Towards compositional CSL model checking

Paolo Ballarini

Doctor of Philosophy

Dipartimento di Informatica

Università di Torino

2004

Abstract

The Continuous Stochastic Logic (CSL) is a powerful means to state properties which refer to Continuous Time Markov Chains (CTMCs). The verification of such properties on a model can be achieved through a suitable algorithm. In this doctoral thesis, the CSL logic has been considered and two major aspects have been addressed: the analysis of its expressiveness and the study of methods for a decomposed verification of formuale. Concerning expressiveness, we have observed that the CSL syntax can lead to formulae with a trivial semantics. As a consequence the idea of *well-formed* CSL formula has been introduced. Furthermore, a simpler and equivalent syntax for referring to ergodic CMTCs have defined as well as a brand new *event-bounded* Until operator. With respect to compositionality, we have referred our study to a specific type of decomposed CTMCs, namely the bidimensional Boucherie framework. A number of basic properties concerning the Boucherie framework have been demonstrated and, relying on this, a compositional semantics for a subset of the CSL syntax has been derived. The considered subset is obtained by disallowing nesting of probabilistic path-formulae, something whose impact on the ability to state useful properties is low.

Acknowledgements

In July 2000, after a period in Munich (Germany), I moved back to Italy to start on my PhD at *Università di Torino*. In Torino I have been a member of the Modelling and Analysis of Computer Systems (MACS) group at *Dipartimento di Informatica*, a group lead by Professor Gianfranco Balbo. My research work there has been supervised by Susanna Donatelli in cooperation with Giuliana Franceschinis. During the first year of my PhD I attended several classes and summer schools some of which were introductory courses on Model Checking. I found that subject very interesting and decided it would be the focus of my research. By the end of the second year of my PhD I moved to United Kingdom to work with Jane Hillston at Laboratory for Foundations of Computer Science (LFCS), University of Edinburgh. Me and Jane agreed on studying a compositional approach to Model Checking techniques for Continuous Time Markov Chains and that became the objective of our research.

This thesis has been written, under the supervision of Jane Hillston, while I was in Edinburgh. I am very pleased I had the opportunity to work with Jane, which is an excellent guide who taught me a lot. I would like to thank Jane for the time she devoted to me and for helping me in finding the necessary motivation to write this thesis. Her knowledge, her precious advice and her "being so picky", have been fundamental to the outcomes of this work. I would also like to thank Susanna Donatelli, who strongly encouraged and supported the cooperation with Jane to take place and for her useful and stimulating comments which made this research work better. Being in Edinburgh for almost two years has given me the chance to improve my English and to be part of a very dynamic, international community, something I am really glad to have had the chance to experience.

I am very grateful to Colin Stirling, the head of LFCS, for supporting my presence as a member of the lab during the last two years. LFCS is an excellent research environment where one can meet very skillful people.

I also wish to thank a number of persons, both in Edinburgh and in Torino, with whom I had the opportunity to get involved in various technical discussions during the years of my doctorate. The patience and interest they showed in answering/discussing my points have been very important to me and helped a lot with keeping my motivation high. Among them are Kousha Etessami, Stephen Gilmore and Leila Kloul, from LFCS, as well as András Horváth and Jeremy Sproston, from MACS group in Turin.

A final thank goes to my family for their endless love.

Paolo Ballarini - Edinburgh, January 2004

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Paolo Ballarini*)

Table of Contents

Chapter 1

Introduction

The continual advances in technology, result in the development of ever more complex systems. Formal modelling is conceived as a means to help in optimising and guiding the design phase. This is achieved by construction of a model of the system, whose analysis allows for the study of the system's behaviour. Different kinds of analysis can be performed on a model referring to different aspects of the system's behaviour. For example, with respect to computer systems, one could be interested in the evaluation of performance characteristics like, the benefit of increasing the number of CPUs in a multiprocessor system or the impact of a scheduling algorithm on the CPU's throughput, or in verifying qualitative properties, like, checking that a mutual exclusion protocol is deadlock-free. Dependability studies are also amongst the relevant types of analysis which one would like to perform on a system's model. With this respect, indices like the mean time to failure of a system's component can be assessed when the system's reliability is of interest.

A model is an abstract representation of the system. Often the system's behaviour can be described in terms of the states it can occupy and by specifying how it can move from one state to another in time. This type of models are referred to as discrete-event state-based models and this work refers to them. The model's dimension depends on the system's complexity and on the details it captures. Clearly complex systems can easily result in very large models which turn out to be intractable. In the literature, this is often referred to as the *state-space explosion* problem, a well known issue the treatment of which is of major interest in research. *Compositionality* is seen as a means to tackle the *state-space explosion* problem. In substance, *compositionality* is a modelling strategy which is meant to help both in modelling and analysing complex systems. The basic idea is to obtain a decomposed representation of the model, in terms of a number of (smaller) sub-models. The model of interest (composed model) can be retrieved by the application of a compositional rule to the sub-models. The purpose of a decomposed analysis technique is to draw information regarding the performance and/or reliability of the composed model, by combining the results of the analysis of the component models (sub-models). This allows huge, hence intractable, models to be studied by means of smaller, tractable, sub-models.

When the future evolution of a system depends only on its current state, the system can be represented by means of a Markov process (Markov chains when the state-space is discrete). The most common type of analysis for Markov chains concerns the evaluation of the probability of being in a given state either *in the long run* (i.e. at infinity) or at a given time instant t. When time is considered as an enumerable quantity, then we refer to Discrete Time Markov Chains (DTMCs). On the other hand if time is considered continuous, we refer to Continuous Time Markov Chains (CTMCs). CTMCs have become a very popular/ widely used formalism for modelling purpose in many diverse areas, not only in computer science. One possible type of analysis of CMTCs is given by model-checking. Generally speaking, model-checking is a technique which permits the verification of properties against a given model. Properties are given in terms of formulae of some temporal logic, while a model, in essence, is expressed as a graph whose paths represent the possible evolutions of the system. An algorithm (*model-checker*) is then supplied with both the system's model and the formula of interest and returns either a positive answer, if the model fulfils the property represented by the formula, or a counterexample (a system evolution) which contradicts the formula. Different types of model-checking have been defined in the last decades, referring to different types of systems and featuring different types of expressivity. The model checking for CTMCs takes its name from the temporal logic it is based on, which is the Continuous Stochastic Logic. Hence it is referred to as CSL model-checking.

With respect to model-checking, compositionality regards the study of decomposed equivalences. If a certain property is to be verified with respect to a decomposed model, it is of interest to investigate whether this is equivalent to checking a number of *derived* formulae with respect to the sub-models. By means of such an approach the verification of a large model can be replaced by the verification of smaller models. In the model-checking literature few works regarding the study of a compositional approach can be found, and, to the best of our knowledge, none at all with respect to CSL model-checking.

In this work CSL model-checking is considered and two major contributions are presented. The first one concerns a study of the CSL expressiveness. Some syntactical bounds are identified in order to characterise sensible CSL formulae. Moreover a simplified syntax is introduced for referring to a subclass of CMTCs. Secondly, a compositional way to check CSL formulae is studied. A decomposed approach for CSL model checking, referring to a specific compositional framework for CTMCs, namely the Boucherie framework, is derived. This result is based on proving a number of equivalences which show that the verification of a certain CSL formula with respect to a bidimensional Boucherie process, corresponds to the verification of a number of derived formulae with respect to the component's processes.

The remainder of the thesis is organised as follows.

In Chapter 2 some background material and definitions are presented. An overview of the principal types of temporal logics and of the corresponding model checking methods is provided. The CSL syntax and semantics are thoroughly described as well as the existing algorithms for checking its formulae against a given CTMC. The Boucherie compositional framework for CMTCs is then introduced together with an example (running example) which will be used throughout the other chapters in order to show the correctness of the derived semantic equivalences.

The syntax and semantics of the logic CSL are meant to refer to arbitrary CMTCs. However, in Chapter 3, it will be shown that when dealing with ergodic CTMCs (CTMCs which correspond to a strongly connected graph), a simpler and equivalent syntax can replace the original one. Moreover *well-formed* CSL formulae are characterised and proved to be the only sensible type of formulae which one would be interested in. In this chapter we also introduce a new, *event-bounded* version of the Until operator for which a verification method is defined.

In Chapter 4, the bidimensional Boucherie framework is considered in more detail and some basic definitions and properties are provided. This constitutes the basis on which the decomposed semantics relies. A partition of the CSL formulae which refer to a bidimensional Boucherie process is characterised: *single component* formulae are those which refer to features of one component's process only; *general* formulae, instead, involve both components. A compositional semantics for both *single-component* and *general* non-path formulae (i.e. formulae which involve neither the Next nor the Until connective) is then proved.

In Chapter 5 Next formulae referring to a bidimensional Boucherie process are considered and a compositional semantics is derived for both the *single-component* and *general* case. It will be proved that checking a "simple" *single-component* bounded Next for a probability bound p with respect to a bidimensional Boucherie process is equivalent, in the *worst-case*, to checking the same formula, for a derived probability bound p' with respect to the component it refers to, or, in the *best-case*, to verifying a simple inequality. Relying on this, it will also be proved, that the verification of the *steady-state* probability of a "simple" *single-component* bounded Next formula against a bound p is equivalent to the verification of the *steady-state* probability of a derived formula with respect to a derived probability bound p' on the component the original formula refer to. Finally, for the *general* case, we will show that, checking a *general* bounded Next formula against the Boucherie process, boils down to checking n derived *single-component* bounded Next formulae for each of the two component's process.

In Chapter 6, the characterisation of a decomposed semantics for Until formulae which refer to a bidimensional Boucherie process is addressed. The relationship between paths of the composed process (*bidimensional paths*) and of the components' processes are assessed and some basic properties are demonstrated. Essentially each path of the product process is given by interleaving of a pair of paths on the components (*projection paths*). It will be shown that the probability measure of a *bidimensional path* can be factored in terms of the probability measure of its *projection paths*. This allows for proving that checking a *single-component* Until formula for a bound p against the product-process is equivalent to checking the same formula for a derived bound p' and against the component process it refers to.

Finally, in Chapter 7, a summary of the results of this work is presented, together with an analysis of directions for future work.

The thesis also includes two appendices. Appendix A contains the formal proof of some lemmas and propositions on which the compositional semantics of Next formulae (Chapter 5) relies. Appendix B, instead, contains a number of definitions and propositions concerning the properties of *bidimensional paths*. The results there proved are those on which the compositional semantics for Until formulae (Chapter 6) is based.

Chapter 2

Background

2.1 Introduction

This chapter contains the background material for this thesis. A compact survey of the model-checking methodologies, ranging from the non-probabilistic framework to the probabilistic one, is provided in the next section. Section 2.3 describes thoroughly the model-checking technique for the Continuous-Time Markov Chains, which is the class of stochastic models we are concerned about in this work. Section 2.4 provides an overview of the idea of compositionality in general, in performance modelling and in particular in the model-checking framework. Since this work focuses on the study of a compositional approach for model-checking of Continuous-Time Markov Chains we need to consider a compositional framework for that type of stochastic process. Section 2.5 is devoted to describing the Boucherie product-form framework, a compositional method for CTMCs featuring a very useful (de)compositional expression for the steady-state distribution of the composed Markov Chain.

2.2 Model-Checking: a survey

Model checking is a methodology for testing a system's model against properties expressed in terms of some temporal logic formulae. A *model checker* is an algorithm (a program) which takes a model/system $\mathcal{M}$ and a formula ϕ as inputs (Figure 2.1) and returns either YES if ϕ is satisfied in $\mathcal{M}$ or NO if it is not, providing, in such a case, a counter-example of the checked property.

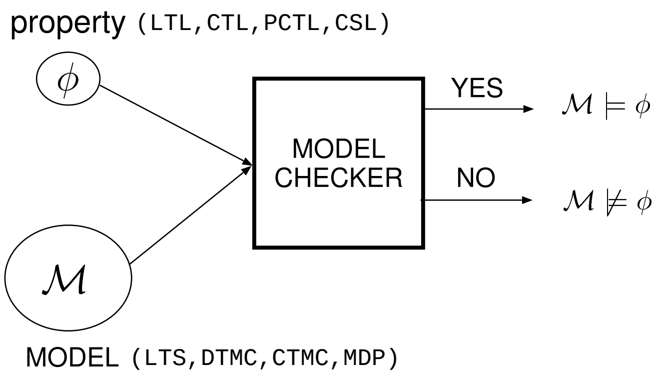


Figure 2.1: Model checking a formula ϕ against a model $\mathcal{M}$

The existing model-checking techniques may be classified with respect to the type of model they refer to. In this sense we can distinguish between two large classes of models:

- **non-probabilistic models**: referring to systems whose behaviour can be *deterministically* determined.
- **probabilistic models**: referring to systems whose behaviour can be *stochastically* determined.

Non-probabilistic systems are modelled in terms of *Labelled Transition Systems* (LTS) (i.e. labelled graphs).

Definition 2.2.1 (Labelled Transition Systems (LTS)) *A Labelled Transition System is a tuple $\mathcal{M} = (S, \mathbf{R}, L)$ where*

- *S set of states*
- *$\mathbf{R} \subseteq S \times S$ set of arcs*
- *$L : S \to 2^{AP}$ labelling function*

where AP is a set of Atomic Propositions.

Given a state s of a LTS $\mathcal{M} = (S, \mathbf{R}, L)$, we will denote at_s the conjunction of all atomic propositions with which s is labelled. In essence:

$$at_s = \bigwedge_{a_i \in L(s)} a_i$$

As we will see, at_s uniquely identifies the state s it refers to (i.e. the formula at_s is valid only in s).

A *path* from a given LTS $\mathcal{M}$ represents a possible execution of the system modelled by $\mathcal{M}$. Formally,

Definition 2.2.2 (Path over a LTS) *Let $\mathcal{M} = (S, \mathbf{R}, L)$ be a LTS. An infinite* path σ *is a sequence of states $s_0 \rightarrow s_1 \rightarrow s_2 \ldots$ such that for any $i \in \mathbb{N}$, $(s_i, s_{i+1}) \in \mathbf{R}$.*

Remark 2.2.1 *Let $\mathcal{M} = (S, \mathbf{R}, L)$ be a LTS. A finite* path σ *denotes the set of all infinite paths of $\mathcal{M}$ whose common* prefix *is* σ.

Given a model $\mathcal{M}$ and a state s we will assume the following notations concerning paths:

- $Path^{\mathcal{M}}()$ is the set of all possible paths of $\mathcal{M}$.
- $Path^{\mathcal{M}}(s)$ is the set of all possible paths of $\mathcal{M}$ starting at s.

Wherever it is unambiguous the superscript $\mathcal{M}$ is omitted from the above notations.

2.2.1 Linear Time Logic model-checking

The Linear Time Logic (LTL) [38] is a language to reason about the future considering time as extending in a linear fashion. Models are LTS and formulae are evaluated with respect to paths extracted from the LTS. Two basic temporal connectives allow one to refer to the future: the *next* operator, denoted $(\mathcal{X})$ and the *until* operator, denoted $(\mathcal{U})$.

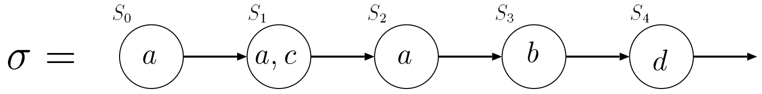
The complete syntax for the formulae ϕ of the LTL is as follows:

$$\phi := a \mid tt \mid \neg\phi \mid \ \phi \wedge \phi \mid X\ \phi \mid \phi\ \mathcal{U}\ \phi \tag{2.2.1}$$

where $a \in AP$ and AP being the set of atomic propositions of the considered LTS.

The semantics of LTL is defined in terms of a relation, denoted by $\models$, which associates paths $\sigma \in Path^{\mathcal{M}}$ and LTL formulae

$$\begin{array}{ll}
\sigma \models tt \ \textbf{ forall } \ \sigma \in Path^{\mathcal{M}} & \sigma \models \phi' \wedge \phi'' \ \textbf{ iff } \ \sigma \models \phi' \wedge \sigma \models \phi'' \\
\sigma \models a \ \textbf{ iff } \ a \in L(\sigma[0]) & \sigma \models X\ \phi \ \textbf{ iff } \ \sigma[1] \models \phi \\
\sigma \models \neg\phi \ \textbf{ iff } \ \sigma \not\models \phi & \sigma \models \phi' \mathcal{U} \phi'' \ \textbf{ iff } \ \exists i \geq 0 : \sigma[i] \models \phi'' \wedge \forall j < i, \sigma[j] \models \phi'
\end{array}$$



$i) \quad \sigma \models a\,\mathcal{U}\ b \qquad iii) \ \sigma \models \mathcal{X}\ c$

$ii) \quad \sigma \models b\,\mathcal{U}\ a \qquad iv) \quad \sigma \not\models a\,\mathcal{U}\ d$

Figure 2.2: Semantics of *next* and *until* formulae

Example 2.2.1 *Figure 2.2 provides an example of the semantics of* next *and* until *formulae. Case $i)$ shows an example of an* until *formula $\phi \equiv a\mathcal{U}b$ which is satisfied by the path σ. In fact σ is such that there exists a future state (s_3) on which b is true and, furthermore, a is satisfied in all the predecessors of s_3 (i.e. s_0, s_1 and s_2). Similarly case $ii)$ ($\phi \equiv b\mathcal{U}\ a$) is meant to show that an* until *formula is satisfied in any path σ whose initial state $\sigma[0]$ satisfies the* target *argument (i.e. a in $\phi \equiv b\ \mathcal{U}\ a$) of the* until, *independently by the satisfiability of the first argument (b). Case $iii)$ exemplifies the* next *semantics: $\phi \equiv Xc$ is satisfied by σ as the* next *argument (c) is satisfied on the successor of σ initial state ($\sigma[1] \models c$). Finally case $iv)$ shows that σ does not satisfy $\phi \equiv a\mathcal{U}\ d$. In fact, even if there is a future state satisfying the* target *d ($s_4 \models d$), it is not the case that for all its predecessors the first argument (a) is satisfied (indeed $s_3 \not\models a$).*

From now on given an until formula $(\phi'\ \mathcal{U}\ \phi'')$ we will refer to the arguments of the until operator as the *premise*, ϕ' and the *target*, ϕ''.

Remark 2.2.2 *The set of propositional and temporal connectives described by 2.2.1 is* adequate*. (i.e. all the others propositional and temporal connectives can be derived from them).*

$\vee$: disjunction	$\phi' \vee \phi'' \equiv \neg(\neg\phi' \wedge \neg\phi'')$
$\vee$: implication	$\phi' \rightarrow \phi'' \equiv \neg\phi' \vee \phi''$
$\Diamond$: sometime in the future	$\diamond\phi \equiv tt\,\mathcal{U}\phi$
$\Box$: always in the future	$\Box\phi \equiv \neg\Diamond\neg\phi$

2.2.2 Computational Tree Logic model-checking

The Computational Tree Logic (CTL) [12] is a temporal logic which allows one to deal with non-deterministic behaviour. While in LTL the time is seen as evolving in a single linear way (hence formulae are evaluated with respect to single paths), in *branching time logic* the model of time is a tree-like structure in which the future is not determined: there can be many different paths in the future, any one of which might be the actual one. The ability to consider non-determinism in the future behaviour of a system is syntactically achieved by the introduction of two *path quantifiers*, E (existential) and A (universal), which are coupled with the temporal connectives *next* and *until*.

The syntax of CTL formulae is the following:

$$\phi := a \mid tt \mid \neg\phi \mid \ \phi \wedge \phi \mid EX\ \phi \mid AX\ \phi \mid E(\phi\ \mathcal{U}\ \phi) \mid A(\phi\ \mathcal{U}\ \phi) \tag{2.2.2}$$

Intuitively, the semantics of a *path quantified* CTL formula $E(\varphi)$ or $A(\varphi)$, where $\varphi ::= \mathcal{X}\phi \ \mid\ \phi'\,\mathcal{U}\phi''$, is the following:

- $E(\varphi)$ (existential) is satisfied in a state s if and only if there exists at least one path σ starting from s which satisfies φ.
- $A(\varphi)$ (universal) is satisfied in a state s if and only if any path σ starting from s satisfies φ.

Formally the CTL semantics requires formulae to be evaluated with respect to states and not to paths. The satisfiability relationship $\models$ for CTL is a direct consequence of the LTL one, except for the (path quantified) *next* and *until* formulae. Hence let $\mathcal{M} = (S, \mathbf{R}, L)$ be a LTS and $s \in S$ a state, $\models \subseteq S \times \phi$ is defined as follows:

$$
\begin{array}{ll}
s \models tt \ \textbf{ forall } \ s \in S & s \models \phi' \wedge \phi'' \ \textbf{ iff } \ s \models \phi' \wedge s \models \phi'' \\
s \models a \ \textbf{ iff } \ a \in L(s) & s \models \neg\phi \ \textbf{ iff } \ s \not\models \phi \\
s \models EX(\phi) \ \textbf{ iff } \ \exists \sigma \in Path(s) : \sigma[1] \models \phi & s \models AX(\phi) \ \textbf{ iff } \ \forall \sigma \in Path(s) : \sigma[1] \models \phi \\
s \models E(\phi' \mathcal{U} \phi'') \ \textbf{ iff } \ \exists \sigma \in Path(s) : & s \models A(\phi' \mathcal{U} \phi'') \ \textbf{ iff } \ \forall \sigma \in Path(s) \\
\quad \exists i \geq 0 : \sigma[i] \models \phi'' \wedge \forall j < i, \sigma[j] \models \phi' & \quad \exists i \geq 0 : \sigma[i] \models \phi'' \wedge \forall j < i, \sigma[j] \models \phi'
\end{array}
$$

Example 2.2.2 *Figure 2.3 depicts examples showing the semantics of the* path quantified next *and* until *formulae of CTL. In case* $i)$ *we have that the state s satisfies the existentially quantified* until *formula* $a\mathcal{U}\ b$*. In fact the tree-like structure shows that there exists at least one path from s satisfying* $a\mathcal{U}b$*. On the other hand in case* $ii)$ *we have that s satisfies the universally quantified* until *formula* $\phi \equiv A(a\mathcal{U}\ b)$ *since all the paths starting at s satisfy* $a\mathcal{U}\ b$*. Finally cases* $iii)$ *and* $iv)$ *provide similar examples though referring to path quantified* next *formulae.*

2.2.3 Probabilistic Computational Tree Logic model-checking

The Probabilistic Computational Tree Logic (PCTL) [19] provides means for verification of quantitative properties, like time deadlines on real-time systems.

While both LTL and CTL refer to an *untimed* type of models (LTS), hence focusing on verification of correctness and qualitative analysis of the modelled system, PCTL relates to probabilistic models which incorporate timing information. In particular PCTL refers to Discrete-Time Markov Chains (DTMC). Markov process analysis techniques allow for computation of typical overall average performance measures, such as throughput of a certain activity or average response time of a given service. PCTL improves the analysis capability of DTMC by introducing the possibility for verification of *soft deadlines* properties of the type: "the probability of a service S to

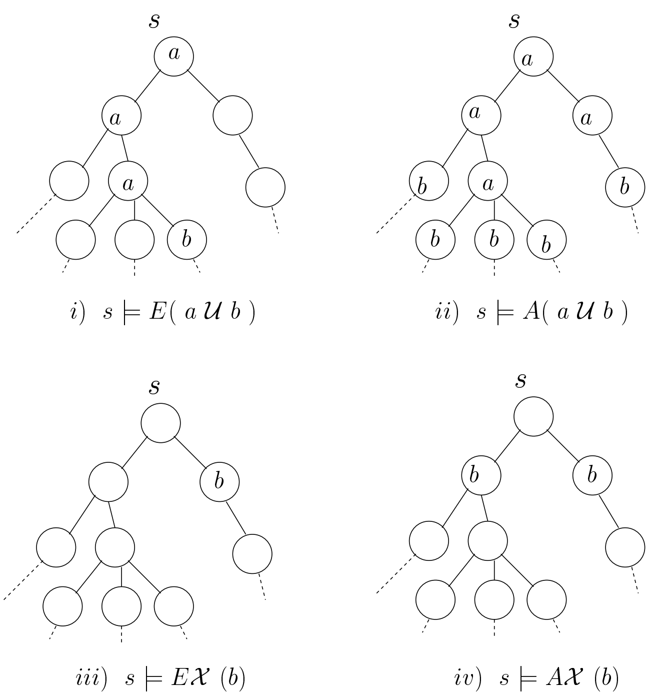


Figure 2.3: Semantics of *path quantified* CTL formulae

be carried out within 2 seconds is at least 98 percent". In this sense PCTL can be seen as a logic for stating and verifying soft deadlines.

The PCTL syntax is defined in the following way:

$$\phi := a \mid tt \mid \neg\phi \mid \ \phi \wedge \phi \mid \mathcal{P}_{\trianglelefteq p}(\varphi) \qquad \text{(state-formulae)} \qquad (2.2.3)$$

$$\varphi := \phi\,\mathcal{U}^{\leq t}\phi \qquad \text{(path-formulae)} \qquad (2.2.4)$$

where $a \in AP$, $p \in [0,1]$, $t \in \mathbb{N}^* \cup \{\infty\}$ and $\trianglelefteq \in \{\geq, >, \leq, <\}$ (i.e. a is an atomic proposition and t is any positive integer or ∞).

Probabilistic CTL is a *branching time logic*, thus formulae are evaluated with respect to a single *source* state by considering all the possible evolutions of the system starting from that state. The *existential* and *universal* path quantifiers of CTL are replaced by a single *continuous* path quantifier, namely $\mathcal{P}_{\trianglelefteq p}$. Intuitively a *path quantified* formula $\mathcal{P}_{\trianglelefteq p}(\varphi)$ is satisfied in a state s if and only if the *probability measure* of the paths σ

validating φ is $\trianglelefteq p$.

Definition 2.2.3 *A labelled DTMC $\mathcal{M}$ is a tuple $(S, \mathbf{P}, L)$ with*

- *S finite set of states*
- *$\mathbf{P} : S \times S \rightarrow [0,1]$ is the* transition probability *matrix and it is such that*

$$\sum_{s' \in S} \mathbf{P}(s, s') = 1 \quad \forall s \in S$$

- *$L : S \rightarrow 2^{AP}$ is the labelling function.*

From a practical point of view DTMC are directed graphs with an arc between each pair of states (s, s') whose correspondent transition probability is greater than zero: $\mathbf{P}(s, s') > 0$. Each arc $s \rightarrow s'$ is labelled with the value $\mathbf{P}(s, s')$ while each node $s \in S$ is labelled with $L(s)$. The value $\mathbf{P}(s, s')$ represents the probability of the transition from s to s' to take place in one *time unit* given that s is the current state.

Definition 2.2.4 (path in a DMTC) *Let $\mathcal{M} = (S, \mathbf{P}, L)$ be a DTMC, a path σ from state s_0 is an infinite sequence*

$$s_0 \rightarrow s_1 \rightarrow \ldots \rightarrow s_n \rightarrow \ldots$$

such that $\forall i \in \mathbb{N}, \mathbf{P}(s_i, s_{i+1}) > 0$

Given a path σ, $\sigma[k]$ denotes the k-th element of σ.

Remark 2.2.3 (n-th prefix of a path) *Given a path σ from a DTMC $\mathcal{M}$, $\sigma \uparrow n$ denotes its n-th prefix:*

$$\sigma \uparrow n = s_0 \rightarrow s_1 \rightarrow \ldots \rightarrow s_n$$

where $\sigma = s_0 \rightarrow s_1 \rightarrow \ldots \rightarrow s_n \rightarrow s_{n+1} \rightarrow \ldots$

Remark 2.2.4 (n-th suffix of a path) *Given a path σ from a DTMC $\mathcal{M}$, $n \uparrow \sigma$ denotes its n-th suffix:*

$$n \uparrow \sigma = s_n \rightarrow s_{n+1} \rightarrow \ldots$$

where $\sigma = s_0 \rightarrow s_1 \rightarrow \ldots \rightarrow s_n \rightarrow s_{n+1} \rightarrow \ldots$

Definition 2.2.5 (*probability measure* of a path) *Let $\mathcal{M} = (S, \mathbf{P}, L)$ be a DTMC then the probability measure of the set of (infinite) paths σ with $\sigma \uparrow n = s_0 \to s_1 \to \ldots \to s_n$ is given by the product*

$$Prob(\sigma \uparrow n) = \prod_{i=0}^{n-1} \mathbf{P}(s_i, s_{i+1})$$

We introduce the following notation concerning sets of paths:

- $Prob(s)$: probability measure of all paths σ starting at state s. (i.e. $Prob(s) = \sum_{\sigma \in Path(s)} Prob(\sigma)$).
- $Prob(s, \varphi)$: probability measure of all paths σ starting at state s satisfying the path formula φ.

The PCTL semantics of state-formulae is exactly the same as the CTL ones but for the *path quantified* formulae, for which it is given by:

$$s \models \mathcal{P}_{\unlhd p}(\varphi) \;\; \textbf{iff} \;\; Prob(s, \varphi) \unlhd p$$

While the *bounded* formulae, defined with respect to a given path σ is given by:

$$\sigma \models \phi' \, \mathcal{U}^{\leq t} \phi'' \;\; \textbf{iff} \;\; \exists i \leq t : \sigma[i] \models \phi'' \wedge \forall j < i, \sigma[j] \models \phi'$$

Example 2.2.3 *Figure 2.2.3 points out examples of the semantics of PCTL* continuously *path quantified formulae. On the left hand side (case i) we have that the probability measure of the paths from $Path(s, (a \; \mathcal{U} \; b))$ is somewhere within the bound of* 0.8 *which we are interested to verify (we pinpoint in bold the subset of paths $Path(s, (a \; \mathcal{U} \; b))$: clearly the probability measure of the whole set $Path(s)$ is equal to* 1 *for any state s).*

On the other hand, the right hand side of Figure 2.2.3 shows the case where $Path(s, a \; \mathcal{U} \; b) = Path(s)$ (i.e. any path out of s is a path satisfying $(a \; \mathcal{U} \; b)$). Clearly the probability measure of $Path(s, a \; \mathcal{U} \; b)$, in such a case, is equal to 1*, hence the* *formula* $\mathcal{P}_{\geq 1}(a\ \mathcal{U}\ b)$ *is satisfied in s. This example is also meant to highlight the fact that PCTL enriches CTL expressiveness with respect to path quantification: indeed* $\mathcal{P}_{\geq 1}(a\ \mathcal{U}\ b)$ *is equivalent to the universally quantified CSL formula* $A(a\ \mathcal{U}\ b)$*, while* $\mathcal{P}_{\geq 0}(a\ \mathcal{U}\ b)$ *is equivalent to the* existentially *quantified CSL formula* $E(a\ \mathcal{U}\ b)$.

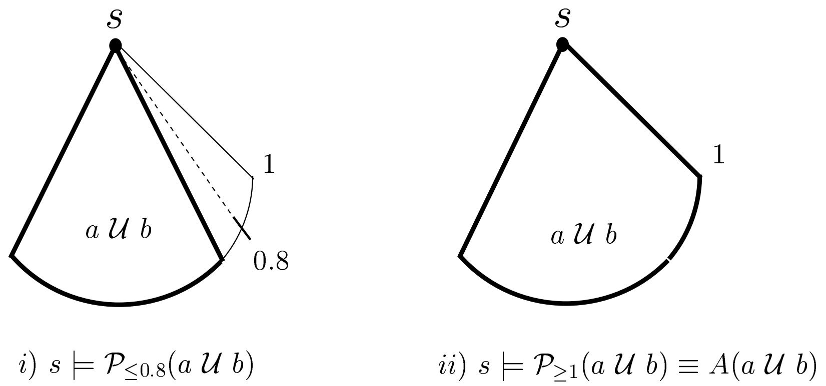


Figure 2.4: Continuously quantified path formulae in PCTL

2.3 Model-Checking for Continuous-Time Markov Chains

In this work we are concerned with model-checking techniques for verification of CTMCs. In [1] Aziz *et al* have introduced a temporal logic, named Continuous Stochastic Logic (CSL), for expressing properties of systems modelled in terms of CTMCs. An algorithm for checking such properties has also been provided. The original definition has then been extended by Katoen *et al* in [9, 4, 5] and revised algorithms have been defined.

The CSL, basically, enriches the standard analysis capability of CTMCs (i.e. steady-state and transient-state analysis) with the ability of specifying possible evolutions (i.e. paths) amongst the factors characterising the states of interest (these properties are usually referred to as *path properties*). Statements like ”the probability for a service *S* to be carried out within time *t* is at least 98 percent”, where *t* is a generic *positive real* time instant (i.e. $t \in \mathbb{R}_{\geq 0}$), can be verified through the CSL model checking algorithm.

Amongst other things, CSL model checking strongly relies on the definition of the *probability measure* of a CMTC's path. Thus before going through the description of the syntax and semantics of the CSL formulae we need to provide some formal definitions which will be the basis for the characterisation of such measures. The definitions, theorems and algorithms included in the remainder of this section are taken from [5].

Basically, CTMCs differ from DTMCs in that time is considered as a continuous quantity whereas in the discrete-time framework time is seen as an infinite but enumerable set of instants. Formally a labelled CTMC is defined as:

Definition 2.3.1 (labelled CTMC) *A labelled CTMC $\mathcal{M}$ is a tuple $(S, \mathbf{Q}, L)$ with*

- *S finite set of states*
- *$\mathbf{Q} : S \times S \rightarrow \mathbb{R}_{\geq 0}$ is the rate matrix*
- *$L :\rightarrow 2^{AP}$ is the labelling function.*

where $\mathbf{Q}(s,s) = \sum_{s' \neq s} \mathbf{Q}(s,s')$.

The binary relation $\mathbf{Q}$ is expressed in terms of a matrix namely, the *infinitesimal generator matrix*. The *transition rate* $\mathbf{Q}(s,s') > 0$ if and only if there is a transition from s to s'. Furthermore, as a consequence of the *memoryless property* of *Markov processes* (see for example [42],[17]), the probability that the transition $s \rightarrow s'$ takes place within t time units (i.e. within the closed interval $[0,t]$) is given by $1 - e^{\mathbf{Q}(s,s')t}$, meaning that the delay of a transition $s \rightarrow s'$ is governed by an exponential distribution whose parameter is the *transition rate* $\mathbf{Q}(s,s')$.

Any state s such that $\mathbf{Q}(s,s') = 0$ for all $s' \in S$ is called *absorbing*. The sum of the outgoing *transition rates* from a state s is called the *total rate* or the *emanating rate* of s and it is denoted by $E(s) = \sum_{s' \neq s} \mathbf{Q}(s,s')$ (clearly the *emanating rate* of an *absorbing* state is zero). Whenever $\mathbf{Q}(s,s') > 0$ for more than one state s', then a competition between different transitions from s exists (*race condition*). In such a case the probability that a transition from s to s' ($s \neq s'$) occurs within t time units is given by

$$\mathbf{P}(s,s',t) = \frac{\mathbf{Q}(s,s')}{E(s)} \cdot \left(1 - e^{-E(s) \cdot t}\right)$$

The above result relies on the fact that the minimum of n exponentially distributed random variables is an exponentially distributed random variable whose parameter is equal to the sum of the parameters $\lambda = \sum_{i=1}^{n} \lambda_i$. Hence $1 - e^{-E(s) \cdot t}$ is the probability for a transition out of s to occur with time t, while $\mathbf{P}(s,s') = \frac{\mathbf{Q}(s,s')}{E(s)}$ is the probability that the delay of going from s to s' finishes before the delay of any other transition from state s.

Embedded discrete-time Markov Chain: given a CTMC $\mathcal{M} = (S, \mathbf{Q}, L)$, the matrix $\mathbf{P}$ is known as the *transition matrix* of the *embedded* discrete-time Markov chain of $\mathcal{M}$ which we denote as $\overline{\mathcal{M}} = (S, \mathbf{P}, L)$.

Embedded Labelled Transition System: given a CTMC $\mathcal{M} = (S, \mathbf{Q}, L)$, we can consider the *embedded* LTS of $\mathcal{M}$ which is defined as $\hat{\mathcal{M}} = (S, \mathbf{R}, L)$, where for all $s, s' \in S$, $(s,s') \in \mathbf{R} \iff \mathbf{Q}(s,s') > 0$.

Definition 2.3.2 (initial-state distribution) *Let $\mathcal{M}$ be a* Markov chain *with state-space S, a function $\alpha : S \to [0,1]$ is an initial-state distribution for $\mathcal{M}$, given that* $\sum_{s \in S} \alpha(s) = 1$.

Definition 2.3.3 (Path of a CTMC) *Let $\mathcal{M} = (S, \mathbf{Q}, L)$ be a labelled CTMC. An infinite path σ on $\mathcal{M}$ is an infinite sequence*

$$s_0 \xrightarrow{t_0} s_1 \xrightarrow{t_1} s_2 \ldots \xrightarrow{t_{n-1}} s_n \xrightarrow{t_n} \ldots$$

where $\forall i \in \mathbb{N}$ $s_i \in S$ and $\mathbf{Q}(s_i, s_{i+1}) > 0$ and $t_i \in \mathbb{R}_{>0}$. A finite path σ is a finite sequence $s_0 \xrightarrow{t_0} s_1 \xrightarrow{t_1} s_2 \ldots \xrightarrow{t_{l-1}} s_l$ where s_l is an absorbing *state.*

Let $\sigma = s_0 \xrightarrow{t_0} s_1 \xrightarrow{t_1} s_2 \ldots \xrightarrow{t_{n-1}} s_n \xrightarrow{t_n} \ldots$ be a path over a CTMC $\mathcal{M}$. The following notations concerning CMTC's paths are widely used throughout the remaining part of this work.

- $\sigma[i]$: is the $(i+1)$-th state of path σ, where $i \in \mathbb{N}$.
- $\delta(\sigma, i)$: time spent by the path σ in the state s_i.
- $\sigma@t$: state which the path σ is in at time t, where $t \in \mathbb{R}_{\geq 0}$

- $\overline{\sigma} = s_0 \longrightarrow s_1 \longrightarrow s_2 \ldots \longrightarrow s_n \longrightarrow \ldots$: is the *embedded untimed* path of σ (i.e. the sequence of states σ consists of). From definition 2.3.3 it clearly follows that $\overline{\sigma}$ is a path over both the *embedded* discrete-time Markov Chain $\overline{\mathcal{M}}$ as well as over the *embedded* Labelled Transition System $\hat{\mathcal{M}}$.

Generator (untimed) path: in the remaining part of the thesis the k-th prefix of a path $\overline{\sigma} = s_o \rightarrow s_1 \longrightarrow s_2 \ldots \longrightarrow s_n \longrightarrow \ldots$ over the *embedded* LTS $\hat{\mathcal{M}}$, is also referred as the *generator* of any cylinder-set $C(s_0, I_0, s_1, \ldots, I_{k-1}, s_k)$ where $(I_o, I_1, \ldots, I_{k-1})$ is any k-tuple of positive real intervals. We will use $\overline{\sigma}$ to denote the *generator* $\overline{\sigma} = s_o \rightarrow s_1 \longrightarrow s_2 \ldots \longrightarrow s_k$. A *generator* $\overline{\sigma}$ characterises all the timed paths determined by the sequence of states $\overline{\sigma}$ consists of.

Generally speaking when we consider the execution represented by a path σ, $t_i = \delta(\sigma, i)$ is the time spent by the system in its i-th state ($\sigma[i]$). Besides the system is in state $\sigma[i]$ in the interval $[a_i, b_i]$ where $a_i = \sum_{0 \leq j < i} \delta(\sigma, j)$ and $b_i = \sum_{0 \leq j \leq i} \delta(\sigma, j)$, which means that, the system enters the state $\sigma[i]$ at time a_i and leaves it at time $b_i = a_i + t_i$.

The next definition shows how the probability measure of CTMC's paths is obtained as a function of the initial-state distribution α.

Definition 2.3.4 (Borel space) *Given a sequence of states $s_0, \ldots, s_k$ from a CTMC $\mathcal{M}$ such that $\mathbf{Q}(s_i, s_{i+1}) > 0$ ($0 \leq i < k$) and a sequence of non empty intervals $I_0, \ldots, I_{k-1}$ in $\mathbb{R}_{\geq 0}$ then $C(s_0, I_o, s_1, \ldots, I_{k-1}, s_k)$ denotes the* cylinder set *containing all the paths $s_0 \xrightarrow{t_0} s_1 \ldots \xrightarrow{t_{k-1}} s_k$ such that $\delta(\sigma, i) \in I_i$ $\forall i < k$. Let $\mathcal{F}(Path)$ denote the smallest σ-algebra containing all cylinder sets $C(s_0, I_o, s_1, \ldots, I_{k-1}, s_k)$, then any initial distribution α yields a probability measure Pr_α[1] on $\mathcal{F}(Path)$, inductively defined on k in the following manner:*

$$Pr_\alpha(C(s_0, I_o \ldots, I_{k-1}, s_k)) = \begin{cases} \alpha(s_0) & \textbf{iff} \quad k = 0 \\ \\ Pr_\alpha(C(s_0, I_o, s_1, \ldots, I_{k-2}, s_{k-1})) \cdot \\ \quad \mathbf{P}(s_{k-1}, s_k) \cdot \left(e^{-E(s_{k-1}) \cdot a} - e^{-E(s_{k-1}) \cdot b}\right) & \textbf{iff} \quad k > 0 \end{cases}$$

where $a = \inf(I_{k-1})$ and $b = \sup(I_{k-1})$ (if $b = \infty$ and $\lambda > 0$, let $e^{-\lambda\infty} = 0$).

[1] with only one initial state s (i.e. $\alpha(s) = 1$), then we adopt the notation Pr_s instead of Pr_α.

Intuitively we have that, given a CTMC $\mathcal{M}$, then any path $\overline{\sigma} = s_0, \ldots, s_k$ over the *embedded* LTS interleaved with a sequence of intervals $I_0, \ldots, I_{k-1}$ characterises a set of paths over $\mathcal{M}$ (i.e. the paths going through the states $\overline{\sigma}$ consists of within t_i time units where each t_i falls in the interval I_i). The definition of Borel space tells us that, provided an initial distribution α has been given, the probability measure of the set of paths characterised by the sequence of states $\overline{\sigma}$ and the sequence of intervals $I_0, \ldots, I_{k-1}$ depends on the probability of each step in $\overline{\sigma}$ as well as on the dimension of each interval I_i.

Steady-state probability and transient-state probability

CTMCs are characterised by two major types of probabilities which concern states. The *steady-state* probability of a state s indicates the likelihood of the system of being in state s on the *long run*, which is, when we imagine observing the system behaviour for an infinite time. On the other hand the *transient-state* probability of a state s at time t provides an indication of how likely it is for the modelled system to be in state s at time t.

The computation of the *steady-state* and *transient-state* distributions of a given CTMC are basic results in the Markov Chains' theory, exhaustively treated in the literature: see for example, [17],[42],[44].

Both *transient-state* and *steady-state* probabilities can be expressed in terms of a probability measure of sets of paths, in the following way:

$$\pi^{\mathcal{M}}(\alpha, s', t) = \Pr_{\alpha}\{\sigma \in Path^{\mathcal{M}} | \sigma@t = s'\} \quad \textit{transient-state}$$

where $\pi^{\mathcal{M}}(\alpha, s', t)$ denotes the probability of being in state s' at time t when the initial distribution for the states of $\mathcal{M}$ is α, while

$$\pi^{\mathcal{M}}(\alpha, s') = \lim_{t \to \infty} \pi^{\mathcal{M}}(\alpha, s', t) \quad \textit{steady-state}$$

is the probability of being in state s' when the time tends to infinite, given an initial distribution α.

In its first definition ([1],[3]) CSL syntax included a single type of *probabilistic* operator ($\mathcal{P}_{\trianglelefteq p}$) and a single type of *path* operator, the *time-bounded* Until. In [9, 4, 5] Kaoten *et* al. have enriched the original CSL expressiveness by adding a second *prob-*

abilistic operator, namely $\mathcal{S}_{\unlhd p}$ allowing reference to be made to steady-state measures, plus a second *path* operator, the *time-bounded* Next.

In this work we are going to refer to the "extended", version of the CSL whose syntax is formally defined as follows:

Definition 2.3.5 (CSL syntax) *The syntax of CSL state-formulae (*ϕ*) and path-formulae (*φ*) is inductively defined as follows with respect to the set of atomic proposition AP:*

$$\phi := a \mid tt \mid \neg\phi \mid \phi \wedge \phi \mid \mathcal{S}_{\unlhd p}(\phi) \mid \mathcal{P}_{\unlhd p}(\varphi) \qquad \textit{(state-formulae)}$$
$$\varphi := X^I\, \phi \mid \phi\, \mathcal{U}^I \phi \qquad \textit{(path-formulae)}$$

where $a \in AP$*,* $p \in [0,1]$ *is a real number,* $\unlhd \in \{<, \leq, >, \geq\}$ *and* $I \in \mathbb{R}_{\geq 0}$ *is a non empty interval.*

The semantics of CSL is defined in terms of a twofold relationship denoted by $\models$, which relates states of a given CTMC to *state-formulae* (ϕ) and paths to *path-formulae* (φ). Let $\mathcal{M} = (S, \mathbf{Q}, L)$ be a labelled CTMC then the relation $\models$ for both *state-formulae* and *path-formulae* is defined as follows:

Definition 2.3.6 (CSL *state-formulae* semantics) *Let* $Sat(\phi) = \{s \in S \mid s \models \phi\}$*. The relation* $\models$ *for the CSL state-formulae is defined by*

$$
\begin{array}{ll}
s \models tt \ \textbf{ forall } \ s \in S & s \models \phi' \wedge \phi'' \ \textbf{ iff } \ s \models \phi' \wedge s \models \phi'' \\
s \models a \ \textbf{ iff } \ a \in L(s) & s \models \neg\phi \ \textbf{ iff } \ s \not\models \phi \\
s \models \mathcal{S}_{\unlhd p}(\phi) \ \textbf{ iff } \ \pi^{\mathcal{M}}(s, Sat(\phi)) \unlhd p & s \models \mathcal{P}_{\unlhd p}(\varphi) \ \textbf{ iff } \ Prob^{\mathcal{M}}(s, \varphi) \unlhd p
\end{array}
$$

where $s \in S$ *and* $Prob^{\mathcal{M}}(s, \varphi)$ *denotes the probability measure of all paths* $\sigma \in Path^{\mathcal{M}}(s)$ *satisfying* φ *when the system starts from state s:*

$$Prob^{\mathcal{M}}(s, \varphi) = Pr_s\{\sigma \in Path^{\mathcal{M}}(s) \mid \sigma \models \varphi\}$$

Definition 2.3.7 (CSL *path-formulae* semantics) *The relation* $\models$ *for the CSL path-formulae is defined by*

$$
\begin{array}{ll}
\sigma \models X^I\, \phi & \textbf{iff } \ \sigma[1] \textit{ is defined and } \sigma[1] \models \phi \wedge \delta(\sigma, 0) \in I \\
\sigma \models \phi' \mathcal{U}^I \phi'' & \textbf{iff } \ \exists t \in I : \sigma@t \models \phi'' \wedge (\forall t' \in [0, t), \sigma@t' \models \phi'
\end{array}
$$

where $\sigma \in Path^{\mathcal{M}}$.

Alternatively the timed-until semantics can be defined as follows:

$$\sigma \models \phi' \mathcal{U}^I \phi'' \ \textbf{iff} \ \exists i \geq 0 : \Big[(\sigma[i] \models \phi' \wedge \phi'' \ \wedge [a_i, b_i] \cap I \neq \emptyset) \ \vee (\sigma[i] \models \neg\phi' \wedge \phi'' \wedge a_i \in I) \Big]$$
$$\wedge \Big[\sigma[j] \models \phi', \forall j < i \Big] \sigma@t \models \phi'' \wedge (\forall t' \in [0,t)), \sigma@t' \models \phi'$$

meaning that a path σ satisfies the timed-until $\phi' \mathcal{U}^I \phi''$ if there exists a future state $\sigma[i]$ in which either

- the *premise* and *target* of the until formula are satisfied in $\sigma[i]$ and some of the time instants spent at $\sigma[i]$ do satisfy the bound I.

- the *target* but not the *premise* of the until formula is satisfied in $\sigma[i]$ and the time instant at which $\sigma[i]$ is entered, $t = a_i$, falls into the bound I ($a_i \in I$). We observe that in this case the only relevant time instant to care about is the time σ enters the state where $\neg\phi' \wedge \phi''$ is satisfied (i.e. $\sigma[i]$). In fact, according to the semantics of the until formula, there must be a time instant $t \in I$ at which σ satisfies the *target* ϕ'' and such that for any preceding time instant $t' < t$, σ satisfies the *premise* ϕ'. Since here we are assuming the *premise* ϕ' to be not satisfied in $\sigma[i]$, then clearly the only possible $t \in [a_i, b_i]$ is $t = a_i$ (in fact $\forall t \in (a_i, b_i]$ we have that $\forall t' \in [a_i, t)$, $\sigma@t' \not\models \phi'$).

From the semantics of CSL formulae, we note that with the empty interval $I = \emptyset$ any time-bounded path formula is clearly not satisfiable. Furthermore, we notice that the usual untimed version of the *next* and *until* can be obtained as a special case of the bounded ones by taking $I = [0, \infty)$. For the sake of simplicity we will omit $I = [0, \infty)$ from the notation, hence the unbounded *next* and *until* formulae will be simply denoted by $\mathcal{X}\phi$ and $(\phi' \, \mathcal{U} \, \phi'')$ respectively.

Remark 2.3.1 *Let* σ *be a path over a CTMC* $\mathcal{M}$ *satisfying the timed-until* $(\phi' \mathcal{U}^I \phi'')$*,* $\sigma \in Path^{\mathcal{M}}(s, \phi' \mathcal{U}^I \phi'')$*, then the* embedded untimed *path* $\overline{\sigma}$ *satisfies the corresponding untimed-until,* $\overline{\sigma} \in Path^{\overline{\mathcal{M}}}(s, \phi' \mathcal{U} \phi'')$.

The above remark is a direct consequence of the semantics of timed-until formulae.

2.3.1 Model-checking CSL formulae

The model-checking algorithm for CSL formulae works the same way as the CTL ones for all the non-probabilistic *state-formulae*: the set $Sat(\phi)$ is recursively computed as the fixed-point of a function which marks the states of $\mathcal{M}$ with sub-formulae of ϕ [12]. The computation of $Sat(\phi)$ for the probabilistic *state-formulae* requires instead a specific treatment.

Computing steady-state measures. From the CSL semantics (definition 2.3.6), we know that the steady-state formula $\mathcal{S}_{\trianglelefteq p}(\phi)$ is satisfied in a state s if and only if the probability measure of paths $\sigma \in Path(s)$ starting from s and satisfying ϕ at time infinite is $\trianglelefteq p$, which is $\pi(s, Sat(\phi)) \trianglelefteq p$.

If G is the underlying directed graph of a CTMC $\mathcal{M}$ then a subgraph B is a *bottom strongly connected component* (BSCC) of G if it is a maximal strongly connected component with no edges outside its vertices (i.e. $Reach(s) = B$ for all $s \in B$). Let $B(\mathcal{M})$ denote the set of BSCC of $\mathcal{M}$. The computation of $\pi(s, Sat(\phi))$ is based on the following proposition:

Proposition 2.3.1 *Let $\mathcal{M} = (S, \mathbf{Q}, L)$ be a CTMC and $s \in S$, $S' \subseteq S$, then*

$$\pi(s, S') = \sum_{B \in B(\mathcal{M})} \Big(Prob(s, \diamond at_B) \cdot \sum_{s' \in B \cap S'} \pi^B(s') \Big)$$

where $\pi^B(s')$ is the steady-state probability of s' in BSCC B and $at_B \equiv \bigvee_{s \in B} at_s$.

Algorithm for $\mathcal{S}_{\trianglelefteq p}(\phi)$. Relying on proposition 2.3.1 it is possible to characterise an algorithm for the computation of $\pi(s, Sat(\phi))$

1. the set of states satisfying ϕ, is recursively determined.

2. the set of BSCC of $\mathcal{M}$ (i.e. $B(\mathcal{M})$) is computed by means of some existing algorithm (e.g. [43]).

3. for each $B \in B(\mathcal{M})$ the steady-state distribution π^B is computed. This implies

the solution of the system of linear equations

$$\sum_{\substack{s\in B\\ s\neq s'}} \pi^B(s)\cdot \mathbf{Q}(s,s') = \pi^B(s')\cdot \sum_{\substack{s\in B\\ s\neq s'}} \cdot\mathbf{Q}(s,s')$$

$$\sum_{s\in B} \pi^B(s) = 1$$

unless $B=\{s'\}$ in which case, trivially, $\pi^B(s')=1$.

4. the steady-state probability of each state $s'\in Sat(\phi)$ is then obtained by weighting the steady-state probability $\pi^B(s')$, given that $s'\in B$, by the probability of reaching B from s. Such probability, denoted $Prob(s,\diamond at_B)$, is given by the solution of the following system of linear equation:

$$Prob(s,\diamond at_B) = \begin{cases} 1 & \textbf{if } s\models at_B \\ \sum_{s'} \mathbf{P}(s,s')\cdot Prob(s',\diamond at_B) & \text{otherwise} \end{cases}$$

If $\mathcal{M}$ consists of a single BSCC, namely B, then

$$\pi(s,S') = \sum_{s\in S'} \pi(s')$$

where $\pi(s')$ stands for $\pi^B(s')$, the steady-state probability of s' with respect to the whole CTMC (i.e. $B=\mathcal{M}$).

Computing probabilistic path measures. The verification of probabilistic path formulae like $\mathcal{P}_{\trianglelefteq p}(\varphi)$ with respect to a state s, relies on the characterisation of the measure $Prob(s,\varphi)$ (see definition 2.3.6). A distinction is needed between timed-next and timed-until formulae. Proposition 2.3.2 characterises the measure of the paths satisfying a timed-next formula. Theorem 2.3.1 concerns the measure of the paths satisfying a timed-until formula. As a consequence of proposition 2.3.2 and theorem 2.3.1, procedures for model checking time bounded Next and Until, respectively, are given.

Proposition 2.3.2 *For $s\in S$ and interval $I\subseteq\mathbb{R}_{\geq 0}$ with $a=\inf I$, $b=\sup I$ and a CSL state-formula ϕ:*

$$Prob(s,\mathcal{X}^I\phi) = \left(e^{-a\cdot E(s)} - e^{-b\cdot E(s)}\right)\cdot \sum_{s'\models\phi} \mathbf{P}(s,s')$$

Proof. Is a direct consequence of the Borel-space construction (see definition 2.3.4).

□

The truth of proposition 2.3.2 with respect to a generic type of interval $I \in \mathbb{R}_{\geq 0}$ (i.e. either open or closed), relies on the fact that

$$Prob(s, \phi' \mathcal{U}^I \phi'') = Prob(s, \phi' \mathcal{U}^{cl(I)} \phi'')$$
$$Prob(s, \mathcal{X}^I \phi) = Prob(s, \mathcal{X}^{cl(I)} \phi)$$

where $cl(I)$ is the closure of I. This is a consequence of the fact that the probability measure of a *cylinder-set* $C(s_o, I_0, s_1, \ldots I_{k-1}, s_k)$ does not change when some of its intervals I_i $(i < k)$ are replaced by their closure.

Algorithm for $\mathcal{P}_{\trianglelefteq p}(X^I \phi)$**.** Relying on proposition 2.3.2 the following algorithm for computing $Sat(\mathcal{P}_{\trianglelefteq p}(\mathcal{X}^I \phi))$ is defined:

1. the set $Sat(\phi)$ is recursively determined.
2. the state vector $b_{I\phi}$ is computed, where
$$b_{I\phi}(s) = \begin{cases} e^{-E(s)\cdot \inf I} - e^{-E(s)\cdot \sup I} & \textbf{if } s \in Sat(\phi) \\ 0 & \text{otherwise} \end{cases}$$
3. the state vector $\underline{Prob}(\mathcal{X}^I \phi) = (\ldots, Prob(s, \mathcal{X}^I \phi), \ldots)$ is computed by multiplication of $\mathbf{P}$ by $b_{I\phi}$
$$\underline{Prob}(\mathcal{X}^I \phi) = \mathbf{P} \cdot b_{I\phi}$$
4. finally, a state s is added to $Sat(\mathcal{P}_{\trianglelefteq p}(\mathcal{X}^I \phi))$ if and only if its correspondent element of $\underline{Prob}(\mathcal{X}^I \phi)$ satisfies the bound p, which is $Prob(s, \mathcal{X}^I \phi) \trianglelefteq p$.

The following theorem provides a recursive algorithm to compute the probability measure of the paths satisfying a timed-until formula $(\phi' \mathcal{U}^I \phi'')$. $Path(s,I)$ and $Prob(s,I)$ denote, respectively, the set of paths starting at s and satisfying the timed-until $(\phi' \mathcal{U} \phi'')$

and its probability measure (i.e. $Prob(s,I) = Prob(s, \phi' \,\mathcal{U}^I\, \phi''))$. Furthermore $I \ominus x$ denotes the set $\{t - x \mid t \in I \wedge t \geq x\}$.

Theorem 2.3.1 (Time Bounded Until probability measure) *For $s \in S$ and interval $I \subseteq \mathbb{R}_{\geq 0}$ with $a = \inf I$ and $b = \sup I$ and ϕ' and ϕ'' CSL state-formulae. The $Prob(s, \phi' \,\mathcal{U}^I \phi'')$ is recursively defined as follows:*

$$Prob(s, \phi' \,\mathcal{U}^I \phi'') = \begin{cases} 1 & \textbf{iff } s \models \phi'' \wedge \neg\phi' \\ & \quad \textit{and} \quad a = 0 \\ \int_0^b \sum_{s' \in S} \mathbf{T}(s,s',x) \cdot Prob(s', \phi' \,\mathcal{U}^{I \ominus x} \phi'')dx & \textbf{iff } s \models \phi' \wedge \neg\phi'' \\ e^{-E(s)\cdot a} + & \\ \quad \int_0^a \sum_{s' \in S} \mathbf{T}(s,s',x) \cdot Prob(s', \phi' \,\mathcal{U}^{I \ominus x} \phi'')dx & \textbf{iff } s \models \phi' \wedge \phi'' \\ 0 & \textit{otherwise} \end{cases} \tag{2.3.1}$$

where $\mathbf{T}(s,s',x) = \mathbf{P}(s,s') \cdot E(s) \cdot e^{-E(s)\cdot x}$ denotes the density of moving from state s to state s' in x time units.

Proof. see [5]. □

The formal proof of the above theorem is out of the scope of this work, nevertheless it is relevant to provide an intuitive explanation of the different cases characterising the function $Prob(s, \phi' \,\mathcal{U}^I \phi'')$.

1. $a = 0$ and $s \models \neg\phi' \wedge \phi''$. In such a case any path starting at s clearly satisfies $(\phi' \,\mathcal{U}^I \phi'')$. This is the case since s satisfies ϕ'' and also because trivially each path starting at s is indeed in s at time $t = 0$, which, in this case, is the infinum of the considered interval I. Therefore $Prob(s, \phi' \,\mathcal{U}^I \phi'') = 1$.

2. $s \models \phi' \wedge \neg\phi''$ then $Path(s,I)$ consists of all the paths σ of the form $\sigma = s \xrightarrow{x} \sigma'$ with $0 \leq x \leq b$ and $\sigma' \in Path(s', I \ominus x)$, which is: if we are in a state s which satisfies ϕ' but not ϕ'' then the paths we have to account for are those ones that

leave s within b time units, say at x, to reach a state s' from which they will satisfy the until formula within a time y such that the sum $x+y$ is in I.

3. $s \models \phi' \wedge \phi''$ then $Path(s,I)$ consists of all the paths σ of the form $\sigma = s \xrightarrow{x} \sigma'$ where either $0 \le x \le a$ and $\sigma' \in Path(s', I \ominus x)$ or $x > a$. In fact, since we are assuming $s \models \phi' \wedge \phi''$, we have to consider not only the paths which leave s within a time units to reach a state s' from which they will satisfy the until formula within time y such that $x+y$ is in I but also those ones which leave s after a time units have elapsed. Since ϕ'' is assumed to be true in s (i.e. the corresponding untimed-until is satisfied in s) then staying in s for a time greater then a ensures that the until is satisfied within the given bound I.

4. Any other case different from the above ones leads to a probability measure equal to zero, as either the corresponding untimed-until is not satisfiable in the source state s or the time bound I is unmatchable.

Corollary 2.3.1 (Unbounded path formulae probability measure) *For $s \in S$ and ϕ', ϕ'' CSL state formulae*

1. $Prob(s, \mathcal{X}\phi') = \sum_{s' \models \phi'} \mathbf{P}(s,s')$.

2.

$$Prob(s, \phi' \, \mathcal{U}^I \phi'') = \begin{cases} 1 & \textbf{iff } s \models \phi' \\ \sum_{s' \in S} \mathbf{P}(s,s') \cdot Prob(s', \phi' \, \mathcal{U} \phi'') & \textbf{iff } s \models \phi' \wedge \neg\phi'' \\ 0 & \textit{otherwise} \end{cases} \tag{2.3.2}$$

Proof. Trivial from proposition 2.3.1 and theorem 2.3.1 with $I = [0, \infty)$.

□

The results in Corollary 2.3.1 are identical to discrete-time framework's ones: the probability for satisfying next and until formulae in the logic PCTL are determined in the same way ([19]).

Algorithm for $\mathcal{P}_{\trianglelefteq p}(X\phi)$. Corollary 2.3.1 suggests the following algorithm for determining $Sat(\mathcal{P}_{\trianglelefteq p}(X\phi))$:

1. the set $Sat(\phi)$ is recursively determined and, as a result, the vector $\underline{i}_\phi$ given by
$$\underline{i}_\phi(s) = \begin{cases} 1 & \textbf{if } s \models \phi \\ 0 & \text{otherwise} \end{cases}$$
is computed.
2. the vector $\underline{Prob}(X\phi) = \mathbf{P} \cdot \underline{i}_\phi$ is computed.
3. a state s is in $Sat(\mathcal{P}_{\trianglelefteq p}(X\phi))$ if and only if $\underline{Prob}(X\phi)(s) \trianglelefteq p$.

Algorithm for $\mathcal{P}_{\trianglelefteq p}(\phi' U \phi'')$. Corollary 2.3.1 also suggests the following algorithm for determining $Sat(\mathcal{P}_{\trianglelefteq p}(\phi' \; U \; \phi''))$:

1. the matrix $\hat{\mathbf{P}}$ defined as
$$\hat{\mathbf{P}}(s,s') = \begin{cases} \mathbf{P}(s,s') & \textbf{if } s \models \phi' \wedge \neg\phi'' \\ 0 & \text{otherwise} \end{cases}$$
is computed.
2. the vector $\underline{Prob}(\phi' \; U \; \phi'')$ is computed as the least solution of the system of linear equations
$$\underline{x} = \hat{\mathbf{P}} \cdot \underline{x} + \underline{i}_{\phi''}$$

So far the algorithms for checking Next formulae (either bounded or unbounded) and Until formulae (unbounded only, i.e. $I = [0,\infty)$), have been presented. Verifying a Next formula, essentially, implies the computation of a matrix-vector product. In contrast, the verification of an unbounded Until requires the solution of a system of linear equations.

Time-bounded Until by means of transient analysis.
As a consequence of Theorem 2.3.1, the computation of the state vector $\underline{Prob}(\phi' \; U^I \; \phi'')$,

for a bounded Until formula, requires the solution of a Volterra integral equation system. This can be done by means of some (computationally expensive) numerical techniques. Alternatively, in [5], the authors, show a number of *correctness-preserving transformations* by means of which the model-checking problem for time-bounded Until formulae reduces to a transient analysis of a transformed CTMC. In essence, it is proved that computing the probability measure for a time-bounded Until with respect to a certain CMTC $\mathcal{M}$ is equivalent to computing the transient probability of certain states with respect to a CMTC $\mathcal{M}'$ obtained by making some states of $\mathcal{M}$ *absorbing*. The formalisation of these *preserving transformations* is provided in the following definition and proposition.

Definition 2.3.8 *Let $\mathcal{M} = (S, \mathbf{Q}, L)$ be a CTMC and ϕ a CSL state formula. The CMTC obtained by making all ϕ-states in $\mathcal{M}$ absorbing is denoted $\mathcal{M}[\phi]$. $\mathcal{M}[\phi] = (S, \mathbf{Q}', L)$ where*

$$\mathbf{Q}'(s,s') = \begin{cases} \mathbf{Q}(s,s') & \textbf{if} \quad s \not\models \phi \\ 0 & \textit{otherwise} \end{cases}$$

It should be noted that $\mathcal{M}[\phi'][\phi''] = \mathcal{M}[\phi' \vee \phi'']$. Relying on the definition of transformed CMTC $\mathcal{M}[\phi]$, the following properties can be proved.

Proposition 2.3.3 *Let ϕ', ϕ'' be two CSL state formulae and $\mathcal{M}$ a CMTC whose ϕ'' states are absorbing (i.e. $\mathcal{M} = \mathcal{M}[\phi'']$), then:*

$$\begin{aligned} Prob^{\mathcal{M}}(s, \phi' \; \mathcal{U}^{[0,t]} \; \phi'') &= Prob^{\mathcal{M}[\neg\phi' \wedge \neg\phi'']}(s, \diamond^{[t,t]} \; \phi'') \\ &= \sum_{s'' \models \phi''} \pi^{\mathcal{M}[\neg\phi' \wedge \neg\phi'']}(s, s'', t) \end{aligned}$$

Proof. The proof can be found in [5].

The above proposition, shows that on a CMTC $\mathcal{M}$, the probability measure for an Until formula $(\phi' \; \mathcal{U} \; \phi'')$ bounded by the interval $[0,t]$, is equivalent to the transient probability at time t, of the ϕ'' states on the CMTC obtained by $\mathcal{M}$ from making every $(\neg\phi' \wedge \neg\phi'')$ state absorbing.

Theorem 2.3.2 *Let $\mathcal{M}$ be an arbitrary CMTC and ϕ', ϕ'' two CSL state formulae, then:*

$$\begin{aligned} Prob^{\mathcal{M}}(s, \phi' \ \mathcal{U}^{[0,t]} \ \phi'') &= Prob^{\mathcal{M}[\phi'']}(s, \phi' \ \mathcal{U}^{[0,t]} \ \phi'') \\ &= \sum_{s'' \models \phi''} \pi^{\mathcal{M}[\neg\phi' \vee \phi'']}(s, s'', t) \end{aligned}$$

Proof. See [5].

The above theorem shows that also for an arbitrary CMTC $\mathcal{M}$, the verification of a bounded Until formula ($\phi' \ \mathcal{U}^I \ \phi''$) with bounding interval $I = [0,t]$, is equivalent to a transient probability analysis, at time t, on a modified CMTC (i.e. $\mathcal{M}[\neg\phi' \vee \phi'']$).

Theorem 2.3.3 *Let $\mathcal{M}$ be an arbitrary CMTC and ϕ', ϕ'' two CSL state formulae and t, t' two time instant such that $0 < t \leq t'$, then:*

$$Prob^{\mathcal{M}}(s, \phi' \ \mathcal{U}^{[t,t']} \ \phi'') = \sum_{s' \models \phi'} \sum_{s'' \models \phi''} \pi^{\mathcal{M}[\neg\phi'}(s, s', t) \cdot \pi^{\mathcal{M}[\neg\phi' \vee \phi'']}(s', s'', t' - t)$$

Proof. See [5].

Corollary 2.3.2 *Let $\mathcal{M}$ be an arbitrary CMTC and ϕ', ϕ'' two CSL state formulae then:*

$$Prob^{\mathcal{M}}(s, \phi' \ \mathcal{U}^{[t,t]} \ \phi'') = \sum_{s' \models \phi' \wedge \phi''} \pi^{\mathcal{M}[\neg\phi'}(s, s', t$$

Proof. See [5].

Finally, the above theorem proves that also in the case of a bounding interval whose infinum is greater than zero (i.e. $I = [t,t']$ and $0 < t \leq t'$), the bounded-until model-checking problem on an arbitrary CMTC $\mathcal{M}$ boils down to the combined transient analysis of two modified CTMCs, namely: $\mathcal{M}[\neg\phi']$ and $\mathcal{M}[\neg\phi' \wedge \phi'']$. Moreover, when the bounding interval coincides with a point (i.e. $I = [t,t]$), a similar result holds (the above corollary): in that case the verification of the bounded Until formula corresponds to the transient analysis of the modified CMTC $\mathcal{M}[\neg\phi']$.

The major consequence of the above properties is that they show that the time-bounded Until model-checking problem (in any possible case) with respect to an arbitrary CMTC can be replaced by the transient analysis of certain modified CMTCs. Hence, the solution of the Volterra integral equation system associated with a time-bounded Until formula is not actually needed. Instead the transient distribution for a derived CTMC can be solved. Formally this is achieved as a solution of the Chapman-Kolmogorov differential equations (see [42]). However, easily implementable methods, such as *Uniformisation* ([42]), can be applied in order to obtain an approximate solution of the transient distribution.

2.4 Compositionality and Model-Checking

In this section the idea of compositionality applied to model-checking techniques is described. Before that, an introductory overview on compositionality, in general, is provided.

2.4.1 Formal modelling and compositionality

Formal methods for systems' verification concern the development of methodologies for the analysis of the behaviour of real systems, in particular computer and telecommunication systems.

The basic idea is to provide a means through which an abstract representation of the system, a *model*, can be built. A model has to enclose those bits of information which are relevant to capture the aspects of the system's behaviour one is interested to analyse. As a result a model comes with a set of parameters which have to be instatiated with proper values in order for a performance study to be carried out.

The model's evaluation is obtained either via the solution of a set of equations leading to analytical results (analytical models) or via simulation, leading to statistical results (a concise but complete course on simulation can be found in see [40]).

High-level modelling formalisms like Petri Nets, Process Algebras and Queueing Networks, easily lead to huge and complex models the solution of which turns to be unfeasible. As a consequence, compositional approaches to performance modelling have increasingly gained interest as means to face the tractability of models which

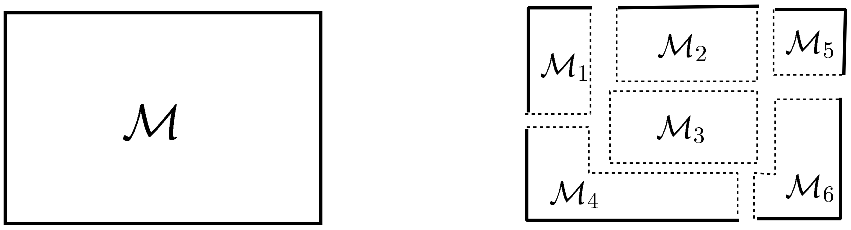


Figure 2.5: A decomposable model M

have a large dimension. These approaches decompose the system into a number of smaller and more easily modelled subsystems, as depicted in figure 2.5. The aim is twofold: helping model construction as well as model solution of large and complex systems, meaning, in the latter case, that the performance evaluation of a big system can be retrieved by the analysis of the subsystems it consists of. In the following a brief introduction to these three modelling formalisms is provided.

Petri Nets. Petri Nets are a formalism appropriate for modelling systems with concurrency. Formally a Petri Net is a bipartite, oriented graph characterised by two class of nodes: places and transitions. Places are connected to transitions and vice versa. Places are filled with *tokens*. Transitions normally represent activities of the modelled system. A transition is enabled whenever each of its input places contains enough tokens. When more than one transition is enabled a competition between the activities they represent takes place. The system's dynamic is captured by transition *firing*: when an enabled transition *fires*, tokens from its input places are removed and tokens into its output places are created; as a result the set of enabled activities can possibly vary.

Since their first definition [10], a plethora of different variants of Petri Nets have been developed. An interesting survey on the classification of Petri Nets can be found in [30]. Different types of structural bounds allow to refer to different types of system. For example places in Conditions/Events Nets [20] are meant to represent boolean conditions, hence they can contain at most one token in any possible marking and, moreover, tokens represent unstructured values. Place/Transition Nets, instead, allow places to be marked with any (positive) integer number of tokens, where again tokens represent unstructured values. High-level Petri Nets like Coloured Petri-Nets (CPN)[25, 26] and Well-formed Nets (WN) [18] allow one to represent structured information: places contain multi-sets of typed tokens.

Concerning performance evaluation analysis, Petri Nets can be classified with respect to the assumptions characterising the duration of the activity modelled by the net's transitions. Timed Petri Nets (TPN) allow one to represent timed activities. Deterministic TPN are suitable to model systems whose activities' duration is supposed to be deterministically known. On the other hand Stochastic Petri Nets (SPN)[36] and their Generalised evolution (GSPN)[34], assume the activity duration to be an exponentially distributed random variable. Although in the literature Petri Nets with *non-Markovian* stochastic behaviours have been widely studied, the most relevant type of stochastic process underlying a (stochastic) Petri Net are CTMCs. Hence Petri Nets are a high-level language for specification of CTMCs.

Though Petri Nets do not come with an inherent compositional rule, many techniques have been developed to build a Petri Nets model by combination of a number of submodels. There are two main ways for composing Petri Nets, either by transitions superposition or by place superposition. An example of a compositional framework for GSPN, based on transition superposition, is given by [41]

Process Algebras. Process Algebras are mathematical theories which model concurrent systems by their algebra. Examples are the Calculus of Concurrent Systems (CCS)[35], Communicating Sequential Processes (CSP)[11] and the Algebra of Communicating Processes (ACP) [24].

Process Algebras differ from Petri-Nets in that they lack of a notion of *entity* or *flow* within the model. On the other hand Process Algebras come with an inherent compositional reasoning: the model is given by composition of terms or processes (submodels) through a defined set of operators. Each component is characterised by the set of actions it undertakes. A component's behaviour is described by means of combinators like the *prefix*, allowing the specification of the first action a component takes, the *choice*, allowing the specification of an alternative between two possible actions and the *cooperation* which permits characterisation of the interaction between two different components.

Pure process algebras do not allow consideration of time: they are meant to describe the behaviour of a system as the set of possible sequences of actions, disregarding the time. Timed extensions of process algebras like TCCS [16] have been realised, providing a means to associate a determined delay to the system's actions.

Stochastic process algebras like, for example, PEPA[21] and EMPA[6], permit the replacement of the nondeterministic choice which comes with pure and timed process algebras with a probabilistic one. The operational semantics of such algebras describe the CTMC which underlies the model. Stochastic process algebras are a useful means which naturally allows for a compositional specification of a CMTC.

In [22] an interesting overview on how the inherent compositionality of stochastic process algebras can be exploited for the solution of the underlying Markov process is given.

Queueing Networks. Queueing Networks are a language for modelling systems which consist of a number of customers competing to access a number of services. Formally a Queueing Network is an oriented graph whose nodes, usually also called *service centres*, are *queues*. A queue is characterise by an arrival process, a buffer where the customers queue for the service and one or more servers representing the resources customers are about to use. A queue is described by five factors, denoted by means of a 5-tuple A$/S/c/m/N$:

A the arrival process, where M is used to denote a Markov process, while G and D denote, respectively, a general and a deterministic distribution.

S is the service process and the above notations M,G and D are again used as distribution identifiers.

c is the number of servers the queue consists of.

m is the buffer capacity

N is the customer population

where infinite is the default assumption for both the buffer capacity and customers population. Hence $M/M/1$ denotes a single-server queue whose customers' arrival time and service time are both Markovian.

The queueing discipline determines how the customer in a queue are going to be served. Typical serving policies are *first-come-first-served* (FCFS), where the longest waiting customer is the first to be served or *process sharing* (PS) where the service capacity is equally shared among the customers in the queue.

Customers in a queueing network can be partitioned in *classes* according to the characteristic they exhibit. The state of a queueing network is typically given by the number of customers of each class at each service centre. Hence a state $s = (s_1, \dots, s_n)$ of a network is completely described by the states s_i ($i \in \{1, \dots, n\}$) the individual queues it is made of are in.

Queueing networks may be *closed* if the number of customers is fixed, *open* whether the population varies, or *mixed* if some classes of customers exhibit an open behaviour while some other have a closed behaviour. More details on queueing networks can be found for example in [31, 32].

Solving a queueing network characterised by exponentially distributed times, requires the computation of the long run distribution $\pi(s)$ In [15] it has been shown that a large class of queueing networks allows for a compositional solution, also termed *product form* solution, of the steady-state distribution: the probability of being at state $s = (s_1, \dots, s_n)$ on the long run can be expressed in terms of the product of the probability of each individual queue to be in sub-state s_i:

$$\pi(s_1, \dots, s_n) = G \cdot \prod_{i=1}^{n} \pi_i(s_i)$$

where G is a normalising constant. This result allows for the computation of many performance measures of a given queueing network without resorting to the underlying Markov process: only Markov processes of individual queues have to be solved.

In the literature on queueing networks a lot of effort has been put on identifying classes of networks which allow for a product-form solution of the equilibrium distribution [29, 15, 45]. Similarly a quite large number of works aiming to the characterisation Petri Nets models whose underlying process has a product form solution can be found (see for example [2, 46, 33]). In this thesis the product form framework for ergodic CTMCs described by Boucherie in [7] has been considered.

2.4.2 Compositional Model-Checking

As a performance evaluation technique, model-checking suffers by the so-called *state-space explosion* problem: complex systems result in models of huge dimension which can not be treated by any existent computational resource. A considerable amount of works aiming to increase the applicability of model-checking with respect to the model's dimension, can be found in literature.

Symbolic model-checking, concerns the study of techniques for a compact representation of the state-space based on specific data structures like Binary Decision Diagrams (BDD)[39] or Multi Terminal Binary Decision Diagrams (MTBDD)[13]. The *symbolic* approach has been applied to both non-probabilistic and probabilistic model-checking showing a great improvement with respect to the tractability of big models. In [27, 28], it has been shown that the use of BDD for the state-space representation have allowed for the verification of CTL formulae over systems that would have otherwise required 10^{20} states. Symbolic Model-checking for Markov-Chains [8, 23], instead, relies on the use of both MTBDD, for representing the linear system's matrix involved in the verification of probabilistic formulae, and BDD for representing the formula itself.

On the other hand *abstraction* in model-checking, is meant to provide means to build an abstract, hence reduced, version of the model of interest. In[14] Clarke et al. have shown that the formulae belonging to the logic $\forall CTL^*$, a subset of the CTL^* in which only the *universal* path quantifier ($\forall$) is allowed, can be verified against the abstract model, while maintaining their truth value with respect to the original system.

Finally *compositional verification* of properties in a given temporal logic, concerns the analysis of the truth of a formula when the given model is obtained by composition of a number of submodels. The goal is to investigate the possibility of inferring the truth of a formula ϕ by the verification of ϕ itself or some other formulae on the component models. In [37], for example, Grumberg and Long define a compositional rule for structures which are model for the $\forall CTL^*$ logic, proving that the validity of a formula ϕ in a component $\mathcal{M}'$ is preserved, through *preorder*, in any system built on $\mathcal{M}'$ (i.e. if ϕ is true in $\mathcal{M}'$ then it is true in any model obtained through an iterative compositional process which at some stage has involved $\mathcal{M}'$).

The goal of this thesis, is to look for a compositional approach for model-checking of CTMCs. Given a property ϕ we want to check against a CTMC $\mathcal{M}$ obtained by composition of n submodels $(\mathcal{M}_1,\ldots,\mathcal{M}_n)$, we aim to search for formulae ϕ_i determining a boolean combination of satisfiability conditions which turns out to be equivalent to the satisfiability of ϕ in $\mathcal{M}$. For example, suppose we are interested in verifying $\phi \equiv a_1 \vee \neg a_2$ with respect to a CTMC $\mathcal{M}$ given by composition of $\mathcal{M}_1$ and $\mathcal{M}_2$ where a_i is an atomic proposition of $\mathcal{M}_i$ with $i \in \{1,2\}$. Intuitively we have that

$$\mathcal{M} \models a_1 \vee \neg a_2 \iff \mathcal{M}_1 \models_1 a_1 \vee \mathcal{M}_2 \not\models_2 a_2$$

meaning that checking $\phi \equiv a_1 \vee \neg a2$ with respect to $\mathcal{M}$ is equivalent to verifying that either a_1 is valid with respect to $\mathcal{M}_1$ or a_2 is not valid in $\mathcal{M}_2$. Clearly the above example is a rather trivial one as it refers to a state-formula given by combination of atomic propositions. The derivation of formulae which lead to an equivalent combination of satisfiability conditions for a given ϕ is indeed not trivial whenever a probabilistic connective like $\mathcal{S}_{\unlhd p}$ and $\mathcal{P}_{\unlhd p}$ is involved. Chapter **??** is devoted to the study of such derived equivalent formulae.

2.5 The Boucherie product-process

In [7], Boucherie establishes a form of CTMC which is susceptible to product form solution. That result relies on the characterisation of an "independence condition" for the components of a multi-dimensional CTMC which models a number of processes competing over shared resources. The Boucherie framework characterisation relies on two basics ideas:

- **Mutual exclusion over resources:** when one process holds a resource, other processes cannot access the resource.
- **Strong blocking:** processes are subject to strong blocking conditions meaning that they cannot evolve until the resource is released.

The Boucherie framework. A collection of K regular and irreducible CTMCs, labelled $\mathcal{M}_k$, with $k = 1,\ldots,K$, at finite or countable state spaces S_k, is considered. Let

$q_k(\bar{n}_k, \bar{n}'_k)$ denotes the transition rate of $\mathcal{M}_k$ where $\bar{n}_k, \bar{n}'_k \in S_k$. Each Markov chain $\mathcal{M}_k$ is assumed to posses a unique equilibrium distribution π_k and $\pi_k(\bar{n}_k)$ is the probability for $\mathcal{M}_k$ to be in state $\bar{n}_k$ on the long-run. For this collection the product process with state space $S = S_1 \times \cdots \times S_K$ and transition rate in dimension k given by q_k, is introduced.

For such a product process it is assumed that in each transition the state in one dimension only changes, that is: in any allowed change of state of the product process, exactly one of the underlying Markov chains changes its state (i.e. synchronisation between components is not permitted). Furthermore, competition over resources (i.e. mutual exclusion) can be modelled as exclusion of parts of state space: the product process can not enter a certain area $A \subset S$.

Under these circumstances, the "independence condition" which guarantee the product form solution roughly states that *if the product process is in state* $\bar{n} = (\bar{n}_1, \ldots, \bar{n}_k, \ldots, \bar{n}_K)$ *then if* $\bar{n}' = (\bar{n}'_1, \ldots, \bar{n}'_k, \ldots, \bar{n}'_K) \in A$ *is a state in the forbidden area* A_i *(i.e. a state which breaks the mutual exclusion condition for a resource* R_i*) with* $\bar{n} = \bar{n}'$ *except for component* $\bar{n}_k$ *(i.e.* $\bar{n}'$ *would be reachable from* $\bar{n}$ *with a transition along k dimension) then Markov chain* $\mathcal{M}_k$ *can not change its state.*

This idea is formalised by means of the following two definitions.

Definition 2.5.1 (Competition) *Let I be an index set. For each k, let* $A_{k,i}, i \in I$*, be a set of mutually exclusive sets such that* $\emptyset \neq A_{k,i} \subset S_k$ *and* $\bigcup_{i \in I} A_{k,i} = S_k$*,* $k = 1, \ldots, K$*. CTMC k uses resource i if the CTMC is in state* $\bar{n}_k \in A_{k,i}$*. CTMCs* k_1 *and* k_2 *compete over resource i if* $\{\bar{n}_{k_1}, \bar{n}_{k_2} : n_{k_1} \in A_{k_1,i}, n_{k_2} \in A_{k_2,i}\} = \emptyset$*. Let* $C_{ki} \subset \{1, \ldots, K\}$ *be the CTMCs that compete over resource i with CTMC k.*

Definition 2.5.2 (Boucherie product process) *The CTMC on state space*

$$S = \prod_{k=1}^{K} S_k \tag{2.5.1}$$

with transition rates

$$q(n, n') = \sum_{k=1}^{K} q_k(n_k, n'_k) \prod_{\ell=1, \ell \neq k}^{K} \mathbf{1}(n_\ell = n'_\ell)\mathbf{1}(\textit{if } n_\ell \in A_{\ell i} \textit{ then } k \notin C_{\ell i})$$ [2]

[2] $\mathbf{1}$ is an indicator function: the value of $\mathbf{1}$(*statement*) is 1 if the statement is true and 0 otherwise.

where $n = (n_1, \dots, n_K)$*,* $n' = (n'_1, \dots, n'_K)$*, is called the* product process *of the collection of CTMCs* $1, \dots, K$*, competing over resources* I*.*

From the transition rate definition it should be noted that a move along dimension k is not allowed whenever component k is competing for a resource i with a component ℓ which actually holds i in the a current state of the process. On the other hand if k is not competing for any of the resources detained by any other component in the current state, then a move along k dimension is permitted. Summarising: a process k in the Boucherie framework is *blocked* in any state where at least one among the resources it is competing on is occupied, while it is *free* to move if none of them is busy..

Two components Boucherie product process. In the simplest case, which is what we consider in the remaining of this thesis, the Boucherie CTMC $\mathcal{M}$ consists of two sub-processes, $\mathcal{M}_1$ and $\mathcal{M}_2$, and two notional resources. There is no competition over the first resource, but the two processes compete over the second resource, which is denoted by R. The competition over R has the effect of partitioning the state space of each component process. If $\mathcal{M}_k$ ($k = \{1,2\}$) has state space S_k, then $S_k = S_{k,\overline{R}} \cup S_{k,R}$, where $S_{k,\overline{R}}$ denotes the set of states in which the resource is not needed, while $S_{k,R}$ denotes the set of states in which the resource is used. Figure 2.6 shows the areas which the product process is partitioned in: $R_{free} = S_{1,\overline{R}} \times S_{2,\overline{R}}$ denotes the area where neither process is using the resource; $R_1 = S_{1,R} \times S_{2,\bar{R}}$ is the area where $\mathcal{M}_1$ is using R; whilst in area $R_2 = S_{1,\overline{R}} \times S_{2,R}$, $\mathcal{M}_2$ is using R.

Transition rates of the product process imply that only one process can change its state, and that process 1 is stopped whenever process 2 holds the resource and vice versa. This is depicted in figure 2.6 where the lines indicate the direction along with transitions can occur. As a consequence of this stopping mechanism the region $A_{12} \times A_{22}$, also denoted $S_{1R} \times S_{2R}$, can not be entered. Therefore in the definition of the Boucherie product process, the state space 2.5.1 can be replaced by $S = S_1 \times S_2 / S_{1R} \times S_{2R}$ and in general:

$$S = \prod_{k=1}^{K} S_k \setminus \left(\prod_{k=1}^{K} \prod_{i \in I} \prod_{j \in C_{ki}} A_{k,i} \times A_{j,i} \right) \tag{2.5.2}$$

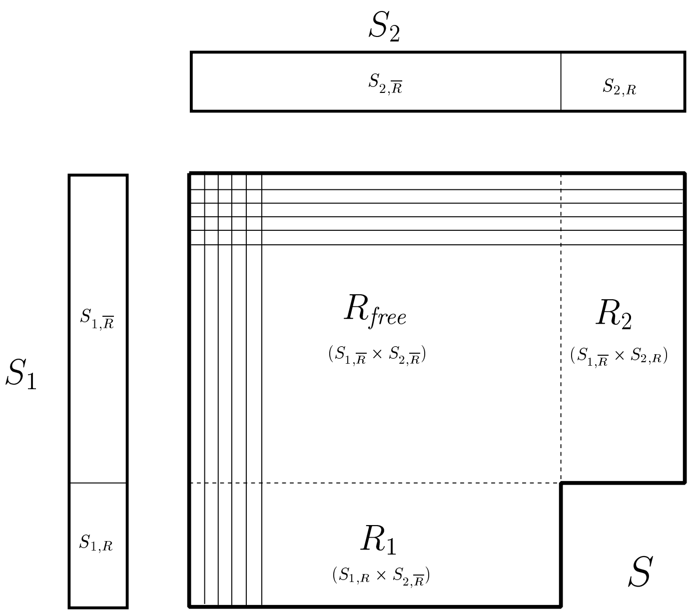


Figure 2.6: A two component Boucherie product process state-space.

Remark 2.5.1 (Trivial case) *For $I = \{1\}$ (i.e. a single resource is considered) we have $A_{k1} = S_k$. In this case the Markov chains are independent. In such a case either $\{\bar{n}_{k_1}, \bar{n}_{k_2} : \bar{n}_{k_1} \in A_{k_1,1}, \bar{n}_{k_2} \in A_{k_2,1}\} = \emptyset$ for some pair (k_1, k_2) or $\{\bar{n}_{k_1}, \bar{n}_{k_2} : \bar{n}_{k_1} \in A_{k_1,1}, \bar{n}_{k_2} \in A_{k_2,1}\} \neq \emptyset$ for all (k_1, k_2). In the first case $S = \emptyset$ and the Boucherie product process is not defined; in the second case all Markov chains operate without influence of each others.*

Theorem 2.5.1 (Product-form distribution) *The product process of the collection of CTMCs $1, \dots K$ competing over resources I and with state space S defined as 2.5.2 and transition rates as in definition 2.5.2, has equilibrium distribution π given by*

$$\pi(n) = G \prod_{k=1}^{K} \pi_k(n_k) \qquad n \in S$$

where G is a normalising constant, determined by the exact form of S, and $\pi_k(\cdot)$ is the equilibrium distribution of process S_k.

Proof. see [7] $\square$

The result of theorem 2.5.1 holds because each process can either operate independently of the other processes or it is blocked. For all $n \in S$, if process ℓ is in state n_ℓ and $\ell \neq k$ then process k either carries out a transition which is not in competition with ℓ with respect to resource i ($1(\text{if } i : n_\ell \in S_{\ell,i} \text{ then } k \notin C_{\ell i}) = 1$) or process k wants to access the resource which ℓ occupies ($1(\text{if } i : n_\ell \in S_{\ell,i} \text{ then } k \notin C_{\ell i}) = 0$). In either case process k will satisfy its own global balance equations:

$$\sum_{\bar{n}'_k \in S_k} \{\pi_k(\bar{n}_k) q_k(\bar{n}_k, \bar{n}'_k) - \pi_k(\bar{n}'_k) q_k(\bar{n}'_k, \bar{n}_k)\} = 0, \;\; \bar{n}_k \in S_k$$

these equations are trivially satisfied when the process is stopped and also true when the process is operating independently. It appears that the exclusion principle maintained by the transition rates of the product process imposes a protocol on the behaviour of the product process that ensures that the CTMCs in the collection behave as if they are independent. For any process k, $1 \leq k \leq K$, if the current state is in the subset $A_{k,i}$ it signifies that the process is presently using the resource i and no other process j, such that $j \in C_{ki}$, can gain access to i and enter its subset of states $A_{j,i}$. Thus the competition and the sets C_{ki} define areas of the state space of the product process which are inaccessible. The transition rates of the product process are defined in a way which ensures this exclusion.

2.5.1 The running example

In the following we introduce a practical example of a two component Boucherie product process. This will be our running example, which we will exploit to show examples of decompositional model checking throughout the remaining of this work.

Example 2.5.1 (Geographical Information System (GIS)) *Let us consider a navigational device consisting of a pair of sensors which maintain complementary data about geographical location. In order to keep the sensors' internal data structures in complementary states, they share data via a register they need to access in a mutually exclusive fashion. Each sensor gains access to the register and locks it while it reads the current data value; it then uses this information to adjust the equipment it controls while also recalculating a value for the shared register based on its own internal data* *structures. It then updates the value in the register and releases it. In addition, sensor 1 maintains an external monitor and will periodically gather data from this monitor and use it to recalculate its internal data structures.*

Each sensor consists of two components: one responsible for resetting the equipment during each cycle, and one responsible for carrying out the data recalculation. In sensor 1, the recalculation component is assumed to be also responsible for interaction with the monitor. Sensors have a cyclic behaviour characterised by the sequence of "actions": idle-reading-resetting/recalculating , where resetting and recalculating are simultaneous. In addition sensor 1 can be involved in gathering information from its monitor, whenever it is idle.

Such a framework represents an example of a two component Boucherie process where the two sensors are the processes competing over the shared register. The labelled CTMCs representing the two sensors are shown in Figure 2.7. They exhibit similar behaviours except for the gather action (state s_{11}) which only sensor 1 can be involved in. States are labelled with elements of the atomic proposition sets $AP_1 = \{idle_1, read_1, res_1, rec_1, gat_1\}$ and $AP_2 = \{idle_2, read_2, res_2, rec_2\}$, representing the actions each sensor is involved in.

From the starting state $s_{10}(s_{20})$, where it is idle, the first(second) sensor reads data at rate r_1, entering state $s_{12}(s_{21})$. Alternatively, sensor 1 only, can gather data from the monitor at rate r_5, entering state s_{11}. When a sensor reads data it acquires the resource (the register). Once it has read the data one of its component recalculates (rate r_2) while the other resets (rate r_4). When both are ready the data is updated (rate r_3) and the register is released. When data has been gathered from the monitor (sensor 1 only) a recalculation is necessary (rate r_2) before returning to the initial state. The state space of each CTMC can be partitioned into two subsets: $S_{1\overline{R}} = \{s_{10}, s_{11}\}$ (no resource held by sensor 1) and $S_{1R} = \{s_{12}, s_{13}, s_{14}, s_{15}\}$, being the partition for $\mathcal{M}_1$ and $S_{2\overline{R}} = \{s_{20}\}$ and $S_{2R} = \{s_{21}, s_{22}, s_{23}, s_{24}\}$, being the partition for $\mathcal{M}_2$.

Figure 2.8 depicts the Boucherie process, obtained by composition of sensors' CTMCs. States of $S_{1R} \times S_{2R}$ are not part of the Boucherie process as a consequence of the competition over the shared register R. The three areas, R_{free}, R_1 and R_2 which the state space S is partioned in, are pointed out: the register is not hold in any state of

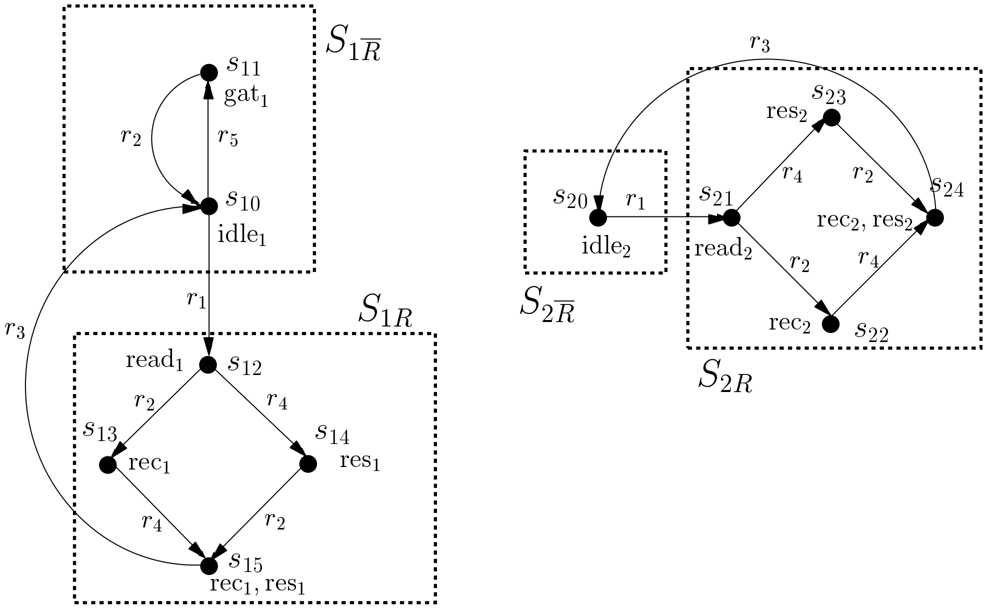


Figure 2.7: State space of the two components $\mathcal{M}_1$ and $\mathcal{M}_2$,showing the states labelling

region R_{free}, whilst it is detained by sensor 1, in any state of region R_1 and by sensor 2, in any state of region R_2.

Considering the sensors in isolation we can deduce that their equilibrium probability distributions. The steady state distribution for sensor 1 is:

$$
\begin{aligned}
\pi_1(s_{10}) &= r_2 r_3 r_4 (r_2 + r_4)/G_1 \\
\pi_1(s_{12}) &= r_1 r_2 r_3 r_4 /G_1 \\
\pi_1(s_{13}) &= r_1 r_2^2 r_3 /G_1 \\
\pi_1(s_{14}) &= r_1 r_3 r_4^2 /G_1 \\
\pi_1(s_{15}) &= r_1 r_2 r_4 (r_2 + r_4)/G_1 \\
\pi_1(s_{11}) &= r_3 r_4 r_5 (r_2 + r_4)/G_1
\end{aligned}
$$

where $G_1 = (r_2 r_4 + r_4^2)(r_1 r_2 + r_1 r_3 + r_2 r_3 + r_3 r_5) + r_1 r_2^2 r_3$, while the steady state dis-

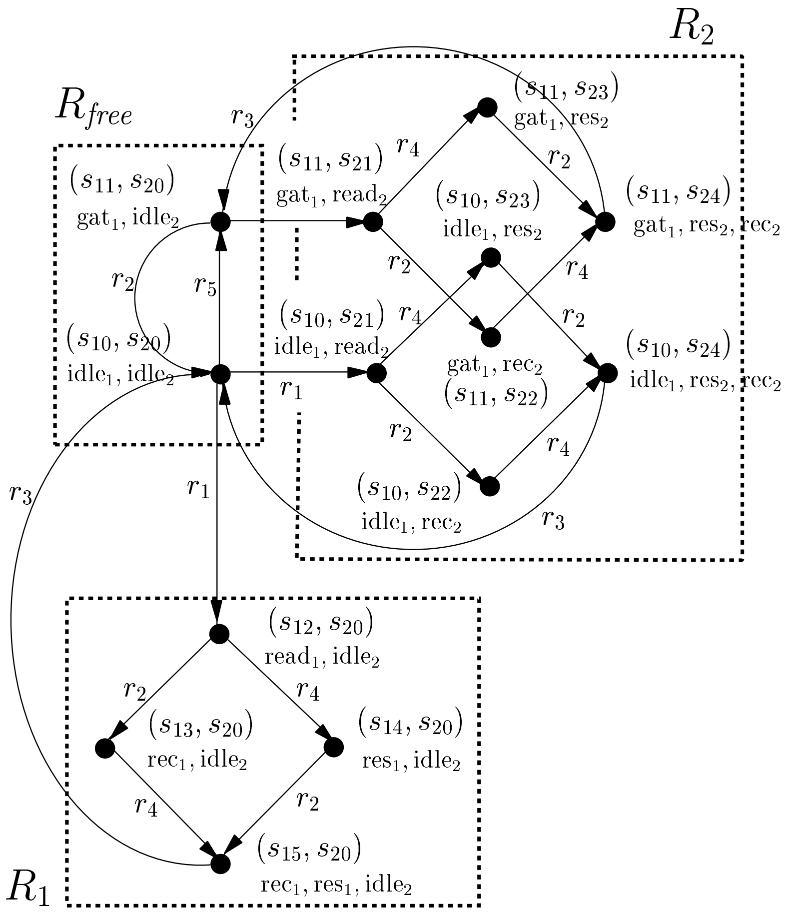


Figure 2.8: State space of the product process $\mathcal{M}$

tribution for sensor 2 is:

$$\begin{aligned}
\pi_2(s_{20}) &= r_3 r_2 r_4 (r_2 + r_4)/G_2 \\
\pi_2(s_{21}) &= r_1 r_3 r_2 r_4/G_2 \\
\pi_2(s_{22}) &= r_1 r_3 r_2^2/G_2 \\
\pi_2(s_{23}) &= r_1 r_3 r_4^2/G_2 \\
\pi_2(s_{24}) &= r_1 r_2 r_4 (r_2 + r_4)/G_2
\end{aligned}$$

where $G_2 = (r_2 r_4 + r_4^2)(r_1 r_3 + r_1 r_2 + r_3 r_2) + r_1 r_3 r_2^2$.

As a consequence of theorem 2.5.1, the equilibrium distribution of the Boucherie process can straightforwardly be derived from the sensors' ones.

Chapter 3

On CSL Expressiveness

3.1 Introduction

The syntax and semantics of the CSL logic, as described in the previous chapter, provide the user with powerful means to state properties concerning CTMCs. Steady-state and transient analysis of the system can be performed in terms of CSL state formulae, as well as analysis of path-based properties. However there are some non-obvious features of the CSL semantics which need to be addressed.

The temporal operators Next and Until allow us to refer to the future behaviour of the system, though, in that respect, they feature different capabilities. A study of their expressiveness is addressed in Section 3.2, where the idea of *time quantification* as opposes to *step quantification*, as a feature of a temporal connective, is presented. As a result a step-bounded version of the Until operator is defined.

Section 3.3 concerns the study of some relevant consequences of the semantics of CSL time bounded path formulae. The characterisation of sensible probabilistic formulae is faced in Section 3.4, where the definition of *well-formed* probabilistic formulae is provided. The effects of the semantics of CSL *steady-state* formulae with respect to ergodic models, is addressed in Section 3.5, where semantic equivalences for formulae involving the *steady-state* operator are found, leading to a modified, but equivalent, CSL syntax to refer to ergodic CTMCs.

3.2 Extending the Until expressiveness

In this section some considerations regarding the expressiveness capability of the two basic temporal connectives, Next and Until, are presented.

The main characteristic of temporal logics (i.e. LTL, CTL, PCTL, CSL), as a means for specifying properties of a system, is that they allow one to refer to future evolutions (i.e. paths) of a system, as "criteria" for selecting the states of interest. This is achieved by means of two temporal connectives, namely Next and Until. Generally speaking, with discrete-events state-based systems two types of quantification with respect to the future appear to be sensible: a *time quantification*, by means of which the evolution of the system is considered with respect to time elapsing, as opposed to a *step quantification* or *event quantification*, through which the system's evolution is considered with respect to events' occurrence. Clearly *time quantification* is sensible only when elapsed time is captured in the modelling framework[1]. We observe that if time is seen as a discrete quantity in the modelling framework, then, usually, *time elapsing* and *event elapsing* coincide (the occurence of an event is assumed to "consume" one time unit). Thus, when referring to such models (e.g. DTMCs), *time quantification* and *event quantification* have the same meaning.

Both Next and Until allow us to refer to the future but with some differences. Referring to their original (untimed) version, some observations can be made. The Until operator $\mathcal{U}$ (and its derivative *sometime in the future*, i.e. $\diamond\phi \equiv (tt\,\mathcal{U}\phi)$) permits one to refer to features of the system's behaviour which one is interested in observing in an *indefinitely long future* (i.e. Until does not naturally imply any sort of quantification, neither *time* nor *step*). Conversely, the Next operator X naturally implies a (very strict *one-transition* only) *step quantification*: $(X\phi)$ identifies those evolutions for which ϕ happens to be true after exactly one transition from the present state.

When referring to *timed* models, either *discrete-time* (e.g. DTMCs) or *continuous-time* (e.g. CTMCs), *time quantification* can be sensible.

In the PCTL logic (the temporal logic for DTMCs), an *event-bounded* version of the Until operator ($\phi\ \mathcal{U}^{\leq n}\ \psi$) allows one to specify an upper bound (i.e. n) for the

[1] We will name *timed* those modelling frameworks which capture time elapsing and *untimed* those other ones which do not capture time elapsing.

number of transitions/time instants within which the Until formulahas to be satisfied.

Conversely, when the time is continuous (as with CTMCs), *time quantification* and *event quantification* are distinct. In the CSL logic, a time-bounded version of both Next and Until is provided. It basically allows us to associate a *continuous-time quantification* with the usual semantics. As a result, a time-bounded Until formula $(\phi\ \mathcal{U}^{[a,b]}\ \psi)$ allows to refer to a behaviour of interest which has to happen in a *time-wise definite* but *step-wise indefinite* future (i.e. a combined *time/event quantification* is not supported by the time-bounded CSL Until operator). On the other hand, the time-bounded Next, incorporates both types of quantification: a formula like $(X^{[a,b]}\phi)$ characterises those evolutions such that ϕ happens to be verified in a *time-distance* $I = [a,b]$ from the starting instant of observation (i.e. the time a state is entered) and also within a (strict) step-distance of exactly one transition. In this sense, the time-bounded Next allows for a strict, combined *time-step quantification* of the future, where the only possible value for the *step quantification* is 1.

In the reminder of this section the definition of *event-bounded* Until is provided. It extends the one which can be found in [19] by allowing the specification of a bounding interval $\{n_1, n_2\}$ with a positive, possibly non-null infinum. We will prove that, contrary to the "standard" *event-bounded* Until ([19]) which refers to bounding intervals like $\{0, n\}$ only, the probability to satisfy an *event-bounded* Until which refers to a single-point bounding interval (i.e. $n_1 = n_2$), can be computed as a function of the *transition probability matrix* **P** rather than as a function of a tailored matrix **M** derived from **P**.

Definition 3.2.1 (event-bounded Until) *Let ϕ' and ϕ'' be two CSL state formulae, n_1, n_2 two natural numbers with $n_1 \leq n_2$ and σ a path on a given CTMC. The step-bounded Until formula $(\phi'\ \mathcal{U}_{\{n_1,n_2\}}\ \phi'')$ is a path-formula whose semantics, with respect to σ, is defined as:*

$$\sigma \models (\phi'\ \mathcal{U}_{\{n_1,n_2\}}\ \phi'') \iff \exists i,\ n_1 \leq i \leq n_2 : \sigma[i] \models \phi'' \wedge (j < i) \Rightarrow \sigma[j] \models \phi'$$

If $n_1 = n_2 = n$ the notation $(\phi'\ \mathcal{U}_{\{n\}}\ \phi'')$ is used.

We recall here that the probability of satisfying an Until formula from a state s of a

CTMC $\mathcal{M}$ is given by the probability of each path that, starting at s, satisfies the Until formula, which is: $Prob^{\mathcal{M}}(s, \phi\, \mathcal{U}_{\{n\}}\, \psi) = Pr_s\{\sigma \in Path(s) : \sigma \models \phi\ \mathcal{U}_{\{n\}}\ \psi\}$. We denote $Path(s, \phi\ \mathcal{U}_{\{n\}}\ \psi)$ the set of such paths.

The following theorem provides a method for computing the probability measure of satisfying a step-bounded *sometime in the future* formula, $\diamond_{\{n\}}\ \phi \equiv (tt\ \ \mathcal{U}_{\{n\}}\ \phi)$ from a state s.

Theorem 3.2.1 (event-bounded Diamond) *For a formula* $\diamond_{\{n\}}\ \phi \equiv (tt\ \ \mathcal{U}_{\{n\}}\ \phi)$ *and a state* s *of a CTMC,*

$$Prob(s, \diamond_{\{n\}}\ \phi) = \begin{cases} \underline{i}_\phi(s) & \textbf{if}\ \ n = 0 \\ \\ Prob(s, X\ \phi) & \textbf{if}\ \ n = 1 \\ \\ \sum_{s' \in S} \mathbf{P}(s, s') \cdot Prob(s', \diamond_{\{n-1\}}\ \phi) & \textbf{if}\ \ n > 1 \end{cases}$$

where $\underline{i}_\phi(s) = 1$ if $s \models \phi$ and $\underline{i}_\phi(s) = 0$, if $s \not\models \phi$.

Proof. We denote $Path(s, \diamond_{\{n\}}\phi)$ the paths starting at s and satisfying $\diamond_n\phi$ and $Prob(s, \diamond_{\{n\}}\phi)$ the measure of their probability. Similarly $Path(s, X\phi)$ denotes the paths starting at s and satisfying $X\phi$ and $Prob(s, X\phi)$ the measure of their probability. Let us consider the different cases.

Case 1. $n = 0$. In this case either every path starting at s satisfies $\diamond_{\{0\}}\phi$ or none. If $s \models \phi$ then $Path(s, \diamond_{\{0\}}\phi) = Path(s)$, hence $Prob(s, \diamond_{\{0\}}\phi) = 1 = \underline{i}_\phi(s)$. If $s \not\models \phi$ then $Path(s, \diamond_{\{0\}}\phi) = \emptyset$, hence $Prob(s, \diamond_{\{0\}}\phi) = 0 = \underline{i}_\phi(s)$.

Case 2. $n = 1$. Trivially $Path(s, \diamond_{\{1\}}\phi) = Path(s, X\phi)$ hence $Prob(s, \diamond_{\{1\}}\phi) = Prob(s, X\phi)$.

Case 3. $n > 1$. In this case $Path(s, \diamond_{\{n\}}\phi)$ consists of all those paths σ of the form $s \to \sigma'$, where $\sigma' \in Path(s', \diamond_{\{n-1\}}\phi)$ (i.e. all those paths whose *first order* suffix satisfies ϕ in $n-1$ steps, $(1 \uparrow \sigma) \models \diamond_{\{n-1\}}\phi$). Hence

$$Prob(s, \diamond_{\{n\}}\phi) = \sum_{s' \in S} \mathbf{P}(s, s') \cdot Prob(s', \diamond_{\{n-1\}}\ \phi)$$

$\square$

Corollary 3.2.1 **($\underline{Prob}(\diamond_{\{n\}}\,\phi)$)** *For a formula $\diamond_{\{n\}}\,\phi$ and a CTMC $\mathcal{M}$ the state vector $\underline{Prob}(\diamond_{\{n\}}\,\phi)$ is given by:*

$$\underline{Prob}(\diamond_{\{n\}}\,\phi) = \begin{cases} \underline{i}_{\phi} & \textbf{if } n = 0 \\ \underline{Prob}(X\ \phi) & \textbf{if } n = 1 \\ \mathbf{P} \cdot \underline{Prob}(\diamond_{\{n-1\}}\,\phi) & \textbf{if } n > 1 \end{cases}$$

Proof. Straightforward consequence of Theorem 3.2.1.

Corollary 3.2.1, shows that the computation of the probability state vector for a step-bounded *sometime in the future* formula results in an iterative matrix-vector product. In the next theorem a way of computing the probability measure for a step-bounded Until formula, $(\phi\ \mathcal{U}_{\{n\}}\ \psi)$ is described. It is based on the result for the step-bounded *sometime in the future* operator.

Theorem 3.2.2 (event-bounded Until) *For a formula $(\phi\ \mathcal{U}_{\{n\}}\ \psi)$ and a state s of a CTMC, the following holds:*

$$Prob(s,(\phi\ \mathcal{U}_{\{n\}}\ \psi)) = \begin{cases} \underline{i}_{\psi}(s) & \textbf{if } n = 0 \\ \underline{i}_{\phi}(s) \cdot Prob(s, X\ \psi) & \textbf{if } n = 1 \\ \underline{i}_{\phi}(s) \cdot \sum_{s' \in S} \mathbf{P}(s,s') \cdot Prob(s',(\phi\ \mathcal{U}_{\{n-1\}}\ \psi)) & \textbf{if } n > 1 \end{cases}$$

Proof. Let us consider the different cases.

Case 1. $n = 0$. In this case either every path starting at s satisfies $(\phi\ \mathcal{U}_{\{0\}}\ \psi)$ or none. If $s \models \psi$ then $Path(s,(\phi\ \mathcal{U}_{\{0\}}\ \psi)) = Path(s)$, hence $Prob(s,(\phi\ \mathcal{U}_{\{0\}}\ \psi)) = 1 = \underline{i}_{\psi}(s)$. If $s \not\models \psi$ $Path(s,(\phi\ \mathcal{U}_{\{0\}}\ \psi)) = \emptyset$, hence $Prob(s,(\phi\ \mathcal{U}_{\{0\}}\ \psi)) = 0 = \underline{i}_{\psi}(s)$.

Case 2. $n = 1$. The set $Path(s, (\phi\ \mathcal{U}_{\{1\}}\ \psi))$ is either equal to $Path(s, X\psi)$ or empty. If $s \models \phi$ then $Path(s, (\phi\ \mathcal{U}_{\{1\}}\ \psi)) = Path(s, X\psi)$. Hence, in this case,

$$Prob(s, (\phi\ \mathcal{U}_{\{1\}}\ \psi)) = Prob(s, X\psi) = \underline{i}_\phi(s) \cdot Prob(s, X\psi)$$

If $s \not\models \phi$, then $Path(s, (\phi\ \mathcal{U}_{\{1\}}\ \psi)) = \emptyset$. Hence, in this case,

$$Prob(s, (\phi\ \mathcal{U}_{\{1\}}\ \psi)) = 0 = \underline{i}_\phi(s) \cdot Prob(s, X\psi)$$

Case 3. $n > 1$. The set $Path(s, (\phi\ \mathcal{U}_{\{n\}}\ \psi))$ is either empty, if $s \not\models \phi$, or it consists of all those paths σ of the form $s \to \sigma'$, where $\sigma' \in Path(s', (\phi\ \mathcal{U}_{\{n-1\}}\ \psi))$, if $s \models \phi$. Hence, $Prob(s, (\phi\ \mathcal{U}_{\{n\}}\ \psi)) = \underline{i}_\phi \cdot [\sum_{s' \in S} \mathbf{P}(s, s') \cdot Prob(s', (\phi\ \mathcal{U}_{\{n-1\}}\ \psi)))]$.

□

Corollary 3.2.2 ($\underline{Prob}(\phi\ \mathcal{U}_{\{n\}}\ \psi)$) *For a formula $(\phi\ \mathcal{U}_{\{n\}}\ \psi)$ and a CTMC $\mathcal{M}$ the state vector $\underline{Prob}(\phi\ \mathcal{U}_{\{n\}}\ \psi)$ is given by:*

$$\underline{Prob}(\phi\ \mathcal{U}_{\{n\}}\ \psi) = \begin{cases} \underline{i}_\psi & \textbf{if } n = 0 \\ \underline{i}_\phi \cdot \underline{Prob}(X\ \psi) & \textbf{if } n = 1 \\ \underline{i}_\phi \cdot [\mathbf{P} \cdot \underline{Prob}(\phi\ \mathcal{U}_{\{n-1\}}\ \psi)] & \textbf{if } n > 1 \end{cases}$$

Proof. Straightforward consequence of Theorem 3.2.2.

Corollary 3.2.2, shows that, also for a step-bounded Until formula, the computation of the probability state vector results in an iterative matrix-vector product.

In this section an event-bounded version of the Until operator has been formally introduced. It allows us to specify a bounding interval in terms of number of events in the executions fulfilling an Until-like property. It has been shown that the computation of the probability measure for an event-bounded Until, can be obtained via an iterative

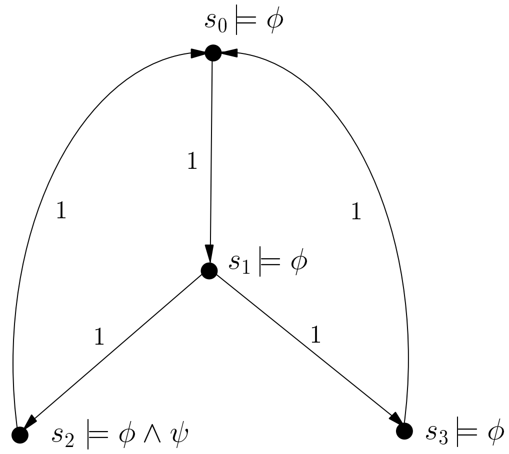


Figure 3.1: A simple arbitrary CMTC $\mathcal{M}$

matrix-vector multiplication. This proves that, differently from its event-unbounded counterpart, the model checking problem for event-bounded Until does not require the solution of a system of linear equations.

Example 3.2.1 (event-bounded Until) *Figure 3.1 illustrates a simple four-states CMTC, with transitions probability matrix* **P***:*

$$\mathbf{P} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 0 & 0 & \frac{1}{2} & \frac{1}{2} \\ 1 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{pmatrix}$$

Let ϕ *and* ψ *be two CSL state-formulae and let us assume that* ψ *is satisfied only in the state* s_2*, while* ϕ *is valid in every state. As a result:*

$$\underline{i}_\phi = (1,1,1,1) \qquad \underline{i}_\psi = (0,0,1,0)) \qquad \underline{Prob}(X\ \psi) = (0,\frac{1}{2},0,0)$$

By application of Corollary 3.2.2, the probability state-vectors $\underline{Prob}(\phi\ \mathcal{U}_{\{n\}}\ X\ \psi)$*, for*

$n \in \mathbb{N}$, can be straighforwardly derived.

$$\begin{aligned}
\underline{Prob}(\phi\ \mathcal{U}_{\{0\}}\ \ \psi) &= (0,0,1,0) \\
\underline{Prob}(\phi\ \mathcal{U}_{\{1+3k\}}\ \ \psi) &= (0,\frac{1}{2},0,0) \\
\underline{Prob}(\phi\ \mathcal{U}_{\{2+3k\}}\ \ \psi) &= (\frac{1}{2},0,0,0) \\
\underline{Prob}(\phi\ \mathcal{U}_{\{3+3k\}}\ \ \psi) &= (0,0,\frac{1}{2},\frac{1}{2})
\end{aligned}$$

with $k \in \mathbb{N}$. We observe that, for example, the probability of fulfilling the Until formula in $n = 5$ steps is non-null only for state s_0. This is, in fact, correct, as s_0 is the only state admitting some (in this case two, $s_0, s_1, s_2, s_0, s_1, s_2$ and $s_0, s_1, s_3, s_0, s_1, s_2$) five step paths satisfying $(\phi\ \mathcal{U}\ \ \psi)$. Furthermore the probabilty of each one such a path (which is given by multiplying the probability of each step) is actually $\frac{1}{4}$, hence their sum is $\frac{1}{2}$. The correctness of the other cases can be easily verified in a similar way. □

3.3 Semantics of single-point bounded path formulae

Proposition 2.3.2 and Theorem 2.3.1 allow for the computation of the probability measure of paths which, starting from a given state s, satisfy, respectively, a bounded Next, and a bounded Until formula. The form of the bounding interval $I = [a,b]$, either single-point ($a = b$) or multiple-points ($a > b$), affects the probability measure of paths satisfying a bounded Next and Until formulae.

Bounded Next. The probability measure of each timed path starting at s and satisfying the Next formula $(X\ \phi)$ within the time boundaries I, is given by the result of Proposition 2.3.2. The following remark points out a peculiarity which arises when the bounding interval I consists of a single time instant.

Remark 3.3.1 *Let $\mathcal{M} = (S, \mathbf{Q}, L)$ be a labelled CTMC and ϕ a CSL state formula. Whenever the considered time interval consists of a single point $I = \{a\}$, the probability measure $Prob(s, X^I\phi) = 0$, independently of the state $s \in S$ and of the formula ϕ.*

Proof. By substitution of $a = b$ in the result of Proposition 2.3.2.

□

Remark 3.3.1 highlights the fact that reaching a ϕ-state in one step from a state s exactly at time $t = a$ is an impossible event. This is consistent with the fact that the delay of any transition $s \rightarrow s'$ in a CTMC is an exponentially distributed, hence continuous, random variable X; thus its probability of assuming any specific value is zero (i.e. $Pr[X = a] = 0$, for all $a \in \mathbb{R}$).

Definition 3.3.1 (Well-formed bounded Next) *Let ϕ be a CSL state formula and $I = [a, b] \subseteq R_{\geq 0}$ a bounding time interval, then the bounded Next formula $\varphi \equiv X^I \phi$ is said to be well-formed if and only if $a < b$.*

Definition 3.3.2 (Well-formed path formulae) *A CSL time bounded path formula φ is said to be well-formed if, in case it is a bounded Next formula it is also well-formed.*

Definitions 3.3.1 and 3.3.2 allow us to rule out the bounded Next formulae which cannot be satisfied in any state of the model.

Bounded Until. The value $Prob(s, \phi' \mathcal{U}^I \phi'')$, defined by means of Theorem 2.3.1, represents the measure of the probability of each timed path starting at s and satisfying the Until formula $(\phi' \; \mathcal{U} \, \phi'')$ within the time boundaries specified by I. Any timed path $s_0 \xrightarrow{t_0} s_1 \xrightarrow{t_1} s_2 \ldots \xrightarrow{t_{n-1}} s_n \xrightarrow{t_n} \ldots$ belongs to the set characterised by its untimed *embedded generator* $\overline{\sigma} = s_0 \longrightarrow s_1 \longrightarrow s_2 \ldots \longrightarrow s_n \longrightarrow \ldots$. In a way, we can say that any timed path σ is generated by its untimed *embedded generator*.

As Remark 2.3.1 points out, whenever a timed path σ satisfies a bounded Until $(\phi \; \mathcal{U}^I \; \psi)$, then its untimed embedded generator $\overline{\sigma}$ satisfies the correspondent unbounded Until $(\phi \; \mathcal{U} \, \psi)$.

In general, the set of paths satisfying an untimed Until formula $(\phi' \; \mathcal{U} \, \phi'')$ can be partitioned by distinguishing between those paths having a future state which satisfies the *target* ϕ'' but not the *premise* ϕ' $(Path(s, (\phi' \wedge \neg\phi'') \mathcal{U} \; (\neg\phi' \wedge \phi'')))$ and those which allow for a future state satisfying both the *target* and the *premise* of the until ($Path(s, (\phi' \wedge \neg\phi'')\ \mathcal{U}\ (\phi' \wedge \phi''))$).

$$Path(s, \phi' \mathcal{U}\ \phi'') = Path(s, (\phi' \wedge \neg\phi'') \mathcal{U}\ (\neg\phi' \wedge \phi'')) \cup Path(s, (\phi' \wedge \neg\phi'')\ \mathcal{U}\ (\phi' \wedge \phi''))$$

The characterisation of that partition of $Path(s, \phi' \mathcal{U}\ \phi'')$, allows us to formulate the following remark which considers a subtlety implied by the result of Theorem 2.3.1.

Remark 3.3.2 *Let* $(\phi'\ \mathcal{U}^I\ \phi'')$ *be a CSL bounded Until formula with* $I = [a,b]$.

$a < b$ **(multi-points interval):** *the timed paths contributing with a non-null value to the measure* $Prob(s, \phi' \mathcal{U}^I\ \phi'')$ *can be generated by untimed paths of both* $Path(s, (\phi' \wedge \neg\phi'') \mathcal{U}\ (\neg\phi' \wedge \phi''))$ *and* $Path(s, (\phi' \wedge \neg\phi'')\ \mathcal{U}\ (\phi' \wedge \phi''))$.

$a = b$ **(single-point interval):** *the timed paths contributing with a non-null value to the measure* $Prob(s, \phi' \mathcal{U}^I\ \phi'')$ *can be generated by untimed paths of* $Path(s, (\phi' \wedge \neg\phi'')\ \mathcal{U}\ (\phi' \wedge \phi''))$ *only.*

The next example shows what Remark 3.3.2 is meant to point out.

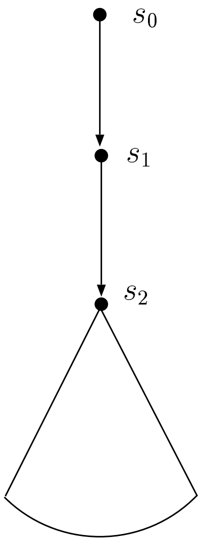
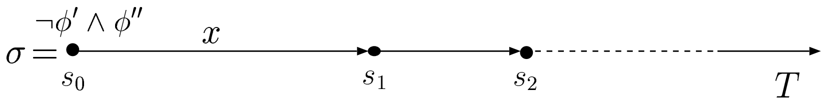


Figure 3.2: Paths unravelling from state s_0

Example 3.3.1 *Let us consider a state s_0 of a labelled CTMC $\mathcal{M}$ and the set of paths starting at s_0 (Figure 3.2 depicts the unravelling of the paths starting at s_0). For the sake of simplicity, we are assuming that s_0 has a single successor state, s_1, which, itself, has a single successor, namely s_2 (i.e. both s_0 and s_1 are states where no competition takes place). As a result the tree representing the unravelling of paths starting at s_0 is given by appending the tree representing the unravelling of paths from s_2 to the finite path $s_0 \to s_1 \to s_2$ (see Figure 3.2).*



$i)\ Prob(s_0, \phi' \mathcal{U}^{[0,b]} \phi'') = 1 \qquad ii)\ Prob(s_0, \phi' \mathcal{U}^{[0.5,b]} \phi'') = 0$

Figure 3.3: Probability measure of timed-until path: case $s_0 \models \neg\phi' \wedge \phi''$

Suppose we are interested in evaluating the probability measure for the time bounded Until formula $(\phi'\ \mathcal{U}^I\ \phi'')$, with respect to s_0. Let us consider different assumptions corresponding to every case of the definition of $Prob(s, \phi'\ \mathcal{U}^I\ \phi'')$ of Theorem 2.3.1.

i) $s_0 \models \neg\phi' \wedge \phi''$.

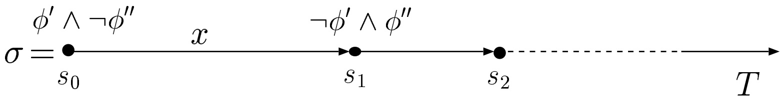
Figure 3.3 shows a path starting at s_0. It is assumed that s_0 is such that the target ϕ'' *is satisfied but not the* premise ϕ'. *If we are considering a lower bound of zero ($a = 0$), then the first case of equation 2.3.1 does apply, thus $Prob(s_o, \phi'\, \mathcal{U}^{[0,b]}\ \phi'') = 1$. On the other hand whenever* $\inf(I)$ *is greater than zero ($a > 0$), the "otherwise" case of equation 2.3.1 applies hence, for example, $Prob(s_o, \phi'\, \mathcal{U}^{[0.5,b]}\ \phi'') = 0$.*

ii) $s_0 \models \phi' \wedge \neg\phi''$ **and** $s_1 \models \neg\phi' \wedge \phi''$.

In Figure 3.4 s_0 is assumed to satisfy ϕ' but not ϕ'', hence the second case of equation 2.3.1 applies. Moreover s_1, the only successor of s_0, is assumed to satisfy ϕ'' but not ϕ'. A distinction between multiple-point intervals (i.e. $a < b$) and single-point intervals (i.e. $a = b > 0$) needs to be made.

- *$a < b$ (multiple-points interval). From equation 2.3.1 we have that*

$$Prob(s_o, \phi'\, \mathcal{U}^{[a,b]}\ \phi'') = \int_0^b \mathbf{Q}(s_0, s_1) \cdot e^{-E(s_o)x} \cdot Prob(s_1, \phi'\, \mathcal{U}^{[a \ominus x, b \ominus x]}\ \phi'')dx$$



$$i)\ Prob(s_0, \phi' \mathcal{U}^{[a,b]} \phi'') = \int_a^b \mathbf{P}(s_0, s_1) \cdot e^{-E(s_0)x} dx$$

$$ii)\ Prob(s_0, \phi' \mathcal{U}^{[a,a]} \phi'') = 0$$

Figure 3.4: Probability measure of timed-until path: case $s_0 \models \phi' \wedge \neg\phi''$

The above integral can be split into the sum of two integrals, resulting in

$$Prob(s_o, \phi' \, \mathcal{U}^{[a,b]} \, \phi'') = \int_0^a \mathbf{Q}(s_0, s_1) \cdot e^{-E(s_o)x} \cdot Prob(s_1, \phi' \, \mathcal{U}^{[a \ominus x, b \ominus x]} \, \phi'') dx +$$
$$+ \int_a^b \mathbf{Q}(s_0, s_1) \cdot e^{-E(s_o)x} \cdot Prob(s_1, \phi' \, \mathcal{U}^{[0, b \ominus x]} \, \phi'') dx$$

Since we are assuming $s_1 \models \phi'' \wedge \neg\phi'$ *then* $Prob(s_1, \phi' \, \mathcal{U}^{[a \ominus x, b \ominus x]} \, \phi'') = 0$ *for any* $x \in [0, a)$*, thus* $\int_0^a \mathbf{Q}(s_0, s_1) \cdot e^{-E(s_o)x} \cdot Prob(s_1, \phi' \, \mathcal{U}^{[a \ominus x, b \ominus x]} \, \phi'') dx = 0$ *while* $Prob(s_1, \phi' \, \mathcal{U}^{[0, b \ominus x]} \, \phi'') = 1$ *(see previous case). Hence*

$$Prob(s_o, \phi' \, \mathcal{U}^{[a,b]} \, \phi'') = \int_a^b \mathbf{Q}(s_0, s_1) \cdot e^{-E(s_o)x} dx$$

- $a = b > 0$ *(single-point interval). Again by application of the second case of equation 2.3.1 we have that*

  $$Prob(s_o, \phi' \, \mathcal{U}^{[a,a]} \, \phi'') = \int_0^a \mathbf{Q}(s_0, s_1) \cdot e^{-E(s_o)x} \cdot Prob(s_1, \phi' \, \mathcal{U}^{[a \ominus x, a \ominus x]} \, \phi'') dx$$

  but since, as we pointed out above, $Prob(s_1, \phi' \, \mathcal{U}^{[a \ominus x, a \ominus x]} \, \phi'') = 0$ *for any* $x \in [0, a)$ *then*

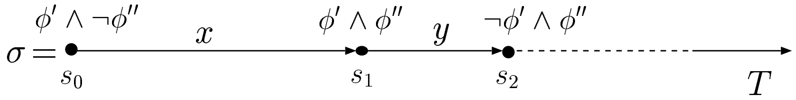
  $$Prob(s_o, \phi' \, \mathcal{U}^{[a,a]} \, \phi'') = \int_0^a \mathbf{Q}(s_0, s_1) \cdot e^{-E(s_o)x} \cdot Prob(s_1, \phi' \, \mathcal{U}^{[a \ominus x, a \ominus x]} \, \phi'') dx = 0$$

  showing that whenever dealing with a single-point bound a path $\sigma \in Path(s_0, (\phi' \wedge \neg\phi'') \; \mathcal{U} \; (\neg\phi' \wedge \phi''))$ *leads to a null probability measure.*

iii) $s_0 \models \phi' \wedge \neg\phi''$ **and** $s_1 \models \phi' \wedge \phi''$.

In Figure 3.5 s_0 *is again assumed to satisfy the* premise *but not the* target *of the until,* *hence again the second case of equation 2.3.1 applies*

$$Prob(s_o, \phi' \mathcal{U}^{[a,b]}\ \phi'') = \int_0^b \mathbf{Q}(s_0,s_1) \cdot e^{-E(s_o)x} \cdot Prob(s_1, \phi' \mathcal{U}^{[a\ominus x, b\ominus x]}\ \phi'')dx$$



$$ii)\ Prob(s_0, \phi' \mathcal{U}^{[a,a]} \phi'') = \mathbf{Q}(s_0,s_1) \cdot \frac{(e^{-a \cdot E(s_0)} - e^{-a \cdot E(s_1)})}{[E(s_1) - E(s_0)]}$$

Figure 3.5: Probability measure of timed-until path: case $s_0 \models \phi' \wedge \neg\phi''$ and $s_1 \models \phi' \wedge \phi''$

though now the recursive call $Prob(s_1, \phi' \mathcal{U}^{I\ominus x}\ \phi'')$ *is done with respect to a state,* s_1*, in which both* ϕ' *and* ϕ'' *are assumed to be true. Hence* $Prob(s_1, \phi' \mathcal{U}^{I\ominus x}\ \phi'')$ *is obtained through the third case of equation 2.3.1, therefore:*

$$Prob(s_o, \phi' \mathcal{U}^{[a,b]}\ \phi'') = \int_0^b \mathbf{Q}(s_0,s_1) \cdot e^{-E(s_o)x} \cdot \Big[e^{-E(s_1)\cdot(a\ominus x)} + \int_0^{a\ominus x} \mathbf{Q}(s_1,s_2) \cdot e^{-E(s_1)\cdot y} \cdot Prob(s_2, I \ominus x \ominus y)dy \Big] dx$$

In the case of a single-point interval (i.e. $b = a$*), as shown in the previous case,* $Prob(s_2, I \ominus x \ominus y) = 0$*,* $\forall x \in [0,a)$ *and* $\forall y \in [0, a-x)$*, thus*

$$\begin{aligned} Prob(s_o, \phi' \mathcal{U}^{[a,a]}\ \phi'') &= \int_0^a \mathbf{Q}(s_0,s_1) \cdot e^{-E(s_o)x} \cdot e^{-E(s_1)\cdot(a-x)}dx \\ &= \mathbf{Q}(s_0,s_1) \cdot \frac{(e^{-a\cdot E(s_0)} - e^{-b\cdot E(s_1)})}{[E(s_1) - E(s_0)]} \end{aligned}$$

which shows that whenever the untimed embedded generator $\overline{\sigma}$ *of a timed path* σ *belongs to* $Path(s_0, (\phi' \wedge \neg\phi'')\ \mathcal{U}\ (\phi' \wedge \phi''))$ *then the probability measure of the set of paths* σ *satisfying the bounded Until with respect to a single-point bound* $I = [a,a]$*, is not necessarily equal to zero.*

The above example has shown a peculiarity which concerns the semantics of bounded Until formulae ($\phi' \mathcal{U}^I\ \phi''$), when the bounding interval I consists of a single-point. In such a case, only paths which reach a future state $\sigma[n]$ satisfying both the *premise* and the *target* of the considered Until (i.e. $\sigma[n] \models \phi' \wedge \phi''$) through states satisfying the *premise*, lead to a measure greater than zero, which is: any path reaching a future state where the *target* but not the *premise* is satisfied (through states satisfying the *premise*), has probability zero to fulfil the single-point time bound $I = [a,a]$.

3.4 Well-formed CSL probabilistic formulae

With respect to the syntax described in Definition 2.3.5, CSL probabilistic formulae are those requiring the comparison of a probability measure with a *bound*: either $\mathcal{S}_{\trianglelefteq p}(\phi)$, namely *steady-state* formulae, or $\mathcal{P}_{\trianglelefteq p}(\varphi)$, that is *probabilistic-path* formulae.

The literature on CSL seems to lack of consistency with respect to the type of comparison operators ($\trianglelefteq$) allowed for formulating probabilistic formulae. In its original definition [1], the only comparison operator allowed was $>$; as a result $\mathcal{P}_{>p}(\varphi)$ was the only possible form of probabilistic formula (the *steady-state* operator was not included in the original CSL syntax). In [4, 23], two comparison operators were permitted: $\leq$ and $\geq$; hence possible probabilistic formulae were either $\mathcal{P}_{\leq p}(\varphi)$ or $\mathcal{P}_{\geq p}(\varphi)$, for *probabilistic-path* formulae, or $\mathcal{S}_{\leq p}(\phi)$ or $\mathcal{S}_{\geq p}(\phi)$, for probabilistic *steady-state* formulae. In its most recent treatment [5], the set of allowed comparison operators has been further enlarged becoming the one described in Definition 2.3.5, which is: $\trianglelefteq \in \{\leq, <, >, \geq\}$.

The analysis of the sensibleness of possible combinations of the comparison operator $\trianglelefteq$ and the probability bound p, leads to the characterisation of relevant semantic equivalences concerning CSL probabilistic formulae which we present here.

Definition 3.4.1 (Semantically equivalent state formulae.) *Let $\mathcal{M} = (S, \mathbf{Q}, L)$ be a labelled CTMC; two CSL state formulae ϕ and ψ are semantically equivalent, denoted $\phi \equiv \psi$, if and only if*

$$s \models \phi \Longleftrightarrow s \models \psi, \quad \forall s \in S$$

or, equivalently:

$$\phi \equiv \psi \iff Sat(\phi) = Sat(\psi)$$

The syntax depicted in Definition 2.3.5 does not impose any restriction on the combinations of the comparison operator $\unlhd \in \{\leq, <, >, \geq\}$ and the probability bound $p \in [0,1]$. However not all the possible combinations $(\unlhd, p)$ lead to sensible formulae. For example, although syntactically correct, comparison combinations like $(<, 0)$ or $(>, 1)$, result in contradictory probabilistic formulae (i.e. formulae equivalent to the contradiction $\neg tt$). In fact it is clearly impossible for a probability measure to fall outside the interval $[0,1]$.

Remark 3.4.1 (Basic contradictions) *Let ϕ be a CSL state formula and φ a CSL path formula, then the following equivalences hold:*

$$\mathcal{S}_{<0}(\phi) \equiv \mathcal{P}_{<0}(\varphi) \equiv \mathcal{S}_{>1}(\phi) \equiv \mathcal{P}_{>1}(\varphi) \equiv \neg tt$$

Symmetrically, comparison combinations like $(\geq, 0)$ or $(\leq, 1)$ lead to valid probabilistic formulae (i.e. formulae equivalent to the validity tt). In fact any probability measure $\bar{p} \in [0,1]$, hence, trivially, it is also greater than or equal to 0 and less than or equal to 1.

Remark 3.4.2 (Basic validities) *Let ϕ be a CSL state formula and φ a CSL path formula, then the following equivalences hold:*

$$\mathcal{S}_{\geq 0}(\phi) \equiv \mathcal{P}_{\geq 0}(\varphi) \equiv \mathcal{S}_{\leq 1}(\phi) \equiv \mathcal{P}_{\leq 1}(\varphi) \equiv tt$$

By allowing the *equality check* $(==)$ among the possible comparison type a probability measure can be verified for (i.e. $\unlhd \in \{\leq, <, >, \geq, ==\}$), the following trivial equivalences hold:

Remark 3.4.3 (Equality check equivalences) *Let ϕ be a CSL state formula and φ a CSL path formula, then the following equivalences hold:*

$$\mathcal{S}_{\geq 1}(\phi) \equiv \mathcal{S}_{==1}(\phi), \quad \mathcal{S}_{\geq 1}(\phi) \equiv \mathcal{S}_{==1}(\phi)$$
$$\mathcal{P}_{\geq 1}(\varphi) \equiv \mathcal{P}_{==1}(\varphi), \quad \mathcal{P}_{\leq 0}(\varphi) \equiv \mathcal{P}_{==0}(\varphi)$$

As for the PCTL, the *existential* and *universal* path quantifiers can be obtained as special cases of *probabilistic-path* formulae, obtained, respectively, by the comparison combinations $(>, 0)$ and $(\geq, 1)$.

Remark 3.4.4 (Existential and Universal path quantifiers) *Let φ be a CSL path formula, then $E(\varphi)$ and $A(\varphi)$ are notations to represent respectively, the existentially and universally path quantified formula $\mathcal{P}_{>0}(\varphi)$ and $\mathcal{P}_{\geq 1}(\varphi)$.*

$$E(\varphi) \equiv \mathcal{P}_{>0}(\varphi)$$
$$A(\varphi) \equiv \mathcal{P}_{\geq 1}(\varphi)$$

The following definition provides a list of logical conditions useful for characterising the type of bound check the combination $(\trianglelefteq, p)$ represents.

Definition 3.4.2 *Let $\trianglelefteq \in \{\leq, <, >, \geq\}$ be a comparison operator and $p \in [0,1]$ a probability value, the following logical conditions characterise the type of the combination $(\trianglelefteq, p)$:*

- $low(\trianglelefteq, p) \equiv [p \in (0,1] \wedge \trianglelefteq \equiv \geq] \vee [p \in [0,1) \wedge \trianglelefteq \equiv >]$.
  The combination $(\trianglelefteq, p)$ represents a lower-bound check if the logical condition $low(\trianglelefteq, p)$ holds.
- $up(\trianglelefteq, p) \equiv [p \in [0,1) \wedge \trianglelefteq \equiv \leq] \vee [p \in (0,1] \wedge \trianglelefteq \equiv <]$.
  The combination $(\trianglelefteq, p)$ represents an upper-bound check if the logical condition $up(\trianglelefteq, p)$ holds.
- $E(\trianglelefteq, p) \equiv [p = 0 \wedge \trianglelefteq \equiv >]$ *(Existential quantifier).*
  The combination $(\trianglelefteq, p)$ characterises an Existential quantifier for probabilistic-path formulae.
- $A(\trianglelefteq, p) \equiv [p = 1 \wedge \trianglelefteq \equiv >] \vee [p = 1 \wedge \trianglelefteq \equiv \geq]$ *(Universal quantifier).*
  The combination $(\trianglelefteq, p)$ characterises an Universal quantifier for probabilistic-path formulae.

The logical conditions introduced in the above definition allow for the characterisation of semantic equivalences for formulae involving the *steady-state* operator. Although the type of check a pair $(\unlhd, p)$ represents (i.e. *lower-bound* or *upper-bound*) could be distinguished only by means of the comparison operator ($\unlhd$), the value of the probability bound p is also relevant. In fact by considering p, it is possible to rule out those combinations $(\unlhd, p)$ leading to either contradictory or always valid probabilistic formulae. That allows for the following definition.

Definition 3.4.3 (Well-formed probabilistic CSL formulae) *Let ϕ be a CSL state formula, φ a CSL well-formed path formula, $p \in [0,1]$ a probability bound and $\unlhd \in \{\leq, <, >, \geq\}$ a comparison operator. The probabilistic formulae $\mathcal{S}_{\unlhd p}(\phi)$ and $\mathcal{P}_{\unlhd p}(\varphi)$ are said to be well-formed if and only if*

$$low(\unlhd, p) \quad or \quad up(\unlhd, q)$$

The above definition allows for ruling out probabilistic formulae whose semantics is trivial (e.g. $\mathcal{S}_{\geq 0}(\phi)$ or $\mathcal{P}_{<0}(\varphi)$). For the remaining part of the thesis, unless otherwise stated, we will assume that any generic CSL probabilistic formula like $\mathcal{S}_{\unlhd p}(\phi)$ or $\mathcal{P}_{\unlhd p}(\varphi)$, is actually a *well-formed* one.

3.5 Nesting of the CSL probabilistic connectives

The mutually recursive structure of CSL state and path formulae syntax (see Definition 2.3.5) allows for nesting of the probabilistic operators $\mathcal{S}_{\unlhd p}$ and $\mathcal{P}_{\unlhd p}$. As a result formulae like

$$\mathcal{S}_{\geq 0.9}(\mathcal{P}_{\geq 0.8}(rec_1 \ \mathcal{U} \ idle_1)) \tag{3.5.1}$$

or

$$\mathcal{P}_{\geq 0.9}(X \ (\mathcal{S}_{\geq 0.8} rec_1)) \tag{3.5.2}$$

showing a probabilistic path formula nested within the steady-state operator and a steady-state formula nested within a probabilistic next, are legitimate examples of properties in the CSL syntax.

In [5] an interesting overview regarding the specification of performance measures in terms of CSL formulae, is given. The authors show how standard steady-state and transient measures can be obtained in CSL. Furthermore it is shown how the expressiveness with respect to performance measuring of CTMCs is improved with CSL by means of time bounded path formulae. Finally the possibility of mutual nesting of the probabilistic connectives $\mathcal{S}_{\trianglelefteq p}$ and $\mathcal{P}_{\trianglelefteq p}$ is considered and it is proved that that provides further means to state useful measures which it would not be possible to express by means of any other CTMCs' analysis technique.

Concerning the issue of nesting the probabilistic operators of CSL a relevant point appears not to have been considered in literature: a distinction has to be made depending on the structure of the considered CTMC, either ergodic (i.e. consisting of a single BSCC) or non-ergodic (i.e. resulting in a number of BSCCs). The ergodicity of the model impacts on the validity of CSL steady-state formulae, as pointed out in the following remark.

Proposition 3.5.1 (Steady-state semantics with respect to ergodic CMTCs) *Let ϕ be a CSL state formula, $\mathcal{M} = (S, \mathbf{Q}, L)$ an ergodic labelled CTMC, $p \in [0,1]$ a probability bound and $\trianglelefteq \in \{<, \leq, >, \geq\}$. The formula $\mathcal{S}_{\trianglelefteq p}(\phi)$ is either satisfied in all or none of the states $s \in S$:*

$$\forall s \in S,\ \ s \models \mathcal{S}_{\trianglelefteq p}(\phi) \qquad or \qquad \forall s \in S,\ \ s \not\models \mathcal{S}_{\trianglelefteq p}(\phi)$$

Proof. From the CSL semantics we know that

$$s \models \mathcal{S}_{\trianglelefteq p}(\phi) \Longleftrightarrow \pi^{\mathcal{M}}(s, Sat(\phi)) \trianglelefteq p$$

Since $\mathcal{M}$ is ergodic, then from Proposition 2.3.1 we know that:

$$\pi^{\mathcal{M}}(s, Sat(\phi)) = \sum_{s' \in Sat(\phi)} \pi(s')$$

which proves the measure $\pi^{\mathcal{M}}(s, Sat(\phi))$ being dependent only on the satisfiability set $Sat(\phi)$ and not on the considered state s.

$\square$

The above remark points out a relevant feature of the CSL logic: steady-state formulae, like $\mathcal{S}_{\trianglelefteq p}(\phi)$, are model dependent, rather than state dependent, whenever the model they refer to is an ergodic CTMC.

Intuitively, for a non-ergodic CTMC $\mathcal{M}$, the satisfiability of a steady-state formula like $\mathcal{S}_{\trianglelefteq p}(\phi)$ with respect to a state s, depends on the measure of *how likely it is to reach a state satisfying* ϕ *when* s *is considered as the starting point* (to be more precise: *how likely it is to reach the BSCC* B *a state satisfying* ϕ *belongs to, when we start from* s). Hence, in the non-ergodic framework, the satisfiability of $\mathcal{S}_{\trianglelefteq p}(\phi)$ strongly depends on the considered starting state s other than on $\mathcal{M}$ itself. Conversely, in the ergodic case, all states belong to the same, unique, BSCC the CTMC consists of; thus the formula $\mathcal{S}_{\trianglelefteq p}(\phi)$ is either valid in $\mathcal{M}$ or false in every state: the satisfiability of $\mathcal{S}_{\trianglelefteq p}(\phi)$ is state independent.

As a consequence, in the ergodic framework, we will use $\mathcal{M} \models \mathcal{S}_{\trianglelefteq p}(\phi)$ ($\mathcal{M} \not\models \mathcal{S}_{\trianglelefteq p}(\phi)$) to denote the fact that the steady-state formula $\mathcal{S}_{\trianglelefteq p}(\phi)$ is satisfied (not satisfied) in every state of $\mathcal{M}$ or, whenever the model $\mathcal{M}$ is clear from the context, simply $\models \mathcal{S}_{\trianglelefteq p}(\phi)$ ($\not\models \mathcal{S}_{\trianglelefteq p}(\phi)$). An alternative formulation of Proposition 3.5.1 is given in the following corollary.

Corollary 3.5.1 *The satisfiability set of* $\mathcal{S}_{\trianglelefteq p}(\phi)$ *with respect to an ergodic CTMC* $\mathcal{M} = (S, \mathbf{Q}, L)$ *is either the whole state space* S *or the empty set:*

$$Sat\big(\mathcal{S}_{\trianglelefteq p}(\phi)\big) = S \quad or \quad Sat\big(\mathcal{S}_{\trianglelefteq p}(\phi)\big) = \emptyset$$

$\square$

3.5.1 Semantics equivalences for nested formulae

The relevance of the model's ergodicity with respect to the semantics of CSL steady-state formulae, calls for checking the existence of semantic equivalences for formulae involving the *steady-state* connective. The following propositions characterise relevant equivalences concerning such types of formulae.

Proposition 3.5.2 (Basic steady-state equivalence.) *Let* $\mathcal{M} = (S, \mathbf{Q}, L)$ *be an ergodic labelled CTMC and* $\mathcal{S}_{\trianglelefteq p}(\phi)$ *a CSL state formula, then* $\mathcal{S}_{\trianglelefteq p}(\phi)$ *is semantically equiva-*

lent either to tt, if $\models \mathcal{S}_{\trianglelefteq p}(\phi)$ *or to* $\neg tt$*, if* $\not\models \mathcal{S}_{\trianglelefteq p}(\phi)$*:*

$$\mathcal{S}_{\trianglelefteq p}(\phi) \equiv \begin{cases} tt & \textbf{if} \ \models \mathcal{S}_{\trianglelefteq p}(\phi) \\ \neg tt & \textbf{if} \ \not\models \mathcal{S}_{\trianglelefteq p}(\phi) \end{cases} \tag{3.5.3}$$

Proof.
Trivial consequence of Proposition 3.5.1, in fact clearly

$$\models \mathcal{S}_{\trianglelefteq p}(\phi) \Longleftrightarrow Sat(\mathcal{S}_{\trianglelefteq p}(\phi)) = S = Sat(tt)$$
$$\not\models \mathcal{S}_{\trianglelefteq p}(\phi) \Longleftrightarrow Sat(\mathcal{S}_{\trianglelefteq p}(\phi)) = \emptyset = Sat(\neg tt)$$

□

Relying on Proposition 3.5.2 semantic equivalences can be found for any possible combination of *steady-state* formulae obtained by means of the CSL connectives.

Proposition 3.5.3 ($\mathcal{S}$ nested in $\mathcal{S}$) . *Let* $\mathcal{M} = (S, \mathbf{Q}, L)$ *be an ergodic labelled CTMC and* ϕ *a CSL state formula. The following semantic equivalence regarding nesting of the steady-state formula* $\mathcal{S}_{\overline{\trianglelefteq} q}(\phi)$ *within a probabilistic steady-state operator holds:*

$$\mathcal{S}_{\trianglelefteq p}(\mathcal{S}_{\overline{\trianglelefteq} q}(\phi)) \equiv \begin{cases} tt & \textbf{if} \Big[\big[low(\trianglelefteq, p)\big] \wedge \big[\models \mathcal{S}_{\overline{\trianglelefteq} q}(\phi)\big] \Big] \vee \\ & \qquad \Big[\big[up(\trianglelefteq, p)\big] \wedge \big[\not\models \mathcal{S}_{\overline{\trianglelefteq} q}(\phi)\big] \Big] \\ \neg tt & otherwise \end{cases} \tag{3.5.4}$$

Proof.
The proof proceeds similarly to the one regarding nesting of $\mathcal{S}$ within an unbounded Next. The validity of the innermost steady-state formula (i.e. $\mathcal{S}_{\overline{\trianglelefteq} q}(\phi)$), directly affects the equivalence. In fact the nested formula $\mathcal{S}_{\trianglelefteq p}(\mathcal{S}_{\overline{\trianglelefteq} q}(\phi))$ states that the long-run probability for the states satisfying $\mathcal{S}_{\overline{\trianglelefteq} q}(\phi)$ respects the bound $\trianglelefteq p$. However the states satisfying the innermost steady-state are either all the states in S or none. As a result if $\mathcal{S}_{\overline{\trianglelefteq} q}(\phi)$ is satisfied in $\mathcal{M}$ (i.e. $\models \mathcal{S}_{\overline{\trianglelefteq} q}(\phi)$), then so is $\mathcal{S}_{\trianglelefteq p}(\mathcal{S}_{\overline{\trianglelefteq} q}(\phi))$ given that the type of bound check it represents is a *lower-bound*. While if $\mathcal{S}_{\overline{\trianglelefteq} q}(\phi)$ is not satisfied in $\mathcal{M}$ (i.e. $\not\models \mathcal{S}_{\overline{\trianglelefteq} q}(\phi)$) then $\mathcal{S}_{\trianglelefteq p}(\mathcal{S}_{\overline{\trianglelefteq} q}(\phi))$ is satisfied in $\mathcal{M}$, given that it represents an *upper-bound* check.

□

The above proposition has shown that formulae given by nesting a *steady-state* connective within another one are equivalent either to the tautology tt or to the contradiction $\neg tt$. The following two propositions, instead, highlight the equivalences for boolean combinations of *steady-state* connectives (i.e. conjunctions and negations of *steady-state* formulae).

Proposition 3.5.4 ($\mathcal{S}$ nested in $\wedge$) *Let $\mathcal{M} = (S, \mathbf{Q}, L)$ be an ergodic labelled CTMC, ϕ and ψ two CSL state formulae, $p, q \in [0,1]$ and $\trianglelefteq \overline{\trianglelefteq} \in \{<, \leq, \geq, >\}$. The following semantic equivalences regarding nesting of steady-state formulae within the conjunctive operator holds:*

$$\psi \wedge (\mathcal{S}_{\trianglelefteq p}(\phi)) \quad \equiv \quad \begin{cases} \psi & \textbf{if } \models \mathcal{S}_{\trianglelefteq p}(\phi) \\ \\ \neg tt & \textit{otherwise} \end{cases}$$

$$(\mathcal{S}_{\trianglelefteq p}(\phi)) \wedge (\mathcal{S}_{\overline{\trianglelefteq} q}(\psi)) \quad \equiv \quad \begin{cases} tt & \textbf{if } \big[\models \mathcal{S}_{\trianglelefteq q}(\phi)\big] \wedge \big[\models \mathcal{S}_{\overline{\trianglelefteq} q}(\psi)\big] \\ \\ \neg tt & \textit{otherwise} \end{cases}$$

Proof.

Trivial consequence of Proposition 3.5.2.

□

Proposition 3.5.5 ($\mathcal{S}$ nested in $\neg$) *Let $\mathcal{M} = (S, \mathbf{Q}, L)$ be an ergodic labelled CTMC, ϕ a CSL state formula, $p \in [0,1]$ and $\trianglelefteq \in \{<, \leq, \geq, >\}$. The following semantic equivalence regarding nesting of the steady-state formula $\mathcal{S}_{\trianglelefteq p}(\phi)$ within the negation operator holds:*

$$\neg(\mathcal{S}_{\trianglelefteq p}(\phi)) \equiv \begin{cases} \neg tt & \textbf{if } \models \mathcal{S}_{\trianglelefteq p}(\phi) \\ \\ tt & \textit{otherwise} \end{cases} \tag{3.5.5}$$

Proof.

Trivial consequence of Proposition 3.5.2. □

So far equivalences for non-path formulae, containing a *steady-state* sub-formula have been proved. In the following four propositions the case of path formulae, both Until and Next either bounded or unbounded, built on some *steady-state* sub-formula is faced and equivalences are proved.

Proposition 3.5.6 ($\mathcal{S}$ nested in a bounded Until.) *Let $\mathcal{M} = (S, \mathbf{Q}, L)$ be an ergodic labelled CTMC, ϕ and ψ two CSL state formulae, $I = [a,b] \subseteq \mathbb{R}_{\geq 0}$ a time interval, $p,q,r \in [0,1]$ and $\trianglelefteq, \overline{\trianglelefteq}, \tilde{\trianglelefteq} \in \{<, \leq, \geq, >\}$. The following semantic equivalences regarding nesting of the steady-state formulae within a probabilistic bounded Until operator hold:*

$$\mathcal{P}_{\trianglelefteq p}(\phi\ \mathcal{U}^I\ \mathcal{S}_{\overline{\trianglelefteq} q}(\psi)) \equiv \begin{cases} tt & \textbf{if } \Big[\big[\models \mathcal{S}_{\overline{\trianglelefteq} q}(\psi)\big] \wedge \big[low(\trianglelefteq, p)\big] \wedge \big[a = 0\big]\Big] \\ & \quad \vee \Big[\big[\not\models \mathcal{S}_{\overline{\trianglelefteq} q}(\psi)\big] \wedge \big[up(\trianglelefteq, p)\big]\Big] \\ \neg\phi \vee [\phi \wedge \mathcal{P}_{\trianglelefteq p}(\phi\ \mathcal{U}^I\ tt)] & \textbf{if } \Big[\big[\models \mathcal{S}_{\overline{\trianglelefteq} q}(\psi)\big] \wedge \big[up(\trianglelefteq, p)\big] \wedge \big[a \neq 0\big]\Big] \\ [\phi \wedge \mathcal{P}_{\trianglelefteq p}(\phi\ \mathcal{U}^I\ tt)] & \textbf{if } \Big[\big[\models \mathcal{S}_{\overline{\trianglelefteq} q}(\psi)\big] \wedge \big[low(\trianglelefteq, p)\big] \wedge \big[a \neq 0\big]\Big] \\ \neg tt & \textbf{if } \Big[\big[\models \mathcal{S}_{\overline{\trianglelefteq} q}(\psi)\big] \wedge \big[up(\trianglelefteq, p)\big] \wedge \big[a = 0\big]\Big] \\ & \quad \vee \Big[\big[\not\models \mathcal{S}_{\overline{\trianglelefteq} q}(\psi)\big] \wedge \big[low(\trianglelefteq, p)\big]\Big] \end{cases}$$

$$
\mathcal{P}_{\trianglelefteq p}(\mathcal{S}_{\overline{\trianglelefteq} q}(\psi)\ \mathcal{U}^I\ \phi) \equiv \begin{cases} \mathcal{P}_{\trianglelefteq p}(\diamond^I \phi) & \textbf{if } \models \mathcal{S}_{\overline{\trianglelefteq} q}(\phi) \\ \phi & \textbf{if } \not\models \mathcal{S}_{\overline{\trianglelefteq} q}(\phi) \wedge [low(\trianglelefteq, p)] \wedge [a = 0] \\ tt & \textbf{if } \not\models \mathcal{S}_{\overline{\trianglelefteq} q}(\phi) \wedge [up(\trianglelefteq, p)] \wedge [a \neq 0] \\ \neg tt & \textit{otherwise} \end{cases}
$$

$$
\mathcal{P}_{\trianglelefteq p}(\mathcal{S}_{\tilde{\trianglelefteq} r}(\phi)\ \mathcal{U}^I\ \mathcal{S}_{\overline{\trianglelefteq} q}(\psi)) \equiv \begin{cases} tt & \textbf{if } \Big[[\models \mathcal{S}_{\overline{\trianglelefteq} q}(\psi)] \wedge [low(\trianglelefteq, p)] \wedge [a = 0]\Big] \\ & \quad \vee \Big[[\not\models \mathcal{S}_{\overline{\trianglelefteq} q}(\psi)] \wedge [up(\trianglelefteq, p)]\Big] \\ & \quad \vee \Big[[\models \mathcal{S}_{\overline{\trianglelefteq} q}(\psi)] \wedge [\models \mathcal{S}_{\tilde{\trianglelefteq} r}(\phi)] \\ & \qquad \wedge\ [up(\trianglelefteq, p)] \wedge [a > 0]\Big] \\ \neg tt & \textbf{if } \Big[[\models \mathcal{S}_{\overline{\trianglelefteq} q}(\psi)] \wedge [up(\trianglelefteq, p)] \wedge [a = 0]\Big] \\ & \quad \vee \Big[[\not\models \mathcal{S}_{\overline{\trianglelefteq} q}(\psi)] \wedge [low(\trianglelefteq, p)]\Big] \\ & \quad \vee \Big[[\models \mathcal{S}_{\overline{\trianglelefteq} q}(\psi)] \wedge [\models \mathcal{S}_{\tilde{\trianglelefteq} r}(\phi)] \\ & \qquad \wedge\ [low(\trianglelefteq, p)] \wedge [a > 0]\Big] \\ \mathcal{P}_{\trianglelefteq p}(\diamond^I tt) & \textbf{if } \Big[[\models \mathcal{S}_{\overline{\trianglelefteq} q}(\psi)] \wedge [\models \mathcal{S}_{\tilde{\trianglelefteq} r}(\phi)] \wedge [a > 0]\Big] \end{cases}
$$

Proof.

The proof relies both on the basic equivalence for *steady-state* formulae (Proposition 3.5.2) and on the result of Theorem 2.3.1 concerning the probability measure for bounded-Until paths. With respect to the first and second cases (i.e. one *steady-state* formula among the operands of a bounded Until), three factors are relevant: the validity of the *steady-state* operand with respect to the model $\mathcal{M}$ (either $\models \mathcal{S}_{\overline{\trianglelefteq} q}(\psi)$ or $\not\models \mathcal{S}_{\overline{\trianglelefteq} q}(\psi)$), the type of bound check involved (either $low(\trianglelefteq, p)$ or $up(\trianglelefteq, p)$) and the value of the bounding interval's infinum (either $a = 0$ or $a > 0$). That leads to a total of eight possible combinations which, are fully caught by the conditions characterising the first and second equivalences. In the third case, (i.e. both the operands of the Until are *steady-state* formulae) also the validity of the second *steady-state* operand of the bounded Until, has to be considered amongst the factors affecting the semantics of $\mathcal{P}_{\trianglelefteq p}(\mathcal{S}_{\bar{\trianglelefteq} r}(\phi)\ \mathcal{U}^I\ \mathcal{S}_{\bar{\trianglelefteq} q}(\psi))$. Hence the possible combinations of conditions are, in this case, sixteen.

i) $\mathcal{P}_{\trianglelefteq p}(\phi\ \mathcal{U}\ \mathcal{S}_{\bar{\trianglelefteq} q}(\psi))$.

Case 1. If $\mathcal{S}_{\bar{\trianglelefteq} q}(\psi)$ is valid in the considered model and $a = 0$, the probability measure of paths satisfying $(\phi\ \mathcal{U}^I\ \mathcal{S}_{\bar{\trianglelefteq} q}(\psi))$ is equal 1 (case 1 of Theorem 2.3.1), for every state s. Hence if we are checking that measure against a *lower-bound* (i.e. $low(\trianglelefteq, p)$) the formula is satisfied in every state s, since clearly $1 \trianglelefteq p$. Similarly if $\mathcal{S}_{\bar{\trianglelefteq} q}(\psi)$ is not satisfied in $\mathcal{M}$ (i.e. not satisfied in any state $s \in S$), then it is equivalent to the contradiction $\neg tt$ (Proposition 3.5.2); thus the original Until formula $\mathcal{P}_{\trianglelefteq p}(\phi\ \mathcal{U}\ \mathcal{S}_{\bar{\trianglelefteq} q}(\psi))$ boils down to $\mathcal{P}_{\trianglelefteq p}(\phi\ \mathcal{U}\ \neg tt)$ which, independently of a and of the considered state s, is satisfied, if and only if $(\trianglelefteq, p)$ represents an upper bound check (i.e. $up(\trianglelefteq, p)$): the probability measure of paths satisfying $\mathcal{P}_{\trianglelefteq p}(\phi\ \mathcal{U}\ \neg tt)$ is zero (case "otherwise" of Theorem 2.3.1).

Case 2. If $\mathcal{S}_{\bar{\trianglelefteq} q}(\psi)$ is valid in the considered model and $a > 0$ then a distinction between states satisfying ϕ and states satisfying $\neg\phi$ needs to be made. First of all, since we are assuming $\models \mathcal{S}_{\bar{\trianglelefteq} q}(\psi)$ then, again, the original Until formula is actually equivalent to $\mathcal{P}_{\trianglelefteq p}(\phi\ \mathcal{U}\ tt)$. If s does not satisfy ϕ (i.e. $s \models \neg\phi$), the probability measure $Prob(s, \phi\ \mathcal{U}^I\ tt)$ is equal zero (case "otherwise" of Theorem 2.3.1). Hence, if $up(\trianglelefteq, p)$, $\mathcal{P}_{\trianglelefteq p}(\phi\ \mathcal{U}\ \mathcal{S}_{\bar{\trianglelefteq} q}(\psi))$ is clearly satisfied in s, as $0 \trianglelefteq p$. On the other hand if s satisfies ϕ, case 3 of Theorem 2.3.1 applies. and the measure $Prob(s, \phi\ \mathcal{U}^I\ tt)$ is equal to the probability of leaving s within the bound I. The formula $\phi \wedge \mathcal{P}_{\trianglelefteq p}(\phi\ \mathcal{U}^I\ tt)$ clearly captures the states fulfilling this second possibility

Case 3. This case is identical to the previous one (case 2) except for the type of bound check, which is supposed, in this case, to be a *lower-bound* check (i.e. $low(\trianglelefteq, p)$). As we know, $Prob(s, \phi\ \mathcal{U}^I\ \mathcal{S}_{\bar{\trianglelefteq} q}(\psi)) = 0 \not\trianglelefteq p$ for any state $s \models \neg\phi$ (see previous case). Hence the only states s for which the measure $Prob(s, \phi\ \mathcal{U}^I\ \mathcal{S}_{\bar{\trianglelefteq} q}(\psi)) \trianglelefteq p$ are those satisfying ϕ and $Prob(s, \phi\ \mathcal{U}^I\ tt)$ (i.e. s is such that the probability measure of $(\phi\ \mathcal{U}^I\ tt)$ is $\trianglelefteq p$).

Case 4. If $\models \mathcal{S}_{\overline{\trianglelefteq}q}(\psi)$ and $a = 0$, then $Prob(s, \phi\ \mathcal{U}^I\ \mathcal{S}_{\overline{\trianglelefteq}q}(\psi)) = 1$, independently of the state s (i.e. case 1 of Theorem 2.3.1). Hence, if $up(\trianglelefteq, p)$ holds, then clearly $Prob(s, \phi\ \mathcal{U}^I\ \mathcal{S}_{\overline{\trianglelefteq}q}(\psi)) = 1 \ntrianglelefteq p$, for all $s \in S$ which proves $\mathcal{P}_{\trianglelefteq p}(\phi\ \mathcal{U}\ \mathcal{S}_{\overline{\trianglelefteq}q}(\psi)) \equiv \neg tt$. Similarly if $\not\models \mathcal{S}_{\overline{\trianglelefteq}q}(\psi)$, then $Prob(s, \phi\ \mathcal{U}^I\ \mathcal{S}_{\overline{\trianglelefteq}q}(\psi)) = 0$, independently of the state s and of a (either $a = 0$ or $a > 0$). Thus if $low(\trianglelefteq, p)$, then again $Prob(s, \phi\ \mathcal{U}^I\ \mathcal{S}_{\overline{\trianglelefteq}q}(\psi)) = 1 \ntrianglelefteq p$. which proves $\mathcal{P}_{\trianglelefteq p}(\phi\ \mathcal{U}\ \mathcal{S}_{\overline{\trianglelefteq}q}(\psi))$ being equivalent to the contradiction $\neg tt$.

ii) $\mathcal{P}_{\trianglelefteq p}(\mathcal{S}_{\overline{\trianglelefteq}q}(\psi)\ \mathcal{U}\ \phi)$.

Case 1. Direct consequence of Proposition 3.5.2 and of the equivalence $\diamond^I \phi \equiv (tt\ \mathcal{U}^I\ \phi)$.

Case 2. If $\not\models \mathcal{S}_{\overline{\trianglelefteq}q}(\psi)$ then $(\mathcal{S}_{\overline{\trianglelefteq}q}(\psi)\ \mathcal{U}^I\ \phi) \equiv (\neg tt\ \mathcal{U}^I\ \phi)$. Then if $a = 0$, a distinction between $s \models \phi$ and $s \models \neg\phi$ has to be made. $Prob(s, (\mathcal{S}_{\overline{\trianglelefteq}q}(\psi)\ \mathcal{U}\ \phi)) = 1$ for any state $s \models \phi$. Thus if $low(\trianglelefteq, p)$, then for each such state also $s \models \mathcal{P}_{\trianglelefteq p}(\mathcal{S}_{\overline{\trianglelefteq}q}(\psi)\ \mathcal{U}\ \phi)$ as clearly $1 \trianglelefteq p$. On the other hand $Prob(s, (\mathcal{S}_{\overline{\trianglelefteq}q}(\psi)\ \mathcal{U}\ \phi)) = 0 \ntrianglelefteq p$ for any state $s \models \neg\phi$.

Case 3. Here again $\not\models \mathcal{S}_{\overline{\trianglelefteq}q}(\psi)$ is assumed but now the infinum of the bounding interval I is supposed to be $a > 0$. If this is the case then $Prob(s, (\mathcal{S}_{\overline{\trianglelefteq}q}(\psi)\ \mathcal{U}\ \phi)) = 0$ both with $s \models \phi$ and with $s \not\models \phi$ (i.e. in both cases the "otherwise" case of Theorem 2.3.1 applies). Hence of $up(\triangleleft, p)$ then $\mathcal{P}_{\trianglelefteq p}(\mathcal{S}_{\overline{\trianglelefteq}q}(\psi)\ \mathcal{U}\ \phi)$ is equivalent to the tautology tt.

Case 4. This case regards the remaining two possibilities, which are, respectively: $\not\models \mathcal{S}_{\overline{\trianglelefteq}q}(\psi) \wedge low(\trianglelefteq, p) \wedge (a > 0)$ and $\not\models \mathcal{S}_{\overline{\trianglelefteq}q}(\psi) \wedge up(\trianglelefteq, p) \wedge (a = 0)$. If $\not\models \mathcal{S}_{\overline{\trianglelefteq}q}(\psi) \wedge low(\trianglelefteq, p) \wedge (a > 0)$ then $Prob(s, \neg tt\, \mathcal{U}^I\ \phi) = 0 \ntrianglelefteq p$ (i.e. we are assuming $low(\trianglelefteq, p)$), independently of s. Hence clearly, for every $s \in S$, $s \not\models \mathcal{P}_{\trianglelefteq p}(\mathcal{S}_{\overline{\trianglelefteq}q}(\psi)\ \mathcal{U}\ \phi)$ which proves the equivalence of $s \not\models \mathcal{P}_{\trianglelefteq p}(\mathcal{S}_{\overline{\trianglelefteq}q}(\psi)\ \mathcal{U}\ \phi)$ with the contradiction $\neg tt$. On the other hand if $\not\models \mathcal{S}_{\overline{\trianglelefteq}q}(\psi) \wedge up(\trianglelefteq, p) \wedge (a = 0)$ then again $Prob(s, \neg tt\, \mathcal{U}^I\ \phi) = 0 \ntrianglelefteq p$ which proves the equivalence with the contradiction also in this case.

iii) $\mathcal{P}_{\unlhd p}(\mathcal{S}_{\tilde{\unlhd} r}(\phi)\ \mathcal{U}^I\ \mathcal{S}_{\overline{\unlhd} q}(\psi))$.
Similar to the previous cases.

□

The above proposition shows the equivalences for bounded Until formulae having *steady-state* formulae amongst their operands. The following proposition, instead, proves results regarding the unbounded Until case.

Proposition 3.5.7 ($\mathcal{S}$ nested in an unbounded Until) *Let $\mathcal{M} = (S, \mathbf{Q}, L)$ be an ergodic labelled CTMC, ϕ and ψ two CSL state formulae, $p, q, r \in [0,1]$ and $\unlhd, \overline{\unlhd}, \tilde{\unlhd}, \in \{<, \leq, \geq, >\}$. The following semantic equivalences regarding nesting of the steady-state formulae within a probabilistic unbounded Until operator hold:*

$$\mathcal{P}_{\unlhd p}(\phi\ \mathcal{U}\ \mathcal{S}_{\overline{\unlhd} q}(\psi)) \quad \equiv \quad \begin{cases} tt \quad \textbf{if} \Big[\big[low(\unlhd, p)\big] \wedge \big[\models \mathcal{S}_{\overline{\unlhd} q}(\psi)\big]\Big] \vee \\ \qquad\qquad \Big[\big[up(\unlhd, p)\big] \wedge \big[\not\models \mathcal{S}_{\overline{\unlhd} q}(\psi)\big]\Big] \\ \neg tt \quad otherwise \end{cases}$$

$$\mathcal{P}_{\unlhd p}(\mathcal{S}_{\overline{\unlhd} q}(\psi)\ \mathcal{U}\ \phi) \quad \equiv \quad \begin{cases} \mathcal{P}_{\unlhd p}(\diamond \phi) & \textbf{if } \models \mathcal{S}_{\overline{\unlhd} q}(\psi) \\ \phi & \textbf{if } \not\models \mathcal{S}_{\overline{\unlhd} q}(\psi) \wedge [low(\unlhd, p)] \\ \neg\phi & \textbf{if } \not\models \mathcal{S}_{\overline{\unlhd} q}(\psi) \wedge [up(\unlhd, p)] \end{cases}$$

$$\mathcal{P}_{\unlhd p}(\mathcal{S}_{\tilde{\unlhd} r}(\phi)\ \mathcal{U}\ \mathcal{S}_{\overline{\unlhd} q}(\psi)) \quad \equiv \quad \begin{cases} tt \quad \textbf{if} \Big[\big[low(\unlhd, p)\big] \wedge \big[\models \mathcal{S}_{\overline{\unlhd} q}(\psi)\big]\Big] \vee \\ \qquad\qquad \Big[\big[up(\unlhd, p)\big] \wedge \big[\not\models \mathcal{S}_{\overline{\unlhd} q}(\psi)\big]\Big] \\ \neg tt \quad otherwise \end{cases}$$

Proof.
The proof relies on Proposition 3.5.2 and Corollary 2.3.1.

i) $\mathcal{P}_{\unlhd p}(\phi\ \mathcal{U}\ \mathcal{S}_{\overline{\unlhd} q}(\psi))$.
If $\models \mathcal{S}_{\overline{\unlhd} q}(\psi)$ then $\mathcal{S}_{\overline{\unlhd} q}(\psi)$ is equivalent to tt. As a consequence $Prob(s, \phi\ \mathcal{U}\ \psi) = 1$, for any state s. Hence if $low(\unlhd, p)$ holds, then clearly $\mathcal{P}_{\unlhd p}(\phi \mathcal{U}\ \mathcal{S}_{\overline{\unlhd} q}(\psi))$ is satisfied in any state, thus it is equivalent to tt. Conversely if $\not\models \mathcal{S}_{\overline{\unlhd} q}(\psi)$ then the Until formula is

not satisfiable by any path σ. As a result the probability measure of paths satisfying $(\phi\ \mathcal{U}\ \mathcal{S}_{\overline{\trianglelefteq}q}(\psi))$ is zero in any state hence $\mathcal{P}_{\trianglelefteq p}(\phi\ \mathcal{U}\ \mathcal{S}_{\overline{\trianglelefteq}q}(\psi))$ is equivalent to *tt* given that $up(\trianglelefteq, p)$.

ii) $\mathcal{P}_{\trianglelefteq p}(\mathcal{S}_{\overline{\trianglelefteq}q}(\psi)\,\mathcal{U}\ \phi))$.
If $\models \mathcal{S}_{\overline{\trianglelefteq}q}(\psi)$ then $\mathcal{S}_{\overline{\trianglelefteq}q}(\psi)$ is equivalent to *tt* hence $(\mathcal{S}_{\overline{\trianglelefteq}q}(\psi)\,\mathcal{U}\ \phi)$ is equivalent to the *sometime in the future* formula $(\diamond\phi)$. If $\not\models \mathcal{S}_{\overline{\trianglelefteq}q}(\psi)$ then the *premise* of the Until formula $(\mathcal{S}_{\overline{\trianglelefteq}q}(\psi)\,\mathcal{U}\ \phi)$ is always false. As a consequence for any state s the probability measure of paths satisfying $(\mathcal{S}_{\overline{\trianglelefteq}q}(\psi)\,\mathcal{U}\ \phi)$ is either 1 if the *target* ϕ is satisfied in s or 0 if it is not. Hence $\mathcal{P}_{\trianglelefteq p}(\mathcal{S}_{\overline{\trianglelefteq}q}(\psi)\,\mathcal{U}\ \phi))$ is equivalent either to ϕ, if it represents a *lower-bound* check, or to $\neg\phi$ if it is an *upper-bound* check.

iii) $\mathcal{P}_{\trianglelefteq p}(\mathcal{S}_{\tilde{\trianglelefteq}r}(\phi)\ \mathcal{U}\ \mathcal{S}_{\overline{\trianglelefteq}q}(\psi))$.
The proof of this case is a direct consequence of the first case $\mathcal{P}_{\trianglelefteq p}(\phi'\ \mathcal{U}\ \mathcal{S}_{\overline{\trianglelefteq}q}(\psi))$ with $\phi' \equiv \mathcal{S}_{\tilde{\trianglelefteq}r}(\phi)$.

$\square$

Finally in the remaining two propositions, the semantic equivalences concerning nesting of *steady-state* properties within a Next operator, either bounded or unbounded, are proved.

Proposition 3.5.8 ($\mathcal{S}$ nested in a bounded Next) *Let $\mathcal{M} = (S, \mathbf{Q}, L)$ be an ergodic labelled CTMC, ϕ a CSL state formula, $I = [a,b] \in \mathbb{R}_{\geq 0}$ a time interval, $p, q \in [0,1]$ and $\trianglelefteq, \overline{\trianglelefteq} \in \{<, \leq, \geq, >\}$. The following semantic equivalence regarding nesting of the steady-state formula $\mathcal{S}_{\overline{\trianglelefteq}q}(\phi)$ within a probabilistic bounded Next operator holds:*

$$\mathcal{P}_{\trianglelefteq p}(X^I\ \mathcal{S}_{\overline{\trianglelefteq}q}(\phi)) \equiv \begin{cases} tt & \textbf{if } \left[[\not\models \mathcal{S}_{\overline{\trianglelefteq}q}(\phi)] \wedge [up(\trianglelefteq, p)]\right] \\ \neg tt & \textbf{if } \left[[\not\models \mathcal{S}_{\overline{\trianglelefteq}q}(\phi)] \wedge [low(\trianglelefteq, p)]\right] \\ \mathcal{P}_{\trianglelefteq p}(X^I\ tt) & \textbf{if } \left[\models \mathcal{S}_{\overline{\trianglelefteq}q}(\phi)\right] \end{cases} \tag{3.5.6}$$

Proof.
If $\not\models \mathcal{S}_{\overline{\trianglelefteq}q}(\phi)$ then the measure $Prob(s,(X^I\ \mathcal{S}_{\overline{\trianglelefteq}q}(\phi))$ is equal to zero for any state $s \in S$. Hence if $up(\trianglelefteq, p)$, the formula $\mathcal{P}_{\trianglelefteq p}(X^I\ \mathcal{S}_{\overline{\trianglelefteq}q}(\phi))$ is valid in every state s as clearly $0 \trianglelefteq p$. On the other hand, if $low(\trianglelefteq, p)$, then $0 \not\trianglelefteq p$, thus $\mathcal{P}_{\trianglelefteq p}(X^I\ \mathcal{S}_{\overline{\trianglelefteq}q}(\phi))$ is equivalent to the contradiction $\neg tt$. The final case (i.e. $\models \mathcal{S}_{\overline{\trianglelefteq}q}(\phi)$) is a trivial consequence of Proposition 3.5.2.

□

Proposition 3.5.9 ($\mathcal{S}$ nested in an unbounded Next) *Let $\mathcal{M} = (S, \mathbf{Q}, L)$ be an ergodic labelled CTMC, ϕ a CSL state formula, $p, q \in [0,1]$ and $\trianglelefteq, \overline{\trianglelefteq} \in \{<, \leq, \geq, >\}$. The following semantic equivalence regarding nesting of the steady-state formula $\mathcal{S}_{\overline{\trianglelefteq}q}(\phi)$ within a probabilistic unbounded Next operator holds:*

$$\mathcal{P}_{\trianglelefteq p}(X\ \mathcal{S}_{\overline{\trianglelefteq}q}(\phi)) \equiv \begin{cases} tt & \textbf{if } \Big[\big[low(\trianglelefteq,p)\big] \wedge \big[\models \mathcal{S}_{\overline{\trianglelefteq}q}(\phi)\big]\Big] \vee \\ & \qquad \Big[\big[up(\trianglelefteq,p)\big] \wedge \big[\not\models \mathcal{S}_{\overline{\trianglelefteq}q}(\phi)\big]\Big] \\ \neg tt & otherwise \end{cases} \tag{3.5.7}$$

Proof.
If the steady-state formula $\mathcal{S}_{\overline{\trianglelefteq}q}(\phi)$ is satisfied in $\mathcal{M}$ (i.e. $\models \mathcal{S}_{\overline{\trianglelefteq}q}(\phi)$), then it is equivalent to tt, hence the probability measure of the paths satisfying $(X\ \mathcal{S}_{\overline{\trianglelefteq}q}(\phi))$ is clearly 1, independently of the starting state. As a result, $\mathcal{P}_{\trianglelefteq p}(X\ \mathcal{S}_{\overline{\trianglelefteq}q}(\phi))$ is satisfied in every state, if it represents a *lower-bound* check for the probability measure of $(X\ \mathcal{S}_{\overline{\trianglelefteq}q}(\phi))$ (i.e. $low(\trianglelefteq, p)$ holds). Similarly, if $\mathcal{S}_{\overline{\trianglelefteq}q}(\phi)$ is not satisfied in $\mathcal{M}$ (i.e. $\not\models \mathcal{S}_{\overline{\trianglelefteq}q}(\phi)$) then it is equivalent to $\neg tt$. Thus there can be no paths satisfying $(X\ \mathcal{S}_{\overline{\trianglelefteq}q}(\phi))$ whatever is the considered starting state (i.e. the probability measure of paths satisfying $(X\ \mathcal{S}_{\overline{\trianglelefteq}q}(\phi))$ is 0). As a consequence, $\mathcal{P}_{\trianglelefteq p}(X\ \mathcal{S}_{\overline{\trianglelefteq}q}(\phi))$ is satisfied in every state, if it represents an *upper-bound* check for the probability measure of $(X\ \mathcal{S}_{\overline{\trianglelefteq}q}(\phi))$ (i.e. $up(\trianglelefteq, p)$ holds).

□

3.5.2 CSL syntax for ergodic models (no nesting of $\mathcal{S}_{\unlhd p}$).

The semantic equivalences described in Propositions 3.5.2-9, suggest a modified version of the CSL syntax introduced in Definition 2.3.5, which can be used to state properties referring to ergodic CTMCs. The main point with such a syntax is to keep *steady-state* formulae apart from the other logical connectives so that recursion is only allowed for boolean and *probabilistic-path* combinators. The steady-state connective $\mathcal{S}_{\unlhd p}$ can be applied, at top level only, to formulae which do not contain it.

Definition 3.5.1 (CSL syntax for ergodic CTMCs) *The syntax of CSL state-formulae (ϕ), boolean and probabilistic-formulae (ψ), path-formulae (φ) and steady-state formulae (ξ) is inductively defined as follows with respect to the set of atomic proposition AP:*

$$\begin{aligned}
\phi &::= \psi \mid \xi \mid \xi \wedge \psi \mid \psi \wedge \xi \mid \xi \wedge \xi \mid \neg \xi && \textit{(state-formulae)}\\
\psi &::= tt \mid a \mid \psi \wedge \psi \mid \neg \psi \mid \mathcal{P}_{\unlhd p}(\varphi) && \textit{(BP-formulae)}\\
\varphi &::= \mathcal{X}^I \psi \mid (\psi\ \mathcal{U}^I\ \psi) && \textit{(path-formulae)}\\
\xi &::= \mathcal{S}_{\unlhd p}(\psi) && \textit{(steady-state-formulae)}
\end{aligned} \tag{3.5.8}$$

where $a \in AP$, $p \in [0,1]$ is a real number, $\unlhd \in \{<, \leq, >, \geq\}$ and $I = [a,b] \subseteq \mathbb{R}_{\geq 0}$ is a non-empty interval.

Proposition 3.5.10 (Equivalent CSL syntax) *The language generated by the CSL syntax (Definition 2.3.5) is semantically equivalent to the one generated by the modified CSL syntax (Definition 3.5.1) given that the considered model is an ergodic CTMC.*

Proof. Straightforward consequence of Propositions 3.5.2-9.

□

Example 3.5.1 *Let us show how the equivalences described in Propositions 3.5.2-9 work in practice by considering some examples of CSL formulae involving the steady-state operator. The formulae in this example refer to the ergodic CTMCs of the GIS system of our running example (see Section 2.5.1). In Figure 3.6 the state-space of the*

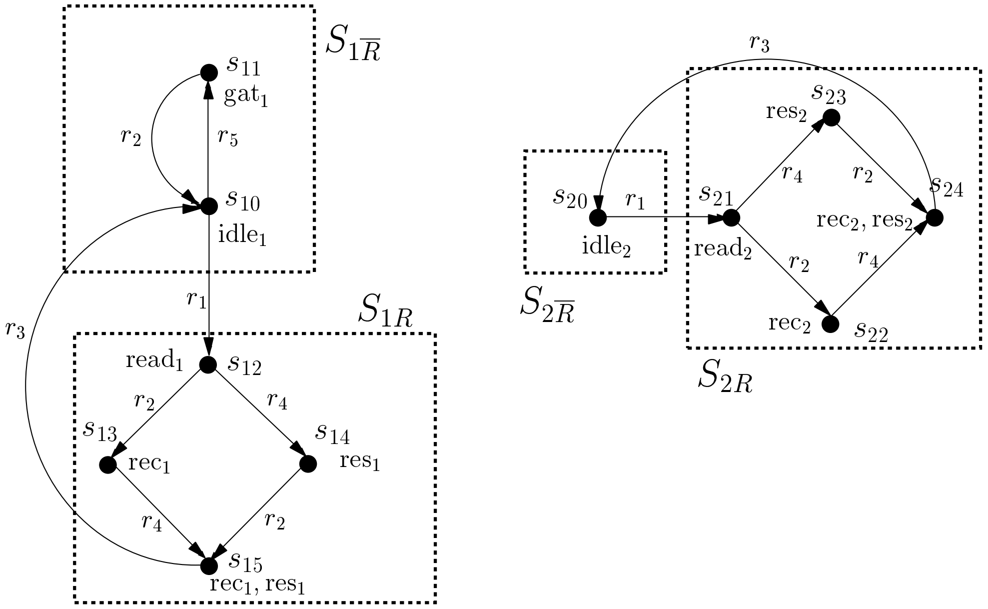


Figure 3.6: State space of the GIS system's components

two CTMCs is depicted. For illustrative purpose only[2]*, let us assume the following values for the steady-state distribution* π_1 *of component* $\mathcal{M}_1$*:*

$$\pi_1(s_{10}) = 0.3, \quad \pi_1(s_{11}) = \pi_1(s_{12}) = 0.2, \quad \pi_1(s_{13}) = \pi_1(s_{14}) = \pi_1(s_{15}) = 0.1$$

Furthermore let us suppose we are interested in the analysis of the steady-state probability of those states of $\mathcal{M}_1$ *which satisfy respectively the formulae* ϕ *and* ψ*:*

$$\phi \equiv idle_1 \qquad\qquad \psi \equiv rec_1 \vee res_1$$

The formula ψ *is valid in state* s_{13}, s_{14} *and* s_{15} *while* ϕ *is satisfied in* s_{10} *only. As a consequence the measure of the steady-state probability for* $Sat(\phi)$ *and* $Sat(\psi)$ *is, respectively, given by:*

$$\begin{aligned} \pi_1(s, (idle_1)) &= \pi_1(s_{10}) = 0.2 \\ \pi_1(s, (rec_1 \vee res_1)) &= \pi_1(s_{13}) + \pi_1(s_{14}) + \pi_1(s_{15}) = 0.3 \end{aligned}$$

[2]The values here assumed may turn out to be impossible with respect to the solution of the *steady-state* distribution π_1, as it is described in Example 2.5.1

independently of the considered state s (i.e. $\mathcal{M}_1$ is ergodic). In the following some examples of steady-state formulae are considered in order to illustrate the basic equivalences which have been proved in the initial part of this section.

Basic steady-state equivalences. *(application of Proposition 3.5.2 and Remark 3.4.2). Let us consider the following examples of CSL steady-state properties.*

$$\begin{array}{lll} i) & \mathcal{S}_{\geq 0.2}(\psi) & \equiv tt \\ ii) & \mathcal{S}_{\leq 0.2}(\psi) & \equiv \neg tt \\ iii) & \mathcal{S}_{\geq 0}(\psi) & \equiv \mathcal{S}_{\leq 1}(\psi) \equiv tt \end{array}$$

Case $i)$ is clearly satisfied in every state of $\mathcal{M}_1$ (i.e. is valid) as the steady-state probability for states satisfying ψ is $0.3 \geq 0.2$. For the same reason $ii)$ is never satisfied in $\mathcal{M}_1$, as 0.2 is not an upper bound for the steady-state probability of states satisfying ψ. Case $iii)$ shows an example of non-well-formed probabilistic formulae: both the pairs $(\geq, 0)$ and $(\leq, 1)$ result in pointless probabilistic formulae (i.e. tautologies), as clearly a probability measure muust fall in $[0,1]$.

$\mathcal{S}$ **nested in bounded Until** *(application of Proposition 3.5.6).*
Here some examples of time-bounded Until formulae with nested steady-state properties are provided. The operands of the Until will be chosen amongst the following three: $read_1$, $\mathcal{S}_{\overline{\trianglelefteq}q}(\psi))$ and $\mathcal{S}_{\tilde{\trianglelefteq}r}(\phi))$. Furthermore two different cases of bounding interval will be considered, namely $I = [0,5]$ or $I = [2,5]$, in order to points out the differences between a null and a non-null infinum.

Case 1.

$\mathcal{P}_{\trianglelefteq p}(read_1 \ \mathcal{U}^{[a,b]} \ \mathcal{S}_{\overline{\trianglelefteq}q}(\psi))$

$$\begin{array}{lll} i) & \mathcal{P}_{\geq 0.7}(read_1 \ \mathcal{U}^{[0,5]} \ \mathcal{S}_{\geq 0.2}(\psi)) & \equiv tt \\ ii) & \mathcal{P}_{\geq 0.7}(read_1 \ \mathcal{U}^{[2,5]} \ \mathcal{S}_{\geq 0.2}(\psi)) & \equiv [read_1 \wedge \mathcal{P}_{\geq 0.7}(read_1 \ \mathcal{U}^{[2,5]} \ tt)] \\ iii) & \mathcal{P}_{\leq 0.7}(read_1 \ \mathcal{U}^{[2,5]} \ \mathcal{S}_{\geq 0.2}(\psi)) & \equiv \neg read_1 \vee [read_1 \wedge \mathcal{P}_{\leq 0.7}(read_1 \ \mathcal{U}^{[2,5]} \ tt)] \\ iv) & \mathcal{P}_{\geq 0.7}(read_1 \ \mathcal{U}^{[2,5]} \ \mathcal{S}_{\leq 0.2}(\psi)) & \equiv \neg tt \end{array}$$

Case i) is clearly satisfied in every state since the proability measure of the target-formula, $\mathcal{S}_{\geq 0.2}(\psi)$, is itself satisfied in every state (see previous case). Hence every path *from every state satisfies the Until formula, which means:* $Prob(s,(read_1\ \mathcal{U}^{[0,5]}\ \mathcal{S}_{\geq 0.2}(\psi))) = 1 \geq 0.7$*, for every state s.*

In case ii) we are concerned with a lower-bound check for the probabiity measure of a time-bounded Until formula. We observe that since we have a non-null infinum (i.e. 2*) of the bounding interval, then a path* σ *must start in a* $read_1$ *state in order to satisfy* $(read_1\ \mathcal{U}^{[2,5]}\ \mathcal{S}_{\geq 0.2}(\psi))$*. Thus the conjunction* $[read_1 \wedge \mathcal{P}_{\geq 0.7}(read_1\ \mathcal{U}^{[2,5]}\ tt)]$ *rules out those state (i.e.* $\neg read_1$ *states) whose paths cannot satisfy* $(read_1\ \mathcal{U}^{[2,5]}\ \mathcal{S}_{\geq 0.2}(\psi))$*. In fact, for every state* $s \not\models read_1$ *the probability measure* $Prob(s,(read_1\ \mathcal{U}^{[2,5]}\ \mathcal{S}_{\geq 0.2}(\psi))) = 0 \not\geq 0.7$*.*

In case iii), we are considering an upper-bound check for the probability measure of the same Until formula. As a consequence, the disjunction $\neg read_1 \vee [read_1 \wedge \mathcal{P}_{\leq 0.7}(read_1\ \mathcal{U}^{[2,5]}\ tt)]$ *reflects the fact that in order for the probability measure* $Prob(s,(read_1\ \mathcal{U}^{[2,5]}\ \mathcal{S}_{\geq 0.2}(\psi)))$ *of a state s to meet the bound* ≤ 0.7 *it suffices* $s \models \neg read_1$*. The conjunntion* $[read_1 \wedge \mathcal{P}_{\leq 0.7}(read_1\ \mathcal{U}^{[2,5]}\ tt)]$*, instead, identifies those amongst the* $read_1$ *states whose measure* $Prob(s,(read_1\ \mathcal{U}^{[2,5]}\ \mathcal{S}_{\geq 0.2}(\psi))) \leq 0.7$*.*

Finally, the formula in case iv) is clearly unsatisfiable, as the target, $\mathcal{S}_{\leq 0.2}(\psi))$ *false in the model (see previous case of the example).*

In the remainder examples regarding the other possible way of nesting a steady-state property within a time-bounded operator, are illustrated. They are obtained by application of Proposition 3.5.6.

case 2. $\mathcal{P}_{\trianglelefteq p}(\mathcal{S}_{\overline{\trianglelefteq} q}(rec_1 \vee res_1)\ \mathcal{U}^{[a,b]}\ read_1)$

$$\begin{aligned}
\mathcal{P}_{\geq 0.7}(\mathcal{S}_{\geq 0.2}(rec_1 \vee res_1)\ \mathcal{U}^{[0,5]}\ read_1) &\equiv \mathcal{P}_{\geq 0.7}(\diamond^{[0,5]}\ read_1)\\
\mathcal{P}_{\geq 0.7}(\mathcal{S}_{\leq 0.2}(rec_1 \vee res_1)\ \mathcal{U}^{[0,5]}\ read_1) &\equiv read_1\\
\mathcal{P}_{\leq 0.7}(\mathcal{S}_{\leq 0.2}(rec_1 \vee res_1)\ \mathcal{U}^{[2,5]}\ read_1) &\equiv tt
\end{aligned}$$

case 3. $\mathcal{P}_{\trianglelefteq p}(\mathcal{S}_{\tilde{\trianglelefteq} r}(idle_1)\ \mathcal{U}^{[a,b]}\ \mathcal{S}_{\overline{\trianglelefteq} q}(rec_1 \vee res_1))$

$$\left.\begin{array}{l}\mathcal{P}_{\geq 0.7}(\mathcal{S}_{\leq 0.2}(idle_1)\ \mathcal{U}^{[0,5]}\ \mathcal{S}_{\geq 0.2}(rec_1 \vee res_1))\\ \mathcal{P}_{\geq 0.7}(\mathcal{S}_{\geq 0.2}(idle_1)\ \mathcal{U}^{[0,5]}\ \mathcal{S}_{\geq 0.2}(rec_1 \vee res_1))\end{array}\right\} \equiv tt$$

$$\left.\begin{array}{l}\mathcal{P}_{\leq 0.7}(\mathcal{S}_{\leq 0.2}(idle_1)\ \mathcal{U}^{[0,5]}\ \mathcal{S}_{\geq 0.2}(rec_1 \vee res_1))\\ \mathcal{P}_{\leq 0.7}(\mathcal{S}_{\geq 0.2}(idle_1)\ \mathcal{U}^{[0,5]}\ \mathcal{S}_{\geq 0.2}(rec_1 \vee res_1))\end{array}\right\} \equiv \neg tt$$

$\square$

Chapter 4

Compositional CSL model checking: non-Path formulae

4.1 Introduction

The Boucherie framework introduced in Chapter 2, provides a method to compose CTMCs which features a *product-form* solution for the composed model. In this chapter the analysis of a compositional semantics for CSL formulae referring to a two component Boucherie process is addressed. Unlike other works which base their results for "compositionally" checking the truth of a formula on the existence of a *preorder* relation between a model and its components [37], the results presented in this chapter depend on the compositional structure of the Boucherie process only: no *preorder* relation holds between a Boucherie process and its components.

Section 4.2 recalls the Boucherie framework description for the case of two components, introducing notations used throughout the rest of the chapter as well as some relevant background properties. In Section 4.3 a subset of the CSL syntax, in which probabilistic path formulae are not permitted, is considered and "compositional" equivalences are proved with respect to a two component Boucherie process. In Section 6.2 the considered logic is extended by allowing also probabilistic path formulae but, differently from the original CSL syntax, restraining the nesting capability of probabilistic operators: probabilistic path connective $\mathcal{P}$ can be nested only within a the *steady-state* operator $\mathcal{S}$, while the converse is not permitted. Compositional semantic equivalences for formulae of that "restricted" CSL logic are then proved in this section, relying on results obtained in the previous section.

4.2 The two component *Boucherie* framework

The n-dimensional Boucherie product-process has been formally introduced in section 2.5. Here we focus on the bidimensional case where $\mathcal{M} = (S, \mathbf{Q}, L)$ is a Boucherie process, with components[1] $\mathcal{M}_1 = (S_1, \mathbf{Q}_1, L_1)$ and $\mathcal{M}_2 = (S_2, \mathbf{Q}_2, L_2)$. The two independent processes $\mathcal{M}_1$ and $\mathcal{M}_2$ compete over a shared resource R.

Boucherie Bidimensional state-space. The state-space S_k of each component $k \in \{1,2\}$, is partitioned according to the resource possession: $S_{k,\overline{R}} \subset S_k$ represents the states where component k does not hold R, while $S_{k,R} \subset S_k$ are the states where $\mathcal{M}_k$ holds the resource.

$$
\begin{aligned}
S_1 &= S_{1,\overline{R}} \cup S_{1,R} \\
S_2 &= S_{2,\overline{R}} \cup S_{2,R}
\end{aligned}
$$

The product-process state-space S (Figure 4.1) is obtained by eliminating the states representing the simultaneous possession of the shared resource R from the Cartesian product of the components' state-spaces:

$$S = S_1 \times S_2 \setminus (S_{1,R} \times S_{2,R})$$

In the remainder, the notation R_1R_2 will be used to refer to the Cartesian product $(S_{1,R} \times S_{2,R})$, representing the *prohibited* area of states which has to be ruled out from the Boucherie state-space definition.

[1]From now on, unless otherwise stated, $\mathcal{M} = (S, \mathbf{Q}, L)$, will denote a generic bidimensional Boucherie process with components $\mathcal{M}_1 = (S_1, \mathbf{Q}_1, L_1)$ and $\mathcal{M}_2 = (S_2, \mathbf{Q}_2, L_2)$.

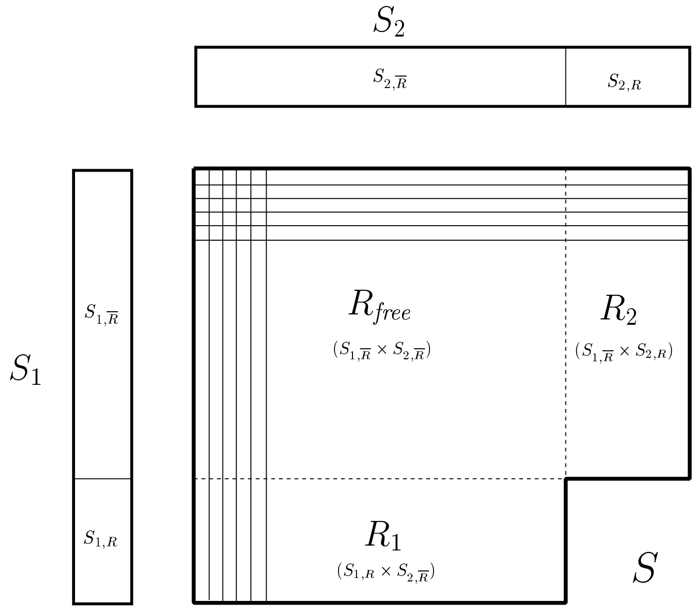


Figure 4.1: The bidimensional state-space of a two components Boucherie process.

Boucherie process transitions. Transitions of the Boucherie process are such that only *moves* along a single component direction are allowed (as components are supposedly independent, synchronisation is not allowed). As Figure 4.1 points out, in every state (s^1, s^2) belonging to the area of S where neither $\mathcal{M}_1$ nor $\mathcal{M}_2$ hold R (i.e. $(s^1, s^2) \in R_{free}$), both types of transition are permitted: either a 1-move (i.e. according to the behaviour of component $\mathcal{M}_1$) leading to a state (t^1, s^2) where the second component's state (s^2) is unchanged, or a 2-move leading to a state (s^1, t^2) where the first component's state (s^1) is unchanged.

On the other hand when a component holds R (i.e. $(s^1, s^2) \in R_k$) the only permitted behaviour is the one of the resource holder's process. Hence, for example, any state (t^1, t^2) reachable from a state $(s^1, s^2) \in R_1$ must be such that the $\mathcal{M}_2$ component state t^2 is unchanged: $t^2 = s^2$ (similarly any state (t^1, t^2) reachable from a state $(s^1, s^2) \in R_2$ must be such that $t^1 = s^1$). This is formally achieved through the definition of the infinitesimal generator matrix $\mathbf{Q}$ for a bidimensional Boucherie process, which is obtained from definition 2.5.2 by considering the two components only case (i.e.

$K = 2$):

$$\mathbf{Q}((s^1,s^2),(t^1,t^2)) = \begin{cases} \mathbf{Q}_1(s^1,t^1) & \textbf{if } s^2 = t^2 \wedge s^2 \notin S_{2,R} \\ \\ \mathbf{Q}_2(s^2,t^2) & \textbf{if } s^1 = t^1 \wedge s^1 \notin S_{1,R} \\ \\ 0 & otherwise \end{cases} \tag{4.2.1}$$

Boucherie process labelling. $L : S \rightarrow AP_1 \cup AP_2$ where

$$L((s^1,s^2)) = L_1(s^1) \cup L_2(s^2)$$

Throughout the remainder we will adopt the indices k and j, with $k, j \in \{1,2\}$, to allow reference to a generic component of a bidimensional Boucherie process and its dual: both j and k can represent either the first or second component but while k represents one j represents the other (i.e. $j = (k \; mod \; 2) + 1$). Furthermore, to ease the description, the notions of *k-move* and *k-projection* are introduced.

***k*-move.** A transition $(s^1,s^2) \longrightarrow (t^1,t^2)$ is called a k-move, if and only if

$$\mathbf{Q}((s^1,s^2),(t^1,t^2)) = \mathbf{Q}_k(s^k,t^k)$$

***k*-projection.** The *k-projection* of a state $(s^1,s^2) \in S$ is its k-th component: s^k.
For example:

s^1	is the 1-*projection* of	(s^1,s^2)
t^2	is the 2-*projection* of	(s^1,t^2)
$(s^1,s^2) \longrightarrow (t^1,s^2)$	is a 1-*move* given that	$\mathbf{Q}_1(s^1,t^1) \neq 0$

Here we are assuming that *self-loops* are not allowed in the Boucherie framework. Hence the *source* and *target* state of a transition in $\mathcal{M}_k$ are always different (i.e. $\forall s^k,t^k \in S_k$, $\mathbf{Q}_k(s^k,t^k) > 0 \Rightarrow s^k \neq t^k$).

Definition 4.2.1 (Probability of a k-move) . *Given a state $(s^1,s^2) \in S$ of a Boucherie process, the probability of a k-move out of (s^1,s^2) is given by:*

$$p^k(s^1,s^2) = \begin{cases} \frac{E_k(s^k)}{E_k(s^k)+E_j(s^j)} & \textbf{if} \quad (s^1,s^2) \in R_{free} \\ 1 & \textbf{if} \quad (s^1,s^2) \in R_k \\ 0 & \textbf{if} \quad (s^1,s^2) \in R_j \end{cases}$$

So for example for any state (s^1,s^2) where neither $\mathcal{M}_1$ nor $\mathcal{M}_2$ possesses R, the probability of a 1-*move* is $p^1(s^1,s^2) = \frac{E_1(s^1)}{E_1(s^1)+E_2(s^2)}$ while the probability for a 2-*move* is $p^2(s^1,s^2) = \frac{E_2(s^2)}{E_1(s^1)+E_2(s^2)}$.

As a consequence of the transition rate for a bidimensional Boucherie process (see equation 4.2.1), two trivial Remarks regarding the emanating rate of states and the probability of transitions, can be straightforwardly derived.

Remark 4.2.1 *Let $\mathcal{M} = (S,\mathbf{Q},L)$ be a bidimensional Boucherie process, the emanating rate of a state $(s^1,s^2) \in S$ is given by the sum of the emanating rates of its components, if the resource R is free in (s^1,s^2), or by the emanating rate of the holder of R, if R is not free.*

$$E\left((s^1,s^2)\right) = \begin{cases} E_1(s^1)+E_2(s^2) & \textbf{if} \;\; (s^1,s^2) \in R_{free} \\ \\ E_1(s^1) & \textbf{if} \;\; (s^1,s^2) \in R_1 \\ \\ E_2(s^2) & \textbf{if} \;\; (s^1,s^2) \in R_2 \end{cases} \tag{4.2.2}$$

The above Remark provides a compositional way to obtain the total rate out of a state of a bidimensional Boucherie process: the emanating rate of any state (s^1,s^2) is obtained from the emanating rate of its *components*. In the next Remark, instead, the probability of the Boucherie process transitions is characterised.

Remark 4.2.2 *Let $\mathcal{M} = (S,\mathbf{Q},L)$ be a bidimensional Boucherie process, then the* *probability of a transition from a state $(s^1,s^2) \in S$ to a state $(t^1,t^2) \in S$, is given by:*

$$\mathbf{P}\left((s^1,s^2),(t^1,t^2) \right) = \begin{cases} \frac{\mathbf{Q}_1(s^1,t^1)}{E_1(s^1)+E_2(s^2)} & \textbf{if } (s^1,s^2) \in R_{free} \wedge (s^2 = t^2) \\ \\ \frac{\mathbf{Q}_2(s^2,t^2)}{E_1(s^1)+E_2(s^2)} & \textbf{if } (s^1,s^2) \in R_{free} \wedge (s^1 = t^1) \\ \\ \frac{\mathbf{Q}_1(s^1,t^1)}{E_1(s^1)} & \textbf{if } (s^1,s^2) \in R_1 \\ \\ \frac{\mathbf{Q}_2(s^2,t^2)}{E_2(s^2)} & \textbf{if } (s^1,s^2) \in R_2 \end{cases} \tag{4.2.3}$$

or equivalently

$$\mathbf{P}\left((s^1,s^2),(t^1,t^2) \right) = \begin{cases} \mathbf{P}_1(s^1,t^1)\cdot p^1(s^1,s^2) & \textbf{if } (s^1,s^2) \in R_{free} \wedge (s^2 = t^2) \\ \\ \mathbf{P}_2(s^2,t^2)\cdot p^2(s^1,s^2) & \textbf{if } (s^1,s^2) \in R_{free} \wedge (s^1 = t^1) \\ \\ \mathbf{P}_1(s^1,t^1) & \textbf{if } (s^1,s^2) \in R_1 \\ \\ \mathbf{P}_2(s^2,t^2) & \textbf{if } (s^1,s^2) \in R_2 \end{cases} \tag{4.2.4}$$

The second formulation of the above Remark (4.2.4) points out how the probability for a transition $(s^1,s^2) \to (t^1,t^2)$ can be determined compositionally, in terms of the probability of the corresponding component's transition. Whenever the *source* state (s^1,s^2) is in R_{free} the transition's probability is given by a factor of the probability of the corresponding $\mathcal{M}_k$'s transition (i.e. $s^k \longrightarrow t^k$) probability, given that $(s^1,s^2) \longrightarrow (t^1,t^2)$ is a *k-move*. On the other hand if the *source* state (s^1,s^2) is in R_k the only possible transitions are *k-move*s, hence the probability of $(s^1,s^2) \longrightarrow (t^1,t^2)$ is equal to the probability of its corresponding $\mathcal{M}_k$'s transition (i.e. $s^k \longrightarrow t^k$).

Product-form. The steady-state distribution π of a bidimensional Boucherie process is given by the product of the steady-state distributions of its components. For every state $(s^1,s^2) \in S$ the following holds:

$$\pi(s^1, s^2) = G \cdot \pi_1(s^1) \cdot \pi_2(s^2)$$

where $\pi_k(s^k)$ is the probability for component $\mathcal{M}_k$ to be in state s^k on the long-run, while G is a *normalisation* constant.

On subsets of components' state-space. Concerning the Boucherie framework we introduce a specific notation to characterise subsets of the component's state-space according to the state space partition. Thus given a subset $A_k \subseteq S_k$ of component $\mathcal{M}_k$ state-space, the two parts it consists of are denoted, respectively, $A_{k,\overline{R}} \subseteq S_{k,\overline{R}}$ and $A_{k,R} \subseteq S_{k,R}$.

Given two subsets, respectively $A_1 \subseteq S_1$ and $A_2 \subseteq S_2$, the intersection of their product with the Boucherie state-space S, can be decomposed into the union of two sub-products, as the following Remark points out.

Remark 4.2.3 *Let $A_1 \subseteq S_1$ and $A_2 \subseteq S_2$ be two subsets with respect to the components of a bidimensional Boucherie process with state-space S, then the following holds:*

$$(A_1 \times A_2) \cap S = (A_{1,\overline{R}} \times A_2) \cup (A_{1,R} \times A_{2,\overline{R}}) = (A_1 \times A_{2,\overline{R}}) \cup (A_{1,\overline{R}} \times A_{2,R})$$

Figure 4.2 shows what Remark 4.2.3 is meant to point out: the part of the product of two subsets A_1 and A_2 which falls in S allows for a bi-partition which is susceptible for two different but equivalent characterisations. In the first characterisation (Figure 4.2.a) one part is obtained by coupling every state of A_1 where $\mathcal{M}_1$ does not hold R (i.e. $A_{1,\overline{R}}$) with every state of A_2, while the other part, is given by coupling the states of A_1 where $\mathcal{M}_1$ holds R (i.e. $A_{1,R}$) only with the states of A_2 where $\mathcal{M}_2$ does not hold R (i.e. so that the mutual-exclusion condition is not breached). In the second characterisation (Figure 4.2.b) the first part is obtained by coupling every state of A_1 with every state of A_2 such that $\mathcal{M}_2$ does not hold R (i.e. $A_{2,\overline{R}}$), while the second part, is given by coupling the states of A_1 where $\mathcal{M}_1$ does not hold R (i.e. $A_{1,\overline{R}}$) only with the states of A_2 such that $\mathcal{M}_2$ holds R (i.e. $A_{2,R}$).

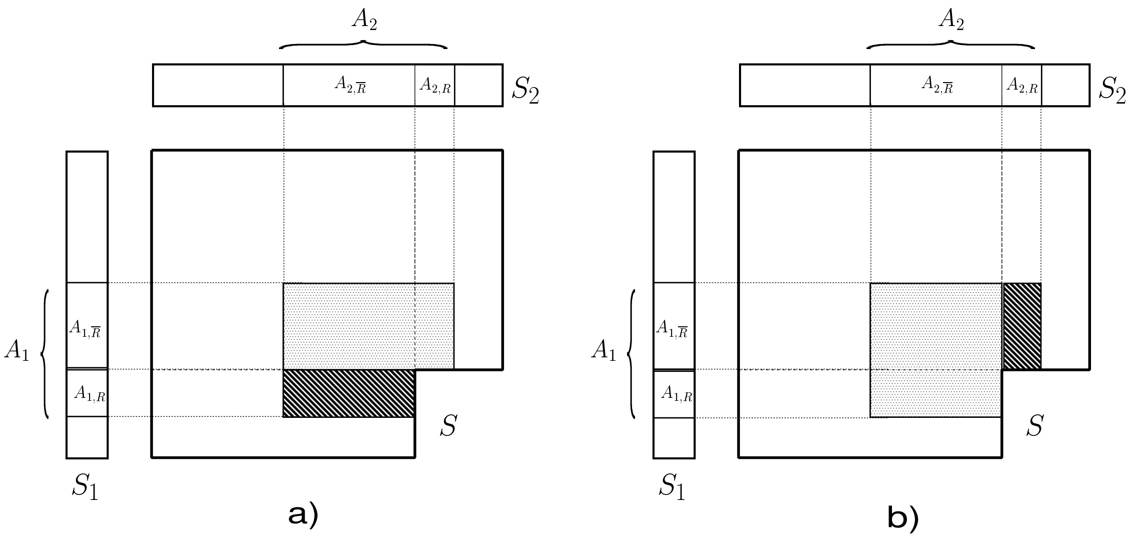


Figure 4.2: $(A_1 \times A_2) \cap S = (A_{1,\overline{R}} \times A_2) \cup (A_{1,R} \times A_{2,\overline{R}}) = (A_1 \times A_{2,\overline{R}}) \cup (A_{1,\overline{R}} \times A_{2,R})$.

Satisfiability sets $Sat(\phi), Sat_k(\phi), Sat_{k,\overline{R}}(\phi), Sat_{k,R}(\phi)$**.** If $\mathcal{M} = (S, \mathbf{Q}, L)$ is a bidimensional Boucherie process with components $\mathcal{M}_1 = (S_1, \mathbf{Q}_1, L_1)$ and $\mathcal{M}_2 = (S_2, \mathbf{Q}_2, L_2)$ and ϕ is a CSL formula the following notations will be adopted throughout the remaining parts of this chapter:

- $Sat(\phi) \subseteq S$ denotes the subset of $\mathcal{M}$'s states which validate ϕ.

- $Sat_k(\phi) \subseteq S_k$ denotes the subset of component $\mathcal{M}_k$'s states which validate ϕ.

- $Sat_{k,\overline{R}}(\phi) \subseteq S_k$ denotes the subset of $Sat_k(\phi)$ consisting only of states where $\mathcal{M}_k$ does not hold R.

- $Sat_{k,R}(\phi) \subseteq S_k$ denotes the subset of $Sat_k(\phi)$ consisting only of states where $\mathcal{M}_k$ holds R.

As will soon be clear, the above characterisation is important in order to find a decompositional expression for the set $Sat(\phi)$.

4.2.1 Partitioning the Atomic Propositions set

In order to find a compositional semantics for formulae of the CSL with respect to bidimensional Boucherie processes, the set of atomic propositions AP on which the formulae are built needs to be partitioned. Since the components of a Boucherie framework are independent processes it is sensible to keep the atomic propositions referring to one process separated from the ones referring to the other process.

$$AP = AP_1 \cup AP_2$$

where AP_k denotes the atomic propositions for component $\mathcal{M}_k$.

As a consequence of the AP's partition, the CSL formulae referring to a bidimensional Boucherie process can also be distinguished according to the number of components they refer to. We will call *single-component formulae*, the formulae which state properties concerning one component only (i.e. those formulae involving atomic proposition of one part only, either AP_1 or AP_2) as opposed to *general formulae*, namely those formulae which refer to both components (i.e. formulae for which atomic propositions of both components are involved). The following two definitions, provide a formalisation of these concepts.

Definition 4.2.2 (Single-component formulae) *Let $\mathcal{M} = (S, \mathbf{Q}, L)$ be a bidimensional Boucherie process labelled over the set $AP = AP_1 \cup AP_2$, where AP_k is the atomic propositions set for component $\mathcal{M}_k$. The formulae ϕ_k characterised by the following syntax are CSL single-component state formulae:*

$$\begin{array}{lll} \phi_k & := a_k \mid tt \mid \neg\phi_k \mid \ \phi_k \wedge \phi_k \mid \mathcal{S}_{\trianglelefteq p}(\phi_k) \mid \mathcal{P}_{\trianglelefteq p}(\varphi_k) & \textit{(state-formulae)} \\ \varphi_k & := \mathcal{X}^I \phi_k \mid \phi_k \ \mathcal{U}^I \phi_k & \textit{(path-formulae)} \end{array} \tag{4.2.5}$$

where $a_k \in AP_k$.

Definition 4.2.3 (General formulae) *Let $\mathcal{M} = (S, \mathbf{Q}, L)$ be a bidimensional Boucherie process labelled over the set $AP = AP_1 \cup AP_2$, where AP_k is the atomic propositions set for component $\mathcal{M}_k$. The formulae ϕ_{12} characterised by the following syntax are CSL general formulae:*

$$\begin{array}{lll} \phi_{12} & := \phi_1 \wedge \phi_2 \mid \phi_2 \wedge \phi_1 \mid \neg\phi_{12} \mid \ \mathcal{S}_{\trianglelefteq p}(\phi_{12}) \mid \mathcal{P}_{\trianglelefteq p}(\varphi_{12}) & \textit{(state-formulae)} \\ \varphi_{12} & := \mathcal{X}^I \phi_{12} \ \mid \phi_{j12} \ \mathcal{U}^I \phi_{12} \ \mid \phi_{12} \ \mathcal{U}^I \phi_k \ \mid \phi_k \ \mathcal{U}^I \phi_{j12} & \textit{(path-formulae)} \end{array} \tag{4.2.6}$$

where ϕ_k are CSL single-component.state formulae as in Definition 4.2.2.

As a consequence of the partition of the atomic proposition set AP the set of CSL state formulae Φ for a bidimensional Boucherie process is is also partitioned in the following manner:

$$\Phi = \Phi_1 \cup \Phi_2 \cup \Phi_{12}$$

Each formula $\phi \in \Phi$ can be classified as *single-component* or *general* by means of a function, named $\mathcal{A}t()$, which returns the set of atoms ϕ is built on.

Definition 4.2.4 (Atoms of a CSL formula) *Let AP be a set of atomic propositions and ϕ a CSL formula built on AP. The function $\mathcal{A}t() : \Phi \cup \overline{\varphi} \rightarrow AP$, is defined as follows:*

$$\mathcal{A}t(\phi) \quad = \quad \begin{cases} a & \textbf{if } \phi \equiv a \\ \mathcal{A}t(\phi') \cup \mathcal{A}t(\phi'') & \textbf{if } \phi \equiv \phi' \wedge \phi'' \\ \mathcal{A}t(\phi') & \textbf{if } \phi \equiv \neg\phi' \\ \mathcal{A}t(\phi') & \textbf{if } \phi \equiv \mathcal{S}_{\unlhd p}(\phi') \\ \mathcal{A}t(\varphi) & \textbf{if } \phi \equiv \mathcal{P}_{\unlhd p}(\varphi) \end{cases}$$

$$\mathcal{A}t(\varphi) \quad = \quad \begin{cases} \mathcal{A}t(\phi') & \textbf{if } \varphi \equiv \mathcal{X}\phi' \\ \mathcal{A}t(\phi') \cup \mathcal{A}t(\phi'') & \textbf{if } \varphi \equiv \phi' \mathcal{U} \phi'' \end{cases}$$

where $\overline{\varphi}$ denotes the set of path formulae φ built on AP.

Relying on the above Definition the classification for the formulae ϕ referring to a bidimensional Boucherie process can straightforwardly be obtained as follows:

$$\phi = \begin{cases} \phi_k & \textbf{iff } \ \mathcal{A}t(\phi) \subseteq AP_k \\ \\ \phi_{12} & \textbf{iff } \ [(\mathcal{A}t(\phi) \cap AP_1) \neq \emptyset] \wedge [(\mathcal{A}t(\phi) \cap AP_2) \neq \emptyset] \end{cases} \tag{4.2.7}$$

In the next section we deal with the problem of deriving a compositional semantics for *single-component* formulae. The results there obtained will be then exploited throughout the subsequent section, where the study of a compositional way to check *general* formulae is faced.

4.3 Model checking *non-Probabilistic* state formulae

In this section a restricted syntax of the CSL logic is considered and a compositional semantics, with respect to a bidimensional Boucherie process, is proved for the formulae belonging to it. The logic is obtained from the one described in Definition 2.3.5, by disallowing probabilistic path formulae. Essentially only Boolean combinations of atomic-propositions and *steady-state* formulae are permitted; furthermore the *steady-state* connective $\mathcal{S}_{\trianglelefteq p}$ cannot be nested (i.e. it can appear at top-level only). This last feature, though, does not affect the logic expressiveness, since models in a Boucherie framework are ergodic CTMCs. Hence, as proved by Proposition 3.5.10, the expressiveness is unchanged: the logic given by eliminating $\mathcal{P}_{\trianglelefteq p}(\varphi)$ from Definition 2.3.5 is semantically equivalent to the one characterised by the following syntax.

$$\begin{aligned}
\phi &::= \psi \ | \ \xi \ | \ \phi \wedge \phi \ | \ \neg\phi \\
\psi &::= \ tt \ | \ a \ | \ \psi \wedge \psi \ | \ \neg\psi \\
\xi &::= \mathcal{S}_{\trianglelefteq p}(\psi)
\end{aligned} \tag{4.3.1}$$

The remainder of this section is split in two parts: the first is devoted to proving the existence of a compositional semantics for *single-component* formulae; the second involves the analysis of the *general* formulae case.

4.3.1 Compositional semantics for *single-component* formulae

We take into account only formulae which are built on atomic propositions belonging to AP_k ($k \in \{1,2\}$). The syntax for *single-component* formulae, results from the one described by equation 4.3.1:

$$\begin{aligned}
\phi_k &::= \psi_k \mid \xi_k \mid \phi_k' \wedge \phi_k'' \mid \neg\phi_k \\
\psi_k &::= tt \mid a_k \mid \psi_k \wedge \psi_k \mid \neg\psi_k \\
\xi_k &::= \mathcal{S}_{\trianglelefteq p}(\psi_k)
\end{aligned} \tag{4.3.2}$$

Given a state (s^1, s^2) of the product-process, we aim to prove that there exists a *transformation* function $f_t : \Phi_k \to \Phi_k$, which applied to a *single-component* formula ϕ_k, returns another *single-component* formula $f_t(\phi_k)$, such that:

$$(s^1, s^2) \models \phi_k \iff s^k \models_k f_t(\phi_k) \tag{4.3.3}$$

where $\models_k$ denotes the semantics relationship with respect to component $\mathcal{M}_k$. The equivalence 4.3.3 provides us with a compositional semantics for *single-component* state-formulae ϕ_k: checking the validity of ϕ_k with respect to a state (s^1, s^2) of a bidimensional Boucherie process is equivalent to check the validity of a *derived* formula $f_t(\phi_k)$ with respect to the *projection* state s^k of component $\mathcal{M}_k$.
As one can easily understand, the main issue with the characterisation of "compositionally" equivalent *single-component* formulae, concerns *steady-state* formulae, namely $\mathcal{S}_{\trianglelefteq p}(\psi_k)$. It will be shown that characterising the equivalence for $\mathcal{S}_{\trianglelefteq p}(\psi_k)$ concerns the derivation of an *equivalent probability bound* $\hat{p}$ which depends on the original one, i.e. p, as well as on the argument formula ψ_k and on the component $\mathcal{M}_k$ it refers to, which is: a formula $\mathcal{S}_{\trianglelefteq p}(\psi_k)$ is valid in a bidimensional Boucherie $\mathcal{M}$ if and only if $\mathcal{S}_{\trianglelefteq \hat{p}}(\psi_k)$ is valid with respect to the component $\mathcal{M}_k$.

The characterisation of the *equivalent probability bound* for *single-component steady-state* formulae, is given by a function named $g()$ whose definition follows.

Definition 4.3.1 (Equivalent steady-state probability bound function $g()$) *Let $\psi_k \in \Psi_k$ be a single-component formula , where Ψ_k is the set of ψ_k formulae described by equation 4.3.2; the function $g() : [0,1] \times \Psi_k \times CTMC \to [0,1]$ is defined as:*

$$g(p,\psi_k,\mathcal{M}_k) = \begin{cases} \frac{p}{G} & \textbf{if } Sat_k(\psi_k) \subseteq S_{k,\overline{R}} \\ \frac{p}{G\cdot C_j} & \textbf{if } Sat_k(\psi_k) \subseteq S_{k,R} \\ \frac{p-G\cdot C_{kj}}{G\cdot C_j} & \textbf{if } (Sat_{k,\overline{R}}(\psi_k) \neq \emptyset) \wedge (Sat_{k,R}(\psi_k) \neq \emptyset) \end{cases} \tag{4.3.4}$$

where G is the product-form normalisation *constant, and C_j and C_{kj} are respectively defined as follows:*

$$\begin{aligned} C_j &= \sum_{t^j \in S_{j,\overline{R}}} \pi_j(t^j) \\ C_{kj} &= \sum_{s^k \in Sat_{k,\overline{R}}(\psi_k)} \pi_k(s^k) \sum_{s^j \in S_{j,R}} \pi_j(s^j) = \Big[\sum_{s^k \in Sat_{k,\overline{R}}(\psi_k)} \pi_k(s^k) \Big] \cdot (1 - C_j) \end{aligned} \tag{4.3.5}$$

The function g distinguishes between three different cases depending on $Sat_k(\psi_k)$. The proof of correctness for $g()$ will be faced in Theorem 4.3.1, here an informal description of the intuitions it relies on is given. Figure 4.3 provides a graphical description of the three possible cases concerning the satisfiability set for a formula ψ_1 with respect to component $\mathcal{M}_1$: $Sat_1(\psi_1) \subseteq S_{1,\overline{R}}$ (Figure 4.3.a), $Sat_1(\psi_1) \subseteq S_{1,R}$ (Figure 4.3.c) or $[Sat_1(\psi_1) \cap S_{1,\overline{R}}] \neq \emptyset$ and $[Sat_1(\psi_1) \cap S_{1,R}] \neq \emptyset$ (Figure 4.3.b). The main point in determining the *equivalent probability bound* for a *steady-state* formula $\mathcal{S}_{\trianglelefteq p}(\psi_k)$, is to find a decomposition of the set $Sat(\psi_k)$ in terms of products of subsets of the components' state-space S_k and S_j.

- if ψ_k is satisfied only in states where component $\mathcal{M}_k$ does not hold R (i.e. $Sat_k(\psi_k) \subseteq S_{k,\overline{R}}$, see figure 4.3.a) then it is straightforward to prove that the set of $\mathcal{M}$'s states satisfying ψ_k is given by[2] $Sat(\psi_k) = Sat_k(\psi_k) \times S_j$. In such a case, as will be proven by Theorem 4.3.1, the *equivalent probability bound* $g(p,\psi_k,\mathcal{M}_k)$ is given by $\frac{p}{G}$.

[2]Here we are assuming $k = 1$ and $j = 2$; clearly if $k = 2$ and $j = 1$, then $Sat(\psi_k) = S_j \times Sat_k(\psi_k)$.

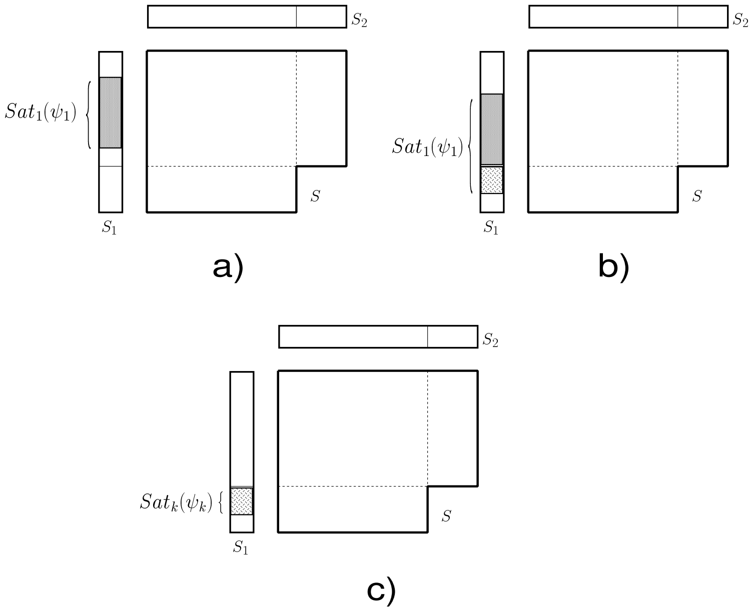


Figure 4.3: Three possibilities concerning $Sat_1(\psi_1)$.

- on the other hand, if ψ_k is satisfied only in states where component $\mathcal{M}_k$ retains R (i.e. $Sat_k(\psi_k) \subseteq S_{k,R}$, see figure 4.3.c) then $Sat(\psi_k) = Sat_k(\psi_k) \times S_{j,\overline{R}}$. As a consequence, (see Theorem 4.3.1), the *equivalent probability bound* $g(p, \psi_k, \mathcal{M}_k)$, is given by $\frac{p}{G \cdot C_j}$. It is relevant to note that the constant C_j represents the long-run probability *of not holding the shared resource* for component $\mathcal{M}_j$. Thus, the meaning of $g(p, \psi_k, \mathcal{M}_k)$ in this case is as follows: checking the *steady-state* probability of ψ_k against a probability bound p with respect to $\mathcal{M}$ is equivalent to check the *steady-state* probability of ψ_k with respect to $\mathcal{M}_k$, against a derived bound whose value depends on the probability that $\mathcal{M}_j$ *does not hold* R in the long-run.

- the most complex situation is when ψ_k is valid both in states where $\mathcal{M}_k$ does and does not holds R (i.e. $(Sat_{k,R}(\psi_k) \neq \emptyset) \wedge (Sat_{k,\overline{R}}(\psi_k) \neq \emptyset)$ see figure 4.3.b) then $Sat(\psi_k)$ is decomposable in two parts $Sat(\psi_k) = [Sat_{k,\overline{R}}(\psi_k) \times S_j] \cup [Sat_{k,R}(\psi_k) \times S_{j,\overline{R}}]$; the first one is given by coupling all those states where $\mathcal{M}_k$ does not hold R but satisfies ψ_k (i.e. $Sat_{k,\overline{R}}(\psi_k)$) with any state of $\mathcal{M}_j$ (i.e. S_j); the second part is obtained by coupling the states where $\mathcal{M}_k$ holds R and satisfies ψ_k (i.e.

$Sat_{k,R}(\psi_k)$), only with those states where $\mathcal{M}_j$ does not hold R (i.e. $S_{j,\overline{R}}$): the states of $S_{j,R}$ must be ruled out in order not to breach the mutually exclusive access to R. The *equivalent probability bound* $g(p,\psi_k,\mathcal{M}_k)$, in that case, is given by $\frac{p-G\cdot C_{kj}}{G\cdot C_j}$. The constant C_{kj} represents the long-run probability for component $\mathcal{M}_j$ *to hold the shared resource* weighted by the probability for component $\mathcal{M}_k$ to satisfy ψ_k while not holding R at steady-state. Summarising, in this case, checking $\mathcal{S}_{\trianglelefteq p}(\psi_k)$ with respect to a bidimensional Boucherie $\mathcal{M}$ is equivalent to check $\mathcal{S}_{\trianglelefteq\hat{p}}(\psi_k)$ with respect to $\mathcal{M}_k$, where the *equivalent probability bound* $\hat{p}$ depends on two factors: the probability of $\mathcal{M}_j$ to retain R in the long-run and the probability of $\mathcal{M}_k$ to not satisfy ψ_k while not holding R in the long-run.

The formal definition of the *transformation function* for the the *single-component* formulae, can be now introduced.

Definition 4.3.2 (Transformation function $f_t()$) *Let $\phi_k \in \Phi_k$ be a single-component formula, where Φ_k is the set of single-component formulae ϕ_k described in equation 4.3.2; the transformation function $f_t() : \Phi_k \rightarrow \Phi_k$ is defined as:*

$$f_t(\phi_k) = \begin{cases} \phi_k & \textbf{if } \phi_k \equiv \psi_k \\ \\ \neg f_t(\phi'_k) & \textbf{if } \phi_k \equiv \neg\phi'_k \\ \\ f_t(\phi'_k) \wedge f_t(\phi''_k) & \textbf{if } \phi_k \equiv (\phi'_k \wedge \phi''_k) \\ \\ \mathcal{S}_{g(p,\psi_k,\mathcal{M}_k)}(\psi_k) & \textbf{if } \phi_k \equiv \xi_k \equiv \mathcal{S}_{\trianglelefteq p}(\psi_k) \end{cases} \tag{4.3.6}$$

Having introduced the *transformation function* $f_t()$, the next step is to prove that it is actually correct (i.e. it provides us with formulae that are "compositionally" equivalent to the *transformed* one). This is the result of the Theorem 4.3.1. Before proceeding with proving the correctness of the *transformation function*, a minor, but relevant result, concerning the characterisation of the satisfiability set $Sat(\phi_k)$ needs to be shown and is done in the following Lemma.

Firstly, though, a notational peculiarity which is assumed in the following, deserves to be clarified: in general $k, j \in \{1,2\}$ are interchangeable indices used to distinguish elements (i.e. sets of states, formulae, set of atomic propositions ...) referring to the components of a bidimensional Boucherie process. In the following, the Cartesian product of subsets of the two components, like, for example, $Sat_k(\psi_k)$ and S_j, even though asymmetric, always appear in the form $Sat_k(\psi_k) \times S_j$ (i.e. with k as the first operand of the product and j the second), which with respect to the Boucherie framework is proper only if $k = 1, j = 2$ is assumed. Nevertheless in the following wherever a result involving the Cartesian product of a k subset times a j (i.e. k first operand and j second operand of the product) is shown, that result also holds in the dual case $k = 2, j = 1$: in such a case, a product like $Sat_k(\psi_k) \times S_j$ has to be intended as if reversed, which is $S_j \times Sat_k(\psi_k)$.

Lemma 4.3.1 *Let $\mathcal{M}$ be a bidimensional Boucherie process and ϕ_k a* single component *state-formula from the syntax described by equation (4.3.2), then the following implication holds:*

$$\left[(s^1,s^2) \models \phi_k \Leftrightarrow s^k \models_k f_t(\phi_k)\right] \Longrightarrow Sat(\phi_k) = (Sat_k(f_t(\phi_k)) \times S_j) \cap S$$

Proof. The equality $Sat(\phi_k) = (Sat_k(f_t(\phi_k)) \times S_j) \cap S$ has to be shown assuming $(s^1,s^2) \models \phi_k \Leftrightarrow s^k \models_k f_t(\phi_k)$ as hypothesis.

($\Rightarrow$) let us suppose that $(s^1,s^2) \in Sat(\phi_k)$ then, $(s^1,s^2) \models \phi_k$ hence from the hypothesis, also $s^k \models_k f_t(\phi_k) \Longrightarrow s^k \in Sat_k(f_t(\phi_k))$. Furthermore, since obviously $Sat(\phi_k) \subseteq S$, then $(s^1,s^2) \in S$, hence clearly $(s^1,s^2) \in (Sat_k(f_t(\phi_k)) \times S_j) \cap S$, which proves that any state $(s^!,s^2) \in Sat(\phi_k)$ is also a state $(s^1,s^2) \in [(Sat_k(f_t(\phi_k)) \times S_j) \cap S]$.

($\Leftarrow$) if $(s^1,s^2) \in [(Sat_k(f_t(\phi_k)) \times S_j) \cap S]$ then $(s^1,s^2) \in S \wedge s^k \in Sat_k(f_t(\phi_k))$. Hence, from the hypothesis, also $(s^1,s^2) \in Sat(\phi_k)$, which proves that any state $(s^1,s^2) \in [(Sat_k(f_t(\phi_k)) \times S_j) \cap S]$ is also a state $(s^1,s^2) \in Sat(\phi_k)$, hence the equality $Sat(\phi_k) = (Sat_k(f_t(\phi_k)) \times S_j) \cap S$.

□

Lemma 4.3.1 shows that by assuming the *transformation function*'s correctness (i.e. $\left[(s^1,s^2) \models \phi_k \Leftrightarrow s^k \models_k f_t(\phi_k)\right]$), the satisfiability set $Sat(\phi_k)$ for a *single-component* formula can be decomposed in terms of the satisfiability set, with respect to component $\mathcal{M}_k$, of the corresponding *transformed* formula $f_t(\phi_k)$ (i.e. $Sat_k(\phi_k)$). This result turns out to be useful in the proving the following Theorem.

Theorem 4.3.1 *Let $\mathcal{M}$ be a bidimensional Boucherie process and ϕ_k a* single component *state-formula as in (4.3.2), then the following implication holds:*

$$(s^1,s^2) \models \phi_k \iff s^k \models_k f_t(\phi_k) \quad \forall (s^1,s^2) \in S$$

where f_t is defined as in (4.3.6).

Proof. By structural induction over the form of ϕ_k.
base case: $\phi_k \equiv \psi_k \equiv a_k$.
The proof is trivial. From Definition 4.3.2 we have that $f_t(a_k) = a_k$. Furthermore an atomic proposition a_k labels a state (s^1,s^2) of the Boucherie process if and only if it labels the state s^k of component $\mathcal{M}_k$ (i.e. $a_k \in L((s^1,s^2) \iff a_k \in L_k(s^k))$. Hence, clearly

$$(s^1,s^2) \models a_k \iff s^k \models_k a_k.$$

induction step: all the other cases have to be considered.

1. $\phi_k \equiv \neg\psi_k$.
Let us assume the following *inductive hypothesis*:

$$(s^1,s^2) \models \psi_k \iff s^k \models_k f_t(\psi_k) \quad \forall (s^1,s^2) \in S \tag{4.3.7}$$

We aim to show that the following bi-implication:

$$(s^1,s^2) \models \neg\psi_k \iff s^k \models_k \neg f_t(\psi_k).$$

($\Rightarrow$) if $(s^1,s^2) \models \neg\psi_k$ then clearly $(s^1,s^2) \notin Sat(\psi_k)$, hence by *inductive hypothesis* (i.e. 4.3.7) also $s^k \notin Sat_k(f_t(\psi_k))$, then clearly $s^k \in Sat_k(\neg f_t(\psi_k)) \Longrightarrow s^k \models_k \neg f_t(\psi_k)$.

($\Leftarrow$) if $s^k \models_k \neg f_t(\psi_k)$ then $s^k \notin Sat_k(f_t(\psi_k))$. Hence, by *inductive hypothesis* (i.e. 4.3.7), also $(s^k, s^j) \notin Sat(\psi_k), \forall s^j \in S_j : (s^k, s^j) \in S$. But then clearly $(s^k, s^j) \in Sat(\neg\psi_k)$ which means $(s^k, s^j) \models \neg\psi_k$, $\forall s^j \in S_j : (s^k, s^j) \in S$.

For the sake of simplicity, we will refer the proof of the remainder of this Theorem to *single-component* formulae ϕ_k which refer to component $\mathcal{M}_1$ (i.e. $k = 1$), having in mind that the argument can straightforwardly be reversed to the case of formulae referring to component $\mathcal{M}_2$ (i.e. $k = 2$).

2. $\phi_1 \equiv \psi_1' \wedge \psi_1''$.
From Definition 4.3.2, we have that $f_t(\psi_1' \wedge \psi_1'') = f_t(\psi_1') \wedge f_t(\psi_1'')$. Let us assume the following *inductive hypothesis*:

$$(s^1, s^2) \models \psi_1' \Leftrightarrow s^1 \models_1 f_t(\psi_1') \ \textbf{and} \ (s^1, s^2) \models \psi_1'' \Leftrightarrow s^1 \models_1 f_t(\psi_1'') \tag{4.3.8}$$

for all states $(s^1, s^2) \in S$.
We aim to show that:

$$(s^1, s^2) \models \psi_1' \wedge \psi_1'' \iff s^1 \models_1 f_t(\psi_1') \wedge f_t(\psi_1'').$$

for any state $(s^1, s^2) \in S$.
($\Rightarrow$) if $(s^1, s^2) \models \psi_1' \wedge \psi_1''$ then clearly $(s^1, s^2) \models \psi_1'$ and $(s^1, s^2) \models \psi_1''$. Hence, from the *inductive hypothesis* (i.e. 4.3.8) also $s^1 \models_1 f_t(\psi_1')$ and $s^1 \models_1 f_t(\psi_1'')$, which is to say $s^1 \models_1 f_t(\psi_1') \wedge f_t(\psi_1'')$.
($\Leftarrow$) if $s^1 \models_1 f_t(\psi_1') \wedge f_t(\psi_1'')$ then clearly $s^1 \models_1 f_t(\psi_1')$ and $s^1 \models_1 f_t(\psi_1'')$. Thus, from from the *inductive hypothesis* (i.e. 4.3.8), we also have that, $\forall s^2 \in S_2 : (s^1, s^2) \in S$, $(s^1, s^2) \models \psi_1'$ and $(s^1, s^2) \models \psi_1''$ which means that $(s^1, s^2) \models \psi_1' \wedge \psi_1''$.

3. $\phi_1 \equiv \xi_1 \equiv \mathcal{S}_{\trianglelefteq p}(\psi_1)$.
Let us assume (4.3.7) as *inductive hypothesis*. From Definition 4.3.2, we have that $f_t(\mathcal{S}_{\trianglelefteq p}(\psi_1)) = \mathcal{S}_{\trianglelefteq g(p,\psi_1,\mathcal{M}_1)} f_t(\psi_1)$, hence we aim to prove that[3]:

[3] Here $\models \mathcal{S}_{\trianglelefteq p}(\psi_1)$ and $\models_1 \mathcal{S}_{g(p,\psi_1,\mathcal{M}_1)}(\hat{f}_t(\psi_1))$ are used instead of $(s^1, s^2) \models \mathcal{S}_{\trianglelefteq p}(\psi_1)$ and $s^1 \models_1 \mathcal{S}_{g(p,\psi_1,\mathcal{M}_1)}(\hat{f}_t(\psi_1))$ since models in a Boucherie framework are ergodic CTMCs, hence, as pointed out in section 3.5, *steady-state* formulae are actually model dependent rather than state dependent.

$$\models \mathcal{S}_{\trianglelefteq p}(\psi_1) \iff \models_1 \mathcal{S}_{g(p,\psi_1,\mathcal{M}_1)}(\hat{f}_t(\psi_1)).$$

($\Rightarrow$) if $\models \mathcal{S}_{\trianglelefteq p}(\psi_1)$ then

$$\Big[\sum_{(t^1,t^2)\in Sat(\psi_1)} \pi(t^1,t^2) \Big] \trianglelefteq p$$

which, since $\mathcal{M}$ has a product form solution, we can rewrite as:

$$G \Big[\sum_{(t^1,t^2)\in Sat(\psi_1)} \pi_1(t^1)\cdot\pi_2(t^2) \Big] \trianglelefteq p$$

Since we are assuming (4.3.7) as *inductive hypothesis*, then from Lemma 4.3.1 also $Sat(\psi_1) = (Sat_1(f_t(\psi_1)) \times S_2) \cap S$, which, from Remark 4.2.3, we can rewrite as, $Sat(\psi_1) = (Sat_{1,\overline{R}}(f_t(\psi_1)) \times S_2) \cup (Sat_{1,R}(f_t(\psi_1)) \times S_{2,\overline{R}})$. As a consequence the sum in the above inequality, can be split as follows:

$$G \big[Sum_a + Sum_b \big] \trianglelefteq p \tag{4.3.9}$$

where:

$$\begin{aligned} Sum_a &= \sum_{t^1\in Sat_{1,\overline{R}}(f_t(\psi_1)} \pi_1(t^1)\cdot \sum_{t^2\in S_2} \pi_2(t^2) \\ Sum_b &= \sum_{t^1\in Sat_{1,R}(f_t(\psi_1)} \pi_1(t^1)\cdot \sum_{t^2\in Sat_{2,\overline{R}}} \pi_2(t^2) \end{aligned}$$

Three different cases, depending on the set $Sat_1(f_t(\psi_1))$, need to be considered:

- $Sat_1(f_t(\psi_1)) \subseteq S_{1,\overline{R}}$:
  in such a case $g(p,\phi_1',\mathcal{M}_1) = \frac{p}{G}$, hence we aim to show that

  $$\Big[G \big[Sum_a + Sum_b \big] \trianglelefteq p\Big] \Longrightarrow \Big[\big[\sum_{t^1\in Sat_1(f_t(\psi_1))} \pi_1(t^1)\big] \trianglelefteq \frac{p}{G}\Big]$$

  Note that

  $$Sat_1(f_t(\psi_1)) \subseteq S_{1,\overline{R}} \Longrightarrow Sat_{1,R}(f_t(\psi_1)) = \emptyset$$

  hence the sum $Sum_b = 0$ in (4.3.9). Thus (4.3.9) results in:

  $$G\big[\sum_{t^1\in Sat_1(f_t(\psi_1))} \pi_1(t^1) \sum_{t^2\in S_2} \pi_2(t^2)\big] \trianglelefteq p$$

which, since π_2 is a distribution over S_2, results in:

$$G\big[\sum_{t^1 \in Sat_1(f_t(\psi_1))} \pi_1(t^1)\big] \trianglelefteq p$$

proving that:

$$s^1 \models_1 \mathcal{S}_{\trianglelefteq \frac{p}{G}}\big(f_t(\psi_1)\big).$$

($\Leftarrow$) By reversing the order of passages in ($\Rightarrow$).

- $Sat_1(f_t(\psi_1)) \subseteq S_{1,R}$.
  In this case $g(p, \psi_1, \mathcal{M}_1) = \frac{p}{G \cdot C_2}$ where $C_2 = \sum_{t^2 \in S_{2,\overline{R}}} \pi_2(t^2)$, obtained from 4.3.5 with $j = 2$, is the probability of not holding R for component $\mathcal{M}_2$ in the long-run. Hence we aim to show that

$$\Big[G\;[\;Sum_a + Sum_b\;] \trianglelefteq p\Big] \Longrightarrow \Big[\big[\sum_{t^1 \in Sat_1(f_t(\psi_1))} \pi_1(t^1)\big] \trianglelefteq \frac{p}{G \cdot C_2}\Big]$$

  Note that

$$Sat_1(f_t(\psi_1)) \subseteq S_{1,R} \Longrightarrow S_{1,\overline{R}} = \emptyset$$

  hence the sum $Sum_a = 0$ in (4.3.9). Thus (4.3.9) results in:

$$G\big[\sum_{t^1 \in Sat_1(f_t(\psi_1))} \pi_1(t^1) \sum_{t^2 \in S_{2,\overline{R}}} \pi_2(t^2)\big] \trianglelefteq p$$

  which, proves that

$$s^1 \models_1 \mathcal{S}_{\trianglelefteq \frac{p}{G \cdot C_2}}\big(f_t(\psi_1)\big).$$

  ($\Leftarrow$) By reversing the order of passages in ($\Rightarrow$).

- $(Sat_{1,\overline{R}}(f_t(\psi_1)) \neq \emptyset) \wedge (Sat_{1,R}(f_t(\psi_1)) \neq \emptyset)$:
  In this case $g(p, \psi_1, \mathcal{M}_1) = \frac{p - G \cdot C_2}{G \cdot C_{12}}$ where C_2 is as in the previous case while

$$C_{12} = \sum_{s^1 \in Sat_{1,\overline{R}}(\psi_1)} \pi_1(s^1) \sum_{s^2 \in S_{2,R}} \pi_2(s^2) = \big[\sum_{s^1 \in Sat_{1,\overline{R}}(\psi_1)} \pi_1(s^1)\big] \cdot (1 - C_2)$$

is the probability for $\mathcal{M}_1$ to satisfy ψ_1 while holding R on the long-run, weighted by the long-run probability for $\mathcal{M}_2$ to hold R (both C_2 and C_{12} are given by 4.3.5 with $k = 1, j = 2$). Hence we aim to show that

$$\Big[G\,\big[\,Sum_a + Sum_b\,\big] \trianglelefteq p\Big] \Longrightarrow \Big[\big[\sum_{t^1 \in Sat_1(f_t(\psi_1))} \pi_1(t^1)\big] \trianglelefteq \frac{p - G\cdot C_2}{G\cdot C_{12}}\Big]$$

Since $((Sat_{1,\overline{R}}(f_t(\phi_1)) \neq \emptyset) \wedge (Sat_{1,R}(f_t(\phi_1)) \neq \emptyset))$, then both Sum_a and Sum_b in (4.3.9) are greater than zero. By factoring out the common states in S_2, and noting that $f_t(\psi_1) = \psi_1$, we obtain:

$$G\big[\sum_{t^1 \in Sat_1(f_t(\psi_1))} \pi_1(t^1) \sum_{t^2 \in S_{2,\overline{R}}} \pi_2(t^2) + \sum_{t^1 \in Sat_{1,\overline{R}}(f_t(\psi_1))} \pi_1(t^1) \sum_{t^2 \in S_{2,R}} \pi_2(t^2)\big] \trianglelefteq p$$

which results in:

$$\big[\sum_{t^1 \in Sat_1(f_t(\psi_1))} \pi_1(t^1)\big] \trianglelefteq \frac{p - G\cdot C_2}{G\cdot C_{12}}$$

hence proving that

$$s^1 \models_1 \mathcal{S}_{\trianglelefteq \frac{p-G\cdot C_2}{G\cdot C_{12}}}\big(f_t(\psi_1)\big).$$

($\Leftarrow$) By reversing the order of passages in ($\Rightarrow$).

The proof for the remaining cases (i.e. $\phi_1 \equiv \phi_1' \wedge \phi_1''$, $\phi_1 \equiv \neg\phi_1'$) is similar to the previous cases, hence, for brevity, we skip it.

□

The above Theorem proves the correctness of the compositional semantics for the *single-component* non-probabilistic state-formulae (i.e. formulae characterised by the syntax 4.3.2), as it is described by the *transformation function* $f_t()$. Hence checking a non-probabilistic state-formula which refers to a single component only, either $\mathcal{M}_1$ or $\mathcal{M}_2$, against $\mathcal{M}$ is equivalent to checking a derived non-probabilistic state-formula against the component it refers to.

Next, formulae involving both components of the Boucherie process are considered and a compositional semantics is derived for them.

4.3.2 Compositional semantics for *general* formulae

CSL *general* formulae are generated by coupling *single component* formulae relating to different components (ϕ_1 and ϕ_2), by means of binary connectives. Their formal characterisation, with respect to the original CSL syntax, has been shown in Definition 4.2.3. Here, though, the same restrictions imposed for *single-component* formulae are considered: probabilistic-path formulae ($\mathcal{P}_{\trianglelefteq p}(\varphi)$) are ruled out and nesting of the steady-state connective ($\mathcal{S}_{\triangleleft p}$) is not permitted.

The resulting syntax for *general* non-probabilistic state-formulae (i.e. which is directly derived from the syntax for generic non-probabilistic state-formulae described in 4.3.1) is as follows:

$$\begin{aligned}
\phi_{12} &::= \phi_1 \wedge \phi_2 \mid \phi_2 \wedge \phi_1 \mid \phi_k \wedge \phi_{12} \mid \phi_{12} \wedge \phi_k \mid \phi_{12} \wedge \phi_{12} \mid \xi_{12} \mid \neg \phi_{12} \\
\psi_{12} &::= \psi_1 \wedge \psi_2 \mid \psi_2 \wedge \psi_1 \mid \psi_k \wedge \psi_{12} \mid \psi_{12} \wedge \psi_k \mid \psi_{12} \wedge \psi_{12} \mid \neg \psi_{12} \\
\xi_{12} &::= \mathcal{S}_{\trianglelefteq p}(\psi_{12})
\end{aligned} \tag{4.3.10}$$

where ϕ_k and ψ_k are as in 4.3.2, while ξ_{12} are steady-state formulae whose argument is a *general* formula (i.e. it refers to both components $\mathcal{M}_1$ and $\mathcal{M}_2$). The use of a specific production for ξ_{12} formulae in 4.3.10, prevents the possibility of nesting $\mathcal{S}_{\trianglelefteq p}$.
The basic idea on which the compositional semantics for *general* formulae relies, is that given a formula ϕ_{12} whose validity is to be checked against a state (s^1, s^2), there exists a *Boolean combination* of satisfiability conditions concerning some derived *single-component* formulae, which turns out to be equivalent to $(s^1, s^2) \models \phi_{12}$.

To explain the intuition on which this idea is based, let us consider an example, referring to the Boucherie process introduced as our *running example* in Section 2.5.1. Suppose we are interested in checking whether the *general* formula $(idle_1 \wedge idle_2)$ is satisfied with respect to the state (s_{10}, s_{20}) of the Boucherie process pictured in Figure 2.8. From the CSL semantics we know that, trivially,

$$(s_{10}, s_{20}) \models (idle_1 \wedge idle_2) \Longleftrightarrow (s_{10}, s_{20}) \models idle_1 \quad \textbf{and} \quad (s_{10}, s_{20}) \models idle_2$$

but then, from Theorem 4.3.1 (i.e compositional semantics for *single-component* formulae), it follows that

$$(s_{10}, s_{20}) \models (idle_1 \wedge idle_2) \Longleftrightarrow s_{10} \models_1 idle_1 \quad \textbf{and} \quad s_{20} \models_2 idle_2$$

The above, trivial, example shows that a compositional semantics for *general* formulae is possible; in fact, checking : the validity of $(idle_1 \wedge idle_2)$ against the state (s_{10}, s_{20}) of the Boucherie process is equivalent to check that $idle_1$ and $idle_2$ are valid with respect to the states s^1 and s^2 of the components' processes.

The formal characterisation of the "decomposed equivalent satisfiability conditions" for a formula ϕ_{12} and a state (s^1, s^2) is by means of the function $cond()$, introduced in Definition 4.3.4. The main point there regards the case of steady-state *general* formulae (i.e. last case of Definition 4.3.4). In order to define the equivalence for $\phi_{12} \equiv \xi_{12} \equiv \mathcal{S}_{\trianglelefteq p}(\psi_{12})$, a decomposed characterisation of $Sat(\psi_{12})$ is needed (i.e. $Sat(\psi_{12})$ must be partitioned in a number of parts each of which is given by the Cartesian product of subsets of the two components). This is achieved through the function $DecSat() : \Psi_{12} \rightarrow 2^{\Psi_1 \times \Psi_2}$, which takes a ψ_{12} formula as argument and returns a set of pair of *single-component* formulae $(\phi_1, \phi_2) \in (\Psi_1 \times \Psi_2)$ characterising a partition of $Sat(\psi_{12})$ (see Lemmma 4.3.2).

Definition 4.3.3 (function $DecSat() : \Psi_{12} \rightarrow 2^{\Psi_1 \times \Psi_2}$**)** *Let ψ_{12} be a Boolean proposition as described in (4.3.10). The value $DecSat(\psi_{12})$ is defined as follows:*

$$DecSat \quad (\psi_{12}) = \begin{cases} \{(\psi_1, \psi_2)\} & \textbf{if } [\psi_{12} \equiv \psi_1 \wedge \psi_2] \vee \\ & \quad [\psi_{12} \equiv \psi_2 \wedge \psi_1] \\ \psi_k \;\; AND_k \;\; DecSat(\psi'_{12}) & \textbf{if } [\psi_{12} \equiv \psi_k \wedge \psi'_{12}] \vee \\ & \quad [\psi_{12} \equiv \psi'_{12} \wedge \psi_k] \\ DecSat(\psi'_{12}) \;\; AND \;\; DecSat(\psi''_{12}) & \textbf{if } [\psi_{12} \equiv \psi'_{12} \wedge \psi''_{12}] \\ \overline{\bigwedge}_{(\alpha_1, \alpha_2) \in DecSat(\psi'_{12})} \{(\neg\alpha_1, tt), (\alpha_1, \neg\alpha_2)\} & \textbf{if } \psi_{12} \equiv \neg\psi'_{12} \end{cases} \tag{4.3.11}$$

where AND_1, AND_2 and AND are binary operators returning, respectively, the conjunction of a single-component formula with a set of pairs of single-component formulae and the pairwise conjunction of two sets of pairs of single-component formulae. Formally

$$
\begin{aligned}
(\gamma_1\ AND_1\ \Gamma) &= \bigcup_{(\alpha_1,\alpha_2)\in\Gamma} \{(\gamma_1 \wedge \alpha_1, \alpha_2)\} \\
(\gamma_2\ AND_2\ \Gamma) &= \bigcup_{(\alpha_1,\alpha_2)\in\Gamma} \{(\alpha_1, \alpha_2 \wedge \gamma_2)\} \\
(\Gamma' AND\ \Gamma'') &= \bigcup_{(\alpha_1',\alpha_2')\in\Gamma'} \{ \bigcup_{(\alpha_1''\alpha_2'')\in\Gamma''} \{(\alpha_1' \wedge \alpha_1'', \alpha_2' \wedge \alpha_2'')\}\}
\end{aligned}
$$

and where $\overline{\bigwedge}$ in (4.3.11) refers to the conjunction AND of sets of pairs of single-component formulae.

The next Lemma proves that the set of pairs of *single-component* formulae provided by $DecSat(\psi_{12})$ actually represents a characterisation of a partition of $Sat(\psi_{12})$. In order to prove such a result a preliminary property concerning the complement of the Cartesian product of two subsets, needs to be introduced.

Proposition 4.3.1 (Complement of a Cartesian product) *Let $A' \subset A$ and $B' \subset B$, be subsets, respectively, of a set A and a set B. The complement of the Cartesian product $A' \times B'$ is given by:*

$$\overline{A' \times B'} = (\overline{A'} \times B) \cup (A' \times \overline{B'})$$

Proof. We need to show that the following bi-implication holds:

$$(a,b) \in \overline{A' \times B'} \Longleftrightarrow (a,b) \in [(\overline{A'} \times B) \cup (A' \times \overline{B'})]$$

($\Rightarrow$) If $(a,b) \in \overline{A' \times B'}$ then $(a \in \overline{A'} \vee b \in \overline{B'})$. But $(a \in \overline{A'} \vee b \in \overline{B'}) \Longrightarrow (a,b) \in (\overline{A'} \times B)$ which proves ($\Rightarrow$).

($\Leftarrow$) If $(a,b) \in [(\overline{A'} \times B) \cup (A' \times \overline{B'})]$ two cases need to be considered. If $(a,b) \in (\overline{A'} \times B)$ then $(a \in \overline{A'} \wedge b \in B)$ but this implies also $(a \in \overline{A'} \vee b \in \overline{B'})$ which proves $(a,b) \in \overline{A' \times B'}$. On the other hand if $(a,b) \in (A' \times \overline{B'})$ then $(a \in A' \wedge b \in \overline{B'})$ but this implies also $(a \in \overline{A'} \vee b \in \overline{B'})$ which proves $(a,b) \in \overline{A' \times B'}$.

□

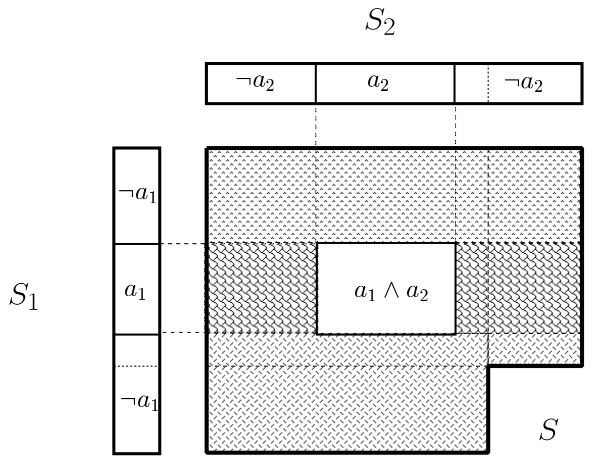


Figure 4.4: Decomposition of $\overline{Sat(a_1 \wedge a_2)} = (Sat_1(\neg a_1) \times S_2) \cup (Sat(a_1) \times Sat(\neg a_2))$.

Figure 4.4 shows an example of application of Proposition 4.3.1 to a bidimensional Boucherie process: the complement of $Sat(a_1 \wedge a_2) = Sat_1(a_1) \times Sat_2(a_2)$ is partitioned in two subsets. The first one being $(\overline{Sat(a_1)} \times S_2) = (Sat_1(\neg a_1) \times S_2)$; the second one being $(Sat(a_1) \times \overline{Sat_2(a_2)}) = (Sat_1(a_1) \times Sat_2(\neg a_2))$.

The result of Proposition 4.3.1 is needed for proving the following Lemma, which shows that $DecSat(\psi_{12})$ actually characterises a partition of $Sat(\psi_{12})$.

Lemma 4.3.2 *Let $\mathcal{M}$ be a bidimensional Boucherie process and ψ_{12} a formula as in the syntax described by (4.3.10), then the following holds:*

$$Sat(\psi_{12}) = \bigcup_{(\alpha_1,\alpha_2) \in DecSat(\psi_{12})} \left[Sat_1(\alpha_1) \times Sat_2(\alpha_2)\right] \setminus (R_1 R_2) \tag{4.3.12}$$

Proof. By structural induction over the definition of $DecSat()$. For simplicity we denote the right hand side of the equality (4.3.12) as $S^*(\psi_{12})$.
Proving the equality (4.3.12) means showing that

$$(s^1, t^2) \in Sat(\psi_{12}) \Longleftrightarrow (s^1, t^2) \in S^*(\psi_{12})$$

base case:[4] $\psi_{12} \equiv \psi_1 \wedge \psi_2$ or $\psi_{12} \equiv \psi_2 \wedge \psi_1$.
From Definition 4.3.3, we know that, in this case, $DecSat(\psi_{12}) = \{(\psi_1, \psi_2)\}$, hence $S^*(\psi_1 \wedge \psi_2) = [Sat_1(\psi_1) \times Sat_2(\psi_2)] \setminus (R_1 R_2)$. Thus we aim to prove that

$$(s^1, s^2) \in Sat(\psi_1 \wedge \psi_2) \Longleftrightarrow [Sat_1(\psi_1) \times Sat_2(\psi_2)] \setminus (R_1 R_2)$$

($\Rightarrow$) if $(s^1, s^2) \in Sat(\psi_1 \wedge \psi_2)$ then $(s^1, s^2) \models \psi_1$ and $(s^1, s^2) \models \psi_2$ and clearly also $(s^1, s^2) \in S$. But then from Theorem 4.3.1, we also have that, $[s^1 \models_1 \psi_1 \wedge s^2 \models_2 \psi_2]$ which proves, $(s^1, s^2) \in S^*(\psi_1 \wedge \psi_2)$.

($\Leftarrow$) by reversing ($\Rightarrow$).

induction step: all the remaing cases in the definition of $DecSat()$ needs to be considered.

1. $\psi_{12} \equiv \psi_k \wedge \psi'_{12}$ or $\psi_{12} \equiv \psi'_{12} \wedge \psi_k$
Again here, for brevity, we consider only the case $\psi_{12} \equiv \psi_k \wedge \psi'_{12}$, knowing that the same result holds also for $\psi_{12} \equiv \psi'_{12} \wedge \psi_k$ as a direct consequence of the simmetricity of the conjunction.
If $\psi_{12} \equiv \psi_1 \wedge \psi'_{12}$ then $DecSat(\psi_{12}) = [\psi_1 \; AND_1 \; DecSat(\psi'_{12})]$, hence we aim to show that

$$(s^1, s^2) \in Sat(\psi_1 \wedge \psi'_{12}) \Longleftrightarrow (s^1, s^2) \in \bigcup_{(\alpha_1, \alpha_2) \in [\psi_1 \; AND_1 \; DecSat(\psi'_{12})]} \big[Sat_1(\alpha_1) \times Sat_2(\alpha_2)\big] \setminus (R_1 R_2).$$

As *inductive hypothesis* let us assume

$$Sat(\psi'_{12}) = S^*(\psi'_{12})$$

[4] For brevity we consider $\psi_{12} \equiv \psi_1 \wedge \psi_2$ as our base-case, knowing that same result holds when $\psi_{12} \equiv \psi_2 \wedge \psi_1$ is considered.

($\Rightarrow$) if $(s^1,s^2) \in Sat(\psi_1 \wedge \psi'_{12})$ then it is true that $(s^1,s^2) \in Sat(\psi_1)$ and $(s^1,s^2) \in Sat(\psi'_{12})$ and clearly also $(s^1,s^2) \in S$. Though, if $(s^1,s^2) \in Sat(\psi'_{12})$ then (*inductive hypothesis*) there exists a pair $(\alpha'_1,\alpha'_2) \in DecSat(\psi'_{12})$ such that $(s^1,s^2) \in [Sat_1(\alpha'_1) \times Sat_2(\alpha'_2)] \setminus (R_1R_2)$. which means $(s^1,s^2) \models \alpha'_1$ and $(s^1,s^2) \models \alpha'_2$. From Theorem 4.3.1 then also $s^1 \models_1 \alpha'_1$ and $s^2 \models_2 \alpha'_2$ and also $s^1 \models_1 \psi_1$ which proves $(s^1,s^2) \in [Sat_1(\psi_1 \wedge \alpha'_1) \times Sat_2(\alpha'_2)] \setminus (R_1R_2)$ hence $(s^1,s^2) \in S^*(\psi_1 \wedge \psi'_{12})$.

($\Leftarrow$) by reversing ($\Rightarrow$).

2. $\psi_{12} \equiv \psi'_{12} \wedge \psi''_{12}$.
If $\psi_{12} \equiv \psi'_{12} \wedge \psi''_{12}$ then $DecSat(\psi_{12}) = [DecSat(\psi'_{12})\ AND\ DecSat(\psi''_{12})]$, hence we aim to show that

$$(s^1,s^2) \in Sat(\psi'_{12} \wedge \psi''_{12}) \Longleftrightarrow (s^1,s^2) \in \bigcup_{(\alpha_1,\alpha_2)\in[DecSat(\psi'_{12})\ AND\ DecSat(\psi''_{12})]} \big[Sat_1(\alpha_1) \times Sat_2(\alpha_2)\big] \setminus (R_1R_2).$$

Let assume the following *inductive hypothesis*:

$$Sat(\psi'_{12}) = S^*(\psi'_{12}) \quad \text{and} \quad Sat(\psi''_{12}) = S^*(\psi''_{12})$$

($\Rightarrow$) if $(s^1,s^2) \in Sat(\psi'_{12} \wedge \psi''_{12})$ then

$$(s^1,s^2) \in Sat(\psi'_{12}) \wedge (s^1,s^2) \in Sat(\psi''_{12})$$

Hence also, (*inductive hypothesis*) $(s^1,s^2) \in S^*(\psi'_{12})$ and $(s^1,s^2) \in S^*(\psi''_{12})$, but that means that there exists a pair $(\alpha'_1,\alpha'_2) \in DecSat(\psi'_{12})$ and a pair $(\alpha''_1,\alpha''_2) \in DecSat(\psi''_{12})$ such that $(s^1,s^2) \in [Sat_1(\alpha'_1) \times Sat_2(\alpha'_2)] \setminus (R_1R_2)$ and $(s^1,s^2) \in [Sat_1(\alpha''_1) \times Sat_2(\alpha''_2)] \setminus (R_1R_2)$. Then, as a consequence of Theorem 4.3.1, also $(s^1,s^2) \in [Sat_1(\alpha'_1 \wedge \alpha''_1) \times Sat_2(\alpha' \wedge \alpha''_2)] \setminus (R_1R_2)$. From the definition of the operator *AND* it is straightforward to show that $(\alpha'_1,\alpha'_2) \in DecSat(\psi'_{12})$ and $(\alpha''_1,\alpha''_2) \in DecSat(\psi''_{12})$ implies $(\alpha'_1 \wedge \alpha''_1, \alpha'_2 \wedge \alpha''_2) \in DecSat(\psi'_{12} \wedge \psi''_{12})$ which proves $(s^1,s^2) \in S^*(\psi'_{12} \wedge \psi''_{12})$.

($\Leftarrow$) by reversing ($\Rightarrow$).

3. $\psi_{12} \equiv \neg\psi'_{12}$
If $\psi_{12} \equiv \neg\psi'_{12}$ then

$$DecSat(\psi_{12}) = \overline{\bigwedge}_{(\alpha_1,\alpha_2)\in DecSat(\psi'_{12})}\{(\neg\alpha_1, true), (\alpha_1, \neg\alpha_2)\}$$

where $\overline{\bigwedge}$ refers to the binary operator *AND* as described in Definition 4.3.3. We denote the right hand side of the above equality as $NOT(\psi'_{12})$. Hence we aim to show that

$$(s^1,s^2) \in Sat(\neg\psi'_{12}) \Longleftrightarrow (s^1,s^2) \in \bigcup_{(\alpha_1,\alpha_2)\in NOT(\psi'_{12})} [Sat_1(\alpha_1)\times Sat_2(\alpha_2)] \setminus (R_1R_2).$$

Let assume the following *inductive hypothesis*:

$$Sat(\psi'_{12}) = S^*(\psi'_{12})$$

($\Rightarrow$) if $(s^1,s^2) \in Sat(\neg\psi'_{12})$ then $(s^1,s^2) \notin Sat(\psi'_{12})$ which also means (*inductive hypothesis*) that $(s^1,s^2) \notin [Sat_1(\alpha_1)\times Sat_2(\alpha_2)] \setminus (R_1R_2), \forall(\alpha_1,\alpha_2) \in DecSat(\psi'_{12})$. Which is $(s^1,s^2) \in [\overline{Sat_1(\alpha_1) \times Sat_2(\alpha_2)}] \setminus (R_1R_2)$, hence:

$$(s^1,s^2) \in \bigcap_{(\alpha_1,\alpha_2)\in DecSat(\psi'_{12})} \left[[\overline{Sat_1(\alpha_1) \times Sat_2(\alpha_2)}] \setminus (R_1R_2)\right]$$

From Proposition 4.3.1 we know that

$$\begin{aligned}
[\overline{Sat_1(\alpha_1) \times Sat_2(\alpha_2)}] &= [\overline{Sat_1(\alpha_1)} \times S_2] \cup [Sat_1(\alpha_1) \times \overline{Sat_2(\alpha_2)}] \\
&= [Sat_1(\neg\alpha_1) \times S_2] \cup [Sat_1(\alpha_1) \times Sat_2(\neg\alpha_2)]
\end{aligned}$$

thus

$$(s^1,s^2) \in \bigcap_{(\alpha_1,\alpha_2)\in DecSat(\psi'_{12})} \left[[Sat_1(\neg\alpha_1) \times S_2] \cup [Sat_1(\alpha_1) \times Sat_2(\neg\alpha_2)]\right] \setminus (R_1R_2)$$

Considering the distribution of $\cap$ with respect to both $\cup$ and $\times$ and also considering that, $Sat_1(\phi'_1) \cap Sat_1(\phi''_1) = Sat_1(\phi'_1 \wedge \phi''_1)$ for any two *single-component* formulae ϕ'_1 and ϕ''_1, then it straightforwardly follows that

$$\bigcap_{(\alpha_1,\alpha_2)\in DecSat(\psi'_{12})} \left[[Sat_1(\neg\alpha_1) \times S_2] \cup [Sat_1(\alpha_1) \times Sat_2(\neg\alpha_2)]\right] \setminus R_1R_2 = \bigcup_{(\delta_1,\delta_2)\in NOT(\psi'_{12})} Sat_1(\delta_1) \times Sat_2(\delta_2) \setminus R_1R_2$$

which proves ($\Rightarrow$).

($\Leftarrow$) By reversing ($\Rightarrow$).

$\square$

In the next example some ψ_{12} formulae are considered and the set $DecSat(\psi_{12})$ is computed, illustrating the application of the above Lemma.

Example 4.3.1 (Decomposition of $Sat(\psi_{12})$ by means of $DecSat(\psi_{12})$) *Let us consider the application $DecSat()$ to some ψ_{12} formulae. We focus on ψ_{12} formulae involving four atomic propositions, namely $\{a_1, b_1, a_2, b_2\}$ (i.e. $\mathcal{At}(\psi_{12}) = \{a_1, b_1, a_2, b_2\}$). We observe that there are four possible situations concerning the relationship between $Sat_1(a_1)$ and $Sat_1(b_1)$ on one hand and $Sat_2(a_2)$ and $Sat_2(b_2)$ on the other and these can be characterised in the following way:*

a) $[Sat_1(a_1) \cap Sat_1(b_1) = \emptyset] \vee [Sat_2(a_2) \cap Sat_2(b_2) = \emptyset]$.
This case relates to one out of the three situations depicted in Figure 4.5.

b) $[Sat_1(a_1) \cap Sat_1(b_1) \neq \emptyset] \wedge [Sat_2(a_2) \cap Sat_2(b_2) \neq \emptyset]$.
This case relates to Figure 4.6.

The distinction concerning the possible relationship between the satisfiability sets for the atoms a_1, b_1 and a_2, b_2 is useful to show that the decomposition provided by $DecSat(\psi_{12})$ is correct in any possible situation. Let us focus on the following examples of ψ_{12}.

1. $\psi_{12} \equiv \psi_1 \wedge \psi_2 \equiv (a_1 \wedge b_1) \wedge (a_2 \wedge b_2)$.
In this case we know that, trivially, $Sat((a_1 \wedge b_1) \wedge (a_2 \wedge b_2))$ is given by the part of the Cartesian product $Sat_1((a_1 \wedge b_1)) \times Sat_2(a_2 \wedge b_2)$ which intersects S which is: $Sat((a_1 \wedge b_1) \wedge (a_2 \wedge b_2)) = Sat_1((a_1 \wedge b_1)) \times Sat_2(a_2 \wedge b_2) \setminus (R_1 R_2)$. Also, from Definition 4.3.11, we have that:

$$DecSat((a_1 \wedge b_1) \wedge (a_2 \wedge b_2)) = \{(a_1 \wedge b_1), (a_2 \wedge b_2)\}$$

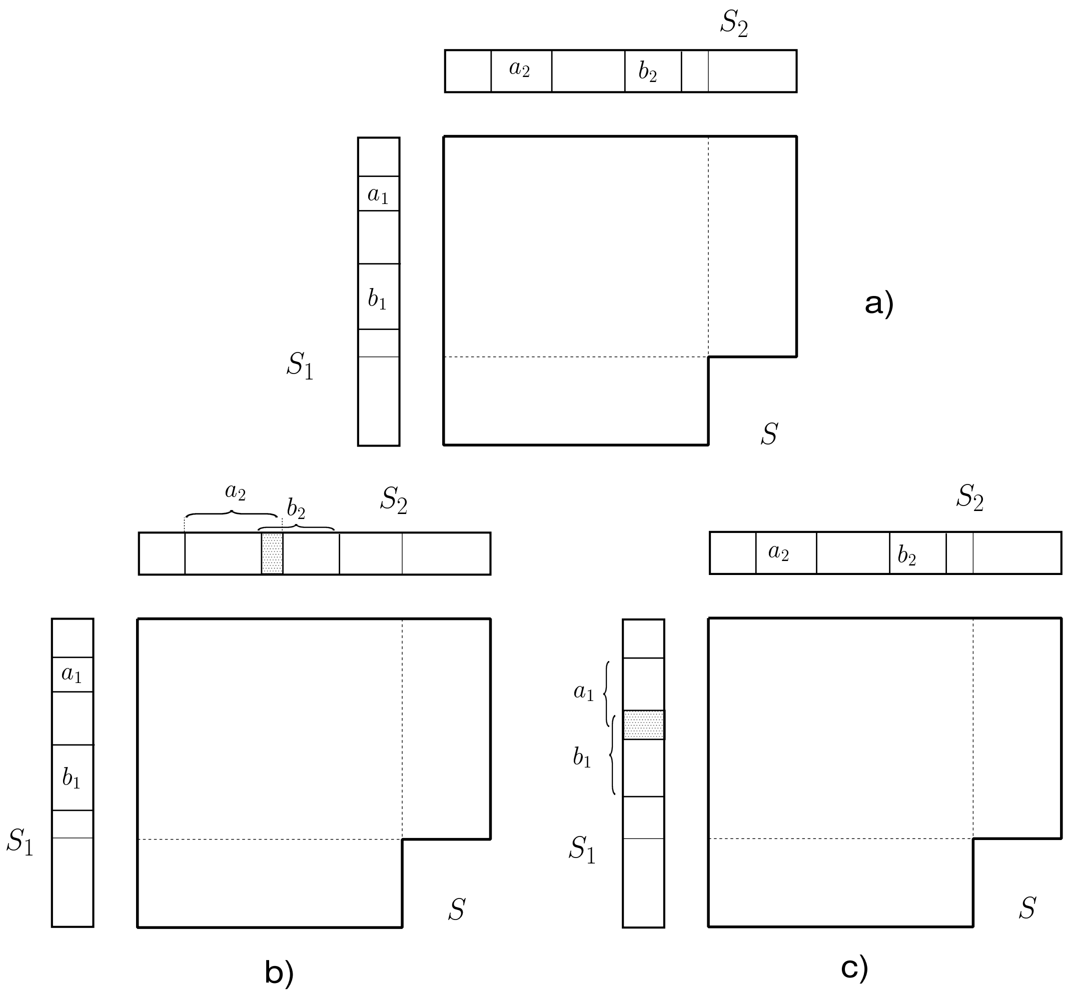


Figure 4.5: $[Sat_1(a_1) \cap Sat_1(b_1) = \emptyset] \vee [Sat_2(a_2) \cap Sat_2(b_2) = \emptyset]$.

which shows the correctness of Lemma 4.3.2 for this case.

Let us consider, one by one, each possibility regarding the satisfiability of the atoms a_1, b_1 and a_2, b_2 and let us show that in any case $Sat((a_1 \wedge b_1) \wedge (a2 \wedge b_2))$ is characterised in terms of the product $[Sat_1(a_1 \wedge b_1) \times Sat_2(a_2 \wedge b_2)] \setminus (R_1 R_2)$. If either $Sat_1(a_1)$ and $Sat_1(b_1)$ or $Sat_2(a_2)$ and $Sat_2(b_2)$ are disjoint (Figure 4.5), then clearly $Sat((a_1 \wedge b_1) \wedge (a_2 \wedge b_2)) = \emptyset$. Though, clearly, also $Sat_1(a_1 \wedge b_1) = \emptyset$ or $Sat_1(a_1 \wedge b_1) = \emptyset$, hence $[Sat_1(a_1 \wedge b_1) \times Sat_2(a_2 \wedge b_2)] = \emptyset$. On the other hand if both $Sat_1(a_1 \wedge b_1)$ and $Sat_2(a_2 \wedge b_2)$ are not empty (Figure 4.6) then the conjunction $(a_1 \wedge b_1) \wedge (a_2 \wedge b_2)$ is not empty and $Sat((a_1 \wedge b_1) \wedge (a_2 \wedge b_2))$ is actually given by coupling all states of $Sat_1(a_1 \wedge$

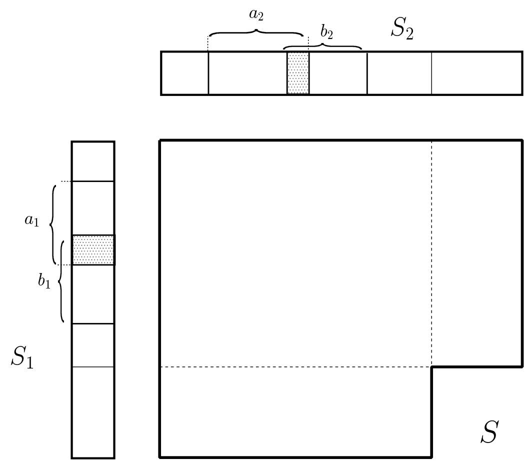


Figure 4.6: $[Sat_1(a_1) \cap Sat_1(b_1) \neq \emptyset] \wedge [Sat_2(a_2) \cap Sat_2(b_2) \neq \emptyset]$.

b_1) with all the states of $Sat_2(a_2 \wedge b_2)$, which proves the result of DecSat$((a_1 \wedge b_1) \wedge (a_2 \wedge b_2))$ being correct also for this case.

In the following, the less trivial case of a negated general formula (DecSat$(\neg\psi_{12})$) is considered. We will focus on two different types of negated general formulae. The first one is given by the negation of a "simple" conjunction, $\neg\psi_{12} \equiv \neg[(a_1 \wedge b_1) \wedge (a_2 \wedge b_2)]$), while the second involves recursion, being the negation of a conjunction whose conjuncts are themselves negated conjunctions, $\neg\psi_{12} \equiv \neg[\neg(a_1 \wedge a_2) \wedge \neg(b_1 \wedge b_2)]$, (we note that, indeed, this is equivalent to the disjunction of two conjunctions $\neg\psi_{12} \equiv (a_1 \wedge a_2) \vee (b_1 \wedge b_2)$). In both cases the set DecSat$(\neg\psi_{12})$ is computed and proved to be correct by considering every possible case concerning the satisfiability of the atoms a_1, b_1 and a_2, b_2 (see Figure 4.5 and Figure 4.6).

2. $\psi_{12} \equiv \neg\psi_{12} \equiv \neg[(a_1 \wedge b_1) \wedge (a_2 \wedge b_2)]$.
From Definition 4.3.3 we know that DecSat$(\neg\psi_{12})$ is given by the pairwise conjunction (AND) between the sets of pairs $\{(\neg\alpha_1, tt), (\alpha_1, \neg\alpha_2)\}$ where (α_1, α_2) are elements of DecSat(ψ_{12}). From the previous case, though, we know that the decomposition of $(a_1 \wedge b_1) \wedge (a_2 \wedge b_2)$ consists of a single pair which is, DecSat$((a_1 \wedge b_1) \wedge (a_2 \wedge b_2)) = \{(a_1 \wedge b_1), (a_2 \wedge b_2)\}$, hence the decomposition of its

negation[5] *is given by:*

$$
\begin{aligned}
DecSat(\neg[(a_1 \wedge b_1) \wedge (a_2 \wedge b_2)]) &= \overline{\bigwedge}_{(\alpha_1,\alpha_2)\in DecSat([(a_1\wedge b_1)\wedge(a_2\wedge b_2)])} \{(\neg\alpha_1, tt), (\alpha_1, \neg\alpha_2)\} \\
&= \{(\neg(a_1 \wedge b_1), tt), ((a_1 \wedge b_1), \neg(a_2 \wedge b_2))\}
\end{aligned}
$$

Let us consider now the different possible situations concerning the satisfiability of the atoms a_1, b_1 *and* a_2, b_2 *and show that in every case the pairs in* $DecSat(\neg[(a_1 \wedge b_1) \wedge (a_2 \wedge b_2)])$ *actually provide a characterisation of* $Sat(\neg[(a_1 \wedge b_1) \wedge (a_2 \wedge b_2)])$.

a) *In this case either* $[Sat_1(a_1) \cap Sat_1(b_1)] = \emptyset$ *or* $[Sat_2(a_2) \cap Sat_2(b_2)] = \emptyset$*, or both. That means that there can be no such a state* $(s^1, s^2) \in S$ *where both the conjuncts* $(a_1 \wedge b_1)$ *and* $(a_2 \wedge b_2)$ *are satisfied, hence* $Sat(\neg[(a_1 \wedge b_1) \wedge (a_2 \wedge b_2)]) = S$*.*

Let us suppose that $Sat_1(a_1)$ *and* $Sat_1(b_1)$ *are disjoint; in this case* $Sat_1(a_1 \wedge b_1)$ *is the empty set (see Figure 4.5.a or Figure 4.5.b), hence* $Sat_1(\neg(a_1 \wedge b_1)) = S_1$ *which means* $Sat_1(\neg(a_1 \wedge b_1)) \times Sat_2(tt) = S$*. This proves that the first pair in* $DecSat(\neg[(a_1 \wedge b_1) \wedge (a_2 \wedge b_2)])$*, namely* $(\neg(a_1 \wedge b_1), tt)$*, characterises a single element partition of* $Sat(\neg[(a_1 \wedge b_1) \wedge (a_2 \wedge b_2)])$*, independently of whether* $Sat_2(a_2)$ *and* $Sat_2(b_2)$ *are disjoint or not.*

On the other hand if $Sat_1(a_1)$ *and* $Sat_1(b_1)$ *are not disjoint while* $Sat_2(a_2)$ *and* $Sat_2(b_2)$ *are, then* $DecSat(\neg[(a_1 \wedge b_1) \wedge (a_2 \wedge b_2)])$*, provides a two element partition of* $Sat(\neg[(a_1 \wedge b_1) \wedge (a_2 \wedge b_2)]) = S$*. Figure 4.6 depicts the form of the two parts, associated, respectively, with the pair* $(\neg(a_1 \wedge b_1), tt)$ *and with the pair* $((a_1 \wedge b_1), \neg(a_2 \wedge b_2))$*.*

b) *In this case both* $(Sat_1(a_1) \cap Sat_1(b_1))$ *and* $(Sat_2(a_2) \cap Sat_2(b_2))$ *are assumed to be not empty. As a result there will exist at least one state in S where* $(a_1 \wedge b_1) \wedge (a_2 \wedge b_2)$ *is true, hence* $Sat(\neg[(a_1 \wedge b_1) \wedge (a_2 \wedge b_2)]$ *has to be a proper subset of S. In such a situation the two pairs in* $DecSat(\neg[(a_1 \wedge b_1) \wedge (a_2 \wedge b_2)])$ *split the complement of* $Sat((a_1 \wedge b_1) \wedge (a_2 \wedge b_2))$ *which indeed is equal to* $Sat(\neg[(a_1 \wedge$

[5] It should be noted that, the number of pairs the decomposition of the negation of a ψ_{12} formula consists of, is given by the n-th power of 2, where n is the number of pairs the decomposition of ψ_{12} is made of: $|DecSat(\neg\psi_{12})| = 2^{|DecSat(\psi_{12})|}$

$b_1) \wedge (a_2 \wedge b_2)])$, *in two parts.*

3. $\psi_{12} \equiv (a_1 \wedge a_2) \vee (b_1 \wedge b_2) \equiv \neg[\neg(a_1 \wedge a_2) \wedge \neg(b_1 \wedge b_2)]$.
Here we consider the negation of the conjunction of two negated general formulae. In order to determine $DecSat(\neg[\neg(a_1 \wedge a_2) \wedge \neg(b_1 \wedge b_2)])$ *we proceed incrementally, starting from determining the decomposition of the conjuncts:*

$$\begin{aligned} DecSat(\neg(a_1 \wedge a_2)) &= \{(\neg a_1, tt), (a_1, \neg a_2)\} \\ DecSat(\neg(b_1 \wedge b_2)) &= \{(\neg b_1, tt), (b_1, \neg b_2)\} \end{aligned}$$

The decomposition of the conjunction $[\neg(a_1 \wedge a_2) \wedge \neg(b_1 \wedge b_2)]$ *is then given by the pairwise conjunction of the conjuncts' decomposition:*

$$\begin{aligned} DecSat([\neg(a_1 \wedge a_2) \wedge \neg(b_1 \wedge b_2)]) &= [DecSat(\neg(a_1 \wedge a_2))]\ AND\ [DecSat(\neg(b_1 \wedge b_2))] \\ &= \{(\neg a_1, tt), (a_1, \neg a_2)\}\ AND\ \{(\neg b_1, tt), (b_1, \neg b_2)\} \\ &= \{(\neg a_1 \wedge \neg b_1, tt), (\neg a_1 \wedge b_1, \neg b_2), \\ &\quad\ (a_1 \wedge \neg b_1, \neg a_2), (a_1 \wedge b_1, \neg a_2 \wedge \neg b_2)\} \end{aligned}$$

Finally $DecSat(\neg[\neg(a_1 \wedge a_2) \wedge \neg(b_1 \wedge b_2)])$ *can be computed from the terms of* $DecSat([\neg(a_1 \wedge a_2) \wedge \neg(b_1 \wedge b_2)])$. *This leads to a set of sixteen pairs, which can straightforwardly be proved equivalent*[6] *to:*

$$\begin{aligned} DecSat(\neg[\neg(a_1 \wedge a_2) \wedge \neg(b_1 \wedge b_2)]) &= \{((a_1 \wedge b_1), (a_2 \vee b_2)), \\ &\quad\ (a_1 \wedge \neg b_1, a_2), \\ &\quad\ (\neg a_1 \wedge b_1, b_2)\} \end{aligned}$$

which suggests a partition of $Sat((a_1 \wedge a_2) \vee (b_1 \wedge b_2))$ *consisting of at most three parts.*

[6] That equivalence relies on the fact that some pair of formulae (α_1, α_2), lead to the empty set (i.e. they are such that $Sat_1(\alpha_1) = \emptyset$ or $Sat_2(\alpha_2) = \emptyset$). For example, when either α_1 or α_2 contains a contradiction (e.g. $(a_1 \wedge \neg a_1, b_2)$) then (α_1, α_2) can be ruled out as clearly $Sat_1(\alpha_1) \times Sat_2(\alpha_2) = \emptyset$. Similarly a pair like $((a_1 \vee b_1) \wedge \neg(\neg a_1 \wedge b_1) \wedge \neg(a_1 \wedge \neg b_1) \wedge \neg(a_1 \wedge b_1), tt)$ which is actually one of the sixteen elements of $DecSat([(a_1 \wedge a_2) \vee \neg(b_1 \wedge b_2)])$, can be easily proved to lead to the empty set, as $Sat_1((a_1 \vee b_1) \wedge \neg(\neg a_1 \wedge b_1) \wedge \neg(a_1 \wedge \neg b_1) \wedge \neg(a_1 \wedge b_1)) = \emptyset$ independently of the relationship between $Sat_1(a_1)$ and $Sat_1(b_1)$.

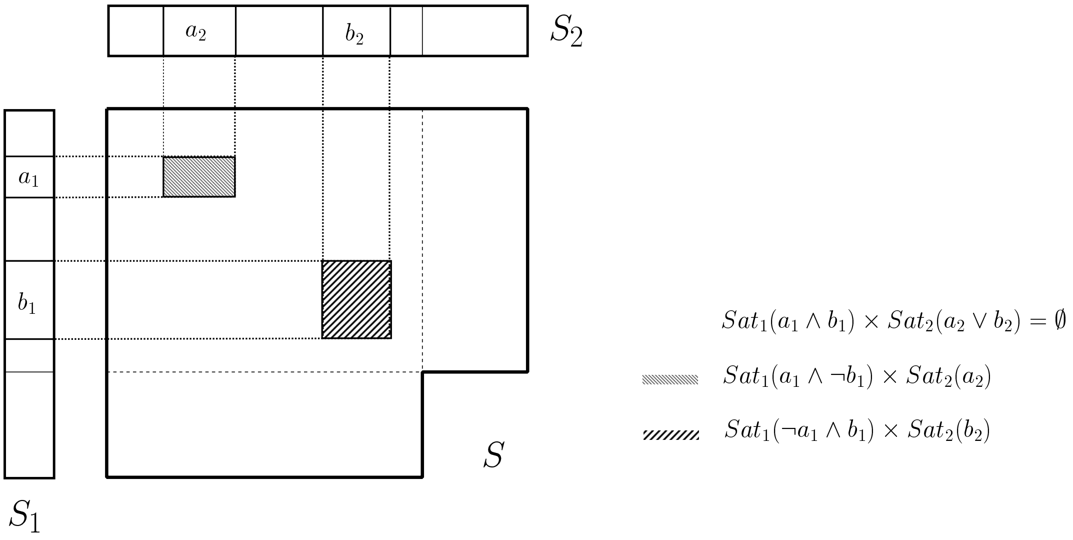


Figure 4.7: $[Sat_1(a_1) \cap Sat_1(b_1) = \emptyset] \wedge [Sat_2(a_2) \cap Sat_2(b_2) = \emptyset]$.

Let us consider some of the possible situations concerning the sets $Sat_1(a_1)$, $Sat_1(b_1)$ and $Sat_2(a_2)$, $Sat_2(b_2)$. In Figure 4.7 both $Sat_1(a_1)$, $Sat_1(b_1)$ and $Sat_2(a_2)$, $Sat_2(b_2)$ are assumed to be disjoint. In this case the three pairs in $DecSat((a_1 \wedge a_2) \vee (b_1 \wedge b_2))$ actually result in a bi-partition of $Sat((a_1 \wedge a_2) \vee (b_1 \wedge b_2))$ as clearly $Sat_1(a_1 \wedge b_1)$ contains no elements (hence the pair $((a_1 \wedge b_1), (a_2 \wedge b_2)) \in DecSat(\neg[\neg(a_1 \wedge a_2) \wedge \neg(b_1 \wedge b_2)])$ leads to the empty set).

A three element partition of $Sat((a_1 \wedge a_2) \vee (b_1 \wedge b_2))$ results, instead, both when the intersection between $Sat_1(a_1)$ and $Sat_1(b_1)$ is not empty while the one between $Sat_2(a_2)$ and $Sat_2(b_2)$ is (Figure 4.8), and also when neither $Sat_1(a_1)$ and $Sat_1(b_1)$ nor $Sat_2(a_2)$ and $Sat_2(b_2)$ are disjoint (Figure 4.9). In those cases all the three pairs in $DecSat((a_1 \wedge a_2) \vee (b_1 \wedge b_2))$ correspond to a non-empty subset of S.

Finally a two element partition occurs whenever the set of states satisfying an atom (either a_k or b_k) is a proper subset of the satisfiability set of the other. Figure 4.10 points out the two parts $Sat((a_1 \wedge a_2) \vee (b_1 \wedge b_2))$ consists of when $Sat_1(b_1) \subset Sat_1(a_1)$ and $Sat_2(b_2) \subset Sat_2(a_2)$, while Figure 4.11 shows a similar result for the case $Sat_1(b_1) \subset Sat_1(a_1)$ and $Sat_2(a_2) \subset Sat_2(b_2)$.

$\square$

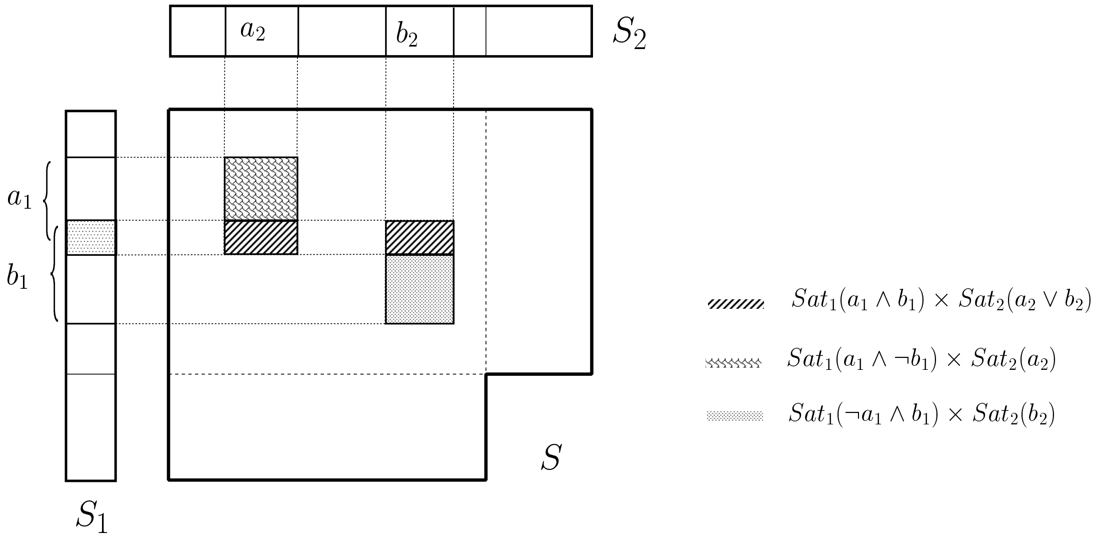


Figure 4.8: $[Sat_1(a_1) \cap Sat_1(b_1) \neq \emptyset] \wedge [Sat_2(a_2) \cap Sat_2(b_2) = \emptyset]$.

Having demonstrated that the decomposition of $Sat(\psi_{12})$ is given by $DecSat(\psi_{12})$, the function $cond()$ can be formalised in the next definition.

Definition 4.3.4 (function $cond()\colon S \times \Phi_{12} \rightarrow \mathcal{B}(Sat)$.**)** *Let (s^1, s^2) be a state of a bidimensional Boucherie process $\mathcal{M}$ and ϕ_{12} a formula as in (4.3.10), the value $cond((s^1, s^2), \phi_{12})$ is a boolean combination of single-component satisfiability condi-*

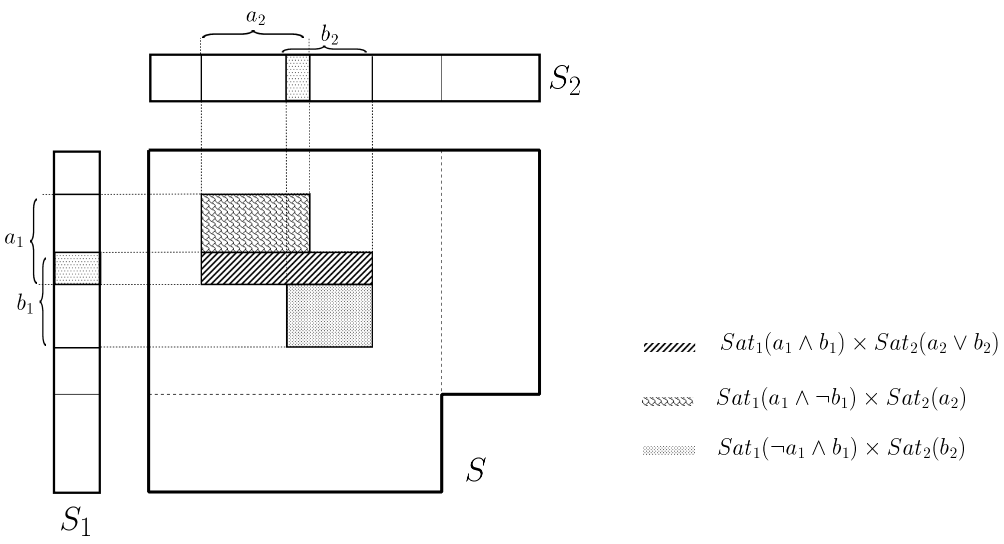


Figure 4.9: $[Sat_1(a_1) \cap Sat_1(b_1) \neq \emptyset] \wedge [Sat_2(a_2) \cap Sat_2(b_2) \neq \emptyset]$.

tions, defined as:

$$cond \quad \big((s^1, s^2), \phi_{12}\big) =$$

$$\begin{cases} s^1 \models_1 f_t(\phi_1) \;\; \textbf{and} \;\; s^2 \models_2 f_t(\phi_2) & \textbf{if} \;\; [\phi_{12} \equiv \phi_1 \wedge \phi_2] \vee \\ & \quad [\phi_{12} \equiv \phi_2 \wedge \phi_1] \\ s^k \models_k f_t(\psi_k) \;\; \textbf{and} \;\; cond\big((s^1, s^2), \phi'_{12}\big) & \textbf{if} \;\; [\phi_{12} \equiv \phi_k \wedge \phi'_{12}] \vee \\ & \quad [\phi_{12} \equiv \phi'_{12} \wedge \phi_k] \\ cond((s^1, s^2), \phi'_{12}) \;\; \textbf{and} \;\; cond\big((s^1, s^2), \phi''_{12}\big) & \textbf{if} \;\; [\phi_{12} \equiv \phi'_{12} \wedge \phi''_{12}] \\ \textbf{not} \;\; cond\big((s^1, s^2), \phi'_{12}\big) & \textbf{if} \;\; \phi_{12} \equiv \neg \phi'_{12} \\ s^k \models_k \mathcal{S}_{\trianglelefteq \frac{\left[p + G[\pi(R_1 R_2, \psi_{12}) - \pi((\alpha_1, \alpha_2), \psi_{12})]\right]}{G\pi_j(\alpha_j)}}(\alpha_k) & \textbf{if} \;\; \phi_{12} \equiv \xi_{12} \equiv \mathcal{S}_{\trianglelefteq p}(\psi_{12}) \wedge \\ & \quad (\alpha_1, \alpha_2) \in DecSat(\psi_{12}) \end{cases} \tag{4.3.13}$$

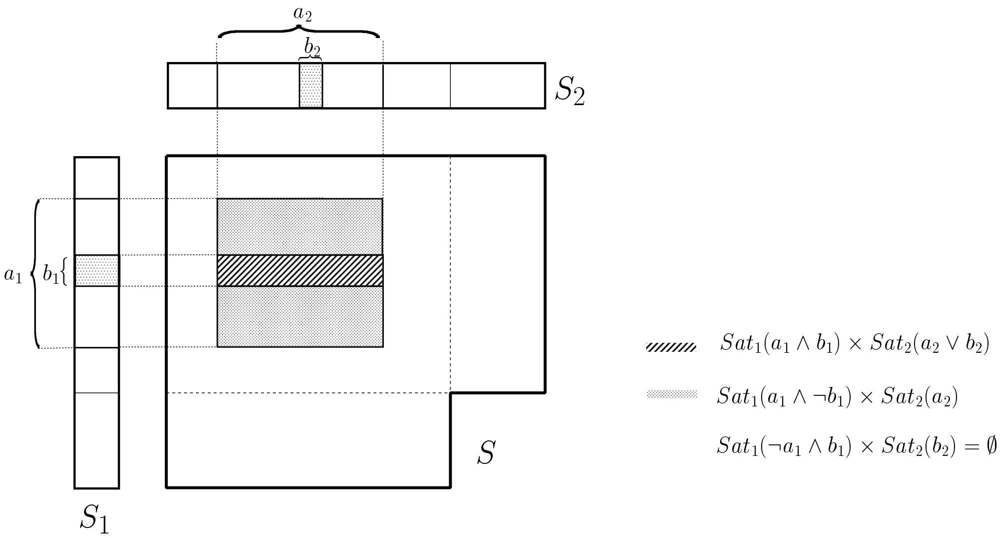


Figure 4.10: $[Sat_1(b_1) \subset Sat_1(a_1)] \wedge [Sat_2(b_2) \subset Sat_2(a_2)]$.

where $\mathcal{B}(Sat)$ is the set of all boolean combinations of propositions belonging to the set $Sat = Sat_1 \cup Sat_2$, Sat_k ($i \in 1,2$) being:

$$Sat_k ::= \{s^k \models_k \phi_k : s^k \in S_k, \phi_k \in \Phi_k\}$$

and G is the product-form normalisation *constant while for any* general *formula* ψ_{12} *and any pair of* single-component *formulae* $(\alpha_1, \alpha_2) \in DecSat(\psi_{12})$ *and the constants* $\pi(R_1R_2, \psi_{12})$, $\pi((\alpha_1, \alpha_2), \psi_{12})$ *and* $\pi_j(\alpha_j)$ *are defined as:*

$$\begin{aligned}
\pi(R_1R_2, \psi_{12}) &= \sum_{(\delta_1,\delta_2) \in DecSat(\psi_{12})} \Big[\sum_{t^k \in Sat_{k,R}(\delta_k)} \pi_k(t^k) \sum_{t^j \in Sat_{j,R}(\delta_j)} \pi_j(t^j) \Big] \\
\pi((\alpha_1, \alpha_2), \psi_{12}) &= \sum_{\substack{(\delta_1,\delta_2) \in DecSat(\psi_{12}) \\ (\delta_1,\delta_2) \neq (\alpha_1,\alpha_2)}} \Big[\sum_{t^k \in Sat_k(\delta_k)} \pi_k(t^k) \sum_{t^j \in Sat_j(\delta_j)} \pi_j(t^j) \Big] \\
\pi_j(\alpha_j) &= \sum_{t^j \in Sat_j(\alpha_j)} \pi_j(t^j)
\end{aligned}$$

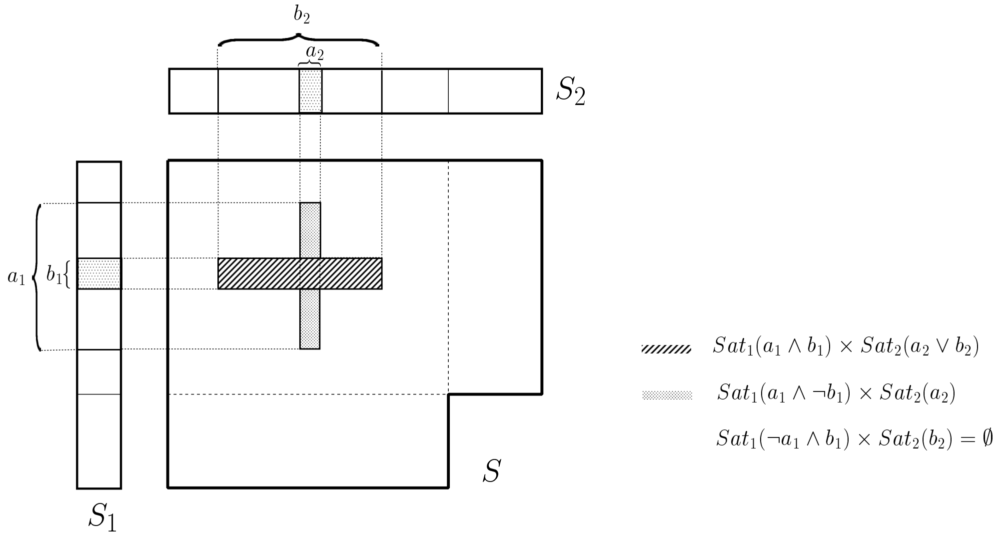


Figure 4.11: $[Sat_1(b_1) \subset Sat_1(a_1)] \wedge [Sat_2(a_2) \subset Sat_2(2_2)]$.

The constants $\pi_j(\alpha_j)$, $\pi((\alpha_1,\alpha_2),\psi_{12})$ and $\pi(R_1R_2,\psi_{12})$ appearing in the definition of *cond*(), are measures concerning the long-run behaviour of the Boucherie process and its components. Let us interpret them. Having in mind that (α_1,α_2) represents one of the partitions of $Sat(\psi_{12})$ by means of $DecSat(\psi_{12})$, then $\pi_j(\alpha_j)$ represents the probability for component $\mathcal{M}_j$ to satisfy the formula α_j in the long-run (i.e. the long-run probability for the *projection* onto S_j of the part of $Sat(\psi_{12})$ associated with (α_1,α_2)).

On the other hand, $\pi((\alpha_1,\alpha_2),\psi_{12})$ is defined as the sum of the *steady-state* probability of states in $Sat_1(\delta_1) \times Sat_2(\delta_2)$ for every pair $(\delta_1,\delta_2) \neq (\alpha_1,\alpha_2)$. It should be noted that this value deviates from the *steady-state* probability of the complement of the part associated with (α_1,α_2) (i.e. $Sat(\psi_{12}) \setminus [Sat_1(\alpha_1) \times Sat_2(\alpha_2) \setminus (R_1R_2)]$) by a factor which depends on the part of $Sat_1(\delta_1) \times Sat_2(\delta_2)$ which falls in the *prohibited* area (i.e. $[Sat_1(\delta_1) \times Sat_2(\delta_2)] \cap [R_1R_2]$). To understand what that means, let us consider an example. Figure 4.12 depicts what the decomposition of $Sat((a_1 \wedge a_2) \vee (b_1 \wedge b_2))$ looks like when some among the states satisfying the atoms a_1, b_1 and a_2, b_2 are such that the component holds R. We already know (see previous example) that

$$DecSat((a_1 \wedge a_2) \vee (b_1 \wedge b_2)) = \{((a_1 \wedge b_1),(a_2 \vee b_2)),(a_1 \wedge \neg b_1, a_2),(\neg a_1 \wedge b_1, b_2)\}$$

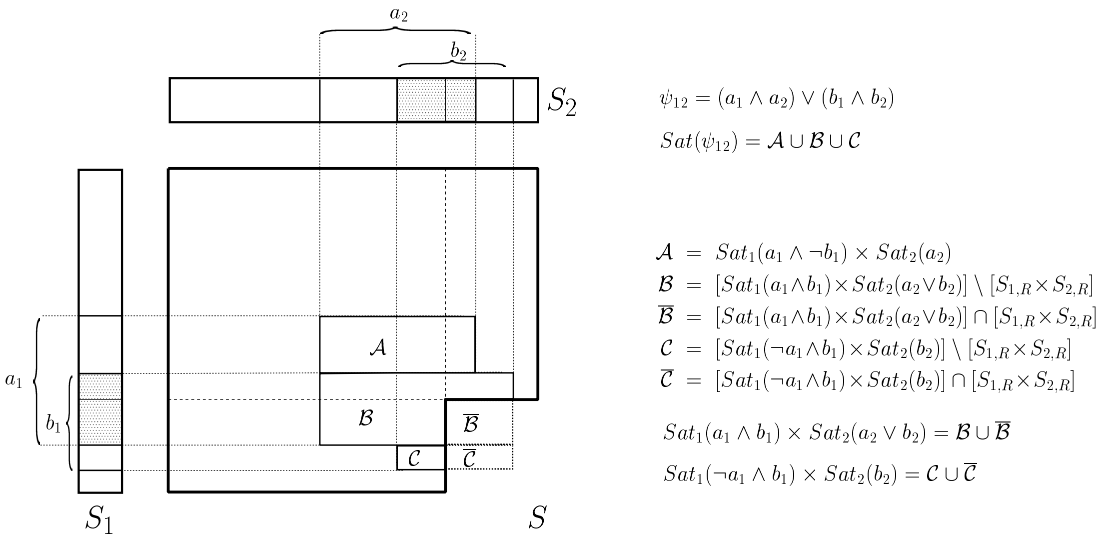


Figure 4.12: Meaning of the *deviation* $\pi(R_1R_2, (a_1 \wedge a_2) \vee (b_1 \wedge b_2)) = \pi(\mathcal{A}) + \pi(\mathcal{B})$.

providing a three part partition of $Sat((a_1 \wedge a_2) \vee (b_1 \wedge b_2))$, whose elements, $\mathcal{A}, \mathcal{B}$ and $\mathcal{C}$ are respectively associated with the pairs $(a_1 \wedge \neg b_1, a_2)$, $((a_1 \wedge b_1), (a_2 \vee b_2))$ and $(\neg a_1 \wedge b_1, b_2)$. We observe that, in this case, the intersection of the products $[Sat_1(\delta_1) \times Sat_2(\delta_2)] \cap [R_1R_2]$ is not empty with two of the three pairs in $DecSat((a_1 \wedge a_2) \vee (b_1 \wedge b_2))$, namely $((a_1 \wedge b_1), (a_2 \vee b_2))$ and $(\neg a_1 \wedge b_1, b_2)$. We named such not empty intersections, respectively $\overline{\mathcal{B}}$ (the one regarding $((a_1 \wedge b_1), (a_2 \vee b_2))$) and $\overline{\mathcal{C}}$ (the one regarding $(\neg a_1 \wedge b_1, b_2)$). Now, if we pick up a pair, say $(\neg a_1 \wedge b_1, b_2)$, then the value of the constant $\pi((a_1 \wedge \neg b_1, a_2), (a_1 \wedge a_2) \vee (b_1 \wedge b_2))$, is given by the sum of the *steady state* probability of the areas determined by the other pairs, namely $\mathcal{A}$ and $(\mathcal{B} \cup \overline{\mathcal{B}})$, hence:

$$\begin{aligned}
\pi((a_1 \wedge \neg b_1, a_2), (a_1 \wedge a_2) \vee (b_1 \wedge b_2)) &= \pi(\mathcal{A}) + \pi(\mathcal{B} \cup \overline{\mathcal{B}}) \\
&= \pi(\mathcal{A}) + \pi(\mathcal{B}) + \pi(\overline{\mathcal{B}})
\end{aligned}$$

This value deviates by a factor $\pi(\overline{\mathcal{B}})$ from $\pi(\mathcal{A} \cup \mathcal{B}) = \pi(\mathcal{A}) + \pi(\mathcal{B})$, which is the probability of satisfying the formula $(a_1 \wedge a_2) \vee (b_1 \wedge b_2)$ without being in any of the states associated with the pair $(\neg a_1 \wedge b_1, b_2)$, in the long-run.

We note that, it is not always the case that the value of the constant $\pi((\alpha_1, \alpha_2), \psi_{12})$ differs from the probability of the complement $Sat(\psi_{12}) \setminus [Sat_1(\alpha_1) \times Sat_2(\alpha_2) \setminus (R_1R_2)]$.

In fact, the *deviation factor* is null either if none of the parts (δ_1, δ_2) intersect the *prohibited* area $(R_1 R_2)$ or if the only part that does that is (α_1, α_2).

Finally, the constant $\pi(R_1R_2, \psi_{12})$, represents the sum of the *steady-state* probability of the amount of each area $(Sat_1(\delta_1) \times Sat_2(\delta_2))$, where $(\delta_1, \delta_2) \in DecSat(\psi_{12})$, which falls in the *prohibited* area. Referring to the example depicted in Figure 4.12 we have that

$$\pi(R_1R_2, (a_1 \wedge a_2) \vee (b_1 \wedge b_2)) = \pi(\overline{\mathcal{B}}) + \pi(\overline{\mathcal{C}})$$

Again, we observe that it is not always the case that $\pi(R_1R_2, \psi_{12}) > 0$. If none of the parts $(\delta_1, \delta_2) \in DecSat(\psi_{12})$ intersects R_1R_2 then clearly $\pi(R_1R_2, \psi_{12}) = 0$. If we refer, for example, to Figure 4.9 then we have that $\pi(R_1R_2, (a_1 \wedge a_2) \vee (b_1 \wedge b_2)) = 0$.

The definition of the function $cond()$ tells us that in order to check that the probability for a Boucherie process to satisfy ψ_{12} at steady-state matches a bound p, we have to chose one of the partitions of $Sat(\psi_{12})$ by $DecSat(\psi_{12})$, namely the part characterised by the pair of formulae (α_1, α_2), and check either that the probability for component $\mathcal{M}_1$ to satisfy α_1 at steady-state respects a derived bound $\hat{p}^1$ or that the probability for component $\mathcal{M}_2$ to satisfy α_2 at steady-state respects a derived bound $\hat{p}^2$, where the derived bounds $\hat{p}^1$ and $\hat{p}^2$ depend on the chosen part (α_1, α_2).

This provides us with the compositional result we were looking for: the computation of the *steady-state* distribution for the Boucherie process's components gives us enough means to check properties involving the *steady-state* probability of the Boucherie process itself.

The next Theorem proves that the results suggested by the definition of the function $cond()$ are actually correct.

Theorem 4.3.2 *Let $\mathcal{M} = (S, \mathbf{Q}, L)$ be a bidimensional Boucherie process, then for any* general *formula ϕ_{12} as in (4.3.10) and any state $(s^1, s^2) \in S$, the following holds:*

$$(s^1, s^2) \models \phi_{12} \iff cond\big((s^1, s^2), \phi_{12})\big)$$

where $cond()$ is as in Definition 4.3.4.

Proof. By structural induction on the definition of $cond()$.

base case: $\phi_{12} \equiv \phi_1 \wedge \phi_2$.
From Definition 4.3.4 we have that

$$cond((s^1, s^2), \phi_1 \wedge \phi_2) = s^1 \models_1 f_t(\phi_1) \ \textbf{ and } \ s^2 \models_2 f_t(\phi_2)$$

Hence, we aim to show that:

$$(s^1, s^2) \models \phi_1 \wedge \phi_2 \Longleftrightarrow s^1 \models_1 f_t(\phi_1) \textbf{ and } s^2 \models_2 f_t(\phi_2)$$

($\Rightarrow$) if $(s^1, t^2) \models \phi_1 \wedge \phi_2$ then $(s^1, t^2) \models \phi_1$ and $(s^1, t^2) \models \phi_2$. Thus, from Theorem 4.3.1,also $s^1 \models_1 f_t(\phi_1)$ and $s^2 \models_2 f_t(\phi_2)$, which proves ($\Rightarrow$).

($\Leftarrow$) By reversing ($\Rightarrow$).

$\phi_{12} \equiv \mathcal{S}_{\trianglelefteq p}(\psi_{12})$.
In this case we have that:

$$cond((s^1, s^2), \phi_1 \wedge \phi_2) = s^k \models_k \mathcal{S}_{\trianglelefteq \frac{\left[p+G[\pi(R_1 R_2, \psi_{12}) - \pi((\alpha_1, \alpha_2), \psi_{12})]\right]}{G\pi_2(\alpha_2)}}(\alpha_k)$$

where $(\alpha_1, \alpha_2) \in DecSat(\psi_{12})$. For brevity here we consider only the case with $k = 1$, hence, we aim to show that:

$$\models \mathcal{S}_{\trianglelefteq p}(\psi_{12}) \Longleftrightarrow \models_1 \mathcal{S}_{\trianglelefteq \frac{\left[p+G[\pi(R_1 R_2, \psi_{12}) - \pi((\alpha_1, \alpha_2), \psi_{12})]\right]}{G\pi_2(\alpha_2)}}(\alpha_1)$$

($\Rightarrow$) if $\models \mathcal{S}_{\trianglelefteq p}(\psi_{12})$ then

$$G\Big[\sum_{(t^1, t^2) \in Sat(\psi_{12})} \pi_1(t^1)\pi_2(t^2)\Big] \trianglelefteq p$$

From Lemma 4.3.2, we know that

$$Sat(\psi_{12}) = \bigcup_{(\delta_1, \delta_2) \in DecSat(\psi_{12})} [Sat_1(\delta_1) \times Sat_2(\delta_2)] \setminus (S_{1,R} \times Sat_{2,R})$$

which, as a consequence of Remark 4.2.3, we can rewrite as

$$Sat(\psi_{12}) = \bigcup_{(\delta_1, \delta_2) \in DecSat(\psi_{12})} \big[[Sat_1(\delta_1) \times Sat_{2,\overline{R}}(\delta_2)] \cup [Sat_{1,\overline{R}}(\delta_1) \times Sat_{2,R}(\delta_2)]\big]$$

By substitution in the above inequality, we then have:

$$G \sum_{(\delta_1,\delta_2)\in DecSat(\psi_{12})} \Big[\sum_{t^1\in Sat_1(\delta_1)} \pi_1(t^1) \sum_{t^2\in Sat_{2,\overline{R}}(\delta_2)} \pi_2(t^2) + \sum_{t^1\in Sat_{1,\overline{R}}(\delta_1)} \pi_1(t^1) \sum_{t^2\in Sat_{2,R}(\delta_2)} \pi_2(t^2) \Big] \trianglelefteq p$$

which by adding to both sides of the inequality the term $G\pi(R_1R_2,\psi_{12})$, where $\pi(R_1R_2,\psi_{12})$ is as in Definition 4.3.4, results in:

$$G \sum_{(\delta_1,\delta_2)\in DecSat(\psi_{12})} \Big[\sum_{t^1\in Sat_1(\delta_1)} \pi_1(t^1) \sum_{t^2\in Sat_2(\delta_2)} \pi_2(t^2) \Big] \trianglelefteq p + G\pi(R_1R_2,\psi_{12})$$

hence

$$G\Big[\sum_{t^1\in Sat_1(\alpha_1)} \pi_1(t^1) \sum_{t^2\in Sat_2(\alpha_2)} \pi_2(t^2) \Big] \trianglelefteq p + G\cdot[\pi(R_1R_2,\psi_{12}) - \pi((\alpha_1,\alpha_2),\psi_{12})]$$

where $(\alpha_1,\alpha_2)\in DecSat(\psi_{12})$. Thus:

$$\Big[\sum_{t^1\in Sat_1(\alpha_1)} \pi_1(t^1) \Big] \trianglelefteq \frac{[p + G[\pi(R_1R_2,\psi_{12}) - \pi((\alpha_1,\alpha_2),\psi_{12})]]}{G\pi_2(\alpha_2)}$$

which proves $(\Rightarrow)$.

$(\Leftarrow)$ By reversing $(\Rightarrow)$.

inductive step:

1. $\phi_{12} \equiv \phi_k \wedge \phi'_{12}$.

From Definition 4.3.4 we know that for any state $(s^1,s^2)\in S$

$$cond((s^1,s^2),\phi_k\wedge\phi'_{12}) = s^k \models_k f_t(\phi_k) \textbf{ and } cond((s^1,s^2),\phi'_{12})$$

Hence we aim to prove that:

$$(s^1,s^2)\models \phi_k\wedge\phi'_{12} \Longleftrightarrow s^k \models_k f_t(\phi_k) \textbf{ and } cond((s^1,s^2),\phi'_{12}).$$

Let us assume that

$$(s^1,s^2)\models \phi'_{12} \iff cond((s^1,s^2),\phi'_{12}).$$

as *inductive hypothesis*.

($\Rightarrow$) if $(s^1,s^2) \models \phi_k \wedge \phi'_{12}$ then $(s^1,s^2) \models \phi_k$ and $(s^1,s^2) \models \phi'_{12}$. Then from Theorem 4.3.1 $s^k \models_k \phi_k$ and from the *inductive hypothesis* also $cond((s^1,s^2),\phi'_{12})$ which proves ($\Rightarrow$).

($\Leftarrow$) By reversing ($\Rightarrow$).

2. $\phi_{12} \equiv \phi'_{12} \wedge \phi''_{12}$
Similar to the one above.

3. $\phi_{12} \equiv \neg\phi'_{12}$.
From Definition 4.3.4 we know that for any state $(s^1,s^2) \in S$

$$cond((s^1,s^2),\neg\phi'_{12}) = \textbf{ not } cond((s^1,s^2),\phi'_{12})$$

Hence we aim to prove that:

$$(s^1,s^2) \models \neg\phi'_{12} \iff \textbf{ not } cond((s^1,s^2),\phi'_{12}).$$

Let us assume the following *inductive hypothesis*:

$$(s^1,s^2) \models \phi'_{12} \iff cond((s^1,s^2),\phi'_{12})$$

($\Rightarrow$) Trivial consequence of the *inductive hypothesis*.

($\Leftarrow$) Trivial consequence of the *inductive hypothesis*.

□

Example 4.3.2 (Decomposed checking for *general* formulae) *Referring to the Boucherie process of our running example, let us suppose we are interested in checking that, in the long-run, there is at least a* 80% *probability of having at least one component in an "operative" state (i.e. not idle). This property can be expressed by means of the following general formula:*

$$\mathcal{S}_{\geq 0.8}(\neg(idle_1 \wedge idle_2))$$

One possibility to check whether such a formula is valid with respect to the Boucherie process, is to calculate the set $Sat((\neg(idle_1 \wedge idle_2)) \subset S$ by applying the CSL model-checking algorithm to the state-space S. By application of the decomposed semantics for general formulae given by the function cond() a different approach is possible. From Definition 4.3.4, we know that

$$\models \mathcal{S}_{\geq 0.8}(\neg(idle_1 \wedge idle_2)) \Longleftrightarrow \models_k \mathcal{S}_{\trianglelefteq \frac{\left[0.8+G[\pi(R_1R_2,\neg(idle_1\wedge idle_2))-\pi((\alpha_1,\alpha_2),\neg(idle_1\wedge idle_2))]\right]}{G\pi_j(\alpha_j)}}(\alpha_k)$$

where $(\alpha_1,\alpha_2) \in DecSat(\neg(idle_1 \wedge idle_2))$. Hence, as a first step, we have to determine $DecSat(\neg(idle_1 \wedge idle_2))$, which is:

$$DecSat(\neg(idle_1 \wedge idle_2)) = \{(\neg idle_1, tt), (idle_1, \neg idle_2)\}$$

Now we can chose a pair, say $(idle_1, \neg idle_2)$, from $DecSat(\neg(idle_1 \wedge idle_2))$ and consequently we compute the value of the constant $\pi((idle_1, \neg idle_2), (\neg(idle_1 \wedge idle_2))$ which is

$$\begin{aligned} \pi((idle_1, \neg idle_2), \neg(idle_1 \wedge idle_2)) &= \pi_1(\neg idle_1)\pi_2(S_2) \\ &= [\pi_1(s_{11}) + \pi_1(s_{12}) + \pi_1(s_{13}) + \pi_1(s_{14}) + \pi_1(s_{15})] \cdot 1 \\ &= 1 - \pi_1(s_{10}) \end{aligned}$$

In order to compute the value of the other constant $\pi(R_1R_2, \neg(idle_1 \wedge idle_2))$, we observe that, we have to consider the intersection with the prohibited area R_1R_2 of each pair in $DecSat(\neg(idle_1 \wedge idle_2))$. These are given by $Sat_{1,R}(\neg idle_1) \times S_{2,R}$ and $Sat_{1,R}(idle_1) \times Sat_{2,R}(\neg idle_2)$. Hence

$$\begin{aligned} \pi(R_1R_2, \neg(idle_1 \wedge idle_2)) &= [1 - \pi_1(idle_1)][1 - \pi_2(idle_2)] + \pi_1(idle_1)[1 - \pi_2(idle_2)] \\ &= 1 - \pi_2(idle_2) = 1 - \pi_2(s_{20}) \end{aligned}$$

Finally, we can choose the component we want to refer to, meaning the component we want to check the derived steady-state property against; say we are interested in component $\mathcal{M}_1$. In that case the derived formula of interest we want to check the steady-state probability of, is the first element of the pair $(idle_1, \neg idle_2)$ we previously picked, *namely $idle_1$. Thus the remaining constant we need to calculate is $\pi_2(\neg idle_2)$ representing the steady-state probability for the other component, $\mathcal{M}_2$, to satisfy $\neg idle_2$. That is given by*

$$\pi_2(\neg idle_2) = 1 - \pi_2(idle_1) = 1 - \pi_2(s_{20})$$

We are now able to compute the derived probability bound against which we aim to check the steady-state probability of $Sat_1(idle_1)$. That is given by:

$$\frac{0.8 + G[\pi(R_1R_2, \neg(idle_1 \wedge idle_2)) - \pi((idle_1, \neg idle_2), \neg(idle_1 \wedge idle_2))]}{G\pi_2(\neg idle_2)}$$

which is equal to

$$\frac{0.8 + G[[1 - \pi_2(s_{20})] - [1 - \pi_1(s_{10})]]}{G[1 - \pi_2(s_{20})]} = \frac{0.8 + G[\pi_2(s_{20}) + \pi_1(s_{10})]}{G[1 - \pi_2(s_{20})]}$$

Hence we have that checking the general formula $\mathcal{S}_{\geq 0.8}(\neg(idle_1 \wedge idle_2))$ with respect to the Boucherie process $\mathcal{M}$ is equivalent to check the single-component formula $\mathcal{S}_{\geq \frac{0.8+G[[1-\pi_2(s_{20})]-[1-\pi_1(s_{10})]]}{G[1-\pi_2(s_{20})]}}(idle_1)$ with respect to component $\mathcal{M}_1$

$$\models \mathcal{S}_{\geq 0.8}(\neg(idle_1 \wedge idle_2)) \Longleftrightarrow \models_1 \mathcal{S}_{\geq \frac{0.8+G[\pi_2(s_{20})+\pi_1(s_{10})]}{G[1-\pi_2(s_{20})]}}(idle_1)$$

Equivalently, we could have chosen to find a decomposed equivalence with respect to the other component, $\mathcal{M}_2$. In that case we would consider the second element of the (previously chosen) pair $(idle_1, \neg idle_2)$ as the target for the steady-state measure and we would need to compute the value for the constant

$$\pi_1(idle_1) = \pi_1(s_{10})$$

As a result the following equivalence holds as well

$$\models \mathcal{S}_{\geq 0.8}(\neg(idle_1 \wedge idle_2)) \Longleftrightarrow \models_2 \mathcal{S}_{\geq \frac{0.8+G[\pi_2(s_{20})+\pi_1(s_{10})]}{G\pi_1(s_{10})}}(\neg idle_2)$$

The advantage of using the compositional semantics, in this case, is that the complexity of computing the satisfiability set for the formula $idle_1$ with respect to the component process $\mathcal{M}_1$ is lower than the complexity for the computation of the satisfiability set of $\neg(idle_1 \wedge idle_2)$ with respect to the product process $\mathcal{M}$. That difference relies on the ratio between the state-spaces' dimension

$$SF = \frac{|S|}{|S_1|}$$

The bigger the ratio SF (Savings Factor) is, the bigger is the saving, in terms of complexity, gained through application of the compositional semantics.

Chapter 5

Compositional CSL model checking: Next formulae

5.1 Introduction

In the previous chapter the existence of a compositional semantics for a subset of the CSL where probabilistic path formulae, like $\mathcal{P}_{\trianglelefteq p}(\varphi)$, were disallowed, has been shown. In this chapter that syntax is extended and Next formulae are considered. However, we observe that, in order to derive compositional equivalences for path formulae (i.e. Next and Until), nesting of path connectives needs to be excluded. Thus, unlike the original CSL (see Definition 2.3.5), in this work we will admit only formulae which do not contain any *probabilistic* connective[1] as possible type of argument of a *probabilistic-path* operator. Complying with this restriction, it will be shown that a compositional method for checking Next formulae which refer to a bidimensional Boucherie process, can be derived. The chapter also presents a further relevant result, which regards the procedure for checking bounded Next formulae with respect to an arbitrary CMTC. It will be shown, in fact, that the algorithm for the computation of the state-vector $Prob(s, X^I\ \phi)$ provided in [5], is not correct and a revised version will be defined.

The chapter is organised in the following way: in the next section the syntax for *single-component* Next formulae is introduced and decomposed semantic equivalences

[1]Not even a probabilistic *steady-state* formula, like $\mathcal{S}_{\trianglelefteq p}(\psi)$, can be used as an argument of a *probabilistic* connective.

for both "simple" *single component* bounded Next formulae (Section 5.2.1) and *steady-state* properties referring to *single-component* bounded Next formulae (Section 5.2.2), are proved. Some examples are also included in order to show the correctness of these results with respect to the GIS Boucherie framework formerly introduced in Section 2.5.1. In Section 5.3 the algorithms for decomposed checking of such Next formulae are introduced (an algorithm for $\mathcal{P}_{\overline{\trianglelefteq p}}(X^I\ \psi)$ and an algorithm for $\mathcal{S}_{\trianglelefteq p}(\mathcal{P}_{\overline{\trianglelefteq p}}(X^I\ \psi))$ are presented). Finally, in Section 5.4, *general* Next formulae are considered and a procedure for their decomposed verification is defined. In this section, the problem with the original version of the algorithm for computing $Prob(s, X^I\ \phi)$, is pointed out by means of a simple example. The revised algorithm is then presented and proved to fix the error with respect the considered example.

5.2 Compositional semantics for *single-component* Next formulae

In this section the application of the time bounded Next connective to *single-component* formulae, ψ_k, is considered and a compositional semantics is derived. The syntax of the logic we refer to is derived from the one described in (4.3.2), by adding the production for probabilistic path formulae φ_k. This results in:

$$
\begin{aligned}
\phi_k &::= \psi_k \mid \varphi_k \mid \xi_k \mid \phi_k \wedge \phi_k \mid \neg\phi_k \\
\psi_k &::= tt \mid a_k \mid \psi_k \wedge \psi_k \mid \neg\psi_k \\
\xi_k &::= \mathcal{S}_{\trianglelefteq p}(\psi_k) \mid \mathcal{S}_{\trianglelefteq p}(\varphi_k) \\
\varphi_k &::= \mathcal{P}_{\trianglelefteq p}(X^I(\psi_k))
\end{aligned}
\tag{5.2.1}
$$

For the time being we admit only the bounded Next as the possible type of path formula φ_k. Moreover, as we have already mentioned, the possibility for nesting path connectives is disallowed. *Steady-state* formulae are also excluded from the possible type of argument for a probabilistic path connective, because of their *model-like* rather than *state-like* semantics with respect to ergodic models (see the analysis of CSL semantic equivalences for ergodic models in Section 3.5). As a result the argument of the bounded Next connective X^I can only be a formula which itself involves neither a path connective, nor the *steady-state* operator, i.e. a ψ_k formula. Finally, the probabilistic *steady-state* operator can be applied also to probabilistic path formulae φ_k other than non-probabilistic formulae ψ_k, enriching, in this way, the expressiveness of the *steady-state* analysis: "future evolutions" (i.e. paths) are added to simple "state properties" in the criteria for characterising the "long-run" behaviour of interest.

In the remainder of this chapter we will show that a compositional approach for checking the formulae of the syntax in (5.2.1) is possible. This basically will require us to extend the *transformation function* introduced in Definition 4.3.2 in order to cope with probabilistic Next formulae. We start with the analysis of a compositional approach for probabilistic time-bounded Next formulae (section 5.2.1); those results will be the basis to determine the compositional semantics for *steady-state* formulae whose argument is a probabilistic bounded Next (see Section 5.2.2).
In order to improve the readability, the proof of some preparatory Lemmas has been moved to Appendix A: only fundamental theorems are reported here together with their proof.

5.2.1 Bounded Next ($\mathcal{P}_{\trianglelefteq p}(X^I(\psi_k))$)

In this section, probabilistic time-bounded Next formulae like, $\mathcal{P}_{\trianglelefteq p}(X^I(\psi_k))$), are considered. The equivalences showing the existence of a compositional semantics for such formulae are given in Theorem 5.2.1. The result of Theorem 5.2.1, relies on the characterisation of an "equivalent" probability bound p' whose value is provided by the function $h()$ introduced in the following definition.

Definition 5.2.1 (Equivalent Next's probability and time bound) *Let* (s^1,s^2) *be a state of a bidimensional Boucherie process,* ψ_k *a non-probabilistic formula as in the syntax described in (5.2.1) referring to component* $\mathcal{M}_k$*,* $p\in[0,1]$ *a probability bound and* $I=[a,b]\subseteq\mathbb{R}_{\geq 0}$ *a time bounding interval. The function*

$$h():([0,1],\Psi_k,CTMC,S,2^{\mathbb{R}_{\geq 0}})\rightarrow[0,1]\times 2^{\mathbb{R}_{\geq 0}}$$

is defined as follows:

$$h(p,\psi_k,\mathcal{M}_k,(s^1,s^2),I)=\begin{cases}\left(\frac{p}{p^k(s^1,s^2)},I'(s^1,s^2)\right) & \textbf{if } [(s^1,s^2)\in R\_free]\wedge[s^k \not\models_k \psi_k]\\ \\ \left(\frac{p-p^j(s^1,s^2)}{p^k(s^1,s^2)},I'(s^1,s^2)\right) & \textbf{if } [(s^1,s^2)\in R\_free]\wedge[s^k \models_k \psi_k]\\ \\ (p,I) & \textbf{if } (s^1,s^2)\in R_k\end{cases} \tag{5.2.2}$$

where $p^k(s^1,s^2)$ *is the probability of making a k-move out of state* (s^1,s^2) *and* $I'(s^1,s^2)=[\frac{a}{p^k(s^1,s^2)},\frac{b}{p^k(s^1,s^2)}]$.

The function $h()$ provides us with a pair representing, respectively, the *equivalent probability bound* and the *equivalent time bound interval* for bounded Next *single-component* formulae. It will be shown, in fact, that checking a formula like $\mathcal{P}_{\trianglelefteq p}(X^I(\psi_k))$ with respect to a state (s^1,s^2) of the Boucherie process, is equivalent, under certain circumstances, to checking the formula $\mathcal{P}_{\trianglelefteq \hat{p}}(X^{\hat{I}}(\psi_k))$ with respect to the state s^k of component $\mathcal{M}_k$, where $\hat{p}$ and $\hat{I}$ are, respectively, the first and second component of the pair $h(p,\psi_k,\mathcal{M}_k,(s^1,s^2),I)=(\hat{p},\hat{I})$.

In Chapter 4, it has been shown that the equivalences characterising the compositional semantics for non-path, *single-component* formulae are obtained by means of a *transformation function*, namely $f_t()$. Unlike the non-path formulae case, the *transformation* of a path formula is state dependent, other than formula dependent: the decomposed equivalent for a path formula like $\mathcal{P}_{\trianglelefteq p}(\varphi)$ which is to be checked against a state $(s^1,s^2)\in S$ of the Boucherie process, depends both on $\mathcal{P}_{\trianglelefteq p}(\varphi)$ and on the considered state (s^1,s^2).

In the next Theorem the compositional equivalences concerning probabilistic time-bounded *single-component* Next formulae, are proved.

Theorem 5.2.1 (Bounded *single-component* Next) *Let* $(s^1,s^2)\in S$ *be a state of a bidimensional Boucherie process,* ψ_k *a non-probabilistic single-component formula as in (5.2.1),* $p\in[0,1]$ *a probability bound,* $\trianglelefteq\in\{<,\leq,\geq,>\}$ *and* $I=[a,b]\subseteq\mathbb{R}_{\geq 0}$ *a time* *interval. The following equivalences hold:*

$$(s^1,s^2) \models \mathcal{P}_{\trianglelefteq p}(X^I(\psi_k)) \Longleftrightarrow \begin{cases} s^k \models_k \mathcal{P}_{\trianglelefteq \hat{p}}(X^{\hat{I}}(\psi_k)) & \textbf{if } (s^1,s^2) \notin R_j \\ s^k \models_k \psi_k & \textbf{if } [(s^1,s^2) \in R_j] \wedge low(\trianglelefteq,p) \wedge \\ & \quad [e^{-E_j(s^j)a} - e^{-E_j(s^j)b}] \trianglelefteq p \\ s^k \models_k \neg\psi_k & \textbf{if } [(s^1,s^2) \in R_j] \wedge up(\trianglelefteq,p) \\ & \quad [e^{-E_j(s^j)a} - e^{-E_j(s^j)b}] \ntrianglelefteq p \\ s^k \models_k tt & \textbf{if } [(s^1,s^2) \in R_j] \wedge up(\trianglelefteq,p) \\ & \quad [e^{-E_j(s^j)a} - e^{-E_j(s^j)b}] \trianglelefteq p \\ s^k \models_k \neg tt & \textit{otherwise} \end{cases} \tag{5.2.3}$$

where $h(p,\psi_k,\mathcal{M}_k,(s^1,s^2),I) = (\hat{p},\hat{I})$ and $low(\trianglelefteq,p)$, $up(\trianglelefteq,p)$ are the conditions characterised in Definition 3.4.2.

Proof. From Proposition 2.3.2 we know that the probability measure for the paths starting at a state (s^1,s^2) and satisfying the bounded Next formula $X^I(\psi_k)$, is given by:

$$Prob((s^1,s^2),X^I(\psi_k)) = \left(e^{-E(s^1,s^2)\cdot a} - e^{-E(s^1,s^2)\cdot b}\right) \cdot \sum_{(t^1,t^2)\models\psi_k} \mathbf{P}((s^1,s^2),(t^1,t^2)) \tag{5.2.4}$$

We need then to distinguish between the three different conditions characterising the equivalence (5.2.4).

1. $(s^1,s^2) \notin R_j$.

We aim to prove that

$$(s^1,s^2) \models \mathcal{P}_{\trianglelefteq p}(X^I(\psi_k)) \Longleftrightarrow s^k \models_k \mathcal{P}_{\trianglelefteq \hat{p}}(X^{\hat{I}}(\psi_k))$$

where $h(p,\psi_k,\mathcal{M}_k,(s^1,s^2),I) = (\hat{p},\hat{I})$. A further distinction is needed, as the complement of R_j is partitioned into R_k and R_{free}.

1.a $(s^1,s^2) \in R_k$. If $(s^1,s^2) \in R_k$, then $\hat{p} = p$ and $\hat{I} = I$ (see Definition 5.2.1). Moreover the emanating rate $E(s^1,s^2) = E_k(s^k)$ depends only on the emanating rate of s^k (see Remark 4.2.2), hence the probability measure described in (5.2.4) becomes :

$$Prob((s^1,s^2),X^I(\psi_k)) = \left(e^{-E_k(s^k)\cdot a} - e^{-E_k(s^k)\cdot b}\right) \cdot \sum_{(t^1,t^2)\models\psi_k} \mathbf{P}((s^1,s^2),(t^1,t^2)) \quad (5.2.5)$$

Furthermore, from the compositional semantics of non-probabilistic formulae (see Theorem 4.3.1), we know that $(t^1,t^2) \models \psi_k \Leftrightarrow t^k \models_k \psi_k$ and also, since we are assuming $(s^1,s^2) \in R_k$, the only admitted *move*s are *k-move*s and they have the same probability to occur in $\mathcal{M}$ as they have in $\mathcal{M}_k$ (see Remark 4.2.2). As a result, the sum in (5.2.5) can be reformulated resulting in:

$$Prob((s^1,s^2),X^I(\psi_k)) = \left(e^{-E_k(s^k)\cdot a} - e^{-E_k(s^k)\cdot b}\right) \cdot \sum_{t^k\models_k\psi_k} \mathbf{P}_1(s^k,t^k) \quad (5.2.6)$$

Hence

$$Prob((s^1,s^2),X^I(\psi_k)) = Prob_k(s^k,X^I(\psi_k))$$

which clearly proves that

$$Prob((s^1,s^2),X^I(\psi_k)) \trianglelefteq p \Longleftrightarrow Prob_k(s^k,X^I(\psi_k)) \trianglelefteq p.$$

1.b $(s^1,s^2) \in R_{free}$. If $(s^1,s^2) \in R_{free}$, the emanating rate $E(s^1,s^2) = E_1(s^1) + E_2(s^2)$ (see Remark 4.2.2) and also, again, $(t^1,t^2) \models \psi_k \Leftrightarrow t^k \models_k \psi_k$.

$$\begin{aligned} Prob((s^1,s^2),X^I(\psi_k)) = \left(e^{-[E_1(s^1)+E_2(s^2)]\cdot a} - e^{-[E_1(s^1)+E_2(s^2)]\cdot b}\right)\cdot \\ \sum_{(t^1,t^2)\models\psi_k} \mathbf{P}((s^1,s^2),(t^1,t^2)) \end{aligned} \quad (5.2.7)$$

A further distinction needs to be considered though:
1.b.1 $s^k \not\models_k \psi_k$. In this case $\hat{p} = \frac{p}{p^k(s^1,s^2)}$ and $\hat{I} = [\frac{a}{p^k(s^1,s^2)}, \frac{b}{p^k(s^1,s^2)}]$ (see Definition 5.2.1). The assumption $s^k \not\models_k \in \psi_k$, allows us to exactly determine which among the successors of (s^1,s^2) satisfy the argument ψ_k of the Next operator. We observe that if $s^k \not\models_k \psi_k$ then $(s^1,s^2) \not\models \psi_k$ but then clearly also every successor state (t^1,t^2) corresponding to a *j-move* from (s^1,s^2) (i.e. such that $\mathbf{Q}((s^1,s^2),(t^1,t^2)) > 0$ and $t^k = s^k$) will not satisfy ψ_k, while a k-successor of (s^1,s^2) (i.e. a state (t^1,t^2) such that $\mathbf{Q}((s^1,s^2),(t^1,t^2)) > 0$

and $t^j = s^j$) will satisfy ψ_k if and only if $t^k \models_k \psi_k$. As a result the sum in (5.2.7) can be re-written as:

$$\sum_{(t^1,t^2)\models\psi_k} \mathbf{P}((s^1,s^2),(t^1,t^2)) = p^k(s^1,s^2) \cdot \sum_{t^k\models_k\psi_k} \mathbf{P}_k(s^k,t^k)$$

which substituted in (5.2.7) gives:

$$\begin{aligned} Prob((s^1,s^2),X^I(\psi_k)) = \Big(e^{-[E_1(s^1)+E_2(s^2)]\cdot a} - e^{-[E_1(s^1)+E_2(s^2)]\cdot b}\Big)\cdot \\ p^k(s^1,s^2) \cdot \sum_{t^k\models_k\psi_k} \mathbf{P}_k(s^k,t^k) \end{aligned} \tag{5.2.8}$$

from which, straightforwardly follows,

$$Prob((s^1,s^2),X^{[a,b]}(\psi_k)) \trianglelefteq p \Longleftrightarrow Prob_k(s^k,X^{[\frac{a}{p^k(s^1,s^2)},\frac{b}{p^k(s^1,s^2)}]}(\psi_k)) \trianglelefteq \frac{p}{p^k(s^1,s^2)}$$

which proves the theorem in this case.

1.b.2 $s^k \models_k \psi_k$. In this case $\hat{p} = \frac{p-p^j(s^1,s^2)}{p^k(s^1,s^2)}$ and $\hat{I} = [\frac{a}{p^k(s^1,s^2)}, \frac{b}{p^k(s^1,s^2)}]$ (see Definition 5.2.1). Again, from the assumption $s^k \models_k \psi_k$, we are able to exactly determine which among the successors of (s^1,s^2) satisfy ψ_k. In fact, if $s^k \models_k \psi_k$ then $(s^1,s^2) \models \psi_k$ but then clearly also every successor state (t^1,t^2) corresponding to a *j-move* from (s^1,s^2) will satisfy ψ_k, while a k-successor of (s^1,s^2) will satisfy ψ_k if and only if $t^k \models_k \psi_k$. As a result the sum in (5.2.7) can be re-written as:

$$\begin{aligned} \sum_{(t^1,t^2)\models\psi_k} \mathbf{P}((s^1,s^2),(t^1,t^2)) &= p^k(s^1,s^2) \cdot \sum_{t^k\models_k\psi_k} \mathbf{P}_k(s^k,t^k) + p^j(s^1,s^2) \cdot \sum_{t^j\in S_j} \mathbf{P}_j(s^j,t^j) \\ &= p^k(s^1,s^2) \cdot \sum_{t^k\models_k\psi_k} \mathbf{P}_k(s^k,t^k) + p^j(s^1,s^2) \end{aligned}$$

which substituted in (5.2.7) gets:

$$\begin{aligned} Prob((s^1,s^2),X^I(\psi_k)) = \Big(e^{-[E_1(s^1)+E_2(s^2)]\cdot a} - e^{-[E_1(s^1)+E_2(s^2)]\cdot b}\Big)\cdot \\ \Big[p^k(s^1,s^2) \cdot \sum_{t^k\models_k\psi_k} \mathbf{P}_k(s^k,t^k) + p^j(s^1,s^2)\Big] \end{aligned} \tag{5.2.9}$$

from which, straightforwardly follows,

$$Prob((s^1,s^2),X^{[a,b]}(\psi_k)) \trianglelefteq p \Longleftrightarrow Prob_k(s^k,X^{[\frac{a}{p^k(s^1,s^2)},\frac{b}{p^k(s^1,s^2)}]}(\psi_k)) \trianglelefteq \frac{p-p^j(s^1,s^2)}{p^k(s^1,s^2)}$$

which proves the theorem in this case.

2. $(s^1,s^2) \in R_j$ and $low(\trianglelefteq,p)$ and $[e^{-E_j(s^j)a} - e^{-E_j(s^j)b}] \trianglelefteq p$.
In this case we aim to prove that

$$Prob((s^1,s^2),X^I(\psi_k)) \trianglelefteq p \Longleftrightarrow s^k \models_k \psi_k$$

Since we are assuming (s^1,s^2) to be in R_j (i.e. component $\mathcal{M}_j$ is holding the resource), then any successor (t^1,t^2) of (s^1,s^2) must be such $t^k = s^k$. Hence, $(s^1,s^2) \models \psi_k \Leftrightarrow (t^1,t^2) \models \psi_k$ for every successor (t^1,t^2). But then, as a consequence of Theorem 4.3.1, also $s^k \models_k \psi_k \Leftrightarrow (t^1,t^2) \models \psi_k$ for every successor (t^1,t^2), which means that only two situations are possible: either all or none amongst the successors of (s^1,s^2) satisfy ψ_k and this is characterisable in terms of the satisfiability of ψ_k with respect to s^k. If $s^k \not\models_k \psi_k$ then none of the successors of (s^1,s^2) satisfy ψ_k, thus clearly $Prob((s^1,s^2),X^I(\psi_k)) = 0$. On the other hand, if $s^k \models \psi_k$ then every successor of (s^1,s^2) satisfies ψ_k too, hence $Prob((s^1,s^2),X^I(\psi_k)) = [e^{-E_j(s^j)a} - e^{-E_j(s^j)b}]$. Then, since we are also assuming $[e^{-E_j(s^j)a} - e^{-E_j(s^j)b}] \trianglelefteq p$, clearly

$$Prob((s^1,s^2),X^I(\psi_k)) = [e^{-E_j(s^j)a} - e^{-E_j(s^j)b}] \trianglelefteq p \Longleftrightarrow s^k \models_k \psi_k$$

which proves this case.

3. $(s^1,s^2) \in R_j$ and $up(\trianglelefteq,p)$.
As for the previous case, we know that either all or none amongst the successors of (s^1,s^2) satisfy ψ_k, hence the probability measure $Prob((s^1,s^2),X^I(\psi_k))$ can be either zero or equal to $[e^{-E_j(s^j)a} - e^{-E_j(s^j)b}]$. However, since we are assuming an upper bound check for such a measure (i.e. $up(\trianglelefteq,p)$), then obviously

$$Prob((s^1,s^2),X^I(\psi_k)) = 0 \trianglelefteq p \Longleftrightarrow s^k \models_k \neg\psi_k$$

which proves also this case of the theorem.

□

Example 5.2.1 *Referring to the Boucherie process representing the GIS system of our running example (Figure 5.1 and Figure 5.2), let us consider the following bounded Next formulae:*

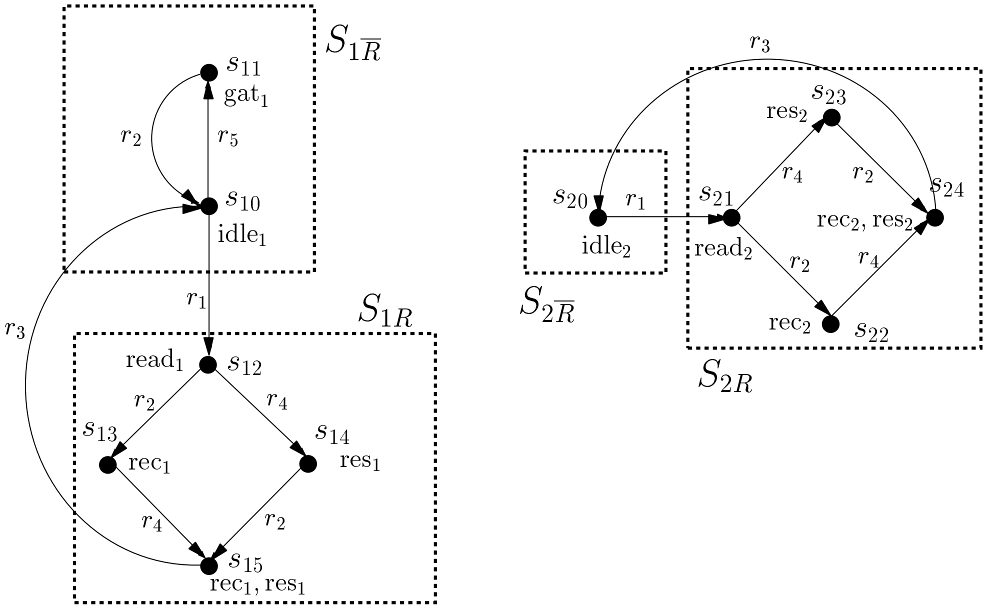


Figure 5.1: State space of the GIS components $\mathcal{M}_1$ and $\mathcal{M}_2$.

i) *Let us suppose we are interested in checking whether the probability of reaching a state where component $\mathcal{M}_1$ is reading the shared register, $\psi_1 \equiv read_1$, in one step and with a delay falling in the interval $I = [2,5]$, from the initial state (s_{10}, s_{20}) (i.e. both components are idle), has $p = 0.3$ as a lower bound. This is the case if*

$$(s_{10}, s_{20}) \models \mathcal{P}_{\geq 0.3}(X^{[2,5]}\ read_1)$$

Since $(s_{10}, s_{20}) \notin R_2$ then we are in the first case of (5.2.3), hence, we know that:

$$(s_{10}, s_{20}) \models \mathcal{P}_{\geq 0.3}(X^{[2,5]}\ read_1) \iff s_{10} \models_1 \mathcal{P}_{\geq \hat{p}}(X^{\hat{I}}\ read_1)$$

where $(\hat{p}, \hat{I}) = h(0.3, read_1, \mathcal{M}_1, (s_{10}, s_{20}), [2,5])$. We observe that, as $read_1$ is not satisfied in s_{10} (i.e. $s_{10} \not\models_1 read_1$), then from the definition of $h()$ (5.2.2) we have

$$h(0.3, read_1, \mathcal{M}_1, (s_{10}, s_{20}), [2,5]) = (\frac{0.3}{p^1(s_{10}, s_{20})}, [\frac{2}{p^1(s_{10}, s_{20})}, \frac{5}{p^1(s_{10}, s_{20})}])$$

and since the probability of making a 1-move out of (s_{10}, s_{20}) is

$$p^1(s_{10}, s_{20}) = \frac{E_1(s_{10})}{E_1(s_{10}) + E_2(s_{20})} = \frac{r_1 + r_5}{2r_1 + r_5}$$

then

$$h(0.3, read_1, \mathcal{M}_1, (s_{10}, s_{20}), [2,5]) = (\frac{0.3(2r_1 + r_5)}{(r_1 + r_5)}, [\frac{2(2r_1 + r_5)}{(r_1 + r_5)}, \frac{5(2r_1 + r_5)}{(r_1 + r_5)}])$$

Hence we would like to verify that checking $(s_{10}, s_{20}) \models \mathcal{P}_{\geq 0.3}(X^{[2,5]}\ read_1)$ *is equivalent to checking* $s_{10} \models_1 \mathcal{P}_{\geq \frac{0.3(2r_1+r_5)}{(r_1+r_5)}}(X^{[\frac{2(2r_1+r_5)}{(r_1+r_5)}, \frac{5(2r_1+r_5)}{(r_1+r_5)}]}\ read_1)$. *From Proposition 2.3.2 we can straightforwardly compute the probability of reaching in one step and with a delay within the bound* $I = [2,5]$ *a state where* $read_1$ *is true:*

$$\begin{aligned} Prob((s_{10}, s_{20}), X^{[2,5]}\ read_1) &= (e^{-(E_1(s_{10})+E_2(s_{20}))\cdot 2} - e^{-(E_1(s_{10})+E_2(s_{20}))\cdot 5}) \cdot \\ &\quad \sum_{(t^1,t^2)\models read_1} \mathbf{P}((s_{10}, s_{20}), (t^1, t^2)) \\ &= (e^{-2\cdot(2r_1+r_5)} - e^{-5\cdot(2r_1+r_5)}) \frac{r_1}{2r_1 + r_5} \end{aligned}$$

Similarly the probability of reaching from state s_{10} *in one step a state satisfying* $read_1$ *within a time in the derived equivalent interval* $\hat{I} = [\frac{2(2r_1+r_5)}{(r_1+r_5)}, \frac{5(2r_1+r_5)}{(r_1+r_5)}]$ *is:*

$$\begin{aligned} Prob(s_{10}, X^{[\frac{2(2r_1+r_5)}{(r_1+r_5)}, \frac{5(2r_1+r_5)}{(r_1+r_5)}]}\ read_1) &= (e^{-(E_1(s_{10}))\cdot\frac{2(2r_1+r_5)}{(r_1+r_5)}} - e^{-(E_1(s_{10}))\cdot\frac{5(2r_1+r_5)}{(r_1+r_5)}}) \cdot \\ &\quad \sum_{t^1\models_1 read_1} \mathbf{P}_1(s_{10}, t^1) \\ &= (e^{-2\cdot(2r_1+r_5)} - e^{-5\cdot(2r_1+r_5)}) \frac{r_1}{r_1 + r_5} \end{aligned}$$

Thus clearly

$$Prob((s_{10}, s_{20}), X^{[2,5]}\ read_1) = Prob(s_{10}, X^{[\frac{2(2r_1+r_5)}{(r_1+r_5)}, \frac{5(2r_1+r_5)}{(r_1+r_5)}]}\ read_1) \cdot p^1(s_{10}, s_{20})$$

which, as expected, proves

$$Prob((s_{10}, s_{20}), X^{[2,5]}\ read_1) \geq 0.3 \Longleftrightarrow Prob(s_{10}, X^{[\frac{2(2r_1+r_5)}{(r_1+r_5)}, \frac{5(2r_1+r_5)}{(r_1+r_5)}]}\ read_1) \geq \frac{0.3(2r_1 + r_5)}{(r_1 + r_5)}$$

ii) *Let us suppose we are interested in checking the probability of reaching in one step with no time bound (i.e.* $I = [0, \infty]$*), a state such that component* $\mathcal{M}_1$ *is idle (i.e.* $idle_1$*),*

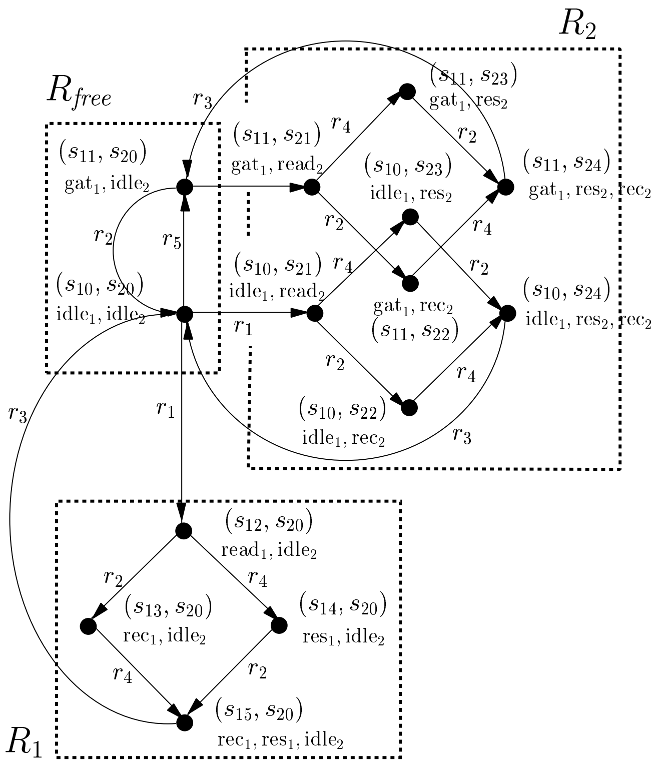


Figure 5.2: State space of the GIS product process $\mathcal{M}$

from the initial state (s_{10}, s_{20})*, with respect to the same probability bound* $p = 0.3$ *as in the previous case, which is:*

$$(s_{10}, s_{20}) \models \mathcal{P}_{\geq 0.3}(X^{[0,\infty]}\ idle_1)$$

Since component $\mathcal{M}_1$ *is actually idle in state* s_{10} *(i.e.* $s_{10} \models_1 idle_1$*), then the equivalent probability and time bound are given by the second case of (5.2.3), which is:*

$$h(0.3, idle_1, \mathcal{M}_1, (s_{10}, s_{20}), [2,5]) = (\frac{0.3 - p^2(s_{10}, s_{20})}{p^1(s_{10}, s_{20})}, [0, \infty])$$

where the probability of a 1*-move and of a* 2*-move out of* (s_{10}, s_{20}) *are respectively:*

$$p^1(s_{10}, s_{20}) = \frac{r_1 + r_5}{2r_1 + r_5} \qquad\qquad p^2(s_{10}, s_{20}) = \frac{r_1}{2r_1 + r_5}$$

As a result we would like to verify that checking $(s_{10}, s_{20}) \models \mathcal{P}_{\geq 0.3}(X^{[0,\infty]}\ idle_1)$ *is equivalent to checking* $s_{10} \models_1 \mathcal{P}_{\geq \frac{0.3(2r_1+r_5)-r_1}{(r_1+r_5)}}(X^{[0,\infty]}\ read_1)$*. The probability of reaching a state* $idle_1$ *from* $(s_{10}, s_{20}$ *is*

$$\begin{aligned} Prob((s_{10}, s_{20}), X^{[0,\infty]}\ read_1) &= 1 \cdot \sum_{(t^1,t^2) \models read_1} \mathbf{P}((s_{10}, s_{20}), (t^1, t^2)) \\ &= \frac{r_1}{2r_1 + r_5} \end{aligned}$$

On the other hand, as none amongst the successors of s_{10} *in* $\mathcal{M}_1$ *satisfies* $idle_1$*, then obviously:*

$$Prob(s_{10}, X^{[0,\infty]}\ read_1) = 0$$

Thus we aim to show that

$$\frac{r_1}{2r_1 + r_5} \geq 0.3 = \frac{3}{10} \Longleftrightarrow 0 \geq \frac{0.3(2r_1 + r_5) - r_1}{(r_1 + r_5)} = \frac{\frac{3}{10}(2r_1 + r_5) - r_1}{(r_1 + r_5)}$$

We have by rearrangement that

$$\frac{r_1}{2r_1 + r_5} \geq \frac{3}{10} \Leftrightarrow r_1 \geq \frac{3}{4} r_5$$

By substituting $r_1 \geq \frac{3}{4} r_5$ *in* $\frac{3}{10}(2r_1 + r_5) - r_1$ *we have that:*

$$\frac{3}{10}(\frac{6}{4} r_5 + r_5) - \frac{3}{4} r_5 \geq \frac{3}{10}(2r_1 + r_5) - r_1$$

Clearly, however $\frac{3}{10}(\frac{6}{4}r_5 + r_5) - \frac{3}{4}r_5 = 0$, *which proves*

$$\frac{r_1}{2r_1 + r_5} \geq 0.3 \Longleftrightarrow 0 \geq \frac{0.3(2r_1 + r_5) - r_1}{(r_1 + r_5)} = \frac{\frac{3}{10}(2r_1 + r_5) - r_1}{(r_1 + r_5)}$$

□

5.2.2 *Steady-state* bounded Next ($\mathcal{S}_{\trianglelefteq p}(\mathcal{P}_{\overline{\trianglelefteq p}}(X^I(\psi_k)))$)

Theorem 5.2.1 describes the equivalences which allow for decomposed checking of *single-component* time-bounded Next formulae. The next step is to consider the *steady-state* formulae whose argument is a *single-component* Next formula, namely formulae like $\mathcal{S}_{\trianglelefteq p}(\mathcal{P}_{\overline{\trianglelefteq p}}(X^I(\psi_k)))$. A number of preliminary definitions are needed in order to determine a compositional semantics for that case.

Definition 5.2.2 *Let $\mathcal{M}$ be a bidimensional Boucherie process, ψ_k a non-probabilistic formula as in (5.2.1), $p \in [0,1]$ a probability bound, $\trianglelefteq \in \{<, \leq, \geq, >\}$ a comparison relation, $I = [a,b] \subseteq \mathbb{R}_{\geq 0}$ a time interval and $s^j \in S_j$ a state of component $\mathcal{M}_j$. The following two formulae are defined:*

$$\mathcal{S}X_{low}(\psi_k, \trianglelefteq, p, I, s^j) \equiv \begin{cases} \bigvee_{s^k \in S_{k,\overline{R}}} \left(at_{s^k} \wedge \psi_k\right) & \textbf{if} \quad s^j \in S_{j,R} \\ \\ \bigvee_{s^k \in S_k} \left(\mathcal{P}_{\trianglelefteq \hat{p}}(X^{\hat{I}}\ \psi_k)\right) & \textbf{if} \quad s^j \in S_{j,\overline{R}} \end{cases} \tag{5.2.10}$$

$$\mathcal{S}X_{up}(\psi_k, \trianglelefteq, p, I, s^j) \equiv \begin{cases} \bigvee_{s^k \in S_{k,\overline{R}}} \left(at_{s^k} \wedge \neg\psi_k\right) & \textbf{if} \quad s^j \in S_{j,R} \\ \\ \bigvee_{s^k \in S_k} \left(\mathcal{P}_{\trianglelefteq \hat{p}}(X^{\hat{I}}\ \psi_k)\right) & \textbf{if} \quad s^j \in S_{j,\overline{R}} \end{cases} \tag{5.2.11}$$

where $(\hat{p}, \hat{I}) = h(\overline{p}, \psi_k, \mathcal{M}_k, (s^1, s^2), I)$ *and* at_{s^k} *represents the conjunction of atomic propositions which uniquely identifies the state* s^k, *namely* $at_{s^k} = \bigwedge_{a_k \in L_k(s^k)} a_k$.

In practice, $\mathcal{S}X_{low}(\psi_k, \trianglelefteq, p, I, s^j)$ and $\mathcal{S}X_{up}(\psi_k, \trianglelefteq, p, I, s^j)$ are template formulae which depend on a given state s^j of component $\mathcal{M}_j$. Their importance is beacuse they allow to characterise those state of component $\mathcal{M}_k$ which coupled with s^j result in a state (s^1, s^2) which validates a *single-component* time-bounded Next formula like $\mathcal{P}_{\trianglelefteq p}(X^I\ \psi_k)$. This result will be shown in the next lemma.

Lemma 5.2.1 *Let $\mathcal{M}$ be a bidimensional Boucherie process, ψ_k a non-probabilistic formula as in (5.2.1), $p \in [0,1]$ a probability bound, $\trianglelefteq \in \{<,\leq,\geq,>\}$ a comparison relation and $I=[a,b] \subseteq \mathbb{R}_{\geq 0}$ a time interval. Then the following holds:*

$$(s^1,s^2) \models \mathcal{P}_{\trianglelefteq p}(X^I(\psi_k)) \Longleftrightarrow \begin{cases} s^k \models_k \mathcal{S}X_{low}(\psi_k,\trianglelefteq,p,I,s^j) & \textbf{if } low(\trianglelefteq,p)\wedge \\ & \quad \left[[s^j \in S_{j,R}] \rightarrow [(e^{-E_j(s^j)a}-e^{-E_j(s^j)b}) \trianglelefteq p]\right] \\ & \\ s^k \models_k \mathcal{S}X_{up}(\psi_k,\trianglelefteq,p,I,s^j) & \textbf{if } up(\trianglelefteq,p) \end{cases} \tag{5.2.12}$$

Proof. See Lemma A.0.1 in Appendix A.

As a consequence of the definition of the template formulae $\mathcal{S}X_{low}(\psi_k,\trianglelefteq,p,I,s^j)$ and $\mathcal{S}X_{up}(\psi_k,\trianglelefteq,p,I,s^j)$, we observe that, those states of $\mathcal{M}_k$ for which there exists at least a state s^j of $\mathcal{M}_j$ by coupling with which they result in a state (s^1,s^2) satisfying a formula $\mathcal{P}_{\trianglelefteq p}(X^I\ \psi_k)$, are identified by means of the disjunction of the formulae $\mathcal{S}X_{low}(\psi_k,\trianglelefteq,p,I,s^j)$ ($\mathcal{S}X_{low}(\psi_k,\trianglelefteq,p,I,s^j)$), which is:

$$\bigvee_{s^j \in S_j} \mathcal{S}X_{low}(\psi_k,\trianglelefteq,p,I,s^j) \quad \text{or} \quad \bigvee_{s^j \in S_j} \mathcal{S}X_{up}(\psi_k,\trianglelefteq,p,I,s^j)$$

In the next definition two templates[2] formulae structurally similar to the ones described in Definition 5.2.2, are introduced. The principal difference is in that they do not depend on a given state s^j of $\mathcal{M}_j$.

Definition 5.2.3 *Let $\mathcal{M}$ be a bidimensional Boucherie process, ψ_k a non-probabilistic formula as in (5.2.1), $p \in [0,1]$ a probability bound, $\trianglelefteq \in \{<,\leq,\geq,>\}$ a comparison relation and $I=[a,b] \subseteq \mathbb{R}_{\geq 0}$ a time interval. The following two formulae are defined:*

[2]It should be noted that, in order to improve the readability, the templates $\mathcal{S}X_{low}(\psi_k,\trianglelefteq,p,I)$ and $\mathcal{S}X_{up}(\psi_k,\trianglelefteq,p,I)$ here defined as well as $\mathcal{S}X_{low}(\psi_k,\trianglelefteq,p,I)$ and $\mathcal{S}X_{up}(\psi_k,\trianglelefteq,p,I)$ in Definition 5.2.2, are expressed as disjunctions, even though the disjunctive connective $\vee$ is not part of the standard CSL syntax. However, since the set of connectives the CSL syntax is based on is adequate, the use of the disjunction is perfectly legal .

$$\begin{aligned}\mathcal{S}X_{low}(\psi_k,\unlhd,p,I)\equiv &\Big[\bigvee_{s^k\in S_{k,\overline{R}}}\big(at_{s^k}\wedge\big[\psi_k\vee\bigvee_{s^j\in S_{j,\overline{R}}}\mathcal{P}_{\unlhd\hat{p}}(X^{\hat{I}}(\psi_k))\big]\big)\Big]\vee\\ &\Big[\bigvee_{s^k\in S_{k,R}}\big[at_{s^k}\wedge\big(\mathcal{P}_{\unlhd p}(X^{I}(\psi_k))\big)\big]\Big]\end{aligned}\tag{5.2.13}$$

$$\begin{aligned}\mathcal{S}X_{up}(\psi_k,\unlhd,p,I)\equiv &\Big[\bigvee_{s^k\in S_{k,\overline{R}}}\big(at_{s^k}\wedge\big[\neg\psi_k\vee\bigvee_{s^j\in S_{j,\overline{R}}}\mathcal{P}_{\unlhd\hat{p}}(X^{\hat{I}}(\psi_k))\big]\big)\Big]\vee\\ &\vee\Big[\bigvee_{s^k\in S_{k,R}}\big[at_{s^k}\wedge\big(\mathcal{P}_{\unlhd p}(X^{I}(\psi_k))\big)\big]\Big]\end{aligned}\tag{5.2.14}$$

where $(\hat{p},\hat{I})=h(p,\psi_k,\mathcal{M}_k,(s^1,s^2),I)$ *and* at_{s^k} *represents the conjunction of atomic propositions which uniquely identifies the state* s^k*, namely* $at_{s^k}=\bigwedge_{a_k\in L_k(s^k)}a_k$*.*

The following proposition shows the semantic equivalences which relate the “state-independent” templates formulae $\mathcal{S}X_{low}(\psi_k,\unlhd,p,I)$ and $\mathcal{S}X_{up}(\psi_k,\unlhd,p,I)$ with their “state-dependent” counterparts $\mathcal{S}X_{low}(\psi_k,\unlhd,p,I,s^j)$ $\mathcal{S}X_{up}(\psi_k,\unlhd,p,I,s^j)$.

Proposition 5.2.1 *Let* $\mathcal{M}$ *be a bidimensional Boucherie process,* ψ_k *a non-probabilistic formula as in (5.2.1),* $p\in[0,1]$*,* $\unlhd\in\{<,\leq,\geq,>\}$ *and* $I=[a,b]\subseteq\mathbb{R}_{\geq 0}$ *a time interval. The following semantic equivalences holds:*

$$\mathcal{S}X_{low}(\psi_k,\unlhd,p,I)\equiv\bigvee_{s^j\in S_j}\mathcal{S}X_{low}(\psi_k,\unlhd,p,I,s^j)$$
$$\mathcal{S}X_{up}(\psi_k,\unlhd,p,I)\equiv\bigvee_{s^j\in S_j}\mathcal{S}X_{up}(\psi_k,\unlhd,p,I,s^j)$$

Proof. Starighforward.

The above result tells us that the states of $\mathcal{M}_k$ which *map* on a state (s^1,s^2) satisfying a formula $\mathcal{P}_{\unlhd p}(X^I\ \psi_k)$, are completely identified by means of the formula $\mathcal{S}X_{low}(\psi_k,\unlhd,p,I)$ ($\mathcal{S}X_{up}(\psi_k,\unlhd,p,I)$).

The formulae $\mathcal{S}X_{low}(\psi_k,\unlhd,p,I)$ and $\mathcal{S}X_{up}(\psi_k,\unlhd,p,I)$ are relevant in aiming for a decomposed semantics of *stedy-state* properties like $\mathcal{S}_{\unlhd p}(\mathcal{P}_{\unlhd p}(X^I(\psi_k)))$ (see Theorem 5.2.2). Next, an example showing a $\mathcal{S}X_{low}(\psi_k,\unlhd,p,I)$ formula in practice, is provided.

Example 5.2.2 *Referring to the Boucherie process of our running example, let us suppose we are interested in the states satisfying the probabilistic bounded Next formula* $\mathcal{P}_{\geq 0.3}(X^{[2,5]}\ read_1)$*. Since* $(\geq, 0.3)$ *represents a lower bound check, then we may consider the template formula* $\mathcal{SX}_{low}(read_1, \geq, 0.3, [2,5])$*, which we can readily compute from (5.2.13) and referring to the state-spaces shown in Figure 5.1 and Figure 5.2.*

$$
\begin{aligned}
\mathcal{SX}_{low}(read_1, \geq, 0.3, [2,5]) \ = \ & \\
& \Big[at_{s_{10}} \wedge (read_1 \vee \mathcal{P}_{\geq \hat{p}_a}(X^{\hat{I}_a}\ read_1))\Big] \vee \\
& \Big[at_{s_{11}} \wedge (read_1 \vee \mathcal{P}_{\geq \hat{p}_b}(X^{\hat{I}_b}\ read_1))\Big] \vee \\
& \Big[[at_{s_{12}} \wedge \mathcal{P}_{\geq 0.3}(X^{[2,5]}\ read_1)] \vee [at_{s_{13}} \wedge \mathcal{P}_{\geq 0.3}(X^{[2,5]}\ read_1)] \vee \\
& \quad [at_{s_{14}} \wedge \mathcal{P}_{\geq 0.3}(X^{[2,5]}\ read_1)] \vee [at_{s_{15}} \wedge \mathcal{P}_{\geq 0.3}(X^{[2,5]}\ read_1)]\Big]
\end{aligned}
$$

where

$$
\begin{aligned}
(\hat{p}_a, \hat{I}_a) &= h(0.3, read_1, \mathcal{M}_1, (s_{10}, s_{20}), [2,5]) \\
(\hat{p}_b, \hat{I}_b) &= h(0.3, read_1, \mathcal{M}_1, (s_{10}, s_{21}), [2,5])
\end{aligned}
$$

□

In essence, we are aiming to prove that checking that the probability for a bidimensional Boucherie process to satisfy, at *steady-state*, the bounded Next formula $(\mathcal{P}_{\trianglelefteq p}(X^I(\psi_k))$ is $\trianglelefteq p$, is equivalent to checking that the probability for the component process $\mathcal{M}_k$ to satisfy $\mathcal{SX}_{low}(\psi_k, \trianglelefteq, p, I)$, (or $\mathcal{SX}_{up}(\psi_k, \trianglelefteq, p, I)$), at *steady-state*, is $\trianglelefteq p'$, where p' is a derived probability whose value depends on p and other factors. The derivation of p' is the issue we are going to address next. For that reason, some relevant sets of states and some constants have to be characterised.

Definition 5.2.4 *Let* $t^k \in S_k$ *be a state of the component* $\mathcal{M}_k$ *of a bidimensional Boucherie process,* ψ_k *a non-probabilistic formula as in (5.2.1),* $p \in [0,1]$ *a probability bound,* $\trianglelefteq \in \{<, \leq, \geq, >\}$ *and* $I = [a,b] \subseteq \mathbb{R}_{\geq 0}$ *a time interval. The following two subsets of the state-space* S_j *are defined:*

$$
\begin{aligned}
Next_j^{low}(t^k, \psi_k, p, \trianglelefteq, I) &= Next_{j,\overline{R}}(t^k, \psi_k, p, \trianglelefteq, I) \cup Next_{j,R}^{low}(t^k, \psi_k, p, \trianglelefteq, I) \\
Next_j^{up}(t^k, \psi_k, p, \trianglelefteq, I) &= Next_{j,\overline{R}}(t^k, \psi_k, p, \trianglelefteq, I) \cup Next_{j,R}^{up}(t^k, \psi_k, p, \trianglelefteq, I)
\end{aligned}
$$

where

$$Next_{j,\overline{R}}(t^k,\psi_k,p,\trianglelefteq,I)=\{t^j\in S_{j,\overline{R}}:Sat_k(\mathcal{P}X_{\overline{R}_j}(\psi_k,t^k,t^j,p,\trianglelefteq,I)\neq\emptyset\}$$

and the template formula $\mathcal{P}X_{\overline{R}_j}(\psi_k,s^k,s^j,p,\trianglelefteq,I)$*, is defined as:*

$$\mathcal{P}X_{\overline{R}_j}(\psi_k,s^k,s^j,p,\trianglelefteq,I)\equiv at_{s^k}\wedge\mathcal{P}_{\trianglelefteq\hat{p}}(X^{\hat{I}}(\psi_k)) \tag{5.2.15}$$

with $(\hat{p},\hat{I})=h(p,\psi_k,\mathcal{M}_k,(s^1,s^2),I)$*, and where*

$$Next^{low}_{j,R}(t^k,\psi_k,p,\trianglelefteq,I)=\begin{cases}\bigcup_{t^j\in S_{j,R}:[e^{-E_j(t^j)a}-e^{-E_j(t^j)b}]\trianglelefteq p}\{t^j\} & \textbf{if}\quad t^k\models_k\psi_k\wedge\\ & \quad t^k\in S_{k,\overline{R}}\\ \\ \emptyset & \textbf{if}\quad (t^k\not\models_k\psi_k)\vee\\ & \quad (t^k\in S_{k,R})\end{cases}$$

$$Next^{up}_{j,R}(t^k,\psi_k,p,\trianglelefteq,I)=\begin{cases}\bigcup_{t^j\in S_{j,R}:[e^{-E_j(t^j)a}-e^{-E_j(t^j)b}]\trianglelefteq p}\{t^j\} & \textbf{if}\quad t^k\models_k\psi_k\wedge\\ & \quad t^k\in S_{k,\overline{R}}\\ \\ S_{j,R} & \textbf{if}\quad t^k\not\models_k\psi_k\wedge\\ & \quad t^k\in S_{k,\overline{R}}\\ \\ \emptyset & \textbf{if}\quad t^k\in S_{k,R}\end{cases}$$

When considering the states of $Sat(\mathcal{P}_{\trianglelefteq p}(X^I(\psi_k)))$ we can refer to a row-by-row (or column-by-column) partition of the state-space, namely $S=\bigcup_{s^1\in S_1}(s^1\times S_2)\setminus R_1R_2$ (or $S=\bigcup_{s^2\in S_2}(S_1\times s^2)\setminus R_1R_2$). In that sense, each row $(s^1,t^1,\ldots)$ or column $(s^2,t^2,\ldots)$, can have either none, one or many states satisfying $\mathcal{P}_{\trianglelefteq p}(X^I(\psi_k))$ and to each of them corresponds an equivalent derived condition, described by Theorem 5.2.1, which s^k is ensured to fulfil. We can describe that situation by saying that each state of a row $s^1\times S_j\setminus R_1R_2$ (or column $S_1\times s^2\setminus R_1R_2$), which satisfies $\mathcal{P}_{\trianglelefteq p}(X^I(\psi_k))$ *maps* on s^k. Moreover each row (column) s^k can be further partitioned according to the resource possession for $\mathcal{M}_j$: $(s^k\times S_{j,\overline{R}})$ ($(S_{j,\overline{R}}\times s^k)$) denotes the s^k row's (column's) states where $\mathcal{M}_j$ does not hold R while $(s^k \times S_{j,R})$ ($(S_{j,R} \times s^k)$) are those states where $\mathcal{M}_j$ holds R.

The sets of states $Next_j^{low}(t^k, \psi_k, p, \trianglelefteq, I)$ and $Next_j^{up}(t^k, \psi_k, p, \trianglelefteq, I)$, introduced in the above definition, allow for the *row-wise* (*column-wise*) partition of $Sat(\mathcal{P}_{\trianglelefteq p}(X^I(\psi_k)))$, as the following two lemmas will show.

Lemma 5.2.2 *Let $(s^1, s^2) \in S$ be a state of a bidimensional Boucherie process, ψ_k a non-probabilistic formula as in (5.2.1), $p \in [0,1]$ a probability bound, $\trianglelefteq \in \{<, \leq, \geq, >\}$, $I = [a,b] \subseteq \mathbb{R}_{\geq 0}$ a time interval. The state (s^1, s^2) satisfies the formula $Sat(\mathcal{P}_{\trianglelefteq p}(X^I(\psi_k)))$ if and only if its j-projection, s^j, is either in $Next_j^{low}(s^k, \psi_k, p, \trianglelefteq, I)$ if $low(\trianglelefteq, p)$, or in $Next_j^{up}(s^k, \psi_k, p, \trianglelefteq, I)$, if $up(\trianglelefteq, p)$.*

$$(s^1, s^2) \in Sat(\mathcal{P}_{\trianglelefteq p}(X^I(\psi_k))) \Longleftrightarrow \begin{cases} s^j \in Next_j^{low}(s^k, \psi_k, p, \trianglelefteq, I) & \textbf{if} \quad low(\trianglelefteq, p) \\ \\ s^j \in Next_j^{up}(s^k, \psi_k, p, \trianglelefteq, I) & \textbf{if} \quad up(\trianglelefteq, p) \end{cases}$$

Proof. See Lemma A.0.2 in Appendix A.

Relying on the above result, the following lemma, showing a *row-wise* (*column-wise*) partition for $Sat(\mathcal{P}_{\trianglelefteq p}(X^I(\psi_k)))$, can straightforwardly be proved.

Lemma 5.2.3 *Let $t^k \in S_k$ be a state of the component $\mathcal{M}_k$ of a bidimensional Boucherie process, ψ_k a non-probabilistic formula as in (5.2.1), $p \in [0,1]$ a probability bound, $\trianglelefteq \in \{<, \leq, \geq, >\}$, $I = [a,b] \subseteq \mathbb{R}_{\geq 0}$ a time interval. The satisfiability set for the formula $\mathcal{P}_{\trianglelefteq p}(X^I(\psi_k))$ is partitionable in the following way:*

$$Sat(\mathcal{P}_{\trianglelefteq p}(X^I(\psi_k))) = \begin{cases} \bigcup_{t^k \in Sat_k(\mathcal{S}X_{low}(\psi_k, \trianglelefteq, p, I))} [t^k \times Next_j^{low}(t^k, \psi_k, p, \trianglelefteq, I)] \\ \hfill \textbf{if } low(\trianglelefteq, p) \\ \\ \bigcup_{t^k \in Sat_k(\mathcal{S}X_{up}(\psi_k, \trianglelefteq, p, I))} [t^k \times Next_j^{up}(t^k, \psi_k, p, \trianglelefteq, I)] \\ \hfill \textbf{if } up(\trianglelefteq, p) \end{cases}$$

Proof. See Lemma A.0.3 in Appendix A.

In aiming for a compositional semantics for $\mathcal{S}_{\trianglelefteq p}(\mathcal{P}_{\overline{\trianglelefteq}\overline{p}}(X^I(\psi_k)))$, it is relevant to provide a characterisation of those states of a row (column) s^k which satisfy $\mathcal{P}_{\overline{\trianglelefteq}\overline{p}}(X^I(\psi_k))$. The sets $Next_j^{low}(t^k,\psi_k,\overline{p},\overline{\trianglelefteq})$ and $Next_j^{up}(t^k,\psi_k,\overline{p},\overline{\trianglelefteq})$, introduced in Definition 5.2.4, provide such a characterisation and by means of them the equivalent bound p' for the *steady-state* probability of the $\mathcal{P}_{\overline{\trianglelefteq}\overline{p}}(X^I(\psi_k))$ states, will be derived. The notation for the *steady-state* probability of the states of the row (column) $Next_j^{low}(t^k,\psi_k,\overline{p},\overline{\trianglelefteq})$ and $Next_j^{up}(t^k,\psi_k,\overline{p},\overline{\trianglelefteq})$ is introduced in the following remark.

Remark 5.2.1 *Let $t^k \in S_k$ be a state of the component $\mathcal{M}_k$ of a bidimensional Boucherie process, ψ_k a non-probabilistic formula as in (5.2.1), $\overline{p} \in [0,1]$ a probability bound, $\overline{\trianglelefteq} \in \{<,\leq,\geq,>\}$, $I=[a,b] \subseteq \mathbb{R}_{\geq 0}$ a time interval. The steady-state probability of the sets $Next_j^{low}(t^k,\psi_k,\overline{p},\overline{\trianglelefteq})$ and $Next_j^{up}(t^k,\psi_k,\overline{p},\overline{\trianglelefteq})$ is denoted as:*

$$\pi_j(Next_j^{low}(t^k,\psi_k,\overline{p},\overline{\trianglelefteq})) = \sum_{t^j \in Next_j^{low}(t^k,\psi_k,\overline{p},\overline{\trianglelefteq})} \pi_j(t^j)$$

while

$$\pi_j(Next_j^{up}(t^k,\psi_k,\overline{p},\overline{\trianglelefteq})) = \sum_{t^j \in Next_j^{up}(t^k,\psi_k,\overline{p},\overline{\trianglelefteq})} \pi_j(t^j)$$

It is relevant to be able to distinguish which, amongst the states of $\mathcal{M}_k$, is the one whose associated set $Next_j^{low}(t^k,\psi_k,\overline{p},\overline{\trianglelefteq})$ (or $Next_j^{up}(t^k,\psi_k,\overline{p},\overline{\trianglelefteq})$) has the highest *steady-state* probability. The following remark introduces the notation adopted for such a state.

Remark 5.2.2 *Let ψ_k be a non-probabilistic formula as in (5.2.1), $\overline{p} \in [0,1]$ a probability bound, $\overline{\trianglelefteq} \in \{<,\leq,\geq,>\}$, $I=[a,b] \subseteq \mathbb{R}_{\geq 0}$ a time interval, we denote by $t^k_{Xmax} \in S_k$, the state of component $\mathcal{M}_k$ which maximises the steady-state probability of its associated set $Next_j^{low}(t^k_{Xmax},\psi_k,\overline{p},\overline{\trianglelefteq})$ ($Next_j^{up}(t^k_{Xmax},\psi_k,\overline{p},\overline{\trianglelefteq})$).*

$$\begin{aligned} &t^k_{Xmax} \in S_k : \forall t^k \in S_k, t^k \neq t^k_{Xmax} \Rightarrow \\ &\qquad \pi_j(Next_j^{low}(t^k_{Xmax},\psi_k,\overline{p},\overline{\trianglelefteq},I)) \geq \pi_j(Next_j^{low}(t^k,\psi_k,\overline{p},\overline{\trianglelefteq},I)) \end{aligned}$$

The following definition introduces the notation adopted to indicate the *steady-state* probability of the set $Next_j^{low}(t^k_{Xmax},\psi_k,\overline{p},\overline{\trianglelefteq})$ ($Next_j^{up}(t^k_{Xmax},\psi_k,\overline{p},\overline{\trianglelefteq})$), which is the maximum value amongst the $\pi_j(Next_j^{low}(t^k,\psi_k,\overline{p},\overline{\trianglelefteq}))$ of every state $t^k \in S_k$.

Definition 5.2.5 *Let ψ_k be a non-probabilistic formula as in (5.2.1), $\overline{p} \in [0,1]$ a probability bound, $\overline{\trianglelefteq} \in \{<,\leq,\geq,>\}$ and $I = [a,b] \subseteq \mathbb{R}_{\geq 0}$ a time interval. The following constant is defined:*

$$\pi_j^{max}(\psi_k,\overline{p},\overline{\trianglelefteq},I) = \begin{cases} \pi_j(Next_j^{low}(t^k_{Xmax},\psi_k,\overline{p},\overline{\trianglelefteq})) & \textbf{iff} \quad low(\overline{p},\overline{\trianglelefteq}) \\ \pi_j(Next_j^{up}(t^k_{Xmax},\psi_k,\overline{p},\overline{\trianglelefteq})) & \textbf{iff} \quad up(\overline{p},\overline{\trianglelefteq}) \end{cases}$$

In the next definition two important constants are introduced, namely $C^{low}(\psi_k,\overline{p},\overline{\trianglelefteq},I)$ and $C^{up}(\psi_k,\overline{p},\overline{\trianglelefteq},I)$. They represent the probability for component $\mathcal{M}_k$ to be in a state s^k, at *steady-state*, weighted with the *deviation* of the *steady-state* probability $\pi_j(Next_j^{low}(t^k,\psi_k,\overline{p},\overline{\trianglelefteq},I))$ $(\pi_j(Next_j^{low}(t^k,\psi_k,\overline{p},\overline{\trianglelefteq},I)))$, from the maximum $\pi_j^{max}(\psi_k,\overline{p},\overline{\trianglelefteq},I)$.

Definition 5.2.6 *Let ψ_k be a non-probabilistic formula as in (5.2.1), $\overline{p} \in [0,1]$ a probability bound, $\overline{\trianglelefteq} \in \{<,\leq,\geq,>\}$ and $I = [a,b] \subseteq \mathbb{R}_{\geq 0}$ a time interval. The following constants are defined:*

$$C^{low}(\psi_k,\overline{p},\overline{\trianglelefteq},I) = \sum_{\substack{t^k \in Sat_k(\mathcal{S}X_{low}(\psi_k,\overline{\trianglelefteq},\overline{p},I)) \\ t^k \neq t^k_{Xmax}}} \pi_k(t^k)[\pi_j^{max}(\psi_k,\overline{p},\overline{\trianglelefteq},I) - \pi_j(Next_j^{low}(t^k,\psi_k,\overline{p},\overline{\trianglelefteq},I))]$$

$$C^{up}(\psi_k,\overline{p},\overline{\trianglelefteq},I) = \sum_{\substack{t^k \in Sat_k(\mathcal{S}X_{up}(\psi_k,\overline{\trianglelefteq},\overline{p},I)) \\ t^k \neq t^k_{Xmax}}} \pi_k(t^k)[\pi_j^{max}(\psi_k,\overline{p},\overline{\trianglelefteq},I) - \pi_j(Next_j^{up}(t^k,\psi_k,\overline{p},\overline{\trianglelefteq},I))]$$

Finally, the following theorem determines the compositional semantics for *steady-state* bounded Next formulae.

Theorem 5.2.2 (*Steady-state* **Bounded** *single-component* **Next)** *Let $\mathcal{M}$ be a bidimensional Boucherie process, ψ_k a non-probabilistic single-component formula as in (5.2.1), $p,\overline{p} \in [0,1]$ two probability bounds, $\trianglelefteq,\overline{\trianglelefteq} \in \{<\leq,\geq,>\}$ two comparison relations and $I = [a,b] \subseteq \mathbb{R}_{\geq 0}$ a time interval. The following equivalences hold:*

$$\models \mathcal{S}_{\trianglelefteq p}(\mathcal{P}_{\overline{\trianglelefteq}\overline{p}}(X^I(\psi_k)) \Longleftrightarrow \begin{cases} \models_k \mathcal{S}_{\trianglelefteq p'_{low}}(\mathcal{S}X_{low}(\psi_k,\overline{\trianglelefteq},\overline{p},I)) & \textbf{if} \;\; low(\overline{p},\overline{\trianglelefteq}) \\ \\ \models_k \mathcal{S}_{\trianglelefteq p'_{up}}(\mathcal{S}X_{up}(\psi_k,\overline{\trianglelefteq},\overline{p},I)) & \textbf{if} \;\; up(\overline{p},\overline{\trianglelefteq}) \end{cases} \tag{5.2.16}$$

where $p'_{low} = \frac{p+GC^{low}(\psi_k,\overline{p},\overline{\trianglelefteq},I)}{G\pi_j^{max}(\psi_k,\overline{p},\overline{\trianglelefteq},I)}$ *and* $p'_{up} = \frac{p+GC^{up}(\psi_k,\overline{p},\overline{\trianglelefteq},I)}{G\pi_j^{max}(\psi_k,\overline{p},\overline{\trianglelefteq},I)}$ *while G is the normalising constant for the product-form solution, of* $\mathcal{M}$.

Proof. We aim to show the validity of the two implications ($\Rightarrow$) and ($\Leftarrow$) of (5.2.16). For brevity we consider here only the first case, i.e. we assume $low(\overline{p},\overline{\trianglelefteq})$ (the proof for $up(\overline{p},\overline{\trianglelefteq})$, is similar). Furthermore we focus on the case $k=1$ (hence $j=2$), which means we consider here a formula ψ_1 which refers to component $\mathcal{M}_1$.

($\Rightarrow$) If $\models \mathcal{S}_{\trianglelefteq p}(\mathcal{P}_{\overline{\trianglelefteq}\overline{p}}(X^I(\psi_1))$ then from the CSL semantics

$$\Big[\sum_{(t^1,t^2)\in Sat(\mathcal{P}_{\overline{\trianglelefteq}\overline{p}}(X^I\psi_1))} \pi(t^1,t^2)\Big] \trianglelefteq p$$

which from the product-form solution, results in

$$\Big[\sum_{(t^1,t^2)\in Sat(\mathcal{P}_{\overline{\trianglelefteq}\overline{p}}(X^I\psi_1))} G\cdot\pi_1(t^1)\pi_2(t^2)\Big] \trianglelefteq p \tag{5.2.17}$$

By applying the *row-wise* partition of $Sat(\mathcal{P}_{\overline{\trianglelefteq}\overline{p}}(X^I\psi_1)$ (see Lemma 5.2.3) to the sum in (5.2.17) we obtain:

$$G\Big[\sum_{t^1\in Sat_1(\mathcal{S}X_{low}(\psi_1,\overline{\trianglelefteq},\overline{p},I))} \pi_1(t^1) \sum_{t^2\in Next_2^{low}(t^1,\psi_1,\overline{p},\overline{\trianglelefteq},I)} \pi_2(t^2)\Big] \trianglelefteq p$$

From Remark 5.2.1 also $\sum_{t^2\in Next_2^{low}(t^1,\psi_1,\overline{p},\overline{\trianglelefteq},I)} \pi_2(t^2) = \pi_2(Next_2^{low}(t^1,\psi_1,\overline{p},\overline{\trianglelefteq},I))$. Hence:

$$G\cdot\Big[\sum_{t^1\in Sat_1(\mathcal{S}X_{low}(\psi_1,\overline{\trianglelefteq},\overline{p},I))} \pi_1(t^1)\cdot\pi_2(Next_2^{low}(t^1,\psi_1,\overline{p},\overline{\trianglelefteq},I))\Big] \trianglelefteq p$$

from which, straightforwardly,

$$G\cdot\pi_2(Next_2^{low}(t^1_{Xmax},\psi_1,\overline{p},\overline{\trianglelefteq},I)) \sum_{t^1\in Sat_1(\mathcal{S}X_{low}(\psi_1,\overline{\trianglelefteq},\overline{p},I))} \pi_1(t^1) \trianglelefteq (p+G\cdot C^{low}(t^1,\psi_1,\overline{p},\overline{\trianglelefteq},I)))$$

Thus,

$$\Big[\sum_{t^1\in Sat_1(\mathcal{S}X_{low}(\psi_1,\overline{\trianglelefteq},\overline{p},I))} \pi_1(t^1)\Big] \trianglelefteq \frac{(p+G\cdot C^{low}(t^1,\psi_1,\overline{p},\overline{\trianglelefteq},I))}{G\cdot\pi_2(Next_2^{low}(t^1_{Xmax},\psi_1,\overline{p},\overline{\trianglelefteq},I))}$$

which proves

$$\models_1 \mathcal{S}_{\unlhd \frac{(p+G\cdot C^{low}(t^1,\psi_1,\overline{p},\overline{\unlhd},I))}{G\cdot\pi_2(Next_2^{low}(t^1_{Xmax},\psi_1,\overline{p},\overline{\unlhd},I))}}(\mathcal{S}X_{low}(\psi_1,\overline{\unlhd},\overline{p},I))$$

($\Leftarrow$) By reversing ($\Rightarrow$).

$\square$

Example 5.2.3 *Still referring to our running example (see Figure 5.1 and Figure 5.2), let us suppose we are interested in checking whether the probability, in the long-run, for those states which satisfy the bounded Next $\mathcal{P}_{\geq 0.3}(X^{[2,5]}\ read_1)$, is ≥ 0.5. In terms of the syntax in (5.2.1), this is represented by the formula*

$$\mathcal{S}_{\geq 0.5}(\mathcal{P}_{\geq 0.3}(X^{[2,5]}\ read_1))$$

According to Theorem 5.2.2, we expect

$$\models \mathcal{S}_{\geq 0.5}(\mathcal{P}_{\geq 0.3}(X^{[2,5]}\ read_1)) \Longleftrightarrow \models_1 \mathcal{S}_{\unlhd p'_{low}}(\mathcal{S}X_{low}(read_1,\geq,0.3,[2,5]))$$

where $p'_{low} = \frac{0.5+GC^{low}(read_1,0.3,\geq,[2,5])}{G\pi_2^{max}(\psi_1,0.3,\geq,[2,5])}$*, which is to say:*

$$\sum_{(t^1,t^2)\models\mathcal{P}_{\geq 0.3}(X^{[2,5]}\ read_1)} G\cdot\pi_1(t^1)\pi_2(t^2)] \geq 0.5 \quad\Longleftrightarrow\quad \sum_{t^1\models_1\mathcal{S}X_{low}(read_1,\geq,0.3,[2,5])} \pi_1(t^1) \geq p'_{low} \qquad (5.2.18)$$

Let us verify whether this is the case. From Figure 2.8, we know that (s_{12},s_{20}) is the only state where component $\mathcal{M}_1$ is reading the shared register, hence the only potential candidate for satisfying the Next formula $\mathcal{P}_{\geq 0.3}(X^{[2,5]}\ read_1)$ is its unique predecessor (s_{10},s_{20}). Let us assume that, in fact, $(s_{10},s_{20}) \models \mathcal{P}_{\geq 0.3}(X^{[2,5]}\ read_1)$. In this case the left sum in the above equivalence is

$$\sum_{(t^1,t^2)\models\mathcal{P}_{\geq 0.3}(X^{[2,5]}\ read_1)} G\cdot\pi_1(t^1)\pi_2(t^2) = G\cdot\pi_1(s_{10})\pi_2(s_{20})$$

In Example 5.2.2, the formula $\mathcal{S}X_{low}(read_1,\geq,0.3,[2,5])$ has been shown. From Figure 2.7, we know that $read_1$ is true only in state s_{12}, which has a unique predecessor, namely s_{10}. Furthermore, since we are assuming $(s_{10},s_{20}) \models \mathcal{P}_{\geq 0.3}(X^{[2,5]}\ read_1)$, then from Theorem 5.2.1 also, as it has been shown in Example 5.2.1,

$$s_{10} \models_1 \mathcal{P}_{\geq \frac{0.3\cdot(2r_1+r_5)}{r_1+r_5}}(X^{[\frac{2\cdot(2r_1+r_5)}{r_1+r_5},\frac{5\cdot(2r_1+r_5)}{r_1+r_5}]}\ read_1)$$

From this it is straightforward to show that there is only one state satisfying $\mathcal{S}X_{low}(read_1, \geq, 0.3, [2,5])$, namely $Sat_1(\mathcal{S}X_{low}(read_1, \geq, 0.3, [2,5])) = \{s_{10}\}$. Hence, s_{10}, is also the maximum (i.e. s_{10} is the state which maximises the steady-state probability of the associated set $Next_2^{low}(s_{10}, read_1, 0.3, \geq)$), which means

$$\pi_2^{max}(read_1, 0.3, \geq, [2,5]) = \pi_2(Next_2^{low}(s_{10}, read_1, 0.3, \geq))$$

As a result, the constant $C^{low}(read_1, 0.3, \geq, [2,5])$ (see Definition 5.2.6), is null. In order to compute the value of $\pi_2^{max}(\psi_1, 0.3, \geq, [2,5])$, we need to determine the set $Next_2^{low}(s_{10}, read_1, 0.3, \geq)$, which from Definition 5.2.4 we know consists of two partitions, namely $Next_{2,\overline{R}}^{low}(s_{10}, read_1, 0.3, \geq)$ and $Next_{2,R}^{low}(s_{10}, read_1)$. As $s_{10} \not\models_1 read_1$ then $Next_{2,R}^{low}(s_{10}, read_1) = \emptyset$. Moreover, as a consequence of the assumptions we made, it is easy to show that $Next_{2,\overline{R}}^{low}(s_{10}, read_1, 0.3, \geq) = \{s_{20}\}$ (i.e. s_{20} is the only state which coupled with s_{10}, results in a state, (s_{10}, s_{20}), which satisfies $\mathcal{P}_{\geq 0.3}(X^{[2,5]}\ read_1)$). Then clearly $\pi_2^{max}(read_1, 0.3, \geq, [2,5]) = \pi_2(s_{20})$. Thus the equivalent probability value p'_{low} is equal to:

$$p'_{low} = \frac{0.5 + GC^{low}(read_1, 0.3, \geq, [2,5])}{G\pi_2^{max}(\psi_1, 0.3, \geq, [2,5])} = \frac{0.5}{G \cdot \pi_2(s_{20})}$$

Since s_{10} is the only state of $\mathcal{M}_1$ satisfying $\mathcal{S}X_{low}(read_1, \geq, 0.3, [2,5])$ then the right-hand side sum in equivalence (5.2.18) is

$$\sum_{t^1 \models_1 \mathcal{S}X_{low}(read_1, \geq, 0.3, [2,5])} \pi_1(t^1) = \pi_1(s_{10})$$

But this proves that the equivalence (5.2.18) actually holds, in fact:

$$G \cdot \pi_1(s_{10})\pi_2(s_{20}) \geq 0.5 \quad \Longleftrightarrow \quad \pi_1(s_{10}) \geq \frac{0.5}{G \cdot \pi_2(s_{20})}$$

□

5.3 Compositional model-checking of *single-component* Next

The properties proved in the previous section of this chapter call for the formal characterisation of methods for a decomposed verification of *single-component* bounded

Next formulae (i.e. $\mathcal{P}_{\overline{\trianglelefteq}\overline{p}}(X^I(\psi_k)))$ as well as *steady-state* bounded Next formulae (i.e. $\mathcal{S}_{\trianglelefteq p}(\mathcal{P}_{\overline{\trianglelefteq}\overline{p}}(X^I(\psi_k))))$. Algorithm 5.3.1 describes a procedure for decomposed checking of a probabilistic bounded Next. Algorithm 5.3.3, instead, shows such a procedure for *steady-state* bounded Next formulae.

Algorithm 5.3.1 ($\mathcal{P}_{\overline{\trianglelefteq}\overline{p}}(X^I(\psi_k))$) *Let (s^1,s^2) be a state of a bidimensional Boucherie process $\mathcal{M}$ with components $\mathcal{M}_1$ and $\mathcal{M}_2$. Furthermore let ψ_k be a single-component formula, $\overline{p} \in [0,1]$ a probability bound, $\overline{\trianglelefteq} \in \{<,\leq,\geq,>\}$ and $I = [a,b] \subseteq \mathbb{R}_{\geq 0}$ a time interval. The following procedure can be used for checking the truth of the formula $\mathcal{P}_{\overline{\trianglelefteq}\overline{p}}(X^I(\psi_k))$ with respect to state (s^1,s^2).*

<u>**Algorithm**</u> *$((s^1,s^2) \models \mathcal{P}_{\overline{\trianglelefteq}\hat{p}}(X^{\hat{I}}(\psi_k)))$.*

IF $(s^1,s^2) \in R_j$ **THEN**
- $(\hat{p},\hat{I}) = h(p,\psi_k,\mathcal{M}_k,s^1,s^2,I)$;
- *compute* $Sat_k(\mathcal{P}_{\overline{\trianglelefteq}\hat{p}}(X^{\hat{I}}(\psi_k))$;
- **IF** $s^k \in Sat_k(\mathcal{P}_{\overline{\trianglelefteq}\hat{p}}(X^{\hat{I}}(\psi_k))$ **THEN** *return YES;* **ELSE** *return NO;*

ELSE
  IF $low(\overline{\trianglelefteq},\overline{p})$ **THEN**
    IF $[e^{-a\cdot E_j(s_j)} - e^{-b\cdot E_j(s_j)}]\ \overline{\not\trianglelefteq}\overline{p}$ **THEN** *return NO;*
      ELSE
        - *compute* $Sat_k(\psi_k)$;
        - **IF** $s^k \in Sat_k(\psi_k)$ **THEN** *return YES* **ELSE** *return NO;*
  ELSE
    IF $[e^{-a\cdot E_j(s_j)} - e^{-b\cdot E_j(s_j)}]\overline{\trianglelefteq}\overline{p}$ **THEN** *return YES;*
      ELSE
        - *compute* $Sat_k(\psi_k)$;
        - **IF** $s^k \in Sat_k(\psi_k)$ **THEN** *return NO;* **ELSE** *return YES;*

□

The procedure for decomposed checking of *steady-state* bounded Next formulae basically requires the instantiation of some template formulae and the application of the model-checking algorithm to those formulae with respect to component $\mathcal{M}_k$.

The following algorithm characterises the procedure for determining which amongst the states of a *row* (*column*) s^k satisfy the formula $\mathcal{P}_{\overline{\trianglelefteq}\overline{p}}(X^I(\psi_k))$. The compositional checking of $\mathcal{S}_{\trianglelefteq p}(\mathcal{P}_{\overline{\trianglelefteq}\overline{p}}(X^I(\psi_k)))$ relies on it.

Algorithm 5.3.2 ($Next_j^{low}(s^k,\psi_k,\overline{p},\overline{\trianglelefteq},I)$**,** $Next_j^{up}(s^k,\psi_k,\overline{p},\overline{\trianglelefteq},I)$) *Let $\mathcal{M}$ be a bidimensional Boucherie process with components $\mathcal{M}_1$ and $\mathcal{M}_2$ and s^k a state of component $\mathcal{M}_k$. Furthermore let ψ_k be a single-component formula, $\overline{p}\in[0,1]$ a probability bound, $\overline{\trianglelefteq}\in\{<,\leq,\geq,>\}$ and $I=[a,b]\subseteq\mathbb{R}_{\geq 0}$ a time interval. The sets $Next_{j,\overline{R}}(s^k,\psi_k,\overline{p},\overline{\trianglelefteq},I)$, $Next_{j,\overline{R}}^{low}(s^k,\psi_k,\overline{p},\overline{\trianglelefteq},I)$ and $Next_{j,\overline{R}}^{up}(s^k,\psi_k,\overline{p},\overline{\trianglelefteq},I)$ can be determined by means of the following procedures:*

<u>**Algorithm A**</u> ($Next_{j,\overline{R}}(s^k,\psi_k,\overline{p},\overline{\trianglelefteq})$).

1. $Next_{j,\overline{R}}(s^k,\psi_k,\overline{p},\overline{\trianglelefteq})=\emptyset$

2 FOR *every* $s^j\in S_{j,\overline{R}}$ **DO**

- *instantiate the template formula*

$$\mathcal{P}X_{\overline{R}_j}(\psi_k,s^k,s^j,\overline{p},\overline{\trianglelefteq})\equiv at_{s^k}\wedge\mathcal{P}_{\overline{\trianglelefteq}\hat{p}}(X^{\hat{I}}(\psi_k))$$

where $(\hat{p},\hat{I})=h(\psi_k,\mathcal{M}_k,s^k,s^j,I)$.

- *check for emptyness the set* $Sat_k(\mathcal{P}X_{\overline{R}_j}(\psi_k,s^k,s^j,\overline{p},\overline{\trianglelefteq}))$. *Then*

$$Next_{j,\overline{R}}(s^k,\psi_k,\overline{p},\overline{\trianglelefteq})=\begin{cases} Next_{j,\overline{R}}(s^k,\psi_k,\overline{p},\overline{\trianglelefteq}) & \textbf{if } Sat_k(\mathcal{P}X_{\overline{R}_j}(\psi_k,s^k,s^j,\overline{p},\overline{\trianglelefteq}))=\emptyset\\ Next_{j,\overline{R}}(s^k,\psi_k,\overline{p},\overline{\trianglelefteq})\cup\{s^k\} & \textbf{if } Sat_k(\mathcal{P}X_{\overline{R}_j}(\psi_k,s^k,s^j,\overline{p},\overline{\trianglelefteq}))\neq\emptyset\end{cases}$$

□

<u>**Algorithm B**</u> ($Next_{j,\overline{R}}^{low}(s^k,\psi_k,\overline{p},\overline{\trianglelefteq},I)$).

1. $Next_{j,\overline{R}}^{low}(s^k,\psi_k,\overline{p},\overline{\trianglelefteq},I)=\emptyset$

2. IF $s^k \in S_{k,R}$ **THEN** *return;*

ELSE

- *compute* $Sat_k(\psi_k)$*;*
- **IF** $s^k \notin Sat_k(\psi_k)$ **THEN** *return;*

ELSE

FOR *every* $s^j \in S_{j,R}$ **DO**

$$Next^{low}_{j,\overline{R}}(s^k,\psi_k,\overline{p},\overline{\trianglelefteq},I) = \begin{cases} Next^{low}_{j,\overline{R}}(s^k,\psi_k,\overline{p},\overline{\trianglelefteq},I) & \textbf{if } [e^{-a\cdot E_j(s_j)} - e^{-b\cdot E_j(s_j)}] \ \overline{\not\trianglelefteq p} \\ Next^{low}_{j,\overline{R}}(s^k,\psi_k,\overline{p},\overline{\trianglelefteq},I) \cup \{s^j\} & \textbf{if } [e^{-a\cdot E_j(s_j)} - e^{-b\cdot E_j(s_j)}] \overline{\trianglelefteq p} \end{cases}$$

□

<u>**Algorithm C**</u> **(**$Next^{up}_{j,\overline{R}}(s^k,\psi_k,\overline{p},\overline{\trianglelefteq},I)$**).**

1. $Next^{up}_{j,\overline{R}}(s^k,\psi_k,\overline{p},\overline{\trianglelefteq},I) = \emptyset$

2. IF $s^k \in S_{k,R}$ **THEN** *return;*

ELSE

- *compute* $Sat_k(\psi_k)$*;*
- **IF** $s^k \notin Sat_k(\psi_k)$ **THEN** $Next^{up}_{j,\overline{R}}(s^k,\psi_k,\overline{p},\overline{\trianglelefteq},I) = S_{j,R}$*;*

ELSE

FOR *every* $s^j \in S_{j,R}$ **DO**

$$Next^{low}_{j,\overline{R}}(s^k,\psi_k,\overline{p},\overline{\trianglelefteq},I) = \begin{cases} Next^{low}_{j,\overline{R}}(s^k,\psi_k,\overline{p},\overline{\trianglelefteq},I) & \textbf{if } [e^{-a\cdot E_j(s_j)} - e^{-b\cdot E_j(s_j)}] \ \overline{\not\trianglelefteq p} \\ Next^{low}_{j,\overline{R}}(s^k,\psi_k,\overline{p},\overline{\trianglelefteq},I) \cup \{s^j\} & \textbf{if } [e^{-a\cdot E_j(s_j)} - e^{-b\cdot E_j(s_j)}] \overline{\triangleleft p} \end{cases}$$

□

Relying on the above algorithms, the method for a decomposed verification of the *steady-state* probability of *single-component* bounded Next formulae is defined in the following manner.

Algorithm 5.3.3 (Compositional model-checking of $\mathcal{S}_{\trianglelefteq p}(\mathcal{P}_{\overline{\trianglelefteq}\overline{p}}(X^I(\psi_k))))$ *Let* (s^1, s^2) *be a state of a bidimensional Boucherie process* $\mathcal{M}$ *with components* $\mathcal{M}_1$ *and* $\mathcal{M}_2$*,* ψ_k *a single-component formula,* $p, \overline{p} \in [0,1]$ *two probability bounds and* $\trianglelefteq, \overline{\trianglelefteq} \in \{<, \leq, \geq, >\}$*. The following algorithm can be applied for decomposed checking of the formula*

$$\mathcal{S}_{\trianglelefteq p}(\mathcal{P}_{\overline{\trianglelefteq}\overline{p}}(X^I(\psi_k)))$$

with respect to $\mathcal{M}$*: it returns YES if* $\mathcal{S}_{\trianglelefteq p}(\mathcal{P}_{\overline{\trianglelefteq}\overline{p}}(X^I(\psi_k)))$ *is satisfied in* $\mathcal{M}$ *or NO if it is not.*

1. *The pair* $(\overline{\trianglelefteq}, \overline{p})$ *is considered and the template formula* $\mathcal{S}X$ *is instantiated by means of the parameters* $\psi_k, \trianglelefteq, \overline{p}$ *and* I*.*

$$\mathcal{S}X = \begin{cases} \mathcal{S}X_{low}(\psi_k, \overline{p}, \trianglelefteq, I) & \textbf{if} \quad low(\overline{\trianglelefteq}, \overline{p}) \\ \mathcal{S}X_{up}(\psi_k, \overline{p}, \trianglelefteq, I) & \textbf{if} \quad up(\overline{\trianglelefteq}, \overline{p}) \end{cases}$$

2. $Sat_k(\mathcal{S}X)$ *is computed by application of the CSL model-checking algorithm with respect to component* $\mathcal{M}_k$*.*

3. FOR *every* $s^k \in Sat_k(\mathcal{S}X)$ **DO**

- *compute* $Next_j(s^k) = Next_{j,\overline{R}}(s^k) \cup Next_{j,R}(s^k)$ *where*

$$Next_{j,\overline{R}}(s^k) = Next_{j,\overline{R}}(s^k, \psi_k, \trianglelefteq, \overline{p}, I)$$

and

$$Next_{j,R}(s^k) = \begin{cases} Next^{low}_{j,R}(s^k, \psi_k, \overline{p}, \trianglelefteq, I) & \textbf{if} \quad low(\overline{\trianglelefteq}, \overline{p}) \\ Next^{up}_{j,R}(s^k, \psi_k, \overline{p}, \trianglelefteq, I) & \textbf{if} \quad up(\overline{\trianglelefteq}, \overline{p}) \end{cases}$$

are computed by means of the procedure described in Algorithm 5.3.2.

4.

- *Determine the state* $s^k_{MAX} \in Sat_k(\mathcal{S}X)$ *such that* $\pi_j(Next_j(s^k_{MAX}))$ *is maximum.*
- *Let* $\pi_j^{MAX} = \pi_j(Next_j(s^k_{MAX}))$*.*

5. Compute the value C given by:

$$C = \sum_{\substack{t^k \in Sat_k(\mathcal{S}X) \\ t^k \neq s^k_{MAX}}} \pi_k(t^k)[\pi_j^{MAX} - \pi_j(Next_j(t^k))];$$

6. Compute the equivalent probability bound p'

$$p' = \frac{p + G \cdot C}{G \cdot \pi_j^{MAX}};$$

7. IF $\pi_k(Sat_k(\mathcal{S}X)) \trianglelefteq p'$ **THEN** return YES; **ELSE** return NO.

□

5.4 Compositional model-checking of *general* Next

So far, in the chapter, we have considered only *single-component* non-probabilistic formulae (i.e. ψ_k) as possible arguments of a bounded Next connective. In this section, we investigate the existence of a compositional method for checking formulae given by the application of the bounded Next operator to a non-probabilistic *general* argument (i.e ψ_{12}). The new version of the syntax for *general* state formulae (i.e. ϕ_{12}) (an extension of the one introduced in Section 4.3.2) is characterised in the following manner[3]:

$$\begin{aligned}
&\phi_{12} ::= \ \phi_1 \wedge \phi_2 \ | \ \phi_2 \wedge \phi_1 \ | \ \phi_k \wedge \phi_{12} \ | \ \phi_{12} \wedge \phi_k \ | \ \neg\phi_{12} \ | \ \phi_{12} \wedge \phi_{12} \ | \ \xi_{12} \ | \ \varphi_{12} \\
&\psi_{12} ::= \psi_1 \wedge \psi_2 \ | \ \psi_2 \wedge \psi_1 \ | \ \psi_k \wedge \psi_{12} \ | \ \psi_{12} \wedge \psi_k \ | \ \psi_{12} \wedge \psi_{12} \ | \ \neg\psi_{12} \\
&\xi_{12} ::= \mathcal{S}_{\trianglelefteq p}(\psi_{12}) \\
&\varphi_{12} ::= \mathcal{P}_{\trianglelefteq p}(X^I \ \psi_{12})
\end{aligned} \tag{5.4.1}$$

where ϕ_1 and ϕ_2 are as in (5.2.1).

---

[3]As one can notice from (5.4.1), the application of the *steady-state* connective to *general* bounded Next formulae is not allowed in the given syntax.

A revised method for computing $Prob(s, X^I\ \phi)$.

Before facing the study of a compositional approach for checking bounded Next *general* formulae, we need to consider the method for checking bounded Next formulae which can be found in the literature.

From the CSL semantics (see Proposition 2.3.2), we know that the probability of reaching a ϕ-state in one step from a state s of an arbitrary CTMC within a time bound $I = [a, b]$, is given by:

$$Prob(s, X^I\ \phi) = [e^{-a\cdot E(s)} - e^{-b\cdot E(s)}] \sum_{t \models \phi} \mathbf{P}(s,t) \tag{5.4.2}$$

In essence, the above equation tells us that the probability of satisfying ϕ in one step from s without violating the bounding interval I is given by the product of the probability of reaching a ϕ state, from s, in one step (i.e $\sum_{t \models \phi} \mathbf{P}(s,t)$), and the probability of leaving s within $I = [a, b]$ (i.e. such a probability being given by $= [e^{-a\cdot E(s)} - e^{-b\cdot E(s)}]$).

In [5] the authors claim that the state vector

$$\underline{Prob}(X^I\ \phi) = (\ldots, Prob(s, X^I\ \phi), \ldots)$$

can be obtained by multiplying the probability matrix $\mathbf{P}$ by the vector $\underline{b}_I$,

$$\underline{Prob}(X^I\ \phi) = \mathbf{P} \cdot \underline{b}_I$$

where $\underline{b}_I$ is defined as:

$$\underline{b}_I[s] = \begin{cases} e^{-a\cdot E(s)} - e^{-b\cdot E(s)} & \textbf{if } s \in Sat(\phi) \\ 0 & otherwise \end{cases} \tag{5.4.3}$$

However, the proposed algorithm, leads to a wrong result. To understand why, let us consider a simple example. In Figure 5.3 the oriented graph representing a very simple three state CTMC is depicted. The arcs of each transition are labelled with the corresponding rate, resulting in the following probability matrix:

$$\mathbf{P} = \begin{pmatrix} 0 & \frac{1}{3} & \frac{2}{3} \\ 1 & 0 & 0 \\ 1 & 0 & 0 \end{pmatrix}$$

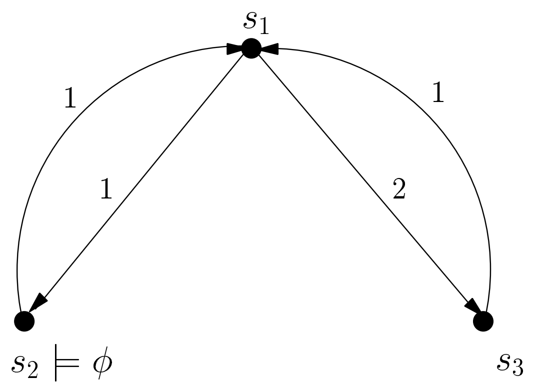


Figure 5.3: A simple arbitrary CMTC $\mathcal{M}$

We then consider an arbitrary state formula ϕ and we further assume that state s_2 is the only one satisfying ϕ. Finally, we consider $I = [2,5]$ as the bounding time interval. By applying (5.4.2) we can compute the probability of satisfying the bounded Next formula $(X^{[2,5]}\ \phi)$ for each one of the three state s_1, s_2 and s_3. This is straightforward because, since we are assuming s_2 to be the only state satisfying ϕ, then the only state with a non-zero probability of reaching a ϕ-state, in one step, is clearly s_1. Hence:

$$\begin{aligned}
Prob(s_2, X^{[2,5]}\ \phi) &= Prob(s_3, X^{[2,5]}\ \phi) = 0 \\
Prob(s_1, X^{[2,5]}\ \phi) &= [e^{-2\cdot 3} - e^{-5\cdot 3}]\mathbf{P}(s_1, s_2) \\
&= \frac{[e^{-6} - e^{-15}]}{3}
\end{aligned}$$

as clearly $E(s_1) = 3$ and $\mathbf{P}(s_1, s_2) = \frac{1}{3}$. The vector $\underline{Prob}(X^{[2,5]}, \phi)$ is then:

$$\underline{Prob}(X^{[2,5]}\ \phi) = (\frac{[e^{-6} - e^{-15}]}{3}, 0, 0)$$

From (5.4.3), we can then derive the elements of the state vector $\underline{b}_{[2,5]}$:

$$\begin{aligned}
\underline{b}_{[2,5]}[s_1] &= \underline{b}_{[2,5]}[s_3] = 0 \\
\underline{b}_{[2,5]}[s_2] &= [e^{-2} - e^{-5}]
\end{aligned}$$

as $E(s_2) = 1$. Hence the state vector $\underline{b}_{[2,5]}$ is:

$$\underline{b}_{[2,5]} = (0, [e^{-2} - e^{-5}], 0)$$

Hence by applying the algorithm proposed in [5], we need to determine the product of the matrix **P** by the vector $\underline{b}_{[2,5]}$, which is:

$$\mathbf{P} \cdot \underline{b}_{[2,5]} = (\frac{[e^{-2} - e^{-5}]}{3}, 0, 0)$$

But this proves that, contrary to what is claimed in [5], actually:

$$\underline{Prob}(X^{[2,5]}\ \phi) \neq \mathbf{P} \cdot \underline{b}_{[2,5]}$$

The problem with the above method, has to do with the definition of the state vector $\underline{b}_I$. In fact, the value of $\underline{b}_I(s)$ for a state s satisfying ϕ (i.e. $\underline{b}_I(s) = [e^{-a \cdot E(s)} - e^{-b \cdot E(s)}]$), represents the probability of *leaving* a ϕ state, within I. However, in order to calculate $\underline{Prob}(X^I\ \phi)$, the probability of *reaching* a ϕ state within I is what is needed.

In the following an alternative algorithm for computing $\underline{Prob}(X^I\ \phi)$, is introduced. It relies on the definition of the diagonal matrix I_{e_I}, whose elements represents the probability of exiting each state within the bounding interval I, and of the state-vector $\underline{i}_{\phi}$, characterising the ϕ states. This is achieved by means of the following formal definitions.

Definition 5.4.1 *Let $\mathcal{M} = (S, \mathbf{Q}, L)$ be an arbitrary labelled CTMC with state-space $S = \{s_1, s_2, \ldots, s_n\}$ and $I = [a, b] \in \mathbb{R}_{\geq 0}$ a time interval. The state-vector $\underline{e}_I$ with elements w*

$$e_I(s) = e^{-a \cdot E(s)} - e^{-b \cdot E(s)}$$

is defined. The coefficient $e_I(s)$ denotes the probability of exiting the state s within I.

We observe that, by means of the above definition, the expression (5.4.2) for the probability of reaching in one step, from s, a state satisfying a certain formula ϕ within the bounding interval I, can be re-formulated in the following way:

$$Prob(s, X^I\ \phi) = [e^{-a \cdot E(s)} - e^{-b \cdot E(s)}] \cdot \sum_{t \models \phi} \mathbf{P}(s,t) = \underline{e}_I(s) \cdot \sum_{t \models \phi} \mathbf{P}(s,t) \qquad (5.4.4)$$

The diagonal matrix consisting of coefficients $e_I(s)$ is formally introduced in the next definition.

Definition 5.4.2 *Let $\mathcal{M} = (S, \mathbf{Q}, L)$ be a labelled CTMC with state-space $S = \{s_1, s_2, \ldots, s_n\}$ and $I = [a,b] \in \mathbb{R}_{\geq 0}$ a time interval. The diagonal matrix I_{e_I} is defined as*

$$I_{e_I} = diag(\underline{e}_I) = \begin{pmatrix} e_I(s_1) & 0 & 0 & \ldots & 0 \\ 0 & e_I(s_2) & 0 & \ldots & 0 \\ 0 & 0 & \ddots & \ldots & 0 \\ \vdots & \vdots & & \ddots & \vdots \\ 0 & 0 & 0 & \ldots & e_I(s_n) \end{pmatrix}$$

where the coefficient $e_I(s)$ is as in Definition 5.4.1.

The vector characterising the states satisfying a given formula ϕ is described in the following definition.

Definition 5.4.3 *Let $\mathcal{M} = (S, \mathbf{Q}, L)$ a labelled CTMC with state-space $S = \{s_1, s_2, \ldots, s_n\}$ and ϕ a CSL state formula. The state vector $\underline{i}_\phi$ is defined as:*

$$\underline{i}_\phi(s) = \begin{cases} 1 & \textbf{if} \quad s \models \phi \\ 0 & otherwise \end{cases}$$

Finally, the following proposition provides us with a method for computing the vector $\underline{Prob}(X^I\ \phi)$ representing the probability for each state of a CTMC to fulfil a bounded Next formula $(X^I\ \phi)$.

Proposition 5.4.1 *Let $\mathcal{M} = (S, \mathbf{Q}, L)$ be a labelled CTMC with state-space $S = \{s_1, s_2, \ldots, s_n\}$, and probability matrix $\mathbf{P}$. Let ϕ be a CSL state formula and $I = [a,b] \in \mathbb{R}_{\geq 0}$ a time interval. The state vector $\underline{Prob}(X^I\ \phi) = (\ldots, Prob(s, X^I\ \phi), \ldots)$ is given by the product:*

$$\underline{Prob}(X^I\ \phi) = (I_{e_I} \cdot \mathbf{P}) \cdot \underline{i}_\phi$$

Proof.

To prove that the state vectors $\underline{Prob}(X^I\ \phi)$ and $(I_{e_I} \cdot \mathbf{P}) \cdot \underline{i}_\phi$ are identical, we need to show that for any state $s_m \in S$, $\underline{Prob}(X^I\ \phi)(s_m) = [(I_{e_I} \cdot \mathbf{P}) \cdot \underline{i}_\phi](s_m)$. The m-th element of $\underline{Prob}(X^I\ \phi)$, is, by definition,

$$\underline{Prob}(X^I\ \phi)(s_m) = Prob(s_m, X^I\ \phi) = e_I(s_m) \cdot \sum_{s_i \models \phi} \mathbf{P}(s_m, s_i)$$

Let us consider the vector $(I_{e_I} \cdot \mathbf{P}) \cdot \underline{i}_\phi$. Let the transition probability matrix $\mathbf{P}$ be:

$$\mathbf{P} = \begin{pmatrix} p_{11} & p_{12} & \cdots & \cdots & p_{1n} \\ p_{21} & p_{22} & \cdots & \cdots & p_{2n} \\ \vdots & \vdots & \ldots & \ldots & \vdots \\ p_{n1} & p_{n2} & \cdots & \cdots & p_{nn} \end{pmatrix}$$

The product of the diagonal matrix I_{e_I} by the transition probability matrix $\mathbf{P}$ leads to the following matrix:

$$I_{e_I} \cdot \mathbf{P} = \begin{pmatrix} e_I(s_1) \cdot p_{11} & e_I(s_1) \cdot p_{12} & \cdots & \cdots & e_I(s_1) \cdot p_{1n} \\ e_I(s_2) \cdot p_{21} & e_I(s_2) \cdot p_{22} & \cdots & \cdots & e_I(s_2) \cdot p_{2n} \\ \vdots & \vdots & \ldots & \ldots & \vdots \\ e_I(s_n) \cdot p_{n1} & e_I(s_n) \cdot p_{n2} & \cdots & \cdots & e_I(s_n) \cdot p_{nn} \end{pmatrix}$$

The m-th element of the vector $(I_{e_I} \cdot \mathbf{P}) \cdot \underline{i}_\phi$, is given by the product of the m-th row of the matrix $I_{e_I} \cdot \mathbf{P}$ by $\underline{i}_\phi$, which is:

$$\begin{aligned} [(I_{e_I} \cdot \mathbf{P}) \cdot \underline{i}_\phi](s_m) &= (e_I(s_m) \cdot p_{m1}, e_I(s_m) \cdot p_{m2}, \ldots, e_I(s_m) \cdot p_{mn}) \cdot \underline{i}_\phi \\ &= \sum_{s_i \models \phi} e_I(s_m) \cdot p_{mi} \\ &= e_I(s_m) \cdot \sum_{s_i \models \phi} \mathbf{P}(s_m, s_i) \end{aligned}$$

which proves the equality between the state vectors $\underline{Prob}(X^I\ \phi)$ and $(I_{e_I} \cdot \mathbf{P}) \cdot \underline{i}_\phi$.

□

Relying on the result of Proposition 5.4.1, the following algorithm can be used, as an alternative to the one proposed in [5], for checking a bounded Next formula with respect to a state s.

Algorithm 5.4.1 *Let s be a state of an arbitrary labelled CMTC $\mathcal{M}$, ϕ a CSL state formula and $I = [a,b] \subseteq \mathbb{R}_{\geq 0}$ a time interval. The following procedure returns YES if the formula $\mathcal{P}_{\trianglelefteq p}(X^I\ \phi)$ is valid in s, and NO otherwise.*

Algorithm

- *Compute $Sat(\phi)$.*

- *Determine the state vector $\underline{i}_\phi$, as:*

$$\underline{i}_\phi(s) = \begin{cases} 1 & \textbf{if} \qquad s \models \phi \\ 0 & otherwise \end{cases}$$

- *Determine the diagonal matrix I_{e_I}.*

- *Compute the state vector*

$$(I_{e_I} \cdot \mathbf{P}) \cdot \underline{i}_\phi$$

- IF $[(I_{e_I} \cdot \mathbf{P}) \cdot \underline{i}_\phi](s) \trianglelefteq p$ **THEN** *return YES;* **ELSE** *return NO;*

□

Let us apply the above algorithm to determine $\underline{Prob}(X^{[2,5]}\ \phi)$ with respect to the CTMC of Figure 5.3. The diagonal matrix $I_{e_{[2,5]}}$ is given by:

$$I_{e_{[2,5]}} = \begin{pmatrix} e^{-2\cdot 3} - e^{-5\cdot 3} & 0 & 0 \\ 0 & e^{-2\cdot 1} - e^{-5\cdot 1} & 0 \\ 0 & 0 & e^{-2\cdot 1} - e^{-5\cdot 1} \end{pmatrix}$$

which multiplied by the transition probability matrix $\mathbf{P}$ gives:

$$I_{e_{[2,5]}} \cdot \mathbf{P} = \begin{pmatrix} 0 & \frac{e^{-6} - e^{-15}}{3} & \frac{2\cdot(e^{-6} - e^{-15})}{3} \\ e^{-2} - e^{-5} & 0 & 0 \\ e^{-2} - e^{-5} & 0 & 0 \end{pmatrix}$$

The vector $\underline{i}_\phi$ is given by:

$$\underline{i}_\phi = (0,1,0)$$

Hence the product $[I_{e_{[2,5]}} \cdot \mathbf{P}] \cdot \underline{i}_\phi$ results in:

$$[I_{e_{[2,5]}} \cdot \mathbf{P}] \cdot \underline{i}_\phi = (\frac{e^{-6} - e^{-15}}{3}, 0, 0)$$

which proves

$$\underline{Prob}(X^{[2,5]}\ \phi) = [I_{e_{[2,5]}} \cdot \mathbf{P}] \cdot \underline{i}_\phi$$

Having provided a method for checking bounded Next formulae with respect to an arbitrary CTMC, we can get back to our original goal, which is the study of a compositional way for verifying *general* bounded Next formulae referring to a bidimensional Boucherie process.

The characterisation of a decomposed approach for checking of bounded Next *general* formulae, relies on a fundamental result which is proved in Theorem 5.4.1. In order to prove that result, some preliminary definitions and properties need to be introduced.

The following definition introduces the idea of *derived* time interval (or k-projection of a time interval). Given $I = [a,b]$ and a state (s^1, s^2) of a bidimensional Boucherie process, the interval $I^k(s^1, s^2)$ is obtained by shifting I with respect to the probability of leaving (s^1, s^2) with a *k-move*, if the shared resource is not held by component $\mathcal{M}_j$ in (s^1, s^2), or with respect to the ratio $\frac{E_j(s^j)}{E_k(s^k)}$, if $\mathcal{M}_j$ holds the resource in (s^1, s^2).

Definition 5.4.4 (k-projection of a time interval I) *Let $I = [a,b] \subset \mathbb{R}_{\geq 0}$ be a time interval, (s^1, s^2) a state of a bidimensional Boucherie process $\mathcal{M}$ and $p^k(s^1, s^2)$ the probability of making a k-move out of (s^1, s^2). The time interval $I^k(s^1, s^2)$ defined as:*

$$I^k(s^1, s^2) = \begin{cases} \left[\frac{a}{p^k(s^1,s^2)}, \frac{b}{p^k(s^1,s^2)}\right] & \textbf{if } (s^1, s^2) \notin R_j \\ \\ \left[\frac{a \cdot E_j(s^j)}{E_k(s^k)}, \frac{b \cdot E_j(s^j)}{E_k(s^k)}\right] & \textbf{if } (s^1, s^2) \in R_j \end{cases}$$

is called the k-projection of I with respect to the state (s^1, s^2).

The relevance of the definition of k-projection of a time interval with respect to a given state of a bidimensional Boucherie process, stands in the result of the following proposition, which is: the probability of exiting a state (s^1,s^2) of a bidimensional Boucherie process within the bound I, is equal to the probability of exiting the s^k of component $\mathcal{M}_k$, within the k-projection of I, $I^k(s^1,s^2)$.

Proposition 5.4.2 *Let (s^1,s^2) be a state of a bidimensional Boucherie process and $I=[a,b]\subseteq\mathbb{R}_{\geq 0}$ a time interval. The following equality holds:*

$$e_I(s^1,s^2) = e_{I^k(s^1,s^2)}(s^k)$$

where $e_I()$ is as in Definition 5.4.1.

Proof.
A distinction with respect to the partitions of the Boucherie's state-space, needs to be considered.

1. $(s^1,s^2)\notin R_j$. In this case, from Definition 5.4.4, we have:

$$I^k(s^1,s^2) = \Big[\frac{a}{p^k(s^1,s^2)}, \frac{b}{p^k(s^1,s^2)}\Big]$$

A further distinction has to be considered:

- $(s^1,s^2)\in R_{free}$. In this case $E(s^1,s^2)=E_1(s^1)+E_2(s^2)$ and also $p^k(s^1,s^2)=\frac{E_k(s^k)}{E_1(s^1)+E_2(s^2)}$. Hence:

$$\begin{aligned}
e_I(s^1,s^2) &= e^{-a\cdot(E_1(s^1)+E_2(s^2))} - e^{-b\cdot(E_1(s^1)+E_2(s^2))}\\
&= e^{-a\cdot\frac{(E_1(s^1)+E_2(s^2))}{E_k(s^k)}E_k(s^k)} - e^{-b\cdot\frac{(E_1(s^1)+E_2(s^2))}{E_k(s^k)}E_k(s^k)}\\
&= e^{-\frac{a}{p^k(s^1,s^2)}E_k(s^k)} - e^{-\frac{b}{p^k(s^1,s^2)}E_k(s^k)}\\
&= e_{\Big[\frac{a}{p^k(s^1,s^2)},\frac{b}{p^k(s^1,s^2)}\Big]}(s^k)\\
&= e_{I^k(s^1,s^2)}(s^k)
\end{aligned}$$

- $(s^1,s^2)\in R_k$. In this case $E(s^1,s^2)=E_k(s^k)$ and also $p^k(s^1,s^2)=1$. Hence,

$$I^k(s^1,s^2) = \Big[\frac{a}{p^k(s^1,s^2)}, \frac{b}{p^k(s^1,s^2)}\Big] = [a,b]$$

and also:

$$e_I(s^1,s^2) = e^{-a\cdot E_k(s^k)} - e^{-b\cdot E_k(s^k)} = e_{[a,b]}(s^k) = e_{I^k(s^1,s^2)}(s^k)$$

2. $(s^1,s^2) \in R_j$. In this case, from Definition 5.4.4, we have:

$$I^k(s^1,s^2) = \left[\frac{a\cdot E_j(s^j)}{E_k(s^k)}, \frac{b\cdot E_j(s^j)}{E_k(s^k)}\right]$$

Furthermore $E(s^1,s^2) = E_j(s^j)$, hence:

$$\begin{aligned}
e_I(s^1,s^2) &= e^{-a\cdot E_j(s^j)} - e^{-b\cdot E_j(s^j)} \\
&= e^{-a\cdot \frac{E_j(s^j)}{E_k(s^k)} E_k(s^k)} - e^{-b\cdot \frac{E_j(s^j)}{E_k(s^k)} E_k(s^k)} \\
&= e_{\left[\frac{a\cdot E_j(s^j)}{E_k(s^k)}, \frac{b\cdot E_j(s^j)}{E_k(s^k)}\right]}(s^k) \\
&= e_{I^k(s^1,s^2)}(s^k)
\end{aligned}$$

□

The next theorem shows a basic equivalence regarding the semantics of *general* bounded Next formulae. This result provides us with a decomposed relation which allows for the definition of an algorithm for compositional checking of such formulae.

Theorem 5.4.1 *Let (s^1,s^2) be a state of a bidimensional Boucherie process, $p \in [0,1]$ a probability bound, $\trianglelefteq \in \{<,\leq,\geq,>\}$, $I = [a,b] \subseteq \mathbb{R}_{\geq 0}$ a time interval, ψ_{12} a formula as in (5.4.1). The following equivalence holds:*

$$(s^1,s^2) \models \mathcal{P}_{\trianglelefteq p}(X^I\ \psi_{12}) \iff \left[\sum_{(\alpha_1,\alpha_2)\in DecSat(\psi_{12})} p^1(s^1,s^2)\cdot \underline{Prob_1}(X^{I^1(s^1,s^2)}\alpha_1)(s^1) + p^2(s^1,s^2)\cdot \underline{Prob_2}(X^{I^2(s^1,s^2)}\alpha_2)(s^2)\right] \trianglelefteq p \tag{5.4.5}$$

Proof.

(⇒) From the CSL semantics we know that if $(s^1,s^2) \models \mathcal{P}_{\trianglelefteq p}(X^I\ \psi_{12})$ then

$$\left[e^{-a\cdot E(s^1,s^2)} - e^{-b\cdot E(s^1,s^2)}\right]\cdot\left[\sum_{(t^1,t^2)\in Sat(\psi_{12})} \mathbf{P}((s^1,s^2),(t^1,t^2))\right] \trianglelefteq p$$

which is:

$$e_I(s^1,s^2)\cdot\Big[\sum_{(t^1,t^2)\in Sat(\psi_{12})}\mathbf{P}((s^1,s^2),(t^1,t^2))\Big]\trianglelefteq p$$

However $Sat(\psi_{12})$ can be decomposed, by means of the Lemma 4.3.2, in the following way:

$$Sat(\psi_{12})=\bigcup_{(\alpha_1,\alpha_2)\in DecSat(\psi_{12})}\big[Sat_1(\alpha_1)\times Sat_2(\alpha_2)\big]\setminus(R_1R_2)$$

Hence the sum in the above inequality can be re-formulated in terms of $DecSat(\psi_{12})$, resulting in[4]*:*

$$e_I(s^1,s^2)\cdot\sum_{(\alpha_1,\alpha_2)\in DecSat(\psi_{12})}\Big[\sum_{(t^1,t^2)\in Sat_1(\alpha_1)\times Sat_2(\alpha_2)}\mathbf{P}((s^1,s^2),(t^1,t^2))\Big]\trianglelefteq p \tag{5.4.6}$$

Since, in a Boucherie process each transition corresponds to a change of state for exactly one component, then the above inequality is equivalent to the following one:

$$e_I(s^1,s^2)\cdot\sum_{(\alpha_1,\alpha_2)\in DecSat(\psi_{12})}\Big[\sum_{(t^1,s^2)\in Sat_1(\alpha_1)\times Sat_2(\alpha_2)}\mathbf{P}((s^1,s^2),(t^1,s^2))+\sum_{(s^1,t^2)\in Sat_1(\alpha_1)\times Sat_2(\alpha_2)}\mathbf{P}((s^1,s^2),(s^1,t^2))\Big]\trianglelefteq p \tag{5.4.7}$$

From Remark 4.2.2, we know that the probability of a 1*-move (*2*-move) from a state* (s^1,s^2) *is a factor of the probability of the corresponding component's transition, which is:*

$$\mathbf{P}((s^1,s^2),(t^1,s^2))=p^1(s^1,s^2)\cdot\mathbf{P}_1(s^1,t^1)$$

where $p^1(s^1,s^2)$ is the probability of a 1*-move from (s^1,s^2). Hence (5.4.7) results in:*

$$e_I(s^1,s^2)\cdot\sum_{(\alpha_1,\alpha_2)\in DecSat(\psi_{12})}\Big[p^1(s^1,s^2)\sum_{t^1\in Sat_1(\alpha_1)}\mathbf{P}_1(s^1,t^1)+p^2(s^1,s^2)\sum_{t^2\in Sat_2(\alpha_2)}\mathbf{P}_2(s^2,t^2)\Big]\trianglelefteq p$$

which we can re-write as:

$$\sum_{(\alpha_1,\alpha_2)\in DecSat(\psi_{12})}\Big[e_I(s^1,s^2)p^1(s^1,s^2)\sum_{t^1\in Sat_1(\alpha_1)}\mathbf{P}_1(s^1,t^1)+e_I(s^1,s^2)p^2(s^1,s^2)\sum_{t^2\in Sat_2(\alpha_2)}\mathbf{P}_2(s^2,t^2)\Big]\trianglelefteq p \tag{5.4.8}$$

[4] It should be noted that the R_1R_2 states need not to be excluded from the innermost sum in (5.4.6), as the probability of reaching such a state from (s^1,s^2) is clearly zero (i.e. $\forall(t^1,t^2)\in R_1R_2$ and $\forall(s^1,s^2)\in S$, $\mathbf{P}((s^1,s^2),(t^1,t^2))=0$)

From Proposition 5.4.2 we have $e_I(s^1,s^2) = e_{I^k(s^1,s^2)}(s^k)$ *which, substituted in (5.4.8), gives:*

$$\sum_{(\alpha_1,\alpha_2)\in DecSat(\psi_{12})} \Big[e_{I^1(s^1,s^2)}(s^1) p^1(s^1,s^2) \sum_{t^1\in Sat_1(\alpha_1)} \mathbf{P}_1(s^1,t^1) + e_{I^2(s^1,s^2)}(s^2) p^2(s^1,s^2) \sum_{t^2\in Sat_2(\alpha_2)} \mathbf{P}_2(s^2,t^2) \Big] \trianglelefteq p \tag{5.4.9}$$

But from Proposition 5.4.1 we know that

$$\underline{Prob}_1(X^{I^1(s^1,s^2)}\ \alpha_1)(s_1) = e_{I^1(s^1,s^2)}(s^1) \sum_{t^1\in Sat_1(\alpha_1)} \mathbf{P}_1(s^1,t^1)$$

and also:

$$\underline{Prob}_2(X^{I^2(s^1,s^2)}\ \alpha_2)(s_2) = e_{I^2(s^1,s^2)}(s^2) \sum_{t^2\in Sat_2(\alpha_2)} \mathbf{P}_2(s^2,t^2)$$

which by substituting in (5.4.9), proves the implication $(\Rightarrow)$*.*

$(\Leftarrow)$*. By reversing* $(\Rightarrow)$*.*

$\square$

The result of the above theorem suggests the definition of the following algorithm for decomposed checking of *general* bounded Next formulae with respect to a bidimensional Boucherie process.

Algorithm 5.4.2 *Let* (s^1,s^2) *be a state of a bidimensional Boucherie process* $\mathcal{M}$ *with components* $\mathcal{M}_1$ *and* $\mathcal{M}_2$*,* ψ_{12} *a general formula as in (5.4.1),* $p \in [0,1]$ *a probability bound,* $\trianglelefteq \in \{<,\leq,\geq,>\}$ *and* $I = [a,b] \subseteq \mathbb{R}_{\geq 0}$ *a time interval. The following algorithm can be applied for checking whether:*

$$(s^1,s^2) \models \mathcal{P}_{\trianglelefteq p}(X^I\ \psi_{12})$$

Algorithm $((s^1,s^2) \models \mathcal{P}_{\trianglelefteq p}(X^I\ \psi_{12}))$.

1. $PX = 0$;

2. *Determine the set of pairs* $DecSat(\psi_{12})$ *by application of Definition 4.3.3.*

3. *Determine the diagonal matrices:*

$$\begin{aligned} M^1 =& I^1_{e_{I^1(s^1,s^2)}} \\ M^2 =& I^2_{e_{I^2(s^1,s^2)}} \end{aligned}$$

for component $\mathcal{M}_1$ *and component* $\mathcal{M}_2$ *respectively.*

4. FOR $(\alpha_1, \alpha_2) \in DecSat(\psi_{12})$ **DO**

- *determine* $Sat_1(\alpha_1)$ *and* $Sat_2(\alpha_2)$*, hence the vectors* $\underline{b}^1_\phi$ *and* $\underline{b}^2_\phi$*;*
- *determine the vectors* $\underline{Prob}_1(X^{I^1(s^1,s^2)}\ \alpha_1)$ *and* $\underline{Prob}_1(X^{I^1(s^1,s^2)}\ \alpha_1)$*:*

$$\begin{aligned} \underline{Prob}_1(X^{I^1(s^1,s^2)}\ \alpha_1) &= (M^1 \cdot \mathbf{P}_1) \cdot \underline{b}^1_\phi; \\ \underline{Prob}_2(X^{I^2(s^1,s^2)}\ \alpha_2) &= (M^2 \cdot \mathbf{P}_2) \cdot \underline{b}^2_\phi; \end{aligned}$$

- $PX = PX + p^1(s^1,s^2)\underline{Prob}_1(X^{I^1(s^1,s^2)}\ \alpha_1)(s_1) + p^2(s^1,s^2)\underline{Prob}_2(X^{I^2(s^1,s^2)}\ \alpha_2)(s^2)$;

5. IF $PX \trianglelefteq p$ **THEN** *return YES;* **ELSE** *return NO.*

$\square$

Complexity analysis.

A precise evaluation of the complexity of the decomposed verification method described by the above algorithm, is not part of this work. However, here some intuitive considerations are given. As a first thing we observe that, by means of the above method, the verification of a bounded Next *general* formula, like $\mathcal{P}_{\trianglelefteq p}(X^I\ \psi_{12})$, on the product process $\mathcal{M}$, is replaced by the verification of some bounded Next *single-component* formulae on the components $\mathcal{M}_1$ and $\mathcal{M}_2$, respectively. The number of corresponding *single-component* formulae is equal to the number of partitions of $Sat(\psi_{12})$. Thus, if $Sat(\psi_{12})$ consists of n partitions (i.e. $|DecSat(\psi_{12})| = n$), then checking $\mathcal{P}_{\trianglelefteq p}(X^{I}\ \psi_{12})$ with respect to the state-space S, boils down to checking n derived formulae $\mathcal{P}_{\trianglelefteq p}(X^{I'}\ \psi_1^i)$, with respect to state space S_1 and n derived formulae $\mathcal{P}_{\trianglelefteq p}(X^{I''}\ \psi_2^i)$, with respect to state space S_2 $(1 \leq i \leq n)$. The computational saving implied by that, depends on both the dimension and the structure of the components' state-space (the computational gain is proportional to the ratios $\frac{|S_{k,R}|}{|S_{k,\overline{R}}|}$). Checking of an arbitrary bounded Next $\mathcal{P}_{\trianglelefteq p}(X^{I}\ \phi)$ on a state-space S (see Algorithm 5.4.1), requires the computation of $Sat(\phi)$ and of the matrix-matrix-vector product $(I_{e_I} \cdot \mathbf{P}) \cdot \underline{i}_\phi$. Its computational cost is clearly proportional to the dimension of S. The state-space's dimension of a bidimensional Boucherie process is:

$$|S| = |S_1| \cdot |S_2| - (|S_{1,R}| \cdot |S_{2,r}|)$$

where the dimension of component $\mathcal{M}_k$'s state-space is:

$$|S_k| = |S_{k,\overline{R}}| + |S_{k,R}|$$

The saving gained through the compositional algorithm, depends on the percentage of states in which each component holds the shared resource (i.e. the cardinality of $S_{k,R}$). For example, if we consider a state-space of 1000 elements for both components (i.e. $|S_1| = |S_2| = 1000$), and we assume that for each component the states representing the resource holding are 5% of the whole (i.e. $|S_{1,R}| = |S_{2,R}| = 50$), then the application of the compositional checking to $\mathcal{P}_{\trianglelefteq p}(X^{I}\ \psi_{12})$ results in checking $2 \cdot n$ bounded Next formulae ($\mathcal{P}_{\trianglelefteq p}(X^{I'}\ \psi_1^k)$) over a 1000 elements state-space, instead of checking a single bounded Next formula with respect to a 997500 elements state-space (in fact, in this case, $|S| = 1000 \cdot 1000 - (50 \cdot 50) = 997500$). We further observe that the number n of partitions $Sat(\psi_{12})$ consists of, depends on the number of nested negation connectives $(\neg)$ contained in ψ_{12} and hardly exceed some units[5].

Finally, to be more precise, we notice that also the cost for computing $DecSat(\psi_{12})$ has also to be considered in the evaluation of the complexity impact of decomposed checking of bounded Next *general* formulae.

[5]In Example 4.3.1 it has been shown that the satisfiability set of a rather complex formula like $\neg[\neg(a_1 \wedge a_2) \wedge \neg(b_1 \wedge b_2)]$ results in three partitions. The study of the existence of a relationship between the structure of a *general* formula ψ_{12} and the number of partitions its satisfiability set results in, is an interesting subject for future works.

Example 5.4.1 *Let us consider an example of general bounded Next formulae, referring to the GIS Boucherie framework of our running example (see Figure 5.1 and Figure 5.2) and let us show that the decomposed approach for checking it is actually correct. Let us suppose we are interested in verifying that there is at least a* 50% *possibility that soon after being started (i.e. initial state* (s_{10}, s_{20})*, Figure 5.2) and within the time interval* $[2,5]$*, the GIS process reaches a state where component* $\mathcal{M}_1$ *is reading the shared register (i.e. read*$_1$*). This corresponds to checking that the following folds:*

$$(s_{10}, s_{20}) \models \mathcal{P}_{\geq 0.5}(X^{[2,5]}\ (read_1 \wedge idle_2))$$

By means of the "standard" approach (non-decompositional checking), we can apply Algorithm 5.4.1 to compute the state vector $\underline{Prob}(X^{[2,5]}\ (read_1 \wedge idle_2))$ *for the product process. Hence*

$$\underline{Prob}(X^{[2,5]}\ (read_1 \wedge idle_2)) = (I_{e_{[2,5]}} \cdot \mathbf{P}) \cdot \underline{i}_{(read_1 \wedge idle_2)}$$

From Figure 5.2, we know that the only state where $(read_1 \wedge idle_2)$ *is valid is* (s_{12}, s_{20})*. Hence the vector* $\underline{i}_{(read_1 \wedge idle_2)}$ *is:*

$$\underline{i}_{(read_1 \wedge idle_2)} = (0,0,0,0,0,0,0,0,0,0,1,0,0,0)$$

The transitions' probability matrix for the product process is:

$$\mathbf{P} = \begin{pmatrix}
0 & \frac{r_1}{2r_1+r_5} & 0 & 0 & 0 & \frac{r_5}{2r_1+r_5} & 0 & 0 & 0 & 0 & \frac{r_1}{2r_1+r_5} & 0 & 0 & 0 \\
0 & 0 & \frac{r_2}{r_2+r_4} & \frac{r_4}{r_2+r_4} & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\frac{r_2}{r_1+r_2} & 0 & 0 & 0 & 0 & 0 & \frac{r_1}{r_1+r_2} & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{r_2}{r_2+r_4} & \frac{r_4}{r_2+r_4} & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & \frac{r_2}{r_2+r_4} & \frac{r_4}{r_2+r_4} & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}$$

It is straightforward to show that the product of the matrix $(I_{e_{[2,5]}} \cdot \mathbf{P})$ *by the vector* $\underline{i}_{(read_1 \wedge idle_2)}$, *leads to the following state vector:*

$$(I_{e_{[2,5]}} \cdot \mathbf{P}) \cdot \underline{b}_{(read_1 \wedge idle_2)} = (\frac{r_1}{r_1 + r_5}\, [e^{-2(2r_2+r_5)} - e^{-5(2r_2+r_5)}], 0,0,0,0,0,0,0,0,0,0,0,0,0,0)$$

which, as expected, tells us that the only state from which there is a non-null probability of reaching, within the time bound $[2,5]$, *a state where component* $\mathcal{M}_1$ *is reading the register while* $\mathcal{M}_2$ *is idle, is the initial state* (s_{10}, s_{20}). *Furthermore, the probability for such a state is equal to:*

$$Prob((s_{10}, s_{20}), X^{[2,5]}\ (read_1 \wedge idle_2)) = \frac{r_1}{r_1 + r_5}[e^{-2(2r_2+r_5)} - e^{-5(2r_2+r_5)}]$$

As a consequence we have that

$$(s_{10}, s_{20}) \models \mathcal{P}_{\geq 0.5}(X^{[2,5]}\ (read_1 \wedge idle_2)) \Leftrightarrow \frac{r_1}{r_1 + r_5}(e^{-2(2r_2+r_5)} - e^{-5(2r_2+r_5)}) \trianglelefteq p \tag{5.4.10}$$

In the above, we have shown how to verify $(s_{10}, s_{20}) \models \mathcal{P}_{\geq 0.5}(X^{[2,5]}\ (read_1 \wedge idle_2))$ *by application of the "standard" method with respect to the product process. Let us see that the application of decomposed checking leads to the an equivalent result.*

As a first step, we need to determine the partitions of $Sat(read_1 \wedge idle_2)$. *Trivially,* $DecSat(read_1 \wedge idle_2) = \{(read_1, idle_2)\}$, *meaning that the satisfiability set for* $(read_1 \wedge idle_2)$ *consists of a single partition:*

$$Sat(read_1 \wedge idle_2) = Sat_1(read_1) \times Sat_2(idle_2) \setminus R_1 R_2$$

As a result we know that we will have to check one bounded Next formula (i.e. $(X^{[\frac{a(2r_1+r_5)}{r_1+r_5}]}\ read_1))$ *on component* $\mathcal{M}_1$ *and another one (i.e.* $(X^{[\frac{a(2r_1+r_5)}{r_5}]}\ idle_2))$, *on component* $\mathcal{M}_2$.

In order to compute the state vector $\underline{Prob}_1(X^{[\frac{a(2r_1+r_5)}{r_1+r_5}]}\ read_1)$, *first we need to determine* $\underline{i}^1_{read_1}$. *But since* s_{12} *is the only state of* $\mathcal{M}_1$ *satisfying* $read_1$ *(i.e.* $Sat_1(read_1) = \{s_{12}\}$*), then:*

$$\underline{i}^1_{read_1} = (0,0,1,0,0,0)$$

Similarly for the vector $\underline{i}^2_{idle_2}$, *we have:*

$$\underline{i}^2_{idle_2} = (1,0,0,0,0)$$

Next we need to consider the diagonal matrices built on the projections of the bounding interval $I = [2,5]$ *with respect to state* (s_{10}, s_{20})*. As the probability of a* 1*-move and of a* 2*-move out of* (s_{10}, s_{20}) *are respectively* $p^1(s_{10}, s_{20}) = \frac{r_1+r_5}{2r_1+r_5}$ *and* $p^2(s_{10}, s_{20}) = \frac{r_5}{2r_1+r_5}$*, then the projections of the time interval I are:*

$$
\begin{aligned}
I^1(s^1,s^2) &= \Big[\frac{a(2r_1+r_5)}{r_1+r_5}, \frac{a(2r_1+r_5)}{r_1+r_5}\Big] \\
I^2(s^1,s^2) &= \Big[\frac{a(2r_1+r_5)}{r_5}, \frac{a(2r_1+r_5)}{r_5}\Big]
\end{aligned}
$$

The derivation of the diagonal matrices $I_{e_{I^1(s^1,s^2)}}$ *and* $I_{e_{I^2(s^1,s^2)}}$ *is then straightforward (however, for brevity, here we skip it). Knowing that the transition probability matrices* $\mathbf{P}_1$ *and* $\mathbf{P}_2$ *are equal to:*

$$
\mathbf{P}_1 = \begin{pmatrix}
0 & \frac{r_5}{r_1+r_5} & \frac{r_1}{r_1+r_5} & 0 & 0 & 0 \\
1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & \frac{r_2}{r_2+r_4} & \frac{r_4}{r_2+r_4} & 0 \\
0 & 0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0 & 0
\end{pmatrix}
$$

$$
\mathbf{P}_2 = \begin{pmatrix}
0 & 1 & 0 & 0 & 0 \\
0 & 0 & \frac{r_2}{r_2+r_4} & \frac{r_4}{r_2+r_4} & 0 \\
0 & 0 & 0 & 0 & 1 \\
0 & 0 & 0 & 0 & 1 \\
1 & 0 & 0 & 0 & 0
\end{pmatrix}
$$

the state vectors $\underline{Prob}_1(X^{[\frac{a(2r_1+r_5)}{r_1+r_5}, \frac{a(2r_1+r_5)}{r_1+r_5}]}\ read_1)$ *and* $\underline{Prob_2}(X^{[\frac{a(2r_1+r_5)}{r_5}, \frac{a(2r_1+r_5)}{r_5}]}\ idle_2)$*,* *can easily be obtained as the products*

$$\underline{Prob}_1(X^{[\frac{2(2r_1+r_5)}{r_1+r_5},\frac{5(2r_1+r_5)}{r_1+r_5}]}\ read_1) = (I_{e_{I^1(s^1,s^2)}} \cdot \mathbf{P}_1) \cdot \underline{i}^1_{read_1}$$
$$= (\frac{r_1}{r_1+r_5}(e^{-2(2r_1+r_5)} - e^{-5(2r_1+r_5)}), 0, 0, 0, 0, 0)$$

$$\underline{Prob}_2(X^{[\frac{2(2r_1+r_5)}{r_5},\frac{5(2r_1+r_5)}{r_5}]}\ idle_2) = (I_{e_{I^2(s^1,s^2)}} \cdot \mathbf{P}_2) \cdot \underline{i}^2_{idle_2}$$
$$= (0, 0, 0, 0, e^{\frac{-2r_3(2r_1+r_5)}{r_1+r_5}} - e^{\frac{-5r_3(2r_1+r_5)}{r_1+r_5}})$$

Finally we have to consider the weighted sum of the elements of vectors $\underline{Prob}_1(X^{[\frac{2(2r_1+r_5)}{r_1+r_5},\frac{5(2r_1+r_5)}{r_1+r_5}]}\ read_1)$ *and* $\underline{Prob}_2(X^{[\frac{2(2r_1+r_5)}{r_5},\frac{5(2r_1+r_5)}{r_5}]}\ idle_2)$ *corresponding to the considered source state of the product process (i.e.* (s_{10}, s_{20})*) and check whether it is* $\trianglelefteq p$*.*

$$\begin{aligned}
PX &= p_1(s_{10}, s_{20})\underline{Prob}_1(X^{[\frac{2(2r_1+r_5)}{r_1+r_5},\frac{5(2r_1+r_5)}{r_1+r_5}]}\ read_1)(s_{10}) \\
&\quad + p_2(s_{10}, s_{20})\underline{Prob}_2(X^{[\frac{2(2r_1+r_5)}{r_5},\frac{5(2r_1+r_5)}{r_5}]}\ idle_2)(s_{20}) \\
&= \frac{r_1+r_5}{2r_1+r5} \cdot \frac{r_1}{r_1+r_5}(e^{-2(2r_1+r_5)} - e^{-5(2r_1+r_5)}) + \frac{r_5}{2r_1+r5} \cdot 0 \\
&= \frac{r_1}{2r_1+r5} \cdot (e^{-2(2r_1+r_5)} - e^{-5(2r_1+r_5)}) \trianglelefteq p
\end{aligned}$$

which proves that by application of the decomposed method with respect to the components $\mathcal{M}_1$ *and* $\mathcal{M}_2$*, we have obtained exactly the same condition (5.4.10) given by the application of the "standard" method on the product process.*

$\square$

Chapter 6

Compositional CSL model checking: Until formulae

6.1 Introduction

In this chapter the study of a compositional way for checking Until formulae on a bidimensional Boucherie process is tackled. At the time being, the results we managed to obtain are not complete. The main issue is the semantics of the Until formulae which makes the derivation of a decomposed technique a difficult task.

From the CSL semantics we know that a path σ from a state s of an arbitrary CTMC $\mathcal{M}$, satisfies an Until formula like $(\phi' \; \mathcal{U}^I \; \phi'')$ (i.e. $\sigma \models (\phi' \; \mathcal{U}^I \; \phi'')$), if and only if, there exists a future time instant $t \in I$ such that the state σ is in at t, $\sigma@t$, validates ϕ'' (i.e. $\sigma@t \models \phi''$), while up until time t σ satisfies ϕ' (i.e. $\sigma@t' \models \phi'$, $\forall t' < t$). For the unbounded Until (i.e. $I = [0, \infty)$), this is equivalent to say that there has to be a future state $n \geq 0$ at which σ validates ϕ'' ($\sigma[n] \models \phi''$) and such that for each predecessor $m < n$, $\sigma[m] \models \phi'$. Furthermore a probabilistic Until formula, like $\mathcal{P}_{\trianglelefteq p}(\phi' \; \mathcal{U}^I \; \phi'')$, is satisfied in a state s, if and only if, the probability measure of the paths (from state s) which satisfy $(\phi' \; \mathcal{U}^I \; \phi'')$ is $\trianglelefteq p$ (i.e. $Prob^{\mathcal{M}}(s, (\phi' \; \mathcal{U}^I \; \phi'')) \trianglelefteq p$). The main problem with the verification of a probabilistic Until formula, is the evaluation of the probability measure $Prob^{\mathcal{M}}(s, (\phi' \; \mathcal{U}^I \; \phi''))$. In Chapter 2, we have seen that, for bounded Until formulae (i.e. $I \neq [0, \infty)$), this, generally, involves the solution of a Volterra integral equation system, requiring complex mathematical methods. However, it has been shown, that by means of proper manipulations, this problem boils down to the computation of transient state probabilities for a transformed CTMC, a good approximation of which can be obtained by applying the Uniformisation method (see [42]). In contrast, the verification of unbounded Until formulae, implies the solution of a system of linear equations. In both cases (i.e. bounded and unbounded Until), the search for a compositional semantics, would require finding a decomposed relationship from the application of solving method (i.e. Uniformisation, for bounded Until or solution of a system of linear equation, for the unbounded Until).

Very recently, we have begun to consider the application of tensorial algebras to a bidimensional Boucherie framework for obtaining a compositional expression of the *infinitesimal generator matrix* (i.e. $\mathbf{Q}$) and of its associated *transition probability matrix* (i.e. $\mathbf{P}$). Our believe is that, that could be of some help in finding a decomposed method for checking both bounded and unbounded Until formulae which refer to a bidimensional Boucherie process.

In the remainder of the chapter, instead, we will show the type of result we have achieved by considering a *path-wise* interpretation of the Until semantics rather than a *state-wise*. As we said, verifying an Until formula with respect to a state s of $\mathcal{M}$, requires the evaluation of the probability measure $Prob^{\mathcal{M}}(s, (\phi' \; \mathcal{U}^I \; \phi''))$. This value, accounts for the probability measure of each path that satisfies $(\phi' \; \mathcal{U}^I \; \phi'')$, which is:

$$Prob^{\mathcal{M}}(s, (\psi'_k \; \mathcal{U} \, \psi''_k)) = \sum_{\sigma \in Path(s, (\psi'_k \; \mathcal{U} \; \psi''_k))} Pr(\sigma)$$

By means of the existing methods the value of $Prob^{\mathcal{M}}(s, (\phi' \; \mathcal{U}^I \; \phi''))$ is worked out in a *state-wise* fashion: a recursive function defined on S, computes that probability value by unravelling only those paths which satisfy $(\phi' \; \mathcal{U}^I \; \phi'')$. In essence the set $Path(s, (\psi'_k \; \mathcal{U} \, \psi''_k))$ is never determined, and the above mentioned methods are proved to provide a value which is equivalent to the sum of the probability measure of the paths satisfying $(\phi' \; \mathcal{U}^I \; \phi'')$.

Our argument here, originates from observing that there exists a close relation between the paths of the product process and the paths of the components' processes in a bidimensional Boucherie framework. In particular, as we will see in detail in the next section, each path of the product-process results from interleaving two corresponding paths (*projection paths*), one from component $\mathcal{M}_1$ and the other from component $\mathcal{M}_2$. As a consequence, it can be shown that the probability measure of a path of the product-process can be factored by means of the probability measure of its two corresponding *projection paths*.

This leads to a non-constructive proof showing the existence of a compositional semantics for Until formulae[1]. In Section 6.3, we will show that checking that the probability measure $Prob((s^1,s^2),(\psi'_k\ \mathcal{U}\ \psi''_k))$ is $\trianglelefteq p$, where (s^1,s^2) is a state of a bidimensional Boucherie process, is equivalent to checking that the probability measure $Prob(s^k,(\psi'_k\ \mathcal{U}\ \psi''_k))$ is $\trianglelefteq p'$, where p' is a derived probability value. This means that the probability measure of the paths satisfying $(\psi'_k\ \mathcal{U}\ \psi''_k)$ with respect to the product-process, is related to the probability measure of the paths of component $\mathcal{M}_k$ which satisfy the same Until formula $(\psi'_k\ \mathcal{U}\ \psi''_k)$. The problem with this approach is that the evaluation of the equivalent probability bound p' would require both a method that, given a state s of a CTMC, returns the set of paths $Path(s,(\psi'_k\ \mathcal{U}\ \psi''_k))$ and a method that, given a path σ_k over component $\mathcal{M}_k$, returns the corresponding set of paths of the product-process, which map on to σ_k. Both those methods, seem to rely on graph analysis techniques, something which we still need to investigate.

The following section is devoted to the analysis of the basics properties relating paths of the product process and paths of the components' process in a bidimensional Boucherie framework.

6.2 Paths in a bidimensional Boucherie process

In any graph-like structure (Transition System, Markov Chain, ...) a path is an infinite sequence of states $\sigma = s_0, s_1, \ldots, s_n, \ldots$ such that each state s_i in the sequence, is connected to its successor s_{i+1} by an arc. In a Markov Chain such an arc reflects the fact that for the state s_i the probability of reaching its successor is greater than zero (see Definition 2.2.4). Furthermore, with respect to CTMCs, paths can be *timed* by interleaving the sequence of states with a sequence of time intervals $I_0, I_1, \ldots, I_n, \ldots$

[1]In this work *single-component* unbounded Until formulae only are treated.

(see Definition 2.3.3).

$$\sigma \equiv s_0, I_0, s_1, I_1, \ldots, I_{n-1}, s_n, \ldots$$

We remember that by *untimed generator* $\bar{\sigma}$ of a timed-path, we mean the sequence of states (i.e. the untimed path) obtained by elimination of the interleaved time intervals, I_i, from the timed path σ, or, alternatively, by assuming every time interval $I_i = [0, \infty)$ $(i \geq 0)$.

$$\bar{\sigma} \equiv s_0, s_1, \ldots, , s_n, \ldots$$

For the time being we concentrate on untimed paths only. In Section 4.3 it has been pointed out that transitions in a bidimensional Boucherie process, can be classified either as 1-*move*s or 2-*move*s, according to the component which is involved[2]. Hence a transition $(s^1, s^2) \rightarrow (t^1, t^2)$ will be a 1-*move* whenever the state change involves component $\mathcal{M}_1$ (i.e. $s^1 \neq t^1$, $s^2 = t^2$), or a 2-*move*, if it reflects a state change regarding component $\mathcal{M}_2$ (i.e. $s^1 = t^1$, $s^2 \neq t^2$). As a result a path $\sigma = (s_0^1, s_0^2), (s_1^1, s_1^2), \ldots, (s_n^1, s_n^2), \ldots$ over a two component Boucherie state-space corresponds to an interleaved sequence of 1-*move*s and 2-*move*s. In the remainder we will refer to paths of a two component Boucherie process, as *bidimensional paths* or *Boucherie paths*.

Intuitively the k-projection of a *bidimensional path* σ, is given by the contraction of σ with respect to the *j-move*s (i.e. the path obtained by eliminating every *j-move* from the sequence it is made of). Thus every *bidimensional path* σ, is associated with a pair of projections, namely its 1-projection and 2-projection[3].

In this section a formal definition of the k-projection of a *Boucherie path* σ, is given and three basic results are proved. The first and trivial one, states that the k-projection of a path σ, formally defined by means of a function named $Proj_k()$, is actually a proper path over component $\mathcal{M}_k$. The formal proof of this intuitively obvious result (see Proposition B.0.3) relies on a number of intermediate results which are treated in the Appendix B. The second and more relevant result, states that the probability measure of a path σ can be factored by means of the probability measure of its two projections. Finally it will be shown that every path σ of a bidimensional Boucherie

[2]In a Boucherie framework synchronisation is not permitted, hence only one component at a time can change its state.

[3]Note that either one projection or the other can be the empty path ε but not both unless the considered σ is itself the empty path.

process, satisfies a *single-component* unbounded Until formula, like $\varphi_k \equiv (\psi'_k \; \mathcal{U} \; \psi''_k)$, if and only if its k-projection $Proj_k(\sigma)$ does. These results are useful in the study of a compositional semantics for unbounded Until formulae with respect to the Boucherie framework.

In the following some notations and remarks regarding generic paths and bidimensional paths are introduced. Let σ be a (generic) path, then:

- ε : is the empty path
- $(\sigma \uparrow n)$: is the prefix of σ up to the n-th element (inclusive).
- $(n \uparrow \sigma)$: is the suffix of σ starting at the n-th element (inclusive).

where $n \in \mathbb{N}$. Generally speaking a path is an infinite sequence of states. However finite paths are used to represent a set of infinite paths characterised by a common prefix.

Remark 6.2.1 *The n-th prefix of a path σ represents the set of all paths which have $\sigma \uparrow n$ as a common prefix.*

From now on, unless explicitly specified, σ will denote a finite sequence

$$\sigma = s_0 \rightarrow s_1 \rightarrow \ldots \rightarrow s_n$$

representing the set of infinite paths which have $s_0 \rightarrow s_1 \rightarrow \ldots \rightarrow s_n$ as common prefix.

Definition 6.2.1 *The length of a (finite) path σ is given by the number of transitions it consists of:*

$$length(s_0 \rightarrow s_1 \rightarrow \ldots \rightarrow s_n) = n$$

Given a (finite) path σ we adopt the following conventions:

- $\sigma[0]$ is the first element of σ while $\sigma[length(\sigma)]$ is the last one.
- $\forall n \in [0, length(\sigma) - 1]$, $\sigma[n]$ is called the *predecessor* of $\sigma[n+1]$, while $\sigma[n+1]$ is called the *successor* of $\sigma[n]$.
- $\forall m, n \in [0, length(\sigma)]$ with $m < n$, we will say that $\sigma[m]$ *precedes* $\sigma[n]$, while $\sigma[n]$ *follows* $\sigma[m]$.

It should be noted that a sequence consisting of a single state, for example $\sigma = s_0$, is a proper path even if it contains no transitions at all. Since a finite path σ is meant to represents the set of all infinite paths which have σ as common prefix, then $\sigma = s_0$ represents the set of all infinite paths starting at s_0. Hence the probability measure of a *single element* path is clearly $Pr(\sigma = s_0) = 1$, as it is given by the sum of the probability measure of every path starting at s_0.

Definition 6.2.2 ($Path(s, \psi' \, \mathcal{U} \, \psi'')$) *Let $s \in S$ be a state of a labelled CTMC $\mathcal{M} = (S, \mathbf{Q}, L)$ and ψ', ψ'' two non-probabilistic formulae as in (4.3.1). $Path(s, \psi' \, \mathcal{U} \, \psi'')$ denotes the set of paths starting at s and satisfying the unbounded Until formula $(\psi' \, \mathcal{U} \, \psi'')$.*

Remark 6.2.2 ($Path((s^1, s^2), \psi' \, \mathcal{U} \, \psi'')$) *Definition 6.2.2 naturally applies to a bidimensional Boucherie process $\mathcal{M} = (S, \mathbf{Q}, L)$, hence $Path((s^1, s^2), \psi' \, \mathcal{U} \, \psi'')$ denotes the set of paths starting at (s^1, s^2) and satisfying $(\psi' \, \mathcal{U} \, \psi'')$ in $\mathcal{M}$, (s^1, s^2) being a state of the Boucherie process.*

In the remainder we will use $\sigma[n]^k$ to denote the k-th component of the n-th state of a Boucherie path σ. Thus, if we consider the following bidimensional path

$$\sigma = (s^1, s^2)(s^1, t^2)(t^1, t^2)(t^1, u^2)(t^1, v^2) \tag{6.2.1}$$

then, for example,

$$\sigma[0]^1 = s^1 \quad \sigma[3]^2 = u^2$$

Moreover, as we have already pointed out, each bidimensional path corresponds to an interleaved sequence of 1-*move*s and 2-*moves*, hence σ maps on the following sequence:

$$2\text{-}move, 1\text{-}move, 2\text{-}move, 2\text{-}move$$

Definition 6.2.3 (Number of *k-move*s in a bidimensional path) *Let σ be a bidimensional path, then $k\_steps(\sigma \uparrow n)$ denotes the number of k-moves contained in the n-th prefix of σ. If σ is finite then $k\_steps(\sigma)$ stands for $k\_steps(\sigma \uparrow length(\sigma))$.*

Fact 6.2.1 (Length of a bidimensional path) *The length of a bidimensional path σ is equal to the sum of the k_steps and j_steps it consists of.*

$$length(\sigma) = k\_steps(\sigma) + j\_steps(\sigma)$$

Fact 6.2.1 is trivially true as any transition in a bidimensional Boucherie process can either be a 1-*move* or a 2-*move*. Therefore it is obvious that the length of any finite sequence of such transitions has to be equal to the sum of 1-*move*s and 2-*move*s it consists of. For example, the path σ shown in (6.2.1) consists of four transitions (i.e. $length(\sigma) = 4$), three of which are 2-*move*s (i.e. $2\_steps(\sigma) = 3$) while the remaining one is a 1-*move* (i.e. $1\_steps(\sigma) = 1$).

Definition 6.2.4 (*k*-path) *A bidimensional path σ is called a k-path if it consists only of k-moves, which is, $j\_steps(\sigma) = 0$.*

For example the bidimensional path $\sigma = (s^1, s^2)(s^1, t^2)(s^1, u^2)(s^1, v^2)$ is a 2-*path* as contains no 1-*moves*[4].

Fact 6.2.2 *The k-component of the target and source state of a j-move taken from a bidimensional path σ, is constant.*

$$\sigma[n] \rightarrow \sigma[n+1] \in j\text{-}move \Longrightarrow \sigma[n]^k = \sigma[n+1]^k$$

$\forall n \in [0, length(\sigma) - 1]$

Fact 6.2.2 is a trivial consequence of the definition of Boucherie product process.

Definition 6.2.5 (*k*-projection of a bidimensional path) *Let σ be a bidimensional (timed) path with respect to a Boucherie process $\mathcal{M}$, then its k-projection $Proj_k(\sigma)$, or simply*

[4]Again we point out that we are referring to CTMCs with no self-loops, hence a transition like $(s^1, s^2) \rightarrow (s^1, t^2)$ can only represents a 2-*move*.

σ^k*, is defined as:*

$$Proj_k(\sigma) \equiv \sigma^k = \begin{cases} \sigma[0]^k.Proj_k(1 \uparrow \sigma) & \textbf{if } \sigma[0] \to \sigma[1] \in k\text{-}moves \\ \\ Proj_k(1 \uparrow \sigma) & \textbf{if } \sigma[0] \to \sigma[1] \in j\text{-}moves \\ \\ \sigma[0]^k & \textbf{if } (1 \uparrow \sigma) \equiv \varepsilon \\ \\ \varepsilon & \textbf{if } \sigma \equiv \varepsilon \end{cases}$$

Next an example of a bidimensional path σ is considered and its two projections $Proj_1(\sigma)$ and $Proj_2(\sigma)$ are computed.

Example 6.2.1 *Let us consider the following bidimensional path* σ

$$\sigma = (s^1, s^2)(s^1, t^2)(t^1, t^2)(u^1, t^2)(u^1, u^2)(v^1, u^2)$$

and let us compute its projections by means of the function $Proj_k(\sigma)$ *described in Definition 6.2.5. As a first thing we notice that* σ *maps on the following sequence of transitions*

$$2\text{-}move, 1\text{-}move, 1\text{-}move, 2\text{-}move, 1\text{-}move$$

which also means that $1\text{-}steps(\sigma) = 3$*, while* $2\text{-}steps(\sigma) = 2$*.*

The iterative application of $Proj_k()$ *to* σ *with respect to both components* $\mathcal{M}_1$ *(i.e.*

$Proj_1(\sigma)$) and $\mathcal{M}_2$ (i.e. $Proj_2(\sigma)$), results in:

$$\begin{aligned}
Proj_1(\sigma) &= Proj_1((s^1,s^2)(s^1,t^2)(t^1,t^2)(u^1,t^2)(u^1,u^2)(v^1,u^2)) \\
&= Proj_1((s^1,t^2)(t^1,t^2)(u^1,t^2)(u^1,u^2)(v^1,u^2)) \\
&= s^1.Proj_1((t^1,t^2)(u^1,t^2)(u^1,u^2)(v^1,u^2)) \\
&= s^1.t^1.Proj_1((u^1,t^2)(u^1,u^2)(v^1,u^2)) \\
&= s^1.t^1.Proj_1((u^1,u^2)(v^1,u^2)) \\
&= s^1.t^1.u^1.Proj_1((v^1,u^2)) \\
&= s^1.t^1.u^1.v^1
\end{aligned}$$

$$\begin{aligned}
Proj_2(\sigma) &= Proj_2((s^1,s^2)(s^1,t^2)(t^1,t^2)(u^1,t^2)(u^1,u^2)(v^1,u^2)) \\
&= s^2.Proj_2((s^1,t^2)(t^1,t^2)(u^1,t^2)(u^1,u^2)(v^1,u^2)) \\
&= s^2.Proj_2((t^1,t^2)(u^1,t^2)(u^1,u^2)(v^1,u^2)) \\
&= s^2.Proj_2((u^1,t^2)(u^1,u^2)(v^1,u^2)) \\
&= s^2.t^2.Proj_2((u^1,u^2)(v^1,u^2)) \\
&= s^2.t^2.Proj_2((v^1,u^2)) \\
&= s^2.t^2.u^2
\end{aligned}$$

Figure 6.1 depicts how the states of the bidimensional path σ are mapped onto the projections' paths. It should be noted that the n-th state of σ, $\sigma[n]$, is mapped, with respect to the 1-projection (2-projection) on the same element on which its predecessor, $\sigma[n-1]$, is mapped, whenever the transition $\sigma[n-1] \rightarrow \sigma[n]$ is a 2-move (1-move). On the other hand the element on which $\sigma[n]$ is mapped, with respect to the 1-projection (2-projection), is the successor of the element on which $\sigma[n-1]$ is mapped, whenever $\sigma[n-1] \rightarrow \sigma[n]$ is a 1-move (2-move).

$\square$

The next fact points out an evident consequence of the definition of k-projection of a bidimensional path.

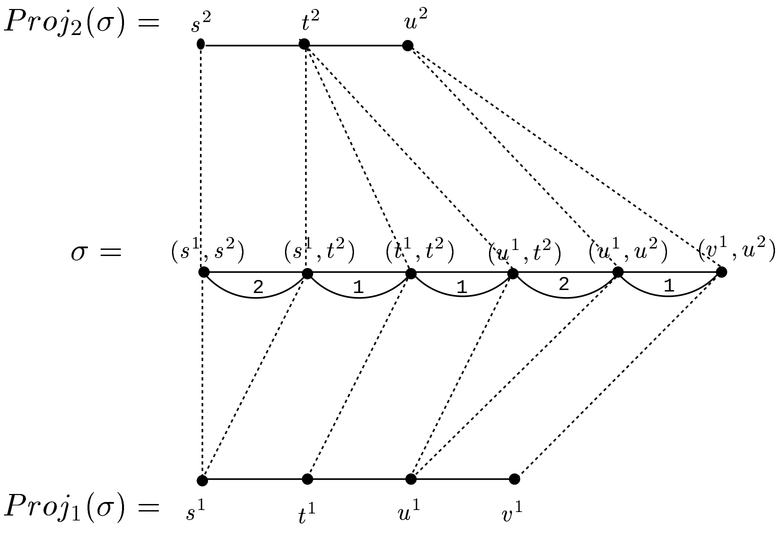


Figure 6.1: The two projections of a bidimensional path σ

Fact 6.2.3 (The k-projection is a k-path) *The k-projection of a bidimensional path σ is a path with respect to component $\mathcal{M}_k$ starting at state $\sigma[0]^k$.*

$$Proj_k(\sigma) \in Path(\sigma[0]^k)$$

Fact 6.2.3 trivially relies on the definition of k-projection of a bidimensional path as well as on the definition of Boucherie process. In fact since $Proj_k(\sigma)$ is obtained by elimination of the j-moves from σ, the result is clearly a sequence of k-*move*s which, according to the definition of Boucherie process, is also a proper sequence of moves, hence a path, over component $\mathcal{M}_k$. The formal proof of the result stated in Fact 6.2.3 can be found, in the Appendix B.

The following two facts point out two other trivial consequences of the definition of k-projection of a bidimensional path.

Fact 6.2.4 (Length of the k-projection of a bidimensional path) *The length of the i-projection of a path σ is equal to the number of i-steps in σ.*

$$length(Proj_i(\sigma)) = i\_steps(\sigma)$$

Fact 6.2.4 is also trivially true. From Fact 6.2.3, we know that $Proj_k(\sigma)$ is a path starting at $\sigma[0]^k$, hence it has a length. Furthermore since $Proj_i(\sigma)$ is made of the k-moves of σ, then clearly its length is given by the number of k_steps in σ.

Remark 6.2.3 (Length of a bidimensional path) *The length of a bidimensional path σ is equal to the sum of the length of its k and j projections:*

$$length(\sigma) = length(Proj_k(\sigma)) + length(Proj_j(\sigma))$$

So far the idea of k-projection of a bidimensional path has been introduced and formal means to derive it have been provided (see Definition 6.2.5). Moreover, it has been shown that the k-projection of a path of a bidimensional Boucherie process is a proper path over the component $\mathcal{M}_k$.

We now aim to show that the *probability measure* of a bidimensional path σ can be expressed in terms of the *probability measure* of its *projections* σ^1 and σ^2. Such a property will be the basis for showing the existence of a compositional semantics for probabilistic *path* formulae $\mathcal{P}_{\trianglelefteq p}(\varphi)$. In order to do that, we first need to prove another property concerning bidimensional paths. The following proposition states that the k-projection of a bidimensional path σ can be split in two parts: the first one being given by the k-projection of its n-th prefix and the second being obtained by elimination of the first element from the k-projection of its n-th suffix.

Proposition 6.2.1 (Splitting the k-projection of a path) *Let σ be a bidimensional path consisting of $l = length(\sigma) > 0$ transitions and $0 \leq n \leq l$. The k-projection of σ can be split in terms of the k-projection of its n-th prefix and suffix, in the following manner:*

$$Proj_k(\sigma) = Proj_k(\sigma \uparrow n).(1 \uparrow (Proj_k(n \uparrow \sigma)))$$

Proof.

By induction on $n \in [0, l)$. Let σ be

$$\sigma = (s_0^1, s_0^2), \dots, (s_{n-1}^1, s_{n-1}^2), (s_n^1, s_n^2), \dots (s_l^1, s_l^2)$$

base case: $n = 0$.

In this case $(\sigma \uparrow n) = (s_0^1, s_0^2)$ hence $Proj_k(\sigma \uparrow n) = s_0^k$. Furthermore $(n \uparrow \sigma) = \sigma$, thus $Proj_k(n \uparrow \sigma) = Proj_k(\sigma)$ hence $(1 \uparrow (n \uparrow \sigma)) = 1 \uparrow \sigma$. We need to distinguish between two cases:

i. $(s_0^1, s_0^2) \to (s_1^1, s_1^2) \in$ *k-move*.

In this case, from Definition 6.2.5, we have that $Proj_k(\sigma) = s_0^k.Proj_k(1 \uparrow \sigma)$, hence here we aim to prove that

$$Proj_k(\sigma \uparrow 0).(1 \uparrow (Proj_k(0 \uparrow \sigma))) = s_0^k.Proj_k(1 \uparrow \sigma)$$

As we have already noticed

$$(1 \uparrow (Proj_k(0 \uparrow \sigma))) = 1 \uparrow Proj_k(\sigma) = (1 \uparrow (s_0^k.Proj_k(1 \uparrow \sigma))) = Proj_k(1 \uparrow \sigma)$$

Thus the above equality is proved.

ii. $(s_0^1, s_0^2) \to (s_1^1, s_1^2) \in$ *j-move*.

In this case a further distinction is needed. If σ is a *j_path* (i.e. it consists only of *j-moves*), then $Proj_k(\sigma) = s_l^k = s_0^k$. Also, since $(n \uparrow \sigma) = \sigma$ with $n = 0$ then $(1 \uparrow (Proj_k(n \uparrow \sigma))) = \varepsilon$, hence

$$Proj_k(\sigma \uparrow n).(1 \uparrow (Proj_k(n \uparrow \sigma))) = s_0^k = Proj_k(\sigma).$$

On the other hand if σ is not a *j_path* then there exists an index $n' \in [1, l)$ such that $(s_{n'}^1, s_{n'}^2) \to (s_{n'+1}^1, s_{n'+1}^2) \in$ *k-move* and $\forall \hat{n}' \in [1, n')$, $(s_{\hat{n}'}^1, s_{\hat{n}'}^2) \to (s_{\hat{n}'+1}^1, s_{\hat{n}'+1}^2) \in$ *j-move*. In this case, from Definition 6.2.5, it is straightforward to show that

$$Proj_k(\sigma) = Proj_k(1 \uparrow \sigma) = s_{n'}^k.Proj_k((n'+1) \uparrow \sigma)$$

where also $s_0^k = s_{n'}^k$.Furthermore, since with $n = 0$

$$(1 \uparrow (Proj_k(0 \uparrow \sigma))) = 1 \uparrow Proj_k(\sigma)$$

then

$$(1 \uparrow (Proj_k(0 \uparrow \sigma))) = Proj_k((n'+1) \uparrow \sigma)$$

Hence

$$(\sigma \uparrow n).(1 \uparrow (Proj_k(0 \uparrow \sigma))) = s_0^k.Proj_k((n'+1) \uparrow \sigma) = Proj_k(\sigma)$$

which proves the base of the induction also in this second case.

inductive step: $0 < n < l$.
We aim to show that

$$Proj_k(\sigma) = Proj_k(\sigma \uparrow (n+1)).(1 \uparrow (Proj_k((n+1) \uparrow \sigma)))$$

assuming

$$Proj_k(\sigma) = Proj_k(\sigma \uparrow n).(1 \uparrow (Proj_k(n \uparrow \sigma)))$$

as *inductive hypothesis*. As before, we need to distinguish between two cases.
i. $(s_n^1, s_n^2) \rightarrow (s_{n+1}^1, s_{n+1}^2) \in$ *j-move*.
In this case, as a consequence of Definition 6.2.5, we have that $Proj_k(\sigma \uparrow n) = Proj_k(\sigma \uparrow (n+1))$. Furthermore, $Proj_k(n \uparrow \sigma) = Proj_k((n+1) \uparrow \sigma)$, hence

$$Proj_k(\sigma \uparrow n).(1 \uparrow (Proj_k(n \uparrow \sigma))) = Proj_k(\sigma \uparrow (n+1)).(1 \uparrow (Proj_k((n+1) \uparrow \sigma)))$$

which, as a consequence of the *inductive hyptosis*, proves that:

$$Proj_k(\sigma) = Proj_k(\sigma \uparrow (n+1)).(1 \uparrow (Proj_k((n+1) \uparrow \sigma)))$$

ii. $(s_n^1, s_n^2) \rightarrow (s_{n+1}^1, s_{n+1}^2) \in$ *k-move*.
In this case from Definition 6.2.5, it is easy to show that

$$Proj_k(\sigma \uparrow n).s_{n+1}^k = Proj_k(\sigma \uparrow (n+1)) \tag{6.2.2}$$

and also $Proj_k(n \uparrow \sigma) = s_n^k.Proj_k((n+1) \uparrow \sigma)$ which implies:

$$(1 \uparrow Proj_k(n \uparrow \sigma)) = Proj_k((n+1) \uparrow \sigma) \tag{6.2.3}$$

Trivially though, we can rewrite $Proj_k((n+1) \uparrow \sigma)$ as the concatenation of its first element to its first suffix, namely:

$$Proj_k((n+1) \uparrow \sigma) = Proj_k((n+1) \uparrow \sigma)[0].(1 \uparrow Proj_k((n+1) \uparrow \sigma))$$

Straightforwardly, $Proj_k((n+1)\uparrow\sigma)[0]$ turns out to be equal to s^k_{n+1}. To prove that a further distinction needs to be considered: if $(1\uparrow Proj_k((n+1)\uparrow\sigma))$ is a *j_path*, then clearly $Proj_k((n+1)\uparrow\sigma) = s^k_l = s^k_{n+1}$; on the other hand if $(1\uparrow Proj_k((n+1)\uparrow\sigma))$ is not a *j_path*, then there will exists an index $n' \in [n+1,l)$ such that $\sigma[n'] \rightarrow \sigma[n'+1] \in$ *k-move* and $\forall \hat{n}' \in [n+1,n')$, $\sigma[\hat{n}'] \rightarrow \sigma[\hat{n}'+1] \in$ *j-move*, hence $Proj_k((n+1)\uparrow\sigma) = s^k_{n'} = s^k_{\hat{n}'}$. which proves $Proj_k((n+1)\uparrow\sigma) = s^k_{n+1}$, in this case too. Relying on this and on (6.2.2) and (6.2.3) we have that:

$$\begin{aligned}
Proj_k(\sigma\uparrow(n+1)).(1\uparrow Proj_k(n\uparrow\sigma)) &= Proj_k(\sigma\uparrow n).s^k_{n+1}.(1\uparrow Proj_k((n+1)\uparrow\sigma)) \\
&= Proj_k(\sigma\uparrow n).Proj_k((n+1)\uparrow\sigma) \\
&= Proj_k(\sigma\uparrow n).(1\uparrow Proj_k(n\uparrow\sigma))
\end{aligned}$$

which, as a consequence of the *inductive hypothesis* proves that:

$$Proj_k(\sigma) = Proj_k(\sigma\uparrow(n+1)).(1\uparrow Proj_k((n+1)\uparrow\sigma))$$

□

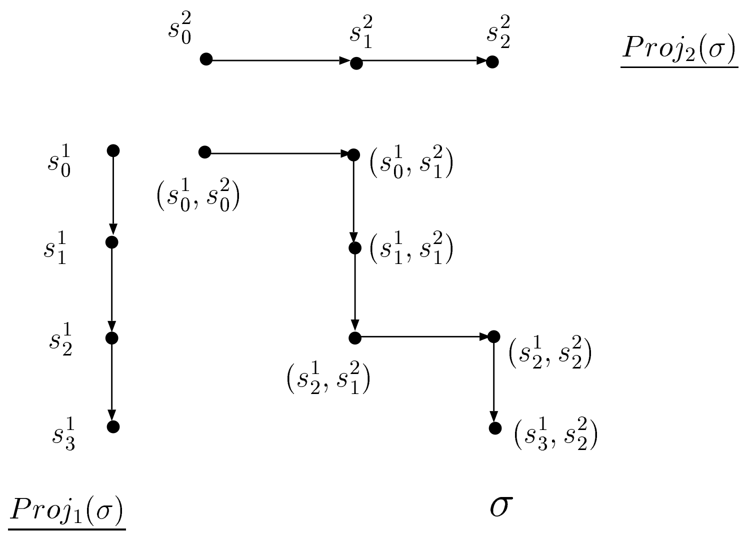


Figure 6.2: Splitting $Proj_k(\sigma)$

To understand the meaning of the result of the above proposition, let us consider the bidimensional path

$$\sigma = (s_0^1, s_0^2), (s_0^1, s_1^2), (s_1^1, s_1^2), (s_2^1, s_1^2), (s_2^1, s_2^2), (s_3^1, s_2^2)$$

which is depicted in Figure 6.2, together with its projections:

$$\begin{aligned} Proj_1(\sigma) &= s_0^1, s_1^1, s_2^1, s_3^1 \\ Proj_2(\sigma) &= s_0^2, s_1^2, s_2^2 \end{aligned}$$

Suppose we are concerned about the 1-projection of σ, namely $Proj_1(\sigma)$, then, according to Proposition 6.2.1 we can randomly choose a splitting point n in $\{0, \ldots, l\}$ where l is the length of σ, i.e. $l = 5$. For example, if we choose $n = 0$ as splitting point (i.e. a splitting point corresponding to a 2-*move*), then the n-th prefix and suffix of σ are respectively: $(\sigma \uparrow n) = (\sigma \uparrow 0) = (s_0^1, s_0^2)$ and $(n \uparrow \sigma) = (0 \uparrow \sigma) = \sigma$. Proposition 6.2.1 tells us that

$$Proj_1(\sigma) = Proj_1(\sigma \uparrow 0).(1 \uparrow Proj_1(0 \uparrow \sigma))$$

Let us verify if this is the case. Trivially $Proj_1(\sigma \uparrow 0) = s_0^1$. Furthermore $Proj_1(0 \uparrow \sigma) = Proj_1(\sigma) = s_0^1, s_1^1, s_2^1, s_3^1$, which by elimination of the head element becomes

$$(1 \uparrow Proj_1(0 \uparrow \sigma)) = s_1^1, s_2^1, s_3^1$$

Then, clearly, the concatenation of $Proj_1(\sigma \uparrow 0) = s_0^1$ and $(1 \uparrow Proj_1(0 \uparrow \sigma)) = s_1^1, s_2^1, s_3^1$ leads to $Proj_1(\sigma) = s_0^1, s_1^1, s_2^1, s_3^1$. If instead we consider $n = 4$ as splitting point, which corresponds to a 1-*move* (i.e. $\sigma[4] \to \sigma[5] \in$ 1-*move*), the n-th prefix and suffix are:

$$\begin{aligned} (\sigma \uparrow n) &= (\sigma \uparrow 4) = (s_0^1, s_0^2), (s_0^1, s_1^2), (s_1^1, s_1^2), (s_2^1, s_1^2), (s_2^1, s_2^2) \\ (n \uparrow \sigma) &= (4 \uparrow \sigma) = (s_2^1, s_2^2), (s_3^1, s_2^2) \end{aligned}$$

and their 1-projections are:

$$\begin{aligned} Proj_1(\sigma \uparrow 4) &= s_0^1, s_1^1, s_2^1 \\ Proj_1(4 \uparrow \sigma) &= s_2^1, s_3^1 \end{aligned}$$

from which, straightforwardly, $Proj_1(\sigma) = Proj_1(\sigma \uparrow 4).(1 \uparrow Proj_1(4 \uparrow \sigma))$.

This proves that the *splitting rule* for $Proj_k(\sigma)$ works properly both when the chosen splitting point corresponds to a *j-move* or to a *k-move*.

The ability to split the k-projection of a *bidimensional path* σ is a basic result for proving that the probability measure of σ can be factored in terms of the probability measure of its projections.

Given a bidimensional path σ, we introduce the further notations:

- *k move*(σ): represents the set of indices $m \in [0, length(\sigma))$ corresponding to a *k-move* in σ, namely such that $\sigma[m] \to \sigma[m+1]$ is a *k-move*.

- p^k_σ: is a constant, accounting for the probability of the *k-move*s of σ. It is defined as follows:

$$p^k_\sigma = \begin{cases} \Pi_{m \in k\text{-}move(\sigma)}[p^k(\sigma[m])] & \textbf{if } n > 0 \text{ and } k\text{-}move(\sigma) \neq \emptyset \\ 1 & \text{otherwise} \end{cases}$$

  where $p^k(\sigma[m])$ represents the probability for a *k-move* to occur when at state $\sigma[m]$ (see Definition 4.2.1).

In essence the coefficient p^k_σ is proportional to the measure of the probability of the *k-move*s in a bidimensional path σ.

Proposition 6.2.2 (Factors of the probability measure of a bidimensional path) *Let σ be a bidimensional path. The probability measure of σ is expressible in terms of the probability measure of its 1-projection and 2-projection, in the following way:*

$$Pr(\sigma) = p^1_\sigma \cdot p^2_\sigma \cdot Pr(Proj_1(\sigma)) \cdot Pr(Proj_2(\sigma))$$

where the constant p^k_σ is as described above.

Proof.

By induction on $n = length(\sigma)$. Let σ be:

$$\sigma = (s^1_0, s^2_0) \ldots (s^1_n, s^2_n)$$

base: $n = 0$.

In this case $\sigma = (s^1_0, s^2_0)$ hence, trivially, $Pr(\sigma) = 1$. Moreover $Proj_1(\sigma) = s^1_0$ and $Proj_2(\sigma) = s^2_0$ then clearly $Pr(Proj_1(\sigma)) = Pr(Proj_2(\sigma)) = 1$ and also $p^1_\sigma = p^2_\sigma = 1$ which proves

$$Pr(\sigma) = p^1_\sigma \cdot p^2_\sigma \cdot Pr(Proj_1(\sigma)) \cdot Pr(Proj_2(\sigma)).$$

induction: $n > 0$.

Trivially, we know that the probability measure of a path is given by the product of the probability measure of its m-th prefix and suffix, hence, with respect to the $(n-1)$-th prefix and suffix of σ:

$$Pr(\sigma) = Pr(\sigma \uparrow (n-1)) \cdot Pr((s^1_{n-1}, s^2_{n-1})(s^1_n, s^2_n)) \tag{6.2.4}$$

Since the length of the $(n-1)$-th prefix of σ is $n-1$, then by *inductive hypothesis*, we know that:

$$Pr(\sigma \uparrow (n-1))) = p^1_{(\sigma\uparrow(n-1))} \cdot Pr(Proj_1(\sigma \uparrow (n-1))) \cdot p^2_{(\sigma\uparrow(n-1))} \cdot Pr(Proj_2(\sigma \uparrow (n-1)))$$

By substituting this result in 6.2.4 we get:

$$\begin{aligned} Pr(\sigma) = p^1_{(\sigma\uparrow(n-1))} \cdot Pr(Proj_1(\sigma \uparrow (n-1))) \cdot p^2_{(\sigma\uparrow(n-1))} \cdot Pr(Proj_2(\sigma \uparrow (n-1))) \\ \cdot Pr((s^1_{n-1}, s^2_{n-1})(s^1_n, s^2_n)). \end{aligned} \tag{6.2.5}$$

We then need to distinguish between two possibilities:

i. $(s^1_{n-1}, s^2_{n-1}) \to (s^1_n, s^2_n) \in$ 1-*move*.

In this case $Proj_1((n-1) \uparrow \sigma) = s^1_{n-1}.s^1_n$, hence

$$Pr((s^1_{n-1}, s^2_{n-1})(s^1_n, s^2_n)) = p^1((s^1_{n-1}, s^2_{n-1})) \cdot Pr(s^1_{n-1}, s^1_n)$$

where $p^1((s^1_{n-1}, s^2_{n-1}))$ is as in Definition 4.2.1. Then by substitution in 6.2.5 we get:

$$\begin{aligned} Pr(\sigma) = p^1_{(\sigma\uparrow(n-1))} \cdot Pr(Proj_1(\sigma \uparrow (n-1))) \cdot p^2_{(\sigma\uparrow(n-1))} \cdot Pr(Proj_2(\sigma \uparrow (n-1))) \\ \cdot p^1((s^1_{n-1}, s^2_{n-1})) \cdot Pr(s^1_{n-1}, s^1_n). \end{aligned} \tag{6.2.6}$$

Moreover, from Proposition 6.2.1 we know that $Proj_k(\sigma)$ can be split in two parts, particularly:

$$Proj_1(\sigma) = Proj_1(\sigma \uparrow (n-1)).(1 \uparrow Proj_1((n-1) \uparrow \sigma))$$
$$Proj_2(\sigma) = Proj_2(\sigma \uparrow (n-1)).(1 \uparrow Proj_2((n-1) \uparrow \sigma))$$

Since we are assuming $(s^1_{n-1}, s^2_{n-1}) \to (s^1_n, s^2_n) \in 1\text{-}move$, then

$$\begin{aligned} Proj_1((n-1) \uparrow \sigma) &= s^1_{n-1}.s^1_n \\ Proj_2((n-1) \uparrow \sigma) &= s^2_n \end{aligned}$$

hence

$$Proj_1(\sigma) = Proj_1(\sigma \uparrow (n-1)).(1 \uparrow (s^1_{n-1}.s^1_n)) = Proj_1(\sigma \uparrow (n-1)).s^1_n$$
$$Proj_2(\sigma) = Proj_2(\sigma \uparrow (n-1)).(1 \uparrow s^2_n) = Proj_2(\sigma \uparrow (n-1))$$

then:

$$Pr(Proj_1(\sigma)) = Pr(Proj_1(\sigma \uparrow (n-1))).Pr(s^1_{n-1}, s^1_n)$$
$$Pr(Proj_2(\sigma)) = Pr(Proj_2(\sigma \uparrow (n-1)))$$

By substituting the above results in 6.2.6, we get

$$Pr(\sigma) = p^1_{(\sigma\uparrow(n-1))} p^2_{(\sigma\uparrow(n-1))} Pr(Proj_1(\sigma)) Pr(Proj_2(\sigma)) \cdot p^1((s^1_{n-1}, s^2_{n-1})) \tag{6.2.7}$$

but since we are assuming $(s^1_{n-1}, s^2_{n-1}) \to (s^1_n, s^2_n) \in 1\text{-}move$, then also

$$\begin{aligned} p^1_\sigma &= p^1_{(\sigma\uparrow(n-1))} \cdot p^1((s^1_{n-1}, s^2_{n-1})) \\ p^2_\sigma &= p^2_{(\sigma\uparrow(n-1))} \end{aligned}$$

which substituted in (6.2.7) proves

$$Pr(\sigma) = p^1_\sigma p^2_\sigma Pr(Proj_1(\sigma)) Pr(Proj_2(\sigma))$$

ii. $(s^1_{n-1}, s^2_{n-1}) \to (s^1_n, s^2_n) \in 2\text{-}move$.
The proof of this case is symmetric to the one of the previous case.

$\square$

The above proposition provides us with a compositional method to compute the probability measure of a bidimensional Boucherie path. Given such a path σ, the value of its probability measure is given by the product of the probability measure of its projections multiplied by two constants, the values of which depend on the transitions σ is made up of. The following example shows how to apply the above result in practice.

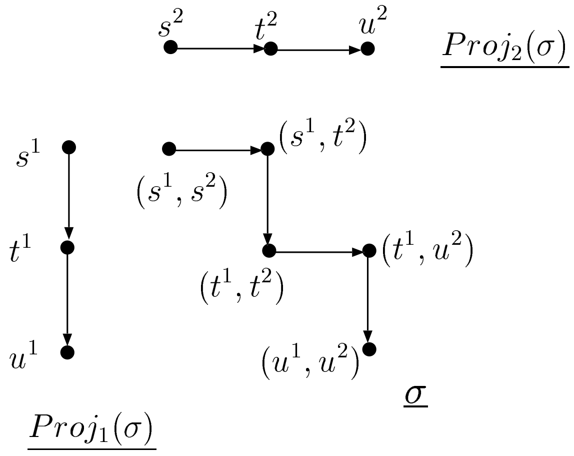


Figure 6.3: factorizing the probability measure of a path σ

Example 6.2.2 *Figure 6.3 shows a path σ together with its projections $Proj_1(\sigma)$ and $Proj_2(\sigma)$. The probability measure of σ is given by the product of the probability of each step, hence:*

$$Pr(\sigma)=\frac{Q_2(s^2,t^2)}{E_1(s^1)+E_2(s^2)}\cdot\frac{Q_1(s^1,t^1)}{E_1(s^1)+E_2(t^2)}\cdot\frac{Q_2(t^2,u^2)}{E_1(t^1)+E_2(t^2)}\cdot\frac{Q_1(t^1,u^1)}{E_1(t^1)+E_2(u^2)} \quad (6.2.8)$$

Having in mind that the probability of a k-move from a state $(\hat{s}^1,\hat{s}^2)$, is given by:

$$p^k(\hat{s}^1,\hat{s}^2)=\frac{E_k(\hat{s}^k)}{E_1(\hat{s}^1)+E_2(\hat{s}^2)}$$

then we can rewrite (6.2.8) as follows:

$$\begin{aligned}Pr(\sigma)=p^2((s^1,s^2))\cdot\frac{Q_2(s^2,t^2)}{E_2(s^2)}\cdot p^1((s^1,t^2))\cdot\frac{Q_1(s^1,t^1)}{E_1(s^1)}\cdot\\ p^2((t^1,t^2))\cdot\frac{Q_2(t^2,u^2)}{E_2(t^2)}\cdot p^1((t^1,u^2))\cdot\frac{Q_1(t^1,u^1)}{E_1(t^1)}\end{aligned}$$

On the other hand, the probability measure of the σ *projections are respectively*

$$Pr(Proj_1(\sigma)) = \frac{Q_1(s^1,t^1)}{E_1(s^1)} \cdot \frac{Q_1(t^1,u^1)}{E_1(t^1)}$$
$$Pr(Proj_2(\sigma)) = \frac{Q_2(s^2,t^2)}{E_2(s^2)} \cdot \frac{Q_1(t^2,u^2)}{E_2(t^2)}$$

Hence, since $p^1_\sigma = p^1((s^1,t^2)) \cdot p^1((t^1,u^2))$ *and* $p^2_\sigma = p^2((s^1,s^2)) \cdot p^2((t^1,t^2))$*, we have that:*

$$Pr(\sigma) = p^1_\sigma \cdot p^2_\sigma \cdot Pr(Proj_1(\sigma)) \cdot Pr(Proj_2(\sigma))$$

proving the validity of Proposition 6.2.2 in this case.

□

Finally, we are going to show a basic result which regards the semantics of *single-component* unbounded Until formulae with respect to bidimensional paths: a bidimensional path satisfies a formula like $(\psi'_k \,\mathcal{U}\, \psi''_k)$ if and only if, its k-projection does.

Proposition 6.2.3 *Let* σ *be a bidimensional path with respect to a bidimensional Boucherie process* $\mathcal{M}$ *and* ψ'_k, ψ''_k *two single-component formulae as in (5.2.1). Then the following holds:*

$$\sigma \models (\psi'_k \,\mathcal{U}\, \psi''_k) \Longleftrightarrow Proj_k(\sigma) \models_k (\psi'_k \,\mathcal{U}\, \psi''_k)$$

Proof.

$(\Rightarrow)$ From the CSL semantics we know that if $\sigma \models (\psi'_k \;\mathcal{U}\; \psi''_k)$ then $\exists n \geq 0 : \sigma[n] \models \psi''_k$ and $\forall m < n\ \sigma[m] \models \psi'_k$. From the decomposed semantics of non-probabilistic *single-component* formulae, we know that

$$\sigma[m] \models \psi'_k \Longleftrightarrow \sigma[m]^k \models_k \psi'_k$$
$$\sigma[n] \models \psi''_k \Longleftrightarrow \sigma[n]_k \models_k \psi''_k$$

Moreover, thanks to Remark **??** (see Appendix B), we know that the n-th element of σ is projected over the $map_k(\sigma,n)$-th of its k-projection. Hence $\exists n' = map_k(\sigma,n) \geq 0$ such that $Proj_i(\sigma)[n'] = \sigma[n]^k \models_k \psi''_k$ and also $\forall m' = map_k(\sigma,m) < n'$, $Proj_k(\sigma)[m'] = \sigma[m]^k \models_k \psi'_k$ which proves $Proj_k(\sigma) \models (\psi'_k \;\mathcal{U}\; \psi''_k)$.

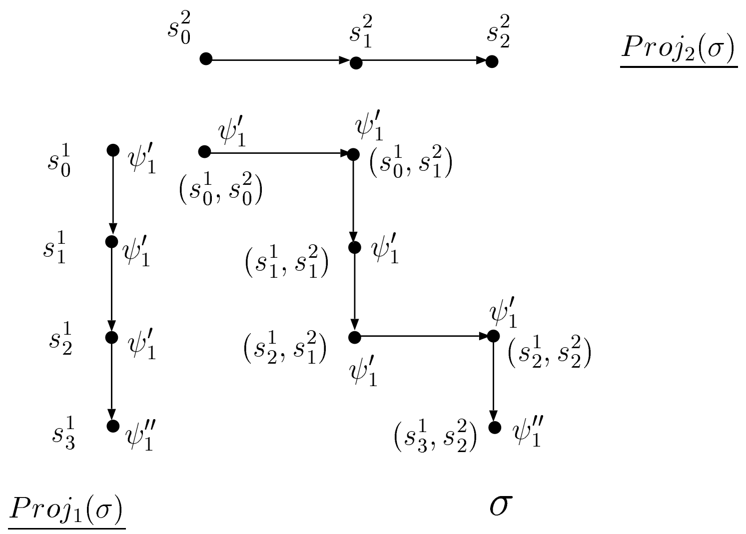


Figure 6.4: Semantics of $(\psi_1' \; \mathcal{U} \; \psi_1'')$ with respect to a bidimensional path

$(\Leftarrow)$ Straightforward by reversing the argument in $(\Rightarrow)$.

□

In Figure 6.4 an example of the property proved by Proposition 6.2.3 is depicted. The bidimensional path σ satisfies $(\psi_1' \; \mathcal{U} \; \psi_1'')$ and also, clearly, its projection on component $\mathcal{M}_1$, σ^1, satisfies $(\psi_1' \; \mathcal{U} \; \psi_1'')$.

A direct consequence of the above result is the one proved in the following proposition. It states that the k-projection of the paths which start from a state (s^1, s^2) and satisfy the formula $(\psi_k' \mathcal{U} \; \psi_k'')$ is equal to the set of paths from state s^k satisfying $\psi_i' \; \mathcal{U} \; \psi_i''$.

Proposition 6.2.4 *Let (s^1, s^2) be a a state of a bidimensional Boucherie process $\mathcal{M}$ and ψ_k', ψ_k'' two single-component formulae as in (5.2.1). Then the following holds:*

$$Proj_k(Path((s^1, s^2), \psi_k' \; \mathcal{U} \; \psi_k'')) = Path(s^k, \psi_k' \; \mathcal{U} \; \psi_k'')$$

Proof.

$(\Rightarrow)$ If $\sigma^k \in Proj_k(Path((s^1, s^2), \psi_k' \; \mathcal{U} \; \psi_k''))$, then from Proposition 6.2.3 also $\sigma^k \models_k (\psi_k' \; \mathcal{U} \; \psi_k'')$, and since clearly $\sigma^k[0] = s^k$, then $\sigma^k \in Path(s^k, \psi_k' \; \mathcal{U} \; \psi_k'')$.

($\Leftarrow$) Let us consider an arbitrary path $\sigma_k \in Path(s^k, \psi'_k \; \mathcal{U} \; \psi''_k)$. We aim to show that for each such a path σ_k, there exists a bidimensional path from (s^1, s^2), which as well satisfies $(\psi'_k \; \mathcal{U} \; \psi''_k)$ and whose k-projection is actually σ_k. This is equivalent to show that the subset of $Path((s^1, s^2), \psi'_k \; \mathcal{U} \; \psi''_k)$ given by the paths whose k-projection is σ_k, namely $Proj_k^{-1}(\sigma_k, ((s^1, s^2), \psi'_k \; \mathcal{U} \; \psi''_k))$, is not empty. In order to do that we need to distinguish between three possible cases, which are:

i) All states in σ_k fall in $S_{k,\overline{R}}$. For simplicity, in the remainder of the proof we refer our argument to the case $k = 1$ hence $j = 2$. We observe that a symmetrical and equivalent derivation can be straightforwardly obtained from the above by swapping the indices k and j. If all states of σ_k are in $S_{k,\overline{R}}$ and $s^j \in S_{j,\overline{R}}$, then $\sigma = (\sigma_k \times s^j) = (\sigma_k[0], s^j) \ldots (\sigma_k[n], s^j)$, where $n = length(\sigma_k)$, is clearly a path from (s^1, s^2) and furthermore, as a consequence of the decomposed semantics of ψ_k formulae, also $\sigma \models (\psi'_k \; \mathcal{U} \; \psi''_k)$. Hence $\sigma \in (Path((s^1, s^2), \psi'_k \; \mathcal{U} \; \psi''_k))$ and also, trivially, $Proj_k(\sigma) = \sigma_k$ which proves $Proj_k^{-1}(\sigma_k, (s^1, s^2), \psi'_k \; \mathcal{U} \; \psi''_k)$ to be non empty.

On the other hand if $s^j \in S_{j,R}$, then since we are dealing here only with ergodic Markov Chains (i.e. any state is reachable from any other state), there will be a path $\bar{\sigma} = s^j \ldots t^j$ on $\mathcal{M}_j$ leading from s^j to a state $t^j \in S_{j,\overline{R}}$. Thus, clearly, $(s^k \times \bar{\sigma})$ is a path on $\mathcal{M}$ from (s^k, s^j) and furthermore $(s^k \times \bar{\sigma}).\sigma \in Path((s^1, s^2), \psi'_k \; \mathcal{U} \; \psi''_k))$ and $Proj_k((s^k \times \bar{\sigma}).\sigma) = \sigma_k$ which again proves $Proj_k^{-1}(\sigma_k, (s^1, s^2), \psi'_k \; \mathcal{U} \; \psi''_k)$ to be non empty.

ii) All states in σ_k are in $S_{k,R}$. In this case s^j must be in $S_{j,\overline{R}}$ (as $S_{k,R} \times S_{j,R}$ is not part of the Boucherie state space). Then $(\sigma_k \times s^j)$ is a path from (s^1, s^2) and, thanks to decomposed semantics of ψ_k formulae, we also know that $(\sigma_k \times s^j)$ satisfies $(\psi'_k \; \mathcal{U} \; \psi''_k)$, while clearly its k-projection is σ_k.

iii) Some states of σ_k are in $S_{k,\overline{R}}$ and some others are in $S_{k,R}$. In the most general case the path σ_1 we consider (we are assuming $k = 1$), is given by a (finite) sequence of $\sigma_{1_{\overline{R}}}$ and σ_{1_R} subpaths (subpaths consisting only of states in $S_{1,\overline{R}}$ and $S_{1,R}$ respectively). However we can consider the simplest case of σ_1 consisting of a sequence of only two such subpaths, either $\sigma_1 = \sigma_{1_{\overline{R}}}.\sigma_{1_R}$ or $\sigma_1 = \sigma_{1_R}.\sigma_{1_{\overline{R}}}$, knowing that the proof for any

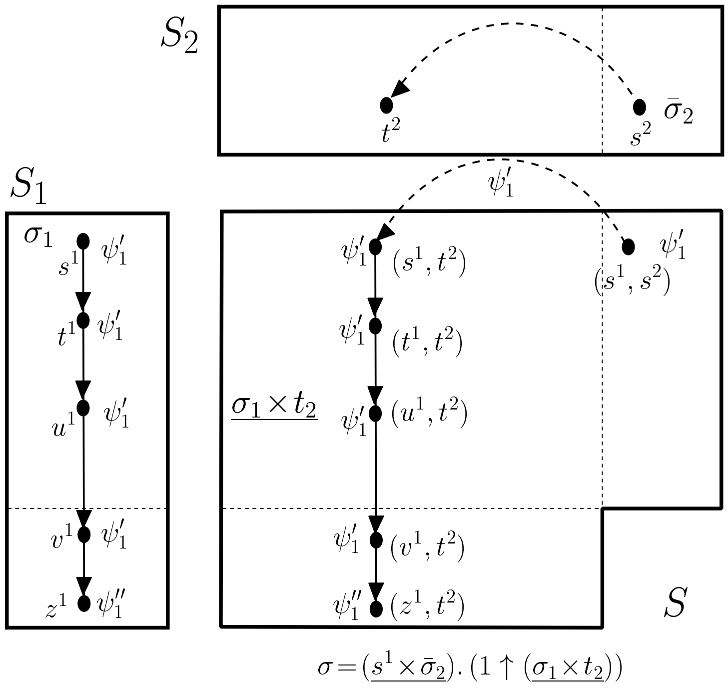


Figure 6.5: $Proj_k(Path((s^1, s^2), \psi_k' \ \mathcal{U} \ \psi_k'')) = Path(s^k, \psi_k' \ \mathcal{U} \ \psi_k'')$

other case is a direct consequence of this simple one. Let us assume

$$\sigma_1 = \sigma_{1_{\overline{R}}}.\sigma_{1_R}$$

Let us denote by $k \in [0, l)$ ($l = length(\sigma_1)$) the last element of $\sigma_{1_{\overline{R}}}$ (i.e. $k = length(\sigma_{1_{\overline{R}}})$). Then clearly, $\sigma_1[k] \in S_{1,\overline{R}}$ and $\sigma[k+1] \in S_{1,R}$. As we have seen in the previous case, if $s^2 \in S_{2,\overline{R}}$ then the bidimensional path given by the product of σ_1 by s^2, $\sigma_a = \sigma_1 \times s^2$, is a proper path from (s^1, s^2). On the other hand (see Figure 6.5), if $s^2 \in S_{2,R}$ there will be a state $t^2 \in S_{2,\overline{R}}$ reachable from s^2 through a path $\bar{\sigma}^2$ such that the bidimensional path obtained by concatenation of the product paths $(\sigma_1[0] \times \bar{\sigma}^2)$ and $((1 \uparrow \sigma_{1_{\overline{R}}}) \times t^2)$, i.e.

$$\sigma_b = (\sigma[0]_1 \times \bar{\sigma}^2).((1 \uparrow \sigma_{1_{\overline{R}}}) \times t^2)$$

is a proper path from the state (s^1, s^2) to the state $(\sigma_1[k], t^2)$.

It is straightforward to show that both σ_a and σ_b are paths from (s^1, s^2) which *project* on σ_1. Moreover since, by the hypothesis, σ_1 satisfies $(\psi_k' \ \mathcal{U} \ \psi_k'')$ then there exists $n' \in [0, l]$ such that $\sigma_1[n'] \models_1 \psi_1''$ and $\forall m' < n'$, $\sigma_1[m'] \models_1 \psi_1'$. Thus, if $s^2 \in S_{2,\overline{R}}$

then thanks to the compositional semantics of ψ_k formulae, also there will exist $n = n'$ such that, with $\sigma = \sigma_a$, $\sigma[n] \models \psi''_1$ and $\forall m = m' < n = n'$ $\sigma[m] \models \psi'_1$, which proves σ_a being in $Path((s^1, s^2), (\psi'_k\ \mathcal{U}\ \psi''_k))$ (hence $Path((s^1, s^2), (\psi'_k\ \mathcal{U}\ \psi''_k)) \neq \emptyset$).

If, instead, $s^2 \in S_{2,R}$, then the path $\sigma = \sigma_b . (\sigma 1_{\overline{R}} \times t^2)$ is such that with $n = n' + length(\bar{\sigma}^2)$, $\sigma[n] \models \psi''_1$ and $\forall m < n$ $\sigma[m] \models \psi'_1$, which means $\sigma \models (\psi'_k\ \mathcal{U}\ \psi''_k)$. Hence, also in the case $s^2 \in S_{2,R}$, the set $Path((s^1, s^2), (\psi'_k\ \mathcal{U}\ \psi''_k))$ is not empty as it contains, at least, $\sigma = \sigma_b . (\sigma 1_{\overline{R}} \times t^2)$.

The proof for the symmetrical case $\sigma_1 = \sigma 1_R . \sigma 1_{\overline{R}}$ is simpler as clearly the only possibility, in this case, is $s^2 \in S_{2,\overline{R}}$ (we are considering a path σ_1 generated from a state $\sigma_1[0] \in S_{1,R}$ where component $\mathcal{M}_1$ holds the shared resource, hence s^1 can only be coupled with states $s^2 \in S_{2,\overline{R}}$). In this case the product path $\sigma = \sigma_1 \times s^2$ is clearly a path from (s^1, s^2) projecting on σ_1. Furthermore it is straightforward to show that, by assuming $\sigma_1 \models_k (\psi'_k\ \mathcal{U}\ \psi''_k)$ then also $\sigma \models (\psi'_k\ \mathcal{U}\ \psi''_k)$.

□

The result of the above proposition relating bidimensional paths and their k-projections, will be the basis for a non-constructive prove of existence of a compositional semantics for *single-component* unbounded Until formulae. This will be the subject of the next section.

6.3 On *single-component* unbounded Until formulae

Relying on the background material about bidimensional paths provided in the previous section, we are now ready to deal with the analysis of a compositional semantics for unbounded Until formulae which refer to a bidimensional Boucherie process. For that purpose, we consider an enriched version of the syntax for *single-component* formulae introduced in (5.2.1), where the Until connective $\mathcal{U}^I$, in its unbounded version (i.e. $I = [0, \infty)$) only, has been added.

$$
\begin{aligned}
\phi_k &::= \psi_k \mid \varphi_k \mid \omega_k \mid \xi_k \mid \phi_k \wedge \phi_k \mid \neg\phi_k \\
\psi_k &::= tt \mid a_k \mid \psi_k \wedge \psi_k \mid \neg\psi_k \\
\xi_k &::= \mathcal{S}_{\trianglelefteq p}(\psi_k) \mid \mathcal{S}_{\trianglelefteq p}(\varphi_k) \\
\varphi_k &::= \mathcal{P}_{\trianglelefteq p}(X^I(\psi_k)) \\
\omega_k &::= \mathcal{P}_{\trianglelefteq p}(\psi_k \,\mathcal{U}\, \psi_k)
\end{aligned}
\tag{6.3.1}
$$

We notice that the same type of restrictions concerning nesting of probabilistic operators (see Section 5.2) still apply. Thus, also for the Until connective, only non-probabilistic *single-component* formulae ψ_k, are admitted as the possible type of argument. Furthermore, Until formulae are kept apart from Next formulae. This is due to the fact that, for the time being, no results have been found showing a compositional semantics of *steady-state* properties which refer to Until paths (while, as shown in Section 5.2.2, there is a decomposed way to check *steady-state* properties of bounded Next *single-component* formulae). As a consequence, the Until formulae cannot appear as argument of the *steady-state* operator, in the syntax of formulae for which a decomposed semantics exists.

The following theorem proves that, relying on the properties of bidimensional paths, the derivation of a compositional semantics for *single-component* unbounded Until formulae is possible.

Theorem 6.3.1 *Let (s^1,s^2) be a state of a bidimensional Boucherie process, ψ'_k, ψ''_k two non-probabilistic formulae as in (6.3.1), $p \in [0,1]$ a probability value and $\trianglelefteq \in \{<,\leq,\geq,>\}$. Then there exists a derived probability value p' such that the following holds:*

$$(s^1,s^2) \models \mathcal{P}_{\trianglelefteq p}(\psi'_k \;\mathcal{U}\; \psi''_k) \Longleftrightarrow s^k \models_k \mathcal{P}_{\trianglelefteq p'}(\psi'_k \;\mathcal{U}\; \psi''_k)$$

Proof. From the CSL semantics we know that a state s of an arbitrary CTMC $\mathcal{M}$ satisfies a probabilistic Until formula like $\mathcal{P}_{\trianglelefteq p}(\psi'_k \;\mathcal{U}\; \psi''_k)$ if and only if the probability measure $Prob^{\mathcal{M}}(s,(\psi'_k \,\mathcal{U}\, \psi''_k)) \trianglelefteq p$. However, $Prob^{\mathcal{M}}(s,(\psi'_k \,\mathcal{U}\, \psi''_k))$ is equal to the sum of the probability measure of the paths starting from s and satisfying $(\psi'_k \,\mathcal{U}\, \psi''_k)$, which is:

$$Prob^{\mathcal{M}}(s,(\psi'_k \,\mathcal{U}\, \psi''_k)) = \sum_{\sigma \in Path(s,(\psi'_k \,\mathcal{U}\, \psi''_k))} Pr(\sigma)$$

where the probability of a path σ is given by the product of the probability of each transition it consists of. Relying on this remark let us prove the two implications.

$(\Rightarrow)$ If $(s^1,s^2) \models \mathcal{P}_{\trianglelefteq p}(\psi'_i \,\mathcal{U}\, \psi''_i)$. Then

$$\Big[\sum_{\sigma \in Path((s^1,s^2),\psi'_i \,\mathcal{U}\, \psi''_i)} Pr(\sigma) \Big] \trianglelefteq p \tag{6.3.2}$$

If we denote by P' the set of k-projections of the paths $\sigma \in Path((s^1,s^2),\psi'_i \,\mathcal{U}\, \psi''_i)$ (i.e. $P' = Proj_k(Path((s^1,s^2),\psi'_i \,\mathcal{U}\, \psi''_i)))$ and by $Proj_k^{-1}(\sigma',(s^1,s^2))$ the set of paths σ from (s^1,s^2) whose k-projection is σ' (i.e. the path σ such that $Proj_k(\sigma) = \sigma'$), then we can rewrite the above sum by factoring out the paths with common k-projection in the following way:

$$\sum_{\sigma \in Path((s^1,s^2),\psi'_k \,\mathcal{U}\, \psi''_k)} Pr(\sigma) = \sum_{\sigma' \in P'} \Big(\sum_{\sigma \in Proj_k^{-1}(\sigma',(s^1,s^2))} Pr(\sigma) \Big) \tag{6.3.3}$$

From Proposition 6.2.2 we know that the probability measure of every bidimensional path σ can be factored in terms of the probability measure of its projections $Proj_1(\sigma)$ and $Proj_2(\sigma)$ and of two constants, namely p^1_σ and p^2_σ:

$$\forall \sigma, \exists p^i_\sigma, p^j_\sigma \ : \ Pr(\sigma) = p^1_\sigma \cdot p^2_\sigma \cdot Pr(Proj_1(\sigma)) \cdot Pr(Proj_2(\sigma))$$

Hence (6.3.3) results in:

$$\begin{aligned} \sum_{\sigma \in Path((s^1,s^2),\psi'_k \,\mathcal{U}\, \psi''_k)} Pr(\sigma) &= \sum_{\sigma' \in P'} \Big(\sum_{\sigma \in Proj_k^{-1}(\sigma',(s^1,s^2))} p^i_\sigma p^j_\sigma Pr(\sigma') Pr(Proj_j(\sigma))) \Big) \\ &= \sum_{\sigma' \in Proj_k(Path((s^1,s^2),\psi'_k \mathcal{U} \psi''_k))} k_{\sigma'} Pr(\sigma') \trianglelefteq p \end{aligned} \tag{6.3.4}$$

with

$$k_{\sigma'} = \sum_{\sigma \in Proj_i^{-1}(\sigma',(s^1,s^2))} p^i_\sigma p^j_\sigma Pr(Proj_j(\sigma)) \tag{6.3.5}$$

But from Proposition 6.2.4, we know that

$$Proj_k(Path((s^1,s^2),\psi'_k \mathcal{U} \psi''_k)) = Path(s^k,(\psi'_k \ \mathcal{U}\ \psi''_k))$$

which substituted in the inequality (6.3.4) results in:

$$\sum_{\sigma' \in Path(s^k,(\psi'_k \ \mathcal{U}\ \psi''_k))} k_{\sigma'} Pr(\sigma') \trianglelefteq p \tag{6.3.6}$$

If we denote by σ'_M the path in $Path(s^k,(\psi'_k \ \mathcal{U}\ \psi''_k))$ which maximises the associated constant $k_{\sigma'_M}$, then we can define the constant C which accounts for the total deviation from the maximum $k_{\sigma'_M}$ of each other path $\sigma' \in Path(s^k,(\psi'_k \ \mathcal{U}\ \psi''_k))$, $\sigma' \neq \sigma'_M$, namely:

$$C = \sum_{\substack{\sigma' \in Path(s^k,(\psi'_k \ \mathcal{U}\ \psi''_k)) \\ \sigma' \neq \sigma'_M}} [k_{\sigma'_m} - k_{\sigma'}] Pr(\sigma')$$

Then from (6.3.6), straightforwardly follows

$$k_{\sigma'_M} \sum_{\sigma' \in Path(s^k,(\psi'_k \ \mathcal{U}\ \psi''_k))} Pr(\sigma') \trianglelefteq (p + C)$$

which, clearly, proves

$$s^k \models_k \mathcal{P}_{\trianglelefteq \frac{(p+C)}{k_{\sigma'_M}}} (\psi'_k \ \mathcal{U}\ \psi''_k)$$

($\Leftarrow$) By reversing ($\Rightarrow$).

$\square$

Relying on the properties of bidimensional paths, the above theorem proves the existence of a compositional semantics for single-component unbounded Until formulae. This result, though correct, does not provide a practical way of decomposed checking of Until formulae. The computation of the equivalent probability bound p' , in fact, requires determing the set of paths which satisfy an Until formula (i.e $Path(s^k,(\psi'_k \ \mathcal{U}\ \psi''_k))$), something which could be achieved by defining some specific graph-analysis technique. In the next section we will show that a decomposed verification of event-bounded Until formulae is easily achievable.

6.4 Compositional Semantics of event-bounded Until formulae

In Section 3.2 an event-bounded version of the Until operator has been formally introduced and a verification method has been demonstrated for the case of a single-point bounding interval. It has been shown that verifying an event-bounded formula $(\psi' \; U_{\{n\}} \; \psi'')$ boils down to the verification of its operands, ψ' and ψ'', plus the verification of the Next formula $(X \; \psi'')$. In Chapter 4 and Chapter 5, we have proved methods for decomposed checking of both simple boolean combinations of atomic propositions (i.e. ψ_kk and ψ_{12}) and Next formulae (i.e. $X\psi_k$ and $X\psi_{12}$). As a consequence, the derivation of an algorithm for compositional verification of event-bounded Until formulae, both single-component and general, is straightforward. In Algorithm 6.4.1 such a method is provided. That procedure requires the computation of the state-vectors $\underline{Prob}(X\psi_k)$ and $\underline{Prob}(X\psi_{12})$ whose elements represent the probability of satisfying, respectively, a single-component and a general Next formula, for the states of a bidimensional Boucherie CTMC. The following two remarks point out that the elements of those vectors can be evaluated in a compositional way (i.e. by computing the probability of Next formulae with respect to the states of the component's processes).

Remark 6.4.1 *Let (s^1,s^2) be a state of a bidimensional Boucherie process and ψ_k a boolean combination of atomic propositions referring to component $\mathcal{M}_k$. The probability of reaching a ψ_k-state from (s^1,s^2) in one-step, is equal to:*

$$\underline{Prob}(X\;\psi_k)(s_1,s_2) = \begin{cases} \underline{Prob}_k(X\;\psi_k)(s^k) & \textbf{if } (s^1,s^2)\in R_k \\ 1 & \textbf{if } (s^1,s^2)\in R_j \\ & \quad \textit{and } s^k \models_k \psi_k \\ 0 & \textbf{if } (s^1,s^2)\in R_j \\ & \quad \textit{and } s^k \not\models_k \psi_k \\ p^k(s^1,s^2)\cdot \underline{Prob}_k(X\;\psi_k)(s^k) & \textbf{if } (s^1,s^2)\in R_{free} \\ & \quad \textit{and } s^k \not\models_k \psi_k \\ p^k(s^1,s^2)\cdot \underline{Prob}_k(X\;\psi_k)(s^k)+p^j(s^1,s^2) & \textbf{if } (s^1,s^2)\in R_{free} \\ & \quad \textit{and } s^k \models_k \psi_k \end{cases}$$

The above remark shows that the probability vector $\underline{Prob}(X\psi_k)$ for a bidimensional Boucherie process is a function of the corresponding probability vector $\underline{Prob}_k(X\psi_k)$ computed with respect to component $\mathcal{M}_k$.

Remark 6.4.2 *Let s^1 and s^2 be a state of a bidimensional Boucherie process and ψ_{12} a general formula as in 5.4.1. The probability of reaching a ψ_{12}-state from (s^1,s^2) in one-step, is equal to:*

$$\underline{Prob}(X\psi_{12})(s^1,s^2) = \sum_{(\alpha_1,\alpha_2)\in DecSat(\psi_{12})} [p^1(s^1,s^2)\cdot\underline{Prob}_1(X\,\alpha_1)(s^1) + p^2(s^1,s^2)\cdot\underline{Prob}_2(X\,\alpha_2)(s^2)]$$

The above remark is a direct consequence of Algorithm 5.4.2. It shows that the statevector $\underline{Prob}(X\psi_{12})$, for a bidimensional Boucherie process, can be obtained as a function of the components' state-vectors $\underline{Prob}_1(X\alpha_1)$ and $\underline{Prob}(X\alpha_2)$, where α_1 and α_2 are the single-component formulae characterising the partition of $Sat(\psi_{12})$.

Having shown that the vectors $\underline{Prob}(X\psi_k)$ and $\underline{Prob}(X\psi_{12})$ for a bidimensional Boucherie CTMC can be computed in a compositional manner, it is easy to determine a method for decomposed verification of both single-component and general event-bounded Until formulae.

Algorithm 6.4.1 *Let $\mathcal{M}$ be a bidimensional Boucherie process with components $\mathcal{M}_1$ and $\mathcal{M}_2$, ψ' and ψ'' two boolean combinations of atomic propositions (either single-component or general) and n a natural number. The following algorithm can be applied for computing the state-vector $\underline{P\mathcal{U}}=\underline{Prob}(\psi'\ \mathcal{U}_{\{n\}}\ \psi'')$*

Algorithm ($\underline{Prob}(\psi'\ \mathcal{U}_{\{n\}}\ \psi'')$).

1. $\underline{P\mathcal{U}}=\underline{0}$;

2. IF $n=0$ **THEN** $\underline{P\mathcal{U}}=\underline{i}_{\psi''}$; **ELSE**

  i. Determine $\underline{Prob}(X\ \psi'')$;

  ii. Determine $\underline{i}_{\psi'}$;

  iii. $\underline{P\mathcal{U}}=\underline{i}_{\psi'}\cdot\underline{Prob}(X\ \psi'')$; $n=n-1$;

  iv. FOR $n>0$ **DO**

    $\underline{P\mathcal{U}}=\underline{i}_{\psi'}\cdot[\mathbf{P}\cdot\underline{P\mathcal{U}}]$;

$n = n - 1;$

3. return $\underline{P\mathcal{U}}$.

□

The above algorithm shows a method for computing the probability of satisfying an event-bounded Until formulae of any type (i.e. with any possible combination of operands, either *single-component* formulae, ψ_k, or general formulae, ψ_{12}) for a bidimensional Boucherie process. We observe that the evaluation of the state-vector $\underline{Prob}(\psi' \ \mathcal{U}_{\{n\}} \ \psi'')$, for the product-process, is obtained in a decomposed way which is, by means of a number of verifications involving the component's processes only (no actual verification with respect to the state-space of the product-process is needed). In fact, in order to determine $\underline{Prob}(\psi' \ \mathcal{U}_{\{n\}} \ \psi'')$ the vectors $\underline{i}_{\psi'}$, $\underline{i}_{\psi''}$ and $\underline{Prob}(X\psi'')$ are needed. However, relying on Theorem 4.3.1 and Theorem 4.3.2, we know that $\underline{i}_{\psi'}$ and $underline i_{\psi''}$ can be evaluated in a compositional way. Moreover, from Remark 6.4.1 and Remark 6.4.2, we know that also $\underline{Prob}(X\psi'')$ can be computed compositionally. Hence, the procedure illustrated by Algorithm 6.4.1 shows that an approach for decomposed verification of event-bounded Until formulae, is possible when referring to a bidimensional Boucherie process.

Chapter 7

Conclusion

7.1 Introduction

In this chapter a summary of the main results of the thesis is presented. The extent to which these address the analysis of CSL expressiveness and the study of a compositional approach to CTMC's model-checking is assessed. Furthermore, in Section 7.3, a description of the ongoing work and directions for further developments are provided.

7.2 Summary

In this work the model-checking technique for CMTCs has been considered and two major aspects have been addressed: the study of the expressiveness of the CSL logic and the analysis of a compositional method to check CSL formulae with respect to a bidimensional Boucherie process.

CSL expressiveness

In Chapter 3 the CSL logic has been considered and its expressiveness analysed. Relevant points have been made regarding each of the following subjects.

Extending the future-quantification: in referring to a behaviour of interest with respect to a system's evolution, the idea of *event-quantification*, as opposed to *time-*

quantification, of the future has been pointed out. In this respect, we have seen that the Until and Next operator perform differently, the former allowing us to refer to an *indefinitely long* (in terms of events) future only, the latter allowing a *one-event-long* only quantification of the future. The extension of the strict (*one-only*) *event-quantification* capability of the CSL logic to the most general case of n events, seemed then to be natural. As a result, an *event-bounded* version of the time-unbounded Until operator has been introduced and a method for its verification has been demonstrated. This has been shown to require the computation of an iterative matrix-vector multiplication, as opposed to the the solution of a system of linear equations which is needed for its event-unbounded counterpart (i.e. the standard unbounded Until).

Single-point time-bounded formulae: time-bounded path formulae (i.e. $X^I\ \phi$ and $\phi\,\mathcal{U}^I\ \psi$) allow us to specify a bounding interval $I = [a,b]$ for the *time-wise distance* of a future behaviour of interest. When the bounding interval consists of a single instant (i.e. $I = [a,a]$) we have pointed out some relevant features of both Next and Until. Concerning a Next formulae, (i.e. $X^{[a,a]}\ \phi$), it has been shown that reaching a ϕ state in one step exactly at time t is an impossible event (i.e. it has probability zero to happen). As a consequence a time-bounded Next formula has been characterised as *well-formed* only if $a < b$. With the Until formulae (i.e. $(\phi\,\mathcal{U}^{[a,a]}\ \psi)$), instead, we have seen that only the evolutions in which a future state where both the *source* (i.e. ϕ) and the *target* (i.e. ψ) formulae are valid is reached through a sequence of states where the *source* is satisfied, have a non-null probability to fulfil the point time-bound $[a,a]$. In contrast, in the non-point interval case, $a < b$, also the paths in which a future state where only the *target* is valid is reached through a sequence of states where the *source* is satisfied, can have a non-zero probability to fulfil the time bound $I = [a,b]$.

Well-formed probabilistic formulae: CSL probabilistic formulae (i.e. $\mathcal{S}_{\trianglelefteq p}(\phi)$ and $\mathcal{P}_{\trianglelefteq p}(\varphi)$) allow us to compare a probability measure (either an *equilibrium probability* measure or a *path probability* measure) with respect to a bound p. The comparison is achieved by means of any operator $\trianglelefteq \in \{<,\leq,\geq,>\}$ and any bound $p \in [0,1]$. We have pointed out that the absence of restrictions in associating $\trianglelefteq$ and p can lead to senseless formulae, like $\mathcal{S}_{\geq 0}(\phi)$ or $\mathcal{P}_{\leq 1}(\varphi)$. Such formulae are trivially valid in any state, because, clearly, a probability measure must fall in the interval $[0,1]$. This has led to the characterisation of *well-formed* probabilistic CSL formulae, achieved by identifying the sensible pairs $(\trianglelefteq, p)$ through proper logical conditions.

Simpler syntax for ergodic CTMCs: the CSL syntax admits nesting of probabilistic operators (i.e. $\mathcal{S}_{\trianglelefteq p}$ and $\mathcal{P}_{\trianglelefteq p}$). This facility allows for expressing complicated properties of a CTMC. However, we have demonstrated that when the considered model is an ergodic CTMC, the complete nesting facility of the original CSL syntax is not actually needed. This is due to the fact that with ergodic CTMCs, *steady-state* formulae like $\mathcal{S}_{\trianglelefteq p}(\phi)$, are model-dependent rather than state-dependent (i.e. they are either valid in every state or in none). As a result, we have proved a number of equivalences which show that nesting of $\mathcal{S}_{\trianglelefteq p}$ within a $\mathcal{P}_{\trianglelefteq p}$ operator is pointless when dealing with ergodic CTMCs. That has led to the characterisation of a simpler, but equivalent, CSL syntax for referring to ergodic models.

Compositional CSL model-checking

Chapters 4, 5 and 6 have been devoted to the analysis of a compositional CSL semantics for bidimensional Boucherie CTMCs. Formulae referring to a two component Boucherie process have been partitioned into *single-component*, for stating properties which refer to features of a single component only, and *general*, which refer to features of both components. A progressive approach has been have adopted in deriving of a compositional semantics.

Non-path formulae: in Chapter 4 decomposed semantic equivalences for non-path formulae (i.e. formulae not involving X nor $\mathcal{U}$) have been proved, first for the *single-component* case, then, relying on those results, for the *general* case. With respect to properties which refer to a single component only, it has been proved that the model-checking problem for a *single-component steady-state* formula (i.e. $\mathcal{S}_{\trianglelefteq p}(\psi_k)$) on the product-process, is equivalent to the model-checking problem of the same *steady-state* formula on component $\mathcal{M}_k$, but with respect to a derived probability bound p' (whose value depends on $Sat_k(\psi_k)$). On the other hand, we have shown that checking of *general* formulae reduces to a combined checking of *single-component* formulae. In particular, a decomposed semantics for *general steady-state* formulae, like $\mathcal{S}_{\unlhd p}(\psi_{12})$, has been derived, relying on the definition of a partition of the set $Sat(\psi_{12})$. We observe that most of the decomposed semantic equivalences for *steady-state* properties (both *single-component* and *general*) we have proved, assume that the *normalisation* constant G, for the product form solution of the Boucherie *steady-state* distribution, is known. The problem of computing G (a well known one in the literature) has not been considered in this work.

Next formulae and new algorithm for $\underline{Prob}(X^I\ \phi)$: in Chapter 5 time-bounded Next formulae have been considered and a decomposed semantics has been provided. For the time being the possibility for nesting of the probabilistic Next operator (i.e. $\mathcal{P}_{\unlhd p}X^I$) has been excluded. Furthermore, since both the product process and the component's processes of a Boucherie framework are ergodic CTMCs, then only non-probabilistic formulae have been allowed as the possible type of argument of $\mathcal{P}_{\unlhd p}X^I$. Under these restrictions we have proved that checking a *single-component* probabilistic time-bounded Next formula, like $\mathcal{P}_{\unlhd p}(X^I\ \psi_k)$, on the product process, is equivalent to checking the same formula on component $\mathcal{M}_k$ but with respect to a derived probability bound p' and a derived time-bound I', in the worst case, and to verifying a simple inequality, in the best case. Furthermore, we have proved that, checking the *steady-state* probability of *single-component* probabilistic time-bounded Next against a bound p (i.e. formulae like $\mathcal{S}_{\unlhd p}(\mathcal{P}_{\overline{\unlhd}\overline{p}}(X^I\ \psi_K)))$, reduces to checking the *steady-state* probability of a derived *single-component* formula (i.e. $\mathcal{S}X_{low}(\psi_k, \overline{\unlhd}, \overline{p}, I)$ or $\mathcal{S}X_{up}(\psi_k, \overline{\unlhd}, \overline{p}, I)$) with respect to a derived probability bound p'_{low} (or p'_{up}). Finally, a decomposed method for checking *general* probabilistic time-bounded Next formulae has been provided. In this chapter another interesting result have been provided. We have demonstrated that the algorithm for computing the probability measure of satisfying a time-bounded Next formula (i.e. the state vector $\underline{Prob}(X^I\ \phi)$) which can be found in the literature, is not correct. A revised (and correct) version has been defined and its use has been shown with respect to an example.

Until formulae: in Chapter 6 the study of a decomposed way for checking *single-component* unbounded Until formulae has been faced. As for the Next operator, for the time being, only non-probabilistic formulae have been considered as the possible type of operand of a probabilistic until operator. Under this condition, we have shown that the verification of a formula $\mathcal{P}_{\trianglelefteq p}(\psi'_k \; \mathcal{U} \; \psi''_k)$ with respect to the Boucherie process, is equivalent to the verification of the same formula on component $\mathcal{M}_k$ but against a derived probability bound p' (i.e. $\mathcal{P}_{\trianglelefteq p'}(\psi'_k \; \mathcal{U} \; \psi''_k)$). Although correct, our argument relies on a *non-constructive* proof. Hence, it does not provide a practical decomposed method for checking those formulae. In order to obtain the equivalent probability bound p' one would need some method which, given a state s of a CTMC, returns the set of paths starting at s and satisfying $(\psi'_k \; \mathcal{U} \; \psi''_k)$.

About excluding nested path-operators. In our work on compositionality we have considered a strict restriction by disallowing nesting of path operators. In practice, we have demonstrated that a decomposed verification is possible only for the formulae belonging to that subset of the CSL. Assessing the extent to which this limits the ability to state properties of interest is relevant. A formal study of this problem will be the object of future work, however here some informal consideration is provided[1]. We observe that the practical relevance of formulae given, for instance, by nesting of a probabilistic Until operator within another probabilistic Until operator is not easily understandable. A formula like

$$\mathcal{P}_{\trianglelefteq p_1}(\phi \, \mathcal{U} \, (\mathcal{P}_{\trianglelefteq p_2}(\psi \; \mathcal{U} \, \xi)))$$

identifies all those states for which, with probability $\trianglelefteq p_1$, there exists a future state, reachable via ϕ states, at which, with probability $\trianglelefteq p_2$, a ξ state is reached through ψ states. The practical utility of such a formula, is far to be crystal clear. Similar considerations hold for the converse case, $\mathcal{P}_{\trianglelefteq p_1}((\mathcal{P}_{\trianglelefteq p_2}(\psi \; \mathcal{U} \, \xi)) \, \mathcal{U} \, \phi)$, and for nesting

[1] A complete analysis of the problem would require all the possible nesting combinations of Next and Until to be considered.

of Until and Next like, $\mathcal{P}_{\trianglelefteq p_1}(\phi\,\mathcal{U}\,(\mathcal{P}_{\trianglelefteq p_2}(X\ \psi)))$ or $\mathcal{P}_{\trianglelefteq p_2}(X\ (\mathcal{P}_{\trianglelefteq p_1}(\phi\,\mathcal{U}\,\psi)))$ even if the practical application of the latter case seems to have some relevance.

Abolishing the nesting facility, on the other hand, affects the ability to express *transient* properties. A formula like $\mathcal{P}_{\trianglelefteq p}(\diamond^{[t,t]}\ \phi)$, can be used to check the states' *transient* probability of matching ϕ at time t, against a bound p. If probabilistic path formulae cannot be used as argument of the $\diamond^{I}$ operator, the expressiveness for identifying the *target* states of interest, hence, the *transient* analysis capability, is reduced[2] .

7.3 Future work

At time of publication of this thesis several aspects of the research are still under process. In this section we provide directions for future developments of this work. In particular, we would like to point out that the (brand) new *event-bounded* Until operator, introduced in Chapter 3, has actually been a very recent "discovery". For this reason we have had no time enough to develop further material concerning it, even if there are many interesting aspects which originate from its definition. In the remainder a list of relevant points is provided.

Expressiveness analysis for nested probabilistic path-formulae. A formal analysis concerning the nesting facility for CSL path-formulae, is an interesting subject for further work. In this respect, it is relevant to work out what type of measures are expressible by means of nested path-properties and how sensible is to resort to that facility of the CSL syntax from a *performance analysis* point of view. Searching for the existence of useful equivalences, is also relevant.

Event-bounded Until. The *event-bounded* Until discloses the possibility for a new type of analysis of path properties. The results proved in Chapter 3 show that the model-checking problem for *event-bounded* Until actually reduces to the verification of

[2]We observe, however, that the use of probabilistic path formulae to identify the *target* state for a *transient* analysis, is not always sensible. A formula like, $\mathcal{P}_{\trianglelefteq p_1}(\diamond^{[t_1,t_1]}(\mathcal{P}_{\trianglelefteq p_2}(\diamond^{[t_2,t_2]}\ \phi))$, which nests a *transient* probability measure within another, appears to be reduceable to an equivalent, simple, *transient* analysis $\mathcal{P}_{\trianglelefteq p}(\diamond^{[}t,t]\ \phi)$, with $t = t_1 + t_2$ and $p = p_1 \cdot p_2$.

a Next property plus the calculation of an iterative matrix-vector product. In Chapter 5 we have demonstrated methods for decomposed checking of Next formulae referring to a bidimensional Boucherie framework. As a result the derivation of an algorithm for decomposed verification of *event-bounded* Until on a Boucherie process, seems to be straightforward. The definition of a combined *time/event-bounded* Until operator is also a relevant aspect which we are currently investigating.

Until decomposed semantics. Very recently we have started to consider the use of tensorial algebras to obtain a decomposed representation of the *infinitesimal generator matrix* of a bidimensional Boucherie process $\mathcal{M}$. We are currently studying if the application of such a decomposition in the Uniformisation method leads to a decomposed approximation of the *transient* distribution of $\mathcal{M}$. This would allow us to exploit the *correctness preserving transformations* described in Chapter 2, by means of which the model-checking problem for time-bounded Until with respect to a Boucherie process, would reduce to a *transient analysis* on the component's processes. .

Complexity. The analysis of the computational costs/savings resulting by application of the decomposed CSL verification (on a bidimensional Boucherie process) needs to be performed. With this respect, it is relevant to assess the computational cost for the function $DecSat()$ which, given a *general* non-probabilistic formula, ψ_{12}, returns a decomposed partition of $Sat(\psi_{12})$.

Appendix A

On the compositional semantics of Next formulae

In this Appendix three lemmas which have been mentioned in Section 5.2.2, are reported and proved. They regard properties which are essential to deriving a compositional semantics for *single-component steady-state* time-bounded Next formulae (i.e. formulae like $\mathcal{S}_{\trianglelefteq p}(\mathcal{P}_{\overline{\trianglelefteq p}}(X^I(\psi_k))))$.

Lemma A.0.1 *Let $\mathcal{M}$ be a bidimensional Boucherie process, ψ_k a non-probabilistic formula as in (5.2.1), $p \in [0,1]$ a probability bound, $\trianglelefteq \in \{<,\leq,\geq,>\}$ a comparison relation and $I = [a,b] \subseteq \mathbb{R}_{\geq 0}$ a time interval. Then a state $(s^1,s^2) \in S$ satisfies the formula $\mathcal{P}_{\trianglelefteq p}(X^I(\psi_k))$ if and only if s^k satisfies $\mathcal{S}X_{up}(\psi_k,\trianglelefteq,p,I)$*

$$(s^1,s^2) \models \mathcal{P}_{\trianglelefteq p}(X^I(\psi_k)) \Longleftrightarrow \begin{cases} s^k \models_k \mathcal{S}X_{low}(\psi_k,\trianglelefteq,p,I,s^j) \quad \textbf{if} \;\; low(\trianglelefteq,p)\wedge \\ \qquad\qquad \big[[s^j \in S_{j,R}] \rightarrow [(e^{-E_j(s^j)a} - e^{-E_j(s^j)b}) \trianglelefteq p]\big] \\ \\ s^k \models_k \mathcal{S}X_{up}(\psi_k,\trianglelefteq,p,I,s^j) \quad \textbf{if} \;\; up(\trianglelefteq,p) \end{cases} \tag{A.0.1}$$

Proof. For brevity we focus only on the first case of the bi-implication. The derivation of results for the case $up(\trianglelefteq,p)$, is similar.

($\Rightarrow$) Let us show that if $(s^1,s^2) \models \mathcal{P}_{\trianglelefteq p}(X^I(\psi_k))$ then $s^k \models_k \mathcal{S}X_{low}(\psi_k,\trianglelefteq,p,I,s^j)$, given that $low(\trianglelefteq,p)$ and $\big[[s^j \in S_{j,R}] \rightarrow [(e^{-E_j(s^j)a} - e^{-E_j(s^j)b}) \trianglelefteq p]\big]$. We have to distinguish between two possibilities which are: $(s^1,s^2) \in R_j$, (i.e. $\mathcal{M}_j$ holds the shared resource) and $(s^1,s^2) \notin R_j$ (i.e. component $\mathcal{M}_j$ does not hold the shared resource).

i) If $(s^1,s^2) \in R_j$ then also $s^k \in S_{k,\overline{R}}$ and $s^j \in S_{j,R}$. Hence, since we are assuming $\big[[s^j \in S_{j,R}] \rightarrow [(e^{-E_j(s^j)a} - e^{-E_j(s^j)b}) \trianglelefteq p]\big]$, also $[(e^{-E_j(s^j)a} - e^{-E_j(s^j)b}) \trianglelefteq p]$ is true. As a result, from Theorem 5.2.1, we then have that $s^k \models_k \psi_k$, but then also the formula $\mathcal{S}X_{low}(\psi_k, \trianglelefteq, p, I, s^j)$, which since $s^j \in S_{j,R}$ is $\mathcal{S}X_{low}(\psi_k, \trianglelefteq, p, I, s^j) = \bigvee_{t^k \in S_{k,\overline{R}}}(at_{t_k} \wedge \psi_k)$ is true since at least the disjunct $(at_{t_k} \wedge \psi_k)$, clearly is satisfied by $t^k = s^k$.

ii) If $(s^1,s^2) \notin R_j$ then, from Theorem 5.2.1, we know that $s^k \models_k \mathcal{P}_{\trianglelefteq\hat{p}}(X^{\hat{I}}(\psi_k))$, where $h(p, \psi_k, \mathcal{M}_k, (s^1,s^2), I) = (\hat{p}, \hat{I})$. Since $(s^1,s^2) \notin R_j$ then also $s^j \in S_{j,\overline{R}}$, but then $\mathcal{S}X_{low}(\psi_k, \trianglelefteq, p, I) = \bigvee_{t^k \in S_k}(at_{t_k} \wedge \mathcal{P}_{\trianglelefteq\hat{p}}(X^{\hat{I}}\ \psi_k))$ (see Definition 5.2.2), which is clearly valid in s^k.

($\Leftarrow$) For simplicity we show the prove of this case by considering $k = 1$ and $j = 2$. The same result can easily be derived for the dual case $k = 2$ and $j = 1$. Here we assume that for $s^2 \in S_2$, $s^1 \models_1 \mathcal{S}X_{low}(\psi_1, \trianglelefteq, p, I, s^2)$ given that $low(\trianglelefteq, p)$ and $\big[[s^2 \in S_{2,R}] \rightarrow [e^{-E_2(s^2)a} - e^{-E_2(s^2)b} \trianglelefteq p]\big]$. We then have to distinguish between two cases.

i) If $s^2 \in S_{2,R}$ then $\mathcal{S}X_{low}(\psi_1, \trianglelefteq, p, I, s^j) = \bigvee_{t^1 \in S_{1,\overline{R}}}(at_{t_1} \wedge \psi_1)$ and then also $s^1 \in S_{1,\overline{R}}$ thus, $(s^1,s^2) \in R_j$. Hence clearly $s^1 \models \psi_1$. Furthermore, since we are assuming $\big[[s^2 \in S_{2,R}] \rightarrow [(e^{-E_2(s^2)a} - e^{-E_2(s^2)b}) \trianglelefteq p]\big]$, then $[e^{-E_2(s^2)a} - e^{-E_2(s^2)b}] \trianglelefteq p$ is true. But then from Theorem 5.2.1 also $(s^1,s^2) \models \mathcal{P}_{\trianglelefteq p}(X^I\ \psi_1)$.

ii) If $s^2 \in S_{2,\overline{R}}$ then clearly $(s^1,s^2) \notin R_j$ and also, in this case, $\mathcal{S}X_{low}(\psi_1, \trianglelefteq, p, I, s^j) = \bigvee_{t^1 \in S_1}(at_{t_1} \wedge \mathcal{P}_{\trianglelefteq\hat{p}}(X^{\hat{I}}\ \psi_1))$. Thus, clearly $s^1 \models_1 \mathcal{P}_{\trianglelefteq\hat{p}}(X^{\hat{I}}\ \psi_1))$, then from Theorem 5.2.1 also $(s^1,s^2) \models \mathcal{P}_{\trianglelefteq p}(X^I\ \psi_1)$.

□

Lemma A.0.2 *Let (s^1,s^2) be a state of a bidimensional Boucherie process, ψ_k a non-probabilistic formula as in (5.2.1), $p \in [0,1]$, $\unlhd \in \{<,\leq,\geq,>\}$ and $I=[a,b] \subseteq \mathbb{R}_{\geq 0}$ a time interval. The following equivalence holds:*

$$(s^1,s^2) \in Sat(\mathcal{P}_{\unlhd p}(X^I(\psi_k))) \Longleftrightarrow \begin{cases} s^j \in Next_j^{low}(s^k,\psi_k,p,\unlhd,I) & \textbf{if} \quad low(\unlhd,p) \\ \\ s^j \in Next_j^{up}(s^k,\psi_k,p,\unlhd,I) & \textbf{if} \quad up(\unlhd,p) \end{cases}$$

Proof.

$(\Rightarrow)$ We need to distinguish between two possibilities: $(s^1,s^2) \notin R_j$ or $(s^1,s^2) \in R_j$.

i) If $(s^1,s^2) \notin R_j$, since we are assuming $(s^1,s^2) \in Sat(\mathcal{P}_{\unlhd p}(X^I(\psi_k)))$ then from Theorem 5.2.1, also $s^k \models_k \mathcal{P}_{\unlhd \hat{p}}(X^{\hat{I}}(\psi_k))$ with $(\hat{p},\hat{I}) = h(p,\psi_k,\mathcal{M}_k,(s^1,s^2),I)$. But this implies also that $Sat_k(\mathcal{P}X_{\overline{R}_j}(\psi_k,s^1,s^2,p,\unlhd,I)) \neq \emptyset$, as

$$\mathcal{P}X_{\overline{R}_j}(\psi_k,s^1,s^2,p,\unlhd,I) = at_{s^k} \wedge \mathcal{P}_{\unlhd \hat{p}}(X^{\hat{I}}(\psi_k))$$

(see Definition 5.2.4) then proving that s^j falls in the first partition $s^j \in Next_{j,\overline{R}}(s^k,\psi_k,p,\unlhd,I)$, of the row(column) s^k independently of the type of check $(\lhd,p)$ represents, which is: $s^j \in Next_j^{low}(s^k,\psi_k,p,\unlhd,I)$, if $low(\lhd,p)$ or $s^j \in Next_j^{up}(s^k,\psi_k,p,\unlhd,I)$, if $up(\lhd,p)$.

ii) If $(s^1,s^2) \in R_j$, a further distinction has to be considered. If $low(\unlhd,p)$, since we are assuming $(s^1,s^2) \models (\mathcal{P}_{\unlhd p}(X^I(\psi_k)))$ then, as a consequence of Theorem 5.2.1, also $[e^{-E_j(s^j)a} - e^{-E_j(s^j)b}] \unlhd p$ (as clearly it is not possible that $s^k \models_k \neg tt$). Since, in this case

$$Next_{j,R}^{low}(s^k,\psi_k,p,\unlhd,I) = \bigcup_{t^j \in S_{j,R}:[e^{-E_j(t^j)a} - e^{-E_j(t^j)b}] \unlhd p} \{t^j\}$$

(see Definition 5.2.4), then clearly $s^j \in Next_{j,R}^{low}(s^k,\psi_k,p,\unlhd,I)$, hence $s^j \in Next_j^{low}(s^k,\psi_k,p,\unlhd,I)$. If $up(\unlhd,p)$ a further distinction has to be considered. If $[e^{-E_j(t^j)a} - e^{-E_j(t^j)b}] \unlhd p$ then from Theorem 5.2.1 $s^k \models_k \psi_k$. Hence from Definition 5.2.4

$$Next_{j,R}^{up}(s^k,\psi_k,p,\unlhd,I) = \bigcup_{t^j \in S_{j,R}:[e^{-E_j(t^j)a} - e^{-E_j(t^j)b}] \unlhd p} \{t^j\}$$

which proves $s^j \in Next^{up}_{j,R}(s^k, \psi_k, p, \unlhd, I) = S_{j,R}$, hence $s^j \in Next^{up}_{j}(s^k, \psi_k, p, \unlhd, I)$. If $[e^{-E_j(t^j)a} - e^{-E_j(t^j)b}] \not\unlhd p$ then from Theorem 5.2.1 $s^k \models_k \neg\psi_k$. Hence from Definition 5.2.4

$$Next^{up}_{j,R}(s^k, \psi_k, p, \unlhd, I) = S_{j,R}$$

thus, clearly, $s^j \in Next^{up}_{j,R}(s^k, \psi_k, p, \unlhd, I)$, which means $s^j \in Next^{up}_{j}(s^k, \psi_k, p, \unlhd, I)$.

($\Leftarrow$) By reversing ($\Rightarrow$).

$\square$

Lemma A.0.3 *Let $\mathcal{M}$ be a bidimensional Boucherie process and t^k a state in component $\mathcal{M}_k$, ψ_k a non-probabilistic formula as in (5.2.1), $p \in [0,1]$, $\unlhd \in \{<, \leq, \geq, >\}$ and $I = [a,b] \subseteq \mathbb{R}_{\geq 0}$ a time interval. The following equivalence holds:*

$$Sat(\mathcal{P}_{\overline{\unlhd}\overline{p}}(X^I(\psi_k))) = \begin{cases} \bigcup_{t^k \in Sat_k(\mathcal{S}X_{low}(\psi_k, \overline{\unlhd}, \overline{p}, I))} [t^k \times Next^{low}_j(t^k, \psi_k, \overline{p}, \overline{\unlhd}, I)] \\ \qquad \textbf{if } low(\overline{\unlhd}, \overline{p}) \\ \\ \bigcup_{t^k \in Sat_k(\mathcal{S}X_{up}(\psi_k, \overline{\unlhd}, \overline{p}, I))} [t^k \times Next^{up}_j(t^k, \psi_k, \overline{p}, \overline{\unlhd}, I)] \\ \qquad \textbf{if } up(\overline{\unlhd}, \overline{p}) \end{cases}$$

Proof. For brevity we consider here only the first case of the above equality, which is: we assume $low(\unlhd, p)$. Furthermore, for simplicity, we suppose to refer to component $\mathcal{M}_1$, which is we further assume $k = 1$ and $j = 2$. Again the proof for the dual case $k = 2$ and $j = 1$, is symmetrical to the following one.
We then aim to prove the following bi-implication:

$$(s^1, s^2) \in Sat(\mathcal{P}_{\unlhd p}(X^I(\psi_1))) \Longleftrightarrow (s^1, s^2) \in \bigcup_{t^1 \in Sat_1(\mathcal{S}X_{low}(\psi_1, \unlhd, p, I))} [t^1 \times Next^{low}_2(t^1, \psi_1, p, \unlhd, I)]$$

($\Rightarrow$) If $(s^1, s^2) \in Sat(\mathcal{P}_{\unlhd p}(X^I(\psi_1)))$ (i.e. $(s^1, s^2) \models \mathcal{P}_{\unlhd p}(X^I(\psi_1))$) then, from Lemma A.0.1, also $s^1 \models_1 \mathcal{S}X_{low}(\psi_k, \unlhd, p, I, s^2)$. Hence

$$s^1 \models_1 \bigvee_{t^2 \in S_2} \mathcal{S}X_{low}(\psi_k, \unlhd, p, I, t^2) \equiv \mathcal{S}X_{low}(\psi_1, \unlhd, p, I))$$

which proves $s_1 \in Sat_1(\mathcal{S}X_{low}(\psi_1, \trianglelefteq, p, I))$. Furthermore from Lemma A.0.2 also $s^2 \in Next_2^{low}(s^1, \psi_1, p, \trianglelefteq, I)$, then, clearly,

$$(s^1, s^2) \in \bigcup_{t^1 \in Sat_1(\mathcal{S}X_{low}(\psi_1, \trianglelefteq, p, I))} [t^1 \times Next_2^{low}(t^1, \psi_1, p, \trianglelefteq, I)]$$

($\Leftarrow$) By reversing ($\Rightarrow$).

$\square$

Appendix B

On the bidimensional paths

This appendix contains some background material regarding paths over a bidimensional Boucherie process (i.e. bidimensional paths). In Section 6.2 the idea of k-projection of a bidimensional path has been introduced. In essence the k-projection of a path σ is obtained by contraction of σ with respect to its *j-move*s. Intuitively, such a contraction is itself a path on $\mathcal{M}_k$. However, since the k-projection of a path has been formally defined (see Definition 6.2.5), a rigorous proof of that is needed. This result is proved in Proposition B.0.3. Before that some definition and preliminary property are introduced. A quick reminder of the principal notations and conventions adopted in the appendix is given. Unless otherwise stated, σ will denote a bidimensional path; $length(\sigma)$ denotes the number of transitions σ consists of; $\sigma[n]$ is the n-th state in the sequence σ and $\sigma[n]^k$ is the k-component of state $\sigma[n]$; a transition (or step) $\sigma[n] \to \sigma[n+1]$ is called a *k-move* if it corresponds to a transition on component $\mathcal{M}_k$; the number of *k-move*s in σ is denoted $k\_steps(\sigma)$; σ is said to be a *k-path* if it consists of *k-move*s only;

Remark B.0.1 *The states in a k-path σ have a constant j-component:*

$$\sigma[n_1]^j = \sigma[n_2]^j$$

for every $n_1, n_2 \in [0, length(\sigma)]$ with $n_1 \neq n_2$.

The above remark is a trivial consequence of the definition of Boucherie process. In fact, in a bidimensional Boucherie framework, every *global* transition corresponds to exactly one *local* transition (i.e. synchronisation is not allowed). Hence, clearly, with a *k-move*, $\sigma[n] \rightarrow \sigma[n+1] \in k\text{-}move$, the j-component of the source and target state must be constant: $\sigma[n]^j = \sigma[n+1]^j$ (see Fact 6.2.2).

Definition B.0.1 *Let σ be a bidimensional path and $\sigma^k = Proj_k(\sigma)$ its k-projection. We denote $map_k(\sigma,n)$, the index of the element of σ^k on which $\sigma[n]$ is mapped. Formally,*

$$map_k(\sigma,n) \in [0, length(\sigma^k)] : \sigma^k[map_k(\sigma,n)] = \sigma[n]^k$$

The index $map_k(\sigma,n)$, introduced in the above definition, provides a means to refer to the state of the k-projection of σ which corresponds to $\sigma[n]$.

Proposition B.0.1 *The n-th element of a bidimensional path σ, maps on the $\big(n-(j\_steps(\sigma\uparrow n))\big)$-th element of its k-projection:*

$$map_k(\sigma,n) = n - j\_steps(\sigma\uparrow n)$$

Proof. by induction on n.
<u>base</u>: $n = 0$. From Definition 6.2.5 we have, $Proj(\sigma)[0] = \sigma[0]_i$.
<u>induction</u>: we aim to show that $Proj_k(\sigma)[(n+1)-m'] = \sigma[(n+1)]^k$, given that $Proj_k(\sigma)[n-m] = \sigma[n]_k$, with $0 < n \le length(\sigma)-1$, is assumed as inductive hypothesis, where m and m' are, respectively:

$$\begin{aligned} m &= j\_steps(\sigma\uparrow n) \\ m' &= j\_steps(\sigma\uparrow(n+1)) \end{aligned}$$

We need to distinguish between two cases:

1. $\sigma[n] \rightarrow \sigma[n+1] \in k\text{-}move$.
In this case, $m' = m$, hence $Proj_k(\sigma)[(n+1)-m'] = Proj_k(\sigma)[(n-m)+1]$. The k-projection of σ can be expressed in terms of its $(n-m)$-th element, in the following manner:

$$Proj_k(\sigma) = (Proj_k(\sigma)\uparrow(n-m-1)) \;.\; Proj_k(\sigma)[n-m] \;.\; ((n-m+1)\uparrow Proj_k(\sigma))$$

Since from the inductive hypothesis we know that $Proj_k(\sigma)[n-m] = \sigma[n]^k$, then relying on Definition 6.2.5 we can rewrite the above, as:

$$Proj_k(\sigma) = (Proj_k(\sigma)\uparrow(n-m-1)) \,.\, \sigma[n]_k \,.\, Proj_k((n+1)\uparrow\sigma)) \qquad \text{(B.0.1)}$$

Let us now consider the term $Proj_k((n+1)\uparrow\sigma))$ in (B.0.1):

- if $n+1 = length(\sigma)$, then from Definition 6.2.5 we have that $Proj_k((n+1)\uparrow\sigma)) = \sigma[n+1]^k$. Hence, by substituting in (B.0.1), we have

$$Proj_k(\sigma) = (Proj_k(\sigma)\uparrow(n-m-1)) \,.\, \sigma[n]_k \,.\, \sigma[n+1]^k$$

which proves the proposition (i.e. $Proj_i(\sigma)[n+1-m'] = \sigma[n+1]_i$).

- if $n+1 < length(\sigma)$ and $(n+1)\uparrow\sigma$ is a *j-path*, then from Definition 6.2.5 it is straightforward to show that $Proj_k((n+1)\uparrow\sigma))$ consists of a single state, which is:

$$Proj_k((n+1)\uparrow\sigma)) = \sigma[length(\sigma)]^k$$

Furthermore, since we are assuming $(n+1)\uparrow\sigma$ to be a *j_path*, then from Remark B.0.1 we know that

$$\sigma[n+1]^k = \sigma[length(\sigma)]^k$$

which by substitution in (B.0.1) proves that $Proj_k(\sigma)[n+1-m'] = \sigma[n+1]^k$.

- if $n+1 < length(\sigma)$ and $\sigma[n+1] \to \sigma[n+2] \in$ *k-move*, then by Definition 6.2.5 it is straightforward to show that

$$Proj_k((n+1)\uparrow\sigma)) = ((n+1)\uparrow\sigma)[0]^k.Proj_k((n+2)\uparrow\sigma)$$

since, again, $((n+1)\uparrow\sigma)[0]^k = \sigma[n+1]^k$ then

$$Proj_k((n+1)\uparrow\sigma)) = \sigma[n+1]^k.Proj_k((n+2)\uparrow\sigma)$$

which substituted in (B.0.1) shows that $Proj_k(\sigma)[n+1-m'] = \sigma[n+1]^k$.

- if $n+1 < length(\sigma)$ and $(n+1)\uparrow\sigma$ is not a *j-path* and $\sigma[n+1] \to \sigma[n+2] \in$ *j-move*, then there exist $q \in [0, length(\sigma)-2]$ such that

$$\sigma[n+1+q] \to \sigma[n+1+(q+1)] \in k\text{-}move$$

Thus, from Definition 6.2.5

$$Proj_k((n+1)\uparrow\sigma) = \sigma[n+1+q]^k.Proj_k[(n+1+(q+1))\uparrow\sigma]$$

furthermore $(((n+1)\uparrow\sigma)\uparrow q)$ is a *j-path* hence, as a consequence of Remark B.0.1, also $\sigma[n+1+q]^k = \sigma[n+1]^k$, by which we can rewrite the above equality as

$$Proj_k((n+1)\uparrow\sigma) = \sigma[n+1]^k.Proj_k[(n+1+(q+1))\uparrow\sigma]$$

which substituted in (B.0.1) proves that $Proj_k(\sigma)[n+1-m'] = \sigma[n+1]^k$.

2. $\sigma[n] \to \sigma[n+1] \in$ *j-move*. In this case, $m' = m+1$, hence the $((n+1)-m')$-th and $(n-m)$-th element of the projected path are actually the same. Furthermore, since we are assuming the n-th transition of σ to be a *j-move*, then, thanks to Proposition 6.2.2, $\sigma[n]^k = \sigma[n+1]^k$, thus, relying on the inductive hypothesis:

$$Proj_k(\sigma)[n+1-m'] = Proj_k(\sigma)[n-m] = \sigma[n]_k = \sigma[n+1]_k$$

which proves the proposition also for the case $\sigma[n] \to \sigma[n+1] \in$ *j-move*.

$\square$

Remark B.0.2 *The k-component of the last element of a path σ maps on the last element of the σ k-projection.*

$$map(\sigma, length(\sigma)) = k\_steps(\sigma)$$

We notice that Remark B.0.2 is a direct consequence of Proposition B.0.1 and Remark 6.2.3.

Definition B.0.2 *Given the n-th element of the k-projection of a path σ, we define $Map_k^{-1}(\sigma,n)$, to be the set of elements in σ which (all) map on $Proj_k(\sigma)[n]$. Formally,*

$$Map_k^{-1}(\sigma,n) = \{m \in [0, length(\sigma)] : map_k(\sigma,m) = n\}$$

Lemma B.0.4 *The k-projections of two distinct elements of a path* σ *are in relation* $\triangleleft$ *if and only if the number of k-steps in their respective prefix are in relation* $\triangleleft$

$$map_k(\sigma, k_1) \triangleleft map_k(\sigma, k_2) \Longleftrightarrow k\_steps(\sigma \uparrow k_1) \triangleleft k\_steps(\sigma \uparrow k_2)$$

$\forall k_1, k_2 \in [0, length(\sigma)]$ *and* $\triangleleft \in \{<, \leq, \geq, >, =\}$.

Proof.
($\Rightarrow$) Let us suppose that $map_k(\sigma, k_1) \triangleleft map_k(\sigma, k_2)$, then from Proposition B.0.1 $k_1 - j\_steps(\sigma \uparrow k_1) \triangleleft k_2 - j\_steps(\sigma \uparrow k_2)$, which means (Proposition 6.2.1) $k\_steps(\sigma \uparrow k_1 \triangleleft k\_steps(\sigma \uparrow k_2))$.
($\Leftarrow$) By reversing ($\Rightarrow$).

$\square$

Fact B.0.1 *For any given path* σ*, the function* $map(\sigma, k)$ *is monotonic.*

Fact B.0.1 points out that the index on which an element $\sigma[n]$ is mapped on the k-projection of σ can only be greater or equal to the index on which any of its predecessor is mapped. This is obviously true as a consequence of definition of k-projection of σ.

Remark B.0.3 *For any path* σ *if* $\sigma[n] \to \sigma[n+1] \in$ *k-move, then*

$$map_k(\sigma, n+1) = map_k(\sigma, n) + 1$$

Remark B.0.3 points out that every k-move on a path σ is actually preserved on its k-projection, but shifted j_steps element ahead. On the other hand every j-move on σ is deleted by $Proj_k(\sigma)$, (as a result both the source and target state $\sigma[n]$ and $\sigma[n+1]$ map on the same element of the k-projection).

Relying on the results proved so far, we now introduce a proposition which is the basis to prove that the k-projection of bidimensional path σ is a path on component $\mathcal{M}_k$.

Proposition B.0.2 *Given a bidimensional path* σ*, the minimum element which maps on the n-th element of its k-projection, is the successor of the maximum element which maps on the* $(n-1)$*-th element.*

$$max(Map_k^{-1}(\sigma, n-1)) = min(Map_k^{-1}(\sigma, n)) - 1$$

$\forall n \in [1, length(Proj_k(\sigma))]$

Proof. Let us assume that m is the maximum amongst the indeces of σ which map on the n-th element of its k-projection (i.e. $max(Map_k^{-1}(\sigma, n)) = m$). Hence, clearly, $map_k(\sigma, m) = n$. Furthermore, thanks to Lemma B.0.4, also $\forall m' > m$, $k\_steps(\sigma \uparrow m) < k\_steps(\sigma \uparrow m')$. But with $m' = m+1$, that implies $k\_steps(\sigma \uparrow (m+1)) = k\_steps(\sigma \uparrow m) + 1$ which means $\sigma[m] \to \sigma[m+1] \in k\text{-}move$, hence also $j\_steps(\sigma \uparrow m) = j\_steps(\sigma \uparrow (m+1))$. From Definition B.0.1 and Proposition B.0.1, we know that

$$map_k(\sigma, m+1) = m+1 - j\_steps(\sigma \uparrow (m+1)) = m - j\_steps(\sigma \uparrow m) + 1 = n+1$$

which proves that the successor of $max(Map_i^{-1}(\sigma, n)) = m$ maps on the successor $(n+1)$ of the element it maps on (n). Relying on Fact B.0.1 we also know that $\forall m'' > m+1 \Rightarrow map_k(\sigma, m'') \geq map_k(\sigma, m+1)$, which proves $m+1$ being the minimum element of σ mapping on the $(n+1)$-th element of its m-projection.

□

Remark B.0.4 *The transition from the maximum element of a path σ whose k-projection is $n-1$ and the minimum element which maps on n, is a k-move.*

$$\sigma[max(Map_k^{-1}(\sigma, n-1))] \to \sigma[min(Map_k^{-1}(\sigma, n))] \in k\text{-}move$$

Proof. contained in the proof of Proposition B.0.2.

□

Proposition B.0.3 *The k-projection of a bidimensional path σ is a path from $\sigma[0]^k$ on component $\mathcal{M}_k$:*

$$Proj_k(\sigma) \in Path_{\mathcal{M}_k}(\sigma[0]^k)$$

Proof. by induction on $l_k = length(Proj_k(\sigma))$.

<u>base</u>: $l_k = 1$. In this case $Proj_k(\sigma) = \sigma[0]_k$ hence clearly $Proj_k(\sigma) \in Path_{\mathcal{M}_K}(\sigma[0]^k)$.

<u>induction</u>: we assume that $Proj_k(\sigma) \in Path_{\mathcal{M}_k}(\sigma[0]^k)$, if $l_k = x > 1$ and we aim to show that, as a consequence, that holds also when $l_k = x+1$. To prove that $Proj_k(\sigma) \in Path_{\mathcal{M}_k}(\sigma[0]^k)$ we need to show that

$$\mathbf{Q}_k(Proj_k(\sigma)[m], Proj_k(\sigma)[m+1]) > 0 \;\; \forall m \in [0, l_k]$$

Let as assume (inductive hypothesis) that this is the case with $l_k = x > 1$. Now let us consider $l_k = x+1$. Thus we only need to show that

$$\mathbf{Q}_k(Proj_k(\sigma)[x], Proj_k(\sigma)[x+1]) > 0$$

holds. From Corollary B.0.4 we know that

$$\sigma[max(Map_k^{-1}(x))] \rightarrow \sigma[min(Map_k^{-1}(x+1))] \in k\text{-}move$$

hence clearly $\mathbf{Q}_k(\sigma[max(Map_k^{-1}(x))], \sigma[min(Map_k^{-1}(x+1))]) > 0$. Since $Proj_k(n-1) = \sigma[max(Map_k^{-1}(x))]$ and $Proj_k(n) = \sigma[min(Map_k^{-1}(n))]$ then $\mathbf{Q}_k(Proj_k(n-1), Proj_k(n)) > 0$, which proves the induction.

$\square$

The result of the above proposition proves that, as expected, the path obtained by application of the function $Proj_k()$ (see Definition 6.2.5) on a bidimensional path σ, is a actually a path on component $\mathcal{M}_k$.

Bibliography

[1] A. Aziz, K. Sanwal, V. Singhal, and R. K. Brayton. Verifying continuous-time Markov chains. In *Proc. of 8th Int. Conf. on Computer Aided Verification*, volume 1102 of *LNCS*, pages 269–276. Springer Verlag, 1996.

[2] A.A. Lazar and T.G. Robertazzi. Markovian Petri net protocols with product form solution. *Performance Evaluation*, Vol. 12:pp. 67–77, 1991.

[3] A. Aziz, K. Sanwal, V. Singhal, and R. Brayton. Model-checking continuous-time Markov chains. *ACM Transactions on Computational Logic*, Vol. 1:pp. 162–170, 2000.

[4] C. Baier, B. Haverkort, H. Hermann, and J.-P. Katoen. Model checking continuous-time Markov chains by transient analysis. In *Proc. of 12th Int. Conf. on Computer Aided Verification*, volume 1855 of *LNCS*. Springer, 2000.

[5] C. Baier, B. Haverkort, H. Hermann, and J.-P. Katoen. Model-checking algorithms for continuous-time Markov chains. *IEEE Trans. on Software Eng.*, Vol. 29(6):pp. 524–541, June 2003.

[6] M. Bernardo. *Theory and Applications of Extended Markovian Process Algebra*. PhD thesis, University of Bologna, Italy, 1999.

[7] R. Boucherie. A characterisation of independence for competing Markov chains with applications to stochastic Petri nets. *IEEE Trans. on Software Eng.*, Vol. 20(7):pp. 536–544, 1994.

[8] C. Baier, E. Clarke, V. Hartonas-Garmhausen, M. Kwiatkowska, and M. Ryan. Symbolic model checking for probabilistic processes. In *Proc. of the 24th Int.*

Coll. on Automata, Languages and Programming, volume 1256 of *LNCS*, pages pp. 430–440. Springer-Verlag, 1997.

[9] C. Baier, J. P. Katoen, and H. Hermann. Approximate symbolic model checking of continuous-time Markov chains. In *Proc. of the 10th Int. Conf. on Concurrency Theory*, volume 1664 of *LNCS*, pages pp. 146–162. Springer Verlag, 1999.

[10] C.A. Petri. *Kommunikation mit Automaten*. PhD thesis, Institut für Instrumentelle Mathematik, Bonn, 1966.

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

[12] E. Clarke, E. Emerson, and A. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, Vol. 8(2):pp. 244–263, 1986.

[13] E. Clarke, M. Fujita, and X. Zhao. *Representations of Discrete Functions*. Kluwer Academic, 1996.

[14] E.M Clarke, O. Grumberg, and D.E. Long. Model checking and abstraction. *ACM Transactions on Programming Languages and Systems*, Vol. 16(5):pp. 1512–1542, 1994.

[15] F. Baskett, K.M. Chandy, R.R. Muntz, and F.G. Palacios. Open closed and mixed networks of queues with different classes of customers. *Journal of the ACM*, Vol. 22(2):pp. 75–93, 1975.

[16] F. Moller and C. Tofts. A temporal calculus for communicating systems. In *Proc. of CONCUR '90: Theories of Concurrency - Unification and Extension*, volume 458 of *LNCS*, pages pp. 401–415. Springer, 1990.

[17] W. Feller. *An introduction to probability theory and its applications*. John Wiley and Sons, 1968.

[18] G. Chiola, C. Dutheillet, G. Franceschinis, and S. Haddad. On well-formed coloured nets and their symbolic reachability graph. In *Proc of the 11th Int.*

Conf. on Applications and Theory of Petri Nets, LNCS, pages 387–410. Springer, 1990.

[19] H. A. Hansson and B. Jonsson. A framework for reasoning about time and reliability. In *Proc. 10th IEEE Real -Time Systems Symposium*, pages 102–111, Santa Monica, Ca., 1989. IEEE Computer Society Press.

[20] H. J. Genrich, K. Lautenbach, and P. S. Thiagarajan. Elements of general net theory. In *Proc. of the Advanced Course on General Net Theory of Processes and Systems*, pages 21–163, 1979.

[21] J. Hillston. *A compositional approach to performance modelling*. Cambridge Univesity Press, 1996.

[22] J. Hillston. *FMPA Lecture Notes*, chapter Exploiting Structure in Solution: Decomposing Composed Models. Springer-Verlag, 2001.

[23] J.-P. Katoen, M. Kwiatkowska, G. Norman, and D. Parker. Faster and symbolic CTMC Model-Checking. In *Proceedings of the Joint International Workshop PAPM-PROBMIV*, volume 2165 of *LNCS*. L. de Alfaro S. Gilmore, 2001.

[24] J.A. Bergstra and J.W. Klop. Algebra for communicating processes with abstraction. *Journal of Theoretical Computer Science*, Vol. 37:pp. 77–121, 1985.

[25] K. Jensen. *Coloured Petri Nets Basic Concepts, Analysis Methods and Practical Use*, volume 1 of *Monographs in Theoretical Computer Science. An EATCS Series*. Springer-Verlag, 1997.

[26] K. Jensen. *Coloured Petri Nets Basic Concepts, Analysis Methods and Practical Use*, volume 2 of *Monographs in Theoretical Computer Science. An EATCS Series*. Springer-Verlag, 1997.

[27] J.R. Burch, E.M. Clarke, D.L. Dill, and K.L. McMillan. Sequential circuit verification using symbolic model checking. In *Proc. of the 27th IEEE/ACM Conf. on Design Automation*, pages pp. 46–51. ACM Press, 1991.

[28] J.R. Burch, E.M. Clarke, D.L. Dill, K.L. McMillan, and J. Hwang. Symbolic model cheking: 10^{20} states and beyond. In *In Proc. of LICS '90*. IEEE Computer Society Press, 1990.

[29] J.R. Jackson. Networks of waiting lines. *Operations Research*, Vol. 5, 1957.

[30] L. Bernardinello and F. De Cindio. A survey of basic net models and modular net classes. In *LNCS Advances in Petri Nets 1992, The DEMON Project*, volume 609, pages 304–351. Springer-Verlag, 1992.

[31] L. Kleinrock. *Queueing Systems, Volume I: Theory*. John Wiley, 1975.

[32] L. Kleinrock. *Queueing Systems, Volume II: Computer Applications*. John Wiley, 1976.

[33] M. Sereno. Computational algorithms for product-form of competing Markov chains. In *Proc. of the 10th IEEE Int. Workshop on Petri Nets and Performance Models*, pages 93–102. IEEE Computer Society, 2003.

[34] M. A. Marsan, G. Balbo, G. Conte, S. Donatelli, and G. Franceschinis. *Modelling with Generalized Stochastic Petri Nets*. Wiley, 1995.

[35] R. Milner. *Communication and Concurrency*. Prentice-Hall, 1989.

[36] M. K. Molloy. Performance analysis using stochastic Petri nets. *IEEE Transactions on Computers*, Vol. C-31(9):pp. 913–917, 1982.

[37] O. Grumberg and D.E. Long. Model-Cheking and modular verification. In *Proc. of the 2nd Int. Conf. on Concurrency Theory*, volume 527 of *LNCS*, pages 250–265. Springer-Verlag, 1991.

[38] A. Pnueli. A temporal logic of programs. In *Proc. of the 18th IEEE Symposium Foundations of Computer Science*, pages 46–57, 1977.

[39] R.E. Bryant. Graph-based algorithms for boolean function manipulation. *IEEE Transtaions on Computers*, Vol. 35:pp. 1035–1044, 1986.

[40] S. Ross. *A Course on Simulation*. Maxwell Macmillan International Editions, 1991.

[41] S.Donatelli and G.Franceschinis. The PSR methodology: integrating hardware and software model. In *Proc. of the 17th Int. Conf.on Application and Theory of Petri Nets*, volume 1091 of *LNCS*, pages 133–152. Springer, June 1996.

[42] W. J. Stewart. *Introduction to numerical solution of Markov Chains*. Princeton, 1994.

[43] R. Tarjan. Depth-first search and linear graph algorithms. *SIAM J. Computing*, 1972.

[44] K. Trivedi. *Probability and statistics with reliability, queuing and computer science applications*. Prentice-Hall, 1982.

[45] W. Henderson and P.G. Taylor. Open networks of queues with batch arrivals and batch services. *Queueing Systems*, Vol. 6:pp. 71–88, 1990.

[46] W. Henderson and P.G. Taylor. Embedded processes in stochastic Petri nets. *IEEE Transactions on Software Engineering*, Vol. 17, 1991.