Università
Ca'Foscari
Venezia

Corso di Laurea magistrale
in Informatica - Computer Science

Tesi di Laurea

Ca' Foscari
Dorsoduro 3246
30123 Venezia

Encoding G-Networks into the Stochastic Process Algebra PEPA

Relatore
Ch. Prof.ssa Sabina Rossi

Correlatore
Ch. Prof. Andrea Marin

Laureando
Filippo Cavallin
Matricola 840031

Anno Accademico
2012 / 2013

Contents

Abstract

In my thesis I study models of G-Networks, encoding them into the Stochastic Process Algebra PEPA. Up to now there was the general belief that G-Networks could not be caught by a traditional stochastic process algebra and my thesis demonstrates the opposite.
G-Networks are a specific case in the queues theory and are used to describe, for example, a computer, a network or a communication system. The encoding I propose allows one to analyze the dynamic behavior and then the performance of those networks using the existing tools for PEPA. These analysis are useful in real-life modern systems, whose complexity and size are indeed very large and the corresponding models are huge and complex. With the aid of the stochastic process algebra PEPA, we can apply a compositional approach to perform not only qualitative analysis but also quantitative ones. This approach consists in decomposing the entire system into small and more simple subsystems. The main idea is that:
"The smaller they are, the more easily they can be modeled and consequently the more easily the analysis can be".
In the thesis I will first analyze the current literature on these topics, then model some G-Networks in PEPA proving that my models are coherent with the original analyzed G-Networks. Finally I apply the product form theorem of Harrison to my model and give some guidelines about how a G-Network can be encoded in PEPA.

Chapter 1

Introduction

1.1 Motivation

My thesis was started with the general idea of making models of G-networks using the process algebra PEPA. Until now there was the general belief that G-networks could not be catch by a process algebra framework and the main target of my thesis is in fact to demonstrate the opposite.
If I could model the G-Networks (which are a specific case in the queues theory) then in the future we could do performance modeling on those networks using the tools which are given for PEPA.
The aim of performance modeling is to capture and analyze the dynamic behavior of, for example, a computer, a Network or a communication system. This purpose is easy for very simple systems, but when real-life modern systems are considered, the complexity and size of them are indeed very large, huge and results in complex models. With the aid of a process algebra, and in particular of PEPA, we can adopt a compositional approach. This approach try to decompose the entire system into small and more easy subsystems.
The main idea is that: *"The smaller they are, the more easily they can be modeled and consequently the more easily the analysis can be"*.
The compositional nature of PEPA can provide a lot of benefits for both model-solutions and also model-constructions. The operational semantics provided by PEPA is also used to make and generate an underlying Markov process for any PEPA model as it is explained and demonstrate in the thesis of Hillston [21].
In PEPA, simplifications of models and state space aggregations have been

studied to tackle problems of huge performance models. This is done with notions of equivalence between the entities of the models and they also can be useful in a general performance modeling context.
The strong structural equivalence of PEPA, and its isomorphism, allow one to generate some equational laws. Those laws form the basis for the techniques of model transformation. This equivalence together with abstraction mechanisms for PEPA goes to form the basis of a model simplification technique.
The efforts of this thesis, to represent G-Networks in terms of PEPA, are done in order to apply the already studied and demonstrated quantitative analysis techniques for process algebras terms.
In this way there will be two main advantages in this field:

- Making the analysis on the single components and not in the overall system
- Demonstrating that PEPA (and in general process algebras) is more expressive of what was thought in the past

Very often it happens that in queue systems a lot of different but simply queues cooperate to form a much complex and huge system. For this reason, the modeling of those queues with PEPA, lead to the possibility of making analysis of the stream quantity of them individually. In this way we do not have to make a model of the entire system (which can be often represented only by a n-dimensional structure with a very large complexity), but we can make the models of the simple components of it, which are much easier to deal with and analyze (and in general can be represented by only a 1-2 dimensional structure). From this, the quantitative analysis of the single components leads to the quantitative analysis of the whole system which is then facilitated and simplified and as well as accelerated.
With my study I demonstrated that PEPA is expressive enough to allow one to represent G-Networks. In the end this brings an increase of the coverage of what PEPA can model and analyze.
I hope that this will also contribute in finding connections between queues networks and Process Algebras.

1.2 Personal Contribution

My personal and main contributions, in this field, will be to have found a way to represent G-networks with a process algebra. In particular I worked with the process algebra called PEPA (Performance Evaluation Process Algebra). This process algebra is more suitable for performance modeling with respect to other ones, developed previously [21]. Other process algebras are timed or probabilistic ones, but PEPA embodies a different and new way to work in this field: *the stochastic process algebras*.
Moreover, my work is composed by four main parts:

- First part, **Studying**: I studied the recent and up to now developed bibliography of this two subjects, G-Networks and PEPA.
- Second part, **Analysis and Modeling**: in this part I modeled some general examples of G-Network with the process algebra PEPA.
- Third part, **Product-form Demonstration**: in this part I demonstrated that also from my PEPA representation of a G-Network I could also satisfy the conditions to apply the Product-form theorem and use it.
- Fourth part, **Guidelines Development**: in this last part I formulated and developed the guidelines for making PEPA models out of G-Networks.

In the first part of my thesis, I simply started to study the bibliography that was already developed in the fields of process algebra and in the analysis of the G-Networks. The study of PEPA was focused on the mechanism of construction of models and on its limitation and bounds. In particular I studied the whole semantic and how it represents the various parts of a general model. I studied also the various borderlines of its expressivity and tried to find out new solution to resolve some existing problem. For example, one of them is the problem of representing two actions of same probability but mutually exclusive (this problem will be shown in the second part of the thesis). In the second part I analyzed some example of G-Networks and I have tried to find various model that represent them. Initially I found five possible solutions that could fit the system. Some of them show problems in representing more complicated G-Networks systems and so I discarded them. For some others I would have to develop an extension of PEPA semantic, if I wanted to use them. One of those last solutions (called $\perp$*-actions solution*) could also be expanded without extending PEPA but having an approximation of the

original system.
Luckily there was a solution that, losing a bit of simplicity, still relies entirely on PEPA original semantic and expresses completely the examples taking into account. So I do not have to expand furthermore the language, maintaining the original expressivity of PEPA.
I demonstrated that the model has the same behavior of the original system, they also share the same underlying Markov Process and, subsequently, my model has the product-form, if the original system has it.
The third part is the longest one, because I had to demonstrate that my model (and its Underlying Markov Process) is in product-form. In order to do that, I had to change my model, without changing its behavior. To resolve this task I created the so called *"phantom states"* F. Those states must be always unreachable in every possible state of the system.
The last step of this part was about the balancing of the equations of the model, to satisfy the conditions of product-form theorem. The first attempts does not succeed entirely in the balancing, so I developed a methodology to generate actions (called *Safe-Impossible actions*) which never synchronized in the system in both Input and Output from all states. In this way the balancing was more easily computed.
When the equations were balanced I had to finally check that also the reverse rates were constant. Furthermore, to ensure that they were constant, I created another method to discharge the surplus rate in same cases. This methods relies on a triangular form of safe-impossible actions on three corresponding different processes.
Those methods will be explained in details in the third part of the thesis.
There is a last demonstration in this part, I checked that the modified system still behaves as the original one.
In the last part of my thesis I have to find out some general rules to create new models of PEPA from different cases of G-Networks. The target of those rules is to create some sort of automatic modeling of some recurrent structures of those network in order to facilitate future works and developments.
Modeling others kind of G-Networks with PEPA can be possible, and more easy, with those rules.
My case study and those rules prove that there is a way, not found until my thesis, to model G-Networks with a process algebra. Moreover I showed how PEPA can be more expressive than what it was thought and how G-Networks can be represented using PEPA.

Chapter 2

Introduction - Background

2.1 Background introduction

My thesis is based on various theories and concepts already been developed, for this reason I will provide some background materials and also some definitions.

2.1.1 Performance Modeling

Performance modeling is dedicated to describe, analyze and then optimize the dynamic behavior of a network, a communication system or a queue.
In order to do this, it studies, within and between each components of a system, the flow of data and control information. It is done for the purpose of understand the behavior and identify the aspects of system itself. Those features are very useful in a performance point of view.
Before making a performance analysis of a system, essential characteristics of it are captured, thanks to the use of models, or abstract representations, of the system. The study of performances will usually discover some target characteristics. They are found searching for several alternatives and represented by the parameters of the model and their values. After that the model will be evaluated,with the set of current parameter values, to define the measures of its performance and its behavior.
To do this evaluation, the model may be simulated or there can be the resolution of a set of equations, using some analytical and possibly numerical techniques.

Usually the performance modeling and its analytical models are based and applied to stochastic models and solved analytically. Often the notation, used for the construction of these models, is of the queueing networks. In many of these cases, it is assumed that the underlying stochastic models are Markov processes.
A lot of modern systems have an huge size and a very high complexity, this leads to have very large and complex corresponding models. This is a problem in both the construction and solution of model and, for this reason, there is a growing interest in compositional approaches to performance modeling. The main idea is to decompose the whole system into smaller and less complex subsystems that are more easily to be modeled. The searching for methods for the resolution of big Markov chains, with finite but exceedingly large state spaces, has been one of the major occupation of performance analysis research.
Due to state explosion, standard numerical techniques cannot deal with those models. The main solutions that have been found are:

- Separating the solution into sub-models thanks to some compositional approaches applied to model solution and construction
- Separate modeling of subsystems even if the main model has to be considered as a single entity for the resolution process
- Using some techniques of model simplification applied to the models of subsystem in isolation. However, this do not affect the integrity of the whole model.

In general a large model can be usually reduced to a convenient dimension using systematic and formal manipulations.

2.1.1.1 Queues

G-networks are based on the concept of the queue. A queue in computer science is similar to real queue in human society (e.g. queue at flower shop or waiting to pay at supermarket), hypothetically people are served in the same order of appearance. Queues can be seen as abstract structures in which all the elements are maintained in order within them.
There are usually only two operators: *enqueue* and *dequeue*.
The first add an entity to the tail of collection (as last element) and the second

remove the first entity from the front position. In this way queues are of type First-In-First-Out (FIFO) in which the first element, that arrives in the queue, will be also the first customer to be served and then removes. Similarly once a new costumer is put in the queue, all elements, that were previously added, have to be removed before the new costumer can be served. These two property always hold in normal FIFO-queues and in most of cases the queues are of this type. In my thesis, all the queues types will be normal FIFO-queues.

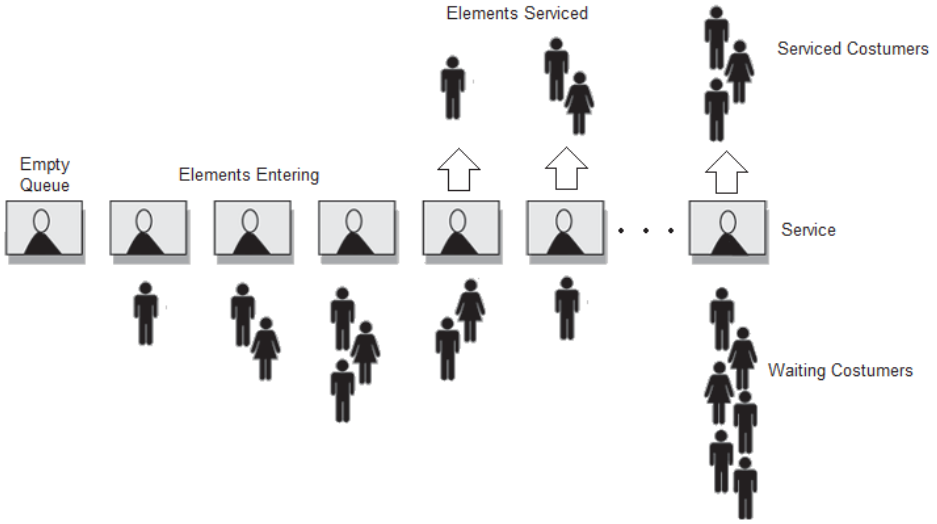

Figure 2.1: Example of the evolution of a queue

2.1.2 Queueing Networks

If queue structures are used in a system, we can speak of queueing network. In the performance modeling there is often the use of those networks.
In this short section I will briefly introduce the main ideas and some terminology to give a general idea of the topic. There is a lot of literature in this field, for more details read [26, 27, 1, 25, 18].

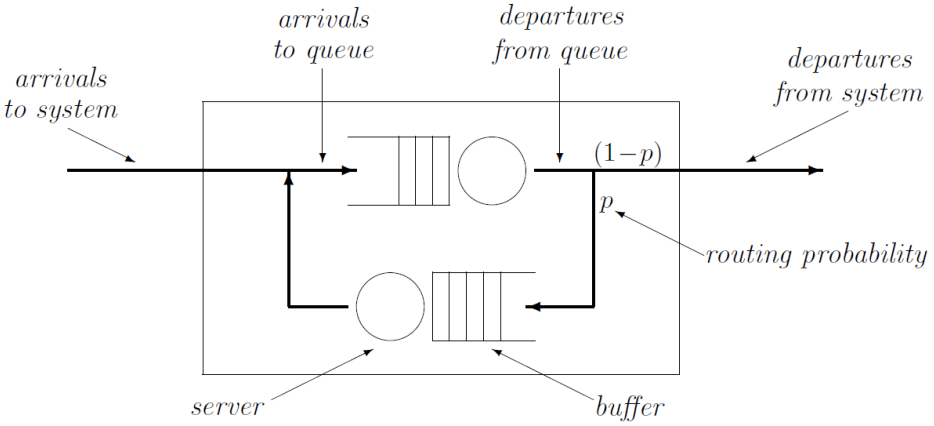

Figure 2.2: Example of a Queueing Network

To define more formally, a queue is composed by:

- an arrival process,
- a buffer,
- one or more servers,
- a service.

The arrival process define the speed rate at which costumers arrive to the queue. The buffer is where costumers await to be serviced, this can be finite and, in case of new arrivals when the queue is full, the new customers can be discarded or redirected to another queue. The servers provide to the customer the service they request and usually are only one. The service represent a resource which the customers must retain, each of them has to keep it for a certain period before leaving the queue.
Moreover, the queue can characterized by five distinct factors:

- the arrival rate,
- the service rate,
- the number of distinct servers,
- the capacity of the buffer,

- the queueing discipline.

We can concisely represent the first four characteristic using the Kendall's annotation for the classification of different queues. Using this notation, a queue can be represented as *A/S/c/m/N*:

- *A*: denotes the rate of arrival process; it is usually M (Markov) if it is exponential distributed, it can be also G, in case of general distribution, or D, if it is a deterministic distribution.
- *S*: represents the service rate and uses the same distribution identifiers of the arrival rate.
- *c*: is the number of servers available to provide the service to the queue.
- *m*: denotes the buffer capacity, it can be infinite by default or have a limited number of seats. If a customer arrives when the buffer is full, it can be lost, blocked, discarded or redirected.
- *N*: represent the current population of customers, it can also be infinite by default.

Also other identifiers may be used for other kinds of distributions, such as *Hk* (hyper-exponential with parameter k). Obviously in the default case, the two last classifiers can be omitted.
The factor of queueing discipline can determines the way in which a server can selects a customer from the queue, who will be then served. The most common discipline is the FIFO (or FCFS, first-come-first-served) in which is served next the most waiting customer. There are many other less frequent discipline, like for example the processor sharing (PS) in which all costumers present at the queue share the service capacity.
A queueing network is a directed graph in which the queues are represented by the nodes. These are often called service centers in this context, and each of them represent a modeled resource in the system. On the other hand, customers represent the various jobs in the whole system. They flow through it and compete for the given resources. Finally, the arcs of the graph are the topology of the system and associated together with the routing probabilities of costumers from one node to others, they determine the various possible paths that customers may take through the network.
In this way, if the demand for resources is high and service rates low, the costumers contention over them will grow up and consequently this will lead to

form a queue of waiting customers.
The typical representation of the state of the system is with the numbers of customers currently occupying each queue. There can be also different types of customers, each ones exhibiting a different behavior within the network. And in that case the state of system is the number of waiting customers of each class. Moreover, there can be classes of customers with fixed or changing patterns of behavior.
Also the networks can belong to different types: closed, open or mixed. This depend on whether a fixed population of customers could arrive from (or depart to) some external environment or simply remain within the system.
There is a large class of queueing networks which have linear and computationally efficient solutions. Although in this class some interesting and important features of the system are excluded, they allow performance analysis witout the use of the underlying Markov process. These solutions, called *product-form solutions*, allow to consider separately the individual queues within a network. Some simple algorithms are based on this, they compute most performance measures looking directly on the parameters of the queueing networks.
Satisfying the hypothesis to use the product-form theorem will be also a part of my thesis.

2.1.3 Process Algebras

Process algebras are part of the mathematical theories. These theories aim to model concurrent systems using their algebra and to provide methods for reasoning about the model behavior and model structure.
The characteristics of a system are its interactions and communications between its active components and the components themselves. One of the main and integral part of the language of these algebras are the compositional reasoning but on the other hand, there isn't a notion of flow or entity within a model, like in queueing networks.
Some examples of process algebras include CCS (Calculus of Communicating Systems), CSP (Communicating Sequential Processes), and ACP (Algebra of Communicating Processes) [29, 22, 3].
In CCS algebra, process (or agents) are the active components of a system. These processes perform actions which represent the possible discrete actions of the analyzed system. Any of these actions can form the links or interactions between neighboring processes or can also be internal to one of those.

Processes may also proceed simultaneously with their internal actions. It is important to notice that their behavior is described as interleaving semantics. Agents can be constructed using the various combinators in the language:

- Prefix: to design a process with an assigned first action
- Choice: if the process can have a choice over various alternatives
- Composition: to model a process with concurrent possibilities

The first two combinators are present also in PEPA, but the last one, composition, is replaced by cooperation.
Many process algebras, as well as CCS, present an operational semantics, using a labeled transition system. Giving this, we can construct a graph or a derivative tree, in which the nodes are language terms and the arcs are transitions. This can be used for analyze the processes and the systems they represent and it forms the basis of the bisimulation style of equivalence.
An agent is characterized by its actions, in this way if two agents perform exactly the same actions, then they are considered to be equivalent. With this style, two forms of equivalence are defined:

- Strong Equivalence: internal actions of a process are considered to be observable
- Weak Equivalence: internal actions of a process are not considered

Correct behavior of systems (in a given specification or in more abstract sense) has been established using widely the models presented with CCS.
This kind of modeling is sometimes called functional or qualitative modeling. Fairness, freedom from deadlock, and other behavioral properties are analyzed unlike the quantitative values found from performance models.
For this reasons, to incorporate also time and probability in process algebras, some extension of them have been made. Then, process algebras without such extensions are sometimes called *pure process algebras*.

2.1.3.1 Timed Extensions of Process Algebras

In pure process algebras, time in processes is often not considered or abstracted away. in this manner there is the assumption that all actions are instantaneous and only relative timing can be represented using traces of

process. The easiest way to incorporate time into an algebra is by making it synchronous. To synchronizing calculations, the presence of a implicit global clock must be assumed, and at each clock tick one action must occur.
Anyway, we need a more sophisticated representation of time to model and represent the real time behavior of a real system. Allowing a process to take a periods of delay, in addition to making actions, can represent time explicitly in a process algebra. Some of those process algebras still assume that actions are instantaneous and natural numbers are taken as the time domain.
Operational semantics of the language contain two different types of transition:

- Action transitions
- Time transitions

The equivalence relation has an additional condition: for all processes must be possible to take any period of delay.
An alternative to this approach can be to associate an absolute time with the completion of actions by processes, these can be called *events*. Moreover, relative time for each action can be specified, or even an interval of time during which an event must be completed.

2.1.3.2 Probabilistic Extensions of Process Algebras

Often there is the need of modeling systems, with process algebras, in which the behavior of their components can contain some uncertainty. This uncertainty of behavior is often not considered or abstracted away, consequently all choices become nondeterministic.
With the replacing of probabilistic choices over nondeterministic ones lead to probabilistic extensions of process algebras. Those extensions allow the quantification of this uncertainty and associate the probability with each possible outcome of a choice.
Also probabilistic process algebras use labeled transition systems in their operational semantics. Those labeled transitions are probabilistic and represent the probabilities associated with each transition.
There is a classification for those systems:

- Reactive Systems: transitions probabilities of processes depend on the environment where the processes are located.

- Generative Systems: probabilities of transitions don't depend on the environment.

In the first case, given that a particular action is performed, there is a probability distribution over the possible derivatives of a process. In the second one, a probability distribution is determined over the possible actions of a process. In some algebras there is also the use of weights to assign probabilities, doing so, prioritized choices are defined.
Probabilistic process algebras are more useful for testing the equivalence between specification and implementation of a system [28]. Two different processes are considered probabilistically equivalent (or bisimilar), if their visible behavior is almost the same (with probability $1 - \epsilon$ and ϵ an arbitrary small number).
Other algebras use pre-orders, which evaluate processes and define some of them probabilistically better than others. This can be use to prove that a implementation of a system is improved with respect to its specification. If there is a probability of 7% of breakdowns in the specification, only an ensured percentage of 6% or less will be satisfactory in the implementation of it.

2.1.4 Process Algebras for Performance Modeling

This section will includes some brief motivations for the studies of applying the process algebras to the performance modeling due to perceived problems of evaluation of performance [21]. There are three distinct problems arisen in performance analysis:

- Integrating Performance Analysis into System Design: it is important to consider timely the performance aspects of system planning but this often does not occur in practice

- Representing Systems as Models: queueing networks have a restricted expressivity, mostly considering real modern developments of computer and telecommunication systems

- Model Tractability: size and complexity of solving models are often too huge and complicate considering many modern systems. In order to manage them, techniques of model simplification and aggregation have been studied

Process algebras can be considered as a tool for design methodology due to their styles of expressing systems and their methods for manipulate the models.

2.1.4.1 Process Algebras for System Design

Designers model systems in a way very similar to the process algebras style of system description. For this reason has been introduced an integration between analysis of performance and methods of design. Languages based on process algebras can be used to do performance models using annotation of already existing systems description for design.
This lead to an involvement for the practice of performance evaluation and the model verification over design methodologies. The use of these formal description languages permit the integration between design process and performance modeling, moreover they allow to make a qualitative (or functional) and quantitative modeling.

2.1.4.2 Process Algebras for Hierarchical Models

Systems are described in a process algebra as a collection of cooperating processes to obtain the system behavior. The paradigm of cooperator (in contrast with terms of operator or operand) fits well for modeling many modern computer systems. Sequential control flow and resource allocation models do not cover fully those systems.
One example can be communications networks components which have their own autonomy and the framework used is one of cooperation. Furthermore, in a process algebra model, each element of the system has equal status and its own individual behaviors and interactions.
Unlike performance modeling, there are also composition and abstraction techniques in process algebras, as well as tools for compositional reasoning. The systematic construction of large models with hierarchical structure are facilitated in this way, using those methods. Moreover they are useful also for solving the underlying Markov process.

2.1.4.3 Process Algebras for Model Simplification and Aggregation

Underlying Markov process or its generator matrix are often the basis of a lot of model simplification and aggregation methods. Unluckily, the size of

the state space can be too large to generate and storage the complete Markov process for very large systems [4].
Nonetheless, process algebra models lead to the introduction of methods for model simplification and aggregation, not based on the underlying stochastic model but on the system description itself. Furthermore, thanks to the compositionality of the process algebra, we can apply those methodologies also to specific parts of the model while the integrity of the model as a whole entity is maintained.
The comparison and manipulation of a model can be done in a formal framework thanks to the formal definition of the process algebra. And finally, the formal definition of equivalence permits the substitution of a model, or a part of it, with another if they have the same observable behavior.

Chapter 3

Introduction - PEPA

3.1 Introducing PEPA

3.1.1 General View

The subject to capture and analyze the dynamic behavior of a network, a communication system or a queue is the performance modeling.
Unfortunately it is difficult to study and analyze many modern systems, due to their size and complexity, resulting in very huge and complex models. To solve this problem, the urge of taking a compositional approach raises. This approach decomposes the whole system into smaller and easier subsystems. They are indeed less complicate and consequently more easily to model. One of the main compositional approach to performance modeling is based on the process algebra named PEPA (Performance Evaluation Process Algebra).
PEPA is a suitably enhanced process algebra and its language has a pronounced compositional nature. This can provide a lot of benefits for the solution of models and in the same way for model construction.
PEPA provides an operational semantics and uses it to create an underlying Markov process. It may be used also as a paradigm for specifying Markov models. This can be done for any PEPA model and the method is well explained and demonstrated in [21].
To deal with the problems of large performance models, some techniques of model simplification and state space aggregation have been proposed. They was developed as notions of equivalence between entities of models and have an intrinsic interest for both process algebra perspective and usefulness in the

context of performance modeling. The basis of these model transformation techniques are formed by the equational laws. These laws are generated and ensured by a strong structural equivalence, the isomorphism. Together with the use of abstraction of PEPA mechanism, this equivalence leads to forming a technique of model simplification. This obviously, providing that certain conditions are satisfied.
These tools may be used to replace one model component by another one which must have the same apparent behavior.
Moreover, developing only a timed and probabilistic process algebra, results to be less suitable for performance modeling than PEPA. This because it is a stochastic process algebra [21]. This process algebra tries also to identify problems of performance evaluation in order to offer a systematic method for modeling also complex systems.
To facilitate the analysis of the whole system, separate components and aspects of it may be considered just individually. They can be also subsequently considered in a more abstract form, as their interactions are developed. Furthermore, the methods of model simplification of PEPA can avoid the generation of the complete state space of the underlying stochastic process.
Finally all those methodologies have been formally defined with its operational semantic, in this way they could lead to automation or machine-assistance for model simplification.

3.1.2 Main Features

One of the main targets of a process algebra able to make performance evaluation is to capture as many features of a normal process algebra as possible while also having characteristics that allow specification of stochastic processes. In this context, the performance evaluation part can be seen as an extension, taking also the normal features of normal process algebras as a basis to be used as a design formalism and thanks to annotations of the design we can develop the performance model.
The following features are considered essential for this process algebra:

- **Parsimony**: elements of the languages must be few in numbers and also simple. With this parsimony the reasonings on the language should be easy and ensure flexibility in the modeling phase. The basic elements of this language are components and activities, corresponding in the underlying stochastic model to states and transitions.

- **Formal Definition**: in the language there is a structured operational semantics, which provides a formal interpretation for each expression. Those rules are the basis of the notions of equivalence, giving a formal way to compare and manipulate both models and components.

- **Compositionality**: the combinator of cooperation is the basis of composition in PEPA. Complementary to it, there are methods of model simplification and aggregation. In this way a part of a model can be simplified in isolation and replaced by a simplified component without compromise the integrity of the whole model.

Quantification of time and uncertainty are main attributes needed for performance evaluation. They are present in PEPA but missing in other process algebras such as CCS. In other algebras time of actions is usually implicit and the models are nondeterministic but it is important to quantify timing behavior and uncertainty for extracting performance measures form performance models.
PEPA associate random variables (representing duration) to all activities, to achieve this. In this way timing behavior of system is represent by delays of each activity in the model. Furthermore, also temporal uncertainty (concerning actions duration) is captured because the duration is a random variable.
Spatial uncertainty (about next events) is obtained because probabilistic choices replace nondeterministic ones. Race conditions between the enabled activities determine probabilities of branchings.
Introducing random variables for all system activities can be considered as annotations of the pure process algebra model.

3.1.3 Language - Informal Description

With PEPA one can describe a system via interactions of its components and actions those elements can engage. The behavior and each part of the system will be corresponded by some components. A queue could be represented by an arrival component and a service component, both of them will interact to form the queue behavior.
Moreover, each component can be atomic or can be composed itself by other components and it is assumed that there is a countable number of possible components, each of them has a behavior defined by its possible activities. Contrarily to other process algebras, in PEPA a random variable representing

duration, with an exponential distributions, is associated to each activity. Every activity has a type (called *action type*) of a countable set A, of all possible types. In case that different activities have the same type, this mean that they are different instances of the same action in the system.
The action type τ, called also unknown type, represent an action (or a sequence of them) that is unknown or not important to identify. This action is considered private to the component in which it occurs and it is not instantaneous, like any other type of action. Nonetheless, different instances of τ don't necessarily represent the same action within the modeled system and they could be represented by their real number parameter of duration.
Generally speaking, this parameter is called the activity rate and must be greater or equal than 0 or $\top$ which mean that it unspecified.
There will be the following conventions about the names of various elements:

- Components are denoted by names starting with large roman letters (P,Q,R,P_i,...).
- Activities are denoted by single small roman letters from the alphabet beginning(a, b, c,...).
- Action types are denoted by small greek letters (α,β,γ,...), or by names starting with a small roman letter ($task$, $service$, $check_i$,...).
- Activity rates are denoted by single roman letters from the alphabet ending (r,s,t,s_j,...). Usually the greek letters μ (for the service rate) and λ (for the arrival rate) are also used.
- Subsets of A are typically denoted by L, K, and M.

In this way, each activity is defined by pairs such as $(\alpha; r)$ where $\alpha \in A$ is the action type and r is the activity rate.
From this, we can say that there exists a set of activities, $Act \in A \times \mathbb{R}^+$, where $\mathbb{R}^+$ is the set of real positive numbers and the symbol $\top$.
There is also some other terminology to introduce:

- System behaves as P: when the component P determines its behavior
- $\mathcal{A}(P)$: current action types of P. The component P can next perform only these action types.
- $\mathcal{A}ct(P)$: multiset of current activities of P. The component P can next perform only these activities.

- Delay period of an enable activity $a = (\alpha, r)$: period of time given by its associated distribution function that is the probability that this activity happens within a period of time t, with $F_a(t) = 1 - e^{rt}$.

- P': is the component which describes the behavior of the system when P completes α for some $\alpha \in \mathcal{A}ct(P)$. P' don't have to be different from P.

- $P \xrightarrow{\alpha} P'$ or $P \xrightarrow{(\alpha,r)} P'$: completion of activity α and the subsequent behavior of the system as P'.

The distinction between action types and activities is the dynamic behavior of a component which depends on the number of instances of each enabled activity. Moreover, $\mathcal{A}(P)$ is a set and $\mathcal{A}ct(P)$ and is a multiset (unless stated otherwise).
The delay period is like a timer set by an activity when becomes enabled and the rate of the activity determines its time. Furthermore,each enabled activities has its own associated timer and the first which finishes implies that its corresponding activity takes place (an external observer could observe the event of an activity of that type). That activity is said to be succeeded or completed and others are considered preempted, or aborted.

3.1.4 Language - Syntax

The primitives of the PEPA language are: components and activities. Remember that activities characterize the behavior of a component and also PEPA has a small set of combinators. Those combinators allow one to construct expressions and terms which can define the behavior of components through interaction between them and activities they can perform. In this way, the behavior of components can be influenced also by the environment in which they are placed.
In PEPA, the syntax of terms is defined as follows:

$$P ::= (\alpha, r).P \mid P + Q \mid P \underset{L}{\bowtie} Q \mid P/L \mid A$$

Now I will give a brief description of names and interpretations of those constructions.

Prefix: $(\alpha, r).P$
The basis of construction of component behavior is the mechanism of Prefix. In this case, the component $(\alpha, r).P$ can perform the activity (α, r), which has action type α and a duration exponentially distributed with parameter r (mean $\frac{1}{r}$). For completing the activity, the time is t, drawn from the distribution. After its completion, the behavior of component will be as component P. If a component reaches the behavior of $(\alpha, r).P$ at some time t', it will complete the action (α, r) in $t' + t$ time. In this way it will become P, enabling all the activities in $\mathcal{A}ct(P)$. If we consider $a = (\alpha, r)$ then we can write the component $(\alpha, r).P$ as $a.P$.
We have to remind also that we assume there is always an underlying implicit resource, which helps the component activities. They are not modeled explicitly but their utilization by the components are represented by the time elapsed before the completion of an activity. Those resources could be time processor, cycles of CPU, some I/O devices or bandwidth of a communication channel, this can depend on which system and at which level it is modeled.

Choice: $P + Q$
A system which can behave both as component P or as component Q, can be represented by the component $P + Q$. This component enables all the current enabled activities of P and all the current enabled activities of Q at the same time.
To be more formal: $\mathcal{A}ct(P + Q) = \mathcal{A}ct(P) \uplus \mathcal{A}ct(Q)$ (where $\uplus$ denotes the multiset of the union) and whatever enabled activity will be completed, it must belong to $\mathcal{A}ct(P)$ or $\mathcal{A}ct(Q)$. It must be in this way, also in the case that P and Q have the same enabled activity since different instances of the same activity are distinguished in PEPA. Doing so, the future behavior will be of P or of Q and it will be distinguished by the first activity to complete. This activity will also discard the other component of the choice.
We have an important thing to remind, **the probability, that both P and Q complete an activity at the same time, is zero**, and this is ensured by the continuous nature of the probability distributions. If we consider that after P completed the activity, it behaves as P' component and similarly Q and Q', then the system will subsequently behave as P' or Q' since $P + Q$ had completed an activity.
Finally, we have to consider that there is always the underlying assumption

that either P and Q compete for the same implicit resource.
In this way, the competition between components is represented by the choice combinator represents.

Cooperation: $P \underset{L}{\bowtie} Q$
The set L is called *cooperation set*, and the interaction between the P and Q components are determined by it. If we consider that $K \neq L$, then almost certainly component $P \underset{L}{\bowtie} Q$ will behave quite differently from the behavior of $P \underset{K}{\bowtie} Q$. This makes the cooperation combinator an indexed family of combinators, a different combinator for each possible set of action types $L \subseteq A$. The action types, on which the components must cooperate or synchronize, are indeed defined by the cooperation set.
Contrary to the choice operator, we assume that in a cooperation, each component has its own implicit resource. Moreover, they independently proceed with any other activities whose types are not in the cooperation set L but those whose action type is in the set L must involve simultaneously both components (and consequently both resources) in an activity of that type. Any cooperation set can't contain any unknown action type τ.
The activities which have types which do not occur in L, are called individual activities of the components and they will proceed unaffected.
On the other hand, shared activities, whose types are in the set L, will only be enabled in $P \underset{L}{\bowtie} Q$, namely when they are enabled in both P and Q. Doing so, one component, which waits for the other component participation, can become blocked and represents situations in the system when, to achieve an action, the components are required to work together.
Generally speaking, each component have to complete some work. This work correspond to their own representation of the action and individual activities of the individual components P and Q are replaced by a new shared activity, formed by the cooperation $P \underset{L}{\bowtie} Q$. The shared rate of this new activity reflects the rate of the slower participant even if the activity will have the same action type as the two contributing ones. Thus this rate (namely the expected duration) of a shared activity, will be greater than or equal to the rates (namely the expected durations) of the corresponding activities in the cooperating components.
In the case of an unspecified rate of an activity in a component, the component is called *passive* with respect to that action type. In other words, the component will not contribute to the work involved, even though its coop-

eration can be required to achieve the completion of an activity of that type. One example can be the role of a channel in a communication system: if we want to transfer a message, the cooperation of the channel is essential, but the transfer don't need any work (i.e. consumption of implicit resource) of the channel itself.
Finally, if the set L is empty, $\underset{L}{\bowtie}$ has the same effect of a parallel composition. This allows the components to proceed concurrently without any kind of interaction between them and this kind of situation will occur quite often, in particular when systems have repeated components. For this reason to represent $P \underset{\varnothing}{\bowtie} Q$, there is the introduction of the more concise notation $P||Q$. The combinator $||$ is called parallel combinator.
We have to notice that also with this syntactic convenience inclusion, there is no expressiveness addition to the language.

Hiding: P/Q
In the component P/Q, all activities, whose types are in the set L, are hidden, this mean that their completion cannot be witnessed and this is the only difference between P/Q and P behavior. Those hidden activities will appear as the unknown type τ and they can represent an internal delay of the component.
The activities that a component can engage individually are not affected by hiding but it affects the possibility to fully witness externally these activities. If a process completes an activity, an external observer can see the type of that completed activity and can also be aware of the length of time from the previous activity completion, i.e. the delay of time in which the activity took place. Rather than, the observer can only witness an hidden activity by its delay and the unknown type τ.
Furthermore those hidden activities cannot be within a cooperation set L with any other component and their action types are no longer accessible outside to both another component or external observer. Anyhow, activities are not affected in their durations even if they are hidden.

Constant: $A \stackrel{def}{=} P$
Assuming that the set of constants is countable, the meaning of constants components is defined by equations such as $A \stackrel{def}{=} P$, in this way the behavior of component P is given to the constant A. Doing so, names can be assigned also to components (behaviors).

Let's suppose that E is a component expression containing also a variable X, the expression $E\{P/X\}$ will then indicate the component made by the replacement of P in every occurrence of X in E. Generally speaking, if we consider an indexed set of variables, then $E\{\tilde{P}/\tilde{X}\}$ is the replacement of an indexed set of variables $\tilde{X}$ by an indexed set of components $\tilde{P}$.

Precedence of the Combinators
In PEPA the precedence of the combinators is also defined and this provides a default interpretation for any kind of expression.
The precedence, from the highest to the lowest, is the following:

1. Hiding
2. Prefix
3. Cooperation
4. Choice

Moreover, forcing alternative parsings or simplifications, to clarify meaning, can be done using brackets.
They can also be used to clarify the meaning of a combination of components. For example, $P \underset{L}{\bowtie} Q \underset{K}{\bowtie} R$ has an unclear scope of the cooperation sets L and K. There are two alternatives:

- $(P \underset{L}{\bowtie} Q) \underset{K}{\bowtie} R$: in this case R, for each action type in $L \setminus K$, can proceed in independently but it has to cooperate with P and Q for any action types in K
- $P \underset{L}{\bowtie} (Q \underset{K}{\bowtie} R)$: here R has to cooperate with P to perform any action types in L but P is free to do independently each action types in $K \setminus L$.

In this way the intended scope of the cooperation set can be delimited by the use of brackets and, in case of missing brackets, it is assumed the left association of cooperation combinator. Considering this and using differing cooperation sets in the cooperation between several different components, we can build layers and levels. Each one of them uses a cooperation combination of just two processes, those components can be in turn formed themselves by cooperations between lower-level components.

One example can be the following:

$$\Big((P_1 \underset{L}{\bowtie} P_2) \underset{K}{\bowtie} P_3\Big) \underset{M}{\bowtie} \Big(P_4 \underset{N}{\bowtie} P_5\Big)$$

This component can be seen as $Q_1 \underset{M}{\bowtie} Q_2$ at the top level. Considering $\equiv$ the syntactic equivalence, then at lower level, $Q_1 \equiv Q_3 \underset{K}{\bowtie} P_3$ and $Q_2 \equiv P_4 \underset{N}{\bowtie} P_5$ and in the lowest level $Q_3 \equiv P_1 \underset{L}{\bowtie} P_2$.
In the lowest level, components which do not contain any cooperation are called *atomic components*, instead in the top level these are called *top-level components*.

Passive Activities: $\top$
In same cases, the components cooperation is not equal and this can represent that one of them can be passive with respect to an action type. This means that each enable activity of that type in the component, has an unspecified activity rate. Those passive activities must be shared with others components which will determine the real rate of this shared activity. A model is called incomplete, if at least one of his passive components is not shared with others or a cooperation set restrict it.
There is also the case in which more than one passive activity type is simultaneously enabled within a component and we must assign a weight to all unspecified activity rates. Those weights have to be natural numbers and they can be used to determine the relative probabilities of the possible outcomes (concerning the activities of that action type).
If we consider this following component:

$$(\alpha, w_1\top).P \; + \; (\alpha, w_2\top).Q$$

It is passive considering the action type α and when that action will be completed, subsequently the component may behave as P with probability $w_1/(w_1+w_2)$ or may behave as Q with probability $w_2/(w_1 + w_2)$.
Furthermore, we can assume that $(\alpha, \top)$ is an abbreviation for $(\alpha, 1\top)$ and that if there is no weights assignation, then the various instances have all equal probabilities to occur.

From this, comparisons and manipulations of unspecified activity rates are defined by the following inequalities and equations:

$$\begin{aligned} r &< w\top && \forall r \in \mathbb{R}^+ \text{ and } \forall w \in \mathbb{N} \\ w_1\top &< w_2\top && \text{if } w_1 < w_2 \quad \forall w_1, w_2 \in \mathbb{N} \\ w_1\top + w_2\top &= (w_1 + w_2)\top && \forall w_1, w_2 \in \mathbb{N} \\ \frac{w_1\top}{w_2\top} &= \frac{w_1}{w_2} && \forall w_1, w_2 \in \mathbb{N} \end{aligned}$$

Apparent Rate: $r_\alpha(P)$
Usually it could be convenient that a single action of the system is represented by more than only one activity in the model at time. Nevertheless the apparent rate of those activities type will always be the same to an external observer of the system or of the model. This because the race condition of the model assure that the α-rate at which an α activity is completed result to be the sum of the rates of all the enabled activities of type α.
For example, there exists systems with the multiple capacity of performing an action, like a queue with multiple servers and n waiting customers with obviously $n > 1$. Now let's consider the apparent rate of *service* action of a PEPA component which enables only one type of *service* activity but with a rate n-times the actual *service* rate of the first presented example. The apparent rate of two service activities would be the same.
From that we can comprehend that the apparent rate, at which action types happen, is very important in the comparison of models with systems and between models themselves.
The formal definition for the **Apparent Rate** [21] is:
"The apparent rate of action of type α in a component P is denoted $r_\alpha(P)$ and it is the sum of the rates of all activities of type α in $\mathcal{A}ct(P)$".

And the formal rules are the following:

$$1.\quad r_\alpha((\beta,r).P) = \begin{cases} r & \text{if } \beta = \alpha \\ 0 & \text{if } \beta \neq \alpha \end{cases}$$

$$2.\quad r_\alpha(P+Q) = r_\alpha(P) + r_\alpha(Q)$$

$$3.\quad r_\alpha(P/L) = \begin{cases} r_\alpha(P) & \text{if } \alpha \notin L \\ 0 & \text{if } \alpha \in L \end{cases}$$

$$4.\quad r_\alpha(P \underset{L}{\bowtie} Q) = \begin{cases} \min(r_\alpha(P), r_\alpha(Q)) & \text{if } \alpha \notin L \\ r_\alpha(P) + r_\alpha(Q) & \text{if } \alpha \in L \end{cases}$$

Remember that an apparent rate can also be unspecified, for example if P is defined like this:

$$P \stackrel{def}{=} (\alpha, w_1\top).P_1 + (\alpha, w_2\top).P_2$$

Then by the previous definitions: $r_\alpha(P)$ will be equal to the sum $r_\alpha(P_1) + r_\alpha(P_2)$, so we can sum $w_1\top + w_2\top$ and we know that this is equal to $(w_1 + w_2)\top$. In conclusion $r_\alpha(P) = (w_1 + w_2)\top$.

Current Action Types: $\mathcal{A}(P)$
The set of action types which are enabled by a component P is called $\mathcal{A}(P)$. This set contains all possible action types which can be seen in the next completion of an activity, when the system behaves as component P.
For any PEPA component, this set (**Set of Current Action Types**) can be constructed by the following definitions:

$$1.\quad \mathcal{A}((\alpha,r).P) = \{\alpha\}$$

$$2.\quad \mathcal{A}(P+Q) = \mathcal{A}(P) + \mathcal{A}(Q)$$

$$3.\quad \mathcal{A}(P/L) = \begin{cases} \mathcal{A}(P) & \text{if } \mathcal{A}(P) \cap L = \varnothing \\ (\mathcal{A}(P) \backslash L) \cup \{\tau\} & \text{if } \mathcal{A}(P) \cap L \neq \varnothing \end{cases}$$

$$4.\quad \mathcal{A}(P \underset{L}{\bowtie} Q) = (\mathcal{A}(P) \backslash L) \cup (\mathcal{A}(Q) \backslash L) \cup (\mathcal{A}(P) \cap \mathcal{A}(Q) \cap L)$$

Current Activities: $\mathcal{A}ct(P)$
The multiset of current activities of P is called Act(P). It will also take a relevant part in the analysis of a component P. These are the enabled activities when the system behaves as component P.
Adopting the following abbreviations:

$$\mathcal{A}ct_{\backslash L}(P) = \{|(\beta, r) \in \mathcal{A}ct(P) | \beta \notin L|\}$$
$$\mathcal{A}ct_{\cap L}(P) = \{|(\beta, r) \in \mathcal{A}ct(P) | \beta \in L|\}$$

Then this multiset (**Activity Multiset**) can be constructed by the following definitions:

1. $\mathcal{A}ct((\alpha, r).P) = \{|(\alpha, r)\}$
2. $\mathcal{A}ct(P + Q) = \mathcal{A}ct(P) \uplus + \mathcal{A}ct(Q)$
3. $\mathcal{A}ct(P/L) = \mathcal{A}ct_{\backslash L}(P) \uplus \{|(\tau, r) | (\alpha, r) \in \mathcal{A}ct_{\cap L}(P)|\}$
4. $\mathcal{A}ct(P \underset{L}{\bowtie} Q) = \mathcal{A}ct_{\backslash L}(P) \uplus \mathcal{A}ct_{\backslash L}(Q) \uplus$
$\uplus \{|(\alpha, r) | \alpha \in L, \exists (\alpha, r_1) \in \mathcal{A}ct_{\cap L}(P), \exists (\alpha, r_2) \in \mathcal{A}ct_{\cap L}(Q),$
and $r = \frac{r_1}{r_\alpha(P)} \frac{r_2}{r_\alpha(Q)} \min(r_\alpha(P), r_\alpha(Q)) |\}$

3.2 Language - Operational Semantics

I present the summary of the formal definition of operational semantics of PEPA in the following figure 3.1.
Those operational rules have the following meaning:
we can infer the transition below the inference line only if the transition(s) above that line can be inferred.
The activities which a component can perform are outlined by the rules, and whenever an activity completes, it cause a transition in the system.
Moreover, there is no explicitly representation of time in these rules but we assume that, for all rules, an activity will take some time to complete. In this way some advance of time is represented by each transition. In these rules, we assume that each activity is (time) homogeneous, this mean that rates and types of activities are time independent with respect of when they occur.

Consequently we also assume that $\mathcal{A}ct(P)$, namely the activity set of a component, is time independent, and it is independent form the time at which it is considered.
The only comment presented is on the third rule for cooperation, which defines shared activities. Furthermore, the apparent rate of a shared actions type in the component $E \underset{L}{\bowtie} F$, when $\alpha \in L$, is chosen such that it is the slowest of the apparent rates of that action type in both E and F. In general ,we assume also that both cooperation components have to complete some work to complete the shared activity, as a reflection of their own version of that activity.
If one component has an unspecified apparent rate, other components will completely determine that rate. Moreover, to represent different possibility of outcome, we can use multiple instances of the same action type in a component. It is also assumed that there is independence between the choice of each shared activity rate, made in order to keep the same outcome probability of each components, and the choice of outcome that each of the cooperating components makes.
Considering for example an instance (α, r_1) of action type $\alpha \in \mathcal{A}ct(E)$ and another instance (α, r_2) of action type $\alpha \in \mathcal{A}ct(F)$. When an α-type activity occurs, the probability, that it is (α, r_1), is $r_1/r_\alpha(E)$ and on the other hand the probability, that it is (α, r_2), is $r_2/r_\alpha(F)$. Instead, if α is a shared type activity in $E \underset{L}{\bowtie} F$, when it occurs, the probability that E and F combine, to form the shared activity, will be $r_1/r_\alpha(E) \times r_2/r_\alpha(F)$.
The activity rate of any activity instance, is the product of the probability of this instance completion (assuming that occur an activity of this type) and the apparent rate in this component of the action type. Consequently there is the following rule:

$$\frac{E \xrightarrow{(\alpha,r_1)} E' \quad F \xrightarrow{(\alpha,r_2)} F'}{E \underset{L}{\bowtie} F \xrightarrow{(\alpha,R)} E' \underset{L}{\bowtie} F'}(\alpha \in L) \quad \text{where } R = r = \frac{r_1}{r_\alpha(E)} \frac{r_2}{r_\alpha(F)} \min(r_\alpha(E), r_\alpha(F))$$

Prefix

$$\frac{}{(\alpha, r).E \xrightarrow{(\alpha,r)} E}$$

Choice

$$\frac{E \xrightarrow{(\alpha,r)} E'}{E + F \xrightarrow{(\alpha,r)} E'} \qquad \frac{F \xrightarrow{(\alpha,r)} F'}{E + F \xrightarrow{(\alpha,r)} F'}$$

Cooperation

$$\frac{E \xrightarrow{(\alpha,r)} E'}{E \underset{L}{\bowtie} F \xrightarrow{(\alpha,r)} E' \underset{L}{\bowtie} F} (\alpha \notin L) \qquad \frac{F \xrightarrow{(\alpha,r)} F'}{E \underset{L}{\bowtie} F \xrightarrow{(\alpha,r)} E \underset{L}{\bowtie} F'} (\alpha \notin L)$$

$$\frac{E \xrightarrow{(\alpha,r_1)} E' \quad F \xrightarrow{(\alpha,r_2)} F'}{E \underset{L}{\bowtie} F \xrightarrow{(\alpha,R)} E' \underset{L}{\bowtie} F'} (\alpha \in L) \quad \text{where } R = \frac{r_1}{r_\alpha(E)} \frac{r_2}{r_\alpha(F)} \min(r_\alpha(E), r_\alpha(F))$$

Hiding

$$\frac{E \xrightarrow{(\alpha,r)} E'}{E/L \xrightarrow{(\alpha,r)} E'/L} (\alpha \notin L) \qquad \frac{E \xrightarrow{(\alpha,r)} E'}{E/L \xrightarrow{(\tau,r)} E'/L} (\alpha \in L)$$

Constant

$$\frac{E \xrightarrow{(\alpha,r)} E'}{A \xrightarrow{(\alpha,r)} E'} (A \stackrel{def}{=} E)$$

Figure 3.1: Operational Semantics of PEPA

Taking those semantic rules, we can define PEPA as a labeled multi-transition system. Generally speaking, a set of states S, a set of transition labels T and a transition relation $\stackrel{t}{\rightarrow} \subseteq S \times S \quad \forall t \in T$ can define a *labeled transition system* $(S, T, \{\stackrel{t}{\rightarrow} | t \in T\})$. Instead, if we consider a multi-transition system, we have to replace the relation with a multi-relation whose instances number of a transition between states is recognized.

In this way PEPA can be seen as a labeled multi-transition system $(\mathcal{C}, \mathcal{A}ct, \{\stackrel{(\alpha,r)}{\longrightarrow} | (\alpha, r) \in \mathcal{A}ct\})$, with $\mathcal{A}ct$ as set of activities, $\mathcal{C}$ as the set of components and rules in figure 3.1 give the multi-relation $\stackrel{(\alpha,r)}{\longrightarrow}$.

3.3 Language - Additional Definitions

In this section I will present some additional definitions for PEPA, resulting from last section. If we consider a graph in which nodes represent the language terms, arc represent the possible transition between them, then operational rules will define how the graph is formed. Considering also that we have already distinguished different instances of the same activity, then the graph will be a multigraph. In this way we will distinguish different instances of an arc between terms.
This underlying derivation graph describes the possible behavior of any component of PEPA. It provides also an useful way to reason about the model behavior.
Finally, those are the formal notions of derivatives:

One-step derivative: *P' is a one-step derivative of P, if* $P \stackrel{(\alpha,r)}{\longrightarrow} P'$

Derivative: *more in general, P' is a derivative of P, if* $P \stackrel{(\alpha_1,r_1)}{\longrightarrow} \ldots \stackrel{(\alpha_n,r_n)}{\longrightarrow} P'$
Those derivatives represent the various states of the labeled multi-transition system. It is usually convenient also to expand those definitions of a component and name each derivative individually. In this way we can define recursively the set of all possible derivatives (behavior) of all PEPA components, in which they can evolve.
Derivate Set: *$ds(C)$ is the notation of the derivative set of a PEPA component, and it is the smallest set of components such that*:

- if $C \stackrel{def}{=} C_0$ then $C_0 \in ds(C)$;
- if $C_i \in ds(C)$ and $\exists a \in \mathcal{A}ct(C_i)$ such that $C_i \stackrel{a}{\longrightarrow} C_j$, then $C_j \in ds(C)$.

Each reachable states of the system is captured by the derivative set of components. In this way we can visualize all possible states of the system and the relationships among them with the transition graph of a system. In fact the *derivation graph* is defined in terms of the derivative set of a system.
Derivation Graph: *the derivation graph $\mathcal{D}(C)$ of a component C and its derivative set $ds(C)$, is the labeled directed multi-graph with set of nodes $ds(C)$ and with multiset of arcs A defined as*:

- *The elements of A given by the set $ds(C) \times ds(C) \times \mathcal{A}ct$;*
- *$\langle C_i, C_j, a\rangle$ occurs in A with the same multiplicity as the number of distinct inference trees which infer $C_i \xrightarrow{a} C_j$.*

We can intuitively define the derivation graph and derivative set of component expressions with E, $ds(E)$ and $\mathcal{D}(E)$.
Leaves of derivation graph are formed by the variables in the expression and the appropriate derivation graph will be attached at that point if a variable is instantiated.
$\vec{\mathcal{A}}(C)$ is the notation of the complete set of action types, used in the derivation graph of a system. This complete set represent all possible action types which can be completed during a component evolution.
Complete Set of Action Types: *the complete set of action type of a component C is*:

$$\vec{\mathcal{A}}(C) = \bigcup_{C_i \in ds(C)} \mathcal{A}(C_i)$$

3.4 The Underlying Stochastic Model

A stochastic process, as a representation of the system, can be built using PEPA and its derivation graphs. Furthermore, the resulting stochastic model is a continuous time Markov process if we assume that the durations of activity are exponentially distributed random variables.
In PEPA there is a technique to solve the process when it is assumed that exists a steady state solution and also there is a relationship between the ergodicity of the Markov process and the structure of the PEPA models.

3.4.1 Markov Process Generation

The generation of the underlying stochastic process we can be based on the derivation graphs, for any finite model of PEPA. We know that in any model there is a component which defines it as its initial node and that its derivation graph is a multigraph. Thus, we have the following characteristics of a derivation graph:

- Initial node
- Other nodes in the graphs represent each subsequent derivative (or component)
- Between the corresponding components, for each possible transition there is an action type and an activity rate which label all arcs between nodes

To build the stochastic process we make the following associations:

- Each node of the graph is associated with a state
- Arcs of the graph define the transitions between states

Remember that in the derivation graph, the number of nodes is finite because we have assumed that also the model is finite. Moreover, in the derivation graph, the sum of all the activity rates, which label the arcs linking two nodes, represents the total transition rate between the corresponding states, since each activity duration is exponentially distributed.
We can summarize it in the following theorem:

Theorem
For any finite PEPA model $C \stackrel{\text{def}}{=} C_0$, if we define the stochastic process $X(t)$, such that $X(t) = C_i$ represents that at time t, the system behaves as component C_i, then $X(t)$ is a Markov process.

The proof of this theorem can be found in [21].

3.4.2 Definitions on the Markov Process Underlying a PEPA Model

There are some notions associated with the Markov Underlying Processes of PEPA Model.

Sojourn Time: In a component C, the sojourn time is an exponentially distributed random variable. Moreover, C enables some activities and the sum of these activity rates represents the parameter in the distribution. Then the expected (i.e. the mean) sojourn time is:

$$\left(\sum_{a \in \mathcal{A}ct(C)} r_a \right)^{-1}$$

Exit Rates: The related notion of the exit rate from C is more convenient to consider. This rate can represent the rate at which an arbitrary activity is completed in the component C (i.e. it does something). The exit rate can also represents the rate at which the system leave the corresponding state of component C for another one.
We denote it as $q(C)$ and its definition is:

$$q(C) = \sum_{a \in \mathcal{A}ct(C)} r_a$$

Transition Rates: The transition rate is the rate at which transitions occur between state C_i and C_j or the system changes from behaving like component C_i to behaving like component C_j. In the derivation graph there are arcs connecting the node corresponding to C_i with the node corresponding to C_j, the sum of the activity rates labeling those arcs is represented by this rate.
We denote as $q(C_i, C_j)$ the transition rate between two components C_i and C_j and so it is:

$$q(C_i, C_j) = \sum_{a \in \mathcal{A}ct(C_i|C_j)} r_a$$

Where:
$\mathcal{A}ct(C_i|C_j) = \{|a \in \mathcal{A}ct(C_i)|C_i \xrightarrow{a} C_j|\}$

Usually the previous multiset contains only a single element. Moreover, if the set of one-step derivative of C_i doesn't contain C_j, then $q(C_i, C_j) = 0$.
The off-diagonal elements of the infinitesimal generator matrix of the Markov process Q are represented by $q(C_i, C_j)$ which can be also denoted with q_{ij}. In this way:

$$\Pr(X(t + \delta t) = C_j | X(t) = C_i) = q(C_i, C_j)\delta t + o(\delta t)$$

With: $i \neq j$

The negative sum of the non-diagonal elements of each row forms the diagonals elements ($q_{ii} = -q(C_i)$).
If exists $\Pi(\cdot)$, the steady state probability distribution for the system, we can compute it solving the following matrix equation:

$$\Pi Q = 0$$

This will be subject to the condition of normalization:

$$\sum(C_i) = 1$$

Conditional Transition Rates: The conditional transition rate is the rate at which, as a result of the completion of an activity of type α, the system changes from behaving like component C_i to behaving like component C_j. In the derivation graph there are arcs connecting the node corresponding to C_i with the node corresponding to C_j, the sum of the activity rates labeling those arcs, which have a label of the action type α, is represented by this rate. We denote as $q(C_i, C_j, \alpha)$ the conditional transition rate between two components C_i and C_j using the action type α.

Conditional Exit Rates: Also this conditional exit rate is consider sometimes. The exit rate represents the rate at which the system leave the corresponding state of component C for another one after the completion of an activity of type α.
Moreover, C enables some activities and the sum of these activity rates of type α represents this conditional exit rate. We denote it as $q(C, \alpha)$.
Furthermore, the apparent rate of α in C is the same as the conditional exit rate of C using activities of type α:

$$q(C, \alpha) = r_\alpha(C)$$

Chapter 4

Introduction - G-Networks

4.1 Introducing G-Networks

In this section I will briefly illustrate the main argument of G-Networks, with some useful concepts.

4.1.1 Introduction

G-Networks enter in the field of queueing theory, in the more general topic of mathematical theory of probability.
G-Network, namely *Generalized Queueing Network* or *Gelenbe Network*, is an open network composed by G-queues. Erol Gelenbe first introduced it as a model for queueing systems and as well as for neural networks, it has specific control functions, which include traffic re-routing and traffic destruction.
The basis of this model is a G-queue, which is composed by:

- *Positive Customers*: they arrive from other queues or from outside the system, they also wait for a service and follow a routing disciplines (like in other conventional network models)

- *Negative Customers*: they arrive from other queues or from outside the system, they don't act as normal costumers but they remove (or kill) another waiting customer of the queue (if the queue is non-empty). They represent the need of remove some traffic in case of network congestion. This include also removing batches of customers

- *Triggers*: they arrive from other queues or from outside the system, they also move other customers and displace them in other queues

A product form solution can be used to compute the stationary distribution of G-networks. Moreover, these networks can be exploited to approximate quite general input-output behaviors because of their nature of universal approximators for continuous and bounded functions [8].

We can briefly resume a G-network usual characteristics with the following draft.
G-network is a network of n interconnected queues and:

- all queues, called *G-Queues* in this network, have a server which service rate is μ_i
- external arrivals, of positive and negative costumers or of triggers and resets, have a rate λ_i
- when a positive costumer (also called normal costumer) is serviced, it can have multiple choices of behavior: to go to another queue remaining a positive costumer; to change its nature and become a negative costumer, a trigger or a reset; or to exit the system. The probabilities of these choices are represented usually as p_{ij}^{+}, p_{ij}^{-} and d_i, respectively.
- the arrival of a positive costumer in a queue, increases the length of the queue by 1.
- the arrival of a negative costumer in a queue, can reduce the length of the queue by some fixed or random number of costumers (if the queue is non-empty and there is at least one positive customer).
- the arrival of a trigger in a queue, moves a positive customer to another queue with some probability
- the arrival of a reset in a queue, resets the queue and set it into its steady-state only if the queue is empty at the arrival time.
- each negative costumer, trigger and reset, will disappear immediately after completion of their actions. For this reason, they are considered *control signals* of the network.

4.1.2 General View

Gelenbe created a new class of queueing networks with two types of customers: positive (or regular) and negative [10, 11, 2].
Interest, in queueing networks and also in the case of single server node, has been increased since his new introduction of the notion of negative customers. With this theory significant progresses have been made in the versatile class of networks analysis. Those progresses have helped in the development of real application systems like manufacturing, communication and indeed computers. Moreover, they have also enriched the queuing theory.
The G-Networks, also known as queueing networks with negative customers, signals, triggers(...), have the characteristics of containing also the presence of negative customers in contrast with the normal positive ones. If those negative costumers arrive in a non-empty queue then there is the witness of a removal of some amount of work from the queue itself. There are various versions of this mechanism but the simplest of them provides that, according to some strategy, a negative customer can delete or kill an ordinary-positive customer.
Some extensions of this model include the removal of a random batch of customers due to the arrival of a negative costumers. This correspond to remove all or some random amount of work from the queue which not always represent an integer number of normal costumers.

Positive Costumers: they are the first type of customers, also known as *regular customers* or *normal costumers*. A server treats those costumers in the usual normal way. The dynamic behavior of the considered network is determined by service and routing disciplines that positive costumers follow.
Negative Costumers: they are the second type of costumers, contrary to the positive ones, they induce (or kill) a positive costumer to immediately leave the node if there is at least one in the queue, for this reason they have the effect of a signal
The modeling of neural networks was the first motivation of the introduction of this queueing network with positive and negative customers. A neuron is represented by a node within this context and excitation/inhibition signals are represented by positive/negative customers routing in the network. Those signals increase or reduce the neuron potential in which they arrive by one unit.
G-Networks provides a basis that unifies queueing and neural networks and

are the result of an extensions of the original network of Gelenbe.
Several extension were made since the introduction of G-networks like cover of triggered movement, batch service networks, multiple classes networks, state-dependent service disciplines, disasters and tandem networks.
G-Networks are a versatile class of networks in which the network behavior is affected in many ways by the negative arrivals of costumers [2]. Some of the different possibilities that have been introduced in this topic are:

- *Individual removal*: the event in which a positive costumer is canceled by a negative customer arrival at the queue. It hasn't effect in case the queue is empty.
- *Batch removal*: the event in which a batch of customers are forced to leave the network by a negative arrival.
- *Disaster*: the event in which a catastrophe in the node is the effect of a negative arrival at the node. This means that all customers are forced to leave the queue immediately and automatically.
- *Triggered movement*: the event in which the instantaneous forcing, of a costumer movement to some other node or a batch of them to leave the network, is triggered by a negative arrival which acts as a signal of that trigger.
- *Random work removal*: the event in which the arrival of a negative customer remove instantaneously a random amount of work from the system.

4.1.3 "Pure" G-Networks

The first G-networks introduced by Gelenbe, (i.e., without any extension) are those with only general positive and negative customers. Considering a Markovian network with n servers (also called nodes), the arrival of customers agrees with an independent Poisson streams of rates, Λ_i for positive costumes and λ_i for negative ones. Costumers are serviced in a server i with times that are exponentially distributed with rate $r(i)$.
If the queue is non-empty when negative customers arrive, its queue length is reduced by one unit. That canceled unit represents a positive customer removal.

When a server completes to service a costumer, it can behave in three ways when leaving the queue i and joins queue j:

- A costumer remain a positive costumer with probability $p^+(i,j)$
- A costumer becomes a negative costumer with probability $p^-(i,j)$
- A costumer leaves the network with probability $d(i)$

In this way, the traffic equations are the following:

$$\lambda^+(i) = \sum_j q_j r(j) p^+(i,j) + \Lambda(i)$$
$$\lambda^-(i) = \sum_j q_j r(j) p^-(i,j) + \lambda(i)$$

where $q_i = \dfrac{\lambda^+(i)}{r(i) + \lambda^-(i)}$ for $i = 1, ..., n$.
These equations are non-linear and this is their main feature. The interesting thing is that if there exists $\{\lambda^+(i), \lambda^-(i)\}$ unique and non-negative solution such that $q_i < 1 \quad \forall i > 0$, then the stationary probability distribution will have the following product form:

$$p(k) = \prod_{i=1}^{n} (1 - q_i) q_i^{k_i}$$

where $k = (k_1, ..., k_n)$ represent the vector of lengths of queue.

4.1.4 Stability

Even if a "pure" G-Network has its product form solution, it has also non-linear traffic equations. This property distances itself from other classical queueing networks. The non-trivial analysis of the stability of the network and of existence of a solution to its traffic equations is the consequence of this. The stability involves the existence of the stationary probability distribution. Thus conditions for general stability in a multiple class G-Network (generalization of the "pure" network) are developed. This introduced method is quite general and it can be used to analyze also extension of G-Network like triggered movement, batch service, G-networks with different service disciplines [16, 13, 9, 23].

The main point is to prove that solutions to the traffic equations exist, subsequently the uniqueness of solution is easy to prove. This is because it involves the stationary solution of a system of Chapman-Kolmogorov equations.
Moreover, **if a positive solution** $p(k)$ **exists then it is unique**.
Considering the following definition of vectors:

$$\Lambda = [\Lambda(i)], \quad \lambda = [\lambda(i)], \quad \lambda^+ = [\lambda^+(i)], \quad \lambda^- = [\lambda^-(i)].$$

And taking into consideration also:

- the matrices, P^+ and P^-, of elements $p^+(i,j)$ and $p^-(i,j)$ respectively
- the diagonal matrix F of the element which represent the fraction of queue entering of positive customers, who survive long enough to be served by server and leave the queue unkilled

We can write the traffic equations in the following way:

$$\lambda^+(I - FP^+) = \Lambda, \qquad \lambda^- = \lambda^+ FP^- + \lambda^+.$$

Now it can be proven that always exists the solution $\{\lambda^+(i), \lambda^-(i)\}$ for $i = 1, ..., n$ [10, 11, 2].

4.1.4.1 Important Property

When dealing with G-networks, the following important property ensure that at equilibrium it doesn't matter from which state of the queue we have started, we will always get the same result we search for. Obviously assuming that conditions for the state equilibrium are satisfied (e.g. it cannot be that positive costumer arrival rate is much higher than the sum of the arrival rate of negative costumer and servicing rate).
The property is this:

When the single node is at equilibrium,
the streams of future arrival of positive and negative customers,
the past departure processes of positive and negative customers
and the current state of the network are **independent**

4.1.5 G-Networks Extensions

In the last years, a lot of different extensions and possibilities have been studied in this topic of G-Networks:

- G-Networks with Triggered Customer Movement: studied by [14, 12, 30, 5]
- G-Networks with Batch Removal: studied by [7, 20]
- Multiple Class G-networks: studied by [17, 24]
- G-Networks with Disasters: studied by [6]
- Tandem G-Networks: studied by [19]

The current literature was made in a short period of time and now I will present the main extension that were made out of G-Networks. This should discern the various kinds of G-Networks and what characterize each type of them.

4.1.5.1 G-Networks with Triggered Customer Movement

For what concern queueing networks in the classical literature, regular costumers are assumed to move into another queue or leave the system after they have been serviced (namely after some amount of service time). They also usually follow a Markovian routing mechanism.
This is the standard idea of costumer transfer and this only occurs after a service completion, however it's possible also that in queueing networks, there are external events occurrences which force and route a costumer to leave its queue.
Gelembe himself introduced this new idea: *G-Networks with positive customers and signals*. A Poisson process of rate $\lambda(i)$ leads the arrivals of signals from outside the queue.
If the queue is a non-empty one, signals can have two different behaviors (and consequently two kinds of consequences in it):

- arrival of a signal implies the movement of a costumer with probability $q(i, j)$ from its queue i to another different queue j

- arrival of a signal trigger a costumer (or a batch of them) to leave the network, from queue i with probability $D(i) = 1 - \sum_j q(i,j)$

With the combination of this description and the routing discipline, after some service is completed, we can comprehend that a signal could be defined by the various movements of regular customers or it could also be exogenous. In both cases, an additional facility, represented by signals, is added in the network for allow others movements of positive customers through the system.
There are other extensions of this type where signals have a similar behavior to triggers. Those are networks with both positive customers and signal emissions. The emission of signal of effective type 0 and the concept of triggers are quite similar each others.
Other extensions, for instance, involve also the use of signals within the system for moving a costumer but considering them like a secondary service process. Those processes have parallel characteristics with respect to regular servicing events. This leads to the superposition of mechanisms of two services and so these signals are no more an extra facility for the movement of regular customers.
Some multiple class G-networks were also developed, with random triggering times. In these systems after the arrival of a signal in the queue, there is some random delay before the trigger of a customer movement. The class of leaving costumer, the queue in which it is and the source of the signal define the random variable which represent that amount of time.
One last example is the incorporation, into probabilities of routing, of some history degree of dependence. This lead to two different models of multiple class of G-Networks:

- when a service is completed or when a negative costumer arrives, the amount of service that has been received defines the various routing probabilities.

- after the completion of a customer servicing time or after its obliged movement due to a signal, the cause of its movement (namely for signal arrival or end of its servicing) and the number of times in which signals interrupted its service define the various routing probabilities.

4.1.5.2 G-Networks with Batch Removal

In G-Networks with batch removal, customers can exit the network in batch mode. The variations assume different batch distributions and also different movements through the system of signals and batches.
The first extension was made with the possibility of batch service when there is a negative arrival. This kind of model keeps assumptions of input and behavior when a regular service ends. On the other hand, when there is a negative arrival on a non-empty queue, the effects are the same of the arrival of a signal. The consequences are two:

- after that a signal arrives, a movement of a normal costumer, with probability $q(i, j)$ from queue i to queue j, is triggered
- with the signal arrival, a service of a batch of normal costumers is completed

The batch size has a random distribution B_i and it only depends on the queue i in which the signal came. In case of a lesser number k_i of costumers in the node with respect to B_i, all and only the available regular customers are removed.
There is also the extension which assumes that signals (or negative arrivals) are secondary servicing processes. In this way both movements, caused by a service completion of normal customers or by the arrival of signals, are the same but they have their own specific parameters of movements triggering.
Could be also that a model has two kinds of different events in a queue i after a service completion:

- type I: with probability p_i, the routing probabilities agree with the a general G-Network.
- type II: with probability $1 - p_i$, a batch of size B_i leaves the network

Another batch removal system has also a symmetric behavior between service completion and negative arrivals time but implies the following operating rules when there is a completion of service in the queue i:

- Costumers in a queue can merge together in a incomplete or partial batch
- if a full batch is moved into a queue with a partial one, the latter is automatically removed

The routing possibilities are the following:

- the full batch joins the queue as a regular and singe costumer with probability $p^+(i,j)$
- the full batch joins the queue as a signal with probability $p^-(i,j)$
- the full batch leaves the system with probability $d(i)$

There can be a further implementation in which both origin and destination queues can define the formation of negative customers batches caused by a triggering movement. For example, the completion of a service in a queue i can force the movement of a departing costumer to another queue j and in the meantime also the simultaneous arrival in the same queue j of a batch of n negative customers with probability $d_{ij}(n)$. These creations of negative batches can also be triggered by the external arrival to node i of a positive customer (with size n and probability $d_{0i}(n)$).

4.1.5.3 Multiple Class G-networks

Another family of alternative is represented by the case of multiple classes of both positive and negative customers. In these extensions, either type of costumers can belong to C different classes. An independent Poisson stream of arrival of positive (with rate Λ_{ic}) and negative (with rate λ_{ic}) customers in queue i characterize each class c. At queue i, c-type positive costumers are served with exponential servicing time distribution r_{ic}. In the case of a normal G-Network $\lambda_{ic} = \lambda_i$, $\forall c$.
In those multi-class G-Networks, the state of system is now defined by:

- vector $k = (k_1, ..., k_n)$
- each component $k_i = (k_{i1}, ..., k_{iC}) \quad \forall i = 1, ..., n$
- each k_{ic} represents the the costumers number of class c in queue i

Negative costumers effects can be described in various ways:

- the negative arrivals of a c-class in the queue i affect only positive costumers of the same type (i.e. the same class c). Thus, if the number of c positive costumers is greater than zero ($k_{ic} > 0$) then the length of the queue is reduced by one unit.

- the negative arrivals trigger a removal of a random class costumer. Thus, if in queue i a negative customer arrives and it is a non-empty queue ($k_i > 0$), then a positive customer of class c is removed with probability $\frac{k_{ic}}{k_i}$.
- the most intricate of the three removal policies: the negative arrivals to empty queues don't have any effect and simply leave the system. The negative arrivals to a non-empty queues select a regular costumer to be removed, according to the service discipline of the current queue. Then, the negative costumer try to kill the selected regular one and succeed with probability $K_{i,m,n}$.
  Where i is the queue, m is the class of the negative costumer and n is the class of the positive costumer.

There are further studies [5, 7, 24] in which several service disciplines have been considered in these G-Networks extensions like:

- first-come-first-served (FIFO or FCFS)
- processor sharing
- last-come-first-served (LCFS or LIFO) with preemptive resume priority
- efforts depending on state of system
- symmetric queues
- arbitrary service times

Moreover, there are the introduction of multiple class also for triggers [15].
Also the routing probabilities can be changed in some models depending on the service times or on the number of service interruptions. Usually those G-networks have n nodes, C classes of positive costumers and only a unique type of negative ones. In this way, node i and class c determine the arrival rates Λ_{ic} whereas only the node i determines the exogenous negative arrival rate λ_i.
We have to remember that different G-Networks usually require various service requirements. Generally speaking, different servicing time distributions belong to different types of costumers but in some cases, to soothe any other condition we assume homogeneous service requirements.
For what concern the deletion choice of costumers, it could be:

- random
- depend on the position of the customer in queue
- the class of the customer
- the service effort offered

All these multiple class G-Networks can bee described also as a particular case of a general state-dependent description. Those more general systems have state-dependent intensities and their signals can be either a trigger or a batch of negative customers. Moreover, two non-negative functions, ϕ and ψ (defined on $\mathbb{Z}^n$), describe the arrival of signals and positive customers.

4.1.5.4 G-Networks with Disasters

In some extensions, G-Networks can also be composed by the presence of a flow of disasters. Contrary to previous G-Networks, in this case there is the possibility of a disaster. These disasters represent an extreme case of costumers exodus. In fact, all the costumers present in the queue, which is affected by the disaster, are automatically removed from the system.
The first studies of a clearing mechanism for single node queues were further developed in more detail with G-Networks with disasters. Virus infections in computer networks and catastrophes in migration processes are some possible applications to those models. All the assumptions of the main (or "pure") G-Network remain valid with one exception for the negative arrivals. In this specific case, a Poisson flow of disasters replace the usual single negative arrivals. These disasters arrive in a queue i with rate $\lambda(i)$ and the rules for arrivals and routing of disasters are the same as the ones for negative arrivals. The only difference is on the behavior, a disaster implies the total destruction of all costumers (and indeed of work) in the destination queue.
In this kind of networks, the following nonlinear system is satisfied by their traffic equations:

$$\lambda^+(i) = \sum_j q_j r(j) p^+(j,i) + \Lambda(i),$$
$$\lambda^-(i) = \sum_j q_j r(j) p^-(j,i) + \lambda(i)$$

where

$$q(i) = \Big\{\lambda^+(i) + \lambda^-(i) + r(i) - (\lambda^+(i)^2 + \lambda^-(i)^2 + r(i)^2 + \\ +2\lambda^+(i)\lambda^-(i) + 2\lambda^-(i)r(i) - 2\lambda^+(i)r(i))^{1/2}\Big\}/2r(i)$$

If either $\lambda^-(i) > 0$ or ($\lambda^-(i) = 0$ and $\lambda^+(i) < r(i)$) are satisfied by the solution of this system for each node i, then the stationary probability distribution will have the product form:

$$p(k) = \prod_i (1 - q_i) q_i^{k_i}$$

4.1.5.5 Tandem G-Networks

Usually the main analysis in G-Networks includes stability, balancing equations and existence of a product form. In the case of tandem G-Networks, the delay times and the response time distribution are the focus of analysis [19]. In a tandem of two Markovian G-Queues, the arrivals of costumers in the queue are determined by Poisson processes. With rate $\Lambda(i)$ for positive costumers and rate $\lambda(i)$ for negatives one (for $i = 1, 2$ where i is the number of queue). Moreover, regular costumers are served with time exponentially distributed with rate $r(i)$. We have to remember that the exponential law has the memoryless property, thus the policies of service and removal, and the queue length distribution are independent.
In this way the limiting probability distribution is the following:

$$p(k_1, k_2) = \prod_{i=1}^{2} (1 - \rho_i) \rho_i^{k_i}$$

where

$$\rho_1 = \frac{\Lambda(1)}{\lambda(1) + r(1)} < 1,$$

$$\rho_2 = \frac{r(1)\rho_1 + \Lambda(2)}{\lambda(2) + r(2)} < 1$$

To analyze the response times distribution (namely end-to-end delays) it is assumed that queueing discipline in the G-Queues pair is FCFS and in case of negative arrival, the removal of costumers starts at the end of the queue (RCE). However this computation of the response time distribution also for simple G-Networks is quite complex due to a phenomenon of overtaking [19].

This event is the overseeing of a positive costumer from its removal: when another regular customers arrives in the queue immediately after the observed customer, positive costumer protects the observed one from removal of negative arrival. In this way the fate of observed costumer depends on the arrival of positive customers after it.
The non-independence in each queue of sojourn times, which on the contrary exist in classical networks without negative arrivals, is explained by this phenomenon.
The probability of a positive costumer of not being deleted jointly the distribution of response time, have the following form (in terms of the Laplace transform):

$$W^*(s) = (1-\rho_1)(1-\rho_2) \times \frac{r(1)}{\Lambda(1)} y_1(s) G(\rho_2, 0, y_1(s), s)$$

Considering a Markovian G-Queue with FCFS and RCE discipline, in this formula the root involved in the sojourn time distribution is represented by $y_1(s)$ and functional equation is satisfied by function G. Dealing with the dependence of sojourn times lead to this complex solution, even so the probability that a costumer is not removed can be reduced to the following simple product formula:

$$W^*(0) = \frac{r(1)}{\lambda(1)+r(1)} \quad \frac{r(2)}{\lambda(2)+r(2)}$$

If we consider the particular case in which $\Lambda(2) = \lambda(2) = 0$, then, as we can expect, the sojourn times are independent in all queues and moreover distributed like considering isolated each corresponding queues.
Finally, in the case of $\Lambda(2) = \lambda(1) = 0$, then costumers in second queue are protected by the costumers departures of first queue and consequently there is still dependence on the sojourn times.

4.1.5.6 The Service Mechanism

Now I will briefly describe the different possible choices of the service mechanism and other relating operating rules for G-Networks. This include also the individual service times. The network design of the whole system is strictly connected with the choice of a service mechanism, this means that we have to strengthen the description of other network components if we relax any assumption on the choice. The final outcome and also the service facilities

are greatly influenced by those compensations. In general, $r(i)$ is the rate of exponential service distribution of time in basic G-Networks at queue i but it can be changed when there is multiple classes of customers and it is modified by the type of costumers and/or the queue considered.
In a multiple class G-Network with tree distinct types of servicing centers which have the following service disciplines:

- First-Come-First-Served (FCFS or FIFO)
- Processor Sharing (PS)
- Last-Come-First-Served with preemptive resume priority (LCFS/PR or LIFO/PR)

Each class of positive customers c can obey a different service rate r_{ic}. In this case, $k = (k_1, ..., k_n)$ is the representation of state of FCFS and LCFS/PR queues and in the queue i, the vector (k_{ij}) represents the state k_i. The length of that vector is the number of positive costumers in the node and the class of costumers in position j is represented by the j-th element of the vector. On the contrary, with PS queues, the state k_i is itself a vector and the number of positive costumers of class c in node i is represented by the c-th element of the vector. In some cases, G-Networks (with general $n + 1$ queues) can label the queue 0 as the "outside" of system and allow the queues to have a negative queue.
Considering the general state-dependent intensities, we can define each node as a source and a costumers queue. The rate of generation of positive customers is the following:

$$\Lambda(i)\frac{\psi(k - e_i)}{\phi(m)}$$

with $k = (k_1, ..., k_n)$ is the vector of lengths of queue, e_0 is a vector of zeros with length n and e_i (with $i > 0$) is a n vector of zeros except of a 1 in the i-th position. Moreover, $\psi(\cdot)$ is a non-negative function in $\mathbb{Z}^n$ and $\phi(\cdot)$ is a positive function in $\mathbb{Z}^n$.
There are also the following definitions:
d_{ij}^+ is the probability of the routing to queue j of the emitted customer, with $0 \leq j \leq n$ and the subsequently addition of one unit in the length of queue j (except for queue 0),

$d_{ij}^{-}(t)$ is the probability of transformation of the emitted costumer into a signal t and its routing to queue j, with $0 \leq j \leq n$ and the negative batch size is denoted by t ($t \geq 0$)
Then the emission embeds the departure process and the choice of probabilities d_{ij}^{+}, $d_{ij}^{-}(t)$ determines triggers and individual (or batch) services dependence.
Let's consider now, mechanisms of service when they are state-dependent, having also symmetric queues and general service times. Furthermore, the system with state-dependent service times is multiple class G-Network. A positive costumer in a queue i, require a servicing time exponentially distributed with rate $r(i)$, so it doesn't depend on the type of class. When k_i costumers are in the queue i the rate of total service is: $\Phi_i(k_i)r(i)$.
A costumer, in position l with $l = 1, ..., k_i$ of the queue i, has an effort service that is a $\gamma_i(l, k_i)$ proportion of the total potential of service. After the completion of the service of that costumer, other customers in positions $l + 1, ..., k_i$ shift to positions $l, ..., k_i - 1$. A positive or signal arrival trigger the reallocation of costumers and this reallocation is determined by two auxiliary probabilities $\delta(l, k_i)$ and $\eta(l, k_i)$. Moreover, standard routing probabilities connect individual queues with one another.
Finally, if we relax the assumption of class homogeneity, assuming that service requests of costumers of class c in queue i have exponential time with rate r_{ic}, then we have to make an assumption that the discipline of service is symmetric ($\gamma_i(l, k_i) = \delta(l, k_i) = \eta(l, k_i)$).
There can be also other models with different assumptions of servicing efforts and the analysis could involve and arbitrary service time distribution $F_{ic}(x)$. In conclusion, even if it depends on the service time distributions, the stationary distribution has also its own product form.

4.2 Applications

There are a lot of different fields of applications, in which complex systems can be modeled by the class provided by G-Networks, such as:

- Computer Networks
- Neural Networks
- Migration Processes

- Telecommunication Systems
- Production Systems
- Maintenance

From the original "pure" G-Networks in which inhibitor signals of neural networks are represented by the negative customers, there has been a wide and huge extension of that first model. All those applications now can cover a lot of different fields.
Now I give a non-exhaustive list of some of those fields of interest in G-Network literature, in order to make a general idea of the potentiality, power and expressivity of this theory:

- Traveling salesman problem
- Problem of minimum graph covering
- Inventory systems
- Computer networks with infection of virus
- Predator-prey models
- Risk processes
- The deletion of transactions in databases
- Synchronization signals in parallel computation
- Artificial textures generation
- Public service systems
- Parallel processing systems
- Steiner trees
- Job inspection
- Processes of migration
- Manufacturing and production systems

- Integrated service digital networks
- Packet switching networks
- Signal modeling in neural networks

Chapter 5

Encoding

5.1 Encoding G-Network with PEPA

In this section of the thesis, I will first analyze a general example of a type of G-Network. I will focus my attention on its behavior that is the main target of my modeling and try to find out the mechanism under the changes of states of the system. In this way I form the basis of the subsequently model phase. Furthermore I search for various aspects of the network:

- dependences between components
- transitions of states in each component
- cooperations between components and their consequences
- limit and particular cases and borderlines (like state 0 or state n)

After that I will try to model it with PEPA language. In this first example I found five possible encodings with PEPA:

- with the use of Exclusive Addition (ExA)
- with the use of $\perp$-Actions ($\perp$)
- with a Partial Index (PI)
- with a Double Index (DI)
- with a Warden Process (WaP)

Each of them present positive and negative characteristics, analyzed in detail in the following sections.
Then I start the analysis of a new example to verify and test the expressiveness and potentiality of each solution. In that phase I found that there are two possible solutions more suitable than others and in those two cases I make their derivation Graphs.
The last phase of this part is the selection of one of the two possible solutions and its comparison with the original G-Network system. In order to do that I will compare each possible state and transition of the original system with those of my model. I will make the comparison using the inductive reasoning.

5.1.1 Analysis and Modeling - Description of the first example

The first example is a **G-Network with trigger** generated after job completion in R_1 that moves a customer from R_3 to R_2. It is represented by the following image:

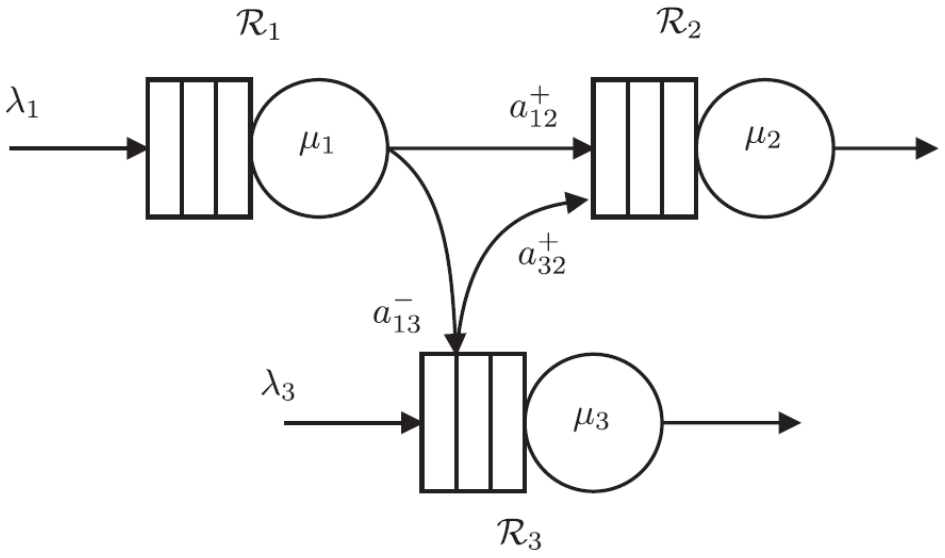

Figure 5.1: G-Network with trigger

In this specific type of G-Network, an homogeneous Poisson processes defines the arrival of positive customers, which arrive from outside the system to queue R_1 and R_3, with rates λ_1 and λ_3, respectively. Moreover, the rates μ_1, μ_2 and μ_3 represent the service times in each queue and so they are exponential random variables, which are independent. A costumer of R_1, after the completion of its service can move to R_2 as a regular costumer (a_{12}^+) or can travel to R_3 and change its nature into a trigger (a_{13}^-). In the second case, when R_3 is a non-empty queue, the length of its queue is reduced by one unit and a positive costumer is added to the queue R_2 (a_{32}^+). If R_3 is empty when the trigger arrives, than nothing happen to R_3. We have to notice that when both R_1 and R_3 are non-empty queues, then the trigger causes the change of states in all of three queues simultaneously.

If we considering the following informal annotations:

(r_1, r_2, r_3) is the state of the system, describing each queue with the number of its costumers
where:
r_1 is the number of positive costumers in R_1,
r_2 is the number of positive costumers in R_2,
r_3 is the number of positive costumers in R_3.

And:

r_i^+ represents the addition of a positive costumer to queue i (with $i = 1, 2, 3$) and is equivalent to $r_i^+ = r_i + 1$
r_i^- represents the completion of a service of a positive costumer and its consequently departure from queue i (with $i = 1, 2, 3$) and is equivalent to $r_i^- = r_i - 1$

An overall informal description of this G-Network behavior can be the following:

1) In R_1 can arrive a positive costumer $\forall r_1 \geq 0$, this implies the change of overall state to (r_1^+, r_2, r_3)

2) In R_3 can arrive a positive costumer $\forall r_3 \geq 0$, this implies the change of overall state to (r_1, r_2, r_3^+)

3) In R_1 a positive costumer can have its service completed $\forall r_1 > 0$ and then it remains a positive costumer and goes to R_2, this implies the change of overall state to (r_1^-, r_2^+, r_3)

4) In R_1 a positive costumer can have its service completed $\forall r_1 > 0$ and then it becomes a trigger and goes to the empty queue R_3 (and it does nothing in R_3), this implies the change of overall state to $(r_1^-, r_2, 0)$

5) In R_1 a positive costumer can have its service completed $\forall r_1 > 0$ and then it becomes a trigger and goes to the non-empty queue R_3 ($\forall r_3 > 0$) forcing a positive costumer in the arrival queue to move in R_2 ($\forall r_2 \geq 0$), this implies the change of overall state to (r_1^-, r_2^+, r_3^-)

6) In R_2 a positive costumer can have its service completed $\forall r_2 > 0$ and then it leaves the system as a positive costumer, this implies the change of overall state to (r_1, r_2^-, r_3)

7) In R_3 a positive costumer can have its service completed $\forall r_3 > 0$ and then it leaves the system as a positive costumer, this implies the change of overall state to (r_1, r_2, r_3^-)

To summarize the arrivals and departures in each state:

- 1: arrival in R_1
- 3,4,5: departure from R_1
- 3,5: arrival in R_2
- 6: departure from R_2

- 2: arrival in R_3
- 5,7: departure from R_3

To summarize relations between state of components and actions:

- $r_1 \geq 0$ can do 1
- $r_1 > 0$ can do 1,3,4,5
- $r_2 \geq 0$ can do 3,5
- $r_2 > 0$ can do 3,5,6
- $r_3 \geq 0$ can do 2,5
- $r_3 > 0$ can do 2,5,7

And actions which change more queues are: 3 and 5

In the following section, I try to capture the behavior of this G-Network with PEPA language.

5.1.2 Analysis and Modeling - Encoding Trials of G-network with Trigger

I will now present my trials with the encoding of the G-network with Trigger using PEPA. I adopt the annotation in which P_i is a generic state of R_1, Q_j of R_2 and R_k of R_3.

5.1.2.1 Exclusive Addition (ExA)

In this first trial I use the concept of exclusive addition. An exclusive addition is an extension of combinators of PEPA and it is very similar to choice. The difference is that the two components of this combinators aren't enabled in the same time but have the same rate, this mean that in $P \oplus Q$ or P is enabled and can perform an action or Q is enabled and can perform an action.
Moreover, in the corresponding underlying Markov process, the arrows exiting from $P \oplus Q$ aren't two but only one, the one of the enabled activity. We have to consider also, and it is quite important, that the rate of the exclusive addition combinator is not the sum of the two identical rates but it is only one at time.
Finally, the concise representation of the exclusive addition combinator in a normal activity of type α is (α_x, r) in which:

- α is the common general name of the exclusive addition
- x is the index which represents one instance of the exclusive addition and can be itself a set of indexes $x = (x_1, ..., x_m)$
- r is the identical rate of all components of exclusive addition

In this way a general exclusive addition $(\alpha_0, r).P_0 \oplus (\alpha_1, r).P_1 \oplus \cdots \oplus (\alpha_n, r).P_n$ can be concisely represented as $(\alpha_x, r).P_x$ with $x = 1, ..., n$.
We have to notice that the name of the action and the arrival state P_j are independent and we can have $(\alpha_x, r).P_y$ with $x = 1, ..., n$ and y a set of numbers $\in \mathbb{N}$.
Furthermore, if the arriving states (namely $P_0, ..., P_n$) are all the same (namely $P = P_0 = ... = P_n$) then the concise description can be $(\alpha_x, r).P$.

The encoding of the G-Network in PEPA is the following:

$$R_1 = \begin{cases} P_0 = (\tau, \lambda_1).P_1 \\ P_n = (\tau, \lambda_1).P_{n+1} + (a, \mu_1 p).P_{n-1} + \underbrace{(a_x, \overbrace{\mu_1(1-p)}^{r}).P_{n-1}}_{(a_0,r).P_{n-1} \oplus (a_1,r).P_{n-1}} & n > 0 \end{cases}$$

$$R_2 = \begin{cases} Q_0 = (a, \top).Q_1 + (a_1, \top).Q_1 \\ Q_n = (a, \top).Q_{n+1} + (a_1, \top).Q_{n+1} + (\tau, \mu_2).Q_{n-1} & n > 0 \end{cases}$$

$$R_3 = \begin{cases} R_0 = (\tau, \lambda_3).R_1 + (a_0, \top).R_0 \\ R_n = (\tau, \lambda_3).R_{n+1} + (\tau, \mu_3).R_{n-1} + (a_1, \top).R_{n-1} & n > 0 \end{cases}$$

With this representation, I can define the following correspondence between informal description and encoding:

- 1: represented by (τ, λ_1) in R_1
- 2: represented by (τ, λ_3) in R_3
- 3: represented by $(a, \mu_1 p)$ in R_1 and by $(a, \top)$ in R_2
- 4: represented by $(a_0, \mu_1(1-p))$ in R_1 and by $(a_0, \top)$ in R_3
- 5: represented by $(a_1, \mu_1(1-p))$ in R_1 and by $(a_1, \top)$ in both R_2 and R_3
- 6: represented by (τ, μ_2) in R_2
- 7: represented by (τ, μ_3) in R_3

5.1.2.2 Solution $\perp$-Actions ($\perp$)

In this second trial I use the concept of $\perp$-actions (or *Bottom Actions*). We can imagine a $\perp$-action as a limit case of a normal action in which its rate is 0. This mean that the activity with 0 rate intuitively takes place immediately after the completion of the previous action, in this way there is no waiting time or duration time of this activity service. In a system, this can represent the change of its state without any work done.

For example, a component $(\alpha, r).(\beta, \bot).(\gamma, \bot).P$ after the completion of α action, it instantly makes also β and γ actions, without any elapsing of time and behaves as P. Furthermore, if there are other components in the system which have actions of type $(\beta, \top)$ or $(\gamma, \top)$ and they have a cooperation in those actions with the first component, then they also perform these actions instantly and change their behavior according to their next activities (i.e. activities after the bottom ones).
Moreover in this encoding, R_1 is aware of the changes of R_3 state and it is informed by R_3 itself, with the use of cooperation combinator in its actions $fill$ and $empty$ which increase or decrease its positive costumers, respectively. In this way R_1 can choose the right action to enable because it knows if R_3 is an empty or a non-empty queues.
This information is stored in R_1 with the apex in the name of its state, thus:

- P_i represents the queue R_1 with i positive costumers and the empty queue R_3 with 0 positive costumers, $\forall i \geq 0$
- P'_i represents the queue R_1 with i positive costumers and the non-empty queue R_3 with the number of positive costumers greater than zero, $\forall i \geq 0$

The encoding of the G-Network in PEPA is the following:

$$R_1 = \begin{cases} P_0 = (\tau, \lambda_1).P_1 + (fill, \top).P'_0 \\ P'_0 = (\tau, \lambda_1).P'_1 + (empty, \top).P_0 \\ P_n = (\tau, \lambda_1).P_{n+1} + (fill, \top).P'_n + (a, \mu_1 p).P_{n-1} + (a_0, \mu_1(1-p)).P_{n-1} & n > 0 \\ P'_n = (\tau, \lambda_1).P'_{n+1} + (empty, \top).P_n + (a, \mu_1 p).P'_{n-1} + (a_1, \mu_1(1-p)).P'_{n-1} & n > 0 \end{cases}$$

$$R_2 = \begin{cases} Q_0 = (a, \top).Q_1 + (a_1, \top).Q_1 \\ Q_n = (a, \top).Q_{n+1} + (a_1, \top).Q_{n+1} + (\tau, \mu_2).Q_{n-1} & n > 0 \end{cases}$$

$$R_3 = \begin{cases} R_0 = (fill, \lambda_3).R_1 + (a_0, \top).R_0 \\ R_1 = (\tau, \lambda_3).R_2 + (empty, \mu_3).R_0 + (a_1, \top).(empty, \bot).R_0 \\ R_n = (\tau, \lambda_3).R_{n+1} + (\tau, \mu_3).R_{n-1} + (a_1, \top).R_{n-1} & n > 1 \end{cases}$$

The only $\bot$-action is in $(a_1, \top).(empty, \bot).R_0$ because R_3 needs to perform both a_1, to satisfy the trigger, and $empty$ to communicate to R_1 that it is become an empty queue.
With this representation, I can define the following correspondence between informal description and encoding:

- 1: represented by (τ, λ_1) in R_1
- 2: represented by $(fill, \lambda_3)$ in empty R_3 (and R_1 is informed with $(fill, \top)$) or by (τ, λ_3) in non-empty R_3
- 3: represented by $(a, \mu_1 p)$ in R_1 and by $(a, \top)$ in R_2
- 4: represented by $(a_0, \mu_1(1-p))$ in R_1 and by $(a_0, \top)$ in R_3
- 5: represented by $(a_1, \mu_1(1-p))$ in R_1 and by $(a_1, \top)$ in both R_2 and R_3
- 6: represented by (τ, μ_2) in R_2
- 7: represented by $(empty, \lambda_3)$ in R_3 if it has only 1 costumer (and R_1 is informed with $(empty, \top)$) or by (τ, μ_3) in R_3 in all other cases

5.1.2.3 Solution Partial Index (PI)

This third trial uses the concept of a partial index. This partial index in R_1 trace partial information about the state of R_3 and also have two different activities with the same rate.
In this sense it is some sort of merging between the first two ideas of encoding. It encapsulates the share of information between R_1 and R_3 in R_1 from the bottom actions and, from exclusive addition, has two activity with the same rate, even if those two activity aren't logically part of the same "family" of activities. We can see this similarity as a limit case of the exclusive addition, in which we take only a main case (a_1) and a limit case (a_{empty}). Here we have two cases anyway, but if we had three cases we would only choose two of them. Furthermore a_{empty} could represent the fusion of $(a_1, \top).(empty, \bot)$ in a single action with the rate $\mu_1(1-p)$ of a_1 in R_1 and the rate $\top$ in R_3.
This is coherent with my vision of $\bot$-action in which the sums behave as:

- $\bot + \top = 0 + \top = \top$
- $\bot + \mu_1(1-p) = 0 + \mu_1(1-p) = \mu_1(1-p)$

Like in the previous encoding, R_1 is aware of the changes of R_3 state and it is informed by R_3 itself, with the use of cooperation combinator in its actions: $fill$ increases its positive costumers and $empty$ or a_{empty} decrease them. In particular those types of actions mark the border of R_3 state between empty and non-empty queue. Thus, R_1 can choose the right action to do because it knows if R_3 is an empty or a non-empty queues but still have uncertainty in the precise state of R_3: it doesn't know if the next decrease of costumers of R_3 will change its state to empty queue.
Anyway, like before, this information is stored in R_1 with the apex in the name of its state, thus:

- P_i represents the queue R_1 with i positive costumers and the empty queue R_3 with 0 positive costumers, $\forall i \geq 0$
- P'_i represents the queue R_1 with i positive costumers and the non-empty queue R_3 with the number of positive costumers greater than zero, $\forall i \geq 0$

The encoding of the G-Network in PEPA is the following:

$$
R_1 = \begin{cases}
P_0 = (\tau, \lambda_1).P_1 + (fill, \top).P'_0 & \\
P'_0 = (\tau, \lambda_1).P'_1 + (empty, \top).P_0 & \\
P_n = (\tau, \lambda_1).P_{n+1} + (fill, \top).P'_n + (a, \mu_1 p).P_{n-1} + (a_0, \mu_1(1-p)).P_{n-1} & n > 0 \\
P'_n = (\tau, \lambda_1).P'_{n+1} + (empty, \top).P_n + (a, \mu_1 p).P^m_{n-1} + (a_1, \mu_1(1-p)).P'_{n-1} + & \\
\qquad +(a_{empty}, \mu_1(1-p)).P_{n-1} & n > 0
\end{cases}
$$

$$
R_2 = \begin{cases}
Q_0 = (a, \top).Q_1 + (a_1, \top).Q_1 + (a_{empty}, \top).Q_1 & \\
Q_n = (a, \top).Q_{n+1} + (a_1, \top).Q_{n+1} + (a_{empty}, \top).Q_{n+1} + (\tau, \mu_2).Q_{n-1} & n > 0
\end{cases}
$$

$$
R_3 = \begin{cases}
R_0 = (fill, \lambda_3).R_1 + (a_0, \top).R_0 & \\
R_1 = (\tau, \lambda_3).R_2 + (empty, \mu_3).R_0 + (a_{empty}, \top).R_0 & \\
R_n = (\tau, \lambda_3).R_{n+1} + (\tau, \mu_3).R_{n-1} + (a_1, \top).R_{n-1} & n > 1
\end{cases}
$$

As usual, with this representation, I can define the following correspondence between informal description and encoding:

- 1: represented by (τ, λ_1) in R_1
- 2: represented by $(fill, \lambda_3)$ in empty R_3 (and R_1 is informed with $(fill, \top)$) or by (τ, λ_3) in non-empty R_3
- 3: represented by $(a, \mu_1 p)$ in R_1 and by $(a, \top)$ in R_2
- 4: represented by $(a_0, \mu_1(1-p))$ in R_1 and by $(a_0, \top)$ in R_3
- 5: represented by $(a_1, \mu_1(1-p))$ in R_1 and by $(a_1, \top)$ in both R_2 and R_3 if R_3 has only 1 costumer or by $(a_{empty}, \mu_1(1-p))$ in R_1 and $(a_{empty}, \top)$ in both R_2 and R_3 in all other cases
- 6: represented by (τ, μ_2) in R_2
- 7: represented by $(empty, \lambda_3)$ in R_3 if it has only 1 costumer (and R_1 is informed with $(empty, \top)$) or by (τ, μ_3) in R_3 in all other cases

5.1.2.4 Solution Double Index (DI)

This fourth trial uses the concept of a double index. Unlike the previous examples, this double index in R_1 trace fully the information about the state of R_3.
This increases the "dependence" of R_1 with respect to R_3 but completely eliminates any possible uncertainty of its possible choices regarding a_0 and a_1. R_1 knows exactly how many positive costumers are in R_3 so it can decide to perform a_0 in case of empty R_3 or a_1 if R_3 has costumers. Furthermore, R_1 knows also if the departure of the following costumer in R_3 will leave R_3 empty or not.
In this way we don't need neither exclusive addition nor double actions (like bottom actions or a_{empty}). Thus, if in one part there is a growing in dependence of R_1 from R_3 and the increase of the space of states, on the other side there is a simplification in the complexity of actions and their correspondence with the behavior of the original system and also in the whole encoding of the system.
Moreover, in previous encoding the space of state for R_1 is $n * 2$ where n is the number of its costumer and 2 represents the possible encapsulated states of R_3 (empty or non-empty); in this case the space is $n * m$ where n is the number of costumers of R_1 and m is the number of costumers in R_3.
Then similarly to previous encodings, R_1 is aware of all changes of R_3 and it is informed by R_3 itself, with the use of cooperation combinator in its actions: $fill$ increases its positive costumers and $empty$ decreases them. In particular those types of actions help R_1 to keep track of the number of costumers in R_3. This information is stored in R_1 with the apex number in the name of its state, thus:

- P_i represents the queue R_1 with i positive costumers and the empty queue R_3 with 0 positive costumers, $\forall i \geq 0$

- P_i^j represents the queue R_1 with i positive costumers and the non-empty queue R_3 with j positive costumers $\forall i \geq 0$ and $\forall j > 0$

The encoding of the G-Network in PEPA is the following:

$$R_1 = \begin{cases} P_0 = (\tau, \lambda_1).P_1 + (fill, \top).P'_0 & \\ P^m_0 = (\tau, \lambda_1).P^m_1 + (fill, \top).P^{m+1}_0 + (empty, \top).P^{m-1}_0 & m > 0 \\ P_n = (\tau, \lambda_1).P_{n+1} + (fill, \top).P'_n + (a, \mu_1 p).P_{n-1} + (a_0, \mu_1(1-p)).P_{n-1} & n > 0 \\ P^m_n = (\tau, \lambda_1).P^m_{n+1} + (fill, \top).P^{m+1}_n + (empty, \top).P^{m-1}_n + (a, \mu_1 p).P^m_{n-1} + & \\ \quad +(a_1, \mu_1(1-p)).P^{m-1}_{n-1} & m > 0; n > 0 \end{cases}$$

$$R_2 = \begin{cases} Q_0 = (a, \top).Q_1 + (a_1, \top).Q_1 & \\ Q_n = (a, \top).Q_{n+1} + (a_1, \top).Q_{n+1} + (\tau, \mu_2).Q_{n-1} & n > 0 \end{cases}$$

$$R_3 = \begin{cases} R_0 = (fill, \lambda_3).R_1 + (a_0, \top).R_0 & \\ R_n = (fill, \lambda_3).R_{n+1} + (empty, \mu_3).R_{n-1} + (a_1, \top).R_{n-1} & n > 0 \end{cases}$$

As usual, with this representation, I can define the following correspondence between informal description and encoding:

- 1: represented by (τ, λ_1) in R_1
- 2: represented by $(fill, \lambda_3)$ in R_3 (and R_1 is informed with $(fill, \top)$)
- 3: represented by $(a, \mu_1 p)$ in R_1 and by $(a, \top)$ in R_2
- 4: represented by $(a_0, \mu_1(1-p))$ in R_1 and by $(a_0, \top)$ in R_3
- 5: represented by $(a_1, \mu_1(1-p))$ in R_1 and by $(a_1, \top)$ in both R_2 and R_3
- 6: represented by (τ, μ_2) in R_2
- 7: represented by $(empty, \lambda_3)$ in R_3 (and R_1 is informed with $(empty, \top)$)

5.1.2.5 Solution Warden Process (WaP)

This last trial uses the concept of a warden process. In this trial I add a process in my model called *warden process* which purpose is to oversee the changes of other states (when necessary) and take the right decisions when it is necessary to. In this specific case the warden process has fully the information about the state of R_3, and take the decisions instead of R_1.

This create a dependence of C with R_1 and R_3 and also R_1 "loses" some of its identity (i.e. the kind of work it can perform) and gives it to C and this leads to the split of the original behavior of R_1 in the G-Network in more processes. On the other hand, this concept completely eliminate any possible uncertainty of the possible choices regarding a_0 and a_1 because C knows exactly how many positive costumers are in R_3 so it can decide to perform a_0 in case of empty R_3 or a_1 if R_3 has costumers. Furthermore, C knows also if the departure of the following costumer in R_3 will leave R_3 empty or not.
Another advantage is the decrease in complexity of encoding (i.e. there are only two kind of state in each process: empty or non-empty), a simplification in the complexity of actions and their correspondence with the behavior of the original system.
Moreover, the space of state for R_1 is only n where n is the number of its costumer and for both C and R is m where m is the number of costumers of R_3.
Then similarly to previous encoding, C is completely aware of all changes of R_3 and it is informed by R_3 itself, with the use of cooperation combinator in its actions: $fill$ increases its positive costumers and $empty$ decreases them. In particular those types of actions help C to keep track of the number of costumers in R_3. In this way I split the complexity of R_1 in the previous encoding in two part: in R_1 and C. Then, without letting R_1 knows anything about R_3, I decrease the whole complexity of R_1.
Information about R_3 is no more stored in an apex of some process but is in the number of C, thus:

- C_0 means that R_3 is an empty queue R_3 with 0 positive costumers

- C_i means that R_3 is a non-empty queue R_3 with i positive costumers $\forall i > 0$

The encoding of the G-Network in PEPA is the following:

$$R_1 = \begin{cases} P_0 = (\tau, \lambda_1).P_1 \\ P_n = (\tau, \lambda_1).P_{n+1} + (a, \mu_1 p).P_{n-1} + (a_0, \top).P_{n-1} + (a_1, \top).P_{n-1} \end{cases} \quad n > 0$$

$$R_2 = \begin{cases} Q_0 = (a, \top).Q_1 + (a_1, \top).Q_1 \\ Q_n = (a, \top).Q_{n+1} + (a_1, \top).Q_{n+1} + (\tau, \mu_2).Q_{n-1} \end{cases} \quad n > 0$$

$$R_3 = \begin{cases} R_0 = (fill, \lambda_3).R_1 + (a_0, \top).R_0 \\ R_n = (fill, \lambda_3).R_{n+1} + (empty, \mu_3).R_{n-1} + (a_1, \top).R_{n-1} \end{cases} \quad n > 0$$

$$C = \begin{cases} C_0 = (fill, \top).C_1 + (a_0, \mu_1(1-p)).C_0 \\ C_n = (fill, \top).C_{n+1} + (empty, \top).C_{n-1} + (a_1, \mu_1(1-p)).C_{n-1} \end{cases} \quad n > 0$$

As usual, I can define the following correspondence between informal description and encoding:

- 1: represented by (τ, λ_1) in R_1
- 2: represented by $(fill, \lambda_3)$ in R_3 (and both R_1 and C are informed with $(fill, \top)$)
- 3: represented by $(a, \mu_1 p)$ in R_1 and by $(a, \top)$ in R_2
- 4: represented by $(a_0, \mu_1(1-p))$ in C and by $(a_0, \top)$ in both R_1 and R_3
- 5: represented by $(a_1, \mu_1(1-p))$ in C and by $(a_1, \top)$ in R_1 and R_2 and R_3
- 6: represented by (τ, μ_2) in R_2
- 7: represented by $(empty, \lambda_3)$ in R_3 (and both R_1 and C are informed with $(empty, \top)$)

5.1.3 Analysis and Modeling - Description of the second example

I will now proceed with the analysis and encoding of the second example. The second example is a **G-network with iterative customer removals** started with the arrival of an external negative trigger β, this system add also the

concept of *chains* of actions in the model, further than the previous example. This concatenation of a lot of instant actions will stress my trials and enlighten their advantages and their weaknesses.
It is represented by the following image:

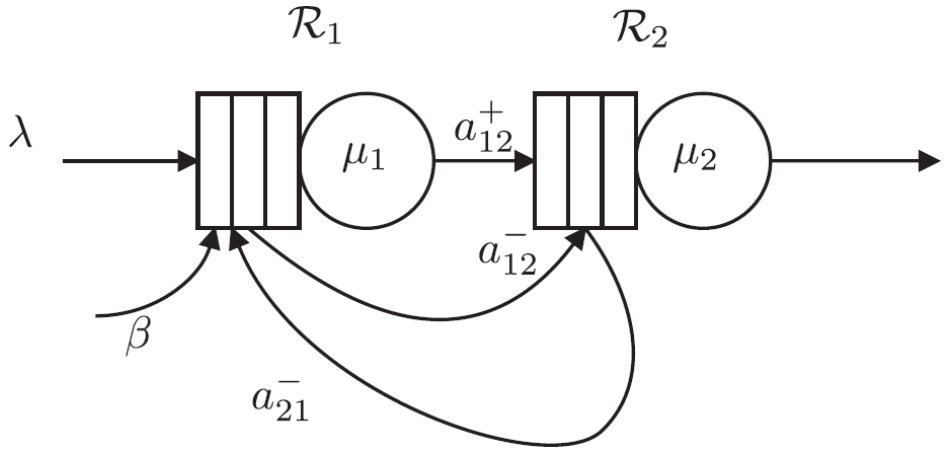

Figure 5.2: G-network with iterative customer removals

In this example there is the use of a negative trigger β. Negative and positive triggers are very similar, except that a negative one transform a removed costumer into a trigger itself (which can be positive or negative). Thus, if a negative trigger is generated from a removed costumer, its arrival to the next queue will be also a trigger and this can lead to the formation of chains of negative triggers in a system. Those triggers will stop only if they move to an empty queue or they become a positive trigger or leave the network (like ordinary negative customers) and have a strictly specific order of their routing (i.e. trigger with route $1, 2, 3$ has different behavior from one with route $1, 3, 2$). In this way a whole subset of queues of the system can be emptied.
In this specific type of G-Network, an homogeneous Poisson processes defines the arrival of positive customers, which arrive from outside the system to queue R_1, with rates λ. Moreover, they are served first in R_1 and then in R_2, the rates μ_1 and μ_2 represent in fact the service times in each queue and so they are exponential random variables, which are independent. A costumer of R_1, after the completion of its service, moves to R_2 as a regular costumer (a_{12}^+). A negative trigger can arrive with rate β in R_1 and it reduces the length

of its queue by one unit if R_1 is a non-empty queue. Moreover, that removed costumer become a negative trigger and it will instantaneously propagate to R_2, then back to R_1, then to R_2 again and so on, until one of the two queue is emptied when the chain of triggers arrives. In this way the chain terminates. (a_{12}^-) represents the propagation of the trigger from R_1 to R_2 and (a_{21}^-) represents the opposite. Thus, this example allows the construction of chains of instantaneous transitions from the two queues. These chains are finite even if they are also unbounded in length. Obviously, we have to notice that when both R_1 and R_2 are non-empty queues, then the trigger causes the change of states in both of them simultaneously and usually with an high change in their numbers of positive costumers.

If we considering the following informal annotations:

(r_1, r_2) is the state of the system, describing each queue with the number of its costumers
where:
r_1 is the number of positive costumers in R_1,
r_2 is the number of positive costumers in R_2,

And:

r_i^+ represents the addition of a positive costumer to queue i (with $i = 1, 2, 3$) and is equivalent to $r_i^+ = r_i + 1$
r_i^- represents the completion of a service of a positive costumer and its consequently departure from queue i (with $i = 1, 2, 3$) and is equivalent to $r_i^- = r_i - 1$

An overall informal description of this G-Network behavior can be the following:

1) In R_1 can arrive a positive costumer $\forall r_1 \geq 0$, this implies the change of overall state to (r_1^+, r_2)

2) In R_1 a positive costumer can have its service completed $\forall r_1 > 0$ and then it remains a positive costumer and goes to R_2, this implies the change of overall state to (r_1^-, r_2^+)

3) In R_2 a positive costumer can have its service completed $\forall r_2 > 0$ and then it leaves the system as a positive costumer, this implies the change of overall state to (r_1, r_2^-)

4) In R_1 when is an empty queue, can arrive a negative trigger from both outside the system or R_2 and it stops itself and does nothing in R_1, the overall state remains $(0, r_2)$

5) In R_1 a negative trigger can arrive from both outside the system or R_2 $\forall r_1 > 0$ and then, after it kills a positive costumer in the queue, it propagates immediately to R_2, this implies the change of overall state to (r_1^-, r_2)

6) In R_2 when is an empty queue, can arrive a negative trigger from R_1 and it stops itself and does nothing in R_2, this doesn't change the overall state $(r_1, 0)$

7) In R_2 a negative trigger can arrive from R_1 $\forall r_2 > 0$ and then, after it kills a positive costumer in the queue, it propagates immediately to R_1, this implies the change of overall state to (r_1, r_2^-)

To summarize the arrivals and departures in each state:

- 1: arrival in R_1
- 2,5: departure from R_1
- 2: arrival in R_2
- 3,7: departure from R_2

To summarize relations between state of components and actions:

- $r_1 \geq 0$ can do 1,4
- $r_1 > 0$ can do 1,2,5
- $r_2 \geq 0$ can do 2,6
- $r_2 > 0$ can do 2,3,7

And the only action which change more queues are 2, but 5 and 7 are part of a chain of changes.

In the following section, I try to capture the behavior of this G-Network with PEPA language.

5.1.4 Analysis and Modeling - Encoding Trials of G-network with Iterative Customer Removals

I will now present the stress test of my trials with the encoding of the G-network with Iterative Customer Removals using PEPA. I adopt the annotation in which P_i is a generic state of R_1 and Q_j of R_2.

5.1.4.1 Exclusive Addition (ExA)

In this trial I use again the concept of exclusive addition but with another concise representation. In this concise representation of the exclusive addition I use the concept of $if/else$ because there can be two main cases, even if the functionality of the underlying exclusive addition doesn't change. The two cases are:

- the number of costumers of R_1 is less than the number of R_2, in this case the negative trigger will empty the R_1 queue
- the number of costumers of R_1 is more than the number of R_2, in this case the negative trigger will subtract only a certain amount of costumers in R_1 queue

It is more or less the same, also in the case of Q process.
Thus, the general exclusive addition $(a_{0n}, \beta).P_{n-1} \oplus (a_{1n}, \beta).P_{n-2} \oplus \cdots \oplus (a_{n\,n}, \beta).P_0 \oplus \cdots$ is split in two parts
$(a_{0n}, \beta).P_{n-1} \oplus (a_{1n}, \beta).P_{n-2} \oplus \cdots \oplus (a_{n-2\,n}, \beta)$ in which $x + 1 < n$ and the arrival process is P_{n-x-1},
$(a_{n-1\,n}, \beta).P_0 \oplus (a_{n\,n}, \beta).P_0 \oplus (a_{n+1\,n}, \beta).P_0 \oplus \cdots$ in which $x + 1 \geq n$ and the arrival process is always P_0.
This is done in order to avoid negative results from subtractions because negative states don't exist in this system.
Like in the other concise description, also here in the extended representation of the exclusive addition, the x will be replaced by all its possible values.

The encoding of the G-Network in PEPA is the following:

$$R_1 = \begin{cases} P_0 = (\tau, \lambda).P_1 + (\tau, \beta).P_0 \\ P_n = (\tau, \lambda).P_{n+1} + (a, \mu_1).P_{n-1} + \\ \qquad + \underbrace{if(x+1<n)then(a_{xn}, \beta).P_{n-x-1} else(a_{xn}, \beta).P_0}_{(a_{0n},\beta).P_{n-1} \oplus (a_{1n},\beta).P_{n-2} \oplus \cdots \oplus (a_{n\,n},\beta).P_0 \oplus \cdots} & n > 0 \end{cases}$$

$$R_2 = \begin{cases} Q_0 = (a, \top).Q_1 + \overbrace{(a_{0x}, \top).Q_0}^{(a_{01},\top).Q_0 \oplus \cdots} \\ Q_n = (a, \top).Q_{n+1} + (\tau, \mu_2).Q_{n-1} + \\ \qquad + \underbrace{if(x<n)then(a_{nx}, \top).Q_{n-x} else(a_{nx}, \top).Q_0}_{(a_{n1},\top).Q_{n-1} \oplus (a_{n2},\top).Q_{n-2} \oplus \cdots \oplus (a_{n\,n},\top).Q_0 \oplus \cdots} & n > 0 \end{cases}$$

With this representation, I can define the following correspondence between informal description and encoding:

- 1: represented by (τ, λ) in R_1
- 2: represented by (a, μ_1) in R_1 and by $(a, \top)$ in R_2
- 3: represented by (τ, μ_2) in R_2
- 4: represented by (τ, β) in R_1 or the conclusion of the subtraction in (a_{xn}, β)-chain, in R_1
- 5: represented by the chain (a_{xn}, β) in R_1
- 6: represented by $(a_{0x}, \top)$ in R_2 or the conclusion of the subtraction in $(a_{nx}, \top)$-chain, in R_2
- 7: represented by the chain $(a_{nx}, \top)$ in R_2

From this example we can see that the complexity of the processes and their actions increases and also the encoding result less readable because the possibilities of exclusive addition increase a lot. In the non-concise description of exclusive addition, a process can have hundreds of possible choices, one for each pair of possible elements of R_1 and R_2.
The other big problem is the fact that in the original PEPA, this exclusive addition compositor is not provided and so I have to study further and give a

formal definition of it. Moreover, I would have to prove all the property and theorems that hold for other compositors if I want to make this extension.

5.1.4.2 Solution $\perp$-Actions ($\perp$)

In this trial I use again the concept of $\perp$-actions but with more concatenations of subsequent bottom actions. In particular when a negative trigger arrives in R_1, it starts the removal chain which goes from a process to another removing each time a positive costumer, and so on. All those removal are considered instantaneous thanks to the $\perp$-actions and the chain will terminate when it will encounter a emptied queue.
For this reason, each process are divided in two parts:

- in the first part there is the normal behavior of the system without the trigger behavior, when a trigger arrives, then the system switches to the second part
- in the second part there is the behavior of the negative trigger chain, the process will remain in this part until the chain ends, then the system returns to the first part

We have to notice that the entire second part of the system is instantaneous because it is composed by only bottom actions.
Moreover, each process keeps track of this information about which of its part it has to currently behaves like. This information is stored in both R_1 and R_2 with the apex in its state, thus:

- P_i represents the first part of the behavior of queue R_1, with i positive costumers $\forall i \geq 0$
- P_i' represents the second part of the behavior of queue R_1, with i positive costumers $\forall i \geq 0$

This holds also for Q.

The encoding of the G-Network in PEPA is the following:

$$R_1 = \begin{cases} P_0 = (\tau, \lambda).P_1 + (\tau, \beta).P_0 & \\ P_n = (\tau, \lambda).P_{n+1} + (a, \mu_1).P_{n-1} + (rem, \beta).P'_{n-1} & n > 0 \\ P'_n = (rem, \bot).P'_{n-1} + (stop, \top).P'_{n-1} & n > 0 \\ P'_0 = (stop, \bot).P_0 & \end{cases}$$

$$R_2 = \begin{cases} Q_0 = (a, \top).Q_1 + (rem, \top).(stop, \bot).Q_0 & \\ Q_n = (a, \top).Q_{n+1} + (\tau, \mu_2).Q_{n-1} + (rem, \top).Q'_{n-1} & n > 0 \\ Q'_n = (rem, \top).Q'_{n-1} + (stop, \top).Q'_{n-1} & n > 0 \\ Q'_0 = (stop, \bot).Q_0 & \end{cases}$$

With this representation, I can define the following correspondence between informal description and encoding:

- 1: represented by (τ, λ) in R_1
- 2: represented by (a, μ_1) in R_1 and by $(a, \top)$ in R_2
- 3: represented by (τ, μ_2) in R_2
- 4: represented by (τ, β) or $(stop, \bot)$ in R_1
- 5: represented by (rem, β) or $(rem, \bot)$ in R_1
- 6: represented by $(stop, \bot)$ in R_2
- 7: represented by $(rem, \top)$ in R_2

From this example we can see that also here the complexity of the processes and their actions increases a bit and also the encoding result less readable because of the two natures of the system behavior. A chain can lead to very large concatenations of bottom actions, in particular for very large numbers of positive costumers in each queue.
The other big problem is the fact that in the original PEPA, these $\bot$-actions are not provided and so I have to study further and give a formal definition of it. Moreover, I would have to prove all the property and theorems that hold for other compositors if I want to make this extension.

Nevertheless if we make an approximation of the instantaneous bottom action, we can re-enter the field of normal PEPA language as rate are no more ≥ 0 but only > 0. In this way, to ensure that bottom actions will take place immediately one after the other we can define their rate as small as possible. So small that the probability that they occurs is about 100%.
To summarize:

$$\perp = \lim_{r \to 0} r \text{ for all } r > 0$$

However, we won't have a strictly representation of the original G-Network but only an approximation of it.

5.1.4.3 Solution Partial Index (PI)

This type of encoding, with the partial index, is an **impracticable** solution: it is useless that R_1 knows only if R_2 is an empty queue or a non-empty one, and similarly if R_2 knows only if R_1 is empty or non-empty. Because we have to know exactly how many positive costumers each queue has, in order to make the correct subtraction, this kind of solution is unsuitable for the modeling of this kind of G-Networks.

5.1.4.4 Double Index (DI)

In this trial I use again the concept of double index but I use a further type of representation. In this encoding I use the concept of $MAX(i, j)$ because there are two main cases in both queues.
The two cases are:

- the number of costumers of R_1 is less than the number of R_2, in this case the negative trigger will empty the R_1 queue
- the number of costumers of R_1 is more than the number of R_2, in this case the negative trigger will subtract only a certain amount of costumers in R_1 queue

It is more or less the same, also in the case of Q process.
This is done in order to avoid negative results from subtractions because negative states don't exist in this system. Moreover, this represents only a single choice because in each instantiation of the model $MAX(i - n; 0)$ and $MAX(n - i - 1; 0)$ are gradually substituted by the result of the MAX operator (which behave as the normal operator of maximum in mathematics) and usually they are complementary (i.e. if the first possibility is chosen in the first MAX then in the second is chosen the second one, and vice versa).
In this case the double indexes in both R_1 and R_2 mutually trace fully the information about the states of R_2 and R_1.
I make this encoding because both R_1 and R_2 need to know the state of the other to take an unique and right decision about the outcome of the negative trigger effect. R_1 knows exactly how many positive costumers are in R_2 so it can decide how many positive costumers removes from its queue when there is the arrival of a trigger and R_2 behaves in the same way but with respect to R_1. Thus, remains the simplification in the complexity of actions and their correspondence with the behavior of the original system and also in the whole encoding of the system.
Then, similarly to previous encodings, R_1 is aware of all changes of R_2 and it is informed by R_2 itself, with the use of cooperation combinator in its action a_2 which decreases the positive costumers in R_2 and similarly R_2 is informed by R_1 itself with the use of cooperation combinator in its action a_0 which increases its positive costumers. In particular those types of actions help R_1 and R_2 to mutually keep track of their number of costumers.
This information is stored in R_1 and R_2 with the apex number in the name of their state, thus:

- P_i represents the queue R_1 with i positive costumers and the empty queue R_2 with 0 positive costumers, $\forall i \geq 0$
- P_i^j represents the queue R_1 with i positive costumers and the non-empty queue R_2 with j positive costumers $\forall i \geq 0$ and $\forall j > 0$

And vice versa for Q in R_2.

The encoding of the G-Network in PEPA is the following:

$$R_1 = \begin{cases} P_0 = (a_0, \lambda).P_1 + (\tau, \beta).P_0 & \\ P_n = (a_0, \lambda).P_{n+1} + (a_1, \mu_1).P'_{n-1} + (\tau, \beta).P_{n-1} & n > 0 \\ P_0^i = (a_0, \lambda).P_1^i + (a_2, \top).P_0^{i-1} + (\tau, \beta).P_0^i & i > 0 \\ P_n^i = (a_0, \lambda).P_{n+1}^i + (a_1, \mu_1).P_{n-1}^{i+1} + (a_2, \top).P_n^{i-1} + (b, \beta).P_{MAX(n-i-1;0)}^{MAX(i-n;0)} & n > 0, i > 0 \end{cases}$$

$$R_2 = \begin{cases} Q_0 = (a_0, \top).Q'_0 & \\ Q_n = (a_0, \top).Q'_n + (a_2, \mu_2).Q_{n-1} & n > 0 \\ Q_0^i = (a_0, \top).Q_0^{i+1} + (a_1, \top).Q_1^{i-1} & i > 0 \\ Q_0^i = (a_0, \top).Q_n^{i+1} + (a_1, \top).Q_{n+1}^{i-1} + (a_2, \mu_2).Q_{n-1}^i + (b, \top).Q_{MAX(n-i;0)}^{MAX(i-n-1;0)} & n > 0, i > 0 \end{cases}$$

With this representation, I can define the following correspondence between informal description and encoding:

- 1: represented by (a_0, λ) in R_1 (and R_2 is informed with $(a_0, \top)$)
- 2: represented by (a_1, μ_1) in R_1 and by $(a_1, \top)$ in R_2
- 3: represented by (a_2, μ_2) in R_2 (and R_1 is informed with $(a_2, \top)$)
- 4: represented by (τ, β) in R_1 or the conclusion of the subtraction caused by (b, β) in R_1
- 5: represented by the subtraction caused by (b, β) in R_1
- 6: represented by the conclusion of the subtraction caused by $(b, \top)$, in R_2
- 7: represented by the subtraction caused by $(b, \top)$ in R_2

5.1.4.5 Warden Process (WaP)

In this trial I use again the concept of warden process but with a further concise representation. In this concise representation of the exclusive addition I use the concept of $x; y$ within the name of an action. This is done because neither R_1 nor R_2 know anything about each other, so they cannot decide how many customers they have to remove and this information is passed by C in the name of the actions in the form of variable x and y. Those two variable will be instantiated in each call of the actions.
It is of great importance to notice the difference between exclusive addition, double index and warden process in this case:

- Exclusive Addition: in each process P_i there are m different choices combinators, one for each possible value of R_2 positive costumers
- Double Index: in each process P_i there is only a single choice combinator of type b and the subtraction is computed by R_1 itself because it knows the numbers of positive costumers in R_2
- Warden Process: in each process P_i there are m different choices combinators of type b, one for each possible outcome of the subtraction of costumers, but the subtraction is computed by C that is the only process that knows the numbers of positive costumers in both R_1 and R_2

It is more or less the same, also in the case of Q process.
Thus, in the extended representation, x and y will be replaced by all its possible values.
Also in this case the *warden process* oversees all the changes of other states and take the right decisions when it is necessary to, in particular it has fully the information about the states of R_1 and R_2, and take the decisions instead of R_1 about the triggering effect.
In this way C knows exactly how many positive costumers are in R_1 and R_2 so it can decide to perform the right subtraction of normal costumers during b_{xy} action. Furthermore, with C the complexity in R_1 and R_2 decreases.
Then similarly to previous encoding, C is completely aware of all changes of other states and it is informed by R_1 and R_2 themselves, with the use of cooperation combinator in its actions: a_0,a_1 and a_2 which inserts, transfers and moves out a costumer in the chain $R_1 - R_2$, respectively. In particular those types of actions help C to keep track of the number of costumers in queues.

Information about R_1 and R_2 is no more stored in an apex of some process but is in the number and apex of C, thus:

- C_0 means that R_1 and R_2 are empty queue with 0 positive costumers
- C_n^i means that R_1 is a non-empty queue with n positive costumers $\forall n > 0$ and R_2 is a non-empty queue with i positive costumers $\forall i > 0$

The encoding of the G-Network in PEPA is the following:

$$R_1 = \begin{cases} P_0 = (a_0, \lambda).P_1 + (\tau, \beta).P_0 \\ P_n = (a_0, \lambda).P_{n+1} + (a_1, \mu_1).P_{n-1} + \underbrace{(b_{xy}, \top).P_x}_{(b_{0y},\top).P_0+(b_{1y},\top).P_1+\cdots+(b_{ny},\top).P_n+\cdots} & n > 0 \end{cases}$$

$$R_2 = \begin{cases} Q_0 = (a_1, \top).Q_1 \\ Q_n = (a_1, \top).Q_{n+1} + (a_2, \mu_2).Q_{n-1} + \underbrace{(b_{xy}, \top).Q_y}_{(b_{x0},\top).Q_0+(b_{x1},\top).Q_1+\cdots+(b_{xn},\top).Q_n+\cdots} & n > 0 \end{cases}$$

$$C = \begin{cases} C_0 = (a_0, \top).C_1 & \\ C_n = (a_0, \top).C_{n+1} + (a_1, \top).C'_{n-1} + (b_{n-1}, \beta).C_{n-1} & n > 0 \\ C_0^i = (a_0, \top).C_1^i + (a_2, \top).C_0^{i-1} & i > 0 \\ C_n^i = (a_0, \top).C_{n+1}^i + (a_1, \top).C_{n-1}^{i+1} + (a_2, \top).C_n^{i-1} + & \\ \qquad +(b_{(MAX(n-i-1;0);MAX(i-n;0))}, \beta).C_{MAX(n-i-1;0)}^{MAX(i-n;0)} & n > 0, i > 0 \end{cases}$$

With this representation, I can define the following correspondence between informal description and encoding:

- 1: represented by (a_0, λ) in R_1 (and C is informed with $(a_0, \top)$)
- 2: represented by (a_1, μ_1) in R_1 and by $(a_1, \top)$ in R_2 (and C is informed with $(a_1, \top)$)
- 3: represented by (a_2, μ_2) in R_2 (and C is informed with $(a_2, \top)$)
- 4: represented by (τ, β) in R_1 or the conclusion of the subtraction caused by $(b_{xy}, \top)$ in R_1

- 5: represented by the subtraction caused by $(b_{xy}, \top)$ in R_1
- 6: represented by the conclusion of the subtraction caused by $(b_{xy}, \top)$, in R_2
- 7: represented by the subtraction caused by $(b_{xy}, \top)$ in R_2

Moreover, in order to make b_{xy} action work, there is the need of (b_{xy}, β) in C.

In this example we can see that there is a bit growth in choice length of the processes due to b_{xy} and this solution seems to be a joint point between exclusive addiction and double index solutions.

5.1.5 Derivation Graphs of G-network with Trigger

In this part I choose two of my trial solutions and draw their derivation Graphs of *G-network with Trigger*, to see visually their behaviors. I select the Double Index and the Warden Process solutions, I have discarded the others for these reasons:

- Exclusive Addition because in the original PEPA, this exclusive addition compositor is not provided and so I have to study further and give a formal definition of it and prove all the property and theorems that hold for other compositors.
- $\perp$-Actions for the same reason of the exclusive addition.
- Partial Index: because it cannot encode the second example, and this is an unbearable limitation.

I will now present the various derivation graphs.

5.1.5.1 Derivation Graphs - Double Index (DI)

As expected the graph grows in two dimensions: horizontal and vertical. The various columns represent the number of normal costumers in R_1 instead the rows represent the information stored in R_1 about the number of positive costumers in R_3. This means that the state $2'$ represents R_1 with 2 positive costumers and R_3 with only 1 positive costumers.

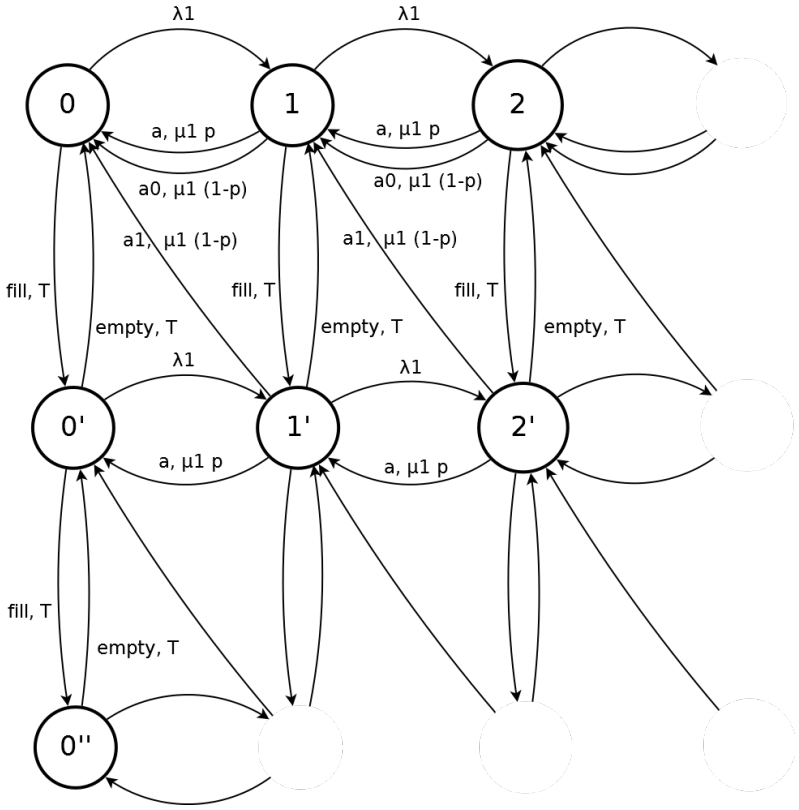

Figure 5.3: DI solution of G-network with Trigger: Derivation Graph of process R_1

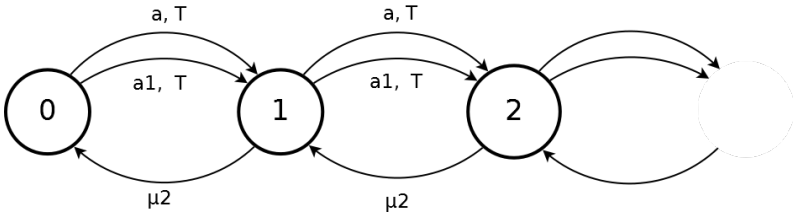

Figure 5.4: DI solution of G-network with Trigger: Derivation Graph of process R_2

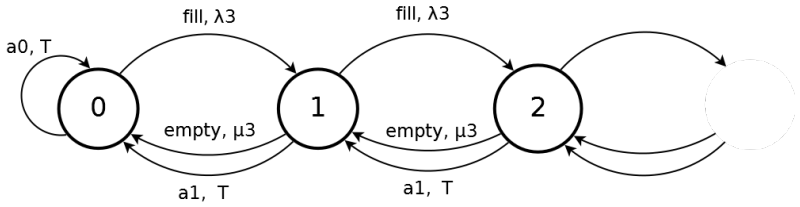

Figure 5.5: DI solution of G-network with Trigger: Derivation Graph of process R_3

Those two diagrams didn't have particular behaviors to argue about.

5.1.5.2 Solution Warden Process (WaP) Diagrams

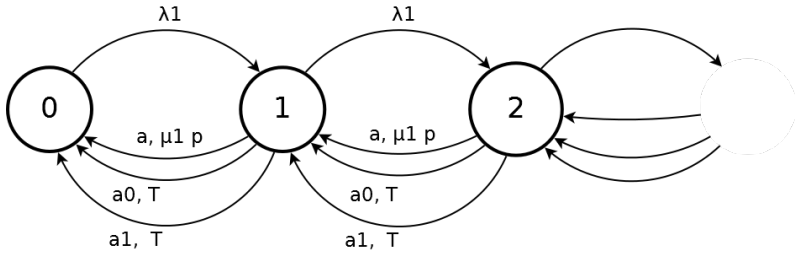

Figure 5.6: WaP solution of G-network with Trigger: Derivation Graph of process R_1

We can notice that this derivation graph is much less complex with respect to the first solution thank to the process C which takes a lot of work and complexity from this process.
Unfortunately there is a flaw in this process, even if they don't behave as exclusive addition because they are not active, a_0 and a_1 are present as a transition to each state of he process. We will see how this can be a problem after I show all derivation graphs.

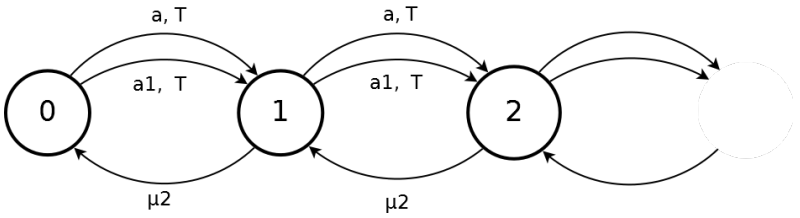

Figure 5.7: WaP solution of G-network with Trigger: Derivation Graph of process R_2

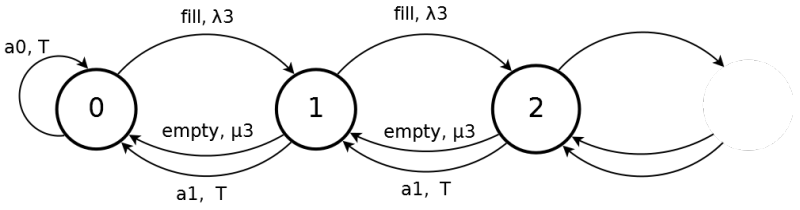

Figure 5.8: WaP solution of G-network with Trigger: Derivation Graph of process R_3

R_2 and R_3 are represented by the same derivation graphs in both double index and warden process solutions.

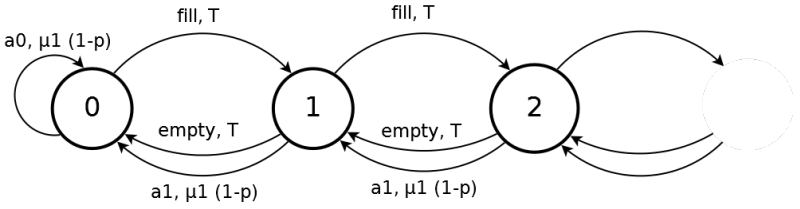

Figure 5.9: WaP solution of G-network with Trigger: Derivation Graph of process C

We can see that C process behavior is almost the same of R_3 except that active and passive actions are switched. From this we can comprehend that must exist a dependence from process R_3 if we want to capture the exact behavior of the original G-Network with PEPA, without introduce any extension of it.

This is caused by the fact that a_0 and a_1 are passive in R_3 but they are dependent on the state (i.e. the number of positive costumers) of R_3 itself. This leads to a dependence on R_3 of anyone who want to perform those kind of actions.
The main difference between double index and warden process solutions is the process which carries this dependence. In DI is R_1 the process which perform a_0 and a_1 and this causes the huge numbers on the space of its states. In WaP is the warden process C the one that carries this burden, and this brings to some sort of repetition of the behavior of R_3.

Analyzing better the two solutions, I found that in the warden process solution, even if the actions which behave similarly as the exclusive addition are passive, they have the same underlying problem. When the system works on the whole, each process cooperates with each other and in this phase every passive action rate $\top$ will be substituted by the positive rate of the action which cooperates. In this way each passive action of R_1 in WaP will have the same rate $\mu_1(1-p)$ and subsequently has the same behavior of the exclusive addition. Thus, there is the problem that in the original PEPA it is not provided and so I would give a formal definition of it if I wanted to use this idea. Moreover, another big problem is satisfaction of the conditions of the product form theorem. An anticipation of that topic is that in each state all the passive actions must be outgoing from each state and all the active actions must be ingoing. This leads to a problem in R_1 because a_0 and a_1 are passive in it but in state 0 they are not outgoing. If I add any transition from state 0 to anywhere with type a_0 it will synchronize with both R_3 and C altering the behavior of my system. This make the behavior of my model encoding different from the original one and this cannot be acceptable.
For this reason I decide to choose the Double Index solution for the further analysis, for comparison with the original system and for the rest of my thesis.

5.1.6 Behavior Comparison between G-Network with Trigger and My Model

In this section I will compare the behavior of my solution with the behavior of the original G-Network. In order to verify if they are the same I have to demonstrate that my model can perform the same actions of the original system. If it is so, then they also share the same underlying Markov Process and, subsequently, my model has the product-form because the original system has it.
To prove this equivalence I use the inductive reasoning. The induction in this case work as the following:

- Base Case: considering each base case, to prove that they are equivalent I have to demonstrate if they could perform the same and only actions, one with respect to the other.

- Inductive Case: considering equivalent a general n case represented by a general state of the two systems, I have to prove that the cases $n + 1$, reached from this case n, are the same (i.e. have the same behavior).

If I can prove this two conditions then the two systems are equivalent and so my encoding of G-Network with trigger, using PEPA, is correct and valid.

Firstly, I will state each possible performable action of the two systems, then I will proceed with the comparison. To prove the base case I will consider each of their base case and analyze if they can perform the same activities. Then to prove the inductive case I will consider that the various actions performed previous to arrive to the general state n are the same (inductive hypothesis) and then I have to demonstrate that from n I can reach the same states $n + 1$ with the same actions (i.e. behavior).
Furthermore, to prove the equivalence I have to demonstrate that my system can perform an action *if and only if* the original system performs it too. This implies that my proof has to go in both directions:

- from original G-Network to my model ($\Rightarrow$)

- from my model to original G-Network ($\Leftarrow$)

In this way I will prove the equivalence of the two systems.

5.1.6.1 Behavioral Correspondence

Now I will demonstrate that my system in PEPA is equivalent to the original one.
The proof by induction is the following:

Assumptions and Notations:
1) The G-network will be represented by ($\overline{n_1}$, $\overline{n_2}$, $\overline{n_3}$), namely the states of his queues, where:

- $\overline{n_1}$: represents the number of positive costumers in queue R_1 of G-Network
- $\overline{n_2}$: represents the number of positive costumers in queue R_2 of G-Network
- $\overline{n_3}$: represents the number of positive costumers in queue R_3 of G-Network

So the original system will be $(\overline{n_1}, \overline{n_2}, \overline{n_3})$

2) My system will also be represented by ((n_{11},n_{13}), n_2, n_3), namely the states of my processes P (the first two numbers), Q and R, where:

- n_{11}: represents the number i in the state name of a general process P_i^j, i.e. the number of positive costumers in process R_1 of my encoding
- n_{13}: represents the number j in the apex name of a general process P_i^j, i.e. the information about the number of positive costumers in process R_3 of my encoding, stored in R_1
- n_2: represents the number i in the state name of a general process Q_i, i.e. the number of positive costumers in process R_2 of my encoding
- n_3: represents the number i in the state name of a general process R_i, i.e. the number of positive costumers in process R_3 of my encoding

So my model will be ((n_{11},n_{13}), n_2, n_3)
Is important to note that that (n_{11}, n_{13}) is the state of process R_1, n_2 of R_2, n_3 of R_3 and more important that $n_{13} = n_3$ always $\forall n_{13} \wedge n_3$.

My system will be represented as $S = ((n_{11}, n_{13}), n_2, n_3)$

where:
$S \stackrel{def}{=} R_1 \underset{a,a_0,a_1,fill,empty}{\bowtie} R_2 \underset{a,a_0,a_1,fill,empty}{\bowtie} R_3$

Thorem:
The two systems $(\overline{n_1}, \overline{n_2}, \overline{n_3})$ and $((n_{11}, n_{13}), n_2, n_3)$ are the same system (i.e. are equivalent and have the same behavior).
To demonstrate this I will show that their behavior are equal and in order to do this I have to demonstrate that:

$(\overline{n_1}, \overline{n_2}, \overline{n_3}) \Leftrightarrow ((n_{11}, n_{13}), n_2, n_3)$

That is equivalent to:

$\forall(\overline{n_1}, \overline{n_2}, \overline{n_3}) \Rightarrow \exists((n_{11}, n_{13}), n_2, n_3)$ and $\forall((n_{11}, n_{13}), n_2, n_3) \Rightarrow \exists(\overline{n_1}, \overline{n_2}, \overline{n_3})$.

5.1.6.2 Possible Actions of two Systems

The original system $(\overline{n_1}, \overline{n_2}, \overline{n_3})$ can perform the following actions (i.e. transition between states), with also the following restrictions and rates:

- Informal description: A positive costumer arrives to queue R_1
  Action 1: from $(\overline{n_1}, \overline{n_2}, \overline{n_3})$ to $(\overline{n_1 + 1}, \overline{n_2}, \overline{n_3})$
  Restriction: $\forall\overline{n_1} \geq 0$
  Rate: λ_1

- Informal description: A positive costumer arrives to queue R_3
  Action 2: from $(\overline{n_1}, \overline{n_2}, \overline{n_3})$ to $(\overline{n_1}, \overline{n_2}, \overline{n_3 + 1})$
  Restriction: $\forall\overline{n_3} \geq 0$
  Rate: λ_3

- Informal description: A positive costumer moves from queue R_1 to queue R_2
  Action 3: from $(\overline{n_1}, \overline{n_2}, \overline{n_3})$ to $(\overline{n_1 - 1}, \overline{n_2 + 1}, \overline{n_3})$
  Restriction: $\forall\overline{n_1} \geq 1$ and $\forall\overline{n_2} \geq 0$
  Rate: $\mu_1 p$

- Informal description: A positive costumer moves from queue R_1 becomes a trigger and arrives to queue R_3 when it is empty

Action 4: from $(\overline{n_1}, \overline{n_2}, 0)$ to $(\overline{n_1 - 1}, \overline{n_2}, 0)$
Restriction: $\forall \overline{n_1} \geq 1$ and $\overline{n_3} = 0$
Rate: $\mu_1(1-p)$

- Informal description: A positive costumer moves from queue R_1 becomes a trigger and arrives to queue R_3 when it is non-empty, then a positive costumer from R_3 is forced to move to R_2
  Action 5: from $(\overline{n_1}, \overline{n_2}, \overline{n_3})$ to $(\overline{n_1 - 1}, \overline{n_2 + 1}, \overline{n_3 - 1})$
  Restriction: $\forall \overline{n_1} \geq 1$ and $\forall \overline{n_2} \geq 0$ and $\forall \overline{n_3} \geq 1$
  Rate: $\mu_1(1-p)$

- Informal description: A positive costumer leaves the system from queue R_2
  Action 6: from $(\overline{n_1}, \overline{n_2}, \overline{n_3})$ to $(\overline{n_1}, \overline{n_2 - 1}, \overline{n_3})$
  Restriction: $\forall \overline{n_2} \geq 1$
  Rate: μ_2

- Informal description: A positive costumer leaves the system from queue R_3
  Action 7: from $(\overline{n_1}, \overline{n_2}, \overline{n_3})$ to $(\overline{n_1}, \overline{n_2}, \overline{n_3 - 1})$
  Restriction: $\forall \overline{n_3} \geq 1$
  Rate: μ_3

From the derivation graphs, I can now analyze each process of my model, their actions (i.e. transition between states), with also their restrictions and rates.
Process P which represents R_1 can perform:

- State P_0
  - Action: τ
    Restriction: none
    Rate: λ_1
  - Action: $fill$
    Restriction: cooperate, has to wait R_3 action of $fill$
    Rate: $\top$
- State P_n with $n > 0$

  - Action: τ
    Restriction: none
    Rate: λ_1
  - Action: $fill$
    Restriction: cooperate, has to wait R_3 action of $fill$
    Rate: $\top$
  - Action: a
    Restriction: none, but there is an active cooperation with R_2
    Rate: $\mu_1 p$
  - Action: a_0
    Restriction: none, but there is an active cooperation with R_3, moreover R_3 must be empty but this is ensured by the fact that there wasn't any $fill$ action in R_3 otherwise the state will be P_n^i with $i > 0$
    Rate: $\mu_1(1-p)$

- State P_0^i with $i > 0$
  - Action: τ
    Restriction: none
    Rate: λ_1
  - Action: $fill$
    Restriction: cooperate, has to wait R_3 action of $fill$
    Rate: $\top$
  - Action: $empty$
    Restriction: cooperate, has to wait R_3 action of $empty$
    Rate: $\top$

- State P_n^i with $n > 0$ and $i > 0$
  - Action: τ
    Restriction: none
    Rate: λ_1
  - Action: $fill$
    Restriction: cooperate, has to wait R_3 action of $fill$
    Rate: $\top$

  - Action: $empty$
    Restriction: cooperate, has to wait R_3 action of $empty$
    Rate: $\top$
  - Action: a
    Restriction: none, but there is an active cooperation with R_2
    Rate: $\mu_1 p$
  - Action: a_1
    Restriction: none, but there is an active cooperation with R_3, moreover R_3 must be non-empty but this is ensured by the fact that there were one or more $fill$ action in R_3 otherwise the state will be only P_n with $i = 0$
    Rate: $\mu_1(1-p)$

Process Q which represents R_2 can perform:

- State Q_0
  - Action: a
    Restriction: cooperate, has to wait R_1 action of a
    Rate: $\top$
  - Action: a_1
    Restriction: cooperate, has to wait R_1 action of a_1 (so R_1 and R_3 must be non-empty queues)
    Rate: $\top$
- State Q_n with $n > 0$
  - Action: a
    Restriction: cooperate, has to wait R_1 action of a
    Rate: $\top$
  - Action: a_1
    Restriction: cooperate, has to wait R_1 action of a_1 (so R_1 and R_3 must be non-empty queues)
    Rate: $\top$
  - Action: τ
    Restriction: none
    Rate: μ_2

Process R which represents R_3 can perform:

- State R_0
  - Action: $fill$
    Restriction: none, but there is an active cooperation with R_1
    Rate: λ_3
  - Action: a_0
    Restriction: cooperate, has to wait R_1 action of a_0
    Rate: $\top$
- State R_n with $n > 0$
  - Action: $fill$
    Restriction: none, but there is an active cooperation with R_1
    Rate: λ_3
  - Action: $empty$
    Restriction: none, but there is an active cooperation with R_1
    Rate: μ_3
  - Action: a_1
    Restriction: cooperate, has to wait R_1 action of a_1
    Rate: $\top$

Where there are no further specifications, when a process waits for a cooperation (i.e. it is passive with respect to that action) means that the conditions for that activity are imposed only by the active process. Moreover, when a process doesn't have any restriction in an activity but there is an active cooperation, means that there are no condition in that activity (except the ones in its state, e.g. if the activity is present only in P_n and not in P_0 this means that one condition on that activity is $n > 0$)

To summarize, my model encoded from original system $((n_{11}, n_{13}), n_2, n_3)$ can perform the following actions (i.e. transition between states), with also the following restrictions and rates:

- Action τ: from $((n_{11}, n_{13}), n_2, n_3)$ to $((n_{11} + 1, n_{13}), n_2, n_3)$
  Restriction: $\forall n_{11} \geq 0$
  Rate: λ_1

- Action $fill$: from $((n_{11}, n_{13}), n_2, n_3)$ to $((n_{11}, n_{13}+1), n_2, n_3+1)$
  Restriction: $\forall n_{13} = n_3 \geq 0$
  Rate: λ_3

- Action a: from $((n_{11}, n_{13}), n_2, n_3)$ to $((n_{11}-1, n_{13}), n_2+1, n_3)$
  Restriction:$\forall n_{11} \geq 1$ and $\forall n_2 \geq 0$
  Rate: $\mu_1 p$

- Action a_0: from $((n_{11}, 0), n_2, 0)$ to $((n_{11}-1, 0), n_2, 0)$
  Restriction:$\forall n_{11} \geq 1$ and $\forall n_{13} = n_3 = 0$
  Rate: $\mu_1(1-p)$

- Action a_1: from $((n_{11}, n_{13}), n_2, n_3)$ to $((n_{11}-1, n_{13}-1), n_2+1, n_3-1)$
  Restriction:$\forall n_{11} \geq 1$ and $\forall n_{13} = n_3 \geq 1$ and $\forall n_2 \geq 0$
  Rate: $\mu_1(1-p)$

- Action τ: from $((n_{11}, n_{13}), n_2, n_3)$ to $((n_{11}, n_{13}), n_2-1, n_3)$
  Restriction: $\forall n_2 \geq 1$
  Rate: μ_2

- Action $empty$: from $((n_{11}, n_{13}), n_2, n_3)$ to $((n_{11}, n_{13}-1), n_2-1, n_3-1)$
  Restriction: $\forall n_{13} = n_3 \geq 1$
  Rate: μ_3

5.1.6.3 Proof of the Theorem

The first proof direction is from original G-Network to my PEPA encoding.

$\Rightarrow$:

Base Case:
We have the following 7 base cases in the original system:

- $(\overline{0}, \overline{0}, \overline{0})$: all queues are empty.

- $(\overline{n}, \overline{0}, \overline{0})$: R_1 has n positive costumers, R_2 and R_3 are empty.

- $(\overline{0}, \overline{n}, \overline{0})$: R_2 has n positive costumers, R_1 and R_3 are empty.

- $(\overline{0}, \overline{0}, \overline{n})$: R_3 has n positive costumers, R_1 and R_2 are empty.

- $(\overline{n_1}, \overline{n_2}, \overline{0})$: R_1 has n_1 positive costumers, R_2 has n_2 positive costumers and R_3 is empty.
- $(\overline{n_1}, \overline{0}, \overline{n_3})$: R_1 has n_1 positive costumers, R_3 has n_3 positive costumers and R_2 is empty.
- $(\overline{0}, \overline{n_2}, \overline{n_3})$: R_2 has n_2 positive costumers, R_3 has n_3 positive costumers and R_1 is empty.

Case 1
From state:
$(\overline{0}, \overline{0}, \overline{0})$

We can have only 2 possible actions with this values: Action 1 and Action 2.
And these are the 2 possible transactions of state:

1. $(\overline{0}, \overline{0}, \overline{0}) \xrightarrow{1,\lambda_1} (\overline{1}, \overline{0}, \overline{0})$
2. $(\overline{0}, \overline{0}, \overline{0}) \xrightarrow{2,\lambda_3} (\overline{0}, \overline{0}, \overline{1})$

There exists the following correspondence in my system $((0,0),0,0)$ with actions: τ and $fill$.
And these are the 2 possible transactions of state:

1. $((0,0),0,0) \xrightarrow{\tau,\lambda_1} ((1,0),0,0)$
2. $((0,0),0,0) \xrightarrow{fill,\lambda_3} ((0,1),0,1)$

Case 2
From state:
$(\overline{n}, \overline{0}, \overline{0})$

We can have only 4 possible actions with this values: Action 1 and Action 2, Action 3, Action 4.
And these are the 4 possible transactions of state:

1. $(\overline{n}, \overline{0}, \overline{0}) \xrightarrow{1,\lambda_1} (\overline{(n+1)}, \overline{0}, \overline{0})$

2. $(\overline{n}, \overline{0}, \overline{0}) \xrightarrow{2,\lambda_3} (\overline{n}, \overline{0}, \overline{1})$

3. $(\overline{n}, \overline{0}, \overline{0}) \xrightarrow{3,\mu_1 p} (\overline{(n-1)}, \overline{1}, \overline{0})$

4. $(\overline{n}, \overline{0}, \overline{0}) \xrightarrow{4,\mu_1(1-p)} (\overline{(n-1)}, \overline{0}, \overline{0})$

There exists the following correspondence in my system $((n, 0), 0, 0)$ with actions: τ, $fill$, a, a_0.
And these are the 4 possible transactions of state:

1. $((n, 0), 0, 0) \xrightarrow{\tau,\lambda_1} ((n+1, 0), 0, 0)$

2. $((n, 0), 0, 0) \xrightarrow{fill,\lambda_3} ((n, 1), 0, 1)$

3. $((n, 0), 0, 0) \xrightarrow{a,\mu_1 p} ((n-1, 0), 1, 0)$

4. $((n, 0), 0, 0) \xrightarrow{a_0,\mu_1(1-p)} ((n-1, 0), 0, 0)$

Case 3
From state:
$(\overline{0}, \overline{n}, \overline{0})$

We can have only 3 possible actions with this values: Action 1 and Action 2, Action 6.
And these are the 3 possible transactions of state:

1. $(\overline{0}, \overline{n}, \overline{0}) \xrightarrow{1,\lambda_1} (\overline{1}, \overline{n}, \overline{0})$

2. $(\overline{0}, \overline{n}, \overline{0}) \xrightarrow{2,\lambda_3} (\overline{0}, \overline{n}, \overline{1})$

3. $(\overline{0}, \overline{n}, \overline{0}) \xrightarrow{6,\mu_2} (\overline{0}, \overline{(n-1)}, \overline{0})$

There exists the following correspondence in my system $((0,0), n, 0)$ with actions: τ, $fill$, τ.
And these are the 3 possible transactions of state:

1. $((0,0), n, 0) \xrightarrow{\tau,\lambda_1} ((1,0), n, 0)$

2. $((0,0), n, 0) \xrightarrow{fill,\lambda_3} ((0,1), n, 1)$

3. $((0,0), n, 0) \xrightarrow{\tau,\mu_2} ((0,0), n-1, 0)$

Case 4
From state:
$(\overline{0}, \overline{0}, \overline{n})$

We can have only 3 possible actions with this values: Action 1 and Action 2, Action 7.
And these are the 3 possible transactions of state:

1. $(\overline{0}, \overline{0}, \overline{n}) \xrightarrow{1,\lambda_1} (\overline{1}, \overline{0}, \overline{n})$

2. $(\overline{0}, \overline{0}, \overline{n}) \xrightarrow{2,\lambda_3} (\overline{0}, \overline{0}, \overline{(n+1)})$

3. $(\overline{0}, \overline{0}, \overline{n}) \xrightarrow{7,\mu_3} (\overline{0}, \overline{0}, \overline{(n-1)})$

There exists the following correspondence in my system $((0,n), 0, n)$ with actions: τ, $fill$, $empty$.
And these are the 3 possible transactions of state:

1. $((0,n), 0, n) \xrightarrow{\tau,\lambda_1} ((1,n), 0, n)$

2. $((0,n),0,n) \overset{fill,\lambda_3}{\longrightarrow} ((0,n+1),0,n+1)$

3. $((0,n),0,n) \overset{empty,\mu_3}{\longrightarrow} ((0,n-1),0,n-1)$

Case 5
From state:
$(\overline{n_1},\overline{n_2},\overline{0})$

We can have only 5 possible actions with this values: Action 1 and Action 2, Action 3, Action 4, Action 6.
And these are the 5 possible transactions of state:

1. $(\overline{n_1},\overline{n_2},\overline{0}) \overset{1,\lambda_1}{\longrightarrow} (\overline{n_1+1},\overline{n_2},\overline{0})$

2. $(\overline{n_1},\overline{n_2},\overline{0}) \overset{2,\lambda_3}{\longrightarrow} (\overline{n_1},\overline{n_2},\overline{1})$

3. $(\overline{n_1},\overline{n_2},\overline{0}) \overset{3,\mu_1 p}{\longrightarrow} (\overline{n-1},\overline{n_2+1},\overline{0})$

4. $(\overline{n_1},\overline{n_2},\overline{0}) \overset{4,\mu_1(1-p)}{\longrightarrow} (\overline{n_1-1},\overline{n_2},\overline{0})$

5. $(\overline{n_1},\overline{n_2},\overline{0}) \overset{6,\mu_2}{\longrightarrow} (\overline{n_1},\overline{n_2-1},\overline{0})$

There exists the following correspondence in my system $((n_1,0),n_2,0)$ with actions: τ, $fill$, a, a_0, τ.
And these are the 3 possible transactions of state:

1. $((n_1,0),n_2,0) \overset{\tau,\lambda_1}{\longrightarrow} ((n_1+1,0),n_2,0)$

2. $((n_1,0),n_2,0) \overset{fill,\lambda_3}{\longrightarrow} ((n_1,1),n_2,1)$

3. $((n_1,0),n_2,0) \overset{a,\mu_1 p}{\longrightarrow} ((n_1-1,0),n_2+1,0)$

4. $((n_1, 0), n_2, 0) \overset{a_0, \mu_1(1-p)}{\longrightarrow} ((n_1 - 1, 0), n_2, 0)$

5. $((n_1, 0), n_2, 0) \overset{\tau, \mu_2}{\longrightarrow} ((n_1, 0), n_2 - 1, 0)$

Case 6
From state:
$(\overline{n_1}, \overline{0}, \overline{n_3})$

We can have only 5 possible actions with this values: Action 1 and Action 2, Action 3, Action 5, Action 7.
And these are the 5 possible transactions of state:

1. $(\overline{n_1}, \overline{0}, \overline{n_3}) \overset{1, \lambda_1}{\longrightarrow} (\overline{n_1 + 1}, \overline{0}, \overline{n_3})$

2. $(\overline{n_1}, \overline{0}, \overline{n_3}) \overset{2, \lambda_3}{\longrightarrow} (\overline{n_1}, \overline{0}, \overline{n_3 + 1})$

3. $(\overline{n_1}, \overline{0}, \overline{n_3}) \overset{3, \mu_1 p}{\longrightarrow} (\overline{n - 1}, \overline{1}, \overline{n_3})$

4. $(\overline{n_1}, \overline{0}, \overline{n_3}) \overset{5, \mu_1(1-p)}{\longrightarrow} (\overline{n_1 - 1}, \overline{1}, \overline{n_3 - 1})$

5. $(\overline{n_1}, \overline{0}, \overline{n_3}) \overset{7, \mu_3}{\longrightarrow} (\overline{n_1}, \overline{0}, \overline{n_3 - 1})$

There exists the following correspondence in my system $((n_1, n_3), 0, n_3)$ with actions: τ, $fill$, a, a_1, $empty$.
And these are the 5 possible transactions of state:

1. $((n_1, n_3), 0, n_3) \overset{\tau, \lambda_1}{\longrightarrow} ((n_1 + 1, n_3), 0, n_3)$

2. $((n_1, n_3), 0, n_3) \overset{fill, \lambda_3}{\longrightarrow} ((n_1, n_3 + 1), 0, n_3 + 1)$

3. $((n_1, n_3), 0, n_3) \overset{a, \mu_1 p}{\longrightarrow} ((n_1 - 1, n_3), 1, n_3)$

4. $((n_1, n_3), 0, n_3) \overset{a_1, \mu_1(1-p)}{\longrightarrow} ((n_1 - 1, n_3 - 1), 1, n_3 - 1)$

5. $((n_1, n_3), 0, n_3) \overset{empty, \mu_3}{\longrightarrow} ((n_1, n_3 - 1), 0, n_3 - 1)$

Case 7
From state:
$(\overline{0}, \overline{n_2}, \overline{n_3})$

We can have only 4 possible actions with this values: Action 1 and Action 2, Action 6, Action 7.
And these are the 4 possible transactions of state:

1. $(\overline{0}, \overline{n_2}, \overline{n_3}) \overset{1, \lambda_1}{\longrightarrow} (\overline{1}, \overline{n_2}, \overline{n_3})$

2. $(\overline{0}, \overline{n_2}, \overline{n_3}) \overset{2, \lambda_3}{\longrightarrow} (\overline{0}, \overline{n_2}, \overline{n_3 + 1})$

3. $(\overline{0}, \overline{n_2}, \overline{n_3}) \overset{6, \mu_2}{\longrightarrow} (\overline{0}, \overline{n_2 - 1}, \overline{n_3})$

4. $(\overline{0}, \overline{n_2}, \overline{n_3}) \overset{7, \mu_3}{\longrightarrow} (\overline{0}, \overline{n_2}, \overline{n_3 - 1})$

There exists the following correspondence in my system $((0, n_3), n_2, n_3)$ with actions: τ, $fill$, a, τ, $empty$.
And these are the 5 possible transactions of state:

1. $((0, n_3), n_2, n_3) \overset{\tau, \lambda_1}{\longrightarrow} ((1, n_3), n_2, n_3)$

2. $((0, n_3), n_2, n_3) \overset{fill, \lambda_3}{\longrightarrow} ((0, n_3 + 1), n_2, n_3 + 1)$

3. $((0, n_3), n_2, n_3) \overset{\tau, \mu_2}{\longrightarrow} ((0, n_3), n_2 - 1, n_3)$

4. $((0, n_3), n_2, n_3) \overset{empty, \mu_3}{\longrightarrow} ((0, n_3 - 1), n_2, n_3 - 1)$

And with this I have proven the Base Case.

Inductive Step:

For inductive hypothesis we have the general case n and I have to show that from here I can reach the same states $n+1$ as explained in the introduction of this section:

From general n state:
$(\overline{n_1}, \overline{n_2}, \overline{n_3})$

We can have only 6 possible actions with this values: Action 1 and Action 2, Action 3, Action 5, Action 6, Action 7.
And these are the 6 possible transactions of state:

1. $(\overline{n_1}, \overline{n_2}, \overline{n_3}) \xrightarrow{1,\lambda_1} (\overline{n_1+1}, \overline{n_2}, \overline{n_3})$

2. $(\overline{n_1}, \overline{n_2}, \overline{n_3}) \xrightarrow{2,\lambda_3} (\overline{n_1}, \overline{n_2}, \overline{n_3+1})$

3. $(\overline{n_1}, \overline{n_2}, \overline{n_3}) \xrightarrow{3,\mu_1 p} (\overline{n-1}, \overline{n_2+1}, \overline{n_3})$

4. $(\overline{n_1}, \overline{n_2}, \overline{n_3}) \xrightarrow{5,\mu_1(1-p)} (\overline{n_1-1}, \overline{n_2+1}, \overline{n_3-1})$

5. $(\overline{n_1}, \overline{n_2}, \overline{n_3}) \xrightarrow{6,\mu_2} (\overline{0}, \overline{n_2-1}, \overline{n_3})$

6. $(\overline{n_1}, \overline{n_2}, \overline{n_3}) \xrightarrow{7,\mu_3} (\overline{0}, \overline{n_2}, \overline{n_3-1})$

There exists the following correspondence in my system $((n_1, n_3), n_2, n_3)$ with actions: τ, $fill$, a, a_1, τ, $empty$.
And these are the 6 possible transactions of state:

1. $((n_1, n_3), n_2, n_3) \xrightarrow{\tau,\lambda_1} ((n_1+1, n_3), n_2, n_3)$

2. $((n_1, n_3), n_2, n_3) \xrightarrow{fill,\lambda_3} ((n_1, n_3+1), n_2, n_3+1)$

3. $((n_1, n_3), n_2, n_3) \xrightarrow{a,\mu_1 p} ((n_1 - 1, n_3), n_2 + 1, n_3)$

4. $((n_1, n_3), n_2, n_3) \xrightarrow{a_1,\mu_1(1-p)} ((n_1 - 1, n_3 - 1), n_2 + 1, n_3 - 1)$

5. $((n_1, n_3), n_2, n_3) \xrightarrow{\tau,\mu_2} ((n_1, n_3), n_2 - 1, n_3)$

6. $((n_1, n_3), n_2, n_3) \xrightarrow{empty,\mu_3} ((n_1, n_3 - 1), n_2, n_3 - 1)$

I have proven the Inductive Case, so from inductive hypothesis, $\forall$ states $(\overline{n_1}, \overline{n_2}, \overline{n_3})$ there exists a correspondence to a state $((n_1, n_3), n_2, n_3)$ which can perform the same actions, has the same behavior and has the equivalent "predecessor" states. In this way I have proven the $\Rightarrow$ direction of equivalence between my model and the original system.

$\Leftarrow$:

It is quite straight the reverse demonstration, since the actions done by my system are the all and only ones permitted in those cases. I will show anyway a short version of demonstration:

Base Case:

We have the following 7 base cases in my encoded system:

- $((0,0),0,0)$: all processes are empty.
- $((n,0),0,0)$: process R_1 has n positive costumers, R_2 and R_3 are empty.
- $((0,0),n,0)$: process R_2 has n positive costumers, R_1 and R_3 are empty.
- $((0,n),0,n)$: process R_3 has n positive costumers, R_1 and R_2 are empty.
- $((n_1,0),n_2,0)$: process R_1 has n_1 positive costumers, R_2 has n_2 positive costumers and R_3 is empty.
- $((n_1,n_3),0,n_3)$: process R_1 has n_1 positive costumers, R_3 has n_3 positive costumers and R_2 is empty.
- $((0,n_3),n_2,n_3)$: process R_2 has n_2 positive costumers, R_3 has n_3 positive costumers and R_1 is empty.

Case 1
From state:
$((0,0),0,0)$

We can have only 2 possible actions with this values: τ and $fill$.
And these are the 2 possible transactions of state:
$((0,0),0,0) \xrightarrow{\tau,\lambda_1} ((1,0),0,0)$
$((0,0),0,0) \xrightarrow{fill,\lambda_3} ((0,1),0,1)$

There exists the following correspondence in the original system $(\overline{0},\overline{0},\overline{0})$ with actions: Action 1 and Action 2.
And these are the 2 possible transactions of state:
$(\overline{0},\overline{0},\overline{0}) \xrightarrow{1,\lambda_1} (\overline{1},\overline{0},\overline{0})$

$(\overline{0},\overline{0},\overline{0}) \xrightarrow{2,\lambda_3} (\overline{0},\overline{0},\overline{1})$

Case 2
From state:
$((n,0),0,0)$

We can have only 4 possible actions with this values: τ,$fill$,a,a_0.
And these are the 4 possible transactions of state:
$((n,0),0,0) \xrightarrow{\tau,\lambda_1} ((n+1,0),0,0)$
$((n,0),0,0) \xrightarrow{fill,\lambda_3} ((n,1),0,1)$
$((n,0),0,0) \xrightarrow{a,\mu_1 p} ((n-1,0),1,0)$
$((n,0),0,0) \xrightarrow{a_0,\mu_1(1-p)} ((n-1,0),0,0)$

There exists the following correspondence in the original system $(\overline{n},\overline{0},\overline{0})$ with actions: Action 1 and Action 2, Action 3, Action 4.
And these are the 4 possible transactions of state:
$(\overline{n},\overline{0},\overline{0}) \xrightarrow{1,\lambda_1} (\overline{(n+1)},\overline{0},\overline{0})$
$(\overline{n},\overline{0},\overline{0}) \xrightarrow{2,\lambda_3} (\overline{n},\overline{0},\overline{1})$
$(\overline{n},\overline{0},\overline{0}) \xrightarrow{3,\mu_1 p} (\overline{(n-1)},\overline{1},\overline{0})$
$(\overline{n},\overline{0},\overline{0}) \xrightarrow{4,\mu_1(1-p)} (\overline{(n-1)},\overline{0},\overline{0})$

Case 3
From state:
$((0,0),n,0)$

We can have only 3 possible actions with this values: τ,$fill$,τ.
And these are the 3 possible transactions of state:
$((0,0),n,0) \xrightarrow{\tau,\lambda_1} ((1,0),n,0)$
$((0,0),n,0) \xrightarrow{fill,\lambda_3} ((0,1),n,1)$
$((0,0),n,0) \xrightarrow{\tau,\mu_2} ((0,0),n-1,0)$

There exists the following correspondence in the original system $(\overline{0},\overline{n},\overline{0})$ with actions: Action 1 and Action 2, Action 6.
And these are the 3 possible transactions of state:

$(\overline{0}, \overline{n}, \overline{0}) \xrightarrow{1,\lambda_1} (\overline{1}, \overline{n}, \overline{0})$
$(\overline{0}, \overline{n}, \overline{0}) \xrightarrow{2,\lambda_3} (\overline{0}, \overline{n}, \overline{1})$
$(\overline{0}, \overline{n}, \overline{0}) \xrightarrow{6,\mu_2} (\overline{0}, \overline{(n-1)}, \overline{0})$

Case 4
From state:
$((0, n), 0, n)$

We can have only 3 possible actions with this values: τ,$fill$,$empty$.
And these are the 3 possible transactions of state:
$((0, n), 0, n) \xrightarrow{\tau,\lambda_1} ((1, n), 0, n)$
$((0, n), 0, n) \xrightarrow{fill,\lambda_3} ((0, n+1), 0, n+1)$
$((0, n), 0, n) \xrightarrow{empty,\mu_3} ((0, n-1), 0, n-1)$

There exists the following correspondence in the original system $(\overline{0}, \overline{0}, \overline{n})$ with actions: Action 1 and Action 2, Action 7.
And these are the 3 possible transactions of state:
$(\overline{0}, \overline{0}, \overline{n}) \xrightarrow{1,\lambda_1} (\overline{1}, \overline{0}, \overline{n})$
$(\overline{0}, \overline{0}, \overline{n}) \xrightarrow{2,\lambda_3} (\overline{0}, \overline{0}, \overline{(n+1)})$
$(\overline{0}, \overline{0}, \overline{n}) \xrightarrow{7,\mu_3} (\overline{0}, \overline{0}, \overline{(n-1)})$

Case 5
From state:
$((n_1, 0), n_2, 0)$

We can have only 5 possible actions with this values: τ,$fill$,a,a_0,τ.
And these are the 5 possible transactions of state:
$((n_1, 0), n_2, 0) \xrightarrow{\tau,\lambda_1} ((n_1 + 1, 0), n_2, 0)$
$((n_1, 0), n_2, 0) \xrightarrow{fill,\lambda_3} ((n_1, 1), n_2, 1)$
$((n_1, 0), n_2, 0) \xrightarrow{a,\mu_1 p} ((n_1 - 1, 0), n_2 + 1, 0)$
$((n_1, 0), n_2, 0) \xrightarrow{a_0,\mu_1(1-p)} ((n_1 - 1, 0), n_2, 0)$
$((n_1, 0), n_2, 0) \xrightarrow{\tau,\mu_2} ((n_1, 0), n_2 - 1, 0)$

There exists the following correspondence in the original system $(\overline{n_1}, \overline{n_2}, \overline{0})$

with actions: Action 1 and Action 2, Action 3, Action 4, Action 6.
And these are the 5 possible transactions of state:
$(\overline{n_1}, \overline{n_2}, \overline{0}) \xrightarrow{1,\lambda_1} (\overline{n_1+1}, \overline{n_2}, \overline{0})$
$(\overline{n_1}, \overline{n_2}, \overline{0}) \xrightarrow{2,\lambda_3} (\overline{n_1}, \overline{n_2}, \overline{1})$
$(\overline{n_1}, \overline{n_2}, \overline{0}) \xrightarrow{3,\mu_1 p} (\overline{n-1}, \overline{n_2+1}, \overline{0})$
$(\overline{n_1}, \overline{n_2}, \overline{0}) \xrightarrow{4,\mu_1(1-p)} (\overline{n_1-1}, \overline{n_2}, \overline{0})$
$(\overline{n_1}, \overline{n_2}, \overline{0}) \xrightarrow{6,\mu_2} (\overline{n_1}, \overline{n_2-1}, \overline{0})$

Case 6
From state:
$((n_1, n_3), 0, n_3)$

We can have only 5 possible actions with this values: τ,$fill$,a,a_1,$empty$.
And these are the 5 possible transactions of state:
$((n_1, n_3), 0, n_3) \xrightarrow{\tau,\lambda_1} ((n_1+1, n_3), 0, n_3)$
$((n_1, n_3), 0, n_3) \xrightarrow{fill,\lambda_3} ((n_1, n_3+1), 0, n_3+1)$
$((n_1, n_3), 0, n_3) \xrightarrow{a,\mu_1 p} ((n_1-1, n_3), 1, n_3)$
$((n_1, n_3), 0, n_3) \xrightarrow{a_1,\mu_1(1-p)} ((n_1-1, n_3-1), 1, n_3-1)$
$((n_1, n_3), 0, n_3) \xrightarrow{empty,\mu_3} ((n_1, n_3-1), 0, n_3-1)$

There exists the following correspondence in the original system $(\overline{n_1}, \overline{0}, \overline{n_3})$
with actions: Action 1 and Action 2, Action 3, Action 5, Action 7.
And these are the 5 possible transactions of state:
$(\overline{n_1}, \overline{0}, \overline{n_3}) \xrightarrow{1,\lambda_1} (\overline{n_1+1}, \overline{0}, \overline{n_3})$
$(\overline{n_1}, \overline{0}, \overline{n_3}) \xrightarrow{2,\lambda_3} (\overline{n_1}, \overline{0}, \overline{n_3+1})$
$(\overline{n_1}, \overline{0}, \overline{n_3}) \xrightarrow{3,\mu_1 p} (\overline{n-1}, \overline{1}, \overline{n_3})$
$(\overline{n_1}, \overline{0}, \overline{n_3}) \xrightarrow{5,\mu_1(1-p)} (\overline{n_1-1}, \overline{1}, \overline{n_3-1})$
$(\overline{n_1}, \overline{0}, \overline{n_3}) \xrightarrow{7,\mu_3} (\overline{n_1}, \overline{0}, \overline{n_3-1})$

Case 7
From state:
$((0, n_3), n_2, n_3)$

We can have only 4 possible actions with this values: τ,$fill$,τ,$empty$.
And these are the 4 possible transactions of state:
$((0, n_3), n_2, n_3) \xrightarrow{\tau,\lambda_1} ((1, n_3), n_2, n_3)$
$((0, n_3), n_2, n_3) \xrightarrow{fill,\lambda_3} ((0, n_3 + 1), n_2, n_3 + 1)$
$((0, n_3), n_2, n_3) \xrightarrow{\tau,\mu_2} ((0, n_3), n_2 - 1, n_3)$
$((0, n_3), n_2, n_3) \xrightarrow{empty,\mu_3} ((0, n_3 - 1), n_2, n_3 - 1)$

There exists the following correspondence in the original system $(\overline{0}, \overline{n_2}, \overline{n_3})$ with actions: Action 1 and Action 2, Action 6, Action 7.
And these are the 5 possible transactions of state:
$(\overline{0}, \overline{n_2}, \overline{n_3}) \xrightarrow{1,\lambda_1} (\overline{1}, \overline{n_2}, \overline{n_3})$
$(\overline{0}, \overline{n_2}, \overline{n_3}) \xrightarrow{2,\lambda_3} (\overline{0}, \overline{n_2}, \overline{n_3 + 1})$
$(\overline{0}, \overline{n_2}, \overline{n_3}) \xrightarrow{6,\mu_2} (\overline{0}, \overline{n_2 - 1}, \overline{n_3})$
$(\overline{0}, \overline{n_2}, \overline{n_3}) \xrightarrow{7,\mu_3} (\overline{0}, \overline{n_2}, \overline{n_3 - 1})$

And with this I have proven the Base Case.

Inductive Step:

For inductive hypothesis we have the general case n and I have to show from here I can reach the same states $n+1$ as explained in the introduction of this section to prove the validity of general case:

From state:
$((n_1, n_3), n_2, n_3)$

We can have only 6 possible actions with this values: τ,$fill$,a,a_1,τ,$empty$.
And these are the 6 possible transactions of state:
$((n_1, n_3), n_2, n_3) \xrightarrow{\tau,\lambda_1} ((n_1+1, n_3), n_2, n_3)$
$((n_1, n_3), n_2, n_3) \xrightarrow{fill,\lambda_3} ((n_1, n_3+1), n_2, n_3+1)$
$((n_1, n_3), n_2, n_3) \xrightarrow{a,\mu_1 p} ((n_1-1, n_3), n_2+1, n_3)$
$((n_1, n_3), n_2, n_3) \xrightarrow{a_1,\mu_1(1-p)} ((n_1-1, n_3-1), n_2+1, n_3-1)$
$((n_1, n_3), n_2, n_3) \xrightarrow{\tau,\mu_2} ((n_1, n_3), n_2-1, n_3)$
$((n_1, n_3), n_2, n_3) \xrightarrow{empty,\mu_3} ((n_1, n_3-1), n_2, n_3-1)$

There exists the following correspondence in the original system $(\overline{n_1}, \overline{n_2}, \overline{n_3})$ with actions: Action 1 and Action 2, Action 3, Action 5, Action 6, Action 7.
And these are the 6 possible transactions of state:
$(\overline{n_1}, \overline{n_2}, \overline{n_3}) \xrightarrow{1,\lambda_1} (\overline{n_1+1}, \overline{n_2}, \overline{n_3})$
$(\overline{n_1}, \overline{n_2}, \overline{n_3}) \xrightarrow{2,\lambda_3} (\overline{n_1}, \overline{n_2}, \overline{n_3+1})$
$(\overline{n_1}, \overline{n_2}, \overline{n_3}) \xrightarrow{3,\mu_1 p} (\overline{n-1}, \overline{n_2+1}, \overline{n_3})$
$(\overline{n_1}, \overline{n_2}, \overline{n_3}) \xrightarrow{5,\mu_1(1-p)} (\overline{n_1-1}, \overline{n_2+1}, \overline{n_3-1})$
$(\overline{n_1}, \overline{n_2}, \overline{n_3}) \xrightarrow{6,\mu_2} (\overline{0}, \overline{n_2-1}, \overline{n_3})$
$(\overline{n_1}, \overline{n_2}, \overline{n_3}) \xrightarrow{7,\mu_3} (\overline{0}, \overline{n_2}, \overline{n_3-1})$

I have proven the Inductive Case, so from inductive hypothesis, $\forall$ states $((n_1, n_3), n_2, n_3)$ there exists a correspondence to a state $(\overline{n_1}, \overline{n_2}, \overline{n_3})$ which can perform the same actions, has the same behavior and has the equivalent "predecessor" states. In this way I have proven the $\Leftarrow$ direction of equivalence between my model and the original system.

From this, for induction reason, my system represents all and only the states of original G-network with trigger, and also their behaviors are equivalent.

I can now state that with my encoding of this system **I have modeled a G-Network using only the definitions of PEPA language**, so I prove that its expressivity can cover also this kind of G-Network.

Chapter 6

Product Form

6.1 Product Form of G-network with Trigger

In this section I will analyze the derivation graph of both Double Index and Warden solutions and search where they don't satisfy the conditions of the product form theorem. After that, I modify the double index solution in order to satisfy those conditions but without changing its behavior that I already proved it is equal to the original one.
I add a new state in my processes, the *Phantom State* F. The purpose of this stratagem is in fact to satisfy the two conditions without changing the behavior thanks to its property of unreachability. The main property of state F is exactly that *from the starting normal states, every possible and permissible transaction in the system cannot lead in any way to the phantom state* F.
After this phase I try to balance the equations of my processes and find out if their reversal rate are always constant. I found some inconclusive solutions which put too many bound. Then to avoiding those bounds I developed new methods for balance the equations and, after their implementation, I finally succeeded to balance them.
After that I verify the constancies of reverse rate and the computation of the steady state distribution. The last step is the comparison between my new model with the original one to state that its behavior is still the same as the original one.

6.1.1 Conditions for the Product Form Theorem

In order to have the product form solution, a system in PEPA must satisfy the following conditions:

1. Each passive action must be outgoing from each state of the processes in which it is used at least one time, i.e. each state must have an outgoing transaction of type of the passive action and with $\top$ rate
2. Each active action must be ingoing to each state of the processes in which it is used at least one time, i.e. each state must have an ingoing transaction of type of the active action and with a non-$\top$ rate
3. all reverse rates of the same action type in a process must be equal, i.e. all the reverse rates of a passive action with $\top$ rate within a process must be all equal

Another important fact is that all these referred actions are in a cooperation set.

To satisfy these conditions I will adopt the following techniques:

1. Outgoing Passive Actions: I add an exiting transaction from all states that don't have it, of that type of passive action (it doesn't matter if it is a self-loop)
2. Ingoing Active Actions: I add an entering transaction to all states that don't have it, of that type of active action (it doesn't matter if it is a self-loop)
3. I will balance the equations of my processes and after that I compute the reverse rates of all the passive actions.

Now I pass to the analysis of the derivation graphs.

6.1.2 Analysis of Derivative Graphs - Double Index (DI) Solution

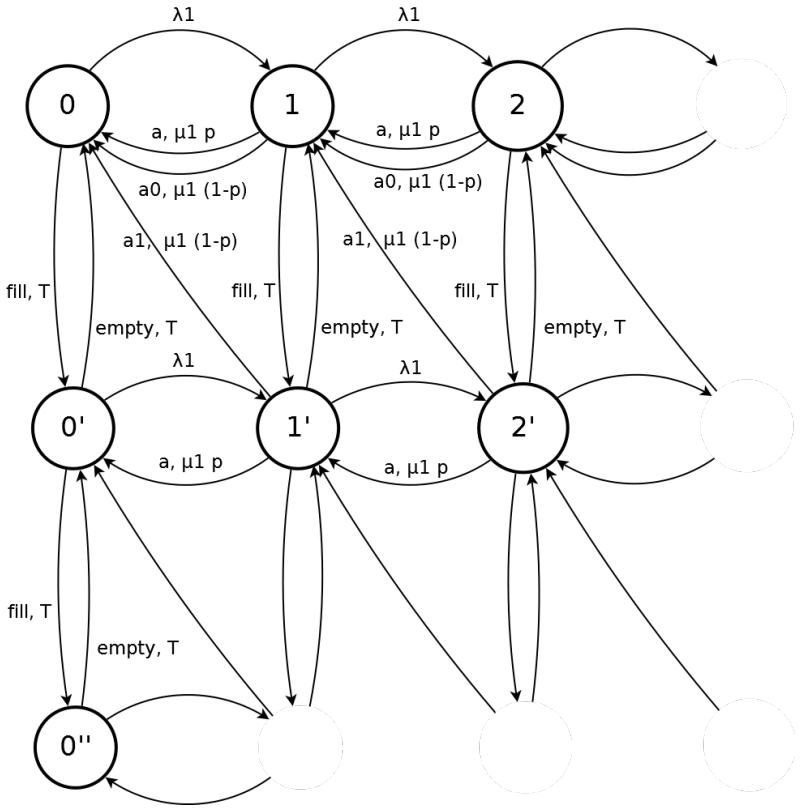

Figure 6.1: DI solution of G-network with Trigger: Derivation Graph of process R_1

Analyzing this graph we have the following information regarding action types:

- Active Actions: a, a_0, a_1 (τ doesn't cooperate with anyone)
- Passive Actions: $fill$, $empty$

And the following information is about the first two conditions of the product form theorem:

- State P_0:
  Ingoing Active Actions: a, a_0, a_1
  Missing Active Actions: none
  Outgoing Passive Actions: $fill$
  Missing Passive Actions: $empty$

- State P_n:
  Ingoing Active Actions: a, a_0, a_1
  Missing Active Actions: none
  Outgoing Passive Actions: $fill$
  Missing Passive Actions: $empty$

- State $P_0 =^i$:
  Ingoing Active Actions: a, a_1
  Missing Active Actions: a_0
  Outgoing Passive Actions: $fill$, $empty$
  Missing Passive Actions: none

- State $P_n =^i$:
  Ingoing Active Actions: a, a_1
  Missing Active Actions: a_0
  Outgoing Passive Actions: $fill$, $empty$
  Missing Passive Actions: none

Analyzing this graph we have the following information regarding action types:

- Active Actions: none (τ doesn't cooperate with anyone)
- Passive Actions: a, a_1

And the following information is about the first two conditions of the product form theorem:

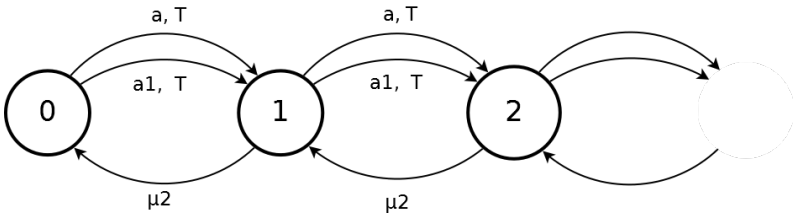

Figure 6.2: DI solution of G-network with Trigger: Derivation Graph of process R_2

- State Q_0:
  Ingoing Active Actions: none
  Missing Active Actions: none
  Outgoing Passive Actions: a, a_1
  Missing Passive Actions: none

- State Q_n:
  Ingoing Active Actions: none
  Missing Active Actions: none
  Outgoing Passive Actions: a, a_1
  Missing Passive Actions: none

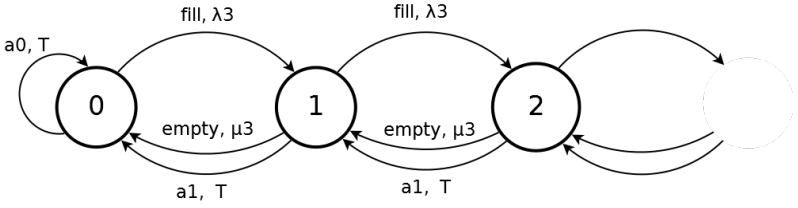

Figure 6.3: DI solution of G-network with Trigger: Derivation Graph of process R_3

Analyzing this graph we have the following information regarding action

types:

- Active Actions: $fill$, $empty$
- Passive Actions: a_0, a_1

And the following information is about the first two conditions of the product form theorem:

- State R_0:
  Ingoing Active Actions: $empty$
  Missing Active Actions: $fill$
  Outgoing Passive Actions: a_0
  Missing Passive Actions: a_1

- State R_n:
  Ingoing Active Actions: $fill$, $empty$
  Missing Active Actions: none
  Outgoing Passive Actions: a_1
  Missing Passive Actions: a_0

6.1.3 Analysis of Derivative Graphs - Warden Process (WaP) Solution

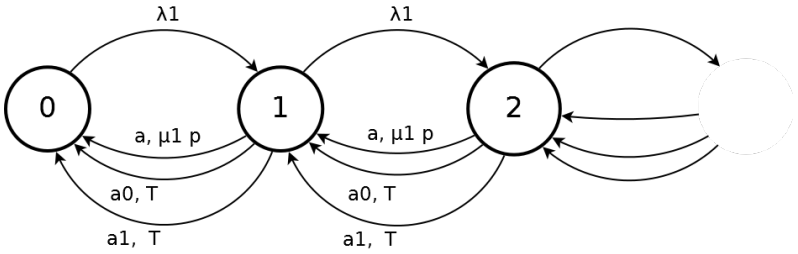

Figure 6.4: WaP solution of G-network with Trigger: Derivation Graph of process R_1

Analyzing this graph we have the following information regarding action types:

- Active Actions: a (τ doesn't cooperate with anyone)
- Passive Actions: a_0, a_1

And the following information is about the first two conditions of the product form theorem:

- State P_0:
  Ingoing Active Actions: a
  Missing Active Actions: none
  Outgoing Passive Actions: none
  Missing Passive Actions: a_0, a_1

- State P_n:
  Ingoing Active Actions: a
  Missing Active Actions: none
  Outgoing Passive Actions: a_0, a_1
  Missing Passive Actions: none

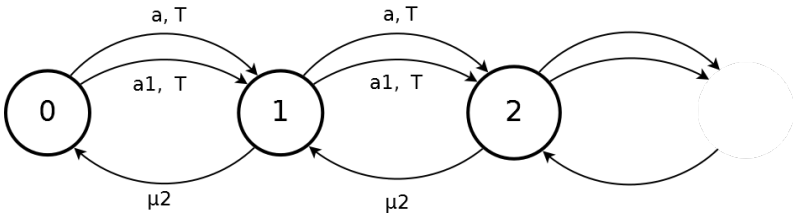

Figure 6.5: WaP solution of G-network with Trigger: Derivation Graph of process R_2

Analyzing this graph we have the following information regarding action types:

- Active Actions: none (τ doesn't cooperate with anyone)
- Passive Actions: a, a_1

And the following information is about the first two conditions of the product form theorem:

- State Q_0:
  Ingoing Active Actions: none
  Missing Active Actions: none
  Outgoing Passive Actions: a, a_1
  Missing Passive Actions: none

- State Q_n:
  Ingoing Active Actions: none
  Missing Active Actions: none
  Outgoing Passive Actions: a, a_1
  Missing Passive Actions: none

Analyzing this graph we have the following information regarding action types:

- Active Actions: $fill$, $empty$

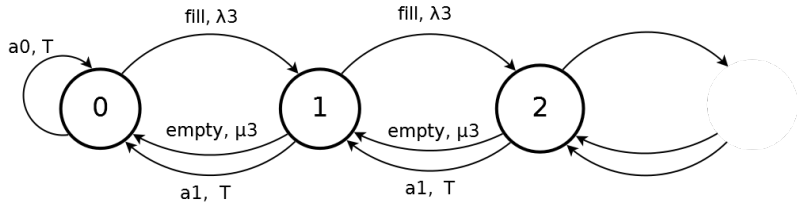

Figure 6.6: WaP solution of G-network with Trigger: Derivation Graph of process R_3

- Passive Actions: a_0, a_1

And the following information is about the first two conditions of the product form theorem:

- State R_0:
  Ingoing Active Actions: $empty$
  Missing Active Actions: $fill$
  Outgoing Passive Actions: a_0
  Missing Passive Actions: a_1

- State R_n:
  Ingoing Active Actions: $fill$, $empty$
  Missing Active Actions: none
  Outgoing Passive Actions: a_1
  Missing Passive Actions: a_0

Analyzing this graph we have the following information regarding action types:

- Active Actions: a_0, a_1
- Passive Actions: $fill$, $empty$

And the following information is about the first two conditions of the product form theorem:

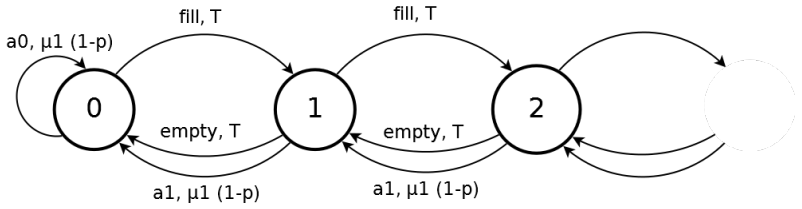

Figure 6.7: WaP solution of G-network with Trigger: Derivation Graph of process C

- State C_0:
  Ingoing Active Actions: a_0, a_1
  Missing Active Actions: none
  Outgoing Passive Actions: $fill$
  Missing Passive Actions: $empty$

- State C_n:
  Ingoing Active Actions: a_1
  Missing Active Actions: a_0
  Outgoing Passive Actions: $fill$, $empty$
  Missing Passive Actions: none

6.1.4 Derivative Graphs Summary

I will write now the states in which an action does not satisfy the conditions. $(\leftarrow)$ represents an outgoing passive problem, $(\rightarrow)$ represents an ingoing active problem.
This is the summary of the two solutions:

DI solution:

Process	Action a	Action a_0	Action a_1	Action $fill$	Action $empty$
P		$(\rightarrow) : P_0^i, P_n^i$			$(\leftarrow) : P_0, P_n$
Q					
R		$(\leftarrow) : R_n$	$(\leftarrow) : R_0$	$(\rightarrow) : R_0$	

WaP solution:

Process	Action a	Action a_0	Action a_1	Action $fill$	Action $empty$
P		$(\leftarrow) : P_0$	$(\leftarrow) : P_0$		
Q					
R		$(\leftarrow) : R_n$	$(\leftarrow) : R_0$	$(\rightarrow) : R_0$	
C		$(\rightarrow) : C_n$			$(\leftarrow) : C_0$

We can see from this summary, as I expected, that there is a big problem in the satisfaction of the first two conditions of the product form theorem in the warden process solution. In P_0 of WaP, we should put a a_0 action to satisfy the condition of outgoing passive actions. If I add any transition from state P_0 to anywhere with type a_0 it will synchronize with both R_0 and C_0 altering the behavior of my system. This make the behavior of this encoding solution different from the original one and this cannot be acceptable.

6.1.5 Addition of Phantom State and Missing "Conditions" Actions

6.1.5.1 Addition in Double Index Solution

In this section I try to add those missing actions to my double index solution. I call them *impossible actions* because they will never occur due to the fact that they will never cooperate with the correspondent active/passive actions in another process. I use the annotation of putting a ! before their name to discern them from the normal actions.
In order to do this, I add a state F, called *Phantom state*. This state will never be reachable, because all the actions which lead to it are impossible actions. In this way all the actions from the unreachable state F will be impossible too, because they will never occur thanks to the non reachability of F.
This is one possible solution:
We can see from this case that F is unreachable because in state P_0, the process R_0 doesn't have any positive costumer and so it cannot perform an $empty$ action. This ensure that $empty$ in P_n with $n \geq 0$ doesn't occur and so F is unreachable.
I add also in the phantom state, the action All which represent all possible actions, in this way the first two conditions of product form theorem are satisfied also for F state. However, they are still impossible.

In the following image, I try also to split the F state in more phantom states. The meaning is the same, F states are still unreachable and their ingoing and outgoing actions still impossible. In this way I show that it is the same to have a single phantom state F or a set of them ($F_0, F_1, ..., F_n$).
Moreover, each F state has its action All and like in the previous example, they satisfy the two conditions and they are still impossible. Another important fact is that *I can add whichever action from any phantom state F_i to any other phantom state F_j* (included $i = j$ with a self-loop) *because they will result still impossible*.
In both solutions I add the passive $empty$ action in all P_n with $n \geq 0$ and a_0 in all P_n^i with $n \geq 0$ and $i > 0$.

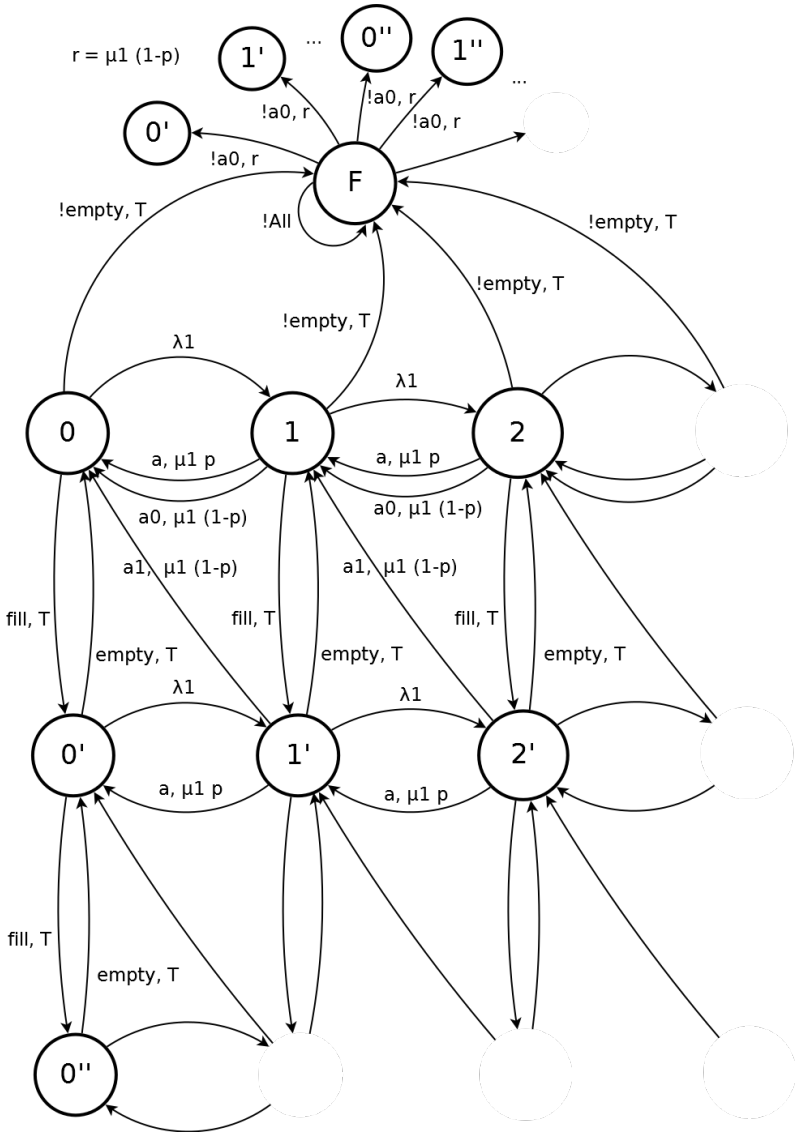

Figure 6.8: R_1 with impossible actions (DI)

Also in this case I add the missing actions with the use of impossible actions and phantom state F. Moreover, it is the same to add only one or n states F

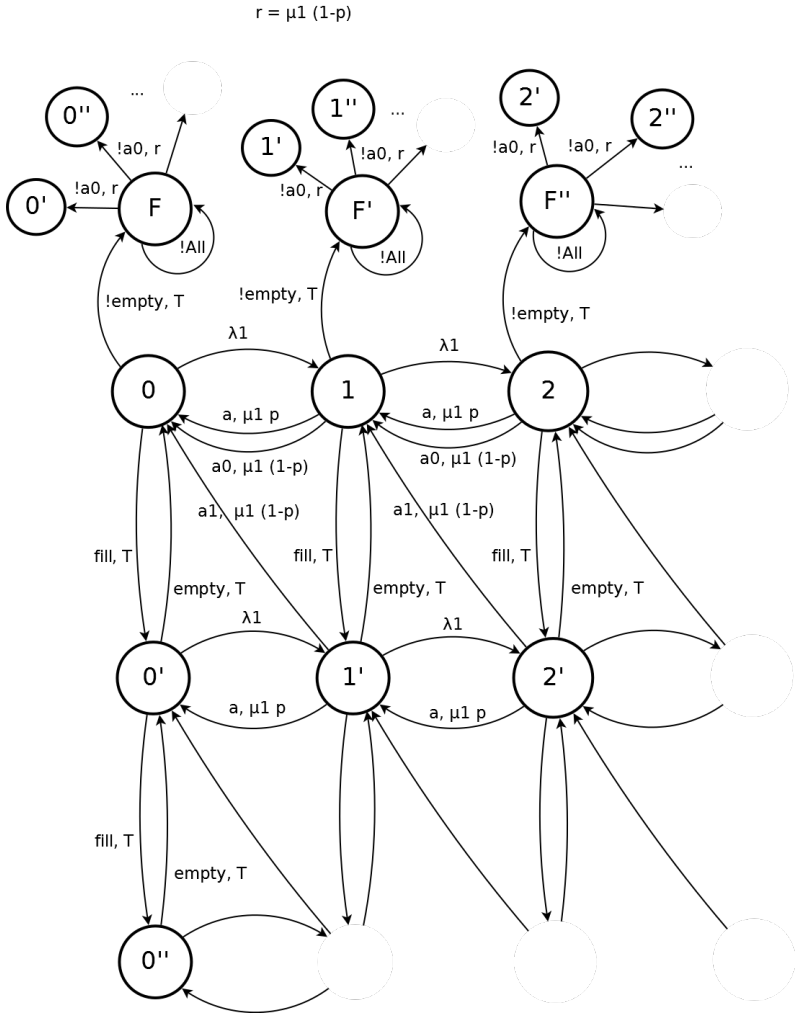

Figure 6.9: R_1 with impossible actions (DI), alternative

like in the previous example and also All represents the same set of actions.

A possible alternative in this case is the use of only one F phantom state for the missing actions of state 0 and transform all the others outing missing actions of states $1, ..., n$ into self-loops. The state 0 needs to have an ingoing missing action and in order to not synchronize with anyone I must add a

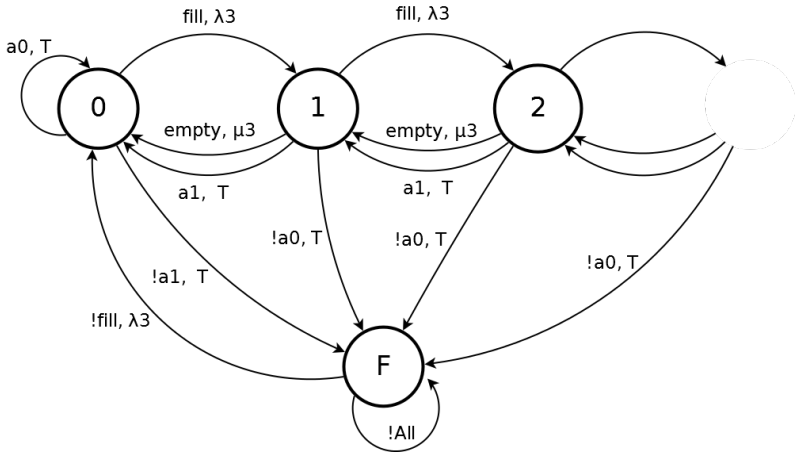

Figure 6.10: R_3 with impossible actions (DI)

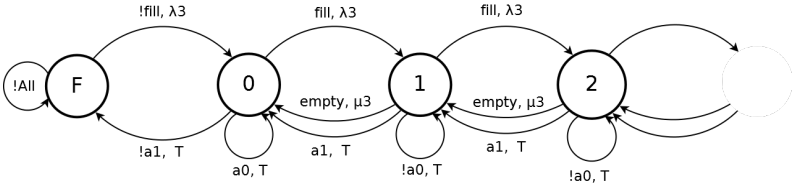

Figure 6.11: R_3 with impossible actions (DI), alternative

phantom state F in this case contrarily to other ones.

6.1.5.2 Addition in Warden Process Solution

Even if I don't choose the Warden Process solution as my solution for the product form theorem, I will show that is it possible to add the missing actions for state R_3 and C and that the problem relies entirely in the process R_1 and the synchronization of its missing action a_0.

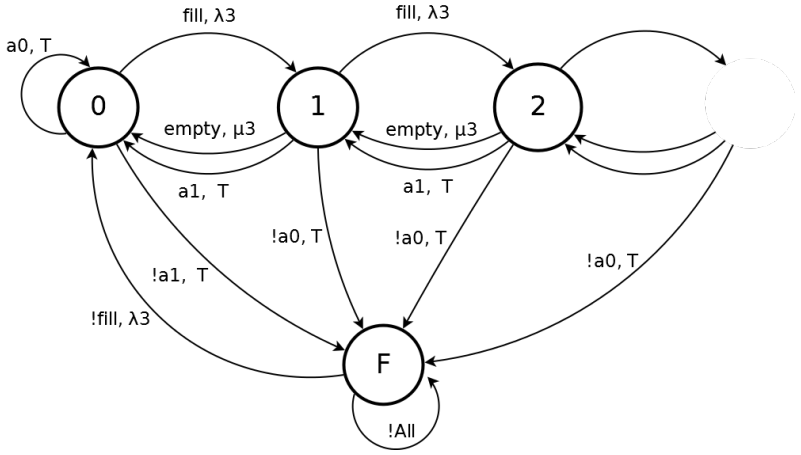

Figure 6.12: R_3 with impossible actions (WaP)

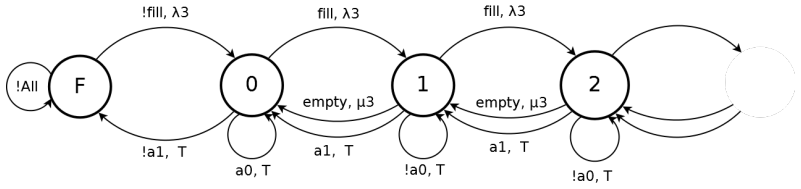

Figure 6.13: R_3 with impossible actions (WaP), alternative

Like in the previous example, also in this case the process R_3 can be transformed in this way. It has also two alternatives of addition of missing actions. For what concerns the process C, it has a specular behavior with respect to R_3 in fact state 0 has an outgoing action, which can be transformed into a self-loop, and ingoing action for all other states, which need $1, ..., n$ phantom states F

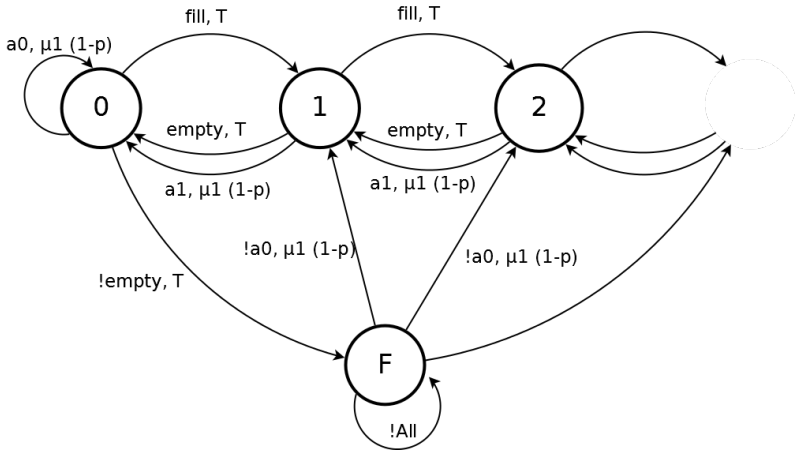

Figure 6.14: C with impossible actions (WaP)

6.1.6 Phantom state F and Impossible Actions

6.1.6.1 Impossible Actions

The definition of an **impossible actions** is:
An action whose type is part of a cooperation set and never synchronize with anyone (i.e. it will never occur) is called a impossible action.
This means that impossible actions will never cooperate with their correspondent active or passive actions in other processes and they are usually marked with a ! before their name to discern them from the normal actions.

To better understand them, I make an example. Considering the following encoding of a system in PEPA:

$$P = \begin{cases} P_0 = (a, \top).P_1 \\ P_1 = (b, \mu_1).P_0 + (c, \mu_2).P_0 \end{cases}$$

$$Q = \begin{cases} Q_0 = (a, \mu).Q_1 + (c, \top).Q_1 \\ Q_1 = (b, \top).Q_0 \end{cases}$$

$$S \stackrel{def}{=} P \underset{a,b,c}{\bowtie} Q$$

And their derivations graphs are the following two:

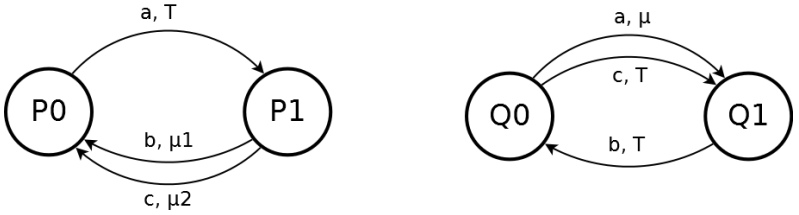

Figure 6.15: Example of Impossible Action c in processes P and Q

We can notice that all actions are in the cooperative set and a,b have the same direction in both processes (from 0 to 1 and from 1 to 0, respectively) but c has an opposite behavior ($1 - 0$ direction in P and $0 - 1$ direction in Q). Furthermore a is passive in P and active in Q and b,c are active in P and passive in Q.
In this way, considering that the starting states are P_0 and Q_0 (or we will incur in a dead-lock situation), when the two processes cooperate in general system S we have the following states:

- General State S_0: corresponds to the states of processes P_0 and Q_0
- General State S_1: corresponds to the states of processes P_1 and Q_1

Moreover, starting from the cooperation of P_0 with Q_0, process S will have the following behavior:

- In S_0:
  - P_0 has only a passive action a and must wait for Q_0
  - Q_0 could perform a and c but both must synchronize with same type of actions in P_0
  - Result: Q_0 can only synchronize in action a with P_0 and it proceeds in Q_1 and similarly P_0 proceeds in P_1 always performing action a

  In this way S_0 becomes state S_1 and c in this case can't occur.

- In S_1:
  - P_1 could perform b and c but both must synchronize with same type of actions in Q_1
  - Q_1 has only a passive action b and must wait for P_1
  - Result: P_1 can only synchronize in action b with Q_1 and it proceeds in P_0 and similarly Q_1 proceeds in Q_0 always performing action b

  In this way S_1 becomes state S_0 and c also in this case can't occur.

- We return to state S_0 so no other behaviors can occur.

With this behavior, the derivative graph of S is the following:

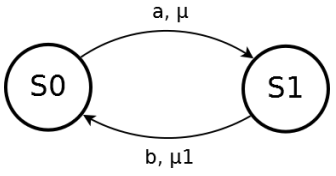

Figure 6.16: Example of Impossible Action c in cooperation process of P and Q

We can see how c never occurs in both directions (i.e. either from S_0 to S_1 or from S_1 to S_0). In this case c never synchronizes and subsequently never occurs so in this systems both c actions are *impossible actions*.

6.1.6.2 Phantom States F

The definition of an **phantom state** F is:
A non-starting state in which all ingoing actions are impossible actions is called a phantom state.
This means that the entering actions in a phantom state F can be of three types (even if still impossible):

- Impossible actions from other kind of states
- Self-loops of impossible actions
- Impossible actions from other phantom states

This definition ensure that a phantom state is unreachable and not present in the derivative set of the process because all the actions which lead to it are impossible actions. In this way all outgoing actions of F are impossible too because of the state will never be a future state of the process and subsequently all its actions will never occur.
So, due to the unreachability of phantom state F, *all the actions, both ingoing and outgoing, of a phantom state F are consequently impossible actions*.
This leads the definition of a **set of phantom states** F:
A set of states in which there isn't any starting state and all actions (ingoing and outgoing) from outside the set are impossible actions is called a set of phantom states $F = (F_1, ..., F_n)$.
Similarly to the previous definition all outgoing action from a set of phantom states are still impossible actions and they can be of three type:

- Actions to outside the set
- Self-loops of actions
- Impossible actions between phantom states

To better understand them, I make an example. Considering the following encoding of a system in PEPA:

$$P = \begin{cases} P_0 = (a, \top).P_1 + (b, \mu_1).F_P \\ P_1 = (b, \mu_1).P_0 + (c, \mu_2).F_P \\ F_P = (a, \top).P_1 \end{cases}$$

$$Q = \begin{cases} Q_0 = (a, \mu).Q_1 + (c, \top).F_Q \\ Q_1 = (b, \top).Q_0 + (a, \mu).F_Q \\ F_Q = (a, \mu).Q_1 \end{cases}$$

$$S \stackrel{def}{=} P \underset{a,b,c}{\bowtie} Q$$

And their derivations graphs are the following two:

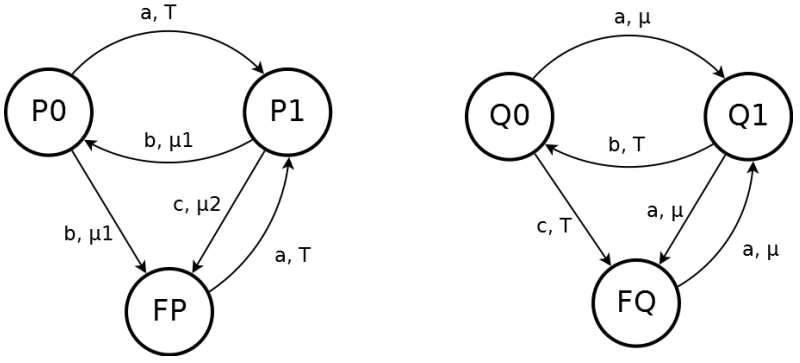

Figure 6.17: Example of Phantom State F in processes P and Q

We can notice that all actions are in the cooperative set but apart a from P_0 and Q_0 and b from P_1 and Q_1, the others actions don't have the same direction in both processes. Furthermore a is passive in P and active in Q and b,c are active in P and passive in Q.
In this way, considering that the starting states are P_0 and Q_0 (or we will incur in a dead-lock situation), when the two processes cooperate in general system S we have the following states:

- General State S_0: corresponds to the states of processes P_0 and Q_0

- General State S_1: corresponds to the states of processes P_1 and Q_1

Moreover, starting from the cooperation of P_0 with Q_0, process S will have the following behavior:

- In S_0:
  - P_0 could perform a and b but both must synchronize with same type of actions in Q_0
  - Q_0 could perform a and c but both must synchronize with same type of actions in P_0
  - Result: Q_0 can only synchronize in action a with P_0 and it proceeds in Q_1 and similarly P_0 proceeds in P_1 always performing action a

  In this way S_0 becomes state S_1 and b and c in this case can't occur.

- In S_1:
  - P_1 could perform b and c but both must synchronize with same type of actions in Q_1
  - Q_1 could perform b and a but both must synchronize with same type of actions in P_1
  - Result: P_1 can only synchronize in action b with Q_1 and it proceeds in P_0 and similarly Q_1 proceeds in Q_0 always performing action b

  In this way S_1 becomes state S_0 and a and c also in this case can't occur.

- We return to state S_0 so no other behaviors can occur.

With this behavior, the derivative graph of S is the following:

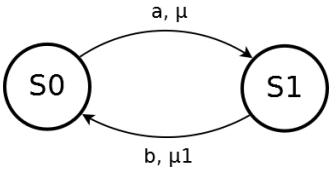

Figure 6.18: Example of Phantom State F in cooperation process of P and Q

The final result process is the same in the two previous examples, this mean that both impossible actions and phantom state F didn't affect the main system S.
It is important to notice that F_P and F_Q could have synchronize and performed action a but since P, Q and in general S have never reached both states F_P and F_Q then their a actions never synchronize and occur so they are impossible actions.
We can see how F is never reached from any state (i.e. neither from S_0 nor from S_1). In this case no action reach F so in this systems both F_P and F_Q states are *phantom states*.

6.2 Global Balance Equations of DI Solution

In this part of my thesis I will try to balance the equations of the processes and to find the reversal rates of the passive actions. The global balance equations taking into analysis are the ones generated in the Double index solution.
The equations are in product-form and must be at equilibrium because they represent the steady state distribution of each state of the processes. Each equation represents a different state and it has all the rates of outgoing actions of its state in their left part and all the rates of ingoing actions in the right one. If they are all balanced I can then compute the steady state distribution in product form of the general system. The last thing to do is to verify that all reverse rate are the same, each one with respect to others of its kind of action. Moreover, in this computation, each of the $\top$ rates will be substituted by a variable (one for each type of action), until I found the exact value derived from the rate of its active counterpart.
First of all, I start the analysis of the states of my actual model to find the equations generated from the product-form. In this phase I have to make a prediction of the solution of the system because in a ergodic system the solution in unique and if my solution is correct then, in this way, it is the only and unique solution of the entire system.
I follow this way because my system and the original G-Network have the same behavior and subsequently are equivalent and have the same underlying Markov chain and because the original G-Network can have the product form, also my model has it. I am trying to prove that we can compute the product form, starting from the global balance equations of my model.
In the second part I show two trials in my balancing operations, the first will result in some bounds on the rate of the model, but this cannot be acceptable because there cannot be further bounds on them. Also the second one doesn't lead to a result but it introduces a nice way of thinking: *first try to balance the system without impossible actions and phantom states and then add those features*.
In the last part of this phase I try to compensate those added actions and states with other inverse actions. In this way:

- for each outgoing impossible action from a normal state I will add an outgoing action from a phantom state with the same rate
- for each ingoing impossible action form a phantom state to a normal state I will add an outgoing action from that normal state to the phantom one with the same rate

- actions in phantom states will be compensated with other impossible actions between them. Since they are all impossible, I can add any amount of them without worrying to alter the cooperation mechanism or the system behavior

This operation will cause some problem in the synchronization of those impossible actions, in particular with the passive actions. This because a passive action must cooperate with an active one with the same type, in some other process.
The problem can be summarize in the following steps:

1. To satisfy the two conditions of the product form theorem: I add an ingoing impossible action from a phantom state F to a normal state (in the case of *active ingoing action*)

2. To compensate its rate I have to add an action in the other direction (i.e. from normal state to phantom one): the outgoing action has to be *passive* if we want to keep the conditions of theorem satisfied

3. For the cooperation of that passive action, there must be an active counterpart in some other process

4. Other processes have all action name "occupied", if can't add an action of those types without altering the system behavior

5. I add another type for that compensating passive action but in order to do this I have to add an active action of the same new type in another process

This introduction of a new type of impossible compensating actions leads to a problem of iterative addition if we consider only two processes where to add those actions:

1. I add an active action of the new type to cooperate with the passive one, it has to be ingoing in all states to satisfy the conditions of the theorem

2. That new ingoing action has to be compensated so I have to add a passive action and it must be outgoing to satisfy the theorem

3. That passive action must cooperate with an active one in another process

4. This addiction in the other process restart the iteration of addition of point 1

In this way there is no ending in the chain of those addictions of actions. To solve this problem I studied a solution which involve 3 processes and not only 2. This method is able to close the previous chains of additions but in order to not alter another existing process and its behavior, I add a new type of process: *the collapsed process* C.
This collapsed state C is a state in which there is only a starting state C_0 surrounded by phantom states and connected by impossible actions. This new kind of process permit to vent the surplus rates of the balancing phase, from the first impossible actions (i.e. the ones needed to satisfy the two conditions of the product form theorem). I will also prove that impossible actions can always be added in a way such that they are cooperating. Furthermore also the active impossible actions must always cooperate with a passive counterpart, otherwise without synchronization they can occur and then they will no more impossible.
The most difficult part of the balancing of global equations is the one related to process R_1 because it is composed by two parts, the one with information about its state of process and the one related to R_3. For this reason its derivative graph has two dimensions and consequently its balancing is more complicate.

6.2.1 Analysis of States of process R_1

Process R_1 is composed by four main types of states:

1. State P_0
2. State P_n with $n > 0$
3. State P_0^i with $i > 0$
4. State P_n^i with $n > 0$ and with $i > 0$

All of its state can be attributed to one of this 4 types.
In the following analysis I will study all and only the actions necessary to satisfy the first two conditions of the product form theorem. I could only add some new actions or new type of them but I cannot eliminate any of those in the following analysis.
I suppose that the product form of R_1 will be the following:

$$\Pi_{rc}$$

Where:
c refers to the number of column of the state (i.e. number of positive costumers in R_1)
r refers to the number of row of the state (i.e. number of positive costumers in R_3)
This because in states of R_1 are stored both information about R_1 and R_3.
Then, thanks to the product form, it can be expressed as:

$$\Pi_{rc} = \alpha \Pi_r \cdot \Pi_c$$

Finally, it can be further decomposed in the following way:

$$\Pi_{rc} = \alpha \Pi_r \cdot \Pi_c = \alpha \rho_1^r \rho_2^c$$

Where α its a generic constant greater than 0 and can be freely chosen and ρ_1 and ρ_2 have to be found.
And this is my hypothesis about it.

6.2.1.1 Analysis of P_0

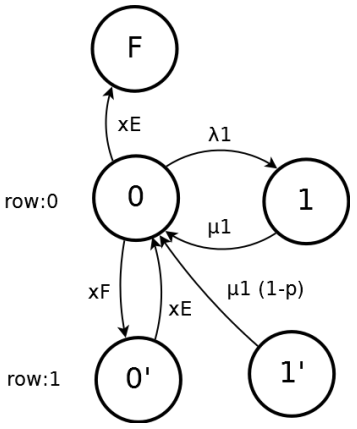

Figure 6.19: State P_0 (DI) and its neighbors states

The general case of process R_1:

$\Pi_{rc} = \alpha\Pi_r \cdot \Pi_c = \alpha\rho_1^r\rho_2^c$

In P_0 has $r = 0$ and $c = 0$.

Moreover, its actions with neighbor states are:

- Ingoing Actions:
  - From state P_1: a with rate $\mu_1 p$ and a_0 with rate $\mu_1(1-p)$, total rate μ_1
  - From state P_1': a_1 with rate $\mu_1(1-p)$
  - From state P_0': $empty$ with rate $\top$, in this case substituted by x_e
- Outgoing Actions:
  - To state P_1: τ with rate λ_1
  - To state P_0': $fill$ with rate $\top$, in this case substituted by x_f
  - To state F: $empty$ with rate $\top$, in this case substituted by x_e

Then the global balance equation for P_0 is:

$$\Pi_{00}(r_{out}) = \Pi_{01}(r_{01}) + \Pi_{11}(r_{11}) + \Pi_{10}(r_{10})$$

Where: r_{out} is the sum of outgoing rates of P_0 and r_{ij} are the sum of ingoing rate from state P_j^i, and it can be decomposed in the following way:

$$\Pi_{00}(\overbrace{x_e + \lambda_1 + x_f}^{\text{Outgoing Actions}}) = \Pi_{01}\mu_1 + \Pi_{11}\mu_1(1-p) + \Pi_{10}x_e$$

$$\begin{aligned}
\frac{\cancel{\Pi_{00}}}{\cancel{\Pi_{00}}}(x_e + \lambda_1 + x_f) &= \frac{\Pi_{01}}{\Pi_{00}}\mu_1 + \frac{\Pi_{11}}{\Pi_{00}}\mu_1(1-p) + \frac{\Pi_{10}}{\Pi_{00}}x_e \\
&= \frac{\cancel{\alpha}\rho_1^{\cancel{0}}\rho_2^1}{\cancel{\alpha}\rho_1^0\rho_2^0}\mu_1 + \frac{\cancel{\alpha}\rho_1^1\rho_2^1}{\cancel{\alpha}\rho_1^0\rho_2^0}\mu_1(1-p) + \frac{\cancel{\alpha}\rho_1^1\rho_2^{\cancel{0}}}{\cancel{\alpha}\rho_1^0\rho_2^0}x_e \\
&= \rho_2\mu_1 + \rho_1\rho_2\mu_1(1-p) + \rho_1 x_e
\end{aligned}$$

In conclusion:
$(x_e + \lambda_1 + x_f) = \rho_2\mu_1 + \rho_1\rho_2\mu_1(1-p) + \rho_1 x_e$

6.2.1.2 Analysis of P_n

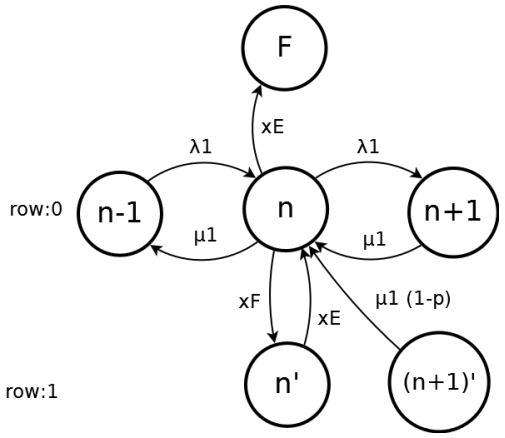

Figure 6.20: State P_n (DI) and its neighbors states

The general case of process R_1:

$\Pi_{rc} = \alpha\Pi_r \cdot \Pi_c = \alpha\rho_1^r\rho_2^c$

In P_n has $r = 0$ and $c = n$ with $n > 0$.

Moreover, its actions with neighbor states are:

- Ingoing Actions:
  - From state P_{n+1}: a with rate $\mu_1 p$ and a_0 with rate $\mu_1(1-p)$, total rate μ_1
  - From state P'_{n+1}: a_1 with rate $\mu_1(1-p)$
  - From state P'_n: $empty$ with rate $\top$, in this case substituted by x_e
  - From state P_{n-1}: τ with rate λ_1
- Outgoing Actions:
  - To state P_{n+1}: τ with rate λ_1
  - To state P'_n: $fill$ with rate $\top$, in this case substituted by x_f

- To state P_{n-1}: a with rate $\mu_1 p$ and a_0 with rate $\mu_1(1-p)$, total rate μ_1
- To state F: $empty$ with rate $\top$, in this case substituted by x_e

Then the global balance equation for P_n is then:

$$\Pi_{0n}(\mu_1 + x_e + \lambda_1 + x_f) = \Pi_{0\,n+1}\mu_1 + \Pi_{1\,n+1}\mu_1(1-p) + \Pi_{1n}x_e + \Pi_{0\,n-1}\lambda_1$$

$$\begin{aligned}\frac{\cancel{\Pi_{0n}}}{\cancel{\Pi_{0n}}}(\mu_1 + x_e + \lambda_1 + x_f) &= \frac{\Pi_{0\,n+1}}{\Pi_{0n}}\mu_1 + \frac{\Pi_{1\,n+1}}{\Pi_{0n}}\mu_1(1-p) + \frac{\Pi_{1n}}{\Pi_{0n}}x_e + \frac{\Pi_{0\,n-1}}{\Pi_{0n}}\lambda_1 \\ &= \frac{\cancel{\alpha}\cancel{\rho_1^0}\rho_2^{n+1}}{\cancel{\alpha\rho_1^0}\rho_2^n}\mu_1 + \frac{\cancel{\alpha}\rho_1^1\rho_2^{n+1}}{\cancel{\alpha\rho_1^0}\rho_2^n}\mu_1(1-p) + \frac{\cancel{\alpha}\rho_1^1\rho_2^n}{\cancel{\alpha\rho_1^0}\rho_2^n}x_e + \frac{\cancel{\alpha}\rho_1^0\rho_2^{n-1}}{\cancel{\alpha}\rho_1^0\rho_2^n}\lambda_1 \\ &= \rho_2\mu_1 + \rho_1\rho_2\mu_1(1-p) + \rho_1 x_e + \frac{1}{\rho_2}\lambda_1\end{aligned}$$

In conclusion:
$(\mu_1 + x_e + \lambda_1 + x_f) = \rho_2\mu_1 + \rho_1\rho_2\mu_1(1-p) + \rho_1 x_e + \frac{1}{\rho_2}\lambda_1$

6.2.1.3 Analysis of P_0^i

The general case of process R_1:

$\Pi_{rc} = \alpha\Pi_r \cdot \Pi_c = \alpha\rho_1^r\rho_2^c$

In P_0^i has $r = i$ and $c = 0$ with $i > 0$.

Moreover, its actions with neighbor states are:

- Ingoing Actions:
  - From state P_1^i: a with rate $\mu_1 p$
  - From state P_1^{i+1}: a_1 with rate $\mu_1(1-p)$
  - From state P_0^{i+1}: $empty$ with rate $\top$, in this case substituted by x_e
  - From state P_0^{i-1}: $fill$ with rate $\top$, in this case substituted by x_f

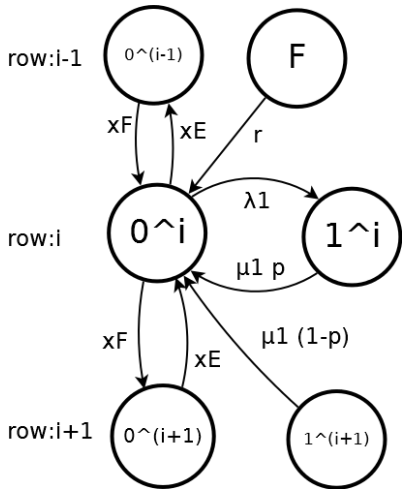

Figure 6.21: State P_0^i (DI) and its neighbors states

  - From state F: a_0 with general rate r since it is an impossible action it can be anything

- Outgoing Actions:
  - To state P_1^i: τ with rate λ_1
  - To state P_0^{i+1}: $fill$ with rate $\top$, in this case substituted by x_f
  - To state P_0^{i-1}: $empty$ with rate $\top$, in this case substituted by x_e

Then the global balance equation for P_0^i is then:

$$\Pi_{i0}(x_e + \lambda_1 + x_f) = \Pi_{i1}\mu_1 p + \Pi_{i+1\,1}\mu_1(1-p) + \Pi_{i+1\,0}x_e + \Pi_{i-1\,0}x_f + \Pi_F r$$

$$\begin{aligned}
\cancel{\Pi_{i0}}(x_e + \lambda_1 + x_f) &= \frac{\Pi_{i1}}{\Pi_{i0}}\mu_1 p + \frac{\Pi_{i+1\,1}}{\Pi_{i0}}\mu_1(1-p) + \frac{\Pi_{i+1\,0}}{\Pi_{i0}}x_e + \frac{\Pi_{i-1\,0}}{\Pi_{i0}}x_f + \frac{\Pi_F}{\Pi_{i0}}r \\
&= \frac{\alpha\cancel{\rho_1^i}\rho_2}{\alpha\cancel{\rho_1^i}\rho_2^0}\mu_1 p + \frac{\alpha\cancel{\rho_1^{i+1}}\rho_2}{\alpha\cancel{\rho_1^i}\rho_2^0}\mu_1(1-p) + \frac{\alpha\cancel{\rho_1^{i+1}}\cancel{\rho_2^0}}{\alpha\cancel{\rho_1^i}\rho_2^0}x_e + \frac{\alpha\cancel{\rho_1^{i-1}}\rho_2^0}{\alpha\cancel{\rho_1^i}\cancel{\rho_2^0}}x_f + \frac{\Pi_F}{\Pi_{i0}}r \\
&= \rho_2\mu_1 p + \rho_1\rho_2\mu_1(1-p) + \rho_1 x_e + \frac{1}{\rho_1}x_f + \frac{\Pi_F}{\Pi_{i0}}r
\end{aligned}$$

In conclusion:
$(x_e + \lambda_1 + x_f) = \rho_2\mu_1 p + \rho_1\rho_2\mu_1(1-p) + \rho_1 x_e + \frac{1}{\rho_1}x_f + \frac{\Pi_F}{\Pi_{i0}}r$

6.2.1.4 Analysis of P_n^i

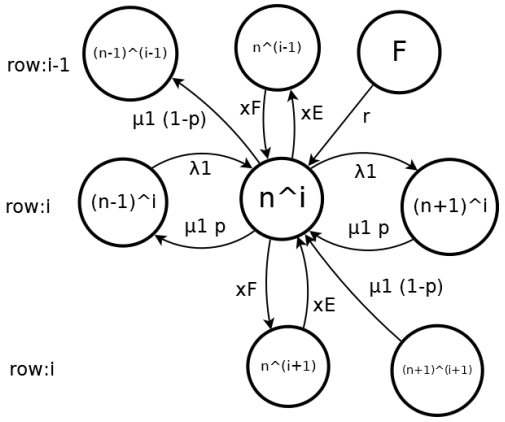

Figure 6.22: State P_n^i (DI) and its neighbors states

The general case of process R_1:

$\Pi_{rc} = \alpha\Pi_r \cdot \Pi_c = \alpha\rho_1^r\rho_2^c$

In P_n^i has $r = i$ and $c = n$ with $n > 0$ and $i > 0$.

Moreover, its actions with neighbor states are:

- Ingoing Actions:
  - From state P_{n+1}^i: a with rate $\mu_1 p$
  - From state P_{n+1}^{i+1}: a_1 with rate $\mu_1(1-p)$
  - From state P_n^{i+1}: $empty$ with rate $\top$, in this case substituted by x_e
  - From state P_{n-1}^i: τ with rate $\lambda_1 p$
  - From state P_n^{i-1}: $fill$ with rate $\top$, in this case substituted by x_f

  - From state F: a_0 with general rate r since it is an impossible action it can be anything

- Outgoing Actions:

  - To state P^i_{n+1}: τ with rate λ_1
  - To state P^{i+1}_n: $fill$ with rate $\top$, in this case substituted by x_f
  - To state P^i_{n-1}: a with rate $\mu_1 p$
  - To state P^{i-1}_{n-1}: a_1 with rate $\mu_1(1-p)$
  - To state P^{i-1}_n: $empty$ with rate $\top$, in this case substituted by x_e

Then the global balance equation for P_n is then:

$$\begin{aligned}\Pi_{in}(\mu_1 p + \mu_1(1-p) + x_e + \lambda_1 + x_f) = \quad & \Pi_{i\,n+1}\mu_1 p + \Pi_{i+1\,n+1}\mu_1(1-p) + \Pi_{i+1\,n}x_e + \\ & + \Pi_{i\,n-1}\lambda_1 + \Pi_{i-1\,n}x_f + \Pi_F r\end{aligned}$$

$$\begin{aligned}\cancel{\Pi_{in}}(\mu_1 + x_e + \lambda_1 + x_f) &= \frac{\Pi_{i\,n+1}}{\Pi_{in}}\mu_1 p + \frac{\Pi_{i+1\,n+1}}{\Pi_{in}}\mu_1(1-p) + \frac{\Pi_{i+1\,n}}{\Pi_{in}}x_e + \frac{\Pi_{i\,n-1}}{\Pi_{in}}\lambda_1 + \frac{\Pi_{i-1\,n}}{\Pi_{in}}x_f + \frac{\Pi_F}{\Pi_{in}}r \\ &= \frac{\cancel{\alpha}\rho_1^{\cancel{i}}\rho_2^{\cancel{n}+1}}{\cancel{\alpha}\rho_1^{\cancel{i}}\rho_2^{n}}\mu_1 p + \frac{\cancel{\alpha}\rho_1^{\cancel{i}+1}\rho_2^{\cancel{n}+1}}{\cancel{\alpha}\rho_1^{\cancel{i}}\rho_2^{n}}\mu_1(1-p) + \frac{\cancel{\alpha}\rho_1^{\cancel{i}+1}\rho_2^{\cancel{n}}}{\cancel{\alpha}\rho_1^{\cancel{i}}\rho_2^{n}}x_e + \frac{\cancel{\alpha}\rho_1^{i}\rho_2^{\cancel{n-1}}}{\cancel{\alpha}\rho_1^{\cancel{i}}\rho_2^{\cancel{n}}}\lambda_1 + \frac{\cancel{\alpha}\rho_1^{\cancel{i-1}}\rho_2^{\cancel{n}}}{\cancel{\alpha}\rho_1^{\cancel{i}}\rho_2^{\cancel{n}}}x_f + \\ &= \rho_2\mu_1 p + \rho_1\rho_2\mu_1(1-p) + \rho_1 x_e + \frac{1}{\rho_2}\lambda_1 + \frac{1}{\rho_1}x_f + \frac{\Pi_F}{\Pi_{in}}r\end{aligned}$$

In conclusion:
$(\mu_1 + x_e + \lambda_1 + x_f) = \rho_2\mu_1 p + \rho_1\rho_2\mu_1(1-p) + \rho_1 x_e + \frac{1}{\rho_2}\lambda_1 + \frac{1}{\rho_1}x_f + \frac{\Pi_F}{\Pi_{in}}r$

6.2.2 First Trial - Balancing of Equations of Process R_1 (DI)

Now I will try to balance the global balance equations of R_1.

6.2.2.1 Summary

The summary of the global balance equations of process R_1 is the following:

$$\begin{cases} x_e + \lambda_1 + x_f & = & \rho_2\mu_1 & + & \rho_1\rho_2\mu_1(1-p) & + & \rho_1 x_e & & & & & & & & s \\ x_e + \lambda_1 + x_f + \mu_1 & = & \rho_2\mu_1 & + & \rho_1\rho_2\mu_1(1-p) & + & \rho_1 x_e & + & \frac{1}{\rho_2}\lambda_1 & & & & & & s \\ x_e + \lambda_1 + x_f & = & \rho_2\mu_1 p & + & \rho_1\rho_2\mu_1(1-p) & + & \rho_1 x_e & + & & & \frac{1}{\rho_1}x_f & + & \frac{\Pi_F}{\Pi_{i0}}r & & s \\ x_e + \lambda_1 + x_f + \mu_1 & = & \rho_2\mu_1 p & + & \rho_1\rho_2\mu_1(1-p) & + & \rho_1 x_e & + & \frac{1}{\rho_2}\lambda_1 & + & \frac{1}{\rho_1}x_f & + & \frac{\Pi_F}{\Pi_{in}}r & & s \end{cases}$$

6.2.2.2 Computation

Now I will try to balance them and find the value of the variables:

- ρ_1 and ρ_2 to compute the steady state distribution of the process R_1
- Π_F which can also be $\Pi_{F_{in}}$ for each normal state because this won't alter the general behavior of the process
- r rate of ingoing action from F which can be any number greater than 0

The rate x_e from states P_0 and P_n to phantom state F cannot be any number because it is passive (i.e. its original action has $\top$ rate which will be substituted during a cooperation).

First of all, let's notice the difference between states $P_1 - P_0$:

$$\begin{array}{lcccccccc} x_e + \lambda_1 + x_f + \mu_1 & = & \rho_2\mu_1 & + & \rho_1\rho_2\mu_1(1-p) & + & \rho_1 x_e & + & \frac{1}{\rho_2}\lambda_1 \\ x_e + \lambda_1 + x_f & = & \rho_2\mu_1 & + & \rho_1\rho_2\mu_1(1-p) & + & \rho_1 x_e & & \\ \hline \mu_1 & = & & & & & & & \frac{1}{\rho_2}\lambda_1 \end{array}$$

From this we have then:

$$\mu_1 = \frac{1}{\rho_2}\lambda_1$$
$$\rho_2\mu_1 = \frac{1}{\cancel{\rho_2}}\cancel{\rho_2}\lambda_1$$

$$\rho_2 \frac{\cancel{\mu_1}}{\cancel{\mu_1}} = \frac{\lambda_1}{\mu_1}$$
$$\rho_2 = \frac{\lambda_1}{\mu_1}$$

So ρ_2 must be equal to $\frac{\lambda_1}{\mu_1}$ in all other equations.

For the difference between states $P_0^i - 0$ we can notice that:

$$\begin{array}{lcllllllll}
x_e + \lambda_1 + x_f & = & \rho_2\mu_1 p & + & \rho_1\rho_2\mu_1(1-p) & + & \rho_1 x_e & + & \frac{1}{\rho_1}x_f & + & \frac{\Pi_F}{\Pi_{i0}}r \\
x_e + \lambda_1 + x_f & = & \rho_2\mu_1 & + & \rho_1\rho_2\mu_1(1-p) & + & \rho_1 x_e & & & & \\
\hline
0 & = & -\rho_2\mu_1(1-p) & + & & & & & \frac{1}{\rho_1}x_f & + & \frac{\Pi_F}{\Pi_{i0}}r
\end{array}$$

And similarly for states $P_n^i - 1$ is almost same:

$$\begin{array}{lcllllllllll}
x_e + \lambda_1 + x_f + \mu_1 & = & \rho_2\mu_1 p & + & \rho_1\rho_2\mu_1(1-p) & + & \rho_1 x_e & + & \frac{1}{\rho_2}\lambda_1 & + & \frac{1}{\rho_1}x_f & + & \frac{\Pi_F}{\Pi_{in}}r \\
x_e + \lambda_1 + x_f + \mu_1 & = & \rho_2\mu_1 & + & \rho_1\rho_2\mu_1(1-p) & + & \rho_1 x_e & + & \frac{1}{\rho_2}\lambda_1 & & & & \\
\hline
0 & = & -\rho_2\mu_1(1-p) & + & & & & & & & \frac{1}{\rho_1}x_f & + & \frac{\Pi_F}{\Pi_{in}}r
\end{array}$$

So to balance the equation it has to be that:

$$\frac{1}{\rho_1}x_f + \frac{\Pi_F}{\Pi_{1z}}r = \rho_2\mu_1(1-p)$$

Where $z = 0 \vee 1$

Since that Π_F can be whatever we want and I can have n distinct F, one for each state (because all of them cannot be reached), I choose $\Pi_F = \Pi_{F_{i0}} = \alpha\rho_1^a\rho_2^b$ for Π_{i0} (state P_0^i) and $\Pi_F = \Pi_{F_{in}} = \alpha\rho_1^a\rho_2^{b+n}$ for Π_{in} (state P_n^i). In this way the two cases are the same because:

$$\frac{\Pi_{F_{i0}}}{\Pi_{i0}} = \frac{\alpha\rho_1^a\rho_2^b}{\rho_1^i\rho_2^0} = \frac{\alpha\rho_1^a\rho_2^b}{\rho_1^i\rho_2^0}\frac{\rho_2^n}{\rho_2^n} = \frac{\alpha\rho_1^a\rho_2^{b+n}}{\rho_1^i\rho_2^{0+n}} = \frac{\alpha\rho_1^a\rho_2^{b+n}}{\rho_1^i\rho_2^n} = \frac{\Pi_{F_{in}}}{\Pi_{in}}$$

So:

$$0 = -\rho_2\mu_1(1-p) + \frac{1}{\rho_1}x_f + \frac{\Pi_F}{\Pi_{i0}}r$$
$$\frac{1}{\rho_1}x_f + \frac{\Pi_F}{\Pi_{i0}}r = \rho_2\mu_1(1-p)$$
$$\frac{1}{\rho_1}x_f + \frac{\cancel{\alpha}\rho_1^a\rho_2^b}{\cancel{\alpha}\rho_1^i\cancel{\rho_2^0}}r = \rho_2\mu_1(1-p)$$

I choose a in such a way that $a = i + a'$, in this way we can simplify the equation in the following way:

$$\frac{1}{\rho_1}x_f + \frac{\rho_1^{\cancel{i}+a'}\rho_2^b}{\rho_1^{\cancel{i}}}r = \rho_2\mu_1(1-p)$$
$$\frac{1}{\rho_1}x_f + \rho_1^{a'}\rho_2^b r = \rho_2\mu_1(1-p)$$

If it weren't for x_f, the solution would be easier, so I think I could add an action of type τ from F states corresponding to the first row of normal states (i.e. $F_1, ..., F_n$) to all the states of the first row (i.e. $P_0, ..., P_n$). I have to get something equal to $\frac{1}{\rho_1}x_f$.
For this reason, I add an ingoing action (τ, r') to all these states $P_0, ..., P_n$. In this way I add an action rate on the equation and this subsequently add it in the left part:

$\frac{\Pi_{F_m}}{\Pi_{0m}}r'$ with $m \geq 0$

And this can be reduced in this way:

$$\frac{\Pi_{F_m}}{\Pi_{0m}}r' = \frac{\alpha\rho_1^c\rho_2^d}{\alpha\rho_1^0\rho_2^m}r'$$

Then I choose the values for $c = -1$, $d = m$ and $r' = x_f$ such that:

$$\Pi_{F_m} = \alpha\rho_1^{-1}\rho_2^m$$

From this I get:

$$\frac{\Pi_{F_m}}{\Pi_{0m}} r' = \frac{\cancel{\alpha}\rho_1^{-1}\cancel{\rho_2^m}}{\cancel{\alpha}\rho_1^0\cancel{\rho_2^m}} x_f = \frac{1}{\rho_1} x_f$$

Now we have the following new equations in the difference between $P_0^i - P_0$:

$$\begin{array}{rcl}
x_e + \lambda_1 + x_f &=& \rho_2\mu_1 p + \rho_1\rho_2\mu_1(1-p) + \rho_1 x_e + \frac{1}{\rho_1}x_f + \frac{\Pi_{F_{i0}}}{\Pi_{i0}} r \\
x_e + \lambda_1 + x_f &=& \rho_2\mu_1 + \rho_1\rho_2\mu_1(1-p) + \rho_1 x_e + \frac{1}{\rho_1}x_f \\
\hline
0 &=& -\rho_2\mu_1(1-p) + \frac{\Pi_{F_{i0}}}{\Pi_{i0}} r
\end{array}$$

And the following new equations in the difference between $P_n^i - P_n$:

$$\begin{array}{rcl}
x_e + \lambda_1 + x_f + \mu_1 &=& \rho_2\mu_1 p + \rho_1\rho_2\mu_1(1-p) + \rho_1 x_e + \frac{1}{\rho_2}\lambda_1 + \frac{1}{\rho_1}x_f + \frac{\Pi_{F_{in}}}{\Pi_{in}} r \\
x_e + \lambda_1 + x_f + \mu_1 &=& \rho_2\mu_1 + \rho_1\rho_2\mu_1(1-p) + \rho_1 x_e + \frac{1}{\rho_2}\lambda_1 + \frac{1}{\rho_1}x_f \\
\hline
0 &=& -\rho_2\mu_1(1-p) + \frac{\Pi_{F_{in}}}{\Pi_{in}} r
\end{array}$$

Now to balance the equation it only has to be that (remember that two cases are similar):

$$\frac{\Pi_{F_{i0}}}{\Pi_{i0}} r = \rho_2\mu_1(1-p)$$

And I said I choose the $\Pi_{F_{i0}}$ such that $\Pi_{F_{i0}} = \alpha\rho_1^{i+a'}\rho_2^b$ and then:

$$\frac{\cancel{\alpha}\rho_1^{i+a'}\rho_2^b}{\cancel{\alpha}\rho_1^i\cancel{\rho_2^0}} r = \rho_2\mu_1(1-p)$$

$$\frac{\rho_1^{\cancel{i}+a'}\rho_2^b}{\cancel{\rho_1^i}} r = \rho_2\mu_1(1-p)$$

$\rho_1^{a'}\rho_2^b r = \rho_2\mu_1(1-p)$

Now I choose $r = \mu_1(1-p)$ such that:

$\rho_1^{a'}\rho_2^b \cancel{\mu_1(1-p)} = \rho_2\cancel{\mu_1(1-p)}$

$\rho_1^{a'}\rho_2^b = \rho_2$

Finally, choosing $a' = 0$ and $b = 1$, I subsequently have:

$\cancel{\rho_1^0}\rho_2^1 = \rho_2$

$\rho_2 = \rho_2$

The last thing to compute is ρ_1 and I can compute it from the equation of state P_n (considering that $\rho_2 = \dfrac{\lambda_1}{\mu_1}$):

$$x_e + \lambda_1 + x_f + \mu_1 = \rho_2\mu_1 + \rho_1\rho_2\mu_1(1-p) + \rho_1 x_e + \frac{1}{\rho_2}\lambda_1 + \frac{\Pi_{F_n}}{\Pi_{0n}}r'$$

$$x_e + \lambda_1 + x_f + \mu_1 = \frac{\lambda_1}{\cancel{\mu_1}}\cancel{\mu_1} + \rho_1\frac{\lambda_1}{\cancel{\mu_1}}\cancel{\mu_1}(1-p) + \rho_1 x_e + \frac{1}{\frac{\lambda_1}{\mu_1}}\cancel{\lambda_1} + \frac{\cancel{\alpha}\rho_1^{-1}\cancel{\rho_2^n}}{\cancel{\alpha}\rho_1^0\cancel{\rho_2^n}}x_f$$

$$x_e + \cancel{\lambda_1} + x_f + \cancel{\mu_1} = \cancel{\lambda_1} + \rho_1\lambda_1(1-p) + \rho_1 x_e + \cancel{\mu_1} + \frac{1}{\rho_1}x_f$$

$$x_e + x_f = \rho_1\lambda_1(1-p) + \rho_1 x_e + \frac{1}{\rho_1}x_f$$

$$\rho_1(x_e + x_f) = \frac{\rho_1^2\lambda_1(1-p) + \rho_1^2 x_e + x_f}{\cancel{\rho_1}}\cancel{\rho_1}$$

$$\rho_1^2 x_e + \rho_1^2(\lambda_1(1-p)) + x_f = \rho_1(x_e + x_f)$$

$$\rho_1^2(x_e + \lambda_1 - \lambda_1 p) - \rho_1(x_e + x_f) + x_f = 0$$

$$\frac{\rho_1^2\cancel{(x_e + \lambda_1 - \lambda_1 p)}}{\cancel{x_e + \lambda_1 - \lambda_1 p}} - \frac{\rho_1(x_e + x_f)}{x_e + \lambda_1 - \lambda_1 p} + \frac{x_f}{x_e + \lambda_1 - \lambda_1 p} = 0$$

$$\rho_1^2 - \frac{\rho_1(x_e + x_f)}{x_e + \lambda_1 - \lambda_1 p} = -\frac{x_f}{x_e + \lambda_1 - \lambda_1 p}$$

$$\rho_1^2 - \frac{\rho_1(x_e + x_f)}{x_e + \lambda_1 - \lambda_1 p} + \frac{(x_e + x_f)^2}{4(x_e + \lambda_1 - \lambda_1 p)^2} = -\frac{x_f}{x_e + \lambda_1 - \lambda_1 p} + \frac{(x_e + x_f)^2}{4(x_e + \lambda_1 - \lambda_1 p)^2}$$

$$\left(\rho_1 - \frac{x_e + x_f}{2(x_e + \lambda_1 - \lambda_1 p)}\right)^2 = \frac{(x_e + x_f)^2}{4(x_e + \lambda_1 - \lambda_1 p)^2} - \frac{x_f}{x_e + \lambda_1 - \lambda_1 p}$$

$$\left(\rho_1 - \frac{x_e + x_f}{2(x_e + \lambda_1 - \lambda_1 p)}\right)^2 = \frac{(x_e + x_f)^2 - 4x_f(x_e + \lambda_1 - \lambda_1 p)}{4(x_e + \lambda_1 - \lambda_1 p)^2}$$

$$\left(\rho_1 - \frac{x_e + x_f}{2(x_e + \lambda_1 - \lambda_1 p)}\right)^2 = \frac{x_e^2 + x_f^2 + 2x_e x_f - 4x_e x_f - 4x_f \lambda_1 + 4x_f \lambda_1 p}{4(x_e + \lambda_1 - \lambda_1 p)^2}$$

Now I have to add the following bound if I want to proceed:

$$x_e^2 + x_f^2 - 2x_e x_f + 4x_f \lambda_1 (p-1) >= 0$$

Supposing it is greater or equal than 0, then I can proceed with the computation:

$$\rho_1 - \frac{x_e + x_f}{2(x_e + \lambda_1 - \lambda_1 p)} = \pm\sqrt{\frac{x_e^2 + x_f^2 - 2x_e x_f + 4x_f \lambda_1 (p-1)}{4(x_e + \lambda_1 - \lambda_1 p)^2}}$$

$$\rho_1 = \frac{x_e + x_f}{2(x_e + \lambda_1 - \lambda_1 p)} \pm \sqrt{\frac{x_e^2 + x_f^2 - 2x_e x_f + 4x_f \lambda_1 (p-1)}{4(x_e + \lambda_1 - \lambda_1 p)^2}}$$

The valid case is the one with $-$, so in conclusion, I have that:

$$\rho_1 = \frac{x_e + x_f}{2(x_e + \lambda_1 - \lambda_1 p)} - \sqrt{\frac{x_e^2 + x_f^2 - 2x_e x_f + 4x_f \lambda_1 (p-1)}{4(x_e + \lambda_1 - \lambda_1 p)^2}}$$

$$\rho_2 = \frac{\lambda_1}{\mu_1}$$

But I have to add another bound to ensure that ρ_1 is greater than 0.
In conclusion this solution is not acceptable due to these two bounds. The only permitted and fixed bound is that *all rates must be greater than* 0.

6.2.2.3 Alternative

Since that $\frac{1}{\rho_1}x_f$ brings complexity to the global balance equations system, an alternative to the previous computation can be found with the following addition:
I will add an action of type τ from all phantom states F, each of them corresponding to a normal states, to all other normal states. So I add an ingoing action (τ, r'') to all the normal states. In this way I add an action rate on the equation and this subsequently add in it the left part:

$\frac{\Pi_{F_{ij}}}{\Pi_{ij}}r''$ with $i \geq 0$ and $j \geq 0$

And this can be reduced in this way:

$$\frac{\Pi_{F_{ij}}}{\Pi_{ij}}r'' = \frac{\alpha\rho_1^{a''}\rho_2^{b''}}{\alpha\rho_1^i\rho_2^j}r'$$

Then I choose the values for $c = i - 1$, $d = j$, $r'' = (n\rho_1 - 1)x_f$ and $n > \frac{1}{\rho_1}$ (to ensure that the rate $r'' > 0$), such that:

$$\Pi_{F_{ij}} = \alpha\rho_1^{i-1}\rho_2^j$$

From this I get:

$$\frac{\Pi_{F_{ij}}}{\Pi_{ij}}r'' = \frac{\alpha\rho_1^{i-1}\rho_2^m}{\alpha\rho_1^i\rho_2^m}x_f = \frac{1}{\rho_1}x_f$$

Then I won't alter the balance of previous equations because this rate is added in the left part of all of them. Then we have the following computation of ρ_1 from state P_n:

$$x_e + \lambda_1 + x_f + \mu_1 = \rho_2\mu_1 + \rho_1\rho_2\mu_1(1-p) + \rho_1 x_e + \frac{1}{\rho_2}\lambda_1 + \frac{\Pi_{F_n}}{\Pi_{0n}}r' + \frac{\Pi_{F_n}}{\Pi_{0n}}r''$$

$$x_e + \lambda_1 + x_f + \mu_1 = \frac{\lambda_1}{\cancel{\mu_1}}\cancel{\mu_1} + \rho_1\frac{\lambda_1}{\cancel{\mu_1}}\cancel{\mu_1}(1-p) + \rho_1 x_e + \frac{\mu_1}{\cancel{\lambda_1}}\cancel{\lambda_1} + \frac{1}{\rho_1}x_f + \frac{1}{\rho_1}(n\rho_1 - 1)x_f$$

$$x_e + \cancel{\lambda_1} + x_f + \cancel{\mu_1} = \cancel{\lambda_1} + \rho_1\lambda_1(1-p) + \rho_1 x_e + \cancel{\mu_1} + \frac{\cancel{1} + n\rho_1 - \cancel{1}}{\rho_1}x_f$$

$$x_e + x_f = \rho_1\lambda_1(1-p) + \rho_1 x_e + \frac{n\cancel{\rho_1}}{\cancel{\rho_1}}x_f$$

$$x_e + x_f(1-n) = \rho_1(\lambda_1(1-p) + x_e)$$

$$\rho_1\frac{\cancel{\lambda_1(1-p)+x_e}}{\cancel{\lambda_1(1-p)+x_e}} = \frac{x_e + x_f(1-n)}{\lambda_1(1-p)+x_e}$$

So, in this case I have that:

$$\rho_1 = \frac{x_e + x_f(1-n)}{\lambda_1(1-p)+x_e}$$

But I have also here to ensure that $\rho_1 > 0$ and to do that I have to put conditions on x_e and x_f. In conclusion also this solution is not acceptable due to the extra bound I have to put on rates.

6.2.3 Second Trial - Balancing of Equations of Process R_1 (DI) using Pre-computation without states F

The second methodology I have tried use the concept of pre-computation. This will find interesting values for ρ_1 and ρ_2. These values will be very similar to those found in the previous literature in the original G-Network.
I will first computing the values of ρ_1 and ρ_2 in the process R_1 without any phantom states F. After that I will add the impossible actions and phantom states and try to balance the global balance equations and calculate the various $\Pi_{F_{ij}}$ (with $i \geq 0$ and $j \geq 0$).
I have the following starting equations (without states F):

$$\begin{cases}
\lambda_1 + x_f = \rho_2\mu_1 + \rho_1\rho_2\mu_1(1-p) + \rho_1 x_e & \text{state } 0 \\
\lambda_1 + x_f + \mu_1 = \rho_2\mu_1 + \rho_1\rho_2\mu_1(1-p) + \rho_1 x_e + \frac{1}{\rho_2}\lambda_1 & \text{state } n \\
x_e + \lambda_1 + x_f = \rho_2\mu_1 p + \rho_1\rho_2\mu_1(1-p) + \rho_1 x_e + \frac{1}{\rho_1}x_f & \text{state } 0^i \\
x_e + \lambda_1 + x_f + \mu_1 = \rho_2\mu_1 p + \rho_1\rho_2\mu_1(1-p) + \rho_1 x_e + \frac{1}{\rho_2}\lambda_1 + \frac{1}{\rho_1}x_f & \text{state } n^i
\end{cases}$$

From this we can compute ρ_2 comparing state 0 and state n (or 0^i and n^i is the same):

$$\begin{array}{lcccccccc}
\lambda_1 + x_f + \mu_1 & = & \rho_2\mu_1 & + & \rho_1\rho_2\mu_1(1-p) & + & \rho_1 x_e & + & \frac{1}{\rho_2}\lambda_1 \\
\lambda_1 + x_f & = & \rho_2\mu_1 & + & \rho_1\rho_2\mu_1(1-p) & + & \rho_1 x_e & & \\
\hline
\mu_1 & = & & & & & & & \frac{1}{\rho_2}\lambda_1
\end{array}$$

And finally get:

$$\mu_1 = \frac{1}{\rho_2}\rho_2\lambda_1$$
$$\rho_2\mu_1 = \frac{1}{\cancel{\rho_2}}\cancel{\rho_2}\lambda_1$$
$$\rho_2\frac{\cancel{\mu_1}}{\cancel{\mu_1}} = \frac{\lambda_1}{\mu_1}$$
$$\rho_2 = \frac{\lambda_1}{\mu_1}$$

Now to find ρ_1, I compare state 0 and 0^i (or n and n^i is the same):

$$\begin{array}{lcccccccc}
x_e + \lambda_1 + x_f & = & \rho_2\mu_1 p & + & \rho_1\rho_2\mu_1(1-p) & + & \rho_1 x_e & + & \frac{1}{\rho_1}x_f \\
\lambda_1 + x_f & = & \rho_2\mu_1 & + & \rho_1\rho_2\mu_1(1-p) & + & \rho_1 x_e & & \\
\hline
x_e & = & -\rho_2\mu_1(1-p) & + & & & & & \frac{1}{\rho_1}x_f
\end{array}$$

And then finally get also $\rho 1$:

$$x_e = -\rho_2\mu_1(1-p) + \frac{1}{\rho_1}x_f$$
$$x_e + \rho_2\mu_1(1-p) = \frac{1}{\rho_1}x_f$$
$$\rho_1(x_e + \frac{\lambda_1}{\cancel{\mu_1}}\cancel{\mu_1}(1-p)) = \frac{1}{\cancel{\rho_1}}x_f\cancel{\rho_1}$$
$$\rho_1\frac{\cancel{x_e + \lambda_1(1-p)}}{\cancel{x_e + \lambda_1(1-p)}} = \frac{x_f}{x_e + \lambda_1(1-p)}$$

$$\rho_1 = \frac{x_f}{x_e + \lambda_1(1-p)}$$

In this case I don't have to add any bound to ensure the positivity of ρ_1 because x_f, x_e and λ_1 are all greater than 0 because they are all rates and $(1-p) \geq 0$ because it is a probability from 0 to 1 so $0 \leq p \leq 1$. From this, ρ_1 will be always greater than 0.

Now I have to add also the impossible actions, but with them also the equations will change. For keeping them balanced I have to fractionate one of them and since among $\rho 1$ and ρ_2, ρ_1 is more complicated to compute, I will add a constant in ρ_1 instead of ρ_2 to have more working space. This constant δ will be greater than 0.
To summarize, I have the following values for ρ:

$$\rho_1 = \delta\frac{x_f}{x_e + \lambda_1(1-p)}$$
$$\rho_2 = \frac{\lambda_1}{\mu_1}$$

Now I will substitute them in the global balance equations of R_1 (without impossible actions and states F).
I add also a $\bigoplus$, in the end of balancing computation of each state, it represents one or more impossible actions, where they have to be inserted:

State P_0

$$\lambda_1 + x_f = \rho_2\mu_1 + \rho_1\rho_2\mu_1(1-p) + \rho_1 x_e$$
$$\lambda_1 + x_f = \frac{\lambda_1}{\cancel{\mu_1}}\cancel{\mu_1} + \delta\frac{x_f}{x_e + \lambda_1(1-p)}\frac{\lambda_1}{\cancel{\mu_1}}\cancel{\mu_1}(1-p) + \delta\frac{x_f}{x_e + \lambda_1(1-p)}x_e$$
$$\cancel{\lambda_1} + x_f = \cancel{\lambda_1} + \delta\frac{x_f}{x_e + \lambda_1(1-p)}\lambda_1(1-p) + \delta\frac{x_f}{x_e + \lambda_1(1-p)}x_e$$
$$x_f = \delta\frac{x_f}{x_e + \lambda_1(1-p)}(\lambda_1(1-p) + x_e)$$
$$x_f = \delta\frac{x_f}{\cancel{x_e + \lambda_1(1-p)}}\cancel{(x_e + \lambda_1(1-p))}$$
$$x_f = \delta x_f$$

Finally, I add the $\bigoplus$ in the left part of the equation:

$\bigoplus + x_f = \delta x_f$

(With basic ρ_1 (i.e. without $\bigoplus$) this would be $x_f = x_f$.)

State P_n

$$\lambda_1 + x_f + \mu_1 = \rho_2\mu_1 + \rho_1\rho_2\mu_1(1-p) + \rho_1 x_e + \frac{1}{\rho_2}\lambda_1$$
$$\lambda_1 + x_f + \mu_1 = \frac{\lambda_1}{\cancel{\mu_1}}\cancel{\mu_1} + \delta\frac{x_f}{x_e + \lambda_1(1-p)}\frac{\lambda_1}{\cancel{\mu_1}}\cancel{\mu_1}(1-p) + \delta\frac{x_f}{x_e + \lambda_1(1-p)}x_e + \frac{1}{\frac{\lambda_1}{\mu_1}}\lambda_1$$
$$\cancel{\lambda_1} + x_f + \mu_1 = \cancel{\lambda_1} + \delta\frac{x_f}{x_e + \lambda_1(1-p)}\lambda_1(1-p) + \delta\frac{x_f}{x_e + \lambda_1(1-p)}x_e + \frac{\mu_1}{\cancel{\lambda_1}}\cancel{\lambda_1}$$
$$x_f + \cancel{\mu_1} = \delta\frac{x_f}{x_e + \lambda_1(1-p)}(\lambda_1(1-p) + x_e) + \cancel{\mu_1}$$
$$x_f = \delta\frac{x_f}{\cancel{x_e + \lambda_1(1-p)}}\cancel{(x_e + \lambda_1(1-p))}$$
$$x_f = \delta x_f$$

Finally, I add the $\bigoplus$ in the left part of the equation:

$\bigoplus + x_f = \delta x_f$

(Also here with basic ρ_1 (i.e. without $\bigoplus$) this would be $x_f = x_f$.)

State P_0^i

$$x_e + \lambda_1 + x_f = \rho_2\mu_1 p + \rho_1\rho_2\mu_1(1-p) + \rho_1 x_e + \frac{1}{\rho_1}x_f$$
$$x_e + \lambda_1 + x_f = \frac{\lambda_1}{\cancel{\mu_1}}\cancel{\mu_1}p + \delta\frac{x_f}{x_e + \lambda_1(1-p)}\frac{\lambda_1}{\cancel{\mu_1}}\cancel{\mu_1}(1-p) + \delta\frac{x_f}{x_e + \lambda_1(1-p)}x_e + \frac{1}{\delta\frac{x_f}{x_e + \lambda_1(1-p)}}x_f$$
$$x_e + \lambda_1 + x_f = \lambda_1 p + \delta\frac{x_f}{x_e + \lambda_1(1-p)}\lambda_1(1-p) + \delta\frac{x_f}{x_e + \lambda_1(1-p)}x_e + \frac{x_e + \lambda_1(1-p)}{\delta\cancel{x_f}}\cancel{x_f}$$
$$x_e + \lambda_1 + x_f = \lambda_1 p + \delta\frac{x_f}{x_e + \lambda_1(1-p)}(\lambda_1(1-p) + x_e) + \frac{x_e + \lambda_1(1-p)}{\delta}$$
$$x_e + \lambda_1 + x_f = \lambda_1 p + \delta\frac{x_f}{\cancel{x_e + \lambda_1(1-p)}}\cancel{(x_e + \lambda_1(1-p))} + \frac{x_e}{\delta} + \frac{\lambda_1(1-p)}{\delta}$$

$$x_e + \lambda_1 + x_f = \lambda_1 p + \delta x_f + \frac{x_e}{\delta} + \frac{\lambda_1(1-p)}{\delta}$$

Finally, I add the $\bigoplus$ in the right part of the equation:

$$x_e + \lambda_1 + x_f = \lambda_1 p + \delta x_f + \frac{x_e}{\delta} + \frac{\lambda_1(1-p)}{\delta} + \bigoplus$$

$$\left(\begin{array}{l} \text{Here with basic } \rho_1 \text{ (i.e. without } \bigoplus\text{) this would be:} \\ x_e + \lambda_1 + x_f = \lambda_1 p + x_f + x_e + \lambda_1(1-p) \\ x_e + \lambda_1 + x_f = x_e + \lambda_1(\cancel{p} + 1 - \cancel{p}) + x_f \\ x_e + \lambda_1 + x_f = x_e + \lambda_1 + x_f \end{array} \right)$$

State P_n^i

$$x_e + \lambda_1 + x_f + \mu_1 = \rho_2\mu_1 p + \rho_1\rho_2\mu_1(1-p) + \rho_1 x_e + \frac{1}{\rho_2}\lambda_1 + \frac{1}{\rho_1}x_f$$

$$x_e + \lambda_1 + x_f + \mu_1 = \frac{\lambda_1}{\cancel{\mu_1}}\cancel{\mu_1} p + \delta\frac{x_f}{x_e + \lambda_1(1-p)}\frac{\lambda_1}{\cancel{\mu_1}}\cancel{\mu_1}(1-p) + \delta\frac{x_f}{x_e + \lambda_1(1-p)}x_e + \frac{1}{\frac{\lambda_1}{\mu_1}}\lambda_1 + \frac{1}{\delta\frac{x_f}{x_e+\lambda_1(1-p)}}x_f$$

$$x_e + \lambda_1 + x_f + \mu_1 = \lambda_1 p + \delta\frac{x_f}{x_e + \lambda_1(1-p)}\lambda_1(1-p) + \delta\frac{x_f}{x_e + \lambda_1(1-p)}x_e + \frac{\mu_1}{\cancel{\lambda_1}}\cancel{\lambda_1} + \frac{x_e + \lambda_1(1-p)}{\delta\cancel{x_f}}\cancel{x_f}$$

$$x_e + \lambda_1 + x_f + \cancel{\mu_1} = \lambda_1 p + \delta\frac{x_f}{x_e + \lambda_1(1-p)}(\lambda_1(1-p) + x_e) + \cancel{\mu_1} + \frac{x_e + \lambda_1(1-p)}{\delta}$$

$$x_e + \lambda_1 + x_f = \lambda_1 p + \delta\frac{x_f}{\cancel{x_e + \lambda_1(1-p)}}\cancel{(x_e + \lambda_1(1-p))} + \frac{x_e}{\delta} + \frac{\lambda_1(1-p)}{\delta}$$

$$x_e + \lambda_1 + x_f = \lambda_1 p + \delta x_f + \frac{x_e}{\delta} + \frac{\lambda_1(1-p)}{\delta}$$

Finally, I add the $\bigoplus$ in the right part of the equation:

$$x_e + \lambda_1 + x_f = \lambda_1 p + \delta x_f + \frac{x_e}{\delta} + \frac{\lambda_1(1-p)}{\delta} + \bigoplus$$

$$\left(\begin{array}{l} \text{Also here with basic } \rho_1 \text{ (i.e. without } \bigoplus\text{) this would be:} \\ x_e + \lambda_1 + x_f = \lambda_1 p + x_f + x_e + \lambda_1(1-p) \\ x_e + \lambda_1 + x_f = x_e + \lambda_1(\not{p} + 1 - \not{p}) + x_f \\ x_e + \lambda_1 + x_f = x_e + \lambda_1 + x_f \end{array}\right)$$

Now, the balance equations to solve are the followings (P_0 is the same to P_n and also P_0^i and P_n^i):

$$\begin{cases} \bigoplus + x_f = \delta x_f \\ x_e + \lambda_1 + x_f = \lambda_1 p + \delta x_f + \dfrac{x_e}{\delta} + \dfrac{\lambda_1(1-p)}{\delta} + \bigoplus \end{cases}$$

In particular the problem is in the second equation because, substituting $\bigoplus$ with general r rate greater than 0, then we have the following cases:

- if $\delta > 1$ then $\delta x_f > x_f$ and we have to add $r < 0$ to balance the equation;
- if $\delta < 1$ then $\dfrac{x_e}{\delta} > x_e$ and we have to add $r < 0$ to balance the equation;
- if $\delta = 1$ then $x_e + \lambda_1 + x_f = \lambda_1 p + x_f + x_e + \lambda(1-p)$ but we have to add $r = 0$ to maintain the balance.

In all cases $r \leq 0$ but the main condition of rates is that all of them must be greater than 0, so this solution is not acceptable.
To solve those problem I will explain two methods for the balance of the equations in the following sections.

6.3 Generate Safe-Impossible actions in Input and Output

Sometime, to balance the equations, we need outgoing actions to add rate in the left part of equation or ingoing actions to add rate in the right part.
In the following example, I will show how to add actions toward and from a "phantom" state F, in such a way that:

- F remains an unreachable state
- The system will behave in the same way

- Added "impossible" Passive actions are outgoing in all the states
- Added "impossible" Active actions are ingoing in all the states

Let's consider the following processes:

$$P = \begin{cases} P_0 = (a, \lambda).P_1 + (b, \top).P_1 + (a, \lambda).P_0 \\ P_1 = (b, \top).P_0 \end{cases}$$

$$Q = \begin{cases} Q_0 = (a, \top).Q_1 + (b, \mu).Q1 \\ Q_1 = (b, \mu).Q_0 + (a, \top).Q1 \end{cases}$$

$$S \stackrel{def}{=} P \underset{a,b}{\bowtie} Q$$

And the starting states are P_0 and Q_0:

As shown in their derivative graphs of figure 6.23, each state has at least one outgoing passive action and one ingoing active action of the types in the cooperation set.

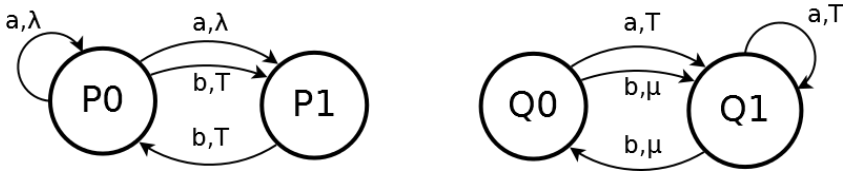

Figure 6.23: Example of two Cooperating Processes P and Q

As already explained before, it is the same to add only one state F or one for each normal state in the process, in this example I will consider only one phantom state F in each process (or also one for each subset of states of process).
Now I add the phantom state F and link it with P (called F_P) and Q (called F_Q) through action c_1. $(c_1, \top)$ is added to P_0 and (c_1, r) is added to F_Q. I must put c_1 in the cooperation set, otherwise the process could go on state F and it

would be unreachable no more.
Remember that passive actions must be outgoing (in this case in P) and active ones must be ingoing (in this case in Q). F states are still unreachable because P_0 must cooperate with Q to go in F_P but from either Q_0 and Q_1 there is no action type c_1 outgoing but only ingoing. P should cooperate with F_Q to go in F_P but the latter is unreachable, so there is no way to go from P_0 or Q_0 to F_P and F_Q respectively.

The system now is:

$$S' = P \underset{a,b,c_1}{\bowtie} Q$$

And the derivative graph is the following:

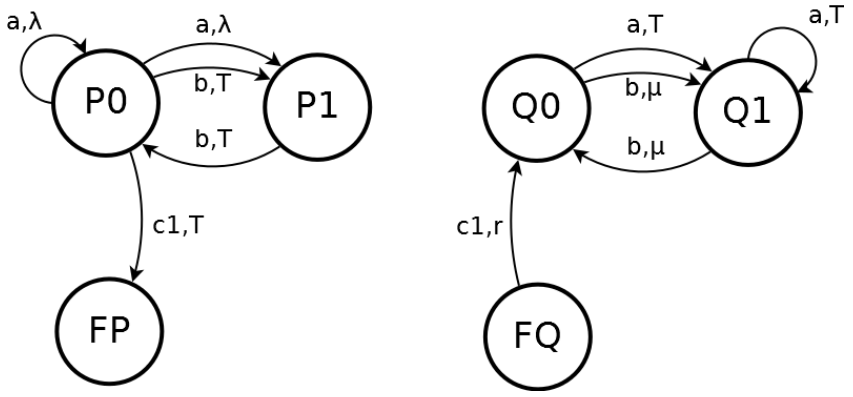

Figure 6.24: Example of two Cooperating Processes P and Q with impossible action c_1

Now the rate x_{c_1} is added on the left of the global balance equation of P_0, and similarly the rate r is added on the right of the global balance equation of Q_0. As I did before, now I will add action c_2, specularly as I did for c_1. In this way, (c_2, r') is added to F_P and $(c_2, \top)$ is added to Q_0. F is still unreachable because P and Q cannot synchronize neither on c_1, nor on c_2, like in the previous passage.

The final system is:

$$S'' = P \underset{a,b,c_1,c_2}{\bowtie} Q$$

And the derivative graph is the following:

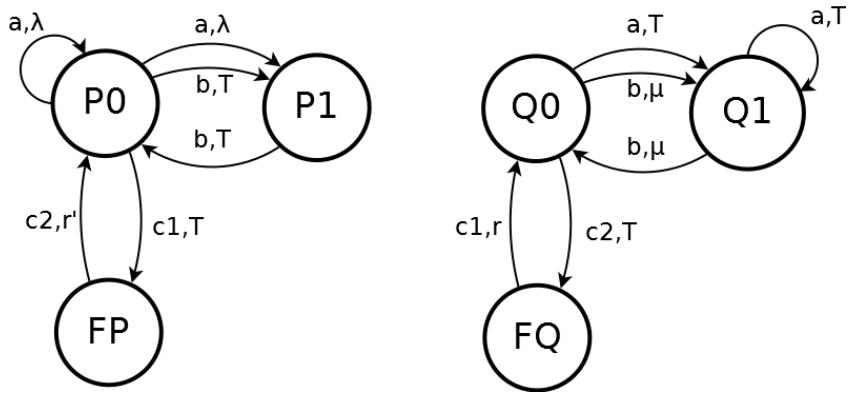

Figure 6.25: Example of two Cooperating Processes P and Q with impossible actions c_1, c_2

The rate r' is added on the right of the global balance equation of P_0 and the rate x_{c_2} is added on the left of the global balance equation of Q_0.
As the final step, I add c_1 and c_2 to all states in the system, taking care that on process P, c_1 actions are always outgoing and c_2 ingoing and vice versa on Q, c_1 actions are ingoing and c_2 outgoing. In this way F_P and F_Q remain unreachable and since all new actions involve these F states, the new system S'' behaves in the same manner as S. Moreover, in all the states, all impossible passive actions are outgoing and all impossible active actions are ingoing.

And the derivative graph of this system is the following:
There is only one exception in the states F_P and F_Q. These two states don't have outgoing passive impossible actions and ingoing active impossible actions, but it is easy to solve, I just add one self-loop for every missing outgoing and ingoing action. Since they are unreachable, these actions are still impossible and all of this doesn't affect the behavior of the entire system, however

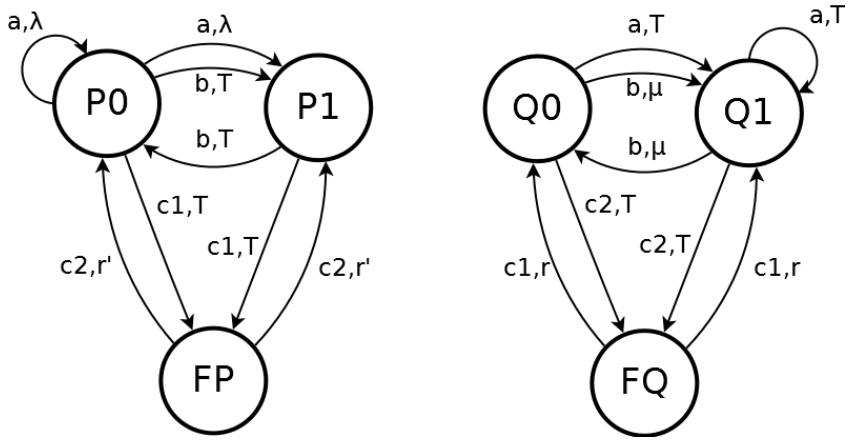

Figure 6.26: Example of Final System with impossible actions c_1, c_2

it satisfy the two conditions of the theorem (regarding impossible actions).

Finally, the derivative graph of the final and complete system is the following:

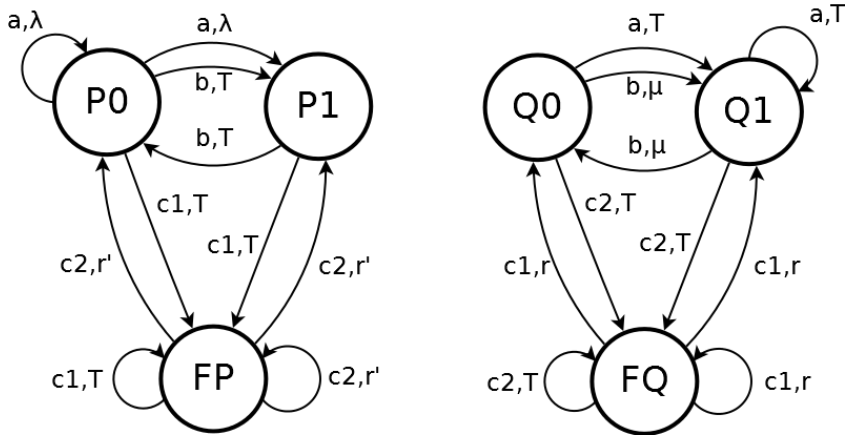

Figure 6.27: Example of Final System with impossible actions c_1, c_2 also in F

The final processes are now:

$$P = \begin{cases} P_0 = (a,\lambda).P_1 + (b,\top).P_1 + (a,\lambda).P_0 + (c_1,\top).F \\ P_1 = (b,\top).P_0 + (c_1,\top).F \\ F_P = (c_1,\top).F_P + (c_2,r').F_P + (c_2,r').P_0 + (c_2,r').P_1 \end{cases}$$

$$Q = \begin{cases} Q_0 = (a,\top).Q_1 + (b,\mu).Q1 + (c_2,\top).F \\ P_1 = (b,\mu).Q_0 + (a,\top).Q1 + (c_2,\top).F \\ F_Q = (c_2,\top).F_P + (c_1,r).F_P + (c_1,r).Q_0 + (c_1,r).Q_1 \end{cases}$$

$$S'' \stackrel{def}{=} P \underset{a,b,c_1,c_2}{\bowtie} Q$$

And the starting states are still P_0 and Q_0:

In conclusion, in any system, I can add rates on the global balance equations, both left and right, without changing the behavior of the initial system. With this method, I can help the balancing especially when the only other solution is to add a negative rate (that is not permitted).

6.4 Balance of Equations between Normal and Phantom States F

If we want to apply the product form theorem of Harrison in a process, all passive actions in the cooperation set must be outgoing in every state and all active actions must be ingoing. In order to ensure these properties, sometimes we have to add impossible actions from and toward a phantom state F. We have seen that we can always add safe-impossible actions in the system, but to balance them there are to main cases:

- Actions added are active from phantom state F to all other normal states
- Actions added are passive from all normal states to phantom state F

I will use a process with only two states in the following examples, but it would be easy to apply to more complex ones. A usual, instead of having

only one phantom state F, I suppose I can have a lattice of phantom states F, one for each normal state.

6.4.1 Balance Passive Actions toward phantom state F

I analyze now the balancing of the first case in which there are passive outgoing actions toward a phantom state F.

Considering the following process:

$$P = \begin{cases} P_0 = (empty, \top).F_0 \\ F_0 = (empty, \top).F_0 \end{cases}$$

With the starting state P_0 and another cooperating process Q. P and Q cooperate with the action $empty$ and $(empty, r)$ is active in Q.
The action $(empty, \top)$ is passive in P and in the following derivation graph the $\top$ rate is substituted with rate x_e as usually happen in the global balance equations where the $\top$ rates are substituted by variables, one for each kind of actions.

The derivation graph of the process P is:

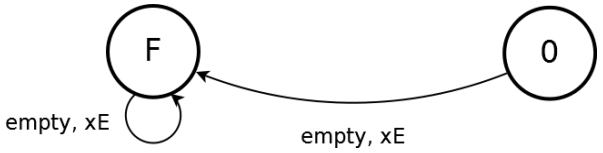

Figure 6.28: Process P_0 with added outgoing passive safe-impossible action $empty$

Usually in the global balance equations, it is safe to assume that $\Pi_{F_0} = \Pi_0$, then in this case, both equations are unbalanced:

State F_0 : $\Pi_{F_0}(x_e) = \Pi_0 x_e + \Pi_{F_0} x_e$

$\cancel{\Pi_{F_0}(x_e)} = \Pi_0 x_e + \cancel{\Pi_{F_0} x_e}$

$0 = \Pi_0 x_e$

State P_0 : $\Pi_0(x_e) = 0$

Remember that both Π_0 and x_e have to be greater than 0.
To solve this problem, I can simply add a $empty$ action from state F_0 to P_0 to balance the equations of both F_0 and P_0 without the need of further changes.

The process is now changed as follow:

$$P = \begin{cases} P_0 = (empty, \top).F_0 \\ F_0 = (empty, \top).F_0 + (empty, \top).P_0 \end{cases}$$

And the derivation graph is now the following:

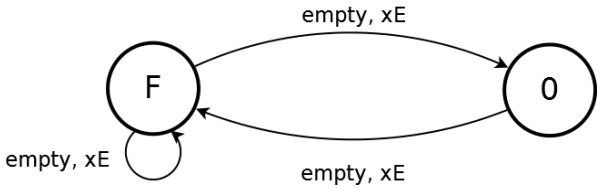

Figure 6.29: Balanced Process P_0 with added outgoing passive safe-impossible action $empty$

The process P_0 is now balanced (remember that $\Pi_{F_0} = \Pi_0$):

State F_0 : $\Pi_{F_0}(x_e + x_e) = \Pi_0 x_e + \Pi_{F_0} x_e$

$$\frac{\Pi_{F_0}}{\Pi_{F_0}}(x_e + x_e) = \frac{\Pi_0}{\Pi_{F_0}} x_e + \frac{\Pi_{F_0}}{\Pi_{F_0}} x_e$$

$$x_e + x_e = x_e + x_e$$

State P_0 : $\Pi_0(x_e) = \Pi_F x_e$

$$\frac{\Pi_0}{\Pi_0}(x_e) = \frac{\Pi_{F_0}}{\Pi_0} x_e$$

$$x_e = x_e$$

6.4.2 Active actions to F to balance

I analyze now the balancing of the second case in which there are active ingoing actions from a phantom state F.
To balance active added actions is more difficult because we have to add an action from state P_0 to a phantom state F which must be in the cooperation set and doesn't synchronize in any way with other processes of the system. In this way the process cannot proceed in F state because of the missing synchronization. Moreover, we cannot usually add an action with type from the preexisting cooperation set because in most cases the majority of states have already all action types already used. Therefore, I just need to add a new action type also to avoid problem in synchronization with preexisting actions. In the following example I will show this problem and how to resolve it, step by step.

Considering the following process P which has an active cooperating action to balance:

$$P = \Big\{ F_P = (fill, \lambda).F_P + (fill, \lambda).P_0$$

With the starting state P_0 and another cooperating process Q. P and Q cooperate with the action $fill$ and $(fill, \top)$ is passive in Q.

The derivation graph of the process P is:

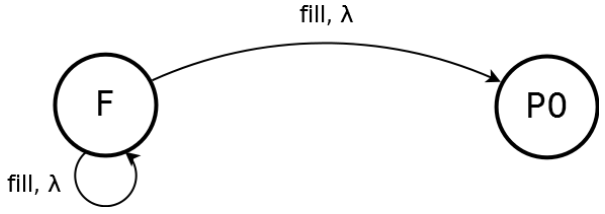

Figure 6.30: Process P with added ingoing active safe-impossible action $fill$

Usually in the global balance equations, it is safe to assume that $\Pi_{F_0} = \Pi_0$, then in this case, both equations are unbalanced:

State F_P : $\Pi_{F_P}(\lambda + \lambda) = \Pi_{F_P}\lambda$

$\cancel{\Pi_{F_P}}(\lambda + \lambda) = \cancel{\Pi_{F_P}}\lambda$

$\lambda = 0$

State P_0 : $0 = \Pi_{F_P}(\lambda)$

Remember that both Π_{F_0} and λ have to be greater than 0.
To solve this problem, first of all I add the new action type imp_{fill}, to make the system balanced. This action is added from state P_0 to the phantom state F and it is put in the cooperation set because it must be impossible or the phantom state F will be no more unreachable. Due to its outgoing direction from P_0 it must also be passive to satisfy the conditions of the product form theorem. Moreover, also in F_0 is add the action $(imp_{fill}, \top)$ to satisfy the conditions but in this case as a self-loop, so it is impossible and doesn't alter the system behavior.
The action added $(imp_{fill}, \top)$ is passive in P and in the following derivation graph the $\top$ rate is substituted with rate x_f as usually happen in the global balance equations where the $\top$ rates are substituted by variables, one for each kind of actions.

This is the derivation graph:

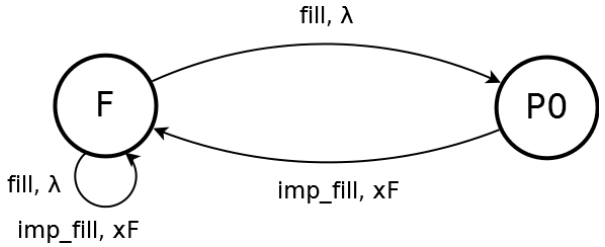

Figure 6.31: Process P with added impossible action imp_{fill} to balance the system

Unfortunately imp_{fill} is passive and there must exist its active counterpart somewhere, in another process. For this reason, I add it in Q which has already a cooperation with P, in the following way:

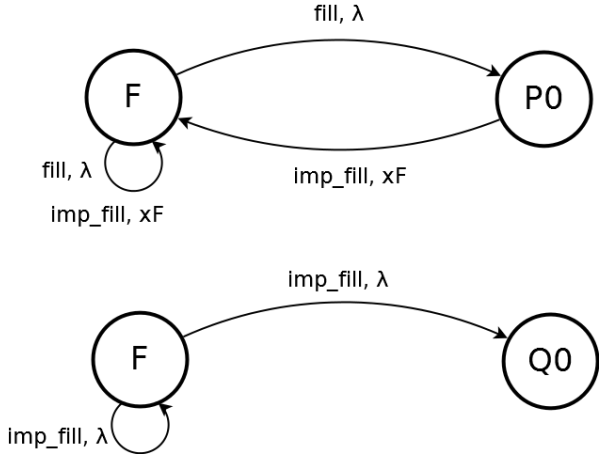

Figure 6.32: Processes P and Q with added impossible action imp_{fill} to balance the system

I don't care (and don't draw) for the other actions of Q and consider them balanced, if they weren't, I would simply apply the method of this section also in Q after its application on P.

Now process P is balanced, in fact x_f, as a passive rate, on cooperation will only take values of active action imp_{fill} in process Q and all rates of imp_{fill} in Q are λ.
Unfortunately this leads to unbalance the process Q (like it was at the beginning off this section). To balance it I cannot add $fill$ because it can be also used from Q_0 to another state Q_n and so with it I can go from Q_0 to F and this cannot be acceptable. I cannot add imp_{fill} either, because it would synchronize with imp_{fill} of P_0 and both P_0 and Q_0 would go in phantom states F.
So I add another new action end_{fill}. This action is added from state Q_0 to the phantom state F and it is put in the cooperation set because it must be impossible or the phantom state F will be no more unreachable. Due to its outgoing direction from Q_0 it must also be passive to satisfy the conditions of the product form theorem. Moreover, also in F_0 is add the action $(end_{fill}, \top)$ to satisfy the conditions but in this case as a self-loop, so it is impossible and doesn't alter the system behavior.
The action added $(end_{fill}, \top)$ is passive in P and also in the following derivation graph the $\top$ rate is substituted with rate x_{ef} as usually happen in the global balance equations where the $\top$ rates are substituted by variables, one for each kind of actions.

This is the derivation graph:

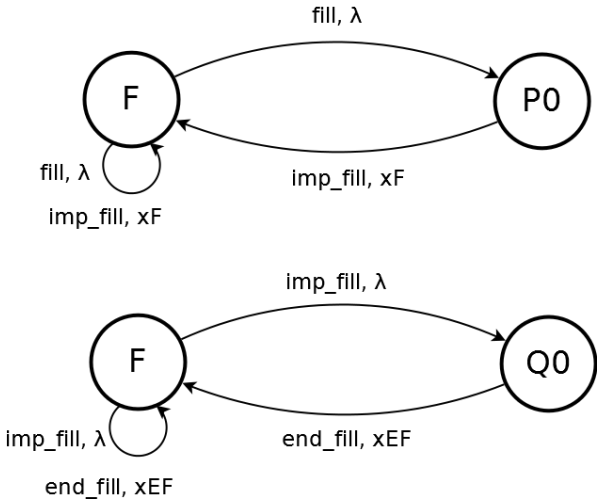

Figure 6.33: Processes P and Q with added impossible actions imp_{fill} and end_{fill} to balance the system

As usual end_{fill} is passive and there must exist its active counterpart somewhere, in another process.
If I use it in P, it would unbalanced the process P and to balance it I have to put a passive impossible action with an active counterpart and if I put that counterpart in Q, I would enter in a cyclic process of adding impossible actions without reaching the balance.
For this reason:
For balance an active impossible action I must have 3 processes in cooperation and only with a triangular addition of impossible action (both active and passive) I can finally balance all of the processes.
For this reason, I will create another process C called *Collapsed Process* to close the "chain" of balancing, and add the active action end_{imp} in it, in the following way:

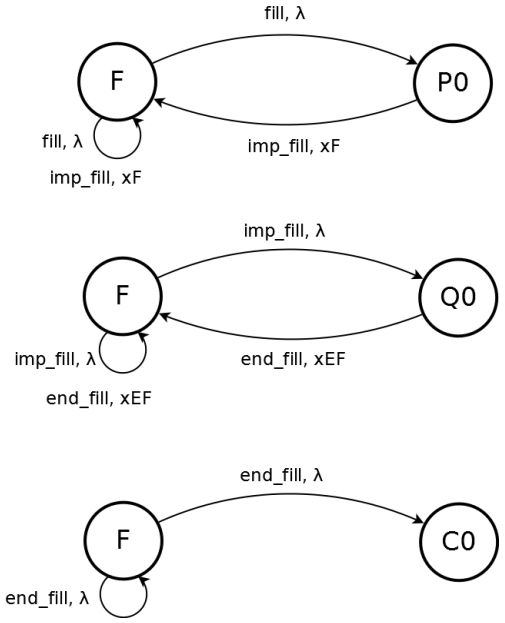

Figure 6.34: Processes P and Q and Collapsed Process C with actions imp_{fill} and end_{fill}

C has only the stating state C_0 and one phantom state F, moreover this process has a cooperation with P and Q in the actions imp_{fill} and end_{fill}.
Now process Q is balanced, in fact x_{ef}, as a passive rate, on cooperation will only take values of active action end_{fill} in process C and all rates of end_{fill} in C are λ.
Unfortunately this leads to unbalance the process C (like it was at the beginning off this section) but now I have the possibility to balance it without any other addition of actions. To balance it I cannot add $fill$ because it can synchronize with Q_0 or P_0 and so with it I can go from C_0 to F and this cannot be acceptable. Moreover, I cannot add end_{fill} or it would synchronize with Q_0 and both Q and C could go into their phantom states F.
However, I can add instead imp_{fill} as a passive action (and this satisfy also the conditions of theorem), because it is active only in process Q. In this way I have P, Q and C cannot cooperate in the action imp_fill because the starting states are P_0, Q_0 and C_0 but only in P_0 and C_0 the action imp_{fill} is outgoing

and in Q_0 it is only ingoing.

The final system is the following:

$$P = \begin{cases} P_0 = (imp_{fill}, \top).F_P \\ F_P = (imp_{fill}, \top).F_P + (fill, \lambda).F_P + (fill, \lambda).P_0 \end{cases}$$

$$Q = \begin{cases} Q_0 = (end_{fill}, \top).F_Q \\ F_Q = (end_{fill}, \top).F_Q + (imp_{fill}, \lambda).F_Q + (imp_{fill}, \lambda).Q_0 \end{cases}$$

$$C = \begin{cases} C_0 = (imp_{fill}, \top).F_C \\ F_C = (imp_{fill}, \top).F_C + (end_{fill}, \lambda).F_C + (end_{fill}, \lambda).C_0 \end{cases}$$

$$S \stackrel{def}{=} P \underset{imp_{fill}, end_{fill}}{\bowtie} Q \underset{imp_{fill}, end_{fill}}{\bowtie} C$$

The starting states are P_0, Q_0 and C_0 and the derivation graph is now the following:

The processes P, Q and C are now balanced (remember that $\Pi_F = \Pi_0$ and $x_f = x_{ef} = \lambda$):

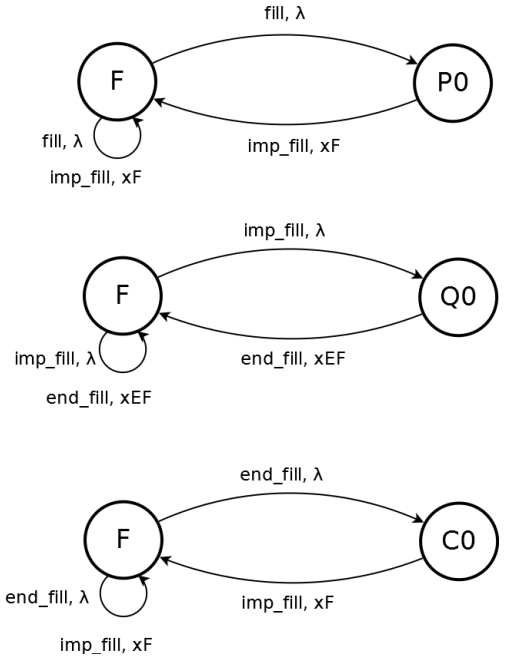

Figure 6.35: Final system of processes P, Q and collapsed process C, all balanced

Process P : State P_0 : $\Pi_0(x_f) = \Pi_{F_P}\lambda$

$$\frac{\Pi_0}{\Pi_0}(x_f) = \frac{\Pi_{F_P}}{\Pi_0}\lambda$$

$$x_f = \lambda$$

State F_P : $\Pi_{F_P}(x_f + \lambda + \lambda) = \Pi_0 x_f + \Pi_{F_P}(x_f + \lambda)$

$$\frac{\Pi_{F_P}}{\Pi_{F_P}}(x_f + \lambda + \lambda) = \frac{\Pi_0}{\Pi_{F_P}}x_f + \frac{\Pi_{F_P}}{\Pi_{F_P}}(x_f + \lambda)$$

$$x_f + \lambda + \lambda = x_f + x_f + \lambda$$

$$\lambda = x_f$$

Process Q : State Q_0 : $\Pi_0(x_{ef}) = \Pi_{F_Q}\lambda$

$$\frac{\cancel{\Pi_0}}{\cancel{\Pi_0}}(x_{ef}) = \frac{\Pi_{F_Q}}{\cancel{\Pi_0}}\lambda$$

$$x_{ef} = \lambda$$

State F_Q : $\Pi_{F_Q}(x_{ef} + \lambda + \lambda) = \Pi_0 x_{ef} + \Pi_{F_Q}(x_{ef} + \lambda)$

$$\frac{\Pi_{F_Q}}{\cancel{\Pi_{F_Q}}}(x_{ef} + \lambda + \lambda) = \frac{\Pi_0}{\cancel{\Pi_{F_Q}}}x_{ef} + \frac{\Pi_{F_Q}}{\cancel{\Pi_{F_Q}}}(x_{ef} + \lambda)$$

$$\cancel{x_{ef}} + \cancel{\lambda} + \lambda = \cancel{x_{ef}} + x_{ef} + \cancel{\lambda}$$

$$\lambda = x_{ef}$$

Process C : State C_0 : $\Pi_0(x_f) = \Pi_{F_C}\lambda$

$$\frac{\cancel{\Pi_0}}{\cancel{\Pi_0}}(x_f) = \frac{\Pi_{F_C}}{\cancel{\Pi_0}}\lambda$$

$$x_f = \lambda$$

State F_C : $\Pi_{F_C}(x_f + \lambda + \lambda) = \Pi_0 x_f + \Pi_{F_C}(x_f + \lambda)$

$$\frac{\Pi_{F_C}}{\cancel{\Pi_{F_C}}}(x_f + \lambda + \lambda) = \frac{\Pi_0}{\cancel{\Pi_{F_C}}}x_f + \frac{\Pi_{F_C}}{\cancel{\Pi_{F_C}}}(x_f + \lambda)$$

$$\cancel{x_f} + \cancel{\lambda} + \lambda = \cancel{x_f} + x_f + \cancel{\lambda}$$

$$\lambda = x_f$$

6.5 Balancing the Global Balance Equations of DI Solution

Now we know that on both sides of equations we can add actions such that they are impossible and they can be balanced. So I restart the computation from the beginning and only when I will compute various ρ, then I will add the actions that involve states F with the addition of new rates of new safe-impossible actions in one or both sides of equations when necessary. Moreover, I assume that:

- when necessary to satisfy the condition of product form theorem, I create a lattice of phantom states F, one for each normal state
- to balance the various ingoing active impossible actions, I create a collapsed process C

I assume also that Π of a general phantom state F is equal to Π of his correspondence normal state, in this way:

- $\Pi_{F_{ij}} = \Pi_{ij}$ with $i, j \geq 0$ in R_1
- $\Pi_{F_m} = \Pi_m$ with $m \geq 0$ in R_2 and R_3

6.5.1 Balancing the Global Balance Equations of R_1

Now I restart the computation of R_1 from the beginning and only when I will compute ρ_1 and ρ_2, then I will add the impossible actions to satisfy the conditions of theorem. In particular I will add rates on the right and on the left of all of the equations. As I said, since we have to add both active and passive safe-impossible actions I need a lattice of phantom states F, one for each state and for simplicity I choose Π of state F_{xy} equal to Π of the correspondence state P_{xy}
After that I prove the balancing of all the new global balance equations of R_1.

For now (from the analysis in the previous sections), we have the following equations without impossible actions:

$$
\begin{cases}
\lambda_1 + x_f & = & \rho_2\mu_1 & + & \rho_1\rho_2\mu_1(1-p) & + & \rho_1 x_e & & & & & \text{state } P_0 \\
\lambda_1 + x_f + \mu_1 & = & \rho_2\mu_1 & + & \rho_1\rho_2\mu_1(1-p) & + & \rho_1 x_e & + & \frac{1}{\rho_2}\lambda_1 & & & \text{state } P_n \\
x_e + \lambda_1 + x_f & = & \rho_2\mu_1 p & + & \rho_1\rho_2\mu_1(1-p) & + & \rho_1 x_e & + & & & \frac{1}{\rho_1}x_f & \text{state } P_0^i \\
x_e + \lambda_1 + x_f + \mu_1 & = & \rho_2\mu_1 p & + & \rho_1\rho_2\mu_1(1-p) & + & \rho_1 x_e & + & \frac{1}{\rho_2}\lambda_1 & + & \frac{1}{\rho_1}x_f & \text{state } P_n^i
\end{cases}
$$

As usual, I compare state P_0 and state P_n (or P_0^i and P_n^i is the same) to find ρ_2:

$$
\begin{array}{lcllllll}
x_e + \lambda_1 + x_f + \mu_1 & = & \rho_2\mu_1 & + & \rho_1\rho_2\mu_1(1-p) & + & \rho_1 x_e & + & \frac{1}{\rho_2}\lambda_1 \\
x_e + \lambda_1 + x_f & = & \rho_2\mu_1 & + & \rho_1\rho_2\mu_1(1-p) & + & \rho_1 x_e & & \\
\hline
\mu_1 & = & & & & & & & \frac{1}{\rho_2}\lambda_1
\end{array}
$$

And finally get:

$$\mu_1 = \frac{1}{\rho_2}\rho_2\lambda_1$$

$$\rho_2\mu_1 = \frac{1}{\cancel{\rho_2}}\cancel{\rho_2}\lambda_1$$

$$\rho_2\frac{\cancel{\mu_1}}{\cancel{\mu_1}} = \frac{\lambda_1}{\mu_1}$$

$$\rho_2 = \frac{\lambda_1}{\mu_1}$$

Now I compare state P_0 and P_0^i (or P_n and P_n^i is the same), to find ρ_1:

$$
\begin{array}{lcllllll}
x_e + \lambda_1 + x_f & = & \rho_2\mu_1 p & + & \rho_1\rho_2\mu_1(1-p) & + & \rho_1 x_e & + & \frac{1}{\rho_1}x_f \\
\lambda_1 + x_f & = & \rho_2\mu_1 & + & \rho_1\rho_2\mu_1(1-p) & + & \rho_1 x_e & & \\
\hline
x_e & = & -\rho_2\mu_1(1-p) & + & & & & & \frac{1}{\rho_1}x_f
\end{array}
$$

And then finally we can get $\rho 1$:

$$x_e = -\rho_2\mu_1(1-p) + \frac{1}{\rho_1}x_f$$
$$x_e + \rho_2\mu_1(1-p) = \frac{1}{\rho_1}x_f$$
$$\rho_1(x_e + \frac{\lambda_1}{\cancel{\mu_1}}\cancel{\mu_1}(1-p)) = \frac{1}{\cancel{\rho_1}}x_f\cancel{\rho_1}$$
$$\rho_1\frac{\cancel{x_e + \lambda_1(1-p)}}{\cancel{x_e + \lambda_1(1-p)}} = \frac{x_f}{x_e + \lambda_1(1-p)}$$
$$\rho_1 = \frac{x_f}{x_e + \lambda_1(1-p)}$$

Also in this case I don't have to add any bound to ensure the positivity of ρ_1 because x_f, x_e and λ_1 are all greater than 0 (due to the fact that they are rates) and $(1-p) \geq 0$ because it is a probability from 0 to 1 so $0 \leq p \leq 1$. From this, ρ_1 will be always greater than 0.
Now I will check the balancing of equations, with all the actions involving phantom states F. This include three kind of actions:

- Actions needed to satisfy condition 1 and 2 of Product-form Theorem (for cooperation actions, all passive must be ingoing and all active must be outgoing from all states)
  - $(empty, \top)$ from states P_0 and P_n which is $(empty, x_e)$ with the $\top$-rate substitution
  - (a_0, r_{i0}) to states P_0^i
  - (a_0, r_{in}) to states P_n^i
- Actions from phantom state F not in cooperation set, they will not influence the satisfaction of condition 1 and 2 (since they don't cooperate) so I can add them freely
  - (τ, r_0) to states P_0
  - (τ, r_n) to states P_n
- Safe-Impossible outgoing passive action imp_1 from all states to their correspondence F states (remember that they also have to be added in the cooperation set)
  - $(imp_1, \top)$ from states P_0, P_n, P_0^i, P_n^i which is (imp_1, x_i) with the $\top$-rate substitution

The rates of a_0 actions, added from states F to P_0^i and P_n^i are active but impossible so I can freely choose their rates (and for now I put them as variables). Moreover, the satisfaction of two conditions in F states is ensured with the self-loops of all actions types in the cooperation set.

Then the summary, with impossible actions, is the following:

$$\begin{cases} x_e+\lambda_1+x_f+x_i = \rho_2\mu_1 + \rho_1\rho_2\mu_1(1-p) + \rho_1 x_e + \dfrac{\Pi_{F_0}}{\Pi_{00}}r_0 \\ x_e+\lambda_1+x_f+\mu_1+x_i = \rho_2\mu_1 + \rho_1\rho_2\mu_1(1-p) + \rho_1 x_e + \dfrac{1}{\rho_2}\lambda_1 + \dfrac{\Pi_{F_n}}{\Pi_{0n}}r_n \\ x_e+\lambda_1+x_f+x_i = \rho_2\mu_1 p + \rho_1\rho_2\mu_1(1-p) + \rho_1 x_e + \dfrac{1}{\rho_1}x_f + \dfrac{\Pi_{F_{i0}}}{\Pi_{i0}}r_{i0} \\ x_e+\lambda_1+x_f+\mu_1+x_i = \rho_2\mu_1 p + \rho_1\rho_2\mu_1(1-p) + \rho_1 x_e + \dfrac{1}{\rho_2}\lambda_1 + \dfrac{1}{\rho_1}x_f + \dfrac{\Pi_{F_{in}}}{\Pi_{in}}r_{in} \end{cases}$$

With:

$$\rho_1 = \frac{x_f}{x_e+\lambda_1(1-p)}$$

$$\rho_2 = \frac{\lambda_1}{\mu_1}$$

Now I will substitute them in the global balance equations:

State P_0

$$x_e+\lambda_1+x_f+x_i = \rho_2\mu_1+\rho_1\rho_2\mu_1(1-p)+\rho_1 x_e+\frac{\Pi_{F_0}}{\Pi_{00}}r_0$$

$$x_e+\lambda_1+x_f+x_i = \frac{\lambda_1}{\cancel{\mu_1}}\cancel{\mu_1}+\frac{x_f}{x_e+\lambda_1(1-p)}\frac{\lambda_1}{\cancel{\mu_1}}\cancel{\mu_1}(1-p)+\frac{x_f}{x_e+\lambda_1(1-p)}x_e+\frac{\Pi_{F_0}}{\Pi_{00}}r_0$$

$$x_e+\cancel{\lambda_1}+x_f+x_i = \cancel{\lambda_1}+\frac{x_f}{x_e+\lambda_1(1-p)}\lambda_1(1-p)+\frac{x_f}{x_e+\lambda_1(1-p)}x_e+\frac{\Pi_{F_0}}{\Pi_{00}}r_0$$

$$x_e+x_f+x_i = \frac{x_f}{x_e+\lambda_1(1-p)}(\lambda_1(1-p)+x_e)+\frac{\Pi_{F_0}}{\Pi_{00}}r_0$$

$$x_e+x_f+x_i = \frac{x_f}{\cancel{x_e+\lambda_1(1-p)}}\cancel{(x_e+\lambda_1(1-p))}+\frac{\Pi_{F_0}}{\Pi_{00}}r_0$$

$$x_e+\cancel{x_f}+x_i = \cancel{x_f}+\frac{\Pi_{F_0}}{\Pi_{00}}r_0$$

$$x_e+x_i = \frac{\cancel{\alpha}\rho_1^a\rho_2^b}{\cancel{\alpha}\rho_1^0\rho_2^0}r_0$$

$$x_e+x_i = \rho_1^a\rho_2^b r_0$$

I choose, as I said before, $a = 0$, $b = 0$ to make the equivalence of two Π:

$x_e + x_i = \cancel{\rho_1^0 \rho_2^0} r_0$
$r_0 = x_e + x_i$

State P_0 is **balanced**.

State P_n

$$x_e + \lambda_1 + x_f + \mu_1 + x_i = \rho_2\mu_1 + \rho_1\rho_2\mu_1(1-p) + \rho_1 x_e + \frac{1}{\rho_2}\lambda_1 + \frac{\Pi_{F_n}}{\Pi_{0n}} r_n$$

$$x_e + \lambda_1 + x_f + \mu_1 + x_i = \frac{\lambda_1}{\cancel{\mu_1}}\cancel{\mu_1} + \frac{x_f}{x_e + \lambda_1(1-p)}\frac{\lambda_1}{\cancel{\mu_1}}\cancel{\mu_1}(1-p) + \frac{x_f}{x_e + \lambda_1(1-p)} x_e + \frac{1}{\frac{\lambda_1}{\mu_1}}\lambda_1 + \frac{\Pi_{F_n}}{\Pi_{0n}} r_n$$

$$x_e + \cancel{\lambda_1} + x_f + \mu_1 + x_i = \cancel{\lambda_1} + \frac{x_f}{x_e + \lambda_1(1-p)}\lambda_1(1-p) + \frac{x_f}{x_e + \lambda_1(1-p)} x_e + \frac{\mu_1}{\cancel{\lambda_1}}\cancel{\lambda_1} + \frac{\Pi_{F_n}}{\Pi_{0n}} r_n$$

$$x_e + x_f + \cancel{\mu_1} + x_i = \frac{x_f}{x_e + \lambda_1(1-p)}(\lambda_1(1-p) + x_e) + \cancel{\mu_1} + \frac{\Pi_{F_n}}{\Pi_{0n}} r_n$$

$$x_e + x_f + x_i = \frac{x_f}{\cancel{x_e + \lambda_1(1-p)}}\cancel{(x_e + \lambda_1(1-p))} + \frac{\Pi_{F_n}}{\Pi_{0n}} r_n$$

$$x_e + \cancel{x_f} + x_i = \cancel{x_f} + \frac{\Pi_{F_n}}{\Pi_{0n}} r_n$$

$$x_e + x_i = \frac{\Pi_{F_n}}{\Pi_{0n}} r_n$$

(Remember that $\Pi_{F_n} = \rho_1^a \rho_2^b$ and I choose $a = 0$, $b = n$ so $\Pi_{F_n} = \rho_1^0 \rho_2^n = \Pi_n$)

$$x_e + x_i = \frac{\cancel{\alpha\rho_1^0\rho_2^n}}{\cancel{\alpha\rho_1^0\rho_2^n}} r_n$$

$x_e + x_i = r_n$
$r_n = x_e + x_i$

State P_n is **balanced**.

State P_0^i

$$x_e + \lambda_1 + x_f + x_i = \rho_2\mu_1 p + \rho_1\rho_2\mu_1(1-p) + \rho_1 x_e + \frac{1}{\rho_1}x_f + \frac{\Pi_{F_{i0}}}{\Pi_{i0}}r_{i0}$$

$$x_e+\lambda_1+x_f+x_i = \frac{\lambda_1}{\cancel{\mu_1}}\cancel{\mu_1}p+\frac{x_f}{x_e+\lambda_1(1-p)}\frac{\lambda_1}{\cancel{\mu_1}}\cancel{\mu_1}(1-p)+\rho_1 x_e+\frac{1}{\rho_1}x_f+\frac{\Pi_{F_{i0}}}{\Pi_{i0}}r_{i0}$$

$$x_e+\lambda_1+x_f+x_i = \lambda_1 p+\frac{x_f}{x_e+\lambda_1(1-p)}\lambda_1(1-p)+\frac{x_f}{x_e+\lambda_1(1-p)}x_e+\frac{1}{\rho_1}x_f+\frac{\Pi_{F_{i0}}}{\Pi_{i0}}r_{i0}$$

$$x_e+\lambda_1+x_f+x_i = \lambda_1 p+\frac{x_f}{x_e+\lambda_1(1-p)}(\lambda_1(1-p)+x_e)+\frac{1}{\frac{x_f}{x_e+\lambda_1(1-p)}}x_f+\frac{\Pi_{F_{i0}}}{\Pi_{i0}}r_{i0}$$

$$x_e+\lambda_1+x_f+x_i = \lambda_1 p+\frac{x_f}{\cancel{x_e+\lambda_1(1-p)}}\cancel{(x_e+\lambda_1(1-p))}+\frac{x_e+\lambda_1(1-p)}{\cancel{x_f}}\cancel{x_f}+\frac{\Pi_{F_{i0}}}{\Pi_{i0}}r_{i0}$$

$$\cancel{x_e}+\lambda_1+\cancel{x_f}+x_i = \lambda_1 p+\cancel{x_f}+\cancel{x_e}+\lambda_1(1-p)+\frac{\Pi_{F_{i0}}}{\Pi_{i0}}r_{i0}$$

$$\lambda_1 + x_i = \lambda_1(\cancel{p}+1-\cancel{p})+\frac{\Pi_{F_{i0}}}{\Pi_{i0}}r_{i0}$$

$$\cancel{\lambda_1}+x_i = \cancel{\lambda_1}+\frac{\Pi_{F_{i0}}}{\Pi_{i0}}r_{i0}$$

$$x_i = \frac{\Pi_{F_{i0}}}{\Pi_{i0}}r_{i0}$$

(Remember that $\Pi_{F_{i0}} = \rho_1^a\rho_2^b$ and I choose $a = i$, $b = 0$ so $\Pi_{F_{i0}} = \rho_1^i\rho_2^0 = \Pi_{i0}$)

$$x_i = \frac{\cancel{\alpha\rho_1^i\rho_2^0}}{\cancel{\alpha\rho_1^i\rho_2^0}}r_{i0}$$

$$x_i = r_{i0}$$
$$r_{i0} = x_i$$

State P_0^i is **balanced**.

State P_n^i

$$x_e+\lambda_1+x_f+\mu_1+x_i = \rho_2\mu_1 p+\rho_1\rho_2\mu_1(1-p)+\rho_1 x_e+\frac{1}{\rho_2}\lambda_1+\frac{1}{\rho_1}x_f+\frac{\Pi_{F_{in}}}{\Pi_{in}}r_{in}$$

$$x_e + \lambda_1 + x_f + \mu_1 + x_i = \frac{\lambda_1}{\cancel{\mu_1}}\cancel{\mu_1}p + \frac{x_f}{x_e + \lambda_1(1-p)}\frac{\lambda_1}{\cancel{\mu_1}}\cancel{\mu_1}(1-p) + \rho_1 x_e + \frac{1}{\frac{\lambda_1}{\mu_1}}\lambda_1 + \frac{1}{\rho_1}x_f + \frac{\Pi_{F_{in}}}{\Pi_{in}}r_{in}$$

$$x_e + \lambda_1 + x_f + \mu_1 + x_i = \lambda_1 p + \frac{x_f}{x_e + \lambda_1(1-p)}\lambda_1(1-p) + \rho_1 x_e + \frac{\mu_1}{\cancel{\lambda_1}}\cancel{\lambda_1} + \frac{1}{\rho_1}x_f + \frac{\Pi_{F_{in}}}{\Pi_{in}}r_{in}$$

$$x_e + \lambda_1 + x_f + \cancel{\mu_1} + x_i = \lambda_1 p + \frac{x_f}{x_e + \lambda_1(1-p)}\lambda_1(1-p) + \frac{x_f}{x_e + \lambda_1(1-p)}x_e + \cancel{\mu_1} + \frac{1}{\rho_1}x_f + \frac{\Pi_{F_{in}}}{\Pi_{in}}r_{in}$$

$$x_e + \lambda_1 + x_f + x_i = \lambda_1 p + \frac{x_f}{x_e + \lambda_1(1-p)}(\lambda_1(1-p) + x_e) + \frac{1}{\frac{x_f}{x_e + \lambda_1(1-p)}}x_f + \frac{\Pi_{F_{in}}}{\Pi_{in}}r_{in}$$

$$x_e + \lambda_1 + x_f + x_i = \lambda_1 p + \frac{x_f}{\cancel{x_e + \lambda_1(1-p)}}\cancel{(x_e + \lambda_1(1-p))} + \frac{x_e + \lambda_1(1-p)}{\cancel{x_f}}\cancel{x_f} + \frac{\Pi_{F_{in}}}{\Pi_{in}}r_{in}$$

$$\cancel{x_e} + \lambda_1 + \cancel{x_f} + x_i = \lambda_1 p + \cancel{x_f} + \cancel{x_e} + \lambda_1(1-p) + \frac{\Pi_{F_{in}}}{\Pi_{in}}r_{in}$$

$$\lambda_1 + x_i = \lambda_1(\cancel{p} + 1 - \cancel{p}) + \frac{\Pi_{F_{in}}}{\Pi_{in}}r_{in}$$

$$\cancel{\lambda_1} + x_i = \cancel{\lambda_1} + \frac{\Pi_{F_{in}}}{\Pi_{in}}r_{in}$$

$$x_i = \frac{\Pi_{F_{in}}}{\Pi_{in}}r_{in}$$

(Remember that $\Pi_{F_{in}} = \rho_1^a \rho_2^b$ and I choose $a = i$, $b = n$ so $\Pi_{F_{in}} = \rho_1^i \rho_2^n = \Pi_{in}$)

$$x_i = \frac{\cancel{\alpha \rho_1^i \rho_2^n}}{\cancel{\alpha \rho_1^i \rho_2^n}} r_{in}$$

$$x_i = r_{in}$$

$$r_{in} = x_i$$

State P_n^i is **balanced**.

In conclusion we have that:

$$\begin{cases} r_0 = x_e + x_i \\ r_n = x_e + x_i \\ r_{i0} = x_i \\ r_{in} = x_i \end{cases}$$

The final global balance equations, with all impossible actions, are the following:

$$\begin{cases} x_e + \lambda_1 + x_f + x_i & = & \rho_2\mu_1 & + & \rho_1\rho_2\mu_1(1-p) & + & \rho_1 x_e & + & & & & & x_e + x_i \\ x_e + \lambda_1 + x_f + \mu_1 + x_i & = & \rho_2\mu_1 & + & \rho_1\rho_2\mu_1(1-p) & + & \rho_1 x_e & + & \frac{1}{\rho_2}\lambda_1 & + & & & x_e + x_i \\ x_e + \lambda_1 + x_f + x_i & = & \rho_2\mu_1 p & + & \rho_1\rho_2\mu_1(1-p) & + & \rho_1 x_e & + & & & \frac{1}{\rho_1}x_f & + & x_i \\ x_e + \lambda_1 + x_f + \mu_1 + x_i & = & \rho_2\mu_1 p & + & \rho_1\rho_2\mu_1(1-p) & + & \rho_1 x_e & + & \frac{1}{\rho_2}\lambda_1 & + & \frac{1}{\rho_1}x_f & + & x_i \end{cases}$$

And the impossible actions are:

- (a_0, x_i) from F_{i0} to states P_0^i
- (a_0, x_i) from F_{in} to states P_n^i
- $(\tau, x_e + x_i)$ from F_0 to states P_0
- $(\tau, x_e + x_i)$ from F_n to states P_n
- (imp_1, x_i) form F states to states P_0, P_n, P_0^i, P_n^i

6.5.1.1 Balancing of F-Lattice

Now I have to check that also F-lattice is balanced. I have to check 4 main kinds of states:

- F_0
- F_n with $n > 0$
- F_{i0} with $i > 0$
- F_{in} with $n > 0$ and $i > 0$

The summary of their global balance equations is the following:

$$\begin{cases} x_i+\delta+x_e+x_0 = x_i+\delta + x_e + \dfrac{\Pi_{F_1}}{\Pi_{F_0}}r_{01} + \dfrac{\Pi_{F_{10}}}{\Pi_{F_0}}r_{02} \\ x_i+\delta+x_e+x_n = x_i+\delta + x_e + \dfrac{\Pi_{F_{n-1}}}{\Pi_{F_n}}r_{n1} + \dfrac{\Pi_{F_{n+1}}}{\Pi_{F_n}}r_{n2} + \dfrac{\Pi_{F_{n0}}}{\Pi_{F_n}}r_{n3} \\ x_i+\delta+x_{i0} = x_i+\delta + \dfrac{\Pi_{F_{i1}}}{\Pi_{F_{i0}}}r_{i01} + \dfrac{\Pi_{F_{i-1\,0}}}{\Pi_{F_{i0}}}r_{i02} + \dfrac{\Pi_{F_{i0\,1}}}{\Pi_{F_{i0}}}r_{i03} \\ x_i+\delta+x_{in} = x_i+\delta + \dfrac{\Pi_{F_{i\,n-1}}}{\Pi_{F_{in}}}r_{in1} + \dfrac{\Pi_{F_{i\,n+1}}}{\Pi_{F_{in}}}r_{in2} + \dfrac{\Pi_{F_{i-1\,n}}}{\Pi_{F_{in}}}r_{in3} + \dfrac{\Pi_{F_{in\,0}}}{\Pi_{F_{in}}}r_{in4} \end{cases}$$

Consider that $\delta = \mu_1 p + \mu_1(1-p) + \mu_1(1-p) + x_e + x_f + x_i = \mu_1 + \mu_1(1-p) + x_e + x_f + x_i$. It represents the rates of all cooperating actions in the self-loop of each phantom states F and it is the rate of the action All studied in the analysis. In this way all active and passive actions in the cooperation set are either ingoing and outgoing in F lattice.
Moreover, x_m and x_{mn} with $n \geq 0$ and $m \geq 0$ represent the sum of outgoing rates, one for each neighbor and I add also in the right part of the equations an ingoing rate of action for each neighbor of each phantom state.

Let's start the analysis:

State F_0

$$x_i+\delta+x_e+x_0 = x_i+\delta+x_e+\frac{\Pi_{F_1}}{\Pi_{F_0}}r_{01}+\frac{\Pi_{F_{10}}}{\Pi_{F_0}}r_{02}$$
$$\cancel{x_i}+\cancel{\delta}+\cancel{x_e}+x_0 = \cancel{x_i}+\cancel{\delta}+\cancel{x_e}+\frac{\alpha\rho_1^0\rho_2^1}{\alpha\rho_1^0\rho_2^0}r_{01}+\frac{\alpha\rho_1^1\rho_2^0}{\alpha\rho_1^0\rho_2^0}r_{02}$$
$$x_0 = \frac{\alpha\rho_1^0\rho_2^1}{\alpha\rho_1^0\rho_2^0}r_{01}+\frac{\alpha\rho_1^1\rho_2^0}{\alpha\rho_1^0\rho_2^0}r_{02}$$

Now I split x_0 in two variables (i.e. two rates of two different actions outgoing from F_0 and toward each of its neighbor), because F_0 has only 2 neighbor.

$$x_{01}+x_{02} = \frac{\cancel{\alpha\rho_1^0}\rho_2^1}{\cancel{\alpha\rho_1^0}\rho_2^0}r_{01}+\frac{\cancel{\alpha}\rho_1^1\cancel{\rho_2^0}}{\cancel{\alpha}\rho_1^0\cancel{\rho_2^0}}r_{02}$$
$$x_{01}+x_{02} = \rho_2 r_{01}+\rho_1 r_{02}$$

State F_0 is balanced.

In conclusion we have that:
$x_{01} = \rho_2 r_{01}$
$x_{02} = \rho_1 r_{02}$

I choose a generic rate ω for both r_{01} and r_{02} so:

- From state F_1 to state F_0 there is an action with rate ω (from the chosen value of r_{01})
- From state F_0 to state F_1 there is an action with rate $\rho_2\omega$ (from value of x_{01})
- From state F_{10} to state F_0 there is an action with rate ω (from the chosen value of r_{02})
- From state F_0 to state F_{10} there is an action with rate $\rho_1\omega$ (from value of x_{02})

In this way, generalizing the previous reasoning, we know that considering F_{mn} a generic F-state, every action (if there is any):

- From $F_{m\,n-1}$ (i.e. column $= n-1$; row $= m$) to F_{mn} there will be and action with rate $\rho_2\omega$
- From $F_{m\,n+1}$ (i.e. column $= n+1$; row $= m$) to F_{mn} there will be and action with rate ω
- From $F_{m-1\,n}$ (i.e. column $= n$; row $= m-1$) to F_{mn} there will be and action with rate $\rho_1\omega$
- From $F_{m+1\,n}$ (i.e. column $= n$; row $= m+1$) to F_{mn} there will be and action with rate ω
- From F_{mn} to $F_{m\,n+1}$ (i.e. column $= n+1$; row $= m$) there will be and action with rate $\rho_2\omega$
- From F_{mn} to $F_{m+1\,n}$ (i.e. column $= n$; row $= m+1$) there will be and action with rate $\rho_1\omega$

Moreover, consider also that we know all Π_{xy} and for all of them holds: $\frac{\Pi_{xy}}{\Pi_{wz}} = \rho_1^{x-w}\rho_2^{y-z}$.
I also split x_n and x_{i0} in 3 variables, one for each neighbor and x_{in} in 4 variables, also here one for each neighbor.
I can update the system of global balance equations of the F-lattice with information about Π and F_0:

$$\begin{cases} x_i+\delta+x_e+\rho_2\omega+\rho_1\omega & = & x_i+\delta+x_e & + & & & \rho_2\omega & + & & \rho \\ x_i+\delta+x_e+x_{n1}+x_{n2}+x_{n3} & = & x_i+\delta+x_e & + & \rho_2^{-1}r_{n1} & + & \rho_2 r_{n2} & + & & \rho \\ x_i+\delta+x_{i01}+x_{i02}+x_{i03} & = & x_i+\delta & + & & & \rho_2 r_{i01} & + & \rho_1^{-1}r_{i02} & + & \rho \\ x_i+\delta+x_{in1}+x_{in2}+x_{in3}+x_{in4} & = & x_i+\delta & + & \rho_2^{-1}r_{in1} & + & \rho_2 r_{in2} & + & \rho_1^{-1}r_{in3} & + & \rho \end{cases}$$

I can now proceed with the analysis:

State F_n

Remember that F_n (column $= n$; row $= 0$) has following rate of actions:

- r_{n1} from column $= n-1$; row $= 0$, so it is $\rho_2\omega$
- r_{n2} from column $= n+1$; row $= 0$, so it is ω
- r_{n3} from column $= n$; row $= 1$, so it is ω
- x_{n2} to column $= n+1$; row $= 0$, so it is $\rho_2\omega$
- x_{n3} to column $= n$; row $= 1$, so it is $\rho_1\omega$

Then we have that:
$x_i+\delta+x_e+x_{n1}+x_{n2}+x_{n3} = x_i+\delta+x_e+\rho_2^{-1}r_{n1}+\rho_2 r_{n2}+\rho_1 r_{n3}$
$\cancel{x_i+\delta+x_e}+x_{n1}+x_{n2}+x_{n3} = \cancel{x_i+\delta+x_e}+\rho_2^{-1}r_{n1}+\rho_2 r_{n2}+\rho_1 r_{n3}$
$x_{n1}+x_{n2}+x_{n3} = \rho_2^{-1}r_{n1}+\rho_2 r_{n2}+\rho_1 r_{n3}$
$x_{n1}+\rho_2\omega+\rho_1\omega = \rho_2^{-1}\rho_2\omega+\rho_2\omega+\rho_1\omega$
$x_{n1}+\cancel{\rho_2\omega}+\cancel{\rho_1\omega} = \cancel{\rho_2^{-1}\rho_2}\omega+\cancel{\rho_2\omega}+\cancel{\rho_1\omega}$
$x_{n1} = \omega$

State F_n is balanced.

If the reasoning is the same, also the actions from a state with same column and previous row will be equal to ω. I can conclude the reasoning and in this case, considering a generic F-state F_{mn}, every action (if there is any):

- From $F_{m\,n-1}$ (i.e. column $= n-1$; row $= m$) to F_{mn} there will be and action with rate $\rho_2\omega$
- From $F_{m\,n+1}$ (i.e. column $= n+1$; row $= m$) to F_{mn} there will be and action with rate ω
- From $F_{m-1\,n}$ (i.e. column $= n$; row $= m-1$) to F_{mn} there will be and action with rate $\rho_1\omega$
- From $F_{m+1\,n}$ (i.e. column $= n$; row $= m+1$) to F_{mn} there will be and action with rate ω
- From F_{mn} to $F_{m\,n-1}$ (i.e. column $= n-1$; row $= m$) there will be and action with rate ω
- From F_{mn} to $F_{m\,n+1}$ (i.e. column $= n+1$; row $= m$) there will be and action with rate $\rho_2\omega$
- From F_{mn} to $F_{m-1\,n}$ (i.e. column $= n$; row $= m-1$) there will be and action with rate ω
- From F_{mn} to $F_{m+1\,n}$ (i.e. column $= n$; row $= m+1$) there will be and action with rate $\rho_1\omega$

Now I can further update the system with this information:

$$\begin{cases} x_i+\delta+x_e+\rho_2\omega+\rho_1\omega & = & x_i+\delta+x_e & + & & & \rho_2\omega & + & & & \rho_1\omega & \text{state } F_0 \\ x_i+\delta+x_e+\omega+\rho_2\omega+\rho_1\omega & = & x_i+\delta+x_e & + & \rho_2^{-1}\rho_2\omega & + & \rho_2\omega & + & & & \rho_1\omega & \text{state } F_n \\ x_i+\delta+\rho_2\omega+\omega+\rho_1\omega & = & x_i+\delta & + & & & \rho_2\omega & + & \rho_1^{-1}\rho_1\omega & + & \rho_1\omega & \text{state } F_{i0} \\ x_i+\delta+\omega+\rho_2\omega+\omega+\rho_1\omega & = & x_i+\delta & + & \rho_2^{-1}\rho_2\omega & + & \rho_2\omega & + & \rho_1^{-1}\rho_1\omega & + & \rho_1\omega & \text{state } F_{in} \end{cases}$$

With a semplification, we get:

$$\begin{cases} x_i+\delta+x_e+\rho_2\omega+\rho_1\omega & = & x_i+\delta+x_e & + & & \rho_2\omega & + & & \rho_1\omega & \text{state } F_0 \\ x_i+\delta+x_e+\omega+\rho_2\omega+\rho_1\omega & = & x_i+\delta+x_e & + & \omega & + & \rho_2\omega & + & & \rho_1\omega & \text{state } F_n \\ x_i+\delta+\rho_2\omega+\omega+\rho_1\omega & = & x_i+\delta & + & & \rho_2\omega & + & \omega & + & \rho_1\omega & \text{state } F_{i0} \\ x_i+\delta+\omega+\rho_2\omega+\omega+\rho_1\omega & = & x_i+\delta & + & \omega & + & \rho_2\omega & + & \omega & + & \rho_1\omega & \text{state } F_{in} \end{cases}$$

And finally it is trivial to see that stat F_{i0} and F_{in} are balanced.

6.5.2 Updating Analysis of Process of R_1 with Safe-impossible Actions

Since I add some actions in my model and I also found the value of some variables, I will now update states and their analysis. All the data I found, are the following:

$\rho_1 = \dfrac{x_f}{x_e + \lambda_1(1-p)}$
$\rho_2 = \dfrac{\lambda_1}{\mu_1}$
$r_0 = x_e + x_i$
$r_n = x_e + x_i$
$r_{i0} = x_i$
$r_{in} = x_i$
$\Pi_{F_{xy}} = \alpha\rho_1^x\rho_2^y$ (with $x \geq 0$ and $y \geq 0$)

Consider that x rates are the substitution of $\top$ rates, the actions, between normal and phantom states, are:

- (a_0, x_i) to states P_0^i and P_n^i
- $(\tau, x_e + x_i)$ to states P_0 and P_n^i
- $(imp_1, x_i q)$ to states P_0, P_n, P_0^i, P_n^i

And between the phantom lattice, the actions are:

- $(\tau, \rho_2\omega)$ from column -1; same row
- (τ, ω) from column $+1$; same row
- $(\tau, \rho_1\omega)$ from same column; row -1
- (τ, ω) from same column; row $= +1$
- (τ, ω) to column -1; same row
- $(\tau, \rho_2\omega)$ to column $+1$; same row
- (τ, ω) to same column; row -1
- $(\tau, \rho_1\omega)$ to same column; row $+1$

The first state to analyze is state P_0 (row = 0, column = 0), I will show action names only of those involved in state F. Remember that $\Pi_{rc} = \alpha\Pi_r \cdot \Pi_c = \alpha\rho_1^r\rho_2^c$

$$\Pi_{00}(\overbrace{x_e + \lambda_1 + x_f + x_i}^{Out}) = \Pi_{01}\mu_1 + \Pi_{11}\mu_1(1-p) + \Pi_{10}x_e + \Pi_{F_0}(x_e + x_i)$$

$$\begin{aligned}
\frac{\cancel{\Pi_{00}}}{\cancel{\Pi_{00}}}(x_e + \lambda_1 + x_f + x_i) &= \frac{\Pi_{01}}{\Pi_{00}}\mu_1 + \frac{\Pi_{11}}{\Pi_{00}}\mu_1(1-p) + \frac{\Pi_{10}}{\Pi_{00}}x_e + \frac{\Pi_{F_0}}{\Pi_{00}}(x_e + x_i) \\
&= \frac{\cancel{\alpha}\rho_1^{\cancel{0}}\rho_2^1}{\cancel{\alpha}\rho_1^0\rho_2^0}\mu_1 + \frac{\cancel{\alpha}\rho_1^1\rho_2^1}{\cancel{\alpha}\rho_1^0\rho_2^0}\mu_1(1-p) + \frac{\cancel{\alpha}\rho_1^1\rho_2^{\cancel{0}}}{\cancel{\alpha}\rho_1^0\rho_2^0}x_e + \frac{\cancel{\alpha}\rho_1^{\cancel{0}}\rho_2^{\cancel{0}}}{\cancel{\alpha}\rho_1^0\rho_2^0}(x_e + x_i) \\
&= \rho_2\mu_1 + \rho_1\rho_2\mu_1(1-p) + \rho_1 x_e + x_e + x_i
\end{aligned}$$

The derivation graph is the following:

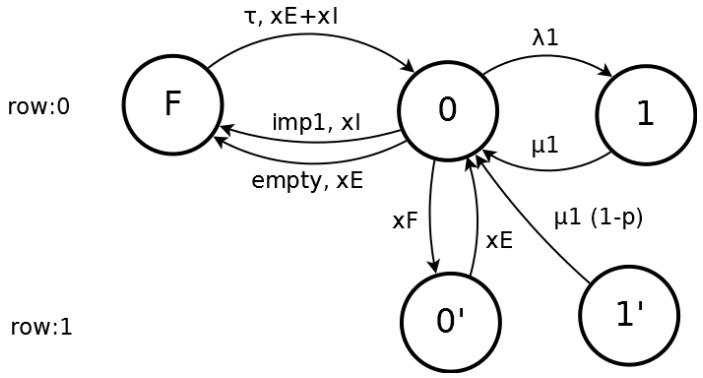

Figure 6.36: State P_0 (DI) and its neighbors states

Then I analyze state P_n (row = 0, column = $n \geq 1$).

$\Pi_{0n}(x_e+\lambda_1+x_f+\mu_1+x_i) = \Pi_{0\,n+1}\mu_1 + \Pi_{1\,n+1}\mu_1(1-p) + \Pi_{1n}x_e + \Pi_{0\,n-1}\lambda_1 + \Pi_{F_n}(x_e+x_i)$

$$\begin{aligned}
\cancel{\Pi_{0n}}(x_e+\lambda_1+x_f+\mu_1+x_i) &= \frac{\Pi_{0\,n+1}}{\Pi_{0n}}\mu_1 + \frac{\Pi_{1\,n+1}}{\Pi_{0n}}\mu_1(1-p) + \frac{\Pi_{1n}}{\Pi_{0n}}x_e + \frac{\Pi_{0\,n-1}}{\Pi_{0n}}\lambda_1 + \frac{\Pi_{F_n}}{\Pi_{0n}}(x_e+x_i) \\
&= \frac{\cancel{\alpha}\rho_1^{\cancel{0}}\rho_2^{n+1}}{\cancel{\alpha}\rho_1^{\cancel{0}}\rho_2^{n}}\mu_1 + \frac{\cancel{\alpha}\rho_1^{1}\rho_2^{n+1}}{\cancel{\alpha}\rho_1^{\cancel{0}}\rho_2^{n}}\mu_1(1-p) + \frac{\cancel{\alpha}\rho_1^{1}\rho_2^{\cancel{n}}}{\cancel{\alpha}\rho_1^{\cancel{0}}\rho_2^{n}}x_e + \frac{\cancel{\alpha}\rho_1^{0}\rho_2^{n-1}}{\cancel{\alpha}\rho_1^{\cancel{0}}\rho_2^{\cancel{n}}}\lambda_1 + \frac{\cancel{\alpha}\rho_1^{0}\rho_2^{n}}{\cancel{\alpha}\rho_1^{\cancel{0}}\rho_2^{n}} \\
&= \rho_2\mu_1 + \rho_1\rho_2\mu_1(1-p) + \rho_1 x_e + \frac{1}{\rho_2}\lambda_1 + x_e + x_i
\end{aligned}$$

The derivation graph is the following:

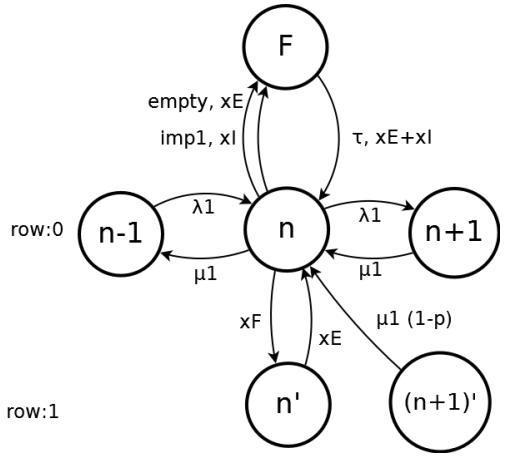

Figure 6.37: State P_n (DI) and its neighbors states

Now I analyze state P_0^i (row = $i \geq 1$, column = 0).

$$\Pi_{i0}(x_e+\lambda_1+x_f+x_i) = \Pi_{i1}\mu_1 p+\Pi_{i+1\,1}\mu_1(1-p)+\Pi_{i+1\,0}x_e+\Pi_{i-1\,0}x_f+\Pi_{F_{i0}}x_i$$

$$\begin{aligned}\cancel{\Pi_{i0}}(x_e+\lambda_1+x_f+x_i) &= \frac{\Pi_{i1}}{\Pi_{i0}}\mu_1 p+\frac{\Pi_{i+1\,1}}{\Pi_{i0}}\mu_1(1-p)+\frac{\Pi_{i+1\,0}}{\Pi_{i0}}x_e+\frac{\Pi_{i-1\,0}}{\Pi_{i0}}x_f+\frac{\Pi_{F_{i0}}}{\Pi_{i0}}x_i \\ &= \frac{\cancel{\alpha}\cancel{\rho_1^i}\cancel{\rho_2^1}}{\cancel{\alpha}\cancel{\rho_1^i}\rho_2^0}\mu_1 p+\frac{\cancel{\alpha}\cancel{\rho_1^{i+1}}\cancel{\rho_2^1}}{\cancel{\alpha}\cancel{\rho_1^i}\rho_2^0}\mu_1(1-p)+\frac{\cancel{\alpha}\cancel{\rho_1^{i+1}}\cancel{\rho_2^0}}{\cancel{\alpha}\cancel{\rho_1^i}\rho_2^0}x_e+\frac{\cancel{\alpha}\cancel{\rho_1^{i-1}}\cancel{\rho_2^0}}{\cancel{\alpha}\cancel{\rho_1^1}\cancel{\rho_2^0}}x_f+\frac{\cancel{\alpha}\cancel{\rho_1^i}\cancel{\rho_2^0}}{\cancel{\alpha}\cancel{\rho_1^i}\cancel{\rho_2^0}}x_i \\ &= \rho_2\mu_1 p+\rho_1\rho_2\mu_1(1-p)+\rho_1 x_e+\frac{1}{\rho_1}x_f+x_i\end{aligned}$$

Its derivation graph is the following:

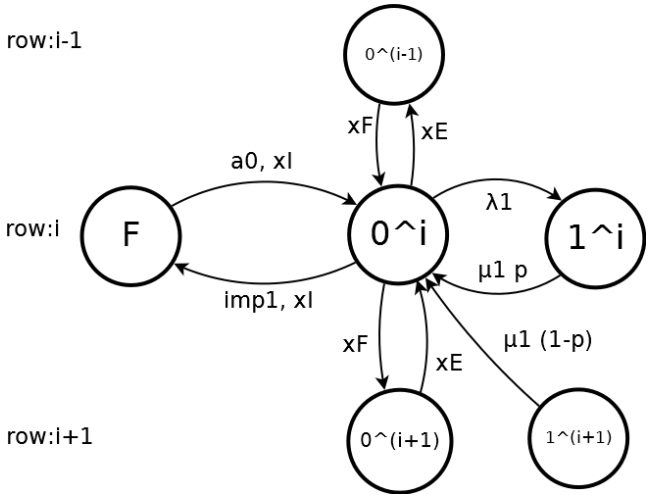

Figure 6.38: State P_0^i (DI) and its neighbors states

Finally I analyze state P_n^i (row = $i \geq 1$, column = $n \geq 1$).

$$\begin{aligned}\Pi_{in}(x_e+\lambda_1+x_f+\mu_1 p+\mu_1(1-p)+x_i) = &\Pi_{i\,n+1}\mu_1 p+\Pi_{i+1\,n+1}\mu_1(1-p)+\Pi_{i+1\,n}x_e+\Pi_{i\,n-1}\lambda_1+ \\ &+\Pi_{i-1\,n}x_f+\Pi_{F_{in}}x_i\end{aligned}$$

$$
\begin{aligned}
\cancel{\Pi_{in}}(x_e+\lambda_1+x_f+\mu_1+x_i) &= \frac{\Pi_{i\,n+1}}{\Pi_{in}}\mu_1 p + \frac{\Pi_{i+1\,n+1}}{\Pi_{in}}\mu_1(1-p) + \frac{\Pi_{i+1\,n}}{\Pi_{in}}x_e + \frac{\Pi_{i\,n-1}}{\Pi_{in}}\lambda_1 + \frac{\Pi_{i-1\,n}}{\Pi_{in}}x_f + \\
&= \frac{\cancel{\alpha\rho_1^i\rho_2^{n+1}}}{\cancel{\alpha\rho_1^i\rho_2^n}}\mu_1 p + \frac{\cancel{\alpha\rho_1^{i+1}\rho_2^{n+1}}}{\cancel{\alpha\rho_1^i\rho_2^n}}\mu_1(1-p) + \frac{\cancel{\alpha\rho_1^{i+1}\rho_2^n}}{\cancel{\alpha\rho_1^i\rho_2^n}}x_e + \frac{\Pi_{i\,n-1}}{\Pi_{in}}\lambda_1 + \frac{\Pi_{i-1} \\
&= \rho_2\mu_1 p + \rho_1\rho_2\mu_1(1-p) + \rho_1 x_e + \frac{\cancel{\alpha\rho_1^i\rho_2^{n-1}}}{\cancel{\alpha\rho_1^i\rho_2^n}}\lambda_1 + \frac{\cancel{\alpha\rho_1^{i-1}\rho_2^n}}{\cancel{\alpha\rho_1^i\rho_2^n}}x_f + \frac{\cancel{\alpha\rho_1^i\rho_2^n}}{\cancel{\alpha\rho_1^i\rho_2^n}}x \\
&= \rho_2\mu_1 p + \rho_1\rho_2\mu_1(1-p) + \rho_1 x_e + \frac{1}{\rho_2}\lambda_1 + \frac{1}{\rho_1}x_f + x_i
\end{aligned}
$$

Its derivation graph is the following:

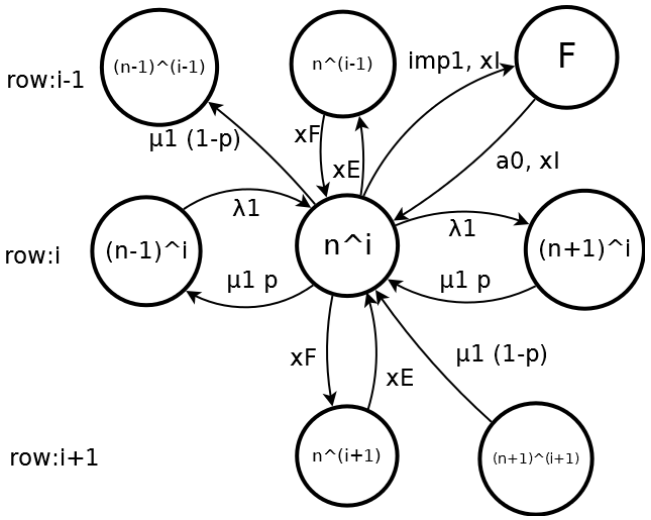

Figure 6.39: State P_n^i (DI) and its neighbors states

Now it is necessary to look also at the phantom lattice of states F and their equations. I will show now how the phantom lattice (and so states F) interacts with the rest of the system and its balance equations.

consider that:
$\delta = \mu_1 p + \mu_1(1-p) + \mu_1(1-p) + x_e + x_f + x_i = \mu_1 + \mu_1(1-p) + x_e + x_f + x_i$
and as a self-loop it doesn't count in the equation because:
$\Pi_{F_{xy}}(\underbrace{\delta}_{outgoing}) = \underbrace{\Pi_{F_{xy}}(\delta)}_{ingoing}$

The first state I analyze is F_0 (row $= 0$, column $= 0$):

$$\Pi_{F_0}(\overbrace{x_i + x_e + \rho_2\omega + \rho_1\omega}^{out}) = \Pi_{00}(x_i + x_e) + \Pi_{F_1}\omega + \Pi_{F_{10}}\omega$$

$$\begin{aligned}
\cancel{\Pi_{F_0}}(x_i + x_e + \rho_2\omega + \rho_1\omega) &= \frac{\Pi_{00}}{\Pi_{F_0}}(x_i + x_e) + \frac{\Pi_{F_1}}{\Pi_{F_0}}\omega + \frac{\Pi_{F_{10}}}{\Pi_{F_0}}\omega \\
&= \frac{\alpha\cancel{\rho_1^0}\cancel{\rho_2^0}}{\alpha\cancel{\rho_1^0}\cancel{\rho_2^0}}(x_i + x_e) + \frac{\cancel{\alpha}\cancel{\rho_1^0}\rho_2^1}{\alpha\cancel{\rho_1^0}\cancel{\rho_2^0}}\omega + \frac{\cancel{\alpha}\rho_1^1\cancel{\rho_2^0}}{\alpha\cancel{\rho_1^0}\cancel{\rho_2^0}}\omega \\
&= x_i + x_e + \rho_2\omega + \rho_1\omega
\end{aligned}$$

The derivation graph is the following:

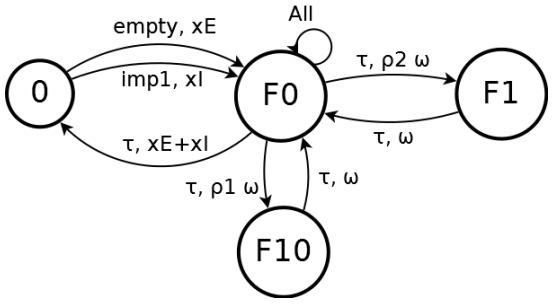

Figure 6.40: State F_0 (DI) and its neighbors states

$All = (a, \mu_1 p).F_0 + (a_0, \mu_1(1-p)).F_0 + (a_1, \mu_1(1-p)).F_0 + (empty, \top).F_0 + (fill, \top).F_0 + (imp_1, x_i).F_0$

Then I analyze the states of the lattice F_n (row $= 0$, column > 0):

$$\Pi_{F_n}(x_i + x_e + \omega + \rho_2\omega + \rho_1\omega) = \Pi_{0n}(x_i + x_e) + \Pi_{F_{n-1}}\rho_2\omega + \Pi_{F_{n+1}}\omega + \Pi_{F_{1n}}\omega$$

$$\begin{aligned}
\cancel{\Pi_{F_n}}(x_i + x_e + \omega + \rho_2\omega + \rho_1\omega) &= \frac{\Pi_{0n}}{\Pi_{F_n}}(x_i + x_e) + \frac{\Pi_{F_{n-1}}}{\Pi_{F_n}}\rho_2\omega + \frac{\Pi_{F_{n+1}}}{\Pi_{F_n}}\omega + \frac{\Pi_{F_{1n}}}{\Pi_{F_n}}\omega \\
&= \frac{\alpha\rho_1^{\cancel{0}}\rho_2^{n}}{\alpha\rho_1^{\cancel{0}}\rho_2^{n}}(x_i + x_e) + \frac{\alpha\rho_1^{\cancel{0}}\rho_2^{n-1}}{\alpha\rho_1^{\cancel{0}}\rho_2^{\cancel{n}}}\rho_2\omega + \rho_2\omega + \rho_1\omega \\
&= x_i + x_e + \omega + \rho_2\omega + \rho_1\omega
\end{aligned}$$

The derivation graph is the following:

Now I analyze the states of the lattice F_{i0} (row > 0, column $= 0$):

$$\Pi_{F_{i0}}(x_i + x_e + \rho_2\omega + \omega + \rho_1\omega) = \Pi_{i0}(x_i + x_e) + \Pi_{F_{i1}}\omega + \Pi_{F_{i-1\,0}}\rho_1\omega + \Pi_{F_{i+1\,0}}\omega$$

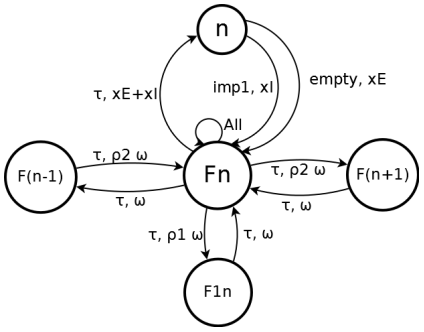

Figure 6.41: State F_n (DI) and its neighbors states

$$\begin{aligned}\cancel{\Pi_{F_{i0}}}(x_i + x_e + \rho_2\omega + \omega + \rho_1\omega) &= \frac{\Pi_{i0}}{\Pi_{F_{i0}}}(x_i + x_e) + \frac{\Pi_{F_{i1}}}{\Pi_{F_{i0}}}\omega + \frac{\Pi_{F_{i-1\,0}}}{\Pi_{F_{i0}}}\rho_1\omega + \frac{\Pi_{F_{i+1\,0}}}{\Pi_{F_{i0}}}\omega \\ &= \frac{\cancel{\alpha\rho_1^i\rho_2^0}}{\cancel{\alpha\rho_1^i\rho_2^0}}(x_i + x_e) + \rho_2\omega + \rho_1^{-1}\rho_1\omega + \rho_1\omega \\ &= x_i + x_e + \rho_2\omega + \omega + \rho_1\omega\end{aligned}$$

Its derivation graph is the following:

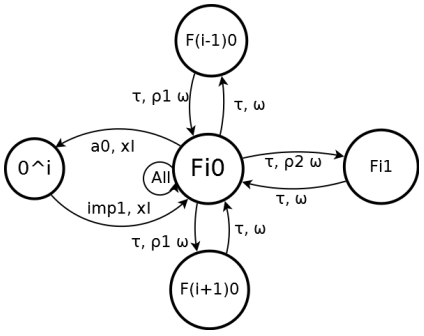

Figure 6.42: State F_0^i (DI) and its neighbors states

Finally I analyze the states of the lattice F_{in} (row > 0, column > 0):

$$\Pi_{F_{in}}(x_i + x_e + \omega + \rho_2\omega + \omega + \rho_1\omega) = \Pi_{in}(x_i + x_e) + \Pi_{F_{i\,n-1}}\rho_2\omega + \Pi_{F_{i\,n+1}}\omega + \Pi_{F_{i-1\,n}}\rho_1\omega + \Pi_{F_{i+1\,n}}\omega$$

$$\begin{aligned}\cancel{\Pi_{F_{in}}}(x_i + x_e + \omega + \rho_2\omega + \omega + \rho_1\omega) &= \frac{\Pi_{in}}{\Pi_{F_{in}}}(x_i + x_e) + \frac{\Pi_{F_{i\,n-1}}}{\Pi_{F_{in}}}\rho_2\omega + \frac{\Pi_{F_{i\,n+1}}}{\Pi_{F_{in}}}\omega + \frac{\Pi_{F_{i-1\,n}}}{\Pi_{F_{in}}}\rho_1\omega + \frac{\Pi_{F_{i+}}}{\Pi_F} \\ &= \frac{\cancel{\alpha\rho_1^i\rho_2^n}}{\cancel{\alpha\rho_1^i\rho_2^n}}(x_i + x_e) + \rho_2^{-1}\rho_2\omega + \rho_2\omega + \rho_1^{-1}\rho_1\omega + \rho_1\omega \\ &= x_i + x_e + \omega + \rho_2\omega + \omega + \rho_1\omega\end{aligned}$$

Its derivation graph is the following:

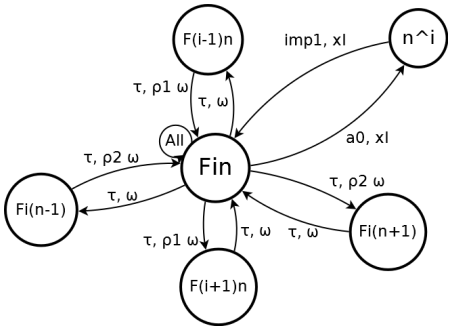

Figure 6.43: State F_n^i (DI) and its neighbors states

The final process R_1 is the following:

$$
R_1 = \begin{cases}
P_0 = & (\tau,\lambda_1).P_1 + (fill,\top).P'_0 + (\overline{empty},\top).F_0 + (\overline{imp_1},\top).F_0 \\
P_0^m = & (\tau,\lambda_1).P_1^m + (fill,\top).P_0^{m+1} + (empty,\top).P_0^{m-1} + (\overline{imp_1},\top).F_{m0} \\
P_n = & (\tau,\lambda_1).P_{n+1} + (fill,\top).P'_n + (a,\mu_1 p).P_{n-1} + (a_0,\mu_1(1-p)).P_{n-1} + \\
& +(\overline{empty},\top).F_n + (\overline{imp_1},\top).F_n \\
P_n^m = & (\tau,\lambda_1).P_{n+1}^m + (fill,\top).P_n^{m+1} + (empty,\top).P_n^{m-1} + (a,\mu_1 p).P_{n-1}^m + \\
& +(a_1,\mu_1(1-p)).P_{n-1}^{m-1}(\overline{imp_1},\top).F_{mn} \\
 & \\
F_0 = & (\overline{\tau},x_e+x_l).P_0 + (\overline{tau},\rho_2\omega).F_1 + (\overline{tau},\rho_1\omega).F_{10} + A_0 \\
F_{m0} = & (\overline{a_0},x_l).P_{m0} + (\overline{tau},\rho_2\omega).F_{m1} + (\overline{tau},\omega).F_{m-1\,0} + (\overline{tau},\rho_1\omega).F_{m+1\,0} + A_{m0} \\
F_n = & (\overline{\tau},x_e+x_l).P_n + (\overline{tau},\omega).F_{n-1} + (\overline{tau},\rho_2\omega).F_{n+1} + (\overline{tau},\rho_1\omega).F_{1n} + A_n \\
P_{mn} = & (\overline{a_0},x_l).P_{mn} + (\overline{tau},\omega).F_{m\,n-1} + (\overline{tau},\rho_2\omega).F_{m\,n+1} + (\overline{tau},\omega).F_{m-1\,n} + \\
& +(\overline{tau},\rho_1\omega).F_{m+1\,n} + A_{mn}
\end{cases}
$$

with $m > 0$ and $n > 0$ and $A_{xy} = (a,\mu_1 p).F_{xy} + (a_0,\mu_1(1-p)).F_{xy} + (a_1,\mu_1(1-p)).F_{xy} + (empty,\top).F_{xy} + (fill,\top).F_{xy} + (imp_1,x_i q).F_{xy} + (imp_3,\top).F_{xy}$

The final derivation graph of R_1 is the following:

6.5.2.1 Steady State Distribution of Process R_1

To find the steady state distribution of process R_1, the last thing to compute is α in such a way that:

$$\left(\sum_{n=0}^{\top}\sum_{m=0}^{\top}\Pi_{nm} + \Pi_{F_{nm}}\right) = 1$$

Since $\Pi_{nm} = \Pi_{F_{nm}}$ I can write:

$$\left(\sum_{n=0}^{\top}\sum_{m=0}^{\top}2\Pi_{nm}\right) = 1$$

But this is trivial, using the sum of geometric series, since:

$$\left(\sum_{n=0}^{\top}\sum_{m=0}^{\top}2\alpha\rho_1^n\rho_2^m\right) = 1$$

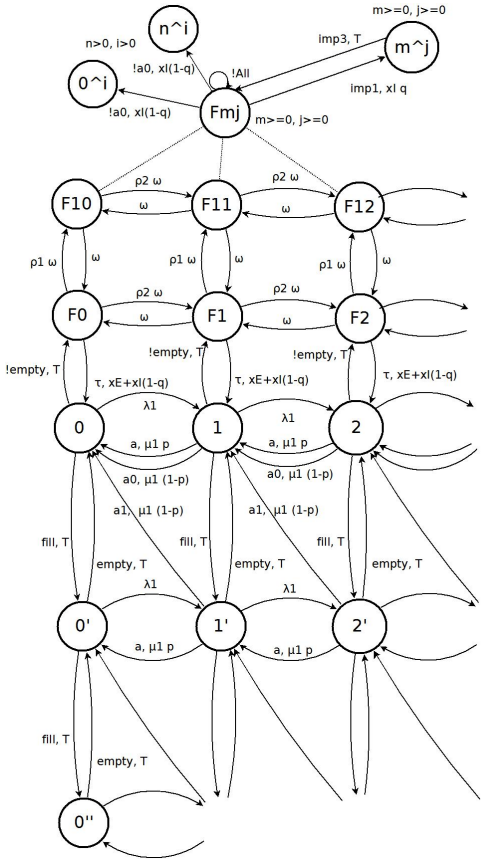

Figure 6.44: Process R_1 (DI) with impossible actions

$$2\alpha\left(\sum_{n=0}^{\top}\sum_{m=0}^{\top}\rho_1^n\rho_2^m\right)=1$$
$$2\alpha\left(\sum_{n=0}^{\top}\rho_1^n\sum_{m=0}^{\top}\rho_2^m\right)=1$$
$$2\alpha\left(\frac{1}{1-\rho_1}\frac{1}{1-\rho_2}\right)=1$$
$$2\alpha\frac{1}{\cancel{1-\rho_1}}\cancel{(1-\rho_1)}\frac{1}{\cancel{1-\rho_2}}\cancel{(1-\rho_2)}=1(1-\rho_1)(1-\rho_2)$$
$$\frac{2}{2}\alpha=\frac{(1-\rho_1)(1-\rho_2)}{2}$$

$$\alpha = \frac{(1-\rho_1)(1-\rho_2)}{2}$$

So in conclusion the steady state distribution of R_1 is the following:
$\Pi_{nm} = \Pi_n \Pi_m = \alpha \rho_1 \rho_2$

Where: $\rho_1 = \dfrac{x_f}{x_e + \lambda_1(1-p)}$
$\rho_2 = \dfrac{\lambda_1}{\mu_1}$
$\alpha = \dfrac{(1-\rho_1)(1-\rho_2)}{2}$

6.5.3 Analysis and Balancing the Global Balance Equations of R_2

Process R_2 is composed by only two main types of states:

1. State Q_0
2. State Q_n with $n > 0$

All of its state can be attributed to one of this 2 types.
I suppose that the product form of R_2 will be the following:

$$\Pi_n = \alpha\rho^n$$

6.5.3.1 Analysis of Q_0

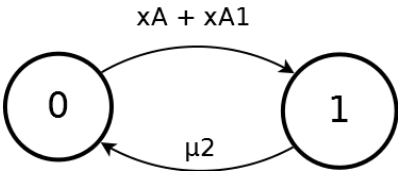

Figure 6.45: State Q_0 (DI) and its neighbors states

The general case of process R_2:

$\Pi_n = \alpha\Pi_n = \alpha\rho^n$

In Q_0 has $n = 0$.

Moreover, its actions with neighbor states are:

- Ingoing Actions:
  - From state Q_1: τ with rate μ_2
- Outgoing Actions:
  - To state Q_1: a with rate $\top$ in this case substituted by x_a and a_1 with rate $\top$ in this case substituted by x_{a_1}, total rate $x_a + x_{a_1}$

Then the global balance equation for Q_0 is:

$$\Pi_0(\overbrace{x_a + x_{a_1}}^{out}) = \Pi_1\mu_2$$

$$\frac{\cancel{\Pi_0}}{\cancel{\Pi_0}}(x_a + x_{a_1}) = \frac{\Pi_1}{\Pi_0}\mu_2$$

$$= \frac{\cancel{\rho}^{1}}{\cancel{\rho}^{0}}\mu_2 = \rho\mu_2$$

In conclusion:

$x_a + x_{a_1} = \rho\mu_2$

6.5.3.2 Analysis of Q_n

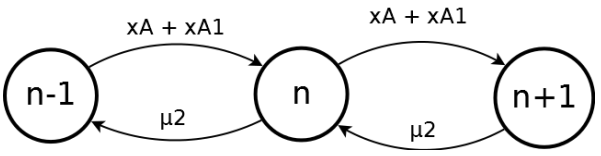

Figure 6.46: State Q_n (DI) and its neighbors states

The general case of process R_2:

$\Pi_n = \alpha\Pi_n = \alpha\rho^n$

In Q_n has $n = n$.

Moreover, its actions with neighbor states are:

- Ingoing Actions:
  - From state Q_{n-1}: a with rate $\top$ in this case substituted by x_a and a_1 with rate $\top$ in this case substituted by x_{a_1}, total rate $x_a + x_{a_1}$
  - From state Q_{n+1}: τ with rate μ_2
- Outgoing Actions:
  - To state Q_{n-1}: τ with rate μ_2
  - To state Q_{n+1}: a with rate $\top$ in this case substituted by x_a and a_1 with rate $\top$ in this case substituted by x_{a_1}, total rate $x_a + x_{a_1}$

Then the global balance equation for Q_n is:

$\Pi_n(x_a + x_{a_1} + \mu_2) = \Pi_{n+1}\mu_2 + \Pi_{n-1}(x_a + x_{a_1})$

$$\begin{aligned}\cancel{\Pi_n}(x_a + x_{a_1} + \mu_2) &= \frac{\Pi_{n+1}}{\Pi_n}\mu_2 + \frac{\Pi_{n-1}}{\Pi_n}(x_a + x_{a_1}) \\ &= \frac{\cancel{\alpha}\rho^{n+1}}{\cancel{\alpha}\rho^{n}}\mu_2 + \frac{\cancel{\alpha}\rho^{n-1}}{\cancel{\alpha}\rho^{n}}(x_a + x_{a_1}) \\ &= \rho\mu_2 + \frac{1}{\rho}(x_a + x_{a_1})\end{aligned}$$

In conclusion:

$$x_a + x_{a_1} + \mu_2 = \rho\mu_2 + \frac{1}{\rho}(x_a + x_{a_1})$$

6.5.3.3 Summary

$$\begin{cases} x_a + x_{a_1} & = & \rho\mu_2 & & & \text{state } Q_0 \\ x_a + x_{a_1} + \mu_2 & = & \rho\mu_2 & + & \dfrac{1}{\rho}(x_a + x_{a_1}) & \text{state } Q_n \end{cases}$$

6.5.3.4 Balancing

First of all, let's try to compare states Q_n and Q_0:

$$\begin{array}{lcll} x_a + x_{a_1} + \mu_2 & = & \rho\mu_2 & + \quad \dfrac{1}{\rho}(x_a + x_{a_1}) \\ x_a + x_{a_1} & = & \rho\mu_2 & \\ \hline \mu_2 & = & & \dfrac{1}{\rho}(x_a + x_{a_1}) \end{array}$$

From this:

$$\rho\mu_2 = \frac{1}{\cancel{\rho}}(x_a + x_{a_1})\cancel{\rho}$$
$$\rho\frac{\cancel{\mu_2}}{\cancel{\mu_2}} = \frac{x_a + x_{a_1}}{\mu_2}$$

$$\rho = \frac{x_a + x_{a_1}}{\mu_2}$$

Now I substitute it in the balance equations:

State Q_0

$$\begin{aligned}
x_a + x_{a_1} &= \rho\mu_2 \\
x_a + x_{a_1} &= \frac{x_a + x_{a_1}}{\cancel{\mu_2}}\cancel{\mu_2} \\
x_a + x_{a_1} &= x_a + x_{a_1}
\end{aligned}$$

State Q_0 is **balanced**.

State Q_n

$$\begin{aligned}
x_a + x_{a_1} + \mu_2 &= \rho\mu_2 + \frac{1}{\rho}(x_a + x_{a_1}) \\
x_a + x_{a_1} + \mu_2 &= \frac{x_a + x_{a_1}}{\cancel{\mu_2}}\cancel{\mu_2} + \frac{1}{\frac{x_a + x_{a_1}}{\mu_2}}(x_a + x_{a_1}) \\
\cancel{x_a} + \cancel{x_{a_1}} + \mu_2 &= \cancel{x_a} + \cancel{x_{a_1}} + \frac{\mu_2}{\cancel{x_a + x_{a_1}}}\cancel{(x_a + x_{a_1})} \\
\mu_2 &= \mu_2
\end{aligned}$$

State Q_n is **balanced**.

6.5.3.5 Steady State Distribution of Process R_2

To find the steady state distribution of process R_2, the last thing to compute is α in such a way that:

$$\left(\sum_{n=0}^{\top} \Pi_n\right) = 1$$

But this is always trivial, using the sum of geometric series, since:

$$\sum_{n=0}^{\top} \alpha\rho^n = 1$$

$$\alpha \sum_{n=0}^{\top} \rho^n = 1$$
$$\alpha \frac{1}{1-\rho} = 1$$
$$\cancel{(1-\rho)}\alpha \frac{1}{\cancel{1-\rho_1}} = 1(1-\rho)$$
$$\alpha = 1 - \rho$$

So in conclusion the steady state distribution of R_2 is the following:
$\Pi_n = \alpha\rho$

Where: $\rho = \dfrac{x_a + x_{a_1}}{\mu_2}$
$\alpha = 1 - \rho$

6.5.4 Analysis and Balancing the Global Balance Equations of R_3

Process R_3 is composed by only two main types of states:

1. State R_0
2. State R_n with $n > 0$

All of its state can be attributed to one of this 2 types.
I suppose that the product form of R_3 will be the following:

$$\Pi_n = \alpha\rho^n$$

Also in this case we have a lattice of phantom states F to coordinate with safe-impossible action of R_1. I will underline the rates related with states F because, like I did for state R_1, I will first analyze the system without the impossible actions.

6.5.4.1 Analysis of R_0

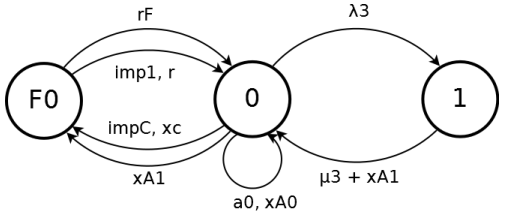

Figure 6.47: State R_0 (DI) and its neighbors states

The general case of process R_3:

$\Pi_n = \alpha\Pi_n = \alpha\rho^n$

In R_0 has $n = 0$.

Moreover, its actions with neighbor states are:

- Ingoing Actions:

  - From state F_0: imp_1 with rate r and $fill$ with rate r_f, total rate $r + r_f$
  - From state R_1: $empty$ with rate μ_3 and a_1 with rate $\top$ in this case substituted by x_{a_1}, total rate $\mu_3 + x_{a_1}$

- Outgoing Actions:
  - To state F_0: a_1 with rate $\top$ in this case substituted by x_{a_1} and imp_c with rate $\top$ in this case substituted by x_c, total rate $x_{a_1} + x_c$
  - To state R_1: $fill$ with rate λ_3

- Self-Loop Actions:
  - From R_0 to itself: a_0 with rate $\top$ in this case substituted by x_{a_0}

Then the global balance equation for R_0 is:

$$\Pi_0(\overbrace{\lambda_3 + x_{a0} + \underline{x_{a1}} + \underline{x_c}}^{out}) = \Pi_0 x_{a0} + \Pi_1(\mu_3 + x_{a1}) + \Pi_{F_0}(\underline{r_f + r})$$

$$\begin{aligned}
\frac{\cancel{\Pi_0}}{\cancel{\Pi_0}}(\lambda_3 + x_{a0} + \underline{x_{a1}} + \underline{x_c}) &= \frac{\cancel{\Pi_0}}{\cancel{\Pi_0}} x_{a0} + \frac{\Pi_1}{\Pi_0}(\mu_3 + x_{a1}) + \frac{\Pi_{F_0}}{\Pi_0}(\underline{r_f + r}) \\
&= x_{a0} + \frac{\cancel{\alpha\rho^1}}{\cancel{\alpha\rho^0}}(\mu_3 + x_{a1}) + \frac{\cancel{\alpha\rho^0}}{\cancel{\alpha\rho^0}}(\underline{r_f + r}) \\
&= x_{a0} + \rho\mu_3 + \rho x_{a1} + \underline{r_f} + \underline{r}
\end{aligned}$$

In conclusion:

$$\lambda_3 + x_{a0} + \underline{x_{a1}} + \underline{x_c} = x_{a0} + \rho\mu_3 + \rho x_{a1} + \underline{r_f} + \underline{r}$$

6.5.4.2 Analysis of R_n

The general case of process R_3:

$$\Pi_n = \alpha\Pi_n = \alpha\rho^n$$

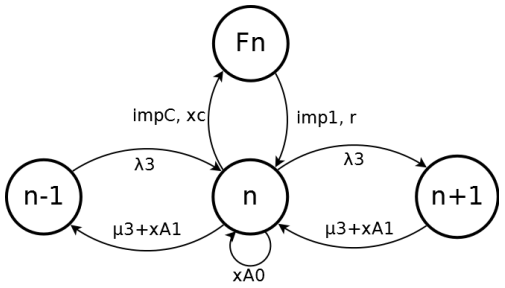

Figure 6.48: State R_n (DI) and its neighbors states

In R_n has $n = n$.

Moreover, its actions with neighbor states are:

- Ingoing Actions:
  - From state F_n: imp_1 with rate r
  - From state R_{n-1}: $fill$ with rate λ_3
  - From state R_{n+1}: $empty$ with rate μ_3 and a_1 with rate $\top$ in this case substituted by x_{a_1}, total rate $\mu_3 + x_{a_1}$
- Outgoing Actions:
  - To state F_n: imp_c with rate $\top$ in this case substituted by x_c
  - To state R_{n-1}: $empty$ with rate μ_3 and a_1 with rate $\top$ in this case substituted by x_{a_1}, total rate $\mu_3 + x_{a_1}$
  - To state R_{n+1}: $fill$ with rate λ_3
- Self-Loop Actions:
  - From R_n to itself: a_0 with rate $\top$ in this case substituted by x_{a_0}

Then the global balance equation for R_n is:

$$\Pi_n(\lambda_3 + \underline{x_{a0}} + x_{a1} + \underline{x_c} + \mu_3) = \Pi_n \underline{x_{a0}} + \Pi_{n+1}(\mu_3 + x_{a1}) + \Pi_{n-1}\lambda_3 + \Pi_{F_n}\underline{r}$$

$$\begin{aligned}\frac{\cancel{\Pi_n}}{\cancel{\Pi_n}}(\lambda_3 + \underline{x_{a0}} + x_{a1} + \underline{x_c} + \mu_3) &= \frac{\cancel{\Pi_n}}{\cancel{\Pi_n}}\underline{x_{a0}} + \frac{\Pi_{n+1}}{\Pi_n}(\mu_3 + x_{a1}) + \frac{\Pi_{n-1}}{\Pi_n}\lambda_3 + \frac{\Pi_{F_n}}{\Pi_n}\underline{r} \\ &= \underline{x_{a0}} + \frac{\cancel{\alpha\rho^1}}{\cancel{\alpha\rho^0}}(\mu_3 + x_{a1}) + \frac{\cancel{\alpha\rho^{n-1}}}{\cancel{\alpha\rho^n}}\lambda_3 + \frac{\cancel{\alpha\rho^0}}{\cancel{\alpha\rho^0}}\underline{r} \\ &= \underline{x_{a0}} + \rho\mu_3 + \rho x_{a1} + \frac{1}{\rho}\lambda_3 + \underline{r}\end{aligned}$$

In conclusion:

$$\lambda_3 + \underline{x_{a0}} + x_{a1} + \underline{x_c} + \mu_3 = \underline{x_{a0}} + \rho\mu_3 + \rho x_{a1} + \frac{1}{\rho}\lambda_3 + \underline{r}$$

6.5.4.3 Summary

As usually, I start without impossible actions:

$$\begin{cases} \lambda_3 + x_{a0} = x_{a0} + \rho\mu_3 + \rho x_{a1} & \text{state } R_0 \\ \lambda_3 + x_{a1} + \mu_3 = \rho\mu_3 + \rho x_{a1} + \frac{1}{\rho}\lambda_3 & \text{state } R_n \end{cases}$$

6.5.4.4 Balancing

First of all, let's try to compare states R_n and R_0:

$$\begin{array}{lcl} \lambda_3 + x_{a1} + \mu_3 = & & \rho\mu_3 + \rho x_{a1} + \frac{1}{\rho}\lambda_3 \\ \lambda_3 + x_{a0} = x_{a0} + & & \rho\mu_3 + \rho x_{a1} \\ \hline \mu_3 + x_{a1} - x_{a0} = -x_{a0} + & & \frac{1}{\rho}\lambda_3 \end{array}$$

From this:
$\mu_3 + x_{a1} - x_{a0} = -x_{a0} + \frac{1}{\rho}\lambda_3$

$$\mu_3 + x_{a1} - \cancel{x_{a0}} = -\cancel{x_{a0}} + \frac{1}{\rho}\lambda_3$$
$$\rho(\mu_3 + x_{a1}) = \frac{1}{\rho}\lambda_3\rho$$
$$\rho\frac{\cancel{\mu_3 + x_{a1}}}{\cancel{\mu_3 + x_{a1}}} = \frac{\lambda_3}{\mu_3 + x_{a1}}$$
$$\rho = \frac{\lambda_3}{\mu_3 + x_{a_1}}$$

Now I substitute it in the following balance equations (with impossible actions):

$$\begin{cases} \lambda_3 + x_{a0} + x_{a1} + x_c = x_{a0} + \rho\mu_3 + \rho x_{a1} + r_f + r & \text{state } 0 \\ \lambda_3 + x_{a0} + x_{a1} + x_c + \mu_3 = x_{a0} + \rho\mu_3 + \rho x_{a1} + \frac{1}{\rho}\lambda_3 + r & \text{state } n \end{cases}$$

State R_0

$$\lambda_3 + x_{a0} + x_{a1} + x_c = x_{a0} + \rho\mu_3 + \rho x_{a1} + r_f + r$$
$$\lambda_3 + \cancel{x_{a0}} + x_{a1} + x_c = \cancel{x_{a0}} + \rho(\mu_3 + x_{a1}) + r_f + r$$
$$\lambda_3 + x_{a1} + x_c = \frac{\lambda_3}{\cancel{\mu_3 + x_{a1}}}\cancel{(\mu_3 + x_{a1})} + r_f + r$$
$$\cancel{\lambda_3} + x_{a1} + x_c = \cancel{\lambda_3} + r_f + r$$
$$x_{a1} + x_c = r_f + r$$

State 0 is **balanced**.

State R_n

$$\lambda_3 + x_{a0} + x_{a1} + x_c + \mu_3 = x_{a0} + \rho\mu_3 + \rho x_{a1} + \frac{1}{\rho}\lambda_3 + r$$
$$\lambda_3 + \cancel{x_{a0}} + x_{a1} + x_c + \mu_3 = \cancel{x_{a0}} + \rho(\mu_3 + x_{a1}) + \frac{1}{\frac{\lambda_3}{\mu_3 + x_{a1}}}\lambda_3 + r$$
$$\lambda_3 + x_{a1} + x_c + \mu_3 = \frac{\lambda_3}{\cancel{\mu_3 + x_{a1}}}\cancel{(\mu_3 + x_{a1})} + \frac{\mu_3 + x_{a1}}{\cancel{\lambda_3}}\cancel{\lambda_3} + r$$
$$\cancel{\lambda_3} + \cancel{x_{a1}} + x_c + \cancel{\mu_3} = \cancel{\lambda_3} + \cancel{\mu_3} + \cancel{x_{a1}} + r$$
$$x_c = r$$

State n is **balanced**.

From this:
$x_c = r$
$x_{a1} + x_c = r_f + r$

So:

$x_{a1} = r_f$

6.5.4.5 Steady State Distribution of Process R_3

To find the steady state distribution of process R_3, the last thing to compute is α in such a way that:

$$\left(\sum_{n=0}^{\top} \Pi_n \right) = 1$$

But this is always trivial, using the sum of geometric series, since:

$$\sum_{n=0}^{\top} \alpha \rho^n = 1$$
$$\alpha \sum_{n=0}^{\top} \rho^n = 1$$
$$\alpha \frac{1}{1-\rho} = 1$$
$$\cancel{(1-\rho)} \alpha \frac{1}{\cancel{1-\rho_1}} = 1(1-\rho)$$
$$\alpha = 1 - \rho$$

So in conclusion the steady state distribution of R_3 is the following:
$\Pi_n = \alpha\rho$

Where: $\rho = \dfrac{\lambda_3}{\mu_3 + x_{a_1}}$
$\alpha = 1 - \rho$

Bibliography

[1] A. Allen. *Probability, Statistics and Queueing Theory with Computer Science Applications*. Academic Press, second edition, 1990.

[2] J.R. Artalejo. *G-networks: A versatile approach for work removal in queueing networks*. European Journal of Operational Research 126, 2000.

[3] J.A. Bergstra and J.W. Klop. *Algebra for Communicating Processes with Abstraction*. Journal of Theoretical Computer Science, 1985.

[4] P. Buchholz. *Hierarchical Markovian Models - Symmetries and Reduction*. Computer performance evaluation, volume x: Modelling techniques and tools edition. R.J. Pooley and J. Hillston.

[5] X. Chao. *A note on queueing networks with signals and random triggering times*. Probability in the Engineering and Informational Sciences 8, 1994.

[6] X. Chao. *A queueing network model with catastrophes and product form solution*. Operations Research Letters 18, 1995.

[7] X. Chao and M. Pinedo. *On generalized networks of queues with positive and negative arrivals*. Probability in the Engineering and Informational Sciences 7, 1993.

[8] Y. Da Li E. Gelenbe, Z. Mao. *Function approximation with spiked random networks*. IEEE Transactions on Neural Networks, 1999.

[9] J.M. Fourneau and E. Gelenbe. *G-networks with multiple classes of signals*. Pergamon Press, New York, 1992.

[10] E. Gelenbe. *Random neural networks with negative and positive signals and product form solution*. Neural Computation 1, 1989.

[11] E. Gelenbe. *Product-form queueing networks with negative and positive customers*. Journal of Applied Probability 28, 1991.

[12] E. Gelenbe. *G-networks with signals and batch removal*. Probability in the Engineering and Informational Sciences, 1993.

[13] E. Gelenbe. *G-networks with triggered customer movement*. Journal of Applied Probability 30, 1993.

[14] E. Gelenbe. *G-networks with triggered customer movement*. Journal of Applied Probability 30, 1993.

[15] E. Gelenbe and A. Labed. *G-networks with multiple classes of signals and positive customers*. European Journal of Operational Research 108, 1998.

[16] E. Gelenbe and R. Schassberger. *Stability of G-networks*. Probability in the Engineering and Informational Sciences 6, 1992.

[17] E. Gelenbe and R. Schassberger. *Stability of G-networks*. Probability in the Engineering and Informational Sciences 6, 1992.

[18] P.G. Harrison and N.M. Patel. *Performance Modelling of Communication Networks and Computer Architectures*. Addison-Wesley, 1992.

[19] P.G. Harrison and E. Pitel. *Response time distributions in tandem G-networks*. Journal of Applied Probability 32, 1995.

[20] W. Henderson. *Queueing networks with negative customers and negative queue lengths*. Journal of Applied Probability 30, 1993.

[21] J. Hillston. *A Compositional Approach to Performance Modelling*. Cambridge University Press, 2005.

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

[23] E. Gelenbe J.M. Fourneau and R. Suros. *G-networks with multiple classes of negative and positive customers*. Theoretical Computer Science 155, 1996.

[24] E. Gelenbe J.M. Fourneau and R. Suros. *G-networks with multiple classes of negative and positive customers*. Theoretical Computer Science 155, 1996.

[25] P.J.B. King. *Computer and Communication Systems Performance Modelling*. Prentice Hall, 1990.

[26] L. Kleinrock. *Queueing Systems, Volume I: Theory*. John Wiley, New York, 1975.

[27] L. Kleinrock. *Queueing Systems, Volume II: Computer Applications*. John Wiley, New York, 1976.

[28] K. Larsen and A. Skou. *Bisimulation through Probabilistic Testing*. Information and Computation, 1991.

[29] R. Milner. *Communication and Concurrency*. Prentice-Hall, 1989.

[30] B.S. Northcote W. Henderson and P.G. Taylor. *State-dependent signalling in queueing networks*. Advances in Applied Probability 26, 1994.