathtt{f}_w, \mathtt{u}_v), (\mathtt{u}_w, \mathtt{f}_v) \mid v \in i(w), t(w) \in \{\mathsf{and}, \mathsf{or}\}\}$.

Proof. Parallel composition does not introduce new non-confluent pair of actions and, moreover, it preserves the confluence of its components (Proposition 4.2). Thus, we look at the components in isolation. First notice that transitions in a IOSA module are defined symbolically. Each symbolic transition in a module describes, in fact, a set of IOSA transitions, which become concrete when the symbolic transition is evaluated on a state that satisfies the guard. Notice also that a state in a module is defined by the current values of its variables. When analyzing that two urgent actions a and b are confluent in a module, for each symbolic transition t_a and t_b defined for those actions in that module, we look for a *non-confluence witness*, i.e, a state that satisfies the guards of t_a and t_b and shows that a and b are not confluent (i.e., the pair does not satisfy Def. 4.5). Note that by only checking reachable states in the component, we are already overapproximating the reachable states in the composition.

For this proof we only analyze the case of the AND gate. For other RFT elements, the proof follows similarly. Let v be a vertex in a RFT such that $l(v) = (\mathsf{and}, 2)$.

We analyze $\mathtt{f}_1$ against $\mathtt{u}_1$ in $[\![(\mathsf{and}, 2)]\!]$ (see the semantics of AND) and show that they are not confluent. Take for instance state s defined by `count=1`, `signalf=false` and `signalu=false`, which can be easily checked to be reachable. There, we find that it enables symbolic transitions at lines 6 (with label $\mathtt{f}_1$) and 14 (with label $\mathtt{u}_1$). On the one hand, transition at line 6 moves to the state where `count=2`, `signalf=true` and `signalu=false` is reached. At this point action $\mathtt{u}_1$ can only be performed through transition at line 13, which yields state s' defined by `count=1`, `signalf=true` and `signalu=true`. On the other hand, transition at line 14 moves to the state where `count=0`, `signalf=false` and `signalu=false`. This state only enables $\mathtt{f}_1$ at line 7, which yields state s'' defined by `count=1`, `signalf=false` and `signalu=false`. Since s' and s'' are two different states, we have proved that $\mathtt{f}_1$ and $\mathtt{u}_1$ are not confluent. Similarly, we can show that the pairs $(\mathtt{f}, \mathtt{u}_i)$ and $(\mathtt{u}, \mathtt{f}_i)$, for $i = 1, 2$, are not confluent.

All other pairs are confluent. Take for instance transitions at lines 7 and 10 which are defined for actions $\mathtt{f}_1$ and $\mathtt{f}_2$ respectively, and the state s defined by `count=0`, `signalf=false` and `signalu=false`. On the one hand, line 7 leads to the state where `count=1`, `signalf=false` and `signalu=false` which in turns enables $\mathtt{f}_2$ only at line 9 yielding state s' defined by `count=2`, `signalf=true` and `signalu=false`. On the other hand, line 10 at state s moves to the state where `count=1`, `signalf=false` and `signalu=false` which only enables

$\mathtt{f}_1$ at line line 6 yielding the same state s'. The proof follows similarly from any other reachable state enabling $\mathtt{f}_1$ and $\mathtt{f}_2$ showing, thus, that $\mathtt{f}_1$ and $\mathtt{f}_2$ are confluent. In some other cases the proof of confluence follows from the fact that the pair of actions are never enabled simultaneously, as it is the case, e.g., of $\mathtt{f}$ and $\mathtt{u}$ (notice that the guards enablyng each one of them are mutually exclusive). □

Proposition 5.3. Let T be a RFT. For each $v \in V$, the triggering relation of $[\![v]\!]$ is given by:

- $\{\}$, if $l(v) \in \{(\mathsf{be}, 0, \mu, \gamma), (\mathsf{rbox}, n)\}$,
- $\{(\mathtt{f}_w, \mathtt{f}_v) \mid w \in i(v)\} \cup \{(\mathtt{u}_w, \mathtt{u}_v) \mid w \in i(v)\}$, if $l(v) \in \{(\mathsf{and}, n), (\mathsf{or}, n)\}$, and
- $\{(\mathtt{u}_w, \mathtt{u}_v) \mid w = i(v)[1]\} \cup \{(\mathtt{f}_w, \mathtt{f}_v) \mid w \in i(v)\}$, if $l(v) = (\mathsf{pand}, 2)$.

Proof (sketch). It suffices to make a satisfiability analysis over guards and postconditions of each pair (t_a, t_b) with t_b an output urgent symbolic transition and t_a any urgent symbolic transition, taking into account only reachable states. □

Theorem 5.1. Let T be a RFT. Then $[\![T]\!]$ is weakly deterministic.

Proof. We look for a, b, c, d and e as well as sets B_i with $i = 1 \ldots n$ as Theorem 4.5 suggests. Since Prop. 5.1 ensures that there are no initially enabled actions in $[\![T]\!]$, c and d should be spontaneously enabled actions. By the same proposition either e is of the form $\mathtt{fl}_v$ for some v and then $\bigcup_{i=1}^{1} B_i = B_1 = \{\mathtt{f}_v\}$, or e is of the form $\mathtt{up}_v$ for some v and then $\bigcup_{i=1}^{1} B_i = B_1 = \{\mathtt{u}_v\}$. In the first case, we get $c = d = \mathtt{f}_v$ for some v, and in the second case $c = d = \mathtt{u}_v$. Furthermore, by Prop. 5.2, either a is of the form $\mathtt{f}_w$ for some w and b is of the form $\mathtt{u}_{w'}$ for some w' or the other way around. As shown by Prop. 5.3, fail actions ($\mathtt{f}_v$ for some v) only trigger fail actions, and up actions ($\mathtt{u}_v$ for some v) only trigger up actions, thus it is impossible that c and d indirectly trigger both a and b respectively. Therefore, it is not possible to find actions a, b, c, d, and e satisfying conditions 1 to 3 in Theorem 4.5, and hence $[\![T]\!]$ is confluent. Since $[\![T]\!]$ is also closed, then it is weakly deterministic. □

5.7 An extended semantics

In this section we extend the semantics of RFTs to spare gates and spare basic elements. As before, we aim to guarantee that IOSA models derived from the RFT is weakly deterministic. In order to do so, we need to bring special attention to two particular scenarios that could introduce non-determinism.

The first scenario arises when a main basic element fails at a spare gate which is served with several spare basic elements. At this point, we have the question of which of the available spare basic elements should be taken by the spare gate. Traditionally, spare elements are selected in order from an ordered set. To generalize this mechanism for the selection of the spares we intend to allow for more complex state-involved policies. It should be always the case that this policy chooses deterministically. The second scenario arises when several spare gates have requested are available SBE, being it broken or bussy by yet another spare request. The non-deterministic situation will arise when the SBE gets repaired or freed by the holding spare gate respectively. At this point, it is unclear which of the requesting spare gates will take the newly available SBE. For this, we define sharing policies on the SBE. Thus, to provide semantics to an SBE, we actually introduce two IOSA modules: one extending the behavior of a BE with the posibility of changing from dormant to enabled state and vice versa, and another one, the so-called *multiplexer* module, which manages the sharing of the SBE.

The Spare basic element (SBE)

The semantics of a *SBE* is a function $[\![(\mathsf{sbe}, n, \mu, \nu, \gamma)]\!] : \mathcal{A}^{7+5*n} \rightarrow$ IOSA, where $[\![(\mathsf{sbe}, n, \mu, \nu, \gamma)]\!](\mathit{fl}, \mathit{up}, \mathit{f}, \mathit{u}, \mathit{r}, \mathit{e}, \mathit{d}, \mathit{rq}_0, \mathit{asg}_0, \mathit{rel}_0, \mathit{acc}_0, \mathit{rj}_0, ..., \mathit{rq}_{n-1}, \mathit{asg}_{n-1}, \mathit{rel}_{n-1}, \mathit{acc}_{n-1}, \mathit{rj}_{n-1})$ results in the following pair of IOSA modules:

```
1  module SBE
2   fc, dfc, rc : clock;
3   inform : [0..2] init 0;
4   active : bool init false;
5   broken : [0..2] init 0;
6
7   [ e??] !active -> (active'=true) & (fc'=μ);
8   [ d??] active -> (active'=false) & (dfc'=ν);
9
10  [ fl!] active & broken=0 @ fc -> (inform'=1) & (broken'=1);
11  [ fl!] !active & broken=0 @ dfc -> (inform'=1) & (broken'=1);
12  [ r??] -> (broken'=2) & (rc'=γ);
13  [ up!] active & broken=2 @ rc -> (inform'=2) & (broken'=0) & (fc'=μ);
```

```
14   [ up!] !active & broken=2 @ rc -> (inform'=2) & (broken'=0) & (dfc'=μ);
15
16   [ f!!] inform=1 -> (inform'=0);
17   [ u!!] inform=2 -> (inform'=0);
18  endmodule
```

```
 1  module MUX
 2   queue[n]: [0..3] init 0; % idle, requesting, reject, using
 3   avail: bool init true;
 4   broken: bool init false;
 5   enable: [0..2] init 0;
 6
 7   [ fl?] -> (broken'=true);
 8   [ up?] -> (broken'=false);
 9
10   [ e!!] enable=1 -> (enable'=0);
11   [ d!!] enable=2 -> (enable'=0);
12
13   [ rq0??] queue[0]=0 & (broken | !avail) -> (queue[0]'=2);
14   [ rq0??] queue[0]=0 & !broken & avail -> (queue[0]'=1);
15   [ asg0!!] queue[0]=1 & !broken & avail -> (queue[0]'=3) & (avail'=false);
16   [ rj0!!] queue[0]=2 -> (queue[0]'=1);
17   [ rel0??] queue[0]=3 -> (queue[0]'=0) & (avail'=true) & (enable'=2);
18   [ acc0??] -> (enable'=1);
19   ...
20   [ rqn−1??] queue[n-1]=0 & (broken | !avail) -> (queue[n-1]'=2);
21   [ rqn−1??] queue[n-1]=0 & !broken & avail -> (queue[n-1]'=1);
22   [ asgn−1!!] queue[n-1]=1 & queue[n-2]=0 & ... & queue[0]=0 & !broken & avail
23                  -> (queue[n-1]'=3) & (avail'=false);
24   [ rjn−1!!] queue[n-1]=2 -> (queue[n-1]'=1);
25   [ reln−1??] queue[n-1]=3 -> (queue[n-1]'=0) & (avail'=true) & (enable'=2);
26   [ accn−1??] -> (enable'=1);
27
28  endmodule
```

Apart from the new MUX model, the model of the SBE differs with the previously given model of the BE in the introduction of a new failing clock (`dfc`) controlling failures in the disabled mode, a new variable (`enabled`) distinguishing between enabled and disabled states, and a few somehow mirrored lines in order to distinguish between active and disabled states and act accordingly (lines 11 and 12, 14 and 16). Furthermore two new actions and their corresponding transitions are introduced in order to be able to enable or disable the SBE when needed (lines 8 and 9).

In the case of the multiplexer, we decided to model it with a priority policy, which prioritizes lower index input spare gates to higher indexed ones (notice assignment transitions at line 16 and 25 of the multiplexer module.)

Other kinds of policies can be defined as for repair box gates. In the model, actions rq_i indicate that the spare gate input i is requesting the spare. acc_i indicates that input i accepts the spare that has previously been assigned to it through action asg_i. On the other hand action rj_i indicates that it rejects it. Action rel_i indicates that input i is releasing the spare that has previously been assigned to such input. Finally actions e and d enable and disable the spare basic element when needed.

Notice that no multiplexer would have been needed in the absences of repair boxes. Since in such cases SBEs do not become available after they are taken or fail, there would be no need to solve any non-determinism. No non-determinism would have arrisen if spare elements where not shared by different spare gates [12, 11]. An other possibile situation of non-determinism would have been race conditions between two spare gates that fail at the same time, as studied in [69]. Nevertheless, we have discarded this race conditions from our semantics by introducing the last two conditions of Definition 5.2 along with the fact that two simultaneous failures of basic elements is not possible in the IOSA deterministic semantics.

The Spare Gate (SG)

The semantics of a *spare gate with priority policy* is a function $[\![(\mathsf{sg}, n)]\!] : \mathcal{A}^{4+7*n} \rightarrow$ IOSA, where $[\![(\mathsf{sg}, n)]\!](f, u, fl_0, up_0, fl_1, up_1, rq_1, asg_1, acc_1, rj_1, rel_1 \ldots, fl_n, up_n, rq_n, asg_n, acc_n, rj_n, rel_n)$ is the following IOSA:

```
 1  module SPAREGATE
 2    state: [0..4] init 0; // on main, request, wait, on spare, broken
 3    inform: [0..2] init 0;
 4    release: [-n..n] init 0;
 5    idx: [1..n] init 1;
 6
 7    [ fl0?] state=0 -> (state'=1) & (idx'=1);
 8    [ up0?] state=4 -> (state'=0) & (inform'=2);
 9    [ up0?] state=3 & idx=1 -> (state'=0) & (idx'=1) & (release'=1);
10    ...
11    [ up0?] state=3 & idx=n -> (state'=0) & (idx'=1) & (release'=n);
12
13    [ fl1?] state=3 & idx=1 -> (release'=1);
14    ...
15    [ fln?] state=3 & idx=n -> (release'=n);
16
17    [ rq1!!] state=1 & idx=1 -> (state'=2);
18    ...
19    [ rqn!!] state=1 & idx=n -> (state'=2);
20
21    [ asg1??] state=0 | state=1 | state=3 -> (release'=1);
```

```
22    [ asg_1??] state=2 & idx=1 -> (release'=-1) & (state'=3);
23    [ asg_1??] state=4 -> (release'=-1) & (state'=3) & (idx'=1) & (inform'=2);
24    ...
25    [ asg_n??] state=0 | state=1 | state=3 -> (release'=n);
26    [ asg_n??] state=2 & idx=n -> (release'=-n) & (state'=3);
27    [ asg_n??] state=4 -> (release'=-n) & (state'=3) & (idx'=n) & (inform'=2);
28
29    [ rj_1??] state=2 & idx=1 -> (idx'=2) & (state'=1);
30    [ rj_2??] state=2 & idx=2 -> (idx'=3) & (state'=1);
31    ...
32    [ rj_n??] state=2 & idx=n -> (state'=4) & (idx'=1) & (inform'=1);
33
34    [ rel_1!!] release=1 & !(state=3 & idx=1) -> (release'= 0);
35    [ rel_1!!] release=1 & state=3 & idx=1 -> (release'= 0) & (state'=1) & (idx'=1);
36    ...
37
38    [ rel_n!!] release=n & !(state=3 & idx=n) -> (release'=0);
39    [ rel_n!!] release=n & state=3 & idx=n -> (release'= 0) & (state'=1) & (idx'=1);
40
41    [ acc_1!!] release=-1 -> (release'= 0);
42    ...
43    [ acc_n!!] release=-n -> (release'=0);
44
45    [ f!!] inform = 1 -> (inform'=0);
46    [ u!!] inform = 2 -> (inform'=0);
47  endmodule
```

The model of the spare gate uses a priority policy over the available Spare BEs. This means that when looking for a Spare BE, it will start asking for it from the lower index input to the higher index input until obtaining a replacement. Other policies can be defined on the spare gate too, just as with the multiplexer and the repair box. In the model of a SG, the variable `state` distinguishes if the SG is working with its main BE, requesting a SBE, waiting for a response from its inputs, working on a SBE or broken. The vector `release` indicates for each SBE input i when the SG has to release (value i) or accept (value $-i$) the assignment of that SBE. Variable `idx` indicates which of the inputs to request next. Line 7 defines the transition which starts with the SBE acquiring protocol whenever the main BE fails. The following transitions up to line 15 release the acquired SBEs whenever they fail or the main BE is repaired. Transitions from lines 17 to 19 request the next possible SBE. After doing so, we need to wait for a response from the corresponding multiplexer (`state'=2`). The request can be rejected (lines 29 to 32), and we proceed by asking for the next SBE by setting `idx` to the corresponding value if there is one, or by failing in case none of the SBE where available (`state'=4` at line 32). A SBE can be assigned to the SG when not needed anymore (lines 21 and 25), or when the SG has requested it

in order to avoid failing (lines 22 and 26), or when the SG had already failed and thus it is repaired by using it (lines 23 and 27). I SG may want to release a SBE when it is being assigned to it but it does not need the SBE (lines 34 and 38) or when the SBE fails while the SG is using it (lines 35 and 39). The SG may accept assigned SBEs at lines 41 to 43. Finally, the SG signals failure at line 45 and repair at line 46. To further understand the meaning and intuition of each transition we refer the reader to the SBE description.

Extended semantics of RFT

We extend the semantic of the RFT with the SBE and SG elements as follows.

Definition 5.4. Given a RFT $T = (V, E)$, we extend Definition 5.3 with the following cases:

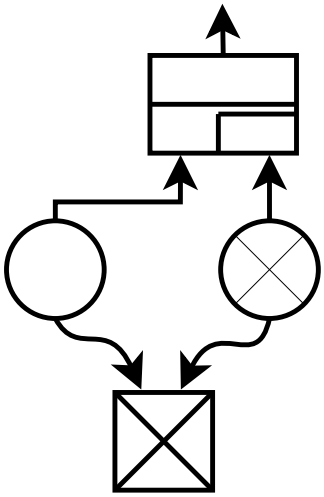
$$[\![v]\!] = \begin{cases} \cdots \\ [\![l(v)]\!](\mathtt{fl}_v, \mathtt{up}_v, \mathtt{f}_v, \mathtt{u}_v, \mathtt{r}_v, \mathtt{e}_v, \mathtt{d}_v, \mathtt{rq}_{(si(v)[0],v)}, \mathtt{asg}_{(v,si(v)[0])}, \\ \qquad \mathtt{rel}_{(si(v)[0],v)}, \mathtt{acc}_{(si(v)[0],v)}, \mathtt{rj}_{(v,si(v)[0])}, .., \mathtt{rj}_{(v,si(v)[n-1])}) \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{if } l(v) = (\mathsf{sbe}, n, \mu, \nu, \gamma) \\ [\![l(v)]\!](\mathtt{f}_v, \mathtt{u}_v, \mathtt{fl}_{i(v)[0]}, \mathtt{up}_{i(v)[0]}, \mathtt{fl}_{i(v)[1]}, \mathtt{up}_{i(v)[1]}, \mathtt{rq}_{(v,i(v)[1])}, \mathtt{asg}_{(i(v)[1],v)}, \\ \qquad \mathtt{acc}_{(v,i(v)[1])}, \mathtt{rj}_{(i(v)[1],v)}, \mathtt{rel}_{(v,i(v)[1])}, ..., \mathtt{rel}_{(v,i(v)[n-1])}) \\ \qquad\qquad\qquad\qquad\qquad\qquad\qquad\qquad \text{if } l(v) = (\mathsf{sg}, n) \end{cases}$$

Notice that in the case of the SBE and SG, several signals are indexed by a pair of elements. This pair indicates who performs the action and who listens to it at synchronization. As an example, $as_{(v,i(v)[0])}$ will indicate that the multiplexer that manages v, assigns its spare basic element to its first connected spare gate ($i(v)[0]$).

Determinism

Unfortunately, we could not find a direct way to prove that this extension is indeed weak deterministic, as we did with the RFT without spares. While the SBE module can be easily proved to be confluent, this is not the case of the models of the multiplexer and the spare gate. In fact, models with spare gates and SBEs are cases of false positive for Theorem 4.5 with respect

to weak determinism. Suppose for instance a simple model of a single input spare gate connected to its corresponding BE and to a single input SBE, with both basic elements connected to the same RBOX (Figure 5.10). In such a model, actions $\mathtt{asg}_1$ and $\mathtt{rj}_1$ are not confluent in the spare gate model, and $\mathtt{rq}_1$ is spontaneously enabled by $\mathtt{fl}_0$ in the same model. Moreover $\mathtt{rq}_1$ approximately indirectly triggers both $\mathtt{asg}_1$ and $\mathtt{rj}_1$ in the multiplexer model of the first input to such a spare gate. Thus conditions of Theorem 4.5 are not met. Nevertheless, it is the case that after composition such a closed model does not have non-conflent actions, and is thus weakly deterministic.



$$
\begin{aligned}
&(V = \{v_0, v_1, v_2, v_3\} \\
&, i = \{(v_0, []), (v_1, []), (v_2, [v_0, v_1]), (v_3, [v_0, v_1])\} \\
&, si = \{(v_1, [v_2])\} \\
&, l = \{(v_0, \mathsf{be}), (v_1, \mathsf{sbe}), (v_2, \mathsf{sg}), (v_3, \mathsf{rbox})\})
\end{aligned}
$$

Figure 5.10: False positive model for Theo. 4.5

Although as stated, we have not been able to prove weak determinism in the generality of combinations between spare gates and SBEs, we have proved confluence for some particular compositions of this models by means of as simple program in Python. We enumerate this results:

- All combinations of up to 3 spare gates connected to up to 3 SBEs are confluent (examples in Figure 5.11).

- A model with a single spare gate connected to up to 8 SBEs is confluent (Figure 5.12).
- A model with a single SBE shared by up to 8 spare gates is confluent (Figure 5.12).

The state space explosion limits our possibilities to check for larger combinations. The FIG rare event simulation tool (http://dsg.famaf.unc.edu.ar/fig) also supports checking for confluence, and hence can be used to check for further combinations, which can be then safely used as parts of larger models.

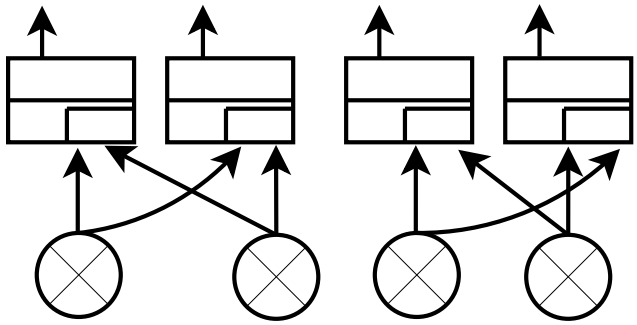


Figure 5.11: Confluent settings.

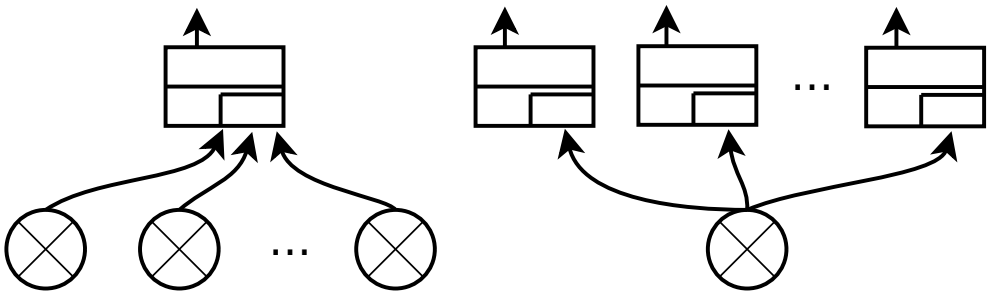


Figure 5.12: Confluent settings.

5.8 RFT Analysis in FIG Simulator

IOSA$_u$ are intended to be a useful and rigorous mean for modeling general continuous distributed systems for the purpose of formal analysis by simulation techniques. In this chapter we present an application example. For this we will first briefly present the FIG simulator, developed at FAMAF-UNC [25, 41] (http://dsg.famaf.unc.edu.ar/fig). We then present a toy example of a cooling system and go through all the steps from defining the model and the properties of interest, to examining the results given by the tool. The FIG simulator is specially tailored to analyzing rare event systems, i.e. systems where the probability of occurrence of the property of interest is very small. Hence we set our RFT example model so that the probability of its top event is very small.

5.8.1 Rare Event Simulation and FIG Simulator

FIG stands for *Finite Improbability Generator* as a homage to Douglas Adam's masterpiece. It is a discrete event simulation tool tailored to rare event properties and it is freely available at (http://dsg.famaf.unc.edu.ar/fig). The tool has been developed by the dependable system group at FAMAF UNC.

The high resilience and dependability required by nowadays system resolve into analyzing properties that fail with an extremely small probability. The complexity of the models make the analysis computationally very demanding. Standard Monte Carlo simulation requires an enormous amount of sampling to acquire a significant confidence level on the estimated probability, in order to compensate for the high variance induced by the rare occurrences of such event. This turns this technique extremely inefficient. Some optimizations over Monte Carlos simulation have been studied in oder to deal with rare events. One of these techniques, the one implemented by FIG, is called *Importance Splitting*, more precisely the so called RESTART method [100, 101]. Importance splitting (IS for short) aims to speed up the occurrence of a rare event without modifications on the

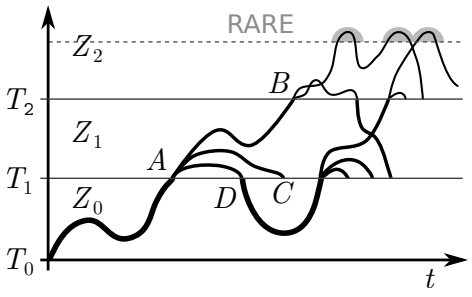


Figure 5.13: Importance Splitting

system dynamics [53, 31]. The general idea of IS is to favor the "promising runs" that approach the rare event by saving the states they visit at certain predefined checkpoints. To do so, IS divides the state space in ascending levels, where ideally as the run goes up from one level to another, the probability of reaching the rare event gets considerably higher. The estimation of the rare probability is obtained as the product of the estimates of the (not so rare) conditional probabilities of moving one level up. The effectiveness of the technique, highly depends on an optimal grouping of the sates. The *importance function* is in charge of such task. Formerly, the task of defining an importance function was given to the engineer who modeled the system or any expert capable of doing so. The FIG tool is capable of building an importance function automatically from the model [31, 28], turning the whole verification process into a push button technique, once the model and the properties of interest have been described. In doing so, FIG ensembles the importance function from local importance function for each component, overcoming the problems emerged from the state space explosion.

5.8.2 The Water Cooling System case study

In this section we present a model of a Water Cooling System is presented. This system is in charge of cooling a high pressure chamber by circulating water around it. The system consists in two replicated subsystems each one composed of a main water fig:water-cooling-system and a switch. Both subsystems share a spare auxiliary water pump as a mechanism to increase their fault tolerance by means of replication. If both subsystems fail, then the colling system fails as no water will be passing around the chamber. In a favorable scenery, if a main water pump fails the auxiliary pump would automatically replace it, while the switch would change the water pipes in order to redirect the water from the auxiliary pipe into the colling section and the subsystem would remain working. Nevertheless some less favorable scenarios can be found, where the auxiliary pump is either broken or already taken by the other subsystem, or the switch is broken and thus the subsystem in question has no way of directing the pipes.

Figure 5.4 shows the graphical description of the RFT model for the Water Cooling System. We are interested in knowing how tolerant to failures this system is, given the corresponding probabilities of failure and repair of its components. A single repair box is connected to all basic elements (by dotted lines in order to ease the understanding of the picture), which are the pumps

and the switches. We study the behaviour of our system in a steady state situation, i.e. the portion of time that it spends on failing states over the amount of time of the whole long run. As explained in [28], let `SYS_FAIL` be the proposition that describes these failing states, i.e. the failing of our top event in our case, then the CSL formula `S(SYS_FAIL)` describes the property to analyze.

Instead of describing the model in a graphical way, we can do so in a textual description language for fault trees. This will ease the compilation to IOSA modules. The syntax for the Kepler Fault Trees Description Language can be found at Appendix B. This language is mainly inspired in Galileo's textual description language [94]. The description in the Kepler language of the Water Cooling model can be found in Figure 5.14. We have developed a compiler that translates Kepler models into IOSA$_u$ models following the theory introduced in this chapter. We have compiled our Kepler model and the obtained IOSA$_u$ model was fed as input into FIG along with the aforementioned property in order to automatically analyze the Water Cooling model without any more user intervention needed.

```
 1 toplevel "FAIL";
 2 "FAIL" and "S1" "S2";
 3 "S1" or "SS1" "PS1";
 4 "S2" or "SS2" "PS2";
 5 "SS1" pand "SW1" "M1";
 6 "PS1" sg "M1" "AUX";
 7 "SS2" pand "SW2" "M2";
 8 "PS2" sg "M2" "AUX";
 9 "M1" exponential(0.01) uniform(1,5);
10 "M2" exponential(0.01) uniform(1,5);
11 "AUX" exponential(0.01) exponential(0.0025) uniform(1,5);
12 "SW1" exponential(0.003) uniform(1,2);
13 "SW2" exponential(0.003) uniform(1,2);
14 "RBOX" priority_rbox "M1" "M2" "SW1" "SW2" "AUX";
```

Figure 5.14: Kepler description of the Water Cooling System.

We wrote a simple program in Python to translate this models writen in Kepler language to a IOSA model in the RFT formalism described in this

Estimated value: **2.39e-08**		
Confidence	Precision	Confidence Interval
80%	1.49e-08	[1.64e-08, 3.14e-08]
85%	2.04e-08	[1.37e-08, 3.41e-08]
90%	2.60e-08	[1.09e-08, 3.69e-08]
99%	4.08e-08	[3.49e-09, 4.43e-08]

Table 5.1

chapter. We ran a Montercarlo analysis on such model using the FIG tool. The property under analysis we chose is `S(SYS_FAIL)`, in order to quantify for the availability of the cooling system. The results delivered by the FIG tool are shown in Table 5.1. The time limit for the experiment was 5 minutes. The obtained value for the top event was 2.39e-08. The experiment was run on a 2.5GHz Intel i5-4200M machine.

5.9 Conclusions

In this chapter, we provided a formal definition of a Repairable version of fault trees. We also embedded it with a formal semantics whose domain are IOSA with urgency. Furthermore, we prove that any RFT model is weak deterministic if the model does not use Spare gates and spare basic elements. Unfortunately, we could not provide such general result when spare systems are used in the model. However, we have proved confluence on several configurations of spare gates and spare basic elements, and hinted on how any possible configuration can be verified for confluency. The chapter ends with a case study that makes use of the FIG tool for the analysis of a rare property. For what we know, this is the first study on rare events simulation analysis of repairable fault trees with general fail and repair probability distributions.

The introduction of the repair model along with general distributions to the analysis of fault trees, turn the models into what Christos Cassandras [32] calls "real world" systems. For this kind of models, usual assumptions and abstractions made in order to ease the analysis have to be dropped, and persuing an analytical solution is not possible any more. Another work that investigates the use of rare event simulation applied to the specific case of Fault Trees is [91]. It includes a repair model, involving complex repair strategies like inspection [90], but it restricts probability measures to Expo-

nential and Erlang.

A possible direction for future work could be introducing Fail dependency gates (like in [16, 91]). This should be done in a way that they do not produce non-determinism. Notice that at the moment no order in the dependent failures has been defined and therefor the non-determinism is intrinsic to the definition. Another line of work would be defining an automatic translation from a graphical modeling tool for fault trees into the IOSA models, in order to automate and ease the engineering of modeling and analysis of RFTs. Furthermore, more complex repair mechanisms could be introduced with the maintenance model of [90].

Chapter 6

Concluding Discussions

A journey from rigorous math to industrial application

In this thesis we presented a framework that allows us to traverse all the way from the essential mathematical foundations, until eventually giving mathematical rigor to the analysis of complex industrial systems. In order to do so we defined a specialization of Stochastic Automata, which we called *Input/Output Stochastic Automata* (IOSA), tailored to analyzing stochastic systems with general distributions by means of discrete event simulation. Analysis through discrete event simulation requires the models to be deterministic. Hence, a considerable part of this thesis works on developing and joining together the mathematical tools to prove that a model in IOSA is deterministic. Finally we use our deterministic modeling language to elaborate a deterministic version of Repairable Fault Trees (RFT), a prominent technique for the analysis of industrial fault tolerant systems. IOSA allows RFT models to endow arbitrary continuous distributions to failure and repair events. RFT models can then be analyzed, for instance, using the Rare Event Simulation tool FIG.

6.1 Achievements

Figure 6.1 graphically resumes the achievements of this thesis. IOSA and IOSA_u build their semantics over NLMP, just as Stochastic Automata (SA) do (arrows going into NLMP). IOSA is a specialization of SA with input and output transitions which happens to be deterministic (arrow from SA to

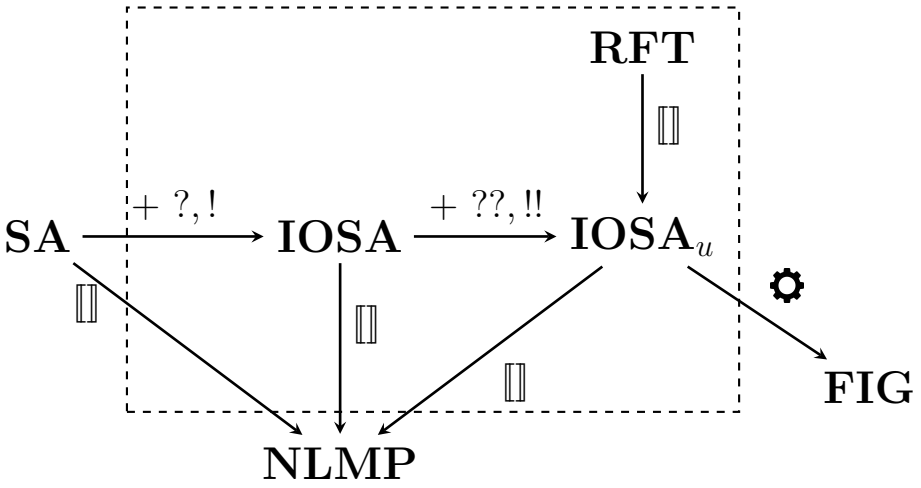


Figure 6.1: Graphical synthesis of the work in this thesis (inside the dashed rectangle).

IOSA). IOSA$_u$ is obtained by extending IOSA with urgent transitions (arrow from IOSA to IOSA$_u$). RFT were formalized in this work and were given a weak deterministic semantics in terms of IOSA$_u$ (arrow from RFT to IOSA$_u$). Finally IOSA$_u$ are the input language for the FIG rare event simulation tool (arrow from IOSA$_u$ to FIG). The dashed rectangle in Figure 6.1 encloses the contribution of this thesis.

We presented the first version of Input/Output Stochastic Automata (IOSA) in Chapter 3. IOSA is compositional and supports arbitrary continuous probability distributions to model the stochastic timed behavior of a system. We defined its semantics in terms of NLMP, imposing a set of conditions in the definition. This conditions finally ensured that a closed IOSA, i.e. a model where all synchronizations have been resolved and no inputs remain, is deterministic. The possibility of using arbitrary continuous distributions, makes IOSA highly suitable for modeling and simulating systems with more realistic results than Markov models such as CTMCs. Moreover, in case the model uses only exponential distributions, the closed IOSA is amenable to numerical analysis since it reduces to a CTMC. The compositional nature of IOSA allows us to concentrate on the clear behavior of the components and the intuitive communication between them, in contrast to error-prone monolithic modeling techniques. Reutilization and maintainability are mentioned as further benefits of compositional modeling. All of this characteristics are

of great advantage when modeling extensive and complex industrial models.

We also extended IOSA with urgent actions. We called this new model IOSA_u. Urgent actions were introduced as a solution to the compositional modeling limitations of the original IOSA, which urged to introduce a delay when synchronizing components. Synchronization through urgent actions drop this condition. Though such extension introduces non-determinism even if the IOSA_u is closed, it does so in a limited manner. We were able to characterize when a IOSA is weak deterministic, which is an important concept since weak deterministic IOSAs are amenable to discrete event simulation. In particular, we showed that closed and confluent IOSAs are weak deterministic. We provided conditions to check compositionally if a closed IOSA is confluent.

Finally we formalized Repairable Fault Trees (RFT) and endowed them with a semantics in terms of IOSA_u. Although Fault Trees are a pervasive formalism in RISK analysis of industrial size systems to the best of our knowledge, there is no other work involving arbitrary continuous distributions for fail and repair times and complex interdependent repairs. Furthermore we have shown that our semantics produces weak deterministic models which are hence amenable for discrete event simulation.

6.2 Future Work

The complexity and necessities of today industrial systems represent an enormous challenge to formal verification. On the one hand, not only qualitative failure analysis is desired, but also quantitative analysis such as costs and performance are usually required. Some future work in this direction would be upgrading IOSA to support costs, rewards, and internal probabilities. In the same direction we could modify our RFT formalism in order to involve maintenance and phase degradation such as [23, 24, 57, 90] suggest. Phase degradation seems to be a simple upgrade which would not affect the weak determinism in the model. On the other hand, especial care has to be taken when upgrading into maintainability, since it involves the dynamics of the interdependent complex repairs. Hence, this extension is more prone to introducing non-determinism into the model.

Proving that the Spare gate model is confluent and thus weak deterministic in the general case is a debt of this work. Spares are used to model redundancy which is of major importance in fault tolerant systems designs.

We emphasized that the intuitive graphical representation of Fault Trees is one of its more appealing characteristics. An interesting work would be developing a graphical interface for the modeling of RFT and a compiler into the IOSA semantics. The FIG simulator could then be used to analyze the compiled model. A graphical interface for modeling and designing the analysis would be an enormous step into the possibilities of delivering a complete tool for industrial use.

Experimentation with RFT in FIG has evidenced the need to enhance the automatic constructions of importance functions for RESTART method. While it was expected a higher efficiency by using RESTART method, the experiments showed that the times to obtain the results did not differ much from those of simple Montecarlo. Analyzing the reasons behind this phenomena might help to build better automatic importance functions for other systems that share similar characteristics with RFTs.

Bibliography

[1] Rajeev Alur and David L. Dill. The theory of timed automata. In J. W. de Bakker, Cornelis Huizing, Willem P. de Roever, and Grzegorz Rozenberg, editors, *Real-Time: Theory in Practice, REX Workshop, Mook, The Netherlands, June 3-7, 1991, Proceedings*, volume 600 of *Lecture Notes in Computer Science*, pages 45–73. Springer, 1991.

[2] Suprasad Amari, Glenn Dill, and Eileen Howald. A new approach to solve dynamic fault trees. In *Reliability and Maintainability Symposium, 2003. Annual*, pages 374–379. IEEE, 2003.

[3] R.B. Ash and C. Doléans-Dade. *Probability and Measure Theory*. Harcourt/Academic Press, 2000.

[4] Franz Baader and Tobias Nipkow. *Term rewriting and all that*. Cambridge University Press, 1998.

[5] Christel Baier, Boudewijn R. Haverkort, Holger Hermanns, and Joost-Pieter Katoen. Model-checking algorithms for continuous-time markov chains. *IEEE Trans. Software Eng.*, 29(6):524–541, 2003.

[6] Christel Baier and Joost-Pieter Katoen. *Principles of model checking*. MIT Press, 2008.

[7] Anis Baklouti, Nga Nguyen, Jean-Yves Choley, Faïda Mhenni, and Abdelfattah Mlika. Free and open source fault tree analysis tools survey. In *2017 Annual IEEE International Systems Conference, SysCon 2017, Montreal, QC, Canada, April 24-27, 2017*, pages 1–8. IEEE, 2017.

[8] Marco Beccuti, Daniele Codetta-Raiteri, Giuliana Franceschinis, and Serge Haddad. Non deterministic repairable fault trees for computing

optimal repair strategy. In *Proceedings of the 3rd international conference on performance evaluation methodologies and tools*, page 56. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008.

[9] Marco Beccuti, Daniele Codetta Raiteri, Giuliana Franceschinis, and Serge Haddad. Non deterministic repairable fault trees for computing optimal repair strategy. In John S. Baras and Costas Courcoubetis, editors, *3rd International ICST Conference on Performance Evaluation Methodologies and Tools, VALUETOOLS 2008, Athens, Greece, October 20-24, 2008*, page 56. ICST/ACM, 2008.

[10] Richard Blute, Josee Desharnais, Abbas Edalat, and Prakash Panangaden. Bisimulation for labelled markov processes. In *Proceedings, 12th Annual IEEE Symposium on Logic in Computer Science, Warsaw, Poland, June 29 - July 2, 1997*, pages 149–158. IEEE Computer Society, 1997.

[11] Andrea Bobbio, Giuliana Franceschinis, Rossano Gaeta, and Luigi Portinale. Parametric fault tree for the dependability analysis of redundant systems and its high-level petri net semantics. *IEEE Trans. Software Eng.*, 29(3):270–287, 2003.

[12] Andrea Bobbio and D Codetta Raiteri. Parametric fault trees with dynamic gates and repair boxes. In *Reliability and Maintainability, 2004 Annual Symposium-RAMS*, pages 459–465. IEEE, 2004.

[13] Jonathan Bogdoll, Luis María Ferrer Fioriti, Arnd Hartmanns, and Holger Hermanns. *Partial Order Methods for Statistical Model Checking and Simulation*, pages 59–74. Springer Berlin Heidelberg, Berlin, Heidelberg, 2011.

[14] Henrik C. Bohnenkamp, Pedro R. D'Argenio, Holger Hermanns, and Joost-Pieter Katoen. MODEST: A compositional modeling formalism for hard and softly timed systems. *IEEE Trans. Software Eng.*, 32(10):812–830, 2006.

[15] Tommaso Bolognesi and Ed Brinksma. Introduction to the ISO specification language LOTOS. *Computer Networks*, 14:25–59, 1987.

[16] Hichem Boudali, Pepijn Crouzen, and Mariëlle Stoelinga. A compositional semantics for dynamic fault trees in terms of interactive markov chains. In Kedar S. Namjoshi, Tomohiro Yoneda, Teruo Higashino, and Yoshio Okamura, editors, *Automated Technology for Verification and Analysis, 5th International Symposium, ATVA 2007, Tokyo, Japan, October 22-25, 2007, Proceedings*, volume 4762 of *Lecture Notes in Computer Science*, pages 441–456. Springer, 2007.

[17] Hichem Boudali, Pepijn Crouzen, and Mariëlle Stoelinga. Dynamic fault tree analysis using input/output interactive markov chains. In *The 37th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2007, 25-28 June 2007, Edinburgh, UK, Proceedings*, pages 708–717. IEEE Computer Society, 2007.

[18] Hichem Boudali, Pepijn Crouzen, and Mariëlle Stoelinga. A rigorous, compositional, and extensible framework for dynamic fault tree analysis. *IEEE Trans. Dependable Sec. Comput.*, 7(2):128–143, 2010.

[19] Hichem Boudali and Joanne Bechta Dugan. A discrete-time bayesian network reliability modeling and analysis framework. *Reliability Engineering & System Safety*, 87(3):337–349, 2005.

[20] Hichem Boudali and Joanne Bechta Dugan. A continuous-time bayesian network reliability modeling, and analysis framework. *IEEE transactions on reliability*, 55(1):86–97, 2006.

[21] Mario Bravetti. *Specification and analysis of stochastic real-time systems*. PhD thesis, PhD thesis, Dottorato di Ricerca in Informatica. Universita di Bologna, Padova, Venezia, 2002.

[22] Mario Bravetti and Pedro R. D'Argenio. *Tutte le Algebre Insieme: Concepts, Discussions and Relations of Stochastic Process Algebras with General Distributions*, pages 44–88. Springer Berlin Heidelberg, Berlin, Heidelberg, 2004.

[23] K. Buchacker. Combining fault trees and petri nets to model safety-critical systems. *High Performance Computing 1999*, pages 439–444, 1999. Cited By :17.

[24] K. Buchacker. Modeling with extended fault trees. In *Proceedings. Fifth IEEE International Symposium on High Assurance Systems Engineering (HASE 2000)*, pages 238–246, Nov 2000.

[25] Carlos E. Budde. *Automation of Importance Splitting Techniques for Rare Event Simulation*. PhD thesis, Universidad Nacional de Córdoba, Argentina, 2017.

[26] Carlos E. Budde. *Automation of Importance Splitting Techniques for Rare Event Simulation*. PhD thesis, Universidad Nacional de Córdoba, 2017.

[27] Carlos E. Budde, Pedro R. D'Argenio, and Holger Hermanns. Rare event simulation with fully automated importance splitting. In Marta Beltrán, William J. Knottenbelt, and Jeremy T. Bradley, editors, *EPEW 2015*, volume 9272 of *LNCS*, pages 275–290. Springer, 2015.

[28] Carlos E. Budde, Pedro R. D'Argenio, and Raúl E. Monti. Compositional construction of importance functions in fully automated importance splitting. In Antonio Puliafito, Kishor S. Trivedi, Bruno Tuffin, Marco Scarpa, Fumio Machida, and Javier Alonso, editors, *Procs. of VALUETOOLS 2016*. ACM, 2017.

[29] Carlos E. Budde, Pedro R. D'Argenio, Pedro Sánchez Terraf, and Nicolás Wolovick. *A Theory for the Semantics of Stochastic and Nondeterministic Continuous Systems*, pages 67–86. Springer Berlin Heidelberg, Berlin, Heidelberg, 2014.

[30] Carlos E. Budde, Christian Dehnert, Ernst Moritz Hahn, Arnd Hartmanns, Sebastian Junges, and Andrea Turrini. JANI: quantitative model and tool interaction. In Axel Legay and Tiziana Margaria, editors, *Tools and Algorithms for the Construction and Analysis of Systems - 23rd International Conference, TACAS 2017, Held as Part of the European Joint Conferences on Theory and Practice of Software, ETAPS 2017, Uppsala, Sweden, April 22-29, 2017, Proceedings, Part II*, volume 10206 of *Lecture Notes in Computer Science*, pages 151–168, 2017.

[31] Carlos E Budde, Pedro R D'Argenio, and Holger Hermanns. Rare event simulation with fully automated importance splitting. In *European Workshop on Performance Engineering*, pages 275–290. Springer, 2015.

[32] Edwin K. P. Chong. Discrete event systems: Modeling and performance analysis - by christos g. cassandras, richard d. irwin, inc., and aksen associates, inc., homewood, il, 1993. xix + 790 pp. ISBN 0-256-11212-6. *Discrete Event Dynamic Systems*, 4(1):113–116, 1994.

[33] Edmund M Clarke, Orna Grumberg, and Doron Peled. *Model checking*. MIT press, 1999.

[34] David Coppit, Kevin J Sullivan, and Joanne Bechta Dugan. Formal semantics of models for computational engineering: A case study on dynamic fault trees. In *Software Reliability Engineering, 2000. ISSRE 2000. Proceedings. 11th International Symposium on*, pages 270–282. IEEE, 2000.

[35] Thomas H. Cormen, Charles E. Leiserson, Ronald L. Rivest, and Clifford Stein. *Introduction to Algorithms, Third Edition*. The MIT Press, 3rd edition, 2009.

[36] Pepijn Crouzen. *Modularity and Determinism in Compositional Markov Models*. PhD thesis, Universität des Saarlandes, Saarbrücken, 2014.

[37] Pedro R. D' Argenio, Joost P. Katoen, and Hendrik Brinksma. *A Stochastic Automata Model and its Algebraic Approach*, pages 1–16. Technical Report. Centre for Telematics and Information Technology (CTIT), Netherlands, 1997.

[38] Vincent Danos, Josée Desharnais, François Laviolette, and Prakash Panangaden. Bisimulation and cocongruence for probabilistic systems. *Information and Computation*, 204(4):503 – 523, 2006. Seventh Workshop on Coalgebraic Methods in Computer Science 2004.

[39] Pedro D'Argenio, Axel Legay, Sean Sedwards, and Louis-Marie Traonouez. Smart sampling for lightweight verification of markov decision processes. *International Journal on Software Tools for Technology Transfer*, 17(4):469–484, Aug 2015.

[40] Pedro R. D'Argenio. *Algebras and Automata for Timed and Stochastic Systems*. PhD thesis, University of Twente, Enschede, 1999.

[41] Pedro R. D'Argenio, Carlos E. Budde, Matias David Lee, Raúl E. Monti, Leonardo Rodríguez, and Nicolás Wolovick. The road from stochastic automata to the simulation of rare events. In Joost-Pieter Katoen, Rom Langerak, and Arend Rensink, editors, *ModelEd, TestEd, TrustEd - Essays Dedicated to Ed Brinksma on the Occasion of His 60th Birthday*, volume 10500 of *Lecture Notes in Computer Science*, pages 276–294. Springer, 2017.

[42] Pedro R. D'Argenio and Joost-Pieter Katoen. A theory of stochastic systems part I: Stochastic automata. *Inf. Comput.*, 203(1):1–38, 2005.

[43] Pedro R. D'Argenio, Joost-Pieter Katoen, and Ed Brinksma. An algebraic approach to the specification of stochastic systems. In David Gries and Willem P. de Roever, editors, *Programming Concepts and Methods, IFIP TC2/WG2.2,2.3 International Conference on Programming Concepts and Methods (PROCOMET '98) 8-12 June 1998, Shelter Island, New York, USA*, volume 125 of *IFIP Conference Proceedings*, pages 126–147. Chapman & Hall, 1998.

[44] Pedro R. D'Argenio and Raúl E. Monti. Input/output stochastic automata with urgency – confluence and weak determinism, 2018. In preparation.

[45] Pedro R. D'Argenio, Pedro Sánchez Terraf, and Nicolás Wolovick. Bisimulations for non-deterministic labelled markov processes. *Mathematical Structures in Computer Science*, 22(1):43–68, 2012.

[46] Pedro R. D'Argenio, Nicolás Wolovick, Pedro Sánchez Terraf, and Pablo Celayes. Nondeterministic labeled markov processes: Bisimulations and logical characterization. In *QEST 2009, Sixth International Conference on the Quantitative Evaluation of Systems, Budapest, Hungary, 13-16 September 2009*, pages 11–20. IEEE Computer Society, 2009.

[47] Josée Desharnais. *Labelled markov processes.* PhD thesis, McGill University, Montréal, 1999.

[48] Josee Desharnais, Abbas Edalat, and Prakash Panangaden. Bisimulation for labelled markov processes. *Inf. Comput.*, 179(2):163–193, 2002.

[49] E. S. DIAMANT and L. M. HEROLD. Thermal performance of cork insulation on minuteman missiles. *Journal of Spacecraft and Rockets*, 3(5):679–684, May 1966.

[50] Ernst-Erich Doberkat and Pedro Sánchez Terraf. Stochastic nondeterminism and effectivity functions. *J. Log. Comput.*, 27(1):357–394, 2017.

[51] J. B. Dugan, S. J. Bavuso, and M. A. Boyd. Dynamic fault-tree models for fault-tolerant computer systems. *IEEE Transactions on Reliability*, 41(3):363–377, Sep 1992.

[52] Christian Eisentraut, Holger Hermanns, Julia Krämer, Andrea Turrini, and Lijun Zhang. Deciding bisimilarities on distributions. In Kaustubh R. Joshi, Markus Siegle, Mariëlle Stoelinga, and Pedro R. D'Argenio, editors, *Quantitative Evaluation of Systems - 10th International Conference, QEST 2013, Buenos Aires, Argentina, August 27-30, 2013. Proceedings*, volume 8054 of *Lecture Notes in Computer Science*, pages 72–88. Springer, 2013.

[53] Marnix Joseph Johann Garvels. *The splitting method in rare event simulation.* PhD thesis, University of Twente, Enschede, Netherlands, 2000.

[54] Daniel Gburek, Christel Baier, and Sascha Klüppelholz. Composition of stochastic transition systems based on spans and couplings. In Ioannis Chatzigiannakis, Michael Mitzenmacher, Yuval Rabani, and Davide Sangiorgi, editors, *43rd International Colloquium on Automata, Languages, and Programming, ICALP 2016, July 11-15, 2016, Rome, Italy*, volume 55 of *LIPIcs*, pages 102:1–102:15. Schloss Dagstuhl - Leibniz-Zentrum fuer Informatik, 2016.

[55] Daochuan Ge, Meng Lin, Yanhua Yang, Ruoxing Zhang, and Qiang Chou. Quantitative analysis of dynamic fault trees using improved sequential binary decision diagrams. *Rel. Eng. & Sys. Safety*, 142:289–299, 2015.

[56] Michèle Giry. A categorical approach to probability theory. In *Categorical aspects of topology and analysis (Ottawa, Ont., 1980)*, volume

915 of *Lecture Notes in Mathematics*, pages 68–85. Springer, Berlin, 1982.

[57] Dennis Guck, Joost-Pieter Katoen, Mariëlle IA Stoelinga, Ted Luiten, and Judi Romijn. Smart railroad maintenance engineering with stochastic model checking. *Proceedings of RAILWAYS. Saxe-Coburg Publications*, pages 950–953, 2014.

[58] Rohit Gulati and Joanne Bechta Dugan. A modular approach for analyzing static and dynamic fault trees. In *Reliability and Maintainability Symposium. 1997 Proceedings, Annual*, pages 57–63. IEEE, 1997.

[59] David F Haasl, NH Roberts, WE Vesely, and FF Goldberg. Fault tree handbook. Technical report, Nuclear Regulatory Commission, Washington, DC (USA). Office of Nuclear Regulatory Research, 1981.

[60] Ernst Moritz Hahn, Arnd Hartmanns, Holger Hermanns, and Joost-Pieter Katoen. A compositional modelling and analysis framework for stochastic hybrid systems. *Formal Methods in System Design*, 43(2):191–232, 2013.

[61] Arnd Hartmanns. *On the analysis of stochastic timed systems*. PhD thesis, Saarland University, 2015.

[62] Thomas A. Henzinger. The theory of hybrid automata. In *Proceedings, 11th Annual IEEE Symposium on Logic in Computer Science, New Brunswick, New Jersey, USA, July 27-30, 1996*, pages 278–292. IEEE Computer Society, 1996.

[63] Thomas A. Henzinger, Xavier Nicollin, Joseph Sifakis, and Sergio Yovine. Symbolic model checking for real-time systems. *Inf. Comput.*, 111(2):193–244, 1994.

[64] Holger Hermanns. *Interactive Markov Chains: The Quest for Quantified Quality*, volume 2428 of *Lecture Notes in Computer Science*. Springer, 2002.

[65] Holger Hermanns, Ulrich Herzog, and Vassilis Mertsiotakis. Stochastic process algebras - between LOTOS and markov chains. *Computer Networks*, 30(9-10):901–924, 1998.

[66] Ulrich Herzog. Formal description, time and performance analysis. A framework. In Theo Härder, Hartmut Wedekind, and Gerhard Zimmermann, editors, *Entwurf und Betrieb verteilter Systeme, Fachtagung des Sonderforschungsbereiche 124 und 182, Dagstuhl, 19.-21. September 1990, Proceedings*, volume 264 of *Informatik-Fachberichte*, pages 172–190. Springer, 1990.

[67] C. A. R. Hoare. *Communicating Sequential Processes*. Prentice-Hall, 1985.

[68] André Joyal, Mogens Nielsen, and Glynn Winskel. Bisimulation from open maps. *Inf. Comput.*, 127(2):164–185, 1996.

[69] Sebastian Junges, Dennis Guck, Joost-Pieter Katoen, and Mariëlle Stoelinga. Uncovering dynamic fault trees. In *46th Annual IEEE/IFIP International Conference on Dependable Systems and Networks, DSN 2016, Toulouse, France, June 28 - July 1, 2016*, pages 299–310. IEEE Computer Society, 2016.

[70] Sebastian Junges, Joost-Pieter Katoen, Mariëlle Stoelinga, and Matthias Volk. One net fits all - A unifying semantics of dynamic fault trees using gspns. In Victor Khomenko and Olivier H. Roux, editors, *Application and Theory of Petri Nets and Concurrency - 39th International Conference, PETRI NETS 2018, Bratislava, Slovakia, June 24-29, 2018, Proceedings*, volume 10877 of *Lecture Notes in Computer Science*, pages 272–293. Springer, 2018.

[71] M. Kwiatkowska, G. Norman, and D. Parker. PRISM: Probabilistic symbolic model checker. In P. Kemper, editor, *Proc. Tools Session of Aachen 2001 International Multiconference on Measurement, Modelling and Evaluation of Computer-Communication Systems*, pages 7–12, September 2001.

[72] Kim Guldstrand Larsen and Arne Skou. Bisimulation through probabilistic testing. *Inf. Comput.*, 94(1):1–28, 1991.

[73] Wen-Shing Lee, Doris L Grosh, Frank A Tillman, and Chang H Lie. Fault tree analysis, methods, and applications - a review. *IEEE transactions on reliability*, 34(3):194–203, 1985.

[74] Axel Legay, Benoît Delahaye, and Saddek Bensalem. Statistical model checking: An overview. In Howard Barringer, Yliès Falcone, Bernd Finkbeiner, Klaus Havelund, Insup Lee, Gordon J. Pace, Grigore Rosu, Oleg Sokolsky, and Nikolai Tillmann, editors, *Runtime Verification - First International Conference, RV 2010, St. Julians, Malta, November 1-4, 2010. Proceedings*, volume 6418 of *Lecture Notes in Computer Science*, pages 122–135. Springer, 2010.

[75] Dong Liu, Weiyan Xing, Chunyuan Zhang, Rui Li, and Haiyan Li. Cut sequence set generation for fault tree analysis. In Yann-Hang Lee, Heung-Nam Kim, Jong Kim, Yongwan Park, Laurence Tianruo Yang, and Sung Won Kim, editors, *Embedded Software and Systems, [Third] International Conference, ICESS 2007, Daegu, Korea, May 14-16, 2007, Proceedings*, volume 4523 of *Lecture Notes in Computer Science*, pages 592–603. Springer, 2007.

[76] Nancy A. Lynch and Mark R. Tuttle. Hierarchical correctness proofs for distributed algorithms. In Fred B. Schneider, editor, *Proceedings of the Sixth Annual ACM Symposium on Principles of Distributed Computing, Vancouver, British Columbia, Canada, August 10-12, 1987*, pages 137–151. ACM, 1987.

[77] Ragavan Manian, David W Coppit, Kevin J Sullivan, and J Bechta Dugan. Bridging the gap between systems and dynamic fault tree models. In *Reliability and Maintainability Symposium, 1999. Proceedings. Annual*, pages 105–111. IEEE, 1999.

[78] Zohar Manna and Amir Pnueli. *The temporal logic of reactive and concurrent systems - specification*. Springer, 1992.

[79] Kenneth L. McMillan. *Symbolic model checking*. Kluwer, 1993.

[80] Guillaume Merle, Jean-Marc Roussel, Jean-Jacques Lesage, and Andrea Bobbio. Probabilistic algebraic analysis of fault trees with priority dynamic gates and repeated events. *IEEE Trans. Reliability*, 59(1):250–261, 2010.

[81] R. Milner. *Communication and Concurrency*. Prentice-Hall, Inc., Upper Saddle River, NJ, USA, 1989.

[82] Michael K. Molloy. Performance analysis using stochastic petri nets. *IEEE Trans. Computers*, 31(9):913–917, 1982.

[83] Tadao Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.

[84] Michael O. Rabin. Probabilistic automata. *Information and Control*, 6(3):230–245, 1963.

[85] Daniele Codetta Raiteri, Giuliana Franceschinis, Mauro Iacono, and Valeria Vittorini. Repairable fault tree for the automatic evaluation of repair policies. In *Dependable Systems and Networks, 2004 International Conference on*, pages 659–668. IEEE, 2004.

[86] Daniele Codetta Raiteri, Mauro Iacono, Giuliana Franceschinis, and Valeria Vittorini. Repairable fault tree for the automatic evaluation of repair policies. In *DSN 2004*, pages 659–668. IEEE Computer Society, 2004.

[87] Antoine Rauzy. Sequence algebra, sequence decision diagrams and dynamic fault trees. *Rel. Eng. & Sys. Safety*, 96(7):785–792, 2011.

[88] Gerardo Rubino and Bruno Tuffin. *Rare Event Simulation Using Monte Carlo Methods*. Wiley Publishing, 2009.

[89] Gerardo Rubino and Bruno Tuffin. *Rare event simulation using Monte Carlo methods*. John Wiley & Sons, 2009.

[90] E. Ruijters, D. Guck, P. Drolenga, and M. Stoelinga. Fault maintenance trees: Reliability centered maintenance via statistical model checking. In *2016 Annual Reliability and Maintainability Symposium (RAMS)*, pages 1–6, Jan 2016.

[91] Enno Ruijters, Daniël Reijsbergen, Pieter-Tjerk de Boer, and Mariëlle Stoelinga. Rare event simulation for dynamic fault trees. In Stefano Tonetta, Erwin Schoitsch, and Friedemann Bitsch, editors, *Computer Safety, Reliability, and Security - 36th International Conference, SAFECOMP 2017, Trento, Italy, September 13-15, 2017, Proceedings*, volume 10488 of *Lecture Notes in Computer Science*, pages 20–35. Springer, 2017.

[92] Enno Ruijters and Mariëlle Stoelinga. Fault tree analysis: A survey of the state-of-the-art in modeling, analysis and tools. *Computer Science Review*, 15:29–62, 2015.

[93] Roberto Segala. *Modeling and verification of randomized distributed real-time systems*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA, 1995.

[94] K. J. Sullivan, J. B. Dugan, and D. Coppit. The galileo fault tree analysis tool. In *Digest of Papers. Twenty-Ninth Annual International Symposium on Fault-Tolerant Computing (Cat. No.99CB36352)*, pages 232–235, June 1999.

[95] Zhihua Tang and J. B. Dugan. Minimal cut set/sequence generation for dynamic fault trees. In *Annual Symposium Reliability and Maintainability, 2004 - RAMS*, pages 207–213, Jan 2004.

[96] Rob J. van Glabbeek, Scott A. Smolka, and Bernhard Steffen. Reactive, generative and stratified models of probabilistic processes. *Inf. Comput.*, 121(1):59–80, 1995.

[97] W. Vesely, J. Dugan, J. Fragola, Minarick, and J. Railsback. Fault Tree Handbook with Aerospace Applications. Handbook, National Aeronautics and Space Administration, Washington, DC, 2002.

[98] Ignacio Viglizzo. *Coalgebras on Measurable Spaces*. PhD thesis, Indiana University, USA, 2005.

[99] Ignacio Dario Viglizzo. Coalgebras on measurable spaces. 2010.

[100] Manuel Villen-Altamirano and Jose Villen-Altamirano. Restart: A method for accelerating rare event simulations. *Analysis*, 3(3), 1991.

[101] Manuel Villén-Altamirano and José Villén-Altamirano. The rare event simulation method RESTART: efficiency analysis and guidelines for its application. In Demetres D. Kouvatsos, editor, *Network Performance Engineering - A Handbook on Convergent Multi-Service Networks and Next Generation Internet*, volume 5233 of *LNCS*, pages 509–547. Springer, 2011.

[102] Nicolás Wolovick. Continuous probability and nondeterminism in labeled transaction systems. Phd, Universidad Nacional de Córdoba, Córdoba, 2012.

[103] Sue-Hwey Wu, Scott A. Smolka, and Eugene W. Stark. Composition and behaviors of probabilistic I/O automata. *Theor. Comput. Sci.*, 176(1-2):1–38, 1997.

[104] Liudong Xing, Akhilesh Shrestha, and Yuanshun Dai. Exact combinatorial reliability analysis of dynamic systems with sequence-dependent failures. *Rel. Eng. & Sys. Safety*, 96(10):1375–1385, 2011.

[105] Wang Yi. Real-time behaviour of asynchronous agents. In Jos C. M. Baeten and Jan Willem Klop, editors, *CONCUR '90, Theories of Concurrency: Unification and Extension, Amsterdam, The Netherlands, August 27-30, 1990, Proceedings*, volume 458 of *Lecture Notes in Computer Science*, pages 502–520. Springer, 1990.

[106] Håkan LS Younes and Reid G Simmons. Probabilistic verification of discrete event systems using acceptance sampling. In *International Conference on Computer Aided Verification*, pages 223–235. Springer, 2002.

Appendix A

IOSA Syntax

The context free grammar from figure A.1 defines the complete IOSA$_u$ symbolic modeling language. Here * stands for "as many times as you want", + for "at least one time", ? means optional, | separates options, and parentheses group productions and elements.

MODEL = (MODULE)+
MODULE = (VARIABLE | ARRAY | CLOCK | TRANSITION)+
VARIABLE = NAME `:` TYPE `init` VALUE `;`
ARRAY = NAME`[`INT`]:` TYPE `init` VALUE `;`
CLOCK = NAME `:` `clock` `;`
TRANSITION = `[` (NAME (`?`|`??`|`!`|`!!`)?)? `]` PRE (`@` NAME)? → POS `;`
PRE = ((NAME = EXPR)(`&` NAME = EXPR)∗)?
POS = (`(` NAME' = EXPR `)`(`&` `(` NAME' = EXPR `)`)∗)?
EXPR = VALUE | NAME | EXPR OP EXPR | `(` EXPR `)` | ! EXPR | DISTR
OP = `|` | `&` | `+` | `-` | `*` | `/` | `=`
NAME = (`a`|`b`|...|`z`|`A`|`B`|...|`Z`)(`a`|`b`|...|`z`|`A`|`B`|...|`Z`|`1`|...|`9`|`_`|`-`)∗
TYPE = `boolean` | `[` INT `..` INT `]`

$$
\begin{aligned}
\text{VALUE} &= \texttt{true} \mid \texttt{false} \mid \text{INT} \\
\text{INT} &= (\texttt{1}|\texttt{2}|...|\texttt{9})(\texttt{0}|\texttt{1}|...|\texttt{9})* \\
\text{FLOAT} &= (\texttt{0}|\texttt{1}|...|\texttt{9}) + (\texttt{.}(\texttt{0}|\texttt{1}|...|\texttt{9})+)? \\
\text{DISTR} &= \texttt{normal(}\text{FLOAT}\texttt{,}\text{FLOAT}\texttt{)} \mid \texttt{exponential(}\text{FLOAT}\texttt{)} \mid \\
&\quad \texttt{uniform(}\text{FLOAT}\texttt{,}\text{FLOAT}\texttt{)} \mid ...
\end{aligned}
$$

Figure A.1: IOSA$_u$ symbolic language grammar.

Appendix B

Kepler Syntax

The following grammar defines the syntax for the Fault Tree Description Language Kepler. The grammar is defined in Parsing Expression Grammar notation (PEG). Productions are described by up case words while any other alphabetic word is a string that goes as it is, as well as other types of words and characters enclosed by simple quotation marks. Parenthesis group productions and slashes separate options. The + symbol indicates the preceding pattern has to be produced at list one time, * indicates one or more time in the same sense, while ? indicates optional.

The top event is written on the top line. Each successive line will describe either a gate, a basic event, or a repair box. A BE starts with its name, then the word be to indicate its a basic event and finally the probability distribution for its fail and repair clocks. A spare be is similar except for that the second probability distribution corresponds to its dormant mode fail clock and a third one corresponds to the failing distribution. There are only seven types of distributions defined, which are the ones currently implemented at FIG rare event simulator. Nevertheless, the language can be simply extended to any other continuous distribution and it will still be possible to compile it into IOSA models. Finally gates are defined by a name, a string determining the type of the gate and a list of its inputs names. Particularly, spare gates and repair boxes require to specify their operational mode, which can be “first come first serve”, or “priority”.

```
1  KEPLER = TOPLEVEL (GATE / MODEGATE / BE / SBE)+
2  TOPLEVEL = toplevel NAME ';'
3  BE = NAME be DIST DIST ';'
4  SBE = NAME sbe DIST DIST DIST ';'
5  GATE = NAME ( and / or / pand / vot / fdep) NAME* ';'
6  MODEGATE = ( sg / rbox ) MODE NAME* ';'
7  INT = 0 / (1/.../9)(0/.../9)*
8  FLOAT = INT+ (. INT+( (e/E) -? INT+)? )?
9  NAME = (a / ... / z / A / ... / Z)+ ( _ / INT / NAME)*
10 MODE = fcfs / priority
11 DIST = exponential '(' FLOAT ')' /
12        normal '(' FLOAT , FLOAT ')' /
13        erlang '(' FLOAT , FLOAT ')'/
14        uniform '(' FLOAT , FLOAT ')'/
15        lognormal '(' FLOAT , FLOAT ')'/
16        weibull '(' FLOAT , FLOAT ')'/
17        rayleigh '(' FLOAT ')'/
18        gamma '(' FLOAT , FLOAT ')'/
```

Figure B.1: FTDL Syntax.