Labelled Markov processes

Josée Desharnais

School of Computer Science

McGill University, Montréal

November 1999

A thesis submitted to the Faculty of Graduate Studies and Research in partial fulfilment of the requirements of the degree of Doctor of Philosophy

0-612-64546-0

Résumé

Nous développons une théorie de systèmes continus probabilistes que nous prévoyons intégrer à une théorie de systèmes concourants. Notre modèle est basé sur les systèmes de transitions étiquettées. Les transitions non déterministes sont agrémentées de probabilités et nous considérons l'espace des états comme étant possiblement continu. Parmi nos principaux résultats, on compte:

- une notion de bisimulation et de simulation,
- une logique qui caractérise la bisimulation et la simulation,
- une construction permettant d'approximer les systèmes continus par des systèmes finis, et
- une métrique sur l'espace des systèmes.

Nous démontrons que la bisimulation est caractérisée par une logique très simple, sans aucune forme de négation. Le fait que cette logique ne contienne pas de négation nous permet d'obtenir une caractérisation logique de la simulation pour les systèmes probabilistes discrets, ce qui n'est pas possible pour les systèmes non probabilistes. Nous utilisons ensuite ces caractérisations pour définir deux algorithmes, l'un nous permettant de dire si deux systèmes sont bisimilaires, l'autre pour déterminer si un système simule un autre système.

Nous montrons comment approximer un système continu à l'aide de systèmes finis et comment reconstruire le système à partir de ses approximations. Parmi ces approximations, il en existe qui se trouvent aussi près que voulu du système original.

Nous définissons une métrique sur l'espace des systèmes continus qui reflète la différence entre les systèmes. Des systèmes qui sont très différents se verront attribuer une grande distance alors que deux systèmes bisimilaires auront une distance nulle. En fait, une famille de métriques est définie et nous démontrons que les approximations finies d'un système continu convergent vers ce système pour toutes ces métriques.

Finalement, nous démontrons que les systèmes dont le graphe de transition est un arbre et dont les probabilités sont rationelles forment une base (dénombrable) de l'espace des systèmes étudiés. Cet espace est donc un espace métrique séparable.

Summary

We develop a theory of probabilistic continuous processes that is meant ultimately to be part of an interactive systems theory. Our model is a generalization of ordinary labelled transition systems to which we add probabilistic transitions. The four main contributions are

- a notion of bisimulation equivalence and simulation preorder,
- a logic for characterizing bisimulation and simulation,
- an approximation scheme and
- a metric on the collection of processes.

We prove that bisimulation is characterized by a very simple logic that neither involves negation nor infinite conjunction. We have a similar result for simulation between discrete processes. Moreover, these characterizations are used to construct two algorithms, one that decides whether two finite-state probabilistic processes are bisimilar, and another that decides whether a state simulates another.

We show how to approximate any continuous process with finite-state processes, and that one can reconstruct the process from its approximations. These finite approximations can be as close as we want to the original process. Moreover, we define a family of metrics that can tell how far apart or how close two processes are. The metrics also witness the fact that the approximations converge to the original process.

Finally, we prove that the processes where the transition graph is a tree and whose transition probabilities are all rational form a basis of the space of labelled Markov processes; this means that labelled Markov processes form a separable metric space.

Acknowledgments

First I would like to warmly thank my thesis supervisor, Prakash Panangaden, for his precious enthusiasm, for his invaluable support and patience, for having so strongly believed in me and for his continuous help during the preparation and writing of this thesis. I also want to thank him for having introduced me to the subject of this thesis, for having given me the opportunity to travel and meet people by sending me to many conferences in many beautiful countries.

I would like to thank Abbas Edalat whose collaboration has been essential for the theory of bisimulation to make sense, especially with the help of his semi-pullback theorem. I want to thank Vineet Gupta, Radha Jagadeesan and again, Prakash Panangaden for highly stimulating discussions that have led to the discovery of many results in this thesis. I also want to thank Rick Blute, Franck van Breugel, Martin Escardo, Patrick Lam, François Laviolette, Ben Rich and Erik de Vink for helpful discussions and comments and Falk Bartels for pointing out a mistake in an earlier version of the finite approximation construction.

Je désire remercier tout spécialement mon compagnon, François, pour avoir travaillé patiemment sur ma confiance fragile, pour m'avoir encouragée à tourner mon intérêt vers l'informatique, pour son extraordinaire et inébranlable caractère et pour avoir pris une foule de choses en charge dans les dernières semaines de la rédaction de cette thèse.

Je remercie également ma famille, mes amis qui m'ont aidée de différentes façons, entre autre par leur affection, leurs encouragements. Je voudrais remercier en particulier Claude Crépeau pour m'avoir permis de rencontrer le meilleur des directeurs de thèse.

This work has been supported by NSERC.

À ma mère, pour m'avoir donné toutes les chances
de choisir mon chemin.

Contents

Chapter 1

Introduction

This thesis is concerned with the analysis of probabilistic systems with a continuous state space. The general goal is to make a step towards establishing formal methods for reasoning about such systems. Normally one associates formal methods with logic and discrete structures. In this thesis, we will explore how the ideas developed for discrete probabilistic systems can be applied to systems with a continuous state space. The mathematical techniques needed are very different from previous work in concurrency theory; in particular, a significant role is played by measure theory in some of the key results.

The systems we consider have a continuous state space in order to model physical systems; they have labelled transitions that are quantified with probabilities; they are reactive in the sense that we study them with respect to how they react to actions taken by the environment, and hence they are meant to be concurrent; finally, even if the state space can be infinite, the transitions are discrete and hence time is discrete. These concepts will be explained and motivated in the following section.

1.1 Motivations

In recent years, physical systems have become important in computer science. A physical system is any system that evolves in a continuous state-space, by involving continuous parameters such as distance, temperature, pressure. These systems arise in hybrid systems theory (see for example [AHS96], [AKNS97]). A hybrid system is a continuous physical system combined with a control process which is discrete and most of the time finite. A typical example is a simple railroad crossing, consisting of a train, a gate and a controller; the controller must ensure that the gate is down whenever the train is in the intersection, and that it is not closed unnecessarily. The motion of the train is continuous but the states of the controller are discrete. Other examples are traffic controllers, heating systems, flight control systems, etc. Even if, in practice, one always discretizes a system before using it or before reasoning about it, a theory of continuous systems is necessary to be able to argue that a discretization of a process is indeed a faithful model of the process. Moreover, it is essential to have a notion that tells if a discrete process is "closer" to, or is a finer approximation of a process than another discrete process.

The systems that we consider have a continuous state-space but make discrete steps – in other words time is discrete. One motivation for studying these systems is that this is a reasonable middle ground before studying the more general case of continuous state space and continuous time. Moreover, it has some practical applications that we encounter in control systems. The following example is taken from [DEP99] and abstracts from a practical investigation in collaboration with industry. We have a physical system, like an airplane, and a controller that takes reading at fixed interval – say every one tenth of a second – of a number of parameters. Depending on the reading, it takes some actions to keep the physical system stable. The number of possible states is really continuous, but the steps are discrete. Of course the dynamics is occurring in continuous time but the sampling occurs in discrete time.

Ultimately, these systems are to be part of an interactive systems theory. Compositionality has been one way of dealing with the formidable problem of complexity of systems. Complex processes are broken up into components that we study separately. We do not study composition of systems in detail but we take a view that is common in concurrency theory. Processes are studied with respect to their interaction with the environment. The environment can be a user or another process: we are interested in what this external observer can observe about a process. Intuitively, a process is a system that evolves in time by executing some actions in response to actions taken by the environment. The system is in a state at a point in time and makes transitions between states depending on which interaction with the environment is taking place. The formalism used for describing the behaviour of processes is *labelled transition systems.* A labelled transition system consists of a set of states, a set of labels and a set of transitions which are labelled. The label indicates which interaction the environment is requesting: a process makes an a-labelled transition only if the environment also simultaneously makes an a-labelled transition. In non-deterministic labelled transition systems, there may be more than one transition from a given state having the same label.

In probabilistic processes, non-determinism is not only enumerated, it is quantified. The fundamental work about discrete probabilistic transition systems is by Larsen and Skou [LS91]. Like ordinary labelled transition systems, probabilistic transition systems consist of a set of states, a set of labels and a set of transitions. The difference is that the state to which the process jumps is determined by a probability function. The work that has been done until now restricts to processes having a finite or countable number of states. We have generalized the theory to continuous state-space systems (which we have called labelled Markov processes). Modeling transitions in this context requires notions of measure theory. Indeed, in uncountable spaces, the probability of jumping to a single state is often zero; therefore we must consider instead the probability of jumping to a set of states. In this manner, each state s and each action a will have an associated sub-probability measure which describes the effect of action a when the process is in state s.

With the advent of computer controlled physical systems has come the need for analyzing and reasoning about them; this is best done with the help of formal methods. Formal methods are mathematically based languages, techniques and tools for specifying (describing) and verifying systems. When designing a system, the ultimate goal is to make it operate reliably, despite its complexity. Formal methods are used more and more in industry, in particular in the last ten years. A good survey of their use can be found in [CW96]. Even when we just use formal methods for specification, they have been proven to greatly improve product quality and error detection time;

they often even reduce the development cost of systems because of the savings in testing that they provide. At the verification level, formal methods allow one to discover subtle errors that could not be detected by humans because of the size of systems; they also provide one with information that aids in debugging. There are examples of existing and heavily used systems in which verification tools have found errors. One famous such example is the Needham-Schroeder protocol [NS78], invented in 1978 and used since then. A bug was discovered in 1995 by Gavin Lowe with the help of a model verification tool at Oxford [Low96].

Establishing formal methods for continuous state-space systems is not an easy task. In our case, we have foundational work to do before we can think about the use of logical formal methods. It is not à priori obvious that logical formulas capture anything interesting about a continuous state-space system. However there are results in the theory of hybrid automata that give one some reason to hope. In particular, the work on linear hybrid automata shows that in certain cases one can exploit the structure of the system to construct an equivalent finite-state system [ACH+95] thus making it amenable to algorithmic analysis. It has even been possible to construct tools – eg. HYTECH – for verifying hybrid systems [HHWT95]. Of course these results are suggestive for our context but do not give any direct technical results that we can use for probabilistic systems.

1.2 Contributions

In algebraic approaches to concurrency theory, the important issue is whether two labelled transition systems are equivalent. In particular, two different descriptions of processes could have the same behaviour and thus lead to the same process. But what do we mean by "having the same behaviour"? There are in fact many notions of equivalence. We are interested in an *observational* equivalence, called strong bisimulation, introduced by Milner [Mil80]. This equivalence is based on what an external observer (the environment) can say about the difference between two processes.

We define bisimulation for labelled Markov processes. Our definition is a generalization (to continuous state-space systems) of Larsen and Skou's probabilistic bisimulation [LS91]. In continuous systems, one practical use of bisimulation is to evaluate if a given process is in fact discrete despite the fact that it might be described as a continuous system. More generally, we show how to construct the "simplest" process bisimilar to a given process. For finite-state systems, this is a state minimization construction.

Apart from knowing that two systems are equivalent, perhaps a more practical issue is refinement, that is, can a given system "replace" another, or can it *simulate* it in the sense that everything the second one can do can also be done by the first one? Generalizing this concept to probabilistic processes, we will say that a process *simulates* another if, from the user point of view, it can make all the transitions the second one can perform with higher probabilities.

The most important contribution of this thesis is a logical characterization of bisimulation for Markov processes. That is, we define a logic which is the analogue of Hennessy-Milner logic for non-probabilistic processes. Two processes are bisimilar if and only if they satisfy the same formulas of the logic. A logic is useful for specifying properties that we want our system to satisfy and for verification. By checking that two processes satisfy the same formulas, we can decide if they are bisimilar, and conversely, by only finding a formula that distinguishes them, we can prove that two processes are not bisimilar. This can be done automatically. The logic we define is surprisingly simple: in particular, it does not involve any kind of negation, nor is it limited to processes having finite branching (i.e., having a finite number of transitions with the same label from any state) in order to characterize bisimulation. This was a surprise because in the non-probabilistic case, infinite conjunction is necessary to witness infinite branching and negation is also necessary even for finite branching systems. In addition, we report five different logics – none of them being equivalent – all characterizing bisimulation. The proofs are of entirely different character from those required for the non-probabilistic case and use technical results from measure theory. A logic without negation carries a notion of simulation and indeed we prove that our logic augmented with disjunction characterizes simulation for discrete systems: a state simulates another state if it satisfies all the formulas the other satisfies.

We give an application of the logical characterizations by defining algorithms, one that can decide whether two finite-state probabilistic processes are bisimilar, and the other that can decide whether a state simulates another. The algorithms exhibit a formula that is satisfied by only one of the processes if they are not bisimilar or if this process is not simulated by the other. The fact that the logic is simple is an advantage for debugging, but for specification of properties, it is better to work with more expressive logics. Moreover, the complexity of the logic has no impact on satisfiability checking.

As we have already said, the analysis of a continuous system is greatly simplified if we can prove that it is equivalent to a finite-state process. Of course, one cannot hope that for every continuous process one can find a bisimilar finite-state process. Nevertheless, we show how to approximate any continuous process with finite-state processes that can be as close as we want to the original process. We show that we can reconstruct a process from its finite-state approximations – more precisely a bisimulation equivalent of the original process –, and prove that the approximations capture all the logically definable properties that the original process satisfies. A finite-state approximation allows one to reason more easily about the continuous process it approximates. However, we should not discard the continuous process by saying that it has become useless since we have the finite-state one. For different purposes, we may need finer approximations to our process.

Though it is a very useful notion, bisimulation has some limitations. Bisimulation tells us when two processes are essentially behaviourally identical. If they differ slightly in the probabilities, bisimulation says that they are not bisimilar. On the other hand, simulation tells us a little more. Two processes, although not bisimilar, can be related by a simulation relation; one can be greater than the other. But simulation does not tell us how distant the processes are in a quantitative fashion and, since it is not a total order, there are pairs of processes for which it tells us nothing precise. Like bisimulation, it is not robust; a very small change in probabilities will likely result in non-bisimilar and "non-similar" processes. We will introduce a *metric* that will allow us to refine our view of processes. This metric will assign a number to every pair of processes, giving so an indication of how far they are from each other. If the metric distance is 0, then the two processes will turn out to be bisimilar, and conversely. Processes that are very "close" will get smaller distance than processes that are "far" apart. In order to define these metrics, we will shift from the traditional view of logical formulas to measurable functions into $[0, 1]$. These functions will play the same role as the logic formulas we mentioned above, but in addition, they will provide us with numbers that we will use to define the metrics.

The definition of bisimulation we work with is given in terms of the existence of a relation between states of processes. In fact, the original presentation of bisimulation for labelled Markov processes was given in categorical terms and the relational view evolved later. The categorical view of bisimulation for labelled Markov processes is based on the ideas of Joyal, Nielsen and Winskel [JNW96].

1.3 Outline of thesis

In the next chapter, we recall well-known definitions and results on non-probabilistic labelled transition systems. We define these terms and explain the notion of bisimulation and simulation and the intuition behind them. We also show how they are characterized by a modal logic due to Hennessy and Milner. In Chapter 3, we state all the basic definitions and motivations pertaining to our model. We explain what labelled Markov processes are, the notions of bisimulation and simulation and how these definitions are generalizations of their non-probabilistic analogues. We also show how they are organized in a category and finally we give a few examples to illustrate the ideas.

In Chapter 4, we define a simple probabilistic variant of Hennessy-Milner logic, written $\mathcal{L}$, as well as four other logics that are extensions of $\mathcal{L}$, and we prove that $\mathcal{L}$ characterizes bisimulation for labelled Markov processes. We also prove that by adding disjunction to $\mathcal{L}$ we can characterize simulation for discrete processes. We use these characterizations to define two algorithms for finite-state systems, one that can decide whether two states of a process are bisimilar, and the other that can decide whether a state simulates another. Finally, we prove that all the logics we define characterize bisimulation and discuss which one can characterize equivalence classes of bisimilar states.

In Chapter 5, we show how to construct a family of finite-state processes from any continuous-state process that approximates the process and show that we can reconstruct the original process from the finite approximations. The finite-state approximations will be shown to be simulated by the process so that, in some sense, they really only capture properties of the original process; conversely, we show that every definable property that is satisfied by the process is satisfied by some finite-state approximation. In Chapter 6, we define metrics for labelled Markov processes that will strengthen these results by showing that the approximations of a process converge to it in the metrics. These metrics are defined with the help of measurable functions into $[0, 1]$. The functions are shown to capture bisimulation in the sense that they give equal value to, and only to, bisimilar states. A similar result is proven for simulation in the case of discrete systems. We also show that the space of labelled Markov processes is a separable metric space.

In Chapter 7, we give the alternative definition of bisimulation in categorical terms and prove that it is equivalent to the relational definition we give in Chapter 3. We also focus on discrete or finite-state processes and give some proofs in these systems that are simpler than the same results on arbitrary systems.

Chapter 8 contains a short summary of the contributions we made and discussions on related and future work.

In the appendix, we give some definitions from measure theory and probability theory that are useful in the reading of this thesis.

Chapter 2

Background on non-probabilistic transition systems

This chapter is a review of well-known definitions and results on non-probabilistic labelled transition systems. Our analysis of probabilistic systems is based on these ideas. A reader familiar with process algebra can skip this chapter.

A process is a dynamical system: it evolves in time by executing some actions or by reacting to events, performing *transitions* between states. These transitions could be purely internal or could be the result of an interaction with the environment. The word "process" is used to refer to such systems and the phrase "transition system" is used to refer to the explicit presentation of the states and transitions of the system. One can think of processes as being defined by some syntax and the transition systems as representing their semantics. In this thesis we will view processes as transition systems and not discuss syntactic descriptions of processes. We recall the definition of labelled transition systems in the next section. In Section 2, we give the intuition behind bisimulation and simulation by describing two-player games that can be played to determine if a system simulates another or if it is bisimilar to it. We then give the formal definition of bisimulation and simulation which are in terms of the existence of a relation between the states of the systems. In Section 4, we recall an alternative formulation of bisimulation and simulation, given in a categorical setting. Finally, we state the well-known characterization of bisimulation by Hennessy-Milner logic. This is the prototype of the logical characterization results that we prove.

2.1 Labelled transition systems

The processes that concern us interact with their environment by synchronizing on labels in the manner familiar from process algebra. Thus we have a set of states, a set of labels and, for each state and label, there can be a transition. Suppose the system is in state s and the environment chooses a label, say a; then the system makes the corresponding a-labelled transition to a new state if the action a is enabled in state s. In traditional process algebra, the resulting state is chosen arbitrarily from a set of possible result states. In our case the result state is chosen according to some given probability distribution. It is assumed that a process makes an a-labelled transition only if the environment also simultaneously makes an a-labelled transition. Thus interaction is synchronous. In practice, processes can perform internal moves and change state between two given interactions with the environment but here we are only interested in what the environment can really *observe* regarding the process, therefore we do not use these internal transitions. In fact, in this thesis, we will not even consider that these unobservable transitions take place; we will consider all transitions as arising from the environment.

Definition 2.1.1 A *labelled transition system* is a tuple $(S, i, \mathcal{A}, \rightarrow)$ where S is a finite or countable set of *states*, $i \in S$ is an initial state, $\mathcal{A}$ is a set of labels or actions, and $\rightarrow \subseteq S \times \mathcal{A} \times S$ is a set of *transitions*. When $(s, a, s') \in \rightarrow$ we write $s \xrightarrow{a} s'$.

We fix the label set to be some $\mathcal{A}$ once and for all; we frequently refer to a labelled transition system by its set of states. A transition system is often represented as a transition graph. The vertices of the graph represent states of the system and an arrow with label a connecting a state s to another state s' represents an a-transition from s to s', thus a triple $(s, a, s') \in \rightarrow$.

Example 2.1.2 *Here are two simple labelled transition systems with initial states s_0 and t_0 respectively. The first one is deterministic, whereas the second one has one indeterminate transition from state t_0. If the environment asks for action a, this* *system can jump to either t_1 or t_2.*

$$\begin{tikzcd}
& s_0 \arrow[d, "a"] & \\
& s_1 \arrow[dl, "b"'] \arrow[dr, "c"] & \\
s_2 & & s_3
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
& t_0 \arrow[dl, "a"'] \arrow[dr, "a"] & \\
t_1 \arrow[d, "b"'] & & t_2 \arrow[d, "c"] \\
t_3 & & t_4
\end{tikzcd}$$

An important question in the theory of concurrency is the notion of process equivalence: when do two labelled transition systems describe the same process? For example, it could happen that a labelled transition system having an infinite number of states behave like a very simple system with a finite number of states. Should we consider the two processes of Example 2.1.2 equivalent? The notion of equivalence between processes is commonly used to check if a given implementation matches its specification. Typically, both the specification and the implementation are described as labelled transition systems, and then we check if the processes derived are equivalent. There are several different notions of equivalence. One of the most basic and mathematically pleasing notions is strong bisimulation due to Milner [Mil80] and Park [Par81]. Strong bisimulation comes with a theory rich in results: the classical characterization is in terms of the existence of a relation between states of the systems, but there are as well a two-player game and a logic that characterize strong bisimulation. More interestingly for concurrency theory, strong bisimulation has a pleasant algebraic theory: it is compositional and can be characterized as a fixed point. Roughly speaking, two labelled transition systems are strongly bisimilar if they are indistinguishable from the point of view of the user. At each point of interaction, every move that can be taken by any of them can be matched by the other. The adjective "strong" refers to the fact that invisible moves are not considered in analyzing systems, all moves are visible. Since this is the only kind of bisimulation we are interested in, we will drop the word "strong" in the sequel. There is a notion of *weak bisimulation* that distinguishes internal actions from visible ones. Another notion of equivalence that is widely used in language theory is *trace-equivalence.* This notion is concerned with which sequences of actions are accepted by a process. Two processes are trace-equivalent if they accept the same sequences of actions. This will be illustrated in Example 2.1.3. The main disadvantage of this equivalence from the point of view of concurrency theory is that it is not compositional. If we compose two processes that are trace-equivalent with a third process, the results may not be trace-equivalent.

Sometimes equivalence relations like bisimulation may be too strong. They only tell us if two processes are equivalent. We may be interested in knowing if a process, though not equivalent to another one, can replace it. The preorder analogue of bisimulation is simulation. It tells when a system is "better than" another. That is, when – from the user's point of view – a system allows *at least* the same possibilities as the other. We can use simulation to check if an implementation simulates its specifications, that is, it is able to do at least what it was required to do.

To see what simulation and bisimulation are, let us recall Milner's interpretation of a process: it is a black box on the top of which there are buttons which are labelled. We want to investigate the behaviour of the process by demanding it to accept labels one at a time. If an a-transition is enabled, button a will be unlocked and go down when pressed. If not, the button will be locked. This is what simulation and bisimulation will witness: how systems interact with their environment, in particular, it will record what transitions are or are not enabled. This is in contrast with trace-equivalence where only enabled transitions are recorded.

Example 2.1.3 *Consider the classical transition systems of Example 2.1.2. These two systems are trace-equivalent: they accept the same language, namely the sequences ab and ac. Now if we study their interaction with their environment, they can be distinguished easily. The first system will always accept an attempt to press button a followed by b, whereas the second may have its b-button locked after accepting a. Hence they are not bisimilar. On the other hand, the former does simulate the latter.*

On the other hand, if we restrict to deterministic labelled transition systems –that is, we do not allow more than one transition with a given label out of a state, we have that bisimulation is the same as trace-equivalence: two systems are bisimilar if they have the same trace set (see [Mil89]). Moreover, a system S simulates another, say T, if the trace set of S contains the trace set of T. Hence in this case, we also have that bisimulation is equivalent to two-way simulation. We say that two systems are *two-way similar* if they simulate each other. For indeterminate processes, the following example illustrates that the situation is different.

Example 2.1.4 *The following two systems simulate each other but are not bisimilar.*

$$\begin{tikzcd}
 & s_0 \arrow[d, "a"] & & & & & t_0 \arrow[dl, "a"'] \arrow[dr, "a"] & \\
 & s_1 \arrow[dl, "b"'] \arrow[dr, "c"] & & & & t_1 \arrow[dl, "b"'] \arrow[dr, "c"] & & t_2 \\
s_2 & & s_3 & & t_3 & & t_4 &
\end{tikzcd}$$

After accepting action a when in state t_0 the second process may jump to state t_2 where no action is enabled. On the contrary, the first process, in state s_0, will always accept action a followed by b. By pressing buttons on both processes, we may not get the same answers; hence they are not bisimilar.

2.2 Simulation and bisimulation: a game description

Before stating the classical formal definition of simulation and bisimulation, we describe a characterization in terms of games that is based on the idea of pressing buttons we mentioned above. This gives a good intuitive feeling for the concepts.

We begin with simulation. Let S and S' be two labelled transition systems, and assume we have two players, one of them, called Player, trying to show that S is simulated by S', and the other one, Opponent, trying to show the contrary. Informally, a play progresses as follows. Opponent starts out by choosing a transition from the initial state of S with a label, say a; if Player cannot match the move with a transition from the initial state of S' with the same label, he loses. Otherwise, he chooses such a matching transition, and it is again Opponent's turn to move. He chooses a transition leading out of the state arrived at in the previous pair of moves.

Again Player is required to match with an equally labelled transition in S'. The play continues like this forever, in which case Player wins, or until either Player or Opponent is unable to move, in which case the other participant wins. S is simulated by S' if and only if Player has a winning strategy. If there is some play where Opponent can win, S is not simulated by S'.

Example 2.2.1 *Let us go back to Example 2.1.2 and show that by playing this game, Opponent can succeed in showing that the second one does not simulate the first one, but that the converse is not true. Opponent starts up by choosing, from state s_0, the transition a to state s_1: to this, Player has no choice but to choose an a-transition from t_0 to t_1 or t_2. In the first case, Opponent then chooses s_3, in the second one, he chooses s_2: in either cases, Player won't be able to match the move, thus the process t_0 does not simulate s_0. On the other hand, s_0 does simulate t_0, since no matter which sequence of transitions on the second system Opponent picks, Player will be able to match the move at each step.*

The game for bisimulation is a variation of the previous one. However, the difference is crucial and it makes bisimulation much stronger than simulation and also much stronger than simulation in both directions. The difference in the rules of the game is that Opponent is not required to play in the same system all the time. When he chooses a transition at the beginning, he can choose it from the initial state of either system, forcing Player to play in the other one, and when he chooses a transition from a state at any point in the game, he can do it from any of the two states arrived at in the previous pair of moves, that is, he can change machine at will. The fact that Opponent is allowed to change machine captures the notion of equivalence needed for bisimulation: if two states are equivalent, it should not matter from which state Opponent performs its move for Player to be able to match it. The winner is determined in the same way, and two systems are bisimilar if and only if Player has a winning strategy.

Example 2.2.2 *To illustrate bisimulation, let us go back to Example 2.1.4. Since s_0 and t_0 simulate each other, Opponent will have to change process during the play to* *show that they are not bisimilar. He starts up by choosing t_0 and the a-transition to t_2. Player must choose s_0 and jump to s_1. But then, Opponent switches processes and chooses the b-transition from s_1 to s_2. Player cannot play, and looses. Therefore, the two systems are not bisimilar.*

2.3 Formal definitions of bisimulation and simulation

The definition of bisimulation was formulated by Milner [Mil80] and described as a fixed point by Park [Par81].

Definition 2.3.1 *Let $(S, i, \rightarrow)$ and $(S', i', \rightarrow')$ be two labelled transition systems. A relation $\mathcal{R} \subseteq S \times S'$ is a* ***simulation*** *if $(s, s') \in \mathcal{R}$ implies that for all $a \in \mathcal{A}$,*

- *if $s \xrightarrow{a} t$, then there exists $t' \in S'$ such that $s' \overset{a}{\rightarrow}{}' t'$ and $(t, t') \in R$.*

A state s is simulated by s' if there exists a simulation $\mathcal{R}$ such that $(s, s') \in \mathcal{R}$; S is simulated by S' if i is simulated by i'. A relation $\mathcal{R} \subseteq S \times S'$ is a ***bisimulation*** *if $(s, s') \in \mathcal{R}$ implies that for all $a \in \mathcal{A}$,*

- *if $s \xrightarrow{a} t$, then there exists $t' \in S'$ such that $s' \overset{a}{\rightarrow}{}' t'$ and $(t, t') \in R$; and*
- *if $s' \overset{a}{\rightarrow}{}' t'$, then there exists $t \in S$ such that $s \xrightarrow{a} t$ and $(t, t') \in R$.*

Two states s, s' are bisimilar if there exists a bisimulation $\mathcal{R}$ such that $(s, s') \in \mathcal{R}$. S is bisimilar to S' if i is bisimilar to i'.

2.4 The categorical definition of bisimulation

In [JNW96], Joyal, Nielsen and Winskel gave a categorical formulation of bisimulation for ordinary labelled transition systems. We will give a similar formulation of bisimulation for labelled Markov processes. See [BW90, Mac71] for definitions relating to category theory.

Simulation is easily formulated in terms of morphisms between processes. Indeed, a system S is simulated by S' if there is a function $f : S \rightarrow S'$ that sends the initial state of S to the initial state of S' and *preserves* transitions, that is, every move that can be done by a state s in S can be imitated by $f(s)$ in S'; moreover, if a move from s leads to say, state $t \in S$, then $f(s)$ can match this move by jumping to a state which again simulates t, namely $f(t)$. At first sight, the direction of the arrow (from the simulated to the simulating process) may appear wrong, but one must keep in mind that we want every state of S to have a corresponding simulating state in S'. Hence it is natural to use a map that gives an image in S' to every state of S.

Formulating bisimulation in this context is done by using spans which are the analogue of relations in a categorical setting. A *span* between two objects S_1 and S_2 of a category is a third object T together with morphisms from T to both S_1 and S_2. Consider **Sets**, the category having sets as objects and functions between sets as morphisms. One can think of a relation as a span. For any relation R between two sets S_1 and S_2, the set of ordered pairs $\{(s_1, s_2) \in S_1 \times S_2 \mid s_1 R s_2\}$ together with the projection morphisms is a span between S_1 and S_2 in **Sets**; conversely, given a span $T, f_i : T \longrightarrow S_i$, $i = 1, 2$, we can define the relation $R \subseteq S_1 \times S_2$ as $s_1 R s_2$ if there is a $t \in T$ such that $f_1(t) = s_1$ and $f_2(t) = s_2$.

The categorical definition of bisimulation will be in terms of the existence of a span of special morphisms, encoding in this manner the equivalence relation of Definition 2.3.1. These special morphisms will relate in particular bisimilar processes; hence, they must satisfy a condition that captures the notion of Definition 2.3.1, and thus not only preserve (as simulation morphisms do) but also "reflect" transitions.

Definition 2.4.1 Let $(S, i, \rightarrow)$ and $(S', i', \rightarrow')$ be two transition systems with the same labeling set $\mathcal{A}$. A *simulation morphism* from S to S' is a function $f : S \rightarrow S'$ such that $f(i) = i'$ and

$$s \xrightarrow{a} s' \Rightarrow f(s) \xrightarrow{a}{}' f(s').$$

The morphism f is called *zigzag* if and only if for all states s of S

if $f(s) \xrightarrow{a}{}' s'$ in S', then $s \xrightarrow{a} u$ in S and $f(u) = s'$, for some state u of S.

It is easy to check that morphisms with labelled transition systems form a category.

Theorem 2.4.2 *S is simulated by S' if and only if there is a simulation morphism from S to S'.*

Theorem 2.4.3 *Two labelled transition systems T, T' are* bisimilar *if and only if there is a span of zigzag morphisms f and f' between them.*

$$\begin{tikzcd}
& S \arrow[dl, "f"'] \arrow[dr, "f'"] & \\
T & & T'
\end{tikzcd}$$

The following example illustrates why bisimulation has to be defined in terms of a span of zigzag morphisms instead of just a zigzag morphism.

Example 2.4.4 *The two following processes are bisimilar but there is no zigzag morphism between them in either direction.*

$$\begin{tikzcd}
& \bullet \arrow[dl, "a"'] \arrow[d, "a"] \arrow[dr, "a"] & \\
\bullet \arrow[d, "b"'] & \bullet & \bullet \\
\bullet & &
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
& \bullet \arrow[dl, "a"'] \arrow[d, "a"] \arrow[dr, "a"] & \\
\bullet \arrow[d, "b"'] & \bullet \arrow[d, "b"'] & \bullet \\
\bullet & \bullet &
\end{tikzcd}$$

If we want to really work in a categorical setting, we can use the statement of Theorem 2.4.3 as the definition of bisimulation. In that case, we have to check that bisimulation is an equivalence relation.

Since the identity morphism is a zigzag morphism, any system is bisimilar to itself and hence bisimulation is reflexive; it is also clearly symmetric. Transitivity of bisimulation is equivalent to the following property: for every pair of zigzag morphisms f_1 and f_2 having domain S_1 and S_2 respectively and a common codomain S, we can always complete the square with zigzag morphisms as in the following diagram.

$$\begin{tikzcd}
& U \arrow[dl, dotted] \arrow[dr, dotted] & \\
S_1 \arrow[dr, "f_1"] & & S_2 \arrow[dl, "f_2"'] \\
& S &
\end{tikzcd}$$

The proof that bisimulation defined as a span is an equivalence can be found in [JNW96] for transition systems with independence[1]: it is proved that in this category we have pullbacks, which implies the result. Technically, the span U needs not be a pullback, and indeed, in the category of labelled Markov processes that we will define, the square can be completed but we do not have pullbacks.

In the following, we give an example of how bisimulation can be used to check if an implementation matches its specifications.

Example 2.4.5 *Suppose we have a cell:*

$$\begin{tikzcd} \mathrm{cell}_e \arrow[r, shift left, "put"] & \mathrm{cell}_f \arrow[l, shift left, "get"] \end{tikzcd}$$

where cell_e *and* cell_f *are understood to be respectively the empty and the full cell. Now suppose we want to implement a bag of size 2 using this cell. Here is our bag of size two:*

$$\begin{tikzcd} \mathrm{bag}_0 \arrow[r, shift left, "put"] & \mathrm{bag}_1 \arrow[l, shift left, "get"] \arrow[r, shift left, "put"] & \mathrm{bag}_2 \arrow[l, shift left, "get"] \end{tikzcd}$$

where the index attached to bag *represents the number of messages present in the bag. We expect that we can implement this bag by putting two cells in parallel* $\mathrm{cell}_e|\mathrm{cell}_e$. *Without going into the details of how two processes are put in parallel, just note that the states are now pairs of states and transitions happen from one state if either one or the other coordinate of the pair can perform the action to the same coordinate of the arriving state. This is easily understood in the next picture.*

$$\begin{tikzcd} & \mathrm{cell}_e|\mathrm{cell}_e \arrow[dl, shift right, "put"'] \arrow[dr, shift left, "put"] & \\ \mathrm{cell}_f|\mathrm{cell}_e \arrow[ur, shift right, "get"'] \arrow[dr, shift left, "put"] & & \mathrm{cell}_e|\mathrm{cell}_f \arrow[ul, shift left, "get"] \arrow[dl, shift right, "put"'] \\ & \mathrm{cell}_f|\mathrm{cell}_f \arrow[ul, shift left, "get"] \arrow[ur, shift right, "get"'] & \end{tikzcd}$$

[1] Transition systems with independence are labelled transition systems with an additional relation on transitions that tells us when two transitions are independent, that is, they can be performed in any order.

In order to show that $\mathrm{cell}_e|\mathrm{cell}_e$ *is a good implementation of* bag_0*, we can verify that they are bisimilar by constructing a zigzag morphism from the former to the latter. The morphism which sends* $\mathrm{cell}_e|\mathrm{cell}_e$ *to* bag_0*,* $\mathrm{cell}_f|\mathrm{cell}_f$ *to* bag_2 *and the two other states where exactly one cell is full to* bag_1 *is easily proved to be zigzag. Thus we can conclude that our implementation is correct.*

2.5 Hennessy-Milner logic

Bisimulation between states of labelled transition systems is characterized by a modal logic due to Hennessy and Milner [HM85]. Two states are bisimilar if and only if they satisfy the same formulas of the logic. What is interesting about this fact is that if we want to verify that two systems are not bisimilar, we only have to find a formula that distinguishes them. Moreover, the witnessing formula gives information about why the states are not bisimilar.

A logic can also be used to describe properties that we want our system to satisfy. Hennessy-Milner logic has the following syntax:

$$\mathrm{HML} := \mathrm{T} \mid \neg\phi \mid \bigwedge_{i\in\mathbf{N}} \phi \mid \langle a\rangle\phi$$

The interpretation of the formulas is as follows. Formula T is satisfied by every state and the modal formula $\langle a\rangle\phi$ is satisfied by a state if this state can make an a-transition to a state that satisfies ϕ. Negation and conjunction are defined in the obvious way.

This logic characterizes bisimulation for labelled transition systems.

Example 2.5.1 *As we saw in a previous example, the two following processes are not bisimilar*

$$\begin{tikzcd}
& s \arrow[d, "a"] & \\
& \cdot \arrow[dl, "b"'] \arrow[dr, "c"] & \\
\cdot & & \cdot
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
& t \arrow[dl, "a"'] \arrow[dr, "a"] & \\
\cdot \arrow[d, "b"'] & & \cdot \arrow[d, "c"] \\
\cdot & & \cdot
\end{tikzcd}$$

and hence we can distinguish them using Hennessy-Milner logic: s satisfies the formula $\langle a\rangle(\langle b\rangle\top \wedge \langle c\rangle\top)$ but t does not. This formulas is satisfied by states that can jump with label a to a state that can perform both actions b and c.

If we remove negation from this logic and restrict to finite conjunction, it is not hard to prove that we obtain a logic (henceforth HML$^+$) that characterizes simulation for finitely branching systems. A state simulates another if it satisfies all the formulas the other satisfies. We have not seen this result proven explicitly in the literature. The proof is as follows. The non-trivial direction is to show that the relation R defined as sRs' if all the formulas of HML$^+$ satisfied by s are also satisfied by s' is a simulation relation between $(S, i, \rightarrow)$ and $(S', i', \rightarrow')$ (for $s \in S$, $s' \in S'$). Let sRs' and assume that $s \xrightarrow{a} t$. Let $\{\phi_1, \phi_2, \ldots\}$ be the formulas satisfied by t. Then $s \models \langle a\rangle\phi_i$ for all $i \geq 1$, and hence $s' \models \langle a\rangle\phi_i$ for all $i \geq 1$. Consider the formula $\langle a\rangle \wedge_{i=1}^{n} \phi_i$. This formula is satisfied by both s and s' for all $n \geq 1$. Then for all $n \geq 1$, there is some t' such that $s' \xrightarrow{a} t'$ and $t' \models \wedge_{i=1}^{n}\phi_i$. Now since S' is finitely branching, there is some t' such that $t' \models \wedge_{i=1}^{n}\phi_i$ for all $n \geq 1$, and hence t' satisfies all the formulas that t satisfies, i.e., tRt', as wanted.

It is not known if any logic characterizes simulation for non-probabilistic processes with infinite branching. However, even for finitely branching systems, there is no logic that characterizes both simulation and bisimulation. The reason is that negation is necessary in HML to characterize bisimulation; on the other hand, no logic containing negation can characterize simulation, for a state that satisfies all the formulas another state satisfies would then satisfy exactly the same formulas as the other. This corresponds to the fact that bisimulation is not equivalent to two-way simulation. We gave examples of processes that are two-way similar but not bisimilar in Example 2.1.4.

Recall that if we restrict to deterministic labelled transition systems, it is known that traces completely determine the systems [Mil89]. Hence, both simulation and bisimulation for deterministic processes are characterized by the logic: $\top \mid \langle a\rangle\phi$, where a is a label.

We will see that for labelled Markov processes, the situation is the same as in the deterministic case. The same logic can characterize both simulation –for countable processes, possibly infinitely branching– and bisimulation –for arbitrary processes, and hence bisimulation is equivalent to two-way simulation. This suggests that probabilistic processes are closer to deterministic processes than to indeterminate processes.

Chapter 3

Labelled Markov processes

This chapter is central to the thesis. It contains all the basic definitions and motivations about our model. We explain what labelled Markov processes are, the notions of bisimulation and simulation and how these definitions are generalizations of their non-probabilistic analogues.

A Markov process –as described more formally below– is a transition system with the property that the transitions depend only on the current state and not on the past history of the system. Moreover, the transitions are indeterminate and are governed by a probabilistic law. The labelled transition systems introduced in the last chapter also have transitions (though not probabilistic) that do not depend on the past history of the process. Transitions depend on the current state and the environment; the interaction with the environment is described by a set of labels. Labelled Markov processes combine the properties of both labelled transition systems and traditional Markov processes. Transitions are labelled to model the interaction with the environment: for each label and each state, there is one transition possible. This transition is indeterminate and the indeterminacy is quantified with a probability distribution that does not depend on the past history of the process. The most significant innovation is that we allow the state-space to be continuous in order to model physical systems. A discrete version of these processes –called probabilistic labelled transition systems– was introduced by Larsen and Skou in [LS91]. In this model, there is no indeterminacy beyond the probabilistically quantified internal choice. In any process algebra based with parallel composition and hiding, pure (unquantified) indeterminacy will arise. How this will be incorporated is the subject of ongoing research. See [DGJP99b] for an example of a process algebra with parallel composition but no hiding. In [GJP99], a probabilistic concurrent constraint programming is presented with parallel composition and hiding but the treatment there exploits special features of constraint programming and does not generalize in any simple way to process algebra.

In traditional Markov processes, the probability distributions always sum up to 1. In labelled Markov processes we will allow this sum to be less than 1. If the sum is 0, we will interpret this as meaning that a transition with this label cannot be performed. What is often done in traditional probability theory is that a state with no possibility of making a transition is modeled by having a transition back to itself. For questions concerning which states will eventually be reached (the bulk of the analysis in the traditional literature) this is convenient. If, however, we are modeling the interactions that the system has with its environment, it is essential that we make a distinction between a state that can make a transition and one that cannot. This is the same situation as in the non-probabilistic case. In a given state, some actions are enabled and some are not. What is usually studied is the corresponding probabilistic situation where the sum is either 0 or 1. We interpret the fact that the sum can be strictly between 0 and 1 with the notion of underspecification. If this sum is at some state, say, 3/4, for some action, it means that part of the behaviour is unknown. More precisely, the probability is 1/4 that the action is not accepted. An example of such a process is a button that "sometimes" rings a bell when pressed. Let us suppose that with probability 3/4 the action "ring" is accepted (i.e., the button goes down) and the bell rings, and with probability 1/4 the bell does not ring. We really want to model this as a one-state system having a transition labelled *ring* of probability 3/4 back to itself. It makes no sense to model the missing probability with another *ring*-transition from the state to itself having probability 1/4. There would be no difference between this system and the one that has probability one of accepting action *ring*. (One could be tempted to add a transition of probability 1/4 from the state to itself or to some other state labelled by *no-ring*, but this is another action; moreover, it does not represent a possible interaction with the environment.) The notion of simulation that we will also define for labelled Markov processes gives us another motivation for underspecifying the transition probabilities. Roughly speaking, we say that a process simulates another one if it can perform the same actions with equal or higher probability. So if the bell above can be activated by another button of higher reliability than the one we considered earlier, for example one having probability 7/8 of ringing, we would like to say that the new one simulates the old one. The only transitions that have to be matched in the simulation relation are the ones that are defined. If the process is underspecified, the part of the transition that is missing will simply not be simulated as it represents unwanted behaviour.

We first describe what traditional Markov processes are, mainly to justify the terminology. A knowledgeable reader can safely skip this section and jump directly to the next one where we recall the definition of probabilistic labelled transition systems introduced by Larsen and Skou. We then define labelled Markov processes which are the generalization to continuous state-space of probabilistic labelled transition systems. We show how these processes are organized in a category by defining simulation morphisms and zigzag morphisms and give the definition of the bisimulation relation we will adopt as well as the simulation relation. We end the chapter with a few examples of bisimulation and simulation.

3.1 Markov processes

Stochastic processes are dynamical systems where the evolution is governed by a probabilistic law. Most of the missing definitions can be found in Appendix A which recalls basic mathematical definitions that will be useful throughout the thesis. We review the standard definition of stochastic processes and relate it to the transition system view.

Definition 3.1.1 *Let $(\Omega, \mathcal{F}, P)$ be a probability space. A* ***stochastic process*** *is an indexed family of random variables $X_t : \Omega \longrightarrow (S, \Sigma)$, where t comes from an indexing set T and (S, Σ) is a measurable space. If T is countable, then it is called a* ***discrete-***

time *stochastic process.*

The index t often represents time, and hence if $\omega \in \Omega$, $X_t(\omega)$ represents the value of ω at time t. In this sense, Ω is the path space. We rarely use it. The actual state space of the transition system we have in mind is S.

For every $t \in T$, we have a probability distribution $P_t : \Sigma \to [0,1]$ defined as $P_t(A) = P(X_t^{-1}(A))$; this is often written $P(X_t \in A)$. One can think of this probability distribution as representing the state of a transition system. $P_t(A)$ is the probability that at time t, the system is in a state in the set A. If the indexing set is $\mathbf{N}$, the passage from P_i to P_{i+1} can be interpreted as a transition of the system. A Markov process is a stochastic process with the property that the transitions depend only on the current state and not on the past history of the process.

Definition 3.1.2 *Let $(\Omega, \mathcal{F}, P)$ be a probability space. A discrete-time* ***Markov process*** *is a stochastic process with* $\mathbf{N}$ *as index set that satisfies*

$$P(X_{n+1} \in A | X_1 = x_1, \ldots, X_n = x_n) = P(X_{n+1} \in A | X_n = x_n).$$

A Markov process is ***time-independent*** *if*

$$P(X_{n+1} \in A | X_n = x_n) = P(X_{n+i+1} \in A | X_{n+i} = x_{n+i}),$$

for all $i \geq 1$.

If a Markov process is time-independent, it can be described with just two consecutive random variables, for example X_1 and X_2. A time-independent Markov process can be viewed as a (probabilistic) transition system in the following way. The state space is the codomain of the Markov process (S, Σ), there is only one label and the transitions are as follows. The probability that the state x makes a transition to the set A is $P(X_2 \in A | X_1 = x)$.

3.2 Probabilistic labelled transition systems

We recall the definitions of *probabilistic labelled transition systems* and probabilistic bisimulation as introduced by Larsen and Skou in [LS91].

We saw in the last chapter that labelled transition systems can have non-deterministic transitions, that is, from one state there may be more than one transition with the same label pointing to different states. In a probabilistic labelled transition system, this indeterminacy is quantified. The process evolves following a probabilistic law: if it interacts with the environment by synchronizing on a label, it makes a transition to a new state according to a transition probability distribution. The transitions are specified by giving, for each label, a probability for going from one state to another.

Definition 3.2.1 *A* ***probabilistic labelled transition system*** *is a tuple* $(S, \mathcal{A}, P)$*, where* S *is a countable set of states (or processes),* $\mathcal{A}$ *is a set of labels (or actions), and for each* $a \in \mathcal{A}$*, we have a function,*

$$P_a : S \times S \longrightarrow [0, 1]$$

satisfying the property

$$\forall a \in \mathcal{A}, s \in S, \sum_{s' \in S} P_a(s, s') = 0 \ \ or \ \ 1.$$

Notice that there is no initial state in these systems. In fact, every state s in S determines a process having s as its initial state.

We interpret the equation $\sum_{s' \in S} P_a(s, s') = 0$ as the fact that state s cannot perform action a. If it can, the sum is 1. Note that there is no probability distribution associated to the external choice. This means that the choice between an action and another is entirely governed by the environment and we do not attach probabilities to it. This is the so-called reactive model (see for example [vGSST90] for a comparison of different models).

We will represent probabilistic labelled transition systems as transition graphs whose edges are labelled with an action and a probability. If the label of the transition is a and the probability p, we will label the arrow of the graph as $a[p]$. We will often drop the probability when $p = 1$ and just write a; on the other hand, in examples where there is only one action enabled, the arrows will be labeled only by the probability.

As we had for non-probabilistic processes, we would like a notion of equivalence between processes. An important first observation is that one cannot treat the probability like another label. To do so would mean that a relation is a bisimulation if whenever two state are related then they can match each other's moves to bisimilar states, where by matching we mean the the label and the probability are both matched. For example, consider the next picture.

$$\begin{tikzcd}
s \arrow[d, "{a[1]}"] & & & t \arrow[dl, "{a[\frac{1}{2}]}"'] \arrow[dr, "{a[\frac{1}{2}]}"] & \\
s_1 \arrow[d, "{b[1]}"] & & t_1 \arrow[d, "{b[1]}"'] & & t_1' \arrow[d, "{b[1]}"] \\
s_2 & & t_2 & & t_2'
\end{tikzcd}$$

If we just try to match the label and the probabilities, then s and t are not bisimilar because s can jump to s_1 with probability 1 whereas t cannot jump to any state with probability 1. However, we expect s and t to be bisimilar because both can jump with probability one to respectively the state s_1 and the states t_1, t_2, which are obviously all bisimilar. This tells us that we need to add the probabilities in some specific way. The definition of bisimulation will capture this; it says intuitively that two states are probabilistically bisimilar if, for every label, they can jump with equal probability to "maximal" sets of bisimilar states, i.e., sets that are closed under the equivalence relation.

In what follows we assume a fixed label set given once and for all and we will frequently suppress explicit mention of the labels.

Definition 3.2.2 *Let $\mathcal{S} = (S, P)$ be a probabilistic labelled transition system. Then a* ***probabilistic bisimulation*** *$\equiv_p$, is an equivalence on S such that, whenever $s \equiv_p t$, we have that for all $a \in \mathcal{A}$ and for every equivalence class $A \in S/\equiv_p$*

$$\sum_{s' \in A} P_a(s, s') = \sum_{s' \in A} P_a(t, s').$$

Two states s and t are said to be ***probabilistically bisimilar*** *($s \sim_{LS} t$) in case (s, t) is contained in some probabilistic bisimulation.*

We can interpret this as saying that two states are bisimilar if we get the same probability when we add up the transition probabilities to all the states in an equivalence class of bisimilar states. The addition is crucial – the probabilities are not just another label. The subtlety in the definition is that one has to somehow know what states are probabilistically bisimilar in order to know what the equivalence classes are, which in turn one needs in order to compute the probabilities to match them appropriately.

As an example of probabilistic bisimulation, we illustrate two processes which are a small variation of the previous example.

$$\begin{tikzcd}
 & s_0 \arrow[d, "a"] & \\
 & s_1 \arrow[dl, "b[\frac{1}{3}]"'] \arrow[dr, "b[\frac{2}{3}]"] & \\
s_2 & & s_2'
\end{tikzcd}
\qquad
\begin{tikzcd}
 & t_0 \arrow[dl, "a[\frac{1}{2}]"'] \arrow[dr, "a[\frac{1}{2}]"] & \\
t_1 \arrow[d, "b"'] & & t_1' \arrow[d, "b"] \\
t_2 & & t_2'
\end{tikzcd}$$

The equivalence relation that relates states having the same index is a probabilistic bisimulation (for example, s_1, t_1 and t_1' all have probability 1 of jumping to the equivalence class $\{s_2, s_2', t_2, t_2'\}$).

3.3 Labelled Markov processes

Labelled Markov Processes extend both Markov processes and probabilistic labelled transition systems. They are Markov processes to which we add interaction with the environment by use of labelled transitions. For every state and every label, the probability that a transition be performed depends only on the current state and not on the past history of the process.

Labelled Markov processes also generalize the notion of probabilistic labelled transition systems to continuous state spaces. When the state space is countable, we can specify transitions by giving, for each label, a probability for going from one state to another. In the case of a continuous state space like the reals, however, one cannot just specify transition probabilities from one state to another because in many interesting systems all such transition probabilities would be zero. More importantly, one cannot determine the probability of any set by adding the probabilities of the points. One can only add the probabilities of countably many disjoint sets. Instead, we must talk about the probability of going from a state s to a *set of* states A. Therefore we must work with probability measures and equip our state space with a σ-field of *measurable sets.* A review of the pertinent definitions appears in the appendix.

Transitions in labelled Markov processes will be modeled with *transition sub-probability functions.*

Definition 3.3.1 *A* ***transition sub-probability function*** *on a measurable space* (X, Σ) *is a function* $\tau : X \times \Sigma \longrightarrow [0,1]$ *such that for each fixed* $x \in X$, *the set function* $\tau(x, \cdot) : \Sigma \longrightarrow [0,1]$ *is a sub-probability measure, and for each fixed* $A \in \Sigma$ *the function* $\tau(\cdot, A) : X \longrightarrow [0,1]$ *is a measurable function.*

$\tau(x, A)$ represents the probability of the system, starting in state x, of making a transition into one of the states in A. The transition probability is really a *conditional probability* of the kind we encounter in traditional Markov processes; it gives the probability of the system being in one of the states of the set A after the transition, *given* that it was in the state x before the transition. In general the transition probabilities could depend on time, in the sense that the transition probability could be different at every step (but still independent of past history); we consider the time-independent case.

The key mathematical construction, as we shall see later, requires an analytic space structure on the set of states. Thus instead of imposing an arbitrary σ-field structure on the set of states, we require that the set of states be an analytic space and the σ-field be the Borel algebra generated by the topology.

Definition 3.3.2 *A* ***labelled Markov process*** *with label set* $\mathcal{A}$ *is a structure* $\mathcal{S} = (S, i, \Sigma, \{\tau_a \mid a \in \mathcal{A}\})$, *where* S *is the set of states, which is assumed to be an analytic space,* $i \in S$ *is the initial state, and* Σ *is the Borel* σ*-field on* S, *and*

$$\forall a \in \mathcal{A}, \tau_a : S \times \Sigma \longrightarrow [0,1]$$

is a transition sub-probability function.

We will fix the label set to be some $\mathcal{A}$ once and for all and write $\mathcal{S} = (S, i, \Sigma, \tau)$ for labelled Markov processes, instead of the more precise $(S, i, \Sigma, \{\tau_a \mid a \in \mathcal{A}\})$. The technical reasons why we assume that the state space is analytic will be discussed later when we will prove that bisimulation is characterized by a simple logic. Note that any discrete space is analytic and all the familiar continuous spaces, for example any of the Borel subsets of $\mathbf{R}^n$, and their images by a measurable function (which are not always Borel) are analytic as well.

One of the characteristics of an analytic space is that its Borel σ-field must contain all singletons. Consequently, when we consider a discrete process, that is, a process whose state space is countable or finite, the σ-field is always the powerset of S. In that case we omit the σ-field and simply write (S, i, τ). It is easy to see that probabilistic labelled transition systems are discrete labelled Markov processes. For these processes, we use the phrase "labelled Markov chain" rather than "discrete, labelled, Markov process" or "probabilistic labelled transition system". Since the transition probabilities are entirely determined by transitions to points, we often describe transitions in labelled Markov chains by specifying only the probabilities of transitions to singletons. In so doing, we usually omit the curly brackets around the singletons.

As in the non-probabilistic case, not all processes should accept any action from any state with probability one. Otherwise, they would all have the same observational behaviour. In the non-probabilistic case, this is implicit in the definition of transitions, which are triples (s, a, s'). An action a is not accepted in a state s if and only if there is no s' such that $(s, a, s') \in \longrightarrow$. In the probabilistic case, transitions are defined as sub-probability measures. Thus an action is not enabled in a state if the corresponding transition probability to jump from that state to the set of all states is 0. The next example illustrates a trivial process from the point of view of interaction. It is a process where all states always accept action a with probability 1.

Example 3.3.3 *Consider the labelled Markov process* $(\mathbf{R}, 0, \mathcal{B}, \tau)$ *having the reals as set of states, 0 as initial state, the Borel sets as σ-field, and τ_a defined as*

$$\tau_a(x_0, [u, v]) = \frac{1}{\pi} \int_u^v \exp(-(x - x_0)^2) dx,$$

where $x_0, u, v \in \mathbf{R}$ (here $\mathcal{A} = \{a\}$). This process appears complicated, but it is very simple if we consider its observed behaviour. Indeed, no matter what happens "inside" the process, no matter what the internal states actually are, this process will always accept label a with probability 1. Thus from the point of view of an external observer, it has an extremely simple behaviour: it is bisimilar to a one-state process which has an a-labelled transition from the state to itself with probability 1.

This example allows us to clarify the discussion at the beginning of the chapter. All of conventional stochastic process theory is concerned with systems like the one above. From our point of view they are trivial. This is to be expected, as we are modeling *interaction* and all such systems are indeed trivial from the point of view of interaction. In order to get nontrivial examples, one has to consider systems with richer label sets, and which are not always capable of making transitions with every label. Recall that in our model, this is reflected in the fact that for each state and action, the transition probability can sum up to less than 1 on the set of all states.

The example above also shows how bisimulation can be useful. We may have different descriptions of processes that have equivalent behaviour. In particular, a process with even an uncountable number of states may be bisimilar to a finite state process. This information is very valuable because reasoning about finite processes is much easier than reasoning about continuous ones.

3.4 The category LMP

We will organize the space of labelled Markov processes in a category. The motivation is mainly in the study of bisimulation between processes. In the next section we formulate bisimulation in a manner similar to the Larsen-Skou definition of probabilistic bisimulation, using relations and transitions to equivalence classes. In Chapter 7 we will give a categorical view of bisimulation for labelled Markov processes, following the ideas of Joyal, Nielsen and Winskel [JNW96], that we have recalled in Chapter 2 for non-probabilistic processes. It is convenient to have a functional version of bisimulation, especially to define quotients; we will indeed use the morphisms of the category defined below for this purpose in Section 4.2.

The objects of the category are labelled Markov processes and the morphisms will be simulation morphisms, as for non-probabilistic processes. Intuitively a simulation says that the simulating process can make all the transitions of the simulated process with greater probability than in the process being simulated. A simulation morphism will witness this fact. For example, consider the processes (that involve only one label) in the next picture, and the function f that maps states of the first process to states having the same index in the second process.

$$\begin{tikzcd}
 & s_0 \arrow[dl, "{[\frac{1}{4}]}"'] \arrow[d, "{[\frac{1}{4}]}"] \arrow[dr, "{[\frac{1}{4}]}"] & & & & & t_0 \arrow[dl, "{[\frac{1}{3}]}"'] \arrow[dr, "{[\frac{2}{3}]}"] & \\
s_1 \arrow[d, "{[1]}"'] & s_2 & s_2' & {} \arrow[r, "f"] & {} & t_1 \arrow[d, "{[1]}"'] & & t_2 \arrow[d, "{[1]}"] \\
s_3 & & & & & t_3 & & t_4
\end{tikzcd}$$

The mapping is done in such a way that if s has probability α of jumping to a set of states A, then $f(s)$ has probability $\geq \alpha$ of jumping to $f(A)$ (for example s_0 has probability 1/2 of jumping to $\{s_2, s_2'\}$, and $f(s_0) = t_0$ has probability 2/3 of jumping to $\{f(s_2), f(s_2')\} = \{t_2\}$). The function f is an example of what we will define to be a simulation morphism.

In this example, every set A that we can consider is countable, hence we can talk about $f(A)$ since it is obviously measurable. For uncountable processes, however, we cannot assume $f(A)$ to be measurable, and thus we must demand that $f(s)$ have probability $\geq \alpha$ of jumping to any measurable set containing $f(A)$; this is equivalent to the property given in the definition below. If a morphism furthermore satisfies the converse, i.e., if $f(s)$ has probability α of jumping to a set of states A', then s has probability $\geq \alpha$ of jumping to $f^{-1}(A')$, it will be called zigzag. Processes related by a zigzag morphism are intuitively expected to be bisimilar.

Definition 3.4.1 *A **simulation morphism** f between two labelled Markov processes, $\mathcal{S} = (S, i, \Sigma, \tau)$ and $\mathcal{S}' = (S', i', \Sigma', \tau')$ is a measurable function $f : (S, \Sigma) \longrightarrow (S', \Sigma')$ such that $f(i) = i'$, and for all $a \in \mathcal{A}$, $s \in S$ and for every measurable set*

$\sigma' \in \Sigma'$,

$$\tau_a(s, f^{-1}(\sigma')) \leq \tau'_a(f(x), \sigma').$$

f is **a zigzag morphism** *if the preceding inequality is an equality.*

We require the morphisms to be measurable[1] for the definition to make sense. If f were not measurable we would not be guaranteed that $f^{-1}(\sigma')$ is measurable. In [BDEP97], we required the zigzag morphisms to be surjective; we have replaced this requirement by initial states in the processes that must be preserved by morphisms. The effect of this is intuitively that every state of $\mathcal{S}$ must be bisimilar to its image, but we do not necessarily have that every state in $\mathcal{S}'$ has a preimage. However, because of the initial state preservation condition, we need to have so to speak "enough" states in $\mathcal{S}'$ for the initial states to be bisimilar. Hence, if a state in the image $\mathcal{S}'$ is "reachable" –whatever this means for continuous state-space systems–, then it will turn out to have a preimage because the condition must be satisfied for every path from the initial state.

Observe that if we are dealing with what is sometimes called a *total* process, that is, a process where all the transitions to the whole space are equal to either 0 or 1, then the inequality in the above definition is strict for some s, a and σ' if and only if s cannot perform action a, i.e., $\tau_a(s, S) = 0$. This means that if $f : \mathcal{S} \longrightarrow \mathcal{S}'$ is a simulation morphism and if $\mathcal{S}$ and $\mathcal{S}'$ are not bisimilar, then the difference between them is entirely witnessed by the fact that there are states in $\mathcal{S}$ that cannot perform actions that $f(s)$ can perform.

It is easy to check that labelled Markov processes with simulation morphisms form a category.

Definition 3.4.2 *The objects of the category* ***LMP*** *are labelled Markov processes, having a fixed set $\mathcal{A}$ as the set of labels, with simulations as the morphisms.*

[1]In older texts, such as Halmos [Hal74] or Rudin [Rud66] measurable is defined to mean that the inverse image of an *open* set is measurable. This means that the composite of two measurable functions need not be measurable. Our definitions are the current standard and, of course, with this definition, the composite of two measurable functions is measurable.

Simulation and zigzag morphisms for labelled Markov processes extend the corresponding standard notions for labelled transition systems that we recalled in Section 2.4. Given a labelled Markov chain (S, i, τ), we can define a labelled transition system (lts) with the same label set as follows. We take the same set of states S and we define a labelled transition relation $\rightarrow \subseteq (S \times \mathcal{A} \times S)$ by $(s, a, t) \in \rightarrow \iff \tau_a(s, t) > 0$. Recall that given two labelled transition systems, $(S, \rightarrow)$ and $(S', \rightarrow')$, a function $f : S \rightarrow S'$ is a simulation morphism if for every states $s, t \in S_1$, $s \xrightarrow{a} t$ implies $f(s) \xrightarrow{a}' f(t)$. The morphism is zigzag if it also satisfies the converse: whenever $f(s) \xrightarrow{a}' t'$, then there exists $t \in S$ such that $f(t) = t'$ and $s \xrightarrow{a} t$. We cannot define simulation morphisms this way for labelled Markov processes because we can easily have systems where all the point-to-point transition probabilities are zero but the Markov process is nontrivial because the transition probabilities are nonzero to "larger" sets.

Proposition 3.4.3 *Given two labelled Markov chains, a simulation morphism (resp. zigzag morphism) between them is also a simulation morphism (resp. zigzag morphism) between the associated labelled transition systems.*

Proof. Suppose that we have two labelled Markov chains (S, i, τ) and (S', i', τ') with f a simulation morphism from S to S'. Now suppose that in the associated lts the transition $s_1 \xrightarrow{a} s_2$ is possible. This means that $\tau_a(s_1, \{s_2\}) > 0$. Since f is a morphism we must have that $\tau'_a(f(s_1), \{f(s_2)\}) \geq \tau_a(s_1, f^{-1}(f(s_2))) \geq \tau_a(s_1, \{s_2\}) > 0$; hence in the lts $f(s_1) \xrightarrow{a} f(s_2)$ is possible. Now if f is zigzag, then for every $s' \in S'$ we have $\tau_a(s_1, f^{-1}(s')) = \tau'_a(f(s_1), \{s'\})$. So if in the associated lts the transition $f(s_1) \xrightarrow{a} s'$ is possible, then $\tau_a(s_1, f^{-1}(s')) > 0$ and hence there is some $s_2 \in f^{-1}(s')$ to which s_1 can make an a-transition. ∎

3.5 Bisimulation relation

The notion of bisimulation for labelled Markov processes is a generalization of the definition of Larsen and Skou for discrete processes, which is a compelling, natural notion. We saw earlier that they define bisimulation as an equivalence relation on the states satisfying the condition that equivalent states have equal probability of making an a-transition to any equivalence class of states. We will adapt this definition to the continuous case, thus we now have to take measurability into consideration. We will rather demand that equivalent states have equal probability of making an a-transition to any measurable set of equivalence classes of states. The reason is that, as we said before, in many continuous processes, transitions to singletons are all zero, and hence so are transitions to countable sets. In these systems, if we used the definition of Larsen and Skou unchanged, any equivalence relation whose equivalence classes are countable would be a bisimulation relation; hence we could relate any two states we want.

Instead of talking about sets of equivalence classes we will rather use the notion of *R-closed* sets. Let R be a relation on a set S. We say a set $X \subseteq S$ is R-closed if $R(X) = \{t | \exists s \in X, sRt\}$ is a subset of X. If R is reflexive, this becomes $R(X) = X$. If R is an equivalence relation, X is a union of equivalence classes.

Definition 3.5.1 *A **bisimulation relation** between two labelled Markov processes $\mathcal{S} = (S, i, \Sigma, \tau)$ and $\mathcal{S}' = (S', i', \Sigma', \tau')$ is an equivalence relation R on $S \uplus S'$ such that, for $s \in S$ and $s' \in S'$, with sRs', for every R-closed set $A \subseteq S \uplus S'$ such that $A \cap S \in \Sigma$ and $A \cap S' \in \Sigma'$, we have*

$$\tau_a(s, A \cap S) = \tau'_a(s', A \cap S')$$

for every $a \in \mathcal{A}$. Two states are bisimilar if they are related by a bisimulation relation. We say that $\mathcal{S}$ and $\mathcal{S}'$ are bisimilar if their initial states are.

Intuitively one is taking the bisimulation relation in "the direct sum" of the two processes. Bisimulation is obviously reflexive and symmetric. It is also transitive but we cannot see how to prove this directly from the definition. To prove transitivity, we will use a result that appears in Section 4.2. We will delay the proof until then.

The following example illustrates how the two processes of Example 3.3.3 are bisimilar.

Example 3.5.2 *We let the label set be the one element set $\{a\}$. Consider a system $\mathcal{S} = (S, i, \Sigma, \tau)$ with S an arbitrarily complicated state space and Σ a σ-field generated* *by some analytic space structure on S. For example, S could be $\mathbf{R}$, the reals with the Borel algebra. We define the transition function, $\tau_a(s, A)$ in any manner we please subject only to the conditions of the definition of a transition function and to the condition that $\forall s \in S.\tau_a(s, S) = 1$; i.e. for every s, the distribution $\tau_a(s, \cdot)$ is a probability measure. The process in Example 3.3.3 satisfies these conditions. Now consider the single state system $\mathcal{O}_a$ having one transition from its state o to itself, labelled a. It is easy to see that the relation relating every state of $\mathcal{S}$ to o is a bisimulation. The only R-closed set is $S \cup \{o\}$. The a-transition in O_a from o to itself has probability 1 and $\tau_a(s, S) = 1$ for any $s \in S$. Hence these two systems are bisimilar!*

We mentioned in the last section that intuitively, zigzag morphisms should relate bisimilar processes. The next proposition shows it is the case.

Proposition 3.5.3 *If there is a zigzag morphism from $\mathcal{S}$ to $\mathcal{S}'$, then $\mathcal{S}$ and $\mathcal{S}'$ are bisimilar.*

Proof . Let f be a zigzag morphism from $\mathcal{S}$ to $\mathcal{S}'$. Consider the equivalence relation on $S \cup S'$ generated by the pairs $(s, f(s))$, for every $s \in S$. We prove that R is a bisimulation relation. R obviously relates i and i'. Now let $s \in S$, so we have a pair $(s, f(s))$ related by R, and let Y be an R-closed set of $S \cup S'$ such that $Y \cap S \in \Sigma$ and $Y \cap S' \in \Sigma'$. Then $Y \cap S = f^{-1}(Y \cap S')$. Since f is a zigzag morphism, we have $\tau_a(s, Y \cap S) = \tau'_a(f(s), Y \cap S')$. ∎

The converse is not necessarily true, but we will see in Chapter 7 that $\mathcal{S}$ and $\mathcal{S}'$ are bisimilar if and only if there exists a *span* of zigzag morphisms between them. This means that there is a third object $\mathcal{U}$ together with zigzag morphisms from $\mathcal{U}$ to each of $\mathcal{S}$ and $\mathcal{S}'$, as in the following diagram.

$$\begin{tikzcd} & \mathcal{U} \arrow[dl] \arrow[dr] & \\ \mathcal{S} & & \mathcal{S}' \end{tikzcd}$$

It is interesting to note that we can take a coalgebraic view of bisimulation [AM89, Rut95, Rut96] as well. We can view a labelled Markov process as a coalgebra of a suitable functor; in fact it is a functor introduced by Giry [Gir81] in order to define a monad on **Mes** analogous to the powerset monad. From this point of view, bisimulation is a span of coalgebra homomorphisms. But if one checks what this means, these are precisely our zigzag morphisms in **LMP**.

Proposition 3.5.4 *If two labelled Markov chains are bisimilar then the associated labelled transition systems are as well.*

Proof. Suppose that we have two labelled Markov chains (S, i, τ) and (S', i', τ') and a bisimulation R between them. Now suppose that sRs' and that in the associated lts the transition $s \xrightarrow{a} t$ is possible. This means that $\tau_a(s, \{t\}) > 0$. Let T be the equivalence class in $S \cup S'$ containing t. We have $\tau_a(s, T \cap S) = \tau'_a(s', T \cap S')$; hence in the lts $s' \xrightarrow{a} T$ is possible so there is some $t' \in T$, and hence $s'Rt'$, such that $s' \xrightarrow{a} t'$, as wanted. ∎

Note that the converse is not true, because different assignments of probabilities in a labelled Markov chain can yield non bisimilar processes that have the same associated lts. However, the following is true.

Proposition 3.5.5 *If two labelled transition systems are bisimilar, there exist bisimilar labelled Markov chains having them as their associated labelled transition systems.*

Proof. Let R be a bisimulation relation between two labelled transition systems $(S, \mathcal{A}, \rightarrow)$ and $(S', \mathcal{A}, \rightarrow')$. Let R^* be the smallest equivalence relation containing R on $S \cup S'$. We will define (S, i, τ) and (S', i', τ') so that they have the former processes as their associated lts and so that R^* is a bisimulation between them. It is easy to check that if sR^*s', then for every equivalence class C of R^*, we have

$$s \xrightarrow{a} C \Leftrightarrow s' \xrightarrow{a}{}' C.$$

We define $\tau_a(s, \cdot)$ and $\tau'_a(s', \cdot)$ in two steps. If s and s' can jump to a finite number of equivalence classes $C_1, \ldots, C_n$ we set $\tau_a(s, C_i) = \tau'_a(s', C_i) = 1/n$. Otherwise, we assign the number $1/2^i$ to that transition. In fact, this step is only to help us define transition probabilities to single states. The state s does not really jump to every state in C_i, because some of them are in S'.

Now let t be a state such that $s \xrightarrow{a} t$ in the lts. Then t is in exactly one equivalence class C of R^*. Let $B \subset C$ be the set of states in $C \cap S$ to which s can jump with action a. Note that B contains t. If $B = \{b_1, \cdots, b_n\}$ we set $\tau_a(s, \{b_i\}) = \tau_a(s, C)/n$. If $B = \{b_1, b_2, \cdots\}$ we assign the number $\tau_a(s, C)/2^i$ to that transition. It is not hard to check that the labelled Markov chains so defined are bisimilar through the relation R^*. ∎

Remark 3.5.6 *The construction in the previous proof allows us to obtain a probabilistic process from a non-probabilistic process. Note that the proof is based on the fact that labelled transition systems have a finite or countable set of states. If the branching of the non-probabilistic process is finite, every indeterminate transition is given the uniform distribution. (One only has to consider the identity as equivalence relation on a single process and then apply the construction.) Thus somebody studying finite branching systems whose indeterminate transitions are assumed to be of equal probability could use their probabilistic translation instead, and hence benefit from some results that we have. For example, the fact that bisimulation and simulation are characterized by a simple logic that involves no negation.*

3.6 Simulation relation

The intuition behind the definition of simulation is that a state simulates another if it captures all the behaviour of the other. In terms of probabilities, we require that the simulated state has a smaller or equal probability of jumping to a set A with some label than has the simulating state to the set of states that simulate A. Once again, as in bisimulation, we somehow need to know in advance which state simulates which state in order to check the inequality. In fact, we cannot talk about the set of states that simulate A in general even if A is measurable because we cannot make sure that this set is measurable. So we use the notion of closedness of relation that was introduced in the last section for the definition of bisimulation. Recall that a set is R-closed if it contains all states related by R to some state in it. It will also be convenient to explicitly define the notion of *direct sum* of two labelled Markov processes.

Definition 3.6.1 *Let $\mathcal{S} = (S, i, \Sigma, \tau)$ and $\mathcal{S}' = (S', i', \Sigma', \tau')$ be two labelled Markov processes. The* ***direct sum*** *$\mathcal{S} + \mathcal{S}'$ of these processes is a process $\mathcal{U} = (U, u_0, \Omega, \rho)$ with $U = S \uplus S' \uplus \{u_0\}$, u_0 is a new state, Ω is the σ-field generated by $\Sigma \cup \Sigma'$, and the transitions are as follows: $\forall a \in \mathcal{A}$, $\rho_a(u_0, \{i\}) = \rho_a(u_0, \{i'\}) = \frac{1}{2}$, and for all $s \in S$, $s' \in S'$, $\rho_a(s, A \uplus A') = \tau_a(s, A)$ and $\rho_a(s', A \uplus A') = \tau'_a(s', A')$.*

The choice of $\frac{1}{2}$ as the transition probability is arbitrary. This construction is purely formal and is only used in order to define a relation on the common state space. With this definition we do not, for example, have an associative direct sum. However this is of no importance for the use that we make of this definition.

In addition to simulation we define the notion of *strict simulation* which says intuitively that a state is strictly simulated by another if whenever the state can make a transition with probability p, the simulating state can make the same transition with probability greater than $p + \epsilon$. Simulation and strict simulation are patterned on the notion of less than or equal and way below from domain theory [Jon90].

Definition 3.6.2 *Let $\mathcal{S} = (S, i, \Sigma, \tau)$ be a labelled Markov process. A reflexive and transitive relation (a preorder) R on S is a* ***simulation relation*** *if whenever sRs', with $s, s' \in S$, we have that for all $a \in \mathcal{A}$ and every R-closed measurable set $A \in \Sigma$,*

$$\tau_a(s, A) \leq \tau_a(s', A).$$

We then say that s is simulated by s'. R is a ***strict*** *simulation if there is an $\epsilon > 0$ such that $\tau_a(s, A) < \tau_a(s', A) - \epsilon$ whenever $\tau_a(s, A) > 0$. We then call R an ϵ-strict simulation and we write R_ϵ instead of R.*

Let $\mathcal{S} = (S, i, \Sigma, \tau)$ and $\mathcal{S}' = (S', i', \Sigma', \tau')$ be a pair of labelled Markov process. $\mathcal{S}$ is (strictly) simulated by $\mathcal{S}'$ if there is a (strict) simulation relation on some process $\mathcal{U}$ of which $\mathcal{S}$ and $\mathcal{S}'$ are direct summands, relating i and i' in $\mathcal{U}$.

Note that we do not require $\mathcal{U}$ to be exactly $\mathcal{S} + \mathcal{S}'$ but rather a direct sum of a number of processes, including $\mathcal{S}$ and $\mathcal{S}'$. The reason for this is that transitivity of simulation would not follow in any obvious way with $\mathcal{U}$ being exactly the direct sum. However, we will prove (see Corollary 4.3.6) that in the particular case where the simulated process $\mathcal{S}$ is discrete, if a simulation exists between $\mathcal{S}$ and $\mathcal{S}'$, then there is a simulation on $\mathcal{S} + \mathcal{S}'$. It is clear from the definitions above that the choice of 1/2 in the definition of direct sum does not affect simulation. The fact that processes or states are related or not by a simulation relation does not depend on this number.

The next two propositions are easy but important for the theory. They prove that simulation and strict simulation are transitive and that every bisimulation is a simulation.

Proposition 3.6.3 *Simulation and strict simulation are transitive.*

In fact we also have that if $\mathcal{S}$ strictly simulates $\mathcal{S}'$ which is simulated in turn by $\mathcal{S}''$, then $\mathcal{S}$ is strictly simulated by $\mathcal{S}''$.

Proof. We prove that simulation is transitive. First let us consider two simulations R_1 and R_2 on a single process $\mathcal{S} = (S, i, \Sigma, \tau)$. Let R be the transitive closure of $R_1 \cup R_2$. Then every measurable R-closed set is also R_i-closed, $i = 1, 2$, then it follows easily that R is a simulation on $\mathcal{S}$.

Now let R_1 be a simulation between $\mathcal{S}$ and $\mathcal{S}'$ through process $\mathcal{U}_1$ and R_2 a simulation between $\mathcal{S}'$ and $\mathcal{S}''$ through process $\mathcal{U}_2$. Then construct the direct sum $\mathcal{U}$ of $\mathcal{U}_1$ and $\mathcal{U}_2$ and consider R the reflexive and transitive closure of $R_1 \circ R_2$ on $\mathcal{U}$ as above. Then R is a simulation on $\mathcal{U}$ that relates i and i'', and $\mathcal{S}$, $\mathcal{S}''$ are direct summands of $\mathcal{U}$.

The proof for strict simulation is similar, just note that we must add the ϵ's to obtain the composite relation. ∎

Proposition 3.6.4 *Every bisimulation relation is a simulation relation.*

Proof. Let R be a bisimulation relation between $\mathcal{S}$ and $\mathcal{S}'$. We prove that it is a simulation relation on the direct sum of $\mathcal{S}$ and $\mathcal{S}'$. Let sRs' and let A be an R-closed set of $\mathcal{S}+\mathcal{S}'$. Then A is a subset of $S \uplus S'$ and satisfies that $A \cap S \in \Sigma$ and $A \cap S' \in \Sigma'$. Hence we have $\tau_a(s, A \cap S) = \tau_a'(s', A \cap S')$ and the required inequality is satisfied. ∎

The definition of "strong simulation" given by Segala and Lynch in [SL94] is slightly stronger. It requires that if a simulating state can perform an action, then so do the states it simulates. This would correspond to the additional condition that if $\tau_a(s', S) > 0$, then $\tau_a(s, S) > 0$. However, we have noted that their definition is usually mentioned without this condition: in that case, the two definitions are the same. This applies to systems which are the common denominator of our model of systems and theirs: they consider only discrete systems and we do not allow different transition probability from a single state with the same label. The proof that the two definitions coincide uses the max-flow min-cut theorem by Ford and Fulkerson. It is easy to modify the proof of Theorem 7.3.4 to get the result. We believe that our definition can be easily extended to indeterminate processes and still coincide with their definition of a simulation R which is in terms of the existence of a weight function on $S \times S$ that must satisfy some properties with respect to the relation R.

Example 3.6.5 *We illustrate a simulation on the following process:*

$$\begin{tikzcd}
 & s_0 \arrow[dl, "{a[\frac{1}{6}]}"'] \arrow[d, "{a[\frac{2}{3}]}"] \arrow[dr, "{a[\frac{1}{6}]}"] & & \\
s_1 & s_2 \arrow[dl, "a"'] \arrow[d, "b"] & s_3 \arrow[d, "{a[\frac{1}{3}]}"'] \arrow[dr, "{a[\frac{2}{3}]}"] & \\
s_4 \arrow[d, "b"] & s_5 & s_6 & s_7 \arrow[d, "b"] \\
s_8 & & & s_9
\end{tikzcd}$$

Consider the reflexive closure of the relation defined as follows. NIL states – s_1, s_5, s_6, s_8, s_9 – are related to every state. s_4, and s_7 are related to each other and to s_2. Finally, s_3 is related to s_0 and s_2. The only closed sets with respect to this relation are the whole set, $\{s_0\}$, $\{s_2\}$, $\{s_0, s_2, s_3\}$, $\{s_2, s_4, s_7\}$ and unions of these sets. It is easy to check that if s is related to t in this manner, then s has smaller probability than t has of jumping to each of these sets, with any label.

Remark 3.6.6 In the definition of simulation we could have included the requirement that $R(A)$ be measurable, but if we had, bisimulation could not be proved to be a simulation, which is a basic requirement, of course. The reason why, in turn, we did not demand that $R(A)$ be measurable for every measurable set A in the definition of bisimulation is that then we would have to prove that the logic (see Chapter 4) does induce that property, that is, for every measurable set, the set of states that satisfy the same formulas is measurable, which we cannot prove in general though it may be true. Note however that if it was true, it would mean that the quotient map from a process to its quotient under the equivalence induced by the logic would send measurable sets to measurable sets.

The following proposition shows that a simulation morphism does relate a process to a process that simulates it.

Proposition 3.6.7 *If there is a simulation morphism from $\mathcal{S}$ to $\mathcal{S}'$, then $\mathcal{S}$ is simulated by $\mathcal{S}'$.*

Proof . Let f be a simulation morphism from $\mathcal{S}$ to $\mathcal{S}'$. Consider the reflexive relation on $S \cup S'$ generated by the pairs $(s, f(s))$, for every $s \in S$. We prove that R is a simulation relation. R obviously relates i and i'. Now let Y be an R-closed set of $S \cup S'$ such that $Y \cap S \in \Sigma$ and $Y \cap S' \in \Sigma'$. Then $Y \cap S \subseteq f^{-1}(Y \cap S')$. Since f is a simulation morphism, we have $\tau_a(s, Y \cap S) \leq \tau_a(s, f^{-1}(Y \cap S')) \leq \tau'_a(f(s), Y \cap S')$. ∎

The converse is not true in general, but we conjecture that if $\mathcal{S}$ is simulated by $\mathcal{S}'$ then there exists a span of morphisms between them, one of these morphisms –from $\mathcal{U}$ to $\mathcal{S}$– being zigzag and the other one being a simulation morphism.

$$\begin{tikzcd} & \mathcal{U} \arrow[dl, "\text{zigzag}"'] \arrow[dr] & \\ \mathcal{S} & & \mathcal{S}' \end{tikzcd}$$

We will give a proof of this for finite processes in Chapter 7.

3.7 Examples of bisimulation and simulation

The first two examples we give are of bisimilar pairs of labelled Markov processes. The first one illustrates a bisimulation between two continuous systems that cannot be reduced to discrete ones. The second one is an example of how bisimulation can be used for verifying that an implementation matches its specifications.

Example 3.7.1 *Consider the labelled Markov process $\mathcal{S} = (\mathbf{R}, 1, \mathcal{B}, \tau)$, over the trivial label set, defined as follows. The states are real numbers, the measurable sets are Borel sets and the transition function is defined on intervals (and then extended to arbitrary Borel sets) as follows:*

$$\tau(x, [r,s]) = \begin{cases} \lambda/2 \int_r^s e^{-\lambda|x-y|} dy & \text{if } x \geq 0, \\ 0 & \text{otherwise.} \end{cases}$$

where the constant factor of λ is chosen to make τ be 1 on the whole space. Intuitively this is a system where a particle makes random jumps with probability exponentially distributed with the length. However, there is an "absorbing wall" at the point $x = 0$ so that if the system jumps to the left of this point it gets stuck there. Note that every positive state has a different probability density for jumping to a negative state. Now consider the system $\mathcal{U} = (\mathbf{R}^2, (1,1), \mathcal{B}^2, \rho)$ defined as

$$\rho((x,y), [r,s] \times [p,q]) = \tau(x, [r,s]) P([p,q]),$$

where P is some arbitrary probability measure over $\mathbf{R}$. This system should behave "observably" just like the first system because, roughly speaking, the first coordinate behaves just like the first system and the second has trivial dynamics, i.e. it is bisimilar to the one-state, one-transition system. Indeed these two systems are bisimilar with the relation on $\mathbf{R} \cup \mathbf{R}^2$ generated by pairs of the form $(x, (x,y))$.

As an application of probabilistic bisimulation, we give a probabilistic version of Example 2.4.5. Recall that here we want to check if a given implementation matches its specification. A specification and an implementation are defined, and then we check if the processes derived are bisimilar.

In the next example, we use a parallel composition combinator. We have not, as yet, carefully studied how processes could be combined, but we will use a similar definition for composition as the one we recalled for non-probabilistic processes in Chapter 2. Since we are dealing with "deterministic" probabilistic processes, we must attach probabilities to a-transitions if an a-action is enabled on both sides of a parallel composition. Let us assume that if this occurs, each action has probability 1/2 of being performed.

Example 3.7.2 *Suppose we have a cell which is not totally reliable and so can lose a message put into it with probability* $1/4$*:*

$$\begin{tikzcd}
\mathrm{cell}_e \arrow[loop below, "put{[1/4]}"] \arrow[r, bend left, "put{[3/4]}"] & \mathrm{cell}_f \arrow[l, bend left, "get"]
\end{tikzcd}$$

cell_e *and* cell_f *are understood to be respectively the empty and the full cell. Now suppose we want to implement a bag of size two having the same reliability using this cell. Here is our specification of a bag of size two:*

$$\begin{tikzcd}
\mathrm{bag}_0 \arrow[loop below, "put{[1/4]}"] \arrow[r, bend left, "put{[3/4]}"] & \mathrm{bag}_1 \arrow[l, bend left, "get"] \arrow[loop below, "put{[1/4]}"] \arrow[r, bend left, "put{[3/4]}"] & \mathrm{bag}_2 \arrow[l, bend left, "get"]
\end{tikzcd}$$

where the index attached to bag *represents the number of messages present in the bag. We expect that we can implement this bag by putting two cells in parallel* $\mathrm{cell}_e|\mathrm{cell}_e$. *As we noted above, we assume that if an action is enabled on both sides of a parallel composition, each action has probability* $1/2$ *of being performed. The states are now pairs of states and a transition happens with probability* $1/2 \cdot p$ *from one state if either one or the other coordinate of the pair can perform the action to the same coordinate of the arriving state with probability* p. *Hence, if the environment requires a put action, process* $\mathrm{cell}_e|\mathrm{cell}_e$ *has probability* $1/2 \cdot 3/4$ *of becoming process* $\mathrm{cell}_e|\mathrm{cell}_f$ *and the same probability of becoming* $\mathrm{cell}_f|\mathrm{cell}_e$. *The resulting process is illustrated in Figure 3.1. In order to show that* $\mathrm{cell}_e|\mathrm{cell}_e$ *is a good implementation of* bag_0, *we just have to verify that they are bisimilar using the equivalence relation generated by the pairs:* $(\mathrm{cell}_e|\mathrm{cell}_e, \mathrm{bag}_0)$, $(\mathrm{cell}_f|\mathrm{cell}_f, \mathrm{bag}_2)$ *and the two other states where exactly one*

$$\begin{tikzcd}
 & \mathrm{cell}_e|\mathrm{cell}_e \arrow[loop above, "put{[1/4]}"] \arrow[dl, bend right, "put{[3/8]}"'] \arrow[dr, bend left, "put{[3/8]}"] & \\
\mathrm{pcell}_f|\mathrm{cell}_e \arrow[loop below, "put{[1/4]}"] \arrow[ur, "get"'] \arrow[dr, bend left, "put{[3/4]}"] & & \mathrm{cell}_e|\mathrm{cell}_f \arrow[loop below, "put{[1/4]}"] \arrow[ul, "get"] \arrow[dl, bend right, "put{[3/4]}"'] \\
 & \mathrm{cell}_f|\mathrm{cell}_f \arrow[ul, "get"] \arrow[ur, "get"'] &
\end{tikzcd}$$

Figure 3.1: The composition of two cells

cell is full both paired with bag_1. *Thus we can conclude that our implementation is correct.*

Example 3.7.3 *Let us reconsider Example 3.7.2. This example can illustrate the role of simulation and how it is linked with partial systems. The way we have specified the processes in that example doesn't allow us to simulate the bag using a more reliable one. A simulating process would try to perform the same transitions with higher probability, not considering the fact that the loss of messages, represented by a put-transition from the process to itself, is not desired. Consider the following partial specifications:*

$$\begin{tikzcd}
\mathrm{cell}_e \arrow[r, shift left, "put{[5/6]}"] & \mathrm{cell}_f \arrow[l, shift left, "get"]
\end{tikzcd}
\qquad
\begin{tikzcd}
\mathrm{bag}_0 \arrow[r, shift left, "put{[3/4]}"] & \mathrm{bag}_1 \arrow[l, shift left, "get"] \arrow[r, shift left, "put{[3/4]}"] & \mathrm{bag}_2 \arrow[l, shift left, "get"]
\end{tikzcd}$$

With these specifications, we can simulate our bag with a process that has a smaller probability of losing the message. Of course, we do not have bag_0 *bisimilar to* $\mathrm{cell}_e|\mathrm{cell}_e$ *but certainly* $\mathrm{cell}_e|\mathrm{cell}_e$ *simulates* bag_0.

The following example illustrates how a continuous process can be simulated by a finite one.

Example 3.7.4 *Consider the following process. The state-space is* $S = [0,1] \cup \{i, f\}$. *The initial state is* i *and* f *is a NIL state, and there is one label,* a. *Transitions go from* i *to* $[0,1]$ *and from states of* $[0,1]$ *to* f, *and are generated by the following.*

$$\tau_a(i, [x,y]) = y - x; \qquad \textit{if } x \in [0,1],\ \tau_a(x, f) = x.$$

This process is really continuous because every two states $x, y \in [0,1]$ have different probabilities of jumping to the whole set S, which is closed with respect to any relation. Indeed, if $x \in [0,1]$, $\tau_a(x,S) = x$. This process can be simulated by a finite process $\mathcal{P}$ consisting of three states p_0, p_1 and p_2. Transitions are from p_0 to p_1 and from p_1 to p_2 with probability 1. We informally illustrate these processes in the following picture.

$$\begin{tikzcd}
i \arrow[d, "{[y-x]\ \mathit{to}\ [x,y]}"] & p_0 \arrow[d, "{[1]}"] \\
{[0,1]} \arrow[d, "{[x]}"] & p_1 \arrow[d, "{[1]}"] \\
f & p_2
\end{tikzcd}$$

The simulation relation that relates (in $\mathcal{S}+\mathcal{P}$) i and p_0, every state of $[0,1]$ to p_1, and f to p_2 is a simulation relation. This example could be easily extended to processes where we allow only the transition probability function to be either 0 or 1 on the whole space by introducing other states: f_1 to which every $x \in [0,1]$ has probability $1-x$ of jumping, and f_2 to which f_1 can jump with probability 1. Then we need to also add one state p_3 in $\mathcal{P}$ to which p_2 can jump with probability 1. These processes are as in the following picture.

$$\begin{tikzcd}
& i \arrow[d, "{[y-x]\ \mathit{to}\ [x,y]}"] & & p_0 \arrow[d, "{[1]}"] \\
& {[0,1]} \arrow[dl, "{[x]}"'] \arrow[dr, "{[1-x]}"] & & p_1 \arrow[d, "{[1]}"] \\
f & & f_1 \arrow[d, "{[1]}"] & p_2 \arrow[d, "{[1]}"] \\
& & f_2 & p_3
\end{tikzcd}$$

The simulation relation is extended in the obvious way.

The following is a paradigmatic example of applying controls to keep a system stable or safe.

Example 3.7.5 *We describe a finite specification that we will want to implement as a continuous process. With this example, we want to illustrate the concept of simulation;*

we do not claim that this is the specification and implementation paradigm for labelled Markov processes. We have a continuous process $\mathcal{S}$. The state-space is the real line, the initial state is the origin. The process describes a particle that jumps randomly with label a. States less than -1 *or greater than* 1 *are dead states, thus no transition is enabled in these states. When the state is in* $[-1, 1]$, *label a is enabled and the transition is exponentially distributed with the length and closeness of intervals (for an example of such a distribution, see 3.7.1). We want to combine $\mathcal{S}$ with a controller in such a way that at each step, the probability of jumping to a dead state is less than* 0.1. *This specification of the combined process can be modeled as a finite process $\mathcal{P}$ that has one state, and one a-transition from that state to itself of probability* 0.9. *We want our implementation of the controller combined with the process $\mathcal{S}$ to simulate $\mathcal{P}$. We describe a possible controller. There is a threshold value, $t \in (0, 1)$. The controller has the power of changing the a-transition of the particle. It leaves the particle free to jump as it wants, as long as it stays inside the interval $(-t, t)$. If the particle crosses the walls $-t$ or t, the a-transition is then to jump by* $1/2$ *back into the safe interval, i.e., label a is enabled, and then the jump is of length* $1/2$, *to the right if the state is in $(-1, -t)$ or to the left if the state is in $(t, 1)$. We want to find a possible value for t so that the implementation of $\mathcal{S}$ with the controller simulates $\mathcal{P}$. Of course we could take t very small, but it is likely that we want the number of adjustments to be minimized. This example illustrates the underlying intuition in many feedback control systems. The particle could be an aircraft whose height must stay within a given range, a chemical plant inside which we want to control the pressure, etc.*

Chapter 4

A modal logic for bisimulation and simulation

We saw in Chapter 2 that in the case of non-probabilistic processes, bisimulation is characterized by Hennessy-Milner logic. Two states are bisimilar if and only if they satisfy the same formulas of that logic.

We will define a simple probabilistic variant to Hennessy-Milner logic, written $\mathcal{L}$, which will be proven to characterize bisimulation for labelled Markov processes. The striking aspect of this logic is that it does not contain any form of negation. Another surprise is that even if we allow infinite branching (in fact we allow continuous branching), we do not need infinite conjunction to characterize bisimulation. These results are not what one expects from the non-probabilistic case. The point is that the probabilistic systems we are considering –without explicit nondeterminism– resemble deterministic systems quite closely, rather than nondeterministic systems. In the latter case –as is well-known [Mil90]– negation is necessary to characterize bisimulation and if we allow infinite branching, we also need infinite conjunction.

In [LS91], Larsen and Skou proposed a logic that characterizes bisimulation for discrete processes. Their logic contains a weak form of negation but more importantly they work under an assumption slightly stronger than finiteness of branching, called the minimal deviation assumption. Hence our result is an improvement over their work even in the discrete case, because our logic does not contain negation. Moreover, it characterizes bisimulation for arbitrary labelled Markov processes.

One advantage of the fact that bisimulation can be characterized by a negation-free logic is that this opens the way to a notion of logical simulation between processes. We say that a state *logically simulates* another state if it satisfies (at least) all the formulas the other satisfies. Of course, if a logic contains negation and a state logically simulates another state according to that logic, then the two states must satisfy exactly the same formulas. Consequently, this logic cannot characterize simulation. Thus this is not a notion that would have been considered with most Hennessy-Milner type logics in the literature. We will prove that a simple extension of $\mathcal{L}$ –$\mathcal{L}$ augmented with disjunction– characterizes simulation (as well as bisimulation) between discrete labelled Markov processes. In fact the two processes need not be discrete, it is enough that one of them be discrete.

Thus, for probabilistic processes, the same logic can characterize both simulation and bisimulation. This is because this logic doesn't contain negation, and more interestingly because two-way simulation is equivalent to bisimulation. This also happens for non-probabilistic determinate processes but not for indeterminate ones. Recall that for non-probabilistic processes with infinite branching, it is not even known if any logic characterizes simulation. This shows that probabilistic systems as we define them are very close to deterministic systems.

We first define five modal logics none of which are equivalent to any of the others and then prove that the simplest one, $\mathcal{L}$, characterize bisimulation. In the third section, we prove that by adding disjunction to $\mathcal{L}$, we can characterize simulation for discrete processes. We use these characterizations to define two algorithms one that can decide whether two states of a finite process are bisimilar, and the other that can decide whether a state simulates another. Finally, we prove that all the logics we define characterize bisimulation and discuss which one can characterize equivalence classes of bisimilar states.

4.1 Modal logics

We now describe five modal logics that will each be proven to characterize bisimulation. Thus *all* these logics play the role of Hennessy-Milner logic for non-probabilistic bisimulation [HM85].

We assume as before that there is a fixed set of "labels" or "actions", we usually use letters like a or b for actions. The simplest logic will be called $\mathcal{L}$ and has as syntax the following formulas:

$$\mathsf{T} \mid \phi_1 \wedge \phi_2 \mid \langle a \rangle_q \phi$$

where a is an action from the fixed (countable) set of actions $\mathcal{A}$ and q is a rational number. Given a labelled Markov process (S, i, Σ, τ) we write $s \models \phi$ to mean that the state s satisfies the formula ϕ. The definition of the relation $\models$ is given by induction on formulas. The definition is obvious for the propositional constant T and conjunction. We say $s \models \langle a \rangle_q \phi$ if and only if there exists $A \in \Sigma$ such that for all $s' \in A$, we have $s' \models \phi$ and $\tau_a(s, A) > q$[1]. In other words, the system in state s can make an a-move to a state, that satisfies ϕ, with probability greater than q. We write $[\![\phi]\!]_S$ for the set $\{s \in S | s \models \phi\}$. We often omit the subscript when no confusion can arise.

Note that since we restrict to rationals in the logic, we have a countable number of formulas. This fact is used in the proof that the logic characterizes bisimulation. Of course expressiveness is affected by this choice, for there is no rational formula that is equivalent to the formula $\langle a \rangle_{\pi/4} \mathsf{T}$. However, since the logic can distinguish non-bisimilar states, it is expressive enough for practical purposes.

In the following table we define four additional logics. They are all syntactic extensions of $\mathcal{L}$.

$$\begin{array}{lcl} \mathcal{L}_\vee & := & \mathcal{L} \mid \phi_1 \vee \phi_2 \\ \mathcal{L}_\Delta & := & \mathcal{L} \mid \Delta_a \\ \mathcal{L}_\neg & := & \mathcal{L} \mid \neg\phi \\ \mathcal{L}_\bigwedge & := & \mathcal{L}_\neg \mid \bigwedge_{i \in \mathbf{N}} \phi_i \end{array}$$

Given a labelled Markov process (S, i, Σ, τ) we write:

$s \models \phi_1 \vee \phi_2$	to mean that $s \models \phi_1$ or $s \models \phi_2$;
$s \models \Delta_a$	to mean that $\tau_a(s, S) = 0$;
$s \models \neg\phi$	to mean that $s \not\models \phi$;
$s \models \bigwedge_{i \in \mathbf{N}} \phi_i$	to mean that $s \models \phi_i$ for all $i \in \mathbf{N}$.

[1]In [BDEP97], we used $\tau_a(s, A) \geq q$. The present choice fits better with the work in Chapter 5.

In Chapter 6 we will use a variation of $\mathcal{L}_\vee$ where the disjunction can be countable. This logic will be written $\mathcal{L}_{\bigvee}$. Although they all characterize bisimulation, they do not have the same expressive power. Clearly all of them are at least as expressive as $\mathcal{L}$, and $\mathcal{L}_{\bigwedge}$ is more expressive than all the others. $\mathcal{L}_\vee$, $\mathcal{L}_\Delta$ and $\mathcal{L}_\neg$ are incomparable. It is interesting to note that none of these differences will have any impact on the characterization of bisimulation, as we have already said. However, we need at least $\mathcal{L}_\vee$ to characterize simulation.

The logic that Larsen and Skou used in [LS91] is the combination of $\mathcal{L}_\vee$ and $\mathcal{L}_\Delta$. They show that for finitely branching systems[2], two states of the same system are bisimilar if and only if they satisfy the same formulas of that logic.

Before proving that $\mathcal{L}$ characterizes bisimulation, we give two examples to give an idea why negation and finite branching are not needed for the logic to characterize bisimulation between probabilistic processes.

Example 4.1.1 *The two following non-probabilistic systems*

$$\begin{tikzcd}
& \bullet \arrow[dl, "a"'] \arrow[dr, "a"] & & & \bullet \arrow[d, "a"'] \\
\bullet & & \bullet \arrow[d, "b"] & & \bullet \arrow[d, "b"'] \\
& & \bullet & & \bullet
\end{tikzcd}$$

can be distinguished with the formula $\langle a\rangle\neg\langle b\rangle\top$, which says that the process can perform an a-action and then be in a state where it cannot perform a b-action. The process on the left satisfies this formula while the process on the right does not. However, it is well-known that they cannot be distinguished by a negation-free formula of Hennessy-Milner logic. If we now consider probabilistic versions of these processes we find that the situation is different. For no assignment of probabilities are the two processes going to satisfy the same formulas of $\mathcal{L}$. Suppose that the two a-labelled branches of the left hand process are given probabilities p and q, assume that the b-labelled transitions have probability 1. *Now if the right hand process has its a-labelled*

[2]They actually use a stronger property, the "minimum deviation condition" which uniformly bounds the degree of branching everywhere.

transition given a probability anything other than $p+q$, say $r > p+q$ we can immediately distinguish the two processes by the formula $\langle a\rangle_{p+q}\mathsf{T}$ which will not be satisfied by the left hand process. If $r = p+q$ then we can use the formula $\langle a\rangle_q\langle b\rangle_0\mathsf{T}$. If the two processes are not bisimilar, in which case $p > 0$, the left hand process cannot satisfy this formula but the right hand one does.

This simple example shows that one can use the probabilities to finesse the need for negation but one cannot actually encode negation with just $\mathcal{L}$. Of course this example does not constitute a proof but it makes it more plausible that indeed negation is not needed. It is tempting to think that the ability to distinguish processes comes from the power to encode negation and infinitary conjunction by manipulations of the probability subscripts in the modal formulas of the form $\langle a\rangle\phi$. In fact this is not the case. With negation we can write a formula which is only satisfied by NIL states assuming that there are only finitely many distinct actions, namely

$$\bigwedge_{a\in\mathcal{A}} \neg\langle a\rangle\mathsf{T}.$$

It is not possible to write a formula that is only satisfied by NIL states using just $\mathcal{L}$. There is no paradox of course. Given two states one can write a $\mathcal{L}$ formula which distinguishes them but this formula may depend on both states and cannot be constructed just by looking at one of them. For example, suppose that there is a family of states s_n, where n is a positive integer, such that the only transition is an a-labelled transition to a NIL state with probability $\frac{1}{n}$. Now no *single* formula of $\mathcal{L}$ can distinguish all these states from the NIL state but given any s_n the formula $\langle a\rangle_{\frac{1}{n+1}}\mathsf{T}$ will work.

The next example shows why we do not need infinite conjunction even if we have infinite branching.

Example 4.1.2 *Consider the processes P and Q of Figure 4.1 and the formula $\langle a\rangle(\bigwedge_n \langle a\rangle^{(n)}\mathsf{T})$ where the notation $\langle a\rangle^{(n)}$ means n nested $\langle a\rangle$ modalities. The conjunction is over all $n \geq 1$. This formula says that the process can jump to a state from which arbitrarily many a-labelled transitions are possible. The process P does*

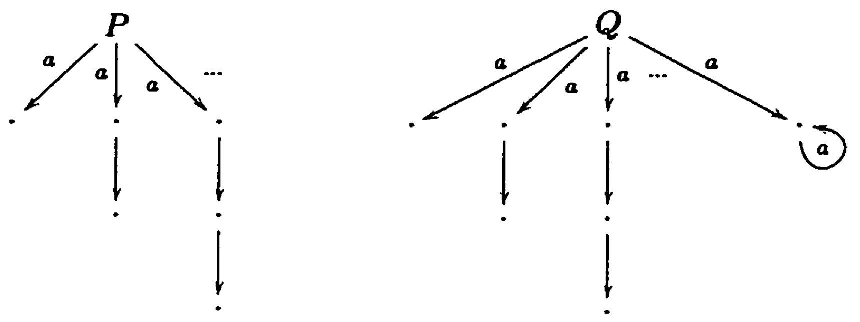


Figure 4.1: Infinite conjunction is necessary to distinguish P and Q.

not satisfy this formula but Q does. Now if we associate probabilities with these transitions we find that we can find distinguishing formulas that do not involve infinite conjunction. To see this assume that both processes satisfy all the same $\mathcal{L}$ formulas. We will show that the probability associated with the extra branch in Q has to be 0, *i.e. it really cannot be present. Now the sum of the initial probabilities have to match since they both satisfy all the same formulas of the form $\langle a\rangle_p\top$. Now in both processes the branch that takes the initial state to a dead state has to have the same probability because they both satisfy all the same formulas of the form $\langle a\rangle_p\langle a\rangle_0\top$. By induction it follows that each branch in P must have the same probability as the corresponding equal length branch in Q. Thus the branch to the looping state in Q must have probability* 0, *because we proved that the sum of the initial probabilities have to match. Consequently, if this probability is not* 0, *in which case the two systems are not bisimilar, they cannot satisfy all the same formulas of $\mathcal{L}$ and hence a distinguishing formula can be constructed which does not involve infinite conjunction.*

4.2 Logical characterization for bisimulation

We prove that bisimulation is characterized by the logic $\mathcal{L}$. The proof relies on various properties of analytic spaces. To show that two bisimilar states satisfy all the same formulas of $\mathcal{L}$ is a relatively easy induction argument. To show the converse, one defines an equivalence relation on states - two states are equivalent if they satisfy the same formulas - and then form the quotient of a process. We need a general theorem to assure us that the result is analytic. If we used Polish spaces, then we would not be assured that the quotient remains Polish. We then define a transition probability on this quotient system in such a way as to ensure that the morphism from the process to its quotient is zigzag. This is the part of the construction where we need most of the measure-theoretic machinery. We use a *unique structure theorem* to show that the measurable sets defined by the formulas of the logic generate the σ-field. Once again, this theorem is only true for analytic spaces and hence is another motivation for imposing an analytic space structure on our processes. We use a theorem on *unique extension of measure* in order to show that the transition probability is well-defined.

The first proposition below says that sets of states definable by formulas in a labelled Markov process are always measurable.

Proposition 4.2.1 *Let (S, i, Σ, τ) be an object of **LMP**. Then for all formulas ϕ, we have $[\![\phi]\!] \in \Sigma$.*

Proof . We proceed by structural induction on ϕ. The base case corresponding to T is trivial since $S \in \Sigma$. Conjunction is trivial because, by definition, a σ-field is closed under intersection. Finally, we have $[\![\langle a\rangle_q\phi]\!] = \tau_a(\cdot, [\![\phi]\!])^{-1}((q, 1]) \in \Sigma$. To justify this first note that, by hypothesis, $[\![\phi]\!] \in \Sigma$ so $\tau_a(s, [\![\phi]\!])$ is meaningful. Secondly, τ_a is a measurable function in its first argument and finally intervals are Borel. ∎

Theorem 4.2.2 *Let $\mathcal{S} = (S, i, \Sigma, \tau)$ and $\mathcal{S}' = (S', i', \Sigma', \tau')$ be labelled Markov processes. If two states $s, s' \in S \cup S'$ are bisimilar then they satisfy the same formulas of $\mathcal{L}$.*

Proof . Let R be a bisimulation between $\mathcal{S}$ and $\mathcal{S}'$. We prove by induction on the structure of formulas that if sRs' then s and s' satisfy the same formulas. The cases of T and conjunction are trivial. Now assume the claim is true for ϕ, i.e., for every pair of R-related states, either both satisfy ϕ or neither of them does. This means that the set $[\![\phi]\!]_{\mathcal{S}} \cup [\![\phi]\!]_{\mathcal{S}'}$ is R-closed. Since R is a bisimulation, $\tau_a(s, [\![\phi]\!]_{\mathcal{S}}) = \tau'_a(s', [\![\phi]\!]_{\mathcal{S}'})$ for all $a \in \mathcal{A}$. So s and s' satisfy the same formulas of the form $\langle a\rangle_q\phi$. ∎

In order to show the logic gives a complete characterization of bisimulation, we also want to show the converse. We write $s \approx s'$ to mean that s and s' satisfy all the same formulas.

We want to show that $\approx$ is a bisimulation between every pair of processes, $\mathcal{S}$ and $\mathcal{S}'$. Thus we want to show that for every pair $s, s' \in S \cup S'$, and every $\approx$-closed measurable set $Y \subseteq S \cup S'$, we have $\tau_a(s, Y \cap S) = \tau'_a(s', Y \cap S')$ for every $a \in \mathcal{A}$. The following lemma is a first step in that direction. It says that the equality is true for sets definable by formulas, i.e., that the transition probabilities to definable sets are completely determined by the formulas, independently of the system.

Lemma 4.2.3 *Let $\mathcal{S} = (S, i, \Sigma, \tau)$ and $\mathcal{S}' = (S', i', \Sigma', \tau')$ be two labelled Markov processes. Then for all formulas ϕ and all pairs (s, s') such that $s \approx s'$, we have $\tau_a(s, [\![\phi]\!]_S) = \tau'_a(s', [\![\phi]\!]_{S'})$.*

Proof . Suppose that the equation does not hold. Then, say, for some ϕ, $\tau_a(s, [\![\phi]\!]_S) < \tau'_a(s', [\![\phi]\!]_{S'})$. We choose a rational number q between these values. Now it follows that $s' \models \langle a \rangle_q \phi$ but $s \not\models \langle a \rangle_q \phi$, which contradicts the assumption that s and s' satisfy all the same formulas. ∎

The plan is then to prove that sets of the form $[\![\phi]\!]_S \cup [\![\phi]\!]_{S'}$ are $\approx$-closed, which is obvious, and that they generate all $\approx$-closed measurable sets. Finally we want to show that the fact that $\tau_a(s, \cdot)$ and $\tau'_a(s', \cdot)$ agree on the sets definable by formulas implies that they agree on every $\approx$-closed set. To do so, we first show that there is a zigzag morphism from any labelled Markov process to its quotient under $\approx$.

If (S, Σ) is a Borel space, the quotient $(S/_{\approx}, \Sigma_{\approx})$ is defined as follows. $S/_{\approx}$ is the set of all equivalence classes. Then the function $f_{\approx} : S \rightarrow S/_{\approx}$ which assigns to each point of S the equivalence class containing it maps onto $S/_{\approx}$, and thus determines a Borel structure on $S/_{\approx}$: by definition a subset E of $S/_{\approx}$ is a Borel set if $f_{\approx}^{-1}(E)$ is a Borel set in S.

The following theorem is a result of joint work [DEP98].

Theorem 4.2.4 *Let (S, i, Σ, τ) be an object of **LMP**. Then $(S/_{\approx}, \Sigma_{\approx})$ is an analytic space and we can define ρ so that the canonical projection $f_{\approx}$ from (S, i, Σ, τ) to $(S/_{\approx}, f_{\approx}(i), \Sigma_{\approx}, \rho)$ is a zigzag morphism.*

In order to prove this proposition we need a few lemmas. The first allows us to work with direct images of $f_{\approx}$. The next two are known results about analytic spaces while the final lemma is a standard uniqueness theorem.

Lemma 4.2.5 *Let $\mathcal{S} = (S, i, \Sigma, \tau)$ be a labelled Markov process.*

(i) *Each equivalence class in $\mathcal{S}$ is a Borel subset.*

(ii) *The equivalence classes in $\mathcal{S}$ refine $[\![\phi]\!]$ for each formula ϕ of the logic.*

(iii) *$f_{\approx}^{-1} f_{\approx}[\![\phi]\!] = [\![\phi]\!]$ for each formula ϕ of the logic.*

Proof . (i): let $t \in S$. Then it is easy to see that the equivalence class containing t is equal to $\bigcap_{t \models \phi} [\![\phi]\!] \setminus \bigcup_{t \not\models \phi} [\![\phi]\!]^c$ which is obviously a Borel subset of $\mathcal{S}$ ($[\![\phi]\!]^c$ denotes the complement of $[\![\phi]\!]$). (ii): Clearly, $[\![\phi]\!] = \bigcup_{t \models \phi} [t]$ where $[t]$ is the equivalence class containing t. (iii): The reversed inclusion is obvious and direct inclusion follows from the fact that if s, t are mapped to the same state, they must satisfy the same formulas, so if $s \in [\![\phi]\!]$ and $t \in f_{\approx}^{-1} f_{\approx}[\![\phi]\!]$, then t must be in $[\![\phi]\!]$ as well. ∎

The next lemmas are Theorem 3.3.5 of [Arv76] and one of its corollaries. We omit the proofs.

Lemma 4.2.6 *Let X be an analytic Borel space and let $\sim$ be an equivalence relation in X. Assume there is a sequence $f_1, f_2, \ldots$ of real valued Borel functions on X such that for any pair of points x, y in X one has $x \sim y$ if and only if $f_n(x) = f_n(y)$ for all n. Then $X/_{\sim}$ is an analytic Borel space.*

Lemma 4.2.7 *Let $(X, \mathcal{B})$ be an analytic Borel space and let $\mathcal{B}_0$ be a countably generated sub-σ-field of $\mathcal{B}$ which separates points in X. Then $\mathcal{B}_0 = \mathcal{B}$.*

The final lemma that we need is a result which gives a condition under which two measures are equal. It is Theorem 10.4 of Billingsley [Bil95] which relies on the famous $\lambda\pi$-theorem of Dynkin.

Lemma 4.2.8 *Let X be a set and $\mathcal{F}$ a family of subsets of X, closed under finite intersections, and such that X is a countable union of sets in $\mathcal{F}$. Let $\sigma(\mathcal{F})$ be the σ-field generated by $\mathcal{F}$. Suppose that μ_1, μ_2 are finite measures on $\sigma(\mathcal{F})$. If they agree on $\mathcal{F}$ then they agree on $\sigma(\mathcal{F})$.*

Proof of Theorem 4.2.4: We first show that $\mathcal{S}/_{\approx}$ is an analytic space. Let $\{\phi_i | i \in \mathbf{N}\}$ be the set of all formulas. We know that $[\![\phi_i]\!]_S$ is a Borel set for each i. Therefore the characteristic functions $\chi_{\phi_i} : S \to \{0,1\}$ are Borel measurable functions. Moreover we have

$$x \approx y \text{ iff } (\forall i \in \mathbf{N}.\, x \in [\![\phi_i]\!]_S \iff y \in [\![\phi_i]\!]_S) \text{ iff } (\forall i \in \mathbf{N}.\, \chi_{\phi_i}(x) = \chi_{\phi_i}(y)).$$

It now follows by Lemma 4.2.6 that $\mathcal{S}/_{\approx}$ is an analytic space.

Let $\mathcal{B} = \{f_{\approx}([\![\phi_i]\!]_S) : i \in \mathbf{N}\}$. We show that $\sigma(\mathcal{B}) = \Sigma_{\approx}$. We have $\mathcal{B} \subseteq \Sigma_{\approx}$, since, by Lemma 4.2.5 (iii), for any $f_{\approx}([\![\phi_i]\!]_S) \in \mathcal{B}$, $f_{\approx}^{-1} f_{\approx}([\![\phi_i]\!]_S) = [\![\phi_i]\!]_S$ which is in Σ by Proposition 4.2.1. Now $\sigma(\mathcal{B})$ separates points in $\mathcal{S}/_{\approx}$, for if x and y are different states of $\mathcal{S}/_{\approx}$, take states $s \in f_{\approx}^{-1}(x)$ and $t \in f_{\approx}^{-1}(y)$. Then since $s \not\approx t$, there is a formula ϕ such that s is in $[\![\phi]\!]_S$ and t is not. By Lemma 4.2.5 (iii), it follows that x is in $f_{\approx}[\![\phi]\!]_S$, whereas y is not. Since $\sigma(\mathcal{B})$ is countably generated, it follows by Lemma 4.2.7, that $\sigma(\mathcal{B}) = \Sigma_{\approx}$.

We are now ready to define $\rho_a(x, \cdot)$ over $\Sigma_{\approx}$ for $x \in S/_{\approx}$. We would like to define it so that $f_{\approx} : \mathcal{S} \to \mathcal{S}/_{\approx}$ turns out to be a zigzag morphism (recall that $f_{\approx}$ is measurable). Hence, for any $B \in \Sigma_{\approx}$ we put

$$\rho_a(x, B) = \tau_a(s, f_{\approx}^{-1}(B)),$$

where $s \in f_{\approx}^{-1}(x)$. Clearly, for a fixed state s, $\tau_a(s, f_{\approx}^{-1}(\cdot))$ is a sub-probability measure on $\Sigma_{\approx}$. We now show that the definition does not depend on the choice of s in $f_{\approx}^{-1}(x)$ for if $s, s' \in f_{\approx}^{-1}(x)$, we know that $\tau_a(s, f_{\approx}^{-1}(\cdot))$ and $\tau_a(s', f_{\approx}^{-1}(\cdot))$ agree over $\mathcal{B}$ again by the fact that $f_{\approx}^{-1}f_{\approx}([\![\phi_i]\!]_S) = [\![\phi_i]\!]_S$ and by Lemma 4.2.3. So, since $\mathcal{B}$ is closed under the formation of finite intersections we have, from Lemma 4.2.8, that $\tau_a(s, f_{\approx}^{-1}(\cdot))$ and $\tau_a(s', f_{\approx}^{-1}(\cdot))$ agree on $\sigma(\mathcal{B}) = \Sigma_{\approx}$.

It remains to prove that for a fixed Borel set B of $\Sigma_{\approx}$, $\rho_a(\cdot, B) : S/_{\approx} \to [0,1]$ is a Borel measurable function. Let A be a Borel set of $[0,1]$. It is easy to check that $\rho_a(\cdot, B)^{-1}(A) = q[\tau_a(\cdot, f_{\approx}^{-1}(B))^{-1}(A)]$, from the zigzag property applied to $f_{\approx}$. We know that $C = \tau_a(\cdot, f_{\approx}^{-1}(B))^{-1}(A)$ is Borel since it is the inverse image of A under a Borel measurable function. Now we claim that $f_{\approx}(C) \in \Sigma_{\approx}$, since $f_{\approx}^{-1}f_{\approx}(C) = C$. To see this, note that if $s_1 \in f_{\approx}^{-1}f_{\approx}(C)$, there exists $s_2 \in C$ such that $f_{\approx}(s_1) = f_{\approx}(s_2)$. We have just proved above that then the $\tau_a(s_i, f_{\approx}^{-1}(\cdot))$'s must agree, so if $\tau_a(s_i, f_{\approx}^{-1}(B)) \in A$ for $i = 2$, then it is also true for $i = 1$, so $s_1 \in C$ as wanted. Thus $\rho_a(\cdot, B)$ is Borel measurable. This concludes the proof that $S/_{\approx}$ is a **LMP** and $f_{\approx}$ a zigzag morphism. ∎

We now state the main result on logical characterization of bisimulation.

Theorem 4.2.9 *Let $\mathcal{S}$ and $\mathcal{S}'$ be labelled Markov processes. Two states $s, s' \in S \cup S'$ are bisimilar if and only if they satisfy the same formulas of $\mathcal{L}$.*

Proof . The left to right direction is given by Theorem 4.2.2. We prove the other direction. Consider $\mathcal{U} = (U, u_0, \Omega, \tau)$, the direct sum of $\mathcal{S}$ and $\mathcal{S}'$. Note that every state of either $\mathcal{S}$ or $\mathcal{S}'$ satisfies the same formulas in $\mathcal{U}$ as in its original process. By using the quotient $\mathcal{U}/_{\approx} = (U/_{\approx}, f_{\approx}(u_0), \Omega_{\approx}, \rho)$, we show that the relation $\approx$ defined on the states of $\mathcal{U}$ is a bisimulation relation. Let $A \in \Omega$ be $\approx$-closed (then $A \cap S \in \Sigma$ and $A \cap S' \in \Sigma'$). Then we have $A = f_{\approx}^{-1}f_{\approx}(A)$ and hence $f_{\approx}(A) \in \Omega_{\approx}$. Now if $s \approx s'$ in $\mathcal{U}$, then $f_{\approx}(s) = f_{\approx}(s')$, and since by Theorem 4.2.4 $f_{\approx}$ is a zigzag morphism, we have $\tau_a(s, A) = \rho_a(f_{\approx}(s), f_{\approx}(A)) = \tau_a(s', A)$, as wanted. ∎

We can now prove an important result that we mentioned in the section where bisimulation is defined and delayed until now.

Corollary 4.2.10 *Bisimulation is an equivalence relation.*

Proof . Suppose that $\mathcal{S}$ and $\mathcal{S}'$ are bisimilar and that $\mathcal{S}'$ and $\mathcal{S}''$ are bisimilar. This means that we have bisimulation relations R between $\mathcal{S}$ and $\mathcal{S}'$ and R between $\mathcal{S}'$ and $\mathcal{S}''$, satisfying $iRi'R'i''$. Then, by the left to right direction of Theorem 4.2.9, i and i'' satisfy the same formulas. Thus, by the other direction of the same theorem, i and i'' are bisimilar, and hence there is a bisimulation relation between $\mathcal{S}$ and $\mathcal{S}''$ relating them. ∎

Example 4.2.11 *We come back to process $\mathcal{S}$ of Example 3.7.1. To show that every pair of states are not bisimilar, we only have to find a formula that distinguishes them. We can see that every positive state has a different probability for jumping to a negative state, and hence to a positive state, since $\tau(s, \mathbf{R}) = 1$ for all $s > 0$. Now, every $s > 0$ has a different number $\tau(s, [\![\langle a\rangle_0 \mathsf{T}]\!])$, since $[\![\langle a\rangle_0 \mathsf{T}]\!]$ is exactly the set of positive states. Hence for every pair of positive states, there is a rational q such that the two states are distinguished by the formula $\langle a\rangle_q \langle a\rangle_0 \mathsf{T}$.*

4.3 Logical characterization of simulation

We just proved that bisimulation is characterized by a simple logic which does not involve negation. One advantage of this feature is that this opens the way to a notion of logical simulation between processes. Recall that a state *logically simulates* another state if it satisfies (at least) all the formulas the other satisfies. A logic containing negation cannot characterize simulation. We prove that $\mathcal{L}_\vee$ characterizes simulation between discrete labelled Markov processes and arbitrary labelled Markov processes.

The notion of simulation accords well with the logic $\mathcal{L}_\vee$ in the sense of the following proposition.

Proposition 4.3.1 *If s is simulated by s', then for all formulas $\phi \in \mathcal{L}_\vee$, $s \models \phi$ implies $s' \models \phi$.*

Proof . Let R be a simulation on a single process $\mathcal{S} = (S, i, \Sigma, \tau)$. We prove by induction on the structure of formulas that for every formula ϕ, $[\![\phi]\!]$ is R-closed, which implies the result. It is obvious for T and conjunction. Now assume it is true for ϕ, and let sRs'. Then, since R is a simulation and $[\![\phi]\!]$ is measurable and R-closed, we have $\tau_a(s, [\![\phi]\!]) \leq \tau_a(s', [\![\phi]\!])$, and hence $[\![\langle a \rangle_q \phi]\!]$ is R-closed for every rational q.

Now if s and s' come from two different processes, observe that if $\mathcal{S}$ is a direct summand of $\mathcal{U}$, a state of $\mathcal{S}$ satisfies exactly the same formulas in $\mathcal{S}$ as in $\mathcal{U}$. Hence the result. ∎

In order to obtain that the logic $\mathcal{L}_\vee$ characterizes simulation, we must prove the converse. The following theorem shows that the logic does characterize simulation in a special case where the simulated state comes from a discrete process.

Theorem 4.3.2 *A state in a discrete process is simulated by a state in an arbitrary process if and only if it is logically simulated by that state with respect to the logic $\mathcal{L}_\vee$.*

Proof. The "only if" direction is given by Proposition 4.3.1. For the "if" part, consider the reflexive relation W induced by the logic on the direct sum of the discrete process $\mathcal{P} = (P, p_0, \pi)$ and an arbitrary process $\mathcal{S}$, defined as follows. Let $p \in P$ be W-related to $s \in S$ if s satisfies all the formulas that p satisfies. We show that W is a simulation relation on $\mathcal{P} + \mathcal{S}$. Let pWs and Y be a W-closed set in the direct sum. We want to prove that $\pi_a(p, Y \cap P) \leq \tau_a(s, Y \cap S)$. We prove that for every set $B \subseteq P$ (not necessarily W-closed), we have $\pi_a(p, B) \leq \tau_a(s, W(B) \cap S)$ (we can write $W(B) \cap S$ because we will see that it is measurable, since B is countable). This will give us the result since $W(Y \cap P) \cap S \subseteq Y \cap S$ because Y is W-closed. Note that if $p' \in B$, then $W(p') \cap P = \cap_{p \models \phi} [\![\phi]\!]_{\mathcal{P}}$. Taking the union over all p' in B, we get

$$W(B) \cap P = \bigcup_{p' \in B} \Big(\bigcap_{p' \models \phi} [\![\phi]\!]_{\mathcal{P}} \Big).$$

Now by definition of W we have

$$W(B) \cap S = \bigcup_{p' \in B} \Big(\bigcap_{p' \models \phi} [\![\phi]\!]_{\mathcal{S}} \Big).$$

Note that this shows that $W(B) \cap S$ is indeed measurable in Σ.

We now prove that $W(B) \cap P$ and $W(B) \cap S$ are limits of decreasing chains of formulas. First assume that B is finite. Let B_k (resp. B'_k) be the set of states in $\mathcal{P}$

(resp. in $\mathcal{S}$) which satisfy the formula $\vee_{b\in B}(\wedge_{b\models\phi\in F_k}\phi)$ where F_k is the (finite) set of formulas of depth $\leq k$ that involves probabilities which are integer multiples of $1/j$ for some $1 \leq j \leq k$ and only the first k actions of $\mathcal{A}$. Then $(B_k)_{k\in\mathbf{N}}$ and $(B'_k)_{k\in\mathbf{N}}$ are decreasing chains. We prove that $\cap B'_k = W(B) \cap S$. If $s \in W(B) \cap S$, then there is a $b \in B$ which is simulated by s, hence $s \models \wedge_{b\models\phi\in F_k}\phi$ for all k. Conversely, if $s \notin W(B) \cap S$, then for all $b \in B$ there is a formula ϕ_b such that $b \models \phi_b$ but $s \not\models \phi_b$. Let k be such that all ϕ_b are in F_k: this is possible because B is finite and all formulas ϕ_b are finite and hence involve a finite number of probabilities. Then $s \notin B'_k$ because $s \not\models \vee_{b\in B}\phi_b$, and hence $s \notin \cap B'_k$. Thus $\cap B'_k = W(B) \cap S$ and similarly $\cap B_k = W(B) \cap P$.

Now since pWs, s satisfies all the formulas of the form $\langle a\rangle_q\phi$ that p satisfies. B_k and B'_k being of the form $[\![\psi]\!]$, we have $\pi_a(p, B_k) \leq \tau_a(s, B'_k)$. This implies that $\pi_a(p, B) \leq \pi_a(p, W(B)\cap P) = \pi_a(p, \cap B_k) \leq \tau_a(s, \cap B'_k) = \tau_a(s, W(B)\cap S)$, and hence we have the result for B finite.

If B is countable, then let $(B_l)_{l\in\mathbf{N}}$ be an increasing chain of finite sets whose union is B. Since every B_l is finite, we have $\pi_a(p, B_l)) \leq \tau_a(s, W(B_l) \cap S)$. Now since the B_l's form an increasing chain that converges to B, and similarly the $W(B_l) \cap S$'s converge to $W(B) \cap S$, we have

$$\pi_a(p, B) = \pi_a(p, \cup B_l) \leq \tau_a(s, \cup W(B_l) \cap S) = \tau_a(s, W(B) \cap S)$$

as wanted. The result is valid in particular for $B = Y \cap S$ and the theorem is proved. ∎

Remark 4.3.3 An extension of this theorem to the uncountable case is not straightforward. The argument of the proof relies on the countability of $Y \cap P$ (generalized to B in the proof). However, we know that if $\mathcal{S}$ and $\mathcal{S}'$ are maximally collapsed (that is, they contain no pair of distinct bisimilar states), then their σ-field is generated by the sets $[\![\phi]\!]$, where ϕ is a formula of $\mathcal{L}_\vee$. In the proof of the logical characterization of bisimulation, we use a theorem saying that if two measures agree on a set of sets that generates the σ-field, then they agree on the whole σ-field. If we want to mimic this proof, the first step is to note that for every formula ϕ, we have $\tau_a(s, [\![\phi]\!] \cap S) \leq \tau'_a(s', [\![\phi]\!] \cap S')$ (where $[\![\phi]\!]$ is taken in the direct sum $\mathcal{S} + \mathcal{S}'$). As far as we know, there is no theorem in the literature that could help us saying that this would imply that $\tau_a(s, \cdot) \leq \tau'_a(s', \cdot)$ on every R-closed set.

We previously made the comment that although $\mathcal{L}$ is enough to characterize bisimulation, characterization of simulation needs disjunction. We now give an example of two simple finite processes, one satisfying all the formulas of $\mathcal{L}$ that the other satisfies but which does not simulate it.

Example 4.3.4 *In the following picture, t satisfies all formulas of $\mathcal{L}$ that s satisfies but t does not simulate s.*

$$\begin{tikzcd}
& s \arrow[dl, "{a[\frac{1}{2}]}"'] \arrow[dr, "{a[\frac{1}{2}]}"] & \\
s_1 \arrow[d, "a"'] & & s_2 \arrow[d, "b"] \\
\bullet & & \bullet
\end{tikzcd}
\qquad
\begin{tikzcd}
& & t \arrow[dll, "{a[\frac{1}{4}]}"'] \arrow[dl, "{a[\frac{1}{4}]}"] \arrow[dr, "{a[\frac{1}{4}]}"'] \arrow[drr, "{a[\frac{1}{4}]}"] & & \\
\bullet \arrow[d, "a"'] & \bullet \arrow[d, "a"'] \arrow[dr, "b"] & & \bullet \arrow[d, "b"'] & t_1 \\
\bullet & \bullet & \bullet & \bullet &
\end{tikzcd}$$

Of course there is a formula of $\mathcal{L}$ that distinguishes s and t, namely the formula $\langle a\rangle_0(\langle a\rangle_0\top \wedge \langle b\rangle_0\top)$. This formula is satisfied by t but not by s. To see that t satisfies all formulas of $\mathcal{L}$ that s satisfies, note that the only relevant formulas of $\mathcal{L}$ that are satisfied by s are: $\langle a\rangle_r\top$, for $0 < r < 1$, $\langle a\rangle_r\langle a\rangle_0\top$ and $\langle a\rangle_r\langle b\rangle_0\top$, for $0 < r < 1/2$. All these formulas are also satisfied by t. To see that t does not simulate s, suppose that there is a simulation relation R that relates s and t. Then the set of all states is R-closed and hence s_1 and s_2 cannot be related to any NIL state (such as t_1) because a NIL state cannot perform any of a and b so we cannot have $\pi_a(s_1, S) \leq \pi'_a(t_1, T)$, similarly for label b and state s_2. Hence the set A of non-NIL states is R-closed but s has probability 1 of jumping to A whereas t has probability 3/4 of making an a-transition to A. This shows that disjunction is indeed necessary for characterizing simulation, because we can find a formula from the logic $\mathcal{L}_\vee$, namely $\langle a\rangle_{3/4}(\langle a\rangle_0\top \vee \langle b\rangle_0\top)$ that is satisfied by s but not by t.

The next result will allow us to use a simpler definition of simulation when we work with discrete processes.

Corollary 4.3.5 *If a process simulates a discrete process, then it simulates it through their direct sum.*

Proof. Assume there is a simulation R between $\mathcal{P}$ and $\mathcal{S}$. Consider the relation W induced by the logic on the direct sum of $\mathcal{P}$ and $\mathcal{S}$ defined as above: $p \in P$ is related to $s \in S$ if s satisfies all the formulas that p satisfies. Then W contains R by Proposition 4.3.1. So W is a simulation (by Theorem 4.3.2) on $\mathcal{P}+\mathcal{S}$ relating every state R relates. ∎

We give a simpler definition of simulation that can be used when a discrete process is involved. This definition has the advantage of not using direct sums. It is easy to check that – in the mentioned particular case – the following definition of simulation is equivalent to the one we have given previously.

Corollary 4.3.6 *A simulation between a discrete process $\mathcal{P} = (P, p_0, \rho)$ and another process $\mathcal{S}$ is a reflexive and transitive relation on $P \cup S$ such that the restrictions of R to $\mathcal{P}$ and $\mathcal{S}$ are simulations and pRs implies that for every R-closed set $A \subseteq P \cup S$ such that $A \cap S \in \Sigma$, we have $\rho_a(p, A \cap P) \leq \tau_a(s, A \cap S)$.*

It is easy to see that if there is such a simulation between two continuous processes, there is a simulation according to definition 3.6.2. The reason why we did not use the last definition directly for arbitrary labelled Markov processes is that we could not prove that it yields a transitive relation; this remains an open problem. For discrete processes, transitivity of simulation (as just defined) is given by the logical characterization.

4.4 Algorithms for bisimulation and simulation

The logical characterizations of bisimulation and simulation given in the last section allow us to use the logics instead of the formal definition of bisimulation and simulation. In particular, if we want to check that two states are bisimilar we can prove that they satisfy the same formulas of $\mathcal{L}$. More interestingly, if we can find a formula that is satisfied only by one of these states, we know that they are not bisimilar. This is easier than proving that there is no bisimulation relation relating them. Moreover, the witnessing formula gives information why the two states are not bisimilar. The same remark applies for simulation. In particular, if we are checking whether an implementation matches its specification and find a formula that is satisfied only by the specification, the structure of the formula gives us a hint on a possible "computation" that makes the implementation fail to be adequate for the specification. This way, it can give us a hint in order to modify our incorrect implementation.

For that purpose, we describe two algorithms for bisimulation and simulation. The algorithm for bisimulation produces a witnessing formula from the logic $\mathcal{L}$ in case the systems are not bisimilar. A small modification to it can be used to check if a state is simulated by another. If it is not, the algorithm exhibits a formula of the logic $\mathcal{L}_\vee$ that is satisfied by the state and not by the other. The algorithms were inspired by an algorithm due to Cleaveland [Cle90] to decide bisimilarity of non-probabilistic processes.

We first describe the algorithm for bisimulation. It operates in two steps. The first step is to compute, given a finite labelled Markov chain, a family D of subsets of states having the following properties. Every set of D is exactly the set of states that satisfy some formula of $\mathcal{L}$, and conversely, every formula of $\mathcal{L}$ corresponds to a set of D. At first sight, this last property may appear strange since there are infinitely many formulas in the logic, but since there are only finitely many states in the process, there are finitely many subsets of states. In order to decide whether two states are bisimilar, we then check if they belong to exactly the same sets of D.

The first step is done with bisim, which has a running time of $O(2^n)$, where n is the number of states. It is illustrated in Figure 4.2. Beginning with D containing only the set S of all states, the algorithm constructs for each B in D and $a \in \mathcal{A}$, nested subsets of S having probability greater than some number of jumping to B with action a. We will prove that for every formula of $\mathcal{L}$, the set of states satisfying this formula is a member of D and conversely, every member of D corresponds to a formula. Consequently, all states satisfying the same formulas will belong to exactly

```
bisim($S, \mathcal{A}, \tau, D$)                          Input: $S, \mathcal{A}, \tau$
$D := \{S\}$                                             Result: $D \subseteq \mathcal{P}(S)$
$F(S) := \mathsf{T}$
for each $B \in D$ and $a \in \mathcal{A}$ do              $F : D \to \mathcal{L}_\vee$
   $\Gamma := \{\tau_a(s,B) : s \in S\}$
   for each $q \in \Gamma$ do                              $\Gamma$: set of numbers
      $C := \{s \in S : \tau_a(s,B) > q\}$
      for each $A \in D$, do                              $C \subseteq S$
      $F(C \cap A) := F(A) \wedge \langle a \rangle_q F(B)$
      $D := D \cup \{C \cap A\}$
```

Figure 4.2: An algorithm for deciding bisimulation

the same sets in D. This shows that the algorithm really relies on the characterization of bisimulation by the logic.

More precisely, for each set B of D and $a \in \mathcal{A}$, bisim collects in Γ all possible values of $\tau_a(s, B)$ for $s \in S$. Then, for every possible value in Γ, the subset C of states that can jump into B with probability greater than this value are added to the set D in a precise way. In fact, in order that D be closed under intersections, we add to D all sets $C \cap A$ such that $A \in D$; the algorithm then assigns a formula to the set and adds the set to D. The greatest value in Γ could be removed since it will always lead to an empty set. In an implementation of the algorithm, if a set has been already assigned a formula, no new assignment should be made, for it would assign to the set a longer formula than the one already computed. Of course, in that case, the set needs not be added to D and the two last lines can just be skipped.

For deciding simulation between states of finite processes, the following modification gives us a correct algorithm that we call sim. We must replace the last for-loop of bisim by

```
for each $A_1 \in D$, $A_2 \in D \cup \emptyset$ do
      $F((C \cap A_1) \cup A_2) := (F(A_1) \wedge \langle a \rangle_q F(B)) \vee F(A_2)$
      $D := D \cup \{(C \cap A_1) \cup A_2\}$
```

These lines correspond to the fact that we need disjunction in the logic to characterize simulation. So when we add a new set C, we must make sure that its intersection and union with every set in D is in D as well as the union and intersection of every pair of sets in D. Instead of adding to D sets of the form $C \cap A$, we add to D all sets $(C \cap A_1) \cup A_2$ for $A_1, A_2 \in D$; we will prove in Proposition 4.4.2 that this is enough to make sure that D is closed under intersections and unions. At the end of running sim, a simulating state will belong to every set containing a state it simulates.

The second step of the algorithm is to decide whether or not a state is bisimilar to another state and exhibit a formula if not; checkbisim does that when we give to it as input the set D computed by bisim and two states, s and t. It simply goes through every set in D and checks if s and t are "distinguished" by that set.

```
checkbisim($s, t, D$)
for each $B \in D$ do
   if ($s \in B$ and $t \notin B$) or ($s \notin B$ and $t \in B$) then
      return $s$ "and" $t$ "are distinguished by the formula" $F(B)$; exit
return $s$ "and" $t$ "are bisimilar"
```

We could look for the "first" set that distinguishes them, hoping to obtain a shorter formula. The algorithm bisim itself does not record the order of creation on the sets B, but it could be easily modified to do so. The formula obtained either way is not guaranteed to be minimal and often it will not be minimal.

The checking algorithm for simulation is very similar. It goes through every set in D and checks if t is in every set that s belongs to.

```
checksim($s, t, D$)
for each $B \in D$ do
   if $s \in B$ and $t \notin B$ then
      return $s$ "satisfies formula" $F(B)$ "but" $t$ "does not."; exit
return $s$ "is simulated by" $t$
```

The following proposition shows that checkbisim really decides if two states are bisimilar.

Proposition 4.4.1 *Two states of a process $\mathcal{S}$ satisfy the same formulas if and only if they belong to exactly the same sets in D at the end of executing the algorithm* bisim. *If the set B distinguishes them, then formula $F(B)$ is satisfied by one state but not by the other.*

Proof . First note that the algorithm must terminate since $2^{|S|}$ is finite.

We prove necessity by showing that in bisim, every element of D corresponds to a formula, i.e., for every $B \in D$, there exists a formula ϕ such that $B = [\![\phi]\!]$. We will prove by induction on the number of iteration of the first for-loop that for every $B \in D$, $F(B)$ is such a formula. The whole set S corresponds to the formula $\top$. Suppose that after n iterations of the first for-loop, every element of D corresponds to a formula. Then we must show that all sets added to D in the last for-loop also correspond to formulas. So we prove that for each $B \in D$, each $q \in \Gamma$ and each $A \in D$, $C \cap A = [\![F(C \cap A)]\!]$, where C is defined from q and B. Since $A = [\![F(A)]\!]$ and $B = [\![F(B)]\!]$ by induction hypothesis and $C = [\![\langle a \rangle_q F(B)]\!]$ by definition, then $C \cap A = [\![F(A) \wedge \langle a \rangle_q F(B)]\!] = [\![F(C \cap A)]\!]$. So each set in D at the end of executing this algorithm corresponds to the set of states that satisfy some formula. This implies that if two states satisfy the same formulas, they must be in the same sets of D.

For sufficiency, we want to show that if two states s, s' do not satisfy the same formulas they are not in the same sets of D. To do so, we will show by structural induction on formulas that every formula corresponds to a set in D when the algorithm is finished, i.e., for every formula ϕ, $[\![\phi]\!] \in D$. So assume the algorithm is finished and hence that D is constructed. $[\![\top]\!] = S \in D$. Now assume $[\![\phi]\!]$ and $[\![\psi]\!]$ are in D. To prove that $[\![\phi \wedge \psi]\!] \in D$, we will prove by induction on the number of iteration of the first for-loop that every intersection of two sets of D is in D. So let D_1 and D_2 be two consecutive status of D in the history of the algorithm, and assume the claim is true for D_1. We want to prove that D_2 is closed under intersection. Obviously, if we take two sets in D_2 that were already in D_1, their intersection is in D_2. So we only have to check that D_2 is closed under intersection of new sets and under intersections of new and old sets. We first want to prove that $X \cap (A \cap C) \in D_2$, where $A, X \in D_1$ and C defined from $B \in D_1$ and $q \in \Gamma$, as above. But this set is equal to $(X \cap A) \cap C$ which is also in D_2 because $X \cap A \in D_1$ by induction hypothesis. Now we also have $(A \cap C) \cap (X \cap C) \in D_2$ because this set is again equal to $(X \cap A) \cap C$. This proves that D_2 is closed under intersection, and hence $[\![\phi \wedge \psi]\!] \in D$.

Now we want to prove that if $[\![\phi]\!] \in D$, then $[\![\langle a \rangle_q \phi]\!] \in D$. Then let $r =$

$\max_{s \in S}\{\tau_a(s, [\![\phi]\!]) \leq q\}$. So since $B = [\![\phi]\!]$ must have been considered in the algorithm, and then at that time $r \in \Gamma$, we have, for $C = [\![\langle a \rangle_r \phi]\!]$, $C \cap S = [\![\langle a \rangle_r \phi]\!] \in D$. But this set is exactly $[\![\langle a \rangle_q \phi]\!] \in D$ ∎

The following proposition shows that checksim really decides if a state simulates another one. It shows that given two states, if the first state does not belong to every set of D the other is in, witnessed by say, set B in D, we get a formula $F(B)$ satisfied by the second state but not by the first one.

Proposition 4.4.2 *A state s logically simulates another state s' in process $\mathcal{S}$ if and only if at the end of running the algorithm sim on $\mathcal{S}$, s belongs to every set of D that s' is in.*

Proof . Note again that the algorithm must terminate since $2^{|S|}$ is finite.

We prove necessity by showing that in sim, every element B of D is equal to $[\![F(B)]\!]$. This will be done by induction on the number of iteration of the first for-loop in sim. Obviously, $S = [\![\top]\!]$. Suppose that after n iterations of the first for-loop, the claim is true for every element of D. Then we must show that it is also true for all sets added to D in the last for-loop. So we prove that for each $B \in D$, each $q \in \Gamma$ and each $A_1 \in D$, $A_2 \in D \cup \emptyset$, $(C \cap A_1) \cup A_2) = [\![F((C \cap A_1) \cup A_2)]\!]$, where C is defined from q and B. Since $A_i = [\![F(A_i)]\!]$ $(i = 1, 2)$ and $B = [\![F(B)]\!]$ by induction hypothesis and $C = [\![\langle a \rangle_q F(B)]\!]$, then $C \cap A = [\![F(A) \wedge \langle a \rangle_q F(B)]\!] = [\![F((C \cap A_1) \cup A_2)]\!]$. So for each $B \in D$ at the end of executing sim we have $B = [\![F(B)]\!]$. This implies that if $s \in S$ satisfy all the formulas that $s' \in S$ satisfies and if $s' \in B \in D$ at the end of the execution of the algorithm, we have $s' \models F(B)$ and hence s also satisfies $F(B)$ and hence is in B.

For sufficiency, we want to show that if two states s, s' do not satisfy the same formulas they are not in the same sets of D. To do so, we will show by structural induction on formulas that every formula corresponds to a set in D when the algorithm is finished, i.e., for every formula ϕ, $[\![\phi]\!] \in D$. So assume the algorithm is finished and hence that D is constructed. The formula $\top = S \in D$. Now assume $[\![\phi]\!]$ and $[\![\psi]\!]$ are in D. To prove that $[\![\phi \wedge \psi]\!]$ and $[\![\phi \vee \psi]\!]$ are in D, we will prove by induction on the number of iterations of the first for-loop that D is closed under unions and intersections. So let D_1 and D_2 be two consecutive status of D in the history of the algorithm, and assume that D_1 is closed under unions and intersections. Obviously, if we take two sets in D_2 that were already in D_1, their intersection and union are in D_2. So we only have to check that D_2 is closed under intersection and unions of new sets and of new and old sets. Let $A, B, A_1, B_1 \in D_1$, $A_2, B_2 \in D \cup \emptyset$. Let C be the set constructed from $q \in \Gamma$ and B, which introduces new sets in D_2. All possible cases are considered in the following list.

1. $A \cup (C \cap A_1) \cup A_2 = (C \cap A_1) \cup (A \cup A_2)$ is of the form $(C \cap A_1) \cup B^* \in D_2$;
2. $A \cap ((C \cap A_1) \cup A_2) = (C \cap (A \cap A_1)) \cup (A \cap A_2)$ is of the form $(C \cap A^*) \cup B^* \in D_2$;
3. $((C \cap A_1) \cup A_2) \cup ((C \cap B_1) \cup B_2) = (C \cap (A_1 \cup B_1)) \cup (A_2 \cup B_2)$ is of the form $(C \cap A^*) \cup B^* \in D_2$;
4. $((C \cap A_1) \cup A_2) \cap ((C \cap B_1) \cup B_2) = (C \cap A_1 \cap B_1) \cup (C \cap A_1 \cap B_2) \cup (C \cap A_2 \cap B_1) \cup (A_2 \cap B_2)$ is of the form $(C \cap A^*) \cup B^* \in D_2$;

In each case, we have A^*, B^* are in D_1 by induction hypothesis. This prove that D_2 is closed under intersections and unions, and hence $[\![\phi \wedge \psi]\!]$ and $[\![\phi \vee \psi]\!]$ are in D.

Now we want to prove that if $[\![\phi]\!] \in D$, then $[\![\langle a \rangle_q \phi]\!] \in D$. Then let $r = \max_{s \in S}\{\tau_a(s, [\![\phi]\!]) \leq q\}$. So since $B = [\![\phi]\!]$ must have been considered in the algorithm, and then at that time $r \in \Gamma$, we have, for $C = [\![\langle a \rangle_r \phi]\!]$, $(C \cap S) \cup \emptyset = [\![\langle a \rangle_r \phi]\!] \in D$. But this set is exactly $[\![\langle a \rangle_q \phi]\!] \in D$.

If a state $s \in S$ belongs to every set of D that $s' \in S$ is in, and if $s' \models \phi$, then there is some $B \in D$ such that $B = [\![\phi]\!]$. Then $s' \in B$ and hence $s \in B$ and this implies that $s \models \phi$. ∎

Example 4.4.3 *We work out a simple example to illustrate how the algorithm operates. Consider the finite labelled Markov process of Figure 4.3. If we run the algorithm sim on this process, the following sequence of steps will be obtained.*

- *Input is* $S = \{s_0, \ldots, s_9\}$, $\mathcal{A} = \{a, b\}$;

$$\begin{tikzcd}
 & s_0 \arrow[dl, "{a[\frac{1}{6}]}"'] \arrow[d, "a", "{[\frac{2}{3}]}"'] \arrow[dr, "{a[\frac{1}{6}]}"] & & \\
s_1 & s_2 \arrow[dl, "{a[\frac{3}{4}]}"'] \arrow[d, "b"] & s_3 \arrow[d, "{a[\frac{1}{3}]}"'] \arrow[dr, "{a[\frac{2}{3}]}"] & \\
s_4 \arrow[d, "b"] & s_5 & s_6 & s_7 \arrow[d, "b"] \\
s_8 & & & s_9
\end{tikzcd}$$

Figure 4.3: A finite process $\mathcal{S}$.

- *$D = \{S\}$, $F(S) = \top$;*

- *$B = S$, label is a: we get $\Gamma = \{0, 3/4, 1\}$ and $C_0 = \{s_0, s_2, s_3\} \notin D$, so $D = \{S, \{s_0, s_2, s_3\}\}$, $F(\{s_0, s_2, s_3\}) = \langle a \rangle_0 \top$. Now $C_{3/4} = \{s_0, s_3\} \notin D$ so $D = \{S, \{s_0, s_2, s_3\}, \{s_0, s_3\}\}$, $F(\{s_0, s_3\}) = \langle a \rangle_{3/4} \top$.*

- *$B = S$, label is b: we get $\Gamma = \{0, 1\}$ and $C_0 = \{s_2, s_4, s_7\} \notin D$, so $D = \{S, \{s_0, s_2, s_3\}, \{s_0, s_3\}, \{s_2, s_4, s_7\}, \{s_2\}, \{s_0, s_2, s_3, s_4, s_7\}\}$, $F(\{s_2, s_4, s_7\}) = \langle b \rangle_0 \top$, $F(\{s_2\}) = \langle a \rangle_0 \top \wedge \langle b \rangle_0 \top$, $F(\{s_0, s_2, s_3, s_4, s_7\}) = \langle a \rangle_0 \top \vee \langle b \rangle_0 \top$*

- *$B = \{s_0, s_2, s_3\}$, label is a: we get $\Gamma = \{0, 5/6\}$ and $C_0 = \{s_0\} \notin D$, so*
  $D = \{S, \{s_0, s_2, s_3\}, \{s_0, s_3\}, \{s_2, s_4, s_7\}, \{s_2\},$
  $\quad \{s_0, s_2, s_3, s_4, s_7\}, \{s_0\}, \{s_0, s_2\}, \{s_0, s_2, s_4, s_7\}\}$,
  $F(\{s_0\}) = \langle a \rangle_0 \langle a \rangle_0 \top$, $F(\{s_0, s_2\}) = \langle a \rangle_0 \langle a \rangle_0 \top \vee \langle a \rangle_0 \top \wedge \langle b \rangle_0 \top$,
  $F(\{s_0, s_2, s_4, s_7\}) = \langle a \rangle_0 \langle a \rangle_0 \top \vee \langle b \rangle_0 \top$;

- *$B = \{s_0, s_2, s_3\}$, label is b: then $\Gamma = \{0\}$ and hence D is not modified;*

- *$B = \{s_2, s_4, s_7\}$, label is a: we get $\Gamma = \{0, 2/3, 1\}$ and $C_0 = \{s_0, s_2, s_3\} \in D$, and $C_{2/3} = \{s_2\} \in D$ so D does not change; label b and the same B does not modify D;*

- *$B = \{s_0, s_3\}$, $\{s_2\}$, $\{s_0\}$, $\{s_0, s_2\}$, $\{s_0, s_2, s_3, s_4, s_7\}$ or $\{s_0, s_2, s_4, s_7\}$, label is a or b: then D does not change;*

The set $D = \{S, \{s_0, s_2, s_3\}, \{s_0, s_3\}, \{s_2, s_4, s_7\}, \{s_2\}, \{s_0, s_2, s_3, s_4, s_7\}, \{s_0\}, \{s_0, s_2\}, \{s_0, s_2, s_4, s_7\}\}$ is returned. The algorithm ends at this stage because it has investigated all possible $B \in D$ and all possible labels occuring in S. We see that all NIL states are always in the same set S and hence are all bisimilar and all simulated by every other state of S. The three states s_0, s_2, s_3 are not bisimilar to any other state of S, and s_4 and s_7 are bisimilar. Moreover, we see that s_4 and s_7 are simulated by s_2 for the latter occurs in every set s_4 and s_7 appear in. For the same reason, s_0 simulates s_3.

Note that the set D generates all W-closed sets of S (where W is the logical simulation) as seen in example 3.6.5. It is not easy to see from D what the simulation relation is; one really has to check carefully, which motivates the need for checksim.

If we ran the algorithm bisim *on S, the set*

$$D = \{S, \{s_0, s_2, s_3\}, \{s_0, s_3\}, \{s_2, s_4, s_7\}, \{s_0\}, \{s_2\}\}$$

would be obtained.

4.5 Further aspects of logical characterization

Now we consider the other logics. The proof of the following proposition is very easy and is only sketched here.

Proposition 4.5.1 *All the logics defined in Section 4.1 characterize bisimulation.*

Proof . There is no need to prove that if two systems satisfy all the same formulas they are bisimilar because all the other logics extend $\mathcal{L}$.

For the other direction we have to show two things just as in Proposition 4.2.1 and Theorem 4.2.2. The first is that the sets definable by formulas are measurable. We show that for all formulas ϕ of all our logics, we have $[\![\phi]\!] \in \Sigma$. $[\![\Delta_a]\!] = \tau_a(\cdot, S)^{-1}(\{0\})$ and hence is in Σ. Now for $\mathcal{L}_\bigwedge$ and $\mathcal{L}_\neg$ we only have to show that if $[\![\phi]\!] \in \Sigma$, then so is $[\![\neg\phi]\!]$ which is straightforward, and if $\forall i \in \mathbf{N}$, $[\![\phi_i]\!] \in \Sigma$, then so is $[\![\bigwedge_{i \in \mathbf{N}} \phi_i]\!]$ which is also straightforward since Σ is a σ-field. The results follow by structural induction.

The second is that bisimilar states satisfy the same formulas. Let R be a bisimulation relation between (S, i, Σ, τ) and (S', i', Σ', τ'), and let sRs'. We have $\tau_a(s, S) = \tau'_a(f(s), S')$ because $S \cup S'$ is R-closed, so $s \models \Delta_a$ if and only if $s' \models \Delta_a$. The result is obvious by structural induction for the connectives $\wedge$, $\vee$ and $\neg$. ∎

Although these logics all characterize bisimulation, they do not all characterize equivalence classes, in the sense that there does not necessarily exist a formula for each equivalence class which is satisfied only by states in that class. The most powerful logic does characterize equivalence classes.

Proposition 4.5.2 *The logic $\mathcal{L}_{\bigwedge}$ characterizes equivalence classes of arbitrary Markov processes.*

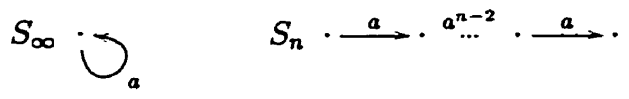
Proof. Let Θ be an equivalence class of processes with respect to $\mathcal{L}_{\bigwedge}$, and $F(\Theta)$ the set of *finite* formulas (i.e. formulas of $\mathcal{L}_{\neg}$) which are satisfied by one member t (hence by all members) of Θ; clearly, this set is countable. Then $\Theta = \bigcap_{\phi \in F(\Theta)} [\![\phi]\!] = [\![\bigwedge_{\phi \in F(\Theta)} \phi]\!]$: indeed, $s \approx t$ if and only if s satisfies all the same formulas of $\mathcal{L}$ as t, i.e., if and only if s satisfies $\bigwedge_{\phi \in F(\Theta)} \phi$. ∎

For finite-state systems negation by itself is enough to characterize equivalence classes.

Proposition 4.5.3 *The logic $\mathcal{L}_{\neg}$ does not characterize equivalence classes of Markov processes, but given a finite Markov chain, for every bisimulation equivalence class, there exists a formula of $\mathcal{L}_{\neg}$ such that a state is in the equivalence class if and only if it satisfies this formula.*

Proof. The last fact of the statement is well known [Arn94]: using the same proof as for the last proposition, we see that given a finite Markov chain, there exists a finite set $A \subseteq F(\Theta)$ such that $\Theta = \bigcap_{\phi \in A} [\![\phi]\!] = [\![\wedge_{\phi \in A} \phi]\!]$, where $\wedge_{\phi \in A} \phi$ is a (finite) formula, as wanted.

This argument does not work if we consider the problem of writing a formula characterizing equivalence classes of arbitrary finite Markov chains and not just the equivalence classes of states within a fixed Markov chain. This happens because there are infinitely many finite Markov chains. We now prove that $\mathcal{L}_\neg$ does not characterize equivalence classes of even finite Markov chains. To do so, consider the equivalence class of the single-state process that can do action a with probability 1 (and then ends up in the same state); this process can do infinitely many a's, call it S_∞. Now let S_n be the process having $n+1$ states that can do the action a n times and then nothing. These processes are illustrated in the following picture.



There is no finite formula that distinguishes S_∞ from all the S_n's at the same time. We prove this by showing that

if $S_\infty \models \phi$, then $\exists N.\forall k \geq N$, $S_n \models \phi$, and
if $S_\infty \not\models \phi$, then $\exists N.\forall k \geq N$, $S_k \not\models \phi$.

The base case corresponding to T is trivial. So assume the statement is true for formulas ϕ, ϕ_1 and ϕ_2. Now assume $S_\infty \models \phi_1 \wedge \phi_2$. Then there exist N_1 and N_2 such that $\forall k \geq N_i, S_k \models \phi_i$, $i = 1,2$. For $N = \max(N_1, N_2)$ we have $\forall k \geq N, S_k \models \phi_1 \wedge \phi_2$. If $S_\infty \not\models \phi_1 \wedge \phi_2$. Then there exists $i \in \{1,2\}$ such that $S_\infty \not\models \phi_i$. So there is N such that $\forall k \geq N_i, S_k \not\models \phi_i$ so $\forall k \geq N, S_k \not\models \phi_1 \wedge \phi_2$. The induction step corresponding to negation is obvious. Finally let $S_\infty \models \langle b \rangle_q \phi$. Then $b = a$ and $S_\infty \models \phi$. By induction hypothesis, there exists an N such that $\forall k \geq N, S_k \models \phi$, so $S_{k+1} \models \langle a \rangle_q \phi$, i.e. $\forall k \geq N+1, S_k \models \langle a \rangle_q \phi$ as wanted. For $S_\infty \not\models \langle b \rangle_q \phi$, there are two cases. Either $b \neq a$ or $b = a$. In the first case, no S_k satisfies $\langle b \rangle_q \phi$, so take $N = 0$; in the second case, we have $S_\infty \not\models \phi$, so there is an N such that $\forall k \geq N, S_k \not\models \phi$. Then for all $k \geq N$ $S_{k+1} \not\models \langle a \rangle_q \phi$, and hence for all $k \geq N+1$ $S_k \not\models \langle a \rangle_q \phi$, and the proof is complete. ∎

Note that the example given in the previous proof cannot be applied to states inside a finite Markov chain, since it involves infinitely many states. Nevertheless, it can be used to show that neither $\mathcal{L}_\neg$ nor $\mathcal{L}_\Delta$ can characterize equivalence classes inside a discrete system satisfying the minimal deviation assumption defined by Larsen and

Skou. Consider the system containing all the S_n's whose initial states are attached to a two-branching tree as in Figure 4.4. This system satisfies the minimal deviation

$$\begin{tikzcd}
i \arrow[r, "a{[1/2]}"] \arrow[d, "a{[1/2]}"'] & S_\infty \\
\cdot \arrow[r, "a{[1/2]}"] \arrow[d, "a{[1/2]}"'] & S_1 \\
\cdot \arrow[r, "a{[1/2]}"] \arrow[d, "a{[1/2]}"'] & S_2 \\
\vdots &
\end{tikzcd}$$

Figure 4.4: A finite-branching process containing S_∞ and all the S_n's

assumption but as argued in the proof, there is no formula that characterizes the equivalence class containing the state of S_∞.

We summarize the results about the different logics as follows. The logic $\mathcal{L}$ characterizes bisimulation of probabilistic processes, without any hypothesis of finite branching and for systems that may have continuous state spaces. The various stronger logics also have this property. In the weak logic $\mathcal{L}$ one cannot write a formula such that any bisimulation equivalence class of states is described by this formula. This holds even for simple finite state systems. On the other hand with just negation added to $\mathcal{L}$ one can characterize the bisimulation equivalence classes of states in a fixed finite state Markov chain but not in countable discrete Markov chains. One cannot characterize equivalence classes of Markov chains with $\mathcal{L}_\neg$. One can characterize bisimulation equivalence classes of states in an arbitrary Markov process using countable conjunction. The logics are not equivalent and we are not obtaining these results just by encoding negation in some way.

Chapter 5

Approximations

We saw previously that one use of bisimulation for labelled Markov processes is to check whether a process is equivalent to a discrete one. The analysis of its behaviour is then greatly simplified. Of course, we cannot possibly expect that this will always happen to any process that we are interested in analyzing. We now change our perspective. Rather than looking for equivalence of a continuous process to a discrete one, we expect to find a discrete process that is somehow "close to" our continuous process. In fact, our goal is to be able to approximate the continuous process "within any bound" that we fix, and hence we want a family of approximations for every process that together contain all the information contained in the continuous one.

We construct such a family of finite-state processes and we show that one can reconstruct the original process –more precisely a bisimulation equivalent of the original process– from the approximants. We do not reconstruct the original state space but we reconstruct all the transition probability information, i.e., the dynamical aspects of the process.

The finite-state approximations will be shown to be simulated by the process so that in some sense they really only capture properties of the original process. This can be useful to verify continuous processes. For example, if we want to check whether a process satisfies some property described by a logical formula, we only have to check that one of its approximations does. Conversely we will show that if a process satisfies a formula, then one of its approximations does, so that the approximants capture all the logically definable properties of the original process.

The construction can be viewed as a kind of "unfolding" construction. As the approximation is refined there are more and more transitions possible. There are two parameters to the approximation, one is a natural number n, and the other is a positive rational ϵ. The number n gives the number of successive transitions possible from the start state. The number ϵ measures the accuracy with which the probabilities approximate the transition probabilities of the original process. Intuitively, every transition in the approximation has probability within ϵ of the corresponding transition in $\mathcal{S}$. We can play with these two parameters when approximating a continuous process. Depending on the intended use of the process, we can work with an approximation or another, for example increasing the accuracy of the probabilities by decreasing ϵ or alternatively increasing the number of possible transitions by letting n be large.

We said earlier that we expect the approximants to be somehow "close to" our continuous process. We make this idea more precise in the next chapter where we introduce a *metric* between processes and prove that the approximants converge in this metric to the process they approximate.

5.1 Finite-state approximation and reconstruction

Given a labelled Markov process $\mathcal{S} = (S, i, \Sigma, \tau)$, an integer n and a rational number $\epsilon > 0$, we construct a finite-state approximation $\mathcal{S}(n, \epsilon)$ to $\mathcal{S}$. The underlying transition system of this approximant forms a *directed acyclic graph (DAG)*; thus a very special kind of finite-state process.

$\mathcal{S}(n, \epsilon)$ is an n-step unfolding approximation of $\mathcal{S}$. Its state-space is divided into $n + 1$ levels which are numbered $0, 1, \ldots, n$. A state is a pair (X, l) where $X \in \Sigma$ and $l \in \{0, 1, \ldots, n\}$. At each level the sets that define states form a partition of S. The initial state of $\mathcal{S}(n, \epsilon)$ is at level n and transitions only occur between a state of one level to a state of one lower level. Thus, in particular, states of level 0 have no outgoing transitions. In the following we omit the curly brackets around singletons. This only happens in finite-state processes.

Definition 5.1.1 *Let (S, i, Σ, τ) be a labelled Markov process, $n \in \mathbf{N}$ and ϵ a positive rational. We denote the finite-state approximation by $\mathcal{S}(n,\epsilon) = (P, p_0, p)$ where P is a subset of $\Sigma \times \{0, \ldots, n\}$. It is defined as follows, for $n \in \mathbf{N}$ and $\epsilon > 0$. $\mathcal{S}(n,\epsilon)$ has $n+1$ levels. States are inductively defined with respect to the level they are in. Level 0 has one state $(S, 0)$. Now, given the m sets from level l, we define states of level $l+1$ as follows. Consider $(B_j)_{j\in I}$ the partition of $[0,1]$ into intervals of size ϵ/m: $\{\{0\}, (0, \epsilon/m], (\epsilon/m, 2\epsilon/m], \ldots\}$. States at level $l+1$ are defined as follows. Let C be a union of sets appearing at level l and a be a label in $\{a_1, \ldots, a_n\}$. For every such choice of C and a we get a partition of S by the sets $\tau_a(\cdot, C)^{-1}(B_j)$, $j \in I$. We take the least common refinement of these partitions obtained by varying over all C and a. Thus if a set X is in this partition of S, $(X, l+1)$ is a state of level $l+1$. Transitions can happen from a state of level $l+1$ to a state of level l, and the transition probability function is given by*

$$p_a((X,k),(B,l)) = \begin{cases} \inf\limits_{t\in X} \tau_a(t,B)) & \text{if } k = l+1, \\ 0 & \text{otherwise.} \end{cases}$$

The initial state p_0 of $\mathcal{S}(n,\epsilon)$ is the state (X, n) such that X contains i, the initial state of $\mathcal{S}$.

If $B = \cup_{j=1}^{k} B_j$, where (B_j, l) is a state of level l in $\mathcal{S}(n,\epsilon)$ for all $j = 1, \ldots, k$, we will often write (B, l) to mean $\{(B_1, l), (B_2, l), \ldots, (B_k, l)\}$. If $s \in S$, we denote by (X_s, l) the unique state (at level l) such that $s \in X_s$. The following lemma is a trivial but useful result. It is true by construction.

Lemma 5.1.2 *Let S be a labelled Markov process, and $s \in S$. In $\mathcal{S}(n,\epsilon)$, if B is a (finite and disjoint) union of sets appearing at level l, then $0 < \tau_a(s, B) - p_a((X_s, l+1), (B, l)) \leq \epsilon$.*

Proof . Let $(X, l+1)$, (B_j, l), $j = 1, \ldots, k$ be states of $\mathcal{S}(n,\epsilon)$. Then for all $s, t \in X$ we have

$$|\tau_a(s, B_j) - \tau_a(t, B_j)| < \epsilon/m,$$

because of the way S is partitioned on level $l+1$ (m is the number of states at level l). Since $k \leq m$, the result follows trivially. ∎

It turns out that every state (X, l) of $\mathcal{S}(n, \epsilon)$ is simulated by every state $s \in X$ from $\mathcal{S}$.

Proposition 5.1.3 *Every labelled Markov process $\mathcal{S}$ simulates all its approximations of the form $\mathcal{S}(n, \epsilon)$. More precisely, every state (X, l) of $\mathcal{S}(n, \epsilon)$ $(l \leq n)$ is simulated by every $s \in X$ from $\mathcal{S}$.*

Proof . Let $\mathcal{S}(n, \epsilon) = (P, p_0, p)$ and $\mathcal{U} = (U, u_0, \Omega, \rho)$ be the direct sum of $\mathcal{S}(n, \epsilon)$ and $\mathcal{S}$. Now let R be the reflexive relation on U relating a state (X, l) from $\mathcal{S}(n, \epsilon)$ to every state $s \in X$ from $\mathcal{S}$. We prove that R is a simulation. Let $X \in \Omega$ be R-closed, that is, $X \cap S \in \Sigma$ and $R(X \cap P) \subseteq X \cap S$. Now consider two related states, (X, l) and $s \in X$. The only positive transition from (X, l) are to states of the form $(B, l-1)$ so we can assume that $X \cap P$ is a union B of states of level $l - 1$. Now observe that $R((B, l - 1)) = B$ and by the preceding lemma we have:

$$\begin{aligned}
\rho_a((X,l),(B,l-1) \cup B) &= p_a((X,l),(B,l-1) \\
&\leq \tau_a(s,B) \\
&= \rho(s,(B,l-1) \cup B),
\end{aligned}$$

and hence the result. ∎

The following lemma shows how the interpretation of logical formulas of $\mathcal{L}$ interacts with the approximation. We use the notation *depth*(ϕ) to stand for the maximum depth of nesting of the modal operator in the formula ϕ.

Lemma 5.1.4 *Let (S, i, Σ, τ) be a labelled Markov process. For every formula $\phi \in \mathcal{L}$ we have*

$$[\![\phi]\!]_S = \cup_j C_j \text{ with } (C_j, l) \in \mathcal{S}(n, \epsilon) \text{ and } C_j \subseteq [\![\phi]\!]_S,$$

where $n \geq l \geq$ depth(ϕ) and all the probabilities occurring in ϕ are integer multiples of ϵ.

Proof . The proof is by induction on the structure of formulas. It is trivial for T. Now assume it is true for ϕ and ψ. Suppose that all the probabilities occurring in $\phi \wedge \psi$ are multiples of ϵ; then this statement is true for both ϕ and ψ. Now let $l \geq depth(\phi \wedge \psi)$; then $l \geq depth(\phi)$ and $l \geq depth(\psi)$. Consequently, we have $[\![\phi]\!]_S = \cup_j C_j$ where the index j runs over all the states (subsets of S) occurring at level l, i.e. $(C_j, l) \in \mathcal{S}(n, \epsilon)$ and $C_j \subseteq [\![\phi]\!]_S$ and similarly for ψ. Now, since sets involved in one level are all disjoint, it is easy to see that the lemma is also true for $\phi \wedge \psi$. Now assume it is true for ϕ and consider the formula $\langle a \rangle_q \phi$. Let l and ϵ be as above for this formula. Then by the induction hypothesis, we have that $[\![\phi]\!]_S$ is a union of sets appearing at level $l-1$ in $\mathcal{S}(n, \epsilon)$ (because $depth(\phi) \leq n-1$). Therefore, sets of level l are partitions of the sets $C_j = \tau_a(\cdot, [\![\phi]\!]_S)^{-1}((j\epsilon/m, (j+1)\epsilon/m])$ and since q is an integer multiple of ϵ/m, there is some k such that $k\epsilon/m = q$ and hence the sets partitioning all C_j for $j \geq k$ form a partition of $[\![\langle a \rangle_q \phi]\!]_S$ and we have the result. ∎

The next theorem is the main result of this section. It shows how the original process can be reconstructed from the approximants.

Theorem 5.1.5 *Let (S, i, Σ, τ) be a labelled Markov process that is maximally collapsed, that is, $S = S/_{\approx}$. If we are given all finite-state approximations $\mathcal{S}(n, \epsilon)$, we can recover (S, i, Σ, τ).*

Proof . We can recover the state space trivially by taking the union of states at any level of any approximation. We know from the fact that $\mathcal{S}$ is maximally collapsed that Σ is generated by the sets of the form $[\![\phi]\!]$, by Theorem 4.2.4. Thus Lemma 5.1.4 implies that

$$\mathcal{B} := \{B : (B, n) \in \mathcal{S}(n, \epsilon) \text{ for some } n \in \mathbf{N} \text{ and some } \epsilon > 0\}$$

generates Σ (obviously, $\mathcal{B} \subseteq \Sigma$).

The main difficulty is that we have to recover the transition probability function. To do so, let $\mathcal{F}(\mathcal{B})$ be the set containing finite unions of sets in $\mathcal{B}$. We first argue that $\mathcal{F}(\mathcal{B})$ forms a field, then we define $\rho_a(s, \cdot)$ on it and we show that $\rho_a(s, \cdot)$ and $\tau_a(s, \cdot)$ agree on it for all $s \in S$. It will imply that $\rho_a(s, \cdot)$ is finitely additive on $\mathcal{F}(\mathcal{B})$ and hence that it can be extended uniquely to a measure on Σ, and hence that ρ_a and τ_a agree on $S \times \Sigma$, as desired.

We show that $\mathcal{F}(\mathcal{B})$ forms a field. It is obviously closed under finite unions. To see that it is also closed under intersection and complementation, note that if $(C, n) \in \mathcal{S}(n, \epsilon)$, then for all $m > n$ and all δ such that ϵ is an integer multiple of δ, C is a union of a family of sets C_i such that $(C_i, m) \in \mathcal{S}(m, \delta)$.

Now let $C \in \mathcal{F}(\mathcal{B})$, $s \in S$, $a \in \mathcal{A}$ and let

$$\rho_a(s, C) := \sup_{n,\epsilon} \sum_{\substack{B \subseteq C \\ (B,n-1) \in \mathcal{S}(n,\epsilon)}} p_a((X_s, n), (B, n-1)).$$

We prove that $\rho_a(s, \cdot)$ and $\tau_a(s, \cdot)$ agree on $\mathcal{F}(\mathcal{B})$ for all $s \in S$. Obviously, $\rho_a(s, C) \leq \tau_a(s, C)$ for $C \in \mathcal{F}(\mathcal{B})$. The reverse inequality follows from Lemma 5.1.2:

$$\begin{aligned}
\sup_{n,\epsilon} \sum_{\substack{B \subseteq C \\ (B,n-1) \in \mathcal{S}(n,\epsilon)}} p_a((X_s, n), (B, n-1)) &= \sup_{n,\epsilon} p_a((X_s, n), (\cup B, n-1)) \\
&\geq \sup_{n,\epsilon} (\tau_a(s, \cup B) - \epsilon) \\
&\geq \sup_{(n,\epsilon) \in I} (\tau_a(s, C) - \epsilon) \\
&= \tau_a(s, C),
\end{aligned}$$

where I is the set of pairs (n, ϵ) such that in $\mathcal{S}(n, \epsilon)$, level n contains a partition of C (note that there are arbitrary small ϵ's that are involved in I). This concludes the proof that ρ and τ agree and we are done. ∎

The next result shows that if one is interested in logical reasoning about processes then any formula is satisfied by one of the finite-state approximants.

Theorem 5.1.6 *If a state $s \in S$ satisfies a formula $\phi \in \mathcal{L}$, then there is some approximation $\mathcal{S}(n, \epsilon)$ such that $(X_s, n) \models \phi$.*

Proof. The proof is by induction on the structure of formulas. However a direct induction proof does not work in any obvious way. We need to prove a significantly stronger result in order to use a stronger induction hypothesis. We prove that for all formula ϕ and every $l \geq depth(\phi)$ there is an increasing sequence $(X_n)_{n \geq l}$ of sets in Σ which satisfy:

(i) $\cup_{n\geq l} X_n = [\![\phi]\!]_S$;

(ii) $\exists (C_j, l) \in \mathcal{S}(n, 1/2^n)$, $j=1,\ldots,m$, such that $X_n = \cup_{j=1}^m C_j$, $n \geq l$;

(iii) the states (C_j, l) satisfy ϕ in $\mathcal{S}(n, 1/2^n)$.

It is obvious for T for which you choose $X_n = S$ for all n. We fix $\epsilon_n = 1/2^n$. Note that every non-trivial formula ϕ is of the form $\wedge_{j=1}^k \langle a_j\rangle_{q_j}\phi_j$, so assume the claim is true for ϕ_j, $j=1,\ldots,k$ and let $l \geq \mathit{depth}(\wedge_{j=1}^k \langle a_j\rangle_{q_j}\phi_j)$. Then $l-1 \geq \mathit{depth}(\phi_j)$ for all $j=1,\ldots,k$. Let $(X_n^j)_{n\geq l-1}$ be the sequence for ϕ_j at level $l-1$. Now define for $n \geq l$, the sequence

$$B_n = \{s \in S : \tau_{a_j}(s, X_n^j) > q_j + \epsilon_n,\ j=1,\ldots,k\}.$$

Note that this is an increasing sequence of sets in Σ. We first prove (i), that is, for all $s \models \phi$, there is some n such that $s \in B_n$. So assume $\tau_{a_j}(s, [\![\phi_j]\!]) > q_j$ for all $j=1,\ldots,k$. Then, since $\tau_{a_j}(s,\cdot)$ is a measure and X_n^j is an increasing sequence which converges to $[\![\phi_j]\!]$, there is some n such that $\tau_{a_j}(s, X_n^j) > q_j$, $j=1,\ldots,k$. Moreover, there is some n such that $\tau_{a_j}(s, X_n^j) > q_j + \epsilon_n$, $j=1,\ldots,k$, because X_n^j is increasing and ϵ_n is decreasing to 0. Thus $s \in B_n$ and (i) is proved. We now prove (ii) and (iii). Let $s \in B_n$, for a fixed $n \geq l$. Then because all states $(X, l-1)$, where $X \subseteq X_n^j$, satisfy ϕ and by Lemma 5.1.2, we have

$$\begin{aligned} p_{a_j}((C_s, l), ([\![\phi_j]\!]_{S(n,\epsilon)}, l-1)) &\geq p_{a_j}((C_s, l), (X_n^j, l-1)) \\ &\geq \tau_{a_j}(s, X_n^j) - \epsilon_n \\ &> q_j + \epsilon_n - \epsilon_n = q_j, \end{aligned}$$

and hence, $(C_s, l) \models \phi$. This means that B_m is a union of sets of level l which satisfy ϕ, as wanted in (ii) and (iii). So the proof is complete. ∎

The next proposition shows how the partitions produced in the construction define equivalence relations which give "in the limit" the bisimulation relation.

Proposition 5.1.7 *Let $\mathcal{S} = (S, i, \Sigma, \tau)$ be a process. Then states $s, t \in S$ are bisimilar if and only if they are in the same partition in all finite-state approximations to $\mathcal{S}$.*

Proof . It is not hard to show by induction that every union of states $(B_j)_{1\leq j\leq k}$ at some level of an approximation corresponds (in $\mathcal{S}$) to a formula of $\mathcal{L}$ to which we add disjunction and negation; this formula is satisfied in $\mathcal{S}$ by and only by the states that belong to $\cup_j B_j$ in $\mathcal{S}$. So if two states don't belong to the same state of an approximation, they are distinguishable by a formula of that logic and hence are not bisimilar by Proposition 4.5.1.

If s and t are not bisimilar, then there is a formula $\phi \in \mathcal{L}$ that distinguishes them. So assume $s \models \phi$ and $t \not\models \phi$. Then by Theorem 5.1.6, there is an approximation $\mathcal{S}(n, \epsilon)$ such that $(X_s, n) \models \phi$. The state t cannot be in X_s because it would then satisfy ϕ, by Theorem 5.1.3, which is not true. ∎

This last result justifies working with the approximants if one is interested in reasoning about bisimulation.

5.2 Example

We compute a few approximations of a simple continuous process. States are from the set $\{s,t\} \cup [0,3]$, the initial state is 1 and transitions are as follows:

- if $x \in [0,1]$, $p_a(x,[0,y)) = \frac{x+y}{4}$, where $0 \leq y \leq 1$.
  $p_a(x,\{1\}) = \frac{1-x}{4}$,
  $p_a(x,(1,1+y]) = \frac{y}{4}$,
  $p_a(x,(2,2+y]) = \frac{xy}{4}$,
- if $x \in (1,2]$, $p_a(x,s) = 1$.
- if $x \in (2,3]$, $p_b(x,t) = 1$.

We draw this process in an informal way in Figure 5.1, where we label the transitions with expressions that should be interpreted as above.

Let us compute the approximation $\mathcal{S}(2,1/2)$. At level 0, we have state $(S,0)$. At level 1, we partition S according to the partition of $[0,1]$ into intervals of size 1/2: $\{\{0\},(0,1/2],(1/2,1]\}$. Note that if $x \in [0,1]$, then $p_a(x,S) = \frac{x+1}{4}+\frac{1-x}{4}+\frac{1}{4}+\frac{x}{4} = \frac{3+x}{4}$. Hence

$$p_a(x,S)\begin{cases} =0 & \text{if } x \notin [0,2] \\ \in (1/2,1] & \text{if } x \in [0,2] \end{cases} \qquad p_b(x,S) = \begin{cases} 0 & \text{if } x \notin (2,3] \\ 1 & \text{if } x \in (2,3]. \end{cases}$$

$$\begin{tikzcd}
 & {[0,1)} \arrow[loop above, "a[\frac{x+y}{4}]"] \arrow[dl, "a[\frac{y}{4}]"'] \arrow[d, bend left, "a[\frac{1-x}{4}]"] \arrow[dr, "a[\frac{xy}{4}]"] & \\
{(1,2]} \arrow[d, "a"'] & 1 \arrow[u, bend left, "a[\frac{1+y}{4}]"] \arrow[l, "a[\frac{y}{4}]"] \arrow[r, "a[\frac{y}{4}]"'] & {(2,3]} \arrow[d, "b"] \\
s & & t
\end{tikzcd}$$

Figure 5.1: A simple continuous process

This yields, for level 1, the sets $[0,2]$, $(2,3]$ and $\{s,t\}$. The transitions are as in the following picture which represents the approximation $\mathcal{S}(1,1/2)$.

$$\begin{tikzcd}
(S,0) & & \\
([0,2],1) \arrow[u, "a[3/4]"] & ((2,3],1) \arrow[ul, "b"'] & (\{s,t\},1)
\end{tikzcd}$$

The initial state is $([0,2],1)$ and from the picture, one can see that this state satisfies the formula $\langle a\rangle_{3/4-\epsilon}\mathsf{T}$, for all $\epsilon > 0$.

Now for level two of $\mathcal{S}(2,1/2)$, the partition of S will be obtained using all transitions to any union of sets that appear at level 1. We must consider the partition of $[0,1]$ into intervals of size $1/2 \times 1/3$ since there are 3 states at level 1.

$$p_a(x,\{s,t\}) = \begin{cases} 0 & \text{if } x \notin (1,2] \\ 1 & \text{if } x \in (1,2] \end{cases} \qquad p_b(x,\{s,t\}) = \begin{cases} 0 & \text{if } x \notin (2,3] \\ 1 & \text{if } x \in (2,3] \end{cases}$$

Now for $x \in [0,1]$, we have

$$\begin{aligned}
p_a(x,[0,3]) &= (3+x)/4 \in \begin{cases} (4/6,5/6] & \text{if } x \in [0,1/3] \\ (5/6,1] & \text{if } x \in (1/3,1] \end{cases} \\
p_b(x,[0,3]) &= 0 \\
p_a(x,[0,2]) &= \frac{x+1}{4} + \frac{1-x}{4} + \frac{1}{4} = 3/4 \\
p_a(x,(2,3]) &= x/4 \in \begin{cases} (0,1/6] & \text{if } x \in (0,2/3] \\ (1/6,2/6] & \text{if } x \in (2/3,1], \end{cases}
\end{aligned}$$

It is easy to see that the sets we have not considered would not be useful for the partition of S, so we ignore them; these sets are formed by unions of the set $\{s, t\}$ with other sets of level one.

Thus we get the following sets constituting the partition of S at level 2: $\{0\}$, $(0, 1/3]$, $(1/3, 2/3]$, $(2/3, 1]$, $(1, 2]$, $(2, 3]$, $\{s, t\}$. Transitions are illustrated in the following picture, where we omit the levels. In order not to clutter up the picture, we have not labelled the dotted lines. Dotted lines represent a-transitions with probability 3/4, so should be labelled $a[\frac{3}{4}]$.

$$\begin{tikzcd}
 & & S & & & & \\
 & {[0,2]} \arrow[ur, "{a[3/4]}"] & & {(2,3]} \arrow[ul, "b"'] & & \{s,t\} & \\
\{0\} \arrow[ur, dotted, no head] \arrow[ur, dotted] & {(0,\frac{1}{3}]} \arrow[u, dotted] & (\frac{1}{3},\frac{2}{3}) \arrow[ul, dotted] \arrow[ur, "{a[\frac{1}{12}]}"] & {(\frac{2}{3},1]} \arrow[ull, dotted] \arrow[u, "{a[\frac{1}{6}]}"'] & {(1,2]} \arrow[ur, "a"'] & {(2,3]} \arrow[u, "b"'] & \{s,t\}
\end{tikzcd}$$

This picture represents the approximation $\mathcal{S}(2, 1/2)$. The initial state is $((2/3, 1], 2)$ and from the picture, it is easy to see that this state satisfies the following formulas where $\epsilon > 0$.

$$\langle a \rangle_{\frac{11}{12}-\epsilon} \top, \qquad \langle a \rangle_{3/4-\epsilon} \langle a \rangle_0 \top, \qquad \langle a \rangle_{1/6-\epsilon} \langle b \rangle_0 \top.$$

This implies that in the original process, the initial state also satisfies these formulas.

We can obtain more complex formulas by studying approximation $\mathcal{S}(3, 1/2)$. The partition of S that is generated isolates state 1 of the original process. We draw part of this approximation in Figure 5.2, keeping only the initial state at level 3. Here again, dotted lines represent a-transitions with probability 3/4. Transitions from the initial state $(\{1\}, 3)$ are all a-transitions.

In this picture we see that the initial state satisfies in particular the following formulas, where $\epsilon > 0$:

$$\langle a \rangle_{1-\epsilon} \top, \qquad \langle a \rangle_{1/2-\epsilon} \langle a \rangle_0 \langle a \rangle_0 \top, \qquad \langle a \rangle_{1/6-\epsilon} \langle a \rangle_0 \langle b \rangle_0 \top.$$

These formulas are also satisfied by the initial state of the original process.

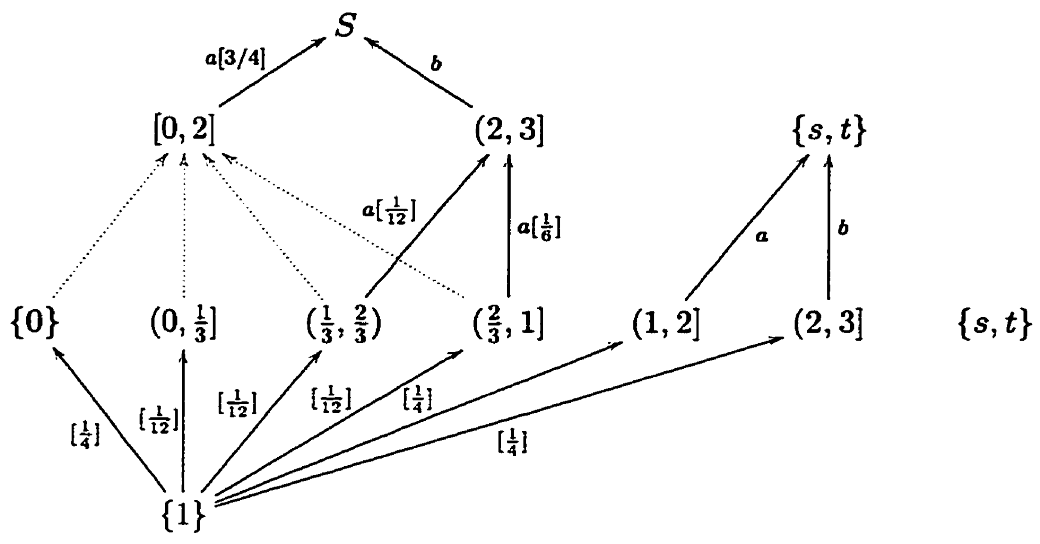


Figure 5.2: Approximation $\mathcal{S}(3, 1/2)$

Note that the set $(1, 2]$ will never be split in any approximations, for it contains only bisimilar states. By changing the probability of state $x \in (1, 2]$ of jumping with label a to state s to the value $x - 1$, one can get a more complex process.

Chapter 6

Metrics and logic via real functions

To compare processes, we now have two notions. The first one is bisimulation which tells us when two processes are essentially equal. If they differ slightly, bisimulation only tells us that they are not bisimilar. The second notion we introduced is simulation which tells us a little more, being a preorder. Two processes, although not bisimilar, can be related by a simulation relation; one can be greater than the other. But simulation does not tell us how far the processes are in a quantitative fashion and, since it is not a total order, there are pairs of processes for which it tells us nothing precise. Like bisimulation, it is not robust; a very small change in probabilities will likely result in non-bisimilar and "non-similar" processes. We will now introduce a *metric* that will allow us to refine our view of processes. This metric will assign a number to every pair of processes, giving so an indication of how far they are from each other. If the metric distance is 0, then the two processes will turn out to be bisimilar, and conversely. Processes that are very "close" to being bisimilar will get smaller distance than processes that are "far" from being bisimilar. The number itself will not be of great importance, as is usually the case with metrics. It is the relative distance that will be of interest, in particular the notion of convergence of processes it engenders. In fact, we will introduce a family of metrics that assign different weights to the difference between processes that appear deeper in the execution or history of the processes.

In order to define these metrics, we need to shift from the traditional view of logical formulas to measurable functions into $[0, 1]$. Working with measurable functions one has the right setting to talk about convergence. These functions will play the same role as the logic formulas from Chapter 4, but in addition, they will provide us with numbers that we will use to define the metrics. Our technical development is based on a key idea by Kozen [Koz85] to generalize logic to handle probabilistic phenomena. What Kozen suggested is that instead of logical formulas we use measurable functions, instead of truth values, we have values in $[0, 1]$ and instead of satisfaction, we have integration. This idea was based on states being distributions on the state space; in our case, states are just ordinary states and hence we use evaluation of the measurable function at a state instead of integration.

In the preceding chapter, we introduced approximations to labelled Markov processes. The metric that we define here witnesses the fact that the approximations are as close as we want to the continuous process they approximate.

We first give the alternate presentation of probabilistic logic using functions into the reals and show that these functions accord well with simulation and bisimulation. We then show that the value of each of these functions on a labelled Markov process is the limit of its value at the approximations to that process. Finally, we define the metrics on processes and show that the approximations to a process converge to that process and that the processes form a separable metric space.

The notion of metric was developed in joint work [DGJP99b]. The results from 6.1.12 onwards are mine alone.

6.1 Probabilistic logic via functions into $[0, 1]$

We define a set of *functional expressions* by giving an explicit syntax. It is worth clarifying our terminology here. A functional expression becomes a function when we interpret it in a process. Sometimes we may loosely say "the same function" when we move from one process to another. What we really mean is the "same functional expression"; obviously it cannot be the same function when the domains are different. This is no different from having syntactically defined formulas of some logic which become boolean-valued functions when they are interpreted on a structure.

Definition 6.1.1 *For each $c \in (0,1]$, we consider a family $\mathcal{F}^c$ of functional expressions generated by the following grammar.*

$$f^c := 1 \mid \langle a\rangle f^c \mid \min(f_1^c, f_2^c) \mid f^c \ominus q \mid \lceil f^c \rceil^q,$$

where q is a rational. The interpretation is as follows. Let $\mathcal{S} = (S, i, \Sigma, \tau)$ be a labelled Markov process. We write $f_{\mathcal{S}}^c : S \longrightarrow [0,1]$ for the interpretation of f^c on $\mathcal{S}$ and drop the subscript when no confusion can arise. Let $s \in S$. Then

$$\begin{aligned}
&1(s) = 1,\\
&\langle a\rangle f^c(s) = c \int_S f^c(t)\tau_a(s, dt),\\
&\min(f_1^c, f_2^c)(s) = \min(f_1^c(s), f_2^c(s))\\
&f^c \ominus q(s) = \max(f^c(s) - q, 0),\\
&\lceil f^c \rceil^q(s) = \min(f^c(s), q).
\end{aligned}$$

When working with functionals, we will always consider functionals of a single family, that is, we always fix $c \in (0,1]$: thus we will often drop the exponent that is attached to the functionals and write $f(s)$ instead of $f^c(s)$. We will use $\langle a\rangle^n f$ to represent $\langle a\rangle \cdots \langle a\rangle f$ where $\langle a\rangle$ appears n times.

Definition 6.1.2 *$\mathcal{F}_\vee^c$ is the family $\mathcal{F}^c$ to which we add the functional* $\max(f_1, f_2)$.

Example 6.1.3 *Consider the first two processes of figure 6.1. All transitions are*

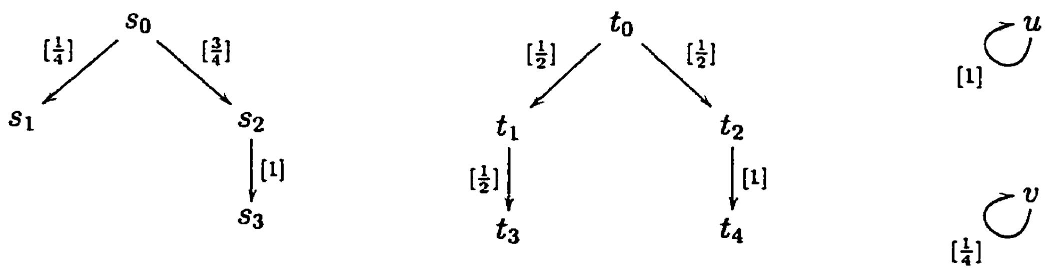


Figure 6.1:

labelled a. The functional expression $(\langle a\rangle 1) \in \mathcal{F}^c$ evaluates to c at states having indices 0 or 2; it evaluates to 0 at states s_1, s_3, t_3, t_4, and it evaluates to $c/2$ at state

t_1. *The functional expression* $(\langle a\rangle\langle a\rangle 1)) \in \mathcal{F}^c$ *evaluates to* $3c^2/4$ *at* s_0 *and* t_0 *and to* 0 *elsewhere. The functional expression* $\langle a\rangle(((\langle a\rangle 1) \ominus \frac{c}{2}) \in \mathcal{F}^c$ *evaluates to* $3c^2/8$ *at state* s_0 *and to* $c^2/4$ *at state* t_0.

Example 6.1.4 *Consider the processes* u *and* v *of figure 6.1. All transitions are labelled* a. *A functional expression of the form* $(\langle a\rangle^n 1)) \in \mathcal{F}^c$ *evaluates to* c^n *at state* u. *On state* v *the same functional expression evaluates to* $(c/4)^n$.

6.1.1 Functional expressions vs bisimulation and simulation

We show that the functional viewpoint is sound and complete for bisimulation and, in the discrete case, for simulation. We do so by relating the logical formulas of $\mathcal{L}_\vee$ to the functionals we have introduced.

The two first results say that functionals reflect bisimulation and simulation. Functionals will assign smaller values to a simulated state than to one that simulates it. This will imply that if two states are bisimilar, then every functional will assign the same value to each of them.

The following lemma is the functional analogue of Proposition 4.3.1.

Lemma 6.1.5 *If* R *is a simulation relation between processes* $\mathcal{S}$ *and* $\mathcal{S}'$, *and if* $s \in S$ *is* R*-related to* $s' \in S'$, *then for all* $f \in \mathcal{F}^c_\vee$, *we have* $f_{\mathcal{S}}(s) \leq f_{\mathcal{S}'}(s')$.

Proof . We prove the lemma for R a simulation on a single process $\mathcal{S}$. For the general case, observe that for every state of a process, the value of a functional is the same in the process as in any direct sum of which it is a summand. The proof proceeds by well-founded induction on the construction of the functional expression f. The key case is $g = \langle a\rangle f$. By the inductive assumption on f, we have $s' \; R \; t' \Rightarrow f(s') \leq f(t')$. Let $s \; R \; t$. We will prove that $g(s) \leq g(t)$.

Let $\mu(A) = \tau_a(s, A)$, and $\nu(A) = \tau_a(t, A)$. Consider simple functions[1] h derived from f as follows: let $v_1 < \ldots < v_n$ be finitely many values of f. Define $h(s) = \max\{v_i \mid v_i \leq f(s)\}$. This satisfies $f(x) \leq f(y) \Rightarrow h(x) \leq h(y)$ and $(\forall x)\, h(x) \leq f(x)$.

[1] A function is simple if its range is finite.

We have

$$\int f d\mu = \sup_h \int h d\mu$$

where the sup ranges over all simple functions h that satisfy the above conditions. Hence, it suffices to prove that for any such simple function h, $\int h d\mu \leq \int h d\nu$.

Consider one such h with range $\{v_1 \dots v_n\}$. Then, for each $1 \leq i \leq n$, the set $S_i = h^{-1}\{v_i \dots v_n\}$ is measurable. It is also R-closed, as if $s \in S_i$ and sRt then $f(s) \leq f(t)$, so $t \in S_i$. Thus, for each S_i, $\mu(S_i) \leq \nu(S_i)$. Now, we have $\int h d\mu = \sum_i (v_i - v_{i-1})\mu(S_i) \leq \sum_i (v_i - v_{i-1})\nu(S_i) = \int h d\nu$ and the result is proved. ∎

Since every bisimulation is a simulation, this last result allows us to prove the following functional analogue of Theorem 4.2.2.

Corollary 6.1.6 *Let $\mathcal{S}$, $\mathcal{S}'$ be labelled Markov processes and $s \in S$, $s' \in S'$ be bisimilar states. Then for all $c \in (0,1]$ and for all $f \in \mathcal{F}^c$ or $\mathcal{F}^c_\vee$, we have $f_\mathcal{S}(s) = f_{\mathcal{S}'}(s')$.*

We would like to prove the converse of the two previous lemmas: we want it to be the case that if the value of any functional is the same on a pair of states, then the states are bisimilar. We also would like that if the value of any functional is always smaller for a state s than for another state s', then s is simulated by s'. This last result will be proven in the case where the simulated state comes from a discrete process, similarly to the characterization of simulation by the logic $\mathcal{L}_\vee$. In order to use the results we have that the logic $\mathcal{L}_\vee$ characterizes bisimulation and simulation, we would like to associate to every formula of $\mathcal{L}_\vee$, a functional in $\mathcal{F}^c_\vee$ that assigns a positive value to states that satisfy the formula and value zero to states that do not.

The following two results will prove only part of this but will be enough to prove the characterizations we are looking for.

Lemma 6.1.7 *Given $\phi \in \mathcal{L}_\vee$, any finite labelled Markov process $\mathcal{P}$, and any $c \in (0,1]$, there exists $f \in \mathcal{F}^c_\vee$ such that*

1. *$\forall p \in P$, $f_\mathcal{P}(p) > 0$ if and only if $p \models \phi$.*
2. *for any other labelled Markov process $\mathcal{S}$, $\forall s \in S$, $f_\mathcal{S}(s) > 0 \Rightarrow s \models \phi$.*

For the $\mathcal{L}$ subfragment of the logic, the resulting function is in $\mathcal{F}^c$.

Proof. Let $\mathcal{P} = (P, p_0, \rho)$ be a finite-state process. The proof is by induction on the structure of ϕ. If $\phi = \mathsf{T}$, the functional expression 1 suffices. If $\phi = \psi_1 \wedge \psi_2$, let f_1 and f_2 be the functional expressions corresponding to ψ_1 and ψ_2. Then the functional expression $\min(f_1(s), f_2(s))$ satisfies the conditions. If $\phi = \phi_1 \vee \phi_2$, let f_i be the functional expression corresponding to ϕ_i, $i = 1, 2$. Then the functional expression $\max(f_1, f_2)$ satisfies the conditions.

If $\phi = \langle a \rangle_q \psi$, let g be the functional expression corresponding to ψ yielded by induction. Let $x = \min\{g(p) \mid p \in [\![\psi]\!]_{\mathcal{P}}\}$. By induction hypothesis, $x > 0$. Consider the functional expression f given by $(\langle a \rangle \lceil g \rceil^x) \ominus cxq$. For all $t \in [\![\psi]\!]_{\mathcal{P}}$, $(\lceil g \rceil^x)(t) = x$. For any state $p \in P$,

$$(\langle a \rangle \lceil g \rceil^x)(p) = cx \sum_{t \in [\![\psi]\!]} \rho_a(p, t) = cx\rho_a(p, [\![\psi]\!]_{\mathcal{P}}).$$

Note that since there are finitely many states, the integral becomes a summation. Now for each state $p \in [\![\phi]\!]_{\mathcal{P}}$, $\rho_a(p, [\![\psi]\!]_{\mathcal{P}}) > q$. Thus f satisfies the first condition.

The second condition holds because for any state s in $\mathcal{S}$, we have

$$(\langle a \rangle \lceil g \rceil^x)(s) \leq cx\tau_a(s, [\![\psi]\!]_S),$$

so if $\tau_a(s, [\![\psi]\!]_S) \leq q$ then $(\langle a \rangle \lceil g \rceil^x) \ominus cxq(s) = 0$. ∎

The results of the earlier sections about approximations enable us to extend the previous lemma to any state of any labelled Markov process. Note that in the following statement, the functional for ϕ also depends on the state s.

Corollary 6.1.8 *Given $\phi \in \mathcal{L}_\vee$, any labelled Markov process $\mathcal{S}$, a state $s \in S$ and any $c \in (0, 1]$, if $s \models \phi$, there exists $f \in \mathcal{F}^c_\vee$ such that*

1. *$f_S(s) > 0$;*

2. *for any other labelled Markov process $\mathcal{S}'$, $\forall s' \in S'$, $f_{S'}(s') > 0 \Rightarrow s' \models \phi$.*

If ϕ is in $\mathcal{L}$, then the resulting function is in $\mathcal{F}^c$.

Proof. Let $\mathcal{S}$ be an arbitrary process and $\phi \in \mathcal{L}_\vee$. Let s be a state in $\mathcal{S}$ such that $s \models \phi$. By Theorem 5.1.6, there is a finite approximation $\mathcal{S}(n,\epsilon)$ of $\mathcal{S}$ such that $(X_s, n) \models \phi$. By Lemma 6.1.7, $\exists f \in \mathcal{F}^c_\vee$ such that $f_{\mathcal{S}(n,\epsilon)}((X_s,n)) > 0$ and for any process $\mathcal{S}'$, $\forall s' \in S'.s' \not\models \phi \Rightarrow f_{\mathcal{S}'}(s') = 0$. It remains to show the first condition, i.e., that $f_\mathcal{S}(s) > 0$. By Proposition 5.1.3, we have that s simulates (X_s, n). Thus by Lemma 6.1.5, $f_\mathcal{S}(s) > f_{\mathcal{S}(n,\epsilon)}((X_s,n)) > 0$, so f satisfies the conditions required. ∎

Example 6.1.9 *Given a formula ϕ of $\mathcal{L}_\vee$, the functional f_ϕ that is computed for the previous two results satisfies: $f_\top = 1$; $f_{\phi\wedge\psi} = \min(f_\phi, f_\psi)$; $f_{\phi\vee\psi} = \max(f_\phi, f_\psi)$; for any state s in process $\mathcal{P}$, $f_{\langle a\rangle_q \top}(s) = ((\langle a\rangle.1) \ominus q)(s) = \max(\tau_a(s,P) - q, 0)$.*

The next result says that functions are sound and complete for bisimulation.

Theorem 6.1.10 *For any labelled Markov process $\mathcal{S}$ and any $c \in (0,1]$, if two states of $\mathcal{S}$ are assigned the same value for every functional of $\mathcal{F}^c$ or $\mathcal{F}^c_\vee$, then they are bisimilar.*

Proof. We prove that two states that do not satisfy the same formulas of $\mathcal{L}$ will get different values for some $f \in \mathcal{F}^c$. Let ϕ be such that $s \models \phi$ and $s' \not\models \phi$. By Lemma 6.1.8, there is a functional expression $f \in \mathcal{F}^c$ such that $f_\mathcal{S}(s) > 0$ and $f_\mathcal{S}(s') = 0$. The result follows by Theorem 4.2.9. ∎

Example 6.1.11 *Consider the two first processes of figure 6.1. The calculations of example 6.1.3 show that s_0 and t_0 are distinguishable. However, the states are indistinguishable if we use only the functionals 1, $\langle a\rangle f$, and $\min(f_1, f_2)$. Thus, example 6.1.3 shows that additional functional expressions are necessary. Note that the fact that $f \ominus q$ and $\lceil f\rceil^q$ are defined from* min *and* max *does not contradict this claim. To define these functionals we need that the constant function which returns q to every state be definable, and moreover, $f \ominus q$ needs that the difference $f - q$ be also definable; these functionals are not definable from the three functionals we enumerate above not even if we add $\max(f_1, f_2)$ to this list.*

We now show the completeness result for simulation.

Proposition 6.1.12 *A state p from a finite process $\mathcal{P}$ is simulated by a state s of another process $\mathcal{S}$ if and only if for every functional $f \in \mathcal{F}^c_\vee$, we have $f_\mathcal{P}(p) \leq f_\mathcal{S}(s)$.*

Proof . Let p be a state of $\mathcal{P} = (P, p_0, \rho)$ and s a state of $\mathcal{S} = (S, i, \Sigma, \tau)$. Necessity is given by Lemma 6.1.5. For sufficiency, assume that for every functional $f \in \mathcal{F}^c_\vee$, we have $f_\mathcal{P}(p) \leq f_S(s)$. We prove that p is logically simulated by s. Let $\phi \in \mathcal{L}_\vee$ such that $p \models \phi$. Then by Lemma 6.1.7, there exists $f \in \mathcal{F}^c_\vee$ such that $f_\mathcal{P}(p) > 0$ and for all $s \in S$, $f_S(s) > 0 \Rightarrow s \models \phi$. But by hypothesis $f_\mathcal{P}(p) \leq f_S(s)$ which implies that $s \models \phi$. Now by Theorem 4.3.2, p is simulated by s. ∎

6.1.2 Finite approximations and functional expressions

In the last chapter, we proved that every formula satisfied by a process was satisfied by one of its approximants and conversely. Hence the formulas that are satisfied by a process are exactly those that are satisfied by its approximants. We now show the corresponding result for functionals, that is, the value of a functional on some state of a process is the limit of its values on the approximants.

Define the depth of $f \in \mathcal{F}^c$ inductively as follows:

$$\begin{aligned}
&depth(1) = 0\\
&depth(\langle a\rangle f) = depth(f) + 1\\
&depth(\min(f_1, f_2)) = \max(depth(f_1), depth(f_2))\\
&depth(f \ominus q) = depth(\lceil f \rceil^q) = depth(f).
\end{aligned}$$

The following result will be essential for proving that the approximants converge in the metric to the process they approximate.

Lemma 6.1.13 *Let (S, i, Σ, τ) be a labelled Markov process, $\mathcal{S}(n, \epsilon)$ one of its approximations, and f a functional expression in $\mathcal{F}^c$ of depth $\leq n$ involving only labels $a_1, \ldots, a_n$. Then*

$$0 \leq f_S(s) - f_{S(n,\epsilon)}((X_s, n)) \leq n\epsilon.$$

Proof . By Proposition 5.1.3, we have that s simulates (X_s, n); thus by Lemma 6.1.5, $f_S(s) - f_{S(n,\epsilon)} \geq 0$ for all s, n, ϵ. Now let $depth(f) = d \leq n$, we prove that for every state $s \in S$ and for every l with $d \leq l \leq n$, $|f_S(s) - f_n((X_s, l))| \leq l\epsilon$ (we write f_n for the evaluation of f in $\mathcal{S}(n,\epsilon)$). This will be proved by induction on the structure of f. Of course, the inequality is true if f is 1. Now assume that the inequality is true for f, i.e.,

$$f_S(t) - f_n(B_t, l) \leq l\epsilon, \text{ for } d \leq l \leq n, t \in S.$$

It is easy to verify that the constructors min, $f \ominus q$ and $\lceil f \rceil^q$ satisfy the inequality. Now for $\langle a \rangle.f$ we have

$$\begin{aligned}
&\langle a \rangle.f_S(s) - \langle a \rangle.f_n((X_s, l+1)) \\
&\quad = c\int_S f_S(t)\tau_a(s,dt) - c\sum_{(B,l)\in\mathcal{S}(n,\epsilon)} f_n((B,l))p_a((X_s,l+1),(B,l)) \\
&\quad = c\sum_{(B,l)\in\mathcal{S}(n,\epsilon)} [\int_B f_S(t)\tau_a(s,dt) - \int_B f_n((B,l))\tau_a(s,dt)] \\
&\qquad + c\sum_{(B,l)\in\mathcal{S}(n,\epsilon)} [\int_B f_n((B,l))\tau_a(s,dt) - f_n((B,l))p_a((X_s,l+1),(B,l))] \\
&\quad = c\sum_{(B,l)\in\mathcal{S}(n,\epsilon)} \int_B [f_S(t) - f_n((B,l))]\tau_a(s,dt) \\
&\qquad + c\sum_{(B,l)\in\mathcal{S}(n,\epsilon)} f_n((B,l))[\tau_a(s,B) - p_a((X_s,l+1),(B,l))] \\
&\quad \leq c\sum_{(B,l)\in\mathcal{S}(n,\epsilon)} l\epsilon\tau_a(s,B) + c\sum_{(B,l)\in\mathcal{S}(n,\epsilon)} \tau_a(s,B) - p_a((X_s,l+1),(B,l)) \\
&\quad \leq cl\epsilon + c(\tau_a(s,S) - p_a((X_s,l+1),(S,l))) \\
&\quad \leq cl\epsilon + c\epsilon \leq (l+1)\epsilon.
\end{aligned}$$

The first inequality follows by induction hypothesis and from the fact that f_n is less than 1, and hence is true for all $d \leq l \leq n$. The last inequality follows from Lemma 5.1.2. So we proved that for $\langle a \rangle.f$, which is of depth $d+1$,

$$\langle a \rangle.f(s) - \langle a \rangle.f_n((X_s, m)) \leq m\epsilon$$

for all (X_s, m), where $d+1 \leq m \leq n$, as wanted. ∎

Corollary 6.1.14 *Let (S, i, Σ, τ) be a labelled Markov process and f a functional expression of $\mathcal{F}^c$ involving labels $a_1, \ldots, a_N$. Then for all $n \geq \max(depth(f), N)$ and for every $s \in S$,*

$$f_S(s) = \lim_{\epsilon\to 0} f_{\mathcal{S}(n,\epsilon)}((X_s, n)).$$

6.2 Metrics and convergence of approximations

We introduce the notion of metric between arbitrary processes and show how the approximants of a process converge to the original process under this metric. Intuitively the metrics measure how "visibly" different the processes are. In terms of logic one can say that two processes are very close if the formulas that tell them apart are very long or complex. To capture this intuition quantitatively we use the functionals introduced in the last section. There is now a second notion of how far apart processes are; the distinguishing functions could have values which are very different or only slightly different. We actually study a family of definitions which assign different weights to these differences[2].

Definition 6.2.1 *The collections of functional expressions $\mathcal{F}^c$ and $\mathcal{F}^c_\vee$ induce distance functions as follows:*

$$d^c(\mathcal{S},\mathcal{S}') = \sup_{f\in\mathcal{F}^c} |f_{\mathcal{S}}(i) - f_{\mathcal{S}'}(i')|$$

We show that each $d^c, c \in (0,1]$ is a metric. In particular, processes at 0 distance are bisimilar.

Theorem 6.2.2 *For each of the families of functional expressions $\mathcal{F}^c$, $\mathcal{F}^c_\vee$, and for every $c \in (0,1]$, d^c is a metric.*

Proof . By definition, d^c is symmetric and the triangle inequality is an easy exercise. The only non-trivial condition to verify is that it gives distance 0 to a pair of processes if and only if they are bisimilar. This is given by Lemma 6.1.6 and Theorem 6.1.10. ∎

We will now prove that the approximations converge in the metric which strengthens the results of Section 5.1 significantly. We are not just saying that the approximations somehow encode the information present in the process being approximated but that they come close in a behavioural sense.

[2]There are also other interesting notions of metric that we do not address here.

Corollary 6.2.3 *If $\mathcal{S}$ involves a finite number of labels, $\mathcal{S}(n, c^n/n)$ converges to $\mathcal{S}$ in the metric d_c, if $c < 1$ with the family of functionals $\mathcal{F}^c$.*

Proof . Assume there is a finite number N of labeis involved in $\mathcal{S}$, that is, for every other label $a \in \mathcal{A}$, $\tau_a(s, S) = 0$ for all $s \in S$. Then for every $n \geq N$, Theorem 6.1.13 will be satisfied for every functional expression of depth $\leq n$, i.e., $|f_S(i) - f_n(p_0)| < c^n$. Now, since $f \in \mathcal{F}^c$, then it is easy to check that for any state s of any process $\mathcal{S}$, $f(s) \leq c^{depth(f)}$. Hence, if $depth(f) \geq n$, we have $0 \leq f \leq c^n$. This implies that $d_c(\mathcal{S}(n, c^n), \mathcal{S}) \leq c^n$. Since $c < 1$, we have that when n increases, $\mathcal{S}(n, c^n/n)$ gets arbitrarily close to $\mathcal{S}$. ∎

The following result shows that the metric behaves as expected with respect to the preorder defined by simulation. More precisely, "in-betweenness" in the order implies in-betweenness in the metric. It is a corollary to Lemma 6.1.5.

Proposition 6.2.4 *If s is simulated by s' which is simulated in turn by s'' then $d^c(s, s') \leq d^c(s, s'')$ and $d^c(s', s'') \leq d^c(s, s'')$.*

We now give a few examples of processes and their distance, and study the family of metrics $\{d^c \mid c \in (0, 1]\}$. These metrics support the spectrum of possibilities of relative weighting of the two factors that contribute to the distance between processes: the length of the functions distinguishing them versus the amount by which each function distinguishes them. d^1 captures only the differences in the probability numbers; probability differences at the first transition are treated on par with probability differences that arise very deep in the evolution of the process. In contrast, d^c for $c < 1$ give more weight to the probability differences that arise earlier in the evolution of the process, i.e. differences identified by simpler functions. As c approaches 0, the future gets discounted more.

As is usual with metrics, the actual numerical values of the metric are less important than the notions of convergence that they engender. Thus, we take the uniformity view of metrics[3], eg. see [Ger85], and will view the metric via properties

[3]Intuitively, a uniformity captures relative distances, eg. x is closer to z than y; it does not tell us what the actual distances are. For example, a uniformity on a metric space M is induced by the collection of all ϵ-balls $\{\{y \mid d(x, y) < \epsilon\} \mid x \in M\}$.

like the significance of zero distance, relative distance of processes and the notion of convergence rather than a detailed justification of the exact numerical values.

Example 6.2.5 *The analysis of example 6.1.3 yields* $d^c(s_0, t_0) = c^2/8$.

Example 6.2.6 *Example 6.1.4 shows the fundamental difference between the metrics* $d^c, c < 1$ *and* d^1. *For* $c < 1$, $d^c(u, v)$ *is witnessed by a functional,* $(\langle a\rangle^n 1)$ *and is given by* $d^c(u, v) = c^n(1 - (1/4)^n)$ *for that* n. *In contrast, for* $c = 1$, *the distance is given by a limit:* $d^1(u, v) = \sup\{1 - (1/4)^n \mid n = 0, 1, \ldots\} = 1$; *no single functional witnesses this.*

Example 6.2.7 *Let* Q *be a process and consider the family of processes* $\{P_\epsilon \mid 0 \leq \epsilon < r\}$ *where* P_ϵ *is the process that makes an* a *with probability* ϵ *and then behaves like* Q. *The functional expression* $(\langle a\rangle 1)$ *evaluates to* $c\epsilon$ *at* P_ϵ. *This functional expression witnesses the distance between any two* P*'s (other functions will give smaller distances). Thus, we get* $d(P_{\epsilon_1}, P_{\epsilon_2}) = c|\epsilon_1 - \epsilon_2|$. *This furthermore ensures that* P_ϵ *converges to* P_r *as* ϵ *tends to* r.

Example 6.2.8 *Consider the processes* P *and* Q *of example 4.1.2. Let us attach probability numbers to these processes. We use* $p_0, p_1, \ldots$ *for the branches of process* P *and* $q_0, q_1, \ldots$ *for the finite ones of process* Q. *We assign probability* q_∞ *to the branch that leads to the state which then has an* a*-labelled transition back to itself. If both processes have the same values on all functional expressions we will show that* $q_\infty = 0$, i.e. *it really cannot be present. The functional expression* $(\langle a\rangle 1)$ *yields* $c(\sum_{i\geq 0} p_i)$ *on* P *and* $c(q_\infty + \sum_{i\geq 0} q_i)$ *on* Q. *The functional expression* $(\langle a\rangle\langle a\rangle 1)$ *yields* $c^2(\sum_{i\geq 1} p_i)$ *on* P *and* $c^2(q_\infty + \sum_{i\geq 1} q_i)$ *on* Q. *Thus, we deduce that* $p_0 = q_0$. *Similarly, considering functional expressions* $(\langle a\rangle\langle a\rangle\langle a\rangle 1)$ *etc, we deduce that* $p_n = q_n$. *Thus,* $q_\infty = 0$.

6.2.1 A countable basis for labelled Markov processes

The space of all labelled Markov processes appears too large to be used in a computational way. In fact this space has a countable subset – the rational trees defined below – which serves to approximate all labelled Markov processes. This is the main result of the present subsection and ultimately it gives a computational handle on the theory.

In the construction of approximations of Section 5.1, the finite processes we construct have the following special structure. The transition graph is a DAG and the states are partitioned into levels. The transitions always go from one level to the next and never go to greater depths. We showed that for every process, there is a sequence of such finite processes that converge to it.

In fact one can just use processes where the transition graph is a tree and with the states partitioned into levels as above. Such finite processes with rational transition probabilities play a special role. We examine their properties below. For brevity we will just say "rational tree" when we mean a finite-state process with a tree-like transition graph and rational transition probabilities.

In the present discussion we need to work with the strict simulation relation rather than plain simulation because the results are not correct with ordinary simulation for technical reasons having to do with strict inequalities providing more "room to maneuver." The notion of strict simulation is inspired by the notion of "way-below" in domain theory. The reader will notice many similarities between the development here and the development in Claire Jones' thesis. The actual mathematical details are somewhat different – her results are domain theoretic and topological rather than measure theoretic.

Lemma 6.2.9 *Let $\mathcal{T}$ be a rational tree that is strictly simulated by a labelled Markov process $\mathcal{S}$. Then there is a finite approximation $\mathcal{S}(n,\epsilon)$ strictly simulating $\mathcal{T}$.*

Proof . Let R_ϵ be the strict simulation between $\mathcal{T} = (T, t_0, \theta)$ and $\mathcal{S}$. Here we use the definition of simulation from 4.3.6. Consider $\mathcal{S}(n, \epsilon/4)$, where n is the height of $\mathcal{T}$. We assume that $\mathcal{T}$ only involves labels $a_1, \ldots, a_n$, but the proof can be adapted easily if it is not the case, because for sure $\mathcal{T}$ only involves a finite number of labels. We first extend R_ϵ to R' in the following way. Let $t \in T$ be at level l, and let $s \in S$. Let R' be the transitive closure of the relation that relates t and s if there is some

$s' \in B$ such that (B, l) is a state of $\mathcal{S}(n, \epsilon/4)$ and $tR_\epsilon s'$. We prove that R' is a strict simulation between $\mathcal{T}$ and $\mathcal{S}$. Observe that R' coincides with R_ϵ on both $\mathcal{T}$ and $\mathcal{S}$. Let $sR't$ and let $Y \subseteq T \cup S$ be an R'-closed set such that $Y \cap S \in \Sigma$. Then it is also R_ϵ-closed and $Y \cap S$ is a union of sets at level $l-1$ of $\mathcal{S}(n, \epsilon/4)$. Hence we have the following:

$$\begin{aligned} \theta_a(t, Y \cap T) &< \tau_a(s', Y \cap S) - \epsilon \\ &\leq \tau_a(s, Y \cap S) + \epsilon/4 - \epsilon \\ &= \tau_a(s, Y \cap S) - 3\epsilon/4 \end{aligned}$$

because s and s' belong to the same set of level l. We have proved that R' is a $3\epsilon/4$-strict simulation between $\mathcal{T}$ and $\mathcal{S}$.

We now define the relation W between $\mathcal{T}$ and $\mathcal{S}(n, \epsilon/4)$. Let W be the transitive closure of the reflexive relation which contains the restriction of R' to $\mathcal{T}$ and relates t and (X, l) if t is at level l in $\mathcal{T}$ and $tR's$ for some $s \in X$. Observe that W coincides with R' on $\mathcal{T}$ and is the identity relation on $\mathcal{S}(n, \epsilon/4)$. Now take two W-related states $t \in T$ and $(X, l) \in \mathcal{S}(n, \epsilon/4)$. Let $Y \subseteq T \cup P$ be W-closed. Let $(B, l-1)$ be the "set" formed by taking the union of sets of the states of $Y \cap \mathcal{S}(n, \epsilon/4)$ restricted to level $l-1$ in $\mathcal{S}(n, \epsilon/4)$. Then B is obviously measurable in $\mathcal{S}$ and $(Y \cap T) \cup B$ is R'-closed. Then if s is the state in X such that $tR's$,

$$\begin{aligned} \theta_a(t, Y \cap T) &< \tau_a(s, B) - 3\epsilon/4 \\ &\leq p_a((X, l), (B, l-1)) + \epsilon/4 - 3\epsilon/4 \\ &< p_a((X, l), Y \cap \mathcal{S}(n, \epsilon/4)) - \epsilon/2 \end{aligned}$$

by Lemma 5.1.2. Thus we are done. ∎

The following theorem shows that rational trees that are strictly simulated by a process form a directed set.

Theorem 6.2.10 *Let $\mathcal{T}$ and $\mathcal{T}'$ be two rational trees that are strictly simulated by a labelled Markov process $\mathcal{S}$. Then there is a rational tree which is strictly simulated by $\mathcal{S}$ and also strictly simulates both $\mathcal{T}$ and $\mathcal{T}'$.*

Proof . Let $\mathcal{T} = (T, t_0, \theta)$ and $\mathcal{T}' = (T', t_0', \theta')$ be strictly simulated by $\mathcal{S}$ by relations R and R'. Then by Lemma 6.2.9, there are $n, n' \in \mathbf{N}$ and $\delta, \delta', \epsilon, \epsilon' > 0$ such that $\mathcal{T}$ is δ-strictly simulated by $\mathcal{S}(n, \epsilon) = (P, p_0, \mathcal{P}(P), \rho)$, and similarly for $\mathcal{T}'$. We choose $\epsilon^* < \delta, \delta'$ such that both ϵ and ϵ' are integral multiples of ϵ^* and let $n^* = \max(n, n')$.

We show that $\mathcal{T}$ and $\mathcal{T}'$ are strictly simulated by $\mathcal{S}(n^*, \epsilon^*)$. Since ϵ is an integral multiple of ϵ^*, the partition at any level l of $\mathcal{S}(n^*, \epsilon^*)$ is a refinement of the partition at level l of $\mathcal{S}(n, \epsilon)$. Let W be the reflexive relation on $T \cup P^*$ which coincides with R on $\mathcal{T}$ and relates t and (X, l) if $tR(B, l)$ for some $(B, l) \in P$ such that $X \subseteq B$. Let $Y \subseteq T \cup P^*$ be W-closed. Then $(Y \cap T) \cup R(Y \cap T)$ is R-closed and $R(Y \cap T)$ is measurable in $\mathcal{S}$ when considered as a set in $\mathcal{S}$. We now prove that $R(Y \cap T)$ is included in $Y \cap P^*$ when they are considered as sets in $\mathcal{S}$. If $(B, l) \in R(Y \cap T)$, then there is some $t \in Y \cap T$ such that $tR(B, l)$ which implies that for all $(X, l) \in \mathcal{S}(n^*, \epsilon^*)$ such that $X \subseteq B$, $(X, l) \in Y \cap P^*$. Since sets of level l in $\mathcal{S}(n^*, \epsilon^*)$ form a refinement of sets of level l in $\mathcal{S}(n, \epsilon)$, we have $B \subseteq Y \cap P^*$ ($Y \cap P^*$ considered as a set in $\mathcal{S}$).

Now let $s \in X \subseteq B$.

$$\begin{aligned}
\theta_a(t, Y \cap T) &< \rho_a((B, l), R(Y \cap T)) - \delta \\
&\leq \tau_a(s, R(Y \cap T)) - \delta \\
&\leq \tau_a(s, Y \cap P^*) - \delta \\
&\leq \rho_a^*((X, l), Y \cap P^*) + \epsilon^* - \delta \\
&< \rho_a^*((X, l), Y \cap P^*),
\end{aligned}$$

as wanted. A similar proof shows that $\mathcal{T}'$ is strictly simulated by $\mathcal{S}(n^*, \epsilon^*)$.

We now construct a rational tree, call it $\mathcal{T}^*$, from $\mathcal{S}(n^*, \epsilon^*)$. The initial state t_0^* is p_0, which is a state at level n in $\mathcal{S}(n^*, \epsilon^*)$. In order to obtain a tree we have to ensure that every state of $\mathcal{T}^*$ has only one incoming transition. For every level strictly below n, in $\mathcal{S}(n^*, \epsilon^*)$, we duplicate the states in such a way that no two transitions arrive in the same state and we take the resulting set T^* to be the set of states of $\mathcal{T}^*$. We define θ^*, the transition function of $\mathcal{T}^*$, by decreasing every probability associated with a transition of $\mathcal{S}(n^*, \epsilon^*)$ to a rational number below it in such a way that $\mathcal{T}^*$ will be strictly simulated by $\mathcal{S}(n^*, \epsilon^*)$ and will still strictly simulate $\mathcal{T}$ and $\mathcal{T}'$. ∎

For every approximation of a process, there exists a monotonic sequence of rational trees that converges to the approximation in the metric d^c.

Lemma 6.2.11 *Given any process of the form $\mathcal{S}(n, \epsilon)$ we can construct a sequence of rational trees $\mathcal{T}_i$ such that $\mathcal{T}_i$ is strictly simulated by $\mathcal{T}_{i+1}$ and all of them are strictly simulated by $\mathcal{S}(n, \epsilon)$ and with* $\lim_{i \to \infty} d^c(\mathcal{T}_i, \mathcal{S}(n, \epsilon)) = 0$.

Proof . By duplicating states of $\mathcal{S}(n, \epsilon)$ as in the previous proof, and keeping only the initial state from level n, we get a tree $\mathcal{T}$ which is bisimilar to $\mathcal{S}(n, \epsilon)$. Now consider a family of trees with the same shape as $\mathcal{T}$ and with the probabilities chosen to be rational and to converge to the probabilities occurring in $\mathcal{T}$. We can always choose these numbers to be strictly increasing. Thus we get immediately that the strict simulation relation holds. It is easy to see that the family of rational trees converge in the metric to $\mathcal{S}(n, \epsilon)$. ∎

The main fact about rational trees is that they form a basis in the sense that every process is the limit of a family of rational trees. This is pleasing because there are only countably many rational trees and it makes the whole space of processes itself into a separable metric space.

Theorem 6.2.12 *For all $c \in (0, 1]$, the metric d^c yields a separable metric space.*

Proof . We show that the rational trees form a countable dense subset. Given any process $\mathcal{S}$ we can consider the countable family of finite approximations $\mathcal{S}(n, 2^{-n})$ to $\mathcal{S}$. For each such finite process we have a countable sequence of rational trees $\{\mathcal{T}_j^{(n)} \mid j, n \in \mathbf{N}\}$, as in lemma 6.2.11. Now, since the rational trees form a directed set with the strict simulation order we can construct a sequence of rational trees as follows. We choose $\mathcal{T}_1$ to be $\mathcal{T}_1^{(1)}$. For $\mathcal{T}_{i+1}$ we compare $\mathcal{T}_{i+1}^{(i+1)}$ with $\mathcal{T}_i$; because we have a directed set we have some rational tree strictly above both, we designate one such to be $\mathcal{T}_{i+1}$. Thus we have a sequence of rational trees ordered by strict simulation and which converge to $\mathcal{S}$ in the metric. ∎

Chapter 7

A categorical definition of bisimulation and simulation

In Section 2.4 we saw that Joyal, Nielsen and Winskel formulated bisimulation for non-probabilistic processes in a categorical setting [JNW96]. In this chapter we study a categorical view of bisimulation and simulation for labelled Markov processes that is based on the same ideas. The original presentation of bisimulation (see [BDEP97]) was given in these terms and the relational view evolved later. The two definitions of bisimulation are equivalent as they are both characterized by the logic $\mathcal{L}$. Using the same argument we will prove that the two definitions of simulation are also the same.

The categorical formulation of bisimulation and simulation is given in the next section. We then give a few examples of how the definition of bisimulation can be checked. In the third section we prove that this view of bisimulation is also characterized by the logic $\mathcal{L}$. We also prove that if two processes $\mathcal{S}$ and $\mathcal{S}'$ are bisimilar (with either definitions of bisimulation) and if every state of $\mathcal{S}$ is bisimilar to a state of $\mathcal{S}'$, then there is a unique zigzag morphism from $\mathcal{S}$ to the quotient of $\mathcal{S}'$ under the logic. In particular, this says that for finite processes, the quotient construction gives the smallest system bisimilar to a given one. The last section is devoted to discrete processes: we give separate proofs of some results that appear simpler in that case and we prove that the categorical definition of simulation is characterized by the logic $\mathcal{L}_\vee$ for finite processes.

7.1 Bisimulation and simulation as spans

For the categorical view, we want to mimic the relation that is used for traditional formulation of bisimulation. As we have seen for nonprobabilistic processes, one can talk about relations by talking about *spans*. Recall that a span in any category between an object S_1 and another object S_2 is a third object T together with morphisms from T to both S_1 and S_2. Given a category of systems, the plan is to say that bisimulation holds between two systems if they are connected by a span of special morphisms. These morphisms should capture bisimulation.

The original idea of Joyal, Nielsen and Winskel was to identify a class of special systems called "observations" or "observable paths" or better still "observable path shapes", and to define the special morphisms as satisfying a kind of path-lifting property, the so-called "open map" property. What one can prove for ordinary labelled transition systems is that if we take paths to be labelled paths in the usual sense, then the open maps are the zigzag morphisms we recalled in Section 2.4.

In the case of labelled Markov processes, the zigzag condition for the morphisms is easy to state and it is easy to see that it corresponds to Larsen-Skou bisimulation in the case of labelled Markov chains. We have introduced the category **LMP** of labelled Markov processes and zigzag morphisms in this category in Section 3.4. We proved that they relate processes that we expect to be bisimilar (with respect to our relational definition of bisimulation).

Following Joyal, Nielsen and Winskel, we define bisimulation as the existence of a span of zigzag morphisms. However, in order to prove that bisimulation is an equivalence relation, we need to go beyond the category **LMP**. One cannot prove transitivity just working with the category **LMP**, we need to introduce another - closely related - category defined below. To prove that bisimulation is a transitive relation, we will use the results of [Eda99]. Unless one is interested in going into all details of the proofs and in reading [Eda99], one can safely skip those parts of the discussion pertaining to what we call *generalized labelled Markov processes.* We use the concept of *universally measurable* sets and functions which are defined in the appendix.

Definition 7.1.1 *A* ***generalized labelled Markov process*** *is a labelled Markov process except that the transition sub-probability function needs only be* universally measurable, *that is, for all $a \in \mathcal{A}$, we have that $\tau_a(\cdot, A)$ is a universally measurable function for $A \in \Sigma$, and for $s \in S$, $\tau_a(s, \cdot)$ is a sub-probability measure on (S, Σ).*

Most of our definitions for labelled Markov processes will be used unchanged and without special comments for generalized labelled Markov processes. For simplicity, we will not define "generalized simulation morphisms" explicitly. Simulation morphisms for generalized labelled Markov processes are defined in exactly the same way as already defined for labelled Markov processes.

Definition 7.1.2 *The objects of the category* ***LMP**** *are generalized labelled Markov processes, having a fixed set $\mathcal{A}$ as the set of labels, with simulations as the morphisms.*

Every labelled Markov process is a generalized labelled Markov process and hence **LMP** is a full subcategory of **LMP***.

We now define bisimulation for labelled Markov processes. We will use the expression "generalized span" between **LMP** objects to mean that we have a span in **LMP*** between the objects. The precise definition is:

Definition 7.1.3 *Let $\mathcal{S}_1$ and $\mathcal{S}_2$ be two labelled Markov processes. $\mathcal{S}_1$ is* ***probabilistically bisimilar to*** *$\mathcal{S}_2$ (written $\mathcal{S}_1 \sim \mathcal{S}_2$) if there is a generalized span of zigzag morphisms between them, i.e. there exists a generalized labelled Markov process $\mathcal{U}$ in* ***LMP**** *and zigzag morphisms f_1 and f_2 such that*

$$\begin{tikzcd} & \mathcal{U} \arrow[dl, "f_1"'] \arrow[dr, "f_2"] & \\ \mathcal{S}_1 & & \mathcal{S}_2 \end{tikzcd}$$

Notice that if there is a zigzag morphism between two systems, they are bisimilar since the identity is a zigzag morphism and because **LMP** is a subcategory of **LMP***. The last fact also implies that if there is a span in **LMP** between two processes, they are bisimilar.

The difficulty in showing that bisimulation is an equivalence is to prove transitivity of the existence of span, since it is obviously reflexive and symmetric. Transitivity relies on the following theorem.

Theorem 7.1.4 ([Eda99]) *Let $\mathcal{S}, \mathcal{S}_1$ and $\mathcal{S}_2$ be three objects of* $\boldsymbol{LMP^*}$ *and f_1 and f_2 be zigzag morphisms from $\mathcal{S}_i$ to $\mathcal{S}$, $i = 1, 2$. Then we can find an object U in* $\boldsymbol{LMP^*}$ *and zigzag morphisms g_1 and g_2 such that the following diagram commutes:*

$$\begin{tikzcd}
 & \mathcal{U} \arrow[dl, dotted, "g_1"'] \arrow[dr, dotted, "g_2"] & \\
\mathcal{S}_1 \arrow[dr, "f_1"] & & \mathcal{S}_2 \arrow[dl, "f_2"'] \\
 & \mathcal{S} &
\end{tikzcd}$$

This is not a pullback because it does not have the universal property, it is not even a weak pullback. We refer to it as the *semi-pullback* construction. The construction heavily relies on properties that are not true for measure spaces in general. In fact, it probably is not true for probabilistic transition systems without some further assumption.

Note that this theorem was proven with zigzag morphisms defined in a slightly different manner as we define them in this thesis. As we recalled after Definition 3.4.1, we now consider systems with initial states, and this allowed us to replace the surjectivity condition in the definition of zigzag morphisms by the condition that the morphisms preserve initial states. It is easy to check that the above theorem is still true in that setting.

There are two important consequences of Theorem 7.1.4.

Corollary 7.1.5 *Bisimulation is an equivalence.*

Corollary 7.1.6 *The categorical definition of bisimulation 7.1.3 and the relational definition 3.5.1 are equivalent.*

Proof . If two processes are bisimilar with respect to definition 7.1.3, then by Proposition 3.5.3 and by the fact that the relational definition of bisimulation is transitive, we have that the two processes are bisimilar with respect to definition 3.5.1. Conversely, if two processes are bisimilar with respect to definition 3.5.1, then by taking the quotient of their direct sum as in Theorem 4.2.9, we can prove that the morphisms that send the states of the processes to the their equivalence classes in the quotient are zigzag morphisms. We then apply Theorem 7.1.4 to conclude that there exists a generalized span between the two processes and hence that they are bisimilar with respect to definition 7.1.3. ∎

In Section 2.4, we saw that for non-probabilistic labelled transition systems, a state s simulates another state s' if and only if there is a simulation morphism that sends s to s'. In the probabilistic case, we cannot say that because then simulation would not compose with bisimulation (and hence would not correspond to the non-categorical definition of simulation). More precisely, if $\mathcal{S}$ is simulated by $\mathcal{T}$ which in turn is bisimilar to $\mathcal{U}$, then it would not follow that $\mathcal{S}$ is simulated by $\mathcal{U}$. Indeed, consider the following picture.

$$\begin{tikzcd}[column sep=small] & s \arrow[dl, "{a[\frac{1}{2}]}"'] \arrow[dr, "{a[\frac{1}{2}]}"] & \\ \bullet & & \bullet \arrow[d, "b"] \\ & & \bullet \end{tikzcd} \longrightarrow \begin{tikzcd} t \arrow[d, "a"] \\ \bullet \arrow[d, "b"] \\ \bullet \end{tikzcd} \longleftarrow \begin{tikzcd}[column sep=small] & u \arrow[dl, "{a[\frac{1}{4}]}"'] \arrow[d, "{a[\frac{1}{4}]}"'] \arrow[dr, "{a[\frac{1}{2}]}"] & \\ \bullet \arrow[d, "b"] & \bullet \arrow[d, "b"] & \bullet \arrow[d, "b"] \\ \bullet & \bullet & \bullet \end{tikzcd}$$

There is a simulation morphism from the first process to the second one sending s to t, and a zigzag morphism from the third process to the second one sending u to t, hence s is simulated by t, and u and t are bisimilar. However, there is no simulation morphism from s to u, though we intuitively see that u should simulate s.

This motivates the need for a span relation in the categorical definition of simulation.

Definition 7.1.7 We say that $\mathcal{S}_1$ is *simulated by* $\mathcal{S}_2$ if there is a span of morphisms

$$\begin{tikzcd} & \mathcal{U} \arrow[dl, "f_1"'] \arrow[dr, "f_2"] & \\ \mathcal{S}_1 & & \mathcal{S}_2 \end{tikzcd}$$

where f_1 is a zigzag morphism and f_2 is a simulation morphism.

This categorical definition relates processes that are related by a simulation relation as defined in Chapter 3.

Proposition 7.1.8 *Let $\mathcal{S}$ and $\mathcal{S}'$ be labelled Markov processes. If $\mathcal{S}$ is simulated by $\mathcal{S}'$ according to Definition 7.1.7, then $\mathcal{S}$ is simulated by $\mathcal{S}'$ according to Definition 3.6.2.*

Proof . The result follows easily from Propositions 3.5.3 and 3.6.7. ∎

We conjecture that the converse is true but we have not proven it yet. However we do have such a proof in the finite case. We will show this in the next section when we will prove that the categorical definition of simulation is characterized by the logic $\mathcal{L}_\vee$ in the case of finite processes. This will allow us to conclude also that the categorical definition of simulation is transitive for finite processes. We strongly believe that it is also transitive for arbitrary processes. The proof of such a result should be of the same flavor as the semi-pullback theorem we use to prove that bisimulation is transitive. In fact, by following the proof and adapting it to the simulation case, we should be able to prove the result.

The categorical definition of simulation is obviously reflexive. To show it is also transitive, we need that if we have two spans connecting $\mathcal{S}_1$ and $\mathcal{S}_3$ then we can complete the square

$$\begin{tikzcd}
& & \mathcal{U} \arrow[dl, dotted, "g_1"', "\text{zigzag}"] \arrow[dr, dotted, "g_2"] & & \\
& \mathcal{S} \arrow[dl, "f_1"', "\text{zigzag}"] \arrow[dr, "f_2"] & & \mathcal{T} \arrow[dl, "f_3"', "\text{zigzag}"] \arrow[dr, "f_4"] & \\
\mathcal{S}_1 & & \mathcal{S}_2 & & \mathcal{S}_3
\end{tikzcd}$$

where the condition that f_3 is zigzag can be lifted through the square to g_1, so that we have a span U between $\mathcal{S}_1$ and $\mathcal{S}_3$ where the morphism $f_1 \circ g_1$ is a zigzag morphism and $f_4 \circ g_2$ is a simulation morphism.

We end this section with a few examples of how the categorical definition of bisimulation can be used between processes. The first example justifies the need for spans of zigzag morphisms and the others illustrate how processes we have seen before are bisimilar with respect to the span definition of bisimulation.

Example 7.1.9 *The first example we give is a discrete one. Consider the two simple discrete labelled Markov processes below*

$$\begin{tikzcd}
 & s_0 \arrow[dl, "{[\frac{1}{2}]}"'] \arrow[dr, "{[\frac{1}{2}]}"] & \\
s_1 & & s_2
\end{tikzcd}
\qquad\qquad
\begin{tikzcd}
 & t_0 \arrow[dl, "{[\frac{1}{3}]}"'] \arrow[d, "{[\frac{1}{3}]}"] \arrow[dr, "{[\frac{1}{3}]}"] & \\
t_1 & t_2 & t_3
\end{tikzcd}$$

It is easy to see that even if, intuitively, these two systems seem to have the same behaviour under interaction, there is no zigzag morphism between them in either direction. Nevertheless, they are bisimilar because there indeed exists a span of zigzag morphism between them. This span is given by a process consisting of 6 transitions from the same state, all with probability $1/6$.

Example 7.1.10 *We recall Example 3.3.3. We let the label set be the one element set* $\{a\}$. *Consider a system* $\mathcal{S} = (S, \Sigma, \tau)$ *with* S, *an arbitrarily complicated state space, and* Σ, *a* σ*-field generated by some analytic space structure on* S. *For example,* S *could be* $\mathbf{R}$, *the reals with the Borel algebra. We define the transition function,* $\tau_a(s, A)$ *in any manner we please subject only to the conditions of the definition of a transition function and to the condition that* $\forall s \in S.\tau_a(s, S) = 1$; *i.e. for every* s, *the distribution* $\tau_a(s, \cdot)$ *is a probability measure. Example 3.3.3 satisfies those conditions. It is easy to see that there is a zigzag morphism from* $\mathcal{S}$ *to the single state system* $\mathcal{O}_a$ *with action* a. *The morphism sends every state to* $o \in O_a$. *Now since* $\{o\}$ *is the only non-trivial measurable set in* $\mathcal{O}_a$ *we only have to check that the* a*-transition in* $\mathcal{O}_a$ *from* o *to itself has probability* $\tau_a(s, S) = 1$ *for any* $s \in S$, *which is given. These two systems are bisimilar.*

Example 7.1.11 *The two systems of Example 3.7.1 are bisimilar since the projection from* $\mathcal{U}$ *to* $\mathcal{S}$ *is easily checked to be a zigzag morphism.*

Example 7.1.12 *We come back to Example 3.7.2. We can verify that the implementation* $\text{cell}_e|\text{cell}_e$ *is bisimilar to its specification* bag_0 *by constructing a zigzag morphism from the former to the latter. The morphism which sends* $\text{cell}_e|\text{cell}_e$ *to* bag_0, $\text{cell}_f|\text{cell}_f$ *to* bag_2 *and the two other states where exactly one cell is full to* bag_1 *is easily proved to be zigzag.*

7.2 Bisimulation and logic

We prove that the categorical view of bisimulation is also characterized by the logic.

The next proposition links zigzag morphisms with formulas in the logic. Recall that **LMP** is a *full* subcategory of **LMP***. Thus whenever we talk about morphisms between labelled Markov processes it makes no difference which category we are talking about. The only place to be careful is when we have an object that is not a labelled Markov process but is a generalized labelled Markov process.

Proposition 7.2.1 *If f is a zigzag morphism from $\mathcal{S}$ in $\boldsymbol{LMP}^*$ to $\mathcal{S}'$ in $\boldsymbol{LMP}$, then for all state $s \in S$ and all formulas $\phi \in \mathcal{L}$,*

$$s \models \phi \iff f(s) \models \phi.$$

Proof . We show that $f^{-1}([\![\phi]\!]_{\mathcal{S}'}) = [\![\phi]\!]_{\mathcal{S}}$ by structural induction on ϕ, which implies the result. Notice that by Proposition 4.2.1, $[\![\phi]\!]_{\mathcal{S}'} \in \Sigma'$ since $\mathcal{S}'$ is in **LMP**, and hence $f^{-1}([\![\phi]\!]_{\mathcal{S}'}) \in \Sigma$, because f is measurable. (This will imply in particular that $[\![\phi]\!]_{\mathcal{S}}$ is measurable in $\mathcal{S}$.) The only nontrivial case corresponds to the modal formula $s \models \langle a\rangle_q\phi$. Observe that by the induction hypothesis and because f is a zigzag morphism, we have

$$\tau_a(s, [\![\phi]\!]_{\mathcal{S}}) = \tau_a(s, f^{-1}([\![\phi]\!]_{\mathcal{S}'})) = \tau_a'(f(s), [\![\phi]\!]_{\mathcal{S}'}).$$

Consequently, s and $f(s)$ satisfy the same formulas of the form $\langle a\rangle_q\phi$, and hence $f^{-1}([\![\langle a\rangle_q\phi]\!]_{\mathcal{S}'}) = [\![\langle a\rangle_q\phi]\!]_{\mathcal{S}}$, as wanted. ∎

From this we get the immediate corollary below, but first we need to say what it means for two systems to satisfy the same formulas. Suppose that (S, i, Σ, τ) and (S', i', Σ', τ') are two systems. We say that they satisfy *all the same formulas* if i and i' satisfy all the same formulas. We write $s \approx s'$ if states s and s' satisfy all the same formulas. Clearly, $\approx$ is an equivalence relation.

Corollary 7.2.2 *If two labelled Markov processes are bisimilar then they satisfy the same formulas of $\mathcal{L}$.*

To show the converse, the general plan is to construct a cospan using logical equivalence and then to use the semi-pullback construction [Eda99] to obtain a span.

Theorem 7.2.3 *Two labelled Markov processes are bisimilar if and only if they obey the same formulas of our logic.*

Proof . One direction has already been shown; what remains is to prove that two systems obeying all the same formulas are bisimilar. Suppose that (S, i, Σ, τ) and (S', i, Σ', τ') satisfy the same formulas. Instead of defining a span of zigzag morphisms directly, we can define a cospan and use the semi-pullback property to infer that $\mathcal{S}$ and $\mathcal{S}'$ are bisimilar. We first construct (T, t_0, Σ_T, j), the direct sum of $\mathcal{S}$ and $\mathcal{S}'$. There are the evident canonical injections ι, ι' which are *not* zigzag because they do not send initial states to initial states. We know from Theorem 4.2.4 that the quotient system $(T/_\approx, t, \Sigma_\approx, h)$ is a **LMP** and that the canonical projection $g_\approx$ from $\mathcal{T}$ to $T/_\approx$ is a zigzag morphism. Here we need to make a small change to the quotient. We will set its initial state to be the equivalence class containing i and i' instead of the initial state of $\mathcal{T}$. Then $g_\approx$ still satisfies the zigzag property even if it does not send the initial state to the initial state. Thus we have the diagram of figure 7.1. The composites

$$\begin{tikzcd}
S \arrow[dr, "\iota"] \arrow[ddr, "f"'] & & S' \arrow[dl, "\iota'"'] \arrow[ddl, "f'"] \\
& T \arrow[d, "g_\approx"'] & \\
& T/_\approx &
\end{tikzcd}$$

Figure 7.1: Constructing a cospan.

$g_\approx \circ \iota$ and $g_\approx \circ \iota'$ are measurable, henceforth we call them f and f' respectively. It remains to prove the zigzag property for f and f'. So take a set B in $\Sigma_\approx$ and $s \in S$. Then

$$\begin{aligned}
\rho_a(f(s), B)) &= j_a(\iota(s), g_\approx^{-1}(B)) \quad \text{by Theorem 4.2.4} \\
&= \tau_a(s, g_\approx^{-1}(B) \cap S) \quad \text{by definition of } j_a \text{ and because } s \in S
\end{aligned}$$

$$= \tau_a(s, f^{-1}(B))$$

This proves that f and similarly f' are zigzag morphisms. Thus we have defined a cospan of zigzag morphisms and using the semi-pullback theorem there is a corresponding span, hence $\mathcal{S}$ and $\mathcal{S}'$ are bisimilar. ∎

Definition 7.2.4 *Given two labelled Markov processes $\mathcal{S}$ and $\mathcal{S}'$ that are bisimilar, we say that two states $s \in S$ and $s' \in S'$ are **bisimilar**, denoted $s \sim s'$, if there is a generalized span $f : \mathcal{U} \dashrightarrow \mathcal{S}, g : \mathcal{U} \dashrightarrow \mathcal{S}'$ such that for some $u \in U$ we have $f(u) = s$ and $g(u) = s'$.*

It follows from Corollary 7.1.5 that $\sim$ is an equivalence relation and we have a corollary to Theorem 7.2.3:

Corollary 7.2.5 *Let $\mathcal{S}$ and $\mathcal{S}'$ be two labelled Markov processes that are bisimilar and let $s \in S$ and $s' \in S'$, then $s \sim s'$ if and only if $s \approx s'$.*

Note that in the semi-pullback construction, the states of $\mathcal{U}$ are exactly the pairs $(s_1, s_2) \in \mathcal{S}_1 \times \mathcal{S}_2$ that have the same image. Hence, we have the following corollary.

Corollary 7.2.6 *Let $\mathcal{S}_1$ and $\mathcal{S}_2$ be two labelled Markov processes, and let $\mathcal{U}$ be defined as in the semi-pullback construction in the following diagram.*

$$\begin{tikzcd} & \mathcal{U} \arrow[dl, "\pi_1"'] \arrow[dr, "\pi_2"] & \\ \mathcal{S}_1 \arrow[dr, "f_1"] & & \mathcal{S}_2 \arrow[dl, "f_2"'] \\ & (\mathcal{S}_1 \uplus \mathcal{S}_2)/_{\approx} & \end{tikzcd}$$

Then for $s_i \in S_i$, $i = 1, 2$, $s_1 \sim s_2 \iff \exists u \in U$ such that $\pi_i(u) = s_i$, $i = 1, 2$.

The quotient construction has the following couniversal property. In the case of finite state systems this says that the quotienting construction gives the minimal finite state system bisimilar to the given one.

Proposition 7.2.7 *Let $\mathcal{S}$ and $\mathcal{T}$ be two labelled Markov processes. If $\mathcal{S} \sim \mathcal{T}$, and every state of $\mathcal{S}$ is bisimilar to a state of $\mathcal{T}$, then there exists a unique zigzag morphism r from $\mathcal{S}$ to $\mathcal{T}/_{\approx}$.*

Proof . If $(S, i, \Sigma, \tau) \sim (T, \Sigma_T, l)$, there is a span (U, Σ_U, j) with zigzag morphisms $f : \mathcal{U} \to \mathcal{S}$ and $g : \mathcal{U} \to \mathcal{T}$. Assume that $\mathcal{U}$ relates every pair of states of $S \times T$ that are bisimilar. This is possible by Corollary 7.2.6. Let $s \in S$. Since f, g and $h_{\approx} : \mathcal{T} \to \mathcal{T}/_{\approx}$ are zigzag morphisms, for every formula ϕ we have,

$$\begin{aligned} s \models \phi &\iff \forall u \in f^{-1}(s).u \models \phi \\ &\iff \forall u \in f^{-1}(s).g(u) \models \phi \\ &\iff \forall u \in f^{-1}(s).h_{\approx}g(u) \models \phi \end{aligned}$$

This implies that all $u \in f^{-1}(s)$ are mapped by $h_{\approx}g$ to the same state $t \in T/_{\approx}$ and that we can set $r(s) = t$. This makes the diagram commute. To see r is Borel measurable, let $A \in \Sigma_{\approx}$. Then $r^{-1}(A) = f(g^{-1}h_{\approx}^{-1}A)$, $B_1 := g^{-1}h_{\approx}^{-1}A$ is obviously Borel in U and so is $B_2 := g^{-1}h_{\approx}^{-1}A^c$. Now we have not only that B_1 and B_2 are disjoint, but their images under f are also disjoint. To see this, suppose the contrary. Then there exist $u_i \in B_i$ such that $fu_1 = fu_2$; but since the diagram commutes, it implies that $h_{\approx}g(u_1) = h_{\approx}g(u_2)$, which is a contradiction to the definition of the B_i's. Thus we have that fB_1 and fB_2 are disjoint analytic sets of $\mathcal{S}$; since analytic sets are separable by Borel sets and since $fB_1 \cup fB_2 = S$, fB_1 and fB_2 must be Borel sets of $\mathcal{S}$, concluding the proof that r is Borel measurable. We now show that r is zigzag; let $s \in S, A \in \Sigma_{\approx}$ and $u \in f^{-1}(s)$. Then

$$\begin{aligned} \rho_a(r(s), A) &= l_a(g(u), h_{\approx}^{-1}A) \\ &= j_a(u, g^{-1}h_{\approx}^{-1}A) \\ &= j_a(u, f^{-1}r^{-1}A) \\ &= \tau_a(s, r^{-1}A) \end{aligned}$$

Finally, r is unique because every state s is mapped in $T/_{\approx}$ to the only state that satisfies the same formulas as it does (since in $T/_{\approx}$ there is no pair of distinct states satisfying the same formulas). ∎

7.3 Discrete processes revisited

We collect here results that we have for discrete or finite processes in the categorical setting. In previous sections, we have proven facts about general labelled Markov processes. Some of these results required quite complicated proofs that can be obtained more easily if we restrict ourselves to discrete processes. It is the case for the logical characterization of bisimulation for which we give an independent proof in the case of discrete processes. This proof turns out to also give us transitivity of bisimulation for free, without refering to the semi-pullback construction which we needed in the general case in order to prove that bisimulation is a transitive relation.

We proved that if a state is simulated by another state, then all the formulas from the logic $\mathcal{L}_\vee$ it satisfies are also satisfied by the other state. We did not, as yet, succeed in proving the converse for arbitrary processes, but we do have such a proof for finite processes.

We begin this section with a proof that the categorical definition of bisimulation for general labelled Markov processes is indeed a generalization of the definition of Larsen & Skou in [LS91], that is, the two definitions agree on discrete processes. Hence we reconsider discrete systems (Markov chains) from the point of view of the bisimulation notion that we have defined for labelled Markov processes and show that the Larsen-Skou definition coincides with the "span of zigzags" definition.

First, we have to say what it means for two Markov chains to be Larsen-Skou bisimilar, since the Larsen-Skou definition involves states of a single process rather than states of two different processes, and so does not apply without an appropriate interpretation. We say that two Markov chains are Larsen-Skou bisimilar (written $\sim_{LS}$) if and only if, in their direct sum, their initial states are Larsen-Skou bisimilar.

Proposition 7.3.1 *Let $\mathcal{P} = (P, i, \pi)$ and $\mathcal{P}' = (P', i', \pi')$ be two labelled Markov chains. $\mathcal{P} \sim_{LS} \mathcal{P}'$ if and only if there exists a span of zigzag morphisms f and f' between them:*

$$\begin{tikzcd} & \mathcal{U} \arrow[dl, "f"'] \arrow[dr, "f'"] & \\ \mathcal{P} & & \mathcal{P}' \end{tikzcd}$$

Proof . $\Leftarrow$: We first show that if $\mathcal{P} \xrightarrow{f} \mathcal{P}'$, where f is a zigzag morphism, then $\mathcal{P} \sim_{LS} \mathcal{P}'$. Let $\mathcal{U} = (U, u_0, \rho)$ be the direct sum of $\mathcal{P}$ and $\mathcal{P}'$. Now f defines the following equivalence relation, R, on U:

$$uRv \iff (u = v) \vee (f(u) = v) \vee (f(u) = f(v));$$

R depends on which sets u and v are in. The equivalence classes are of the form $\{p'\} \cup f^{-1}(p')$ for each $p' \in P'$; thus each equivalence class can be represented uniquely by an element of P'. Let $a \in \mathcal{A}, u, v \in U$ such that uRv, and choose any $p' \in P'$ that represents an equivalence class, namely $\{p'\} \cup f^{-1}(p')$. We want to show that $\rho_a(u, -)$ and $\rho_a(v, -)$ agree on $\{p'\} \cup f^{-1}(p')$.

First assume $u \in P$ and $v \in P'$, meaning that $f(u) = v$. Then

$$\begin{aligned} \rho_a(u, \{p'\} \cup f^{-1}(p')) &= \pi_a(u, f^{-1}(p')) \text{ since } u \in P \\ &= \pi'_a(f(u), p') \text{ since } f \text{ is a zigzag morphism} \\ &= \rho_a(v, \{p'\} \cup f^{-1}(p')), \end{aligned}$$

which is precisely the condition for Larsen-Skou bisimulation. Now if u and v are both in P (and still R-related), they have the same image, so $u\,R\,f(u)$ and $f(u) = f(v)$ and $f(v)\,R\,v$ and we can simply apply the above calculation. Finally we have the trivial case where u and v are both in P'. They are then equal and we are done since, f being a surjective function, every state of P is R-equivalent to a state of P' and vice versa. Since we know that Larsen-Skou bisimulation is an equivalence relation it follows that whenever we have a span of zigzags connecting two labelled Markov chains they are Larsen-Skou bisimilar.

$\Rightarrow$: Assume $\mathcal{P} \sim_{LS} \mathcal{P}'$, and denote by $\equiv$ the probabilistic bisimulation over $\mathcal{U} = (U, u_0, \rho)$, the disjoint union of $\mathcal{P}$ and $\mathcal{P}'$. We need to construct a span of zigzag morphisms

$$\begin{tikzcd} & \mathcal{U} \arrow[dl, "f"'] \arrow[dr, "f'"] & \\ \mathcal{P} & & \mathcal{P}' \end{tikzcd}$$

To do this, let $\mathcal{T} = (T, \theta)$ where $T = \{(p, p') \in P \times P' : p \equiv p' \text{ in } \mathcal{U}\}$ and where the transition function is given by, for $a \in \mathcal{A}$,

$$\theta_a((p,p'),(q,q')) = \frac{\pi_a(p,q)\,\pi'_a(p',q')}{\rho_a(p,[q]_\equiv)}$$

where $[q]_\equiv$ denotes the equivalence class containing q in U. Since $p \equiv p'$ and $q \equiv q'$, we have by definition of $\equiv$ that $\rho_a(p,[q]_\equiv) = \rho_a(p',[q]_\equiv) = \rho_a(p',[q']_\equiv)$.

To prove that $\mathcal{T}$ is a labelled Markov chain, we need that for any $(p,p') \in T$,

$$\sum_{(q,q')\in T} \theta_a((p,p'),(q,q')) \leq 1.$$

This will follow from the proof that we have zigzag morphisms from $\mathcal{T}$ to $\mathcal{P}$ and $\mathcal{P}'$. As morphisms $f : \mathcal{T} \to \mathcal{P}$ and $f' : \mathcal{T} \to \mathcal{P}'$, we simply take the left and right projections which send initial states to initial states by definition of Larsen-Skou probabilistic bisimulation. We prove that they are zigzag morphisms. First note that

$$\forall q \in P.f^{-1}(q) = \{q\} \times (P' \cap [q]_\equiv).$$

For any $a \in \mathcal{A}$, $(p,p') \in T, q \in P$, we have

$$\begin{aligned}
\theta_a((p,p'),f^{-1}(q)) &= \sum_{q'\in P'\cap[q]_\equiv} \theta_a((p,p'),(q,q')) \\
&= \sum_{q'\in P'\cap[q]_\equiv} \frac{\pi_a(p,q)\,\pi'_a(p',q')}{\rho_a(p,[q]_\equiv)} \\
&= \frac{\pi_a(p,q)}{\rho_a(p',[q]_\equiv)} \sum_{q'\in P'\cap[q]_\equiv} \pi'_a(p',q') \\
&= \frac{\pi_a(p,q)}{\pi'_a(p',[q]_\equiv \cap P')}\pi_a(p,[q]_\equiv \cap P) \\
&= \pi_a(p,q) = \pi_a(f(p,p'),q).
\end{aligned}$$

and f is thus a zigzag morphism. The same argument applies to f'. ∎

The result of Section 7.2 gives for arbitrary labelled Markov processes a characterization of bisimulation in terms of the logic $\mathcal{L}$ which does not contain negation and is unexpectedly weak. The proof uses machinery that is unconventional in concurrency theory. We show here that for discrete systems we can prove a completely constructive version of the characterization result.

The following theorem is a special case of the general theorem but is proved without the powerful tools invoked for the general theorem. Of course it already follows from the general theorem but it is of interest to see what types of arguments are needed to prove the purely finite state case. In the proof below a result from measure theory *is* used but one does not need all the machinery that is needed for the general theorem. This gives some indication that one is unlikely to have thought of the finite-state version using only ideas from concurrency theory; it is unlikely that one would have thought of theorem 3.2 of Billingsley's book [Bil95] on purely combinatorial grounds.

Theorem 7.3.2 *Two finite Markov chains satisfying the same formulas of $\mathcal{L}$ are bisimilar.*

Proof. Let $\mathcal{P} = (P, i, \pi)$ and $\mathcal{P}' = (P', i', \pi')$ be two Markov chains satisfying all the same formulas. We will show that we can construct a new Markov chain $\mathcal{U}$ and zigzag morphisms $f : \mathcal{U} \longrightarrow \mathcal{P}$ and $f' : \mathcal{U} \longrightarrow \mathcal{P}'$.

Let $U = \{(p, p') \in P \times P' \mid p \approx p'\}$, the initial state u_0 be $(i, i) \in U$ and let $f : \mathcal{U} \to \mathcal{P}$ and $f' : \mathcal{U} \to \mathcal{P}'$ be the projection maps. We define $\rho_a : U \times U \to [0, 1]$ as follows:

$$\rho_a((p, p'), (q, q')) = \frac{\pi_a(p, q)\pi'_a(p', q')}{\pi_a(p, [q]_{\mathcal{P}})},$$

where $[q]_{\mathcal{P}}$ is the $\approx$-equivalence class in $\mathcal{P}$ containing q.

Claim: *for all $a \in \mathcal{A}$, for all $(p, p'), (q, q') \in U$, we have $\pi_a(p, [q]_{\mathcal{P}}) = \pi'_a(p', [q']_{\mathcal{P}'})$.*

To prove the claim, let $\mathcal{T} = (T, t_0, j_a)$ be the *direct sum* of $\mathcal{P}$ and $\mathcal{P}'$. Of course we have that for all formulas ϕ, $\pi_a(p, [\![\phi]\!]_{\mathcal{P}}) = \pi'_a(p', [\![\phi]\!]_{\mathcal{P}'})$, which implies that $j_a(p, [\![\phi]\!]_{\mathcal{T}}) = j_a(p', [\![\phi]\!]_{\mathcal{T}})$. Now we know that the set $\mathcal{F}$ of formulas is closed under intersections, so it is a π-system as defined by Billingsley in [Bil95]. Let $\mathcal{C}$ be the class of subsets C of T satisfying

$$j_a(p, C) = j_a(p', C).$$

Then $\mathcal{C}$ contains $\mathcal{F}$ and is a λ-system, that is

1. $T \in \mathcal{C}$;

2. $C \in \mathcal{C}$ implies $C^c \in \mathcal{C}$; and

3. $C_1, C_2, \ldots, \in \mathcal{C}$ and $C_n \cap C_m = \emptyset$ for $n \neq m$ imply $\cup_n C_n \in \mathcal{C}$.

So by Theorem 3.2 of [Bil95], $\mathcal{C}$ contains the σ-field generated by $\mathcal{F}$. This means that, since the equivalence classes $[q]_\mathcal{T}$ are in this σ-field, we have

$$\pi_a(p, [q]_\mathcal{P}) = j_a(p, [q]_\mathcal{T}) = j_a(p', [q']_\mathcal{T}) = \pi'_a(p', [q']_{\mathcal{P}'})$$

and the claim is proved.

It remains to prove that the projections are zigzag morphisms. So let $a \in \mathcal{A}$, $(p, p') \in U$, and $q' \in P'$. We have

$$\begin{aligned}
\rho_a((p,p'), f'^{-1}(q')) &= \rho_a((p,p'), [q']_\mathcal{P} \times \{q'\}) \\
&= \sum_{q \in [q']_\mathcal{P}} \frac{\pi_a(p,q)\pi'_a(p',q')}{\pi_a(p,[q]_\mathcal{P})} \\
&= \sum_{q \in [q']_\mathcal{P}} \frac{\pi_a(p,q)\pi'_a(p',q')}{\pi'_a(p',[q']_{\mathcal{P}'})} \\
&= \frac{\pi'_a(p',q')}{\pi'_a(p',[q']_{\mathcal{P}'})} \sum_{q \in [q']_\mathcal{P}} \pi_a(p,q) \\
&= \frac{\pi'_a(p',q')}{\pi'_a(p',[q']_{\mathcal{P}'})} \pi_a(p,[q']_\mathcal{P}) \\
&= \pi'_a(p',q').
\end{aligned}$$

To show that f is a zigzag morphism is even more straightforward. ∎

Corollary 7.3.3 *Probabilistic bisimulation defined as a span between labelled Markov chains is an equivalence relation.*

In fact, for discrete processes, it is not hard to prove this result directly with a proof that is independent of the logic and also from the semi-pullback theorem.

We now turn to simulation. We have the same correspondence between the categorical definition of simulation and the logic $\mathcal{L}_\vee$ as we had for the relational definition.

Theorem 7.3.4 *In finite processes, a state simulates another state if and only if it logically simulates it using the logic $\mathcal{L}_\vee$.*

Proof . The first direction is given by Proposition 7.1.8 and Theorem 4.3.2. For the other direction, let $\mathcal{P} = (P, i, \pi)$ and $\mathcal{Q} = (Q, i', \tau)$ be such that i is logically simulated by i'. We want to construct a span $\mathcal{U}, f, g$ where $f : \mathcal{U} \longrightarrow \mathcal{P}$ is zigzag and $g : \mathcal{U} \longrightarrow \mathcal{Q}$ is a simulation morphism. Let $U = \{(p, q) : p \text{ is logically simulated by } q\}$, $u_0 = (i, i')$. Let f and g be respectively the left and right projections; they send initial states to initial states. The transition sub-probability function of $\mathcal{U}$ will be given by $\rho_a((p, q), (p', q')) = w(p', q')$, where $w : P \times Q$ is defined as follows.

The plan is to use the max-flow min-cut theorem by Ford and Fulkerson to define w, that will be given the value of the flow (see [Bol79] for a proof of this theorem). Thus we construct a network having a source and a sink. The nodes of the network are states $P \cup Q$. There are two additional nodes, the source p^* and the sink q^*. There is an arrow from p^* to $p' \in P$ having capacity c if $\pi_a(p, p') = c$. Similarly, there is an arrow from $q' \in Q$ to q^* having capacity c if $\tau_a(q, q') = c$. Finally, there are arrows of capacity one between $p' \in P$ and $q' \in Q$ if p' is logically simulated by q'. The network is illustrated in the following picture.

$$\begin{tikzcd}
 & p' \arrow[drr, "1"] & & \vdots \arrow[dr] & \\
p^* \arrow[ur, "\pi_a(p,p')"] \arrow[dr] & \vdots & & q' \arrow[r, "\tau_a(q,q')"] & q^* \\
 & \bullet & & \bullet \arrow[ur] &
\end{tikzcd}$$

The max-flow min-cut theorem says that the maximal flow through the network is equal to the minimal cut. We show that $\{p^*\}$ is a minimal cut. Recall that C is a cut if $p^* \in C$ and $q^* \notin C$; the weight of the cut is the total capacity of the arrows coming out of the nodes that are in C. If a cut C of our network involves an edge of capacity one –that is, there is some $p' \in P$ logically simulated by some $q' \in Q$ such that $p' \in C$ and $q' \notin C$, then $\{p^*\}$ is a cut of smaller or equal weight, since it is of weight ≤ 1. So we can assume that if $C \neq \{p^*\}$ is a minimal cut, then if $p' \in C \cap P$, then $W(p') \cap Q \subseteq C$, where W is the relation induced by the logic.

Now let C be a minimal cut not containing an edge of capacity one and $B = C \cap P$. Then $W(B) \cap Q$ is included in C. Then the weight of C is $\pi_a(p, P \setminus B) + \tau_a(q, W(B))$.

Now from the proof of Theorem 4.3.2, we know that $\pi_a(p, B) \leq \tau_a(s, W(B) \cap Q)$ and hence the cut $\{p^*\}$ is of smaller than or equal weight to C.

Observe that the maximum flow w through the network satisfies the following properties:

$$1.\ \forall p' \in P, \sum_{p'Wq'} w(p', q') = \pi_a(p, p'),$$

because $\{p^*\}$ is a minimal cut, and hence the capacity of the edge $[p^*, p']$ is fully used; and

$$2.\ \forall q' \in Q, \sum_{p'Wq'} w(p', q') \leq \tau_a(q, q'),$$

because w is a flow and hence the flow that get in the node q' cannot exceed the capacity of the edge out of q'.

We now prove that f is zigzag. Let $(p, q) \in U$ and $A \subseteq P$. Then 1. above implies $\rho_a((p,q), f^{-1}(A)) = \sum_{p' \in A} \sum_{p'Wq'} w(p', q') = \sum_{p' \in A} \pi_a(p, p') = \pi_a(p, A)$ and hence f is zigzag. Now g is a simulation morphism, because if $A \subseteq Q$, then $\rho_a((p,q), g^{-1}(A)) = \sum_{q' \in A} \sum_{p'Wq'} w(p', q') \leq \sum_{q' \in A} \tau_a(q, q') = \tau_a(q, A)$. ∎

Corollary 7.3.5 *For finite processes, the categorical definition of simulation is transitive and corresponds to the relational definition of simulation.*

Chapter 8

Conclusions

8.1 Summary

The main point of this thesis is that one can use the same logical principles developed for discrete probabilistic processes to reason about continuous-state Markov processes. The fundamental results are

- a notion of bisimulation equivalence and simulation preorder,
- a logic for characterizing bisimulation and simulation,
- an approximation scheme and
- a metric on the collection of processes.

The bisimulation relation actually closely corresponds to the classical notion of "lumpability" in queuing theory [KS60]. We proved that bisimulation is characterized by a very simple logic that neither involves negation nor infinite conjunction. The logical characterization suggests that if one can model a system in terms of labelled Markov processes, one may obtain more information than by modeling the system with non-probabilistic processes. Moreover we have two algorithms, one that can decide whether two finite-state probabilistic processes are bisimilar, the other that can decide whether a state simulates another. We show how to approximate any continuous process with finite-state processes, and that one can reconstruct the process from its approximations. These approximations can be as close as we want to the original process. Indeed, we define a family of metrics that witness the fact that the approximations converge to the original process. Finally, we proved that the space of labelled Markov processes is a separable metric space.

8.2 Related work

There has been a significant growth of activity in the general area of probabilistic systems with several papers on equivalences, simulation and testing [vGSST90, JS90, LS91, JL91, CSZ92, SL94, JY95], on model checking [HK96] and reasoning about average behaviour [dA98].

The fundamental work on probabilistic processes is by Larsen and Skou [LS91]. They study bisimulation as well as testing, in the context of discrete processes. We extend their results in the sense that we work with continuous state-spaces, but also because the logic that we prove to characterize bisimulation does not involve negation. Moreover, in their characterization of bisimulation, they restricted to not only discrete systems, but to finitely branching systems. The extension of the results to continuous state-spaces required new techniques and new types of arguments.

In [SL94], Segala and Lynch also proposed definitions of simulation, bisimulation and their weak versions, for discrete probabilistic systems. Their strong bisimulation is the same as our bisimulation and their strong simulation slightly stronger than the simulation relation we defined. We have pointed out the difference in Section 3.6. However, they consider non-deterministic probabilistic systems, that is, they allow different transitions from a single state with the same label. Their definition of simulation in our notation for non-deterministic total discrete processes can be formulated as follows (here we remove the additional condition that makes their definition stronger than ours). A relation R is a simulation if whenever sRt, then for all $a \in \mathcal{A}$, either $\tau_a(s,S) = 0$ or there exists a weight function $\delta : S \times S \longrightarrow [0,1]$ such that

$$\forall x \in S \sum_{y \in S} \delta(x,y) = \tau_a(s,x) \quad \text{and} \quad \forall y \in S \sum_{x \in S} \delta(x,y) = \tau_a(t,y).$$

We believe that our formulation of simulation can be extended to a generalization of labelled Markov processes to non-deterministic continuous processes in such a way that it would naturally coincide with strong simulation of Segala and Lynch in the discrete case. Our formulation of simulation is in our point of view more natural and intuitive and it seems that the logic is more easily handled with it. However, there is an advantage to using the definition in terms of the existence of a weight function. This is seen, for example, in the algorithm developed by Baier in [Bai96] to decide simulation between finite processes in polynomial time. The existence of the weight function is determined with the use of the well-known polynomial-time algorithm to compute the maximum flow in a network. This way, the condition that our simulation must satisfy for every closed subset of the state-space is checked in one step. However, Baier's algorithm does not produce a witnessing formula when a state is not simulated by another state. We do not see how that algorithm can be modified to construct a formula. It is likely that, since we cannot use negation in the logic as for bisimulation, we really need nested sets to witness formulas.

Norman [Nor97] has notions of preorders for non-deterministic discrete probabilistic processes that are defined through the use of tests. All these preorders are weaker than our notion of simulation, for as he points out, they give rise to equivalences on processes that are weaker than Larsen and Skou's bisimulation.

We have already mentioned that one can take a coalgebraic view of bisimulation [AM89, Rut95, Rut96] as well. One can view a labelled Markov process as a coalgebra of a suitable functor; this functor was introduced by Giry [Gir81] in order to define a monad on **Mes** analogous to the powerset monad. From this point of view, bisimulation is a span of coalgebra homomorphisms. In fact, these coalgebra homomorphisms are precisely our zigzag morphisms in **LMP**, and hence their bisimulation is the same as ours. The work of de Vink and Rutten [dVR97] is the only one we know that studies systems with continuous state-spaces. However, they work with ultrametric spaces which are not as general as our analytic spaces. While interesting, their work is not likely to adapt easily to real-life examples. For example, the reals do not form an ultrametric space. Nevertheless the coalgebraic view could perhaps give useful insights into logic.

In [Hil94], Hillston made interesting progress toward applications. She developed a compositional approach to performance evaluation through a process algebra. The setting is not the same as ours since she works with discrete Markov chains with temporal delay which capture the notion of continuous time. There is a rate associated to each type of action and indeterminacy between transitions is resolved by races between events executing at different rates. She also defines a bisimulation based on matching the transition rates instead of the transition probabilities. Other areas of application where probabilistic systems are important are telecommunication [AJKvO97], real-time systems [BFGL95] and modeling physical systems [GSS95].

Kozen developed a probabilistic dynamic logic [Koz85]. He discovered a Stone-type duality in the context of probabilistic semantics and pioneered the use of Markov kernels. In particular, he made the key suggestion that measurable functions should be thought of as the analogue of formulas – an idea which we use in our work on metrics.

In timed-automata theory [AD94], a notion of bisimulation appears in the so-called region construction for quotienting the state space, which is continuous, into finite pieces in particular situations. The basic framework is ordinary automata theory to which we add real-time clocks, hence modeling some kind of continuity. The same kind of idea is used in linear hybrid automata. The restrictions on the models – essentially linearity and limitations on the ability to compare clocks – are what guarantee that there are finitely many regions. We also are interested in reducing continuous systems to finite-states systems whenever possible. However, instead of limiting the description of systems, we work with a notion of approximation. Thus while general systems cannot be collapsed to finite-state systems, they can be approximated arbitrarily closely by finite-state systems. How this could be used in practice remains to be seen.

We have used the work of Joyal, Nielsen and Winskel [JNW96] for giving categorical formulations of bisimulation and simulation. In their work they have a notion of $\mathcal{P}$-open morphisms as a general notion of functional bisimulation. They define bisimulation as a span of $\mathcal{P}$-open morphisms. We have not succeeded in expressing our zigzag morphisms as $\mathcal{P}$-open morphisms in an appropriate sense. They show that for ordinary labelled transition systems, $\mathcal{P}$-open morphisms correspond to non-probabilistic zigzag morphisms and it is this analogy which guided us in the early stages of this work. In their examples, they use the existence of pullbacks to show that bisimulation is transitive. In our case, the corresponding pullbacks do not exist and we had to use Edalat's semi-pullback construction instead.

The metric that we have defined bears a formal resemblance to the Hutchinson metric [Hut81]. The resemblance consists of the fact that the metric is defined as

$$d(\mu,\nu) := \sup_{f \in F} |\int f d\mu - \int f d\nu|$$

for an appropriate class of functions. For the Hutchinson metric, F is the class of Lipschitz functions whereas for us it is the class of functions defined by our functional expressions, i.e., the class $\mathcal{F}^c$. The fundamental difference between these classes is the functional $\langle a \rangle f$ that reflects the fact that we are working with transition systems and provides us with a way of reasoning about transitions.

Our algorithms for bisimulation and simulation have been inspired by an algorithm due to Cleaveland [Cle90] to decide bisimilarity of non-probabilistic processes. Since the logic considered in this paper contains negation, the algorithm is based on partitioning the state-space. We used instead nested families of subsets that correspond to the modal formula. These algorithms are oriented towards finding a formula that witnesses non-bisimilarity or non-similarity of states. This is why the complexity is worse than the one obtained from the algorithms of Baier [Bai96] mentioned above which can decide bisimilarity and similarity in polynomial time. However we have discovered recently that by using $\mathcal{L}_\neg$, we can slightly modify our algorithm for bisimulation and obtain a polynomial time algorithm that constructs a distinguishing formula when two processes are not bisimilar. This is done by partitioning the state-space as in Cleaveland's algorithm. Of course, this cannot be applied to simulation since negation cannot be part of a logic characterizing simulation.

8.3 Future work

There are two directions to pursue: the extension of the basic theory and the development of applications.

Extension of the basic theory

We proved that two processes are bisimilar if and only if they satisfy the same formulas of the logic; we also have a similar result for simulation between processes. A process simulates another one if it satisfies all the formula of $\mathcal{L}_\vee$ that the other satisfies. The result holds for discrete systems or even pairs of systems one of which is discrete and the other is not. An obvious goal is to extend the work to cover the case of two continuous state space systems.

The results of Chapter 6 suggest that there ought to be a domain-theoretic interpretation of the space of labelled Markov processes. We proved that for every process $\mathcal{S}$, there is an increasing chain of rational trees (with respect to the simulation preorder) that converge in the metric to $\mathcal{S}$. We would like that $\mathcal{S}$ is indeed the least upper bound of this chain. Note that proving this result is equivalent to showing that simulation is characterized by the logic for arbitrary labelled Markov processes. If we can prove these results and that we have a complete partial order (cpo), we would then have an algebraic cpo. Linking these results with Jones-Plotkin's powerdomain theory [JP89, Jon90] is an interesting subject of investigation. The Plotkin powerdomain is a functor that constructs a domain from another domain. Jones and Plotkin have defined a probabilistic analogue of this functor [JP89]. We can write equations like $D \simeq \mathcal{A} \longrightarrow \mathcal{J}(D)$ where $\mathcal{J}$ is the Jones-Plotkin powerdomain functor. An interesting question is whether our category **LMP** is this domain D where rational trees are exactly finite elements of D.

Another very interesting subject of investigation is the extension of the theory to continuous time. In this case we cannot talk about transitions as steps. We must adapt the formalism, maybe by using the notion of transition rates [BHK99, KNSS99].

When one combines two systems, one must be able to work with the composite system as if it was a single one. In particular, if the two systems communicate by synchronizing on an action, the observer should not notice this move because it is not associated with any interaction with the environment. Such a move should be silent even if it happens. All the observer can notice is that there may be different actions enabled because of that silent move. Our notion of bisimulation does not distinguish silent actions and hence treat them on par with other actions. Every transition that a process performs must be matched exactly by a bisimilar process. Roughly speaking, a *weak* version of bisimulation would not require silent actions to be matched.

In order to use any of our results for verification, we need to define a calculus or programming language for describing continuous probabilistic systems. This involves inventing syntax for the language which would be a non-trivial effort. We have to somehow specify physical systems in a syntactic way. A process algebra is defined for finite probabilistic processes in [DGJP99a]. In [GJP99], a concurrent constraint language was defined for the description and programming of concurrent probabilistic systems. They use recursion to encode continuous distributions and show how many distributions can be expressed.

Once we have a syntax we can define contexts. A context is an expression of the language containing variables. For example, in Milner's calculus for communicating processes [Mil80] (called CCS), we write $C[X] := a.X$ for the expression representing the process that can do an a-action and then behaves like X (the dot preceded by a label is an operator of CCS and represents prefixing). Now suppose two systems A, B are equivalent (whatever the equivalence may be) and we place them in a common context $C[X]$, we can ask if $C[A]$ and $C[B]$ are equivalent. If it is always the case, we call the equivalence a congruence. Without a language to define contexts, this makes no sense. This type of question is fundamental in process algebra. In [DGJP99b], a process algebra is defined for discrete probabilistic processes and it is proved that bisimulation is a congruence with respect to the operations of this process algebra. Moreover, it is shown that process combinators do not increase distance in any of the metrics we have defined.

Development of applications

For a continuous process, a finite bisimulation quotient may exist, hence existing model checking [BCGH$^+$97] techniques can be used. Unlike in timed automata or hybrid systems, we do not provide conditions that guarantee that there is a finite quotient.

Perhaps the approximations can be used for approximate verification. This is under investigation. Unfortunately it seems unlikely that logical approximation results would hold for richer logics. Thus approximate reasoning would take a different form from the verification formalisms we see used now.

We gave an algorithm that decides if a finite-state process simulates another finite-state process, and if it does not, it constructs a formula that witnesses this fact. These results open the way to using simulation in verification and other applications rather than bisimulation. Exactly how this might be used is one of the issues we are exploring.

Appendix A

Relevant mathematical concepts

For completeness, we give the relevant definitions from measure theory and probability theory in this appendix. The reader can find more complete explanations in the following references: "Probability and Measure" by Billingsley [Bil95], "Real Analysis and Probability" by Ash [Ash72], the book with the same title by Dudley [Dud89] and "Introduction to Measure and Probability" by Kingman and Taylor [KT66].

Definition A.1 *A* **σ-field** *on a set S is a family of subsets of S which includes S itself and which is closed under complementation and countable unions. A* **measurable space** *is a pair (S, Σ) where S is a set and Σ is a σ-field on S.*

We use the expression "measurable sets" for members of the σ-field. Given a topological space $(S, \mathcal{T})$, we can define the σ-field, often written $\mathcal{B}$, generated by the open sets (or, equivalently, by the closed sets). This is usually called the *Borel σ-field.*

Definition A.2 *Given a measurable space (S, Σ), a* **subprobability measure** *on S is a $[0,1]$-valued set function, μ, defined on Σ such that*

- $\mu(\emptyset) = 0$,
- *if $\{A_i | i \in \mathbf{N}\}$ is a pairwise disjoint collection of sets in Σ, then $\mu(\bigcup_{i\in\mathbf{N}} A_i) = \sum_{i\in\mathbf{N}} \mu(A_i)$.*

μ is a probability measure if $\mu(S) = 1$. A probability space (S, Σ, μ) is a measurable space equipped with a probability measure.

Definition A.3 *A set A in a measurable space (S, Σ) is said to be* ***universally measurable*** *if for every finite measure μ there exist B and C in Σ such that $B \subseteq A \subseteq C$ and $\mu(B) = \mu(C)$.*

It is easy to check that universally measurable sets form a σ-field.

Definition A.4 *A function $f : (S, \Sigma) \longrightarrow (S', \Sigma')$ between measurable spaces is said to be* **measurable** *if for all $B' \in \Sigma'$, $f^{-1}(B') \in \Sigma$. A measurable function is called* ***simple*** *if its range is finite.*

Theorem A.5 *The supremum of any countable family of simple functions is a measurable function.*

In probability theory, a measurable function from a probability space to a measure space is called a random variable and is typically written with capital letters X, Y. If X is a random variable on a probability space $(\Omega, \mathcal{F}, P)$ to the measurable space (S, Σ) and $A \in \Sigma$, we write $P(X \in A)$ to mean $P(X^{-1}(A))$.

Definition A.6 *A function $f : (S, \Sigma) \longrightarrow (S', \Sigma')$ between measurable spaces is said to be* ***universally measurable*** *if for all $B' \in \Sigma'$, $f^{-1}(B')$ is universally measurable.*

The next several definiticns and results pertain to analytic spaces.

Definition A.7 *A* ***Polish*** *space is the topological space underlying a complete, separable metric space; i.e. it has a countable dense subset.*

Definition A.8 *An* ***analytic*** *set is the image of a Polish space under a continuous function from one Polish space to another. An analytic space is an analytic set with the topology induced by the Polish space that contains it.*

Analytic sets do not form a σ-field. In fact, if an analytic set has a complement which is also analytic, then it is a Borel set, and its complement is Borel as well.

Theorem A.9 *Analytic sets are universally measurable.*

We use the following result to infer that labelled Markov chains all have the powerset as σ-field.

Proposition A.10 *In an analytic space, singletons are measurable.*

The following proposition [Dud89] gives equivalent definitions of analytic set.

Proposition A.11 *Suppose that S and S' are Polish spaces and f is a function from S to S'. Then A is an analytic set if and only if A is the image of B under f, where f is either measurable or continuous and B is either the whole space S or a Borel subset of it.*

Analytic spaces are more general than Polish spaces but they also have the basic property that regular conditional probability distributions can be defined on them. These regular conditional probability distributions are the basic building blocks for proving transitivity of bisimulation in the categorical view of the theory – see Chapter 7. The sub-probability transition functions that need to be constructed in Edalat's work [Eda99] are based on regular conditional probability distributions. However, we abstract from them in this thesis.

Bibliography

[ACH+95] R. Alur, C. Courcoubetis, N. Halbwachs, T.A. Henzinger, P.-H. Ho, X. Nicollin, A. Olivero, J. Sifakis, and S. Yovine. The algorithmic analysis of hybrid systems. *Theoretical Computer Science*, 138:3–34, 1995.

[AD94] R. Alur and D. Dill. A theory of timed automata. *Theoretical Computer Science*, 126:183–235, 1994.

[AHS96] R. Alur, T. Henzinger, and E. Sontag, editors. *Hybrid Systems III*, number 1066 in Lecture Notes in Computer Science. Springer-Verlag, 1996.

[AJKvO97] R. Alur, L. Jagadeesan, J. J. Kott, and J. E. von Olnhausen. Model-checking of real-time systems: A telecommunications application. In *Proceedings of the 19th International Conference on Software Engineering*, 1997.

[AKNS97] P. Antsaklis, W. Kohn, A. Nerode, and S. Sastry, editors. *Hybrid Systems IV*, volume 1273 of *Lecture Notes In Computer Science*. Springer-Verlag, 1997.

[AM89] P. Aczel and N. Mendler. A final-coalgebra theorem. In *Category Theory and Computer Science*, Lecture Notes In Computer Science, pages 357–365, 1989.

[Arn94] A. Arnold. *Finite Transition Systems*. Prentice-Hall, 1994.

[Arv76] W. Arveson. *An Invitation to C^*-Algebra*. Springer-Verlag, 1976.

[Ash72] R. B. Ash. *Real Analysis and Probability*. Academic Press, 1972.

[Bai96] C. Baier. Polynomial time algorithms for testing probabilistic bisimulation and simulation. In *Proceedings of the 8th International Conference on Computer Aided Verification (CAV'96)*, number 1102 in Lecture Notes in Computer Science, pages 38–49, 1996.

[BCGH$^+$97] C. Baier, E. Clarke, V. Garmhausen-Hartonas, M. Kwiatkowska, and M. Ryan. Symbolic model checking for probabilistic processes. In *ICALP'97*, volume 1256 of *Lecture notes in computer science*, 1997.

[BDEP97] R. Blute, J. Desharnais, A. Edalat, and P. Panangaden. Bisimulation for labelled Markov processes. In *Proceedings of the Twelfth IEEE Symposium On Logic In Computer Science, Warsaw, Poland*, 1997.

[BFGL95] A. Benveniste, E. Fabre, P. Le Guernic, and B. C. Levy. A calculus of stochastic systems for the specification, simulation and hidden state estimation of mixed stochastic/nonstochastic systems. *Theoretical Computer Science*, 152(2):171–217, 1995.

[BHK99] C. Baier, H. Hermanns, and J.-P. Katoen. Approximative symbolic model checking of continuous-time markov chains. In *Proceedings of CONCUR 99*, Lecture Notes In Computer Science. Springer-Verlag, 1999.

[Bil95] P. Billingsley. *Probability and Measure*. Wiley-Interscience, 1995.

[Bol79] B. Bollobás. *Graph theory, an introductory course*. Springer-Verlag, 1979.

[BW90] M. Barr and C. Wells. *Category Theory for Computing Science*. prentice-Hall, 1990.

[Cle90] R. Cleaveland. On automatically explaining bisimulation inequivalence. In E.M. Clarke and R.P. Kurshan, editors, *Computer-Aided Verification CAV 90*, number 531 in Lecture Notes in Computer Science, pages 364–372, 1990.

[CSZ92] R. Cleaveland, S. Smolka, and A. Zwarico. Testing preorders for probabilistic processes. In *Proceedings of the International Colloquium On Automata Languages And Programming 1992*, number 623 in Lecture Notes In Computer Science. Springer-Verlag, 1992.

[CW96] E. M. Clarke and J. M. Wing. Formal methods: state of the art and future directions. *ACM computing surveys*, 28A(4):626–643, 1996.

[dA98] L. de Alfaro. How to specify and verify the long-run average behavior of probabilistic systems. In *Proceedings of the 13th IEEE Symposium On Logic In Computer Science, Indianapolis*, pages 454–465. IEEE Press, June 1998.

[DEP98] J. Desharnais, A. Edalat, and P. Panangaden. A logical characterization of bisimulation for labeled Markov processes. In *Proceedings of the 13th IEEE Symposium On Logic In Computer Science, Indianapolis*, pages 478–489. IEEE Press, June 1998.

[DEP99] J. Desharnais, A. Edalat, and P. Panangaden. Bisimulation for labeled Markov processes. *Information and Computation*, 1999.

[DGJP99a] J. Desharnais, V. Gupta, R. Jagadeesan, and P. Panangaden. Approximating continuous Markov processes. Submitted for publication. Available from `www.sable.mcgill.ca/~prakash`, 1999.

[DGJP99b] J. Desharnais, V. Gupta, R. Jagadeesan, and P. Panangaden. Metrics for labeled Markov processes. In *Proceedings of CONCUR99*, Lecture Notes in Computer Science. Springer-Verlag, 1999.

[Dud89] R. M. Dudley. *Real Analysis and Probability.* Wadsworth and Brookes/Cole, 1989.

[dVR97] E. de Vink and J. J. M. M. Rutten. Bisimulation for probabilistic transition systems: A coalgebraic approach. In *Proceedings of the 24th International Colloquium On Automata Languages And Programming*, 1997.

[Eda99] A. Edalat. Semi-pullbacks and bisimulation in categories of Markov processes. *Mathematical Structures in Computer Science*, 1999.

[Ger85] Robert Geroch. *Mathematical Physics.* Chicago Lectures in Physics. University of Chicago Press, 1985.

[Gir81] M. Giry. A categorical approach to probability theory. In B. Banaschewski, editor, *Categorical Aspects of Topology and Analysis*, number 915 in Lecture Notes In Mathematics, pages 68–85. Springer-Verlag, 1981.

[GJP99] V. Gupta, R. Jagadeesan, and P. Panangaden. Stochastic processes as concurrent constraint programs. In *Proceedings of the 26th Proceedings Of The Annual ACM Symposium On Principles Of Programming Languages*, 1999.

[GSS95] V. Gupta, V. Saraswat, and P. Struss. A model of a photocopier paper path. In *Proceedings of the 2nd IJCAI Workshop on Engineering Problems for Qualitative Reasoning*, 1995.

[Hal74] P. Halmos. *Measure Theory.* Number 18 in Graduate Texts in Mathematics. Springer-Verlag, 1974. Originally published in 1950.

[HHWT95] T. Henzinger, P.-H. Ho, and H. Wong-Toi. Hytech: the next generation. In *Proceedings of the 16th Annual Real-time Systems Symposium*, pages 55–65. IEEE Computer Society Press, 1995.

[Hil94] J. Hillston. *A Compositional Approach to Performance Modelling.* PhD thesis, University of Edinburgh, 1994. To be published as a Distinguished Dissertation by Cambridge University Press.

[HK96] M. Huth and M. Kwiatkowska. On probabilistic model checking. Technical Report CSR-96-15, University of Birmingham, 1996. Available from http://www.cs.bham.ac.uk/ mzk/.

[HM85] M. Hennessy and R. Milner. Algebraic laws for nondeterminism and concurrency. *Journal of the ACM*, 32(1):137–162, 1985.

[Hut81] J. E. Hutchinson. Fractals and self-similarity. *Indiana Univ. Math. J.*, 30:713–747, 1981.

[JL91] B. Jonsson and K. Larsen. Specification and refinement of probabilistic processes. In *Proceedings of the 6th Annual IEEE Symposium On Logic In Computer Science*, 1991.

[JNW96] A. Joyal, M. Nielsen, and G. Winskel. Bisimulation from open maps. *Information and Computation*, 127(2):164–185, 1996.

[Jon90] C. Jones. *Probabilistic Non-determinism.* PhD thesis, University of Edinburgh, 1990. CST-63-90.

[JP89] C. Jones and G. D. Plotkin. A probabilistic powerdomain of evaluations. In *Proceedings of the Fourth Annual IEEE Symposium On Logic In Computer Science*, pages 186–195, 1989.

[JS90] C.-C. Jou and S. A. Smolka. Equivalences, congruences, and complete axiomatizations for probabilistic processes. In J.C.M. Baeten and J.W. Klop, editors, *CONCUR 90 First International Conference on Concurrency Theory*, number 458 in Lecture Notes In Computer Science. Springer-Verlag, 1990.

[JY95] B. Jonsson and W. Yi. Compositional testing preorders for probabilistic processes. In *Proceedings of the 10th Annual IEEE Symposium On Logic In Computer Science*, pages 431–441, 1995.

[KNSS99] M. Kwiatkowska, G. Norman, R. Segala, and J. Sproston. Automatic verification of real-time systems with discrete probability distributions. In *Proceedings of ARTS99*, Lecture Notes in Computer Science. Springer-Verlag, 1999.

[Koz85] D. Kozen. A probabilistic PDL. *Journal of Computer and Systems Sciences*, 30(2):162–178, 1985.

[KS60] Kemeny and Snell. *Finite Markov Chains.* van Nostrand, Princeton, New Jersey, 1960.

[KT66] J. F. C. Kingman and S. J. Taylor. *Introduction to Measure and Probability.* Cambridge University Press, 1966.

[Low96] G. Lowe. Breaking and fixing the needham-schroeder public-key protocol using fdr. In *Tools and algorithms for the construction and analysis of systems*, Lecture Notes in Computer Science. Springer-Verlag, 1996.

[LS91] K. G. Larsen and A. Skou. Bisimulation through probablistic testing. *Information and Computation*, 94:1–28, 1991.

[Mac71] Saunders Mac Lane. *Categories for the Working Mathematician*, volume 5 of *Graduate texts in Mathematics.* Springer-Verlag, New York, 1971.

[Mil80] R. Milner. *A Calculus for Communicating Systems*, volume 92 of *Lecture Notes in Computer Science.* Springer-Verlag, 1980.

[Mil89] R. Milner. *Communication and Concurrency.* Prentice-Hall, 1989.

[Mil90] R. Milner. *Handbook of Theoretical Computer Science: Volume B*, chapter Operational and Algebraic Senmantics of Concurrent Processes, pages 1201–1242. MIT Press, 1990.

[Nor97] Gethin Norman. *Metric Semantics for Reactive Probabilistic Processes.* PhD thesis, University of Birmingham, 1997. Technical Report CRS-98-03.

[NS78] R. Needham and M. Schroeder. Using encryption for authentication in large networks of computers. *Communications of the ACM*, 21:393–399, 1978.

[Par81] D. Park. Concurrency and automata on infinite sequences. In *Proceedings of the Fifth GI Conference*, number 154 in Lecture Notes in Computer Science, pages 561–572. Springer-Verlag, 1981.

[Rud66] W. Rudin. *Real and Complex Analysis.* McGraw-Hill, 1966.

[Rut95] J. J. M. M. Rutten. A calculus of transition systems (towards universal coalgebra). In A. Ponse, M. de Rijke, and Y. Venema, editors, *Modal Logic and Process Algebra, a bisimulation perspective*, number 53 in CSLI Lecture Notes, 1995. Available electronically from `www.cwi.nl/~janr`.

[Rut96] J. J. M. M. Rutten. Universal coalgebra: a theory of systems. Technical Report CS-R9652, CWI AMsterdam, 1996. Available from URL `www.cwi.nl/~janr/papers/`.

[SL94] R. Segala and N. Lynch. Probabilistic simulations for probabilistic processes. In B. Jonsson and J. Parrow, editors, *Proceedings of CONCUR94*, number 836 in Lecture Notes In Computer Science, pages 481–496. Springer-Verlag, 1994.

[vGSST90] R. van Glabbeek, S. Smolka, B. Steffen, and C. Tofts. Reactive generative and stratified models for probabilistic processes. In *Proceedings of the 5th Annual IEEE Symposium On Logic In Computer Science*, 1990.